Automatic Feature Detection in Hardware Performance Metric Data based on User Feedback

Sayef Azad Sakin

Abstract— Understanding large scale runtime system behavior becomes harder as the hardware performance metric data grows significantly over time. Identifying relationship within different performance metric could be accessory to assessing and evaluating these data. The Traveler project provides an integrated visualization system for visualizing performance metric data. It uses OTF2 execution traces and present them over several interactive views like - Gantt view, Utilization view, Time series view etc. Highlighting key features from these views is crucial to artifact detection and optimization of a runtime system. Therefore, providing prediction of different features in advance will be helpful for user to browse, lookup, and identify them. Also the prediction could be useful for any backend caching mechanism to load data in advance, hence improving user interactivity. In this project, an automated learning based feature prediction model has been proposed for analysing temporal performance metric data labeled by the user and providing prediction of interesting features for any future performance data. A dedicated python class has been developed for the Traveler platform to provide the feature prediction, which could be utilized by any backend data model or directly integrated as a module with the frontend controller.

1 Overview

Large-scale parallel applications have become the backbone of a wide variety of scientific and technological research [2, 10]. Deciphering features from the runtime system of these appications as well as optimizing them is a significant and largely unsolved problem [7]. Optimizing resources related to time-oriented process scheduling is essential for productivity. Time series visualization provides a practical medium for exploratory visual data analysis of these parallel systems [1]. Typically, high-end scientific simulations require time consuming and intensive operations, incurring massive temporal dataset to decipher. Interactive visual data exploration of these massive temporal dataset is becoming increasingly challenging due to the rapid generation of data, in the order of terabytes or petabytes. Therefore, an automated feature recommender system of a temporal dataset is essential for exploratory data analysis. It will also give clues on how different hardware metrics change over time, which could be valuable information for scientists required to run, analyze, and tweak high-end time consuming simulations.

Phylanx [20] is a general purpose computation system of distributed arrays to enhance operations over large-scale distributed system. It is an actively-developed system to provide the benefits of distributed resources for data scientists. Phylanx first translates the code into PhySL, an intermediate representation in a domain-specific, functional language. The function calls, control flow operations, data operations, and blocks found in the PhySL representation are referred to as primitives. These primitives are translated to tasks in the HPX, a C++ standard library and asynchronous tasking runtime. The dependencies of each primitive are the arguments it needs to execute (which may be other primitives), data access operations, or any other constraints on variables the primitive uses. Temporal execution information of Phylanx is printed on a trace file and visualized using Traveler.

The Traveler [6] is an integrated graph visualization system for asynchronous multi-task execution. It uses OTF2 [8] [14] trace data generated from Phylanx execution, parse them, and represent them in different interactive views—Gantt view, Utilization view, Tree view, and Time series view, etc. Due to the massive number of temporal data, the loading time of these views becomes slower and conspicuous lagging becomes apparent during interactive inspection over these views.

• *Sayef Azad Sakin is a graduate student at the University of Arizona. E-mail:sayefsakin@email.arizona.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

The goal of my project is the identification and discovery of interesting features—like cache overflow, unnecessary memory overhead, CPU exhaustion or starvation, etc .—from the temporal dataset of hardware performance metric. In the aim of achieving this goal, an automated learning based recommender system has been developed to predict and recommend interesting features which will be useful for further analysis of the runtime system (the trace data it is used to learn from). The contributions of my project are summarized as follows:

- Recommender Engine modeling - The dual space model (DSM) [12] proposed by Huang et al. has been adapted to design a recommender engine.
- Feature abstraction - Temporal data of a performance metric from different hardware locations have been converted into individual input features for the recommender engine.
- Data adapter - An intermediate parser has been designed to receive data from the Traveler platform to the recommender engine, parse them and sent back through a convenient interface.
- Performance evaluation - An elementary performance evaluation has been conducted to assess the practicality and feasibility of the designed recommender model.

