

1. Introduction
2. Approach
3. Details
4. Conclusions

Detection and Utilization of Potential Parallelism in Software Systems

Christian Wulf

30.11.2012



Outline

1. Introduction
2. Approach
3. Details
4. Conclusions

1 Introduction

2 Approach

3 Details

4 Conclusions

Evolution of Multi-core Processors

In the context of desktop computers



(a) 2000: 1 core



(b) 2005: 2 cores



(c) 2006: 4 cores



(d) 2011: 8 cores

⇒ Parallel programming is **no longer optional** for increased speed-up

⁰ <http://www.intel.de>, <http://www.amd.de>

Challenges of Automating Parallelization



- Synchronization \Rightarrow Deadlocks, Starvation, etc.
- Detection of promising parallelization regions
- Non-faulty transformation

⁰ Acknowledgements to FreeDigitalPhotos.net and Sira Anamwong

Requirements

1. Introduction

2. Approach

3. Details

4. Conclusions

- Automatic or assisting tools
- Program restructuring
- Dependence analysis



Related Work

1. Introduction

2. Approach

3. Details

4. Conclusions

- Loops and arrays [ROR⁺08, YTT⁺00], rarely **I/O access**
- Static [HAM⁺05] or dynamic analysis, rarely both
- C or Java byte code [FGN12], rarely Java source code
- **No business applications** for evaluation

⇒ Goal: A more high-level approach

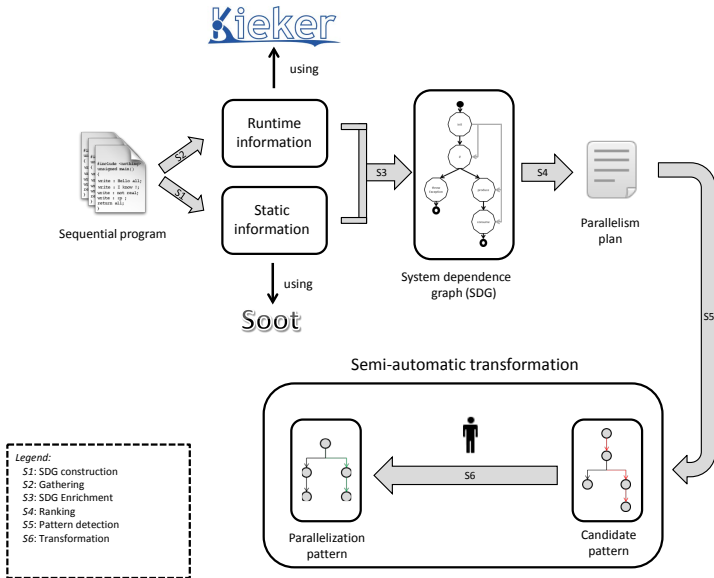
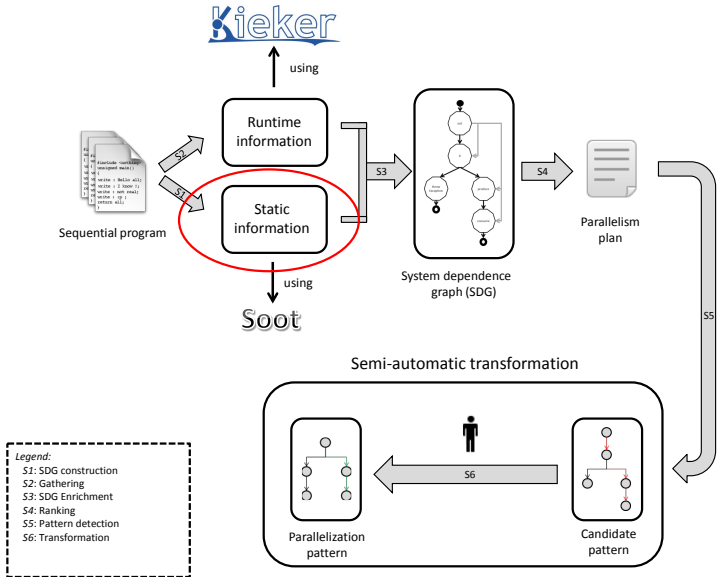


