Assessing Fault Sensitivity in MPI Applications

Charng-Da Lu

Center for Computational Research

SUNY at Buffalo

Daniel A. Reed

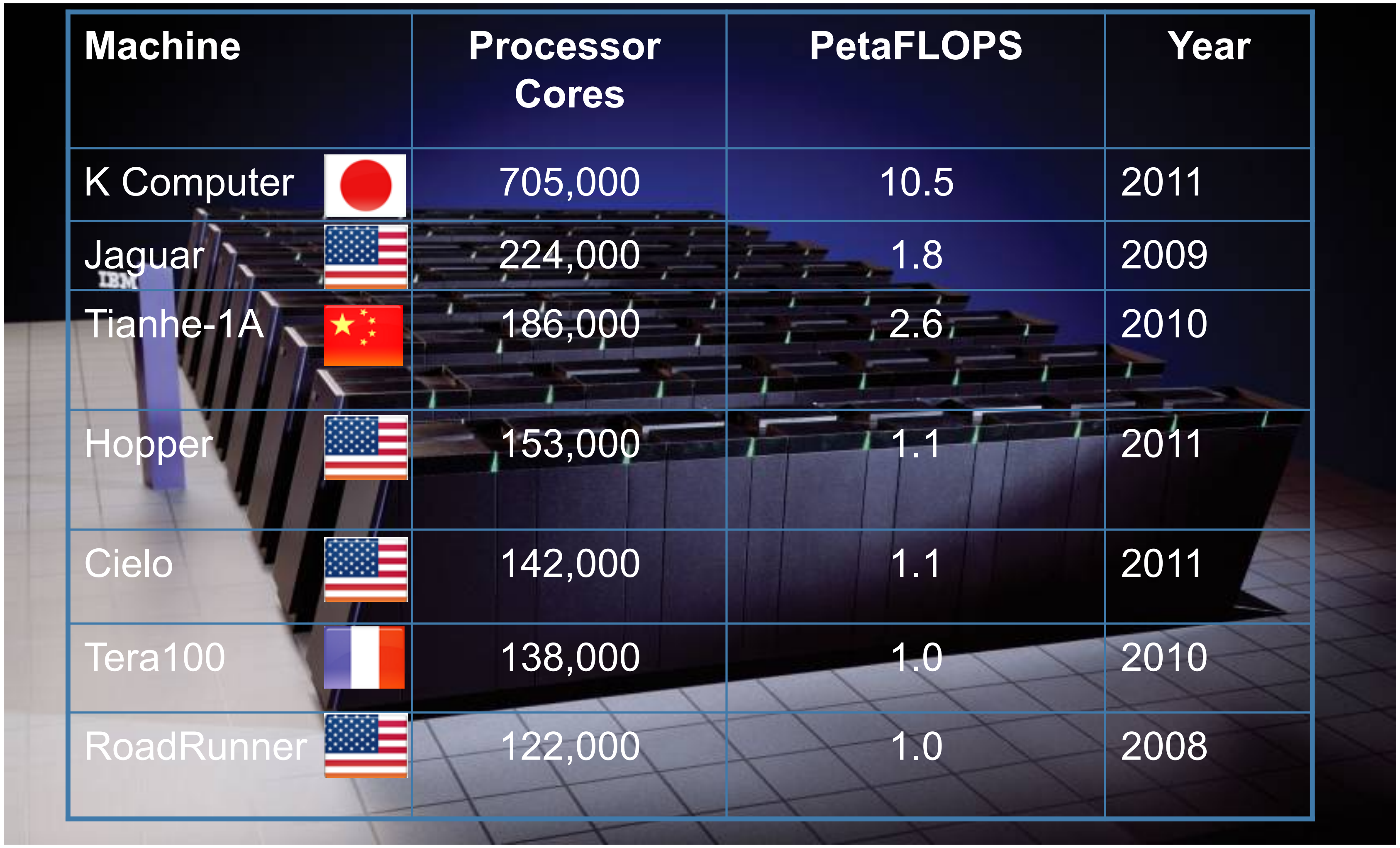
Microsoft Research

CENTER FOR COMPUTATIONAL RESEARCH
University at Buffalo The State University of New York

Outline

- **Introduction**
  - background and motivations
  - reliability challenges of large PC clusters
- **Failure modes**
  - memory and communication errors
- **Fault injection experiments**
  - methodology and experiments
  - analysis and implications
- **Conclusions**
  - large-scale cluster design
  - software strategies for reliability

Large Computing Systems



Machine	Processor Cores	PetaFLOPS	Year
K Computer 🇯🇵	705,000	10.5	2011
Jaguar 🇺🇸	224,000	1.8	2009
Tianhe-1A 🇨🇳	186,000	2.6	2010
Hopper 🇺🇸	153,000	1.1	2011
Cielo 🇺🇸	142,000	1.1	2011
Tera100 🇫🇷	138,000	1.0	2010
RoadRunner 🇺🇸	122,000	1.0	2008

- **Dominant constraints on size**
  - power consumption, reliability and usability

Node Failure Challenges

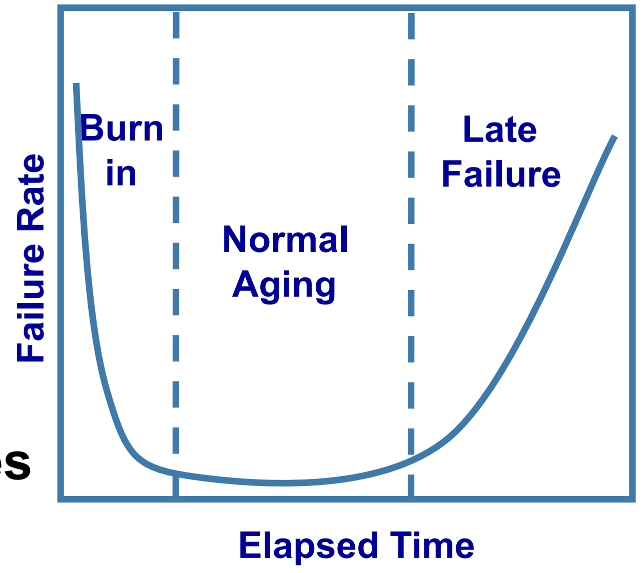
- **Domain decomposition**
  - spreads vital data across all nodes
  - each spatial cell exists in one memory
    - except possible ghost or halo cells
- **Single node failure**
  - causes blockage of the overall simulation
  - data is lost and must be recovered
- **"Bathtub" failure model operating regimes**
  - infant mortality
  - normal mode
  - late failure mode
- **Simple checkpointing helps; the optimum interval is roughly**

