STAC-ML™ Pack for myrtle.ai VOLLO™ (Rev C) with Silicom Artena (AMD VP1802 FPGA) on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

STAC-ML™ Markets (Inference) Benchmarks

Tacana Suite

Test date: 10 April, 2026

Release: v1, 24 April, 2026

STAC REPORT

These tests followed STAC benchmark specifications proposed or approved by the STAC Benchmark Council (see www.stacresearch.com). Be sure to check the version of any specification used in a report. Different versions may not yield results that can be compared to one another.

References

The following materials are accessible by qualified members of the STAC Benchmark Council. Contact info@STACresearch.com to request access.

[1] Specifications used for this benchmark: STAC-ML Markets (Inference) Benchmark Specifications, Rev F– https://docs.STACresearch.com/stac-ml-markets-inference-benchmark-specs-rev-f
[2] STAC Configuration Disclosure for this SUT*
[3] iPython Performance Notebook from this project (in HTML)*
[4] iPython Efficiency Notebook from this project (in HTML)*
[5] iPython Quality Notebook from this project (in HTML)*

* Available at https://stacresearch.com/MRTL260323 (Same URL as this report)

Summary

Myrtle.ai recently performed STAC-ML Markets (Inference) benchmark tests on a stack consisting of STAC-ML™ Pack for myrtle.ai VOLLO™ (Rev C) with Silicom Artena (AMD VP1802 FPGA) on Supermicro AS-2015CS-TNR. This report provides test results related to the performance, efficiency, and quality of the algorithms in the benchmark implementation. It also describes the solution that was tested and salient aspects of the test project.

STAC-ML Markets (Inference) is the technology benchmark standard for solutions that may be used to run inference on realtime market data. Designed by quants and technologists from some of the world's leading financial firms, STAC-ML Markets (Inference) reports the performance, resource efficiency, and quality of any technology stack capable of performing inference using the provided models. The specifications are agnostic to the architectures of the stack under test (SUT), inference engine, and underlying precision of the computations. This report highlights results from the sliding-window suite (code named Tacana). See the Overview later in this report for more background.

Benchmarks that use realistic workloads to measure business-relevant dimensions of performance help firms make sense of a wide variety of technology stacks. The STAC Benchmark Council[1] has created and maintained such benchmarks—STAC Benchmarks™— for over a decade for other business-critical analytic workloads such as derivatives valuation, strategy backtesting, and enterprise tick analytics.[2]

These tests were performed using a Myrtle.ai-authored STAC Pack. The STAC Pack used the VOLLO SDK 27.0.0 that includes the vollo-rt, vollo-torch and vollo-compiler libraries. The benchmark driver is a Rust program compiled with Rust 1.94.1. For more information on the implementation and the system on which it was tested, see Section 3.

In all, the STAC-ML Markets (Inference) specifications deliver scores of test results, which are detailed in this report and accompanying notebooks. Myrtle.ai would like to point out the following:

- Compared with all other public submissions for the Tacana suite, this SUT demonstrated the following:
  - For all three models, in all NMI configurations, the 99p latency was lower than any previously reported for that model regardless of NMI

- This SUT also demonstrates the following:
  - First sub 2µs 99p latency for LSTM_A
  - First sub 3µs 99p latency for LSTM_B
  - First sub 8µs 99p latency for LSTM_C

- Compared with the previous SUT submitted by myrtle.ai (MRTL230426) using VOLLO SDK v0.2.0
  - For LSTM_A (the smallest model:
    - The 99p latencies were:
      - 2.7x lower with 1 NMI (1.89µs vs. 5.07µs)
      - 2.6x lower with 2 NMI (1.98µs vs 5.07µs)
      - 2.0x lower with 4 NMI (2.58µs vs 5.08µs)
      - 2.0x lower with 8 NMI (3.02µs vs 5.97µs)
    - The 99p error was 44% lower (0.00498 vs. 0.00889)
  - For LSTM_B (the medium model):
    - The 99p latencies were:
      - 3.0x lower with 1 NMI (2.30µs vs 6.89µs)
      - 2.7x lower with 2 NMI (2.55µs vs 6.77µs)
    - The 99p error was 55% lower (0.00573 vs. 0.0127)
  - For LSTM_C (the largest model):

[1] www.STACresearch.com/council

[2] See www.STACresearch.com/a2, www.STACresearch.com/a3, and www.STACresearch.com/m3, respectively.

- The 99p latencies were:
  - 4.0x lower with 1 NMI (7.83µs vs. 31.0µs)
- The 99p error was 63% lower (0.00870 vs. 0.0237)

LSTM_A Report Card

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

LSTM_A: Report Card

Error Benchmark T.LSTM_A.ERR.v1 (99th Percentile): 0.00498
Comparable to SUTs with T.LSTM_A.ERR.v1 (99th Percentile) > 0.00166

NMI	1	2	4	8
T.LSTM_A.[NMI].TPUT.v1 **Total Throughput** **Inferences / sec**	52,453	104,312	156,943	200,946
T.LSTM_A.[NMI].INST_TPUT.v1 (Minimum) **Worst-Case Instance Throughput** **Inferences / sec**	52,370	52,144	39,231	25,106
T.LSTM_A.[NMI].LAT.v1 **99th Percentile Latency** **ms**	0.00189	0.00198	0.00258	0.00302
T.LSTM_A.[NMI].LAT.v1 **Max Latency** **ms**	0.0339	0.0238	0.0282	0.0302
T.LSTM_A.[NMI].ENERG_EFF.v1 **Energy Efficiency** **Inferences / sec / kW**	185,347	357,738	517,509	645,774
T.LSTM_A.[NMI].SPACE_EFF.v1 **Space Efficiency** **Inferences / sec / ft³**	14,660	29,154	43,863	56,162

Source: STAC
www.STACresearch.com

LSTM_B Report Card

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

LSTM_B: Report Card

Error Benchmark T.LSTM_B.ERR.v1 (99th Percentile): 0.00573
Comparable to SUTs with T.LSTM_B.ERR.v1 (99th Percentile) > 0.00191

NMI	1	2
T.LSTM_B.[NMI].TPUT.v1 **Total Throughput** **Inferences / sec**	12,170	15,693
T.LSTM_B.[NMI].INST_TPUT.v1 (Minimum) **Worst-Case Instance Throughput** **Inferences / sec**	12,168	7,846
T.LSTM_B.[NMI].LAT.v1 **99th Percentile Latency** **ms**	0.00230	0.00255
T.LSTM_B.[NMI].LAT.v1 **Max Latency** **ms**	0.0291	0.0182
T.LSTM_B.[NMI].ENERG_EFF.v1 **Energy Efficiency** **Inferences / sec / kW**	41,258	51,665
T.LSTM_B.[NMI].SPACE_EFF.v1 **Space Efficiency** **Inferences / sec / ft³**	3,401	4,386

Source: STAC
www.STACresearch.com

LSTM_C Report Card

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

LSTM_C: Report Card

Error Benchmark T.LSTM_C.ERR.v1 (99th Percentile): 0.00870
Comparable to SUTs with T.LSTM_C.ERR.v1 (99th Percentile) > 0.00290

NMI	1
T.LSTM_C.[NMI].TPUT.v1 **Total Throughput** **Inferences / sec**	737
T.LSTM_C.[NMI].INST_TPUT.v1 (Minimum) **Worst-Case Instance Throughput** **Inferences / sec**	737
T.LSTM_C.[NMI].LAT.v1 **99th Percentile Latency** **ms**	0.00783
T.LSTM_C.[NMI].LAT.v1 **Max Latency** **ms**	0.0212
T.LSTM_C.[NMI].ENERG_EFF.v1 **Energy Efficiency** **Inferences / sec / kW**	2,363
T.LSTM_C.[NMI].SPACE_EFF.v1 **Space Efficiency** **Inferences / sec / ft³**	206

Source: STAC
www.STACresearch.com

Vendor Commentary

Myrtle.ai provided the following comments:

> *The audit was performed using myrtle.ai's VOLLO accelerator running on a Silicom Artena FPGA accelerator card featuring an AMD VP1802 FPGA. This card was chosen due to its large amount of on-chip memory and DSPs, which enable VOLLO to run a wide range of AI models on a single device.*
>
> *By using VOLLO, the implementation is able to achieve lower latencies than all previous submissions without requiring low-level hardware-specific code.*
>
> *All models were compiled directly from PyTorch using myrtle.ai's VOLLO software. All the latencies achieved include the transfer of data to and from the accelerator over a PCIe bus.*
> *A single VOLLO firmware binary was used on the VP1802 for all tests.*

The stack under test

Any stack under test SUT to be assessed by STAC-ML Markets (Inference) benchmarks has two layers:

- An implementation of the benchmarks. This is software and/or firmware that implements the Workloads and other logic required by the benchmark specifications. For example, this could be a software program written in C++, VHDL code for an FPGA, an Inference Engine, scripts, etc. It is delivered with documentation in a "STAC Pack".
- Other hardware and software required to carry out the operations defined by the benchmark specification, including processors, accelerators, memory, operating system, compiler, math libraries, etc.

