OpenMP®

SC'20 Booth Talk Series

LB4OMP: A Load Balancing Portfolio for OpenMP

Jonas H. Müller Korndörfer, PhD Student
University of Basel, Switzerland

Outline

- ✧ Motivation
- ✧ History
- ✧ LB4OMP
  - ✧ Scheduling Techniques
  - ✧ Performance Measurement Features
  - ✧ Usage
- ✧ Performance Evaluation
- ✧ Take Home Messages

Florina Ciorba

Ali Mohammed

DMI HPC

hpc.dmi.unibas.ch

Ahmed Eleliemy

Jonas H. Müller Korndörfer

University of Basel

FNSNF Swiss National Science Foundation

MLS: Multilevel Scheduling in Large Scale High Performance Computers

Motivation

- ✧ Load imbalance in OpenMP codes
  - ✧ Lowers performance
  - ✧ Wastes resources and energy
  - ✧ Increases waiting times in jobs queues

- ✧ Missing implementation for the state of the art in literature for load balancing
  - ✧ Hinders research for novel load balancing algorithms
  - ✧ Hinders performance optimization

History

IWOMP 2018: "**OpenMP loop scheduling revisited: making a case for more schedules**", Ciorba, Florina M., Christian Iwainsky, and Patrick Buder.

- ✧ GNU OpenMP runtime library

ISPDC 2019: "**Exploring loop scheduling enhancements in OpenMP: an LLVM case study**", Kasielke, Franziska, Ronny Tschüter, Christian Iwainsky, Markus Velten, Florina M. Ciorba, and Ioana Banicescu.

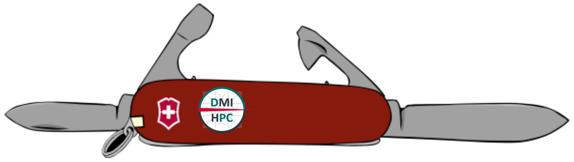
- ✧ LLVM OpenMP runtime library

SC19 poster: "**A Runtime Approach for Dynamic Load Balancing of OpenMP Parallel Loops in LLVM**", Müller Korndörfer, Jonas H., Ciorba, Florina M., Yilmaz, A., Iwainsky, C., Doerfert, J., Finkel, H., Kale, V. and Klemm, M.

- ✧ LLVM OpenMP runtime library

Today: LB4OMP

- ✧ LLVM OpenMP runtime library
- ✧ Swiss Army knife for load balancing in OpenMP



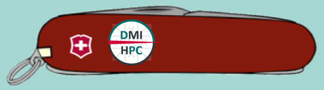
In a Nutshell

- ✧ LB4OMP: A Load Balancing Portfolio for OpenMP
  - ✧ ***Bridges the gap*** between the *state of the art* load balancing literature and *state of the practice* in multithreaded applications
  - ✧ Enhanced LLVM OpenMP runtime library
    - ✧ Dynamic and ***non-adaptive*** self-scheduling techniques (9)
    - ✧ Dynamic and ***adaptive*** self-scheduling techniques (8)
    - ✧ Performance measurement features

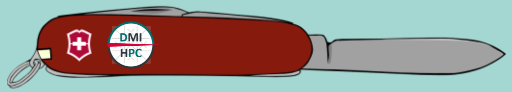
✓ LB4OMP: github.com/unibas-dmi-hpc/LB4OMP

LB4OMP: Scheduling Techniques

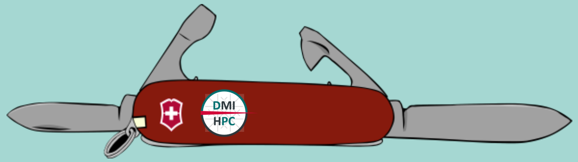
- Static scheduling technique (1)
  - ***Static*** *→ OpenMP standard*



- Dynamic and ***non-adaptive*** self-scheduling techniques (9)
  - ***SS****: Dynamic, 1 → OpenMP standard*
  - ***GSS****: Guided → OpenMP standard*
  - **FSC**: Fixed size chunk → requires profiling
  - ***TSS****: Trapezoid self-scheduling* → LLVM
  - **FAC**: Factoring → requires profiling
  - **mFAC**: modified implementation of FAC → requires profiling
  - **FAC2**: practical variant of factoring
  - **TAP**: Tapering → requires profiling
  - **WF2**: practical variant of weighted factoring



