Building Distributed, Wide-Area Applications with WheelFS

Jeremy Stribling, Emil Sit,
Frans Kaashoek, Jinyang Li, and Robert Morris

MIT CSAIL and NYU

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

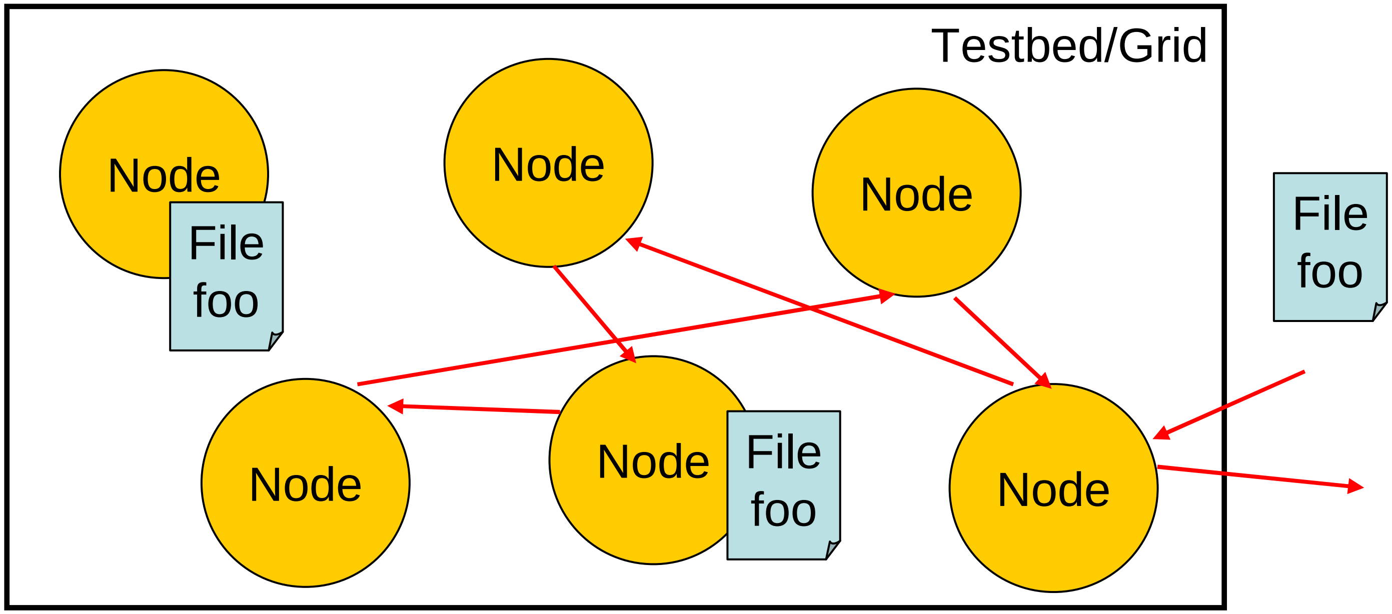
So Do Users and Distributed Apps

- Shared home directory for testbeds (*e.g.*, PlanetLab, RON)
- Distributed apps reinvent the wheel:
  - Distributed digital research library
  - Wide-area measurement experiments
  - Cooperative web cache

- Can we invent a shared data layer once?

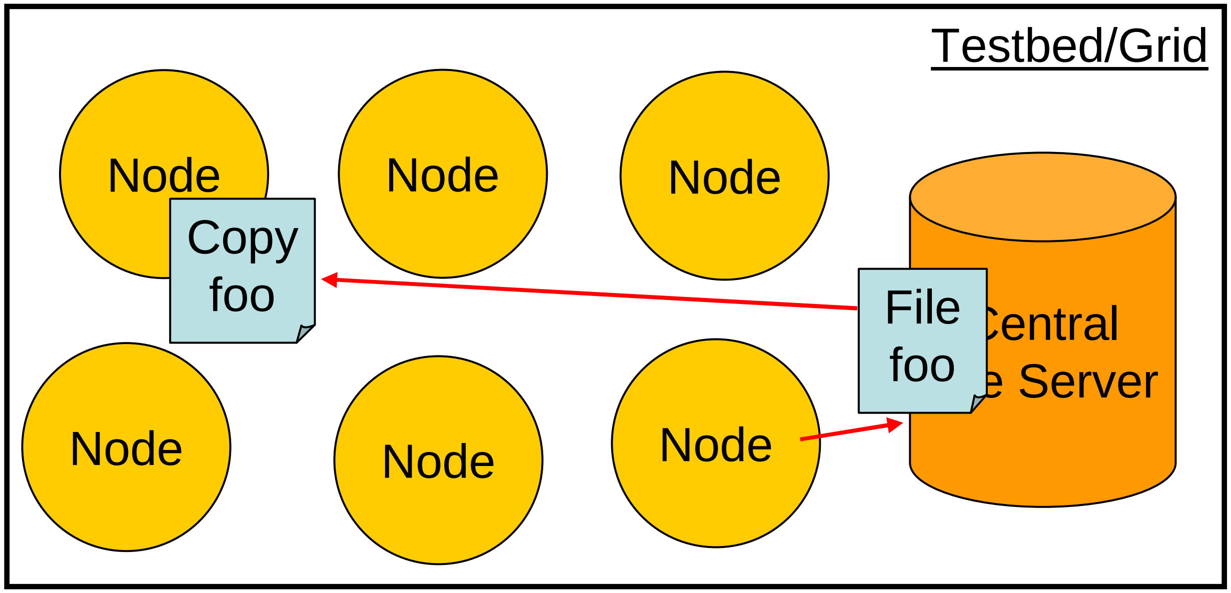
Our Goal

- Distributed file system for testbeds/Grids



- App can share data between nodes
- Users can easily access data
- Simple-to-build distributed apps

