Los Alamos
NATIONAL LABORATORY
EST.1943

Delivering science and technology
to protect our nation
and promote world stability

NNSA
National Nuclear Security Administration
Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Confluence on the path to exascale?

Galen M. Shipman

LA-UR-16-28559

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

National Strategic Computing Initiative (NSCI)

- Create systems that can apply **exaflops of computing power to exabytes of data**
- Keep the United States at the forefront of HPC capabilities
- Improve HPC application developer productivity
- Make HPC readily available
- Establish hardware technology for future HPC systems

Exascale Computing Project

DOE is a lead agency within NSCI with the responsibility that the DOE Office of Science and DOE National Nuclear Security Administration will execute a joint program focused on advanced simulation through a **capable** exascale computing program emphasizing **sustained performance** on relevant applications.

ECP Goals

- Develop a broad set of modeling and simulation applications that meet the requirements of the scientific, engineering, and nuclear security programs of the Department of Energy and the NNSA
- Develop a productive exascale capability in the US by 2023, including the required software and hardware technologies
- Prepare two or more DOE Office of Science and NNSA facilities to house this capability
- Maximize the benefits of HPC for US economic competitiveness and scientific discovery

Exascale will be driven by application needs and the demands of changing technology

Exascale challenges for future application capability

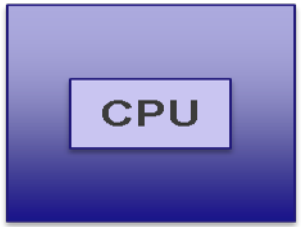
– Performance and productivity at extreme-scale

– Agile response to new scientific questions; integrating new physics

Change is driven by computing technology evolution: growth in scale, and node complexity

- **Massive parallelism of many-core/GPU nodes**
  - Leads to a push away from bulk synchrony
  - Task- and data-parallel programming models
- **Deep memory hierarchies (on node)**
  - Cache and scratchpad management
  - Challenge of spatial complexity in codes
  - *Need to get granularity of the tasks right*
- **Extreme scales**
  - Power, load balance, and performance variability
  - Reliability and resilience
  - Data management, and data analysis

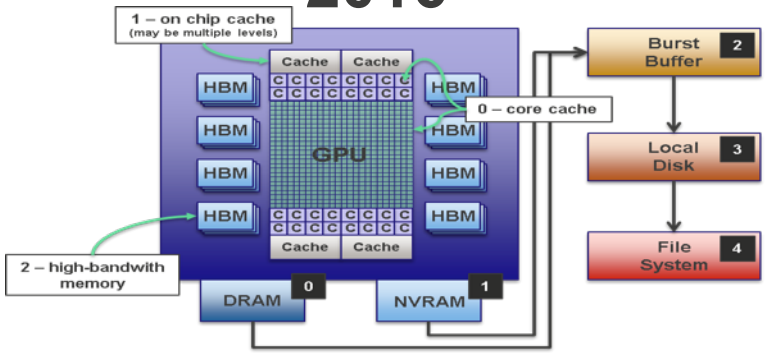
Common theme: methods that can tolerate latency variability within a node and across an extreme-scale system



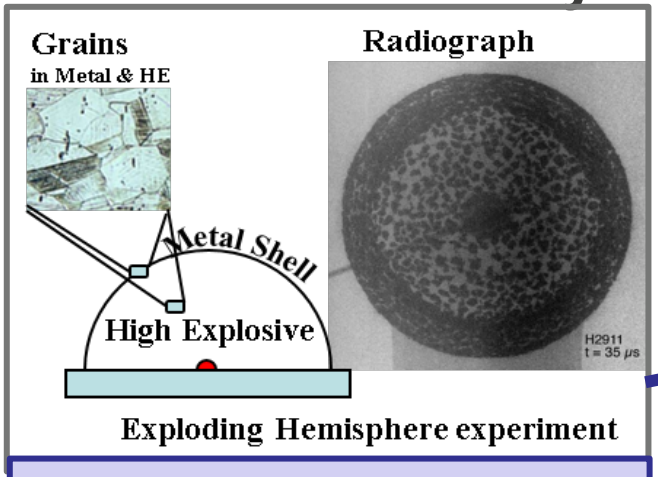
1996

The complexity of node architecture that applications must consider to make effective use of the system has increased significantly

