SAND2014-15458PE

Sandia National Laboratories

Exceptional service in the national interest

Position Talk: Auto-Tuning for Unreliable HPC

Keita Teranishi

Sandia National Laboratories, Livermore, CA

iWAPT 2014, July 1st, 2014

Eugene, Oregon, U.S.A.

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

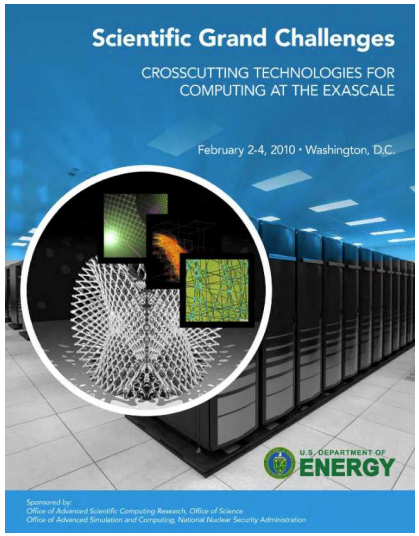
Our Luxury in Life is OVER (wrt FT/Resilience)

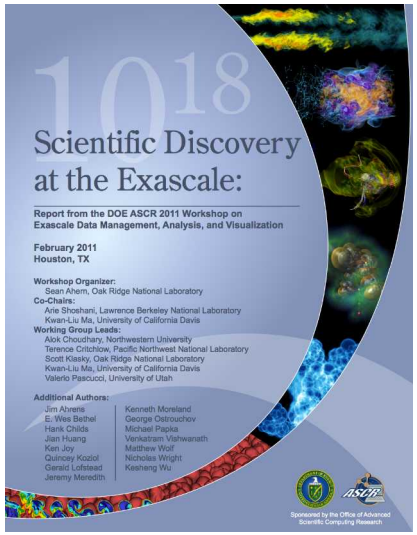
The privilege to think of a computer as a *reliable, digital* machine.

Next generation platforms (NGPs) will have significant increases in concurrency

Sandia National Laboratories

System Parameter	2011	2018		Factor Change
System Peak	2 Pf/s	1 Ef/s		500
Power	6 MW	≤20 MW		3
System Memory	0.3 PB	32-64 PB		100-200
Total Concurrency	225K	1 BX10	1B X100	40000-400000
Node Performance	125 GF	1 TF	10 TF	8-80
Node Concurrency	12	1000	10000	83-830
Network Bandwidth	1.5 GB/s	100 GB/s	1000 GB/s	66-660
System Size (nodes)	18700	1000000	100000	50-500
I/O Capacity	15 PB	30-100 PB		20-67
I/O Bandwidth	0.2 TB/s	20-60 TB/s		10-30

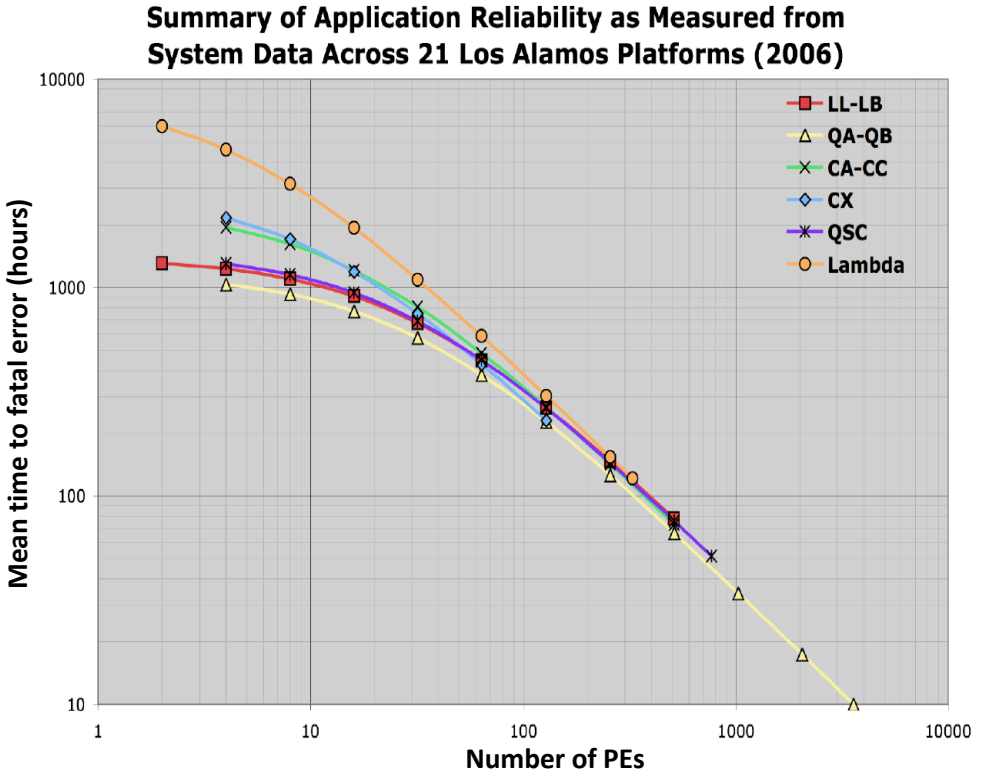




NGPs will experience errors/faults much more frequently than petascale systems

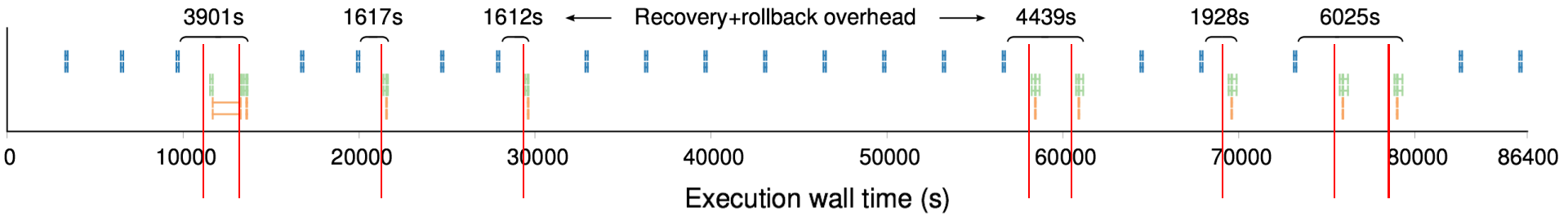
Sandia National Laboratories

- Significant increase in number of components
- Shrinking of semiconductor (CMOS) will reach the size of a few atoms per wire
- Insufficient improvements in mean time between failures (MTTF) for each component
- Majority of failures: single node
  - Today's rate: ~2-10 a day
  - 2020: every 30-60 minutes?



(Courtesy of John Daly)

Example – S3D production runs



- 24-hour tests using Titan (125k cores)
- 9 process/node failures over 24 hours
- Failures are promoted to **job failures**, causing all 125k processes to exit.

As a result, checkpoint (5.2 MB/core) has to be done to the PFS

- Checkpoint data: **55 s** Total: 1.72 %
- Restarting processes: **470 s** Total: 5.67 %
- Loading checkpoint: **44 s** Total: 1.38 %
- Rollback overhead: **1654 s** Total: 22.63 %
- Total overhead due to fault tolerance: **31.40 %**

- **Courtesy of Hemanth Kolla.**
- ***Exploring Automatic Online Failure Recovery for Scientific Applications at Extreme Scales*, M. Gamell, M, Parashar, D. Katz, H. Kolla and J. Chen, to appear in SC14.**

We are already dealing with the (un)reliability of HPC systems, TODAY!

