Memory and File Systems

SOSP-25 Retrospective

Mahadev Satyanarayanan

School of Computer Science

Carnegie Mellon University

Four Drivers of Progress

The quest for ***scale*** — *from early 1950s*

The quest for ***speed*** — *from early 1950s*

The quest for ***transparency*** — *from early-1960s*

The quest for ***robustness*** — *from mid- to late-1960s*

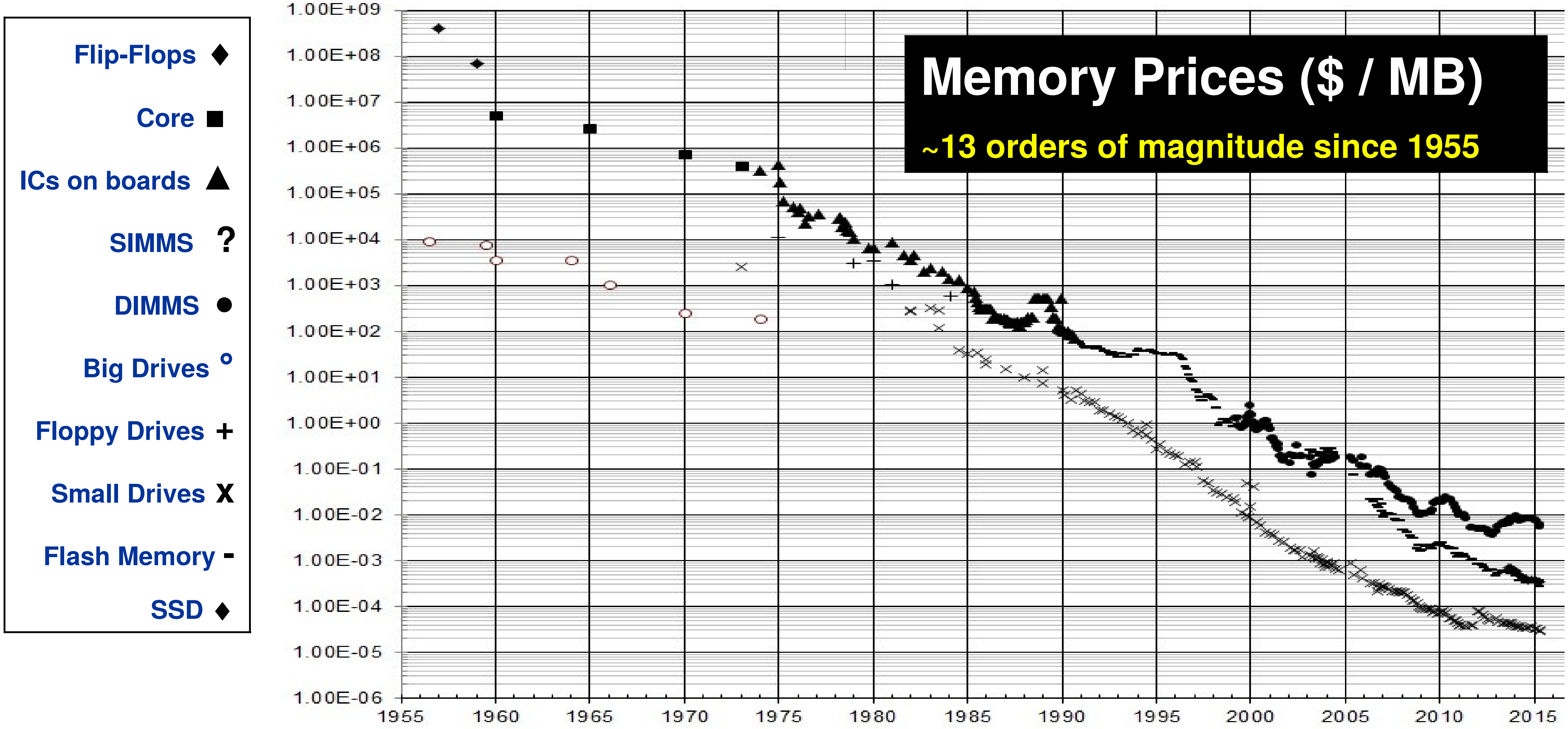
(both system and human errors)

Complex Interactions

The Quest for Scale

Cost of Memory & Storage

(Source: John C. MacCallum http://jcmit.com)



Naming and Addressability

Consistently too few bits in addressing (12-bit, 16-bit, 18-bit, 32-bit, …)

re-learned in DOS/ Win3.1 (memory extenders); hopefully 64 bits will last us a while

Semantic addressing

hierarchical name spaces, SQL, search engines

Content Addressable Storage (aka deduplication)

