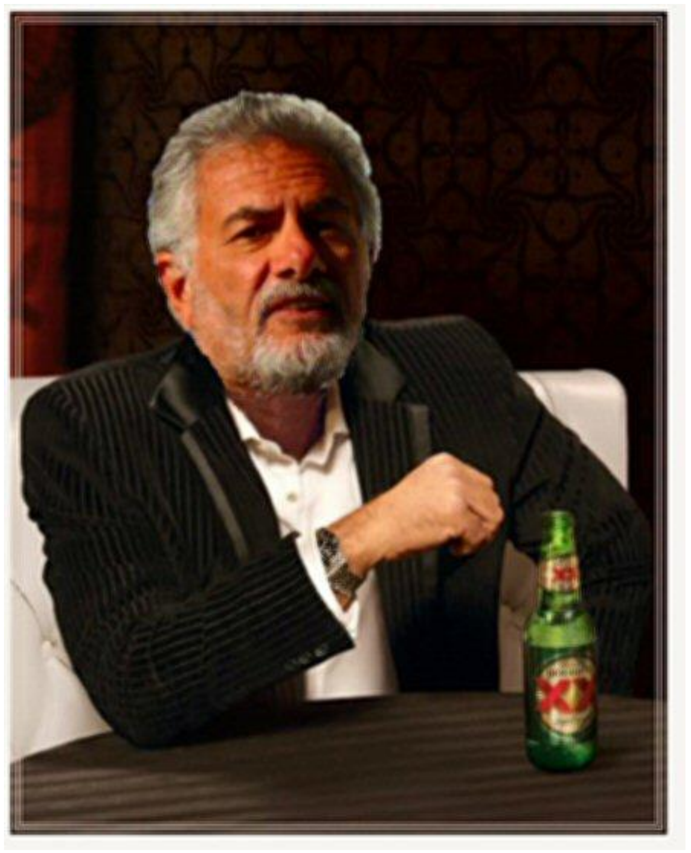
Tough People to Follow

Yale N. Patt, 2011 Recipient



Josh A. Fisher, 2012 Recipient

A Tribute to my Fellow Travelers

- HPS - the beginning
- IMPACT – an Era with Bob Rau
- GPU Computing
- Some thoughts

MICRO-18 (1985)

ce of
iodel.
flow
: the
' tier
ighly
ictive
f the
' very
eeded
. flow

uting
First,
e few
th to
essing
there
ich as
nents,

ith its
above
odel is
their
ine at
id I/O
stalls.
n fire.
" The
Since
ig due
des to
The

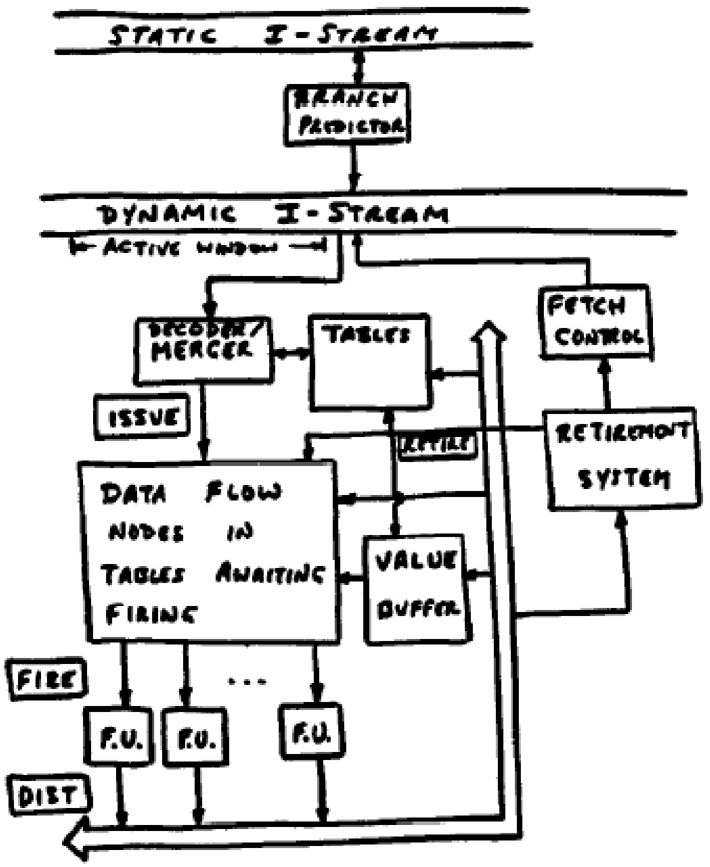


FIGURE 1.

data flow graph for the entire program is in the machi
at one time. We define the active window as the set
ISP instructions whose corresponding data flow nodes are

- Patt, Hwu, Shebanow, "HPS, A New Microarchitecture: Rationale and Initial Results"
- Work started in 1984
- Most importantly, I met Sabrina in 1984
- The vision is complete with the ISCA-92 paper by Yeh and Patt

