Resource Adaptivity in the GPTune Autotuning Framework

Can Ersoy, Dominik Huber, Martin Schreiber, Martin Schulz

Technical University of Munich

Read the thesis!

1. Problem: Autotuning Frameworks have dynamic resource requirements!

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, GPTune spawns all processes inside of a **single, static allocation, leading to idle resources.**

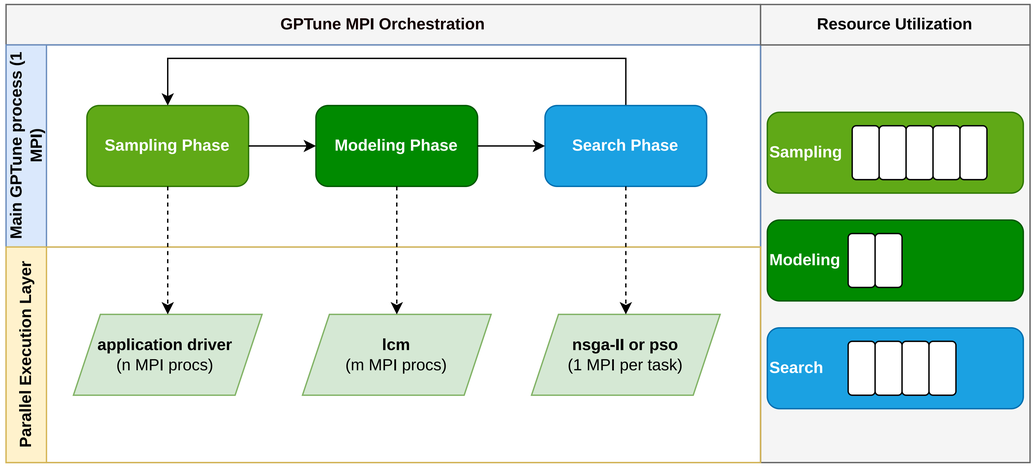


Figure 1 MPI orchestration of GPTune phases. GPTune spawns different numbers of processes for *sampling*, *modeling* and *search*.

2. Approach: Dynamic Resource Management to the rescue

Dynamic resource management (DRM):

- Allows **changing resources** allocated to a job **at runtime**.
- Requires interactions across the HPC software stack (Fig. 3)
- DPP provides a generic interaction mechanism

Dynamic Processes with PSets (DPP) [3, 1]:

- **Abstraction to express dynamic resource requirements.**
- Allows **dynamic resource reallocation** by the resource manager

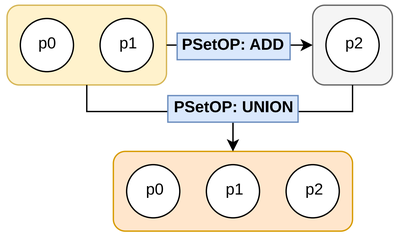


Figure 2 Graph-based abstraction for dynamic resource management. Vertices represent sets of application processes (PSets), and hyperedges represent reconfiguration operations (PSetOps).

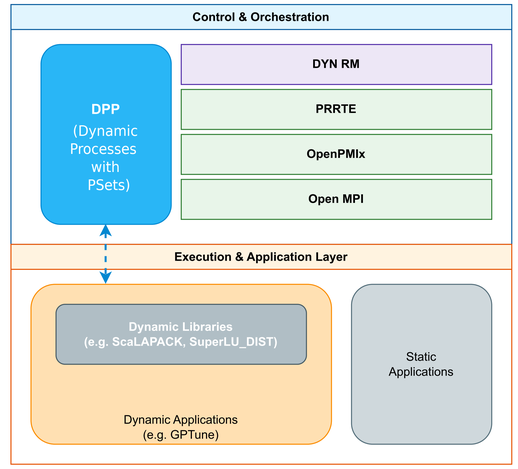


Figure 3 Software stack for Dynamic Resource management. DPP provides a common abstraction for DRM across the various layers of the HPC software stack.

