Democratizing Scalable Cloud Applications

An Approach Using Stateful Functions on Streaming Dataflows

Kyriakos Psarakis

George Christodoulou

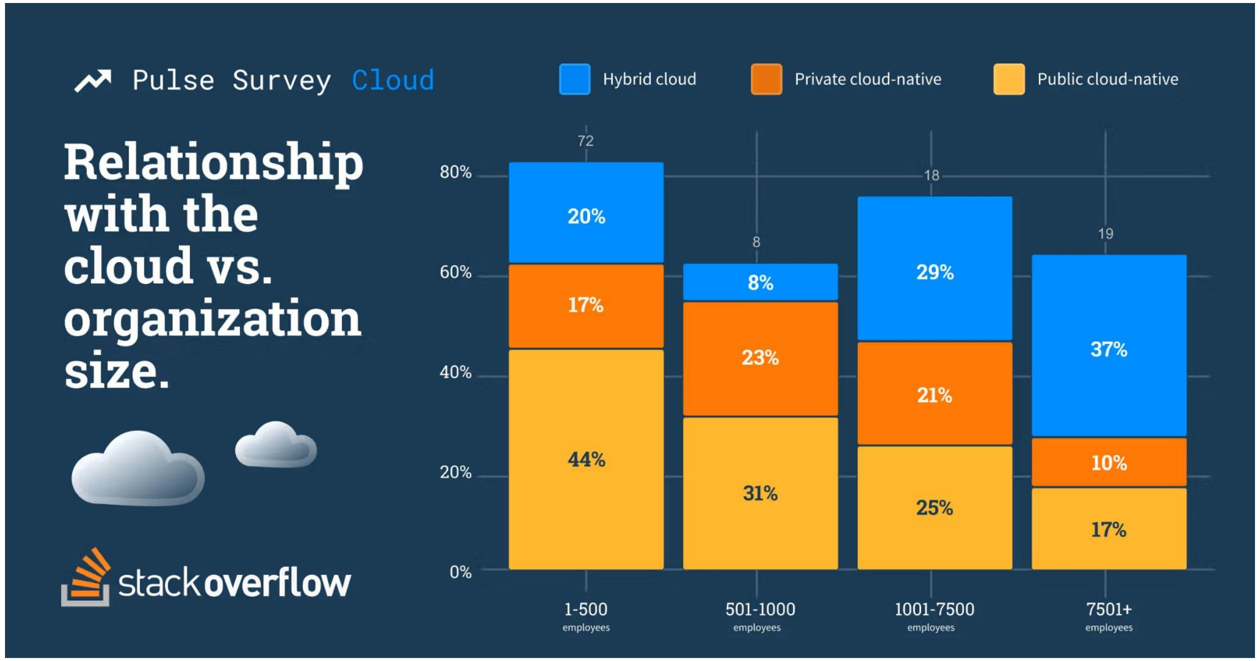
Marios Fragkoulis

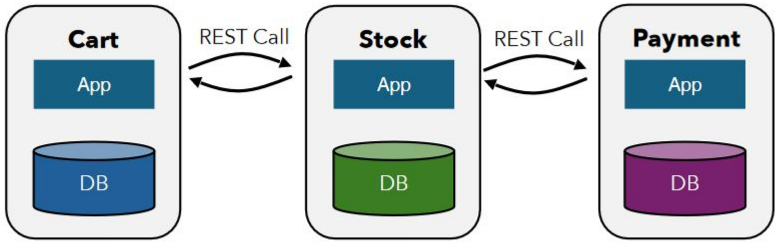
Asterios Katsifodimos

TUDelft

Cloud Transition

- Cloud adoption has reached levels above 60% in 2021
- A part of the cloud landscape is scalable cloud applications



