Part 2. High-Performance Computing Issues for Simulation of Turbulent Combustion

Jacqueline H. Chen
Sandia National Laboratories
jhchen@sandia.gov
Princeton Combustion Institute
Summer Shool
June 23-28, 2019
# Technology Scaling for Exascale Systems

Can we simply scale our way up to exascale computing??

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2018</th>
<th>Factor Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Pf/s</td>
<td>1 Ef/s</td>
<td>500</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>20 MW</td>
<td>3</td>
</tr>
<tr>
<td>System Memory</td>
<td>0.3 PB</td>
<td>10 PB</td>
<td>33</td>
</tr>
<tr>
<td>Node Performance</td>
<td>0.125 Gf/s</td>
<td>10 Tf/s</td>
<td>80</td>
</tr>
<tr>
<td>Node Memory BW</td>
<td>25 GB/s</td>
<td>400 GB/s</td>
<td>16</td>
</tr>
<tr>
<td>Node Concurrency</td>
<td>12 cpus</td>
<td>1,000 cpus</td>
<td>83</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>50 GB/s</td>
<td>33</td>
</tr>
<tr>
<td>System Size (nodes)</td>
<td>20 K nodes</td>
<td>1 M nodes</td>
<td>50</td>
</tr>
<tr>
<td>Total Concurrency</td>
<td>225 K</td>
<td>1 B</td>
<td>4,444</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>300 PB</td>
<td>20</td>
</tr>
<tr>
<td>Input/Output bandwidth</td>
<td>0.2 TB/s</td>
<td>20 TB/s</td>
<td>100</td>
</tr>
</tbody>
</table>

Exascale Challenges

Parallelism

Data Movement

Programmability

Reliability

From Bill Harrod Presentation at Exascale PI Meeting, Portland, Ore, 2012.
Data Locality Management – Energy Cost of Moving Data

Figure 1: Energy cost of data movement relative to the cost of a flop for now and 2018 systems. The biggest delta in energy cost is movement of data off-chip. Therefore, future programming-environments must support the ability of algorithms and applications to exploit locality, which will, in turn, be necessary to achieve performance and energy efficiency.

Figure 1: Due to the stall in clock frequency improvements, future performance improvements will be derived from increased explicit parallelism. 2018 systems may have as many as 1 billion way parallelism.
Why Exascale for Combustion?

• Predict behavior of new fuels in different combustion scenarios at realistic pressure and turbulence conditions
  – Develop more efficient combustors
  – Explore diverse range of fuels
  – Extend the longevity of fossil fuel and reduce CO2 and other emissions
  – Government mandates (25% reduction in petroleum use by 2020 and 80% reduction in GHG emissions by 2050)

• Co-design center is focusing on high-fidelity direct numerical simulation methodologies
  – Chemical fidelity to differentiate effects of fuels where there is strong coupling with turbulence
  – Uncertainties in thermo-chemical properties
  – Not addressing complexity of geometry in engineering design codes
Multi-Regime Combustion

- Technology is pushing combustion into regimes with strong turbulence-chemistry interactions (multiple chemical scales)

- Applications operate at high pressure and under preheated conditions (>750-1000 K) where practical fuels exhibit NTC and LTHR

- Staged combustors, product recirculation, cavity driven flows, pilots provide stratification of heat and composition
Why Exascale?

- Turbulent combustion consists of phenomena occurring over a **wide range of scales** that are closely coupled
  - More grid points needed to resolve larger dynamic range of scales (Reynolds number, pressure)
  - More time steps or bigger domain needed for larger statistical ensembles
- **More complex fuel blends** require larger number of equations per grid point (O(100) species, O(1000) reactions)
- In situ uncertainty quantification with adjoint sensitivity
  - reverse causality – **uncertainties in chemical inputs**
- In situ analytics/visualization
- **More complex multi-physics** (spray and soot, hybrid Eulerian-Lagrangian particle solver)
Houston we have a problem...

