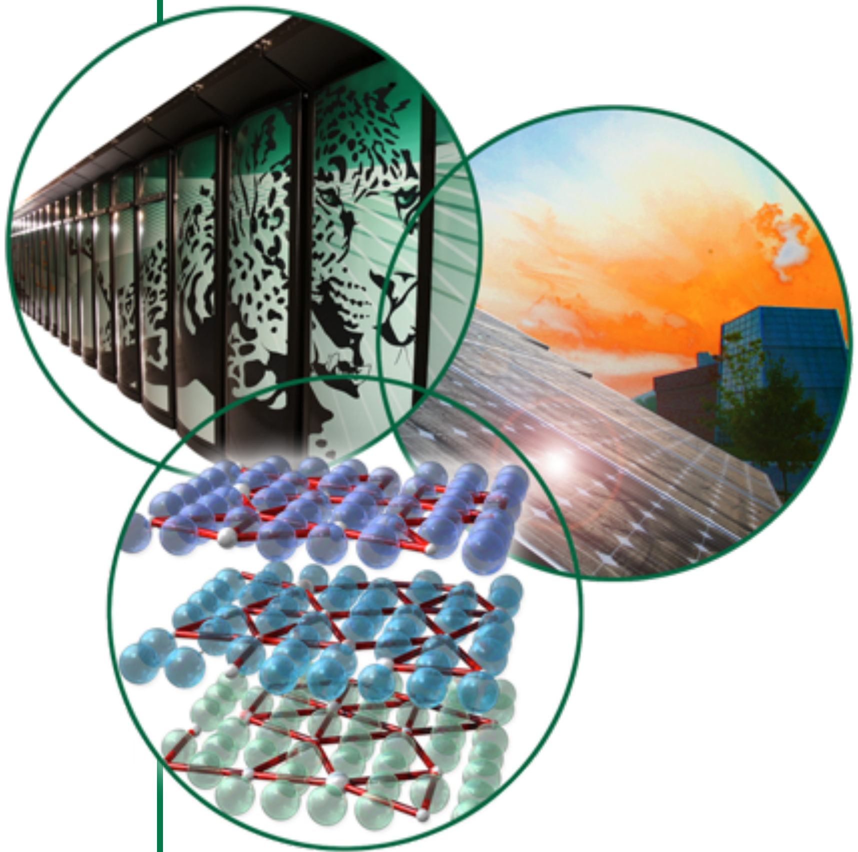
ALGORITHM-ARCHITECTURE CO-DESIGN

Does it Work? Five Years of Experience



Al Geist
Oak Ridge National Laboratory

SOS 18

St. Moritz, Switzerland
March 17-20, 2014

U.S. DEPARTMENT OF ENERGY

Research supported by DOE ASCR

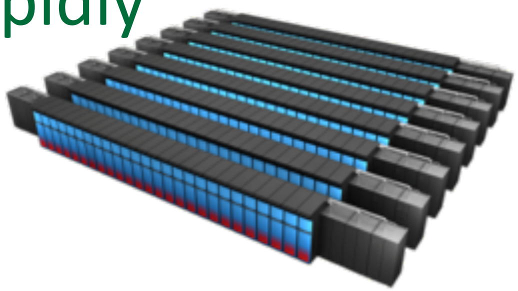
OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Our Goal Aligns Well with SOS 18 Theme

This talk will focus on our research goal of:
Closing the "application-architecture performance gap"

The difference between the peak performance of a system and the performance achieved by real science applications.

This gap continues to get wider due to rapidly increasing parallelism, and increasing heterogeneity of the hardware



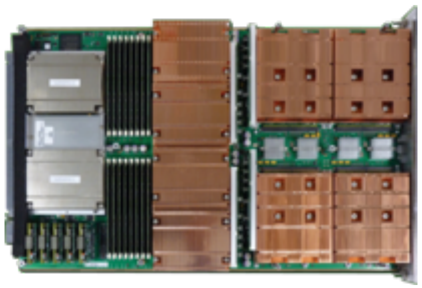
 Managed by UT-Battelle
for the U.S. Department of Energy

OAK RIDGE National Laboratory

Heterogeneous Architectures, example ORNL's "Titan" System

CRAY SYSTEM SPECIFICATIONS:

- Peak performance of 27.1 PF
  - 24.5 GPU + 2.6 CPU
- 710 TB total system memory
- 8.8 MW peak power
- Space 4,352 ft^2 (404 m^2)
- 200 Cabinets

