Quantifying Availability/Performance Tradeoffs in Distributed Data Structures

Noah Treuhaft
UC Berkeley ROC Group

ROC Retreat, January 2002
Outline

• Motivation
• Distributed data structures
• A shared-disk DB toolkit
• Quantifying the tradeoffs
• Status
Motivation

• Many interactions between availability and performance in systems
  - some are synergies (DB index “structure modifying operations” as “nested top actions”)
  - others are tradeoffs (transaction throughput)

• ROC principle: availability is not subordinate to performance
  - the application determines the appropriate balance...
  - and that guides us through the tradeoffs
Motivation (2)

• Implication for systems research: lead by building tunable systems
  - but must ensure that people understand how to tune them!
  - unlabeled knobs are useless

• Key insight: quantify availability/performance tradeoffs with availability benchmarking
  - hard work, so don’t make system users do their own benchmarking
Outline

• Motivation
• Distributed data structures
• A shared-disk DB toolkit
• Quantifying the tradeoffs
• Status
What’s a distributed data structure (DDS)?

- Interface like a centralized data structure
  - uniform access from all cluster nodes
- Updates
  - consistency model
- Persistent
- Out-of-core
- Building block for Internet-style services
  - provides persistent state management
  - “high” throughput AND “high” availability
  - service inherits tradeoffs from DDS
Gribble's prototype DDS: distributed hash table

clients interact with any service “front-end” [all persistent state is in DDS and is consistent across cluster]

service interacts with DDS via library [library is 2PC coordinator, handles partitioning, replication, etc., and exports hash table API]

“brick” is durable single-node hash table plus RPC skeletons for network access

element of a distributed HT partition with 3 replicas in group

from a presentation by Steve Gribble
Outline

- Motivation
- Distributed data structures
- A shared-disk DB toolkit
- Quantifying the tradeoffs
- Status
Berkeley DB overview

- Great for persistent state management
  - and more
- Access methods for unordered and ordered data
  - hash table and B-tree
- Transactions
- Runs on a single machine
Berkeley DB architecture

Applications

Access Methods

Lock

Transactions

Buffer Pool

Log
Shared-disk DB architecture

Cluster node
Outline

• Motivation
• Distributed data structures
• A shared-disk DB toolkit
• Quantifying the tradeoffs
• Status
Two tradeoffs

• *Concurrent intersystem page modification*
  - log merge required during recovery
  - reduced page contention
  - page transfers replaced by log-record transfers

• *“Hot” page replication*
  - immediate page recovery
  - reduced logging?
  - memory overhead
  - two-phase commit overhead
Availability benchmarking 101

- Availability benchmarks quantify system behavior under failures, maintenance, recovery

- They require
  - a realistic workload for the system
  - quality of service metrics and tools to measure them
  - fault-injection to simulate failures
  - human operators to perform repairs

from a presentation by Dave Patterson
Outline

• Motivation
• Distributed data structures
• A shared-disk DB toolkit
• Quantifying the tradeoffs
• Status
Status

• Getting familiar with Berkeley DB
  - implemented TPC-B
  - looking through the source code

• Combing through shared-disk DB research literature

• Identifying availability/performance tradeoffs
  - others will appear during implementation