The next two sections discuss each layer in turn.

Understanding this implementation of STAC-ML Markets (Inference)

This project used a STAC-ML Markets (Inference) Implementation developed by Myrtle.ai, called the STAC-ML Markets (Inference) Pack for Myrtle.ai VOLLO (Rev C). The source code is available to qualified STAC subscribers. Myrtle.ai provided the following description:

> *The implementation uses the VOLLO Compiler to compile PyTorch models to VOLLO programs. The pre-defined ONNX models provided by the STAC Benchmark are rewritten as PyTorch models that maintain a sliding window of input as state between inferences. The rest of the compilation is automatic.*
>
> *The inference driver, written in Rust, uses the VOLLO Runtime C API to queue inference requests to the VOLLO accelerator, and then polls until the inference results have been returned from the accelerator. A single CPU thread is used across all workloads.*
>
> *NMIs >1 are handled using VOLLO's support for multi-model programs. In the implementation, each model instance is assigned its own subset of VOLLO cores on the accelerator at compile time and the Runtime selects between them.*

Understanding the rest of the SUT

The implementation relied on the following other key components in this project:

- VOLLO SDK 27.0.0
  - vollo-rt, vollo-torch, vollo-compiler libraries
- rustc 1.94.1, ONNX 1.17.0, torch 2.5.1
- Ubuntu 22.04.5 LTS
- Silicom Artena PCIe card FBAP4@VP18-2L0S!E[3]
  - AMD VP1802 FPGA
- Supermicro AS-2015CS-TNR Server
  - 1 x AMD EPYC 9575F Processor
  - 12 x 16GiB DDR5 DIMMs @ 6000MT/s (192GiB Total)

A detailed STAC Configuration Disclosure [2] for the SUT in this report is available to qualified STAC subscribers. That document provides the exact product version numbers, detailed tuning options, and other important information. Additional configuration details such as a sosreport may also be available, depending on the SUT platform.

The SUT server was configured to mitigate the full range of Spectre/Meltdown threats.

Myrtle.ai provided the following information about its products used in this SUT:

- *Since its first Tacana audit three years ago, VOLLO has been used by multiple STAC members for hundreds of thousands of hours in production trading. Continual improvements in both performance and efficiency have been delivered to all subscribers during that time. This report confirms those improvements and demonstrates the very low latencies that FPGAs can achieve.*
- *VOLLO was designed to be easy for ML developers to use for their models, without the need for FPGA expertise or tools. Models written in PyTorch, TensorFlow or ONNX can be compiled by VOLLO. These audit results were achieved with that standard flow, requiring no special skills.*
- *VOLLO achieves this high performance by running on hardware in the form of FPGAs, or Adaptable SoCs. Despite this, models may be switched very rapidly to facilitate the use cases often required by financial trading houses.*
- *VOLLO performance can readily be evaluated for STAC members' own models without the need to purchase hardware or licenses. This can be achieved using the Sandbox or downloadable Virtual Machine, accessible through vollo.myrtle.ai.*
- *For those who have in-house FPGA expertise and tools, VOLLO is also available as an FPGA netlist, enabling higher levels of integration.*

AMD provided the following information about its products used in this SUT:

> *The AMD Versal Premium VP1802 is based on 7nm Versal architecture that combines adaptable programmable logic with hardened connectivity providing high throughput and low latency, which proves ideal for trading applications. The device features PCIe Gen 5 along with a large FPGA fabric allowing for a fully programmable platform to implement customized inference and trading logic, making it ideal for myrtle.ai's inference application for use in very low latency trading applications.*

Supermicro provided the following information about its products used in this SUT:

> *Supermicro offers the industry's broadest portfolio of AMD-based servers through its Data Center Building Block Solutions (DCBBS) architecture, enabling organizations to tailor systems precisely to their compute, acceleration, and storage requirements.*

[3] The board used in this test was an engineering sample of the FBAP4@VP18-2L0SG, the generic and generally available version of the card codenamed ***Artena***. Silicom asserts that the engineering sample is "identical in form, fit and function" to the GA card, has "the same bill of material", and any differences only affect the PCB design "for manufacturability".

The system used for this benchmark, the Supermicro AS -2015CS-TNR, is a 2U modular rack server engineered for high-performance, latency-sensitive workloads. Powered by AMD EPYC 9005 Series processors, the platform delivers high single-core performance and extreme core density required by modern capital-markets infrastructure. This system also supports high-speed DDR5 memory, PCIe 5.0 I/O, and flexible acceleration options.

This configuration demonstrates how a modular platform can be optimized for real time ML inference using a combination of high-frequency CPU resources and deterministic FPGA acceleration, directly addressing the consistency requirements of STAC-ML workloads.

For more information, including detailed datasheets and system configurations, you can visit the Supermicro H13 Solutions page

Silicom provided the following information about its products used in this SUT:

The Silicom Artena Ultra-high-capacity FBAP4@VP18 series FPGA accelerator card has the following features:

- *Powerful AMD Versal Premium VP1802 Adaptable SoC.*
- *PCIe Gen5 2×8 for high-throughput and low-latency.*
- *Flexible power/thermal design with single-slot and dual-slot passive cooling option, allowing high resource utilization in dense trading server deployments.*
- *Designed for ultra-low-latency execution of custom logic, including financial feed-handlers, risk checks, and compact AI/ML inference models for trading strategies.*
- *Supports myrtle.ai VOLLO inference engine, enabling FPGA-based inference with deterministic latency for signal generation and decisioning.*

Project contributors and their roles

The following firms contributed to this project:

- Myrtle.ai
- AMD
- Supermicro
- Silicom
- STAC

The Project Participants had the following responsibilities:

- Myrtle.ai wrote the STAC Pack in accordance with the benchmark specifications, configured the hardware and software to be tested, provided the test lab, and sponsored the report
- Supermicro provided the server
- Silicom provided the PCIe accelerator card
- STAC edited the final STAC Report and STAC Configuration Disclosure, based on information provided by the sponsor(s).

Contacts

- Myrtle.ai: vollo@myrtle.ai

- AMD: dc_inquiries@amd.com
- Supermicro: financialservices@supermicro.com
- Silicom: contactus@silicom.dk
- STAC: info@STACresearch.com

Overview of the STAC-ML Markets (Inference) benchmark specifications

Business context

Financial firms sometimes develop neural network models to analyze time series of market data. They then use those models to run inference on incoming realtime market data. Such inference is latency sensitive and usually occurs in datacenters co-located with exchanges.

There are several use cases for co-located inference, such as valuations used in high frequency market making, price predictions for short- and medium-term position taking, creating price bands for pre- and post-trade risk checks, automated hedging and position reduction, and brokers generating fair-value prices that are used to make firm prices to their customers.

The benchmarks in this project are the first set of STAC-ML Markets benchmarks. They address a use case in which a solution performs inference using one or more instances of the same model, each operating on different streams of input data. For example, a firm might apply a single equities prediction model to multiple portfolios with different constituent equities.

Scope

For a given SUT, the objective is to measure the upper-bound performance of inference only. While end-to-end inference performance in the real world also depends on tasks such as data ingest, parsing, feature creation, and sometimes online retraining, the working group responsible for developing the benchmark decided that it would be most helpful to study the performance of these tasks separately.[4]

Some use cases call for minimum latency, while others call for maximum throughput or power/space-efficiency at an acceptable latency. A single solution may be capable of providing different tradeoffs depending on its configuration. For a given configuration, the goal of the benchmarks is to measure the upper-bound of performance and efficiency, thus elucidating the theoretical limits that a solution involving more functions could provide in the real world.

In order to satisfy the requirement to measure upper-bound performance, the workload is consumption driven. That is, rather than being forced to react to an asynchronous event stream (as in other realtime STAC Benchmarks), the SUT accepts each new input set as it completes the last. This eliminates queuing of inbound data from the latency measurement.

Benchmark construction

Models

For this use case, the STAC-ML Working Group chose a set of Stacked LSTM models that take a window consisting of N timesteps of M features and predict a single value per window. Similar to algorithms in other STAC Benchmarks, the models do not represent a production-ready strategy but were chosen by the Working Group because they induce workloads similar to those of real-world models.

[4] The performance characteristics of some tasks, such as low-latency market data parsing, are already well understood. See www.STACresearch.com/m1 for benchmarks the STAC Benchmark Council established in this domain long ago.

The current benchmark suite includes three LSTM Models, which are variations of the Stacked LSTM mentioned above, each with a unique combination of features, timesteps, layers, and units/layer:

- **Number of timesteps:** The amount of history affects the amount of data transferred to the inference engine. Since LSTM inference is sequential, the number of timesteps has a direct impact on inference latency.

- **Number of features:** This number also affects the amount of data transferred and has some effect on model size. However, for LSTM models, the number of features only affects the complexity of the initial LSTM layer. Increasing the number of features and the complexity of the model (as measured by the following two metrics) would likely go hand-in-hand in a real-world application.