<table>
<thead>
<tr>
<th>Old Constraints</th>
<th>New Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Peak clock frequency</strong>: as primary limiter for performance improvement</td>
<td><strong>Power</strong>: primary design constraint for future HPC system design</td>
</tr>
<tr>
<td><strong>Cost</strong>: FLOPs are biggest cost for system: optimize for compute</td>
<td><strong>Cost</strong>: Data movement dominates: optimize to minimize data movement</td>
</tr>
<tr>
<td><strong>Concurrency</strong>: Modest growth of parallelism by adding nodes</td>
<td><strong>Concurrency</strong>: Exponential growth of parallelism within chips</td>
</tr>
<tr>
<td><strong>Locality</strong>: MPI+X model (uniform costs within node &amp; between nodes)</td>
<td><strong>Locality</strong>: must reason about data locality and possibly topology</td>
</tr>
<tr>
<td><strong>Memory Scaling</strong>: maintain byte per flop capacity and bandwidth</td>
<td><strong>Memory Scaling</strong>: Compute growing 2x faster than capacity or bandwidth, no global hardware cache coherence</td>
</tr>
<tr>
<td><strong>Uniformity</strong>: Assume uniform system performance</td>
<td><strong>Heterogeneity</strong>: Architectural and performance non-uniformity increase</td>
</tr>
</tbody>
</table>

- **Future algorithms, programming environments, runtimes, hardware need to**:
  - Express data locality (sometimes at the expense of FLOPS) and independence
  - Allow expression of massive parallelism
  - Minimize data movement and reduce synchronization
  - Detect and address faults
ExaCT Co-Design Vision and Goal

http://www.exactcodesign.org

• Goal of combustion exascale co-design is to consider all aspects of the combustion simulation process from formulation and basic algorithms to programming environments to hardware characteristics needed to enable combustion simulations (including in situ UQ and analytics) on exascale architectures
  – Interact with vendors to help define hardware requirements, computer scientists on requirements for programming environment and software stack, and applied mathematics community locality-aware algorithms for PDE’s, UQ, and analytics
  – High-fidelity block structured adaptive mesh refinement (AMR) with embedded UQ and in situ analytics

• Combustion is a surrogate for a much broader range of multi-physics computational science areas
Co-Design Approach: Proxy Architectures and Proxy Apps

- Mini-Applications and Proxy Apps (https://cse.lbl.gov/ExaCT/index.html)
  - Express essential parts of applications that are essential to science
  - Express pain points for performance
  - Hide complexities of REAL apps that are inconsequential to science

- Proxy Hardware Models are a Reflection of the Underlying Machine Architecture
  - Express what is important for performance
  - Hide complexity that is not consequential to performance

- Use Proxy Applications on Proxy Hardware Architectures to understand what choices offer the highest value (using modeling and simulation)
  - Not absolute performance predictions
  - Scientific experiments that gauge relative improvements over control

- Still Need Full-Applications
Petascale Codes Provide Starting Point for Co-design

- **S3D**
  - Compressible formulation
  - Eighth-order finite difference discretization
  - Fourth-order Runge-Kutta temporal integrator
  - Detailed kinetics and transport
  - Hybrid parallel model with MPI + OpenMP
  - OpenACC (pragmas for GPU’s)
  - Legion (deferred execution and hides latencies)

- **LMC**
  - Low Mach Number model that exploits separation of scales between acoustic wave speed and fluid motion
  - Second-order projection formulation
  - Detailed kinetics and transport
  - Block-structure adaptive mesh refinement
  - Hybrid parallel model with MPI + OpenMP

Expectation is that exascale will require new code base
Solvers and UQ

- Goal is to develop high-order compressible and low Mach number adaptive mesh solvers

- Exact is pursuing several research activities to achieve these goals
  - Alternative time stepping strategies
  - Higher-order AMR
  - Iterative linear solvers (multigrid)
  - Improved data distribution methodology for AMR data layout
  - Incorporation of adjoints for sensitivity analysis
Improved Temporal Integration

