Bulk Compute Prospectus

Shared Open-Weight Inference Capacity, 9-to-5 SLO

Shared Rented GPU Capacity for Open-Weight Models

A costed proposal for pooling a bulk purchase of rented GPU capacity to serve open-weight language models at **full native context** with real concurrency, on a 9-to-5 weekday duty cycle. Catamaran supplies the engineering: images, quantised checkpoints, deployment manifests, model hosting, routing and observability. Contributors supply share of card rent and receive concurrency slots.

Field	Value
Duty cycle priced	9-to-5, Mon-Fri = **173.3 hours/month** (23.7% of always-on)
Reference card	NVIDIA RTX PRO 6000 Blackwell, 96 GB GDDR7
Price floor used	**$0.66/GPU-hr** on-demand (packet.ai), read 2026-09-08
Market median	$2.20/GPU-hr on-demand across 42 tracked providers
Serving unit	**G2** = a tensor-parallel pair = 176 GB = **17 concurrent sessions**, replicated behind the router
Headline slot price	**$13.46**/session/month on shared capacity; **$35.18** on the 99.9% SLA tier
Baseline being displaced	Claude Max 20x - $200/mo for ~1.5 sustained concurrent

All prices are marketplace snapshots, not contracts. GPU rental moves daily. Every figure in this document is either a published vendor rate or an explicitly stated calculation from one; assumptions are labelled where they occur.

Executive Summary

Renting GPU capacity and serving open weights ourselves is between 4x and 15x cheaper per concurrent agent session than any per-seat subscription available today, and the advantage widens with scale because model weights amortise across sessions while subscriptions do not.

> **A tensor-parallel pair of cards at $228.76/month serves 17 simultaneous sessions of qwen38-flash-next at its full 262,144-token context. That is $13.46 per session per month, against $133 for the equivalent on a Claude Max 20x subscription. Twelve cards - six such pairs - cover ten contributors at ten sessions each.**

The six findings that drive everything

- **Context, not parameters, sets the price.** A model's weights are paid once; its KV cache is paid per concurrent session. Halving context roughly doubles session density on identical hardware.
- **qwen38-flash-next is the outlier.** Its hybrid Gated-DeltaNet / QSA stack carries KV on only 12 of 48 layers, so a full 262K session costs 6.2 GB against 24.6 GB for a dense 27B. It is the cheapest full-context concurrency available.
- **VRAM does not pool across a WAN.** Nebula and Ray cannot make two rented boxes act as one. The answer is replication behind a router, which we already run.
- **Nor does it pool well across eight cards in one chassis.** The RTX PRO 6000 has **no NVLink**. Published benchmarks put 8-way tensor parallelism on it ~3x behind an H100. The serving unit is therefore a **TP=2 pair**, replicated - not one large engine. See Part VI.
- **Shared capacity and SLO capacity are different products.** The $0.66/hr rate is explicitly shared infrastructure; the 99.9% SLA sits on a $299/month tier. The user tier has to be priced on the latter.
- **9-to-5 on-demand beats 24/7 spot.** At a 23.7% duty cycle, hourly on-demand capacity costs less in absolute dollars than always-on interruptible capacity, and it cannot be evicted mid-session.

What a contributor gets

Tier	Sessions	Agent rail (shared, at cost)	Agent rail (+20% ops)	User rail (99.9% SLA)	Nearest subscription equivalent
S	3	$40.38	**$48.46**	**$105.54**	~2 Claude Max 20x = $400
M	10	$134.60	**$161.52**	**$351.80**	~7 Claude Max 20x = $1,333
L	25	$336.50	**$403.80**	**$879.50**	~17 Claude Max 20x = $3,333
XL	50	$673.00	**$807.60**	**$1,759.00**	~33 Claude Max 20x = $6,667

Slots are sessions of qwen38-flash-next at full 262,144-token context, priced on a TP=2 pair at $13.46 (shared) or $35.18 (SLA) per session. Other models cost more per slot; see Part IV. The user rail stays 3.7x under a subscription even carrying a real uptime guarantee, and the agent rail - which tolerates a shared card - is 8.3x under.

Part I - Method and Constants

Every number in this prospectus derives from four constants. They are stated here so any figure can be recomputed or challenged.

Constant	Value	Derivation
Priced duty cycle	**173.3 h/month**	8 h/day x 5 days/week x (52/12) weeks/month. 23.7% of the 730 h always-on month.
Usable VRAM per card	**88 GB** of 96 GB	vLLM gpu_memory_utilization ~0.92, leaving headroom per the fleet build rule of >=1 GiB/GPU free under load.
Reference rate	**$0.66/GPU-hr**	packet.ai on-demand RTX PRO 6000, cheapest verified in-stock of 42 tracked providers on 2026-09-08.
Card-month	**$114.38**	$0.66/hr x 173.3 h. This is the atomic cost unit used throughout.

Why 9-to-5 on-demand, and not spot or reserved

At a 23.7% duty cycle the ranking of purchase modes inverts from the usual advice. Per-second on-demand billing for 173.3 hours costs less than always-on interruptible capacity, while remaining non-evictable. Reserved capacity, which bills the full 730-hour month, only wins above roughly a 60% duty cycle.

