Logan: Automatic Management for Evolvable, Large-Scale, Archival Storage

Mark W. Storer Kevin M. Greenan Ian F. Adams
Ethan L. Miller Darrell D.E. Long Kaladhar Voruganti

Petascale Data Storage Workshop
November 17, 2008

Baskin Engineering
UC SANTA CRUZ

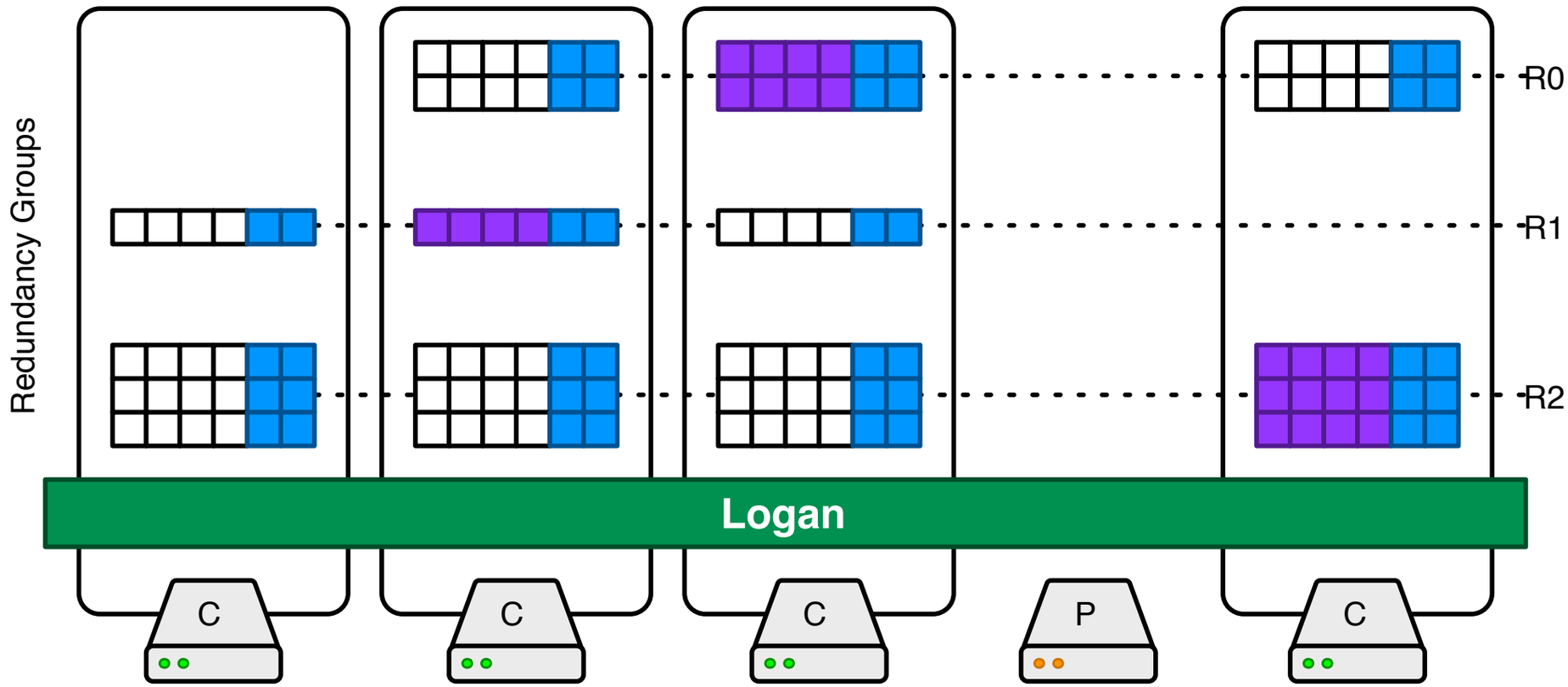
Archival Storage Problem

	Archival Storage	Traditional, Enterprise Storage
Workload	**write once**, (read maybe)	variable
Performance	adequate latency, decent throughput	low latency, high throughput
Cost	cheaper is better	commensurate to performance
Reliability	long-term	short-term
Scalability	time, capacity, technology, vendors	capacity

- Archival storage is a unique class
- Well-suited to a distributed architecture
  - Intelligent devices foster evolution
- How to manage and administer a totally decentralized system?

