LTAT.06.007 Distributed Systems
Lecture 13 – Monitoring

Huber Flores, PhD
ASSOCIATE PROFESSOR

Tartu, Estonia 02/05/2022
Recap

- System architectures
  - Centralized and decentralized
  - Application layering
- Scalability
  - Functional decomposition
  - Partitioning
  - Duplication
- Dependable systems

![Diagram showing API Gateway and services A, B, C]
Think about

System usage
Think about

System usage
Agenda

• **Goal:** To study approaches to monitor the runtime behavior of a distributed system, such that it is possible to detect and identify failures

• **Content:**
  - Monitoring
  - System performance and metrics
  - Basics of system modelling

After this lecture, you should be able to:

• To explain the relation between system performance and system metrics

• To apply monitoring methods to a distributed system to extract quantifiable metrics about its service quality level
Monitoring

Essence

- Monitoring is primarily used to detect failures that impact users in production and trigger notifications (alerts) sent to human operators responsible for mitigating them.

- Monitoring involves quantifying and estimating metrics from the system.

- Metrics depict the **performance** of the system during execution (instantiation)
System performance

Essence

**Metric**: a numeric representation of information measures over a time interval and represented as a time series, e.g., number of requests handled by a service.

Conceptually, a metric is a list of samples, where is sample is represented by a floating-point number and a timestamp,

Observation

Modern monitoring systems allow a metric to be tagged with a set of key-value pairs called, **labels** which increases the dimensionality of the metric.
System performance

Observation

• A service should emit metrics about its load, internal state, and availability and performance of downstream service dependencies
  Fine-grained metrics require explicit code changes and a deliberate effort by developers to instrument their code

• White-box and black-box approaches

Observation

• Black box requires external observation of the system – good to detect the symptoms that emerged from failures (when something is broken)

• White box requires instrumentation through logs - good to identify root causes of failures.
## System performance

### Multiple tools and dashboards

- Azure Monitor’s log-based metrics
- Amazon monitoring
- JMeter
- Taurus
- Tsung
- Locust
- Fortio
- K6
- WebLOAD
- TestRail
- Postman
- BlazeMeter Continuous Testing Platform
## Server performance

### Essence

The ability to operate (execute processes/transactions/jobs/etc) reliably and dependably to meet interaction and behavior expectations.

### Quality of Service (QoS)

Service-level indicators that measure one aspect of the level of service provided by a service to its users

- Response time
- Throughput
- Availability
- Reliability
- Security
- Scalability
- Extensibility
Service-level indicators (SLI)s provide an overview as the ratio of good events over the total number of events, where an event is a feature/component/aspect of the system in execution.
Server performance

QoS attributes

- **Response time**: the time it takes a system to react to a human request
- **Throughput**: the rate at which requests are completed from a computer system and is measured in operations per unit time.
- **Availability**: the fraction of time that a system is up and available to its customers
- **Reliability**: the probability that it functions properly and continuously over a fixed period of time. Reliability and availability are closely related concepts but are different. When the time period during which the reliability is computed becomes very large, the reliability tends to the availability.
- **Security**: A combination of Confidentiality, Data integrity and Non-repudiation
- **Scalability**: A system is said to be *scalable* if its performance does not degrade significantly as the number of users, or equivalently, the load on the system increases
- **Extensibility**: is the property of a system to easily evolve to cope with new functional and performance requirements.
Service-level objective (SLO) define a range of acceptable values from an SLI within which the service is considered to be in a healthy state. – 9999?
Example: Throughput

Assume that an I/O operation at a disk in an OLTP system takes 10 msec on average. If the disk is constantly busy (i.e., its utilization is 100%), then it will be executing I/O operations continuously at a rate of one I/O operation every 10 msec or 0.01 sec. So, what is the maximum throughput of the disk?
Example: Throughput

Assume that an I/O operation at a disk in an OLTP system takes 10 msec on average. If the disk is constantly busy (i.e., its utilization is 100%), then it will be executing I/O operations continuously at a rate of one I/O operation every 10 msec or 0.01 sec. So, what is the maximum throughput of the disk?

