Opportunities & Challenges in Adopting Microservice Architecture for Enterprise Workloads

Shriram Rajagopalan, Priya Nagpurkar, Tamar Eilam, and Hani Jamjoom
IBM Watson Research

Etai Lev-Ran, and Vita Bortnikov
IBM Research, Haifa

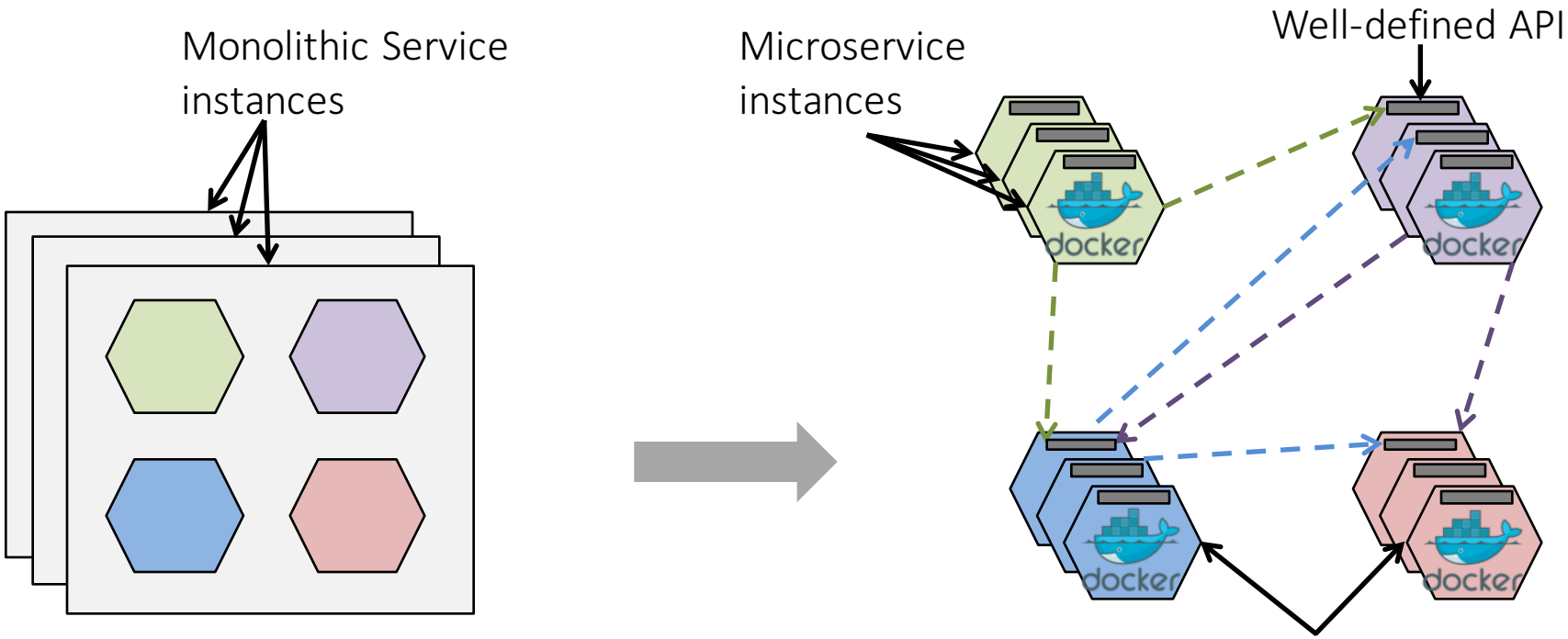
Frank Budinsky
IBM

Contact: shriram@us.ibm.com

- Emergence of microservices & DevOps
- Challenges & Opportunities
- Adopting a SDN perspective of microservices
- Version/Content-aware routing
- Systematic resilience testing

- **Emergence of microservices & DevOps**
- Challenges & Opportunities
- Adopting a SDN perspective of microservices
- Version/Content-aware routing
- Systematic resilience testing

