Project Assignments Done

Meet with your mentor ASAP

- **work out details of your project, get equipment**
- **get started working on it, so you get to grips with the real issues fast**
- **“Fail early” – if any changes needed to project, discover this soon**

Checkpoint-1 is on September 17 **(two weeks from now)**

Checkpoint-1: Sep 17

online score sheet

What we are trying to figure out:

- **Are you clear on the problem being addressed?**
- **Have you broken up the work into small steps?**
- **Are you realistic about how long each step will take?**
- **Will you need any special resources**
  (e.g. hardware, cloud resources, cloudlet resources, etc.)?
- **Do you know how to acquire these resources?**
- **Do you have a good sense of what might go wrong?**
- **Do you have backup plans if Murphy strikes?**
- ***Have you gotten your hands dirty yet?***

Ubiquitous Data Access

15-821 / 18-843
Fall 2024

Mahadev Satyanarayanan
School of Computer Science
Carnegie Mellon University

Readings for Today

[Kistler1992]	Kistler, J.J., Satyanarayanan, M. ***Disconnected Operation in the Coda File System*** ACM Transactions on Computer Systems 10(1), February, 1992.
[Muthitacharoen2001]	Muthitacharoen, A., Chen, B., Mazieres, D. ***A Low-bandwidth Network File System*** In Proceedings of the 18th ACM Symposium on Operating Systems Principles. Chateau Lake Louise, Alberta, October, 2001.
[Lee2002b]	Lee, Y.W., Leung, K.S., Satyanarayanan, M. ***Operation Shipping for Mobile File Systems*** IEEE Transactions on Computers 51(12), December, 2002.
[Flinn2003]	Flinn, J., Sinnamohideen, S., Tolia, N., Satyanarayanan, M ***Data Staging on Untrusted Surrogates*** In Proceedings of FAST'03: 2nd USENIX Conference on File and Storage Technologies. San Francisco, CA, March, 2003
[Tolia2007]	Tolia, N., Satyanarayanan, M., Wolbach, A. ***Improving Mobile Database Access over Wide-area Networks without Degrading Consistency*** In MobiSys '07: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services. San Juan, Puerto Rico, 2007.
[Bai2016]	Bai, Y., Zhang, X., Zhang, Y. ***Improving Cloud Storage Usage Experience for Mobile Applications*** In Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems. Hong Kong, China, August, 2016

Roots of These Concepts

The earliest reading is from 1992 – 30+ years ago!

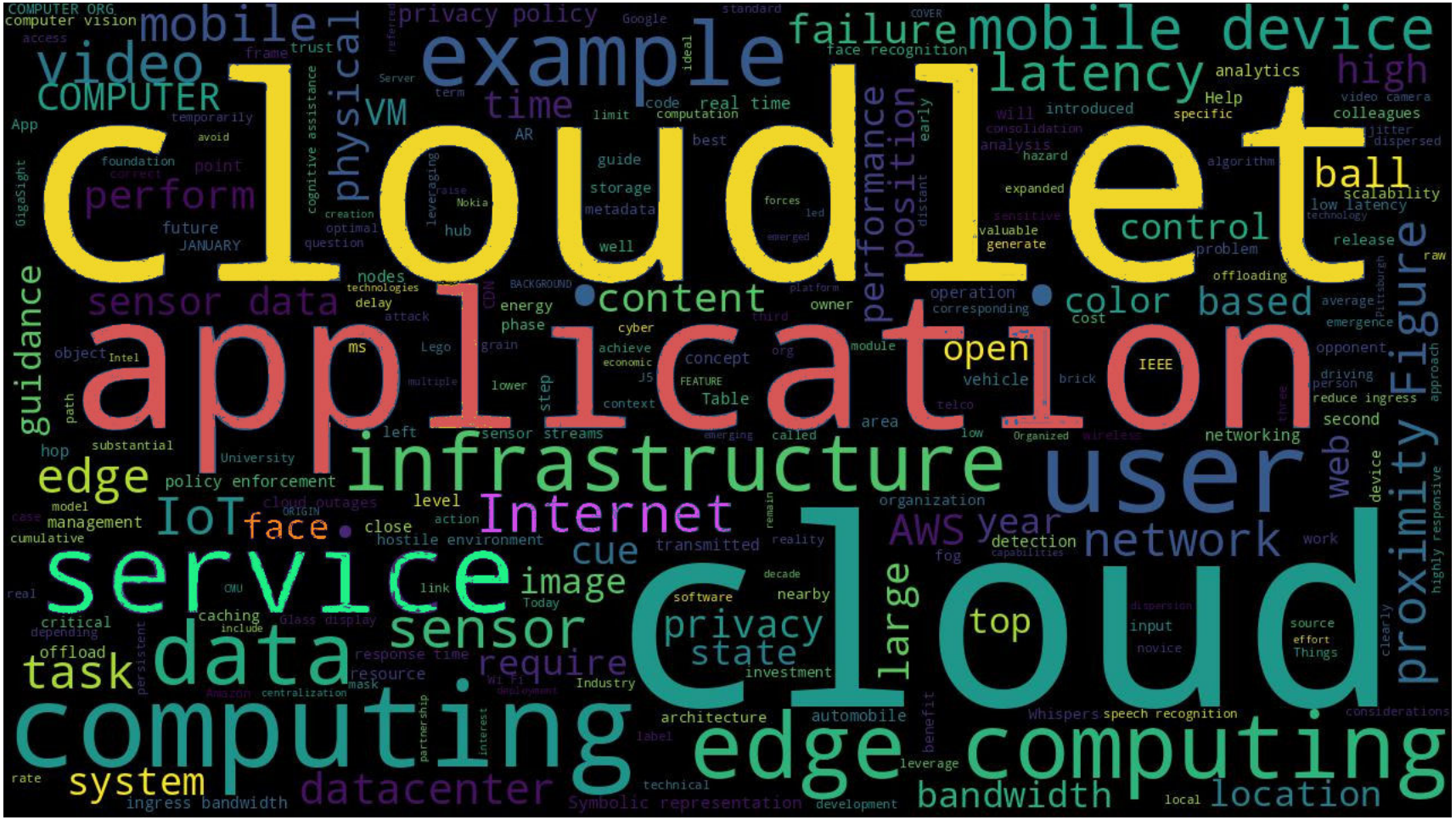
How could there possibly be anything useful to say back then for this course?

To understand how/why, we need to go back even earlier to the late 1970s

And then work our way forward

Let's start with today …

Bewildering Complexity of Cyberspace



déjà vu?

Bewildering complexity of physics in the late 19th Century

- **proliferation of entities (elements)**
- **vague similarities, weak patterns, imprecise understanding**
- **missing abstractions**
  we did not even know the difference between Atomic Weight & Atomic Number!

