StarPU Handbook - StarPU Installation

for StarPU 1.4.7

This manual documents the usage of StarPU version 1.4.7. Its contents was last updated on 2024-05-15.

Chapter 1

Organization

This parts shows a basic usage of StarPU and how to execute the provided examples or your own applications.

- Chapter Building and Installing StarPU shows how to build and install StarPU.

- Chapter Compilation Configuration shows how to tune StarPU building process through configuration options.

- Chapter Execution Configuration Through Environment Variables lists environment variables that can be used to tune StarPU when executing an application.

Finally, Chapter Configuration and Initialization shows a brief overview of how to configure and tune StarPU.

Chapter 2

Building and Installing StarPU

Depending on the level of customization required for the library installation, we offer several solutions.

1. **Basic Installation or Evaluation:** If you are looking to simply try out the library, assess its performance on simple cases, run examples, or use the latest stable version, we recommend the following options:

   - For Linux Debian or Ubuntu distributions, consider using the latest StarPU Debian package (see Installing a Binary Package).
   - For macOS, you can opt for Brew and follow the steps in Installing a Source Package.
   - Using an already installed module on a cluster, as explained in Using a Module

2. **Customization for Specific Needs:** If you intend to use StarPU but require modifications, such as switching to another version (git branch), changing the default MPI, utilizing a preferred compiler, or altering source code, consider these options:

   - Guix or Spack can be useful, as these package managers allow dynamic changes during source-based builds. Refer to Installing a Source Package for details.
   - Alternatively, you can directly build from the source using the native build system of the library (Makefile, GNU autotools). Instructions can be found in Building from Source.

3. **Experiment Reproducibility:** If your focus is on experiment reproducibility, we recommend using Guix. Refer to Installing a Source Package for guidance.

Whichever solution you choose, you can utilize the tool `bin/starpu_config` to view all the configuration parameters used during StarPU installation.

Please refer to the provided documentation for specific installation steps and details for each solution.

2.1 Installing a Binary Package

One of the StarPU developers being a Debian Developer, the packages are well integrated and very up-to-date. To see which packages are available, simply type:

```
$ apt-cache search starpu
```

To install what you need, type for example:

```
$ sudo apt-get install libstarpu-dev
```

2.2 Installing a Source Package

StarPU is available from different package managers.

- Guix `https://gitlab.inria.fr/guix-hpc/guix-hpc`

- Spack `https://github.com/spack/spack/`

- Brew `https://gitlab.inria.fr/solverstack/brew-repo`

Documentation on how to install StarPU with these package managers is directly available from the links specified above. We give below a brief overview of the spack installation.

2.2.1 Installing the Spack Package

Here is a quick guide to install StarPU with spack.

```
$ git clone git@github.com:spack/spack.git
$ source ./spack/share/spack/setup-env.sh # if you use bash or zsh
$ spack install starpu
```

By default, the latest release will be installed, one can choose to install a specific release or even the master version.

```
$ spack install starpu@master
$ spack install starpu@1.3.5
```

We strongly advise reading the detailed reference manual at https://spack.readthedocs.↵io/en/latest/getting_started.html

2.2.2 Using a Module

On some clusters, StarPU is provided as a module, for example on the Jean Zay cluster. The information is available at http://www.idris.fr/jean-zay/cpu/jean-zay-cpu-starpu.html

2.3 Building from Source

StarPU can be built and installed by the standard means of the GNU autotools. The following chapter is intended to briefly remind how these tools can be used to install StarPU.

2.3.1 Optional Dependencies

The `hwloc` (http://www.open-mpi.org/software/hwloc) topology discovery library is not mandatory to use StarPU, but strongly recommended. It allows for topology aware scheduling, which improves performance. `hwloc` is available in major free operating system distributions, and for most operating systems. Make sure to not only install a `hwloc` or `libhwloc` package, but also `hwloc-devel` or `libhwloc-dev` to have `hwloc` headers etc.

If `libhwloc` is installed in a standard location, no option is required, it will be detected automatically, otherwise --with-hwloc=<directory> should be used to specify its location.

If `libhwloc` is not available on your system, the option --without-hwloc should be explicitly given when calling the script `configure`.

2.3.2 Getting Sources

StarPU's sources can be obtained from the download page of the StarPU website (https://starpu.↵gitlabpages.inria.fr/files/).

All releases and the development tree of StarPU are freely available on StarPU SCM server under the LGPL license. Some releases are available under the BSD license.

The latest release can be downloaded from the StarPU download page (https://starpu.gitlabpages.↵inria.fr/files/).

The latest nightly snapshot can be downloaded from the StarPU website (https://starpu.↵gitlabpages.inria.fr/files/testing/).

And finally, the current development version is also accessible via git. It should only be used if you need the very latest changes (i.e. less than a day old!).

```
$ git clone git@gitlab.inria.fr:starpu/starpu.git
```

2.3.3 Configuring StarPU

Running `autogen.sh` is not necessary when using the tarball releases of StarPU. However, when using the source code from the git repository, you first need to generate the script `configure` and the different Makefiles. This requires the availability of `autoconf` and `automake` >= 2.60.

```
$ ./autogen.sh
```

You then need to configure StarPU. Details about options that are useful to give to `configure` are given in Compilation Configuration.

```
$ ./configure
```

If `configure` does not detect some software or produces errors, please make sure to post the contents of the file `config.log` when reporting the issue.

By default, the files produced during the compilation are placed in the source directory. As the compilation generates a lot of files, it is advised to put them all in a separate directory. It is then easier to clean up, and this allows to compile several configurations out of the same source tree. To do so, simply enter the directory where you want the compilation to produce its files, and invoke the script `configure` located in the StarPU source directory.

```
$ mkdir build
$ cd build
$ ../configure
```

By default, StarPU will be installed in `/usr/local/bin`, `/usr/local/lib`, etc. You can specify an installation prefix other than `/usr/local` using the option `-prefix`, for instance:

```
$ ../configure --prefix=$HOME/starpu
```

2.3.4 Building StarPU

```
$ make
```

Once everything is built, you may want to test the result. An extensive set of regression tests is provided with Star↩PU. Running the tests is done by calling `make check`. These tests are run every night and the result from the main profile is publicly available (https://starpu.gitlabpages/files/testing/master/).

```
$ make check
```

2.3.5 Installing StarPU

In order to install StarPU at the location which was specified during configuration:

```
$ make install
```

If you have let StarPU install in `/usr/local/`, you additionally need to run

```
$ sudo ldconfig
```

so the libraries can be found by the system.

Libtool interface versioning information are included in libraries names (`libstarpu-1.4.so`, `libstarpumpi-1.↩4.so` and `libstarpufft-1.4.so`).

Chapter 3

Compilation Configuration

The behavior of the StarPU library and tools may be tuned thanks to the following configure options.

3.1   Common Configuration

–enable-debug   Enable debugging messages.

–enable-spinlock-check   Enable checking that spinlocks are taken and released properly.

–enable-fast   Disable assertion checks, which saves computation time.

–enable-verbose   Increase the verbosity of the debugging messages. This can be disabled at runtime by setting the environment variable STARPU_SILENT to any value. `-enable-verbose=extra` increase even more the verbosity.

```
$ STARPU_SILENT=1 ./vector_scal
```

–enable-coverage   Enable flags for the coverage tool `gcov`.

–enable-quick-check   Specify tests and examples should be run on a smaller data set, i.e allowing a faster execution time

–enable-long-check   Enable some exhaustive checks which take a really long time.

–enable-new-check   Enable new testcases which are known to fail.

–with-hwloc   Specify `hwloc` should be used by StarPU. `hwloc` should be found by the means of the tool `pkg-config`.

–with-hwloc=`prefix`   Specify `hwloc` should be used by StarPU. `hwloc` should be found in the directory specified by `prefix`

–without-hwloc   Specify `hwloc` should not be used by StarPU.

–disable-build-doc   Disable the creation of the documentation.   This should be done on a machine which does not have the tools `doxygen` and `latex` (plus the packages `latex-xcolor` and `texlive-latex-extra`).

–enable-build-doc-pdf   By default, only the HTML documentation is generated. Use this option to also enable the generation of the PDF documentation. This should be done on a machine which does have the tools `doxygen` and `latex` (plus the packages `latex-xcolor` and `texlive-latex-extra`).

–enable-icc   Enable the compilation of specific ICC examples. StarPU itself will not be compiled with ICC unless specified with `CC=icc`

–disable-icc   Disable the usage of the ICC compiler. Otherwise, when a ICC compiler is found, some specific ICC examples are compiled as explained above.

–with-check-flags   Specify flags which will be given to C, CXX and Fortran compilers when valid

Additionally, the script `configure` recognize many variables, which can be listed by typing `./configure -help`. For example, `./configure NVCCFLAGS="-arch sm_20"` adds a flag for the compilation of CUDA kernels, and `NVCC_CC=gcc-5` allows to change the C++ compiler used by nvcc.

3.2 Configuring Workers

–enable-data-locality-enforce   Enable data locality enforcement when picking up a worker to execute a task. This mechanism is by default disabled.

–enable-blocking-drivers   By default, StarPU keeps CPU workers awake permanently, for better reactivity. This option makes StarPU put CPU workers to real sleep when there are not enough tasks to compute.

–enable-worker-callbacks   If blocking drivers are enabled, enable callbacks to notify an external resource manager about workers going to sleep and waking up.

