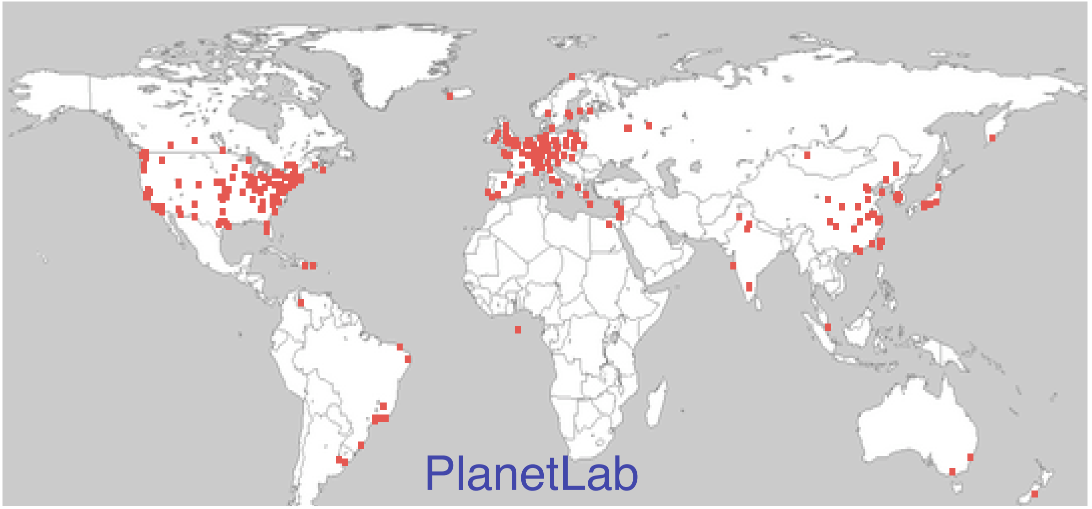
Flexible, Wide-Area Storage for Distributed Systems with WheelFS