Nagging suspicion of a deep underlying order

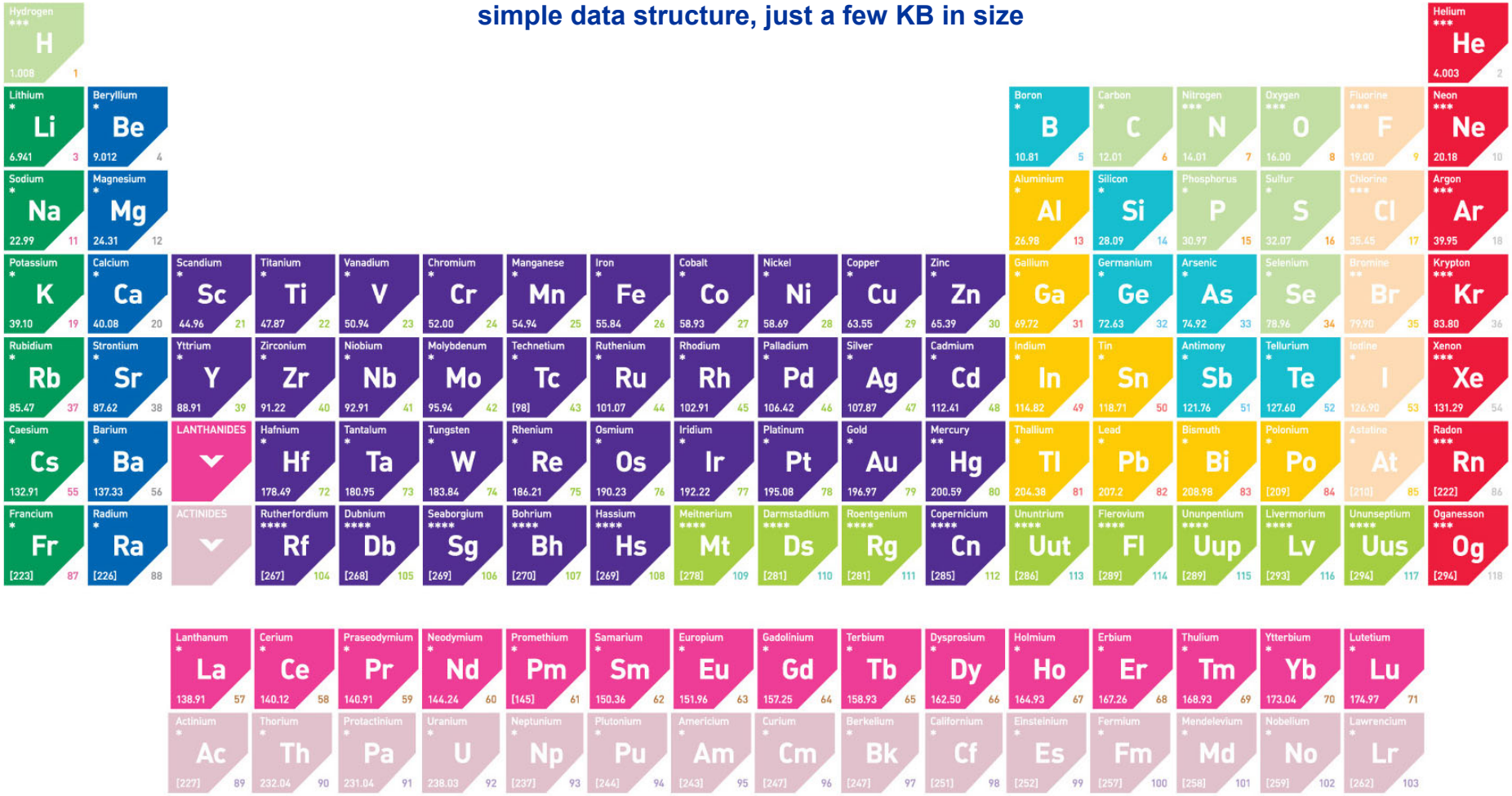
We developed a brilliant solution that reduced complexity

It has withstood the test of time (with significant evolution)

The Periodic Table

Amazing Distillation of Knowledge

simple data structure, just a few KB in size



Can We Do Something Similar for Computing?

Bring order to the explosive diversity that we see today

A tiered model of computing appears to have the right attributes

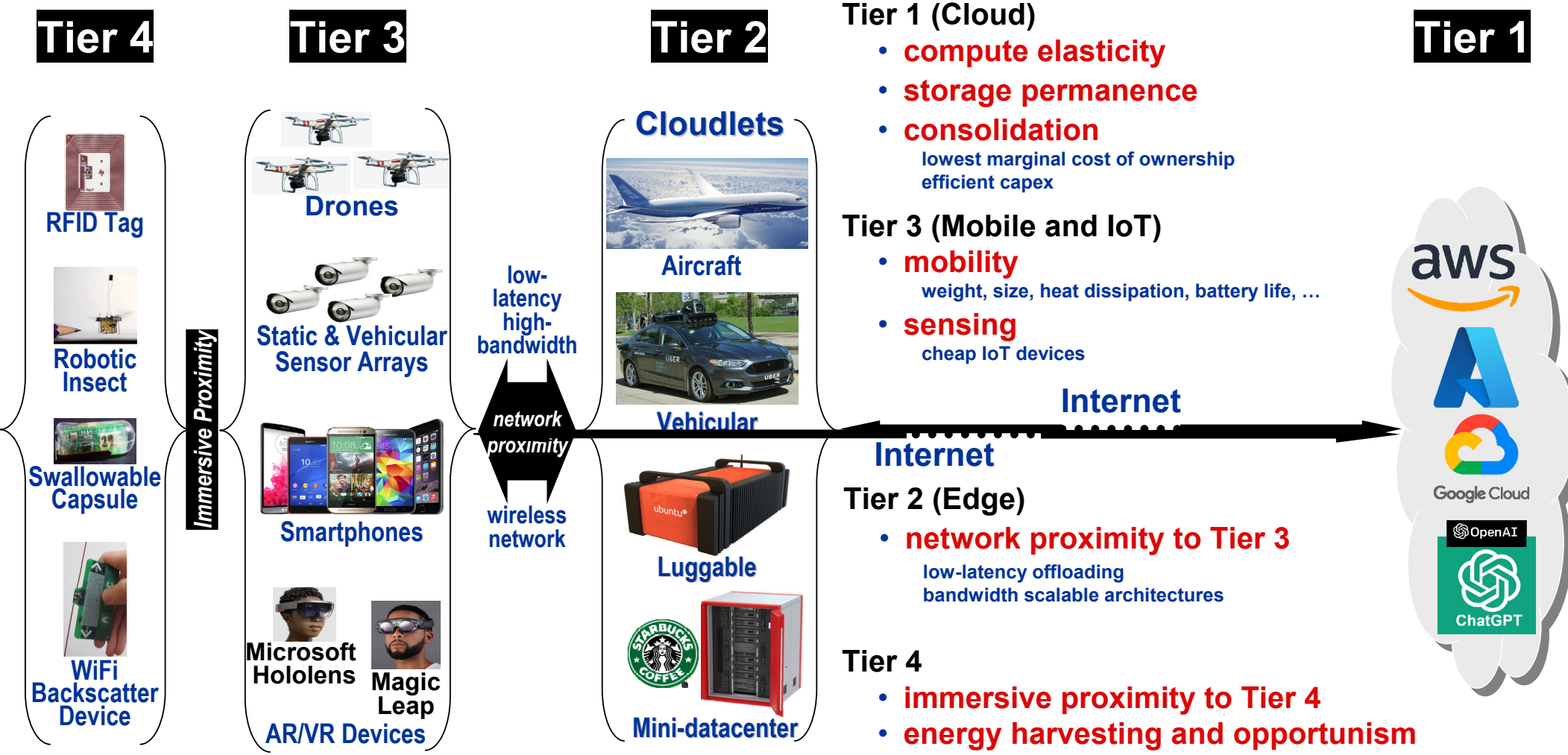
Each tier→ set of design constraints that dominate attention

Many alternative implementations of any given tier

- **but all are subject to the unique constraints of that tier**
- **essential similarity in spite of design freedom**

Major shifts in computing involve appearance, disappearance or repurposing of tiers

- **e.g., batch processing, timesharing, personal computing, mobile computing, IoT/pervasive computing, edge computing, …**
- **tier structure appears to be stable on the order of a decade or so**



Tier 1 (Cloud)

- **compute elasticity**
- **storage permanence**
- **consolidation**
  - lowest marginal cost of ownership
  - efficient capex

Tier 3 (Mobile and IoT)

- **mobility**
  - weight, size, heat dissipation, battery life, …
- **sensing**
  - cheap IoT devices

Tier 2 (Edge)

- **network proximity to Tier 3**
  - low-latency offloading
  - bandwidth scalable architectures

Tier 4

- **immersive proximity to Tier 4**
- **energy harvesting and opportunism**
  - "intermittent computing"
- **no chemical energy source**
  - zero maintenance long-term deployments

Optional Followup Reading

"The Computing Landscape of the 21st Century"

Satyanarayanan, M., Gao, W., Lucia, B.
In Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19)
Santa Cruz, CA, February 2019

A Tiered View of the Past

In the beginning …

Remote Job Entry

Only Tier 1 existed

No other tier could exist as long as computing hardware was big & expensive

Batch Processing (1950-60s)

No networking, but remote job entry and telemetry existed using point-to-point links

Telemetry

Mainframe

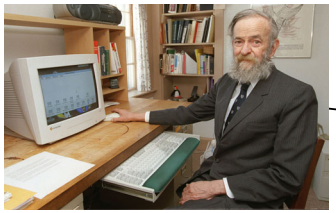
Improving Elasticity: Timesharing

Mainstream by 1970s

Batch → serial resuse

Timesharing → elasticity, as long as below saturation

Leverage statistical multiplexing of resource demands by different users



Telemetry

Mainframe

Low Latency: Personal Computing

the emergence of Tier 2

Queueing delays of shared mainframe became intolerable at high levels of sharing

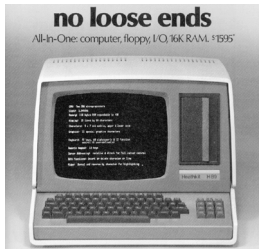
Lampson & Thacker: why bother sharing?