–enable-maxcpus=count   Use at most `count` CPU cores. This information is then available as the macro STARPU_MAXCPUS.

The default value is `auto`. it allows StarPU to automatically detect the number of CPUs on the build machine. This should not be used if the running host has a larger number of CPUs than the build machine.

–enable-maxnumanodes=count   Use at most `count` NUMA nodes. This information is then available as the macro STARPU_MAXNUMANODES.

The default value is `auto`. it allows StarPU to automatically detect the number of NUMA nodes on the build machine. This should not be used if the running host has a larger number of NUMA nodes than the build machine.

–disable-cpu   Disable the use of CPUs of the machine. Only GPUs etc. will be used.

–enable-maxcudadev=count   Use at most `count` CUDA devices. This information is then available as the macro STARPU_MAXCUDADEVS.

–disable-cuda   Disable the use of CUDA, even if a valid CUDA installation was detected.

–with-cuda-dir=prefix   Search for CUDA under `prefix`, which should notably contain the file `include/cuda.↩ h`.

–with-cuda-include-dir=dir   Search for CUDA headers under `dir`, which should notably contain the file `cuda.h`. This defaults to `/include` appended to the value given to --with-cuda-dir.

–with-cuda-lib-dir=dir   Search for CUDA libraries under `dir`, which should notably contain the CUDA shared libraries—e.g., `libcuda.so`. This defaults to `/lib` appended to the value given to --with-cuda-dir.

–disable-cuda-memcpy-peer   Explicitly disable peer transfers when using CUDA 4.0.

–enable-maxopencldev=count   Use at most `count` OpenCL devices. This information is then available as the macro STARPU_MAXOPENCLDEVS.

–disable-opencl   Disable the use of OpenCL, even if the SDK is detected.

–with-opencl-dir=prefix   Search for an OpenCL implementation under `prefix`, which should notably contain `include/CL/cl.h` (or `include/OpenCL/cl.h` on Mac OS).

–with-opencl-include-dir=dir   Search for OpenCL headers under `dir`, which should notably contain `CL/cl.↩ h` (or `OpenCL/cl.h` on Mac OS). This defaults to `/include` appended to the value given to --with-opencl-dir.

–with-opencl-lib-dir=dir   Search for an OpenCL library under `dir`, which should notably contain the Open↩ CL shared libraries—e.g. `libOpenCL.so`. This defaults to `/lib` appended to the value given to --with-opencl-dir.

–enable-opencl-simulator   Enable considering the provided OpenCL implementation as a simulator, i.e. use the kernel duration returned by OpenCL profiling information as wallclock time instead of the actual measured real time. This requires the SimGrid support.

–enable-maximplementations=count   Allow for at most `count` codelet implementations for the same target device. This information is then available as the macro STARPU_MAXIMPLEMENTATIONS macro.

–enable-max-sched-ctxs=count   Allow for at most `count` scheduling contexts This information is then available as the macro STARPU_NMAX_SCHED_CTXS.

–disable-asynchronous-copy   Disable asynchronous copies between CPU and GPU devices. The AMD implementation of OpenCL is known to fail when copying data asynchronously. When using this implementation, it is therefore necessary to disable asynchronous data transfers.

–disable-asynchronous-cuda-copy   Disable asynchronous copies between CPU and CUDA devices.

–disable-asynchronous-opencl-copy   Disable asynchronous copies between CPU and OpenCL devices. The AMD implementation of OpenCL is known to fail when copying data asynchronously. When using this implementation, it is therefore necessary to disable asynchronous data transfers.

–disable-asynchronous-hip-copy   Disable asynchronous copies between CPU and HIP devices.

–disable-asynchronous-mpi-master-slave-copy   Disable asynchronous copies between CPU and MPI Slave devices.

–disable-asynchronous-tcpip-master-slave-copy   Disable asynchronous copies between CPU and MPI Slave devices.

–disable-asynchronous-fpga-copy   Disable asynchronous copies between CPU and Maxeler FPGA devices.

–enable-maxnodes=`count`   Use at most `count` memory nodes. This information is then available as the macro STARPU_MAXNODES. Reducing it allows to considerably reduce memory used by StarPU data structures.

–with-max-fpga=`dir`   Enable the Maxeler FPGA driver support, and optionally specify the location of the Maxeler FPGA library.

–disable-asynchronous-max-fpga-copy   Disable asynchronous copies between CPU and Maxeler FPGA devices.

3.3   Extension Configuration

–enable-starpupy   Enable the StarPU Python Interface (PythonInterface)

–enable-python-multi-interpreter   Enable the use of multiple interpreters in the StarPU Python Interface (MultipleInterpreters)

–disable-mpi   Disable the build of libstarpumpi. By default, it is enabled when MPI is found.

–enable-mpi   Enable the build of libstarpumpi. This is necessary when using Simgrid+MPI.

–with-mpicc=`path`   Use the compiler `mpicc` at `path`, for StarPU-MPI. (MPISupport).

–enable-mpi-pedantic-isend   Before performing any MPI communication, StarPU-MPI waits for the data to be available in the main memory of the node submitting the request. For send communications, data is acquired with the mode STARPU_R. When enabling the pedantic mode, data are instead acquired with the STARPU_RW which thus ensures that there is not more than 1 concurrent `MPI_Isend` calls accessing the data and StarPU does not read from it from tasks during the communication.

–enable-mpi-master-slave   Enable the MPI Master-Slave support. By default, it is disabled.

–enable-mpi-verbose   Increase the verbosity of the MPI debugging messages. This can be disabled at runtime by setting the environment variable STARPU_SILENT to any value. `-enable-mpi-verbose=extra` increase even more the verbosity.

        $ STARPU_SILENT=1 mpirun -np 2 ./insert_task

–enable-mpi-ft   Enable the MPI checkpoint mechanism. See MPI Fault Tolerance Support

–enable-mpi-ft-stats   Enable the statistics for the MPI checkpoint mechanism. See MPI Fault Tolerance Support

–enable-tcpip-master-slave   Enable the TCP/IP Master-Slave support (TCPIPSupport). By default, it is disabled.

–enable-nmad   Enable the NewMadeleine implementation for StarPU-MPI. See Nmad for more details.

–disable-fortran   Disable the fortran extension. By default, it is enabled when a fortran compiler is found.

–disable-socl   Disable the SOCL extension (SOCLOpenclExtensions). By default, it is enabled when an OpenCL implementation is found.

–enable-openmp   Enable OpenMP Support (OpenMPRuntimeSupport)

–enable-openmp-llvm   Enable LLVM OpenMP Support (OMPLLVM)

–enable-bubble   Enable Hierarchical dags support (HierarchicalDAGS)

–enable-parallel-worker   Enable parallel worker support (ParallelWorker)

–enable-eclipse-plugin   Enable the StarPU Eclipse Plugin. See EclipsePlugin to know how to install Eclipse.

3.4   Advanced Configuration

–enable-perf-debug   Enable performance debugging through gprof.

–enable-model-debug   Enable performance model debugging.

–enable-fxt-lock   Enable additional trace events which describes locks behaviour. This is however extremely heavy and should only be enabled when debugging insides of StarPU.

–enable-maxbuffers   Define the maximum number of buffers that tasks will be able to take as parameters, then available as the macro STARPU_NMAXBUFS.

–enable-fxt-max-files=`count`   Use at most `count` mpi nodes fxt files for generating traces. This information is then available as the macro STARPU_FXT_MAX_FILES. This information is used by FxT tools when considering multi node traces. Default value is 64.

–enable-allocation-cache   Enable the use of a data allocation cache to avoid the cost of it with CUDA. Still experimental.

–enable-opengl-render   Enable the use of OpenGL for the rendering of some examples.

–enable-blas-lib=`prefix`   Specify the blas library to be used by some of the examples. Libraries available :

- `none` [default] : no BLAS library is used
- `atlas:` use ATLAS library
- `goto:` use GotoBLAS library
- `openblas:` use OpenBLAS library
- `mkl:` use MKL library (you may need to set specific `CFLAGS` and `LDFLAGS` with –with-mkl-cflags and –with-mkl-ldflags)

–enable-leveldb   Enable linking with LevelDB if available

–enable-hdf5   Enable building HDF5 support.

–with-hdf5-include-dir=`path`   Specify the directory where is stored the header file `hdf5.h`.

–with-hdf5-lib-dir=`path`   Specify the directory where is stored the library `hdf5`.

–disable-starpufft   Disable the build of libstarpufft, even if `fftw` or `cuFFT` is available.

–enable-starpufft-examples   Enable the compilation and the execution of the libstarpufft examples. By default, they are neither compiled nor checked.

–with-fxt=`prefix`   Search for FxT under `prefix`. FxT (http://savannah.nongnu.org/projects/fkt) is used to generate traces of scheduling events, which can then be rendered them using ViTE (Off-line↩ PerformanceFeedback). `prefix` should notably contain `include/fxt/fxt.h`.

–with-perf-model-dir=`dir`   Store performance models under `dir`, instead of the current user's home.

–with-goto-dir=`prefix`   Search for GotoBLAS under `prefix`, which should notably contain `libgoto.so` or `libgoto2.so`.

–with-atlas-dir=`prefix` Search for ATLAS under `prefix`, which should notably contain `include/cblas.h`.

–with-mkl-cflags=`cflags` Use `cflags` to compile code that uses the MKL library.

