Carnegie Mellon University

Observability

Outline

- **Introduction**
- Utilization
- Logging
- Tracing

Observability

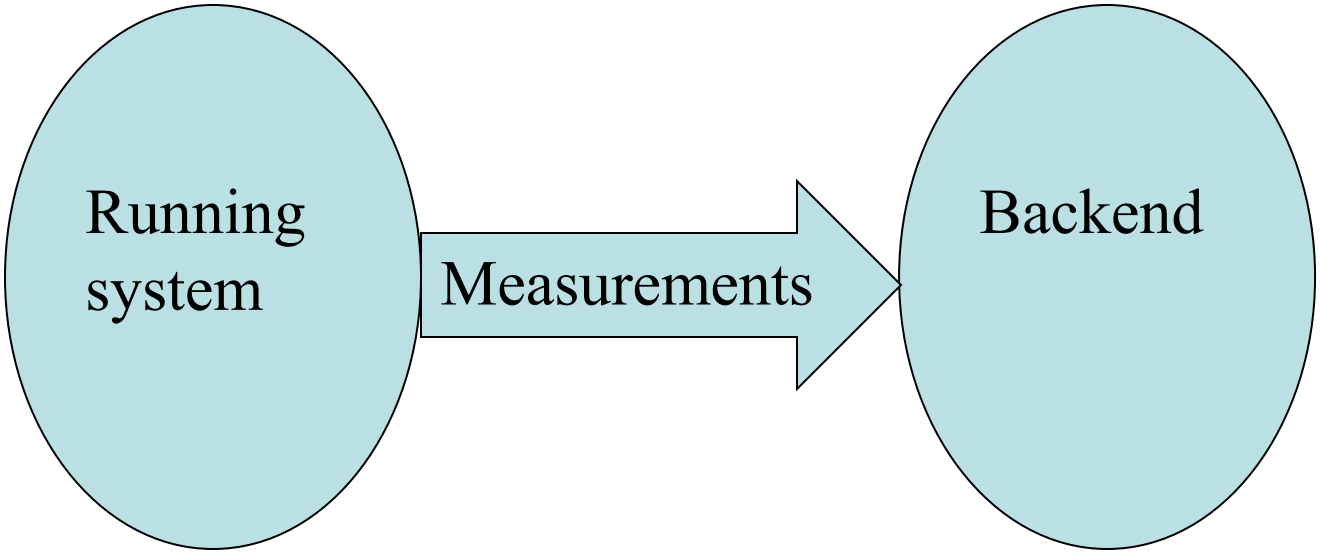
- The extent to which you can understand the internal state or condition of a system based only on knowledge of its external outputs

- Two aspects
  - Collecting data - metrics, traces and logs
  - Displaying data – alerts, dashboards

Purposes of gathering data

- After a system is in operation, information is gathered for three purposes:
  - Alerting – Detecting that there is a problem
  - Forensics – determining what caused a problem
  - Improvement – finding bottlenecks in systems or determining causes of internet traffic.

General picture



Measurements are taken from a running system and its environment and sent to a back end.

Back end

- The back end has a database (usually a time series database). It
  - Generates alerts
  - Generates reports
  - Allows drilling down into aggregate information to get more detailed information
  - Has a dashboard to give fast indication of problems.

Outline

- Introduction
- **Utilization**
- Logging
- Tracing

Utilization

- Utilization is measure of some activity over some period of time
- Collected automatically by infrastructure over externally visible activities of VM or container
  - CPU – percentage busy
  - I/O – amount of traffic
  - etc
- Collected by back end from infrastructure

