EECS 491
Introduction to Distributed Systems

Windows Azure Storage

November 12, 2019

Harsha V. Madhyastha
Announcements

- **Thursday:**
  - I’ll be out of town
  - GSIs will discuss solutions to midterm exam

- **Next Thursday:**
  - Guest lecture from Google Ads back-end tech lead
  - Lecture won’t be recorded
Midterm exam statistics

- Median: 75 and Mean: 77
- Standard deviation: 15.75
- Min: 37
- Max: 100
Overall grade status

- Median score will likely result in B+
- Likely A or better if midterm $\geq 90$
- Likely C+ or lower if midterm $\leq 60$

- C or higher if total score $\geq 78$
  - Implication: Pass if you complete all projects and average 45% on the exams
Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency

Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam Mc Kelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, Leonidas Rigas

Microsoft
Windows Azure Storage

- Focus on storage within a data center

- Goals:
  - Durability
  - Scalability
  - Strong consistency
  - High availability

- Primary-backup replication
Revisiting Project 2

- Assume viewservice is Paxos-based RSM
- Why would client’s Append fail?
- Viewservice may not have detected failure
- Key idea in Azure: Client triggers view change
- Why didn’t we do this in Project 2?
  - View change was a heavyweight operation
Thought experiment

- In Project 2, what if we stored a log of views?
- How to leverage this to make view change lightweight?
- Hint: All replicas don’t fail when view changes
- Don’t transfer any state upon view change
- Serve GETs based on state from all views
WAS: High-level idea

- Treat state as **append-only log**
  - PUTs from clients applied as appends to log

- Log comprises concatenation of extents
  - Only last extent can be modified; all others read-only

- View change has two steps:
  - Seal last extent
  - Add new extent
Logical view of data in WAS

Stream //foo/myfile.dat

Ptr E1  Ptr E2  Ptr E3  Ptr E4  Ptr E5

Extent E1  Extent E2  Extent E3  Extent E4  Extent E5
Creating an Extent

Client

Create Stream/Extent

EN1 Primary
EN2, EN3 Secondary

Stream Master

Paxos

Allocate Extent replica set

EN 1
Primary

EN 2
Secondary A

EN 3
Secondary B

EN
Replication Flow

Upon failure, how to determine size of extent before sealing it?
Extent Sealing (Scenario 1)

Client

Append

120

120

Seal Extent

Seal Extent

Stream Master

Seal Extent

Sealed at 120

Ask for current length

EN 1

Primary

EN 2

Secondary A

EN 3

Secondary B

EN 4

November 12, 2019

EECS 491 – Lecture 18
Extent Sealing (Scenario 1)

Client

EN 1
Primary

EN 2
Secondary A

EN 3
X
Secondary B

EN 4

Seal Extent
Sealed at 120

Sync with SM

November 12, 2019
EECS 491 – Lecture 18
Extent Sealing (Scenario 2)

Scenario in which servers would differ in responses?
Extent Sealing (Scenario 2)

Client

Seal Extent

SM

Seal Extent

Sealed at 100

Append

EN 1
Primary

EN 2
Secondary A

EN 3
Secondary B

EN 4

Ask for current length

120

100

November 12, 2019
Extent Sealing (Scenario 2)

- Client
- EN 1: Primary
- EN 2: Secondary A
- EN 3: Secondary B
- EN 4

Sync with SM
Seal Extent
Sealed at 100

November 12, 2019
Serving GETs

- How to lookup latest value of any key?
- Ask Stream Master (aka view service)?
  - Stream Master only has definitive copy of metadata: Number of extents, Length of sealed extents, …
  - Unaware of what updates clients performed
Serving GETs

- Need to maintain index of data in each stream
  - “which offset in which extent has latest value for key?”

- How to execute a GET on a key?
  - Lookup index for (extent, offset), then lookup stream

- Implication for execution of PUTs?
  - Must not only append to stream, but also update index
Access blob storage via the URL: http://<account>.blob.core.windows.net/

Layering in Azure Storage

Storage Stamp

LB

Front-Ends

Partition Layer

Stream Layer

Intra-stamp replication

Storage Stamp

Partition Layer

Stream Layer

Intra-stamp replication

Storage Stamp

Inter-stamp (Geo) replication
Partition Layer – Index Range Partitioning

Blob Index

<table>
<thead>
<tr>
<th>Account Name</th>
<th>Container Name</th>
<th>Blob Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaaa</td>
<td>aaaa</td>
<td>aaaaa</td>
</tr>
<tr>
<td>........</td>
<td>........</td>
<td>........</td>
</tr>
<tr>
<td>........</td>
<td>........</td>
<td>........</td>
</tr>
<tr>
<td>harry</td>
<td>pictures</td>
<td>sunrise</td>
</tr>
</tbody>
</table>

Storage Stamp

A-H: PS1
H’-R: PS2
R’-Z: PS3

Partition Map

PS 1

PS 2

PS 3

November 12, 2019

EECS 491 – Lecture 18
Overall Architecture

Front End Layer

Partition Layer

Partition Server
Partition Server
Partition Server
Partition Server

Partition Master

Lock Service

Incoming Write Request

Stream Layer

Extent Nodes (EN)

Paxos

M

M

M

M
Beating CAP Theorem

- Stream layer
  - Availability: Seal extent and switch to new replica set
  - Consistency: Replicas agree on commit length

- Partition layer
  - Consistency: Disjoint partition ranges across servers
  - Availability: Partition servers only cache state

- Key enabler: Append-only design
  - Drawbacks?
  - Garbage collection overhead
Enabling Scalability

- Use replicated Paxos-based service to make centralized decisions
  - Stream Master
  - Partition Master
  - Lock service at partition layer

- But …
  - Ensure central service is off the data path
  - Cache state from central service when possible
Project 4

- Due December 5th
- Two types of sharded key-value stores

Part A:
- Consistent hashing

Parts B and C:
- Servers partitioned into Paxos groups
- Shardmaster assigns shards to replica groups
Project 4

- Option to opt in to research study aimed at developing a debugging assistant
- **Download different version of handout code**

- Every time you run `go test`
  - Takes a snapshot of your code
  - Captures output of the test cases
  - Option of annotating snapshot with a message
- **Completely optional. Can opt-out at any time.**
Replica Selection

- How should stream master choose 3 replicas for an extent?

- Different copies in different racks
  - Ideally, racks with different sources of power
Optimizing Read Latency

- Can read from any replica of a sealed extent
- What if replica that a client chooses to read from is currently under high load?
- How to reduce latency impact?

- Read from all 3 replicas; wait for first response
  - 3x increase in load
- Read from any replica; give deadline to respond
  - Retry at another replica if deadline violation
Optimizing Storage Cost

● Storing 3 replicas $\rightarrow$ 3x storage cost
  ◆ How to reduce amount of data stored?

● Exploit benefit of "sealing" an extent:
  ◆ Content of sealed extent never changes in future
Erasure Coding

- Split data into $M$ chunks
- Use coding to generate $N$ additional chunks
- Data recoverable using any $M$ of $(M + N)$ chunks
  - How many chunks must be lost to lose data?
- Better durability than 3x replication at 1.3–1.5x storage overhead
Combining Optimizations

- How to address tradeoff?
  - Erasure coding: Lower storage cost
  - Replication: Lower read latency

- For any extent
  - Store one original copy
  - Store \((M + N)\) erasure coded chunks

- If read from replica misses deadline, reconstruct data from coded chunks
Regrade requests

● Type your request clearly explaining why you believe your exam is incorrectly graded

● Submit request before class next Tuesday

● Will re-grade entire exam
  ◆ Your score may go up or down