Cross-Layer Reliability Metrics

Subteam Activities
Observations
Brainstorm with Whole Team
July 8, 2009

Agenda

- Subteam Members
- Subteam Timeline
- Problem Statement and Subteam Mission
- Why Cross-Layer Metrics?
- Vision for variable reliability at Runtime
- Call for better prediction (Kudva & Recchia et. al.)
- Hard error behaviors (Schroeder et. al.)

Reliability Metrics Subteam

- Name (alphabetical order): affiliation (subteam activities focus)

- Carter, Nick: Intel (errors of all sorts)
- Dekel, Eliezer: IBM Haifa Research (system software)
- Kudva, Prabhakar: IBM Watson Research (prognostics)
- Recchia, Charles: Intel (prognostics)
- Seager, Mark: LLNL (HPC, accurate predictions is scale-out)
- Mitra, Subhasish: Stanford University (co-leader, variable reliability of all sorts)
- Sanda, Pia: IBM Systems & Technology Group (co-leader, variable reliability for soft errors)
- Schroeder, Bianca: Univ. of Toronto (errors of all sorts)
- Xenidis, Jimi: IBM Austin Research (runtime software)

Subteam Timeline

- June 25 subteam meeting – introductions & brainstorm
- July 2 subteam meeting – problem definition & mission scoping – variable reliability whitepaper forming (draft available)
- July 6 focus group meeting on prognostics whitepaper forming (Kudva, Reccia, Mitra & Sanda)
- July 8 – today – Seek input / “cross-pollination” from broader CCC Team
- Post workshop subteam meeting
  - Incorporate broader team input into Whitepapers
- Whitepapers & Presentations completed and distributed

Problem & Subteam Mission

- Future systems will potentially combine many different components coming from many different suppliers
- This significantly complicates how we estimate / quantify / design for the overall reliability of a system.
- We need to describe the reliability of these systems, and the reliability (e.g. data integrity performance) must be ***predicted, verified***, and ***validated*** as a function of the ***workloads*** performed.
- We need metrics to be able to characterize and classify data integrity in this new paradigm of heterogeneous computing.
- Ultimately, we want the metrics to help enable the ***variable reliability*** to be delivered as needed at ***runtime***
- The context is not only the hardware, but the runtime software and applications.
- We seek a holistic view across hardware components, system architecture, operating systems and runtime software, and user applications.
- We will contain the scope of the study to hardware errors
  but look for metrics to quantify their effects across the various layers.

Layers

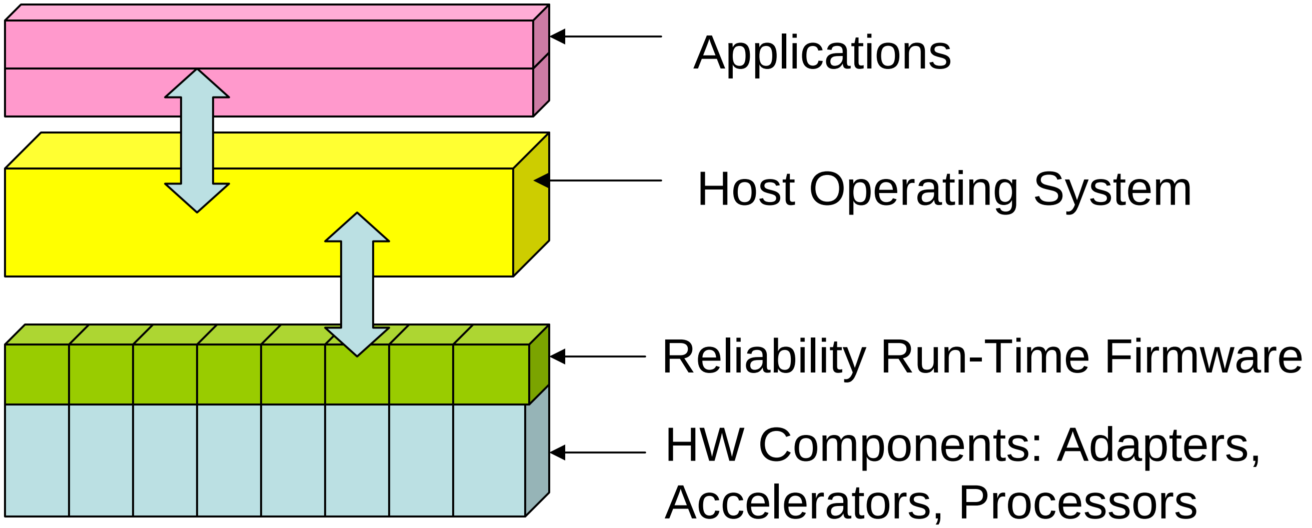
Error rates are not “just” the sum of hardware components

