The Challenges of Dependable High-Performance Computing at Exascale

Mootaz Elnozahy
KAUST

Summary

- Exascale systems:
  - Seriously constrained in energy, resources and performance
  - Errors may creep in undetected
    - Estimate: 1 in 15 days
- MTBF to go lower and more capacity to be lost due to:
  - At least a tenfold increase in future system size & number of components
  - Reduced reliability due to SER by almost an order of magnitude with each 0.5v reduction
  - Interactions with power and power management
- Need a new research agenda whose goals are for exa-scale systems:
  - Reduce the capacity lost to failures and recovery to 1%
  - Bring MTBF to at least 15 days

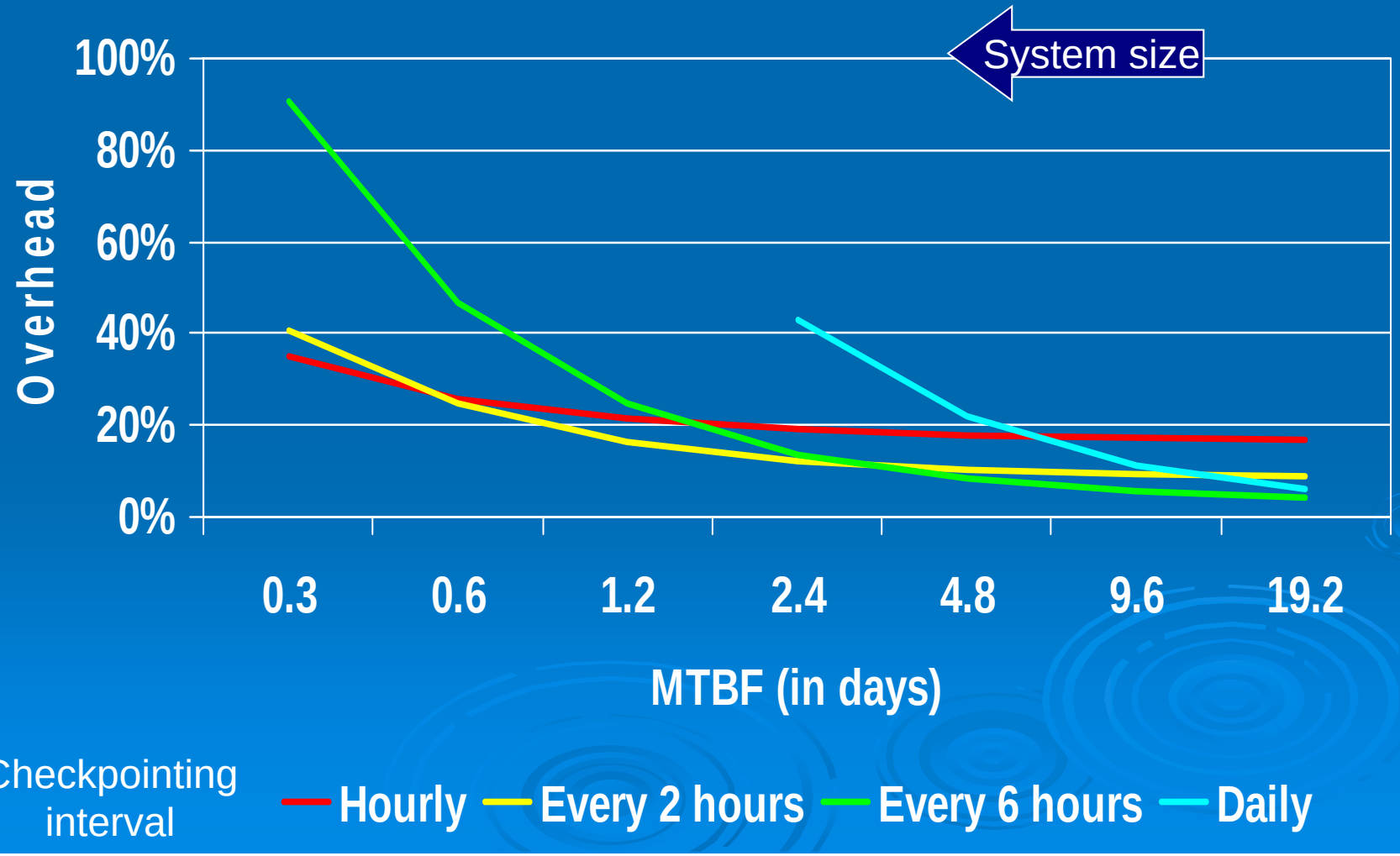
Reliability in Large Scale Systems

- Commercial (Big Data)
  - Centered around data availability for many short computations
  - Measured in 9's (e.g. banking apps require 99.999)
  - Techniques used include transactions, bag of tasks (map-reduce, hadoop, etc), dht's, and/or replication
- High-Performance
  - Centered around completing a single long computation
