**Wiera**: Towards Flexible Multi-Tiered Geo-Distributed Cloud Storage Instances

**Kwang sung Oh**, Abhishek Chandra, and Jon Weissman

Department of Computer Science and Engineering
University of Minnesota Twin Cities

**HPDC 2016**
Cloud Providers Publicly Available

- HP Public Cloud
- Google
- IBM
- Amazon Web Services
- Netflix
- Uber
- Airbnb
- Windows Azure

Powered by Intel Cloud Technology
Multiple DCs on Earth

From http://www.datacentermap.com
Users are Around the World
Geo-Distributed Users, DCs and Applications

Where are the best locations for storing data?
Factors that Influence Data Placement

- SLA
- Consistency Model
- Desired Cost
- Desired Fault Tolerance
- Data Access Pattern
- Users’ Locations
- And many more…
Challenges

• Different applications will emphasize different factors to yield **diverse policies**
  - How easy is to create data placement policies?

• **Complexities** from multiple storage tiers across cloud providers
  - How can nearby DC’s storage be accessed easily?

• **Dynamics** from cloud environment
  - How can applications achieve their desired goals even with dynamics?
Multiple DCs Offer Benefits

- Performance
- Cost
- Fault tolerance
Challenges

• Different applications will emphasize different factors to yield diverse policies
  - How easy is to create data placement policies?

• Complexities from multiple storage tiers across cloud providers
  - How can multi-DC storage be accessed easily?

• Dynamics from cloud environment
  - How can applications achieve their desired goals even with dynamics?
Dynamics Happen Because

- Cloud Service providers do not guarantee consistent performance
- Users are moving
- Users’ access patterns keep changing
Challenges

• Different applications will emphasize different factors to yield diverse policies
  - How easy is it to create data placement policies?

• Complexities from multiple storage tiers across cloud providers
  - How can nearby DC’s storage be accessed easily?

• **Dynamics** from cloud environment
  - How can applications achieve their desired goals in the presence of dynamics?
Our Goal

• Enabling applications to achieve their desired goals in **geo-distributed cloud environment** by

1. Supporting flexible storage policies
2. Exploiting multiple storage tiers across DCs
3. Handling dynamics at runtime
Roadmap

• Motivations & Goals

• **Wiera Storage System**

• Policy Examples

• Experimental Evaluations
Wiera Storage System

- Policy driven key-value storage system for geo-distributed environment
- Built upon Tiera which hides complexities of multiple storage tiers
- Provides easy way to specify and change policies
Background: Tiera [RaghavanCW14]

✓ **Handles data placement** among storage tiers within a single DC

✓ Provides **Get/Put** APIs to applications

✓ Uses **Event-Response** mechanism
  - Action (Put/Get), Timer, Threshold
  - Store, Copy, Queue, Compress, Encrypt
Multiple Storage Tiers in a DC

Diagram showing various storage tiers including S3, ElastiCache, and EBS.
Tiera Storage Instance
Tiera Policy Example: Write Back Policy

Asynchronously
Tiera Policy Example: Write Through Policy

Synchronously
Wiera Architecture

Global Policy Manager

Policy

Meta Data

Tiera Instances Manager

Tiera Server Manager

Wiera User Interface

Data

Control

Tiera Server

Tiera Instance 1

Tiera Instance 2

Tiera Instance 3

Tiera Instance N

Region 1

Region 2

Region N
How Wiera Works
Roadmap

• Motivations & Goals
• Wiera storage system

• Policy examples
• Experimental evaluations
Flexible Storage Policies

- Consistency Policies
  - Access Nearby DC’s memory Policy
  - Changing Consistency at run-time
StrongConsistency Policy

```plaintext
event(put.into) : response {
    lock(what:insert.key)
    store(what:insert.object, to:local_instance)
    copy(what:insert.object, to:all_regions)
    release(what:insert.key)
}
```

```plaintext

gWriteLock = getGlobalWriteLock(key);
gWriteLock.acquire(); //Getting the lock

Version = m_tier1Instance.put(key, value, strTierName, tag); //Write to local instance
broadcastToPeers(key, Version, value, strTierName, tag, 0, latencyInfo); //Broadcast

gWriteLock.release(); //Release the lock
```
EventualConsistency Policy

```
event(put.into) : response {
    store(what:insert.object, to:local_instance)
    queue(what:insert.object, to:all_regions)
}
```

Broadcasting in Background
Flexible Storage Policies

- Consistency Policies
- Access Nearby DC’s memory Policy
- Changing Consistency Policy at runtime
AccessNearDCMemory Policy

event(put.into) : response {
    store(what:insert.object, to:local_instance)
    copy(what:insert.object, to:NearDC.Memory)
}

event(get.from) : response {
    retrieve(what:get.object, from:NearDC.Memory)
}
Flexible Storage Policies

- Consistency Policies
- Nearby DC’s memory for performance
- Changing Consistency Policy at runtime
Strong Consistency is Expensive

Network Delay

Storage Delay
Eventual Consistency for SLA
event(threshold.type == put) : response {
    if (threshold.latency > 800 ms && threshold.period > 30 seconds)
        change_policy(what:consistency, to:EventualConsistency);
    else if (threshold.latency < 800 ms && threshold.period > 30 seconds)
        change_policy(what:consistency, to:StrongConsistency);
}

ElastiCache
EBS
S3

Wiera

SLA: 800 ms
Period: 30 sec
Roadmap

• Motivations & Goals
• Wiera storage system
• Policy examples

• Experimental evaluation
Evaluation

- Amazon and Azure Cloud
  - US East, US West, Europe West, and Asia Southeast
- YCSB, SysBench, and RUBiS
- Evaluation illustrates
  - Wiera helps applications *achieve desired goal*
  - *Unmodified* applications can get benefits from Wiera
  - Wiera *handles dynamics* by changing the consistency model
Local Disk vs. Nearby DC’s Memory

SysBench Configure 1:
Local Disk (Not Using Wiera)
Local Disk vs. Nearby DC’s Memory

SysBench Configure 2: Nearby DC’s Memory (Using Wiera)
RUBiS - Unmodified Application on Wiera

RUBiS Configure 1:
MySQL uses Local Disk
(Not Using Wiera)

RUBiS Configure 2:
MySQL uses Nearby DC’s Memory (Using Wiera)
RUBiS Throughput
Local-disk vs. Nearby DC’s memory

- Local Disk (Not using Wiera)
- Remote Memory (Using Wiera)

<table>
<thead>
<tr>
<th></th>
<th>Basic A2</th>
<th>Standard D1</th>
<th>Standard D2</th>
<th>Standard D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request/s</td>
<td>16</td>
<td>18</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>16</td>
<td>24</td>
<td>27</td>
</tr>
</tbody>
</table>

Distributed Computing Systems Group
event(threshold.type == put) : response {
if(threshold.latency > 800 ms
&& threshold.period > 30 seconds)
change_policy(what:consistency,
to:EventualConsistency);
else if(threshold.latency < 800 ms
&& threshold.period > 30 seconds)
change_policy(what:consistency,
to:StrongConsistency);
}
User-Perceived Latency

SLA Threshold: 800 ms

SLA violation more than 30 seconds

No SLA violation in 30 seconds
Conclusion

• Applications can achieve their desired goals through Wiera
  ✓ Providing an easy way to specify policies
  ✓ Exploiting multiple storage tiers across DCs
  ✓ Handling dynamics at runtime
Thank You!