Reproducibility Report for ACM SIGMOD 2022 Paper: "Hybrid Deterministic and Nondeterministic Execution of Transactions in Actor Systems"

AUNN RAZA, EPFL, Switzerland

This report tries to reproduce the Snapper system on an AWS instance. We can reproduce most figures and major findings in the paper. All experiments succeed and produce numbers supporting the authors' claim; however, scalability experiments with the TPC-C benchmark at 32 cores (all available resources) are slightly slower than the reported numbers but follow the same pattern as the paper and support the author's claims.

1 INTRODUCTION

This report tries to reproduce the result of the Snapper system [1], a transactional library on top of the Orleans system, which is an actor-oriented system. Snapper-Orleans implements a transaction library in an actor-based system, enabling multi-actor transactions through 1) deterministic schedules and 2) S2PL CC protocol.

Submitted experiments support the central results and claims of the paper, and reproduced results depict the same pattern and conclusion as submitted in the original paper. The reproducibility scripts are easy to use and well-documented. The only experiment slightly deviating from reported numbers in the paper is the macro-benchmark with TPC-C workload (Fig. 17B of paper) when executed on all available 32-cores. The reason reported by the author is resource contention in a 32-core setup and is alleviated if the 32-core experiment is executed on a 36-core machine.

2 SUBMISSION

The submission includes the source course of Snapper-Orleans through Github with a detailed readme explaining the deployment environment and setup instructions, as well as scripts for running all experiments and generating the graphical plots.

A list with a summary of the submission contents is also useful. For example:

- GitHub repository with code and scripts at: https://github.com/diku-dk/Snapper-Orleans
- A detailed readme file at https://github.com/diku-dk/Snapper-Orleans/blob/main/README.md
- Data generation code at https://github.com/diku-dk/Snapper-Orleans/tree/main/SmallBank.DataGenerator

3 HARDWARE AND SOFTWARE ENVIRONMENT

Table 1 describes the resources used in the original paper and in our reproducibility setup.

The original paper uses AWS EC2 c5n instances for all experiments except for the scalability experiments, which use the c5n.9xlarge. For reproducibility setup, we employ the AWS EC2 c5n.9xlarge instance for all experiments to avoid re-setting the environment across experiments. The AWS EC2 c5n instance has a 4-core processor with 10.5GB memory.

4 REPRODUCIBILITY EVALUATION

4.1 Process

The experiments are reproduced on the two datasets (SmallBank and TPCC), which are generated through the accompanied scripts in the submission. The server/client scripts run all the experiments one after another, reproducing each reported figure in sequence. The scripts also parse reported numbers and generate Matlab scripts to generate plot figures shown in the paper. It was possible to

Table 1. Hardware & Software environment

	Server	Client
Cloud Provider	AWS	
AWS EC2 Instance Type	c5n.9xlarge	
CPU	3.0 GHz Intel Xeon Platinum Processor	
cores	36	
RAM	96GB	
Network	100Gbps	
Storage	128GB io2 SSD (64K IOPS)	40GB gp2 SSD
AMI	Microsoft Windows Server 2019 Base	
Platform	.NET Core SDK 3.1.301	
Cluster Membership Table	AWS DynamoDB	

follow the reproducibility instructions without the author's help and run all the experiments in one go. There was one instance where the result parsing code could not adapt to the different text encoding in the reproducibility setup, which the authors fixed on request. Further, the authors were quite responsive in discussing and explaining the deviation for one of the scalability experiments.

4.2 Results

The following figures have been reproduced: Figure 1, Figure 2, Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, and Figure 9. The obtained numbers and plots appear to be close enough to the paper's reported values, following the same pattern and claims.

However, the scalability experiment with the TPC-C benchmark (figures 8 and 9) shows deviation from reported numbers at 32-cores. This deviation is attributed to resource contention when all available resources are utilized at 32 cores.

