

# StarPU Internal Handbook

---

for StarPU 1.4.8

---

<b>1 Introduction</b>	<b>3</b>
1.1 Motivation	3
<b>2 StarPU Core</b>	<b>5</b>
2.1 StarPU Core Entities	5
<b>3 Module Index</b>	<b>11</b>
3.1 Modules	11
<b>4 Module Documentation</b>	<b>13</b>
4.1 Workers	13
<b>5 File Index</b>	<b>15</b>
5.1 File List	15
<b>6 StarPU File Documentation</b>	<b>19</b>
6.1 barrier.h File Reference	19
6.2 barrier_counter.h File Reference	19
6.3 bound.h File Reference	20
6.4 cg.h File Reference	21
6.5 coherency.h File Reference	23
6.6 combined_workers.h File Reference	31
6.7 config.h File Reference	31
6.8 copy_driver.h File Reference	38
6.9 data_concurrency.h File Reference	40
6.10 data_interface.h File Reference	40
6.11 data_request.h File Reference	41
6.12 datastats.h File Reference	42
6.13 datawizard.h File Reference	43
6.14 debug.h File Reference	43
6.15 detect_combined_workers.h File Reference	45
6.16 disk.h File Reference	45
6.17 disk_unistd_global.h File Reference	47
6.18 driver_common.h File Reference	48
6.19 driver_cpu.h File Reference	49
6.20 driver_cuda.h File Reference	49
6.21 driver_disk.h File Reference	50
6.22 driver_mpi_common.h File Reference	50
6.23 driver_mpi_sink.h File Reference	51
6.24 driver_mpi_source.h File Reference	51
6.25 driver_opencl.h File Reference	52
6.26 driver_opencl_utils.h File Reference	53
6.27 drivers.h File Reference	53
6.28 errorcheck.h File Reference	53

---

6.29	<a href="#">fifo_queues.h File Reference</a>	54
6.30	<a href="#">filters.h File Reference</a>	55
6.31	<a href="#">footprint.h File Reference</a>	55
6.32	<a href="#">fxt.h File Reference</a>	56
6.33	<a href="#">graph.h File Reference</a>	62
6.34	<a href="#">helper_mct.h File Reference</a>	64
6.35	<a href="#">idle_hook.h File Reference</a>	65
6.36	<a href="#">implicit_data_deps.h File Reference</a>	65
6.37	<a href="#">jobs.h File Reference</a>	65
6.38	<a href="#">knobs.h File Reference</a>	69
6.39	<a href="#">malloc.h File Reference</a>	73
6.40	<a href="#">memalloc.h File Reference</a>	74
6.41	<a href="#">memory_manager.h File Reference</a>	74
6.42	<a href="#">memory_nodes.h File Reference</a>	74
6.43	<a href="#">memstats.h File Reference</a>	77
6.44	<a href="#">mp_common.h File Reference</a>	78
6.45	<a href="#">multiple_regression.h File Reference</a>	78
6.46	<a href="#">node_ops.h File Reference</a>	78
6.47	<a href="#">openmp_runtime_support.h File Reference</a>	80
6.48	<a href="#">perfmmodel.h File Reference</a>	84
6.49	<a href="#">prio_deque.h File Reference</a>	86
6.50	<a href="#">prio_list.h File Reference</a>	86
6.51	<a href="#">profiling.h File Reference</a>	87
6.52	<a href="#">progress_hook.h File Reference</a>	88
6.53	<a href="#">rbtree.h File Reference</a>	88
6.54	<a href="#">rbtree_i.h File Reference</a>	92
6.55	<a href="#">regression.h File Reference</a>	95
6.56	<a href="#">rwlock.h File Reference</a>	95
6.57	<a href="#">sched_component.h File Reference</a>	97
6.58	<a href="#">sched_ctx.h File Reference</a>	97
6.59	<a href="#">sched_ctx_list.h File Reference</a>	101
6.60	<a href="#">sched_policy.h File Reference</a>	103
6.61	<a href="#">simgrid.h File Reference</a>	105
6.62	<a href="#">sink_common.h File Reference</a>	107
6.63	<a href="#">sort_data_handles.h File Reference</a>	107
6.64	<a href="#">source_common.h File Reference</a>	107
6.65	<a href="#">starpu_parallel_worker_create.h File Reference</a>	107
6.66	<a href="#">starpu_data_cpy.h File Reference</a>	108
6.67	<a href="#">starpu_debug_helpers.h File Reference</a>	108
6.68	<a href="#">starpu_fxt.h File Reference</a>	109
6.69	<a href="#">starpu_spinlock.h File Reference</a>	109
6.70	<a href="#">starpu_task_insert_utils.h File Reference</a>	109

---

6.71 tags.h File Reference . . . . .	110
6.72 task.h File Reference . . . . .	111
6.73 task_bundle.h File Reference . . . . .	114
6.74 thread.h File Reference . . . . .	116
6.75 timing.h File Reference . . . . .	116
6.76 topology.h File Reference . . . . .	117
6.77 utils.h File Reference . . . . .	121
6.78 uthash.h File Reference . . . . .	123
6.79 write_back.h File Reference . . . . .	125
<b>7 StarPU MPI File Documentation</b>	<b>127</b>
7.1 starpu_mpi_cache.h File Reference . . . . .	127
7.2 starpu_mpi_driver.h File Reference . . . . .	127
7.3 starpu_mpi_init.h File Reference . . . . .	127
7.4 starpu_mpi_nmad_backend.h File Reference . . . . .	128
7.5 starpu_mpi_stats.h File Reference . . . . .	128
7.6 starpu_mpi_cache_stats.h File Reference . . . . .	129
7.7 starpu_mpi_early_data.h File Reference . . . . .	129
7.8 starpu_mpi_mpi.h File Reference . . . . .	130
7.9 starpu_mpi_nmad_unknown_datatype.h File Reference . . . . .	130
7.10 starpu_mpi_sync_data.h File Reference . . . . .	131
7.11 starpu_mpi_comm.h File Reference . . . . .	131
7.12 starpu_mpi_early_request.h File Reference . . . . .	131
7.13 starpu_mpi_mpi_backend.h File Reference . . . . .	132
7.14 starpu_mpi_private.h File Reference . . . . .	133
7.15 starpu_mpi_tag.h File Reference . . . . .	138
7.16 starpu_mpi_datatype.h File Reference . . . . .	138
7.17 starpu_mpi_fxt.h File Reference . . . . .	138
7.18 starpu_mpi_nmad.h File Reference . . . . .	140
7.19 starpu_mpi_select_node.h File Reference . . . . .	140
7.20 starpu_mpi_task_insert.h File Reference . . . . .	140
7.21 load_balancer_policy.h File Reference . . . . .	141
7.22 load_data_interface.h File Reference . . . . .	141
7.23 data_movements_interface.h File Reference . . . . .	142
<b>8 StarPU Resource Manager File Documentation</b>	<b>143</b>
8.1 starpurm_private.h File Reference . . . . .	143

This manual documents the usage of StarPU version 1.4.8. Its contents was last updated on 2025-06-16.

Copyright © 2009-2025 University of Bordeaux, CNRS (LaBRI UMR 5800), Inria

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# Chapter 1

## Introduction

### 1.1 Motivation



# Chapter 2

## StarPU Core

### 2.1 StarPU Core Entities

TODO

#### 2.1.1 Overview

Execution entities:

- **worker**: A worker (see [Workers](#), [Workers and Scheduling Contexts](#)) entity is a CPU thread created by StarPU to manage one computing unit. The computing unit can be a local CPU core, an accelerator or GPU device, or — on the master side when running in master-slave distributed mode — a remote slave computing node. It is responsible for querying scheduling policies for tasks to execute.
- **sched\_context**: A scheduling context (see [Scheduling Contexts](#), [Workers and Scheduling Contexts](#)) is a logical set of workers governed by an instance of a scheduling policy. It defines the computing units to which the scheduling policy instance may assign work entities.
- **driver**: A driver is the set of hardware-dependent routines used by a worker to initialize its associated computing unit, execute work entities on it, and finalize the computing unit usage at the end of the session.

Work entities:

- **task**: A task is a high level work request submitted to StarPU by the application, or internally by StarPU itself.
- **job**: A job is a low level view of a work request. It is not exposed to the application. A job structure may be shared among several task structures in the case of a parallel task.

Data entities:

- **data handle**: A data handle is a high-level, application opaque object designating a piece of data currently registered to the StarPU data management layer. Internally, it is a `_starpu_data_state` structure.
- **data replicate**: A data replicate is a low-level object designating one copy of a piece of data registered to StarPU as a data handle, residing in one memory node managed by StarPU. It is not exposed to the application.

#### 2.1.2 Workers

A **worker** is a CPU thread created by StarPU. Its role is to manage one computing unit. This computing unit can be a local CPU core, in which case, the worker thread manages the actual CPU core to which it is assigned; or it can be a computing device such as a GPU or an accelerator (or even a remote computing node when StarPU is running in distributed master-slave mode.) When a worker manages a computing device, the CPU core to which the worker's thread is by default exclusively assigned to the device management work and does not participate to computation.

### 2.1.2.1 States

#### Scheduling operations related state

While a worker is conducting a scheduling operations, e.g. the worker is in the process of selecting a new task to execute, flag `state_sched_op_pending` is set to `!0`, otherwise it is set to `0`.

While `state_sched_op_pending` is `!0`, the following exhaustive list of operations on that workers are restricted in the stated way:

- adding the worker to a context is not allowed;
- removing the worker from a context is not allowed;
- adding the worker to a parallel task team is not allowed;
- removing the worker from a parallel task team is not allowed;
- querying state information about the worker is only allowed while `state_relax_refcnt > 0`;
  - in particular, querying whether the worker is blocked on a parallel team entry is only allowed while `state_relax_refcnt > 0`.

Entering and leaving the `state_sched_op_pending` state is done through calls to `_starpw_worker_enter↔_sched_op()` and `_starpw_worker_leave_sched_op()` respectively (see these functions in use in functions `_starpw_get_worker_task()` and `_starpw_get_multi_worker_task()`). These calls ensure that any pending conflicting operation deferred while the worker was in the `state_sched_op_pending` state is performed in an orderly manner.

#### Scheduling contexts related states

Flag `state_changing_ctx_notice` is set to `!0` when a thread is about to add the worker to a scheduling context or remove it from a scheduling context, and is currently waiting for a safe window to do so, until the targeted worker is not in a scheduling operation or parallel task operation anymore. This flag set to `!0` will also prevent the targeted worker to attempt a fresh scheduling operation or parallel task operation to avoid starving conditions. However, a scheduling operation that was already in progress before the notice is allowed to complete.

Flag `state_changing_ctx_waiting` is set to `!0` when a scheduling context worker addition or removal involving the targeted worker is about to occur and the worker is currently performing a scheduling operation to tell the targeted worker that the initiator thread is waiting for the scheduling operation to complete and should be woken up upon completion.

#### Relaxed synchronization related states

Any StarPU worker may participate to scheduling operations, and in this process, may be forced to observe state information from other workers. A StarPU worker thread may therefore be observed by any thread, even other StarPU workers. Since workers may observe each other in any order, it is not possible to rely exclusively on the `sched_mutex` of each worker to protect the observation of worker state flags by other workers, because worker A observing worker B would involve locking workers in (A B) sequence, while worker B observing worker A would involve locking workers in (B A) sequence, leading to lock inversion deadlocks.

In consequence, no thread must hold more than one worker's `sched_mutex` at any time. Instead, workers implement a relaxed locking scheme based on the `state_relax_refcnt` counter, itself protected by the worker's `sched↔_mutex`. When `state_relax_refcnt`

`0`, the targeted worker state flags may be observed, otherwise the thread attempting

the observation must repeatedly wait on the targeted worker's `sched_cond` condition until `state_relax_refcnt > 0`.

The relaxed mode, while on, can actually be seen as a transactional consistency model, where concurrent accesses are authorized and potential conflicts are resolved after the fact. When the relaxed mode is off, the consistency model becomes a mutual exclusion model, where the `sched_mutex` of the worker must be held in order to access or change the worker state.

#### Parallel tasks related states

When a worker is scheduled to participate to the execution of a parallel task, it must wait for the whole team of workers participating to the execution of this task to be ready. While the worker waits for its teammates, it is not available to run other tasks or perform other operations. Such a waiting operation can therefore not start while conflicting operations such as scheduling operations and scheduling context resizing involving the worker are on-going. Conversely these operations and other may query whether the worker is blocked on a parallel task entry with `starpw_worker_is_blocked_in_parallel()`. The `starpw_worker_is_blocked_in_parallel()` function is allowed to proceed while and only while `state_relax_refcnt > 0`. Due to the relaxed worker locking scheme, the `state_blocked_in_parallel` flag of the targeted worker may change after it has been observed by an observer thread. In consequence, flag `state_blocked_in_parallel_observed` of the targeted worker is set to 1 by the observer immediately after the observation to "taint" the targeted worker. The targeted worker will clear the `state_blocked_in_parallel_observed` flag tainting and defer the processing of parallel task related requests until a full scheduling operation shot completes without the `state_blocked_in_parallel_observed` flag being tainted again. The purpose of this tainting flag is to prevent parallel task operations to be started immediately after the observation of a transient scheduling state. Worker's management of parallel tasks is governed by the following set of state flags and counters:

- `state_blocked_in_parallel`: set to !0 while the worker is currently blocked on a parallel task;
- `state_blocked_in_parallel_observed`: set to !0 to taint the worker when a thread has observed the `state_blocked_in_parallel` flag of this worker while its `state_relax_refcnt` state counter was >0. Any pending request to add or remove the worker from a parallel task team will be deferred until a whole scheduling operation shot completes without being tainted again.
- `state_block_in_parallel_req`: set to !0 when a thread is waiting on a request for the worker to be added to a parallel task team. Must be protected by the worker's `sched_mutex`.
- `state_block_in_parallel_ack`: set to !0 by the worker when acknowledging a request for being added to a parallel task team. Must be protected by the worker's `sched_mutex`.
- `state_unblock_in_parallel_req`: set to !0 when a thread is waiting on a request for the worker to be removed from a parallel task team. Must be protected by the worker's `sched_mutex`.
- `state_unblock_in_parallel_ack`: set to !0 by the worker when acknowledging a request for being removed from a parallel task team. Must be protected by the worker's `sched_mutex`.
- `block_in_parallel_ref_count`: counts the number of consecutive pending requests to enter parallel task teams. Only the first of a train of requests for entering parallel task teams triggers the transition of the `state_block_in_parallel_req` flag from 0 to 1. Only the last of a train of requests to leave a parallel task team triggers the transition of flag `state_unblock_in_parallel_req` from 0 to 1. Must be protected by the worker's `sched_mutex`.

### 2.1.2.2 Operations

#### Entry point

All the operations of a worker are handled in an iterative fashion, either by the application code on a thread launched by the application, or automatically by StarPU on a device-dependent CPU thread launched by StarPU. Whether a worker's operation cycle is managed automatically or not is controlled per session by the field `not_launched_drivers` of the `starpu_conf` struct, and is decided in `_starpu_launch_drivers` function.

When managed automatically, cycles of operations for a worker are handled by the corresponding driver specific `starpu<DRV>_worker()` function, where DRV is a driver name such as `cpu` (`_starpu_cpu_worker`) or `cuda` (`_starpu_cuda_worker`), for instance. Otherwise, the application must supply a thread which will repeatedly call `starpu_driver_run_once()` for the corresponding worker.

In both cases, control is then transferred to `_starpu_cpu_driver_run_once` (or the corresponding driver specific func). The cycle of operations typically includes, at least, the following operations:

- **task scheduling**
- **parallel task team build-up**
- **task input processing**
- **data transfer processing**
- **task execution**

When the worker cycles are handled by StarPU automatically, the iterative operation processing ends when the running field of `_starpu_config` becomes false. This field should not be read directly, instead it should be read through the `_starpu_machine_is_running()` function.

#### **Task scheduling**

If the worker does not yet have a queued task, it calls `_starpu_get_worker_task()` to try and obtain a task. This may involve scheduling operations such as stealing a queued but not yet executed task from another worker. The operation may not necessarily succeed if no tasks are ready and/or suitable to run on the worker's computing unit.

#### **Parallel task team build-up**

If the worker has a task ready to run and the corresponding job has a size `>1`, then the task is a parallel job and the worker must synchronize with the other workers participating to the parallel execution of the job to assign a unique rank for each worker. The synchronization is done through the job's `sync_mutex` mutex.

#### **Task input processing**

Before the task can be executed, its input data must be made available on a memory node reachable by the worker's computing unit. To do so, the worker calls `_starpu_fetch_task_input()`

#### **Data transfer processing**

The worker makes pending data transfers (involving memory node(s) that it is driving) progress, with a call to `__starpu_datawizard_progress()`,

#### **Task execution**

Once the worker has a pending task assigned and the input data for that task are available in the memory node reachable by the worker's computing unit, the worker calls `_starpu_cpu_driver_execute_task` (or the corresponding driver specific function) to proceed to the execution of the task.

### 2.1.3 Scheduling Contexts

A scheduling context is a logical set of workers governed by an instance of a scheduling policy. Tasks submitted to a given scheduling context are confined to the computing units governed by the workers belonging to this scheduling context at the time they get scheduled.

A scheduling context is identified by an unsigned integer identifier between 0 and `STARPU_NMAX_SCHED_CTXS - 1`. The `STARPU_NMAX_SCHED_CTXS` identifier value is reserved to indicate an unallocated, invalid or deleted scheduling context.

Accesses to the scheduling context structure are governed by a multiple-readers/single-writers lock (`rwlock` field). Changes to the structure contents, additions or removals of workers, statistics updates, all must be done with proper exclusive write access.

### 2.1.4 Workers and Scheduling Contexts

A worker can be assigned to one or more **scheduling contexts**. It exclusively receives tasks submitted to the scheduling context(s) it is currently assigned at the time such tasks are scheduled. A worker may add itself to or remove itself from a scheduling context.

#### **Locking and synchronization rules between workers and scheduling contexts**

A thread currently holding a worker `sched_mutex` must not attempt to acquire a scheduling context `rwlock`, neither for writing nor for reading. Such an attempt constitutes a lock inversion and may result in a deadlock.

A worker currently in a scheduling operation must enter the relaxed state before attempting to acquire a scheduling context `rwlock`, either for reading or for writing.

When the set of workers assigned to a scheduling context is about to be modified, all the workers in the union between the workers belonging to the scheduling context before the change and the workers expected to belong to the scheduling context after the change must be notified using the `notify_workers_about_changing_ctx_pending` function prior to the update. After the update, all the workers in that same union must be notified for the update completion with a call to `notify_workers_about_changing_ctx_done`.

The function `notify_workers_about_changing_ctx_pending` places every worker passed in argument in a state compatible with changing the scheduling context assignment of that worker, possibly blocking until that worker leaves incompatible states such as a pending scheduling operation. If the caller of `notify_workers_about_changing_ctx_pending()` is itself a worker included in the set of workers passed in argument, it does not notify itself, with the assumption that the worker is already calling `notify_workers_about_changing_ctx_pending()` from a state compatible with a scheduling context assignment update. Once a worker has been notified about a scheduling context change pending, it cannot proceed with incompatible operations such as a scheduling operation until it receives a notification that the context update operation is complete.

### 2.1.5 Drivers

Each driver defines a set of routines depending on some specific hardware. These routines include hardware discovery/initialization, task execution, device memory management and data transfers.

While most hardware dependent routines are in source files located in the `/src/drivers` subdirectory of the StarPU tree, some can be found elsewhere in the tree such as `src/datawizard/malloc.c` for memory allocation routines or the subdirectories of `src/datawizard/interfaces/` for data transfer routines. The driver ABI defined in the `_starpu_driver_ops` structure includes the following operations:

- `.init`: initialize a driver instance for the calling worker managing a hardware computing unit compatible with this driver.
- `.run_once`: perform a single driver progress cycle for the calling worker (see [Operations](#)).
- `.deinit`: deinitialize the driver instance for the calling worker
- `.run`: executes the following sequence automatically: call `.init`, repeatedly call `.run_once` until the function `_starpu_machine_is_running()` returns false, call `.deinit`.

The source code common to all drivers is shared in `src/drivers/driver_↔common/driver_common.[ch]`. This file includes services such as grabbing a new task to execute on a worker, managing statistics accounting on job startup and completion and updating the worker status

### 2.1.5.1 Master/Slave Drivers

A subset of the drivers corresponds to drivers managing computing units in master/slave mode, that is, drivers involving a local master instance managing one or more remote slave instances on the targeted device(s). This includes devices such as discrete manycore accelerators (e.g. Intel's Knight Corners board, for instance), or pseudo devices such as a cluster of cpu nodes driver through StarPU's MPI master/slave mode. A driver instance on the master side is named the **source**, while a driver instances on the slave side is named the **sink**.

A significant part of the work realized on the source and sink sides of master/slave drivers is identical among all master/slave drivers, due to the similarities in the software pattern. Therefore, many routines are shared among all these drivers in the `src/drivers/mp_common` subdirectory. In particular, a set of default commands to be used between sources and sinks is defined, assuming the availability of some communication channel between them (see enum `_starpu_mp_command`)

TODO

### 2.1.6 Tasks and Jobs

TODO

### 2.1.7 Data

TODO

# Chapter 3

## Module Index

### 3.1 Modules

Here is a list of all modules:

Workers . . . . . 13



## Chapter 4

# Module Documentation

### 4.1 Workers



# Chapter 5

## File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">starpu_config.h</a>	??
<a href="#">config.h</a>	31
<a href="#">starpu_driver.h</a>	??
<a href="#">starpu_worker.h</a>	??
<a href="#">data_movements_interface.h</a>	142
<a href="#">load_balancer_policy.h</a>	141
<a href="#">load_data_interface.h</a>	141
<a href="#">starpu_mpi_comm.h</a>	131
<a href="#">starpu_mpi_driver.h</a>	127
<a href="#">starpu_mpi_early_data.h</a>	129
<a href="#">starpu_mpi_early_request.h</a>	131
<a href="#">starpu_mpi_mpi.h</a>	130
<a href="#">starpu_mpi_mpi_backend.h</a>	132
<a href="#">starpu_mpi_sync_data.h</a>	131
<a href="#">starpu_mpi_tag.h</a>	138
<a href="#">starpu_mpi_nmad.h</a>	140
<a href="#">starpu_mpi_nmad_backend.h</a>	128
<a href="#">starpu_mpi_nmad_unknown_datatype.h</a>	130
<a href="#">starpu_mpi_cache.h</a>	127
<a href="#">starpu_mpi_cache_stats.h</a>	129
<a href="#">starpu_mpi_datatype.h</a>	138
<a href="#">starpu_mpi_fxt.h</a>	138
<a href="#">starpu_mpi_init.h</a>	127
<a href="#">starpu_mpi_private.h</a>	133
<a href="#">starpu_mpi_select_node.h</a>	140
<a href="#">starpu_mpi_stats.h</a>	128
<a href="#">starpu_mpi_task_insert.h</a>	140
<a href="#">barrier.h</a>	19
<a href="#">barrier_counter.h</a>	19
<a href="#">fxt.h</a>	56
<a href="#">graph.h</a>	62
<a href="#">knobs.h</a>	69
<a href="#">list.h</a>	??
<a href="#">prio_list.h</a>	86
<a href="#">rbtree.h</a>	88
<a href="#">rbtree_i.h</a>	92
<a href="#">rwlock.h</a>	95
<a href="#">starpu_spinlock.h</a>	109
<a href="#">thread.h</a>	116
<a href="#">timing.h</a>	116

uthash.h	123
utils.h	121
combined_workers.h	31
debug.h	43
cg.h	21
data_concurrency.h	40
implicit_data_deps.h	65
tags.h	110
detect_combined_workers.h	45
disk.h	45
disk_unistd_global.h	47
drivers.h	53
errorcheck.h	53
idle_hook.h	65
jobs.h	65
multiple_regression.h	78
perfmodel.h	84
regression.h	95
progress_hook.h	88
sched_ctx.h	97
sched_ctx_list.h	101
sched_policy.h	103
simgrid.h	105
task.h	111
task_bundle.h	114
topology.h	117
workers.h	??
coherency.h	23
copy_driver.h	38
data_request.h	41
datastats.h	42
datawizard.h	43
filters.h	55
footprint.h	55
data_interface.h	40
malloc.h	73
memalloc.h	74
memory_manager.h	74
memory_nodes.h	74
memstats.h	77
node_ops.h	78
sort_data_handles.h	107
write_back.h	125
starpu_debug_helpers.h	108
starpu_fxt.h	109
driver_cpu.h	49
driver_cuda.h	49
driver_disk.h	50
driver_common.h	48
mp_common.h	78
sink_common.h	107
source_common.h	107
driver_mpi_common.h	50
driver_mpi_sink.h	51
driver_mpi_source.h	51
driver_opencl.h	52
driver_opencl_utils.h	53
starpu_parallel_worker_create.h	107

---

<a href="#">bound.h</a>	20
<a href="#">profiling.h</a>	87
<a href="#">fifo_queues.h</a>	54
<a href="#">helper_mct.h</a>	64
<a href="#">prio_deque.h</a>	86
<a href="#">sched_component.h</a>	97
<a href="#">openmp_runtime_support.h</a>	80
<a href="#">starp_datacpy.h</a>	108
<a href="#">starp_task_insert_utils.h</a>	109
<a href="#">starpurm_private.h</a>	143



# Chapter 6

## StarPU File Documentation

### 6.1 barrier.h File Reference

```
#include <starpu_thread.h>
```

#### Data Structures

- [struct \\_starpu\\_barrier](#)

#### Functions

- `int _starpu_barrier_init` ([struct \\_starpu\\_barrier](#) \*barrier, int count)
- `int _starpu_barrier_destroy` ([struct \\_starpu\\_barrier](#) \*barrier)
- `int _starpu_barrier_wait` ([struct \\_starpu\\_barrier](#) \*barrier)

#### 6.1.1 Data Structure Documentation

##### 6.1.1.1 struct \_starpu\_barrier

###### Data Fields

unsigned	count	
unsigned	reached_start	
unsigned	reached_exit	
double	reached_flops	
starpu_pthread_mutex_t	mutex	
starpu_pthread_mutex_t	mutex_exit	
starpu_pthread_cond_t	cond	

### 6.2 barrier\_counter.h File Reference

```
#include <common/utils.h>  
#include <common/barrier.h>
```

#### Data Structures

- [struct \\_starpu\\_barrier\\_counter](#)

## Functions

- `int _starpu_barrier_counter_init (struct _starpu_barrier_counter *barrier_c, unsigned count)`
- `int _starpu_barrier_counter_destroy (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_wait_for_empty_counter (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_wait_until_counter_reaches_down_to_n (struct _starpu_barrier_counter *barrier_c, unsigned n)`
- `int _starpu_barrier_counter_wait_until_counter_reaches_up_to_n (struct _starpu_barrier_counter *barrier_c, unsigned n)`
- `int _starpu_barrier_counter_wait_for_full_counter (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_decrement_until_empty_counter (struct _starpu_barrier_counter *barrier_c, double flops)`
- `int _starpu_barrier_counter_increment_until_full_counter (struct _starpu_barrier_counter *barrier_c, double flops)`
- `int _starpu_barrier_counter_increment (struct _starpu_barrier_counter *barrier_c, double flops)`
- `int _starpu_barrier_counter_check (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_get_reached_start (struct _starpu_barrier_counter *barrier_c)`
- `int _starpu_barrier_counter_get_reached_exit (struct _starpu_barrier_counter *barrier_c)`
- `double _starpu_barrier_counter_get_reached_flops (struct _starpu_barrier_counter *barrier_c)`

## 6.2.1 Data Structure Documentation

### 6.2.1.1 struct \_starpu\_barrier\_counter

#### Data Fields

<code>struct _starpu_barrier</code>	barrier	
unsigned	min_threshold	
unsigned	max_threshold	
<code>starpu_pthread_cond_t</code>	cond2	

## 6.3 bound.h File Reference

```
#include <starpu.h>
#include <starpu_bound.h>
#include <core/jobs.h>
```

## Functions

- `void _starpu_bound_record (struct _starpu_job *j)`
- `void _starpu_bound_tag_dep (starpu_tag_t id, starpu_tag_t dep_id)`
- `void _starpu_bound_task_dep (struct _starpu_job *j, struct _starpu_job *dep_j)`
- `void _starpu_bound_job_id_dep (starpu_data_handle_t handle, struct _starpu_job *dep_j, unsigned long job_id)`
- `void starpu_bound_clear (void)`

## Variables

- `int _starpu_bound_recording`

### 6.3.1 Function Documentation

### 6.3.1.1 `_starpu_bound_record()`

```
void _starpu_bound_record (  
    struct _starpu_job * j )
```

Record task for bound computation

### 6.3.1.2 `_starpu_bound_tag_dep()`

```
void _starpu_bound_tag_dep (  
    starpu_tag_t id,  
    starpu_tag_t dep_id )
```

Record tag dependency: id depends on dep\_id

### 6.3.1.3 `_starpu_bound_task_dep()`

```
void _starpu_bound_task_dep (  
    struct _starpu_job * j,  
    struct _starpu_job * dep_j )
```

Record task dependency: j depends on dep\_j

### 6.3.1.4 `_starpu_bound_job_id_dep()`

```
void _starpu_bound_job_id_dep (  
    starpu_data_handle_t handle,  
    struct _starpu_job * dep_j,  
    unsigned long job_id )
```

Record job id dependency: j depends on job\_id

### 6.3.1.5 `starpu_bound_clear()`

```
void starpu_bound_clear (  
    void )
```

Clear recording

## 6.3.2 Variable Documentation

### 6.3.2.1 `_starpu_bound_recording`

```
int _starpu_bound_recording [extern]  
Are we recording?
```

## 6.4 cg.h File Reference

```
#include <starpu.h>  
#include <common/config.h>
```

### Data Structures

- struct [\\_starpu\\_cg\\_list](#)
- struct [\\_starpu\\_cg](#)
- union [\\_starpu\\_cg.succ](#)
- struct [\\_starpu\\_cg.succ.succ\\_apps](#)

### Macros

- #define [STARPU\\_DYNAMIC\\_DEPS\\_SIZE](#)

## Typedefs

- typedef [struct](#) \_starpu\_notify\_job\_start\_data **\_starpu\_notify\_job\_start\_data**

## Enumerations

- enum **\_starpu\_cg\_type** { STARPU\_CG\_APPS , STARPU\_CG\_TAG , STARPU\_CG\_TASK }

## Functions

- void **\_starpu\_notify\_dependencies** ([struct](#) \_starpu\_job \*j)
- void **\_starpu\_job\_notify\_start** ([struct](#) \_starpu\_job \*j, [struct](#) starpu\_perfmodel\_arch \*perf\_arch)
- void **\_starpu\_job\_notify\_ready\_soon** ([struct](#) \_starpu\_job \*j, \_starpu\_notify\_job\_start\_data \*data)
- void **\_starpu\_cg\_list\_init0** ([struct](#) \_starpu\_cg\_list \*list)
- void **\_starpu\_cg\_list\_deinit** ([struct](#) \_starpu\_cg\_list \*list)
- int **\_starpu\_add\_successor\_to\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, [struct](#) \_starpu\_cg \*cg)
- int **\_starpu\_list\_task\_successors\_in\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, unsigned ndeps, [struct](#) starpu\_task \*task\_array[])
- int **\_starpu\_list\_task\_scheduled\_successors\_in\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, unsigned ndeps, [struct](#) starpu\_task \*task\_array[])
- int **\_starpu\_list\_tag\_successors\_in\_cg\_list** ([struct](#) \_starpu\_cg\_list \*successors, unsigned ndeps, starpu\_tag\_t tag\_array[])
- void **\_starpu\_notify\_cg** (void \*pred, [struct](#) \_starpu\_cg \*cg)
- void **\_starpu\_notify\_cg\_list** (void \*pred, [struct](#) \_starpu\_cg\_list \*successors)
- void **\_starpu\_notify\_job\_start\_cg\_list** (void \*pred, [struct](#) \_starpu\_cg\_list \*successors, \_starpu\_notify\_↔ job\_start\_data \*data)
- void **\_starpu\_notify\_task\_dependencies** ([struct](#) \_starpu\_job \*j)
- void **\_starpu\_notify\_job\_start\_tasks** ([struct](#) \_starpu\_job \*j, \_starpu\_notify\_job\_start\_data \*data)

## 6.4.1 Data Structure Documentation

### 6.4.1.1 [struct](#) \_starpu\_cg\_list

Completion Group list, records both the number of expected notifications before the completion can start, and the list of successors when the completion is finished.

#### Data Fields

<a href="#">struct</a> _starpu_spinlock	lock	Protects atomicity of the list and the terminated flag
unsigned	ndeps	Number of notifications to be waited for
unsigned	ndeps_completed	
unsigned	terminated	Whether the completion is finished. For restartable/restarted tasks, only the first iteration is taken into account here.
unsigned	nsuccs	List of successors
unsigned	succ_list_size	How many allocated items in succ
<a href="#">struct</a> _starpu_cg **	succ	

### 6.4.1.2 [struct](#) \_starpu\_cg

Completion Group

#### Data Fields

unsigned	ntags	number of tags depended on
unsigned	remaining	number of remaining tags
enum _starpu_cg_type	cg_type	

## Data Fields

union <a href="#">_starpu_cg.succ</a>	succ	
---------------------------------------	------	--

6.4.1.3 union [\\_starpu\\_cg.succ](#)

## Data Fields

<a href="#">struct _starpu_tag *</a>	tag	STARPU_CG_TAG
<a href="#">struct _starpu_job *</a>	job	STARPU_CG_TASK
<a href="#">struct _starpu_cg.succ.succ_apps</a>	succ_apps	STARPU_CG_APPS in case this completion group is related to an application, we have to explicitly wake the waiting thread instead of reschedule the corresponding task

6.4.1.4 [struct \\_starpu\\_cg.succ.succ\\_apps](#)

STARPU\_CG\_APPS in case this completion group is related to an application, we have to explicitly wake the waiting thread instead of reschedule the corresponding task

## Data Fields

unsigned	completed	
<a href="#">starpu_pthread_mutex_t</a>	cg_mutex	
<a href="#">starpu_pthread_cond_t</a>	cg_cond	

## 6.4.2 Macro Definition Documentation

## 6.4.2.1 STARPU\_DYNAMIC\_DEPS\_SIZE

```
#define STARPU_DYNAMIC_DEPS_SIZE
```

we do not necessarily want to allocate room for 256 dependencies, but we want to handle the few situation where there are a lot of dependencies as well

## 6.5 coherency.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <common/rwlock.h>
#include <common/timing.h>
#include <common/fxt.h>
#include <common/list.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/datastats.h>
#include <datawizard/memstats.h>
#include <datawizard/data_request.h>
```

## Data Structures

- [struct \\_starpu\\_data\\_replicate](#)

- [struct \\_starpu\\_jobid\\_list](#)
- [struct \\_starpu\\_task\\_wrapper\\_list](#)
- [struct \\_starpu\\_task\\_wrapper\\_dlist](#)
- [struct \\_starpu\\_data\\_state](#)

## Macros

- `#define STARPU_UNMAPPED`

## Typedefs

- `typedef void(* _starpu_data_handle_unregister_hook) (starpu_data_handle_t)`

## Enumerations

- `enum _starpu_cache_state { STARPU_OWNER , STARPU_SHARED , STARPU_INVALID }`

## Functions

- `int _starpu_fetch_data_on_node (starpu_data_handle_t handle, int node, struct \_starpu\_data\_replicate *replicate, enum starpu_data_access_mode mode, unsigned detached, struct starpu\_task *task, enum starpu_is_prefetch is_prefetch, unsigned async, void(*callback_func)(void *), void *callback_arg, int prio, const char *origin)`
- `void _starpu_release_data_on_node (struct \_starpu\_data\_state *state, uint32_t default_wt_mask, enum starpu_data_access_mode down_to_mode, struct \_starpu\_data\_replicate *replicate)`
- `void _starpu_update_data_state (starpu_data_handle_t handle, struct \_starpu\_data\_replicate *requesting_↵_replicate, enum starpu_data_access_mode mode)`
- `uint32_t _starpu_get_data_refcnt (struct \_starpu\_data\_state *state, unsigned node)`
- `size_t _starpu_data_get_size (starpu_data_handle_t handle)`
- `size_t _starpu_data_get_alloc_size (starpu_data_handle_t handle)`
- `starpu_ssize_t _starpu_data_get_max_size (starpu_data_handle_t handle)`
- `uint32_t _starpu_data_get_footprint (starpu_data_handle_t handle)`
- `void __starpu_push_task_output (struct \_starpu\_job *j)`
- `void _starpu_push_task_output (struct \_starpu\_job *j)`
- `STARPU_ATTRIBUTE_WARN_UNUSED_RESULT int _starpu_fetch_task_input (struct starpu\_task *task, struct \_starpu\_job *j, int async)`
- `void _starpu_fetch_task_input_tail (struct starpu\_task *task, struct \_starpu\_job *j, struct \_starpu\_worker *worker)`
- `void _starpu_fetch_nowhere_task_input (struct \_starpu\_job *j)`
- `int _starpu_select_src_node (struct \_starpu\_data\_state *state, unsigned destination)`
- `int _starpu_determine_request_path (starpu_data_handle_t handle, int src_node, int dst_node, enum starpu_data_access_mode mode, int max_len, unsigned *src_nodes, unsigned *dst_nodes, unsigned *handling_nodes, unsigned write_invalidation)`
- `struct starpu\_data\_request * _starpu_create_request_to_fetch_data (starpu_data_handle_t handle, struct \_starpu\_data\_replicate *dst_replicate, enum starpu_data_access_mode mode, struct starpu\_task *task, enum starpu_is_prefetch is_prefetch, unsigned async, void(*callback_func)(void *), void *callback_arg, int prio, const char *origin)`
- `void _starpu_init_data_replicate (starpu_data_handle_t handle, struct \_starpu\_data\_replicate *replicate, int workerid)`
- `void _starpu_data_start_reduction_mode (starpu_data_handle_t handle)`
- `void _starpu_data_end_reduction_mode (starpu_data_handle_t handle, int priority)`
- `void _starpu_data_end_reduction_mode_terminate (starpu_data_handle_t handle)`
- `void _starpu_data_unmap (starpu_data_handle_t handle, unsigned node)`
- `void _starpu_data_set_unregister_hook (starpu_data_handle_t handle, _starpu_data_handle_↵_t_unregister_hook func) STARPU_ATTRIBUTE_VISIBILITY_DEFAULT`

## Variables

- `int _starpu_has_not_important_data`

### 6.5.1 Data Structure Documentation

#### 6.5.1.1 `struct _starpu_data_replicate`

this should contain the information relative to a given data replicate

##### Data Fields

<code>starpu_data_handle_t</code>	<code>handle</code>	
<code>void *</code>	<code>data_interface</code>	describe the actual data layout, as manipulated by data interfaces in <code>*_interface.c</code>
<code>int</code>	<code>refcnt</code>	How many requests or tasks are currently working with this replicate
<code>char</code>	<code>memory_node</code>	
<code>enum _starpu_cache_state</code>	<code>state: 2</code>	describes the state of the local data in term of coherency
<code>unsigned</code>	<code>relaxed_coherency:2</code>	A buffer that is used for SCRATCH or reduction cannot be used with filters.
<code>unsigned</code>	<code>initialized:1</code>	We may need to initialize the replicate with some value before using it.
<code>unsigned</code>	<code>allocated:1</code>	is the data locally allocated ?
<code>unsigned</code>	<code>automatically_allocated:1</code>	was it automatically allocated ? (else it's the application-provided buffer, don't ever try to free it!) perhaps the allocation was perform higher in the hierarchy for now this is just translated into <code>!automatically_allocated</code>
<code>unsigned</code>	<code>map_write:1</code>	is the write side enabled on the mapping? This is important for drivers which may actually make a copy instead of a map. Only meaningful when mapped <code>!= STARPU_UNMAPPED</code>
<code>int</code>	<code>mapped</code>	<code>&gt;= 0</code> when the data just a mapping of a replicate from that memory node, otherwise <code>STARPU_UNMAPPED</code>
<code>uint32_t</code>	<code>requested</code>	To help the scheduling policies to make some decision, we may keep a track of the tasks that are likely to request this data on the current node. It is the responsibility of the scheduling <i>policy</i> to set that flag when it assigns a task to a queue, policies which do not use this hint can simply ignore it.
<code>struct _starpu_data_request *</code>	<code>request[STARPU_MAXNODES]</code>	This tracks the list of requests to provide the value

## Data Fields

<a href="#">struct</a> _starpu_data_request *	last_request[ <a href="#">STARPU_MAXNODES</a> ]	This points to the last entry of request, to easily append to the list
<a href="#">struct</a> _starpu_data_request *	load_request	
unsigned	nb_tasks_prefetch	The number of prefetches that we made for this replicate for various tasks This is also the number of tasks that we will wait to see use the mc before we attempt to evict it.
<a href="#">struct</a> _starpu_mem_chunk *	mc	Pointer to memchunk for LRU strategy

## 6.5.1.2 struct\_starpu\_jobid\_list

## Data Fields

unsigned long	id	
<a href="#">struct</a> _starpu_jobid_list *	next	

## 6.5.1.3 struct\_starpu\_task\_wrapper\_list

This structure describes a simply-linked list of task

## Data Fields

<a href="#">struct</a> starpu_task *	task	
<a href="#">struct</a> _starpu_task_wrapper_list *	next	

## 6.5.1.4 struct\_starpu\_task\_wrapper\_dlist

This structure describes a doubly-linked list of task

## Data Fields

<a href="#">struct</a> starpu_task *	task	
<a href="#">struct</a> _starpu_task_wrapper_dlist *	next	
<a href="#">struct</a> _starpu_task_wrapper_dlist *	prev	

## 6.5.1.5 struct\_starpu\_data\_state

This is initialized in both \_starpu\_register\_new\_data and \_starpu\_data\_partition

## Data Fields

int	magic	
<a href="#">struct</a> _starpu_data_requester_prio_list	req_list	
unsigned	refcnt	the number of requests currently in the scheduling engine (not in the req_list anymore), i.e. the number of holders of the current_mode rwlock

## Data Fields

unsigned	unlocking_reqs	whether we are already unlocking data requests
enum starpu_data_access_mode	current_mode	Current access mode. Is always either STARPU_R, STARPU_W, STARPU_SCRATCH or STARPU_REDUX, but never a combination such as STARPU_RW.
<a href="#">struct_starpu_spinlock</a>	header_lock	protect meta data
unsigned	busy_count	Condition to make application wait for all transfers before freeing handle busy_count is the number of handle->refcnt, handle->per_node[*]->refcnt, number of starpu_data_requesters, and number of tasks that have released it but are still registered on the implicit data dependency lists. Core code which releases busy_count has to call _starpu_data_check_not_busy to let starpu_data_unregister proceed
unsigned	busy_waiting	Is starpu_data_unregister waiting for busy_count?
starpu_pthread_mutex_t	busy_mutex	
starpu_pthread_cond_t	busy_cond	
<a href="#">struct_starpu_data_state</a> *	root_handle	In case we use filters, the handle may describe a sub-data
<a href="#">struct_starpu_data_state</a> *	father_handle	root of the tree
starpu_data_handle_t *	active_children	father of the node, NULL if the current node is the root
unsigned	active_nchildren	The currently active set of read-write children
starpu_data_handle_t **	active_readonly_children	
unsigned *	active_readonly_nchildren	The currently active set of read-only children
unsigned	nactive_readonly_children	Size of active_readonly_children[i] array
unsigned	nsiblings	Size of active_readonly_children and active_readonly_nchildren arrays. Actual use is given by 'partitioned' Our siblings in the father partitioning
starpu_data_handle_t *	siblings	How many siblings
unsigned	sibling_index	
unsigned	depth	indicate which child this node is from the father's perspective (if any)
starpu_pthread_mutex_t	unpartition_mutex	what's the depth of the tree ?
starpu_data_handle_t	children	Synchronous partitioning

## Data Fields

	unsigned	nchildren	
	unsigned	nplans	How many partition plans this handle has
	<a href="#">struct starpu_codelet *</a>	switch_cl	Switch codelet for asynchronous partitioning
	unsigned	switch_cl_nparts	size of dyn_nodes recorded in switch_cl
	unsigned	partitioned	Whether a partition plan is currently submitted and the corresponding unpartition has not been yet Or the number of partition plans currently submitted in readonly mode.
	unsigned	part_readonly:1	Whether a partition plan is currently submitted in readonly mode
	unsigned	active:1	Whether our father is currently partitioned into ourself
	unsigned	active_ro:1	
	<a href="#">struct _starpu_data_replicate</a>	per_node[STARPU_MAXNODES]	describe the state of the data in term of coherency This is execution-time state.
	<a href="#">struct _starpu_data_replicate *</a>	per_worker	
	<a href="#">struct starpu_data_interface_ops *</a>	ops	
	uint32_t	footprint	Footprint which identifies data layout
	unsigned	is_not_important:1	in some case, the application may explicitly tell StarPU that a piece of data is not likely to be used soon again
	unsigned	ooc:1	Can the data be pushed to the disk?
	unsigned	sequential_consistency:1	Does StarPU have to enforce some implicit data-dependencies ?
	unsigned	readonly:1	Whether we shall not ever write to this handle, thus allowing various optimizations
	int	home_node	where is the data home, i.e. which node it was registered from ? -1 if none yet
	uint32_t	wt_mask	what is the default write-through mask for that data ?
	unsigned	aliases	for a readonly handle, the number of times that we have returned again the same handle and thus the number of times we have to ignore unregistration requests
	starpu_data_handle_t	readonly_dup	for a non-readonly handle, a readonly-only duplicate, that we can return from starpu_data_dup_ro

## Data Fields

starpu_data_handle_t	readonly_dup_of	for a readonly handle, the non-readonly handle that is referencing is in its readonly_dup field.
unsigned	initialized:1	Is the data initialized, or a task is already submitted to initialize it This is submission-time initialization state.
unsigned	removed_from_context_hash:1	
unsigned char	lazy_unregister	Whether lazy unregistration was requested through starpu_data_unregister_submit
starpu_pthread_mutex_t	sequential_consistency_mutex	This lock should protect any operation to enforce sequential_consistency
enum starpu_data_access_mode	last_submitted_mode	The last submitted task (or application data request) that declared it would modify the piece of data ? Any task accessing the data in a read-only mode should depend on that task implicitly if the sequential_consistency flag is enabled.
<a href="#">struct</a> starpu_task *	last_sync_task	
<a href="#">struct_starpu_task_wrapper_dlist</a>	last_submitted_accessors	
unsigned	last_submitted_ghost_sync_id_is_valid	If FxT is enabled, we keep track of "ghost dependencies": that is to say the dependencies that are not needed anymore, but that should appear in the post-mortem DAG. For instance if we have the sequence f(Aw) g(Aw), and that g is submitted after the termination of f, we want to have f->g appear in the DAG even if StarPU does not need to enforce this dependency anymore.
unsigned long	last_submitted_ghost_sync_id	
<a href="#">struct_starpu_jobid_list</a> *	last_submitted_ghost_accessors_id	
<a href="#">struct_starpu_task_wrapper_list</a> *	post_sync_tasks	protected by sequential_consistency_mutex
unsigned	post_sync_tasks_cnt	
<a href="#">struct</a> starpu_codelet *	redux_cl	During reduction we need some specific methods: redux_func performs the reduction of an interface into another one (eg. "+="), and init_func initializes the data interface to a default value that is stable by reduction (eg. 0 for +=).
<a href="#">struct</a> starpu_codelet *	init_cl	
void *	redux_cl_arg	
void *	init_cl_arg	

## Data Fields

unsigned	reduction_refcnt	Are we currently performing a reduction on that handle ? If so the reduction_refcnt should be non null until there are pending tasks that are performing the reduction.
struct _starpu_data_requester_prio_list	reduction_req_list	List of requesters that are specific to the pending reduction. This list is used when the requests in the req_list list are frozen until the end of the reduction.
starpu_data_handle_t *	reduction_tmp_handles	
struct starpu_data_request *	write_invalidation_req	Final request for write invalidation
void *	mpi_data	Used for MPI
_starpu_memory_stats_t	memory_stats	
unsigned int	mf_node	
_starpu_data_handle_unregister_hook	unregister_hook	hook to be called when unregistering the data
struct starpu_arbiter *	arbiter	
struct _starpu_data_requester_prio_list	arbitered_req_list	This is protected by the arbiter mutex
int	last_locality	Data maintained by schedulers themselves Last worker that took this data in locality mode, or -1 if nobody took it yet
unsigned	dimensions	Application-provided coordinates. The maximum dimension (5) is relatively arbitrary.
int	coordinates[5]	
void *	user_data	A generic pointer to data in the user land (could be anything and this is not manage by StarPU)
void *	sched_data	A generic pointer to data in the scheduler (could be anything and this is managed by the scheduler)

## 6.5.2 Function Documentation

### 6.5.2.1 \_starpu\_fetch\_data\_on\_node()