- **Layers:** The number of LSTM layers directly impacts the model size and given the sequential evaluation of layers, has an equally direct impact on inference latency.

- **Units/Layer:** The number of units per layer also directly impacts model size, even though an implementation can parallelize the computation of a layer.

The three models are LSTM_A, LSTM_B, and LSTM_C. Relative to LSTM_A, LSTM_B is roughly 6 times the size and LSTM_C is roughly 2 orders of magnitude bigger. As the size of the model grows, features, timesteps, layers and units/layer either grow or remain the same.

For each LSTM Model there is a corresponding LSTM Null Model designed to elucidate the platform and communication overhead of the system. An LSTM Null Model performs a bare minimum of computation. Each LSTM Null Model requires the same number and shape of inputs as its corresponding LSTM Model and produces the same number and shape of outputs.

Understanding the STAC-ML Markets (Inference) Benchmark Suites

The STAC-ML Markets (Inference) Benchmark Specifications comprise 2 separate *suites*: Sumaco and Tacana. These two suites are identical in many respects. The suites differ primarily in how implementations handle data and the rules regarding the re-use of computations. Data generation, benchmark models, and performance, quality, and efficiency metrics are identical between the two suites.

In the benchmark specifications, the SUT performs inference on rectangular (Timesteps x Features) arrays (windows) of data. In the Sumaco suite, each full window of contiguous data is selected randomly from a larger data array and transmitted in whole to the Inference Engine. This suite models a use case where an external event invokes an inference based on the most recent history prior to the event. The Sumaco suite does not allow any pre-computation or reuse of computations from window to window. This inference approach is typically supported in generic Inference Engines (e.g., ONNX, TensorFlow) for evaluating the LSTM models that the benchmark specifications define.

In the Tacana suite, inference is performed on a sliding window. Each inference operation inserts the newest randomly-selected (1 x Features) vector into the window while removing the least recent (1 x Features) vector from the window prior to inference on the entire window. This suite models a use case where inference is performed on every tick or bar of market data using a fixed-length history of the most recent input data. In the Tacana suite, the SUT may choose to only transmit the most recent Timestep of data to the Inference Engine–reducing data overhead for SUTs based on external accelerators—and manage the sliding windows in the Inference Engine. The SUT can also re-use computations based on any of the data that remains in the window. SUTs that implement the Tacana suite will likely use custom Inference Engines.

Workloads

A Workload is the combination of a Model and Number of Model Instances (NMI) simultaneously performing inference. While the benchmark requires every SUT to execute each of the Models, the SUT provider chooses the NMI.

SUT providers will often benchmark multiple Workloads for each Model to demonstrate interesting tradeoffs in latency, throughput, and efficiency. For example, CPUs may be able to reduce latency by parallelizing small NMI, or FPGAs may move from loop-unrolled to iterative designs as the NMI increases.

Test framework

At a high level, generating STAC-ML Markets (Inference) benchmark results requires four things:

1. An Implementation. This is the logic that carries out the algorithms and measurements specified by STAC-ML Markets (Inference). It is described in Section 3.1.
2. Something on which to run the Implementation (whatever hardware and software is required). It is described in Section 3.2.
3. The STAC-ML Markets (Inference) Orchestration Scripts. These scripts direct the Implementation through all of the performance and quality test sequences required by the benchmark specifications, telling it at each step what algorithm to run, what problem size to use, etc.
4. A general-purpose processor on which to run the Orchestration Scripts.

#1 and #2, taken together, form the SUT, as shown in Figure 1. The general-purpose processor in #4 may or may not be part of the SUT. The SUT may also host the Orchestration Scripts (which occupy a tiny amount of memory), or those scripts can run remotely.

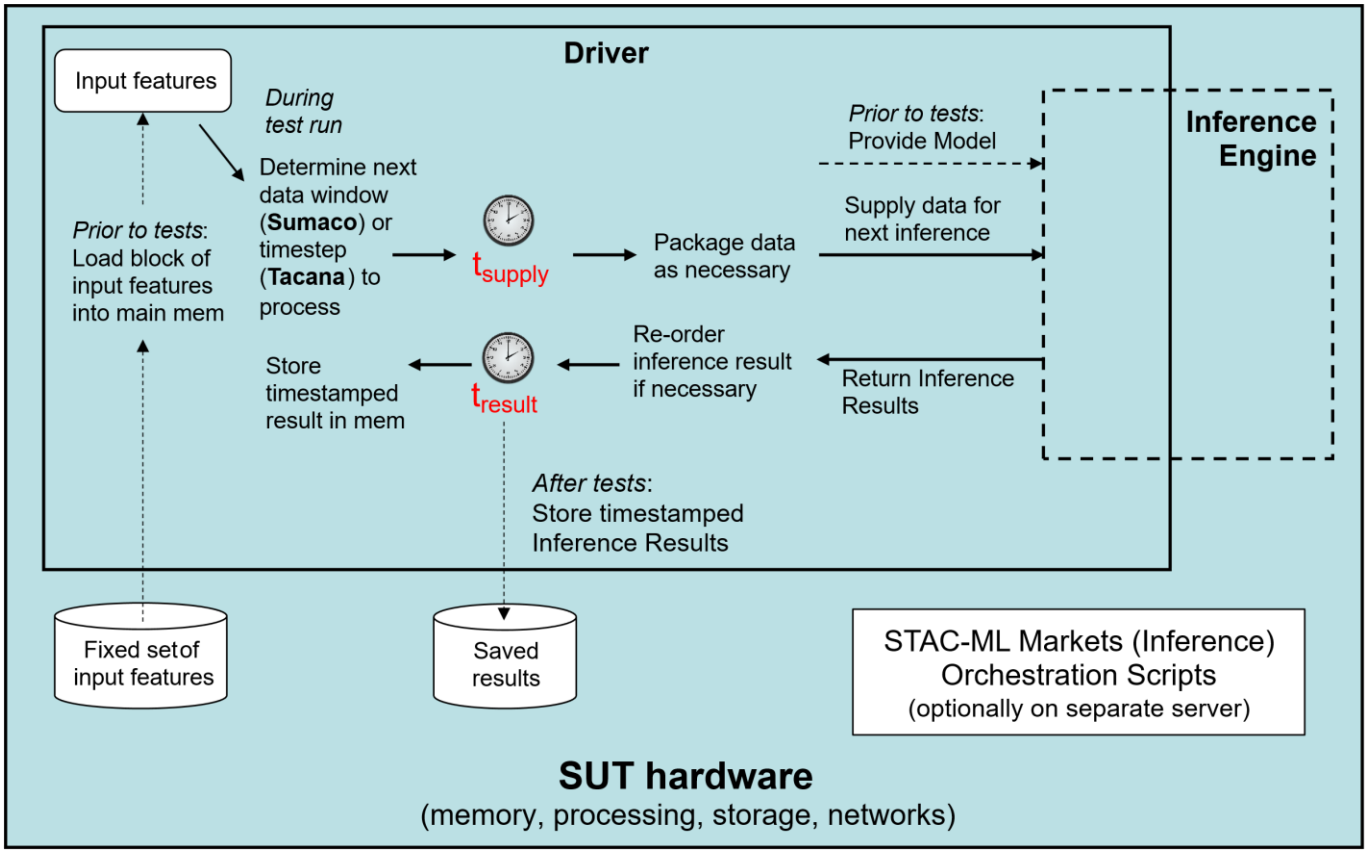


Figure 1 – High-level benchmark diagram

The Driver is the application that drives the benchmark operations. The Driver runs on a general-purpose CPU. The Inference Engine implementation may be completely contained within or linked with the Driver or distributed across other processes on the host CPU or accelerators. If the SUT uses one or more accelerators, the Driver calls out to the accelerator(s).

The rest of the Driver is benchmark-specific code written by the Implementation Provider that takes instruction from the Orchestration Scripts. STAC checks Drivers for conformance to specifications through code inspection and "black box" validation.

During test runs, the Driver provides a unique stream of features to each Model Instance (selected from the same in-memory store of input features). An Implementation Provider may implement a Model Instance in one of two ways: 1) as a single copy of the defined model, with inferences occurring serially, or 2) as multiple copies of the defined model, inferring in parallel or in a pipelined fashion. In either mode inferences may be processed singly or in batches. For a given workload on a given SUT, all Model Instances have the same construction.

The Driver locates either a full window of data (Sumaco) or a single timestep of data (Tacana) for each Model Instance, using a random selection scheme. As soon as it locates the data, the Driver obtains a timestamp (t_{supply}), which it stores in memory.

The Driver then does whatever is necessary to enable the Inference Engine to operate on the full data window (Sumaco) or sliding data window formed by the incoming timestep (Tacana). The Inference Engine always requires a complete window of features to perform inference. In the Tacana suite the Driver author is free to transfer the required data in whatever way is most efficient for the architecture (e.g., transfer a given timestep only once vs. transfer it multiple times as part of each window that includes it).