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

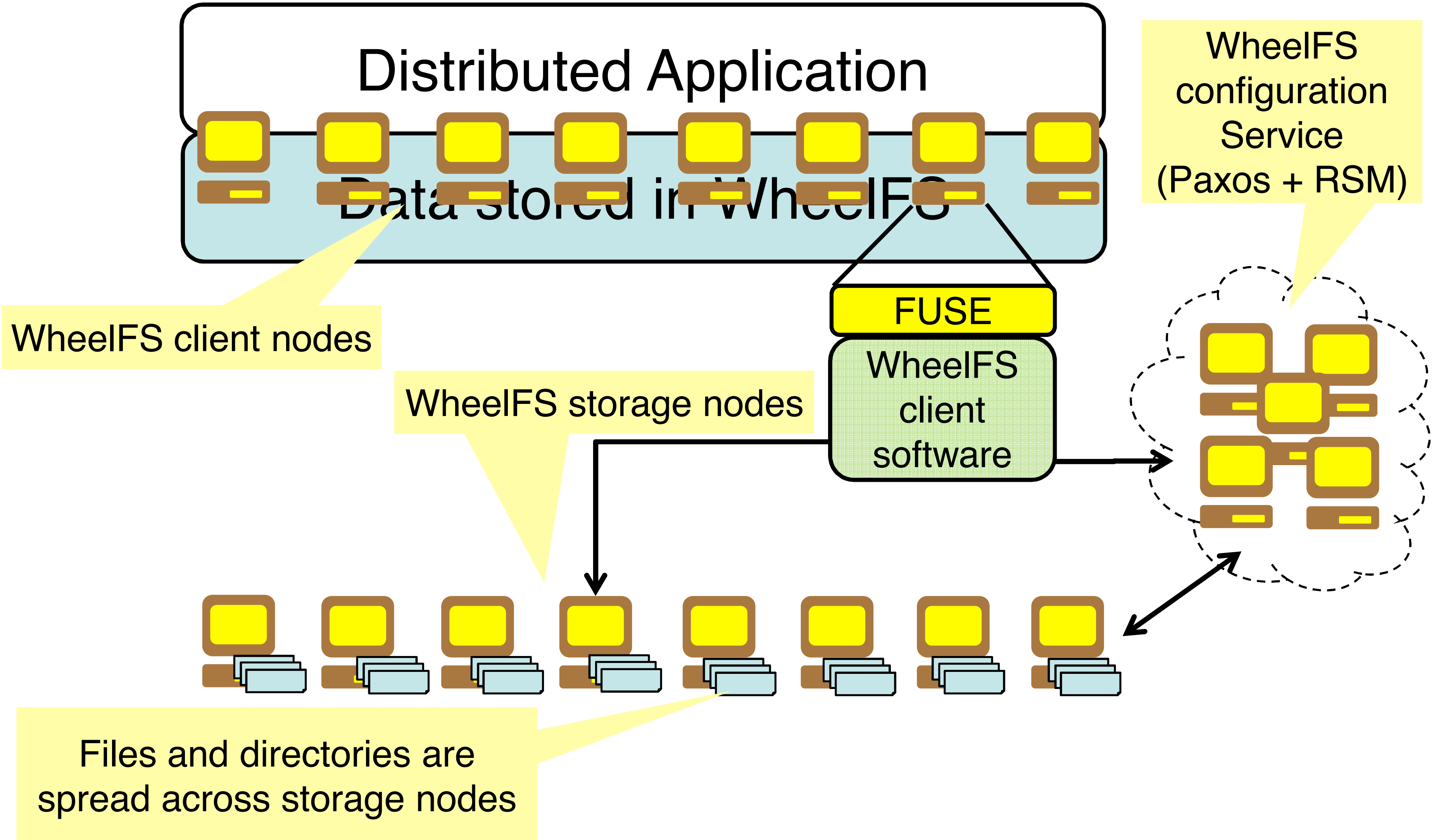
Solution: Semantic Cues

- 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

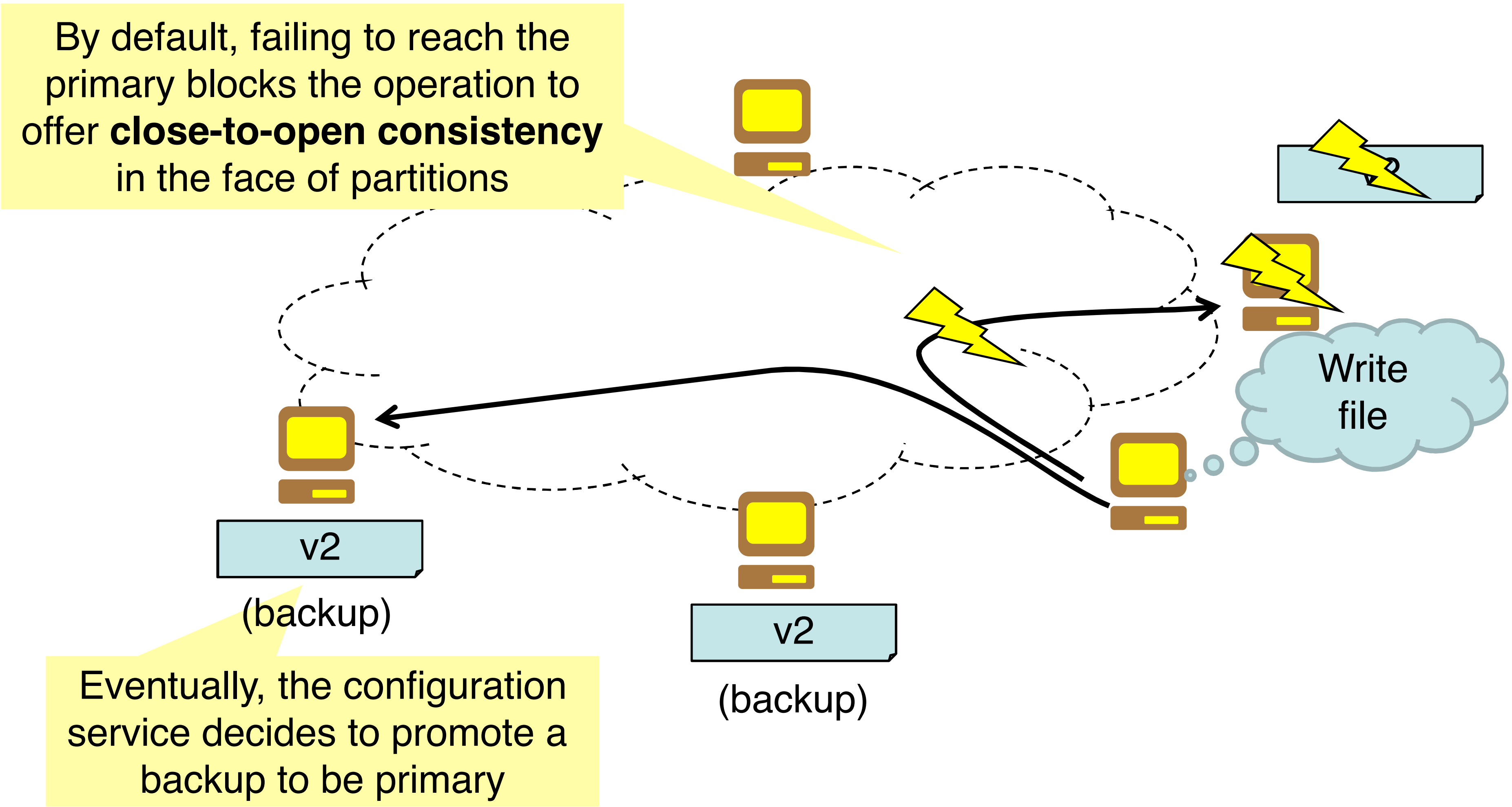
WheelFS Design Overview



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



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/a/b/foo

*/wfs/cache/a/b/**EventualConsistency**/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

