Learning Heterogeneous Cloud Storage Configuration for Data Analytics

Ana Klimovic
Stanford University
anakli@stanford.edu

Heiner Litz
UC Santa Cruz
hlitz@ucsc.edu

Christos Kozyrakis
Stanford University
kozyraki@stanford.edu

1 INTRODUCTION AND MOTIVATION

The public cloud, with its promise of elasticity and reduced total cost of ownership, is experiencing unprecedented growth. However, performance and cost efficiency are only achieved by choosing a suitable configuration for each given application.

For data-intensive analytics commonly hosted in the cloud, the choice of storage is essential. Cloud vendors offer a wide variety of options: object, file and raw block storage. A block storage volume can be supported by one or more hard disks (*HDD*), a solid-state drive (*SSD*), or a high bandwidth, low-latency NVMe Flash device (*NVMe*). The devices may be local (*l*) to the cloud instances running the application or remote (*r*). These options alone lead to storage configurations that can differ by orders of magnitude in terms of throughput, latency, and cost per bit. Storage options continue to diversify as 3D X-point-based technologies emerge [6, 11].

The large number of instance and storage configuration options available make it challenging to select the right resources for an application. Even if we limit ourselves to a single VM type, a few storage options, and focus only on performance, Figure 1 shows the choice of storage is non-trivial. The first workload is I/O-bound and benefits from the higher throughput of local NVMe Flash whereas BigBench-q3 is CPU-bound. The third application is I/O-bound but performs best with the hybrid storage option since this configuration minimizes the interference of read and write operations, which have asymmetric performance on Flash [12]. Part of the challenge is that analytics workloads access multiple data streams, including input/output files, logs, and intermediate data (e.g., shuffle data). Each stream has distinct data access patterns and lifetime, which make different streams suitable for different types storage mediums. In addition to storage, users must choose from a variety of VM types to set the number of cores, amount of memory and network bandwidth in their cluster. These choices impact storage and must be considered together.

To navigate the complex cloud configuration space, we present *Selecta*, a tool that learns near-optimal VM and storage configurations for analytics applications and makes recommendations to satisfy user-specified performance-cost objectives. Selecta targets analytics jobs that are continuously re-run on newly arriving data (around 40% of jobs in large production clusters [1, 9, 15]). Selecta predicts application performance across all candidate configurations using latent-factor collaborative filtering, a machine-learning technique commonly used in recommender systems [3, 4, 17] and recently applied to cluster scheduling [7, 8]. Selecta leverages performance data from training application runs on various cloud configurations, as well as execution time of the target application profiled on only two configurations. Hence, Selecta learns significantly faster and more cost-effectively than exhaustive search. The approach also improves on recent systems such as CherryPick and Ernest whose performance prediction models require more information about the target application and hence require more application runs to converge [2, 22]. Moreover, past work does not consider the heterogeneous cloud storage options or the varying preferences of different data streams within each application [23].

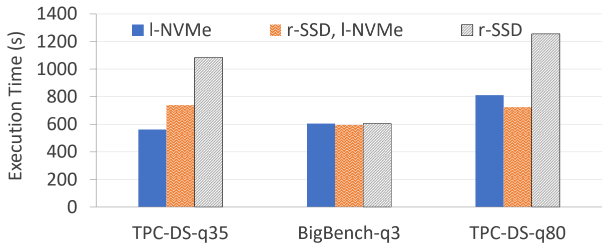

Figure 1: Performance of three applications on 8-node cluster of `i3.xl` instances with different storage configurations.

We evaluate Selecta with over one hundred Spark SQL and ML applications, showing that Selecta chooses a near-optimal performance configuration (within 10% of actual optimal) with 94% probability and a near-optimal cost configuration with 80% probability.

2 SELECTA DESIGN

As shown in Figure 2, to make recommendations, Selecta takes as input: i) execution time of training applications, each profiled on several configurations, ii) execution time for the target application on two reference configurations, and iii) the performance-cost objective. Selecta uses collaborative filtering to predict target application performance on the remaining candidate configurations. A configuration defines the number of nodes, the CPU cores and memory per node, as well as the storage type and capacity used for input/output data and intermediate data streams – Selecta considers heterogeneous storage configurations for the different streams. Given unit time costs, Selecta uses its predictions to recommend a configuration that optimizes the performance-cost objective. For example, Selecta can recommend configurations that minimize execution time, cost, or execution time within a budget. Application performance on recommended configurations is fed back to improve future predictions and reduce measurement noise.