$$\tau = \sqrt{2\delta(M + R)}$$

where δ is time to complete a checkpoint

M is the time before failure

R is the restart time due to lost work



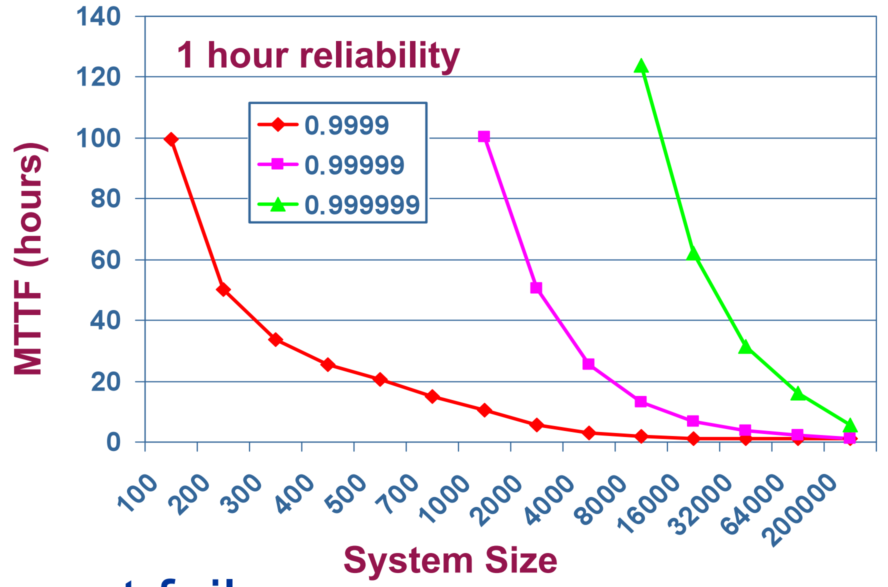
Large Systems Reliability

Machine	Core Count	Reliability
ASCI Q	8,192	MTBI 6.5 hr. 114 unplanned outages/month. HW outage sources: storage, CPU, memory *
ASCI White	8,192	MTBF 5 hr ('01) and 40 hr ('03) HW outage sources: storage, CPU, 3rd party hardware **
NERSC Seaborg	6,656	MTBI 14 days. MTTR 3.3 hr Availability 98.74%. SW is main outage source. ***
PSC Lemieux	3,016	MTBI 9.7 hr Availability 98.33% ****
Google	~15,000	20 reboots/day. 2-3% machines replaced/year. HW outage sources: storage, memory *****

*J. Morrison (LANL): "The ASCI Q System at Los Alamos," SOS7, 2003
** M. Seager (LLNL): "Operational machines: ASCI White," SOS7, 2003
*** http://hpcf.nersc.gov/computers/stats/AvailStats
**** M. Levine (PSC): "NSF's terascale computing system," SOS7, 2003
***** J. Hennessy et al, "Computer Architecture: A Quantitative Approach", 3rd edition, 2002

Large System Reliability

- **Facing the issues**
  - component MTBF
  - system size
  - usable capability

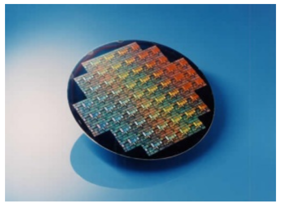


- **A few assumptions**
  - assume independent component failures
    - » an optimistic and not realistic assumption
  - *N* is the number of processors
  - *r* is probability a component operates for 1 hour
  - *R* is probability the system operates for 1 hour
- **Then** $R = r^N$ **or** $R \approx \frac{1}{e^{\lambda N}}$ **for large *N***

Component Reliability

- **Two basic types**
  - hard (permanent) errors
  - soft (recoverable) errors
- **Hard errors**
  - permanent physical defects
  - memory: 160-1000 years MTBF for 32-64 Mb DRAM chips
  - disk: 50-100 years MTBF (?)
  - node: 3-5 years (warranty period)

- **Soft errors**
  - transient faults in semiconductor devices
    - » alpha particles, cosmic rays, overheat, poor power supplies, ..
  - ECC memory isn't 100% secure
    - » 80-95% protection rate
  - much more likely than hard errors
    - » 10 days MTBF for 1GB RAM
  - continues to worsen as chip geometries shrink



Memory Soft Error Rates

Memory Type	MTBF in days (1 GB)
Commercial CMOS memory	0.8
4M SRAM	> 1.2
1Gb memory (NightHawk)	2.3
SRAM and DRAM	2.6-5.2
8.2 Gb SRAM (Cray YMP-8)	4
SRAM	5.2
256 MB	7.4
160 Gb DRAM (FermiLab)	7.4
32 Gb DRAM (Cray YMP-8)	8.7
MoSys 1T-SRAM (no ECC)	10.4
Micron estimates, 256 MB	43-86

Source: Tazzaron Semiconductor, “Soft Errors in Electronic Memory – A White Paper”

Communication Errors

- **Soft errors occur on networks as well**
  - routers, switches, NICs, links ...
- **Link-level checksum = Reliable transmission?**
  - Stone and Patridge's study* shows
    - » probability of Ethernet's 32-bit CRC not catching errors
      - ■ 1/1,100 to 1/32,000
  - theoretically, it should be 1/(4 billion)
- **To make things worse**
  - performance-oriented computing favors OS-bypass protocols
    - » relative to TCP
  - message integrity solely relies on link-level checksum

