A Proposal for Autotuning Linear Algebra Routines on Multicore Platforms

Javier Cuenca, Luis P. García
Domingo Giménez

Parallel Computing Group
University of Murcia, SPAIN
Introduction

- The arrival of multicores: parallel computing more available to the scientific groups
- To describe parallelism to the compiler: OpenMP, a directive-based syntax
- Problem:
  - there are no standardized OpenMP compilers
  - number of threads → the performance greatly varies
- Our goal: a Poly-Compilation Engine (PCE)
  - generates different executables of each routine (one for each compiler in the system)
  - selects the executable which best fits the problem characteristics
Outline

- Introduction

- Our approach: Poly-Compilation Engine (PCE)
  - PCB: Benchmarking Routines of the PCE
  - Proof of concept of the PCE

- Conclusions
Our approach: Poly-Compilation Engine (PCE)

BR source

X source

PCE
Our approach:
Poly-Compilation Engine (PCE)

Compiler A
BR source

Compiler B

BR\(_A\)
BR\(_B\)
PCB

PCE

X source

A Proposal for Autotuning Linear Algebra Routines on Multicore Platforms
Our approach: Poly-Compilation Engine (PCE)

 Compiler A

 BR source

 Compiler B

 PCB

 \[ \text{BR}_A \quad \text{BR}_B \]

 \[ \text{SP}_A \quad \text{SP}_B \]

 other SP

 SP

 X source

 PCE
Our approach: Poly-Compilation Engine (PCE)

Compiler A
BR source

Compiler B

Compiler A
X source

Compiler B

BR\textsubscript{A}

BR\textsubscript{B}

PCB

SP\textsubscript{A}

SP\textsubscript{B}

other SP

SP

X\textsubscript{A}

X\textsubscript{B}

PCE
Our approach: Poly-Compilation Engine (PCE)
Our approach: Poly-Compilation Engine (PCE)
Our approach: Poly-Compilation Engine (PCE)

- **BR source**
  - Compiler A
  - Compiler B

- **X source**
  - Compiler A
  - Compiler B

- **Decision Engine**
  - **BR_A**
  - **BR_B**
  - **X_A**
  - **X_B**
  - **SP_A**
  - **SP_B**

- **PCB**
  - other SP

- **PCE**
  - problem size
  - X model

- **AP**
  - (version of X to use, number of threads, other AP)
Outline

- Introduction

- Our approach: Poly-Compilation Engine (PCE)
  - PCB: Benchmarking Routines of the PCE
  - Proof of concept of the PCE

- Conclusions
To test the efficiency of OpenMP primitives for threads creation and management

- **R-generate**
  - Creating a series of threads with a fixed quantity of work to do per thread
  - To compare the time of creating and managing threads

- **R-pfor**
  - A simple for loop where there is a significant work inside each iteration
  - To compare the time of distributing dynamically a set of homogeneous tasks

- **R-barriers**
  - A barrier primitive set after a parallel working area
  - To compare the times to perform a global synchronization of all the threads
PCB: Benchmarking Routines of the PCE

- Experiments: Multicore platforms + compilers:
  - **P2c**
    - Intel Pentium, 2.8 GHz, with 2 cores.
    - Compilers: icc 10.1 and gcc 4.3.2.
  - **A4c**
    - Alpha EV68CB, 1 GHz, with 4 cores.
    - Compilers: cc 6.3 and gcc 4.3.
  - **X4c**
    - Intel Xeon, 3 GHz, with 4 cores.
    - Compilers: icc 10.1 and gcc 4.2.3.
  - **X8c**
    - Intel Xeon, 2 GHz, with 8 cores.
    - Compilers: icc 10.1 and gcc 3.4.6
PCB: Benchmarking Routines of the PCE
R-generate

P2c

icc
gcc

Threads

time (seconds) (log. scale)

A4c

cc
gcc

Threads

time (seconds) (log. scale)

X4c

icc
gcc

Threads

time (seconds) (log. scale)

X8c

icc
gcc

Threads

time (seconds) (log. scale)
PCB: Benchmarking Routines of the PCE

R-generate

Execution time model

- number of generated threads ≤ number of available cores

\[ T_{R\text{--generate}} = PT_{gen} + NT_{work} \]

