Building Courses Around MPI & Cuda with a LittleFe

Karl Frinkle
Mike Morris
Building Courses Around MPI & Cuda with a LittleFe

Karl Frinkle

Mike Morris
What is a LittleFe?

BigFe

LittleFe

Remember, Fe = Iron
Our first cluster – made from junk – but it worked!

This was a “SluggishFe”
Our present LittleFe

It’s called “The Kraken”
Our present LittleFe

It’s called “The Kraken”
Welcome to LittleFe.net

SC12 HPC Educators Program Buildout application is available, apply now for a free LittleFe!

Many institutions and teaching environments do not have access to parallel platforms for parallel and distributed computing education. Teaching key concepts such as speedup, efficiency, and load balancing are much more effectively done on a parallel platform. LittleFe is a complete 6 node Beowulf style portable computational cluster which supports shared memory parallelism (OpenMP), distributed memory parallelism (MPI), and GPGPU parallelism (CUDA).

LittleFe weighs less than 50 pounds, easily and safely travels via checked baggage on the airlines, and sets-up in 10 minutes wherever there is a 110/220 VAC outlet and a wall to project an image on. By leveraging the Bootable Cluster CD project, and its associated curriculum modules, LittleFe makes it possible to have a powerful ready-to-run computational science and HPC educational platform for less than $3,000. The parts list and illustrated assembly instructions are available under the "Resources” tab above.
What is MPI?

**Message Passing Interface**

**MPI** was designed for distributed memory systems, such as clusters.

As a side note, **OpenMP** is more suited to shared memory systems, such as p-threads on a single computer.
What is CUDA?

CUDA is the computing engine developed by Nvidia for high speed graphics rendering and physics calculations.

CUDA allows languages such as C/C++ to take advantage of high-speed GPUs in computationally intensive programs.
Building Courses Around MPI & Cuda with a LittleFe
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
   a. You’re here!
   b. Attend a summer workshop
      http://www.shodor.org/
      http://www.computationalscience.org/workshops/
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
II. Buy or build a cluster
   a. Try your school’s discard stash
   b. Apply for a FREE LittleFe
   c. Purchase a LittleFe (< $3,000)
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
II. Buy or build a cluster
III. Get your chair/admin to OK a course
   a. It doesn’t have to be permanent
   b. Use a “Special Seminar”
   c. Be sure to make it upper level
   d. Be prepared to teach it for free
   e. Advertise & promote the course
(Advertise & Promote the Course)

i. Exhaust the CS crowd
ii. Court the CIS/MIS/MATH students
iii. Contact other disciplines (even artsy ones)
iv. Waive most CS pre-reqs – you’ll be pleasantly surprised how quickly students adapt
v. Use your (and others’) imaginations!
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
II. Buy or build a cluster
III. Get your chair/admin to OK a course
IV. Ask your network techs to help you . . .
   a. Connect your cluster to the internet
   b. Get your cluster to a status where you can SSH into it from off-site
   c. Get MPI & CUDA running properly on your machine
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
II. Buy or build a cluster
III. Get your chair/admin to OK a course
IV. Ask your network techs to help you . . .
V. Work “Hello World” to death
   a. Start with the MPI version
   b. Make students modify it (Karl will show us)
   c. Consider waiting on CUDA until you get some MPI stuff running well
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
II. Buy or build a cluster
III. Get your chair/admin to OK a course
IV. Ask your network techs to help you . . .
V. Work “Hello World” to death
VI. Come up with a single project
   a. We used the ol’ reliable matrix multiplication
   b. Scorn us if you wish – there’s enough material there for a whole semester
Recipe for Course Development

I. Get involved with Henry Neeman, OSCER & others
II. Buy or build a cluster
III. Get your chair/admin to OK a course
IV. Ask your network techs to help you . . .
V. Work “Hello World” to death
VI. Come up with a single project
VII. Turn the students loose! (but watch over them)
... and now, we’ll let Karl take over and...
... and now, we’ll let Karl take over and...

Release The Kraken!
Our first course:  
CS4973  
Special Studies – Parallel Programming  

... from Hello World to ...  
... memory management ...  
... matrix multiplication ...  
... etc.
The Matrix: Evolution
Teach necessities as you go

Phase I: You gotta start somewhere!

1. Matrix stored in ASCII text file with rows/columns
2. Program is on one node, and in C only
3. You learn File I/O here
The Matrix: Evolution
Teach necessities as you go

Phase 2: A little better?
(1) Matrix stored in ASCII text file with no rows/columns
(2) Program is on one node, and in C only
(3) You learn a little mathematical indexing here
The Matrix: Evolution
Teach necessities as you go

Phase 3: Even Distribution over MPI?
(1) Matrix stored in binary file
(2) Program is simply distributed
(3) Learn about
  (a) MPI file I/O structure
  (b) Binary files (read, write)
  (c) Validation process to ensure working code
  (d) Computer architecture breakdown
The Matrix: Evolution
Teach necessities as you go

Phase 4: Uneven workload

1. Design programs that deal with architecture
2. Program requires more sophisticated MPI code
3. Learn about
   (a) coding to the architecture
   (b) performing tasks in the classroom with people!
   (c) program analysis
The Matrix: Evolution
Teach necessities as you go

Phase 5: Tweaking time

1. Learning new MPI commands to cut down on runtime
2. Streamlining code, how fast can your code run a problem?
3. Continue streamlining, a good program is never complete.
### Runtimes

<table>
<thead>
<tr>
<th>Program</th>
<th># Nodes</th>
<th>Dimension</th>
<th>Real Runtime</th>
<th>User Runtime</th>
<th>System Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even Distribution</td>
<td>12</td>
<td>100X100</td>
<td>12m48.744s</td>
<td>0m56.396s</td>
<td>4m31.217s</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>1000X1000</td>
<td>&gt;24 hours</td>
<td>&gt;24 hours</td>
<td>&gt;24 hours</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1000X1000</td>
<td>77m33.990s</td>
<td>23m3.086s</td>
<td>53m26.540s</td>
</tr>
<tr>
<td>Head Node Distribute</td>
<td>12</td>
<td>100X100</td>
<td>0m3.604s</td>
<td>0m0.504s</td>
<td>0m1.744s</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>1000X1000</td>
<td>3m32.956s</td>
<td>0m0.504s</td>
<td>0m1.744s</td>
</tr>
<tr>
<td>Head Node Double Distribute</td>
<td>12</td>
<td>100X100</td>
<td>0m4.134s</td>
<td>0m0.436s</td>
<td>0m1.244s</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>1000X1000</td>
<td>2m7.069s</td>
<td>0m26.554s</td>
<td>3m25.397s</td>
</tr>
</tbody>
</table>
Our second course: CS4973
Special Studies – CUDA Parallel Programming

... from Hello World in CUDA to ...
... new team projects ...
... C + MPI + CUDA ...
Our second course: CS4973
Special Studies – CUDA Parallel Programming

(1) Large number multiplication
(2) Prime number checking
(3) password cracking
Our third course:
CS4973
Special Studies – whatever we want

... pondering content at this time ...
Thank You!

We especially thank Henry Neeman, Charlie Peck, Tom Murphy, Bob Panoff and everyone else associated with OU IT, the LittleFe project and all of our colleagues and friends in the educational community involved with HPC, for all the help we have received.

Karl Frinkle – Mike Morris