

STAC-ML™ Markets (Inference) Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper™ Superchip

NVIDIA Implementation Optimized for Latency using FP16 Precision

SUT ID: SMC250910

STAC-ML™ Markets (Inference) Benchmarks

Tacaná

Test date: 10th September 2025

Release: v1, 6th October 2025



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.

This document was prepared by the Securities Technology Analysis Center (STAC) at the request of Supermicro. This document is provided for your internal use only and may not be redistributed, retransmitted, or published in any form without the prior written consent of STAC. "STAC" and all STAC names are registered trademarks or trademarks of the Securities Technology Analysis Center LLC. All other trademarks in this document belong to their respective owners.

The test results contained in this report are made available for informational purposes only. Neither STAC nor the vendor(s) supplying the information in this report guarantee similar performance results. All information contained herein is provided on an "AS IS" BASIS WITHOUT WARRANTY OF ANY KIND. STAC explicitly disclaims any liability whatsoever for any errors or otherwise.

Copyright © 2025, STAC. "STAC" and all STAC names are trademarks or registered trademarks of the Securities Technology Analysis Center, LLC. Other company and product names are trademarks of their respective Owners.

STAC REPORT

References.....	3
Summary	4
Vendor Commentary	8
The stack under test	8
Understanding this implementation of STAC-ML Markets (Inference)	8
Understanding the rest of the SUT	9
Project contributors and their roles.....	10
Contacts.....	10
Overview of the STAC-ML Markets (Inference) benchmark specifications	11
Business context	11
Scope	11
Benchmark construction	11
Metrics	14
SUT Flexibility.....	15
Interpreting and comparing results	15
Specification particulars	16
Version	16
Limitations and clarifications	16
Results Consistency	17
Results Introduction	17
Performance and Quality Results	17
Efficiency Results	18
Results for Model LSTM_A	19
Performance Results	19
Efficiency Results	22
Results for Model LSTM_B	24
Performance Results	24
Efficiency Results	27
Results for Model LSTM_C	29
Performance Results	29
Efficiency Results	32

References

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

- [1] Specifications used for this benchmark: STAC-ML Markets (Inference) Benchmark Specifications, Rev F – <https://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/SMC250910> (Same URL as this report)

Summary

STAC recently audited STAC-ML Markets (Inference) benchmark tests on a stack consisting of STAC-ML™ pack for CUDA and cuDNN running on an NVIDIA GH200 Grace Hopper Superchip with 96GiB HBM3 480GiB LDDR5 memory connected with NVLINK C2C in a Supermicro SuperServer® ARS-111GL-NHR. 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 Council¹ 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 an NVIDIA-authored STAC Pack. The STAC Pack used comprises a custom C++ driver compiled with gcc 11.4.0, the NVIDIA CUDA toolkit 12.0, and the NVIDIA CUDA Deep Neural Network library (cuDNN) 9.10.2. 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. Supermicro and NVIDIA would like to point out the following:

Compared to a previous SUT submitted by Myrtle.ai (MRTL230426) on FPGAs, this SUT featuring an NVIDIA GH200 Superchip in a Supermicro ARS-111GL-NHR server demonstrated the following:

- For LSTM_A (the smallest model) the 99p latency was between 7% and 20% lower
 - 7% lower with 1 NMI (4.70μs vs. 5.07μs)
 - 20% lower with 8 NMI (4.67μs vs 5.97μs)
 - The 99p error benchmark was 8 times lower (0.00111 vs 0.00889)
- For LSTM_B (the medium model) the 99p latency was between 3% higher and 8% lower
 - 3% higher with 1 NMI (7.10μs vs. 6.89μs)
 - 8% lower with 4 NMI (7.10 μs vs 7.73μs)
 - The 99p error benchmark was 12 times lower (0.00102 vs 0.0127)
- For LSTM_C (the largest model) with 1 NMI:
 - The 99p latency was 49% lower (15.8μs vs. 31.0μs)
 - The throughput was 15% higher (3,910 vs. 3.387)
 - The 99p error benchmark was 13 times lower (0.00172 vs 0.0237)
 - The energy efficiency was 44% higher (8,312 vs 5,785)
- The largest ratio of maximum to median latency is 9.65 ~ 38.3μs / 3.97μs (occurs at LSTM_A with NMI=8). The smallest ratio of maximum to median latency is 2.16 ~ 32.2μs / 14.9μs (occurs at LSTM_C with NMI=1).

¹ www.STACresearch.com/council

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

LSTM_A Report Card

STAC-ML™ Markets (Inference) - Tacana Suite				
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation				
Optimized for Latency using FP16 Precision				
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip				
SUT ID: SMC250910				
LSTM_A: Report Card				
Error Benchmark T.LSTM_A.ERR.v1 (99th Percentile): 0.00111				
Comparable to SUTs with T.LSTM_A.ERR.v1 (99th Percentile) > 0.000369				
NMI	1	2	4	8
T.LSTM_A.[NMI].TPUT.v1 Total Throughput Inferences / sec	36,982	70,320	117,194	136,458
T.LSTM_A.[NMI].INST_TPUT.v1 (Minimum) Worst-Case Instance Throughput Inferences / sec	36,318	34,411	28,943	16,488
T.LSTM_A.[NMI].LAT.v1 99th Percentile Latency ms	0.00470	0.00467	0.00461	0.00467
T.LSTM_A.[NMI].LAT.v1 Max Latency ms	0.0269	0.0248	0.0369	0.0383
T.LSTM_A.[NMI].ENERG_EFF.v1 Energy Efficiency Inferences / sec / kW	119,601	220,051	350,506	391,099
T.LSTM_A.[NMI].SPACE_EFF.v1 Space Efficiency Inferences / sec / ft³	24,574	46,727	77,875	90,676

Source: STAC
www.STACresearch.com
Copyright © 2025 STAC



LSTM_B Report Card

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation
Optimized for Latency using FP16 Precision
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip

SUT ID: SMC250910

LSTM_B: Report Card

Error Benchmark T.LSTM_B.ERR.v1 (99th Percentile): 0.00102
Comparable to SUTs with T.LSTM_B.ERR.v1 (99th Percentile) > 0.000341

NMI	1	2	4
<i>T.LSTM_B.[NMI].TPUT.v1</i> Total Throughput Inferences / sec	25,159	46,358	71,451
<i>T.LSTM_B.[NMI].INST_TPUT.v1 (Minimum)</i> Worst-Case Instance Throughput Inferences / sec	25,049	22,659	17,669
<i>T.LSTM_B.[NMI].LAT.v1</i> 99th Percentile Latency ms	0.00710	0.00688	0.00710
<i>T.LSTM_B.[NMI].LAT.v1</i> Max Latency ms	0.0302	0.0342	0.0420
<i>T.LSTM_B.[NMI].ENERG_EFF.v1</i> Energy Efficiency Inferences / sec / kW	74,147	124,181	170,771
<i>T.LSTM_B.[NMI].SPACE_EFF.v1</i> Space Efficiency Inferences / sec / ft³	16,718	30,805	47,479

