THE UNIVERSITY OF TEXAS AT AUSTIN

Toward Exascale Resilience

Part 2:

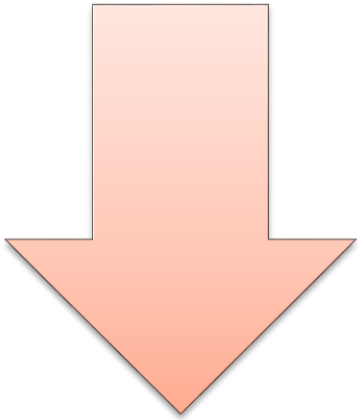
System Architecture and Components Resilience fundamentals

Mattan Erez

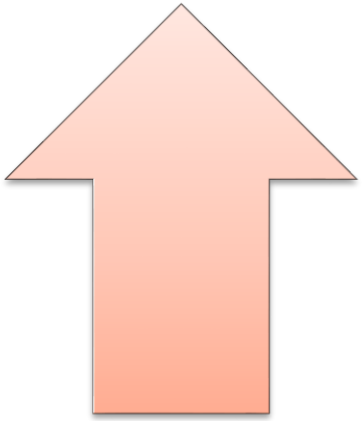
The University of Texas at Austin

July 2015

Algorithm
Program
Runtime
Operating system
Compiler
HW Architecture
HW Devices



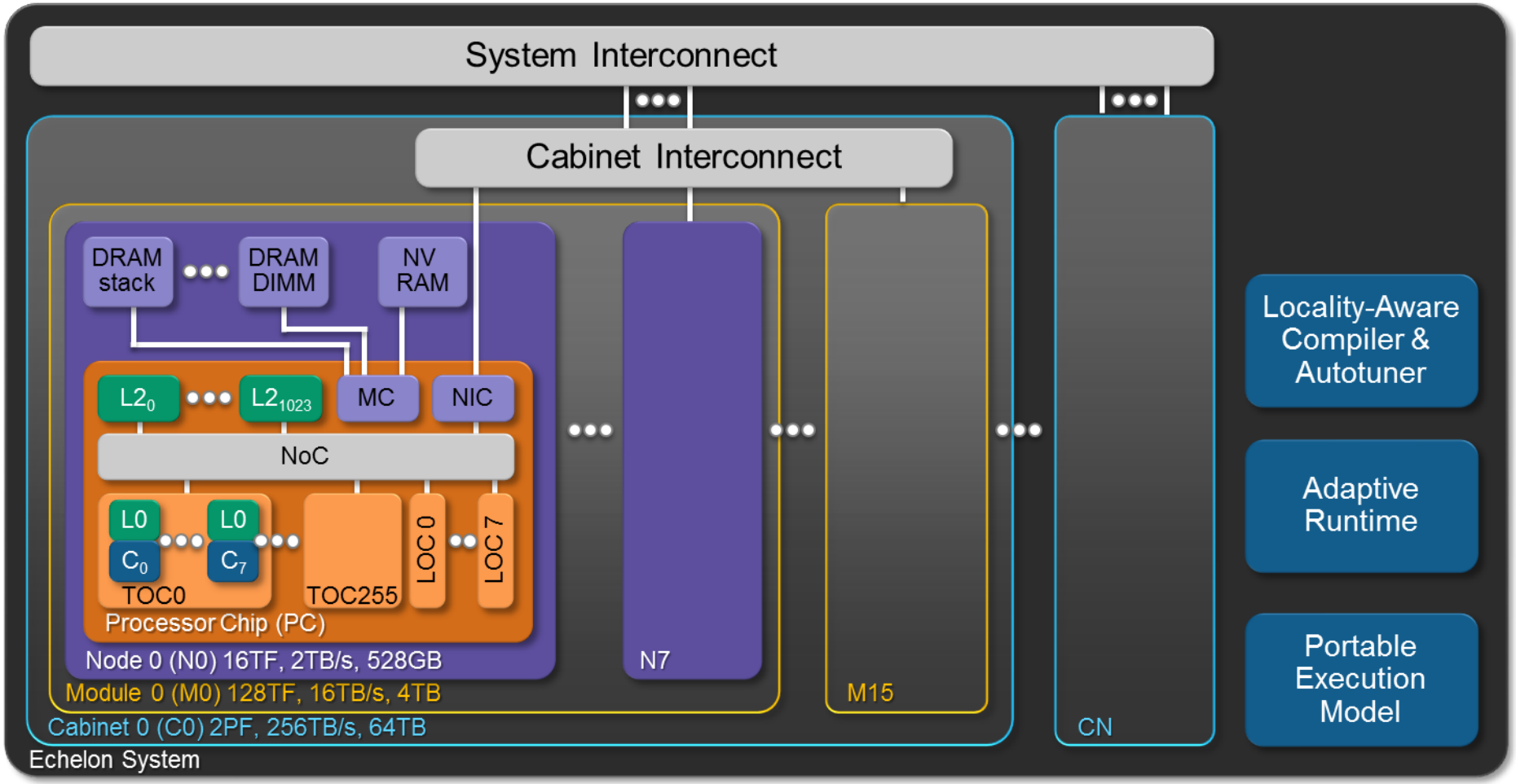
Resilience

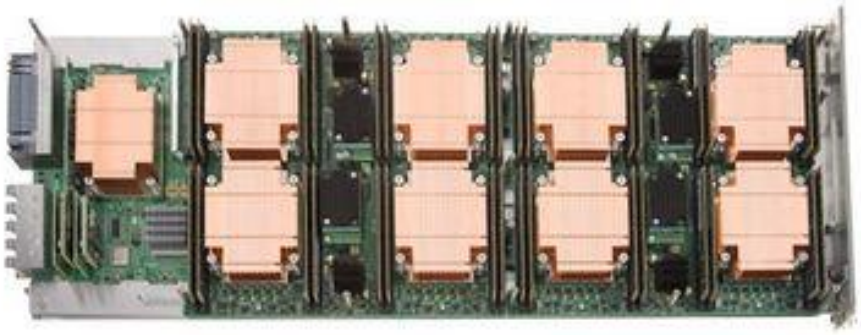


Reliability

A concentrated exaflop

– Cohesive system for cohesive problems





From a Cray XC30

Hardware summary

- Cabinets (w/ VRU)
- Shared power, cooling, and links (w/ VRU)
- Modules (w/ VRU)
- Sockets
- Processors + memories
- Gates, wires, and storage devices
- Storage separated out (for now)

System software manages resources

- Node operating system
- Distributed file system and I/O nodes
- System management and supervision

