Novel File Systems

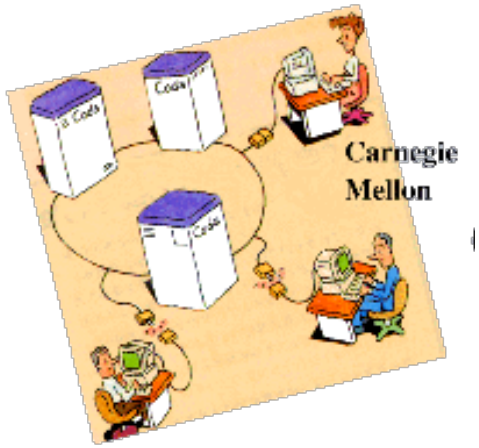
Presented by Ki Suh Lee
(Based on slides from
Hakim Weatherspoon and
Mahadev Sataynarayanan)

M. Satyanarayanan

- Systems faculty at Carnegie-Mellon University
- A principal architect and implementor of the *Andrew File System (AFS)*

- Inspired CODA and another 20 years of research

Coda

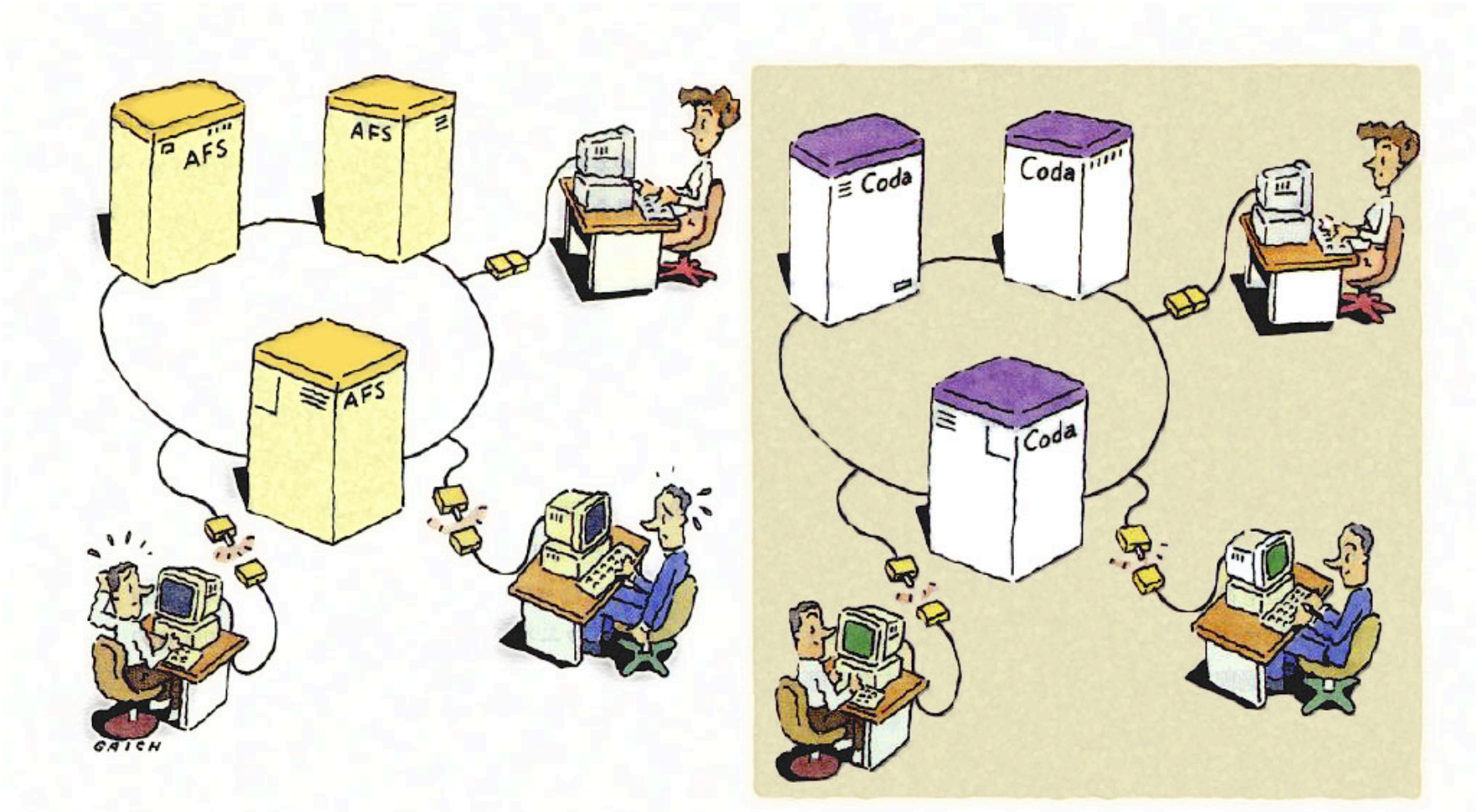


- Coda is a distributed file system
- Since 1987
- The latest release is Coda 6.9.4 (Jan 09)
- 47 publications including 6 PhD Theses
- Evolved with usage experience.