Once the Inference Engine returns the Inference Result, the Driver ensures proper ordering (which may require some work, as multiple inferences may be in-flight simultaneously), then obtains a second timestamp (t_{result}), which it stores in memory.

After a test run, the Driver persists the inferences and timestamps to a file for analysis.

Metrics

For a given SUT, the objective is to measure inference in isolation. End-to-end inference performance in the real world also depends on tasks such as data ingest, parsing, feature creation, and sometimes online retraining, but the Working Group believed it is most helpful to study those tasks separately.

The performance metrics of interest are latency and throughput of inference when various numbers of Model Instances are running on the SUT. As resources in co-located datacenters come at a premium, power and space efficiency metrics are also important. The quality of inference results is also of key interest.

Latency, throughput, and efficiency are only reported for the Inferring Period, which is a fixed amount of time in a test run following a vendor-specified warm-up time.

Latency

Inference latency is the elapsed time to produce a single inference result ($t_{result} - t_{supply}$). Note that this latency definition does *not* include the time to select the inference data or store the results in memory, which is considered external to the work to be measured. However, latency does include the time it takes to transfer data to and from the inference engine, including transfer time to any accelerators the SUT uses. Also, since inference results are required to be returned in the order of the triggering timestep, latency for parallel implementations may include waiting for the inference results of previously provided timesteps. Latency is only reported for inferences that take place after a SUT-specfic warmup time is complete.

Software timestamps have two potential sources of latency-measurement error:

1) Host clock error. The frequency error and timer resolution of the host clock must combine to less than the least significant digit reported for latency measurements. The Configuration Disclosure [2] provides details on the timer call used in the source code and the underlying OS timer used for latency measurements.

2) Application-level error. Often the largest errors in software latency measurement are due to delays between the moment the application code asks for a timestamp and the moment the timestamp is assigned to an event. These range from small delays due to the timestamping instructions and much larger delays caused by task switching. However, in STAC-ML Markets (Inference), the call to acquire the timestamp for t_{supply} must be completed before any of the measured operation begins, which means that any timestamp-assignment delays can only increase reported latency. Likewise, the call to acquire the timestamp for t_{result} cannot begin until the entirety of the measured operation is completed, which means that any delays between assignment and return of the timestamp also serve to increase reported latency. The positioning of the timestamp calls is verified by code review. In other words, the reported latencies are conservative measurements.

Throughput

Throughput is the number of inferences returned during the Inferring Period divided by the duration of the Inferring Period, reported as inferences per second. Two throughput metrics are of interest:

- Total Throughput is the throughput of the entire SUT, averaged across all test runs.
- For a given Workload, the worst-case Instance Throughput is the lowest Throughput exhibited by any instance in any test run.

Error

The measure of error is the absolute difference between the SUT's inference values and the corresponding inference values obtained by the Quality Reference implementation. The test harness reports statistics on these measurements across all Model Instances and test runs for a given Workload to derive a single Error metric. The specifications do not impose a minimum standard for Error. See "SUT Flexibility", below, for details.

Efficiency

For SUTs not hosted in a public cloud, there are two types of efficiency metrics: energy and space. The benchmark defines ways to measure the energy and space consumed by the SUT, as well as metrics that relate that consumption to the total throughput (i.e., the SUT's efficiency). For SUTs hosted in a public cloud, the dollar cost of the resources in the SUT during a test replaces energy and space.

SUT Flexibility

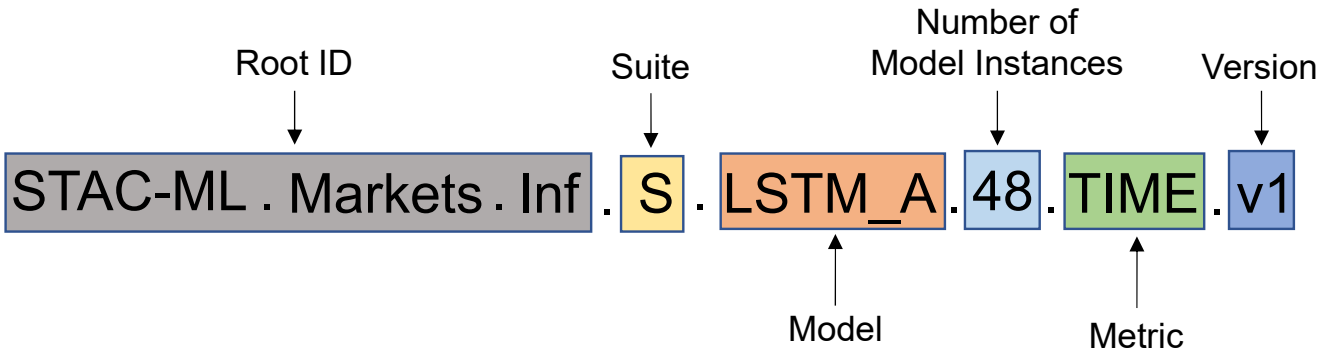
STAC-ML Markets (Inference) provides the SUT provider with great latitude in choosing what to test:

- The specs place no constraints on the architecture of the SUT, except that it must be controllable by the Orchestration Scripts, maintain the inputs and outputs in memory during tests, and use persistent media to read input and write output data (outside of timing windows).
- The SUT Provider choses whether to run the Sumaco or Tacana suite.
- A SUT is allowed to host any Number of Model Instances (NMI). A single Driver instance can control all Model instances or each Model Instance can have its own Driver. For a given Workload, all Driver instances must use the same code.
- A SUT provider is free to test any NMI values it believes demonstrate the SUT's range of capability, is free to use different NMI ranges for each Model and is free to use different configurations for each Workload, subject to the requirement that the Model precision must be the same for all Workloads.
- The SUT provider is free to optimize a SUT for a given set of goals, such as demonstrating minimum latency or revealing the throughput and latency tradeoffs for multiple Workloads.
- The SUT provider is free to test with whatever Model precision they wish. However, different precisions entail different amounts of work for the SUT. Therefore, we group SUTs into "weight classes" when comparing their

performance or efficiency. These classes are based on the Error they exhibit when tested (allowing some wiggle room for numerical variation and randomness).

Interpreting and comparing results

STAC-ML Markets (Inference) is a suite of benchmarks. The tables and charts in this report identify each benchmark unambiguously, in a way similar to this:



In charts, the ID is sometimes decomposed, with part of it in the chart title or labels. Most benchmarks in this report include a field for the Number of Model Instances, which is sometimes shown as a parameter [NMI] in tables. Each individual STAC Benchmark specification has its own version number. Versioning individual specs enables the reader to compare a discrete result from this SUT to the corresponding result from another SUT. When making comparisons, be sure that the identifiers match exactly, including the benchmark suite. Otherwise, the benchmark results cannot be fairly compared.

For one SUT to claim superior latency, throughput, or efficiency over another, the ratio of their Error to that of the other SUT must be less than three. Superiority claims are not transitive. It is possible for SUT A to be superior to SUT B, SUT B to be superior to SUT C, but SUT A and SUT C to be incomparable in terms of Error.

Specification particulars

Version

These benchmark specifications were developed by the STAC-ML Working Group of the STAC Benchmark Council. This project followed the STAC-ML Markets (Inference) benchmark specifications workbook Rev F. Qualified members of the STAC Benchmark Council can access these specifications and accompanying STAC Test Harness Software, as well as request vendor STAC Packs at www.STACresearch.com/ml.

Note that the revision letter of the workbook is irrelevant to whether these results can be compared to other STAC-ML Markets (Inference) results. Only the ID for each benchmark (including its version number) can be used to determine results compatibility. See Section “Interpreting and comparing results”, above, for details.

Limitations and clarifications

- The SUT creates and records the timestamps using methods that meet the benchmark specifications. While a code review ensures that the Implementation adheres to the “Timestamps Method” policy in the Benchmark Specifications [1], STAC does not empirically verify that the SUT meets the accuracy requirements for timestamps.

- In the Tacana suite, subsequent inferences share much of the data window, so the errors of inferences occurring close to each other may not be independent. The fewer inferences a SUT does during each run, the more likely it is for this to impact the reliability of error estimates. Sumaco inferences should be independent for practical purposes.

- Since STAC does not limit the quantization techniques that may be employed by the SUT, the specifications do not define a pass/fail metric for correctness. STAC's code review of the STAC Pack focuses on conformance to timing and reporting requirements, not on the correctness of the Inference Engine. Readers may evaluate the suitability of the SUT for their applications based on whether the reported errors are acceptable and consistent with the employed quantization techniques. The specifications base the Error metric on absolute error, but trained production models may use other error metrics.

- The spirit of the specifications is that the Null Model associated with each LSTM Model should be configured and run exactly as the LSTM Model. Due to differences in Model structure, complex quantization schemes may not be applied equivalently to the two models, so their error behaviors may diverge significantly in this case.

