Carver: Finding Important Parameters for Storage System Tuning

Authors : Zhen Cao, Geoff Kuenning and Erez Zadok

Background

Storage systems are critical components of modern computing infrastructure, and their performance can significantly impact the overall performance and quality of service of applications and services that rely on them

However, tuning storage systems for optimal performance can be challenging, as there are often many configurable parameters that can affect performance, and it can be time-consuming and resource-intensive to identify the optimal combination of parameter values.

Therefore there is a need for effective and efficient methods for optimizing the performance of storage systems

Introduction

Carver is a tool designed in a way that can select the optimal subset of important parameters needed for tuning the storage systems within a limited number of configurations

It consists of three parts : 1) a variance-based metric to quantify the importance of a storage parameter; 2) a sampling method to intelligently pick a small number of configurations representing the whole parameter space; and 3) a greedy algorithm to select important parameters

1. Measuring Parameter Importance

The parameter importance PI, can be represented as

$$PI(P) = \mathrm{Var}(S) - \sum_{i=1}^{n} \frac{|S_{P=p_i}|}{|S|} \mathrm{Var}(S_{P=p_i})$$

Here S is the original set of configurations, and SP =pi is the subset of configurations with the parameter P taking the value pi

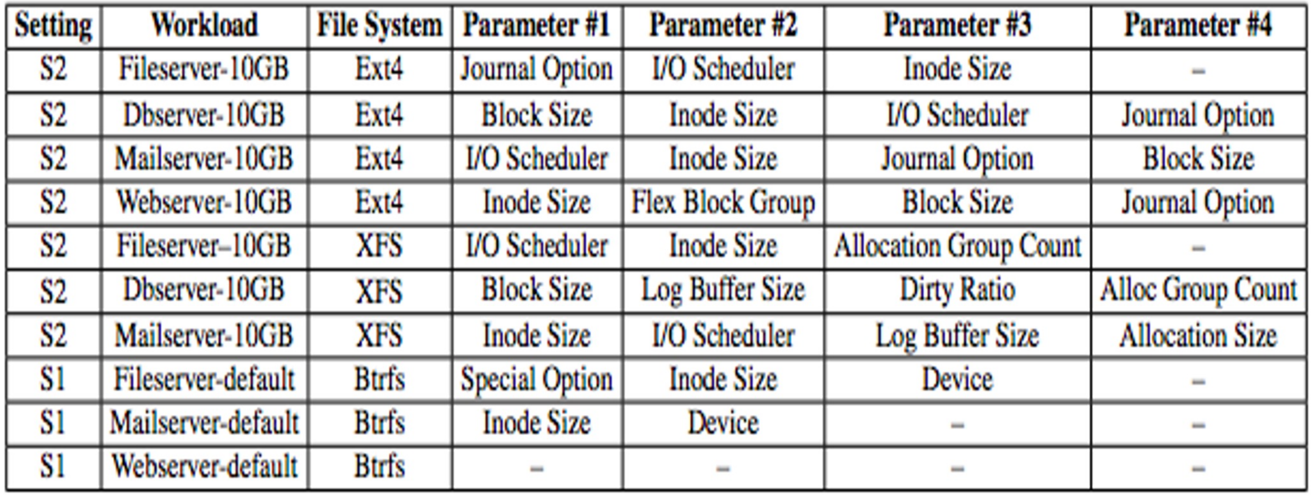
If CPI is the conditional parameter importance, for parameter Q with a total of m possible categorical values {q1, ..., qm}, m > 1, we define the conditional parameter importance for Q, given P = p as

$$CPI(Q|P = p) =$$
$$\mathrm{Var}(S_{P=p}) - \sum_{j=1}^{m} \frac{|S_{Q=q_j,P=p}|}{|S_{P=p}|} \mathrm{Var}(S_{Q=q_j|P=p})$$

where SQ=qj ,P =p denotes the set of configurations with parameters P and Q taking values p and qj , respectively