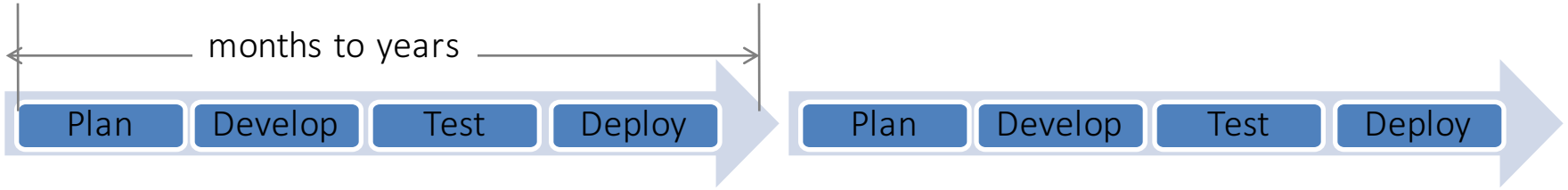
From Monoliths to Microservices



- A single service serves multiple purposes
- **Tight-coupling** across services

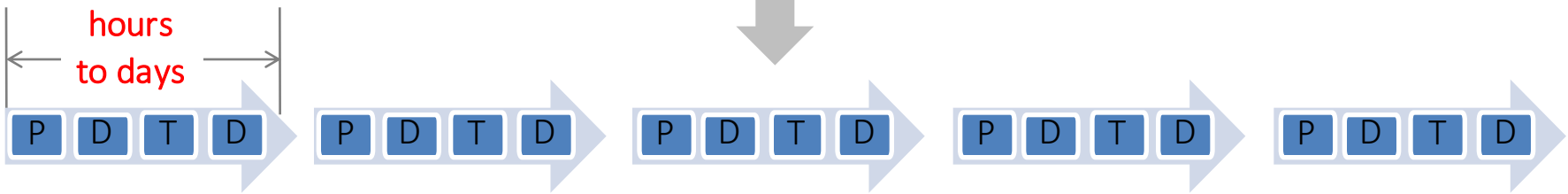
- Each service serves a single purpose (functionality)
- Many **loosely-coupled** microservices communicate over the network

From Waterfall to DevOps



Features, performance improvements, bug fixes, etc., are **periodically** delivered as one **big update**

