Computer Communications and Networks

Series editor
A.J. Sammes
Centre for Forensic Computing
Cranfield University, Shrivenham Campus
Swindon, UK

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers, and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

More information about this series at http://www.springer.com/series/4198

Veljko Milutinovic • Jakob Salom • Dragan Veljovic
Nenad Korolija • Dejan Markovic • Luka Petrovic

DataFlow Supercomputing Essentials

Research, Development and Education

Springer

Veljko Milutinovic
University of Belgrade
Belgrade, Serbia

Jakob Salom
Serbian Academy of Sciences and Arts
Belgrade, Serbia

Dragan Veljovic
Motionlogic GmbH
Berlin, Germany

Nenad Korolija
University of Belgrade
Belgrade, Serbia

Dejan Markovic
University of Belgrade
Belgrade, Serbia

Luka Petrovic
University of Belgrade
Belgrade, Serbia

ISSN 1617-7975 ISSN 2197-8433 (electronic)
Computer Communications and Networks
ISBN 978-3-319-66127-8 ISBN 978-3-319-66128-5 (eBook)
DOI 10.1007/978-3-319-66128-5

Library of Congress Control Number: 2017951392

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

In general, there are two major approaches to classical computing: control flow and dataflow. In the control flow approach to computing, the program micro-controls the flow of data through the hardware. In the dataflow approach to computing, the program configures the hardware; ideally, it is the voltage difference between the input and output of the hardware that micro-controls the flow of data through the hardware in the theoretically ideal dataflow machines (in reality, for now, this theoretical ideal has not been achieved yet).

Dataflow Taxonomy

According to Flynn's taxonomy, control flow machines could be of the SISD type (single instruction and single data) and of the MIMD type (multiple instructions and multiple data). The MIMD type is basically subdivided into multicores (e.g., Intel) and manycores (e.g., NVidia) and WSN (wireless sensor networks). The control flow approach is used in almost all computers of today.

According to taxonomies used in dataflow literature, the dataflow concept could be defined on a number of different levels of abstraction related to computing. The three levels relevant for this book are the software level (application software), the hardware level (digital hardware), and analog levels used for implementation of digital hardware (transistors and wires).

On the software level, the concept exists for a long time now, but became extremely popular only recently, since Google announced that it gave up MapReduce and switched to dataflow. Software dataflow is not a subject of this book. For the reader of this book, it is only important to understand that a number of different software approaches to dataflow could be defined and that the best hardware support for dataflow in software is dataflow in hardware.

On the hardware level, the concept exists for decades now, in three basic forms: (a) general purpose dataflow related to the early research at MIT, (b) special purpose

dataflow related to the early research in systolic arrays, and (c) the dataflow that accelerates loops, which is applicable to both special purpose and general purpose computing and whose effectiveness depends on the contribution of loop execution times to the overall execution time of the application or the multiscale dataflow (MSDF). The first and the second approach (a and b) are outside the scope of this book. The third approach (MSDF) offers the best potentials and is the focus of this book.

Maxeler

The MSDF approach could be implemented using (a) FPGAs (Altera or Xilinx or any other FPGA vendor) or (b) aFPGAs (asynchronous FPGAs) (a research concept exemplified by recent research at Caltech, Cornell, Yale, and others) or (c) SoG (sea-of-gates) that enables the most effective mapping of execution graphs onto the hardware infrastructure underneath, which combines analog communications and digital storage.

Examples in this book are based on the Maxeler MSDF concept. The Maxeler machines are currently implemented using FPGAs. However, Maxeler is not an FPGA company. It uses FPGAs as the least bad hardware infrastructure of all those currently available for the implementation of the MSDF concept. Therefore, FPGAs, looking from the Maxeler viewpoint, are a "necessary evil" that has to be used until a more appropriate hardware support reaches the production level. Although FPGAs do not allow the Maxeler MSDF concept to blossom fully, they are tolerated, since they, in spite of all their drawbacks, do allow the Maxeler concept to demonstrate superiority over the existing control flow approaches, mostly in the amount of computation per cubic foot and Watt.

The Maxeler MSDF concept covers relevant issues in three different aspects of software: (a) operating system, (b) compiler, and (c) application software. The first two issues (a and b) are outside the scope of this book. The focus of this book is on the research that could improve the overall concept and on the applications thereof.