```
int _starpu_fetch_data_on_node (
    starpu_data_handle_t handle,
    int node,
    struct _starpu_data_replicate * replicate,
    enum starpu_data_access_mode mode,
    unsigned detached,
    struct starpu_task * task,
    enum starpu_is_prefetch is_prefetch,
    unsigned async,
    void(*) (void *) callback_func,
    void * callback_arg,
```

```
int prio,
const char * origin )
```

This does not take a reference on the handle, the caller has to do it, e.g. through `_starpu_attempt_to_submit_data` ↔ `_request_from_apps()` detached means that the core is allowed to drop the request. The caller should thus *not* take a reference since it can not know whether the request will complete async means that `_starpu_fetch_data_on_node` will wait for completion of the request

### 6.5.2.2 `_starpu_release_data_on_node()`

```
void _starpu_release_data_on_node (
    struct _starpu_data_state * state,
    uint32_t default_wt_mask,
    enum starpu_data_access_mode down_to_mode,
    struct _starpu_data_replicate * replicate )
```

This releases a reference on the handle

### 6.5.2.3 `_starpu_push_task_output()`

```
void _starpu_push_task_output (
    struct _starpu_job * j )
```

Version with driver trace

### 6.5.2.4 `_starpu_fetch_task_input()`

```
STARPU_ATTRIBUTE_WARN_UNUSED_RESULT int _starpu_fetch_task_input (
    struct starpu_task * task,
    struct _starpu_job * j,
    int async )
```

Fetch the data parameters for task `task` Setting `async` to 1 allows to only start the fetches, and call `_starpu` ↔ `_fetch_task_input_tail` later when the transfers are finished

### 6.5.2.5 `_starpu_create_request_to_fetch_data()`

```
struct _starpu_data_request * _starpu_create_request_to_fetch_data (
    starpu_data_handle_t handle,
    struct _starpu_data_replicate * dst_replicate,
    enum starpu_data_access_mode mode,
    struct starpu_task * task,
    enum starpu_is_prefetch is_prefetch,
    unsigned async,
    void(*) (void *) callback_func,
    void * callback_arg,
    int prio,
    const char * origin )
```

`is_prefetch` is whether the DSM may drop the request (when there is not enough memory for instance `async` is whether the caller wants a reference on the last request, to be able to wait for it (which will release that reference).

## 6.6 combined\_workers.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

## 6.7 config.h File Reference

### Macros

- `#define CONFIG_FUT`

- #define **HAVE\_AIO\_H**
- #define **HAVE\_AYUDAME\_H**
- #define **HAVE\_CBLAS\_SGEMV**
- #define **HAVE\_CLENQUEUEMARKERWITHWAITLIST**
- #define **HAVE\_CLGETEXTENSIONFUNCTIONADDRESSFORPLATFORM**
- #define **HAVE\_CLOCK\_GETTIME**
- #define **HAVE\_CL\_CL\_EXT\_H**
- #define **HAVE\_COPY\_FILE\_RANGE**
- #define **HAVE\_CUBLASLT\_H**
- #define **HAVE\_CUDA\_GL\_INTEROP\_H**
- #define **HAVE\_CXX11**
- #define **HAVE\_DECL\_CUSPARSESETSTREAM**
- #define **HAVE\_DECL\_ENABLE\_FUT\_FLUSH**
- #define **HAVE\_DECL\_FUT\_SETUP\_FLUSH\_CALLBACK**
- #define **HAVE\_DECL\_FUT\_SET\_FILENAME**
- #define **HAVE\_DECL\_HWLOC\_CUDA\_GET\_DEVICE\_OSDEV\_BY\_INDEX**
- #define **HAVE\_DECL\_HWLOC\_DISTANCES\_OBJ\_PAIR\_VALUES**
- #define **HAVE\_DECL\_HWLOC\_HIP\_GET\_DEVICE\_OSDEV\_BY\_INDEX**
- #define **HAVE\_DECL\_NVMLDEVICEGETTOTAENERGYCONSUMPTION**
- #define **HAVE\_DECL\_SMPI\_PROCESS\_SET\_USER\_DATA**
- #define **HAVE\_DLB\_H**
- #define **HAVE\_DLFCN\_H**
- #define **HAVE\_DLOPEN**
- #define **HAVE\_ENABLE\_FUT\_FLUSH**
- #define **HAVE\_FUT\_SETUP\_FLUSH\_CALLBACK**
- #define **HAVE\_FUT\_SET\_FILENAME**
- #define **HAVE\_FXT\_BLOCKEV\_LEAVE**
- #define **HAVE\_FXT\_CLOSE**
- #define **HAVE\_GETPAGESIZE**
- #define **HAVE\_GETRLIMIT**
- #define **HAVE\_GLPK\_H**
- #define **HAVE\_HDF5\_H**
- #define **HAVE\_HIP\_HIP\_RUNTIME\_API\_H**
- #define **HAVE\_HIP\_HIP\_RUNTIME\_H**
- #define **HAVE\_HWLOC\_CPUKINDS\_GET\_NR**
- #define **HAVE\_HWLOC\_GET\_AREA\_MEMLOCATION**
- #define **HAVE\_HWLOC\_GLIBC\_SCHED\_H**
- #define **HAVE\_HWLOC\_TOPOLOGY\_DUP**
- #define **HAVE\_HWLOC\_TOPOLOGY\_SET\_COMPONENTS**
- #define **HAVE\_INTTYPES\_H**
- #define **HAVE\_LEVELDB\_DB\_H**
- #define **HAVE\_LIBATLAS**
- #define **HAVE\_LIBBLAS\_OPENBLAS**
- #define **HAVE\_LIBCBLAS**
- #define **HAVE\_LIBDL**
- #define **HAVE\_LIBDLB**
- #define **HAVE\_LIBF77BLAS**
- #define **HAVE\_LIBGFORTTRAN**
- #define **HAVE\_LIBGL**
- #define **HAVE\_LIBGLPK**
- #define **HAVE\_LIBGLU**
- #define **HAVE\_LIBGLUT**
- #define **HAVE\_LIBGOTO**
- #define **HAVE\_LIBGOTO2**
- #define **HAVE\_LIBHDF5**

- #define **HAVE\_LIBIFCORE**
- #define **HAVE\_LIBLEVELDB**
- #define **HAVE\_LIBOPENBLAS**
- #define **HAVE\_LIBRT**
- #define **HAVE\_LIBSIMGRID**
- #define **HAVE\_LIBWS2\_32**
- #define **HAVE\_MALLOC\_H**
- #define **HAVE\_MEMALIGN**
- #define **HAVE\_MKDTEMP**
- #define **HAVE\_MKOSTEMP**
- #define **HAVE\_MMAP**
- #define **HAVE\_MPI\_COMM\_F2C**
- #define **HAVE\_MSG\_ENVIRONMENT\_GET\_ROUTING\_ROOT**
- #define **HAVE\_MSG\_GET\_AS\_BY\_NAME**
- #define **HAVE\_MSG\_HOST\_GET\_SPEED**
- #define **HAVE\_MSG\_MSG\_H**
- #define **HAVE\_MSG\_PROCESS\_ATTACH**
- #define **HAVE\_MSG\_PROCESS\_SELF\_NAME**
- #define **HAVE\_MSG\_PROCESS\_USERDATA\_INIT**
- #define **HAVE\_MSG\_ZONE\_GET\_BY\_NAME**
- #define **HAVE\_MSG\_ZONE\_GET\_HOSTS**
- #define **HAVE\_NM\_TRACE\_ADD\_SYNCHRO\_POINT**
- #define **HAVE\_PIOMAN**
- #define **HAVE\_PIOM\_LTASK\_SET\_BOUND\_THREAD\_OS\_INDEXES**
- #define **HAVE\_POSIX\_MEMALIGN**
- #define **HAVE\_POTI\_INIT\_CUSTOM**
- #define **HAVE\_POTI\_USER\_NEWEVENT**
- #define **HAVE\_PREAD**
- #define **HAVE\_PTHREAD\_SETAFFINITY\_NP**
- #define **HAVE\_PTHREAD\_SPIN\_LOCK**
- #define **HAVE\_PWRITE**
- #define **HAVE\_PYTHON\_H**
- #define **HAVE\_ROCBLAS\_ROCBLAS\_H**
- #define **HAVE\_SCANDIR**
- #define **HAVE\_SG\_ACTOR\_ATTACH**
- #define **HAVE\_SG\_ACTOR\_ATTACH\_PTHREAD**
- #define **HAVE\_SG\_ACTOR\_DATA**
- #define **HAVE\_SG\_ACTOR\_EXECUTE**
- #define **HAVE\_SG\_ACTOR\_GET\_DATA**
- #define **HAVE\_SG\_ACTOR\_INIT**
- #define **HAVE\_SG\_ACTOR\_ON\_EXIT**
- #define **HAVE\_SG\_ACTOR\_REF**
- #define **HAVE\_SG\_ACTOR\_SELF**
- #define **HAVE\_SG\_ACTOR\_SELF\_EXECUTE**
- #define **HAVE\_SG\_ACTOR\_SET\_DATA**
- #define **HAVE\_SG\_ACTOR\_SET\_STACKSIZE**
- #define **HAVE\_SG\_ACTOR\_SLEEP\_FOR**
- #define **HAVE\_SG\_CFG\_SET\_INT**
- #define **HAVE\_SG\_CONFIG\_CONTINUE\_AFTER\_HELP**
- #define **HAVE\_SG\_HOST\_GET\_PROPERTIES**
- #define **HAVE\_SG\_HOST\_GET\_PROPERTY\_NAMES**
- #define **HAVE\_SG\_HOST\_GET\_ROUTE**
- #define **HAVE\_SG\_HOST\_GET\_ROUTE\_LINKS**
- #define **HAVE\_SG\_HOST\_GET\_SPEED**
- #define **HAVE\_SG\_HOST\_LIST**

- #define **HAVE\_SG\_HOST\_ROUTE**
- #define **HAVE\_SG\_HOST\_SELF**
- #define **HAVE\_SG\_HOST\_SENDTO**
- #define **HAVE\_SG\_HOST\_SEND\_TO**
- #define **HAVE\_SG\_HOST\_SPEED**
- #define **HAVE\_SG\_LINK\_BANDWIDTH\_SET**
- #define **HAVE\_SG\_LINK\_GET\_NAME**
- #define **HAVE\_SG\_LINK\_NAME**
- #define **HAVE\_SG\_LINK\_SET\_BANDWIDTH**
- #define **HAVE\_SG\_ZONE\_GET\_ALL\_HOSTS**
- #define **HAVE\_SG\_ZONE\_GET\_BY\_NAME**
- #define **HAVE\_SG\_ZONE\_GET\_HOSTS**
- #define **HAVE\_SIMCALL\_PROCESS\_CREATE**
- #define **HAVE\_SIMGRID\_ACTOR\_H**
- #define **HAVE\_SIMGRID\_BARRIER\_H**
- #define **HAVE\_SIMGRID\_COND\_H**
- #define **HAVE\_SIMGRID\_ENGINE\_H**
- #define **HAVE\_SIMGRID\_GET\_CLOCK**
- #define **HAVE\_SIMGRID\_HOST\_H**
- #define **HAVE\_SIMGRID\_INIT**
- #define **HAVE\_SIMGRID\_LINK\_H**
- #define **HAVE\_SIMGRID\_MSG\_H**
- #define **HAVE\_SIMGRID\_MUTEX\_H**
- #define **HAVE\_SIMGRID\_SEMAPHORE\_H**
- #define **HAVE\_SIMGRID\_SET\_MAESTRO**
- #define **HAVE\_SIMGRID\_SIMDAG\_H**
- #define **HAVE\_SIMGRID\_VERSION\_H**
- #define **HAVE\_SIMGRID\_ZONE\_H**
- #define **HAVE\_SMPI\_PROCESS\_SET\_USER\_DATA**
- #define **HAVE\_SMPI\_THREAD\_CREATE**
- #define **HAVE\_SMX\_ACTOR\_T**
- #define **HAVE\_STDINT\_H**
- #define **HAVE\_STDIO\_H**
- #define **HAVE\_STDLIB\_H**
- #define **HAVE\_STRINGS\_H**
- #define **HAVE\_STRING\_H**
- #define **HAVE\_SYSCONF**
- #define **HAVE\_SYS\_PARAM\_H**
- #define **HAVE\_SYS\_STAT\_H**
- #define **HAVE\_SYS\_TYPES\_H**
- #define **HAVE\_UNISTD\_H**
- #define **HAVE\_VALGRIND\_HELGRIND\_H**
- #define **HAVE\_VALGRIND\_MEMCHECK\_H**
- #define **HAVE\_VALGRIND\_VALGRIND\_H**
- #define **HAVE\_XBT\_BARRIER\_INIT**
- #define **HAVE\_XBT\_BASE\_H**
- #define **HAVE\_XBT\_CONFIG\_H**
- #define **HAVE\_XBT\_MUTEX\_TRY\_ACQUIRE**
- #define **HAVE\_XBT\_SYNCHRO\_H**
- #define **LT\_OBJDIR**
- #define **PACKAGE**
- #define **PACKAGE\_BUGREPORT**
- #define **PACKAGE\_NAME**
- #define **PACKAGE\_STRING**
- #define **PACKAGE\_TARNAME**

- #define **PACKAGE\_URL**
- #define **PACKAGE\_VERSION**
- #define **SIZEOF\_VOID\_P**
- #define **STARPURM\_DLB\_VERBOSE**
- #define **STARPURM\_HAVE\_DLB**
- #define **STARPURM\_HAVE\_DLB\_CALLBACK\_ARG**
- #define **STARPURM\_STARPU\_HAVE\_WORKER\_CALLBACKS**
- #define **STARPURM\_VERBOSE**
- #define **STARPU\_ARMPL**
- #define **STARPU\_ATLAS**
- #define **STARPU\_BUBBLE**
- #define **STARPU\_BUBBLE\_VERBOSE**
- #define **STARPU\_BUILD\_DIR**
- #define **STARPU\_BUILT\_IN\_MIN\_DGELS**
- #define **STARPU\_COVERITY**
- #define **STARPU\_DATA\_LOCALITY\_ENFORCE**
- #define **STARPU\_DEBUG**
- #define **STARPU\_DEVEL**
- #define **STARPU\_DISABLE\_ASYNCHRONOUS\_COPY**
- #define **STARPU\_DISABLE\_ASYNCHRONOUS\_CUDA\_COPY**
- #define **STARPU\_DISABLE\_ASYNCHRONOUS\_MAX\_FPGA\_COPY**
- #define **STARPU\_DISABLE\_ASYNCHRONOUS\_MPI\_MS\_COPY**
- #define **STARPU\_DISABLE\_ASYNCHRONOUS\_OPENCL\_COPY**
- #define **STARPU\_DISABLE\_ASYNCHRONOUS\_TCPIP\_MS\_COPY**
- #define **STARPU\_EXTRA\_VERBOSE**
- #define **STARPU\_FXT\_LOCK\_TRACES**
- #define **STARPU\_FXT\_MAX\_FILES**
- #define **STARPU\_GDB\_PATH**
- #define **STARPU\_GOTO**
- #define **STARPU\_HAVE\_ATOMIC\_COMPARE\_EXCHANGE\_N**
- #define **STARPU\_HAVE\_ATOMIC\_COMPARE\_EXCHANGE\_N\_8**
- #define **STARPU\_HAVE\_ATOMIC\_EXCHANGE\_N**
- #define **STARPU\_HAVE\_ATOMIC\_EXCHANGE\_N\_8**
- #define **STARPU\_HAVE\_ATOMIC\_FETCH\_ADD**
- #define **STARPU\_HAVE\_ATOMIC\_FETCH\_ADD\_8**
- #define **STARPU\_HAVE\_ATOMIC\_FETCH\_OR**
- #define **STARPU\_HAVE\_ATOMIC\_FETCH\_OR\_8**
- #define **STARPU\_HAVE\_ATOMIC\_TEST\_AND\_SET**
- #define **STARPU\_HAVE\_BLAS**
- #define **STARPU\_HAVE\_BUSID**
- #define **STARPU\_HAVE\_CBLAS\_H**
- #define **STARPU\_HAVE\_CUDA\_CANMAPHOST**
- #define **STARPU\_HAVE\_CUDA\_MEMCPY\_PEER**
- #define **STARPU\_HAVE\_CUDA\_MNGMEM**
- #define **STARPU\_HAVE\_CUDA\_PAGEABLEMEM**
- #define **STARPU\_HAVE\_CUDA\_POINTER\_TYPE**
- #define **STARPU\_HAVE\_CUDA\_UNIFIEDADDR**
- #define **STARPU\_HAVE\_CUFFTDOUBLECOMPLEX**
- #define **STARPU\_HAVE\_CURAND**
- #define **STARPU\_HAVE\_CXX11**
- #define **STARPU\_HAVE\_DARWIN**
- #define **STARPU\_HAVE\_DOMAINID**
- #define **STARPU\_HAVE\_F77\_H**
- #define **STARPU\_HAVE\_FC**
- #define **STARPU\_HAVE\_FFTW**

- #define STARPU\_HAVE\_FFTWF
- #define STARPU\_HAVE\_FFTWL
- #define STARPU\_HAVE\_GLPK\_H
- #define STARPU\_HAVE\_HDF5
- #define STARPU\_HAVE\_HELGRIND\_H
- #define STARPU\_HAVE\_HIP\_MEMCPY\_PEER
- #define STARPU\_HAVE\_HWLOC
- #define STARPU\_HAVE\_ICC
- #define STARPU\_HAVE\_LEVELDB
- #define STARPU\_HAVE\_LIBCUBLASLT
- #define STARPU\_HAVE\_LIBCUSOLVER
- #define STARPU\_HAVE\_LIBCUSPARSE
- #define STARPU\_HAVE\_LIBNUMA
- #define STARPU\_HAVE\_MAGMA
- #define STARPU\_HAVE\_MALLOC\_H
- #define STARPU\_HAVE\_MEMALIGN
- #define STARPU\_HAVE\_MEMCHECK\_H
- #define STARPU\_HAVE\_MPI\_COMM\_CREATE\_GROUP
- #define STARPU\_HAVE\_MPI\_EXT
- #define STARPU\_HAVE\_MPI\_SYNC\_CLOCKS
- #define STARPU\_HAVE\_MSG\_MSG\_H
- #define STARPU\_HAVE\_NEARBYINTF
- #define STARPU\_HAVE\_NVML\_H
- #define STARPU\_HAVE\_POSIX\_MEMALIGN
- #define STARPU\_HAVE\_POTI
- #define STARPU\_HAVE\_PROGRAM\_INVOCATION\_SHORT\_NAME
- #define STARPU\_HAVE\_PTHREAD\_BARRIER
- #define STARPU\_HAVE\_PTHREAD\_SETNAME\_NP
- #define STARPU\_HAVE\_PTHREAD\_SPIN\_LOCK
- #define STARPU\_HAVE\_RINTF
- #define STARPU\_HAVE\_S4U\_ON\_TIME\_ADVANCE\_CB
- #define STARPU\_HAVE\_SCHED\_YIELD
- #define STARPU\_HAVE\_SETENV
- #define STARPU\_HAVE\_SIMGRID\_ACTOR\_H
- #define STARPU\_HAVE\_SIMGRID\_BARRIER\_H
- #define STARPU\_HAVE\_SIMGRID\_COND\_H
- #define STARPU\_HAVE\_SIMGRID\_ENGINE\_H
- #define STARPU\_HAVE\_SIMGRID\_HOST\_H
- #define STARPU\_HAVE\_SIMGRID\_LINK\_H
- #define STARPU\_HAVE\_SIMGRID\_MSG\_H
- #define STARPU\_HAVE\_SIMGRID\_MUTEX\_H
- #define STARPU\_HAVE\_SIMGRID\_SEMAPHORE\_H
- #define STARPU\_HAVE\_SIMGRID\_SIMDAG\_H
- #define STARPU\_HAVE\_SIMGRID\_VERSION\_H
- #define STARPU\_HAVE\_SIMGRID\_ZONE\_H
- #define STARPU\_HAVE\_SMX\_ACTOR\_T
- #define STARPU\_HAVE\_STATEMENT\_EXPRESSIONS
- #define STARPU\_HAVE\_STRERROR\_R
- #define STARPU\_HAVE\_STRUCT\_TIMESPEC
- #define STARPU\_HAVE\_SYNC\_BOOL\_COMPARE\_AND\_SWAP
- #define STARPU\_HAVE\_SYNC\_BOOL\_COMPARE\_AND\_SWAP\_8
- #define STARPU\_HAVE\_SYNC\_FETCH\_AND\_ADD
- #define STARPU\_HAVE\_SYNC\_FETCH\_AND\_ADD\_8
- #define STARPU\_HAVE\_SYNC\_FETCH\_AND\_OR
- #define STARPU\_HAVE\_SYNC\_FETCH\_AND\_OR\_8

- #define **STARPU\_HAVE\_SYNC\_LOCK\_TEST\_AND\_SET**
- #define **STARPU\_HAVE\_SYNC\_SYNCHRONIZE**
- #define **STARPU\_HAVE\_SYNC\_VAL\_COMPARE\_AND\_SWAP**
- #define **STARPU\_HAVE\_SYNC\_VAL\_COMPARE\_AND\_SWAP\_8**
- #define **STARPU\_HAVE\_UNISTD\_H**
- #define **STARPU\_HAVE\_UNSETENV**
- #define **STARPU\_HAVE\_VALGRIND\_H**
- #define **STARPU\_HAVE\_WINDOWS**
- #define **STARPU\_HAVE\_X11**
- #define **STARPU\_HAVE\_XBT\_BASE\_H**
- #define **STARPU\_HAVE\_XBT\_CONFIG\_H**
- #define **STARPU\_HAVE\_XBT\_SYNCHRO\_H**
- #define **STARPU\_HISTORYMAXERROR**
- #define **STARPU\_LINUX\_SYS**
- #define **STARPU\_LONG\_CHECK**
- #define **STARPU\_MAJOR\_VERSION**
- #define **STARPU\_MAXCPUS**
- #define **STARPU\_MAXCUDADEVES**
- #define **STARPU\_MAXHIPDEVES**
- #define **STARPU\_MAXIMPLEMENTATIONS**
- #define **STARPU\_MAXMAXFPGADEVS**
- #define **STARPU\_MAXMPIDEVES**
- #define **STARPU\_MAXNODES**
- #define **STARPU\_MAXNUMANODES**
- #define **STARPU\_MAXOPENCLDEVES**
- #define **STARPU\_MAXTCPIPDEVES**
- #define **STARPU\_MEMORY\_STATS**
- #define **STARPU\_MINOR\_VERSION**
- #define **STARPU\_MKL**
- #define **STARPU\_MLR\_MODEL**
- #define **STARPU\_MODEL\_DEBUG**
- #define **STARPU\_MPI\_EXTRA\_VERBOSE**
- #define **STARPU\_MPI\_PEDANTIC\_ISEND**
- #define **STARPU\_MPI\_VERBOSE**
- #define **STARPU\_NATIVE\_WINTHREADS**
- #define **STARPU\_NEW\_CHECK**
- #define **STARPU\_NMAXBUFS**
- #define **STARPU\_NMAXDEVES**
- #define **STARPU\_NMAXWORKERS**
- #define **STARPU\_NMAX\_COMBINEDWORKERS**
- #define **STARPU\_NMAX\_SCHED\_CTXS**
- #define **STARPU\_NON\_BLOCKING\_DRIVERS**
- #define **STARPU\_NO\_ASSERT**
- #define **STARPU\_OPENBLAS**
- #define **STARPU\_OPENBSD\_SYS**
- #define **STARPU\_OPENCL\_SIMULATOR**
- #define **STARPU\_OPENGL\_RENDER**
- #define **STARPU\_OPENMP**
- #define **STARPU\_OPENMP\_LLVM**
- #define **STARPU\_PAPI**
- #define **STARPU\_PARALLEL\_WORKER**
- #define **STARPU\_PERF\_DEBUG**
- #define **STARPU\_PERF\_MODEL\_DIR**
- #define **STARPU\_PROF\_TOOL**
- #define **STARPU\_PTHREAD\_COND\_INITIALIZER\_ZERO**

- #define **STARPU\_PTHREAD\_MUTEX\_INITIALIZER\_ZERO**
- #define **STARPU\_PTHREAD\_RWLOCK\_INITIALIZER\_ZERO**
- #define **STARPU\_PYTHON\_HAVE\_CLOUDPICKLE**
- #define **STARPU\_PYTHON\_HAVE\_JOBLIB**
- #define **STARPU\_PYTHON\_HAVE\_NUMPY**
- #define **STARPU\_QUICK\_CHECK**
- #define **STARPU\_RELEASE\_VERSION**
- #define **STARPU\_SC\_HYPERVISOR\_DEBUG**
- #define **STARPU\_SIMGRID**
- #define **STARPU\_SIMGRID\_HAVE\_SIMGRID\_INIT**
- #define **STARPU\_SIMGRID\_HAVE\_XBT\_BARRIER\_INIT**
- #define **STARPU\_SIMGRID\_MC**
- #define **STARPU\_SPINLOCK\_CHECK**
- #define **STARPU\_SRC\_DIR**
- #define **STARPU\_STATIC\_ONLY**
- #define **STARPU\_SYSTEM\_BLAS**
- #define **STARPU\_USE\_ALLOCATION\_CACHE**
- #define **STARPU\_USE\_AYUDAME1**
- #define **STARPU\_USE\_AYUDAME2**
- #define **STARPU\_USE\_CPU**
- #define **STARPU\_USE\_CUDA**
- #define **STARPU\_USE\_CUDA0**
- #define **STARPU\_USE\_CUDA1**
- #define **STARPU\_USE\_CUDA\_MAP**
- #define **STARPU\_USE\_DRAND48**
- #define **STARPU\_USE\_ERAND48\_R**
- #define **STARPU\_USE\_FXT**
- #define **STARPU\_USE\_HIP**
- #define **STARPU\_USE\_HIPBLAS**
- #define **STARPU\_USE\_MAX\_FPGA**
- #define **STARPU\_USE\_MP**
- #define **STARPU\_USE\_MPI**
- #define **STARPU\_USE\_MPI\_FT**
- #define **STARPU\_USE\_MPI\_FT\_STATS**
- #define **STARPU\_USE\_MPI\_MASTER\_SLAVE**
- #define **STARPU\_USE\_MPI\_MPI**
- #define **STARPU\_USE\_MPI\_NMAD**
- #define **STARPU\_USE\_OPENCL**
- #define **STARPU\_USE\_SC\_HYPERVISOR**
- #define **STARPU\_USE\_TCPIP\_MASTER\_SLAVE**
- #define **STARPU\_VALGRIND\_FULL**
- #define **STARPU\_VERBOSE**
- #define **STARPU\_WORKER\_CALLBACKS**
- #define **STDC\_HEADERS**
- #define **VERSION**
- #define **X\_DISPLAY\_MISSING**
- #define **restrict**

## 6.8 copy\_driver.h File Reference

```
#include <common/config.h>
#include <common/list.h>
```

## Data Structures

- struct [\\_starpus\\_disk\\_backend\\_event](#)
- struct [\\_starpus\\_disk\\_event](#)
- union [\\_starpus\\_async\\_channel\\_event](#)
- struct [\\_starpus\\_async\\_channel](#)

## Enumerations

- enum [\\_starpus\\_may\\_alloc](#) { [\\_STARPU\\_DATAWIZARD\\_DO\\_NOT\\_ALLOC](#) , [\\_STARPU\\_DATAWIZARD\\_DO\\_ALLOC](#) , [\\_STARPU\\_DATAWIZARD\\_ONLY\\_FAST\\_ALLOC](#) }

## Functions

- void [\\_starpus\\_wake\\_all\\_blocked\\_workers\\_on\\_node](#) (unsigned nodeid)
- int [\\_starpus\\_driver\\_copy\\_data\\_1\\_to\\_1](#) (starpus\_data\_handle\_t handle, [struct\\_starpus\\_data\\_replicate](#) \*src←\_replicate, [struct\\_starpus\\_data\\_replicate](#) \*dst\_replicate, unsigned donotread, [struct\\_starpus\\_data\\_request](#) \*req, enum [\\_starpus\\_may\\_alloc](#) may\_alloc, enum [starpus\\_is\\_prefetch](#) prefetch)
- int [\\_starpus\\_copy\\_interface\\_any\\_to\\_any](#) (starpus\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, [struct\\_starpus\\_data\\_request](#) \*req)
- unsigned [\\_starpus\\_driver\\_test\\_request\\_completion](#) ([struct\\_starpus\\_async\\_channel](#) \*async\_channel)
- void [\\_starpus\\_driver\\_wait\\_request\\_completion](#) ([struct\\_starpus\\_async\\_channel](#) \*async\_channel)

### 6.8.1 Data Structure Documentation

#### 6.8.1.1 struct\_starpus\_disk\_backend\_event

##### Data Fields

void *	backend_event	
--------	---------------	--

#### 6.8.1.2 struct\_starpus\_disk\_event

##### Data Fields

unsigned	memory_node	
unsigned	node	
<a href="#">struct_starpus_disk_backend_event_list</a> *	requests	
void *	ptr	
size_t	size	
starpus_data_handle_t	handle	

#### 6.8.1.3 union\_starpus\_async\_channel\_event

this is a structure that can be queried to see whether an asynchronous transfer has terminated or not

##### Data Fields

char	data[40]	
------	----------	--

#### 6.8.1.4 struct\_starpus\_async\_channel

## Data Fields

union <a href="#">_starpu_async_channel_event</a>	event	
const <a href="#">struct _starpu_node_ops</a> *	node_ops	
<a href="#">struct _starpu_mp_node</a> *	polling_node_sender	Which node to polling when needing ACK msg
<a href="#">struct _starpu_mp_node</a> *	polling_node_receiver	
volatile int	starpu_mp_common_finished_sender	Used to know if the acknowledgment msg is arrived from sinks
volatile int	starpu_mp_common_finished_receiver	

## 6.9 data\_concurrency.h File Reference

```
#include <core/jobs.h>
```

## Functions

- void [\\_starpu\\_job\\_set\\_ordered\\_buffers](#) ([struct \\_starpu\\_job](#) \*j)
- unsigned [\\_starpu\\_concurrent\\_data\\_access](#) ([struct \\_starpu\\_job](#) \*j)
- void [\\_starpu\\_submit\\_job\\_enforce\\_arbitered\\_deps](#) ([struct \\_starpu\\_job](#) \*j, unsigned buf, unsigned nbufers)
- void [\\_starpu\\_submit\\_job\\_take\\_data\\_deps](#) ([struct \\_starpu\\_job](#) \*j)
- void [\\_starpu\\_enforce\\_data\\_deps\\_notify\\_job\\_ready\\_soon](#) ([struct \\_starpu\\_job](#) \*j, [\\_starpu\\_notify\\_job](#) \*start\_data \*data)
- int [\\_starpu\\_notify\\_data\\_dependencies](#) ([starpu\\_data\\_handle\\_t](#) handle, enum [starpu\\_data\\_access\\_mode](#) down\_to\_mode)
- void [\\_starpu\\_notify\\_arbitered\\_dependencies](#) ([starpu\\_data\\_handle\\_t](#) handle, enum [starpu\\_data\\_access\\_mode](#) down\_to\_mode)
- unsigned [\\_starpu\\_attempt\\_to\\_submit\\_data\\_request\\_from\\_apps](#) ([starpu\\_data\\_handle\\_t](#) handle, enum [starpu\\_data\\_access\\_mode](#) mode, void(\*callback)(void \*), void \*argcb)
- unsigned [\\_starpu\\_attempt\\_to\\_submit\\_arbitered\\_data\\_request](#) (unsigned request\_from\_codelet, [starpu\\_data\\_handle\\_t](#) handle, enum [starpu\\_data\\_access\\_mode](#) mode, void(\*callback)(void \*), void \*argcb, [struct \\_starpu\\_job](#) \*j, unsigned buffer\_index)

## 6.10 data\_interface.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/uthash.h>
#include <util/openmp_runtime_support.h>
```

## Data Structures

- [union \\_starpu\\_interface](#)

## Macros

- [#define \\_starpu\\_data\\_check\\_not\\_busy](#)(handle)
- [#define \\_starpu\\_data\\_is\\_multiformat\\_handle](#)(handle)

## Functions

- void **\_starpu\_data\_free\_interfaces** (starpu\_data\_handle\_t handle)
- int **\_starpu\_data\_handle\_init** (starpu\_data\_handle\_t handle, struct starpu\_data\_interface\_ops \*interface\_ops, unsigned int mf\_node)
- void **\_starpu\_data\_initialize\_per\_worker** (starpu\_data\_handle\_t handle)
- void **\_starpu\_data\_interface\_init** (void)
- int **\_\_starpu\_data\_check\_not\_busy** (starpu\_data\_handle\_t handle) STARPU\_ATTRIBUTE\_WARN\_UNUSED\_RESULT
- void **\_starpu\_data\_interface\_shutdown** (void)
- struct starpu\_data\_interface\_ops \* **\_starpu\_data\_interface\_get\_ops** (unsigned interface\_id) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- void **\_starpu\_data\_deinitialize\_submit\_noplan** (starpu\_data\_handle\_t handle)
- void **\_starpu\_data\_invalidate\_submit\_noplan** (starpu\_data\_handle\_t handle)

## Variables

- struct starpu\_data\_interface\_ops [starpu\\_interface\\_multiformat\\_ops](#)
- struct starpu\_arbiter \* **\_starpu\_global\_arbiter**

### 6.10.1 Data Structure Documentation

#### 6.10.1.1 union\_starpu\_interface

Generic type representing an interface, for now it's only used before execution on message-passing devices but it can be useful in other cases.