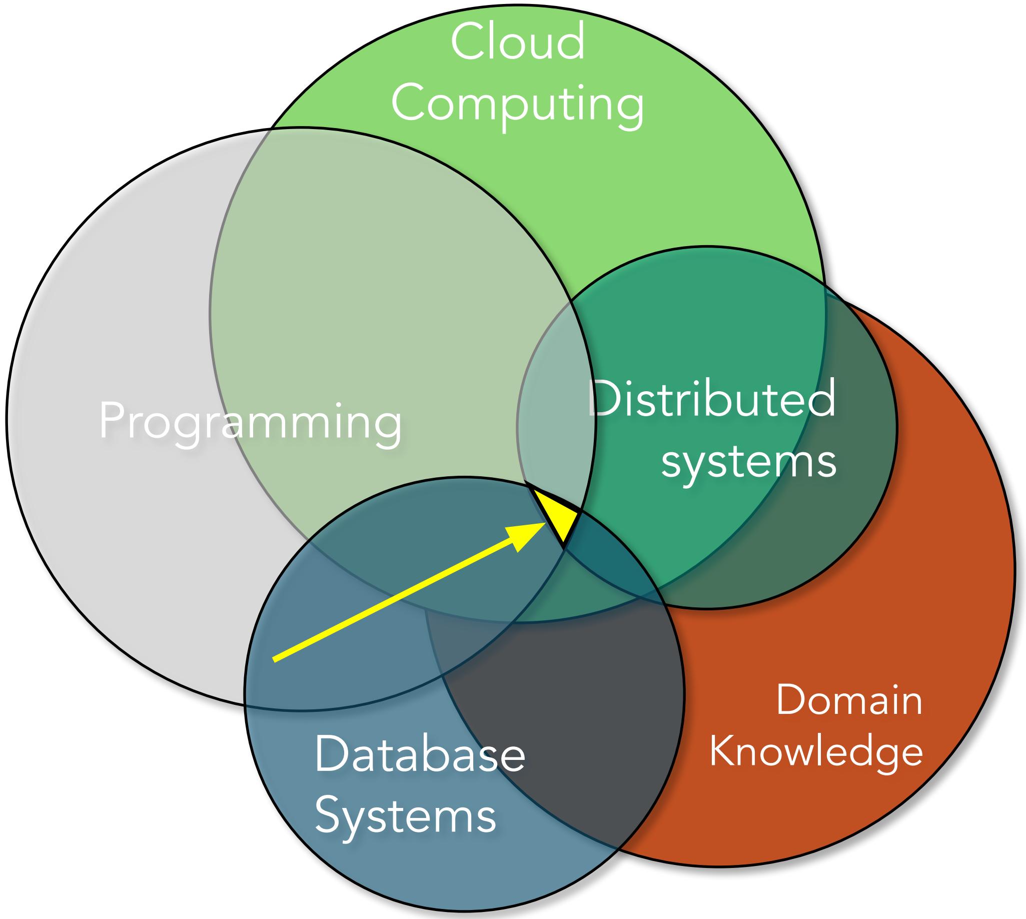


Shopping Cart Cloud Deployment

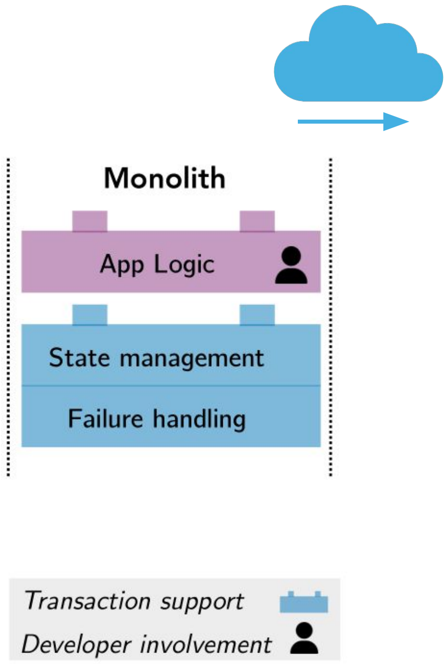
Challenges

- Deep expertise in multiple domains

- Resurfaces issues solved by database systems:
  - Transactions
  - Fault tolerance