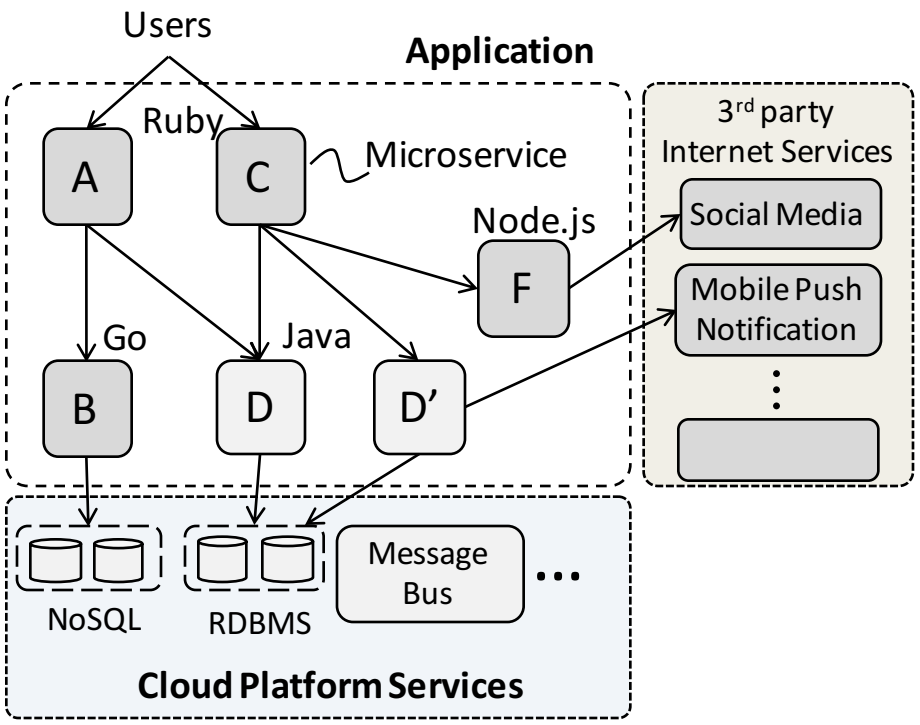
Culture + Automation + Instrumentation



Continuous delivery of incremental updates

Emphasizes constant experimentation & feedback-driven development

Microservices + DevOps



- Polyglot applications with loosely-coupled microservices
- Small "*two pizza*" teams per microservice
  - Autonomy & accountability
  - Own the roadmap for the feature/service
  - Independent launch schedules
    - Develop, deploy, scale
  - "*You build it, you run it*"
- **10s to 100s of deployments a day across the application**
  - E.g., Orbitz, GrubHub, HubSpot
- **Multiple versions co-exist simultaneously**

“Traditional” Enterprises are moving or have moved to Microservices + DevOps