Figure 1: Overview of the approach

- 1. Introduction
- 2. Approach
- 3. Details
- 4. Conclusions



Legend:
 S1: SDG construction
 S2: Gathering
 S3: SDG Enrichment
 S4: Ranking
 S5: Pattern detection
 S6: Transformation

Figure 2: S1: SDG construction

- 1. Introduction
- 2. Approach
- 3. Details
- 4. Conclusions

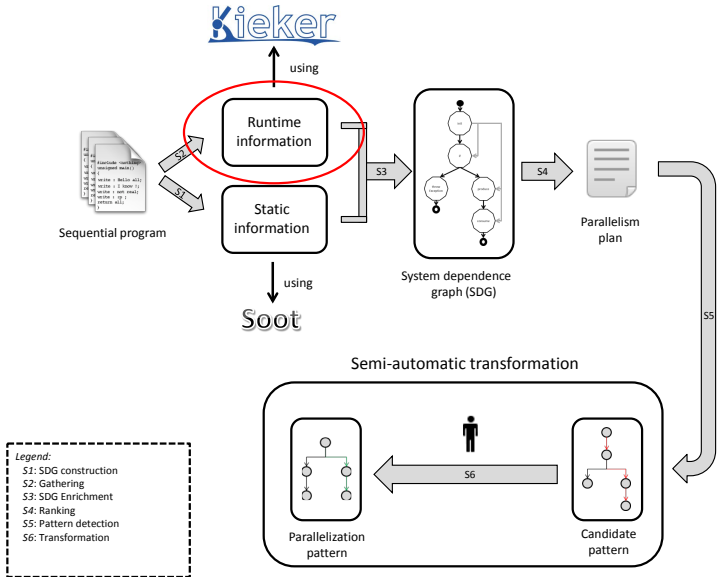


Figure 3: S2: Gathering runtime information

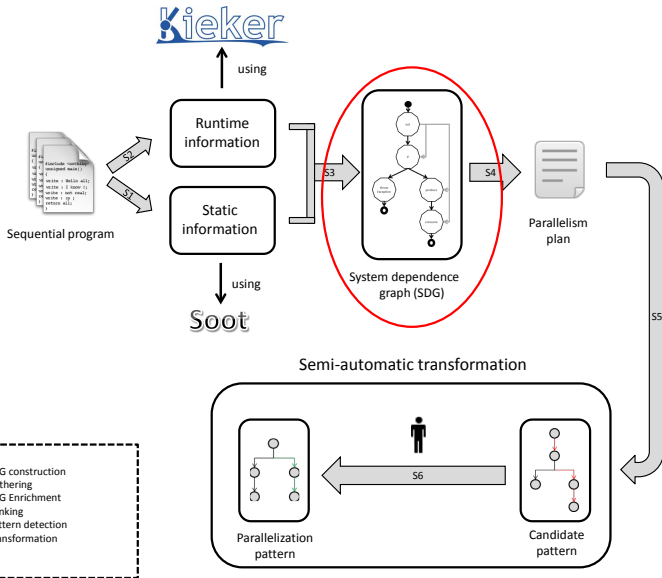


Figure 4: S3: SDG enrichment

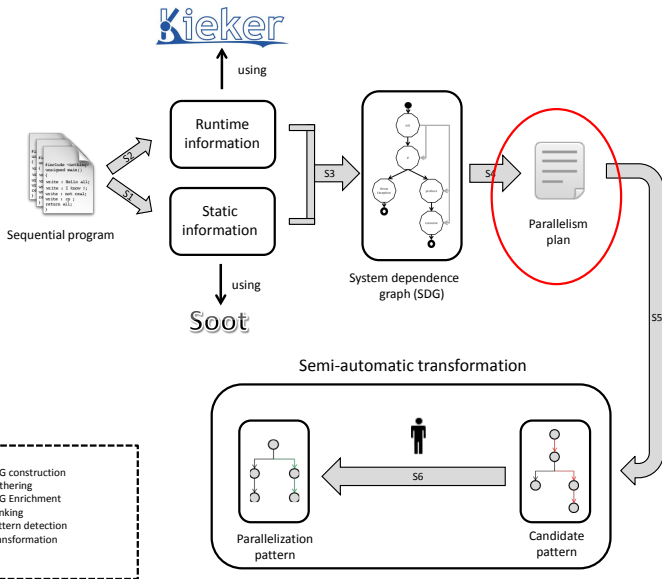


Figure 5: S4: Ranking