Source: STAC
www.STACresearch.com
Copyright © 2025 STAC



LSTM_C Report Card

STAC-ML™ Markets (Inference) - Tacana Suite	
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation	
Optimized for Latency using FP16 Precision	
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip	
SUT ID: SMC250910	
LSTM_C: Report Card	
Error Benchmark T.LSTM_C.ERR.v1 (99th Percentile): 0.00172	
Comparable to SUTs with T.LSTM_C.ERR.v1 (99th Percentile) > 0.000574	
NMI	1
T.LSTM_C.[NMI].TPUT.v1 Total Throughput Inferences / sec	3,910
T.LSTM_C.[NMI].INST_TPUT.v1 (Minimum) Worst-Case Instance Throughput Inferences / sec	3,904
T.LSTM_C.[NMI].LAT.v1 99th Percentile Latency ms	0.0158
T.LSTM_C.[NMI].LAT.v1 Max Latency ms	0.0322
T.LSTM_C.[NMI].ENERG_EFF.v1 Energy Efficiency Inferences / sec / kW	8,312
T.LSTM_C.[NMI].SPACE_EFF.v1 Space Efficiency Inferences / sec / ft³	2,598

Source: STAC
www.STACresearch.com
Copyright © 2025 STAC

Vendor Commentary

Supermicro provided the following comments:

The audit was performed on a server equipped with a single GPU, highlighting the solution's efficiency in environments with strict space and energy constraints, such as market colocation. Our results demonstrate that multiple independent instances can be executed simultaneously on a single GPU by utilizing green contexts and concurrent CUDA streams.

We would like to highlight that this stack achieves not only comparable (LSTM B) or lower latencies than all previous submissions—including those using FPGAs—at the 99th percentile of the measured distribution but also avoids extreme outliers, i.e., the maximum latency is less than 10 times the median latency for all workloads. Consequently, the most extreme latency measurements are near the median, making them predictable. In addition, our solution consistently demonstrates superior accuracy.

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 NVIDIA, called the STAC-ML Markets (Inference) Pack for CUDA and cuDNN. The source code is available to qualified STAC subscribers. NVIDIA provided the following description:

This implementation of STAC-ML showcases the NVIDIA H200 GPU and Grace CPU, our flagship data center product for AI workloads. The code was developed in C++20 code. The implementation uses some of the highly optimized libraries delivered with CUDA:

- [*cuBLAS*](#): the GPU-enabled implementation of the linear algebra package BLAS
- [*CUDA Streams*](#): enables running multiple kernels in parallel on a single device
- [*CUDA Graphs*](#): removes kernel launch overhead
- [*CUDA Green contexts*](#): allow to serve multiple instances with consistent latencies

The benchmark consists of two distinct phases:

- **Precomputation:**
This stage produces the necessary inputs for the final time step of the sliding window LSTM. Notably, this precomputation phase is excluded from all latency timing measurements. But not from the throughput measurements.
- **Inference:**
In this phase, the last LSTM time step is processed using new input data.

Inference Implementation

Inference utilizes a persistent kernel approach, ensuring the kernel remains active for the entire duration of the application. This persistence enhances performance by loading weights into shared memory and registers just once during kernel initialization.

- *Timing is coordinated by CPU, as specified by the benchmark.*
- *Synchronization between host and device employs standard CPU/GPU synchronization primitives.*
- *The host signals the device when new input arrives in host memory and simultaneously starts the timer.*
- *The kernel polls for this signal, then reads the input and initiates computation.*
- *The computed floating-point output also acts as the host's signal to stop the timer.*

Maximizing Throughput:

To further boost throughput when handling multiple NMIs, each persistent kernel is launched within its own green context. Every green context is managed by an independent CPU thread.

Understanding the rest of the SUT

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

- NVIDIA CUDA Toolkit 12.9
- NVIDIA CUDA Deep Neural Network library (cuDNN) 9.10.2
- Supermicro ARS-111GL-NHR
 - NVIDIA 72-core NVIDIA Grace CPU on GH200 Grace Hopper™ Superchip @ 3.1Ghz
 - Integrated H100 GPU using NVLink® Chip-2-Chip (C2C) 900GiB/s interlink
 - 480GiB of ECC LPDDR5X memory @ 6400Mhz
 - 96GiB of ECC HBM3 memory
- Ubuntu 22.04.5 LTS Server with HWE kernel

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

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

*The [ARS-111GL-NHR](https://www.supermicro.com/en/products/system/gpu/1u/ars-111gl-nhr) (<https://www.supermicro.com/en/products/system/gpu/1u/ars-111gl-nhr>) is a powerful **1U GPU system** specifically engineered for demanding AI workloads, which makes it highly relevant for the STAC-ML Markets Inference benchmark, focusing on high-frequency, low-latency financial inference. Its core capability lies in the integrated **NVIDIA GH200 Grace Hopper Superchip**, which combines the NVIDIA Grace CPU and a high-performance NVIDIA Hopper GPU with 900 GB/s **NVLink Chip-2-Chip (C2C) interconnect**.*

*This single-node, high-density architecture is crucial for the **Tacana suite**'s requirement for extremely low and predictable **99th-percentile latency** in continuous, sliding-window inference on time-series data.*

*Features supporting this include up to 576 GB of **coherent memory** (including 96 GB HBM3 for the GPU) for handling large models like those used in LLMs, and **PCIe 5.0 x16 slots** to support high-speed networking like **NVIDIA BlueField-3** or **ConnectX-7** DPUs, which are vital for ultra-low-latency market data processing and feed handling—key aspects of optimizing for STAC-ML inference benchmarks.*

Supermicro has a long-standing history of providing high-performance, purpose-built server solutions for AI and machine learning workloads, including those in the financial sector. Our expertise is reflected in our robust, application-optimized systems that are designed to deliver exceptional performance on industry benchmarks like the STAC-ML Markets Inference benchmarks.

