Seeking a sustainable approach for scientific simulation

Robert J. Harrison

harrisonrj@ornl.gov

robert.harrison@utk.edu
Recent workshop organized by PNNL
http://xsci.pnnl.gov/ppme

Key Objectives

• Highlight the effectiveness of high-level programming models within the computational and computer science communities
• Present the research and development roadmap for library and language based productive programming models
• Engage the computational sciences community to understand their current computational challenges and future application needs and requirements
• Engage the computer science community to share ideas on interoperability and scalability of programming models for next-generation computer architectures.
Some of the titles/talks I liked

- Exascale: your opportunity to create a decent HPC language
  - Gotta love Brad's optimism – 90% agreement
- Poking the Soft Underbelly of Programmer Productivity on Exascale
  - Ditto for George's pessimism – 100% agreement
- Performance, Correctness, and Programmability: Challenges for Parallel Programming at Exascale – 50% agreement
  - We will piece solution from small, mostly orthogonal concepts & components – but OMG ugh!!!!!
Exascale myths?

- Resilience – the sky is falling (when matters)?
  - Read that petascale book – were we right then?
  - Al Gara is not the only one to have said aloud that we'd be stupid to buy (and XXX to sell) a completely unreliable machine
  - Huge industries already compute on unreliable hardware
- Application S/W explicit power management?
  - Chip designers don't seem to need or want this
  - This is different from power efficient/aware algorithms
- Radically new ideas needed to express concurrency?
  - Trust (but verify) Levesque – we've seen (most of) it before
- Exascale complexity is more complex than our other complexities?
  - Ask Karol Kowalski to explain how to compute the Raman spectrum using MR-EOMCCSD(T)-f(R12) with local correlation and linear scaling w.r.t. #electrons with implicit solvation and scalar-relativistic potentials.
Do new science with

O(1) programmers
O(100,000) nodes
O(1000,000) cores
O(100,000,000)
threads & growing

- Increasing intrinsic complexity of science
- Complexity kills … sequential or parallel
  - Expressing concurrency at extreme scale
  - Managing the memory hierarchy
- Semantic gap (Colella)
  - Why are equations O(100) lines but program is O(1M)
  - What’s in the semantic gap – and how to shrink it?
Predictions

• Chemistry codes will run on first exascale machines without worrying about resilience much more than now
  • Defensive driving allowed and will be on the path to full(ish) resilience
  • The system software (detection, notification, recovery) will be the limiting factor not the science software
• Exascale-era chemistry codes will do more science and be shorter / simpler / better-by-measure-x than they are now
  • No-one will pay us to just rewrite existing functionality – and we don’t want to!
  • New capabilities with expanded science goals
• Most existing performance sensitive code must be discarded/rewritten
  • Phew! Bits of NWChem are now 40 years old!
  • But lots of code is not performance sensitive
• Chemists won't be writing (a lot of) traditional sci. code
  • Why are you planning to? How quaint! (unless U are a tool builder)
Exascale humor

• At least for Office of Science applications
More exascale myths

• It’s all about that machine in 2018
  • 2018 is just one point on the path into our future
  • Need to think even further out. What about 2030? The science applications and related software cost a lot more than the hardware and some will be alive then.

• We need a completely new, exascale-specific programming model
  • We need greatly enhanced programming models that run everywhere – laptop to exaflop

• Exascale H/W is necessarily heterogeneous
  • The c**p we have to put up with now is but don't you think that we can learn from bitter, miserable, soul crushing, graduate student wasting experiences?
Exascale technologies

• Architecture – data is everything
  • power 0.1 → 100 GFLOP/Watt  memory 0.3 → 0.03 byte/FLOP
  • cores 8 → 64-1024+ per node  total no. cores 100K → 100+M
  • concurrency $10^6 \rightarrow 10^9$

• Will be just a corner of entire ecosystem
  • In 2020 1EF = $100M = 1000$ PF
    $\rightarrow 1$PF $\leq 0.1$M
  • Most science will happen at petascale or below

• Hardware
  • Will leverage high-end server and professional computing platforms

