Caching

"Making Far Look Near"

The *only* technique to reduce end-to-end latency

Basic Concept

Bare-bones idea

1. **You need to make multiple references to an object**
2. **The object is far away**
3. **You make a copy close by and access it instead**
4. **Do all this transparently to programs and users**

Local storage used for copies is referred to as "cache"

"Multiple references" $\rightarrow$ caching useless for just one reference

- **on any reference *i* how do you know there will more?**
- **typical assumption: one reference $\Rightarrow$ others likely**
- **empirical observation about real systems in real use**
- ***"temporal locality of reference"* or just *"temporal locality"***
- **assumption sometimes fails to be true $\rightarrow$ caching wasteful**

Simple Cache Metrics

References $\rightarrow$ number of attempts to find an object in the cache

Hits $\rightarrow$ number of successes

Misses $\rightarrow$ number of failures

Miss Ratio **= Misses/References**

Hit Ratio = Hits/References = (1 − Miss Ratio)

Expected cost of a reference = (Miss Ratio * cost of miss)
+ (Hit Ratio * cost of hit)

Cache Advantage **= (Cost of Miss / Cost of Hit)**

(where cost is measured in time delay to access object)

Key Questions

1. *What data should you cache and when?*

   Fetch policy

2. *How do updates get propagated?*

   Update propagation policy

3. *What old data do you throw out to free up space?*

   Cache replacement policy

Caching is Widely Applicable

User

Applications
(Outlook, …)

Middleware
(WebSphere, Grid tools, …)

Distributed Systems
(distrib. file sys, Web, DSM, …)

OS
(virtual memory, file systems, databases, …)

Hardware
(on-chip, off-chip, disk controllers, …)

- Variable size more common
- More time for decision making
- More space for housekeeping
- More complex success criteria
- Less temporal locality
- Less spatial locality
- Higher cache advantage common

Caveat: these are "soft" differences

- Fixed size almost universal
- Fast, cheap decisions essential
- Miss ratio says it all (all misses equally bad)
- Greater temporal & spatial locality

In this class, we’ll focus on caching in distributed file systems

Just one of many levels at which caching can be applied

Fetch Policy

How Do You Know What to Cache?

Approach 1: ***Full Replication***

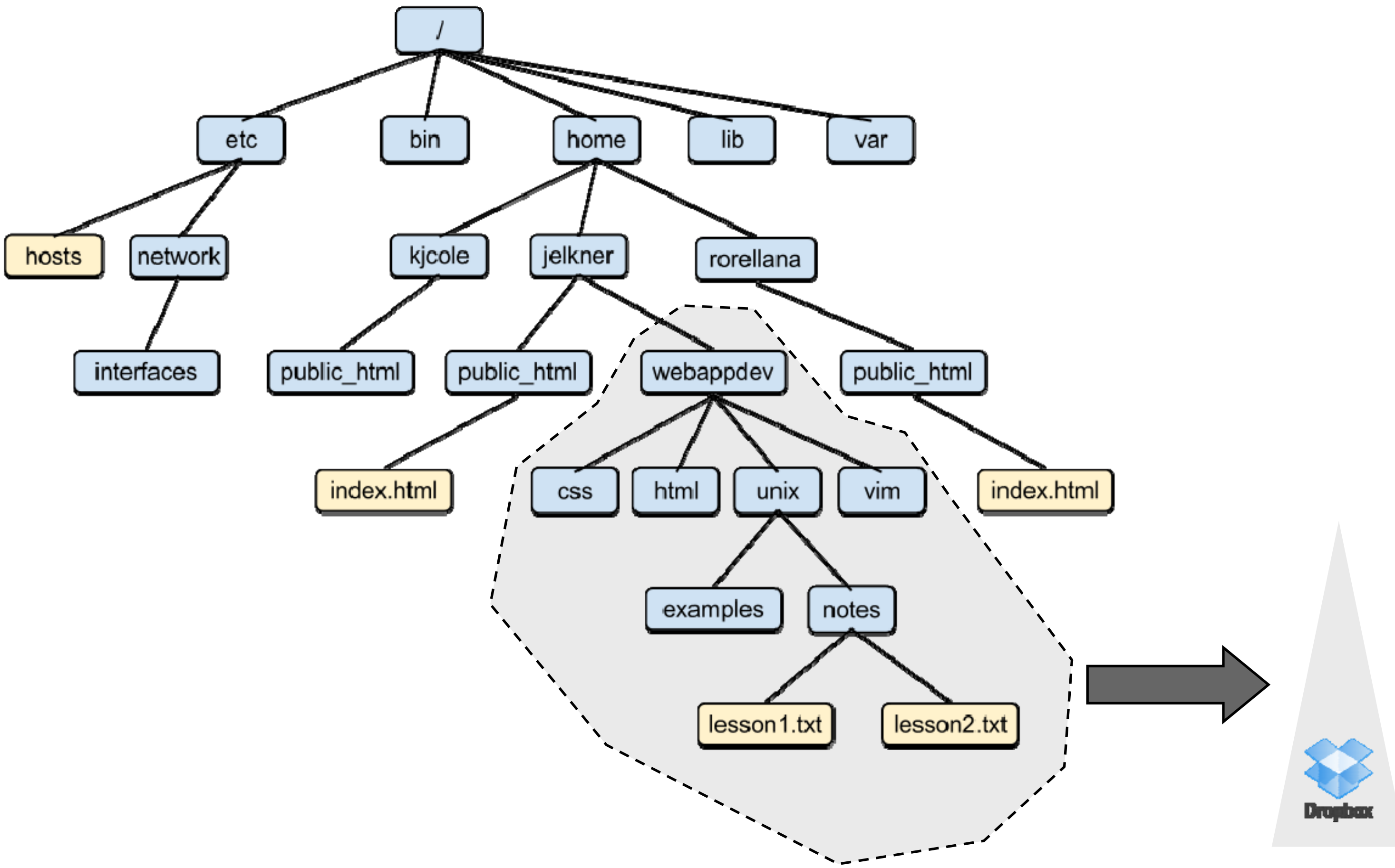
Used by DropBox and other similar services

