Impact of Data Distribution and Schedulers for the LU Factorization on Multi-Core Clusters Impact of Data Distribution and Schedulers for the LU Factorization on Multi-Core Clusters

Otho José Sirtoli Marcondes, Lucas Mello Schnorr, Phillipe Olivier Alexandre Navaux
Instituto de Informática, UFRGS

Ocotber 28th, 2025

Context

- HPC is the backbone for groundbreaking research and innovation across numerous scientific and engineering disciplines.
- Modern clusters have thousands of multi-core nodes.
- Dense linear algebra (e.g., LU factorization) is a core workload.

Introduction and Motivation

Load balancing across nodes is a critical challenge.

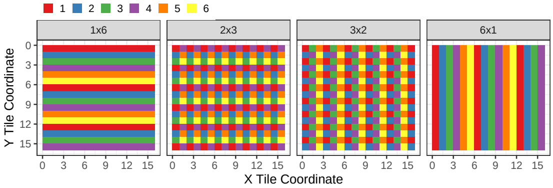
- Static data distribution: excellent data locality but lacks adaptability.
- Dynamic scheduling: great adaptability but incurs overhead.
- Hybrid strategies aim to combine both efficiently (StarPU-MPI and CHAMELEON).

Objective

- Analyze how data distribution and scheduling heuristics affect LU factorization performance.

Static Data Distribution

- Block-Cyclic (BC) used in ScaLAPACK, HPL benchmark.
- Balances computation and communication via P×Q grid.
- Limitations with prime node counts or heterogeneous resources.



Task-Based Runtime Systems

- Frameworks: StarPU, PaRSEC, OmpSs.
- Express computations as task DAGs.
- Adaptability and performance portability.
- Schedulers dynamically assign tasks.

Hardware Environment

Resource	Specification
Cluster	PCAD @ INF/UFRGS
Nodes	6
Cores per node	24 (2$\times$12 Xeon Silver 4116)
Memory per node	96 GB DDR4
Network	10G Ethernet (X540-AT2)

Software Environment

- CHAMELEON 1.3.0 for dense linear algebra.
- StarPU-MPI 1.4.7 runtime system.
- OpenMPI 4 transport layer.
- GNU Guix for reproducible package management.
- Data analysis in R suing StarVZ framework.

Application: LU Factorization

- Decomposes matrix A into L (lower) and U (upper).
- Kernels used:
  - DGEMM – matrix multiplication
  - DTRSM – triangular solve
  - DGETRF_NOPIV – LU without pivoting
- Hybrid execution:
  - Static inter-node block distribution
  - Dynamic intra-node scheduling via StarPU

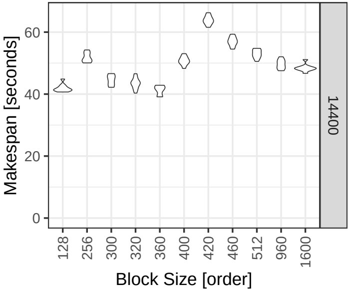
Experimental Design

Square matrix size of 14.4K (double precision).

1. Phase 1: Tile size tuning (128–1600)
2. Phase 2: Full factorial 4×4 experiment
   - Schedulers: lws, random, dmda, dmdas
   - Distributions: 1×6, 2×3, 3×2, 6×1
3. Phase 3: Detailed trace analysis with StarVZ
   - Fixed `lws` scheduler
   - Varying PxQ (BC) parameters.

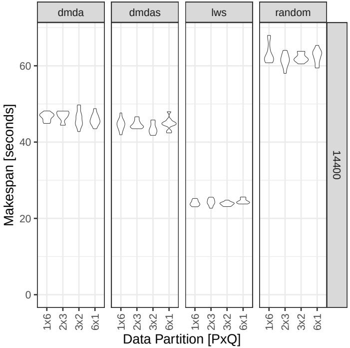
Preliminary Study

- Tested 10 tile sizes with `lws` scheduler.
- Best performance at 360×360 blocks.



Overview of the Comparison

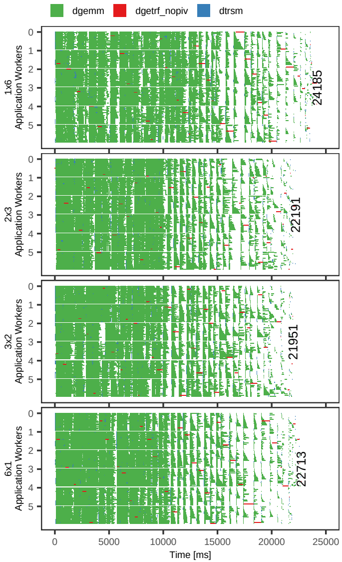
- Varying data distribution and schedulers.
- 1×6 and 6×1: $\approx$3900 MPI operations.
- 2×3 and 3×2: $\approx$2300 MPI operations.
- Similar makespans across different data distributions and schedulers.



OpenMPI Delays

- Fixed `lws` scheduler, varying data distributions.
- More dense behavior of `dgemm` tasks until 10s.
  - 1×6 (worst): mean idle $\approx$1500 ms
  - 2×3 (best): mean idle 800 ms
- Bottleneck from network latency, not algorithmic imbalance.

OpenMPI Delays



Schedulers Comparison

- Fixed 3×2 distribution, `lws` and `dmdas` schedulers.
- Per-node optimistic makespan (ABE):
  - ≈14605 ms for `lws`
  - ≈15045 ms for `dmdas`
- LWS temporal gaps between tasks ≈7841 ms; `dmdas` ≈9154 ms.
- 200 more outlier tasks in `dmdas` explain longer runtime.

Data distribution similarities

- Makespan difference <3%.
- ABE difference between nodes 1 and 2:
  - 3×2: ≈2249 ms
  - 6×1: ≈2194 ms
- Similar load imbalance between the two.

Conclusion and Future Work

- Impact of data distribution and scheduler heuristics.
- `dmda` and `dmdas` presented similar performances.
- `lws` best performing scheduler.
- Data partition (P) had minimal impact on performance.
- Scale experiments using SimGrid simulation.
- Study repetitive network delays in StarPU-MPI.

Acknowledgments

The experiments in this work used the PCAD infrastructure, http://gppd-hpc.inf.ufrgs.br, at INF/UFRGS. We also acknowledge the Brazilian National Council for Scientific Technological Development (CNPq) for their financial scholarship support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the FAPERGS (16/354-8, 16/348-8), Petrobras (2020/00182-5).

Contact

Thank you for your attention!

Otho José Sirtoli Marcondes <otho.marcondes@inf.ufrgs.br>
Lucas Mello Schnorr <schnorr@inf.ufrgs.br>
Phillipe Olivier Alexandre Navaux <navaux@inf.ufrgs.br>