System Overview

Logan: management layer running on storage *tomes*



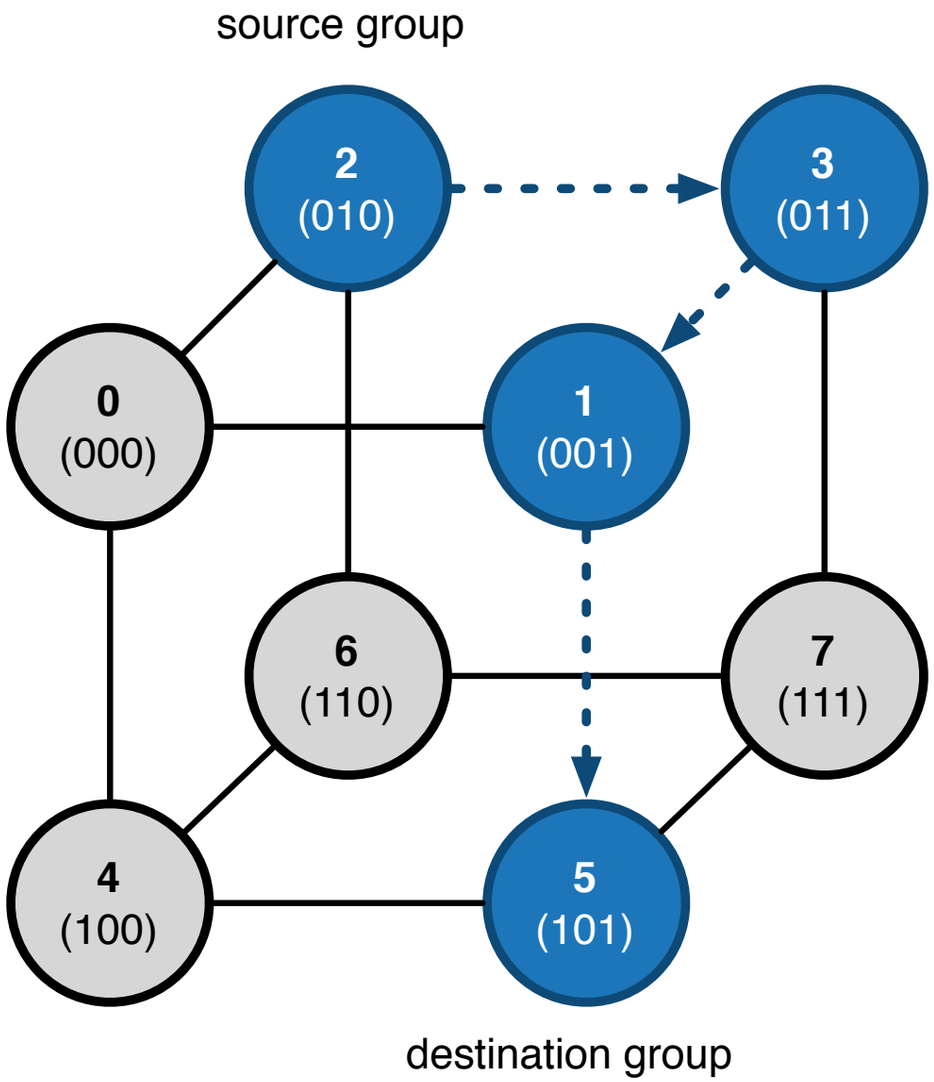
- Fixed-sized blocks arranged in fixed sized segments
- Two level reliability model:
  - Intra: Reliability encoding within segment for media faults
  - Inter: Reliability Groups (RG) across device segments

Management Goals

- Scale to hundreds of thousands (millions?) of tomes
  - Archival storage systems must scale to exabytes
- Avoid the need for global knowledge
  - Even a small amount of information per tome requires too much space
  - Global decisions don't scale well
- Avoid centralized management points
  - Potential single point of failure
  - Don't want to have "specialized" nodes
  - Monitoring issues
    - Potential bottleneck for reliability checking
    - May create problems if network is partitioned