Venti (late 1990s), LBFS (early 2000s), many others since,

continuing concerns regarding collisions (Val Henson)

Capability-based

- short term (seconds, minutes, hours lifetime)

  can be viewed as a form of caching expensive/cumbersome access checks

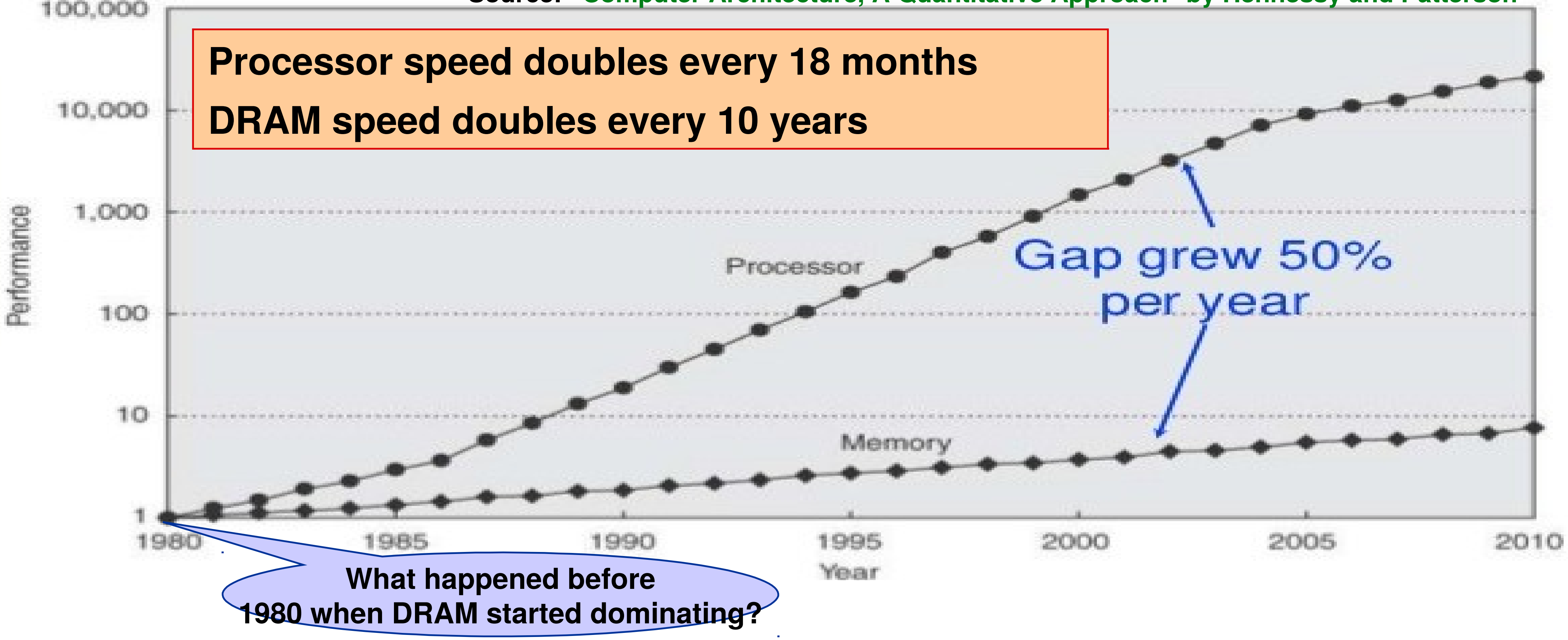
- long term (infinite life)

  Hydra on C.mmp (mid 1970s) pushed this concept to the limit
  Intel iAPX 432 (3 papers in SOSP 1981!)

The Quest for Speed

Processor-Memory Speed Gap

Source: "Computer Architecture, A Quantitative Approach" by Hennessy and Patterson



Before 1980

IBM System/360

Source: Wikipedia

Model	Shipped	Scientific Performance (KIPS)	Commercial Performance (KIPS)	CPU Bandwidth (MB/s)	Memory Bandwidth (MB/s)	Memory Size (KB)
30	1965	10.2	29.0	1.3	0.7	8–64
40	1965	40.0	75.0	3.2	0.8	16–256
50	1965	133.0	169.0	8.0	2.0	64–512
65	1965	563.0	567.0	40.0	21.0	128–1024
75	1966	940.0	670.0	41.0	43.0	256–1024
91	1967	1900.0	1800.0	133.0	164.0	1024–4096

IBM System/370

Source: Wikipedia

Model	Shipped	Processor Cycle Time	Memory Access Time	Memory Size (KB)
155	1971	115 ns	2 ms	256–2048
165	1971	80 ns	2 ms	512–3072

Creating an Illusion of Scale *and* Speed

Memory hierarchies

- scale appears to be that of slower but more scalable technology
- speed appears to be that of faster but less scalable technology
- essentially probabilistic in character (worst case can be bad)

Working set **characterizes the goodness of fit**

Exploiting parallel data paths for increased bandwidth

- striping
- sharding
- bit-torrent, etc

Managing Data Across Levels

LRU and variants work amazingly well!

