Institute for Advanced Architectures and Algorithms

DOE IAA: Scalable Algorithms for Petascale Systems with Multicore Architectures

Al Geist and George Fann; ORNL
Mike Heroux and Ron Brightwell; SNL

Cray User Group Meeting
May 7, 2009

It's All About Enabling Science

Science is getting harder to solve on new supercomputer architectures and the trends are in the wrong direction.

Application Challenges *

- Scaling limitations of present algorithms
- Innovative algorithms for multi-core, heterogeneous nodes
- Software strategies to mitigate high memory latencies
- Hierarchical algorithms to deal with BW across the memory hierarchy
- Need for automated fault tolerance, performance analysis, and verification
- More complex multi-physics requires large memory per node
- Model coupling for more realistic physical processes
- Dynamic memory access patterns of data intensive applications
- Scalable IO for mining of experimental and simulation data

*** List of challenges comes from survey of HPC application developers**

Algorithms Project Goals

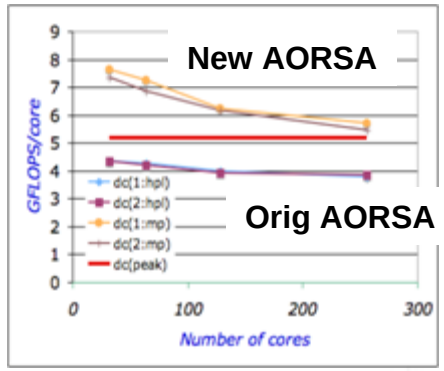
The Algorithms project goal is closing the "application-architecture performance gap" by developing:

Architecture-aware algorithms and runtime that will enable many science applications to better exploit the architectural features of DOE's petascale systems. Near-term high impact on science

Simulation to identify existing and future application-architecture performance bottlenecks. Disseminate this information to apps teams and vendors to influence future designs. Longer-term impact on supercomputer design

Much to be gained – Two recent examples

- **New Algorithms and muti-core specific tweaks give a tremendous boost to AORSA performance on Leadership systems.**



Exploiting features in the multi-core architecture

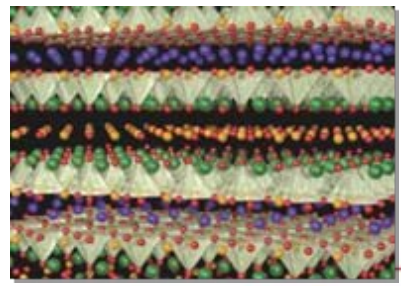
- Quad-core Opteron can do four flops per cycle/core
- Shared memory on node
- Multiple SSE units

New Algorithms

- Single precision numerical routines coupled with
- Double precision iterative refinement

Helps multi-core:
Doubles BW to socket
Doubles cache size
Doubles peak flop rate

- **New multi-precision algorithm developed for DCA++ more efficiently uses the multi-core nodes in Cray XT5**



- Science Application sustained 1.35 PF on Jaguar XT5
- Wins 2008 Gordon Bell Award