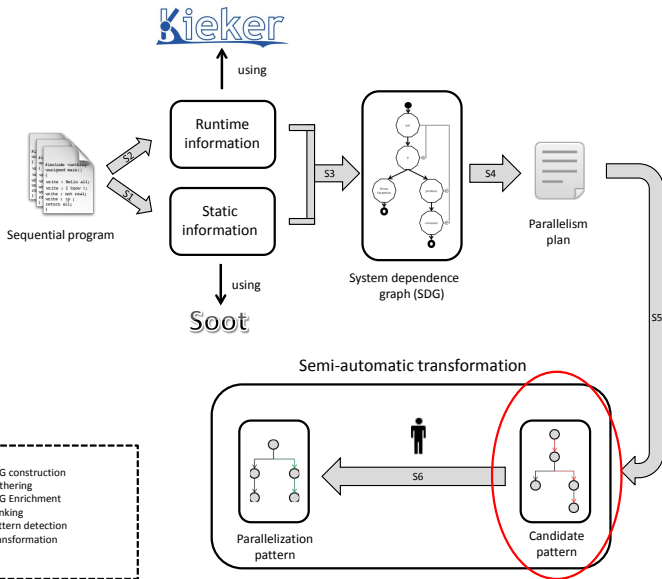


Figure 6: S5: Pattern detection

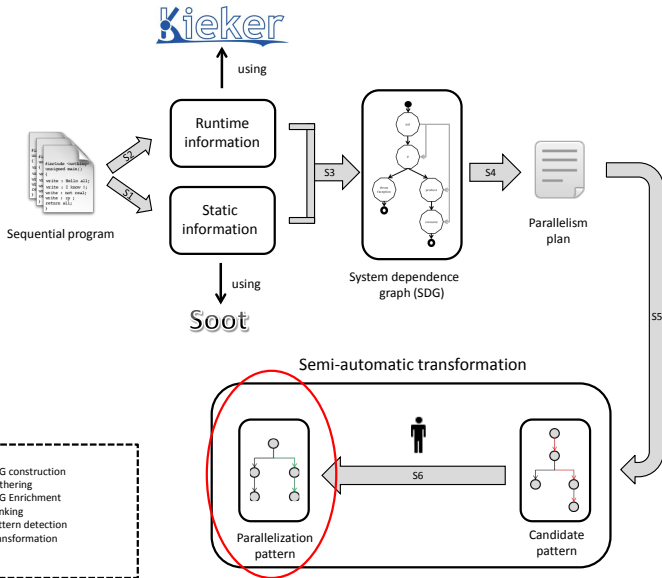


Figure 7: S6: Semi-automatic transformation

- 1. Introduction
- 2. Approach
- 3. Details
- 4. Conclusions

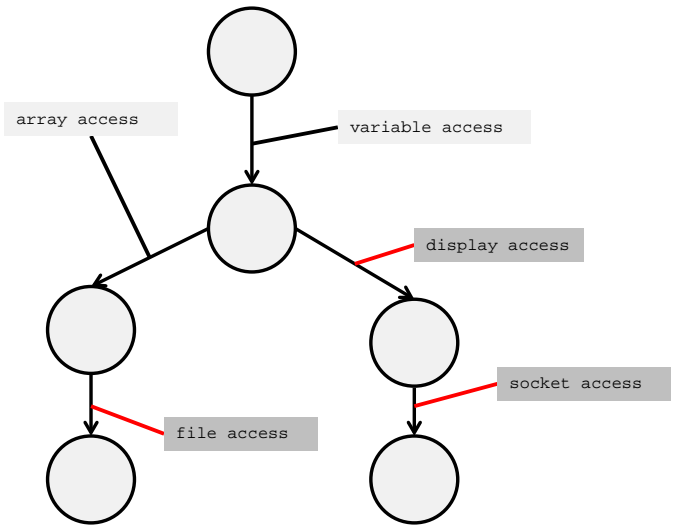



Figure 8: More high-level: Parallelize I/O accesses automatically

Distinguish I/O accesses


An Example

```
1 for (int i = 0; i < a.length; i++) {  
2     FileInputStream fis = new FileInputStream(a[i]);  
3     int fileContent = fis.read();  
4     ...  
5 }
```