* J. Stone and C. Partridge "When the CRC and TCP checksum disagree" in ACM SIGCOMM 2000

Terminology

- **Error/failure**
  - system behavior deviates from specification
  - omission
    - » occasionally no response…
  - response
    - » incorrect
  - performance
    - » response is correct but not timely
  - crash/hang

is the source of

is the manifestation of

- **Fault**
  - single event upset
    - » bit flips
  - single event burnout
    - » power surge
  - Bohrbug
    - » determinism
  - Heisenbug
    - » race condition
    - » rare input
  - ageing
    - » resource exhaustion

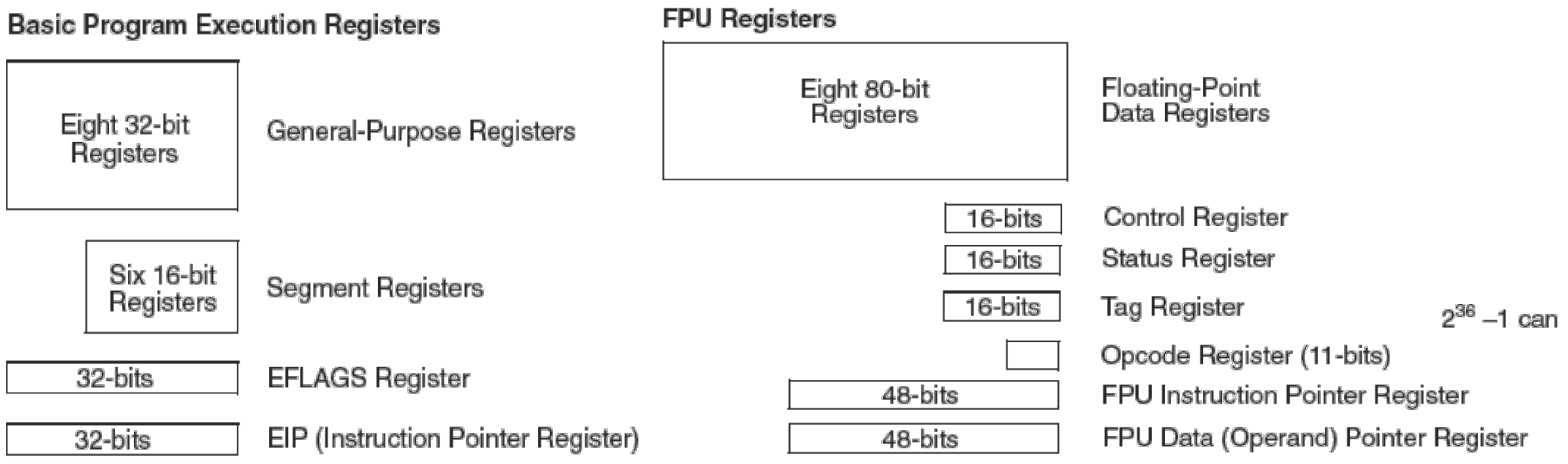
Experiments

- **Goal: study the impact of bit-flip faults on MPI codes**
- **Rationale**
  - it is easier to detect hard errors and assess their damage
  - what about transient faults?
  - crash? hang? incorrect output? …

- **Approach: fault injection**
  - ✓ Software-based
    - inexpensive and portable
    - targets a wide range of components
    - OS, libraries, applications ...
    - address bus, ALU, memory ...
  - Hardware-based
    - expensive
    - heavy ion bombarding or lasers
    - pin-level probes and sockets
    - Alpha particles, bit-flips, power surge, 0/1 stuck-at ...

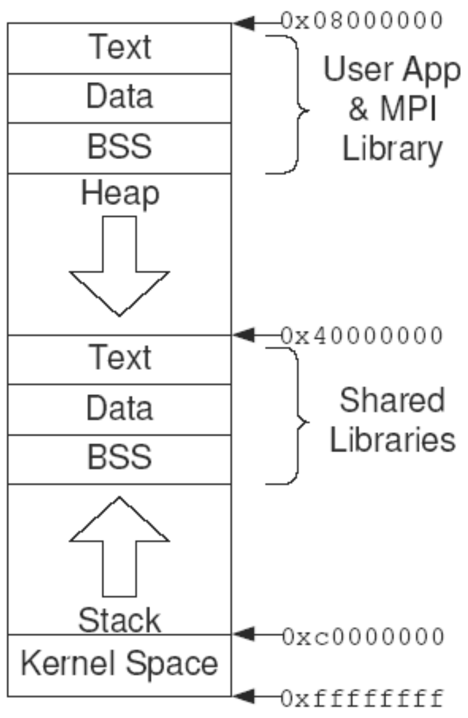
Register Fault Injection

- **Processor (x86)**
  - User-space injection
  - Regular registers and x87 FPU registers
  - No injection to special purpose registers (need root privilege)
    - » System control registers, debug and performance registers
    - » Virtual memory management registers, MMX/SSE..
  - No injection to L2/L3 caches, TLB


Basic Program Execution Registers
Eight 32-bit Registers
General-Purpose Registers
Six 16-bit Registers
Segment Registers
32-bits
EFLAGS Register
32-bits
EIP (Instruction Pointer Register)
FPU Registers
Eight 80-bit Registers
Floating-Point Data Registers
16-bits
Control Register
16-bits
Status Register
16-bits
Tag Register
Opcode Register (11-bits)
48-bits
FPU Instruction Pointer Register
48-bits
FPU Data (Operand) Pointer Register
2^36 −1 can

