Online Tuning of Stream Programs
or
How To Get The Most Out Of Your Multicore

Walter F. Tichy
Where is Karlsruhe?

University of Karlsruhe - KIT, Germany

Faculty of Computer Science

One of the leading CS departments in Europe
>40 faculty, >400 PhD students in CS
The changing parallel computing landscape

Cray vector computer 1976
The first five-core mobile phone

HTC One X, Feb. 1912,
Powered by Nvidia Tegra 3
Nvidia Tegra 3
Nvidia Tegra 3 Schematic

1 core at 500 MHz (battery saver)

4 cores at 1.5 GHz

1 GPU
AMD Opteron 12 cores
~1.8 Bill. T. on 2x3.46cm²

Sun Niagara3 16 cores
~1 Bill. T. on 3.7cm²

Intel SCC
48 cores
~1.3 Bill. T. on 5.6 cm²

Intel
2 cores
~167 Mio. T. on 1.1 cm²

Intel
8 cores
~2.3 Bill. T. on 6.8 cm²
The 2011 Intel Sandy Bridge

Currently: 4 CPUs, 6 graphics Execution Units
Later:       8 CPUs, 12 graphics Execution Units
Parallelism Everywhere

What is Missing?

Software!

Victor Pankratius
Fixing Parallel Performance Problems

- Parallelization is complex and error-prone
- Parallel programs contain a number of tuning parameters
- Manual optimization difficult and time-consuming
- Each target platform may require re-tuning

**Auto-Tuning**: Let the computer do the tuning!

### Examples for Tuning Parameters

- Number of pipeline stages
- Choice of best algorithm implementation
- Order of execution
- Size of data partitions
- Number of workers
- Type of core
- Load balancing strategy
Online Auto-Tuning

- Auto-Tuning Cycle:

- Example (pseudo code)

```java
TuningParameter numthreads(3, 64);
TuningParameter blocksize(100, 900, 100);

for(int i=0; i<numfiles; ++i) {
    startMeasurement();
    compress(files[i], blocksize, numthreads);
    stopMeasurement();
}
```
Parallelized BZip2, compressing 50 files on a machine with 8 cores
Initial tuning parameter values:
3 threads, block size 700 kB

Runtime without tuning: 22.9 s
Runtime with Auto-Tuner: 8 s

Best possible time (start with best configuration): 6.5 s
Stream Programming Paradigm

- A stream of elements flows through a graph of processing modules called *filters*.

- Task parallelism

- Pipeline parallelism

- Data parallelism (by filter replication)
(Some) Implicit Tuning Parameters

- **Replication factor:**

- **Cut-off depth:**

- **Alternative Algorithms/Cores:**
Measurement Sections in Stream Programs

- „Classic“ Fork/Join pattern:

- Stream program:

- Solution:
  - Count „heart beats“ (events triggered by stream elements)
  - Use heart beats to evaluate performance
Using Heartbeats for Online Tuning

- Heartbeats are emitted by sink filters
- The faster the heartbeat, the better the performance
- Heartbeats serve as an input signal for online auto-tuners

**Illustrating Example:**

![Diagram showing the process of using heartbeats for online tuning](image.png)
Benchmark 1: Video zoom

**Scale I**:
- Read
- Scale II
- Cut
- Write

**Scale II**:
- Read

**Execution time**

- Quadcore
- Dell
- Niagara

**Preconditions**: Statically predicted
**Tuned**: On-line auto-tuned
**Best**: Started with best known configuration, w/o Auto-Tuning

*replicable
First Come/First Serve
Benchmark 2: Electric
(Placement of circuits on a die)

- Part of VLSI design application
- 5 Filters with feedback loop and teleports
- 4 Tuning parameters

Diagram:

Producer → Calculate forces → Movement → Repair overlaps → Finish

* replicable
Electric: Results

Execution time

<table>
<thead>
<tr>
<th></th>
<th>Quadcore</th>
<th>Dell</th>
<th>Niagara</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tun</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Benchmarks on 4 cores

Fractions of best parallel performance (= 100%)
Benchmarks on 64 cores (Niagara)
Related Work (Selection)

- **ATLAS/AEOS** (Whaley et al., 2000)
  - Auto-tuning system for algebraic operations and algorithms
  - Domain specific approach
  - No support for parallel programs

- **Active Harmony** (Tapus et al., 2002)
  - Search-based auto-tuning system for library optimization
  - Comprehensive analysis of search algorithms
  - Not applicable for parallel programs

- **MATE** (Morajko et al., 2007)
  - Model-based tuning system for distributed PVM programs
  - Provides good performance predictions
  - Limited to special program structures

- **ATUNE** (Schaefer, Tichy, 2010)
  - General-purpose auto-tuner
  - Offline tuner (trial runs)
  - Pattern language for expressing parallel patterns (TADL)
Benchmark 3: Desktop search

Execution time

<table>
<thead>
<tr>
<th>Execution time</th>
<th>pre</th>
<th>tun</th>
<th>best</th>
<th>pre</th>
<th>tun</th>
<th>best</th>
<th>pre</th>
<th>tun</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadcore</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Dell</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Niagara</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
</tbody>
</table>

*replicable
Summary

- Computers are not the bottleneck.
- Programmers are!
- Stream programming simplifies parallel programming
  - Typical parallel patterns easy to write
  - Auto-tuning finds optimal operating conditions
  - Saves lots of tuning work

Further research

- Improved online search algorithms
- Use static model to predict good starting values
- Use auto-tuning to distribute work over heterogeneous cores
THANK YOU!
QUESTIONS?

With many thanks to
Frank Otto, Thomas Karcher, Jonas Thederer, Victor Pankratius

For more information, see: http://www.ipd.kit.edu/Tichy/
BACKUP SLIDES
Benchmarks on 8 cores

- DS
- Electric
- Series
- Vscale
- Vzoom

Bar chart showing performance percentages for each benchmark category.