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

The NVIDIA GH200 Grace Hopper Superchip broadens the robust 64-bit Arm processor ecosystem, supporting a wide range of containers, application binaries, and operating systems—these run seamlessly on Grace Hopper without the need for modification. The Superchip is fully compatible with the NVIDIA software suite, including NVIDIA HPC and AI platforms. The advanced Hopper architecture, offering a theoretical peak of 67 TFlops for single-precision and 34 TFlops for double-precision operations with its 16,896 CUDA cores. It also includes 528 fourth-generation TensorCore modules, delivering up to 990 TFlops for FP16/BF16 computations. Employing CoWoS HBM3 technology, the Hopper GPU integrates compute and memory within a single silicon package, which results in approximately double the memory bandwidth compared to the Ampere A100 generation. Its distributed shared memory feature minimizes kernel latency by enabling direct SM-to-SM communication for loads, stores, and atomics across multiple SM shared memory blocks. Equipped with 96 GiB CoWoS HBM3, the GPU achieves a maximum theoretical memory bandwidth of 4 TB/s, and interfaces with the Grace CPU at up to 900 GB/s via memory-coherent NVIDIA C2C NVLink.

Project contributors and their roles

The following firms contributed to this project:

- Supermicro
- NVIDIA
- STAC

The Project Participants had the following responsibilities:

- NVIDIA wrote the STAC-ML Pack in accordance with the STAC-ML Markets (Inference) benchmark specifications; provided the SUT server, the GPU, test lab, and power- and temperature-monitoring equipment; configured the hardware and software.
- Supermicro provided server specifications and sponsored the report.
- STAC edited the final STAC Report and STAC Configuration Disclosure, based on information provided by the sponsor(s).

Contacts

- Supermicro: financialservices@supermicro.com
- NVIDIA: nmarkovskiy@NVIDIA.com
- 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.³

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.

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

³ 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.

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

Copyright © 2025 STAC

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.

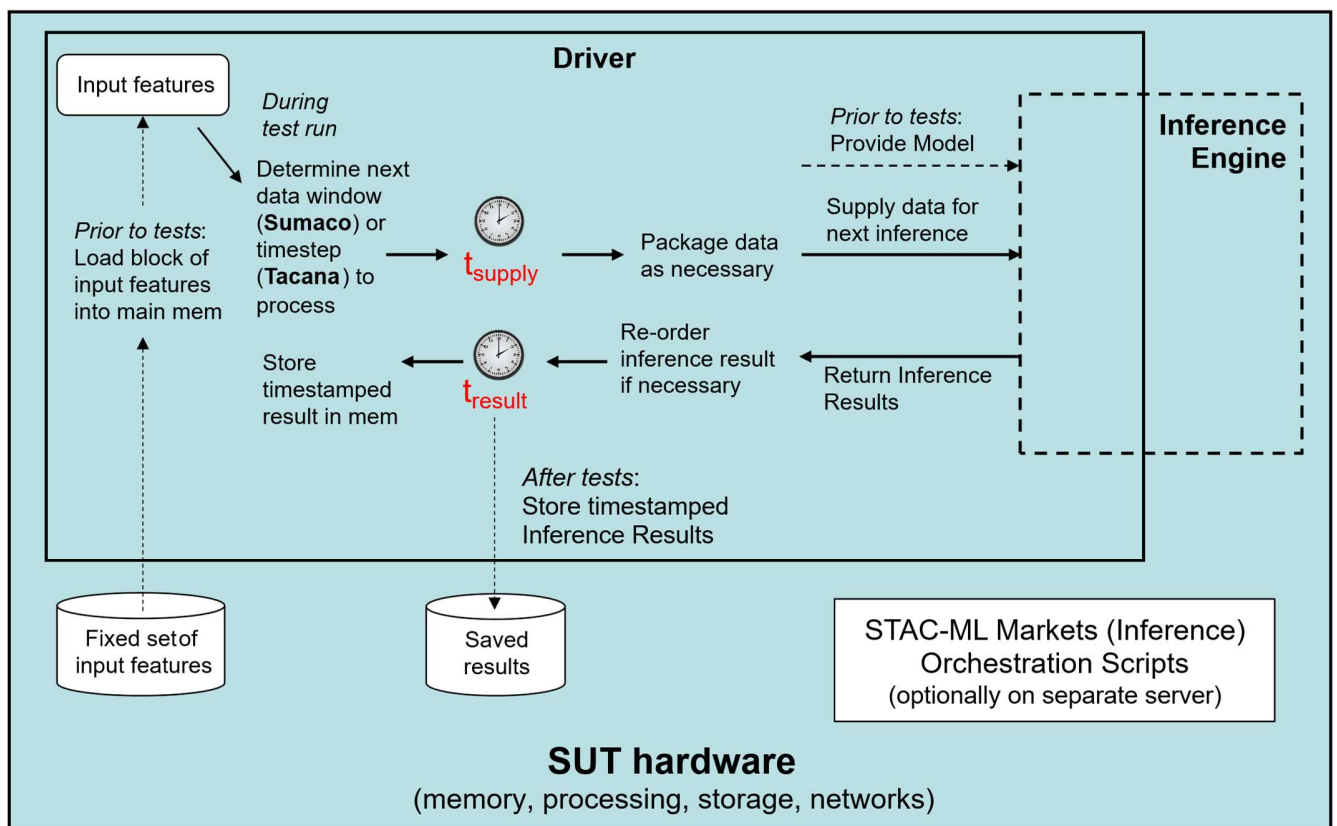


Figure 1 – High-level benchmark diagram

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-specific 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.

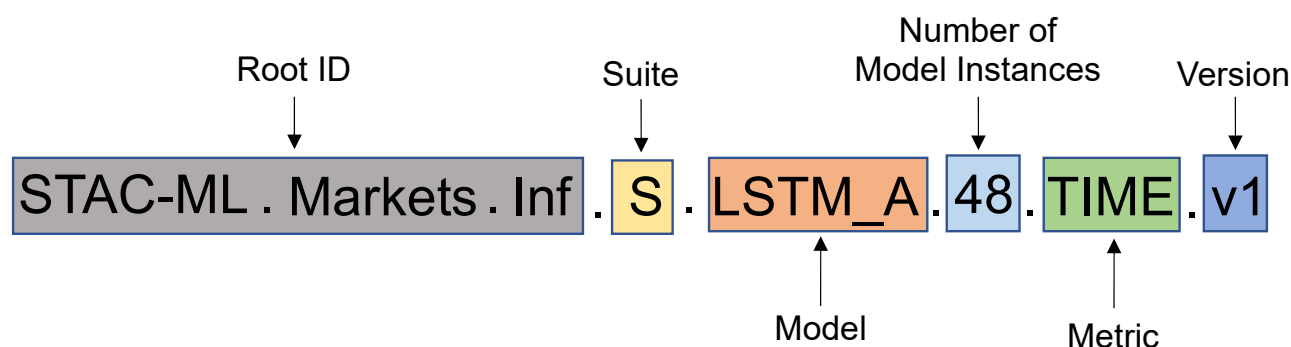
SUT Flexibility

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 chooses 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 due to the use of non-deterministic inference algorithms are allowed but must be explained.

