Automated Scheduling
Algorithm Selection in OpenMP

Ali Mohammed†, Jonas H. Müller Korndörfer‡, Ahmed Eleliemy‡, and Florina M. Ciorba‡
†HPE’s HPC/AI EMEA Research Lab (ERL), Switzerland
‡Department of Mathematics and Computer Science, University of Basel, Switzerland

http://github.com/unibas-dmi-hpc/LB4OMP
Load Imbalance and Impact on Performance

- Increasing #cores/node
- Increased complexity to parallelize and exploit available compute power
- Load imbalance degrades performance, scalability, and wastes energy

#Cores/node in top systems on the Top500 list

Thread 1
Thread 2
Thread 3
Thread 4
Scheduling of Worksharing Loops in OpenMP

**static** (default)
dynamic runtime
guided auto

#iteration assigned/round → static/dynamic
threshold for #iteration assigned → guided

Default chunk
static: N/P
dynamic: 1
guided: 1

```c
#pragma omp parallel for schedule(kind, chunk)
for(i = 0; i < size; i++)
{
    computations ...
}
```

https://arxiv.org/abs/1809.03188


“Decision Paralysis”

• (Too?) Many choices for OpenMP `schedule(kind)`
• (Way?) Too many possible values for OpenMP `chunk`
• Scheduling choice needed per loop, per time-step, per application, and per system tuples

```c
#pragma omp parallel for schedule(auto)
for(i = 0; i < size; i++)
{
    computations ...
}
```

The schedule kind `auto` allows* an OpenMP implementation to choose any possible mapping of iterations in a loop construct to threads in the team.

* According to OpenMP Specification 5.2
Proposed Approach: Auto4OMP

- **Leverages auto** as a scheduling option for `schedule(kind)`
- **3 scheduling algorithm selection methods**
  - RandomSel
  - ExhaustiveSel
  - ExpertSel
- **Selects** from a portfolio of scheduling algorithms
  - STATIC, SS, GSS, GAC, TSS, Static Steal, mFAC2, AWF-B, AWF-C, AWF-D, AWF-E, mAF

**Extends** the LLVM OpenMP runtime library with the **expert chunk parameter**


http://github.com/unibas-dmi--hpc/LB4OMP
Auto4OMP: Scheduling Algorithm Selection Methods

**RandomSel**

- Random selection
- If LIB > 10 changes selected technique

**ExhaustiveSel**

- Exhaustive search
- If LIB > 10, re-evaluate the selected technique
Auto4OMP: Scheduling Algorithm Selection Methods

**ExpertSel**

- Employs fuzzy logic
- Constantly evaluates across execution instances
  - Loop execution time
  - Load imbalance
  - Change of loop execution time
  - Change of load imbalance

### Expert rules

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>Moderate</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{par}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short</td>
<td>Simple</td>
<td>Simple</td>
<td>Simple</td>
</tr>
<tr>
<td>Medium</td>
<td>Simple</td>
<td>Moderate</td>
<td>Moderate</td>
</tr>
<tr>
<td>Long</td>
<td>Simple</td>
<td>Moderate</td>
<td>Aggressive</td>
</tr>
</tbody>
</table>

Fuzzification → Expert rules → Defuzzification
Auto4OMP: Setup for Experimental Evaluation

Applications

- ALYA (computational mechanics)
- SPEC OMP 2012 352.nab (molecular modeling)
- SPHYNX (astrophysics)
- Mandelbrot (mathematics)
- GROMACS (molecular dynamics)

Algorithm Selection Methods

- LLVM auto (GAC)
- Manual selection
  - ManualBest
  - Oracle (knows all)
- Auto4OMP selection
  - RandomSel
  - ExhaustiveSel
  - ExpertSel

Systems

- miniHPC-Broadwell
  - 2 sockets, 10 cores each
- Piz Daint-Haswell
  - 1 socket, 12 cores
- miniHPC-Cascade-Lake
  - 2 sockets, 28 cores each

Experiments

25 loops x 5 applications x 3 systems using x 18 schedule(kinds) x 2 chunk parameters

= 2’700 total experiments
Hypothesis 1. Auto4OMP achieves high performance and provides the smallest variation of performance across application-system pairs.