Current Solutions



Usual drawbacks:

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

Our Proposal: WheelFS

- A decentralized, wide-area FS
- Main contributions:

  1) Provide good performance according to *Read Globally, Write Locally*

  2) Give apps control with *semantic cues*

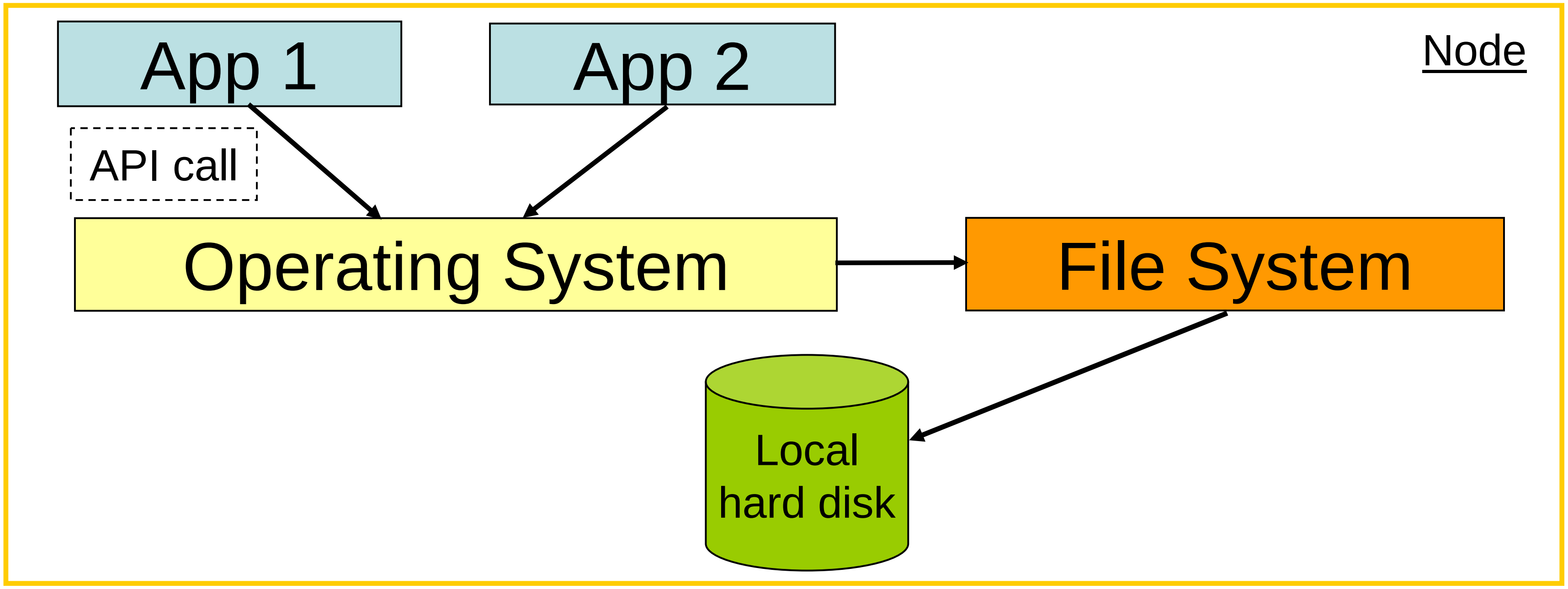
Talk Outline

1. How to decentralize your file system
2. How to control your files

What Does a File System Buy You?