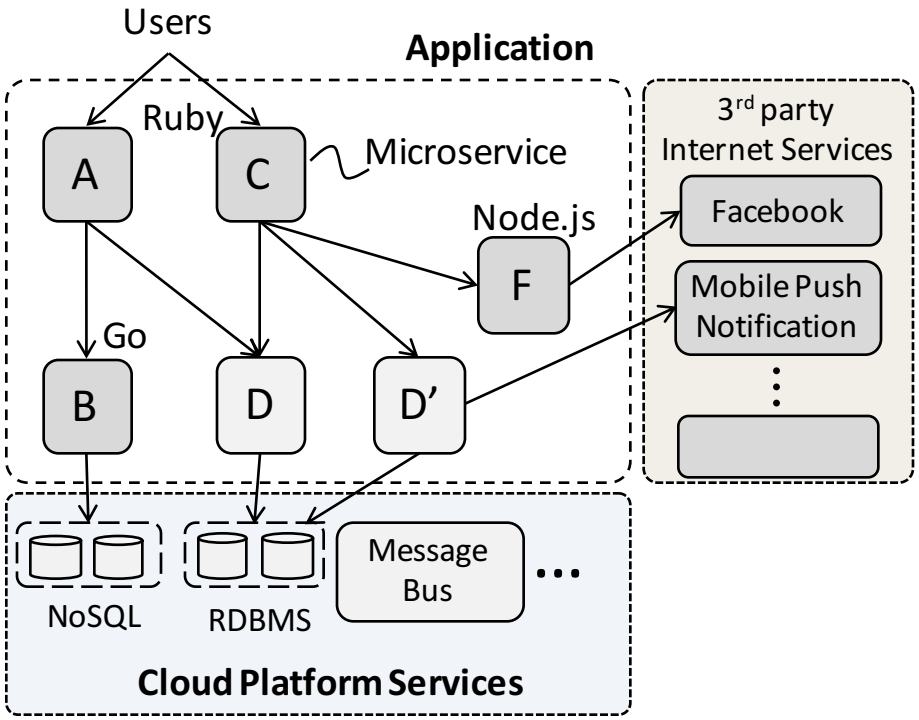
- Emergence of microservices & DevOps
- **Challenges & Opportunities**
- Adopting a SDN perspective of microservices
- Version/Content-aware routing
- Systematic resilience testing

Opportunities

- Enterprises are
  - Re-architecting legacy applications to microservice architecture
  - Developing in-house platforms to host sensitive apps on premise
    - E.g. Fidelity's Mako
  - Still experimenting with different design alternatives
  - Heavily leveraging open-source technologies

- Opportunity for the research community to engage
  - Influence infrastructure & application design
  - Integrate ideas into open-source platforms and solutions

