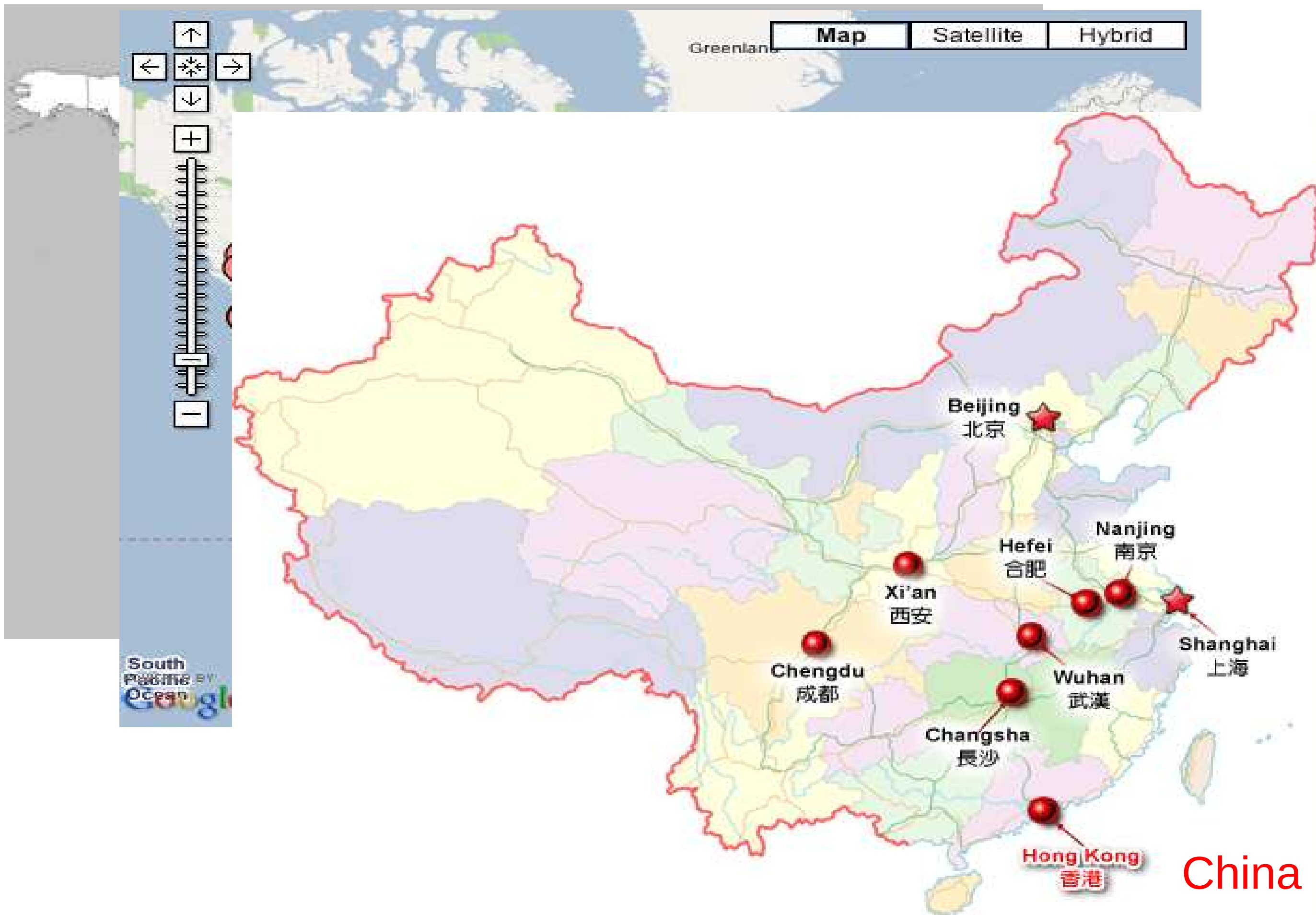
Simplifying Wide-Area Application Development with WheelFS

Jeremy Stribling

In collaboration with Jinyang Li, Frans Kaashoek, Robert Morris

MIT CSAIL & New York University

Resources Spread over Wide-Area Net



China grid

Grid Computations Share Data

Nodes in a distributed computation share:

- Program binaries
- Initial input data
- Processed output from one node as intermediary input to another node

So Do Users and Distributed Apps

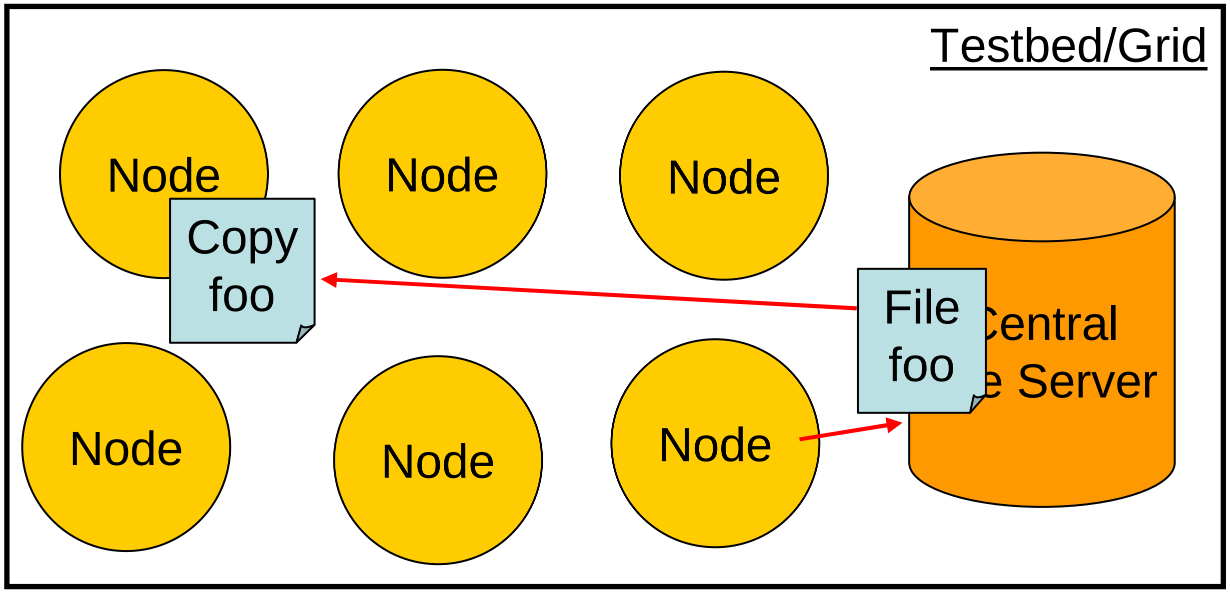
- Apps aggregate disk/computing at hundreds of nodes
- Example apps
  - Content distribution networks (CDNs)
  - Backends for web services
  - Distributed digital research library

All applications need distributed storage

State of the Art in Wide-Area Storage

- Existing wide-area file systems are inadequate
  - Designed only to store files for users
  - *E.g.,* No hundreds of nodes can write files to the same dir
  - *E.g.*, Strong consistency at the cost of availability
- Each app builds its own storage!
  - Distributed Hash Tables (DHTs)
  - ftp, scp, wget, *etc*.

