Catching up on the Internet Computer

May 2022

Yvonne-Anne Pignolet
Senior Research Manager, DFINITY Foundation

We are hiring!
www.dfinity.org/careers
What is the Internet Computer
Platform to run *any computation* using blockchain technology for decentralisation and security.
ICP | Internet Computer Protocol

Coordination of independent datacenters, jointly performing any computation for anyone

- Create Internet Computer blockchains.
- Ensures machines agree on sequence of computations carried out.
Canister smart contracts are fast, run in parallel, and scale...
Developers build dapps by uploading canisters to the IC.

No cloud computing necessary
Developers build dapps by uploading canisters to the IC. No cloud computing necessary.
Launched May 2021. Growing more powerful daily...

- Blocks: 8,428,492,251
- Flow: 35.70 Blocks/s
- Cycle Burn Rate: 4,933,218,936 Cycles/s
- NNS Community Fund: 1,283,707 ICP

https://dashboard.internetcomputer.org/
Fast growing blockchain ecosystem
Over 1,000 developers now building
## Comparison with other Blockchain Systems

### Layer-1 Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>Ethereum</th>
<th>Cardano</th>
<th>Solana</th>
<th>Avalanche</th>
<th>Algorand</th>
<th>Internet Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transaction Speed</strong></td>
<td>15-20 TPS</td>
<td>2 TPS</td>
<td>2,000-3,000 TPS</td>
<td>4,500 TPS</td>
<td>20 TPS</td>
<td>11,500 TPS 250,000 QPS</td>
</tr>
<tr>
<td><strong>Transaction Finality</strong></td>
<td>14 minutes</td>
<td>10-60 minutes</td>
<td>21-46 seconds</td>
<td>2-3 seconds</td>
<td>4-5 seconds</td>
<td>1 second</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Not very scalable</td>
<td>Not very scalable</td>
<td>Not very scalable</td>
<td>Not very scalable</td>
<td>More scalability</td>
<td>Indefinite scalability</td>
</tr>
<tr>
<td><strong>Node Count</strong></td>
<td>6,000 nodes</td>
<td>3,173 nodes</td>
<td>1,603 nodes</td>
<td>1,243 nodes</td>
<td>1,997 nodes</td>
<td>443 nodes</td>
</tr>
<tr>
<td><strong>Storage Costs</strong></td>
<td>$73,000,000 / GB</td>
<td>Inadequate data storage</td>
<td>$1,000,000 / GB</td>
<td>$988,000 / GB</td>
<td>IPFS off-chain storage</td>
<td>$5 / GB</td>
</tr>
<tr>
<td><strong>Cloud Service Dependency</strong></td>
<td>70% of nodes run on AWS</td>
<td>Unclear how many are cloud</td>
<td>Most nodes run on cloud</td>
<td>Unclear how many are cloud</td>
<td>Most nodes run on cloud</td>
<td>Independent data centers</td>
</tr>
</tbody>
</table>

Internet Computer Architecture
The Internet Computer is powered by a myriad of nodes

Nodes are partitioned into **subnets**.

Canister smart contracts are assigned to different subnets.
Nodes are partitioned into **subnets**.

Canister smart contracts are assigned to different subnets.

One subnet is special: it host the NNS canisters - the Network Nervous System that governs the IC

ICP token holders vote on
- Creation of new subnets
- Upgrades to new protocol version
- Replacement of nodes
- …
State:
• canisters and their queues

Inputs:
• new canisters to be installed,
• messages from users and other canisters

Outputs:
• responses to users and other canisters

Transition function:
• message routing and scheduling
• canister code

Each subnet is a replicated state machine
The layers of the Internet Computer Protocol

- Execution Environment
- Message Routing
- Consensus
- P2P

Deterministic computation

Message acquisition and ordering
The layers of the Internet Computer Protocol

- **Execution**
- **Message Routing**
- **Consensus**
- **P2P**

**Blocks**

**State**
State tree

- Huge piece of tree-structured data.
- Size: up to several GiB per canister, 100s GiBs total.
- 64KiB memory pages for efficient storage on disk
Honest replicas can fall behind
• Temporary network outage
• Power cycle
• Reboot after maintenance
• ...

