GPU-accelerated strategy search: a measured comparison with CPU

Valynox Research (AlphaPrime by Valynox) · 1 November 2026 · Valynox Research white paper no. 1
https://alphaprime.trading/en/research/gpu-vs-cpu-strategy-search/

Abstract. We measure how fast one consumer GPU and one desktop CPU generate and backtest rule-based trading strategies when everything else is held fixed: the same software build, the same one-minute data, the same pool of entry and exit conditions, the same random seed and the same batch size. On an NVIDIA RTX 3090 the search completed 6,920 to 10,054 candidate strategies per second across an equity-index future (@NQ), a forex future (@EU) and a crypto future (@BTC), each covering about 12 weeks of one-minute bars. On an AMD Ryzen 9 9950X3D using all 32 threads, the same work ran at 2,978 to 4,100 per second, so the GPU was 2.3× to 2.5× faster at an identical batch of 16,384 candidates and 4.1× to 4.4× faster when each engine picked its own batch size (11,716 to 17,124 per second on the GPU). Each figure is the median of three steady-state measurements of 30 seconds, with GPU utilisation sampled every second. Selecting the full catalogue of 106,460 conditions instead of a 1,500-condition sample changed the steady-state rate by −4.3% on the GPU and +8.7% on the CPU, while the time to the first batch roughly doubled. We report the method, every run, the per-second timelines and the steps needed to repeat the measurements.

Keywords: backtesting; GPU computing; strategy generation; data mining; algorithmic trading; benchmark; CUDA; futures; forex; crypto. **JEL:** C63, C88, G17, C52.

1. Introduction

Rule-based strategy search builds a large number of candidate trading strategies from a menu of entry and exit conditions and backtests each one on historical data. The number of candidates a researcher can test in a working day sets how much of the rule space gets explored and how much room is left for out-of-sample and robustness checks afterwards. Most published speed figures for backtesting software count different units (bars, backtests, strategies, passes) and seldom state the hardware, the data length or the test size behind them.

This paper answers one narrow question with a controlled measurement: on one desktop machine, how many candidate strategies per second does the same search complete on the graphics card and on the processor? Only the compute engine changes between the two arms. We also check whether the size of the condition pool changes the speed. Strategy validation (out-of-sample tests, walk-forward analysis, Monte Carlo checks) is outside the scope and is switched off in every run; it adds time on top of the figures reported here.

2. Method

2.1 What is measured

A *candidate* is one generated strategy (entry conditions, exit conditions, stop-loss and profit target) together with one complete backtest of it over the whole data window, including the first-pass screen applied to every backtest. The software works in *batches*: each batch is one block of candidates computed

together, and the engine starts the next batch as soon as one finishes. The search draws conditions at random, with replacement, from the condition pool the user selected; this pool is a sampling pool, so the engine prepares indicator series for a random subset of it rather than for every selected condition at start-up.

The speed figure is the steady-state rate: the change in the software's own count of evaluated candidates between the first and the last batch boundary seen within a 30-second window, divided by the time between those two boundaries. The window opens at the first completed batch. Start-up, data loading, indicator precomputation and the storing of results after the stop are therefore excluded. The status of the run was polled every 250 ms. Microseconds per candidate is the inverse of the rate. This is the same definition the software's own release check uses for its speed floor of 10,000 candidates per second on the RTX 3090.

2.2 Hardware and software

- GPU: NVIDIA GeForce RTX 3090, 24 GB, driver 610.88, power limit 370 W, PCIe 4.0 x16.
- CPU: AMD Ryzen 9 9950X3D, 16 cores / 32 threads; 64 GB DDR5 at 5,200 MT/s.
- Operating system: Windows 11 Pro, build 26200.
- Software: AlphaPrime by Valynox 1.16.4212 (build g211148156f), the same executable for both arms. The CPU arm ran with the GPU hidden from the process (`CUDA_VISIBLE_DEVICES=-1`) and the CPU engine selected; one second after each CPU run started we confirmed that the process held no GPU context.

