CSci 8980
Mobile Cloud Computing

Mobile Cloud Programming
Introduction

- Mobile applications are multi-platform, multi-user
Introduction

• Code and data spread across many systems
The Problem

• Deployment logic is complex
  – where data and computation should be located
  – what data should be replicated or cached
  – what data consistency level is needed
  – what kind of failures to deal with
  – ...

• Application logic often contains deployment logic

=> Applications are too complex and inflexible
The Problem

• Deployment decisions are classic distributed systems problems
  – coordinating data and computation
  – fault tolerance
  – dealing with diverse hardware
The Problem

• Applications are not “one-size-fit-all”
• Some need:
  – Reliable RPC
  – Caching
  – Code offloading
  – Replication
  – …
• This is left to the application programmer
Solution

• We need a distributed OS for mobile cloud applications!

• Goto pdf
Customizable and Extensible Deployment of Mobile/Cloud Applications

Irene Zhang
Adriana Szekeres    Dana Van Aken    Isaac Ackerman,
Steven D. Gribble  Arvind Krishnamurthy  Henry M. Levy

University of Washington
Sapphire

A new programming system for deploying mobile/cloud applications.
Our Goals

1. Separate application logic from deployment code.

2. Allow programmers to easily choose and change application deployment.
Our Solution

A new system architecture that supports pluggable and extensible deployment managers.
Outline

1. Sapphire Architecture
2. Deployment Managers
3. Experience and Evaluation
Sapphire Architecture

Sapphire Application

Deployment Management Layer

Deployment Kernel

Android OS

OS

OS
Applications

• A new programming abstraction: distributed objects
Partitioned into **Sapphire Objects**, which:

- Run in a single address space with RPC.
- Execute anywhere and move transparently.
- Provide a **unit of distribution** for deployment managers.
Sapphire Architecture

Deployment Management Layer

Deployment Kernel

OS

DK Server

Mobile App

App

DK Server

Deployment Kernel

Deployment Management Layer

Home 14 Visitor 19

Game Board

Atomic Structure

Deployment Kernel

Deployment Management Layer

Mobile App

App

DK Server

Deployment Kernel

Deployment Management Layer

Mobile App

App

DK Server

Deployment Kernel

Deployment Management Layer
public sapphireclass User uses ConsistentCaching {

    // user handle
    String username;
    // people who follow me
    User[] followers;
    // people who I follow
    User[] friends;

    ...

    public String getUsername() {
        return username;
    }

    public User[] getMyFollowers() {
        return followers;
    }

    public User[] getPeopleIFollow() {
        return friends;
    }

    public Tweet[] getMyTweets() {
        return myTweets.getTweets();
    }
}
Provides *best-effort distribution services*, including:

- Sapphire object tracking, mobility and replication.
- Making and routing RPC to Sapphire objects.
- Managing, distributing and running deployment managers.
Consists of deployment managers, which:

- Extend the functions and guarantees of the deployment kernel.
- Interpose on Sapphire object events.
- Easy to choose and change without modifying the application.
Sapphire Architecture

 Lease Caching  

 Replication  

 Code-offloading  

 DK Server  

 OTS Server  

 Android OS  

 OS  

 OS
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
- Consistent Caching

Checkpoint
- Explicit Checkpoint
- Periodic Checkpoint
- Durable Transactions

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

Mobility
- Explicit Migration
- Dynamic Migration
- Explicit Code-offload
- Code-offload

Scalability
- LoadBalanced Frontend
- Scale-up Frontend
- LB Master-slave

Extensible with the Sapphire Deployment Manager API!
Outline

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

Deployment manager components, which the Sapphire kernel creates, deploys and invokes:

- **Instance Manager**: Co-located with the Sapphire Object.
- **Proxy**: Co-located with remote references to the Sapphire Object.
- **Coordinator**: Co-located with fault-tolerant Object Tracking Service (OTS).
Deployment Manager Architecture
Deployment Manager Architecture
DM Code