To remove the restriction to a given value p, we define CPI(Q|P) as the maximum of CPI(Q|P = pi) over all possible values pi ∈ {p1, ..., pn} that parameter P can take

$$CPI(Q|P) = \max_{i=1}^{n} CPI(Q|p = p_i)$$



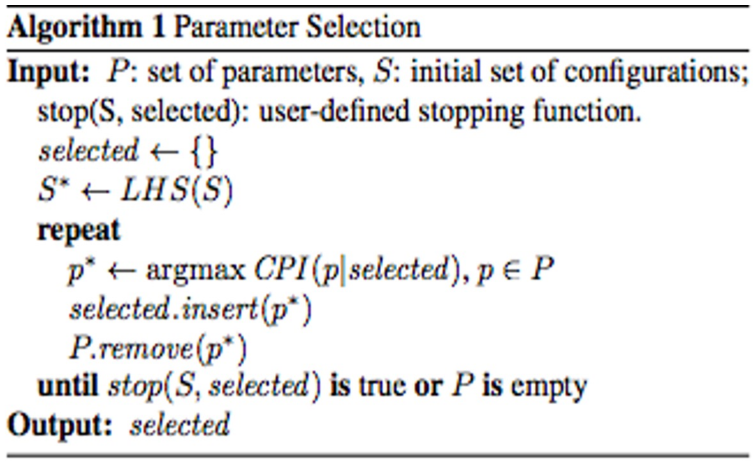
Setting	Workload	File System	Parameter #1	Parameter #2	Parameter #3	Parameter #4
S2	Fileserver-10GB	Ext4	Journal Option	I/O Scheduler	Inode Size	–
S2	Dbserver-10GB	Ext4	Block Size	Inode Size	I/O Scheduler	Journal Option
S2	Mailserver-10GB	Ext4	I/O Scheduler	Inode Size	Journal Option	Block Size
S2	Webserver-10GB	Ext4	Inode Size	Flex Block Group	Block Size	Journal Option
S2	Fileserver–10GB	XFS	I/O Scheduler	Inode Size	Allocation Group Count	–
S2	Dbserver-10GB	XFS	Block Size	Log Buffer Size	Dirty Ratio	Alloc Group Count
S2	Mailserver-10GB	XFS	Inode Size	I/O Scheduler	Log Buffer Size	Allocation Size
S1	Fileserver-default	Btrfs	Special Option	Inode Size	Device	–
S1	Mailserver-default	Btrfs	Inode Size	Device	–	–
S1	Webserver-default	Btrfs	–	–	–	–

2. Sampling

Sampling is required to limit the number of experimental runs/configurations required to select the important parameters

Latin Hypercube Sampling (LHS) is used. In this method, the parameter space is divided into intervals or bins for each parameter, and a random sample is taken from each interval or bin. The resulting samples are arranged in a Latin hypercube pattern, which ensures that each parameter has an equal probability of being sampled and that there are no duplicate samples.

3. Parameter selection algorithm



Algorithm 1 Parameter Selection

```
Input: $P$: set of parameters, $S$: initial set of configurations;
  stop(S, selected): user-defined stopping function.
  $selected \leftarrow \{\}$
  $S^* \leftarrow LHS(S)$
  repeat
    $p^* \leftarrow \text{argmax } CPI(p|selected), p \in P$
    $selected.insert(p^*)$
    $P.remove(p^*)$
  until $stop(S, selected)$ is true or $P$ is empty
Output: $selected$
```

This stopping condition is used as a the Relative Standard Deviation (RSD) or Coefficient of Variation for this experiment

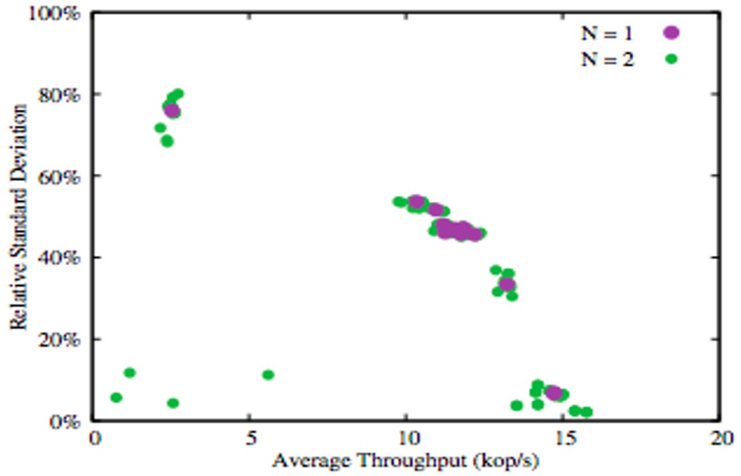
RSD for a set S of configurations can be defined as

