SLOs and SLIs in the Real World:
A Deep Dive

Elisa Binette  Matthew Flaming
@elisabPDX    @mflaming
<table>
<thead>
<tr>
<th>Service Level Indicator</th>
<th>Service Level Objective</th>
<th>Service Level Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>X should be true...</td>
<td>Y proportion of the time</td>
<td>or else...</td>
</tr>
<tr>
<td>“10 key takeaways about SLIs delivered in 30 minutes”</td>
<td>99.9% of the time</td>
<td>We owe the conference organizers a drink</td>
</tr>
</tbody>
</table>
“Data is being collected properly, and customers can login to the system and view their data 99.9% of the time”
System Boundaries

Login Service: Authenticate user credentials
A System of Systems

- UI/API Tier
- Login Service
- Data Storage/Query Tier
- Legacy Data Tier
- Data Ingest & Routing
SLIs + SLOs: A Simple Recipe

1. Identify system boundaries
2. Define capabilities exposed by each system
3. Define Plain-Language definition of “available” for each capability
4. Define corresponding technical SLIs
5. Start measuring to get a baseline
6. Define SLO targets (per SLI or per capability)
7. Iterate and tune
Data Ingest Tier
Multiple capabilities
- Data ingested
- Data routed
One (or more) SLIs per Capability

Data Ingest SLI
Percent of well-formed payloads accepted

Data Ingest SLO
99.9%

Data Routing SLI
Time to deliver message to correct destination

Data Routing SLO
99.5% of messages in under 5 sec
Choosing SLO targets

SLO numbers need to be:
- What the team *actually* commits to supporting
- What the org *actually* commits to supporting
- Reflective of technical reality

SLOs represent an ongoing commitment!

When in doubt, measure first
SLIs Act as Broad Proxies for Availability

Horizontally Scaled Data Tier
Multiple Capabilities
• Insert data
• Query data

Multiple SLIs
• Insert latency
• Query response time
• Query error rate

ACME MONITORING PRODUCT
Compound SLOs

99.95% of well formed queries will receive well formed responses

99.9% of queries will be answered in less than 1000ms
SLIs/SLOs for Hard Sharded Systems

Hard Sharded Legacy Data Tier

Multiple Capabilities
- Insert data
- Query data

Multiple SLIs
- Insert latency
- Query response time
- Query error rate
Sharded vs. Horizontally Scaled SLOs

Horizontally Scaled

- 66% SLO

Hard Sharded

- 0% SLO
- 100% SLO
- 100% SLO
Defining SLIs/SLOs for Core Infrastructure

Container Orchestration/Runtime

Network
SLIs + SLOs: A Simple Recipe

1. Identify system boundaries
2. Define capabilities exposed by each system
3. Plain-Language definition of “available” for each capability
4. Define corresponding technical SLIs
5. Start measuring to get a baseline
6. Define SLO targets (per SLI or per capability)
7. Iterate and tune

@elisabPDX  @mflaming
Ask the Customer!

What do you use the network tier for?

What kinds of guarantees would you like to see?

What assumptions do you make in your code?
Networking Tier SLOs

Network

Multiple Capabilities
• Load balancing
• Intra-AZ routing
• Inter-AZ routing

Multiple SLIs
• Load balancer endpoint uptime
• Intra-AZ latency/packet loss
• Inter-AZ latency/packet loss

One SLO per capability
• 99.99% goal
Hard Dependencies Require Higher SLOs

- MTTR typically increases with # impacted service tiers
- Graceful degradation harder to implement for infrastructure outages
Capabilities Clarify Contracts

Container scheduler/cluster

Single Capability
• Run jobs with expected resources

Multiple SLIs??

“Accepted jobs will run with the quota of requested resources available to them 99.99% of the time.”
“Accepted jobs will run with the quota of requested resources available to them 99.99% of the time.”

**Expected resources available to jobs**
- CPU and memory quotas FTW
- Network saturation is possible
“Accepted jobs will run with the quota of requested resources available to them 99.99% of the time.”

**Jobs in runnable state 99.99% of time**

- Potential uptime vs. job correctness

```c
/**
 * Describes possible task states. IMPORTANT: Mesos assumes tasks that
 * enter terminal states (see below) imply the task is no longer
 * running and thus clean up any thing associated with the task
 * (ultimately offering any resources being consumed by that task to
 * another task).
 */
enum TaskState {
  TASK_STAGING = 6; // Initial state. Framework status updates should not use.
  TASK_STARTING = 8; // The task is being launched by the executor.
  TASK_RUNNING = 1;
  TASK_KILLING = 2; // The task is being killed by the executor.
  TASK_FINISHED = 3; // The task finished successfully on its own without external interference.
  TASK_FAILED = 4; // TERMINAL: The task failed to finish successfully.
  TASK_KILLED = 5; // TERMINAL: The task was killed by the executor.
  TASK_ERROR = 7; // TERMINAL: The task description contains an error.
  TASK_LOST = 8; // TERMINAL: the task failed to launch because of a transient error.
  TASK_UNREACHABLE = 9; // The task was running on an agent that has lost contact with the master
  TASK_GONE = 10; // TERMINAL. This can occur if the agent has been terminated along with all of its tasks
  TASK_GONE_OPERATOR = 11; // The task was running on an agent that the master cannot contact
  TASK_UNKNOWN = 12; // The master has no knowledge of the task.
}
```
UI Tier
One Capability
• Deliver fast, error-free UI

Multiple SLIs
• Page load time (by geo)
• Browser interaction time (by geo)
• Server error rate
• JS error rate (by browser)
SLIs/SLOs for UIs

Welcome to your data, let's get querying

```
NewRelic Administration > SELECT percentile(duration, 95) FROM BrowserInteraction WHERE `mobileOptimized` IS NULL and 'userAgentName' like 'Chrome' TIMESERIES FACET category SINCE 30 minutes AGO LIMIT 500
```
Overall

Multiple Capabilities
- Collect data
- Login
- View data

Single dumb SLI
- Does sample workflow succeed

Allows us to sanity check individual system SLIs/SLOs
Customer Specific SLOs
Recap

1. Worry about SLIs more than SLOs.

2. Start with Plain-Language descriptions of availability, not with technical underpinnings.

3. Define SLIs and SLOs for specific capabilities at system boundaries.

4. Each logical instance of a system (e.g., hard shard) gets its own SLO.

5. SLIs are not the same as alerts, and are not a replacement for thorough alerting.

6. “AND” together SLIs for a given capability into a single SLO for that capability.

7. Write down your SLI/SLO contracts and share them.

8. Key customers may need their own SLOs/SLIs.

9. Assume SLOs and SLIs will both evolve over time.

10. SLOs represent an ongoing commitment.

@elisabPDX  @mflaming
<table>
<thead>
<tr>
<th>Service Level Indicator</th>
<th>Service Level Objective</th>
<th>Service Level Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>“10 key takeaways about SLIs delivered in 30 minutes”</td>
<td>99.9% of the time</td>
<td>…or we owe the conference organizers a drink.”</td>
</tr>
</tbody>
</table>
Thank you

Elisa Binette  
@elisabPDX

Matthew Flaming  
@mflaming