TÉCNICO LISBOA

Distributed Programming in Cloud Computing Platforms

Enes UYSAL

Thesis to obtain the Master of Science Degree in xx

Information Systems and Computer Engineering

Supervisor(s): Prof. José Carlos Martins Delgado

Examination Committee

Chairperson: Prof. Full Name
Supervisor: Prof. Full Name 1 (or 2)
Member of the Committee: Prof. Full Name 3

May 2016

Dedicated to someone special...

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Inserir o resumo em Português aqui com o máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave...

Palavras-chave: palavra-chave1, palavra-chave2,...

Abstract

Insert your abstract here with a maximum of 250 words, followed by 4 to 6 keywords...

Keywords: keyword1, keyword2,...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Insert your chapter material here...

1.1 Motivation

Relevance of the subject...

1.2 Topic Overview

Provide an overview of the topic to be studied...

1.3 Objectives

Explicitly state the objectives set to be achieved with this thesis...

1.4 Thesis Outline

Briefly explain the contents of the different chapters...

Chapter 2

State-of-the-art

2.1 Overview

(description of existing technologies and tools that you will use) Existing Technologies

- SOA (Youtube)

- REST (YouTube)

Tools Tomcat .Net Framework 4.5

Language (C - Java) Deployment - Azure Web Services Cloud Technoloies Insert your chapter material here...

2.2 Web Services

Web Services are exposed to the Internet for programmatic access. They are online APIs that you can call from your code.

When you want to call any API that written by someone else to your java code, you basically add jar or classes to your class path and executions are done inside of machine or single environment. In the case of web services however you have different pieces of code deployed over different machines and they call methods of each other over the network. For example, you must have seen that different apps or games, which can post to your Facebook wall even these games, not designed by Facebook. So you ask that how they can do that or how they can post to a wall of completely different system or application. Basically they do this by calling online APIs. Companies like Facebook or Twitter publish web services that let other developers call them from their code, so other application developers can actually write code to consume these services and they can post things on Facebook or Twitter. They can read or access data from Facebook or Twitter using the APIs of the web services that Facebook or Twitter has provided.

Web services are similar to web pages. For example Twitter has web side URL as “www.twitter.com“ when you access this URL on your browser you get an HTML response that let you read and write tweets. They have HTML elements for data and also CSS files for styling, this is because web pages

that you see is made human conception. They know that there is actually human is behind of browser on a laptop or devices who was reading these tweets so they want to make sure about its format properly, so it is easy to access and read. Twitter has also other URL as “api.twitter.com“ that does a lot of same things as “www.twitter.com“ does, but it behaves a bit differently for instance this API gives you response which doesn’t have HTML or CSS code. It contains data but it is xml or json format and there are specific URLs for different operations this is what the developers can use from their code to read or write to twitter, so this data is actually very easy for parsing and converting then using in their objects and their code for developers. In this case there is no need HTML and CSS files.
There are primarily two different types of web services. One of these types called as SOA web service and another type called as REST web service. SOA is older of these two and REST is newer entry to web services world, but both of them are used popular. Next chapter will be focusing on SOA and REST web services.

2.3 SOA Web Services

DEFINITON

xXXX https://books.google.fr/books/about/Understanding$_{W}eb_{S}ervices.html?id = SHSBri-rMyQCredir_{e}sc = y$

EXAMPLE

Let’s start an example with java application. Let’s say we have implementation class and I want to share this implementation class with other developer projects. How I would share this with a consumer class. The best way to share this implementation class would be to contract it with an interface and other consumers would consume this class through this interface. They would call implementation through interface so they get contract and they get the methods, arguments, written types through interface. They actually call the methods of implementation class, so how this works in case of web service, let’s say I have web service implementation and I want to share details of this web service to consumers. Is it works with an interface? Probably it will not work because as we discussed before you don’t know what technology is consuming it. It might be C application or C++ application, so if you have a java web service you might want to give some kind of information that its consumer respects to that technology and that can actually consume. Let’s say consumer is .Net and if I give this .Net application a java interface then most probably it will not work because they are different technologies, so the technology that I am going to share with web service consumer has to be technology independent. It should be something that any application or any technology can understand so creators of SOA web service talked about that problem and what to give format understandable by all technology with all consumers and decided with XML. So what you do is in case of web service, you actually share that contract as an XML document. This XML document is actually called as WSDL.

