Resource Adaptivity in the GPTune Autotuning Framework

Can Ersoy
Dominik Huber
Martin Schulz
Technical University of Munich
Garching, Germany
{can.ersoy,domi.huber,martin.w.j.schulz}@tum.de

Martin Schreiber
Université Grenoble Alpes
Grenoble, France
martin.schreiber@univ-grenoble-alpes.fr

***Abstract*—Autotuning Frameworks often consist of various phases, such as sampling, modeling, and search, each with different resource requirements. Hence, traditional, static resource management for such frameworks often leads to wasted resources since not all allocated resources are used throughout all phases of the tuning process.**

To address this inefficiency, we demonstrate the use of dynamic resource management for autotuning frameworks, building on the well-known GPTune framework. The original GPTune implementation uses MPI Spawn to launch processes for each phase within a static allocation. We adapt the implementation to use MPI Sessions and Dynamic Processes with PSets (DPP), expressing resource needs of each phase using DPP's generic, graph-based abstraction. This allows resource managers to adapt allocation to current resource needs.

Our evaluation shows that dynamic resource management can reduce the resource usage of GPTune by 46% compared to the static GPTune version.

***Index Terms*—Autotuning, Resource Management, MPI Sessions**

I. Resource inefficiency in Autotuning Frameworks

We investigate the resource inefficiency of autotuning frameworks based on the well-known GPTune library. GPTune [2] is an AI-driven autotuning framework that repeatedly executes HPC applications to find optimal parameters. The number and runtime of evaluations vary throughout the tuning process, creating dynamic resource demands. Currently, for each phase of the tuning process, GPTune spawns MPI processes within a single, static allocation, leaving resources idle during phases where not all resources are needed. Thus, a more dynamic resource management approach is required to efficiently execute autotuning frameworks on HPC systems.

II. Dynamic Resource Management as a new execution paradigm

Dynamic Resource Management (DRM) has gained significant attention as a potential means to improve the efficiency of HPC system operation. DRM allows the resource manager to dynamically adapt the resources allocated to a job during its runtime. To enable this, running jobs need to be able to indicate their dynamic resource requirements to the resource manager in a generic way, allowing the scheduler to perform global optimization.

Dynamic processes with PSets (DPP) [1], [3] provide such a generic abstraction, allowing applications to express their dynamic application structure as an evolving graph of process sets and associated reconfiguration operations.

III. Applying Dynamic Resource Management to GPTune

In our work, we leverage an MPI Sessions implementation that provides DPP functionality to make GPTune resource-adaptive. We represent the different phases of the autotuning process as PSet graphs that show the framework's current resource requirements. The resource manager can then adapt the resource allocation to fit the current resource needs.

IV. Evaluation

In our evaluation on 32 nodes, we compare the node utilization of our adapted, dynamic GPTune version with the static baseline version when autotuning a SuperLU-Dist routine. We observe that the dynamic version reduces resource usage by 46% by avoiding idle resources, at the expense of longer overall execution time.

References

[1] Huber et al. "Dynamic Resource Management in HPC Systems Using Dynamic Processes with PSets". In: 2025 IEEE 32nd International Conference on High Performance Computing, Data, and Analytics (HiPC). 2025, pp. 279–289. DOI: 10.1109/HiPC66333.2025.00036.

[2] Liu et al. "GPTune: multitask learning for autotuning exascale applications". In: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Program- ming. PPoPP '21. New York, NY, USA: ACM, 2021, pp. 234–246. DOI: 10.1145/3437801.3441621.

[3] Dominik Huber, Martin Schreiber, Martin Schulz, Howard Pritchard, and Daniel Holmes. Design Principles of Dynamic Resource Management for High-Performance Parallel Programming Models. 2024. arXiv: 2403.17107