<table>
<thead>
<tr>
<th>App-Sys pair</th>
<th>Selection</th>
<th>Prior work (no or manual selection)</th>
<th>Auto4OMP</th>
<th>Oracle exec. time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Static</td>
<td>Steal</td>
<td>GAC (LLVM-auto)</td>
</tr>
<tr>
<td>ALYA-Piz Daint-Haswell</td>
<td></td>
<td>11.38%</td>
<td>2.41%</td>
<td>2.03%</td>
</tr>
<tr>
<td>GROMACS-miniHPC-Broadwell</td>
<td></td>
<td>22.11%</td>
<td>0.44%</td>
<td>0.11%</td>
</tr>
<tr>
<td>GROMACS-Piz Daint-Haswell</td>
<td></td>
<td>23.23%</td>
<td>0.27%</td>
<td>0.17%</td>
</tr>
<tr>
<td>Mandelbrot-miniHPC-Broadwell</td>
<td></td>
<td>0.36%</td>
<td>0.38%</td>
<td>0.14%</td>
</tr>
<tr>
<td>Mandelbrot-Piz Daint-Haswell</td>
<td></td>
<td>0.26%</td>
<td>0.37%</td>
<td>0.09%</td>
</tr>
<tr>
<td>Mandelbrot-miniHPC-Cascade-Lake</td>
<td></td>
<td>0.46%</td>
<td>0.45%</td>
<td>0.35%</td>
</tr>
<tr>
<td>SPECOmp 2012 352.nab-miniHPC-Broadwell</td>
<td></td>
<td>4.64%</td>
<td>3.08%</td>
<td>0.20%</td>
</tr>
<tr>
<td>SPECOmp 2012 352.nab-Piz Daint-Haswell</td>
<td></td>
<td>3.84%</td>
<td>2.33%</td>
<td>1.59%</td>
</tr>
<tr>
<td>SPHYNX-miniHPC-Broadwell</td>
<td></td>
<td>22.01%</td>
<td>10.46%</td>
<td>5.44%</td>
</tr>
<tr>
<td>SPHYNX-Piz Daint-Haswell</td>
<td></td>
<td>38.81%</td>
<td>12.38%</td>
<td>3.04%</td>
</tr>
<tr>
<td>SPHYNX-miniHPC-Cascade-Lake</td>
<td></td>
<td>26.08%</td>
<td>11.02%</td>
<td>2.45%</td>
</tr>
</tbody>
</table>

Low performance variation across applications and systems

High performance variation across applications and systems
Hypothesis 2. Auto4OMP adapts to the various scheduling needs of applications when they execute on different systems.