Motivation for Coda

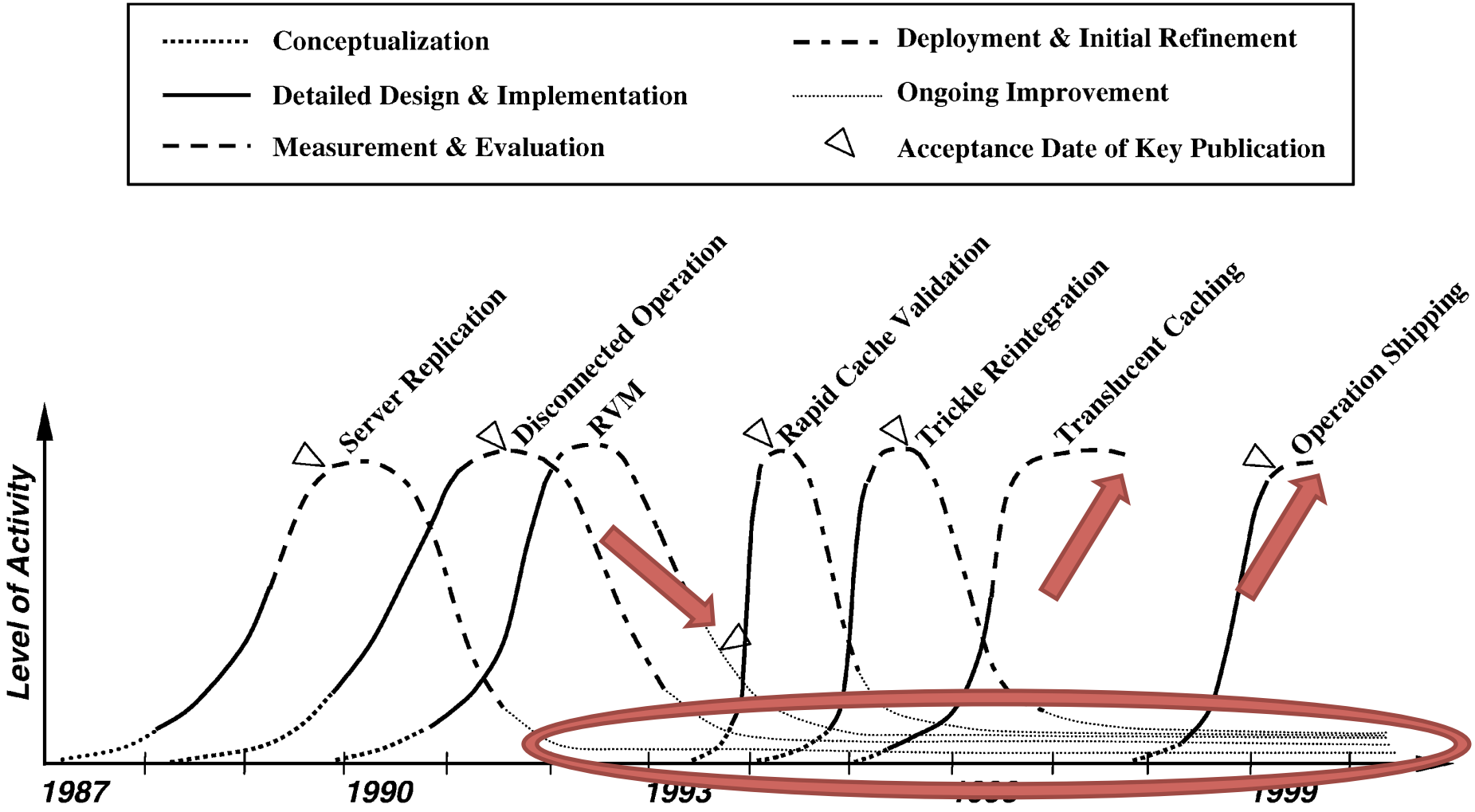
- Epilogue to the Andrew File System (AFS)
- AFS was found to be vulnerable to ***server and network failures***
  - Not that different from NFS
  - Limits scalability of AFS
- Coda provides high data availability
  - ***Server replication*** (optimistic replication)
  - ***Disconnected operation***

Coda Cartoon



http://www.cs.cmu.edu/~coda/images/nikkei-PAGE-213.JPG

Timeline


Conceptualization
Detailed Design & Implementation
Measurement & Evaluation
Deployment & Initial Refinement
Ongoing Improvement
Acceptance Date of Key Publication
Level of Activity
Server Replication
Disconnected Operation
RVM
Rapid Cache Validation
Trickle Reintegration
Translucent Caching
Operation Shipping
1987
1990
1993
1999

Lessons Learned

- Optimistic Replication
- Real systems need real users
- Timing
- Long software tail
- Moore's Law
- Code reuse
- System admins
- Short projects never die

