MARA Utility Scale Mining (USM) Stratum V2 Benchmarking

Abstract

This document presents the results of a 10-day benchmarking experiment comparing the Stratum V1 (Sv1) and Stratum V2 (Sv2) protocols in a mining setting. Using a fleet of six S19k Pros, the study compared hashrate performance, latency, and network traffic between the two protocols, highlighting significant advantages for Sv2, including faster block propagation and lower stale share rates. The results demonstrate the superiority of Sv2 over Sv1 in several key areas, with implications for large-scale mining operations looking to optimize their technology stack.

Introduction

This report presents the results of a comprehensive benchmarking experiment comparing the performance of Stratum V1 (Sv1) and Stratum V2 (Sv2) protocols in a real-world mining scenario. The study aimed to evaluate the advantages and disadvantages of each protocol stack, with a focus on latency, network bandwidth usage, and mining efficiency.

The results are presented in this report, along with accompanying graphs and charts that provide a visual representation of the key findings. These graphs can be used to compare the performance of Sv1 and Sv2 protocols in different scenarios, and gain insights into their strengths and weaknesses. To directly inspect our data using Prometheus/Grafana, please see Restoring Data to Prometheus.

For those interested in learning more about the benchmarking tool, please visit the benchmarking-tool repository on GitHub. This repository includes the source code for the benchmarking tool, as well as documentation on how to run the experiment and initial analysis using cpu-miners.

Note that this report focuses on the results of our specific experiment, and does not provide a general guide on how to use the benchmarking tool or replicate the experiment. If you have any questions or would like to learn more about the experiment methodology, please reach out to gary.krause@mara.com.

Methodology

The Stratum V2 (Sv2) protocol team provides a suite of Docker containers that can be run to observe Sv2 vs it's predecessor, SV1. This suite provides metrics across several domains for each protocol and provides objective measurement from an engineering viewpoint. Researching how to OPTIMIZE MARA USM's engineering technology stack, we conducted a 10 day benchmarking experiment. A fleet of six S19k Pros were allocated to the experiment. Half

would mine with the Stratum v1 (Sv1) protocol and the half with Sv2 for the duration of the experiment. A VM was allocated for the benchmarking suite with 16 vCPUs, 24GB RAM and 750GB SSD storage; running Ubuntu 22.04 and Docker v27.0. The benchmarking tool git repository was at commit hash 7c2d8e04645395c9d7fe0c36a637f3a0ea451d66 (a diff showing specific configuration changes can be found in this repository) when the benchmark was ran. This is a few commits ahead of the 0.1.0 release. Improvements to the suite should be expected for any future testing as the tool is currently under active development.

Two configurations are provided for benchmarking. Configuration A allows for a miner to use Sv1 firmware to mine with a Sv2 pool while selecting their own block template using Job Declaration. Configuration C also allows Sv1 firmware mining using a Translator Proxy but does not use Job Declaration. Configuration A uses the full capabilities of Sv2 so we opted to utilize that in our benchmarking. Additionally, MARA USM team finds the modularity and optionality of these various configurations to be a positive development for pool mining technology. As defined on the website, there are also Configurations B and D, but these assume miner firmware to be Sv2 enabled. Future iterations of benchmarking may include these after firmware support is more readily available. Any interested parties are encouraged to run and review the benchmarking-tool while providing feedback and improvements to the repository.

During the experiment there are two instances where the whole suite had to be restarted due to technical issues. Aside from very visible skews in the graphs, we did not find these outages impacted the results of the testing.

Tool Suite

A list of Docker containers in the suite of tools used during the benchmark. More details can be found in the Sv2 benchmarking-tool repository. Configuration changes for each tool is also listed.

Builders Several builder images were used for creating artifacts for use in other containers.

- sv2-roles-builder
- log-server-builder
- sv2-custom-proxy-builder
- sv1-custom-proxy-builder
- pool-latency-calculator-builder

Bitcoin Nodes

- Template Provider Miner - The bitcoind node used by the miners to create block templates
- Template Provider Pool - The bitcoind node used by the pool to validate templates

- Stratum v1 Node - The bitcoind node used by the Sv1 pool for block template creation and broadcast

For the Template Provider (TP) nodes, a modified version of Bitcoin Core that enables the bitcoind daemon to utilize Sv2 binary messaging instead of using traditional JSON RPC. At the time of this writing (BH 863824), this change has not been merged into Bitcoin Core.

Pool The Pool role of Sv2. This pool does not have user account nor other features a normal public pool and just serves as a centralized coordinator for hashrate in this setup. For this benchmarking, the pool config (custom-configs/sri-roles/config-a/pool-config-a-docker-example.toml) was modified to mine to an appropriate `coinbase_outputs` and update the `pool_signature`.