This SUT demonstrated no inconsistencies.

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™ Markets (Inference) Nvidia Low Latency Implementation Optimized for Latency using FP16 Precision Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip SUT ID: SMC250910 LSTM_A: Performance Summary Error Benchmark T.LSTM_A.ERR.v1 (99th Percentile): 0.00111 Comparable to SUTs with T.LSTM_A.ERR.v1 (99th Percentile) > 0.000369				
NMI	1	2	4	8
<i>T.LSTM_A.[NMI].TPUT.v1</i> Total Throughput Inferences / sec	36,982	70,320	117,194	136,458
<i>T.LSTM_A.[NMI].INST_TPUT.v1 (Minimum)</i> Worst-Case Instance Throughput Inferences / sec	36,318	34,411	28,943	16,488
<i>T.LSTM_A.[NMI].INST_TPUT.v1</i> Median Instance Throughput Inferences / sec	37,197	35,276	29,296	17,071
<i>T.LSTM_A.[NMI].INST_TPUT.v1</i> Max Instance Throughput Inferences / sec	37,452	35,517	29,607	17,396
<i>T.LSTM_A.[NMI].LAT.v1</i> Min Latency ms	0.00298	0.00304	0.00307	0.00298
<i>T.LSTM_A.[NMI].LAT.v1</i> Median Latency ms	0.00381	0.00368	0.00384	0.00397
<i>T.LSTM_A.[NMI].LAT.v1</i> 99th Percentile Latency ms	0.00470	0.00467	0.00461	0.00467
<i>T.LSTM_A.[NMI].LAT.v1</i> Max Latency ms	0.0269	0.0248	0.0369	0.0383

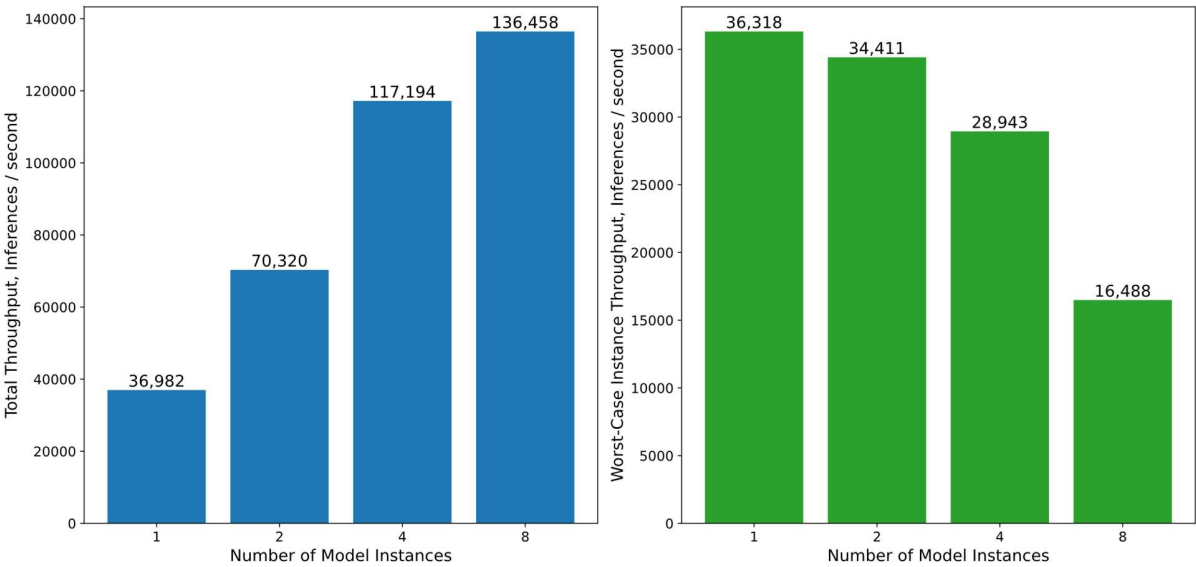
Source: STAC
 www.STACresearch.com
 Copyright © 2025 STAC

Throughput Bar Charts

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation
Optimized for Latency using FP16 Precision
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip
SUT ID: SMC250910

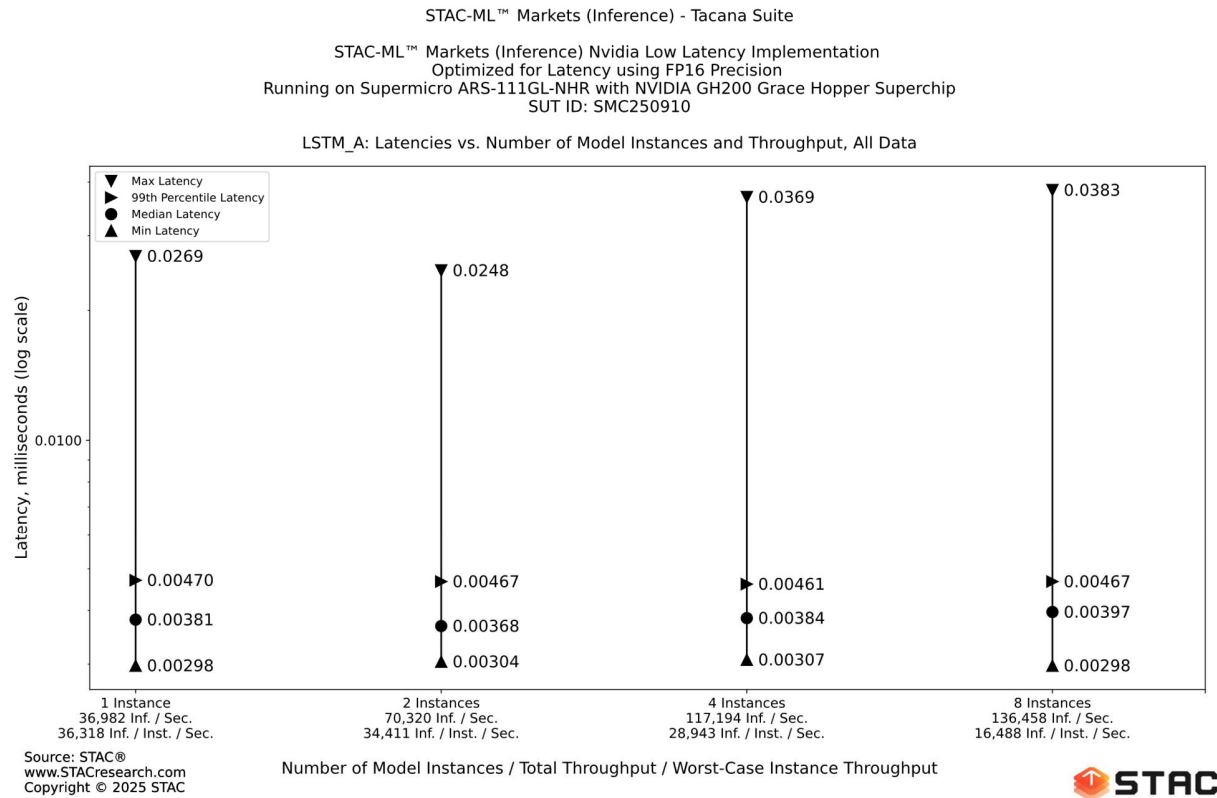
LSTM_A: Total Throughput and Worst-Case Instance Throughput vs. Number of Model Instances
All Data From All Model Instances Across All Runs