HPS – from my vantage point

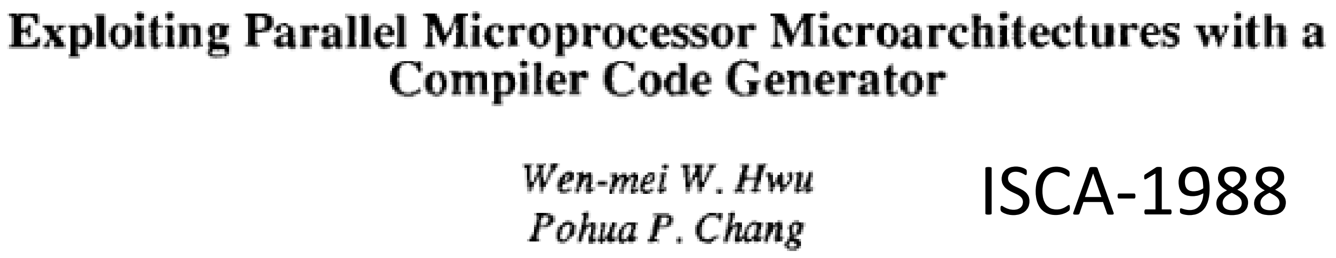
- Patt, Melvin, Hwu, Shebanow, “Critical Issues Regarding HPS, A High Performance Microarchitecture,” Micro-1985
- Hwu, Melvin, Shebanow, Chen, Wei, Patt, “An HPS Implementation of VAX; Initial Design and Analysis,” HICSS-1986
- Hwu, Patt, "HPSm, a High Performance Restricted Data Flow Architecture Having Minimal Functionality”, ISCA-1986
- Melvin, Hwu, Shebanow, Chen, Wei, Patt, “Run-Time Generation of HPS Microinstructions from a VAX Instruction Stream” Micro-1986
- Hwu, Patt, “Design Choices for the HPSm Microprocessor Chip, HICSS-1987
- Hwu, Patt, Checkpoint Repair for Out-of-order Execution Machines, ISCA-1987, IEEE TC

Emer (and Clark) - 1984

Smith (and Pleszkun) – 1985

Sohi (Vajapeyam) -1987

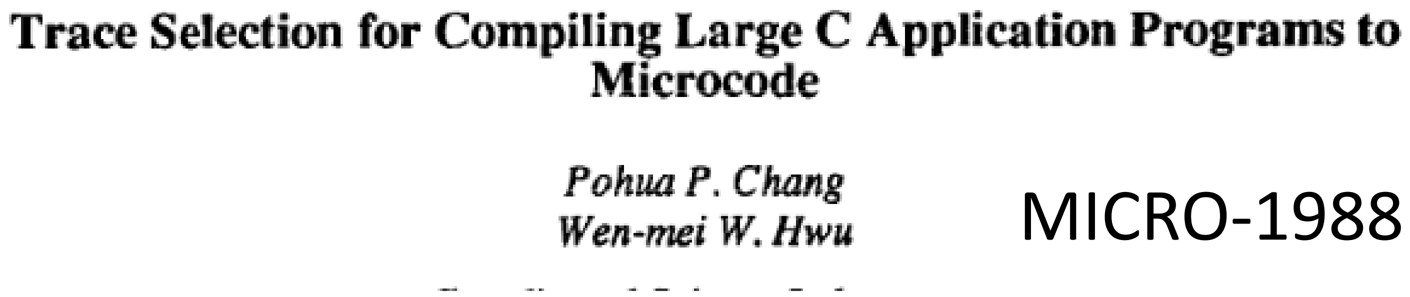
The Beginning of IMPACT



Exploiting Parallel Microprocessor Microarchitectures with a Compiler Code Generator

