UROP 2011 Project - Are profilers fundamentally broken?

A profiler measures where in the program the system spends most of its time. The goal of profiling is to find those parts of the program that, when improved, lead to a big overall performance gain. There are two entirely different ways to profile a program, that is, to determine where (in which method) your program spends its time: (1) instrumentation and (2) sampling.

With **instrumentation**, you inject (manually or automatically) some extra code at the beginning and end of each method to measure the time spent in the method.

```java
void m() {
    long start = System.nanoTime(); // get current time in nanoseconds
    // ... (original method body)
    long end = System.nanoTime(); // get current time in nanoseconds
    Profiler.incrementTimeSpentInMethod("m", end-start);
}
```

If you instrument all methods, this approach provides you with the exact measured time the program spends in each method.

With **sampling**, you don’t add any code to the methods. Instead, from time to time, while the program is running, you check which method is currently executing. This approach provides you with a statistical sample of the distribution of execution time over the different methods of your program.

The advantage of sampling is its adjustably low overhead. There is no need to change (instrument) the program, and only a small amount of extra code is executed (periodically checking which method is running; something which can be done very rarely, say, only a few times per second). The disadvantage of sampling is that you only look from time to time, thus you may miss something, which means you may produce an inaccurate profile. This second argument has been used in a report by Sun to motivate the introduction of the JFluid profiling approach [1] which is now widely deployed in their NetBeans Profiler. The same argument is also used in the motivation for the open-source JiP profiler.

We believe that the *instrumentation approach to profiling is fundamentally broken*. This is because the presence of the instrumentation code in the program affects the measured times in *unpredictable* ways. In your UROP project you will conduct a scientific experiment to validate our hypothesis. Your study will closely follow the study design we used for our PLDI 2010 paper [2], in which we showed that many production-quality sampling-based profilers are based on a flawed, and thus inaccurate, approach. Our goal is that your UROP project will lead to a scientific publication about the fallacy that instrumentation-based profiling leads to more accurate profiles.

You will acquire intimate knowledge about approaches to profiling and specific Java profiler implementations (such as NetBeans, JIP, hprof). You will devise and conduct experiments involving state-of-the-art benchmark suites (SPEC, Dacapo) on multiple platforms (different Java virtual machines, different operating systems and hardware). You will need to write code to extract information from the profilers, and to analyze and compare the various profiles. You will also need to visualize the resulting data. In this project you will learn how to do good experimental computer science. You have to have excellent Java and/or C programming skills, and you have to be self-motivating, independent, and persistent. If you generally under-promise and over-deliver, this project can give you the chance to get your first scientific publication.

Many programs use recursive data structures, such as linked lists, trees, or graphs. In this UROP project you will work on a dynamic analysis for identifying individual instances of recursive data structures in Java programs and for automatically tracking their sizes (i.e., find all trees that have been created, and for each tree, determine the number of its nodes).

```java
public class TreeNode {
    TreeNode left;
    TreeNode right;
    TreeNode(TreeNode l, TreeNode r) {left=l; right=r;}
    TreeNode() {left=null; right=null}
}
public class Main {
    public static void main(String[] args) {
        TreeNode tree1 = new TreeNode(new TreeNode(), new TreeNode());
        TreeNode tree2 = build(Integer.parseInt(args[0]));
    }
    static TreeNode build(int h) {
        return h==0 ? null : new TreeNode(build(h-1), build(h-1));
    }
}
```

Given the above example program, the dynamic analysis would be able to determine (1) that the program allocates two tree instances (referred to by tree1 and tree2). Moreover, (2) it would be able to track the size of each tree instance, i.e., that the first instance consists of three TreeNodes, while (in a run using the value 3 as a command line argument) the second instance consists of seven TreeNodes. The analysis is dynamic, so it counts the number of objects that are allocated, and it observes all updates of reference fields (e.g. changes to the TreeNode.left and TreeNode.right fields) to track how the nodes of a data structure point to each other. To do that, it instruments the program by injecting extra code at each allocation site (NEW instruction) and each field write (PUTFIELD instruction) using a bytecode editing framework such as ASM.