$$100 \left(= \frac{1}{.01}\right) \text{ I/Os per second.}$$

But if the rate at which I/O requests are submitted to the disk is less than 100 requests/sec, then its throughput will be equal to the rate at which requests are submitted. This leads to the expression

$$\text{throughput} = \text{minimum} \ [\text{servercapacity}, \text{offeredworkload}]$$
Example: Availability

If the availability of a system is 99.99% over a period of thirty days, how long the system was unavailable?
Example: Availability

If the availability of a system is 99.99% over a period of thirty days, how long the system was unavailable?

\[(1-0.9999) \times 30 \text{ days} \times 24 \text{ hours/day} \times 60 \text{ min/hr} = 4.32 \text{ minutes}\]
Alerts

Essence

• Alerting is the part of a monitoring system that triggers an action when a specific condition happens, e.g., like crossing a threshold.

• Depending on the severity and type of alert, the action triggered can range from running some automation, like restarting a service instance, to ringing the phone of a human operator (aka administrator) who is on-call. Human operators are still a fundamental part of operation a service.

• Alerts are presented in dashboards (decision making interfaces)
## Observability

### Essence

- Observability is a set of tools that provide granular insights into a system in production, allowing us to understand its emergent behaviors.
- Besides telemetry sources like metrics, event logs and traces can be used to observe a system.
- Metrics are mainly used for monitoring, while event logs and traces mainly for debugging.

### Insight

A distributed system is never 100% healthy at any given time as there can always be something failing.
Observability

Logs

A log is an immutable list of time-stamped events that happened over time. An event can have different formats. Its simplest form, it’s just free-form text.

```javascript
var log = log4js.getLogger("app");

var routes = require('./routes/index');
var users = require('./routes/users');

var app = express();

// development error handler
// will print stacktrace
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    log.error("Something went wrong:", err);
    res.status(err.status || 500);
    res.render('error', {
      message: err.message,
      error: err
    });
  });
}
```
Observability

Trace

- Tracing captures the entire lifespan of a request as it propagates throughout the services of a distributed system.
- A trace is a list of casually-related spans that represent the execution flow of a request in a system.
- A span represents an interval of time that maps to a logical operation or work unit, and contains a bag of key-value pairs.
Observability

Execution flow of a transaction depicted with spans
Modelling server performance

Models

• Approximation
• Simulation
• Analytic
• And others (do not rely on intuitive ones)

Note

• A model should not be made more complex than is necessary to achieve its goals.
## Modelling server performance

### Analytic

A distributed computer system is composed of a collection of resources, where each resource usage is regulated by a queue.

- Network of queues, or Queuing Network (QN)
- Request/transaction/process $\rightarrow$ Customer

### Essence

- Analytic models are composed of a set of formular and/or computational algorithms that provide the values of desired performance measures as a function of a set of input workload parameters.
Modelling server performance

(a) Single queue with one resource server (b) Single queue with m resource servers.

Service demand

The total average service time of a transaction (single class) is called its service demand.
Modelling server performance
Modelling server performance

Example: Queuing network for a simple database server
Modelling multiple request types

**Essence**
The workload consists of different types of transactions (Multiclass QN model)

**Characteristics**

**Heterogeneous service demands:** the requests that form the workload can be clustered into groups

**Different types of workload:** the requests in the workload are different in nature

**Different service level objectives:** the requirements of each group of requests is different
Modelling multiple request types

Example: Summary statistics for the database server

<table>
<thead>
<tr>
<th>Transaction Group</th>
<th>Percentage of Total</th>
<th>Average CPU Time (sec)</th>
<th>Avg. Number of I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trivial</td>
<td>45%</td>
<td>0.04</td>
<td>5.5</td>
</tr>
<tr>
<td>Medium</td>
<td>25%</td>
<td>0.18</td>
<td>28.9</td>
</tr>
<tr>
<td>Complex</td>
<td>30%</td>
<td>1.20</td>
<td>85.0</td>
</tr>
</tbody>
</table>

Overall arrival rate = 1.5 tps
Arrival rate per type of request? (class)
Modelling multiple request types