Disaggregated mainframe → large collection of personal computers

Tier 1 completely replaced by Tier 2

Positive consequence: fine-grain improvement of user experience via small investments

(not quite replacing capex with opex, but close)



Need for "the cloud" Emerges

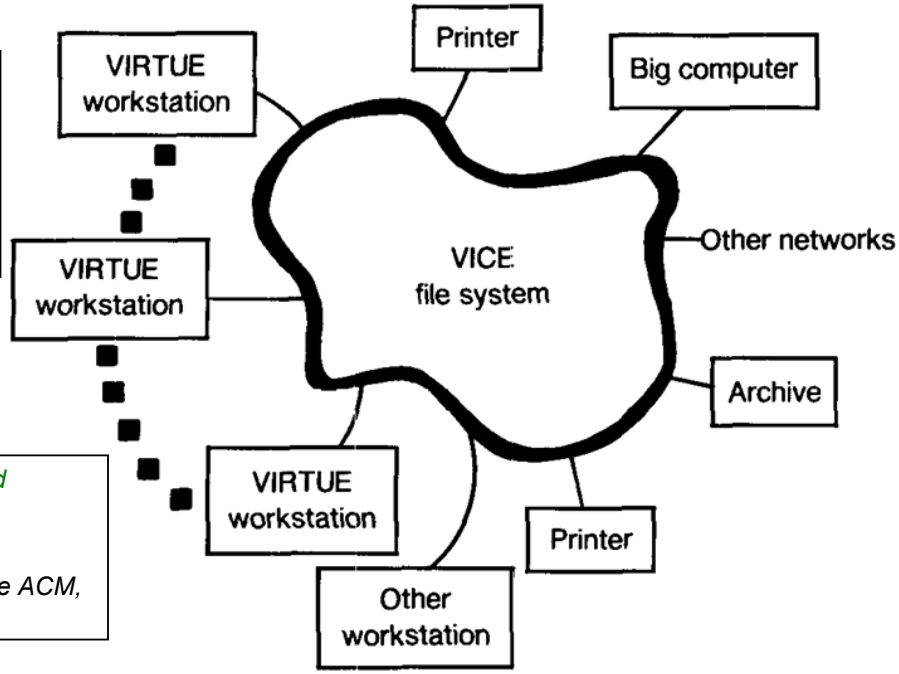
Enterprise data sharing became easy in timesharing era

- **Single shared file system**
- **Data permanence becomes crucial**
- **Ease of information sharing improved productivity**
- **Tight access control**
- **"Single System Image"**

This vanished with disaggregation

Can easy data sharing of timesharing file systems be combined with new benefits of personal computing?

Solution (AFS):
store in the cloud and cache at the edge



"Andrew: A Distributed Personal Computing Environment"

Communications of the ACM, April 1986

User mobility is supported: A user can walk to any workstation in the system and access any file in the shared name space. A user's workstation is personal only in the sense that he owns it.

System administration is easier: Operations staff can focus on the relatively small number of servers, ignoring the more numerous and physically dispersed clients. Adding a new workstation involves merely connecting it to the network and assigning it an address.

Mobile Data Access in a Cloud/Edge World

Tension between *centralization* and decentralization (autonomy versus interdependence)

Mobile computing

- **all about *freedom* and lack of constraints**
  - **"anything, anytime, anywhere"**
- **consequence: bewildering complexity**
  - ***Which device has my data? Where is the latest version?***
  - ***How did this update conflict happen?***
  - ***The network is broken, I can't reach the cloud***

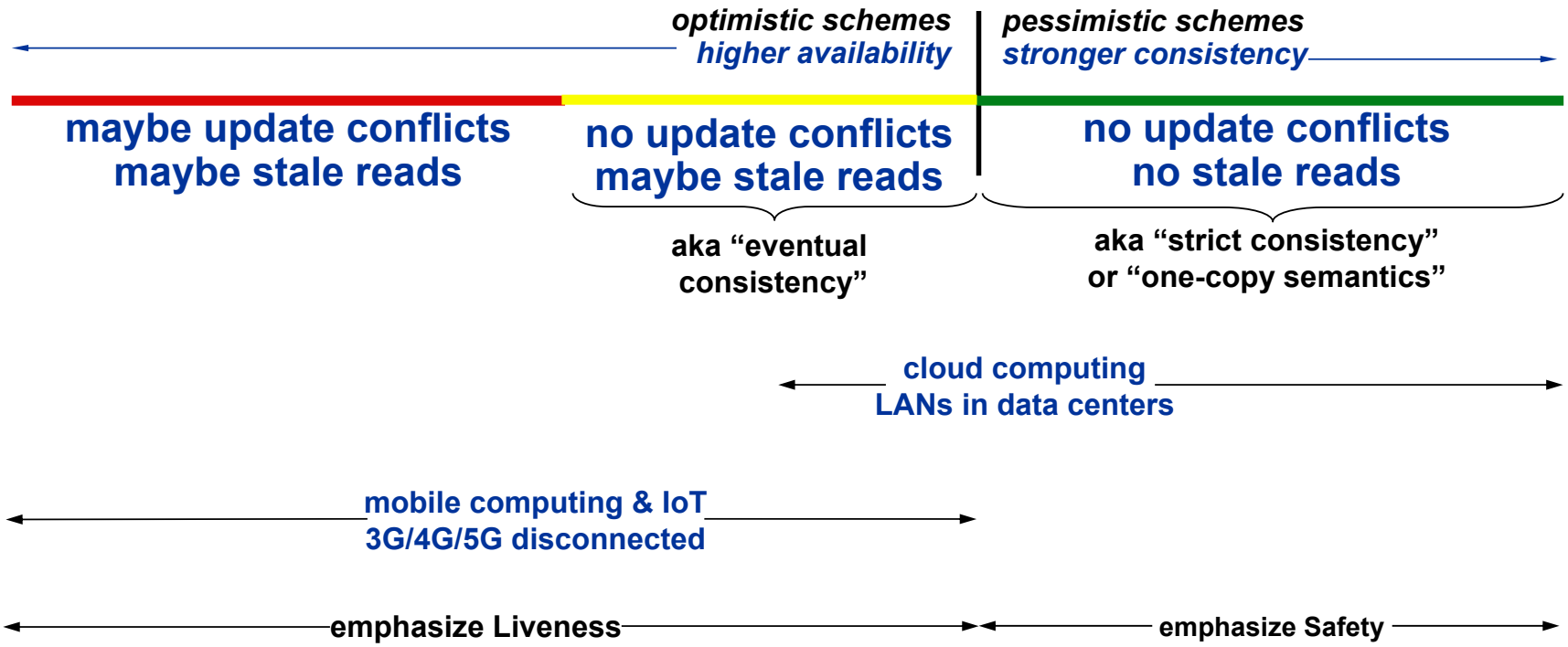
Cloud computing

- **all about *simplicity* through centralization**
  - **"like flipping a switch or turning on a faucet"**
- **reduced user complexity, reduced TCO**
- **just pay bill at the end of the month**

How to reconcile these opposing forces?

Disconnected Operation in the Coda File System

Replica Control Strategies



Disconnected Operation

Key enabling technology for mobile computing

- masks *temporary isolation* from file servers
- exploits *caching* for availability
- effectively zero bandwidth & infinite latency
- uniform handling of *voluntary* and *involuntary* disconnections
- applications/users are unaware that they are disconnected
- simplifies creation of failure-resilient applications

Conceived and first demonstrated in Coda File System

Benefits

- worst-case fallback position for connectivity
- *valuable even when communication feasible*
  - lowers cost & power for communication
  - allows radio silence in military applications

Caching Functions

Venus cache performs 3 functions for disconnected operation

- ***hoarding*** **user-augmented LRU caching**
- ***emulation*** **of services while disconnected**
- ***reintegration*** **resyncing and resolving conflicts**

User-Augmented Cache Management

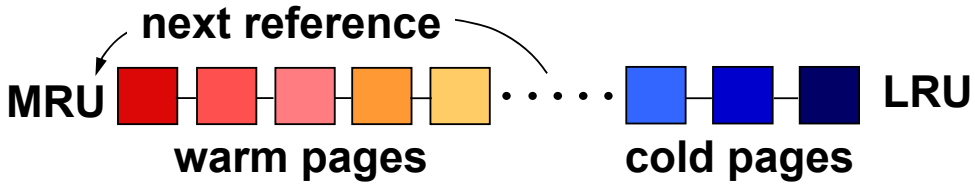
Each cached object, *f*, has a *current priority, p(f)*

Composed of two parts

- *hoard priority, h(f)* static part (user-specified)
- *reference priority, r(f)* dynamic part (observed)

Weight of hoard priority is α: $p(f) = \alpha\, h(f) + (1 - \alpha)\, r(f)$

default value of α is 0.75
(value of zero gives classic LRU)