Towards the Ideal Cloud Runtime



Monolith

App Logic

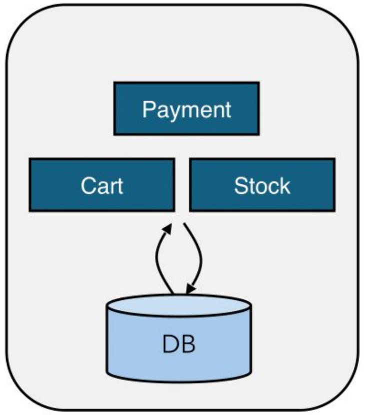
State management

Failure handling

Transaction support

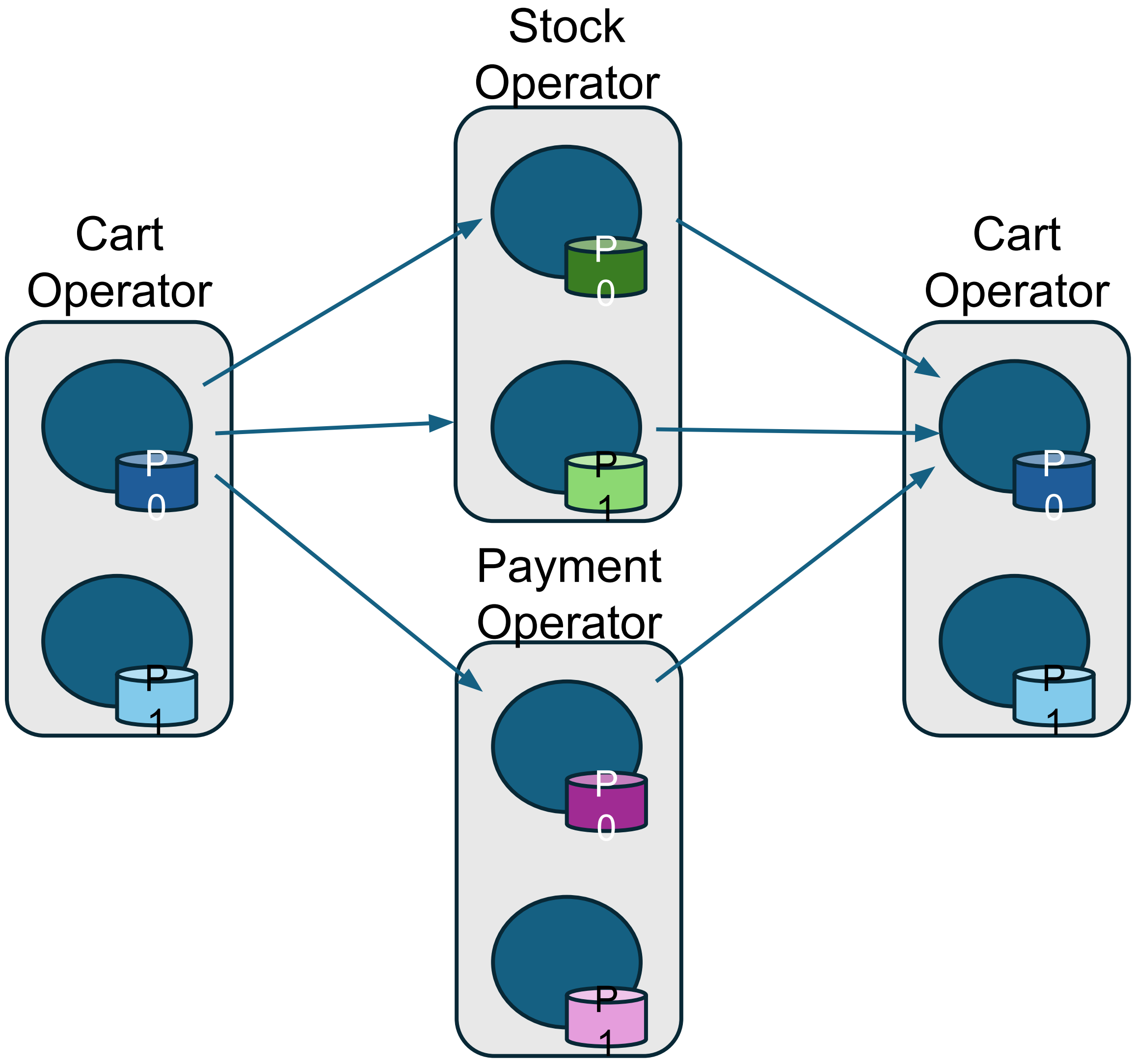
Developer involvement

The Dataflow Realization



a) Monolith