  - No agreed-upon system of measurements
  - Techniques used mostly checkpoint and restart

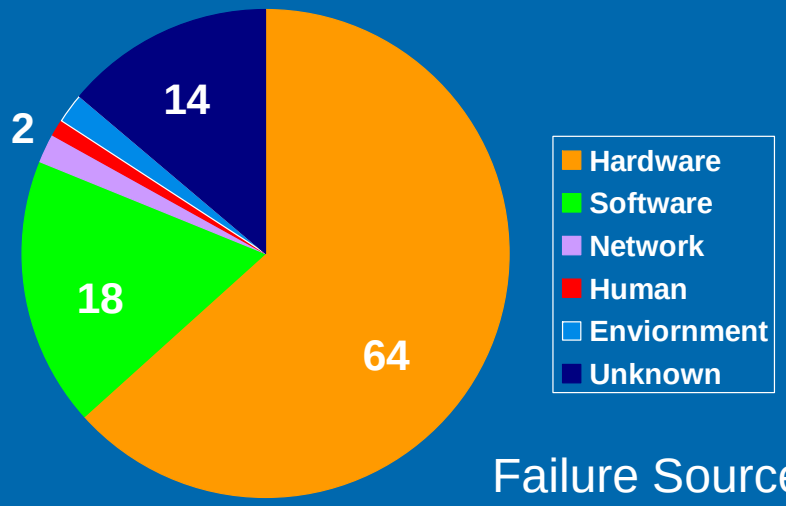
Checkpointing Hits its Limit

- Today, systems waste up to 20% of their capacity because of reliability problems
  - Choice of checkpointing interval ad hoc
    - Too frequent, wasting precious cycles to excessive checkpointing
    - Infrequent, increasing impact of failures
  - Primitive techniques used therefore high checkpointing overhead
- These overheads translate to wasted $'s:
  - Loss of the amortized cost of capital
  - Loss due to energy consumed in checkpointing and recovery
  - Loss of opportunity in delaying results

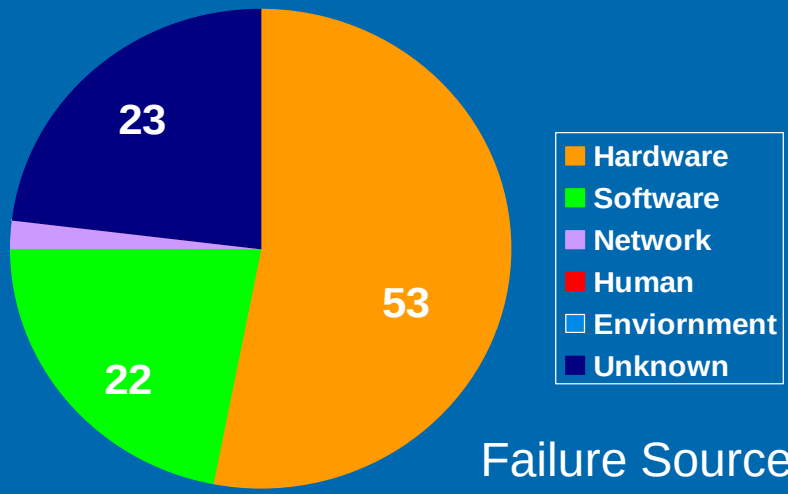
Effect of MTBF



Failures Occur and With Frequency



Failure Sources
LANL



Failure Sources
Other

- Study surveyed 22 production-level systems at LANL (various processor types and sizes)
- One large system at an undisclosed location, 512 processor/node (x86)
- Hardware failures: CPU, memory
- Software failures: OS, parallel file system, middleware/application
- Data sources: Schroeder & Gibson, DSN ’07, others

