Benchmarking Tool Overview: Metrics, Data Collection, and Analysis

Introduction

The purpose of this document is to describe the objective, design, and functionality of a benchmarking tool developed to evaluate and compare the performance of the current Stratum mining protocol with Stratum V2. This tool is intended to facilitate the adoption of Stratum V2 by providing comprehensive performance metrics and insights across various mining scenarios. By enabling industry professionals to assess the improvements in decentralization, security, and efficiency brought by SV2, this tool plays a crucial role in promoting a smoother transition to the upgraded protocol, ultimately strengthening the Bitcoin network.

Overview of the Benchmarking Tool 👁️

The benchmarking tool is composed of several core components and metrics collectors, each designed to operate as Docker containers for streamlined networking and precise metric measurement. The tool covers the so-called Configuration A and C for the Stratum V2 part, always together with the SV1 scenario to compare with. Below an overview of the components needed for this and their roles.

Core Components

The following components form the backbone of the system and are crucial for running benchmarking scenarios for both Stratum V1 (SV1) and Stratum V2 (SV2):

- **SRI (SV2) Roles:** This includes Docker images for every role available in the Stratum V2 Reference Implementation (SRI) stack:
    - SV2 Pool
    - SV2 Job Declarator Server
    - SV2 Job Declarator Client
    - Translator Proxy
    - Template Provider - Pool Side
    - Template Provider - Miner Side
- **SV1 Pool:** This is the service that will be used to benchmark SV1 scenarios:
    - Public Pool by @benjamin-wilson**:** GitHub Repository

Metrics Collectors and Observers

These components are responsible for collecting all defined metrics, providing insights into system performance, and are also run as Docker containers:

- **Custom Proxies:** Collect specific time measurements and latency metrics in particular scenarios, developed using the library by @fi3: Demand Easy SV2
- **Cadvisor:** Captures fundamental metrics at a container level, such as CPU, memory, and networking usage. Metrics are exposed for collection by Prometheus. More info
- **Prometheus + Node Exporter:** Prometheus is a reliable and scalable monitoring and alerting toolkit that organizes metrics into a time-series database. Node Exporter exposes hardware and kernel-related metrics. Prometheus | Node Exporter
- **Grafana:** A visualization tool that integrates with Prometheus to create interactive and customizable dashboards.
- **Other Tools:** These include additional utilities to compute specific metrics and fulfill remaining requirements:
    - **network-traffic-metrics by @zaneclaes:** A tool based on tcpdump for retrieving specific socket usage. More info
    - **reporter by @IzakMarais:** Used for generating PDF reports from Grafana data and dashboards. More Info

Metrics analyzed 📐

Following the grafana dashboard structure, which wraps all the metrics currently captured by the benchmarking tool, here's the list of measurements, with related explanation.

- **Shares, blocks and templates stats**
    - SV1 shares
        - *Description*: it tracks metrics about shares submitted by the miner(s) used to benchmark SV1
        - *Data Collection Method*: a SV1 custom proxy located between the miner and the SV1 Pool forwards all the traffic and analyze it, extracting the shares metrics
        - *Data Analysis and usage*: shares are analyzed and tracked as
            - number of submitted shares
            - number of valid shares
            - number of stale shares
            - acceptance rate percentage
    - SV1 mined blocks
        - *Description*: it tracks metrics about blocks mined by the miner(s) used to benchmark SV1
        - *Data Collection Method*: a SV1 custom proxy located between the Bitcoin node and the SV1 Pool forwards all the traffic and analyzes it, extracting the blocks submissions to the network
        - *Data Analysis and usage*: blocks mined are added to a counter

    - SV2 shares
        - *Description*: it tracks metrics about shares submitted by the miner(s) used to benchmark SV2
        - *Data Collection Method*: a SV2 custom proxy located between the miner and the SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyze it, extracting the shares metrics
        - *Data Analysis and usage*: shares are analyzed and tracked as
            - number of submitted shares
            - number of valid shares
            - number of stale shares
            - acceptance rate percentage
    - SV2 mined blocks
        - *Description*: it tracks metrics about blocks mined by the miner(s) used to benchmark SV2
        - *Data Collection Method*: a SV2 custom proxy located between the miner and SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyze it, extracting the blocks submission
        - *Data Analysis and usage*: blocks mined are added to a counter
    - SV1 - block templates value
        - *Description*: it tracks metrics about block templates value extracted while mining on SV1 block templates
        - *Data Collection Method*: a SV1 custom proxy located between the Bitcoin node and SV1 Pool forwards all the traffic and analyze it, extracting the block templates value
        - *Data Analysis and usage*: block templates value is used to compare it with the SV2 ones
    - SV2 - block templates value
        - *Description*: it tracks metrics about block templates value extracted while mining on SV2 block templates
        - *Data Collection Method*: a SV2 custom proxy located between the Bitcoin node (TP) and SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyze it, extracting the block templates value
        - *Data Analysis and usage*: block templates value is used to compare it with the SV1 ones and with the current blocks mined on the network

- **Latency and time measurements**
    - Average latency with major pools (RTT)
        - *Description*: it tracks the average latency with major public pools
        - *Data Collection Method*: a script automatically computes the latency (round trip time, RTT) by subscribing to a list of public pools, computing the average value, every minute
        - *Data Analysis and usage*: the computed average latency is read by prometheus, and this value is used to enforce the latency between containerized Pool roles (Pool, Job Declarator Server) and Miner roles (Job Declarator Client, Translator) by using tc

- SV1 - time to get a new job from Pool
  - *Description*: it tracks the time it takes to receive a new job from the SV1 Pool
  - *Data Collection Method*: a SV1 custom proxy located between the miner and the SV1 Pool forwards all the traffic and analyzes it, extracting the new job notifications, tracking the timestamp. Another custom proxy between the Bitcoin node and the SV1 Pool forwards all the traffic and analyzes it, extracting the new templates sent to the SV1 Pool
  - *Data Analysis and usage*: the new templates timestamp is read by prometheus, and this value is used to compute the time it took to receive the updated job by the miner
- SV2 - time to get a new job from Pool or JDC
  - *Description*: it tracks the time it takes to receive a new job from the SV2 Pool or SV2 Job Declarator Client (JDC)
  - *Data Collection Method*: a SV2 custom proxy located between the miner and the SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyzes it, extracting the new job notifications, tracking the timestamp. Another custom proxy between the Bitcoin node (TP) and the SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyzes it, extracting the NewTemplate message sent to the SV2 Pool or SV2 Job Declarator Client (JDC)
  - *Data Analysis and usage*: the new templates timestamp is read by prometheus, and this value is used to compute the time it took to receive the updated job by the miner
- SV1 - time to get a new job (after a new block found) from Pool
  - *Description*: it tracks the time it takes to receive a new job from the SV1 Pool
  - *Data Collection Method*: a SV1 custom proxy located between the miner and the SV1 Pool forwards all the traffic and analyzes it, extracting the new job notifications, tracking the timestamp. Another custom proxy between the Bitcoin node and the SV1 Pool forwards all the traffic and analyzes it, extracting the new templates (based on a new prev-hash) sent to the SV1 Pool
  - *Data Analysis and usage*: the new templates timestamp is read by prometheus, and this value is used to compute the time it took to receive the updated job by the miner
- SV2 - time to get a new job (after a new block found) from Pool or JDC
  - *Description*: it tracks the time it takes to receive a new job based on a new prev-hash (so after a new block is found in the network) from the SV2 Pool or SV2 Job Declarator Client (JDC)
  - *Data Collection Method*: a SV2 custom proxy located between the miner and the SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyzes it, extracting the new job notifications, tracking the timestamp. Another custom proxy between the Bitcoin node (TP) and the SV2 Pool or SV2 Job Declarator Client (JDC)

forwards all the traffic and analyzes it, extracting the SetNewPrevHash message sent to the SV2 Pool or SV2 Job Declarator Client (JDC)
        - *Data Analysis and usage*: the new templates timestamp is read by prometheus, and this value is used to compute the time it took to receive the updated job by the miner
    - SV1 block propagation time
        - *Description*: it tracks the time it takes to propagate a valid block found by the SV1 miner to the Bitcoin network
        - *Data Collection Method*: a SV1 custom proxy located between the Bitcoin node and the SV1 Pool forwards all the traffic and analyzes it, extracting the block's submission. Another custom proxy between the miner and the SV1 Pool forwards all the traffic and analyzes it, extracting the shares submission, tracking their timestamp
        - *Data Analysis and usage*: the shares submission timestamp is read by prometheus, and this value is used to compute the time it took to receive the block submission on the Bitcoin node
    - SV2 block propagation time
        - *Description*: it tracks the time it takes to propagate a valid block found by the SV2 miner to the Bitcoin network
        - *Data Collection Method*: a SV2 custom proxy located between the Bitcoin node (TP) and the SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyzes it, extracting the block's submission. Another custom proxy between the miner and the SV2 Pool or SV2 Job Declarator Client (JDC) forwards all the traffic and analyzes it, extracting the shares submission, tracking the timestamp
        - *Data Analysis and usage*: the shares submission timestamp is read by prometheus, and this value is used to compute the time it took to receive the block submission on the Bitcoin node