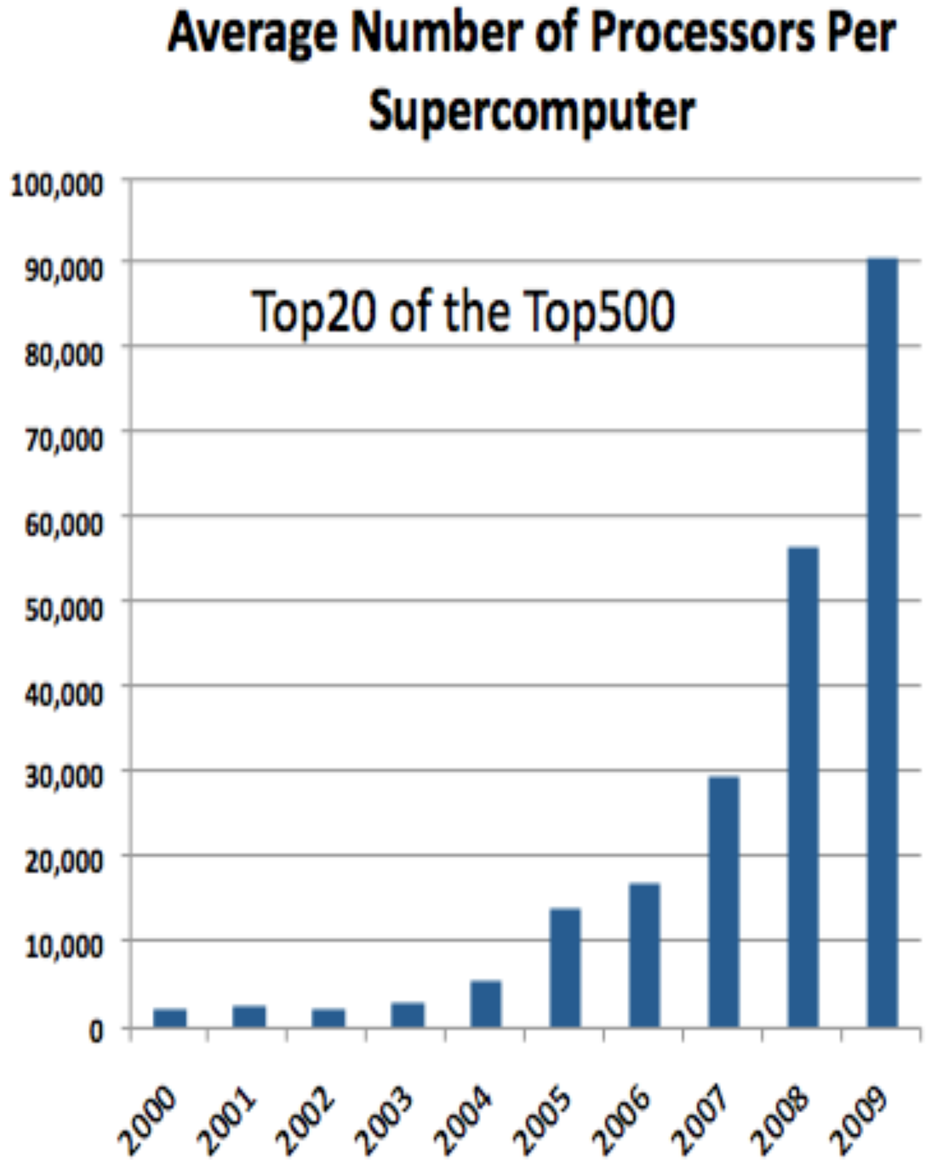


- 18,688 Compute Nodes each with:
  - 16-Core AMD Opteron CPU
  - NVIDIA Tesla "K20x" GPU
  - 32 + 6 GB memory
- 512 Service and I/O nodes
- Cray Gemini 3D Torus Interconnect