Source: STAC®
www.STACresearch.com
Copyright © 2025 STAC



Latency Statistics



Efficiency Results

Efficiency Summary

STAC-ML™ Markets (Inference) - Tacana Suite				
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation				
Optimized for Latency using FP16 Precision				
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip				
SUT ID: SMC250910				
LSTM_A: Energy and Space Efficiency				
NMI	1	2	4	8
<i>T.LSTM_A.[NMI].TPUT.v1</i> Total Throughput Inferences / sec	36,982	70,320	117,194	136,458
Workload Power kW	0.309	0.320	0.334	0.349
<i>T.LSTM_A.[NMI].ENERG_EFF.v1</i> Energy Efficiency Inferences / sec / kW	119,601	220,051	350,506	391,099
Instance Power Watts	309	160	83.6	43.6
Effective Volume ft³	1.5049	1.5049	1.5049	1.5049
<i>T.LSTM_A.[NMI].SPACE_EFF.v1</i> Space Efficiency Inferences / sec / ft³	24,574	46,727	77,875	90,676
Instance Volume ft³	1.5049	0.75245	0.37622	0.18811

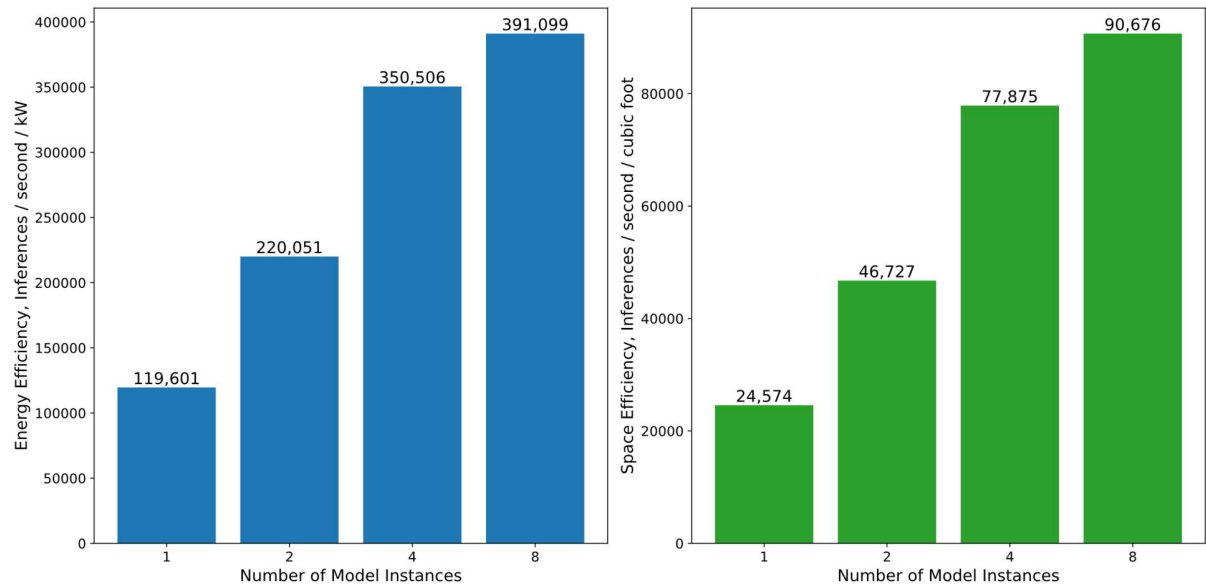
Source: STAC
www.STACresearch.com
Copyright © 2025 STAC

Efficiency Bar Charts

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation
Optimized for Latency using FP16 Precision
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip
SUT ID: SMC250910

LSTM_A: Energy and Space Efficiency vs. Number of Model Instances



Source: STAC®
www.STACresearch.com
Copyright © 2025 STAC



Results for Model LSTM_B

Performance Results

Performance Summary

STAC-ML™ Markets (Inference) - Tacana Suite STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation Optimized for Latency using FP16 Precision Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip SUT ID: SMC250910 LSTM_B: Performance Summary Error Benchmark T.LSTM_B.ERR.v1 (99th Percentile): 0.00102 Comparable to SUTs with T.LSTM_B.ERR.v1 (99th Percentile) > 0.000341			
NMI	1	2	4
<i>T.LSTM_B.[NMI].TPUT.v1</i> Total Throughput Inferences / sec	25,159	46,358	71,451
<i>T.LSTM_B.[NMI].INST_TPUT.v1 (Minimum)</i> Worst-Case Instance Throughput Inferences / sec	25,049	22,659	17,669
<i>T.LSTM_B.[NMI].INST_TPUT.v1</i> Median Instance Throughput Inferences / sec	25,168	23,309	17,868
<i>T.LSTM_B.[NMI].INST_TPUT.v1</i> Max Instance Throughput Inferences / sec	25,316	23,369	18,022
<i>T.LSTM_B.[NMI].LAT.v1</i> Min Latency ms	0.00522	0.00525	0.00490
<i>T.LSTM_B.[NMI].LAT.v1</i> Median Latency ms	0.00627	0.00573	0.00586
<i>T.LSTM_B.[NMI].LAT.v1</i> 99th Percentile Latency ms	0.00710	0.00688	0.00710
<i>T.LSTM_B.[NMI].LAT.v1</i> Max Latency ms	0.0302	0.0342	0.0420