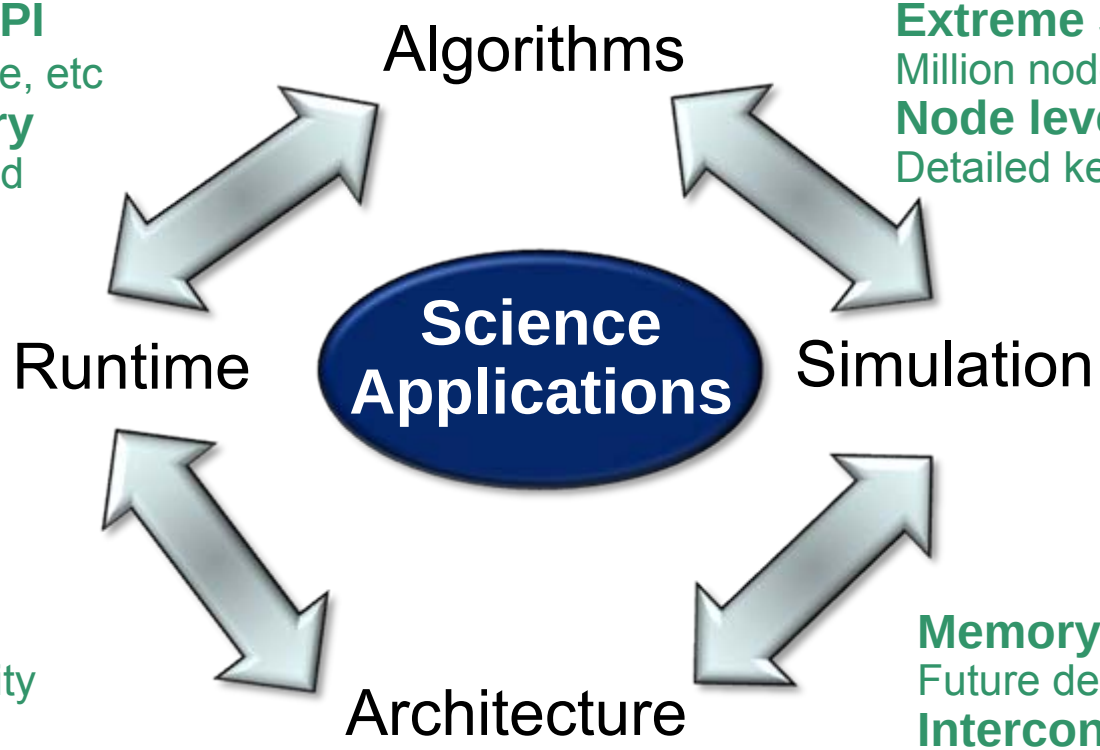
Algorithms Project Overview

It all revolves around the science

Multi-core Aware
Hybrid, Parallel in time

Multi-precision
Krylov, Poisson, Helmholtz

Hierarchical MPI
MPI_Comm_Node, etc
Shared memory
MPI_Alloc_Shared



Extreme Scale
Million node systems
Node level
Detailed kernel studies

Multi-core
Processor affinity
Memory affinity
Scheduling

Memory hierarchy
Future designs
Interconnect
Latency/BW effects

Influence design

Maximizing Near-term Impact

Architecture aware algorithms demonstrated in real applications providing immediate benefit and impact

- **Climate (HOMME)**
- **Materials and Chemistry (MADNESS)**
- **Semiconductor device physics (Charon)**

New algorithms and runtime delivered immediately to scientific application developers through popular libraries and frameworks

- **Trilinos**
- **Open MPI**
- **SIERRA/ARIA**

Technical Details

Architecture Aware Algorithms

Develop robust multi-precision algorithms:

- Multi-precision Krylov and block Krylov solvers.
- Multi-precision preconditioners: multi-level, smoothers.
- Multi-resolution, multi-precision solver fast Poisson and Helmholtz solvers coupling direct and iterative methods

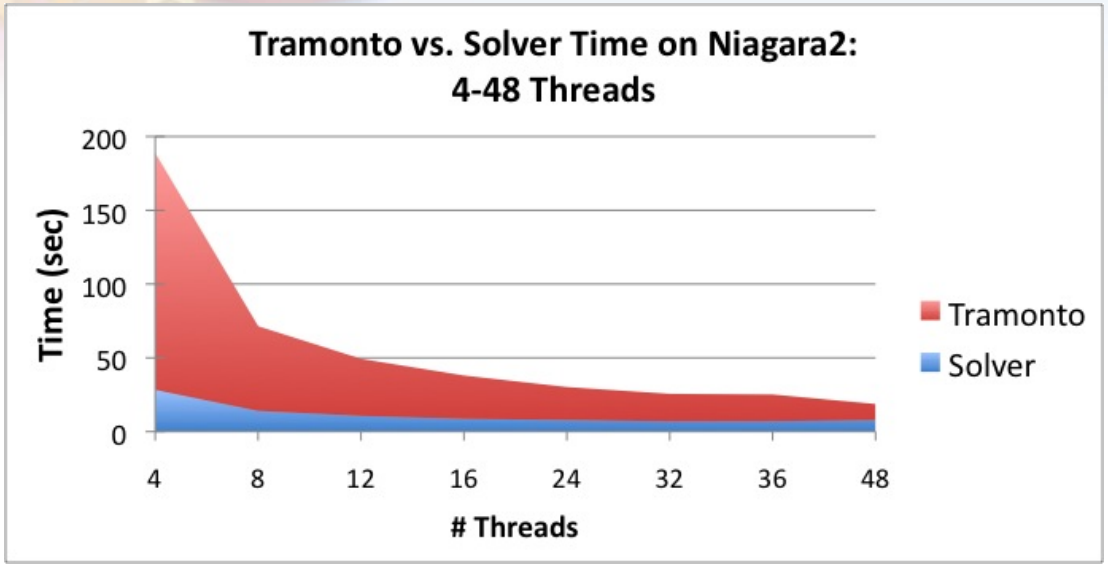
Develop multicore-aware algorithms:

- Hybrid distributed/shared preconditioners.
- Develop hybrid programming support: Solver APIs that support MPI-only in the application and MPI+multicore in the solver.
- Parallel in time algorithms such as Implicit Krylov Deferred Correction

Develop the supporting architecture aware runtime:

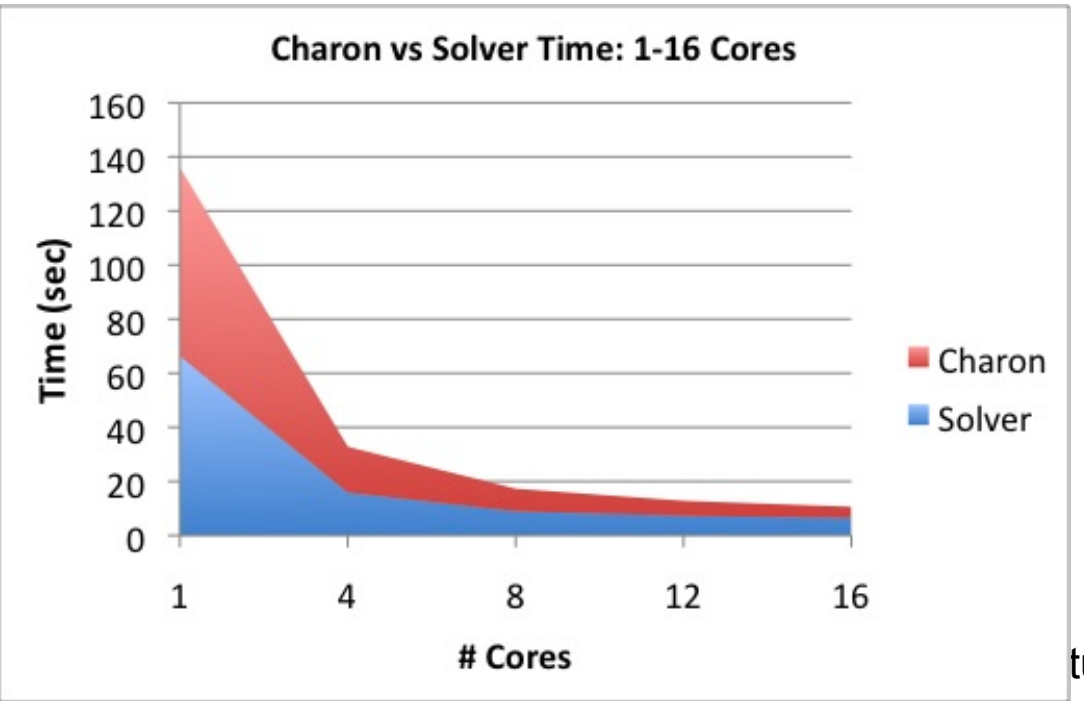
- Multi-level MPI communicators (Comm_Node, Comm_Net).
- Multi-core aware MPI memory allocation (MPI_Alloc_Shared).
- Strong affinity - process-to-core, memory-to-core placement.
- Efficient, dynamic hybrid programming support for hierarchical MPI plus shared memory in the same application.

Multicore Scaling: App vs. Solver



Application:

- Scales well (sometimes superlinear)
- MPI-only sufficient.



Solver:

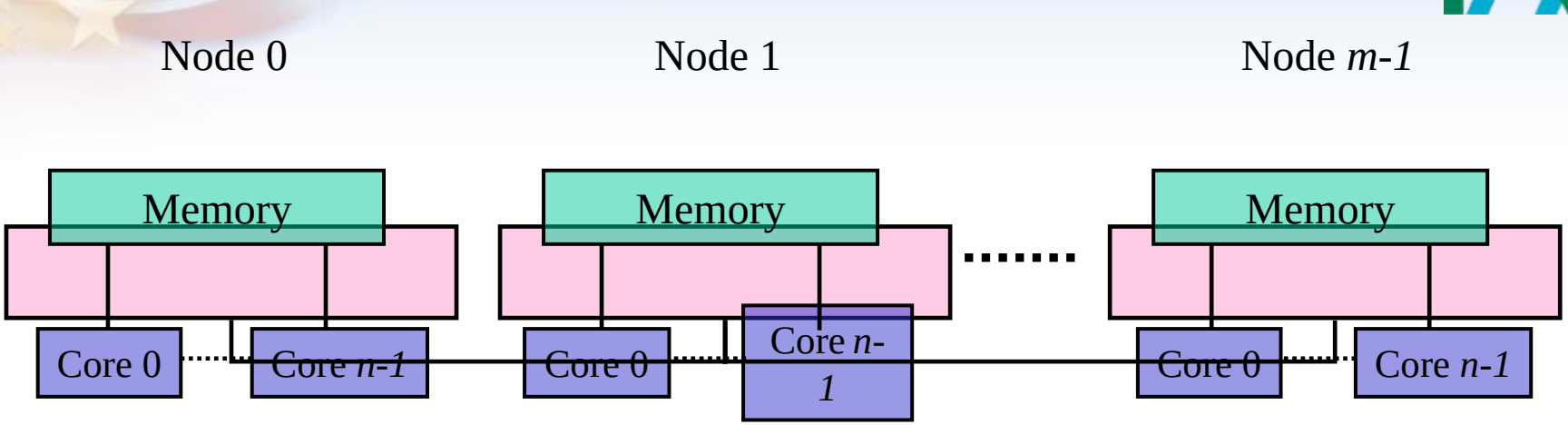
- Scales more poorly.
- Memory system-limited.
- MPI+threads can help.

*** All Charon Results:**
Lin & Shadid TLCC Report

*Courtesy: Mike Heroux

Sandia National Laboratories

Parallel Machine Block Diagram



– Parallel machine with $p = m * n$ processors:
  - m = number of nodes.
  - n = number of shared memory processors per node.
– Two ways to program:
  - **Way 1**: p MPI processes.
  - **Way 2**: m MPI processes with n threads per MPI process.
- New third way:
  - "Way 1" in some parts of the execution (the app).
  - "Way 2" in others (the solver).

Overcoming Key MPI Limitations on Multi-core Processors

- **Hierarchy**
  - **Use MPI communicators to expose locality**
    - **MPI_COMM_NODE, MPI_COMM_SOCKET, etc.**
  - **Allow application to minimize network communication**
  - **Explore viability of others communicators**
    - **MPI_COMM_PLANE_{X,Y,Z}**
    - **MPI_COMM_CACHE_L{2,3}**
- **Shared memory**
  - **Extend API to support shared memory allocation**
    - **MPI_ALLOC_SHARED_MEM()**
  - **Only works for subsets of MPI_COMM_NODE**
  - **Avoids significant complexity associated with using MPI and threads**
  - **Hides complexity of shared memory implementation from application**

Affinity and Scheduling Extensions

- **Processor affinity**
  - **Provide a method for the user to give input about their algorithms requirements**
- **Memory affinity**
  - **Expose the local memory hierarchy**
  - **Enable “memory placement” during allocation**
- **Scheduling**
  - **Provide efficient communication in the face of steadily increasing system load**
  - **Attempt to keep processes 'close' to the memory they use**
  - **Interaction between MPI and the system scheduler**

Ultimate goal is to expose enough information to application scientists to enable the implementation of new algorithms for multi-core platforms

Initial Targets are Open MPI and Cray XT

- **Open MPI is a highly portable, widely used MPI package**
  - **Our extensions should work across a wide range of platforms**
  - **Already has hierarchical communicators and shared memory support at the device level**
  - **We will expose these to the application level**
- **ORNL and SNL have large Cray XT systems**
  - **We have significant experience with system software environment**
  - **Open MPI is the only open-source MPI supporting Cray XT**
  - **We will target both Catamount and Cray Linux environments**
- **Standardizing our effort**
  - **Extension – potential proposals for MPI-3**
  - **ORNL and SNL have leadership roles in MPI-3 process**
    - **Al Geist, Steering Committee**
    - **Rich Graham, Steering Committee, Forum Chair, Fault Tolerance lead**
    - **Ron Brightwell, Point-to-point Communications lead**

Technical Details

Influencing Future Architectures

Evaluate the algorithmic impact of future architecture choices through simulation at the node and system levels

Detailed performance analysis of key computational kernels on different simulated node architectures using SST. For example, discovering that address generation is a significant overhead in important sparse kernels

Analysis and development of new memory access capabilities with the express goal of increasing the effective use of memory bandwidth and cache memory resources.

Simulation of system architectures at scale (10^5—10^6 nodes) to evaluate the scalability and fault tolerance behavior of key science algorithms.

