RailCloud:
A Reliable PaaS Cloud for Railway Applications

Bijun Li, Rüdiger Kapitza
TU Braunschweig
06.10.2016

This work is supported by Siemens international Rail Automation Graduate School (iRAGS)
RailCloud

- A PaaS cloud for railway applications
- Shared by small and medium-sized transportation companies
- Reliability and safety guarantee
PaaS Cloud in a Nutshell

What is a PaaS Cloud?
- A computing platform for software development delivered over the Internet

How to use it?
- Software developers can quickly deploy applications, without infrastructure management tasks

Existing PaaS Clouds?
- Google App Engine (GAE), Microsoft Azure, OpenShift, Cloud Foundry etc.
- Recent evolvement with container technology
Existing PaaS Clouds?

Reliability?

Horizontal Scalability
- Usually for load balancing
- Tolerate crash-stop failures
- Mostly for stateless applications

Issues
- Lack of support for replicated stateful applications
- Cannot tolerate malicious attackers
- Complex deployment and coordination for cloud customers

Goal of RailCloud
- Easy deployment of replicated stateful applications with automatic coordination to guarantee reliability
- Easy deployment of legacy railway applications
Outline

- Reliability in PaaS Clouds
- RailCloud Design
  - Byzantine Fault-Tolerant Applications in the Cloud
  - Trusted Proxy: Making Replicated Systems Transparent
- Conclusion and Future Work
RailCloud Architecture

Request/response Flow
BFT Message Exchanges
Application Deployment
Built-in Service of RailCloud

Client 1

Client 2

Host 1

Host 2

Host 3

Host 4

Developer

06.10.2016 | Bijun Li | RailCloud: A Reliable PaaS Cloud for Railway Applications | Page 6
Byzantine Fault Tolerance in the Cloud

Byzantine Fault-Tolerance (BFT) Protocols
- Tolerate crash-stop failures and arbitrary and malicious behaviors
- $3f + 1$ replicas to tolerate $f$ faults
- Message exchanges for agreement

Integration of BFT into Cloud Infrastructures
- Infrastructure level: Depsky, Fitch, TClouds etc.
- Middleware level: Thema etc.

RailCloud: PaaS Level + Automatic Deployment Extension
Base of RailCloud

OpenShift Origin v3

- Docker container packaging
- Kubernetes container cluster management
- Application lifecycle management
Implementation and Deployment

BFT Service Layer
- BFT image (BFT-SMaRt)
- BFT pods
- BFT services

Application Deployment

Networking
- Connect each BFT service to application service
- Expose BFT services
Trusted Proxy: Making Replicated Systems Transparent

Why transparent?

Minimum modifications to clients
- HTTPS connections
- Web-based railway applications
- Implements client-side BFT library

Friendly to low-bandwidth clients
- No redundant requests/replies

Hide details of replicated system
- Simple and secure interface to clients

Throughput improvement
Trusted Proxy in RailCloud

- **Message Flow**
- **Read Optimization**

**Host 1**
- Trusted Proxy
  - En/decrypt
  - Reply vote
  - (Fast read cache)
- Connection Handler
- Ordering + Execution

**Host 2**
- Trusted Proxy
  - En/decrypt
  - Reply vote
  - (Fast read cache)
- Connection Handler
- Ordering + Execution

**Host 3**
- Trusted Proxy
  - En/decrypt
  - Reply vote
  - (Fast read cache)
- Connection Handler
- Ordering + Execution

**Host 4**
- Trusted Proxy
  - En/decrypt
  - Reply vote
  - (Fast read cache)
- Connection Handler
- Ordering + Execution

Client
- Secure connection

Client
- Secure connection
Outline

- Reliability in PaaS Clouds
- RailCloud Design
  - Byzantine Fault-Tolerant Applications in the Cloud
  - Trusted Proxy: Making Replicated Systems Transparent
- Conclusion and Future Work
Conclusion and Future Work

RailCloud
- Integrate BFT for reliability demands into PaaS Cloud
- Easy deployment of BFT applications
- Automatic coordination among replicated services
- Using trusted proxy to make replicated system transparent

Future Work
- Explore more functions of trusted proxy
Appendix

Related Works


