CDOSim: Simulating Cloud Deployment Options for Software Migration Support

Florian Fittkau, Sören Frey, and Wilhelm Hasselbring

Software Engineering Group,
Kiel University, Germany

24.09.2012

Motivation

- Migration of enterprise software to the cloud
- Many different cloud deployment options
- Simulation helps to find the best trade-off between high performance and low costs

Cloud Deployment Option (CDO)

In the context of a deployment of software on a cloud platform, a cloud deployment option is a **combination of decisions** concerning the

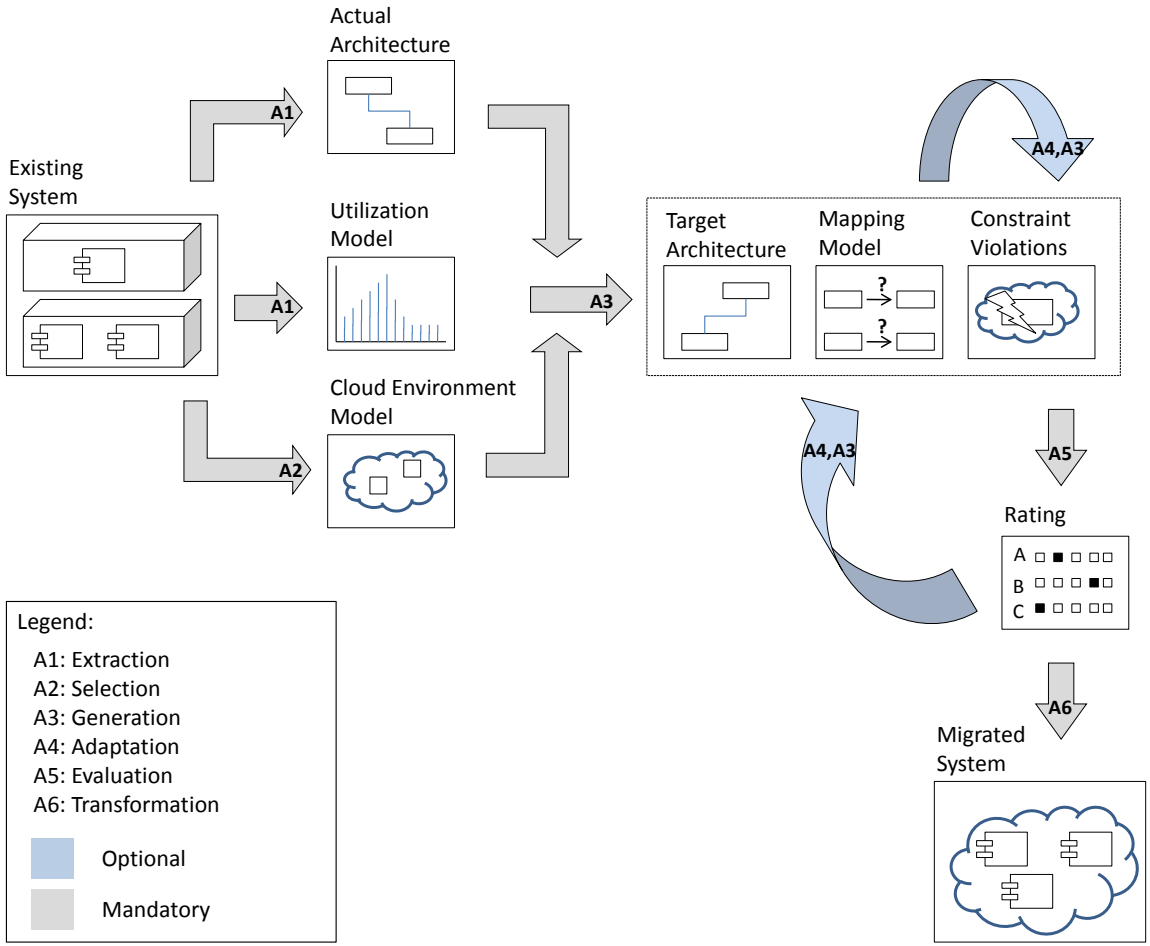
- selection of a cloud provider,
- the deployment of components to virtual machine instances,
- the virtual machine instances' configuration,
- and specific adaptation strategies.

