The science behind the report:

Gain enterprise-ready agentic AI with the Dell PowerEdge XE7740

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Gain enterprise-ready agentic AI with the Dell PowerEdge XE7740.

We concluded our hands-on testing on June 11, 2026. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on June 8, 2026 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to https://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Manufacturing workflows

Table 1: Our results on our manufacturing workflows. Note that counterintuitively, the 8-GPU run was the most energy-efficient, despite drawing the most power, because it was able to finish the workload much faster than the other configurations.

Dell PowerEdge XE7740 with Intel Xeon 6747P processors and NVIDIA RTX PRO 6000 Blackwell GPUs			
	4-GPU config	6-GPU config	8-GPU config
Estimated concurrent # of agentic users supported at SLA (higher is better)	8	12	15
Sustained throughput at SLA in tokens per second (higher is better)	144	274	321
Energy per run consumed at load (kWh) (lower is better)	4.4712	3.4745	2.4606
Energy per MTok (J/Mtok) (lower is better)	1,203	1,052	1,089

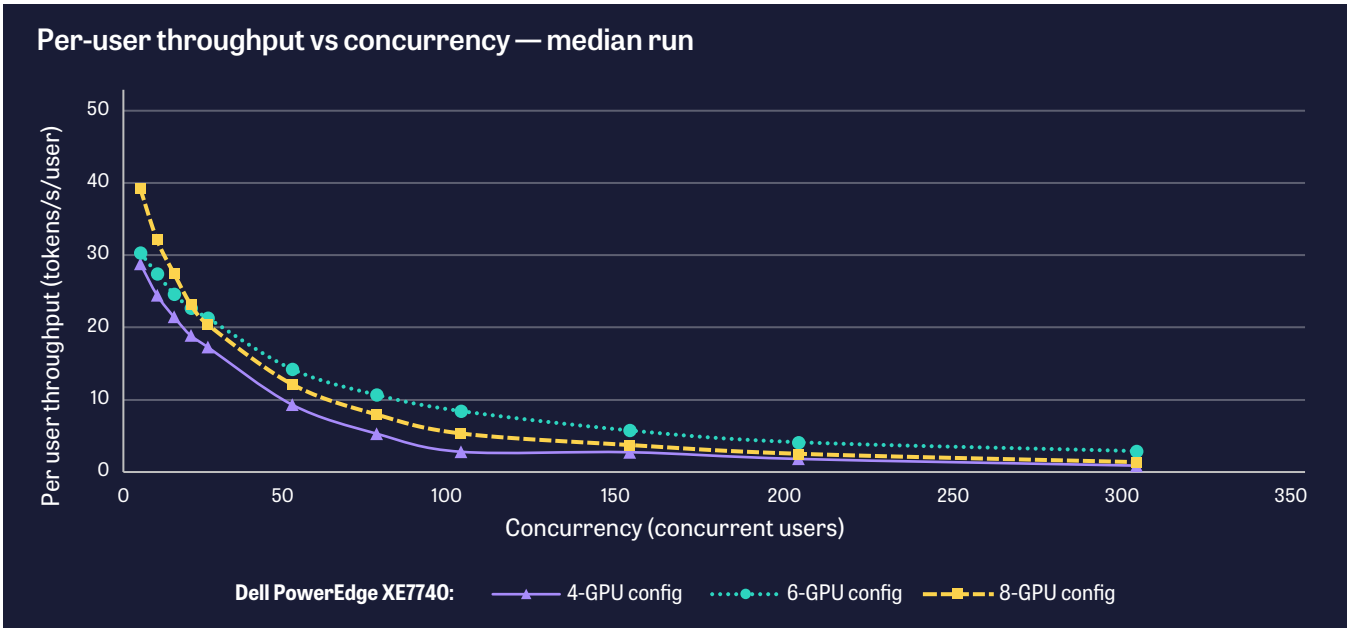

Figure 1: Per-user throughput, in tokens per second per agentic user, that each GPU configuration of the Dell PowerEdge XE7740 delivered on our manufacturing workload as the number of concurrent agentic users increased. Higher is better.