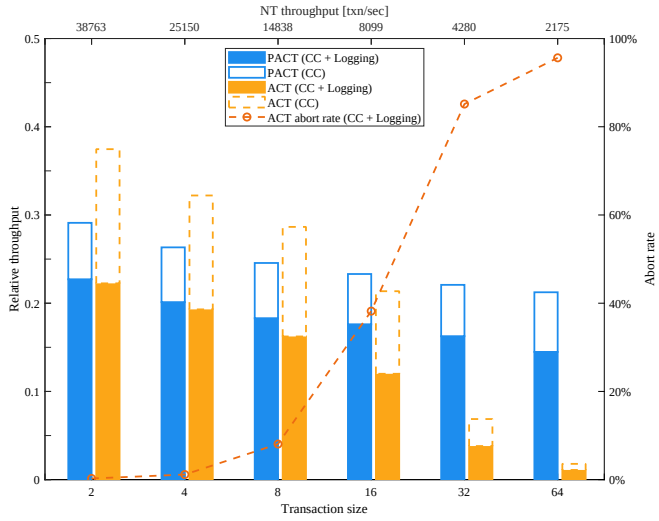

Fig. 1. Transaction overhead (corresponding to original paper Fig 12)

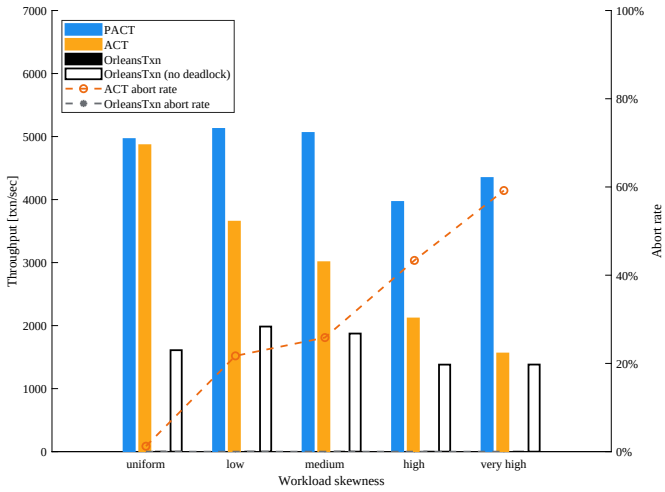

Fig. 2. Throughput (corresponding to original paper Fig 14)

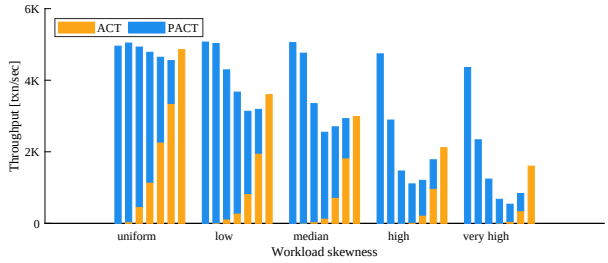

Fig. 3. Hybrid execution: Throughput (corresponding to original paper Fig 16a)

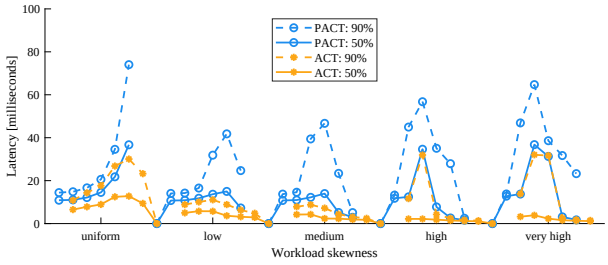

Fig. 4. Hybrid execution: Latency (corresponding to original paper Fig 16b)

5 SUMMARY

The major figures and experimental conclusions in the paper have been reproduced, supporting the patterns reported in the paper.

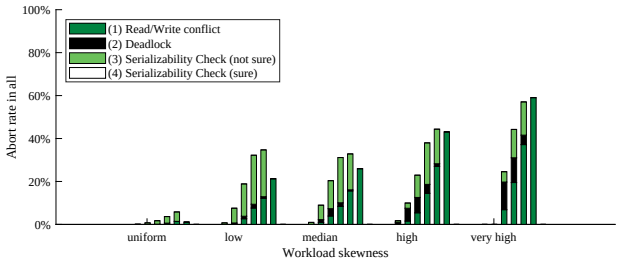

Fig. 5. Hybrid execution: Abort rate (corresponding to original paper Fig 16c)

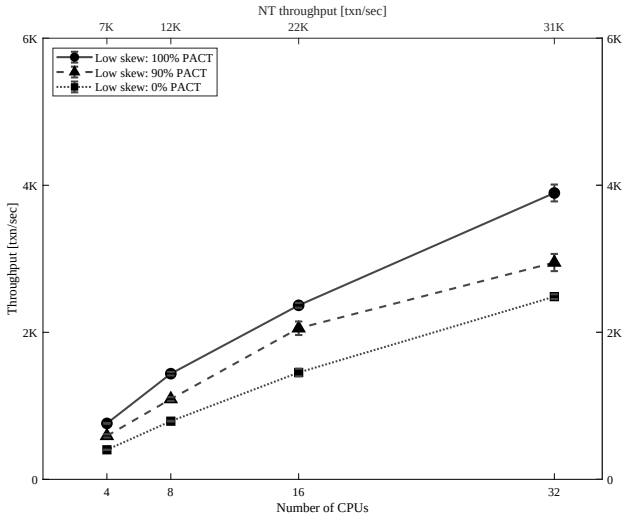

Fig. 6. Scalability: Smallbank-uniform (corresponding to original paper Fig 17a(left))