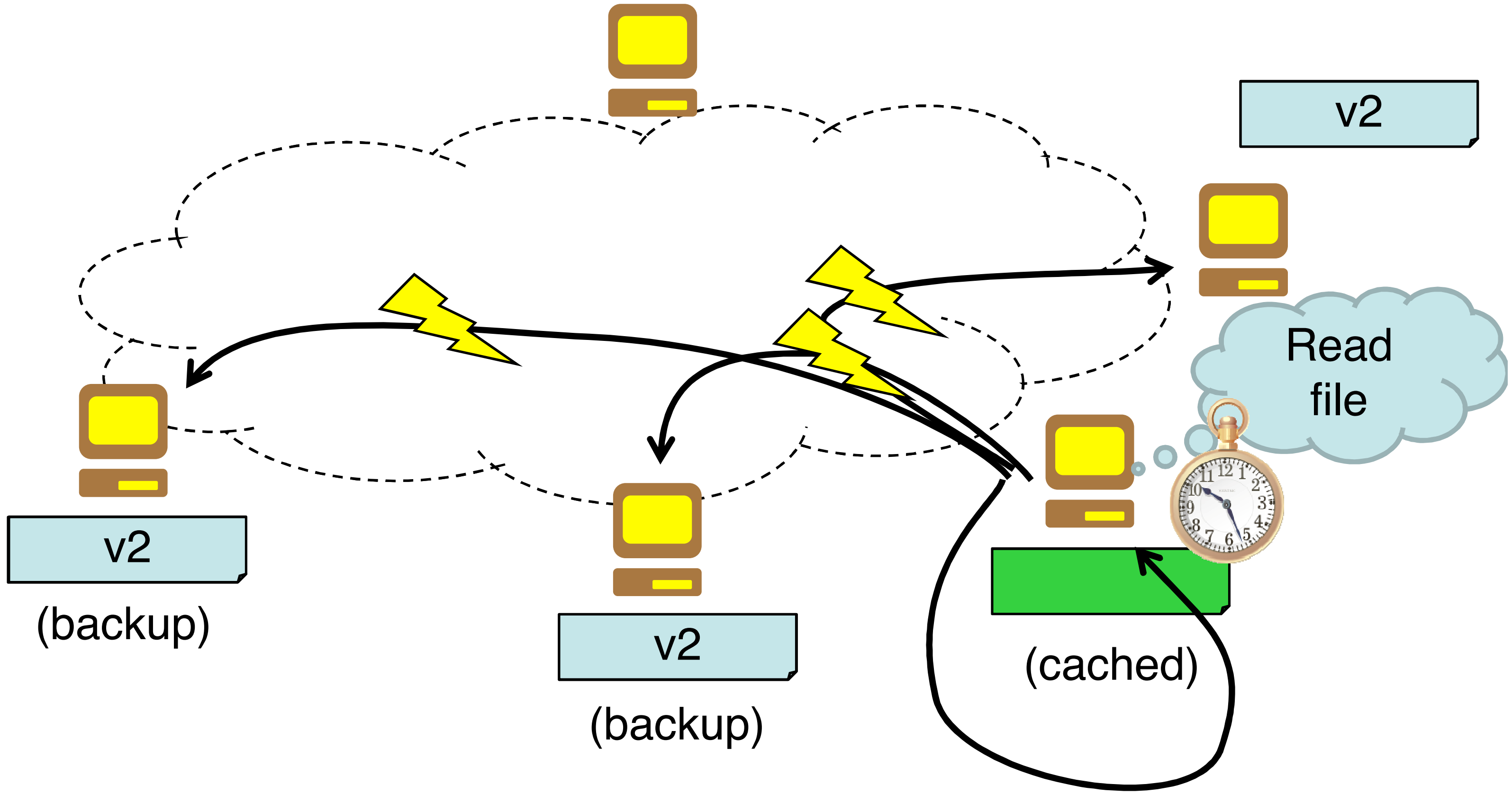
A Few WheelFS Cues

	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

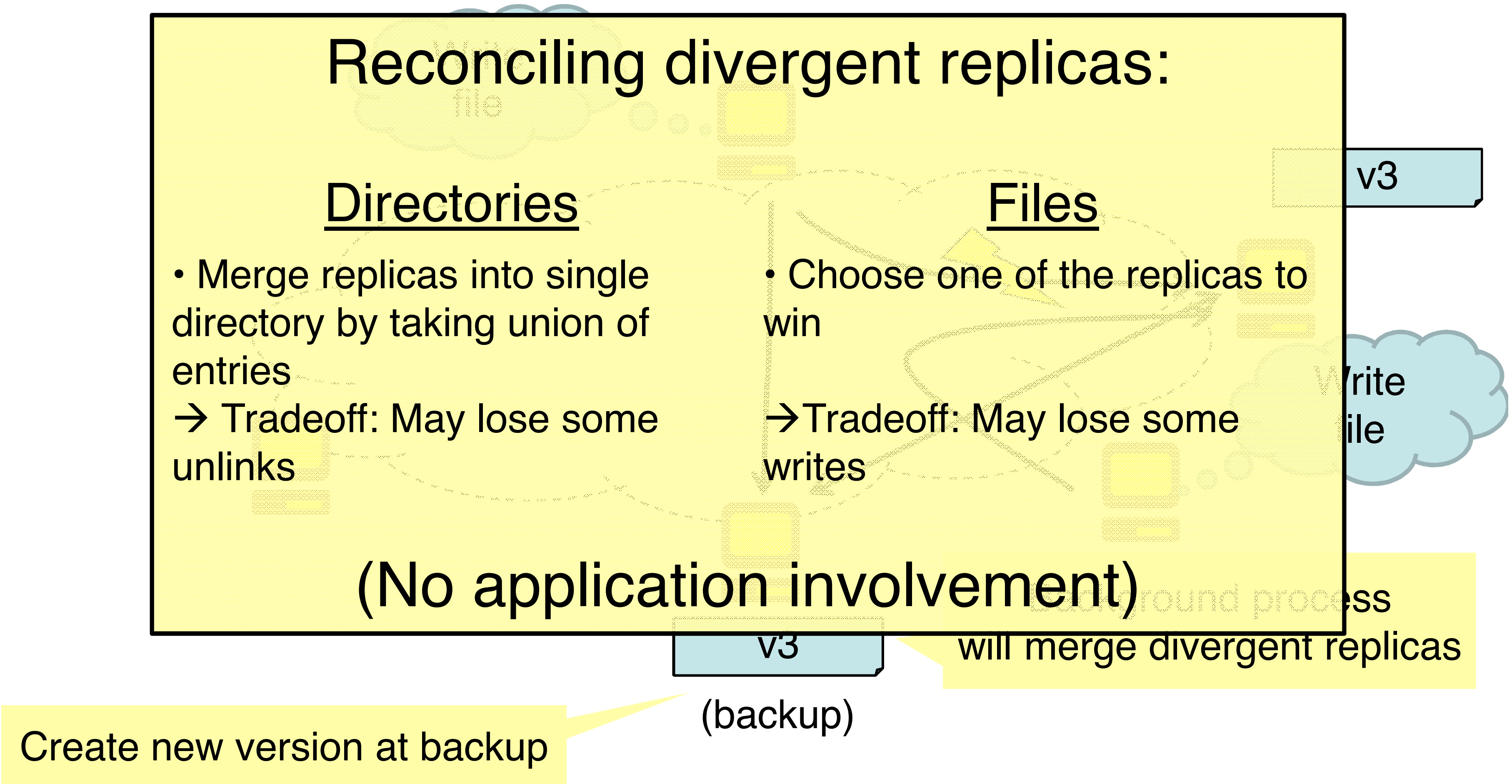
