FLAT DATACENTER STORAGE

CS 744 - Big Data Systems
Fall 2018
Presenter - Arjun Balasubramanian

FLAT DATACENTER STORAGE

- Motivation
- Design
- Discussions/Questions

FLAT DATACENTER STORAGE

- Motivation
- Design
- Discussions/Questions

FDS - Motivation

What is Flat Datacenter Storage?

It's all in the name! Hereafter, we shall refer it to as FDS.

It's Flat.
It's for the Datacenters.
And of course, it's for Storage.

Claims to have read/write bandwidths in the order of GBs!
Achieved world record timing for disk-to-disk sorting in 2012.
Apache Spark now holds the record :(

FDS - Motivation

What does it offer?

Essentially a blob store.

Offers

- High Performance
- Fault Tolerance
- Large scale
- Locality-oblivious

Wait, why did we prefer locality in the first place?

FDS - Motivation

Why locality oblivious?

Why did we prefer locality in the first place?

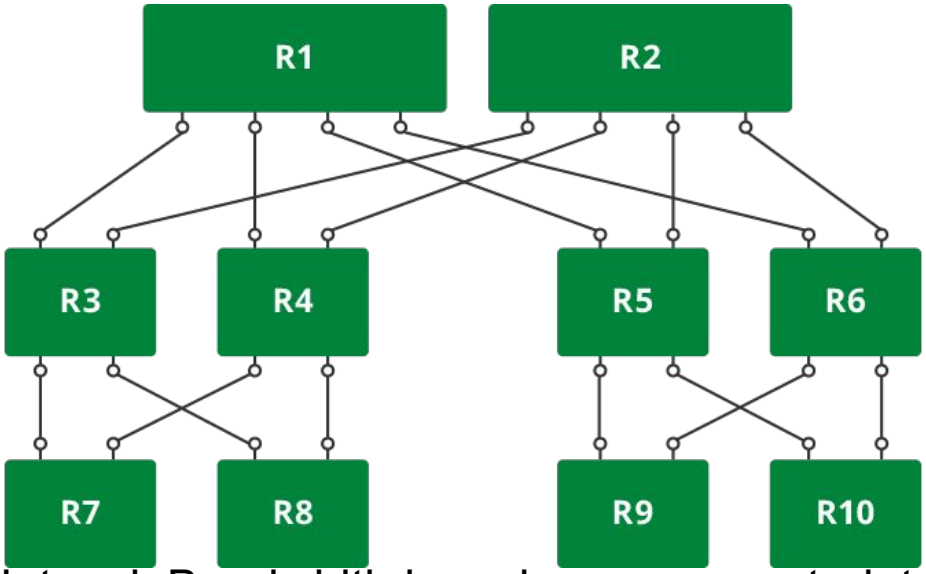
Because network bandwidth is a bottleneck!

Locality hinders computation -

- Stragglers
- Inefficient resource utilization

FDS - Motivation

What happens when we incorporate CLOS networks?



Network Bandwidth is no longer a constraint

=> Locality is no longer an advantage!

FLAT DATACENTER STORAGE

- Motivation
- Design
- Discussions/Questions

FDS - Design

- Data Management
- Architecture
- Data Placement
- APIs
- Per-Blob Metadata
- Handling Concurrent Writes
- Failure Recovery
- Replicated Data Layout

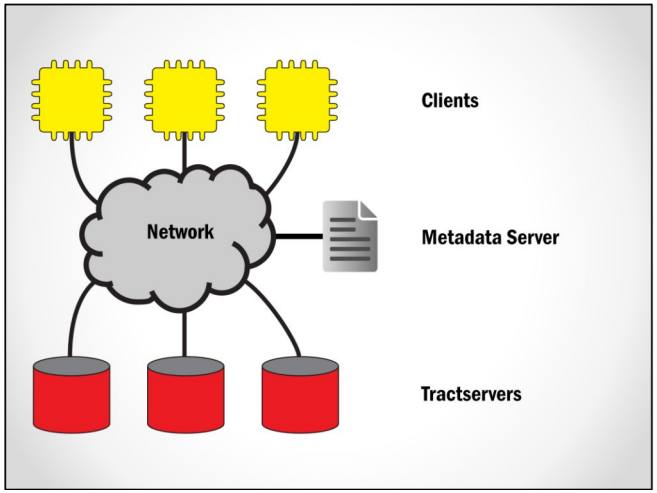
FDS - Design

Data Management

- Data stored in blobs (128 bit GUID).
- Reads/Writes done in units called “tracts” (8 MB each).
- Tracts in Blob numbered sequentially from 0.

FDS - Design

Architecture



- Metadata Server : Recall the role of Metadata Server on GFS?
- What do you think is a drawback in the Metadata Server design on GFS? Is this really a drawback?
- FDS: Metadata server collects a list of active tract-servers and gives it to the client. This list is called Tract Locator Table (TLT).

