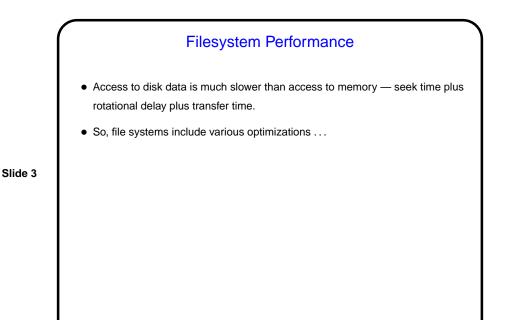


Slide 1

Slide 2

Minute Essay From Last Lecture — Some Responses

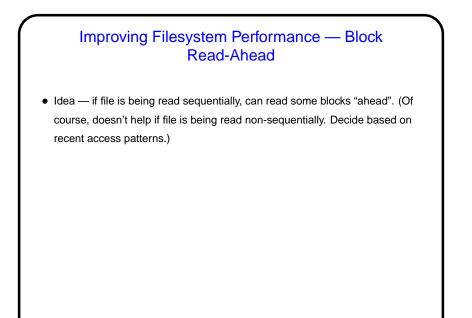
- Different requirements based on hardware, what the system is used for, etc. Different goals.
- "Better ideas".
- Lots of design choices, so people have tried out different ones.
- No one filesystem will be good for all uses.
- Everybody likes his/her ideas best (companies too).
- Avoidance of copyright infringement (?).
- · Sometimes have to keep old ones around for compatibility.
- (Also review added slide filesystem quotas.)





- Idea keep some disk blocks in memory; keep track of which ones are there using hash table (base hash code on device and disk address).
- When cache is full and we must load a new block, which one to replace?
 Could use algorithms based on page replacement algorithms, could even do
 LRU accurately though that might be wrong (e.g., want to keep data blocks being filled).
- When should blocks be written out?
 - If block is needed for file system consistency, could write out right away. If block hasn't been written out in a while, also could write out, to avoid data loss in long-running program.
 - Two approaches: "Write-through cache" (Windows) always write out modified blocks right away. Periodic "sync" to write out (UNIX).

Slide 4



Slide 5

Improving Filesystem Performance — Reducing Disk Arm Motion

 Group blocks for each file together — easier if bitmap is used to keep track of free space. If not grouped together — "disk fragmentation" may affect performance.

Slide 6

• If i-nodes are being used, place them so they're fast to get to (and so maybe we can read an i-node and associated file block together).