Temperature is a good metaphor for object priority

- *object never cools below hoard priority*
- it can heat up substantially, depending on reference priority

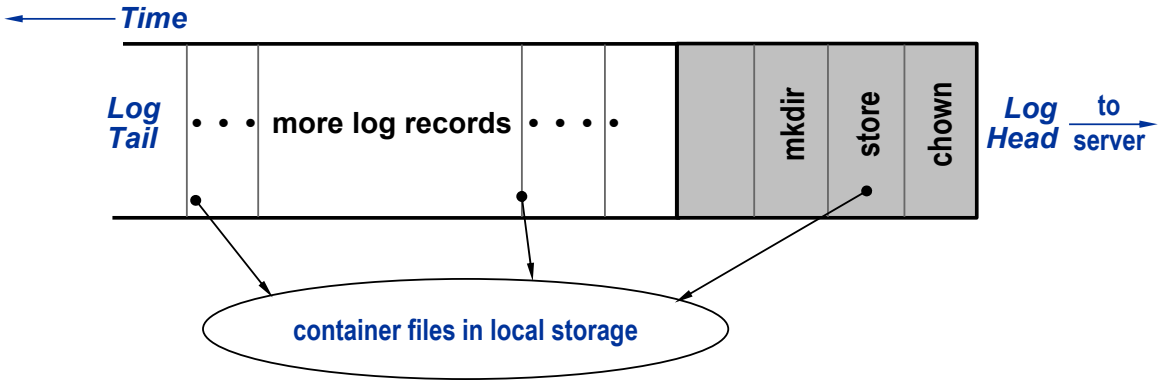
Emulation

Updates have to be buffered until connectivity restored

- **disconnection may be minutes, hours, or possibly days**
- ***persistent cache absolutely critical***

Venus records mutations in per-volume *client modification log (CML)*

- **temporal log of all updates, one log record per operation**
- **log is kept in persistent transactional memory (RVM)**
  simplifies correctness in the face of failures
- **store records point to *container file* in local file cache**



Log Optimizations

Shrinking of log whenever possible

- performed when logging each disconnected update
- log kept at shortest possible length at all times
- exploits semantics of file system operations

Exploits ***temporal locality of updates***

- ***"if an object is modified, it will be modified again soon"***
- example: rm -rf <some directory>
- example: cp -pr <sourcedir> <targetdir>
- example: edit-debug cycle, editor checkpoint file

Key to long-term disconnected operation

- ***reduces cache space usage*** (especially container file space)
- ***speeds reintegration*** when connectivity is restored

Reintegration

Reintegration ≈ cache resync + CML replay

- deferred until authentication tokens available
- temporary FIDs replaced by permanent (if necessary)
- CML is applied first; file data is backfetched
- transparent unless conflicts are detected

Implementation uses *RVM* for fault tolerance

- greatly simplify cleanup
- wireless networks can be very flaky!

Coda approach to conflict handling

- *syntactic detection* (fast path, version check)
- *semantic resolution* (slow but sure, type-specific)
  "conflict" is in the eye of the beholder (e.g. checkbooks, calendars)

Syntactic vs. Semantic Consistency

Entire responsibility for detection rests with client

Syntactic **definition of consistency based only on interleaving**

- **same as "one-copy semantics"**
- **no attempt to use semantics of data, replicas are black boxes**
- **restrictive but cheap**
  (no application-specific code execution)

Semantic **definition of consistency takes full account of data semantics**

- **more liberal, allows may interleavings disallowed by syntactic defn**
- **usually more expensive (involves app-specific code execution)**

Coda approach

- ***syntactic → normal case***
- ***semantic → exception handling***

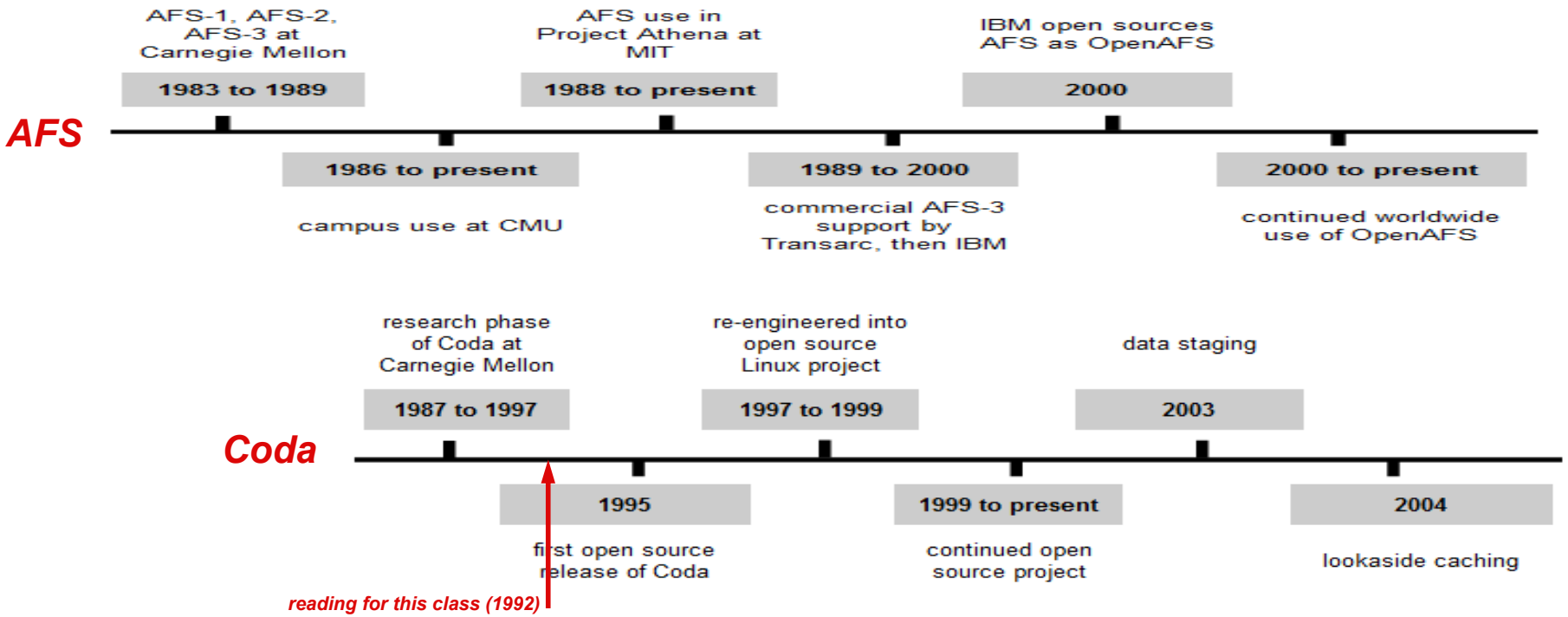
Contrast with systems like Bayou, that always use semantics

Key Insights from AFS & Coda

A distributed file system that uses *on-demand caching* is

1. **an excellent user abstraction for cloud-mobile convergence**
2. **it extends the familiar hierarchical name space to the cloud**
3. **all local applications work seamlessly on cloud data**
   - **no manual steps to download, upload, etc.**
   - **chances for human error**

Success of DropBox validates this in the marketplace today



1999 lustre™

2007 Dropbox

2010 Maginatics

Peter Braam

http://lustre.org/about/

Worked on Coda at CMU 1997-1999
Founded Cluster File Systems 1999
Sold to Sun in 2007
Open sourced by Oracle 2010
Worldwide use in Supercomputers (HPC)

Drew Houston

Andrew W. "Drew" Houston is an American internet entrepreneur and is best known for being the founder and CEO of Dropbox, an online backup and storage service. According to Forbes magazine, he has a net worth of $400 million US dollars. Wikipedia

Born: March 10, 1983 (age 29), Acton

Education: Massachusetts Institute of Technology

en.wikipedia.org

Arash Ferdowsi

Arash Ferdowsi is an Iranian-American entrepreneur. He is co-founder and chief technology officer of Dropbox. Wikipedia

Born: October 7, 1985 (age 26), United States of America

Education: Massachusetts Institute of Technology

forbes.com

Inspiration for Dropbox from AFS and Coda
Wired Magazine, December 2011

Used AFS in Project Athena at MIT in 2000s
Founded DropBox in 2007

Dropbox

Jay Kistler

1993 CMU PhD on Coda
Founded Maginatics 2010
Purchased by EMC in 2014
EMC acquired by Dell in 2016

How Do You Know What to Cache?