–with-mkl-ldflags=`ldflags` Use `ldflags` when linking code that uses the MKL library. Note that the MKL website (`http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/`) provides a script to determine the linking flags.

–disable-glpk Disable the use of `libglpk` for computing area bounds.

–disable-build-tests Disable the build of tests.

–disable-build-examples Disable the build of examples.

–enable-sc-hypervisor Enable the Scheduling Context Hypervisor plugin (SchedulingContextHypervisor). By default, it is disabled.

–enable-memory-stats Enable memory statistics (MemoryFeedback).

–enable-simgrid Enable simulation of execution in SimGrid, to allow easy experimentation with various numbers of cores and GPUs, or amount of memory, etc. Experimental.

The path to SimGrid can be specified through the `SIMGRID_CFLAGS` and `SIMGRID_LIBS` environment variables, for instance:

```
export SIMGRID_CFLAGS="-I/usr/local/simgrid/include"
export SIMGRID_LIBS="-L/usr/local/simgrid/lib -lsimgrid"
```

–with-simgrid-dir Similar to the option --enable-simgrid but also allows to specify the location to the SimGrid library.

–with-simgrid-include-dir Similar to the option --enable-simgrid but also allows to specify the location to the SimGrid include directory.

–with-simgrid-lib-dir Similar to the option --enable-simgrid but also allows to specify the location to the SimGrid lib directory.

–with-smpirun=`path` Use the smpirun at `path`

–enable-simgrid-mc Enable the Model Checker in simulation of execution in SimGrid, to allow exploring various execution paths.

–enable-calibration-heuristic Allow to set the maximum authorized percentage of deviation for the history-based calibrator of StarPU. A correct value of this parameter must be in [0..100]. The default value of this parameter is 10. Experimental.

–enable-mlr Allow to enable multiple linear regression models (see PerformanceModelExample)

–enable-mlr-system-blas Allow to make multiple linear regression models use the system-provided BLAS for dgels (see PerformanceModelExample)

Chapter 4

Execution Configuration Through Environment Variables

The StarPU library and tools's behavior can be tuned using the following environment variables. To access these variables, you can use the provided functions.

- starpu_getenv() retrieves the value of an environment variable.

- starpu_get_env_string_var_default() retrieves the value of an environment variable as a string. If the variable is not set, you can provide a default value.

- starpu_get_env_size_default() retrieves the value of an environment variable as a size in bytes, or a default value if the environment variable is not set.

These functions allow to fine-tune the behavior of StarPU according to your preferences and requirements by leveraging environment variables.

4.1 Configuring Workers

4.1.1 General Configuration

STARPU_WORKERS_NOBIND Setting it to non-zero will prevent StarPU from binding its threads to CPUs. This is for instance useful when running the test suite in parallel.

STARPU_WORKERS_GETBIND By default StarPU uses the OS-provided CPU binding to determine how many and which CPU cores it should use. This is notably useful when running several StarPU-MPI processes on the same host, to let the MPI launcher set the CPUs to be used. Default value is 1.

If that binding is erroneous (e.g. because the job scheduler binds to just one core of the allocated cores), you can set STARPU_WORKERS_GETBIND to 0 to make StarPU use all cores of the machine.

STARPU_WORKERS_CPUID Passing an array of integers in STARPU_WORKERS_CPUID specifies on which logical CPU the different workers should be bound. For instance, if STARPU_WORKERS_CPUID="0 1 4 5", the first worker will be bound to logical CPU #0, the second CPU worker will be bound to logical CPU #1 and so on. Note that the logical ordering of the CPUs is either determined by the OS, or provided by the library `hwloc` in case it is available. Ranges can be provided: for instance, STARPU_WORKERS_CPUID="1-3 5" will bind the first three workers on logical CPUs #1, #2, and #3, and the fourth worker on logical CPU #5. Unbound ranges can also be provided: STARPU_WORKERS_CPUID="1-" will bind the workers starting from logical CPU #1 up to last CPU.

Note that the first workers correspond to the CUDA workers, then come the OpenCL workers, and finally the CPU workers. For example, if we have STARPU_NCUDA=1, STARPU_NOPENCL=1, STARPU_NCPU=2 and STARPU_WORKERS_CPUID="0 2 1 3", the CUDA device will be controlled by logical CPU #0, the OpenCL device will be controlled by logical CPU #2, and the logical CPUs #1 and #3 will be used by the CPU workers.

If the number of workers is larger than the array given in STARPU_WORKERS_CPUID, the workers are bound to the logical CPUs in a round-robin fashion: if STARPU_WORKERS_CPUID="0 1", the first and the third (resp. second and fourth) workers will be put on CPU #0 (resp. CPU #1).

This variable is ignored if the field starpu_conf::use_explicit_workers_bindid passed to starpu_init() is set.

Setting STARPU_WORKERS_CPUID or STARPU_WORKERS_COREID overrides the binding provided by the job scheduler, as described for STARPU_WORKERS_GETBIND.

STARPU_WORKERS_COREID Same as STARPU_WORKERS_CPUID, but bind the workers to cores instead of PUs (hyperthreads).

STARPU_NTHREADS_PER_CORE Specify how many threads StarPU should run on each core. The default is 1 because kernels are usually already optimized for using a full core. Setting this to e.g. 2 instead allows exploiting hyperthreading.

STARPU_MAIN_THREAD_BIND Tell StarPU to bind the thread that calls starpu_initialize() to a reserved CPU, subtracted from the CPU workers.

STARPU_MAIN_THREAD_CPUID Tell StarPU to bind the thread that calls starpu_initialize() to the given CPU ID (using logical numbering).

STARPU_MAIN_THREAD_COREID Same as STARPU_MAIN_THREAD_CPUID, but bind the thread that calls starpu_initialize() to the given core (using logical numbering), instead of the PU (hyperthread).

STARPU_WORKER_TREE Define to 1 to enable the tree iterator in schedulers.

STARPU_SINGLE_COMBINED_WORKER Tell StarPU to create several workers which won't be able to work concurrently. It will by default create combined workers, which size goes from 1 to the total number of CPU workers in the system. STARPU_MIN_WORKERSIZE and STARPU_MAX_WORKERSIZE can be used to change this default.

STARPU_MIN_WORKERSIZE Specify the minimum size of the combined workers. Default value is 2.

STARPU_MAX_WORKERSIZE Specify the minimum size of the combined workers. Default value is the number of CPU workers in the system.

STARPU_SYNTHESIZE_ARITY_COMBINED_WORKER Specify how many elements are allowed between combined workers created from `hwloc` information. For instance, in the case of sockets with 6 cores without shared L2 caches, if STARPU_SYNTHESIZE_ARITY_COMBINED_WORKER is set to 6, no combined worker will be synthesized beyond one for the socket and one per core. If it is set to 3, 3 intermediate combined workers will be synthesized, to divide the socket cores into 3 chunks of 2 cores. If it set to 2, 2 intermediate combined workers will be synthesized, to divide the socket cores into 2 chunks of 3 cores, and then 3 additional combined workers will be synthesized, to divide the former synthesized workers into a bunch of 2 cores, and the remaining core (for which no combined worker is synthesized since there is already a normal worker for it).

Default value is 2, thus makes StarPU tend to build binary trees of combined workers.

STARPU_DISABLE_ASYNCHRONOUS_COPY Disable asynchronous copies between CPU and GPU devices. The AMD implementation of OpenCL is known to fail when copying data asynchronously. When using this implementation, it is therefore necessary to disable asynchronous data transfers. One can call starpu_asynchronous_copy_disabled() to check whether asynchronous data transfers between CPU and accelerators are disabled.

See also STARPU_DISABLE_ASYNCHRONOUS_CUDA_COPY and STARPU_DISABLE_ASYNCHRONOUS_OPENCL_COP

STARPU_EXPECTED_TRANSFER_TIME_WRITEBACK Set to 1 to make task transfer time estimations artificially include the time that will be needed to write back data to the main memory.

STARPU_DISABLE_PINNING Disable (1) or Enable (0) pinning host memory allocated through starpu_malloc(), starpu_memory_pin() and friends. Default value is Enable. This permits to test the performance effect of memory pinning.

STARPU_BACKOFF_MIN Set minimum exponential backoff of number of cycles to pause when spinning. Default value is 1.

STARPU_BACKOFF_MAX Set maximum exponential backoff of number of cycles to pause when spinning. Default value is 32.

STARPU_SINK Defined internally by StarPU when running in master slave mode.

STARPU_ENABLE_MAP Disable (0) or Enable (1) support for memory mapping between memory nodes. The default is Disabled. One can call starpu_map_enabled() to check whether memory mapping support between memory nodes is enabled.

STARPU_DATA_LOCALITY_ENFORCE Enable (1) or Disable(0) data locality enforcement when picking up a worker to execute a task. Default value is Disable.

4.1.2 CPU Workers

STARPU_NCPU Specify the number of CPU workers (thus not including workers dedicated to control accelerators). Note that by default, StarPU will not allocate more CPU workers than there are physical CPUs, and that some CPUs are used to control the accelerators.

STARPU_RESERVE_NCPU Specify the number of CPU cores that should not be used by StarPU, so the application can use starpu_get_next_bindid() and starpu_bind_thread_on() to bind its own threads.

This option is ignored if STARPU_NCPU or starpu_conf::ncpus is set.

STARPU_NCPUS Deprecated. You should use STARPU_NCPU.