• Developing new temporal integration approaches based on Spectral Deferred Corrections (SDC)
  – Recast temporal evolution as an integral in time
  – Iteratively solve resulting spectral collocation system

• SDC offers a number of advantages over traditional Runge Kutta schemes
  – Multirate integration – different processes on different time scale
  – Algorithmically resilient temporal integrations
  – Multilevel SDC coupled to AMR – multigrid in space / time
Effective use of exascale hardware will require programming environment that effectively maps algorithms to hardware

- Driven by programmability of combustion applications and characterization of algorithms on different designs of architectures
  - Simplify programming to express locality and independence
  - Automate discovery of parallelism and hide latencies
  - Simplify programming of extensible workflows, block-structured PDE’s, analytics, UQ for performance, scalability, portability, and productivity on heterogeneous architectures
Parallel Programming 101 - Productivity

- Functionally correct application code
- Mapping to target machine
- Extraction of parallelism
- Management of data transfers
- Task scheduling
- Latency hiding
- Data-Dependent Behavior
Parallel Programming 101

- Functionally correct application code
- Mapping to target machine
- Extraction of parallelism
- Management of data transfers
- Compiler/Runtime understanding of data
- Task scheduling
  - Latency hiding
- Data-Dependent Behavior
Legion Programming Model

http://legion.stanford.edu

• **Motivation:**
  – Performance-oriented, portable, productive

• **Model:**
  – Capture the structure of program data
  – Decouple specification from mapping to memory hierarchy and processors
  – Enable flexible programming choices at node level – e.g. TiDA
  – Automate: data movement, parallelism discovery, synchronization, hiding long latency
  – Support for task- and data-parallel forms

• **Approach:**
  – Proxy applications reveal little detail of impact a programming model has at the scale of a full application – therefore we ported a full application to Legion.
Describing Data with Regions

• Logical regions (array of objects):
  – Have no implied layout
  – Have no implied location

• Described by:
  – Index space (set of keys)
  – Field space (set of fields)

• Operations include:
  – Partitioning into subregions
  – Slicing by fields
Legion Tasks

- Hierarchical Tree of Tasks
- Legion tasks specify:
  - (Sub)region usage
  - Field usage
  - Access modes
- Legion runtime:
  - Infers data dependences
  - Inserts copies
Mapping Interface

• Application selects:
  – Where tasks run
  – Where regions are placed

• Mapping computed dynamically

• Decouple correctness from performance
S3D

- Production combustion simulation
- Written in ~200K lines of Fortran
- Direct numerical simulation using explicit MOL methods
S3D Versions

- Supports many chemical mechanism
  - DME (30 species)
  - Heptane (52 species)

- Fortran + MPI
  - Vectorizes well
  - MPI used for multi-core

- “Hybrid” OpenACC
  - Recent work by Cray/Nvidia/DOE

- Legion interoperates with MPI
Parallelism in S3D

• Data is large 3D cartesian grid of cells

• Typical per-node subgrid is $48^3$ or $64^3$ cells
  – Nearly all kernels are per-cell
  – Embarrassingly data parallel

• Hundreds of tasks
  – Significant task-level parallelism

• Except...
  – Computational intensity is low
  – Large working sets per cell (1000s of temporaries)
  – Performance limiter is data, not compute
S3D Task Parallelism

- One call to Right-Hand-Side-Function (RHSF) as seen by the Legion runtime
  - Called 6 times per time step by Runge-Kutta solver
  - Width == task parallelism
  - H2 mechanism (only 9 species)
  - Heptane (52 species) is significantly wider
- Manual task scheduling would be difficult!

