Parallel Computing and Monte Carlo Methods

Benson Muite

benson.muite@ut.ee
http://math.ut.ee/~benson
http://en.wikibooks.org/wiki/Parallel_Spectral_Numerical_Methods

2 April 2014

Outline

- Parallel Computing
- Monte Carlo Methods for Computing Integrals

Parallel Computing

- Due to limitations in device physics

$$\text{power} = \text{resistance} \times \text{current}^2$$

  to get faster results, instead of increasing the frequency and hence the current, want to have many devices at same current, so get a linear increase in power, keeping all other things fixed
- For more on this, take distributed computing course next semester. We give a brief glimpse here.
- Will use Rocket, Fortran (primarily Fortran90 though knowledge of Fortran77 is still useful) and MPI, though similar ideas and methods work on other parallel computers such as your cell phone, laptop, video game console etc.

Parallel Computing

- Main idea is to split up work into independent tasks with as few synchronisations as possible
- Kalev, Kamau and Khruschev (or Olga, 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?

Parallel Computing: MPI

- 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/`

Hello World Example Program

https://github.com/openmichigan/PSNM/blob/master/IntroductionToParallelProgramming/Programs/HelloworldMpi/helloworld.f90

Hello World Example Rocket Submission Script

```
#!/bin/bash

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

module load intel_cluster_studio

export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

srun --cpu_bind=cores 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 $\bar{f}$

$$\int_a^b f(x)\mathrm{d}x = (b-a)\bar{f}$$

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

$$\iint_R f(x,y)\,dA \;\approx\; A(R)\bar{f} \pm A(R)\sqrt{\frac{\overline{f^2} - (\bar{f})^2}{N}}\,,$$

- Approximate $\bar{f}$ by random sampling

$$\bar{f} \approx \frac{\sum_{i=1}^N f(x_i,y_i)}{N} \quad \text{and} \quad \overline{f^2} \approx \frac{\sum_{i=1}^N (f(x_i,y_i))^2}{N}$$

Monte Carlo Method: Python Program

```
"""
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
I2d=0.0    # initialize value
I2dsquare=0.0    # initialize to allow for calculation of variance
for n in xrange(numpoints):
      x=numpy.random.uniform()
      y=4.0*numpy.random.uniform()
      I2d=I2d+x*x+2.0*y*y
      I2dsquare=I2dsquare+(x*x+2.0*y*y)**2

# we scale the integral by the total area and divide by the number of
# points used
I2d=I2d*4/numpoints
I2dsquare=I2dsquare*4/numpoints
EstimError=4*numpy.sqrt( (I2d**2-I2dsquare)/numpoints) # estimated error
print "Value:  %f" %I2d
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

N	Value	Error Estimate
16	41.3026	+/- 30.9791
256	47.1855	+/- 9.0386
4096	43.4527	+/- 2.0280
65536	44.0026	+/- 0.5151
∞	44	0.0

Monte Carlo Method: Serial Fortran Program

https://github.com/openmichigan/PSNM/blob/master/IntroductionToParallelProgramming/Programs/montecarloserial/montecarloserial.f90

Monte Carlo Method: Parallel Fortran Program

https://github.com/openmichigan/PSNM/blob/master/IntroductionToParallelProgramming/Programs/montecarloparallel/montecarloparallel.f90

New Key Concepts

- Parallel Computing – MPI
- Monte Carlo Method – method of calculating integrals

Further Work

- Several projects possible, either as summer projects or coursework
- Some of these are extensions of the homework, consider trying the optional exercises
- Please also contact me if interested in building a SUPERCOMPUTER

References

- Vainikko E. Fortran 95 Ja MPI Tartu Ülikool Kirjastus (2004) http://kodu.ut.ee/~eero/PC/F95jaMPI.pdf
- Chen G., Cloutier B., Li N., Muite B.K., Rigge P. and Balakrishnan S., Souza A., West J. “Parallel Spectral Numerical Methods” http://shodor.org/petascale/materials/UPModules/Parallel_Spectral_Methods/
- Corral M. Vector Calculus http://www.mecmath.net/
- Greenbaum A. and Chartier T.P. Numerical Methods Princeton University Press (2012)
- Sauer T. Numerical Analysis Pearson (2012)
- Petersen W.P. and Arbenz P. Introduction to Parallel Computing Oxford University Press (2004)

Acknowledgements

- Oleg Batrashev
- Leonid Dorogin
- Michael Quell
- Eero Vainikko
- The Blue Waters Undergraduate Petascale Education Program administered by the Shodor foundation
- The Division of Literature, Sciences and Arts at the University of Michigan