An Ecosystem for Combining Performance and Correctness for Many-Cores

Pieter Hijma
pieter@cs.vu.nl
Friday 16 May 2018

Fourth NIRICT GPGPU Reconnaissance Workshop
In coming years, software will have to adapt to hardware more than previously.

Our solution: An Ecosystem for Combining Performance and Correctness for Many-Cores
Look into the Past

Moore's law

# transistors

Moore's law

10^3
10^4
10^5
10^6
10^7
10^8
10^9
10^10


10086 100286 100386 100486 Pentium II
Pentium III
Pentium 4
Core i7 (Nehalem)
Core i7 (Sandy)
Core i7 (Haswell)
Core i7 (Broadwell)

Pentium II
Pentium III
Pentium 4
Core i7 (Nehalem)
Core i7 (Sandy)
Core i7 (Haswell)
Core i7 (Broadwell)
Look into the Past

Moore's law

# transistors

clockspeed (MHz)

103


10

100

1000

10000

2/28
Look into the Past

Moore's law

Single-core era

# transistors

clockspeed (MHz)

10^10
10^9
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10
0.1
1
10
100
1000
10000

Look into the Past

Moore's law

- # transistors vs. clockspeed (MHz)
- Core 2
- Core i7 (Nehalem)
- Core i7 (Sandy)
- Core i7 (Haswell)
- Core i7 (Broadwell)

Lucky time
Moore's law

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

Performance and Power efficiency of #1 TOP500

Performance (TFLOPS)  
Efficiency (TFLOPS/W)  

Many-core processors

features

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

Difficult to program, especially for high-performance
Future processors
Moore's Law

Moore's law graph showing the number of transistors over time from 1970 to 2015. The x-axis represents the years 1970 to 2015, and the y-axis represents the number of transistors in a logarithmic scale. Key processors like pentiums and i7s are marked on the graph.
Moore’s Law ending

Lithography over years

Manufactoring process (nm)
Walls

- energy wall
- memory wall
- Moore’s law $\rightarrow$ Moore’s wall

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

- programming wall
Large demand for computational power

Chemistry

- in vitro $\rightarrow$ in silico

Machine Learning

- Shooting with a computational cannon

Increase in data to process

- For example gene-sequence alignment
Increase in data

Moore’s law
Moore’s law against the SRA genetic database.
Many-core era

- window of 5-10 years to figure out:
  - what hardware is going to look like
  - how to program for performance well
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
- Increase in performance relies on software
Ecosystem for Performance and Correctness

- clusters of many-cores
- obtain high performance
- understanding performance
- correctness with model checking

Diagram:

- Application
- MCL
- Cashmere
- Constellation
A program is an algorithm mapped to hardware

Solution
Incorporate hardware descriptions in the programming model
Hierarchy of hardware descriptions
Stepwise-refinement for performance

Feedback
Using 1/8 blocks per smp. Reduce the amount of shared memory used by storing/loading shared memory in phases
Model checking: mCRL2

- effective tool for software flaws
- support rich data structure
- versatile
  - memory access problems
  - correctness of optimizations

Goals

- non-intrusive
- feed back verified properties into the compiler for optimization
Performance-correctness co-refinement

extract model
check property $p$

check equivalence

extract refinement of model
check property $p$

check equivalence

extract refinement of model
check property $p$

check equivalence
Accelerating Verification

- exploit symmetry in many-core programs
- use many-cores to accelerate model checking
  - accelerate the term-rewriting core in mCRL2
Many-core cluster computers

- Supports heterogeneous many-core clusters
- Can handle large-scale applications
- Excellent load balancing and scalability
## Scalability results forensics application

The table below shows the scalability results for different data sets and devices:

<table>
<thead>
<tr>
<th>name data set</th>
<th>Pentax</th>
<th>Praktica</th>
<th>Olympos</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of images</td>
<td>638</td>
<td>1095</td>
<td>4980</td>
</tr>
<tr>
<td>#jobs</td>
<td>2075</td>
<td>1128</td>
<td>73920</td>
</tr>
<tr>
<td>time 1 node</td>
<td>47m 14s</td>
<td>44m 44s</td>
<td>53h 25m</td>
</tr>
<tr>
<td>time 16 nodes</td>
<td>2m 55s</td>
<td>3m 16s</td>
<td>3h 10m</td>
</tr>
</tbody>
</table>
Load balancing
Visualizing kernel execution

Hardware descriptions designed such that they can be drawn:
Bioinformatics application

Motif-aware multiple sequence alignment

A

>Sequence1
CATGGGTA

>Sequence2
CTGCTGGTA

CLUSTAL 2.1 multiple sequence alignment

<table>
<thead>
<tr>
<th>Sequence1</th>
<th>CATG----CGGTA 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence2</td>
<td>CTGTCGTA 12</td>
</tr>
</tbody>
</table>

* *** * ****

B

1
CATGGGTA
CATGGGTA
CTGCTGGTA
ATGCTGGGTA

2
CαββαCGGTA
CαβγββαGTA
αβαββαCGGTA
AαβγββαCGGTA

3

4


C

A C G T
A 1
C 0 1
G 0 0 1
T 0 0 0 1

A C G T
A 1
C 0 1
G 0 0 1
T 0 0 0 1

α β γ

0 0 0 1 MMW
0 0 1 0 MSW MMW
0 1 0 0 MSW MSW MMW
Natural Language Processing application

Word embeddings

- Map words to vectors or real numbers (word2vec)
- Take large corpus, create large multi-dimensional vector space
In coming years, software will have to adapt to hardware more than previously.

Our solution: An Ecosystem for Combining Performance and Correctness for Many-Cores
FPGAs

- high-level synthesis: as successful as automagically parallelizing compilers
- only for:
  - extremely low latency applications
  - extremely power efficient
  - prototyping hardware
- tools are of low quality