[GRAPH] -¿ interface similar to video

WSDL document contains the contract to web service and so that's are the things you have to do when you create the web service and you share WSDL document of that web service to the consumers, so this is not something you would have to do it manually. You would do it manually but there are tools which generate WSDL for the web service but it is something that you need to share this WSDL to consumers and it is a XML document, so it respective whatever application because applications such as .Net, C++ or Java can all parse this XML and get to know about service information and typically the content of this WSDL is kind of similar to an interface content. It has operations, arguments and types to return that consumer applications will have idea what to call and how to call.

[GRAPH] WSDL similar to video

The new question is that how this exchange happens, how you actually send this information, let's say you have a method in your java application and input argument is a string so you have a java string with you and you need to send to web service and let's say output return type is a list, so how you get this information because that could be .Net application and string in java obviously different from string in C. How do you exchange this between client app and web service? When you exchange information input argument or return type you need to exchange it in the format that all different technologies can understand what you are passing and it should be able to send return type back in language that all these technologies can understand. Again this format is XML. When you are sending any information across the network from a client to the web services and return type back to the client, the data has to be in XML format. You are not really sending java string or a list. So it has to be language natural format which is XML. There is specification about how you need to send all these different input type and output argument basically any type needs to be send specific XML format.

It is a protocol that is a way in which both sender and receiver and this XML is called SOAP (Simple Object Access Protocol). It is a way in which these different technologies can access objects can access data it supposedly simple so that a part of the name is called simple object access protocol. So with this protocol all different technologies written in different languages can kind of understand what they all taking about.

[GRAPH] SOAP

Now you know what is the mechanism you know what need to be send and you know how to send which is using SOAP protocol but who does conversion? So for example you have your string object or complex object, so how do you convert it from java object to a soap message? The conversion is actually done with intermedia class so this class takes care of converting all your objects into a SOAP message. The whole method calls itself is actually done by SEI. The SEI access interface to your web service endpoint so you have an interface at your client app to the service endpoint which translate all web service call to a SOAP message and then it makes sure that the other things is able to understand

this message. So we don't have to write this class and all the conversion ourselves. We can have it automatically generated for us. When you are making a web service call you don't worry about where the web service is. When you need to call, all you need to do is have this endpoint interface and good thing about this service endpoint that you can actually have an interface that specific to what you are developing. When you have a java application you will have a specific SEI for java application and it knows to convert java objects to SOAP message. Let's say your .Net application calling the same web service so you will have SEI for .NET that know to convert .Net objects to SOAP message.

GRAPH SEI

2.4 Theoretical Model 1

The research should be supported with a comprehensive list of references. These should appear whenever necessary, in the limit, from the first to the last chapter.

A reference can be cited in any of the following ways:

- Citation mode #1 - [1]
- Citation mode #2 - Jameson et al. [1]
- Citation mode #3 - [1]
- Citation mode #4 - Jameson, Pierce, and Martinelli [1]
- Citation mode #5 - [1]
- Citation mode #6 - Jameson et al. 1
- Citation mode #7 - 1
- Citation mode #8 - Jameson et al.
- Citation mode #9 - 1998
- Citation mode #10 - [1998]

Several citations can be made simultaneously as [2, 3].

This is often the default bibliography style adopted (numbers following the citation order), according to the options:
`\usepackage{natbib}` in file `Thesis_Preamble.tex`,
`\bibliographystyle{abbrvnat}` in file `Thesis.tex`.
Notice however that this style can be changed from numerical citation order to authors' last name with the options:

\usepackage[numbers]{natbib} in file Thesis_Preamble.tex,

\bibliographystyle{abbrvunsrtnat} in file Thesis.tex.

2.5 Theoretical Model 2

Other models...

Chapter 3

Implementation

Insert your chapter material here...

3.1 Numerical Model

Description of the numerical implementation of the models explained in Chapter **??**...

3.2 Verification and Validation

Basic test cases to compare the implemented model against other numerical tools (verification) and experimental data (validation)...