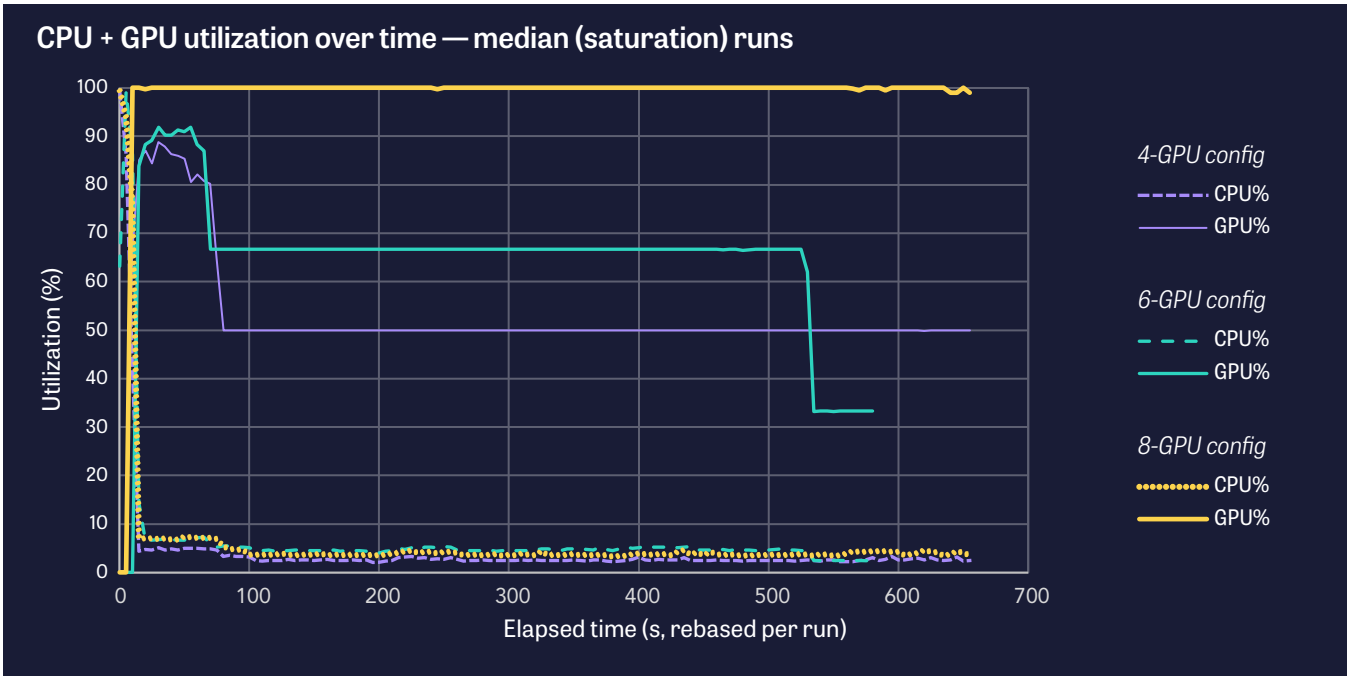

Figure 2: CPU and GPU utilization for all configurations over the course of our manufacturing workflows.

Financial services workflows

Table 2: Our results on our financial services workflows. Note that while the 8-GPU configuration did consume more wall power at load, its run also finished faster than the 6-GPU or 4-GPU configurations, so it was more energy-efficient over the course of the run.