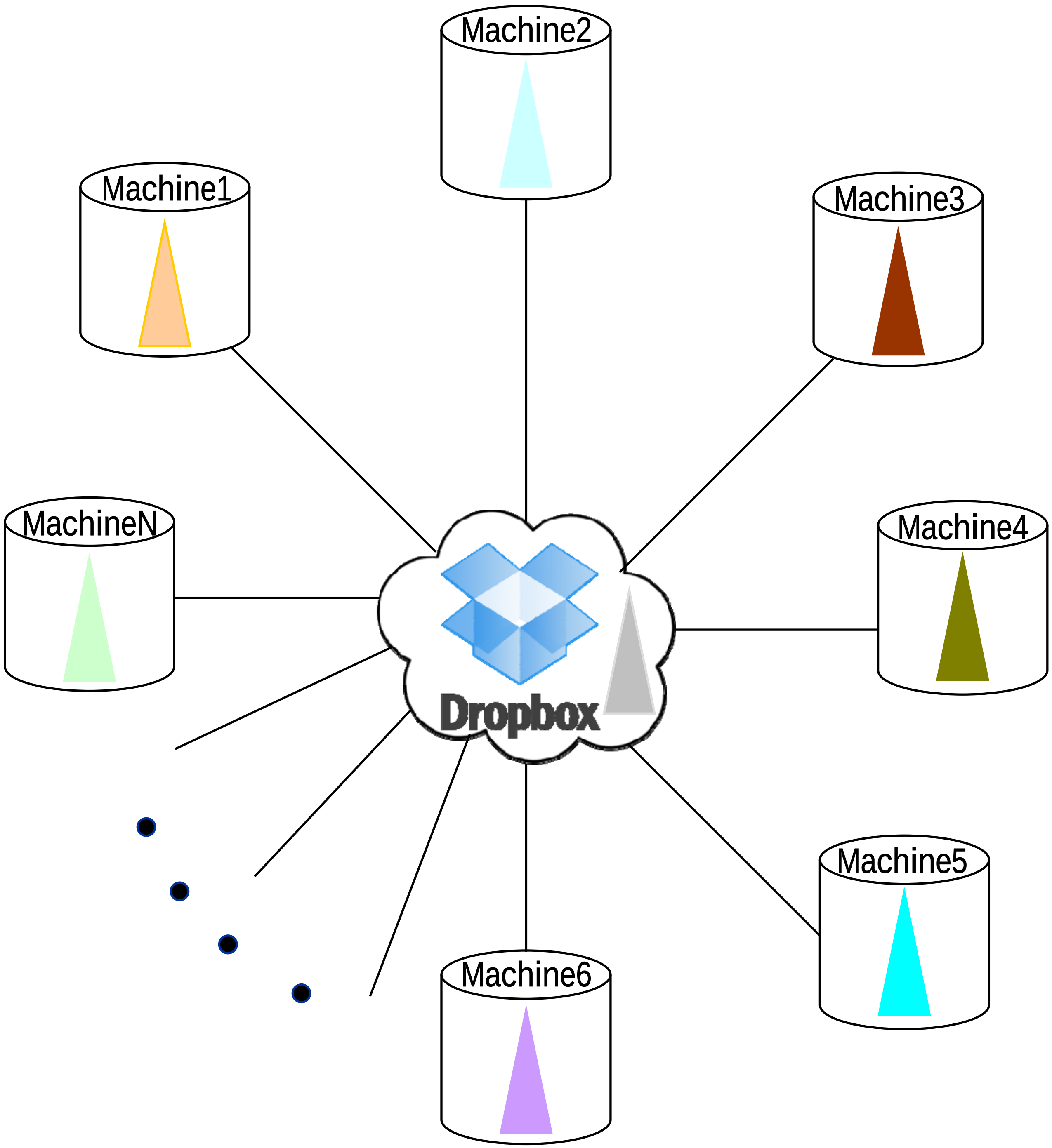
Designate a subtree as backed by DropBox

