Introduction to Parallel Computing and Accelerators

Benson Muite

Tartu Ülikool
benson.muite@ut.ee
http://math.ut.ee/~benson

20 November 2014
Outline

- Motivation
- Top500
- Accelerators
- Communication
- Desktop supercomputing
- Monte Carlo Method
- MPI
- Hadoop
Motivation

- Solve scientific problems quickly and at low cost
High spatial resolution, long time simulations
Optimize design of cars, trains, airplanes, ships
Computational Fluid Dynamics

- https://wci.llnl.gov/simulation/computer-codes/visit/gallery/gallery-08

- Optimize design of cars, trains, airplanes, ships
Optimize design of cars, trains, airplanes, ships
Optimize design of structures, consumer products, roads, bridges, cars, trains, airplanes, ships

https://wci.llnl.gov/simulation/computer-codes/visit/gallery/gallery-24
Predict and control micro structural morphology to determine macroscopic characteristics, such as fracture resistance
- Predict and control instabilities

https://wci.llnl.gov/simulation/computer-codes/visit/gallery/gallery-38
- Data Analysis
- Machine Learning
Top 500
K computer


- Picture courtesy of Riken
- 11.3 Pflops on 705,024 SPARC64 VIIIfx 8C cores
- 4th most powerful known computer in the world
Sequoia Computer

- https://asc.llnl.gov/

- 17.2 Pflops on 1,572,864 Power BGC cores
- 3rd most powerful known computer in the world
Titan Supercomputer at Oak Ridge National Laboratory

- [https://www.olcf.ornl.gov/](https://www.olcf.ornl.gov/)

- 27 Pflops on 18,688 AMD Opteron 6274 16-core CPUs and 18,688 Nvidia Tesla K20X GPUs
- 2nd most powerful known computer in the world
Tianhe II

- 33.86 Pflops on 32,000 Intel Xeon E5-2692 chips with 48,000 Xeon Phi 31S1P coprocessors
- American chips, but Fat Tree Topology Interconnect is made in China
- World’s Most Powerful Known Computer

http://www.nscc-gz.cn/
Top 500

- [link: http://top500.org/]
- Top 500 not indicative of performance for all applications
- Doesn’t consider energy efficiency ⇒ Green 500
- Graph 500 gives indication of performance for communication heavy programs
- Blue Waters has not run Linpack!
Accelerated Supercomputing
Nvidia GPUs

- http://www.nvidia.com

- 1 Tflop double precision performance
- Programming APIs CUDA, CUDA Fortran, OpenCL, OpenACC
- For compute and graphics
AMD Firepro GPUs

- [http://www.amd.com](http://www.amd.com)

- 1 Tflop double precision performance
- Programming APIs OpenCL, OpenACC
- For compute and graphics
Intel Xeon Phi

- http://www.intel.com

- 1 Tflop double precision performance
- Programming APIs OpenCL, OpenMP, MPI, CILK, Fortran, C
- For compute and graphics
Parallella

- [link](http://www.parallella.org/)
- [link](https://en.wikipedia.org/wiki/Adapteva)

- 0.1 Tflops double precision
- Programming APIs OpenCL, C, pthreads
- Embedded applications
- Energy efficient computing 50 Gflops/Watt
Nvidia Tegra K1

- http://www.nvidia.com/
- https://en.wikipedia.org/wiki/Tegra#Tegra_K1

- 0.19 Tflops double precision
- Programming APIs OpenCL
AMD APU

- http://www.amd.com
- https://en.wikipedia.org/wiki/AMD_Accelerated_Processing_Unit

- 0.700 Tflops single precision
- Programming APIs OpenCL, OpenACC, Fortran, C
- For compute and graphics
Intel HD graphics

- [http://www.intel.com](http://www.intel.com)

- Programming OpenCL
- For compute and graphics
Communication
Gigabit ethernet

- Cheapest option
- Good for minimal communication
- Low bandwidth and high latency
High performance gigabit ethernet

- Being improved
- 10 Gb/s-40Gb/s speeds
- Depending on speed and latency can be comparable to infiniband
- Price can also be comparable to infiniband
- Larger community with knowledge of ethernet networking
- 50 Gb/s speed
- commodity product
- Has typically been better than ethernet
Proprietary Interconnects

- Used on the best computers for communication intensive programs
- Cray Aries
- Fujitsu Tofu
- IBM proprietary (no marketing people have named this yet!)
Interconnects

- For high end supercomputer can be half of the hardware cost
- Worth evaluating whether extra cost needed in custom applications
Desktop Supercomputing
Personal Clusters

- Small cluster 6 - 10 nodes
- Good for evaluating new computer chip and accelerator architectures
- Good for identifying talented student programmers
- Product pre-prototypes for custom designed hardware for specific applications, eg. tomography, animation rendering, computer aided design, database search, computer games
- 10 node cluster with 10-20 Tflops using accelerators, particularly useful for sensitive/proprietary data.
- For undergraduates only: http://www.studentclustercomp.com/
- Would you want to build your own game machine and be Estonia’s next master gaming computer builder?
Main idea is to split up work into independent tasks with as few synchronizations as possible.

Kalev, Kamau and Khruschev (or Akhmatova, Algi and Atieno) need to make 99 salads.