Job Declaration Server The Job Declaration Server (JDS) role of Sv2. The JDS propagates blocks found by miners and is responsible for providing data to the Job Declaration Clients (JDC) so the JDC can create mining jobs for the mining devices. For this benchmarking, the configuration (custom-configs/sri-roles/config-a/jds-config-a-docker-example.toml) was updated with an appropriate `coinbase_outputs`.

Job Declaration Client The Job Declaration Client (JDC) role of Sv2. The JDC coordinates with the JDS and distributes mining jobs to mining devices. For this benchmarking, the `coinbase_outputs` and `pool_signature` were modified. This configuration for the JDC provides a solo-mining fall back in case of connectivity loss.

Translator The Sv2 Translator proxy provides a translation between mining devices that only speak Sv1 protocol. Mining devices allocated to Sv2 connected to this container and shares were proxied to the pool.

Proxies Several custom proxies were written for this benchmarking tool which allow for logs and metrics to be gathered for links in the network architecture. These proxies enabled measure of latency and bandwidth between the given roles. Source code for these proxies exist within the benchmarking-tool repository.

- BTC Node <-> JDC
- Translator <-> JDC
- Translator <-> miner
- BTC Node <-> Sv2 Pool
- Sv1 Pool <-> miner
- BTC Node <-> Sv1 Pool

Sv1 Pool Public Pool was used as the mining pool for the Sv1 stack. It is an open source software project that enables solo miners and charges no fee. Source code can be found here.

Metric and Logging Many of the running container facilitated measurements of the experiment and are listed below.

- monitor-traffic-dump - A tool for capturing network traffic.
- prometheus - an industry standard for metrics and alerting
- node-exporter - export machine (hardware/OS) metrics for consumption by prometheus
- cadvisor - analyzes resource usage and performance characteristics of running containers.
- grafana - an analytics and monitoring solution (graphs and charts)
- reporter - a tool for exporting a PDF of grafana dashboards
- renderer - headless browser rendering for grafana panels and dashboards
- pool-latency-calculator - Custom tooling to measure latency to a wide array of public pools so the average latency can be applied within the benchmarking framework
- loki - log aggregation system
- promtail - A log shipper for loki
- log-server - custom code to query loki for logs and bundle them

Results

This commentary will follow the PDF file SRI-Benchmarking-Tool-2.pdf and reference a graph(s) by titled name.

Shares and Blocks

Shares and blocks found are considered to ensure both protocol stacks are roughly equal when it comes to hashrate. With ~50k shares for both Sv1 and Sv2, this is a good confirmation that each protocol stack had roughly the same hashrate. Of note, there are 0 stale shares in the Stratum v2 stack, something that is particularly desireable for large scale mining operations. More analysis on stale shares can be found in Time to Get New Job and Block Propagation.

Block Template values

Since this benchmarking was conducted on testnet4 with little to no fees, there is a tiny amount of variance in block template values. This delta in value is assessed as non-impactful to the overall experiment.

Latency with Major Pools

A latency measure was calculated every minute against a list of public pools (in `benchmarking-tool/pools-latency-calculator/src/main.rs`) and averaged. A prometheus metric server then provided this latency to the suite of tools. This latency was then applied for connections between pool-side and miner-side containers to replicate “normal” network conditions.

Time to Get New Job and Block Propagation

This metric is particularly important to a mining operation at scale, as latency in new jobs arriving at miners results in hash wasted on stale block templates. Similarly, propagating a block to the network quickly improves chances to win a chain split race. With Sv2's Job Declaration, a miner can position a JDC closer (network-wise) to their ASICs which results in lower latency than connecting directly to a pool.

Our data clearly shows an advantage to using Sv2 as the average time for a new job is ~10x quicker with the Sv2 protocol. This delta can be an explanation for why Sv1 had stale shares submitted and Sv2 did not. Additionally, the block propagation time for Sv2 was ~40x quicker.

Average Time	Sv1	Sv2
New Job	115ms	**16ms**
New Job after New Block	102ms	**15ms**
Block Propagation	79ms	**1.5ms**

Prometheus gauges were used within the custom proxies to measure timing for these metrics. (`sv2-custom-proxy/src/main.rs` and `sv1-custom-proxy/src/main.rs`)

Network

Network bandwidth usage on a per second basis was captured and measured for all roles relevant to bitcoin mining. There are two defined "levels" to network reporting: pool level - which encompasses all roles assumed to be managed by the entity running the pool, and farm level - which encompasses all roles assumed to be managed by those managing the ASICs. The benchmark assumes that farm level daemons will be run within the same LAN as the mining devices to have optimal latency, so their network measurements are grouped together. The farm level group is comprised of the Translator Proxy, JDC, and Template Provider. The pool level daemons are assumed to be running in an internet accessible location. Pool level daemons are comprised of the Pool, Job Declaration Server, and Template Provider/Bitcoin Node.