The Dataflow Realization

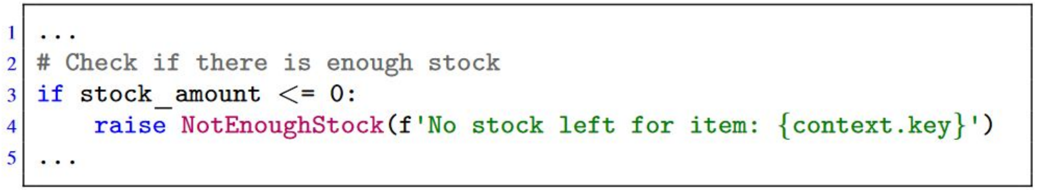


Benefits of the dataflow design:

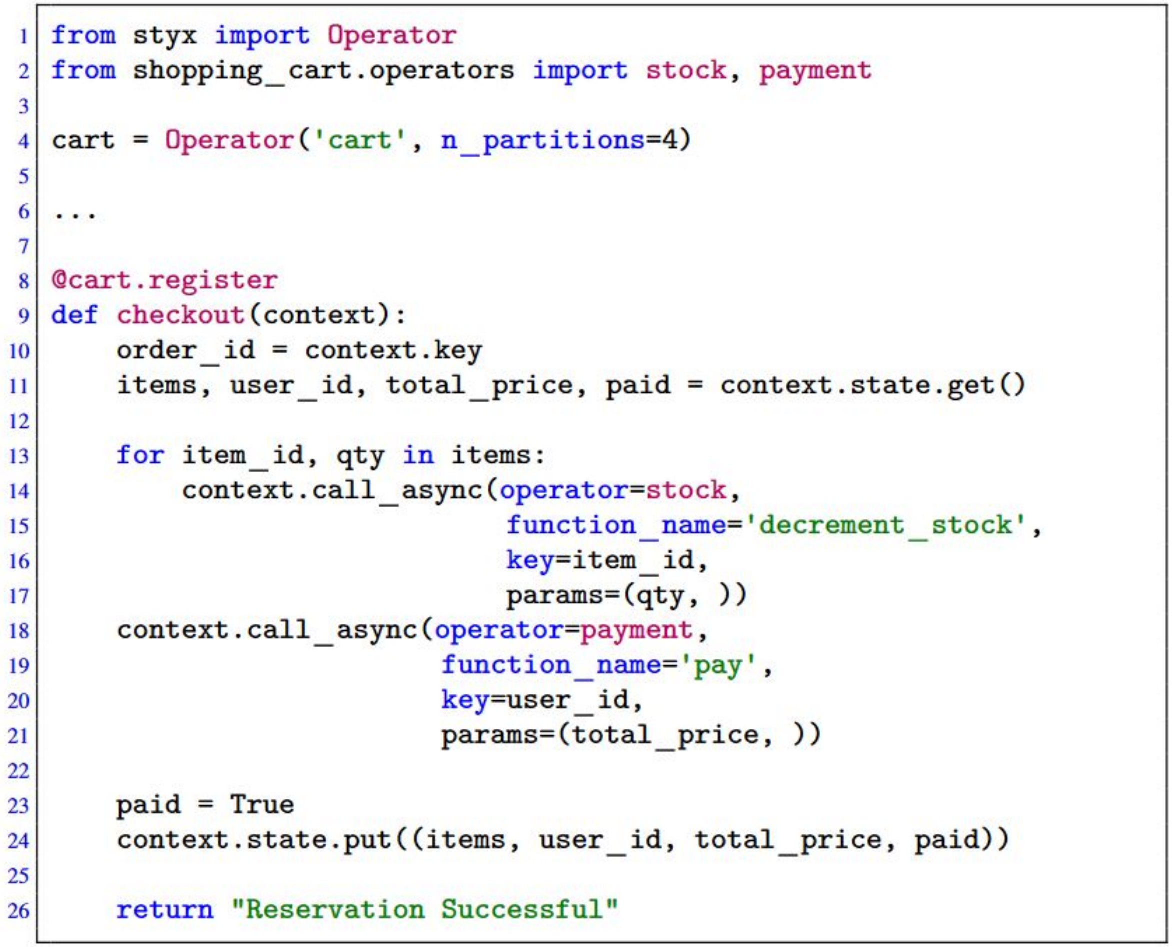
- Solution native to the problem
- Strong guarantees
- High performance
- Coarse grained fault tolerance
- Clean API

