Amino

A distributed runtime for edge applications

Futurewei CloudBU Lab

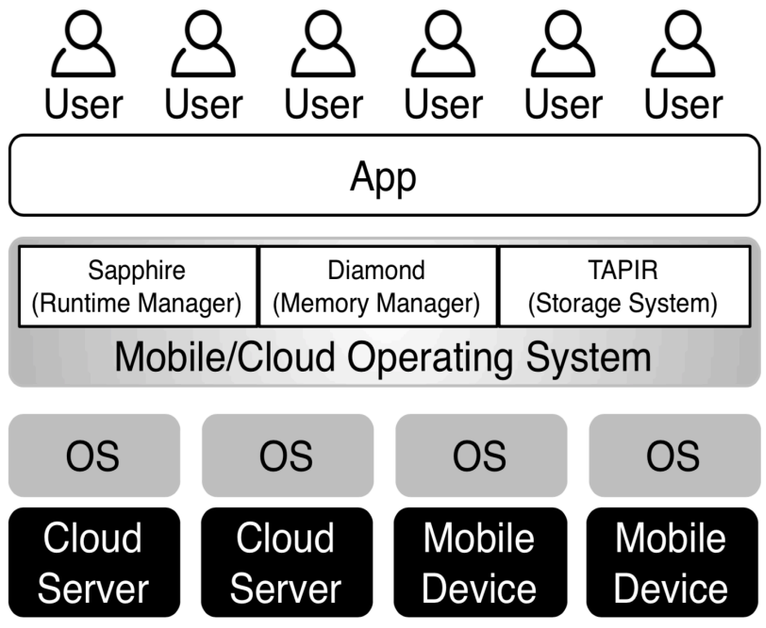
Origin of Amino

- Amino is built on top of several years' research result from UW system lab

- Amino is the result of collaboration between Huawei Seattle Lab and UW System Lab

What is Amino?

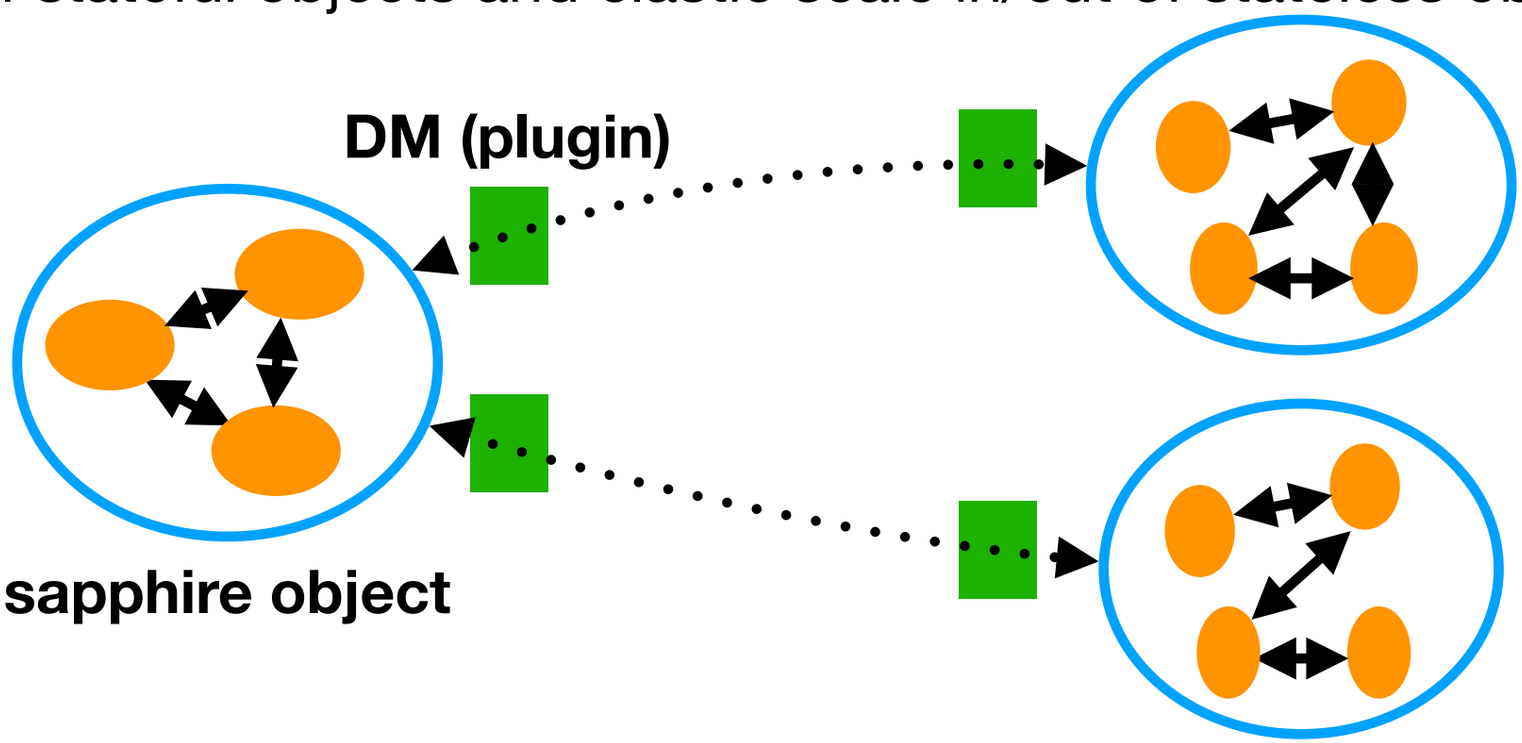
- Amino is an umbrella project whose goal is to create a distributed platform for coding and running edge applications. It has four big components:
  - *AminoRun*: A distributed runtime
  - *AminoSync*: A reactive data synchronization service that provides configurable consistency guarantees
  - *AminoStore*: A distributed transactional storage service
  - *AminoSafe*: A distributed security manager



