Implementing Caching

What broad principles have we learned in ~50 years?

Hardware Support

Needed for efficiency when caching used at lower levels of system

Two distinct purposes

- ***Dereferencing:*** obtaining content from address
- ***Bookkeeping:*** implementing replacement policy

At chip speeds, both need hardware support

- entire cache implementation is in hardware (including microcode)
- extremely simple replacement policy; fixed line width objects

At OS level (virtual memory)

- only support for dereferencing essential
  TLB always; hardware-interpreted page tables often (except RISC)
- bookkeeping support optional
  referenced bits in IBM/370, none in other architectures
- large body of OS research on simulating hardware support
  clock-based schemes, use of memory protection, …

No hardware support needed at file system, database, and higher levels

Fetch Policy

Closer to hardware

- *fetch on demand* dominates
- fixed amount of *read-around*
  partly due to storage granularity, partly to exploit spatial locality
  (hardware cache line, block read in disk I/O, …)

Closer to user

- anticipatory policies (aka "prefetching") more important
- greater sophistication in policies, multivariate optimization
- greater opportunity for extensive prefetching

We will assume demand fetch for now

Prefetching is topic of project 2

Fixed vs. Variable Size Objects

Vast body of research and experience on fixed size objects

- **hardware: cache entries, lines**
- **disk I/O: blocks, sectors**
- **OS: virtual memory pages, disk blocks**

Fixed size offers considerable simplification

- **bit maps for allocation, trivial accounting**
- **uniform cost for all misses**
- **uniform importance of all misses**

Variable size better reflects user-perceived abstractions

- **more complex accounting & miss cost estimation**
- **some misses more serious than others**
- **huge range in size distribution (e.g. files)**
- **semantics-assisted strategies feasible**
  (font substitution, low-fidelity object substitution, …)

Cost of Fetch & Replacement - I

Uniform fetch cost assumption reasonable near hardware

Non-uniform cost more likely at higher system layers

- **variable size objects are obvious source of non-uniform cost**
- **rotational/seek delay in disks is another source**
- **differing network quality to different servers**

So picking victim object for replacement may involve many factors

- **how soon it might be needed again**
- **how expensive it might be to fetch again**
- **how much space is freed up if it is a victim**

Ideal victim → a very large object that is not needed for a long time, and is very cheap to fetch again

In the face of failures, *pain of non-serviceable miss* also relevant
(defer discussion of this to disconnected operation)

Cost of Fetch & Replacement - II

Body of research work in this space is small

- **Vast majority of work assumes**
  fixed size, fixed cost, equal importance
- **Hence only significant variable is**
  distance in the future to next reference to the object

Simple problem formulation from above assumptions

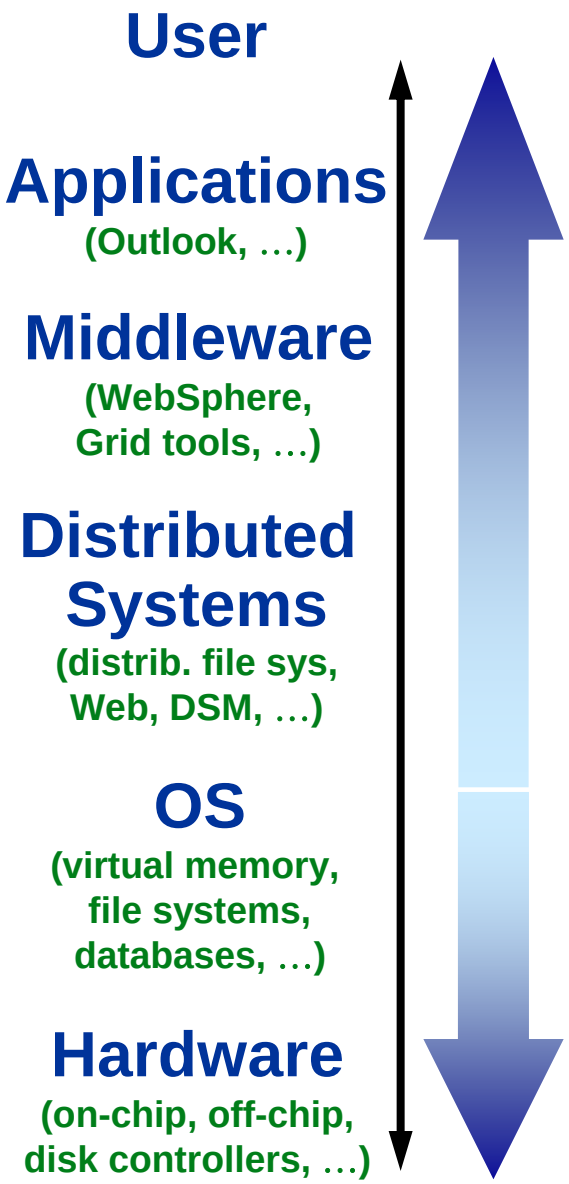
- **set of equal-sized data containers (aka "frames" in VM mgmt)**
- **large set of equal-sized, equal-importance data objects (aka "pages")**
- **sequence of integers ("reference string") → ids of accessed pages**
- **only metric of interest is miss ratio**