Outline

- Intro to Coda
- Server Replication
- Recoverable Virtual Memory (RVM)
- Disconnected Operation
- Conflict Resolution

Server Replication: 1987-1991

- ***Optimistic replication control protocols*** allow access in ***every*** disconnected mode
  - Tolerate temporary inconsistencies
  - Promise to detect them later
  - Provide ***much higher data availability***
- Optimistic replication control requires a reliable tool for detecting inconsistencies among replicas
  - Better than LOCUS tool

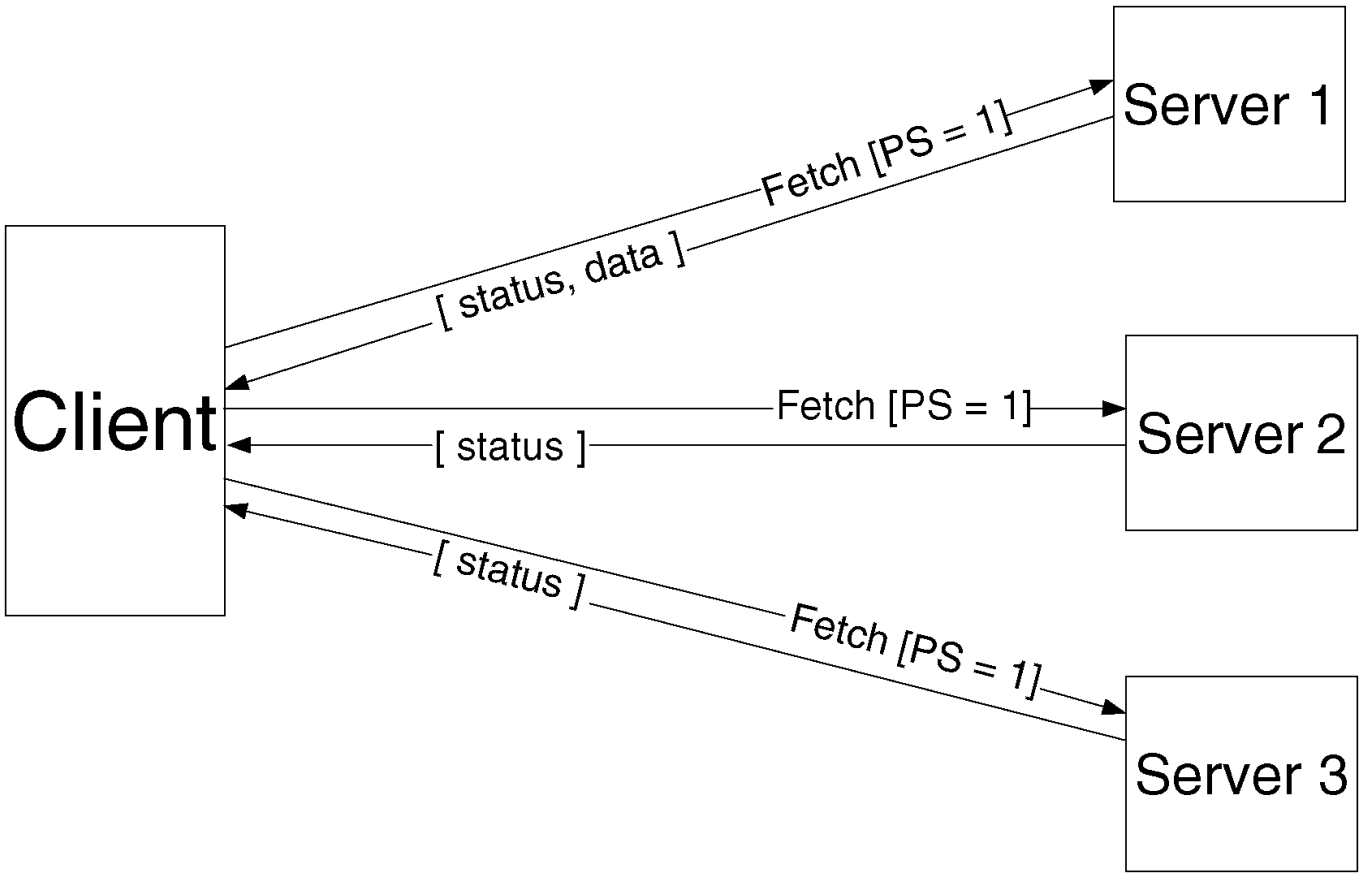
Replica Control

- Volume
- Volume storage group (VSG)
- accessible volume storage group (AVSG)
- Preferred server (PS)