Styx's Low-Level API

- Styx works with Stateflow[1]
- Styx low-level API does not expose:
  - Failure handling
  - Transactional semantics
  - Scalability



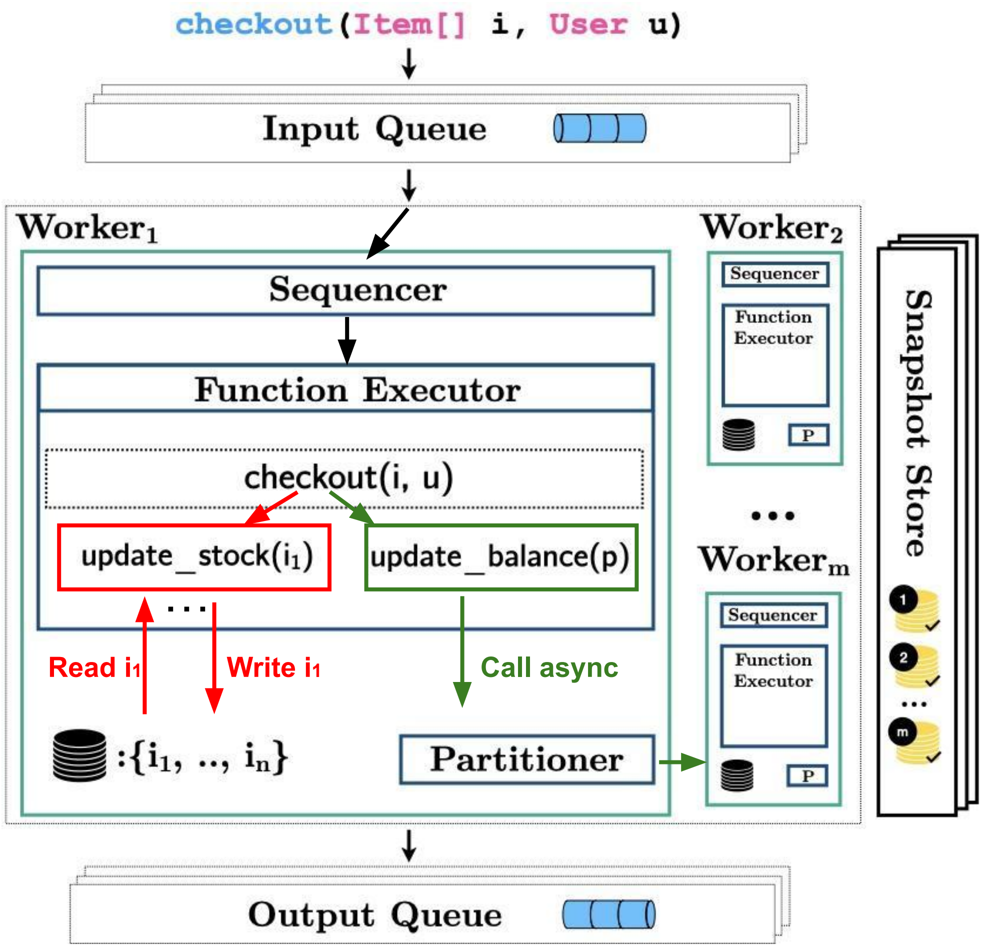
```
...
# Check if there is enough stock
if stock_amount <= 0:
    raise NotEnoughStock(f'No stock left for item: {context.key}')
...
```



```
from styx import Operator
from shopping_cart.operators import stock, payment

cart = Operator('cart', n_partitions=4)

...

@cart.register
def checkout(context):
    order_id = context.key
    items, user_id, total_price, paid = context.state.get()

    for item_id, qty in items:
        context.call_async(operator=stock,
                           function_name='decrement_stock',
                           key=item_id,
                           params=(qty, ))
    context.call_async(operator=payment,
                       function_name='pay',
                       key=user_id,
                       params=(total_price, ))

    paid = True
    context.state.put((items, user_id, total_price, paid))

    return "Reservation Successful"
```

