

Design and Qualification of High Performance Computer Systems

**2007 Salishan Conference
High Speed Computing**

April 23-27, 2007

**Jeff Sonsalla
Engineering Director
SGI**

Presentation Overview

- The correctness stack
 - View from the top down
 - Middle ground – a data digression
 - View from the bottom up
- Trends and challenges
- That was then...This is now

Correctness Stack: View From Above

- A problem is defined
- A solution is expressed as a numerical algorithm
 - Equations are constructed in a manner that preserves precision
 - Equations are verified to produce stable results
- The algorithm is represented in software
 - Validate that the algorithm's intent is accurately modeled
 - Implement run-time sanity checks.
 - Enact integration and version management

Correctness Stack: View From Above

- Establish compiler accuracy
 - Optimizations qualified to preserve precision
 - Output examined to prove exactness
- Linked library environment defined
 - Enact integration and version management
 - Accommodate math library error rates in overall accuracy calculations

http://www.gnu.org/software/libc/manual/html_node/Errors-in-Math-Functions.html#Errors-in-Math-Functions

Correctness Stack: View From Above

- Operating system certified
 - Perform automated regression and scalability testing
 - Perform software fault injection testing
 - Perform installation/upgrade/configuration testing
 - Perform certification testing (distros)
 - Perform library testing (MPI/MPT)
 - Execute suite of applications for confidence, regression and performance baseline testing
 - Execute suite of functional, load and stress testing combinations
 - Validate on clusters and large shared memory SSI configs

Correctness Stack: View From Above

- Processor arithmetic capability quantified
 - Inherent standards identified and conformed
 - IEEE754 –
http://www.cs.berkeley.edu/%7Ewkahan/ARITH_17.pdf
- Control parameters delineated
 - Precision
 - Rounding
 - Exception handling

Correctness Stack: View From Above

- Interconnect safeguards understood
 - Links can be a source of large numbers of errors
 - Strong end-to-end error detection, with retry (soft errors corrected in-transit)
 - Data and address path protection (parity, ECC)
 - Backplane innovations for high signal reliability
 - Alpha immune latches to reduce soft errors.

Correctness Stack: View From Above

- **Memory resiliency prescribed**
 - **Memory errors most common source of system errors**
 - **Soft errors 30x persistent errors**
 - **Extended ECC schemes correcting whole devices**
 - **Memory scrubbing**
 - **Clean up CEs before they become UCEs**
 - **Identify weakening areas of memory**
 - **Memory mirroring and sparing**
 - **FBDIMM technology to reduce transient errors**
 - **Alpha immune latches to reduce soft errors**

Correctness Stack: View From Above

- Firmware arena stabilized
- Storage configured
 - Alternate path management (failover)
 - File system host back-up (failsafe)
 - Heartbeats, retries for transients, etc...

Correctness Stack: The Middle Ground

- System health monitored
 - Errors logged
 - Failed components replaced
 - Deteriorating components replaced upon exceeding set threshold
 - Replacement as transparent as possible
 - Confidence tests executed
 - Physical environment controlled
 - Internal state and consistency checks

Correctness Stack: A Data Digression

- **Data is managed**
 - **Origins identified**
 - Accuracy of data sources (sensors, archives, etc.) established
 - **Input transformed**
 - Accuracy of translations (binary to decimal, digital to analog, etc.) ensured
 - **Content manipulated**
 - Enact version management
 - Data integrity preserved across dynamic and static data types
 - **Information saved**
 - Database and longer-term storage (DMF)

Correctness Stack: View From Down Under

- A problem is defined
- A solution is expressed as a system architecture
- The system architecture is represented in hardware

Correctness Stack: View From Down Under

- The logic design is verified
 - Design verification (DV) – follow logic and circuit design rules
 - RTL simulation environment, controlled diagnostics: chip level – multi-chip – node – 2 nodes...
 - Cross-checking occurs as non-LD pour through specs to implement diagnostics, functional simulators, and system software

