Federating Clouds: Solutions and Research Directions

Attila Kertesz

Department of Software Engineering
University of Szeged, Hungary

http://www.inf.u-szeged.hu/~keratt
keratt@inf.u-szeged.hu
Outline

Part I
- Introduction to Cloud Computing
- Solutions for Cloud Federations
- Cloud Federation research issues

Part II
- Enhancing data processing capabilities of mobile devices
- Application porting to Infrastructure and Personal Clouds
- Unifying Personal Clouds
Cloud Computing

LAST YEAR WE RECOGNIZED THAT OUR PROCESSES WERE FAR TOO COMPLEX

SO WE PUT THEM INTO THE CLOUD
A 'cloud' is an elastic execution environment of resources involving multiple stakeholders and providing a metered service at multiple granularities for a specified level of quality.

Cloud service levels

SaaS

PaaS

IaaS

Value Visibility to End Users

Software as a Service

Platform as a Service

Infrastructure as a Service
IaaS cloud usage

1. Upload
2. Delivery
3. Deployment
4. Access

Repository

Virtual Appliance
- Support
- Libs
- OS
- Environment

Host

Infrastructure as a Service Cloud
ARCHITECTURAL MODELS OF CLOUDS
EC view

Private Cloud

Public Cloud

Hybrid Cloud

Community Cloud
European Network and Information Security Agency (ENISA) view
National Institute of Standards and Technology (NIST) view
EUROPEAN PROJECTS ADDRESSING CLOUD FEDERATIONS
EU FP7 project from June 2010 to May 2013

Goal: improved cloud service ecosystem

Holistic approach to multiple co-existing clouds

Cloud life-cycle optimization (cost, trust, risk, eco-efficiency)

Market-oriented multi-cloud architectures

http://www.optimis-project.eu
OPTIMIS

- 1. Federated Cloud Architecture: The Service Provider (SP) assesses an Infrastructure provider (IP). IPs can share resources among each other.
- 2. Multi-Cloud Architecture: Different infrastructure providers are used separately by an SP.
- 3. Hybrid Cloud Architecture: A Private Cloud (PC) is used by the SP, which can utilize resources of different IPs. (~Cloud bursting)
RESERVOIR

- EU FP7 project from February 2008 to January 2011
- Resources and Services Virtualization without Barriers
- *Small and medium clouds* enable an *infinite* computing utility
- Complete service stack to form a *federation*
- *Massive scale deployment* and management across administrative domains
- Resources and services transparently provisioned on-demand
- Server consolidation

http://www.reservoir-fp7.eu/
Reservoir

A Reservoir Cloud consists of different Reservoir sites (RS) operated by different IPs. Each RS has resources that are partitioned into isolated Virtual Execution Environments (VEE). Service applications can use dedicated VEE hosts coming from different RSs. Each application is deployed using a service manifest that formally defines an SLA contract.
BonFIRE

- EU FP7 project from June 2010 to December 2013
- Exploring the interactions between novel service and network infrastructures
  - Extension of cloud offerings towards a federated facility
    - heterogeneous virtualized resources
    - best-effort Internet interconnectivity
  - Experimental network emulation platform where topology configuration and resource usage is under full control of the experimental researcher
  - Exemplify federation mechanisms by interconnecting BonFIRE sites with FEDERICA, Open Cirrus and Panlab

http://www.bonfire-project.eu
mOSAIC

- EU FP7 project from September 2010 to February 2013
- Cloud ontology + *innovative API* = specification of service requirements
- Higher degree of *portability* and vendor *independence*
- *Application programming interfaces* for building applications using services from multiple cloud providers
- Self-adaptive distributed scheduling platform
- Foster competition between cloud providers

http://www.mosaic-cloud.eu/
RESEARCH GROUPS ADDRESSING CLOUD FEDERATIONS
Inter-Cloud overview*
Benefits of an Inter-Cloud

For users:
- Diverse geographical locations
  - Legislation compilant services: decide where data is stored
- Better application resilience
  - Avoid cloud service outages, multiple data centers for fault tolerance
- Avoidance of vendor lock in
  - Distribute workload, price-sensitive usage, easy migration

