Complications - I

Reality 1: ***Cost of remote data access often not uniform***

- **often takes the form *Ax + B* for *x* bytes**
- **may be more complex as *x* gets larger (e.g. TCP file transfer)**

Reality 2: ***"Nearby" objects often accessed soon after object access***

- **another empirical observation about real systems in real use**
- **referred to as *"spatial locality of reference"* or *"spatial locality"***

Reality 3: ***Remote data more coarsely addressable than local***

- **typically a scalability tradeoff at next level of memory hierarchy**
  - **same number of address bits can cover larger volume of data**
  - **e.g. cache line width, page size, whole file, tape mount**
- **"wholesale" versus "retail"**

Combining observations ⇒ *fetch more than you need on miss*

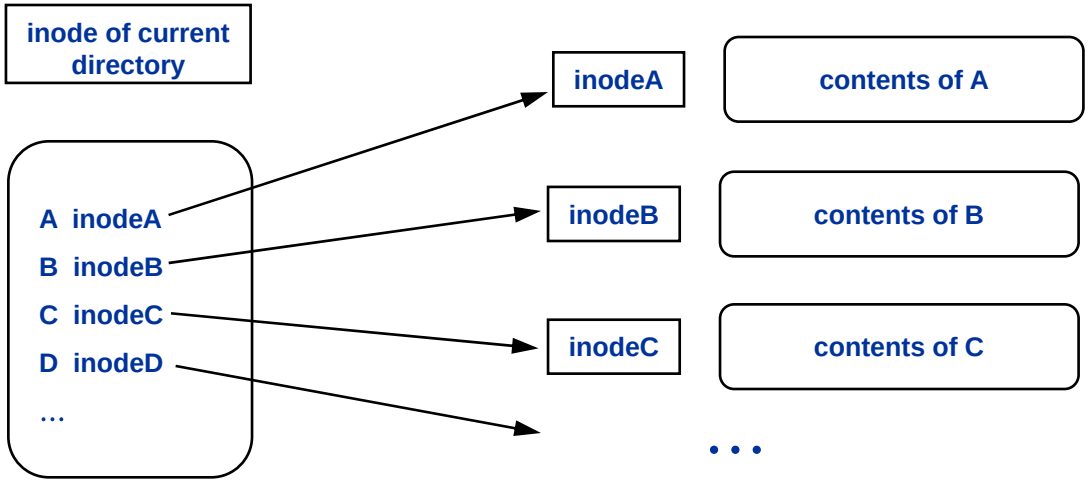
- **effectively amortizes cost of fetch**
- **assumption sometimes violated**
- **extra data fetched is then useless (may even hurt)**
  - **can be viewed as crude form of prefetching**

Complications - II

Temporal and spatial locality are very different properties

- **caching implementations often tightly combine these assumptions**
- **one can exist without the other**
  - **spatial without temporal: linear scan of huge file**
  - **temporal without spatial: tight loop accessing just one object**

For example, consider typical implementation of "rm -f *"



shell expands "*" into list

loop iterates through list

- **stat object**
- **unlink object**

parent directory exhibits temporal locality

directory entries exhibit spatial locality

Complications - III

Local storage management

- **overhead typically makes first reference more expensive**
  local copy allocation, lookup table update, etc.
- **remote storage often much bigger than local storage**
  ⇒ recycling of local storage for copies
- ***replacement* policy becomes significant**

Updates to remote copy need to be propagated to local copy

- **and vice versa (local updates need to be made visible everywhere)**
- ***cache consistency* is a significant problem**
- **goal is *one-copy semantics***

Refinements of Basic Idea

Cache idea can be applied recursively ⇒ *"multi-level cache"* or *"memory hierarchy"*

Persistent caches (typically on-disk) increase longevity

Local copy can be used to mask remote failures ⇒ *caching for disconnected operation*

Cooperative caching (across users) exploits *"communal locality"*

- empirical observation about groups of users

Outsourced caching to third parties (e.g. Akamai)

Users notice long fetch delays → *translucent caching*

A Brief History of Caching

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

- ***"Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store,"*** **John Fotheringham, Communications of the ACM, 1961, pp 435-436**
- **revisited by OS researchers (1970's - 1990s)**

Hardware caches came next *(1968)*

- **"*Structural Aspects of the System/360 Model 85, Part II: The Cache,*" J. S. Liptay, IBM Systems Journal, Vol. 7, No. 1, 1968, pp. 15-21**
- **extensive study of hardware cache coherence (1970-1990s)**

Distributed file systems came after that *(~1983-84, ~1988-1991)*

- ***AFS, NFS, Sprite, Coda***
- **extensive study by researchers (1980's - present)**

Web caching *(~1994)*

- **cooperative caching, outsourced caching (mid-1990s to present)**

Virtual machine state caching *(~2002-2008)*

- ***Internet Suspend/Resume***

Result caching in discard-based search *(~2007-2009)*

- ***Diamond***