Why is Cloud-Native Application Development Still So Hard?

Integrating the Amino OS Distributed Cloud-native Programming Platform with Kubernetes (github.com/Amino-OS)

Venugopal Reddy K (Lead Architect, Huawei)
Irene Zhang (Univ. of Washington, & Microsoft Research)
Overview

A brief history of the (microservice) universe

App devs ≠ Sys devs ≠ SREs

Amino OS from 30,000'

Demo, Remaining Challenges + Q&A

Amino.Run: Evaluation and some Data

Amino.Run: How it Works
1. A Brief History of the (Microservice) Universe
Once upon a time, applications were:

- single user
- single platform
- single node
Life was good for mere mortal app devs...

- Single-machine OS’s work well
- Local procs, virtual memory, files, locks...
- Pick one (or two?) good programming languages
- App devs could understand their platform
Then “Suddenly” Everything Changed…

- Cloud Computing
- “Mobile-first”
- Ubiquitous Connectivity (Wifi… 3G… 4G… 5G…)

Mobile App

OS

Cloud Computing

“Mobile-first”

Ubiquitous Connectivity (Wifi… 3G… 4G… 5G…)
So Now Today’s Applications are Very Different…

- Multi-user,
- Multi-platform,
- Multi-language,
- Multi-node,
- Always-on,
- Autoscaling,
- Distributed Systems Nightmares!
So Containers, Kubernetes and Microservices Saved the Day

Apps could be:

• Decomposed into independently deployable Containers
• Programatically orchestrated, driven by declarative configuration
• Developed in many different languages Java/Kotlin for Android, ObjC/Swift for IOS, Go/Java/Python/C/C++/... for Linux/Windows...
• Hooked together using service meshes Linkerd, Envoy, Istio...
• Configured, deployed, monitored and upgraded by expert devops/SREs (basically Ninjas).
Turns out, it’s still really, really difficult...

- distributed concurrency, synchronization,
- reliable RPC, fault tolerance,
- replication, leader election, sharding,
- code and data migration,
- observability, fault diagnosis
- As well as all the obvious
- remote invocation, load balancing, etc...

Developers still have to write the (really hard) stuff in the containers:
These sound like distributed systems problems!

PROFESSIONAL SYSTEMS PROGRAMMER REQUIRED.
DO NOT ATTEMPT AT HOME.
2.
App devs $\neq$
Sys devs $\neq$
SREs
App Devs
• Know their app domain very well.
  • Social Networking
  • Travel
  • Finance
  • ...
• Need to move really fast.
• Don’t give a hoot about distributed systems algorithms, exponential backoff, PAXOS/Raft,...

Sys Devs
○ Are really interested in understanding and solving hard distributed systems problems.
○ Are in very short supply.
○ Typically don’t understand your specific business needs.

SREs/DevOps Engineers
○ Understand what happens when your specific customers hit your specific app, e.g.
  ○ Capacity/scaling requirements
  ○ Optimal sharding schemes
  ○ What breaks and why.
  ○ What needs to be replicated, updated etc and how.
3. Amino OS from 30,000’
What is Amino OS?

Amino OS is an umbrella project, the goal of which is to create a distributed platform for coding and running distributed (cloud, edge and mobile) microservice-based applications. It has four main components:

- **Amino.Run**: A distributed microservice runtime (we’ll focus on this today).

- **Amino.Sync**: A reactive data synchronization service that provides configurable consistency guarantees.

- **Amino.Store**: A distributed transactional storage service.

- **Amino.Safe**: A distributed privacy and security manager.
Amino OS
Distributed Cloud-native Application Programming Platform

Users (often mobile)

Distributed Cloud-native Application

Amino.Run
(Process Manager)

Amino.Sync
(Memory Manager)

Amino.Store
(Storage System)

Amino.Safe
(Security System)

Distributed Cloud-native Application Programming Platform

OS
Central Cloud Server

OS
Central Cloud Server

OS
Edge Cloud Server

OS
Edge Cloud Server

OS
Mobile Device (Phone)

OS
Mobile Device (IoT)
Amino OS is based on several years of distributed systems research done by Irene and her team at the University of Washington Systems Lab in Seattle, WA. Amino OS is the result of 2 years of collaboration between Quinton, Venu and Irene’s teams.