Dependence graph of tasks for RHS for hydrogen
S3D Legion GPU Performance

Figure 7: GPU Usage by Task on Titan

Figure 8: Roofline Analysis of Key GPU Kernels
Leaf Tasks

- Legion treats tasks as black boxes
  - Doesn’t care how tasks are written

- Still need fast leaf tasks for computationally expensive chemistry, diffusion, viscosity
  - For CPUs & GPUs
  - For multiple mechanisms

- Singe* is a DSL compiler for chemistry kernels

*Bauer et al. PPoPP’14
Combustion Challenges

• GPU programming models emphasize data parallelism
  – Not always the best choice for performance

• Large working sets (per point)
  – PRF chemistry needs 1722 double precision reaction rates (per point)
  – GPU register file only store 128 per thread

• Multi-phase computations
  – Fissioned kernels limited by memory bandwidth, slow
Warp Specialization

- Leverage knowledge of underlying hardware
  - GPUs execute warps: streams of 32-wide vector instructions
  - All threads in warp execute the same program (data parallel unit)

- Each warp can run different computation
  - Generate code that specializes each warp, leverage task parallelism
  - Different warps do different computations on the same data
  - Allows much better use of memory while keeping processors busy
  - Fit large working sets on chip
In-Situ Chemical Explosive Mode Analytics (CEMA)

- CEMA: eigenvalue solve on the reaction rate Jacobian to determine the mode of combustion

\[
\text{sign}(\text{Re}(\lambda)) \times \log_{10}(1 + |\text{Re}(\lambda)|), \text{ 1/s)}
\]

- Run CEMA at each time step as a diagnostic to steer mesh refinement
- CEMA computation takes longer than a single explicit RK stage (6 stages/timestep)
- Dividing CEMA across RK stages and interleaving with other computation so as not to impact other critical operations would be hard to schedule manually
- Asynchronous task execution, schedule CEMA on CPU resources
- Interoperate Fortran CEMA with Legion code – took a day to implement
Legion S3D Execution with In-situ Analytics on a Titan node

Figure 13: Example All-GPU Mapping Strategy with CEMA In-Situ Analysis for S3D on a Titan Node.
Performance Results

• Chemistry Kernel
  – All Singe kernels significantly faster than current production versions
  – Warp specialized SINGE code is up to 3.75 times faster than previously optimized data-parallel CUDA kernels

• Multi-Node Heterogeneous Testbeds S3D Legion:
  – Keeneland: 128 nodes, 16 Sandy Bridge cores (24 GB RAM), 3 Fermis (6 GB RAM each)
  – Titan: 18K nodes, 16 Interlagos cores (32GB), 1 Kepler K20X GPU (6GB), Cray Gemini interconnect (2nd on Top500)
  – Piz Daint: 5272 nodes, 8-core Sandy Bridge-EP CPU (32GB), 1 Kepler K20X (6GB) GPU, and Cray Aries interconnect (6th on Top500)
S3D Legion Performance on Titan – Weak Scaling

$48^3$ PRF 116 species

Figure 9: Weak Scaling of PRF on Titan
Execution Overhead of In-situ Analytics (CEMA)

<table>
<thead>
<tr>
<th>System</th>
<th>Time per Time Step (s)</th>
<th>Without CEMA</th>
<th>With CEMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI Fortran</td>
<td>6.79</td>
<td>7.30</td>
<td></td>
</tr>
<tr>
<td>Piz Daint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legion</td>
<td>1.79</td>
<td>1.80</td>
<td></td>
</tr>
<tr>
<td>Piz Daint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI Fortran</td>
<td>7.25</td>
<td>8.42</td>
<td></td>
</tr>
<tr>
<td>Titan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legion</td>
<td>2.25</td>
<td>2.44</td>
<td></td>
</tr>
<tr>
<td>Titan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

98% 84%
Dual Fuel RCCI combustion – controlled HCCI

Reactivity Controlled Compression Ignition
Optimized in-cylinder fuel blending of high cetane diesel with high octane gasoline: control phasing (ignition timing relative to piston motion) and combustion rate

Port injected gasoline
Early direct injected diesel

Control of combustion duration by ratio of fuels

- Cool Flame: Primarily n-heptane
- PRF Burn: n-heptane + entrained iso-octane
- Iso-octane Burn: Primarily iso-octane

AHRR [J/°]

Crank [°ATDC]
Legion S3D: Reactivity Controlled Compression Ignition Primary Reference Fuel Gasoline Blend
RCCI Combustion Mode: Premixed Flames or Spontaneous Autoignition

High cetane fuel promotes premixed flame propagation

Figure 3: Volume rendering of the heat release rate at the time corresponding to 50% of total heat release. Values are in J/m^3/s.

