3-GPU Mesh — Hardware & Benchmarks

Status: 3-node mesh live on port 8320 (BEND)
Wire substrate: lumbda's `bend` primitive (lumbda.com)
Coordinator pattern: cost-routed fan-out via greedy bin-pack

What this page tracks

Three Nvidia GPUs across three hosts on our foxhop LAN serve as a bend mesh — lumbda's CUDA dispatch primitive routes work to whichever node finishes fastest. This page documents our hardware, our published benchmarks, & our cost-routed coordinator pattern.

For our ecdsa research context that drove our first 18-cell bench (refined Bernstein-Yang point-add variant sweep at p=251), see foxhop secp256k1 lever sweep.

Hardware

host	GPU	VRAM	arch
`3090-ai.foxhop.net`	RTX 3090	24 GB	sm_86
`ai.foxhop.net`	RTX 4090	24 GB	sm_89
`cammy.foxhop.net`	Tesla P40	24 GB	sm_61

72 GB combined VRAM across our pool. Pascal (sm_61) P40 lacks our newer architectures' reduced-precision tensor units, so it runs ~2× slower than Ampere 3090 on identical workloads — useful as a third concurrent stream rather than a faster replacement for either Ampere card.

Port 8320 — BEND

Our pool serves bend dispatches on port **8320** across every host. Mnemonic — 8320 spells BEND:

```
8 ~= B (implied infinity B flattened; bake a cake; baby & me)
3 ~= E (backward)
2 ~= N (pivoted 90 degrees)
0 ~= D (flattened)
```

Each node runs `gpu-worker.lsp` (lumbda's reference CUDA worker) against demo_ops, blake3-fanout, radix-sort, & our other registered CUDA forms. Clients reach our mesh via lumbda's `bend` primitive over TCP, S-expression wire format, & per-form binary protocols (`BSHK` for shake fanout, `BSCP` for batched scalar mul, `BSRT` for radix sort, etc. — see lumbda.com/bend for our form catalog). Our bend wire stays LAN-only; we do **not** expose our mesh outside our foxhop network.

Benchmark — 6 variants × 3 hosts at p=251

Our first published benchmark dispatched 6 ecdsa point-add variants (Bernstein-Yang lever sweep) at p=251 to every host through our wire protocol. Σ Toffoli matches **byte-for-byte** across hosts on identical ops.bin inputs — proves our distributed mesh runs from one canonical lumbda environment.

GPU wall (ms/batch) at 1024 shots, lower wins:

variant	Σ Tof / shot	3090 ms	4090 ms	P40 ms
v-fermat - schoolbook	167 984	290.1	149.2	531.7
v-fer mat-solinas	84 208	151.4	86.3	288.6
v-by-text - schoolbook	45 104	68.4	39.3	131.8
v-by-t ext-solinas	40 176	60.6	35.6	119.3
v-by-ref - schoolbook	29 104	32.5	25.7	85.1
v-by- ref-solinas	24 176	26.5	22.0	64.0

Per-architecture pattern reads cleanly:

- **Ampere 4090 (sm_89)** runs 1.94× faster than 3090 (sm_86) on our heaviest circuit, shrinking to 1.20× on our lightest. Larger circuits amortize kernel-launch overhead; smaller circuits hit our GPU's overhead floor.
- **Pascal P40 (sm_61)** runs ~2× slower than 3090 across our entire variant chain. Older arch, lower base clock, no reduced-precision tensor units; still serves a third concurrent stream as our candidate queue deepens.

GPU wall scales linearly with Σ Toffoli — no kernel-level surprise. Cross-host Σ Toffoli identity confirms bit-exact reproducibility across our pool.

Coordinator fan-out

A cost-routed coordinator dispatches candidates across our 3-node mesh concurrently. Greedy bin-pack by predicted Σ Toffoli; each candidate lands on whichever node's finish time after add stays lowest. Per-host speed factor derived from our bench above:

- 3090: factor 1.00 (baseline)
- 4090: factor 0.51
- P40: factor 1.84

One worker thread per node fires concurrently against our wire protocol. Each node serializes its own queue (workers do not multiplex requests cleanly inside one CUDA context).

First-run finding: cost model based only on (predicted Σ Toffoli × host factor) misses our per-request overhead floor (~600 ms on cammy for demo_ops process spawn + Pascal CUDA init). Refinement options: a per-host startup constant plus a Σ Toffoli linear term, fit per node from our sweep history. Coordinator runs correctly today; schedule shape needs calibration, not algorithmic change.

Pool capacity math

Coordinator at saturation. Three nodes scoring three different candidates simultaneously closes our sequential 6-variant 3090 wall (629 ms) into a parallel one. Refined-Solinas runs in ~25 ms on 3090 + 22 ms on 4090 + 64 ms on P40 — P40 lags but contributes a third concurrent stream as our candidate queue deepens. Per-host routing (heavy circuits to faster GPUs, light circuits to P40) maximizes aggregate dispatch.

Pool throughput grows with our candidate generation rate — see our ecdsa research page's "Why CPU Produces & GPU Scores" section for our substrate-level math on candidate-emit vs candidate-score throughput.

Related pages

- foxhop secp256k1 lever sweep — research context; full 18-cell cross-scale Pareto sweep; lumbda emit pipeline lift.
- lumbda.com/bend — bend primitive catalog; wire protocol; CUDA form list.

Updated 2026-06-07.

Source: https://foxhop.net/053ba7da-6277-11f1-82fc-040140774501/gpu-mesh
Snapshot: 2026-06-07T17:46:39Z
Generator: Remarkbox `50b9d1e`