Challenges



- 10s to 100s of deployments a day across the application
- Multiple versions co-exist simultaneously
- *Complexity shifted to the network and orchestration across services*
- *Cascading failures despite the microservices being designed for failure*

Ad-hoc Designs & Implementations

- Two Options:

- Adopt open-source frameworks from large scale internet applications (e.g., Netflix OSS)
  - These frameworks are point solutions that fit the needs & environment of the companies that operate these applications (e.g., Java only support)

- Shoehorn the service-oriented web application into clustering frameworks like Kubernetes, Marathon, etc., and write ad-hoc tools on top to control the microservices

- Emergence of microservices & DevOps
- Challenges & Opportunities
- **Adopting a SDN perspective of microservices**
- Version/Content-aware routing
- Systematic resilience testing

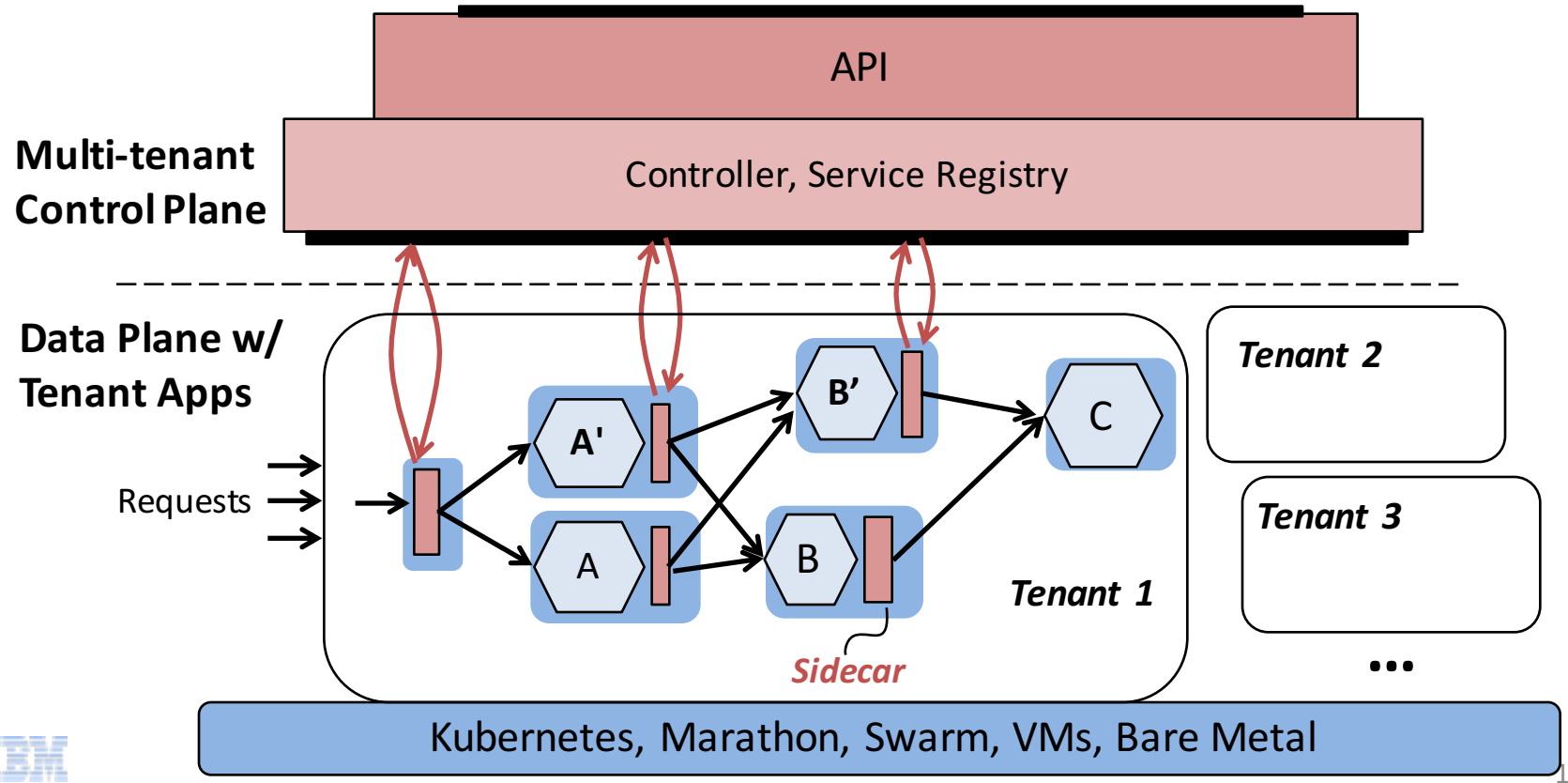
Microservice Application Requirements

- Integration
  - Service registration & discovery
  - Load balancing of requests across microservice instances
- Version & content-aware routing
  - Hypothesis driven-development (i.e. A/B testing)
  - Canary deployments (feature release to % of users)
  - Red/Black deployments (gradual rollout to all users)
  - Etc.
- Operational testing in production
  - E.g., does failure recovery work as expected?