They depend on the RAS functions including firmware implementations

They depend on what the operating system does

They depend on the applications running

Reliability Metrics Need to Cross Layers



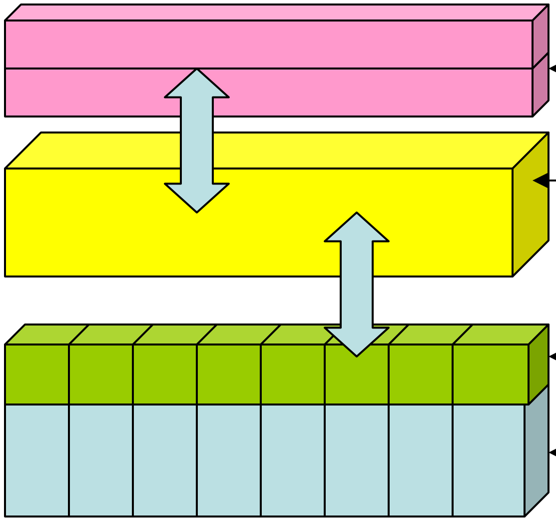
Variable Reliability

Vision: Future Heterogeneous Systems will have Standard Interfaces for Tunable Reliability

Interfaces will pass attributes to execute tunable reliability

Metrics provide the measures by which the ***System*** can quantitatively assess and control its reliability based upon its components..

Reliability Metrics Need to Be Passed Across Layers



Applications

Host Operating System

Reliability Run-Time Firmware

HW Components: Adapters, Accelerators, Processors

Variable Reliability

- Example 1:
  - Chip xyz is showing signs of wearout
  - It switches a wear indicator bit to “on” *(reliability metric)*
  - System middleware detects the xyz wear indicator bit has flipped to “on” and sends message to console “XYZ running in degraded mode” and field repair action is initiated *(reliability metric is passed between component to host)*

- Example 2:
  - Chip xyz1 is showing signs of wearout
  - It switches a wear indicator bit to “on” *(reliability metric)*
  - System firmware fails xyz1 out to spare xyz2 and sends message to console “XYZ failed and swapped to spare” and field repair action is initiated to replace spare *(reliability metric is passed between component to host)*

- Example 3:
  - Workload abc requires mainframe data integrity on accelerator efg but efg is a commodity part
  - Workload task carries a QoS bit *(reliability metric)* that indicates “mainframe reliability” and runtime software launches duplicate tasks on duplicate accelerators *(reliability metric is acted upon by runtime software)*
  - Results are crosschecked to be correct and result is sent to host

Metrics for Accurate Error Rate Prediction

Requirements for metrics

- Common language for system integration requirements
- Measure both current and prognostic reliability
- Common requirements for composition of large scale systems, and smaller systems
- Correlation between system components
- Capture both SER and HER

Error Rate Dependencies

- Error rates (both current and prognostic) for both components and system are affected by
  - Environment
  - Configuration
  - Utilization

Component Error Rate

- Define error rate as a range COMP_i ER = [min-max] for components/hierarchies over:
  - Configurations in which component is used within system
  - Environment in which a component may be used
  - How the workload uses the component (hit rate and line usage in memory for example)

System integration

- $EFF\_ER^{COMP\_i} = FUNCTION1\ (ER^{COMP\_i}, CORR_j^{COMP\_i}, UTIL^{COMP\_i})$
- Such a function will be computed hierarchically where each node in the hierarchy becomes a component at the next level
  - ER is rated error rate of component i under certain conditions
  - $CORR_j^{COMP\_i}$ is the correlation variable that captures the relationship between the error rate of component i and other components j in the system
    - for example, the data rate of an IO device connected to a bus may be limited by data rate of bus
    - DIMMs may be configured many different ways based on other components in system
  - $UTIL^{COMP\_i}$ is a variable that captures the dependency of the error rate on the workload
- ER can be defined as any one of SDC, Checkstops, performance loss etc.