Purchase mode	Rate	Hours billed	Monthly	Evictable?
On-demand, 9-to-5 (this proposal)	$0.66/hr	173.3	**$114.38**	No
Spot / interruptible, always-on	$0.09/hr	730	$65.70	**Yes**
On-demand, always-on	$0.66/hr	730	$481.80	No
Reserved 1-month commit	$0.48/hr	730	$350.40	No
Flat monthly subscription	-	730	$299.00	No

Spot at $0.09/hr is the cheapest headline on the board and is the correct rail for the throttled agent tier, where a re-provision is invisible. It is the wrong rail for an interactive user session under an SLO, and the eviction rate on that marketplace is still unmeasured by us.

Part II - The Vendor Floor

RTX PRO 6000 96 GB, on-demand, read 2026-09-08 across 42 tracked providers (142 of 333 configs in stock). Market median on-demand is $2.20/GPU-hr; the three cheapest sit at less than half of it.

#	Vendor	$/GPU-hr	$/card-month @ 9-to-5	Host RAM (1x)	Serves flash-next?
1	**packet.ai**	**$0.66** shared $299/mo flat + SLA	**$114.38** $299.00	not published	Needs a lease to confirm
2	**HyperAI**	$0.80	$138.64	90 GB	No - fails 110 GB floor
3	**GPUhub**	$0.91	$157.70	**110 / 220 / 440 / 880 GB** (1x / 2x / 4x / 8x)	Yes - confirmed, scales linearly
4	Vast.ai	$1.01	$175.03	126 GB	Yes - safe margin
-	Market median	$2.20	$381.26	varies	-

Host RAM reorders the vendor table. qwen38-flash-next needs at least 110 GB of system RAM for its 51B-parameter n-gram table, which lives outside VRAM. HyperAI is second-cheapest and disqualified on that constraint alone. GPUhub is confirmed at 110 GB per card and scales linearly. packet.ai publishes no host-RAM figure at all - not on its pricing page, its product page, or a reachable docs site - so the cheapest rate on the board is the one we cannot yet qualify.

packet.ai sells two different products at one card

The $0.66/hr headline is the **Dynamic** tier, which the vendor describes as *shared infrastructure*. The **99.9% uptime SLA sits on the $299/month flat tier**, and single-tenant dedicated capacity is listed as launching rather than live. That distinction is load-bearing for this prospectus: an SLO-backed user session cannot honestly be priced on shared capacity with no uptime guarantee.

Tier	Rate	Month @ 9-to-5	Isolation	SLA	Use for
Dynamic	$0.66/hr	**$114.38**	Shared	None stated	**Agent tier** - throttleable, queue-tolerant
Flat	$299/mo	**$299.00**	Shared	**99.9%**	**User tier** - SLO-backed sessions
Dedicated	on request	-	Single-tenant	on request	Not yet available

Other confirmed specifics: packet.ai runs the **Workstation Edition (600W)**, not the Server Edition; the card is **PCIe Gen 5 with no NVLink** (which drives Part VI); instances ship with CUDA 12.8 and vLLM preinstalled and full root SSH; provisioning is claimed under five minutes.

Why this card

- **96 GB is the concurrency unit.** A 32 GB card cannot hold a single 262K session of a dense 27B model alongside its weights. The jump from 32 GB to 96 GB is not 3x the capacity, it is the difference between zero and seven sessions.

- **Blackwell sm_120 runs NVFP4 on native FP4 tensor cores.** Catamaran's quantised checkpoints are NVFP4, so this card computes them in their stored format rather than dequantising first. That is a throughput advantage, *not* an exclusivity claim - Ada, Hopper and Ampere all serve this catalogue too, and Addendum A prices them.

- **Rental economics have decoupled from purchase economics.** The card itself now lists around $16,000 and rose 87% in 18 months on a GDDR7 shortage, while rental rates stayed roughly flat. Renting is the rational side of that trade.

Part III - What a Session Actually Costs

A served model consumes VRAM in two pools. Weights are paid **once** per pod, regardless of how many users are attached. The KV cache is paid **per concurrent session**, and scales linearly with context length. This asymmetry is the whole economic story.

Model	Weights	KV per token	Full native context	GB per full-ctx session
qwen38-flash-next	66 GB W4A16	~0.024 MB	262,144	**6.2 GB**
qwen38-27b	14 GB NVFP4	0.094 MB fp8	262,144	**24.6 GB**
glm-5.3-flash REAP50	99 GB NVFP4	~0.052 MB MLA	1,310,720	**67.9 GB**
glm-5.3-flash at 262K	99 GB NVFP4	~0.052 MB MLA	262,144	13.6 GB
glm-5.3	~380 GB	~0.052 MB MLA	1,310,720	67.9 GB
qwen38-max	~1.2 TB	-	1,000,000	not viable

The architecture bonus

qwen38-flash-next is 6.5x the parameter count of qwen38-27b and costs **4x less per session**. Its 48-layer stack repeats three Gated-DeltaNet layers for every one Qwen Sparse Attention block. Gated DeltaNet holds O(1) recurrent state rather than a growing KV cache, so only 12 of 48 layers pay per-token KV at all. This is not a tuning trick; it is the model's architecture, and it makes it the single best value for full-context concurrency in the catalogue.

The 1M-context trap

glm-5.3-flash advertises a 1,310,720-token context. That number is real and it costs **67.9 GB of VRAM per session to actually use**. On a 192 GB pod, after 99 GB of weights, it buys exactly one session. The same model at 262K buys thirteen. Long context and high concurrency are directly traded against each other, and any capacity plan that quotes a million-token window without pricing its KV is understating cost by roughly 5x.

