SAND2011-3557C

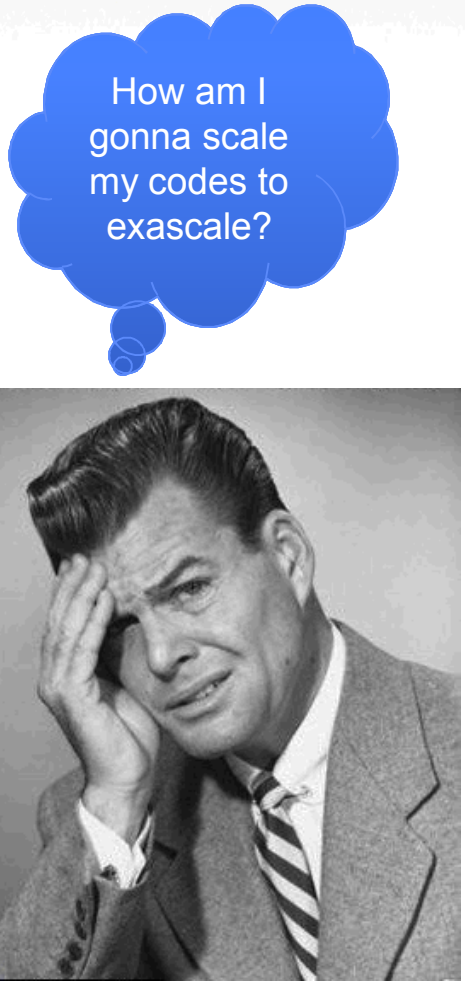
Challenges of Programming Exascale Class Machines

PDSEC-2011 Keynote
Robert L. Clay
Sandia National Laboratories
May 20, 2011
Anchorage, AK

What's this talk about?

- Preview of exascale computing challenges, from an application developer's perspective
- Some of what we're doing at Sandia to address these
- Suggestions on how to prepare for exascale

My assertion – exascale presents some serious challenges to apps developers



A little about me

- First off, I'm an apps person at heart
- And specifically, an apps system developer
- PhD CMU Chemical Engineering
- ER&E – advanced control and RTO
- Terascale LLC – parallel FEM codes/tools
- Sandia National Laboratories
  - Then: Scalable solvers/systems, FEM frameworks/tools, etc
  - Now: Manager, Scalable Modeling and Analysis Systems

My perspectives on exascale are fundamentally born from my apps background.

Preview of exascale computing, from an apps perspective

“Exascale Changes Everything”

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

DOE Exascale Initiative Roadmap, Architecture and Technology Workshop, San Diego, December, 2009.

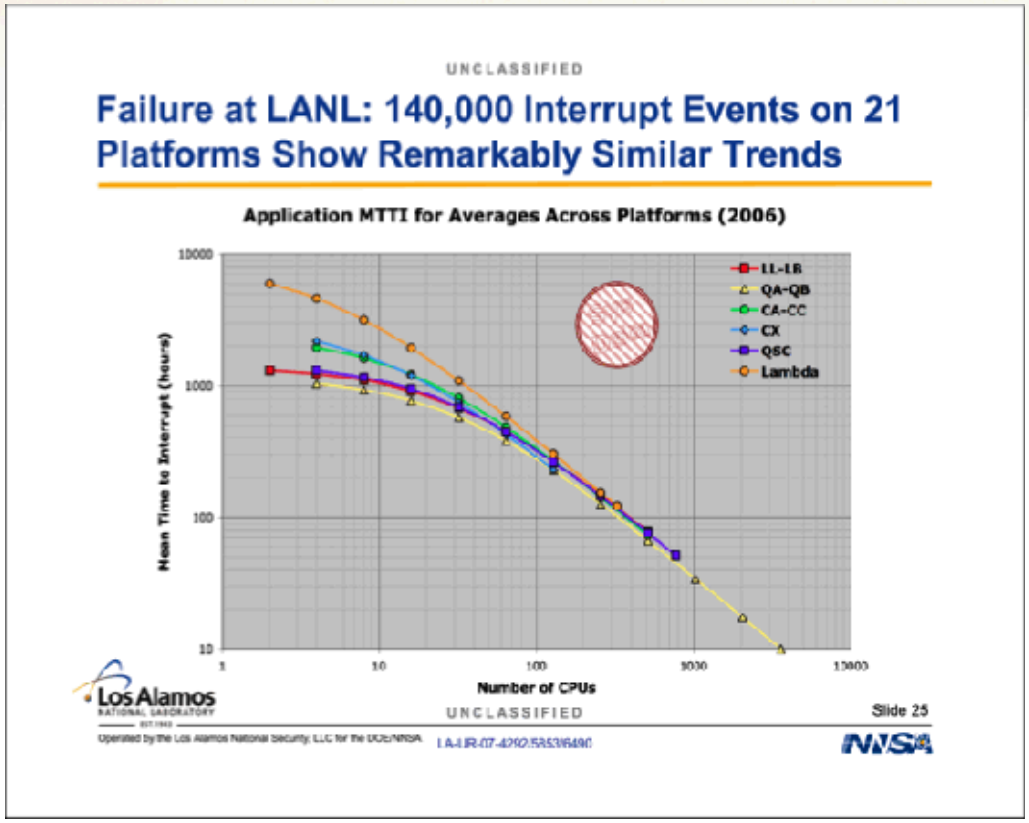
Courtesy of Lucy Nowell & Sonia Sachs, DOE Office of Science

