Evaluation of geo-distributed databases using the DeathStar Hotel Reservation benchmark

Author: Aidan Eickhoff
Supervisors: Oto Mráz, Asterios Katsifodimos

1. Introduction

Geo-distributed database: A large database composed of multiple databases spread across large physical distances.
Database benchmark: A method of comparing database performance using a simulated workload.
DeathStar[1] Hotel Reservation: A benchmark which aims to represent a modern travel booking application.

2. Research Question

Research Question: How do geo-distributed database systems perform when benchmarked using the DeathStarBench hotel reservation scenario?

This evaluation will be done under several **scenarios**, namely:

- Baseline
- Skewed
- Sunflower
- Network delays
- Packet loss
- Scalability

Performance will be measured using the following metrics:

- Throughput
- Latency
- Bytes transferred
- Cost

2. Background

Database benchmarking is important to determine the strengths and weaknesses of a system before deciding on it's use.

Traditional benchmarks may no longer be representative of modern workloads, creating a mismatch between expected and actual performance, especially for geo-distributed databases.

This project and the others in the group aim to implement several benchmarks in an appropriate environment for a comparison of some modern database benchmarks.

Which geo-distributed databases will be measured:

- Detock[2]
- Calvin[3]
- SLOG[4]
- Janus[5]

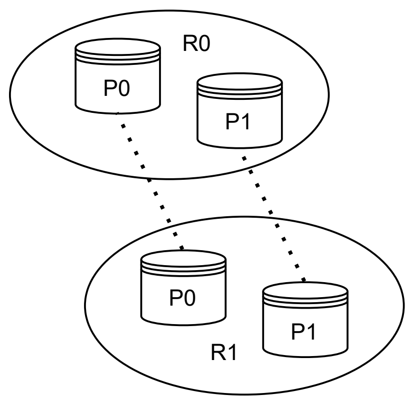
4. Experimental Setup

users	
PK	username
	password

hotels	
PK	hotel_id
	latitude
	longitude
	rating
	price
	capacity

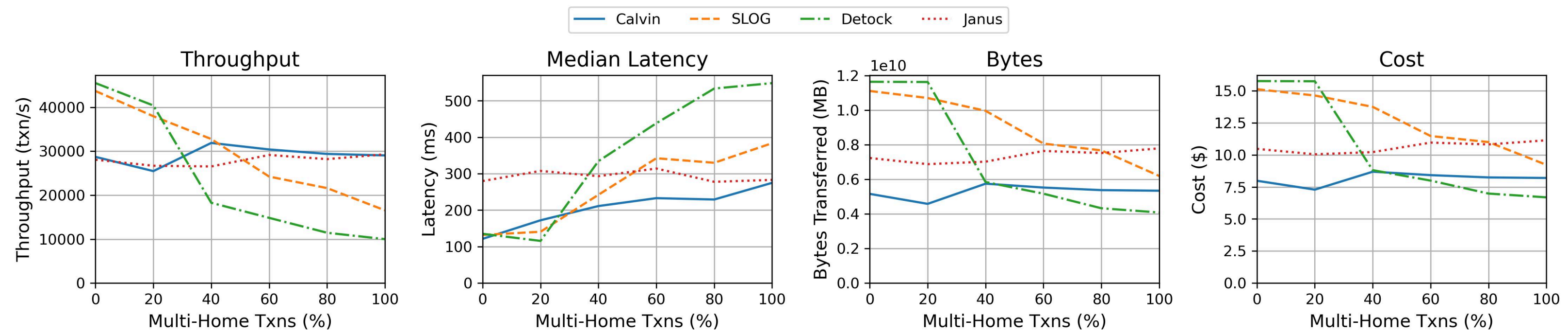
reservations	
PK	reservation_id
PK + FK	hotel_id
	customer_name
	in_date
	out_date
	num_rooms

