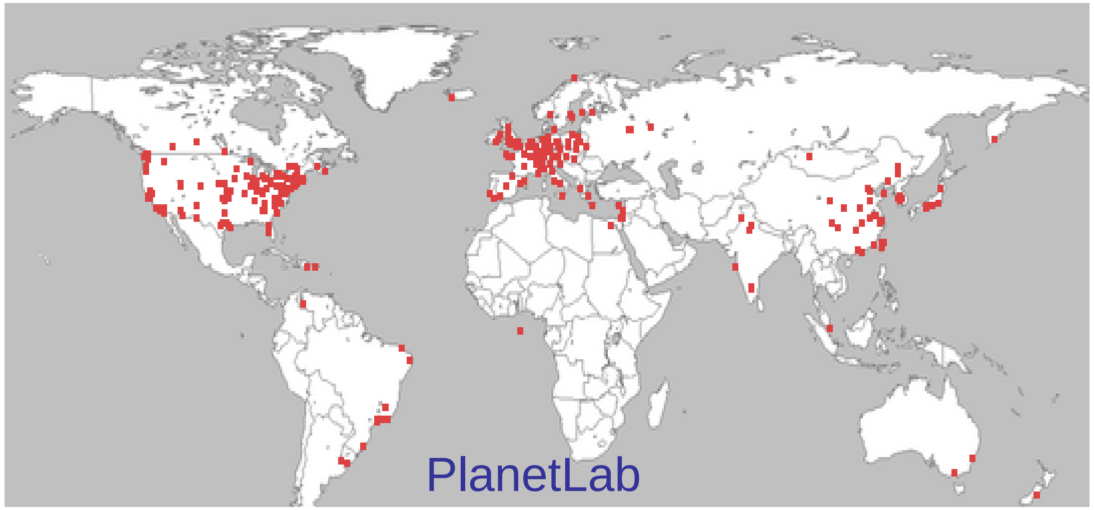
Flexible, Wide-Area Storage for Distributed Systems with WheelFS

Jeremy Stribling,
Yair Sovran, Irene Zhang, Xavid Pretzer,
Jinyang Li, M. Frans Kaashoek, and Robert Morris

MIT CSAIL & New York University

Wide-Area Storage: The Final Frontier



- Apps store data on widely-spread resources
  - Testbeds, Grids, data centers, etc.
  - Yet there's no universal storage layer
- What's so hard about the wide-area?
  - Failures and latency and bandwidth, oh my!

Apps Handle Wide-Area Differently

- CoralCDN prefers low delay to strong consistency (Coral Sloppy DHT)
- Google stores email near consumer (Gmail's storage layer)
- Facebook forces writes to one data center (Customized MySQL/Memcached)

→ Each app builds its own storage layer

Problem: No Flexible Wide-Area Storage

- Apps need control of wide-area tradeoffs
  - Fast timeouts vs. consistency
  - Fast writes vs. durability
  - Proximity vs. availability
- Need a common, familiar API: File system
  - Easy to program, reuse existing apps
- No existing DFS allows such control