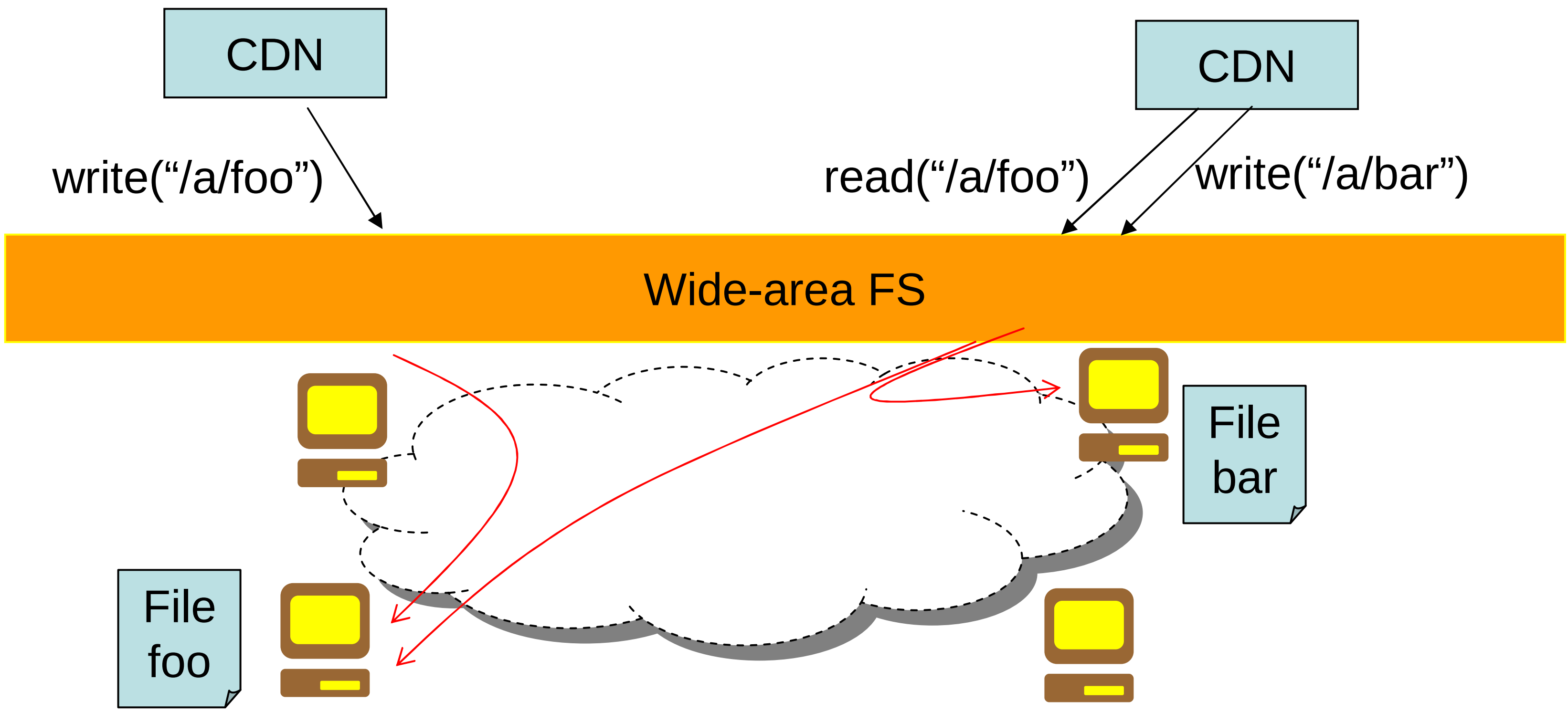
Current Solutions



Usual drawbacks:

- All data flows through one node
- File systems are too transparent
  - Mask failures
  - Incur long delays

If We Had a Good Wide-Area FS?



- FS makes building apps simpler

Why Is It Hard?

- Apps care a lot about performance
- WAN is often the bottleneck
  - High latency, low bandwidth
  - Transient failures
- How to give app control without sacrificing ease of programmability?

Our Contribution: WheelFS

- Suitable for wide-area apps
- Gives app control through *cues* over:
  - Consistency vs. availability tradeoffs
  - Data placement
  - Timing, reliability, *etc*.
- Prototype implementation

Talk Outline

- Challenges & our approach
- Basic design
- Application control
- Running a Grid computation over WheelFS

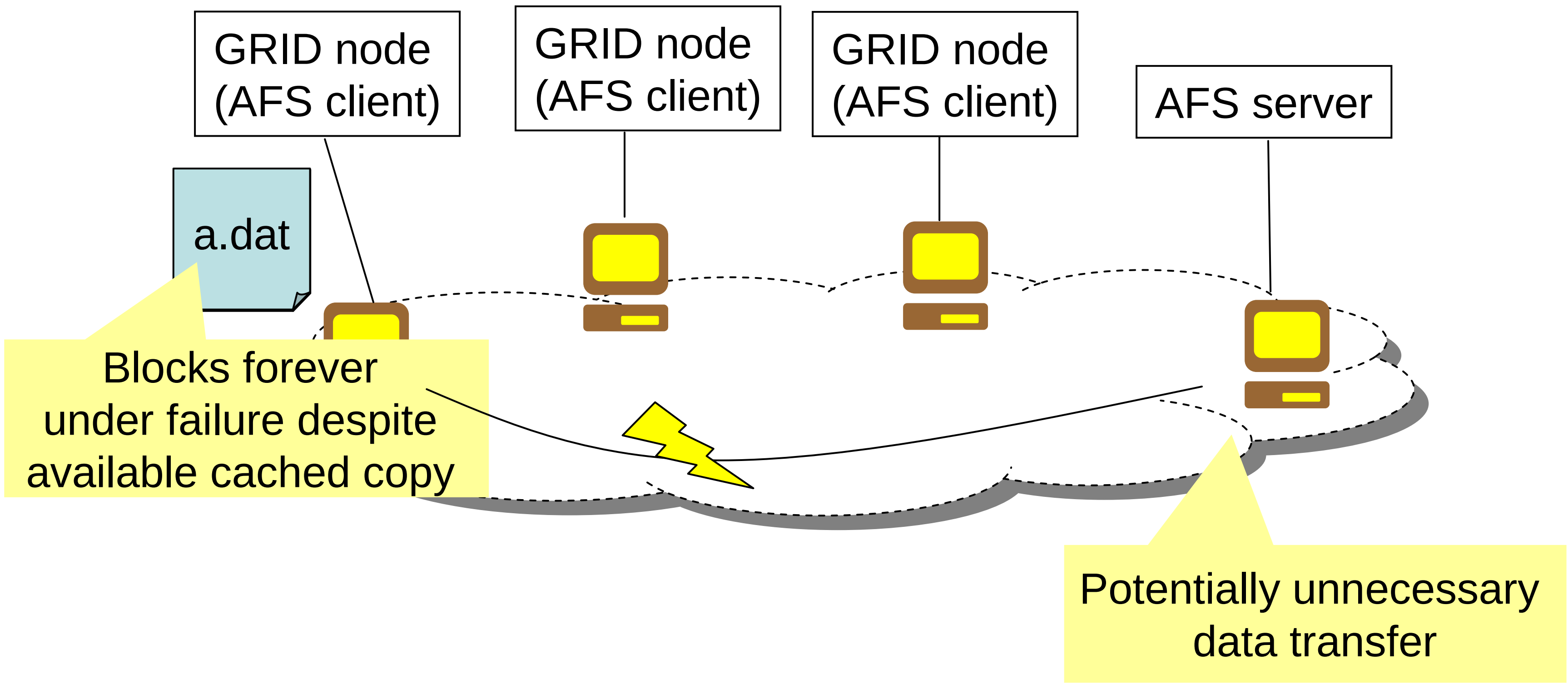
What Does a File System Buy You?