→ Tradeoff: May lose some unlinks

Files

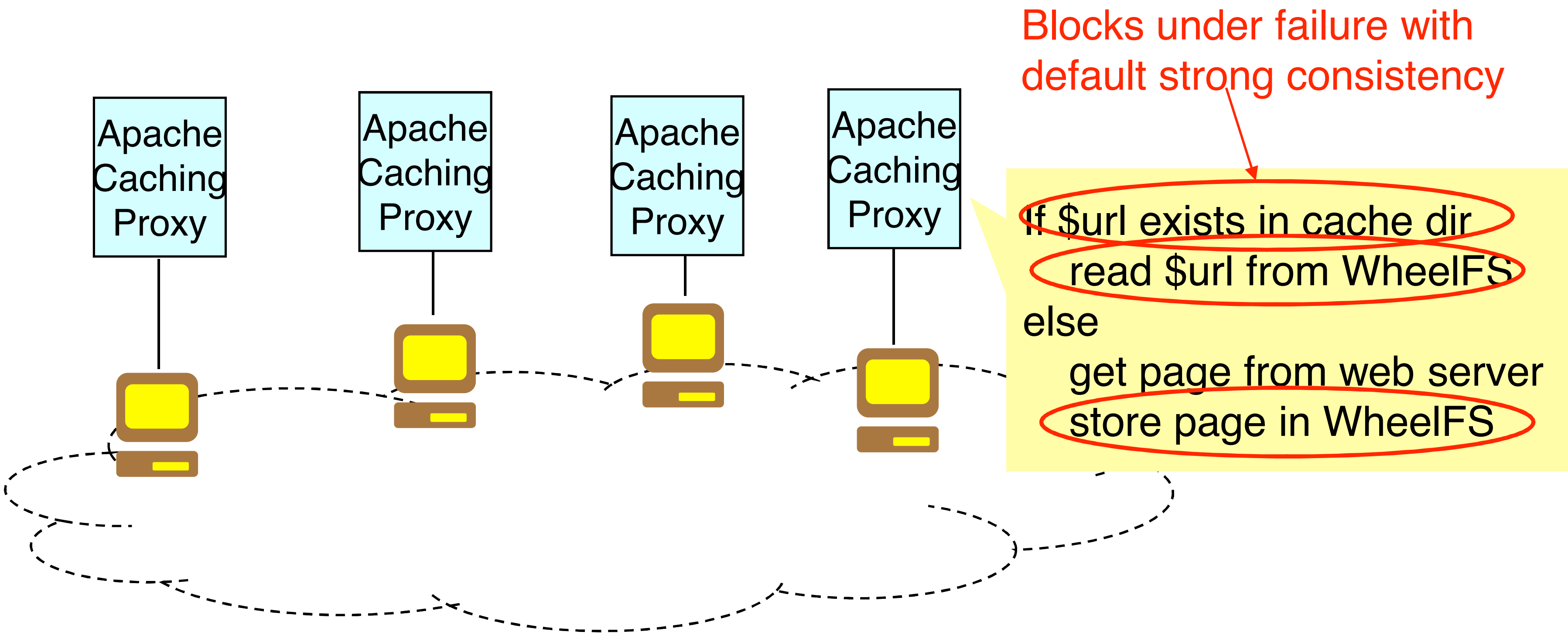
- Choose one of the replicas to win

→ Tradeoff: May lose some writes

(No application involvement)