Challenges of Increasing Parallelism

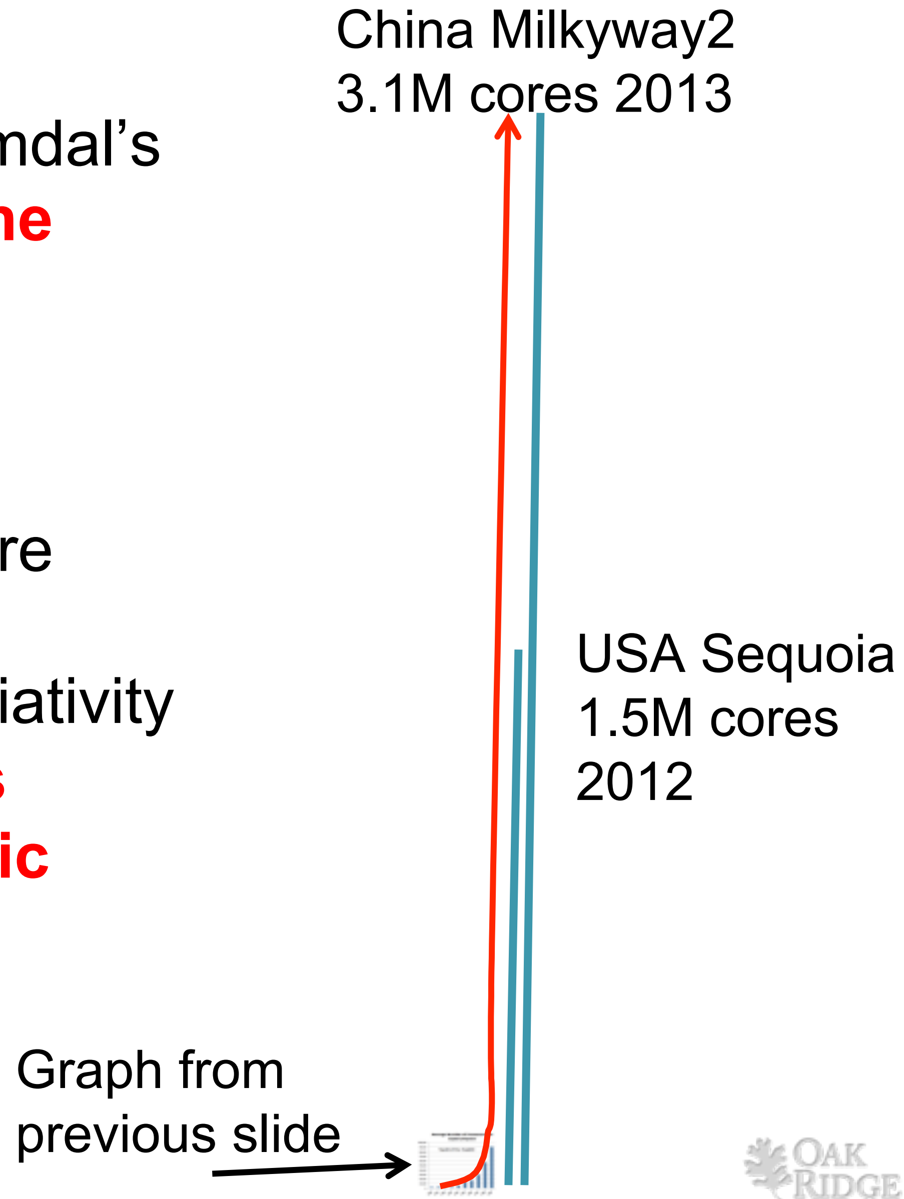
Challenges

- Fundamental assumptions of applications and system software design did not anticipate **exponential growth in parallelism**
- Mean time between failures of components is proportional to number of components. (Designed FIT rate constant)
- **Undetected error rates increasing** assuming they are a fixed percentage of detected errors.
- **Increased danger of wrong answers**



Exponential Growth of Parallelism

- Growth of parallelism and Amdal's Law is a **leading driver of the application-architecture performance gap**.
- To avoid synchronization algorithms are becoming more asynchronous and by not maintaining numerical associativity (among other things) **results becoming less deterministic**



Five Years of Algorithm-Architecture Co-Design (3 projects I've led)

The IAA Algorithms Project (ORNL-SNL) begun in FY2009
Focused on homogeneous multi-core systems, and extreme scale system simulations. Hierarchical programming models

EASI Joint Math/CS Institute begun in FY2010
Focused on heterogeneous systems with accelerators and application resilience. Hybrid programming models

EASIR RX-Solvers Project begun in FY2013
Heterogeneous systems. Communication reducing and resilient algorithms. Numerical Reproducibility

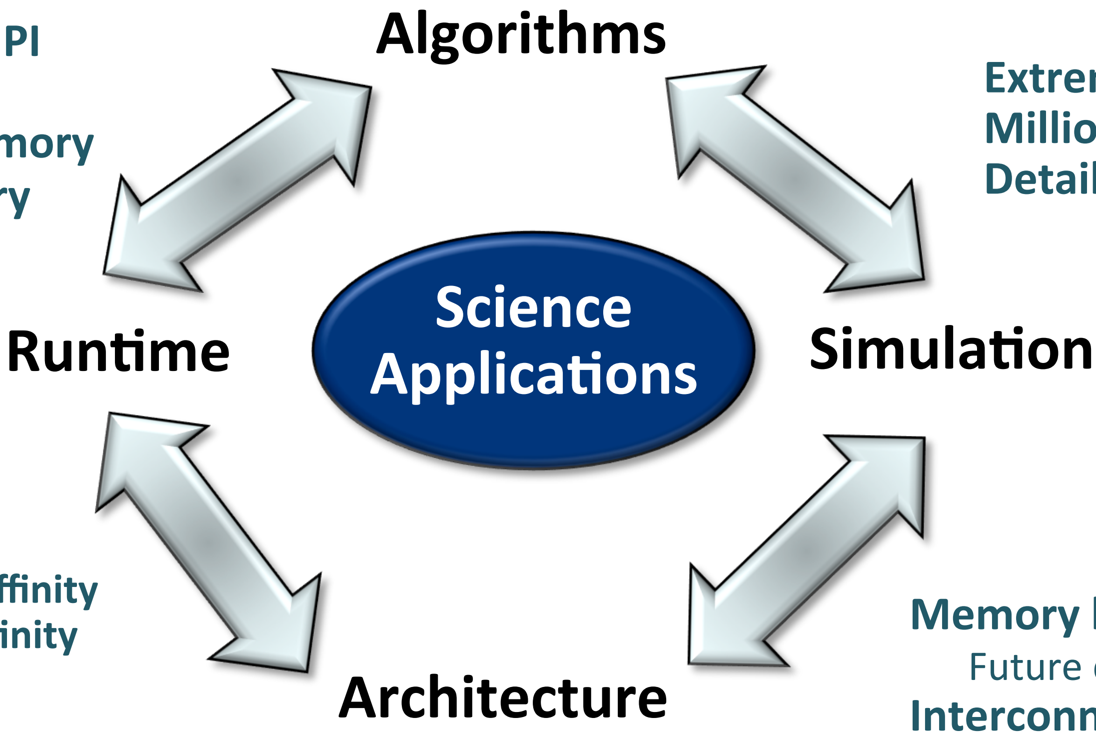
Our Co-Design Model
It all revolves around the science