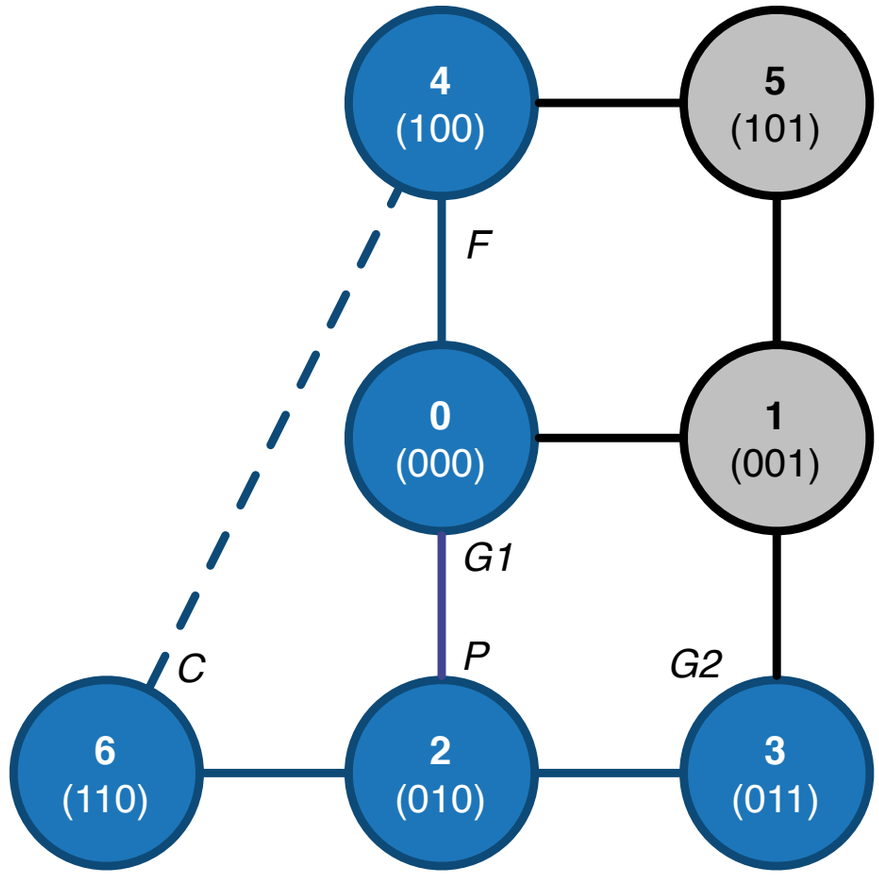
Management Groups

- Approach: group tomes into Management Groups (MG)
  - MG elects a group leader to handle group decisions
- System starts with one MG
- Groups make inter-group edges to form a hypercube
  - Inter-MG routing in log time
- When a single MG gets too large, it splits
  - Scalable hashing using LH*



Management Group Scaling

- Parent P splits to make child C
- P informs C of grandparents G1, G2,...
- C ⊕ P ∨ G = F
  - if F < C, build edge FC