• Software
  • Must still run everywhere; still more expensive than H/W
Exascale good news: Fine & medium grain parallelism

• Limits of coarse grain parallelism
  • Many science and engineering applications tapped out between 100K to 10M nodes
  • Power and bandwidth impose similar limits

• Technology is delivering more transistors
  • New parallelism will be primarily on chip
  • Vectors, hiding latency
  • NVIDIA GPGPU, Intel MIC, IBM BG/Q, AMD/ATI, Intel/AMD x86 etc. all converging from different directions
More exascale good news (sort of): Bandwidth and power

- Mobile and HPC have common interests
- How fast can you compute on an infinitely fast processor?
  - As fast as you can get data to it
- Which is most expensive (in both $ & W) – FLOPS/s or bandwidth?
  - Bandwidth
  - Sometimes even on chip
    (actually sometimes FPUs can dominate power)
Exascale S/W challenges

- Expressing architecture agnostic / future proof intent
- Hierarchical parallelism
  - Moving data up/down memory hierarchy
  - Co-locating work and data
  - Processes, tasks, threads, vectors
- Multiple logical threads per core
  - Sharing caches and FP units
  - Must collaborate not compete
- Resilience (when?), esp. soft / silent errors
Wish list

• Eliminate gulf between theoretical innovation in small groups and realization on high-end computers
• Eliminate the semantic gap so that efficient parallel code is no harder than doing the math
• Enable performance-portable “code” that can be automatically migrated to future architectures
• Reduce cost at all points in the life cycle

• Much of this is pipe dream – but what can we aspire to?
Scientific vs. WWW or mobile software

- Why are we not experiencing similar exponential growth in functionality?
  - Level of investment; no. of developers?
  - Lack of software interoperability and standards?
  - Competition not cooperation between groups?
  - Shifting scientific objectives?
  - Are our problems intrinsically harder?
  - Failure to embrace/develop higher levels of composition?
  - Different hardware complexity?
Dead code

- Requires human labor
  - to migrate to future architectures, or
  - to exploit additional concurrency, or
  - ...

- By these criteria most extant code is dead

- Sanity check
  - How much effort is required to port to hybrid cpu+GPGPU?
What is productivity?

• Achieve objectives
  • within people/$$/time budget
  • without exhaustive expertise
  • sustainably in face of increasing complexity or changing specs

• To be measured over total life of code
  • develop
  • tune/port/tune/port/...
  • fix bugs, extend, collaborate, embed, interface, …
  • staff turnover, student developers, …
How do we write code for a machine that does not yet exist?

- Nothing too exotic, e.g., the mix of SIMD and scalar units, registers, massive multi-threading, software/hardware managed cache, fast/slow & local/remote memory that we expect in 2018+

- Answer 1: presently cannot
  - but it’s imperative that we learn how and deploy the necessary tools

- Answer 2: don’t even try!
  - where possible generate code from high level specs
  - provides tremendous agility and freedom to explore diverse architectures
Conventional solution

- Problem statement + brain $\rightarrow$ algorithm
- Algorithm + language + brain $\rightarrow$ program
- Compile program $\rightarrow$ executable
- Computer + executable + input $\rightarrow$ result

- The brain is
  - Expensive
  - Finite
  - Not growing exponentially

The only step currently employing HPC in most applications
Cost perspectives

- 250,000 processors running for 12 hours
  - 342 processor years
- Devoting 1+% of runtime resources to load balance and scheduling is quite reasonable
  - 2,500+ processors
- Similarly for transformation, generation, compilation
  - 3.42+ year cpu time
  - What additional transformations are possible?
  - What wall time is acceptable?
  - There is no parallel compiler – “heal thyself?”
The language of many-body physics

\[ \Phi_{GW} = \frac{1}{2} - \frac{1}{2} \quad - \quad \frac{1}{4} \quad - \quad \frac{1}{6} \quad - \quad \frac{1}{8} \quad - \quad \ldots \]

Hartree       Fock       Infinite chain of *dressed* electron-hole bubbles
The Tensor Contraction Engine: A Tool for Quantum Chemistry