For providers:
- Expand on demand
  - Offload resource utilization to other providers
- Better SLA to customers
  - Support worst-case scenarios, survive outages
FCM Architecture*

- Autonomously manage the interconnected cloud infrastructures
- Forms a federation with the help of Cloud Brokers

Other research approaches

- Bernstein et al. - 2009:
  - use cases of multi-cloud systems highlighting VM mobility problems
  - they propose interoperability solutions only for low-level functionality of clouds

- Celesti et al. - 2010:
  - Cross-Cloud Federation Manager
  - Three phases: discovery, match-making and authentication
CLASSIFICATION OF FEDERATION APPROACHES

- Hybrid cloud
- Cloud Fusion
- Cross-Cloud
- Federation
- Inter-Cloud
- Cloud bursting
- Multi-Cloud
## Classification

<table>
<thead>
<tr>
<th></th>
<th>Hierarchical</th>
<th>Horizontal</th>
<th>Heterogeneity</th>
<th>Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIMIS</td>
<td>X</td>
<td>-</td>
<td>Yes</td>
<td>Legislation awareness</td>
</tr>
<tr>
<td>Reservoir</td>
<td>-</td>
<td>X</td>
<td>No</td>
<td>Reservoir service stack</td>
</tr>
<tr>
<td>Contrail</td>
<td>X</td>
<td>-</td>
<td>Yes</td>
<td>SLA contracts</td>
</tr>
<tr>
<td>BonFIRE</td>
<td>-</td>
<td>X</td>
<td>Yes</td>
<td>Controlled networking</td>
</tr>
<tr>
<td>mOSAIC</td>
<td>-</td>
<td>X</td>
<td>Yes</td>
<td>Cloud ontology, API</td>
</tr>
<tr>
<td>EGI FedCloud</td>
<td>-</td>
<td>X</td>
<td>Yes</td>
<td>Virtualised EGI environments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hierarchical</th>
<th>Horizontal</th>
<th>Heterogeneity</th>
<th>Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterCloud - Buyya</td>
<td>X</td>
<td>-</td>
<td>Yes</td>
<td>Market-oriented</td>
</tr>
<tr>
<td>Cross-Cloud - Celesti</td>
<td>-</td>
<td>X</td>
<td>Yes/No</td>
<td>Authentication</td>
</tr>
<tr>
<td>Multi-Cloud - Bernstein</td>
<td>X</td>
<td>-</td>
<td>Yes</td>
<td>VM Mobility</td>
</tr>
<tr>
<td>FCM – Marosi</td>
<td>X</td>
<td>-</td>
<td>Yes</td>
<td>Meta-brokering</td>
</tr>
</tbody>
</table>
RESEARCH ISSUES OF FEDERATING CLOUDS
Generalized FCM

Federated Cloud Management

CloudBroker

IP

IP

IP

IP

IP

IP

IP
Monitoring solution in FCM*

Legal aspects of data management in Cloud federations*

- Resource provision in Clouds opens new legal issues, such as data protection, licensing and intellectual property rights.

- Responsibilities for legal compliance can be determined by the Data Protection Directive of the European Union.

- We mapped these roles to Cloud use cases, and identified dynamic role-changing, which may also affect the national law applicable during service execution.

Energy efficient Clouds

Reducing the carbon footprint of European countries is also a must and expected by the European Commission, as well as to increase the number and size of European Cloud providers.
Energy-aware strategies

Parallel simulation of multiple clouds to allow the analysis of federated use cases

Consumption Controller models the on/off policy for physical machines

Current Physical Machine lifecycle:

- OFF – currently no consumption,
- SWITCHINGON, SWITCHINGOFF – simulates the boot/shutdown procedure, consumes energy but does not accept VMs
- ON – consumes energy and accepts VMs

Current consumption model:

- Linear interpolation between idle and max wattage depending on the VM number

Energy-aware VM scheduling problem

- Energy-aware management of datacenters of a single cloud provider specialized for provisioning *task-based* cloud applications

<table>
<thead>
<tr>
<th>Energy consumption of VMs</th>
<th>120W</th>
<th>110W</th>
<th>150W</th>
<th>180W</th>
<th>0W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load of VMs</td>
<td>20%</td>
<td>10%</td>
<td>50%</td>
<td>70%</td>
<td>0%</td>
</tr>
</tbody>
</table>

$\sum \text{MIN}$
Decision making process

- VMs in the Cloud are described with three properties:
  - Power usage counter (PUC): gives the load of CPU usage at the given simulation time.
  - Power consumption counter (PCC): gives the energy usage of its host at a given time. The value is generally between 40 and 120 W, but it depends on its actual physical processor.
  - Number of processors (PROC): gives the number of available processors of its host.