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

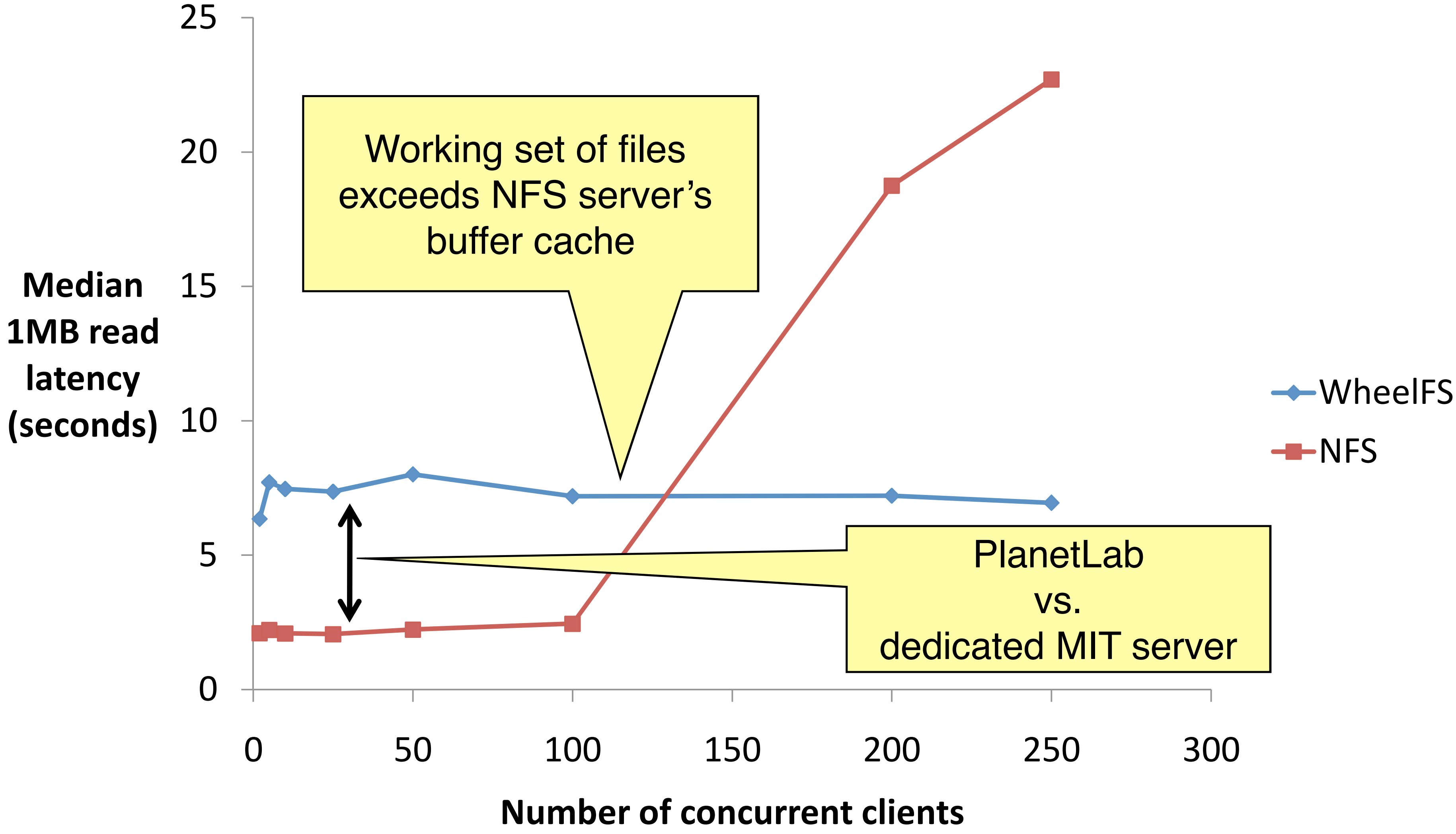
Applications Evaluation

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



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

Thoughts

- Is it:
  - really good?
  - really trivial?
- Similarities to self-certifying pathnames?
  - it's all about the interface
  - but this means only legacy apps benefit

PADS: Policy Architecture for Distributed Storage Systems

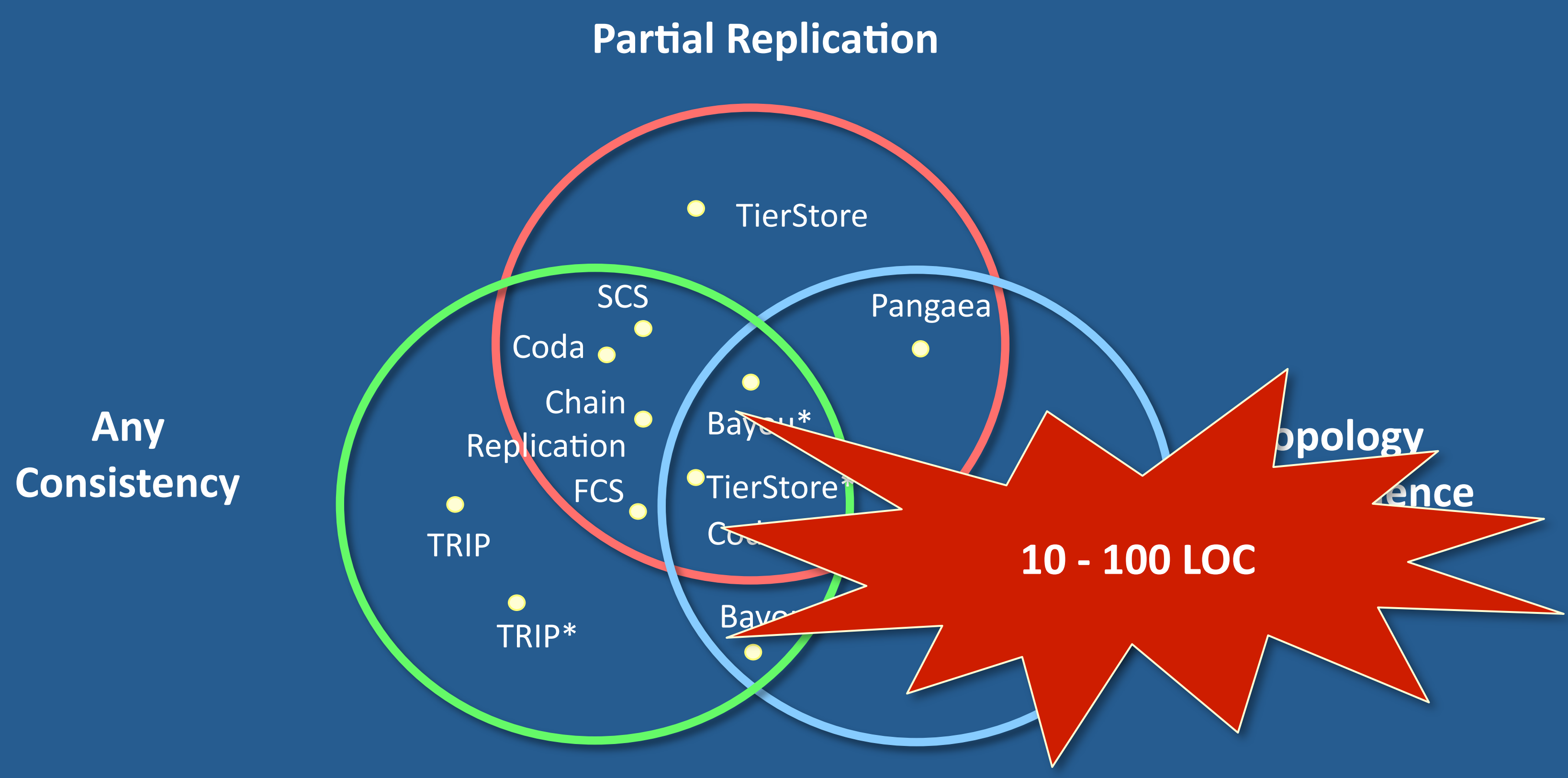
Nalini Belaramani, Jiandan Zheng, Amol Nayate,
Robert Soulé, Mike Dahlin and Robert Grimm.