3. Realization: Making GPTune resource adaptive

We use MPI Sessions and DPP to make the GPTune implementation *resource-adaptive*:

- **GPTune phases are expressed as a dynamic PSet graph** towards the resource manager (Fig. 4).
- The resource manager applies **reallocation decisions**.

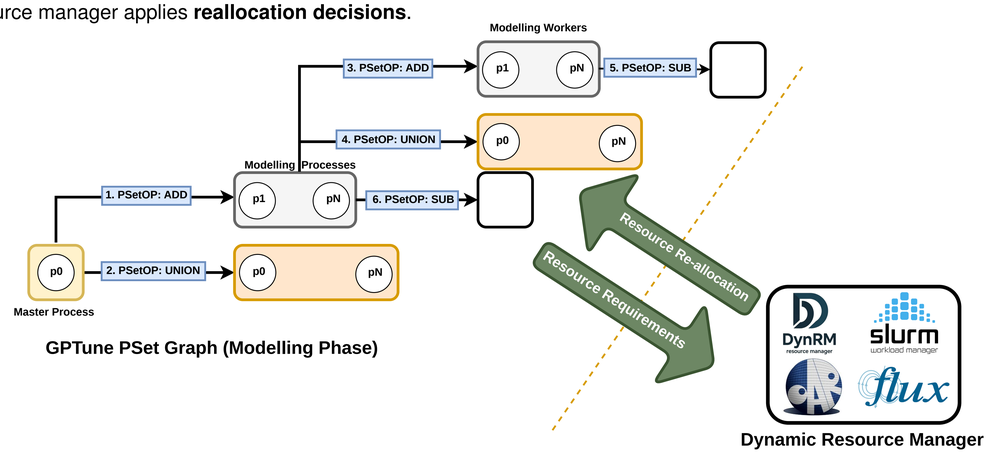


Figure 4 DPP abstraction for DRM applied to GPTune

4. Evaluation: Resource adaptivity improves resource efficiency

We compare node allocation of **dynamic GPTune (ours) vs. static GPTune (baseline)** autotuning SuperLU_DIST's PDGEQRF routine on 32 nodes (Fig. 5). **Our dynamic version reduces resource usage by** 47% **at the cost of longer total runtime**.

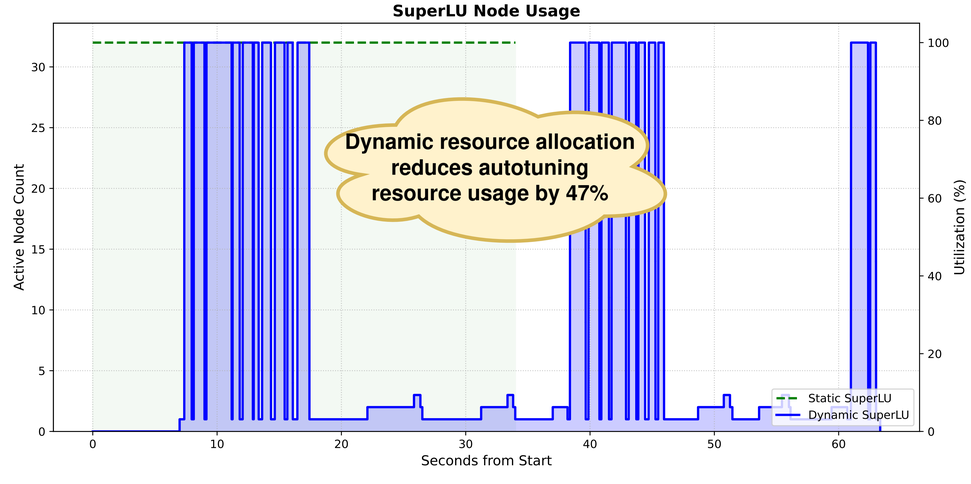


Figure 5 Node allocation over time. Blue and green areas show node allocation for dynamic and static GPTune, respectively.

[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 Programming*. 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.