Dell PowerEdge XE7740 with Intel Xeon 6747P processors and NVIDIA RTX PRO 6000 Blackwell GPUs			
	4-GPU config	6-GPU config	8-GPU config
Estimated concurrent # of agentic users supported at SLA (higher is better)	57	70	74
Sustained throughput at SLA in tokens per second (higher is better)	905	990	1,179
Energy per run consumed at load (kWh) (lower is better)	0.1895	0.1970	0.1816
Energy per MTok (J/Mtok) (lower is better)	162	194	144

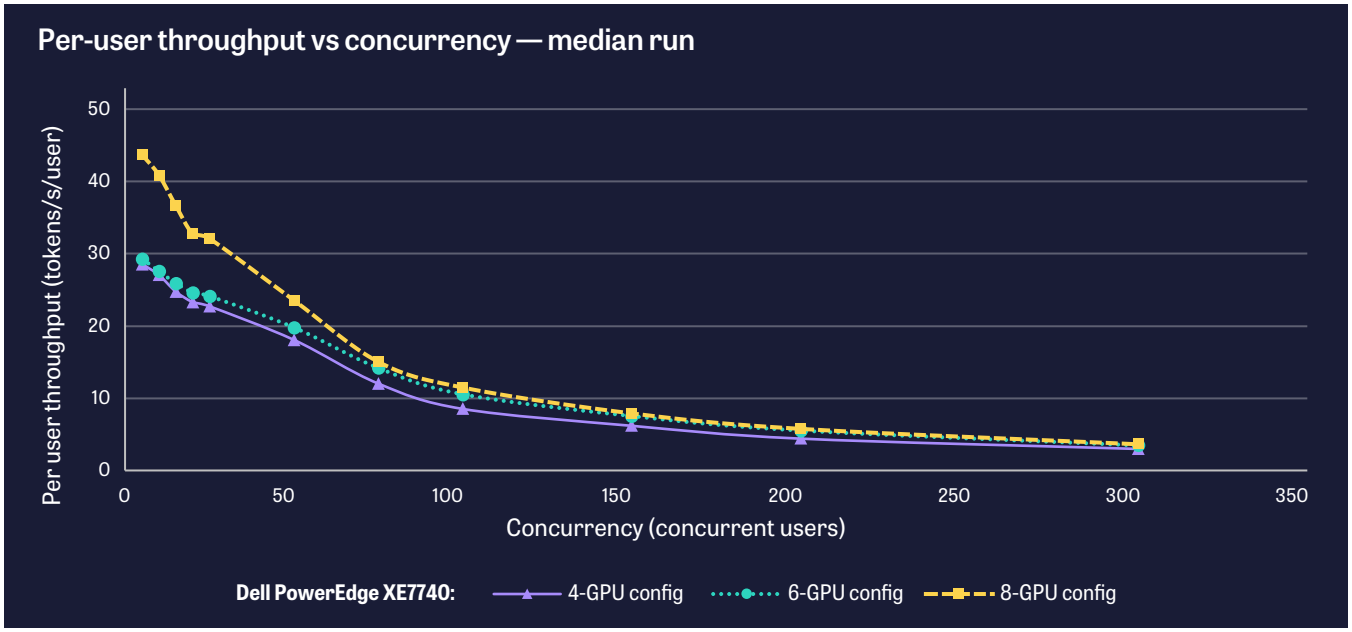

Figure 3: Per-user throughput, in tokens per second per agentic user, that each GPU configuration of the Dell PowerEdge XE7740 delivered on our financial services workload as the number of concurrent agentic users increased. Higher is better.