Memory Fault Injection

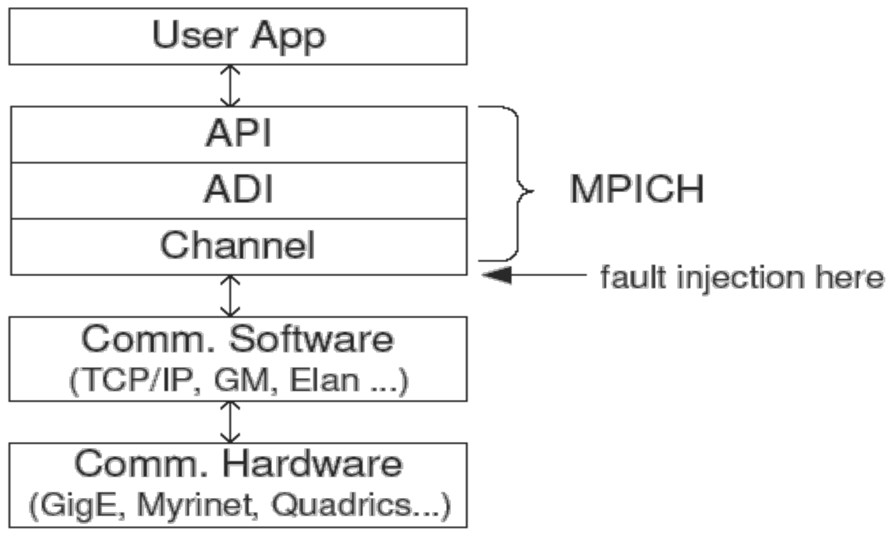
- **Memory**
  - Focus on application memory
  - Injection addresses have uniform distribution.
  - Skip library memory
    - » MPI and shared libraries
  - Text, Data, BSS
  - Heap and stack



Linux Process Memory Model

Message Fault Injection

- **Simulate faults that link-level checksums miss**
  - Use MPICH for communication
  - Inject at the level closest to operating system
    - » but avoid perturbing the operating system (for testability)
  - Can affect all kinds of messages
    - » Control, point-to-point, collective operations…

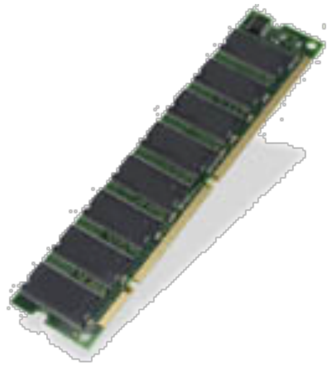


Memory Fault Injector

- `ptrace` **UNIX system call**
  - Attach to and halt a host process
  - Peek/poke register and memory contents (like gdb)

- **Static objects (Text, Data, BSS)**
  - Used **`nm`** and **`objdump`** utilities to find the range of injection
  - Skipped all MPI objects

- **Dynamic objects (Heap and stack)**
  - Created customized **`malloc/free`**
    - » separates application objects from MPI objects
  - Examined return addresses in stack frames
    - » determine the range of stack injection



Message Fault Injector

- **MPICH**
  - Developed by Argonne National Laboratory
  - Highly portable MPI implementation
  - Adopted by many hardware vendors

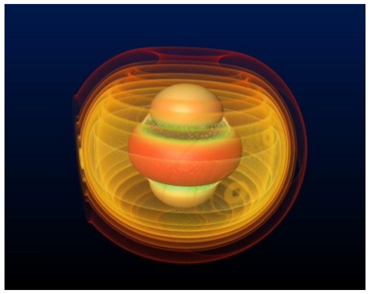
- **Fault injector**
  - Modified MPICH library
  - Uses “ch_p4” channel (TCP/IP)
  - Faults injected in the payload
    - » immediately after receipt from a socket
  - Both MPICH and user applications are vulnerable to message faults

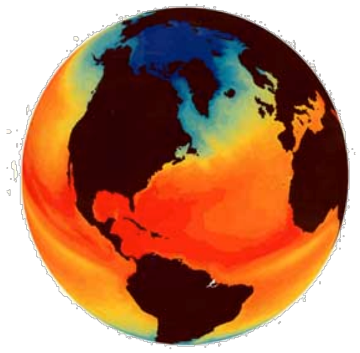
Experimental Environment

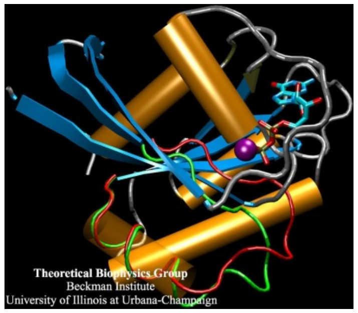
- **A meta-cluster formed from two clusters**
  - Rhapsody
    - » 32 dual 930 MHz Pentium III nodes
    - » 1 GB RAM/node
    - » 10/100 Gigabit Ethernet
  - Symphony
    - » 16 dual 500 MHz Pentium II nodes
    - » 512 MB RAM/node
    - » Ethernet and Myrinet

Fault Assessment Code Suite

- **Cactus Wavetoy**
  - PDE solver for wave functions in physics
  - Test problem
    - » 150x150x150 for 100 steps
    - » 196 processes
- **CAM**
  - **C**ommunity **A**tmospheric **M**odel
  - Test problem
    - » default test dataset for 24 hours of simulated time
    - » 64 processes
- **NAMD**
  - Molecular dynamics code
  - Test problem
    - » 92,000 atoms and 20 steps
    - » 96 processes

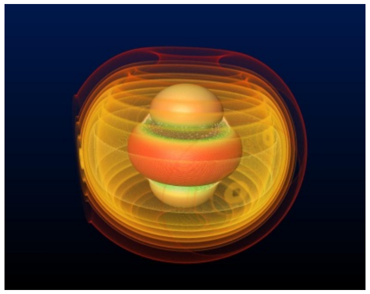


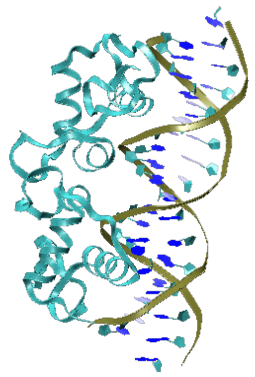



Theoretical Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

Test Code Suite Characteristics

Injection Location	Cactus	NAMD	CAM
Memory	1.1 MB	25-30 MB	80 MB
Text Size	330 KB	2 MB	2 MB
Data Size	130 KB	110 KB	32 MB
BSS Size	5 KB	598 KB	38 MB
Heap Size	450-500 KB	22-27 MB	8 MB
Message	2.4-4.8 MB	13-33 MB	125-150 MB