2.3 Data

One-minute trade bars of three exchange-traded front-month continuous futures, unadjusted, from 7 July 2025 to 26 September 2025, in exchange time (America/New_York): @NQ, E-mini Nasdaq-100 futures (CME), 82,429 bars; @EU, Euro FX futures, EUR/USD (CME), 81,024 bars; @BTC, Bitcoin futures (CME), 58,228 bars. The three cover an equity index, a currency pair and a cryptocurrency. The last four weeks (29 August to 26 September 2025) were marked as the out-of-sample range, as a user would set them; out-of-sample results are computed in the same backtest. Bitcoin futures have fewer one-minute bars over the same dates because minutes without a trade produce no bar.

2.4 Strategy space and condition pool

Every run used random generation of simple strategies: exactly one entry condition and zero or one exit condition, indicator periods drawn from 2 to 60, bar shifts from 0 to 2, a required stop-loss (8 to 60 ticks fixed, or 1 to 3 ATR with an ATR period of 14 to 20) and a required profit target (8 to 80 ticks, or 1 to 4 ATR), with exits by rule, stop-loss or profit target only. Trading was allowed around the clock with no session filter. The main experiments draw entry and exit conditions from a pool of 250 classic indicator conditions (moving averages, oscillators, volatility and price-pattern conditions). The pool-size experiment uses the software's full catalogue of 106,460 selectable conditions and a 1,500-condition sample of it drawn with a fixed seed (20261101), so both pools have the same mix of condition types.

2.5 Sampling and batches

Both arms use the same random seed (20261021), so they draw from the same sequence of candidates. In the identical-batch experiment (S1, and S3 for the pool size) both engines were given a declared batch of 16,384 candidates. In the default experiment (S2) each engine chose its own batch size, as it does for a user:

the GPU engine grows its batch during the first seconds of a run, and the CPU engine publishes progress in smaller blocks spread across its worker threads.

2.6 Cost assumptions

Each backtest charged a fixed commission of USD 20 per trade on one contract (fixed size, starting capital USD 10,000), with no additional slippage setting. The cost model changes the profit and loss of each trade, not the amount of computation, so it has no effect on the speed figures; it is stated so the workload can be repeated exactly.

2.7 Protocol

- Three settings: S1 identical batch (3 markets × 2 engines), S2 engine default batch (3 × 2), S3 pool size (@NQ, 2 pools × 2 engines). Each combination was run three times; we report the median. 48 of 48 runs were valid.
- Runs were measured one at a time and interleaved (repeat → market → engine, with the engine order alternating between repeats) so slow drift in temperature or background load does not fall on one arm.
- Each run starts a fresh host process, measures a 30-second steady-state window and stops; no run, start-up included, took longer than 3 minutes.
- Before each run we checked that the GPU was idle (median utilisation below 50% over 3 seconds and no other generation run active) and took the machine's GPU measurement lock for the duration of the run only.
- GPU utilisation, memory in use, power and SM clock were sampled once per second with `nvidia-smi`; whole-machine CPU busy time and the host process's working set once per second.
- A run counts as valid only if it ran on the intended engine, never waited for the GPU lock and completed at least three batch boundaries in the window.

3. Results

3.1 GPU versus CPU at an identical batch

With both engines computing batches of 16,384 candidates, the RTX 3090 completed 6,920 on @NQ, 7,474 on @EU, 10,054 on @BTC candidates per second; the CPU completed 3,003, 2,978, 4,100 respectively. The GPU was 2.3× to 2.5× faster (Table 1, Figure 1). On @NQ that is 144.5 µs per candidate on the GPU against 333.0 µs on the CPU.

The differences between markets follow the number of bars. Multiplying each rate by the bars in its window gives 570, 606, 585 million bar-candidates per second on the GPU and 248, 241, 239 million on the CPU for @NQ, @EU, @BTC: @BTC is faster per candidate because its window has about 30% fewer bars, not because the asset class is cheaper to compute. The three GPU repeats of each setting were within 2.4% of each other; the CPU repeats varied by up to 32%, which is why we report medians.

Table 1. Steady-state speed at an identical declared batch of 16,384 candidates (median of 3 runs each).

