KVU :: I/O Profile :: Max BW via KV Size

- Eva Winterschön

- 2025-07-22

📝 Summary

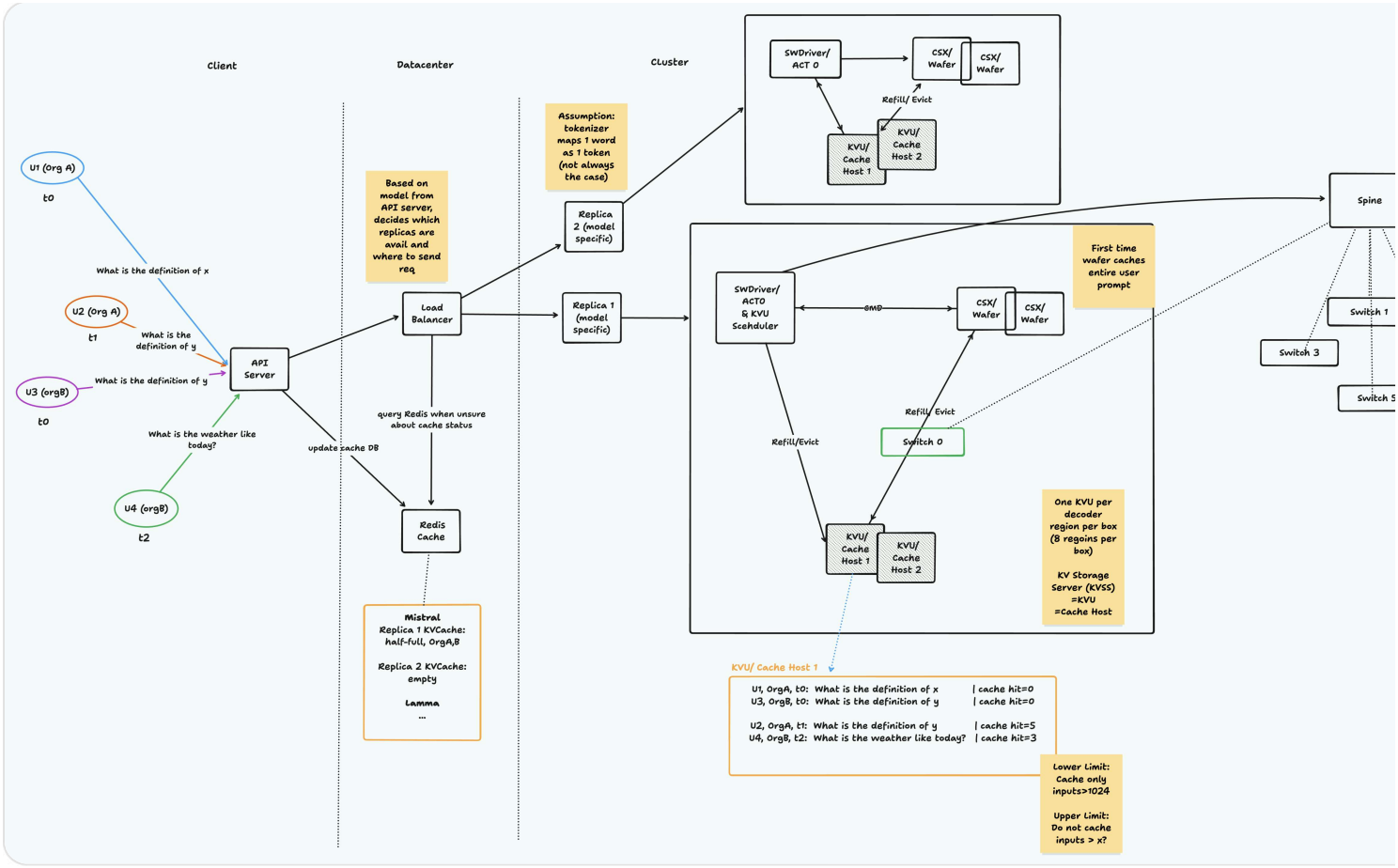
Identifying the KV Token Size in Bytes, on a per-model basis, in order to derive a calculation for **Max IO BW** at the network level, for each KVU node.

