The **Fellow Research Group**: Pattern-based Parallelization (AParT)
A research cooperation between the KIT and Siemens Corporate Technology

Korbinian Molitorisz
Location and Current Research

Karlsruhe Institute of Technology (KIT)
- Faculty of Computer Science
- Chair Prof. Dr. Walter F. Tichy
- Research Group AParT

Fellow Research Group AParT
- Parallelizing existing „legacy“ software
- Support engineering parallel software
- Active collaboration with Siemens Corporate Technology
About myself

- Employee at the chair of Prof. Walter F. Tichy
  - 2003 – 2014: Activity in the ESCde
    - Support institution in cooperation with Microsoft
  - 2003 – 2006: Support engineer, Head of QA
  - 2006 – 2014: General Manager
- Since 2009: Ph.D. in Multicore
  - Design pattern-based parallelization of existing software systems
  - Member in the Fellow Research Group (FRG)
    [http://frg.ipd.kit.edu](http://frg.ipd.kit.edu)
- Since 2014: Head of FRG
  - Research group with 4 members and 8 students (BA/MA/grad. assistant)
  - Integrated support concept for students‘ theses

Contact information

- Building 50.34, room 370,
  Tel. 0721 / 608 - 4 7155
Why do we deal with parallelization at all?

- „The free lunch is over“ – Multicore processors need parallel software
- But: Engineering parallel software is hard

**Where** should we parallelize?

**How** should we parallelize?

**Is the parallel program correct?**

**Does it execute faster?**
What makes engineering parallel software hard?

- Parallelization is time-consuming, tedious, and error-prone
  - Several weeks of work for implementing a pipeline (parallelization of a video processing software [OS+10])
  - Only the third parallelization strategy yielded a speedup (desktop search [MT10])
  - „Multicore processors are ubiquitous by now. The skills to program them is not“ [VM11]

Pattern-based parallelization process for legacy software is urgently needed [Mol13]

[OS+10] – Frank Otto, Christoph Schafer et al. A language-based tuning mechanism for task and pipeline parallelism, Euro-Par 2010
Pattern-based parallelization process

- **Flexible parallelization**
  - 1: Fully-automatic detection and transformation
  - 2: Semi-automatic transformation using architecture description
  - 3: Performance optimization of existing parallel architectures

---

1. Analysis
2. Pattern Detection
3. Compilation
4. Verification

---


Components of pattern-based parallelization

- **Pattern catalog**
  - Source patterns
  - Target patterns

- **Procedural method**
  - Detection (Pattern instances, runtime parameters)
  - Transformation

- **Specification of detected patterns**

- **Correctness validation**
  - Creation of unit tests
  - Dynamic data race detector

- **Runtime optimization**
  - Auto tuning using explicit runtime parameters
Detection of Source Patterns

- Code region with **high runtime share**
- Nesting of statements
  - Expression statement – statement block – return-statement
- Stepwise application of several image filters
- Program structures
  - Program loop: Iterative stream of statements
  - Statement sequence: Consecutive statements

```plaintext
01 AviStream Process(AviStream aviIn)
02 {
03  AviStream aviOut = new AviStream();
04  foreach(Image i in aviIn.Images)
05  {
06    Image crop = cropFilter.Apply(i);
07    Image histo = histogramFilter.Apply(i);
08    Image oil = oilFilter.Apply(i);
09    Image res = ConvTo32bpp.Apply(crop, histo, oil);
10    aviOut.Images.Add(res);
11  }
12  return aviOut;
13 }
```
Detection of Source Patterns

Program structure: Program iteration
- One thread per iteration or iteration sequence
- One thread per statement

Program structure: Statement sequence
- One thread per statement
- One thread per statement sequence

```
01 AviStream Process(AviStream aviIn)
02 {
03 AviStream aviOut = new AviStream();
04 foreach(Image i in aviIn.Images) {
05   Image crop = cropFilter.Apply(i);
06   Image histo = histogramFilter.Apply(i);
07   Image oil = oilFilter.Apply(i);
08   Image res = ConvTo32bpp.Apply(crop, histo, oil);
09   aviOut.Images.Add(res);
10 }
11 return aviOut;
12 }
```
Transformation to Target Patterns

