Effective Database Design for Efficient Workflow Orchestration

Author: Kabir Vats | UC San Diego Jacobs School of Engineering Department of Electrical and Computer Engineering
Mentors: Rusty Davis | Los Alamos National Lab, HPC-DES and Andres Quan | Los Alamos National Lab, CCS-7

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Department of Electrical and Computer Engineering

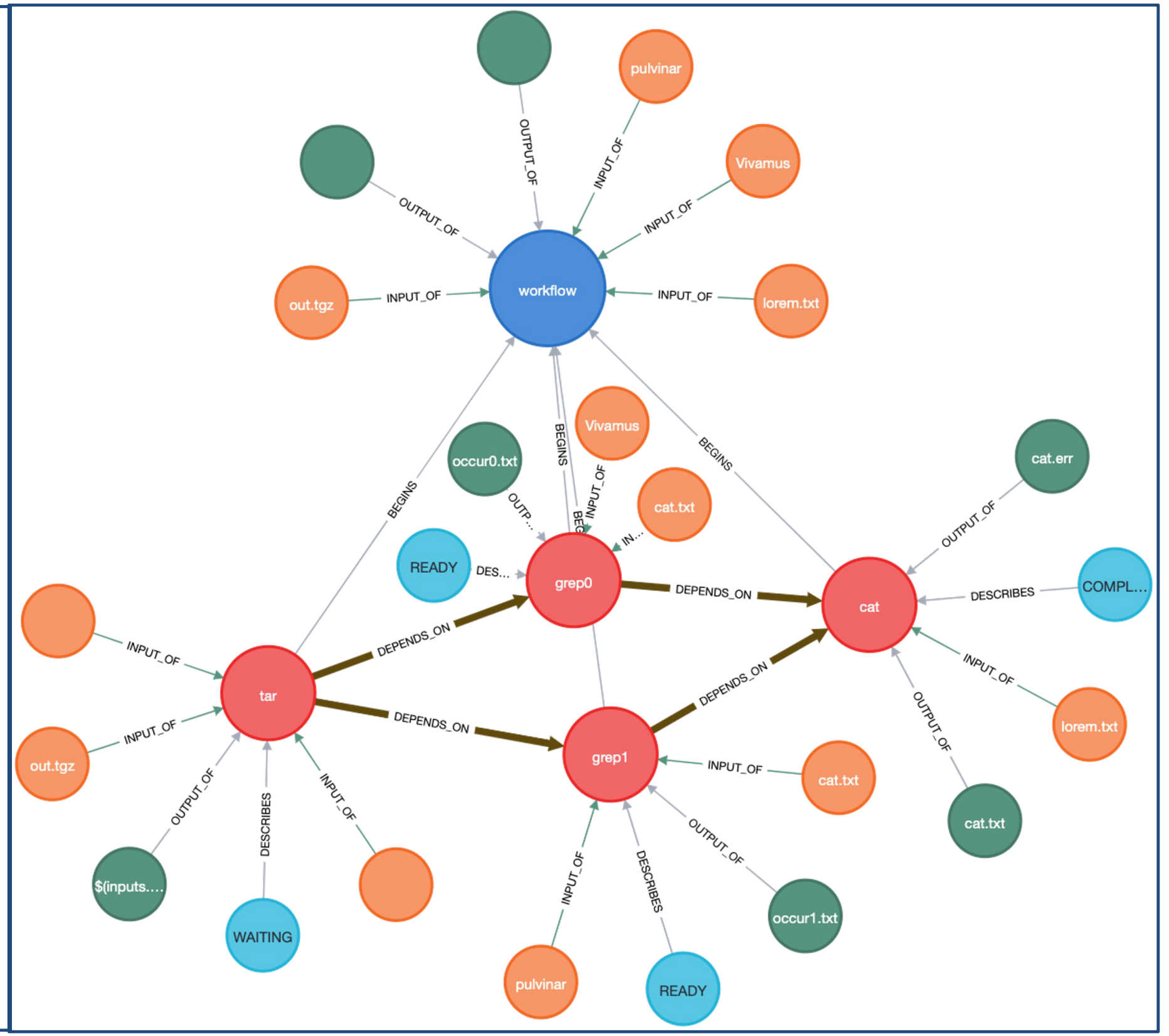
BEE Build and Execution Environment

Introduction

INTRO TO BUILD AND EXECUTION ENVIRONMENT (BEE)