Prognostic error metrics

- Predict future error rate (system fragility) based on knowledge of components
  - Example: if spare (processor/redundant line) etc are already used up, prognostic error rate is high
- Predictability of such an error rate and/or sensitivity of a component
  - Will help pre-empt failure
  - Identify critical components on the verge of failure AND whose failure would cause system wide outages and/or SDCs
  - Focus service requests requirements

A typical prognostic equation

- Prognostic System Error rate =
  - $\sum$ FUNCTION2 (EFF_ER$^{\mathrm{COMP\_i}}$, CRIT$^{\mathrm{COMP\_i}}$, UTIL$^{\mathrm{COMP\_i}}$, FAIL/ENV_STATUS$^{\mathrm{COMP\_i}}$);

- Variable FAIL/ENV_STATUS$^{\mathrm{COMP\_i}}$ is used to capture the current state of fragility of component or hierarchy

Work to do

- In the context of some full systems, and diverse application domains (HPC to consumer), define:
  - 1. Define $\sum$j CORRj $^{COMP\_i}$, for each component/hierarchy i, sum the correlation of error rate between component i and all other j components in the system precisely
  - 2. Define CRIT $^{COMP\_i}$, i.e., criticality of component i in the system precisely
  - 3. FAIL/ENV_STATUS $^{COMP\_i}$, potential of component error rate to increase (either SER or HER based on current failure/environmental conditions precisely.
  - 4. Precise definition for UTIL$^{COMP\_i}$. This may be tricky based on component type (processor, memory, IO, disk etc).
- Identify case study systems and evaluate these and other required metrics

Reliability Metrics Study Group: Hard Error Metrics

Bianca Schroeder

Computer Science Department
University of Toronto

Slides based on discussions in phone con-call arranged by Pia and e-mail exchange with LANL folks.

Hard errors

- What is a hard error?
  - A repeatable error, due to permanent hardware problem
- Why important?
  - Growing component count => more errors in future systems
  - Significant frequency: E.g. in DRAM an estimated 60% of uncorrectable errors due to hard errors.

- <u>Our question:</u>
  - What are the right metric(s) for hard errors?

Why do we need metrics?

- Good metrics should be quantities we can measure & that aid in:
  - Management of current systems
    - Predict interrupt frequency apps see
    - Predict component failures
  - Planning of future systems
    - Predict interrupt frequency of future systems
    - Determine requirements for components in future systems

The standard metric: FIT

- Frequency per time
  - FIT = failures in time per billion hours
  - Or at device level: FIT / Mbit

- Is FIT good enough?

Is FIT good enough?

- No, not all errors are created equal!
- Take into account <u>impact:</u>
  1. Detected & corrected
  2. Detected & uncorrectable => failure
  3. Undetected => silent data corruption / crash

Can we just focus on 2. and 3.?

- No, because:
  - Measure of ``fragility'' of system
  - Can be predictor of permanent component failure
  - 1. is often easier to measure than 2. and 3.

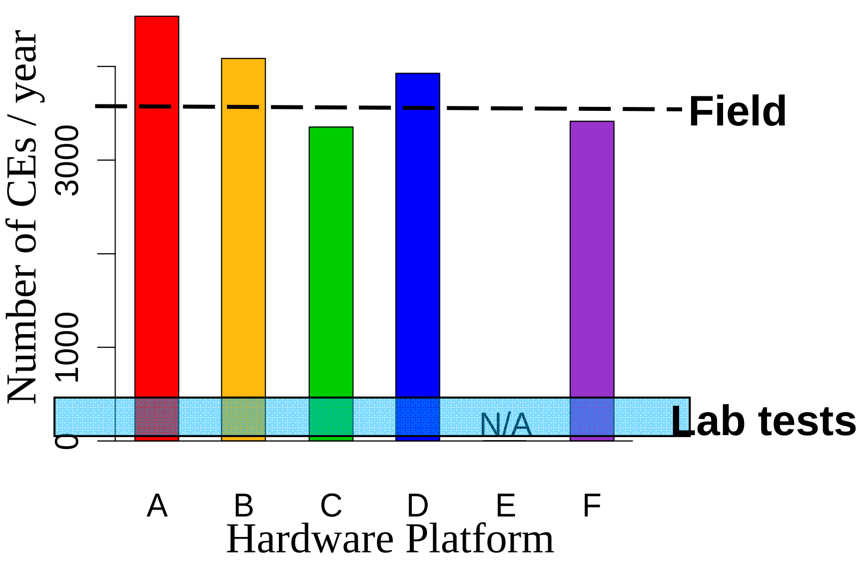
Is FIT good enough?