##### Data Fields

<a href="#">struct</a> starpu_variable_interface	variable	
<a href="#">struct</a> starpu_vector_interface	vector	
<a href="#">struct</a> starpu_matrix_interface	matrix	
<a href="#">struct</a> starpu_block_interface	block	
<a href="#">struct</a> starpu_tensor_interface	tensor	
<a href="#">struct</a> starpu_csr_interface	csr	
<a href="#">struct</a> starpu_bcsr_interface	bcsr	
<a href="#">struct</a> starpu_coo_interface	coo	

### 6.10.2 Variable Documentation

#### 6.10.2.1 starpu\_interface\_multiformat\_ops

```
struct starpu_data_interface_ops starpu_interface_multiformat_ops [extern]
```

Some data interfaces or filters use this interface internally

## 6.11 data\_request.h File Reference

```
#include <datawizard/coherency.h>
#include <semaphore.h>
#include <datawizard/copy_driver.h>
#include <common/list.h>
#include <common/prio_list.h>
```

```
#include <common/starpu_spinlock.h>
```

## Data Structures

- struct [\\_starpu\\_callback\\_list](#)

## Macros

- #define **MAX\_PENDING\_REQUESTS\_PER\_NODE**
- #define **MAX\_PENDING\_PREFETCH\_REQUESTS\_PER\_NODE**
- #define **MAX\_PENDING\_IDLE\_REQUESTS\_PER\_NODE**
- #define [MAX\\_PUSH\\_TIME](#)

## Enumerations

- enum **\_starpu\_data\_request\_inout** { **\_STARPU\_DATA\_REQUEST\_IN** , **\_STARPU\_DATA\_REQUEST\_OUT** }

### 6.11.1 Macro Definition Documentation

#### 6.11.1.1 MAX\_PUSH\_TIME

```
#define MAX_PUSH_TIME
```

Maximum time in us that we can afford pushing requests before going back to the driver loop, e.g. for checking GPU task termination

## 6.12 datastats.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <stdint.h>
#include <stdlib.h>
```

## Macros

- #define **\_starpu\_msi\_cache\_hit**(node)
- #define **\_starpu\_msi\_cache\_miss**(node)
- #define **\_starpu\_allocation\_cache\_hit**(node)
- #define **\_starpu\_data\_allocation\_inc\_stats**(node)

## Functions

- void **\_starpu\_datastats\_init** ()
- static int **starpu\_enable\_stats** (void)
- void **\_\_starpu\_msi\_cache\_hit** (unsigned node)
- void **\_\_starpu\_msi\_cache\_miss** (unsigned node)
- void **\_starpu\_display\_msi\_stats** (FILE \*stream)
- void **\_\_starpu\_allocation\_cache\_hit** (unsigned node STARPU\_ATTRIBUTE\_UNUSED)
- void **\_\_starpu\_data\_allocation\_inc\_stats** (unsigned node STARPU\_ATTRIBUTE\_UNUSED)
- void **\_starpu\_display\_alloc\_cache\_stats** (FILE \*stream)

## Variables

- `int __starpu_enable_stats`

## 6.13 datawizard.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/utils.h>
#include <datawizard/coherency.h>
#include <datawizard/filters.h>
#include <datawizard/copy_driver.h>
#include <datawizard/footprint.h>
#include <datawizard/data_request.h>
#include <datawizard/interfaces/data_interface.h>
#include <core/dependencies/implicit_data_deps.h>
```

## Functions

- `int __starpu_datawizard_progress` (enum `__starpu_may_alloc may_alloc`, unsigned `push_requests`)
- `void __starpu_datawizard_progress` (enum `__starpu_may_alloc may_alloc`)
- `void __starpu_datawizard_handle_all_pending_node_data_requests` (unsigned `memnode`)

### 6.13.1 Function Documentation

#### 6.13.1.1 `__starpu_datawizard_progress()`

```
int __starpu_datawizard_progress (
    enum __starpu_may_alloc may_alloc,
    unsigned push_requests )
```

Make data transfers progress on all memory nodes driven by the current worker.

If `push_requests` is 1, it can start new transfers

If `may_alloc` is `_STARPU_DATAWIZARD_DO_ALLOC`, it can allocate destination data for transfers (this is not possible e.g. when spinning for a handle lock)

#### 6.13.1.2 `__starpu_datawizard_progress()`

```
void __starpu_datawizard_progress (
    enum __starpu_may_alloc may_alloc )
```

Call `__starpu_datawizard_progress` with `push_requests = 1`

#### 6.13.1.3 `__starpu_datawizard_handle_all_pending_node_data_requests()`

```
void __starpu_datawizard_handle_all_pending_node_data_requests (
    unsigned memnode )
```

Check for all pending data request progress on node `memory_node`

## 6.14 debug.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <common/config.h>
#include <core/workers.h>
```

## Macros

- #define **STARPU\_AYU\_EVENT**
- #define **STARPU\_AYU\_PREINIT()**
- #define **STARPU\_AYU\_INIT()**
- #define **STARPU\_AYU\_FINISH()**
- #define **STARPU\_AYU\_ADDDEPENDENCY**(previous, handle, next\_job)
- #define **STARPU\_AYU\_REMOVETASK**(job\_id)
- #define **STARPU\_AYU\_ADDTASK**(job\_id, task)
- #define **STARPU\_AYU\_PRERUNTASK**(job\_id, workerid)
- #define **STARPU\_AYU\_RUNTASK**(job\_id)
- #define **STARPU\_AYU\_POSTRUNTASK**(job\_id)
- #define **STARPU\_AYU\_ADDTOTASKQUEUE**(job\_id, worker\_id)
- #define **STARPU\_AYU\_BARRIER()**

## Functions

- void [\\_starpu\\_open\\_debug\\_logfile](#) (void)
- void [\\_starpu\\_close\\_debug\\_logfile](#) (void)
- void [\\_starpu\\_print\\_to\\_logfile](#) (const char \*format,...) STARPU\_ATTRIBUTE\_FORMAT printf
- void [\\_starpu\\_watchdog\\_init](#) (void)
- void [\\_starpu\\_watchdog\\_shutdown](#) (void)

## Variables

- void int [\\_starpu\\_use\\_fxt](#)

### 6.14.1 Function Documentation

#### 6.14.1.1 [\\_starpu\\_open\\_debug\\_logfile\(\)](#)

```
void _starpu_open_debug_logfile (
    void )
```

Create a file that will contain StarPU's log

#### 6.14.1.2 [\\_starpu\\_close\\_debug\\_logfile\(\)](#)

```
void _starpu_close_debug_logfile (
    void )
```

Close StarPU's log file

#### 6.14.1.3 [\\_starpu\\_print\\_to\\_logfile\(\)](#)

```
void _starpu_print_to_logfile (
    const char * format,
    ... )
```

Write into StarPU's log file

### 6.14.2 Variable Documentation

#### 6.14.2.1 [\\_starpu\\_use\\_fxt](#)

```
void int _starpu_use_fxt [extern]
```

Tell gdb whether FXT is compiled in or not

## 6.15 detect\_combined\_workers.h File Reference

```
#include <starpu.h>
```

### Functions

- void [\\_starpu\\_sched\\_find\\_worker\\_combinations](#) (int \*workerids, int nworkers)

### Variables

- int [\\_starpu\\_initialized\\_combined\\_workers](#)

### 6.15.1 Function Documentation

#### 6.15.1.1 \_starpu\_sched\_find\_worker\_combinations()

```
void _starpu_sched_find_worker_combinations (
    int * workerids,
    int nworkers )
```

Initialize combined workers

## 6.16 disk.h File Reference

```
#include <datawizard/copy_driver.h>
#include <datawizard/malloc.h>
```

### Macros

- #define [STARPU\\_DISK\\_ALL](#)
- #define [STARPU\\_DISK\\_NO\\_RECLAIM](#)

### Functions

- void \* [\\_starpu\\_disk\\_alloc](#) (unsigned node, size\_t size) STARPU\_ATTRIBUTE\_MALLOC
- void [\\_starpu\\_disk\\_free](#) (unsigned node, void \*obj, size\_t size)
- int [\\_starpu\\_disk\\_read](#) (unsigned src\_node, unsigned dst\_node, void \*obj, void \*buf, off\_t offset, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- int [\\_starpu\\_disk\\_write](#) (unsigned src\_node, unsigned dst\_node, void \*obj, void \*buf, off\_t offset, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- int [\\_starpu\\_disk\\_full\\_read](#) (unsigned src\_node, unsigned dst\_node, void \*obj, void \*\*ptr, size\_t \*size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- int [\\_starpu\\_disk\\_full\\_write](#) (unsigned src\_node, unsigned dst\_node, void \*obj, void \*ptr, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- int [\\_starpu\\_disk\\_copy](#) (unsigned node\_src, void \*obj\_src, off\_t offset\_src, unsigned node\_dst, void \*obj\_dst, off\_t offset\_dst, size\_t size, [struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- void [starpu\\_disk\\_wait\\_request](#) ([struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- int [starpu\\_disk\\_test\\_request](#) ([struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- void [starpu\\_disk\\_free\\_request](#) ([struct\\_starpu\\_async\\_channel](#) \*async\_channel)
- int [\\_starpu\\_disk\\_can\\_copy](#) (unsigned node1, unsigned node2)
- void [\\_starpu\\_set\\_disk\\_flag](#) (unsigned node, int flag)
- int [\\_starpu\\_get\\_disk\\_flag](#) (unsigned node)
- void [\\_starpu\\_disk\\_unregister](#) (void)
- void [\\_starpu\\_swap\\_init](#) (void)
- static [struct\\_starpu\\_disk\\_event](#) \* [\\_starpu\\_disk\\_get\\_event](#) (union [\\_starpu\\_async\\_channel\\_event](#) \*\_event)

## 6.16.1 Function Documentation

### 6.16.1.1 `_starpu_disk_alloc()`

```
void * _starpu_disk_alloc (
    unsigned node,
    size_t size )
```

interface to manipulate memory disk

### 6.16.1.2 `_starpu_disk_read()`

```
int _starpu_disk_read (
    unsigned src_node,
    unsigned dst_node,
    void * obj,
    void * buf,
    off_t offset,
    size_t size,
    struct _starpu_async_channel * async_channel )
```

`src_node` is a disk node, `dst_node` is for the moment the STARPU\_MAIN\_RAM

### 6.16.1.3 `_starpu_disk_write()`

```
int _starpu_disk_write (
    unsigned src_node,
    unsigned dst_node,
    void * obj,
    void * buf,
    off_t offset,
    size_t size,
    struct _starpu_async_channel * async_channel )
```

`src_node` is for the moment the STARU\_MAIN\_RAM, `dst_node` is a disk node

### 6.16.1.4 `starpu_disk_wait_request()`

```
void starpu_disk_wait_request (
    struct _starpu_async_channel * async_channel )
```

force the request to compute

### 6.16.1.5 `starpu_disk_test_request()`

```
int starpu_disk_test_request (
    struct _starpu_async_channel * async_channel )
```

return 1 if the request is finished, 0 if not finished

### 6.16.1.6 `_starpu_disk_can_copy()`

```
int _starpu_disk_can_copy (
    unsigned node1,
    unsigned node2 )
```

interface to compare memory disk

### 6.16.1.7 `_starpu_set_disk_flag()`

```
void _starpu_set_disk_flag (
    unsigned node,
    int flag )
```

change disk flag

### 6.16.1.8 \_starpu\_disk\_unregister()

```
void _starpu_disk_unregister (
    void )
unregister disk
```

## 6.17 disk\_unistd\_global.h File Reference

```
#include <fcntl.h>
```

### Data Structures

- struct [starpu\\_unistd\\_global\\_obj](#)

### Macros

- #define **O\_BINARY**
- #define **STARPU\_UNISTD\_USE\_COPY**

### Typedefs

- typedef off\_t **starpu\_loff\_t**

### Functions

- void \* **starpu\_unistd\_global\_alloc** (struct [starpu\\_unistd\\_global\\_obj](#) \*obj, void \*base, size\_t size)
- void **starpu\_unistd\_global\_free** (void \*base, void \*obj, size\_t size)
- void \* **starpu\_unistd\_global\_open** (struct [starpu\\_unistd\\_global\\_obj](#) \*obj, void \*base, void \*pos, size\_t size)
- void **starpu\_unistd\_global\_close** (void \*base, void \*obj, size\_t size)
- int **starpu\_unistd\_global\_read** (void \*base, void \*obj, void \*buf, off\_t offset, size\_t size)
- int **starpu\_unistd\_global\_write** (void \*base, void \*obj, const void \*buf, off\_t offset, size\_t size)
- void \* **starpu\_unistd\_global\_plug** (void \*parameter, starpu\_ssize\_t size)
- void **starpu\_unistd\_global\_unplug** (void \*base)
- int **\_starpu\_get\_unistd\_global\_bandwidth\_between\_disk\_and\_main\_ram** (unsigned node, void \*base)
- void \* **starpu\_unistd\_global\_async\_read** (void \*base, void \*obj, void \*buf, off\_t offset, size\_t size)
- void \* **starpu\_unistd\_global\_async\_write** (void \*base, void \*obj, void \*buf, off\_t offset, size\_t size)
- void \* **starpu\_unistd\_global\_async\_full\_write** (void \*base, void \*obj, void \*ptr, size\_t size)
- void \* **starpu\_unistd\_global\_async\_full\_read** (void \*base, void \*obj, void \*\*ptr, size\_t \*size, unsigned dst\_node)
- void **starpu\_unistd\_global\_wait\_request** (void \*async\_channel)
- int **starpu\_unistd\_global\_test\_request** (void \*async\_channel)
- void **starpu\_unistd\_global\_free\_request** (void \*async\_channel)
- int **starpu\_unistd\_global\_full\_read** (void \*base, void \*obj, void \*\*ptr, size\_t \*size, unsigned dst\_node)
- int **starpu\_unistd\_global\_full\_write** (void \*base, void \*obj, void \*ptr, size\_t size)

### 6.17.1 Data Structure Documentation

#### 6.17.1.1 struct [starpu\\_unistd\\_global\\_obj](#)

##### Data Fields

int	descriptor	
char *	path	
size_t	size	

## Data Fields

	int	flags	
starpu_pthread_mutex_t		mutex	

## 6.18 driver\_common.h File Reference

```
#include <starpu.h>
#include <starpu_util.h>
#include <core/jobs.h>
#include <common/utills.h>
```

### Functions

- void `_starpu_driver_start_job` (struct `_starpu_worker` \*args, struct `_starpu_job` \*j, struct `starpu_perfmodel_arch` \*perf\_arch, int rank, int profiling)
- void `_starpu_driver_end_job` (struct `_starpu_worker` \*args, struct `_starpu_job` \*j, struct `starpu_perfmodel_arch` \*perf\_arch, int rank, int profiling)
- void `_starpu_driver_update_job_feedback` (struct `_starpu_job` \*j, struct `_starpu_worker` \*worker\_args, struct `starpu_perfmodel_arch` \*perf\_arch, int profiling)
- struct `starpu_task` \* `_starpu_get_worker_task` (struct `_starpu_worker` \*args, int workerid, unsigned memnode)
- int `_starpu_get_multi_worker_task` (struct `_starpu_worker` \*workers, struct `starpu_task` \*\*tasks, int nworker, unsigned memnode)
- void \* `_starpu_map_allocate` (size\_t length, unsigned node)
- int `_starpu_map_deallocate` (void \*map\_addr, size\_t length)
- char \* `_starpu_get_fdname_from_mapaddr` (uintptr\_t map\_addr, size\_t \*offset, size\_t length)
- void \* `_starpu_sink_map` (char \*fd\_name, size\_t offset, size\_t length)
- int `_starpu_sink_unmap` (uintptr\_t map\_addr, size\_t length)

### 6.18.1 Function Documentation

#### 6.18.1.1 `_starpu_driver_start_job()`

```
void _starpu_driver_start_job (
    struct _starpu_worker * args,
    struct _starpu_job * j,
    struct starpu_perfmodel_arch * perf_arch,
    int rank,
    int profiling )
```

The task job is about to start (or has already started when kernels are queued in a pipeline), record profiling and trace information.

#### 6.18.1.2 `_starpu_driver_end_job()`

```
void _starpu_driver_end_job (
    struct _starpu_worker * args,
    struct _starpu_job * j,
    struct starpu_perfmodel_arch * perf_arch,
    int rank,
    int profiling )
```

The task job has ended, record profiling and trace information.

### 6.18.1.3 `_starpu_driver_update_job_feedback()`

```
void _starpu_driver_update_job_feedback (
    struct _starpu_job * j,
    struct _starpu_worker * worker_args,
    struct starpu_perfmodel_arch * perf_arch,
    int profiling )
```

Feed performance model with the terminated job statistics

### 6.18.1.4 `_starpu_get_worker_task()`

```
struct starpu_task * _starpu_get_worker_task (
    struct _starpu_worker * args,
    int workerid,
    unsigned memnode )
```

Get from the scheduler a task to be executed on the worker `workerid`

### 6.18.1.5 `_starpu_get_multi_worker_task()`

```
int _starpu_get_multi_worker_task (
    struct _starpu_worker * workers,
    struct starpu_task ** tasks,
    int nworker,
    unsigned memnode )
```

Get from the scheduler tasks to be executed on the workers `workers`

## 6.19 driver\_cpu.h File Reference

```
#include <core/workers.h>
#include <common/config.h>
```

### Functions

- void `_starpu_cpu_preinit` (void)
- void `_starpu_cpu_busy_cpu` (unsigned num)
- void `_starpu_init_cpu_config` (`struct _starpu_machine_topology` \*topology, `struct _starpu_machine_config` \*config)

### Variables

- `struct starpu_driver_ops` `_starpu_driver_cpu_ops`

## 6.20 driver\_cuda.h File Reference

```
#include <common/config.h>
#include <cuda.h>
#include <cuda_runtime_api.h>
#include <nvml.h>
#include <starpu.h>
#include <core/workers.h>
#include <datawizard/node_ops.h>
```

### Functions

- void `_starpu_cuda_preinit` (void)

- void `_starpu_cuda_init` (void)
- unsigned `_starpu_get_cuda_device_count` (void)
- `hwloc_obj_t` `_starpu_cuda_get_hwloc_obj` (`hwloc_topology_t` topology, int devid)
- void `_starpu_cuda_discover_devices` (`struct_starpu_machine_config *`)
- void `_starpu_init_cuda_config` (`struct_starpu_machine_topology *`topology, `struct_starpu_machine_config *`)
- void `_starpu_cuda_init_worker_binding` (`struct_starpu_machine_config *`config, int no\_mp\_config, `struct_starpu_worker *`workerarg)
- void `_starpu_cuda_init_worker_memory` (`struct_starpu_machine_config *`config, int no\_mp\_config, `struct_starpu_worker *`workerarg)
- void `_starpu_init_cuda` (void)
- void `_starpu_init_cublas_v2_func` (void)
- void `_starpu_shutdown_cublas_v2_func` (void)
- void `_starpu_cublas_v2_init` (void)
- void `_starpu_cublas_v2_shutdown` (void)
- void \* `_starpu_cuda_worker` (void \*)
- `nvmlDevice_t` `_starpu_cuda_get_nvmldev` (`struct_cudaDeviceProp *`props)
- `__typeof__` (nvmlInit) \* `_starpu_nvmlInit`
- `__typeof__` (nvmlDeviceGetNvLinkState) \* `_starpu_nvmlDeviceGetNvLinkState`
- `__typeof__` (nvmlDeviceGetNvLinkRemotePciInfo) \* `_starpu_nvmlDeviceGetNvLinkRemotePciInfo`
- `__typeof__` (nvmlDeviceGetHandleByIndex) \* `_starpu_nvmlDeviceGetHandleByIndex`
- `__typeof__` (nvmlDeviceGetHandleByPciBusId) \* `_starpu_nvmlDeviceGetHandleByPciBusId`
- `__typeof__` (nvmlDeviceGetIndex) \* `_starpu_nvmlDeviceGetIndex`
- `__typeof__` (nvmlDeviceGetPciInfo) \* `_starpu_nvmlDeviceGetPciInfo`
- `__typeof__` (nvmlDeviceGetUUID) \* `_starpu_nvmlDeviceGetUUID`

## Variables

- `struct_starpu_driver_ops` `_starpu_driver_cuda_ops`
- `struct_starpu_node_ops` `_starpu_driver_cuda_node_ops`
- int `_starpu_nworker_per_cuda`
- int `_starpu_cuda_bus_ids` [`STARPU_MAXCUDADEVS+STARPU_MAXNUMANODES`][`STARPU_MAXCUDADEVS+STARPU_MAXNUMANODES`]

## 6.21 driver\_disk.h File Reference

### Functions

- void `_starpu_disk_preinit` (void)

## 6.22 driver\_mpi\_common.h File Reference

```
#include <drivers/mp_common/mp_common.h>
#include <drivers/mpi/driver_mpi_source.h>
```

### Macros

- `#define SYNC_TAG`
- `#define ASYNC_TAG`
- `#define NOTIF_TAG`

## Functions

- int `_starpu_mpi_common_mp_init` ()
- void `_starpu_mpi_common_mp_deinit` ()
- int `_starpu_mpi_common_is_src_node` ()
- int `_starpu_mpi_common_get_src_node` ()
- int `_starpu_mpi_common_is_mp_initialized` ()
- int `_starpu_mpi_common_recv_is_ready` (const [struct](#) `_starpu_mp_node` \*mp\_node)
- int `_starpu_mpi_common_notif_recv_is_ready` (const [struct](#) `_starpu_mp_node` \*mp\_node)
- int `_starpu_mpi_common_notif_send_is_ready` (const [struct](#) `_starpu_mp_node` \*mp\_node)
- void `_starpu_mpi_common_mp_initialize_src_sink` ([struct](#) `_starpu_mp_node` \*node)
- void `_starpu_mpi_common_send` (const [struct](#) `_starpu_mp_node` \*node, void \*msg, int len, void \*event)
- void `_starpu_mpi_common_recv` (const [struct](#) `_starpu_mp_node` \*node, void \*msg, int len, void \*event)
- void `_starpu_mpi_common_mp_send` (const [struct](#) `_starpu_mp_node` \*node, void \*msg, int len)
- void `_starpu_mpi_common_mp_recv` (const [struct](#) `_starpu_mp_node` \*node, void \*msg, int len)
- void `_starpu_mpi_common_nt_send` (const [struct](#) `_starpu_mp_node` \*node, void \*msg, int len)
- void `_starpu_mpi_common_nt_recv` (const [struct](#) `_starpu_mp_node` \*node, void \*msg, int len)
- void `_starpu_mpi_common_recv_from_device` (const [struct](#) `_starpu_mp_node` \*node, int src\_devid, void \*msg, int len, void \*event)
- void `_starpu_mpi_common_send_to_device` (const [struct](#) `_starpu_mp_node` \*node, int dst\_devid, void \*msg, int len, void \*event)
- unsigned int `_starpu_mpi_common_test_event` ([struct](#) `_starpu_async_channel` \*event)
- void `_starpu_mpi_common_wait_request_completion` ([struct](#) `_starpu_async_channel` \*event)
- void `_starpu_mpi_common_barrier` (void)
- void `_starpu_mpi_common_measure_bandwidth_latency` (double bandwidth\_dtod[STARPU\_↔  
MAXMPIDEVS][STARPU\_MAXMPIDEVS], double latency\_dtod[STARPU\_↔  
MAXMPIDEVS])

## Variables

- int `_starpu_mpi_common_multiple_thread`

## 6.23 driver\_mpi\_sink.h File Reference

```
#include <drivers/mp_common/sink_common.h>
```

## Functions

- void `_starpu_mpi_sink_init` ([struct](#) `_starpu_mp_node` \*node)
- void `_starpu_mpi_sink_bind_thread` (const [struct](#) `_starpu_mp_node` \*mp\_node STARPU\_ATTRIBUTE↔  
\_UNUSED, int coreid, int \*core\_table, int nb\_core)

## 6.24 driver\_mpi\_source.h File Reference

```
#include <core/workers.h>
#include <drivers/mp_common/mp_common.h>
#include <datawizard/node_ops.h>
```

## Functions

- void `_starpu_mpi_ms_preinit` (void)
- `struct _starpu_mp_node * _starpu_mpi_ms_src_get_actual_thread_mp_node` ()
- unsigned `_starpu_mpi_src_get_device_count` ()
- void \* `_starpu_mpi_src_worker` (void \*arg)
- void `_starpu_init_mpi_config` (`struct _starpu_machine_topology` \*topology, `struct _starpu_machine_config` \*config, `struct starpu_conf` \*user\_conf, int no\_mp\_config)
- void `_starpu_mpi_init_worker_binding` (`struct _starpu_machine_config` \*config, int no\_mp\_config STARPU\_ATTRIBUTE\_UNUSED, `struct _starpu_worker` \*workerarg)
- void `_starpu_mpi_init_worker_memory` (`struct _starpu_machine_config` \*config, int no\_mp\_config STARPU\_ATTRIBUTE\_UNUSED, `struct _starpu_worker` \*workerarg)
- void `_starpu_deinit_mpi_config` (`struct _starpu_machine_config` \*config)
- void `_starpu_mpi_source_init` (`struct _starpu_mp_node` \*node)
- void `_starpu_mpi_source_deinit` (`struct _starpu_mp_node` \*node)

## Variables

- `struct _starpu_node_ops _starpu_driver_mpi_ms_node_ops`

### 6.24.1 Function Documentation

#### 6.24.1.1 `_starpu_mpi_ms_src_get_actual_thread_mp_node()`

```
struct _starpu_mp_node * _starpu_mpi_ms_src_get_actual_thread_mp_node ( )
```

Array of structures containing all the information useful to send and receive information with devices

## 6.25 `driver_opencl.h` File Reference

```
#include <CL/cl.h>
#include <core/workers.h>
#include <datawizard/node_ops.h>
```

## Macros

- `#define _GNU_SOURCE`
- `#define CL_TARGET_OPENCL_VERSION`

## Functions

- void `_starpu_opencl_preinit` (void)
- void `_starpu_opencl_discover_devices` (`struct _starpu_machine_config` \*config)
- void `_starpu_opencl_init` (void)
- int `_starpu_opencl_init_context` (int devid)
- int `_starpu_opencl_deinit_context` (int devid)
- unsigned `_starpu_opencl_get_device_count` (void)
- `hwloc_obj_t _starpu_opencl_get_hwloc_obj` (`hwloc_topology_t` topology, int devid)
- void `_starpu_init_opencl_config` (`struct _starpu_machine_topology` \*topology, `struct _starpu_machine_config` \*)
- void `_starpu_opencl_init_worker_binding` (`struct _starpu_machine_config` \*config, int no\_mp\_config STARPU\_ATTRIBUTE\_UNUSED, `struct _starpu_worker` \*workerarg)
- void `_starpu_opencl_init_worker_memory` (`struct _starpu_machine_config` \*config, int no\_mp\_config STARPU\_ATTRIBUTE\_UNUSED, `struct _starpu_worker` \*workerarg)
- void \* `_starpu_opencl_worker` (void \*)
- `cl_device_type _starpu_opencl_get_device_type` (int devid)

## Variables

- [struct \\_starp1\\_node\\_ops](#) `_starp1_driver_openc1_node_ops`
- [struct \\_starp1\\_driver\\_ops](#) `_starp1_driver_openc1_ops`
- `char * _starp1_openc1_program_dir`

## 6.26 driver\_openc1\_utils.h File Reference

### Macros

- `#define _STARPU_OPENC1_PLATFORM_MAX`

### Functions

- `char * _starp1_openc1_get_device_type_as_string (int id)`

## 6.27 drivers.h File Reference

### Data Structures

- [struct \\_starp1\\_driver\\_ops](#)

## 6.28 errorcheck.h File Reference

```
#include <starp1.h>
```

### Enumerations

- `enum _starp1_worker_status_index` {  
`STATUS_INDEX_INITIALIZING` , `STATUS_INDEX_EXECUTING` , `STATUS_INDEX_CALLBACK` ,  
`STATUS_INDEX_WAITING` ,  
`STATUS_INDEX_SLEEPING` , `STATUS_INDEX_SCHEDULING` , `STATUS_INDEX_NR` }
- `enum _starp1_worker_status` {  
`STATUS_INVALID` , `STATUS_UNKNOWN` , `STATUS_INITIALIZING` , `STATUS_EXECUTING` ,  
`STATUS_CALLBACK` , `STATUS_WAITING` , `STATUS_SLEEPING` , `STATUS_SCHEDULING` }

### Functions

- `void _starp1_add_worker_status (struct _starp1_worker *worker, enum _starp1_worker_status_index st, struct timespec *time)`
- `void _starp1_add_local_worker_status (enum _starp1_worker_status_index st, struct timespec *time)`
- `void _starp1_clear_worker_status (struct _starp1_worker *worker, enum _starp1_worker_status_index st, struct timespec *time)`
- `void _starp1_clear_local_worker_status (enum _starp1_worker_status_index st, struct timespec *time)`
- `enum _starp1_worker_status _starp1_get_local_worker_status (void)`
- `unsigned _starp1_worker_may_perform_blocking_calls (void)`

### 6.28.1 Enumeration Type Documentation

#### 6.28.1.1 \_starp1\_worker\_status\_index

```
enum _starp1_worker_status_index
```

This type enumerates the actions that can be done by a worker. Some can be happening during others, that is why `enum _starp1_worker_status` is a bitset indexed by the values of `enum _starp1_worker_status_index`.

### 6.28.1.2 `_starpu_worker_status`

enum `_starpu_worker_status`

This type describes in which state a worker may be.

Enumerator

<code>STATUS_INVALID</code>	invalid status (for instance if we request the status of some thread that is not controlled by StarPU)
<code>STATUS_UNKNOWN</code>	Nothing particular, thus just overhead
<code>STATUS_INITIALIZING</code>	during the initialization
<code>STATUS_EXECUTING</code>	during the execution of a codelet
<code>STATUS_CALLBACK</code>	during the execution of the callback
<code>STATUS_WAITING</code>	while waiting for a data transfer
<code>STATUS_SLEEPING</code>	while sleeping because there is no task to do
<code>STATUS_SCHEDULING</code>	while executing the scheduler code

## 6.28.2 Function Documentation

### 6.28.2.1 `_starpu_add_worker_status()`

```
void _starpu_add_worker_status (
    struct _starpu_worker * worker,
    enum _starpu_worker_status_index st,
    struct timespec * time )
```

Specify what the local worker is currently doing (eg. executing a callback). This permits to detect if this is legal to do a blocking call for instance.

### 6.28.2.2 `_starpu_clear_worker_status()`

```
void _starpu_clear_worker_status (
    struct _starpu_worker * worker,
    enum _starpu_worker_status_index st,
    struct timespec * time )
```

Clear the fact that the local worker was currently doing something(eg. executing a callback).

### 6.28.2.3 `_starpu_get_local_worker_status()`

```
enum _starpu_worker_status _starpu_get_local_worker_status (
    void )
```

Indicate what type of operation the worker is currently doing.

### 6.28.2.4 `_starpu_worker_may_perform_blocking_calls()`

```
unsigned _starpu_worker_may_perform_blocking_calls (
    void )
```

It is forbidden to do blocking calls during some operations such as callback or during the execution of a task. This function indicates whether it is legal to call a blocking operation in the current context.

## 6.29 `fifo_queues.h` File Reference

```
#include <core/task.h>
```

## Data Structures

- struct [starpu\\_st\\_fifo\\_taskq](#)

### 6.29.1 Data Structure Documentation

#### 6.29.1.1 struct starpu\_st\_fifo\_taskq

##### Data Fields

<a href="#">struct</a> starpu_task_list	taskq	the actual list
unsigned	ntasks	the number of tasks currently in the queue
unsigned	pipeline_ntasks	the number of tasks already pushed to the worker
unsigned *	ntasks_per_priority	the number of tasks currently in the queue corresponding to each priority
unsigned	nprocessed	the number of tasks that were processed
double	exp_start	only meaningful if the queue is only used by a single worker
double	exp_end	Expected start date of next item to do in the queue (i.e. not started yet). This is thus updated when we start it.
double	exp_len	Expected end date of last task in the queue
double *	exp_len_per_priority	Expected duration of the set of tasks in the queue
double	pipeline_len	Expected duration of the set of tasks in the queue corresponding to each priority

## 6.30 filters.h File Reference

```
#include <stdarg.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
#include <starpu.h>
#include <common/config.h>
```

### Functions

- void [\\_starpu\\_data\\_partition\\_access\\_submit](#) (starpu\_data\_handle\_t target, int write)

#### 6.30.1 Function Documentation

##### 6.30.1.1 \_starpu\_data\_partition\_access\_submit()

```
void _starpu_data_partition_access_submit (
    starpu_data_handle_t target,
    int write )
```

submit asynchronous unpartitioning / partitioning to make target active read-only or read-write

## 6.31 footprint.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <core/jobs.h>
```

## Functions

- `uint32_t _starpu_compute_buffers_footprint (struct starpu_perfmodel *model, struct starpu_perfmodel_arch *arch, unsigned nimpl, struct _starpu_job *j)`
- `uint32_t _starpu_compute_data_footprint (starpu_data_handle_t handle)`
- `uint32_t _starpu_compute_data_alloc_footprint (starpu_data_handle_t handle)`

### 6.31.1 Function Documentation

#### 6.31.1.1 \_starpu\_compute\_buffers\_footprint()

```
uint32_t _starpu_compute_buffers_footprint (
    struct starpu_perfmodel * model,
    struct starpu_perfmodel_arch * arch,
    unsigned nimpl,
    struct _starpu_job * j )
```

Compute the footprint that characterizes the job and cache it into the job structure.

#### 6.31.1.2 \_starpu\_compute\_data\_footprint()

```
uint32_t _starpu_compute_data_footprint (
    starpu_data_handle_t handle )
```

Compute the footprint that characterizes the layout of the data handle.

#### 6.31.1.3 \_starpu\_compute\_data\_alloc\_footprint()

```
uint32_t _starpu_compute_data_alloc_footprint (
    starpu_data_handle_t handle )
```

Compute the footprint that characterizes the allocation of the data handle.

## 6.32 fxt.h File Reference

```
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <common/config.h>
#include <common/utils.h>
#include <starpu.h>
```

## Macros

- `#define _GNU_SOURCE`
- `#define _STARPU_FUT_WORKER_KEY(kind)`
- `#define _STARPU_FUT_KEY_WORKER(key)`
- `#define _STARPU_FUT_WORKER_INIT_START`
- `#define _STARPU_FUT_WORKER_INIT_END`
- `#define _STARPU_FUT_START_CODELET_BODY`
- `#define _STARPU_FUT_END_CODELET_BODY`
- `#define _STARPU_FUT_JOB_PUSH`
- `#define _STARPU_FUT_JOB_POP`
- `#define _STARPU_FUT_UPDATE_TASK_CNT`
- `#define _STARPU_FUT_START_FETCH_INPUT_ON_TID`
- `#define _STARPU_FUT_END_FETCH_INPUT_ON_TID`
- `#define _STARPU_FUT_START_PUSH_OUTPUT_ON_TID`

- #define `_STARPU_FUT_END_PUSH_OUTPUT_ON_TID`
- #define `_STARPU_FUT_TAG`
- #define `_STARPU_FUT_TAG_DEPS`
- #define `_STARPU_FUT_TASK_DEPS`
- #define `_STARPU_FUT_DATA_COPY`
- #define `_STARPU_FUT_WORK_STEALING`
- #define `_STARPU_FUT_WORKER_DEINIT_START`
- #define `_STARPU_FUT_WORKER_DEINIT_END`
- #define `_STARPU_FUT_WORKER_SLEEP_START`
- #define `_STARPU_FUT_WORKER_SLEEP_END`
- #define `_STARPU_FUT_TASK_SUBMIT`
- #define `_STARPU_FUT_CODELET_DATA_HANDLE`
- #define `_STARPU_FUT_MODEL_NAME`
- #define `_STARPU_FUT_DATA_NAME`
- #define `_STARPU_FUT_DATA_COORDINATES`
- #define `_STARPU_FUT_HANDLE_DATA_UNREGISTER`
- #define `_STARPU_FUT_CODELET_DATA_HANDLE_NUMA_ACCESS`
- #define `_STARPU_FUT_NEW_MEM_NODE`
- #define `_STARPU_FUT_START_CALLBACK`
- #define `_STARPU_FUT_END_CALLBACK`
- #define `_STARPU_FUT_TASK_DONE`
- #define `_STARPU_FUT_TAG_DONE`
- #define `_STARPU_FUT_START_ALLOC`
- #define `_STARPU_FUT_END_ALLOC`
- #define `_STARPU_FUT_START_ALLOC_REUSE`
- #define `_STARPU_FUT_END_ALLOC_REUSE`
- #define `_STARPU_FUT_USED_MEM`
- #define `_STARPU_FUT_TASK_NAME`
- #define `_STARPU_FUT_DATA_WONT_USE`
- #define `_STARPU_FUT_TASK_COLOR`
- #define `_STARPU_FUT_DATA_DOING_WONT_USE`
- #define `_STARPU_FUT_TASK_LINE`
- #define `_STARPU_FUT_START_MEMRECLAIM`
- #define `_STARPU_FUT_END_MEMRECLAIM`
- #define `_STARPU_FUT_START_DRIVER_COPY`
- #define `_STARPU_FUT_END_DRIVER_COPY`
- #define `_STARPU_FUT_START_DRIVER_COPY_ASYNC`
- #define `_STARPU_FUT_END_DRIVER_COPY_ASYNC`
- #define `_STARPU_FUT_START_PROGRESS_ON_TID`
- #define `_STARPU_FUT_END_PROGRESS_ON_TID`
- #define `_STARPU_FUT_USER_EVENT`
- #define `_STARPU_FUT_SET_PROFILING`
- #define `_STARPU_FUT_TASK_WAIT_FOR_ALL`
- #define `_STARPU_FUT_EVENT`
- #define `_STARPU_FUT_THREAD_EVENT`
- #define `_STARPU_FUT_CODELET_DETAILS`
- #define `_STARPU_FUT_CODELET_DATA`
- #define `_STARPU_FUT_LOCKING_MUTEX`
- #define `_STARPU_FUT_MUTEX_LOCKED`
- #define `_STARPU_FUT_UNLOCKING_MUTEX`
- #define `_STARPU_FUT_MUTEX_UNLOCKED`
- #define `_STARPU_FUT_TRYLOCK_MUTEX`
- #define `_STARPU_FUT_RDLOCKING_RWLOCK`
- #define `_STARPU_FUT_RWLOCK_RDLOCKED`
- #define `_STARPU_FUT_WRLOCKING_RWLOCK`

- #define \_STARPU\_FUT\_RWLOCK\_WRLOCKED
- #define \_STARPU\_FUT\_UNLOCKING\_RWLOCK
- #define \_STARPU\_FUT\_RWLOCK\_UNLOCKED
- #define \_STARPU\_FUT\_LOCKING\_SPINLOCK
- #define \_STARPU\_FUT\_SPINLOCK\_LOCKED
- #define \_STARPU\_FUT\_UNLOCKING\_SPINLOCK
- #define \_STARPU\_FUT\_SPINLOCK\_UNLOCKED
- #define \_STARPU\_FUT\_TRYLOCK\_SPINLOCK
- #define \_STARPU\_FUT\_COND\_WAIT\_BEGIN
- #define \_STARPU\_FUT\_COND\_WAIT\_END
- #define \_STARPU\_FUT\_MEMORY\_FULL
- #define \_STARPU\_FUT\_DATA\_LOAD
- #define \_STARPU\_FUT\_START\_UNPARTITION\_ON\_TID
- #define \_STARPU\_FUT\_END\_UNPARTITION\_ON\_TID
- #define \_STARPU\_FUT\_START\_FREE
- #define \_STARPU\_FUT\_END\_FREE
- #define \_STARPU\_FUT\_START\_WRITEBACK
- #define \_STARPU\_FUT\_END\_WRITEBACK
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_PUSH\_PRIO
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_POP\_PRIO
- #define \_STARPU\_FUT\_START\_WRITEBACK\_ASYNC
- #define \_STARPU\_FUT\_END\_WRITEBACK\_ASYNC
- #define \_STARPU\_FUT\_HYPERVISOR\_BEGIN
- #define \_STARPU\_FUT\_HYPERVISOR\_END
- #define \_STARPU\_FUT\_BARRIER\_WAIT\_BEGIN
- #define \_STARPU\_FUT\_BARRIER\_WAIT\_END
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_START
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_END
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_PUSH
- #define \_STARPU\_FUT\_WORKER\_SCHEDULING\_POP
- #define \_STARPU\_FUT\_START\_EXECUTING
- #define \_STARPU\_FUT\_END\_EXECUTING
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_NEW
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_CONNECT
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_PUSH
- #define \_STARPU\_FUT\_SCHED\_COMPONENT\_PULL
- #define \_STARPU\_FUT\_TASK\_SUBMIT\_START
- #define \_STARPU\_FUT\_TASK\_SUBMIT\_END
- #define \_STARPU\_FUT\_TASK\_BUILD\_START
- #define \_STARPU\_FUT\_TASK\_BUILD\_END
- #define \_STARPU\_FUT\_TASK\_MPI\_DECODE\_START
- #define \_STARPU\_FUT\_TASK\_MPI\_DECODE\_END
- #define \_STARPU\_FUT\_TASK\_MPI\_PRE\_START
- #define \_STARPU\_FUT\_TASK\_MPI\_PRE\_END
- #define \_STARPU\_FUT\_TASK\_MPI\_POST\_START
- #define \_STARPU\_FUT\_TASK\_MPI\_POST\_END
- #define \_STARPU\_FUT\_TASK\_WAIT\_START
- #define \_STARPU\_FUT\_TASK\_WAIT\_END
- #define \_STARPU\_FUT\_TASK\_WAIT\_FOR\_ALL\_START
- #define \_STARPU\_FUT\_TASK\_WAIT\_FOR\_ALL\_END
- #define \_STARPU\_FUT\_HANDLE\_DATA\_REGISTER
- #define \_STARPU\_FUT\_START\_FETCH\_INPUT
- #define \_STARPU\_FUT\_END\_FETCH\_INPUT
- #define \_STARPU\_FUT\_TASK\_THROTTLE\_START
- #define \_STARPU\_FUT\_TASK\_THROTTLE\_END

