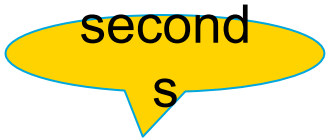
I’m going down!

(How to recover from failure in milliseconds)

Charlie Johnson

Pervasively Responsive Memory-Centric Data Centers

How responsive should middleware systems be in order to function properly in the face of failures (hardware, kernel panic, process death, etc.)?



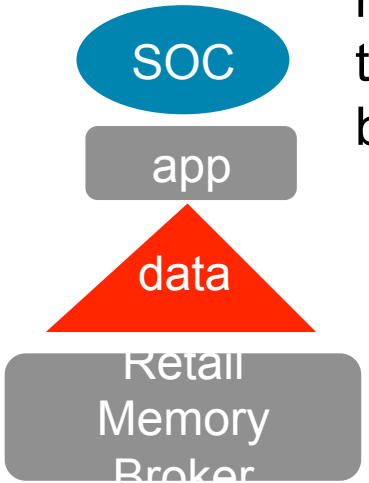
Batch systems < online systems < operating system services

Hadoop

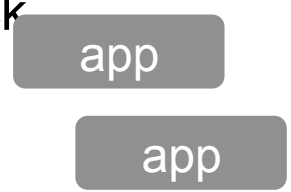
Web-service

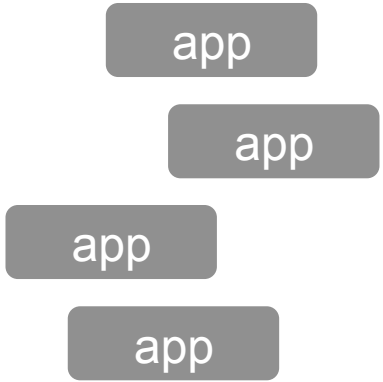
E.g., flash-trading systems, Retail memory broker system: High-throughput, low latency networks

While the Retail Memory Broker is unavailable, apps block.



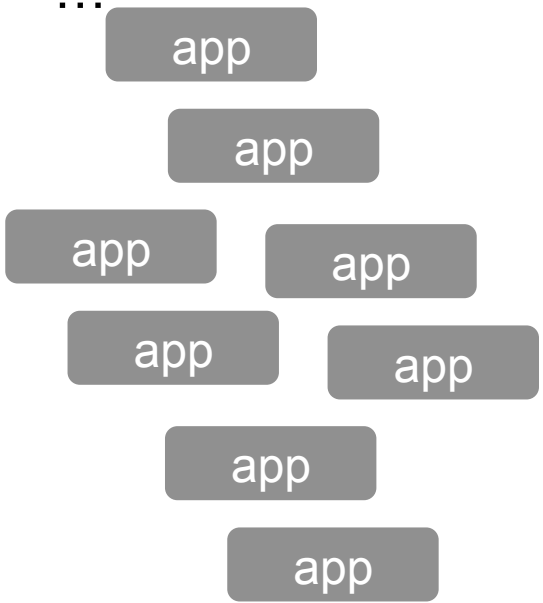
Apps waiting on resources held by the blocked app block





Apps waiting on resources held by the blocked apps block

And it can escalate …

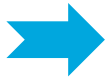


Availability depends upon Time to Recover

Mean Time Between Failures (MTBF) is important.
Mean Time to Recover (MTR) is much more important.

Availability = A = MTBF / (MTBF + MTR)

lim (Availability) ≈ 1
MTR->0



As the Mean Time to Recover decreases, Availability approaches 100%

And then …

Probability of Failure = F = 1-A ≈ 0

Reliability = 1/F = 1/(1-A) ≈ ∞

Existing systems spend a lot of time detecting failure…

e.g. Nonstop heartbeats, Paxos failure handling

- Election time outs (depend on heartbeats)
- Leader election voting time (or preselected leader notification)
- Odd cases: missing or failing heir, partitioned disputed succession, etc.

… but modern systems can tell us when they are failing by multicasting a death message (e.g. OpenHPI <-> OpenSAF)

How to Recover from Failure in Milliseconds

- Instead of heartbeats, **reduce time to detect failure** by leveraging baked-in (OpenHPI) failure detection and notification in the kernel (faster takeovers)
- Instead of leader election and huge clusters, **keep clusters small** and **fail over** to designated alternates and **invoke cluster recovery gracefully**.
- **Simplicity** avoids odd cases that increase recovery time.

Utilize failure detection existing in OS/firmware

Simplify fail-over: keep clusters small, and leverage application

KEEP IT SIMPLE!