FDS - Design

Data Placement

Let’s say that a client wants to read/write on tract “i” from blob “g”

Tract_Locator = (Hash(g) + i) mod TLT_Length

Why not Hash(g+i) mod TLT_length?

Consider an example below -

4 Disks - D1, D2, D3, D4 (means we have four tract-servers)

Let’s assume that Hash(g) returns “0”

1 blob “g” divided into 8 tracts - T1, T2, .., T8.

FDS - Design

Data Placement

Sample TLT

D1, D3
D2, D3
D1, D4
D4

Tracts in each disk

Disk Number	Tracts held by this disk
D1	T1, T3
D2	T2
D3	T1, T2
D4	T4

FDS - Design

APIs are asynchronous in nature.

Getting access to a blob
`CreateBlob(UINT128 blobGuid)`
`OpenBlob(UINT128 blobGuid)`
`CloseBlob(UINT128 blobGuid)`
`DeleteBlob(UINT128 blobGuid)`
Interacting with a blob
`GetBlobSize()`
`ExtendBlobSize(UINT64 numberOfTracts)`
`WriteTract(UINT64 tractNumber, BYTE *buf)`
`ReadTract(UINT64 tractNumber, BYTE *buf)`
`GetSimultaneousLimit()`

FDS - Design

Data Placement

Let’s say a client wants to read/write tract 3 in blob “g”.

Tractor_locator = Hash(g) + i = 0 + 3 = 3.

Now look at third entry in TLT - D1, D4.

For a read request, the client would interface with either D1 or D4.
For a write request, the client would push data to both D1 and D4.

How does this compare with GFS?

FDS - Design

Important Design Considerations

- TLT changes only during cluster reconfigurations/failure.
- Clients can cache TLTs for long periods.
- Deciding replication factor “k”.

FDS - Design

Per-Blob Metadata

- Uses distributed metadata mechanism. Recall how GFS manages metadata.
- Why is distributed metadata an advantage?
- When a new blob is created, blob metadata is created along with it.
- New blobs have length 0.
- ExtendBlob() to be called before appending new data (write).
- How does ExtendBlob() help to maintain consistency when multiple writers are involved?

FDS - Design

Concurrent Writes to a Blob

Sample TLT

D1, D3
D2, D3
D1, D4
D4

Tracts in each disk

Disk Number	Tracts held by this disk
D1	T1, T3
D2	T2
D3	T1, T2
D4	T3, T4

- Clients write to all TLT entries.
- For metadata operations, client sends these operations only to a “primary” tract-server (indicated in TLT). Executed in a two-phase commit - First update replicas, then commit.
- How is this different from GFS?

FDS - Design

Failure Recovery

When metadata server detects a “dead” tract-server -

1. Invalidate current TLT. Increment version of every TLT entry where the dead tract-server appears.
2. Assign random tract-servers to fill the empty spots.
3. Send updated TLTs to all tract-servers.
4. Wait for ack on new assignment from each tract-server.

For example, if tract-server on disk D3 fails -

2	D1, D3
4	D2, D3
1	D1, D4
2	D4

becomes

3	D1, D4
5	D2, D1
1	D1, D4
3	D4

FDS - Design

Failure Recovery

Client Requests

- All client operations are tagged with version number from TLT.
- If the version number is stale, the request errors out.
- In response to error, client should invalidate cache TLT value and contact metadata server for new value.

How to handle transient failures?

- Partial Failure Recovery: When tract-server comes back up, complete failure recovery as if it never came up, or, use other replicas to get tract-server upto data with what it missed.

FDS - Design

Failure Recovery

What happens when metadata server fails?

- Should we go with a persistent design or an in-memory design? What are the trade-offs in both scenarios?
- FDS has an operator that creates a new metadata server.
- Each tract-server informs the metadata server of it's tract assignments.

Compare this design with the design of metadata server in GFS. Why do you think that they have contrasting design choices?

What happens when metadata server and tract server fail simultaneously?

FDS - Design

Replicated Data Layout

Let's assume that tract-server for D4 fails

Disk Number	Tracts held
D1	T1, T3
D2	T2
D3	T1, T2
D4	T3, T4

becomes

Disk Number	Tracts held
D1	T1, T3
D2	T2
D3	T1, T2

Tract T4 is no more available!

FDS - Design

Replicated Data Layout

Approach 1 - Let's say there are "n" (n=4) disks and replication factor of 2. Consider a TLT with each row having disk i and disk (i+1)

D1, D2
D2, D3
D3, D4
D4, D1

What happens if a tract-server fails?

Can recover the tracts from disk (i+1) and disk (i-1)

What happens if two tract-servers fail?