Correctness Stack: View From Down Under

- System software and diagnostics simulated
 - Functional simulator modeling hardware behavior for software development prior to HW power-on
 - Cross-checking mechanism as non-LD pour through specs to model ASIC behavior
- Components sourced from top-tier suppliers
 - ISO 9001 quality management system certification

Correctness Stack: View From Down Under

- Board designs optimized
 - High level of design quality enables insulation from Mfg tolerance variations, electrical noise and voltage and thermal margins
 - Low-level system control and management coming on-line tightly coupled for configuration controls

Correctness Stack: View From Down Under

- System booted
- System diagnostics executed to validate design at lowest level
- Signal integrity margin established
- Chassis environment certified
 - Agency (emissions) compliance achieved
 - Electrical compliance achieved (UL approved)

Correctness Stack: View From Down Under

- Design Verification Testing (DVT) commences
 - Early proto-type hardware failure modes, stemming from environmental stress factors, characterized
 - AC/DC voltage margins, AC or clock freq margins as applicable
 - High temp environment soak tests
 - Four corners (temp and humidity stresses)
 - Hot/cold power cycling on sampling of product configurations
 - Environmental Stress Screening (ESS)
 - Dynamic temp swings and vibration
 - Shock, Vibration, Package drop
 - Failures analyzed for root-cause and resolution, with results incorporated into design

Correctness Stack: View From Down Under

- Design manufacturability/repeatability demonstrated
 - Ensure mass produced hardware retains base level of quality (established during DVT phase) and adheres within the original design parameters
 - Hardware and software failure modes characterized through fault injection testing

Correctness Stack: View From Down Under

- Design Maturity Testing (DMT) performed
 - Overall product reliability is validated through extended run-time of a significant pool of release quality hardware in a simulated customer environment
 - Combination of diagnostics, confidence tests and customer applications may be used to exercise the system
 - Objective is to demonstrate a practical percentage of the predicted MTBF goal
 - Failures analyzed for root-cause and resolution and considered for the incorporation into product design

Correctness Stack: The Middle Ground

- System health monitored
 - Errors logged
 - Failed components replaced
 - Deteriorating components replaced upon exceeding set threshold
 - Replacement as transparent as possible
 - Confidence tests executed
 - Physical environment controlled
 - Internal state and consistency checks

Trends and Challenges

- **RAS+++**
 - Despite all rigor to the correctness stack – components will still fail.
 - System fault detection and containment initiatives
 - Memory, cache, register and interconnect data path checks
 - End-to-end ECC data checking
 - Self-checking/checkpointing applications (many cores/many days)
 - Hardware and software error injection capabilities to verify proper error detection and response/recovery
 - The dark side of economics (TTM, cost) effect on correctness stack rigor placing even more importance on comprehensive RAS

Trends and Challenges

- Partner relationships
 - To exploit full rigor of correctness stack, earliest practical engagement with vendor details is tantamount
- Schedule alignment
 - Significant infrastructure updates in software to accommodate upcoming hardware must be submitted prior to hardware power-on to align the software release schedule with the HW launch
 - Simulation reliance (chicken/egg on community acceptance)

Trends and Challenges

- Commodity component cut-in
 - **Validate that components meet appropriate specifications and perform as expected through Mfg screening**
 - **Identify system software to recognize new processor/DIMM type – attempt to boot and observe behavior**
 - **Validate via standard battery of tests: off-lines, on-lines, HW error handling regressions, HW/SW fault injections, OS, apps (perf engr analysis)**
 - **Specific new features/capabilities validated**
 - **DMT**

That Was Then...

- One stop shopping
- Short path to recovery
- Simpler set of concerted activities with full control of timeline
- Better availability and depth of information much earlier in the process
- Less overall exposure translated to greater chance of being incorrect

This Is Now...

- Shop 'til you drop
- Long lead times and extended path for fix availability
- Complicated set of moving pieces and conflicting tensions/priorities across sources
- Information much less detailed and available much later in the process, but components should meet appropriate specifications and perform as expected
- Greater overall exposure translates to greater chance of being correct

The End

- A unified priority on correctness is requisite to establish system integrity

sgi