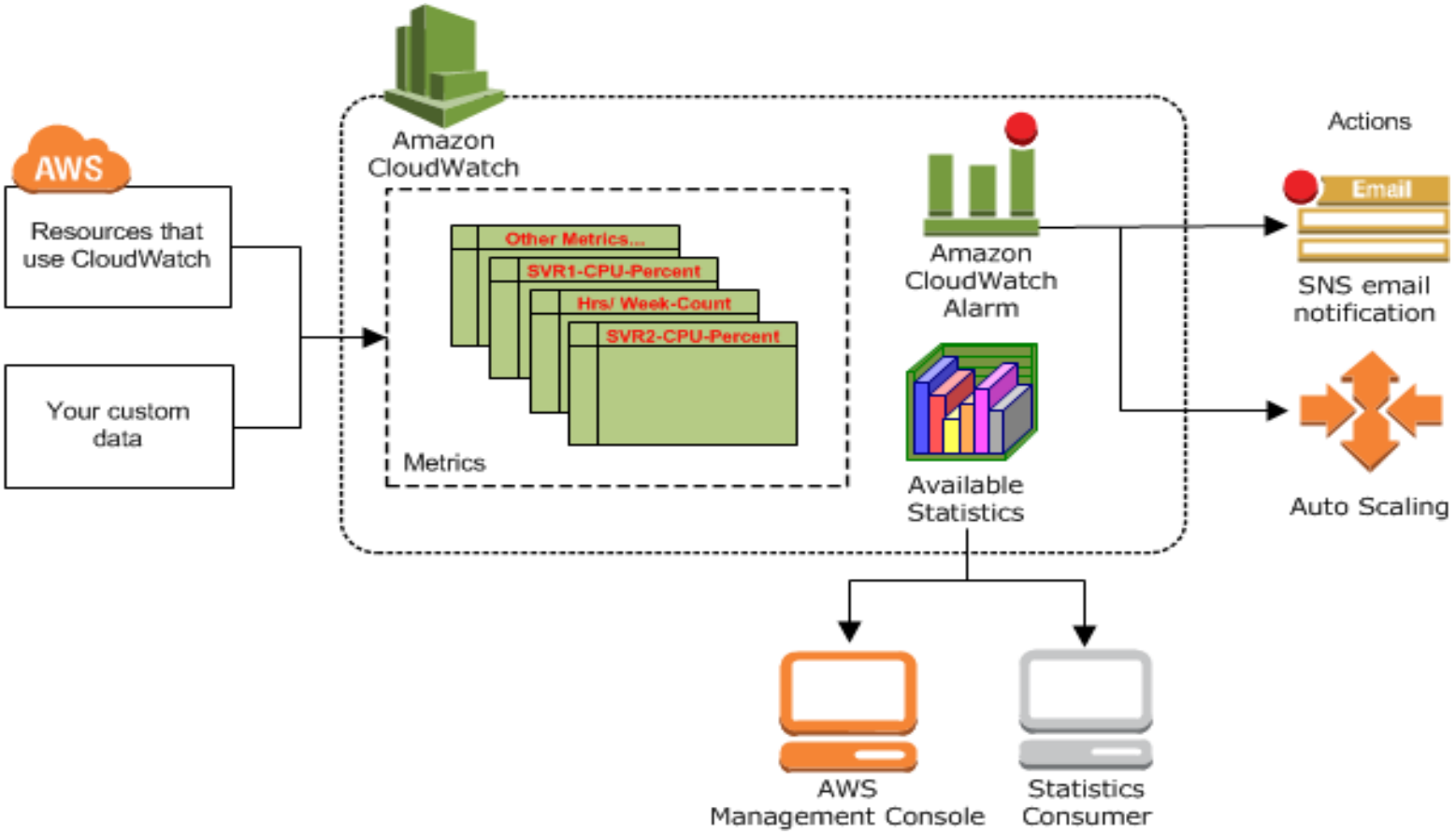
Utilization measures

- Taken over period of time
- Instantaneous measurement is not meaningful.
  - At any instant a CPU is either busy or idle.
  - You want to know whether the CPU is overloaded. I.e. what percentage of the last five minutes (for example) was the CPU busy

Alerts

- Back end has set of rules to establish when to send an alert. E.g. alert if CPU utilization is over 80% for 15 minutes.
- Utilization numbers are bursty. The period must be sufficiently long to indicate problem.
- False positives and false negatives are both problems.
- An alert causes a page to be sent.

AWS Cloudwatch



S3D

Outline

- Introduction
- Utilization
- Logging
- Tracing

Logs

- A log is an append only data structure
- Written by each software application.
- Located in a fixed directory within the operating system
- Enumerates events from within application
  - Entry/exit and associated parameters
  - Troubleshooting
  - DB modifications
  - …

Log Daemon

- A daemon resides on each server.
- It copies logs generated by applications on that server to back end.
- Logs can be voluminous and so daemon may also clean off log file periodically – once the contents have been sent to the back end.

What happened?