### What is Amino OS?

- **AminoRun**: Sapphire
  - **Requirement**: Run-time Manager
  - **Availability**: Auto-restart on crash
  - **Responsiveness**: Automatic process migration
  - **Scalability**: Automatic process spin-up
  - **Consistency**: Distributed locks
  - **Fault-tolerance**: Periodic process checkpoint
  - **Reactivity**: Notifications

- **AminoSync**: Diamond
  - **Requirement**: Memory Manager
  - **Availability**: Auto-sync w/ storage
  - **Responsiveness**: In-memory caching
  - **Scalability**: In-memory caching
  - **Consistency**: Atomic memory operations
  - **Fault-tolerance**: Auto-sync w/ storage
  - **Reactivity**: Sync across address spaces

- **AminoStore**: Tapir
  - **Requirement**: Storage Manager
  - **Availability**: Replication
  - **Responsiveness**: Storage caching
  - **Scalability**: Partitioning
  - **Consistency**: Transactions
  - **Fault-tolerance**: Log to disk
  - **Reactivity**: Triggers
We’ll Focus on Amino.Run in this Talk

- Goals
- Architecture and How it Works
- Deployment Managers
- Experience and Evaluation
- Demo
- Q&A
Amino.Run Goals

1. Separate application logic from deployment code.

2. Make application code very simple and intuitive

3. Allow devs and SRE’s to easily make, combine and change automated application deployment choices across arbitrary servers and devices (cloud, edge, mobile, IoT etc)

4. Support arbitrary programming languages

5. Performance!

6. Optionally integrate with external infrastructure systems (like Kubernetes, Istio etc) in a very natural way.
Our Solution

A new system architecture that supports:

• pluggable and extensible deployment managers

• across arbitrary programming languages

• and operating systems
Amino.Run Architecture

- Distributed Application
- Deployment Management Layer
- Deployment Kernel
  - Android OS
  - OS
  - OS
Partitioned into *Microservices*, which:

- Run in a single address space with RPC.
- Execute anywhere and move transparently.
- Provide a unit of distribution for deployment managers.
- May be written in any programming language (using GraalVM)
- Can pass data structures transparently between programming languages (using GraalVM Polyglot)
A brief word about multi-language and GraalVM

- High-performance polyglot VM (think JVM)
- Native via Ahead-of-Time compilation, or JIT
- Embeddable
- Allows Microservices, Amino Kernel and DMs all in different languages
Amino.RunArchitecture

Distributed Application

Deployment Management Layer

Deployment Kernel

OS

OS

OS
Provides **best-effort distribution services**, including:

- Microservice instantiation, tracking, mobility and replication.
- Making and routing RPC to Microservice replicas.
- Managing, distributing and running deployment managers.
Amino.Run Architecture

Deployment Management Layer

Deployment Kernel

OS

OS

OS
Consists of deployment managers, which:

• Extend the functions and guarantees of the deployment kernel.
  • Sharding, Method Replication, Caching etc
• Interpose on Microservice calls and events.
• Easy to choose and change without modifying the application.
• Can be arbitrarily combined! (with some obvious restrictions)
  • E.g. Replicated shards, Transactional replicas, Retries over sharded transactions, etc…
Amino.Run Architecture
Sapphire Deployment Manager Library

Primitives
- Immutable
- AtLeastOnce RPC
- Keep In Place
- Keep On Device
- Keep In Cloud

Caching
- Explicit Caching
- Lease Caching
- Writethrough Caching
- Consistent Caching

