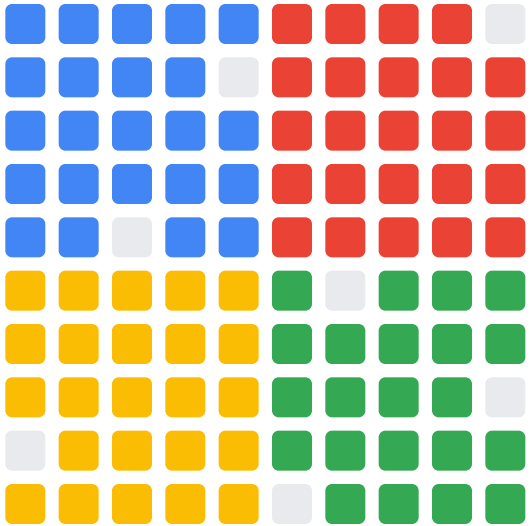
NBER MARKET DESIGN WORKING GROUP · STANFORD GSB · OCTOBER 16–17, 2026

Quota Marketplace

Dynamic Pricing for Efficient Allocation of ML Training Resources



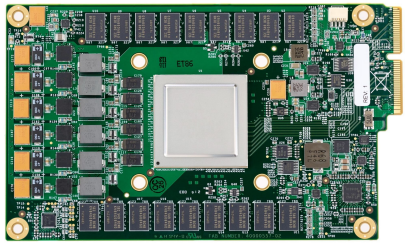
Balasubramanian Sivan · **Renato Paes Leme** · Mihai Tiuca
Ian McFarlane · Vasilis Gkatzelis · Nehal Mehta
Soheil Hassas Yeganeh · Vahab Mirrokni · Amin Vahdat

Google Research + Google Cloud

arXiv:2607.09802

The resource: ML accelerators

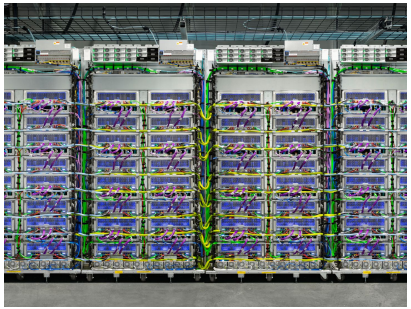
Training a frontier model takes **10^4–10^5 chips** for weeks to months, in purpose-built data centers.



Google TPU



NVIDIA H100 GPUs



Racks of TPUs



A Google data center

Scarce

Demand far exceeds supply; lead times for new capacity are months to years.

Expensive

A single chip costs $10K+; fleets are worth $10Bs.

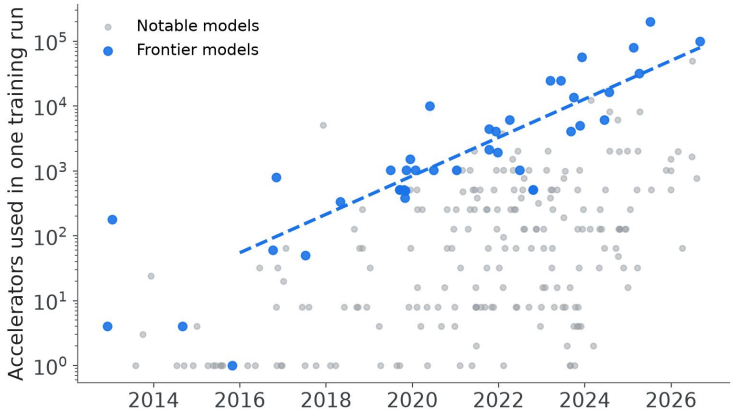
Heterogeneous & fluctuating

Many chip generations × 100+ locations.

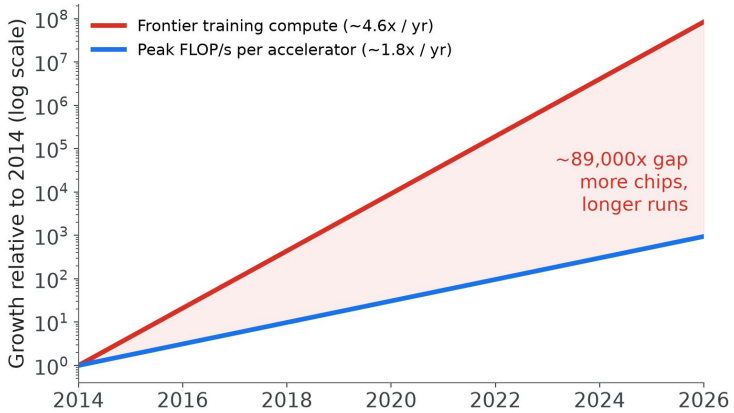
Photos (Wikimedia Commons): TPU v4 — Jouppi et al., CC BY 4.0; H100 — Geekerwan, CC BY 3.0; Frontier — OLCF at ORNL, CC BY 2.0; Council Bluffs — Chad Davis, CC BY 2.0