- Logs are used to determine what happened.
- Either
  - immediately after an alert to get the system working again or
  - later to determine frequent types of problems and their causes

Outline

- Introduction
- Utilization
- Logging
- **Tracing**

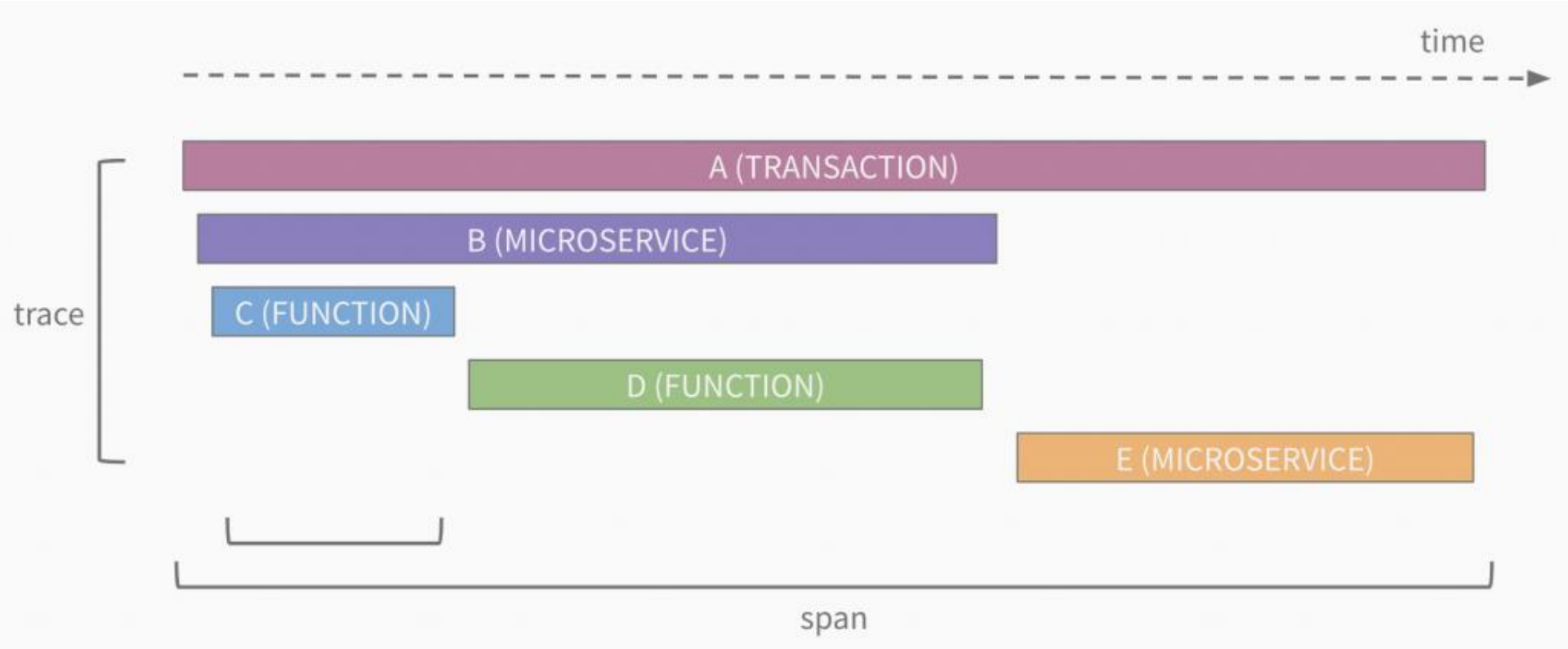
Tracing

- Tracing provides an end to end picture of a request.
  - Utilization helps determine whether there is a problem.
  - Logging provides picture of a single event
  - Tracing provides a sequence of events
- Tracing is used to understand the big picture

Spans

- A trace captures the end to end actions in response to a user request.
- A span is a named, timed operation that represents a piece of the trace.
- Spans may have child spans
- Displaying spans on a time axis allows you to see:
  - Parallelism
  - Where time is being spent

Sample span display*



*https://sflanders.net/2019/03/28/an-intro-to-distributed-tracing/

S3D

How does tracing work? - 1

- This is one of many possible implementations.
- Request enters the system from external source – user or external system
- Request is given a unique ID that reflects the context
- Context description is kept in a data base so that with context ID analyst can know details of the context.

How does tracing work? – 2

- Context id becomes a portion of HTTP header. World Wide Web Consortium is standardizing how this will work.
- The context ID is inserted by the HTTP server accepting the request and propagated by every service as it fans out the request. This is transparent to the requester.
- Context can be used to control behavior of a service.

Examples of context

- Test version. Use context to affect behavior or routing.
- Application. Google might want to know what percentage of their network traffic might be due to search, Gmail, etc
- Traffic prioritization. Give priority to certain requests to maintain quality of service.
- Bottleneck determination. Where is the most time being spent in a collection of transactions?

Summary

- Utilization is the measurement of how much the resources in an environment are being used.
- Logging is the recording of specific events and information associated with those events.
- Tracing ties together a sequence of services.