Market	Bars	GPU candidates/s	GPU µs/candidate	CPU candidates/s	CPU µs/candidate	GPU ÷ CPU
@NQ · Equity-index futures	82,429	6,920	144.5	3,003	333.0	2.3×
@EU · Forex	81,024	7,474	133.8	2,978	335.8	2.5×
@BTC · Crypto	58,228	10,054	99.5	4,100	243.9	2.5×

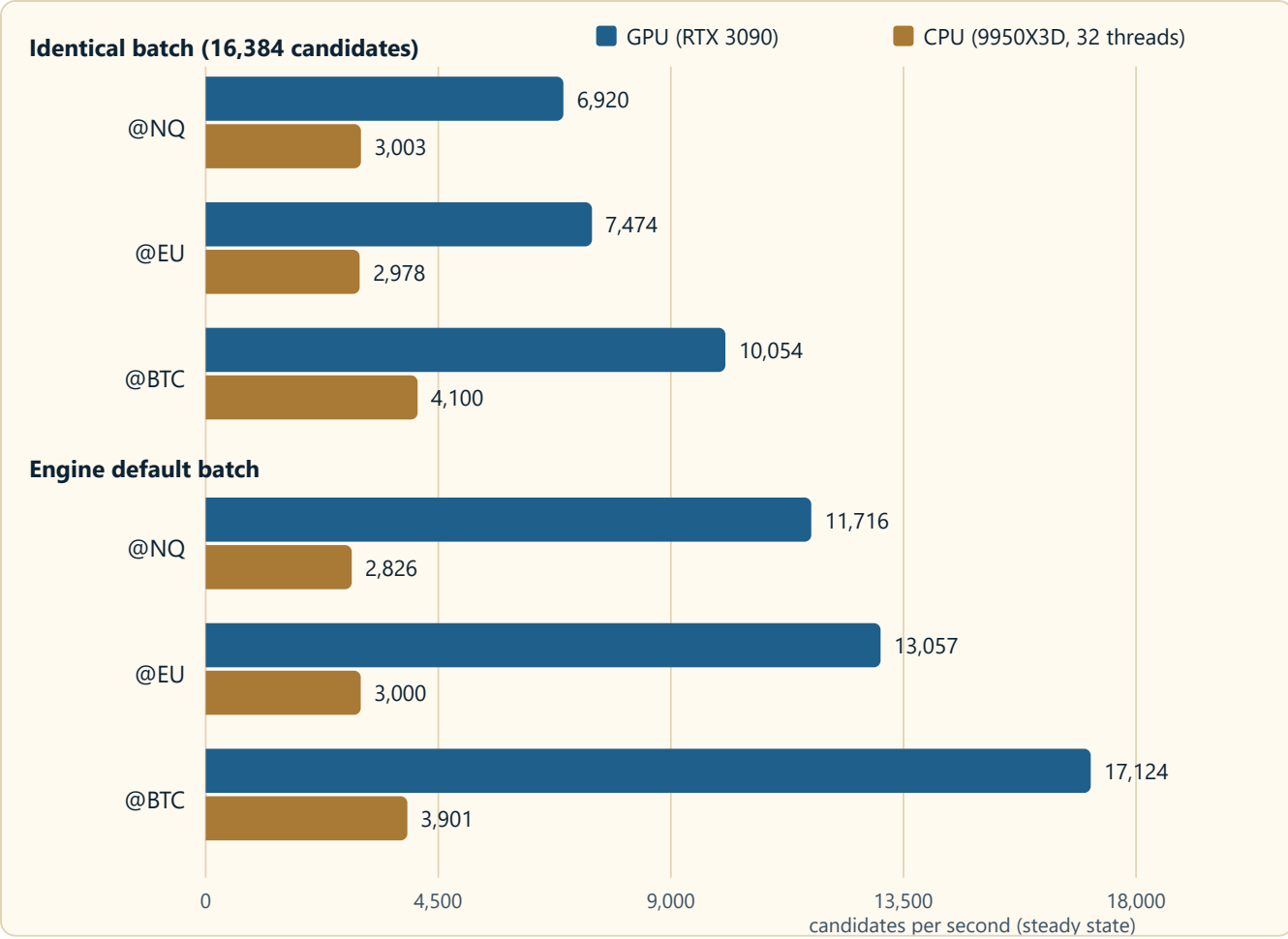


Figure 1. Steady-state candidates per second (median of three 30-second measurements) on each market. Top: both engines run the identical declared batch of 16,384 candidates. Bottom: each engine uses its default batch size. GPU: NVIDIA RTX 3090. CPU: AMD Ryzen 9 9950X3D, 32 threads.

