

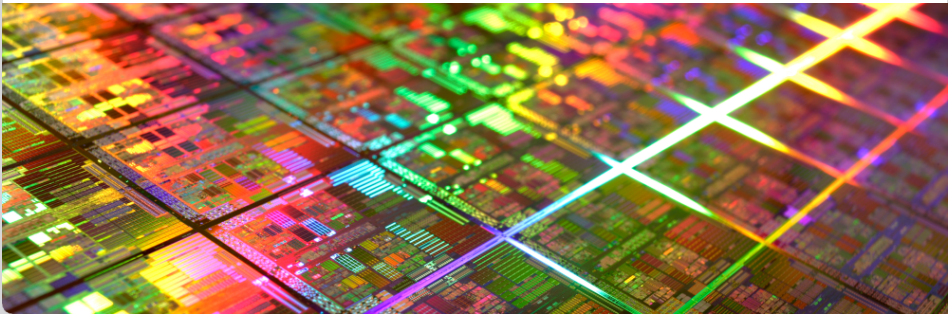
# Compiler-Directed Performance Model Construction for Parallel Programs

Martin Schindewolf,<sup>1</sup> David Kramer,<sup>1</sup> and Marcelo Cintra<sup>2</sup>

<sup>1</sup>Karlsruhe Institute of Technology - Institute of Computer Science & Engineering

<sup>2</sup>School of Informatics - University of Edinburgh

• February 25th, 2010



## Why Performance Prediction

- Cost estimation
- Optimization
- Comparison of computer systems

## Performance Prediction for Supercomputers is challenging

- Processor
- Interconnect
- Compiler optimizations
- Source Code,...

## Previous Work

- 1 Simulation
- 2 Deterministic analytical models
  - Manual construction [1, 2, 3]
  - Construction difficult and time consuming
  - Evaluation easy and adoptable
  - Quadratic programming [7, 8]
  - Modeling Assertions (MA) with linear symbolic models [4, 5]
- 3 Neural networks [6]
- 4 Hybrid approaches [13, 9]

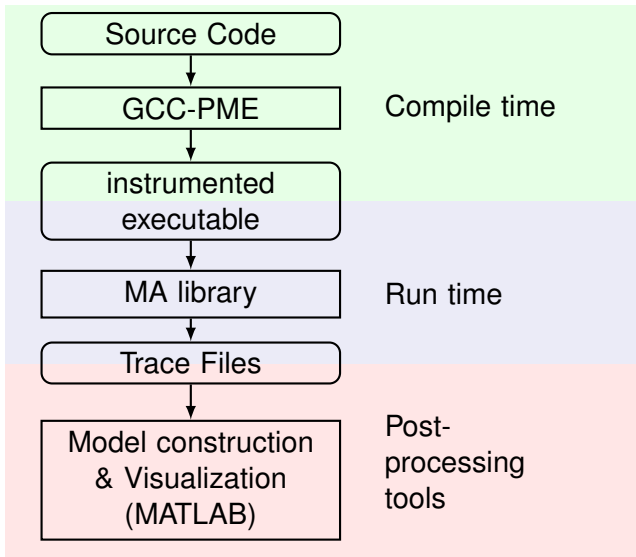
## Our Approach

- Compiler-directed performance models
- Semi-automated model construction

- 1 GCC with Program Modeling Enhancements
- 2 GCC – Structure and Implementation
- 3 Quadratic Programming – Results
- 4 Conclusion & Outlook

- Compiler-directed model construction
  - Programmer uses new compiler switches to guide instrumentation
  - No knowledge of application required
  - Compiler instruments executable
  - Manual instrumentation not necessary
  - Simple models (e.g. floating point operations)
  - Few parameters
  - Goal: Precise prediction
- Validated through use of performance counters

# Overview of the frame work



## PAPI Performance Counter

- Uniform interface
- Start, stop, reset and read counters
- Supports various architectures
- MA Library addresses PAPI

## NASA Parallel Benchmark Suite (NPB)

- Defines problem classes (S, W, A, B, C, D)
- Classes depend on input data-set
- Parallelized with OpenMP or **MPI**

- 1 GCC with Program Modeling Enhancements
- 2 GCC – Structure and Implementation**
- 3 Quadratic Programming – Results
- 4 Conclusion & Outlook

# Fundamentals: Structure of GCC

## ■ Front End

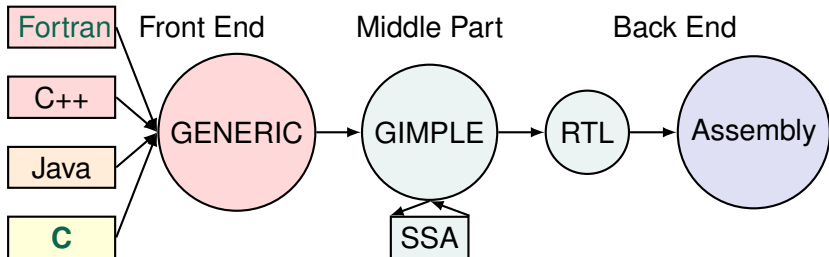
- Depends on Source Language

## ■ Back End

- Architecture-dependent

## ■ Middle Part

- Optimization
- Language-independent
- Architecture-independent



## Where?

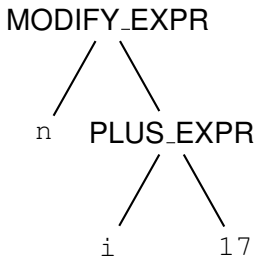
- Middle Part
- Written in C
- Program represented as tree