Research Issues

The major inspiration for future research in dataflow computing of the MSDF type comes from the heritage of four different Nobel laureates. Their heritage includes not only scientific results but also observations that torch the light onto possible future research avenues.

Richard Feynman, in his book on lecture notes in computer science, commented that speed and power could be traded and also shared the observation that arithmetic and logic could be performed with power that converges to zero, while communi-

cating the results to the next usage point in memory or in another processor does consume power that could diverge to infinity for NP complete problems.

These observations mean that the ultimate goal of computer architects is to come up with an architecture that possibly does not move data at all, or if it has to move data around, which is unavoidable since the dimensions of arithmetic and logic units are non-zero, then the movements should go only over the shortest possible distances.

Ilya Prigogine, in his book on chaos, commented that injecting energy could decrease entropy of a system, which could be understood as the reminder of the fact that one could also define the entropy of a computing system, meaning that the lower entropy of such a system creates better conditions for the system software to do optimizations more effectively.

These observations led to the conclusion that the energy spent in separation of temporal and spatial data could help both the speedups and power reductions. Temporal data are those that get revisited over and over again (loop control variables, counter control variables, semaphores, etc.). Spatial data are those that get traversed, one by one or in strides, such as 4 by 4 and the like (vectors, matrices, complex data structures of any kind, etc.). If temporal and spatial data are separated, then a hybrid approach based on control flow and dataflow could become very effective, if the cranking of temporal data is somehow attributed to control flow and of spatial data to dataflow.

Daniel Kahneman, in his book on thinking fast and thinking slow, indicated about the importance of approximate computing. One possible explanation of the importance of approximate computing is through chess analogy. In chess, from time to time, we sacrifice something that is of importance, but not of ultimate importance, like the tower or queen, and by doing that we establish potential energy that enables us to achieve what is of ultimate importance—to win the game. In computing, that means that from time to time we can do computation with lower precision, which enables us to release resources that could be reinvested into a benefit that is of a much higher importance.

These observations bring one to a conclusion that the wisdom of Daniel Kahneman could and should be built into the compiler of dataflow machines, together with a set of mechanisms that enable one to select the tradeoffs of interest.

Andre Geim, in his research related to thin films, studied the relationships between latency (the speed at which a process could be done) and precision (with which the process achieves the final results). An interested reader could port these conclusions into the world of computing.

These observations, if implemented in the compiler of a dataflow machine, could enable users to trade latency and precision at compile time, which could be especially of importance for applications that require low latency, like those in online trading and those in defense.

Application Issues

One can port an application from control flow to dataflow by applying rules, but that will not bring major speedups, especially if the contribution of loops is negligible and the level of data reusability within loops is relatively low.

One can play with issues like input data choreography, changes or orders of operations without jeopardizing the integrity of results (commutation, association, distribution), changes in the floating-point precision or precision in general, and on the wise utilization of internal pipelines that the compiler generates, in the effort to create a consistent execution graph.

One can compare existing algorithms for the same application. Sometimes, the algorithm which is the best in control flow is the worst in dataflow and vice versa. One such example is sorting. Bitonic sorting is the worst in control flow and the best in dataflow.

One can think of new algorithms for new applications that did not exist before. Creating a new algorithm from scratch is a challenge, and it can turn into a success only and especially if it is done by an experienced dataflow programmer.

Conclusion

What was the goal and what was achieved? The goal was to underline what are the basic potentials for dataflow computing and to explain what types of research would lead to the ultimate goals: high speed, low power, high precision, and small sizes of computing equipment.

To whom may it be of benefit? The above is of benefit to those who plan to invest into further developments of dataflow computing, as well as to those who are eager to select the topics for their future research.

What are the new open research problems? The new goals could be classified as: (a) developing a new type of sea-of-gates that would enable a more effective mapping of the execution graphs onto the hardware infrastructure, (b) developing new aspects of the operating system that would enable the dataflow paradigm to become more effective, (c) implementing into the compiler all the wisdoms of the four Nobel laureates (Feynman, Prigogine, Kahneman, and Geim)who served as the major sources of inspiration, and (d) working on new algorithms for existing and new applications that would enable the paradigm to demonstrate its full strengths (appgallery.maxeler.com).

Belgrade, Serbia Veljko Milutinovic
July 2017

Contents