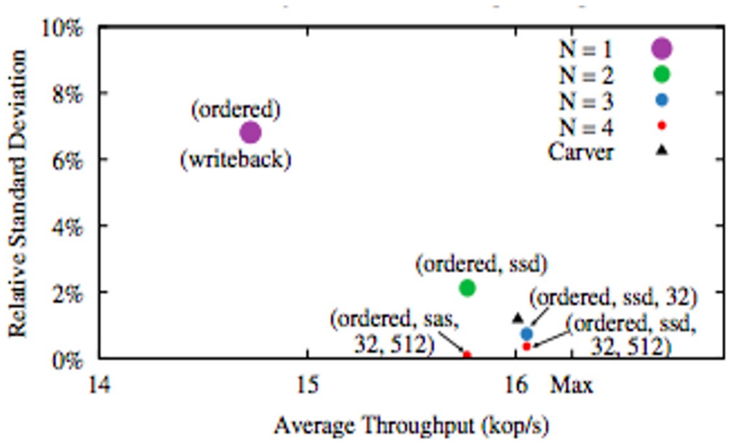
$$\mathrm{RSD}(S) = \frac{1}{\mu}\sqrt{\frac{\mathrm{Var}(S)}{N-1}}$$

where N is the number of configurations and μ is the mean throughput of configurations within S.

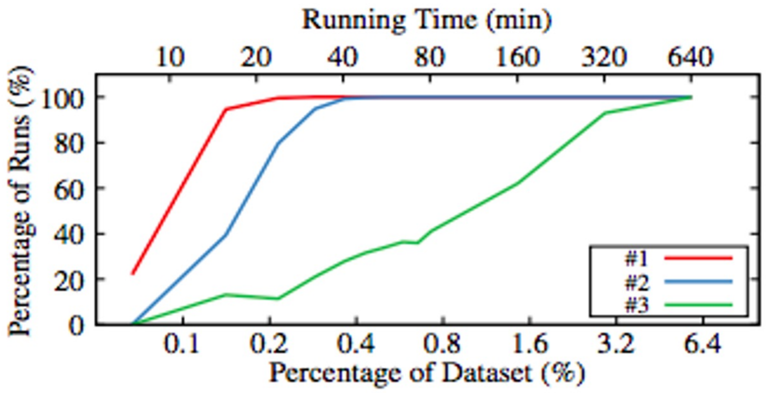
RSD is chosen because it is normalized to the mean throughput and is represented as a percentage

Evaluating the Greedy Algorithm

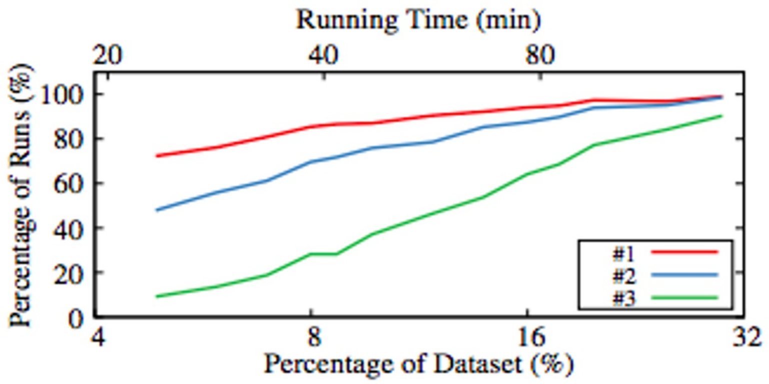




Evaluating Craver



(a) Ext4, Fileserver-default



(b) Btrfs, Fileserver-default

Positive Points

1. Craver is a open source tool which is available freely to use by anyone
2. The authors demonstrate the effectiveness of the Carver tool in tuning a real-world storage system
3. The automatic tuning process of the parameters saves time
4. It uses a novel approach of statistical methods and dimensionality reduction ML approach