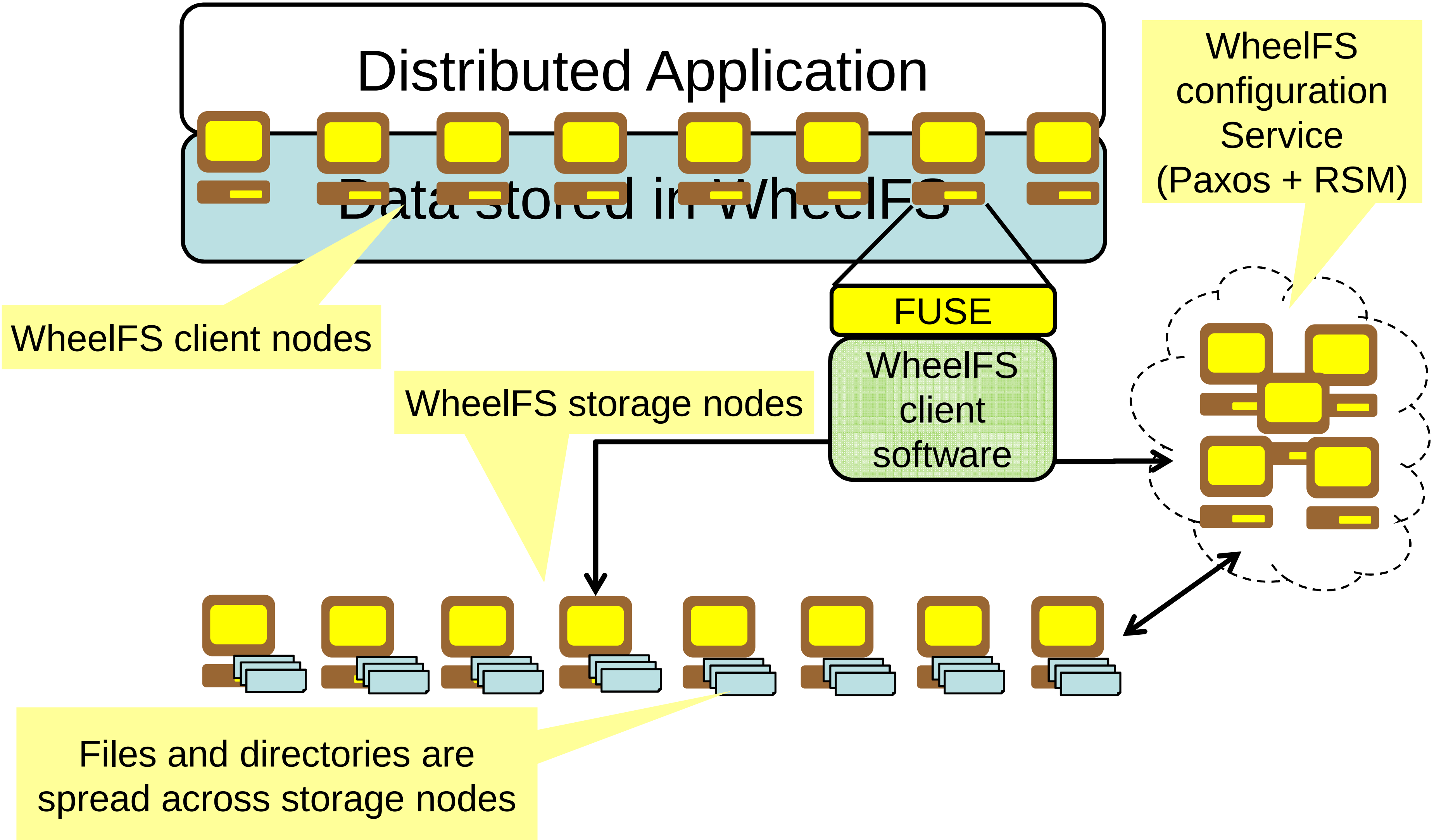
Solution: Semantic Cues

- Small set of app-specified controls
- Correspond to wide-area challenges:
  - **EventualConsistency**: relax consistency
  - **RepLevel**=*N:* control number of replicas
  - **Site=***site*: control data placement
- Allow apps to specify on per-file basis
  - */fs/.EventualConsistency/file*

Contribution: WheelFS

- Wide-area file system
- Apps embed cues directly in pathnames
- Many apps can reuse existing software
- Multi-platform prototype w/ several apps