Wen-mei W. Hwu
Pohua P. Chang

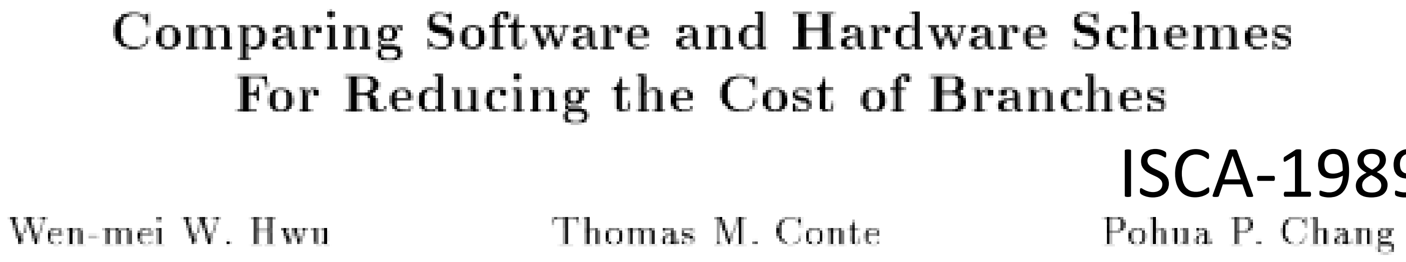
ISCA-1988



Trace Selection for Compiling Large C Application Programs to Microcode

Pohua P. Chang
Wen-mei W. Hwu

MICRO-1988



Comparing Software and Hardware Schemes For Reducing the Cost of Branches

ISCA-1989

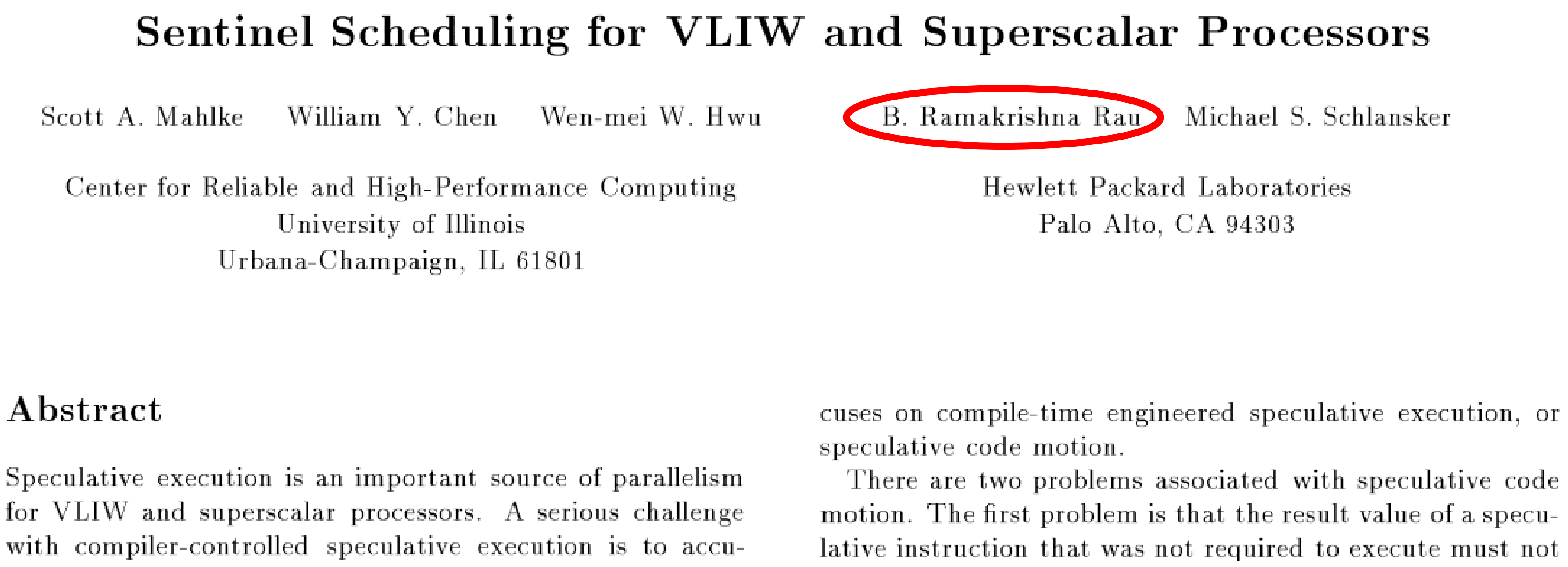
Wen-mei W. Hwu   Thomas M. Conte   Pohua P. Chang

IMPACT 1987-2001

- Chang, Mahlke, Chen, Warter, Hwu, "IMPACT: An Architectural Framework for Multiple-Instruction-Issue Processors.", ISCA-1991
- Gallagher, Chen, Mahlke, Gyllenhaal, Hwu, "Dynamic Memory Disambiguation Using the Memory Conflict Buffer.", ASPLOS-1994
- Mahlke, Hank, McCormick, August, Hwu, "A Comparison of Full and Partial Predicated Execution Support for ILP Processors.", ISCA-1995.
- Lavery, Hwu, "Unrolling-Based Optimizations for Modulo Scheduling.", MICRO-1995.
- August, Connors, Mahlke, Sias, Crozier, Cheng, Eaton, Olaniran, Hwu, "Integrated Predicated and Speculative Execution in the IMPACT EPIC Architecture.", ISCA-1998.
- Hwu, Sias, Merten, Nystrom, Barnes, Shannon, Ryoo, Olivier, "Itanium Performance Insights.", Microprocessor Forum, 2001.

Hennessy – 1987, Flynn – 1987 Davidson – 1987, Ebcioglu – 1987, Kuck – 1987, Belgard – 1985, Colwell – 1990, Rau – 1991, Fisher – 1992, Valero - 1994

ASPLOS-V (1992)



Sentinel Scheduling for VLIW and Superscalar Processors

Scott A. Mahlke William Y. Chen Wen-mei W. Hwu B. Ramakrishna Rau Michael S. Schlansker

Center for Reliable and High-Performance Computing
University of Illinois
Urbana-Champaign, IL 61801

Hewlett Packard Laboratories
Palo Alto, CA 94303

Abstract

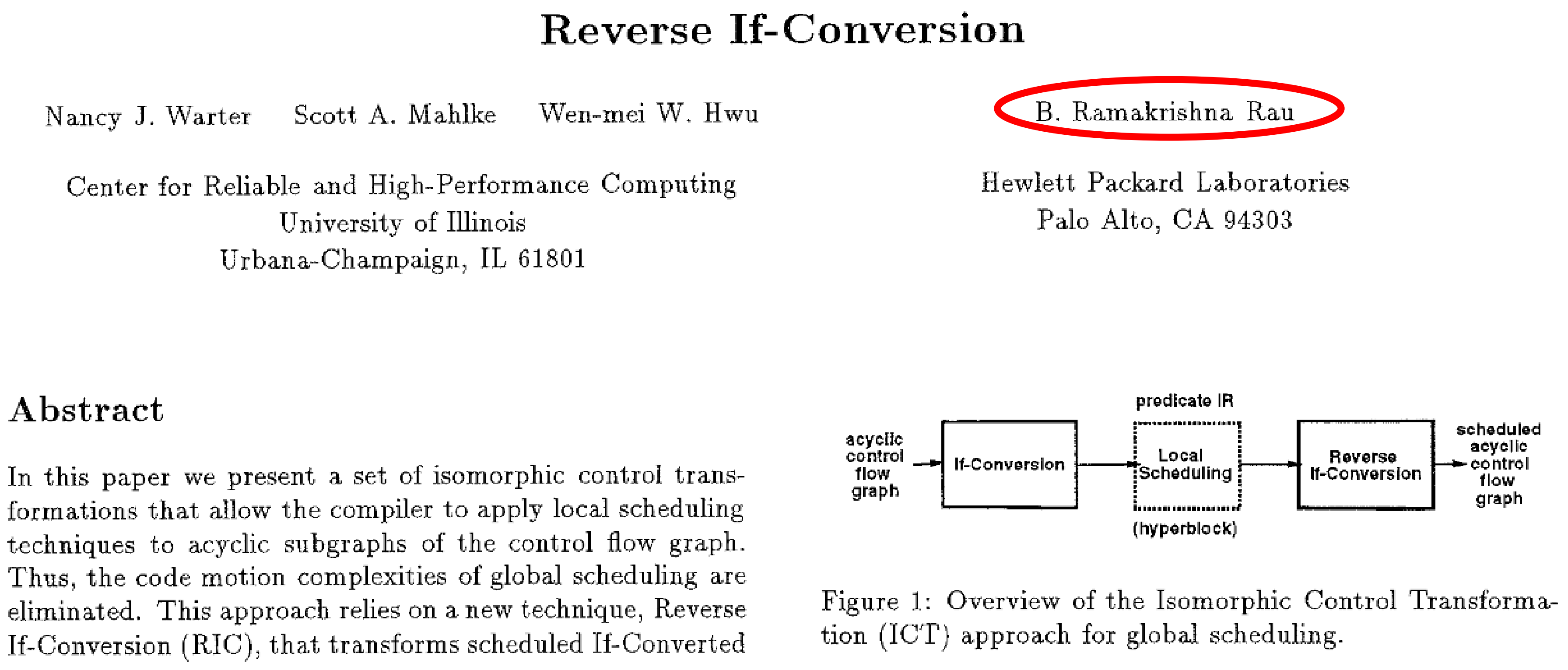
Speculative execution is an important source of parallelism for VLIW and superscalar processors. A serious challenge with compiler-controlled speculative execution is to accu-