“BEE is a workflow orchestration system designed to build containerized HPC applications and orchestrate workflows across HPC and cloud systems” (BEE). To orchestrate multi-step workflows, the workflow’s steps are interpolated into dependencies between tasks. BEE uses a Neo4j Graph Database to track satisfaction of dependencies.

Figure 1. Graph Database during the “Cat Grep Tar” Workflow. The various nodes store information for each of the tasks (highlighted in red) to use during execution. The dependencies between tasks are also boldened, and the blue ‘Metadata’ nodes track a task’s state. By using a graph database to track workflow state, BEE knows when it can execute tasks after a dependency is satisfied.



CHANGE TO THE DATABASE DESIGN

Prior to this project, BEE would launch an instance of Neo4j for every submitted workflow. At scale, multiple workflows’ Neo4j instances running on the same BEE client would be using significantly more system resources which would add to BEE’s overhead in workflow orchestration. The solution needed to use Neo4j database to incorporate all workflows associated with a BEE instance. The big-picture database change is the insertion of a ‘Head’ node that connects to each workflow node.

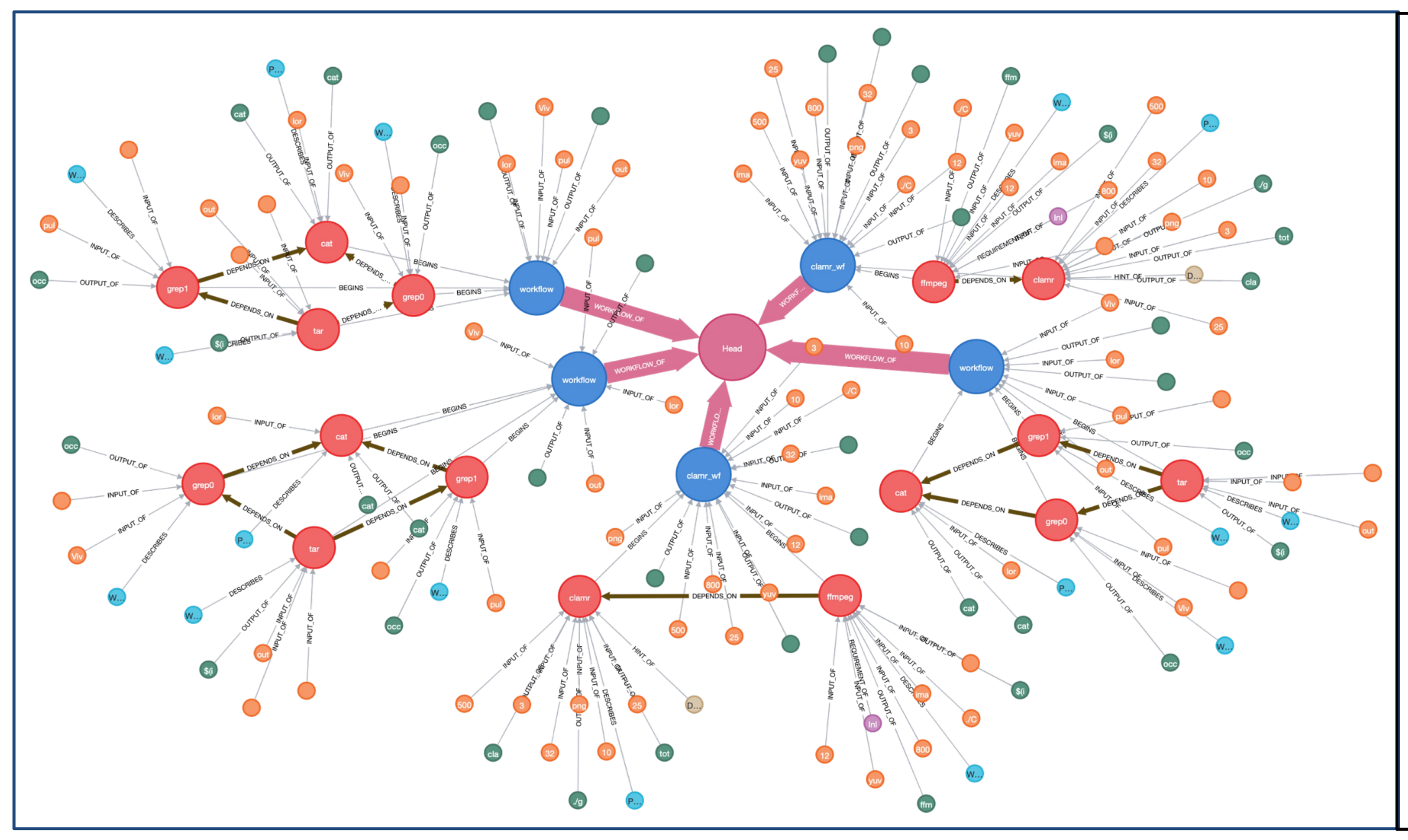


Figure 2. New design of the graph database during the execution of three “Cat Grep Tar” workflows and two “CLAMR” workflows (See Performance Study). The highlighted pink nodes and edges show the added ‘Head’ node that connects to each workflow.

To execute this change, the Neo4j Cypher queries were changed to ensure that each query would only impact one workflow, eliminating the chance of workflows interfering with each other. Many launches and connections to the graph database were also refactored to ensure that workflows did not interfere with each other.

Efficiency Study

METHODS

To test the effectiveness of running one database as opposed to launching multiple, a performance study was performed on LANL’s Darwin Cluster’s Skylake-Platinum partition. The Neo4j process(es)’ memory usage (RSS) was recorded over time. First, the sample workflows “Cat Grep Tar” and “CLAMR” were run in isolation. Then, two instances of “CLAMR and three instances of “Cat-Grep-Tar” were run simultaneously (Five concurrent workflows total).