- ***Read-one-data, read-all-status, write-all***

http://www.cs.cmu.edu/~coda/images/nikkei-PAGE-216.JPG

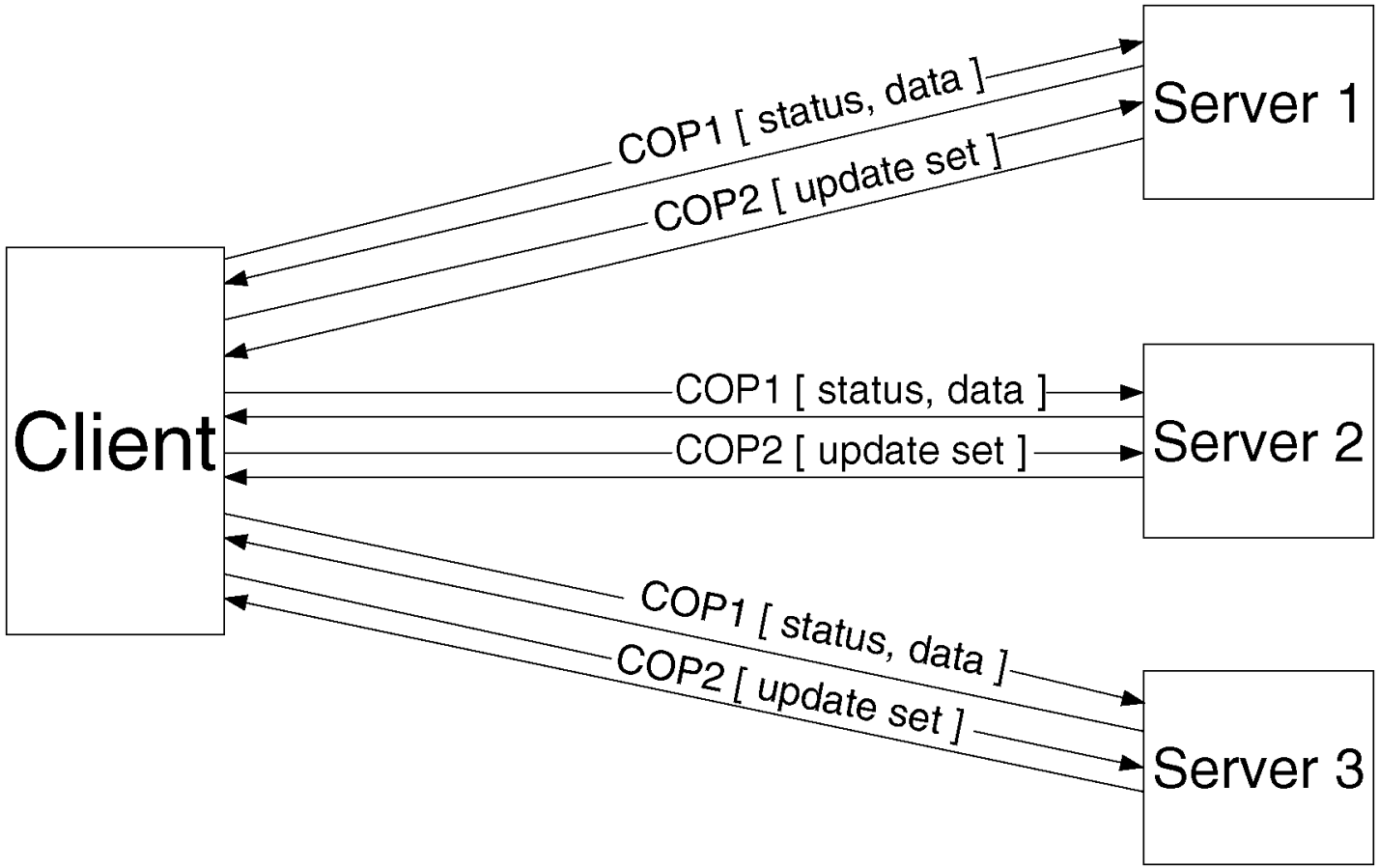
Read Protocol



Read Protocol

- Client fetches data from PS
- Also collects version information from AVSG
- If a conflict is detected
  - Abort the read
  - Update stale replicas
  - Restart the fetch

Update Protocol



Update Protocol

- File modifications and directory updates are written to all members of AVSG
- Two phases commit
  - In COP1, new status and data are sent
  - In COP2, *update set* is sent asynchronously
  - ***Non-blocking***

Consistency Model

- Accessible universe: set of replicas that the user can access
  - Accessible universe varies over time
- One-copy serializability at open-close granularity in the accessible universe
  - Will read from the latest replica in accessible universe
  - Will update all replicas in accessible universe

Fault-tolerance

- Correctness of update protocol requires **atomicity** and **permanence** of metadata updates
- ***Recoverable virtual memory*** (RVM)
  - Implemented as a library

Lightweight Recoverable Virtual Memory

- RVM updates persistent data structures in a fault-tolerant manner.
- Allows independent control over the transactional properties.
- Minimalism
  - Value Simplicity
  - portability
  - User-level library with minimal programming constraints.