- **Bandwidth Usage - Mining Farm Level**
    - SV1 - Network Tx
        - *Description*: it tracks the amount of data transmitted from the SV1 mining farm (machines) to the SV1 Pool.
        - *Data Collection Method*: network-traffic-metrics tool (which is based on *tcpdump*) is used to constantly track the number of bytes transmitted between every role included in the testing environment
        - *Data Analysis and usage*: the output tracked by reporter is then filtered according to the IPs of interested to get the outgoing traffic
    - SV1 - Network Rx
        - *Description*: it tracks the amount of data transmitted from the SV1 Pool to the SV1 mining farm (machines)
        - *Data Collection Method*: network-traffic-metrics tool (which is based on *tcpdump*) is used to constantly track the number of bytes transmitted between every role included in the testing environment
        - *Data Analysis and usage*: the output tracked by reporter is then filtered according to the IPs of interested to get the incoming traffic

    - SV2 - Network Tx
        - *Description*: it tracks the amount of data transmitted from the SV2 mining farm roles (SV2 Translator, SV2 JDC, and TP) to the SV2 Pool, SV2 JDS, and other Bitcoin peers.
        - *Data Collection Method*: network-traffic-metrics tool (which is based on *tcpdump*) is used to constantly track the number of bytes transmitted between every role included in the testing environment
        - *Data Analysis and usage*: the output tracked by reporter is then filtered according to the IPs of interested to get the outgoing traffic
    - SV2 - Network Rx
        - *Description*: it tracks the amount of data transmitted from the SV2 mining farm roles (SV2 Translator, SV2 JDC, and TP) to the SV2 Pool, SV2 JDS, and other Bitcoin peers.
        - *Data Collection Method*: network-traffic-metrics tool (which is based on *tcpdump*) is used to constantly track the number of bytes transmitted between every role included in the testing environment
        - *Data Analysis and usage*: the output tracked by reporter is then filtered according to the IPs of interested to get the outgoing traffic

- **Bandwidth Usage - Pool Level**
    - SV1 - Network Tx
        - same as for the aforementioned, but from the Pool's perspective
    - SV1 - Network Rx
        - same as for the aforementioned, but from the Pool's perspective
    - SV2 - Network Tx
        - same as for the aforementioned, but from the Pool's perspective
    - SV2 - Network Rx
        - same as for the aforementioned, but from the Pool's perspective

- **SV1 roles performances**
    - Pool roles - CPU usage
        - *Description*: it tracks the CPU usage (percentage) due to SV1 roles running as Pool's infrastructure (SV1 Pool and SV1 Bitcoin node)
        - *Data Collection Method*: cadvisor is used to constantly track the CPU usage of every Docker container
    - Pool roles - memory usage
        - *Description*: it tracks the memory usage due to SV1 roles running as Pool's infrastructure (SV1 Pool and SV1 Bitcoin node)
        - *Data Collection Method*: cadvisor is used to constantly track the memory usage of every Docker container
    - Pool roles - Network Tx
        - *Description*: it tracks the amount of data transmitted from the SV1 roles running as Pool's infrastructure (SV1 Pool and SV1 Bitcoin node)
        - *Data Collection Method*: cadvisor is used to constantly track the bandwidth usage of every Docker container

        - *Data Analysis and usage*: since cadvisor tracks metrics on a container level, the total amount of bandwidth measured even includes the one consumed between SV1 Pool and Bitcoin node, which are typically ran in the same network, so it's not meaningful
    - Pool roles - Network Rx
        - *Description*: it tracks the amount of data received by the SV1 roles running as Pool's infrastructure (SV1 Pool and SV1 Bitcoin node)
        - *Data Collection Method*: cadvisor is used to constantly track the bandwidth usage of every Docker container
        - *Data Analysis and usage*: since cadvisor tracks metrics on a container level, the total amount of bandwidth measured even includes the one consumed between SV1 Pool and Bitcoin node, which are typically ran in the same network, so it's not meaningful

- **SV2 roles performances**
    - Pool roles - CPU usage
        - *Description*: it tracks the CPU usage (percentage) due to SV2 roles running as Pool's infrastructure (SV2 Pool, SV2 Bitcoin node (TP), SV2 Job Declarator Server)
        - *Data Collection Method*: cadvisor is used to constantly track the CPU usage of every Docker container
    - Pool roles - memory usage
        - *Description*: it tracks the memory usage due to SV2 roles running as Pool's infrastructure (SV2 Pool, SV2 Bitcoin node (TP), SV2 Job Declarator Server)
        - *Data Collection Method*: cadvisor is used to constantly track the CPU usage of every Docker container
    - Pool roles - Network Tx
        - *Description*: it tracks the amount of data transmitted from the SV2 roles running as Pool's infrastructure (SV2 Pool, SV2 Bitcoin node (TP), SV2 Job Declarator Server)
        - *Data Collection Method*: cadvisor is used to constantly track the bandwidth usage of every Docker container
        - *Data Analysis and usage*: since cadvisor tracks metrics on a container level, the total amount of bandwidth measured even includes the one consumed between SV2 Pool and Bitcoin node (TP), which are typically ran in the same network, so it's not meaningful
    - Pool roles - Network Rx
        - *Description*: it tracks the amount of data received by the SV2 roles running as Pool's infrastructure (SV2 Pool, SV2 Bitcoin node (TP), SV2 Job Declarator Server)
        - *Data Collection Method*: cadvisor is used to constantly track the bandwidth usage of every Docker container
        - *Data Analysis and usage*: since cadvisor tracks metrics on a container level, the total amount of bandwidth measured even includes the one

consumed between SV2 Pool and Bitcoin node (TP), which are typically ran in the same network, so it's not meaningful

- Miner roles - CPU usage
  - *Description*: it tracks the CPU usage (percentage) due to SV2 roles running as Miner's infrastructure (SV2 Job Declarator Client, SV2 Bitcoin node (TP), SV2 Translator)
  - *Data Collection Method*: cadvisor is used to constantly track the CPU usage of every Docker container
- Miner roles - memory usage
  - *Description*: it tracks the memory usage due to SV2 roles running as Miner's infrastructure (SV2 Job Declarator Client, SV2 Bitcoin node (TP), SV2 Translator)
  - *Data Collection Method*: cadvisor is used to constantly track the memory usage of every Docker container
- Miner roles - Network Tx
  - *Description*: it tracks the amount of data transmitted from the SV2 roles running as Miner's infrastructure (SV2 Job Declarator Client, SV2 Bitcoin node (TP), SV2 Translator)
  - *Data Collection Method*: cadvisor is used to constantly track the bandwidth usage of every Docker container
  - *Data Analysis and usage*: since cadvisor tracks metrics on a container level, the total amount of bandwidth measured even includes the one consumed between SV2 Job Declarator Client and Bitcoin node (TP), which are typically ran in the same network, so it's not meaningful
- Miner roles - Network Rx
  - *Description*: it tracks the amount of data received by the SV2 roles running as Miner's infrastructure (SV2 Job Declarator Client, SV2 Bitcoin node (TP), SV2 Translator)
  - *Data Collection Method*: cadvisor is used to constantly track the bandwidth usage of every Docker container
  - *Data Analysis and usage*: since cadvisor tracks metrics on a container level, the total amount of bandwidth measured even includes the one consumed between SV2 Job Declarator Client and Bitcoin node (TP), which are typically ran in the same network, so it's not meaningful

- **Host info**
  The following metrics are relative to the host machine where the benchmarking tool is running:
  - Uptime
  - CPU cores
  - CPU usage
  - Host Memory
  - Memory usage
  - Filesystem usage
  - Node Memory

Integration and Automation 🐳

Tool Workflow: The benchmarking tool's architecture ensures seamless integration and smooth data collection across various components. The workflow is organized as follows:

1. **Initialization:** All Docker containers, including those for SV1 and SV2 roles, metrics collectors, and supporting tools, are initiated.
2. **Data Collection:** Custom proxies, cadvisor, Prometheus, and other data collectors start monitoring and gathering metrics as defined in the configurations. These tools capture detailed performance data related to shares, blocks, latency, propagation times, bandwidth usage, and resource consumption.
3. **Metrics Aggregation:** Prometheus collects and organizes metrics from various collectors into a time-series database. Node Exporter provides additional hardware and kernel-level data to Prometheus.
4. **Visualization:** Grafana interfaces with Prometheus to present the collected data in interactive and customizable dashboards, allowing users to analyze performance metrics in real-time.
5. **Report Generation:** The reporter tool fetches data from Grafana dashboards and generates comprehensive PDF reports summarizing the performance insights.

Automation: To streamline the benchmarking process, automation is employed at various stages:

1. **Container Management:** Docker Compose scripts are used to manage the lifecycle of all Docker containers, ensuring they are correctly started, stopped, and restarted as necessary.
2. **Data Collection Automation:** Custom scripts and Prometheus configuration files automate the data collection process. These scripts ensure continuous and accurate monitoring by scheduling regular data pulls and pushing metrics into the Prometheus database.
3. **Latency Measurement:** A script automatically computes average latency with major public pools by periodically subscribing to these pools and recording round trip times (RTT).
4. **Data Analysis Automation:** Prometheus rules and alerting configurations automatically analyze collected metrics, identifying trends, anomalies, and key performance indicators.
5. **Report Automation:** The reporter tool is configured to automatically generate PDF reports at predefined intervals or upon completion of specific benchmarking scenarios, ensuring timely delivery of insights.
6. **Network Emulation:** Traffic control (tc) configurations enforce network latency between containerized components, simulating real-world conditions for accurate benchmarking.