4.1.3 CUDA Workers

STARPU_NCUDA Specify the number of CUDA devices that StarPU can use. If STARPU_NCUDA is lower than the number of physical devices, it is possible to select which GPU devices should be used by the means of the environment variable STARPU_WORKERS_CUDAID. By default, StarPU will create as many CUDA workers as there are GPU devices.

STARPU_NWORKER_PER_CUDA Specify the number of workers per CUDA device, and thus the number of kernels which will be concurrently running on the devices, i.e. the number of CUDA streams. Default value is 1.

STARPU_CUDA_THREAD_PER_WORKER Specify whether the cuda driver should use one thread per stream (1) or to use a single thread to drive all the streams of the device or all devices (0), and STARPU_CUDA_THREAD_PER_DEV determines whether is it one thread per device or one thread for all devices. Default value is 0. Setting it to 1 is contradictory with setting STARPU_CUDA_THREAD_PER_DEV.

STARPU_CUDA_THREAD_PER_DEV Specify whether the cuda driver should use one thread per device (1) or to use a single thread to drive all the devices (0). Default value is 1. It does not make sense to set this variable if STARPU_CUDA_THREAD_PER_WORKER is set to to 1 (since STARPU_CUDA_THREAD_PER_DEV is then meaningless).

STARPU_CUDA_PIPELINE Specify how many asynchronous tasks are submitted in advance on CUDA devices. This for instance permits to overlap task management with the execution of previous tasks, but it also allows concurrent execution on Fermi cards, which otherwise bring spurious synchronizations. Default value is 2. Setting the value to 0 forces a synchronous execution of all tasks.

STARPU_WORKERS_CUDAID Select which CUDA devices should be used to run CUDA workers (similarly to the STARPU_WORKERS_CPUID environment variable). On a machine equipped with 4 GPUs, setting STARPU_WORKERS_CUDAID="1 3" and STARPU_NCUDA=2 specifies that 2 CUDA workers should be created, and that they should use CUDA devices #1 and #3 (the logical ordering of the devices is the one reported by CUDA).

This variable is ignored if the field starpu_conf::use_explicit_workers_cuda_gpuid passed to starpu_init() is set.

STARPU_DISABLE_ASYNCHRONOUS_CUDA_COPY Disable asynchronous copies between CPU and CUDA devices. One can call starpu_asynchronous_cuda_copy_disabled() to check whether asynchronous data transfers between CPU and CUDA accelerators are disabled.

See also STARPU_DISABLE_ASYNCHRONOUS_COPY and STARPU_DISABLE_ASYNCHRONOUS_OPENCL_COPY.

STARPU_ENABLE_CUDA_GPU_GPU_DIRECT   Enable (1) or Disable (0) direct CUDA transfers from GPU to GPU, without copying through RAM. Default value is Enable. This permits to test the performance effect of GPU-Direct.

STARPU_CUDA_ONLY_FAST_ALLOC_OTHER_MEMNODES   Specify if CUDA workers should do only fast allocations when running the datawizard progress of other memory nodes. This will pass the internal value _STARPU_DATAWIZARD_ONLY_FAST_ALLOC to allocation methods. Default value is 0, allowing CUDA workers to do slow allocations.

> This can also be specified with starpu_conf::cuda_only_fast_alloc_other_memnodes.

4.1.4   OpenCL Workers

STARPU_NOPENCL   Specify the number of OpenCL devices that StarPU can use. If STARPU_NOPENCL is lower than the number of physical devices, it is possible to select which GPU devices should be used by the means of the environment variable STARPU_WORKERS_OPENCLID. By default, StarPU will create as many OpenCL workers as there are GPU devices.

> Note that by default StarPU will launch CUDA workers on GPU devices. You need to disable CUDA to allow the creation of OpenCL workers.

STARPU_WORKERS_OPENCLID   Select which GPU devices should be used to run OpenCL workers (similarly to the STARPU_WORKERS_CPUID environment variable) On a machine equipped with 4 GPUs, setting `STARPU_WORKERS_OPENCLID="1 3"` and `STARPU_NOPENCL=2` specifies that 2 OpenCL workers should be created, and that they should use GPU devices #1 and #3.

> This variable is ignored if the field starpu_conf::use_explicit_workers_opencl_gpuid passed to starpu_init() is set.

STARPU_OPENCL_PIPELINE   Specify how many asynchronous tasks are submitted in advance on OpenCL devices. This for instance permits to overlap task management with the execution of previous tasks, but it also allows concurrent execution on Fermi cards, which otherwise bring spurious synchronizations. Default value is 2. Setting the value to 0 forces a synchronous execution of all tasks.

STARPU_OPENCL_ON_CPUS   Specify that OpenCL workers can also be run on CPU devices. By default, the OpenCL driver only enables GPU devices.

STARPU_OPENCL_ONLY_ON_CPUS   Specify that OpenCL workers can ONLY be run on CPU devices. By default, the OpenCL driver enables GPU devices.

STARPU_DISABLE_ASYNCHRONOUS_OPENCL_COPY   Disable asynchronous copies between CPU and OpenCL devices. The AMD implementation of OpenCL is known to fail when copying data asynchronously. When using this implementation, it is therefore necessary to disable asynchronous data transfers. One can call starpu_asynchronous_opencl_copy_disabled() to check whether asynchronous data transfers between CPU and OpenCL accelerators are disabled.

> See also STARPU_DISABLE_ASYNCHRONOUS_COPY and STARPU_DISABLE_ASYNCHRONOUS_CUDA_COPY.

4.1.5   Maxeler FPGA Workers

STARPU_NMAX_FPGA   Specify the number of Maxeler FPGA devices that StarPU can use. If STARPU_NMAX_FPGA is lower than the number of physical devices, it is possible to select which Maxeler FPGA devices should be used by the means of the environment variable STARPU_WORKERS_MAX_FPGAID. By default, StarPU will create as many Maxeler FPGA workers as there are GPU devices.

STARPU_WORKERS_MAX_FPGAID   Select which Maxeler FPGA devices should be used to run Maxeler FPGA workers (similarly to the STARPU_WORKERS_CPUID environment variable). On a machine equipped with 4 Maxeler FPGAs, setting `STARPU_WORKERS_MAX_FPGAID="1 3"` and `STARPU_NMAX_FPGA=2` specifies that 2 Maxeler FPGA workers should be created, and that they should use Maxeler FPGA devices #1 and #3 (the logical ordering of the devices is the one reported by the Maxeler stack).

STARPU_DISABLE_ASYNCHRONOUS_MAX_FPGA_COPY   Disable asynchronous copies between CPU and Maxeler FPGA devices. One can call starpu_asynchronous_max_fpga_copy_disabled() to check whether asynchronous data transfers between CPU and Maxeler FPGA devices are disabled.

4.1.6 MPI Master Slave Workers

STARPU_NMPI_MS Specify the number of MPI master slave devices that StarPU can use.

STARPU_NMPIMSTHREADS Specift the number of threads to use on the MPI Slave devices.

STARPU_MPI_MS_MULTIPLE_THREAD Specify whether the master should use one thread per slave, or one thread for driver all slaves. Default value is 0.

STARPU_MPI_MASTER_NODE Specify the rank of the MPI process which will be the master. Default value is 0.

STARPU_DISABLE_ASYNCHRONOUS_MPI_MS_COPY Disable asynchronous copies between CPU and MPI Slave devices. One can call starpu_asynchronous_mpi_ms_copy_disabled() to check whether asynchronous data transfers between CPU and MPI Slave devices are disabled.

4.1.7 TCP/IP Master Slave Workers

STARPU_NTCPIP_MS Specify the number of TCP/IP master slave devices that StarPU can use.

STARPU_TCPIP_MS_SLAVES Specify the number of TCP/IP master slave processes that are expected to be run. This should be provided both to the master and to the slaves.

STARPU_TCPIP_MS_MASTER Specify (for slaves) the IP address of the master so they can connect to it. They will then automatically connect to each other.

STARPU_TCPIP_MS_PORT Specify the port of the master, for connexions between slaves and the master. Default value is 1234.

STARPU_NTCPIPMSTHREADS Specify the number of threads to use on the TCP/IP Slave devices.

STARPU_TCPIP_MS_MULTIPLE_THREAD Specify whether the master should use one thread per slave, or one thread for driver all slaves. Default value is 0.

STARPU_DISABLE_ASYNCHRONOUS_TCPIP_MS_COPY Disable asynchronous copies between CPU and TCP/IP Slave devices. One can call starpu_asynchronous_tcpip_ms_copy_disabled() to check whether asynchronous data transfers between CPU and TCP/IP Slave devices are disabled.

4.1.8 HIP Workers

STARPU_NHIP Specify the number of HIP devices that StarPU can use. If STARPU_NHIP is lower than the number of physical devices, it is possible to select which HIP devices should be used by the means of the environment variable STARPU_WORKERS_HIPID. By default, StarPU will create as many HIP workers as there are HIP devices.

STARPU_WORKERS_HPIID Select which HIP devices should be used to run HIP workers (similarly to the STARPU_WORKERS_HIPID environment variable). On a machine equipped with 4 HIP devices, setting `STARPU_WORKERS_HIPID="1 3"` and `STARPU_NHIP=2` specifies that 2 HIP workers should be created, and that they should use HIP devices #1 and #3.

This variable is ignored if the field starpu_conf::use_explicit_workers_hip_gpuid passed to starpu_init() is set.