Negative Points

1. There is little or no comparison of craver with other similar storage system tuning tools
2. Although craver has been evaluated on real world storage system, it has only used single storage system with different file systems and workloads.
3. Carver can only measure storage importance for one objective at a time (e.g., throughput, latency)

Towards Better Understanding of Black-box Auto-Tuning: A Comparative Analysis for Storage Systems

Authors Zhen Cao, Vasily Tarasov, Sachin Tiwari and Erez Zadok

- This paper addresses the problem of tuning storage systems using black-box auto-tuning techniques
- The paper evaluates different auto-tuning techniques for storage systems: Simulated Annealing (SA), Genetic Algorithms (GA), Bayesian Optimization (BO), and Deep Q-Networks (DQN) and also Random search (RS) and provides a detailed comparative analysis of all these methods based on different metrics.
- Both these papers are address the problem of improving the performance of storage systems by automating them but use different techniques
- While "Carver" identifies important parameters for tuning, this paper provides a broader evaluation of auto-tuning techniques. The results of this paper can inform the implementation of auto-tuning techniques in a broader aspect, while "Carver" is a specific tool that can be used to identify critical parameters to tune.

Adaptive Control and Grid Search

Adaptive control method uses a feedback control loop to adjust the system parameters based on the measured system performance. It starts with a set of default parameters and adjusts them based on feedback from the system. It continuously monitors the system and adjusts the parameters to maintain optimal performance.

Grid search method performs an exhaustive search of the parameter space by trying every possible combination of parameter values within a predefined range. It evaluates the performance of the system for each combination and selects the combination that gives the best performance.

Random Search and Optimization methods

Random search method randomly selects parameter values from within predefined ranges and evaluates the performance of the system for each set of values. It continues this process for a specified number of iterations and selects the set of values that gives the best performance.

Model based optimization method builds a model of the system performance based on a set of training data. The model is then used to predict the performance of the system for a given set of parameters. It uses an optimization algorithm to find the set of parameters that maximizes the predicted performance.

Results and Comparison

The authors used two different metrics to evaluate the performance of each auto-tuning approach: throughput and latency

The authors evaluated the performance of each auto-tuning approach on each storage system under different workloads. They found that Ad-hoc Auto-tuning consistently outperformed the other approaches in terms of throughput and latency. They also found that Bayesian optimization performed well in some cases but was less effective when the system was under heavy load.

Overall, the comparison analysis provides insights into the strengths and weaknesses of each auto-tuning method and their suitability for different types of storage systems and workloads.

Questions

- What other method can be used in place of variance based metric?   Bayesian optimization or random search can be used
- What else can be used instead of LHS? Grid sampling or random sampling can be used
- Can unsupervised learning be used for parameter selection? Yes but supervised learning methods would be more effective in this case as the authors have already identified the set of parameters that are potentially important, and their goal is to determine the relative importance of each parameter.
- Why LHS ? Because of better coverage of the parameter space
- Limitations of craver ?
- Can Carver be used for any type of storage system? May or may not.

- What metrics are used to evaluate performance of craver? Throughput, latency
- Security or privacy issues? No not any because it does not access any private data from the storage systems
- Can craver be used on other OS other than linux? No because the storage systems discussed are linux based
- How can the carver algorithm can be improved? Incorporating more advanced ML techniques, extending to other storage systems
- What is the assumption for Latin Hypercube Sampling (LHS)? the distribution of one parameter does not affect the distribution of another parameter
- How can you make sure that the LHS-picked configurations can be representative for the whole set of configurations?
- How does Carver handle the case where multiple parameters have similar importance scores?

References

- Cao, Z., Kuenning, G., & Zadok, E. (2020, February). Carver: Finding Important Parameters for Storage System Tuning. In *FAST* (pp. 43-57).

- Cao, Z., Tarasov, V., Tiwari, S., & Zadok, E. (2018). Towards better understanding of black-box auto-tuning: A comparative analysis for storage systems. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)* (pp. 893-907).