Enhancing Autotuning Capability
with a History Database

Younghyun Cho
UC Berkeley

James Demmel
UC Berkeley

Xiaoye S. Li
LBNL

Yang Liu
LBNL

Hengrui Luo
LBNL

Acknowledgement

• Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.
Challenges in autotuning

• HPC applications have a huge parameter search space and are expensive to evaluate
  • Many performance data samples are needed to build an accurate surrogate performance model or to explore the search space

• Existing autotuning frameworks do not provide sufficient reusability of collected performance data samples
  • Re-using historical data is usually only possible for the same user and for the identical tuning problem
Our history database

• A shared repository and API to share autotuning performance data

Autotuning use cases
- Checkpointing/restarting
- Query the best result
- Transfer learning
- Crowd-tuning

Achieving reproducibility
- Human readable (JSON)
- Record information for reproducibility (e.g. software, machine)

Useful interfaces
- Interactive dashboard
- Programmable API

Performance data from autotuning

Shared repo.
https://gptune.lbl.gov
Implementation in GPTune

• GPTune is a performance autotuner based on Gaussian Process Regression [1]

- Initial sampling
- Surrogate modeling
- Search next sample

- Expected improvement

• The history database interface is built into GPTune (we believe the history database can also be used for other autotuners too)

Design of the history database

- Machine configuration (e.g. #nodes, #cores)
- Software configuration (e.g. dependencies, versions)

Reproducibility information (JSON or Python Dict)

GPTune autotuning framework

Surrogate Modeling → Search next sample

Save/Load
- Function evaluation results
- Surrogate performance models

Recording info.
- User input
- Automatic parsing

User’s local storage

 JSON File(s)

Upload/Download
- Interactive Dashboard
- Programmable API

Shared Repository
https://gptune.lbl.gov

MongoDB
(perf. data)

SQLite
(user data)
Database layout

Example: gptune.db/PDGEQRF.json

```
{
    "func_eval": [ 
        { ... },  # each function evaluation result
        { ... }
    ],
    "surrogate_model": [ 
        { ... },  # each surrogate model
        { ... }
    ]
}
```

Reproducibility information

Example function evaluation result

```
{
    "task_parameter": { "m": 10000, "n": 10000 },
    "tuning_parameter": { "mb": 6, "nb": 9, "nproc": 5, "p": 203 },
    "evaluated_result": { "r": 9.94401 },
    "machine_configuration": {
        "machine_name": "Cori",
        "Haswell": { "nodes": 1, "cores": 32 }
    },
    "software_configuration": {
        "scalapack": { "version_split": [2,1,0] }
    }
}
```

Example surrogate model

```
{
    "hyperparameters": [ 1.59484,
        1295127.9634998, ... ],
    "model_stats": {
        "log_likelihood": -22.19576,
        "gradients": [ -9.37384, -
            7.43426, ... ],
        "iteration": 77
    },
    "modeler": "model_LCM"
}
```
Auto-parsing reproducibility information

- **Software configuration**: Automatic if the software is installed using an automation tool such as CK [2] or Spack [3]
- **Machine configuration**: Automatic if the HPC job is using a specific resource manager such as Slurm [4]
- Automatic parsing of such environment information is still evolving


Shared repository (https://gptune.lbl.gov)

- The database is growing!
  - Support various ECP software packages (e.g. ScalAPACK, SuperLU_DIST, Hypre, etc.)
  - Plan to use the database for other domains of tuning problems (e.g. ML algorithms, randomized linear algebra, etc.)
Data sharing policy

• Registered users can upload performance data
  • Options for users who do not wish to display their information

• Options for choosing a different level of accessibility
  • Publicly available
  • Sharing with registered users or specific users/groups
  • Private
Case study: Transfer learning in GPTune

- Leverage knowledge of historical data trained from different problem sizes or different machine/software configurations

![Diagram]

Leverage GPTune’s built-in multitask learning technique [1] to leverage knowledge from task B

Case study: Transfer learning (cont’d)

- Tuning ScaLAPACK’s PDGEQRF on 8 and 64 Intel Haswell nodes in NERSC Cori supercomputer
  - Comparing the performance of the best tuning result for three input sizes from 20 evaluations

- Comparisons
  - SLA: single-task learning
  - TLA (worst): worst historical dataset was used
  - TLA (best): best historical dataset was used

- Intelligent TLA dataset selection is future work
Summary

• Our history database is designed for sharing performance data to enhance autotuning capabilities
  • High reusability and reproducibility of collected performance data
  • Can be used for transfer learning and crowd-tuning

• Ongoing and future work
  • Our community is growing. We are collecting performance data samples from various ECP software packages (e.g. xSDK (https://xsdk.info)).
    • We are seeking external users to join our community
  • Continuing research on improving transfer learning methodologies
Thank you for your attention!

• Resources
  • History database repository: https://gptune.lbl.gov
  • History database user guide: https://gptune.lbl.gov/docs
  • GPTune Github: https://github.com/gptune/GPTune
  • GPTune user guide: https://gptune.lbl.gov/documentation/gptune-user-guide/

• Team email: gptune-dev@lbl.gov