- Dynamic and **adaptive** self-scheduling techniques (8)
  - **BOLD** → requires profiling
  - **AWF-B,C,D,E**: Adaptive weighted factoring and its variants
  - **AF**: Adaptive factoring
  - **mAF**: modified implementation of AF



LB4OMP: Performance Measurement Features

Each thread: → *loop occurrence, location, iterations, thread ID, thread execution time*

Each parallel loop: → *location, iterations, parallel loop execution time*

Each scheduling round: → *location, lower bound, upper bound, chunk size, thread ID*

Warning: it can produce very large files!

Profiling: → *location, mean iteration execution time, standard deviation* execution time all iterations

LB4OMP: Usage

✧ Basic configuration

Do the target OpenMP loops in the application contain *schedule(runtime)* clause?

If yes, no recompilation is required

Add the path to the compiled LB4OMP to the linker environment variable

In Linux:

LD_LIBRARY_PATH=path/LB4OMP

OMP_SCHEDULE=technique,chunk
KMP_CPU_SPEED=clock frequency in MHz

In Linux:

cat /proc/cpuinfo

✧ Performance measurement features configuration

Each thread
Each parallel loop

KMP_TIME_LOOPS=path/file

Each scheduling round

KMP_PRINT_CHUNKS=1
KMP_TIME_LOOPS=path/file

Profiling

OMP_SCHEDULE=profiling
KMP_PROFILE_DATA=path/file

Performance Evaluation

- Application
  - **SPHYNX** executed **5 times for each configuration**, available at astro.physik.unibas.ch/people/ruben-cabezon/sphynx.html
  - **2 main OpenMP loops**, each with 1,000,000 iterations, executed 20 times for each SPHYNX execution
  - **L0**, find neighbours
  - **L1**, gravity calculation

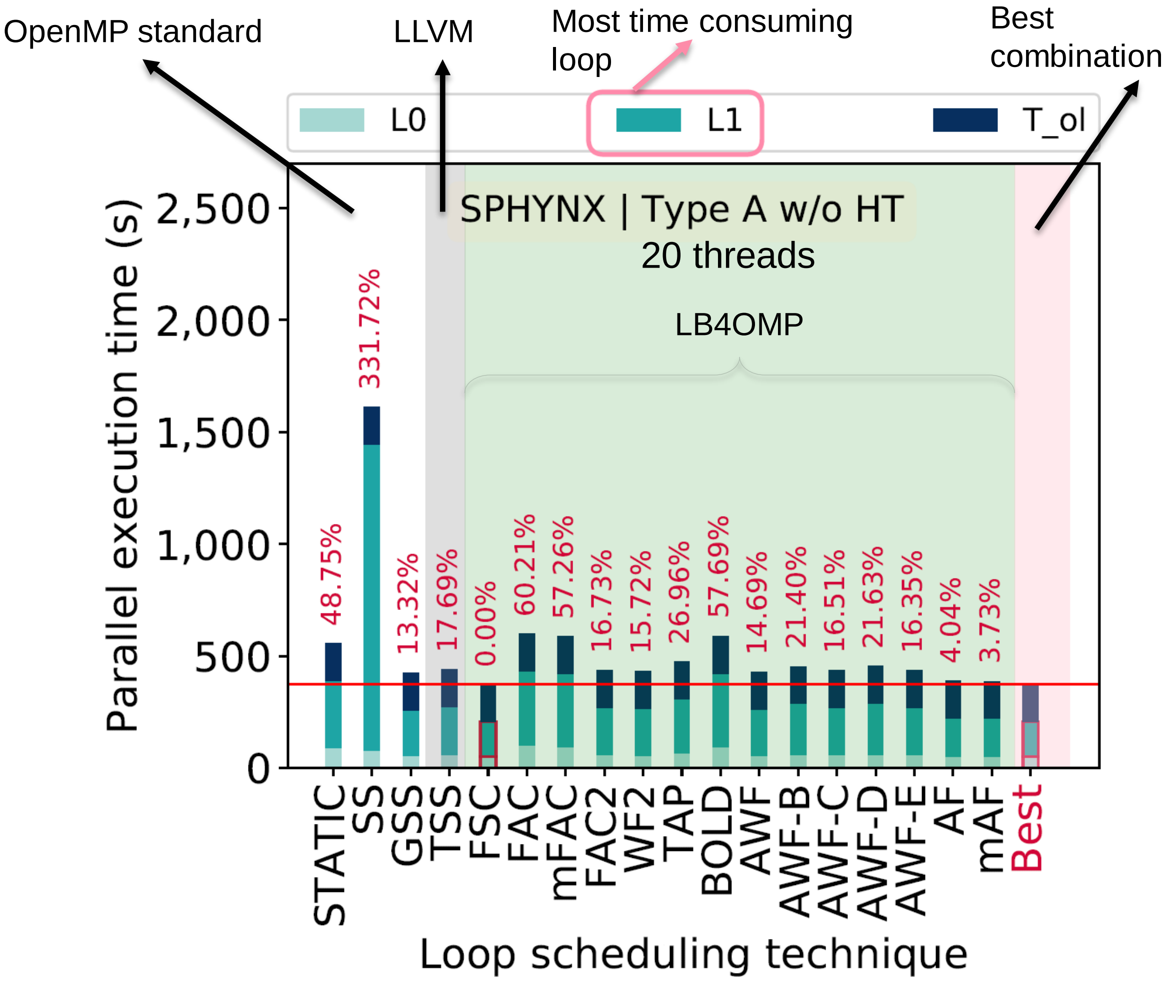
- Node types
  - **Type A**, Intel Broadwell E5-2640 v4 (**2 sockets, 10 cores each**)
  - **Type B**, Intel Xeon Phi KNL 7210 (**1 socket, 64 cores**)
  - **Type C**, Intel Xeon E5-2690 v3 (**1 socket, 12 cores**)