Camelot

- Used for two phase optimistic replication protocol by CODA
- *Poor scalability, programming constraints*, and *difficulty of maintenance*
  - Considerable paging and context switching
  - Mach threads
  - Code size, tight dependence on Mach features
- Coda only wanted recoverable virtual memory

Design Rationale

- Keep it simple!
- Factor out
  - Nesting
  - Distribution
  - Concurrency control
  - Resiliency
- Instead, a layered approach
- Less dependence on OS
- Structured as a library

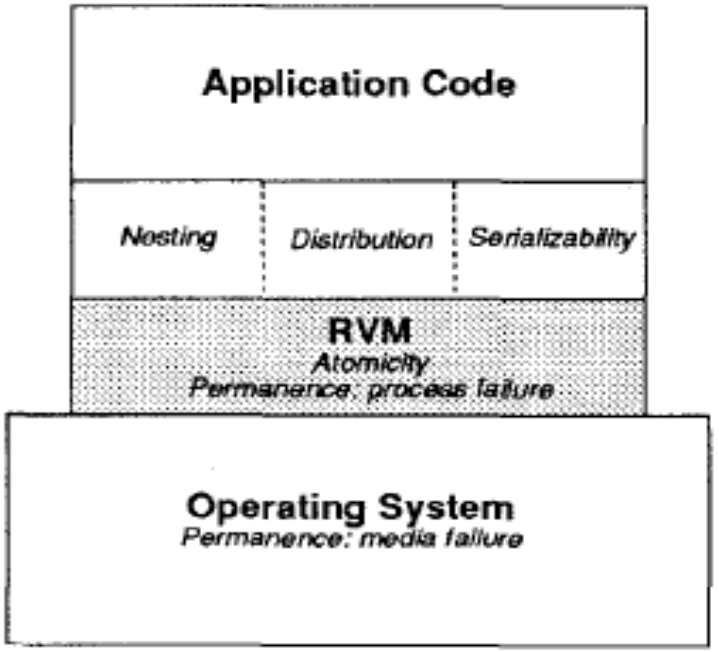
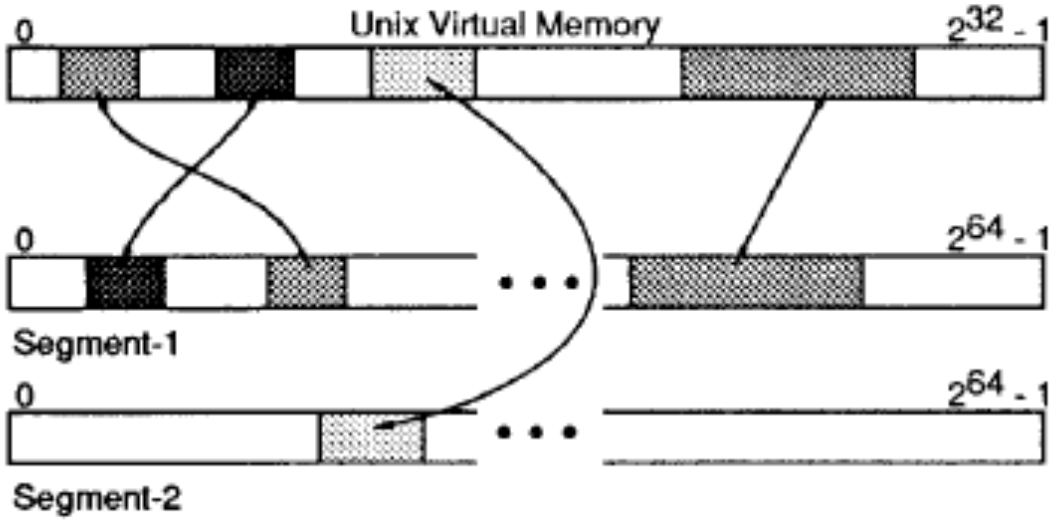


Figure 2: Layering of Functionality in RVM

LRVM: Segments and Regions

- Applications map *regions* of *segments* into their virtual memory.



Architecture

- Newly mapped data represents the committed image of the region
- No region of a segment can be mapped more then once by the same process
- No overlap
- Can be unmapped at any time, as long as no uncommitted transactions outstanding
- simple APIs

LRVM: Crash Recovery

- Recovery consists of reading the log from tail to head and then reconstructing the last committed changes.
- Modifications are applied to the external data segment.
- Log is emptied.

