MPI+X: Opportunities and Limitations for Heterogeneous Systems

Karl Rupp

https://karlrupp.net/

now:
Freelance Scientist

formerly:
Institute for Microelectronics, TU Wien

in collaboration with colleagues at TU Wien:
J. Weinbub, F. Rudolf, A. Morhammer, T. Grasser, A. Jüngel
...and in discussion with colleagues all over the world

MS35 - To Thread or Not To Thread
SIAM PP, Paris
April 13, 2016
Overview

Part 1: Why bother about MPI+X?

Part 2: What about X = threads?
The Big Picture

- The next large NERSC production system “Cori” will be Intel Xeon Phi KNL (Knights Landing) architecture:
  - >60 cores per node, 4 hardware threads per core
  - Total of >240 threads per node
- Your application is very likely to run on KNL with simple port, but high performance is harder to achieve.
- Many applications will not fit into the memory of a KNL node using pure MPI across all HW cores and threads because of the memory overhead for each MPI task.
- Hybrid MPI/OpenMP is the recommended programming model, to achieve scaling capability and code portability.
- Current NERSC systems (Babbage, Edison, and Hopper) can help prepare your codes.
NWChem FMC, Add OpenMP to HotSpots (OpenMP #1)

- Total number of MPI ranks=60; OMP=N means N threads per MPI rank.
- Original code uses a shared global task counter to deal with dynamic load balancing with MPI ranks.
- Loop parallelize top 10 routines in TExAS package (75% of total CPU time) with OpenMP. Has load-imbalance.
- OMP=1 has overhead over pure MPI.
- OMP=2 has overall best performance in many routines.
• OpenMP is a fun and powerful language for shared memory programming.

• Hybrid MPI/OpenMP is recommended for many next generation architectures (Intel Xeon Phi for example), including NERSC-8 system, Cori.

• You should explore to add OpenMP now if your application is flat MPI only.
## Future Systems

### ASCR Computing Upgrades at a Glance

<table>
<thead>
<tr>
<th>System attributes</th>
<th>NERSC Now</th>
<th>OLCF Now</th>
<th>ALCF Now</th>
<th>NERSC Upgrade</th>
<th>OLCF Upgrade</th>
<th>ALCF Upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Planned Installation</td>
<td>Edison</td>
<td>TITAN</td>
<td>MIRA</td>
<td>Cori 2016</td>
<td>Summit 2017-2018</td>
<td>Aurora 2018-2019</td>
</tr>
<tr>
<td>System peak (PF)</td>
<td>2.4</td>
<td>27</td>
<td>10</td>
<td>&gt;30</td>
<td>150</td>
<td>&gt;150</td>
</tr>
<tr>
<td>Peak Power (MW)</td>
<td>3</td>
<td>8.2</td>
<td>4.8</td>
<td>&lt;3.7</td>
<td>10</td>
<td>~13</td>
</tr>
<tr>
<td>System memory per node</td>
<td>64 GB</td>
<td>38 GB</td>
<td>16 GB</td>
<td>64-128 GB DDR4</td>
<td>&gt; 512 GB (High Bandwidth memory and DDR4)</td>
<td>TBA</td>
</tr>
<tr>
<td>Node performance (TF)</td>
<td>0.460</td>
<td>1.452</td>
<td>0.204</td>
<td>&gt;3</td>
<td>&gt;40</td>
<td>&gt;15 times Mira</td>
</tr>
<tr>
<td>Node processors</td>
<td>Intel Ivy Bridge</td>
<td>AMD Opteron</td>
<td>64-bit PowerPC A2</td>
<td>Intel Knights Landing many core CPUs Intel Haswell CPU in data partition</td>
<td>Multiple IBM Power9 CPUs &amp; multiple Nvidia Voltas Gpus</td>
<td>TBA</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>5,200 nodes</td>
<td>18,688 nodes</td>
<td>49,152</td>
<td>9,300 nodes</td>
<td>~3,500 nodes</td>
<td>~50,000 nodes</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>Aries</td>
<td>Gemini</td>
<td>5D Torus</td>
<td>Aries</td>
<td>Dual Rail EDR-IB</td>
<td>TBA</td>
</tr>
<tr>
<td>File System</td>
<td>17.6 PB, 169 GBs, Lustre®</td>
<td>32 PB, 1 TB/s, Lustre®</td>
<td>GPFS™</td>
<td>28 PB, 744 GB/sec, Lustre®</td>
<td>120 PB, 1 TB/s, GPFS™</td>
<td>TBA</td>
</tr>
</tbody>
</table>
Some Systems are Hybrid

- Multi-core CPUs + many-core GPUs
- Multi-core CPUs + FPGAs?
- Multi-core CPUs + custom accelerators?

Why MPI+X for Hybrid Systems?

- MPI not available on GPUs/FPGAs/accelerators

http://geco.mines.edu/tesla/cuda_tutorial.mio/pic/mpi.cuda.jpg
Simple MPI Model

Hybridization Easy?