Oak Ridge National Laboratory
David E. Bernholdt, Venkatesh Choppella, Robert Harrison

Pacific Northwest National Laboratory
So Hirata

Louisiana State University
J Ramanujam,

Ohio State University
Gerald Baumgartner, Alina Bibireata, Daniel Cociorva, Xiaoyang Gao, Sriram Krishnamoorthy, Sandhya Krishnan, Chi-Chung Lam, Quingda Lu, Russell M. Pitzer, P Sadayappan, Alexander Sibiryakov

University of Waterloo
Marcel Nooijen, Alexander Auer

http://www.cis.ohio-state.edu/~gb/TCE/
Highly parallel codes are needed in order to apply the CC theories to larger molecular systems.

Symbolic algebra systems for coding complicated tensor expressions: Tensor Contraction Engine (TCE)
Other challenges for comp. chem.

Robust and power efficient algorithms for one-body Schrodinger

**Background:** Density functional theory in atomic orbitals, block-sparse trees with fast summation

**Science objective:** Run at scaling limit for thermodynamic integration of energy-related materials

**Issues:** Interconnect, power, resilience, scaling, numerical robustness, at scaling limit data motion dominates, irregular and small non-square matrices

Efficient and resilient algorithms to evaluate two-electron integrals

**Background:** Multiple algorithms – recursion, special functions, quadrature; near min.op. algorithms obtain ~40% peak on x86-64, but no satisfactory solution yet on current accelerators

**Science objective:** Increased accuracy and speed, more types of bases and integral

**Issues:** CPU/memory architecture, resilience, power, optimal algorithm hard to find (graph search)

Quantum locality can be exploited for data- and load-balancing via space-filling curves, from atoms (A-B) through matrices (C) to the product space (D).

![Image of molecular structures and diagrams](image-url)
<table>
<thead>
<tr>
<th>MADNESS</th>
<th>Multiresolution Adaptive Numerical Scientific Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multiresolution Adaptive Numerical Scientific Simulation

Ariana Beste¹, George I. Fann¹, Robert J. Harrison¹,², Rebecca Hartman-Baker¹, Judy Hill¹, Jun Jia¹,
¹Oak Ridge National Laboratory
²University of Tennessee, Knoxville

in collaboration with

Gregory Beylkin⁴, Lucas Monzon⁴, Martin Mohlenkamp⁵, and Hideo Sekino⁶
⁴University of Colorado
⁵Ohio University
⁶Toyohashi Technical University, Japan

harrisonrj@ornl.gov

MADNESS 2009
Funding

- MADNESS started as a DOE SciDAC project and the majority of its support still comes from the DOE
- DOE SciDAC, divisions of Advanced Scientific Computing Research and Basic Energy Science, under contract DE-AC05-00OR22725 with Oak Ridge National Laboratory, in part using the National Center for Computational Sciences.
- DARPA HPCS2: HPCS programming language evaluation
- NSF CHE 0625598: Cyber-infrastructure and Research Facilities: Chemical Computations on Future High-end Computers
- NSF CNS-0509410: CAS-AES: An integrated framework for compile-time/run-time support for multi-scale applications on high-end systems
- NSF OCI-0904972: Computational chemistry and physics beyond the petascale
What is MADNESS?

- A general purpose numerical environment for reliable and fast scientific simulation
  - Chemistry, nuclear physics, atomic physics, material science, nanoscience, climate, fusion, ...
- A general purpose parallel programming environment designed for the peta/exa-scales
- Addresses many of the sources of complexity that constrain our HPC ambitions

http://code.google.com/p/m-a-d-n-e-s-s
http://harrison2.chem.utk.edu/~rjh/madness/
Why MADNESS?

