Computer Architecture for Scientists

The dramatic increase in computer performance has been extraordinary, but not for all computations: it has key limits and structure. Software architects, developers, and even data scientists need to understand how to exploit the fundamental structure of computer performance to harness it for future applications.

Using a principled approach, *Computer Architecture for Scientists* covers the four key pillars of computer performance and imparts a high-level basis for reasoning with and understanding these concepts. These principles and models provide approachable high-level insights and quantitative modeling without distracting low-level detail. The pillars include:

- Small is fast: how size scaling drives performance.
- Hidden parallelism: how a sequential program can be executed faster with parallelism.
- Dynamic locality: skirting physical limits, by arranging data in a smaller space.
- Explicit parallelism: increasing performance with teams of workers.

Finally, the text covers the GPU and machine-learning accelerators that have become important for more and more mainstream applications. Ideal for upper-level undergraduates.

Andrew A. Chien is the William Eckhardt Professor at the University of Chicago, Director of the CERES Center for Unstoppable Computing, and a Senior Scientist at Argonne National Laboratory. Since 2017, he has served as Editor-in-Chief of the Communications of the ACM. Chien is a global research leader in parallel computing, computer architecture, clusters, and cloud computing, and has received numerous awards for his research. In 1994 he was named an National Science Foundation Young Investigator. Dr. Chien served as Vice President of Research at Intel Corporation from 2005 to 2010, and on advisory boards for the National Science Foundation, Department of Energy, Japan RWCP, and distinguished universities such as Stanford, UC Berkeley, EPFL, and the University of Washington. From 1998 to 2005, he was the SAIC Chair Professor at UCSD, and prior to that a professor at the University of Illinois. Dr. Chien is a Fellow of the ACM, Fellow of the IEEE, and Fellow of the AAAS, and earned his PhD, MS, and BS from the Massachusetts Institute of Technology.

Computer Architecture for Scientists

Principles and Performance

ANDREW A. CHIEN
University of Chicago

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/highereducation/isbn/9781316518533
DOI: 10.1017/9781009000598

First published 2022

Printed in the United Kingdom by TJ Books Limited, Padstow, Cornwall, 2022

A catalogue record for this publication is available from the British Library.

ISBN 978-1-316-51853-3 Hardback

Additional resources for this publication at www.cambridge.org/chien

For my father, who inspired a deep love of science, technology, and teaching, and for my mother, who inspired a deep joy of life

Contents

Preface

Because of computing's history, the pedagogy of computer architecture has been aimed at future computer designers and engineers. However, a growing number of data scientists, engineers who are not computer engineers, and even the majority of computer scientists (i.e. artificial intelligence and more) see computing as an intellectual tool. These scientists and engineers need an understanding of computer architecture for insights into how hardware enables, shapes, and limits performance. They need an understanding of computer architecture that allows them to reason about performance today and scaling into the future. It is for them that I have undertaken this book.

In fall 2011 I joined the University of Chicago faculty and taught undergraduate computer architecture. As the quarter progressed, it became clear that my students had different backgrounds, motivations, and interests – a marked contrast to the engineering students I taught at the University of Illinois and University of California, San Diego. Over the course of several quarters, I realized that the traditional bottom-up approach to teaching computer architecture (from gates to sequential circuits, from instruction sets to pipelines and caches – with an emphasis on how it works and how to optimize it) was not reaching these students. This traditional pedagogy is designed for computer engineering students, and increasingly even many computer science students do not find it compelling.

The University of Chicago students were interested in the "scientific principles" of computer architecture, principles that would enable them to reason about hardware performance for higher-level ends. Their view of computation was as an intellectual multiplier, and they were interested in a top-down view of capabilities, scaling, and limits – not mechanisms. Their point of view reflects computing's broad expansion, most recently manifest in artificial intelligence and data science, that has shifted computer science's center of mass upward and its boundary outward – into a wide variety of sciences (physical, biological, and social) as well as nearly every aspect of society, commerce, and even government.

For these data science, artificial intelligence, and applied machine learning students, a foundation in computer architecture should provide scientific understanding of how the remarkable power of computing is possible, and what limits that power – and, as we look forward to the future, how the continued progress in technology will

increase computing capabilities. This book provides such a principles-based grounding in the current and future capabilities of computing hardware. It is designed for several different curricula and settings:

- A quarter-long course in a Computer Science program in a liberal arts college or Data Science program: Chapters 1–3, 5, and 7. Depending on the curricular fit, perhaps add parts of microarchitecture (Chapter 4), framing CPUs as practically universal (Chapter 6), and accelerators (Chapter 8).
- A semester-long course in a Computer Science program in a liberal arts or Data Science program: Chapters 1–5, 7, and 9. If there is time, perhaps add back the hot topic of accelerators (Chapter 8).
- A semester-long course in a Math–Computer Science or Data Science program at an institution with a deep engineering culture: Chapters 1–9. This includes the full text.

I would like to thank the students in my "Computer Architecture for Scientists" course at the University of Chicago for their contributions – their thoughtful questions and responses were critical to increasing the clarity of the text, lectures, and exercises.

Finally, a heartfelt thanks to my family, including my wife, Ellen, and my two daughters, Jennifer and Athena. For their steadfast support for my writing this book and my absence on many late nights, I am grateful. Without your patience, tolerance, and encouragement, it would have been impossible. Thank you!

I am indebted to the University of Chicago, whose spirit of innovative scholarship and pedagogy has been both an inspiration and support for this book!