Progress of Project

Built a Strong Project Team

Mix of math, CS, and application experts

Climate (HOMME)

- Mike Heroux, Mark Taylor, Chris Baker (SNL)
- George Fann, Jun Jia, Kate Evans (ORNL)

Materials and Chemistry (MADNESS)

- George Fann, Judith Hill, Robert Harrison (ORNL)
- Mike Heroux, Curt Janssen (SNL)

Semiconductor device physics (Charon)

- George Fann, John Turner (ORNL)
- Mike Heroux, John Shadid, Paul Lin (SNL)

Project team includes key application developers

Runtime and Affinity

- Ron Brightwell, Kevin Pedretti, Brian Barrett (SNL)
- Al Geist, Geoffroy Vallee, Gregg Koenig, Hong Ong (ORNL)

Staying on track with Bi-weekly telecoms

Simulation

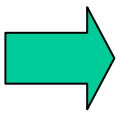
- Arun Rodrigues, Scott Hemmert (SNL),
- Christian Engelmann, Phil Roth, Sadaf Alam (ORNL)
- Bob Numrich (UM), Bruce Jacobs (U Maryland), Sudhakar (GaTech)

Parallel in Time Method
Spectral Deferred Correction

- Blackbox framework completed
  - Templated construction in place
  - Preconditioner in time inherited from existing low-order method
  - Parallel interface under development using Trilinos
  - Example test codes working in MADNES in parallel multicore mode
  - Building interface to Trilinos to use iterative solvers

Improving Solution Methods in HOMME

- BDF2 time stepping option added, in addition to the existing Crank-Nicholson
- A prototype flexible code structure has been added to use different customized iterative solvers and preconditioner in HOMME (e.g. calls GMRES or CG or … from the Trilinos library) with the Trilinos solver framework
- Establishing a baseline for future comparisons and benchmarks
- Anticipating multicore kernels from Trilinos

Next Steps 

- Add code so the existing semi-implicit methods can be used as preconditioners
- Extension to iterative solver logic so that a multi-level method can be added as a black box within HOMME and interface with Trilinos

Status of level scheduling work

- Completed level scheduling algorithm implementation
- Completed permutation functions to support forming explicit PAP^T into D_i and F_i blocks.
- Next step: threaded version of SpMV.
- Continue to make progress on the mixed precision efforts, adding support for graphs in Tpetra (required to compute the level schedules)

Application Performance and Run-time System, Simulator

- Began investigating threading behavior and performance behavior of MADNESS using simulators (e.g. SNL's, ORNL and …). Preliminary assessment on applicability.

- Defining, displaying and understanding "memory and cache access pattern" that is useful to compilers and applications for improving performance—DAG analysis

Runtime Progress

Overcoming key MPI limitations on multi-core processors

Building on Open MPI – a highly portable, widely used MPI package

- Our extensions should work across a wide range of platforms
- The extensions are needed by the architecture aware algorithms
- Our focus is the Cray XT, which SNL and ORNL have large systems

Hierarchal MPI programming

- MPI_COMM_NODE
- MPI_COMM_SOCKET
- MPI_COMM_NETWORK
- MPI_COMM_CACHE Design phase

Shared memory

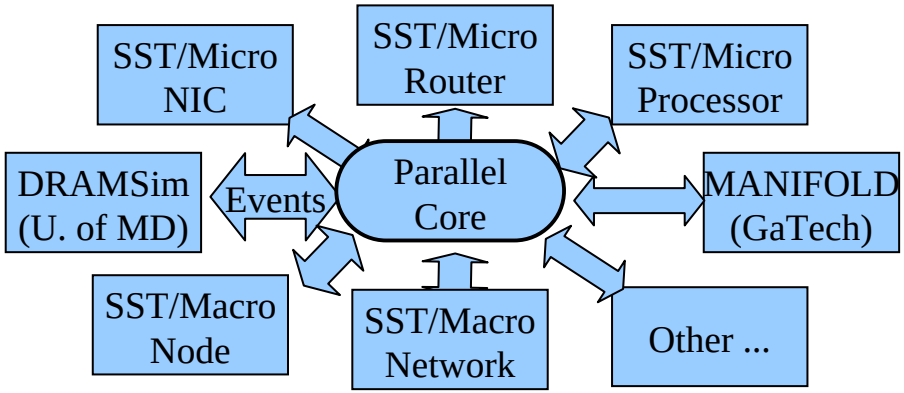
- MPI_ALLOC_SHARED_MEM Design phase

This feature will allow algorithm developers to avoid significant complexity associated with using MPI and threads