2 Related Works

Large-scale web-based pan/zoom visualizations developed by Tao et al. [19] is more relevant to my project. Their technique was mostly to predict next pan/zoom location which required less intensive learning based techniques. They developed a declarative language for easy specification of user behavior and data management model. Battle et al. [5] did thorough investigation of explorative data analysis by showing comparison between latency, task complexity, and user performance. An earlier work [3] from the same author analysed different statistical characteristics (e.g. histogram) among multiple user interactions over temporal dataset. They also proposed a general purpose tool, *ForeCache* [4], for exploratory browsing of large dataset. By learning first from the user's movements, they prefetched the data using different statistical data characteristics to find similarity to the user's past behavior. Their data model and the prediction engine design has been considered a baseline for measurement in visual data exploratory domain. Liu et al. [17] proposed a specialized data structure called multivariate data tiles for dynamic loading and pre-processing of data. They also synthesis the data for better representation and processing parallelization. Huang el al. [12] [11] used active learning framework to leverage the subspatial context and conjunctive context of database queries. Their developed dual space model algorithm able to bypass the slow convergence problem of active learning techniques. In my

project, most of their proposed mechanisms have been adapted due to the high similarity of the problem domain and use cases. *Skyrec* [15] of Liu et al. used query session summary method to ease-up the query recommendations. They also developed a frontend interface named *SkyServer Surfliner* for interactive recommendation.

3 Technical Details

Most of the approaches of my project is adapted from huang et al. [11] dual space method (DSM).

3.1 Data Space

With the bundling process, different OTF2 traces are being parsed and loaded in the Traveler. The trace file I worked with was generated from a fibonacci calculation of size 23 on phylanx system. The contents of a OTF2 trace data can be printed using *otf2-print* command and read using *less* [13] on unix environment from the root directory of the project.

```
otf2-print data/20190906-lra-t16/OTF2_archive/\
APEX.otf2 | less -S
```

Figure. 1 represents a sample print of the OTF2 trace data. This sample data has been used in this milestone to study and test different components of traveler-integrated project. In this trace file, each row contains the following attributes:

- Event (Categorical) : {ENTER, LEAVE, METRIC}, Indicating the arrival (ENTER) and departure (LEAVE) event of a particular primitive and relevant metric (METRIC) values for the preceding rows event.
- Location (Categorical) : The hardware core number starting from 1,2,..., etc. - where the actual processing executed.
- Timestamp (Quantitative) : Actual temporal data in nanoseconds.
- Additional Attributes : This column contains some additional categorical attributes like GUID; parent GUID; primitive name; metric id, name, value, etc.

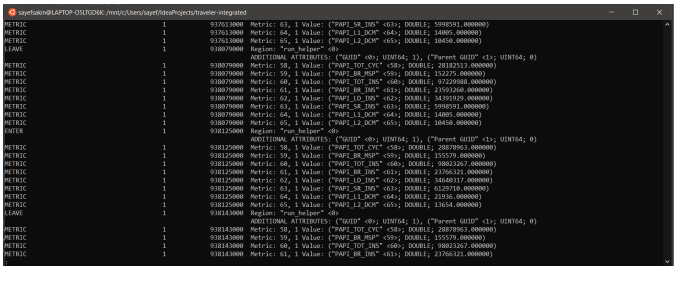

Fig. 1. Sample print of the OTF2 trace data

In the frontend of the Traveler platform, when user comes to browse the dataset, a time series chart is presented for the performance metric data (METRIC values in the OTF2) as Figure. 2.

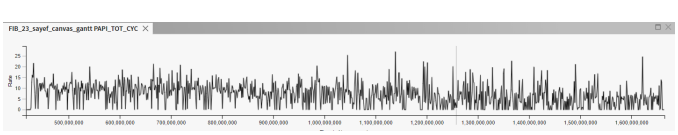

Fig. 2. CPU cycle rate

Here, the x-axis represents time in milliseconds and y-axis represents corresponding metric rate (PAPI_TOT_CYC in Figure.2). For PAPI_TOT_CYC, the corresponding value is monotinically increamental data. Let, at CPU thread location l, X_j^l be the value for leave event at timestamp T_j^l, X_i^l be the value for enter event at timestamp T_i^l, then the metric rate for CPU thread l is,

$$r_j^l = \frac{abs(X_j - X_i)}{T_j - T_i}$$

Here, the value for r_j^l is plotted on y-axis for time value T_j^l.

