Impact of memory allocation on the performance of a delegation synchronization algorithm: How to use the testing platform

Written by Riyane SID-LAKHDAR (M1 MoSIG)
Supervised by Thomas ROPARS (LIG, team ERODS)

June 22, 2016

This project is made of the following directories:

1 allocator:

Contains the source code of all the considered allocators (except the "custom allocator" and "ptmalloc"). To compile this allocators separately, we can use the make file contained in each allocator directory. A README file has also been created within each allocator directory.

2 tester:

Contains all the C programs that implement the delegation algorithms and the shared objects.

2.1 Compilation

During the compilation, the default allocator used by the makefile is the "custom allocator". Another allocator may be specified by setting the environment variable ***"allocatorName"***.

The default backoff value used by the "mp-..." executable is 0. Another value may be specified by setting the environment variable ***"backoffValue"***.

2.2 Execution

All the created executable may be used as follows: <executableName> [<Option> <Value>] where option belongs to:

- maxThread <Maximum number of threads used>
- testTime <duration in seconds of each run (for each number of threads)>
- allocName <Name of the allocator> This value will be written in the result files. It will not change the allocator library (see Compilation section).
- statFile <name of the file that will conatin the results>
- nbrRun <number of run for each number of threads>
- -help

3 statistic

Contains the files with the row data of the tests. During the previous experiments, the following convention has been used to name the directories:
<#cores>-<city>-<serverName>-<processor>-<options>-<backoffValue>-<executableName>.
Different other names may be specified. In each directory, 1 file exists per tested allocator

4 resultPrinter

Contains all the scripts used to evaluate the performance of the "test" programs and to print this performance. Two main python programs may be used:

On a local machine, we may run tests and print charts using the command
python printer.py [option] [attributes to print]
Option:

- -noPlot : Run the specified test and creates the specified static files. No result print is done.
- -noTest : Print the specified charts without re-executing the run (allows to do not change the current statistic files).
- -noTransactionalMemory: Uses no transactional memory during the tests (forbid group of load and store instructions to execute in an atomic way).
- -exec=<name of the executable>
- -statDir=<list of the names of the directories> Specify the directories containing the results. Different directories must be separated with '+'. The directories may be found within ../statistic.
- -allocator=<list of the allocators separated with ','>
- -backoff=<value of the backoff (in cpu cycles)>
- -plotThroughputPerCore=<list of number of threads separated with ','>
- -allThroughputPerCore
- -plotErrorBar
- -noAllocatorCompilation
- -help

Attributes:

- -nt: Number of threads
- -tp: Throughput
- -lt: Latency
- -fr: Fairness
- -allAttribute

In order to execute the previous tests on different remote machines (typically on the Grid5000 network) we can use the command:
python remotePrinter.py [option]
Option:

- -resultDir=<name of the result directory>
- -execTest=<name of the executable>

- -backoff=<value of the backoff (integer >= 0)>
- -importRes : Import previously computed results (no test is run)
- -grid5000 : Start the executions on the remote servers (need to be launched on the Grid500 network)
- -currentServer=<name of the current server> (optional)
- -currentCity=<name of the current city> (optional)
- -experience=<name of the experience> (optional)
- -help