Node Simulation Progress

Creating an open source, parallel, modular simulator

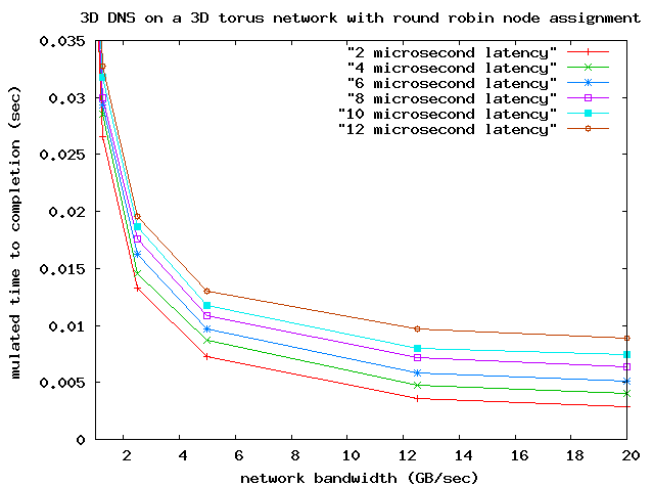
Engaging with labs, universities, and industries, which is key to having the SST become a popular community resource



Modularity: SST/Macroscale simulator code has been redesigned to use a general parallel discrete event simulator (PDES) core

Parallel core: Several PDES cores have been evaluated or developed to study design and performance.

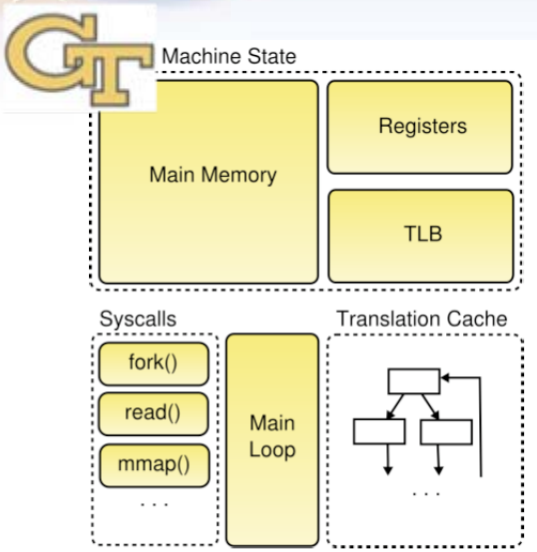
Skeleton applications have been developed to provide a rapid way to test algorithm design choices under a variety of conditions



Study of sensitivity to latency and bandwidth of a nearest neighbor communication pattern

OAK RIDGE National Laboratory

Academic Simulation Partners



Qemu


DRAMSim II

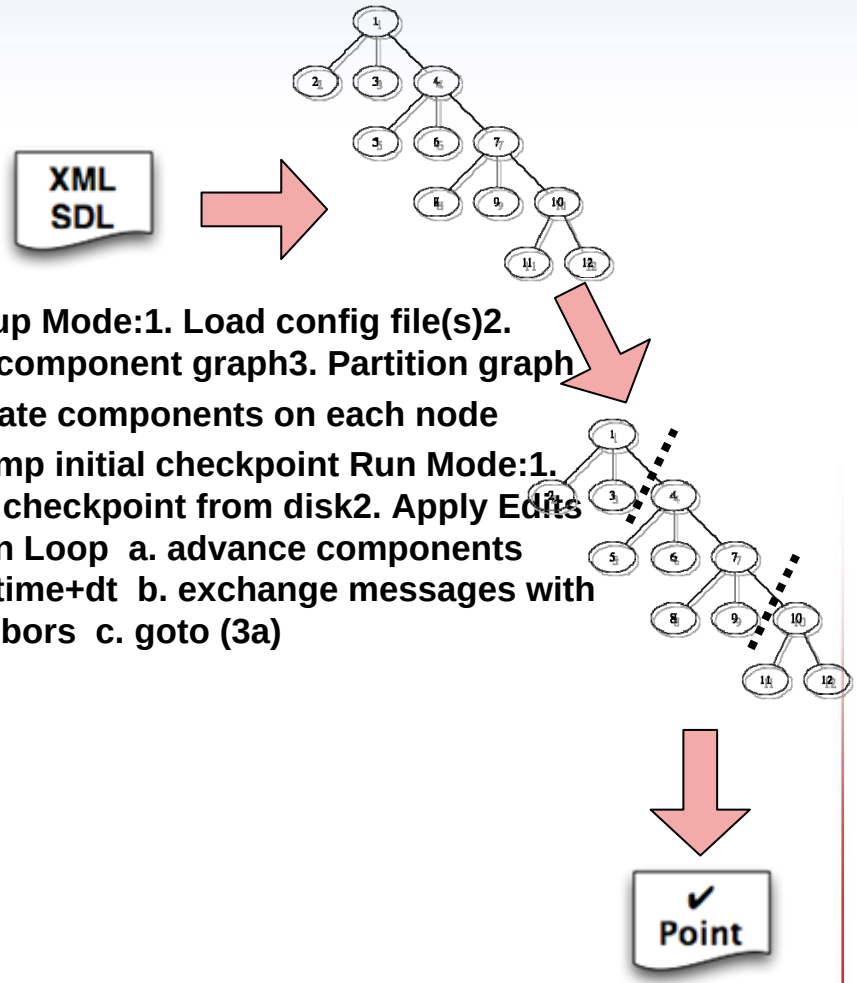
- Georgia Tech
  - Summer student starting in May
  - Work on Qemu emulation front-end
  - Allows emulation of x86, PPC, SPARC, ARM, m68k
- Univ. Maryland
  - New version of DRAMSim II
  - GUI Front end for visualization and configuration
- New Mexico State
  - Jeanine Cook on sabbatical at SNL
  - Extending stochastic processor models

