Flat Datacenter Storage

Venkata Sandeep Kumar Yerlanki
Paper Outline

- Introduction
- System overview
- Design overview
  - Blobs and Tracts
- Summary
Introduction

- Flat Datacenter Storage (FDS) is a high-performance, fault-tolerant, large-scale, locality-obliviouse blob store
- Simple and Clear
- FDS – Storage System
- Idle case to reality
Idle Case

- Processors and disks
- Writing and Storing
- Magic RAID
- Fine-granularity across the disks
- High Utilization
- Statistical Multiplexing
- Disks stay busy
- Easy to software
Easy to adjust the ratio of processors and disks

Metadata management

Physical data transport

“Consider a centralized file server in a small computer science department. Data stored by any computer can be retrieved by any other. This conceptual simplicity makes it easy to use: computation can happen on any computer, even in parallel, without regard to first putting data in the right place.” Is GFS a centralized file system? To achieve good performance for an I/O-intensive program running on GFS, how does the data placement affect its performance? [Hint: consider the case where processes of the program run on the chunkservers.]
Questions

• “The root of this cascade of consequences was the locality constraint, itself rooted in the datacenter bandwidth shortage.” Show example consequences of relying on locality in program’s execution. Why does a sufficient I/O bandwidth help remove the constraint?

• “Even when computation is co-located with storage, all storage is treated as remote; in FDS, there are no “local” disks.” Explain why the distinction of local and remote accesses is largely removed.
Location Awareness adds **Complexity**

Why “move the computation to the data”

Remote Data Access is slow. Why?

The Network is Oversubscribed
Aggregate Bandwidth Above Less Than Aggregate Demand Below Sometimes by 100x or more
Consequences

- No local vs Remote disk distinction
- Simpler work scheduler
- Simpler programming models
- FDS, object storage assuming no oversubscription
- Built on a CLOS network with distributed scheduling
Network

Metadata Server
Questions

- “In FDS, data is logically stored in blobs. ... Reads from and writes to a blob are done in units called tracts.” What are blob and tracts? Are they of constant sizes?
- “In our cluster, tracts are 8MB”. Why is a tract in FDS sized this large?
- “Tractservers do not use a file system.” Explain this design choice.
- “Figure 1: FDS API”. Read the API. Does FDS support file operations? Is a blob similar to a file in a (distributed) file system?
Getting access to a blob | Interacting with a blob
--- | ---
CreateBlob(UINT128 blobGuid) | GetBlobSize()
OpenBlob(UINT128 blobGuid) | ExtendBlobSize(UINT64 numberOfTracts)
CloseBlob(UINT128 blobGuid) | WriteTract(UINT64 tractNumber, BYTE *buf)
DeleteBlob(UINT128 blobGuid) | ReadTract(UINT64 tractNumber, BYTE *buf)

// create a blob with the specified GUID
CreateBlob(GUID, &blobHandle, doneCallbackFunction);

//...

// Write 8mb from buf to tract 0 of the blob.
blobHandle->WriteTract(0, buf, doneCallbackFunction);

// Read tract 2 of blob into buf
blobHandle->ReadTract(2, buf, doneCallbackFunction);
Questions ?
Reference

- https://www.youtube.com/watch?v=YbOjxCxtMpU
- https://www.usenix.org/conference/osdi12/technical-sessions/presentation/nighting