Tu., 05/15/12:

HW6: reports okay so far. Check:

- conclusions
- description of hardware and software
- check references

  Ex.: internet resource, such as a paper online

  give URL, but also author, title, etc. (year)

  "Accessed on <date>"
- Consider if you want to post your e-mail address.

Two possible outlines of a paper on the issue of Introduction vs. Conclusions:

Customary in engineering and many fields is to always have "Discussion and Conclusions". In this case, Sec. 1 may <u>not</u> tell the reader much at all. Often, math papers are long and later sections contain predictable material such as proofs. So, Sec. 1 in math papers often is the only section with interesting information!

=> one option is to move or copy conclusion to the end of Sec. 1, so that a reader is told right in the beginning, what your results will show. It is a possibility to repeat or summarize results as well

HW7: - finalize all conclusions!

Also run on the "thyme" partition which has InfiniBand (from sview)

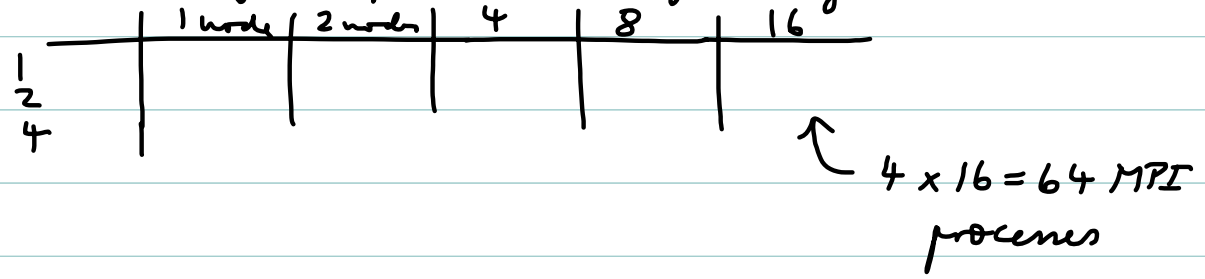
16 nodes with 8 cores each and 32GB per node

Use with --partition=thyme (no --constraint)

Step 1: do a few serial runs of cases that are no longer than 10 minutes ⇒ Goal is only to know how much faster or slower the cores are.

Step 2: Run cases with largest number of nodes first → all teams!

How many MPI processes did you have for 4 cores?



⇒ Let's restrict ourselves to 8 nodes (with ≤ 8 processes)

⇐) 64 MPI processes

What are the questions to answer:

- scalability good or not on all hardware
  i.e., is speedup good
- blocking vs. non-blocking
- comparison of public 4 cores vs. phyin 8 cores

Step 3: all other runs.

Ch. 3: Programming Exercise 1 ⟺ Sec. 13.1

Ringsend problem: All processes send a message to the next process, and the last process to Process 0.

basic idea in Pacheco: send to (id+1) % np.

Notice: % is remainder of integer division

sprintf(message, ...)

MPI_Send (message, ... (id+1)%np, ...)

MPI_Recv (message, ... (id-1)%np, ...)

printf("...%s\n", message)

not correct! Should be (id-1+np)%np

to give correct result for id=0, namely

correct = np-1 (and not "-1")

→ Sec. 13.1

Some take-home messages:

- do a test output of integers before using them, like indices into vectors or source, dest
- Always put id in output like
  printf("[%2d] ---- \n", id, ...)
- be cautious with re-using or overwriting variables like message in this example ↑ local

Ch. 4 trapezoidal rule:

Good test printout would be of l_n (to see if they all add up to n) and l_ia and l_ib from each process (to see if l_ia on Process id+1 is equal to l_ib on Process id)

Also: Focus on testing integers! Then compute l_a = a + l_ia * h from l_ia

---

Solution to load-balancing in trap for n%np != 0

```
l_n = n / np
rem = n %o np   /* remainder */
if (rem != 0) {
    if (id < rem)  l_n ++
}
```

This way the largest l_n is only 1 larger than the smallest => best load-balancing.