Multi-core Aware
Communication reducing

Hybrid Node Aware, Multi-precision
Fault Tolerant, Reproducibility

Hierarchical MPI
Resilient MPI
Persistent Memory
Shared memory

Extreme Scale
Million node systems
Detailed Node level



Multi-core
- **Processor affinity**
- **Memory affinity**
- **Scheduling**
- **Threading**

Memory hierarchy
- Future designs

Interconnect
- Latency/BW effects

Influence design

EASI Project Overview
Addressing Heterogeneity and Resilience

Research Areas in Institute

Applications
HOMME
MADNESS
Charon

Algorithms
Krylov
Poisson
Helmholtz
Dense NLA
BLAS

- Heterogeneous programming API
- Robust multi-precision algorithms
- Hybrid programming
- Resilient algorithms
- Communication optimal algorithms
- Auto-tuned BLAS (API)
- New parallelization methods

Runtime
MPI
Shared-memory
Processor affinity
Memory affinity

- Task placement and scheduling
- Memory management
- Architecture-aware MPI

Architecture **Heterogeneous, multi-core, extreme-scale**

Community Outreach

Deliver codes to community through:
ScaLAPACK
Trilinos
Open MPI
MPICH2
MADNESS
HOMME

Workshops
Training
Publications

EASIR Project Team
Extreme-scale Algorithms & Solver Resilience

Architecture-aware Algorithms for Scalable Performance and Resilience on Heterogeneous Architectures

PI: **Al Geist (ORNL)**

Co-PIs:

Mike Heroux (SNL)
Bill Gropp (U ILL)
Jack Dongarra (UTK)
Jim Demmel (UC Berkeley)
Clayton Webster (ORNL)

Extreme-scale Algorithms & Solver Resilience (EASIR) Research Project

Solver algorithms for extreme-scale heterogeneous systems

Advances in Solvers

Communication optimal algorithms

Synchronization reducing algorithms

- Latency hiding
- Minimizing data movement
- Variable precision arithmetic

Resilient Algorithms

Revolutionary methods for algorithm resilience

- Ability to survive silent errors
- Tunable reproducibility
- Local persistent storage
- Hierarchical Schwarz framework

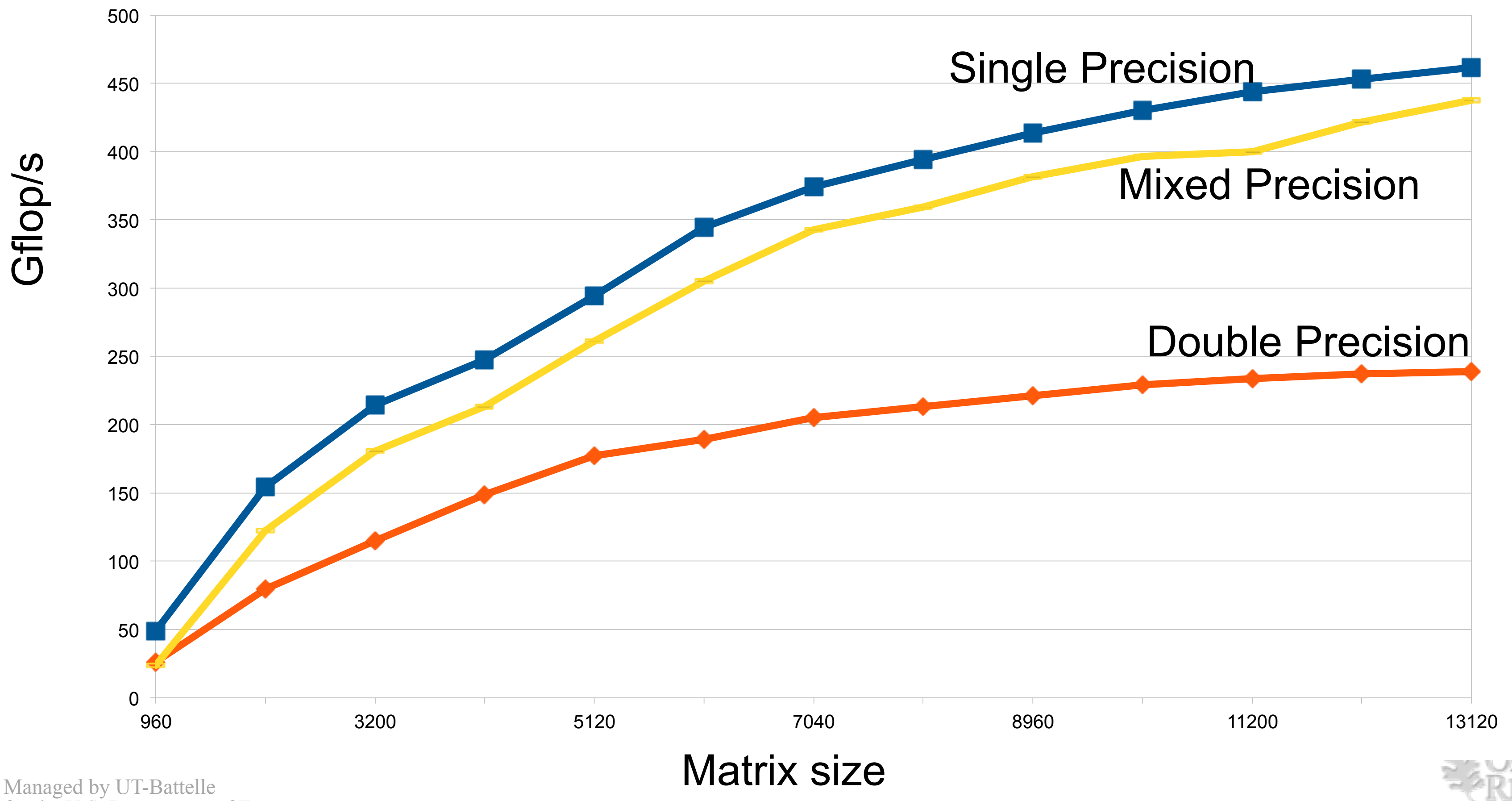
Develop robust multi-precision algorithms