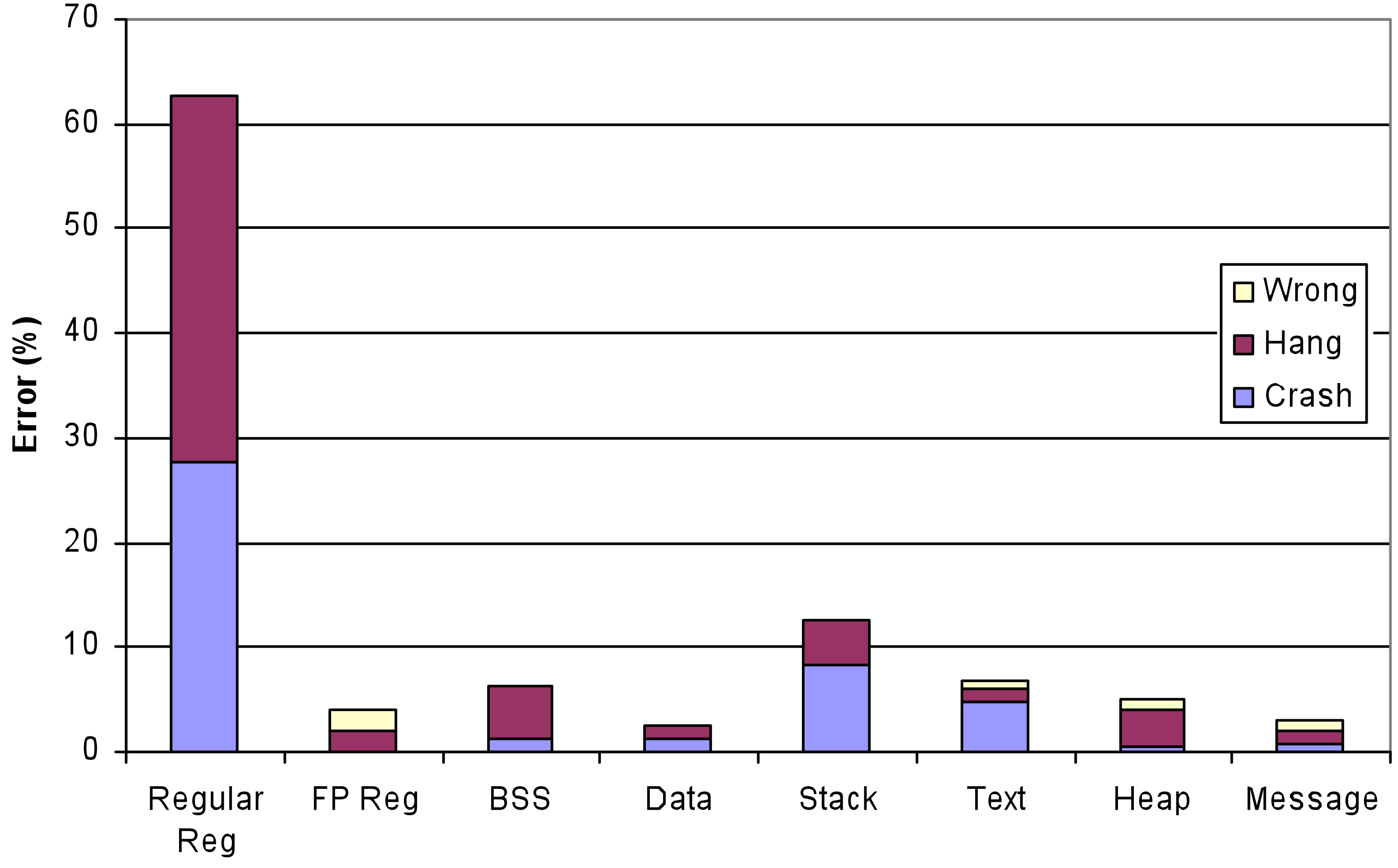




Experimental Fault Assessment

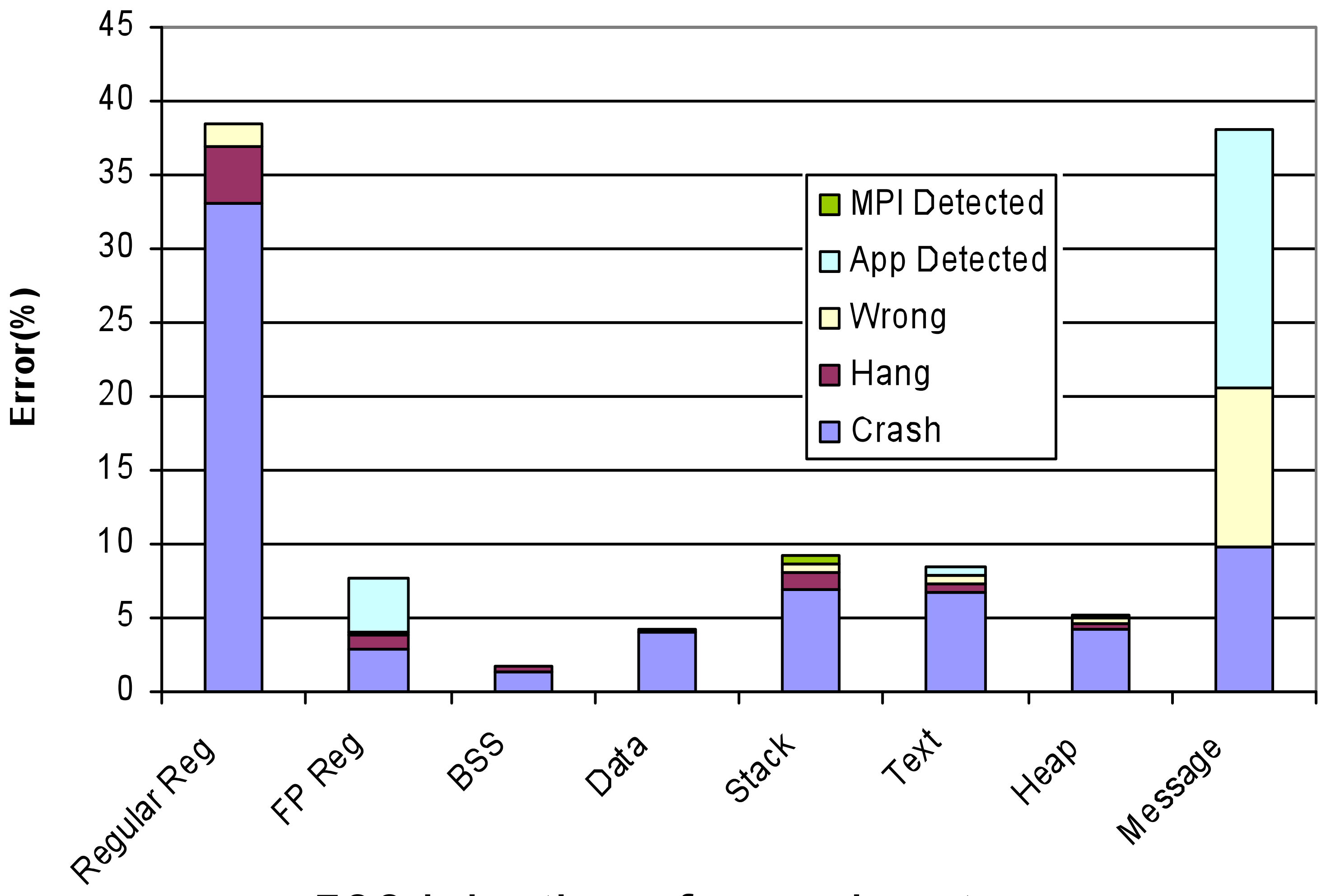
- **Failure modes**
  - Application crash
    - » MPI error detected via MPI error handler
    - » Application detected via assertion checks
    - » Other(e.g., Segmentation fault)
  - Application hang (no termination)
  - Application execution completion
    - » correct (fault not manifest) or incorrect output