For networking communication outside of assumed LANs:

- Template Provider/Bitcoin Node - p2p connection to other bitcoin nodes
- JDS connects with JDC
- Sv1 Pool connects with mining devices
- Translator Proxy connects with the Sv2 pool

Farm level network traffic for Sv2 has more bandwidth usage than Sv1 which is anticipated since more network-connected daemons are running. However, because job processing, block template generation, and block broadcast is done at the LAN/farm level, there is a significant latency improvement as seen in the

previous section. This bandwidth vs latency trade-off should be considered for anyone assessing Sv2 for their own use, but MARA finds this trade off desireable. Of note, the JDC provides fallback to other JDS's or solo mining so if the connection with the main pool's JDS goes down, the farm is unaffected.

At the pool level, we see slightly less network bandwidth usage for Sv2 than Sv1. Assuming pool infrastructure is stood up with high bandwidth as a priority, this impact should be negligible unless a pool is operating on a metered network consumption model.

Machine Metrics

Container level CPU, Memory and network usage were collected for each mining related container. Due to a couple of restarts of the benchmarking suite during the experiment, we have duplicate entries for multiple roles. MARA does not find this impacting to the analysis. Furthermore, given the difference in technology stacks for the implementations we drew no conclusions relevant to assessing the protocols from machine metrics. For interested parties, these metrics may serve as a reference point for technical considerations.

Conclusion

The results of this benchmarking experiment demonstrate the superiority of Stratum V2 (Sv2) over Stratum V1 (Sv1) in several key areas. With a significant advantage in terms of latency, Sv2's Job Declaration feature allowed miners to position their clients closer to their ASICs, resulting in faster block propagation and lower stale share rates. The benchmarking experiment also showed that Sv2 has higher network bandwidth usage than Sv1, particularly at the farm level. However, this increased bandwidth usage was offset by the significant latency improvement offered by Sv2's Job Declaration feature.

Overall, the results of this benchmarking experiment suggest that Stratum V2 is a more efficient and effective protocol for mining operations. The faster block propagation and lower stale share rates offered by Sv2 could result in significant improvements in mining efficiency and profitability. While the increased bandwidth usage at the farm level may be a consideration for some mining operations, it appears to be a trade-off worth making for the benefits offered by Sv2's Job Declaration feature. As the Bitcoin network continues to evolve, Stratum V2 is likely to become an increasingly important protocol for mining operations looking to optimize their performance and profitability.

Notes

Running this benchmark

If you intend to run this benchmark, the suite may be erroneous at first until all bitcoind nodes have finished syncing to testnet4.

Restoring Data to Prometheus

The file `20240920T150926Z-139666818b386e0e-prometheus-snapshot-benchmark.zip` is a ~14GB snapshot of the prometheus data from the 10 day benchmark and available via S3. The benchmarking-tool repository can be leveraged to reference the data for interested parties. Extract the zip file to a directory and update the docker compose file to only run prometheus and grafana. You'll also want ot modify the `prometheus` service definition in the docker compose file to use the extracted directory as the data directory.

```
-       - prometheus_data_config_a:/prometheus
+       - ./prom_snapshot_extracted_data/:/prometheus
```

`prom_snapshot_extracted_data` should have multiple folders within it when extracted that start with `01J9...`. When you start the prometheus container it will pick up this snapshot data and load it automatically. Once the containers are running you can navigate to the grafana web interface and set your time range for the graphs to be within `Wed Sep 11 01:55:00 UTC 2024` and `Fri Sep 20 19:00:00 UTC 2024`. The data should then be viewable and queryable within Grafana/Prometheus.

The following was run on an Apple M3 MBP using zsh.

```
 ~/repos/Sv2-Benchmark-Report/ shasum -a 256 ./20240920T150926Z-139666818b386e0e-prometheus-
9dacd3d5bc997627a370b24a12fa06cbc5acbc30bd4ddc94475b4b7d39cb0eb7  ./20240920T150926Z-139666
```

Once running, you can access the Grafana dashboard with appropriate time-frame using this link: http://localhost:3000/d/64nrElFmk/sri-benchmarking-tool?orgId=1&from=1726012500000&to=1726851600000

Misc

An article interview with the creator of the benchmarking tool: https://atlas21.com/progress-for-stratum-v2-the-stratum-benchmarking-tool-is-here/