• Reduces S/W complexity
  • MATLAB-like level of composition of scientific problems with guaranteed speed and precision
  • Programmer not responsible for managing dependencies, scheduling, or placement

• Reduces numerical complexity
  • Solution of integral not differential equations
  • Framework makes latest techniques in applied math and physics available to wide audience
E.g., with guaranteed precision of $1e-6$ form a numerical representation of a Gaussian in the cube $[-20,20]^3$, solve Poisson’s equation, and plot the resulting potential (all running in parallel with threads+MPI)

Let

$$\Omega = [-20, 20]^3$$
$$\epsilon = 1e-6$$
$$g = x \rightarrow \exp\left(-\left(x_0^2 + x_1^2 + x_2^2\right)\right) \ast \pi^{-1.5}$$

In

$$f = \mathcal{F} g$$
$$u = \nabla^{-2} (-4 * \pi * f)$$

print "norm of f", \langle f \rangle, "energy", \langle f | u \rangle * 0.5
plot u

End

output: norm of f 1.000000000e+00 energy 3.98920526e-01

There are only two lines doing real work. First the Gaussian (g) is projected into the adaptive basis to the default precision. Second, the Green’s function is applied. The exact results are norm=1.0 and energy=0.3989422804.
Big picture

- Want robust algorithms that scale correctly with system size and are easy to write

- Robust, accurate, fast computation
  - Gaussian basis sets: high accuracy yields dense matrices and linear dependence – $O(N^3)$
  - Plane waves: force pseudo-potentials – $O(N^3)$
  - $O(N \log^m N \log^k \epsilon)$ is possible, guaranteed $\epsilon$

- Semantic gap
  - Why are our equations just $O(100)$ lines but programs $O(1M)$ lines?

- Facile path from laptop to exaflop
“Fast” algorithms

- Fast in mathematical sense
  - Optimal scaling of cost with accuracy & size
- Multigrid method – Brandt (1977)
  - Iterative solution of differential equations
  - Analyzes solution/error at different length scales
  - Fast application of dense operators
  - Exploits smoothness of operators
- Multiresolution analysis
  - Exploits smoothness of operators and functions
Essential techniques for fast computation

- Multiresolution

\[ V_0 \subset V_1 \subset \cdots \subset V_n \]

\[ V_n = V_0 + (V_1 - V_0) + \cdots + (V_n - V_{n-1}) \]

- Low-separation rank

\[ f(x_1, \ldots, x_n) = \sum_{l=1}^{M} \sigma_l \prod_{i=1}^{d} f_i^{(l)}(x_i) + O(\epsilon) \]

\[ \| f_i^{(l)} \|_2 = 1 \quad \sigma_l > 0 \]

- Low-operator rank

\[ A = \sum_{\mu=1}^{r} u_{\mu} \sigma_{\mu} v_{\mu}^T + O(\epsilon) \]

\[ \sigma_{\mu} > 0 \quad v_{\mu}^T v_{\lambda} = u_{\mu}^T u_{\lambda} = \delta_{\mu \lambda} \]
Let
\[ \Omega = [-20, 20]^3 \]
\[ r = x \rightarrow \sqrt{x_0^2 + x_1^2 + x_2^2} \]
\[ g = x \rightarrow \exp(-r(x)) \]
\[ v = x \rightarrow -r(x)^{-1} \]

In
\[ \psi = \mathcal{F} \, g \]
\[ \nu = \mathcal{F} \, v \]
\[ S = \langle \psi | \psi \rangle \]
\[ V = \langle \psi | \nu \ast \psi \rangle \]
\[ T = \frac{1}{2} * \sum_{i=0}^{2} (\langle \nabla_i \psi | \nabla_i \psi \rangle) \]

print \( S, V, T, \frac{T + V}{S} \)

End
H atom actual source

Let

\[ \Omega = [-20, 20]^3 \]
\[ r = x \rightarrow \sqrt{x_0^2 + x_1^2 + x_2^2} \]
\[ g = x \rightarrow \exp(-r(x)) \]
\[ v = x \rightarrow -r(x)^{-1} \]

In

\[ \psi = F g \]
\[ \nu = F v \]
\[ S = < \psi | \psi > \]
\[ V = < \psi | \nu \ast \psi > \]
\[ T = 1/2 \ast \sum_{i=0}^2 < \delta_{i} \psi | \delta_{i} \psi > \]
print S, V, T, (T + V)/S