Approach 1: ***Full Replication***

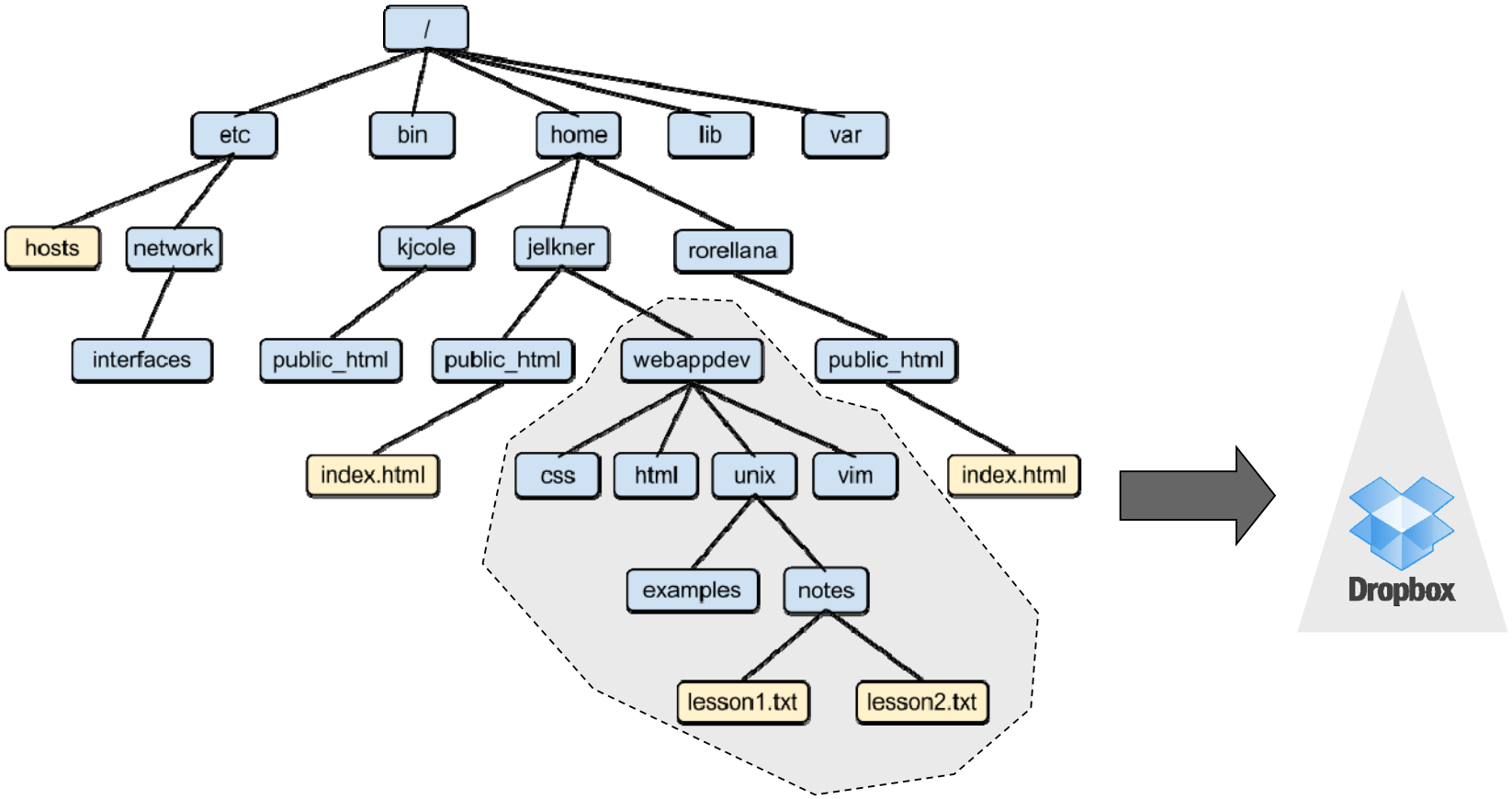
Used by DropBox and other similar services

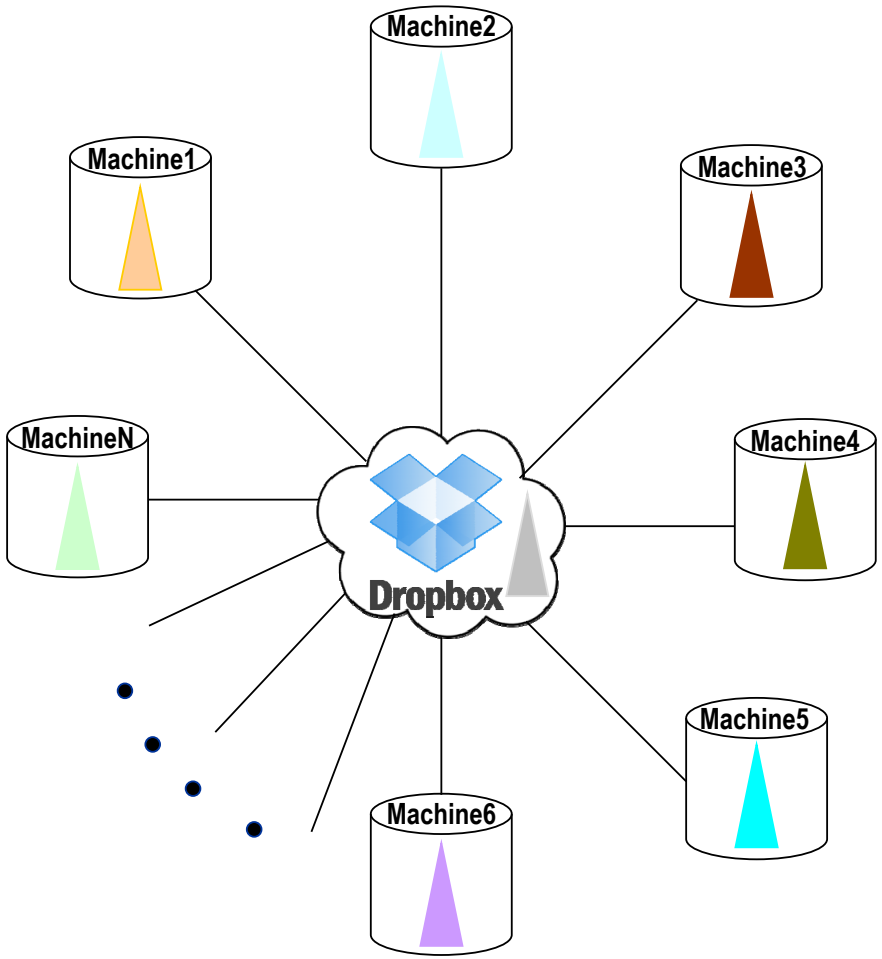
Designate a subtree as backed by DropBox

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 to point of use

Place Entire Subtree in DropBox





Shortcomings of DropBox Approach

(all such solutions are known as "sync solutions" in industry jargon)

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"

Dropbox shares soared today in biggest tech IPO since Snapchat

DBX stock jumped 36 percent on its NASDAQ debut

By Nick Statt | @nickstatt | Mar 23, 2018, 4:44pm EDT

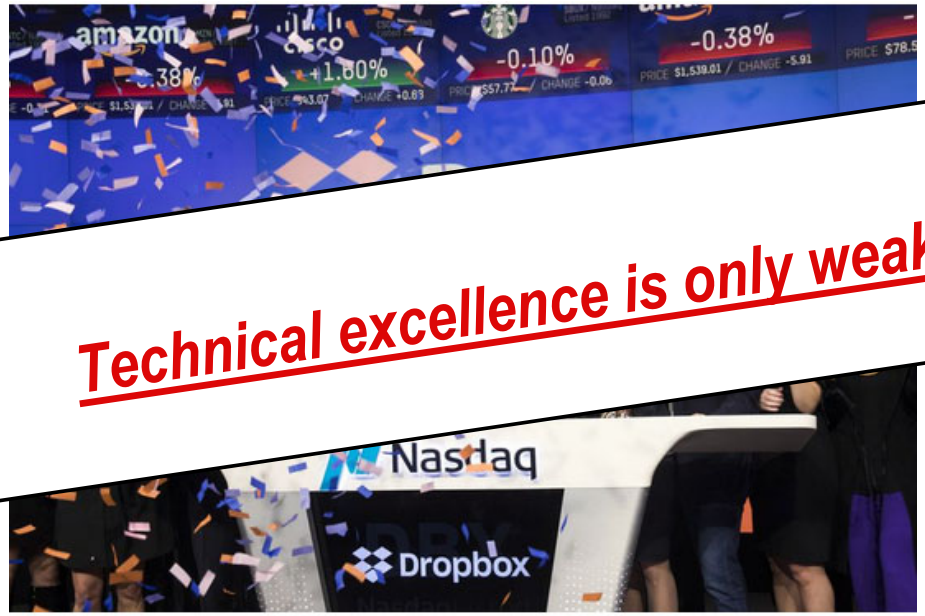


Photo by Drew Angerer/Getty Images

21.17 USD

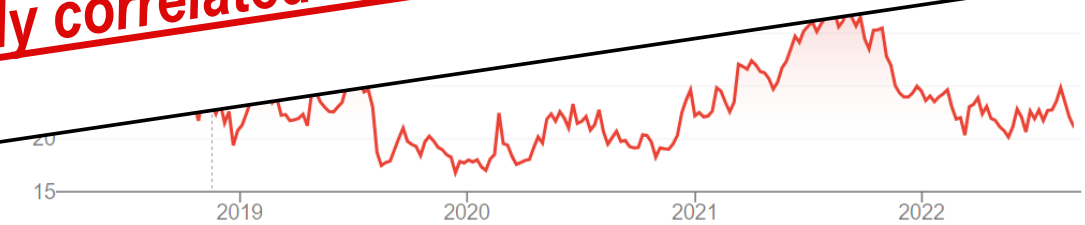
-7.31 (-25.67%) ↓ all time

Closed: Sep 2, 6:43 PM EDT • Disclaimer

After hours 21.05 −0.12 (0.57%)