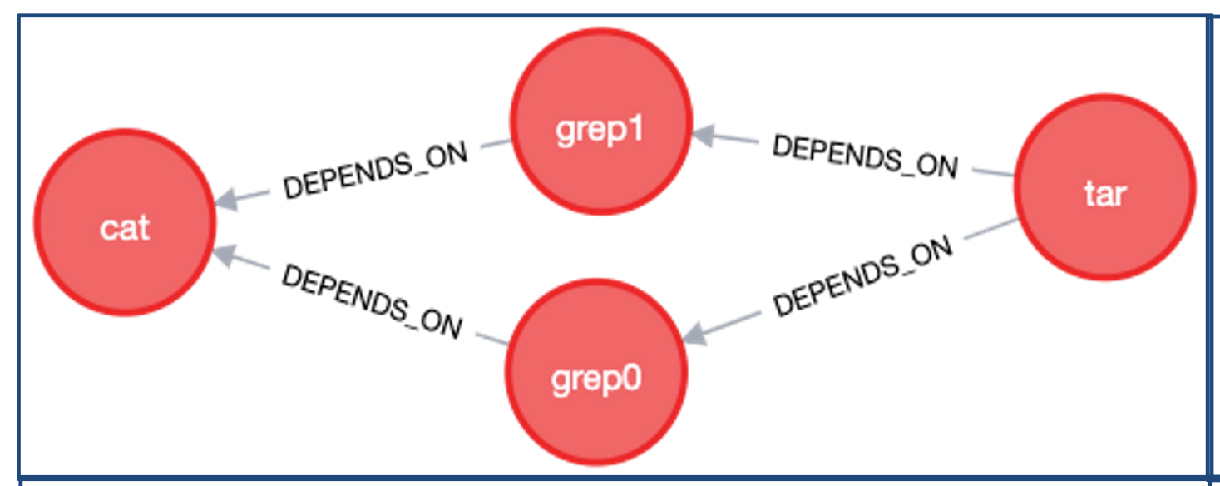


Figure 3. “Cat Grep Tar” workflow. This workflow has simpler tasks with a more complex dependency structure

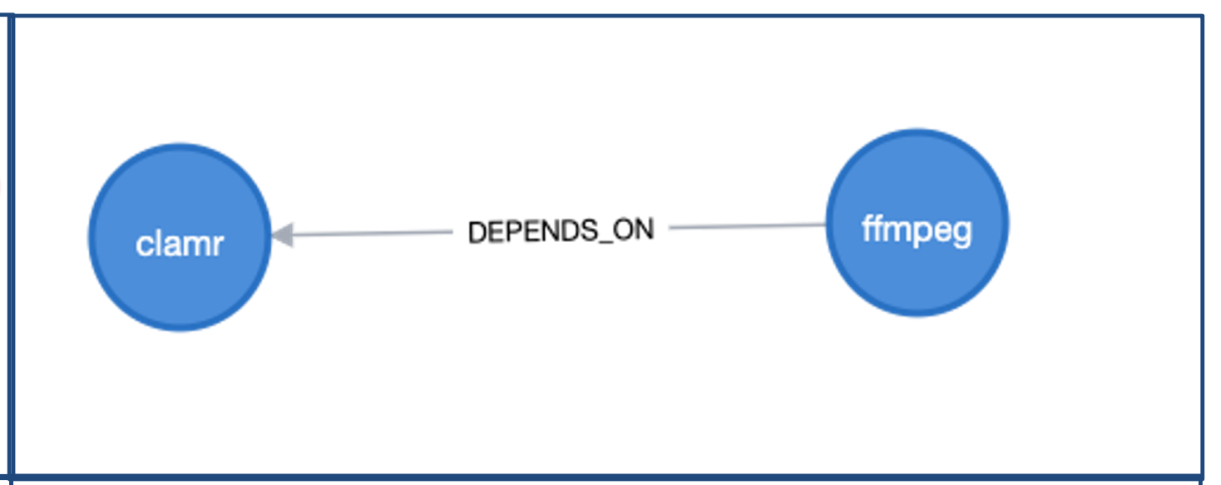


Figure 4. “CLAMR” workflow. This workflow has more complex tasks but has a simple dependency structure