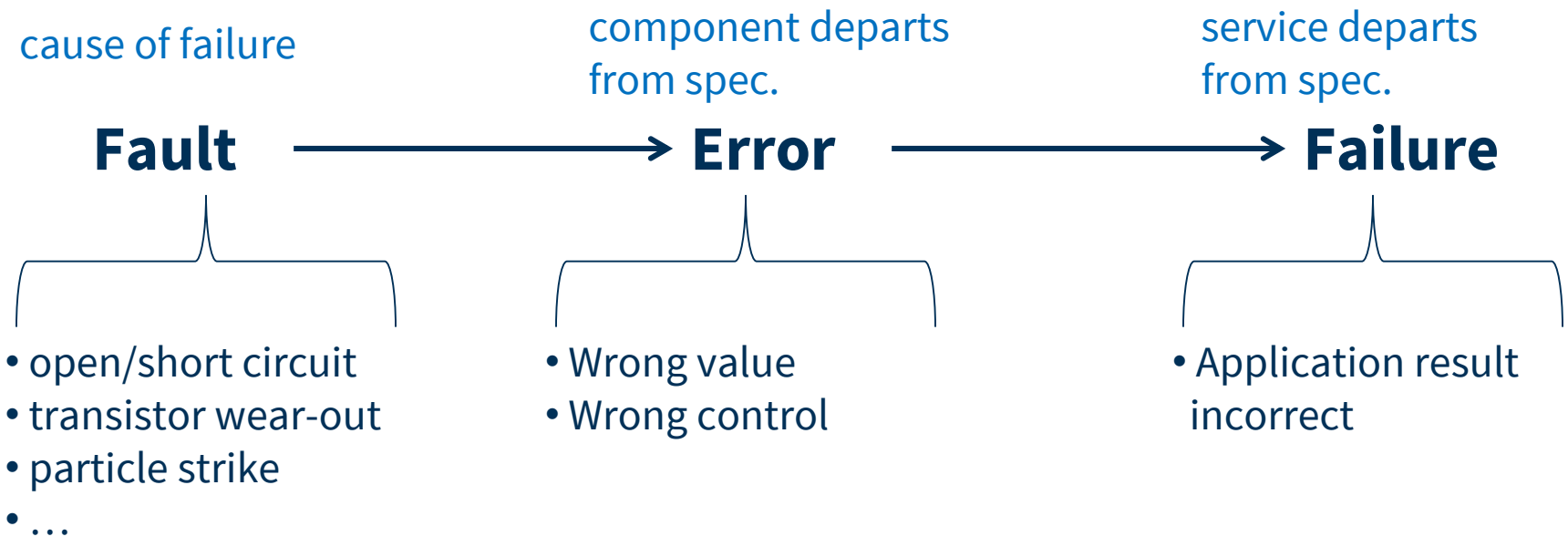
Terminology

Principles

Basic actions

RESILIENCE FUNDAMENTALS

Faults, errors, and failures



Fault taxonomy (others exist)

- What?
  - Component malfunction
  - May lead to errors
- Where?
  - What component? Hardware or software?
- How?
  - Natural / accidental / intentional?
- When?
  - Permanent / transient
  - Active / dormant
  - Reproducible / random
  - 0-day?

When do we care about faults?

Error taxonomy

- What?
  - State that differs from fault free state
- Where?
  - Values or control state
- How?
  - Data dependent / independent
- When?
  - Permanent / transient / intermittent
  - Reproducible / random
- Impact?
  - Masked / detected / silent

Failure taxonomy

- What?
  - Actionable outputs not within spec
  - Performance / power not within spec
    degraded operation
- Where?
  - Can be hierarchical to a subsystem
- When?
  - Signaled / silent
  - Failstop / failthrough
  - Consistent / inconsistent

Resilience = operating without failure in the presence of faults and errors

- Fault tolerance
- Fault mitigation
- Fault avoidance
- Error tolerance
- Error mitigation

(Forecast)

Detect

Contain

Recover

Repair

Fault/failure forecast

- Use past experience to forecast future problems
- Avoid faults and failures
  - Reconfiguration
  - Preventative-repair
- Traditionally done offline
  - Stress diagnostics
- Online work looks promising
  - Mostly machine learning approaches
- Requires effective monitoring

Detection

- Diagnostics for faults
- **Redundancy** for errors
  - Redundancy may be implicit

- Detected errors may identify faults
  - Root-cause analysis
- Requires effective reporting

Containment

- Error propagation increases costs
  - More to mitigate
  - More likely to lead to a failure
- Containment enables resilience
- Careful design
- Timely reporting

Recover (error mitigation)

- Forward recovery (masking)
  - Requires redundancy
  - Some errors ignorable
- Backward recovery (re-execution)
  - Requires preserved state

Repair (fault mitigation)

- Reconfigure to degraded operation
- Replace with hot spare
- Replace with cold spare
- Fix
  - Usually for software

Metrics

- Why measure
- Who is measuring

If you cannot measure it, you cannot improve it

Lord Kelvin

FIT – the unit of resilience and reliability

Generic “faults in time”

1 FIT → 1 fault in 10^9 hours of operation

- Really annoying unit
- But, rates accumulate

What do users care about?

What do users care about?

- Time to completion
- Cost of computation
- Correctness of answer
- Are they part of the solution

Relative efficiency

- Performance, power, time, energy, … with resilience relative to assuming perfectly-reliable system
- How much are overhead is a user paying for a particular resilience scheme
- Setting baseline in reality is tricky
  - Differences between components

Relative correctness

- Bit-by-bit perfect
- Same printout
- Matches expectations
  - Need to quantify expectations

Relative effort

- How much is required from the user?
- Users are accustomed to reliable systems
- Users always resist change
- How often are they interrupted

What about systems people?

What about systems people?

- Underlying system failure characteristics
- Availability
  - Overall / "Scheduled" $\frac{100*Uptime}{TotalTime}$ $\frac{100*Uptime}{Uptime+UDown}$
- *MTTI (mean time to interrupt) $\frac{Uptime}{NumInterrupts}$
- *MTBI (mean time between interrupts) $\frac{TotalTime}{NumInterrupts}$
- *MTTR (mean time to repair) $\frac{UDown}{NumInterrupts}$
- * =
  - System, application, job, node, …

Unscheduled donwtime