CloudSim

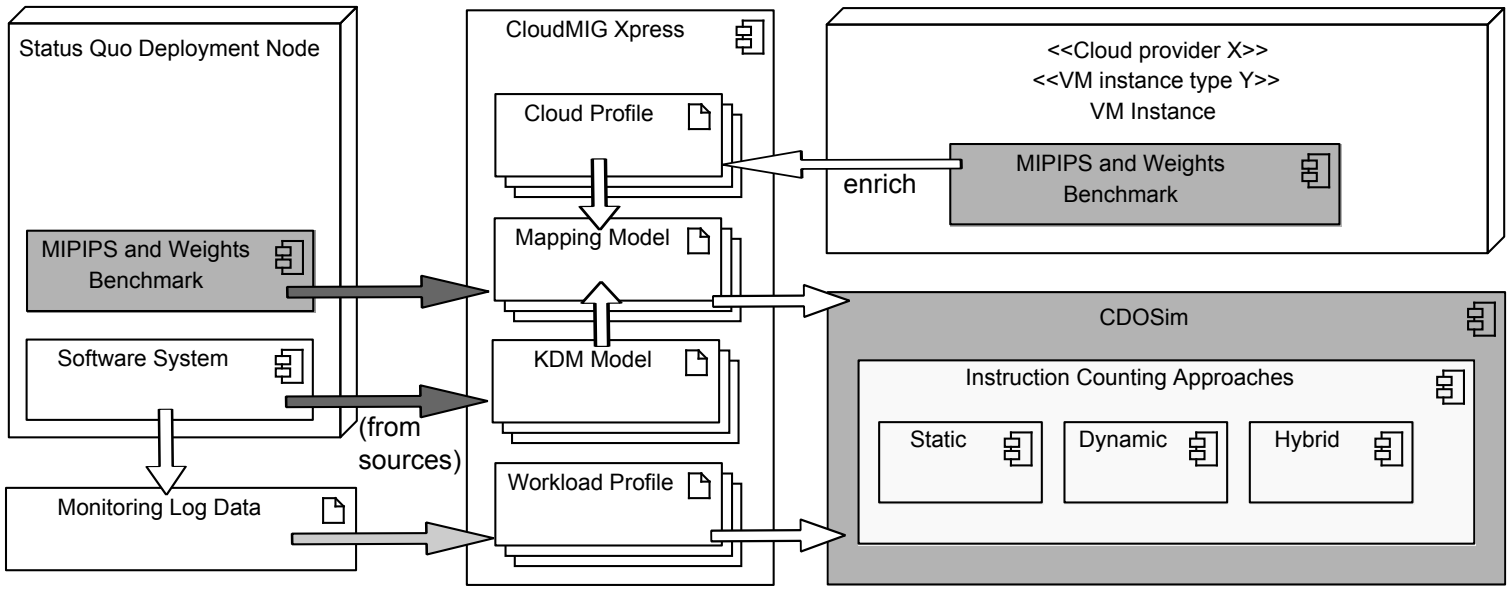
- CloudSim [CRDRB09]
  - Cloud computing system and application simulator
  - Cloud provider perspective
  - We extended it by cloud user perspective

CloudMIG

- CloudMIG approach and its prototype CloudMIG Xpress [FHS12, FH11]

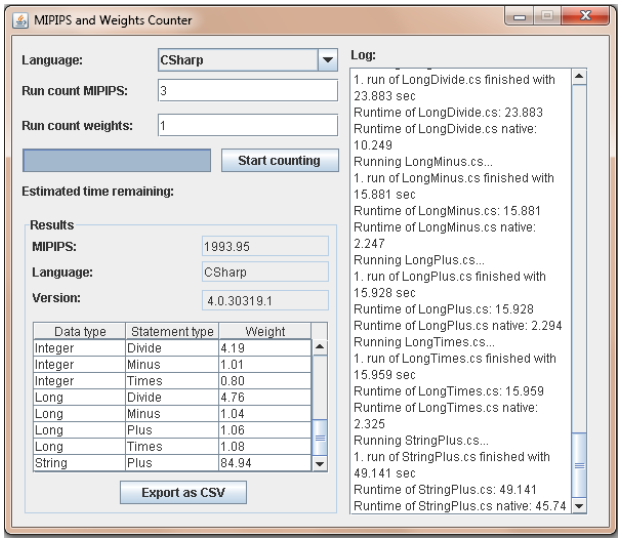


CDOSim – Simulation Architecture



Million Integer Plus Instructions Per Second (MIPIPS)