Alas, a few workloads defeat LRU

- ***purely sequential access*** → **zero temporal locality**
  caching cannot help at all; only adds overhead
- ***purely random access*** → **being smart is useless**
  ratio of cache size to total data size is all that matters
- **these access patterns are observed in the real world**
  file scans in data mining, video/audio playback, hash-based data structures, …

Multi-decade quest for improvement over LRU for these workloads

ARC: adaptive replacement cache (Megiddo & Modha 2003) best so far

The Quest for Transparency

Transparency

"Indistinguishable from original abstraction"

- *no application changes:* programs behave as expected
- *no unpleasant surprises for users:* good user experience
- importance increases as hardware to human cost ratio shifts

Hugely important in industry, less important in academic research

Achieved by interposing new functionality at widely-used interfaces

- memory abstraction (hardware caches)
- POSIX distributed file systems
- x86 virtual machines

$\forall$…

Some Transparency Landmarks

Caching (not overlays or other software-visible abstractions)

- *consistency of distributed caches*
- strict consistency vs. weak / eventual consistency

Shared memory in multiprocessor systems

- *UMA:* "uniform memory access" (e.g. C.mmp and many others)
- *NUMA:* "non-uniform memory access" (e.g. Cm* and many others)
- *NORMA:* "no remote memory access" (Berkeley NOW project, and others)

Distributed Shared Memory

- hot topic in 1990s; long dormant
- it is coming back! (OSDI 2012: COMET)

A Brief History of Caching

Demand paging was first known use of caching idea *(1961)*

John Fotheringham, CACM, 1961, pp 435-436

Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store*

John Fotheringham

Ferranti Electric, Inc., Plainview, New York

1. Introduction

This paper is concerned with the method of address interpretation in the Atlas computer. The Atlas has been designed and built as a joint exercise between the Computer Departments of Manchester University and of Ferranti Ltd., and the ideas and concepts described in this

characters within a word for certain special functions, and the leading digit of these three is also used for identifying the half-word operand for 24-bit functions such as index register operations. The remaining 21 bits address a word, giving a range of over two million words; this range is divided into halves of which the first half is the main

Hardware caches *(1968)*

"Structural Aspects of the System/360 Model 85, Part II: The Cache,"
J. S. Liptay, IBM Systems Journal, Vol. 7, No. 1, 1968

Distributed file systems *(~1983)*

- *AFS, NFS, Sprite, Coda*

Web caching *(mid 1990s)*

- *SQUID, Akamai (CDNs)*

Virtual machine state caching *(early 2000s)*

- *Internet Suspend/Resume, Collective, Olive*

Key-Value caches *(mid 2000s)*

- REDIS

Caching is Universal

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

The Importance of Demand Fetch

Assumes ability to detect read operations

- ability to detect cache misses
- ability to interpose cache logic
- result is *total transparency*

In a file system this requires OS support

- distributed file systems (e.g AFS, Coda, …)
- FUSE interface

Systems like DropBox cannot do this

- lack of OS support simplifies implementation
- improves portability of code across OSes
- DropBox needs complete replicas everywhere

(aka "sync solution")

Without OS intercept

1. Even viewing one small file requires whole replica
2. Every update has to be propagated everywhere

Cache Consistency Strategies

emulate one-copy semantics of memory

Natural consequence of distribution + caching

Crucial dimension of transparency

Avoids changes to application software

Meets user expectations of system behavior

1. ***Broadcast invalidations***
2. ***Check on Use***
3. ***Callbacks***
4. ***Leases***
5. ***Skip Scary Parts***
6. ***Faith-based Caching***
7. ***Pass the Buck***

Many variants over the years, but these lie at their core

The Quest for Robustness

Coping With System Failures

ECC memory

Erasure coding

RAID (including mirroring)

Bad-block mapping

Wear-leveling of flash storage

Data replication and disaster recovery

Disconnected operation

…

Coping With Human Error

Use of separate address spaces (threads vs. processes)

Easy retrospection of file systems by users

- periodic read-only snapshots (AFS)
- Apple Time Machine, Elephant File System, …

Why is memory distinct from file system?

Single level stores have been proposed in the past

- but separation offers enhanced robustness
- well-formed open / read / write / close unlikely to be accidental
- contrast with wild memory write

Are Classic File Systems Dead?

Hot Topic Today

the death watch has begun

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 the Dalvik VM, 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"