Example: Summary statistics for the database server

<table>
<thead>
<tr>
<th>Transaction Group</th>
<th>Percentage of Total</th>
<th>Average CPU Time (sec)</th>
<th>Avg. Number of I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trivial</td>
<td>45%</td>
<td>0.04</td>
<td>5.5</td>
</tr>
<tr>
<td>Medium</td>
<td>25%</td>
<td>0.18</td>
<td>28.9</td>
</tr>
<tr>
<td>Complex</td>
<td>30%</td>
<td>1.20</td>
<td>85.0</td>
</tr>
</tbody>
</table>

Overall arrival rate = 1.5 tps
Arrival rate per type of request? (class)

0.675 (=1.5 x 0.45) tps, 0.375 (=1.5 x 0.25) tps, 0.45 (=1.5 x 0.30) tps
## Modelling other request types

### Essence

#### Open class
- Workload intensity is specified by an arrival rate
- Unbounded number of customers in the system
- Throughput is an input parameter

#### Closed class
- Workload intensity specified by the customer population (not transactions, but batch jobs)
- Bounded and known number of customers in the system
- Throughput is an output parameter
Modelling other request types

Example: Queuing network for a database server with closed workload
Performance goals of servers based on different metrics, including response time, throughput and availability, among others.

Example: SLAs for the Database Server

- 99.99% availability during 8:00 am – 11:00 pm, and 99.9% at other times
- Minimum throughput of 2,000 page downloads per second

<table>
<thead>
<tr>
<th>Transaction Group</th>
<th>Maximum Average Response Time (sec)</th>
<th>Minimum Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trivial</td>
<td>1.2</td>
<td>-</td>
</tr>
<tr>
<td>Medium</td>
<td>2.5</td>
<td>-</td>
</tr>
<tr>
<td>Complex</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>Batch Reports</td>
<td>-</td>
<td>20 per hour</td>
</tr>
</tbody>
</table>
Mixed classes

Mixed queuing network for a database server
Modelling different types of resources

Types of resources

**Load independent (LI):** these resources have a constant service rate that does not dependent on the load (e.g., CPU, disk)

**Load dependent (LD):** the service rate is a function of the number of requests in the queue (e.g., LAN)

**Delay (D):** there is no waiting line. A request that arrives at a delay resource is served immediately (e.g., client)
Modelling different types of resources

QN for client/server applications (database server with clients and LAN)

No queue (think/wait)

Arrow across
Example: Summary Statistics for the Database Server

<table>
<thead>
<tr>
<th>Transaction Group</th>
<th>Percentage of Total</th>
<th>Average CPU Time (sec)</th>
<th>Avg. Number of I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trivial</td>
<td>45%</td>
<td>0.04</td>
<td>5.5</td>
</tr>
<tr>
<td>Medium</td>
<td>25%</td>
<td>0.18</td>
<td>28.9</td>
</tr>
<tr>
<td>Complex</td>
<td>30%</td>
<td>1.20</td>
<td>85.0</td>
</tr>
</tbody>
</table>

Overall arrival rate = 1.5 tps
Arrival rate per class = 0.675 (=1.5 x 0.45) tps, 0.375 (=1.5 x 0.25) tps, 0.45 (=1.5 x 0.30) tps

Average I/O time = 0.01 seconds
Service demand at disk per each class?
Modelling different types of resources

Example: Summary Statistics for the Database Server

*Service demand at disk per each class?*

*Service demand = average number of I/Os x average time per I/O*

<table>
<thead>
<tr>
<th>Open QN</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class (r)</td>
<td>Type</td>
<td>( \lambda_r ) (tps)</td>
</tr>
<tr>
<td>1 (Trivial)</td>
<td>open</td>
<td>0.675</td>
</tr>
<tr>
<td>2 (Medium)</td>
<td>open</td>
<td>0.375</td>
</tr>
<tr>
<td>3 (Complex)</td>
<td>open</td>
<td>0.450</td>
</tr>
</tbody>
</table>
Summary

• Explored the important of system monitoring over time.
• Learned the performance metrics that can be extracted from a system
• Studied the modelling of system performance
References

Part of this material is inspired by:

- Understanding Distributed Systems, Version 1.1.1., Roberto Vitillo, 2021
Next lecture
Planning
Questions?

E-mail: huber.flores@ut.ee