TagTwitter: Micro-Blogging Dissemination with Tags

project supervisors
Antonio Carzaniga and Michele Papalini
December 2012

Background

The notion of information-centric networking is based on the idea that communication is often driven by content rather than on the location of that content. In other words, an information-centric network allows users to address the content they want to access, as opposed to the host where that content might be located. The information-centric network architectures proposed so far, and specifically the Content-Centric Networking (CCN) system developed at the Palo Alto Research Center (PARC), allow users to retrieve data on-demand, using requests based on the name of the data as is typically done with Web content. However, a lot of traffic on the Internet is now based on a different form of communication called “publish/subscribe”. In this case, senders simply publish messages that the network then delivers to all the subscribers that are interested in those messages. Examples of application that use this kind of communication are Twitter, the Facebook notification system, RSS feeds and many others. Here at USI, we have been working on an extension of the CCN system that implements publish/subscribe communication natively.

Project Description

The goal of this project is to implement a micro-blogging system (like Twitter) in which readers could follow chosen topics of discussions as well as chosen authors. In other words, each message (or “tweet”) would be disseminated to the followers of the author but also to those interested in the content of the message. The idea is to describe content through a set of “tags,” and then to implement such a tag-based system over our new content-based communication network. The concrete expected outcomes of this project are (1) a client application perhaps running on smartphones, and (2) a study of the use of this application on a content-based network.

Organization and Requirements

The project will be carried out in cooperation with and under the supervision of Antonio Carzaniga and Michele Papalini, and will last the full 8 week period of the UROP program. Candidate students must have an interest in networking. At a minimum, they must have successfully completed the Computer Networking class. Further, applicants should be comfortable with programming in Java but possibly also C. Most importantly, applicants must enjoy learning about and using new technologies, in particular new network architectures and protocols.
UROP 2013

A framework for autonomic control on the EC² Amazon Cloud

advisor: Mauro Pezzè

Cloud applications must satisfy their service level agreement while reducing the operational costs. To meet these contradicting requirements researcher develop self-adaptive autonomic controllers.

This project aims to develop a framework for evaluating self-adaptive strategies on the cloud. The framework will be deployed on the EC2 Amazon platform as part of the AWS in Education Research grant that Amazon recently granted to the Star (Software testing and analysis research) laboratory. The student will learn cloud development technologies, advanced techniques for cloud control and different approaches to cloud infrastructures.

We require good knowledge of the Java programming languages and Web development technologies and strong interest in software design and development and enthusiasm in learning new technologies.

If you are interested in knowing more, please contact Mauro Pezzè (mauro.pezze@usi.ch)
Introduction

In modern seismology (the study of earthquakes), scientists rely heavily on computational resources in their attempt to model earthquakes and utilize the energy released by an earthquake to image the structure of the earth. Already, seismologists can simulate earthquakes on a global scale with surprising accuracy given how little is known about the detailed structure of the earth. Improving this earth structure model, in a process called tomography, is what is often called an inverse problem. In other words, given earthquake data recorded at stations around Switzerland and the globe, we try to work backwards to figure out what kind of earth structure would have created the data we have from a given earthquake.

Full-Waveform Adjoint-tomography

There are many techniques and variations within techniques to improve the resulting earth-model when doing tomography. One specific approach called adjoint-tomography utilizes full-3D earthquake simulations as the comparison between real recorded data and our best-approximation to earthquake. At a basic level, we have software capable of simulating an earthquake on a regional or global scale and existing earthquake data recorded at various stations in our region of interest (including information about the earthquake location and magnitude). For each earthquake (Map phase):

1. Simulate the earthquake
2. Simulate the adjoint-source using the error between simulation and the real data
3. Compare “forward” and “adjoint” simulations to create source contribution to earth-model update.

After we simulate all the earthquakes in our database, we combine the source contributions to build the first-order update to our earth-model (Reduce phase). We repeat this update using various nonlinear-optimization algorithms, yielding a better earth-model after each step. The current software can only simulate one source at a time, requiring the user or script to launch multiple sources manually.

