Bigtable Data Model

(row: string, column: string, time: int64) → string

Row key: up to 64KB, 10-100B typically, sorted by reverse URL

cell w/ timestamped versions + GC

column families
Bigtable API Example

```cpp
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");

// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");

r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");

Operation op;
Apply(&op, &r1);
```

Chubby Lock Service

- A chubby cell consists of a small set of servers (replicas) – Placed in different racks, so as to minimize chance of correlated failures
- A master is elected from the replicas via Paxos – Master lease: several seconds – If master fails, a new one will be elected, but only after master leases expire
- Client talks to the master via the chubby library – All replicas are listed in DNS; clients discover master by talking to any replica
Bigtable Hierarchy

Distributed Storage Systems
RAID 0: Striping

- RAID 0 uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.

- Mirroring keeps a duplicate of each disk.
- Block interleaved parity uses much less redundancy.

RAID 1: Mirroring

- RAID 1 provides reliability via redundancy.
- Disk striping uses a group of disks as one storage unit.

- RAID is arranged into six different levels.
RAID 3: Parity Disk

RAID 5: Distributed Parity
RAID 1+0: Mirroring and Striping

Problems of Disk Arrays:

- Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes

FYI, RAID 6 extends RAID 5 with extra parity block

- Striping + mirroring
- High storage overhead/cost
- For small write-intensive apps, may be better than RAID-5 (write data twice, but no reads or XORs required)

RAID-1+0

- Mirroring + striping (strip of mirrors)
- High storage overhead/cost
- Preferable RAID level for IO-intensive applications like database, email, and web servers
- Provides better throughput and latency than all other levels (except RAID 0, which wins in throughput)

But what about the network?

- How does the network complicate things?
- What can we do about it?
- Can you think of any papers we have read that address some of these storage issues?
- What new challenges are introduced by a distributed file system in addition to scalable storage?

Distributed File System Requirements

- Transparency
  - Access, location, mobility, performance, scaling
- Concurrent file updates
- File replication
- Hardware and OS heterogeneity
- Fault tolerance
- Consistency
- Security
- Efficiency

Distributed File Systems and RAID

- Distributed file systems require higher reliability and capacity at higher loads than a local file system
- RAID was created to address the limitations
- Most work in distributed file systems ignores the disk level details (like which RAID is being used)
- But all work assumes that the disks themselves are reliable, scalable, and have high performance which can be accomplished using RAID

Next up: Distributed file systems (NFS, AFS, and GFS)