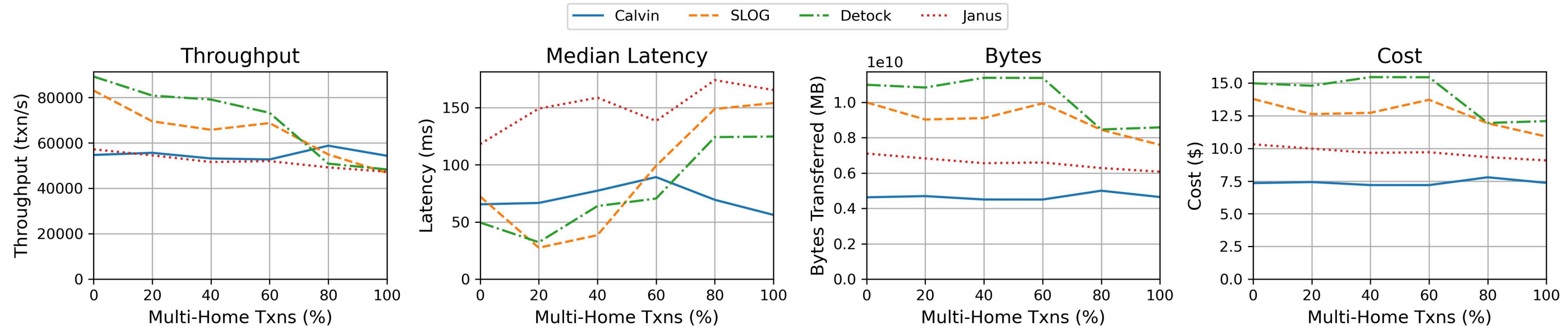
reservation_cnt	
PK	in_date
PK + FK	hotel_id
	reserved_rooms



5. Results

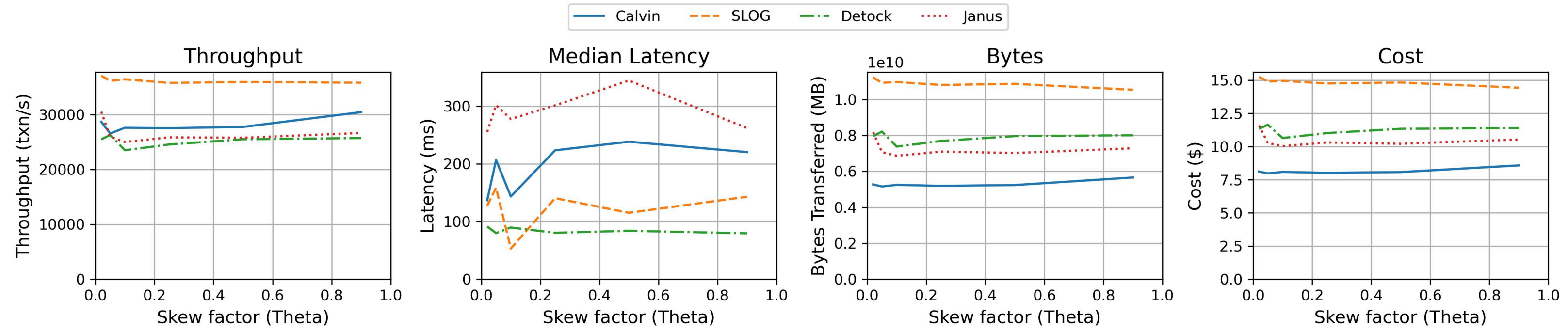
The baseline scenario with large transactions. Calvin and Janus stay constant, and are significantly more cost efficient for high values

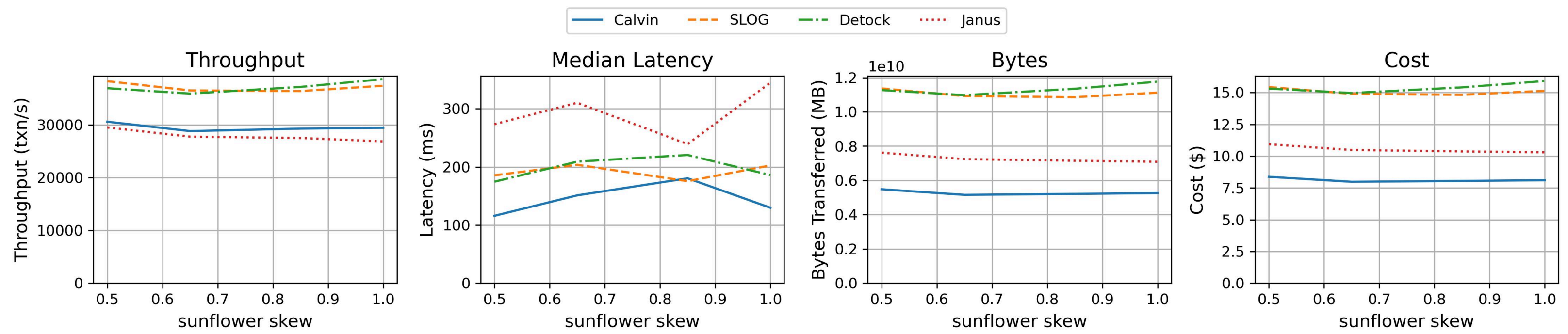




The baseline scenario with small transactions. Detock and SLOG keep an advantage for longer.