- Re-use existing software
- Simplify the construction of new applications
  - A hierarchical namespace
  - A familiar interface
  - Language-independent usage

Why Is It Hard To Build a FS on WAN?

- High latency, low bandwidth
  - 100s ms instead of 1s ms latency
  - 10s Mbps instead of 1000s Mbps bandwidth
- Transient failures are common
  - 32 outages over 64 hours across 132 paths [Andersen’01]

What If Grid Uses AFS over WAN?



Design Challenges

- High latency
  - Store data close to where it is needed
- Low wide-area bandwidth
  - Avoid wide-area communication if possible
- Transient failures are common
  - Cannot block *all* access during partial failures

Only applications have the needed information!

WheelFS Gives Apps Control

	AFS,NFS, GFS	WheelFS
Goal	Total network transparency	Application control
Can apps control how to handle failures?	X	✓
Can apps control data placement?	X	✓

WheelFS: Main Ideas

- Apps control
  - Apps embed *semantic cues* to inform FS about failure handling, data placement ...
- Good default policy
  - Write locally, strict consistency

Talk Outline

- Challenges & our approach
- Basic design
- Application control
- Running a Grid computation over WheelFS

File Systems 101

- Basic FS operations:
  - Name resolution: hierarchical name → flat id

```
open("/wfs/a/foo", …) → id: 1235
```

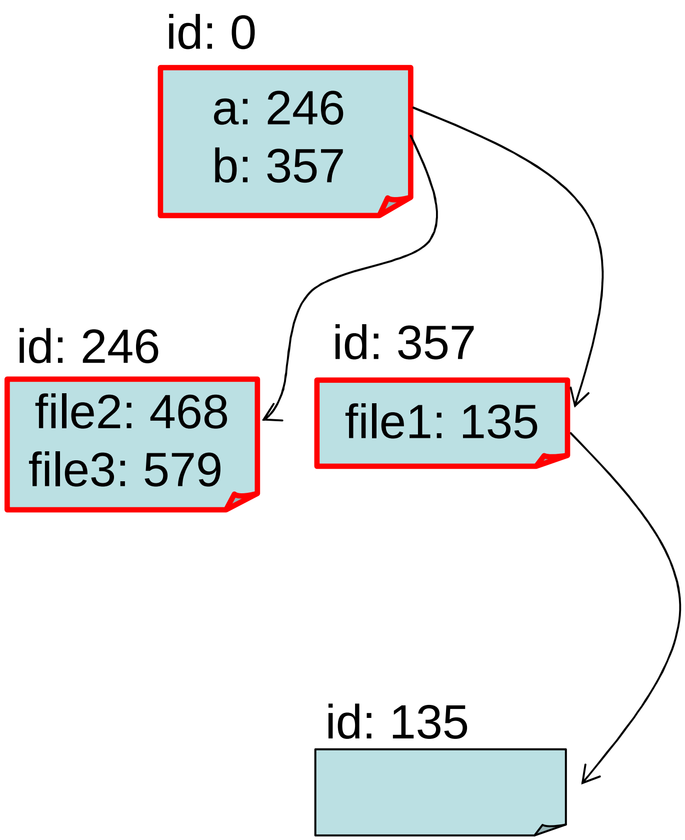
  - Data operations: read/write file data

```
read(1235, …)
write(1235, …)
```

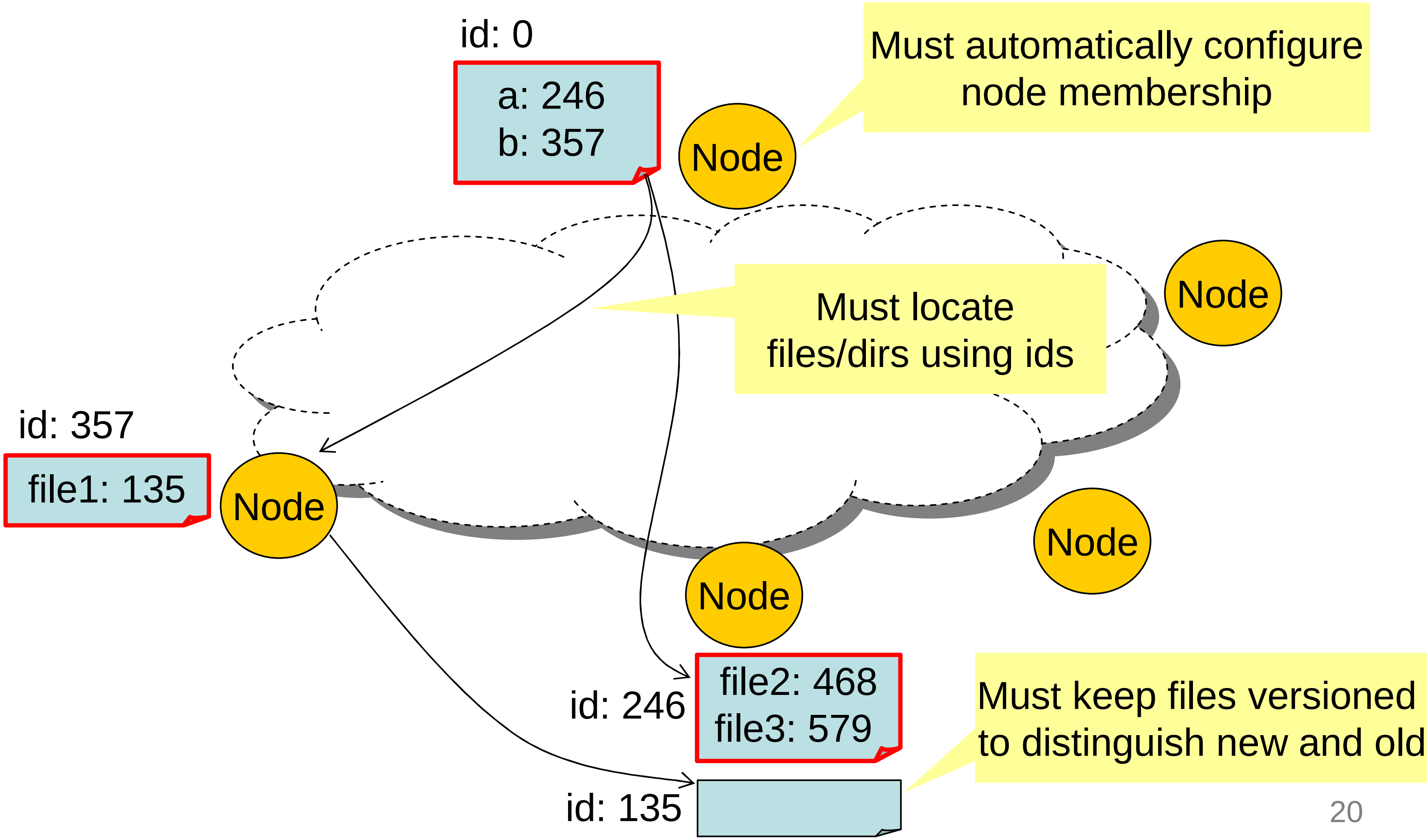
  - Namespace operations: add/remove files or dirs

```
mkdir("/wfs/b", …)
```