Resilience Problems: Already Here, Already Being Addressed, Algorithms & Co-design Are Key

Sandia National Laboratories

- Already impacting performance: Performance variability.
  - HW fault prevention and recovery introduces variability.
  - Latency-sensitive collectives impacted.
  - MPI non-blocking collectives + new algorithms address this.
- Localized failure:
  - Now: local failure, global recovery.
  - Needed: local recovery (via persistent local storage).
  - MPI FT features + new algorithms: Leverage algorithm reasoning.
- Soft errors:
  - Now: Undetected, or converted to hard errors.
  - Needed: Apps handle as performance optimization.
  - MPI reliable messaging + PM enhancement + new algorithms.
- *Key to addressing resilience: algorithms & co-design.*

Four Resilient Programming Models

Sandia National Laboratories

- Skeptical Programming. (SP)
- Relaxed Bulk Synchronous (rBSP)
- Local-Failure, Local-Recovery (LFLR)
- Selective (Un)reliability (SU/R)
- Old models
  - BSP
  - Reliable digital systems
  - Checkpoint/Restart

Toward Resilient Algorithms and Applications
Michael A. Heroux
arXiv:1402.3809v2 [cs.MS]

Sandia National Laboratories

SKEPTICAL PROGRAMMING

What is Needed for Skeptical Programming?

Sandia National Laboratories

- Skepticism
  - **(Rough) Probabilistic model of computer arithmetic**
- Meta-knowledge:
  - Algorithms,
  - Mathematics,
  - Problem domain.
- Nothing else, at least to get started.

Skeptical Programming

I might not have a reliable digital machine

Sandia National Laboratories

- Expect rare faulty computations
- Use analysis to derive cheap "detectors" to filter large errors
- Use numerical methods that can absorb *bounded error*

Algorithm 1: GMRES algorithm

```
for l = 1 to do
    r := b − Ax^(j−1)
    q_1 := r/||r||_2
    for j = 1 to restart do
        w_0 := Aq_j
        for i = 1 to j do
            h_{i,j} := q_i · w_{i−1}
            w_i := w_{i−1} − h_{i,j} q_i
        end
        h_{j+1,j} := ||w||_2
        q_{j+1} := w/h_{j+1,j}
        Find y = min ||H_j y − ||b|| e_1||_2
        Evaluate convergence criteria
        Optionally, compute x_j = Q_j y
    end
```

GMRES

Theoretical Bounds on the Arnoldi Process

$$\|\mathbf{w}_0\| = \|\mathbf{A}\mathbf{q}_j\| \leq \|\mathbf{A}\|_2 \|\mathbf{q}_j\|_2$$
$$\|\mathbf{w}_0\| \leq \|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$$

From isometry of orthogonal projections,

$$|h_{i,j}| \leq \|\mathbf{A}\|_F$$

- h_{ij} form Hessenberg Matrix
- Bound only computed once, valid for entire solve

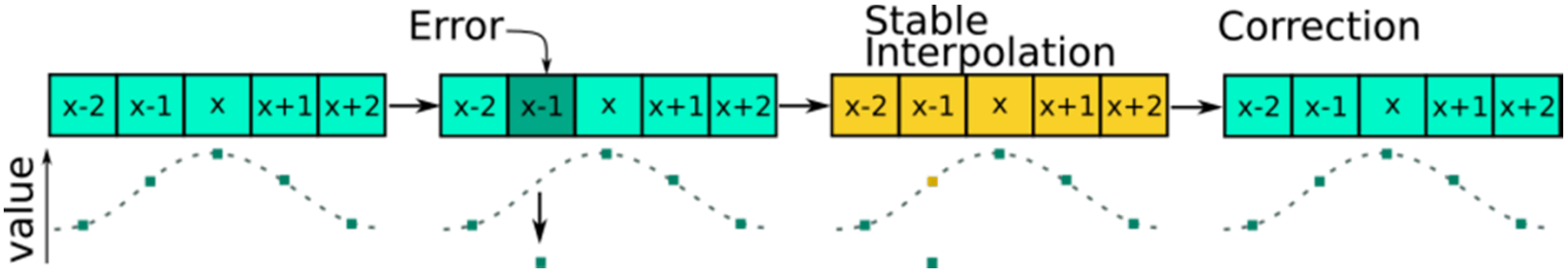
Evaluating the Impact of SDC in Numerical Methods
J. Elliott, M. Hoemmen, F. Mueller, SC'13

Skeptical programming can mitigate silent errors & offer new co-design options

- Even at commodity scale, ECC memory & ECC processors show the rising need for error correction

ECC memory

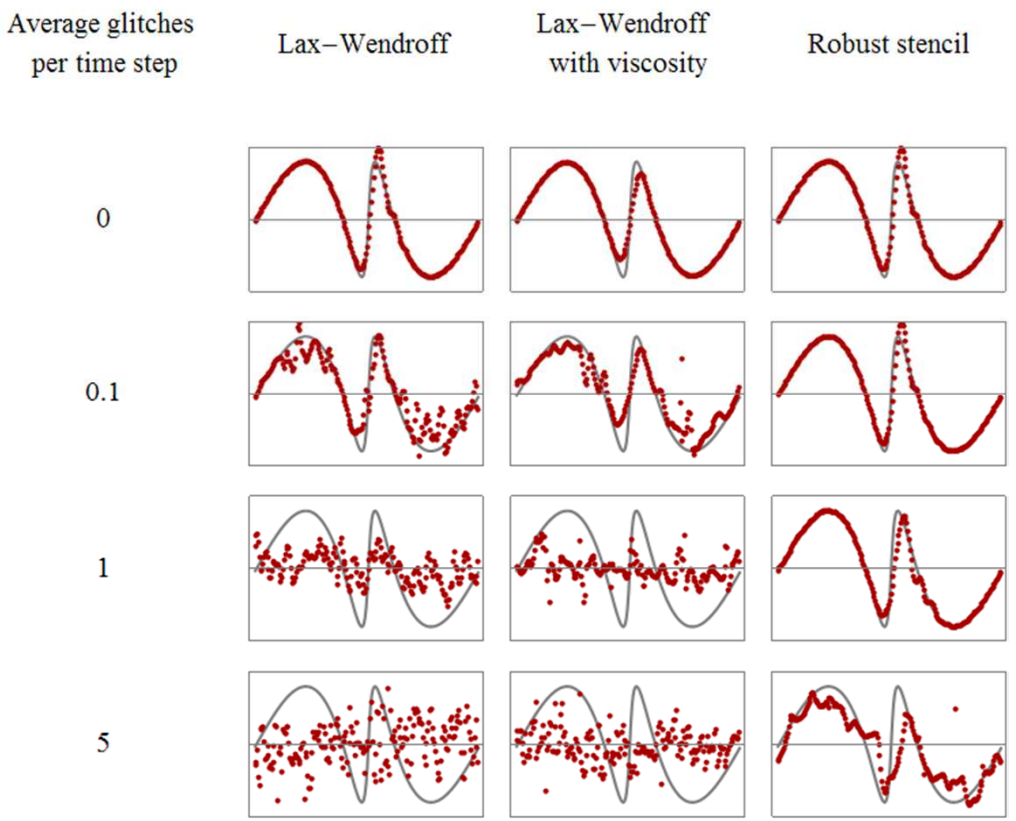
- With increasing scale and with power limitations, errors can occur "silently" without indication that something is wrong
- Numerical algorithms already deal with error from truncation, etc.; specially designed algorithms can mitigate silent bit flips as well



