Parallel Computing: Modern Trends in Research, Education, and Application

Peter Salhofer
FH JOANNEUM
peter.salhofer@fh-joanneum.at

1. History and Motivation

The mini-track on "Parallel Computing: Modern Trends in Research, Education and Application" is definitely one of the youngest mini-tracks at HICSS since it is just going into its second round. Of course there are many specialized conferences that are exclusively dedicated to parallel and high performance computing and which might probably be perceived a better place to meet experts in the field and exchange expertise. So why should there be a need to place a topic like this at a more general system sciences conference?

Well, intensive research in parallel computing is done for decades. In its early stage it's been an exclusive field requiring specialized and extremely expensive hardware. On the other side, classes in parallel computing are typically still rare in computer science curricula and probably most of them are elective. These look like the perfect constraints to make sure that the topic stays in a niche.

On the other side we see that parallel infrastructures have become commodity. Nowadays every cell-phone has more computing power than a performance workstation had some years ago. Taking for example a look at the ever-increasing number of cores in contemporary computers, we clearly see that creating programs that effectively use parallel infrastructures should be a standard skill for every software engineer. So it's really time to leave the niche and to approach a broader community in the field of IT. But where's the problem in doing so? In fact we all know that writing parallel programs can lead to all sorts of problems, most prominently race-conditions and dead-locks. Thus, it is still perceived to be tricky and complex and it actually really can be. So what we need are ways to create efficient parallel programs with minimal complexity. Thus, the idea behind this mini-track is actually to give such approaches a chance.

2. This Years Content

Apparently the call for papers worked pretty well this year since there will be three very good an interesting papers. We start with a contribution that analyzed several task-parallel frameworks that support so called "implicit dependencies". This should actually reduce the complexity of writing parallel programs, avoid typical errors and achieve best efficiency.

The next paper demonstrates how different strategies in splitting up work to parallel resources can be used to achieve the best performance in solving a very common problem. It presents new ideas that seem to work pretty well on executing XPath-Queries.

Finally there is a paper that actually analyses the hypothetical effect of bringing real parallelism to a very hot field in modern computer applications, which is the execution of JavaScript. This is especially interesting, since JavaScript is deliberately single threaded but uses the concept of asynchronous programming a lot. The paper contains the results of a thorough analysis of common, popular but resource-intensive JavaScript libraries and how specific language features, that might probably hamper the efficient use of parallel computing, are used by software engineers writing such programs. It gives a nice idea of what could happen if parallel programming manages to really enter the world of mainstream software engineering.

URI: http://hdl.handle.net/10125/41913
ISBN: 978-0-9981331-0-2