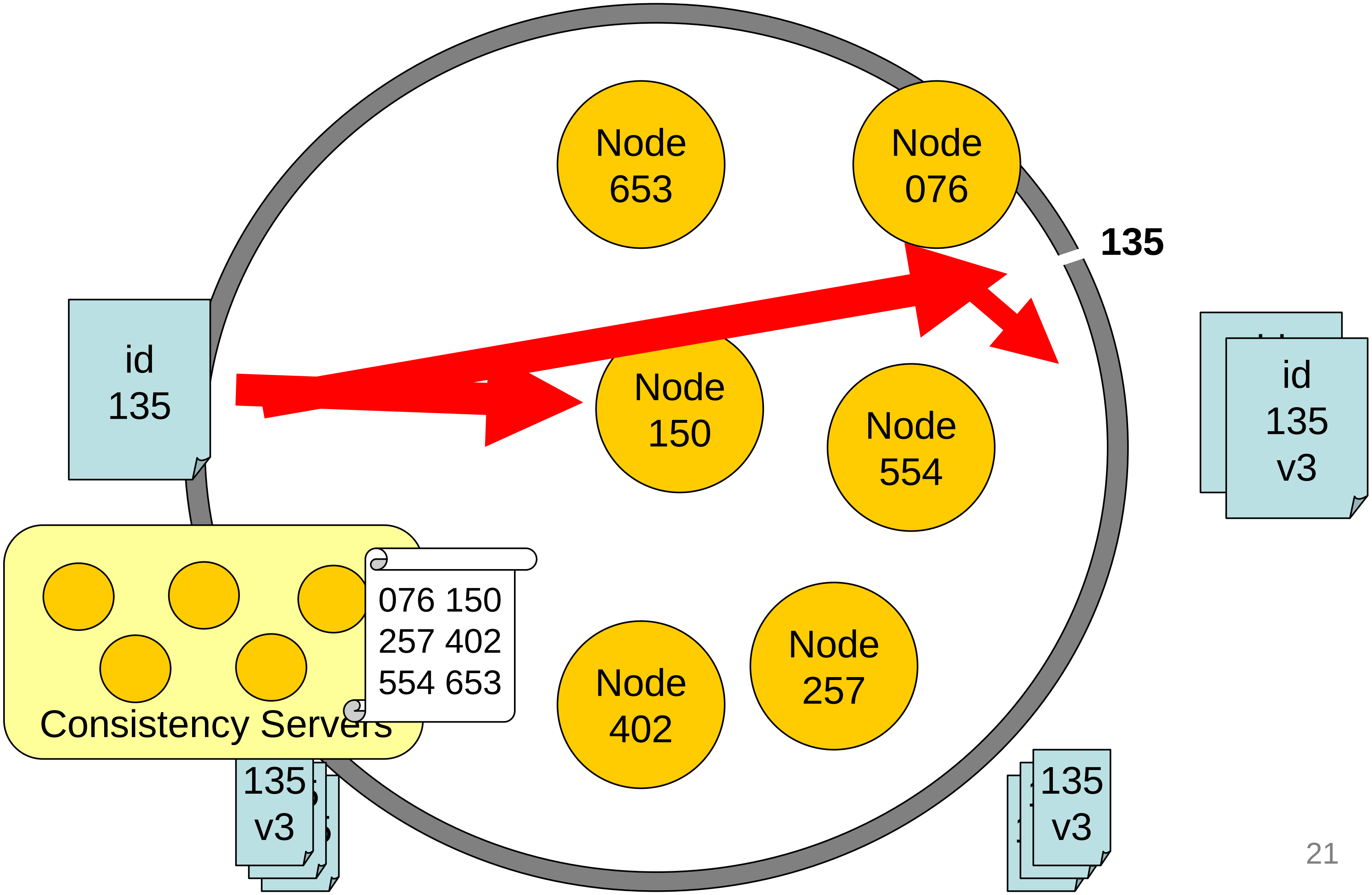
File Systems 101



Distribute a FS across nodes



Basic Design of WheelFS

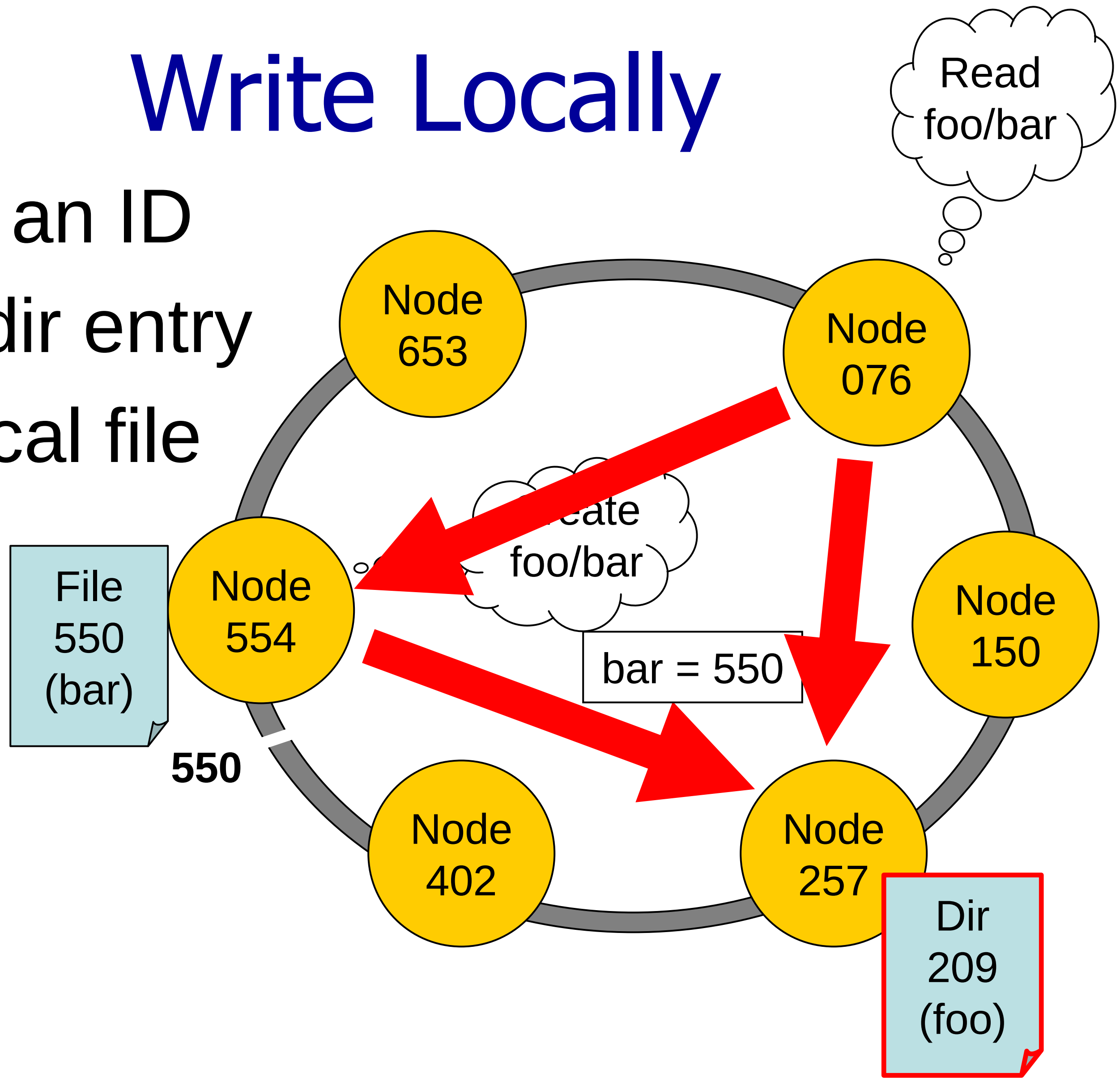


Default Behavior: Write Locally, Strict Consistency

- Write locally: Store newly created files at the writing node
  - Writes are fast
  - Transfer data lazily for reads when necessary
- Strict consistency: data behaves as in a local FS
