Decentralized and Stateful Serverless Computing on the Internet Computer Blockchain

Maksym Arutyunyan, Andriy Berestovskyy, Adam Bratschi-Kaye, Ulan Degenbaev, Manu Drijvers, Islam El-Ashi, Stefan Kaestle, Roman Kashitsyn, Maciej Kot, Yvonne-Anne Pignolet, Rostislav Rumenov, Dimitris Sarlis, Alin Sinpalean, Alexandru Uta, Bogdan Warinschi, Alexandra Zapuc

Presentation by: Alexandru Uta
What is the Internet Computer?

Vision:
Platform to run efficiently any computation in a decentralized and secure manner
Internet Computer – Bird’s Eye View

- Collection of replicated state machines
- **Nodes** in independent data-centers
- Nodes are partitioned into **subnets**
- Each subnet is a replicated state machine
- Each subnet runs **canisters** (smart contracts)
Canister Smart Contracts

- Can be programmed in Rust, Motoko, Python, Javascript
- 64 GB memory/storage
- Calls possible to:
  - canisters on the same/different subnets
  - exterior (http outcalls)
IC Layers

- Networking & P2P
- Consensus
- Message Routing
- Execution Environment

Message acquisition and ordering

Deterministic computation, Replicated State Machine
## Calls exposed by canisters

<table>
<thead>
<tr>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (goes through consensus)</td>
<td>Fast (no consensus)</td>
</tr>
<tr>
<td>Persists state changes</td>
<td>State changes discarded</td>
</tr>
<tr>
<td>Replicated</td>
<td>Non-replicated</td>
</tr>
</tbody>
</table>
What’s different about the Internet Computer?

- **Byzantine** fault tolerance
  - Up to f malicious out of 3f + 1 nodes
  - Individual nodes cannot be trusted

- **Geo-replicated**
  - 549 nodes, DCs in 18 countries

- **Decentralized** (88 node providers)
  - DFINITY (or any other entity) can only access their own nodes

- **Self-governing**
  - No single entity in control of the IC
  - Votes to apply changes
Execution – Focus of This Presentation

Networking & P2P

Consensus

Message Routing

Execution Environment

- Message acquisition and ordering
- Deterministic computation, Replicated State Machine
Systems Challenges

1. Statefulness
2. Deterministic Scheduling
3. Scalability*
4. Security*

*not in this talk, details in the article
Challenge 1 – Statefulness

- statefulness through *orthogonal persistence*
- Canisters are “forever-running” processes
- State is kept after replicated message execution
- No (or little) programmer work to achieve this

Motoko Key-Value store Canister

```cpp
let state = HashMap<Text, Nat64>();

public func add(key: Text, value: Nat64): async () {
    state.put(key, value)
}

public query func get(key: Text): async ?Nat64 {
    state.get(key)
}
```

Motoko Playground
Execution of Canister Messages

- Execution of message instantiates Wasmtime VM running in sandbox
- Execution happens in rounds
- Each canister can execute 1 or more messages per round
- Every N = 500 rounds, canister state is checkpointed

[Diagram of Canister Message Execution]

- Message
- Compiles to Wasm Module
- Instantiates Wasm Binary
- Sandbox
- Wasmtime
Memory Architecture
Statefulness: Tracking Changes

For Performance: Map memory pages on demand
- Page protection & signal handler to catch accesses

Canister call
1. Initially: no page is mapped
2. Read access: page fault → map r/o
3. Write access: page fault → create delta, (re-)map r/w,
Memory Optimizations

**Figure 4:** The performance improvement given by memory faulting optimizations (lower is better). Note the logarithmic vertical axis. Speedups range from 1.25X to 3.5X.
Challenge 2 – Deterministic Scheduling

- **(Sub-)Challenge 1: Determinism**
  Replicated state machine → all nodes in the state machine execute the same computation, in the same order

- **(Sub-)Challenge 2: Charging**
  Because of replicated computation can’t use “time”, but instruction counts

- **(Sub-)Challenge 3: Fairness, DoS protection**

- **(Sub-)Challenge 4: Scale**, need to support 100K+ canisters
Challenge 2 – Deterministic “Time” Slicing

* does not work on time units, but rather on numbers of instructions
Is the IC serverless?

<table>
<thead>
<tr>
<th>Serverless</th>
<th>Internet Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devs do not admin machines/nodes</td>
<td>✔️ (node providers, IC protocol)</td>
</tr>
<tr>
<td>Devs break code into small functions</td>
<td>✔️ (canisters, update/query calls)</td>
</tr>
<tr>
<td>Functions usually short-running</td>
<td>✔️ (ideally calls &lt; 1s)</td>
</tr>
<tr>
<td>Fine-grained billing</td>
<td>✔️ (at the level of instructions)</td>
</tr>
<tr>
<td>Single-cloud provider</td>
<td>✗ Decentralized</td>
</tr>
<tr>
<td>Need to use external service</td>
<td>✔️ Stateful</td>
</tr>
</tbody>
</table>
IC vs. Serverless Performance Comparison

- Compute-intensive workload
- Just execution time, no networking or other overheads
- Comparison with one of top-3 serverless platforms
- Promising performance, overheads to improve
Performance Overhead – Memory Intensive Workload

Consensus, P2P, Networking, Crypto 20%

Memory Faulting 9%

Execution Management 11%

Workload - 50%
The Internet Computer in Data

- Launch in May 2021
- Data as of Jan 2023 (more growth in the meantime)
Conclusion

- The IC has been running since May 2021
- Steadily growing in terms of users and workload
- Performance is good, but still room for improvement
- Large team effort, many thanks to all collaborators!

- Lots of systems challenges
- Join us in solving them!
- Join the IC in building new (d)apps!

IC code: [https://github.com/dfinity/ic](https://github.com/dfinity/ic)
Dashboard: [https://dashboard.internetcomputer.org/](https://dashboard.internetcomputer.org/)
Dataset API: [https://ic-api.internetcomputer.org/api](https://ic-api.internetcomputer.org/api)
Backup Slides
Message Rate
# End-to-End Performance per Subnet

<table>
<thead>
<tr>
<th>Op</th>
<th>Throughput (ops / s)</th>
<th>Latency (s)</th>
<th>Overheads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>78,000</td>
<td>0.05-0.2</td>
<td>Networking</td>
</tr>
<tr>
<td>Update</td>
<td>950</td>
<td>1-4</td>
<td>Networking, Consensus, Replicated Execution, Statefulness</td>
</tr>
</tbody>
</table>
Developers and users interact directly with Canisters
Internet Computer Consensus

Assumption: \( n > 3f \)

Guarantees **agreement** even under asynchrony

Guarantees **termination** under partial synchrony

https://internetcomputer.org/how-it-works/consensus
Chain Key Cryptography

Single 48-byte public key

for a secret-shared private key

https://internetcomputer.org/how-it-works/chain-key-technology