- No, error frequency depends on many factors:
  - Operating conditions (temperature)
  - Utilization / workload
  - Age
  - System configuration / interaction between components
  - …. any many others
- So, which do we take into account?
  - All possible factors => not practical
  - Only the relevant ones => what are those?
- Don’t know, errors in the field not well understood …

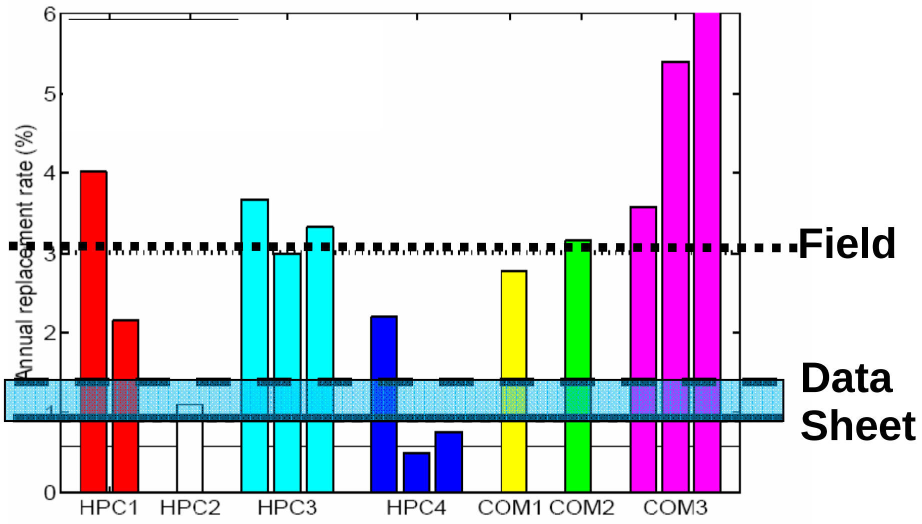
Frequency of errors in today’s systems

- Example 1: [sigmetrics’09]
  DRAM errors in the field

Correctable errors (CEs)



- Example 2: [FAST’06,TOS’07]
  HDD replacements in the field



- Accelerated lab tests and vendor data sheets are not enough
- Need real field data!

Effect of age?

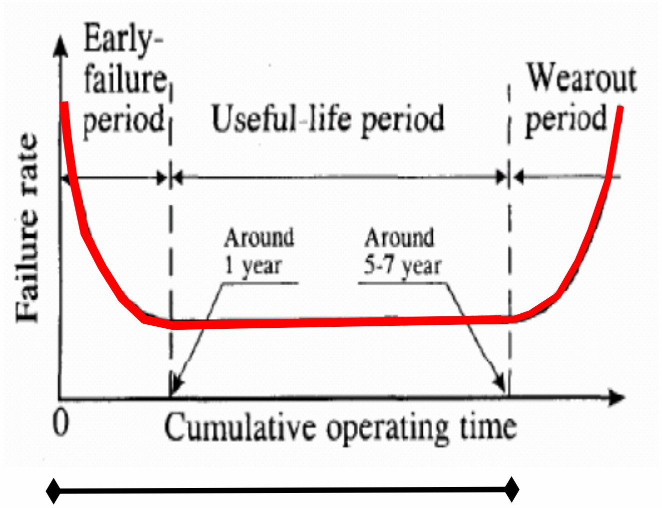
- Theory:

  Little effect during nominal lifetime

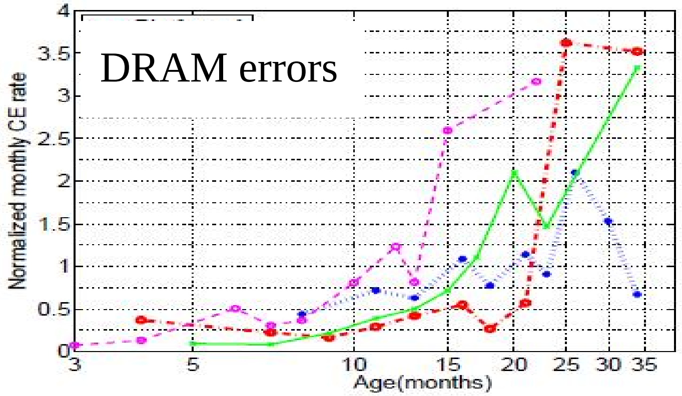
- Practice: [FAST'06,sigmetrics'09]