	AminoRun	AminoSync	AminoStore
	Sapphire	Diamond	Tapir
Requirement	**Run-time Manager**	**Memory Manager**	**Storage Manager**
Availability	Auto-restart on crash	Auto-sync w/ storage	Replication
Responsiveness	Automatic process migration	In-memory caching	Storage caching
Scalability	Automatic process spin-up	In-memory caching	Partitioning
Consistency	Distributed locks	Atomic memory operations	Transactions
Fault-tolerance	Periodic process checkpoint	Auto-sync w/ storage	Log to disk
Reactivity	Notifications	Sync across address spaces	Triggers

What is Amino Runtime?

- In simple term:
  - it is Distributed Object Manager with plugins, aka DMs
  - Amino Runtime provides built-in DMs each of which handles one specific distribution task, e.g. caching, state persistence, sharding, code offloading etc.
  - Developers write single-threaded sapphire objects. They apply DMs on sapphire objects.
  - Amino Runtime manages sapphire objects, takes care of fault tolerance of stateful objects and elastic scale in/out of stateless objects



Motivation

- Moore's law is slowing down… People turn to software parallelism for performance improvement
- Ubiquitous Computing [Poslad 2009]
  - Highly Distributed - large scale distribution
  - Highly Interactive - real time interaction
  - Context Aware - stateful
  - Autonomy - self-healing, self-management
  - Intelligent
- *But, creating large scale, highly reliable, distributed application is difficult*

Stanford

John Hennessy

- John Hennessy (Jan 2007):
  - "When we start talking about parallelism and ease of use of truly parallel computers, we're talking about a problem that's as hard as any that computer science has faced."
  - "I would be panicked if I were in industry."

