

# HIPEAC – Adaptive Compilation Cluster

## Research Topic: Automatic Parallelization

Collected by Mikel Lujan  
The University of Manchester 24/January/2009

Submitted Research Interests and Future Work Ideas (ordered as received plus mine last)

- Konrad Trifunovic and Albert Cohen  
INRIA – Alchemy Research Group  
<https://alchemy.futurs.inria.fr/site/>
- Andy Nisbet  
Manchester Metropolitan University  
<http://www.docm.mmu.ac.uk/STAFF/A.Nisbet/>
- Bilha Mendelson  
IBM Research Lab in Haifa  
[http://domino.research.ibm.com/comm/research\\_people.nsf/pages/bilha.index.html](http://domino.research.ibm.com/comm/research_people.nsf/pages/bilha.index.html)
- Hans Vandierendonck  
Ghent University  
<http://www.elis.UGent.be/~hvdieren/>
- Martin Schindewolf  
University of Karlsruhe  
<http://itec.uka.de/~schindew/>
- Eduard Ayguade  
Barcelona Supercomputing Center & UPC  
<http://personals.ac.upc.edu/eduard/>
- Marcelo Cintra  
The University of Edinburgh  
<http://homepages.inf.ed.ac.uk/mc/>
- Enrique S. Quintana Orti  
Universitat Jaume I  
<http://www3.uji.es/~quintana/>
- Mikel Lujan and Ian Watson  
The University of Manchester  
<http://www.cs.manchester.ac.uk/apt>

## INRIA – Alchemy Research Group

Konrad Trifunovic      konrad.trifunovic at inria.fr  
Albert Cohen      Albert.Cohen at inria.fr

We are interested in coarse grained parallelisation. We are doing research in static analysis for extracting coarse grained (but also fine grained) parallelism, using polyhedral model abstraction.

Polyhedral model allows us to extract data dependences, iteration domains and other statically extractable information from program loops (C, Fortran or any other imperative language). This information allows us to compute ALL the possible parallelism, respecting the dependences that we have extracted.

We do not advocate for a new programming model, as our goal is to extract parallelism from already existing, legacy, numerical kernels and scientific applications written in C or Fortran.

Though it works well with static control programs (fixed or parametrized loop bounds, array accesses expressed as linear functions, no irregular control flow), we are not able to represent all the classes of programs inside polyhedral model. Pointers, dynamic control flow, function calls etc. are not yet representable in this model.

What kind of collaboration I'm interested in:

1. What I will be interested in is to seek for possible collaborations on dynamic parallelization.
2. Also we need some insight into the work on efficient code generation for synchronization of multiple threads.

## Manchester Metropolitan University

Andy Nisbet      A.Nisbet at mmu.ac.uk

I am interested in parallelisation targetting multicore architectures where each processor has a flexible instruction set architecture that is supported using a combination of reconfigurable and embedded nonconfigurable logic blocks.

I am also interested in parallelisation for customisation of instruction sets in order to perform loop nest acceleration. I am interested in performing an evaluation of what might be possible for FPGAs using multicore MicroBlaze/open source processor cores augmented with custom instructions.

## IBM Research Lab in Haifa

Bilha Mendelson      BILHA at il.ibm.com

We in IBM Haifa are interested in the area of parallelisation with respect to the GCC compiler effort. We are working with the Graphite effort for this. This includes auto-parallelisation and auto-vectorisation.

## Ghent University

Hans Vandierendonck

Hans.Vandierendonck at elis.UGent.be

My main research interest is auto-parallelization, focussing on techniques to parallelize control-intensive code with irregular data flow. In the past, our work has depended strongly on profile-based techniques, but currently we are using conservative static analysis.

In my ideal research project, I would like to investigate auto-parallelization techniques (static analysis), forms of parallelism (beyond DO-ALL and pipeline parallelism), how to intelligently employ speculative parallelism, combining thread-level parallelism at multiple levels (coarse-grain vs. fine-grain; creating and scheduling threads), automatically extracting code for execution on accelerator architectures and hardware support for thread-level parallelism (fast thread creation and synchronization are essential to support short threads).

A research project containing all of the above is of course not ideal, as it contains too many topics, but at least it gives you some insight on my research interests.

## University of Karlsruhe

Martin Schindewolf

schindew at ira.uka.de

The ideal research project would focus on advanced optimisations in parallel programming environments (especially TM-based). Semantics of transactions provide convenient and desired properties to the programmer. Atomicity, Consistency and Isolation form the basis for transactional programming. At the same time their persistence prevents useful compiler optimisations and especially STM systems come with a significant performance penalty. These are limitations that could be mitigated by weakening the semantics of transactions and, thus, enabling more advanced compiler optimisations.

## Barcelona Supercomputing Center & UPC

Eduard Ayguade

eduard.ayguade at bsc.es

My interests is to have a clear boundary defined between the compiler and the programming model, and between the compiler and the runtime system/architecture. This boundary defines the responsibilities (or co-responsibilities) of each one of the 4 elements mentioned. Some possible topics are: autotuning for different parameters that the programming model offers to the programmer, code transformations for some code restructuring pragmas and their interaction with the pragmas for parallelization, code and data transformations for pragmas oriented to specifying changes in data layout association, ...

## The University of Edinburgh

Marcelo Cintra

mc at staffmail.ed.ac.uk

A promising approach to facilitate writing parallel code is speculative multithreading (SpMT). It has been extensively explored in three major flavors: thread-level speculation (TLS), transactional memory (TM), and helper threads (aka subordinate microthreads and run-ahead threads). While reasonable success has been achieved in each of these three SpMT models, the current state-of-the-art is still not yet a compelling case. The main reason for this shortcoming is a lack of a holistic approach to it in terms of its different models. What we propose is a concerted effort to push the state-of-the-art in SpMT compilation in a way that all models can be considered and used simultaneously for different program constructs and phases.

Universitat Jaume I

Enrique S. Quintana Orti

quintana at icc.uji.es

Supportive Tools for Parallelizing Applications on Heterogeneous Multicores

Objectives:

- Tools: compilers and runtime systems for an OpenMP-like parallelization framework. Auto-tuning and scheduling should receive special attention here.
- Applications: numerical/non numerical, scientific/industrial
- Target architectures: general-purpose multicores, hardware accelerators (Cell, ClearSpeed), GPUs, and FPGAs

Note: There is a strong link between this project and programming models that will be surely investigated in the research cluster on "programming models and operating systems".

The University of Manchester

Mikel Lujan

mikel.lujan at manchester.ac.uk

Ian Watson

Watson at cs.man.ac.uk

We have a strong background on adaptive Just-in-Time compilation (such as Java Virtual Machines) and parallel computer architecture. We have been working on run-time parallelization of applications relying on speculation.

We are interested in:

- Improving information flow among programming model <-> static compiler <-> run-time compiler <-> multi-core architecture as to maximize parallelization opportunities.
- Advancing the state-of-the-art of hardware support for speculation; current examples being Transactional Memory and Thread Level Speculation
- Developing techniques to control when and where run-time speculative parallelization will be profitable
- Developing run-time parallelization techniques to deal with dynamic applications