Problem Solving in Computer Science: Week 10

Scribe on duty: Hossein Hojjat

November 18, 2008

1 Transactional Memory

In this section we review the concept of transactional memories from [1].
A traditional way to make threads communicating and synchronizing with each other is by using locks and condition variables. These methods of synchronization are fundamentally flawed, since they do not compose well. Concurrent programs based on locking are also hard to follow and debug. In many cases, programming with locks sounds to be absurdly hard, e.g. for a scalable double-ended queue when the queue size is small. The problem in a sequential settings is easy to solve, but when the calls become concurrent it turns out to be an academic challenge to solve it.
The transactional memory approach is an easy way to look at the synchronization problem. Similar to databases, we use transactions in order to access the memory. A set of instructions are wrapped with an *atomic* command to show that they obey the "all or nothing" semantics. So whenever an exception occurs in the middle of executing a transaction none of the instructions in the transaction can affect the state of the program. A direct consequence of using transactions is to avoid deadlocks, since nowhere during this approach something is locked. An effective way to make transactions is to use optimistic concurrency. Here, any access to the memory is logged into a transaction log. After the transaction finished, it is the duty of the system to commit all the reads and writes in the transaction log into the memory in one step. If the system could not commit the log, nothing would be changed in the memory. One has to merely re-run the transaction so that the changes take effect. So, instead of using condition variables to wake up threads, the transactions are run again and again until they successfully finish. The main problem with transactional memories is their run-time cost. In transactional memories most of the synchronizing burden is put on the system whereas the programming task becomes easier.

2 Writing a Paper: Abstract and Introduction

There are mainly two approaches to come up with a paper:

- Idea-Research-Write: In this approach we have initially an idea, we do research and after that we write down our results.
- Idea-Write-Research: In this approach we have initially an idea, we write it down and do research in parallel with writing.

Usually the second approach helps to develop the initial idea better. Writing down the research idea allows us to gain feedback from different people, and to find the interesting points. After finding the important points, it is nice to focus on the most interesting one in the paper. The paper should point to the most interesting result. The idea of a paper should not be the computer science breakthrough of the year. To write a papet one should not necessarily have an idea that is difficult to understand.
A paper usually consists of the following sections [2]:

1. Abstract
2. Introduction
3. The problem
4. My idea
5. The details
6. Related work
7. Conclusions and further work

Some papers do the related work section in the introduction part or around it. It is important that at the very end of the paper there exists a comparison with the competing ideas. The comparison should be concrete and concise. A bad idea is to put a dozen of related works with the hope to cite the papers of the possible reviewers and make them happy. It is more harmful rather than helpful to put a list of 20 papers and do not say anything about them. It is more logical to pick a small set of papers and do the comparison only for that set.
In this document we focus on the first two parts, abstract and introduction.

2.1 Abstract

The main purpose of an abstract is to help the reader to decide as soon as possible if she wants to read the paper or not. It is also useful when someone wants to search the paper in a database. It is quite rational to write the abstract last. An Abstract contains 4 or 5 sentences. First of all it states the problem, describes why it is interesting, shows the results of the solutions and finally demonstrates what follows from the solutions.

2.2 Introduction

One of the best ways to introduce the problem is to give a small concrete example. It illustrates the problem in a very short manner. After reading a good introduction the reader wonders how interesting the paper is and why she has not thought about it herself. A clear introduction gives the feeling to the reader that she can continue the paper herself. The rest of the paper is only there if the reviewer does not trust the author, and wants to check if the claims are true. The results should be clear and can be checked by the reviewer. One should not use emotional words such as *fantastic*, *wonderful* or *marvelous* for the results. It is also helpful to state the contributions in a bullet style.

References

[1] Simon Peyton-Jones, keynote speech at OSCON 2007 on Software Transactional Memory.

[2] Simon Peyton-Jones, “How to write a great research paper”, Invited talk at the Technical University of Vienna, October 2004.