A similar analysis for C programs has been described by Raman and August [1]. Your task is to implement a simplified version of that analysis for Java.

For this project you need excellent Java programming skills, and you have to be self-motivating, independent, and persistent. The analysis requires implementing of some graph algorithms (e.g. to find strongly connected components in the graph of nodes). Instrumenting Java classes requires an understanding of Java bytecode (if you are excited about understanding what goes on under the hood of a Java program, this skill can be acquired during the project). The project is extensible in multiple directions, depending on your skills and interests.

Replication is a well-established approach to increasing database availability. By replicating data items in multiple servers, the failure of some servers does not prevent clients from executing transactions against the system. Database replication in the context of crash-stop failures has been largely studied in the past years. When a crash-stop server fails, it silently stops its execution. More recently, a few works have considered database replication under byzantine failures. Byzantine failures are more severe than crash-stop failures since failed servers can present arbitrary behavior.

Several protocols for the crash-stop failure model are based on deferred update replication. Deferred update replication scales better than state-machine replication and primary-backup replication. With state-machine replication, every update transaction must be executed by all servers. Thus, adding servers does not increase the throughput of update transactions. With primary-backup replication, the primary first executes update transactions and then propagates the database changes to the backups, which apply them without re-executing the transactions. The throughput of update transactions is limited by the capacity of the primary, not by the number of replicas. Deferred update replication scales better because it allows all servers to act as “primaries”, locally executing transactions and then propagating the modifications to the other servers. As applying transaction modifications to the database is usually cheaper than executing transactions, the technique provides better throughput and scalability.

The goal of this project is to implement a Byzantine fault-tolerant deferred update replication protocol that has been developed within the Distributed Systems group.

For more details, please contact Fernando Pedone.
Virtual 3D Table Tennis


(Supervisors: Profs. Michael Bronstein, Kai Hormann, and Marc Langheinrich)

A user plays the table tennis game at the Virtual Reality 2005 event in Bonn, Germany. Image Credit: Rusdorf, Stephan, et al. © 2006 IEEE.

Imagine there is a table tennis athlete and neither his trainer nor his friends are around to help him practising. With the system that you will develop in this UROP project, he does not have to worry about this situation anymore, because it allows him to still continue improving his skills! Your goal in this project is to combine 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.

In detail, your tasks are

- Model a basic 3D table tennis setup with table, net, ball etc. and display it with the 3D projector using OpenGL.
- Simulate the correct ball movement based on simple physical principles.
- Use colour segmentation on the Kinect’s video data to detect the racket position.
- Use plane fitting to the Kinect’s depth data to reconstruct the racket orientation.
- Calibrate the 3D projector and the Kinect in order to have a perfect 3D illusion.

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](mailto:kai.hormann@usi.ch)).*
Summer project in Geometric Algorithms – Design and Implementation

The medial axis of rectilinear polygons in the $L_\infty$ metric.

This project will introduce students to research in the area of geometric algorithms – design and implementation. The Voronoi diagram is a simple but powerful geometric object that encodes nearest neighbor information with numerous applications in diverse areas. Given a set of point sites $S$ in the plane, the Voronoi diagram is a partitioning of the plane into regions, one for each site, such that the Voronoi region of a site $s$ is the locus of points closer to $s$ than to any other site. The following figure illustrates the ordinary Voronoi diagram of a set of points in the Euclidean plane.

In the interior of a simple polygon, the Voronoi diagram is called medial axis. The medial axis captures the shape of the given polygon in a natural manner. The $L_\infty$ metric is a simple distance function that measures distance in a square-like fashion and provides a simpler alternative to the standard Euclidean distance. In the $L_\infty$ metric the medial axis always simplifies to a straight line skeleton. The goal of this project is to implement a plane sweep approach to compute the $L_\infty$ medial axis of a rectilinear polygon. This is part of a larger project that aims in offering generalized $L_\infty$ Voronoi diagrams in CGAL – the Computational Geometry Algorithms Library. CGAL (http://www.cgal.org/) is an open source C++ library that provides access to efficient and reliable geometric algorithms. The topic of the project can adapt to more theoretical or practical according to the interests of the student.