Idea Goes Like This…

- **Exploit single precision floating point as much as possible.**
  (Single precision is faster than double precision because
  - Higher parallelism within floating point units
    - 4 ops/cycle (usually) instead of 2 ops/cycle
  - Reduced data motion
    - 32 bit data instead of 64 bit data
  - Higher locality in cache
    - More data items in cache
- **Correct or update the solution with selective use of 64 bit floating point to provide a refined results**
- **Intuitively:**
  - Compute a 32 bit result,
  - Calculate a correction to 32 bit result using selected higher precision and,
  - Perform the update of the 32 bit results with the correction using high precision.

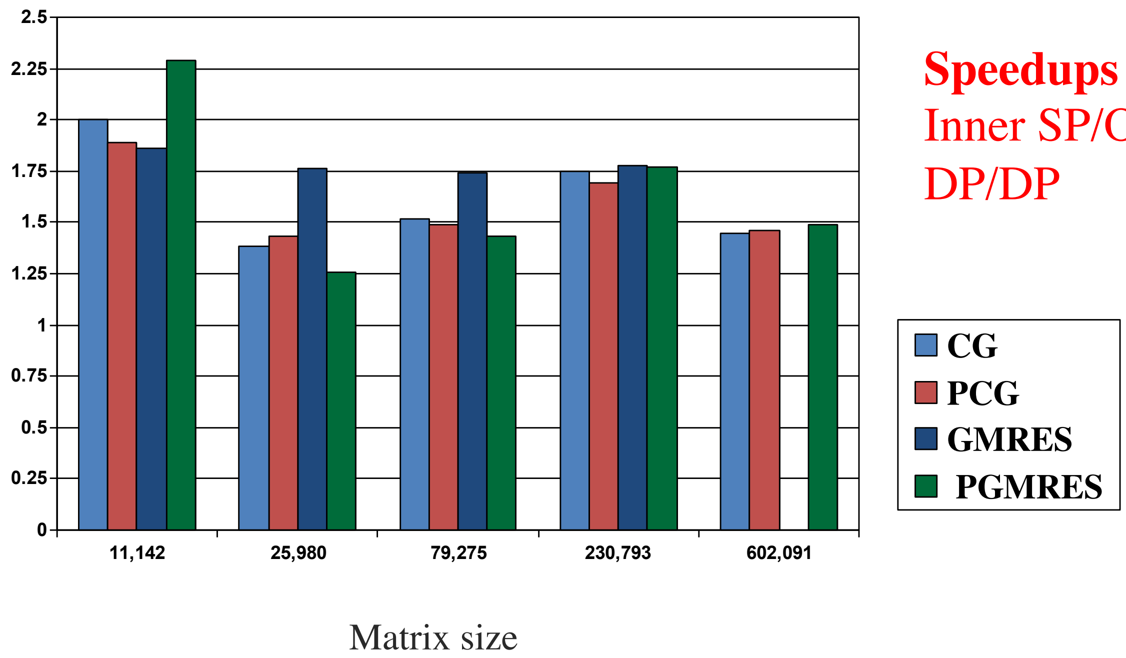
Results for Multi-precision Iterative Refinement for Dense Matrix *Ax = b*

Tesla C2050, 448 CUDA cores (14 multiprocessors x 32) @ 1.15 GHz.,
3 GB memory, connected through PCIe to a quad-core Intel @2.5 GHz.



Highlight: Trilinos Library incorporates Multi-Precision Algorithms for Sparse Matrix Solvers

Trilinos is an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. The latest release utilizes C++ templates to allow users to mix precisions in their solvers.



Speedups for mixed precision up to 2X Inner SP/Outer DP (SP/DP) iter. methods *vs* DP/DP

New communication-avoiding orthogonalization method developed and added to Trilinos

Challenge

- Orthogonalization ften dominates iterative solve of Ax=b and Ax=λx
- Existing algorithms (Gram-Schmidt) communicate (move data) too much
- Communication much slower than floating-point arithmetic