That’s not all…

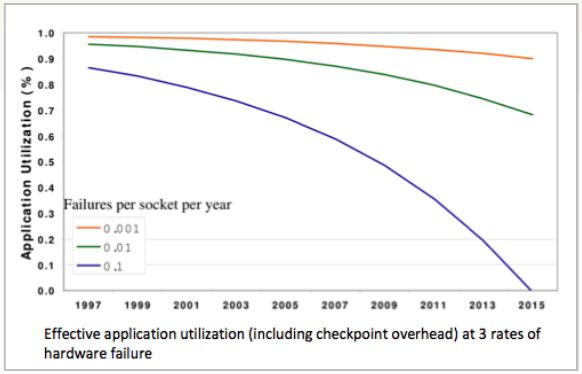
System complexity is increasing

- Heterogeneous node architecture (accelerators)
  - CPUs
  - GP-GPUs?
  - FPGAs?
  - Custom accelerators?
- Hierarchical memory layers growing
  - On chip (shared cache?)
  - On board (shared memory?)
  - Inter-node
  - SSD (on-node)
  - HD

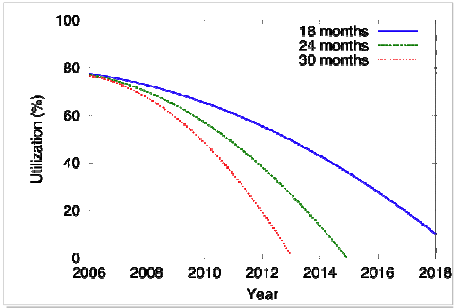
Resilience is a major issue



(Courtesy of John Daly)



(Courtesy of Lucy Nowell & Sonia Sachs)



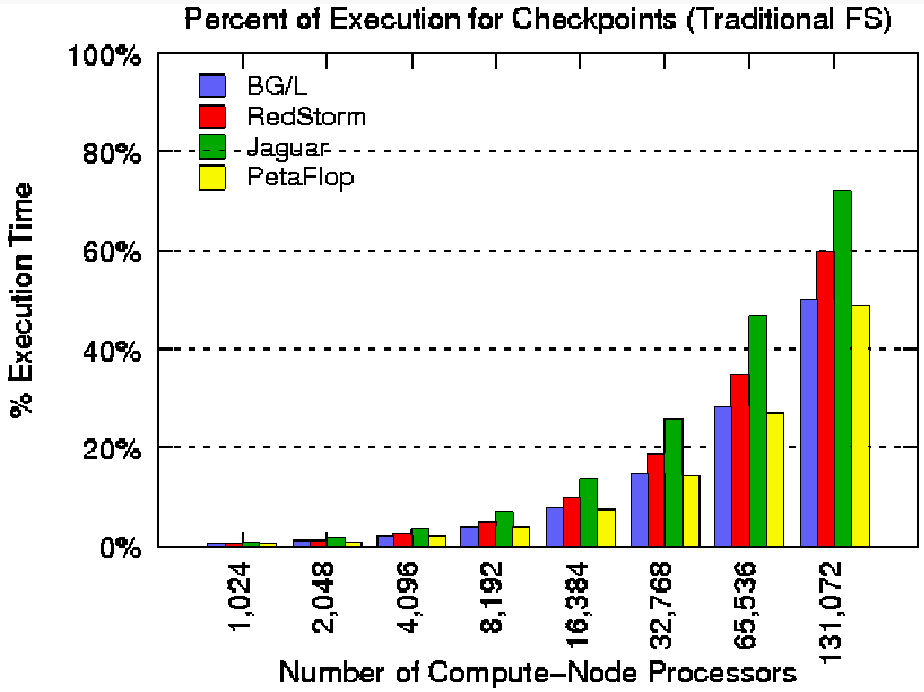
Schroeder and Gibson, *Understanding Failures in Petascale Computers*. Journal of Physics, 2007

(assuming that the number of cores per socket grows by a factor of 2 every 18, 24 and 30 months)

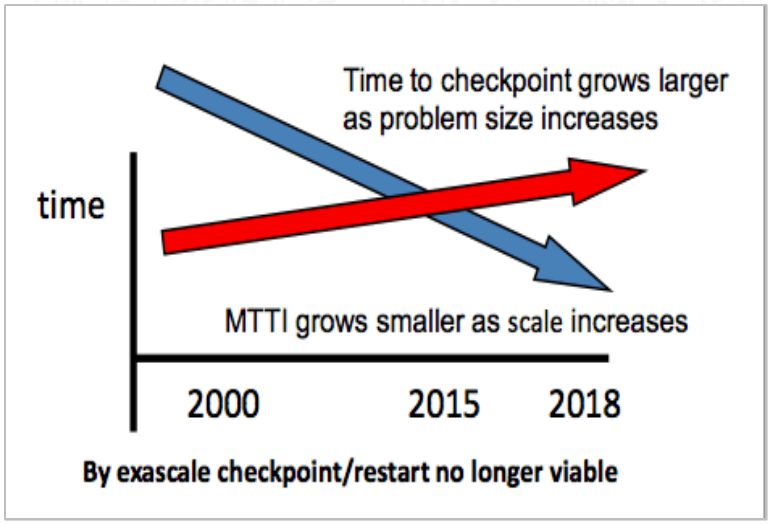
- A "dynamic" view of the machine state is needed.

Sandia National Laboratories

Checkpoint trend isn't good



Oldfield et al., *Modeling the Impact of Checkpoints on Next-Generation Systems*. MSST, 2007



(Courtesy of Lucy Nowell & Sonia Sachs)

- We need more than traditional (system-wide) checkpoint/restart.

Exascale architecture is uncertain

- First exascale system is expected ~2018
- Multiple 'swim lanes' are being worked
- The system design(s) do not exist today
- And, we're not in control of the HW evo/revolution
- So, we can't say for sure what we'll get …

- However, we do know what to expect
  - Heterogeneous nodes (accelerators)
  - Power usage is critical (communication limiting)
  - Many cores/node (1000s)
  - Deeper memory hierarchy (chip, intra-node, inter-node, SSD, …)
  - RAS improvements (but not enough)

Which programming models?

- We don't know yet, and that's a problem.
- Because, it's probably on the critical path for code revisions.
- However, we have ideas and initial guidance
  - Near future: MPI + X (OpenMP, OpenCL, …)
  - Longer term: Action lists, fault-oblivious models, PGAS models, DSLs, etc – this is an active question
  - Question: How to efficiently transition from short-term to long-term solutions?
- This is called out as a critical issue for the DOE (ASC & ASCR) exascale working groups.