RESULTS

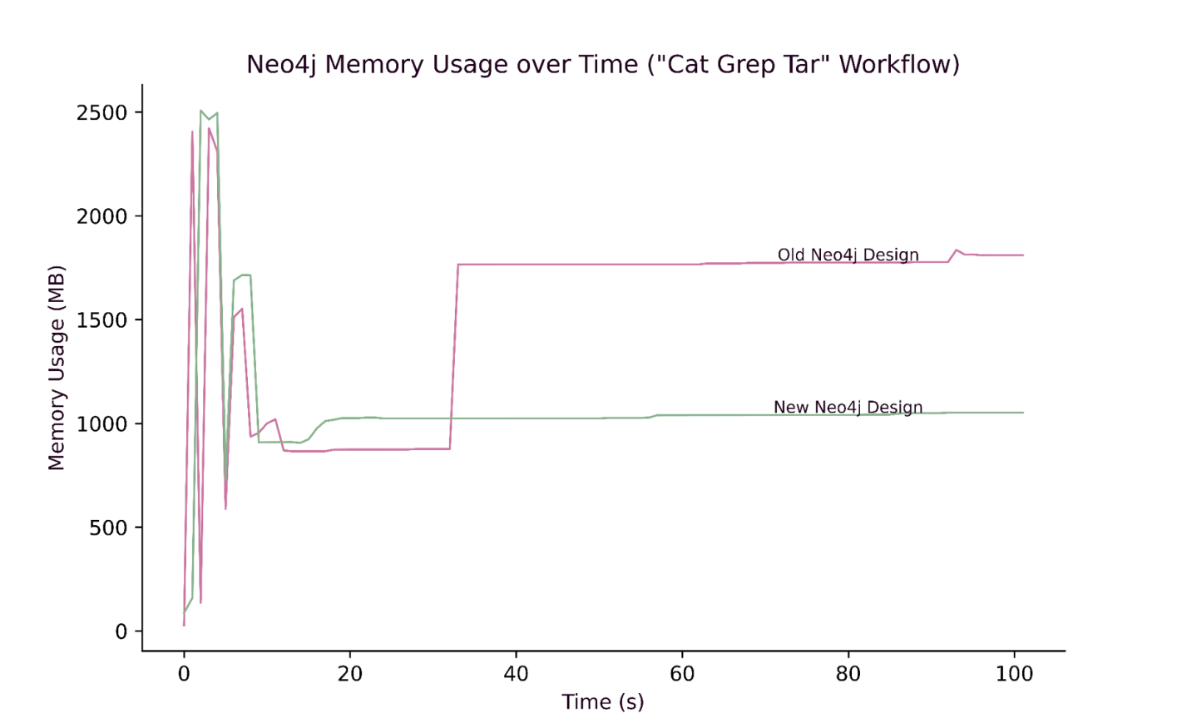


Figure 5. Comparison of memory usage over time from Neo4j during the “Cat Grep Tar” workflow