## What is added?

- Pass for operation count analysis
- Recognize frequently executed parts
- Pass for annotating program parts
  - Loops, basic blocks
  - Function calls
  - Uses naming scheme
- Annotations target MA library

Example:

```
n = i + 17;
```



# Example Loop Annotation

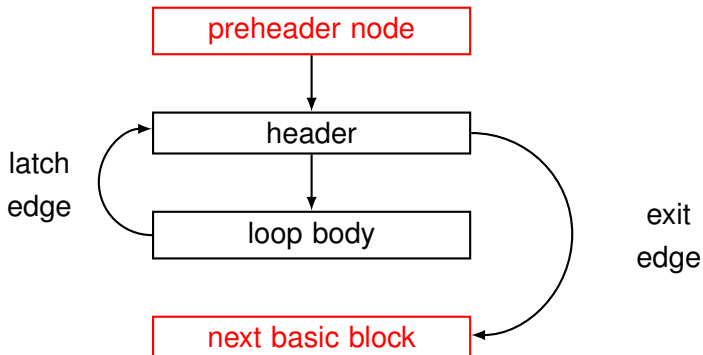


Figure: GCC internal representation of a single exit loop [15].

# Example Loop Annotation

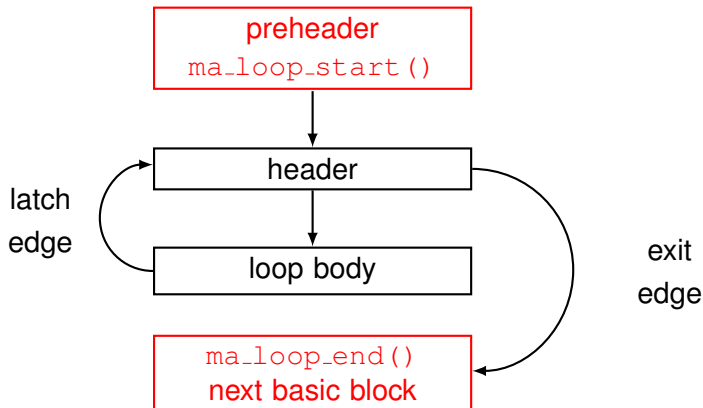


Figure: GCC internal representation of a single exit loop [15].

## XC 6000 of Steinbuch Centre for Computing at KIT

- Itanium II Cluster
- 113 nodes
- 298 processors

## Applications

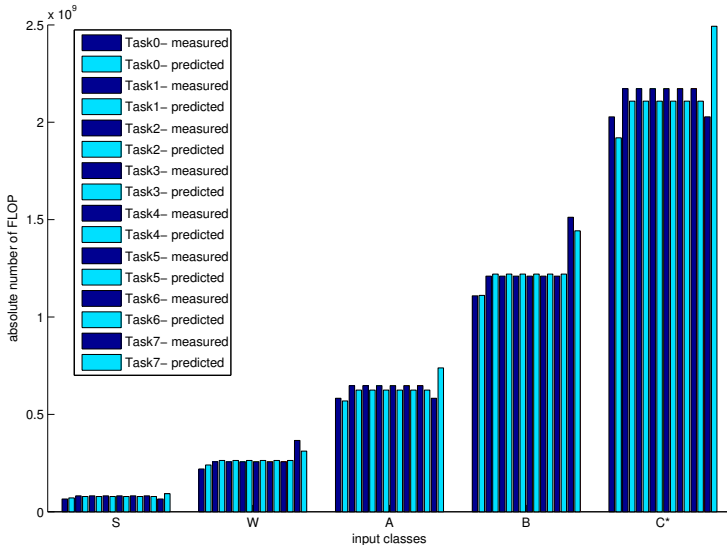
- IS and CG from NPB parallelized with MPI
- Lattice-Boltzmann with MPI

- 1 GCC with Program Modeling Enhancements
- 2 GCC – Structure and Implementation
- 3 Quadratic Programming – Results**
- 4 Conclusion & Outlook

## Model Construction

- Run small input data-sets (S, W, A, B)
- Construct model and predict run with larger problem size (C)
- Method: Quadratic Programming
  - Relates input parameters and event counts
  - Model can be constructed semi-automatically
  - Results from more input data-sets implicitly refine the model

# QP Model - Lattice-Boltzmann



# Quadratic Programming - Results

	Number of MPI Tasks			
Benchmark	2	8	16	32
IS	0%	0%	0%	0%
CG	19.8%	21.1%	23.3%	29%
LB	0.2%	0%	-1.3%	-12.1%

