A Case for Epidemic Fault Detection and Group Membership in HPC Storage Systems

Shane Snyder, **Philip Carns**, Jonathan Jenkins, Kevin Harms, Robert Ross
Argonne National Laboratory

Misbah Mubarak, Christopher Carothers
Rensselaer Polytechnic Institute

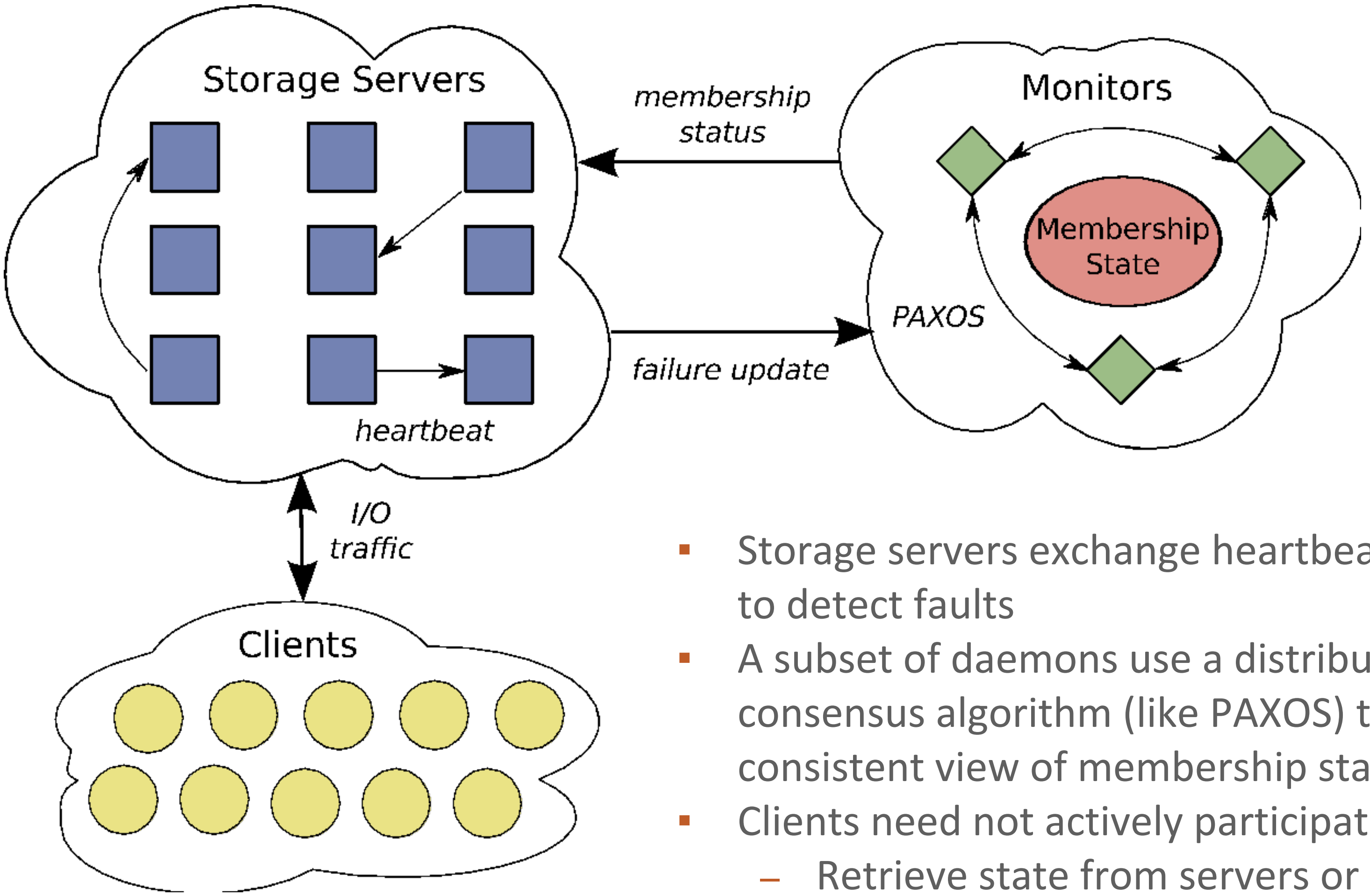
Background and motivation

- Fault detection and group membership are a critical to fault tolerance in **large-scale storage systems**:
  - Server joins group → migrate data to new server to improve load balance
  - Server leaves group → re-replicate data to maintain redundancy