- #define `_STARPU_FUT_DATA_STATE_INVALID`
- #define `_STARPU_FUT_DATA_STATE_OWNER`
- #define `_STARPU_FUT_DATA_STATE_SHARED`
- #define `_STARPU_FUT_DATA_REQUEST_CREATED`
- #define `_STARPU_FUT_PAPI_TASK_EVENT_VALUE`
- #define `_STARPU_FUT_TASK_EXCLUDE_FROM_DAG`
- #define `_STARPU_FUT_TASK_END_DEP`
- #define `_STARPU_FUT_TASK_BUBBLE`
- #define `_STARPU_FUT_START_PARALLEL_SYNC`
- #define `_STARPU_FUT_END_PARALLEL_SYNC`
- #define `_STARPU_FUT_KEYMASK_META`
- #define `_STARPU_FUT_KEYMASK_USER`
- #define `_STARPU_FUT_KEYMASK_TASK`
- #define `_STARPU_FUT_KEYMASK_TASK_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_DATA`
- #define `_STARPU_FUT_KEYMASK_DATA_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_WORKER`
- #define `_STARPU_FUT_KEYMASK_WORKER_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_DSM`
- #define `_STARPU_FUT_KEYMASK_DSM_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_SCHED`
- #define `_STARPU_FUT_KEYMASK_SCHED_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_LOCK`
- #define `_STARPU_FUT_KEYMASK_LOCK_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_EVENT`
- #define `_STARPU_FUT_KEYMASK_EVENT_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_MPI`
- #define `_STARPU_FUT_KEYMASK_MPI_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_HYP`
- #define `_STARPU_FUT_KEYMASK_HYP_VERBOSE`
- #define `_STARPU_FUT_KEYMASK_TASK_VERBOSE_EXTRA`
- #define `_STARPU_FUT_KEYMASK_MPI_VERBOSE_EXTRA`
- #define `_STARPU_TRACE_NEW_MEM_NODE(nodeid)`
- #define `_STARPU_TRACE_REGISTER_THREAD(cpuid)`
- #define `_STARPU_TRACE_WORKER_INIT_START(a, b, c, d, e, f)`
- #define `_STARPU_TRACE_WORKER_INIT_END(workerid)`
- #define `_STARPU_TRACE_START_CODELET_BODY(job, nimpl, perf_arch, workerid, rank)`
- #define `_STARPU_TRACE_END_CODELET_BODY(job, nimpl, perf_arch, workerid, rank)`
- #define `_STARPU_TRACE_START_EXECUTING(job)`
- #define `_STARPU_TRACE_END_EXECUTING(job)`
- #define `_STARPU_TRACE_START_PARALLEL_SYNC(job)`
- #define `_STARPU_TRACE_END_PARALLEL_SYNC(job)`
- #define `_STARPU_TRACE_START_CALLBACK(job)`
- #define `_STARPU_TRACE_END_CALLBACK(job)`
- #define `_STARPU_TRACE_JOB_PUSH(task, prio)`
- #define `_STARPU_TRACE_JOB_POP(task, prio)`
- #define `_STARPU_TRACE_UPDATE_TASK_CNT(counter)`
- #define `_STARPU_TRACE_START_FETCH_INPUT(job)`
- #define `_STARPU_TRACE_END_FETCH_INPUT(job)`
- #define `_STARPU_TRACE_START_PUSH_OUTPUT(job)`
- #define `_STARPU_TRACE_END_PUSH_OUTPUT(job)`
- #define `_STARPU_TRACE_TAG(tag, job)`
- #define `_STARPU_TRACE_TAG_DEPS(a, b)`
- #define `_STARPU_TRACE_TASK_DEPS(a, b)`
- #define `_STARPU_TRACE_TASK_END_DEP(a, b)`

- #define \_STARPU\_TRACE\_GHOST\_TASK\_DEPS(a, b)
- #define \_STARPU\_TRACE\_TASK\_EXCLUDE\_FROM\_DAG(a)
- #define \_STARPU\_TRACE\_TASK\_NAME\_LINE\_COLOR(a)
- #define \_STARPU\_TRACE\_TASK\_NAME(a)
- #define \_STARPU\_TRACE\_TASK\_LINE(a)
- #define \_STARPU\_TRACE\_TASK\_COLOR(a)
- #define \_STARPU\_TRACE\_TASK\_DONE(a)
- #define \_STARPU\_TRACE\_TAG\_DONE(a)
- #define \_STARPU\_TRACE\_DATA\_NAME(a, b)
- #define \_STARPU\_TRACE\_DATA\_COORDINATES(a, b, c)
- #define \_STARPU\_TRACE\_DATA\_COPY(a, b, c)
- #define \_STARPU\_TRACE\_DATA\_WONT\_USE(a)
- #define \_STARPU\_TRACE\_DATA\_DOING\_WONT\_USE(a)
- #define \_STARPU\_TRACE\_START\_DRIVER\_COPY(a, b, c, d, e, f)
- #define \_STARPU\_TRACE\_END\_DRIVER\_COPY(a, b, c, d, e)
- #define \_STARPU\_TRACE\_START\_DRIVER\_COPY\_ASYNC(a, b)
- #define \_STARPU\_TRACE\_END\_DRIVER\_COPY\_ASYNC(a, b)
- #define \_STARPU\_TRACE\_WORK\_STEALING(a, b)
- #define \_STARPU\_TRACE\_WORKER\_DEINIT\_START
- #define \_STARPU\_TRACE\_WORKER\_DEINIT\_END(a)
- #define \_STARPU\_TRACE\_WORKER\_SCHEDULING\_START
- #define \_STARPU\_TRACE\_WORKER\_SCHEDULING\_END
- #define \_STARPU\_TRACE\_WORKER\_SCHEDULING\_PUSH
- #define \_STARPU\_TRACE\_WORKER\_SCHEDULING\_POP
- #define \_STARPU\_TRACE\_WORKER\_SLEEP\_START
- #define \_STARPU\_TRACE\_WORKER\_SLEEP\_END
- #define \_STARPU\_TRACE\_TASK\_SUBMIT(job, a, b)
- #define \_STARPU\_TRACE\_TASK\_SUBMIT\_START()
- #define \_STARPU\_TRACE\_TASK\_SUBMIT\_END()
- #define \_STARPU\_TRACE\_TASK\_THROTTLE\_START()
- #define \_STARPU\_TRACE\_TASK\_THROTTLE\_END()
- #define \_STARPU\_TRACE\_TASK\_BUILD\_START()
- #define \_STARPU\_TRACE\_TASK\_BUILD\_END()
- #define \_STARPU\_TRACE\_TASK\_MPI\_DECODE\_START()
- #define \_STARPU\_TRACE\_TASK\_MPI\_DECODE\_END()
- #define \_STARPU\_TRACE\_TASK\_MPI\_PRE\_START()
- #define \_STARPU\_TRACE\_TASK\_MPI\_PRE\_END()
- #define \_STARPU\_TRACE\_TASK\_MPI\_POST\_START()
- #define \_STARPU\_TRACE\_TASK\_MPI\_POST\_END()
- #define \_STARPU\_TRACE\_TASK\_WAIT\_START(job)
- #define \_STARPU\_TRACE\_TASK\_WAIT\_END()
- #define \_STARPU\_TRACE\_TASK\_WAIT\_FOR\_ALL\_START()
- #define \_STARPU\_TRACE\_TASK\_WAIT\_FOR\_ALL\_END()
- #define \_STARPU\_TRACE\_START\_ALLOC(memnode, size, handle, is\_prefetch)
- #define \_STARPU\_TRACE\_END\_ALLOC(memnode, handle, r)
- #define \_STARPU\_TRACE\_START\_ALLOC\_REUSE(a, size, handle, is\_prefetch)
- #define \_STARPU\_TRACE\_END\_ALLOC\_REUSE(a, handle, r)
- #define \_STARPU\_TRACE\_START\_FREE(memnode, size, handle)
- #define \_STARPU\_TRACE\_END\_FREE(memnode, handle)
- #define \_STARPU\_TRACE\_START\_WRITEBACK(memnode, handle)
- #define \_STARPU\_TRACE\_END\_WRITEBACK(memnode, handle)
- #define \_STARPU\_TRACE\_USED\_MEM(memnode, used)
- #define \_STARPU\_TRACE\_START\_MEMRECLAIM(memnode, is\_prefetch)
- #define \_STARPU\_TRACE\_END\_MEMRECLAIM(memnode, is\_prefetch)
- #define \_STARPU\_TRACE\_START\_WRITEBACK\_ASYNC(memnode)

- #define `_STARPU_TRACE_END_WRITEBACK_ASYNC`(memnode)
- #define `_STARPU_TRACE_START_PROGRESS`(memnode)
- #define `_STARPU_TRACE_END_PROGRESS`(memnode)
- #define `_STARPU_TRACE_USER_EVENT`(code)
- #define `_STARPU_TRACE_SET_PROFILING`(status)
- #define `_STARPU_TRACE_TASK_WAIT_FOR_ALL`()
- #define `_STARPU_TRACE_EVENT_ALWAYS`(S)
- #define `_STARPU_TRACE_EVENT`(S)
- #define `_STARPU_TRACE_EVENT_VERBOSE`(S)
- #define `_STARPU_TRACE_THREAD_EVENT`(S)
- #define `_STARPU_TRACE_LOCKING_MUTEX`()
- #define `_STARPU_TRACE_MUTEX_LOCKED`()
- #define `_STARPU_TRACE_UNLOCKING_MUTEX`()
- #define `_STARPU_TRACE_MUTEX_UNLOCKED`()
- #define `_STARPU_TRACE_TRYLOCK_MUTEX`()
- #define `_STARPU_TRACE_RDLOCKING_RWLOCK`()
- #define `_STARPU_TRACE_RWLOCK_RDLOCKED`()
- #define `_STARPU_TRACE_WRLOCKING_RWLOCK`()
- #define `_STARPU_TRACE_RWLOCK_WRLOCKED`()
- #define `_STARPU_TRACE_UNLOCKING_RWLOCK`()
- #define `_STARPU_TRACE_RWLOCK_UNLOCKED`()
- #define `_STARPU_TRACE_LOCKING_SPINLOCK`(file, line)
- #define `_STARPU_TRACE_SPINLOCK_LOCKED`(file, line)
- #define `_STARPU_TRACE_UNLOCKING_SPINLOCK`(file, line)
- #define `_STARPU_TRACE_SPINLOCK_UNLOCKED`(file, line)
- #define `_STARPU_TRACE_TRYLOCK_SPINLOCK`(file, line)
- #define `_STARPU_TRACE_COND_WAIT_BEGIN`()
- #define `_STARPU_TRACE_COND_WAIT_END`()
- #define `_STARPU_TRACE_BARRIER_WAIT_BEGIN`()
- #define `_STARPU_TRACE_BARRIER_WAIT_END`()
- #define `_STARPU_TRACE_MEMORY_FULL`(size)
- #define `_STARPU_TRACE_DATA_LOAD`(workerid, size)
- #define `_STARPU_TRACE_START_UNPARTITION`(handle, memnode)
- #define `_STARPU_TRACE_END_UNPARTITION`(handle, memnode)
- #define `_STARPU_TRACE_SCHED_COMPONENT_PUSH_PRIO`(workerid, ntasks, exp\_len)
- #define `_STARPU_TRACE_SCHED_COMPONENT_POP_PRIO`(workerid, ntasks, exp\_len)
- #define `_STARPU_TRACE_HYPERVISOR_BEGIN`()
- #define `_STARPU_TRACE_HYPERVISOR_END`()
- #define `_STARPU_TRACE_SCHED_COMPONENT_NEW`(component)
- #define `_STARPU_TRACE_SCHED_COMPONENT_CONNECT`(parent, child)
- #define `_STARPU_TRACE_SCHED_COMPONENT_PUSH`(from, to, task, prio)
- #define `_STARPU_TRACE_SCHED_COMPONENT_PULL`(from, to, task)
- #define `_STARPU_TRACE_HANDLE_DATA_REGISTER`(handle)
- #define `_STARPU_TRACE_HANDLE_DATA_UNREGISTER`(handle)
- #define `_STARPU_TRACE_WORKER_START_FETCH_INPUT`(job, id)
- #define `_STARPU_TRACE_WORKER_END_FETCH_INPUT`(job, id)
- #define `_STARPU_TRACE_DATA_STATE_INVALID`(handle, node)
- #define `_STARPU_TRACE_DATA_STATE_OWNER`(handle, node)
- #define `_STARPU_TRACE_DATA_STATE_SHARED`(handle, node)
- #define `_STARPU_TRACE_DATA_REQUEST_CREATED`(handle, orig, dest, prio, is\_pre, req)
- #define `_STARPU_TRACE_PAPI_TASK_EVENT`(event\_id, task, value)
- #define `_STARPU_TRACE_BUBBLE_TASK_DEPS`(a, b)
- #define `_STARPU_TRACE_BUBBLE`(a)

## Functions

- static unsigned long `_starpu_fxt_get_job_id` (void)

## Variables

- unsigned long `_starpu_job_cnt`

## 6.33 graph.h File Reference

```
#include <core/task.h>
#include <common/list.h>
```

## Data Structures

- struct `_starpu_graph_node`

## Functions

- void `_starpu_graph_init` (void)
- void `_starpu_graph_wrllock` (void)
- void `_starpu_graph_rdlock` (void)
- void `_starpu_graph_wrunlock` (void)
- void `_starpu_graph_rdlunlock` (void)
- void `_starpu_graph_add_job` (struct `_starpu_job` \*job)
- void `_starpu_graph_add_job_dep` (struct `_starpu_job` \*job, struct `_starpu_job` \*prev\_job)
- void `_starpu_graph_drop_job` (struct `_starpu_job` \*job)
- void `_starpu_graph_drop_dropped_nodes` (void)
- void `_starpu_graph_compute_depths` (void)
- void `_starpu_graph_compute_descendants` (void)
- void `_starpu_graph_foreach` (void(\*func)(void \*data, struct `_starpu_graph_node` \*node), void \*data)
- struct `_starpu_graph_node` \* `_starpu_graph_task_node` (struct `starpu_task` \*task)
- struct `starpu_task` \* `_starpu_graph_node_task` (struct `_starpu_graph_node` \*node)
- void `_starpu_graph_node_outgoing` (struct `_starpu_graph_node` \*node, unsigned \*n\_outgoing, struct `_starpu_graph_node` \*\*\*outgoing)

## Variables

- int `_starpu_graph_record`

### 6.33.1 Data Structure Documentation

#### 6.33.1.1 struct `_starpu_graph_node`

##### Data Fields

<code>starpu_thread_mutex_t</code>	mutex	protects access to the job
<code>struct _starpu_job *</code>	job	pointer to the job, if it is still alive, NULL otherwise
<code>struct _starpu_graph_node_multilist_top</code>	top	Fields for graph analysis for scheduling heuristics Member of list of all jobs without incoming dependency
<code>struct _starpu_graph_node_multilist_bottom</code>	bottom	Member of list of all jobs without outgoing dependency
<code>struct _starpu_graph_node_multilist_all</code>	all	Member of list of all jobs

## Data Fields

<code>struct _starpu_graph_node_multilist_dropped</code>	dropped	Member of list of dropped jobs
<code>struct _starpu_graph_node **</code>	incoming	set of incoming dependencies May contain NULLs for terminated jobs
<code>unsigned *</code>	incoming_slot	Index within corresponding outgoing array
<code>unsigned</code>	n_incoming	Number of slots used
<code>unsigned</code>	alloc_incoming	Size of incoming
<code>struct _starpu_graph_node **</code>	outgoing	set of outgoing dependencies
<code>unsigned</code>	total_incoming	Total number of incoming dependencies, including those who completed
<code>unsigned *</code>	outgoing_slot	Index within corresponding incoming array
<code>unsigned</code>	n_outgoing	Number of slots used
<code>unsigned</code>	alloc_outgoing	Size of outgoing
<code>unsigned</code>	depth	Rank from bottom, in number of jobs Only available if <code>_starpu_graph_compute_depths</code> was called
<code>unsigned</code>	descendants	Number of children, grand-children, etc. Only available if <code>_starpu_graph_compute_descendants</code> was called
<code>int</code>	graph_n	Variable available for graph flow

## 6.33.2 Function Documentation

### 6.33.2.1 `_starpu_graph_add_job()`

```
void _starpu_graph_add_job (
    struct _starpu_job * job )
```

Add a job to the graph, called before any `_starpu_graph_add_job_dep` call

### 6.33.2.2 `_starpu_graph_add_job_dep()`

```
void _starpu_graph_add_job_dep (
    struct _starpu_job * job,
    struct _starpu_job * prev_job )
```

Add a dependency between jobs

### 6.33.2.3 `_starpu_graph_drop_job()`

```
void _starpu_graph_drop_job (
    struct _starpu_job * job )
```

Remove a job from the graph

### 6.33.2.4 `_starpu_graph_drop_dropped_nodes()`

```
void _starpu_graph_drop_dropped_nodes (
    void )
```

Really drop the nodes from the graph now

### 6.33.2.5 `_starpu_graph_compute_depths()`

```
void _starpu_graph_compute_depths (
    void )
```

This make StarPU compute for each task the depth, i.e. the length of the longest path to a task without outgoing dependencies. This does not take job duration into account, just the number

### 6.33.2.6 `_starpu_graph_compute_descendants()`

```
void _starpu_graph_compute_descendants (
    void )
```

Compute the descendants of jobs in the graph

### 6.33.2.7 `_starpu_graph_foreach()`

```
void _starpu_graph_foreach (
    void(*) (void *data, struct _starpu_graph_node *node) func,
    void * data )
```

This calls *func* for each node of the task graph, passing also *data* as it Apply func on each job of the graph

## 6.34 helper\_mct.h File Reference

### Data Structures

- struct [\\_starpu\\_mct\\_data](#)

### Functions

- struct [\\_starpu\\_mct\\_data](#) \* **starpu\_mct\_init\_parameters** (struct starpu\_sched\_component\_mct\_data \*params)
- unsigned **starpu\_mct\_compute\_execution\_times** (struct starpu\_sched\_component \*component, struct starpu\_task \*task, double \*estimated\_lengths, double \*estimated\_transfer\_length, unsigned \*suitable\_↔ components)
- void **starpu\_mct\_compute\_expected\_times** (struct starpu\_sched\_component \*component, struct starpu\_task \*task, double \*estimated\_lengths, double \*estimated\_transfer\_length, double \*estimated\_↔ \_ends\_with\_task, double \*min\_exp\_end\_of\_task, double \*max\_exp\_end\_of\_workers, unsigned \*suitable\_↔ \_components, unsigned nsuitable\_components)
- double **starpu\_mct\_compute\_fitness** (struct [\\_starpu\\_mct\\_data](#) \*d, double exp\_end, double min\_exp\_end, double max\_exp\_end, double transfer\_len, double local\_energy)
- int **starpu\_mct\_get\_best\_component** (struct [\\_starpu\\_mct\\_data](#) \*d, struct starpu\_task \*task, double \*estimated\_lengths, double \*estimated\_transfer\_length, double \*estimated\_ends\_with\_task, double \*local\_↔ \_energy, double min\_exp\_end\_of\_task, double max\_exp\_end\_of\_workers, unsigned \*suitable\_components, unsigned nsuitable\_components)
- void **starpu\_mct\_compute\_energy** (struct starpu\_sched\_component \*component, struct starpu\_task \*task, double \*local\_energy, unsigned \*suitable\_components, unsigned nsuitable\_components)
- int **eager\_calibration\_push\_task** (struct starpu\_sched\_component \*component, struct starpu\_task \*task)

### 6.34.1 Data Structure Documentation

#### 6.34.1.1 struct [\\_starpu\\_mct\\_data](#)

##### Data Fields

double	alpha	
double	beta	
double	_gamma	
double	idle_power	
starpu_pthread_mutex_t	scheduling_mutex	

## 6.35 idle\_hook.h File Reference

### Functions

- void `_starpu_init_idle_hooks` (void)
- unsigned `_starpu_execute_registered_idle_hooks` (void)

## 6.36 implicit\_data\_deps.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

### Functions

- `struct` `starpu_task *` `_starpu_detect_implicit_data_deps_with_handle` (`struct` `starpu_task *` `pre_sync`, `task`, `int` `*submit_pre_sync`, `struct` `starpu_task *` `post_sync_task`, `struct` `_starpu_task_wrapper_dlist *` `post_sync_task_dependency_slot`, `starpu_data_handle_t` `handle`, `enum` `starpu_data_access_mode` `mode`, `unsigned` `task_handle_sequential_consistency`)
- `int` `_starpu_test_implicit_data_deps_with_handle` (`starpu_data_handle_t` `handle`, `enum` `starpu_data_access_mode` `mode`)
- void `_starpu_detect_implicit_data_deps` (`struct` `starpu_task *` `task`)
- void `_starpu_release_data_enforce_sequential_consistency` (`struct` `starpu_task *` `task`, `struct` `_starpu_task_wrapper_dlist *` `task_dependency_slot`, `starpu_data_handle_t` `handle`)
- void `_starpu_release_task_enforce_sequential_consistency` (`struct` `_starpu_job *` `j`)
- void `_starpu_add_post_sync_tasks` (`struct` `starpu_task *` `post_sync_task`, `starpu_data_handle_t` `handle`)
- void `_starpu_unlock_post_sync_tasks` (`starpu_data_handle_t` `handle`, `enum` `starpu_data_access_mode` `mode`)
- void `_starpu_implicit_data_deps_write_hook` (`void(*func)(starpu_data_handle_t)` `STARPU_ATTRIBUTE_VISIBILITY_DEFAULT`)
- `int` `_starpu_data_wait_until_available` (`starpu_data_handle_t` `handle`, `enum` `starpu_data_access_mode` `mode`, `const char *` `sync_name`)
- void `_starpu_data_clear_implicit` (`starpu_data_handle_t` `handle`)

### 6.36.1 Function Documentation

#### 6.36.1.1 `_starpu_implicit_data_deps_write_hook()`

```
void _starpu_implicit_data_deps_write_hook (
    void(*) (starpu_data_handle_t) func )
```

Register a hook to be called when a write is submitted

#### 6.36.1.2 `_starpu_data_wait_until_available()`

```
int _starpu_data_wait_until_available (
    starpu_data_handle_t handle,
    enum starpu_data_access_mode mode,
    const char * sync_name )
```

This function blocks until the handle is available in the requested mode

## 6.37 jobs.h File Reference

```
#include <starpu.h>
#include <semaphore.h>
#include <stdio.h>
```

```

#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <stdarg.h>
#include <common/config.h>
#include <common/timing.h>
#include <common/list.h>
#include <common/fxt.h>
#include <core/dependencies/tags.h>
#include <datawizard/datawizard.h>
#include <core/perfmodel/perfmodel.h>
#include <core/errorcheck.h>
#include <common/barrier.h>
#include <common/utils.h>

```

## Data Structures

- [struct \\_starpu\\_data\\_descr](#)
- [struct \\_starpu\\_job](#)

## Macros

- `#define _STARPU_MAY_PERFORM(j, arch)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_INDEX(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_HANDLE(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_MODE(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_NODE(job, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFER_ORIG_NODE(job, i)`
- `#define _STARPU_JOB_SET_ORDERED_BUFFER(job, buffer, i)`
- `#define _STARPU_JOB_GET_ORDERED_BUFFERS(job)`
- `#define _STARPU_JOB_GET_DEP_SLOTS(job)`

## Typedefs

- `typedef void(* _starpu_cl_func_t) (void **, void *)`

## Functions

- `void _starpu_job_init (void)`
- `void _starpu_job_fini (void)`
- `struct _starpu_job * _starpu_job_create (struct starpu_task *task) STARPU_ATTRIBUTE_MALLOC`
- `void _starpu_job_destroy (struct _starpu_job *j)`
- `int _starpu_job_finished (struct _starpu_job *j)`
- `void _starpu_wait_job (struct _starpu_job *j)`
- `int _starpu_test_job_termination (struct _starpu_job *j)`
- `void _starpu_job_prepare_for_continuation_ext (struct _starpu_job *j, unsigned continuation_resubmit, void(*continuation_callback_on_sleep)(void *arg), void *continuation_callback_on_sleep_arg)`
- `void _starpu_job_prepare_for_continuation (struct _starpu_job *j)`
- `void _starpu_job_set_omp_cleanup_callback (struct _starpu_job *j, void(*omp_cleanup_callback)(void *arg), void *omp_cleanup_callback_arg)`
- `void _starpu_exclude_task_from_dag (struct starpu_task *task)`
- `unsigned _starpu_enforce_deps_and_schedule (struct _starpu_job *j)`
- `unsigned _starpu_enforce_deps_starting_from_task (struct _starpu_job *j)`
- `unsigned _starpu_reenforce_task_deps_and_schedule (struct _starpu_job *j)`
- `unsigned _starpu_take_deps_and_schedule (struct _starpu_job *j)`

- void `_starpus_enforce_deps_notify_job_ready_soon` (`struct _starpus_job` \*j, `_starpus_notify_job_start_data` \*data, int tag)
- void `_starpus_handle_job_submission` (`struct _starpus_job` \*j)
- void `_starpus_handle_job_termination` (`struct _starpus_job` \*j)
- size\_t `_starpus_job_get_data_size` (`struct starpus_permodel` \*model, `struct starpus_permodel_arch` \*arch, unsigned nimpl, `struct _starpus_job` \*)
- `struct starpus_task` \* `_starpus_pop_local_task` (`struct _starpus_worker` \*worker)
- int `_starpus_push_local_task` (`struct _starpus_worker` \*worker, `struct starpus_task` \*task)

## 6.37.1 Data Structure Documentation

### 6.37.1.1 `struct _starpus_data_descr`

#### Data Fields

<code>starpus_data_handle_t</code>	handle	
enum <code>starpus_data_access_mode</code>	mode	
int	orig_node	
int	node	This is the original node in the codelet
int	index	This is the value actually chosen, only set by <code>_starpus_fetch_task_input</code> for coherency with <code>_starpus_fetch_task_input_tail</code> and <code>_starpus_push_task_output</code>
int	orderedindex	

## 6.37.2 Typedef Documentation

### 6.37.2.1 `_starpus_cl_func_t`

```
typedef void(*_starpus_cl_func_t) (void **, void *)
codelet function
```

## 6.37.3 Function Documentation

### 6.37.3.1 `_starpus_job_create()`

```
struct _starpus_job * _starpus_job_create (
    struct starpus_task * task )
```

Create an internal struct `_starpus_job` \*structure to encapsulate the task.

### 6.37.3.2 `_starpus_job_destroy()`

```
void _starpus_job_destroy (
    struct _starpus_job * j )
```

Destroy the data structure associated to the job structure

### 6.37.3.3 `_starpus_job_finished()`

```
int _starpus_job_finished (
    struct _starpus_job * j )
```

Test for the termination of the job

**6.37.3.4 \_starpu\_wait\_job()**

```
void _starpu_wait_job (
    struct _starpu_job * j )
```

Wait for the termination of the job

**6.37.3.5 \_starpu\_test\_job\_termination()**

```
int _starpu_test_job_termination (
    struct _starpu_job * j )
```

Test for the termination of the job

**6.37.3.6 \_starpu\_job\_prepare\_for\_continuation\_ext()**

```
void _starpu_job_prepare_for_continuation_ext (
    struct _starpu_job * j,
    unsigned continuation_resubmit,
    void(*) (void *arg) continuation_callback_on_sleep,
    void * continuation_callback_on_sleep_arg )
```

Prepare the job for accepting new dependencies before becoming a continuation.

**6.37.3.7 \_starpu\_exclude\_task\_from\_dag()**

```
void _starpu_exclude_task_from_dag (
    struct starpu_task * task )
```

Specify that the task should not appear in the DAG generated by debug tools.

**6.37.3.8 \_starpu\_enforce\_deps\_and\_schedule()**

```
unsigned _starpu_enforce_deps_and_schedule (
    struct _starpu_job * j )
```

try to submit job j, enqueue it if it's not schedulable yet. The job's sync mutex is supposed to be held already

**6.37.3.9 \_starpu\_reenforce\_task\_deps\_and\_schedule()**

```
unsigned _starpu_reenforce_task_deps_and_schedule (
    struct _starpu_job * j )
```

When waking up a continuation, we only enforce new task dependencies

**6.37.3.10 \_starpu\_handle\_job\_submission()**

```
void _starpu_handle_job_submission (
    struct _starpu_job * j )
```

Called at the submission of the job

**6.37.3.11 \_starpu\_handle\_job\_termination()**

```
void _starpu_handle_job_termination (
    struct _starpu_job * j )
```

This function must be called after the execution of a job, this triggers all job's dependencies and perform the callback function if any.

**6.37.3.12 \_starpu\_job\_get\_data\_size()**

```
size_t _starpu_job_get_data_size (
    struct starpu_perfmodel * model,
    struct starpu_perfmodel_arch * arch,
    unsigned nimpl,
    struct _starpu_job * j )
```

Get the sum of the size of the data accessed by the job.

### 6.37.3.13 `_starpu_pop_local_task()`

```
struct starpu_task * _starpu_pop_local_task (
    struct _starpu_worker * worker )
```

Get a task from the local pool of tasks that were explicitly attributed to that worker.

### 6.37.3.14 `_starpu_push_local_task()`

```
int _starpu_push_local_task (
    struct _starpu_worker * worker,
    struct starpu_task * task )
```

Put a task into the pool of tasks that are explicitly attributed to the specified worker.

## 6.38 knobs.h File Reference

```
#include <stdint.h>
#include <starpu.h>
#include <common/config.h>
```

### Data Structures

- struct [starpu\\_perf\\_counter](#)
- struct [starpu\\_perf\\_counter\\_set](#)
- union [starpu\\_perf\\_counter\\_value](#)
- struct [starpu\\_perf\\_counter\\_listener](#)
- struct [starpu\\_perf\\_counter\\_sample](#)
- struct [starpu\\_perf\\_counter\\_sample\\_cl\\_values](#)
- struct [starpu\\_perf\\_knob\\_value](#)
- struct [starpu\\_perf\\_knob\\_group](#)
- struct [starpu\\_perf\\_knob](#)
- struct [starpu\\_perf\\_counter\\_sample\\_cl\\_values.task](#)
- union [starpu\\_perf\\_knob\\_value.\\_\\_unnamed3\\_\\_](#)

### Macros

- #define [STARPU\\_ASSERT\\_PERF\\_COUNTER\\_SCOPE\\_DEFINED\(t\)](#)
- #define [STARPU\\_ASSERT\\_PERF\\_COUNTER\\_TYPE\\_DEFINED\(t\)](#)
- #define [\\_\\_STARPU\\_PERF\\_COUNTER\\_ID\\_SCOPE\\_BITS](#)
- #define [\\_\\_STARPU\\_PERF\\_COUNTER\\_UPDATE\\_32BIT\(OPNAME, OP, TYPENAME, TYPE\)](#)
- #define [\\_\\_STARPU\\_PERF\\_COUNTER\\_UPDATE\\_64BIT\(OPNAME, OP, TYPENAME, TYPE\)](#)
- #define [\\_\\_STARPU\\_PERF\\_COUNTER\\_UPDATE\\_ACC\\_FLOAT\(TYPENAME, TYPE\)](#)
- #define [STARPU\\_PERF\\_COUNTER\\_ADD64\(ptr, val\)](#)
- #define [\\_\\_STARPU\\_PERF\\_COUNTER\\_SAMPLE\\_SET\\_TYPED\\_VALUE\(String, TYPE\)](#)
- #define [\\_\\_STARPU\\_PERF\\_COUNTER\\_REG\(PREFIX, SCOPE, CTR, TYPESTRING, HELP\)](#)
- #define [STARPU\\_ASSERT\\_PERF\\_KNOB\\_SCOPE\\_DEFINED\(t\)](#)
- #define [STARPU\\_ASSERT\\_PERF\\_KNOB\\_TYPE\\_DEFINED\(t\)](#)
- #define [\\_\\_STARPU\\_PERF\\_KNOBS\\_ID\\_SCOPE\\_BITS](#)
- #define [\\_\\_STARPU\\_PERF\\_KNOB\\_REG\(PREFIX, SCOPE, CTR, TYPESTRING, HELP\)](#)

### Typedefs

- typedef int32\_t [starpu\\_perf\\_counter\\_int64\\_t](#)
- typedef float [starpu\\_perf\\_counter\\_double](#)
- typedef void(\* [starpu\\_perf\\_counter\\_sample\\_updater](#)) ([struct starpu\\_perf\\_counter\\_sample](#) \*sample, void \*context)

## Functions

- `__STARPU_PERF_COUNTER_UPDATE_32BIT` (max,>=, int32, int32\_t)
- `__STARPU_PERF_COUNTER_UPDATE_32BIT` (max,>=, float, float)
- `__STARPU_PERF_COUNTER_UPDATE_64BIT` (max,>=, int64, starpu\_perf\_counter\_int64\_t)
- `__STARPU_PERF_COUNTER_UPDATE_64BIT` (max,>=, double, starpu\_perf\_counter\_double)
- `__STARPU_PERF_COUNTER_UPDATE_32BIT` (min,<=, int32, int32\_t)
- `__STARPU_PERF_COUNTER_UPDATE_32BIT` (min,<=, float, float)
- `__STARPU_PERF_COUNTER_UPDATE_64BIT` (min,<=, int64, starpu\_perf\_counter\_int64\_t)
- `__STARPU_PERF_COUNTER_UPDATE_64BIT` (min,<=, double, starpu\_perf\_counter\_double)
- `__STARPU_PERF_COUNTER_UPDATE_ACC_FLOAT` (float, float)
- `__STARPU_PERF_COUNTER_UPDATE_ACC_FLOAT` (double, starpu\_perf\_counter\_double)
- static enum starpu\_perf\_counter\_scope `starpu_perf_counter_id_get_scope` (const int counter\_id)
- static int `starpu_perf_counter_id_get_index` (const int counter\_id)
- static int `starpu_perf_counter_id_build` (const enum starpu\_perf\_counter\_scope scope, const int index)
- void `starpu_perf_counter_sample_init` ([struct starpu\\_perf\\_counter\\_sample](#) \*sample, enum starpu\_↔perf\_counter\_scope scope)
- void `starpu_perf_counter_sample_exit` ([struct starpu\\_perf\\_counter\\_sample](#) \*sample)
- void `starpu_perf_counter_init` ([struct starpu\\_machine\\_config](#) \*pconfig)
- void `starpu_perf_counter_exit` (void)
- int `starpu_perf_counter_register` (enum starpu\_perf\_counter\_scope scope, const char \*name, enum starpu\_perf\_counter\_type type, const char \*help)
- void `starpu_perf_counter_unregister_all_scopes` (void)
- void `starpu_perf_counter_register_updater` (enum starpu\_perf\_counter\_scope scope, void(\*updater)([struct starpu\\_perf\\_counter\\_sample](#) \*sample, void \*context))
- void `starpu_perf_counter_update_global_sample` (void)
- void `starpu_perf_counter_update_per_worker_sample` (unsigned workerid)
- void `starpu_perf_counter_update_per_codelet_sample` ([struct starpu\\_codelet](#) \*cl)
- `__STARPU_PERF_COUNTER_SAMPLE_SET_TYPED_VALUE` (int32, int32\_t)
- `__STARPU_PERF_COUNTER_SAMPLE_SET_TYPED_VALUE` (int64, starpu\_perf\_counter\_int64\_t)
- `__STARPU_PERF_COUNTER_SAMPLE_SET_TYPED_VALUE` (float, float)
- `__STARPU_PERF_COUNTER_SAMPLE_SET_TYPED_VALUE` (double, starpu\_perf\_counter\_double)
- void `starpu__task_c__register_counters` (void)
- static int `starpu_perf_knob_id_get_scope` (const int knob\_id)
- static int `starpu_perf_knob_id_get_index` (const int knob\_id)
- static int `starpu_perf_knob_id_build` (const enum starpu\_perf\_knob\_scope scope, const int index)
- void `starpu_perf_knob_init` (void)
- void `starpu_perf_knob_exit` (void)
- [struct starpu\\_perf\\_knob\\_group](#) \* `starpu_perf_knob_group_register` (enum starpu\_perf\_knob\_↔scope scope, void(\*set\_func)(const [struct starpu\\_perf\\_knob](#) \*const knob, void \*context, const [struct starpu\\_perf\\_knob\\_value](#) \*const value), void(\*get\_func)(const [struct starpu\\_perf\\_knob](#) \*const knob, void \*context, [struct starpu\\_perf\\_knob\\_value](#) \*const value))
- void `starpu_perf_knob_group_unregister` ([struct starpu\\_perf\\_knob\\_group](#) \*group)
- int `starpu_perf_knob_register` ([struct starpu\\_perf\\_knob\\_group](#) \*group, const char \*name, enum starpu\_↔perf\_knob\_type type, const char \*help)
- void `starpu_perf_knob_unregister_all_scopes` (void)
- void `starpu__workers_c__register_knobs` (void)
- void `starpu__task_c__register_knobs` (void)
- void `starpu__dmda_c__register_knobs` (void)
- void `starpu__workers_c__unregister_knobs` (void)
- void `starpu__task_c__unregister_knobs` (void)
- void `starpu__dmda_c__unregister_knobs` (void)

## Variables

- `starpu_perf_counter_int64_t_starpu_task_g_total_submitted_value`
- `starpu_perf_counter_int64_t_starpu_task_g_peak_submitted_value`
- `starpu_perf_counter_int64_t_starpu_task_g_current_submitted_value`
- `starpu_perf_counter_int64_t_starpu_task_g_peak_ready_value`
- `starpu_perf_counter_int64_t_starpu_task_g_current_ready_value`

### 6.38.1 Data Structure Documentation

#### 6.38.1.1 struct starpu\_perf\_counter

##### Data Fields

int	id	
const char *	name	
const char *	help	
enum starpu_perf_counter_type	type	

#### 6.38.1.2 struct starpu\_perf\_counter\_set

##### Data Fields

enum starpu_perf_counter_scope	scope	
int	size	
int *	index_array	

#### 6.38.1.3 union starpu\_perf\_counter\_value

##### Data Fields

int32_t	int32_val	
starpu_perf_counter_int64_t	int64_val	
float	float_val	
starpu_perf_counter_double	double_val	

#### 6.38.1.4 struct starpu\_perf\_counter\_sample

##### Data Fields

enum starpu_perf_counter_scope	scope	
<a href="#">struct starpu_perf_counter_listener</a> *	listener	
union <a href="#">starpu_perf_counter_value</a> *	value_array	
<a href="#">struct _starpu_spinlock</a>	lock	

#### 6.38.1.5 struct starpu\_perf\_counter\_sample\_cl\_values

##### Data Fields

struct <a href="#">starpu_perf_counter_sample_cl_values.task</a>	task	
--	------	--

**6.38.1.6 struct starpu\_perf\_knob\_value**

## Data Fields

enum starpu_perf_knob_type	type	
union starpu_perf_knob_value.__unnamed3__	__unnamed__	

**6.38.1.7 struct starpu\_perf\_knob**

## Data Fields

int	id	
int	id_in_group	
const char *	name	
const char *	help	
enum starpu_perf_knob_type	type	
struct starpu_perf_knob_group *	group	

**6.38.1.8 struct starpu\_perf\_counter\_sample\_cl\_values.task**

## Data Fields

starpu_perf_counter_int64_t	total_submitted	
starpu_perf_counter_int64_t	peak_submitted	
starpu_perf_counter_int64_t	current_submitted	
starpu_perf_counter_int64_t	peak_ready	
starpu_perf_counter_int64_t	current_ready	
starpu_perf_counter_int64_t	total_executed	
starpu_perf_counter_double	cumul_execution_time	

**6.38.1.9 union starpu\_perf\_knob\_value.\_\_unnamed3\_\_**

## Data Fields

int32_t	val_int32_t	
starpu_perf_counter_int64_t	val_int64_t	
float	val_float	
starpu_perf_counter_double	val_double	

**6.38.2 Macro Definition Documentation****6.38.2.1 STARPU\_ASSERT\_PERF\_COUNTER\_SCOPE\_DEFINED**

```
#define STARPU_ASSERT_PERF_COUNTER_SCOPE_DEFINED (
    t )
```

Performance Monitoring

**6.38.2.2 \_\_STARPU\_PERF\_COUNTER\_UPDATE\_ACC\_FLOAT**

```
#define __STARPU_PERF_COUNTER_UPDATE_ACC_FLOAT (
```

```

        TYPENAME,
        TYPE )

```

Floating point atomic accumulate

## 6.39 malloc.h File Reference

```

#include <common/list.h>
#include <common/utills.h>

```

### Data Structures

- struct [block](#)

### Macros

- #define [CHUNK\\_SIZE](#)
- #define [CHUNK\\_ALLOC\\_MAX](#)
- #define [CHUNK\\_ALLOC\\_MIN](#)
- #define [CHUNKS\\_NFREE](#)
- #define [CHUNK\\_NBLOCKS](#)

### Functions

- void [\\_starpu\\_malloc\\_init](#) (unsigned dst\_node)
- void [\\_starpu\\_malloc\\_shutdown](#) (unsigned dst\_node)
- int [\\_starpu\\_malloc\\_flags\\_on\\_node](#) (unsigned dst\_node, void \*\*A, size\_t dim, int flags)
- int [\\_starpu\\_free\\_flags\\_on\\_node](#) (unsigned dst\_node, void \*A, size\_t dim, int flags)
- int [\\_starpu\\_malloc\\_willpin\\_on\\_node](#) (unsigned dst\_node) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

## 6.39.1 Data Structure Documentation

### 6.39.1.1 struct block

Data Fields

int	length	
int	next	

## 6.39.2 Macro Definition Documentation

### 6.39.2.1 CHUNK\_SIZE

```
#define CHUNK_SIZE
```

On CUDA which has very expensive malloc, for small sizes, allocate big chunks divided in blocks, and we actually allocate segments of consecutive blocks.

We try to keep the list of chunks with increasing occupancy, so we can quickly find free segments to allocate.