- Each Workload runs for a fixed amount of time. This time should be long enough to activate any thermal effects on performance and efficiency (e.g., increased leakage, thermal throttling, high fan power, etc.), However, longer run times might not achieve the same efficiency.

- The latency is not guaranteed to stay the same for lower throughputs.

Results Consistency

The STAC-ML Test Harness checks the consistency of inference results across runs, Model Instances, and NMI. Consistency is defined as producing bit-identical results from all inference requests given bit-identical input data sets. Inconsistent results are allowed but must be explained.

This SUT demonstrated inconsistencies for the LSTM_A model between the NMI pairs (1, 8), (2, 8) and (4, 8). Myrtle.ai offered the following explanation:

> *For NMI 1, the model is compiled and run using all 8 VOLLO cores on the Artena. For larger numbers of model instances, each model instance is given its own subset of VOLLO cores to compile and run on. This difference in the number of cores that a model is compiled for can result in a different schedule of floating point operations, and therefore inconsistencies across workloads.*

Detailed results of consistency checks are contained in the Quality Notebook [5].

Results Introduction

The remainder of this report summarizes the performance, quality, and efficiency results of the SUT. Qualified members of the STAC Benchmark Council have access to iPython notebooks [3, 4, 5] with more detailed analysis of these test results. Please contact info@STACresearch.com for information on how to obtain access to these results.

Each report section summarizes the results for one of the three LSTM Models. The structure of each section follows the structure below.

Important notes about NMI

The SUT Provider chose the Number of Model Instances (NMI) to test for each Model, and how to optimize for that NMI. These scaling points may not be the same across Models but are required to be the same between a Model and its equivalent Null Model. Note that for a given Model, only one configuration for each NMI has been tested. The fact that the same NMI was tested for two Models does not necessarily imply an analogous configuration of the SUT for the two Models at that NMI.

Visualizations of metrics versus NMI are often presented as bar charts and never presented as line charts. This is to avoid suggesting that the metrics should trend in a certain way as the NMI changes. This is especially relevant for SUTs that may mix latency-optimized and throughput-optimized configurations in the same test sequence.

Performance and Quality Results

Performance and Quality Summary

Total Throughput and worst-case Instance Throughput are reported for each Workload. Latency statistics for a given Workload describe the results from all Model Instances and all test runs and are reported in milliseconds. The table header includes the value of the quality (error) benchmark. Competitive comparisons of this SUT to another SUT are only allowed if the other SUT meets the indicated quality standard.

Detailed analysis of the SUT's error for each NMI, the Model overall, and its associated Null Model is available in the Quality Notebook [5].)

Throughput Bar Charts

These bar charts illustrate Total Throughput and worst-case Instance Throughput for each NMI.

Latency Plots

These plots illustrate the latency distribution for each NMI. The y-axis uses a logarithmic scale to improve visualization across Workloads and latency ranges. The x-axis is annotated with the Total Throughput and worst-case Instance Throughput of each Workload.

Additional latency analysis such as histograms, outlier analysis, and time plots for each run are available in the Performance Notebook [3]. The notebook also compares selected latency metrics of the LSTM Models to their associated Null Models, which sheds light on the SUT's inference overhead, such as the time to transfer data to an accelerator.

Efficiency Results

Efficiency Summary

Efficiency results are reported for each Workload. The Workload Power is the average power consumed during all Inferring Periods. The Energy Efficiency Benchmark is Total Throughput / Workload Power. Instance Power (Workload Power / NMI) is also reported for context.

Effective Volume is the amount of data center space that the SUT renders unavailable for other resources of the same vendor architecture and is constant for all NMI. The Space Efficiency Benchmark is Total Throughput / Effective Volume. Instance Volume (Effective Volume / NMI) is also reported for context.

Efficiency Bar Charts

A pair of bar charts illustrate the Energy Efficiency and Space Efficiency for each NMI.

> Additional analysis of efficiency such as static and active idle power, and detailed power traces are available in the Efficiency Notebook [4].

Results for Model LSTM_A

Performance Results

Performance Summary

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

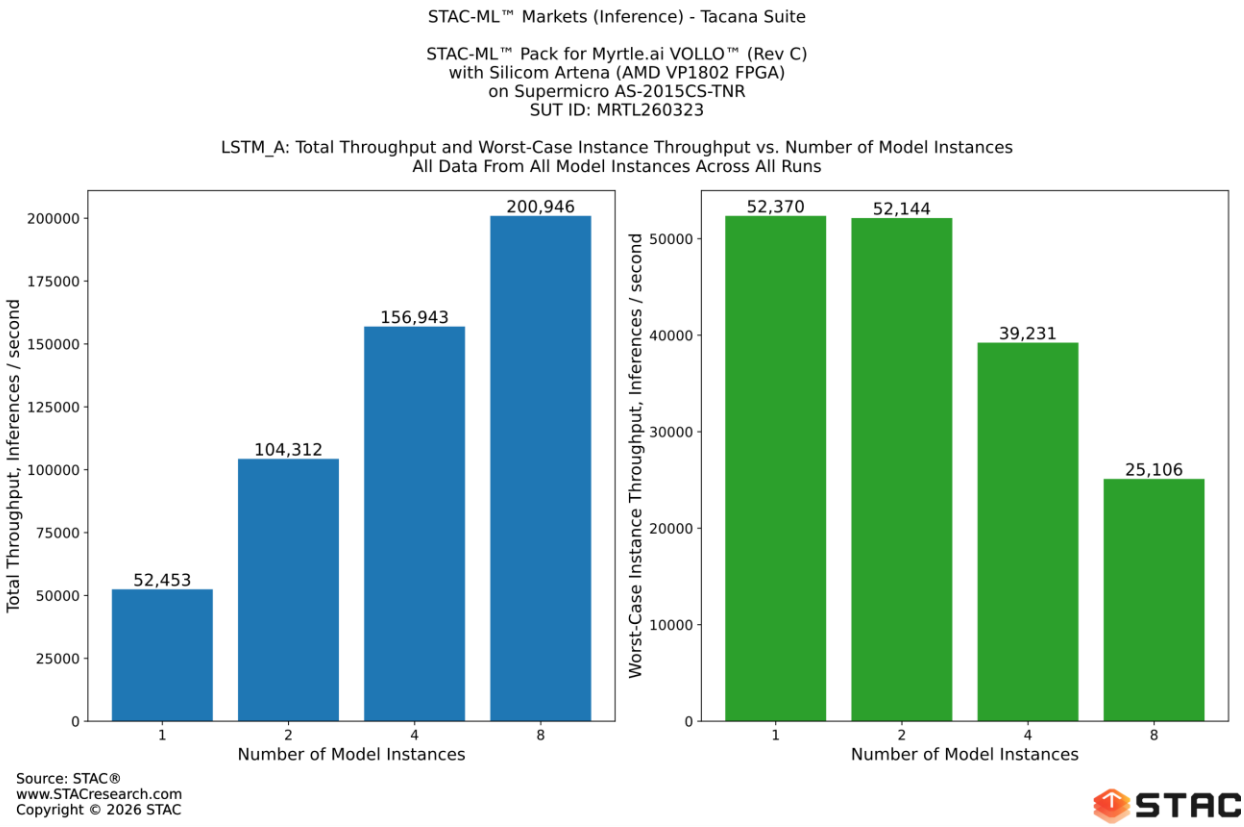
LSTM_A: Performance Summary

Error Benchmark T.LSTM_A.ERR.v1 (99th Percentile): 0.00498
Comparable to SUTs with T.LSTM_A.ERR.v1 (99th Percentile) > 0.00166

