MLS: Multilevel Scheduling in Large Scale High Performance Computers

University of Basel

Florina M. Ciorba (Principle Investigator), Ahmed Eleliemy, and Jonas H. Müller Korndörfer
Department of Mathematics and Computer Science, University of Basel, Switzerland
{firstname.lastname}@unibas.ch
hpc.dmi.unibas.ch/HPC/MLS.html

FNSNF Swiss National Science Foundation

Duration: September 1, 2017 – August 31, 2020

1. Project Description

Research Question

Given massive parallelism, at multiple levels, and of diverse forms and granularities, how can it be ***exposed***, ***expressed***, and ***exploited*** by parallel and distributed applications such that execution times are reduced, performance targets are achieved, and acceptable efficiency is maintained?

Objectives

1. Leverage all available parallelism **within** each single hardware parallelism level and **across** the three hardware parallelism levels (system, node, core).
2. Achieve robustness against perturbations (including variations and failures) while minimizing execution time, maintaining acceptable efficiency, and maximizing resource utilization.

Methodology

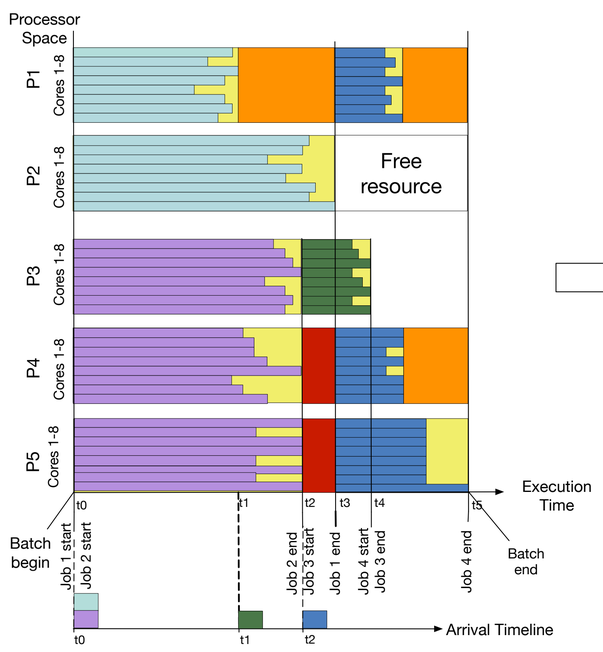
Multilevel scheduling (MLS) extends and bridges the most successful batch (jobs scheduled on a system), application (application processes scheduled on allocated nodes/processors/cores), and thread (application threads scheduled on the allocated cores) scheduling approaches beyond a single or a couple levels of parallelism (scaling up) and beyond their current scale (scaling out).

Envisioned Outcomes

A prototype multilevel scheduling solution that integrates live feedback information from three, currently disjoint, scheduling levels: job, process, and thread.

3. Exposing, Expressing, and Exploiting Multilevel Parallelism

Current State in Batch, Application, and Thread Level Scheduling



• *Four jobs, five compute nodes, eight cores per compute node, one process per node and one thread per core*

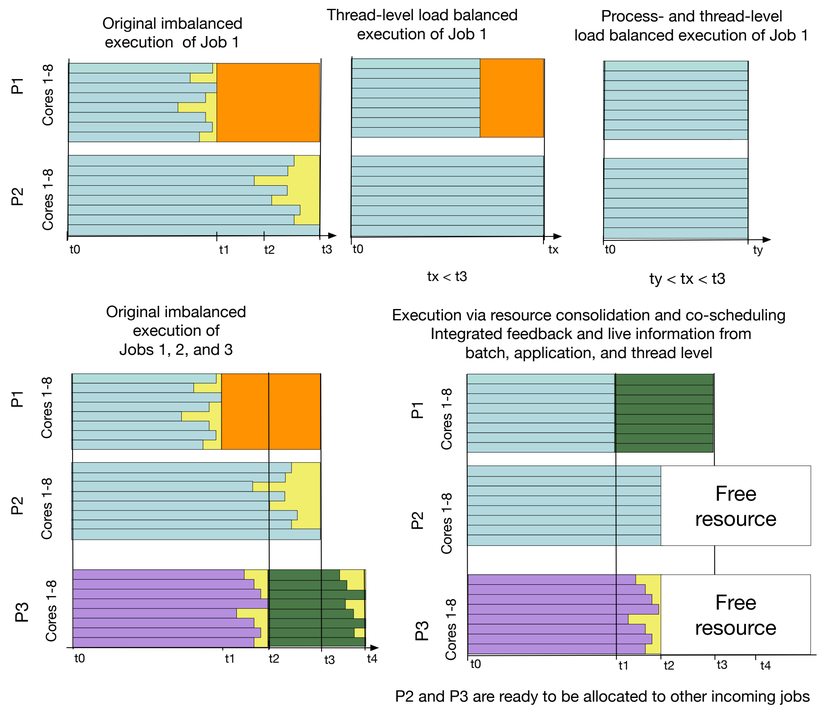
Job 1 · Job 2 · Job 3 · Job 4 · Load imbalance at the thread level · Load imbalance at the process level · Load imbalance at the batch level · Unexploited hardware parallelism