Cactus Wavetoy Results



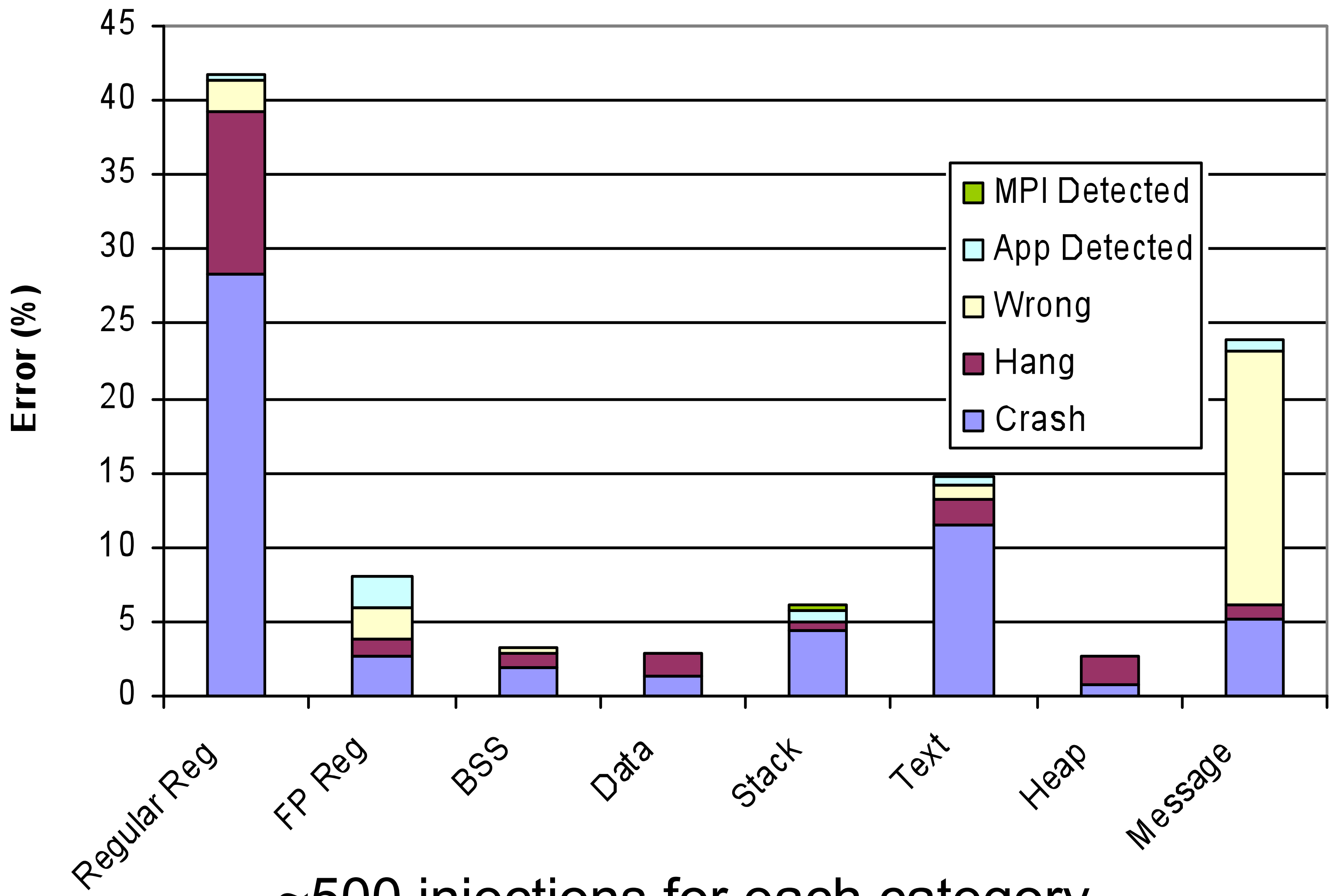
500-2000 injections for each category

NAMD Results



~500 injections for each category

CAM Results



~500 injections for each category

Register Injection Analysis

- **Registers are the most vulnerable to transient faults**
  - 39-63% error rate overall
  - Results could depend on register management
    - » Live register allocation and size of register file
    - » Optimization increases register use
- **Error rates for floating point registers are much lower**
  - 4-8% error rate
  - Most injections into control registers do not generate errors
    - » Except the Tag Word register, which turns a number into NaN
  - Injections into data registers do not yield high error rates