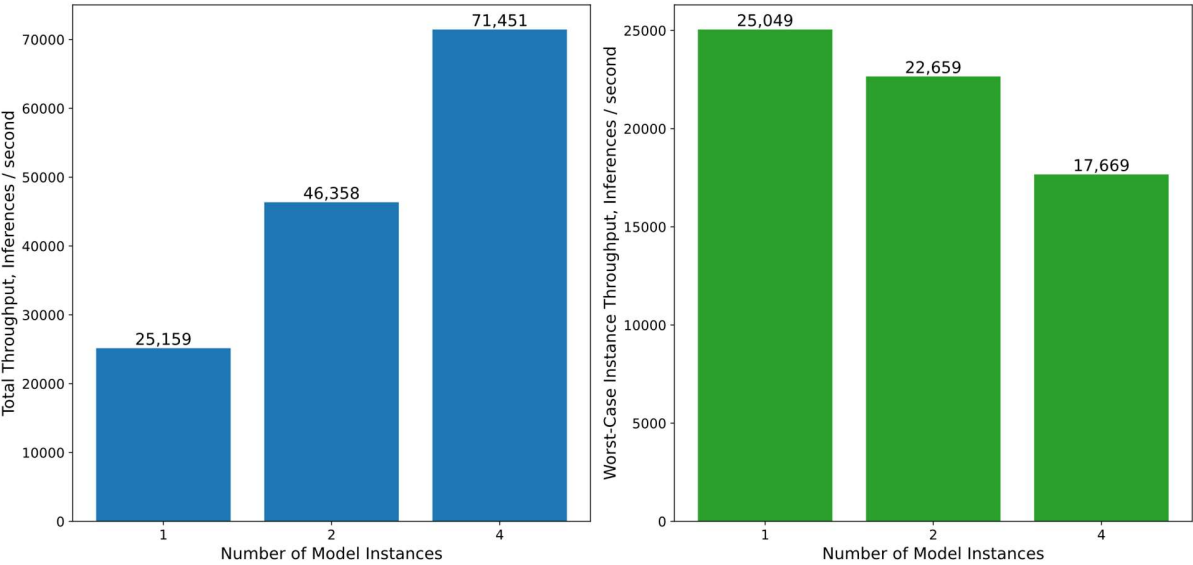
Source: STAC
 www.STACresearch.com
 Copyright © 2025 STAC

Throughput Bar Charts

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation
Optimized for Latency using FP16 Precision
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip
SUT ID: SMC250910

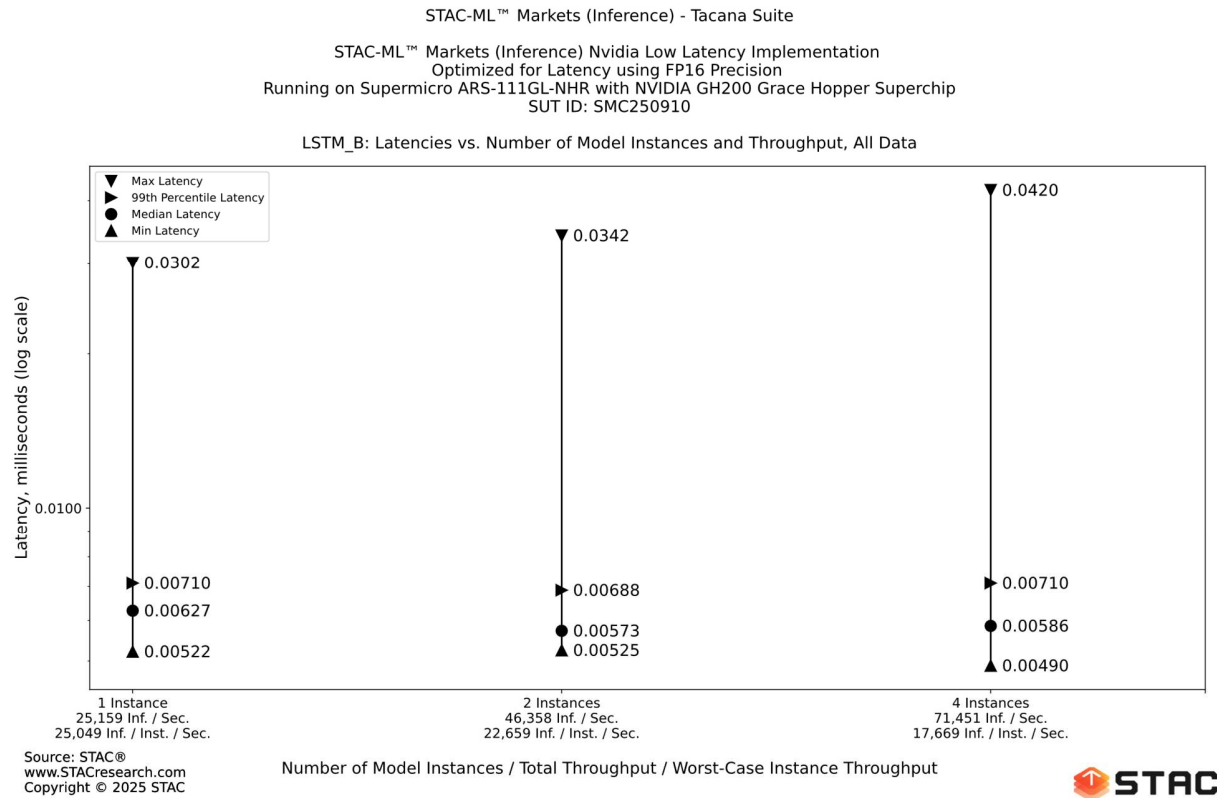
LSTM_B: Total Throughput and Worst-Case Instance Throughput vs. Number of Model Instances
All Data From All Model Instances Across All Runs