STARPU_DISABLE_ASYNCHRONOUS_HIP_COPY Disable asynchronous copies between CPU and HIP devices. One can call starpu_asynchronous_hip_copy_disabled() to check whether asynchronous data transfers between CPU and HIP accelerators are disabled.

4.1.9 MPI Configuration

STARPU_MPI_THREAD_CPUID Tell StarPU to bind its MPI thread to the given CPU id, subtracted from the CPU workers (unless STARPU_NCPU is defined).

Default value is -1, it will let StarPU allocate a CPU.

STARPU_MPI_THREAD_COREID Same as STARPU_MPI_THREAD_CPUID, but bind the MPI thread to the given core ID, instead of the PU (hyperthread).

STARPU_MPI_NOBIND   Setting it to non-zero will prevent StarPU from binding the MPI to a separate core. This is for instance useful when running the testsuite on a single system.

STARPU_MPI_GPUDIRECT   Enable (1) or disable (0) MPI GPUDirect support. Default value (-1) is to enable if available. If STARPU_MPI_GPUDIRECT is explicitly set to 1, StarPU-MPI will warn if MPI does not provide the GPUDirect support.

STARPU_MPI_PSM2   This variable allows to supercede PSM2 detection when asking for MPI GPUDirect support. This is helpful when using old intel compilers, for which PSM2 detection is always true. The default (1) is to enable it. If PSM2 is detected whereas it should not be, this variable can be set to 0.

STARPU_MPI_REDUX_ARITY_THRESHOLD   The arity of the automatically-detected reduction trees follows the following rule: when the data to be reduced is of small size a flat tree is unrolled i.e. all the contributing nodes send their contribution to the root of the reduction. When the data to be reduced is of big size, a binary tree is used instead. The default threshold between flat and binary tree is 1024 bytes. By setting the environment variable with a negative value, all the automatically detected reduction trees will use flat trees. If this value is set to 0, then binary trees will always be selected. Otherwise, the setup value replaces the default 1024.

4.2 Configuring The Scheduling Engine

STARPU_SCHED   Select the scheduling policy from those proposed by StarPU: work random, stealing, greedy, with performance models, etc.

Use `STARPU_SCHED=help` to get the list of available schedulers.

STARPU_SCHED_LIB   Specify the location of a dynamic library to choose a user-defined scheduling policy. See UsingaNewSchedulingPolicy for more information.

STARPU_MIN_PRIO   Set the minimum priority used by priorities-aware schedulers. The flag can also be set through the field starpu_conf::global_sched_ctx_min_priority.

STARPU_MAX_PRIO   Set the maximum priority used by priorities-aware schedulers. The flag can also be set through the field starpu_conf::global_sched_ctx_max_priority.

STARPU_CALIBRATE   Set to 1 to calibrate the performance models during the execution. Set to 2 to drop the previous values and restart the calibration from scratch. Set to 0 to disable calibration, this is the default behaviour.

Note: this currently only applies to `dm` and `dmda` scheduling policies.

STARPU_CALIBRATE_MINIMUM   Define the minimum number of calibration measurements that will be made before considering that the performance model is calibrated. Default value is 10.

STARPU_BUS_CALIBRATE   Set to 1 to recalibrate the bus during initialization.

STARPU_PREFETCH   Enable (1) or disable (0) data prefetching. Default value is Enable.

If prefetching is enabled, when a task is scheduled to be executed e.g. on a GPU, StarPU will request an asynchronous transfer in advance, so that data is already present on the GPU when the task starts. As a result, computation and data transfers are overlapped.

STARPU_SCHED_ALPHA   To estimate the cost of a task StarPU takes into account the estimated computation time (obtained thanks to performance models). The alpha factor is the coefficient to be applied to it before adding it to the communication part.

STARPU_SCHED_BETA   To estimate the cost of a task StarPU takes into account the estimated data transfer time (obtained thanks to performance models). The beta factor is the coefficient to be applied to it before adding it to the computation part.

STARPU_SCHED_GAMMA   Define the execution time penalty of a joule (Energy-basedScheduling).

STARPU_SCHED_READY   For a modular scheduler with sorted queues below the decision component, workers pick up a task which has most of its data already available. Setting this to 0 disables this.

STARPU_SCHED_SORTED_ABOVE For a modular scheduler with queues above the decision component, it is usually sorted by priority. Setting this to 0 disables this.

STARPU_SCHED_SORTED_BELOW For a modular scheduler with queues below the decision component, they are usually sorted by priority. Setting this to 0 disables this.

STARPU_IDLE_POWER Define the idle power of the machine (Energy-basedScheduling).

STARPU_PROFILING Enable on-line performance monitoring (EnablingOn-linePerformanceMonitoring).

STARPU_CODELET_PROFILING Enable on-line performance monitoring of codelets (Per-codeletFeedback). (enabled by default)

STARPU_ENERGY_PROFILING Enable on-line energy monitoring of tasks (Per-codeletFeedback). (disabled by default)

STARPU_PROF_PAPI_EVENTS Specify which PAPI events should be recorded in the trace (PapiCounters).

4.3 Configuring The Heteroprio Scheduler

4.3.1 Configuring LAHeteroprio

STARPU_HETEROPRIO_USE_LA Enable the locality aware mode of Heteroprio which guides the distribution of tasks to workers in order to reduce the data transfers between memory nodes.

STARPU_LAHETEROPRIO_PUSH Choose between the different push strategies for locality aware Heteroprio: `WORKER`, `LcS`, `LS_SDH`, `LS_SDH2`, `LS_SDHB`, `LC_SMWB`, `AUTO` (by default: AUTO). These are detailed in LAHeteroprio

STARPU_LAHETEROPRIO_S_[ARCH] [ARCH] Specify the number of memory nodes contained in an affinity group. An affinity group will be composed of the closest memory nodes to a worker of a given architecture, and this worker will look for tasks available inside these memory nodes, before considering stealing tasks outside this group. ARCH can be `CPU`, `CUDA`, `OPENCL`, `SCC`, `MPI_MS`, etc.

STARPU_LAHETEROPRIO_PRIO_STEP_[ARCH] [ARCH] Specify the number of buckets in the local memory node in which a worker will look for available tasks, before this worker starts looking for tasks in other memory nodes' buckets. ARCH indicates that this number is specific to a given arch which can be: `CPU`, `CUDA`, `OPENCL`, `SCC`, `MPI_MS`, etc.

4.3.2 Configuring AutoHeteroprio

STARPU_HETEROPRIO_USE_AUTO_CALIBRATION Enable the auto calibration mode of Heteroprio which assign priorities to tasks automatically

STARPU_HETEROPRIO_DATA_DIR Specify the path of the directory where Heteroprio stores data about program executions. By default, these are stored in the same directory used by perfmodel.

STARPU_HETEROPRIO_DATA_FILE Specify the filename where Heteroprio will save data about the current program's execution.

STARPU_HETEROPRIO_CODELET_GROUPING_STRATEGY Choose how Heteroprio groups similar tasks. It can be `0` to group the tasks with the same perfmodel or the same codelet's name if no perfmodel was assigned. Or, it could be `1` to group the tasks only by codelet's name.

STARPU_AUTOHETEROPRIO_PRINT_DATA_ON_UPDATE Enable the printing of priorities' data every time they get updated.

STARPU_AUTOHETEROPRIO_PRINT_AFTER_ORDERING Enable the printing of priorities' order for each architecture every time there's a reordering.

STARPU_AUTOHETEROPRIO_PRIORITY_ORDERING_POLICY Specify the heuristic which will be used to assign priorities automatically. It should be an integer between 0 and 27.

STARPU_AUTOHETEROPRIO_ORDERING_INTERVAL Specify the period (in number of tasks pushed), between priorities reordering operations.

STARPU_AUTOHETEROPRIO_FREEZE_GATHERING Disable data gathering from task executions.

4.4 Extensions

SOCL_OCL_LIB_OPENCL Set the location of the file `libOpenCL.so` of the OCL ICD implementation. The SOCL test suite is only run when SOCL_OCL_LIB_OPENCL is defined.

OCL_ICD_VENDORS Set the directory where ICD files are installed. This is useful when using SOCL with Open←CL ICD (`https://forge.imag.fr/projects/ocl-icd/`). Default directory is `/etc/Open←CL/vendors`. StarPU installs ICD files in the directory `$prefix/share/starpu/opencl/vendors`.

STARPU_COMM_STATS Deprecated. You should use STARPU_MPI_STATS.

STARPU_MPI_STATS Enable (!= 0) or Disable (0) communication statistics for starpumpi (MPIDebug). Default value is Disable.

STARPU_MPI_CACHE Disable (0) or Enable (!= 0) communication cache for starpumpi (MPISupport). Default value is Enable.

STARPU_MPI_COMM Enable (1) communication trace for starpumpi (MPISupport). Also needs for StarPU to have been configured with the option --enable-verbose.

STARPU_MPI_CACHE_STATS Enable (1) statistics for the communication cache (MPISupport). Messages are printed on the standard output when data are added or removed from the received communication cache.

STARPU_MPI_PRIORITIES Disable (0) the use of priorities to order MPI communications (MPISupport).

STARPU_MPI_NDETACHED_SEND Set the number of send requests that StarPU-MPI will emit concurrently. Default value is 10. Setting it to 0 removes the limit of concurrent send requests.