MapReduce for Seismic-Imaging

Many groups doing these inversions do a significant amount of this work by hand, using bash, or at most, using python scripts to help automate the work. However the overall process seems an ideal fit for the MapReduce framework. MapReduce, originally pioneered by Google, is now a robust framework with many available implementations including Hadoop, an open source java toolkit for managing large batch-processing tasks on clusters.

1 Project Goals

The goal of this project is to utilize Hadoop (or equivalent popular MapReduce implementation) to simplify imaging projects specific to seismology. Additionally, the project will use the existing simulation software SPECFEM3D, which is currently developed in part by the PhD-student advisor of this project. Specifically the student would:

1. Get Hadoop running on a supercomputing cluster such as the ones located at the Swiss National Supercomputing Centre (CSCS).
2. Adapt Hadoop to work with SPECFEM3D.
4. Use Hadoop to combine the source contributions into a single gradient update.
1.1 Student Requirements

Ideally the student will be in the computer or computational science program at USI Lugano. Some Linux experience is a must, along with some amount of Linux software development experience, which will be required to compile and run tests using the SPECFEM3D simulation [1] software (written in Fortran). Some Java experience would also be useful when using the Hadoop framework, especially when adapting it to work on the CSCS systems.

1.2 Student Achievements

At the end of this project, the student will have experience with

- Interfacing with a real, production-quality scientific computing software package,
- Running code on a world-class supercomputing cluster,
- Linux, Java, and Fortran development.

Above all this is a very practical project, with a much desired goal to leverage an existing and very successful batch-processing framework to automate tasks for seismology.

References

Student Project Proposal: A Testing Framework for Scientific Codes (SPECFEM3D)
Max Rietmann, Prof. Olaf Schenk
Advanced Computing Laboratory, Institute of Computational Science, Faculty of Informatics

Introduction

In many cases, scientific software (or “codes” as commonly called), are not developed using modern techniques or tools. Many of the scientists writing these codes are self-trained in software development, and their tools do not easily lend themselves to modern methodologies. In the field of computational seismology, the code SPECFEM3D, which simulates earthquakes on global or local scales is a very good example of a modern scientific code. Written in Fortran90, it is actually very well structured and generally easy to read, understand, and develop. The code itself is packaged using the well known unix autotools, requiring only a ./configure and make to setup and compile. In the last few years, this code has been growing in complexity, as additional physics, algorithms, and numerical routines are added to the code. Beyond physical and numerical changes, the code has also added GPU support via CUDA [1].

Testing

As the code grows, and new developers extend and add new routines, the code might fail to compile, fail to finish pre-processing stages, or possibly fail to run correctly. SPECFEM3D specifically lacks a testing suite, which can be run automatically on new versions committed to the version control system (currently SVN). Developers adding code will test their own additions as they are added, but are generally unable to test functionality that is not covered in their specific application domain. An automatic testing framework would be useful for the following reasons

• Developers will catch errors introduced by their changes quickly
• Developers can follow existing test recipes to build their own tests, guaranteeing that the functionality they are responsible for will continue to work in the future.
• Scientific results produced by the software are more reliable because the software has been better tested across more of its functionality.

Project Goals

The goal of this project is to build a testing framework for the SPECFEM3D package. The initial goals would be threefold:

1. Develop methodology and code to test SPECFEM3D
2. Automate testing, possibly using resources at the Swiss National Supercomputing Centre in Lugano (CSCS).
3. Make it easy for other developers to add tests for their existing routines, and for new developers to test their changes and add new tests.

Ideally this work would be extendable for other codes as well.

Student Requirements

A student working on this project will have to install, compile, run, and eventually test the SPECFEM3D software package. This will require some experience using and developing software using Linux and standard GNU build systems such as Make. Ideally the student will have background or coursework that relates to software testing. Several developers have been using Python to automate using SPECFEM3D, and Python may be a good choice for building a testing framework.
**Student Achievements**

At the end of the project, the student will have experience with

- production-quality scientific software,
- running and testing software on a world-class supercomputer,
- Linux, Fortran, and C software development.

This project fills a real need within a group of scientific software developers and would give the student a wide-margin of creativity to implement a robust and use-able software framework.

**References**

ASQ: HTML5 Presentations with user feedback and peer assessment

UROP 2013 Project Proposal

Professor: Cesare Pautasso  
Supervisor: Vasileios Trigianos

ASQ\[1\] is a web application for creating HTML5 presentations. It currently supports multiple viewers and allows the lecturer to receive feedback from the audience by aggregating the answers to questions embedded in the slides.