1. ***every participating machine gets a full and complete copy***
2. **every new file gets transmitted to all replicas**
3. **every updated file gets propagated**
   no well-defined semantics for when updates are propagated

All data is fetched in advance

Place Entire Subtree in DropBox




Machine2
Machine1
Machine3
MachineN
Machine4
Dropbox
Machine5
Machine6

Shortcomings of DropBox Approach

1. Storage for entire subtree consumed on every replica

2. Significant update traffic on hot spots

   painful on metered networks (e.g. 4G LTE)

   no well-defined semantics for when you see updates

3. Machines receive updates whether they care or not

   aka “push” model of update propagation

Coarse-grain, non-selective management of data

DropBox Approach Works "Well Enough"

Technical excellence is only weakly correlated with business success

Dropbox Has Raised $350M In New Funding At A $10B Valuation

Posted Feb 24, 2014 by Anthony Ha (@anthonyha)

981 SHARES

Next Story

A regulatory filing seems to confirm reports from the past couple of months that Dropbox has raised a large round of additional funding.

Back in January, the Wall Street Journal said that the cloud storage and sharing company had raised an additional $250 million at a $10 billion valuation. Then, in February, it updated that number to $350 million (at the same valuation) from investors including BlackRock, T. Rowe Price, and Morgan Stanley. However, Dropbox did not confirm the stories.

CrunchBase

A Much Better Approach

Transparently fetch file only if needed: ***on-demand caching***
(aka “demand caching”)

- **approach used in AFS**
  inherited and extended from AFS-2 by Coda File System
- **requires integration with the operating system**