cuses on compile-time engineered speculative execution, or speculative code motion.

There are two problems associated with speculative code motion. The first problem is that the result value of a speculative instruction that was not required to execute must not

Bob also convinced me to chair MICRO-25 in 1992.

PLDI 1993



Reverse If-Conversion

Nancy J. Warter Scott A. Mahlke Wen-mei W. Hwu

Center for Reliable and High-Performance Computing
University of Illinois
Urbana-Champaign, IL 61801

B. Ramakrishna Rau

Hewlett Packard Laboratories
Palo Alto, CA 94303

Abstract

In this paper we present a set of isomorphic control transformations that allow the compiler to apply local scheduling techniques to acyclic subgraphs of the control flow graph. Thus, the code motion complexities of global scheduling are eliminated. This approach relies on a new technique, Reverse If-Conversion (RIC), that transforms scheduled If-Converted

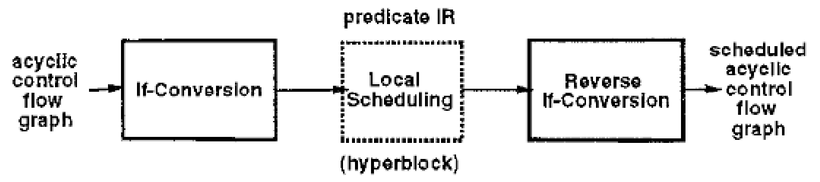
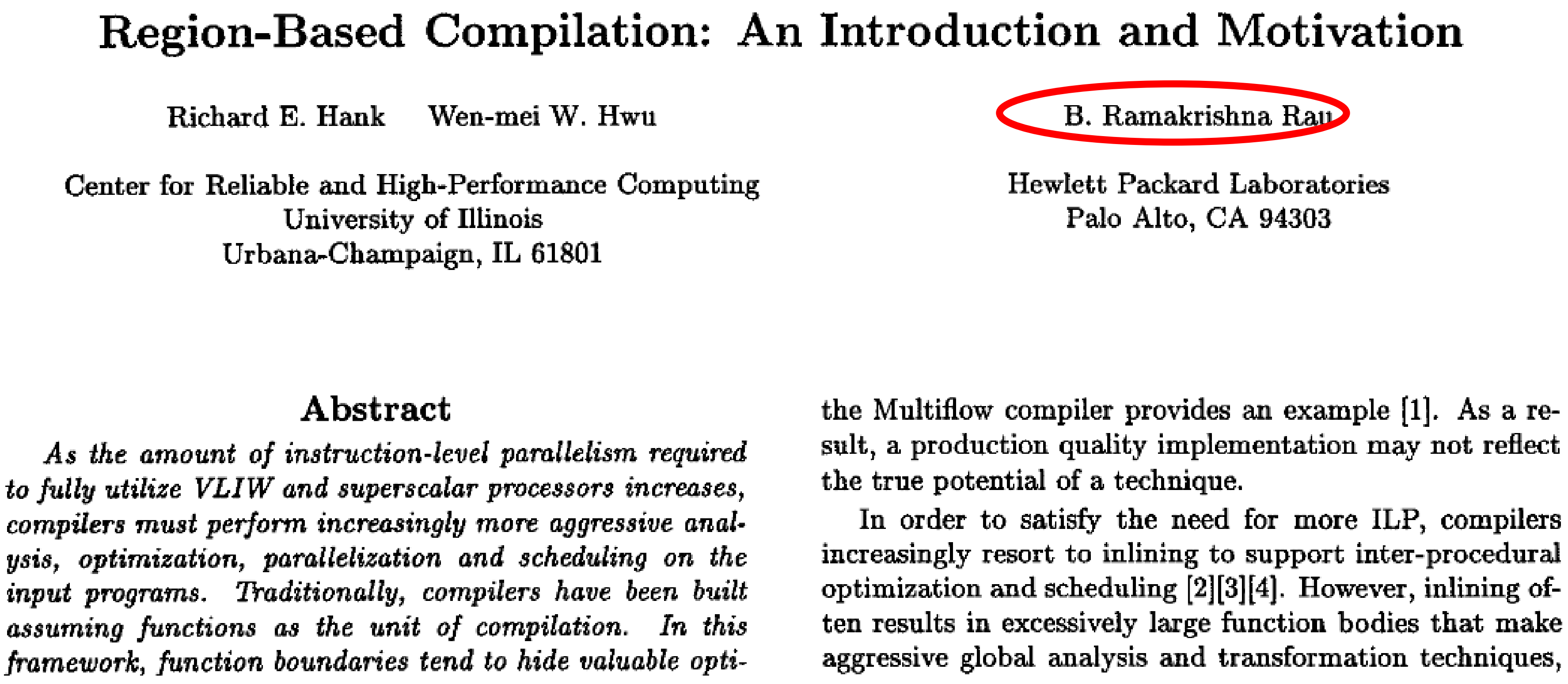


Figure 1: Overview of the Isomorphic Control Transformation (ICT) approach for global scheduling.

