Panel: Reliability of Data Centers: Hardware vs. Software

Organizers:
- Mehdi Tahoori – KIT, Germany
- Ishwar Parulkar

Panelists:
- Dan Alexandrescu – iRoc, France
- Kevin Granlund – EMC, USA
- Allan Silburt – Cisco, USA
- Bapi Vinnakota – Intel, USA

In today’s life, data centers are integral part of daily life. From web search to online banking, online shopping to medical records, we rely on data centers for almost everything. Malfunctions in the operation of such data centers have become an inseparable part of our daily lives as well. Major malfunction causes include hardware and software failures, design errors, malicious attacks and incorrect human interactions. The consequences of such malfunctions are enormous: loss of human life, financial loss, fraud, wastage of time and energy, loss of productivity, and frustrations with computing. Therefore, reliability of these systems plays a critical role in all aspects of our day to day life.

These enterprise information systems (data centers) typically hold several hundred disks which can be protected via RAID protocols. The internal bussing architecture provides for a high degree of redundancy so that a failure in any component on a link does not disconnect other components from the system. In their front-end subsystem, they contain multiple processor cores. The operating system (OS) manages all functions of the information system and supports some basic configuration operations. Other software utilities might also be running on the host.

As device geometries continue to shrink, the hardware components in the information systems, such as microprocessor, memory and ASIC devices will experience an increasing number of transient and permanent failures. For instance single event upsets (SEUs), due to alpha particles from packaging materials and neutrons from cosmic radiations, are exponentially increasing due to Moore’s law and smaller device sizes. Increasing process variation coupled with reduced noise margins in aggressive designs in high performance computing result in increased intermittent failures during normal operation. Another source of errors is transistor aging due to Negative Bias Temperature Instability (NBTI). Errors are caused because of timing degradation due to a shift in $V_{th}$, the threshold voltage of transistors. Temperature gradients across chips have increased, which cause imbalances in power distribution, leading to errors. Increased use of high-speed SERDES interfaces and other analog circuitry has introduced failure modes susceptible to power supply noise, temperature and data content.

Software failures (bugs) in the operating system and/or transaction processing software can also adversely affect overall system reliability. The high complexity of software is the major contributing factor of Software Reliability problems. Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems. Software failure mechanisms are different than hardware failure mechanisms. Hardware faults are mostly physical faults, while software faults are design faults, which are harder to visualize, classify, detect, and correct. Therefore, the quality of software does not change once it is uploaded into the storage and start running. Trying to achieve higher reliability by simply duplicating the same software modules does not work, because design faults cannot be masked off by voting. Software reliability is not typically a function of operational time.

Recently, there is significant evidence of increased (hardware) failure rates due to technology scaling (more susceptibility to radiation, increased parametric variations, reduced noise margins, aging effects, etc). However, some data suggest that software bugs, in the utility and operating systems of enterprise information systems, might be the major detractor in the overall reliability of these systems.

This panel aims to address this debate and discuss possible approaches to improve overall reliability of such systems. Questions to be addressed and discussed in the panel include:

- “Admitting the occurrence of failures”: Do customers need to know about failures? Does the organization need to admit it? Does “talking failure” lead to “talking reliability”?
- Data corruption vs system downtime (unavailability): which is more “serious” or “worth solving”?
- How to improve the collaboration among hardware designers, software designers, and field engineers for overall reliability improvement?
- What are the challenges for automated hardware and software failure diagnostic and repair (user-transparent)?
- What architectural features in HW can improve SW reliability? Has there been enough investigation of this in the community? Is this worth doing?