End
Let
\[
\Omega = [-20, 20]^6
\]
\[
r_1 = x \rightarrow \sqrt{x_0^2 + x_1^2 + x_2^2}
\]
\[
r_2 = x \rightarrow \sqrt{x_3^2 + x_4^2 + x_5^2}
\]
\[
r_{12} = x \rightarrow \sqrt{(x_0 - x_3)^2 + (x_1 - x_4)^2 + (x_2 - x_5)^2}
\]
\[
g = x \rightarrow \left(1 + \frac{1}{2} \cdot r_{12}(x)\right) \cdot \exp\left(-2 \cdot (r_1(x) + r_2(x))\right)
\]
\[
v = x \rightarrow -\frac{2}{r_1(x)} - \frac{2}{r_2(x)} + \frac{1}{r_{12}(x)}
\]

In
\[
\psi = F g
\]
\[
\nu = F v
\]
\[
S = \langle \psi | \psi \rangle
\]
\[
V = \langle \psi | \nu \ast \psi \rangle
\]
\[
T = \frac{1}{2} \ast \sum_{i=0}^{5} (\langle \nabla_i \psi | \nabla_i \psi \rangle)
\]

print \( S, V, T, \frac{T + V}{S} \)

End
Let
\[ \Omega = [-20, 20]^3 \]
\[ r = x \rightarrow \sqrt{x_0^2 + x_1^2 + x_2^2} \]
\[ g = x \rightarrow \exp (-2 \ast r (x)) \]
\[ v = x \rightarrow -\frac{2}{r (x)} \]

In
\[ \nu = \mathcal{F} v \]
\[ \phi = \mathcal{F} g \]
\[ \lambda = -1.0 \]

for \( i \in [0, 10] \)
\[ \phi = \phi \ast \| \phi \|^{-1} \]
\[ V = \nu - \nabla^{-2} (4 \ast \pi \ast \phi^2) \]
\[ \psi = -2 \ast (-2 \ast \lambda - \nabla^2)^{-1} (V \ast \phi) \]
\[ \lambda = \lambda + \frac{\langle V \ast \phi | \psi - \phi \rangle}{\langle \psi | \psi \rangle} \]
\[ \phi = \psi \]

print "iter", i, "norm", \| \phi \|, "eval", \lambda
end

End
Hartree-Fock

- What I really wanted to type was
  \[
  \min E[\phi] \quad \text{s.t.} \quad \|\phi\|_2 = 1
  \]

- But had to
  - Provide E (or rather dE/d\phi)
  - Describe inexact-Newton algorithm with stopping criterion
  - Transform to integral representation for efficiency and accuracy
- Can automate some steps, c.f. Maple, Mathematica
  - But properties of computation in the underlying basis are crucial for accuracy and efficiency
MADNESS architecture

MADNESS applications - chemistry, physics, nuclear, ...

MADNESS math and numerics

MADNESS parallel runtime

MPI  Global Arrays  ARMCI  GPC/GASNET

Intel Thread Building Blocks being considered as alternative for multicore
Runtime Objectives

• Scalability to 1+M processors ASAP
• Runtime responsible for
  • scheduling and placement,
  • managing data dependencies,
  • hiding latency, and
  • Medium to coarse grain concurrency
• Compatible with existing models
  • MPI, Global Arrays
• Borrow successful concepts from Cilk, Charm++, Python
• Anticipating next gen. languages
Key elements

• Futures for hiding latency and automating dependency management
• Global names and name spaces
• Non-process centric computing
  – One-sided messaging between objects
  – Retain place=process for MPI/GA legacy compatibility
• Dynamic load balancing
  – Data redistribution, work stealing, randomization
• Map-reduce and continuation-passing models
  – Successful experiments including aggregating small tasks for use of PCI attached accelerator
Summary

- We need radical changes in how we compose scientific S/W ... not just because of exascale
  - Complexity at limits of cost and human ability
  - Need extensible tools/languages with support for code transformation not just translation

- Students need to be prepared for computing and data in 201x and 202x not 2000
  - Pervasive, massive parallelism
  - Bandwidth limited computation and analysis
  - An intrinsically multidisciplinary activity