MICRO-28 (1995)



Region-Based Compilation: An Introduction and Motivation

Richard E. Hank   Wen-mei W. Hwu

Center for Reliable and High-Performance Computing
University of Illinois
Urbana-Champaign, IL 61801

B. Ramakrishna Rau

Hewlett Packard Laboratories
Palo Alto, CA 94303

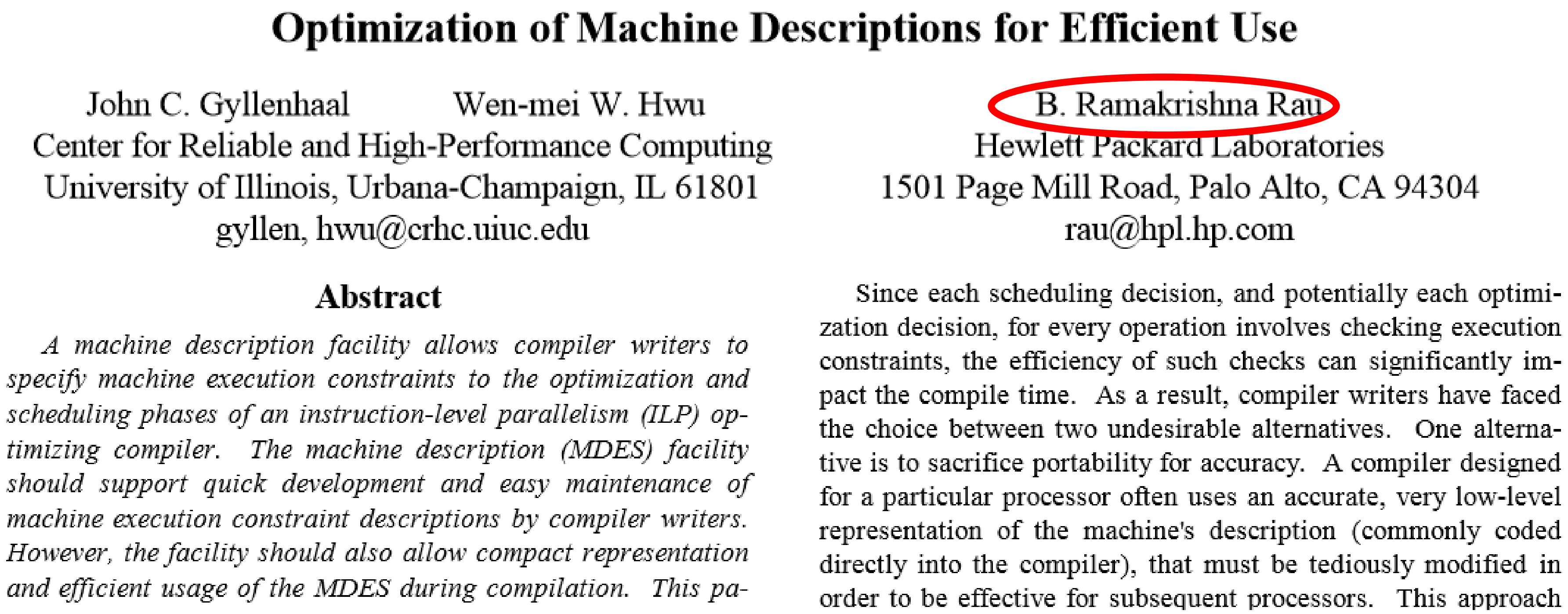
Abstract

As the amount of instruction-level parallelism required to fully utilize VLIW and superscalar processors increases, compilers must perform increasingly more aggressive analysis, optimization, parallelization and scheduling on the input programs. Traditionally, compilers have been built assuming functions as the unit of compilation. In this framework, function boundaries tend to hide valuable opti-

the Multiflow compiler provides an example [1]. As a result, a production quality implementation may not reflect the true potential of a technique.

In order to satisfy the need for more ILP, compilers increasingly resort to inlining to support inter-procedural optimization and scheduling [2][3][4]. However, inlining often results in excessively large function bodies that make aggressive global analysis and transformation techniques,

MICRO-29 (1996)



Optimization of Machine Descriptions for Efficient Use

John C. Gyllenhaal Wen-mei W. Hwu
Center for Reliable and High-Performance Computing
University of Illinois, Urbana-Champaign, IL 61801
gyllen, hwu@crhc.uiuc.edu

