Geographically-aware scaling for real-time persistent websocket applications.

Master's Project in Software Engineering

Lukasz Harezlak
lukasz.harezlak@gmail.com

Summer 2015, 23 pages

Supervisor: Tijs van der Storm
Host organisation: Instamrkt, https://instamrkt.com

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Master Software Engineering
http://www.software-engineering-amsterdam.nl

Contents

Abstract

This section summarises the content of the thesis for potential readers who do not have time to read it whole, or for those undecided whether to read it at all. Sum up the following aspects:

- relevance and motivation for the research
- research question(s) and a brief description of the research method
- results, contributions and conclusions

Kent Beck [JBB+93] proposes to have four sentences in a good abstract:

1. The first states the problem.
2. The second states why the problem is a problem.
3. The third is the startling sentence.
4. The fourth states the implication of the startling sentence.

Chapter 1

Introduction

Software scalability. Cloud scalability. Why is it important Importance of measuring in the clouds and testing optimal architectures.

1.1 Initial Study

What we found out researching.

1.2 Problem Statement

State the general problem that you are trying to solve with this research.

1.2.1 Research Questions

Precise research questions here.

1.2.2 Solution Outline

How our solution works in short.

1.2.3 Research Method

Why we went for a controlled experiment.

1.3 Contributions

Case study on AWS. Architecture decomposition analysis. Deliverable piece of code - metrics framework and auto-scaling scripts.

1.4 Related Work

The big paper on measuring scalability in the clouds. The big paper on deploying different scalability structures. Small mentions of the others.

1.5 Outline

Here we outline the structure of the thesis. A short paragraph on what happens in each chapter.

Chapter 2

Background

2.1 Scalability of Software Systems

Scalability seems to be a notion that everyone intuitively understands, but has difficulties when it comes to clear explanations. In my literature study I came across a few similar definitions, which might help with that, of which 3 can be found below:

> [WG06] Scalability is a measure of an application systems ability to, without modification, cost-effectively provide increased throughput, reduced response time and/or support more users when hardware resources are added.

2.2 Cloud Scalability

General.

2.3 Data-layer Scalability

2.4 Websocket Scalability

2.5 Geographical Distribution

2.6 Measuring Scalability

What we have from literature study. ONLY THE GENERAL STUFF - WE HAVE A FULL CHAPTER 4 ON THIS.

Chapter 3

Experiment outline

1. Baseline architecture on a local network.
2. Baseline architecture deployed in the cloud.
3. Improved architecture in the cloud.

All of them measured with the same set of metrics with a prepared framework for gathering them. Details below.

3.1 Baseline architecture on a local network

Architecture diagrams. Technical details - local server capabilities. Load testing.

3.2 Baseline architecture deployed in the cloud

3.3 Improved architecture deployed in the cloud

Chapter 4

Scalability Measurements

The previous description 2.6 concerned general stuff. Here we describe the metrics we settled for.

4.1 Selected model

[3] With CRUMs, SPMs etc.

4.2 Baseline, Local network

Lab notes from 11 of May go here:
Basic metrics

- cpu, memory, disk usage (pidstat / CloudWatch)
- network i/o (wireshark, list others analyzed)
- latency, throughput, concurrent connections, messages dropped?

4.2.1 Load Testing

all the analyzed tools, list also in lab notes from may 11

4.3 Baseline, Deployed in the cloud

4.4 Improved, Deployed in the cloud

Chapter 5

Experiment results

Chapter 6

Evaluation

Chapter 7

Conclusion

Chapter 8

Further work

Chapter 9

Everything below that needs to be removed (except for bib)

Chapter 10

Front Matter

The first thing is to connect the class by saying:

```
\documentclass{uvamscse}
```

10.1 Title

Specify the title of the thesis with \title and \subtitle commands:

```
\title{MetaThesis}
\subtitle{A Thesis Template Leading by Example}
```

Any thesis can survive without a \subtitle, but the \title is mandatory.

10.2 Author

Introduce yourself with \author and \authemail:

```
\author{Vadim Zaytsev}
\authemail{vadim@grammarware.net}
```

Again, \authemail is not mandatory. If you need anything fancier, just put it inside \author.

```
\author{Vadim Zaytsev\footnote{Yes, that one.}}
```

The footnote would be printed on the bottom of the title page, and will be referred to by a symbol, not by a number as any footnotes within the main document body.