MLS

Eliminate Resource Fragmentation

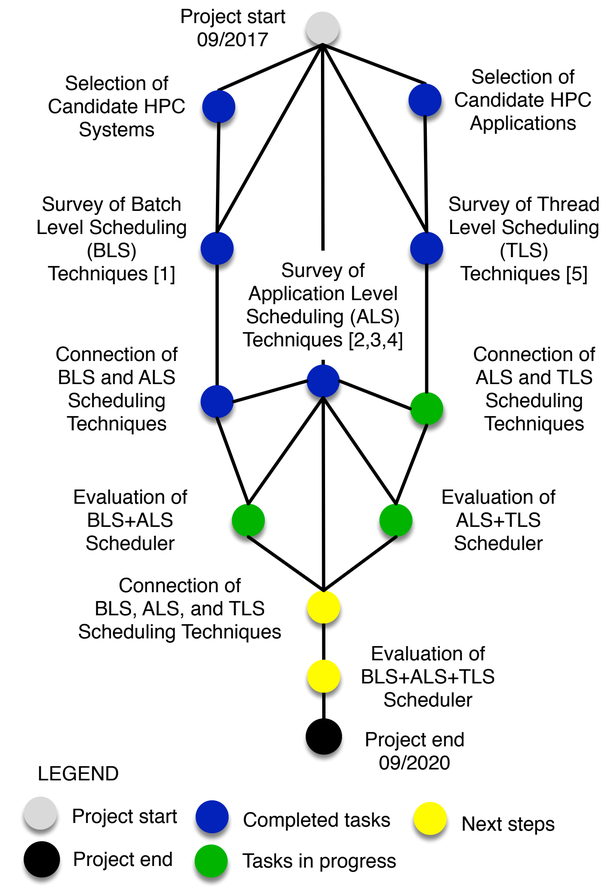
Exploit Unavoidable Resource Fragmentation

Multilevel Load Balanced Execution



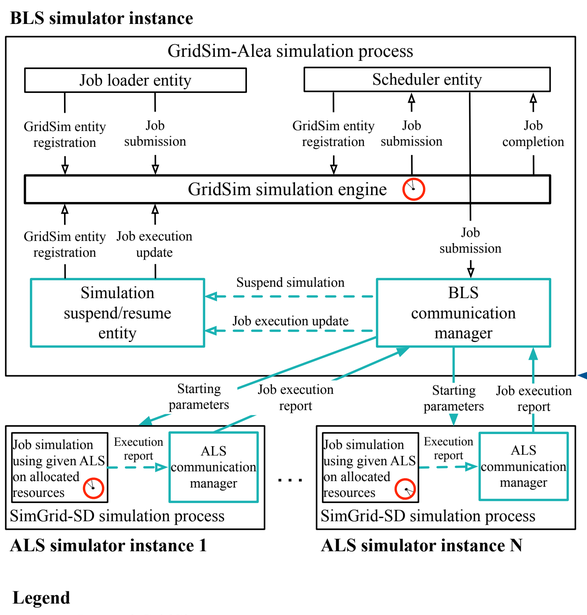
Batch (job) scheduling: SLURM, OpenPBS, TORQUE, etc.
Application (MPI) scheduling: self-scheduling, work-stealing, etc.
Thread (OpenMP) scheduling: static, guided, dynamic

4. Project State

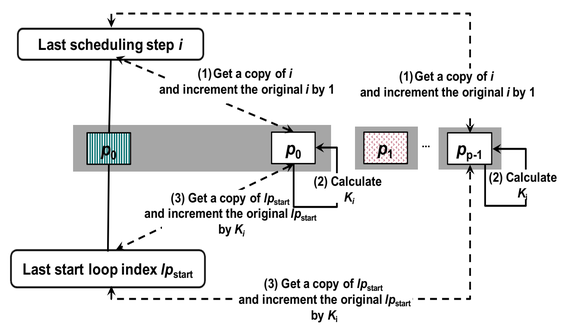


5. Selected Results

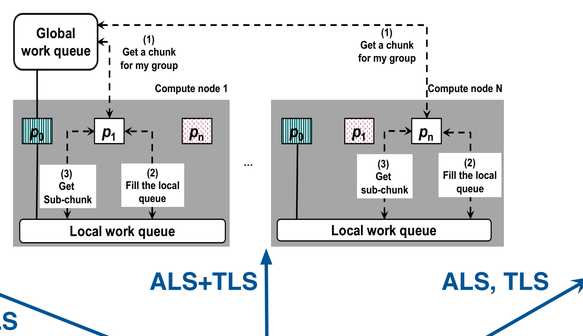
Result 1: A two-level simulator for **batch** and **application** level scheduling [ISPDC 2017]