    - » At most 4 out of 8 data registers are in use
    - » A data register is actually 80-bit long, but only 64 bits can be read out.

Memory Injection Analysis

- **Error rates for memory injections are very low**
  - 3-15% error rate
  - Spatial locality: Memory is not accessed
  - Temporal locality: Memory is overwritten before reuse

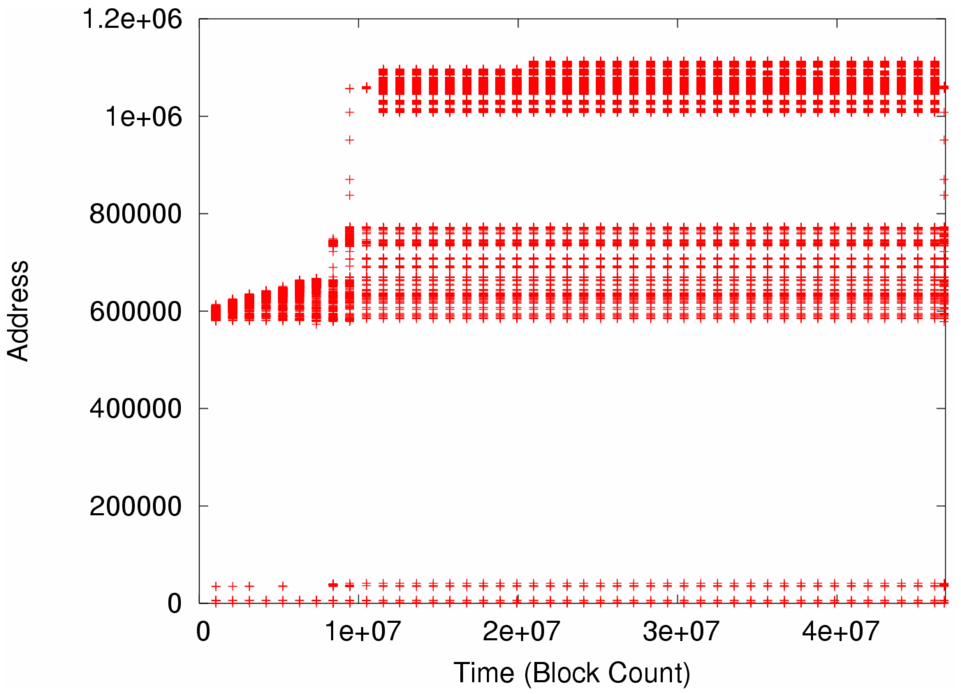
- **Working set analysis**
  - To understand memory access behavior
  - Collected memory load data
    - » Using Valgrind, an open-source x86 memory debugging tool

Working Set Analysis

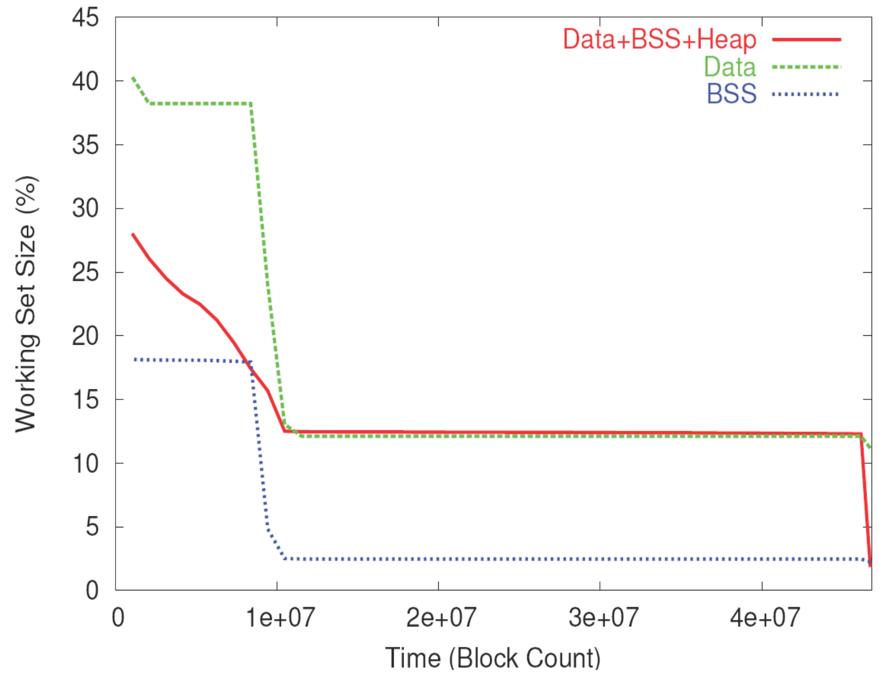
- **Definition of working set at time *t***
  - Size of accessed memory since *t*
  - Non-increasing
- **Larger working size → Higher chance of fault-induced errors**

