

# Understanding Resource Usage of NoSQL Databases Through System call Trace

Changho Seo, Yunchang Chae, Jaeryun Lee, Euiseong Seo\*, and Byungchul Tak

Kyungpook National University, \*Sungkyunkwan University  
[changho.seo, cyc, jrlee, bctak]@knu.ac.kr, \*euiseong@skku.ac.kr

## Abstract

NoSQL databases have quickly become an indispensable component for big data processing and AI applications. However, it is very challenging to select the best NoSQL database that fits the performance and scalability requirements of the big data and AI application at hand. To address these challenges, we investigate the feasibility of the approach that tries to build accurate resource consumption models of NoSQL databases from low-level system call information. Accurate resource consumption models would enable us to reason about the expected performance and the scalability capacities for given workloads and hardware specifications. Our study reveals that NoSQL databases exhibit unique resource consumption behaviors even for functionally the same operations. Also, we verify that it is feasible to build accurate resource consumption models with only a small number of experiments.

## I. Introduction

NoSQL databases are established as the essential component for big data and AI applications. Selecting the right NoSQL database as the backend storage for the big data and AI is critical for the performance and efficiency. However, it is challenging to select the most fitting one for the type of applications to be run. Static information, such as software features or capabilities from various document sources can provide only the general information. Key information about the actual performance characteristics and scalability under the expected workloads are not easily derivable from such documentations.

Performance benchmarking is one way to build performance models and infer the scalability [1], [2]. Unfortunately, these black-box methods have several shortcomings. Benchmarking approaches can cover only a fraction of vast search space made of workload types, configurations, and environment preferences. Furthermore, it is difficult to explain the observed performance outcomes and reason for various scenarios, such as different workload intensity and mix.

In this work, we address these challenges by investigating the gray box techniques for building accurate resource consumption models of NoSQL databases. We use system call activities directly related to the resource consumption such as read, write, send, recv and futex system calls. These system calls decorated with CPU cycles, memory bandwidth, and I/O byte accounting information from the basis of our resource model.

In our grey-box technique, system call-level information is used because they offer several advantages. First, developing a technique does not require us to have application-specific knowledge such as internal function descriptions or software architecture. Second, system calls directly provide I/O resource accounting data through return values. Third, source code is not required since we do not intend to instrument the code. Lastly, the amount of traced data we need to process is significantly less than those of function-level or instruction-level traces.

We were able to develop a set of novel algorithms to handle all

challenges mentioned above and built the desired resource consumption models of three key operations (insert, select, delete) for Redis, Memcached and MongoDB NoSQL databases.

## II. Design

We describe the detailed architectural design of our system for resource consumption modeling in Figure 1. We divide the functionality into three nodes - Experiment, Analysis, and Control node.



Figure 1. System Architecture

Our target events for tracing are the entry and exit point of all system calls. We capture the metrics at the entry and exit of a system call invocation, and calculates the elapsed time. This elapsed time indicates the number of CPU cycles used during the system call execution. Similarly, the difference between the exit of one system call and the entry of the next system call gives us the CPU cycles consumed by the application (i.e. NoSQL database of choice) during the user mode. The memory bandwidth usage can be obtained similarly. The network and disk I/O bandwidth usage is obtained from the return values of these system calls.

There are a few known tools available for tracing the system calls from target applications and gather resource usage information.

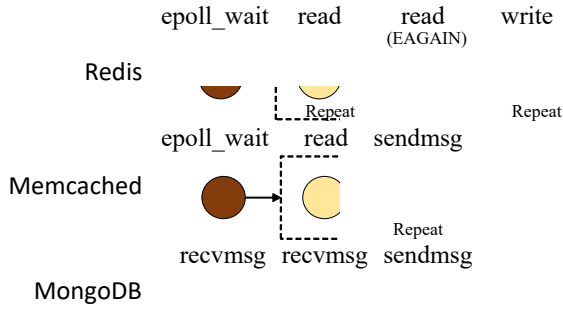


Figure 2. System call sequence model of Redis, Memcached, and MongoDB

Figure 3. Resource consumption result of Insert operation

Figure 4. Resource consumption result of Select operation

Figure 5. Resource consumption result of Delete operation

SystemTap, one of those, is a tool that traces the Linux kernel activities without recompiling the kernel. It allows users can specify various actions for events and even add custom helper functions. In the current implementation, SystemTap is selected as a tool for system call tracing and resource usage accounting. We have added the CPU and memory accounting logic into the SystemTap module.

### III. Evaluation

We chose three NoSQL databases (Redis, Memcached and MongoDB) as the targets of resource model construction. Depending on our key objectives of achieving the explainability, we present here the modeling results of three NoSQL databases produced by our

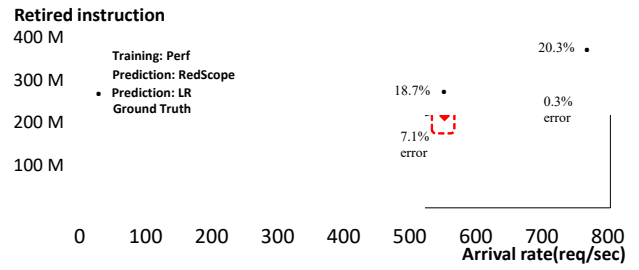


Figure 6. Accuracy of our technique and the LR (Linear Regression) model on MongoDB insert operation

framework and provide descriptions of internal behaviors. Figure 2 shows the system call sequences we discovered for three NoSQL databases. We have applied three operations (insert, delete, and select) to all three. However, the system call patterns were identical for three operations within the same database. Figure 3, 4, and 5 describes the resource consumption amount for unhalted cycle and retired instruction of each three operations constructed by our technique.

Figure 6 shows the instruction consumption according to each operation predicted by ours and their actual observations. The experiment was designed to predict the resource usage used by the target NoSQL database when a series of requests are issued by varying the arrival rate from the client. For comparison, we set up a simple linear regression model based on the values collected from the Perf tool. The retired instruction model for MongoDB insert operation was evaluated. In the graph, dotted lines show trends obtained from linear regression model. In MongoDB, as the request frequency increases, the inclination of resource usage becomes less stiff, unlike the linear regression result.

### IV. Conclusion

In this work, we have demonstrated the feasibility of the grey-box approach based on the system call traces to building the resource consumption models of NoSQL databases. Using the proposed framework, we investigated three NoSQL databases to gain new insights and analyzed their scalability potentials. Our findings revealed that NoSQL databases all have unique and complex internal operations that affect their overall scalability.

### REFERENCES

- [1] P. Malakar, P. Balaprakash, V. Vishwanath, V. Morozov, and K. Kumaran. Benchmarking machine learning methods for performance modeling of scientific applications. In 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), pages 33–44, 2018.
- [2] C. Witt, M. Bux, W. Gusew, and U. Leser. Predictive performance modeling for distributed batch processing using black box monitoring and machine learning. Information Systems, 82:33–52, 2019.