So, let's recap the situation as viewed from an application developer's perspective.

Apps developers have some serious challenges ahead

- Extreme parallelism – concurrency is everything
- High cost of moving data – locality is everything
- Reduced system (HW & SW) reliability – static model is broken; need to think dynamic model
- "Unknown" architectures – variations on themes
- Programming model is unclear

Code rewrite could be a huge deal, and take years. It would be good to understand how codes would run on various architectures.

Or, another way to put it is …

- The machines will have ~1000x theoretical peak flops performance, but …
- The machines are getting harder to use in almost every measurable way as viewed from a developer perspective.
- And, we may be in for a near total code rewrite before it's over.

There is hope

- We have ideas on how to address some of the key challenges.
- We have an approach (co-design) that brings a holistic, integrated system engineering methodology to bare.
- Commercial sector is moving in the right direction by default, at least in some key areas.
- We can do strategic acceleration of critical technologies with national-scale program funding.

We will build exascale computers, and we will figure out how to run our codes on them.

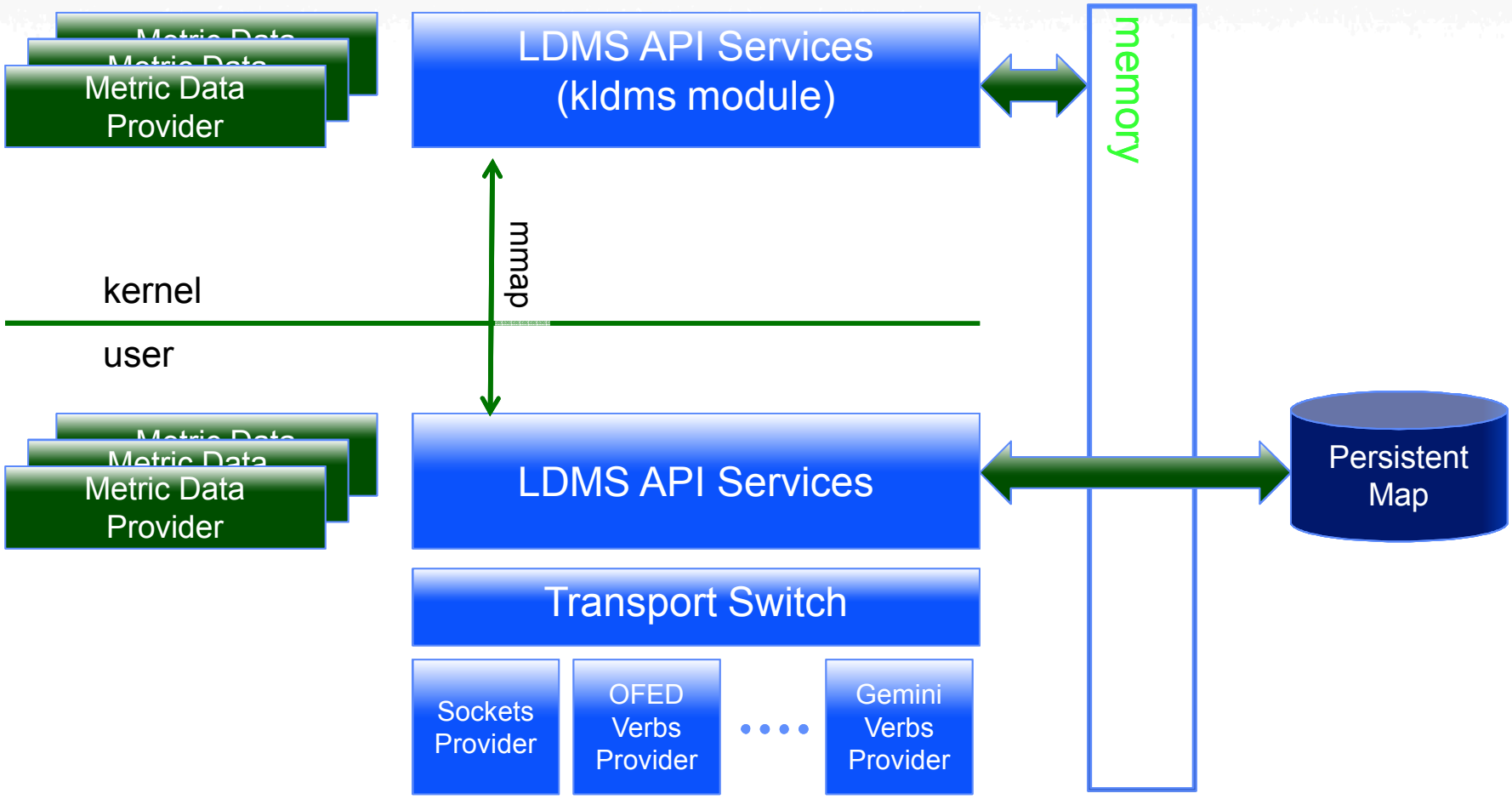
Some work at Sandia to address some of these issues

Focus on selected work at Sandia

- Sandia has a lot going on in exascale R&D
- So do the other DOE labs, and elsewhere
- Main DOE programs are ASCR and ASC
- I'm going highlight a few activities addressing:
  - "Zero-overhead" real-time information collection
  - System state characterization
  - Fault characterization and modeling
  - Dynamic response to state and faults
  - Architecture performance simulation
  - Data-intensive (scalable) architectures

