PDC HPC Summer Course DN2258 2011

Attendance, two weeks:
Lectures and labs: 3.0 ECTS
Get Lab attendance sheet signed !
Project, finished Fall '11: 4.5 ECTS
Grade: Grad.: P, Undergrad. : E... A
Support:
Lab assistant, Project advisor, Examiner
Project:
For some application and HPC architecture of your choice:
• Develop efficient program for non-trivial problem
• Demonstrate and report how efficient it is.

4.5 ECTS = 3 weeks of work ***incl.*** report writing
Deadline for reports: Nov 1, 2011.

The project is ***not*** about:
- *Substantial* development of *new* code.
- *Scientific results* obtained with code

So:
Prioritize measurements and analysis/interpretation!
Demonstrate use of tools (profiling, vampir, …)
and simple performance model.
NO TIME for development of new **significant** *code.*

Examples:
* Parallelize a code you know and/or work with; choose interesting part.
* Write a simple code for key algorithm of bigger solution process
* Write a simple code for a simple problem

Now – during lab-afternoons

• Discuss with instructors & course participants, form groups of size *G*.
• Define project and choose supervisor: Michael, Jesper, ...
• Write very short synopsis, check with supervisor !
• Submit synopsis to ***supervisor before end of HPC course***

Later -

• Start the work ***ASAP:***
• Finish the work; Get in touch with supervisor !!
• Submit report to ***supervisor***.

The report will be graded and sent back with comments; you may have to complete some parts and hand in again. ***We need email and paper mail address!***

- KTH and SU students: LADOK
- Other students
  - You, for registration with PhD advisor etc.
  - NGSSC, if relevant

1. Develop initial version of program;
2. Develop Performance model = theoretical prediction:
   time = *f*(problem size *N*, #processors *P*, problem partitioning parameters, …)
   Try to assess the *communication* and *computation* times separately.
3. *Measure* performance, e.g. $t = f(N,P,...)$, for different problem partitionings, if relevant
   x = wall clock time start to finish, or CPUtime, …

Size \ # proc	1	2	4 …	*n*
N_1	x	x	x	x
N_2	x	x	x	x
…				
N_M	x	x	x	x

4. If suitable, plot "speedup" and/or "efficiency", MFLOPS?, …

- Make several measurements to discover variations – discuss sources of variability. (interactive nodes, dedicated,…)
- Compare w. prediction; Interpret: ***Why these numbers?***
- Identify "bottlenecks" by profiling tools; find remedy & make changes
- Check improvement by measurements
- Write report with description of problem, ***algorithm***, and design decisions, pertinent graphs of measurements and profiling, "before and after".

Single processor performance

Algorithm:
  BLAS etc. library
Memory hierarchy
  Disk - main - cache - register;
Organization of loops
  data layout (cache misses)
  index strides (-"-)
  "unrolling"
Compute vs. save
Compiler directives (“-O3”)

Multi-processor performance

Algorithm: Communication !
  Latency vs. bandwidth
  # messages vs. size

Problem partitioning
Load balancing

Other

- Group size *G*: *G* = 2 recommended.
- "Standard" grade C. A requires exceptional work
  Requirements for grade ≥ C increase with *G*.
- Proposed schedule
  - < 11-09-22 First iteration: status report, quick feedback from advisors
  - < 11-10-24 Second (final ?) iteration, results, quick feedback/grading
  - -------------- 11-11-01 --------------
  - > 11-12-10 … evaluation may take a while
  - ***> 2012-01-01 evaluation turnaround time may be very long***
- **Report:**
  - Background, formulas, relevant problem sizes, …:
  - Algorithm, parallelization principle,...
  - “Embarrassingly parallel” OK, but performance model (?)
  - Performance model and measurements.
  - Graphs, and textual description of what the graphs show, what we learn from them
  - ***Interpretation: WHY these numbers?***

Example

Compute electric field in N gridpoints $\mathbf{g}_j$, radiated by photonic crystal rods located in $\mathbf{r}_k$, $k = 1,\ldots,K$ using M:th order Bessel expansion with coefficients

C_{km}, $m = -M,\ldots,M$

Assumption: $MK << N$

1. Compute $\mathbf{c} = \{C_{km}\}$ to satisfy field self-consistency condition: solve (small) linear system

   Ac = s **A**: $2(M+1)K$ x $2(M+1)K$

2. Compute field in the N gridpoints:

   for $j = 1{:}N$

$$E(\mathbf{g}_j) = \sum_{k=1}^{K}\left(\sum_{m=-M}^{M} C_{mk} B^m(|\mathbf{g}_j - \mathbf{r}_k|)\right)$$

"Embarrassingly parallel" algorithm:

Assign N/P gridpoints to each of the P processors!

1. Set up and solve **Ac** = **s** on proc 1.
2. Proc 1 sends **c** to all other processors ("broadcast")
3. All processors compute assigned grid points
4. All processors send their assigned gridpoints to proc. 1
5. Proc 1 plots the field

Performance model: b = 2M+1, P = # processors (cores? nodes?)

1. 3. 4. ??? better model ???

$T = c_1(bK)^3 + n\tau_F N/P + (P-1)(\tau_S + N/P\ \tau_b)$

$= \text{Const.} + n\tau_F N/P + P\tau_s$

Min. for $P = \sqrt{\dfrac{n\tau_F N}{\tau_S}}$

Measure T for different N and P !