- Why is it so important to get this right?
  - Inefficient (i.e., slow) fault detection may result in data loss
    - Slow recovery increases the window of vulnerability to coincident failures
  - Inaccurate fault detection interferes with performance and availability
    - False positives can trigger (unnecessary) costly rebuilds of the storage system and job failures

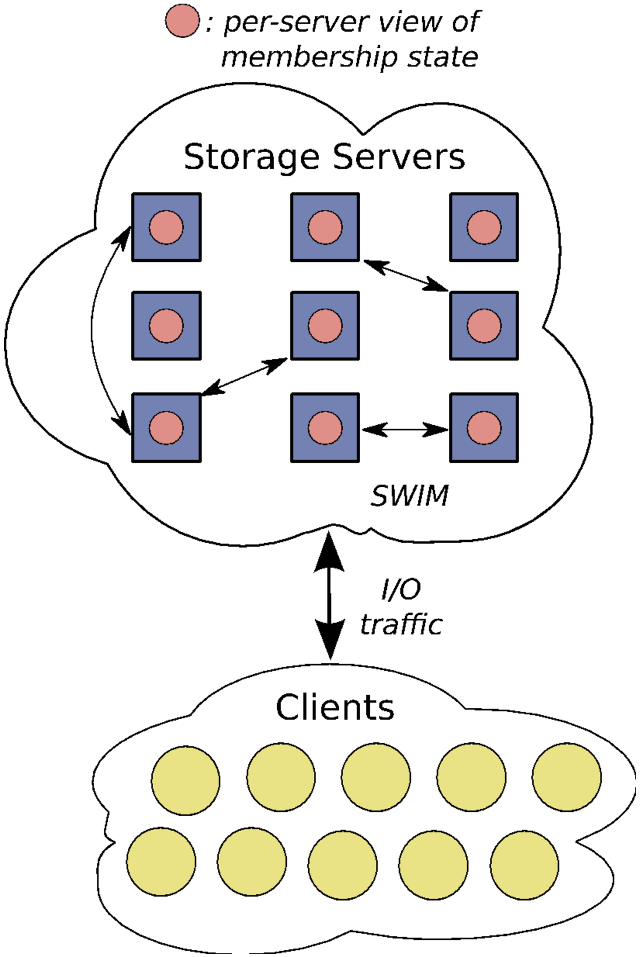
- Approach: use discrete event simulation to evaluate candidate algorithms at scale
  - What algorithms are viable?
  - Identify parameters needed for HPC storage systems
  - Explore long-running behavior not captured by analytical models

Background: conventional group membership



- Storage servers exchange heartbeat messages to detect faults
- A subset of daemons use a distributed consensus algorithm (like PAXOS) to maintain a consistent view of membership state
- Clients need not actively participate
  - Retrieve state from servers or monitors when needed
  - Limit the scaling requirements

Alternative: group membership with SWIM



- Similarities:
  - Clients need not actively participate
  - Servers exchange heartbeat messages to detect faults
- Differences:
  - No dedicated service for distributed consensus
  - Each storage server maintains its own view of the system
  - Disseminate updates using epidemic principles

SWIM does not provide strongly consistent ordering of group updates, but it does guarantee convergence and time-bounded completeness.

These semantic differences may require some accommodations from the storage service.