Target patterns

- Program iteration → Pipeline
  - Statements: Stages
  - Video stream: Data
- Statement sequence → Master-/Worker
  - Control thread: Master
  - Statements: Worker

Speedup (on an 8-core machine)

- Manual: 6.47. Effort: 8 hours
- Automatic: 7.18. Effort: 10 minutes
Transformation to Target Patterns

- Code annotations
  - Defined operands and operators for **architecture description** and interconnecting **architecture components**
- Compiler with runtime library
  - Input: **Architecture description** and architecture components
  - Output: **Instances** from runtime library and tuning file

```plaintext
01 AviStream Process(AviStream aviIn)
02 {
03 AviStream aviOut = new AviStream();
04 #region TADL: (A || B || C+) => D => E
05 foreach(Image i in aviIn.Images)
06 {
07 #region A: Image c = cropFilter.Apply(i);    #endregion
08 #region B: Image h = histogramFilter.Apply(i); #endregion
09 #region C: Image o = oilFilter.Apply(i);  #endregion
10 #region D: Image r = Conv32bpp.Apply(c, h, o); #endregion
11 #region E: aviOut.Images.Add(r);         #endregion
12 }
13 #endregion
14 return aviOut;
15 }
```

```plaintext
01 AviStream Process(AviStream aviIn)
02 {
03 Item p1 = new Item (cropFilter.Apply());
04 Item p2 = new Item (histogramFilter.Apply());
05 Item p3 = new Item (oilFilter.Apply());
06 Item p4 = new Item (ConvTo32bpp.Apply());
07 Item p5 = new Item (aviOut.Images.Add());
08 MasterWorker mw = new MasterWorker (p1, p2, p3);
09 mw.Item(p3).replicable = true;
10 Pipeline p = new Pipeline (mw, p4, p5);
11 p.Input = aviIn.Images;
12 p.Run();
13 return p.Output;
14 }
```
Evaluation: Benchmark

Assessment of the pattern-based parallelization process for *pipelines* and *master/worker* using 26,500 LOC

<table>
<thead>
<tr>
<th>Project</th>
<th>#LOC</th>
<th>#Methods</th>
<th>Program structures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sequence</td>
</tr>
<tr>
<td>MergeSort</td>
<td>94</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>RayTracer</td>
<td>522</td>
<td>37</td>
<td>58</td>
</tr>
<tr>
<td>DesktopSearch</td>
<td>315</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>CompGeo</td>
<td>1,058</td>
<td>73</td>
<td>149</td>
</tr>
<tr>
<td>ImageProcessing</td>
<td>110</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>PowerCollections</td>
<td>24,481</td>
<td>1,252</td>
<td>3,354</td>
</tr>
</tbody>
</table>
## Evaluation:
### Search Space Reduction and Recall

<table>
<thead>
<tr>
<th>Project</th>
<th>#Program structures</th>
<th>Target patterns</th>
<th>Search space reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>MergeSort</td>
<td>19</td>
<td>1</td>
<td>95 %</td>
</tr>
<tr>
<td>RayTracer</td>
<td>63</td>
<td>6</td>
<td>90 %</td>
</tr>
<tr>
<td>DesktopSearch</td>
<td>47</td>
<td>2</td>
<td>96 %</td>
</tr>
<tr>
<td>CompGeo</td>
<td>169</td>
<td>1</td>
<td>98 %</td>
</tr>
<tr>
<td>ImageProcessing</td>
<td>16</td>
<td>1</td>
<td>94 %</td>
</tr>
<tr>
<td>PowerCollections</td>
<td>3,641</td>
<td>172</td>
<td>97 %</td>
</tr>
</tbody>
</table>