Serializable RPC
- Locking Transactions
- Optimistic Transactions
- durable Transactions

Serializability
- Explicit
- Lease
- Writethrough
- Consistent

Checkpoint
- Explicit Checkpoint
- Periodic Checkpoint
- Durable Checkpoint

Replication
- RSM-Cluster
- RSM-Geo
- RSM-P2P

Locking
- Transactions
- Transactions
- Transactions

Transactions
- Optimistic
- Locking
- Locking

Optimistic
- Transactions

Transactions

Serializability

Checkpoints

Dynamic Migration
- Code-offload

Load-balanced
- Frontend

Fronndend
- Scale-up

Frontend

LB Master-Slave

Scalability

Load-balanced

Frontend

Scale-up

Frontend

LB Master-Slave

Scalability

Load-balanced

Frontend

Scale-up

Frontend

LB Master-Slave

Extensible with the Sapphire Deployment Manager API!
Outline

1. Architecture
2. Deployment Managers
3. Experience and Evaluation
Deployment Manager API

Deployment Manager ("DM") components, which the Amino.Run kernel creates, deploys and invokes automatically:

- **Server-Side DMs**: Co-located with the Microservice Replica (i.e. process).
- **Client-Side DMs**: Co-located with remote references to the Microservice.
- **Group Coordinator DMs**: Co-located with fault-tolerant Microservice Management Service (MMS aka OMS).
Deployment Manager Architecture
Replicating a Microservice
Offloading a Microservice

- Server DM
- Client
- Server DM
- Microservice
- Cloud
- Stub
- Client DM
- Microservice
- Server DM
- DK Server
- OS
- DK Server
- OS
- DK Server
- OS
• Automatic Object Migration – Stateful Offloading

• See more in the demo later.
Caching a Microservice state
Outline

1. Sapphire Architecture
2. Deployment Managers
3. Experience and Evaluation
# Experimental Setup

<table>
<thead>
<tr>
<th>Device</th>
<th>CPU Type</th>
<th>Frequency</th>
<th>RAM</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Server</td>
<td>8-core Intel Xeon</td>
<td>2GHz</td>
<td>8GB</td>
<td>Ubuntu Experimental</td>
</tr>
<tr>
<td>Nexus 7</td>
<td>4-core ARM Cortex A9</td>
<td>1.3GHz</td>
<td>1GB</td>
<td>Android</td>
</tr>
<tr>
<td>Nexus S</td>
<td>1-core ARM Cortex A8</td>
<td>1GHz</td>
<td>512MB</td>
<td>Android Caramel</td>
</tr>
</tbody>
</table>
Peer-to-Peer Multiplayer Game

- **Read**
- **Write**

**Axes**
- Y-axis: milliseconds
- X-axis: Play...

**Legend**
- Keep In Cloud
- Keep On Device
- RSM-P2P
Code-offloading for Physics Engine

- Base: 1190 ms
- WiFi: 0 ms
- 4G: 200 ms

Network

milliseconds

<table>
<thead>
<tr>
<th>Base</th>
<th>WiFi</th>
<th>4G</th>
<th>Base</th>
<th>WiFi</th>
<th>4G</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>100</td>
<td>200</td>
</tr>
</tbody>
</table>

Phone

Tablet

42
Summary

Modern applications implement difficult distributed deployment tasks.

**Amino.Run** is a new programming system for deploying interesting distributed applications including cloud-native, mobile/cloud, edge/cloud.

**Deployment managers** makes it easy to choose, combine, and customize deployment options.
• Migrating state that’s not inside the application or Amino system (e.g. local files, Linux timers etc).
• Some rough edges between certain language combinations.
• Additional plugins for external systems (Istio, etcd, TiKV, etc)
• Federations and disconnected Edge scenarios.
Get Involved

• Slack Channel: Amino-OS.slack.com
• Web site: www.Amino-OS.io
• Contributions most welcome
• Repo: github.com/Amino-OS/Amino.Run