Outline

- Software Faults, Errors, and Failures
- Static Methods for Finding/Avoiding Software Faults
- Dynamic Software Error Detection
- Fault/Failure Isolation
- Interactions Between SW and HW Fault Tolerance

Fault/Failure Isolation

- As with hardware, we’d like to be able to isolate a fault/failure and prevent it from affecting the rest of the system
- Some of the tricks we used for hardware fault isolation involved using software
  - Logical partitioning
  - Virtual machines
- Can these schemes tolerate software bugs?
- Other options for software fault/failure isolation
  - Clustering (Google, Amazon.com)
  - Modular OS (Hive)

Fault/Failure Isolation with Clustering

- “Web Search for a Planet: The Google Cluster Architecture” (Barroso et al.)
- Note: this paper is obviously about much more than just fault/failure isolation!

Fault/Failure Isolation by the OS

- “Hive: Fault Containment for Shared-memory Multiprocessors” (Chapin et al.)
- We will NOT cover this paper in this class, but you may want to read it if you’re interested in OS issues
  - ECE 254 doesn’t have OS as a prerequisite
Outline

- Software Faults, Errors, and Failures
- Static Methods for Finding/Avoiding Software Faults
- Dynamic Software Error Detection
- Fault/Failure Isolation
- Interactions Between SW and HW Fault Tolerance

Hardware for Software FT

- Difficult for HW to know that SW is in error, because HW doesn’t know what SW is trying to do
- Some sanity checks possible
  - E.g., it’s unlikely that a program really wants to divide by zero
  - Any others?
- Recent work at Duke explored hardware support for detecting livelock and starvation
  - Use hardware to detect if a thread is spinning (i.e., not making forward progress)
  - Detects spinning by comparing state of the system each time thread reaches backward branch
  - If all threads are spinning at same time, this may be livelock
  - If one or more threads are spinning for a very long time, this might be mutual livelock or starvation

More Hardware for Software FT

- “A ‘Flight Data Recorder’ for Enabling Full-System Multiprocessor Deterministic Replay” (Xu, Bodik, and Hill, ISCA 2003)
- A “flight data recorder” (FDR) helps software debuggers by allowing them to replay the last few moments before the crash
- Keys: replay is deterministic!
  - This is non-trivial for a multiprocessor system
  - FDR must keep track of certain event orders

Software for Hardware FT

- Many examples of using software to tolerate HW faults (e.g., SWIFT, Google)
- In fact, all schemes for tolerating software errors will detect hardware errors that manifest themselves in the same way (i.e., they have the same error model)
  - E.g., self-checking software will detect a hardware fault if it leads to an incorrect result
More Software for Hardware FT

- Why not just use software for FT?
- Why bother with hardware FT?
- Doesn’t the “End-to-End” argument suggest that hardware FT is too low-level?

Outline

- Software Faults, Errors, and Failures
- Static Methods for Finding/Avoiding Software Faults
- Dynamic Software Error Detection
- Fault/Failure Isolation
- Interactions Between SW and HW Fault Tolerance