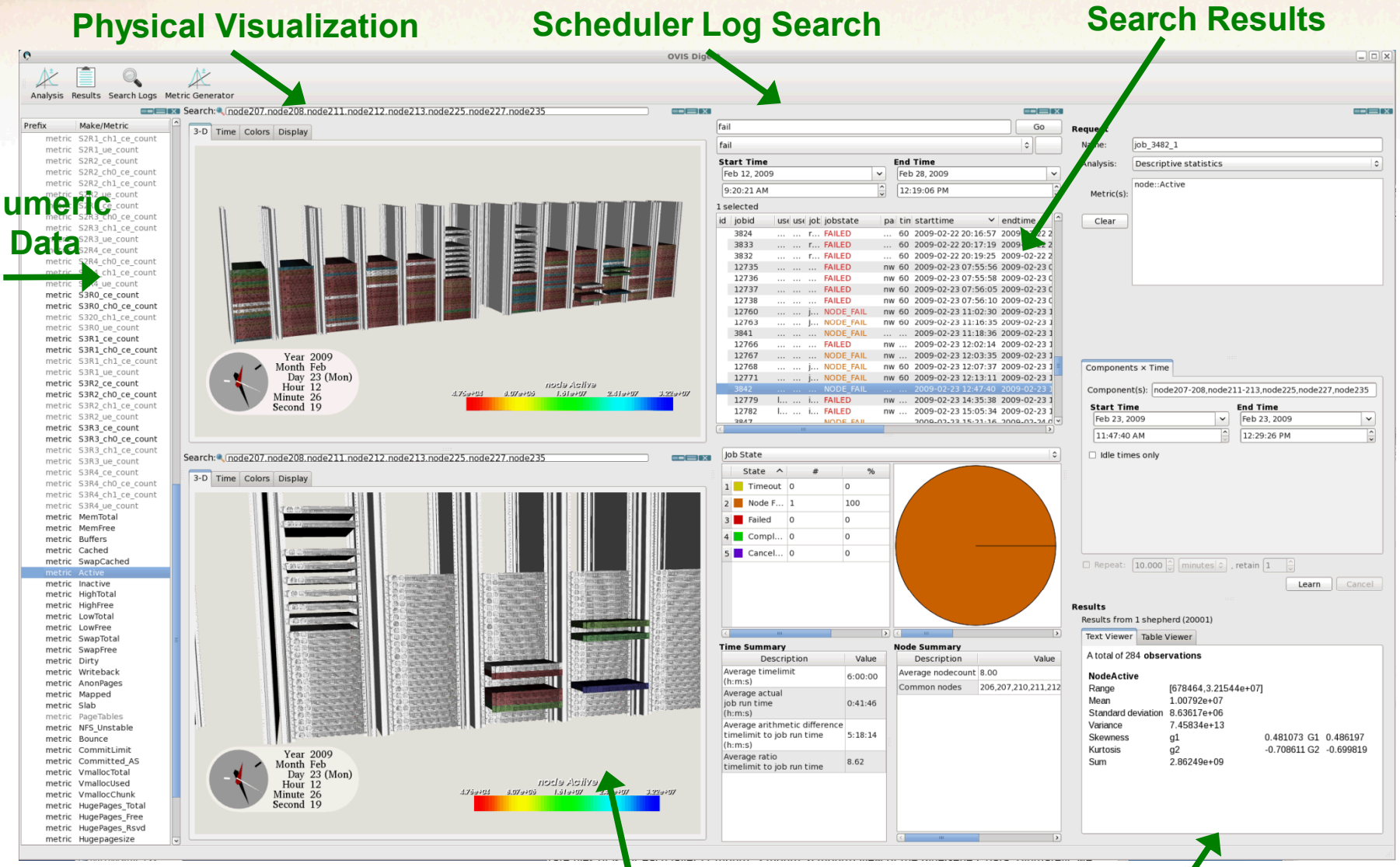
Scalable Collection of Observables

Lightweight Distributed Metric Service (LDMS)



Zero jitter/overhead by integrating with scheduler and using RDMA

OVIS: Data Exploration Toward System Understanding

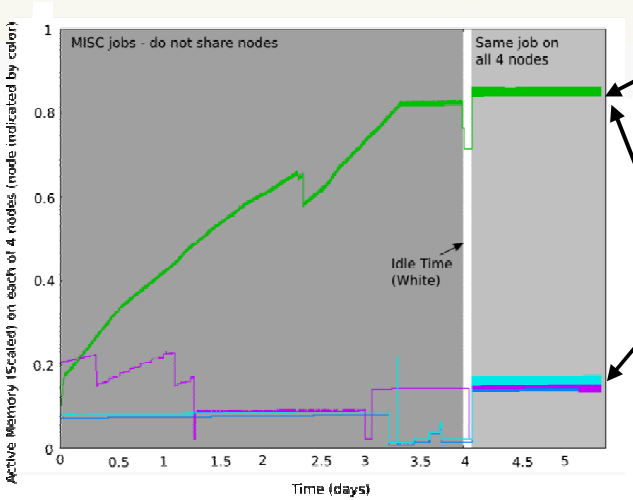


Sandia National Laboratories

Detection of dangerous and anomalous condition invokes mitigating response

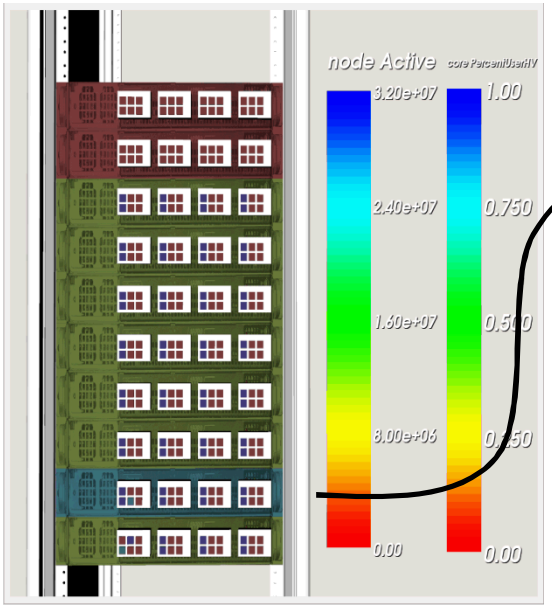
Impending Failure (Memory)

- **Threshold checking – positive**
- **Anomaly checking – positive**
- **Additional resource allocation in coordination with RM**
- **Notify app**
- **Migration**
- **Notify RM**

