Improving Adaptivity in Mobile File Systems

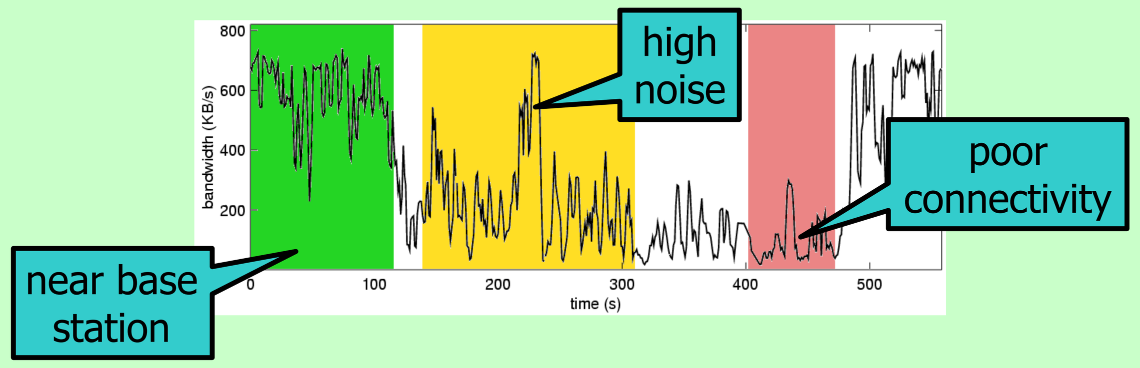
Benjamin Atkin and Kenneth P. Birman

Reliable Distributed Systems Group

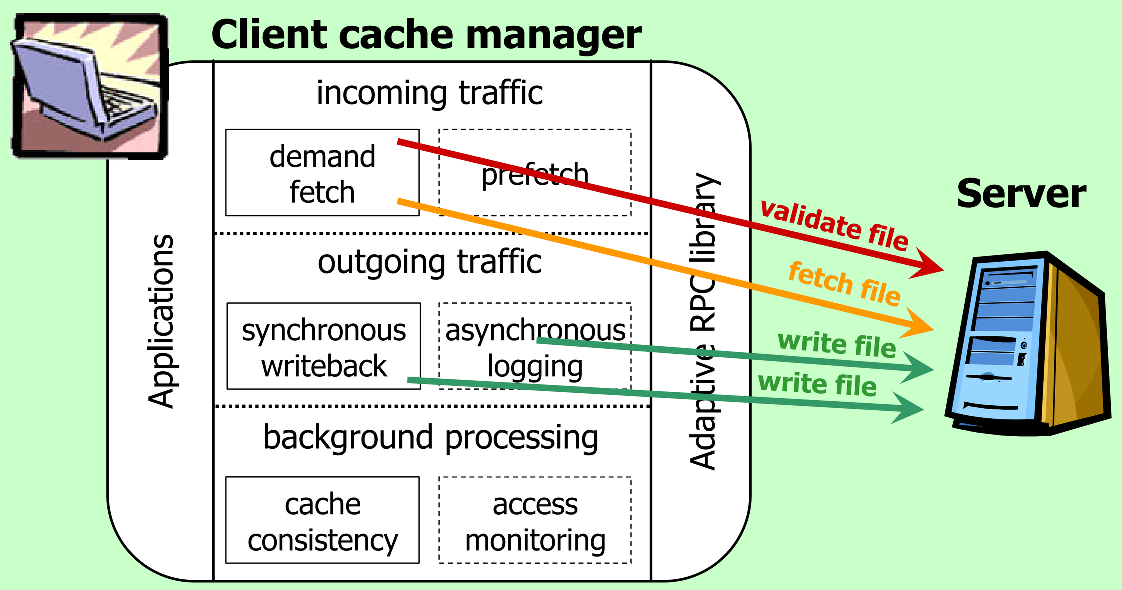
Cornell University

Challenges in mobile file systems design

Today's mobile file systems must adapt to wide variations in network bandwidth in order to provide good performance.

