Towards a scalable load balancing for productivity-aware tree-search

G. Helbecque†, J. Gmys†, T. Carneiro*, N. Melab†, P. Bouvry*

†Université de Lille, CNRS/CRIStAL UMR 9189, Centre Inria de l'Université de Lille, France
*Université du Luxembourg, DCS-FSTM/SnT, Luxembourg

10th annual Chapel Implementers and Users Workshop (CHIUW)
June 2, 2023
Beginning of the exascale era\(^1\) (June 2022);

- Increasingly large (millions of cores), heterogeneous (CPU-GPU, etc.) and less and less reliable (Mean Time Between Failures – MTBF < 1h) systems\(^1\);
- Evolutionary school (MPI+X) vs. revolutionary school (Partitioned Global Address Space (PGAS) - based environments).

\(^1\) Top500 ranking: https://www.top500.org/.
Context

Focus on exact Branch-and-Bound (B&B) optimization methods to solve combinatorial optimization problems:

- Large tree size → Efficient data structure;
- High irregularity → Efficient dynamic load balancing mechanism.

Motivating example: Permutation Flowshop Scheduling Problem (PFSP). Search trees for very hard PFSP instances contain up to $10^{15}$ explored nodes.

Fig. 3: Mapping B&B to hardware.

Fig. 4: Solution of a PFSP instance of 4 jobs and 3 machines.
State-of-the-art overview

- Most of existing parallel B&B algorithms are only guided by performance and benefit from problem-specific optimizations:
  - Multi-core CPUs: [Mezmaz2014], [Gmys2016];
  - GPU and many-core: [Chakroun2013], [Melab2018];
  - Clusters of GPUs: [Vu2016];
  - Grid computing: [Mezmaz2007], [Drozdowski2011].

- Emergence of the PGAS-based Chapel productivity-aware parallel programming language (HPE/Cray): [Callahan2004], [Carneiro2020].

- Few studies investigate the PGAS-oriented approach in the parallel optimization setting: [Machado2013], [Munera2013].
Parallel design

- Asynchronous parallel tree exploration model:
  - Unpredictable communications;
  - Unbalanced work units → Work Stealing (WS).

- Depth-First Search (DFS):
  - Memory Efficiency;
  - Stack (LIFO).

Fig. 6: Illustration of the parallel tree exploration model.
Parallel design

Collegial multi-pool [Gendron1994].

At the process level

$L_0$

$L_1$

$L_N$
DistBag^2 ("distributed bag"): parallel-safe distributed multi-pool implementation;

→ not suitable for DFS.

Revisited into DistBag-DFS:\n- Work pools → non-blocking split deques [vanDijk2014], [Dinan2009];

Fig. 8: Simplified view of a non-blocking split deque.

- New WS mechanism:
  - Bi-level (locality-aware);
  - Random victim selection;
  - Steal half.

3. The DistBag-DFS data structure: https://github.com/Guillaume-Helbecque/P3D-DFS/DistBag-DFS.
Productivity-awareness

Sequential vs. distributed parallel$^4$:

→ Few more lines of code are required;
→ Generic approach: e.g. PFSP, Unbalanced Tree-Search benchmark (UTS), N-Queens.

Experimental protocol

• **Hardware:** ULHPC facilities
  2x AMD Epyc ROME 7H12 @ 2.6 GHz (64 cores), 256 GB RAM; Fast InfiniBand HDR100 network.

• **Applications:**
  ➢ PFSP → Taillard’s instances (20 jobs x 20 machines) [Taillard1993];
  ➢ UTS → synthetic trees.

• **Experiments:**
  ➢ Memory consumption of DistBag vs. DistBag-DFS;
  ➢ P3D-DFS vs. MPI+X best known counterparts (≠ approaches).

---

5. ULHPC supercomputers - Aion system: https://hpc-docs.uni.lu/systems/aion/.
DistBag VS. DistBag-DFS

- DistBag cannot ensure DFS exploration order → poor memory efficiency;
- Memory consumption remains bounded using DistBag-DFS.

Fig. 10: Bag size according to the processing time when solving PFSP instances in DFS.

≈ 11GB
Experimental results at the inter-node level

Fig. 11: Absolute speed-up P3D-DFS vs. MPI-PBB in distributed-memory experiments.

- P3D-DFS vs. MPI-PBB (MPI+pthread);
- P3D-DFS competitive against its counterpart:
  - ≠ WS mechanisms.
- MPI-PBB performs better solving the largest instance with the finest granularity:
  - DistBag-DFS overheads (?).

94% of ideal speed-up

Experimental results at the inter-node level

- P3D-DFS vs. MPI-PUTS\(^7\) (MPI+MPI);
- P3D-DFS outperforms its counterpart, at medium- and coarse-grain:
  - \(\neq\) WS mechanisms.
- P3D-DFS outperformed at fine-grain:
  - DistBag-DFS overheads (?);
  - poor intra-node speed-up.

Conclusion & future works

- DistBag-DFS allows high-productivity and good performance for coarser-grained applications; → P3D-DFS competitive to MPI+X baselines, in terms of performance and productivity-awareness.

- Investigate and benchmark DistBag-DFS low-level mechanisms;

- Extend P3D-DFS to other combinatorial optimization problems: e.g. Quadratic assignment problems, Traveling salesman problems.

- Extend experiments to larger systems;

- Develop a Chapel's DistributedBag-DFS package module (?)

Suggestions are welcomed!
Some references


Some references


Thank you for your attention.

Contact:
Guillaume HELBECQUE
guillaume.helbecque@univ-lille.fr

https://github.com/Guillaume-Helbecque