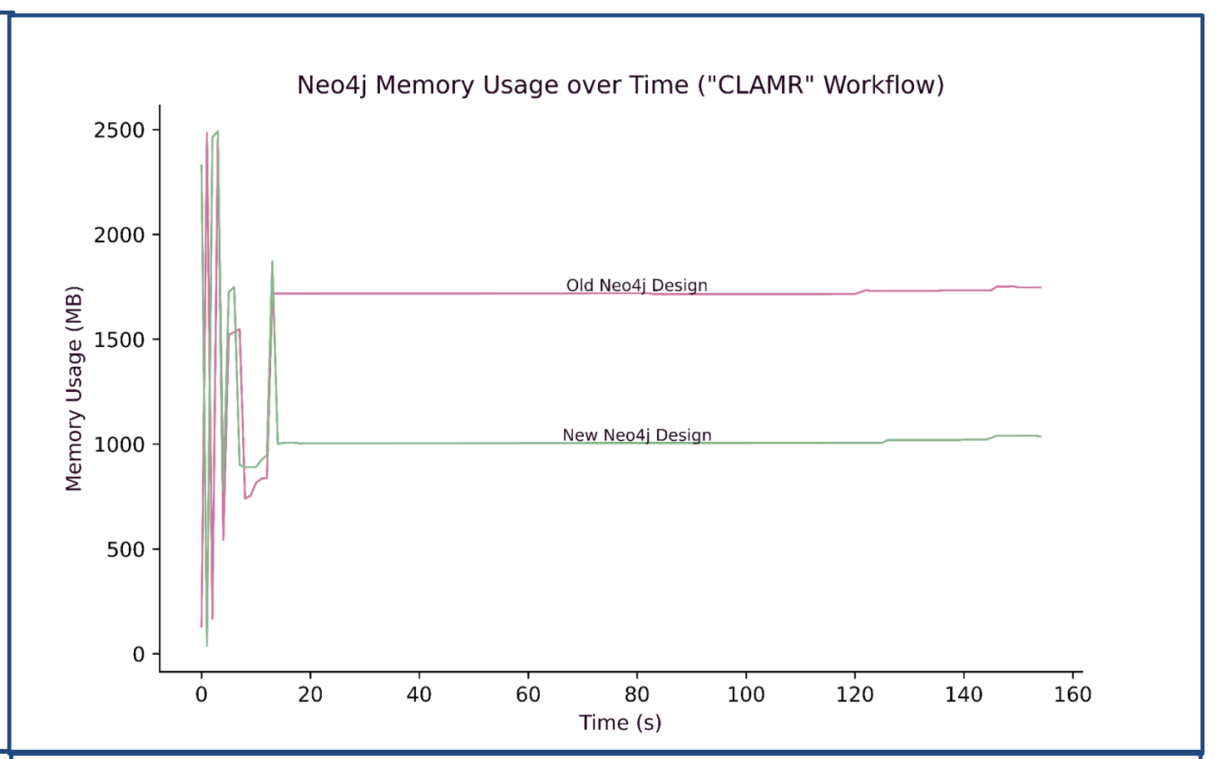


Figure 6. Comparison of memory usage over time from Neo4j during the “CLAMR” workflow