2016



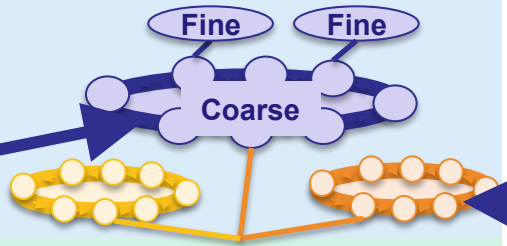
LANL's Next Generation Code (NGC):
Multi-Physics simulation at exascale



Resolving grain-level physics: improved fidelity in experiment (DARHT, MaRIE) *and* simulation

- Models at different scales (*fine* to *coarse*) & bridging between them (*multi-scale methods*)
- Coarse: multi-physics coupling
- Fine: higher fidelity *and* asynchronous concurrency

Diverse questions of interest: diverse physics topologies



Control & state manager

Asynchronous MPI + threads

MPI + threads

Legion

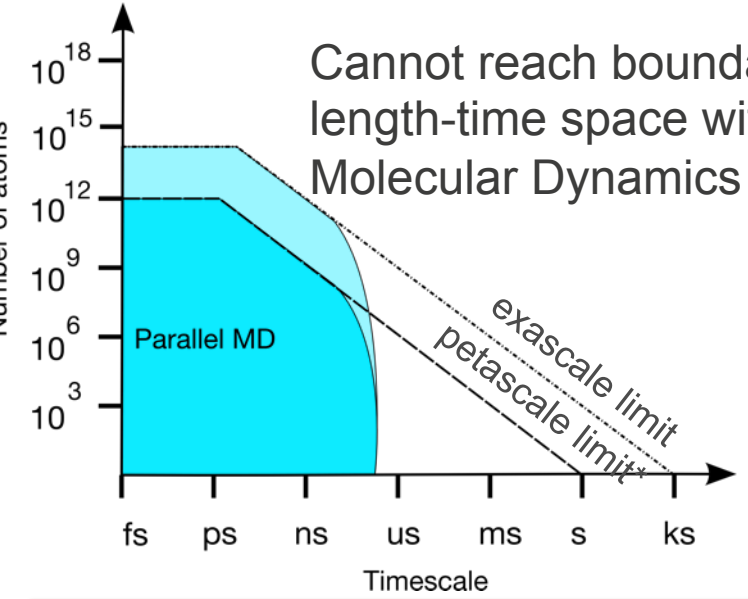
Diverse architectures

Common theme at exascale: need for asynchronous methods tolerant of latency variability within a computational node, and across an extreme-scale system

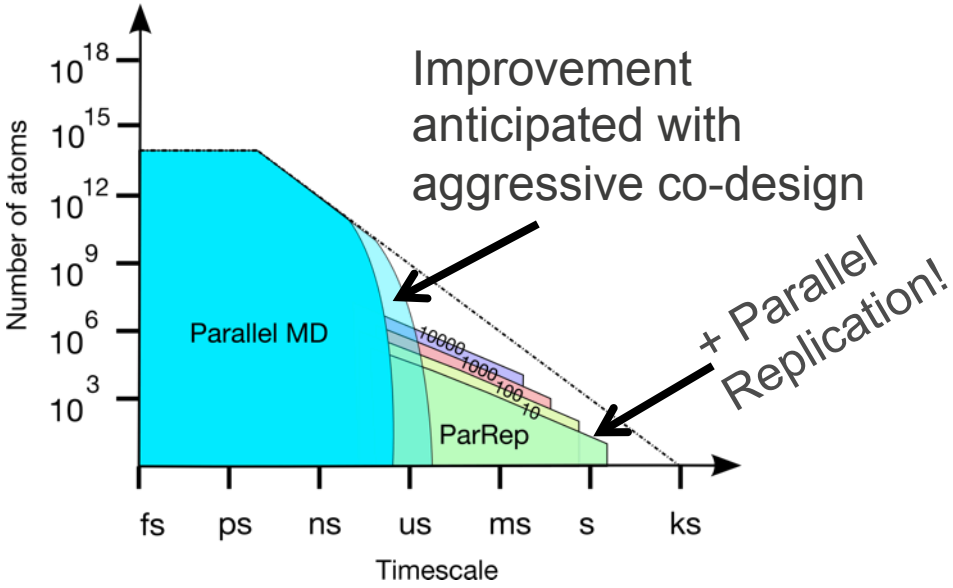
- **Traditional physics and CS methods (operator split, MPI) have poor asynchrony**
- **New programming models expose more parallelism for asynchronous execution**