> **Practical consequence: sell context as a tier, not as a feature. A user tier at full native context and an agent tier clamped to 90K-121K are different products with a 3-10x cost difference, served from the same pod.**

Part IV - Cost by Concurrency

Three concurrency levels against the three cheapest vendors, at full native context. Card counts assume 88 GB usable per card, with weights and KV both sharded across cards by tensor parallelism inside one chassis.

qwen38-flash-next - full 262,144 context

Sessions	VRAM needed	Cards	packet.ai	GPUhub	Vast.ai	$/session (best)
3x	84.6 GB	1	**$114.38**	$157.70	$175.03	$38.13
5x	97 GB	2	**$228.76**	$315.40	$350.06	$45.75
15x	159 GB	2	**$228.76**	$315.40	$350.06	**$15.25**

HyperAI is excluded here on the 110 GB host-RAM floor. Note that 15x costs the same as 5x - the second card's VRAM is mostly unused at 5x, so buying card-shaped concurrency rather than round numbers is free money.

qwen38-27b - full 262,144 context

Sessions	VRAM needed	Cards	packet.ai	HyperAI	GPUhub	$/session (best)
3x	87.8 GB	1	**$114.38**	$138.64	$157.70	$38.13
5x	137 GB	2	**$228.76**	$277.28	$315.40	$45.75
15x	383 GB	5	**$571.90**	$693.20	$788.50	$38.13

glm-5.3-flash REAP50 - full 1,310,720 context

Sessions	VRAM needed	Cards	packet.ai	HyperAI	GPUhub	$/session (best)
3x	302.7 GB	4	**$457.52**	$554.56	$630.80	$152.51
5x	438.5 GB	5	**$571.90**	$693.20	$788.50	$114.38
15x	1,117.5 GB	13	**$1,486.94**	$1,802.32	$2,050.10	$99.13

The same model clamped to 262K: 3x costs $228.76 (2 cards), 5x costs $228.76 (2 cards), 15x costs $457.52 (4 cards) - $76.25 / $45.75 / $30.50 per session. The million-token window is a 3-5x multiplier on identical silicon.

The two models that do not clear

Model	3x	5x	15x	Verdict
glm-5.3 (380 GB weights)	7 cards, $800.66	9 cards, $1,029.42	16 cards, $1,830.08	Buy tokens
qwen38-max (~1.2 TB)	16 cards, $1,830.08	18 cards, $2,058.84	26 cards, $2,974.88	Buy tokens

A single $168/month vendor subscription covers both glm-5.3 and glm-5.3-flash. These two models should be bought, not hosted. Being honest about that is what makes the rest of the proposal credible.

Part V - Can We Pool VRAM Across Cards?

The natural question for a bulk buy is whether many small rentals can be welded into one large memory domain over our existing Nebula mesh, with Ray as the orchestrator. The answer is no, and the margin is not close.

Tensor parallelism over a WAN: 512x to 72,000x too slow

Interconnect	Bandwidth	Verdict for tensor parallelism
NVLink (in-chassis)	~900 GB/s	Designed for it
PCIe Gen5 x16 (in-chassis)	~64 GB/s	Workable - what we would use
100 GbE / InfiniBand (datacenter)	~12.5 GB/s	What vLLM multi-node assumes
Nebula over WAN	**0.0125 - 0.125 GB/s**	512x to 72,000x short

Tensor parallelism performs an all-reduce at every layer - on the order of a hundred collective operations per generated token. At Nebula's WAN bandwidth the collectives dominate compute by three to four orders of magnitude. This is a physics result, not a configuration problem, and no amount of tuning changes it.

Pipeline parallelism: bandwidth-feasible, latency-fatal

Pipeline parallelism passes activations only at stage boundaries, so bandwidth is not the binding constraint. Round-trip latency is. Each stage adds a full network RTT to **every token**. Eight stages at 20-50 ms RTT adds 160-400 ms per token, yielding 2.5 to 6 tokens per second. That is unusable for interactive work, and Catamaran has already measured vLLM-side pipeline limits on this model family independently.

Ray does not change the answer

Ray is the orchestration layer vLLM uses to coordinate multi-node serving. It schedules and supervises workers; it does not alter the interconnect. vLLM's multi-node path presumes datacenter fabric. Catamaran's own vendor documentation already records the conclusion: there is **no cross-node tensor parallelism** across the mesh, which is exactly why fleet VRAM has never been treated as a single 152 GB pool.

The correct architecture is not pooling. It is replication behind a router - which Catamaran already operates. The cremeapi broker resolves a model slug to a currently-running member and round-robins across all members of a pooled slug. Adding a pod adds capacity with zero client-side change.

Why replication is better than pooling anyway

- **Fault isolation.** A pooled 8-node TP group dies entirely if one node drops. Eight independent pods lose one eighth of capacity.
- **Linear scaling.** No collective overhead means capacity scales exactly with spend. Pooled TP has sublinear returns past a point.
- **Vendor independence.** Pods can sit on different vendors, so a single provider's outage or price move is survivable.
- **Rolling upgrades.** Pods can be replaced one at a time behind the router.

Part VI - The Serving Unit, and Why It Is Not a Whole Chassis

Because weights are paid once per pod and KV is paid per session, bigger pods are strictly more efficient. The marginal cost of a session falls as the fixed weight cost spreads across more of them.

qwen38-flash-next at full 262,144 context, packet.ai shared rate

Grp	GPU	TP	VRAM	Sess.	$/month	$/session/mo	PCIe OK?
G1	1	1	88 GB	3	$114.38	$38.13	Yes
G2	**2**	**2**	**176 GB**	**17**	**$228.76**	**$13.46**	**Yes - default**
G4	4	4	352 GB	46	$457.52	$9.95	Probably - validate
G8	8	8	704 GB	102	$915.04	$8.97	NO - see below

An earlier draft made G8 the flagship. That was wrong, and the reason is the card's interconnect - not its memory.