Source: STAC®
www.STACresearch.com
Copyright © 2025 STAC



Latency Statistics



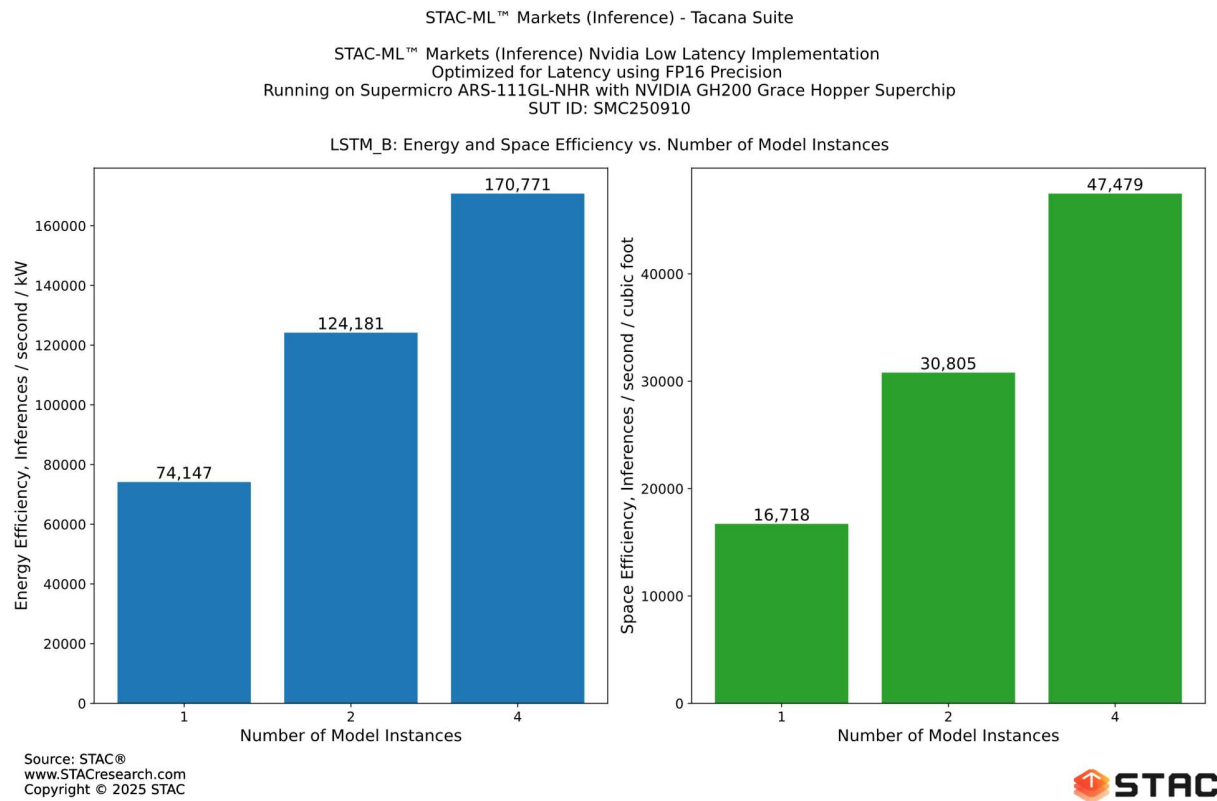
Efficiency Results

Efficiency Summary

STAC-ML™ Markets (Inference) - Tacana Suite			
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation			
Optimized for Latency using FP16 Precision			
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip			
SUT ID: SMC250910			
LSTM_B: Energy and Space Efficiency			
NMI	1	2	4
<i>T.LSTM_B.[NMI].TPUT.v1</i> Total Throughput Inferences / sec	25,159	46,358	71,451
Workload Power kW	0.339	0.373	0.418
<i>T.LSTM_B.[NMI].ENERG_EFF.v1</i> Energy Efficiency Inferences / sec / kW	74,147	124,181	170,771
Instance Power Watts	339	187	105
Effective Volume ft³	1.5049	1.5049	1.5049
<i>T.LSTM_B.[NMI].SPACE_EFF.v1</i> Space Efficiency Inferences / sec / ft³	16,718	30,805	47,479
Instance Volume ft³	1.5049	0.75245	0.37622

Source: STAC
www.STACresearch.com
Copyright © 2025 STAC

Efficiency Bar Charts



Results for Model LSTM_C

Performance Results

Performance Summary

STAC-ML™ Markets (Inference) - Tacana Suite	
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation	
Optimized for Latency using FP16 Precision	
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip	
SUT ID: SMC250910	
LSTM_C: Performance Summary	
Error Benchmark T.LSTM_C.ERR.v1 (99th Percentile): 0.00172	
Comparable to SUTs with T.LSTM_C.ERR.v1 (99th Percentile) > 0.000574	
NMI	1
T.LSTM_C.[NMI].TPUT.v1 Total Throughput Inferences / sec	3,910
T.LSTM_C.[NMI].INST_TPUT.v1 (Minimum) Worst-Case Instance Throughput Inferences / sec	3,904
T.LSTM_C.[NMI].INST_TPUT.v1 Median Instance Throughput Inferences / sec	3,910
T.LSTM_C.[NMI].INST_TPUT.v1 Max Instance Throughput Inferences / sec	3,914
T.LSTM_C.[NMI].LAT.v1 Min Latency ms	0.0139
T.LSTM_C.[NMI].LAT.v1 Median Latency ms	0.0149
T.LSTM_C.[NMI].LAT.v1 99th Percentile Latency ms	0.0158
T.LSTM_C.[NMI].LAT.v1 Max Latency ms	0.0322

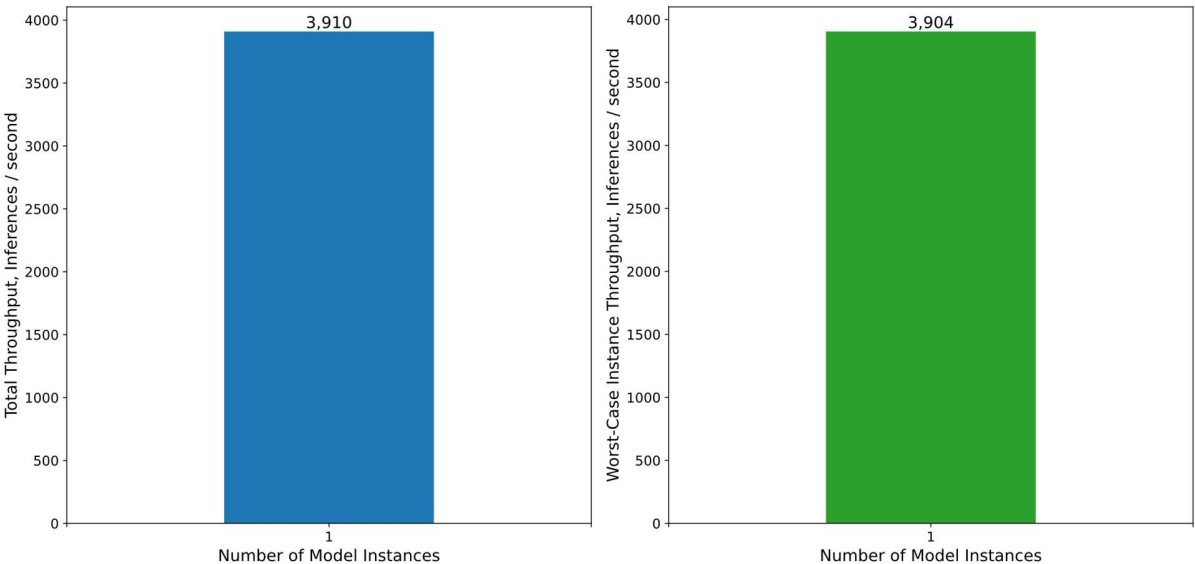
Source: STAC
www.STACresearch.com
Copyright © 2025 STAC