- Metrics
  - Parallel execution time
    - Parallel execution time per loop
  - Loss of performance compared to the **Best** combination of scheduling technique per loop

SPHYNX and LB4OMP were compiled with Intel compiler version 19.0.1.144
The threads were always configured with OMP_PLACES=cores OMP_PROC_BIND=close

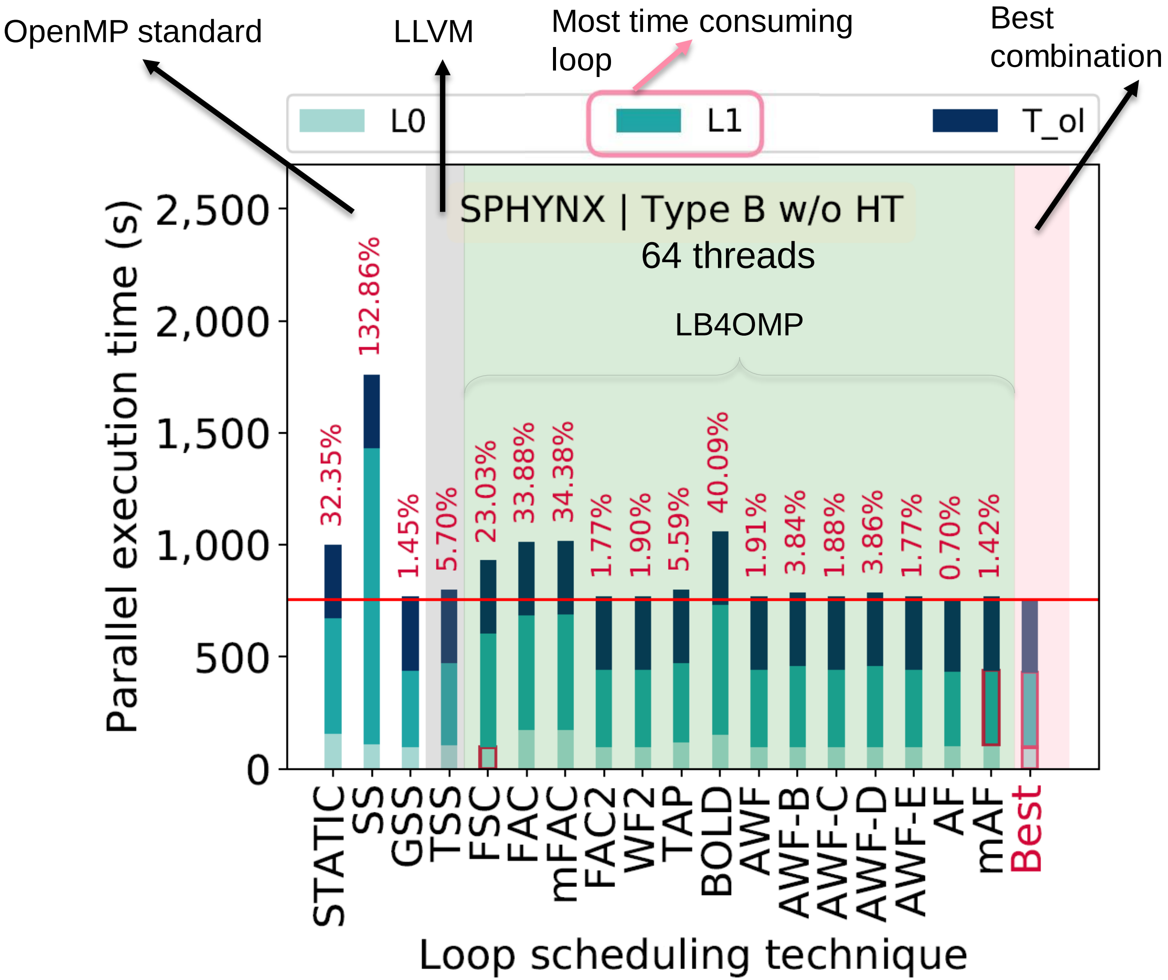
Performance Evaluation

- **Best** per loop ☐ performing scheduling technique
- **Best** ☐☐ · · · is a combination of scheduling techniques
  - **In this case, FSC achieves the highest performance alone**
- Performance degradation (xx.xx%) by executing the application with a single scheduling technique
- **Best** achieves up to 13.32% **higher performance** than the best standard, in this case **GSS**
- **mAF** alone achieves up to 9.59% **higher performance** than the best standard, in this case **GSS**