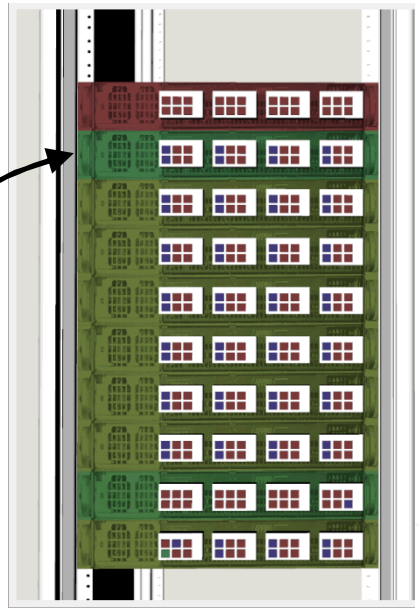


Dangerous

Anomalous



Anomalous condition detected

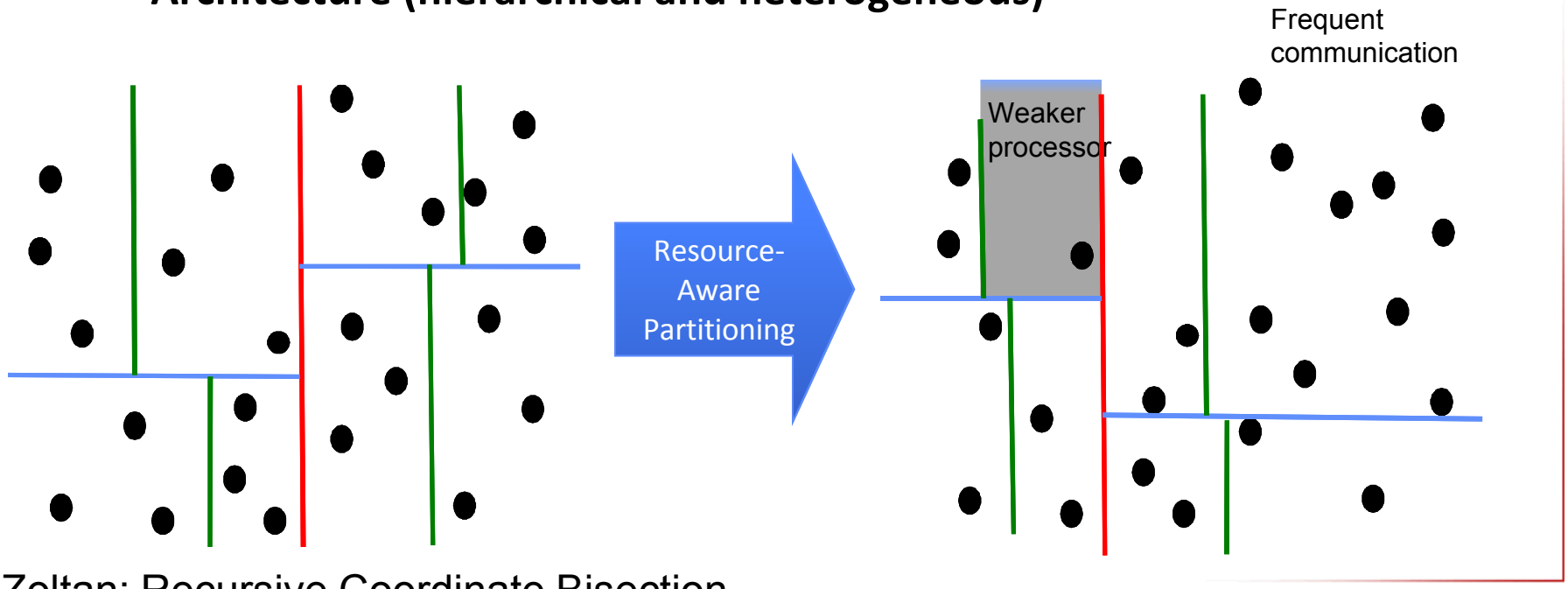


Anomalous condition mitigated via dynamic resource allocation and migration

Sandia National Laboratories

Feedback driven load balancing remaps app based on changing system conditions

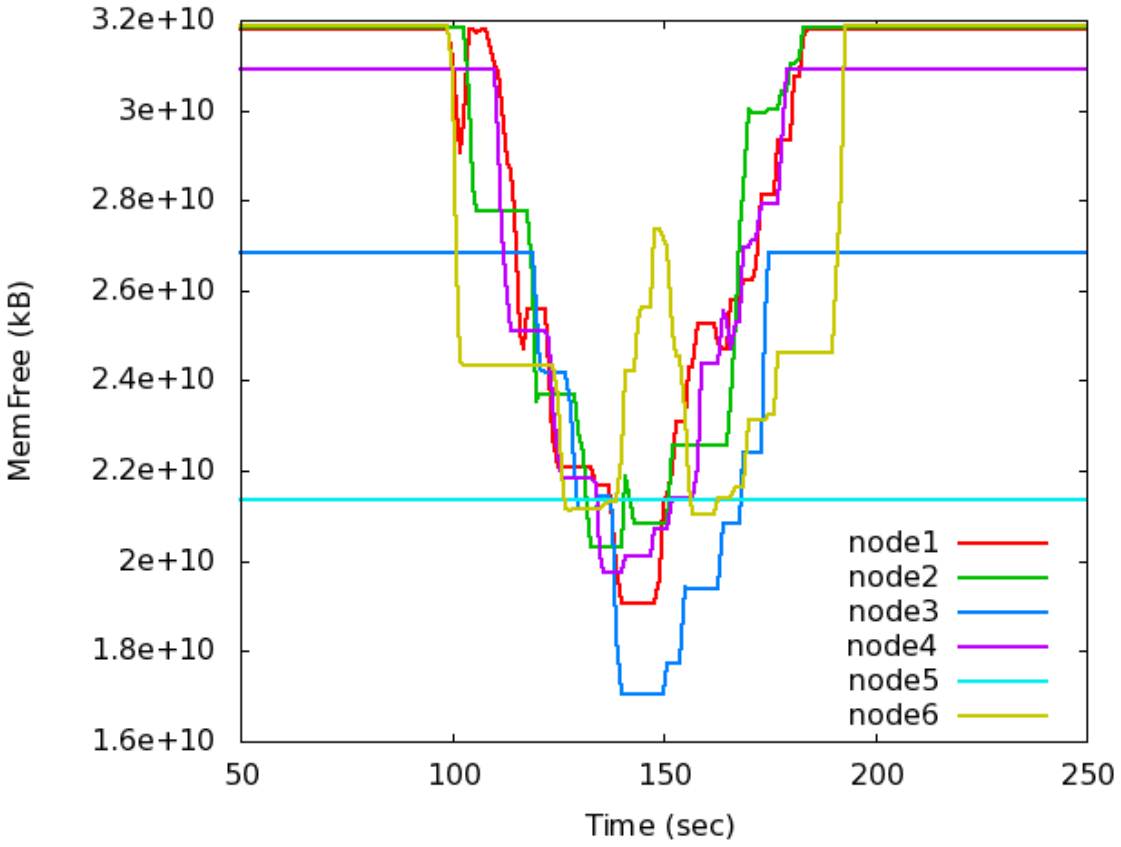
- Dynamically map applications to resources based on:
  - Communication
  - Processor frequency
  - Health
  - Architecture (hierarchical and heterogeneous)