Therefore, for all CPU thread location $l \in L$, the set of rate at particular time i is $R_i = \{r_i^l\}, l = 1, \ldots L$. This can be projected as a tuple which is mapped on a L-dimensional space. The resulting data space is where the user exploration will take place. This has been used as input feature set for the recommender model.

3.2 Initial data labeling

A user feedback is incorporated with the tuple as a output label from the set $\{-1, 0, 1\}$, where 1 depicts interesting, -1 being not interesting and 0 represents unlabeled by the user. This can help to bootstrap the data exploration. If it is not possible to collect user feedback, the system can run initial sampling described in [16] [9], or use some statistical measurements. During my training phase, as I didn't have access to collect user feedback, I used standard deviation with certain threshold to label the data tuples.

3.3 Active Learning

To dynamically find the next tuple for labeling, I used Support Vector Machine (SVM) [18] as a classification model. The unlabeled samples can be labeled with the prediction of the SVM. Here, uncertainties are measured by the distance of each example from the decision boundary of the SVM. More description about the usage of the classifier can be found in [11].

3.4 Convergence Algorithm

The convergence algorithm utilizes the dual space model described in [12]. Here, to update the positive region, convex hull of the positive example points are used and checked if it is inside the hull. For the negative example, I checked if it is outside the convex hull generated from the negative example points. Although huang et al. used convex cone generated by the conical combination of the vectors from the positive example to the given negative example. The reason behind this is that, the outside region of a convex hull can eliminate regions more faster than the conve cone (since it covers more area than the convex cone).

4 Evaluation

Due to the lack of real users presence, a comprehensive real life user feedback cannot be collected for evaluation. After completing the training, I've presented an accuracy result, which is the ratio of the number of positive region data to unknown region data. Due to the faster elimination technique described in section3, my model quickly removed the unlabeled data (hence, label them with wither 1 or -1). The iterative learning process is terminated based on the accuracy achieved by this process. More details and reasoning behind this approach can be found in [12].

After completing the learning process, another evaluation accuracy result is printed, which is calculated as follows:

$$\alpha = \frac{\text{predicted labels equal to the user label}}{\text{total number of labeled data by user}}$$

In the table 1, corresponding evaluation accuracy is presented. Note that, in this case, the total number of CPU thread location was fixed at 10. Here, the accuracy drops if the λ value reduced, which is evidential that, the algorithm was being stopped prematurely (with any required training). Again, increasing the time range also helps to increase the accuracy. Real inspection on the time series visualization revealed that, this is because of the higher number of negative label data which owes to the fixed bin size that eliminated some of the interesting data points.

It is obvious that, measuring evaluation accuracy in this way is not representative of what actually being learned. Appropriate labels

Table 1. Evaluation Results

Epochs	Begin	end	bins	λ	α
44	1242233747	1664725001	1000	0.8	0.519
62	416663000	1664725001	2000	0.6	0.713
56	972219900	1664725001	2000	0.6	0.625
40	416663000	972219900	2000	0.5	0.503

generated from an exhaustive user case study would be helpful to actually measure the real performance of the recommender model.

5 Future Direction

There are several limitations and pending modifications exist in my project which will be a good starting point for further work in the future. First of all, an extensive user data collection should be conducted to make the prediction engine more valid and practically representative. A seperate script can be used to either get new user input or incorporate old user data for the recommender system. Secondly, the implemented dual space model only initiates with the labeled user data once prior to the iterations. The original DSM implementation used a probabilty function to choose between labeled user data and DSM predicted data in successive iterations as the input for the consecutive iterations. This uncertainty sampling technique is necessary to reduce the version space, the space of all configuration of the classification model consistent with the labeled data, for enabling an approximation of the optimal algorithm. Thirdly, The increased number of hardware locality in a large-scale systems can easily exhaust the feature dimension. Therefore, factorization techniques should be adapted to reduce the high-dimensional data space to a lower-dimensional spaces. Lastly, a comprehensive user study should be conducted to evaluate the performance of the recommender system. Also, how well the prediction aligned with real life hardware artifacts can also be tracked and used to create a validation dataset for the system.