- We have developed a *Pliant decision making algorithm* that takes into account the above-mentioned properties and decides to which VM a task should be submitted.
The Pliant solution

We have developed *two* Pliant decision making algorithms that take into account the above-mentioned properties and decide to which VM a cloudlet should be submitted.

One *optimizes* cloudlet executions for *time*, and the other one for *energy*. We use different normalization for the two strategies.

*First* we start with a *normalization step* and we *apply* different kinds of *Sigmoid functions* to normalize the environment's property value. This table shows the predefined values of the parameters of the normalization functions:

<table>
<thead>
<tr>
<th>Property</th>
<th>Time</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alpha</td>
<td>Lambda</td>
</tr>
<tr>
<td>PUC</td>
<td>0.5</td>
<td>-4.0</td>
</tr>
<tr>
<td>PCC</td>
<td>85.0</td>
<td>-0.08</td>
</tr>
<tr>
<td>PROC</td>
<td>1.0</td>
<td>0.8</td>
</tr>
</tbody>
</table>
The Pliant solution

We wanted to emphasize that it is better to use less power, therefore we created two different parameter sets: one for time-aware and one for energy-aware scheduling.

Here we can see that if the number of power consumption is increasing then the value of the normalized function is decreasing. The opposite is true for the number of processors.
Simulation of Clouds

- We used the Cloudsim Toolkit for evaluation
Prezi workload

In order to investigate the energy consumption of cloud providers, we used real-world trace files of Prezi Inc:

- an international company, who offers a presentation editing service, which needs conversion of media files to other formats before they can display them on all devices.

In April 2013, they launched a competition titled "Scale Contest" for university students to test their knowledge of control and queueing theories on real-life problems.

Their conversion processes are carried out on virtual machines: at peak times, they need to launch more instances of these VMs, but over the weekend they can stop most of them.
Evaluation – Initial strategies

To develop Pliant-based algorithms, we created three initial strategies:

- MINIMUM: uses only one VM to execute all submitted jobs,
- MAXIMUM: deploys a new VM for all jobs,
- SMARTRANDOM: uses randomized VM selection from the available VMs (smartly prioritizing the less loaded ones), and deploys a new one, if no free VM is found.
From these results we can see that utilizing the lowest number of VMs results in the lowest energy consumption, but of course on the expense of the execution time, which is the highest in this case.
Based on these preliminary evaluations we have created a Pliant-based strategy (PLIANTDEFAULT), first focusing on *execution time reduction* with some energy savings.

It could achieve *significant performance gains* in terms of execution time as expected, but it also had much higher energy consumption than the MINIMUM and SMARTRANDOM initial strategy.

<table>
<thead>
<tr>
<th>Hosts</th>
<th>Cloudlets</th>
<th>VMs</th>
<th>Energy (kWh)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1000</td>
<td>14</td>
<td>0.26</td>
<td>749</td>
</tr>
<tr>
<td>100</td>
<td>10000</td>
<td>16</td>
<td>2.87</td>
<td>3768</td>
</tr>
<tr>
<td>100</td>
<td>50000</td>
<td>24</td>
<td>17.26</td>
<td>14240</td>
</tr>
<tr>
<td>100</td>
<td>100000</td>
<td>25</td>
<td>53.21</td>
<td>39304</td>
</tr>
</tbody>
</table>

Evaluation results for PLIANTDEFAULT
Finally, we have modified the parameters of the applied Pliant system, and created more focused algorithms:

- **PLIANTTIME**: uses a Pliant version that is more focused on *execution time savings*,
- **PLIANTENERGY**: uses Pliant parameters to focus on *energy savings*. 
Result comparison*