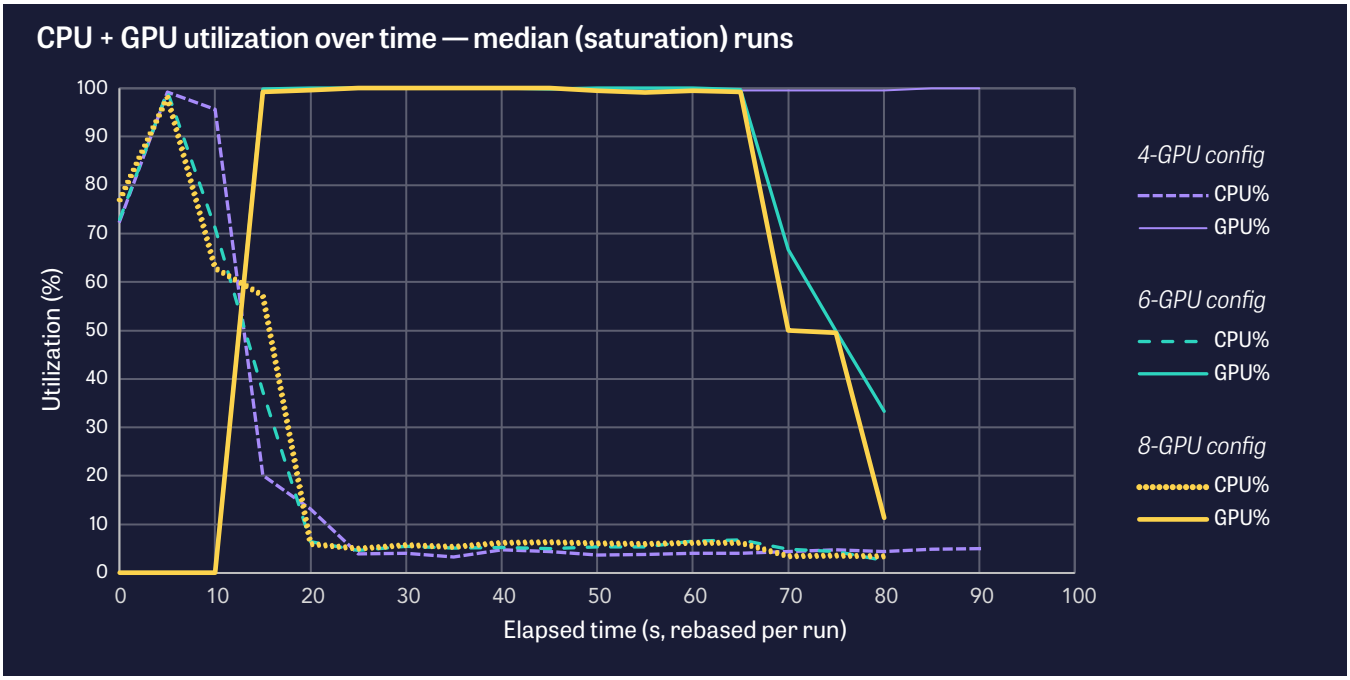

Figure 4: CPU and GPU utilization for all configurations over the course of our financial services workflows.

Our cost assessment

Table 3 shows our example tokenomics assessment, highlighting the 8-GPU configuration of the Dell PowerEdge XE7740 running our financial services workflows.

For our PowerEdge XE7740 calculations, we use the following costs:

- Cost of server: $263,446. Dell provided this price to us on 6/23/26. Your price may vary depending on your configuration, your seller, and other factors.
- Cost of energy: We used the average price of electricity for commercial customers in the United States, which was $0.1385 per kWh as of April 2026 (the most recent available data at the time of this report). See the second table on page 12 at https://www.eia.gov/electricity/monthly/current_month/june2026.pdf.

We did not consider costs for cooling, manageability, networking, storage, software licenses, or data center space.

We amortized the hardware straight-line over a five-year service life; the five-year cost combines the full hardware CAPEX with five years of PSU-wall energy at the utilization we saw in testing. The per-Mtok figure divides that total by the tokens the system produces over those five years.

For the Amazon Bedrock solution, we looked at the price per 1M input tokens and price per 1M output tokens under Llama 3.1 On-Demand and Batch pricing available at https://aws.amazon.com/bedrock/pricing. We compared against Llama 3.1 Instruct (70B) (w/ latency optimized inference) because, given that we were testing against an on-premises system, it was the closest comparison to the workloads we were running. For region, we selected US East (N. Virginia), the region physically closest to where we completed testing in Durham, North Carolina.

For Amazon Bedrock, we did not include any IT cost to keep up with billing or other cloud management challenges.