✅ Max I/O Functional Variables

In reference to starting conversations on the topic, we have the following variables.

```
A> KV size for some models (text and multi-modal), per token
B> C> CS-4 throughput
A * B == should be peak rate of IO each CS4 requires
```

🏛️ Existing TestFlow Architecture and Design



- QA Test Plan for Prompt Caching
- QA Test Strategy for Prompt Caching
- Release Roadmap (June 25, 2025)

⚙️ Test Flows Framework + Release Qualification Flow

About the Test Flow Framework

The TestFlow framework is a unified infrastructure for automating inference model testing at Cerebras. It provides a standardized foundation for different testing methodologies, reducing code duplication while enhancing configurability and observability. TestFlow serves as the backbone of Inference Release Qualification Flow, providing consistent test execution patterns, configuration management, and result reporting.

- Test Flow Framework Wiki

About the Release Qualification Flow

The Release Qualification Flow provides a comprehensive one-button solution for testing and qualifying inference models before release. This system transforms what was previously a complex, multi-day manual process into a streamlined, automated pipeline.

- Inference Release Qualification Flow
- Inference Release Qualification Flow :: End-2-End CLI Testing (manual equivalent to CI Testing)

Test Flow Directory Structure and Artifacts

Each TestFlow workspace follows a consistent structure, making it easy to understand and navigate:

```
flow_name/ # Test-specific workspace
├── ... # EvalRunner/BatchTest-specific file artifacts
├── flow_config.yaml # Complete configuration snapshot
├── flow_override_config.yaml # Non-default configuration values
├── run_metrics.json # Detailed execution metrics
├── eval_results.json/test_results.json # Processed evaluation results  <---- Where are these kept?
└── eval_logmessages.log/test_logmessages.log # Test-specific logs
```

Test Configuration Persistence

To enhance observability and ensure reproducibility, TestFlow automatically persists configuration information in each test's isolated working directory. This feature is critical for debugging, audit trails, and experiment reproduction. Reference: https://github.com/adama-cerebras/gorilla || https://github.com/ShishirPatil/gorilla

📈 Metrics Gathering Process

Test Flow Framework Data Results (reference)

- Single iteration: `eval_results.json` (see bold section above)
- Batch iteration: `test_results.json` (see bold section above)
- Finer-Grained metrics: `run_metrics.json`
- Observability Metadata-Tracking

Grafana Metrics + Dashboards

Dashboards are available here on AWS Grafana, with various options. When seeking data for this project, see the "Remaining Questions" section below.

🔒 Assessment Workflow

Accessing Shehnaz Islam's performance test results, analyze with the following process in mind.

a. Define standardized data model for querying *Q/A Test Plan* results in 1st normal form
b. Parse test plan results from past N time period, favoring past month of tests on initial scan
c. Identify potential data-types which may require conversion from *Q/A Test Plan* result sets into our equation variable
d. Using data model, calculate values for implementing the tri-partite function to determine Max bound per KVU node

Relevant Prompt Cache Test Fixtures for Recompute Speed

1. While inspecting code used for the test-runners, from *Shehnaz Islam's* python script, which is mentioned in

the TestFlow Directory Structure and Artifacts from earlier in this document, I've located the following sections which are relevant to our needs. @Reza @Lichen Liu

- Source: prompt_cache/test_prompt_cache_main.py
- Specific lines, referencing the logs workdir that I need to access, as well as several variables which we may be able to use for calculating the recompute speed variable, which would enable our 10x calculation for a complete equation. (lines #L1122-L1151)
- More specifically, sections of the test which calculate median tokens/sec on aggregates, given four separate tests using standardized cache invalidation modes, ***which could be informative to identifying our required Recompute Speed***. (lines #L2255-L2290)

1. Annotations on Multimodal Tests, as a secondary component to Reza's initial equation.

The following are from the same python test script, lines #L2299-L2300, as follows are an embedded link and Jira ticket reference.