By leveraging these automation techniques, the benchmarking tool minimizes manual intervention, ensuring a reliable and efficient benchmarking process that provides consistent and accurate performance insights for both Stratum V1 and Stratum V2 protocols.

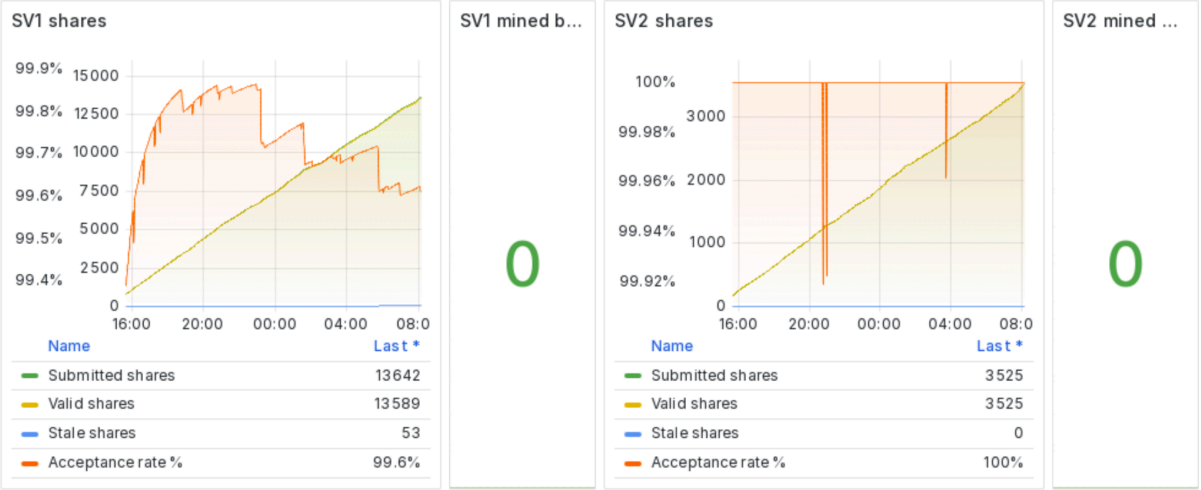
Results and Interpretation 📊

Setup for Benchmarking: For the benchmarking, Configuration A and Configuration C were utilized, each running for a duration of **16.5 hours**. In both cases, **two CPU miners** were connected to the **SV1 endpoint** and **two CPU miners** to the **SV2 endpoint**, all operating on the **testnet4** network. Both SV1 Pool and SV2 Job Declarator Client (JDC) were producing new block **templates every 60 seconds**. All the system was running on a mac equipped with a *4GHz Quad-Core Intel Core i7* processor and a *16 GB 1600 MHz DDR3* memory.

Interpretation of Results: To make informed decisions based on the presented results, consider the following analysis which can be used as guidelines for future benchmarks.

Configuration A Analysis:

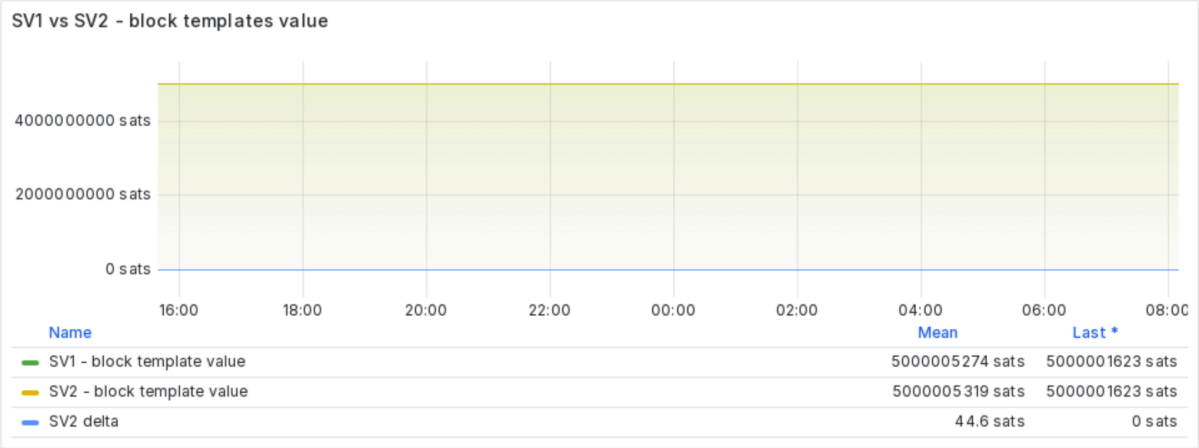
- ***Shares, blocks and templates stats***



Results:

- Shares submitted to **SV1 Pool** from the two miners have been 13642, of which 53 were stale. **Acceptance ratio: 99.6%**
- Shares submitted to **SV2 Pool** from the two miners have been 3525, of which 0 were stale. **Acceptance ratio: 100%**
- No blocks have been mined by both SV1 and SV2 pools

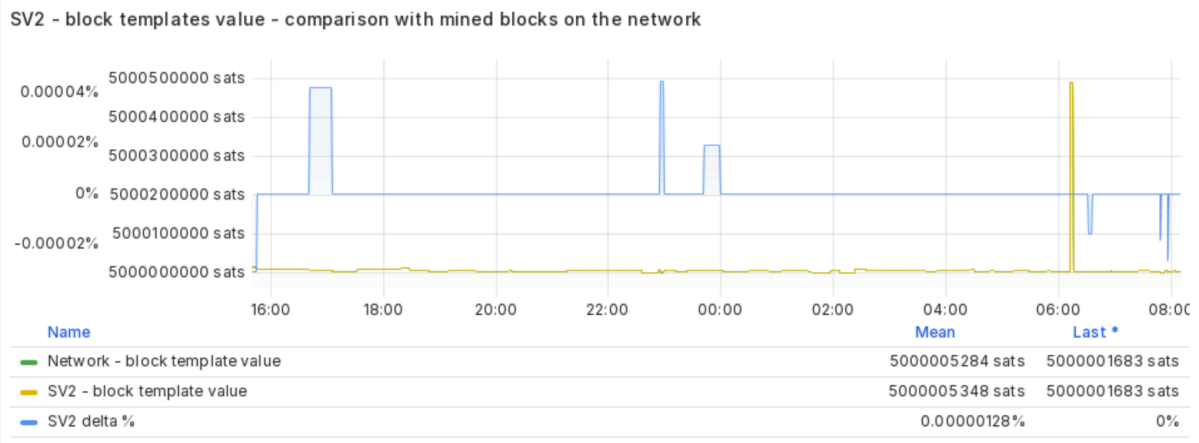
Analysis:
The presence of the SV2 Translator proxy, which aggregates downstream connections into a single one opened with the SV2 Pool has helped in reducing the number of shares sent to the pool.
In this configuration (A), the proxy is directly connected to a local Job Declarator Client (JDC) which provides the updated jobs to miners. In this way downstreams are updated with fresher jobs in a faster way, and this is the reason behind the 100% acceptance ratio in the shares sent to the SV2 Pool.



Results:

- **SV1** block template average value: **500.005.274 sats**
- **SV2** block template average value: **500.005.319 sats**
- SV2 delta: **44.6 sats**

Analysis:
Since this benchmark has been done on testnet4, the current number of txs in the mempool (and so the amount of fees) are very little. Both SV1 Pool and SV2 Job Declarator Client (JDC) were producing new block templates every 60 seconds, but since the JDC is run locally, it could have been set up to produce updated templates more often (without affecting latency measurements). The advantage of doing that it's more important on the mainnet, as it will be shown in future more specific benchmarks.

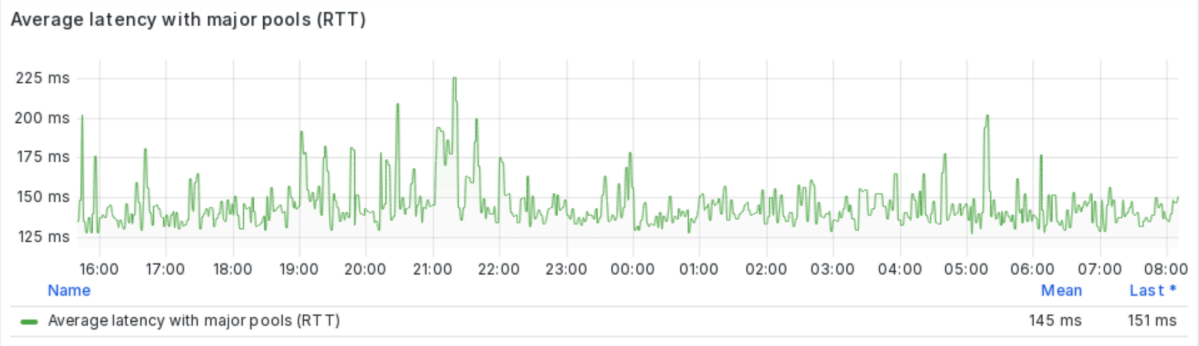


Results:

- **Network** block template average value: **500.005.284 sats**
- **SV2** block template average value: **500.005.348 sats**
- SV2 delta: **64 sats (0.00000128%)**