- number of generated threads > number of available cores

\[ T_{R\text{--generate}} = PT_{gen} + NT_{work} \frac{P}{C} \left( 1 + \frac{T_{sw}}{T_{cpu}} \right) \]
PCB: Benchmarking Routines of the PCE

R-pfor

![Graphs showing performance of routines on different platforms](image)

<table>
<thead>
<tr>
<th>Platform</th>
<th>Threads</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>icc</td>
<td>gcc</td>
<td></td>
</tr>
<tr>
<td>P2c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X4c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X8c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PCB: Benchmarking Routines of the PCE
R-pfor
Execution time model

- number of generated threads ≤ number of available cores

\[ T_{R-pfor} = P T\_gen + \frac{N_T}{P} T\_work \]

- number of generated threads > number of available cores

\[ T_{R-pfor} = P T\_gen + \frac{N_T}{C} T\_work \left( 1 + \frac{T\_sw}{T\_cpu} \right) \]
PCB: Benchmarking Routines of the PCE

R-barriers

![Graphs showing performance comparison of ICC and GCC compilers for different programs and thread counts.](image)

These graphs compare the performance of ICC and GCC compilers for various programs (P2c, A4c, X4c, X8c) across different thread counts (1, 10, 100, 1000). The x-axis represents the number of threads, and the y-axis shows the execution time in logarithmic scale.
PCB: Benchmarking Routines of the PCE

R-barriers

Execution time model

- number of generated threads \( \leq \) number of available cores

\[
T_{R \text{-barriers}} = PT_{gen} + NT_{work} + PT_{syn}
\]

- number of generated threads \( > \) number of available cores

\[
T_{R \text{-barriers}} = PT_{gen} + NT_{work} \frac{P}{C} \left( 1 + \frac{T_{sw}}{T_{cpu}} \right) + PT_{syn}
\]
Outline

- Introduction

- Our approach: Poly-Compilation Engine (PCE)
  - PCB: Benchmarking Routines of the PCE
  - Proof of concept of the PCE

- Conclusions
Proof of concept of the PCE

- BR source
- Compiler A
- Compiler B
- X source
- Compiler A
- Compiler B

Decision Engine

- \( BR_A \)
- \( BR_B \)
- \( X_A \)
- \( X_B \)

SP
- \( SP_A \)
- \( SP_B \)
- other SP

PCE

AP

(version of X to use, number of threads, other SP)

problem size
X model
Proof of concept of the PCE routine X

- A simple example of how the PCE can work with a routine X
  1. estimating the SP by means of the PCB
  2. estimated SP + model of routine X → image of the behaviour of X
  3. image of the behaviour of X → PCE decides the AP (to reduce its execution time)
Proof of concept of the PCE
routine R-strassen
Execution time model

- number of generated threads ≤ number of available cores

\[
T_{R-strassen} = P T_{gen} + 7 \frac{2 \left( \frac{n}{2} \right)^3}{P} T_{mult} + \frac{18n^2}{4} T_{add}
\]

\[
T_{R-strassen} = P_1 P_2 T_{gen} + 49 \frac{2 \left( \frac{n}{2} \right)^3}{P_2 P_1} T_{mult} + 7 \frac{18n^2}{P_1} T_{add} + 18 \frac{n^2}{4} T_{add}
\]

- number of generated threads > number of available cores

\[
T_{R-strassen} = P T_{gen} + 7 \frac{2 \left( \frac{n}{2} \right)^3}{C} T_{mult} \left( 1 + \frac{T_{sw}}{T_{cpu}} \right) + \frac{18n^2}{4} T_{add}
\]

\[
T_{R-strassen} = P_1 P_2 T_{gen} + 49 \frac{2 \left( \frac{n}{2} \right)^3}{C} T_{mult} \left( 1 + \frac{T_{sw}}{T_{cpu}} \right) + 7 \frac{18n^2}{\min(P_1, C)} T_{add} \left( 1 + \frac{T_{sw}}{T_{cpu}} \right) + 18 \frac{n^2}{4} T_{add}
\]
### Proof of concept of the PCE routine R-strassen