## 6.39.3 Function Documentation

### 6.39.3.1 `_starpu_malloc_willpin_on_node()`

```
int _starpu_malloc_willpin_on_node (
    unsigned dst_node )
```

Returns whether when allocating data on `dst_node`, we will do pinning, i.e. the allocation will be very expensive, and should thus be moved out from the critical path

## 6.40 `memalloc.h` File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/coherency.h>
#include <datawizard/copy_driver.h>
#include <datawizard/data_request.h>
```

## 6.41 `memory_manager.h` File Reference

```
#include <starpu.h>
```

### Functions

- `int _starpu_memory_manager_init ()`
- `void _starpu_memory_manager_set_global_memory_size (unsigned node, size_t size)`
- `size_t _starpu_memory_manager_get_global_memory_size (unsigned node)`
- `int _starpu_memory_manager_test_allocate_size (unsigned node, size_t size)`

### 6.41.1 Function Documentation

#### 6.41.1.1 `_starpu_memory_manager_init()`

```
int _starpu_memory_manager_init ( )
```

Initialises the memory manager

#### 6.41.1.2 `_starpu_memory_manager_set_global_memory_size()`

```
void _starpu_memory_manager_set_global_memory_size (
    unsigned node,
    size_t size )
```

Initialises the global memory size for the given node

#### 6.41.1.3 `_starpu_memory_manager_get_global_memory_size()`

```
size_t _starpu_memory_manager_get_global_memory_size (
    unsigned node )
```

Gets the global memory size for the given node

## 6.42 `memory_nodes.h` File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <datawizard/coherency.h>
```

```
#include <datawizard/memalloc.h>
#include <datawizard/node_ops.h>
#include <common/utils.h>
#include <core/workers.h>
#include <core/simgrid.h>
```

## Data Structures

- [struct \\_starpu\\_cond\\_and\\_worker](#)
- [struct \\_starpu\\_memory\\_node\\_descr](#)

## Macros

- `#define _starpu_node_needs_map_update(node)`
- `#define starpu_node_get_kind`
- `#define starpu_memory_nodes_get_count`
- `#define starpu_worker_get_memory_node`
- `#define starpu_worker_get_local_memory_node`

## Functions

- `void _starpu_memory_nodes_init (void)`
- `void _starpu_memory_nodes_deinit (void)`
- `static void _starpu_memory_node_add_nworkers (unsigned node)`
- `void _starpu_worker_drives_memory_node (struct _starpu_worker *worker, unsigned memnode)`
- `static const struct _starpu_node_ops * _starpu_memory_node_get_node_ops (unsigned node)`
- `static unsigned _starpu_memory_node_get_nworkers (unsigned node)`
- `static void _starpu_simgrid_memory_node_set_host (unsigned node, starpu_sg_host_t host)`
- `static starpu_sg_host_t _starpu_simgrid_memory_node_get_host (unsigned node)`
- `void _starpu_memory_node_set_mapped (unsigned node)`
- `unsigned _starpu_memory_node_get_mapped (unsigned node)`
- `unsigned _starpu_memory_node_register (enum starpu_node_kind kind, int devid)`
- `void _starpu_memory_node_register_condition (struct _starpu_worker *worker, starpu_pthread_cond_t *cond, unsigned nodeid)`
- `static struct _starpu_memory_node_descr * _starpu_memory_node_get_description (void)`
- `static enum starpu_node_kind _starpu_node_get_kind (unsigned node)`
- `static unsigned _starpu_memory_nodes_get_count (void)`
- `static unsigned _starpu_worker_get_memory_node (unsigned workerid)`
- `static unsigned _starpu_worker_get_local_memory_node (void)`

## Variables

- `char _starpu_worker_drives_memory [STARPU_NMAXWORKERS][STARPU_MAXNODES]`
- `struct _starpu_memory_node_descr _starpu_descr`

### 6.42.1 Data Structure Documentation

#### 6.42.1.1 struct \_starpu\_cond\_and\_worker

##### Data Fields

<code>starpu_pthread_cond_t *</code>	<code>cond</code>	
<code>struct _starpu_worker *</code>	<code>worker</code>	

### 6.42.1.2 struct \_starpu\_memory\_node\_descr

#### Data Fields

unsigned	nnodes	
enum <a href="#">starpu_node_kind</a>	nodes[ <a href="#">STARPU_MAXNODES</a> ]	
const <a href="#">struct _starpu_node_ops</a> *	node_ops[ <a href="#">STARPU_MAXNODES</a> ]	
int	devid[ <a href="#">STARPU_MAXNODES</a> ]	Get the device id associated to this node, or -1 if not applicable
unsigned	nworkers[ <a href="#">STARPU_MAXNODES</a> ]	
<a href="#">starpu_sg_host_t</a>	host[ <a href="#">STARPU_MAXNODES</a> ]	
<a href="#">starpu_pthread_rwlock_t</a>	conditions_rwlock	Every worker is associated to a condition variable on which the worker waits when there is task available. It is possible that multiple worker share the same condition variable, so we maintain a list of all these condition variables so that we can wake up all worker attached to a memory node that are waiting on a task.
<a href="#">struct _starpu_cond_and_worker</a>	conditions_attached_to_node[ <a href="#">STARPU_MAXNODES</a> ][ <a href="#">STARPU_NMAXWORKERS</a> ]	
<a href="#">struct _starpu_cond_and_worker</a>	conditions_all[ <a href="#">STARPU_MAXNODES</a> * <a href="#">STARPU_NMAXWORKERS</a> ]	
unsigned	total_condition_count	the number of queues attached to each node
unsigned	condition_count[ <a href="#">STARPU_MAXNODES</a> ]	
unsigned	mapped[ <a href="#">STARPU_MAXNODES</a> ]	

## 6.42.2 Function Documentation

### 6.42.2.1 \_starpu\_memory\_node\_add\_nworkers()

```
static void _starpu_memory_node_add_nworkers (
    unsigned node ) [inline], [static]
```

Record that there is an additional worker that uses this memory node

### 6.42.2.2 \_starpu\_worker\_drives\_memory\_node()

```
void _starpu_worker_drives_memory_node (
    struct \_starpu\_worker * worker,
    unsigned memnode )
```

Record that this worker will driver data transfers for this memory node.

### 6.42.2.3 \_starpu\_memory\_node\_get\_nworkers()

```
static unsigned _starpu_memory_node_get_nworkers (
    unsigned node ) [inline], [static]
```

Get the number of workers that use this memory node

### 6.42.2.4 \_starpu\_memory\_node\_set\_mapped()

```
void _starpu_memory_node_set_mapped (
    unsigned node )
```

Note that this memory node can map CPU data

#### 6.42.2.5 `_starpu_memory_node_get_mapped()`

```
unsigned _starpu_memory_node_get_mapped (
    unsigned node )
```

Returns whether this memory node can map CPU data

#### 6.42.2.6 `_starpu_memory_node_register()`

```
unsigned _starpu_memory_node_register (
    enum starpu_node_kind kind,
    int devid )
```

Registers a memory node. Returns the memory node number

#### 6.42.2.7 `_starpu_memory_node_register_condition()`

```
void _starpu_memory_node_register_condition (
    struct _starpu_worker * worker,
    starpu_pthread_cond_t * cond,
    unsigned nodeid )
```

Register a condition variable associated to worker which is associated to a memory node itself.

#### 6.42.2.8 `_starpu_memory_node_get_description()`

```
static struct _starpu_memory_node_descr * _starpu_memory_node_get_description (
    void ) [inline], [static]
```

See [starpu\\_memory\\_node\\_get\\_description\(\)](#)

#### 6.42.2.9 `_starpu_node_get_kind()`

```
static enum starpu_node_kind _starpu_node_get_kind (
    unsigned node ) [inline], [static]
```

See [starpu\\_node\\_get\\_kind\(\)](#)

#### 6.42.2.10 `_starpu_memory_nodes_get_count()`

```
static unsigned _starpu_memory_nodes_get_count (
    void ) [inline], [static]
```

See [starpu\\_memory\\_nodes\\_get\\_count\(\)](#)

#### 6.42.2.11 `_starpu_worker_get_memory_node()`

```
static unsigned _starpu_worker_get_memory_node (
    unsigned workerid ) [inline], [static]
```

See [starpu\\_worker\\_get\\_memory\\_node\(\)](#) This workerid may either be a basic worker or a combined worker  
We have a combined worker

#### 6.42.2.12 `_starpu_worker_get_local_memory_node()`

```
static unsigned _starpu_worker_get_local_memory_node (
    void ) [inline], [static]
```

See [starpu\\_worker\\_get\\_local\\_memory\\_node](#)

## 6.43 memstats.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

## Typedefs

- typedef void \* `_starpu_memory_stats_t`

## Functions

- void `_starpu_memory_stats_init` (starpu\_data\_handle\_t handle)
- void `_starpu_memory_stats_init_per_node` (starpu\_data\_handle\_t handle, unsigned node)
- void `_starpu_memory_stats_free` (starpu\_data\_handle\_t handle)
- void `_starpu_memory_display_handle_stats` (FILE \*stream, starpu\_data\_handle\_t handle)
- void `_starpu_memory_handle_stats_cache_hit` (starpu\_data\_handle\_t handle, unsigned node)
- void `_starpu_memory_handle_stats_loaded_shared` (starpu\_data\_handle\_t handle, unsigned node)
- void `_starpu_memory_handle_stats_loaded_owner` (starpu\_data\_handle\_t handle, unsigned node)
- void `_starpu_memory_handle_stats_shared_to_owner` (starpu\_data\_handle\_t handle, unsigned node)
- void `_starpu_memory_handle_stats_invalidated` (starpu\_data\_handle\_t handle, unsigned node)

## 6.44 mp\_common.h File Reference

```
#include <semaphore.h>
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <common/barrier.h>
#include <common/thread.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/copy_driver.h>
```

## 6.45 multiple\_regression.h File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <core/perfmodel/perfmodel.h>
#include <starpu.h>
```

## Functions

- int `_starpu_multiple_regression` ([struct](#) starpu\_perfmodel\_history\_list \*ptr, double \*coeff, unsigned ncoeff, unsigned nparameters, const char \*\*parameters\_names, unsigned \*\*combinations, const char \*codelet\_name)

## 6.46 node\_ops.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <datawizard/copy_driver.h>
```

## Data Structures

- [struct \\_starpu\\_node\\_ops](#)

## Typedefs

- typedef int(\* [copy\\_interface\\_func\\_t](#)) (starpu\_data\_handle\_t handle, void \*src\_interface, unsigned src\_node, void \*dst\_interface, unsigned dst\_node, [struct \\_starpu\\_data\\_request](#) \*req)
- typedef int(\* [copy\\_data\\_t](#)) (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t dst\_offset, unsigned dst\_node, size\_t ssize, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- typedef int(\* [copy2d\\_data\\_t](#)) (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks, size\_t ld\_src, size\_t ld\_dst, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- typedef int(\* [copy3d\\_data\\_t](#)) (uintptr\_t src\_ptr, size\_t src\_offset, unsigned src\_node, uintptr\_t dst\_ptr, size\_t dst\_offset, unsigned dst\_node, size\_t blocksize, size\_t numblocks\_1, size\_t ld1\_src, size\_t ld1\_dst, size\_t numblocks\_2, size\_t ld2\_src, size\_t ld2\_dst, [struct \\_starpu\\_async\\_channel](#) \*async\_channel)
- typedef uintptr\_t(\* [map\\_t](#)) (uintptr\_t src, size\_t src\_offset, unsigned src\_node, unsigned dst\_node, size\_t size, int \*ret)
- typedef int(\* [unmap\\_t](#)) (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, unsigned dst\_node, size\_t size)
- typedef int(\* [update\\_map\\_t](#)) (uintptr\_t src, size\_t src\_offset, unsigned src\_node, uintptr\_t dst, size\_t dst\_offset, unsigned dst\_node, size\_t size)

## Functions

- const char \* [\\_starpu\\_node\\_get\\_prefix](#) (enum [starpu\\_node\\_kind](#) kind)

### 6.46.1 Typedef Documentation

#### 6.46.1.1 copy\_interface\_func\_t

```
typedef int(* copy_interface_func_t) (starpu_data_handle_t handle, void *src_interface, unsigned src_node, void *dst_interface, unsigned dst_node, struct \_starpu\_data\_request *req)
```

Request copying some data interface for handle handle: from interface src\_interface that exists on node src\_node to interface dst\_interface that exists on node dst\_node.

If req is non-NULL, this can be used to start an asynchronous copy, in which case -EAGAIN should be returned. Otherwise, 0 should be returned.

[\\_starpu\\_copy\\_interface\\_any\\_to\\_any](#) can be used as a generic version, that assumes that the data\_interface implements the any\_to\_any method, and copy\_data\_t will be used to queue the actual transfers.

#### 6.46.1.2 copy\_data\_t

```
typedef int(* copy_data_t) (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size_t dst_offset, unsigned dst_node, size_t ssize, struct \_starpu\_async\_channel *async_channel)
```

Request copying ssize bytes of data from src\_ptr (plus offset src\_offset) in node src\_node to dst\_ptr (plus offset dst\_offset) in node dst\_node.

If async\_channel is non-NULL, this can be used to start an asynchronous copy, in which case -EAGAIN should be returned. Otherwise, 0 should be returned.

#### 6.46.1.3 copy2d\_data\_t

```
typedef int(* copy2d_data_t) (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks, size_t ld_src, size_t ld_dst, struct \_starpu\_async\_channel *async_channel)
```

This is like copy\_data\_t, except that there are numblocks blocks of size blocksize bytes to be transferred. On the source, their respective starts are ld\_src bytes apart, and on the destination their respective starts have to be ld\_dst bytes apart. (leading dimension)

#### 6.46.1.4 copy3d\_data\_t

```
typedef int(* copy3d_data_t) (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks_1, size_t ld1_src, size_t ld1_dst, size_t numblocks_2, size_t ld2_src, size_t ld2_dst, struct _starpu_async_channel *async_channel)
```

This is like `copy_data_t`, except that there are `numblocks_2` metablocks to be transferred. On the source, their respective starts are `ld2_src` bytes apart, and on the destination their respective starts have to be `ld2_dst` bytes apart.

The metablocks are composed of `numblocks_1` blocks of size `blocksize` bytes. On the source, their respective starts are `ld1_src` bytes apart, and on the destination their respective starts have to be `ld1_dst` bytes apart.

#### 6.46.1.5 map\_t

```
typedef uintptr_t(* map_t) (uintptr_t src, size_t src_offset, unsigned src_node, unsigned dst_node, size_t size, int *ret)
```

Map `size` bytes of data from `src` (plus offset `src_offset`) in node `src_node` on node `dst_node`. If successful, return the resulting pointer, otherwise fill `*ret`

#### 6.46.1.6 unmap\_t

```
typedef int(* unmap_t) (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, unsigned dst_node, size_t size)
```

Unmap `size` bytes of data from `src` (plus offset `src_offset`) in node `src_node` on node `dst_node`.

#### 6.46.1.7 update\_map\_t

```
typedef int(* update_map_t) (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size)
```

Update cache coherency for the mapping of `size` bytes of data from `src` (plus offset `src_offset`) in node `src_node` on node `dst_node`.

## 6.47 openmp\_runtime\_support.h File Reference

```
#include <starpu.h>
#include <common/list.h>
#include <common/starpu_spinlock.h>
#include <common/uthash.h>
#include <ucontext.h>
```

### Data Structures

- struct [starpu\\_omp\\_numeric\\_place](#)
- struct [starpu\\_omp\\_place](#)
- struct [starpu\\_omp\\_data\\_environment\\_icvs](#)
- struct [starpu\\_omp\\_device\\_icvs](#)
- struct [starpu\\_omp\\_implicit\\_task\\_icvs](#)
- struct [starpu\\_omp\\_global\\_icvs](#)
- struct [starpu\\_omp\\_initial\\_icv\\_values](#)
- struct [starpu\\_omp\\_task\\_group](#)
- struct [starpu\\_omp\\_task\\_link](#)
- struct [starpu\\_omp\\_condition](#)
- struct [starpu\\_omp\\_critical](#)

## Macros

- #define [\\_XOPEN\\_SOURCE](#)
- #define [STARPU\\_OMP\\_MAX\\_ACTIVE\\_LEVELS](#)

## Enumerations

- enum [starpu\\_omp\\_place\\_name](#) { [starpu\\_omp\\_place\\_undefined](#) , [starpu\\_omp\\_place\\_threads](#) , [starpu\\_omp\\_place\\_cores](#) , [starpu\\_omp\\_place\\_sockets](#) , [starpu\\_omp\\_place\\_numerical](#) }
- enum [starpu\\_omp\\_task\\_state](#) { [starpu\\_omp\\_task\\_state\\_clear](#) , [starpu\\_omp\\_task\\_state\\_preempted](#) , [starpu\\_omp\\_task\\_state\\_terminated](#) , [starpu\\_omp\\_task\\_state\\_zombie](#) , [starpu\\_omp\\_task\\_state\\_target](#) }
- enum [starpu\\_omp\\_task\\_wait\\_on](#) { [starpu\\_omp\\_task\\_wait\\_on\\_task\\_childs](#) , [starpu\\_omp\\_task\\_wait\\_on\\_region\\_tasks](#) , [starpu\\_omp\\_task\\_wait\\_on\\_barrier](#) , [starpu\\_omp\\_task\\_wait\\_on\\_group](#) , [starpu\\_omp\\_task\\_wait\\_on\\_critical](#) , [starpu\\_omp\\_task\\_wait\\_on\\_ordered](#) , [starpu\\_omp\\_task\\_wait\\_on\\_lock](#) , [starpu\\_omp\\_task\\_wait\\_on\\_nest\\_lock](#) }
- enum [starpu\\_omp\\_task\\_flags](#) { [STARPU\\_OMP\\_TASK\\_FLAGS\\_IMPLICIT](#) , [STARPU\\_OMP\\_TASK\\_FLAGS\\_UNDEFERRED](#) , [STARPU\\_OMP\\_TASK\\_FLAGS\\_FINAL](#) , [STARPU\\_OMP\\_TASK\\_FLAGS\\_UNTIED](#) }

## Variables

- [starpu\\_pthread\\_key\\_t omp\\_thread\\_key](#)
- [starpu\\_pthread\\_key\\_t omp\\_task\\_key](#)

### 6.47.1 Data Structure Documentation

#### 6.47.1.1 struct starpu\_omp\_numeric\_place

##### Data Fields

int	<a href="#">excluded_place</a>	
int *	<a href="#">included_numeric_items</a>	
int	<a href="#">nb_included_numeric_items</a>	
int *	<a href="#">excluded_numeric_items</a>	
int	<a href="#">nb_excluded_numeric_items</a>	

#### 6.47.1.2 struct starpu\_omp\_place

OpenMP place for thread affinity, defined by the OpenMP spec

##### Data Fields

	int	<a href="#">abstract_name</a>	
	int	<a href="#">abstract_excluded</a>	
	int	<a href="#">abstract_length</a>	
	<a href="#">struct starpu_omp_numeric_place *</a>	<a href="#">numeric_places</a>	
	int	<a href="#">nb_numeric_places</a>	

#### 6.47.1.3 struct starpu\_omp\_data\_environment\_icvs

Internal Control Variables (ICVs) declared following OpenMP 4.0.0 spec section 2.3.1

## Data Fields

int	dyn_var	parallel region icvs
int	nest_var	
int *	nthreads_var	
int	thread_limit_var	nthreads_var ICV is a list
int	active_levels_var	
int	levels_var	
int *	bind_var	
int	run_sched_var	bind_var ICV is a list loop region icvs
unsigned long long	run_sched_chunk_var	
int	default_device_var	program execution icvs
int	max_task_priority_var	

**6.47.1.4 struct starpu\_omp\_device\_icvs**

## Data Fields

int	max_active_levels_var	parallel region icvs
int	def_sched_var	loop region icvs
unsigned long long	def_sched_chunk_var	
int	stacksize_var	program execution icvs
int	wait_policy_var	

**6.47.1.5 struct starpu\_omp\_implicit\_task\_icvs**

## Data Fields

int	place_partition_var	parallel region icvs
-----	---------------------	----------------------

**6.47.1.6 struct starpu\_omp\_global\_icvs**

## Data Fields

int	cancel_var	program execution icvs
-----	------------	------------------------

**6.47.1.7 struct starpu\_omp\_initial\_icv\_values**

## Data Fields

int	dyn_var	
int	nest_var	
int *	nthreads_var	
int	run_sched_var	
unsigned long long	run_sched_chunk_var	
int	def_sched_var	
unsigned long long	def_sched_chunk_var	
int *	bind_var	
int	stacksize_var	
int	wait_policy_var	

## Data Fields

	int	thread_limit_var	
	int	max_active_levels_var	
	int	active_levels_var	
	int	levels_var	
	int	place_partition_var	
	int	cancel_var	
	int	default_device_var	
	int	max_task_priority_var	
<a href="#">struct starpu_omp_place</a>		places	not a real ICV, but needed to store the contents of OMP_PLACES

## 6.47.1.8 struct starpu\_omp\_task\_group

## Data Fields

	int	descendent_task_count	
<a href="#">struct starpu_omp_task *</a>		leader_task	
<a href="#">struct starpu_omp_task_group *</a>		p_previous_task_group	

## 6.47.1.9 struct starpu\_omp\_task\_link

## Data Fields

<a href="#">struct starpu_omp_task *</a>	task	
<a href="#">struct starpu_omp_task_link *</a>	next	

## 6.47.1.10 struct starpu\_omp\_condition

## Data Fields

<a href="#">struct starpu_omp_task_link *</a>	contention_list_head	
---	----------------------	--

## 6.47.1.11 struct starpu\_omp\_critical

## Data Fields

<a href="#">UT_hash_handle</a>	hh	
<a href="#">struct starpu_spinlock</a>	lock	
unsigned	state	
<a href="#">struct starpu_omp_task_link *</a>	contention_list_head	
const char *	name	

## 6.47.2 Macro Definition Documentation

## 6.47.2.1 \_XOPEN\_SOURCE

```
#define _XOPEN_SOURCE
```

ucontexts have been deprecated as of POSIX 1-2004 `_XOPEN_SOURCE` required at least on OS/X

TODO: add detection in configure.ac

### 6.47.2.2 STARPU\_OMP\_MAX\_ACTIVE\_LEVELS

```
#define STARPU_OMP_MAX_ACTIVE_LEVELS
```

Arbitrary limit on the number of nested parallel sections

## 6.47.3 Enumeration Type Documentation

### 6.47.3.1 starpu\_omp\_place\_name

```
enum starpu_omp_place_name
```

Possible abstract names for OpenMP places

### 6.47.3.2 starpu\_omp\_task\_state

```
enum starpu_omp_task_state
```

Enumerator

starpu_omp_task_state_target	target tasks are non-preemptible tasks, without dedicated stack and OpenMP Runtime Support context
------------------------------	--

## 6.48 perfmodel.h File Reference

```
#include <common/config.h>
#include <starpu.h>
#include <core/task_bundle.h>
#include <stdio.h>
```

### Data Structures

- struct [\\_starpu\\_perfmodel\\_state](#)

### Macros

- #define [\\_STARPU\\_PERFMODEL\\_VERSION](#)
- #define [PATH\\_LENGTH](#)
- #define [STR\\_SHORT\\_LENGTH](#)
- #define [STR\\_LONG\\_LENGTH](#)
- #define [STR\\_VERY\\_LONG\\_LENGTH](#)

### Functions

- void [\\_starpu\\_init\\_perfmodel](#) (void)
- void [\\_starpu\\_find\\_perf\\_model\\_codelet](#) (const char \*symbol, const char \*hostname, char \*path, size\_t maxlen)
- void [\\_starpu\\_find\\_perf\\_model\\_codelet\\_debug](#) (const char \*symbol, const char \*hostname, const char \*arch, char \*path, size\_t maxlen)
- void [\\_starpu\\_set\\_default\\_perf\\_model\\_codelet](#) (const char \*symbol, const char \*hostname, char \*path, size\_t maxlen)
- char \* [\\_starpu\\_get\\_perf\\_model\\_dir\\_default](#) ()
- char \*\* [\\_starpu\\_get\\_perf\\_model\\_dirs\\_codelet](#) () STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

- char \* [\\_starpu\\_get\\_perf\\_model\\_dir\\_bus](#) ()
- double [\\_starpu\\_history\\_based\\_job\\_expected\\_perf](#) ([struct starpu\\_perfmodel](#) \*model, [struct starpu\\_perfmodel\\_arch](#) \*arch, [struct \\_starpu\\_job](#) \*j, unsigned nimpl)
- double [\\_starpu\\_history\\_based\\_job\\_expected\\_deviation](#) ([struct starpu\\_perfmodel](#) \*model, [struct starpu\\_perfmodel\\_arch](#) \*arch, [struct \\_starpu\\_job](#) \*j, unsigned nimpl)
- void [\\_starpu\\_load\\_history\\_based\\_model](#) ([struct starpu\\_perfmodel](#) \*model, unsigned scan\_history)
- void [\\_starpu\\_init\\_and\\_load\\_perfmodel](#) ([struct starpu\\_perfmodel](#) \*model)
- void [\\_starpu\\_initialize\\_registered\\_performance\\_models](#) (void)
- void [\\_starpu\\_deinitialize\\_registered\\_performance\\_models](#) (void)
- void [\\_starpu\\_deinitialize\\_performance\\_model](#) ([struct starpu\\_perfmodel](#) \*model)
- double [\\_starpu\\_regression\\_based\\_job\\_expected\\_perf](#) ([struct starpu\\_perfmodel](#) \*model, [struct starpu\\_perfmodel\\_arch](#) \*arch, [struct \\_starpu\\_job](#) \*j, unsigned nimpl)
- double [\\_starpu\\_non\\_linear\\_regression\\_based\\_job\\_expected\\_perf](#) ([struct starpu\\_perfmodel](#) \*model, [struct starpu\\_perfmodel\\_arch](#) \*arch, [struct \\_starpu\\_job](#) \*j, unsigned nimpl)
- double [\\_starpu\\_multiple\\_regression\\_based\\_job\\_expected\\_perf](#) ([struct starpu\\_perfmodel](#) \*model, [struct starpu\\_perfmodel\\_arch](#) \*arch, [struct \\_starpu\\_job](#) \*j, unsigned nimpl)
- void [\\_starpu\\_update\\_perfmodel\\_history](#) ([struct \\_starpu\\_job](#) \*j, [struct starpu\\_perfmodel](#) \*model, [struct starpu\\_perfmodel\\_arch](#) \*arch, unsigned cpuid, double measured, unsigned nimpl, unsigned number)
- int [\\_starpu\\_perfmodel\\_create\\_comb\\_if\\_needed](#) ([struct starpu\\_perfmodel\\_arch](#) \*arch)
- int [\\_starpu\\_create\\_bus\\_sampling\\_directory\\_if\\_needed](#) (int location)
- void [\\_starpu\\_create\\_codelet\\_sampling\\_directory\\_if\\_needed](#) (int location)
- void [\\_starpu\\_load\\_bus\\_performance\\_files](#) (void)
- void [\\_starpu\\_init\\_bus\\_performance](#) (void)
- int [\\_starpu\\_get\\_perf\\_model\\_bus](#) ()
- int [\\_starpu\\_set\\_default\\_perf\\_model\\_bus](#) ()
- void [\\_starpu\\_set\\_calibrate\\_flag](#) (unsigned val)
- unsigned [\\_starpu\\_get\\_calibrate\\_flag](#) (void)
- unsigned \* [\\_starpu\\_get\\_cuda\\_affinity\\_vector](#) (unsigned gpuid)
- unsigned \* [\\_starpu\\_get\\_opencil\\_affinity\\_vector](#) (unsigned gpuid)
- void [\\_starpu\\_save\\_bandwidth\\_and\\_latency\\_disk](#) (double bandwidth\_write, double bandwidth\_read, double latency\_write, double latency\_read, unsigned node, const char \*name)
- void [\\_starpu\\_write\\_double](#) (FILE \*f, const char \*format, double val) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- int [\\_starpu\\_read\\_double](#) (FILE \*f, char \*format, double \*val) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- void [\\_starpu\\_simgrid\\_get\\_platform\\_path](#) (int version, char \*path, size\_t maxlen)
- void [\\_starpu\\_perfmodel\\_realloc](#) ([struct starpu\\_perfmodel](#) \*model, int nb)
- void [\\_starpu\\_free\\_arch\\_combs](#) (void)
- hwloc\_topology\_t [\\_starpu\\_perfmodel\\_get\\_hwtopology](#) ()

## Variables

- unsigned [\\_starpu\\_calibration\\_minimum](#)

## 6.48.1 Data Structure Documentation

### 6.48.1.1 struct starpu\_perfmodel\_state

#### Data Fields

<a href="#">struct starpu_perfmodel_per_arch</a> **	per_arch	
int **	per_arch_is_set	
starpu_pthread_rwlock_t	model_rwlock	
int *	nimpls	
int *	nimpls_set	

## Data Fields

	int	ncombs	The number of combinations currently used by the model
	int	ncombs_set	The number of combinations allocated in the array nimpls and ncombs
	int *	combs	

## 6.48.2 Macro Definition Documentation

### 6.48.2.1 `_STARPU_PERFMODEL_VERSION`

```
#define _STARPU_PERFMODEL_VERSION
```

Performance models files are stored in a directory whose name include the version of the performance model format. The version number is also written in the file itself. When updating the format, the variable `_STARPU_PERFMODEL_VERSION` should be updated. It is then possible to switch easily between different versions of StarPU having different performance model formats.

## 6.49 `prio_deque.h` File Reference

```
#include <core/task.h>
```

### Data Structures

- struct [starpu\\_st\\_prio\\_deque](#)

### 6.49.1 Data Structure Documentation

#### 6.49.1.1 struct `starpu_st_prio_deque`

## Data Fields

<a href="#">struct</a> <code>starpu_task_prio_list</code>	list	
unsigned	ntasks	
unsigned	nprocessed	
double	exp_start	
double	exp_end	
double	exp_len	

## 6.50 `prio_list.h` File Reference

```
#include <common/rbtree.h>
```

### Macros

- #define `PRIO_LIST_INLINE`
- #define `PRIO_struct`
- #define `PRIO_LIST_CREATE_TYPE(ENAME, PRIOFIELD)`

## 6.51 profiling.h File Reference

```
#include <starp_u.h>
#include <starp_u_profiling.h>
#include <starp_u_util.h>
#include <common/config.h>
```

### Functions

- [struct](#) `starp_u_profiling_task_info` \* [\\_starp\\_u\\_allocate\\_profiling\\_info\\_if\\_needed](#) ([struct](#) `starp_u_task` \*`task`)
- [void](#) [\\_starp\\_u\\_worker\\_update\\_profiling\\_info\\_executing](#) (`int` `workerid`, `int` `executed_tasks`, `uint64_t` `used_cycles`, `uint64_t` `stall_cycles`, `double` `consumed_energy`, `double` `flops`)
- [void](#) [\\_starp\\_u\\_worker\\_start\\_state](#) (`int` `workerid`, `enum` [\\_starp\\_u\\_worker\\_status\\_index](#) `index`, [struct](#) `timespec` \*`start_time`)
- [void](#) [\\_starp\\_u\\_worker\\_stop\\_state](#) (`int` `workerid`, `enum` [\\_starp\\_u\\_worker\\_status\\_index](#) `index`, [struct](#) `timespec` \*`stop_time`)
- [void](#) [\\_starp\\_u\\_initialize\\_busid\\_matrix](#) (`void`)
- [int](#) [\\_starp\\_u\\_register\\_bus](#) (`int` `src_node`, `int` `dst_node`)
- [void](#) [\\_starp\\_u\\_bus\\_update\\_profiling\\_info](#) (`int` `src_node`, `int` `dst_node`, `size_t` `size`)
- [void](#) [\\_starp\\_u\\_profiling\\_set\\_task\\_push\\_start\\_time](#) ([struct](#) `starp_u_task` \*`task`)
- [void](#) [\\_starp\\_u\\_profiling\\_set\\_task\\_push\\_end\\_time](#) ([struct](#) `starp_u_task` \*`task`)
- [void](#) [\\_starp\\_u\\_profiling\\_init](#) (`void`)
- [void](#) [\\_starp\\_u\\_profiling\\_start](#) (`void`)
- [void](#) [\\_starp\\_u\\_profiling\\_terminate](#) (`void`)

### Variables

- `int` [\\_starp\\_u\\_codelet\\_profiling](#)
- `int` [\\_starp\\_u\\_energy\\_profiling](#)

### 6.51.1 Function Documentation

#### 6.51.1.1 [\\_starp\\_u\\_allocate\\_profiling\\_info\\_if\\_needed\(\)](#)

```
struct starp_u_profiling_task_info * \_starp\_u\_allocate\_profiling\_info\_if\_needed (
    struct starp_u_task * task )
```

Create a task profiling info structure (with the proper time stamps) in case profiling is enabled.

#### 6.51.1.2 [\\_starp\\_u\\_worker\\_update\\_profiling\\_info\\_executing\(\)](#)

```
void \_starp\_u\_worker\_update\_profiling\_info\_executing (
    int workerid,
    int executed_tasks,
    uint64_t used_cycles,
    uint64_t stall_cycles,
    double consumed_energy,
    double flops )
```

Update the per-worker profiling info after a task (or more) was executed. This tells StarPU how much time was spent doing computation.

### 6.51.1.3 `_starpu_worker_start_state()`

```
void _starpu_worker_start_state (
    int workerid,
    enum _starpu_worker_status_index index,
    struct timespec * start_time )
```

Record the date when the worker entered this state. This permits to measure how much time was spent in this state. `start_time` is optional, if unspecified, `_starpu_worker_start_state` will just take the current time.

### 6.51.1.4 `_starpu_initialize_busid_matrix()`

```
void _starpu_initialize_busid_matrix (
    void )
```

When StarPU is initialized, a matrix describing all the bus between memory nodes is created: it indicates whether there is a physical link between two memory nodes or not. This matrix should contain the identifier of the bus between two nodes or -1 in case there is no link.

### 6.51.1.5 `_starpu_register_bus()`

```
int _starpu_register_bus (
    int src_node,
    int dst_node )
```

Tell StarPU that there exists a link between the two memory nodes. This function returns the identifier associated to the bus which can be used to retrieve profiling information about the bus activity later on.

### 6.51.1.6 `_starpu_bus_update_profiling_info()`

```
void _starpu_bus_update_profiling_info (
    int src_node,
    int dst_node,
    size_t size )
```

Tell StarPU that "size" bytes were transferred between the two specified memory nodes.

### 6.51.1.7 `_starpu_profiling_init()`

```
void _starpu_profiling_init (
    void )
```

This function needs to be called before other `starpu_profile_*` functions

### 6.51.1.8 `_starpu_profiling_start()`

```
void _starpu_profiling_start (
    void )
```

This function starts profiling if the `STARPU_PROFILING` environment variable was set

## 6.52 `progress_hook.h` File Reference

### Functions

- `void _starpu_init_progression_hooks` (void)
- `unsigned _starpu_execute_registered_progression_hooks` (void)

## 6.53 `rbtree.h` File Reference

```
#include <stddef.h>
#include <assert.h>
#include <stdint.h>
#include <sys/types.h>
```

```
#include <starp_uutil.h>
#include "rbtree_i.h"
```

## Macros

- #define **MACRO\_BEGIN**
- #define **MACRO\_END**
- #define **STARPU\_RBTREE\_LEFT**
- #define **STARPU\_RBTREE\_RIGHT**
- #define [STARPU\\_RBTREE\\_INITIALIZER](#)
- #define [starp\\_u\\_rbtree\\_entry](#)(node, type, member)
- #define [starp\\_u\\_rbtree\\_lookup](#)(tree, key, cmp\_fn)
- #define [starp\\_u\\_rbtree\\_lookup\\_nearest](#)(tree, key, cmp\_fn, dir)
- #define [starp\\_u\\_rbtree\\_insert](#)(tree, node, cmp\_fn)
- #define [starp\\_u\\_rbtree\\_lookup\\_slot](#)(tree, key, cmp\_fn, slot)
- #define [starp\\_u\\_rbtree\\_first](#)(tree)
- #define [starp\\_u\\_rbtree\\_last](#)(tree)
- #define [starp\\_u\\_rbtree\\_prev](#)(node)
- #define [starp\\_u\\_rbtree\\_next](#)(node)
- #define [starp\\_u\\_rbtree\\_for\\_each\\_remove](#)(tree, node, tmp)

## Functions

- static void [starp\\_u\\_rbtree\\_init](#) (struct [starp\\_u\\_rbtree](#) \*tree)
- static void [starp\\_u\\_rbtree\\_init0](#) (struct [starp\\_u\\_rbtree](#) \*tree STARPU\_ATTRIBUTE\_UNUSED)
- static void [starp\\_u\\_rbtree\\_node\\_init](#) (struct [starp\\_u\\_rbtree\\_node](#) \*node)
- static void [starp\\_u\\_rbtree\\_node\\_init0](#) (struct [starp\\_u\\_rbtree\\_node](#) \*node)
- static int [starp\\_u\\_rbtree\\_node\\_unlinked](#) (const struct [starp\\_u\\_rbtree\\_node](#) \*node)
- static int [starp\\_u\\_rbtree\\_empty](#) (const struct [starp\\_u\\_rbtree](#) \*tree)
- static void [starp\\_u\\_rbtree\\_insert\\_slot](#) (struct [starp\\_u\\_rbtree](#) \*tree, uintptr\_t slot, struct [starp\\_u\\_rbtree\\_node](#) \*node)
- void [starp\\_u\\_rbtree\\_remove](#) (struct [starp\\_u\\_rbtree](#) \*tree, struct [starp\\_u\\_rbtree\\_node](#) \*node)

### 6.53.1 Macro Definition Documentation

#### 6.53.1.1 STARPU\_RBTREE\_INITIALIZER

```
#define STARPU_RBTREE_INITIALIZER
Static tree initializer.
```

#### 6.53.1.2 starp\_u\_rbtree\_entry

```
#define starp_u_rbtree_entry(
    node,
    type,
    member )
```

Macro that evaluates to the address of the structure containing the given node based on the given type and member.

### 6.53.1.3 starpu\_rbtrees\_lookup

```
#define starpu_rbtrees_lookup(
    tree,
    key,
    cmp_fn )
```

Look up a node in a tree.

Note that implementing the lookup algorithm as a macro gives two benefits: First, it avoids the overhead of a callback function. Next, the type of the `cmp_fn` parameter isn't rigid. The only guarantee offered by this implementation is that the key parameter is the first parameter given to `cmp_fn`. This way, users can pass only the value they need for comparison instead of e.g. allocating a full structure on the stack.

See [starpu\\_rbtrees\\_insert\(\)](#).

### 6.53.1.4 starpu\_rbtrees\_lookup\_nearest

```
#define starpu_rbtrees_lookup_nearest(
    tree,
    key,
    cmp_fn,
    dir )
```

Look up a node or one of its nearest nodes in a tree.

This macro essentially acts as [starpu\\_rbtrees\\_lookup\(\)](#) but if no entry matched the key, an additional step is performed to obtain the next or previous node, depending on the direction (left or right).

The constraints that apply to the key parameter are the same as for [starpu\\_rbtrees\\_lookup\(\)](#).

### 6.53.1.5 starpu\_rbtrees\_insert

```
#define starpu_rbtrees_insert(
    tree,
    node,
    cmp_fn )
```

Insert a node in a tree.

This macro performs a standard lookup to obtain the insertion point of the given node in the tree (it is assumed that the inserted node never compares equal to any other entry in the tree) and links the node. It then checks red-black rules violations, and rebalances the tree if necessary.

Unlike [starpu\\_rbtrees\\_lookup\(\)](#), the `cmp_fn` parameter must compare two complete entries, so it is suggested to use two different comparison inline functions, such as `myobj_cmp_lookup()` and `myobj_cmp_insert()`. There is no guarantee about the order of the nodes given to the comparison function.

See [starpu\\_rbtrees\\_lookup\(\)](#).

### 6.53.1.6 starpu\_rbtrees\_lookup\_slot

```
#define starpu_rbtrees_lookup_slot(
    tree,
    key,
    cmp_fn,
    slot )
```

Look up a node/slot pair in a tree.

This macro essentially acts as [starpu\\_rbtrees\\_lookup\(\)](#) but in addition to a node, it also returns a slot, which identifies an insertion point in the tree. If the returned node is null, the slot can be used by [starpu\\_rbtrees\\_insert\\_slot\(\)](#) to insert without the overhead of an additional lookup. The slot is a simple `uintptr_t` integer.

The constraints that apply to the key parameter are the same as for [starpu\\_rbtrees\\_lookup\(\)](#).

### 6.53.1.7 starpu\_rbtrees\_first

```
#define starpu_rbtrees_first(
    tree )
```

Return the first node of a tree.

### 6.53.1.8 starpu\_rbtree\_last

```
#define starpu_rbtree_last(  
    tree )
```

Return the last node of a tree.

### 6.53.1.9 starpu\_rbtree\_prev

```
#define starpu_rbtree_prev(  
    node )
```

Return the node previous to the given node.

### 6.53.1.10 starpu\_rbtree\_next

```
#define starpu_rbtree_next(  
    node )
```

Return the node next to the given node.

### 6.53.1.11 starpu\_rbtree\_for\_each\_remove

```
#define starpu_rbtree_for_each_remove(  
    tree,  
    node,  
    tmp )
```

Forge a loop to process all nodes of a tree, removing them when visited.

This macro can only be used to destroy a tree, so that the resources used by the entries can be released by the user. It basically removes all nodes without doing any color checking.

After completion, all nodes and the tree root member are stale.

## 6.53.2 Function Documentation

### 6.53.2.1 starpu\_rbtree\_init()

```
static void starpu_rbtree_init (  
    struct starpu_rbtree * tree ) [inline], [static]
```

Initialize a tree.

### 6.53.2.2 starpu\_rbtree\_init0()

```
static void starpu_rbtree_init0 (  
    struct starpu_rbtree *tree STARPU_ATTRIBUTE_UNUSED ) [inline], [static]
```

This version assumes that the content of tree was already zeroed

### 6.53.2.3 starpu\_rbtree\_node\_init()

```
static void starpu_rbtree_node_init (  
    struct starpu_rbtree_node * node ) [inline], [static]
```

Initialize a node.

A node is in no tree when its parent points to itself.

### 6.53.2.4 starpu\_rbtree\_node\_init0()

```
static void starpu_rbtree_node_init0 (  
    struct starpu_rbtree_node * node ) [inline], [static]
```

This version assumes that the content of node was already zeroed

### 6.53.2.5 starpu\_rbtrees\_node\_unlinked()

```
static int starpu_rbtrees_node_unlinked (
    const struct starpu_rbtrees_node * node ) [inline], [static]
```

Return true if node is in no tree.