B. Ramakrishna Rau
Hewlett Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94304
rau@hpl.hp.com

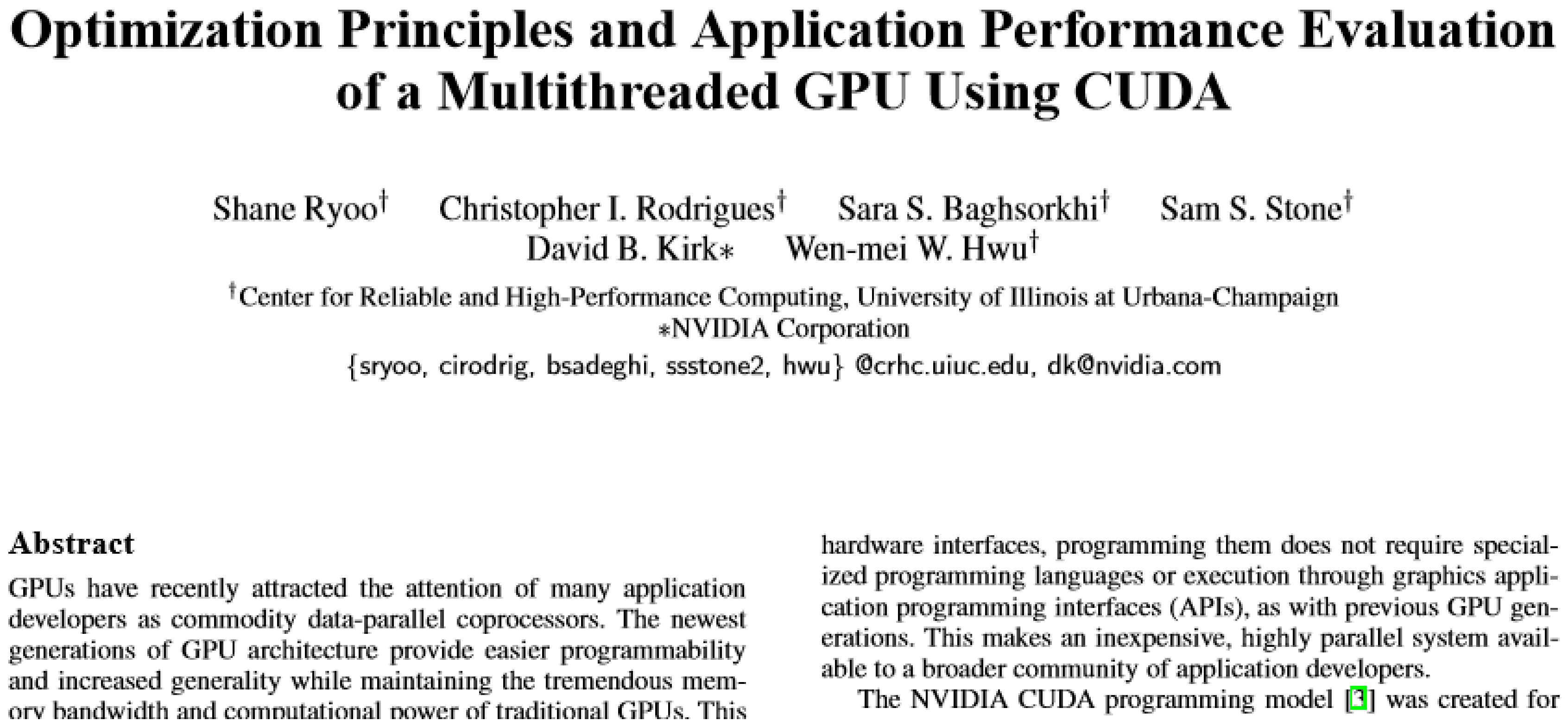
Abstract

A machine description facility allows compiler writers to specify machine execution constraints to the optimization and scheduling phases of an instruction-level parallelism (ILP) optimizing compiler. The machine description (MDES) facility should support quick development and easy maintenance of machine execution constraint descriptions by compiler writers. However, the facility should also allow compact representation and efficient usage of the MDES during compilation. This pa-

Since each scheduling decision, and potentially each optimization decision, for every operation involves checking execution constraints, the efficiency of such checks can significantly impact the compile time. As a result, compiler writers have faced the choice between two undesirable alternatives. One alternative is to sacrifice portability for accuracy. A compiler designed for a particular processor often uses an accurate, very low-level representation of the machine's description (commonly coded directly into the compiler), that must be tediously modified in order to be effective for subsequent processors. This approach

GPU Computing 2006-present

PPoPP-13, 2008



Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA

Shane Ryoo[†] Christopher I. Rodrigues[†] Sara S. Baghsorkhi[†] Sam S. Stone[†]
David B. Kirk[*] Wen-mei W. Hwu[†]

[†]Center for Reliable and High-Performance Computing, University of Illinois at Urbana-Champaign
[*]NVIDIA Corporation
{sryoo, cirodrig, bsadeghi, ssstone2, hwu} @crhc.uiuc.edu, dk@nvidia.com

Abstract

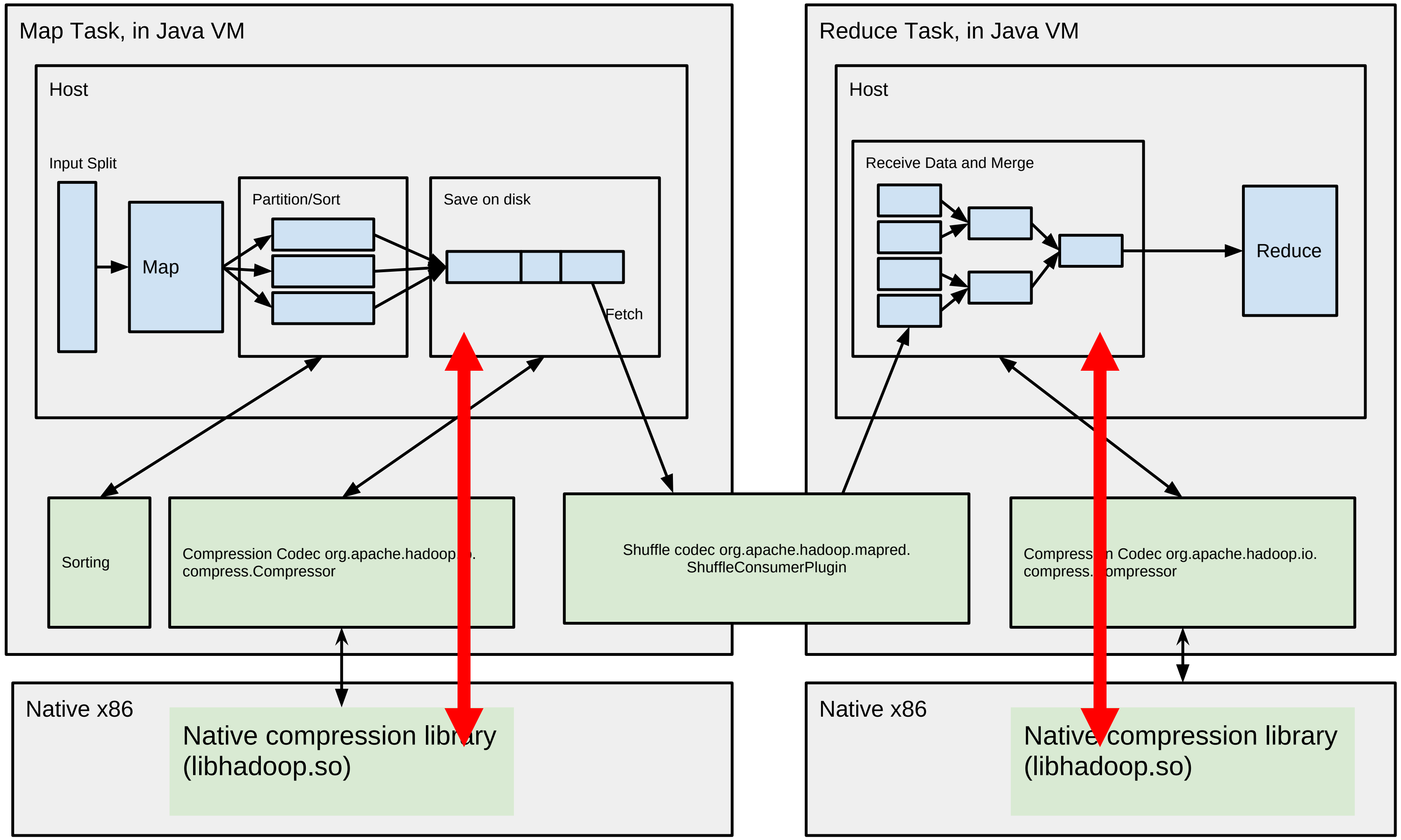
GPUs have recently attracted the attention of many application developers as commodity data-parallel coprocessors. The newest generations of GPU architecture provide easier programmability and increased generality while maintaining the tremendous memory bandwidth and computational power of traditional GPUs. This hardware interfaces, programming them does not require specialized programming languages or execution through graphics application programming interfaces (APIs), as with previous GPU generations. This makes an inexpensive, highly parallel system available to a broader community of application developers.