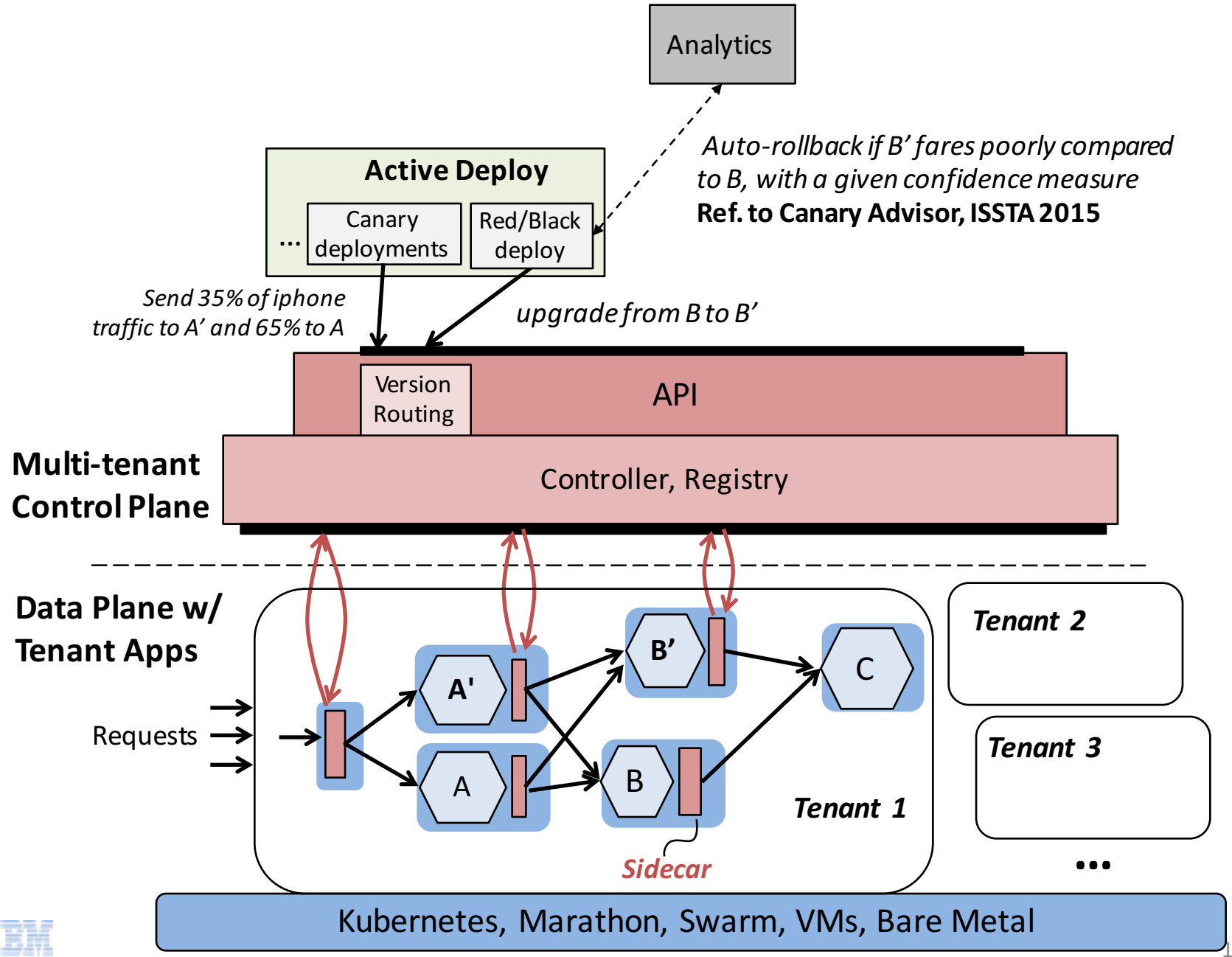
Introducing Amalgam8

- Observation:
  - Microservices interact only over the network predominantly using HTTP(s)
  - Existing solutions lack the ability to dynamically control the routing of requests between two microservices
- Insight:
  - Think of requests as packets and microservices as switches
  - A Layer-7 SDN will simplify integration and routing
- Design:
  - Sidecar: A programmable layer-7 proxy process attached to each microservice
  - Controller: The equivalent of an SDN controller, except at Layer-7

Simplifying Integration



- Emergence of microservices & DevOps
- Challenges & Opportunities
- Adopting a SDN perspective of microservices
- **Version/Content-aware routing**
- Systematic resilience testing


Analytics
Auto-rollback if B' fares poorly compared to B, with a given confidence measure
Ref. to Canary Advisor, ISSTA 2015
Active Deploy
Canary deployments
Red/Black deploy
Send 35% of iphone traffic to A' and 65% to A
upgrade from B to B'
Version Routing
API
Multi-tenant Control Plane
Controller, Registry
Data Plane w/ Tenant Apps
Requests
A'
A
B'
B
C
Tenant 1
Sidecar
Tenant 2
Tenant 3
...
Kubernetes, Marathon, Swarm, VMs, Bare Metal

- Emergence of microservices & DevOps
- Challenges & Opportunities
- Adopting a SDN perspective of microservices
- Version/Content-aware routing
- **Systematic resilience testing**