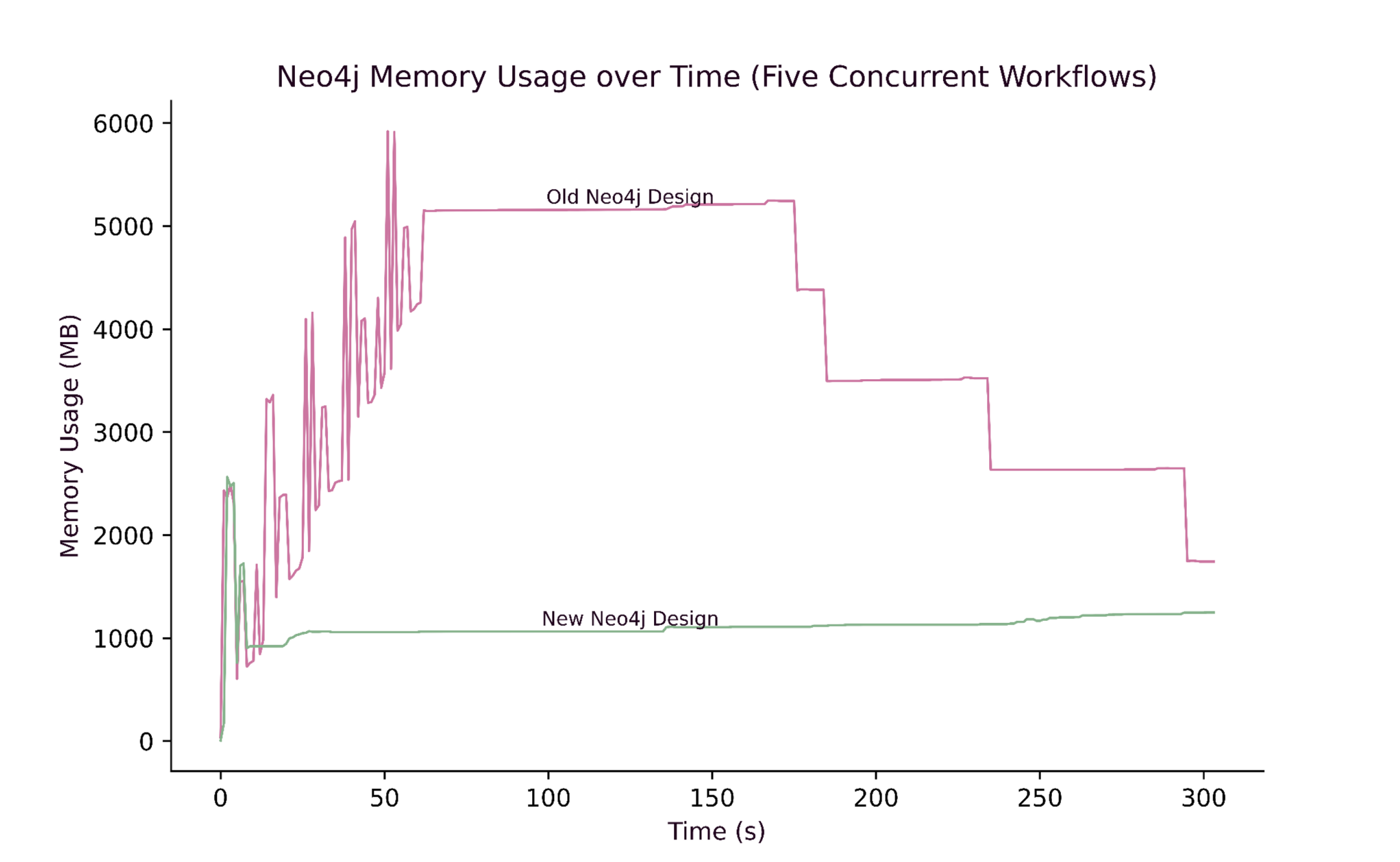


Figure 7. Comparison of memory usage over time from Neo4j for two instances of the “CLAMR” workflow and three instances of the “Cat-Grep-Tar” workflow.

Memory usage was higher during all experiments (“Cat Grep Tar”, “CLAMR”, and five concurrent workflows) for the old database design. The difference in memory is especially apparent for the concurrent workflows run, which shows multiple step-up increments of the memory usage after the creation of each instance of Neo4j database. At its max usage, the old Neo4j design was using approximately 5000 MB of RAM to the new design’s 1000 MB, revealing the memory-efficiency of restricting the Neo4j database to a single instance.

Results Analysis

CONCLUSION

The results of the experiment signify that the change from using one Neo4j database per workflow to using one Neo4j database regardless of the number of workflows resulted in significantly more efficient memory usage from Neo4j. The memory efficiency increase for the new design is particularly pronounced during concurrent execution of multiple workflows, appearing to consume five times less memory during the execution of five workflows. From these results, it appears likely that this change will greatly improve BEE’s efficiency, especially during large scale deployments.

Another conclusion that can be drawn from this study is that the execution of BEE workflows while using a singular database did not result in any debilitating database query errors, meaning BEE’s functionality as a workflow orchestration software was not reduced at all by the changes made.

DISCUSSION