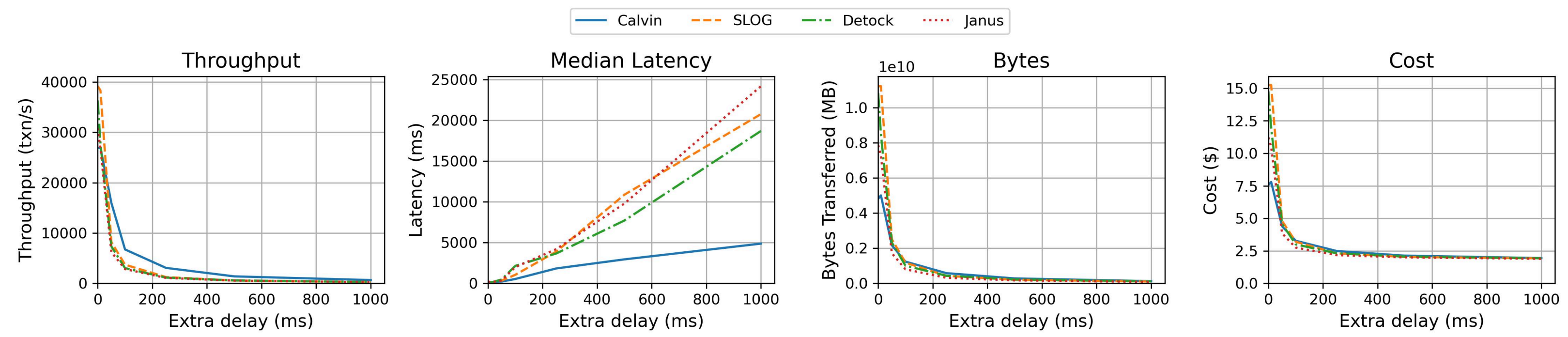
The skew scenario. Performance is not significantly affected.

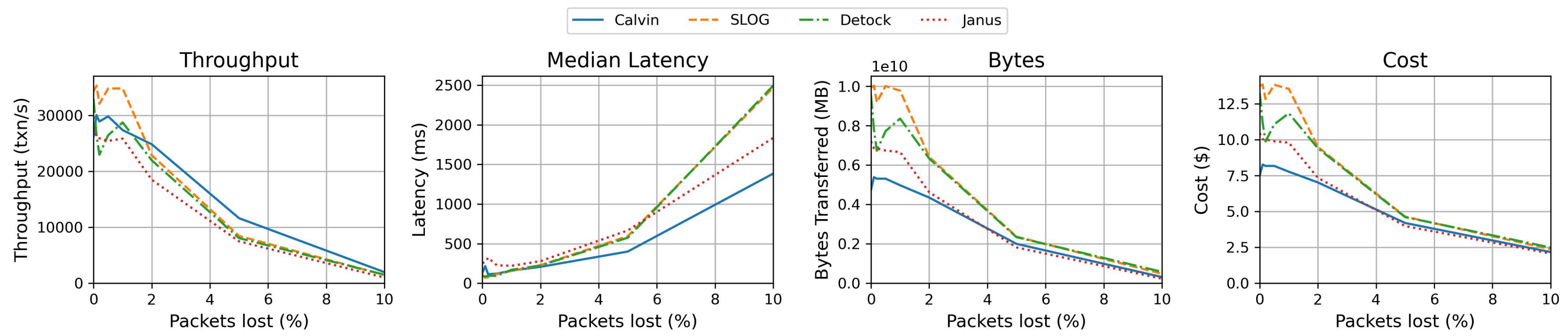




The sunflower scenario. Performance is not significantly affected.

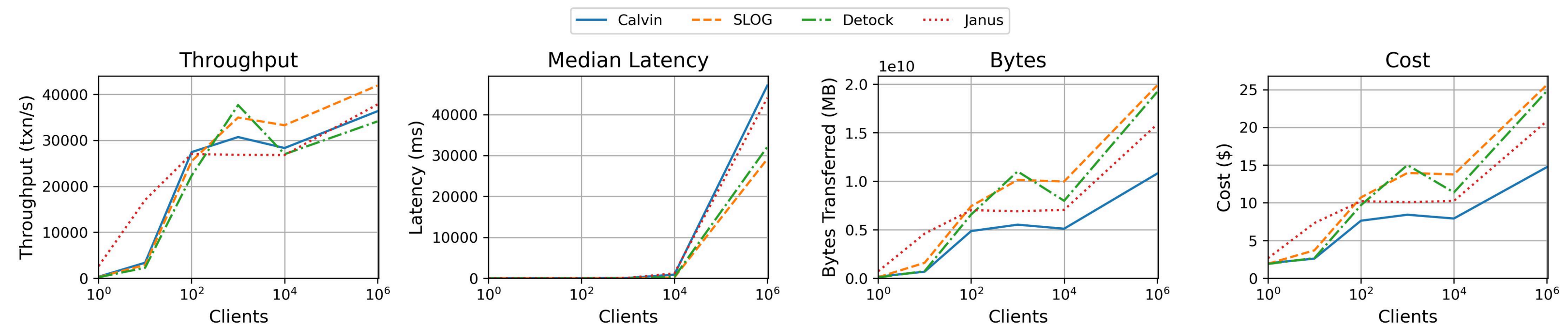
The network latency scenario. Performance drops quickly. Calvin outperforms the others.