Demand for training compute is exploding

Chips per frontier training run: ~2.0× / year



Compute demand outpaces chip performance



>10× / yr

growth in Google's AI compute demand, sustained for 8 years [Vahdat '25]

~3× / yr

growth in the compute cost of a frontier training run, now $100M+ per run

>$XXXB

Investment in AI infrastructure across major tech companies

Inside a firm, **who gets the chips** is a first-order economic decision: ROI on $100Bs of hardware.

Data: Epoch AI, 'Notable AI Models' & 'ML Hardware' (epoch.ai/data, CC-BY); log-linear fits by us. [Vahdat '25]: Google Cloud Next '25 keynote.

Traditional allocation strategies

Static allocation + opportunistic use

Each team gets a fixed chip quota, renegotiated quarterly; anyone can run preemptible jobs on idle chips.

Drawbacks

- **No prioritization:** idle chips go to whoever grabs them, not to the most valuable job
- **No incentive to save:** unused quota earns no credit toward a later deadline
- **Preemptible:** opportunistic jobs can be killed at any time; quotas are still set quarterly

Chip-hour allocation

Each team gets a budget of chip-hours, and unused hours are banked; chips go first to the teams that have used the least [Karma, OSDI '23].

Drawbacks

- **Withholding:** teams hold back low-value demand to save hours, leaving chips idle
- **Fixed exchange rate:** an hour costs the same at peak and off-peak, so teams can't say how much a job matters

> Root cause: the "price" of a chip-hour never moves.

Vuppalapati, Fikioris, Agarwal, Cidon, Khandelwal, Tardos. 'Karma: Resource Allocation for Dynamic Demands.' OSDI 2023. Impossibility results: paper Sec. 4.1 (Thms. 1–2).

The allocation problem inside the firm

Setting

- **Principal:** the company owns a fleet of chips; supply changes over time
- **Teams:** n teams, priority weights w_i set by leadership
- **Dynamic demand:** $d_i(t)$ = chips needed to run jobs
- **Heterogeneous values:** jobs have values $v_i(t)$
- **Private information:** the principal knows w_i, but not demands or values; teams do

Desiderata

- **Efficiency:** Pareto efficient; no idle chip while someone has demand
- **Fairness:** weighted max-min: team i receives ≈ w_i / Σ w of the chip-time
- **Incentives:** scarcity breeds strategic behavior
- **Agility:** reallocate in minutes, not quarters
- **No money:** teams do not pay real dollars

> A dynamic allocation problem without money: how do we elicit both the **timing** and the **intensity** of demand?

THE IDEA

Credits instead of chips, prices instead of quotas

Teams receive **credits** (purchasing power), not chips. Every ~30 seconds the market sets a **price** for each chip type and location, and chips go to the teams whose bids clear it.

Roadmap

1. **Basic model**
   Teams, credits, bidding and market clearing for a single business unit
2. **Evidence from deployment**
   What the market delivered at Google scale
3. **Business units**
   Governance, owned chips and currencies on top of one market
4. **Auxiliary resources**
   CPU and RAM: from attach rates to a joint market with a price vector

Complex constraints

1

Basic model

Teams, credits, bidding and market clearing for a single business unit

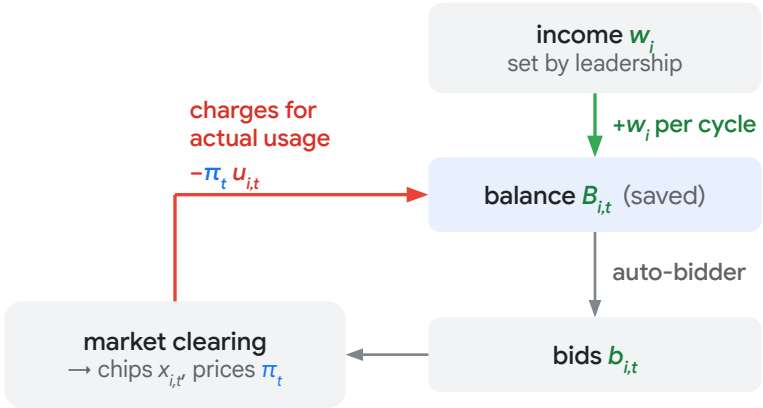
Basic model: teams, credits, and income

Setup. One resource type (TPUs). Teams $i \in N$; time runs in market cycles $t = 1, 2, \ldots$ (≈ 30 s). Market determines allocation during the next market cycle.

