Fibonacci: JD’s Function as a Service Platform

Yuan Chen, Xin Tong, Hui Tu, Dongdong Dai, Junyuan Zeng, Fuze Sun
JD.com
About JD.com

China’s largest retailer, online or offline
- The third largest Internet company by revenue
- Over 300 million active users

A Fortune Global 200 company

Largest nationwide e-commerce logistics infrastructure in China
- Covering 99% of the population
- Able to deliver 90% of orders same- or next-day

Strategic partnerships

- Walmart
- Google
- Tencent
JD Technological Infrastructure

Provide and manage containerized infrastructure and platform for JD retail, finance and logistics businesses

• Everything in containers
• One of the largest Kubernetes clusters in the world
  – Kubernetes since January 2016
  – Tens of thousands of physical servers and hundreds of thousands of containers
  – Multiple clusters across geo-distributed data centers, max cluster with 9000+ nodes
• CNCF Platinum Member
JD Container Platform

- **Business**
  - Online Services
  - Big Data
  - AI
  - IoT

- **Application**
  - Middleware Platform
  - Storage Service
  - Elastic Compute Service

- **Service & Platform**
  - JDOS (Jingdong Datacenter OS)
  - Containerized Infrastructure

- **OS**

- **Infrastructure**
  - Geographically Distributed Datacenters
  - Retail Stores
  - Thailand
  - Indonesia
  - 7FRESH
  - TOFLIFE
  - Business

- **Intelligent Operations Management**
Overview

1. Introduction
2. Function as a Service (FaaS)
3. Fibonacci FaaS Platform
4. Use Cases
5. Conclusions
Function as a Service
Function as a Service (FaaS)

- A form of serverless computing
- Function (small unit of work)
  - Development, deployment, maintenance, operation, monitoring, resource scaling
- Event-driven execution
FaaS at JD: Why?

Diverse use case demands: simplified development and automated deployment & management
Faas at JD: Why?

Containerized Platform and Ecosystem

Application
- Order System
- Inventory System

Middleware
- Elastic Database
- JIMDB
- JMQ
- ES

Compute
- Spark
- Flink
- Storm
- TensorFlow
- Caffe

JDOS Scheduling and Management

Fine-grained Elasticity for Resource Efficiency

Data Center
- IDC-N
  - Cluster 1
  - Container 1
- IDC-B
  - Cluster 2
  - Container 2
- IDC-C
  - Cluster N

Containerized Server
FaaS at JD: Why?

**Service: Productivity**
- Simplification
- Automation

**Infrastrucure: Efficiency**
- Fine-grained resource management
- Elastic scaling

Complexity

Utilization
Fibonacci: JD’s FaaS Platform
Fibonacci: From PaaS To FaaS

JD’s enterprise-grade FaaS platform

- **Usability:** Enhanced functionality and features
- **Simplicity:** Customized templates and ecosystem integration for multiple use cases
- **Efficiency:** Optimized elastic scheduling
Function Triggers

- Request dispatching
- Runtime data collection
- Access control
- Extensibility: DNS, log
Function Execution

- Request to input, output to response conversions
- Function call
- Process and thread invocations for different cases
Function Watcher

- **Innovation logic**
- **Health check**: monitoring data collection, heartbeat, status, etc.
- **Hot update**: container volumes, environment variables

![Diagram of Function Watcher]

- **Function invocation**
- **Data collection**
- **Hot replacement**
- **Heartbeat detection**
- **Status**

**Update methods**
- Container configuration & environment variables
- Directory, volume, extremal storage

*JD.COM*
Multiple methods for deployment and update

- Function templates
  - Efficient
- Directory, volume & extremal storage
  - Smaller images
  - Update without image rebuilding

Container configuration & environment variables

- No image rebuilding
- No re-deployment
- Rapid update
Monitoring

- Fine-granularity monitoring & logging
- Demand-based elastic scaling

eg: `sum by(function_name) (rate(function_invoke_total{code="200",rate="50"}[10s])) / 50 /0.85 > sum(function_service_count) by (function_name)`
Scheduling Optimization

- Multiple pod pools (min and max sizes)
- Round-robin scheduling
- Hot deployment for latency-sensitive functions
- Cold deployment for latency-tolerant functions

- Hot deployment: configuration update and function push
- Update label to link/remove a function to/from K8S service
- Ready status control
Function Template Customization

Minimum Runtime
- Static Dependency
- Core APIs

Dynamic Dependency
- Function
- Configuration

Watcher Service

Function Template
Customization
Startup

FROM openjdk:8-jdk-alpine
WORKDIR /home/app/lib/
COPY lib/.
COPY handler-1.0-SNAPSHOT.jar .
WORKDIR /home/app/
COPY functionCode .
ENV jimb_url=jim://XXXXXXXXXXXXXXXXXX
CMD ["java",
"-cp",
"/home/app/lib/*:/home/app/handler-1.0-SNAPSHOT.jar",
"fib.jd.com.java.env.server"]
Security

• Function Code Scanning
  – Java, Python

• Real time monitoring and alarming
  – FIM, port scanning

• Access Control
  – Token-based authentication & authorization

• Network Traffic Monitor
  – DDOS attack
  – Network traffic analysis
Fibonacci Platform Summary

Complete FaaS capabilities: development, deployment, operation, and maintenance

Enhanced features and functionalities
- Function lifecycle management
- Customized function templates
- Multi-language support with extensibility: Python, Java, Node.js, Golang
- Multi-dimension monitoring and visualization

Seamless integration with JD’s container platform
- JDOS – JD’s Kubernetes
- Triggers: http, JSF gateway, timer, JMQ/Kafka
- Storage: JIMDB, CFS, Vitess database

Performance optimization and elastic scheduling

Security
Use Cases
Use Case 1: JD Intelligent Speaker Developer Platform

Third-party application development, deployment and management
Use Case 2: JD Intelligent Image Processing

- Machine learning templates
- Storage integration
Use Case 3: JD Mobile Content Services

- Monolithic applications to FaaS based lightweight services
- Elastic scheduling
Conclusions

Fibonacci: JD’s enterprise-grade FaaS platform

• Complete and enhanced FaaS capabilities for more efficient development, deployment and management of applications and services.
• Optimized elastic resource management for improved IT resource efficiency.

Challenges

• Use cases: barriers to adopting FaaS
• Performance and scalability
• Building an ecosystem
• The support of Java
Open Innovation

Academia

Open Source

Industry

JD
Thank you!
谢谢！

Contacts:
Yuan Chen (yuan.chen@jd.com, Wechat: yuan_gt)
Xin Tong (tongxin5@jd.com, Wechat: shangshant)