REFERENCES

[1] Yijian Liu, Li Su, Vivek Shah, Yongluan Zhou, and Marcos Antonio Vaz Salles. 2022. Hybrid Deterministic and Nondeterministic Execution of Transactions in Actor Systems. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 65–78. https://doi.org/10.1145/3514221.3526172

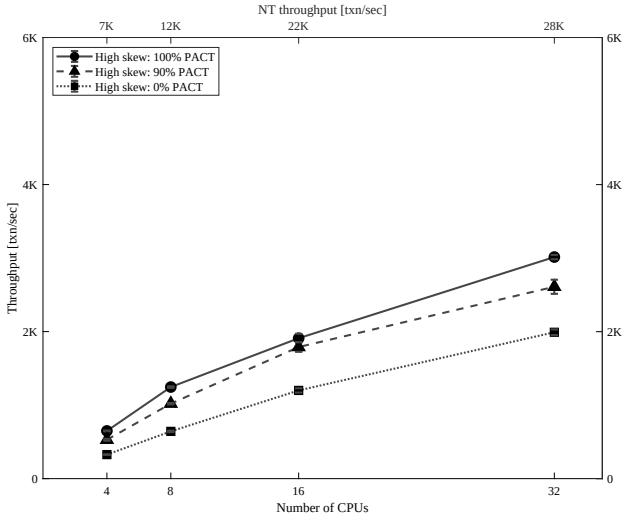

Fig. 7. Scalability: Smallbank-skewed (corresponding to original paper Fig 17a(right))

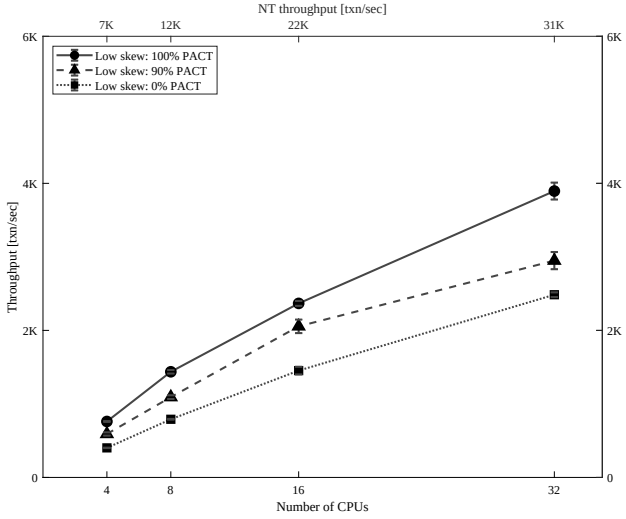

Fig. 8. Scalability: TPCC-uniform (corresponding to original paper Fig 17b(left))

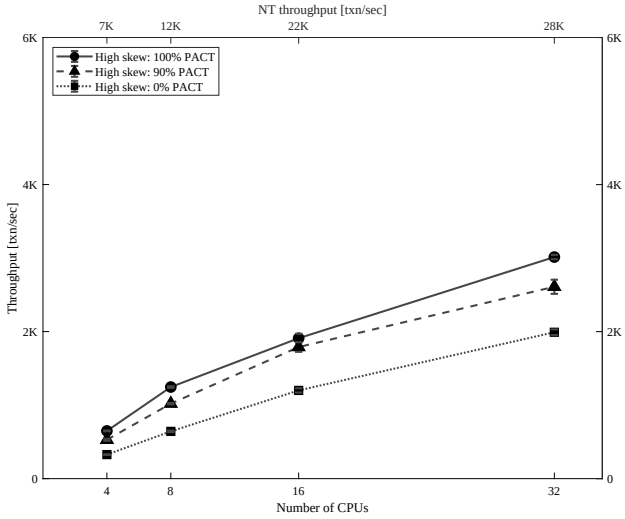

Fig. 9. Scalability: TPCC-skewed (corresponding to original paper Fig 17b(right))