FDS - Design

Replicated Data Layout

Approach 1 - Let's say there are "n" (n=4) disks and replication factor of 2. Consider a TLT with each row having disk i and disk (i+1)

D1, D2
D2, D3
D3, D4
D4, D1

What happens if a tract-server fails?

Can recover the tracts from disk (i+1) and disk (i-1)

What happens if two tract-servers fail? A tract would be lost!

FDS - Design

Replicated Data Layout

Approach 2 - Assume you have 4 disks, a blob “g” with 8 tracts T1, T2, .., T8.

D1, D2
D1, D3
D1, D4
D2, D3
D2, D4
D3, D4

Tracts layout would look like

D1	T1, T2, T3, T7, T8
D2	T1, T4, T5, T7
D3	T2, T4, T6, T8
D4	T3, T5, T6

What happens when one tract-server fails here?

FDS - Design

Replicated Data Layout

Approach 2 - Assume you have 4 disks, a blob “g” with 8 tracts T1, T2, .., T8.

D1, D2
D1, D3
D1, D4
D2, D3
D2, D4
D3, D4

Tracts layout would look like

D1	T1, T2, T3, T7, T8
D2	T1, T4, T5, T7
D3	T2, T4, T6, T8
D4	T3, T5, T6

What happens when one tract-server fails here?

Each tract can be replicated from another tract-server.

FDS - Design

Replicated Data Layout

Approach 2 - Assume you have 4 disks, a blob “g” with 8 tracts T1, T2, .., T8.

D1, D2
D1, D3
D1, D4
D2, D3
D2, D4
D3, D4

Tracts layout would look like

D1	T1, T2, T3, T7, T8
D2	T1, T4, T5, T7
D3	T2, T4, T6, T8
D4	T3, T5, T6

Which approach (Approach 1 vs. Approach 2) do you think is better?

FDS - Design

Replicated Data Layout

Approach 2 - Assume you have 4 disks, a blob “g” with 8 tracts T1, T2, .., T8.

D1, D2
D1, D3
D1, D4
D2, D3
D2, D4
D3, D4

Tracts layout would look like

D1	T1, T2, T3, T7, T8
D2	T1, T4, T5, T7
D3	T2, T4, T6, T8
D4	T3, T5, T6

Which approach (Approach 1 vs. Approach 2) do you think is better?

Approach 2 has faster recovery since it can replicate from more tract-servers.

FDS - Design

Replicated Data Layout

Assume you have 4 disks, a blob “g” with 8 tracts T1, T2, .., T8.

D1, D2
D1, D3
D1, D4
D2, D3
D2, D4
D3, D4

Tracts layout would look like

D1	T1, T2, T3, T7, T8
D2	T1, T4, T5, T7
D3	T2, T4, T6, T8
D4	T3, T5, T6

What happens when two tract-servers fails here?

FDS - Design

Replicated Data Layout

Assume you have 4 disks, a blob “g” with 8 tracts T1, T2, .., T8.

D1, D2
D1, D3
D1, D4
D2, D3
D2, D4
D3, D4

Tracts layout would look like

D1	T1, T2, T3, T7, T8
D2	T1, T4, T5, T7
D3	T2, T4, T6, T8
D4	T3, T5, T6

What happens when two tract-servers fails here? Still, data can be lost!

FDS - Design

Replicated Data Layout

- Generally, k>2 replication factor is used.
- First two replications are done pair-wise to maximize resource utilization.
- Remaining (k-2) replications are done with random tract-servers.

Memory considerations for metaserver

- Above approach returns in O(n^2) TLT size.
- Each row in the TLT has k entries.
- Can reduce memory overheads by limiting number of disks that participate in recovery.

FDS - Design

Failure Domains

Failure Domains are a set of machines that can have correlated failures.

FDS ensures that Failure Domains do not feature on a single row.

FDS - Design

Cluster Growth

- When tract-server is added, TLT entries are taken away from other tract-servers and given to new servers.
- The assignment is given to the new tract-server, version number is incremented, and is marked as pending.
- Once replication is done in new tract-server, the update in TLT is committed and made available to clients.

FDS - Design

Consistency Model

- Let’s say client is writing to update tract T1 in D1, D2, D3.
- Client writes to D1 and then and then crashes.
- Results in in-consistency. This is a weak consistency model.
- What consistency model does GFS have?

FLAT DATACENTER STORAGE

- Motivation
- Design
- Benchmarking
- Discussions/Questions

FDS - Discussions/Questions

Some questions to think about -

1. What is the difference in workload models for which GFS and FDS are designed?
2. Why do we need to split blobs into tracts?
3. Do FDS APIs really need to be asynchronous?
4. In FDS, why are writes not sent through the primary with pipelining enabled like in GFS?
5. Why are chunks 64MB in GFS and tracts 8MB in FDS?

Any questions from the audience?

FLAT DATACENTER STORAGE

Thank you!