End-to-End Service Performance and Dependability Analytics

Hong-Linh Truong
Faculty of Informatics, TU Wien

hong-linh.truong@tuwien.ac.at
http://www.infosys.tuwien.ac.at/staff/truong
Outline

- Fundamental
- End-to-end dependability in hybrid computing systems
- Summary
Fundamental properties of a system
- Functionality
- Performance, dependability, security, cost
  - Called non-functional properties
- Usability, manageability, adaptability/elasticity

Structure of a system
- A set of composite and atomic components
- A composite component is composed of a set of components
Clients require “correct service” w.r.t function and non-functional properties in an end-to-end view.

Non-functional properties about performance, dependability, security and cost can be very subjective.

System behavior

Example of service behavior

Normal: based on the service specification and design
Failure classification

Type of failures

- Crash failures
- Omission failures
- Timing failures
- Response failures
- Arbitrary/Byzantine failures
Quality of service improvement

Industry view: https://guidingmetrics.com/content/cloud-services-industrys-10-most-critical-metrics/

SOCloud 2017
Dealing with service failures and quality

- Determines clearly **system boundaries**
  - The system under study, the system used to judge, and the environment

- **Understands dependencies**, e.g.
  - Among components in distributed systems
  - Single layer as well as cross-layered dependencies

- Determines **types of metrics and failures** and break down problems along the dependency path

Performance metrics

- **Timing behaviors**
  - Communication
    - Latency/Transfer time
    - Data transfer rate, bandwidth
  - Processing
    - Response time
    - Throughput

- **Utilization**
  - Network utilization
  - CPU utilization
  - Service utilization

- **Efficiency**

- **Data quality**

---

Examples

![Diagram showing client-server interaction with timing behaviors and metrics highlighted]
Measurement, Monitoring and Analysis

- **Instrumentation and Sampling**
  - Instrumentation: insert probes into systems so that you can measure system behaviors directly
  - Sampling: use components to take samples of system behaviors

- **Monitoring**
  - Probes or components perform sampling or measurements, storing and sharing measurements

- **Analysis**
  - Evaluate and interpret measurements for specific contexts
  - Can be subjective!
Composable methods and views

- Composable method
  - Divide a complex structure into basic common structures
  - Each basic structure has different ways to analyze specific failures/metrics
- Interpretation based on context/view
  - Client view or service provider view?
  - Conformity versus specific requirement assessment

Dependency Structure:
- Client: Server is failed
- Provider: OK
- Slow
Performability

- What happens if the performance is unacceptable, e.g., the service cannot be scaled, the service is unreliable
- Technically, the system may still deliver its function
  - it may fail to deliver the expected non-functional properties as well as its function may fail eventually
- Performability measures a system performance and its dependability
  - Performance is currently not an attribute of dependability
Dependability Attributes, Threats and Means

Dependability and Security
  Attributes
    Availability
    Reliability
    Safety
    Confidentiality
    Integrity
    Maintainability
  Threats
    Faults
    Errors
    Failures
  Means
    Fault Prevention
    Fault Tolerance
    Fault Removal
    Fault Forecasting

Sub mechanisms of resilience techniques


Personal note: Performance should be an attribute as well!
System View

- Smart Cities
- IoT applications
- Predictive maintenance
- System optimization

Application

Machine-based Compute Units

- Big data analytics
- Cloud services

Human-based Compute Units

- IoT infrastructures
- Sensing and actuating

Compute Units Collective

- Crowdsourcing platforms
- Collective intelligence
- Social networks of experts
- On-premise domain experts


SOCloud 2017
Scenario/Application View

Middleware View

Cloud-based M2M Platform

Sensors → Gateways → Sensor data → Load Balancer → EventHandling Web Service → Sensor data → NoSQL BigData

