A Parallelizing Compiler for Multicore Systems

José M. Andión, Manuel Arenaz, Gabriel Rodríguez and Juan Touriño

17th International Workshop on Software and Compilers for Embedded Systems (SCOPES 2014)
June 10-11, 2014 — Schloss Rheinfels, Sankt Goar, Germany
Outline

• Motivation: The Parallel Challenge
• KIR: A diKernel-based IR
• Automatic Partitioning driven by the KIR
• Experimental Evaluation
• Conclusions
Outline

- **Motivation: The Parallel Challenge**
- KIR: A diKernel-based IR
- Automatic Partitioning driven by the KIR
- Experimental Evaluation
- Conclusions
The Parallel Challenge

David A. Patterson and John L. Hennessy.
Computer Organization and Design: The Hardware/Software Interface.
The Parallel Challenge

- libraries

- compiler directives

- programming languages

- parallelizing compilers
Outline

• Motivation: The Parallel Challenge

• KIR: A diKernel-based IR

• Automatic Partitioning driven by the KIR

• Experimental Evaluation

• Conclusions
Outline

• Motivation: The Parallel Challenge

• **KIR: A diKernel-based IR**

• Automatic Partitioning driven by the KIR

• Experimental Evaluation

• Conclusions
diKernel: Domain-Independent Computational Kernel

- Characterizes the computations carried out in a program without being affected by how they are coded
- Exposes multiple levels of parallelism

Standard statement-based IR

1. `for (i = 0; i < n; i++) {`
2. `t = 0;`
3. `for (j = 0; j < m; j++) {`
4. `t = t + A[i][j] * x[j];`
5. `}`
6. `y[i] = t;`
7. `}`
Building the KIR (I)

\[
i = 0; \\
t = 0; \\
j = 0; \\
t = t + A[i][j] \times x[j]; \\
y[i] = t; \\
i++; \\
\]

- i = 0 dominates i++
- DEF(i, i=0) ⊇ USE(i, i++)
Building the KIR (II)
Building the KIR (and III)

1. \textbf{for} (i = 0; i < n; i++) {
2. \hspace{10pt} t = 0;
3. \hspace{10pt} \textbf{for} (j = 0; j < m; j++) {
4. \hspace{20pt} t = t + A[i][j] \times x[j];
5. \hspace{10pt} }
6. \hspace{10pt} y[i] = t;
7. \}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{image}
\caption{ROOT EXECUTION SCOPE}
\end{figure}
Outline

• Motivation: The Parallel Challenge
• KIR: A diKernel-based IR
• Automatic Partitioning driven by the KIR
• Experimental Evaluation
• Conclusions
Outline

• Motivation: The Parallel Challenge
• KIR: A diKernel-based IR
• **Automatic Partitioning driven by the KIR**
• Experimental Evaluation
• Conclusions
Automatic Partitioning driven by the KIR (I)

1. for (i = 0; i < n; i++) {
2.   t = 0;
3.   for (j = 0; j < m; j++) {
4.     t = t + A[i][j] * x[j];
5.   }
6.   y[i] = t;
7. }

\( t \) is a privatizable scalar variable
Automatic Partitioning driven by the KIR (II)

1. \textbf{for} (i = 0; i < n; i++) { 
2. \quad t = 0; \ 
3. \quad \textbf{for} (j = 0; j < m; j++) { 
4. \quad \quad t = t + A[i][j] \times x[j]; 
5. \quad } 
6. \quad y[i] = t; 
7. }

spurious diKernel-level dependence

\begin{itemize}
\item \textless \textbf{t}\textsubscript{BB1} \textgreater \ 
\text{scalar assignment}
\item \textless \textbf{t}\textsubscript{BB2} \textgreater \ 
\text{scalar reduction}
\item \textless y\textsubscript{BB4} \textgreater \ 
\text{regular assignment}
\end{itemize}
Automatic Partitioning driven by the KIR (III)

1. for (i = 0; i < n; i++) {
2.   t = 0;
3.   for (j = 0; j < m; j++) {
4.     t = t + A[i][j] * x[j];
5.   }
6.   y[i] = t;
7. }

Critical path
Automatic Partitioning driven by the KIR (and IV)

1. 

```
#pragma omp parallel for
2.  shared(A,x,y) private (t,i,j)
3.  for (i = 0; i < n; i++) {
4.    t = 0;
5.  for (j = 0; j < m; j++) {
6.    t = t + A[i][j] * x[j];
7.  }
8.  y[i] = t;
9.}
```

Critical path

Diagram:

- ES_for_i (Fig. 1, lines 1-7)
- ES_for_j (Fig. 1, lines 3-5)
- Regular assignment
- Scalar assignment
- Scalar reduction
Outline

• Motivation: The Parallel Challenge

• KIR: A diKernel-based IR

• Automatic Partitioning driven by the KIR

• Experimental Evaluation

• Conclusions
Outline

• Motivation: The Parallel Challenge
• KIR: A diKernel-based IR
• Automatic Partitioning driven by the KIR

• **Experimental Evaluation**

• Conclusions
Experimental Evaluation

- Built on top of GCC 4.4.0

- EQUAKE from SPEC CPU2000 on 2 Intel Xeon E5520 quad-core processors

- The Intel compiler is unable to parallelize this case study properly while our approach reduces the execution time

Outline

• Motivation: The Parallel Challenge

• KIR: A diKernel-based IR

• Automatic Partitioning driven by the KIR

• Experimental Evaluation

• Conclusions
Outline

- Motivation: The Parallel Challenge
- KIR: A diKernel-based IR
- Automatic Partitioning driven by the KIR
- Experimental Evaluation
- Conclusions
1. The KIR: a diKernel-based IR
   - diKernels
   - diKernel-level dependences
   - execution scopes

2. Automatic Partitioning Technique
   - coarse-grain parallelism
   - global OpenMP parallelization strategy
Future Work

• Locality exploitation techniques

• Fine-grain parallelism

• Many-core architectures such as GPUs

A Parallelizing Compiler for Multicore Systems

José M. Andión, Manuel Arenaz, Gabriel Rodríguez and Juan Touriño

17th International Workshop on Software and Compilers for Embedded Systems (SCOPES 2014)
June 10-11, 2014 — Schloss Rheinfels, Sankt Goar, Germany