Figure 6: Comparison of the diffusion rate and reaction rate of the OH radical along a flame surface (left) and an ignition front (right).
Turbulent n-C12H26 / air mixing layer autoignition at 25 bar – Borghesi and Chen 2016

- **Pressure:** 25 bar
- **Air stream:** 15% $X_{O_2}$ + 85% $X_{N_2}$, $T=960$ K
- **Fuel stream:** $n$-dodecane, $\xi = 0.3$, $T=450$
- **Kinetics:** 35-species non-stiff reduced model with low and high-T by T. Lu
- **Fuel jet velocity:** 21 m/s, $Re_j = 7000$
- **Setup:**
  - Grid: 1200 x 1500 x 1000 nodes
  - Resolution: 4 micron in fine-grid region
  - Dimensions: 3.6 mm x 4.0 mm x 3.0 mm
  - BCs: X and Z periodic, Y NSCBC outflows
  - $Re_j = 6600$, $Da = 0.4$

**Figure:** heat release rate after hot ignition, blue line is stoichiometric
Legion S3D Lessons Learned

• Legion
  – S3D shows potential of data-centric, task-based models
  – Enables new simulation capabilities (physics, and in situ analytics)
  – Code is easier to modify and maintain
    • Ports are just new mappings, easy to tune for performance
    • New functionality usually just means new tasks
    • Legion will figure out the dependences and scheduling
    • Productivity requires higher level abstraction layer for scientists to write in

• Co-Design
  – The Legion/S3D experience is a tribute to co-design
  – Computer and computational scientists worked closely
  – Major progress on important problems resulted
Exascale Targets: Science at Relevant Conditions

- Reactivity Controlled Compression Ignition (RCCI engine combustion) – ‘Chemical’ engine with high diesel-like efficiency without NOx and soot, tailor the charge stratification through direct injection of high-cetane fuel to control combustion phasing, burn rate, and soot formation/oxidation.


- Include in-situ analytics and uncertainty quantification.
Combustion generates massive data that needs to be analyzed and visualized and to steer the computation.
The current workflow of compute first, analyze later does not scale on projected high performance computing architectures.
The current workflow of compute first, analyze later does not scale on projected high performance computing architectures.

- There is a widening gap between compute power & available I/O rates
- It will be difficult to save data for post processing at a high enough temporal frequency
- We need new concurrent workflow designs
Workflow designs can have a significant impact on design and implementation of analysis algorithms

- **Explore the design space of new workflows**
  - Location of analysis compute resources
  - Data access, placement, and persistence
  - Synchronization and scheduling

- **Investigate impact of workflows on analysis algorithms**
  - In-situ
  - In-transit
  - Hybrid in-situ + in-transit
Data management for adjoint based UQ workflow

1. Initial DNS solution of the forward model over a long window
   - Infrequent analytics (every 100 timesteps)
   - Checkpoints to Disk (for Resiliency) every thousand timesteps.

2. An “offline” stage: analytics results target potential events for UQ.

3. For the duration of each temporal event: re-launch the forward model from Resiliency checkpoint. solution for SA. If in the event window checkpoint.

4. At the end of the event: Reverse the adjoint model to the beginning of the event, solving an adjoint model for each QoI.
   - Sensitivity contributions are accumulated over each timestep and stored.
   - Forward state data is regenerated/read from storage (Heuristics)
   - Precede to the next event, and return to Step Three.