  - Once new data is written and the file is closed, the next open will see the new data

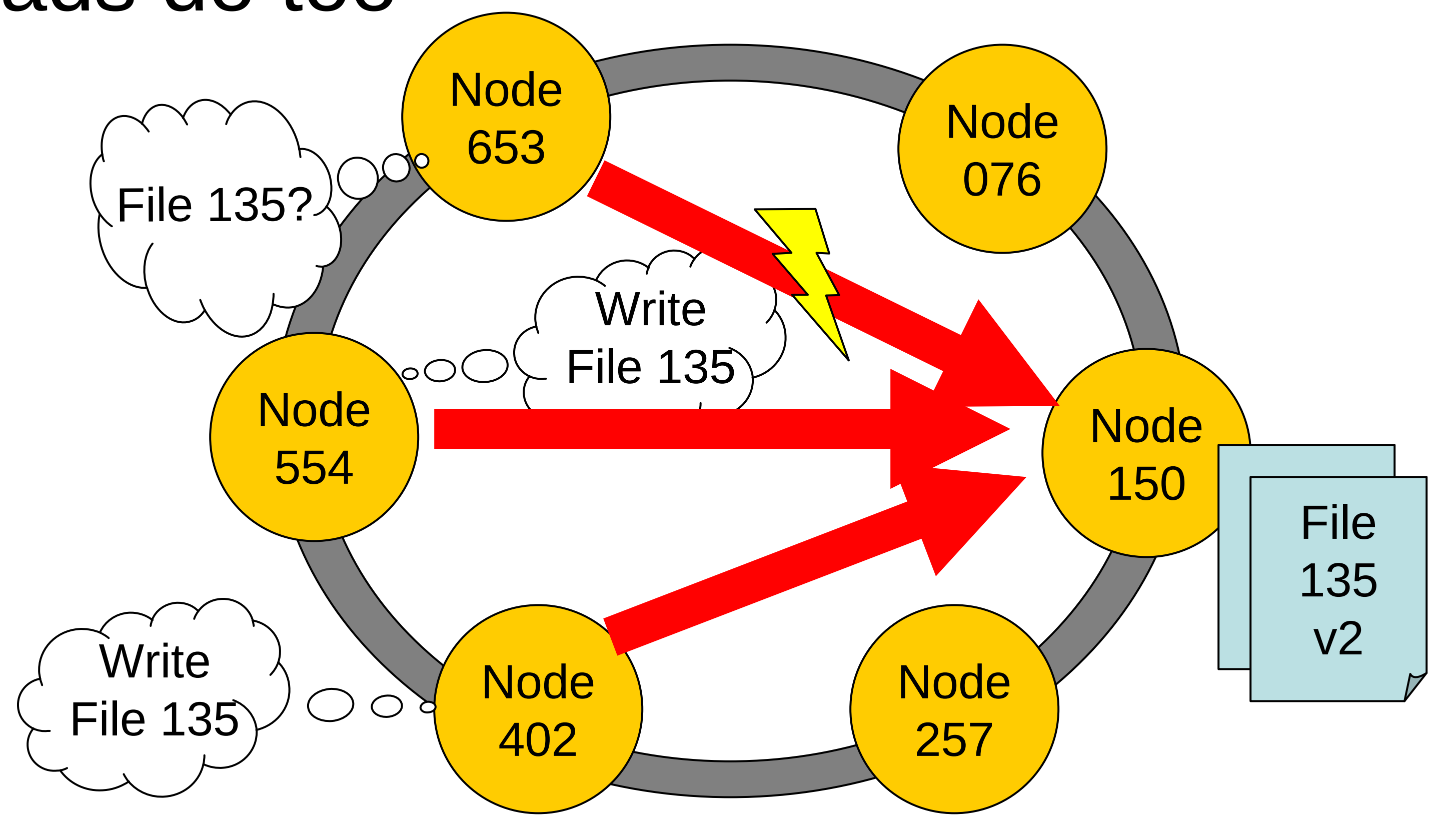
Write Locally

1. Choose an ID
2. Create dir entry
3. Write local file



Strict Consistency

- All writes go to the same node
- All reads do too



Talk Outline

- Challenges & our approach
- Basic design
- Application control
- Running a Grid computation over WheelFS