\[ \bar{\Omega} = 95\% \]
**Evaluation: Precision and Speedup**

- Test system: AMD FX 8120h, 8 processor cores, 3.1 GHz each

<table>
<thead>
<tr>
<th>Project</th>
<th>Target pattern</th>
<th>Correctness</th>
<th>Precision</th>
<th>Speedup</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>¬C</td>
<td>S</td>
<td>¬S</td>
</tr>
<tr>
<td>MergeSort</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100 %</td>
<td>1</td>
</tr>
<tr>
<td>RayTracer</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>17 %</td>
<td>1</td>
</tr>
<tr>
<td>DesktopSearch</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>50 %</td>
<td>1</td>
</tr>
<tr>
<td>CompGeo</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100 %</td>
<td>1</td>
</tr>
<tr>
<td>ImageProcessing</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100 %</td>
<td>1</td>
</tr>
<tr>
<td>PowerCollections</td>
<td>15*</td>
<td>13</td>
<td>2</td>
<td>27 %</td>
<td>6</td>
</tr>
</tbody>
</table>

| ∑ = 26 | ∑ = 24 | ∑ = 2 | Ø = 66% | ∑ = 11 | ∑ = 15 |
Parallelization process

- Generalization from C# to general object-oriented source code
- Operate on sequential and parallel source code
- Code coverage analysis for data race detection
- Model-based data race detection
## Related Work

### Scientific publications

<table>
<thead>
<tr>
<th>Scientific publications</th>
<th>Assessment of parallel potential</th>
<th>Generation of parallel suggestion</th>
<th>Automatic transformation</th>
<th>Focus on parallel patterns</th>
<th>Determination of data dependencies</th>
<th>Optimization of parallel program on target platform</th>
<th>Verification of parallel correctness</th>
<th>Adjustable pattern detection</th>
<th>Support of whole parallelization process</th>
<th>Supported platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GJ+11] - S. Garcia</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C/C++</td>
</tr>
<tr>
<td>[HS+09] - C. Hammacher</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Java</td>
</tr>
<tr>
<td>[K88] - M. Kumar</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Fortran</td>
</tr>
<tr>
<td>[KK+10] - M. Kim</td>
<td>-</td>
<td>+</td>
<td>MW, PL</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>OpenMP</td>
</tr>
<tr>
<td>[MC+07] - T. Moseley</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C++</td>
</tr>
<tr>
<td>[MF+10] - J. Mak</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>MW, PL</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Cilk</td>
</tr>
<tr>
<td>[RV+10] - S. Rul</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MW, PL</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C++</td>
</tr>
<tr>
<td>[TC+07] - W. Thies</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>PL</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C</td>
</tr>
<tr>
<td>[TF+10] - G. Tournavitis</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>PL</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C/C++</td>
</tr>
<tr>
<td>[TW+09] - G. Tournavitis</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>MW, PL</td>
<td>dyn</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C/C++</td>
</tr>
<tr>
<td>[W11] - A. Wilhelm</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>FJ, MW</td>
<td>stat</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>C/C++/Java</td>
</tr>
</tbody>
</table>

### Works of the FRG

<table>
<thead>
<tr>
<th>Works of the FRG</th>
<th>Assessment of parallel potential</th>
<th>Generation of parallel suggestion</th>
<th>Automatic transformation</th>
<th>Focus on parallel patterns</th>
<th>Determination of data dependencies</th>
<th>Optimization of parallel program on target platform</th>
<th>Verification of parallel correctness</th>
<th>Adjustable pattern detection</th>
<th>Support of whole parallelization process</th>
<th>Supported platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MS+12] - K. Molitorisz</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>Futures</td>
<td>stat</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>Java</td>
</tr>
<tr>
<td>[SM+13] - J. Schimmel</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>MW</td>
<td>dyn</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>.NET</td>
</tr>
</tbody>
</table>

### Commercial tools

<table>
<thead>
<tr>
<th>Commercial tools</th>
<th>Assessment of parallel potential</th>
<th>Generation of parallel suggestion</th>
<th>Automatic transformation</th>
<th>Focus on parallel patterns</th>
<th>Determination of data dependencies</th>
<th>Optimization of parallel program on target platform</th>
<th>Verification of parallel correctness</th>
<th>Adjustable pattern detection</th>
<th>Support of whole parallelization process</th>
<th>Supported platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Parallel Studio</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>FJ, MW</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>C/C++</td>
</tr>
<tr>
<td>Critical Blue Prism</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>FJ, MW</td>
<td>dyn</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>Windows, Linux</td>
</tr>
</tbody>
</table>
Thank you!

molitorisz@kit.edu
http://frg.ipd.kit.edu