Table 3: $/Mtok for the 8-GPU PowerEdge XE7740 configuration running our financial services workflows and Amazon Bedrock costs for the same LLM we used in testing.

	Dell™ PowerEdge™ XE7740 Intel® Xeon® 6747P CPUs 8x NVIDIA RTX PRO 6000 Blackwell GPUs	**Amazon Bedrock** Llama 3.1 Instruct (70B) (w/ latency optimized inference)
$/Mtok (50% utilization)	$0.09	$0.90
$/Mtok (100% utilization)	$0.05	$0.90
5-year cost (50% utilization)	$272,032	$2,590,212
5-year cost (100% utilization)	$280,618	$5,180,423

System configuration information

Table 4: Detailed information on the systems we tested.

Server configuration information	**Dell PowerEdge XE7740**
BIOS name and version	Dell 1.6.4 (11/02/2025)
Non-default BIOS settings	N/A
Operating system name and version/build number	Ubuntu 24.04.4 LTS
Date of last OS updates/patches applied	06/20/2026
Power management policy	Performance Optimized
Processor	
Number of processors	2
Vendor and model	Intel Xeon 6747P
Core count (per processor)	48
Core frequency (GHz)	2.7
Stepping	Model 173 Stepping 1
Memory module(s)	
Total memory in system (GB)	1,024
Number of memory modules	32
Vendor and model	Samsung® M321R4GA3EB2-CCPKF
Size (GB)	32
Type	DDR5 RDIMM
Speed (MT/s)	6,400
Speed running in the server (MT/s)	5,200

Server configuration information	Dell PowerEdge XE7740
Storage controller	
Vendor and model	Dell BOSS-N1 Marvell 88NR2241 (OS), direct-attached NVMe (data)
Cache size	N/A
Firmware version	2.2.13.2037
Driver version	In-kernel NVMe™ (Linux 6.8.0-117)
Local storage (OS)	
Number of drives	2 (RAID1)
Drive vendor and model	Micron® 7450
Drive size (GB)	480
Drive information (speed, interface, type)	PCIe Gen4 M.2 NVMe® SSD
Local storage (data)	
Number of drives	3
Drive vendor and model	Samsung PM9D3a
Drive size (GB)	1,920
Drive information (speed, interface, type)	PCIe® Gen5 NVMe SSD
Network adapter	
Vendor and model	Broadcom® BCM57414 2x25G OCP Ethernet NIC
Number and type of ports	2x SFP28 (10/25GbE)
Driver version	bnxt_en 6.8.0-117-generic
Accelerators	
Vendor and model	NVIDIA RTX™ PRO 6000 Blackwell Server Edition
Number	4 / 6 / 8
Memory type	GDDR7 with ECC
Memory size (GB)	96
Memory bandwidth (GB/s)	1,597
Power cap limit (W)	600
Connection	PCIe Gen 5 x16
Driver	580.142
Cooling fans	
Vendor and model	Dell Platinum
Number of cooling fans	20

Server configuration information	Dell PowerEdge XE7740
Power supplies	
Vendor and model	Dell 0J0KHHA01
Number of power supplies	8
Wattage of each (W)	3,200

How we tested

We developed a proprietary agentic AI benchmark framework that runs AI agents completing representative tasks in the manufacturing and financial services domains. The tasks are real-world; the data they reference is synthetic and generated by the benchmark framework.

Suite-level orchestration: Parallelism in this benchmark operates between workflow instances, never within them. Each instance executes its state graph sequentially over its own scenario data, and the harness sustains a fixed target concurrency by admitting instances from an ordered queue, a new instance admitted the moment one completes. The queue is ordered by workflow, so a run proceeds as overlapping waves: all scenarios of one workflow drain as the next workflow's instances are admitted behind them, holding the system at full concurrency through each transition. Every execution of a given workflow traverses an identical path, making runs directly comparable.

Manufacturing workflows

Our manufacturing suite has five telemetry workflows and three vision workflows.

