Dynamic Program Code Distribution in Infrastructure-as-a-Service Clouds

Rostyslav Zabolotnyi, Philipp Leitner, Schahram Dustdar

Distributed Systems Group
Vienna University of Technology
Outline

- Introduction
  - Cloud Computing
  - CloudScale

- Code distribution
  - Motivation Scenario
  - Code Distribution Strategy
  - Evaluation

- Conclusion
Cloud Computing Tiers

**SaaS**
- Office 365
- Google Workspace

**PaaS**
- Heroku
- Windows Azure

**IaaS**
- Google Compute Engine
- openstack
- EC2
Introducing CloudScale
CloudScale in Cloud Computing

- **SaaS**
  - Office 365
  - Windows Azure

- **PaaS**
  - Heroku

- **IaaS**
  - Google Compute Engine
  - OpenStack
  - EC2

SIMPLICITY

CONTROL

SaaS

PaaS

Cloud Scale

IaaS
How to Use CloudScale

1. Write multi-thread applications like regular Java software

2. Define scaling behavior via Java annotations

   @CloudObject
   public class MyWorker {

       public void doWork() { ... }

       @DestructCloudObject
       public int getResults() { ... }
   }

3. Distribution logic is weaved into with AspectJ
How CloudScale Works

MyWorker worker = new MyWorker();
worker.doWork();

1. Selecting cloud host for an object
2. Instantiating object on the host
3. Creating proxy object on the client
4. Forwarding invocations to cloud host/object
what?
Motivating Scenario

In case of:
• Parallel development
• Concurrent testing
• Multiple version support

We need:
• Seamless code versioning
• Transparent code distribution
• Version independency

Alternatives:
• All versions in VM image
• Special code server
• Code distribution framework
Mixing in Code Distribution
Code Distribution Strategy

1. Missing code load attempt
   - Code is missing in cache
     - Query code from code storage
     - Verification failed, correct code provided
     - Register new code in cache
     - Resume execution
   - Code is in cache
     - Check cache for code
     - Verify code with code storage
     - Verification succeeded
Evaluation Setup

- Only first class access per run causes overhead
- Sentiment analysis application (≈50MB footprint)
- Comparison against baseline (when no code distribution is used)

Different code loading strategies are evaluated
- Complete
- Class-based
- Smart batching
Code Distribution Evaluation

Class loading strategies:

- **Complete**: we load everything on startup
- **Class-based**: we load each class on request
- **Smart batching**: files, packages, hierarchies…

![Bar chart](image)
Conclusions

- Popularization of cloud computing
- Code distribution challenge
  - Seamless and effective code distribution
  - Code versioning and multi-tenancy
- Future work:
  - Code prefetching
  - Historical information
Service Engineering in the Cloud: Code Distribution

- Dynamic code distribution
- Code versioning and validation
- Code caching inside the cloud
Thanks for attention!

P23313-N23 (Audit 4 SOAs)

Vienna PhD School of Informatics