Zoltan: Recursive Coordinate Bisection

Resource-aware computing

Application-HERMES run-time interaction to dynamically determine work-to-resource mapping .



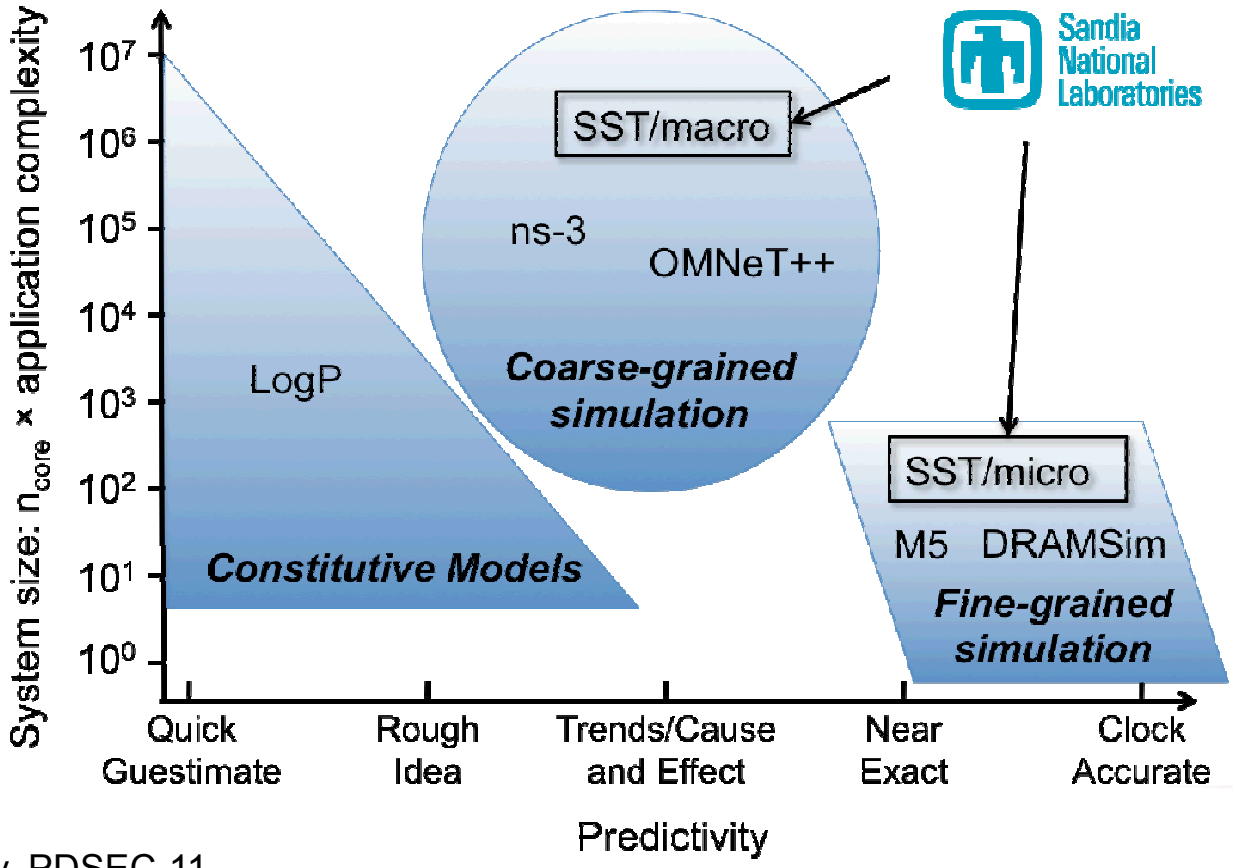
(Adalsteinsson, Brandt, Gentile 2010)

Applications runs one rank/node. Application tasks will require a variable amount of memory. Processes can spawn tasks upon remote nodes.

Processes query local HERMES entity during run to determine node upon which to spawn next task. HERMES framework determines suitable target resource based upon maximum resource available memory.

Simulation permits study of future HPC systems

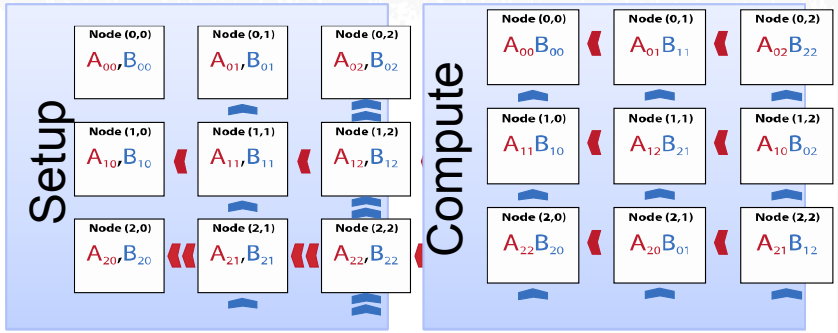
Structural Simulation Toolkit (SST) – create a multi-scale computer architecture for design and procurement of large-scale parallel machines as well as in the design of algorithms for these machines.