Chapter 4

Results

Insert your chapter material here...

4.1 Problem Description

Description of the baseline problem...

4.2 Baseline Solution

Analysis of the baseline solution...

4.3 Enhanced Solution

Quest for the optimal solution...

4.3.1 Figures

Insert your section material and possibly a few figures...

Make sure all figures presented are referenced in the text!

Images

Figure 4.1: Caption for figure.

(a) Airbus A320

(b) Bombardier CRJ200

Figure 4.2: Some aircrafts.

Make reference to Figures 4.1 and 4.2.

By default, the supported file types are *.png,.pdf,.jpg,.mps,.jpeg,.PNG,.PDF,.JPG,.JPEG*.

See `http://mactex-wiki.tug.org/wiki/index.php/Graphics_inclusion` for adding support to other extensions.

Drawings

Insert your subsection material and for instance a few drawings...

The schematic illustrated in Fig. 4.3 can represent some sort of algorithm.

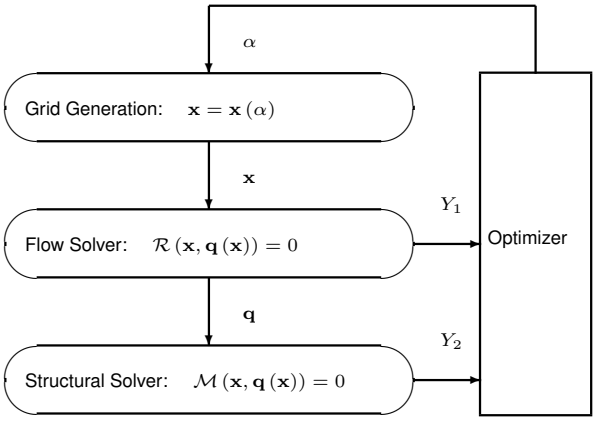


Figure 4.3: Schematic of some algorithm.

4.3.2 Equations

Equations can be inserted in different ways.

The simplest way is in a separate line like this

$$\frac{\mathrm{d}q_{ijk}}{\mathrm{d}t} + \mathcal{R}_{ijk}(\mathbf{q}) = 0\,. \tag{4.1}$$

If the equation is to be embedded in the text. One can do it like this $\partial\mathcal{R}/\partial\mathbf{q} = 0$.

It may also be split in different lines like this

$$\begin{aligned} \text{Minimize} \quad & Y(\alpha, \mathbf{q}(\alpha)) \\ \text{w.r.t.} \quad & \alpha\,, \\ \text{subject to} \quad & \mathcal{R}(\alpha, \mathbf{q}(\alpha)) = 0 \\ & C(\alpha, \mathbf{q}(\alpha)) = 0\,. \end{aligned} \tag{4.2}$$

It is also possible to use subequations. Equations 4.3a, 4.3b and 4.3c form the Naver–Stokes equations 4.3.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_j\right) = 0\,, \tag{4.3a}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j + p\delta_{ij} - \tau_{ji}\right) = 0, \quad i = 1, 2, 3\,, \tag{4.3b}$$

$$\frac{\partial}{\partial t}\left(\rho E\right) + \frac{\partial}{\partial x_j}\left(\rho E u_j + p u_j - u_i \tau_{ij} + q_j\right) = 0\,. \tag{4.3c}$$

4.3.3 Tables

Insert your subsection material and for instance a few tables...

Make sure all tables presented are referenced in the text!

Follow some guidelines when making tables:

- Avoid vertical lines
- Avoid “boxing up” cells, usually 3 horizontal lines are enough: above, below, and after heading
- Avoid double horizontal lines
- Add enough space between rows

Model	C_L	C_D	C_{My}
Euler	0.083	0.021	-0.110
Navier–Stokes	0.078	0.023	-0.101

Table 4.1: Table caption.

Make reference to Table 4.1.

Tables 4.2 and 4.3 are examples of tables with merging columns:

An example with merging rows can be seen in Tab.4.4.

If the table has too many columns, it can be scaled to fit the text widht, as in Tab.4.5.

	Virtual memory [MB]	
	Euler	Navier–Stokes
Wing only	1,000	2,000
Aircraft	5,000	10,000
(ratio)	$5.0\times$	$5.0\times$

Table 4.2: Memory usage comparison (in MB).