The Real Problem at Exascale

- Failure containment is poor
  - One failure results in total system rollback
  - As system size increases, MTBF will be lower and therefore more system-level rollbacks
  - Can be mitigated by checkpointing with synchronous message logging but this is very expensive
- Failure propagation is property of the programming style and model
  - Can be mitigated by inserting some form of transactions into the system, but this is seldom done

Failures will not be Clean

- Existing art detects a variety of failures and converts them into a crash failure
  - E.g. Soft errors detected by hardware are reported to OS, which issues a panic call
  - Similar for system software bugs
  - Dealing with crash failure is clean
  - But what about silent failures that escape the detection mechanism?
- Failure coverage may be limited, poorly documented

Power Management Mechanisms

- Many components will consume significant power, e.g. cores, memories, storage, supply losses
- Power management mechanisms will have to include:
  - Slowing resources down (DVFS cores independently, multi-speed disks), voluntary or involuntary
  - Deactivating resources (individual cores, cache lines, memory ranks, disks)
  - Moving workloads around (to create idleness or control temperature)
  - Capping power consumed by specific subsystems
- Power management will be performed by a combination of programmer hints, system software (runtime, OS, VMM, and/or load distribution system), and hardware

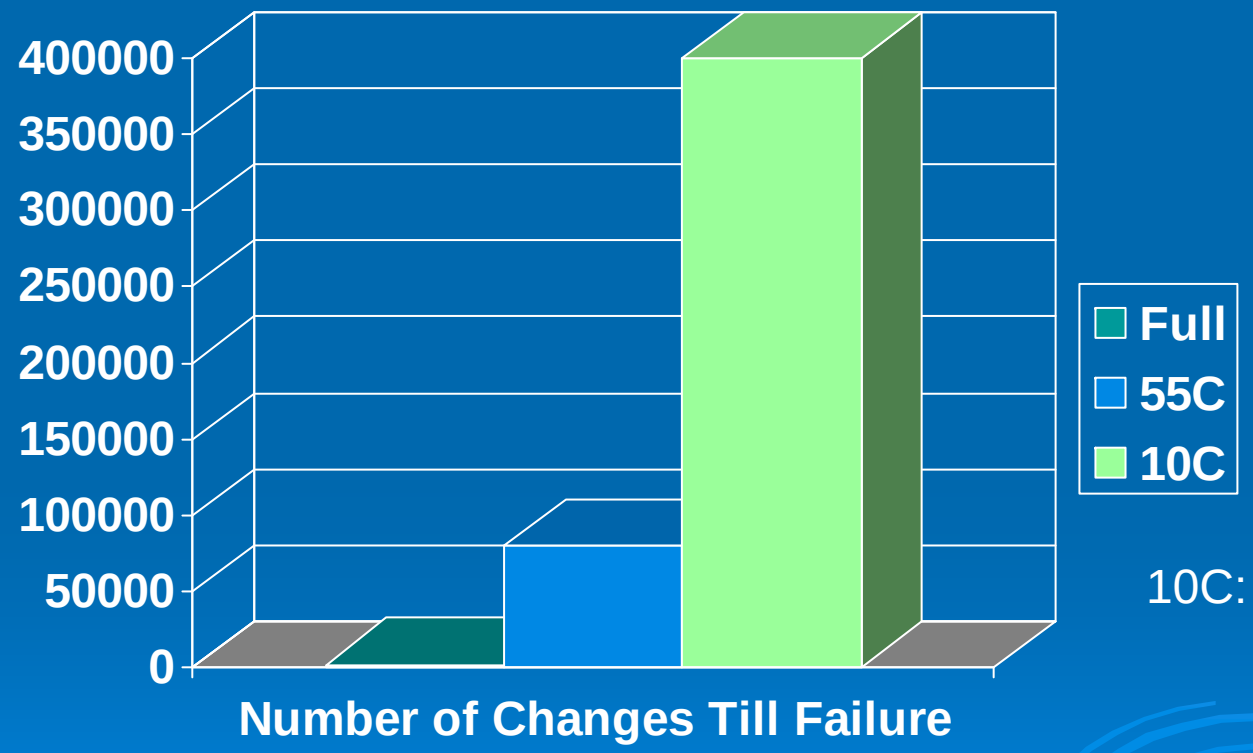
Focus on HPC Systems

- Administrators will use system management interface to specify:
  - Performance-power tradeoffs
  - Power consumption policies at various granularities (data center, specific rows of racks, rack-level, system level, etc.)
  - Specify power caps on specific subsystems
- Power saving opportunities in HPC systems include
  - Setting cores to low-power modes (e.g., when there are fewer threads than cores, when workload is memory-bound, or while waiting for a barrier to complete)
  - Conserve memory/disk energy by intelligently placing data and setting ranks/disks to low power (e.g. between checkpoints)
  - Control temperature by (re-)distributing load in response to hot spots and by dynamically managing fans and ACs

Power Management and Reliability

- Power management will create thermal variations that will induce mechanical stresses at chip and board levels
  - Mechanical stresses may lead to board-level failures in the form of separated connections, shorts