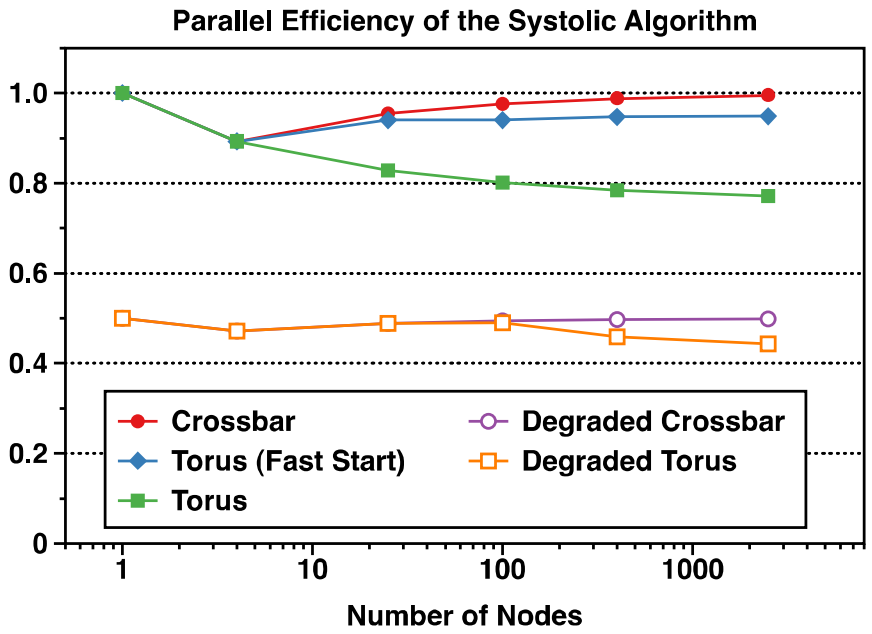


Sandia National Laboratories

Programming model exploration: MPI application

Systolic Matrix Multiplication Algorithm





Skeleton Code Fragment

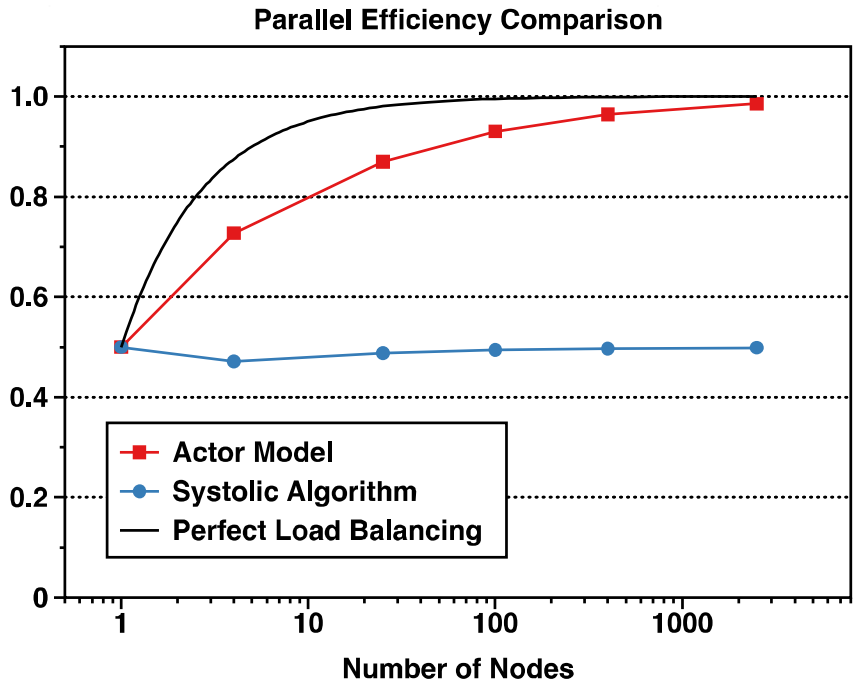
```
for (int i=0; i<nblocks-1; i++) {
  std::vector<sstmac::mpiapi::mpirequest_t> reqs;
  // Begin non-blocking left shift of A blocks
  sstmac::mpiapi::mpirequest_t req;
  mpi()->isend(blocksize, datatype, myleft,
               tag, world, req);
  reqs.push_back(req);
  mpi()->irecv(blocksize, datatype, myright,
               tag, world, req);
  reqs.push_back(req);
  // Likewise for B shifting down ...
  // Simulate computation with current blocks
  compute_api()->compute(instructions);
  mpi()->waitall(reqs, statuses);}
// Finish last block
compute_api()->compute(instructions);
```

- The implicitly synchronous systolic algorithm cannot recover from node degradation

C. L. Janssen, H. Adalsteinsson, J. P. Kenny, *Using simulation to design extreme-scale applications and architectures: programming model exploration*, ACM SIGMETRICS Performance Evaluation Review, **38**, pp. 4-8, 2011.

Programming model exploration: actor model app.