3.2 Engine default batch

Left to choose its own batch size, the GPU engine grew its batches over the first seconds and settled at 20,992 candidates per batch; the CPU engine reported progress in blocks of 1,024 to 2,048. The GPU reached 11,716, 13,057, 17,124 candidates per second on @NQ, @EU, @BTC and the CPU 2,826, 3,000, 3,901, a ratio of 4.1× to 4.4× (Table 2). On @NQ the default setting was 69% above the declared-batch rate on the GPU, while the CPU changed by −5.9%. A batch about 28% larger does not by itself account for a gain of this size; the declared batch is a control for the comparison, not the setting a user runs, and we did not separate the causes further. The default-batch figures are the ones a user of the software sees.

Table 2. Steady-state speed with the batch size each engine chooses by default (median of 3 runs each).

Market	GPU candidates/s	GPU typical batch	CPU candidates/s	CPU typical batch	GPU ÷ CPU
@NQ · Equity-index futures	11,716	20,992	2,826	1,024	4.1×
@EU · Forex	13,057	20,992	3,000	1,024	4.4×
@BTC · Crypto	17,124	20,992	3,901	2,048	4.4×

3.3 GPU utilisation over time

Figure 2 shows the per-second GPU utilisation of the median @NQ run of each GPU setting. In the identical-batch run the 10th percentile of utilisation within the window was 99% and the median 99%. With the default batch the 10th percentile was 87% and the median 99%: the low samples fall in the first seconds after the first batch, while the batch size is still growing, and the card stays near full load after that. One-second sampling cannot resolve gaps shorter than a second, so a reading of 99% means the card was busy for most of that second, not that every core was occupied.

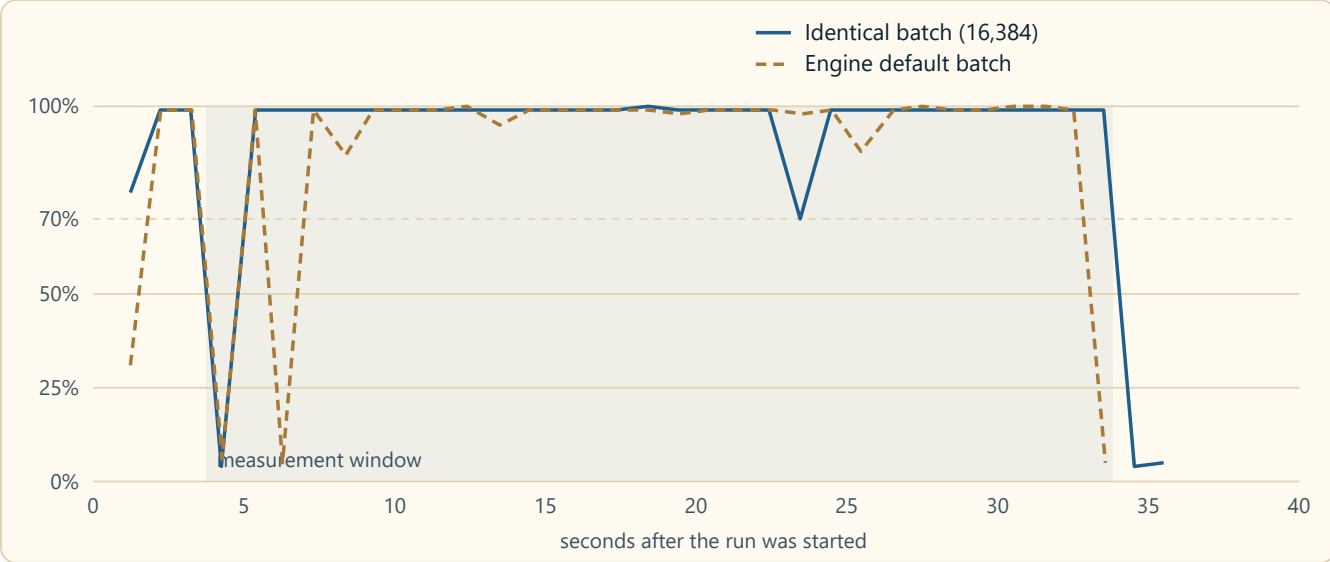


Figure 2. GPU utilisation sampled once per second, @NQ, for the median run of each GPU setting. The shaded band is the measurement window of the identical-batch run (first batch to end of window); the dashed horizontal line marks 70%. The idle reading of this card with the desktop and remote-display software running was 5–22%.