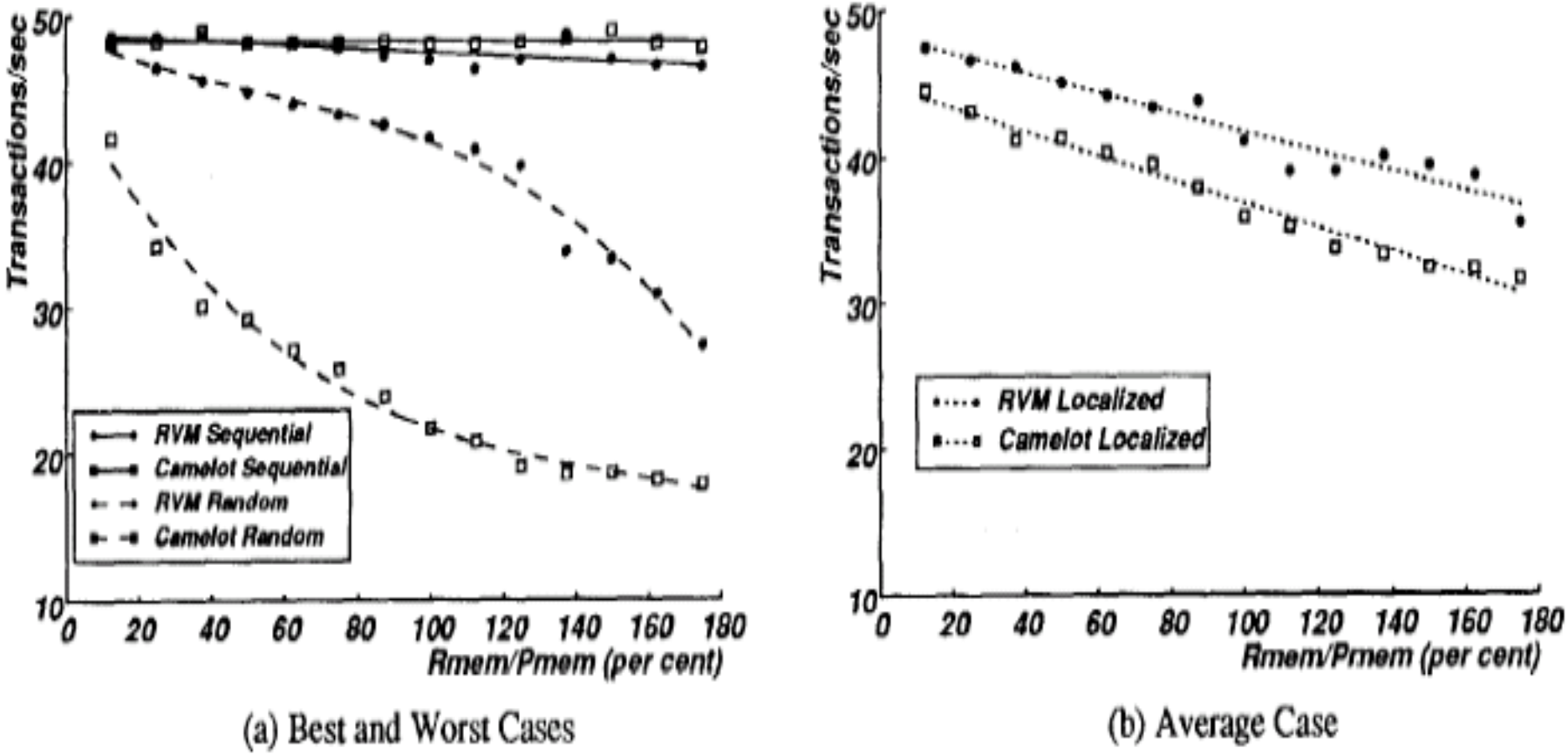
LRVM: Truncation

- Reclaiming space in the log by applying changes to the external data segment.
- Necessary because space is finite.

Evaluation

- 10K lines of C, compared to 60K lines in Camelot
- Performance
  - Lack of RVM and VM integration
  - Scalability
  - Optimization
- Overall, RVM is better then Camelot.

