Desmistificando Microsserviços e DevOps: Projetando Arquiteturas Efetivamente Escaláveis

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: @vinicius3w :: assertlab.com

[IF1004] - Seminários em SI 3
https://github.com/vinicius3w/if1004-DevOps

ASSERT Lab
Advanced System and Software Engineering Research Technologies

Licença do material

Este Trabalho foi licenciado com uma Licença

Creative Commons - Atribuição-NãoComercial-CompartilhaIgual 3.0 Não Adaptada

Mais informações visite

http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt

Crosscuting Concerns

Monitoring

“With Adnene Guabtni and Kanchana Wickremasinghe

First get your facts; then you can distort them at your leisure.”
— Mark Twain

Introduction

- Software monitoring comprises myriad types of monitoring and the considerations that come with them
- Activities as varied as collecting metrics at various levels (resources/OS/middleware/application-level),
  - graphing and analyzing metrics,
  - logging,
  - generating alerts concerning system health status, and
  - measuring user interactions
- all are a portion of what is meant by monitoring

Introduction

- As Richard Hamming said: “The purpose of computing is insight, not numbers.”
- The insights available from monitoring fall into five different categories
    1. “Identifying failures and the associated faults both at runtime and during postmortems held after a failure has occurred
    2. Identifying performance problems of both individ“ual systems and collections of interacting systems
    3. Characterizing workload for both short- and long-term capacity planning and billing purposes
    4. Measuring user reactions to various types of interfaces or business offerings. We discussed A/B testing in Lectures 6 and 7
    5. Detecting intruders who are attempting to break into the system

Introduction

- We use the term monitoring to refer to the process of observing and recording system state changes and data flows
  - State changes can be expressed by direct measurement of the state or by logs recording updates that impact part of the state
  - Data flows can be captured by logging requests and responses between both internal components and external systems

Main challenges

- Monitoring under continuous changes is difficult
  - Traditional monitoring relies heavily on anomaly detection
  - You know the profile of your system during normal operation
  - You set thresholds on metrics and monitor to detect abnormal behavior
  - If your system changes, you may have to readjust them

Main challenges

- The cloud environment introduces different levels from application programming interface (API) calls to VM resource usage
  - Choosing between a top-down approach and a bottom-up approach for different scenarios and balancing the tradeoffs is not easy

Main challenges

- When adopting the microservice architecture, monitoring requires attention to more moving parts
  - It also requires logging more inter-service communication to ensure a user request traversing through a dozen services still meets your service level agreements
  - If anything goes wrong, you need to determine the cause through analysis of large volumes of (distributed) data

Main challenges

- Managing logs becomes a challenge in large-scale distributed systems
  - When you have hundreds or thousands of nodes, collecting all logs centrally becomes difficult or prohibitively expensive
  - Performing analysis on huge collections of logs is challenging as well, because of the sheer volume of logs, noise, and inconsistencies in logs from multiple independent sources

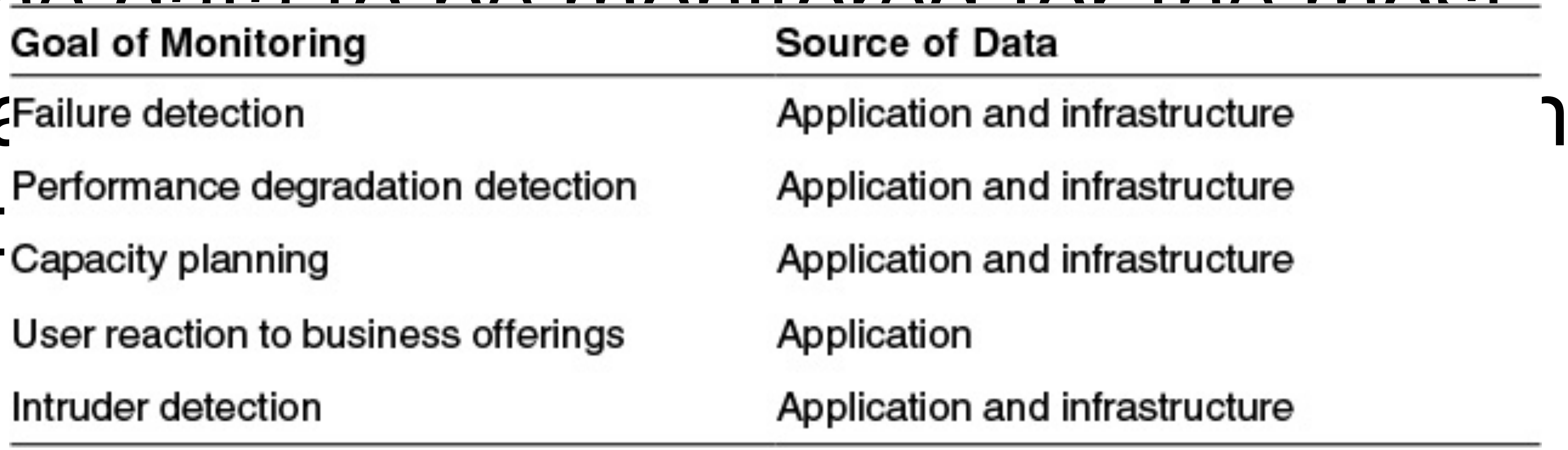
Testing

- Testing a monitoring solution in your various environments is one portion of the testing
- But the scale of your non-production environments may not approach the scale of your production
- Which implies that your monitoring environments may be only partially tested prior to being placed into production

What to Monitor

What to Monitor

- The data to be monitored for the most pa ne st



Goal of Monitoring	Source of Data
Failure detection	Application and infrastructure
Performance degradation detection	Application and infrastructure
Capacity planning	Application and infrastructure
User reaction to business offerings	Application
Intruder detection	Application and infrastructure

What to Monitor

- The fundamental items to be monitored consist of inputs, resources, and outcomes
- The resources can be hard resources such as CPU, memory, disk, and network—even if virtualized
- They can also be soft resources such as queues, thread pools, or configuration specifications
- The outcomes include items such as transactions and business-oriented activities