constructor of I/O type



filename as argument



Distinguish I/O accesses

An Example

```
1 for (int i = 0; i < a.length; i++) {  
2     FileInputStream fis = new FileInputStream(a[i]);  
3     int fileContent = fis.read();  
4     ...  
5 }
```

file operation

1. Introduction
2. Approach
3. Details
4. Conclusions

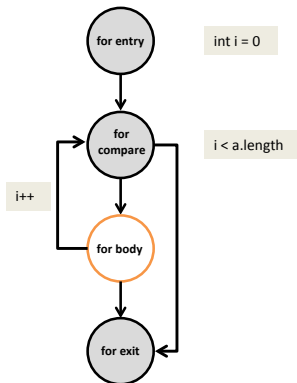


Figure 9: Exemplary candidate pattern

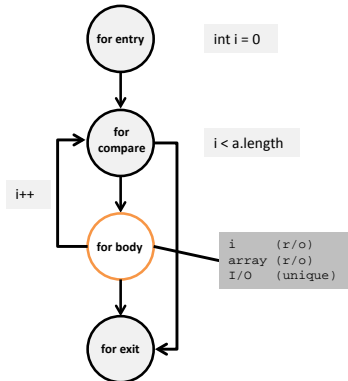


Figure 10: Exemplary candidate pattern

1. Introduction
2. Approach
3. Details
4. Conclusions

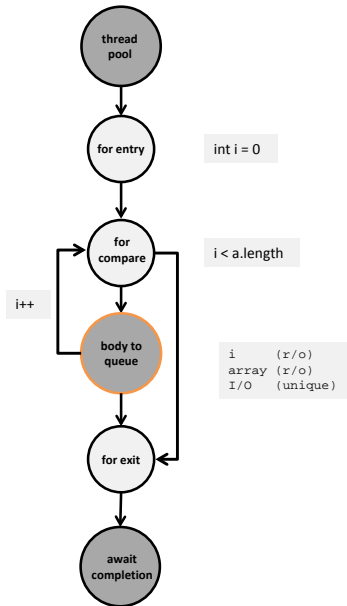


Figure 11: One associated parallelization pattern

Conclusions

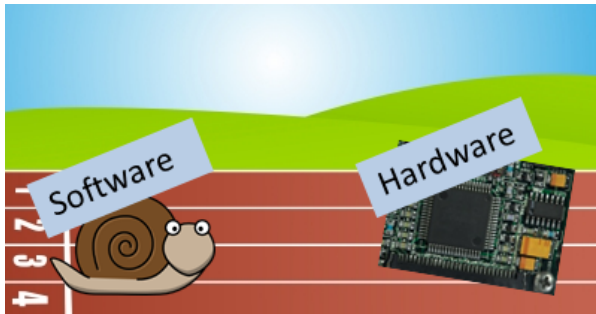


Figure 12: Software lags behind hardware development

- No standard supporting parallelization tools available
- Pattern-matching approach to assist in parallelization

⁰ Acknowledgements to FreeDigitalPhotos.net and digitalart

References



www.freedigitalphotos.net.



Min Feng, Rajiv Gupta, and Iulian Neamtiu.

Effective parallelization of loops in the presence of i/o operations.

In Proceedings of the 33rd ACM SIGPLAN conference on Programming Language Design and Implementation, PLDI '12, pages 487–498, New York, NY, USA, 2012. ACM.



Mary W. Hall, Saman P. Amarasinghe, Brian R. Murphy, Shih-Wei Liao, and Monica S. Lam.

Interprocedural parallelization analysis in suif.

ACM Trans. Program. Lang. Syst., 27(4):662–731, July 2005.



Easwaran Raman, Guilherme Ottoni, Arun Raman, Matthew J. Bridges, and David I. August.

Parallel-stage decoupled software pipelining.

In Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization, CGO '08, pages 114–123, New York, NY, USA, 2008. ACM.



Chao-Tung Yang, Shian-Shyong Tseng, Chang-Jiun Tsai, Cheng-Der Chuang, and Sun-Wen Chuang.

1. Introduction
2. Approach
3. Details
4. Conclusions

A new model of exploiting loop parallelization using knowledge-based techniques.

In [Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops, ICPADS '00](#), pages 9–, Washington, DC, USA, 2000. IEEE Computer Society.