http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=445&page=3

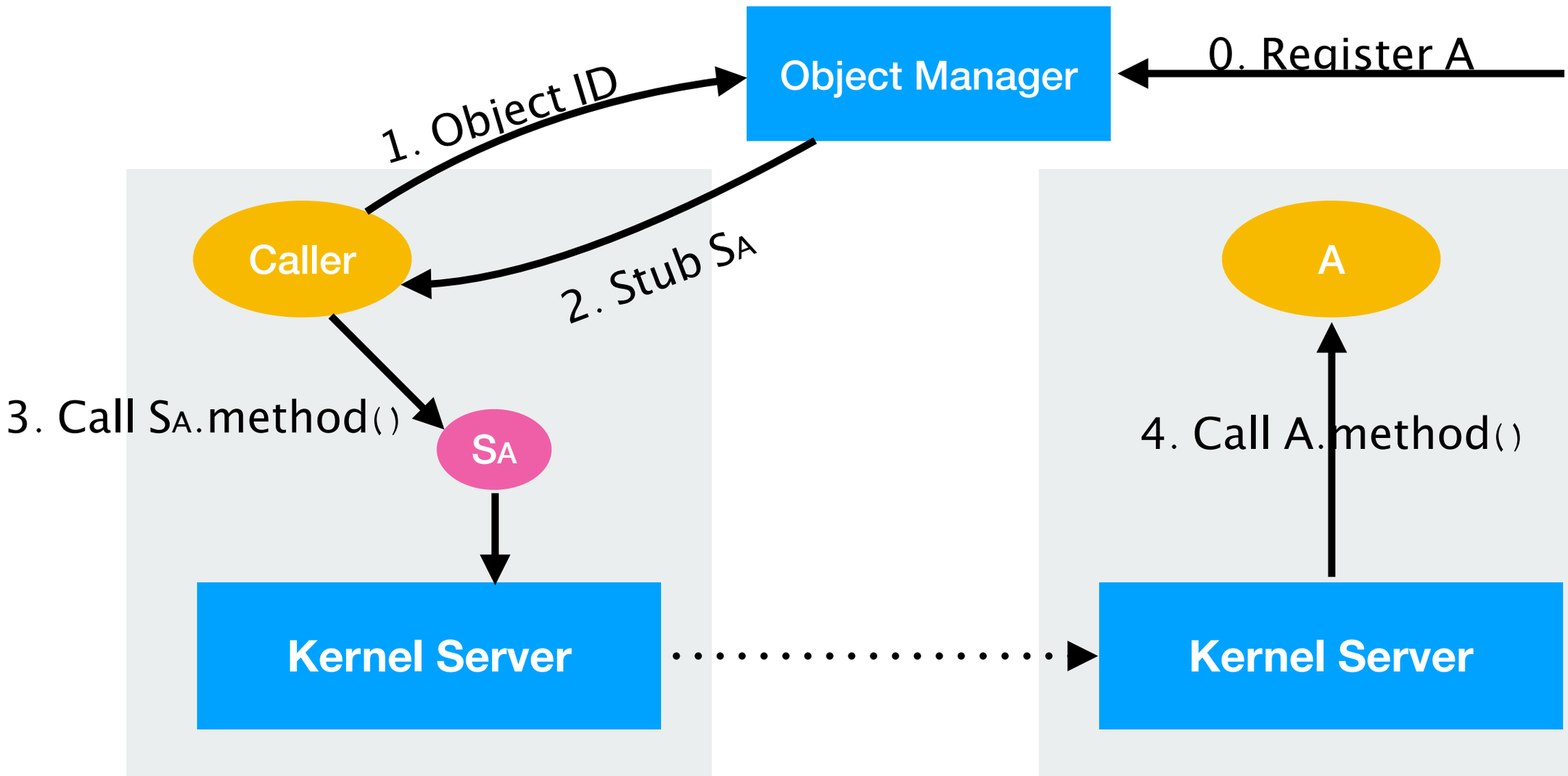
8.12.2009 Copyright Teemu Kerola 2009 16

Motivation

- Better Developer Productivity
  - Familiar OO programming paradigm
  - Developers only write single-threaded business logics
- Better Software Quality
  - DMs (plugins) are written by distributed system experts
  - DMs are thoroughly tested and maintained by as part of the infrastructure
- More Flexible
  - Developers can modify application behavior by changing DMs without modifying and compiling the application

Programming Model

- Sapphire objects need to be registered in *Object Manager* before being used
- To invoke a method on sapphire object A, the caller first fetches stub *SA* from Object Manager
- Caller invokes method on *SA - SA.method()*
- SA sends request to remote *Kernel Server* via local Kernel Server
- Remote Kernel Server invokes method on *Object A*