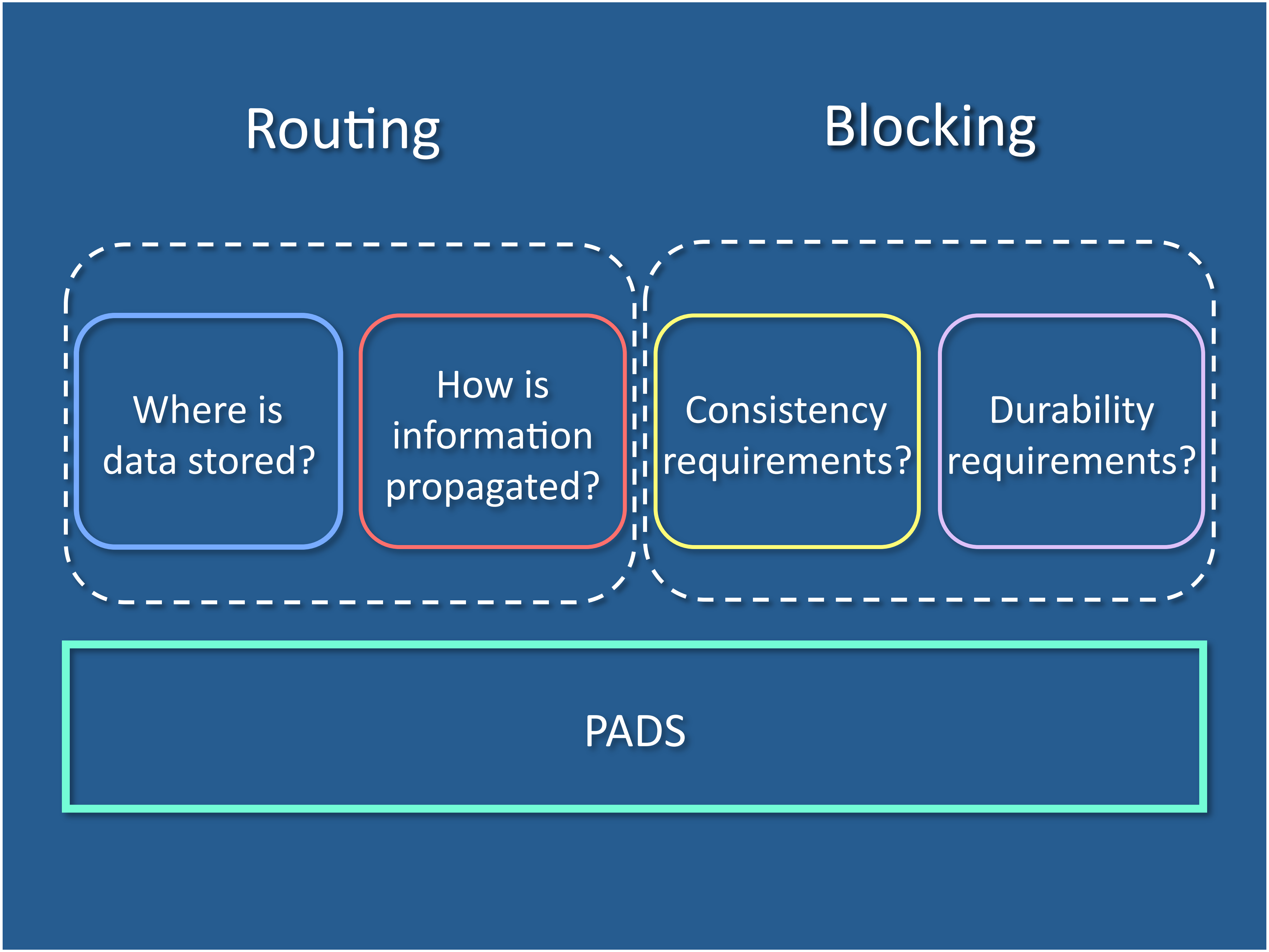
University of Texas at Austin, Amazon Inc.,
IBM T.J. Watson, New York University

Yes it is!

