Interoperable Data Management Approaches for Cloud and Mobile Computing

Attila Kertesz, Tamas Pflanzner and Roland Tornyai

University of Szeged
Department of Software Engineering
http://www.inf.u-szeged.hu/~keratt
keratt@inf.u-szeged.hu

Joint CLEEN and ACROSS Workshop on Cloud Technology and Energy Efficiency in Mobile Communications at EUCNC’15, Paris, France - 29 June, 2015
Outline

- Cloud Federation research issues
- Enhancing data processing capabilities of mobile devices
- Application porting to Infrastructure and Personal Clouds
- Autonomous data management over Personal Clouds
Motivation

- Advent of Cloud Federations
- Data retrieval and sharing problems in IaaS Clouds
- Rapidly growing number of mobile devices
- Exploiting Cloud solutions for scientific workflow execution
- Data privacy issues
- Big data
Federated Cloud Management (FCM)

- An autonomic resource management solution developed by SZTAKI
- Provides an entry point to a cloud federation
- Provides transparent service execution for users
- Following challenges are considered:
  - Varying load of user requests
  - Enabling virtualized management of applications
  - Establishing interoperability and provider selection
  - Minimizing Cloud usage costs
FCM Architecture: overview

- Top-level brokering
- Autonomously manage the interconnected cloud infrastructures
- Forms a federation with the help of Cloud-Brokers
Personal Clouds

- Storing data online
- Synchronization
- Data sharing
- Backup
- Version control
- Encryption
Dropbox

- Popular (>50 million users)
- 2GB free (+500MB/invitation, max. 16GB)
- Additional bonus (+3GB with automatic media upload, +26GB Space Race, etc.)
- Wide OS support (Win, Mac, Linux, iOS, Android)
- Versioning, encryption
- Wide API, SDK support (Java, Android, iOS, Python, Ruby)
Managing mobile data in Clouds

- A *real-world* use-case for interoperable data management among cloud infrastructures
- Our approach is to utilize cloud infrastructure services to execute *compute-intensive* applications on mobile data stored in cloud storages.
- Services for data management are running *in one or more* IaaS systems that keep tracking the cloud storage of a user, and execute data manipulation processes when new files appear in the storage.
Architecture for enhancing data processing capabilities of mobile devices

1. Upload mobile data, instructions
2. Download data by compute VMs
3. Upload results
4. Get computed data to device
The mobile application

- We have created a concrete application called *FolderImage*, which can be used to manipulate pictures produced by mobile devices.
- This program creates thumbnails of each image of the appropriate *folder* then ensembles them into a single image that represents the folder and gives an overview of its contents to the user.
- This app can be really useful by providing a *glimpse* of a directory, when a user has thousands of pictures spread over numerous directories, and she is looking for a specific one.
App interface
Performed steps

The folder image generation steps:

■ 1. list: to generate a list of the images the actual folder contains;
■ 2. download: to access the images of the folder;
■ 3. resize: to generate thumbnails of the images;
■ 4. create: to ensemble the thumbnails;
■ 5. upload: to save the created folder image.
Image generation in the Cloud

- We have also created a Java application encapsulated to a VA, deployed, and started it as a web service running in the SZTAKI cloud.
- It has a direct connection with the user's Dropbox storage, and it can continuously synchronize the image directory.
- Once the Android application triggers the folder image generation, the VA performs it.
- If we deployed web services of similar VAs into different IaaS providers, we could handle and manage data in an interoperable way among different IaaS solutions.
## Evaluation - Devices

<table>
<thead>
<tr>
<th></th>
<th>Samsung Galaxy Mini (phone)</th>
<th>Asus Slider SL101 (tablet)</th>
<th>Cloud VM1</th>
<th>Cloud VM2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>Android 2.2</td>
<td>Android 4.0</td>
<td>Ubuntu 12.04 64bit</td>
<td>Ubuntu 12.04 64bit</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>600 MHz</td>
<td>1 GHz (dual-core)</td>
<td>1 CPU</td>
<td>4 CPUs</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>384 MBs</td>
<td>1 GB</td>
<td>1 GB</td>
<td>4 GBs</td>
</tr>
</tbody>
</table>
Evaluation - results