[1] K. Psarakis, W. Zorgdrager, M. Fragkoulis, G. Salvaneschi, and A. Katsifodimos. Stateful Entities: Object-Oriented Cloud Applications as Distributed Dataflows. In EDBT, 2024.

Styx Function Flow

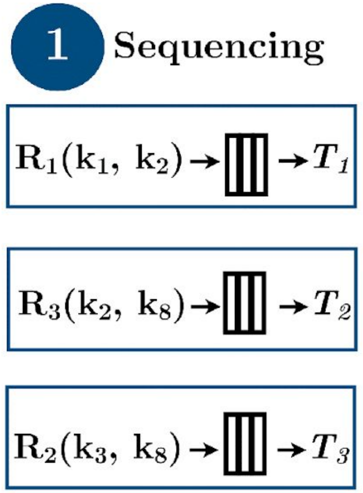


Guarantees:

1. Exactly-once processing
2. Exactly-once output
3. Durability in the Snapshot store and I/O queues

Deterministic Transaction Execution

- Epoch-based deterministic protocol (ensures serializability of the sequence)
- Enables Early-Commit replies (replies to clients before persistence in durable storage)



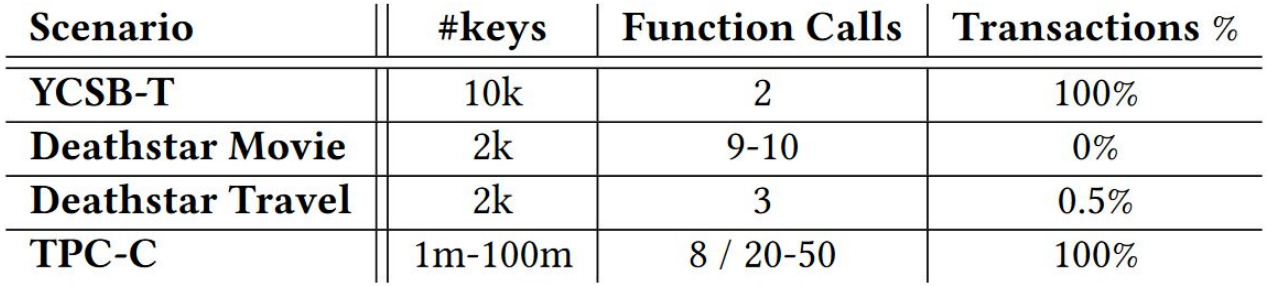
1 Sequencing

$R_1(k_1,\ k_2) \rightarrow$ $\rightarrow T_1$

$R_3(k_2,\ k_8) \rightarrow$ $\rightarrow T_2$

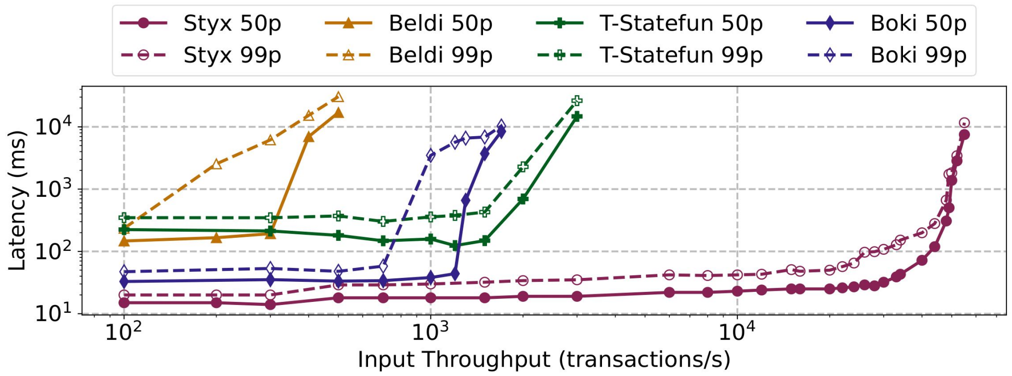
$R_2(k_3,\ k_8) \rightarrow$ $\rightarrow T_3$

Experimental Results

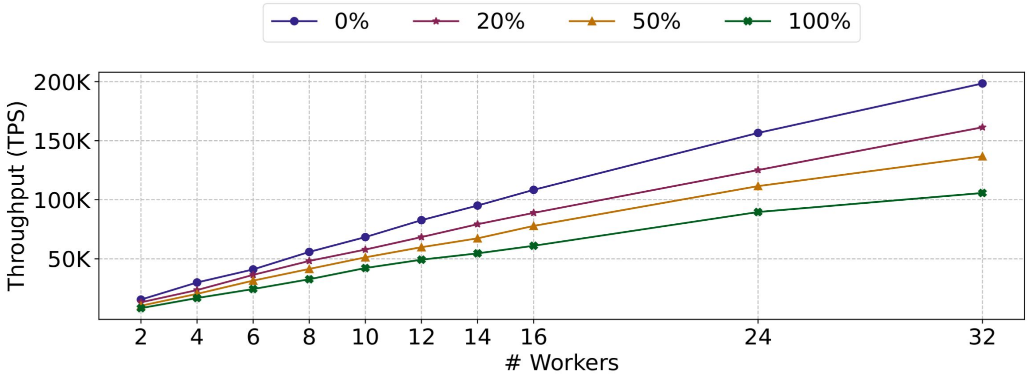


Scenario	#keys	Function Calls	Transactions %
YCSB-T	10k	2	100%
Deathstar Movie	2k	9-10	0%
Deathstar Travel	2k	3	0.5%
TPC-C	1m-100m	8 / 20-50	100%

YCSB-T Performance



YCSB-T Scalability (multipartition%)



[Beldi] H. Zhang et al. Fault-tolerant and transactional stateful serverless workflows. (OSDI, 2020)
