GIGA+: Scalable Directories for Shared File Systems

(or, How to build directories with trillions of files)

Swapnil V. Patil and Garth A. Gibson

Problem: Scalable Directories

Need high performance metadata services

- Most file systems store a directory on a single MDS
- New trends need large metadata services
  - Apps generating millions of small files in a directory, like a simple database
  - Large apps run in parallel on clusters of 100,000s of CPUs

Build scalable directories for shared file-systems

- POSIX-compliant, maintain UNIX file system semantics
- Store trillions of files and handle >100K operations/second

Goal: More Scalability Through More Parallelism

Minimize serialization

- Avoid ordered splitting of partitions, like LH* [Litwin96]

Eliminate system-wide synchronization

- Avoid using cache consistency and distributed locking, like GPFS [Schmuck02]

GIGA+ distributed indexing divides divides a directory into partitions, spread across multiple servers

- Enables highly incremental, unsynchronized, and load-balanced growth

GIGA+ Technique

Allows servers to grow their partitions independently

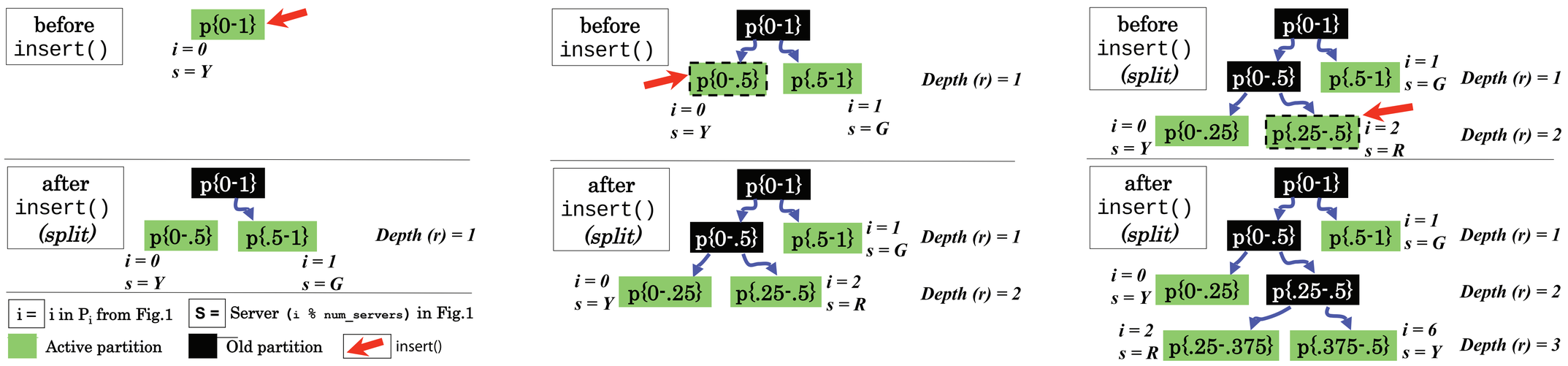
- Only maintain local state about their partitions
- Keep "split history" of their partitions

Tolerates out-of-date partition-to-server maps at the client

- Due to unsynchronized growth, map becomes state & inconsistent
- Copies updated lazily, on addressing an incorrect server

Unique, self-describing bitmap to map partitions on a server

- Tracks presence or absence of a partition and its split history
- Deterministic search of best server to send the request
- Efficiently send many bitmap updates to erroneous clients
- Compact: billion file directory, in 16 KB



GIGA+ optimizations

Power-of-2 optimization (when number of servers = 2^D)

- Below tree depth D, all split operations create partitions on the same server
- Splitting network traffic becomes zero
- Client bitmap errors go to zero (client bitmap only needs to represent first D rows of split tree)

Addition of servers with minimal redistribution

- If the number of servers is doubled, half the partitions of every current server move to the new servers

Other GIGA+ systems issues

Handles client and server failures

- Use "uniform de-clustered replication" that deterministically replicates each server's state spread across all remaining servers
- Enables load-balanced failover and fast, parallel recovery

Investigating schemes for efficiently adding any number of servers (not just doubling the number of servers)

- Inspired by consistent hashing

Carnegie Mellon

pdsi

Parallel Data Laboratory