Primitives. Leadership assigns each team an income $w_i > 0$ (credits per cycle) that reflects its priority.

Balance dynamics. Income accrues to the team's balance every cycle; teams are charged market prices π_t for the chips $u_{i,t}$ they **actually use**:

$$B_{i,t+1} = B_{i,t} + w_i - \pi_t u_{i,t}$$



Savings carry over

Unspent credits are saved: the balance $B_{i,t}$ carries over.

Credits are not chips

Not tied to a chip type or location, and not zero-sum.

Reprioritize via w_i

Change one team's w_i. Adding or removing supply needs no change to anyone's credits.

Market output: entitlements, not assignments

Output. Each cycle the scheduler receives prices π^r and allocations x_i^r.

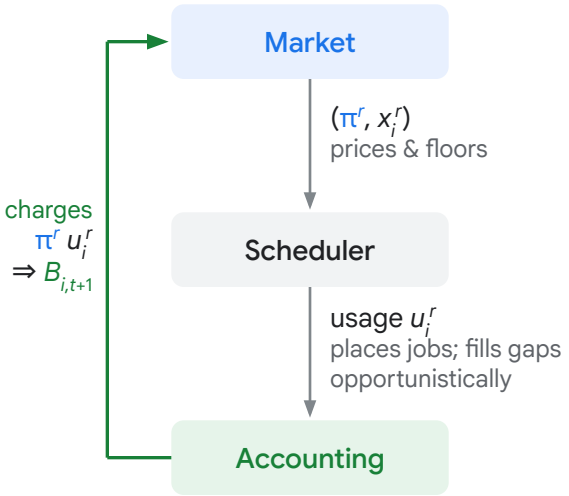
Definition (entitlement). x_i^r is a **floor**, not an assignment: team *i* is entitled to at least x_i^r chips of *r*.

Scheduler. Takes entitlements into account when scheduling jobs, and resolves what the market ignores:

- **integrality:** a job requesting 10 TPUs may be allocated 9 or 8.5;
- **topology:** e.g., a 64-chip job needs a 4 × 4 × 4 cube;
- exact cell placement, …

Accounting. Balances are updated from actual usage u_i^r, not from entitlements:

$$B_{i,t+1} = B_{i,t} + w_i - \sum_r \pi_t^r \, u_{i,t}^r.$$



95+% of market decisions are implemented in practice.

≈5% vs **≈34%** of usage is opportunistic in market pools vs. static pools (allowed whenever $u_i^r \neq x_i^r$).

Numbers from paper Sec. 2.1 and Sec. 3 (occupancy: 5.7% opportunistic in the market pool vs. 33.5% in a conventional pool).

Demand: engineers submit jobs, not bids

Resources. $r = (k, \ell) \in R$: a chip type k at a location $\ell \in \{\text{global}\} \cup \{\text{cells } c\}$. Most jobs are **global** ("run anywhere"); some are **pinned** to a specific cell: data locality, regulation (e.g., HIPAA), …

Jobs. Engineers place demand by **submitting jobs**, as they always have: job j of team i asks for n_j chips of one resource r_j. Jobs wait in the team's priority queue Q_i, so team demand is

$$d_i^r = \sum_{j \in Q_i : r_j = r} n_j$$

Main lever: income leverage *L*. It controls how many credits the team spends per cycle:

$$s_i = \min\left(L\, w_i,\ B_{i,t}\right)$$

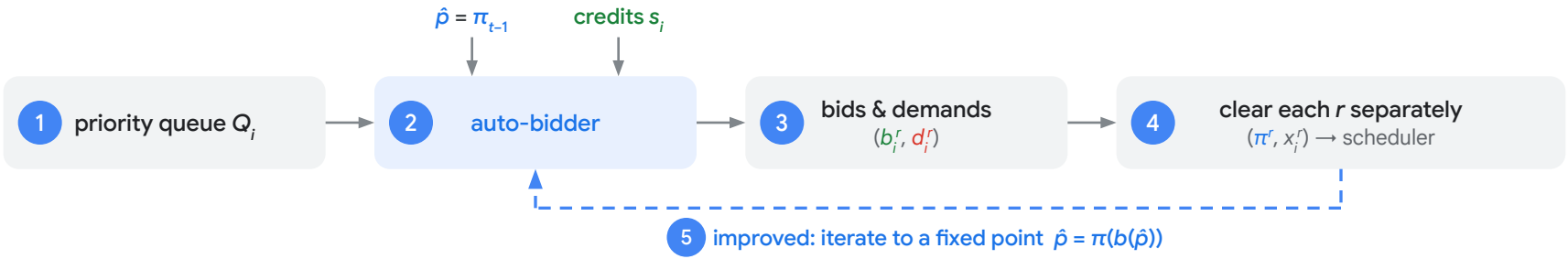
$L < 1$: the team is **saving**;
$L > 1$: it is **drawing down** its savings.