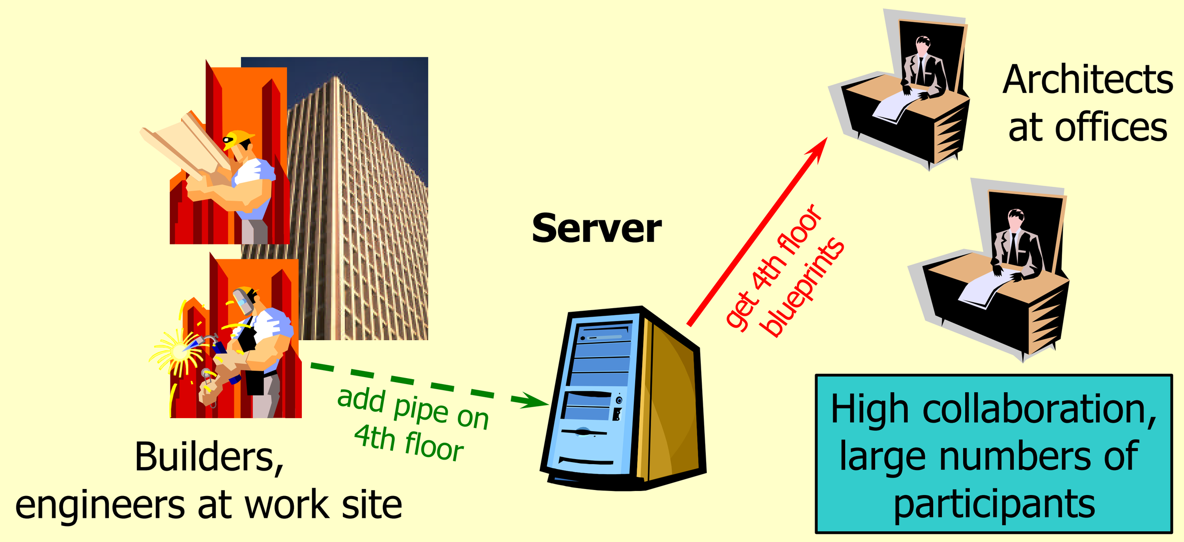


MFS: a Mobile File System



Cache consistency

Asynchronous writes reduce delays when bandwidth is low, but allow a client to hold an update for a file without the server knowing it's dirty.



MFS/CC consistency algorithm

1. Invalidate files if explicit invalidation is required
   - Only invalidate when there's concurrent writeback traffic
2. Separate writeback of shared and non-shared files to reduce delay before changes are visible

MFS design principles

1. Not all Remote Procedure Calls (RPCs) are equally important.
2. Bandwidth can be conserved by making decisions about allocation based on the current set of queued RPCs.