The NVIDIA CUDA programming model [3] was created for

Some Lessons from GPU Computing

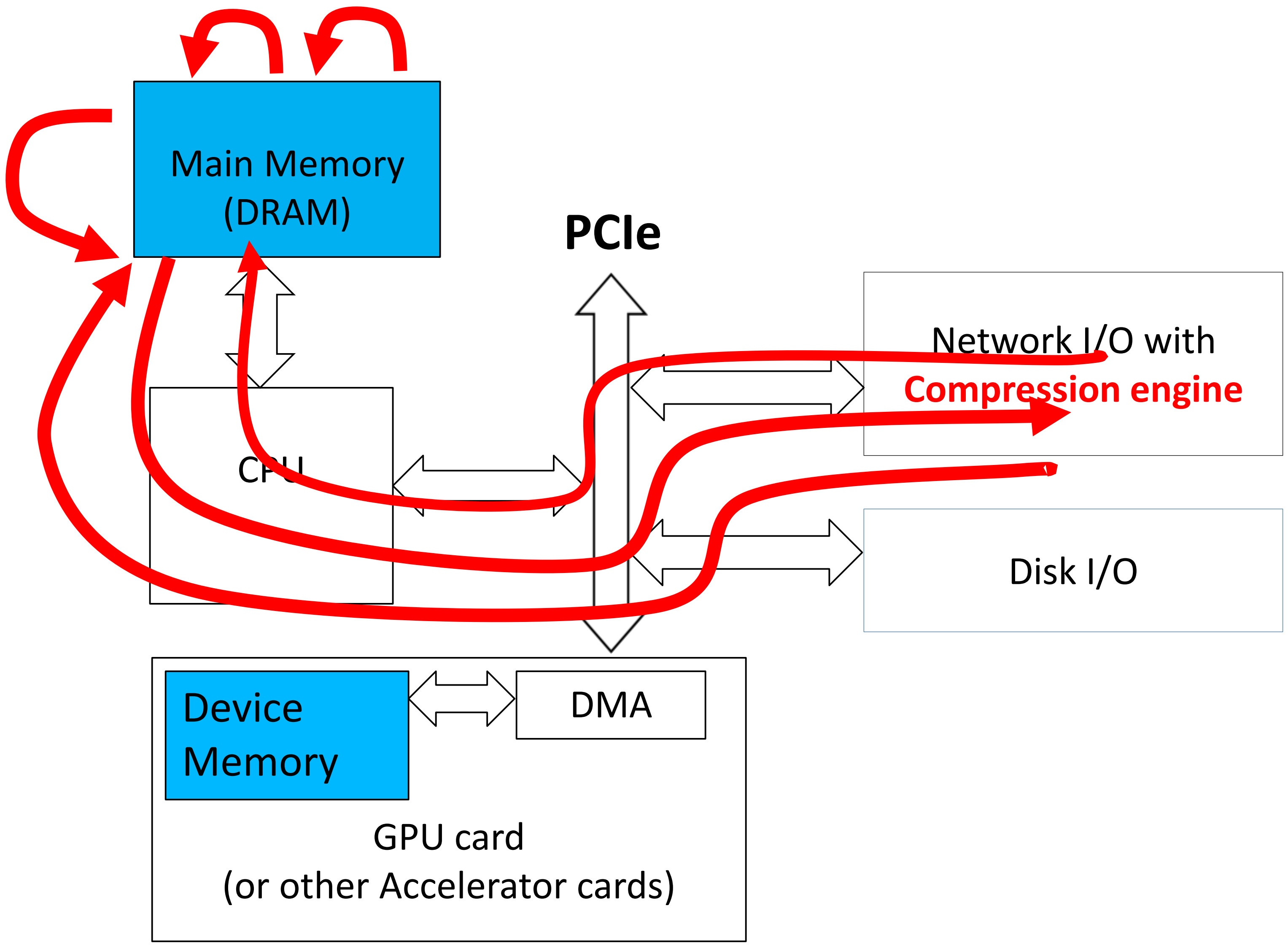
- Practical JIT eliminates traditional ISA as a major limiter of progress.
- Reduced clock frequency and power enabled continued performance scaling of valuable parallel applications
- Numerical libraries were badly lacking in parallelism before 2006 (e.g., Chang, et al 2012).
- CUDA/OpenCL programming is effective, useful but not for more than a few thousand programmers.
- Data transfer cost in the modern PC architecture badly limits the use of non-CPU compute devices.
- Software abstractions are poorly implemented in modern systems.