Transactional Throughput

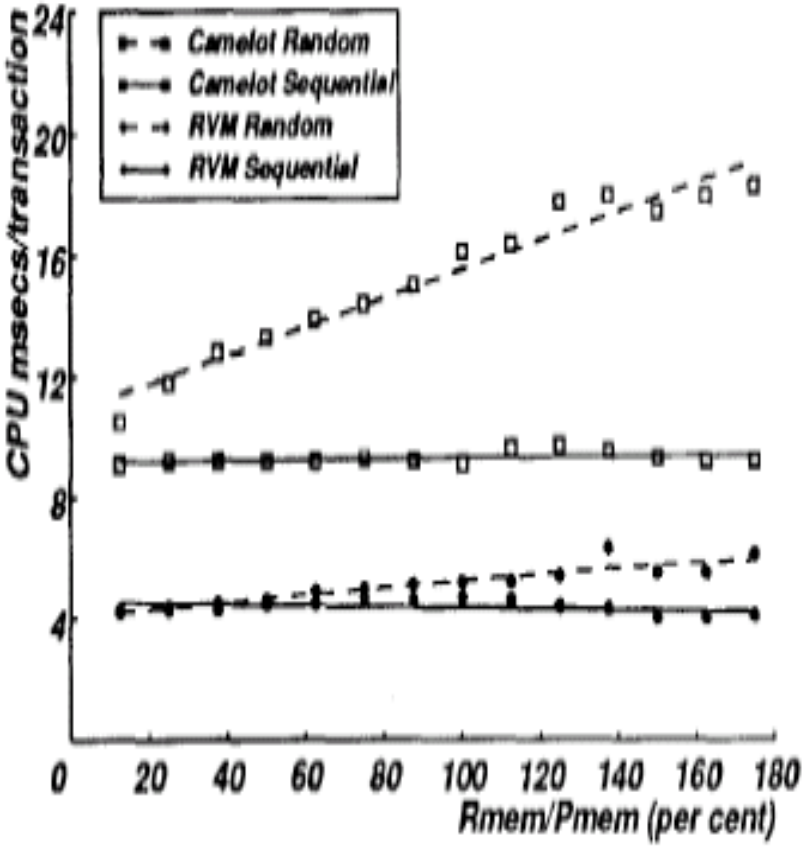


(a) Best and Worst Cases

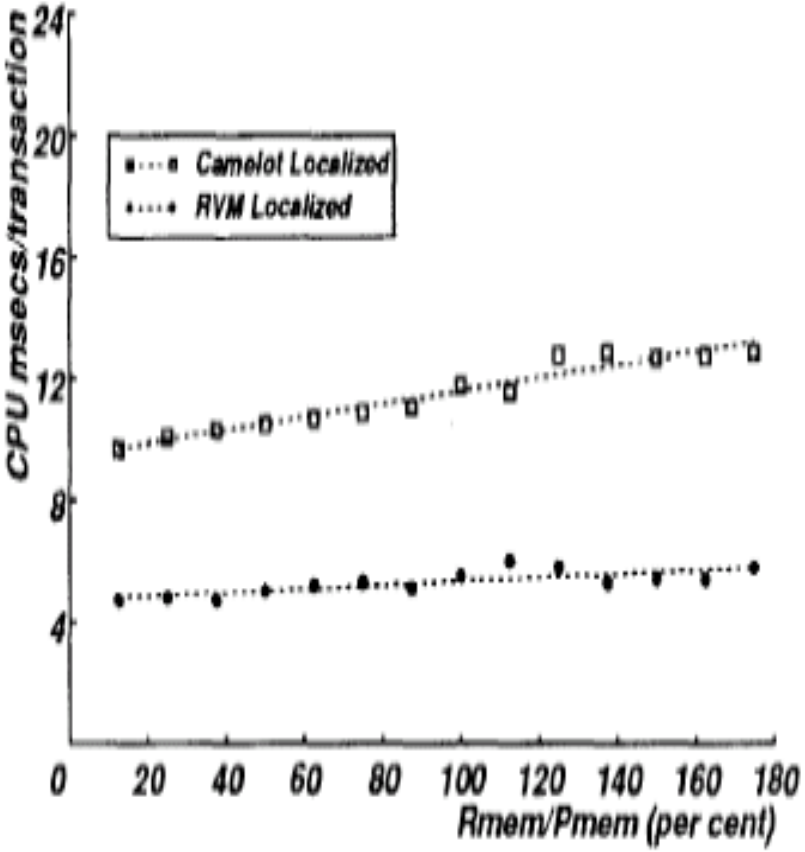
(b) Average Case

These plots illustrate the data in Table 1. For clarity, the average case is presented separately from the best and worst cases.

Scalability



(a) Worst and Best Cases



(b) Average Case

Outline

- Intro to Coda
- Server Replication
- Recoverable Virtual Memory (RVM)
- Disconnected Operation
- Conflict Resolution