> "We don't want to turn engineers into day-traders."
>
> — common feedback on direct bidding

No direct bids. Engineers never bid themselves.

Limit orders. An engineer **may** attach a limit order to a job; it is unset in the majority of cases.

Every ~30 s: auto-bidding, then market clearing



1 **Inputs.** Each cycle t, for each team i: its priority queue Q_i, price estimates $\hat{p}$ from previous cycles, and its available credits s_i.

2 **Auto-bidder.**

- Walk down the queue, highest-priority job first.
- For each job, estimate its cost from previous cycles' prices and bid that amount on the job's resource.
- Bid only what the team can **sustain for 1 hour**, to prevent thrashing.
- Stop when the credits run out; the remaining jobs wait for a later cycle.

3 **Output.** For every resource r, a bid b_i^r (credits) and a demand d_i^r (chips) for each team.

4 **Clearing.** A **separate market for each resource type** r: a price π^r and allocations x_i^r, which go to the scheduler.

5 **Price estimates.** Initial design: last cycle's prices, $\hat{p} = \pi_{t-1}$. Improved design: iterate within the cycle to a fixed point $\hat{p} = \pi(b(\hat{p}))$.

Market clearing: a Fisher market (Eisenberg–Gale)

Setup. One resource (chip type × location) with supply S. Team i bids b_i credits and has demand cap d_i.

Demand
buy what you can afford

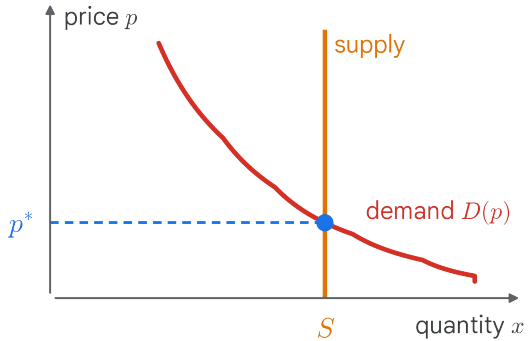
$$D_i(p) = \min\left(d_i, \frac{b_i}{p}\right)$$

Clearing price
lowest price at which demand fits supply; team i gets $x_i = D_i(p^*)$

$$p^* = \min\{\, p \geq 0 \;:\; \textstyle\sum_i D_i(p) \leq S \,\}$$

Same answer as
Eisenberg–Gale: x solves it, and p^* is the multiplier on supply

$$\max_{0 \leq x_i \leq d_i} \sum_i b_i \log x_i \quad \text{s.t.} \quad \sum_i x_i \leq S$$



p^* is where aggregate demand $D(p)$ meets supply S.

Welfare
The objective is budget-weighted Nash welfare.

Computation
Aggregate demand $D(p) = \Sigma_i D_i(p)$ is decreasing, so binary search finds p^* in nearly linear time.

Satiation
Satiated teams ($x_i = d_i$) keep their unspent credits.

Eisenberg & Gale (1959); Fisher-market equilibrium with capped (satiated) linear utilities.

Location: clear cells first, then a global “rising tide”

Setup. Cells c with supply S_c. Each team is **pinned** to one cell or **global**. $D_c(p)$: pinned demand in c; $D_G(p)$: global demand.

Algorithm (rising tide)

1 · Cell prices
each cell clears on its pinned demand

$$p_c = \min\{p : D_c(p) \le S_c\}$$

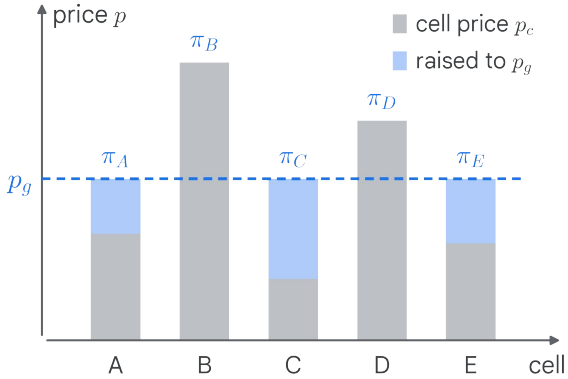
2 · Global price
leftover supply covers global demand

$$p_g = \min\left\{p : \textstyle\sum_c \left[S_c - D_c(p)\right]^+ \ge D_G(p)\right\}$$

3 · Final prices
cheap cells are lifted to the global price

$$\pi_c = \max(p_c,\ p_g)$$

> **Proposition.** π is the Eisenberg–Gale equilibrium on the bipartite network where pinned teams buy only in their cell and global teams buy in any cell.