- These robust stencil algorithms not only address scale-up of current silent-error rates, but may enable new "lossy" architecture options with more power-efficient accelerators or reduced latency

Example: “Robust stencils” discard outliers to mitigate bit flips in PDE solving

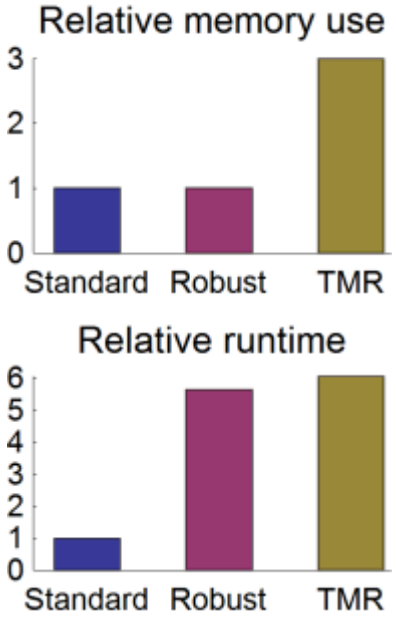
- A simple 1D advection equation $\partial u/\partial t = \partial u/\partial x$ illustrates the behavior of finite-difference schemes
- The robust stencil here computes a second-order updat position i from one of these subsets after discar the most extreme value:
  - $\{ i-3, \quad i-1, \quad i+1, \quad i+3$
  - $\{ \quad i-2, \quad i, \quad i+2 \quad \}$
  - $\{ \quad i-1, \; i, \; i+1 \quad \}$

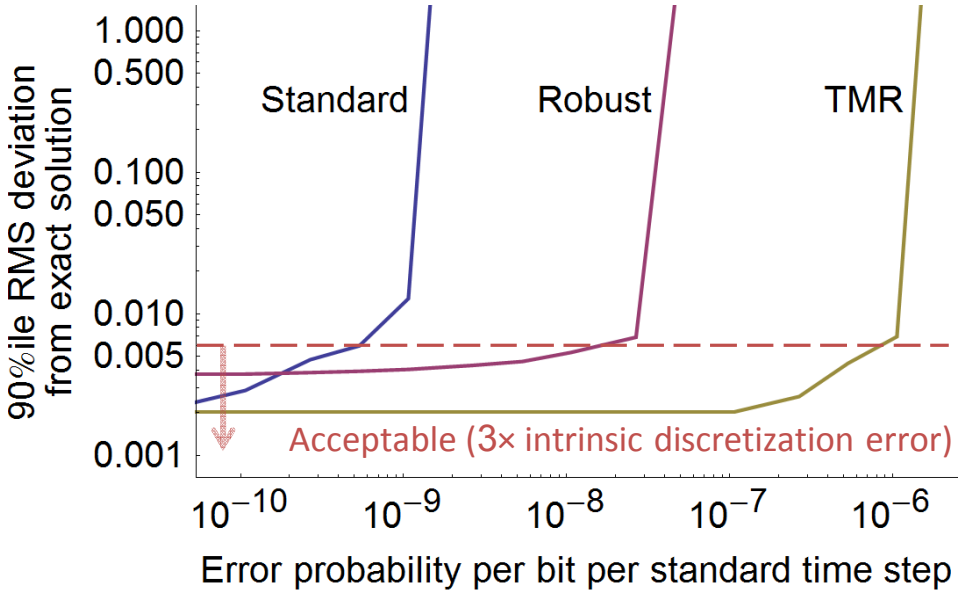


Bit-flip injection at machine level confirms effectiveness of our robust stencil

Sandia National Laboratories

- Focus on silent-error models affecting floating-point
  - Relaxing FP correctness may benefit designs (e.g., GPUs)
- Test: During C++ PDE simulation, asynchronously perform raw memory bit flips in the FP solution array
  - Can also be a proxy for *processor* bit flips that corrupt FP ops
- Compare brute-force triple modular redundancy (TMR)





Here, the robust stencil provides substantial bit-flip tolerance at lower cost than TMR

Courtesy of Jackson Mayo, Rob Armstrong and Jaideep Ray

Sandia National Laboratories

RELAXED BULK SYNCHRONOUS PROGRAMMING (RBSP)