3.4 Memory and processor load

During the GPU runs the processor was not idle: preparing indicator series for the next batches and screening results kept the whole machine at a median of 34% CPU busy, against 100% in the CPU runs (Table 3). The GPU run used about 4,341 MiB more graphics memory than the idle desktop (3,816 MiB) and a larger host working set, mainly for the device-side copy of the data and the precomputed indicator series. Neither engine came close to the 24 GB of the card or the 64 GB of the machine.

Table 3. Resource use during the measurement window, @NQ, identical batch (median of 3 runs; GPU utilisation sampled once per second).

Engine	GPU utilisation p10	GPU utilisation p50	Whole-machine CPU busy (p50)	Peak host working set	Peak GPU memory in use
GPU	99%	99%	34%	1,136 MiB	8,157 MiB
CPU	3%	4%	100%	612 MiB	3,869 MiB

3.5 Size of the condition pool

Selecting all 106,460 conditions instead of a 1,500-condition sample of the same catalogue changed the steady-state rate by −4.3% on the GPU and +8.7% on the CPU (Table 4): small changes next to the 70-fold difference in pool size, and on the CPU within its run-to-run spread. Because the pool is sampled, the engine prepares indicator series for a random subset of the selection and never for all of it at once. The selection size shows elsewhere: the time from start to first batch rose from 5.5 s to 11.1 s on the GPU and from 6.7 s to 12.9 s on the CPU, and with the full catalogue the GPU spent 12.9% of its one-second samples below 50% utilisation (10th percentile 7%), waiting for series to be prepared, against 0.0% with 1,500 conditions. Both catalogue pools ran slower on the GPU than the classic 250-condition pool (5,435 against 6,920 per second), most likely because the catalogue contains computationally heavier indicators; on the CPU the three pools ran at similar rates.

Table 4. Condition pool size, @NQ, identical batch: a fixed-seed sample of 1,500 conditions versus the full catalogue of 106,460 (median of 3 runs).

Engine	Pool	Candidates/s	Start-up to first batch	Peak host working set	Change vs 1,500
GPU	1,500	5,435	5.5 s	4,027 MiB	—
GPU	106,460	5,202	11.1 s	3,594 MiB	−4.3%
CPU	1,500	3,058	6.7 s	4,876 MiB	—
CPU	106,460	3,323	12.9 s	3,458 MiB	+8.7%

4. What this means for a fixed research budget

At the default-batch rates on @NQ, the settings a user runs, one hour of search covers about 42.2 million candidates on the GPU and 10.2 million on the CPU; an eight-hour working day covers about 337 million and 81 million. Put the other way, a search the CPU finishes overnight in 12 hours takes the GPU about 2.9 hours (5.2 hours at the identical batch). For a fixed amount of research time, the faster engine buys either a larger share of the rule space or more time for the validation steps that decide which strategies deserve trust: out-of-sample testing, walk-forward analysis and robustness checks, which are not part of these figures.

The two rates do not simply add up on one machine: during the GPU runs about a third of the processor was already busy preparing the next batches, so a CPU search run alongside would share those cores with the GPU search.

5. Limitations

- One machine, one GPU model and one CPU model. Other cards, processors, drivers and memory configurations give other figures; GPU speed in particular scales with the card's compute units and memory bandwidth.
- Simple strategies (one entry condition, at most one exit condition). Strategies with more conditions take longer per candidate on both engines, and the ratio between the engines may differ.
- About 12 weeks of one-minute data per market (58,000 to 82,000 bars). Longer histories increase the time per candidate roughly in proportion to the number of bars on both engines; we did not measure how the ratio changes with history length.
- Thirty-second steady-state windows. Long runs can be affected by thermal limits and by background work on the machine; the measurement windows were kept short by design, and the per-second timelines are published so drift within a window can be checked.
- The machine was a working desktop with display and remote-access software running; the idle GPU utilisation reading was 5–22% before the runs started, which slightly overstates the GPU utilisation figures but does not enter the speed figures.
- Strategy validation was switched off. With validation on, both engines spend additional time per strategy that passes the first screen.
- The forex and crypto markets are represented by exchange-traded futures (Euro FX and Bitcoin futures), which share the bar structure of the equity-index future; spot forex or crypto data with a different bar count per day would change the per-candidate time in proportion to the number of bars.
- The software was measured by its developer. To make an independent check possible, every run, the timelines and the settings are published with this paper.