- Measure for the computing performance of a computer / virtual machine instance
- Idea: Measure the execution time and divide by instruction count
- Example: 10 seconds for 200 million instructions results in 20 MIPIPS
- Benchmark generated from meta-model with current support for Java, C, C++, C#, Ruby, Python



MIPIPS

```
1  int x = 0;
2
3
4  long startTime = System.currentTimeMillis();
5
6  int i = -2147483647;
7  while (i < 2147483647) {
8    x = x + 2;
9
10   i += 1;
11 }
12
13 long endTime = System.currentTimeMillis();
14 long difftime = endTime - startTime;
15 System.out.println(difftime);
16 System.out.println(x);
17
```

MIPIPS

```
1  int x = 0;
2  int y = 0;
3
4  long startTime = System.currentTimeMillis();
5
6  int i = -2147483647;
7  while (i < 2147483647) {
8    x = x + 2;
9    y = y + 3;
10   i += 1;
11 }
12
13 long endTime = System.currentTimeMillis();
14 long difftime = endTime - startTime;
15 System.out.println(difftime);
16 System.out.println(x);
17 System.out.println(y);
```

Instruction Count Overview

Approach	**Preconditions**
Dynamic approach	1. Part of source code 2. Response times 3. MIPIPS
Static approach	1. Full source code
Hybrid approach	1. Full source code 2. Response times 3. MIPIPS

Dynamic Approach

- Approach: MIPIPS divided by the response time reveals instruction count
- Example: 200 MIPIPS / 0.1 seconds = 20 million integer plus instructions

Static Approach

- Approach: Count each instruction and convert to integer plus instruction through weight
- Example: Convert a double times to an integer plus instruction

Static Approach - Example

Equation for loop instruction count derivation:

$$ic_{for\_loop} = ic_{init} + (iter_{count} \cdot (ic_{cond} + ic_{iter} + ic_{loop}))$$

Example:

```
1 for (int i = 0; i < 10; i++) {
2   x = i + 3;
3 }
```

$$ic_{for\_loop} = 1 + (10 \cdot (1 + 1 + 1)) = 31$$

Hybrid Approach

- Dynamic approach: Most often no data from a fully-instrumentated system is available, but the monitored data is accurate
- Static approach: Detailed insight but imprecise
- Hybrid approach combines the advantages of both
- Idea: Use dynamic analysis results for correction of static analysis results

Hybrid Approach

```
1
2  public void method3000() { // from static:  3000 IC
3    for (int i = 0; i < 1000; i++) {
4      x = i + 3;
5    }
6  }
7
8  public void method50() {   // from dynamic: 50 IC
9    method3000();
10 }
```

Hybrid Approach

```
1                                   // from hybrid:  50 IC
2  public void method3000() { // from static:  3000 IC
3    for (int i = 0; i < 1000; i++) {
4      x = i + 3;
5    }
6  }
7
8  public void method50() {   // from dynamic: 50 IC
9    method3000();
10 }
```

Weights Per Statement

- For example, a double divide instruction takes more time than an integer plus instruction on most platforms
- Idea: Convert double divide instruction into integer plus instruction
- Approach: Divide MIPIPS by million double divide instructions per seconds (MDDIPS) from adapted benchmark
- Example: 400 MIPIPS / 100 MDDIPS = 4

Simulation Output

- Costs
- Response times
- SLA violations
- Rating: Rate each output from 1 (best) to 5 (worst)

Evaluation Overview

- E1: MIPIPS benchmark evaluation
- E2: Accuracy evaluation for single core instances
- E3: Inter-cloud accuracy evaluation