STARPU_MPI_NREADY_PROCESS Set the number of requests that StarPU-MPI will submit to MPI before polling for termination of existing requests. Default value is 10. Setting it to 0 removes the limit: all requests to submit to MPI will be submitted before polling for termination of existing ones.

STARPU_MPI_FAKE_SIZE Setting to a number makes StarPU believe that there are as many MPI nodes, even if it was run on only one MPI node. This allows e.g. to simulate the execution of one of the nodes of a big cluster without actually running the rest. Of course, it does not provide computation results and timing.

STARPU_MPI_FAKE_RANK Setting to a number makes StarPU believe that it runs the given MPI node, even if it was run on only one MPI node. This allows e.g. to simulate the execution of one of the nodes of a big cluster without actually running the rest. Of course, it does not provide computation results and timing.

STARPU_MPI_COOP_SENDS Disable (0) dynamic collective operations: grouping same requests to different nodes until the data becomes available and then use a broadcast tree to execute requests.
By now, it is only supported with the NewMadeleine library (see Nmad).

STARPU_MPI_RECV_WAIT_FINALIZE Disable (1) releasing the write acquire of receiving handles when data is received but the communication library still needs the data. Set to 0 by default to unlock as soon as possible tasks which only require a read access on the handle; write access will become possible for tasks when the communication library will not need the data anymore.
By now, it is only supported with the NewMadeleine library (see Nmad).

STARPU_MPI_TRACE_SYNC_CLOCKS When `mpi_sync_clocks` is available, this library will be used to have more precise clock synchronization in traces coming from different nodes. However, the clock synchronization process can take some time (several seconds) and can be disabled by setting this variable to `0`. In that case, a less precise but faster synchronization will be used. See TraceMpi for more details.

STARPU_MPI_DRIVER_CALL_FREQUENCY When set to a positive value, activates the interleaving of the execution of tasks with the progression of MPI communications (MPISupport). The starpu_mpi_init_conf() function must have been called by the application for that environment variable to be used. When set to 0, the MPI progression thread does not use at all the driver given by users, and only focuses on making MPI communications progress.

STARPU_MPI_DRIVER_TASK_FREQUENCY   When set to a positive value, the interleaving of the execution of tasks with the progression of MPI communications mechanism to execute several tasks before checking communication requests again (MPISupport). The starpu_mpi_init_conf() function must have been called by the application for that environment variable to be used, and the STARPU_MPI_DRIVER_CALL_FREQUENCY environment variable set to a positive value.

STARPU_MPI_MEM_THROTTLE   When set to a positive value, this makes the starpu_mpi_*recv* functions block when the memory allocation required for network reception overflows the available main memory (as typically set by STARPU_LIMIT_CPU_MEM)

STARPU_MPI_EARLYDATA_ALLOCATE   When set to 1, the MPI Driver will immediately allocate the data for early requests instead of issuing a data request and blocking. Default value is 0, issuing a data request. Because it is an early request and we do not know its real priority, the data request will assume STARPU_DEFAULT_PRIO. In cases where there are many data requests with priorities greater than STARPU_DEFAULT_PRIO the MPI drive could be blocked for long periods.

STARPU_SIMGRID   When set to 1 (default value is 0), this makes StarPU check that it was really build with simulation support. This is convenient in scripts to avoid using a native version, that would try to update performance models...

STARPU_SIMGRID_TRANSFER_COST   When set to 1 (which is the default value), data transfers (over PCI bus, typically) are taken into account in SimGrid mode.

STARPU_SIMGRID_CUDA_MALLOC_COST   When set to 1 (which is the default value), CUDA malloc costs are taken into account in SimGrid mode.

STARPU_SIMGRID_CUDA_QUEUE_COST   When set to 1 (which is the default value), CUDA task and transfer queueing costs are taken into account in SimGrid mode.

STARPU_PCI_FLAT   When unset or set to 0, the platform file created for SimGrid will contain PCI bandwidths and routes.

STARPU_SIMGRID_CUDA_QUEUE_COST   When unset or set to 1, simulate within SimGrid the GPU transfer queueing.

STARPU_MALLOC_SIMULATION_FOLD   Define the size of the file used for folding virtual allocation, in MiB. Default value is 1, thus allowing 64GiB virtual memory when Linux's `sysctl vm.max_map_count` value is the default 65535.

STARPU_SIMGRID_TASK_SUBMIT_COST   When set to 1 (which is the default value), task submission costs are taken into account in SimGrid mode. This provides more accurate SimGrid predictions, especially for the beginning of the execution.

STARPU_SIMGRID_TASK_PUSH_COST   When set to 1 (which is the default value), task push costs are taken into account in SimGrid mode. This provides more accurate SimGrid predictions, especially with large dependency arities.

STARPU_SIMGRID_FETCHING_INPUT_COST   When set to 1 (which is the default value), fetching input costs are taken into account in SimGrid mode. This provides more accurate SimGrid predictions, especially regarding data transfers.

STARPU_SIMGRID_SCHED_COST   When set to 1 (0 is the default value), scheduling costs are taken into account in SimGrid mode. This provides more accurate SimGrid predictions, and allows studying scheduling overhead of the runtime system. However, it also makes simulation non-deterministic.

STARPUPY_MULTI_INTERPRETER   Enable (1) or disable (0) multi interpreters in the StarPU Python interface (MultipleInterpreters). Default value is Disable.

STARPUPY_OWN_GIL   Enable (1) or disable (0) using per-interpreter GIL (PythonParallelism). Default value is Disable for now, until python is fully ready for this.

4.5   Miscellaneous And Debug

STARPU_HOME   Specify the main directory in which StarPU stores its configuration files. Default value is `$HOME` on Unix environments, and `$USERPROFILE` on Windows environments.

STARPU_PATH   Only used on Windows environments. Specify the main directory in which StarPU is installed (RunningABasicStarPUApplicationOnMicrosoft)

STARPU_PERF_MODEL_DIR   Specify the main directory in which StarPU stores its performance model files. Default value is `$STARPU_HOME/.starpu/sampling`. See Storing_Performance_Model_Files for more details.

STARPU_PERF_MODEL_PATH   Specify a list of directories separated with ':' in which StarPU stores its performance model files. See Storing_Performance_Model_Files for more details.

STARPU_PERF_MODEL_HOMOGENEOUS_CPU   When set to 0, StarPU will assume that CPU devices do not have the same performance, and thus use different performance models for them, thus making kernel calibration much longer, since measurements have to be made for each CPU core.

STARPU_PERF_MODEL_HOMOGENEOUS_CUDA   When set to 1, StarPU will assume that all CUDA devices have the same performance, and thus share performance models for them, thus allowing kernel calibration to be much faster, since measurements only have to be once for all CUDA GPUs.

STARPU_PERF_MODEL_HOMOGENEOUS_OPENCL   When set to 1, StarPU will assume that all OpenCL devices have the same performance, and thus share performance models for them, thus allowing kernel calibration to be much faster, since measurements only have to be once for all OpenCL GPUs.

STARPU_PERF_MODEL_HOMOGENEOUS_MPI_MS   When set to 1, StarPU will assume that all MPI Slave devices have the same performance, and thus share performance models for them, thus allowing kernel calibration to be much faster, since measurements only have to be once for all MPI Slaves.

STARPU_HOSTNAME   When set, force the hostname to be used when managing performance model files. Models are indexed by machine name. When running for example on a homogenenous cluster, it is possible to share the models between machines by setting `export STARPU_HOSTNAME=some_global_name`.

STARPU_MPI_HOSTNAMES   Similar to STARPU_HOSTNAME but to define multiple nodes on a heterogeneous cluster. The variable is a list of hostnames that will be assigned to each StarPU-MPI rank considering their position and the value of starpu_mpi_world_rank() on each rank. When running, for example, on a heterogeneous cluster, it is possible to set individual models for each machine by setting `export STARPU_MPI↵_HOSTNAMES="name0 name1 name2"`. Where rank 0 will receive `name0`, rank1 will receive `name1`, and so on. This variable has precedence over STARPU_HOSTNAME.

STARPU_OPENCL_PROGRAM_DIR   Specify the directory where the OpenCL codelet source files are located. The function starpu_opencl_load_program_source() looks for the codelet in the current directory, in the directory specified by the environment variable STARPU_OPENCL_PROGRAM_DIR, in the directory `share/starpu/opencl` of the installation directory of StarPU, and finally in the source directory of StarPU.

STARPU_SILENT   Disable verbose mode at runtime when StarPU has been configured with the option --enable-verbose. Also disable the display of StarPU information and warning messages.

STARPU_MPI_DEBUG_LEVEL_MIN   Set the minimum level of debug when StarPU has been configured with the option --enable-mpi-verbose.

STARPU_MPI_DEBUG_LEVEL_MAX   Set the maximum level of debug when StarPU has been configured with the option --enable-mpi-verbose.

STARPU_LOGFILENAME   Specify in which file the debugging output should be saved to.

STARPU_FXT_PREFIX   Specify in which directory to save the generated trace if FxT is enabled.

STARPU_FXT_SUFFIX   Specify in which file to save the generated trace if FxT is enabled.

STARPU_FXT_TRACE   Enable (1) or disable (0) the FxT trace generation in `/tmp/prof_file_XXX_YYY` (the directory and file name can be changed with STARPU_FXT_PREFIX and STARPU_FXT_SUFFIX). Default value is Disable.