<table>
<thead>
<tr>
<th>App-Sys pair</th>
<th>Selection</th>
<th>Prior work (no or manual selection)</th>
<th>Auto4OMP</th>
<th>Oracle exec. time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Static</td>
<td>Steal</td>
<td>S,expert</td>
<td>chunk</td>
</tr>
<tr>
<td>ALYA-Piz Daint-Haswell</td>
<td>0.48%</td>
<td>2.44%</td>
<td>0.29%</td>
<td>0.23%</td>
</tr>
<tr>
<td>GROMACS-miniHPC-Broadwell</td>
<td>22.11%</td>
<td>0.44%</td>
<td>1.18%</td>
<td>0.56%</td>
</tr>
<tr>
<td>GROMACS-Piz Daint-Haswell</td>
<td>23.12%</td>
<td>0.27%</td>
<td>0.75%</td>
<td>0.49%</td>
</tr>
<tr>
<td>Mandelbrot-miniHPC-Broadwell</td>
<td>0.36%</td>
<td>0.38%</td>
<td>0.02%</td>
<td>0.36%</td>
</tr>
<tr>
<td>Mandelbrot-Piz Daint-Haswell</td>
<td>0.20%</td>
<td>0.37%</td>
<td>0.13%</td>
<td>0.37%</td>
</tr>
<tr>
<td>Mandelbrot-miniHPC-Cascade-Lake</td>
<td>0.46%</td>
<td>0.45%</td>
<td>1.20%</td>
<td>1.64%</td>
</tr>
<tr>
<td>SPEC OMP 2012 352.nab-miniHPC-Broadwell</td>
<td>4.64%</td>
<td>3.08%</td>
<td>2.83%</td>
<td>0.96%</td>
</tr>
<tr>
<td>SPEC OMP 2012 352.nab-Piz Daint-Haswell</td>
<td>3.84%</td>
<td>2.33%</td>
<td>2.30%</td>
<td>1.99%</td>
</tr>
<tr>
<td>SPHYNX-miniHPC-Broadwell</td>
<td>22.01%</td>
<td>10.36%</td>
<td>13.48%</td>
<td>1.58%</td>
</tr>
<tr>
<td>SPHYNX-Piz Daint-Haswell</td>
<td>38.81%</td>
<td>12.38%</td>
<td>15.89%</td>
<td>1.65%</td>
</tr>
<tr>
<td>SPHYNX-miniHPC-Cascade-Lake</td>
<td>26.08%</td>
<td>11.02%</td>
<td>13.71%</td>
<td>1.19%</td>
</tr>
</tbody>
</table>

Low performance variation across systems, relative to Oracle.
Evaluation and Results

Hypothesis 3. The use of the **expert chunk parameter** improves application performance at no additional cost.

High variation of loop execution time across various scheduling techniques.
Evaluation and Results

Hypothesis 4. Auto4OMP adapts to the various scheduling needs of applications’ various loops within a single time-step (and across time-steps).

Auto4OMP selects different scheduling algorithms for SPHYNX’ L0 and L1
Hypothesis 5. Reducing OpenMP thread-level load imbalance improves overall performance of hybrid process+thread parallel applications (MPI+OpenMP).

**Improvement** over STATIC cumulated over 200 time-steps:
- **GAC**: 17.39%
- **SS,expert chunk parameter**: 22.08%
- **RandomSel**: 21.17%
- **ExhaustiveSel**: 21.65%
- **ExpertSel**: 21.22%

Thread-level load balancing plays a **significant** role in improving the performance of MPI+OpenMP applications.
Summary

- ManualBest and Oracle require **exhaustive offline experimentation**

- **Auto4OMP** is a fully automated load balancing solution for (work sharing) loops in **OpenMP** for various applications-systems pairs at **no additional cost** from the user.

- **Auto4OMP** provides:
  - An expert chunk parameter
  - Three scheduling algorithm selection methods
    - RandomSel
    - ExhaustiveSel
    - ExpertSel

- Up to **11%** performance improvement over current schedule **auto** implementation in LLVM

- **First step** towards automated load balancing in OpenMP
Next Steps

Automated Scheduling Algorithm Selection for Process-level Load Balancing

MPI+OpenMP Algorithm Selection During a Single Loop Execution

Expert-based vs. Machine-learning-based

More OpenMP constructs

More Scheduling/Selection Methods
OpenMP applications can now benefit from an automated approach of selecting the scheduling algorithm that fits the loop, in a given time-step, on a given platform.

Download Auto4OMP, use it, and tell us how it works for your application.

http://github.com/unibas-dmi--hpc/LB4OMP
OpenMP

SC22 Booth Talk Series

openmp.org  OpenMP API spec, videos, reference guides, and more

link.openmp.org/sc22  Videos and PDFs of OpenMP SC22 presentations