#### SP values

<table>
<thead>
<tr>
<th></th>
<th>P2c</th>
<th>X4c</th>
<th>A4c</th>
<th>X8c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>icc   gcc</td>
<td>icc   gcc</td>
<td>cc   gcc</td>
<td>icc   gcc</td>
</tr>
<tr>
<td>$T_{gen}$ $(\mu s)$</td>
<td>75     25</td>
<td>75   25</td>
<td>75   25</td>
<td>75    25</td>
</tr>
<tr>
<td>$T_{sw}/T_{cpu}$</td>
<td>$2 + 0.01P$ $7 - 0.01P$</td>
<td>$0.9 + 0.3P$ $0.9 + 0.01P$</td>
<td>$0.8 + 0.2P$ $0.8 + 0.02P$</td>
<td>$6 + 0.05P$ $0.5 + 0.01P$</td>
</tr>
<tr>
<td>$T_{add}$ $(ns)$</td>
<td>$20 + 0.05P$ $20$</td>
<td>$23 + 0.3P$ $30 - 0.3P$</td>
<td>$40 + P$ $40 - 0.1P$</td>
<td>$10$ $10$</td>
</tr>
<tr>
<td>$T_{mul}$ $(ps)$</td>
<td>$400 + 100P$ $400 + 0.1P$</td>
<td>$140 + 10P$ $140 - P$</td>
<td>$60$ $60 - 0.5P$</td>
<td>$100$ $100$</td>
</tr>
</tbody>
</table>
Proof of concept of the PCE routine R-strassen
Experimental Vs. Theoretical times
Proof of concept of the PCE routine R-strassen
Experimental Vs. Theoretical times

A4c

Time (seconds)

Threads

X8c

Time (seconds)

Threads

A4c (theoretical)

Time (seconds)

Threads

X8c (theoretical)

Time (seconds)

Threads
Proof of concept of the PCE routine R-strassen taking decisions: AP values

<table>
<thead>
<tr>
<th>Algorithmic Parameters values, for the R-strassen routine, taken by the Poly-Compilation Engine for different platforms and compilers (Problem size=1000).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compiled version</strong></td>
</tr>
<tr>
<td>gcc</td>
</tr>
<tr>
<td><strong>Number of threads for the 1st level</strong></td>
</tr>
<tr>
<td><strong>Number of threads for the 2nd level</strong></td>
</tr>
</tbody>
</table>
Proof of concept of the PCE routine **R-strassen** executing the routine

<table>
<thead>
<tr>
<th></th>
<th>P2c</th>
<th>X4c</th>
<th>A4c</th>
<th>X8c</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCE</td>
<td>1.19</td>
<td>0.50</td>
<td>0.49</td>
<td>0.16</td>
</tr>
<tr>
<td>ORA</td>
<td>1.17</td>
<td>0.49</td>
<td>0.45</td>
<td>0.11</td>
</tr>
<tr>
<td>HW</td>
<td>1.37</td>
<td>0.55</td>
<td>0.65</td>
<td>0.12</td>
</tr>
<tr>
<td>SW</td>
<td>1.22</td>
<td>1.31</td>
<td>1.20</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Execution times of the R-strassen routine obtained with different Algorithmic Parameters values sets (times in seconds). Problem size=100
Outline

- Introduction

- Our approach: Poly-Compilation Engine (PCE)
  - PCB: Benchmarking Routines of the PCE
  - Proof of concept of the PCE

- Conclusions
Conclusions

- A good choice of compiler in a multicore contributes to accelerating the solution of problems.
- Best compiler depends on: the type of routine, the number of threads, the problem size, ...
- Working on the design and implementation of a complete Poly-Compilation Engine (PCE).
  - Calculates the basic System Parameters (SP) for platform-compiler.
  - Selects the Algorithmic Parameters (the compiled version and the number of threads), by means of the theoretical model of the execution time of the routine with the SP measurement.
- The behaviour of a PCE prototype has been shown with a routine.
- Similar studies with other linear algebra routines are being performed.
- Extending to heterogeneous platforms.