10.3 Date

By default, the date inserted in your PDF is the day of the build, e.g., "March 25, 2014". If you want it to be formatted differently or be more vague or outright fake, use \date:

```
\date{Spring 2014}
```

The argument is just a string, the format is unrestricted:

```
\date{Tomorrow. Honestly.}
```

10.4 Host

If your hosting organisation is not the UvA, specify it with \host. The logo on the bottom of the title page will still be the UvA one, because this is the organisation guaranteeing your degree.

```
\host{Grammarware, Inc., \url{http://grammarware.github.io}}
```

NB: footnotes will not work, unless you know how to \protect them.

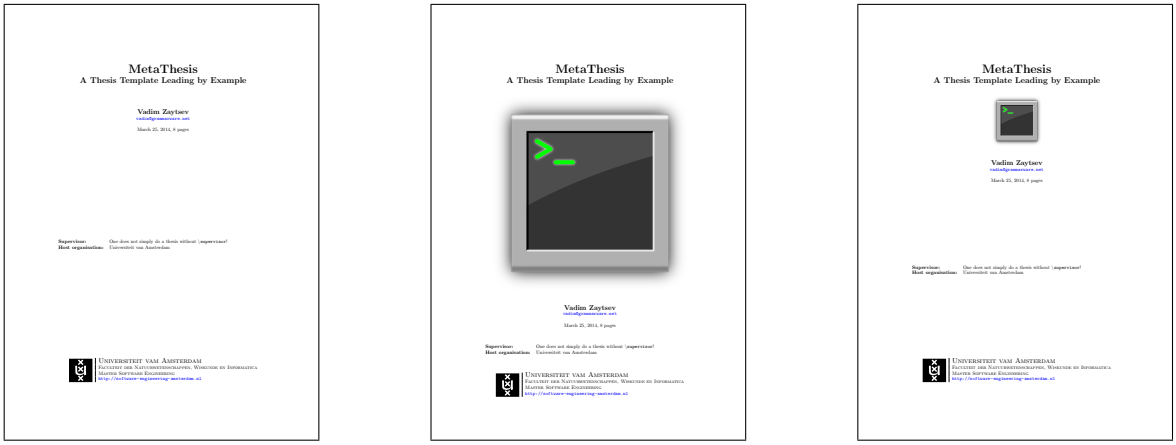

Figure 10.1: A hypothetical thesis title page without a cover picture (on the left), with an overly large one (in the centre) and with a tiny pic (on the right).

10.5 Cover picture

If the first page of your thesis looks too blunt, add a picture to it:

```
\coverpic{figures/terminal.png}
```

You can even specify the picture's width as an optional argument:

```
\coverpic[100pt]{figures/terminal.png}
```

How these three options look, you can see from Figure 10.1.

10.6 Abstract

A thesis is fine without an abstract, if you do not feel like writing it and your supervisor does not feel like enforcing it. If you do want an abstract, make it with the `\abstract` command:

```
\abstract{This is not a thesis.}
```

The abstract is just like any other section of your thesis, so you can use any LaTeX tricks there. If you think that the name "abstract" is too abstract for your abstract, you can still use `\abstract` without being too abstract:

```
\abstract[Confession]{I am a cenosillicaphobiac.}
```

Kent Beck [JBB+93] proposes to have four sentences in a good abstract:

1. The first states the problem.
2. The second states why the problem is a problem.
3. The third is the startling sentence.
4. The fourth states the implication of the startling sentence.

In practice, each of these "sentences" can be longer than an actual sentence, but it is in general a good rule of thumb to condense the summary of your thesis into these four tiny messages. Do not write too much, make it tweetable.

Chapter 11

Core Chapters

The structure of your thesis is up to you and your supervisor. Whatever you do, do not consider the guidelines below as dogmas.

11.1 Classic structure

Problem statement and motivation. You describe in detail what problem the research is addressing, and what is the motivation to address this problem. There is a concise and objective statement of the research questions, hypotheses and goals. It is made clear why these questions and goals are important and relevant to the world outside the university (assuming it exists). You can already split the main research question into subquestions in this chapter. This section also describes an analysis of the problem: where does it occur and how, how often, and what are the consequences? An important part is also to scope the research: what aspects are included and what aspects are deliberately left out, and why?

Research method. Here you describe the methods used to answer the research questions. A good structure of this section often follows the subquestions by providing a method for each. The research method needs a thorough motivation grounded in theory in order to be acceptable. As a part of the method, you can introduce a number of hypotheses — these will be tested by the research, using the methods described here. An important part of this section is validation. How will you evaluate and validate the outcomes of the research?