Prediction approach: Selecta uses a latent factor model for collaborative filtering. The model characterizes configurations and applications in terms of latent (i.e., 'hidden') features [3]. These features are *automatically inferred* from training application performance data [16]. We use Singular Value Decomposition (SVD) to uncover latent factors of a sparse input matrix in which each element p_{ij} represents the normalized performance of application i on configuration j. Since SVD requires a fully populated input matrix, we randomly initialize missing entries and run Stochastic

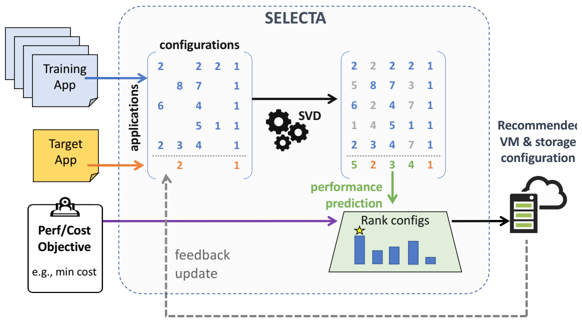

Figure 2: An overview of performance prediction and configuration recommendation with Selecta.

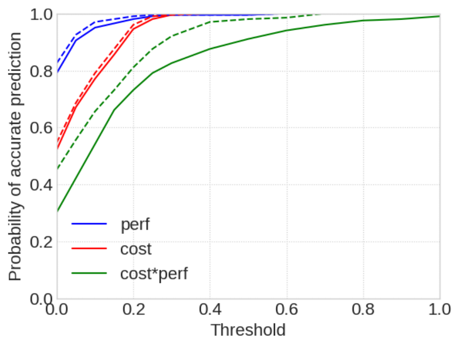

Figure 3: Probability of accurate recommendation for perf, cost and cost*perf within a threshold from optimal. Dotted lines show accuracy after a feedback iteration.

Gradient Descent (SGD) with an objective function that minimizes the mean squared error on known matrix entries [5]. Although the model does not tell us what the latent factors physically represent, a hypothetical example of a factor Selecta might infer is sequential I/O throughput. This factor would correlate how much an application's performance depends on sequential I/O throughput and the sequential I/O throughput each configuration provides.

We chose collaborative filtering since the approach works well with sparse inputs and is agnostic to the application type [16]. Alternative, content-based approaches such as linear regression, random forests, and neural networks build a model based on features of the application (e.g., bytes read/write) and configuration (e.g., CPU cores per VM). We find that such predictors do not provide sufficient accuracy unless they rely on features like CPU utilization which require running the target application on each target configuration.

Using the tool: Each new target application is profiled on two reference configurations. Profiling involves running the application to completion and recording execution time and CPU utilization (including iowait). Feedback helps Selecta detect when a recurring application's properties have changed and a new recommendation is needed. In addition to picking the storage type (e.g., SSD vs. NVMe), Selecta allocates the right capacity based on statistics of bytes read/written from run logs [20].

3 SELECTA EVALUATION

Methodology: We consider 17 different 9-node configurations in Amazon EC2 using the `i3` and `r4` instance families, `xlarge`, `2xlarge` and `4xlarge` instances sizes, combined with the following storage options: *r*-HDD, *r*-SSD, *l*-NVMe, and S3 object storage. We consider Spark [24] as a representative data analytics framework and evaluate Selecta with over one hundred SQL and ML applications, each with two different dataset scales (204 workloads in total). We use TPC-DS, TPC-BB, terasort and a SQL equijoin query as benchmarks [10, 13, 14, 21]. We run each application on all candidate configurations to obtain the ground truth performance and optimal configurations for each application. To account for noise in the cloud [19], we run each experiment (i.e. each application on each candidate configuration) 3 times and take the mean. We train and test Selecta using leave-one-out cross validation [18].

Prediction accuracy: Selecta predicts performance with a relative RMSE of 36%, on average across applications. To understand how effectively Selecta's performance predictions translate into recommendations, in Figure 3 we plot recommendation accuracy for performance, cost and cost*performance objectives. The plots shows the probability of near-optimal recommendations as a function of the threshold defining what percentage from optimal is considered close enough. Using a threshold makes our evaluation more robust to noise and allows us to make more meaningful conclusions about Selecta's accuracy, since a second-best configuration may have similar or significantly worse performance than the best configuration. When searching for the best performing configuration, Selecta has a 94% probability of recommending a configuration within 10% of optimal. For a minimum cost objective, Selecta has a 80% probability of recommending a configuration within 10% of the lowest cost. Predicting cost*performance is more challenging since errors in Selecta's relative execution time predictions for an application across candidate configurations are squared: $cost * perf = (execution\_time)^2 * cost\_per\_hour$. The dotted lines in Figure 3 show how accuracy improves after one feedback round.