### 450 images:

<table>
<thead>
<tr>
<th>Device</th>
<th>1. list (ms)</th>
<th>2. download (ms)</th>
<th>3. resize (ms)</th>
<th>4. create (ms)</th>
<th>5. upload (ms)</th>
<th>Sum (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android phone</td>
<td>879</td>
<td>199738</td>
<td>872</td>
<td>10631</td>
<td>1587</td>
<td>213707</td>
</tr>
<tr>
<td>Android tablet</td>
<td>304</td>
<td>68334</td>
<td>286</td>
<td>3480</td>
<td>491</td>
<td>72895</td>
</tr>
<tr>
<td>Cloud VM1</td>
<td>20</td>
<td>173</td>
<td>135</td>
<td>277</td>
<td>326</td>
<td>931</td>
</tr>
<tr>
<td>Cloud VM2</td>
<td>14</td>
<td>189</td>
<td>68</td>
<td>203</td>
<td>181</td>
<td>655</td>
</tr>
</tbody>
</table>

### 900 images:

<table>
<thead>
<tr>
<th>Device</th>
<th>1. list (ms)</th>
<th>2. download (ms)</th>
<th>3. resize (ms)</th>
<th>4. create (ms)</th>
<th>5. upload (ms)</th>
<th>Sum (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android phone</td>
<td>2972</td>
<td>401312</td>
<td>1496</td>
<td>21643</td>
<td>3173</td>
<td>430596</td>
</tr>
<tr>
<td>Android tablet</td>
<td>971</td>
<td>133575</td>
<td>509</td>
<td>6957</td>
<td>998</td>
<td>143010</td>
</tr>
<tr>
<td>Cloud VM1</td>
<td>44</td>
<td>220</td>
<td>300</td>
<td>575</td>
<td>702</td>
<td>1841</td>
</tr>
<tr>
<td>Cloud VM2</td>
<td>24</td>
<td>191</td>
<td>73</td>
<td>541</td>
<td>239</td>
<td>1068</td>
</tr>
</tbody>
</table>
Evaluation - results

- Regarding the Android devices, the tablet performed the generation 3 times faster than the phone in both rounds of experiments.
- The web service running in VM2 type virtual machine performed two times faster than the other deployment at VM1.
- The local execution on the Android devices are significantly slower (more than 100 times) than the image generations performed in the cloud.
- These measurements shown that both computation time and energy can be saved by moving computation-intensive tasks to clouds from mobile devices.
Generalized, interoperable approach for scientific applications
Use case - TINKER Conformer Generator

Our use case is a biochemical application that

- generates conformers of flexible molecules by unconstrained molecular dynamics at high temperature to overcome conformational bias,
- then finishes each conformer by simulated annealing and energy minimization to obtain reliable structures.
Use Case details

- Start
- Read input
- Peptide definition: Sequence, Chirality
- Generate 1000 K trajectory snapshots
- Parameter File 1

**T1**
- Minimize
- Parameter File 2

**Ti**
- Minimize

**Tn**

**300 K dynamics**
- Parameter File 3
- Parameter File 2

**TD1**
- Minimize

**TDM1**
- Minimize

**Simulated annealing**
- Parameter File 4
- Parameter File 2

**TSA1**
- Minimize

**TSAM1**

n = 50 000
Execution steps

- It uses five different conformer finishing methods:
  - minimizing the initial conformational states generated at high temperature (TM),
  - performing a short low temperature (e.g. 300 K) dynamics with the high temperature conformations to simulate a low temperature thermodynamical ensemble (TD),
  - minimizing the above low temperature states (TDM),
  - cooling the high temperature states by simulated annealing, e.g. to 50 K, or completely to 0 K (TSA),
  - minimizing the annealed states (TSAM).

