Performance Engineering and Debugging HPC Applications

David Skinner

deskinner@lbl.gov
Today: Tools for Performance and Debugging

• **Principles**
  – Topics in performance scalability
  – Examples of areas where tools can help

• **Practice**
  – Where to find tools
  – Specifics to NERSC and Hopper
Big Picture of Scalability and Performance
To your goals
- Time to solution, $T_{\text{queue}} + T_{\text{run}}$
- Your research agenda
- Efficient use of allocation

To the
- application code
- input deck
- machine type/state

Performance is Relative

Suggestion:
Focus on specific use cases as opposed to making everything perform well. Bottlenecks can shift.
Performance is Hierarchical

- Registers: instructions & operands
- Caches
- Local Memory
- Remote Memory
- Disk / Filesystem

Think Globally, Compute Locally

blocks, files
Tools are Hierarchical

- Registers
- Caches
- Local Memory
- Remote Memory
- Disk / Filesystem

- PAPI
- valgrind
- PMPI
- SAR

- Craypat
- IPM
- Tau
Using the right tool

Tools can add overhead to code execution
• What level can you tolerate?

Tools can add overhead to scientists
• What level can you tolerate?

Scenarios:
• Debugging code that isn’t working
• Performance debugging
• Performance monitoring in production
One tool example: IPM on XE

1) Do “module load ipm”, link with $IPM, then run normally
2) Upon completion you get

```bash
##IPM2v0.xx##################################################
####
# command   : ./fish -n 10000
# start     : Tue Feb 08 11:05:21 2011   host      : nid06027
# stop      : Tue Feb 08 11:08:19 2011   wallclock : 177.71
# mpi_tasks : 25 on 2 nodes            %comm     : 1.62
# mem [GB]   : 0.24                      gflop/sec : 5.06
...```

Maybe that’s enough. If so you’re done.
Have a nice day 😊
HPC Tool Topics

• **CPU and memory usage**
  – FLOP rate
  – Memory high water mark

• **OpenMP**
  – OMP overhead
  – OMP scalability (finding right # threads)

• **MPI**
  – % wall time in communication
  – Detecting load imbalance
  – Analyzing message sizes
Examples of HPC tool usage
Scaling: definitions

• Scaling studies involve changing the degree of parallelism. Will we be change the problem also?

• **Strong scaling**
  – Fixed problem size

• **Weak scaling**
  – Problem size grows with additional resources

• **Speed up** = $T_s/T_p(n)$

• **Efficiency** = $T_s/(n*T_p(n))$

Be aware there are multiple definitions for these terms.
Conducting a scaling study

With a particular goal in mind, we systematically vary concurrency and/or problem size

Example:
How large a $3D \ (n^3)$ FFT can I efficiently run on 1024 cpus?

Looks good?
Let’s look a little deeper....
The scalability landscape

3D complex-complex FFTW (N=n*n*n)

<table>
<thead>
<tr>
<th>MPI Tasks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td></td>
</tr>
</tbody>
</table>

Why so bumpy?
- Algorithm complexity or switching
- Communication protocol switching
- Inter-job contention
- ~bugs in vendor software
Main loop in jacobi_omp.f90; ngrid=6144 and maxiter=20
Load (Im)balance

Communication Time: 64 tasks show 200s, 960 tasks show 230s

MPI ranks sorted by total communication time

- MPI_Alltoall
- MPI_Allreduce
- MPI_Bcast
- MPI_Reduce
- MPI_Send
- MPI_Comm_rank
- MPI_Comm_size
- MPI_Recv
Load Balance: cartoon

Unbalanced:

Task 1
Task 2
Task 3
Task 4

Balanced:

Task 1
Task 2
Task 3
Task 4

Time saved by load balance
Too much communication

Sharks and Fish (MPI)

percent communication

#cores (hopper)

"c1000" +
"c2000" ×
"c10000" *

BERKELEY LAB
National Laboratory
Simple Stuff: What’s wrong here?

Communication Event Statistics (100.00% detail)

<table>
<thead>
<tr>
<th>Event</th>
<th>Buffer Size</th>
<th>Ncalls</th>
<th>Total Time</th>
<th>Min Time</th>
<th>Max Time</th>
<th>%MPI</th>
<th>%Wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_Allreduce</td>
<td>8</td>
<td>3278848</td>
<td>124132.547</td>
<td>0.000</td>
<td>114.920</td>
<td>59.35</td>
<td>16.88</td>
</tr>
<tr>
<td>MPI_Comm_rank</td>
<td>0</td>
<td>35173439489</td>
<td>43439.102</td>
<td>0.000</td>
<td>41.961</td>
<td>20.77</td>
<td>5.91</td>
</tr>
<tr>
<td>MPI_Wait</td>
<td>98304</td>
<td>13221888</td>
<td>15710.953</td>
<td>0.000</td>
<td>3.586</td>
<td>7.51</td>
<td>2.14</td>
</tr>
<tr>
<td>MPI_Wait</td>
<td>196608</td>
<td>13221888</td>
<td>5331.236</td>
<td>0.000</td>
<td>5.716</td>
<td>2.55</td>
<td>0.72</td>
</tr>
<tr>
<td>MPI_Wait</td>
<td>589824</td>
<td>206848</td>
<td>5166.272</td>
<td>0.000</td>
<td>7.265</td>
<td>2.47</td>
<td>0.70</td>
</tr>
</tbody>
</table>
Not so simple: Comm. topology

MILC

MAESTRO

GTC

PARATEC

IMPACT-T

CAM
The transition to many-core has brought complexity to the once orderly space of hardware performance counters. NERSC, UCB, and UTK are all working on improving things.

- IPM on XE, currently just the banner is in place. We think PAPI is working (recently worked with Cray on bug fixes).
Next up…Richard.