SWIM protocol background

Scalable **W**eakly-consistent **I**nfection-style Process Group **M**embership Protocol [1]

- Scalability
  - Probe-based (ping/ack) failure detection
    - The failure of a probe triggers ***indirect ping requests*** from other peers
    - A node is ***suspected*** to be failed if both direct and indirect pings fail
  - Infection-style (a.k.a. epidemic-style or gossip-style) dissemination
    - Membership updates are piggybacked on ping/ack messages
    - A ***suspected*** node is ***confirmed*** as failed after a suspicion timeout with no live messages

- Other properties:
  - Expected network load & time to detect a failed node is independent of group size
  - Epidemic dissemination and random pinging is robust against message loss
  - Parameters can be tuned to adjust sensitivity, network utilization, dissemination capacity, etc.

[1] Das, A., Gupta, I., Motivala, A.: Swim: Scalable weakly-consistent infection-style process group membership protocol. In: Proceedings of the 2002 International Conference on Dependable Systems and Networks. pp. 303–312. DSN '02, IEEE Computer Society Press, Washington, DC, USA (2002)

Simulation methodology

- We developed a high-resolution model of the SWIM protocol using the CODES framework [2]
  - Leverages ROSS, a high-throughput, optimistic PDES
  - Individual network message costs are calculated using the LogGP network model
  - Full-duplex network message queueing at each node
- Simulation strategy:
  - Use existing analytical models from the literature to choose initial parameters
  - Cross-validate analytical and simulation predictions
  - Use simulation to evaluate behavior that can’t be predicted using analytical models
  - Assess if the SWIM protocol is viable for further comparative studies

[2] Cope, J., Liu, N., Lang, S., Carns, P., Carothers, C., Ross, R.: Codes: Enabling co-design of multilayer exascale storage architectures. In: Proceedings of the Workshop on Emerging Supercomputing Technologies (2011)

Target: adapting SWIM for HPC

- O(thousands) of file servers
  - Protocol does not execute on compute nodes
- Low latency network and RTT
  - Enables short protocol periods (if desired)
- Tolerate transient errors < 15 seconds
  - Long enough to absorb NIC firmware restarts, busy servers, etc.
- Take action (confirm failure) within 30 seconds
  - Based on expectations from HA deployments in the field
- Keep network load “low”
  - What is an acceptable threshold here?

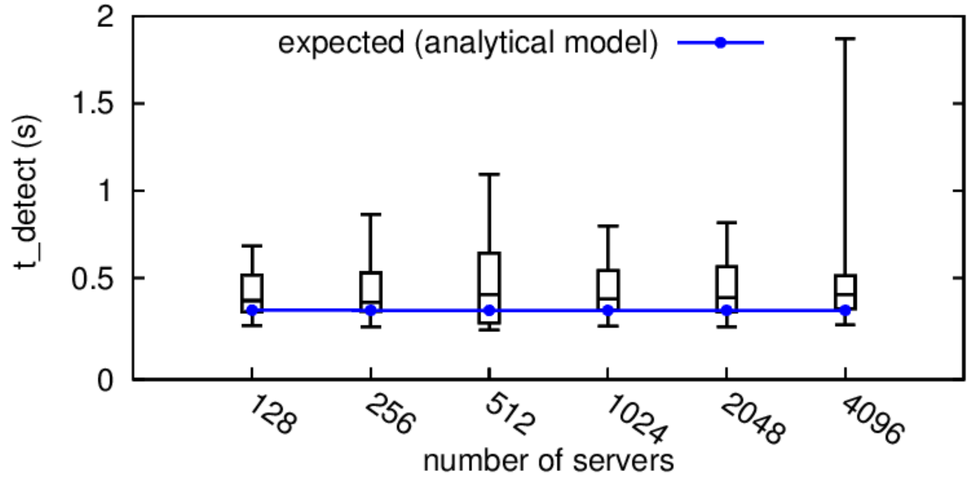
Different targets could be chosen for different use cases.

Initial parameters

Starting points chosen based on existing analytical models.