6. Reproducing the measurements

1. Install AlphaPrime by Valynox (the 7-day trial runs at full GPU speed) on a Windows machine with an NVIDIA GPU. Note the version shown under About; these results are for 1.16.4212 (build g211148156f).
2. Import one-minute bars for the three markets, or any market you trade, and select 7 July 2025 to 26 September 2025 with 29 August to 26 September 2025 as the out-of-sample range.
3. Set the generation options listed in `workload.json` (data files): random generation, one entry condition, zero to one exit condition, the stop-loss and profit-target ranges in section 2.4, strategy validation and walk-forward analysis off, fixed seed 20261021, and a time limit of 3 minutes.
4. Select the classic indicator conditions (or record which pool you use; Table 4 shows how the pool size affects the figure). For the CPU arm, choose the CPU engine in the compute settings.
5. Start the run and read the evaluated-candidate counter at two batch boundaries at least 25 seconds apart, starting after the first batch; divide the difference by the elapsed time. Repeat three times per setting and take the median. Sampling `nvidia-smi --query-gpu=utilization.gpu --format=csv -lms 1000` alongside gives the utilisation timeline.

Our measurements drove the same host process programmatically, with the status polled every 250 ms, so the batch boundaries are timed more precisely than by reading the screen; the per-change counter readings of every run are in `evaluated_points.csv`.

Appendix A. Every run

The table lists every run in measurement order. Run names read engine, market, pool, declared batch (b16384, or none for the engine default) and repeat number. The data files hold the same rows with additional columns, the per-second GPU and CPU timelines of every run and the counter readings behind each speed figure.

Table A1. Every run, in the order it was measured (steady-state candidates per second; S1 identical batch, S2 engine default, S3 pool size).

Set	Run	Valid	Candidates/s	µs/candidate	Start-up (s)	GPU p10	CPU busy p50
S1	gpu-NQ-classic250-b16384-r1	true	6,992	143.0	3.82	97%	33.5%
S1	cpu-NQ-classic250-b16384-r1	true	3,003	333.0	5.52	4%	100.0%
S1	gpu-EU-classic250-b16384-r1	true	7,499	133.3	3.44	98%	29.8%
S1	cpu-EU-classic250-b16384-r1	true	2,978	335.8	5.05	3%	100.0%
S1	gpu-BTC-classic250-b16384-r1	true	9,886	101.2	2.78	88%	27.7%
S1	cpu-BTC-classic250-b16384-r1	true	4,101	243.9	3.63	3%	100.0%
S2	gpu-NQ-classic250-r1	true	11,472	87.2	2.63	98%	30.3%
S2	cpu-NQ-classic250-r1	true	2,773	360.6	2.39	4%	91.3%
S2	gpu-EU-classic250-r1	true	13,057	76.6	2.32	99%	33.0%
S2	cpu-EU-classic250-r1	true	2,878	347.4	2.68	3%	91.2%
S2	gpu-BTC-classic250-r1	true	17,124	58.4	1.77	99%	31.6%
S2	cpu-BTC-classic250-r1	true	3,901	256.3	1.85	4%	88.9%
S3	gpu-NQ-n1500-b16384-r1	true	5,429	184.2	5.46	99%	36.9%
S3	cpu-NQ-n1500-b16384-r1	true	3,068	325.9	6.5	2%	100.0%
S3	gpu-NQ-all-b16384-r1	true	5,202	192.2	10.47	7%	40.4%
S3	cpu-NQ-all-b16384-r1	true	2,889	346.1	9.46	3%	98.5%
S1	cpu-NQ-classic250-b16384-r2	true	2,683	372.8	5.25	3%	100.0%
S1	gpu-NQ-classic250-b16384-r2	true	6,920	144.5	3.75	99%	27.5%
S1	cpu-EU-classic250-b16384-r2	true	3,435	291.1	4.9	3%	100.0%
S1	gpu-EU-classic250-b16384-r2	true	7,473	133.8	3.43	99%	31.4%
S1	cpu-BTC-classic250-b16384-r2	true	3,708	269.7	3.51	3%	100.0%
S1	gpu-BTC-classic250-b16384-r2	true	10,124	98.8	2.74	99%	27.1%
S2	cpu-NQ-classic250-r2	true	2,963	337.5	1.89	2%	89.8%
S2	gpu-NQ-classic250-r2	true	11,716	85.4	2.24	87%	27.8%
S2	cpu-EU-classic250-r2	true	3,000	333.4	2.55	3%	89.7%
S2	gpu-EU-classic250-r2	true	13,036	76.7	2.27	76%	32.4%
S2	cpu-BTC-classic250-r2	true	3,866	258.7	2.6	3%	91.9%
S2	gpu-BTC-classic250-r2	true	16,974	58.9	1.86	94%	40.6%
S3	cpu-NQ-n1500-b16384-r2	true	3,022	330.9	6.66	3%	100.0%
S3	gpu-NQ-n1500-b16384-r2	true	5,435	184.0	5.76	98%	47.1%