The likely reason the singular instance consumed less memory during even the “CLAMR” and “Cat-Grep-Tar” runs in isolation is likely due to a change in the Neo4j boot method, where the new solution boots Neo4j “Console” which runs in the foreground rather than ‘Start” which runs in the background in order to access the Neo4j status as a subprocess. However, this distinction is not enough to explain the discrepancy seen during the concurrent workflow run, where the new design shows significantly lower RAM usage. The asymptotic RAM usage with respect to workflows is linear for both implementations O(n), but the new design shows a significantly smaller increase in Neo4j’s RAM usage per workflow, making it far more desirable to be run on HPC systems.

Further development of this new design aims to incorporate the ability to archive the state of a particular workflow within the Neo4j database. If HPC environments experience outages during a multi-day workflow run, it is important for BEE to be able to resume from its last state, including its most recent Graph Database state. These changes will likely incorporate a query into the database for the workflow’s state, which will be saved within the workflow’s working directory.

Future studies on the efficiency of this new design should analyze the differences between the runtime of both designs during the execution of concurrent workflows. Neo4j’s driver operates on a single thread to avoid race conditions, so many read and write operations could lead to an increase in runtime for a workflow.

Los Alamos NATIONAL LABORATORY

NNSA National Nuclear Security Administration

Managed by Triad National Security, LLC, for the U.S. Department of Energy’s NNSA.

LA-UR-24-28101

References

“BEE: Build and Execution Environment.” *GitHub*, Los Alamos National Laboratory, github.com/lanl/BEE. Accessed 1 Aug. 2024.