WheelFS Design Overview

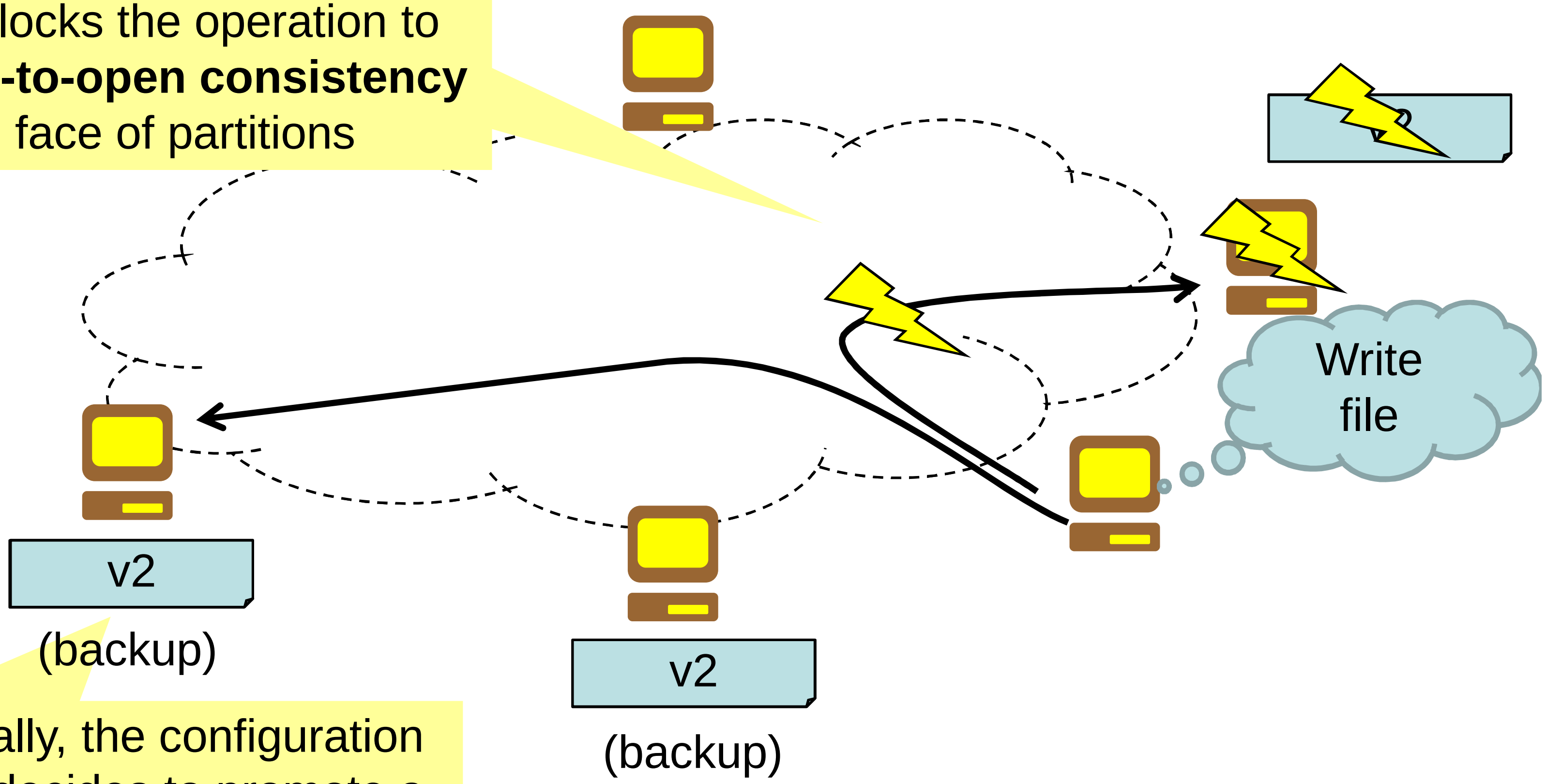


WheelFS Default Operation

- Files have a primary and two replicas
  - A file's primary is its creator
- Clients can cache files
  - Lease-based invalidation protocol
- Strict close-to-open consistency
  - All operations serialized through the primary

Enforcing Close-to-Open Consistency

By default, failing to reach the primary blocks the operation to offer **close-to-open consistency** in the face of partitions



Eventually, the configuration service decides to promote a backup to be primary

Wide-Area Challenges

- Transient failures are common
  - Fast timeouts vs. consistency
- High latency
  - Fast writes vs. durability
- Low wide-area bandwidth
  - Proximity vs. availability

Only applications can make these tradeoffs

Semantic Cues Gives Apps Control

- Apps want to control consistency, data placement ...
- How? Embed cues in path names