Set	Run	Valid	Candidates/s	µs/candidate	Start-up (s)	GPU p10	CPU busy p50
S3	cpu-NQ-all-b16384-r2	true	3,453	289.6	13.53	3%	98.7%
S3	gpu-NQ-all-b16384-r2	true	5,497	181.9	11.05	70%	51.0%
S1	gpu-NQ-classic250-b16384-r3	true	6,824	146.6	3.79	99%	36.5%
S1	cpu-NQ-classic250-b16384-r3	true	3,647	274.2	9.82	3%	100.0%
S1	gpu-EU-classic250-b16384-r3	true	7,474	133.8	3.66	99%	32.5%
S1	cpu-EU-classic250-b16384-r3	true	2,729	366.4	4.99	3%	100.0%
S1	gpu-BTC-classic250-b16384-r3	true	10,054	99.5	2.79	96%	30.3%
S1	cpu-BTC-classic250-b16384-r3	true	4,100	243.9	3.48	2%	99.8%
S2	gpu-NQ-classic250-r3	true	11,765	85.0	2.39	78%	37.9%
S2	cpu-NQ-classic250-r3	true	2,826	353.9	1.88	3%	93.0%
S2	gpu-EU-classic250-r3	true	13,175	75.9	2.34	77%	35.0%
S2	cpu-EU-classic250-r3	true	3,039	329.0	2.64	3%	90.1%
S2	gpu-BTC-classic250-r3	true	17,445	57.3	1.77	84%	33.5%
S2	cpu-BTC-classic250-r3	true	3,998	250.1	1.86	3%	87.9%
S3	gpu-NQ-n1500-b16384-r3	true	5,436	183.9	5.47	84%	35.0%
S3	cpu-NQ-n1500-b16384-r3	true	3,058	327.0	6.74	2%	100.0%
S3	gpu-NQ-all-b16384-r3	true	5,197	192.4	11.28	6%	48.1%
S3	cpu-NQ-all-b16384-r3	true	3,323	300.9	12.87	3%	93.9%

Appendix B. Data files

All files are released under CC BY 4.0 and sit next to this paper at `https://alphaprime.trading/en/research/gpu-vs-cpu-strategy-search/data/`. Times in the timeline files are seconds after the run was started.

- summary.csv: medians of the three repeats for every setting.
- summary.json: the same, with every repeat and the software and machine details.
- runs.csv: one row per run, in the order measured.
- gpu_timelines.csv: GPU utilisation, memory, power and clock, once per second, every run.
- cpu_timelines.csv: whole-machine CPU busy percentage and host working set, once per second, every run.
- evaluated_points.csv: the evaluated-candidate counter at every change, every run (the input of the speed figures).
- timelines.json: the two runs drawn in Figure 2.
- workload.json: the generation settings used for every run.

This paper reports software speed on stated hardware. It does not report or imply trading results; past or simulated performance does not indicate future results. Data files: https://alphaprime.trading/en/research/gpu-vs-cpu-strategy-search/data/