The RTX PRO 6000 has no NVLink, and 8-way TP pays for it

The card is **PCIe Gen 5 only, with no NVLink** - confirmed on the vendor's own product page. Tensor parallelism all-reduces at every layer, so its cost rises steeply with group size once the collectives cross PCIe rather than a dedicated fabric. Published third-party benchmarking on an 8x RTX PRO 6000 node (64-core EPYC Genoa, NVMe RAID0) measured exactly that gradient:

Configuration	Result	Reading

1x PRO 6000 vs 1x H100 SXM	**3,140 vs 2,987 tok/s**	PRO 6000 wins outright when no collectives are involved
8x as independent replicas	15,713 tok/s = **63% scaling**	Sub-linear, but attributed to load-balancer and batching overhead, not the bus
4-GPU tensor parallel	H100 **+31%**	Tolerable - the gap is real but not disqualifying
8-GPU tensor parallel	H100 **~3x**, H200 ~4x	The bus is now the bottleneck
Cost per Mtok at 8-way TP	PRO 6000 **$1.72** vs H100 $0.76	Blackwell is 2.3x WORSE per token here

Pipeline parallelism does not rescue it - the same benchmark reports PP "resulted in overall lower performance." These figures are third-party and on other models (GLM-4.5-Air, GLM-4.6, Qwen3-Coder-480B), so the exact multipliers will differ for our catalogue. The *direction* is not in doubt, and it is enough to disqualify G8 as a default.

So the unit is a pair, replicated

qwen38-flash-next's 66 GB of weights fit on a single 96 GB card, so tensor parallelism here is not enabling the model - it only pools KV to raise session count. That makes group size a free choice, and the cheapest safe choice is small. An eight-card chassis is therefore **four independent G2 engines behind the router**, not one G8:

Topology on one 8-card chassis	Sess.	$/sess (shared)	$/sess (SLA tier)	Verdict
8x G1 (replicas, no TP)	24	$38.13	$99.67	Wastes VRAM - weights copied 8 times
4x G2 (TP=2, replicated)	**68**	**$13.46**	**$35.18**	**Default - small collectives, good density**
2x G4 (TP=4, replicated)	92	$9.95	$26.00	Upside if our benchmark clears it
1x G8 (TP=8)	102	$8.97	$23.45	Rejected - ~3x throughput penalty

This is a better fit for the architecture we already run. The broker round-robins across members of a pooled slug, so four G2 engines on one chassis register as four members and load-balance with no client change - the same mechanism that scales across chassis in Part V. The max unit is not a chassis. It is a TP=2 pair, and there is no maximum on how many of them sit behind the slug.

The full matrix - one 8-card chassis as 4x G2, all five models

Model	Weights	Context	GB/sess	Per G2	Per chassis	$/session
qwen38-flash-next	66 GB	**262,144 (native)**	6.2	17	**68**	**$13.46**
qwen38-flash-next	66 GB	121,000	2.9	37	148	**$6.18**
qwen38-flash-next	66 GB	32,768	0.8	137	548	$1.67
glm-5.3-flash	99 GB	**1,310,720 (native)**	67.9	1	4	$228.76
glm-5.3-flash	99 GB	262,144	13.6	5	20	$45.75
glm-5.3-flash	99 GB	**121,000**	6.3	12	**48**	**$19.06**
qwen38-27b	14 GB	**262,144 (native)**	24.6	6	24	$38.13
qwen38-27b	14 GB	121,000	11.4	14	56	$16.34
qwen38-27b	14 GB	32,768	3.1	52	208	$4.40
glm-5.3	380 GB	any	-	0 - exceeds a G2 and a G4	-	n/a
qwen38-max	~1,200 GB	any	-	0 - exceeds the chassis	-	n/a

Dropping G8 costs glm-5.3 its place entirely. Its 380 GB of weights exceed both a G2 (176 GB) and a G4 (352 GB), so on a no-NVLink card it is only servable at the TP=8 size we just rejected. glm-5.3 is therefore a buy on this hardware - or a Hopper purchase, where NVLink makes TP=8 work (8x H100 at 121K gives ~32 sessions for $2,828/month, or $88 each, still well above a $168 subscription).

What survives is a clean three-model self-host list: qwen38-flash-next, qwen38-27b and glm-5.3-flash. Within it, reduced context is still the biggest lever - glm-5.3-flash goes from 4 sessions per chassis at its native 1.31M window to 48 at 121K, and flash-next from 68 to 148.

qwen38-max is excluded on architecture, not price

At roughly 1.2 TB of 4-bit weights, qwen38-max does not fit even a full 8-card chassis's 704 GB at *any* context - it cannot even load. Fourteen cards are needed for weights alone, which is two chassis, which means cross-chassis tensor parallelism over InfiniBand: a different and considerably more expensive product than the single-chassis pods priced here. Costed anyway, for completeness:

Sessions @ 121,000	Cards	Topology	Monthly	$/session
3x	14	2 chassis + InfiniBand	$1,601.32	$533.77
5x	15	2 chassis + InfiniBand	$1,715.70	$343.14
15x	16	2 chassis + InfiniBand	$1,830.08	$122.01

Against $10/month for a vendor subscription that includes it, or $1.60/$4.80 per million tokens on a zero-markup router, this is not close. qwen38-max is a buy, permanently. Its KV rate here is an **assumption** (0.078 MB/token, GQA-shaped) since Alibaba has not published the attention config - but the weights alone settle the question before KV enters it.