[Boki] Z. Jia and E. Witchel. Boki: Stateful serverless computing with shared logs. (SOSP, 2021)

Takeaways

Collocation of state and processing is essential for low-latency & high throughput

Takeaways

Thank you!

Collocation of state and processing is essential for low-latency & high throughput

Exactly-Once processing guarantees allow for a simple & easy to use API

Deterministic transactions act as a performance enabler

Coarse grained fault tolerance adds minimal overhead

References:

[1] K. Psarakis, G. Christodoulou, M. Frankoulis, and A. Katsifodimos. **Transactional Cloud Applications Go with the (Data)Flow**. In CIDR, 2025 (to appear).

[2] K. Psarakis, W. Zorgdrager, M. Fragkoulis, G. Salvaneschi, and A. Katsifodimos. **Stateful Entities: Object-Oriented Cloud Applications as Distributed Dataflows**. In EDBT, 2024.

[3] K. Psarakis, G. Siachamis, G. Christodoulou, M. Fragkoulis, and A. Katsifodimos. **Styx: Transactional Stateful Functions on Streaming Dataflows**. In arXiv:2312.06893, 2024.

[4] M. de Heus, K. Psarakis, M. Fragkoulis, and A. Katsifodimos. **Distributed transactions on serverless stateful functions**. In DEBS 2021.