New algorithm

- Tall Skinny QR factorization (TSQR)
- Communicates less & more accurate

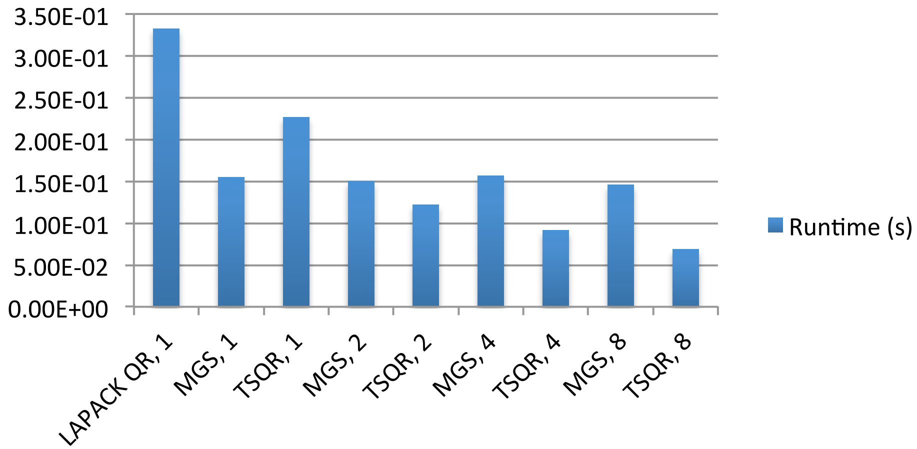
TSQR Implementation

- Inter- and intranode parallelism
- Memory hierarchy optimizations
- Generic on scalar data type

Performance comparison

- LAPACK QR (DGEQRF) -- sequential only
- Modified Gram-Schmidt (parallel -- Intel TBB)
- TSQR (parallel -- Intel TBB)

Run time in seconds of different methods shows 3X-10X improvements with TSQR



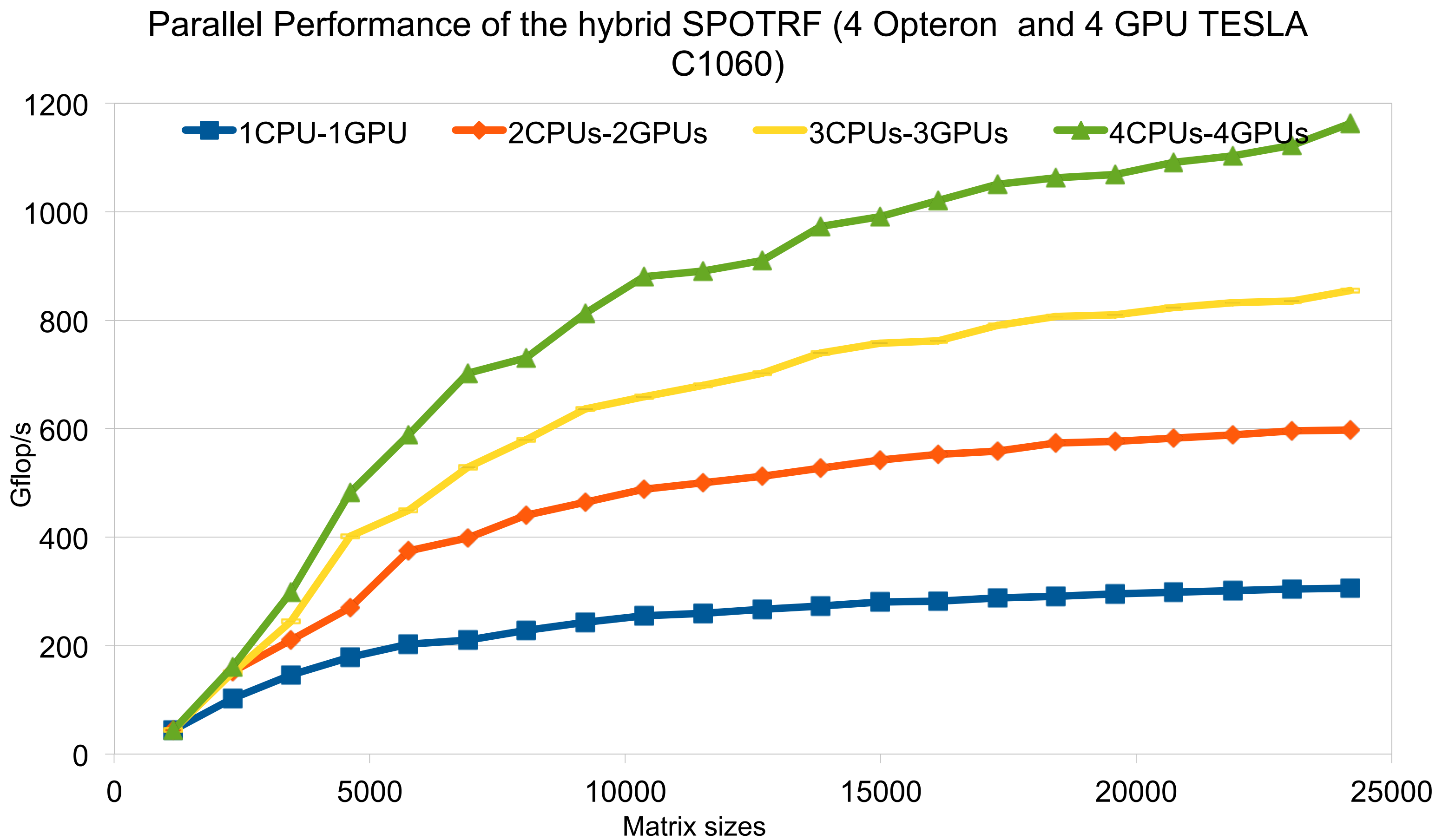
Highlight:
Developed heterogeneous programming API