Possible organizational options:

- Kalev, Kamau and Khruschev each make 33 salads.
- Simultaneously Kalev chops beetroots, Kamau chops tomatoes, Khruschev chops carrots. They then mix the ingredients in one bowl.
- Can you come up with others? What is the fastest assuming each of the three people are identically good at all the tasks? What if some are better at some tasks than others?
MPI (Message Passing Interface) – one way of making a program parallel

Standard can be found at:
http://www.mpi-forum.org

Use a library to allow computers to talk to each other by sending messages and having some explicit co-ordination

MPI works for C and Fortran programs. Some unofficial bindings available for other programs, such as Python:
http://mpi4py.scipy.org/

Many online resources available including:
https://computing.llnl.gov/tutorials/mpi/
http://www.shodor.org/refdesk/Resources/Tutorials/BasicMPI/
https://github.com/openmichigan/PSNM/blob/master/IntroductionToParallelProgramming/Programs/HelloworldMpi/helloworld.f90
#!/bin/bash

#SBATCH --nodes 2
#SBATCH --sockets-per-node=2
#SBATCH --cores-per-socket=10
#SBATCH --threads-per-core=1
#SBATCH --time=00:10:00

module load intel_cluster_studio_xe_2015

export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

srun helloworld

Listing 1: An example submission script for use on Rocket.
Monte Carlo Method: A Probabilistic Way to Calculate Integrals

- Recall

\[ \bar{f} = \frac{1}{b - a} \int_{a}^{b} f(x) \, dx \]

- Hence given \( \bar{f} \)

\[ \int_{a}^{b} f(x) \, dx = (b - a) \bar{f} \]

- Doing the same in 2 dimensions and estimating the error using the standard deviation

\[ \int_{R} \int f(x, y) \, dA \approx A(R) \bar{f} \pm A(R) \sqrt{\frac{\bar{f}^2 - (\bar{f})^2}{N}} \]

- Approximate \( \bar{f} \) by random sampling

\[ \bar{f} \approx \frac{\sum_{i=1}^{N} f(x_i, y_i)}{N} \quad \text{and} \quad \bar{f}^2 \approx \frac{\sum_{i=1}^{N} (f(x_i, y_i))^2}{N} \]
A program to approximate an integral using a Monte Carlo method

This could be made faster by using vectorization, however it is kept as simple as possible for clarity and ease of translation into other languages.

```
import math
import numpy
import time
numpoints=4096  # number of random sample points
l2d=0.0  # initialize value
l2dsquare=0.0  # initialize to allow for calculation of variance
for n in xrange(numpoints):
    x=numpy.random.uniform()
    y=4.0*numpy.random.uniform()
    l2d=l2d+x*x+2.0*y*y
    l2dsquare=l2dsquare+(x*x+2.0*y*y)**2

# we scale the integral by the total area and divide by the number of points used
l2d=l2d*4/numpoints
l2dsquare=l2dsquare*4/numpoints
EstimError=4*numpy.sqrt((l2d**2-l2dsquare)/numpoints)  # estimated error
print "Value:  \%f  \%f" %l2d
print "Error estimate:  \%f" %EstimError
```

Listing 2: A Python program which demonstrates how to use the Monte Carlo method to calculate the volume below \( z = x^2 + 2y^2 \), with \((x, y) \in (0, 1) \times (0, 4)\).
## Sample Results of Monte Carlo Program

<table>
<thead>
<tr>
<th>N</th>
<th>Value</th>
<th>Error Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>41.3026</td>
<td>+/- 30.9791</td>
</tr>
<tr>
<td>256</td>
<td>47.1855</td>
<td>+/- 9.0386</td>
</tr>
<tr>
<td>4096</td>
<td>43.4527</td>
<td>+/- 2.0280</td>
</tr>
<tr>
<td>65536</td>
<td>44.0026</td>
<td>+/- 0.5151</td>
</tr>
<tr>
<td>∞</td>
<td>44</td>
<td>0.0</td>
</tr>
</tbody>
</table>
https://github.com/openmichigan/PSNM/blob/master/IntroductionToParallelProgramming/Programs/montecarloserial/montecarloserial.f90
https://github.com/openmichigan/PSNM/blob/master/IntroductionToParallelProgramming/Programs/montecarloparallel/montecarloparallel.f90
- Random number generation - $o(n/p)$
- Local addition - $o(n/p)$
- Log p tree reduction on fat tree topology network $o(\log p)$
- Total time to solution $\frac{c_1 n}{p} + C_2 \log(p)$, $C_1$ and $C_2$ are constants
- Optimal $p$ given by $\frac{c_1 n}{p^2} = \frac{c_2}{p} \rightarrow p = \max \left( \frac{c_1 n}{C_2}, n \right)$
Parallel computing can help achieve scientific results faster.
Parallel computing can help solve large problems.
Many moderate size problems can be solved in a performance and cost effective manner using off the shelf hardware.
Software developers who can write architecture specific software are needed.
Strong scaling on Mira for a $4096^3$ discretization.

http://www-personal.umich.edu/~alberliu/
http://www-personal.umich.edu/~brianleu/
http://www-personal.umich.edu/~pssheth/
http://web.student.tuwien.ac.at/~e1226394/
This research used resources of:

- Mira at the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357

- Trestles at the San Diego Supercomputing Center both through the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575

- Vienna Scientific Cluster 2
Acknowledgements and References

- The Blue Waters Undergraduate Petascale Education Program administered by the Shodor foundation
- The Division of Literature, Sciences and Arts at the University of Michigan
- http://2decomp.org
- Beamer https://en.wikipedia.org/wiki/Beamer_(LaTeX)