National Nuclear Security Administration

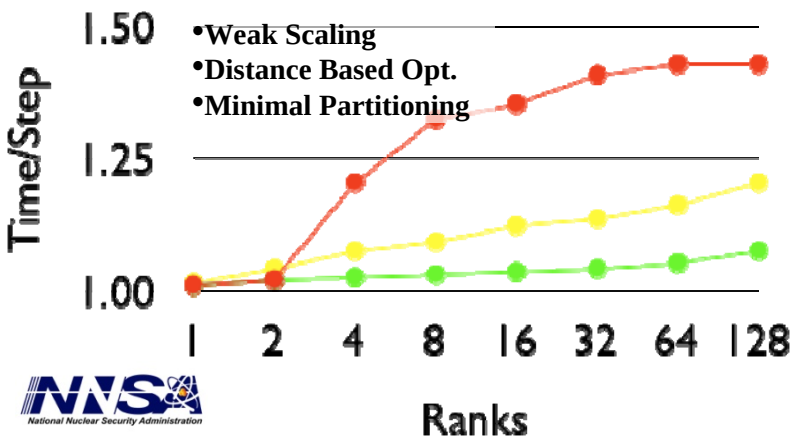
Sandia National Laboratories

Parallel SST Strawman

- **Strawman**
  - **“API Testbed”**
  - **< 1000 lines of code**
  - **Demonstrates basic functionality**
    - **Sim startup**
    - **Component partitioning**
    - **Checkpointing**
    - **Event passing**
- **Current work**
  - **System Description Language**
  - **Refining Parallel DES**
  - **Event interfaces**
  - **Scaling**