WheelFS Gives Apps Control with Cues

- Apps want to control consistency, data placement ...
- How? Embed cues in path names
  - Flexible and minimal interface change

*/wfs/cache/**.cue**/a/b/*

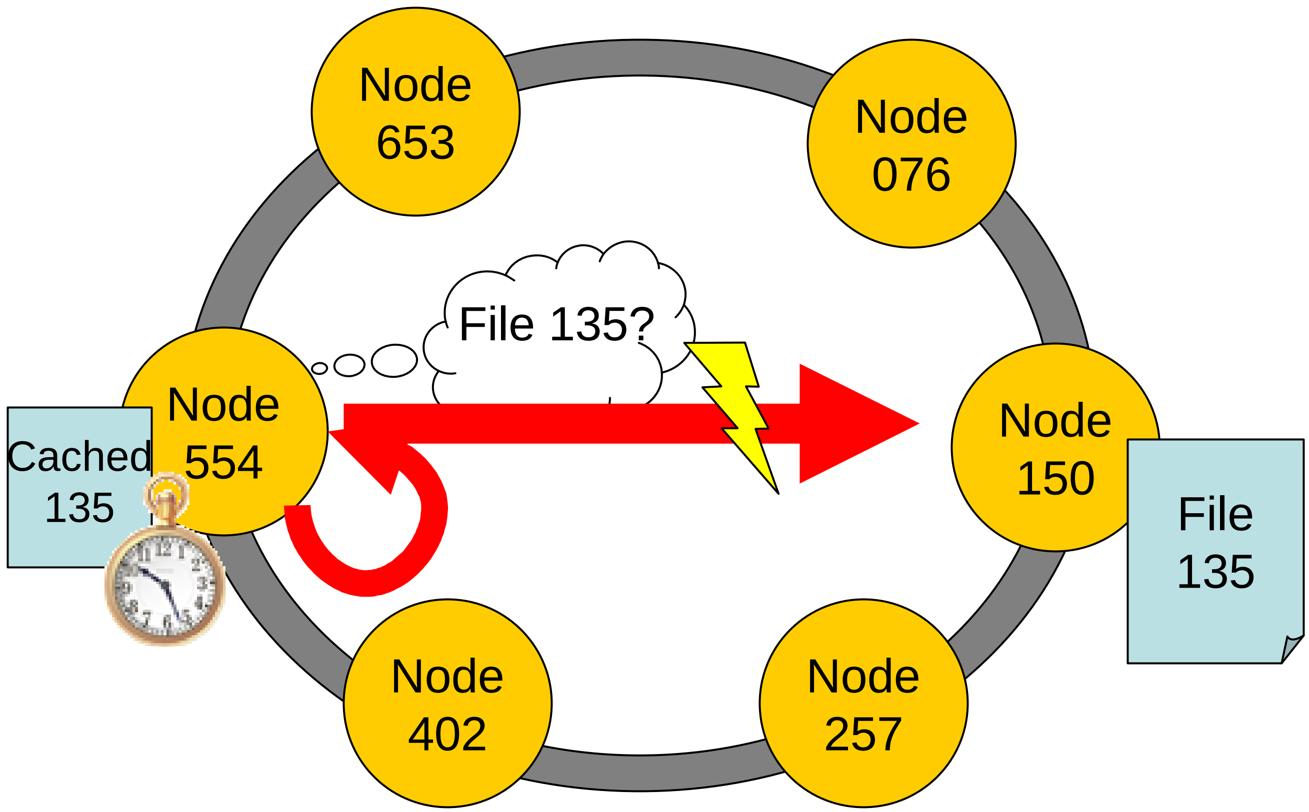
*/wfs/cache/a/b/**.cue**/foo*

Coarse-grained:
Cues apply recursively over an entire subtree of files

Fine-grained:
Cues can apply to a single file

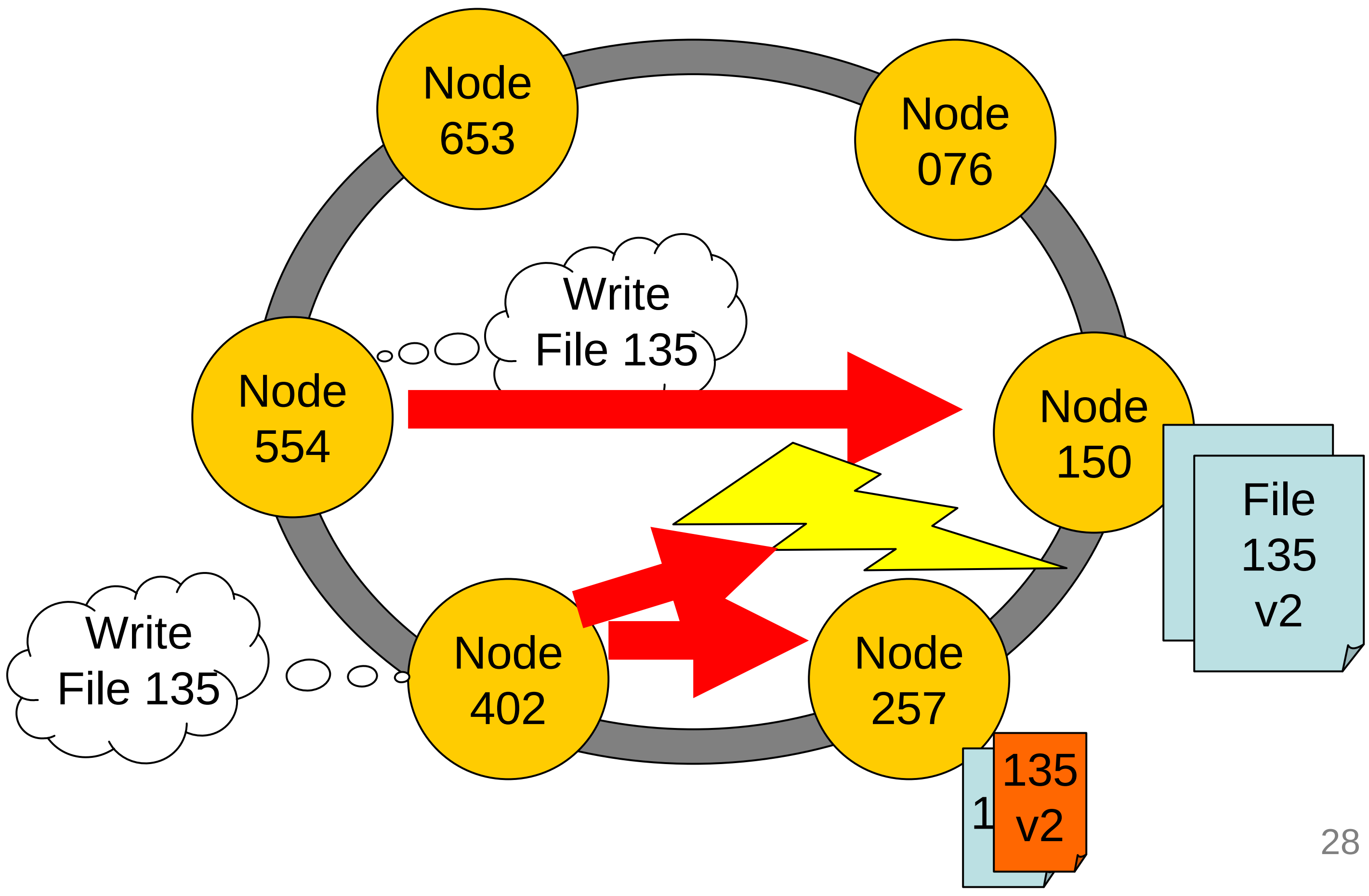
Eventual Consistency: Reads

- Read *any* version of the file you can find
- In a given time limit



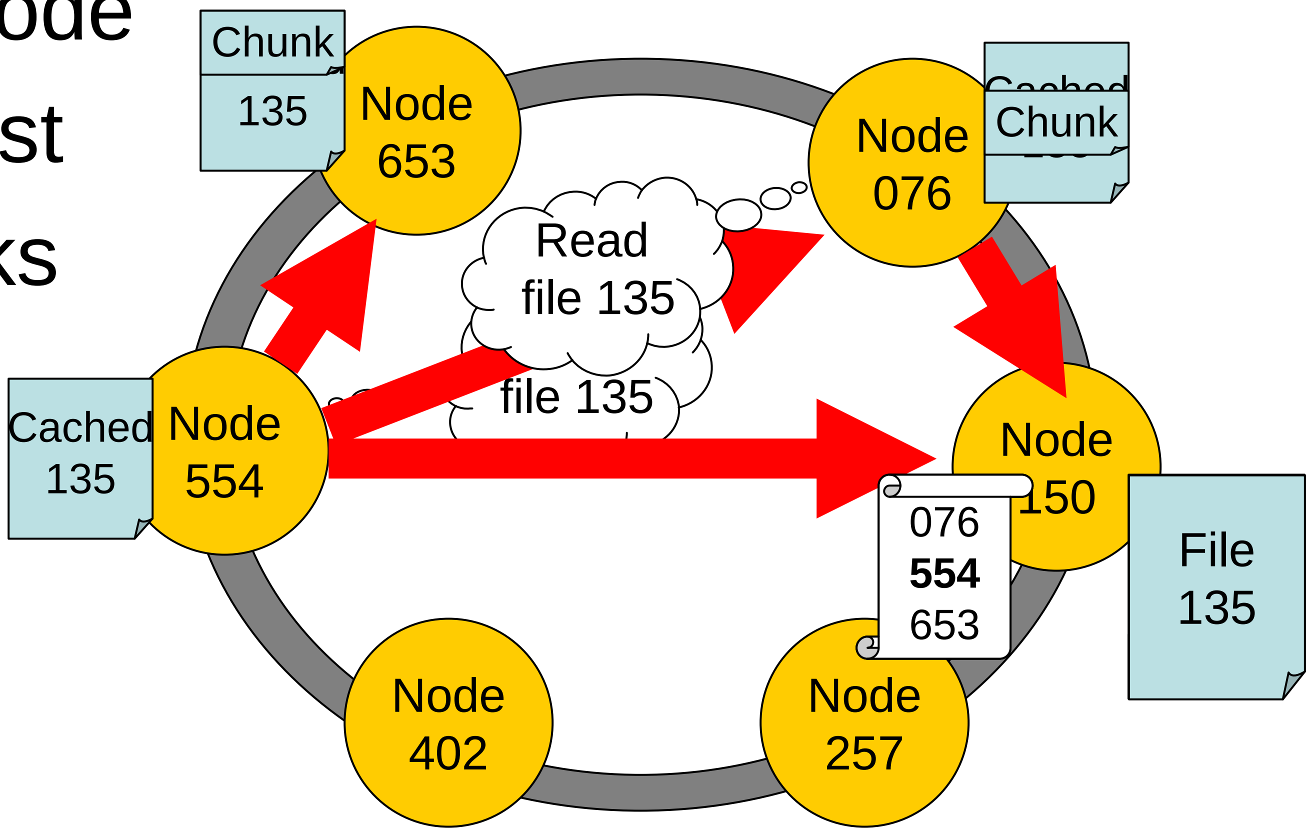
Eventual Consistency: Writes

- Write to any replica of the file



Handle Read Hotspots

1. Contact node
2. Receive list
3. Get chunks



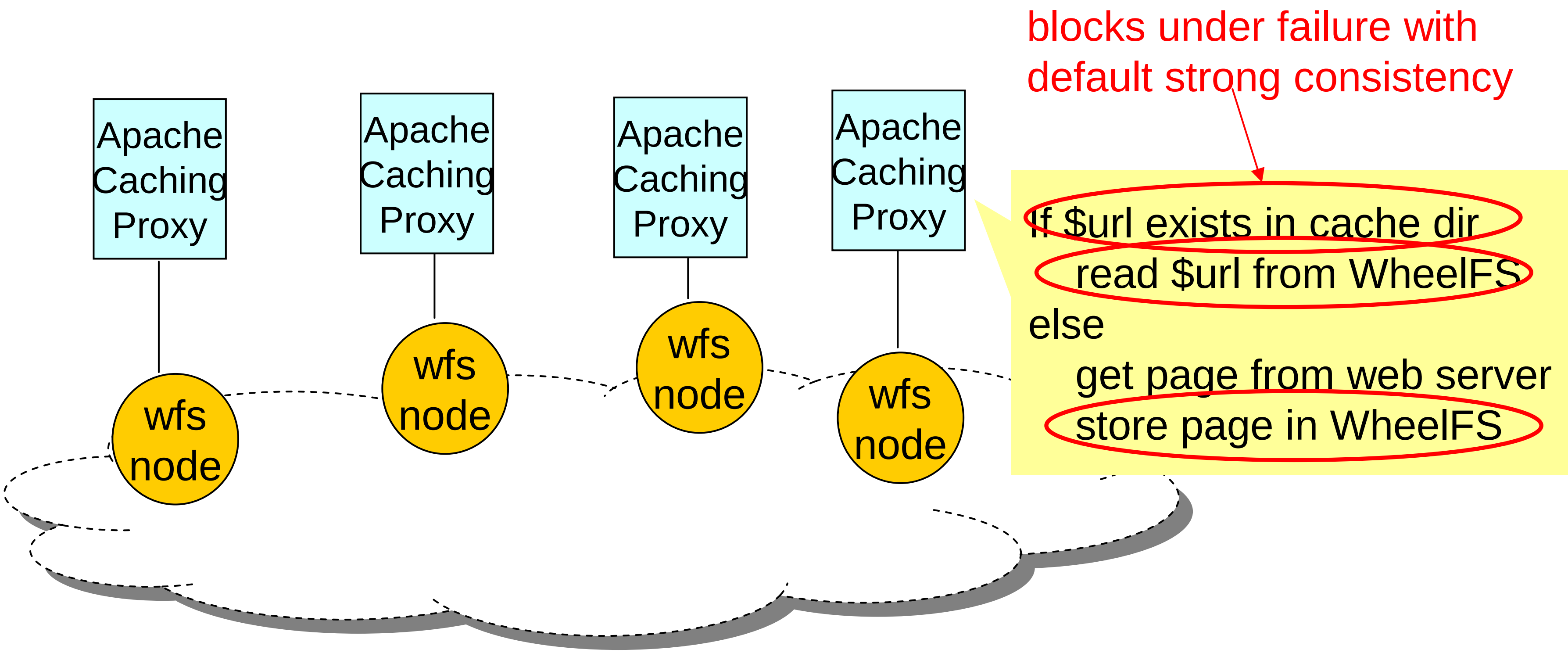
WheelFS Cues

	Name	Purpose
Control consistency	**Eventual-Consistency**	Control whether reads must see fresh data, and whether writes must be serialized