Fine-grained and selective approach to data management

Optional reading

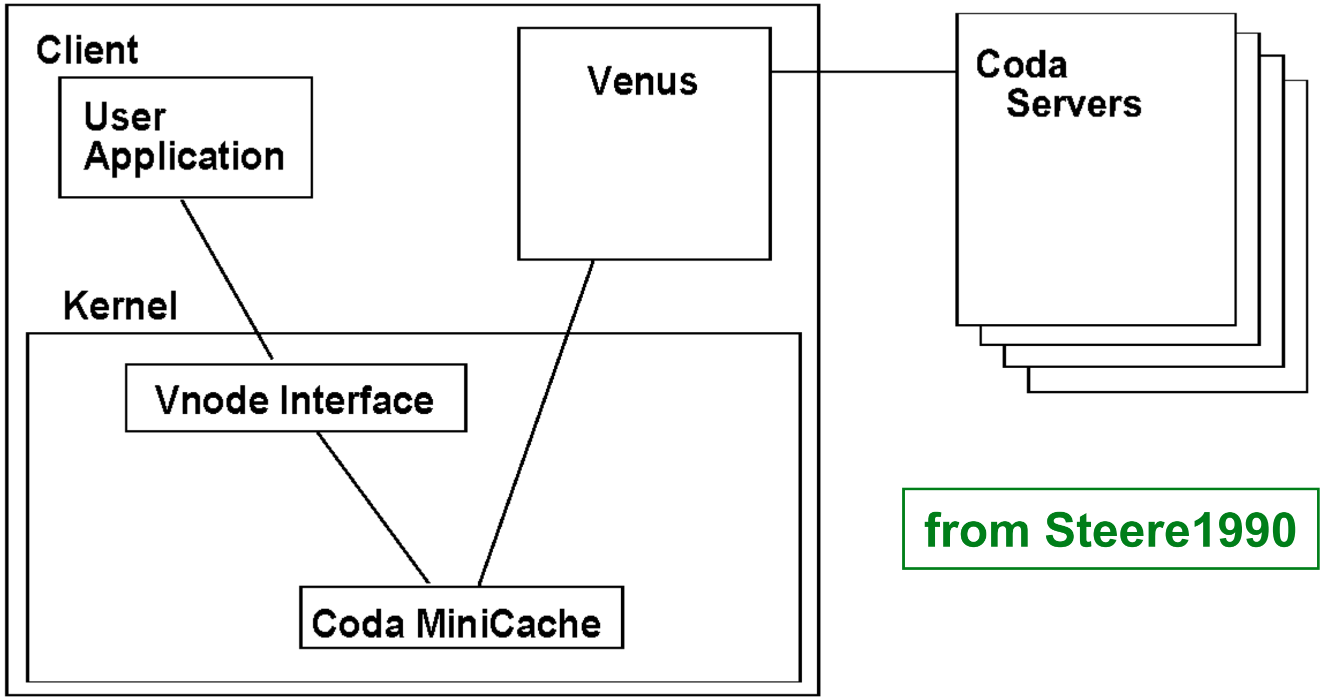
“Efficient User-Level File Cache Management on the Sun Vnode Interface”

Steere, D. C., Kistler, J. J. , Satyanarayanan, M.

Proceedings of the Summer Usenix Conference, Anaheim, CA, June 1990

Support first introduced into Linux by Coda File System

- **now standardized as FUSE module**
- **“FUSE” → “file system in user space”**
- **original Coda kernel module continues to exist in Linux kernel**