<table>
<thead>
<tr>
<th>Step</th>
<th>Execution time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>13</td>
</tr>
<tr>
<td>TM</td>
<td>28</td>
</tr>
<tr>
<td>TD</td>
<td>3</td>
</tr>
<tr>
<td>TDM</td>
<td>28</td>
</tr>
<tr>
<td>TSA</td>
<td>26</td>
</tr>
<tr>
<td>TSAM</td>
<td>28</td>
</tr>
</tbody>
</table>
Execution steps

- Steps 1 to 5 denote communication between an Infrastructure and a Personal Clouds:
  - 1, the *configuration file is downloaded* to a VM, and the first yet not reserved task is selected.
  - 2, the *modified configuration file is uploaded back* to the storage containing the new reservation.
  - 3, the *VM downloads the data* to be processed in the selected task.
  - 4, once the data processing is finished, the *results are uploaded* to the storage.
  - 5, the *configuration file is refreshed* denoting the successful execution of the selected task.
Requirements

- In our approach users *only need to make available* their data in a Personal Cloud, and to specify with a configuration file the order of data processing (by linking VM methods to data).

- *Once this configuration file is available* and at least *one VM* (executing the necessary service for processing user data) *is running* in an IaaS Cloud, the autonomous data processing *starts and goes on* till all data is processed.
Porting challenge

The greatest challenge in porting TCG to this special multi-cloud environment was to manage the configuration file properly for the maximum 50 competing parallel workers.

- A synchronization problem occurs when more than one VM try to modify properties of our configuration file (typically for reserving a task).
- Solution: Ubuntu One – ListDelta, then Amazon SimpleDB
Evaluation in a private Cloud

- We have performed our evaluations by using a private IaaS Cloud called *SZTAKI Cloud* based on OpenNebula.

- The ported TCG application has been deployed in VMs started at SZTAKI Cloud by a desktop application using the Amazon AWS API. With this tool *we can deploy a certain number* of VMs in the SZTAKI Cloud that start the TCG application in a web service.

- First these TCG instances (capable of behaving as masters, workers or uploaders) *connect* to Ubuntu One and to the Amazon SimpleDB service, and *query* the configuration parameters stored there in a loop till there is any task to perform.
Evaluation results with Ubuntu One and OpenNebula

![Graph showing evaluation results with Ubuntu One and OpenNebula. The graph indicates the time spent in different phases for 3 VMs, 6 VMs, and 9 VMs.](image-url)
Evaluation with heterogeneous Clouds

- To increase heterogeneity, we considered a scenario when *academic and commercial* IaaS Clouds are interoperated through a Personal Cloud.

- We created *another evaluation* by using Dropbox, OpenNebula and Amazon. We used the same template configuration for OpenNebula to start 3 VMs in SZTAKI Cloud, and for Amazon we also started 3 VMs with Linux Micro instances.
Evaluation results with Dropbox, OpenNebula and Amazon
Detailed evaluation results
- The enormous data users produce with mobile devices are continuously posted to online services – they require the use of several Cloud providers at the same time to efficiently handle these data.
- Aim: to unite and manage separate Personal Clouds in an autonomous way.
Considered providers

<table>
<thead>
<tr>
<th>Provider</th>
<th>Initial Storage (GB)</th>
<th>Bonus Storage (GB)</th>
<th>Max. Storage (GB)</th>
<th>Supported OS</th>
<th>Mobile Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Drive</td>
<td>15</td>
<td>-</td>
<td>15</td>
<td>Win, Mac</td>
<td>iOS, Android</td>
</tr>
<tr>
<td>Dropbox</td>
<td>2</td>
<td>0.5</td>
<td>8</td>
<td>Win, Mac, Linux</td>
<td>iOS, Android</td>
</tr>
<tr>
<td>SugarSync</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>Win, Mac</td>
<td>iOS, Android</td>
</tr>
<tr>
<td>Box.com</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>Win, Mac</td>
<td>iOS, Android</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Provider</th>
<th>Version Control</th>
<th>Encryption</th>
<th>Num. of devices</th>
<th>API</th>
<th>SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Drive</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>Java, Python, PHP, .NET, Ruby</td>
</tr>
<tr>
<td>Dropbox</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>iOS, Android, Python, Ruby, Java, OS X</td>
</tr>
<tr>
<td>SugarSync</td>
<td>+</td>
<td>+</td>
<td>1</td>
<td>+</td>
<td>Java</td>
</tr>
<tr>
<td>Box.com</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>iOS, Android, Python, Ruby, Win, Java, C#</td>
</tr>
</tbody>
</table>
Proposed solution
Main components