Building leadership in computational science from advanced materials to novel programming models

EXascale Atomistics for Accuracy, Length and Time



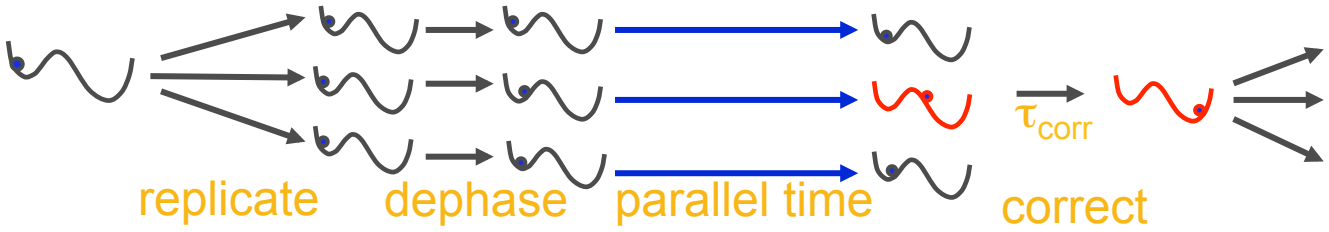
Cannot reach boundaries of length-time space with Molecular Dynamics alone.



Improvement anticipated with aggressive co-design

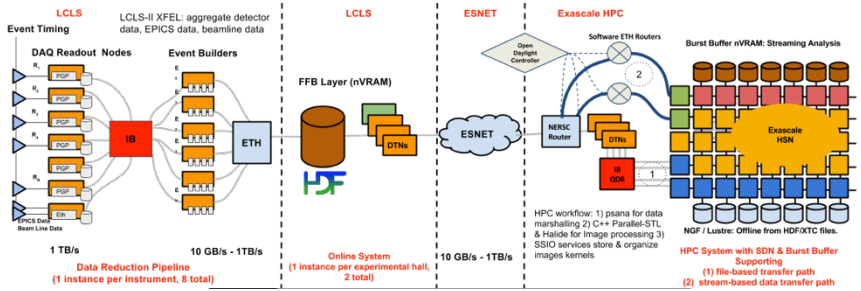
+ Parallel Replication!

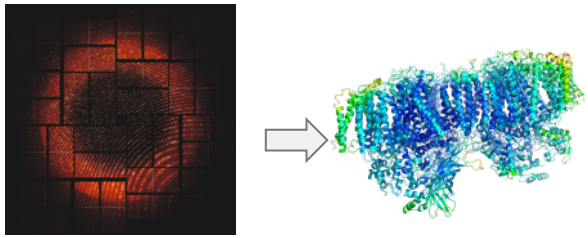
ParSplice - parallel replica dynamics using **data driven asynchronous tasks**



Data Analytics at the Exascale for Free Electron Lasers (ExaFEL)

- Perform prompt LCLS data analysis on next generation DOE supercomputers
- LCLS will increase its data throughput by three orders of magnitude by 2025
- Enabling new photon science from the LCLS will require near real-time analysis (~10 min) of data bursts, requiring burst computational intensities exceeding an exaflop





From detected signal to a model of the sample

- High-throughput analysis of individual images
- Ray-tracing for inverse-modeling of the sample
- Requires data-driven asynchronous computation
  - A distributed task-based runtime

These applications require a data-aware asynchronous programming environment

Legion: a data-aware task based programming system

Tasks
(execution model)

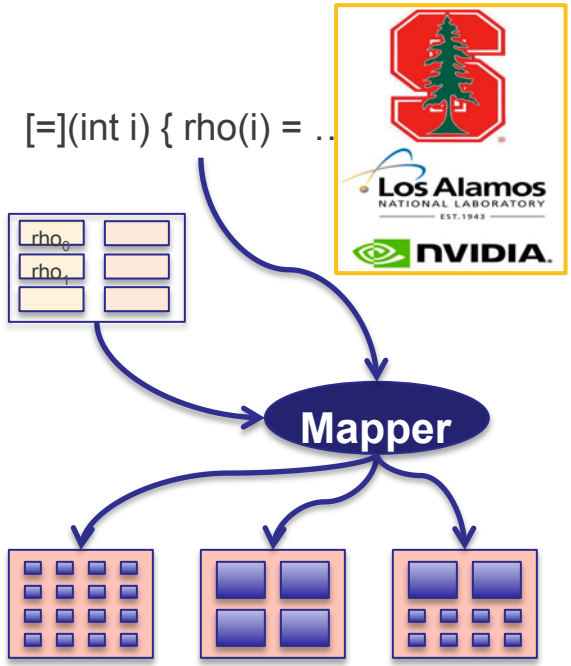
Regions
(data model)

Mapper

Describe parallel execution elements and algorithmic operations
Sequential semantics, with out of order execution, in order completion.
Describe decomposition of computational domain, and

- **Privleges** (read-only, read+write, reduce)
- **Coherence** (exclusive, atomic, etc.)