The five telemetry workflows share a single orchestration pattern in which the model always acts last. The full engineering computation (control-charting, OEE roll-ups, genealogy tracing, vibration signal processing, trend statistics) runs to completion in-process before the benchmark invokes the LLM as the interpretive step. This creates a strict generate → compute → judge sequence with no interleaving.

The three vision workflows are nested prefixes of one four-stage QC pipeline (image inspection → telemetry correlation → disposition decision → report generation). Inspection runs stage one alone, QC + root cause the first two, and the full pipeline all four, so each successive workflow extends the same orchestrated sequence one stage further.

Table 5 offers details on the manufacturing workflows.

Table 5: The manufacturing tasks in our agentic AI benchmark.

Task name	What it exercises	How it stresses hardware
mfg_spc_analysis	Loads multi-series production telemetry (think: 50 sensors, each producing one reading per second for a week), runs the full SPC battery on every series—control chart limits, Cp/Cpk capability indices, Western Electric rule scans, CUSUM and EWMA change detection, normality tests—and asks the LLM to identify which parameters are out of control and what the likely root causes are.	Heavy CPU compute, mostly numpy/scipy.
mfg_oee_calculation	Simulates a production event log for a multi-line factory across a multi-week window (machine starts, stops, downtime reasons, output counts, scrap counts), rolls up per-line-per-shift OEE—the industry-standard metric defined as Availability x Performance x Quality—identifies the worst-performing lines, and explains the dominant loss drivers (changeover time, micro-stops, scrap rate).	Pandas roll-ups across event logs, then a small LLM call. CPU + memory bandwidth on big event-log volumes; stresses analytical pipelines more than LLM inference.
mfg_batch_genealogy	Constructs a real directed graph of a supply chain (component supplier to component lot to batch to shipment to customer), simulates a recall on a specific component lot, traces every downstream batch and customer that received material from the contaminated lot using BFS, and recommends a containment action.	Networkx graph operations on a directed acyclic graph that can grow into the millions of edges. Stresses CPU and graph-algorithm libraries; LLM call is small.

Task name	What it exercises	How it stresses hardware
mfg_predictive_maintenance	Generates synthetic vibration-sensor data for a fleet of rotating machinery (motors, pumps, gearboxes), runs the diagnostic battery—rolling RMS, rolling kurtosis, FFT band energies, two-state Kalman-style projection filter—on every channel, and asks the LLM to classify each machine's risk level and estimate Remaining Useful Life (RUL) in days.	Heavy DSP compute (FFT, rolling-window stats) on a fleet of synthetic time series. CPU + memory bandwidth dominant. The narrative LLM step is small. Good showcase for CPU performance.
mfg_defect_trend_investigation	Simulates daily defect counts split by (defect type × production line) across a multi-month window, runs the full trend-analysis battery (rolling stats, linear-regression slope, autocorrelation at lags 1/7/14 to detect weekly patterns, weekend-vs-weekday ratios, CUSUM change-point detection), then runs a multi-turn ReAct loop where the LLM investigates specific spikes by querying for related context and proposing root causes.	Statistics-heavy CPU compute, then a multi-turn LLM loop (3-8 turns typical). Mixed CPU + LLM-latency profile.
mfg_quality_inspection	The simplest of the three vision scenarios. Receives a production image (rendered from a synthetic 3D scene of a product on a conveyor), asks a vision-language model to determine pass/fail and identify any defect category visible in the image. One round trip per part.	Vision-language model inference. Stresses the dedicated vision GPU(s) more than the text GPUs. Use this scenario to demonstrate the value of having vision and text on separate GPU groups (the framework's "vision-heavy" topology mode).
mfg_quality_analysis	The previous scenario plus a second stage. After the vision model classifies a defect, the agent pulls the relevant production telemetry from the time window of that part's manufacture and asks the text LLM to correlate the defect with telemetry signals—was a temperature out of spec, did a tool wear into a known pattern, was a machine ramping back up after a changeover?	Mixed vision + text. Stresses the multi-instance topology, with vision running on its own GPU group while the text LLM works on a different group.
mfg_quality_full_pipeline	The most ambitious scenario in the suite. Four-stage pipeline: (1) vision inspection, (2) telemetry root-cause correlation, (3) disposition decision (quarantine? rework? continue?), (4) report generation including suggested process-engineering follow-up. Multiple round trips between vision model and text LLM with intermediate analytical steps.	The most demanding workload in the suite. Vision GPU + text GPU + CPU all engaged simultaneously. Best for showcasing dual-NUMA, multi-GPU topology benefits.