The tide lifts cheap cells to the global price; hot cells keep theirs.

Global price

Global teams pay the cheapest price, $p_g = \min_c \pi_c$.

Hot cells

Hot cells ($p_c > p_g$) serve pinned demand only.

Computation

Both searches are monotone: bisection in nearly linear time.

2

Evidence from deployment

What the market delivered at Google scale

What the market delivered: four benefits

Agility under fluctuating supply and demand

- Supply moves daily: new capacity lands, chips are clawed back at short notice.
- Demand is bursty: demos, research, conference sprints.
- Reallocation every **~minute**, not every quarter; idle buffers become usable supply.

Shock absorber

- Absorbs big shocks (a marquee LLM launch, a high-priority demo, a cloud surge) with **no manual intervention**.
- Gains and pains are shared by **organizational priority**.
- Temporary, fragmented capacity from serving buffers and holding pools is put to use.

Incentive alignment under intense scarcity

- Prices reward **deferring** in time, location and chip type.
- No “use-it-or-lose-it”: low-value work releases chips.
- Prices double as a public **congestion signal** (company-wide dashboard).

Productivity: less engineering toil

- Far fewer mid-quarter negotiations for capacity.
- Admins just raise a team’s credits: credits are **not zero-sum** and type/location agnostic.
- No need to decide whose chips to take back.

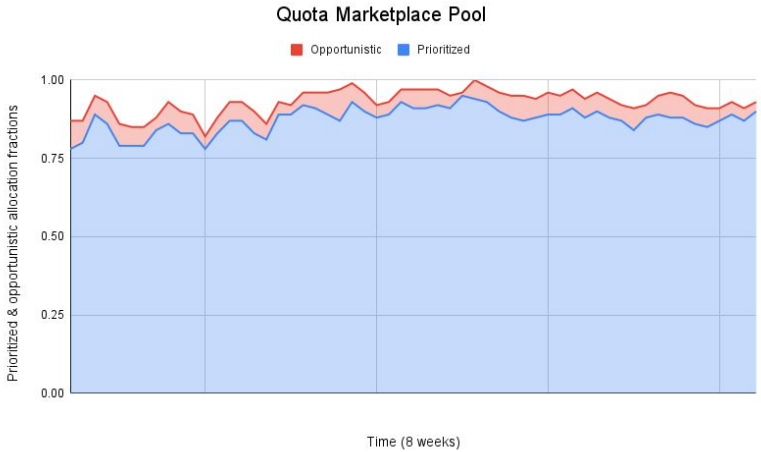
Paper Sec. 1 (benefits and opportunities unlocked by the Quota Marketplace).