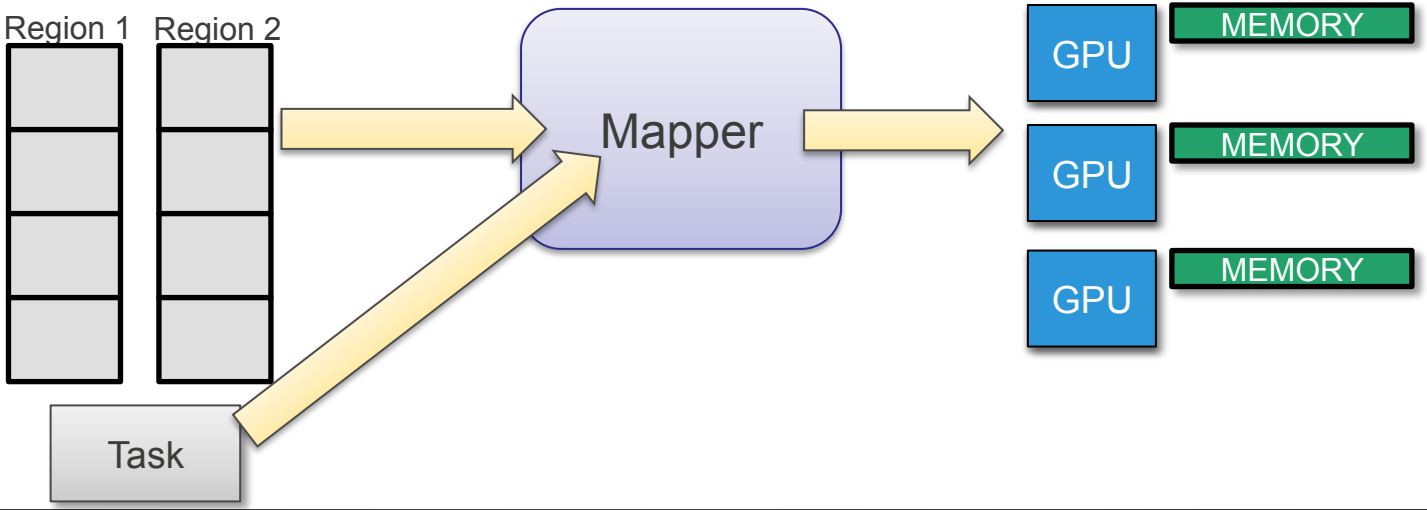
Describes how tasks and regions should be mapped to the target architecture



Mapper allows architecture-specific optimization without affecting the correctness of the task and domain descriptions

Mapping Tasks and Data to Hardware Resources

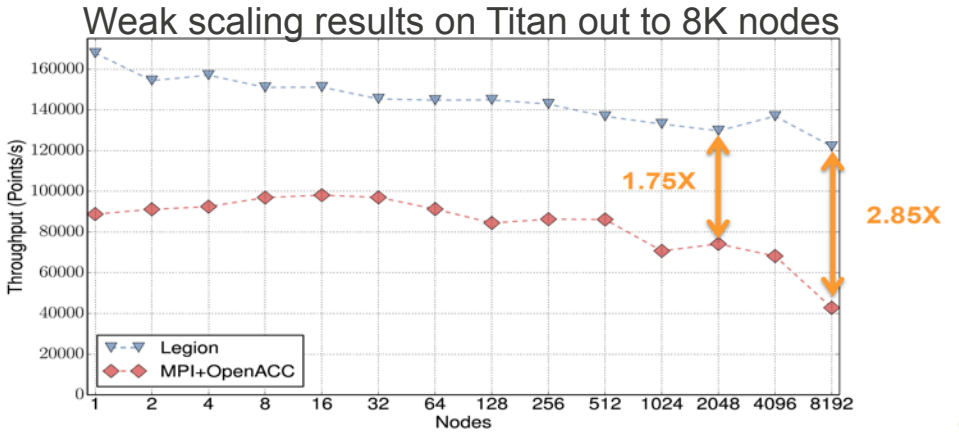
- **Application selects:**
  - Where tasks run and where regions are placed
  - Computed dynamically
  - Decouple correctness from performance