Analysis:
Same as above

- ***Latency and time measurements***

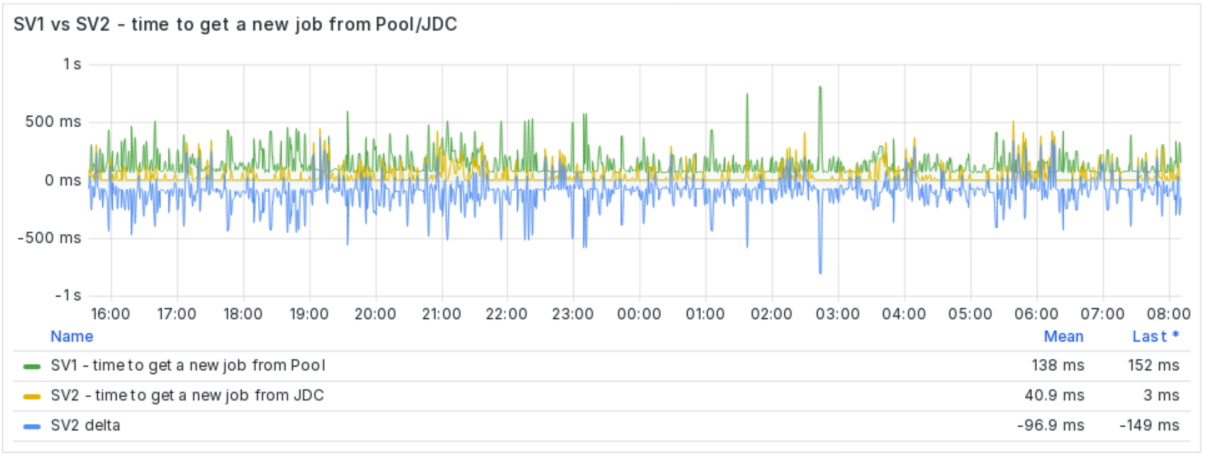


Results:

- **Average latency** with major pools (RTT): **145ms**

Analysis:

This is the average latency computed by subscribing to a list of major public pools. Similarly to a ping, it measures the Round Trip Time of packets.



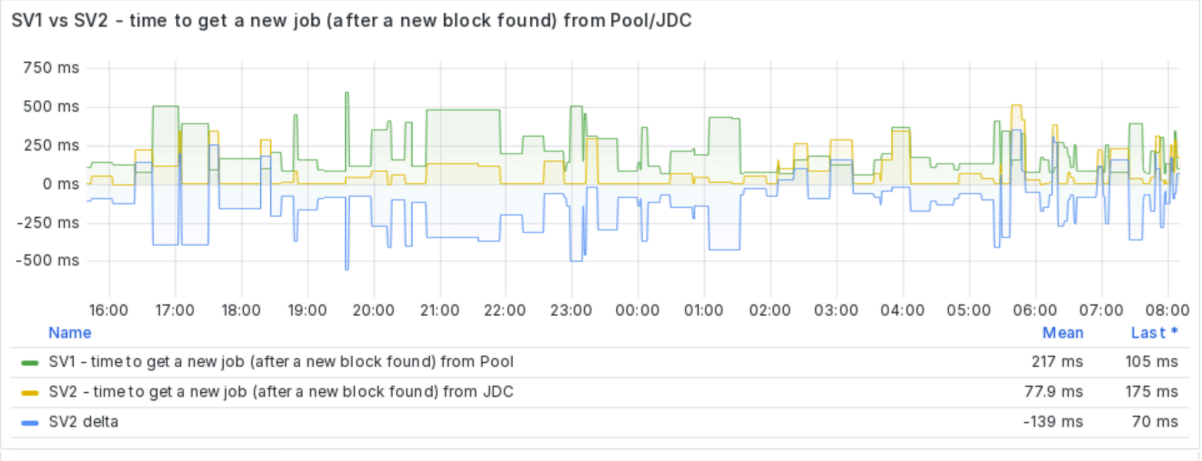
Results:

- **SV1** average time to get a new job from Pool: **138ms**
- **SV2** average time to get a new job from Job Declarator Client (JDC): **40.9ms**
- SV2 delta: **-96.9ms (-70.2%)**

Analysis:

The presence of the Job Declarator Client (JDC) in the same network in the mining farm, is drastically reducing the time it takes to send updated jobs to downstream miners. The result shows almost **70% time** which is **saved** because of it, with respect to the SV1 case, in which is the remote SV1 Pool which is sending new jobs to miners.

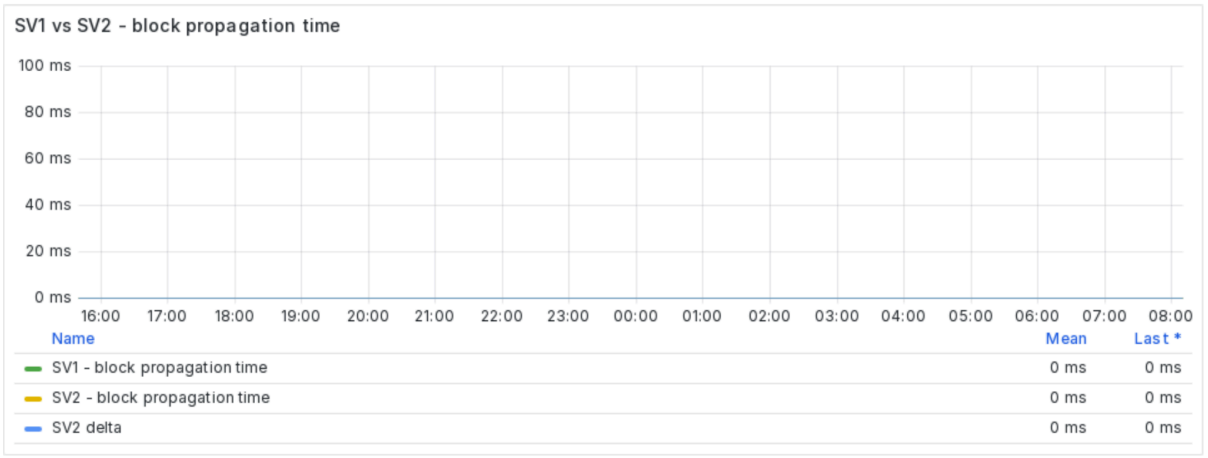
Considering a **template refresh interval** of **60s** (as in this benchmark), it means that **0,0969s** are saved **every 60s**, which leads to 139,536s every day, which means that **~849 minutes** are **saved** every **year**.



Results:

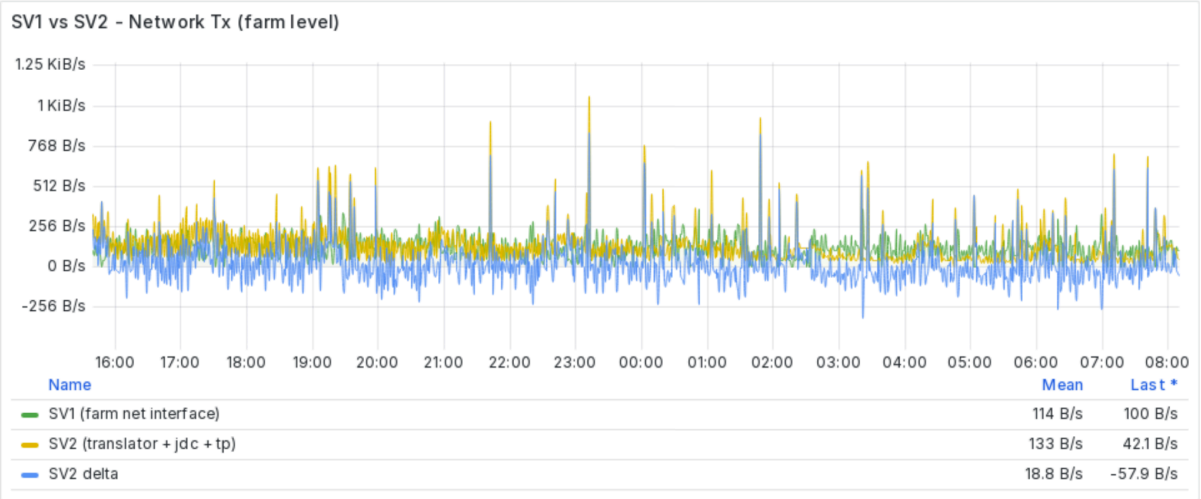
- **SV1** average time to get a new job (based on a new prev-hash) from Pool: **217ms**
- **SV2** average time to get a new job (based on a new prev-hash) from Job Declarator Client (JDC): **77.9ms**
- SV2 delta: **-139ms (-64%)**

Analysis:
The presence of the Job Declarator Client (JDC) in the same network in the mining farm, is drastically reducing the time it takes to send updated jobs (based on a new prev-hash, so after a block is found in the network) to downstream miners. The result shows **64% time** which is **saved** because of it, with respect to the SV1 case, in which is the remote SV1 Pool which is sending new jobs to miners.
Considering a **template refresh interval** of **60s** (as in this benchmark), it means that **0,139s** are saved **every 60s**, which leads to 200,16s every day, which means that **~1218 minutes** are **saved** every **year**.