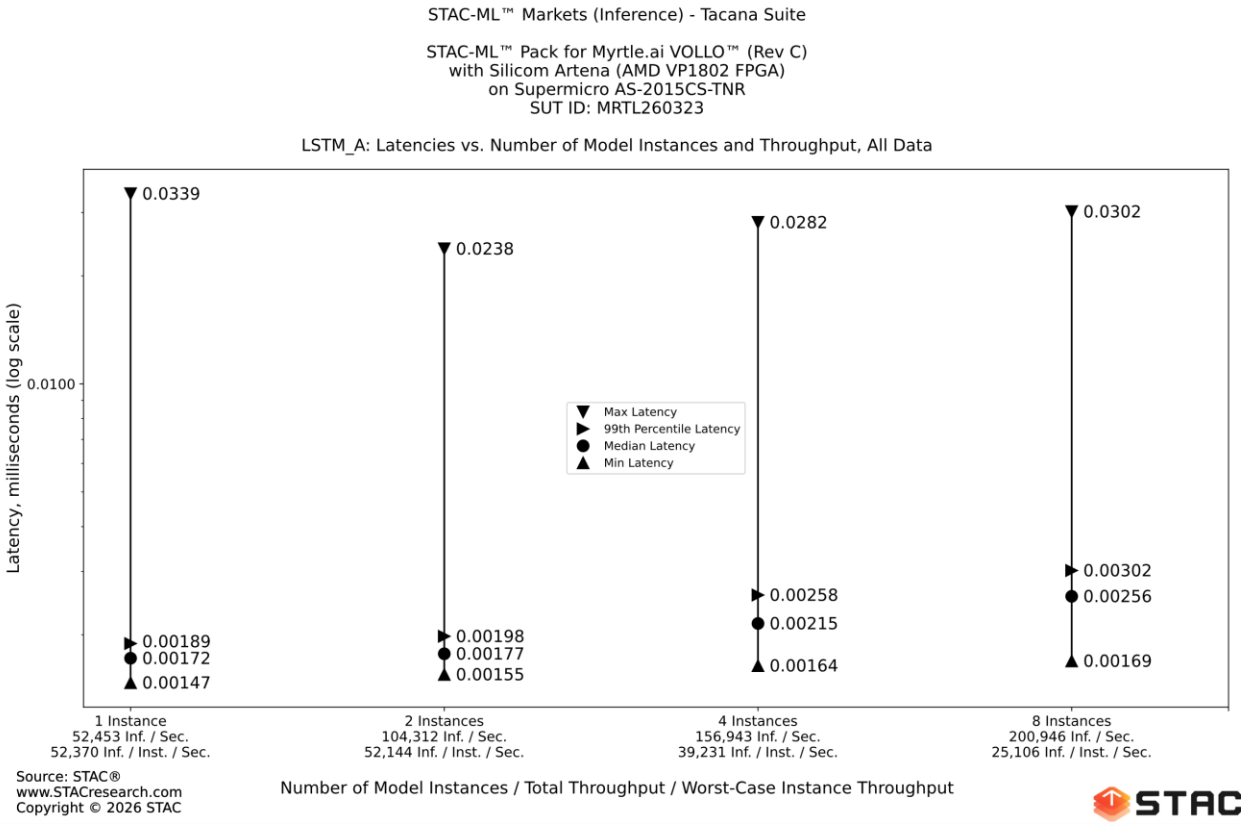
NMI	1	2	4	8
T.LSTM_A.[NMI].TPUT.v1 **Total Throughput Inferences / sec**	52,453	104,312	156,943	200,946
T.LSTM_A.[NMI].INST_TPUT.v1 (Minimum) **Worst-Case Instance Throughput Inferences / sec**	52,370	52,144	39,231	25,106
T.LSTM_A.[NMI].INST_TPUT.v1 **Median Instance Throughput Inferences / sec**	52,478	52,151	39,236	25,119
T.LSTM_A.[NMI].INST_TPUT.v1 **Max Instance Throughput Inferences / sec**	52,521	52,171	39,240	25,125
T.LSTM_A.[NMI].LAT.v1 **Min Latency ms**	0.00147	0.00155	0.00164	0.00169
T.LSTM_A.[NMI].LAT.v1 **Median Latency ms**	0.00172	0.00177	0.00215	0.00256
T.LSTM_A.[NMI].LAT.v1 **99th Percentile Latency ms**	0.00189	0.00198	0.00258	0.00302
T.LSTM_A.[NMI].LAT.v1 **Max Latency ms**	0.0339	0.0238	0.0282	0.0302

Source: STAC
www.STACresearch.com

Throughput Bar Charts



Latency Statistics



Efficiency Results

Efficiency Summary

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

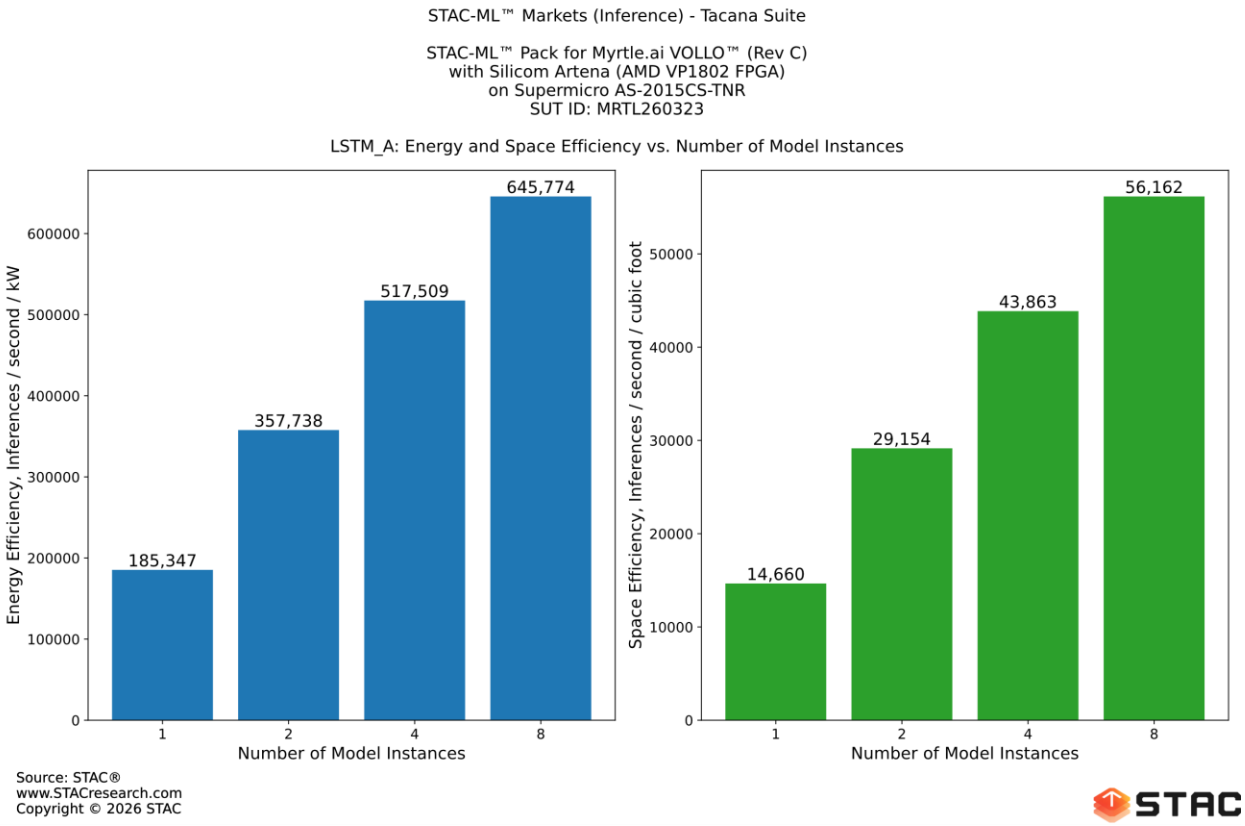
SUT ID: MRTL260323

LSTM_A: Energy and Space Efficiency

NMI	1	2	4	8
T.LSTM_A.[NMI].TPUT.v1 **Total Throughput Inferences / sec**	52,453	104,312	156,943	200,946
Workload Power kW	0.283	0.292	0.303	0.311
T.LSTM_A.[NMI].ENERG_EFF.v1 **Energy Efficiency Inferences / sec / kW**	185,347	357,738	517,509	645,774
Instance Power Watts	283	146	75.8	38.9
Effective Volume ft³	3.5780	3.5780	3.5780	3.5780
T.LSTM_A.[NMI].SPACE_EFF.v1 **Space Efficiency Inferences / sec / ft³**	14,660	29,154	43,863	56,162
Instance Volume ft³	3.5780	1.7890	0.89450	0.44725