Sensitivity Analysis: Training matrix rows only need to be around 25% dense for accurate predictions at steady state. To jump start Selecta with dense training data and achieve desirable accuracy, a user needs to provide 2.5× more training applications than the number of candidate configurations.

4 INSIGHTS FROM ANALYSIS

Our analysis leads to several key insights about cloud storage systems and their use by analytics workloads. First, NVMe-based configurations offer not only high performance but also low execution cost for a wide range of applications. Second, our analysis motivates cloud storage that supports fine-grain capacity and bandwidth allocation along with high throughput. Finally, there is a need for end-to-end cloud storage optimization – application frameworks, operating systems, and cloud services – as several configurations under-perform due to inefficiencies in the storage stack.

5 CONCLUSION

As data-intensive workloads grow in complexity and cloud options for compute and storage increase, we believe that tools like Selecta, which learn near-optimal cloud configurations based on previous performance data, will become increasingly useful for end users, systems researchers, and even cloud providers (e.g., for scheduling 'serverless' application code).

REFERENCES

[1] Sameer Agarwal, Srikanth Kandula, Nicolas Bruno, Ming-Chuan Wu, Ion Stoica, and Jingren Zhou. Re-optimizing data-parallel computing. In *Proc. of the USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 21–21, 2012.

[2] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. CherryPick: Adaptively unearthing the best cloud configurations for big data analytics. In *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 469–482, 2017.

[3] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 95–104, 2007.

[4] Robert M. Bell, Yehuda Koren, and Chris Volinsky. The BellKor 2008 Solution to the Netflix Prize. Technical report, 2008.

[5] Léon Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent*, pages 177–186. Physica-Verlag HD, 2010.

[6] IBM Corporation. Intel Optane SSD DC P4800X Available Now on IBM Cloud. https://www.ibm.com/blogs/bluemix/2017/08/intel-optane-ssd-dc-p4800x-available-now-ibm-cloud, 2017.

[7] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, pages 77–88, 2013.

[8] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and qos-aware cluster management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 127–144, 2014.

[9] Andrew D. Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. Jockey: Guaranteed job latency in data parallel clusters. In *Proc. of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 99–112, 2012.

[10] Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1197–1208, 2013.

[11] Intel. Intel optane technology. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html , 2017.

[12] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. Reflex: Remote flash == local flash. In *Proc. of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 345–359, 2017.

[13] High performance IO Research Group at IBM Research Zurich. Example terasort program. https://github.com/zrlio/crail-spark-terasort, 2017.

[14] High performance IO Research Group at IBM Research Zurich. Spark SQL benchmarks. https://github.com/zrlio/sql-benchmarks, 2017.

[15] Adrian Daniel Popescu, Vuk Ercegovac, Andrey Balmin, Miguel Branco, and Anastasia Ailamaki. Same Queries, Different Data: Can we Predict Query Performance? In *Proc. of the International Workshop on Self Managing Database Systems*, 2012.

[16] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., 1st edition, 2010.

[17] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, 2007.

[18] Claude Sammut and Geoffrey I. Webb, editors. *Leave-One-Out Cross-Validation*, pages 600–601. Springer US, 2010.

[19] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: Observing, analyzing, and reducing variance. *Proc. VLDB Endow.*, 3(1-2):460–471, September 2010.

[20] Apache Spark. Monitoring and instrumentation. https://spark.apache.org/docs/latest/monitoring.html , 2017.

[21] Transaction Processing Performance Council TPC. TPC-DS is a Decision Support Benchmark. http://www.tpc.org/tpcds/, 2017.

[22] Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient performance prediction for large-scale advanced analytics. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 363–378, Santa Clara, CA, 2016.

[23] Neeraja J. Yadwadkar, Bharath Hariharan, Joseph E. Gonzalez, Burton Smith, and Randy H. Katz. Selecting the *best* VM across multiple public clouds: a data-driven performance modeling approach. In *Proceedings of the 2017 Symposium on Cloud Computing*, SOCC'17, pages 452–465, 2017.

[24] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, 2010.