Adaptive, Large-Scale Computing Systems: Need vs. Want

Daniel A. Reed
Vice President for Research and Economic Development
University Computational Science and Bioinformatics Chair
Computer Science, Electrical Engineering & Computer Engineering, and Medicine

www.hpcdan.org  dan-reed@uiowa.edu
DOE SEAB HPC committee charge

The justification for an exascale computing capability initiative.

- DOE missions
- Fundamental research opportunities
- Broader societal benefits from an open, non-classified exascale program and potential market barriers inhibiting private development of exascale computing

Related basic research necessary to enable next generation high performance computing (e.g. mathematics, computer science, etc., including quantum and superconducting computing)

The current state of technology and plans for an exascale program in the Department of Energy and other federal agencies

Role of the Department of Energy in leading the development of exascale computing - including its involvement and collaboration with industry, universities and other government agencies on high performance computing

Implications of data centric computing for exascale computing

energy.gov/seab/secretary-energy-advisory-board-seab-task-force-next-generation-high-performance-computing
U.S. DOE SEAB HPC committee

SEAB Members
- Shirley Ann Jackson, Co-Chair, Rensselaer Polytechnic Institute
- Michael McQuade, Co-Chair, United Technologies Corporation
- Ram Shenoy, ConocoPhillips
- Steve Koonin, NYU Center for Urban Science and Progress

External Members
- Roscoe Giles, Boston University
- Jim Hendler, Rensselaer Polytechnic Institute
- Peter Highnam, IARPA
- Anita Jones, University of Virginia
- John Kelly, IBM
- Craig Mundie, Microsoft
- Thomas Ohki, Raytheon BBN Technologies
- Dan Reed, University of Iowa
- Kord Smith, Massachusetts Institute of Technology
- John Tracy, Boeing (Ted Colbert)
1. Investable needs exist for an exaX class machine
2. Significant, but projectable technology development can enable one last “current” generation machine (1-10 exascale level)
3. Classical high end simulation machines are already significantly impacted by many of the data volume and architecture issue
4. Data centric at the exascale is already important for DOE missions
   • Rapidly scaling to and beyond levels of performance that are comparable to hose needed for classic high performance floating point computation
5. Common challenges and under-girding technologies span compute needs
6. Factors driving DOE’s historical role in leadership computing still exist and will continue to do so
7. A broad and healthy ecosystem is critical to the development of exascale and beyond systems
8. It is timely to invest in science, technology and human investments for “Beyond Next”
   • Superconducting, quantum, biological/cognitive (neurosynaptic)

Note: emphasis mine
Culture and economics: getting what you need

Industry
• Capital is really, really cheap (look at interest rates)
• Labor is increasingly expensive
• ROI drives behavior

Academia and government
• Capital is very expensive
• Labor is still cheap
• Other metrics drive success

To change the game, change the metrics ...
• Infrastructure, personnel, social and political

Put another way, match what you want with what you need
Cloud computing lessons

Generic server design
- OpenCompute
- Workload-specific optimization
  - Functional accelerators
- ODM, not OEM partnerships

Energy optimization
- Substations and generation
- Switchgear control

Programming efficiency
- Rich toolkits and expression

Systemic resilience
- Failure management, not avoidance

Network optimization
- Flatter networks
- Software virtualization and flow

Supply chain optimization
- The advantage of scale
Netflix Simian Army

**Chaos Monkey**
- Random service termination to ensure other services continue operation

**Latency Monkey**
- Simulates service degradation and ensures services react

**Janitor Monkey**
- Searches for and turns off unused resources

**Conformity Monkey**
- Ensures virtual machines meet specified standards

**Doctor Monkey**
- Monitors the “health” of various virtual machines

**Security Monkey**
- Monitors and analyzes system security
Integration is challenging in both directions

**Application Level**
- Mahout, R and Applications

**Middleware & Management**
- Hive
- Pig
- Sqoop
- Flume
- Zookeeper (coordination)
- Map-Reduce
- Storm
- Hbase BigTable (key-value store)
- HDFS (Hadoop File System)

**System Software**
- VMs and Cloud Services (optional)
- Linux OS variant

**Cluster Hardware**
- Ethernet Switches
- Local Node Storage
- Commodity X86 Racks

**Data Analytics Software Stack**
- Applications and Community Codes
- FORTRAN, C, C++ and IDEs
- Domain-specific Libraries
- MPI-OpenMP CUDA/OpenCL
- NA Libs
- PFS (e.g., Lustre)
- Batch Scheduler
- System Monitoring

**Computational Science Software Stack**
- IB+ Enet Switches
- SAN+Local Storage
- x86 +GPUs or Accelerators

**The University of Iowa**
Diverging cultures and loss

Scientific application complexity is rising
- Multidisciplinary fusion
- Temporal and spatial adaptation
- Data assimilation and processing

... along with multiple optimization axes
- Massive parallelism with heterogeneous cores
- Resilience/reliability at large scale
- Energy optimization for utility

Cognitive complexity must decline ...
- ... else the number of parallel software developers will asymptotically approach zero

C, Fortran, C++, MPI, OpenMP
CUDA/OpenCL
Python, Ruby, PIG, FLUME, R
Cloud/Web Services

Technical and mainstream software development have diverged
Shifting ratios: shifting optimizations