	$w = 2$			$w = 4$		
	$t = 0$	$t = 1$	$t = 2$	$t = 0$	$t = 1$	$t = 2$
$dir = 1$						
c	0.07	0.16	0.29	0.36	0.71	3.18
c	-0.86	50.04	5.93	-9.07	29.09	46.21
c	14.27	-50.96	-14.27	12.22	-63.54	-381.09
$dir = 0$						
c	0.03	1.24	0.21	0.35	-0.27	2.14
c	-17.90	-37.11	8.85	-30.73	-9.59	-3.00
c	105.55	23.11	-94.73	100.24	41.27	-25.73

Table 4.3: Another table caption.

ABC	header			
	1.1	2.2	3.3	4.4
IJK	group		0.5	0.6
			0.7	1.2

Table 4.4: Yet another table caption.

Variable	a	b	c	d	e	f	g	h	i	j
Test 1	10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000	90,000	100,000
Test 2	20,000	40,000	60,000	80,000	100,000	120,000	140,000	160,000	180,000	200,000

Table 4.5: Very wide table.

4.3.4 Mixing

If necessary, a figure and a table can be put side-by-side as in Fig.4.4

Legend		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Figure 4.4: Figure and table side-by-side.

Chapter 5

Conclusions

Insert your chapter material here...

5.1 Achievements

The major achievements of the present work...

5.2 Future Work

A few ideas for future work...

Bibliography

[1] A. Jameson, N. A. Pierce, and L. Martinelli. Optimum aerodynamic design using the Navier–Stokes equations. In *Theoretical and Computational Fluid Dynamics*, volume 10, pages 213–237. Springer-Verlag GmbH, Jan. 1998.

[2] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 2^{nd} edition, 2006. ISBN:978-0387303031.

[3] A. C. Marta, C. A. Mader, J. R. R. A. Martins, E. van der Weide, and J. J. Alonso. A methodology for the development of discrete adjoint solvers using automatic differentiation tools. *International Journal of Computational Fluid Dynamics*, 99(9–10):307–327, Oct. 2007. doi:10.1080/10618560701678647.

Appendix A

Vector calculus

In case an appendix if deemed necessary, the document cannot exceed a total of 100 pages...

Some definitions and vector identities are listed in the section below.

A.1 Vector identities

$$\nabla \times (\nabla \phi) = 0 \tag{A.1}$$

$$\nabla \cdot (\nabla \times \mathbf{u}) = 0 \tag{A.2}$$

Appendix B

Technical Datasheets

It is possible to add PDF files to the document, such as technical sheets of some equipment used in the work.

B.1 Some Datasheet

SUNPOWER

C60 SOLAR CELL

MONO CRYSTALLINE SILICON

BENEFITS

Maximum Light Capture
SunPower's all-back contact cell design moves gridlines to the back of the cell, leaving the entire front surface exposed to sunlight, enabling up to 10% more sunlight capture than conventional cells.

Superior Temperature Performance
Due to lower temperature coefficients and lower normal cell operating temperatures, our cells generate more energy at higher temperatures compared to standard c-Si solar cells.

No Light-Induced Degradation
SunPower n-type solar cells don't lose 3% of their initial power once exposed to sunlight as they are not subject to light-induced degradation like conventional p-type c-Si cells.

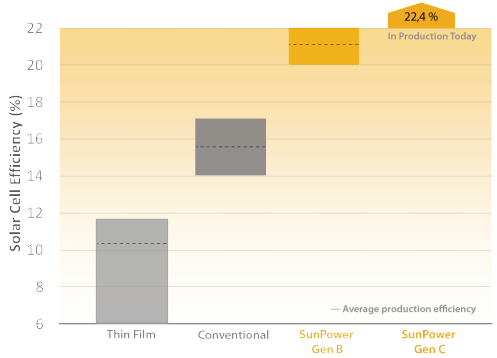
Broad Spectral Response
SunPower cells capture more light from the blue and infrared parts of the spectrum, enabling higher performance in overcast and low-light conditions.