Part VII - Bulk-Buy Scenarios

Scenario A - 10 contributors, 10x concurrency each

One hundred concurrent sessions of qwen38-flash-next at full 262K, served as G2 pairs at 17 sessions each, needs **six pairs - twelve cards**, provisioning 102 slots. That is one and a half chassis, or six independent engines behind one slug; the router does not care which.

Line item	Agent rail (shared)	User rail (99.9% SLA)
Topology	6 x G2 = 12 cards	6 x G2 = 12 cards
Rate	$0.66/hr, 9-to-5	$299/month flat
Monthly cost	**$1,372.56**	**$3,588.00**
Slots provisioned	102 (100 sold, 2 spare)	102 (100 sold, 2 spare)
Concurrency each	10x at full 262,144 context	10x at full 262,144 context
Cost per contributor	**$137.26 / month**	**$358.80 / month**
Same on Claude Max 20x	~6.7 subs = $1,333/mo	~6.7 subs = $1,333/mo
Advantage	**9.7x cheaper**	**3.7x cheaper**

The user rail carries a real 99.9% uptime commitment and still lands under a third of the subscription cost. That is the number to quote to anyone who asks what the guarantee costs.

Scenario B - Mixed tiers filling one pod

Contributors need not buy the same tier. One 8-card chassis run as 4x G2 provisions 68 slots on the agent rail, sellable as any mix. One worked fill, with a 20% operating reserve for the ops burden Catamaran carries ($13.46 at cost becomes $16.15 a slot):

Tier	Slots each	Subscribers	Slots used	Price each	Line total
S	3	11	33	$48.46	$533.06
M	10	1	10	$161.52	$161.52
L	25	1	25	$403.80	$403.80
Total	-	**13**	**68**	-	**$1,098.38**
Chassis cost, 4x G2	-	-	-	-	*$915.04*
Operating reserve	-	-	-	-	**$183.34 (20%)**

The reserve funds provisioning automation, checkpoint distribution, monitoring, and a buffer against marketplace price moves. At cost with no reserve the same fill is $13.46/slot; on the SLA rail it is $35.18 before reserve.

Slots are a ballpark, and they are tradeable

"Ten people at ten sessions" is an illustration, not a product shape. A slot is a unit of reserved concurrency, and nothing requires it to be assigned evenly or permanently. Three properties follow, and all three are cheap to implement because the admission layer already exists:

- **Toggleable.** A subscriber's slot count is a number in the tenant record, not a provisioning event. Moving from 3 slots to 12 for a busy fortnight is a configuration change against a pod that is already running.
- **Tradeable in-network.** Slots can be transferred between contributors on platform. Somebody who bought 25 and is using 6 can sell the difference into a window rather than let it idle - the same idle capacity that makes the second customer on a paid-for card nearly pure margin.
- **Window-priced.** One subscriber at 100 sessions may be purchasable in some windows and not others. An entire pod is 102 slots; if it is quiet on a Tuesday morning, that is a sellable block.

This asks for a **rolling capacity window** rather than a monthly allocation - the same mechanism the major vendors use when they meter on a 5-hour rolling basis alongside a weekly cap. The exact cycle is ours to pick: a 9-to-5 pod has a natural 8-hour daily window, and a shorter 2- or 4-hour rolling window would give finer-grained resale without making planning unpredictable. Whatever the period, the capacity-planning property is what matters: **reserved concurrency that expires is inventory, and inventory can be resold.**

Scenario C - 100 contributors, 1,000x aggregate concurrency

One thousand slots at 17 per G2 requires **59 pairs - 118 cards**, roughly 15 chassis, all replicated behind the existing slug-pooling router. Per-contributor economics are **identical** to Scenario A, because the architecture scales linearly - and it now scales in units of two cards rather than eight, which makes the capacity granularity considerably finer.

Line item	Value
Topology	59 x G2 = 118 cards, ~15 chassis
Aggregate slots	1,003 provisioned, 1,000 sold
Monthly cost, 9-to-5 (agent rail)	**$13,496.84**
Cost per slot	$13.46
Contributors	100, at 10x each
Cost per contributor	**$134.97 / month**
Failure domain	One G2 = **1.7%** of capacity, router re-routes automatically

Note the second-order benefit of the smaller unit: losing one engine costs 1.7% of the pool instead of the 10% a chassis-sized unit would have cost. Rejecting G8 on throughput grounds also bought a 6x smaller blast radius.

Scenario D - the ceiling

One hundred contributors at 1,000x **each** would be 100,000 slots: 5,883 G2 pairs, 11,766 cards, roughly $1.35M per month. The arithmetic is linear and holds, but the **market does not**. The entire tracked RTX PRO 6000 supply across 42 providers is 142 in-stock configurations. A 118-card order is already a visible fraction of available inventory, and anything beyond roughly 200 cards would need to be negotiated as reserved capacity across several vendors rather than clicked on a marketplace.

> **Honest ceiling for a marketplace-sourced bulk buy: roughly 200-250 cards, or 1,700-2,100 concurrent full-context sessions, before supply rather than budget becomes the binding constraint. Addendum A halves that problem by qualifying Ada as a second supply pool.**

Part VIII - Service Tiers

Catamaran's model catalogue already carries the fields this split needs - **max_context**, **interactive_ctx_ceiling** and **prefill_budget_tokens** are live on every running model today. The change is to make the ceiling a property of the *tier* rather than a per-model constant.