### 6.53.2.6 starpu\_rbtrees\_empty()

```
static int starpu_rbtrees_empty (
    const struct starpu_rbtrees * tree ) [inline], [static]
```

Return true if tree is empty.

### 6.53.2.7 starpu\_rbtrees\_insert\_slot()

```
static void starpu_rbtrees_insert_slot (
    struct starpu_rbtrees * tree,
    uintptr_t slot,
    struct starpu_rbtrees_node * node ) [inline], [static]
```

Insert a node at an insertion point in a tree.

This macro essentially acts as `starpu_rbtrees_insert()` except that it doesn't obtain the insertion point with a standard lookup. The insertion point is obtained by calling `starpu_rbtrees_lookup_slot()`. In addition, the new node must not compare equal to an existing node in the tree (i.e. the slot must denote a null node).

### 6.53.2.8 starpu\_rbtrees\_remove()

```
void starpu_rbtrees_remove (
    struct starpu_rbtrees * tree,
    struct starpu_rbtrees_node * node )
```

Remove a node from a tree.

After completion, the node is stale.

## 6.54 rbtrees\_i.h File Reference

```
#include <assert.h>
```

### Data Structures

- struct [starpu\\_rbtrees\\_node](#)
- struct [starpu\\_rbtrees](#)

### Macros

- #define [STARPU\\_RBTREE\\_COLOR\\_MASK](#)
- #define [STARPU\\_RBTREE\\_PARENT\\_MASK](#)
- #define [STARPU\\_RBTREE\\_COLOR\\_RED](#)
- #define [STARPU\\_RBTREE\\_COLOR\\_BLACK](#)
- #define [STARPU\\_RBTREE\\_SLOT\\_INDEX\\_MASK](#)
- #define [STARPU\\_RBTREE\\_SLOT\\_PARENT\\_MASK](#)

### Functions

- static int [starpu\\_rbtrees\\_check\\_alignment](#) (const struct starpu\_rbtrees\_node \*node)
- static int [starpu\\_rbtrees\\_check\\_index](#) (int index)
- static int [starpu\\_rbtrees\\_d2i](#) (int diff)
- static struct starpu\_rbtrees\_node \* [starpu\\_rbtrees\\_parent](#) (const struct starpu\_rbtrees\_node \*node)
- static uintptr\_t [starpu\\_rbtrees\\_slot](#) (struct starpu\_rbtrees\_node \*parent, int index)
- static struct starpu\_rbtrees\_node \* [starpu\\_rbtrees\\_slot\\_parent](#) (uintptr\_t slot)

- static int `starpu_rbtree_slot_index` (uintptr\_t slot)
- void `starpu_rbtree_insert_rebalance` (struct `starpu_rbtree` \*tree, struct `starpu_rbtree_node` \*parent, int index, struct `starpu_rbtree_node` \*node)
- struct `starpu_rbtree_node` \* `starpu_rbtree_nearest` (struct `starpu_rbtree_node` \*parent, int index, int direction)
- struct `starpu_rbtree_node` \* `starpu_rbtree_firstlast` (const struct `starpu_rbtree` \*tree, int direction)
- struct `starpu_rbtree_node` \* `starpu_rbtree_walk` (struct `starpu_rbtree_node` \*node, int direction)
- struct `starpu_rbtree_node` \* `starpu_rbtree_postwalk_deepest` (const struct `starpu_rbtree` \*tree)
- struct `starpu_rbtree_node` \* `starpu_rbtree_postwalk_unlink` (struct `starpu_rbtree_node` \*node)

## 6.54.1 Data Structure Documentation

### 6.54.1.1 struct `starpu_rbtree_node`

Red-black node structure.

To reduce the number of branches and the instruction cache footprint, the left and right child pointers are stored in an array, and the symmetry of most tree operations is exploited by using left/right variables when referring to children.

In addition, this implementation assumes that all nodes are 4-byte aligned, so that the least significant bit of the parent member can be used to store the color of the node. This is true for all modern 32 and 64 bits architectures, as long as the nodes aren't embedded in structures with special alignment constraints such as member packing.

#### Data Fields

uintptr_t	parent	
struct <code>starpu_rbtree_node</code> *	children[2]	

### 6.54.1.2 struct `starpu_rbtree`

Red-black tree structure.

#### Data Fields

struct <code>starpu_rbtree_node</code> *	root	
--	------	--

## 6.54.2 Macro Definition Documentation

### 6.54.2.1 `STARPU_RBTREE_COLOR_MASK`

```
#define STARPU_RBTREE_COLOR_MASK
```

Masks applied on the parent member of a node to obtain either the color or the parent address.

### 6.54.2.2 `STARPU_RBTREE_COLOR_RED`

```
#define STARPU_RBTREE_COLOR_RED
```

Node colors.

### 6.54.2.3 `STARPU_RBTREE_SLOT_INDEX_MASK`

```
#define STARPU_RBTREE_SLOT_INDEX_MASK
```

Masks applied on slots to obtain either the child index or the parent address.

## 6.54.3 Function Documentation

**6.54.3.1 starpu\_rbtrees\_check\_alignment()**

```
static int starpu_rbtrees_check_alignment (
    const struct starpu_rbtrees_node * node ) [inline], [static]
```

Return true if the given pointer is suitably aligned.

**6.54.3.2 starpu\_rbtrees\_check\_index()**

```
static int starpu_rbtrees_check_index (
    int index ) [inline], [static]
```

Return true if the given index is a valid child index.

**6.54.3.3 starpu\_rbtrees\_d2i()**

```
static int starpu_rbtrees_d2i (
    int diff ) [inline], [static]
```

Convert the result of a comparison into an index in the children array (0 or 1).

This function is mostly used when looking up a node.

**6.54.3.4 starpu\_rbtrees\_parent()**

```
static struct starpu_rbtrees_node * starpu_rbtrees_parent (
    const struct starpu_rbtrees_node * node ) [inline], [static]
```

Return the parent of a node.

**6.54.3.5 starpu\_rbtrees\_slot()**

```
static uintptr_t starpu_rbtrees_slot (
    struct starpu_rbtrees_node * parent,
    int index ) [inline], [static]
```

Translate an insertion point into a slot.

**6.54.3.6 starpu\_rbtrees\_slot\_parent()**

```
static struct starpu_rbtrees_node * starpu_rbtrees_slot_parent (
    uintptr_t slot ) [inline], [static]
```

Extract the parent address from a slot.

**6.54.3.7 starpu\_rbtrees\_slot\_index()**

```
static int starpu_rbtrees_slot_index (
    uintptr_t slot ) [inline], [static]
```

Extract the index from a slot.

**6.54.3.8 starpu\_rbtrees\_insert\_rebalance()**

```
void starpu_rbtrees_insert_rebalance (
    struct starpu_rbtrees * tree,
    struct starpu_rbtrees_node * parent,
    int index,
    struct starpu_rbtrees_node * node )
```

Insert a node in a tree, rebalancing it if necessary.

The index parameter is the index in the children array of the parent where the new node is to be inserted. It is ignored if the parent is null.

This function is intended to be used by the [starpu\\_rbtrees\\_insert\(\)](#) macro only.

**6.54.3.9 starpu\_rbtrees\_nearest()**

```
struct starpu_rbtrees_node * starpu_rbtrees_nearest (
    struct starpu_rbtrees_node * parent,
```

```
int index,
int direction )
```

Return the previous or next node relative to a location in a tree.

The parent and index parameters define the location, which can be empty. The direction parameter is either STARPU\_RBTREE\_LEFT (to obtain the previous node) or STARPU\_RBTREE\_RIGHT (to obtain the next one).

#### 6.54.3.10 starpu\_rbtree\_firstlast()

```
struct starpu_rbtree_node * starpu_rbtree_firstlast (
    const struct starpu_rbtree * tree,
    int direction )
```

Return the first or last node of a tree.

The direction parameter is either STARPU\_RBTREE\_LEFT (to obtain the first node) or STARPU\_RBTREE\_RIGHT (to obtain the last one).

#### 6.54.3.11 starpu\_rbtree\_walk()

```
struct starpu_rbtree_node * starpu_rbtree_walk (
    struct starpu_rbtree_node * node,
    int direction )
```

Return the node next to, or previous to the given node.

The direction parameter is either STARPU\_RBTREE\_LEFT (to obtain the previous node) or STARPU\_RBTREE\_RIGHT (to obtain the next one).

#### 6.54.3.12 starpu\_rbtree\_postwalk\_deepest()

```
struct starpu_rbtree_node * starpu_rbtree_postwalk_deepest (
    const struct starpu_rbtree * tree )
```

Return the left-most deepest node of a tree, which is the starting point of the postorder traversal performed by [starpu\\_rbtree\\_for\\_each\\_remove\(\)](#).

#### 6.54.3.13 starpu\_rbtree\_postwalk\_unlink()

```
struct starpu_rbtree_node * starpu_rbtree_postwalk_unlink (
    struct starpu_rbtree_node * node )
```

Unlink a node from its tree and return the next (right) node in postorder.

## 6.55 regression.h File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <core/perfmodel/perfmodel.h>
#include <starpu.h>
```

### Functions

- `int starpu_regression_non_linear_power` ([struct](#) starpu\_perfmodel\_history\_list \*ptr, double \*a, double \*b, double \*c)

## 6.56 rwalk.h File Reference

```
#include <stdint.h>
#include <starpu.h>
```

## Data Structures

- [struct \\_starpw\\_lock](#)

## Functions

- [void \\_starpw\\_init\\_rw\\_lock](#) ([struct \\_starpw\\_lock](#) \*lock)
- [void \\_starpw\\_take\\_rw\\_lock\\_write](#) ([struct \\_starpw\\_lock](#) \*lock)
- [void \\_starpw\\_take\\_rw\\_lock\\_read](#) ([struct \\_starpw\\_lock](#) \*lock)
- [int \\_starpw\\_take\\_rw\\_lock\\_write\\_try](#) ([struct \\_starpw\\_lock](#) \*lock)
- [int \\_starpw\\_take\\_rw\\_lock\\_read\\_try](#) ([struct \\_starpw\\_lock](#) \*lock)
- [void \\_starpw\\_release\\_rw\\_lock](#) ([struct \\_starpw\\_lock](#) \*lock)

### 6.56.1 Data Structure Documentation

#### 6.56.1.1 struct \_starpw\_lock

Dummy implementation of a RW-lock using a spinlock.

##### Data Fields

uint32_t	busy	
uint8_t	writer	
uint16_t	readercnt	

### 6.56.2 Function Documentation

#### 6.56.2.1 \_starpw\_init\_rw\_lock()

```
void _starpw_init_rw_lock (
    struct _starpw_lock * lock )
```

Initialize the RW-lock

#### 6.56.2.2 \_starpw\_take\_rw\_lock\_write()

```
void _starpw_take_rw_lock_write (
    struct _starpw_lock * lock )
```

Grab the RW-lock in a write mode

#### 6.56.2.3 \_starpw\_take\_rw\_lock\_read()

```
void _starpw_take_rw_lock_read (
    struct _starpw_lock * lock )
```

Grab the RW-lock in a read mode

#### 6.56.2.4 \_starpw\_take\_rw\_lock\_write\_try()

```
int _starpw_take_rw_lock_write_try (
    struct _starpw_lock * lock )
```

Try to grab the RW-lock in a write mode. Returns 0 in case of success, -1 otherwise.

#### 6.56.2.5 \_starpw\_take\_rw\_lock\_read\_try()

```
int _starpw_take_rw_lock_read_try (
    struct _starpw_lock * lock )
```

Try to grab the RW-lock in a read mode. Returns 0 in case of success, -1 otherwise.

### 6.56.2.6 \_starpu\_release\_rw\_lock()

```
void _starpu_release_rw_lock (
    struct _starpu_rw_lock * lock )
```

Unlock the RW-lock.

## 6.57 sched\_component.h File Reference

```
#include <starpu_sched_component.h>
```

### Functions

- void [\\_starpu\\_sched\\_component\\_lock\\_all\\_workers](#) (void)
- void [\\_starpu\\_sched\\_component\\_unlock\\_all\\_workers](#) (void)
- void [\\_starpu\\_sched\\_component\\_workers\\_destroy](#) (void)
- [struct \\_starpu\\_worker](#) \* [\\_starpu\\_sched\\_component\\_worker\\_get\\_worker](#) ([struct](#) [starpu\\_sched\\_↵](#)↵ component \*)
- [struct](#) [starpu\\_bitmap](#) \* [\\_starpu\\_get\\_worker\\_mask](#) (unsigned [sched\\_ctx\\_id](#))

### 6.57.1 Function Documentation

#### 6.57.1.1 \_starpu\_sched\_component\_lock\_all\_workers()

```
void _starpu_sched_component_lock_all_workers (
    void )
```

lock and unlock drivers for modifying schedulers

## 6.58 sched\_ctx.h File Reference

```
#include <starpu.h>
#include <starpu_sched_ctx.h>
#include <starpu_sched_ctx_hypervisor.h>
#include <starpu_scheduler.h>
#include <common/config.h>
#include <common/barrier_counter.h>
#include <common/utils.h>
#include <profiling/profiling.h>
#include <semaphore.h>
#include <core/task.h>
#include "sched_ctx_list.h"
#include <hwloc.h>
```

### Data Structures

- [struct \\_starpu\\_sched\\_ctx](#)
- [struct \\_starpu\\_ctx\\_change](#)

### Macros

- [#define NO\\_RESIZE](#)
- [#define REQ\\_RESIZE](#)
- [#define DO\\_RESIZE](#)
- [#define STARPU\\_GLOBAL\\_SCHED\\_CTX](#)

- #define **STARPU\_NMAXSMS**
- #define **\_starpu\_sched\_ctx\_get\_sched\_ctx\_for\_worker\_and\_job(w, j)**
- #define **STARPU\_SCHED\_CTX\_CHECK\_LOCK(sched\_ctx\_id)**

## Functions

- void **\_starpu\_init\_all\_sched\_ctxs** (struct **\_starpu\_machine\_config** \*config)
- struct **\_starpu\_sched\_ctx** \* **\_starpu\_create\_sched\_ctx** (struct **starpu\_sched\_policy** \*policy, int \*workerid, int nworkerids, unsigned is\_init\_sched, const char \*sched\_name, int min\_prio\_set, int min\_prio, int max\_prio←\_set, int max\_prio, unsigned awake\_workers, void(\*sched\_policy\_callback)(unsigned), void \*user\_data, int nsub\_ctxs, int \*sub\_ctxs, int nsms)
- void **\_starpu\_delete\_all\_sched\_ctxs** ()
- int **\_starpu\_wait\_for\_all\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id)
- int **\_starpu\_wait\_for\_n\_submitted\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id, unsigned n)
- void **\_starpu\_decrement\_nsubmitted\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id)
- void **\_starpu\_increment\_nsubmitted\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id)
- int **\_starpu\_get\_nsubmitted\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id)
- int **\_starpu\_check\_nsubmitted\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id)
- void **\_starpu\_decrement\_nready\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id, double ready\_flops)
- unsigned **\_starpu\_increment\_nready\_tasks\_of\_sched\_ctx** (unsigned sched\_ctx\_id, double ready\_flops, struct **starpu\_task** \*task)
- int **\_starpu\_wait\_for\_no\_ready\_of\_sched\_ctx** (unsigned sched\_ctx\_id)
- int **\_starpu\_get\_workers\_of\_sched\_ctx** (unsigned sched\_ctx\_id, int \*pus, enum **starpu\_worker\_archtype** arch)
- void **\_starpu\_worker\_gets\_out\_of\_ctx** (unsigned sched\_ctx\_id, struct **\_starpu\_worker** \*worker)
- unsigned **\_starpu\_worker\_belongs\_to\_a\_sched\_ctx** (int workerid, unsigned sched\_ctx\_id)
- unsigned **\_starpu\_sched\_ctx\_last\_worker\_awake** (struct **\_starpu\_worker** \*worker)
- unsigned **\_starpu\_sched\_ctx\_get\_current\_context** () STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- int **\_starpu\_workers\_able\_to\_execute\_task** (struct **starpu\_task** \*task, struct **\_starpu\_sched\_ctx** \*sched\_ctx)
- unsigned **\_starpu\_sched\_ctx\_allow\_hypervisor** (unsigned sched\_ctx\_id)
- struct **starpu\_perfmodel\_arch** \* **\_starpu\_sched\_ctx\_get\_perf\_archtype** (unsigned sched\_ctx)
- void **\_starpu\_sched\_ctx\_post\_exec\_task\_cb** (int workerid, struct **starpu\_task** \*task, size\_t data\_size, uint32\_t footprint)
- void **starpu\_sched\_ctx\_add\_combined\_workers** (int \*combined\_workers\_to\_add, unsigned n←combined\_workers\_to\_add, unsigned sched\_ctx\_id)
- struct **\_starpu\_sched\_ctx** \* **\_\_starpu\_sched\_ctx\_get\_sched\_ctx\_for\_worker\_and\_job** (struct **\_starpu\_worker** \*worker, struct **\_starpu\_job** \*j)
- static struct **\_starpu\_sched\_ctx** \* **\_starpu\_get\_sched\_ctx\_struct** (unsigned id)
- static int **\_starpu\_sched\_ctx\_check\_write\_locked** (unsigned sched\_ctx\_id)
- static void **\_starpu\_sched\_ctx\_lock\_write** (unsigned sched\_ctx\_id)
- static void **\_starpu\_sched\_ctx\_unlock\_write** (unsigned sched\_ctx\_id)
- static void **\_starpu\_sched\_ctx\_lock\_read** (unsigned sched\_ctx\_id)
- static void **\_starpu\_sched\_ctx\_unlock\_read** (unsigned sched\_ctx\_id)
- static unsigned **\_starpu\_sched\_ctx\_worker\_is\_master\_for\_child\_ctx** (unsigned sched\_ctx\_id, unsigned workerid, struct **starpu\_task** \*task)
- void **\_starpu\_worker\_apply\_deferred\_ctx\_changes** (void)

## 6.58.1 Data Structure Documentation

### 6.58.1.1 struct **\_starpu\_ctx\_change**

per-worker list of deferred ctx\_change ops

#### Data Fields

int	sched_ctx_id	
-----	--------------	--

## Data Fields

int	op	
int	nworkers_to_notify	
int *	workerids_to_notify	
int	nworkers_to_change	
int *	workerids_to_change	

## 6.58.2 Function Documentation

### 6.58.2.1 `_starpu_init_all_sched_ctxs()`

```
void _starpu_init_all_sched_ctxs (
    struct _starpu_machine_config * config )
init sched_ctx_id of all contextes
```

### 6.58.2.2 `_starpu_create_sched_ctx()`

```
struct _starpu_sched_ctx * _starpu_create_sched_ctx (
    struct starpu_sched_policy * policy,
    int * workerid,
    int nworkerids,
    unsigned is_init_sched,
    const char * sched_name,
    int min_prio_set,
    int min_prio,
    int max_prio_set,
    int max_prio,
    unsigned awake_workers,
    void(*) (unsigned) sched_policy_callback,
    void * user_data,
    int nsub_ctxs,
    int * sub_ctxs,
    int nsms )
```

allocate all structures belonging to a context

### 6.58.2.3 `_starpu_delete_all_sched_ctxs()`

```
void _starpu_delete_all_sched_ctxs ( )
delete all sched_ctx
```

### 6.58.2.4 `_starpu_wait_for_all_tasks_of_sched_ctx()`

```
int _starpu_wait_for_all_tasks_of_sched_ctx (
    unsigned sched_ctx_id )
```

This function waits until all the tasks that were already submitted to a specific context have been executed.

### 6.58.2.5 `_starpu_wait_for_n_submitted_tasks_of_sched_ctx()`

```
int _starpu_wait_for_n_submitted_tasks_of_sched_ctx (
    unsigned sched_ctx_id,
    unsigned n )
```

This function waits until at most n tasks are still submitted.

**6.58.2.6 `_starpu_decrement_nsubmitted_tasks_of_sched_ctx()`**

```
void _starpu_decrement_nsubmitted_tasks_of_sched_ctx (
    unsigned sched_ctx_id )
```

In order to implement `starpu_wait_for_all_tasks_of_ctx`, we keep track of the number of task currently submitted to the context

**6.58.2.7 `_starpu_get_workers_of_sched_ctx()`**

```
int _starpu_get_workers_of_sched_ctx (
    unsigned sched_ctx_id,
    int * pus,
    enum starpu_worker_archtype arch )
```

Get workers belonging to a certain context, it returns the number of workers take care: no mutex taken, the list of workers might not be updated

**6.58.2.8 `_starpu_worker_gets_out_of_ctx()`**

```
void _starpu_worker_gets_out_of_ctx (
    unsigned sched_ctx_id,
    struct _starpu_worker * worker )
```

Let the worker know it does not belong to the context and that it should stop popping from it

**6.58.2.9 `_starpu_worker_belongs_to_a_sched_ctx()`**

```
unsigned _starpu_worker_belongs_to_a_sched_ctx (
    int workerid,
    unsigned sched_ctx_id )
```

Check if the worker belongs to another `sched_ctx`

**6.58.2.10 `_starpu_sched_ctx_last_worker_awake()`**

```
unsigned _starpu_sched_ctx_last_worker_awake (
    struct _starpu_worker * worker )
```

indicates whether this worker should go to sleep or not (if it is the last one awake in a context he should better keep awake)

**6.58.2.11 `_starpu_sched_ctx_get_current_context()`**

```
unsigned _starpu_sched_ctx_get_current_context ( )
```

If `starpu_sched_ctx_set_context()` has been called, returns the context id set by its last call, or the id of the initial context

**6.58.2.12 `_starpu_workers_able_to_execute_task()`**

```
int _starpu_workers_able_to_execute_task (
    struct starpu_task * task,
    struct _starpu_sched_ctx * sched_ctx )
```

verify that some worker can execute a certain task

**6.58.2.13 `_starpu_sched_ctx_post_exec_task_cb()`**

```
void _starpu_sched_ctx_post_exec_task_cb (
    int workerid,
    struct starpu_task * task,
    size_t data_size,
    uint32_t footprint )
```

Notifies the hypervisor that a tasks was popped from the workers' list

**6.58.2.14** `__starpu_sched_ctx_get_sched_ctx_for_worker_and_job()`

```
struct __starpu_sched_ctx * __starpu_sched_ctx_get_sched_ctx_for_worker_and_job (
    struct __starpu_worker * worker,
    struct __starpu_job * j )
```

if the worker is the master of a parallel context, and the job is meant to be executed on this parallel context, return a pointer to the context

**6.58.2.15** `_starpu_worker_apply_deferred_ctx_changes()`

```
void _starpu_worker_apply_deferred_ctx_changes (
    void )
```

Go through the list of deferred ctx changes of the current worker and apply any ctx change operation found until the list is empty

**6.59 sched\_ctx\_list.h File Reference****Data Structures**

- [struct \\_\\_starpu\\_sched\\_ctx\\_list](#)
- [struct \\_\\_starpu\\_sched\\_ctx\\_elt](#)
- [struct \\_\\_starpu\\_sched\\_ctx\\_list\\_iterator](#)

**Functions**

- [struct \\_\\_starpu\\_sched\\_ctx\\_elt \\* \\_\\_starpu\\_sched\\_ctx\\_elt\\_find](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [void \\_\\_starpu\\_sched\\_ctx\\_elt\\_ensure\\_consistency](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [void \\_\\_starpu\\_sched\\_ctx\\_elt\\_init](#) (struct \_\_starpu\_sched\_ctx\_elt \*elt, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [struct \\_\\_starpu\\_sched\\_ctx\\_elt \\* \\_\\_starpu\\_sched\\_ctx\\_elt\\_add\\_after](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [struct \\_\\_starpu\\_sched\\_ctx\\_elt \\* \\_\\_starpu\\_sched\\_ctx\\_elt\\_add\\_before](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [struct \\_\\_starpu\\_sched\\_ctx\\_elt \\* \\_\\_starpu\\_sched\\_ctx\\_elt\\_add](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [void \\_\\_starpu\\_sched\\_ctx\\_elt\\_remove](#) (struct \_\_starpu\_sched\_ctx\_list \*list, struct \_\_starpu\_sched\_ctx\_elt \*elt) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [int \\_\\_starpu\\_sched\\_ctx\\_elt\\_exists](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [int \\_\\_starpu\\_sched\\_ctx\\_elt\\_get\\_priority](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [struct \\_\\_starpu\\_sched\\_ctx\\_list \\* \\_\\_starpu\\_sched\\_ctx\\_list\\_find](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned prio) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [struct \\_\\_starpu\\_sched\\_ctx\\_elt \\* \\_\\_starpu\\_sched\\_ctx\\_list\\_add\\_prio](#) (struct \_\_starpu\_sched\_ctx\_list \*\*list, unsigned prio, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [int \\_\\_starpu\\_sched\\_ctx\\_list\\_add](#) (struct \_\_starpu\_sched\_ctx\_list \*\*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [void \\_\\_starpu\\_sched\\_ctx\\_list\\_remove\\_elt](#) (struct \_\_starpu\_sched\_ctx\_list \*\*list, struct \_\_starpu\_sched\_ctx\_elt \*rm) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [int \\_\\_starpu\\_sched\\_ctx\\_list\\_remove](#) (struct \_\_starpu\_sched\_ctx\_list \*\*list, unsigned sched\_ctx) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [int \\_\\_starpu\\_sched\\_ctx\\_list\\_move](#) (struct \_\_starpu\_sched\_ctx\_list \*\*list, unsigned sched\_ctx, unsigned prio\_to) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- [int \\_\\_starpu\\_sched\\_ctx\\_list\\_exists](#) (struct \_\_starpu\_sched\_ctx\_list \*list, unsigned prio) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

- void `_starpuschedctxlist_remove_all` (struct `_starpuschedctxlist` \*list) STARPU\_ATTRIBUTE\_↔ VISIBILITY\_DEFAULT
- void `_starpuschedctxlist_delete` (struct `_starpuschedctxlist` \*\*list) STARPU\_ATTRIBUTE\_↔ VISIBILITY\_DEFAULT
- int `_starpuschedctxlist_push_event` (struct `_starpuschedctxlist` \*list, unsigned sched\_ctx)
- int `_starpuschedctxlist_pop_event` (struct `_starpuschedctxlist` \*list, unsigned sched\_ctx)
- int `_starpuschedctxlist_pop_all_event` (struct `_starpuschedctxlist` \*list, unsigned sched\_ctx)
- int `_starpuschedctxlist_iterator_init` (struct `_starpuschedctxlist` \*list, struct `_starpuschedctxlist_iterator` \*it) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- int `_starpuschedctxlist_iterator_has_next` (struct `_starpuschedctxlist_iterator` \*it) STARPU\_↔ ATTRIBUTE\_VISIBILITY\_DEFAULT
- struct `_starpuschedctxelt` \* `_starpuschedctxlist_iterator_get_next` (struct `_starpuschedctxlist_iterator` \*it) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

## 6.59.1 Data Structure Documentation

### 6.59.1.1 struct `_starpuschedctxlist`

#### Data Fields

<code>struct _starpuschedctxlist *</code>	prev	
<code>struct _starpuschedctxlist *</code>	next	
<code>struct _starpuschedctxelt *</code>	head	
unsigned	priority	

### 6.59.1.2 struct `_starpuschedctxelt`

Represents a circular list of sched context.

#### Data Fields

<code>struct _starpuschedctxelt *</code>	prev	
<code>struct _starpuschedctxelt *</code>	next	
<code>struct _starpuschedctxlist *</code>	parent	
unsigned	sched_ctx	
long	task_number	
unsigned	last_poped	

### 6.59.1.3 struct `_starpuschedctxlist_iterator`

#### Data Fields

<code>struct _starpuschedctxlist *</code>	list_head	
<code>struct _starpuschedctxelt *</code>	cursor	

## 6.59.2 Function Documentation

### 6.59.2.1 `_starpuschedctxelt_find()`

```
struct _starpuschedctxelt * _starpuschedctxelt_find (
    struct _starpuschedctxlist * list,
```

```
unsigned sched_ctx )
```

Element (sched\_ctx) level operations

### 6.59.2.2 \_starpu\_sched\_ctx\_list\_find()

```
struct _starpu_sched_ctx_list * _starpu_sched_ctx_list_find (
    struct _starpu_sched_ctx_list * list,
    unsigned prio )
```

List (priority) level operations

### 6.59.2.3 \_starpu\_sched\_ctx\_list\_push\_event()

```
int _starpu_sched_ctx_list_push_event (
    struct _starpu_sched_ctx_list * list,
    unsigned sched_ctx )
```

Task number management

### 6.59.2.4 \_starpu\_sched\_ctx\_list\_iterator\_init()

```
int _starpu_sched_ctx_list_iterator_init (
    struct _starpu_sched_ctx_list * list,
    struct _starpu_sched_ctx_list_iterator * it )
```

Iterator operations

## 6.60 sched\_policy.h File Reference

```
#include <starpu.h>
#include <signal.h>
#include <core/workers.h>
#include <core/sched_ctx.h>
#include <starpu_scheduler.h>
#include <core/simgrid.h>
```

### Macros

- #define **\_STARPU\_SCHED\_BEGIN**
- #define **\_STARPU\_SCHED\_END**
- #define **\_STARPU\_TASK\_BREAK\_ON**(task, what)

### Functions

- void **\_starpu\_sched\_init** (void)
- [struct](#) starpu\_sched\_policy \* **\_starpu\_get\_sched\_policy** ([struct](#) \_starpu\_sched\_ctx \*sched\_ctx)
- void **\_starpu\_init\_sched\_policy** ([struct](#) \_starpu\_machine\_config \*config, [struct](#) \_starpu\_sched\_ctx \*sched\_ctx, [struct](#) starpu\_sched\_policy \*policy)
- void **\_starpu\_deinit\_sched\_policy** ([struct](#) \_starpu\_sched\_ctx \*sched\_ctx)
- [struct](#) starpu\_sched\_policy \* **\_starpu\_select\_sched\_policy** ([struct](#) \_starpu\_machine\_config \*config, const char \*required\_policy)
- void **\_starpu\_sched\_task\_submit** ([struct](#) starpu\_task \*task)
- void **\_starpu\_sched\_do\_schedule** (unsigned sched\_ctx\_id)
- int **\_starpu\_push\_task** ([struct](#) \_starpu\_job \*task)
- int **\_starpu\_repush\_task** ([struct](#) \_starpu\_job \*task)
- int **\_starpu\_push\_task\_to\_workers** ([struct](#) starpu\_task \*task)
- [struct](#) starpu\_task \* **\_starpu\_pop\_task** ([struct](#) \_starpu\_worker \*worker)
- void **\_starpu\_sched\_post\_exec\_hook** ([struct](#) starpu\_task \*task)
- int **\_starpu\_pop\_task\_end** ([struct](#) starpu\_task \*task)

- `struct starpu_task * _starpu_create_conversion_task` (starpu\_data\_handle\_t handle, unsigned int node) STARPU\_ATTRIBUTE\_MALLOC
- `struct starpu_task * _starpu_create_conversion_task_for_arch` (starpu\_data\_handle\_t handle, enum starpu\_node\_kind node\_kind) STARPU\_ATTRIBUTE\_MALLOC
- `void _starpu_sched_pre_exec_hook` (`struct starpu_task *task`)
- `void _starpu_print_idle_time` ()

## Variables

- `struct starpu_sched_policy _starpu_sched_lws_policy`
- `struct starpu_sched_policy _starpu_sched_ws_policy`
- `struct starpu_sched_policy _starpu_sched_prio_policy`
- `struct starpu_sched_policy _starpu_sched_random_policy`
- `struct starpu_sched_policy _starpu_sched_dm_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_policy` STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- `struct starpu_sched_policy _starpu_sched_dmda_prio_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_ready_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_sorted_policy`
- `struct starpu_sched_policy _starpu_sched_dmda_sorted_decision_policy`
- `struct starpu_sched_policy _starpu_sched_eager_policy`
- `struct starpu_sched_policy _starpu_sched_peager_policy`
- `struct starpu_sched_policy _starpu_sched_heteroprio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_eager_policy`
- `struct starpu_sched_policy _starpu_sched_modular_eager_prefetching_policy`
- `struct starpu_sched_policy _starpu_sched_modular_eager_prio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_gemm_policy`
- `struct starpu_sched_policy _starpu_sched_modular_prio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_prio_prefetching_policy`
- `struct starpu_sched_policy _starpu_sched_modular_random_policy`
- `struct starpu_sched_policy _starpu_sched_modular_random_prio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_random_prefetching_policy`
- `struct starpu_sched_policy _starpu_sched_modular_random_prio_prefetching_policy`
- `struct starpu_sched_policy _starpu_sched_modular_parallel_random_policy`
- `struct starpu_sched_policy _starpu_sched_modular_parallel_random_prio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_ws_policy`
- `struct starpu_sched_policy _starpu_sched_modular_dmda_policy`
- `struct starpu_sched_policy _starpu_sched_modular_dmdap_policy`
- `struct starpu_sched_policy _starpu_sched_modular_dmdar_policy`
- `struct starpu_sched_policy _starpu_sched_modular_dmdas_policy`
- `struct starpu_sched_policy _starpu_sched_modular_heft_policy`
- `struct starpu_sched_policy _starpu_sched_modular_heft_prio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_heft2_policy`
- `struct starpu_sched_policy _starpu_sched_modular_heteroprio_policy`
- `struct starpu_sched_policy _starpu_sched_modular_heteroprio_heft_policy`
- `struct starpu_sched_policy _starpu_sched_modular_parallel_heft_policy`
- `struct starpu_sched_policy _starpu_sched_graph_test_policy`
- `struct starpu_sched_policy _starpu_sched_tree_heft_hierarchical_policy`
- `long _starpu_task_break_on_push`
- `long _starpu_task_break_on_sched`
- `long _starpu_task_break_on_pop`
- `long _starpu_task_break_on_exec`

### 6.60.1 Function Documentation

### 6.60.1.1 `_starpu_push_task_to_workers()`

```
int _starpu_push_task_to_workers (
    struct starpu_task * task )
```

actually pushes the tasks to the specific worker or to the scheduler

### 6.60.1.2 `_starpu_pop_task()`

```
struct starpu_task * _starpu_pop_task (
    struct _starpu_worker * worker )
```

pop a task that can be executed on the worker

## 6.61 simgrid.h File Reference

```
#include <xbt/xbt_os_time.h>
```

### Data Structures

- struct [\\_starpu\\_pthread\\_args](#)

### Macros

- `#define _STARPU_SIMGRID_MAIN_RETURN`
- `#define MAX_TSD`
- `#define STARPU_MPI_AS_PREFIX`
- `#define _starpu_simgrid_running_smpi()`
- `#define _starpu_simgrid_cuda_malloc_cost()`
- `#define _starpu_simgrid_cuda_queue_cost()`
- `#define _starpu_simgrid_task_submit_cost()`
- `#define _starpu_simgrid_task_push_cost()`
- `#define _starpu_simgrid_fetching_input_cost()`
- `#define _starpu_simgrid_sched_cost()`
- `#define _SIMGRID_TIMER_BEGIN(cond)`
- `#define _SIMGRID_TIMER_END`
- `#define _starpu_simgrid_data_new(size)`
- `#define _starpu_simgrid_data_increase(size)`
- `#define _starpu_simgrid_data_alloc(size)`
- `#define _starpu_simgrid_data_free(size)`
- `#define _starpu_simgrid_data_transfer(size, src_node, dst_node)`

### Typedefs

- typedef int `_starpu_simgrid_main_ret`
- typedef SD\_link\_t `starpu_sg_link_t`

### Functions

- `_starpu_simgrid_main_ret starpu_simgrid_thread_start` (int argc, char \*argv[])
- void `_starpu_start_simgrid` (int \*argc, char \*\*argv)
- void `_starpu_simgrid_init_early` (int \*argc, char \*\*\*argv)
- void `_starpu_simgrid_init` (void)
- void `_starpu_simgrid_cpp_init` (void)
- void `_starpu_simgrid_deinit` (void)
- void `_starpu_simgrid_deinit_late` (void)
- void `_starpu_simgrid_actor_setup` (void)

- void **\_starpu\_simgrid\_wait\_tasks** (int workerid)
- void **\_starpu\_simgrid\_submit\_job** (int workerid, int sched\_ctx\_id, [struct \\_starpu\\_job](#) \*job, [struct starpu\\_↔](#) perfmodel\_arch \*perf\_arch, double length, double energy, unsigned \*finished)
- int **\_starpu\_simgrid\_transfer** (size\_t size, unsigned src\_node, unsigned dst\_node, [struct \\_starpu\\_data\\_↔](#) request \*req)
- int **\_starpu\_simgrid\_wait\_transfer\_event** (void \*event)
- int **\_starpu\_simgrid\_test\_transfer\_event** (void \*event)
- void **\_starpu\_simgrid\_sync\_gpus** (void)
- int **\_starpu\_simgrid\_get\_nbhosts** (const char \*prefix)
- unsigned long long **\_starpu\_simgrid\_get\_memsize** (const char \*prefix, unsigned devid)
- const char \* **\_starpu\_simgrid\_get\_devname** (const char \*prefix, unsigned devid)
- starpu\_sg\_host\_t **\_starpu\_simgrid\_get\_host\_by\_name** (const char \*name)
- starpu\_sg\_host\_t **\_starpu\_simgrid\_get\_memnode\_host** (unsigned node)
- starpu\_sg\_host\_t **\_starpu\_simgrid\_get\_host\_by\_worker** ([struct \\_starpu\\_worker](#) \*worker)
- void **\_starpu\_simgrid\_get\_platform\_path** (int version, char \*path, size\_t maxlen)
- msg\_as\_t **\_starpu\_simgrid\_get\_as\_by\_name** (const char \*name)
- int **starpu\_mpi\_world\_rank** (void)
- int **\_starpu\_mpi\_simgrid\_init** (int argc, char \*argv[])
- void **\_starpu\_simgrid\_count\_ngpus** (void)
- void **\_starpu\_simgrid\_set\_stack\_size** (size\_t stack\_size)
- void **\_starpu\_simgrid\_xbt\_thread\_create** (const char \*name, starpu\_pthread\_attr\_t \*attr, void\_f\_pvoid\_t code, void \*param)

## Variables

- starpu\_pthread\_queue\_t **\_starpu\_simgrid\_transfer\_queue** [[STARPU\\_MAXNODES](#)]
- starpu\_pthread\_queue\_t **\_starpu\_simgrid\_task\_queue** [[STARPU\\_NMAXWORKERS](#)]
- size\_t **\_starpu\_default\_stack\_size**

## 6.61.1 Macro Definition Documentation

### 6.61.1.1 **\_starpu\_simgrid\_data\_new**

```
#define _starpu_simgrid_data_new(  
    size )
```

Experimental functions for OOC stochastic analysis

## 6.61.2 Function Documentation

### 6.61.2.1 **\_starpu\_simgrid\_get\_nbhosts()**

```
int _starpu_simgrid_get_nbhosts (  
    const char * prefix )
```

Return the number of hosts prefixed by PREFIX

### 6.61.2.2 **\_starpu\_simgrid\_count\_ngpus()**

```
void _starpu_simgrid_count_ngpus (  
    void )
```

Called at initialization to count how many GPUs are interfering with each bus

## 6.62 sink\_common.h File Reference

```
#include <common/config.h>
```

## 6.63 sort\_data\_handles.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <stdlib.h>
#include <stdarg.h>
#include <core/jobs.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
```

### Functions

- void [\\_starpu\\_sort\\_task\\_handles](#) (struct [\\_starpu\\_data\\_descr](#) descr[], unsigned nbuffers)
- int [\\_starpu\\_handles\\_same\\_root](#) (starpu\_data\_handle\_t dataA, starpu\_data\_handle\_t dataB)

### 6.63.1 Function Documentation

#### 6.63.1.1 \_starpu\_sort\_task\_handles()

```
void _starpu_sort_task_handles (
    struct _starpu_data_descr descr[],
    unsigned nbuffers )
```

To avoid deadlocks, we reorder the different buffers accessed to by the task so that we always grab the rw-lock associated to the handles in the same order.

#### 6.63.1.2 \_starpu\_handles\_same\_root()

```
int _starpu_handles_same_root (
    starpu_data_handle_t dataA,
    starpu_data_handle_t dataB )
```

The reordering however puts alongside some different handles, just because they have the same root. When avoiding to lock/acquire/load the same handle several times, we need to keep looking among those.