Performance Variability is a Resilience Issue

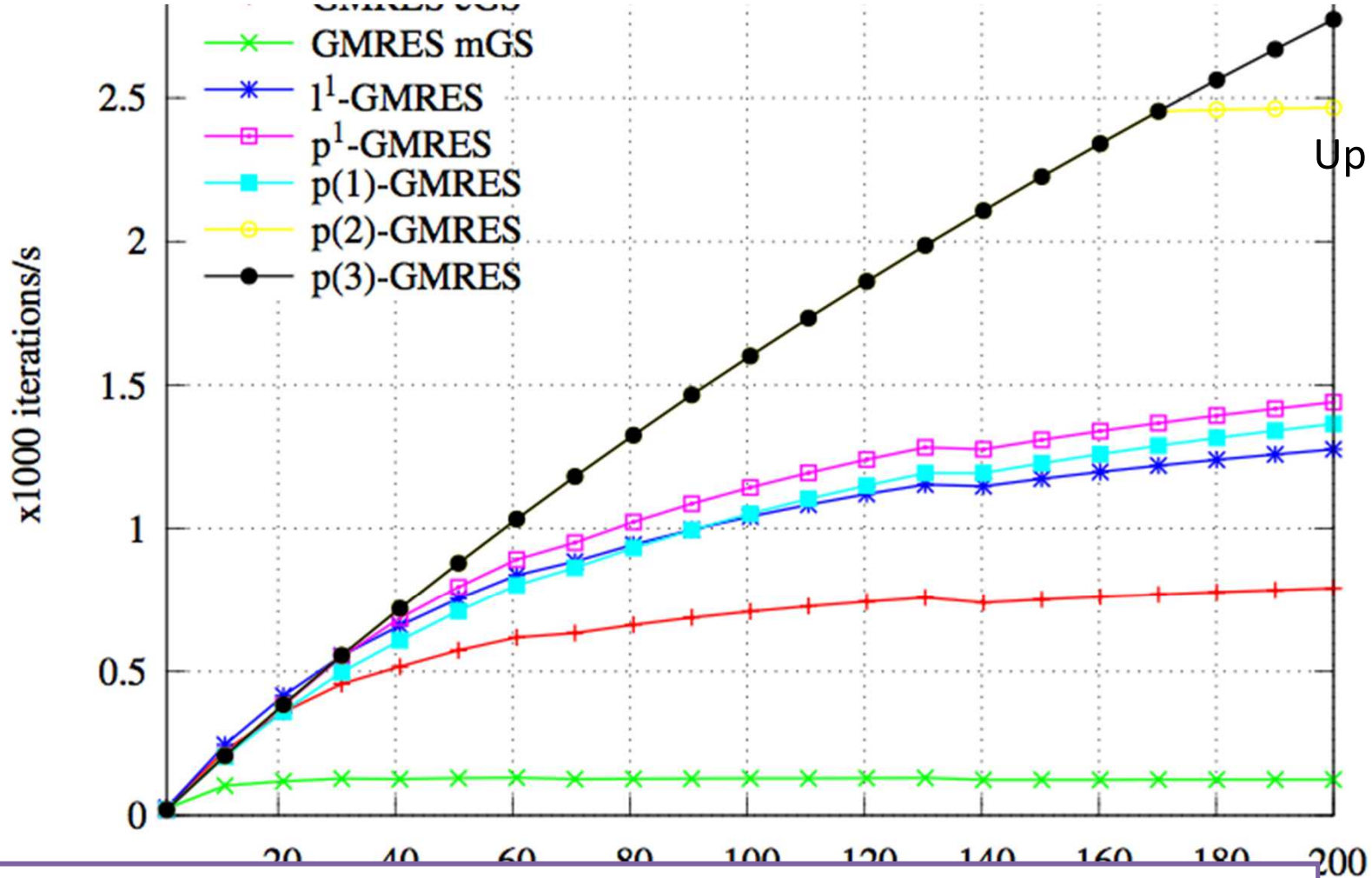
Sandia National Laboratories

- Ideal:
  equal work +
  equal data access =>
  equal execution time.
- Reality:
  - Lots of variation.
  - Variations increasing.

- First impact of unreliable HW?
  - Vendor efforts to hide it.
  - Slow & correct vs. fast & wrong.
- Result:
  - Unpredictable timing.
  - Non-uniform execution across cores.
- Blocking collectives:
  - $t_c = \max_i\{t_i\}$
- Also called “Limpware”:
  - Haryadi Gunawi, University of Chicago
  - http://www.anl.gov/events/lights-case-limping-hardware-tolerant-systems

Latency-tolerant Algorithms + MPI 3: Recovering scalability

Sandia National Laboratories



Up is good

Hiding global communication latency in the GMRES algorithm on massively parallel machines, P. Ghysels T.J. Ashby K. Meerbergen W. Vanroose, Report 04.2012.1, April 2012, ExaScience Lab Intel Labs Europe

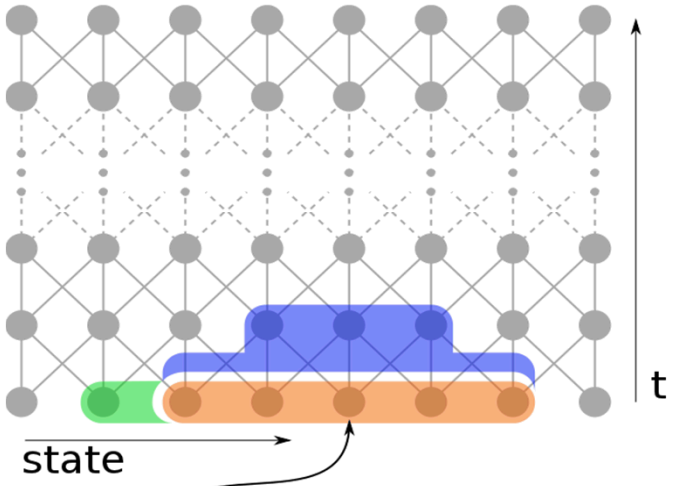
What is Needed to Support Latency Tolerance?

Sandia National Laboratories

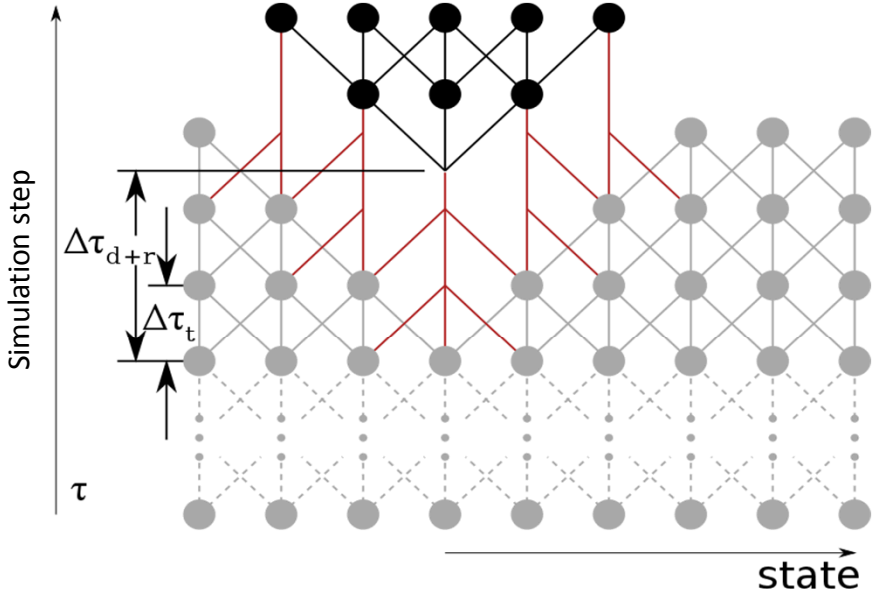
- MPI 3 (SPMD):
  - Asynchronous global and neighborhood collectives.
- A “relaxed” BSP programming model:
  - Start a collective operation (global or neighborhood).
  - Do “something useful”.
  - Complete the collective.
- The pieces are coming online.
- With new algorithms we can recover some scalability.