Analysis:
No blocks have been found by both SV1 and SV2 pools, so no particular comment for this benchmark. A more specific analysis on this comparison will be done in a future more specific benchmark done with asic miners.

- ***Bandwidth Usage - Mining Farm Level***

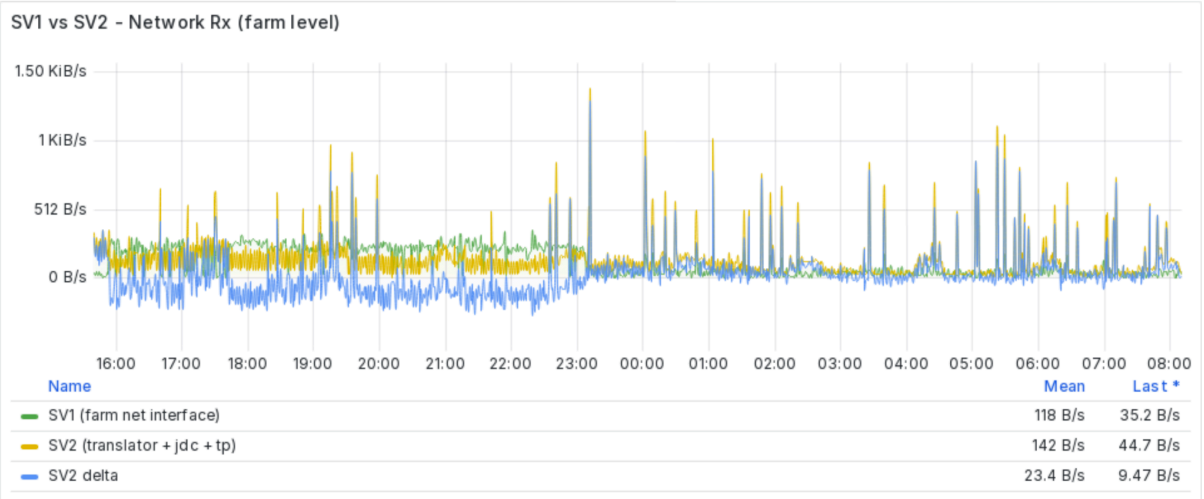


Results:

- **SV1** average bandwidth used: **114 B/s**
- **SV2** average bandwidth used: **133 B/s**
- SV2 delta: **+18.8 B/s (+16.5%)**

Analysis:

The presence of a Template Provider (which is a patched bitcoind) in the mining farm increases the bandwidth used for network transmissions. This benchmark has been done on testnet4, so values for mainnet would be even higher because of the more messages exchanged between peers in the network. It will be better analyzed in a future more specific benchmark. For sure, this configuration (A) is better suited for miners who don't have restrictions on bandwidth at a farm level.

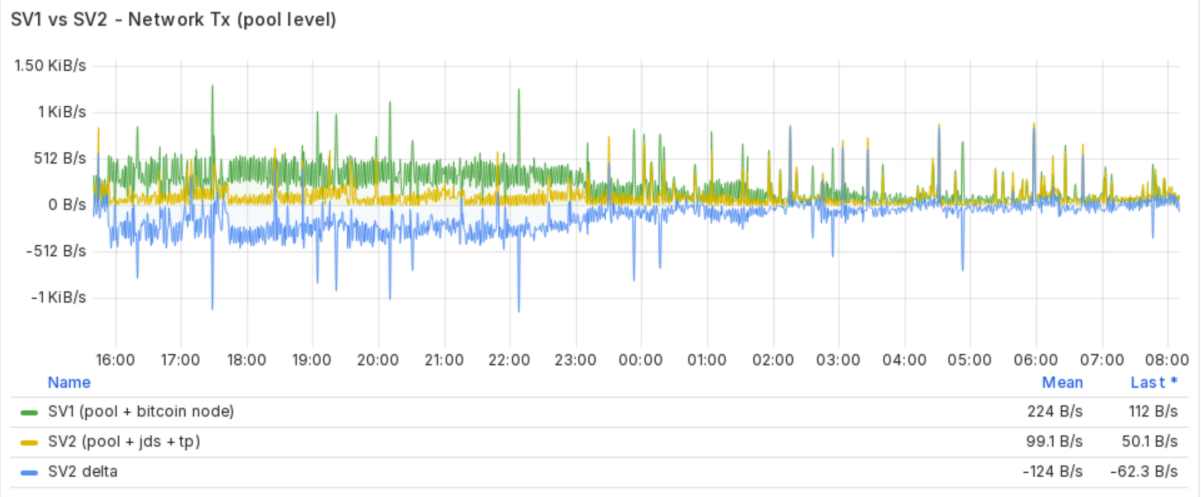


Results:

- **SV1** average bandwidth used: **118 B/s**
- **SV2** average bandwidth used: **142 B/s**
- SV2 delta: **+23.4 B/s (+19.8%)**

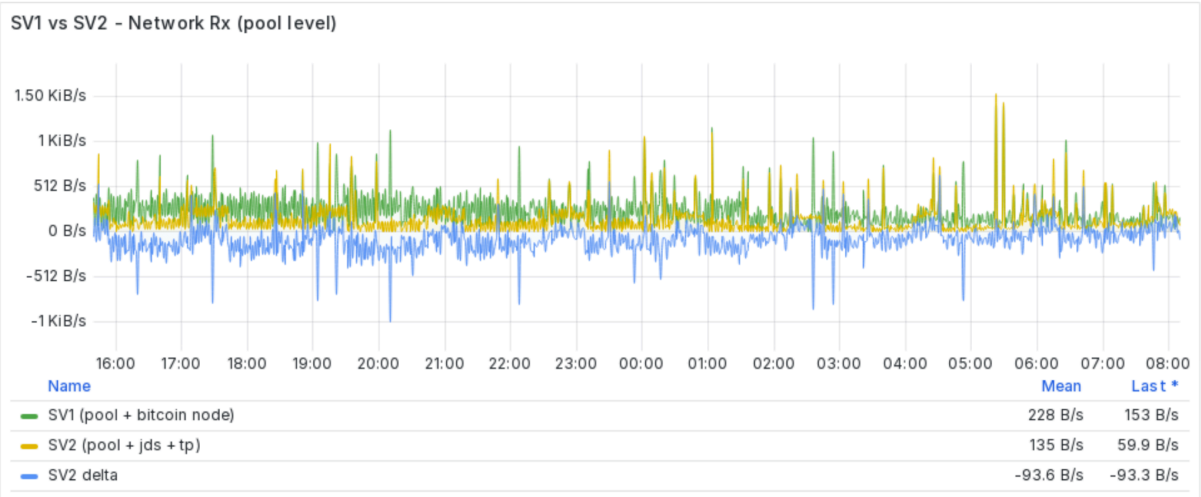
Analysis:
The presence of a Template Provider (which is a patched bitcoind) in the mining farm increases the bandwidth used for network receptions. This benchmark has been done on testnet4, so values for mainnet would be even higher because of the more messages exchanged between peers in the network. It will be better analyzed in a future more specific benchmark. For sure, this configuration (A) is better suited for miners who don't have restrictions on bandwidth at a farm level.

- ***Bandwidth Usage - Pool Level***



Results:
- **SV1** average bandwidth used: **224 B/s**
- **SV2** average bandwidth used: **99.1 B/s**
- SV2 delta: **-124 B/s (-55.35%)**

Analysis:
The presence of a Job Declarator Client (JDC) miner side drastically reduced the amount of bandwidth used by the SV2 Pool to send updated jobs downstream. This benchmark has been done with 2 miners pointed to the SV1 setup and 2 miners pointed to the SV2 setup, so the benefit would probably be even higher for a larger number of machines. This will be one point of focus in a future benchmark.

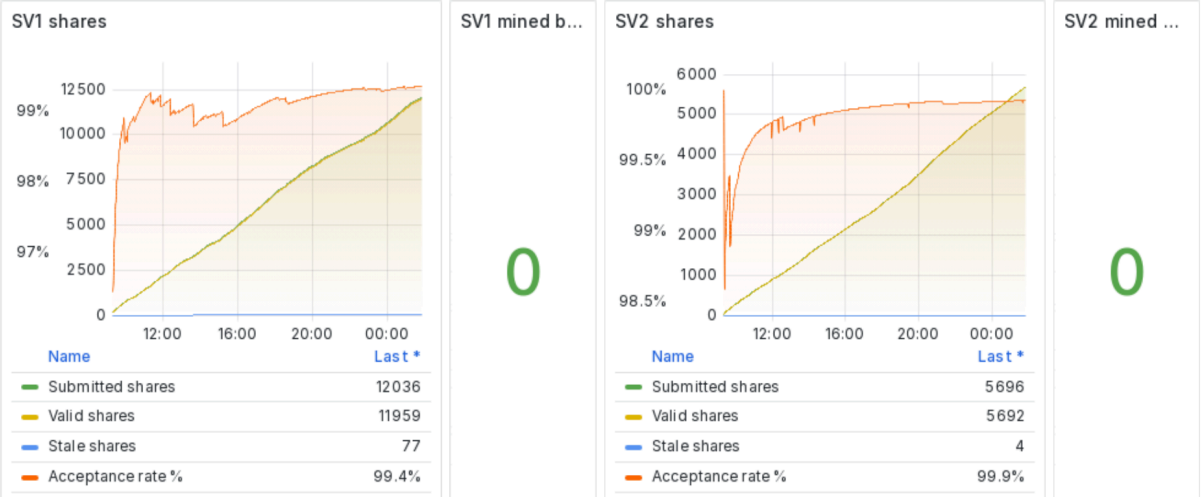


Results:
- **SV1** average bandwidth used: **228 B/s**
- **SV2** average bandwidth used: **135 B/s**
- SV2 delta: **-93.6 B/s (-41%)**

Analysis:
The presence of proxies which aggregates downstream connections miner side drastically reduced the amount of bandwidth used by the SV2 Pool to receive shares from the mining farm. This benchmark has been done with 2 miners pointed to the SV1 setup and 2 miners pointed to the SV2 setup, so the benefit would probably be even higher for a larger number of machines. This will be one point of focus in a future benchmark.