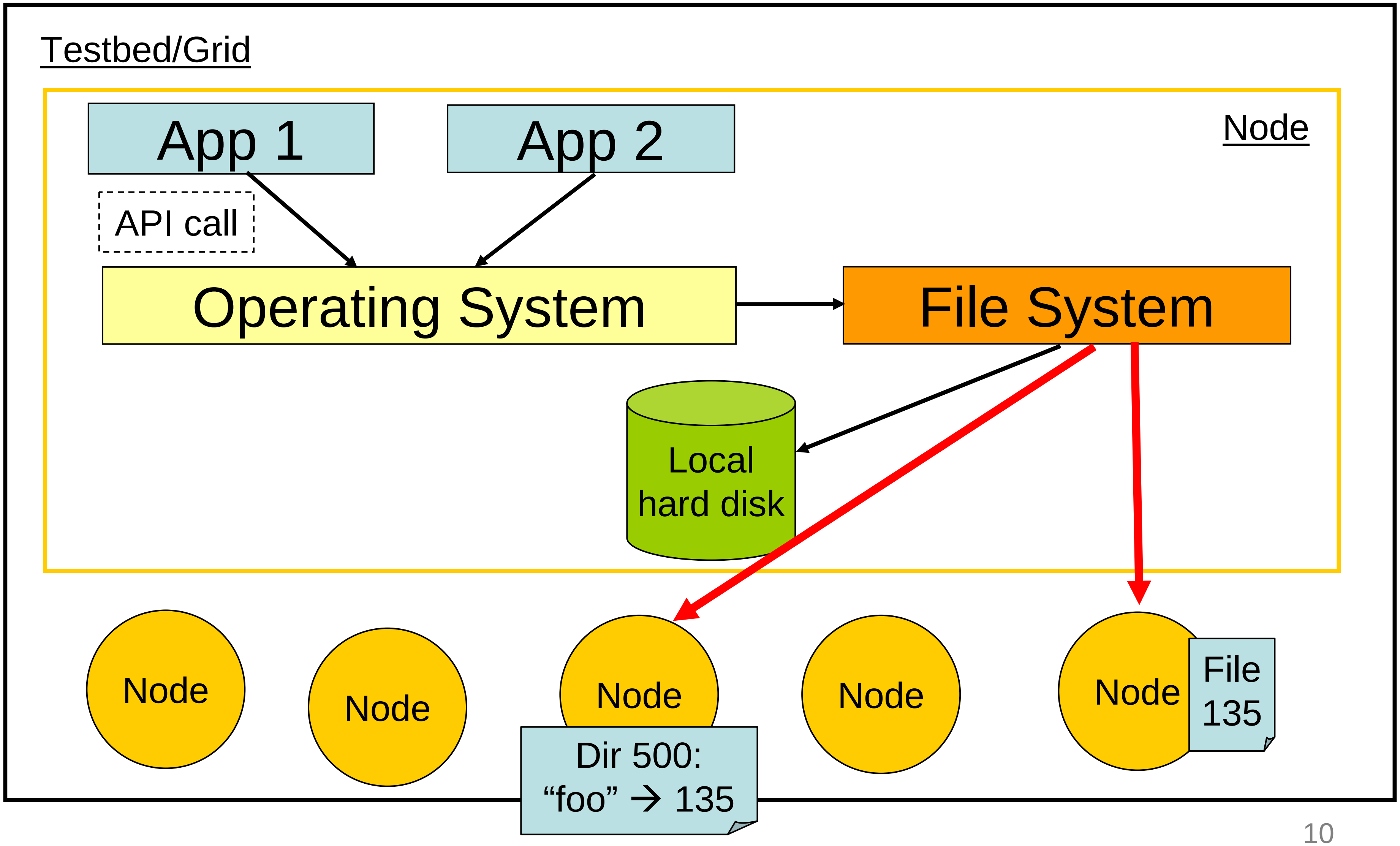
- A familiar interface
- Language-independent usage model
- Hierarchical namespace useful for apps
- Quick-prototyping for apps