Actor Model Matrix Multiplication Algorithm



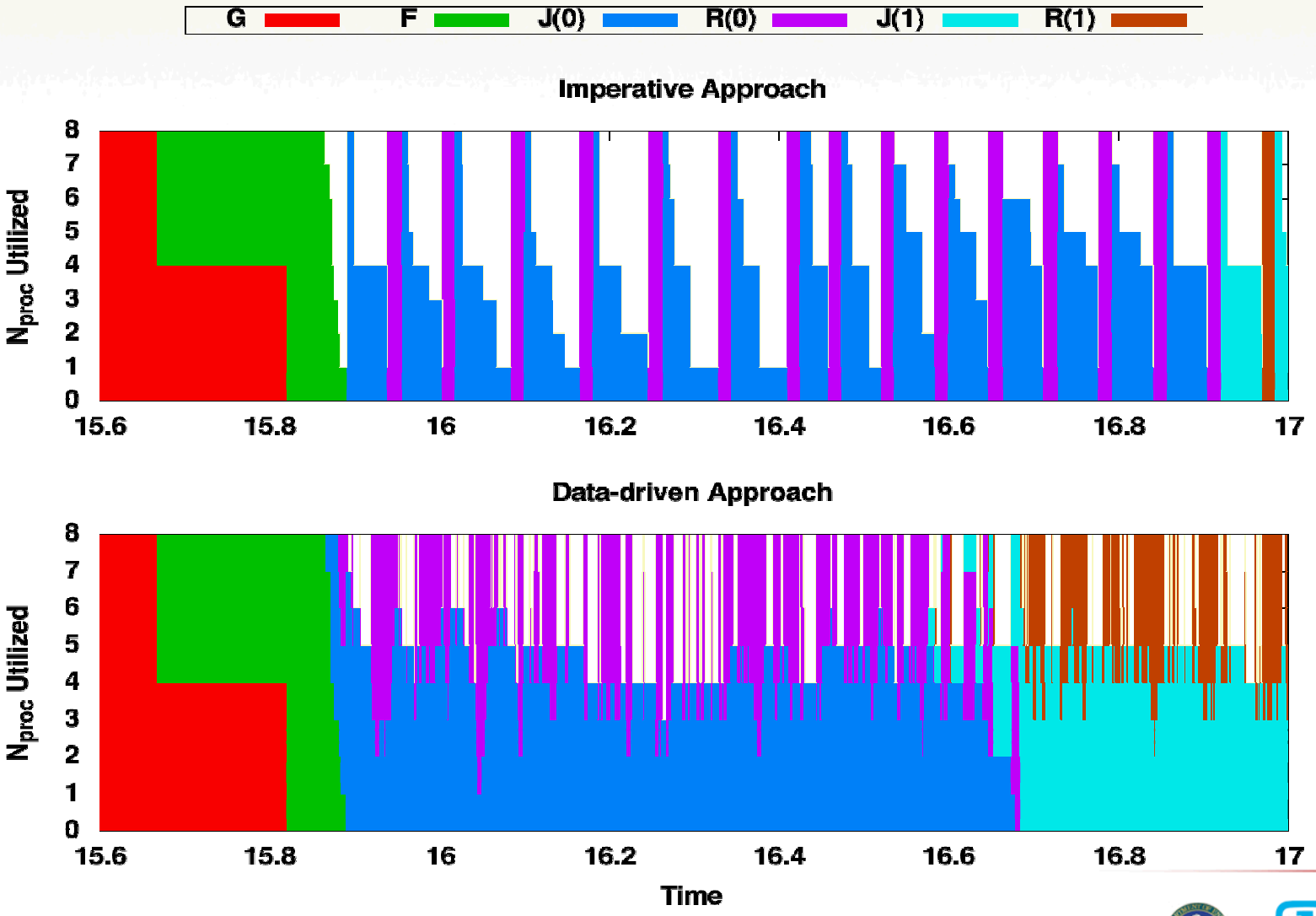
Skeleton Code Fragments

```
// actormatmul run loop body
simplembox::recvresult_t reply =
mbox()->recv(actorid::any(),
        actorpattern::any());
actorid id = reply.first;
shared_ptr<base> msg
 = dynamic_pointer_cast<base>(reply.second);
msg->handle(id, self_.lock());

// actormatmul::compute run loop body
simplembox::recvresult_t res =
mbox()->recv(actorid::any(),actorpattern::any());
boost::shared_ptr<work> msg
 = boost::dynamic_pointer_cast<work>(res.second);
compute(msg);
mbox()->send(msg->store_to(),
        store::construct(msg->iteration()),
        msg->matdim()*msg->matdim());
```

- Simulation permits straightforward investigation of alternative programming models
- Work-stealing approaches will play a role in dealing with large-scale machines lacking perfect homogeneity

Simulated timings for 16 shells on 8 processors



Sandia National Laboratories

Some suggestions on how to prepare for exascale

Recommendations to developers

- Pay attention to the exascale developments
  - It will happen; just a question of how fast
  - The systems components will be pervasive
  - Co-design centers are good exemplars

- Start rethinking your algorithms and applications
  - Probabilistic computing fit? (e.g., UQ, V&V, …)
  - Fault-tolerant algorithms?
  - Adaptive application mapping?
  - Extreme locality and limited communications
  - What would it take to get your apps to exascale?

More recommendations

- Some exciting work to track
  - DSLs (e.g., Hanrahan)
  - Exascale-ready algorithms/libs (e.g., Heroux)
  - Effective use of accelerators (e.g., Dongarra)
  - Auto-tuners (e.g., Dongarra)
  - Fault-oblivious computing (e.g., Minnich/Janssen)
  - New programming models (e.g., Adalsteinsson et al)
  - New compiler R&D (e.g., Quinlan)
  - The list goes on… we heard a lot of it this week

- Keep an eye on ASC and ASCR exascale workshops

In conclusion…

Closing Remarks

- Exascale means massive change, not just massive scale
- The impact on apps is huge – if we're not already rethinking (recoding?) our apps we're probably behind schedule
- It's time to start thinking about reformulating the way we do applications computation

Acknowledgements

Sandia Collaborators: Helgi Adalsteinsson, Jim Brandt, Sudip Dosanjh, Ann Gentile, Curtis Janssen, Ron Minnich, Philippe Pebay, Ali Pinar, Craig Ulmer

Peter Strazdins and Laurence Yang for the opportunity to address this group

Thank you

Contact Info
rlclay@sandia.gov