Occupancy: market pool vs. static pool

Quota Marketplace pool

93%

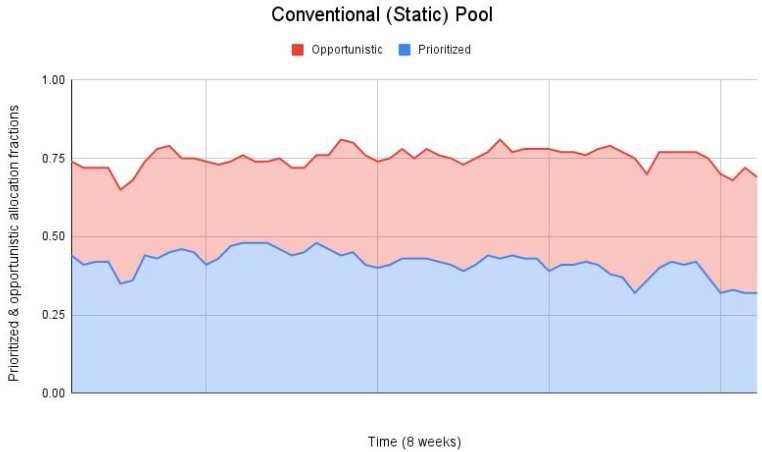
Prioritized + only 5.7% opportunistic



Static pool

75%

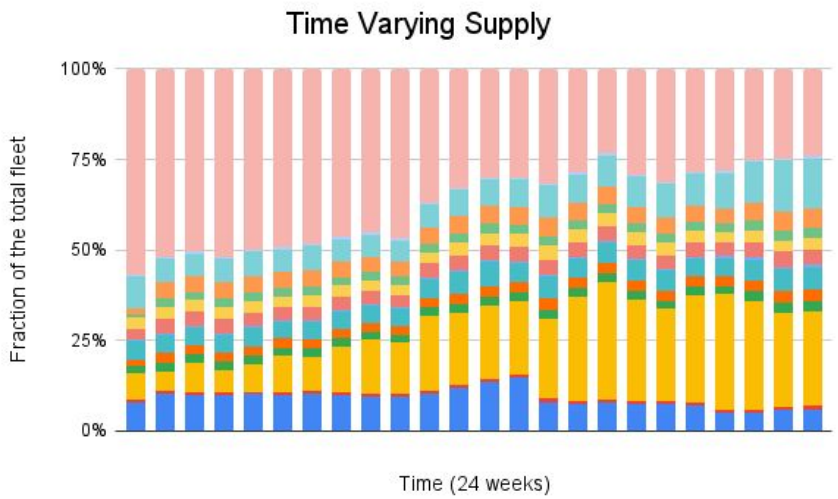
Prioritized + 33.5% opportunistic



~24% more total occupancy, and almost all of it is company-prioritized. Same popular chip type, two pools of comparable size (hundreds of thousands of chips), 8 weeks.

Paper Sec. 3, Figs. 4–5. Blue: prioritized allocation; red: opportunistic use on top of it.

Supply is a moving target



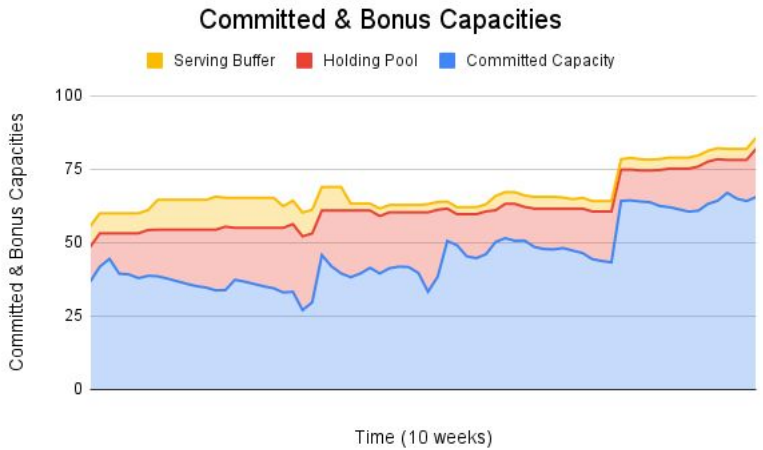
Fleet managed by the market, by chip type (one color per type), over 24 weeks.

- **Week to week:** the mix of chip types in the fleet keeps shifting.
- **Frequent, large changes:** clawbacks to reprioritize, temporary capacity injections, reallocation across chip types, slipping delivery dates.
- **Across space, too:** within a chip type, capacity moves across a 3-digit number of data centers.

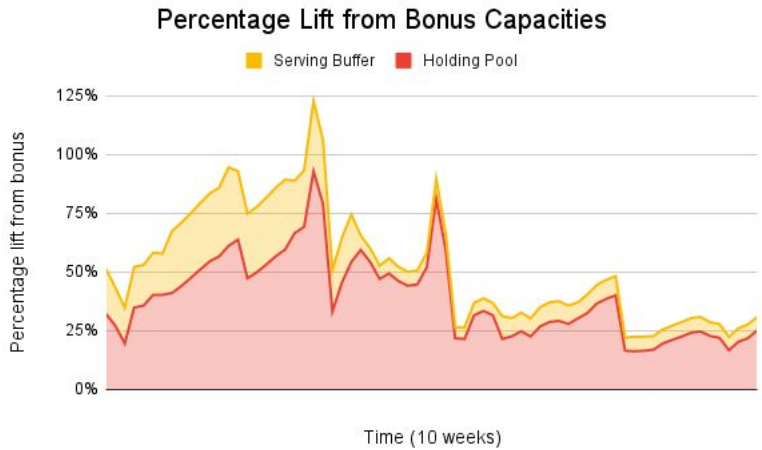
A quarterly plan is stale within weeks; the market re-prices every cycle.