With PADS:

2 grad students + 4 months = 12 diverse systems




Routing
Blocking
Where is data stored?
How is information propagated?
Consistency requirements?
Durability requirements?
PADS

Outline

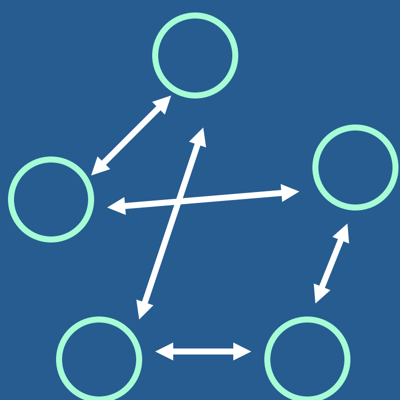
- PADS approach
- Policy
  - Routing
  - Blocking
- Evaluation

Routing

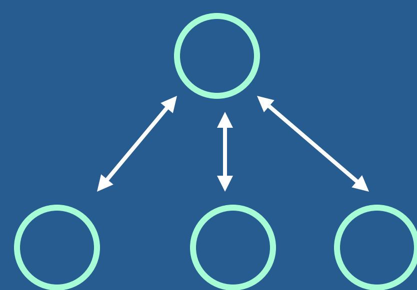
Data flows among nodes

When and where to send an update?

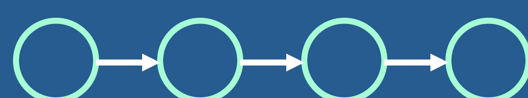
Who to contact on a local read miss?



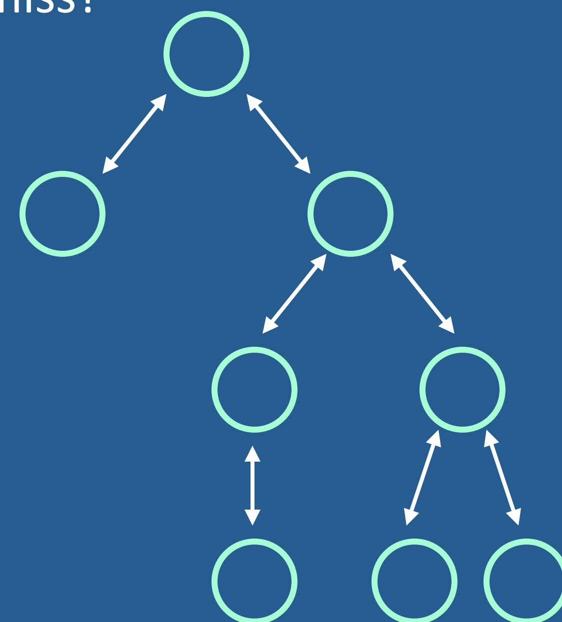
Bayou



Coda



Chain Replication



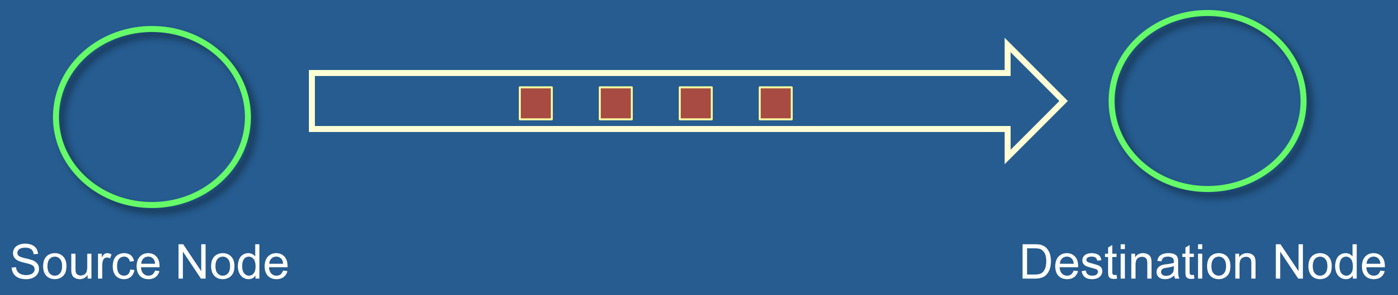
TierStore

Subscription

Primitive for update flow

Options:

- Data set of interest (e.g. /vol1/*)
- Notifications (invalidations) in causal order or updates (bodies)
- Logical start time



Routing Actions

Routing Actions	
Add Inval Sub	srcId, destId, objS, [startTime], LOG|CP|CP+Body
Add Body Sub	srcId, destId, objS, [startTime]
Remove Inval Sub	srcId, destId, objS
Remove Body Sub	srcId, destId, objS
Send Body	srcId, destId, objId, off, len, writerId, time
Assign Seq	objId, off, len, writerId, time
B Action	<policy defined>