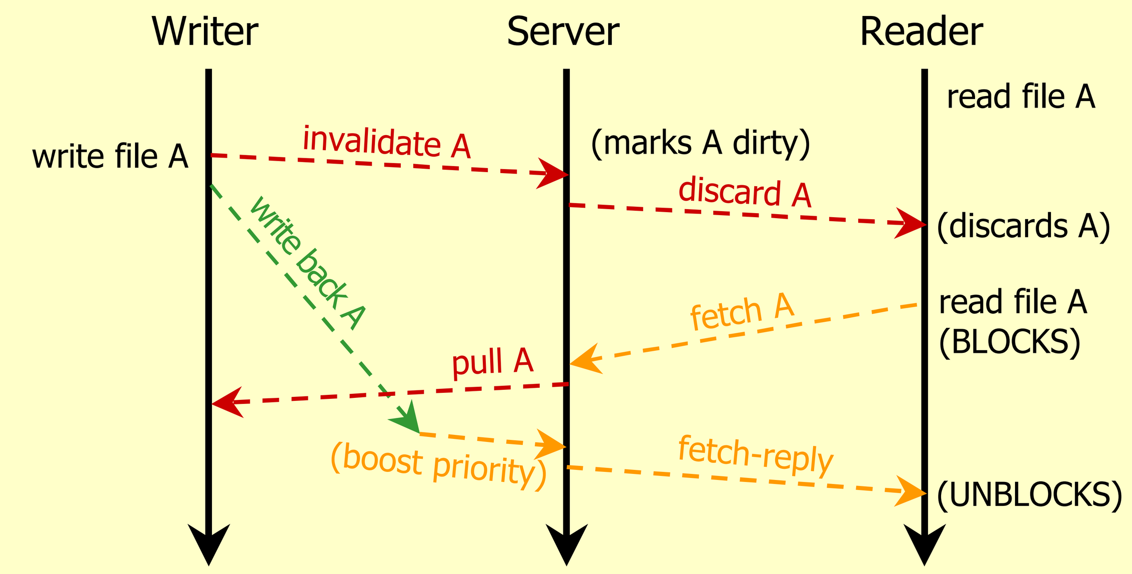
Priorities for RPCs

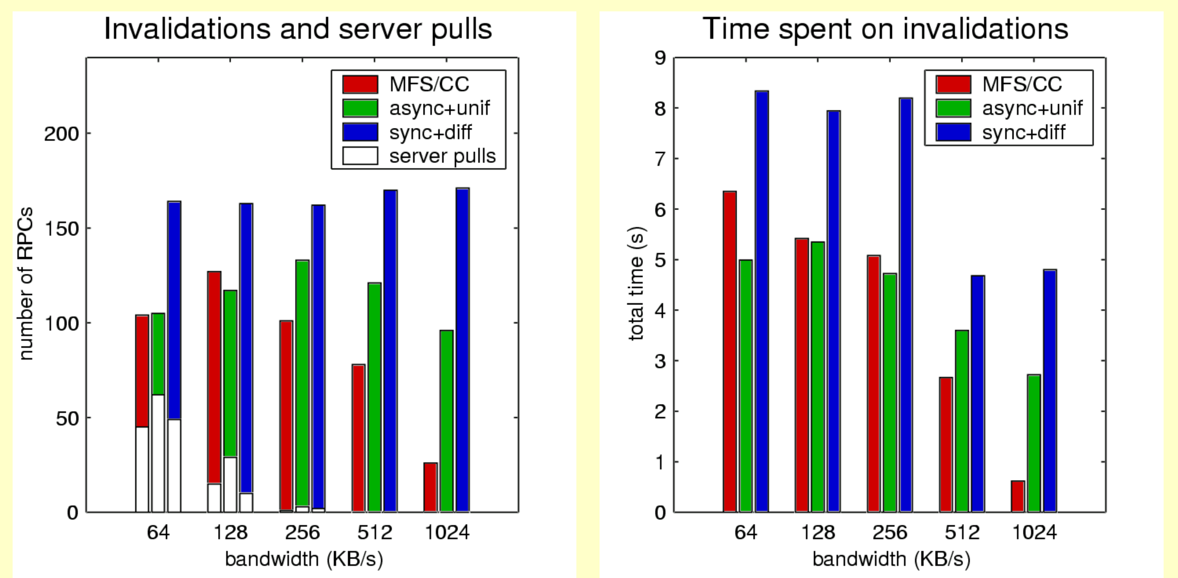
MFS assigns RPC priorities according to how long they delay applications

- High priority for small RPCs
- High priority for reads
- Low priority for "background" RPCs (writes, prefetches)

Making writes asynchronous can further decrease application delay.

MFS/CC in action

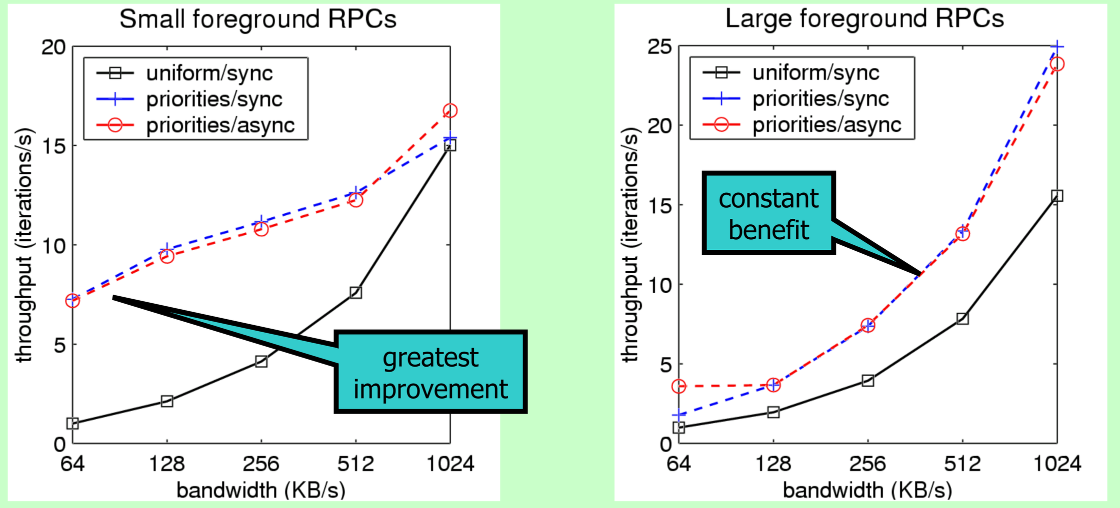




MFS/CC uses asynchronous invalidations (async) and different priorities for writing back shared and unshared files (diff). We compare with synchronous invalidations (sync) and uniform writes (unif).