File Systems 101

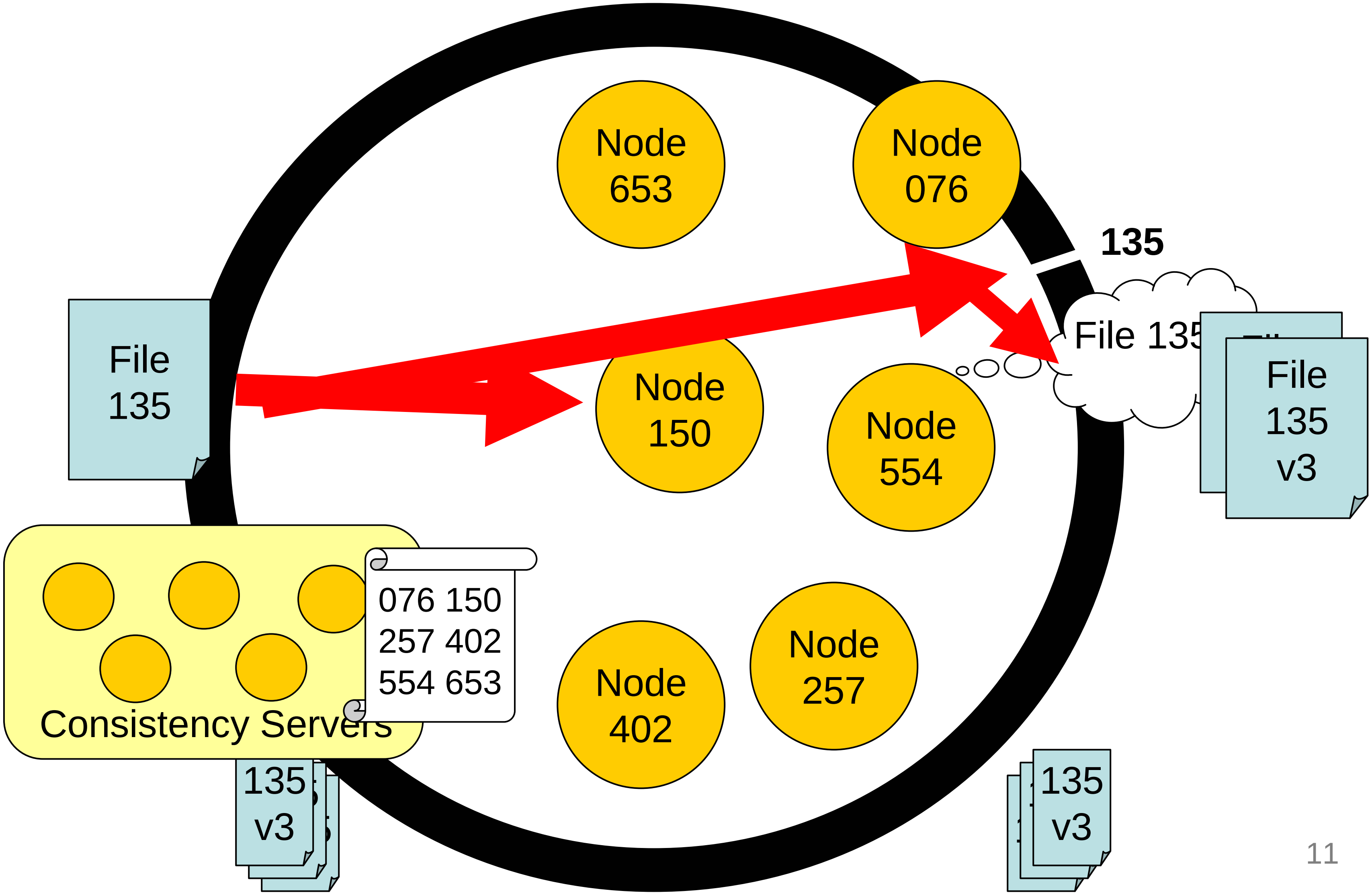


- File system (FS) API:
  - Open <*filename*> → <*file_id*>
  - {Close/Read/Write} <*file_id*>
- Directories translate file names to IDs

Distributed File Systems



Basic Design of WheelFS

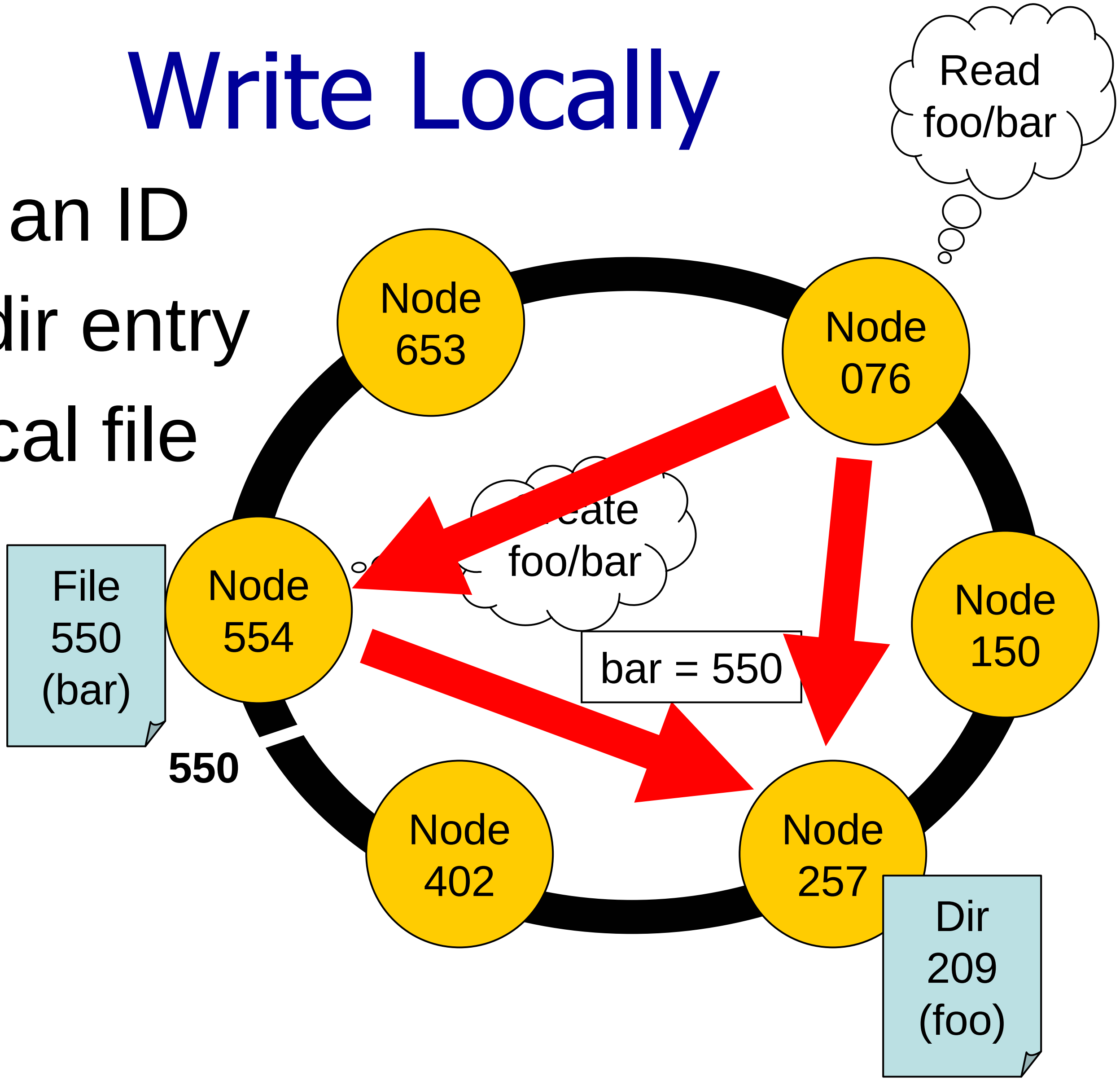


Read Globally, Write Locally

- Perform writes at local disk speeds
- Efficient bulk data transfer
- Avoid overloading nodes w/ popular files