+ Follow

1D 5D 1M



Cloud storage and collaboration company Dropbox — which started 10 years ago as a small startup in the San Francisco-based Y Combinator incubator program — went public today, and its shares were up nearly 36 percent as of market close this afternoon. The successful performance makes Dropbox the biggest tech IPO since

Technical excellence is only weakly correlated with business success

A Much Better Approach

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

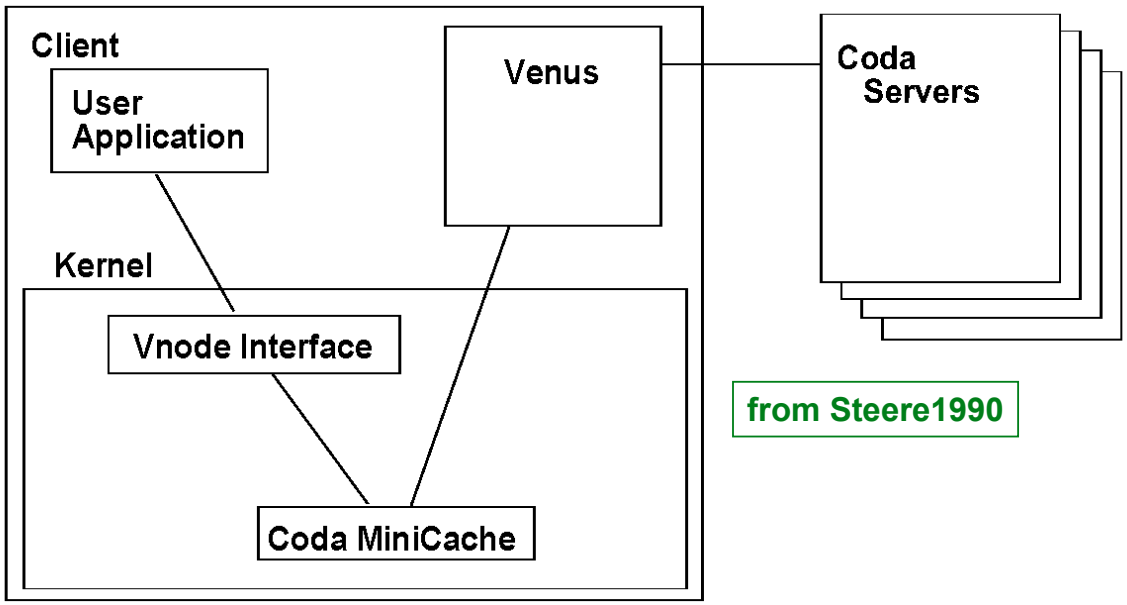
- **approach used in AFS** **(inherited and extended from AFS-2 by Coda File System)**
- ***requires integration with the operating system***
- **fine-grained and selective approach to data management**

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

Inspiration for FUSE interface in Linux today

- requires operating system modifications

+ *total application transparency*

+ enable demand caching

Price of Ignoring These Lessons

Bai, Y., Zhang, X., Zhang, Y.
"Improving Cloud Storage Usage Experience for Mobile Applications"
In Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems. Hong Kong, China, August, 2016.

"However, today's commercial services for mobile access to cloud storage have ignored some useful insights and practical experience of this multi-decade research. ***Most notably, they choose to avoid client side OS-level monitoring and support, in exchange for fast and easy service deployment.*** *As the result of this implementation and deployment strategy, many of the existing mobile apps fall short of using cloud storage service efficiently, and thus leading to poor usage experience, such as unnecessary energy consumption, extended folder synchronization time, and redundant network transmission traffic. Here we summarize our findings as follows."*

Why Did DropBox Go Retro?

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

- DropBox was created ≈ 2007
- AFS/Coda approach assumes sophisticated OS
  - Unix adequate even as early as 1984
  - Windows was DOS-like internally until XP (2001)
- DOS / Windows had ~90% client marketshare

Founders of DropBox used AFS extensively at MIT

- AFS part of Athena at MIT since ~1987
- pain: AFS not accessible after graduation
  - created DropBox to address this pain

DropBox approach simplifes OS portability

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

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

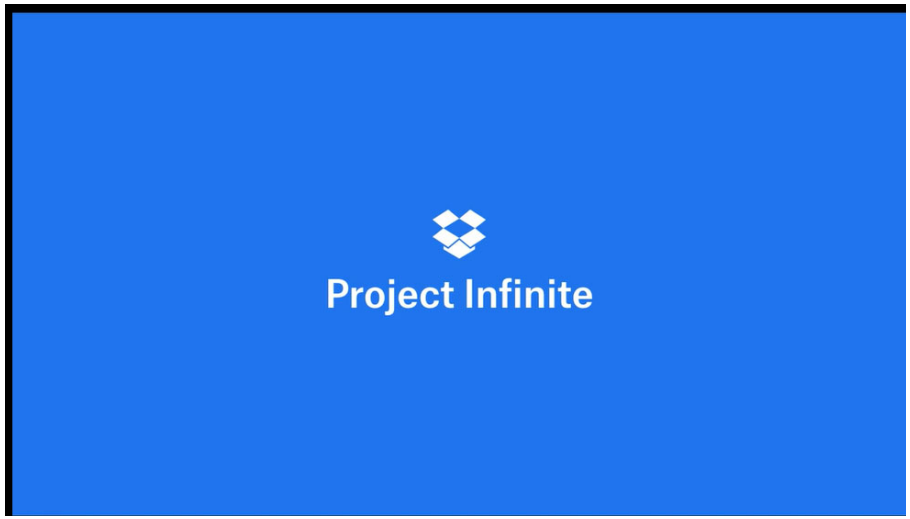
BY RACHEL SWABY 12.22.11 | 6:30 AM | PERMALINK

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

DropBox is essentially AFS−−

- 2011 Wired Magazine article
- on Wayback Machine, see http://www.wired.com/2011/12/backdrop-dropbox/all/

But Even DropBox Dreams of Caching



April 2016 video from DropBox

Never released as a product!

Multi-OS On-Demand Caching

It is possible, but takes enormous technical skill

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

- 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)

- EMC purchased by Dell for $67B in October 2015

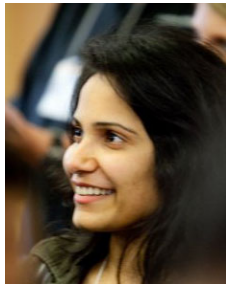
Very Strong CMU Roots!