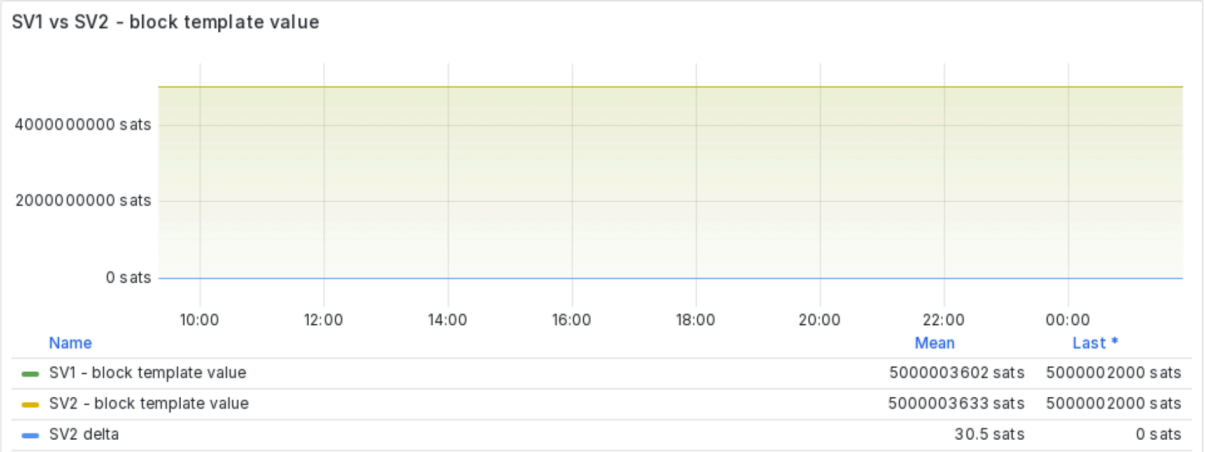
Configuration C Analysis:

- ***Shares, blocks and templates stats***



Results:
- Shares submitted to **SV1 Pool** from the two miners have been 12036, of which 77 were stale. **Acceptance ratio: 99.4%**
- Shares submitted to **SV2 Pool** from the two miners have been 5896, of which 4 were stale. **Acceptance ratio: 99.9%**
- No blocks have been mined by both SV1 and SV2 pools

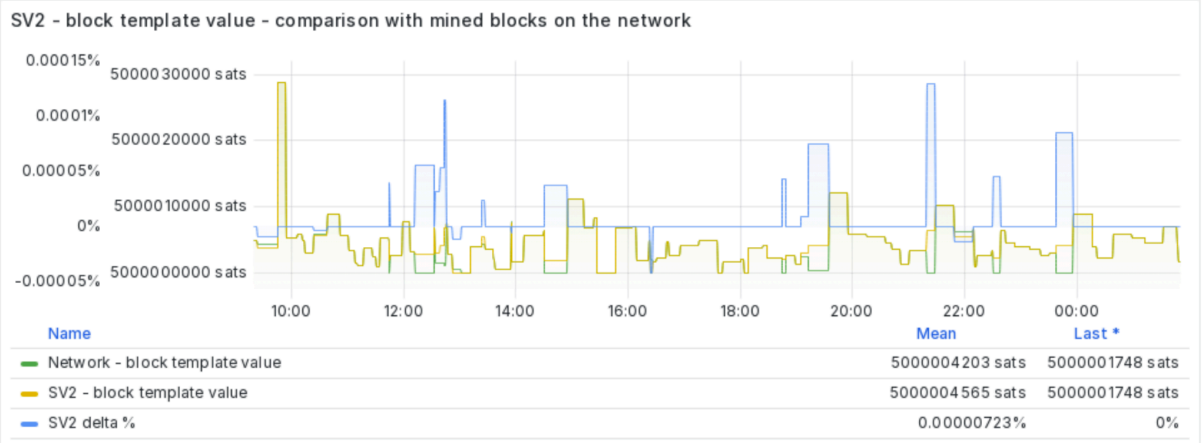
Analysis:
The presence of the SV2 Translator proxy, which aggregates downstream connections into a single one opened with the SV2 Pool has helped in reducing the number of shares sent to the pool.
In this configuration (C), SV2 mining jobs are created and sent by the SV2 Pool, as it is for the SV1 context. Even if there's a proxy in the SV2 setup, this configuration still pays the price related to latency between the pool and the miner, and that's the reason behind the 4 stale shares (which were 0 in the configuration A).



Results:

- **SV1** block template average value: **500.003.602 sats**
- **SV2** block template average value: **500.003.633 sats**
- SV2 delta: **30.5 sats**

Analysis:
Since this benchmark has been done on testnet4, the current number of txs in the mempool (and so the amount of fees) are very little. Both SV1 and SV2 Pool were producing new block templates every 60 seconds.

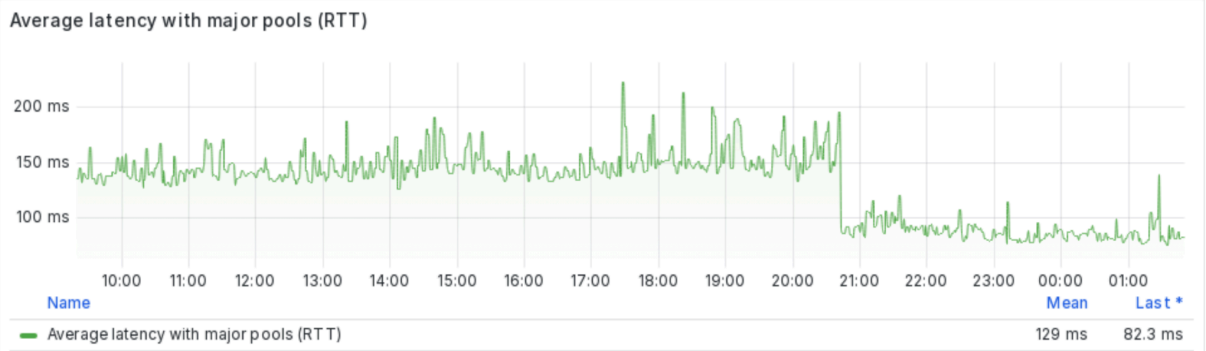


Results:

- **Network** block template average value: **500.004.203 sats**
- **SV2** block template average value: **500.004.565 sats**
- SV2 delta: **362 sats (0.00000723%)**

Analysis:
Same as above

- ***Latency and time measurements***

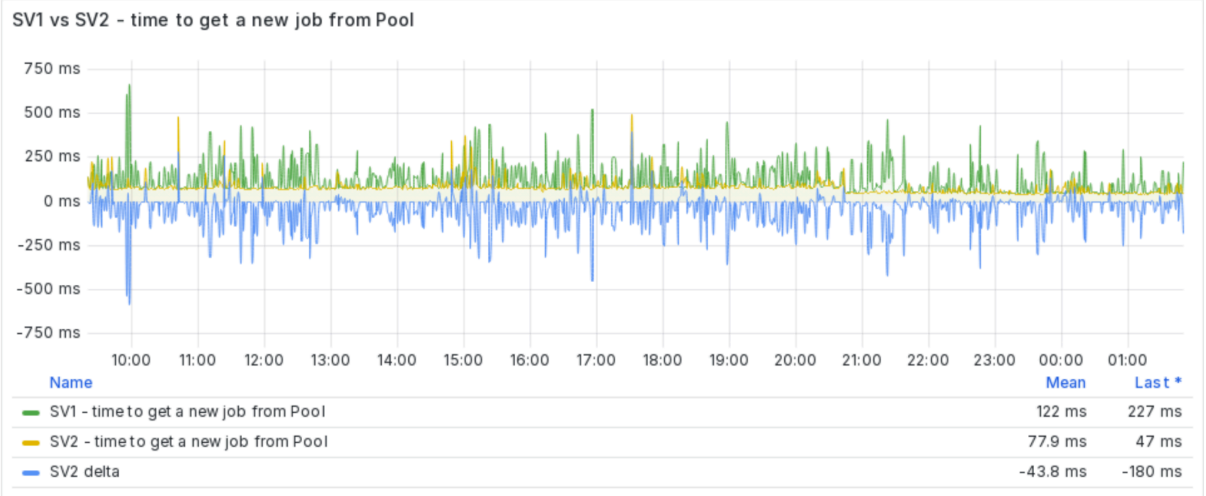


Results:

- **Average latency** with major pools (RTT): **129ms**

Analysis:

This is the average latency computed by subscribing to a list of major public pools. Similarly to a ping, it measures the Round Trip Time of packets.