*/wfs/cache/**EventualConsistency**/foo*

/wfs/cache/a/b/foo

/wfs/cache/a/b/foo

→ Flexible and minimal interface change

Semantic Cue Details

- Cues can apply to directory subtrees

*/wfs/cache/**.EventualConsistency**/a/b/foo*

Cues apply recursively over an entire subtree of files

- Multiple cues can be in effect at once

*/wfs/cache/**.EventualConsistency/.RepLevel=2**/a/b/foo*

Both cues apply to the entire subtree

- Assume developer applies cues sensibly

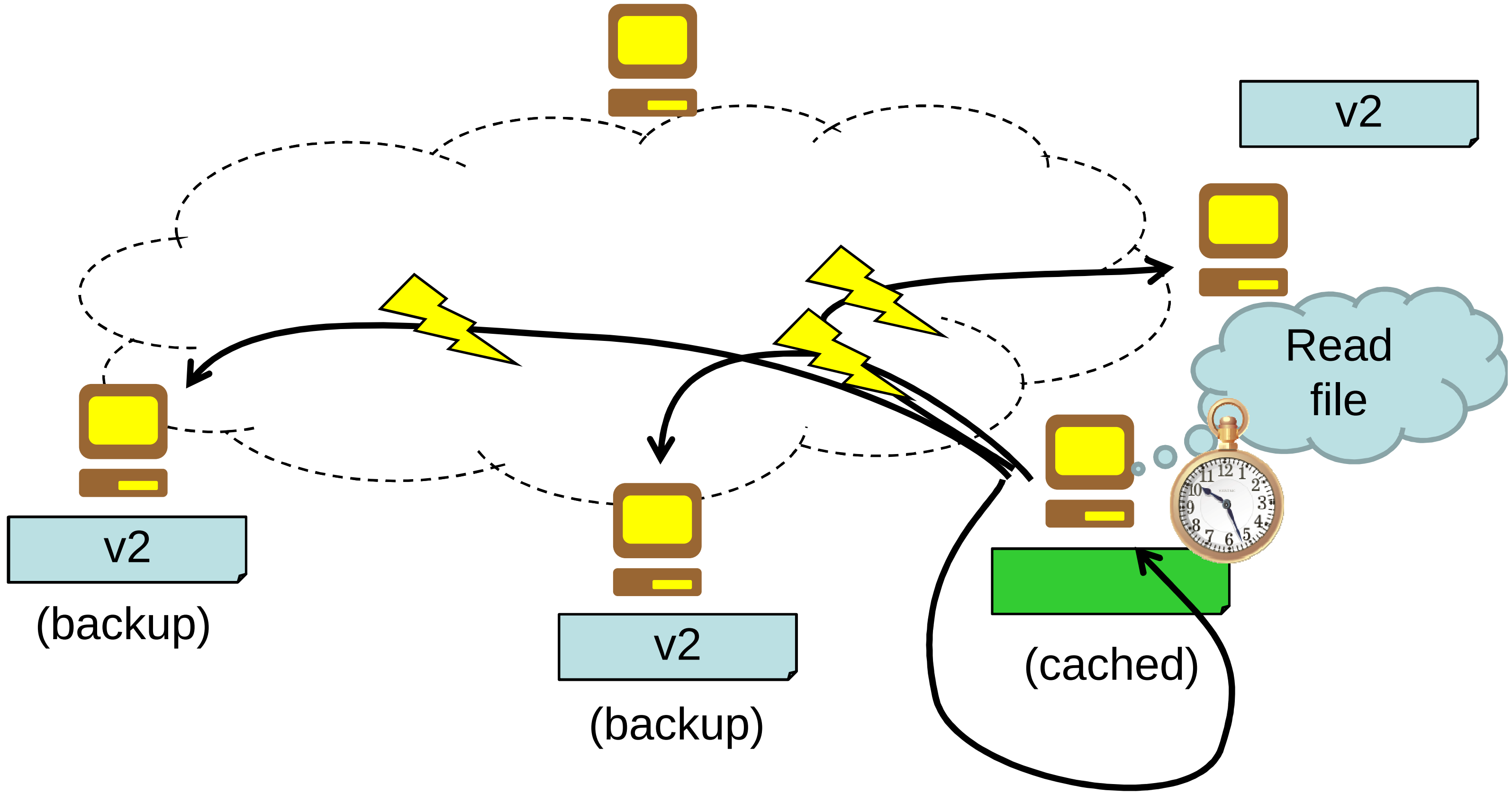
A Few WheelFS Cues

	Name	Purpose
Durability	**RepLevel=** *(permanent)*	How many replicas of this file should be maintained
Large reads	**HotSpot** *(transient)*	This file will be read simultaneously by many nodes, so use p2p caching
Hint about data placement	**Site=** *(permanent)*	Hint which group of nodes a file should be stored
Consistency	**Eventual-Consistency** *(trans/perm)*	Control whether reads must see fresh data, and whether writes must be serialized

Cues designed to match wide-area challenges

Eventual Consistency: Reads

- Read latest version of the file you can find quickly
- In a given time limit (**.MaxTime=**)



Eventual Consistency: Writes

- Write to any replica of the file

Reconciling divergent replicas:

Directories

- Merge replicas into single directory by taking union of entries

→ Tradeoff: May lose some unlinks

Files

- Choose one of the replicas to win

→ Tradeoff: May lose some writes

(No application involvement)

v3

Write file

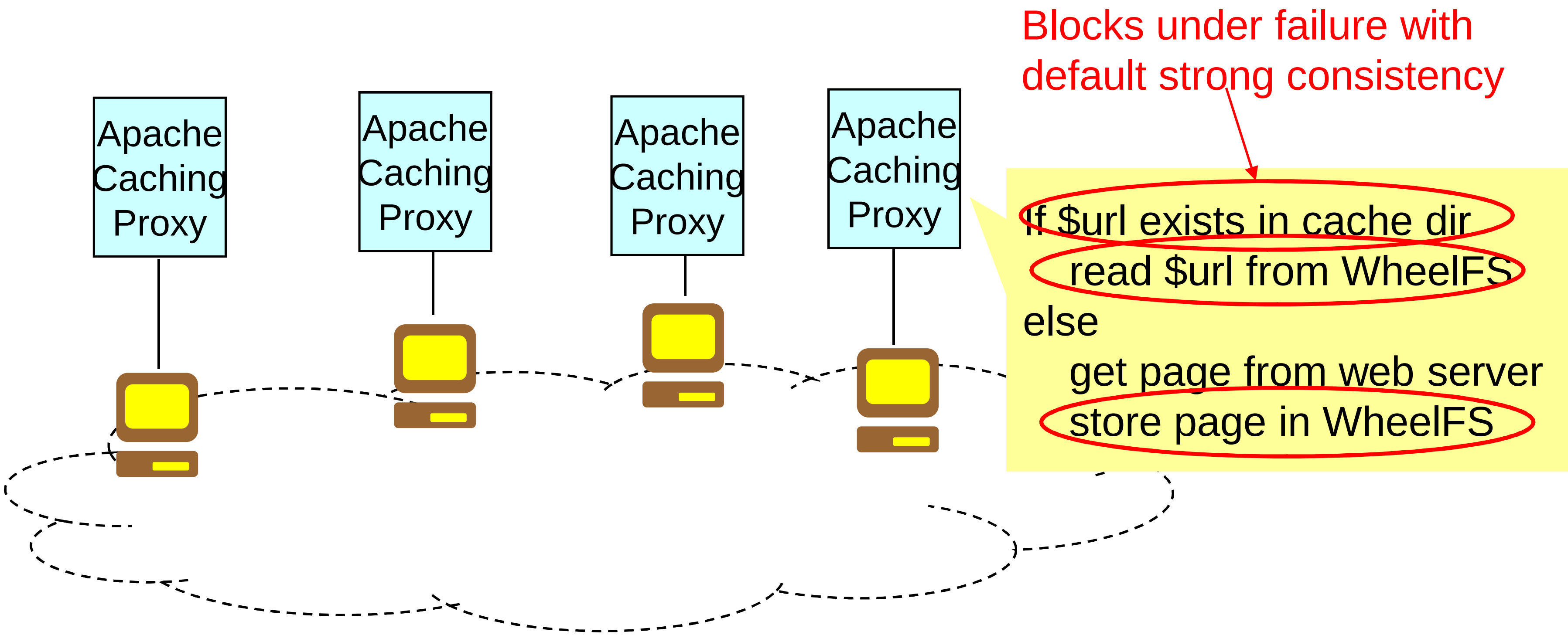
background process will merge divergent replicas

v3

(backup)

Create new version at backup

Example Use of Cues: Cooperative Web Cache (CWC)



One line change in Apache config file: /wfs/cache/$URL

Example Use of Cues: CWC

- Apache proxy handles potentially stale files well
  - The freshness of cached web pages can be determined from saved HTTP headers

Cache dir: */wfs/cache/***.EventualConsistency/.MaxTime=200/.HotSpot**

Read a cached file even when the corresponding primary cannot be contacted

Write the file data anywhere even when the corresponding primary cannot be contacted

Reads only block for 200 ms; after that, fall back to origin server

Tells WheelFS to read data from the nearest client cache it can find

WheelFS Implementation

- Runs on Linux, MacOS, and FreeBSD
- User-level file system using FUSE
- 20K+ lines of C++
- Unix ACL support, network coordinates
- Deployed on PlanetLab and Emulab