Source: STAC
www.STACresearch.com

Efficiency Bar Charts



Results for Model LSTM_B

Performance Results

Performance Summary

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

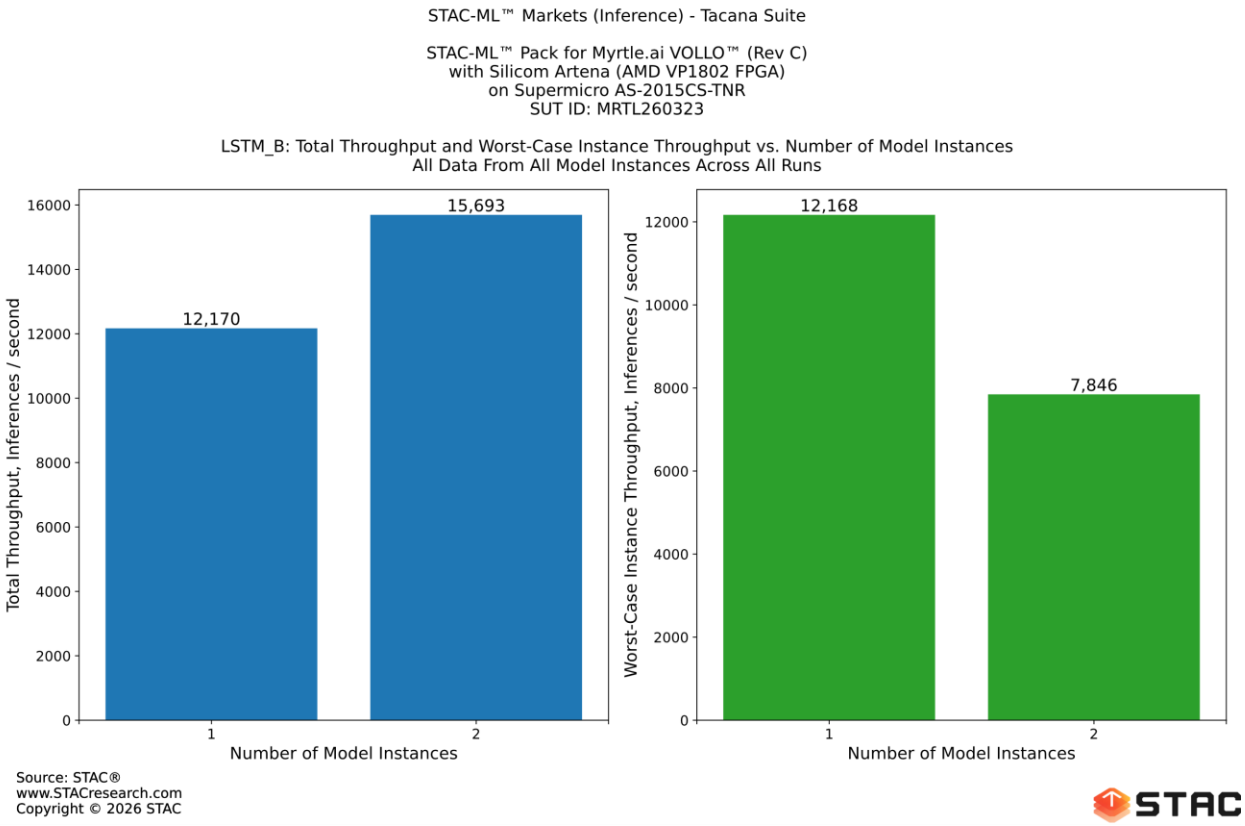
LSTM_B: Performance Summary

Error Benchmark T.LSTM_B.ERR.v1 (99th Percentile): 0.00573
Comparable to SUTs with T.LSTM_B.ERR.v1 (99th Percentile) > 0.00191

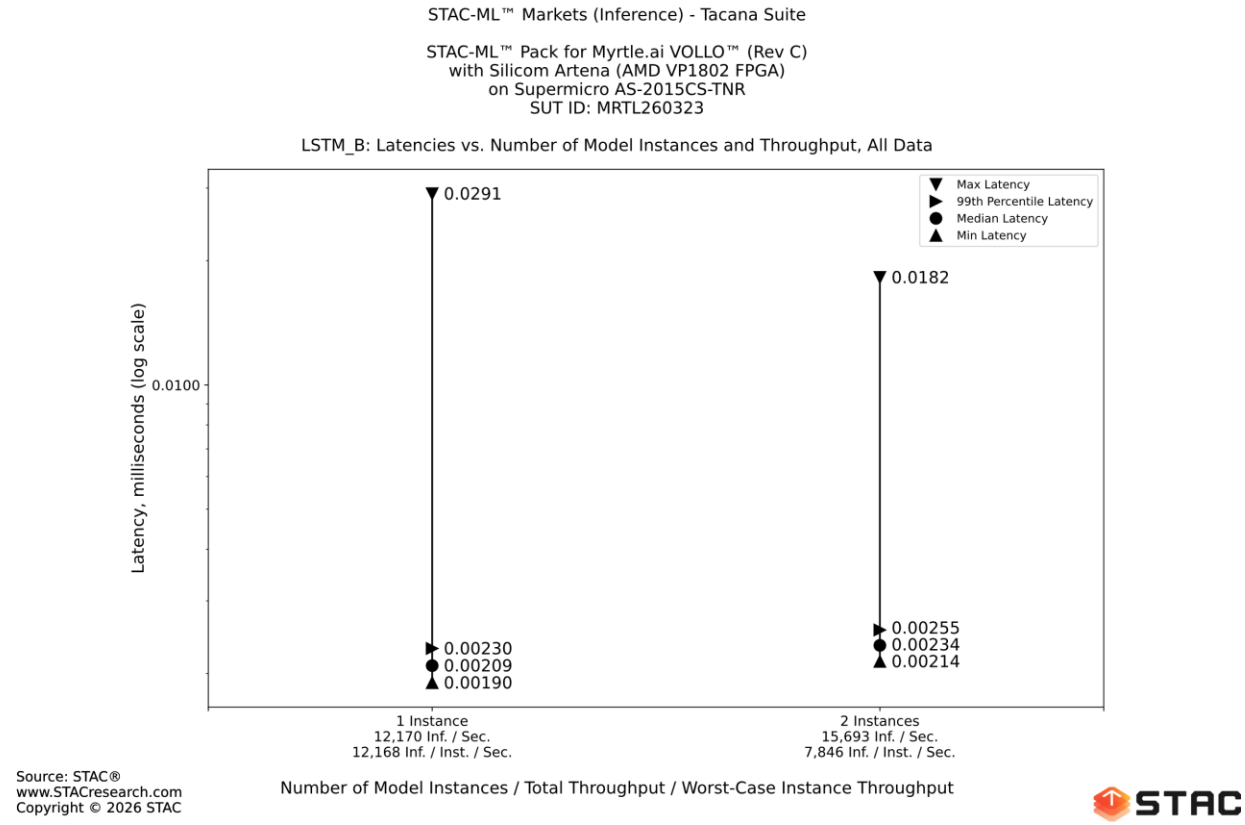
NMI	1	2
T.LSTM_B.[NMI].TPUT.v1 **Total Throughput** **Inferences / sec**	12,170	15,693
T.LSTM_B.[NMI].INST_TPUT.v1 (Minimum) **Worst-Case Instance Throughput** **Inferences / sec**	12,168	7,846
T.LSTM_B.[NMI].INST_TPUT.v1 **Median Instance Throughput** **Inferences / sec**	12,170	7,847
T.LSTM_B.[NMI].INST_TPUT.v1 **Max Instance Throughput** **Inferences / sec**	12,172	7,848
T.LSTM_B.[NMI].LAT.v1 **Min Latency** **ms**	0.00190	0.00214
T.LSTM_B.[NMI].LAT.v1 **Median Latency** **ms**	0.00209	0.00234
T.LSTM_B.[NMI].LAT.v1 **99th Percentile Latency** **ms**	0.00230	0.00255
T.LSTM_B.[NMI].LAT.v1 **Max Latency** **ms**	0.0291	0.0182