  - Mechanical and thermal stresses may affect the longevity of chips and the system in general
- Data center power grid may be subjected to sharper power swings of potentially MW/microseconds range
  - No one has experience with the effect of simultaneously reducing the voltage (and hence the power) of 100,000's processors and associated memory banks and disks within a few microseconds
  - No one has experience with doing this on a continuous basis at the scale envisioned for future large-scale systems

Effect of Thermal Cycles



10C: Estimates

Power management is useful, but if done carelessly it can impact reliability
Need: Temperature and reliability-aware power management algorithms

Power Management and Reliability

- Two opposing effects due to operation at lower voltage: I
  - Will reduce tolerance to soft errors
  - But will reduce temperature and white noise
- Power management will cause system speed at all or different components to vary
  - Software is designed and tested with all components running at the same speed
  - May expose timing bugs (at all levels of software)
- Disk reliability may be affected:
  - Longevity of disks inversely proportional to the number of power cycles
  - Varying the motor speed induces mechanical stresses
  - No models or understanding of the effect on reliability exists

Effects on Disk Reliability

- Disks designed to sustain a power on-off cycle every 8 hours (Seagate)
- Each power on-off is estimated to cut the life of a disk by 10 hours (Google)
- These are not verified

Recap of Power & Reliability

- Summary of effects of power management on resilience:
  - Electrical and mechanical stresses at component and data center levels
  - Unknown effects on soft error rates
  - Potential complexities due to software running at different speeds
  - Repeated power cycles of disks can accelerate mean time to failure
- Plenty of open questions yet lack of modeling, understanding and experience with actual systems
- Hypothesis
  - Effect of power management on reliability is an open area for research

Future Trends: System Size

- Technology imperatives:
  - Processor speed improvement essentially over
  - Disk speeds will continue to improve but not at the rate of the rest of the components
- Performance demand will force systems to increase in size
  - 100,000's of processors will be in HPCS class machines
  - Improving system performance will result from more horizontal scaling, and thus future systems will have millions of processors, memory chips, and disks
- Effects on reliability:
  - More failures and MTBF will likely get worse