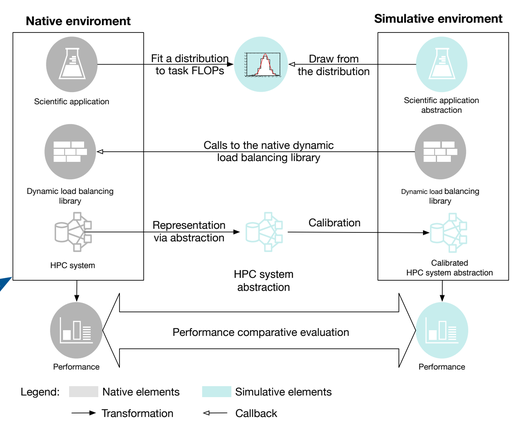
Result 2: A **distributed chunk-calculation** approach for dynamic loop scheduling at ALS on **heterogeneous distributed-memory** systems [PDP 2019]



Result 3: Hierarchical distributed chunk calculation approach for **intra- and inter node** load balancing using an **MPI+MPI approach** [IPDPSW 2019]



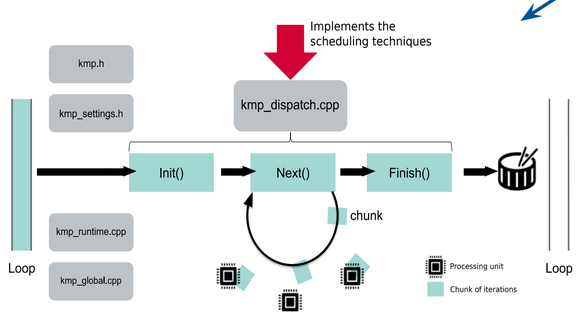
Result 4: Generic methodology for **realistic simulations** of parallel applications on HPC systems [FGCS 2019]



ALS · ALS+TLS · ALS, TLS · BLS+ALS · TLS · ALS+TLS

MLS

Result 5: A Runtime Approach for Dynamic Load balancing of OpenMP Applications [SC'19 poster]



Result 6: Implemented dynamic loop scheduling algorithms in **OpenMP** (LLVM) & an **MPI** Library DLS4LB [PASC19] [SIAM PP 20]

Non-adaptive dynamic scheduling	Adaptive dynamic scheduling
Self-scheduling (SS)	Adaptive weighted factoring (AWF) and its variants (B, C, D, E)
Guided self-Scheduling (GSS)	Adaptive factoring (AF)
Fixed size chunking (FSC)	
Factoring (FAC)	
Taper (TAP)	
Weighted factoring (WF)	
Bold (BOLD)	

Terminology

- Chunk refers to a set of independent tasks that are self-scheduled in a single assignment round following a process' work request

More MLS Details

hpc.dmi.unibas.ch/HPC/MLS.html

6. Selected Project Publications

[1] Eleliemy, A., Mohammed, A., and Ciorba, F. M. "*Exploring the Relation Between Two Levels of Scheduling Using a Novel Simulation Approach*", In Proceedings of the 16th International Symposium on Parallel and Distributed Computing (ISPDC), 2017.
[2] Eleliemy, A. and Ciorba, F. M. "*Dynamic Loop Scheduling Using MPI Passive-Target Remote Memory Access*", In Proceedings of the 27th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2019.
[3] Eleliemy, A. and Ciorba, F. M. "*Hierarchical Dynamic Loop Scheduling on Distributed-Memory Systems Using an MPI+MPI Approach*", In Proceedings of the 20th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC), 2019.
[4] Mohammed, A. Eleliemy, F. M. Ciorba, F. Kasielke, and I. Banicescu "*An Approach for Realistically Simulating the Performance of Scientific Applications on High Performance Computing Systems*", Future Generation Computer Systems (FGCS), 2019
[5] J. H. Müller Korndörfer, F. M. Ciorba, A. Yilmaz, C. Iwainsky, J. Doerfert, H. Finkel, and V. Kale, M. Klemm "*A Runtime Approach for Dynamic Load Balancing of OpenMP Parallel Loops in LLVM*", Poster at the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2019.
[6] Mohammed, A., Ciorba, F. M., Cavelan, A., Cabezon, R., and Banicescu, I. "*Identifying Performance Challenges in Smoothed Particle Hydrodynamics Simulations*", Poster at the Platform for Advanced Scientific Computing Conference (PASC), 2019.
[7] Eleliemy, A., Mohammed, A., and Ciorba, F. M. "*Efficient Generation of Parallel Spin-images Using Dynamic Loop Scheduling*", In Proceedings of the 8th International Workshop on Multicore and Multithreaded Architectures and Algorithms (M2A2) of the 19th IEEE International Conference for High Performance Computing and Communications (HPCC), 2017.