Event-driven API

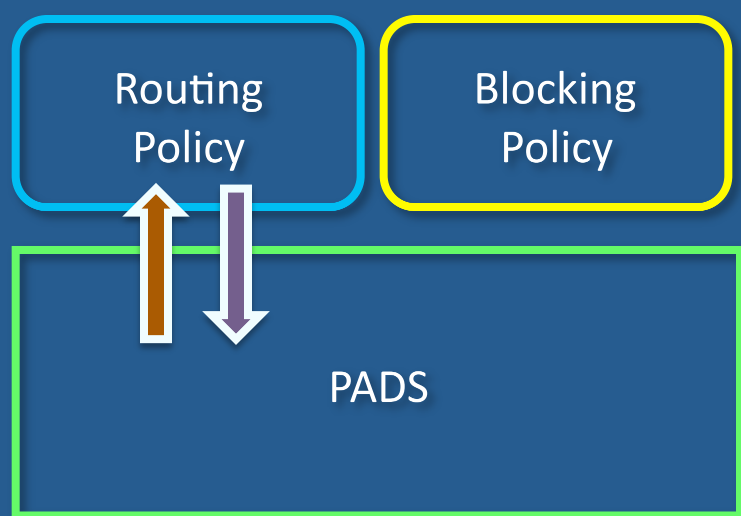
To set up routing

Events

Operation block
Write
Delete

Inval arrived
Send body succ
Send body failed

Subscription start
Subscription caught-up
Subscription end



Actions

Add inval sub
Add body sub

Remove inval sub
Remove body sub

Send body
Assign seq

B_action

Triggers from Routing API

Local Read/Write Triggers	
Operation block	obj, off, len, blocking_point, failed_predicates
Write	obj, off, len, writerId, time
Delete	obj, writerId, time

Message Arrival Triggers	
Inval arrives	srcId, obj, off, len, writerId, time
Send body success	srcId, obj, off, len, writerId, time
Send body failed	srcId, destId, obj, off, len, writerId, time

Connection Triggers	
Subscription start	srcId, destId, objS, Inval|Body
Subscription caught-up	srcId, destId, objS, Inval
Subscription end	srcId, destId, objS, Reason, Inval|Body

Domain-specific language

To specify routing

- R/Overlog
  - Routing language based on Overlog[*]
  - declarative rules fired by events
- Policy written as rules
  - invoke actions when events received

[*] "Implementing Declarative Overlays". Boon Thau Loo, Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Timothy Roscoe, Ion Stoica. SOSP 2005.

Blocking policy

Is it safe to access local data?

Consistency

What version of data can be accessed?

Durability

Whether updates have propagated to safe locations?

Block until semantics guaranteed

How to specify blocking policy?

Where to block?

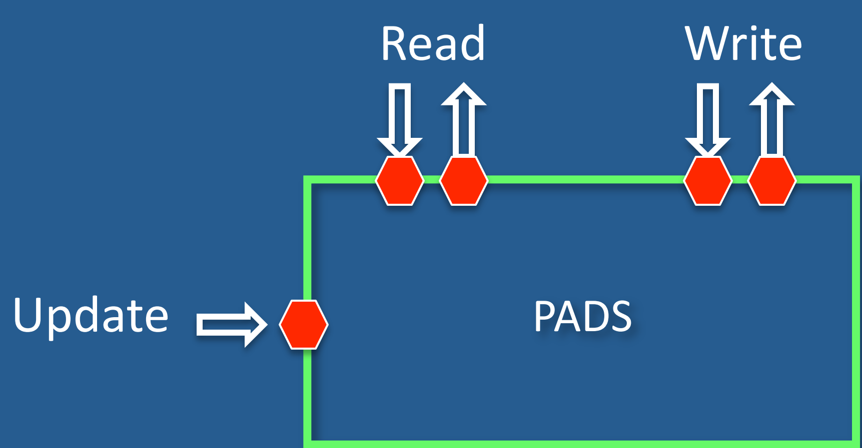
- At data access points

What to specify?

- List of conditions

PADS provides

- 4 built-in conditions (local bookkeeping)
- 1 extensible condition




```
Is valid
Is causal
Is sequenced
Max staleness
R_Msg
```

Blocking Predicates

Predefined Conditions on Local Consistency State	
isValid	Block until node has received the body corresponding to the highest received invalidation for the target object
isComplete	Block until object's consistency state reflects all updates before the node's current logical time
isSequenced	Block until object's total order is established
maxStaleness *nodes, count, t*	Block until all writes up to *(operationStartTime-t)* from *count* nodes in *nodes* have been received.
User Defined Conditions on Local or Distributed State	
B_Action *event-spec*	Block until an event with fields matching *event-spec* is received from routing policy

Blocking policy examples

Consistency:

- Read only causal data

  `Read at block: Is_causal`

Durability:

- Block write until update reaches server

  `Write after block : R_Msg (ackFromServer)`

Is PADS a better way to build distributed storage systems?

- General enough?
- Easy to use?
- Easy to adapt
- Overheads?

General enough?

	SCS	FCS	Coda	TRIP	Tier Store	Chain Repl	Bayou	Pangaea
Topology	Client/ Server	Client/ Server	Client/ Server	Client/ Server	Tree	Chains	Ad-Hoc	Ad-Hoc
Replication	Partial	Partial	Partial	Full	Partial	Full	Full	Partial
Demand caching	✓	✓	✓	✓				
Cooperative caching		✓						
Prefetching			✓	✓	✓	✓	✓	✓
Consistency	Sequential	Sequential	Open/ Close	Sequential	Mono-Reads	Linearizable	Causal	Mono-Reads
Callbacks	✓	✓	✓					
Leases	✓	✓	✓					
Disconnected operation			✓	✓	✓	✓		✓
Inval v. update progagation	Inval	Inval	Inval	Inval	Update	Update	Update	Update

Easy to use?

System	Routing Rules	Blocking Conditions
P-Bayou	9	3
P-Bayou*	9	3
P-Chain Rep	75	5
P-Coda	31	5
P-Coda*	44	5
P-FCS	43	6
P-Pangaea	75	1
P-TierStore	14	1
P-TierStore*	29	1
P-TRIP	6	3
P-TRIP*	6	3

Thoughts

- Kind of “PRACTI: The Next Generation”

Real question:

- How expressive is it?
- Did they
  - start w/ the 12 systems and define the API
  - or the reverse?