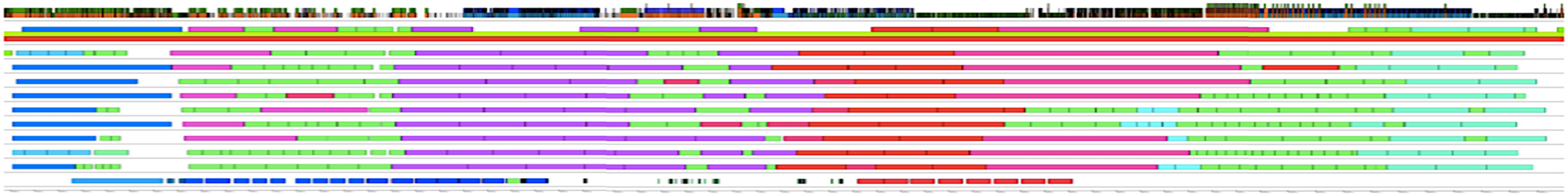


Can a new programming system address the needs of simulation, analysis, workflow, and “big data”?

Simulation: Legion S3D Execution and Performance Details

- **Mapping for 96^3 Heptane**
  - Top line shows runtime workload
  - Different species required mapping changes (e.g., due to limited GPU memory size) – i.e. tuning is often not just app and system specific…





Analysis: A Unified Approach for Programming *in situ* Analysis & Visualization

Challenge:

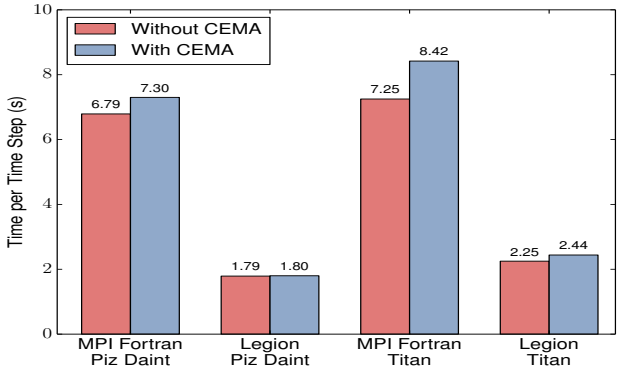
- Data produced by applications is too large for post-processing, so need in situ analysis and visualization.
- In situ processing works best when tightly coupled with applications in order to avoid unnecessary data movement and copies and to share compute resources between the application and in situ.
- Manual data mapping and task scheduling impacts application portability and productivity.

Approach: Use data-centric programming approach to scheduling and mapping between application and in situ

- Legion runtime developed as part of the ExaCT Co-Design Center. http://legion.stanford.edu
- Promotes data to a first-class programming construct
- Separates implementation of computations from mapping to hardware resources
- Implement data transformation and sublinear algorithms as well as visualization pipeline abstractions

Results:

- **Flexible data-driven tasking model reduced overhead** of in situ calculations by a factor of 10
- **Time-to-solution improved by 9x** and obtains **over 80% of the achievable performance** on Titan and Piz Daint.
- **Enabled building blocks for new science**: first large-scale 3-D simulation of a realistic primary reference fuel (PRF) blend of iso-octane and n-heptane, involving *116 chemical species and 861 reactions*



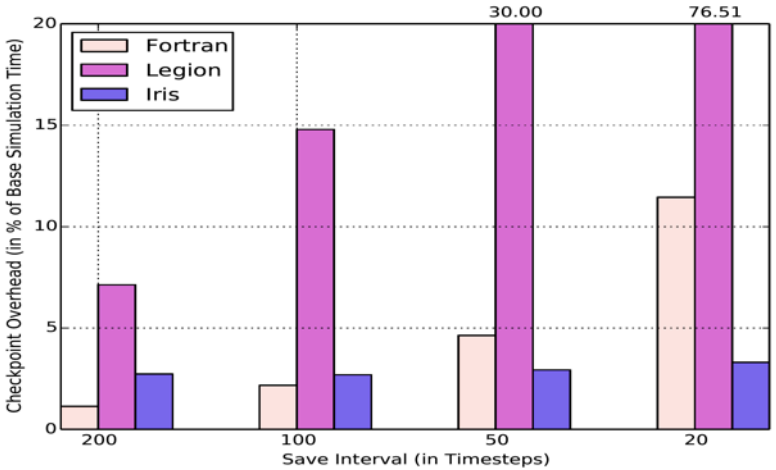
In situ Chemical Explosive Mode Analysis (CEMA).