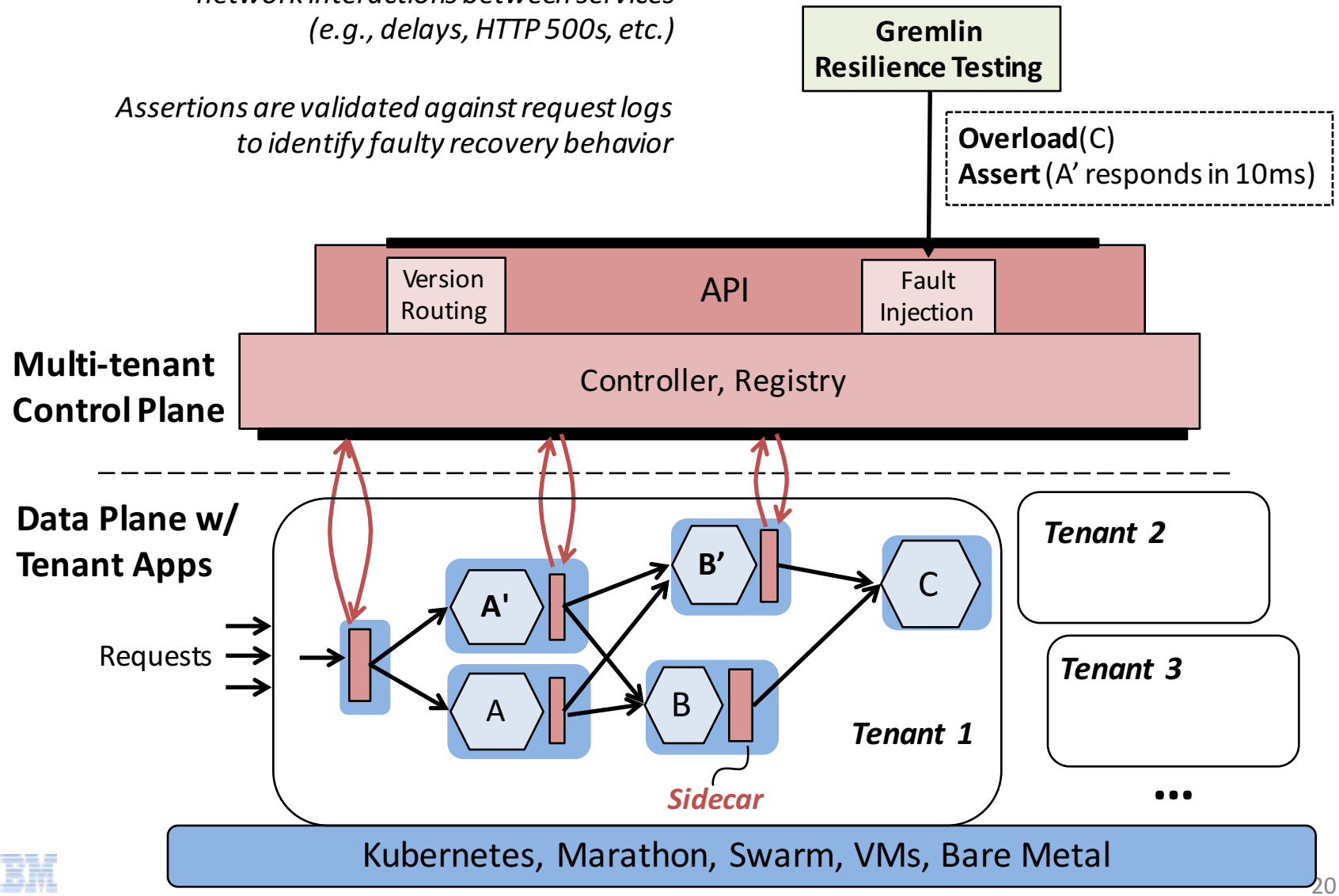
Resilience Testing

- Microservices designed but "seldom" tested for failures
- Randomized fault injection (e.g., Netflix Chaos Monkey) is insufficient
  - Manual effort to validate whether application recovered properly or not
- Gremlin – systematic resilience testing
  - Script failure scenarios and expectations
  - Faults injected from the network
  - Run assertions on the logs to validate expectations
  - Exposes faulty recovery behavior, conflicting failure handling policies across services, etc.

Ref. to Gremlin, ICDCS 2016

Failures are emulated by manipulating network interactions between services (e.g., delays, HTTP 500s, etc.)

Assertions are validated against request logs to identify faulty recovery behavior



Thank You

- https://amalgam8.io
- https://github.com/amalgam8/examples

Backup

Research Challenges in the Face of Continuous Change

- Managing stateful services and data stores
- Problem determination gains many dimensions
  - The problem may not just be in your code
  - Many dimensions change simultaneously such as infrastructure, runtime, etc.
  - Can we pinpoint the issue down to the Git commit by correlating runtime logs and development history?
- Too much data, too little insights
  - Logs emitted by all layers of the software stack, by automated build tools, etc.
  - Yet, we are no where close to pinpointing the problem and fixing it when things go wrong!

Opportunities to Fix Issues Before They Occur

- Software build, test and deployment phases are completely automated
- Provides a unique opportunity to catch security vulnerabilities, buggy implementations, etc., even before software is deployed
- However, existing tools and techniques do not scale to the extreme code churn (100s of deployments)