STARPU_FXT_EVENTS   Specify which events will be recorded in traces. By default, all events (but `VERBOSE←_EXTRA` ones) are recorded. One can set this variable to a comma- or pipe-separated list of the following categories, to record only events belonging to the selected categories:

- `USER`
- `TASK`
- `TASK_VERBOSE`
- `TASK_VERBOSE_EXTRA`
- `DATA`
- `DATA_VERBOSE`
- `WORKER`
- `WORKER_VERBOSE`
- `DSM`
- `DSM_VERBOSE`
- `SCHED`
- `SCHED_VERBOSE`
- `LOCK`
- `LOCK_VERBOSE`
- `EVENT`
- `EVENT_VERBOSE`
- `MPI`
- `MPI_VERBOSE`
- `MPI_VERBOSE_EXTRA`
- `HYP`
- `HYP_VERBOSE`

The choice of which categories have to be recorded is a tradeoff between required information for offline analyzis and acceptable overhead introduced by tracing. For instance, to inspect with ViTE which tasks workers execute, one has to at least select the `TASK` category.

Events in `VERBOSE_EXTRA` are very costly to record and can have an important impact on application performances. This is why there are disabled by default, and one has to explicitly select their categories using this variable to record them.

STARPU_LIMIT_CUDA_devid_MEM   Specify the maximum number of megabytes that should be available to the application on the CUDA device with the identifier `devid`. This variable is intended to be used for experimental purposes as it emulates devices that have a limited amount of memory. When defined, the variable overwrites the value of the variable STARPU_LIMIT_CUDA_MEM.

STARPU_LIMIT_CUDA_MEM   Specify the maximum number of megabytes that should be available to the application on each CUDA devices. This variable is intended to be used for experimental purposes as it emulates devices that have a limited amount of memory.

STARPU_LIMIT_OPENCL_devid_MEM   Specify the maximum number of megabytes that should be available to the application on the OpenCL device with the identifier `devid`. This variable is intended to be used for experimental purposes as it emulates devices that have a limited amount of memory. When defined, the variable overwrites the value of the variable STARPU_LIMIT_OPENCL_MEM.

STARPU_LIMIT_OPENCL_MEM   Specify the maximum number of megabytes that should be available to the application on each OpenCL devices. This variable is intended to be used for experimental purposes as it emulates devices that have a limited amount of memory.

STARPU_LIMIT_HIP_devid_MEM   Specify the maximum number of megabytes that should be available to the application on the HIP device with the identifier `devid`. This variable is intended to be used for experimental purposes as it emulates devices that have a limited amount of memory. When defined, the variable overwrites the value of the variable STARPU_LIMIT_HIP_MEM.

STARPU_LIMIT_HIP_MEM   Specify the maximum number of megabytes that should be available to the application on each HIP devices. This variable is intended to be used for experimental purposes as it emulates devices that have a limited amount of memory.

STARPU_LIMIT_CPU_MEM   Specify the maximum number of megabytes that should be available to the application in the main CPU memory. Setting it enables allocation cache in main memory. Setting it to zero lets StarPU overflow memory.

Note: for now not all StarPU allocations get throttled by this parameter. Notably MPI reception are not throttled unless STARPU_MPI_MEM_THROTTLE is set to 1.

STARPU_LIMIT_CPU_NUMA_devid_MEM   Specify the maximum number of megabytes that should be available to the application on the NUMA node with the OS identifier `devid`. Setting it overrides the value of STARPU_LIMIT_CPU_MEM.

STARPU_LIMIT_CPU_NUMA_MEM   Specify the maximum number of megabytes that should be available to the application on each NUMA node.   This is the same as specifying that same amount with STARPU_LIMIT_CPU_NUMA_devid_MEM for each NUMA node number.   The total memory available to StarPU will thus be this amount multiplied by the number of NUMA nodes used by StarPU. Any STARPU_LIMIT_CPU_NUMA_devid_MEM additionally specified will take over STARPU_LIMIT_CPU_NUMA_MEM.

STARPU_LIMIT_BANDWIDTH   Specify the maximum available PCI bandwidth of the system in MB/s. This can only be effective with simgrid simulation. This allows to easily override the bandwidths stored in the platform file generated from measurements on the native system. This can thus be used accelerate or slow down the system bandwidth.

STARPU_SUBALLOCATOR   Enable (1) or disable (0) the StarPU suballocator. Default value is to enable it to amortize the cost of GPU and pinned RAM allocations for small allocations: StarPU allocate large chunks of memory at a time, and suballocates the small buffers within them.

STARPU_MINIMUM_AVAILABLE_MEM   Specify the minimum percentage of memory that should be available in GPUs, i.e. not used at all by StarPU (or in main memory, when using out of core), below which a eviction pass is performed. Default value is 0%.

STARPU_TARGET_AVAILABLE_MEM   Specify the target percentage of memory that should be available in GPUs, i.e. not used at all by StarPU (or in main memory, when using out of core), when performing a periodic eviction pass. Default value is 0%.

STARPU_MINIMUM_CLEAN_BUFFERS   Specify the minimum percentage of number of buffers that should be clean in GPUs (or in main memory, when using out of core), i.e. used by StarPU, but for which a copy is available in memory (or on disk, when using out of core), below which asynchronous writebacks will be issued. Default value is 5%.

STARPU_TARGET_CLEAN_BUFFERS   Specify the target percentage of number of buffers that should be reached in GPUs (or in main memory, when using out of core), i.e. used by StarPU, but for which a copy is available in memory (or on disk, when using out of core), when performing an asynchronous writeback pass. Default value is 10%.

STARPU_DISK_SWAP   Specify a path where StarPU can push data when the main memory is getting full.

STARPU_DISK_SWAP_BACKEND   Specify the backend to be used by StarPU to push data when the main memory is getting full. Default value is `unistd` (i.e. using read/write functions), other values are `stdio` (i.e. using fread/fwrite), `unistd_o_direct` (i.e. using read/write with O_DIRECT), `leveldb` (i.e. using a leveldb database), and `hdf5` (i.e. using HDF5 library).

STARPU_DISK_SWAP_SIZE   Specify the maximum size in MiB to be used by StarPU to push data when the main memory is getting full. Default value is unlimited.

STARPU_LIMIT_MAX_SUBMITTED_TASKS Allow users to control the task submission flow by specifying to StarPU a maximum number of submitted tasks allowed at a given time, i.e. when this limit is reached task submission becomes blocking until enough tasks have completed, specified by STARPU_LIMIT_MIN_SUBMITTED_TASKS. Setting it enables allocation cache buffer reuse in main memory. See HowToReduceTheMemoryFootprintOfInternalDataStructures.

STARPU_LIMIT_MIN_SUBMITTED_TASKS Allow users to control the task submission flow by specifying to StarPU a submitted task threshold to wait before unblocking task submission. This variable has to be used in conjunction with STARPU_LIMIT_MAX_SUBMITTED_TASKS which puts the task submission thread to sleep. Setting it enables allocation cache buffer reuse in main memory. See HowToReduceTheMemory↩ FootprintOfInternalDataStructures.

STARPU_TRACE_BUFFER_SIZE Set the buffer size for recording trace events in MiB. Setting it to a big size allows to avoid pauses in the trace while it is recorded on the disk. This however also consumes memory, of course. Default value is 64.

STARPU_GENERATE_TRACE When set to `1`, indicate that StarPU should automatically generate a Paje trace when starpu_shutdown() is called.

STARPU_GENERATE_TRACE_OPTIONS When the variable STARPU_GENERATE_TRACE is set to `1` to generate a Paje trace, this variable can be set to specify options (see `starpu_fxt_tool -help`).

STARPU_ENABLE_STATS Enable gathering various data statistics (DataStatistics).

STARPU_MEMORY_STATS When set to 0, disable the display of memory statistics on data which have not been unregistered at the end of the execution (MemoryFeedback).

STARPU_MAX_MEMORY_USE When set to 1, display at the end of the execution the maximum memory used by StarPU for internal data structures during execution.

STARPU_BUS_STATS Enable the display of data transfers statistics when calling starpu_shutdown() (Profiling). By default, statistics are printed on the standard error stream, use the environment variable STARPU_BUS_STATS_FILE to define another filename.

STARPU_BUS_STATS_FILE Define the name of the file where to display data transfers statistics, see STARPU_BUS_STATS.

STARPU_WORKER_STATS Enable the display of workers statistics when calling starpu_shutdown() (Profiling). When combined with the environment variable STARPU_PROFILING, it displays the energy consumption (Energy-basedScheduling). By default, statistics are printed on the standard error stream, use the environment variable STARPU_WORKER_STATS_FILE to define another filename.

STARPU_WORKER_STATS_FILE Define the name of the file where to display workers statistics, see STARPU_WORKER_STATS.

STARPU_STATS When set to 0, data statistics will not be displayed at the end of the execution of an application (DataStatistics).

STARPU_WATCHDOG_TIMEOUT When set to a value other than 0, allows to make StarPU print an error message whenever StarPU does not terminate any task for the given time (in μs), but lets the application continue normally. Should be used in combination with STARPU_WATCHDOG_CRASH (see DetectionStuck↩ Conditions).

STARPU_WATCHDOG_CRASH When set to a value other than 0, trigger a crash when the watch dog is reached, thus allowing to catch the situation in gdb, etc (see DetectionStuckConditions)

STARPU_WATCHDOG_DELAY Delay the activation of the watchdog by the given time (in μs). This can be convenient for letting the application initialize data etc. before starting to look for idle time.