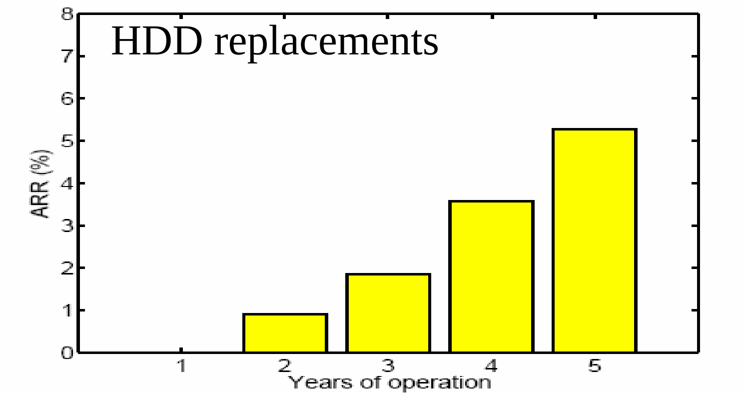
  Surprisingly early wear-out

  Infant mortality no concern



Nominal lifetime – 5 years

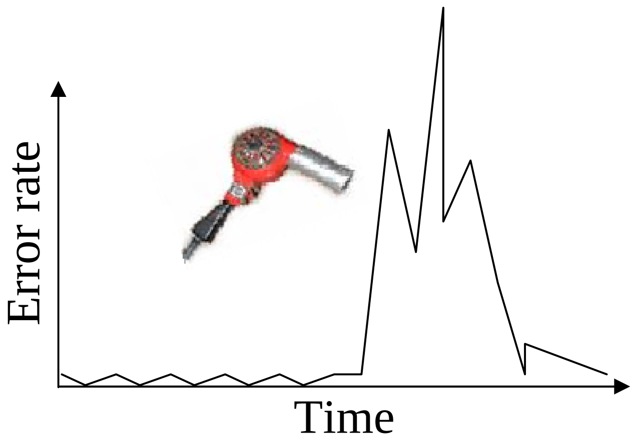




Effect of temperature?

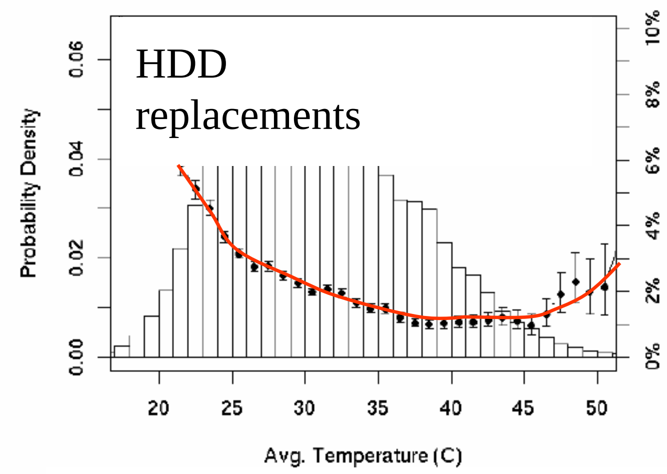
- <u>Theory:</u>

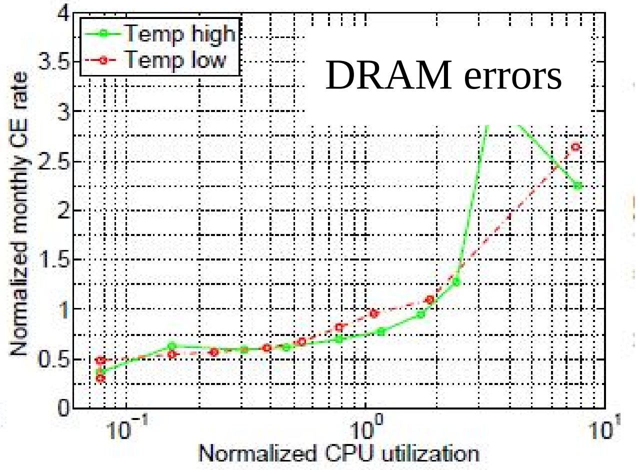
  Effect known from lab experiments



- <u>Practice:</u> [FAST'06,sigmetrics'09]

  Unclear effect in the field





Conclusion

- FIT alone is not enough
  - Need to distinguish different error modes / impact of error.
  - Take into account factors that impact FIT
- But what factors to include?
  - Could include <u>ALL</u> possible factors
    - Impractical
  - Could include only <u>relevant</u> factors
    - But what are those?
- Many open problems
  - Keep in mind what goals we have for metrics.
  - Need field data to guide the process.