Source: STAC
www.STACresearch.com

Throughput Bar Charts



Latency Statistics



Efficiency Results

Efficiency Summary

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

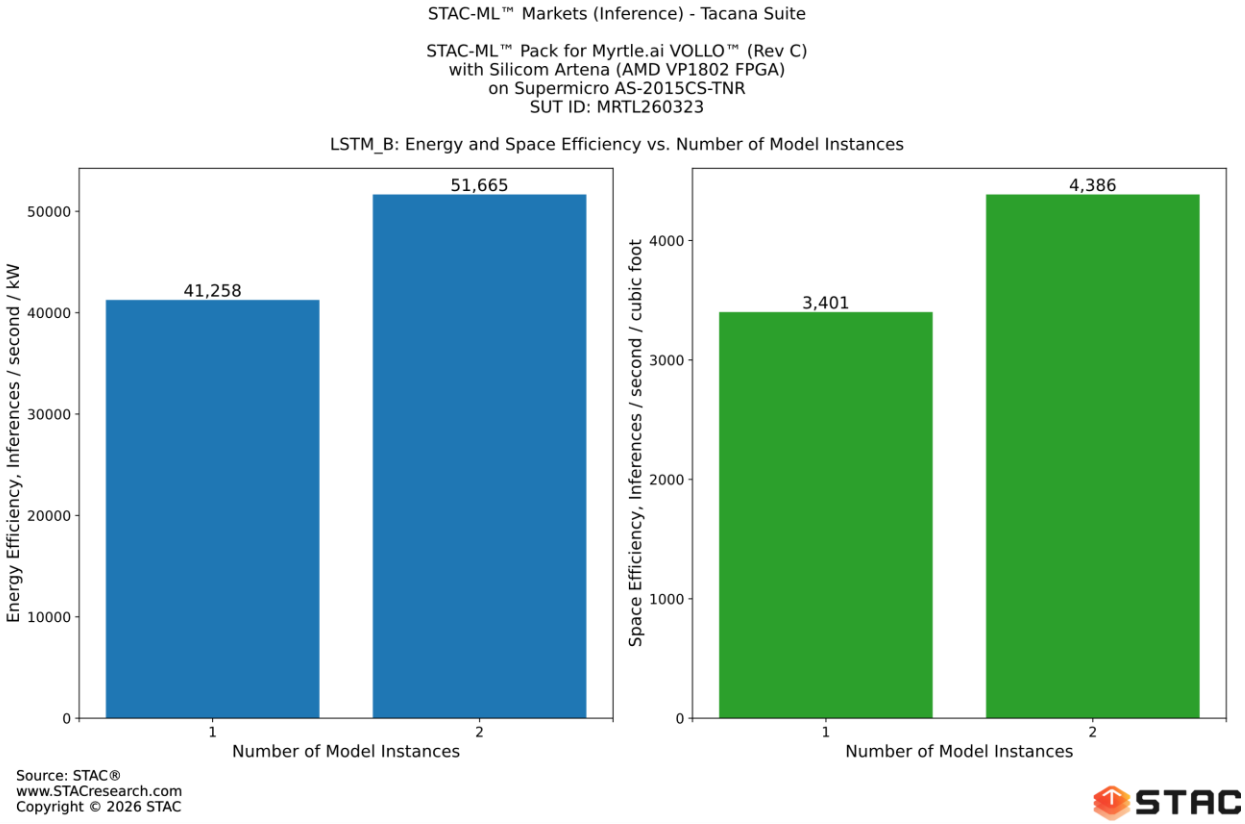
SUT ID: MRTL260323

LSTM_B: Energy and Space Efficiency

NMI	1	2
T.LSTM_B.[NMI].TPUT.v1 **Total Throughput Inferences / sec**	12,170	15,693
Workload Power kW	0.295	0.304
T.LSTM_B.[NMI].ENERG_EFF.v1 **Energy Efficiency Inferences / sec / kW**	41,258	51,665
Instance Power Watts	295	152
Effective Volume ft³	3.5780	3.5780
T.LSTM_B.[NMI].SPACE_EFF.v1 **Space Efficiency Inferences / sec / ft³**	3,401	4,386
Instance Volume ft³	3.5780	1.7890

Source: STAC
www.STACresearch.com

Efficiency Bar Charts



Results for Model LSTM_C

Performance Results

Performance Summary

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

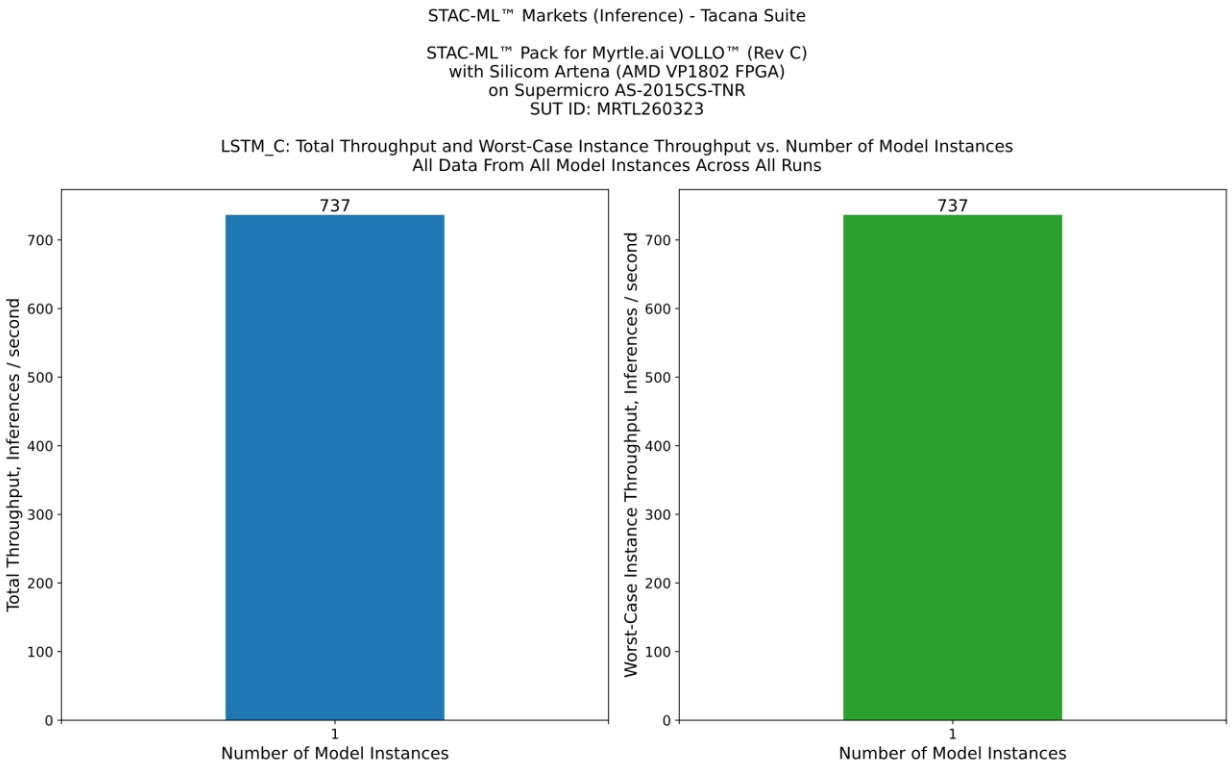
LSTM_C: Performance Summary

Error Benchmark T.LSTM_C.ERR.v1 (99th Percentile): 0.00870
Comparable to SUTs with T.LSTM_C.ERR.v1 (99th Percentile) > 0.00290

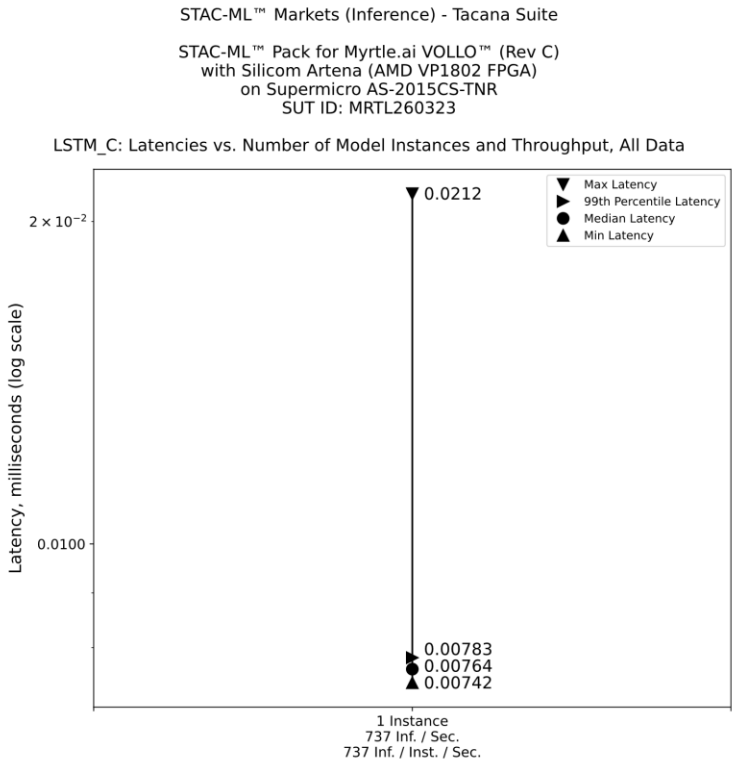
NMI	1
T.LSTM_C.[NMI].TPUT.v1 **Total Throughput** **Inferences / sec**	737
T.LSTM_C.[NMI].INST_TPUT.v1 (Minimum) **Worst-Case Instance Throughput** **Inferences / sec**	737
T.LSTM_C.[NMI].INST_TPUT.v1 **Median Instance Throughput** **Inferences / sec**	737
T.LSTM_C.[NMI].INST_TPUT.v1 **Max Instance Throughput** **Inferences / sec**	737
T.LSTM_C.[NMI].LAT.v1 **Min Latency** **ms**	0.00742
T.LSTM_C.[NMI].LAT.v1 **Median Latency** **ms**	0.00764
T.LSTM_C.[NMI].LAT.v1 **99th Percentile Latency** **ms**	0.00783
T.LSTM_C.[NMI].LAT.v1 **Max Latency** **ms**	0.0212

Source: STAC
www.STACresearch.com

Throughput Bar Charts



Latency Statistics



Efficiency Results

Efficiency Summary

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Pack for Myrtle.ai VOLLO™ (Rev C)
with Silicom Artena (AMD VP1802 FPGA)
on Supermicro AS-2015CS-TNR

SUT ID: MRTL260323

LSTM_C: Energy and Space Efficiency

NMI	1
T.LSTM_C.[NMI].TPUT.v1 **Total Throughput** **Inferences / sec**	737
Workload Power **kW**	0.312
T.LSTM_C.[NMI].ENERG_EFF.v1 **Energy Efficiency** **Inferences / sec / kW**	2,363
Instance Power **Watts**	312
Effective Volume **ft³**	3.5780
T.LSTM_C.[NMI].SPACE_EFF.v1 **Space Efficiency** **Inferences / sec / ft³**	206
Instance Volume **ft³**	3.5780

Source: STAC
www.STACresearch.com

Efficiency Bar Charts