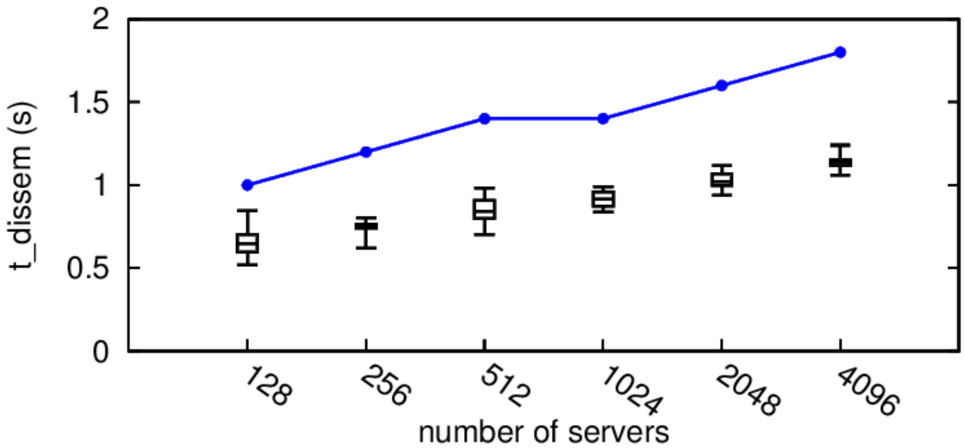
- Protocol period length: 200 ms
  - Time between randomized probes
- Suspicion timeout: 15 seconds (75 protocol periods)
  - Time before a suspected node is confirmed
- Packet size: 256 bytes
  - Allows up to 12 updates to be piggybacked per probe message
- Subgroup size (k) is critical as well; more on that later
- Expect 10s of messages per server per second

Validation with analytical model: detection



- **t_detect**: elapsed time between a failure and the first suspicion by a single peer
  - Expected to be constant with scale
- Simulation results:
  - 15 samples per box plot
  - randomized failure time and failed node
- Variability
  - Initial detection time as slow as ~2 seconds in the worst case
  - Due to random ordering of probes, not congestion

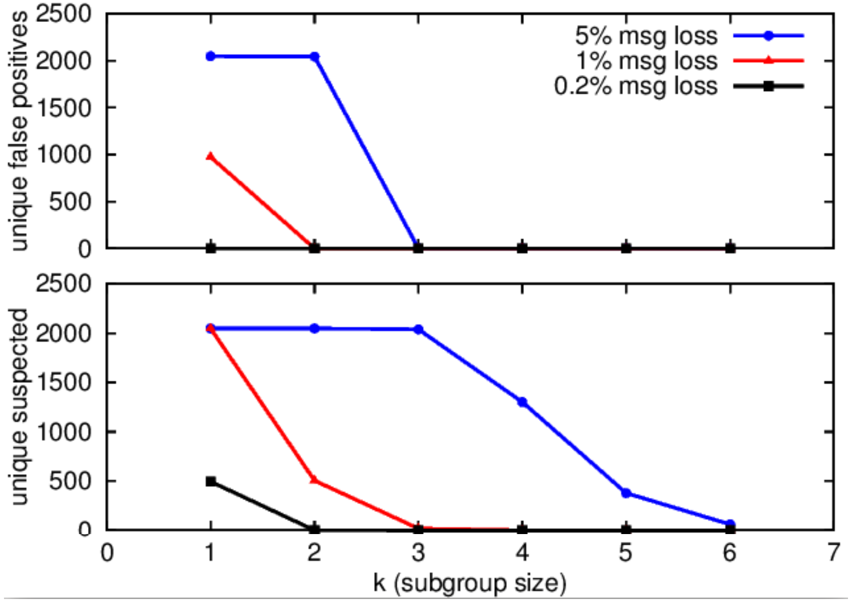
Validation with analytical model: dissemination