(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

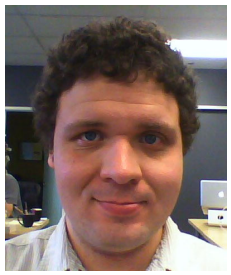
Konteya Joshi
MS-SE 2006

Vaibhav Kamra
BS-ECE 2003
MS-ECE 2004

Akshay Moghe
MS-ECE 2008

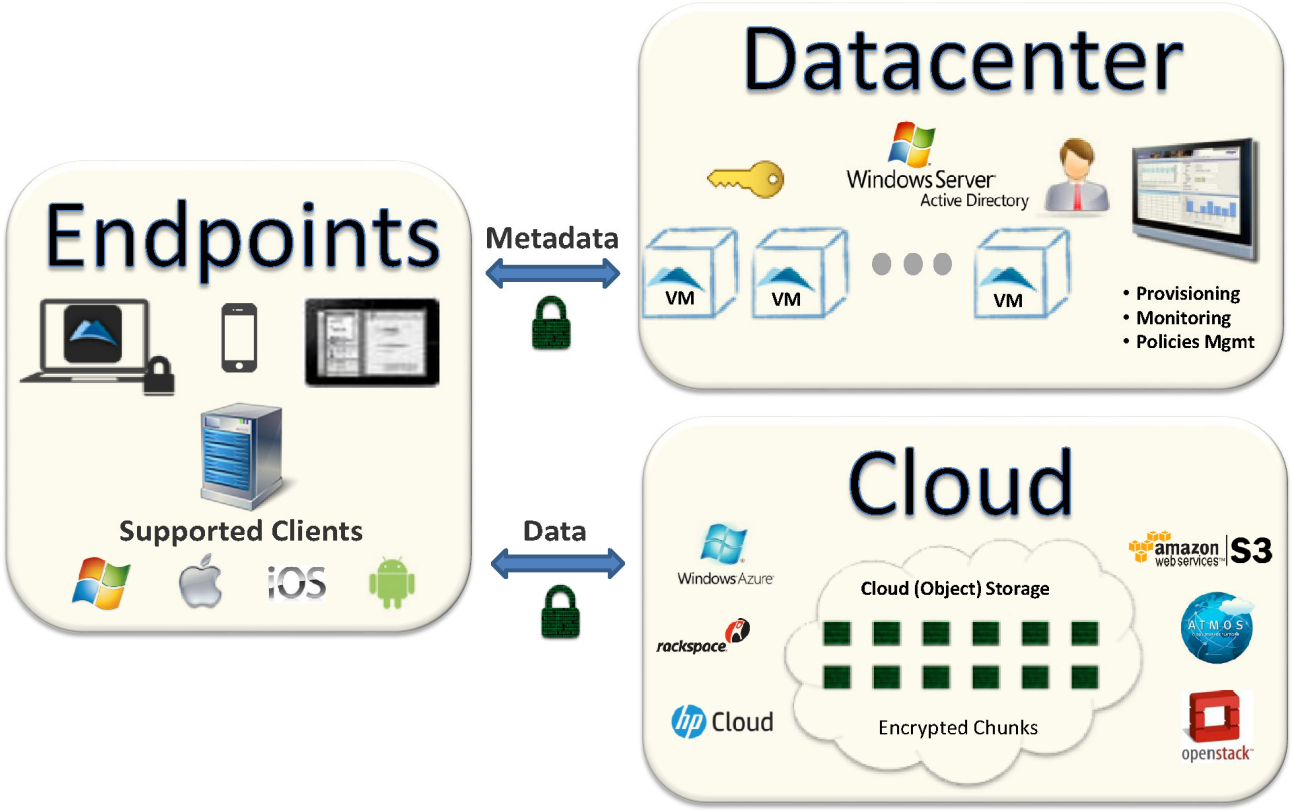
Vijay Panghal
MS-INI 2009

Vibhav Sreekanti
BS-CS 2009



Mark Schreiber
BS-CS 2003

MagFS Architecture



What Happened to MagFS?

Maginatics acquired by EMC in 2014

- **MagFS targeted at an EMC security product (unannounced)**

EMC acquired by Dell in 2015

- **EMC security product killed by Dell**
- **MagFS killed with it**
- **Development frozen as of 2015**
  all stock vested for MagFS team, so they all left Dell

MagFS code base gathering dust somewhere inside Dell

No incentive for Dell to open source MagFS

Sad story of a huge amount of innovation

Are Classic File Systems Dead?

Hot Topic Today

the death watch has begun

ACM DL DIGITAL LIBRARY

Hierarchical file systems are dead

Authors: Margo Seltzer Harvard School of Engineering and Applied Sciences
Nicholas Murphy Harvard School of Engineering and Applied Sciences

Published in:
· Proceeding
HotOS'09 Proceedings of the 12th conference on Hot topics in oper
systems
Pages 1-1
USENIX Association Berkeley, CA, USA ©2009
table of contents

Every Page is Page One

Readers can enter anywhere. Is your content ready to receive them?

Home | Contact | About | The Book | Publications | Speaking | Examples of EPPO topics

The Death of Hierarchy

by Mark Baker on 2014/09/29 in Every Page is Page One, Hide and Seek, Writer vs. Reader

Hierarchy as a form of content organization is dying. A major milestone — I want to say tombstone — in its demise is the shutdown of the Yahoo directory, which will occur at the end of the year according to an article in Ars Technica, Yahoo killing off Yahoo after 20 years of hierarchical organization. (Actually it seems to be offline already.)

Take the Every Page is Page One Course!

Learn to write in the Every Page is Page One style with a two-day course customized for your group's needs and using your own material for examples and exercises. Contact us for more information.

Get the Book!

The Cloud And the Death of the File System

Posted on April 2, 2014

One of the things I neglected to discuss in my eBook, Web Development in the Cloud, was something that seemed so obvious to me that I simply missed including it. And that is the simple fact that if you develop web sites on the Cloud, you need to understand that the conventional file system process is dead.

Appears True at High Level

E.g. Android software focuses on Java classes and SQLite

- Android users never see a classic file system
- But, underneath Android, is the Linux native environment
- classic hierarchical file system continues to live on

This model may indeed become common

Will the lower layer vanish completely some day?

Not a New Viewpoint!

The Death of File Systems

by **JAKOB NIELSEN** on February 1, 1996

Topics: Human Computer Interaction Predictions & Milestones Technology

Summary: The file system has been a trusted part of most computers for many years, and will likely continue as such in operating systems for many more. However, several emerging trends in user interfaces indicate that the basic file-system model is inadequate to fully satisfy the needs of new users, despite the flexibility of the underlying code and data structures.

Originally published as: 145. Nielsen, J. (1996). The impending demise of file systems. IEEE Software 13, 2 (March).

Why are File Systems Hierarchical?

Ken Thompson made radical changes in creating Unix

- **why was the Unix file system so conventional and hierarchical?**
- **mere sentiment? lack of imagination?**

"The Architecture of Complexity"
Herbert A. Simon, *Proceedings of the American Philosophical Society*, Vol. 106, No. 6., Dec. 12, 1962, pp. 467-482.

"Empirically, a large proportion of the complex systems we observe in nature exhibit hierarchic structure. On theoretical grounds we could expect complex systems to be hierarchies in a world in which complexity had to evolve from simplicity. In their dynamics, hierarchies have a property, ***near-decomposability,*** *that greatly simplifies their behavior."*

Near-Decomposability

Key property of human-created hierarchical systems (Simon 1962)

Consequence of human cognitive limitations

Allows focus on immediate neighborhood (current directory + children)

- *apparent shrinking of scale*
- valuable to exploit in achieving scalability
- exploitable in concurrency control, failure resiliency, consistency, etc.

Hierarchical file systems reflect the limitations of human cognition

- without external tools, that's the best organization for human minds
- "external tools": e.g., SQL databases and search engines

How Hierarchy Helps

Hierarchical file systems conflate search and access

- well-matched to limitations of human cognition,
- locality is an emergent property (temporal and spatial)
- locality is precious performance-wise for direct human exploration of data

Retrospective use of old unstructured data (e.g., decades later) →

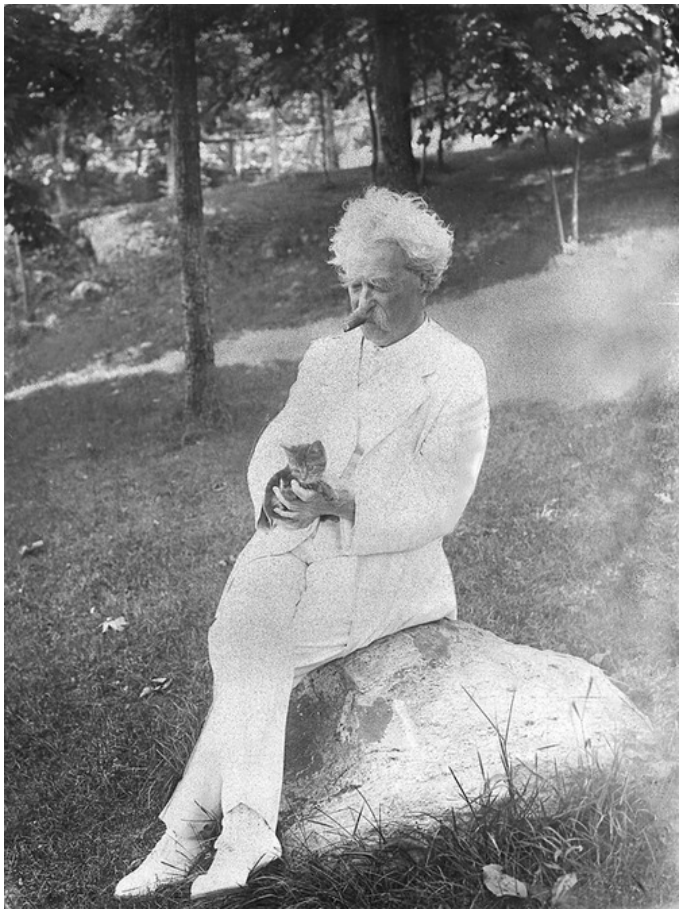
- even the features for indexing may be unclear
- manual exploration may be necessary

Need for manual exploration (even if rare) →

- hierarchical file systems will not disappear
- but the hierarchical nature may remain deeply buried

The Death of File Systems?