Memory Access Behavior

- **Cactus Wavetoy phase behavior**
  - Initialization and computation phases
  - working set size drops in computation phase (28% to 12%)

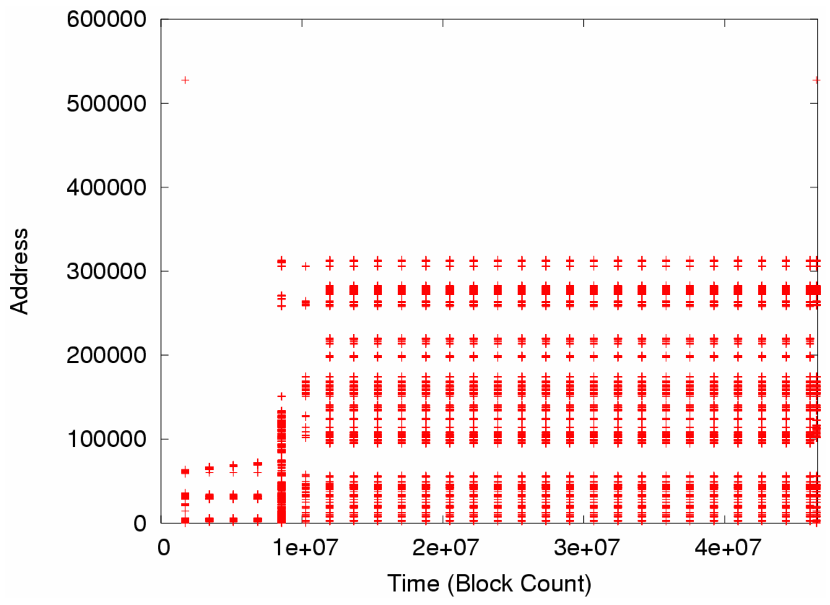


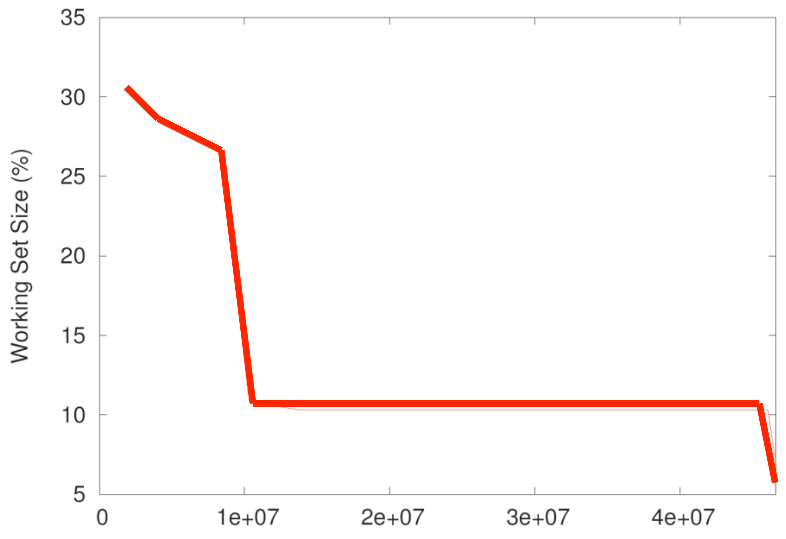
Data+BSS+Heap



Memory Access Behavior

Text





Message Injection Analysis

- **NAMD and CAM are sensitive to message faults**
  - 38% and 24% error rates, respectively
- **NAMD**
  - Built-in message integrity checks are lightweight and effective
  - 46% of errors are detected, only 28% of errors are incorrect output
- **CAM**
  - only 3% of errors are caught, 71% of errors are incorrect output
- **Cactus Wavetoy’s error rate is very low**
  - The output we used to verify correctness is in plain text format
  - Low order decimal digits are not reported
  - Only perturbation in significant bits will manifest in a short run
  - After more steps of execution, the error will manifest

What is an Exascale System?

INTERNET IN A BOX™

The Complete Internet Solution

- **Embrace failure, complexity, and scale**
  - a mind set change

Failures and Autonomic Recovery

- **10^6 hours for component MTTF**
  - Sounds like a lot until you divide by 10^5!
- ***It's time to take RAS seriously***
  - *Systems do provide warnings*
    - » *Soft bit errors – ECC memory recovery*
    - » *Disk read/write retries, packet loss and retransmission*
  - *Status and health provide guidance*
    - » *Node temperature/fan duty cycles*
- **Software and algorithmic responses**
  - Diagnostic-mediated checkpointing
  - Algorithm-based fault tolerance
  - Domain-specific fault tolerance
  - Loosely synchronous algorithms
  - Optimal system size for minimum execution time

Fault Tolerance Support in MPI

- **MPI is a standard, not an implementation**
  - MPI standard: "After an error is detected, the state of MPI is undefined"
  - Most implementations: Abort whenever there is *any* error.
- **What about `MPI_Errhandler_set` API in MPI 1 ?**
  - Not what you think !
  - Only handles semantic errors such as sending messages to a non-existing MPI process.
- **What about MPI 2 standard?**
  - Can spawn MPI processes dynamically.
  - Has `listen`/`accept`/`connect` BSD socket-like APIs.
- **MPI 3 work-in-progress**
  - Redefines MPI semantics: e.g. Failed MPI processes treated as non-existing MPI processes
  - MPI 3 FT Working Group: http://www.mpi-forum.org

Conclusions

- **The most damaging soft bit errors**
  - Register and message contents
- **Memory errors, albeit less likely**
  - Are still a critical failure mode for large systems
- **Application internal checks can catch errors**
  - Defensive programming is important at scale
- **MPI Standard**
  - Supports very minimal error detection and recovery
  - Fault-tolerant MPI support and extensions are needed
- **It’s time to take reliability seriously**
  - RAS is critical to continued system scaling