These assumptions less true at higher levels

- **but results from lower levels often used without too much thought!**
- **hard to tell if policy is bad (no catastrophic failures)**
- **few exceptions (e.g. Coda)**

Policies based on more realistic assumptions less tractable for analysis

Caching is Widely Applicable



- 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

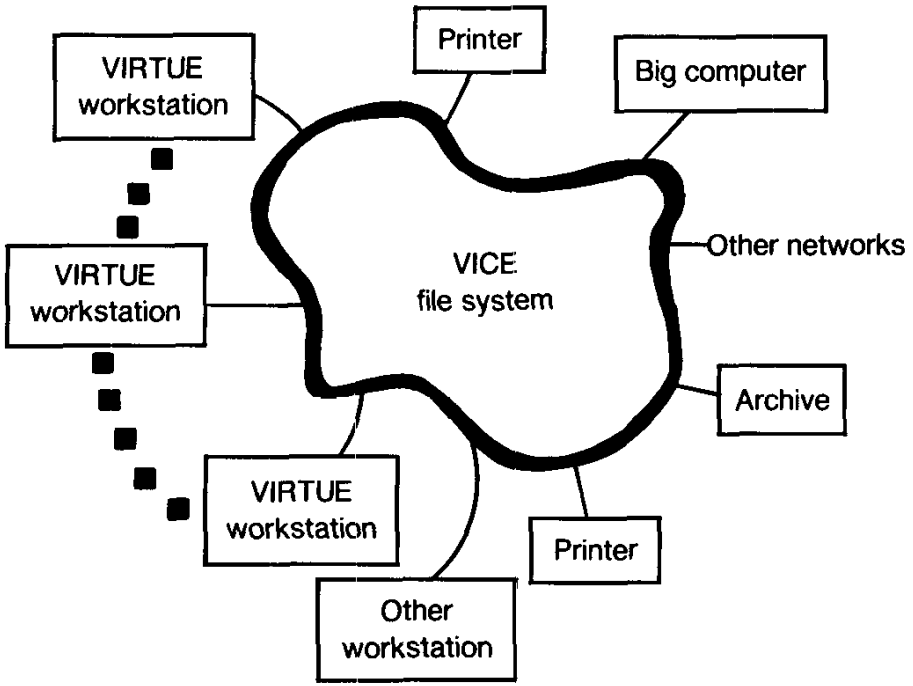
- Fixed size almost universal
- Fast, cheap decisions essential
- Miss ratio says it all (all misses equally bad)
- Greater temporal & spatial locality

Caching:
Key to Cloud-Mobile Convergence

AFS, Coda and ISR

Cloud Computing *circa* 1986

IBM-CMU Andrew Project (1983-1993)



from

"Andrew: a Distributed Personal Computing Environment"

Morris, J.H., Satyanarayanan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S., Smith, F.D.
Communications of the ACM, March 1986

The amoebalike structure in the middle, called VICE, is a collection of communication and computational resources serving as the backbone of a user community. Individual workstations, called VIRTUEs, are attached to VICE and provide users with the computational cycles needed for actual work as well as a sophisticated user–machine interface. (VICE stands for "Vast, Integrated Communications Environment"; VIRTUE, for "Virtue is reached through UNIX® and EMACS.")

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.

from

"Scalable, Secure and Highly-Available File Access"

Satyanarayanan, M.,
IEEE Computer, May 1990

1986 → 2011

Key new technology: *Virtual Machines (VMs)*

- **in fact, a very old technology redux**
- **invented by IBM in late 1960s**
- **obsoleted by personal computing in 1980s**

Why are VMs relevant to mobile computing?