- **Is MPI-X the right direction?**

Fritz: Simulation of a task-DAG programming model for scaling amid failures

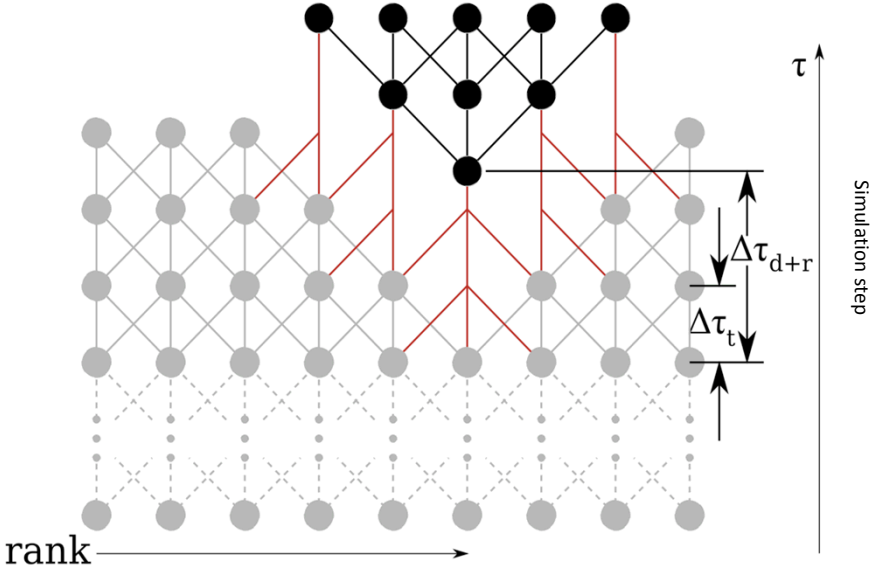


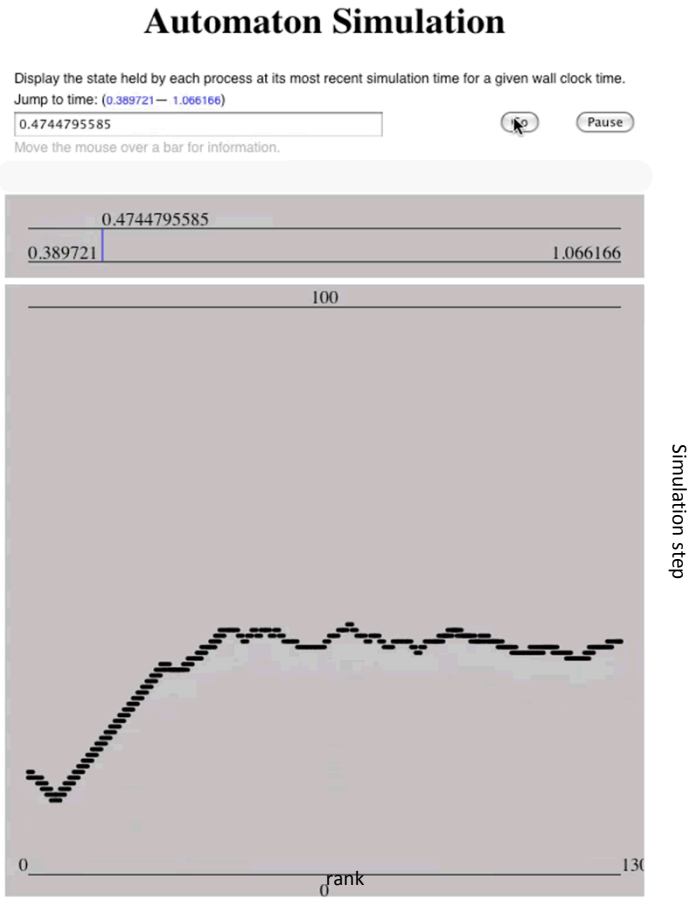
Each process holds a subset of state. Process holdings overlap entirely for redundant, in-memory recovery.



Failures & delays propagate from process to process as state dependencies are communicated.

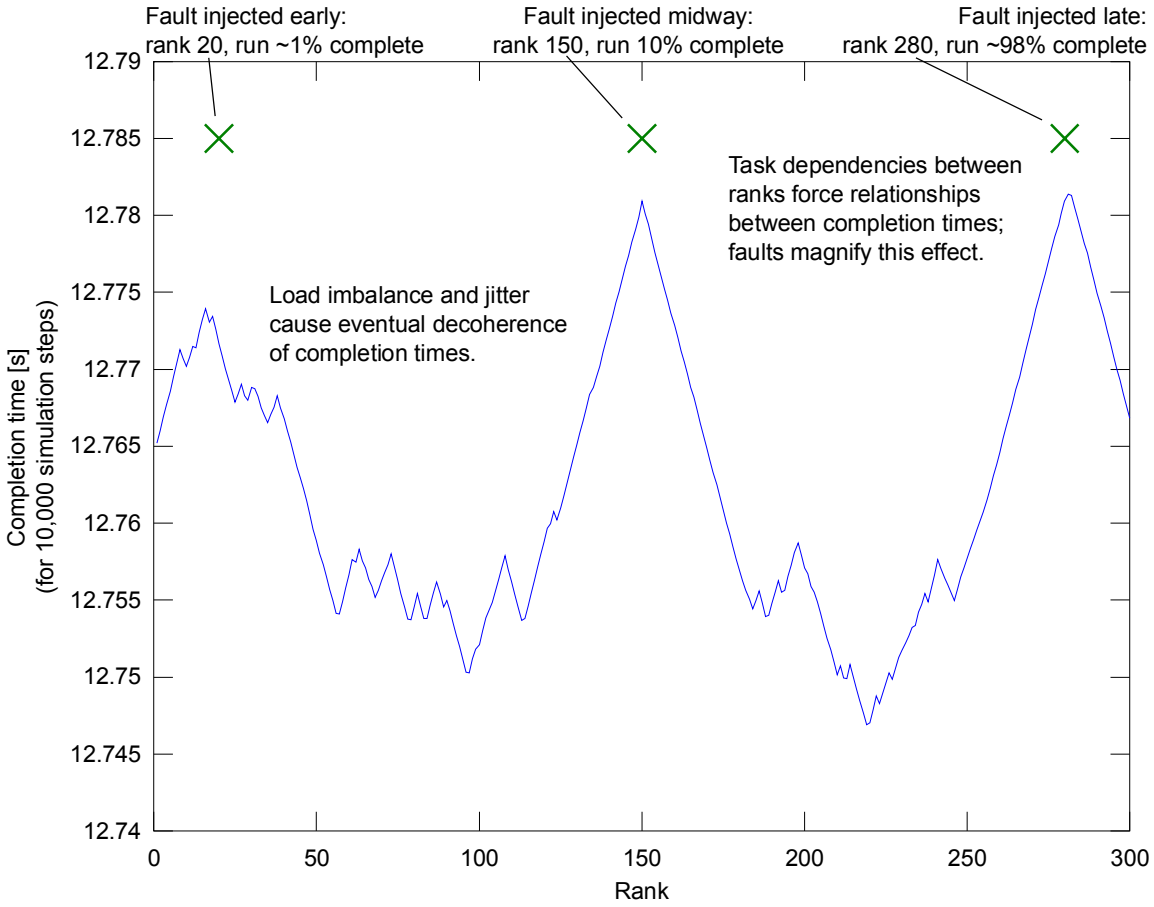
Task-DAGs provide resilience





The movie shows a sequence of horizontal slices through the diagram on the left. Each slice indicates to which simulation step each process has advanced.

Delay scaling of Fritz shows promise under a Poisson failure model



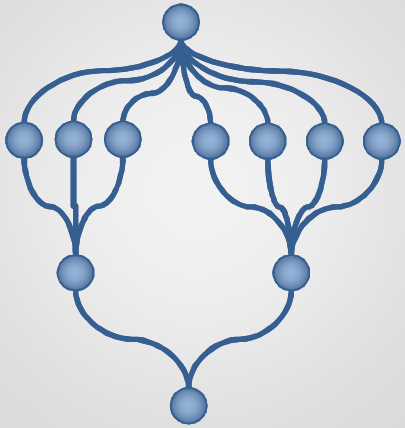
Snapshot of three failure + recovery delays induced on different processes at different simulation time steps