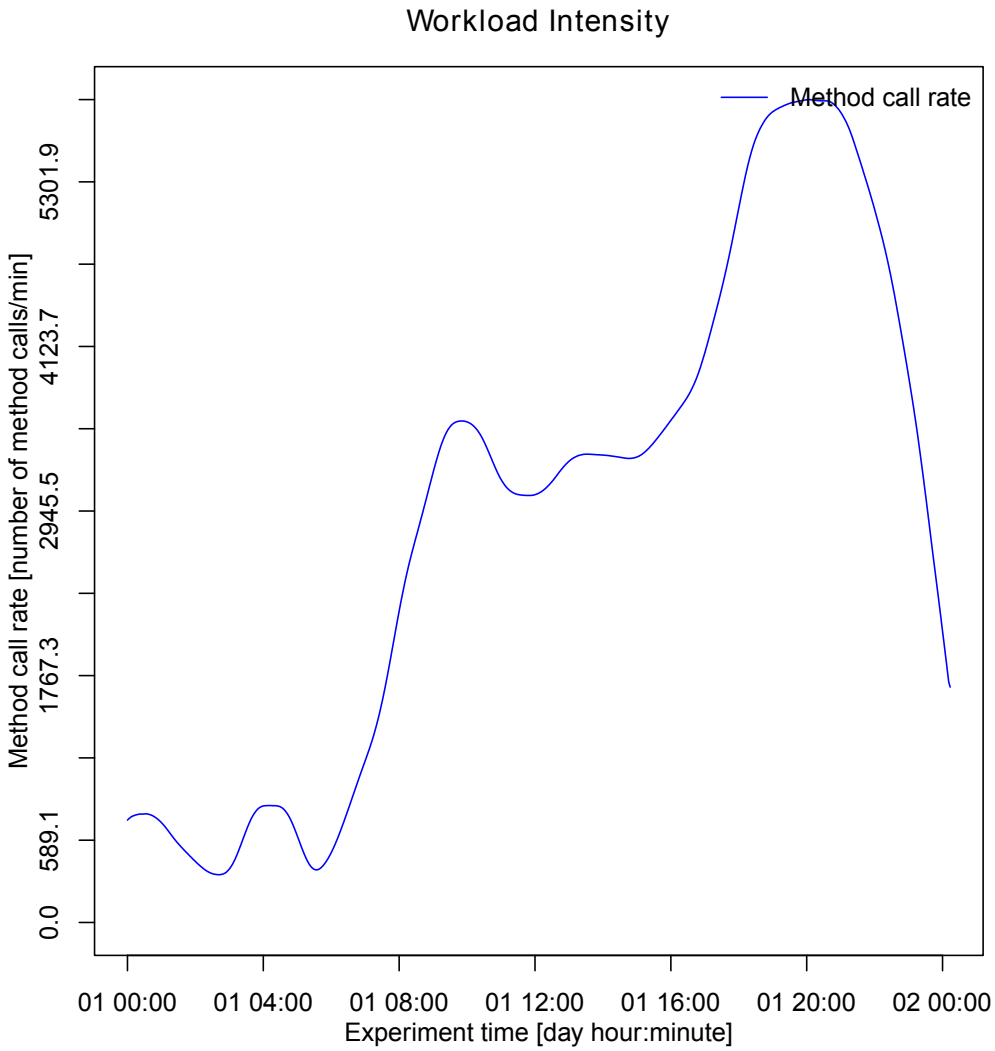
More evaluations in [Fit12]

Experiment Setup for E2 and E3

- Adapted JPetStore
- JMeter with Markov4JMeter
- Kieker [vHWH12] (monitoring framework) `kieker-monitoring.net`
- Eucalyptus[1] and Amazon EC2
- Quantifying the relative error (RE) by comparing simulated values with measured values