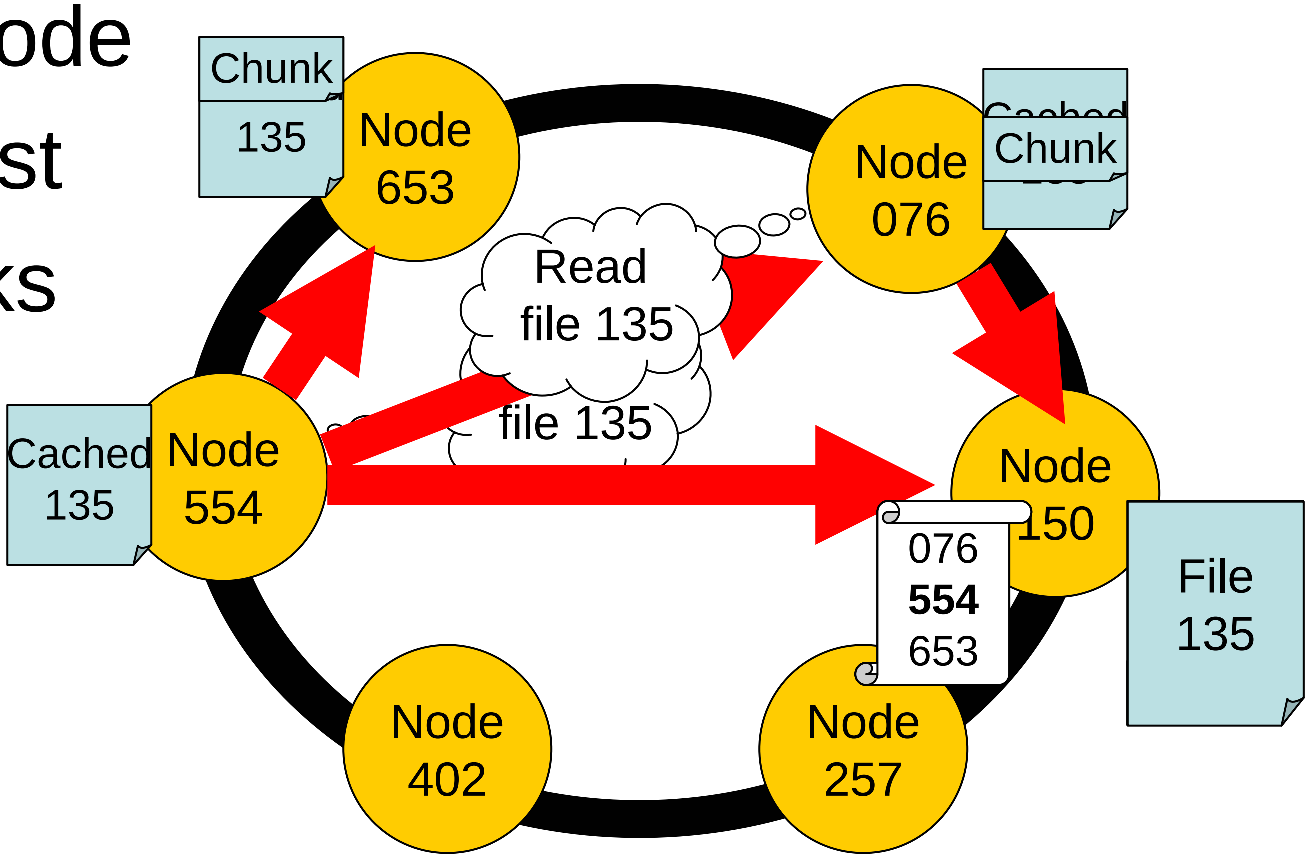
Write Locally

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



Read Globally

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



Example: BLAST

- DNA alignment tool run on Grids
- Copy separate DB portions and queries to many nodes
- Run separate computations
- Later fetch and combine results

Example: BLAST

- With WheelFS, however:
  - No explicit DB copying necessary
  - Efficient initial DB transfers
  - Automatic caching for reused DBs and queries
- Could be better since data is never updated

Example: Cooperative Web Cache

Collection of nodes that:

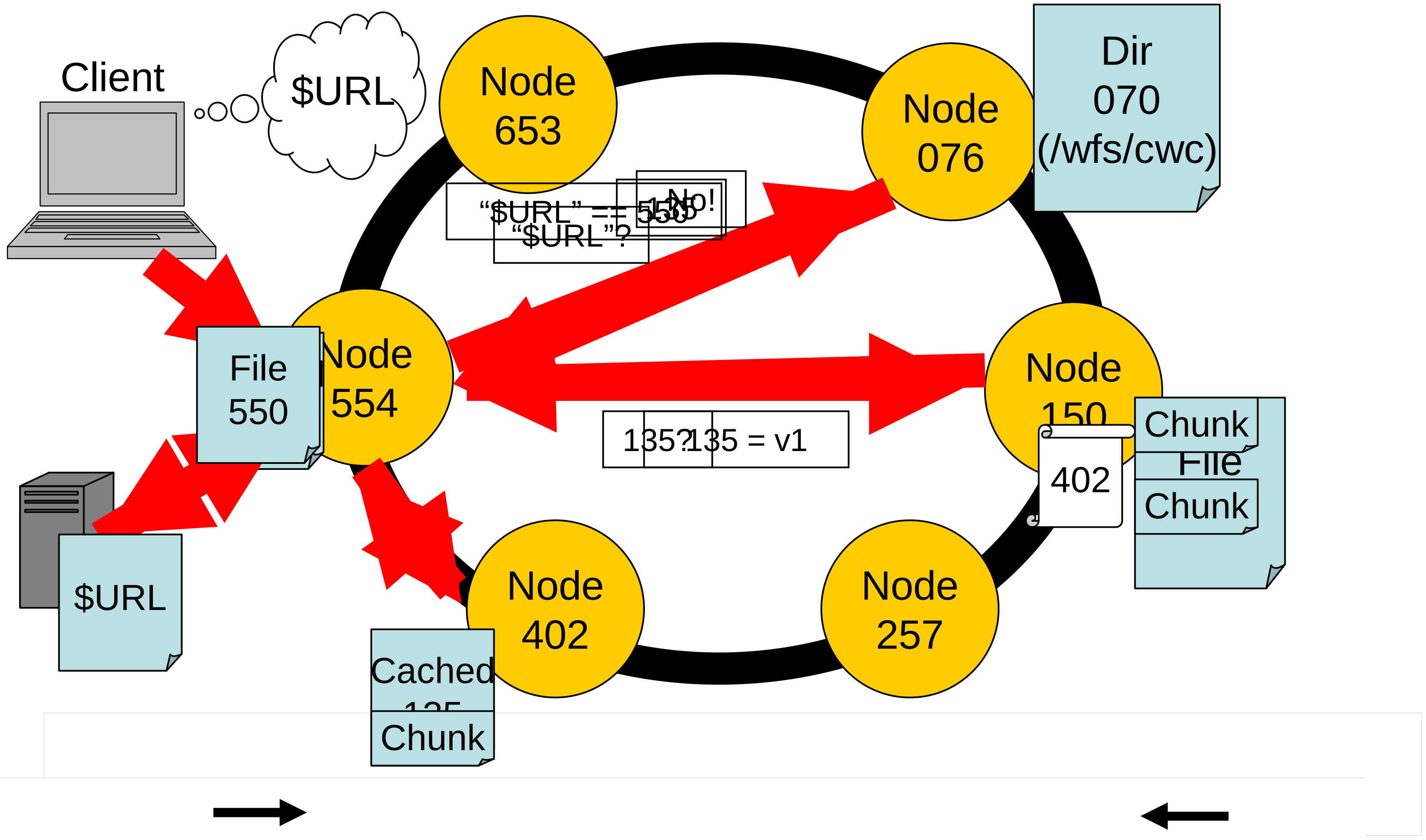
- – Serve redirected web requests
- – Fetch web content from original web servers
- – Cache web content and serve it directly
- – Find cached content on other CWC nodes

Example: Cooperative Web Cache

```
if [ –f /wfs/cwc/$URL ]; then
    if notexpired /wfs/cwc/$URL; then
        cat /wfs/cwc/$URL
        exit
    fi
fi
wget $URL –O – | tee /wfs/cwc/$URL
```

- Avoid hotspots

Example: Cooperative Web Cache



Talk Outline

1. How to decentralize your file system
2. How to control your files

Example: Cooperative Web Cache

```
if [ –f /wfs/cwc/$URL ]; then
    if notexpired /wfs/cwc/$URL; then
        cat /wfs/cwc/$URL
        exit
    fi
fi
wget $URL –O – | tee /wfs/cwc/$URL
```

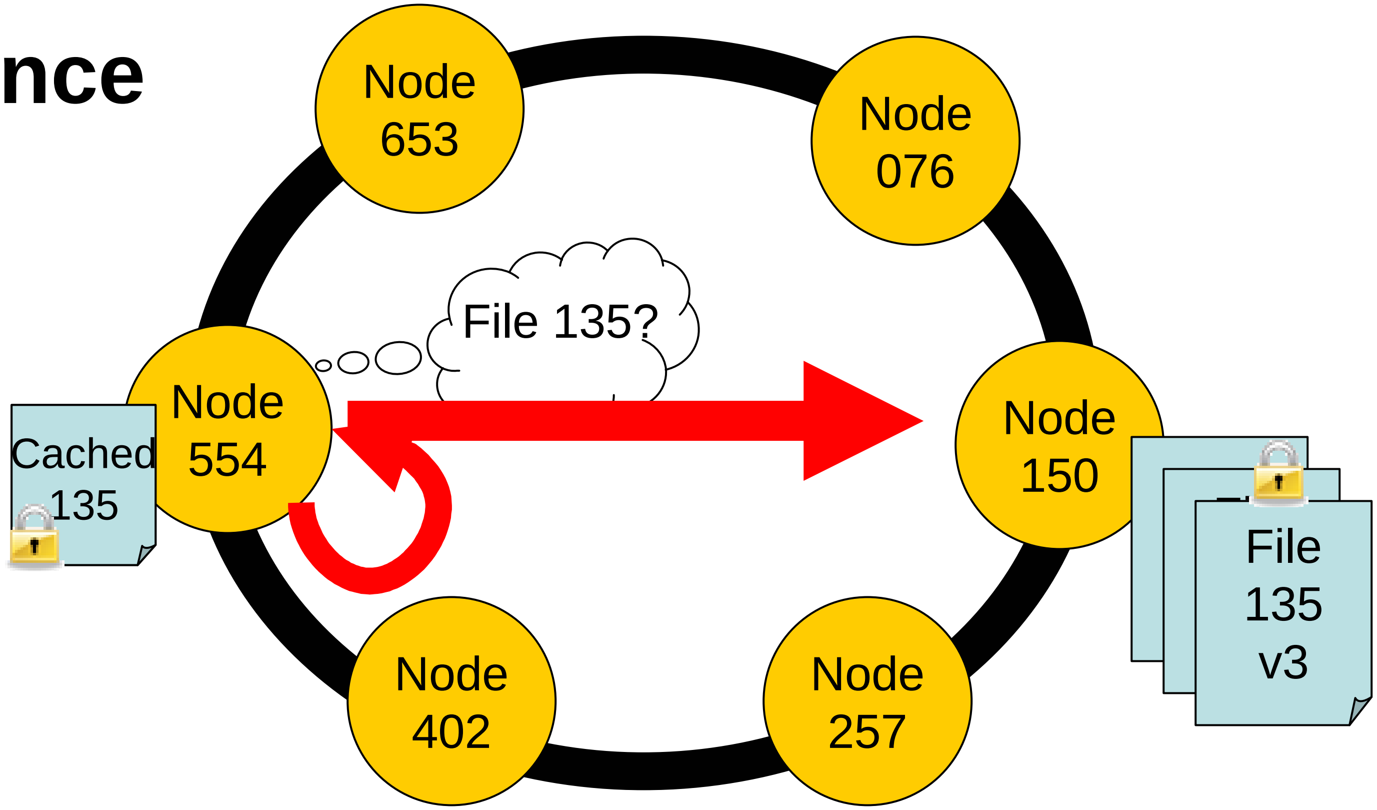
- Would rather fail and refetch than wait
- Perfect consistency isn’t crucial