## 6.64 source\_common.h File Reference

## 6.65 starpu\_parallel\_worker\_create.h File Reference

```
#include <starpu.h>
#include <core/workers.h>
#include <common/list.h>
#include <string.h>
#include <omp.h>
#include <mkl_service.h>
```

### Data Structures

- struct [starpu\\_parallel\\_worker\\_config](#)
- struct [\\_starpu\\_parallel\\_worker\\_parameters](#)

## 6.65.1 Data Structure Documentation

### 6.65.1.1 struct starpu\_parallel\_worker\_config

#### Data Fields

unsigned	id	
hwloc_topology_t	topology	
unsigned	nparallel_workers	
unsigned	ngroups	
<a href="#">struct _starpu_parallel_worker_group_list *</a>	groups	
<a href="#">struct _starpu_parallel_worker_parameters *</a>	orig_params	
<a href="#">struct _starpu_parallel_worker_parameters *</a>	params	

## 6.66 starpu\_data\_cpy.h File Reference

```
#include <starpu.h>
```

### Functions

- int [\\_starpu\\_data\\_cpy](#) (starpu\_data\_handle\_t dst\_handle, starpu\_data\_handle\_t src\_handle, int asynchronous, void(\*callback\_func)(void \*), void \*callback\_arg, int reduction, [struct starpu\\_task \\*](#)reduction\_↔ dep\_task, int priority)

## 6.67 starpu\_debug\_helpers.h File Reference

```
#include <starpu.h>
#include <starpu_config.h>
#include <starpu_util.h>
```

### Functions

- void [\\_starpu\\_benchmark\\_ping\\_pong](#) (starpu\_data\_handle\_t handle, unsigned node0, unsigned node1, unsigned niter) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- void [\\_starpu\\_debug\\_display\\_structures\\_size](#) (FILE \*stream) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

### 6.67.1 Function Documentation

#### 6.67.1.1 \_starpu\_benchmark\_ping\_pong()

```
void _starpu_benchmark_ping_pong (
    starpu_data_handle_t handle,
    unsigned node0,
    unsigned node1,
    unsigned niter )
```

Perform a ping pong between the two memory nodes

#### 6.67.1.2 \_starpu\_debug\_display\_structures\_size()

```
void _starpu_debug_display_structures_size (
    FILE * stream )
```

Display the size of different data structures

## 6.68 starpu\_fxt.h File Reference

```
#include <starpu.h>
#include <starpu_config.h>
#include <common/config.h>
```

## 6.69 starpu\_spinlock.h File Reference

```
#include <errno.h>
#include <stdint.h>
#include <common/config.h>
#include <common/fxt.h>
#include <common/thread.h>
#include <starpu.h>
```

### Data Structures

- struct [\\_starpu\\_spinlock](#)

### Macros

- #define [\\_starpu\\_spin\\_destroy](#)(\_lock)
- #define [\\_starpu\\_spin\\_checklocked](#)(\_lock)
- #define [\\_starpu\\_spin\\_lock](#)(lock)
- #define [\\_starpu\\_spin\\_trylock](#)(lock)
- #define [\\_starpu\\_spin\\_unlock](#)(lock)
- #define [STARPU\\_SPIN\\_MAXTRY](#)

### Functions

- static int [\\_starpu\\_spin\\_init](#) (struct [\\_starpu\\_spinlock](#) \*lock)
- static int [\\_\\_starpu\\_spin\\_lock](#) (struct [\\_starpu\\_spinlock](#) \*lock, const char \*file STARPU\_ATTRIBUTE\_UNUSED, int line STARPU\_ATTRIBUTE\_UNUSED, const char \*func STARPU\_ATTRIBUTE\_UNUSED)
- static int [\\_\\_starpu\\_spin\\_trylock](#) (struct [\\_starpu\\_spinlock](#) \*lock, const char \*file STARPU\_ATTRIBUTE\_UNUSED, int line STARPU\_ATTRIBUTE\_UNUSED, const char \*func STARPU\_ATTRIBUTE\_UNUSED)
- static int [\\_\\_starpu\\_spin\\_unlock](#) (struct [\\_starpu\\_spinlock](#) \*lock, const char \*file STARPU\_ATTRIBUTE\_UNUSED, int line STARPU\_ATTRIBUTE\_UNUSED, const char \*func STARPU\_ATTRIBUTE\_UNUSED)

### 6.69.1 Data Structure Documentation

#### 6.69.1.1 struct [\\_starpu\\_spinlock](#)

##### Data Fields

<a href="#">starpu_pthread_spinlock_t</a>	<a href="#">lock</a>	
---	----------------------	--

## 6.70 starpu\_task\_insert\_utils.h File Reference

```
#include <stdlib.h>
#include <stdarg.h>
#include <starpu.h>
```

## Typedefs

- typedef void(\* [\\_starpu\\_callback\\_func\\_t](#)) (void \*)

## Functions

- int [\\_starpu\\_task\\_insert\\_create](#) ([struct starpu\\_codelet](#) \*cl, [struct starpu\\_task](#) \*task, va\_list varg\_list) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- int [\\_fstarpu\\_task\\_insert\\_create](#) ([struct starpu\\_codelet](#) \*cl, [struct starpu\\_task](#) \*task, void \*\*arglist) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

## 6.71 tags.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <core/dependencies/cg.h>
```

## Data Structures

- [struct \\_starpu\\_tag](#)

## Macros

- `#define \_STARPU\_TAG\_SIZE`

## Enumerations

- enum [\\_starpu\\_tag\\_state](#) {  
[STARPU\\_INVALID\\_STATE](#) , [STARPU\\_ASSOCIATED](#) , [STARPU\\_BLOCKED](#) , [STARPU\\_READY](#) ,  
[STARPU\\_DONE](#) }

## Functions

- void [\\_starpu\\_init\\_tags](#) (void)
- void [\\_starpu\\_notify\\_tag\\_dependencies](#) ([struct \\_starpu\\_tag](#) \*tag)
- void [\\_starpu\\_notify\\_job\\_start\\_tag\\_dependencies](#) ([struct \\_starpu\\_tag](#) \*tag, [\\_starpu\\_notify\\_job\\_start\\_data](#) \*data)
- void [\\_starpu\\_tag\\_declare](#) ([starpu\\_tag\\_t](#) id, [struct \\_starpu\\_job](#) \*job)
- void [\\_starpu\\_tag\\_set\\_ready](#) ([struct \\_starpu\\_tag](#) \*tag)

### 6.71.1 Data Structure Documentation

#### 6.71.1.1 [struct \\_starpu\\_tag](#)

##### Data Fields

<a href="#">struct _starpu_spinlock</a>	lock	Lock for this structure. Locking order is in dependency order: a tag must not be locked before locking a tag it depends on
<a href="#">starpu_tag_t</a>	id	an identifier for the task
<a href="#">enum _starpu_tag_state</a>	state	
<a href="#">struct _starpu_cg_list</a>	tag_successors	
<a href="#">struct _starpu_job *</a>	job	which job is associated to the tag if any ?
unsigned	is_assigned	
unsigned	is_submitted	

## 6.71.2 Enumeration Type Documentation

### 6.71.2.1 `_starpu_tag_state`

```
enum _starpu_tag_state
```

Enumerator

<code>STARPU_INVALID_STATE</code>	this tag is not declared by any task
<code>STARPU_ASSOCIATED</code>	<code>_starpu_tag_declare</code> was called to associate the tag to a task
<code>STARPU_BLOCKED</code>	some task dependencies are not fulfilled yet
<code>STARPU_READY</code>	the task can be (or has been) submitted to the scheduler (all deps fulfilled)
<code>STARPU_DONE</code>	the task has been performed

## 6.71.3 Function Documentation

### 6.71.3.1 `_starpu_tag_set_ready()`

```
void _starpu_tag_set_ready (
    struct _starpu_tag * tag )
```

lock should be taken, and this releases it

## 6.72 task.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <core/jobs.h>
#include <common/starpu_spinlock.h>
```

### Data Structures

- `struct _starpu_trs_epoch`
- `struct starpu_transaction`

### Macros

- `#define _STARPU_JOB_UNSET`
- `#define _STARPU_JOB_SETTING`
- `#define _STARPU_TASK_SET_INTERFACE(task, interface, i)`
- `#define _STARPU_TASK_GET_INTERFACES(task)`

### Enumerations

- `enum _starpu_trs_state` { `_starpu_trs_uninitialized` , `_starpu_trs_initialized` }
- `enum _starpu_trs_epoch_state` { `_starpu_trs_epoch_uninitialized` , `_starpu_trs_epoch_inactive` , `_starpu_trs_epoch_confirmed` , `_starpu_trs_epoch_cancelled` , `_starpu_trs_epoch_terminated` }

## Functions

- void [\\_starpu\\_task\\_destroy](#) ([struct starpu\\_task](#) \*task)
- int [\\_starpu\\_task\\_test\\_termination](#) ([struct starpu\\_task](#) \*task)
- void [\\_starpu\\_task\\_init](#) (void)
- void [\\_starpu\\_task\\_deinit](#) (void)
- void [\\_starpu\\_set\\_current\\_task](#) ([struct starpu\\_task](#) \*task)
- int [\\_starpu\\_submit\\_job](#) ([struct \\_starpu\\_job](#) \*j, int ndeps)
- void [\\_starpu\\_task\\_declare\\_deps\\_array](#) ([struct starpu\\_task](#) \*task, unsigned ndeps, [struct starpu\\_task](#) \*task\_array[], int check)
- [struct \\_starpu\\_job](#) \* [\\_starpu\\_get\\_job\\_associated\\_to\\_task\\_slow](#) ([struct starpu\\_task](#) \*task, [struct \\_starpu\\_job](#) \*job)
- static [struct \\_starpu\\_job](#) \* [\\_starpu\\_get\\_job\\_associated\\_to\\_task](#) ([struct starpu\\_task](#) \*task)
- int [\\_starpu\\_task\\_submit\\_internally](#) ([struct starpu\\_task](#) \*task)
- int [\\_starpu\\_handle\\_needs\\_conversion\\_task](#) ([starpu\\_data\\_handle\\_t](#) handle, unsigned int node)
- int [\\_starpu\\_handle\\_needs\\_conversion\\_task\\_for\\_arch](#) ([starpu\\_data\\_handle\\_t](#) handle, enum [starpu\\_node\\_kind](#) node\_kind)
- void [\\_starpu\\_task\\_prepare\\_for\\_continuation\\_ext](#) (unsigned continuation\_resubmit, void(\*continuation\_↔callback\_on\_sleep)(void \*arg), void \*continuation\_callback\_on\_sleep\_arg)
- void [\\_starpu\\_task\\_prepare\\_for\\_continuation](#) (void)
- void [\\_starpu\\_task\\_set\\_omp\\_cleanup\\_callback](#) ([struct starpu\\_task](#) \*task, void(\*omp\_cleanup\_↔callback)(void \*arg), void \*omp\_cleanup\_callback\_arg)
- int [\\_starpu\\_task\\_uses\\_multiformat\\_handles](#) ([struct starpu\\_task](#) \*task)
- int [\\_starpu\\_task\\_submit\\_conversion\\_task](#) ([struct starpu\\_task](#) \*task, unsigned int workerid)
- void [\\_starpu\\_task\\_check\\_deprecated\\_fields](#) ([struct starpu\\_task](#) \*task)
- void [\\_starpu\\_codelet\\_check\\_deprecated\\_fields](#) ([struct starpu\\_codelet](#) \*cl)
- static [starpu\\_cpu\\_func\\_t](#) [\\_starpu\\_task\\_get\\_cpu\\_nth\\_implementation](#) ([struct starpu\\_codelet](#) \*cl, unsigned nimpl)
- static [starpu\\_cuda\\_func\\_t](#) [\\_starpu\\_task\\_get\\_cuda\\_nth\\_implementation](#) ([struct starpu\\_codelet](#) \*cl, unsigned nimpl)
- static [starpu\\_hip\\_func\\_t](#) [\\_starpu\\_task\\_get\\_hip\\_nth\\_implementation](#) ([struct starpu\\_codelet](#) \*cl, unsigned nimpl)
- static [starpu\\_opencl\\_func\\_t](#) [\\_starpu\\_task\\_get\\_opencl\\_nth\\_implementation](#) ([struct starpu\\_codelet](#) \*cl, unsigned nimpl)
- static [starpu\\_max\\_fpga\\_func\\_t](#) [\\_starpu\\_task\\_get\\_fpga\\_nth\\_implementation](#) ([struct starpu\\_codelet](#) \*cl, unsigned nimpl)
- static const char \* [\\_starpu\\_task\\_get\\_cpu\\_name\\_nth\\_implementation](#) ([struct starpu\\_codelet](#) \*cl, unsigned nimpl)
- void [\\_starpu\\_watchdog\\_init](#) (void)
- void [\\_starpu\\_watchdog\\_shutdown](#) (void)
- int [\\_starpu\\_task\\_wait\\_for\\_all\\_and\\_return\\_nb\\_waited\\_tasks](#) (void)
- int [\\_starpu\\_task\\_wait\\_for\\_all\\_in\\_ctx\\_and\\_return\\_nb\\_waited\\_tasks](#) (unsigned sched\_ctx)

### 6.72.1 Data Structure Documentation

#### 6.72.1.1 [struct \\_starpu\\_trs\\_epoch](#)

##### Data Fields

enum <a href="#">_starpu_trs_epoch_state</a>	state	
int	do_sync	if 1, the epoch entry task will wait on some user-supplied handle TODO: only used for first epoch on transaction opening for now, add for next epoch
int	is_begin	if 1, the epoch is the first of the transaction
int	is_end	if 1, the epoch will be the last, and the transaction will be closed after its execution

## Data Fields

void *	do_start_arg	inline argument supplied by the user and passed to the user function deciding whether to start or cancel the epoch execution
--------	--------------	--

## 6.72.2 Enumeration Type Documentation

### 6.72.2.1 `_starpu_trs_state`

enum `_starpu_trs_state`  
transaction states

### 6.72.2.2 `_starpu_trs_epoch_state`

enum `_starpu_trs_epoch_state`  
transaction epoch states

#### Enumerator

<code>_starpu_trs_epoch_inactive</code>	epoch is initialized but its entry task has not yet been executed to decide whether to confirm or cancel its execution
<code>_starpu_trs_epoch_confirmed</code>	epoch has been confirmed for execution, its tasks will be actually executed
<code>_starpu_trs_epoch_cancelled</code>	epoch has been cancelled, its task will be skipped
<code>_starpu_trs_epoch_terminated</code>	the exit task of the epoch has been executed

## 6.72.3 Function Documentation

### 6.72.3.1 `_starpu_task_destroy()`

```
void _starpu_task_destroy (
    struct starpu_task * task )
```

Internal version of `starpu_task_destroy`: don't check `task->destroy` flag

### 6.72.3.2 `_starpu_task_test_termination()`

```
int _starpu_task_test_termination (
    struct starpu_task * task )
```

Test for the termination of the task. Call `starpu_task_destroy` if required and the task is terminated.

### 6.72.3.3 `_starpu_task_init()`

```
void _starpu_task_init (
    void )
```

A pthread key is used to store the task currently executed on the thread. `_starpu_task_init` initializes this pthread key and `_starpu_set_current_task` updates its current value.

### 6.72.3.4 `_starpu_get_job_associated_to_task_slow()`

```
struct _starpu_job * _starpu_get_job_associated_to_task_slow (
    struct starpu_task * task,
    struct _starpu_job * job )
```

Returns the job structure (which is the internal data structure associated to a task).

### 6.72.3.5 `_starpu_task_submit_internally()`

```
int _starpu_task_submit_internally (
    struct starpu_task * task )
```

Submits starpu internal tasks to the initial context

### 6.72.3.6 `_starpu_task_prepare_for_continuation_ext()`

```
void _starpu_task_prepare_for_continuation_ext (
    unsigned continuation_resubmit,
    void(*) (void *arg) continuation_callback_on_sleep,
    void * continuation_callback_on_sleep_arg )
```

Prepare the current task for accepting new dependencies before becoming a continuation.

## 6.73 `task_bundle.h` File Reference

```
#include <starpu_thread.h>
```

### Data Structures

- [struct `\_starpu\_task\_bundle\_entry`](#)
- [struct `\_starpu\_task\_bundle`](#)
- [struct `\_starpu\_handle\_list`](#)

### Functions

- [void `\_starpu\_task\_bundle\_destroy`](#) ([starpu\\_task\\_bundle\\_t](#) bundle)
- [void `\_starpu\_insertion\_handle\_sorted`](#) ([struct `\_starpu\_handle\_list`](#) \*\*listp, [starpu\\_data\\_handle\\_t](#) handle, [enum `starpu\_data\_access\_mode`](#) mode)

### 6.73.1 Data Structure Documentation

#### 6.73.1.1 `struct _starpu_task_bundle_entry`

#### 6.73.2 `struct _starpu_task_bundle_entry`

#### 6.73.3 Purpose

Structure used to describe a linked list containing tasks in [\\_starpu\\_task\\_bundle](#).

#### 6.73.4 Fields

`task` Pointer to the task structure.

`next` Pointer to the next element in the linked list.

#### Data Fields

<a href="#">struct <code>starpu_task</code></a> *	<code>task</code>	
<a href="#">struct <code>_starpu_task_bundle_entry</code></a> *	<code>next</code>	

**6.73.4.1 struct \_starpu\_task\_bundle****6.73.5 struct \_starpu\_task\_bundle****6.73.6 Purpose**

Structure describing a list of tasks that should be scheduled on the same worker whenever it's possible. It must be considered as a hint given to the scheduler as there is no guarantee that they will be executed on the same worker.

**6.73.7 Fields**

mutex Mutex protecting the structure.

list Array of tasks included in the bundle.

closed Used to know if the user is still willing to add/remove some tasks in the bundle. Especially useful for the runtime to know whether it is safe to destroy a bundle.

**Data Fields**

starpu_pthread_mutex_t	mutex	Mutex protecting the bundle
struct _starpu_task_bundle_entry *	list	
int	closed	

**6.73.7.1 struct \_starpu\_handle\_list****6.73.8 struct \_starpu\_handle\_list****6.73.9 Purpose**

Structure describing a list of handles sorted by address to speed-up when looking for an element. The list cannot contain duplicate handles.

**6.73.10 Fields**

handle Pointer to the handle structure.

access\_mode Total access mode over the whole bundle.

next Pointer to the next element in the linked list.

**Data Fields**

starpu_data_handle_t	handle	
enum starpu_data_access_mode	mode	
struct _starpu_handle_list *	next	

**6.73.11 Function Documentation****6.73.11.1 \_starpu\_task\_bundle\_destroy()**

```
void _starpu_task_bundle_destroy (
    starpu_task_bundle_t bundle )
```

**6.73.12 \_starpu\_task\_bundle\_destroy****6.73.13 Purpose**

Destroy and deinitialize a bundle, memory previously allocated is freed.

### 6.73.14 Arguments

bundle (input) Bundle to destroy.

#### 6.73.14.1 `_starpu_insertion_handle_sorted()`

```
void _starpu_insertion_handle_sorted (
    struct _starpu_handle_list ** listp,
    starpu_data_handle_t handle,
    enum starpu_data_access_mode mode )
```

### 6.73.15 `_starpu_insertion_handle_sorted`

#### 6.73.16 Purpose

Insert an handle in a `_starpu_handle_list`, elements are sorted in increasing order, considering their physical address. As the list doesn't accept duplicate elements, a handle with the same address as an handle contained in the list is not inserted, but its mode access is merged with the one of the latter.

#### 6.73.17 Arguments

listp (input, output) Pointer to the first element of the list. In the case of an empty list or an inserted handle with small address, it should have changed when the call returns.

handle (input) Handle to insert in the list.

mode (input) Access mode of the handle.

## 6.74 `thread.h` File Reference

```
#include <common/utils.h>
#include <core/simgrid.h>
```

### Macros

- `#define starpu_pthread_spin_init`
- `#define starpu_pthread_spin_destroy`
- `#define starpu_pthread_spin_lock`
- `#define starpu_pthread_spin_trylock`
- `#define starpu_pthread_spin_unlock`

### Functions

- `static int _starpu_pthread_spin_init (starpu_pthread_spinlock_t *lock, int pshared STARPU_ATTRIBUTE_UNUSED)`
- `static int _starpu_pthread_spin_destroy (starpu_pthread_spinlock_t *lock STARPU_ATTRIBUTE_UNUSED)`
- `static int _starpu_pthread_spin_lock (starpu_pthread_spinlock_t *lock)`
- `static void _starpu_pthread_spin_checklocked (starpu_pthread_spinlock_t *lock STARPU_ATTRIBUTE_UNUSED)`
- `static int _starpu_pthread_spin_trylock (starpu_pthread_spinlock_t *lock)`
- `static int _starpu_pthread_spin_unlock (starpu_pthread_spinlock_t *lock)`

## 6.75 `timing.h` File Reference

```
#include <stdint.h>
#include <common/config.h>
#include <starpu.h>
```

```
#include <starp_uutil.h>
```

## Functions

- void `_starp_u_timing_init` (void)
- void `_starp_u_clock_gettime` (struct timespec \*ts) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT

### 6.75.1 Function Documentation

#### 6.75.1.1 `_starp_u_timing_init()`

```
void _starp_u_timing_init (
    void )
```

`_starp_u_timing_init` must be called prior to using any of these timing functions.

## 6.76 topology.h File Reference

```
#include <starp_u.h>
#include <common/config.h>
#include <common/list.h>
#include <common/fxt.h>
#include <common/uthash.h>
```

## Macros

- #define `ALLOC_WORKER_SET`
- #define `DEVID_PER_WORKER`
- #define `STARPU_NOWORKERID`
- #define `STARPU_ACTIVETHREAD`
- #define `STARPU_NONACTIVETHREAD`

## Functions

- int `_starp_u_build_topology` (struct `_starp_u_machine_config` \*config, int no\_mp\_config)
- void `_starp_u_initialize_workers_deviceid` (int \*explicit\_workers\_gpuid, int \*current, int \*workers\_gpuid, const char \*varname, unsigned nhwgpuid, enum `starp_u_worker_archtype` type)
- int `_starp_u_get_next_devid` (struct `_starp_u_machine_topology` \*topology, struct `_starp_u_machine_config` \*config, enum `starp_u_worker_archtype` arch)
- void `_starp_u_topology_check_ndevices` (int \*ndevices, unsigned nhwdevices, int overflow, unsigned max, int reserved, const char \*nname, const char \*dname, const char \*configurename)
- void `_starp_u_topology_configure_workers` (struct `_starp_u_machine_topology` \*topology, struct `_starp_u_machine_config` \*config, enum `starp_u_worker_archtype` type, int devnum, int devid, int homogeneous, int worker\_devid, unsigned nworker\_per\_device, unsigned ncores, struct `_starp_u_worker_set` \*worker\_set, struct `_starp_u_worker_set` \*driver\_worker\_set)
- unsigned `_starp_u_get_next_bindid` (struct `_starp_u_machine_config` \*config, unsigned flags, unsigned \*preferred\_binding, unsigned npreferred)
- void `_starp_u_destroy_machine_config` (struct `_starp_u_machine_config` \*config, int no\_mp\_config)
- void `_starp_u_destroy_topology` (struct `_starp_u_machine_config` \*config)
- hwloc\_obj\_t `_starp_u_numa_get_obj` (hwloc\_obj\_t obj)
- unsigned `_starp_u_topology_get_nhwcpu` (struct `_starp_u_machine_config` \*config)
- unsigned `_starp_u_topology_get_nhwpu` (struct `_starp_u_machine_config` \*config)
- unsigned `_starp_u_topology_get_nhwmanodes` (struct `_starp_u_machine_config` \*config)

- unsigned `_starpu_topology_get_nnumanodes` (`struct _starpu_machine_config *config`)
- unsigned `_starpu_topology_get_numa_core_binding` (`struct _starpu_machine_config *config`, `const unsigned *numa_binding`, `unsigned nnuma`, `unsigned *binding`, `unsigned nbinding`)
- int `starpu_memory_nodes_numa_hwloclogid_to_id` (`int logid`)
- int `_starpu_get_logical_numa_node_worker` (`unsigned workerid`)
- unsigned `_starpu_get_nhyperthreads` (`STARPU_ATTRIBUTE_VISIBILITY_DEFAULT`)
- void `_starpu_topology_filter` (`hwloc_topology_t topology`)
- int `_starpu_bind_thread_on_cpu` (`int cpuid`, `int workerid`, `const char *name`)
- void `_starpu_bind_thread_on_cpus` (`struct _starpu_combined_worker *combined_worker`)
- `struct _starpu_worker * _starpu_get_worker_from_driver` (`struct starpu_driver *d`)
- unsigned `starpu_memory_nodes_get_numa_count` (`void`) `STARPU_ATTRIBUTE_VISIBILITY_DEFAULT`
- int `starpu_memory_nodes_numa_id_to_hwloclogid` (`unsigned id`)
- int `_starpu_task_data_get_node_on_node` (`struct starpu_task *task`, `unsigned index`, `unsigned target_node`)
- int `_starpu_task_data_get_node_on_worker` (`struct starpu_task *task`, `unsigned index`, `unsigned worker`)

## Variables

- unsigned `_starpu_may_bind_automatically` [`STARPU_NARCH`]

## 6.76.1 Macro Definition Documentation

### 6.76.1.1 ALLOC\_WORKER\_SET

```
#define ALLOC_WORKER_SET
```

Configures the topology according to the desired worker distribution on the device.

- `homogeneous` tells to use `devid 0` for the `perfmmodel` (all devices have the same performance)
- `worker_devid` tells to set a `devid` per worker, and `subworkerid` to `0`, rather than sharing the `devid` and giving a different `subworkerid` to each worker. Request to allocate a worker set for each worker

### 6.76.1.2 DEVID\_PER\_WORKER

```
#define DEVID_PER_WORKER
```

Request to set a different `perfmmodel devid` per worker

## 6.76.2 Function Documentation

### 6.76.2.1 \_starpu\_build\_topology()

```
int _starpu_build_topology (
    struct _starpu_machine_config * config,
    int no_mp_config )
```

Detect the number of memory nodes and where to bind the different workers.

### 6.76.2.2 \_starpu\_initialize\_workers\_deviceid()

```
void _starpu_initialize_workers_deviceid (
    int * explicit_workers_gpuid,
    int * current,
    int * workers_gpuid,
    const char * varname,
    unsigned nhwgpus,
    enum starpu_worker_archtype type )
```

Initialize a series of workers.

- If `explicit_workers_gpuid` is non-null, it will be used as the list of device IDs of the actual hardware devices to be used.
- If `current` is non-null, it points to the next device ID to be used
- `workers_gpuid` is filled with the set of device IDs actually used in the end
- `varname` is the name of the environment variable that users can use to override the set of device IDs to be used.
- `nhwgpbus` is the number of actual devices available on the system.
- `type` is the type of devices.

### 6.76.2.3 `_starpu_get_next_devid()`

```
int _starpu_get_next_devid (
    struct _starpu_machine_topology * topology,
    struct _starpu_machine_config * config,
    enum starpu_worker_archtype arch )
```

Get the next devid for architecture type

### 6.76.2.4 `_starpu_topology_check_ndevices()`

```
void _starpu_topology_check_ndevices (
    int * ndevices,
    unsigned nhwdevices,
    int overflow,
    unsigned max,
    int reserved,
    const char * nname,
    const char * dname,
    const char * configurename )
```

Check that `*ndevices` is not larger than `nhwdevices` (unless `overflow` is 1), and is not larger than `max`. Cap it otherwise, and advise using the `configurename` `./configure` option in the `max` case.

### 6.76.2.5 `_starpu_get_next_bindid()`

```
unsigned _starpu_get_next_bindid (
    struct _starpu_machine_config * config,
    unsigned flags,
    unsigned * preferred_binding,
    unsigned npreferred )
```

This function gets the identifier of the next core on which to bind a worker. In case a list of preferred cores was specified (logical indexes), we look for an available core among the list if possible, otherwise a round-robin policy is used.

### 6.76.2.6 `_starpu_destroy_machine_config()`

```
void _starpu_destroy_machine_config (
    struct _starpu_machine_config * config,
    int no_mp_config )
```

Should be called instead of `_starpu_destroy_topology` when `_starpu_build_topology` returns a non zero value.

### 6.76.2.7 `_starpu_destroy_topology()`

```
void _starpu_destroy_topology (
    struct _starpu_machine_config * config )
```

Destroy all resources used to store the topology of the machine.

**6.76.2.8 \_starpu\_numa\_get\_obj()**

```
hwloc_obj_t _starpu_numa_get_obj (
    hwloc_obj_t obj )
```

Return the hwloc object of the NUMA node corresponding to the given hwloc object

**6.76.2.9 \_starpu\_topology\_get\_nhwcpu()**

```
unsigned _starpu_topology_get_nhwcpu (
    struct _starpu_machine_config * config )
```

returns the number of physical cpus

**6.76.2.10 \_starpu\_topology\_get\_nhwpu()**

```
unsigned _starpu_topology_get_nhwpu (
    struct _starpu_machine_config * config )
```

returns the number of logical cpus

**6.76.2.11 \_starpu\_topology\_get\_nhwnumanodes()**

```
unsigned _starpu_topology_get_nhwnumanodes (
    struct _starpu_machine_config * config )
```

returns the number of hardware NUMA nodes

**6.76.2.12 \_starpu\_topology\_get\_nnumanodes()**

```
unsigned _starpu_topology_get_nnumanodes (
    struct _starpu_machine_config * config )
```

returns the number of NUMA nodes to be exposed by StarPU as memory nodes, can be just 1 when STARPU\_USE\_NUMA is 0

**6.76.2.13 \_starpu\_topology\_get\_numa\_core\_binding()**

```
unsigned _starpu_topology_get_numa_core_binding (
    struct _starpu_machine_config * config,
    const unsigned * numa_binding,
    unsigned nnuma,
    unsigned * binding,
    unsigned nbinding )
```

given a list of numa nodes (logical indexes) *numa\_binding*, fill *binding* with the corresponding cores (logical indexes)

**6.76.2.14 \_starpu\_get\_nhyperthreads()**

```
unsigned _starpu_get_nhyperthreads ( )
```

returns the number of hyperthreads per core

**6.76.2.15 \_starpu\_topology\_filter()**

```
void _starpu_topology_filter (
    hwloc_topology_t topology )
```

Small convenient function to filter hwloc topology depending on HWLOC API version

**6.76.2.16 \_starpu\_bind\_thread\_on\_cpu()**

```
int _starpu_bind_thread_on_cpu (
    int cpuid,
    int workerid,
    const char * name )
```

Bind the current thread on the CPU logically identified by "cpuid". The logical ordering of the processors is either that of hwloc (if available), or the ordering exposed by the OS.

#### 6.76.2.17 `_starpu_bind_thread_on_cpus()`

```
void _starpu_bind_thread_on_cpus (
    struct _starpu_combined_worker * combined_worker )
```

Bind the current thread on the set of CPUs for the given combined worker.

#### 6.76.2.18 `_starpu_task_data_get_node_on_node()`

```
int _starpu_task_data_get_node_on_node (
    struct starpu_task * task,
    unsigned index,
    unsigned target_node )
```

Get the memory node for data number `i` when task is to be executed on memory node `target_node`. Returns -1 if the data does not need to be loaded.

#### 6.76.2.19 `_starpu_task_data_get_node_on_worker()`

```
int _starpu_task_data_get_node_on_worker (
    struct starpu_task * task,
    unsigned index,
    unsigned worker )
```

Get the memory node for data number `i` when task is to be executed on worker `worker`. Returns -1 if the data does not need to be loaded.

## 6.77 utils.h File Reference

```
#include <common/config.h>
#include <starpu.h>
#include <sys/stat.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>
```

### Macros

- `#define _STARPU_STRINGIFY(x)`
- `#define _STARPU_STRINGIFY(x)`
- `#define DO_CREQ_v_WW(_creqF, _ty1F, _arg1F, _ty2F, _arg2F)`
- `#define DO_CREQ_v_W(_creqF, _ty1F, _arg1F)`
- `#define ANNOTATE_HAPPENS_BEFORE(obj)`
- `#define ANNOTATE_HAPPENS_BEFORE_FORGET_ALL(obj)`
- `#define ANNOTATE_HAPPENS_AFTER(obj)`
- `#define VALGRIND_HG_DISABLE_CHECKING(start, len)`
- `#define VALGRIND_HG_ENABLE_CHECKING(start, len)`
- `#define VALGRIND_STACK_REGISTER(stackbottom, stacktop)`
- `#define VALGRIND_STACK_DEREGISTER(id)`
- `#define RUNNING_ON_VALGRIND`
- `#define STARPU_RUNNING_ON_VALGRIND`
- `#define STARPU_HG_DISABLE_CHECKING(variable)`
- `#define STARPU_HG_ENABLE_CHECKING(variable)`
- `#define STARPU_DEBUG_PREFIX`
- `#define _STARPU_UYIELD()`

- #define **STARPU\_VALGRIND\_YIELD()**
- #define **STARPU\_UYIELD()**
- #define **\_STARPU\_DEBUG**(fmt, ...)
- #define **\_STARPU\_DEBUG\_NO\_HEADER**(fmt, ...)
- #define **\_STARPU\_EXTRA\_DEBUG**(fmt, ...)
- #define **\_STARPU\_LOG\_IN()**
- #define **\_STARPU\_LOG\_OUT()**
- #define **\_STARPU\_LOG\_OUT\_TAG**(outtag)
- #define **HOST\_NAME\_MAX**
- #define **\_STARPU\_MSG**(fmt, ...)
- #define **\_STARPU\_DISP**(fmt, ...)
- #define **\_STARPU\_ERROR**(fmt, ...)
- #define **\_STARPU\_DECLTYPE**(x)
- #define **\_STARPU\_MALLOC**(ptr, size)
- #define **\_STARPU\_CALLOC**(ptr, nmemb, size)
- #define **\_STARPU\_REALLOC**(ptr, size)
- #define **\_STARPU\_IS\_ZERO**(a)

## Enumerations

- enum **initialization** { **UNINITIALIZED** , **CHANGING** , **INITIALIZED** }

## Functions

- char \* **\_starpu\_mkdtmp\_internal** (char \*tmpl) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- char \* **\_starpu\_mkdtmp** (char \*tmpl) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- int **\_starpu\_mkpath** (const char \*s, mode\_t mode)
- void **\_starpu\_mkpath\_and\_check** (const char \*s, mode\_t mode) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT ↔
- char \* **\_starpu\_mktemp** (const char \*directory, int flags, int \*fd)
- char \* **\_starpu\_mktemp\_many** (const char \*directory, int depth, int flags, int \*fd)
- void **\_starpu\_rmtmp\_many** (char \*path, int depth)
- void **\_starpu\_rmdir\_many** (char \*path, int depth)
- int **\_starpu\_fftruncate** (FILE \*file, size\_t length)
- int **\_starpu\_ftruncate** (int fd, size\_t length)
- int **\_starpu\_frlock** (FILE \*file)
- int **\_starpu\_frdunlock** (FILE \*file)
- int **\_starpu\_fwrlock** (FILE \*file)
- int **\_starpu\_fwrunlock** (FILE \*file)
- char \* **\_starpu\_get\_home\_path** (void)
- void **\_starpu\_gethostname** (char \*hostname, size\_t size) STARPU\_ATTRIBUTE\_VISIBILITY\_DEFAULT
- void **\_starpu\_drop\_comments** (FILE \*f)
- const char \* **\_starpu\_job\_get\_model\_name** (struct\_starpu\_job \*j)
- const char \* **\_starpu\_job\_get\_task\_name** (struct\_starpu\_job \*j)
- const char \* **\_starpu\_codelet\_get\_model\_name** (struct\_starpu\_codelet \*cl)
- const char \* **\_starpu\_codelet\_get\_name** (struct\_starpu\_codelet \*cl)
- int **\_starpu\_check\_mutex\_deadlock** (starpu\_pthread\_mutex\_t \*mutex)
- void **\_starpu\_util\_init** (void)

### 6.77.1 Function Documentation

**6.77.1.1 \_starpu\_mktemp\_many()**

```
char * _starpu_mktemp_many (
    const char * directory,
    int depth,
    int flags,
    int * fd )
```

This version creates a hierarchy of n temporary directories, useful when creating a lot of temporary files to be stored in the same place

**6.77.1.2 \_starpu\_drop\_comments()**

```
void _starpu_drop_comments (
    FILE * f )
```

If FILE is currently on a comment line, eat it.

**6.77.1.3 \_starpu\_job\_get\_model\_name()**

```
const char * _starpu_job_get_model_name (
    struct _starpu_job * j )
```

Returns the symbol associated to that job if any.

**6.77.1.4 \_starpu\_job\_get\_task\_name()**

```
const char * _starpu_job_get_task_name (
    struct _starpu_job * j )
```

Returns the name associated to that job if any.

**6.77.1.5 \_starpu\_codelet\_get\_model\_name()**

```
const char * _starpu_codelet_get_model_name (
    struct starpu_codelet * cl )
```

Returns the symbol associated to that job if any.

**6.77.1.6 \_starpu\_codelet\_get\_name()**

```
const char * _starpu_codelet_get_name (
    struct starpu_codelet * cl )
```

Returns the name of a codelet, or fallback to the name of the perfmodel.

**6.78 uthash.h File Reference**

```
#include <string.h>
#include <stddef.h>
#include <inttypes.h>
```

**Data Structures**

- struct [UT\\_hash\\_bucket](#)
- struct [UT\\_hash\\_table](#)
- struct [UT\\_hash\\_handle](#)

**Macros**

- #define **DECLTYPE(x)**
- #define **DECLTYPE\_ASSIGN(dst, src)**

- #define **UTHASH\_VERSION**
- #define **uthash\_fatal**(msg)
- #define **uthash\_malloc**(sz)
- #define **uthash\_free**(ptr, sz)
- #define **uthash\_noexpand\_fyi**(tbl)
- #define **uthash\_expand\_fyi**(tbl)
- #define **HASH\_INITIAL\_NUM\_BUCKETS**
- #define **HASH\_INITIAL\_NUM\_BUCKETS\_LOG2**
- #define **HASH\_BKT\_CAPACITY\_THRESH**
- #define **ELMT\_FROM\_HH**(tbl, hhp)
- #define **HASH\_FIND**(hh, head, keyptr, keylen, out)
- #define **HASH\_BLOOM\_MAKE**(tbl)
- #define **HASH\_BLOOM\_FREE**(tbl)
- #define **HASH\_BLOOM\_ADD**(tbl, hashv)
- #define **HASH\_BLOOM\_TEST**(tbl, hashv)
- #define **HASH\_MAKE\_TABLE**(hh, head)
- #define **HASH\_ADD**(hh, head, fieldname, keylen\_in, add)
- #define **HASH\_CHECK\_KEY**(hh, head, keyptr, keylen, out)
- #define **HASH\_ADD\_KEYPTR**(hh, head, keyptr, keylen\_in, add)
- #define **HASH\_TO\_BKT**(hashv, num\_bkts, bkt)
- #define **HASH\_DELETE**(hh, head, delptr)
- #define **HASH\_FIND\_STR**(head, findstr, out)
- #define **HASH\_ADD\_STR**(head, strfield, add)
- #define **HASH\_FIND\_INT**(head, findint, out)
- #define **HASH\_ADD\_INT**(head, intfield, add)
- #define **HASH\_FIND\_PTR**(head, findptr, out)
- #define **HASH\_ADD\_PTR**(head, ptrfield, add)
- #define **HASH\_DEL**(head, delptr)
- #define **HASH\_FSCK**(hh, head)
- #define **HASH\_EMIT\_KEY**(hh, head, keyptr, fieldlen)
- #define **HASH\_FCN**
- #define **HASH\_BER**(key, keylen, num\_bkts, hashv, bkt)
- #define **HASH\_SAX**(key, keylen, num\_bkts, hashv, bkt)
- #define **HASH\_FNV**(key, keylen, num\_bkts, hashv, bkt)
- #define **HASH\_OAT**(key, keylen, num\_bkts, hashv, bkt)
- #define **HASH\_JEN\_MIX**(a, b, c)
- #define **HASH\_JEN**(key, keylen, num\_bkts, hashv, bkt)
- #define **get16bits**(d)
- #define **HASH\_SFH**(key, keylen, num\_bkts, hashv, bkt)
- #define **HASH\_KEYCMP**(a, b, len)
- #define **HASH\_FIND\_IN\_BKT**(tbl, hh, head, keyptr, keylen\_in, out)
- #define **HASH\_ADD\_TO\_BKT**(head, addhh)
- #define **HASH\_DEL\_IN\_BKT**(hh, head, hh\_del)
- #define **HASH\_EXPAND\_BUCKETS**(tbl)
- #define **HASH\_SORT**(head, cmpfcn)
- #define **HASH\_SRT**(hh, head, cmpfcn)
- #define **HASH\_SELECT**(hh\_dst, dst, hh\_src, src, cond)
- #define **HASH\_CLEAR**(hh, head)
- #define **HASH\_ITER**(hh, head, el, tmp)
- #define **HASH\_COUNT**(head)
- #define **HASH\_CNT**(hh, head)
- #define **HASH\_SIGNATURE**
- #define **HASH\_BLOOM\_SIGNATURE**

## Typedefs

- typedef [struct UT\\_hash\\_bucket](#) **UT\_hash\_bucket**
- typedef [struct UT\\_hash\\_table](#) **UT\_hash\_table**
- typedef [struct UT\\_hash\\_handle](#) **UT\_hash\_handle**

### 6.78.1 Data Structure Documentation

#### 6.78.1.1 struct UT\_hash\_bucket

##### Data Fields

<a href="#">struct UT_hash_handle</a> *	hh_head	
unsigned	count	
unsigned	expand_mult	

#### 6.78.1.2 struct UT\_hash\_table

##### Data Fields

<a href="#">UT_hash_bucket</a> *	buckets	
unsigned	num_buckets	
unsigned	log2_num_buckets	
unsigned	num_items	
<a href="#">struct UT_hash_handle</a> *	tail	
ptrdiff_t	hho	
unsigned	ideal_chain_maxlen	
unsigned	nonideal_items	
unsigned	ineff_expands	
unsigned	noexpand	
uint32_t	signature	

#### 6.78.1.3 struct UT\_hash\_handle

##### Data Fields

<a href="#">struct UT_hash_table</a> *	tbl	
void *	prev	
void *	next	
<a href="#">struct UT_hash_handle</a> *	hh_prev	
<a href="#">struct UT_hash_handle</a> *	hh_next	
void *	key	
unsigned	keylen	
unsigned	hashv	

## 6.79 write\_back.h File Reference

```
#include <starp.h>
#include <datawizard/coherency.h>
```

## Functions

- void [\\_starpu\\_write\\_through\\_data](#) (starpu\_data\_handle\_t handle, unsigned requesting\_node, uint32\_t write\_through\_mask)

### 6.79.1 Function Documentation

#### 6.79.1.1 [\\_starpu\\_write\\_through\\_data\(\)](#)

```
void _starpu_write_through_data (
    starpu_data_handle_t handle,
    unsigned requesting_node,
    uint32_t write_through_mask )
```

If a write-through mask is associated to that data handle, this propagates the the current value of the data onto the different memory nodes in the write\_through\_mask.