- **t_dissem**: time needed to propagate a state update to all servers
  - Expected to be logarithmic
- Simulation results
  - 15 samples per box plot
  - randomized failure time and failed node
- Simulation consistently faster than analytical prediction
  - round robin probing insures maximum dispersal
  - de-synchronized probe intervals reduces per-round latency
- (detection + dissemination) < 4s, but additional 15s suspicion timeout is used to avoid false positives

Tolerating packet loss

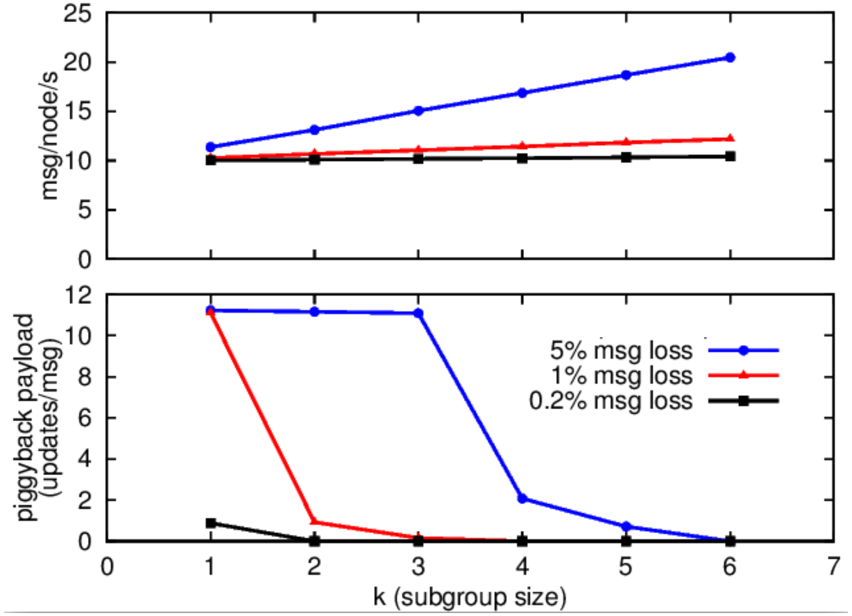
- Subgroup size (k): the number of peers to use for indirect pings
- Figure shows 30 minutes of simulated time for 2048 servers with ***no*** true failures, just lost packets
- Vary k from 1 to 6



- Increasing k:
  - reduces number of false suspicions by requiring more confirmation from indirect pings
  - reduces number of false positives (i.e. false confirmations) by increasing epidemic capacity and opportunities to revoke suspicion
- What is the downside?

Tolerating packet loss

- Figure shows utilization metrics from the same 30-minute simulations
- **msg/node/s**: average number of messages transmitted by each server per second
- **updates/msg**: average number of piggyback slots used per message



- Per server load increases linearly with k in lossy network environment
  - Total load is still modest
- Piggyback slot usage indicates if the epidemic dissemination protocol is saturated or not
- *K=6 imposes minimal overhead to insure robust message loss tolerance*

Conclusions

- The SWIM protocol is a promising approach to group membership in large-scale HPC storage systems
  - Robust against transient failures
  - Rapid detection and dissemination
  - Low network overhead
- We successfully modeled the SWIM protocol using parallel discrete event simulation in the CODES framework
  - Especially useful in exploring long-running behavior
  - Offers the potential to scale to much larger sizes (not ready in time for this presentation)
  - Can be integrated with other CODES models
- Future work
  - Comparison with other group membership protocols
  - More complex failure scenarios
  - Impact of semantic differences in group membership protocols
  - Consider the use of SWIM for detection of client failures

Acknowledgements

- **Sponsor**

  This research was supported by the U.S. Department of Defense. This material also was based on work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computer Research Program under contract DE-AC02-06CH11357. The research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is a DOE Office of Science User Facility.

- www.mcs.anl.gov/research/projects/codes

- Thank you for your time!

- **Questions?**