Preparing Applications for Next-Generation HPC Architectures

Andrew Siegel
Argonne National Laboratory
**Exascale Computing Project**

- Department of Energy push from 10PF to 1EF

- Exascale Computing Initiative (ECI)
  1. 2 Exascale platforms (2021)
  2. Hardware R&D
  3. System software/middleware
  4. 25 Mission critical energy applications
### Pre-Exascale Systems

<table>
<thead>
<tr>
<th>Year</th>
<th>System</th>
<th>Manufacturer/Partner</th>
<th>Open/Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>Mira</td>
<td>Argonne IBM BG/Q</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Titan</td>
<td>ORNL Cray/NVidia K20</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Sequoia</td>
<td>LLNL IBM BG/Q</td>
<td>Secure</td>
</tr>
<tr>
<td>2016</td>
<td>Theta</td>
<td>Argonne Intel/Cray KNL</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Titan</td>
<td>LBNL Cray/Intel Xeon/KNL</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Trinity</td>
<td>LLNL/SNL Cray/Intel Xeon/KNL</td>
<td>Secure</td>
</tr>
<tr>
<td>2018</td>
<td>Sierra</td>
<td>ORNL IBM/NVidia P9/Volta</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Summit</td>
<td>LBNL TBD</td>
<td>Open</td>
</tr>
<tr>
<td>2020</td>
<td>NERSC-9</td>
<td>LBNL TBD</td>
<td>Open</td>
</tr>
</tbody>
</table>

### Exascale Systems

<table>
<thead>
<tr>
<th>Year</th>
<th>System</th>
<th>Manufacturer/Partner</th>
<th>Open/Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-2023</td>
<td>Frontier</td>
<td>ORNL TBD</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>El Capitan</td>
<td>LLNL TBD</td>
<td>Secure</td>
</tr>
<tr>
<td>2021</td>
<td>A21</td>
<td>Argonne Intel/Cray TBD</td>
<td>Open</td>
</tr>
</tbody>
</table>
Preparing Applications for Exascale

1. What are challenges?

1. What are we doing about it?
Harnessing FLOPS at Exascale

• Will an exascale machine be too specialized? Assumes
  – Extreme parallelism in application
  – High computational intensity (not getting worse)
  – Low aggregate RAM (5%)
  – Weak scaling: High machine value of $N^{1/2}$
  – Optimized data movement, otherwise e.g. reduced clock speed
  – Vectorizable with wider vectors
  – Specialized instruction mixes (FMA)
  – Sufficient instruction level parallelism (multiple issue)
  – Amdahl headroom
ECP Approach

• Each project begins with a mission critical science or engineering challenge problem

• The challenge problem represents a capability currently beyond the reach of existing platforms.

• Must demonstrate
  – Ability to execute problem on exascale machine
  – Ability to achieve a specified Figure of Merit
How are applications programmed?

• What changes are needed
  – To build/run code? *readiness*
  – To make efficient use of hardware? *Figure of Merit*

• Can these be expressed with current programming models?

### ECP Applications – Distribution of Programming Models

<table>
<thead>
<tr>
<th>Node\Internode</th>
<th>Explicit MPI</th>
<th>MPI via Library</th>
<th>PGAS, CHARM++, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>High</td>
<td>High</td>
<td>N/A</td>
</tr>
<tr>
<td>OpenMP</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>CUDA</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Something else</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Bottom Line: All MPI and MPI+OpenMP ubiquitous
Heavy dependence on MPI built into middleware (PetsC, Trilinos, etc)
Will we need new programming models?

- Potentially large software cost + risk to adopting new PM

- However, abstract machine model underlying both MPI and OpenMP have shortcomings, e.g.
  - Locality for OpenMP
  - Cost of synchronization for typical MPI bulk synchronous

- Good news: Standards are evolving aggressively to meet exascale needs

- Concerns remain, though
  - Can we do better with single (e.g. task-based) model
  - Can we retain performance portability?
  - What role do non-traditional accelerators play?
The role of Accelerators

• Given performance per watt, specialized accelerators (LOC/TOC combinations) lie clearly on path to exascale

• Accelerators are heavier lift for OpenMP

• Integrating MPI with accelerators for inter-node also an open question

• Low apparent software cost might be fool’s gold

• What we have seen: Current situation favors applications that follow 90/10 type rule
Programming Model Issues

• *Software cost* of adopting new PM

• *Risk* of adopting new PM

• Hardware lock-in of new PM in future
Programming Model Approaches

• Power void of MPI and OpenMP leading to zoo of new developments in programming models.
  – This is natural and not a bad thing, will likely coalesce at some point

• Plans include MPI+OpenMP but …
  – On node: Many project are experimenting with new approaches that aim at device portability: OCCA, KOKKOS, RAJA, OpenACC, OpenCL, Swift
  – Internode: Some projects are looking beyond MPI+X and adopting new or non-traditional approaches: Legion, UPC++, Global Arrays
Middleware/Solvers

• Many applications depend on MPI implicitly via middleware, eg.
  – Solvers: Petsc, Trilinos, Hypre
  – Frameworks: Chombo (AMR), Meshlib

• Major focus is to ensure project-wide that these developments lead the applications!
Rethinking algorithmic implementations

• Reduced communication/data movement
  – Sparse linear algebra, Linpack, etc.