MTBF in Future Systems

- Increasing the number of components will reduce MTBF unless component-level reliability is improved at the same rate of *system-level* horizontal scaling (unlikely and unaffordable)
- Lower MTBF will require higher rate of checkpointing, but checkpointing latency will have to go up
  - Number of disks required to maintain checkpointing latency at *today's dismal levels* is going to increase at the same rate of system-level horizontal scale (likely unaffordable, therefore expect fewer disks)
  - Centralized disk storage for checkpoint support will require ever-more increasing network bandwidth (likely unaffordable)

Future Trends: Problem Size

- In the past, weak scaling (increasing problem size) was able to increase performance despite scalability constraints
- In the future
  - Some problems will continue to benefit from weak scaling
  - But an increasing number of problems will need to benefit from strong scaling (shorter time to answer with a given problem size)
- Effect on reliability:
  - Indirect but powerful: Strong scaling will emphasize the loss due to checkpoint/restart, rollbacks. Users will need systems that will provide at least 10X reduction in checkpointing overhead if not more

Future Trends: Soft Errors

- Moore's law is alive and well: We will continue shrinking the devices well into next decade, if not longer
- Soft error rate will increase
  - Regardless of power issues and even if you cool the system to sub 0 Celsius
  - Components for future systems will be less reliable, especially those targeted at cost-sensitive markets, which are typically leveraged in HPC clusters
- A similar problem exists in disks, where the increasing density will likely be accompanied by increased failures.
- Expect more overhead in time and power:
  - More frequent scrubbing, both in memory and disk!
  - More use of erasure codes in disks and memory

Future Trends: Heterogeneity

- Future (current?) systems will include heterogeneous hardware:
  - Main processors of different speeds
  - Main processors of different types
  - Vector accelerators
  - FPGA's
  - Graphic processors
  - Game processors
  - ASIC's
- Each of these components will introduce unknown reliability factors due to
  - Potentially different MTBF and unknown failure modes
  - Interactions with power management

Heterogeneity & Reliability

- Heterogeneity will be exposed to the software (OS, compilers, libraries, applications, etc.)
  - More potential for software failures due to complexity?
  - How to make such systems reliable? Some may not be running an operating system at all?
  - Checkpointing a heterogeneous system is not a well established art (doable, but careful planning of resources and checkpoint setting will not be trivial)
    - E.g. To include the coding for checkpointing the state of an FPGA is not trivial, and stopping the system so that a checkpoint can be taken will force artificial delays
    - Heterogeneous systems will fail differently and at different rates, so system software will have to detect and recover, but complexity will increase, leading to a reduction in robustness

