Walking toward moving goalposts: agile management for evolving systems

Richard Golding, Theodore Wong
IBM Almaden Research Center

16 June 2006
Main points

- **Bolt-on management considered harmful**
- Proponents of building self-management into system
  - Bolt-on = management separate from and external to system under management
- An architecture pattern for building distributed systems: layering and federation
- Investigating simplest possible specifications
Research direction

• Self-management without a central management authority
  • In a storage system as an example
• Can we make a system administratorless?

• Almost all storage systems use centralized management
  • Metadata server
  • Exception is peer-to-peer... but most of those are limited in function (e.g. read-only)
Why is this decentralization worth while?

• Some environments require it
  • Cooperating organizations—no single business authority
  • On-demand provisioning from competing service bureaux
• Possible route to aligning vendor economic incentives
  • Systems *do* get smarter over time
  • Currently: system vendors have incentive for incompatible differentiation
• Can a higher-level standardized interface help?
Architecture: layering

• For any given problem:
  • Delegate to lower level?
  • Use global view of higher level?
Simplest possible specification

• Desire for human understandability
• How well will it go if we start from minimum possible?
  • Existing storage management started from high-fidelity
  • Is directionally accurate sufficient? 90% solution? Iterative tuning?
• Can we mask local complexity?
  • Makes global decision algorithms easier
  • Smart local resource management
K2 distributed storage system

• Vehicle for research—*not* a product
• No central administration; federate when global view needed
• Delegate function to as low a level as possible
• Provide support to higher-level application management
Resource pools: external view

- A virtual collection of storage
- One per user or application
- Each pool is independent
- Specified by:
  - Capacity, Performance, Reliability
  - Reserve and limit

- Initially: capacity = bytes; performance = IO/s; reliability = MTTDL
Implementing pools

- Virtual pool backed by physical allocation pools
- Pools contain objects for storing user data
- Decision algorithm: how much to put where
- Storage server enforces resource allocation
Resource allocation decisions: normal

- Normal case: online decision for one pool
  - Creating or modifying a pool’s requirements
  - Load balancing
- Use constrained multidimensional bin packing
- Constraints derived from reliability requirements
Resource allocation decisions: failure

- Multi-pool assignment required
- Backtracking search for feasible solution (better is possible)
Resource allocation decisions: failure

- Multi-pool assignment required
- Backtracking search for feasible solution (better is possible)
Making decisions

- Each resource pool is an independent group
- APs elect a manager; manager watches over pool
- Manager is disposable
- Manager runs decision algorithm
- All information in allocation pools
Local resource management

- Goal: isolation between pools
- Capacity: just accounting
- Performance: requires scheduler
- Tradeoff: performance vs. efficiency
- Provides reserve and limit, plus fair sharing
- Working to add cache, network
Contacts and information

• Richard Golding  rgolding@us.ibm.com  
  http://soe.ucsc.edu/~golding

• Theodore Wong  theowong@us.ibm.com  
  http://www.tmwong.org