PART II.
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

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>OS</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>CPU</td>
<td>600 MHz</td>
<td>1 GHz (dual-core)</td>
<td>1 CPU</td>
<td>4 CPUs</td>
</tr>
<tr>
<td>RAM</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

User PC or mobile

App data, data_conf.txt

Personal Cloud

VM1 VM2 VM3
IaaS Cloud1

VM1 VM2 VM3
IaaS Cloud2

1. data_conf.txt
2. data_conf.txt
3. Get data
4. Put results
5. data_conf.txt
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
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>
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

![Bar chart showing evaluation results with different VM configurations.](chart.png)
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

![Bar chart showing evaluation results for 3 ONe VMs + 3 AM VMs across different phases.]

- Phase 1: 11:04
- Phase 2: 5:11
- Phase 3: 1:10
Detailed evaluation results
Unifying Personal Clouds

- 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 [14]</td>
<td>15</td>
<td>-</td>
<td>15</td>
<td>Win, Mac</td>
<td>iOS, Android</td>
</tr>
<tr>
<td>Dropbox [15]</td>
<td>2</td>
<td>0.5</td>
<td>8</td>
<td>Win, Mac, Linux</td>
<td>iOS, Android</td>
</tr>
<tr>
<td>SugarSync [17]</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>Win, Mac</td>
<td>iOS, Android</td>
</tr>
<tr>
<td>Box.com [18]</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 [14]</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>Java, Python, PHP, .NET, Ruby</td>
</tr>
<tr>
<td>SugarSync [17]</td>
<td>+</td>
<td>+</td>
<td>1</td>
<td>+</td>
<td>Java</td>
</tr>
<tr>
<td>Box.com [18]</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 graph shows the results of MeasureTool for different file sizes (5MB, 10MB, 50MB, 100MB) across various cloud storage services: Box.com, Dropbox, Google Drive, and SugarSync. The y-axis represents the file transfer speed in kB/s, and the x-axis represents the file size in MB.
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:
  - $r = 0, 0.1, 0.5, \text{ and } 0.9$. 
Evaluation configurations

- **historical perf.**
  - 0%: 1615
  - 10%: 1085
  - 20%: 292
  - 30%: 5392
  - 40%: 1615
  - 50%: 1085
  - 60%: 292
  - 70%: 5392
  - 80%: 1615
  - 90%: 1085
  - 100%: 292

- **latest perf.**
  - 0%: 1403
  - 10%: 1111
  - 20%: 301
  - 30%: 2200
  - 40%: 1403
  - 50%: 1111
  - 60%: 301
  - 70%: 2200
  - 80%: 1403
  - 90%: 1111
  - 100%: 301

- **r = 0.1**
  - 0%: 1755.3
  - 10%: 1196.1
  - 20%: 322.1
  - 30%: 5612
  - 40%: 1755.3
  - 50%: 1196.1
  - 60%: 322.1
  - 70%: 5612
  - 80%: 1755.3
  - 90%: 1196.1
  - 100%: 322.1

- **r = 0.5**
  - 0%: 2316.5
  - 10%: 1640.5
  - 20%: 442.5
  - 30%: 6492
  - 40%: 2316.5
  - 50%: 1640.5
  - 60%: 442.5
  - 70%: 6492
  - 80%: 2316.5
  - 90%: 1640.5
  - 100%: 442.5

- **r = 0.9**
  - 0%: 2877.7
  - 10%: 2084.9
  - 20%: 562.9
  - 30%: 7372
  - 40%: 2877.7
  - 50%: 2084.9
  - 60%: 562.9
  - 70%: 7372
  - 80%: 2877.7
  - 90%: 2084.9
  - 100%: 562.9

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

The measured speed of Cloud providers during the evaluation
Evaluation results

Evaluation results for the proposed application with different configurations
IoT integration – The future

- Internet of Things is here
- Gartner reports: 30 billion devices will be online, and 200 billion offline by 2020!
- With the help of cloud solutions, user data can be stored in a remote location, and can be accessed and processed from anywhere
Envisioned IoT-Cloud Ecosystem
Simulator for IoT Cloud systems

- MobIoTSim: to simulate IoT devices
- Gateways to receive, store and process sensor data
- Research issues: scalability, security, data protection
Related publications


Thank you for your attention!