Case Study: Implementing SLOs for a new service

Arnaud Lawson
@arnolawson
Who am I?

- Arnaud Lawson, Sr Site Reliability Engineer @Squarespace
  - Twitter: @arnolawson
  - Email: alawson@squarespace.com
Outline

- What is this service?
- Why SLOs?
- Our approach for defining & measuring SLOs
- Benefits gained & lessons learned
What is this service?

- Ceph Object Storage (COS) service
  - S3-compatible
  - Geo-distributed
Ceph Object Storage (COS)

How do we use it?

- **Apps**
  - Internal webapps
  - Production data pipelines
  - Performance monitoring systems
- **Backups**
  - Production data stores
Why SLOs?

What are SLOs?

- Service level objectives
  - set performance & *reliability targets* for a service as seen by its users *over a period of time*
- Service level indicator
  - performance *metrics that inform SLOs*
Why SLOs?

Example

- API availability SLO: 99.9% of API requests will not fail over $n$ weeks

- API availability SLI: the percentage of API requests that do not fail
Why SLOs?

Why are SLOs important for COS?

- COS usage grew
- Define performance & reliability targets
- Measuring & meeting SLOs guarantees users’ happiness
- Better prioritize our work around the life of this service
SLO implementation process

1- **Determine SLI types** that best capture our users’ experience

2- **Define SLIs** - the things to measure

3- **Choose how to measure** these SLIs

4- **Collect SLIs** for a few weeks & **estimate initial SLOs**

5- **Infer error budgets** from the initial SLOs

6- **Publish** SLOs
SLO implementation

1- Determine SLI types that best capture users’ experience

a- Understand how users most often interact with COS
   - User actions in server logs:
     ○ Create & delete bucket
     ○ Upload, download & delete object

b- Understand COS components & choose SLI types that best reflect users’ experience
   - request-driven RESTful interface
     ○ availability & latency SLIs
   - distributed storage backend
     ○ durability SLI
2- Define SLIs

- **Request-driven HTTP server**
  - **Availability** SLI: percentage of http requests that do not fail
  - **Latency** SLI: percentage of http requests that successfully complete in less than $x$ milliseconds

- **Storage backend**
  - **Durability** SLI: percentage of objects written to COS that can be successfully re-read without corruption even after a failure
SLO implementation

3- Choose how to measure these SLIs and capture the user experience

- Collect SLIs from COS load balancer logs
- Instrument COS S3 client programs
- Deploy probers which perform common user actions
4- Collect SLIs & set SLOs

- Deployed probers
- Record **success** & **latency** metrics per request type and across all http requests
SLO implementation

4- Collect SLIs & set SLOs

- p90 & p99 Latency SLI over 4 weeks for all HTTP requests issued by probers
SLO implementation

4- Collect SLIs & set SLOs

- Latency per request type
- We can drill down and identify requests that negatively impact our overall latency SLO
SLO implementation

4- Collect SLIs & set SLOs

● **Availability SLO**: 99.9% of requests will complete successfully over 4 weeks

● **Latency SLOs**:
  ○ a) 90% of requests will complete successfully in < 300 ms over 4 weeks
  ○ b) 99% of requests will complete successfully in < 2000 ms over 4 weeks

● **Durability SLO**: 99.999999% of objects written to COS will not be lost or compromised in the event of a failure over 1 year
SLO implementation

5- Infer error budgets from initial SLOs

- Error budget
  - Amount of headroom there is above an SLO
  - Degree to which we can afford to not be within SLO and not frustrate users significantly
5- Infer error budgets from initial SLOs

- 99.9% availability over 4 weeks → **0.1% requests could fail over 4 weeks**
- 90% requests will complete successfully in < 300 ms over 4 weeks → ~**10% requests are allowed to complete in >= 300 ms over 4 weeks**
- 99% requests will complete successfully in < 2000 ms over 4 weeks → ~**1% of requests are allowed to complete >= 2000 ms or longer over 4 weeks**
- 99.999999% durability of objects per year → a **loss of ~0.000001% of objects is allowed per year**
SLO implementation

6- Publish SLOs

- Produced documentation that outlines
  - What COS does
  - How it is actually used
  - Types of SLIs being measured
  - A definition of the actual SLIs - what is being measured
  - A definition of the SLOs that are being informed by the SLIs
  - A rationale for why these SLOs & SLIs were chosen
Conclusion

Benefits

- SLIs inform decisions for prioritizing reliability projects, doing capacity planning, etc
- SLI graphs help identify service issues
- Users easily determine whether our service is appropriate for a particular use case based on SLOs
- Use SLIs for monitoring & don’t have to page engineers if we are within SLOs
Conclusion

Lessons learned

- Choose a metrics collection service with a powerful query language
- Data durability SLO implementation for storage systems can be tricky
Tips for defining & measuring SLOs

- *Never strive for 100% reliability*
- *Understand the components* of the system
- *Know how users interact* with the system
- *Collect SLIs* that measure the aspects of the system that matter to users
- *Use SLO results to prioritize work* on reliability engineering projects
Merci