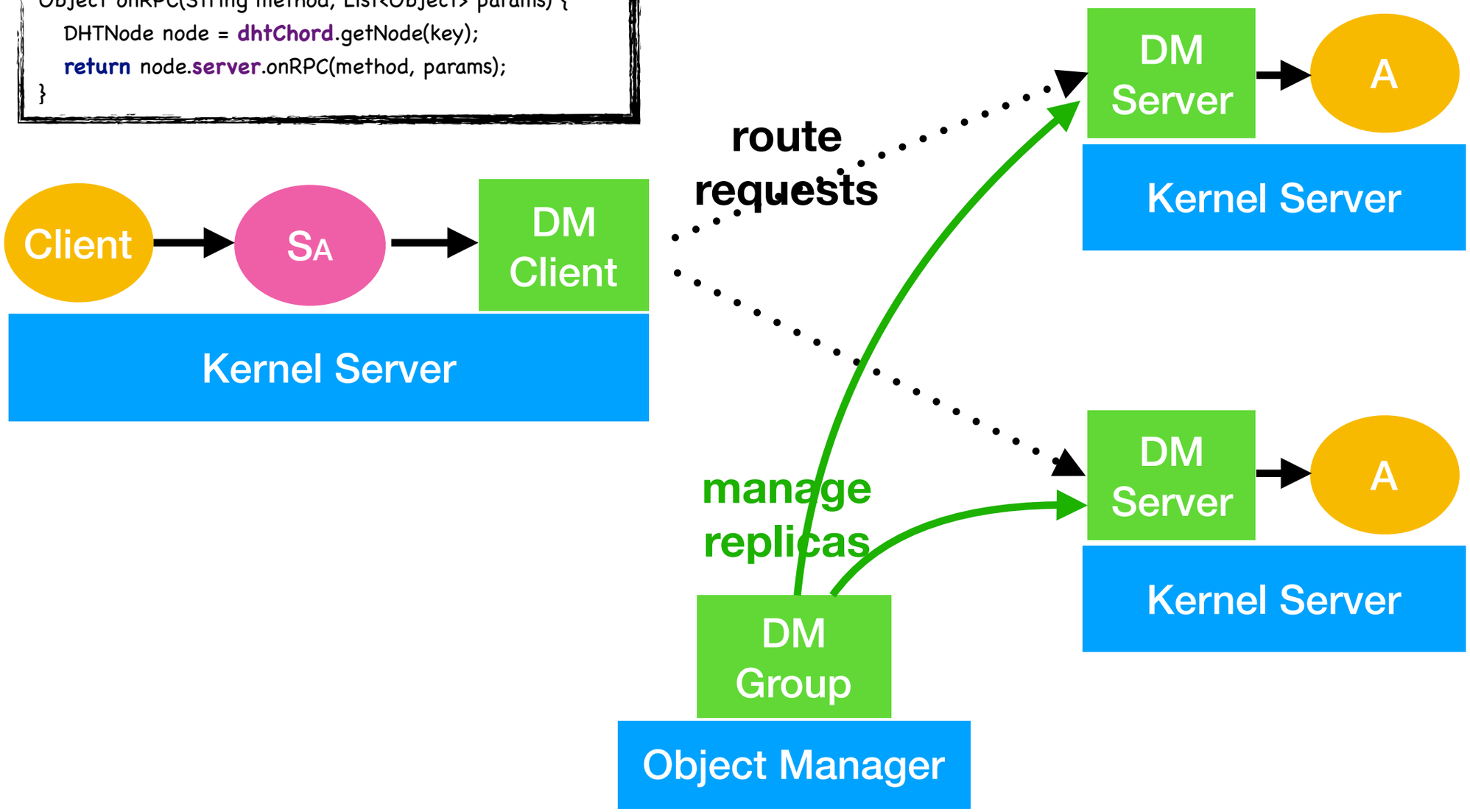
What is DM (Plugin)?

- DMs are plugins, aka *Deployment Managers*
- DMs are *developed* by distributed system experts
- Application developers selectively apply DMs on sapphire objects
- DMs are injected into sapphire objects during object creation
- Amino Runtime ships a collection of built-in DMs each of which handles one specific distribution task. For example:
  - Checkpoint DM - persist object state
  - Caching DM - simple client cache
  - DHT DM - distributed hash table
  - Master Slave DM - use master slave protocol to manage 2+ replicas
  - Raft DM - use raft protocol to guarantee consistency of 3+ replicas

What is DM (plugin)?