DHARMA: Distributed Heterogeneous Adaptive Resilient Management & Applications

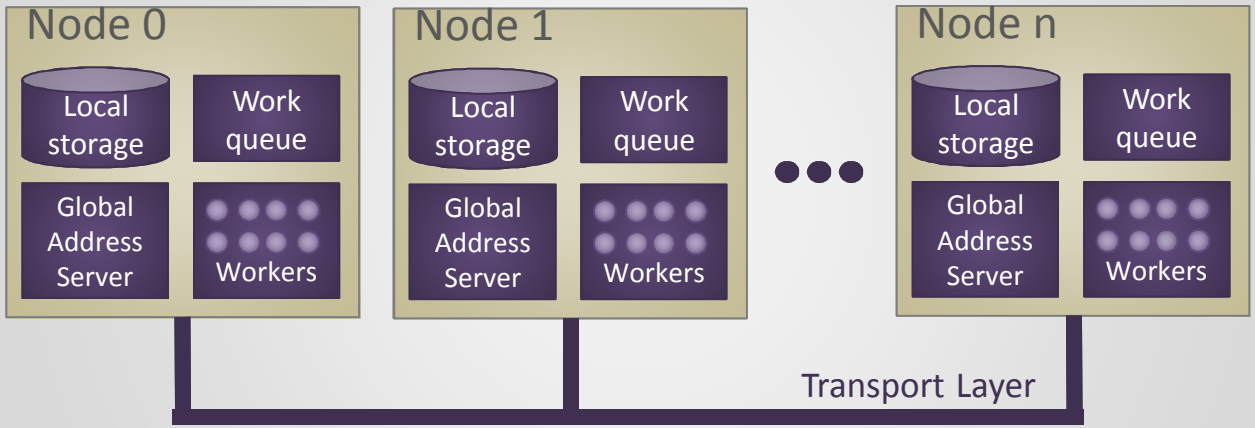
Sandia National Laboratories

- Asynchronous Many Task (AMT) programming models show potential for sustaining performance despite node degradation/failure
  - Work-stealing enables load balancing
  - Failed tasks can be re-executed
- Natural fit for rBSP ☺
  - Achieves scalability and resiliency together!

Task Graph



Typical AMT Runtime Architecture

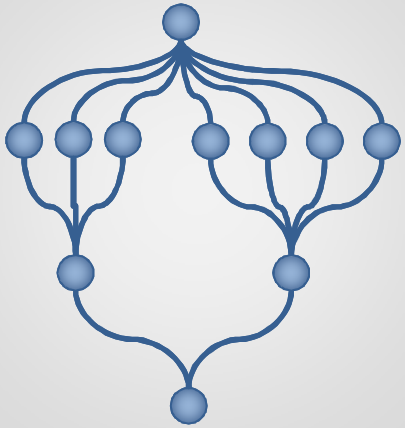


Extreme Scale viability of collective communication for resilient task scheduling, Jeremiah Wilke, John Floren, Hemanth Kolla, KT, Janine Bennett, Nicole Slanttengren, FTXS-14 (See the slides).

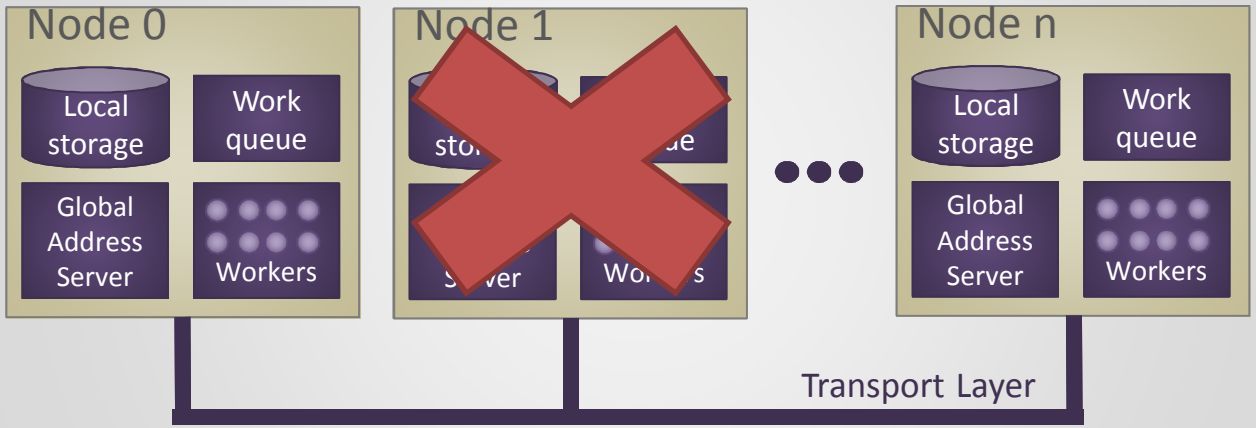
AMT has several challenges

- Recovery (beyond checkpoint/restart) is challenging
- Enormous distributed coherency problem
- Care is required to identify lost tasks due to work-stealing and asynchrony

Task Graph



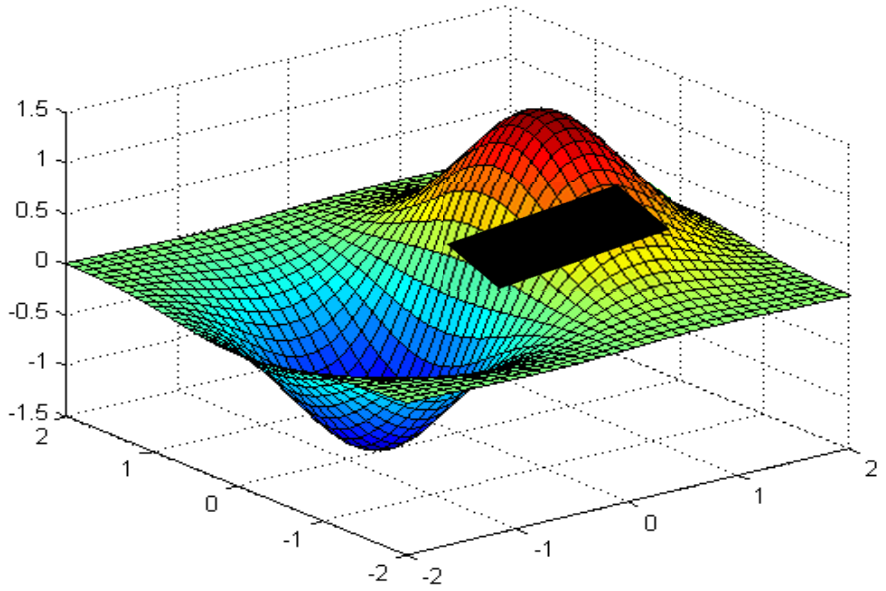
Typical AMT Runtime Architecture



Sandia National Laboratories

LOCAL FAILURE LOCAL RECOVERY

Enabling Local Recovery from Local Faults



- Current recovery model:
  Local node failure,
  global kill/restart.

- Different approach:
  - App stores key recovery data in persistent local (per MPI rank) storage (e.g., buddy, NVRAM),
    and registers recovery function.
  - Upon rank failure:
    - MPI brings in reserve HW, assigns to failed rank, calls recovery fn.
    - App restores failed process state via its persistent data (& neighbors'?).
    - All processes continue.