Control consistency	**MaxTime=**	Specify time limit for operations
Large reads	**HotSpot**	This file will be read simultaneously by many nodes, so use p2p caching
Hint about data placement	**Site**=, **Node**=	Hint which node or group of nodes a file should be stored

Other types of cues: Durability, system information, *etc*.

Example Use of Cues: Content Distribution Networks

- CDNs prefer availability over consistency



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

Example Use of Cues: CDN

- 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 /.HotSpot**

Tells WheelFS to read a cached file even when the corresponding file server cannot be contacted

Tells WheelFS to write the file data anywhere even when the corresponding file server cannot be contacted

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

Example Use of Cues: BLAST Grid Computation

- DNA alignment tool run on Grids
- Copy separate DB portions and queries to many nodes
- Run separate computations
- Later fetch and combine results
- Read binary using .HotSpot
- Write output using .EventualConsistency

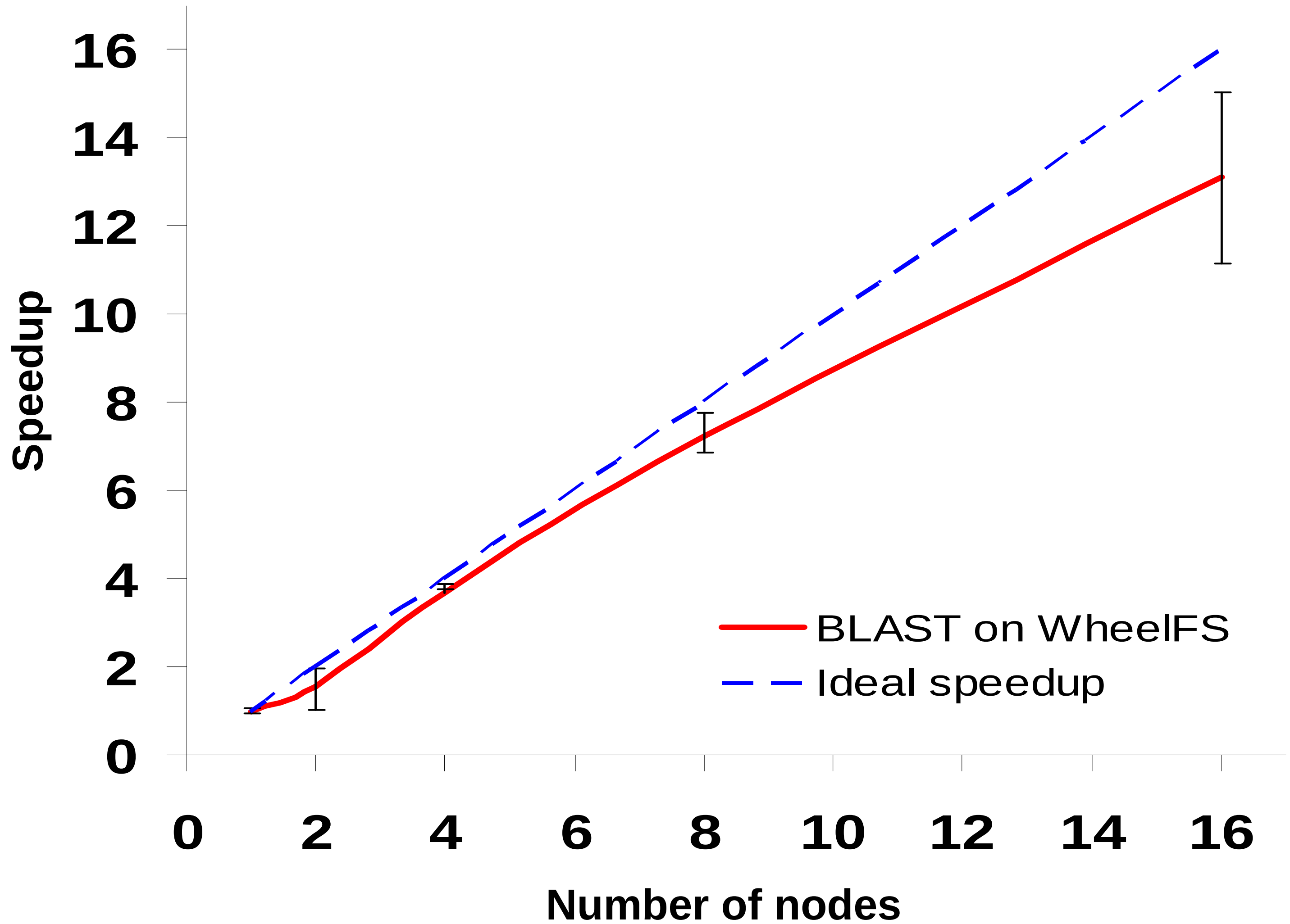
Talk Outline

- Challenges & our approach
- Basic design
- Application control
- Running a Grid computation over WheelFS

Experiment Setup

- Up to 16 nodes run WheelFS on Emulab
  - 100Mbps access links
  - 12ms delay
  - 3 GHz CPUs
- “nr” protein database (673 MB), 16 partitions
- 19 queries sent to all nodes

BLAST Achieves Near-Ideal Speedup on WheelFS



Related Work

- Cluster FS: Farsite, GFS, xFS, Ceph
- Wide-area FS: JetFile, CFS, Shark
- Grid: LegionFS, GridFTP, IBP
- POSIX I/O High Performance Computing Extensions

Conclusion

- A WAN FS simplifies app construction
- FS must let app control data placement & consistency
- WheelFS exposes such control via cues