6 Conclusion

Learning based interactive database exploration is an ongoing research problem in scientific community. I started working on this project to make the data exploration of Traveler more interactive and convenient for the user based on an active learning based prediction technique. After doing some literature study, I've came to know about state-of-the art works like *ForeCahce* [4], *Kyrix* [19], *Skyrec* [15], *AIDEme* [11], etc. I designed a parser to fetch live formatted OTF2 performance data from the Traveler. To make the data labeling easier, a separate script is written to label the dataset and present to the recommender model. The parsed data is formated accordingly to the feature abstraction described in Section3. Then by following the DSM algorithm, the basic classifier engine has been developed with slight tweaking according to the domain needs. A conveninent interface has been provided to make prediction feedback available towards the Traveler system. Lastly, A priliminary performance evaluation has been conducted to assess the engine on how well it predicted based on the labelling provided by the user. A comprehensive report has been created and necessary descriptions provided in this report with the insights learned through this project.

References

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of time-oriented data*. Springer Science & Business Media, 2011.

[2] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, et al. The opportunities and challenges of exascale computing. *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, pp. 1–77, 2010.

[3] L. Battle, R. Chang, and M. Stonebraker. Dynamic generation and prefetching of data chunks for exploratory visualization. *IEEE InfoVis Posters Track*, 2014.

[4] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *Proceedings of the 2016 International Conference on Management of Data*, pp. 1363–1375, 2016.

[5] L. Battle, R. J. Crouser, A. Nakeshimana, A. Montoly, R. Chang, and M. Stonebraker. The role of latency and task complexity in predicting visual search behavior. *IEEE transactions on visualization and computer graphics*, 26(1):1246–1255, 2019.

[6] A. Bigelow, K. Williams, and K. Isaacs. Traveler integrated. `https://github.com/alex-r-bigelow/traveler-integrated`, 2019.

[7] J. Daly, J. Hollingsworth, P. Hovland, C. Janssen, O. Marques, J. Mellor-Crummey, B. Miller, D. Quinlan, P. Roth, M. Schulz, et al. Tools for exascale computing: Challenges and strategies. In *ASCR Exascale Tools Workshop*, 2011.

[8] O. developer community. Open trace format version 2 (OTF2), 2019. doi: 10.5281/zenodo.3356709

[9] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 517–528, 2014.

[10] A. Hoisie, D. Kerbyson, R. Lucas, A. Rodrigues, J. Shalf, and J. Vetter. Ascr modeling and simulation of exascale systems and applications workshop. Technical report, Technical report, United States Department of Energy, 2012.

[11] E. Huang, L. Palma, L. Cetinsoy, Y. Diao, and A. Liu. Aideme: An active learning based system for interactive exploration of large datasets. In *NeurIPS 2019*, 2019.

[12] E. Huang, L. Peng, L. D. Palma, A. Abdelkafi, A. Liu, and Y. Diao. Optimization for active learning-based interactive database exploration. *Proceedings of the VLDB Endowment*, 12(1):71–84, 2018.

[13] M. Kerrisk. *The Linux programming interface: a Linux and UNIX system programming handbook*. No Starch Press, 2010.

[14] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, et al. Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir. In *Tools for High Performance Computing 2011*, pp. 79–91. Springer, 2012.

[15] J. Liu, Z. Zolaktaf, R. Pottinger, and M. Milani. Improvement of sql recommendation on scientific database. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*, pp. 206–209, 2019.

[16] W. Liu, Y. Diao, and A. Liu. An analysis of query-agnostic sampling for interactive data exploration. *Communications in Statistics-Theory and Methods*, 47(16):3820–3837, 2018.

[17] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, vol. 32, pp. 421–430. Wiley Online Library, 2013.

[18] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[19] W. Tao, X. Liu, Y. Wang, L. Battle, Ç. Demiralp, R. Chang, and M. Stonebraker. Kyrix: Interactive pan/zoom visualizations at scale. In *Computer Graphics Forum*, vol. 38, pp. 529–540. Wiley Online Library, 2019.

[20] R. Tohid, B. Wagle, S. Shirzad, P. Diehl, A. Serio, A. Kheirkhahan, P. Amini, K. Williams, K. Isaacs, K. Huck, et al. Asynchronous execution of python code on task-based runtime systems. In *2018 IEEE/ACM 4th International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*, pp. 37–45. IEEE, 2018.