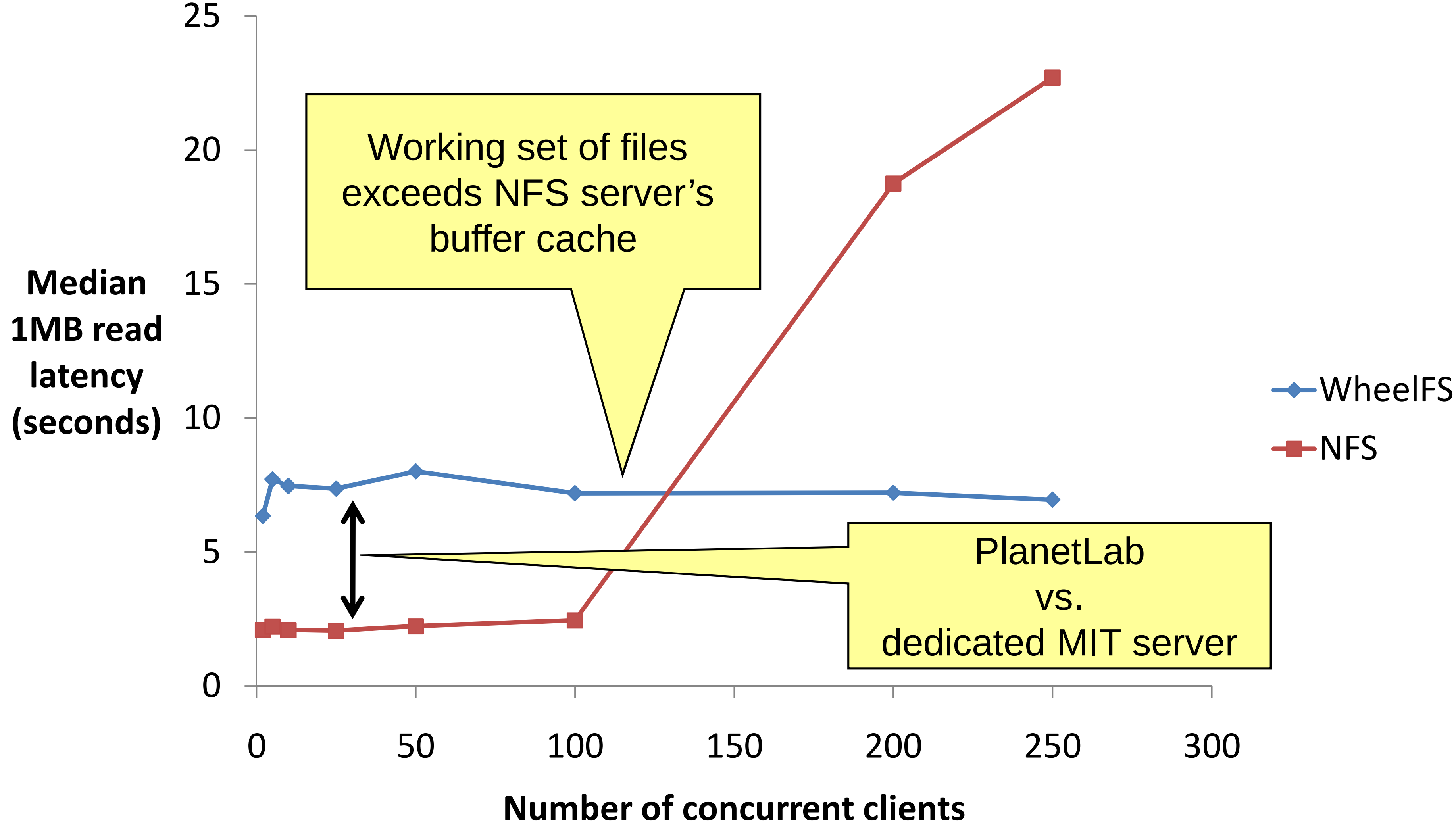
Applications Evaluation

App	Cues used	Lines of code/configuration written or changed
Cooperative Web Cache	**.EventualConsistency, .MaxTime, .HotSpot**	1
All-Pairs-Pings	**.EventualConsistency, .MaxTime, .HotSpot, .WholeFile**	13
Distributed Mail	**.EventualConsistency, .Site, .RepLevel, .RepSites, .KeepTogether**	4
File distribution	**.WholeFile, .HotSpot**	N/A
Distributed make	**.EventualConsistency** (for objects)**, .Strict** (for source)**, .MaxTime**	10

Performance Questions

1. Does WheelFS scale better than a single-server DFS?
2. Can WheelFS apps achieve performance comparable to apps w/ specialized storage?
3. Do semantic cues improve application performance?