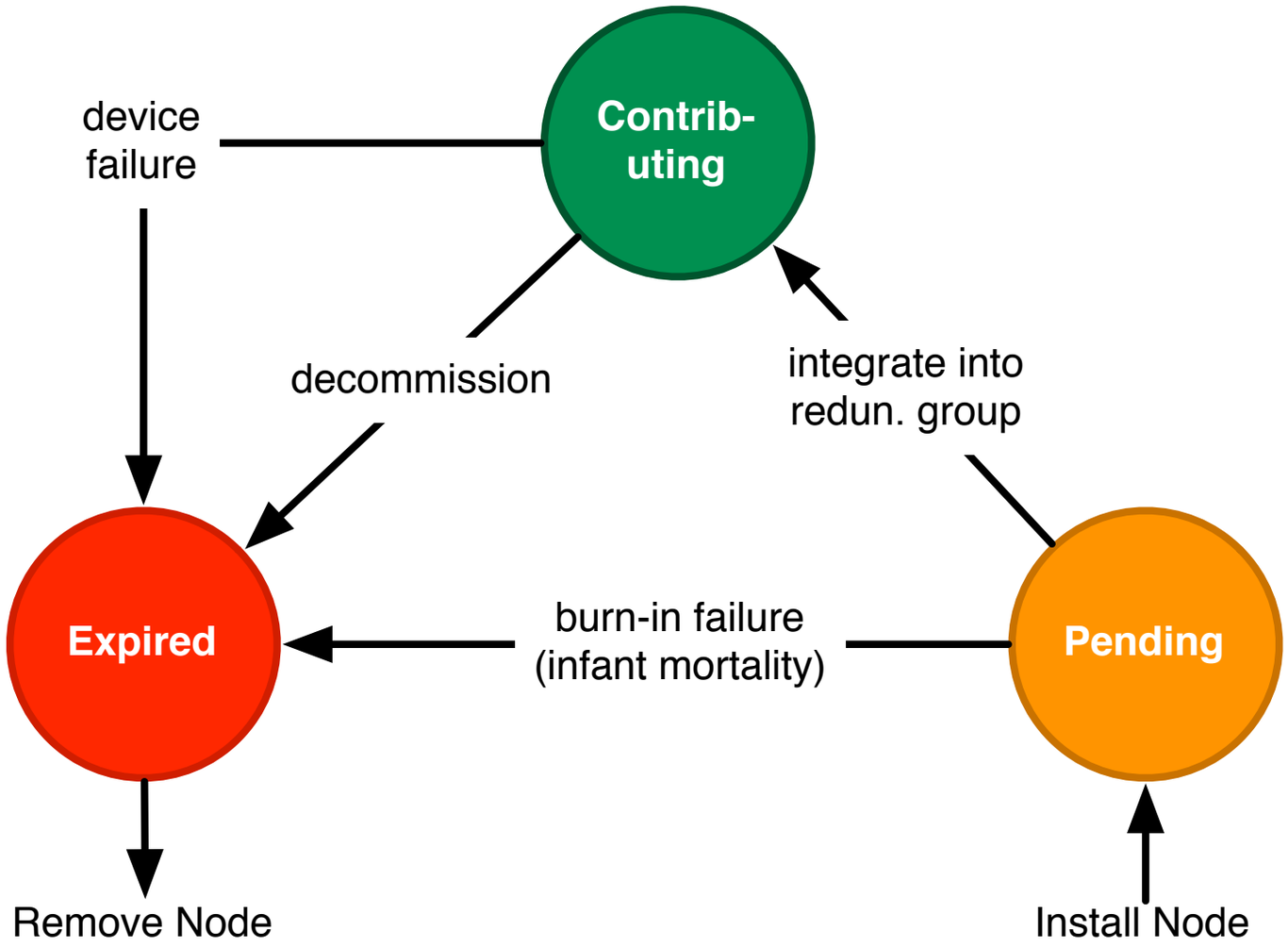


Example: Node 2 (010) splits to form Node 6 (110)

G1: ⑥ ⊕ ② v ⓪ = 100 = ④ < ⑥ build ⑥ ④

G2: ⑥ ⊕ ② v ③ = 111 = ⑦

Device States



New device enters system

1. Broadcast for other nodes
2. Obtain LH* info
3. Calculate MG
4. Contact MG leader
5. Enter MG in pending state

- Nodes progress through three states:
  - **Pending**: alive, known to the system, not in any RGs
  - **Contributing**: active member of one or more RGS
  - **Expired**: failed or decommissioned

Data-Driven Decision Making

- Devices maintain list of self-descriptive attributes
  - e.g.: segment count, power consumption, age, etc.
- Leader collects attributes to develop group statistics
- Heuristic algorithms use statistics to find "good" segment to RG mapping

- Use simulated annealing to make decisions:
  - Solution space (*X*): viable mappings of segments to RGs
    - Either new mappings or replacement mappings
  - Neighbor func. (*N*): randomized segment to RG mapping
    - Differs per management task
  - Objective func. (*P*): minimize cost per segment

Management Tasks

- Scale-out: adding capacity to the system
  - Create new redundancy groups
  - Add device to existing redundancy group
  - *N*: random mapping of unassigned segments to RGs

- Recovery: map new segments to failed segments
  - *N*: random mapping of new, replacement segments

- Maintenance: max. efficiency by decommissioning wasteful devices
  - *N*: randomly swap out wasteful devices and replace them with segments from other devices
  - Maintenance decision can be done opportunistically
    e.g. wait for next scrub cycle to scrub & copy

Status and Future Work

- Current: implementation and experimentation
  - How many devices per management group?
  - How best to weight attributes in heuristic algorithms?
  - Secure leadership elections

- Dealing with network partitions
- Geographically diverse redundancy
- Optimizing device dependency lists
- Recovery schedule

Questions

- Thanks to our sponsors:
  - Petascale Data Storage Institute
  - SSRC industrial sponsors

- Thanks to team members
  - Mark W. Storer
  - Kevin M. Greenan
  - Ian F. Adams
  - Ethan L. Miller
  - Darrell D. E. Long
  - Kaladhar Voruganti