> **Model-Dependent Features**:
> Features like multimodal, speculative decoding, and SWA are likely server/model configuration dependent.
>
> --- Multi-modality Checks ---
> TODO https://cerebras.atlassian.net/browse/SW-202756

Post-Assessment Workflow

Due to performance test result logs not being identified via expected means, which is not to say that they are not available, rather that I did locate sufficient data to use for metrics analysis. Instead, I've been analyzing Grafana dashboards for metrics on the **"Internal E2E Testing"** data.

E2E Testing - Traffic Patterns

On the screenshot below, with this dashboard we can see the E2E testing with a specific peak on traffic, likely during one of the performance load tests from which we would want to gather data. While this spike does not correlate to a similar spike in the mass of different models in the chart one row above, it's worth looking into for a query to the testing team.

Grafana as a Beneficial Resource

One benefit of using the grafana data is that queries can be run which facilitate our need to gather KV size for various models, and the graphs show that total requests/sec as well as tokens/sec are being tracked alongside the models' overall tokens/sec. At the same time, req/sec are being tracked on a per-replica basis.

Post Assessment Next-Steps

I propose that we identify the following requirements as next-steps:

a. **Engineering point of contact**: who is most familiar with the metrics going into Grafana during the E2E timestamp periods for the performance tests, as this is where our desired maximum values could identified.
b. **Access to Grafana's database backend**: so that I can query our metrics database. This should be sufficiently reproducible across test-runs without concerns for log files being purged or removed after test runs.
c. **Calculate our necessary Total KV Bytes per Token** *(on a per-model basis)*, as long as the necessary data is being tracked during the performance tests as it has been described in earlier sections of the document.

Future Analysis - Ongoing

Once the queries are generated, the resulting graphs can be added to our own dashboard which we may then use to track how these values are changing over time based on variable inputs (software changes, model changes, hardware changes, etc).

Grafana Dashboard for Throughput Analysis

📝 Communicated Research

Reza's Research on Text vs Visual/Image Models

BW requirements for Total KV units during T timeframe will be lower on visual/imagery models compared to text-only models.

Current Short Term Estimates for Cached vs Recompute

Based on Eugene's estimates:

- Cached-Prompt Perf Target: ~10x the recompute timing*
- Current: ~6x, probably still shippable

[*] Utilizing a cached prompt should be 10x faster than a non-cached recompute

- https://cerebras.atlassian.net/wiki/spaces/AIC/pages/4627660802/Prompt+Caching+-+Jul+18+2025

Forward Movement, Equation Planning

Reza's version adapted based on Lichen's notes

> *re: Generating speed is diff from Recompute Speed. Recompute is what we're saving. Therefore, Recompute gets the 10x value.*

⚠️ Remaining Questions

1. **Pending**: **${throughput_CS_pc_assisted}** requires knowing an accurate value of **${throughput_CS}** for recompute speed.
   - Logs for the performance tests are not readily searchable via Github, Confluence, Jenkins.
   - Let's possible get the metrics via Grafana's backend database (Prometheus, etc), which I can query.
2. **Grafana**: upon seeking metrics for "Tokens per Model Dashboard", it appears to be empty when seeking backwards till the start of June.
   - Was this dashboard ever populated correctly?
   - Should we care about it, and if so, who can fix it?

Equation Flow - Version 0.1

Recompute Speed == throughput_CS

1) If we know at what rate our CS-X can RECOMPUTE tokens (for all users combined): == ${throughput_CS}

Adjust 10x for Target Speed-up == throughput_CS_pc_assisted

2) We can apply 10x (based on target speed-up from Eugene), to arrive at the rate prompt caching-assisted token generation: == ${throughput_CS_pc_assisted}

Identify KV Size per Token via KVSS LMR Simulator

3) Identify KV sizes per token: ${token_kv_size}

- per Lichen: #3 refers to the bytes of a single token worth of KVCache
- Taking average "Per Token KV bytes", across all 8 nodes in the KVSS LMR Table, we get 34,560 bytes.
- Ref: "Lichen-KVSS-LMR-Table.ods calc sheet, same as the screenshot from Slack but as spreadsheet.