Software Abstraction Creates Barriers

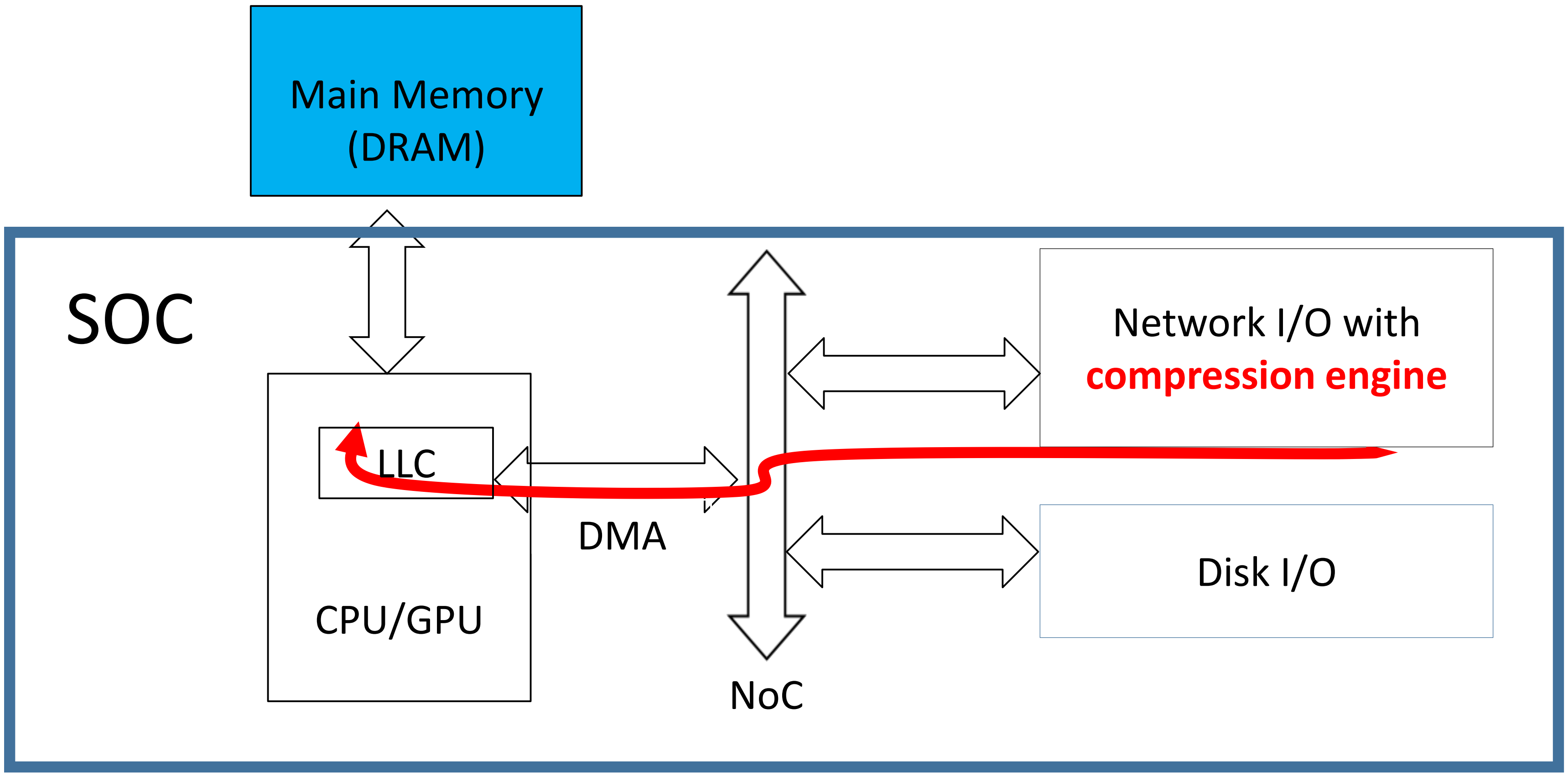

Map Task, in Java VM
Host
Input Split
Map
Partition/Sort
Save on disk
Fetch
Sorting
Compression Codec org.apache.hadoop.io.compress.Compressor
Shuffle codec org.apache.hadoop.mapred.ShuffleConsumerPlugin
Reduce Task, in Java VM
Host
Receive Data and Merge
Reduce
Compression Codec org.apache.hadoop.io.compress.Decompressor
Native x86
Native compression library (libhadoop.so)
Native x86
Native compression library (libhadoop.so)

Hwu

State of the Art Data Transfer Behavior



Desired SoC Data Transfer Behavior



Potential >30X savings in energy

A few thoughts for our early-career colleagues

- That is, everyone here except for Yale
  - Yale is mid-career
- Processor cores, accelerators, NICs, and storage controllers are the new data path components in the SOC era.
- Impactful microarchitecture research will likely need to involve OS and compiler innovations.
  - Java Bytecode is arguably the most important ISA
  - OS and VM implementations are long overdue for revamping.
- Performance/efficiency of many applications can be improved by ~100x by squeezing out inefficiency from the implementation stack.
  - Will likely take a decade or more to materialize in a scaling manner
  - Much of the progress will need microarchitecture support
  - Be radical!

To Bob Rau, a gentleman in every sense of the word.