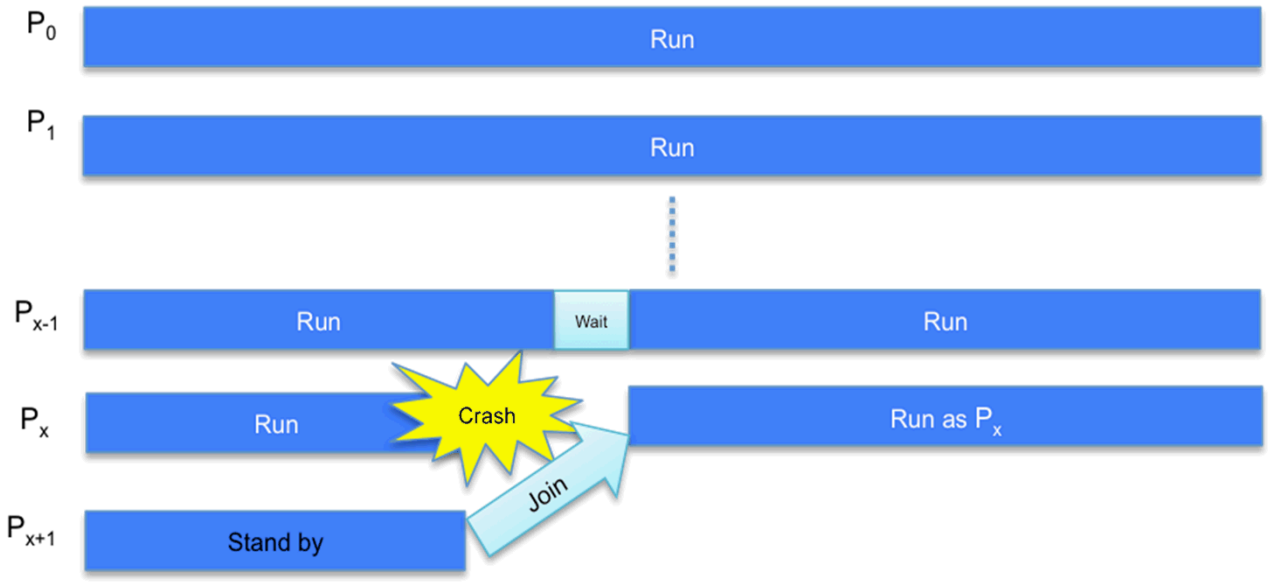
LFLR Algorithm Opportunities & Challenges

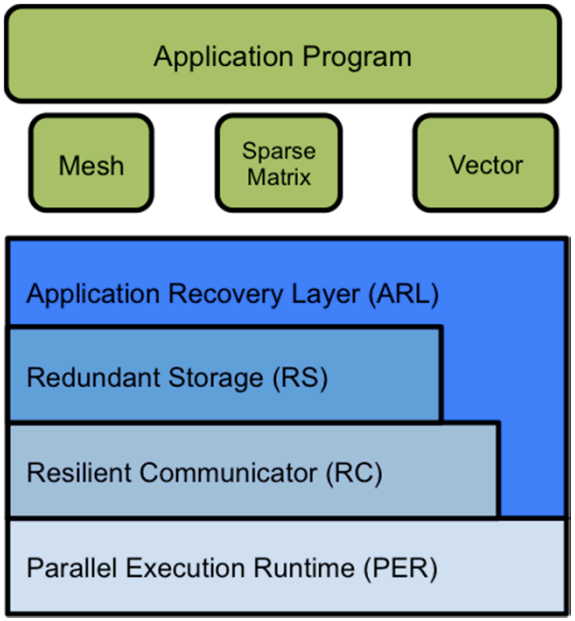
Sandia National Laboratories

- Enables fundamental algorithms work to aid fault recovery:
  - Straightforward app redesign for explicit apps.
  - Enables reasoning at approximation theory level for implicit apps:
    - What state is required?
    - What local discrete approximation is sufficiently accurate?
    - What mathematical identities can be used to restore lost state?
  - Enables practical use of many exist algorithms-based fault tolerant (ABFT) approaches in the literature.
- Lots of requirements from runtime, middleware and OS
  1. Runtime that allows a program to continue on the remaining processes after a process failure
  2. Resource management to assign compute processes for local recovery
  3. Redundant storage for data persistence and recovery
  4. Tools and framework to build application specific recovery schemes and reason the failure caused by process/node loss

LFLR Framework allows building resilience apps

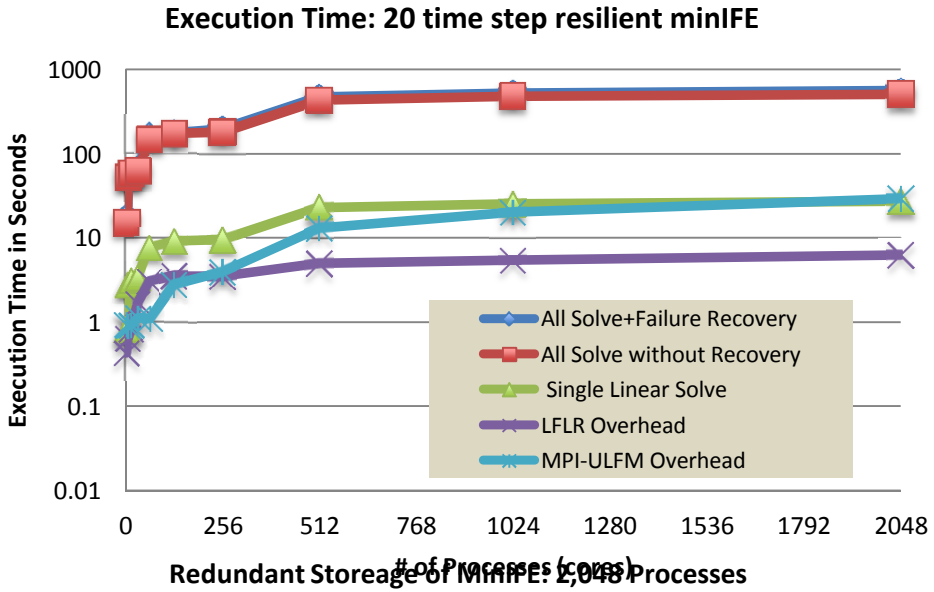
- Software framework to integrate existing apps with resilience capability
  - **The remaining processes stay alive with single process/node failures**
  - Extend data structures and classes to inherit the resilience runtime
  - Multiple implementation options for recovery
    - Roll-back, roll-forward, asynchronous, algorithm specific, etc.
  - Active Hot Spare Process for recovery

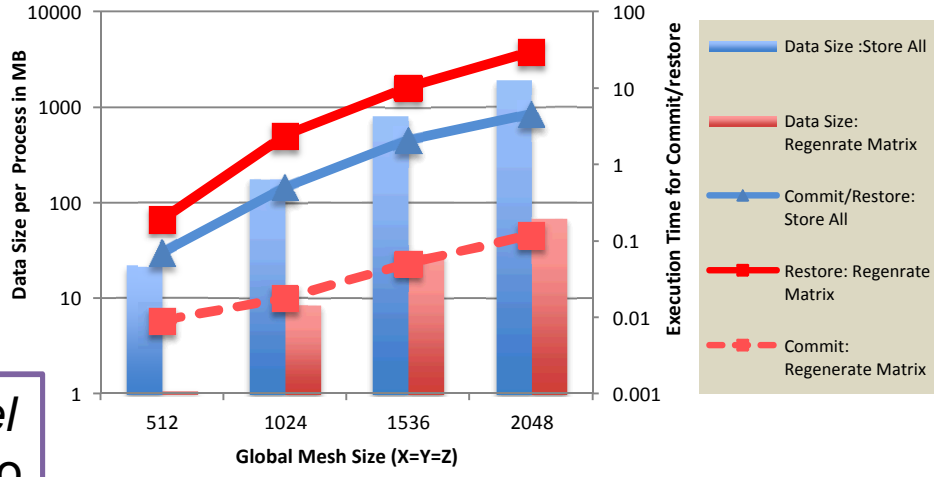




Local Failure Local Recovery: LFLR Enabled MiniFE

- Implemented a software framework to enable LFLR.
  - Software components to support 4 major requirements for LFLR
  - The LFLR-enabled MiniFE code achieves scalable recovery from **real process failures** on 2,048 processes.
- Implemented application specific recovery through the LFLR framework.
  - Significant reduction in the redundant data and **commit (checkpoint)** for MiniFE.
  - Frequent calls of **commit** amortize the relatively high overhead of **restore**.