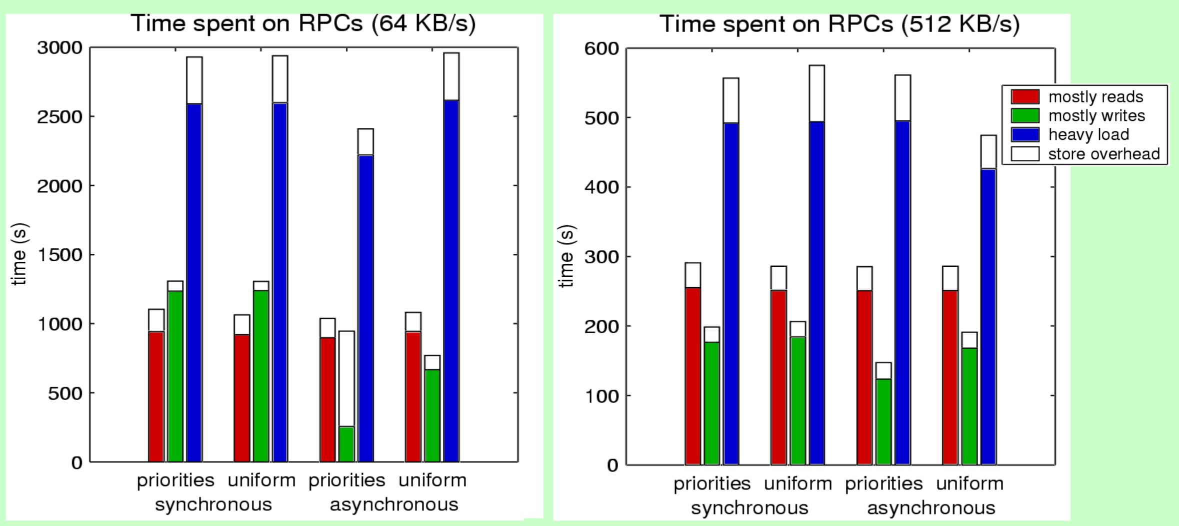
MFS/CC is equivalent to synchronous invalidations in terms of avoiding inconsistencies (number of server pulls), but reduces the invalidation overhead.

Microbenchmarks

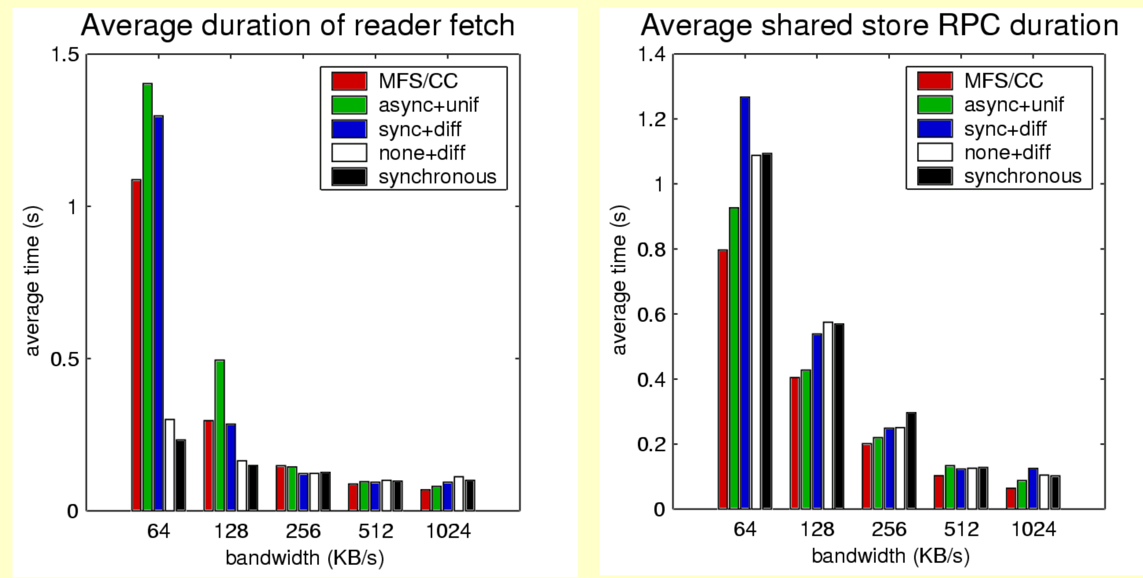


We compare the speedup of a foreground workload running at the same time as a stream of file writes, relative to the time taken with synchronous writes at a bandwidth of 64 KB/s.

NT file system macrobenchmarks



Performance of priority and writeback for 3 different NTFS file traces. Asynchronous writes with priorities generally improve the overall time from the first to the last operation, and the time to write back all files.



MFS/CC decreases the time readers and writers must wait to access shared files, since it prioritises modifications to these files over changes to unshared files. This reduces the waiting time for a reader accessing a "dirty" file, and the likelihood a file will be dirty when a reader accesses it.

Current work

- Submitted to FAST'04
- Automatically generating caching policies for files based on access patterns
- Applying MFS-style adaptation techniques in other application domains

The authors were supported in part by DARPA under AFRL grant RADC F30602-99-1-0532, and by AFOSR under MURI grant F49620-02-1-0233, with additional support from Microsoft Research and from the Intel Corporation.