Background and context. This chapter contains all the information needed to put the thesis into context. It is common to use a revised version of your literature survey for this purpose. It is important to explicitly refer from your text to sources you have used, they will be listed in your bibliography. For example, you can write "A small number of programming languages account for most language use [MR13]", where the following entry would be included in your bibliography:

> [MR13] Leo A. Meyerovich and Ariel S. Rabkin. Empirical Analysis of Programming Language Adoption. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA, pages 1–18. ACM, 2013. doi:10.1145/2509136.2509515.

Have a look at § 12 to learn more about citation.

Research. This chapter reports on the execution of the research method as described in an earlier chapter. If the research has been divided into phases, they are introduced, reported on and concluded individually. If needed, this chapter could be split up to balance out the sizes of all chapters.

Results. This chapter presents and clarifies the results obtained during the research. The focus should be on the factual results, not the interpretation or discussion. Tables and graphics should be used to increase the clarity of the results where applicable.

Analysis and conclusions. This chapter contains the analysis and interpretation of the results. The research questions are answered as best as possible with the results that were obtained. The analysis also discussed parts of the questions that were left unanswered.

An important topic is the validity of the results. What methods of validation were used? Could the results be generalised to other cases? What threats to validity can be identified? There is room here to discuss the results of related scientific literature here as well. How do the results obtained here relate to other work, and what consequences are there? Did your approach work better or worse? Did you learn anything new compared to the already existing body of knowledge? Finally, what could you say in hindsight on the research approach by followed? What could have done better? What lessons have been learned? What could other researchers use from your experience? A separate section should be devoted to "future work", i.e., possible extension points of your work that you have identified. Even other researchers should be able to use those as a starting point.

11.2 Reporting on replications

Here are the guidelines to report on replicated studies [Car10]:

Information about the original study

Research question(s) that were the basis for the design

Participants, their number and any other relevant characteristics

Design as a graphical or textual description of the experimental design

Artefacts, the description of them and/or links to the artefacts used

Context variables as any important details that affected the design of the study or interpretation of the results

Summary of the results in a brief overview of the major findings

Information about the replication

Motivation for conducting the replication as a description of why the replication was conducted: to validate the results, to broaden the results by changing the participant pool or the artifacts.

Level of interaction with original experimenters. The level of interaction between the original experimenters and the replicators should be reported. This interaction could range from none (i.e. simply read the paper) to them being the same people. There is quite a lot of discussion of the level of interaction allowed for the replication to be "successful", but this level should be reported even without addressing the controversy.

Changes to the original experiment. Any changes made to the design, participants, artifacts, procedures, data collected and/or analysis techniques should be discussed along with the motivation for the change.

Comparison of results to original

Consistent results, when replication results supported results from the original study, and

Differences in results, when results from the replication did not coincide with the results from the original study. Authors should also discuss how changes made to the experimental design (see above) may have caused these differences.

Drawing conclusions across studies

NB: this section contains portions of text repeated directly from Carver [Car10] and only slightly massaged. Do not do this for your thesis, write your own thoughts down.

11.3 LaTeX details

11.3.1 Environments

A LaTeX environment is something with opening and closing tags, which look like `\begin{name}` and `\end{name}`. Some useful environments to know:

`itemize`	bullet lists
`enumerate`	numbered lists
`description`	definition lists
`center`	centered line elements
`flushright`	right aligned lines
`flushleft`	left aligned lines
`tabular`	table
`longtable`	multi-page table (needs the `longtable` package)
`sideways`	rotates some text
`quote`	block quote
`verbatim`	unformatted text
`minipage`	compound box with elements inside
`boxedminipage`	compound box with elements inside and a border around it
`table`	floating table (needs to have `tabular` nested inside)
`figure`	floating figure
`sourcecode`	floating listing
`equation`	mathematical equation
`lstlisting`	pretty-printed syntax highligted listing
`multline`	mathematical equation spanning over multiple lines
`eqnarray`	system of mathematical equations
`gather`	bundled mathematical equations
`align`	bundled and aligned mathematical equations
`array`	matrix
`CD`	commutative diagrams

11.4 Listings

```
1  define(Ps1,G1,G2)
2   ⇐
3      usedNs(G1,Uses),
4      ps2n(Ps1,N),
5      require(
6        member(N,Uses),
7        'Nonterminal ~q must not be fresh.',
8        [N]),
9      new(Ps1,N,G1,G2),
10     !.
```