Skills required
Good background in Algorithms and Data Structures, C++ programming, familiarity with Discrete Mathematics.

Contact
Prof. Evanthia Papadopoulou, SI-209
Summer project in pattern recognition for sets of rectangles

Methods for pattern recognition among sets of rectangles.

This project will investigate 2-dimensional pattern recognition techniques for sets of rectangles. The problem is motivated by the need in the semiconductor industry to identify critical patterns in a chip layout design.

From a layout perspective, a VLSI design is a collection of layers (representing different materials) where each layer consists of geometric shapes. The geometric shapes are typically axis parallel polygons and certain contact layers are even simpler, consisting of only rectangles. To print the VLSI design into actual chips very advanced lithographic techniques are being used. VLSI designs are typically very large, consisting of millions or billions of shapes. The ability to extract representative patterns is essential for lithographers as such patterns could be used by the lithographers in their study of printing the actual VLSI shapes in the very small dimensions of the near future.

In this project, we would like to study geometric pattern recognition techniques for such sets of rectangles representing VLSI shapes. The first step is the study of existing literature on pattern recognition among sets of rectangles and to create a summary of existing pattern recognition techniques for rectangular shapes. Then we would like to consider geometric approaches to define “similarity” between sets of rectangles. If successful, we would like to implement and try such techniques experimentally.

Skills required
Creativity, good background in algorithms and data structures, and good programming skills.

Contact
Prof. Evanthia Papadopoulou, SI-209
Funsquare – Facilitating Social Interaction
Through Shared Experiences

Proposal for a UROP 2011 Project in Oulu, Finland

(Supervisor: Prof. Marc Langheinrich)
Funsquare is a public-display service that aims to facilitate social interaction between strangers, acquaintances, and friends. It displays context-aware "fun facts" about a place using an arbitrary range of topics, e.g., sustainability, history, or science. By providing conversational topics and becoming a special feature of the place, Funsquare can help stimulating socialization through ‘triangulation’, an effect where special feature of the place acts as a link between people.

As part of the international research project "PD-Net", two students will have the opportunity to participate in this UROP project that will take place in Oulu, Finland, over the summer. Travel to Oulu, as well as room and board will be provided, in addition to the UROP stipend. Students will form part of an international research team from four European countries – Germany, Great Britain, Portugal, and Switzerland – that will develop, deploy, and evaluate the above described “Funsquare” application over the course of 8 weeks in the city of Oulu. The researchers will join 3 other international teams who will deploy 3 additional applications on the Oulu public display infrastructure during the summer.

In detail, the various tasks are

- Create a prototypical core system engine (“Fun Fact Generator”) and a corresponding user interface (the “Funsquare Community Interface”) as part of a bachelor project
- Assemble, test, and deploy the system on the existing public display infrastructure in Oulu
- Assist project members during public trials (data collection, system maintenance)
- Enjoy the Finish summer with students and researchers from all over Europe!

In order to qualify for the UROP project, you should have completed one of the available “Funsquare” bachelor projects.

*If you are interested, please contact Prof. Langheinrich ([marc.langheinrich@usic.ch](mailto:marc.langheinrich@usic.ch)).*
Nature-inspired algorithms such as Particle Swarm Optimization and Evolutionary Algorithms are among the most powerful algorithms for global optimization problems. In 2009 Yang formulated a new metaheuristic by combining Lévy flights with the search strategy via the Firefly Algorithm.

A Lévy flight is a random walk in which the step-lengths are distributed according to a heavy-tailed probability distribution. Specifically, the distribution used is a power law of the form $y = x^{-\alpha}$ where $1 < \alpha < 3$ and therefore has an infinite variance. Numerical studies and results suggest that the proposed Lévy-flight firefly algorithm (LFA) can be superior to existing metaheuristic algorithms at least for certain problems.