• Proxy: caller-side tasks
  – E.g. use caller side cache

• Instance manager: callee-side tasks
  – E.g. checkpoints

• Coordinator: centralized tasks
  – E.g. load balance
### DK API

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke(RPC)</td>
<td>Invoke RPC on the local SO</td>
</tr>
<tr>
<td>invoke(SO,RPC)</td>
<td>Invoke RPC on a specific SO</td>
</tr>
<tr>
<td>getNode()</td>
<td>Get ID for local node</td>
</tr>
<tr>
<td>getNodes()</td>
<td>Get list of all nodes</td>
</tr>
<tr>
<td>pin(node)</td>
<td>Move SO to a node</td>
</tr>
<tr>
<td>setHighLatency(ms)</td>
<td>Set limit for RPC latency</td>
</tr>
<tr>
<td>durable_put(SO)</td>
<td>Save copy of the SO</td>
</tr>
<tr>
<td>durable_get(key)</td>
<td>Retrieve SO</td>
</tr>
<tr>
<td>replicate()</td>
<td>Create a replica</td>
</tr>
<tr>
<td>destroyReplica(IM)</td>
<td>Eliminate a replica</td>
</tr>
<tr>
<td>getReplicas()</td>
<td>Get list of replicas for SO</td>
</tr>
<tr>
<td>getReplica()</td>
<td>Get ref to SO instance</td>
</tr>
<tr>
<td>setReplica(SO)</td>
<td>Set ref to SO instance</td>
</tr>
<tr>
<td>copy(SO)</td>
<td>Create a copy of the SO instance</td>
</tr>
<tr>
<td>diff(SO,SO)</td>
<td>Diff two SO instances</td>
</tr>
<tr>
<td>sync(SO)</td>
<td>Synchronize two SO instances</td>
</tr>
<tr>
<td>getIM()</td>
<td>Get ref to DM Instance Mgr</td>
</tr>
<tr>
<td>setIM(IM)</td>
<td>Set reference to DM Instance Mgr</td>
</tr>
<tr>
<td>getCoordinator()</td>
<td>Get ref to DM Coordinator</td>
</tr>
<tr>
<td>getReference(IM)</td>
<td>Create DM Proxy for IM</td>
</tr>
<tr>
<td>registerMethod(m)</td>
<td>Register a custom method for DM</td>
</tr>
<tr>
<td>getRegion()</td>
<td>Get ID for local region</td>
</tr>
<tr>
<td>getNode()</td>
<td>Get ID for local node</td>
</tr>
<tr>
<td>pin(region)</td>
<td>Move SO to region</td>
</tr>
<tr>
<td>pin(node)</td>
<td>Move SO to node</td>
</tr>
<tr>
<td>getRegions()</td>
<td>Get list of server regions</td>
</tr>
<tr>
<td>getNodes()</td>
<td>Get list of nodes in local region</td>
</tr>
</tbody>
</table>
public class LeasedCaching extends DManager {
    public class LCProxy extends Proxy {
        Lease lease;
        SapphireObject so;

        public Object onRPC(SapphireRPC rpc) {
            if (!lease.isValid() || lease.isExpired()) {
                lease = Sapphire.getReplica().get Lease();
                if (!lease.isValid()) {
                    throw new SNotFoundAvailableException(
                        "Could not get lease."");
                } else {
                    so = lease.getSO();
                }
            }
        }

        SapphireObject oldSO = Sapphire.copy(so);
        Sapphire.invoke(so, rpc);
        SStream diff = Sapphire.diff(oldSO, so);
        if (diff) Sapphire.getReplica().update(diff);
    }

    public class LCRReplica extends InstanceManager {
        public synchronized Lease get Lease();
        public synchronized void update(SStream);  
        // Code for Instance Manager methods
    }
}
# Upcalls

DK calls in DM using upcalls => customization

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onCreate</td>
<td>Creation of SO instance</td>
</tr>
<tr>
<td>onRPC</td>
<td>Method invocation on SO</td>
</tr>
<tr>
<td>onFailure</td>
<td>Replica failed</td>
</tr>
<tr>
<td>onDestroy</td>
<td>Coordinator eliminated SO</td>
</tr>
<tr>
<td>onHighLatency</td>
<td>Avg. RPC latency &gt; limit</td>
</tr>
<tr>
<td>onLowMemory</td>
<td>Node running out of memory</td>
</tr>
<tr>
<td>onMemberChange</td>
<td>New replica added to group</td>
</tr>
<tr>
<td>onRefRequest</td>
<td>Request for an SO reference</td>
</tr>
</tbody>
</table>
Replicating a Sapphire Object
Replicating a Sapphire Object
Replicating a Sapphire Object
Replicating a Sapphire Object
Replicating a Sapphire Object
Replicating a Sapphire Object

DK Server

DK Server

DK Server

Instance Manager

Instance Manager

Instance Manager

Sapphire Object

Sapphire Object

Sapphire Object

Proxy

Stub

Leader
Replicating a Sapphire Object
Caching a Sapphire Object
Caching a Sapphire Object
Caching a Sapphire Object
Caching a Sapphire Object
Outline

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

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

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

<table>
<thead>
<tr>
<th>miliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>Keep In Cloud</td>
</tr>
</tbody>
</table>
Modern applications implement difficult distributed deployment tasks.

Sapphire is a new programming system for deploying mobile/cloud applications.

Deployment managers makes it easy to choose, change and build deployment.