- Each DM has three components
  - *Client*: pre-process requests before sending them to servers
  - *Server*: pre-process requests before sending them to sapphire objects
  - *Group*: manages DM servers

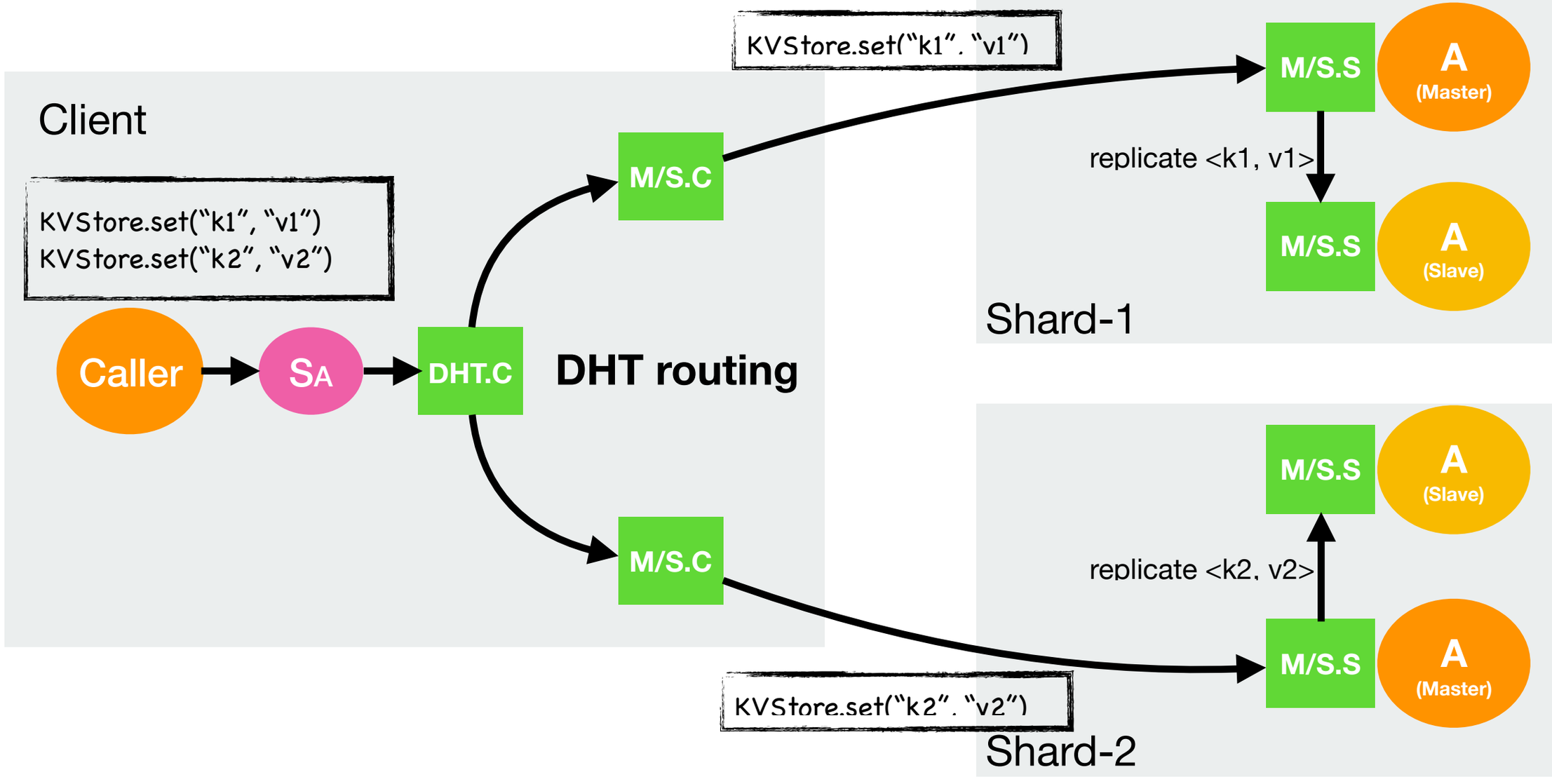
```
Object onRPC(String method, List<Object> params) {
    DHTNode node = dhtChord.getNode(key);
    return node.server.onRPC(method, params);
}
```