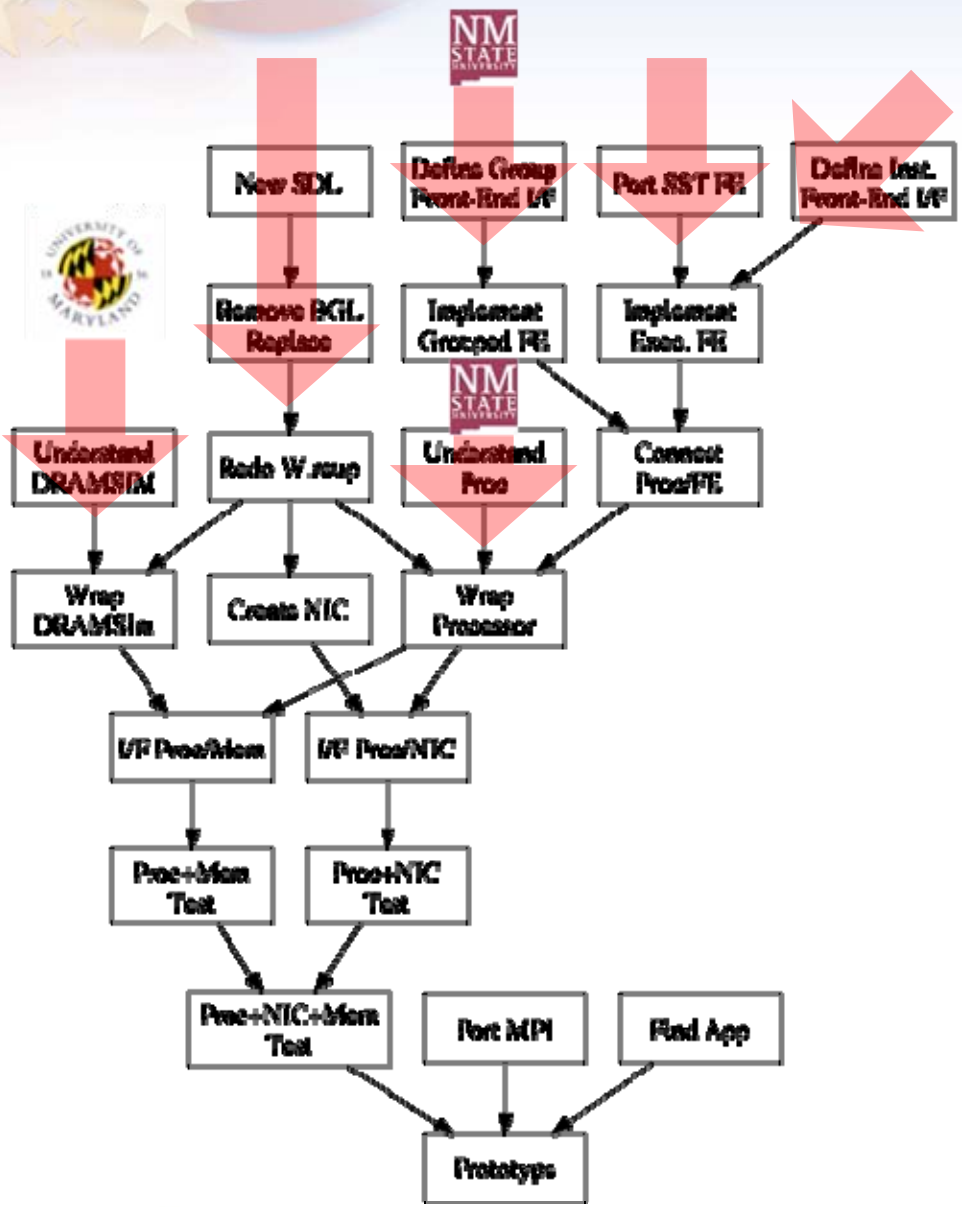
- **Initial/Setup Mode:1. Load config file(s)2. Generate component graph3. Partition graph**
- **4. Instantiate components on each node**
  - **5. Dump initial checkpoint Run Mode:1. Read checkpoint from disk2. Apply Edits 3. Run Loop a. advance components upto time+dt b. exchange messages with neighbors c. goto (3a)**



NNSA National Nuclear Security Administration

Sandia National Laboratories

Implementation Status

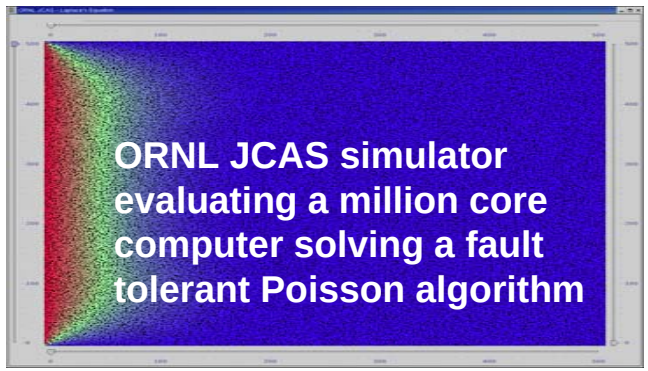


- **Components**
  - DRAMSIM II
  - NMSU Stochastic models
  - Front Ends
    - SST PPC Execution
    - 'Grouped' Model
  - Simple NIC/Net model
- Framework
  - SDL
  - Checkpointing
  - Parallel DES

System Simulation Progress
Investigating algorithm behavior at extreme scale

Evaluating scalability, performance, and fault tolerance of algorithms on millions of nodes.

JCAS runs in parallel on a Linux cluster and is designed to study fault tolerance of C and Fortran MPI algorithms at extreme scale.



Working to replacing JCAS core with μπ a highly scalable PDES that would extend JCAS to allow performance (timing) studies.
•μπ (MUPI) – "micro parallel performance investigator"

•Improved MPI communication support in the simulator
•Investigating design choices
  – interface for simulated machine model and
  – support for models generated by cycle-accurate simulations

Questions?