Two tiers, one pod

	User on-demand	Agent on-demand
Context ceiling	Full native (262,144)	Clamped 90,000 - 121,000
Who	A human in a harness session	Orchestrated sub-agents, batch, CI
Throttleable	No - SLO-backed	Yes - queued and preemptible
Slots per chassis (4x G2)	68	148 - 200
Rental rail	**$299/mo flat, 99.9% SLA**	**$0.66/hr shared, or spot**
$/slot/month at cost	**$35.18**	**$4.58 - $6.18**

The two tiers now differ in *rail* as well as context. A user session that carries an SLO belongs on capacity that carries one; an agent session that can be queued and retried belongs on the cheapest card available, shared or evictable. That is a 5.7x to 7.7x spread on the same silicon, earned by matching the guarantee to the workload rather than buying the strongest guarantee for everything.

The agent tier is where the rented card pays for itself. Most orchestrated tasks do not need a quarter-million tokens of context; a task estimated at 90K should be admitted at 90K. Doing so roughly triples session density on hardware already paid for, and the tokens saved are the expensive kind.

What Catamaran contributes

- **Quantised checkpoints.** NVFP4 and REAP50-pruned builds produced in-house, Blackwell-native, already staged and validated.
- **Runtime images.** Digest-pinned vLLM and llama.cpp images built on our own BuildKit, never floating tags.
- **Deployment manifests and provisioning.** Lease lifecycle with TTL sweeper, so a forgotten pod cannot bill 730 hours instead of 173.
- **Routing.** The cremeapi broker and proxy, with slug-based pooling across pods and ten pre-configured client harnesses.
- **Observability and SLO.** Existing Prometheus, Grafana and alerting stack, with the fleet's best-effort SLA standard applied to the shared pool.

Part IX - Risks and Open Questions

Stated plainly, because a prospectus that hides these is not worth reading.

#	Risk	Status	Mitigation
1	**Checkpoint distribution.** Our quantised artefacts (99-173 GB) must reach a rented box quickly enough that a 9-to-5 pod is not spent pulling weights.	DESIGN TASK	**Not a blocker.** catamaran-3 and the catx nodes are on 10G, expandable to 40/100/400G, and we already run an S3 idiom. A public S3 endpoint moves 99 GB in 1-2 minutes at 10G line rate, or ~13 minutes against a 1 Gbit rented-box ingress. The 73 KB/s figure that first flagged this describes the Zot-over-Nebula relay path, not the hosts' network. Build the S3 front door; do not route bulk checkpoints over the mesh.
2	**packet.ai publishes no host-RAM figure** - not on its pricing page, product page, or a reachable docs site. 110 GB is a hard floor for qwen38-flash-next, and packet.ai is the cheapest rate on the board.	NEEDS A LEASE	Confirmed without a lease: GPUhub is **110/220/440/880 GB** at 1x/2x/4x/8x, and L40S runs 62-147 GB per card across providers. Only packet.ai is unknown, and it is the one we most want to use. A single short lease settles it. Expect to run all three vendors, not pick one.
2b	**packet.ai's $0.66/hr tier is shared infrastructure** with no stated uptime guarantee; the 99.9% SLA is on the $299/month tier.	PRICED IN	The two-rail pricing throughout this document is the response: agent slots on shared capacity at $13.46, user slots on SLA capacity at $35.18. Do not sell an SLO-backed session off a shared card.
3	Marketplace prices move daily; $0.66/hr is a snapshot, not a contract.	ONGOING	20% operating reserve; multi-vendor pods so one provider's move is absorbed; re-read rates before each billing cycle. Vendor diversity is the hedge.
4	Spot eviction rate is unmeasured, and a re-provision costs a cold start we have never clocked at pod scale.	TEST	Two measurements, both early: eviction frequency over a full month, and **cold-launch wall-clock** for a replacement pod - image pull, checkpoint fetch, engine warm-up, router registration. Cold start sets the agent tier's real SLO, so it is a first-class benchmark, not a footnote.
5	Balance-at-zero stops running instances on some marketplaces.	KNOWN	Autobilling plus a per-lease ceiling is mandatory, not optional.
6	Supply ceiling at roughly 20-30 pods on marketplace inventory.	KNOWN	Negotiate reserved capacity across several vendors above ten pods.
7	Throughput per session is modelled, not yet benchmarked at 8-card scale.	BENCHMARK	VRAM capacity is proven arithmetic; tokens/sec under 102-way concurrency is not. Benchmark early and often - on the first card, again on the first pod, and on every model/context tier before it is sold.

Item 7 deserves emphasis. Everything in this document sizes memory, which is a hard constraint and simple arithmetic. It does not size throughput. A pod that fits 102 sessions in VRAM may deliver unacceptable tokens-per-second to all 102 at once. The first pod is a measurement, not a product.

Part X - The Ask

A staged commitment, with a real decision point after each stage.