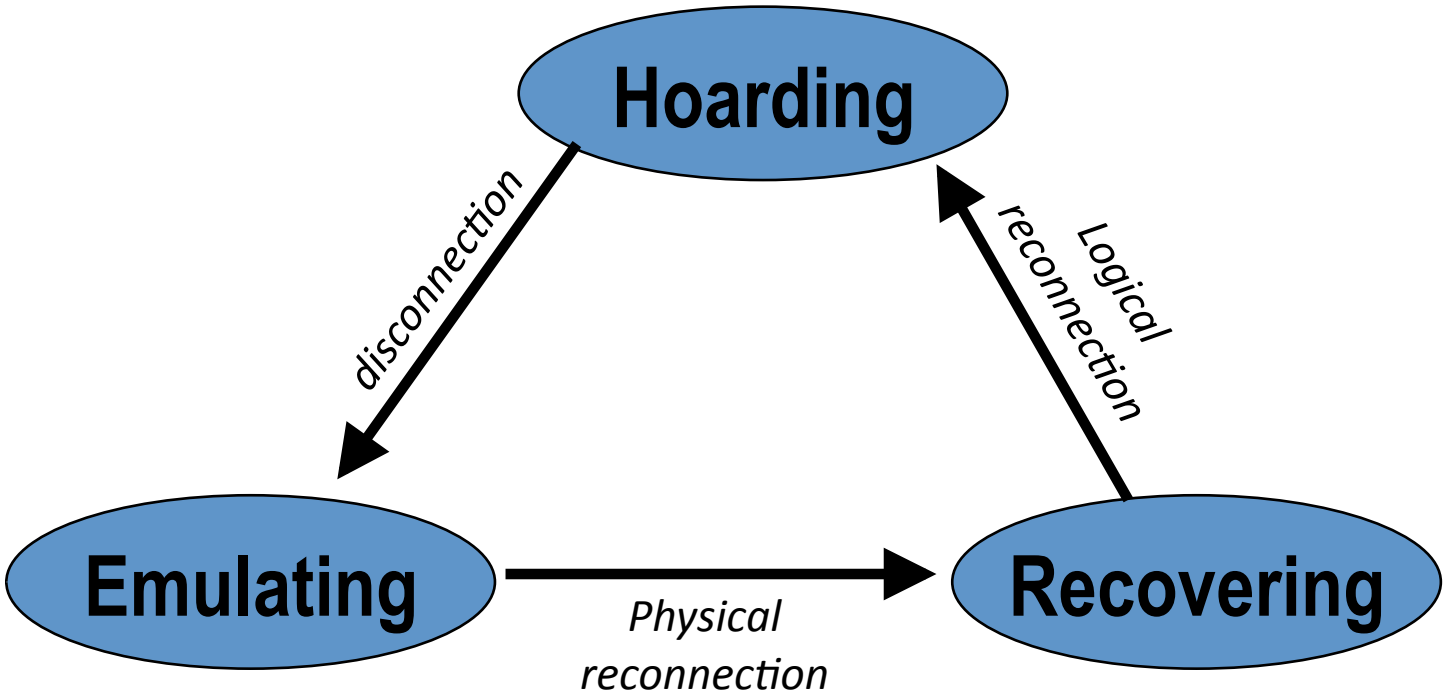
Disconnected Operation: 1988-1993

- Network isolation
  - Correlated server crashes
  - Overloaded or faulty routers
- Caching
  - Optimistic replication
  - Improve availability

Disconnected Operation

- Complimentary to server replication
  - Server replicas are *first-class replicas*
  - Cached replicas are *secondary replicas*

Implementation



Hoarding

- Implicit / Explicit sources of implementation
- Hoard database (HDB)
  - Specifies files to be cached
  - Users update directly or via command scripts
  - Venus periodically reevaluates every ten minutes

Emulation / Reintegration

- While disconnected, All changes are written to a log, Client modification log (CML)
- After reconnection, perform reintegration for each volume independently.
  - Venus sends CML to all volumes
  - Each volume performs a log replay algorithm

Outline

- Intro to Coda
- Server Replication
- Recoverable Virtual Memory (RVM)
- Disconnected Operation
- Conflict Resolution

Conflict Resolution

- **Syntactic approach** to detect absence of conflicts.
  - Server replication
  - Version checks during the hoarding and reintegration
- **Semantic approach** to resolve conflicts
  - Different control for directory and file resolution

Conflict Resolution: Objectives

- *No updates should ever be lost without explicit user approval*
- *The common case of no conflict should be fast*
- *Conflicts are ultimately an application-specific concept*
- *The buck stops with the user*

Directory Resolution

- Automatic resolution by Coda
- After disconnected operation
  - Apply the CML
- During connected operation
  - Resolution log
  - Recovery protocol locks the replicas merges the logs and distributes the merged logs.

Application-Specific File resolution

- ***Application-specific resolvers*** (ASRs) are executed entirely on clients.

Conflict representation

```
$ ls -l
total 364
-rw-r--r--   1 satya    4976 Jun 28 08:42 cevol99.aux
lr--r--r--   1 root       27 Jun 29 11:08 cevol99.bib -> @7f0004e6.0000a52c.0000b7dc
-rw-r--r--   1 satya     528 Jun 28 08:42 cevol99.err
-rw-r--r--   1 satya   87070 Jun 28 08:41 cevol99.mss
-rw-r--r--   1 satya    6937 Jun 28 08:42 cevol99.otl
-rw-r--r--   1 satya  267914 Jun 28 08:42 cevol99.ps
```

(a) Before repair

```
$ls -lR cevol99.bib
total 75
-rw-r--r--   1 satya   26290 Jun 29 11:04 marais.coda.cs.cmu.edu
-rw-r--r--   1 satya   20286 Jun 29 11:03 mozart.coda.cs.cmu.edu
-rw-r--r--   1 satya   26290 Jun 29 11:04 verdi.coda.cs.cmu.edu
```

(b) During repair

```
$ ls -l
total 390
-rw-r--r--   1 satya    4976 Jun 28 08:42 cevol99.aux
-rw-r--r--   1 ras     26290 Jun 29 11:09 cevol99.bib
-rw-r--r--   1 satya     528 Jun 28 08:42 cevol99.err
-rw-r--r--   1 satya   87070 Jun 28 08:41 cevol99.mss
-rw-r--r--   1 satya    6937 Jun 28 08:42 cevol99.otl
-rw-r--r--   1 satya  267914 Jun 28 08:42 cevol99.ps
```

(c) After repair

Lessons Learned

- Optimistic Replication
- Real systems need real users
- Timing
- Long software tail
- Moore's Law
- Code reuse
- System admins
- Short projects never die

Discussion