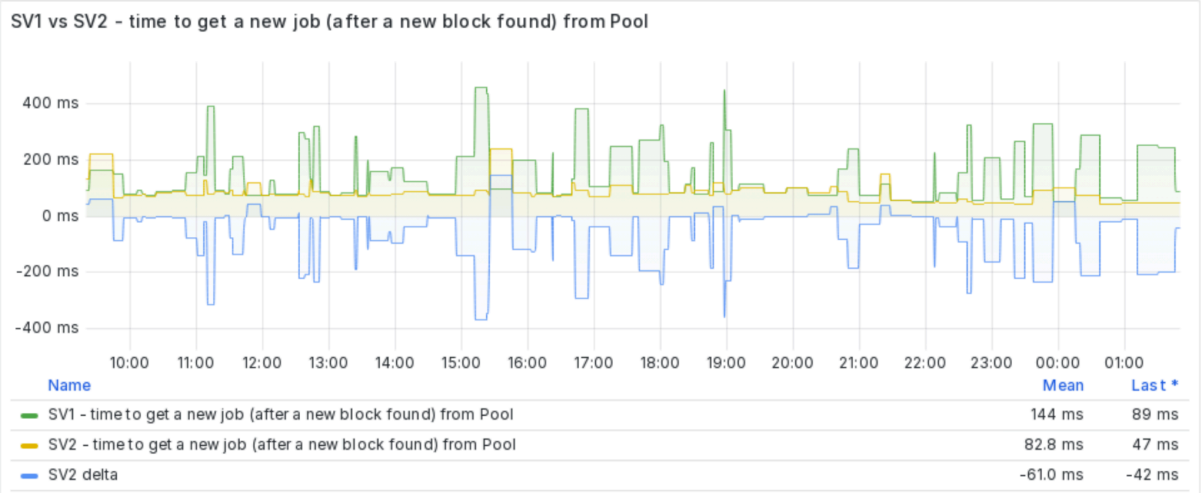
Results:

- **SV1** average time to get a new job from Pool: **122ms**
- **SV2** average time to get a new job from Job Declarator Client (JDC): **77.9ms**
- SV2 delta: **-43.8ms (-35.9%)**

Analysis:

Even if SV2 miners receive jobs from the SV2 Pool (as for SV1 scenario), the time it takes to send updated jobs to downstream miners is reduced. This is probably due to the efficiency of SV2 message framing, which results in a faster communication between pool and miners, leading to a **36% time** which is **saved** because of it.

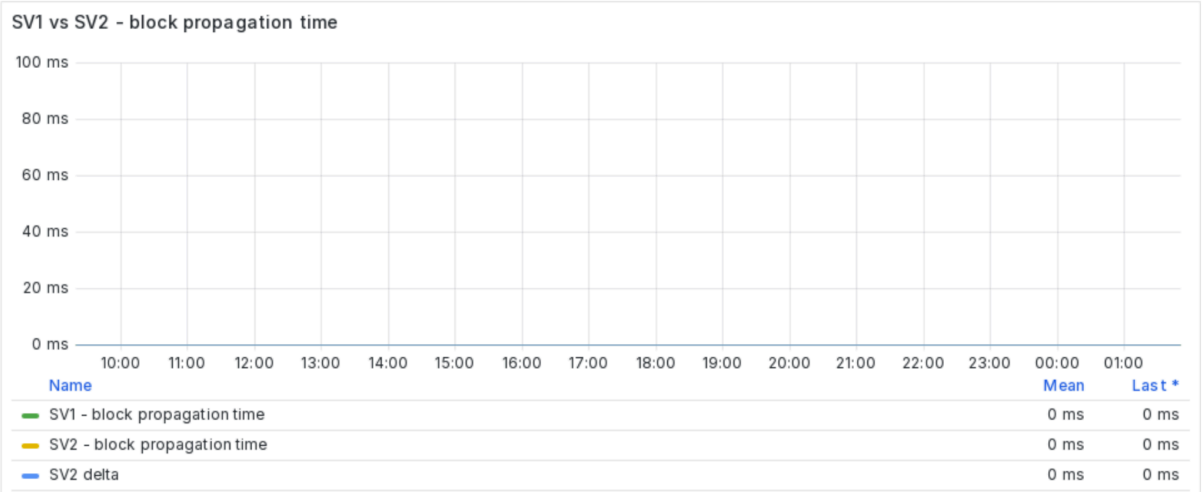
Considering a **template refresh interval** of **60s** (as in this benchmark), it means that **0,0779s** are saved **every 60s**, which leads to 112,176s every day, which means that **~682 minutes** are **saved** every **year**.



Results:

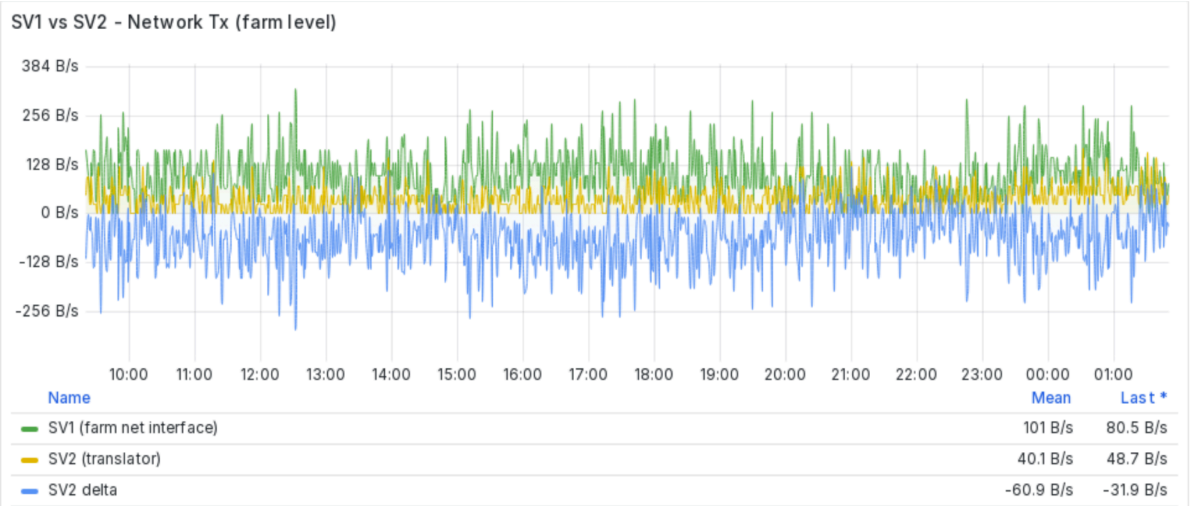
- **SV1** average time to get a new job (based on a new prev-hash) from Pool: **144ms**
- **SV2** average time to get a new job (based on a new prev-hash) from Job Declarator Client (JDC): **82.8ms**
- SV2 delta: **-61ms (-42.4%)**

Analysis:
Even if SV2 miners receive jobs from the SV2 Pool (as for SV1 scenario), the time it takes to send updated jobs (based on a new prev-hash, so after a block is found in the network) to downstream miners is reduced. This is probably due to the efficiency of SV2 message framing, which results in a faster communication between pool and miners, leading to a **42% time** which is **saved** because of it.
Considering a **template refresh interval** of **60s** (as in this benchmark), it means that **0,0828s** are saved **every 60s**, which leads to 119,232s every day, which means that **~725 minutes** are **saved** every **year**.



Analysis:
No blocks have been found by both SV1 and SV2 pools, so no particular comment for this benchmark. A more specific analysis on this comparison will be done in a future more specific benchmark done with asic miners.

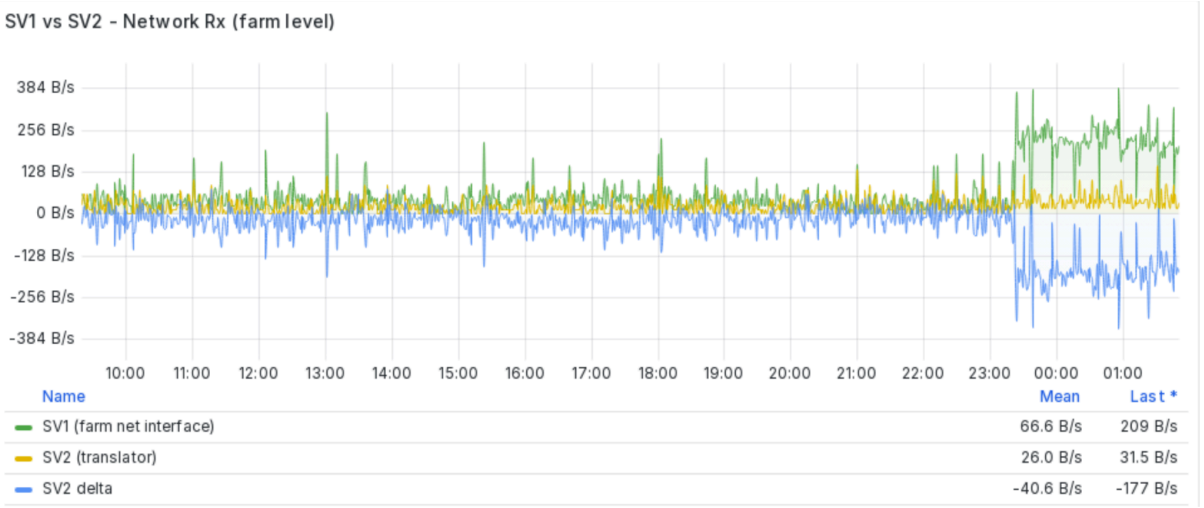
- ***Bandwidth Usage - Mining Farm Level***



Results:

- **SV1** average bandwidth used: **101 B/s**
- **SV2** average bandwidth used: **40.1 B/s**
- SV2 delta: **-60.9 B/s (-60.3%)**

Analysis:
Since in this configuration (C) there is just the SV2 Translator proxy running on site, the bandwidth used for network transmissions is just the one used to send shares upstream. Given that the proxy aggregates downstream miners connections, the result is a **60%** enhancement in bandwidth usage. Differently from configuration (A), this setup is better suited for miners who could have some restrictions on bandwidth at a farm level, and cannot afford running a Bitcoin node in their mining farm.