Stage	What happens	Cost	Decision gate
0. Probe	One-shot leases on packet.ai, GPUhub and Vast.ai reporting host RAM, disk, topology and real throughput. Stand up the public S3 checkpoint endpoint on the 10G side.	< $20	Which vendor fits which model? Expect to keep all three.
1. One U1	Single card, qwen38-flash-next at full 262K, 3 slots. Benchmark tokens/sec under load **and cold-launch wall-clock**. Validate provisioner, TTL sweeper and router registration.	$114.38 / mo	Does a rented pod serve our artefact at acceptable speed, and how fast does a replacement come up?
2. One G2, then a chassis	First production pair: 17 slots at full 262K. **Benchmark TP=2 against TP=4 on our own models** - the published PCIe numbers are on someone else's. Then fill a chassis as 4x G2 and onboard 10-15 contributors across tiers S/M/L.	$228.76, then $915.04 / mo	Does TP=4 clear? If so, slots go from 68 to 92 per chassis at no extra cost.
3. Replicate	Add G2 pairs behind the existing slug-pooling router as demand requires. Linear scaling in two-card increments to roughly 200-250 cards.	$228.76 / pair	Demand-led, no further architecture work.

The one-line case

> *Ten people who each want ten simultaneous full-context agent sessions can pay $1,333 a month each for seven Claude Max 20x subscriptions, or $137.26 a month each for a share of twelve cards running weights we already quantise, host and route - $358.80 if they want a 99.9% uptime guarantee with it. The engineering is done. What is missing is the card rent and a decision about who is on the pool.*

Addendum A - Ada, Hopper and Ampere

An earlier draft of this document said renting Ada-class cards "would force an AWQ/GPTQ requantisation ladder that does not currently exist." That is wrong on both halves, and the fleet's own catalogue is the refutation. Correcting it matters commercially, not just editorially: Blackwell supply is the tightest constraint in Part VII, and three other architectures serve this catalogue today.

What Ada actually runs, from the live catalogue

catamaran-3 is four Ada cards (**dra:adax4**). Its viable vLLM entries alone span **six NVFP4 builds, three AWQ, and one FP8** - including **c3-qwen38-27b-nvfp4-vllm at the full 262,144 context**. NVFP4 checkpoints load and serve on Ada; what Ada lacks is FP4 *tensor cores*, so it dequantises to compute. That is a throughput cost, not a compatibility gate, and the requantisation ladder is not hypothetical - it is already built.

Runtime	Slots	Context	Representative catalogue entry
vLLM	adax4	**262,144**	c3-qwen38-27b-**nvfp4**-vllm
vLLM	adax4	262,144	c3-thinkingcap-27b-**fp8**-vllm
vLLM	adax2	262,144	c3-thinkingcap-27b-**awq**-vllm
vLLM	adax4	131,072	c3-glm-45-air-reap-82b-awq-vllm
vLLM	adax2	20,480	c3-**glm**-47-flash-awq4-vllm - the MLA wall, in our own data
llama.cpp	adax2	**202,752**	c3-**glm**-47-flash-uncensored-llamafile - same model, 10x the context
llama.cpp	adax4	131,072	c3-qwen38-flash-next-gguf-llamacpp (running now)
SGLang	adax4	-	c3-qwen3-next-80b-a3b-awq-sglang

The two GLM rows are the real Ada limit, and they are worth reading together. The same model gets **20,480 tokens on vLLM and 202,752 on llama.cpp**. GLM's sparse/DSA attention path is gated on compute-capability major in {9, 10, 12} - Hopper and Blackwell - and Ada is major 8. So on Ada, GLM is a llama.cpp model. Qwen has no such gate and runs at full native context on vLLM.

Capability by architecture

Architecture	SM	FP4 cores	FP8 cores	NVFP4 ckpt	GLM sparse on vLLM	Notes
Blackwell RTX PRO 6000	12.0	Native	Yes	Native	Yes	Best throughput on our artefacts
Hopper H100 / H200	9.0	No	Yes	Loads	Yes	Only non-Blackwell route to GLM sparse
Ada L40S / 4090 / 4060 Ti	8.9	No	Yes	Loads (proven)	No - llama.cpp	Full-context Qwen on vLLM; GLM via llama.cpp
Ampere A100 80 GB	8.0	No	No	Loads	No	Only card with the 100 KB shared memory for **dense MLA**
Ampere RTX 3090	8.6	No	No	Loads	No	Cheapest VRAM on the board

The Ampere caveat that decides its economics. Ampere has no FP8 tensor cores. Every KV figure in this prospectus assumes an fp8 KV cache. vLLM will still store 8-bit KV on Ampere by converting on read, but that path is not the native one and must be benchmarked, not assumed. If a lane has to fall back to fp16 KV, per-session VRAM doubles and Ampere's price advantage disappears entirely. Measure this before buying Ampere for a concurrency tier.

Price per GB of VRAM per month, 9-to-5

VRAM is what buys concurrent sessions, so the honest cross-architecture comparison is dollars per gigabyte per month at the same duty cycle. Cheapest verified in-stock on-demand rate for each card, 2026-09-08.

Card	Arch	VRAM	$/hr	$/card-month	$/GB/month	Spot $/GB/mo
RTX 3090	Ampere	24 GB	$0.15	$26.00	**$1.08**	-
RTX PRO 6000	Blackwell	96 GB	$0.66	$114.38	**$1.19**	**$0.16**
L40S	**Ada**	48 GB	$0.38	$65.85	**$1.37**	$0.97
A100 80 GB	Ampere	80 GB	$0.81	$140.37	$1.75	-
A100 40 GB	Ampere	40 GB	$0.43	$74.52	$1.86	$0.56
RTX 4090	**Ada**	24 GB	$0.27	$46.79	$1.95	$0.51
H100	Hopper	80 GB	$2.04	$353.53	$4.42	-
H200	Hopper	141 GB	$4.44	$769.45	$5.46	-

Blackwell is **not** the cheapest VRAM - the RTX 3090 is, at $1.08/GB. Blackwell is second at $1.19, and Ada's L40S is a close third at $1.37 with a real datacenter card. Hopper is 3.7-4.6x the price per gigabyte: you buy it for the GLM sparse gate, never for capacity.

