Byzantine fault-tolerant erasure-coded storage

James Hendricks, Gregory R. Ganger
Carnegie Mellon University

Michael K. Reiter
University of North Carolina at Chapel Hill
Motivation

• As systems grow in size and complexity…
  • Must tolerate more faults, more *types* of faults
  • Modern storage systems take ad-hoc approach

• Not clear which faults to tolerate
• Instead: tolerate arbitrary (*Byzantine*) faults
• But, Byzantine fault-tolerance = expensive?
Comparison of write throughput

Bandwidth (MB/s)

Number of faults tolerated \((f)\)

- Crash fault-tolerant erasure-coded block storage (non-Byzantine)
- Low-overhead erasure-coded Byzantine fault-tolerant block storage
- Replicated Byzantine fault-tolerant block storage

- \(f+1\)-of-\(2f+1\) erasure-coded
- \(f+1\) replicas
- \(3f+1\) replicas
Homomorphic fingerprinting

- Each server gets one fragment and a small checksum value.
- Decoding from consistent fragments yields unique block.

If \{d_1, d_2\} & \{d_3, d_4\} consistent, then B = B'
Summary and status

Byzantine fault-tolerant storage can rival crash-only storage performance

Verifying distributed erasure-coded data [PODC07]
Low-overhead Byzantine fault-tolerant storage [SOSP07]

Current work: Good performance under faults
Prevent concurrency livelock (i.e., wait-freedom)
Minimize communication in worst case
Improve recovery performance