**Table:** Average relative error predicting input class C for IS, CG, and LB.

- 1 GCC with Program Modeling Enhancements
- 2 GCC – Structure and Implementation
- 3 Quadratic Programming – Results
- 4 Conclusion & Outlook

## Presented

- Compiler-directed performance models
- Semi-automated model construction
- Requires no knowledge about application
- MA Library
  - Collects event counter of MPI tasks
  - Generates event traces
- MATLAB constructs and visualizes performance model
- Results
  - LB, IS with good results
  - CG with results (error > 20%)
  - Limit of the model

## Future Work

- Enrich model with architectural details
  - MPI-Events
  - Load/Store Operations
  - Integer Operations
- Combine models with different event types
- Target different models (e.g. linear symbolic models)
- Better integration of PME passes with optimization passes

Thank you for your attention!

Questions?

- [1] Matthew I. Frank, Anant Agarwal, and Mary K. Vernon.  
LoPC: modeling contention in parallel algorithms.  
In *SIGPLAN Not.*, vol. 32, number 7, year 1997, issn 0362-1340,  
pages 276–287, ACM.
- [2] Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and  
Chris Scheiman.  
LogGP: Incorporating Long Messages into the LogP Model — One  
step closer towards a realistic model for parallel computation.  
University of California at Santa Barbara, year 1995, CA, USA.

[3] David E. Culler, Richard M. Karp, David Patterson, Abhijit Sahay, Eunice E. Santos, Klaus Erik Schauer, Ramesh Subramonian, and Thorsten von Eicken.

LogP: a practical model of parallel computation.

In *Commun. ACM*, volume 39, number 11, year 1996, issn 0001-0782, pages 78–85, ACM.

[4] S.R. Alam and J.S. Vetter.

A framework to develop symbolic performance models of parallel applications.

In *IEEE International Parallel & Distributed Processing Symposium*, volume 0, year 2006, isbn 1-4244-0054-6, pages 368ff, IEEE Computer Society.

[5] Nikhil Bhatia, Sadaf R. Alam, and Jeffrey S. Vetter.

Performance Modeling of Emerging HPC Architectures.

In *HPCMP Users Group Conference*, volume 0, year 2006, isbn 0-7695-2797-3, pages 367–373, IEEE Computer Society.

[6] Engin Ipek, Bronis R. de Supinski, Martin Schulz, and Sally A. McKee.

An Approach to Performance Prediction for Parallel Applications.

In *Euro-Par 2005 Parallel Processing*, pages 196–205, year 2005, Springer-Verlag.

[7] Gabriel Marin.

Semi-Automatic Synthesis of Parameterized Performance Models for Scientific Programs.

Department of Computer Science, Rice University, Houston, Texas, April, 2003.

[8] Gabriel Marin and John Mellor-Crummey.

Cross-architecture performance predictions for scientific applications using parameterized models.

In *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on Measurement and modeling of computer systems*, year 2004, isbn 1-58113-873-3, pages 2–13, ACM.

[9] Leo T. Yang, Xiaosong Ma, and Frank Mueller.

Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution.

In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, year 2005, isbn 1-59593-061-2, pages 40, IEEE Computer Society.

[10] Laura Carrington, Allan Snavely, and Nicole Wolter.

A performance prediction framework for scientific applications.

In *Future Generation Computer Systems*, volume 22, number 3, year 2006, issn 0167-739X, pages 336–346, Elsevier Science Publishers.

[11] Benjamin C. Lee and David Brooks.

Efficiency trends and limits from comprehensive microarchitectural adaptivity.

In *SIGOPS Oper. Syst. Rev.*, volume 42, number 2, year 2008, issn 0163-5980, pages 36–47, ACM.

[12] Benjamin C. Lee, Jamison Collins, Hong Wang, and David Brooks.

CPR: Composable Performance Regression for Scalable Multiprocessor Models.

*In MICRO: 41st International Symposium on Microarchitecture*, November, year 2008.

[13] Ewa Deelman, Aditya Dube, Adolfo Hoisie, Yong Luo, Richard L. Oliver, David Sundaram-Stukel, Harvey J. Wasserman, Vikram S. Adve, Rajive Bagrodia, James C. Browne, Elias N. Houstis, Olaf M. Lubeck, John R. Rice, Patricia J. Teller, and Mary K. Vernon.

POEMS: end-to-end performance design of large parallel adaptive computational systems.

*In WOSP '98: Proceedings of the 1st International Workshop on Software and Performance*, pages 18–30, year 1998.

[14] Sameer S. Shende and Allen D. Malony.

The Tau Parallel Performance System.

In *Int. J. High Perform. Comput. Appl.*, volume 20, number 2, year 2006, issn 1094-3420, pages 287–311, Sage Publications, Inc.

[15] Zdeněk Dvořák.

Loop optimizer cheatsheet.

<http://gcc.gnu.org/wiki/GettingStarted?action=AttachFile&do=get&target=loopcheat.ps>.