Flexible scheduling and mapping reduces analysis overhead to less than 1% of overall execution time.

Additional benefits from improvement in overall application performance.

Workflow: Integration of External Resources into the Programming Model

- **We can't ignore the full workflow!**
  - Amdahl's law sneaks in if we consider I/O from tasks – 15-76% overhead vs. 2-12% of original Fortran code!
- **Introduce new semantics for operating with external resources (e.g. storage, databases, etc.).**
  - Incorporates these resources into deferred execution model
  - Maintains consistency between different copies of the same data
  - Underlying parallel I/O handled by HDF5 but scheduled by runtime
- **Allow applications to adjust the snapshot interval based on available storage and system fault concerns instead of overheads.**

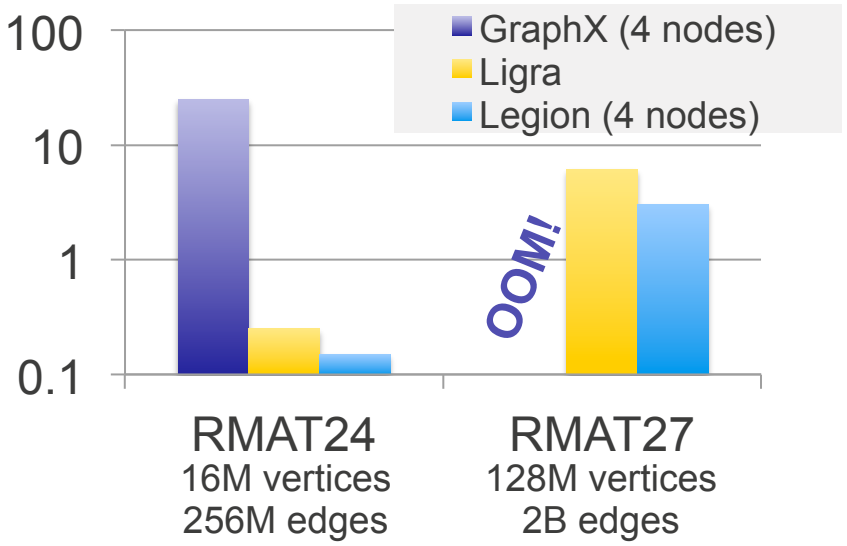


Performance of S3D checkpoints running on 64 nodes (i.e., 1,024 cores) of Titan.

THANKS OLCF!

Big Data: Exploring the use of Legion in Graph Applications

- **Big Data: High frequency, massive, and irregular data access**
  - E.g., Graph traversal randomly accesses vertices
  - Need **deep memory hierarchy** to achieve both performance and scalability
- **Research interests is shifting to heterogeneous systems**
  - E.g., GPUs, CPUs, and FPGAs
  - Need for a generic runtime that coordinates different processors
- **PageRank: a graph-based application widely used to sort webpages**
  - Legion CPU version: 600 lines of code
  - Legion GPU version: 160 lines of code
  - Legion Mapper: 260 lines of code



GraphX: A distributed graph engine on top of Spark
Ligra: State-of-the-art shared-memory graph engine

- Exascale should push us to rethink how we program applications
  - Data-aware: to maximize available throughput in a complex distributed hierarchy of memories
  - Declarative: Describing what needs to be computed rather than precisely how it needs to be computed on a complex system
  - Runtime assisted: Greater reliance on runtime mechanisms to efficiently schedule computation and data movement
- Big-data has embraced many of these concepts
- Will we see a confluence of approaches?

Acknowledgements

Alex Aiken, Michael Bauer, Ben Bergen, Timo Bremer, Jacqueline Chen, David Daniel, Kei Davis, Mattan Erez, Charles Ferenbaugh, Sam Gutierrez, Zhihao Jia, Quincey Koziol, Yongkee Kwon, Wonchan Lee, Carlos Maltzahn, Pat McCormick, Nick Moss, Scott Pakin, Rob Ross, Christine Sweeney, Brad Settlemyer, Elliot Slaughter, Sean Treichler, Noah Watkins and many more…