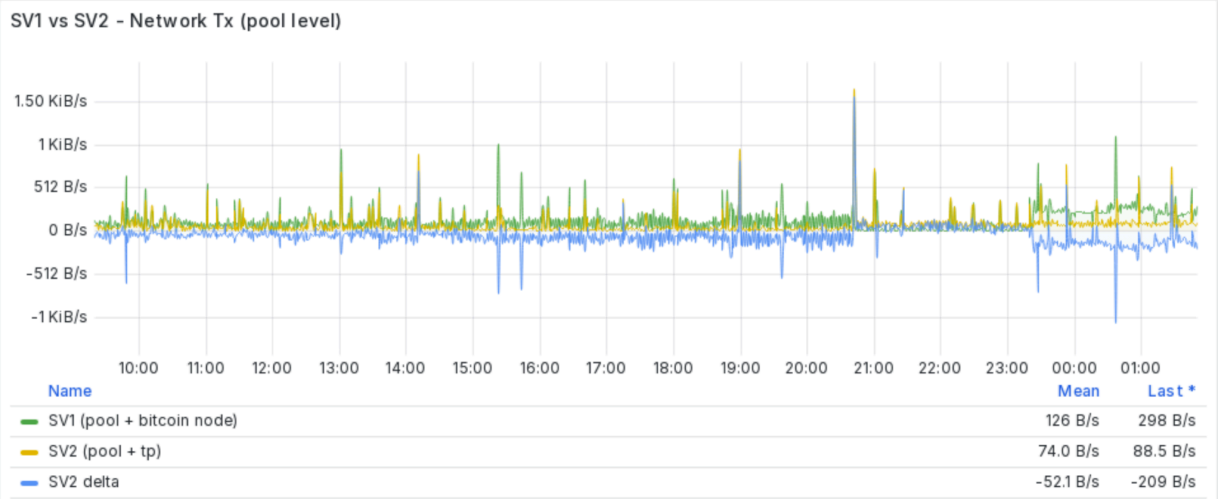


Results:

- **SV1** average bandwidth used: **66.6 B/s**
- **SV2** average bandwidth used: **26.0 B/s**
- SV2 delta: **-40.6 B/s (-61%)**

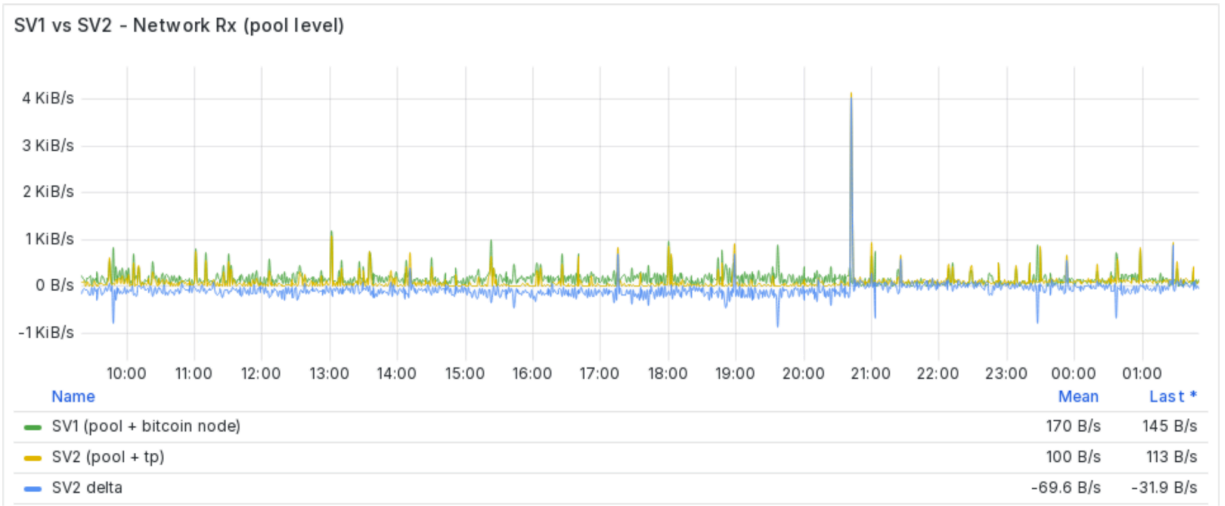
Analysis:
Since in this configuration (C) there is just the SV2 Translator proxy running on site, the bandwidth used for network receptions is just the one used to receive new jobs from the SV2 Pool. Given that the proxy aggregates downstream miners connections, the result is a **61%** enhancement in bandwidth usage. Differently from configuration (A), this setup is better suited for miners who could have some restrictions on bandwidth at a farm level, and cannot afford running a Bitcoin node in their mining farm.

- ***Bandwidth Usage - Pool Level***



Results:
- **SV1** average bandwidth used: **126 B/s**
- **SV2** average bandwidth used: **74.0 B/s**
- SV2 delta: **-52.1 B/s (-41.35%)**

Analysis:
The presence of the SV2 Translator proxy in the mining farm reduced the amount of bandwidth used by the SV2 Pool to manage downstreams. This benchmark has been done with 2 miners pointed to the SV1 setup and 2 miners pointed to the SV2 setup, so the benefit would probably be even higher for a larger number of machines. This will be one point of focus in a future benchmark.



Results:
- **SV1** average bandwidth used: **170 B/s**
- **SV2** average bandwidth used: **100 B/s**
- SV2 delta: **-69.6 B/s (-41%)**

Analysis:
The presence of the SV2 Translator proxy in the mining farm reduced the amount of bandwidth used by the SV2 Pool to receive shares from the mining farm. This benchmark has been done with 2 miners pointed to the SV1 setup and 2 miners pointed to the SV2 setup, so the benefit would probably be even higher for a larger number of machines. This will be one point of focus in a future benchmark.

Conclusions and Future Prospects ➡️SOON

Summary of Key Points:

The benchmarking results provide a detailed comparison between Configuration A and Configuration C, each tested under the same conditions. Key findings include:

1. **Share Submission and Acceptance**: Configuration A showed a higher acceptance ratio (100%) for shares submitted to the SV2 pool compared to Configuration C (99.9%), indicating the effectiveness of the SV2 Translator proxy and the local Job Declarator Client (JDC) in reducing latency and improving efficiency.
2. **Block Template Values**: Both configurations demonstrated minor differences in block template values on testnet4, with SV2 consistently showing slightly higher average values, reflecting its potential for better fee capture in a real-world scenario.
3. **Latency and Time Measurements**: Configuration A significantly reduced latency for job updates and new jobs based on a new prev-hash due to the local JDC, saving substantial time over a year. Configuration C also showed improvements, albeit less pronounced, due to the efficiency of SV2 message framing.
4. **Bandwidth Usage**: Configuration A, with a local Template Provider, increased bandwidth usage at the mining farm level but reduced it at the pool level. Configuration C, with only the SV2 Translator proxy, significantly reduced bandwidth usage both at the mining farm and pool levels, making it more suitable for environments with bandwidth constraints.

Future Improvements:

To enhance the benchmarking tool and provide more comprehensive insights, the following improvements and additional metrics could be considered:

1. **Incorporate ASIC Miners**: Future benchmarks should include ASIC miners to evaluate performance and efficiency under real-world conditions, as the current test was limited to CPU miners.

2. **Mainnet Testing**: Conducting benchmarks on the mainnet will provide more accurate data, particularly regarding transaction fees and block template values.
3. **Extended Time Frames**: Running benchmarks over longer periods will help capture variations and provide more robust data on performance and efficiency.
4. **Scalability Tests**: Assessing how each configuration scales with an increasing number of miners will help determine their suitability for larger mining operations.

By addressing these areas, future benchmarks will offer deeper insights, enabling more informed decisions for optimizing mining setups and improving overall efficiency.

Appendices

- PDF with complete benchmarks analyzed in this document : https://github.com/stratum-mining/benchmarking-tool/reports/

- Benchmarking tool main **repository**: https://github.com/stratum-mining/benchmarking-tool

- Benchmarking tool **discussion**: https://discord.com/channels/950687892169195530/1107964065936060467

- Benchmarking tool initial **documentation** and **requirements** definition: Stratum v2 (SRI) - Benchmarking Tool Documentation

- Benchmarking tool **system design**: Stratum v2 (SRI) - Benchmarking Tool System Design