ACM/IFIP Middleware 2019

Serverless-Enabled Permissioned Blockchain for Elastic Transaction Processing

Byungsoo Oh, Daeyoung Kim

School of Computing
Korea Advanced Institute of Science and Technology (KAIST)
{bsoh, kimd}@kaist.ac.kr

Overview

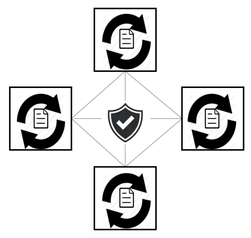
Increased variety of smart contract tasks

↓

Increased variety of computational requirements

→

Scalability issues



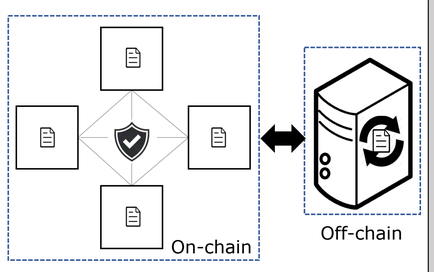
Distributed consensus on every transaction **significantly limits throughput** of the entire system

Exacerbated if
✓ degree of concurrent requests is high
✓ a node has limited compute resources

→

Serverless-based off-chain computation

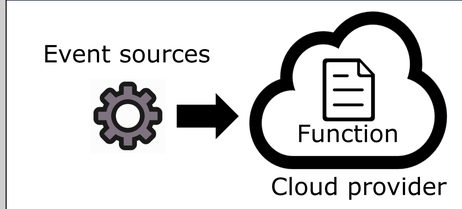
Off-chain computation



Allowing **centralized** execution **outside** blockchain

↔

Serverless computing



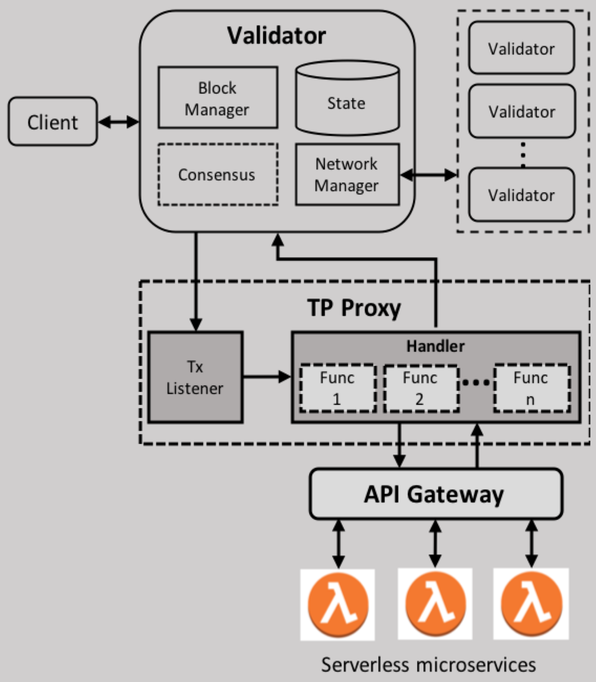
✓ Event-driven architecture
✓ No overheads for infrastructure management (resource provision)
✓ Cost effectiveness (micro-billing)

Highly **elastic** and **scalable** transaction processing enabled by launching serverless smart contracts on demand

System Design

System Components

- **Client**: creates batches of transactions and sends them to a validator via REST API. Access control and a workload generator are included.
- **Validator**: requests transaction processor proxy to execute batches of transactions, and handles persistence of blocks and states, networking, and consensus.
- **Transaction processor proxy**: handles incoming txs via transaction listener by supporting a hybrid execution model of on-chain and off-chain computations. Parallel invoker enables concurrent function invocation.
- **API gateway and serverless functions**: API gateway invokes the corresponding function instance. Given input states and parameters, a stateless function computes state transition and returns output states.



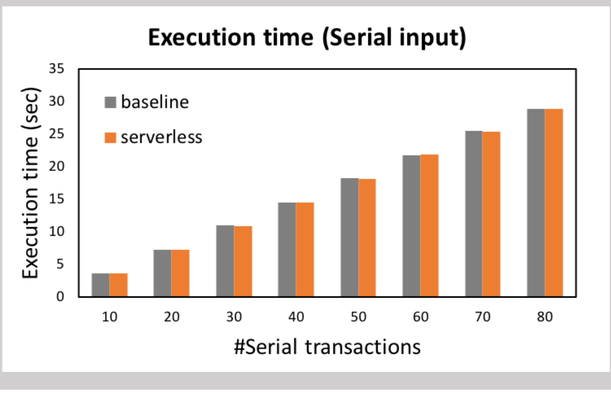
Experiments and Results

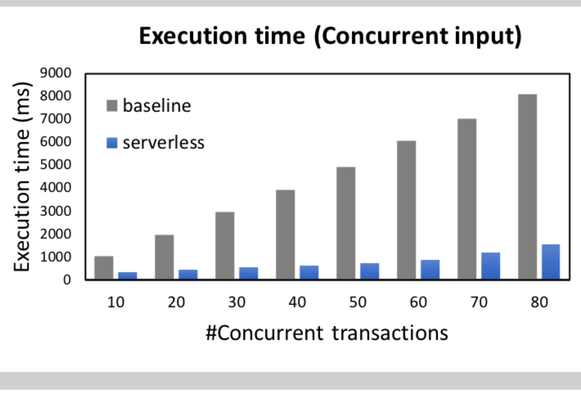
Experiment description

Goal
Our experiments on prototype system we built are aimed at understanding the performance impacts of serverless execution layer and gaining insights for proper design decisions.
Baseline
Deploying single "serverful" off-chain sandboxed process (container)
Experiments
1. Metric: execution time (variable: # of serial transactions)
2. Metric: execution time (variable: # of concurrent transactions)

Results and Discussion

✓ Performance overheads of communications with external cloud services (API gateway and serverless functions) is negligible for **processing normal serial input**.
✓ our model shows superior results when **processing parallel bursts of input**. While level of parallelism for baseline is 3.18, it is 17.78 for serverless model on average.





Contributions and Future Work

Summary & Contribution of this work

In this work, we propose a novel off-chain compute model using serverless framework. Event-driven nature of serverless functions coupled with supports for parallel invocations lead to increased elasticity of off-chain transaction executions.

The contributions of this work are as follows.

- We highlight the need for high-performance off-chain model in permissioned blockchain.
- We show that serverless computing naturally fits with permissioned blockchain's off-chain computation tasks.
- We demonstrate that serverless-based off-chain computation model has significant advantages over serverful counterpart including performance gains.

Work-in-progress & Future direction

We are continuously working on this project to enhance the performance and robustness of our proposed system model. In this context, we are currently exploring the in-depth design and implementation of hybrid execution model. To further improve on the robustness of our model, we are studying the performance with diverse tasks and workloads, and examine security aspects of off-chain model for permissioned blockchain.

In short, our current work-in-progress and future work encompasses:
× Design and implementation of hybrid execution model
× Empirical study on diverse real-world and synthetic workloads
× Support for securing integrity of execution result in off-chain serverless using trusted computing

Acknowledgments

This research was supported by Energy Cloud Technology Development Project through the Ministry of Science and ICT (MSIT) and National Research Foundation of Korea (NRF-2019M3F2A1073036), Smart City R&D project of the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (MOLIT), and the Technology Innovation Program (20006862, Development of Automated Driving Systems and Evaluation) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).