Financial services workflows

For our financial services workflows, the benchmark orchestrates each financial workflow as a compiled state graph with a fixed, deterministic node sequence. For most of the workflows, it stages scenario data from the firm's database first, then executes the workflow's analytical phase and invokes the LLM at consistent synthesis points.

There are three workflows with somewhat different flows. For the fraud investigation workflow, after a deterministic SQL triage surfaces candidate accounts, sequencing passes to the agent itself, which orders its own investigation, selecting lookup, scan, and rule-check tools in an iterative reasoning loop until it reaches an escalation decision. The filing Q&A and board-pack narrative are intentionally single-step graphs, isolating long-context ingestion and sustained long-form generation from multi-step orchestration effects.

Table 6 offers details on the financial services workflows.

Table 6: The financial services tasks in our agentic AI benchmark.

Task name	What it exercises	How it stresses hardware
filing_discrepancy	Reads a synthetic 10-K filing as a PDF, pulls key financial figures (revenue, net income, EPS, segment results), cross-references those figures against an authoritative database of the same company's reported numbers, and writes a structured risk summary that flags any mismatches with materiality grading.	PDF text extraction stresses CPU, the cross-reference query hits PostgreSQL, and the final structured-summary generation is an LLM call against the long extracted text. Typical of "intelligent document processing" pitches.
txn_reconcile	At the end of a banking day, replays every posted transaction across 500 branches and 50,000 accounts, computes what the closing balance should be for each account, compares against the ledger's actual posted balance, flags accounts that don't reconcile, and writes a per-exception narrative for the operations team.	Heavy SQL aggregation against a 1M-row transaction table on PostgreSQL, then a relatively short LLM call to narrate flagged exceptions. Stresses database I/O, CPU concurrency, and storage subsystem.
portfolio_valuation	For a given investment portfolio, loads every position the portfolio holds, joins each position against the most recent market-close price, computes the portfolio's net asset value (NAV), ranks the top three sector exposures by percent of NAV, and writes a brief explanatory note about NAV movement.	Pandas/numpy compute over 100k positions × 30 historical snapshots = 3 million rows, joined against 1.26M market-data rows. CPU + RAM are the constraint; LLM is a small narrate-the-result step at the end. Good "make NumPy go fast" demo.
reg_ratios	Loads a slice of bank balance-sheet inputs (capital, assets, liabilities, off-balance-sheet exposures), computes the regulatory ratios that bank regulators care about—Basel capital adequacy, Liquidity Coverage Ratio (LCR), Net Stable Funding Ratio (NSFR), leverage ratio, debt-service coverage—and flags any ratio that breaches its threshold.	Small structured-data pull, modest CPU compute, then an LLM call to explain breaches in plain English. Less compute-intensive than the other financial scenarios—useful for showing baseline LLM throughput on small inputs.
peer_olap	Picks a target ticker, finds every other publicly traded company in the same industry sector, computes trailing-window returns and volatility for the cohort using SQL window functions, ranks the target against its peers, and writes a sector-commentary paragraph that names the leaders and laggards.	SQL window functions over 1.26M market-data rows—heavy on database CPU and the analytical query optimizer. The LLM narration is small. Good for demonstrating analytical-database performance.