- The main components of our proposed service are:
  - the MeasureTool component for performing monitoring processes,
  - the DistributeTool component for splitting and distributing files, and
  - the CollectTool component for retrieving splitted parts of a required file.
Evaluation architecture
MeasureTool evaluation

- We have performed our evaluations on a private IaaS Cloud called SZTAKI Cloud based on OpenNebula.
- For users, the most important metric for measuring provider performance is the *data transfer speed*, so we used this metric to monitor the providers.
- A monitoring process for measuring the performance of a provider consists of:
  - *generating* a file of a predefined size with randomized content,
  - *uploading* this file to the provider's storage under a given user account, then
  - *downloading* this file back to the host of the application.
MeasureTool results

The diagram shows the performance of different cloud storage services: Box.com, Dropbox, Google Drive, and SugarSync, measured in kB/s over datasets of 5MB, 10MB, 50MB, and 100MB. The highest performance is observed with Google Drive for the 50MB and 100MB datasets, indicating a superior performance in these categories.
MeasureTool results

- **Google Drive** had the *best performance* values followed by Dropbox and Box.com, while SugarSync has the worst values.

- It is not easy to compare Box.com and Dropbox, but in general many *small files* are better handled by Dropbox, while *bigger files* are transferred faster by Box.com.

- Regarding reliability, we also measured the *number of failures* experienced during up- and downloading the files.

- For Box.com we experienced a relatively *high number of failures* by *downloading big files* resulted in abortion of the transactions. On the other hand, SugarSync was proved to be the *most reliable* provider without a single failure.
Data distribution evaluation

- Based on the results of the evaluation of the MeasureTool component, our initial hypothesis that service quality levels *differ* for various Cloud providers *has been proven*.
- The DistributeTool component works for a *predefined configuration* based on the *ratio* of the aggregated *historical* performance values and the *latest* performance values.
- We evaluated the performance of our proposed application with 4 *different* configurations:
  - \( r = 0, 0.1, 0.5, \) and \( 0.9 \).
Evaluation configurations

- Historical perf.: 1615, 1085, 5392, 292
- Latest perf.: 1403, 1111, 2200, 301
- r = 0.1: 1755.3, 1196.1, 5612, 322.1
- r = 0.5: 2316.5, 1640.5, 6492, 442.5
- r = 0.9: 2877.7, 2084.9, 7372, 562.9

Legend:
- Dropbox
- Box.com
- Google Drive
- Sugarsync
Evaluation results

The measured speed of Cloud providers during the evaluation
Evaluation results for the proposed application with different configurations
Conclusions

- We addressed the problem area of *data interoperability* in clouds by proposing approaches to manage and share user data in different IaaS clouds.
- We have shown how to develop an image generator application that *interconnects* mobile devices, IaaS and Personal Clouds.
- We introduced an approach for *porting a biochemical application* into a heterogeneous environment consisting of an *infrastructure and a Personal Cloud*, and executing it in an *interoperable and autonomous* way.
- Finally, we proposed a solution for autonomously managing and distributing user data over Personal Cloud providers in an interoperable way.
- Our future work aims at further exploiting Cloud capabilities to support the Internet of Things, and we are *open for collaborations* in related research issues.
Related publications


- R. Tornyai, A. Kertesz, **Towards Autonomous Data Sharing Across Personal Clouds**, 2nd Workshop on *Dependability and Interoperability in Heterogeneous Clouds* (DIHC) in conjunction with EuroPar’14, Porto, Portugal, August 2014.

- T. Planzner, A. Kertesz, **Towards Data Interoperability of Cloud Infrastructures using Cloud Storage Services**, 1st Workshop on *Dependability and Interoperability in Heterogeneous Clouds* in conjunction with EuroPar’13, Aachen, Germany, August 2013.

