Analysis of the Influence of the Compiler on Multicore Performance

Javier Cuenca, Luis P. García
Domingo Giménez, Manuel Quesada

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 $\rightarrow$ 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)

- BR source
- Compiler A
  - BR_A
- Compiler B
  - BR_B
  - PCB
- X source
- PCE

Analysis of the Influence of the Compiler on Multicore Performance
Euromicro PDP 2010, Pisa (Italy)
Our approach: Poly-Compilation Engine (PCE)

Compiler A

BR source

Compiler B

PCB

BR$_A$

BR$_B$

SP$_A$

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 source

 Compiler A

 BR A

 BR B

 PCB

 Compiler B

 X source

 Compiler A

 X A

 X B

 Compiler B

 SP A

 SP B

 other SP

 SP

 PCE
Our approach: Poly-Compilation Engine (PCE)

- BR source
  - Compiler A
  - Compiler B

- X source
  - Compiler A
  - Compiler B

- Decision Engine
- PCB
- BR<sub>A</sub>
- BR<sub>B</sub>
- X<sub>A</sub>
- X<sub>B</sub>
- SP<sub>A</sub>
- SP<sub>B</sub>
- other SP

PCE
Our approach: Poly-Compilation Engine (PCE)

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

- Decision Engine
  - $B_{RA}$
  - $B_{RB}$
  - $X_{A}$
  - $X_{B}$
  - $S_{PA}$
  - $S_{PB}$
  - other SP
  - PCB

- PCE

Connections:
- BR source to Compiler A
- X source to Compiler A
- Compiler B to Compiler B
- Decision Engine to PCE
- Problem size
- X model
Our approach: Poly-Compilation Engine (PCE)

- BR source
  - Compiler A
  - Compiler B

- X source
  - Compiler A
  - Compiler B

- Decision Engine
  - BR_A
  - BR_B
  - PCB
  - SP_A
  - SP_B
  - other SP
  - X_A
  - X_B
  - 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
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

Analysis of the Influence of the Compiler on Multicore Performance
Euromicro PDP 2010, Pisa (Italy)
number of generated threads ≤ number of available cores

\[ T_{R\text{-}generate} = P T_{gen} + N T_{work} \]

number of generated threads > number of available cores

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

R-pfor

Analysis of the Influence of the Compiler on Multicore Performance

Euromicro PDP 2010, Pisa (Italy)
PCB: Benchmarking Routines of the PCE

R-pfor

Execution time model

- number of generated threads ≤ number of available cores

\[ T_{R-pfor} = PT_{\text{gen}} + \frac{N_T}{P} T_{\text{work}} \]

- number of generated threads > number of available cores

\[ T_{R-pfor} = PT_{\text{gen}} + \frac{N_T}{C} T_{\text{work}} \left( 1 + \frac{T_{\text{swap}}}{T_{\text{cpu}}} \right) \]
PCB: Benchmarking Routines of the PCE
R-barriers

![Graphs showing the performance of different compilers (icc and gcc) for varied thread counts for PCB routines in the PCE architecture.](image)

Analysis of the Influence of the Compiler on Multicore Performance
Euromicro PDP 2010, Pisa (Italy)
PCB: Benchmarking Routines of the PCE

R-barriers

Execution time model

- number of generated threads ≤ number of available cores

\[
T_{R-barriers} = P T_{gen} + N T_{work} + P T_{syn}
\]

- number of generated threads > number of available cores

\[
T_{R-barriers} = P T_{gen} + N T_{work} \frac{P}{C} \left( 1 + \frac{T_{swap}}{T_{cpu}} \right) + P T_{syn}
\]
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

- Compiler A
- Compiler B
- BR source
- X source
- Decision Engine
- BR_A
- BR_B
- PCB
- SP_A
- SP_B
- other SP
- X_A
- X_B
- problem size
- X model
- PCE
- AP
- (version of X to use, number of threads, other SP)
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 could decide the AP (to reduce its execution time)

Compiler A

BR_A

SP_A

other

BR source

X source

Compiler B

X_B

Engine

X model

PCE

AP

(version of X to use, number of threads, other SP)
Proof of concept of the PCE routine $R$-jacobi

Execution time model

- number of generated threads $\leq$ number of available cores

$$T_{R-jacobi} = P T_{gen} + \frac{11n^2}{P} T_{work}$$

- number of generated threads $>$ number of available cores

$$T_{R-jacobi} = P T_{gen} + \frac{11n^2}{C} T_{work} \left(1 + \frac{T_{swap}}{T_{cpu}}\right)$$
Proof of concept of the PCE routine R-jacobi

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</td>
<td>gcc</td>
<td>icc</td>
<td>gcc</td>
</tr>
<tr>
<td>$T_{gen}$ (µs)</td>
<td>75</td>
<td>25</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>$T_{work}$ (ns)</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>$T_{swap} / T_{cpu}$</td>
<td>2</td>
<td>1.5</td>
<td>15</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Proof of concept of the PCE routine R-jacobi
Experimental Vs. Theoretical times

- Graph showing the time (seconds) for different thread numbers for icc and gcc compilers.
Proof of concept of the PCE routine R-jacobi
Experimental Vs. Theoretical times

![Graph showing experimental vs. theoretical times for icc and gcc compilers. The x-axis represents the number of threads, and the y-axis represents time in seconds (log scale). The graph shows the performance of the X4c compiler under different conditions.]
Proof of concept of the PCE routine R-jacobi
Experimental Vs. Theoretical times
Proof of concept of the PCE routine R-jacobi
Experimental Vs. Theoretical times
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 (matrix-vector multiplication, different matrix-matrix multiplications, Strassen method, ...) are being performed.