Evaluation Metrics

- Understanding energy/performance tradeoffs
  - Frequency of check point
  - Size of the memory window used for saving state
  - Re-computing every time step vs. saving intermediate states
  - Traditional storage (e.g., HDD) vs. NVRAM (e.g., SSD)
Reducing Data Sizes Through Regions of Influence

Use Case I: Lifted Flame

- Goal: Analyze sensitivity of ignition kernels
- Global analysis is has infeasible data requirements
- Study different scenarios to restrict analysis to RoIs: static fixed size (S1), moving fixed size (S2), fully dynamic (S3)

- Restricting adjoint computation to RoIs can drastically lower the data requirements:
  - Identify events with insitu topological analysis
  - Store RoI with padding depending on the scenario
  - Analyze each RoI independently offline
  - New adjoint techniques able to processes fully dynamic RoIs may result in additional orders of magnitude data reduction
In situ Feature Detection using Topological Techniques

- Topological structure such as the merge tree (right) provide highly flexible feature definitions
- Features can be extracted with minimal overheads (<1%) insitu
- Provides tracking graphs for all features (bottom) to inform data storage and adjunct calculation

Use case II: HCCI
- Track ignition kernels till space-filling flame appears
Feature segmentations are used to understand relationships between multiple scalar fields.
Tracking Ignition Kernels Merging with the Lifted Flame
In situ Visualization

- Developed an interface to ease the integration of simulation and visualization modules & optimize memory usage
- Highly scalable parallel volume rendering, particle rendering, and image compositing
- Visualization time is less than 1% of simulation time when performed every 10th time step (based on the experimental results with 15,360 cores, 1620x1280x320 volume size, and 1024x1024 image size on JaguarPF at ORNL)

Selected zoomed-in views of mix rendering of volume and particle data (volume variable CH2O and particle variable HO2)

Concurrent analysis will enable robust tracking of features with short time-scales
Combining In-situ and In-transit Processing to Enable Extreme-Scale Scientific Analysis (J. Bennett, H. Abbasi, P-T Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, H. Yu, F. Zhang, and J. Chen, submitted for review.)
Combining In-situ and In-transit Processing to Enable Extreme-Scale Scientific Analysis (J. Bennett, H. Abbasi, P-T Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, H. Yu, F. Zhang, and J. Chen, SC’14.)
Take aways about Exascale

- **Motivation**
  - Many sources of machine performance variation (e.g. power management, failure recovery, congestion....)
  - Many sources of algorithmic variation
  - Difficult to coordinate 1 million processors to do the same thing simultaneously (bulk synchronous)

- **Value Proposition**
  - Describe task dependencies and have the computer handle the complex scheduling (including in situ analytics/viz)
  - Reduces workload on user to manage the scheduling (we use bulk sync because it requires no thought...)

- **Implementations**
  - OCR: Intel’s open community runtime (serves multiple impls.)
  - Legion
  - Charm++: Pre-dates MPI < and many others >
Combustion Science at Relevant Conditions

- **Homogeneous Charge Compression Ignition (HCCI engine combustion)** – ‘Chemical’ engine with high diesel-like efficiency without NOx and soot, tailor the charge stratification to control ignition and burn rate
- **Turbulent Jet Flames (Swirl, transverse, cavity)** – low-swirl Injector gas turbines with staged lean premixed combustion, flame stabilization, emissions
- **Lifted Diesel Jet flames** – lifted autoignitive diesel jet flame stabilization with multi-stage ignition fuels
- **Include UQ with respect to chemistry and transport properties**
Acknowledgments: Center for Exascale Simulation of Combustion in Turbulence (ExaCT)

Michael Bauer
Pat McCormick
Zhihao Jia
Sam Guiterrez
Elliott Slaughter
Galen Shipman
Sean Treichler
Wonchan Lee
Alex Aiken
Yuki Minamoto
Ankit Bhagatwala
Ramanan Sankaran
John Bell
Ray Grout
Marc Day
Cy Chan
Didem Unat
Scott Klasky
Hasan Abbasi
Valerio Pascucci
Timo Bremer

www.exactcodesign.org