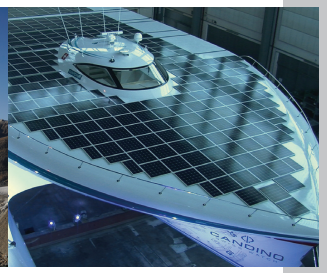
Broad Range Of Application
SunPower cells provide reliable performance in a broad range of applications for years to come.

The SunPower™ C60 solar cell with proprietary Maxeon™ cell technology delivers today's highest efficiency and performance.

The anti-reflective coating and the reduced voltage-temperature coefficients provide outstanding energy delivery per peak power watt. Our innovative all-back contact design moves gridlines to the back of the cell, which not only generates more power, but also presents a more attractive cell design compared to conventional cells.

SunPower's High Efficiency Advantage





C60 SOLAR CELL

SUNPOWER

C60 SOLAR CELL

MONO CRYSTALLINE SILICON

Electrical Characteristics of Typical Cell at Standard Test Conditions (STC)

STC: 1000W/m², AM 1.5g and cell temp 25°C

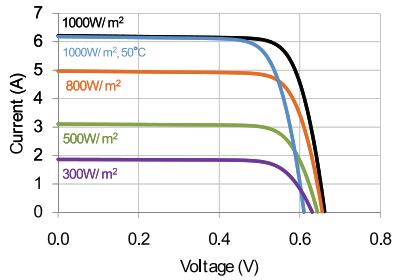
Bin	Pmpp (Wp)	Eff. (%)	Vmpp (V)	Impp (A)	Voc (V)	Isc (A)
G	3.34	21.8	0.574	5.83	0.682	6.24
H	3.38	22.1	0.577	5.87	0.684	6.26
I	3.40	22.3	0.581	5.90	0.686	6.27
J	3.42	22.5	0.582	5.93	0.687	6.28

All Electrical Characteristics parameters are nominal
Unlaminated Cell Temperature Coefficients
Voltage: -1.8 mV / °C Power: -0.32% / °C

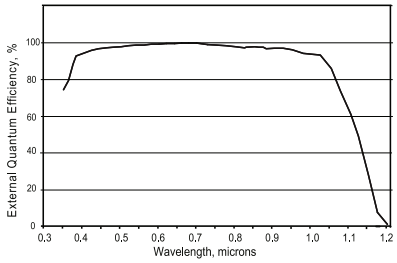
Positive Electrical Ground

Modules and systems produced using these cells must be configured as "positive ground systems".

TYPICAL I-V CURVE



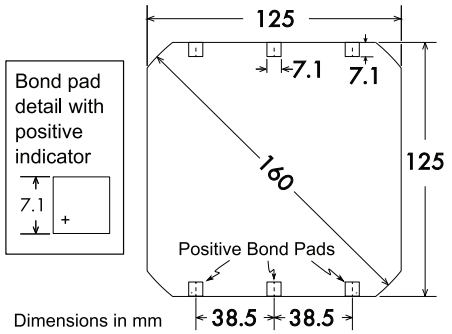
SPECTRAL RESPONSE



Physical Characteristics

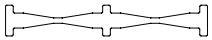
Construction:	All back contact
Dimensions:	125mm x 125mm (nominal)
Thickness:	165µm ± 40µm
Diameter:	160mm (nominal)

Cell and Bond Pad Dimensions



Bond pad area dimensions are 7.1mm x 7.1mm
Positive pole bond pad side has "+" indicator on leftmost and rightmost bond pads.

Interconnect Tab and Process Recommendations



Tin plated copper interconnect. Compatible with lead free process.

Packaging

Cells are packed in boxes of 1,200 each; grouped in shrink-wrapped stacks of 150 with interleaving. Twelve boxes are packed in a water-resistant "Master Carton" containing 14,400 cells suitable for air transport.

Interconnect tabs are packaged in boxes of 1,200 each.

About SunPower

SunPower designs, manufactures, and delivers high-performance solar electric technology worldwide. Our high-efficiency solar cells generate up to 50 percent more power than conventional solar cells. Our high-performance solar panels, roof tiles, and trackers deliver significantly more energy than competing systems.

sunpowercorp.com

Document #001-66352 Rev** A4_en