• Much greater locality awareness
  – Likely must be exposed by programming model

• Much higher cost of global synchronization
  – Favor maxim asynchrony where physics allows

• Value to mixed precision where possible
  – Huge role in AI, harder to pin down for PDEs

• Fault resilience?
  – Likely handled outside of applications
Beyond implementations

• For applications we see hardware realities forcing new thinking beyond implementation of known algorithms
  – Adopting Monte Carlo vs. Deterministic approaches
  – Exchanging on-the-fly recomputation vs. data table lookup (e.g. neutron cross sections)
  – Moving to higher-order methods (e.g. CFD)
  – The use of ensembles vs. time-equilibrated ergodic averaging
Co-design with hardware vendors

- HPC vendors need deep engagement with applications prior to final hardware design

- *Proxy Applications* are a critical vehicle for co-design
  - ECP includes Proxy Apps Project
  - Focus on motif coverage
  - Early work with performance analysis tools and simulators

- Interest (in theory) in more complete applications.
Summary

• Major challenge for mission-critical HPC applications to get proportional performance moving toward exascale

• From application perspective high risk in being passive
  – Engage now with HPC vendors
  – Be aware of emerging technologies, particularly new ideas for programmability
  – Drive new science/engineering opportunities and numerical approaches by key features of hardware
Key Results

• Conjugate Gradient
  – If vector reductions performed in software
    • $\eta=0.5 \times n/P \geq 8500–12000$ for $P = 10^6–10^9$
  – If vector reductions performed in hardware
    • $\eta=0.5 \times n/P \geq 1200$ for $P = 10^6–10^9$

• Multigrid
  – $\eta=0.5 \times n/P \sim 10000-20000$ on machine like BG/Q
  – 2-4 times faster if hardware support for addition prefix ops

  – Bottom line: enables same simulation to run faster
Modern high throughput cores

• Also, negative impact of modern GPU-like processors

• High throughput processors do not perform near their peak in starvation limit
  – Require abundance of fine-grained tasks that are efficiently scheduled on available resources
  – Otten et al. e.g. demonstrate that, on Titan, certain problems can run faster by exploiting the additional granularity afforded through the all-CPU model rather than using the highly-tuned GPU code (albeit with a 2.5 increase in power)
Pliant Algorithms

• A bulk synchronous programming (BSP) style might not perform well on exascale architectures

• Bulk synchronous
  – compute, communicate, compute, … in unison
  – Ubiquitous style for broad class of PDEs
  – Easy/natural in MPI, but MPI != bulk synchronous

• What is potential problem?
Uneven processor execution rates

• Next generation HPC computing platforms likely characterized by significant non-uniformity in processor execution time.

• This comes from a variety of sources
  – manufacturing discrepancies, **dynamic power management**, runtime component failure, OS jitter, software mediated resiliency, and TLB/cache performance variations, etc.
  – Regardless of source $\rightarrow$ performance degradation
  – Consider recasting bulk synchronous algorithms into more asynchronous, coarse grained parallelism
Task based parallelism

• This type of “heterogeneity” has renewed interest (created a lot of hype) for task based programming models

• In some cases task based formulations are obvious and trivial to code in MPI
  – e.g. Monte Carlo, MOC

• In other cases can be much more complex
  – Demmel et al.: Reduced synchronization sparse solvers
  – Explicit timestepping (see next slide);
• Consider classic bulk synchronous formulation of explicit timestepping with arbitrary operator A, i.e. $x_{n+1} = A x_n$

• This is carried out most naturally as
  – Compute $Ax$ locally
  – Exchange stencil_width/2 values at boundaries
  – Compute $Ax$ locally
  – ....

• How will this algorithm execute in the presence of extreme processor variability?

• A. Hammouda, A.R. Siegel, S.F. Siegel, “Noise-tolerant explicit stencil computations for non-uniform process execution rates”, ACM Transactions on Parallel Computing, 2 (1), 7
Pliant version of 1D explicit timestepping
2D effect of perturbation for pliant algorithm