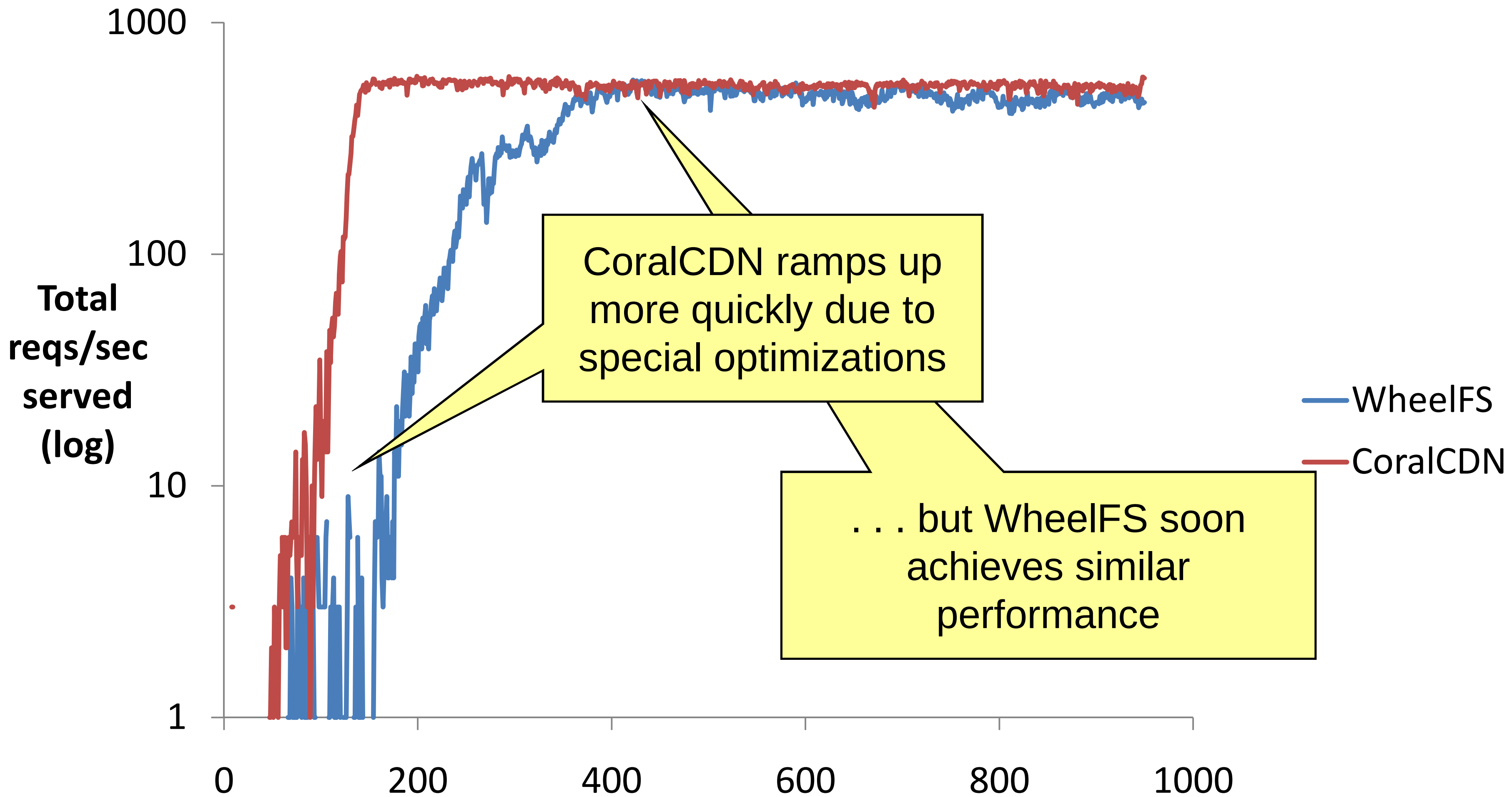
WheelFS Out-scales NFS on PlanetLab



CWC Evaluation

- 40 PlanetLab nodes as Web proxies
- 40 PlanetLab nodes as clients
- Web server
  - 400 Kbps link
  - 100 unique 41 KB pages
- Each client downloads random pages
  - (Same workload as in CoralCDN paper)
- CoralCDN vs. WheelFS + Apache