Rising costs
• Personnel (no Moore’s law here)
• Energy costs (capital and operating)

Declining costs
• Hardware for given performance level
• $/FLOP continues to decline rapidly

Hardware reliability
• Partially determined by market
• Mass market rewards low cost, not high reliability
  • Think mobile devices

Follow the money ...
Exascale resilience, provenance and scale

System sizes are rising rapidly
- The law of large numbers applies
- Failures are frequent
- Component MTBF is not that high
- Disk, power supplies, fans, DRAM

System resilience dominates
- Components are less important

Fail in place performability
- Over-provision and accept failure
- Weak consistency
- Eventual reconciliation

Biologically inspired resilience models
- Systemic resilience with unreliable components

Assessing Fault Sensitivity in MPI Applications

Changyu Yu
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

David A. Reed
Renison Computing Institute
University of North Carolina
Chapel Hill, North Carolina 27514

Abstract

Today, clusters built from commodity (C) clusters (or large performance computers, with failures occurring thousands of processors now being deployed) is rare. Errors are usually handled by system recovery processes initiated to detect and correct errors. Typically, errors are identified through signatures or patterns that are stored in a database or a history file. Such signatures are often used to determine system failures and failures are often considered to be critical. Consequently, the reliability of the system is critical to maintaining confidence in the system.

Using software fault injection, we simulated the effect of a memory access error, which failed and MPI recovery failed when and corrected the behavior of a suite of MPI applications. The experiments showed that most applications are not consistent across single errors. However, most errors were on the processors, which were often not detected, leading to recovery of the system. For some, these minimal internal application errors did not affect the performance of the system. Some errors are not random and can affect the performance, which is the subject of the future.

1 Introduction

Today, clusters built from commodity (C) clusters (or large performance computers, with failures occurring thousands of processors now being deployed) is rare. Errors are usually handled by system recovery processes initiated to detect and correct errors. Typically, errors are identified through signatures or patterns that are stored in a database or a history file. Such signatures are often used to determine system failures and failures are often considered to be critical. Consequently, the reliability of the system is critical to maintaining confidence in the system.

In this paper, we consider the impact of soft errors on MPI applications by injecting faults into the system. The process is as follows. First, in [4], we evaluate the reliability of the fault injection framework for handling failures and simulate the number of faults and the corresponding system. In this paper, we follow the guidelines of the fault injection framework for handling failures and simulate the number of faults and the corresponding system. This paper is divided into three sections: performance, utility, and threshold. In [4], we evaluate the reliability of the fault injection framework for handling failures and simulate the number of faults and the corresponding system. This paper is divided into three sections: performance, utility, and threshold. In [4], we evaluate the reliability of the fault injection framework for handling failures and simulate the number of faults and the corresponding system. This paper is divided into three sections: performance, utility, and threshold.
Field replaceable module (FPM) optimization

IBM

Google

Microsoft
Over-provisioned hardware performability

Balance node RAS against system RAS
• NRE versus acquisition/repair
A simple, optimistic provisioning model
• N servers and S spares
• Servers fail independently at rate $\mu$
  • Spares may ($\lambda > 0$) or may not repaired ($\lambda = 0$)

$$p_k(t) = \binom{N+S}{k} (e^{-\mu t})^k (1-e^{-\mu t})^{N+S-k} (\lambda = 0)$$
Modeling approaches

Analytic models
- Simple, aggregate node models
- Only analytically tractable distributions
- State space explosion challenges
- Easy to explore large parameter spaces

Discrete event simulation
- Detailed, hierarchical models
- Generalized failure distributions
- More difficult to explore large parameter spaces
Scaling and near complete decomposability

Weakly interacting partitions (racks or FPMs)

- Implications
- Solution to state space explosion
  - State transition matrix $P$ is strongly diagonal
  - Partition models can be solved separately
  - Numerically tractable even at large scale

Failures (usually) have local hardware impact
Rack failures rarely affect other racks
Rethinking computing energy

Multiple energy sources
• Electrical grid, solar, wind, fuel cell, ...

Multiple cost functions
• Energy pricing, carbon taxes, varying availability
• Hardware, data transfer bandwidth/latency ...

Multivariate optimization and prediction
• Workload demand (diurnal and seasonal)
• Workload location subject to service level agreements (SLAs)
• Weather and seasonal models
• Auction-based energy pricing
• Infrastructure (UPS, optical fiber and computing)

Scheduling subject to energy and reliability
• Cost, availability, resilience ...
Energy constrained scheduling

Two options, generalizable to auction-based pricing

- Peak/off-peak pricing but no limit on energy availability
- Fixed pricing but peak/off-peak energy availability

Given

- System size $N$ nodes
- Total system energy budget $B$
- Sequence of jobs $J = j_1, j_2, \ldots, j_M$

Choose an optimal subset $S$ of $J$ such that

$$\sum_{j_i \in S} \text{Size}(j_i) \leq N \quad \sum_{j_i \in S} \text{Energy}(j_i) \leq B$$
User allocations are now of substantial value
- Depreciated capital cost over system lifetime
- Energy consumption for parallel job

Limited incentives for user responsibility
- Capital or operating cost efficiency

Exacerbated by
- Sub-linear parallel speedups
- Accelerators and node heterogeneity
- Multiple energy envelopes

New reward models must be explored …
Discussion