STARPU_TASK_PROGRESS Print the progression of tasks. This is convenient to determine whether a program is making progress in task execution, or is just stuck.

STARPU_TASK_BREAK_ON_PUSH When this variable contains a job id, StarPU will raise `SIGTRAP` when the task with that job id is being pushed to the scheduler, which will be nicely caught by debuggers (see DebuggingScheduling)

STARPU_TASK_BREAK_ON_SCHED When this variable contains a job id, StarPU will raise `SIGTRAP` when the task with that job id is being scheduled by the scheduler (at a scheduler-specific point), which will be nicely caught by debuggers. This only works for schedulers which have such a scheduling point defined (see DebuggingScheduling)

STARPU_TASK_BREAK_ON_POP When this variable contains a job id, StarPU will raise `SIGTRAP` when the task with that job id is being popped from the scheduler, which will be nicely caught by debuggers (see DebuggingScheduling)

STARPU_TASK_BREAK_ON_EXEC When this variable contains a job id, StarPU will raise `SIGTRAP` when the task with that job id is being executed, which will be nicely caught by debuggers (see DebuggingScheduling)

STARPU_DISABLE_KERNELS When set to a value other than 1, it disables actually calling the kernel functions, thus allowing to quickly check that the task scheme is working properly, without performing the actual application-provided computation.

STARPU_HISTORY_MAX_ERROR History-based performance models will drop measurements which are really far froom the measured average. This specifies the allowed variation. Default value is 50 (%), i.e. the measurement is allowed to be x1.5 faster or /1.5 slower than the average.

STARPU_RAND_SEED The random scheduler and some examples use random numbers for their own working. Depending on the examples, the seed is by default juste always 0 or the current time() (unless SimGrid mode is enabled, in which case it is always 0). STARPU_RAND_SEED allows to set the seed to a specific value.

STARPU_GLOBAL_ARBITER When set to a positive value, StarPU will create a arbiter, which implements an advanced but centralized management of concurrent data accesses (see ConcurrentDataAccess).

STARPU_USE_NUMA When defined to 1, NUMA nodes are taking into account by StarPU, i.e. StarPU will expose one StarPU memory node per NUMA node, and will thus schedule tasks according to data locality, migrated data when appropriate, etc.

STARPU_MAIN_RAM is then associated to the NUMA node associated to the first CPU worker if it exists, the NUMA node associated to the first GPU discovered otherwise. If StarPU doesn't find any NUMA node after these steps, STARPU_MAIN_RAM is the first NUMA node discovered by StarPU.

Applications should thus rather pass a `NULL` pointer and a -1 memory node to `starpu_data_*_↩register` functions, so that StarPU can manage memory as it wishes.

If the application wants to control memory allocation on NUMA nodes for some data, it can use starpu_malloc_on_node and pass the memory node to the `starpu_data_*_register` functions to tell StarPU where the allocation was made. starpu_memory_nodes_get_count_by_kind() and starpu_memory_node_get_ids_by_type() can be used to get the memory nodes numbers of the CPU memory nodes.

starpu_memory_nodes_numa_id_to_devid() and starpu_memory_nodes_numa_devid_to_id() are also available to convert between OS NUMA id and StarPU memory node number.

If this variable is unset, or set to 0, CPU memory is considered as only one memory node (STARPU_MAIN_RAM) and it will be up to the OS to manage migration etc. and the StarPU scheduler will not know about it.

STARPU_IDLE_FILE When defined, a file named after its contents will be created at the end of the execution. This file will contain the sum of the idle times of all the workers.

STARPU_HWLOC_INPUT When defined to the path of an XML file, `hwloc` will use this file as input instead of detecting the current platform topology, which can save significant initialization time.

To produce this XML file, use `lstopo file.xml`

STARPU_CATCH_SIGNALS By default, StarPU catch signals `SIGINT`, `SIGSEGV` and `SIGTRAP` to perform final actions such as dumping FxT trace files even though the application has crashed. Setting this variable to a value other than 1 will disable this behaviour. This should be done on JVM systems which may use these signals for their own needs. The flag can also be set through the field starpu_conf::catch_signals.

STARPU_DISPLAY_BINDINGS Display the binding of all processes and threads running on the machine. Setting it to 1 displays the binding masks. Setting it to 2 displays the topology. If MPI is enabled, display the binding of each node.
Users can manually display the binding by calling starpu_display_bindings().

4.6 Configuring The Hypervisor

SC_HYPERVISOR_POLICY Choose between the different resizing policies proposed by StarPU for the hypervisor: `idle`, `app_driven`, `feft_lp`, `teft_lp`, `ispeed_lp`, `throughput_lp` etc.

Use `SC_HYPERVISOR_POLICY=help` to get the list of available policies for the hypervisor

SC_HYPERVISOR_TRIGGER_RESIZE Choose how should the hypervisor be triggered: `speed` if the resizing algorithm should be called whenever the speed of the context does not correspond to an optimal precomputed value, `idle` it the resizing algorithm should be called whenever the workers are idle for a period longer than the value indicated when configuring the hypervisor.

SC_HYPERVISOR_START_RESIZE Indicate the moment when the resizing should be available. The value correspond to the percentage of the total time of execution of the application. Default value is the resizing frame.

SC_HYPERVISOR_MAX_SPEED_GAP Indicate the ratio of speed difference between contexts that should trigger the hypervisor. This situation may occur only when a theoretical speed could not be computed and the hypervisor has no value to compare the speed to. Otherwise the resizing of a context is not influenced by the the speed of the other contexts, but only by the the value that a context should have.

SC_HYPERVISOR_STOP_PRINT By default the values of the speed of the workers is printed during the execution of the application. If the value 1 is given to this environment variable this printing is not done.

SC_HYPERVISOR_LAZY_RESIZE By default the hypervisor resizes the contexts in a lazy way, that is workers are firstly added to a new context before removing them from the previous one. Once this workers are clearly taken into account into the new context (a task was popped there) we remove them from the previous one. However if the application would like that the change in the distribution of workers should change right away this variable should be set to 0

SC_HYPERVISOR_SAMPLE_CRITERIA By default the hypervisor uses a sample of flops when computing the speed of the contexts and of the workers. If this variable is set to `time` the hypervisor uses a sample of time (10% of an approximation of the total execution time of the application)

Chapter 5

Configuration and initialization

This section explains the relationship between configure options, compilation options and environment variables used by StarPU.

1. Configure options are used during the installation process to enable or disable specific features and libraries. These options are set using flags like --enable-maxcpus, which can be used to set the maximum number of CPUs that can be used by StarPU.

2. Compilation options are used to set specific parameters during the compilation process, such as the optimization level, architecture type, and debugging options.

3. Environment variables are used to set runtime parameters and control the behavior of the StarPU library. For example, the STARPU_NCPUS environment variable can be used to specify the number of CPUs to use at runtime, overriding the value set during compilation or installation.

Options can also be set with the different fields of the starpu_conf parameter given to starpu_init(), such as starpu_conf::ncpus, which is used to specify the number of CPUs that StarPU should use for computations.

Part I

Appendix

Chapter 6

The GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright

2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

1. PREAMBLE

    The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

    This License is a kind of ``copyleft'', which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

    We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

2. APPLICABILITY AND DEFINITIONS

    This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ``Document'', below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ``you''. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

    A ``Modified Version'' of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

    A ``Secondary Section'' is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

    The ``Invariant Sections'' are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ``Cover Texts'' are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ``Transparent'' copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not `Transparent'' is called `Opaque''.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ``Title Page'' means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ``Title Page'' means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The ``publisher'' means any person or entity that distributes copies of the Document to the public.

A section ``Entitled XYZ'' means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as `Acknowledgements'', `Dedications'', `Endorsements'', or `History''.) To ``Preserve the Title'' of such a section when you modify the Document means that it remains a section ``Entitled XYZ'' according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a

computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

(a) Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

(b) List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

(c) State on the Title page the name of the publisher of the Modified Version, as the publisher.

(d) Preserve all the copyright notices of the Document.

(e) Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

(f) Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

(g) Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

(h) Include an unaltered copy of this License.

(i) Preserve the section Entitled ``History'', Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled ``History'' in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

(j) Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the ``History'' section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

(k) For any section Entitled `Acknowledgements'' or`Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

(l) Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

(m) Delete any section Entitled ``Endorsements''. Such a section may not be included in the Modified Version.

(n) Do not retitle any existing section to be Entitled ``Endorsements'' or to conflict in title with any Invariant Section.

(o) Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ``Endorsements'', provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-↩ Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ``History'' in the various original documents, forming one section Entitled `History''`; likewise combine any sections Entitled`Acknowledgements'', and any sections Entitled `Dedications''`. You must delete all sections Entitled`Endorsements."

7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ``aggregate'' if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled `Acknowledgements''`, Dedications", or ``History'', the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

10. TERMINATION

    You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

    However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

    Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

    Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

11. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ``or any later version'' applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

12. RELICENSING

    `Massive Multiauthor Collaboration Site''` (or MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A `Massive Multiauthor Collaboration''` (or MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    ``CC-BY-SA'' means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    ``Incorporate'' means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is ``eligible for relicensing'' if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

6.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) *year your name*. 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''.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the ``with...Texts.'' line with this:

with the Invariant Sections being *list their titles*, with the Front-Cover Texts being *list*, and with the Back-Cover Texts being *list*.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.