Paper Sec. 3, Fig. 6.

Bonus capacity: idle chips enter the market



Committed capacity plus bonus supply (normalized).



Percentage lift from bonus supply: never below ~25%.

≥25%

additional capacity from transient sources

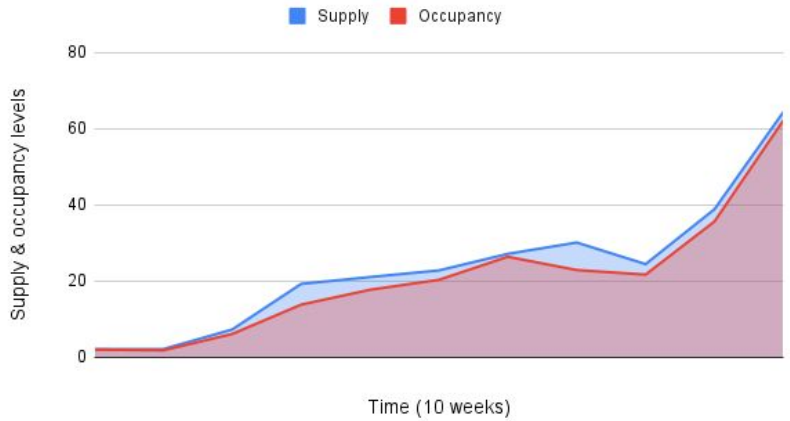
Because the market can claw chips back in minutes, **serving buffers** (reserve for traffic surges) and **holding pools** (hardware delivered before its owner can be charged) are lent to it: consistently hundreds of thousands of chips.

Paper Sec. 3, Figs. 7–8 (10 weeks).

New capacity is found by prices, not announcements

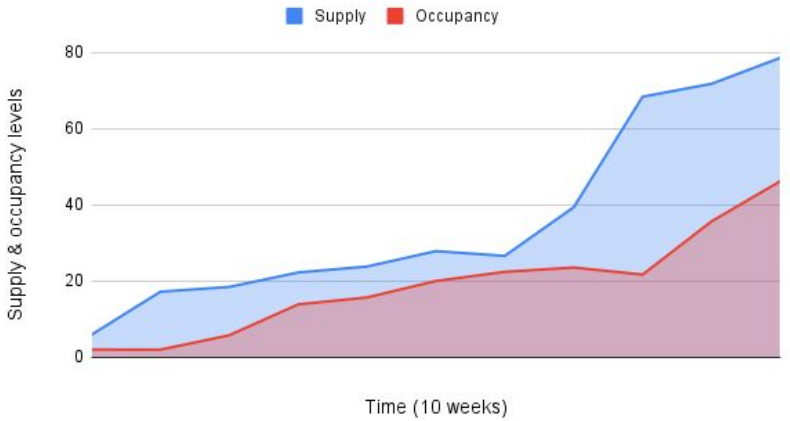
Quota Marketplace pool

Occupancy tracks supply closely, even through spikes



Conventional (static) pool

Occupancy substantially lags supply



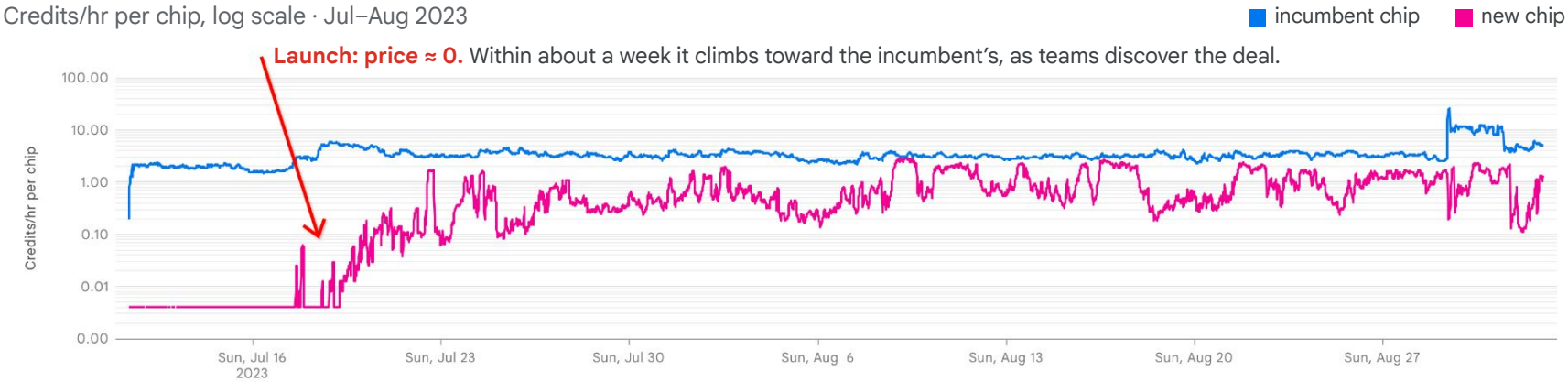
Both pools received new accelerators at the same time. In the market pool the new chips were cheap, and users found and filled them **without any announcement** about their availability or location.