The objective is to allow a constructive bi-directional communication between the lecturer and the audience. More specifically, in the context of a classroom, a teacher can: get feedback in real time about the level of comprehension of the presented material by the students; reduce the time for filling and delivering questionnaires; extract useful statistics; monitor presence; allow for peer-assessment of the answers.

The goal of this project is to add functionality to ASQ to support multiple questions per slide, user authentication, various forms of assessment (self assessment\[2\], peer assessment, staff assessment), advanced statistics and multiple question types.

The ideal candidate should have experience in modern web technologies like HTML5, CSS3, javascript, jQuery and mongoDB. Familiarity with the “impress.js”\[3\] javascript library for presentations is mandatory. Additionally, a good understanding of the field of Human Computer Interaction is encouraged.

1. ASQ github repository: https://github.com/jacquesd/ASQ
3. impress.js github repository: https://github.com/bartaz/impress.js
UROP Project Proposal

Concurrent Evaluation of Recurrent Compute-Tasks In Multilevel Methods

Advisor: Prof. Dr. Rolf Krause
Co-Advisor: Patrick Zulian

Abstract
Multilevel methods for the solution of systems of elliptic partial differential equations are well known to be optimal in terms of convergence and complexity. Finding the solution of large systems of equations is nevertheless an expensive task. To speed-up the computation of the solution is necessary the employment of parallel algorithms. These algorithms are usually designed for specific multi-processor architectures. The main objective of this project is to design and implement a parallel multigrid solver for CPU and shared-memory based architectures by means of asynchronous compute-tasks evaluation. The use of asynchronous task evaluations allow schedulers to make better use of hardware resources. A well known and efficient scheduling strategy is work-stealing. Here, asynchronous computations involve concurrency, and concurrency involve several challenges with respect to data consistency and efficiency.

The efforts of the undergraduate would consists of:
- Understanding and analyzing the multigrid method, mostly on its algorithmic and implementation aspects.
- Understanding work-stealing scheduling.
- Researching and understanding existing shared-memory based parallelizations for iterative solvers, and basic linear algebra operations and techniques.
- Identifying which computational tasks of the multigrid method can be parallelized both logically and with respect to the solution memory space.
- Designing, implementing, and documenting the parallel multigrid solver
- Evaluating the implementation in terms of its performance by means of scaling studies.

The project will involve the use of software libraries which enable a user-friendly and almost transparent use of work-stealing scheduling (e.g., XKAAP! by INRIA). Familiarity with C++, OOP and meta-programming is required. This project is an opportunity for an undergraduate to be followed by experts of the field of Multigrid methods, and to contribute to a new exciting parallel approach.
Incremental SAT Solving

Prof. Natasha Sharygina and Dr. Antti Hyvärinen

Propositional satisfiability (SAT) solvers are widely used as a tool for computing combinatorial subproblems emerging from a multitude of applications, such as software verification and AI planning. SAT solvers play also a key role in modern automated reasoning engines such as the solvers for SAT modulo theories (SMT).

One of the recent directions in SAT solver research is based on an observation that in many applications the problem solving consists of several calls to SAT solvers on a sequence of SAT formulas. Typically the formulas in the sequence share a large common part, making them highly interrelated. The goal of this UROP project is to study how the work done in proving a previous formula in the sequence can be maximally recycled in solving the subsequent formulas.

The work, including experimentation and implementation, will be carried out as part of the ongoing development effort of the OpenSMT solver from the Verification group at USI.

We are looking for a motivated student who wants to improve his/her knowledge on software verification and AI planning. This project will give the student an excellent overview of a quickly developing field while being sufficiently approachable. Prior knowledge in SAT based modeling is not required, though is a plus.

The aim of this project is to integrate incremental solving to OpenSMT. The student will be coached while:

1. Getting familiar with the internals of SAT solvers
2. Designing an interface between the solver and applications. For this task preliminary experience with Python and C++ is required.
3. Designing and running a set of experiments.
Specification-based software versioning system integrated with model checker eVolCheck and Eclipse IDE.