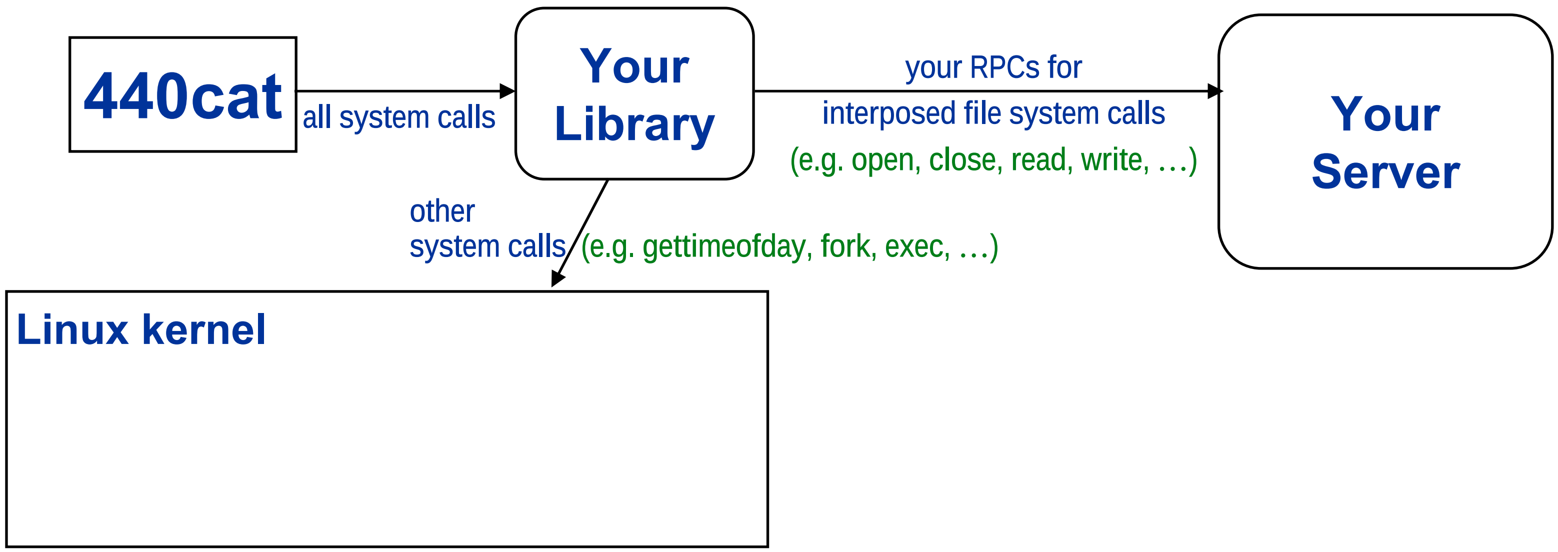
from Steere1990

- requires operating system modifications

+ *total application transparency*

+ enable demand caching

Project 1



Project 1 also avoided kernel modifications

Simplifies development and debugging

Application transparency sacrificed

hence need for 440cat, 440ls, etc.
the "real" cat, ls etc. use fopen, fread, etc.

Why Did DropBox Go Retro?

The AFS/Coda approach dates back to the mid-1980s

- **DropBox was created ≈ 2007**
- **founders of DropBox had used AFS extensively at MIT**
  AFS was part of the Athena environment at MIT since ~1987
- **felt pain when the AFS no longer accessible to them after graduation**
  created DropBox to address this pain

With Sync Solved, Dropbox Squares Off With Apple's iCloud

BY RACHEL SWABY 12.22.11 | 6:30 AM | PERMALINK

Share 0 Tweet 0 +1 137 Share Pin it

DropBox approach simplifes OS portability

- **Linux, Windows, iOS, Android, …**
- **simplifies software development time/cost**

DropBox is essentially AFS - -

- **2011 Wired Magazine article**
- **see www.wired.com/2011/12/backdrop-dropbox/all/**