The same tier, built four ways

15 concurrent sessions of qwen38-flash-next at full 262,144 context needs 159 GB (66 GB weights + 15 x 6.2 GB KV). Card counts assume the same ~92% usable fraction and a tensor-parallel size of 2, 4 or 8.

Build	Arch	Cards	TP	Sessions	$/month	Verdict
2x RTX PRO 6000	**Blackwell**	**2**	**2**	17	**$228.76**	Default - fewest cards, native FP4, smallest collectives
4x L40S	**Ada**	4	4	17	**$263.40**	Second source - 15% more, doubles addressable supply
4x A100 80 GB	Ampere	4	4	36	$561.48	Buy for dense-MLA models; check the fp8-KV caveat first
4x H100	Hopper	4	4	36	$1,414.12	Buy for GLM sparse, or anything needing TP=8
8x RTX 3090	Ampere	8	**8**	-	$208.00	Disqualified - 66 GB of weights needs TP=8 on 24 GB cards
8x RTX 4090	Ada	8	**8**	-	$374.32	Disqualified - same reason

The no-TP=8 rule from Part VI reaches back into this table and removes the two cheapest rows. A 24 GB card cannot hold 66 GB of weights below TP=4, and TP=4 on 24 GB leaves almost no KV - so the 3090's headline $1.08/GB never converts into sessions for this model class. Small cards are for small models and short context, which is exactly the agent tier, and not for full-context flash-next.

Among what survives, 2x RTX PRO 6000 wins on card count and native FP4 - but only by 15% over Ada's L40S, and that matters because Part VII's binding constraint is supply, not price. The tracked L40S market is 35 providers and 91 in-stock configurations, an entirely separate pool from the PRO 6000's 42 providers and 142 configs. Qualifying Ada roughly doubles how many cards a bulk buy can source. L40S host RAM is 62-147 GB per card depending on provider, so a 4-card group clears the 110 GB flash-next floor everywhere.

Hopper is no longer only a capability purchase

The first version of this addendum called Hopper a capability buy for the GLM sparse gate and nothing else, on the grounds that it costs 3.7-4.6x per gigabyte. The PCIe benchmark in Part VI complicates that. **At 8-way tensor parallelism, H100 delivers roughly 3x the throughput of a PRO 6000 and costs $0.76 per million tokens against $1.72** - so for any model that genuinely requires a large TP group, NVLink hardware is *cheaper per token* despite being far more expensive per gigabyte.

Model shape	Right hardware	Why
Weights fit 1-2 cards	**Blackwell or Ada**	Small collectives; PCIe is a non-issue and Blackwell wins single-card throughput outright (3,140 vs 2,987 tok/s against H100)
Weights need TP=4	**Blackwell or Ada**	H100 leads by 31% - real, but not enough to justify 3.7x the price per GB
Weights need **TP=8**	**Hopper**	PCIe becomes the bottleneck. H100 is ~3x faster and 2.3x cheaper per token. This is where glm-5.3's 380 GB lands.

Which architecture for which lane

Lane	Architecture	Why
Qwen at full native context	Blackwell, or **Ada L40S**	Both serve 262,144 on vLLM. Ada is the second source; price gap is 15%.
GLM at long context	Blackwell or Hopper	Sparse/DSA is gated on major in {9,10,12}. On Ada, GLM is capped near 20K on vLLM - use llama.cpp instead and accept lower throughput.
Anything needing TP=8	**Hopper**	NVLink. ~3x the throughput and 2.3x cheaper per token than a PCIe card at that group size - the one place Hopper's price per GB is not the relevant number.
Dense MLA models	**A100 80 GB**	The 100 KB shared-memory requirement is met by A100 and not by consumer cards. Akash's A100 floor just fell 41% to $1.07/hr with 78 available.
Agent tier, throttled context	**Ada 4090 or 3090 spot**	$0.51 and $0.56 per GB per month on spot. Eviction is invisible to a queued agent, and short context means small KV.
Cold spare / burst	Whatever is in stock	Pods are independent behind the router. A mixed-architecture fleet is a feature here, not debt.

Net: the plan does not need to be Blackwell-only, and should not be. Blackwell stays the default for full-context Qwen pods; Ada L40S is the qualified second source that makes the supply ceiling recede; Hopper is a capability purchase for GLM; Ampere is the cheap agent-tier rail, pending the fp8-KV measurement above.

Sources

Figure	Source
GPU rates, RTX PRO 6000 and RTX 5090	getdeploying.com provider trackers, read 2026-09-08 (42 and 18 providers respectively)
packet.ai hourly and flat rates	packet.ai published pricing page
Model architecture and context windows	Hugging Face model cards (Qwen/Qwen3.8-Flash-Next, Qwen/Qwen3.8-27B, zai-org/GLM-5.3)
Weights, quantisation sizes, host-RAM floor	Catamaran internal measurement, staged checkpoints on fleet storage
Catalogue fields and running fleet state	cremeapi /catalog/public, read 2026-09-08
Claude Max 20x concurrency baseline	Third-party trackers; Anthropic does not publish concurrency limits

Prepared by Catamaran AI. All marketplace prices are snapshots taken 2026-09-08 and require re-reading before any commitment. Memory sizing is arithmetic from published architecture parameters; throughput is not yet measured at pod scale.