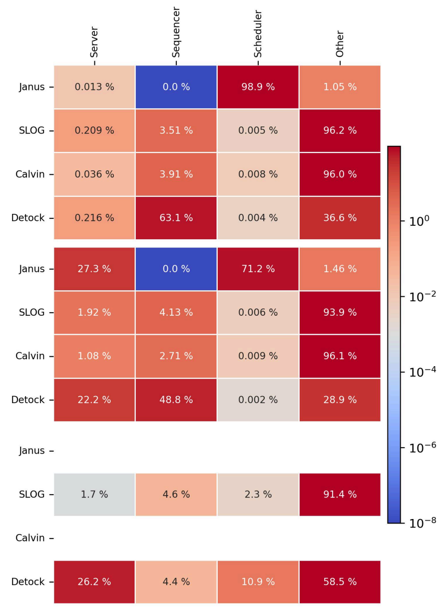


The packet loss scenario has results similar to the network latency scenario. Performance drops quickly and Calvin outperforms the others.

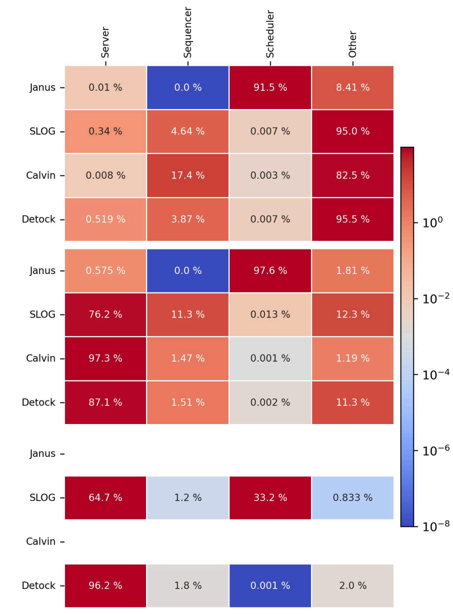
The scalability scenario. Throughput increases until fully saturated. At this point, it stagnates and latency, bytes, and cost increase.



5. Results (cont.)



Latency breakdown for the baseline scenario with large transactions.



Latency breakdown with small transactions. More time is spent coordinating, rather than idle.

6. Conclusions

We can draw three main conclusions from this data:

- Detock and SLOG perform better when there are **few multi-home** transactions

- Calvin and Janus perform consistently, and are best when there are **many multi-home transactions**.

- Calvin is the most **cost-effective** and performs best in **poor network conditions**.

Future work:
- Add more databases.
- Evaluate the effect of multi-partition transactions on performance.

7. Limitations

The main limitations of this work are the following:
- Client number may be too high and impacting latency.
- Latency breakdown is not detailed enough.
- DeathStar hotel benchmark does not have enough contention for skew or sunflower.
- Cross region latency is too low (not accurate simulation of real database configurations).

8. References

[1] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In Proceedings of Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '19, page 3–18. ACM, April 2019.

[2] Cuong D. T. Nguyen, Johann K. Miller, and Daniel J. Abadi. Detock: High performance multi-region transactions at scale. Proceedings of the ACM on Management of Data, 1(2):1–27, June 2023.

[3] Alexander Thomson, Thaddeus Diamond, Shu-Chun Weng, Kun Ren, Philip Shao, and Daniel J. Abadi. Calvin: fast distributed transactions for partitioned database systems. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD/PODS '12. ACM, May 2012.

[4] Kun Ren, Dennis Li, and Daniel J. Abadi. Slog: serializable, low-latency, geo-replicated transactions. Proceedings of the VLDB Endowment, 12(11):1747–1761, July 2019.

[5] Shuai Mu, Lamont Nelson, Wyatt Lloyd, and Jinyang Li. Consolidating concurrency control and consensus for commits under conflicts. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 517–532, Savannah, GA, November 2016. USENIX Association.