Towards Local Failure Local Recovery Model Using MPI-ULFM, KT and Michael Heroux, to appear in EuroMPI/Asia 2014.

Sandia National Laboratories

SELECTIVE (UN)RELIABILITY

Every calculation matters

Sandia National Laboratories

Description	Iters	FLOPS	Recursive Residual Error	Solution Error
All Correct Calcs	35	343 M	4.6e-15	1.0e-6
Iter=2, y[1] += 1.0 SpMV incorrect Ortho subspace	35	343 M	6.7e-15	3.7e+3
Q[1][1] += 1.0 Non-ortho subspace	N/C	N/A	7.7e-02	5.9e+5

- Small PDE Problem: ILUT/GMRES
- Correct result:35 Iters, 343M FLOPS
- 2 examples of a single bad op.
- Solvers:
  - 50-90% of total app operations.
  - Soft errors most likely in solver.
- Need new algorithms for soft errors:
  - Well-conditioned wrt errors.
  - Decay proportional to number of errors.
  - Minimal impact when no errors.

Soft Error Resilience

- New Programming Model Elements:
  - SW-enabled, highly reliable:
    - Data storage, paths.
    - Compute regions.
- Idea: *New algorithms with minimal usage of high reliability.*
- First new algorithm: FT-GMRES.
  - Resilient to soft errors.
  - Outer solve: Highly Reliable
  - Inner solve: "bulk" reliability.
- General approach applies to many algorithms.

Fault-tolerant linear solvers via selective reliability,
Patrick G. Bridges, Kurt B. Ferreira, Michael A. Heroux, Mark Hoemmen
arXiv:1206.1390v1 [math.NA]

What is Needed for Selective Reliability?

Sandia National Laboratories

- A lot, lot.
- A programming model.
  - Expressing data/code reliability or unreliability.
- Algorithms.
  - Basic approaches:
    - Nest an unreliable algorithm in a reliable version of the same.
    - Dispatch unreliable task subgraph from reliable graph node.
- Lots of runtime/OS infrastructure.
  - Provision of reliable data, paths, execution.
  - Portable interfaces to HW solutions.
- Hardware support?
  - Special HW components that are
    - slower and more reliable or
    - faster and less reliable

Selective Reliability enabled by Programming Language Extensions

Sandia National Laboratories

Code Sections

Syntax:

```
#pragma robust-detect sentinel <variable = . . . >
{
    <code>
}
#pragma robust-correct sentinel <variable = . . . >
{
    <code>
}
```

Semantics:

Automatic duplicate/triplicate threads spawned by runtime system

Compiler injected code transformation for error detection/majority voting

A Programming Model for Resilience in Extreme Scale Computing, Saurabh Hukerikar, Pedro Diniz and Bob Lucas, FTXS-12

FT-GMRES Algorithm

Sandia National Laboratories

Input: Linear system $Ax = b$ and initial guess x_0
  $r_0 := b - Ax_0$, $\beta := \|r_0\|_2$, $q_1 := r_0/\beta$
  for $j = 1, 2, \ldots$ until convergence **do**
    Inner solve: Solve for z_j in $q_j = Az_j$
    $v_{j+1} := Az_j$
    for $i = 1, 2, \ldots, k$ **do** ▷ Orthogonalize v_{j+1}
      $H(i,j) := q_i^* v_{j+1}$, $v_{j+1} := v_{j+1} - q_i H(i,j)$
    end for
    $H(j+1, j) := \|v_{j+1}\|_2$
    Update rank-revealing decomposition of $H(1:j, 1:j)$
    if $H(j+1, j)$ is less than some tolerance **then**
      if $H(1:j, 1:j)$ not full rank **then**
        Try recovery strategies
      else
        Converged; return after end of this iteration
      end if
    else
      $q_{j+1} := v_{j+1}/H(j+1, j)$
    end if
    $y_j := \text{argmin}_y \|H(1:j+1, 1:j)y - \beta e_1\|_2$ ▷ GMRES projected problem
    $x_j := x_0 + [z_1, z_2, \ldots, z_j]y_j$ ▷ Solve for approximate solution
  end for

"Unreliably" computed.
Standard solver library call.
Majority of computational cost.

Captures true linear operator issues, AND
Can use some "garbage" soft error results.

Resilient Application Programming

- Standard approach:
  - System over-constrains reliability
  - “Fail-stop” model
  - Checkpoint / restart
  - Application is ignorant of faults
- New approach:
  - System lets app control reliability
  - Tiered reliability
  - “Run through” faults
  - App listens and responds to faults

What Auto-tuning can do?

Sandia National Laboratories

- Auto-tuning can do:
  - Parameter exploration and optimization
    - Off-line and on-fly
  - Code generation
  - Code (method) selection
- Extend optimization space:
  - Performance (execution time, memory and power usage)
  - **Reliability (e.g. 99% for 7 days)**
  - **Error in output (against failure-free environment)**

Auto-Tuning can explore complex models of HPC resilience

Sandia National Laboratories

- Exploitation of multiple resilient programming models
  - What is the good combination?
  - New Performance Tuning Methodologies
    - Skeptical Programming + rBSP + Selective Reliability
- Auto-tuning facilitate to investigate some co-design questions
- (Caveat) Both hard and soft failures are typically emulated by software
  - We cannot use neutron beam everyday

Runtime VS Off-line Tuning

Sandia National Laboratories

- Runtime tuning requires a good infrastructure support
  - Redundant threads and processes
    - Threads (USC)
    - Processes (SNL, Rutgers, TiTech&LLNL)
  - Need middleware/system support
    - Active Harmony (UMD)
  - Compiler assisted tuning (applicable for off-line tuning)
    - Rose-FTT (LLNL) and LLVM (LLNL, Utah)

- Off-line tuning requires a good fault model in addition to emulated fault injections

Conclusions

Sandia National Laboratories

- Resilience is an imminent issue in HPC
- 4 Different Programming Models
  - Skeptical Programming. (SP)
  - Relaxed Bulk Synchronous (rBSP)
  - Local-Failure, Local-Recovery (LFLR)
  - Selective (Un)reliability (SU/R)
- Auto-tuners should explore these models to optimize scalability, performance and resilience together.

Acknowledgement

- DoE NNSA ASC
- DoE Office of Science ASCR
- Michael Heroux and Mark Hoemmen (SNL, Albuquerque)
- Rob Armstrong, Janine Bennett, Robert Clay, John Floren, Hemanth Kolla, Jackson Mayo, Jaideep Ray, Nicole Slattengren and Jeremiah Wilke (SNL, Livermore)
- Marc Gamell and Manish Parashar (Rutgers U)
- Saurabh Hukerikar, Pedro Diniz and Bob Lucas (ISI, USC)
- James Elliott (NCSU)
- George Bosilca (U of Tennessee)