Prof. Natasha Sharygina and Grigory Fedyukovich
USI Formal Verification Lab
www.verify.inf.usi.ch

Model Checking[1] is a well-known scientific approach to check correctness of software. It is a fully automatic approach to decide whether a program is safe or not with respect to a given specification. A specification may represent any property over program variables, in any program state. It usually appears in the code as specific commands called “assertions”. There are already existent model checkers for software [2,3,4,5]. The model checkers FunFrog and eVolCheck [5,6], developed at the University of Lugano, have GUI integrated with the Eclipse IDE which makes their use more convenient for a software developer.

During its lifetime any program evolves. A developer can fix a bug, improve the effectiveness using some optimizations, or add new functionality to the existent code. After such changes the developer should be sure, the new code doesn’t break the already existent one, i.e., old specifications still hold. In order to do it efficiently, a technique called Upgrade Checking (and in particular, eVolCheck [6]) has been developed by researchers at USI. eVolCheck verifies each version of a program and reuses some efforts (namely, function summaries) between consequent runs. For example, if just one function is changed in the program, it may be enough to re-check only this function, and do not touch the rest of the program.

In the GUI of eVolCheck integrated with Eclipse IDE, the process is visualized as follows. First, the syntactic diff between the current version and the last one is shown. Depending on the amount of the modified code, the user decides to run the upgrade checking tool. Then the Eclipse IDE internally runs eVolCheck, it configures the tool automatically and keeps the settings in a subsidiary storage, so the user doesn’t need to adjust the environment for every check. After eVolCheck is terminated, the positive/negative result is returned to the user. In the former case, the change impact (namely, the number of re-checked summaries) is drawn - it represents the semantic influence of the change on the whole program. In the latter case, if a property violation is found (i.e., an error is reachable), the Eclipse IDE shows the trace to the error location.

Software versioning systems, integrated to IDE are quite essential and useful for upgrade checking. They might be guided by the result of eVolCheck’s verification. If a current version is proven correct, it might be good justification to create a new instance in the versioning system, which is currently in use. Otherwise, if at least one property of the version is violated, the developer should analyze it, then fix it, and then repeat the check. So the instances in the versioning system will become trustworthy. Now, every new version will be confirmed safe with respect to the current specification, and its specification will play a role of identifier of the version. We propose to student to implement such technology on the top of the existent tools.

We are sure the proposed technology can be easily implemented because its main ingredients already exist: the Eclipse IDE with available source code and APIs, eVolCheck, and eVolCheck plugin to Eclipse. The student should investigate how to use software versioning systems, and how these systems are being handled by Eclipse. We believe such an integration will make easier to apply formal verification in
most software development projects, which are currently unfortunately still not able to use this successful technology. And this will ultimately increase the quality of software being developed.

There are two main benefits for a student who is going to implement the proposed technology. First, the student will become familiar with formal verification and state-of-the-art verification systems. It will give the young researcher a taste of what kind of research is being done in the modern Computer Science. No theoretical background in Formal Verification is required. Second, the student will earn an important experience in software development. Eclipse IDE is known in developers community, and experience in working with its source code will indicate the student is familiar with modern trends in software development. For this, initial experience in Java programming language is required.

References:
An invariant generation for the BOOGIE verification system

Prof. Natasha Sharygina and Francesco Alberti

The goal of this project is to integrate an invariant generation engine, SAFARI [3, 1], with a program verifier, BOOGIE [6, 2].

BOOGIE is a program verifier developed at Microsoft research and integrated/exploited in several Microsoft projects (e.g., [8, 11, 7, 10, 9]). It can be thought as a compiler producing, instead of executable code, verification conditions, i.e., first order formulae which validity infers the validity of the input code. Verification-condition generation offers a high degree of precision, compared with other static analysis techniques. However, for this approach, loop invariants are needed for a successful verification of the code. The user in the BOOGIE input file may specify loop invariants. Some loop invariants are key ideas in the verification of a program and thus it is useful to incorporate them in the source. On the other hand, others loop invariants - in particular, those that state which heap locations are left untouched by the loop body - are boring or unintuitive to the programmer. Finally, for some programs, it might be too difficult to infer the right loop invariant (e.g., invariants with quantifiers over unbounded data structures).