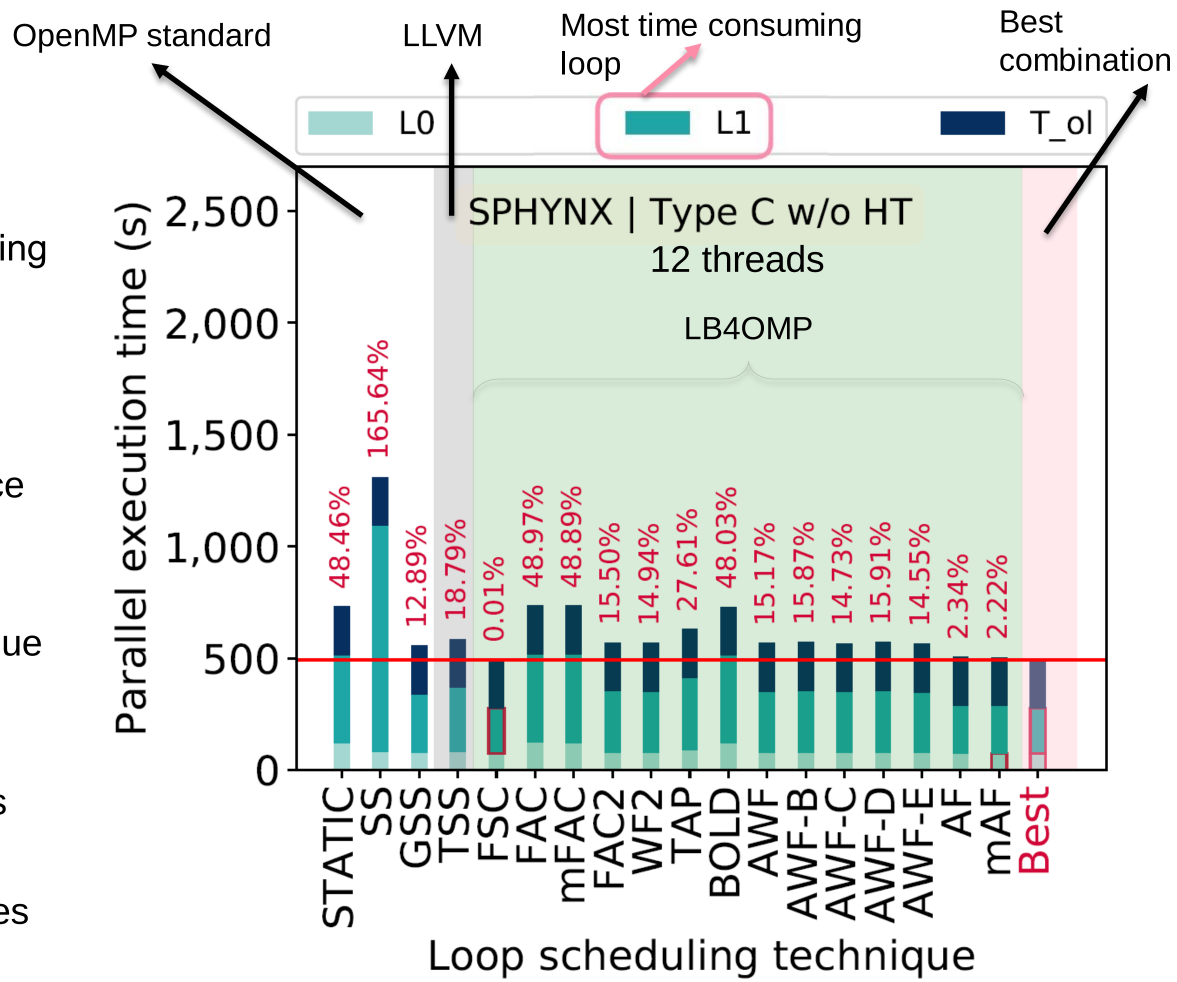
Performance Evaluation

- ✧ **Best** per loop ▭ performing scheduling technique
- ✧ **Best** ▭▭ · · · is a combination of scheduling techniques
  - ✧ **In this case, FSC and mAF**
- ✧ This time the performance of FSC alone is 23.03% lower than **Best**
- ✧ **AF** alone achieves up to 0.75% **higher performance** than the best standard, in this case **GSS**
- ✧ The **adaptive** techniques achieve **comparable or higher performance** than the best standard, this case **GSS**



Performance Evaluation

- **Best** per loop ▭ performing scheduling technique
- **Best** ▭▭ · · · is a combination of scheduling techniques
  - **In this case, FSC and mAF**
  - FSC is practically the best alone achieving only 0.01% lower performance than **Best**
- **Best** achieves up to 12.89% higher performance than the best standard technique alone, in this case GSS
- **mAF** alone achieves up to 10.67% **higher performance** than the best standard, in this case **GSS**
- The performance of the **adaptive** techniques remains constant across different platforms



Take Home Messages

- LB4OMP portfolio ***bridges the gap*** between the load balancing literature and practice in multithreaded applications
  - LB4OMP contains 14 additional to the OpenMP standard and ready to use dynamic (and **adaptive**) self-scheduling techniques
- First and necessary step for an auto-tuning load balancing approach in OpenMP
- Loops are frequently different presenting divergent load balancing needs
  - The **Best** combination of scheduling techniques frequently outperform the usage of a single technique
  - It is impractical to achieve **Best** only by experimentation
  - The dynamic and **adaptive** self-scheduling techniques are a promising alternative to achieve a performance close to **Best**
- **What's next?**
  - Patch and upstream the scheduling techniques to LLVM
  - Ongoing working on an automated approach to achieve **Best** performance

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

OpenMP®

SC'20 Booth Talk Series

openmp.org	OpenMP API specs, forum, reference guides, and more
link.openmp.org/sc20	Videos and PDFs of OpenMP SC'20 presentations