- Completed a portable API for multicore CPUs and GPUs.
- Allows writing portable parallel linear algebra software that can use pthreads, OpenMP, CUDA, or Intel TBB (even more than one within the same executable)
- API is extensible to other programming models as needed.
- Using the API, we demonstrated compiling and running the same software kernel using pthread, Intel Threading Building Blocks and CUDA.
- The Trilinos Tpetra and Kokkos packages incorporate this API in Trilinos 10.0.
- **The API is documented in** http://www.cs.sandia.gov/~maherou/docs/TrilinosNodeAPI.pdf

Developed new High Performance Cholesky Factorization for Multicore with GPU nodes

Dense solvers for multicore/GPUs – MAGMA Project

MAGMA - based on LAPACK and extended for hybrid systems (multi-GPUs + multicore)



More details are available at: http://www.netlib.org/lapack/lawnspdf/lawn223.pdf

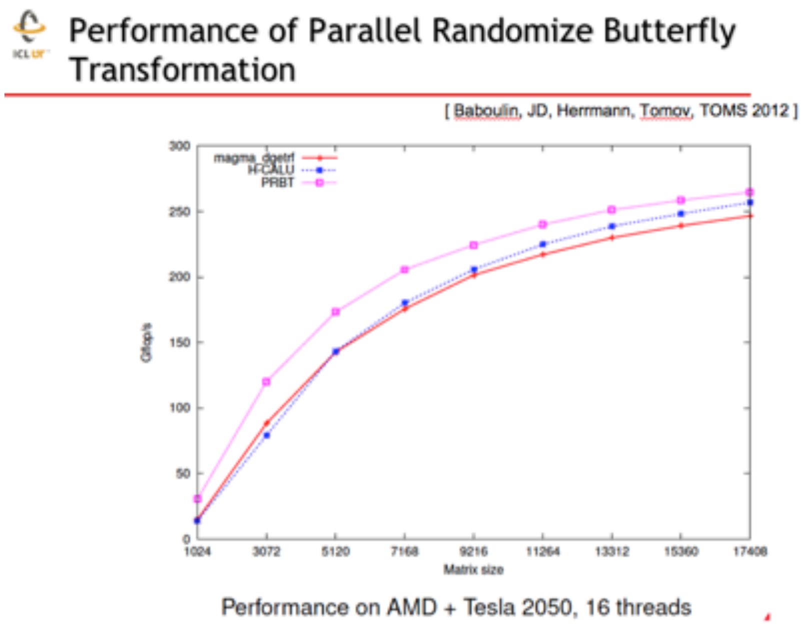
Highlight: New Hybrid Node LU Algorithm

Objectives

- Develop a scalable, high performance solver that outperforms other algorithms on hybrid CPU +GPU nodes such as those on Titan.
- Avoid the need for pivoting by randomly permuting the dense matrix in parallel before starting factorization.

Impact

- New LU algorithm is nearly as fast as Cholsky
- Plan to distribute in the PLASMA software library
- By avoiding pivoting the amount of data moved is reduced lowering power consumption during factorization.



Accomplishments

- Using a parallel randomized butterfly transformation reduces the probability of getting a zero pivot to essentially zero.
- Iterative refinement used to achieve full precision

- Paper in TOMS
  Baboulin, JD, Herman, Tomov, **TOMS** 2012

Contact: Jack Dongarra <dongarra@eecs.utk.edu>

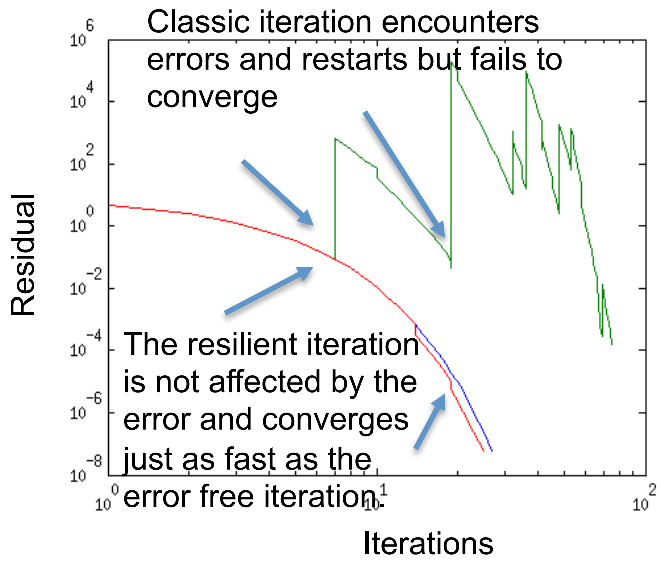
Highlight – New Resilience Method for Soft Errors including Proof of Correctness

Objectives

- To Create the next generation extreme scale algorithms that can produce reliable results, even when executed on unreliable hardware
- Even in the presence of multiple silent errors, the resilient algorithm converges at the same rate as the fault free iteration.