SAFARI is an SMT-based infinite-state model checker developed at USI in the Formal Verification and Security Group. One of the main features of SAFARI is the ability of generating quantified invariants. SAFARI takes in input a symbolic description of the source code and produces invariants for the input code.

A preliminary feasibility study of this project has been presented at the 2nd International Workshop on Intermediate Verification Languages [4].

We are looking for a motivated student who wants to improve his/her knowledge on software verification. Prior knowledge in verification is not required, though is a plus. In details, the goal of this project is to integrate SAFARI and BOOGIE. The student will be coached while:

1. Identifying the appropriate point in the BOOGIE internal pipeline for a suited and efficient integration with SAFARI.

2. Designing the new module for the integration. For this point preliminary experience with Python or C# is required.

3. Designing and running a set of experiments.

This project will firstly give the student the occasion to improve his/her verification skills: automatic invariant generation is a long standing scientific challenge, and plays a central role in program analysis, testing, and verification. The student will also have the occasion to learn how state-of-the-art tools used in program verification (like BOOGIE and SAFARI) work, and strengthen his/her skills in software development.


References


EXPRESSO: OPINION EXPRESSION ON NETWORKED PUBLIC DISPLAYS

UROP Project Proposal

As part of the international research project "PD-Net", we are deploying a number of large public screens on the USI campus. Contrary to existing public display systems, those screens will form part of an open network of displays that can be customized by both their owners and their viewers.

As part of the UROP project, a student will join an international research team from four European countries – Germany, Great Britain, Portugal, and Switzerland – in order to design, develop, and deploy a number of interactive applications that will allow students/users to express their opinion about the application content. The focus of the project will lie both on application and toolkit design, as well as in participating in actual deployments.

In detail, the various tasks are

- Create tools and interfaces that allow user expression about the content as well as sharing on existing social networks, e.g., Facebook.
- Assemble, test, and deploy the set of applications on a public display infrastructure in Lugano, and possibly at partner sites across Europe (additional travel funds available)
- Assist project members during public trials (data collection, system maintenance)

Applicants should have strong skills in Web design and HTML5/JavaScript coding. Knowledge of Facebook’s APIs is a plus.

Ideally, the applicant has already worked on interactive public display systems, e.g., in the corresponding Bachelor project offered in Spring 2013.

Professor: Marc Langheinrich
Supervisor: Nemanja Memarovic, Ivan Elhart
Given today’s large software systems, consisting of tens or hundreds of thousands of classes, wouldn’t it be nice to be able to automatically distinguish between their essential and non-essential parts? More specifically, wouldn’t it be nice to be able to quantify the amount of algorithmically essential code and the amount of code that primarily serves design? And wouldn’t it be nice to have a tool that automatically locates the essential code?

The Sape research group recently introduced a novel software design metric, *algorithmic essence*, that does just that. Our metric quantifies the amount of (behavioral and structural) indirection in a software system. We have developed a static program analysis to compute essence, and we have implemented a tool that measures essence for arbitrary Java programs. The tool is written in Java. It reads in a JAR file (the Java program) as input, and it returns information about the essence of that program.

In this UROP project, you will develop a dynamic web site that allows developers to upload their code to get a report on the essence. The site, called “Essentializer”, is to be based on *node.js* on the server side. The node.js application will invoke our Java-based tool to compute essence.

The Essentializer will provide a stable, clean, user friendly user interface to allow developers to upload applications, to compare their application’s essence to the essence of other applications (or to other versions of the same application), or to reference applications that are known to be designed well.

Optionally, the Essentializer could also help developers to investigate the essence of various parts of an application. For example, it could list all packages of the application, sorted by their essence. Or it could produce essence-driven visualizations of the application.

The student for this project will need very good programming skills, experience in JavaScript, and ideally experience in developing web applications on top of *node.js*.

For more on the concept of “essence”, and for the two relevant research papers [1,2], check out our web page: [http://sape.inf.usi.ch/essence](http://sape.inf.usi.ch/essence)

References
UROP 2013 Project - InformaWeb: A Web-Based Classroom Clicker

Advisor: Matthias Hauswirth

Informa is an extensible group response system that can quickly gather deep feedback from all students in a classroom.