DM Chaining

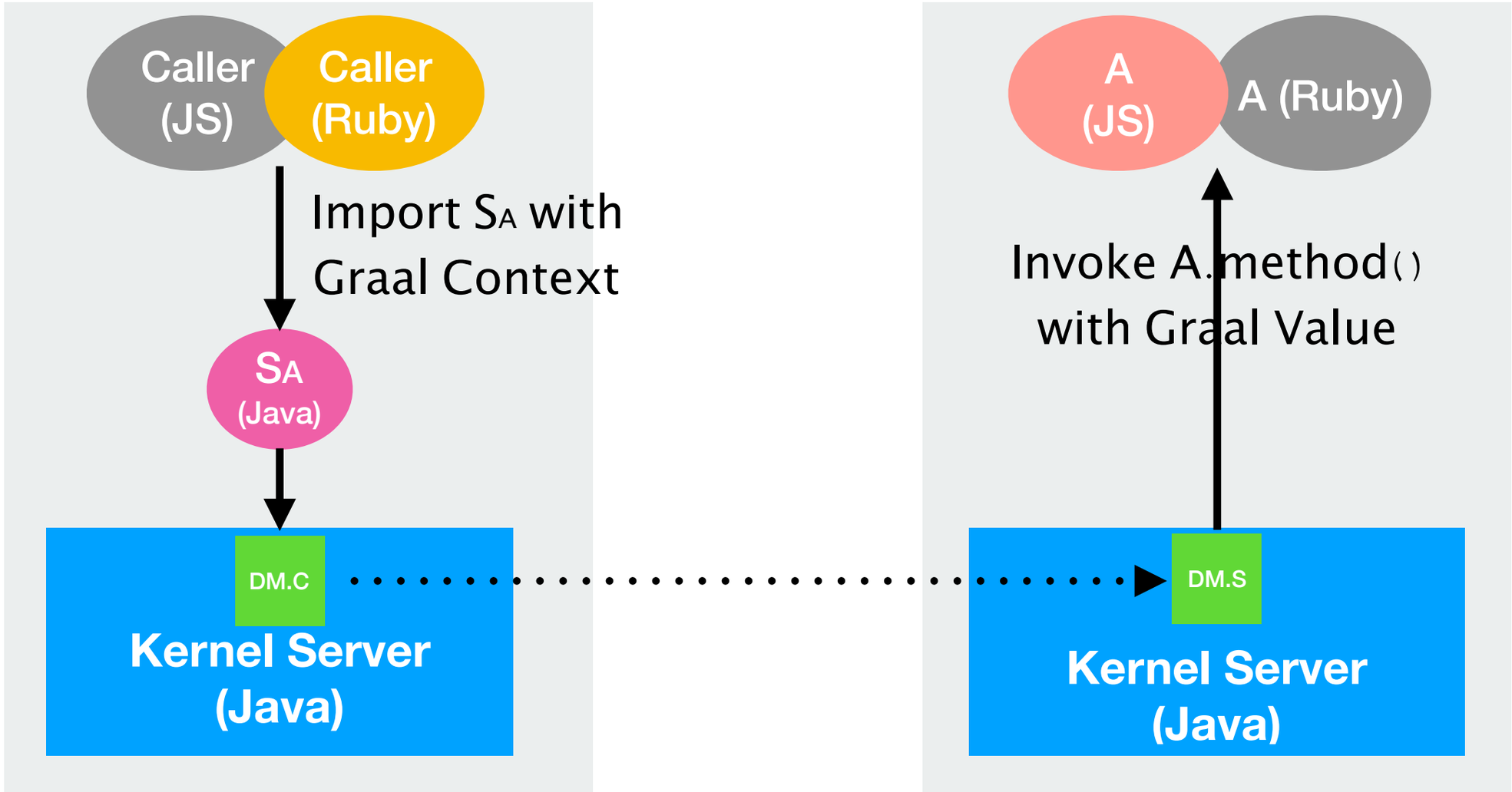
- What if I want to apply multiple DMs?
  - You chain them together…

```
!!sapphire.app.SapphireObjectSpec
dmList:
- configs:
  - !!sapphire.policy.dht.DHTPolicy$Config
{numOfShards: 2}
  name: sapphire.policy.dht.DHTPolicy
- configs:
  name: sapphire.policy.MasterSlavePolicy
```



Multi Language

- Kernel Server and DMs are written in Java
- Sapphire objects and applications can be written in other languages, e.g. JS, Ruby, Python, etc
- Relies on cross language interoperability capability provided by Oracle Graal
- Still at experimental phase… looks promising



Future Work

- Support intelligent code offloading - dynamically migrate objects between device, edge, and cloud
- Reduce footprint of Kernel Server - run kernel server in small devices
- Make Object Manager highly reliable and highly scalable
- Create cloud IDE to allow developers code and deploy sapphire objects directly in browser

Demo

- Using DHT + MasterSlave to achieve
  - Horizontal scale out
  - High availability