#pragma omp parallel for
Simple MPI Model

Mind the Details

NUMA domains: Data locality matters
CPU internals: Ring buses

Intel® Xeon® Processor E5 v4 Product Family HCC

http://images.anandtech.com/doci/10158/v4_24coresHCC.png
NUMA-aware MPI Model
Hybridization

One or multiple multi-core CPUs
One or multiple GPUs/FPGAs/accelerators

CPU Starvation

Use one MPI rank per GPU (shepard process)
Problem: Waste of CPU resources
GPUs in the MPI Model
GPUs in the MPI Model

- GPUs in the MPI Model
  - MPI
  - MPI
  - node
  - GPU

- MPI
  - MPI
  - node
  - GPU

- MPI
  - MPI
  - node
  - GPU

- MPI
  - MPI
  - node
  - GPU

- MPI
  - MPI
  - node
  - GPU
Reflection

MPI+X for Hybrid Systems

Important question: What to compute where?
Unimportant question: What is X?

Sparse Matrix–Matrix Product, \( B = A^*A \)
PCI-Express Bottleneck

Vector Addition

\[ x = y + z \] with \( N \) elements each

1 FLOP per 24 byte in double precision

Limited by memory bandwidth \( \Rightarrow T_2(N) \approx 3 \times 8 \times N / \text{Bandwidth} + \text{Latency} \)
PCI-Express Bottleneck

Unpreconditioned Conjugate Gradients, NVIDIA Tesla K20m

Time per Iteration (sec)

Unpreconditioned Conjugate Gradients, NVIDIA Tesla K20m

ViennaCL
PARALUTION
MAGMA
CUSP
PETSc (2x Ivy Bridge)
Total Solver Execution Times, Poisson Equation in 2D

- Dual INTEL Xeon E5-2670 v3, No Preconditioner
- Dual INTEL Xeon E5-2670 v3, Smoothed Aggregation
- AMD FirePro W9100, No Preconditioner
- AMD FirePro W9100, Smoothed Aggregation
- NVIDIA Tesla K20m, No Preconditioner
- NVIDIA Tesla K20m, Smoothed Aggregation
- INTEL Xeon Phi 7120, No Preconditioner
- INTEL Xeon Phi 7120, Smoothed Aggregation
Example: Algebraic Multigrid

SpGEMM on CPU (AMG setup)
SpMV & friends on GPUs (AMG solve)

Model 1: Side-by-Side

Use MPI ranks for CPU
How to decompose problem?

Model 2: GPU shepards only

Overlapping of GPU and CPU work within rank
Reimplement message passing on MPI rank level?

N. Bell et al., SISC 34(4), 2012
Part 2: What about $X = \text{threads}$?
NUMA-aware MPI Model

Node 1

Node 2

Node 3

Node 4
Memory Bandwidth vs. Parallelism

STREAM Benchmark Results

- E5-2670 v3 (Haswell)
- E5-2650 v2 (Ivy Bridge)
- E5-2620 (Sandy Bridge)
- Xeon Phi 7120

https://www.karlrupp.net/2015/02/stream-benchmark-results-on-intel-xeon-and-xeon-phi/
Threads and Library Interfaces

Attempt 1

Library spawns threads

```c
void library_func(double *x, int N) {
    #pragma omp parallel for
    for (int i=0; i<N; ++i) x[i] = something_complicated();
}
```

Problems

Call from multi-threaded environment?

```c
void user_func(double **y, int N) {
    #pragma omp parallel for
    for (int j=0; j<M; ++j) library_func(y[j], N);
}
```

Incompatible OpenMP runtimes (e.g. GCC vs. ICC)
Thread and Library Interfaces

Attempt 2

Use pthreads/TBB/etc. instead of OpenMP to spawn threads
Fixes incompatible OpenMP implementations (probably)

Problems

Still a problem with multi-threaded user environments

```c
void user_func(double **y, int N) {
    #pragma omp parallel for
    for (int j=0; j<M; ++j) library_func(y[j], N);
}
```
Threads and Library Interfaces

Attempt 3

Hand back thread management to user

```c
void library_func(ThreadInfo ti, double *x, int N) {
    int start = compute_start_index(ti, N);
    int stop  = compute_stop_index(ti, N);
    for (int i=start; i<stop; ++i)
        x[i] = something_complicated();
}
```

Implications

Users can use their favorite threading model
API requires one extra parameter
Extra boilerplate code required in user code
Threads and Library Interfaces

Reflection

Extra thread communication parameter

```c
void library_func(ThreadInfo ti, double *x, int N) {...}
```

Rename thread management parameter

```c
void library_func(Thread_Comm c, double *x, int N) {...}
```

Compare:

```c
void library_func(MPI_Comm comm, double *x, int N) {...}
```

Conclusion

Prefer flat MPI over MPI+OpenMP for a composable software stack

MPI automatically brings better data locality
Summary

MPI+X Opportunities

- CPUs for sequential potions
- Non-CPUs for fine-grained parallel potions
- Get funding for reimplementing existing things

MPI+X Challenges

- Use all components of a hybrid system
- Productivity?
- Scientific progress?