Listing 11.1: Code in Prolog

```
module Syntax

imports Numbers
imports basic/Whitespace

exports
  sorts
    Program Function Expr Ops Name Newline

  context−free syntax
    Function+                          → Program
    Name Name+ "=" Expr Newline+       → Function
    Expr Ops Expr                      → Expr   {left ,prefer,cons(binary)}
    Name Expr+                         → Expr   {avoid,cons(apply)}
    "if" Expr "then" Expr "else" Expr  → Expr   {cons(ifThenElse)}
    "(" Expr ")"                       → Expr   {bracket}
    Name                               → Expr   {cons(argument)}
    Int                                → Expr   {cons(literal)}
    "−"                                → Ops    {cons(minus)}
    "+"                                → Ops    {cons(plus)}
    "=="                               → Ops    {cons(equal)}
```

Listing 11.2: Code in SDF

```
1  import types.∗;
2  import org.antlr.runtime.∗;
3
4  public class TestEvaluator
5      public static void main(String[] args) throws Exception {
6
7          // Parse file to program
8          ANTLRFileStream input = new ANTLRFileStream(args[0]);
9          FLLexer lexer = new FLLexer(input);
10         CommonTokenStream tokens = new CommonTokenStream(lexer);
11         FLParser parser = new FLParser(tokens);
12         Program program = parser.program();
13
14         // Parse sample expression
15         input = new ANTLRFileStream(args[1]);
16         lexer = new FLLexer(input);
17         tokens = new CommonTokenStream(lexer);
18         parser = new FLParser(tokens);
19         Expr expr = parser.expr();
20
21         // Evaluate program
22         Evaluator eval = new Evaluator(program);
23         int expected = Integer.parseInt(args [2]);
```

Listing 11.3: Code in Java

```
#!/usr/local/bin/python
# wiki: BGF
import os
import sys
import slpsns
import elementtree.ElementTree as ET

# root::nonterminal* production*
class Grammar:
  def __init__(self):
    self.roots = []
    self.prods = []
  def parse(self,fname):
    self.roots = []
    self.prods = []
    self.xml = ET.parse(fname)
    for e in self.xml.findall('root'):
      self.roots.append(e.text)
    for e in self.xml.findall(slpsns.bgf_('production')):
      prod = Production()
      prod.parse(e)
      self.prods.append(prod)
```

Listing 11.4: Code in Python

Chapter 12

Literature

BibTeX is a JSON-like format for bibliographic entries. Encode each source once as a BibTeX entry, give it a name and refer to it from any place in your thesis. The bibliography at the end of the thesis will be compiled automatically from those entries that are referenced at least once, it will also be automatically sorted and fancyfied (URLs, DOIs, etc).

DOI is a digital object identifier, it is uniquely and immutably assigned to any paper published in a well-established journal or conference proceedings and can be used to refer to it. When used in a browser, it resolves to a publisher's website where paper can be obtained. Including DOIs in citations is considered good practice and lets the readers of your thesis get to the text of the paper in one click. Books do not have DOIs, only ISBNs; some workshop proceedings and most unofficial publications do not have DOIs. If you want to get a DOI assigned to your work such as a piece of code, upload it to FigShare.

Keys in key-value pairs within each BibTeX entry are never quoted, values usually are, but can also be included within curly brackets or left as is, which works fine for numbers (e.g., years). If you want to preserve the value from any adjustments (e.g., no recapitalisation in titles), use curlies *and* quotes. Separate authors and editors by "and", which will automatically be mapped to commas or left as "and"s as necessary.

12.1 Books

[GJ08] is just as good as the Dragon Book, but newer and has an awesome extended bibliography available for free.

```
@book{GruneJacobs,
  author    = "D. Grune and C. J. H. Jacobs",
  title     = "{Parsing Techniques: A Practical Guide}",
  series    = "Monographs in Computer Science",
  edition   = 2,
  publisher = "Springer",
  url       = "http://www.cs.vu.nl/~dick/PT2Ed.html",
  year      = 2008,
}
```

12.2 Journal papers

Not all TOSEM papers are hard to read [KLV05].

```
@article{GrammarwareAgenda,
  author      = "Paul Klint and Ralf L{\"a}mmel and Chris Verhoef",
  title       = "{Toward an Engineering Discipline for Grammarware}",
  journal     = "ACM Transactions on Software Engineering Methodology (TOSEM)",
  volume      = 14,
  number      = 3,
  year        = 2005,
  pages       = "331--380",
}
```