**What is a classroom clicker?** In traditional group response systems, an instructor projects a multiple-choice question on the classroom projector, and students use clickers, special purpose remote controls, to submit their answers by pressing one of the buttons on their clicker. The instructor can immediately see the responses aggregated in the form of a histogram and can adopt her lecture to the level of understanding of the students.

**How is Informa different?** Informa is a software implementation of a clicker system. It is implemented as two Java applications that communicate over Java RMI. The Informa instructor application runs on the instructor’s computer. It allows the instructor to control the session, and it presents visualizations on the classroom projector. The Informa student application runs on the computers of all students. It connects to the instructor application and allows the students to solve problems and evaluate solutions.

The main idea behind Informa is that a software-based clicker is able to support much richer interactions than traditional hardware clickers. In Informa, multiple-choice questions are just one of an extensible number of problem types. For example, Informa provides a plugin supporting so-called text-highlighting problems, where students are presented with a question and a body of text, and then have to highlight all the relevant sections in the text. This type of problem is richer than a multiple-choice question, because students have to "construct" the solution (an arbitrary set of highlights) themselves, instead of picking among a set of solutions predetermined by the instructor.

In this UROP project, you will develop InformaWeb, a web-based version of Informa. The site is to be based on node.js on the server side, so the entire application will be implemented in JavaScript. You will use the original Java application as an inspiration, but you will develop the InformaWeb application from scratch. The goal is to support the use of InformaWeb in a classroom without the need to install any application, just by going to the InformaWeb site.

The student for this project will need very good programming skills, experience in JavaScript, and ideally experience in developing web applications on top of node.js.

For more on the ideas behind Informa, and for related research papers [1,2], check out our web page: [http://sape.inf.usi.ch/informa](http://sape.inf.usi.ch/informa)

References
Virtual 3D Table Tennis

Proposal for a UROP 2013 Project

(Supervisor: Prof. Kai Hormann)

Imagine there is a table tennis athlete and neither his trainer nor his friends are around to help him practising. With our virtual 3D table tennis system, he does not have to worry about this situation anymore, because it allows him to still continue improving his skills! Our system combines Microsoft’s Kinect and a 3D projector to create a virtual reality where the computer becomes the table tennis partner. With this system the user can practise shots on pre-defined or random ball trajectories, just as if standing at a real table. Your task is to revise and improve the current version.

In detail, your tasks are

• Improve the precision by using Microsoft’s libraries instead of the ones we are using currently.
• Add new features and improve the user interface.
• Improve the robustness of the system.
• Present it at the Sportech event 2014 in Tenero.

In order to qualify for the project you should attend or have attended the Computer Graphics course.

If you are interested, then please contact Prof. Kai Hormann (kai.hormann@usi.ch).
The Voronoi diagram is one of the fundamental geometric data-structure that encodes proximity information of a given set of geometric objects called sites (such as points, segments, curves, etc). Voronoi diagrams find application in many real life application areas, such as computer graphics, computer vision, geographic information systems, computer-aided design (CAD), computational science, etc. There are many variants of Voronoi diagram such as the nearest neighbor, the higher order, and the farthest neighbor.

In this project, we focus on the farthest line-segment hull and the farthest line-segment Voronoi diagram. These are analogues of the convex hull and the farthest Voronoi diagram of points, respectively. It is known that the convex hull can be used as a starting point to build the farthest Voronoi diagram. Our aim in this project is to adapt and implement methods from points to line segments, in order to build the farthest line-segment Voronoi diagram from the farthest line-segment hull. We suggest implementation in the framework of the Computational Geometry Algorithms Library (CGAL, http://www.cgal.org).

**Benefit**

You will be able to acquire profound knowledge about fundamental geometric data-structures, convex-hull and Voronoi diagrams, which are useful in many real life applications. You will acquire experience in programming with the Computational Geometry Algorithms Library (CGAL).

**Contact**

Prof. Evanthia Papadopoulou, SI-209, evanthia.papadopoulou@usi.ch

Dr. Panagiotis Cheilaris, SI-205, panagiotis.cheilaris@usi.ch

Sandeep Kumar Dey, Open Space (2nd Floor), deys@usi.ch

**References**