Impact

- Developed a fully resilient fixed-point iteration that can be used to improve a number of existing solvers including Jacobi, Gauss-Seidel, GMRES, etc.
- Demonstrated how "selective reliability" can be used to control the propagation of hardware error



Accomplishments

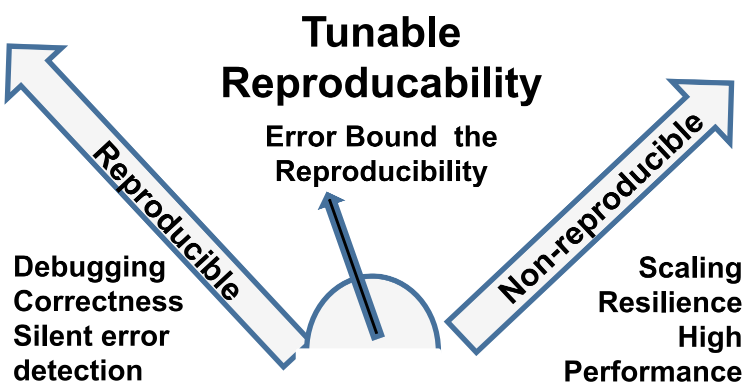
- We provide rigorous mathematical definitions of hardware error and convergence, both with respect to silent hardware faults
- Paper: M. Stoyanov and C. Webster, Numerical Analysis in the Presence of Hardware Faults: Fixed Point Methods, ***SIAM Journal of Scientific Computing***, 2014 (Submitted).

Contact: Clayton Webster <webstercg@ornl.gov>

Highlight – Release of Reproducable and Resilient Packages

Release of Reproducable BLAS.

This capability can often be very important in debugging and performance tuning.



Release of a resilient algorithms in Scalapack.

The LU, QR, Cholesky, among others, algorithms in this release can survive any number of node failures as long as they don't happen more than one at a time.

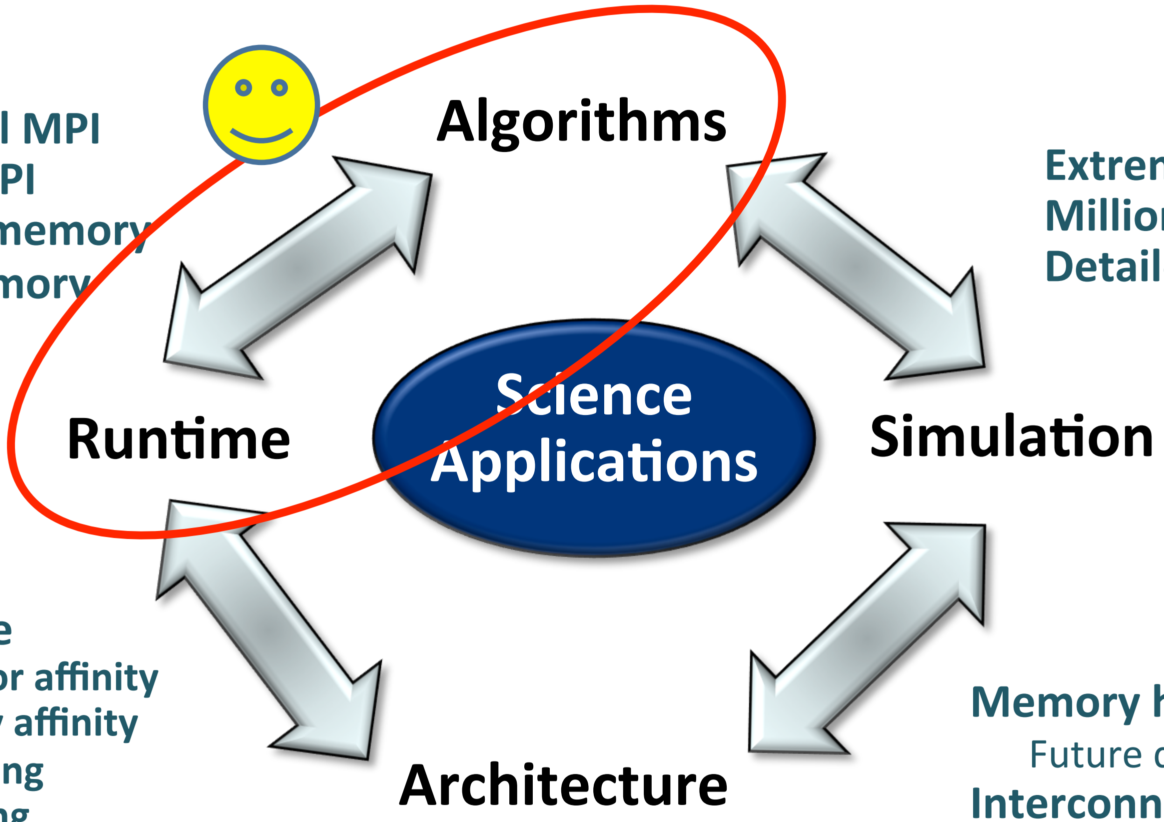
The overhead for this resilient implementation is 3% plus an additional 3% to recover from each fault.

Summary: Co-Design What Has Worked

Multi-core Aware
Communication reducing

Hybrid Node Aware, Multi-precision
Fault Tolerant, Reproducibility

Hierarchical MPI
Resilient MPI
Persistent memory
Shared memory



Extreme Scale
Million node systems
Detailed Node level

Multi-core
- **Processor affinity**
- **Memory affinity**
- **Scheduling**
- **Threading**

Memory hierarchy
- Future designs

Interconnect
- Latency/BW effects

Small Influence design

Thank You