Explicit Semantic Cues

- Allow direct control over system behavior
- Meta-data that attach to files, dirs, or refs
- Apply recursively down dir tree
- Possible impl: intra-path component
  - *–/wfs/cwc/.**cue**/foo/bar*

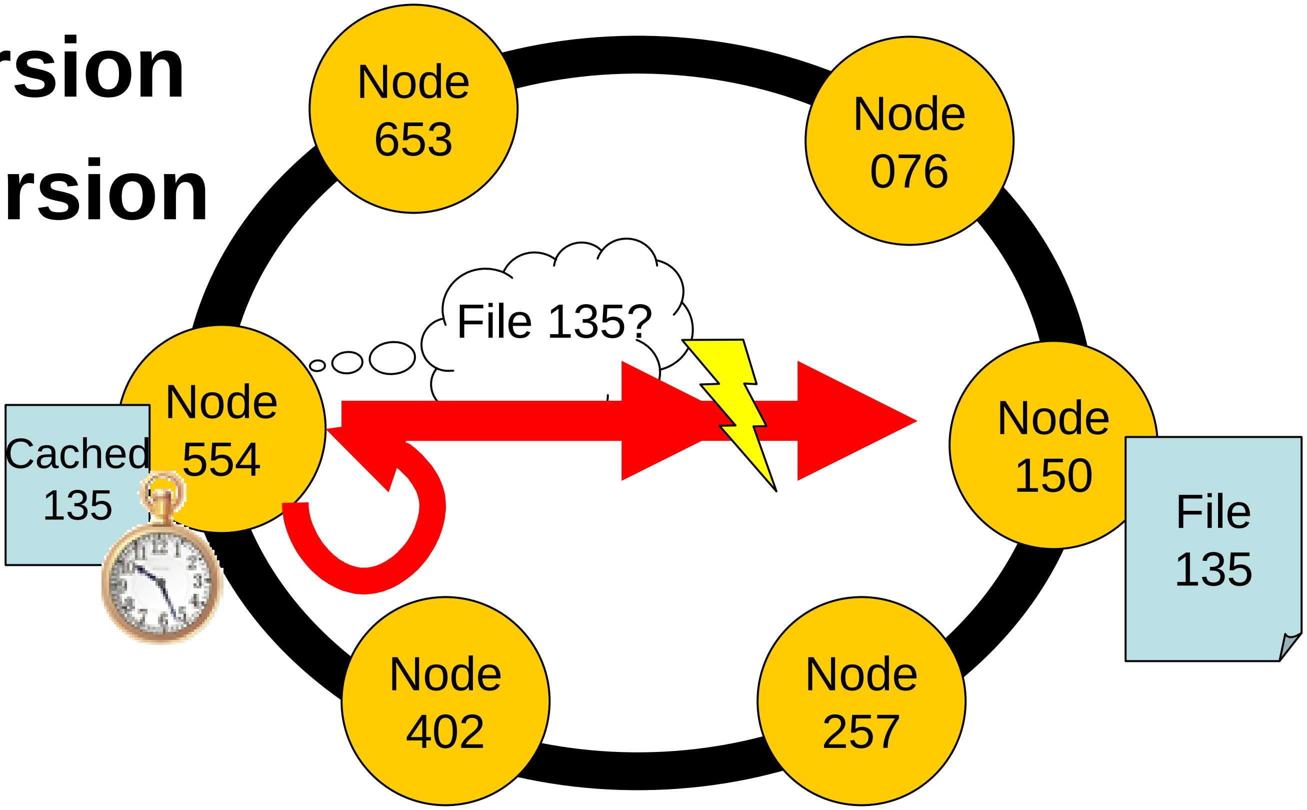
Semantic Cues: Writability

- Applies to files
- **WriteMany** (default)
- **WriteOnce**



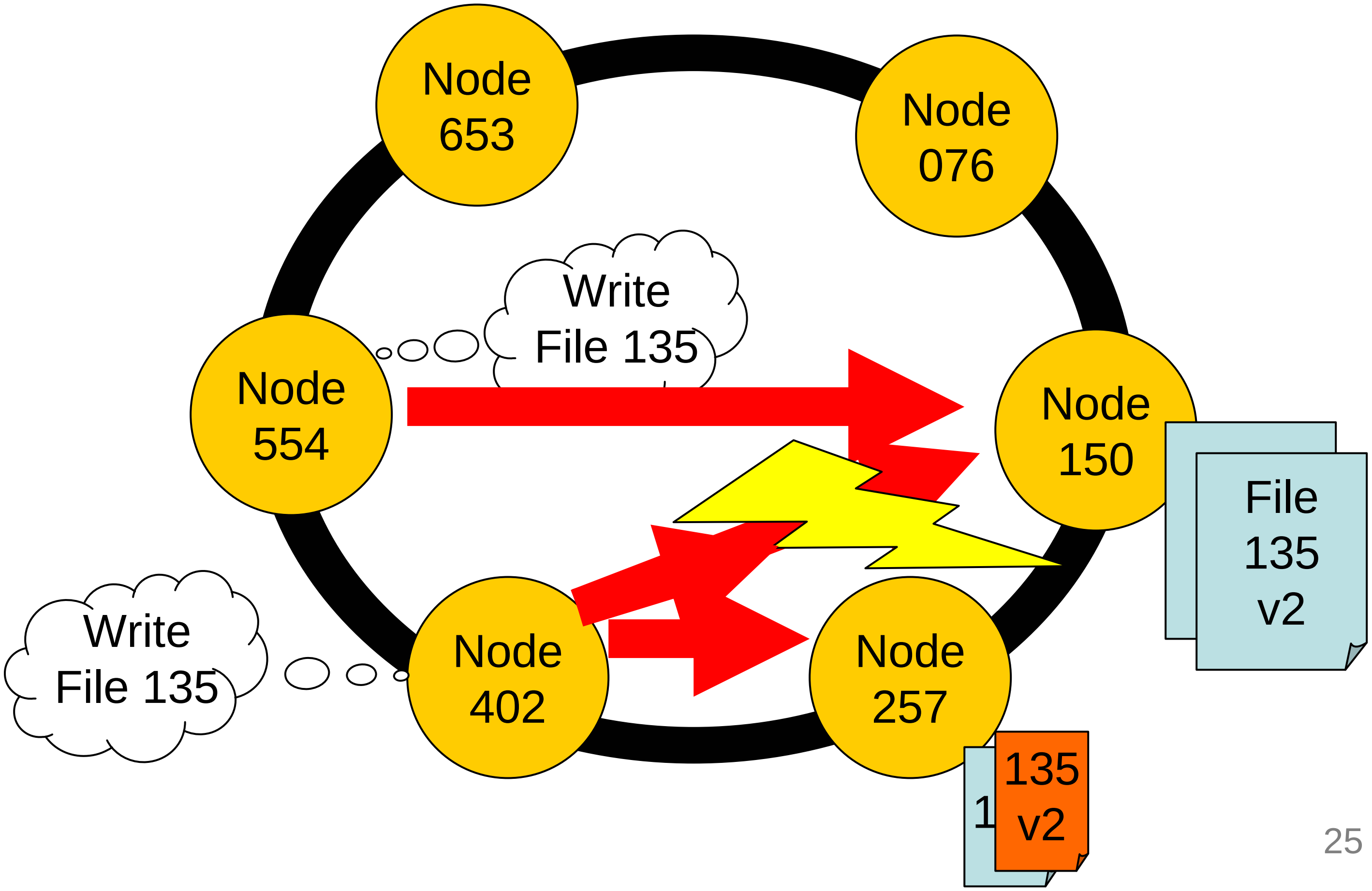
Semantic Cues: Freshness

- Applies to file references
- **LatestVersion** (default)
- **AnyVersion**
- **BestVersion**



Semantic Cues: Write Consistency

- Applies to files or directories
- **Strict** (default)
- **Lax**



Example: BLAST

- **WriteOnce** for all:
  - DB files
  - Query files
  - Result files
- Improves cachability of these files

Example: Cooperative Web Cache

- Reading an older version is ok:
  - cat /wfs/cwc/.maxtime=250,bestversion/foo
- Writing conflicting versions is ok:
  - wget http://foo > /wfs/cwc/.lax,writemany/foo

```
if [ –f /wfs/cwc/.maxtime=250,bestversion/$URL ]; then
    if notexpired /wfs/cwc/.maxtime=250,bestversion/$URL; then
        cat /wfs/cwc/.maxtime=250,bestversion/$URL
        exit
    fi
fi
wget $URL –O – | tee /wfs/cwc/.lax,writemany/$URL
```

Discussion

- Must break data up into files small enough to fit on one disk
- Stuff we swept under the rug:
  - Security
  - Atomic renames across dirs
  - Unreferenced files

Related Work

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

Conclusion

- WheelFS: distributed storage layer for newly-written applications
- Performance by reading globally and writing locally
- Control through explicit semantic cues