12.3 Conference papers

There is no limit to how many grammars can be used in one paper, but the current record stands at 569 [Zay13].

```
@inproceedings{Micropatterns2013,
  author = "Vadim Zaytsev",
  title = "{Micropatterns in Grammars}",
  booktitle = "{Proceedings of the Sixth International Conference on Software Language Engineering
               (SLE 2013)}",
  year = 2013,
  editor = "Martin Erwig and Richard F. Paige and Eric Van Wyk",
  volume = "8225",
  series = "LNCS",
  pages = "117--136",
  address = "Switzerland",
  month = oct,
  publisher = "Springer International Publishing",
  doi = "10.1007/978-3-319-02654-1_7",
}
```

12.4 Theses

The seventh PhD student of Paul Klint was Jan Rekers [Rek92].

```
@phdthesis{Rekers92,
 author   = "J. Rekers",
 title    = "{Parser Generation for Interactive Environments}",
 school   = "University of Amsterdam",
 year     = 1992,
 url      = "http://homepages.cwi.nl/~paulk/dissertations/Rekers.pdf",
}
```

There is also `mastersthesis` type with exactly the same structure for referring to Master's theses.

12.5 Technical reports

The original seminal work introducing two-level grammars was never published in any book or conference, but there is a technical report explaining it [vW65]. SMC, or *Stichting Matematisch Centrum*, was the old name of CWI fifty years ago.

```
@techreport{Wijngaarden65,
        author      = "Adriaan van Wijngaarden",
        title       = "{Orthogonal Design and Description of a Formal Language}",
        month       = oct,
        year        = 1965,
        institution = "SMC",
        type        = "{MR 76}",
        url         = "http://www.fh-jena.de/~kleine/history/languages/VanWijngaarden-MR76.pdf",
}
```

12.6 Wikipedia

You do not refer to Wikipedia from academic writing, it works the other way around.

12.7 Anything else

You can refer to pretty much anything (websites, blog posts, software) through `misc` type of entry [Par08]:

```
@misc{ANTLR,
 author       = "Terence Parr",
 title        = "{ANTLR---ANother Tool for Language Recognition}",
 howpublished = "Software",
 url          = "http://antlr.org",
 year         = "2008"
}
```

Bibliography

[Car10] Jeffrey C. Carver. Towards Reporting Guidelines for Experimental Replications: A Proposal. In *Proceedings of the International Workshop on Replication in Empirical Software Engineering Research*, RESER, May 2010.

[GJ08] Dick Grune and C. J. H. Jacobs. *Parsing Techniques: A Practical Guide*. Monographs in Computer Science. Springer, 2 edition, 2008. URL: http://www.cs.vu.nl/~dick/PT2Ed.html.

[JBB+93] Ralph E. Johnson, Kent Beck, Grady Booch, William R. Cook, Richard P. Gabriel, and Rebecca Wirfs-Brock. How to Get a Paper Accepted at OOPSLA. In Timlynn Babitsky and Jim Salmons, editors, *Proceedings of the Eighth Annual Conference on Object-Oriented Programming Systems, Languages and Applications*, OOPSLA, pages 429–436. ACM, 1993.

[KLV05] Paul Klint, Ralf Lämmel, and Chris Verhoef. Toward an Engineering Discipline for Grammarware. *ACM Transactions on Software Engineering Methodology (TOSEM)*, 14(3):331–380, 2005.

[MR13] Leo A. Meyerovich and Ariel S. Rabkin. Empirical Analysis of Programming Language Adoption. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA, pages 1–18. ACM, 2013. doi:10.1145/2509136.2509515.

[Par08] Terence Parr. ANTLR—ANother Tool for Language Recognition. Toolkit website, 2008. URL: http://antlr.org.

[Rek92] J. Rekers. *Parser Generation for Interactive Environments*. PhD thesis, University of Amsterdam, 1992. URL: http://homepages.cwi.nl/~paulk/dissertations/Rekers.pdf.

[vW65] Adriaan van Wijngaarden. Orthogonal Design and Description of a Formal Language. MR 76, SMC, October 1965. URL: http://www.fh-jena.de/~kleine/history/languages/VanWijngaarden-MR76.pdf.

[WG06] Charles Weinstock and John Goodenough. On system scalability. Technical Report CMU/SEI-2006-TN-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2006. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7887.

[Zay13] Vadim Zaytsev. Micropatterns in Grammars. In Martin Erwig, Richard F. Paige, and Eric Van Wyk, editors, *Proceedings of the Sixth International Conference on Software Language Engineering (SLE 2013)*, volume 8225 of *LNCS*, pages 117–136, Switzerland, October 2013. Springer International Publishing. doi:10.1007/978-3-319-02654-1_7.