This assignment covers the Message Passing Interface (MPI).

Literature

- *Intel®, Intel®MPI Benchmarks*:
  `https://software.intel.com/en-us/articles/intel-mpi-benchmarks/`
- *MPI Forum, MPI 2.2*:
  `http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf`
- *Intel®, Intel MPI Library*:
  `https://software.intel.com/en-us/articles/intel-mpi-library-documentation`
- *SuperMIC, documentation*:
  `http://www.lrz.de/services/compute/supermuc/supermic/`
- *J.R. Shewchuk, An Introduction to he Conjugate Gradient Method Without the Agonizing Pain*:
  `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`

1 Amdahl's Law

The SpeedUp *Sp* of an parallel application running with p processes in comparison to a serial execution can be formalized through Amdahl's Law:

$$Sp = \frac{1}{s + (1 - s)/p} \tag{1}$$

$s \in [0, 1]$ represents the serial portion of the application (often we have parts, which cannot parallelized). Given s we can compute the parallel efficiency *Eff* = *Sp/p* of a certain application.

1. Through tests you were able to prove that the sequential portion of a solver for linear systems of equations is s =10%. How big is the maximum number of processes p if you want to achieve a parallel efficiency of at least 70%?
2. Amdahl's law is some kind of pessimistic. Please explain this property w.r.t. to the max. number of processes p! Are there different/other laws in order to classify the parallel behavior of an application?

2 Probing the Network

In this task we examine the Infiniband network of the MAC-Cluster and SuperMIC in different settings.

1. Do a literature research on the infiniband switches the MAC-Cluster (ati, bdz, snb) and SuperMIC (ignore the coprocessors for the moment). Explain why the bandwidth depends on the message size! What latencies and peak bandwidths can you expect in the different settings?

2. Give a short explanation of the sub-benchmarks in the *Intel®MPI Benchmarks*.

3. Run the *Intel®MPI Benchmarks* on the MAC-Cluster (ati, bdz, snb) and SuperMIC (without the coprocessors) using four nodes! Provide plots and interpretations for at least one *Single Transfer Benchmark*, one *Parallel Transfer Benchmark* and one *Collective Benchmark*.

4. Run the benchmarks in native mode on up to four SuperMIC-nodes! Provide plots and interpretations! Do you observe the full Infiniband bandwidth of host-only execution?

3 Broadcast

1. Implement a method *broadcast* using MPI, which sends an array containing $\mathtt{n}$ `doubles` from root process 0 to all other processes within the ring.

2. Therefore, study three different algorithms:

   - `trivial`: root-process sends the array to all other $\mathtt{p}-1$ processes.
   - `tree`: data is send according to a tree structure. First process 0 sends the data to process $\mathtt{p}/2$. Afterwards, both, process 0 and process $\mathtt{p}/2$, repeat these calls in a recursive manner until all processes hold the data.
   - `bonus`: each process sends at most $\mathtt{O}(\mathtt{n})$ of `doubles`. In addition, also on the critical path only $\mathtt{O}(\mathtt{n})$ elements are sent.

3. Derive for all of your implementations the number of required MPI-messages and the amount of data sent on the critical path. In addition, measure the achieved bandwidth (bytes/time) w.r.t. to message size of your entire broadcast routine and compare these results to MPI's built-in routine `MPI_Bcast`!

4 Parallel CG

The conjugate gradient method (CG-method) is an iterative method for solving symmetric positive definite linear systems of equations given as $\mathtt{Ax} = \mathtt{b}$. Here, we are just solving the Laplacian as discussed during the lecture. `poisson.cpp` provides a serial implementation of the CG-method.

1. Parallelize this application using MPI!
   Hints: For best results sub-divide the grid into equal-sized patches (e.g. 2×2, 3×3 or 3×4, ...). According to this distribution, please create a virtual MPI communicator

topology. During execution, processes need to exchange non-compact data; you should use MPI_Vector for this task.

2. After you have a working version (you can verify your code by comparing results with the sequential version and by plotting results with gnuplot!), please perform a detailed performance analysis study featuring different grid sizes, different process-grids, as well as SpeedUps and parallel efficiencies!

Deliverables

The following deliverables have to be handed in no later than 08:00 AM, Monday, 15th December, 2014. Small files (<1 MB in total) can be send as an attachment directly to *breuera AT in.tum.de*, larger files have to be uploaded at a place of your choice, e.g. `https://github.com/`, `http://home.in.tum.de/`, `https://www.dropbox.com`. In either case inform us about the final state of your solution via e-mail.

- All of your code.
- Slides for the presentation during the next meeting; remember to address all questions in the tasks
- Output of all runs. Figures (e.g. scaling graphs) if applicable.
- Documentation how to build and use your code.