[1] 2x AMD Opteron 2384 (8 cores), 24 GB DDR2-667 RAM

Workload

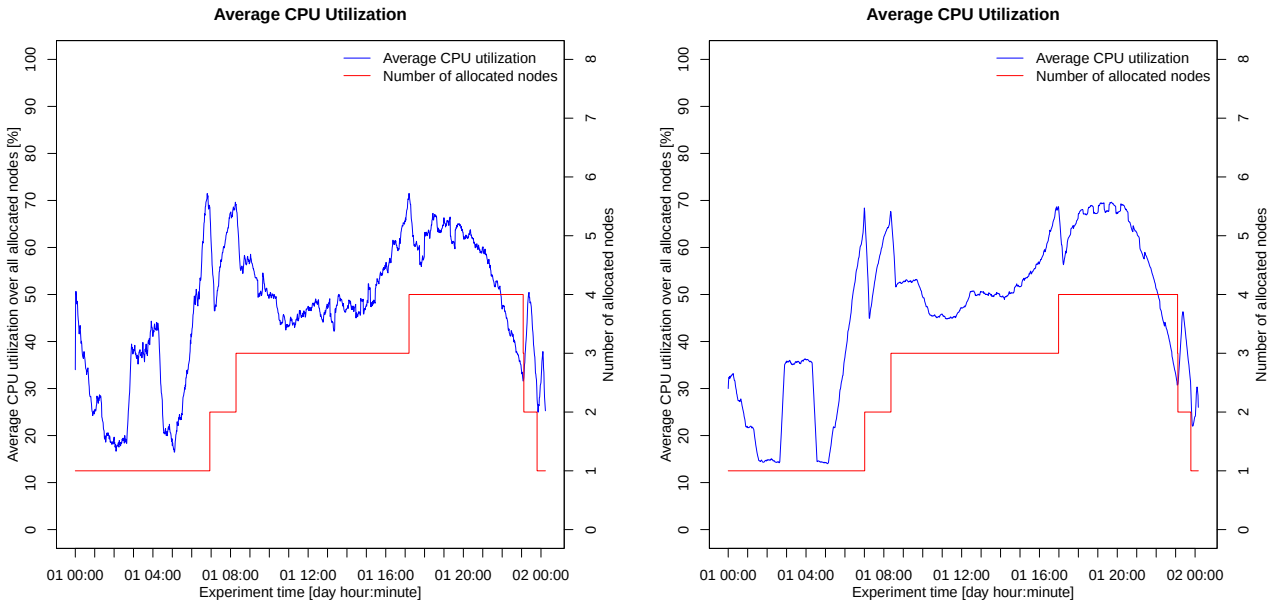

Workload Intensity
Method call rate
Method call rate [number of method calls/min]
0.0
589.1
1767.3
2945.5
4123.7
5301.9
01 00:00
01 04:00
01 08:00
01 12:00
01 16:00
01 20:00
02 00:00
Experiment time [day hour:minute]

E1: MIPIPS Benchmark Evaluation – Reasonable to Other Measures

Amazon EC2 instance type	**MIPIPS**	**EC2 compute units per core**
t1.micro	4.11	up to 2
m1.small	20.65	1
m1.large	142.13	2
c1.medium	148.81	2.5
m2.xlarge	235.57	3.25