Throughput Bar Charts

STAC-ML™ Markets (Inference) - Tacana Suite

STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation
Optimized for Latency using FP16 Precision
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip
SUT ID: SMC250910

LSTM_C: Total Throughput and Worst-Case Instance Throughput vs. Number of Model Instances
All Data From All Model Instances Across All Runs

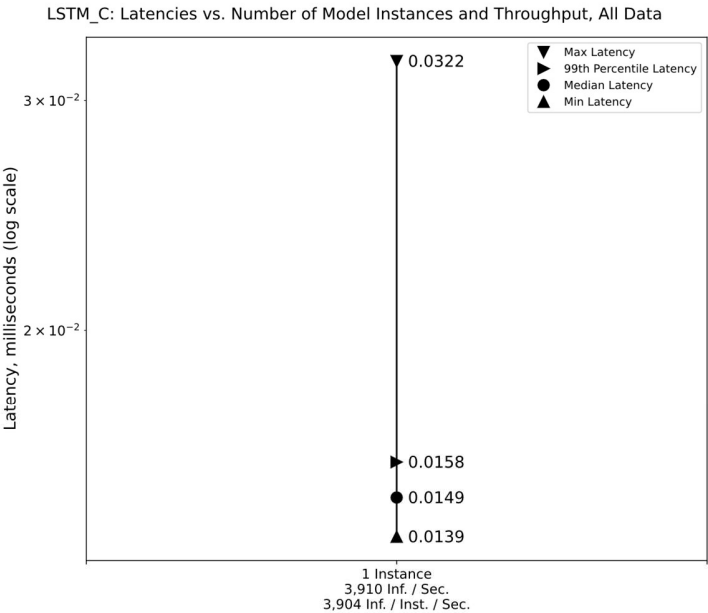


Source: STAC®
www.STACresearch.com
Copyright © 2025 STAC



Latency Statistics

STAC-ML™ Markets (Inference) - Tacana Suite
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation
Optimized for Latency using FP16 Precision
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip
SUT ID: SMC250910



Source: STAC®
www.STACresearch.com
Copyright © 2025 STAC

Number of Model Instances / Total Throughput / Worst-Case Instance Throughput



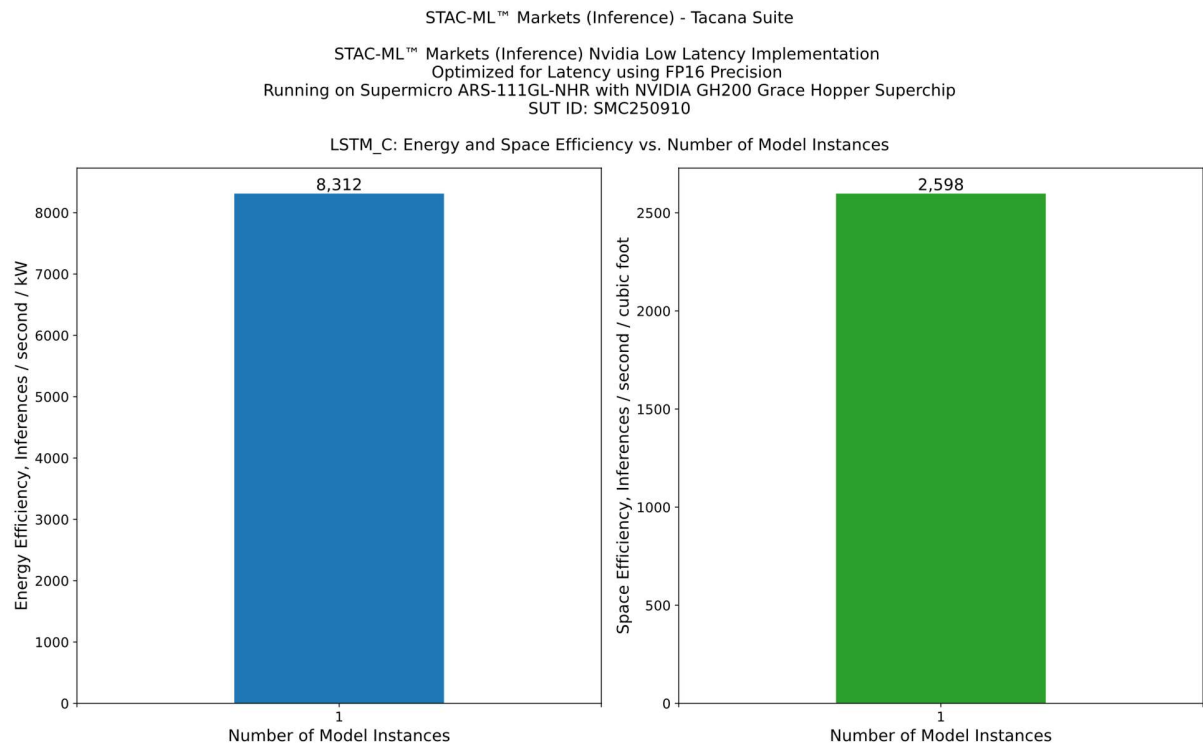
Efficiency Results

Efficiency Summary

STAC-ML™ Markets (Inference) - Tacana Suite	
STAC-ML™ Markets (Inference) Nvidia Low Latency Implementation	
Optimized for Latency using FP16 Precision	
Running on Supermicro ARS-111GL-NHR with NVIDIA GH200 Grace Hopper Superchip	
SUT ID: SMC250910	
LSTM_C: Energy and Space Efficiency	
NMI	1
<i>T.LSTM_C.[NMI].TPUT.v1</i> Total Throughput Inferences / sec	3,910
Workload Power kW	0.470
<i>T.LSTM_C.[NMI].ENERG_EFF.v1</i> Energy Efficiency Inferences / sec / kW	8,312
Instance Power Watts	470
Effective Volume ft³	1.5049
<i>T.LSTM_C.[NMI].SPACE_EFF.v1</i> Space Efficiency Inferences / sec / ft³	2,598
Instance Volume ft³	1.5049

Source: STAC
www.STACresearch.com
Copyright © 2025 STAC

Efficiency Bar Charts



Source: STAC®
www.STACresearch.com
Copyright © 2025 STAC