The general idea of the firefly algorithm is that new solutions are created by moving towards good existing solutions in the search space. The quality of a solution is associated with the brightness of a firefly and varies in the simplest case according to the inverse square law or according to some exponential decrease. This is combined with moving particles according to the Lévy distribution, which is inspired by various studies that have demonstrated the typical characteristics of Lévy flights for many animals. A study by Reynolds and Frye shows that fruit flies explore their landscape using a series of straight flight paths punctuated by a sudden 90° turn, leading to a Lévy-flight-style intermittent scale free search pattern as shown in the figure.

Studies on human behavior also show the typical feature of Lévy flights. Subsequently, Lévy flights have been applied to optimization and search, and preliminary results show its promising capability.

**Hints for the solution:**

1) Literature research about the state of the art for particle swarm optimization like techniques.
2) Summary of the results and description of the Lévy-flight firefly algorithm.
3) Implementation of the algorithm and test with different standard benchmark functions.
4) Comparison of the results with other algorithms as Standard PSO or Ensemble-based Threshold Accepting (which are already implemented).
5) Conclusion and systematization of the results, discussion of strengths and weaknesses, future prospects.

**Contact:** Dr. Jörg Lässig, ICS, Lab. Building, 3rd Floor; Email: joerg.laessig@usi.ch
Paralleling OpenSMT Towards Cloud Computing

The success of OpenSMT, open-source SMT-solver born at USI two years ago, motivates us to explore new ideas in its development: now we want it to become parallel! Although, the general problem of paralleling of SAT– and SMT–computation is known to be very hard, we believe OpenSMT can do it and thus benefit from the modern cloud–computing resources. We propose two projects to USI students that target this challenge.

Project #1:

Recent work at Microsoft Research shows that the massive speedup can be achieved from running the same formula on slightly different versions of the solver (ManySat, Z3). Theoretically not very efficient, the approach benefits from the fact that the solver instances share the facts they learned during computation. Student is expected to implement clause sharing in OpenSMT and investigate the speedup and the overhead of this sharing. Besides the infrastructure to run multiple instances of the solver with different search strategies should be established.

Project #2:

We propose to explore existing computation-intensive parts of the OpenSMT algorithms, e.g. linear rational arithmetic solver (LRA), from the paralleling point of view. Various instructions of the algorithm are not limited with sequential ordering and can be executed simultaneously. The student is expected to build and implement the parallel version of the algorithm (for multi–core and multi–node running environments) and investigate the speedup and the overhead of it.

Background:

OpenSMT is a compact and open–source SMT–solver written in C++ with the main goal of making SMT–solvers easy to understand and use as a computational engine of formal verification (more details are available at the OpenSMT project page).

What we expect from a student:

- C/C++ experience (OpenMP/MPI experience is a big advantage);
- Willing to learn and understand new frontiers;
- Passion to open–source development.

What we can give to a student:

- Experience in C++ development for clouds/multi–core machines;
- Understanding of how complex industrial problems are solved with "state–of–the–art" research techniques;
- Knowledge in fields of decision procedures and formal methods.

Both projects will be conducted under a supervision of the Lab members, mainly Roberto Bruttomesso and Aliaksei Tsitovich. Don't think that it is too complex. Yes, it is, but we will help you to "capture this flag".

Contact: aliaksei.tsitovich@usi.ch
EverMail
UROP Project 2011

Advisor: Prof. Michele Lanza and Alberto Bacchelli

Project Description

The goal of this project is to improve and lighten REmail—the Eclipse plugin shown in figure that integrates email communication in the IDE [1]—by creating EverMail, a web service for managing email data storage, analysis, and retrieval.

In the first period, the student will implement the backend of EverMail, in the second period the focus will be on visualizing data available using EverMail APIs.

On the backend EverMail must (1) handle the importing of email data from at least two kinds of data sources (i.e., MBox files and Markmail service [2]); (2) automatically fetch newly available data; and (3) persist data in a document-oriented database (such as CouchDB [3] or MongoDB [4]) to allow seamless email meta-model evolution and the usage of Map/Reduce techniques. EverMail must provide access to its data through RESTful JSON APIs, which will be used by REemail to provide results faster and more effectively.

On the front-end, new visualizations will be created to explore the available email data. They can be either in the form of a web application or included in REemail.