## Chapter 7

# StarPU MPI File Documentation

### 7.1 `starpu_mpi_cache.h` File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

#### Functions

- `void _starpu_mpi_cache_init` (MPI\_Comm comm)
- `void _starpu_mpi_cache_shutdown` (void)
- `void _starpu_mpi_cache_data_init` (starpu\_data\_handle\_t data\_handle)
- `void _starpu_mpi_cache_data_clear` (starpu\_data\_handle\_t data\_handle)

#### Variables

- `int _starpu_cache_enabled`

### 7.2 `starpu_mpi_driver.h` File Reference

```
#include <starpu.h>
```

#### Functions

- `void _starpu_mpi_driver_init` (struct starpu\_conf \*conf)
- `void _starpu_mpi_driver_shutdown` ()

### 7.3 `starpu_mpi_init.h` File Reference

```
#include <starpu.h>
#include <starpu_mpi.h>
```

#### Functions

- `void _starpu_mpi_do_initialize` (struct \_starpu\_mpi\_argc\_argv \*argc\_argv)

## 7.4 starpu\_mpi\_nmad\_backend.h File Reference

```
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <nm_sendrecv_interface.h>
#include <nm_session_interface.h>
#include <nm_mpi_nmad.h>
```

### Data Structures

- [struct\\_starpu\\_mpi\\_req\\_backend](#)

### 7.4.1 Data Structure Documentation

#### 7.4.1.1 struct\_starpu\_mpi\_req\_backend

##### Data Fields

MPI_Request	data_request	
starpu_pthread_mutex_t	req_mutex	
starpu_pthread_cond_t	req_cond	
starpu_pthread_cond_t	posted_cond	
<a href="#">struct_starpu_mpi_req</a> *	other_request	In the case of a Wait/Test request, we are going to post a request to test the completion of another request
MPI_Request	size_req	
<a href="#">struct_starpu_mpi_envelope</a> *	envelope	
unsigned	is_internal_req:1	
unsigned	to_destroy:1	
<a href="#">struct_starpu_mpi_req</a> *	internal_req	
<a href="#">struct_starpu_mpi_early_data_handle</a> *	early_data_handle	
<a href="#">UT_hash_handle</a>	hh	
nm_gate_t	gate	
nm_session_t	session	
nm_sr_request_t	data_request	
piom_cond_t	req_cond	
int	posted	
int	has_received_data	
int	finalized	
int	to_destroy	
<a href="#">struct_starpu_spinlock</a>	finalized_to_destroy_lock	
<a href="#">struct nm_data_s</a>	unknown_datatype_data	When datatype is unknown
<a href="#">struct iovec</a>	unknown_datatype_v[2]	

## 7.5 starpu\_mpi\_stats.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
```

## Functions

- void [\\_starpu\\_mpi\\_comm\\_amounts\\_init](#) (MPI\_Comm comm)
- void [\\_starpu\\_mpi\\_comm\\_stats\\_disable](#) (void)
- void [\\_starpu\\_mpi\\_comm\\_stats\\_enable](#) (void)
- void [\\_starpu\\_mpi\\_comm\\_amounts\\_shutdown](#) (void)
- void [\\_starpu\\_mpi\\_comm\\_amounts\\_inc](#) (MPI\_Comm comm, unsigned memnode, unsigned dst, MPI\_Datatype datatype, int count)
- void [\\_starpu\\_mpi\\_nb\\_coop\\_inc](#) (int nb\_nodes\_in\_coop)
- void [\\_starpu\\_mpi\\_comm\\_amounts\\_display](#) (FILE \*stream, int node)

## 7.6 starpu\_mpi\_cache\_stats.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

## Macros

- [#define \\_starpu\\_mpi\\_cache\\_stats\\_inc](#)(dst, data\_handle)
- [#define \\_starpu\\_mpi\\_cache\\_stats\\_dec](#)(dst, data\_handle)

## Functions

- void [\\_starpu\\_mpi\\_cache\\_stats\\_init](#) ()
- void [\\_starpu\\_mpi\\_cache\\_stats\\_shutdown](#) ()
- void [\\_starpu\\_mpi\\_cache\\_stats\\_update](#) (unsigned dst, starpu\_data\_handle\_t data\_handle, int count)

## 7.7 starpu\_mpi\_early\_data.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
#include <common/uthash.h>
#include <starpu_mpi_private.h>
```

## Data Structures

- [struct \\_starpu\\_mpi\\_early\\_data\\_handle](#)
- [struct \\_starpu\\_mpi\\_early\\_data\\_handle\\_tag\\_hashlist](#)

## Functions

- void [\\_starpu\\_mpi\\_early\\_data\\_init](#) (void)
- void [\\_starpu\\_mpi\\_early\\_data\\_check\\_termination](#) (void)
- void [\\_starpu\\_mpi\\_early\\_data\\_shutdown](#) (void)
- [struct \\_starpu\\_mpi\\_early\\_data\\_handle \\* \\_starpu\\_mpi\\_early\\_data\\_create](#) ([struct \\_starpu\\_mpi\\_envelope](#) \*envelope, int source, MPI\_Comm comm) STARPU\_ATTRIBUTE\_MALLOC
- [struct \\_starpu\\_mpi\\_early\\_data\\_handle \\* \\_starpu\\_mpi\\_early\\_data\\_find](#) ([struct \\_starpu\\_mpi\\_node\\_tag](#) \*node\_tag)
- void [\\_starpu\\_mpi\\_early\\_data\\_add](#) ([struct \\_starpu\\_mpi\\_early\\_data\\_handle](#) \*early\_data\_handle)
- void [\\_starpu\\_mpi\\_early\\_data\\_delete](#) ([struct \\_starpu\\_mpi\\_early\\_data\\_handle](#) \*early\_data\_handle)

- [struct\\_starpu\\_mpi\\_early\\_data\\_handle\\_tag\\_hashlist](#) \* [\\_starpu\\_mpi\\_early\\_data\\_extract](#) ([struct\\_starpu\\_mpi\\_node\\_tag](#) \*node\_tag)

## 7.7.1 Data Structure Documentation

### 7.7.1.1 struct\_starpu\_mpi\_early\_data\_handle

#### Data Fields

<a href="#">starpu_data_handle_t</a>	handle	
<a href="#">struct_starpu_mpi_req</a> *	req	
void *	buffer	
size_t	size	
unsigned	buffer_node	
<a href="#">struct_starpu_mpi_node_tag</a>	node_tag	
<a href="#">starpu_pthread_mutex_t</a>	req_mutex	
<a href="#">starpu_pthread_cond_t</a>	req_cond	

### 7.7.1.2 struct\_starpu\_mpi\_early\_data\_handle\_tag\_hashlist

#### Data Fields

<a href="#">struct_starpu_mpi_early_data_handle_list</a>	list	
<a href="#">UT_hash_handle</a>	hh	
<a href="#">starpu_mpi_tag_t</a>	data_tag	

## 7.8 starpu\_mpi\_mpi.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

### Functions

- [int\\_starpu\\_mpi\\_progress\\_init](#) ([struct\\_starpu\\_mpi\\_argc\\_argv](#) \*argc\_argv)
- [void\\_starpu\\_mpi\\_progress\\_shutdown](#) (void \*\*value)
- [void\\_starpu\\_mpi\\_wait\\_for\\_initialization](#) ()
- [int\\_starpu\\_mpi\\_barrier](#) (MPI\_Comm comm)
- [int\\_starpu\\_mpi\\_wait\\_for\\_all](#) (MPI\_Comm comm)
- [int\\_starpu\\_mpi\\_wait](#) ([starpu\\_mpi\\_req](#) \*public\_req, MPI\_Status \*status)
- [int\\_starpu\\_mpi\\_test](#) ([starpu\\_mpi\\_req](#) \*public\_req, int \*flag, MPI\_Status \*status)
- [void\\_starpu\\_mpi\\_wake\\_up\\_progress\\_thread](#) ()
- [void\\_starpu\\_mpi\\_isend\\_size\\_func](#) ([struct\\_starpu\\_mpi\\_req](#) \*req)
- [void\\_starpu\\_mpi\\_irecv\\_size\\_func](#) ([struct\\_starpu\\_mpi\\_req](#) \*req)

## 7.9 starpu\_mpi\_nmad\_unknown\_datatype.h File Reference

```
#include <common/config.h>
#include <nm_sendrecv_interface.h>
#include <nm_mpi_nmad.h>
```

## Functions

- void `_starpu_mpi_isend_prepare_unknown_datatype` ([struct \\_starpu\\_mpi\\_req](#) \*req, [struct nm\\_data\\_s](#) \*data)
- void `_starpu_mpi_isend_unknown_datatype` ([struct \\_starpu\\_mpi\\_req](#) \*req)
- void `_starpu_mpi_irecv_unknown_datatype` ([struct \\_starpu\\_mpi\\_req](#) \*req)

## 7.10 starpu\_mpi\_sync\_data.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

## Functions

- void `_starpu_mpi_sync_data_init` (void)
- void `_starpu_mpi_sync_data_check_termination` (void)
- void `_starpu_mpi_sync_data_shutdown` (void)
- [struct \\_starpu\\_mpi\\_req](#) \* `_starpu_mpi_sync_data_find` ([starpu\\_mpi\\_tag\\_t](#) data\_tag, int source, [MPI\\_Comm](#) comm)
- void `_starpu_mpi_sync_data_add` ([struct \\_starpu\\_mpi\\_req](#) \*req)
- int `_starpu_mpi_sync_data_count` ()

## 7.11 starpu\_mpi\_comm.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <mpi/starpu_mpi_mpi_backend.h>
```

## Functions

- void `_starpu_mpi_comm_init` ([MPI\\_Comm](#) comm)
- void `_starpu_mpi_comm_shutdown` ()
- void `_starpu_mpi_comm_register` ([MPI\\_Comm](#) comm)
- void `_starpu_mpi_comm_post_recv` ()
- int `_starpu_mpi_comm_test_recv` ([MPI\\_Status](#) \*status, [struct \\_starpu\\_mpi\\_envelope](#) \*\*envelope, [MPI\\_Comm](#) \*comm)
- void `_starpu_mpi_comm_cancel_recv` ()

## 7.12 starpu\_mpi\_early\_request.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

## Data Structures

- [struct \\_starpu\\_mpi\\_early\\_request\\_tag\\_hashlist](#)

## Functions

- void `_starpu_mpi_early_request_init` (void)
- void `_starpu_mpi_early_request_shutdown` (void)
- int `_starpu_mpi_early_request_count` (void)
- void `_starpu_mpi_early_request_check_termination` (void)
- void `_starpu_mpi_early_request_enqueue` (`struct _starpu_mpi_req` \*req)
- `struct _starpu_mpi_req` \* `_starpu_mpi_early_request_dequeue` (`starpu_mpi_tag_t` data\_tag, int source, MPI\_Comm comm)
- `struct _starpu_mpi_early_request_tag_hashlist` \* `_starpu_mpi_early_request_extract` (`starpu_mpi_tag_t` data\_tag, int source, MPI\_Comm comm)

### 7.12.1 Data Structure Documentation

#### 7.12.1.1 `struct _starpu_mpi_early_request_tag_hashlist`

##### Data Fields

<code>struct _starpu_mpi_req_list</code>	list	
<code>UT_hash_handle</code>	hh	
<code>starpu_mpi_tag_t</code>	data_tag	

## 7.13 `starpu_mpi_mpi_backend.h` File Reference

```
#include <common/config.h>
#include <common/uthash.h>
```

### Data Structures

- `struct _starpu_mpi_envelope`
- `struct _starpu_mpi_req_backend`

### Macros

- `#define _STARPU_MPI_TAG_ENVELOPE`
- `#define _STARPU_MPI_TAG_DATA`
- `#define _STARPU_MPI_TAG_SYNC_DATA`
- `#define _STARPU_MPI_TAG_CP_ACK`
- `#define _STARPU_MPI_TAG_CP_RCVRY`
- `#define _STARPU_MPI_TAG_EXT_DATA`
- `#define _STARPU_MPI_TAG_CP_INFO`

### Enumerations

- `enum _starpu_envelope_mode` { `_STARPU_MPI_ENVELOPE_DATA` , `_STARPU_MPI_ENVELOPE_SYNC_READY` }

### Variables

- int `_starpu_mpi_tag`

### 7.13.1 Data Structure Documentation

#### 7.13.1.1 `struct _starpu_mpi_envelope`

## Data Fields

enum_starpu_envelope_mode	mode	
starpu_ssize_t	size	
starpu_mpi_tag_t	data_tag	
unsigned	sync	

## 7.13.1.2 struct\_starpu\_mpi\_req\_backend

## Data Fields

MPI_Request	data_request	
starpu_pthread_mutex_t	req_mutex	
starpu_pthread_cond_t	req_cond	
starpu_pthread_cond_t	posted_cond	
struct_starpu_mpi_req *	other_request	In the case of a Wait/Test request, we are going to post a request to test the completion of another request
MPI_Request	size_req	
struct_starpu_mpi_envelope *	envelope	
unsigned	is_internal_req:1	
unsigned	to_destroy:1	
struct_starpu_mpi_req *	internal_req	
struct_starpu_mpi_early_data_handle *	early_data_handle	
UT_hash_handle	hh	
nm_gate_t	gate	
nm_session_t	session	
nm_sr_request_t	data_request	
piom_cond_t	req_cond	
int	posted	
int	has_received_data	
int	finalized	
int	to_destroy	
struct_starpu_spinlock	finalized_to_destroy_lock	
struct nm_data_s	unknown_datatype_data	When datatype is unknown
struct iovec	unknown_datatype_v[2]	

## 7.14 starpu\_mpi\_private.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/uthash.h>
#include <starpu_mpi.h>
#include <starpu_mpi_fxt.h>
#include <common/list.h>
#include <common/prio_list.h>
#include <common/starpu_spinlock.h>
#include <core/simgrid.h>
```

## Data Structures

- struct [\\_starpu\\_simgrid\\_mpi\\_req](#)
- struct [\\_starpu\\_mpi\\_node](#)
- struct [\\_starpu\\_mpi\\_node\\_tag](#)
- struct [\\_starpu\\_mpi\\_coop\\_sends](#)
- struct [\\_starpu\\_mpi\\_data](#)
- struct [\\_starpu\\_mpi\\_req](#)
- struct [\\_starpu\\_mpi\\_argc\\_argv](#)
- struct [\\_starpu\\_mpi\\_backend](#)

## Macros

- #define [STARPU\\_MPI\\_ASSERT\\_MSG](#)(x, msg, ...)
- #define [\\_STARPU\\_MPI\\_MALLOC](#)(ptr, size)
- #define [\\_STARPU\\_MPI\\_CALLOC](#)(ptr, nmemb, size)
- #define [\\_STARPU\\_MPI\\_REALLOC](#)(ptr, size)
- #define [\\_STARPU\\_MPI\\_COMM\\_DEBUG](#)(ptr, count, datatype, node, tag, utag, comm, way)
- #define [\\_STARPU\\_MPI\\_COMM\\_TO\\_DEBUG](#)(ptr, count, datatype, dest, tag, utag, comm)
- #define [\\_STARPU\\_MPI\\_COMM\\_FROM\\_DEBUG](#)(ptr, count, datatype, source, tag, utag, comm)
- #define [\\_STARPU\\_MPI\\_DEBUG](#)(level, fmt, ...)
- #define [\\_STARPU\\_MPI\\_DISP](#)(fmt, ...)
- #define [\\_STARPU\\_MPI\\_MSG](#)(fmt, ...)
- #define [\\_STARPU\\_MPI\\_LOG\\_IN](#)()
- #define [\\_STARPU\\_MPI\\_LOG\\_OUT](#)()

## Enumerations

- enum [\\_starpu\\_mpi\\_request\\_type](#) {  
**SEND\_REQ**, **RECV\_REQ**, **WAIT\_REQ**, **TEST\_REQ**,  
**BARRIER\_REQ**, **PROBE\_REQ**, **UNKNOWN\_REQ** }

## Functions

- int [\\_starpu\\_mpi\\_simgrid\\_mpi\\_test](#) (unsigned \*done, int \*flag)
- void [\\_starpu\\_mpi\\_simgrid\\_wait\\_req](#) (MPI\_Request \*request, MPI\_Status \*status, starpu\_thread ↔ queue\_t \*queue, unsigned \*done)
- [struct \\_starpu\\_mpi\\_req](#) \* [\\_starpu\\_mpi\\_isend\\_cache\\_aware](#) (starpu\_data\_handle\_t data\_handle, int dest, starpu\_mpi\_tag\_t data\_tag, MPI\_Comm comm, unsigned detached, unsigned sync, int prio, void(\*callback)(void \*), void \* \_arg, int sequential\_consistency, int \*cache\_flag)
- [struct \\_starpu\\_mpi\\_req](#) \* [\\_starpu\\_mpi\\_irecv\\_cache\\_aware](#) (starpu\_data\_handle\_t data\_handle, int source, starpu\_mpi\_tag\_t data\_tag, MPI\_Comm comm, unsigned detached, unsigned sync, void(\*callback)(void \*), void \* \_arg, int sequential\_consistency, int is\_internal\_req, starpu\_ssize\_t count, int \*cache\_flag)
- char \* [\\_starpu\\_mpi\\_get\\_mpi\\_error\\_code](#) (int code)
- void [\\_starpu\\_mpi\\_env\\_init](#) (void)
- [struct \\_starpu\\_mpi\\_data](#) \* [\\_starpu\\_mpi\\_data\\_get](#) (starpu\_data\_handle\_t data\_handle)
- void [\\_starpu\\_mpi\\_submit\\_ready\\_request](#) (void \*arg)
- void [\\_starpu\\_mpi\\_release\\_req\\_data](#) (struct [\\_starpu\\_mpi\\_req](#) \*req)
- void [\\_starpu\\_mpi\\_isend\\_irecv\\_common](#) (struct [\\_starpu\\_mpi\\_req](#) \*req, enum starpu\_data\_access\_mode mode, int sequential\_consistency)
- void [\\_starpu\\_mpi\\_coop\\_send](#) (starpu\_data\_handle\_t data\_handle, [struct \\_starpu\\_mpi\\_req](#) \*req, enum starpu\_data\_access\_mode mode, int sequential\_consistency)
- void [\\_starpu\\_mpi\\_submit\\_coop\\_sends](#) (struct [\\_starpu\\_mpi\\_coop\\_sends](#) \*coop\_sends, int submit\_control, int submit\_data)
- void [\\_starpu\\_mpi\\_redux\\_fill\\_post\\_sync\\_jobid](#) (const void \*const redux\_data\_args, long \*const post ↔ sync\_jobid)

- void `_starpu_mpi_request_init` (`struct _starpu_mpi_req **req`)
- `struct _starpu_mpi_req * _starpu_mpi_request_fill` (`starpu_data_handle_t data_handle`, `int srcdst`, `starpu_mpi_tag_t data_tag`, `MPI_Comm comm`, `unsigned detached`, `unsigned sync`, `int prio`, `void(*callback)(void *)`, `void *arg`, `enum _starpu_mpi_request_type request_type`, `void(*func)(struct _starpu_mpi_req *)`, `int sequential_consistency`, `int is_internal_req`, `starpu_ssize_t count`)
- void `_starpu_mpi_request_destroy` (`struct _starpu_mpi_req *req`)
- char \* `_starpu_mpi_request_type` (`enum _starpu_mpi_request_type request_type`)
- `struct _starpu_mpi_req * _starpu_mpi_irecv_common` (`starpu_data_handle_t data_handle`, `int source`, `starpu_mpi_tag_t data_tag`, `MPI_Comm comm`, `unsigned detached`, `unsigned sync`, `void(*callback)(void *)`, `void *arg`, `int sequential_consistency`, `int is_internal_req`, `starpu_ssize_t count`, `int prio`)
- int `_starpu_mpi_choose_node` (`starpu_data_handle_t data_handle`, `enum starpu_data_access_mode mode`)
- void `_starpu_mpi_data_flush` (`starpu_data_handle_t data_handle`)
- void `_starpu_mpi_tags_init` (`void`)

## Variables

- `starpu_pthread_wait_t _starpu_mpi_thread_wait`
- `starpu_pthread_queue_t _starpu_mpi_thread_dontsleep`
- int `_starpu_debug_rank`
- int `_starpu_mpi_comm_debug`
- int `_starpu_mpi_fake_world_size`
- int `_starpu_mpi_fake_world_rank`
- int `_starpu_mpi_use_prio`
- int `_starpu_mpi_nobind`
- int `_starpu_mpi_thread_cpuid`
- int `_starpu_mpi_thread_multiple_send`
- int `_starpu_mpi_use_coop_sends`
- int `_starpu_mpi_mem_throttle`
- int `_starpu_mpi_rcv_wait_finalize`
- int `_starpu_mpi_has_cuda`
- int `_starpu_mpi_cuda_devid`
- `PRIO_struct _starpu_mpi_req`
- `struct _starpu_mpi_backend _mpi_backend`

### 7.14.1 Data Structure Documentation

#### 7.14.1.1 struct\_starpu\_simgrid\_mpi\_req

##### Data Fields

<code>MPI_Request *</code>	<code>request</code>	
<code>MPI_Status *</code>	<code>status</code>	
<code>starpu_pthread_queue_t *</code>	<code>queue</code>	
<code>unsigned *</code>	<code>done</code>	

#### 7.14.1.2 struct\_starpu\_mpi\_node

##### Data Fields

<code>MPI_Comm</code>	<code>comm</code>	
<code>int</code>	<code>rank</code>	

### 7.14.1.3 struct\_starpu\_mpi\_node\_tag

#### Data Fields

<a href="#">struct_starpu_mpi_node</a>	node	
starpu_mpi_tag_t	data_tag	

### 7.14.1.4 struct\_starpu\_mpi\_coop\_sends

One bag of cooperative sends

#### Data Fields

starpu_data_handle_t	data_handle	
<a href="#">struct_starpu_mpi_req_multilist_coop_sends</a>	reqs	List of send requests
<a href="#">struct_starpu_mpi_data *</a>	mpi_data	
<a href="#">struct_starpu_spinlock</a>	lock	Array of send requests, after sorting out
<a href="#">struct_starpu_mpi_req **</a>	reqs_array	
unsigned	n	
unsigned	redirects_sent	
long	pre_sync_jobid	

### 7.14.1.5 struct\_starpu\_mpi\_data

Initialized in starpu\_mpi\_data\_register\_comm

#### Data Fields

int	magic	
<a href="#">struct_starpu_mpi_node_tag</a>	node_tag	
char *	cache_sent	
unsigned int	cache_received	
unsigned int	ft_induced_cache_received:1	
unsigned int	ft_induced_cache_received_count:1	
unsigned int	modified:1	
char *	redux_map	Array used to store the contributing nodes to this data when it is accessed in (MPI_)REDUX mode.
<a href="#">struct_starpu_spinlock</a>	coop_lock	Rendez-vous data for opportunistic cooperative sends, Needed to synchronize between submit thread and workers
<a href="#">struct_starpu_mpi_coop_sends *</a>	coop_sends	Current cooperative send bag
unsigned	nb_future_sends	When provided, wait the given number of sends to start a coop, instead of just waiting that data are ready

### 7.14.1.6 struct\_starpu\_mpi\_argc\_argv

## Data Fields

int	initialize_mpi	
int *	argc	
char ***	argv	
MPI_Comm	comm	
int	fargc	Fortran argc
char **	fargv	Fortran argv
int	rank	
int	world_size	

## 7.14.2 Function Documentation

7.14.2.1 `_starpu_mpi_submit_ready_request()`

```
void _starpu_mpi_submit_ready_request (
    void * arg )
```

To be called to actually submit the request

7.14.2.2 `_starpu_mpi_release_req_data()`

```
void _starpu_mpi_release_req_data (
    struct _starpu_mpi_req * req )
```

To be called when request is completed

7.14.2.3 `_starpu_mpi_coop_send()`

```
void _starpu_mpi_coop_send (
    starpu_data_handle_t data_handle,
    struct _starpu_mpi_req * req,
    enum starpu_data_access_mode mode,
    int sequential_consistency )
```

Try to merge with send request with other send requests

7.14.2.4 `_starpu_mpi_submit_coop_sends()`

```
void _starpu_mpi_submit_coop_sends (
    struct _starpu_mpi_coop_sends * coop_sends,
    int submit_control,
    int submit_data )
```

Actually submit the coop\_sends bag to MPI. At least one of submit\_control or submit\_data is true. `_starpu_mpi_submit_coop_sends` may be called either

- just once with both parameters being true,
- or once with submit\_control being true (data is not available yet, but we can send control messages), and a second time with submit\_data being true. Or the converse, possibly on different threads, etc.

7.14.2.5 `_starpu_mpi_tags_init()`

```
void _starpu_mpi_tags_init (
    void )
```

To be called at initialization to set up the tags upper bound

## 7.15 starpu\_mpi\_tag.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

### Functions

- void **\_starpu\_mpi\_tag\_init** (void)
- void **\_starpu\_mpi\_tag\_shutdown** (void)
- void **\_starpu\_mpi\_tag\_data\_register** (starpu\_data\_handle\_t handle, starpu\_mpi\_tag\_t data\_tag)
- int **\_starpu\_mpi\_tag\_data\_release** (starpu\_data\_handle\_t handle)
- starpu\_data\_handle\_t **\_starpu\_mpi\_tag\_get\_data\_handle\_from\_tag** (starpu\_mpi\_tag\_t data\_tag)

## 7.16 starpu\_mpi\_datatype.h File Reference

```
#include <starpu_mpi.h>
#include <starpu_mpi_private.h>
```

### Functions

- void **\_starpu\_mpi\_datatype\_init** (void)
- void **\_starpu\_mpi\_datatype\_shutdown** (void)
- void **\_starpu\_mpi\_datatype\_allocate** (starpu\_data\_handle\_t data\_handle, struct **\_starpu\_mpi\_req** \*req)
- void **\_starpu\_mpi\_datatype\_free** (starpu\_data\_handle\_t data\_handle, MPI\_Datatype \*datatype)
- MPI\_Datatype **\_starpu\_mpi\_datatype\_get\_user\_defined\_datatype** (starpu\_data\_handle\_t data\_handle, unsigned node)

## 7.17 starpu\_mpi\_fxt.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/fxt.h>
```

### Macros

- #define **\_STARPU\_MPI\_FUT\_POINT\_TO\_POINT\_SEND**
- #define **\_STARPU\_MPI\_FUT\_COLLECTIVE\_SEND**
- #define **\_STARPU\_MPI\_FUT\_START**
- #define **\_STARPU\_MPI\_FUT\_STOP**
- #define **\_STARPU\_MPI\_FUT\_BARRIER**
- #define **\_STARPU\_MPI\_FUT\_ISEND\_SUBMIT\_BEGIN**
- #define **\_STARPU\_MPI\_FUT\_ISEND\_SUBMIT\_END**
- #define **\_STARPU\_MPI\_FUT\_IRecv\_SUBMIT\_BEGIN**
- #define **\_STARPU\_MPI\_FUT\_IRecv\_SUBMIT\_END**
- #define **\_STARPU\_MPI\_FUT\_ISEND\_COMPLETE\_BEGIN**
- #define **\_STARPU\_MPI\_FUT\_ISEND\_COMPLETE\_END**
- #define **\_STARPU\_MPI\_FUT\_DATA\_SET\_RANK**
- #define **\_STARPU\_MPI\_FUT\_IRecv\_TERMINATED**
- #define **\_STARPU\_MPI\_FUT\_ISEND\_TERMINATED**
- #define **\_STARPU\_MPI\_FUT\_TESTING\_DETACHED\_BEGIN**
- #define **\_STARPU\_MPI\_FUT\_TESTING\_DETACHED\_END**

- #define \_STARPU\_MPI\_FUT\_TEST\_BEGIN
- #define \_STARPU\_MPI\_FUT\_TEST\_END
- #define \_STARPU\_MPI\_FUT\_IRecv\_COMPLETE\_BEGIN
- #define \_STARPU\_MPI\_FUT\_IRecv\_COMPLETE\_END
- #define \_STARPU\_MPI\_FUT\_SLEEP\_BEGIN
- #define \_STARPU\_MPI\_FUT\_SLEEP\_END
- #define \_STARPU\_MPI\_FUT\_DTESTING\_BEGIN
- #define \_STARPU\_MPI\_FUT\_DTESTING\_END
- #define \_STARPU\_MPI\_FUT\_UTESTING\_BEGIN
- #define \_STARPU\_MPI\_FUT\_UTESTING\_END
- #define \_STARPU\_MPI\_FUT\_UWAIT\_BEGIN
- #define \_STARPU\_MPI\_FUT\_UWAIT\_END
- #define \_STARPU\_MPI\_FUT\_POLLING\_BEGIN
- #define \_STARPU\_MPI\_FUT\_POLLING\_END
- #define \_STARPU\_MPI\_FUT\_DRIVER\_RUN\_BEGIN
- #define \_STARPU\_MPI\_FUT\_DRIVER\_RUN\_END
- #define \_STARPU\_MPI\_FUT\_DATA\_SET\_TAG
- #define \_STARPU\_MPI\_FUT\_IRecv\_NUMA\_NODE
- #define \_STARPU\_MPI\_FUT\_ISEND\_NUMA\_NODE
- #define \_STARPU\_MPI\_FUT\_CHECKPOINT\_BEGIN
- #define \_STARPU\_MPI\_FUT\_CHECKPOINT\_END
- #define \_STARPU\_MPI\_TRACE\_START(a, b)
- #define \_STARPU\_MPI\_TRACE\_STOP(a, b)
- #define \_STARPU\_MPI\_TRACE\_BARRIER(a, b, c, d)
- #define \_STARPU\_MPI\_TRACE\_ISEND\_SUBMIT\_BEGIN(a, b, c)
- #define \_STARPU\_MPI\_TRACE\_ISEND\_SUBMIT\_END(a, b, c)
- #define \_STARPU\_MPI\_TRACE\_IRecv\_SUBMIT\_BEGIN(a, b)
- #define \_STARPU\_MPI\_TRACE\_IRecv\_SUBMIT\_END(a, b)
- #define \_STARPU\_MPI\_TRACE\_ISEND\_COMPLETE\_BEGIN(a, b, c)
- #define \_STARPU\_MPI\_TRACE\_COMPLETE\_BEGIN(a, b, c)
- #define \_STARPU\_MPI\_TRACE\_COMPLETE\_END(a, b, c)
- #define \_STARPU\_MPI\_TRACE\_TERMINATED(a)
- #define \_STARPU\_MPI\_TRACE\_ISEND\_COMPLETE\_END(a, b, c)
- #define \_STARPU\_MPI\_TRACE\_IRecv\_COMPLETE\_BEGIN(a, b)
- #define \_STARPU\_MPI\_TRACE\_IRecv\_COMPLETE\_END(a, b)
- #define \_STARPU\_MPI\_TRACE\_SLEEP\_BEGIN()
- #define \_STARPU\_MPI\_TRACE\_SLEEP\_END()
- #define \_STARPU\_MPI\_TRACE\_DTESTING\_BEGIN()
- #define \_STARPU\_MPI\_TRACE\_DTESTING\_END()
- #define \_STARPU\_MPI\_TRACE\_UTESTING\_BEGIN(a, b)
- #define \_STARPU\_MPI\_TRACE\_UTESTING\_END(a, b)
- #define \_STARPU\_MPI\_TRACE\_UWAIT\_BEGIN(a, b)
- #define \_STARPU\_MPI\_TRACE\_UWAIT\_END(a, b)
- #define \_STARPU\_MPI\_TRACE\_DATA\_SET\_RANK(a, b)
- #define \_STARPU\_MPI\_TRACE\_DATA\_SET\_TAG(a, b)
- #define \_STARPU\_MPI\_TRACE\_TESTING\_DETACHED\_BEGIN()
- #define \_STARPU\_MPI\_TRACE\_TESTING\_DETACHED\_END()
- #define \_STARPU\_MPI\_TRACE\_TEST\_BEGIN(peer, data\_tag)
- #define \_STARPU\_MPI\_TRACE\_TEST\_END(peer, data\_tag)
- #define \_STARPU\_MPI\_TRACE\_POLLING\_BEGIN()
- #define \_STARPU\_MPI\_TRACE\_POLLING\_END()
- #define \_STARPU\_MPI\_TRACE\_DRIVER\_RUN\_BEGIN()
- #define \_STARPU\_MPI\_TRACE\_DRIVER\_RUN\_END()
- #define \_STARPU\_MPI\_TRACE\_CHECKPOINT\_BEGIN(cp\_instance, cp\_domain)
- #define \_STARPU\_MPI\_TRACE\_CHECKPOINT\_END(cp\_instance, cp\_domain)

## Functions

- void `_starpu_mpi_fxt_init` (void \*arg)
- void `_starpu_mpi_fxt_shutdown` ()

## 7.18 `starpu_mpi_nmad.h` File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

## Functions

- int `_starpu_mpi_progress_init` (struct `_starpu_mpi_argc_argv` \*argc\_argv)
- void `_starpu_mpi_progress_shutdown` (void \*\*value)
- int `_starpu_mpi_barrier` (MPI\_Comm comm)
- int `_starpu_mpi_wait_for_all` (MPI\_Comm comm)
- int `_starpu_mpi_wait` (starpu\_mpi\_req \*public\_req, MPI\_Status \*status)
- int `_starpu_mpi_test` (starpu\_mpi\_req \*public\_req, int \*flag, MPI\_Status \*status)
- void `_starpu_mpi_isend_func` (struct `_starpu_mpi_req` \*req)
- void `_starpu_mpi_irecv_func` (struct `_starpu_mpi_req` \*req)
- void `_starpu_mpi_handle_request_termination` (struct `_starpu_mpi_req` \*req)
- void `_starpu_mpi_handle_pending_request` (struct `_starpu_mpi_req` \*req)
- void `_starpu_mpi_handle_received_data` (struct `_starpu_mpi_req` \*req)

## 7.19 `starpu_mpi_select_node.h` File Reference

```
#include <mpi.h>
```

## Macros

- `#define _STARPU_MPI_NODE_SELECTION_MAX_POLICY`

## Functions

- void `_starpu_mpi_select_node_init` ()
- int `_starpu_mpi_select_node` (int me, int nb\_nodes, struct `starpu_data_descr` \*descr, int nb\_data, int policy)

## 7.20 `starpu_mpi_task_insert.h` File Reference

## Functions

- int `_starpu_mpi_find_executtee_node` (starpu\_data\_handle\_t data, enum `starpu_data_access_mode` mode, int me, int \*do\_execute, int \*inconsistent\_execute, int \*xrank)
- int `_starpu_mpi_exchange_data_before_execution` (starpu\_data\_handle\_t data, enum `starpu_data_access_mode` mode, int me, int xrank, int do\_execute, int prio, MPI\_Comm comm)
- int `_starpu_mpi_task_postbuild_v` (MPI\_Comm comm, int xrank, int do\_execute, struct `starpu_data_descr` \*descrs, int nb\_data, int prio)
- void `_starpu_mpi_redux_wrapup_data_all` ()
- void `_starpu_mpi_redux_wrapup_data` (starpu\_data\_handle\_t data\_handle)
- void `_starpu_mpi_pre_submit_hook_call` (struct `starpu_task` \*task)

## 7.21 load\_balancer\_policy.h File Reference

```
#include <starpu_mpi_lb.h>
```

### Data Structures

- struct [load\\_balancer\\_policy](#)

### Variables

- [struct load\\_balancer\\_policy](#) [load\\_heat\\_propagation\\_policy](#)

## 7.22 load\_data\_interface.h File Reference

```
#include <starpu.h>
```

### Data Structures

- struct [load\\_data\\_interface](#)

### Macros

- #define [LOAD\\_DATA\\_GET\\_NSUBMITTED\\_TASKS](#)(interface)
- #define [LOAD\\_DATA\\_GET\\_SLEEP\\_THRESHOLD](#)(interface)
- #define [LOAD\\_DATA\\_GET\\_WAKEUP\\_THRESHOLD](#)(interface)

### Functions

- void [load\\_data\\_data\\_register](#) (starpu\_data\_handle\_t \*handle, unsigned home\_node, int sleep\_task\_↔ threshold, double wakeup\_ratio)
- int [load\\_data\\_get\\_sleep\\_threshold](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_get\\_wakeup\\_threshold](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_get\\_current\\_phase](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_get\\_nsubmitted\\_tasks](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_get\\_nfinished\\_tasks](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_inc\\_nsubmitted\\_tasks](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_inc\\_nfinished\\_tasks](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_next\\_phase](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_update\\_elapsed\\_time](#) (starpu\_data\_handle\_t handle)
- double [load\\_data\\_get\\_elapsed\\_time](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_update\\_wakeup\\_cond](#) (starpu\_data\_handle\_t handle)
- int [load\\_data\\_wakeup\\_cond](#) (starpu\_data\_handle\_t handle)

### 7.22.1 Data Structure Documentation

#### 7.22.1.1 struct load\_data\_interface

interface for load\_data

##### Data Fields

double	start	Starting time of the execution
double	elapsed_time	Elapsed time until the start time and the time when event "launch a load balancing phase" is triggered

## Data Fields

int	phase	Current submission phase, i.e how many balanced steps have already happened so far.
int	nsubmitted_tasks	Number of currently submitted tasks
int	nfinished_tasks	Number of currently finished tasks
int	sleep_task_threshold	Task threshold to sleep the submission thread
int	wakeup_task_threshold	Task threshold to wake-up the submission thread
double	wakeup_ratio	Ratio of submitted tasks to wait for completion before waking up the submission thread

## 7.23 data\_movements\_interface.h File Reference

```
#include <starpu.h>
```

### Data Structures

- struct [data\\_movements\\_interface](#)

### Macros

- #define **DATA\_MOVEMENTS\_GET\_SIZE\_TABLES**(interface)
- #define **DATA\_MOVEMENTS\_GET\_TAGS\_TABLE**(interface)
- #define **DATA\_MOVEMENTS\_GET\_RANKS\_TABLE**(interface)

### Functions

- void **data\_movements\_data\_register** (starpu\_data\_handle\_t \*handle, unsigned home\_node, int \*ranks, starpu\_mpi\_tag\_t \*tags, int size)
- starpu\_mpi\_tag\_t \*\* **data\_movements\_get\_ref\_tags\_table** (starpu\_data\_handle\_t handle)
- int \*\* **data\_movements\_get\_ref\_ranks\_table** (starpu\_data\_handle\_t handle)
- int **data\_movements\_reallocate\_tables** (starpu\_data\_handle\_t handle, unsigned node, int size)
- starpu\_mpi\_tag\_t \* **data\_movements\_get\_tags\_table** (starpu\_data\_handle\_t handle)
- int \* **data\_movements\_get\_ranks\_table** (starpu\_data\_handle\_t handle)
- int **data\_movements\_get\_size\_tables** (starpu\_data\_handle\_t handle)

### 7.23.1 Data Structure Documentation

#### 7.23.1.1 struct data\_movements\_interface

interface for data\_movements

## Data Fields

starpu_mpi_tag_t *	tags	Data tags table
int *	ranks	Ranks table (where to move the corresponding data)
int	size	Size of the tables

## Chapter 8

# StarPU Resource Manager File Documentation

## 8.1 starpurm\_private.h File Reference

### Data Structures

- struct [s\\_starpurm](#)

### Enumerations

- enum **e\_state** { **state\_uninitialized** , **state\_init** }
- enum **e\_starpurm\_unit\_type** { **starpurm\_unit\_cpu** , **starpurm\_unit\_opencil** , **starpurm\_unit\_cuda** , **starpurm\_unit\_ntypes** }

### 8.1.1 Data Structure Documentation

#### 8.1.1.1 struct s\_starpurm

##### Data Fields

hwloc_topology_t	topology	Machine topology as detected by hwloc.
int	max_ncpus	Current upper bound on the number of CPU cores selectable for computing with the runtime system.
int	selected_ncpus	Number of currently selected CPU workers
int	selected_nworkers	Number of currently selected workers (CPU+devices)
int	state	Initialization state of the RM instance.
int	dynamic_resource_sharing	Boolean indicating the state of the dynamic resource sharing layer. !0 indicates that dynamic resource sharing is enabled. 0 indicates that dynamic resource sharing is disabled.
unsigned	sched_ctx_id	Id of the StarPU's sched_ctx used by the RM instance.
int	unit_ntypes	Number of unit types supported by this RM instance.
int *	nunits_by_type	Number of units available for each type.
int	nunits	Number of units.
int *	unit_offsets_by_type	Offset of unit numbering for each type.
<a href="#">struct s_starpurm_unit *</a>	units	Array of units.
hwloc_cpuset_t	global_cpuset	Cpuset of all the StarPU's workers (CPU+devices).

## Data Fields

hwloc_cpuset_t	all_cpu_workers_cpuset	Cpuset of all StarPU CPU workers.
hwloc_cpuset_t	all_opencl_device_workers_cpuset	Cpuset of all StarPU OpenCL workers.
hwloc_cpuset_t	all_cuda_device_workers_cpuset	Cpuset of all StarPU CUDA workers.
hwloc_cpuset_t	all_device_workers_cpuset	Cpuset of all StarPU device workers.
hwloc_cpuset_t	selected_cpuset	Cpuset of all selected workers (CPU+devices).
hwloc_cpuset_t	initially_owned_cpuset_mask	Cpuset mask of initially owned cpuset or full if not used.
int	max_worker_id	maximum value among worker ids
int *	worker_unit_ids	worker id to unit id table
unsigned int	max_temporary_ctxs	Temporary contexts accounting.
unsigned int	avail_temporary_ctxs	
starpu_pthread_mutex_t	temporary_ctxs_mutex	
starpu_pthread_cond_t	temporary_ctxs_cond	
int	starpu_in_pause	Global StarPU pause state
pthread_t	event_thread	Event list.
starpu_pthread_mutex_t	event_list_mutex	
starpu_pthread_cond_t	event_list_cond	
starpu_pthread_cond_t	event_processing_cond	
int	event_processing_enabled	
int	event_processing_ended	
struct s_starpurm_event *	event_list_head	
struct s_starpurm_event *	event_list_tail	