Analysis and Visualization of Important Performance Counters To Enhance Interpretability of Autotuner Output

Mohammad Zaeed[1], Tanzima Z. Islam[1], Younghyun Cho[3], Xiaoye Sherry Li[2], Hengrui Luo[2], Yang Liu[2]
[1]Texas State University , [2]Lawrence Berkeley National Laboratory, [3]University of California, Berkeley

ABSTRACT

Autotuning is a widely used method for guiding developers of large-scale applications to achieve high performance. However, autotuners typically employ black-box optimizations to recommend parameter settings at the cost of users missing the opportunity to identify performance bottlenecks. Performance analysis fills that gap and identifies problems and optimization opportunities that can result in better runtime and utilization of hardware resources. This work combines the best of the both worlds by integrating a systematic performance analysis and visualization approach into a publicly available autotuning framework, GPTune, to suggest users which configuration parameters are important to tune, to what value, and how tuning the parameters affect hardware-application interactions. Our experiments demonstrate that a subset of the tuning parameters impact the execution time of the NIMROD application for a specific task; the Plasma-DGEMM routine spends a significant amount of time waiting for responses from the offcore L3 resource on Cori-Haswell.

BACKGROUND

- GPTune is an online autotuning framework for suggesting user optimal parameter configurations using Bayesian optimization methodologies.
- DASHING is a performance analytics tool for analyzing and visualizing application-system interactions collected via performance counters.

CHALLENGES

- Large number of performance counters describing application-system interactions makes it hard for users to understand bottlenecks.
- Analysis and visualization of performance needs to occur on a real-time web interface.

Approach

- Group counters according to resources they pertain to.
- Identify important groups that can predict the objective function, e.g., execution time, accurately.
- Visualize the information in a hierarchical manner to enable a comparative view of the relative importance across groups.

Analysis from a simulated application

METHODOLOGY AND PRELIMINARY RESULT

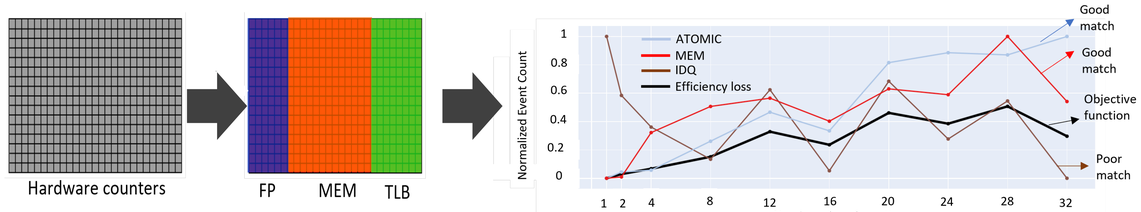


Fig 1. Divide performance counters into meaningful groups

Compute counter Importance

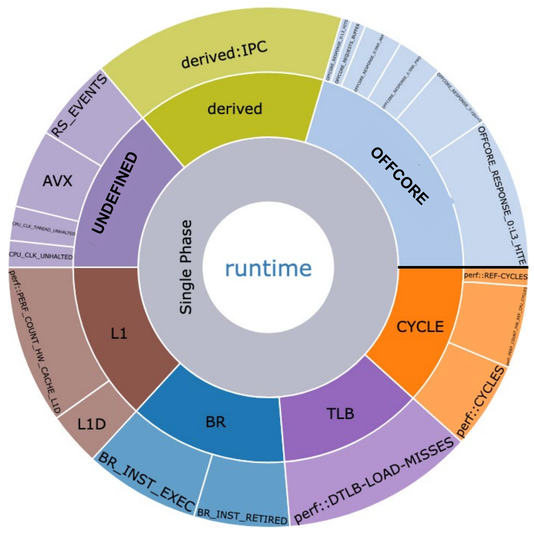
- Compute prediction error $\mathbf{e}_i = \|\mathbf{t} - \mathbf{d}_i a_i\|_2^2$
- Convert error to "Belief" $\alpha_i = exp(-\mathbf{e}_i)$

Compute resource importance

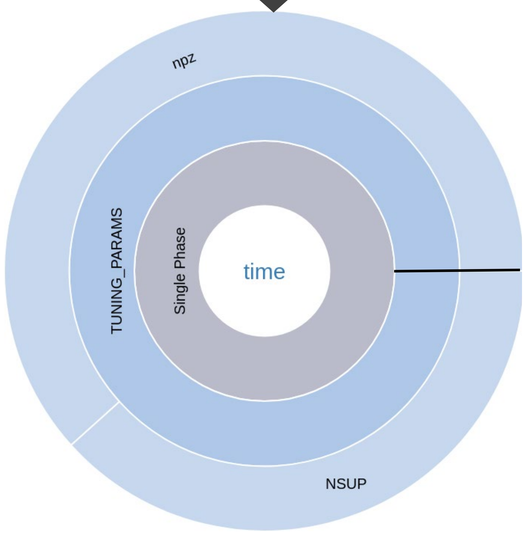
- Accumulate beliefs from all selected counters per resource

$$\text{Group Importance} = 1 - \prod_{j \in L_r} (1 - \alpha_j)$$

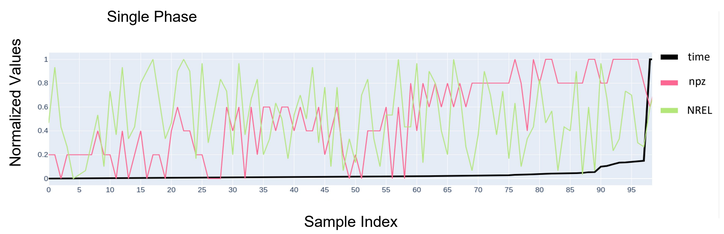
Fig 2. Calculate a belief score for every group to show how well it can explain the objective function



(a) Analysis on PLASMA-DGEMM kernel's performance counters



(b) Analysis on NIMROD's Tuning parameters



(c) Plotting the dataset to validate the importance calculations

Fig 3. Visualize with interactivity

OBSERVATIONS

- (From Fig 3a) One can infer OFFCORE to be the most important resource (has the largest area). The DTLB load misses is an important event, hence, users can increase the TLB page sizes to reduce this event.
- (From Fig 3b and 3c) npz and NSUP are the most important parameters for the NIMROD application. The rightmost figure shows that both runtime and npz increase along the X-axis, while NREL shows unpredictable behavior. Therefore, decreasing the value of npz will decrease the runtime of NIMROD.

Future Work

- Visualize sensitivity of applications on the tuning parameters.
- Make the on-line importance analysis and visualization code available via GPTune's public repository.

Acknowledgement

This project was partially funded by the Research Enhancement Program at Texas State University, AMD, and Lawrence Berkeley National Laboratory.