WheelFS Achieves Same Rate As CoralCDN

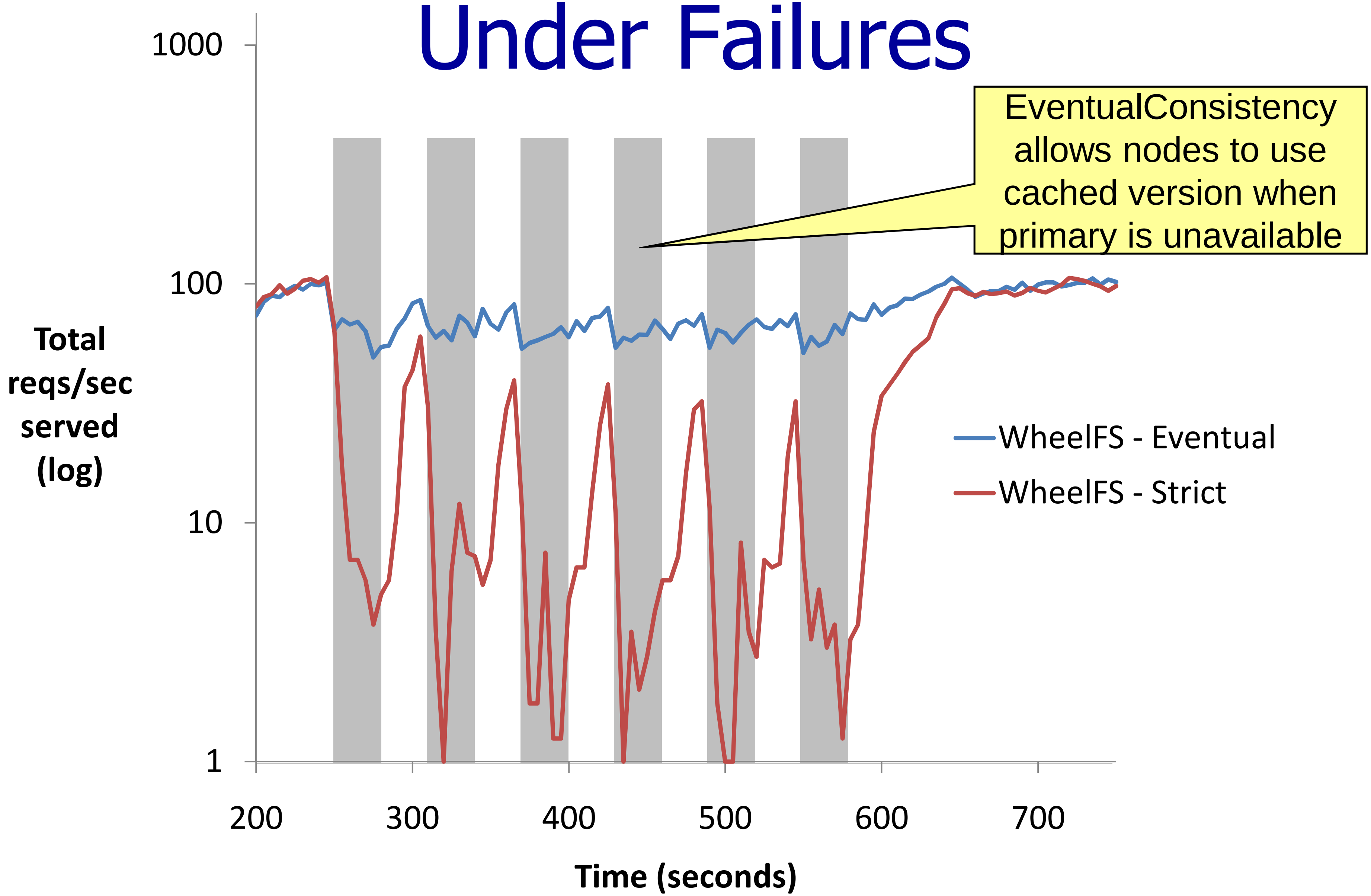


Total reqs/unique page: > 32,000
Origin reqs/unique page: 1.5 (CoralCDN) 2.6 (WheelFS)

CWC Failure Evaluation

- 15 proxies at 5 wide-area sites on Emulab
- 1 client per site
- Each minute, one site offline for 30 secs
  - Data primaries at site unavailable
- Eventual vs. strict consistency

EC Improves Performance Under Failures



Related File Systems

- Single-server FS: NFS, AFS, SFS
- Cluster FS: Farsite, GFS, xFS, Ceph
- Wide-area FS: Shark, CFS, JetFile
- Grid: LegionFS, GridFTP, IBP

- WheelFS gives applications control over wide-area tradeoffs

Storage Systems with Configurable Consistency

- PNUTS [VLDB ‘08]
  – Yahoo!’s distributed, wide-area database
- PADS [See next talk]
  – Flexible toolkit for creating new storage layers

- WheelFS offers broad range of controls in the context of a single file system

Conclusion

- Storage must let apps control data behavior
- Small set of *semantic cues* to allow control
  - **Placement, Durability**, **Large reads** and **Consistency**
- WheelFS:
  - Wide-area file system with semantic cues
  - Allows quick prototyping of distributed apps

http://pdos.csail.mit.edu/wheelfs