"…report of my death was an exaggeration"



The report of my illness grew out of his illness, the report of my death was an exaggeration.

Mark Twain

Coda Cartoons
from
Nikkei Electronics, September 1990
translation of IEEE Computer May 1990 article

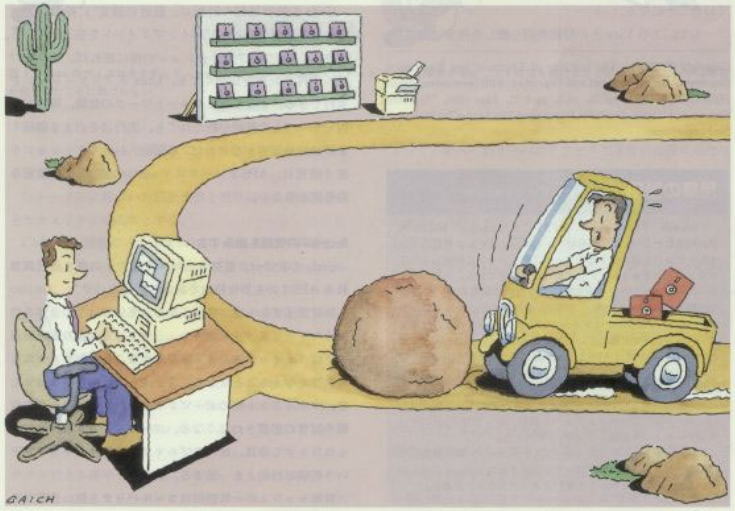
論文

ネットワーク障害を考慮した Coda ファイル・システム

Andrew File System の後継分散ファイル・システム

Mahadev Satyanarayanan
＊Carnegie Mellon University

Coda は，Andrew File System（AFS）をベースにして設計した Unix 準拠の分散ファイル・システムである。AFS に比べてアベイラビリティを高めた。クライアントのローカル・ディスクだけを使って処理するメカニズムを組み込んだ。ポータブルなコンピュータでも分散ファイル・システムを使うことができる。ネットワークの故障などが原因でファイル・サーバとアクセス不能になっても処理を続けることができる。本号では前号で掲載した AFS の論文の後半部分を掲載する。（本誌による要約）



AFS and Coda Difference

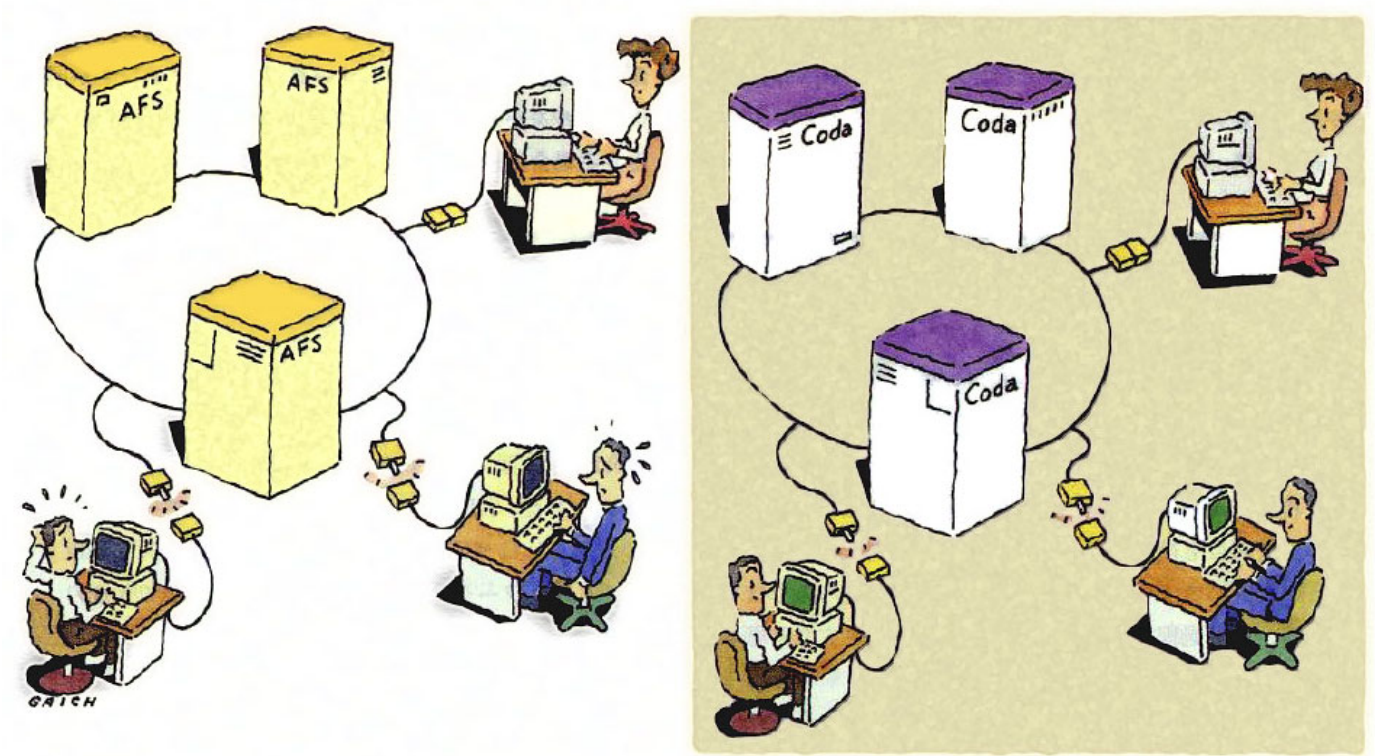


図 1　Coda のディスコネクテッド・オペレーション　Coda ではネットワークに障害が発生しても処理を続けることができる。ユーザは障害が発生したことに気づかない。

Impact of Hoarding on Cache Size

図５　Stickyファイルは最後までキャッシュにとどまる　Codaではキャッシュするファイルやディレクトリに優先度をつけることができる。高い優先度を備えたファイルやディレクトリほどローカル・ディスクにとどまっている確率が高い。一番高い優先度をStickyという。

Trust Model

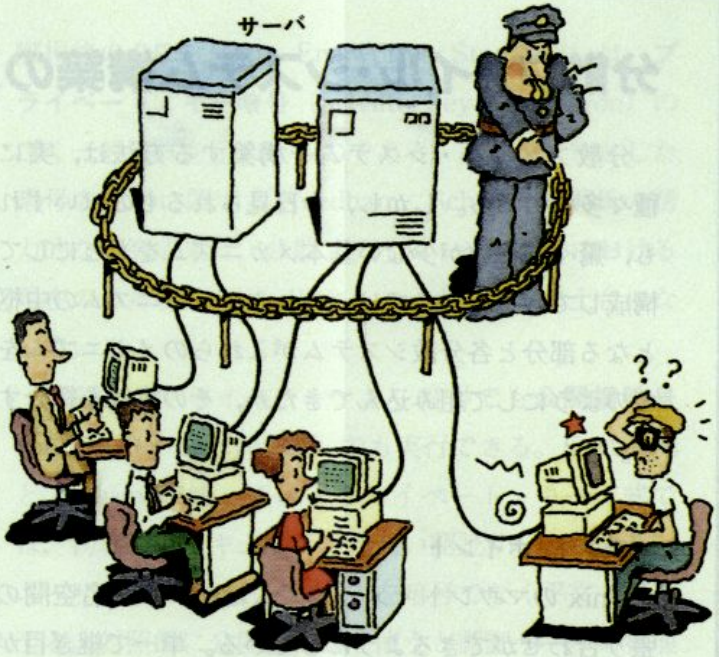


図６　信頼性の高いサーバがワークステーションを管理する
Andrew File System や Coda では，サーバ側に信頼性の高い枠組みを備えて運営する。ワークステーション側にはセキュリティのための機構はない。この方がユーザやワークステーションの数が増えてもシステムの管理・運営が容易である。