- **big and clunky, hardly the picture of mobility!**
- **benefit: *perfect re-creation of execution state***

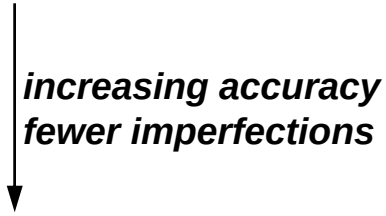
What happens if you combine this with *near-zero-cost hardware?*

Seamless Mobility

Driving vision behind AFS, Coda and ISR

"Wherever you go, your world follows you"

- ***efficiently (AFS)***
- ***reliably (Coda)***
- ***completely (ISR)***



Same vision as today's buzz about DropBox, iCloud, …

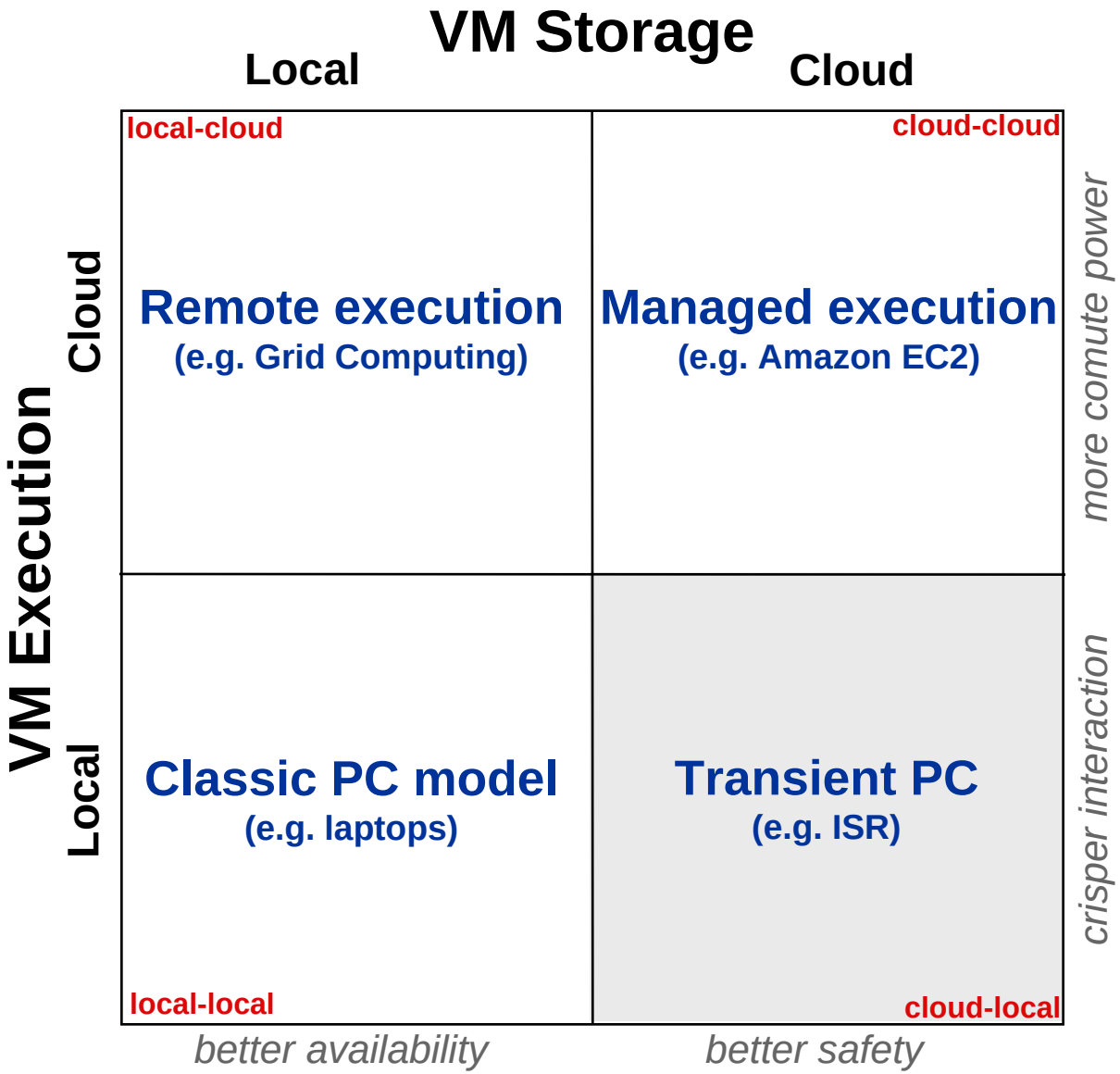
Enormous simplification for the user

- **never have to think about where my stuff is**
- **machine failure/shutdowns only minor inconvenience**

Also simplifies enterprise management

- **machines are just empty boxes (valuable state on servers)**
- **easy upgrade, backup, etc.**
- **basis for entire *ecosystem of management practices***

VM-based Cloud Computing Today



ISR Demo

ISR Client Architecture