Table 1 : MIPIPS benchmark results for Amazon EC2

E2: Accuracy Evaluation for Single Core Instances

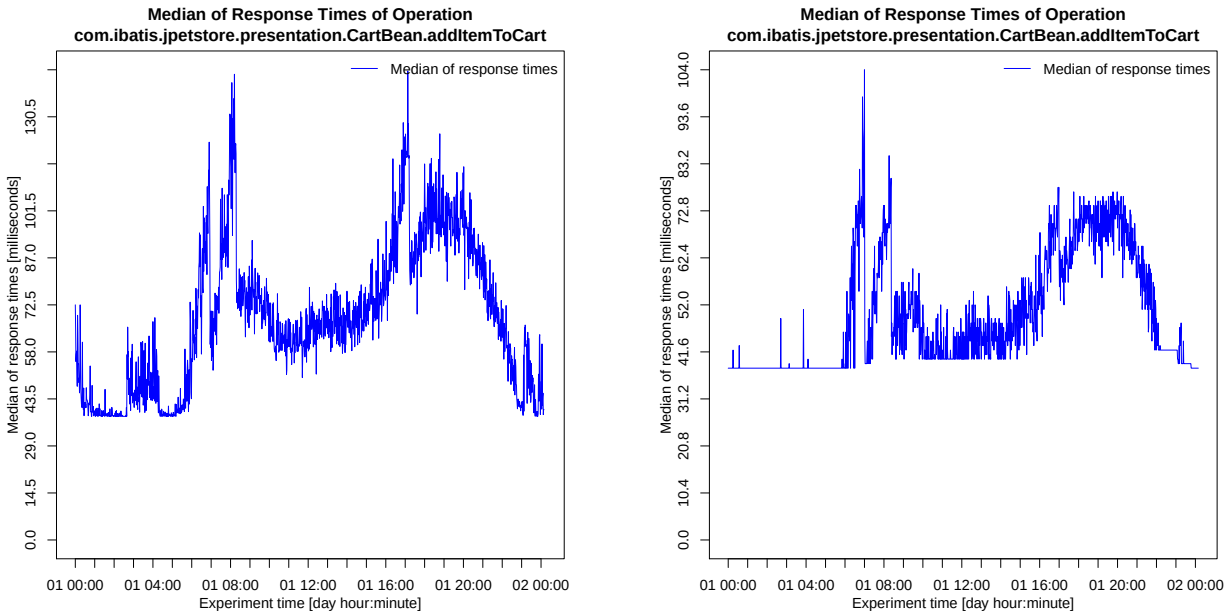


(a) Measured CPU utilization (b) Simulated CPU utilization

Figure 1 : Average CPU utilization of allocated nodes using Eucalyptus

$RE_{CPU} = 29.18\ \%$ $RE_{InstanceCount} = 0.64\ \%$ $RE_{Costs} = 6.34\ \%$

E2: Accuracy Evaluation for Single Core Instances

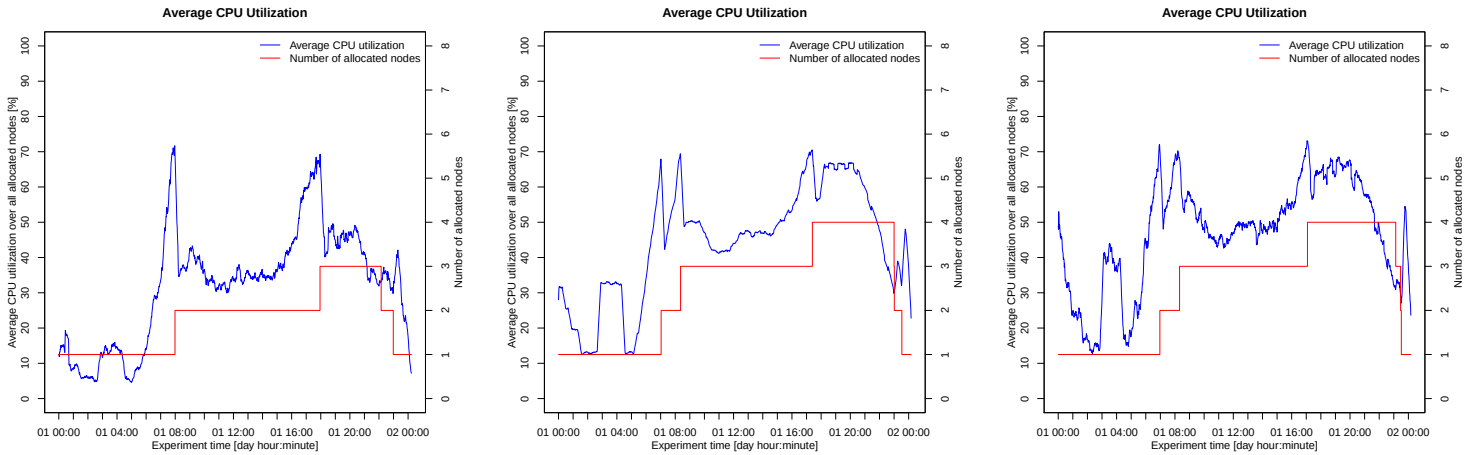


(a) Measured response times (b) Simulated response times

Figure 2 : Median of response times using Eucalyptus

$RE_{RT} = 24.85\ \%$

E3: Inter-Cloud Accuracy Evaluation



(a) Amazon EC2 run (b) Simulation for Eucalyptus (c) Eucalyptus run

Figure 3 : Average CPU utilization of allocated nodes

$RE_{CPU} = 21.60\ \%$ $RE_{InstanceCount} = 1.32\ \%$

$RE_{Costs} = 1.53\ \%$ $RE_{RT} = 38.62\ \%$

Related Work

- GroudSim [OPPF10] (alternative to CloudSim)
- SLAstic.SIM [vMvHH11] (performance simulator based on Palladio Component Model)
- iCanCloud [NCVP+11] (cloud tool with manual application modelling)
- Cloudstone toolkit [SSS+08] (benchmark and measurement tools for Web 2.0)
- SMICloud [GVB11] (framework for comparing different cloud providers)

Future Work

- Framework for parallelizing CDOSim's simulations
- Extend elementary model for computing network costs
- Simulate further properties, e.g., memory consumption and I/O performance
- Use CDOSim for a simulation-based evolutionary optimization of CDOs

Conclusions

- CDOSim helps assessing CDO candidates and finding best suited CDO
- Three approaches for instruction count derivation
- MIPIPS and weights benchmark
- Simulation results can be used to appropriately predict costs, response times, and SLA violations of specific CDOs
- CDOSim is provided as part of our tool CloudMIG Xpress[2]

[2] http://www.cloudmig.org/

Methology

E1:

- Mean value and the standard deviation

E2 and E3:

- Quantifying the relative error (RE) by comparing simulated values with measured values

$$re(t) = \frac{|m(t) - s(t)|}{m(t)}, \; m(t) \neq 0, \; t \in T$$

$$RE = \frac{\Sigma_t \; re(t)}{|T|}$$

$$OverallRE = \frac{RE_{CPU} + RE_{InstanceCount} + RE_{Costs} + RE_{RT}}{4}$$

References

Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, and Rajkumar Buyya.
CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services.
CoRR, abs/0903.2525, 2009.

Sören Frey and Wilhelm Hasselbring.
The cloudmig approach: Model-based migration of software systems to cloud-optimized applications.
International Journal on Advances in Software, 4(3 and 4):342–353, 2011.

Sören Frey, Wilhelm Hasselbring, and Benjamin Schnoor.
Automatic Conformance Checking for Migrating Software Systems to Cloud Infrastructures and Platforms.
Journal of Software Maintenance and Evolution: Research and Practice, 2012.
doi: 10.1002/smr.582.

Florian Fittkau.
Simulating Cloud Deployment Options for Software Migration Support.

Master's thesis, Software Engineering Group, University of Kiel, Kiel, Germany, March 2012.

S.K. Garg, S. Versteeg, and R. Buyya.
SMICloud: A Framework for Comparing and Ranking Cloud Services.
In Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC 11), pages 210–218, December 2011.

A. Nuñez, G.G. Castane, J.L. Vazquez-Poletti, A.C. Caminero, J. Carretero, and I.M. Llorente.
Design of a flexible and scalable hypervisor module for simulating cloud computing environments.
In 2011 International Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS), pages 265 –270, June 2011.

Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer.
GroudSim: An Event-based Simulation Framework for Computational Grids and Clouds.
In CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing. Springer, August 2010.

Will Sobel, Shanti Subramanyam, Akara Sucharitakul, Jimmy Nguyen, Hubert Wong, Arthur Klepchukov, Sheetal Patil, O Fox, and David Patterson.

Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0.
In Proceedings of the 1st Workshop on Cloud Computing (CCA 08), October 2008.

André van Hoorn, Jan Waller, and Wilhelm Hasselbring.
Kieker: A framework for application performance monitoring and dynamic software analysis.
In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012), pages 247–248. ACM, April 2012.

Robert von Massow, André van Hoorn, and Wilhelm Hasselbring.
Performance simulation of runtime reconfigurable component-based software architectures.
In Ivica Crnkovic, Volker Gruhn, and Matthias Book, editors, Software Architecture, volume 6903 of Lecture Notes in Computer Science, pages 43–58. Springer Berlin / Heidelberg, 2011.