Preparing for the Future of Computing

Pieter Hijma
p.hijma@uva.nl

Friday 15 April 2016

Coffee & Data on Infrastructures
Broaden the view

Anticipate risks

Become aware of current issues
In coming years, software will have to adapt to hardware more than previously: **Software Crisis**
My background

- PhD. on programming many-cores
- High-performance computing: front-runner
Look into the Past

Moore's law

- # transistors vs. clockspeed (MHz)
- 1970: 8008
- 1975: 8086
- 1980: 8088
- 1985: 80386
- 1990: 80486
- 1995: Pentium
- 2000: Pentium II
- 2005: Pentium III
- 2010: Pentium 4
- 2015: Core 2

- 2020: Core i7 (Nehalem)
- 2025: Core i7 (Sandy)
- 2030: Core i7 (Haswell)
Look into the Past

Moore's law

- # transistors
- clockspeed (MHz)

- 8008
- 8088
- 8086
- 80286
- 80386
- 80486
- Pentium
- Pentium II
- Pentium III
- Pentium IV
- Core 2
- Core i7 (Nehalem)
- Core i7 (Sandy)
- Core i7 (Haswell)

- 1970
- 1975
- 1980
- 1985
- 1990
- 1995
- 2000
- 2005
- 2010
- 2015

- 10^3
- 10^4
- 10^5
- 10^6
- 10^7
- 10^8
- 10^9
- 10^10
Look into the Past

Moore's law

# transistors
clockspeed (MHz)

single-core era
Look into the Past

Moore's law

# transistors vs. clockspeed (MHz)

- 8008
- 8088
- 8080
- 8086
- 80286
- 80386
- 80486
- Pentium
- Pentium II
- Pentium III
- Pentium 4
- Core 2
- Core i7 (Nehalem)
- Core i7 (Sandy)
- Core i7 (Haswell)

lucky time
Look into the Past

Moore's law

- # transistors
- clockspeed (MHz)

multi-core era
Processor types

- Single-core
  - Optimized for latency
- Multi-core
  - Still optimized for latency, but just more than one
- Many-core
  - Optimized for throughput
  - High performance/Watt
Performance per Watt

- 1972 - 2007:
  - 10,000 $\times$ higher performance
  - 300 $\times$ higher efficiency
- 2005 - 2015
  - 250 $\times$ higher performance
  - 10 $\times$ higher efficiency
Performance per Watt

Performance and Power efficiency of #1 TOP500

Performance (TFLOPS) - Purple Line
Efficiency (MFLOPS/W) - Green Line

Yearly Performance and Efficiency Chart
Many-core processors

features

• throughput oriented
• fast evolution of the architecture
• architectural features for high performance

Difficult to program, especially for high-performance
Walls

- memory wall
- energy wall

result

hardware without compromises to the interface to programmers $\rightarrow$ difficult to program $\rightarrow$

- programming wall
Recap

To deal with energy problems hardware will be:

- highly parallel
- throughput oriented
- architectural details for performance
- difficult to program

Result

More responsibility for software developers
Soft-/hardware

Traditional view

- software is soft, we can change it easily
- hardware is hard, we cannot change it
Soft-/hardware

Traditional view

• software is soft, we can change it easily
• hardware is hard, we cannot change it

New view

• software is expensive, big, long-lived and therefore hard, changing it is difficult
• hardware is inexpensive, short-lived and therefore soft, changing hardware is easy
Soft-/hardware

Traditional view

• software receives less attention
• hardware receives much attention
• compiler can solve your performance problems

New view

• software should gain more recognition
• to adapt to future hardware, software has to become soft again
• compiler cannot help you as much
Examples

Parallel Ocean Program

- written in ’90s
- different context: memory cheap, compute expensive

Linpack

- we still evaluate supercomputers with the same software
- not representative anymore
How to prepare for this?

Abstractions

- Programmer uses high-level abstractions
- Compiler optimizes and translates to lower levels for computer
- Hardware extracts (limited) parallelism

This works reasonably well with:

- a stable interface to hardware
- limited parallelism on a processor
How to prepare for this?

In the many-core era?

- Interface to hardware not stable
- Highly parallel, parallelism is explicit
- Compilers are limited and cannot adapt fast enough

Abstractions

- Also hide the architectural details necessary for performance/efficiency
A program is an algorithm mapped to hardware

Program

Algorithm

Mapping

Hardware

Solution
Incorporate hardware descriptions in the programming model
Hierarchy of hardware descriptions

portability

perfect

mic  gpu

xeon phi  nvidia  amd

fermi  kepler

gtx480  gtx680

performance control
Role compiler: advisory

Compilers

- conservative
- no application knowledge

Programmers

- application knowledge
- can oversee correctness problems better
Optimization process recorded

Feedback
Using 1/8 blocks per smp. Reduce the amount of shared memory used by storing/loading shared memory in phases.
Making software soft again

Marver
Semi-automatically transform legacy programs

- Ben van Werkhoven
Overview of people involved

Henri Bal
High Performance Distributed Computing

Cees de Laat
System and Network Engineering

Ana Varbanescu
Performance Prediction

Ben van Werkhoven
Applications

Ismail El-Helw
Programming Models

Alessio Sclocco
Radio Astronomy

Me
Programming Models

Merijn Verstraaten
Graph Processing

Clemens Grelck
High-level Programming

Raphael Poss
Hard-/software Interface

Souley Madougou
Performance Prediction

Catalin Ciobanu
Reconfigurable Computing

VU UNIVERSITY OF AMSTERDAM
Conclusion

In coming years, software will have to adapt to hardware more than previously: **Software Crisis**