Control, analyze, deploy

COMOT – Elasticity Management Platform

Near-Realtime Data Analytics

Offline Data Analytics

CloudLyra

Critical situation msg

Critical situation msg

Critical situation msg

Critical situation msg

Control
Analysis
Configuration

Smart Communication

Collective Provisioning

Human Interface
(Mobile, Mail, Web)

Human-based Services

Hybrid compute units provisioning platform

External Service

critical situation msg

critical situation msg

critical situation msg

critical situation msg/task

task

forms

Hybrid Cloud Computing 2017
View: end-to-end resource slice

CPS Applications/Virtual infrastructures

End-to-end Resource slice

http://sincconcept.github.io/
End-to-end dependability

- What does it mean end-to-end? Examples?
  - Reflect the entire system
  - E.g., data reliability: from sensors to the final analytics results

- The user expects end-to-end dependability
  - E.g., specified in the expected QoR

- Providers/operators want to guarantee end-to-end dependability
  - Need to monitor different parts, each has subsystems/components
  - Coordination-aware dependability assurance
    - Autonomic and/or elasticity principles
Dependability in the cloud

- Infrastructure dependability and software dependability
- Fact: failures are inevitable! Why?
  - A lot of customers with different requirements
  - A lot data and services
  - Software are developed and deployed in a very short cycle.
- Design perspective: accept failures and think how to deal with failures


Hiranya Jayathilaka, Chandra Krintz, Rich Wolski:
Performance Monitoring and Root Cause Analysis for Cloud-hosted Web Applications. WWW 2017: 469-478
Dealing with dependability problems

- Traditional techniques
  - E.g., Replication & Redundancy
- Virtualization
  - Hide dependability problems and allow quick recovery through virtualization techniques
- Elasticity
  - Compensate dependability problems with elasticity of resources, costs and quality

Dependability for IoT

- Different levels
  - Infrastructures: Things and communication networks
  - Software: sensors, gateways, and actuators,
- Computation dependability versus protocols versus data dependability
- Well-established work
  - Network dependability
- Not understood well
  - Data dependability and its impacts

Monitoring Tools (1)

From metrics to insight
Power your metrics and alerting with a leading open-source monitoring solution.

Prometheus v2.0 is available now — Read the announcement blog post!

- **Dimensional data**
  Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

- **Powerful queries**
  A flexible query language allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

- **Great visualization**
  Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

- **Efficient storage**
  Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

- **Simple operation**
  Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.

- **Precise alerting**
  Alerts are defined based on Prometheus’s flexible query language and maintain dimensional information. An alertmanager handles notifications and silencing.

- **Many client libraries**
  Client libraries allow easy instrumentation of services. Over ten languages are supported already and custom libraries are easy to implement.

- **Many integrations**
  Existing exporters allow bringing of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.

From: https://prometheus.io/
Monitoring Tools (2)

From: https://www.fluentd.org/

SOCloud 2017 23
Monitoring Tools (3)

It is part of the TICK stack and is a plugin-driven server agent for collecting and reporting metrics. Telegraf has plugins or integrations to source a variety of metrics directly from the system it's running on, pull metrics from third-party APIs, or even listen for metrics via a StatsD and Kafka consumer services. It also has output plugins to send metrics to a variety of other datastores, services, and message queues, including InfluxDB, Graphite, OpenTSDB, Datadog, Librato, Kafka, MQTT, NSQ, and many others.

From: https://www.influxdata.com/
Monitoring Tools (4)

From: https://www.elastic.co/
Your next assignment

1. Tools and end-to-end metrics for IoT Cloud Systems
   - Which are interesting metrics (and tools for analysis)
   - Technical debt and performance/dependability
   - Instrumentation and (micro)service/system engineering

2. Too much monitoring data
   - How can machine learning help for performance and dependability of IoT Cloud systems?
   - How can big data analytics help for performance and dependability analysis of IoT Cloud systems?
Thanks for your attention

Hong-Linh Truong
Faculty of Informatics, TU Wien
hong-linh.truong@tuwien.ac.at
http://www.infosys.tuwien.ac.at/staff/truong