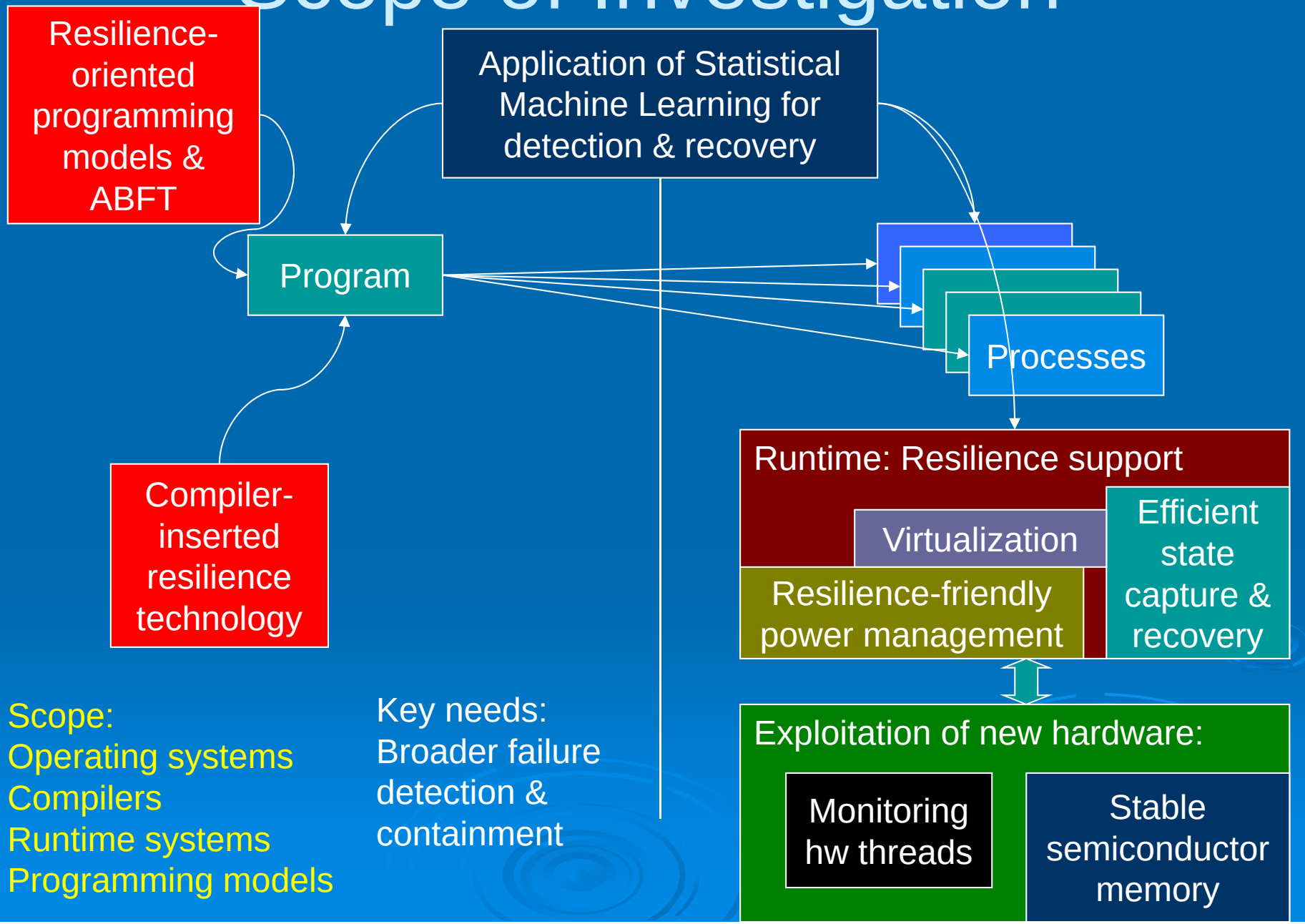
Heterogeneity in Software

- The software model based on MPI may not be applicable to heterogeneous systems without important extensions
  - Components may share memory within a system and across the network
- Other software models will include other forms of sharing (e.g. Partitioned Global Address Space systems)
- Restarting an application from a checkpoint may not be supported on some architectures
- Doing this at the application program level will reduce productivity and will be error prone
- New reliability technology will be needed
- Quantifying these effects is difficult at this stage

Effects of Heterogeneity

- Checkpointing when processors differ:
  - Solution #1: partition the processors into pools which may checkpoint/recover interchangably.
    - Limits flexibility and requires spares in each pool.
    - Wasteful of resources.
  - Solution #2: take portable checkpoints
    - Requires source code translation, strong type identification, and checkpoint translation.
    - Prototypes exist [Strumpen98,Marques05]
    - Unrealistic for large systems.
- What does this mean?
  - Automatic checkpointing of limited use in heterogeneous systems.
  - Program model should make program state explicit and reliable, so that the processor architecture plays less of a role in recovery.
  - Separation of process state and program state mitigates heterogeneity.

Scope of Investigation



Acknowledgements

- Ricardo Bianchini, Rutgers
- David Brooks, Harvard
- Tarek El-Ghazawi, George Washington University
- Armando Fox, University of California, Berkeley
- Forest Godfrey, Cray
- Adolfy Hoisie, Los Alamos National Laboratory
- Kathryn McKinley, University of Texas, Austin
- Rami Melhem, University of Pittsburgh
- James Plank, Univ. of Tennessee, Knoxville
- Partha Ranganathan, HP
- Vivek Sarkar, Rice University
- Joshua Simons, Sun