${token_kv_size} == 34560 bytes for AVG token size across 8 simulator nodes

---

```
INFO     root:inference_job_qor.py:772
KVSS LMR Table (By Node)
Node             Base Memory Bytes    Per Token KV Bytes      KVSS Count
---------------  -------------------  --------------------  ------------
net001-ax-sr04   128,849,018,880      39,456                          12
net001-ax-sr05   128,849,018,880      49,776                          12
net002-ax-sr04   107,374,182,400      35,136                          16
net002-ax-sr05   107,374,182,400      34,512                          16
net001-ax-sr03   96,636,764,160       30,480                           9
net002-ax-sr02   96,636,764,160       32,736                           9
net001-ax-sr02   96,636,764,160       32,064                           9
net002-ax-sr03   85,899,345,920       22,320                           8
```

Enabling a conclusion at peak usage:

MAX BW Prompt caching needs to supply the following equation, in bytes per CS-X node.

```
== ${token_kv_size} x ${throughput_CS_pc_assisted}
```

Current Equation, pending data on 'Recompute Speed'

```
== 34560 x ${throughput_CS_pc_assisted}
```

Equation Flow - Version 0.2

Adjustment is required to clarify that the equation uses a model dependent value structure.

Model Dependent Equation

`upperbound_BW` = `${model_N_per_token_kv_size} x ${throughput_CS_pc_assisted_model_N}`

Both numbers will change based on the model

The KV value changes according to the model which the simulator is using.

`== 34560 Bytes x ${throughput_CS_pc_assisted}` is not generic, it represents the KV size for that model's sim.

Model Size Impacts to Prompt-Cache Throughput Assist

When a larger model has larger `per_token_kv_size`, but would be slower computationally, it may potentially have lower `throughput_CS_pc_assisted` value.

Token/sec Optimization Potential

- llama3 1B == `~80k t/s` on a single system
- llama3 1B (Future Optimized) == `~300k t/s`
- llama 70b == `30k t/s` per node (using 9 replicas)
- Deepseek 671b == `15-18k t/s` per node (using 31 replicas)

Models Optimized using GQA

- llama 70b: `prompt_processing_tok/s` x `KV_size_per_tok_per_box` == `11 GB/s` (per system)
- llama 1B: `prompt_processing_tok/s` x `KV_size_per_tok_per_box` == `10 GB/s` (per system)

Equation Flow - Tokens/sec on Full-Replica

> Usually we process in the ballpark of 30k-40k tokens/s -- @Andrei Hagiescu

Therefore, we can provide the following analysis.

a. 40k tok/s max rate
b. 4KB token size from KVSS LMR (round-up on 3.5KB avg)
c. 10x Speed-Up target
d. 2x Amplify for hardware perf gen CS-3 vs CS-4

Max Throughput Values Calculated

At present we have obtained performance values using two observed methods.

Max Throughput Calculation - Replicas using 4KB Token == 3.2GB/s or 25 Gbps

The following values are based on information relayed by @Andrei Hagiescu to Reza

- 10x multiplier for Speed-Up value perf-target, via Eugene
- 2x multiplier for CS-3 vs CS-4 performance

```
40k tok/s x 4KB per tok x 10 x 2 == 3.2 GB/s or 25 Gbps
```

Max Throughput Calculation - Optimized using GQA == 880 GB/s or 7 Tbps

The following values are based on information relayed by @Rohan Gupta to Reza

- Basing on llama 70b == `11 GB/s` using GQA

CS-3 Calculation = 880 GB/s per System

```
11GB/s x 8 (MHA over GQA multiplier) x 10 (Speed-Up perf target) == 880 GB/s per system
```

CS-4 Calculation == 1,760 GB/s per System

We expect a 2x multiplier for performance on the CS-4 platform. Thus, our former CS-3 equation becomes:

```
11GB/s x 8 (MHA over GQA multiplier) x 10 (Speed-Up perf target) == 880 GB/s per system
```