Task name	What it exercises	How it stresses hardware
fraud_investigation	Scans a week of bank transactions for three classic fraud patterns (rapid-fire same-account activity, suspiciously round-number debits, and large rapid debits), surfaces 10-30 candidate accounts, then runs a multi-turn investigation loop where the AI pulls account history, runs additional rule checks, and decides which accounts to escalate to a human investigator with written justification.	The SQL pre-filter is database-heavy; the multi-turn investigation is the most LLM-intensive scenario in the suite (six to ten back-and-forth turns per investigation). Stresses LLM inference latency more than throughput—investigators want fast turns.
filing_qa	Loads ten full synthetic 10-K filings into a single context window, takes a batch of ten questions about specific numbers in those filings (one question per filing—"what was Apex Data Group's 2023 revenue in USD millions?"), and answers all ten in one LLM call.	Long-context LLM inference. Stresses model memory (KV cache) and compute together—the larger the context, the more the GPU has to hold and process. Excellent demo of why you want more VRAM and faster GPUs for context-heavy workloads.
board_pack_narrative	Takes a structured "facts table" for one company-quarter (revenue, net income, segment results, headline risks, forward outlook) and generates a five-section board pack: executive summary, revenue and earnings, segment results, key risks, and outlook. Each section has a length budget (80-400 words) and a list of must-mention keywords.	Longest single LLM generation in the suite—5 sections × up to 400 words each is roughly 3,000 output tokens per pack. Stresses output throughput (tokens/second). Good demo for "we want to write quarterly content at scale."

This benchmark is proprietary, and as such, we are not releasing its full code. We would be happy to discuss any questions you have. Please email info@principledtechnologies.com for more information.

Software versions we used

Table 7 covers the main software components we used in testing.

Table 7: Software versions we used in testing.

Component	Version
Operating system	Ubuntu 24.04.4 LTS (kernel 6.8.0-117-generic)
NVIDIA driver	580.142
CUDA	13.02 serving runtime
Container runtime	Docker 29.4.0
Python	3.11.15
Inference servers	vLLM 0.20.0
Agent framework	LangGraph 1.2.6; LangChain 1.3.11 (langchain-core 1.4.8)
Benchmark tooling	AIPerf 0.7.0
Database	PostgreSQL 16.13
LLM (text inference)	Llama-3.1-70B-Instruct (FP8)
VLM (vision inference)	Qwen3.5-2B

GPU topology

We used 2 GPUs per NUMA node for the 4 GPU configuration, 4 GPUs per node for the 8 GPU config. For the 6-GPU configuration, we paired 2 GPUs within the first node, 2 within the second, and a final 2 that spanned both nodes (one GPU on each).

This project was commissioned by Dell Technologies.

Read the report ▶

Primary contributors

Tech: Warren S., Chris B.

Writing: Sarah Van Name

Design: Jared White

PM: Sarah Van Name

How we created this report

A PT team, which includes the contributors we've listed and others, created this report and performed the technical work behind it. In addition to our use of AI in testing, we used AI to draft and proofread sections of this report.

Facts matter.®

Principled Technologies is a registered trademark of Principled Technologies, Inc. All other product names are the trademarks of their respective owners. For additional information, review the science behind this report.

DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY:

Principled Technologies, Inc. has made reasonable efforts to ensure the accuracy and validity of its testing, however, Principled Technologies, Inc. specifically disclaims any warranty, expressed or implied, relating to the test results and analysis, their accuracy, completeness or quality, including any implied warranty of fitness for any particular purpose. All persons or entities relying on the results of any testing do so at their own risk, and agree that Principled Technologies, Inc., its employees and its subcontractors shall have no liability whatsoever from any claim of loss or damage on account of any alleged error or defect in any testing procedure or result.

In no event shall Principled Technologies, Inc. be liable for indirect, special, incidental, or consequential damages in connection with its testing, even if advised of the possibility of such damages. In no event shall Principled Technologies, Inc.'s liability, including for direct damages, exceed the amounts paid in connection with Principled Technologies, Inc.'s testing. Customer's sole and exclusive remedies are as set forth herein.