In early 2009, just months after Drew Houston and Arash Ferdowsi launched

Multi-OS On-Demand Caching

It is possible, but takes enormous technical skill

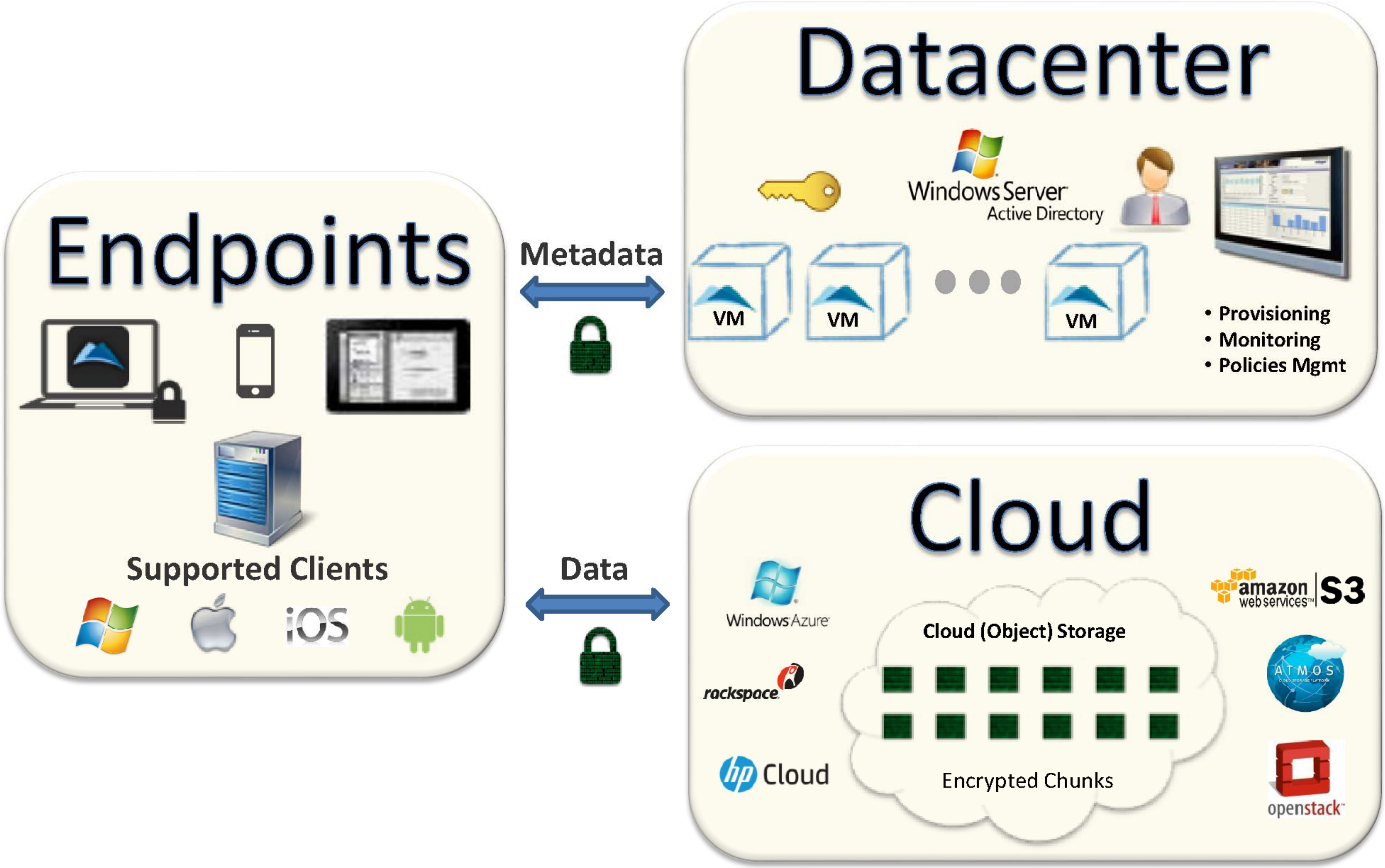
Implemented in MagFS (2010-2014), by CMU founders of Maginatics

- uses on-demand caching based on FUSE
- completely transparent to applications (just like AFS and Coda)

Purchased by EMC in November 2014

(purchase price large, but not public)

MagFS Deployment Model

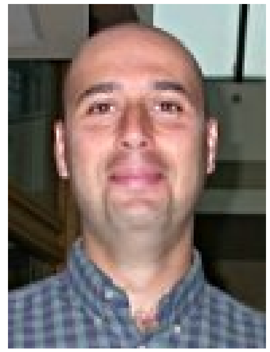


Very Strong CMU Roots!

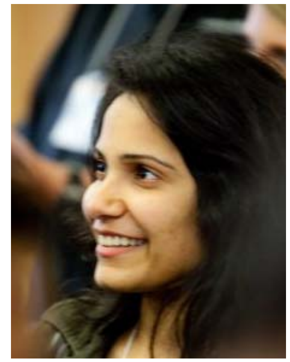
(over one-third of the company)

Jay Kistler
CTO & co-founder
PhD-CSD 1993

Niraj Tolia
Chief Architect
BS-ECE 2002
MS-ECE 2003
PhD-ECE 2008



Julio Lopez
PhD-ECE 2007



Deepti Chheda
MS-INI 2007

Rajiv Desai
MS-INI 2008

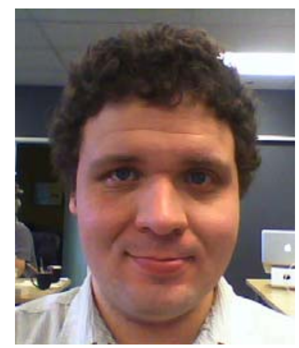
Konteya Joshi
MS-SE 2006

Vaibhav Kamra
BS-ECE 2003
MS-ECE 2004

Akshay Moghe
MS-ECE 2008

Vijay Panghal
MS-INI 2009

Vibhav Sreekanti
BS-CS 2009



Mark Schreiber
BS-CS 2003