How do they catch up with the rest?
State Synchronization
Requirements

- Cope with Byzantine parties
Requirements

- Cope with Byzantine parties
- Bounded Memory and disk space
Requirements

• Cope with Byzantine parties

• Bounded Memory and disk space

• Minimize bandwidth and computation complexity
Catching Up
What a node needs to fully participate in the protocol

- **Execution Environment**: 
  - Canister state: to process messages

- **Message Routing**: 
  - Queue state: to schedule and route messages

- **Consensus**: 
  - Key material: to sign and verify messages

- **P2P**: 
  - Peers info: who to connect to and how
NNS generates key of subnets and certifies them.
Basic premise: only connect to subnet overlay neighbors, at any time (to mitigate DOS attacks)

Resuming node $v$
- $v$ is initialized with NNS public key
- Can verify NNS responses
- Repeatedly NNS subnet membership
- Determine other nodes in $v$’s subnet and subnet key 🍏
What a node needs to fully participate in the protocol

- **Execution Environment**
  - Canister state: to process messages

- **Message Routing**
  - Queue state: to schedule and route messages
  - Key material: to sign and verify messages

- **Consensus**
  - Peers info: who to connect to and how
Consensus Resumability

- Easy case: missing information still available from peers

just fetch missing messages, construct blocks and execute the messages contained in them
• More difficult case: peers have purged missing information

Consensus Resumability

Replica A

24 ← 25 ← 26 ← 27 ← 28 ← 29

Replica B

24 ← 25
Checkpointing

Execution
Message Routing
Consensus
P2P

Checkpoints
State
Blocks
• More difficult case: peers have purged missing information

Consensus Resumability

Catch-up package (CUP) containing
- Key material
- Consensus information
- Hash of checkpoint

Signed with subnet key
What a node needs to fully participate in the protocol

- **Execution Environment**
  - Canister state: to process messages

- **Message Routing**
  - Queue state: to schedule and route messages

- **Consensus**
  - Key material: to sign and verify messages

- **P2P**
  - Peers info: who to connect to and how
Chunking
Security Problem: Delivery Tampering Attacks

The problem with large artifacts

- need long download timeout
  e.g., 8 MB artifact over 1Gbps connection shared by 30 nodes in rack, 25 peers per node
  = 47s expected download time → timeout > 1m30s

- can be exploited by bad peer to prevent (timely) delivery

E.g., bad peer can block statesync by

- being first peer to advertize it (skipping checks)
- send bogus data until download times out
- repeat with other bad peers until lower-ranked block finalized

Advertise large artifact
Request artifact
Send very slowly, or send bogus data
Solution: split up in smaller chunks that can be
  • requested separately
  • downloaded in parallel

Advantages:
  • shorter download timeouts
    → fail earlier
  • parallelize download from multiple peers
    → lower latency
    → better bandwidth utilisation
Tree structure of state

Up-to-date replica
- state@5000
  - canisters
    - canister 1
      - input queues
      - output queues
      - memory0
        - 0xcf624a15dbf7f70...
        - 0x13d7232f4e05100...
        - ...
    - canister 2
    - ...
  - system metadata
    - ingress history
    - streams state

Catching-up replica
- state@4000
  - canisters
    - canister 1
      - input queues
      - output queues
      - memory0
        - 0x2cd76eb1f594551...
        - 0x13d7232f4e05100...
        - ...
  - system metadata
    - ingress history
    - streams state
Design Overview

Announce state as one big artifact, use tree structure to request chunks

- Chunking mechanism
  - first request manifest with leaf/subtree hashes
  - then determine chunks to fetch (as opposed to always fetching all chunks)
Announce state as one big artifact, use tree structure to request chunks

- Chunking mechanism
  - first request manifest with leaf/subtree hashes
  - then determine chunks to fetch (as opposed to always fetching all chunks)
Design Overview

Announce state as one big artifact, use tree structure to request chunks

- Chunking mechanism
  - first request manifest with leaf/subtree hashes
  - then determine chunks to fetch (as opposed to always fetching all chunks)
Design Overview

Announce state as one big artifact, use tree structure to request chunks

- Chunking mechanism
  - first request manifest with leaf/subtree hashes
  - then determine chunks to fetch (as opposed to always fetching all chunks)
- Natural, efficient diff and de-duplication
e.g., empty (all-zero) page transmitted at most once
Summary

The Internet computer can

• Run canister smart contracts
• Serve requests at web speed
• Despite byzantine nodes

In particular, nodes can catch up quickly thanks to

• One public key per subnet, certified by NNS
• Catch Up Package containing a block with key info and checkpoint hash
• Chunking mechanism
  • first request manifest with leaf/subtree hashes
  • then determine chunks to fetch (as opposed to always fetching all chunks) from any peer