Paper Sec. 3, Figs. 9–10 (10 weeks; levels normalized).

A new chip type, no mandate: users follow the price

Market price: incumbent chip vs. new chip type

Credits/hr per chip, log scale · Jul–Aug 2023



> "[The new chip] is an even more outrageously good deal."
>
> — ML engineer, internal chat, launch week

> "It was 10x cheaper the other day before folks realized it was so readily available to purchase."
>
> — ML engineer, same thread

Internal price dashboard and chat (names and chip codenames removed). Same pattern as paper Fig. 11.

Users optimize credits per unit of work

> "We did some runs and figured out the throughput ratio between [chip types]… So we figured it would then be straightforward to minimize our credit spend by running on the cheapest topology (in terms of credits per throughput)."
>
> — ML researcher, internal chat

> "I know there's been a push throughout research to move to [the new chip] because they are cheaper."
>
> — ML researcher, internal chat

> "I switched from [older chips] to [the new chip] because they were cheaper (although now this is less true than it was initially). But, I don't switch on a daily basis. I would consider switching maybe once a month if one accelerator is consistently a better deal. There is significant switching cost when trying to run controlled experiments."
>
> — ML researcher, internal chat

Prices turn chip choice into a **cost-per-throughput** decision; **switching costs** keep it from being a daily one.

Internal chat (names and chip codenames removed).

Quota Marketplace at fleet scale

50%+

of Google's ML fleet is allocated through the market

Sep 2026

93% vs 75%

occupancy vs. comparable static pools (same chip type, 8 weeks)

Paper Sec. 3, Figs. 4–5

≥ 25%

additional capacity from transient sources: serving buffers and holding pools

Paper Sec. 3, Figs. 7–8 · 10-week low

33.5% → 5.7%

opportunistic share of occupancy: static pool vs. market pool

Paper Sec. 3, Figs. 4–5

~9K

ML developers, running 5M production workloads

Oct 2025

~30 s

market cycle: the whole fleet is re-priced and re-allocated, continuously

p50 cycle < 30 s

Paper Sec. 3 (deployment metrics, Figs. 4–11). Pools compared are of similar size (hundreds of thousands of chips).

3

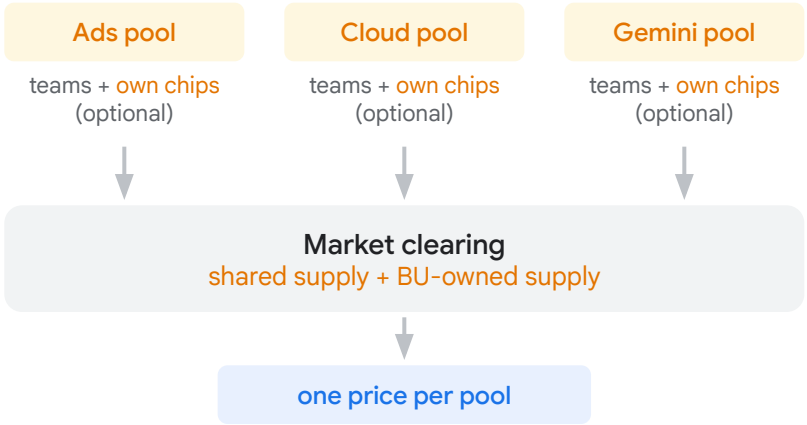
Business units

Governance, owned chips and currencies on top of one market

Business units: governance on top of the market

Why we need it

- **Autonomy:** Ads, Search, Cloud, Gemini, … have their own goals, budgets and governance. Each wants to prioritize its own teams; Google wants to set priorities **across** BUs.
- **Owned hardware:** some BUs already bought TPUs and want **preferential access** to them.
- **Real money:** BUs are charged back in dollars for the purchasing power they receive.



Pool

A business unit's teams, with its own admins, accounting and (optionally) its own chips.

Shared vs. BU-specific supply

Most chips sit in the shared pool; chips a BU already owns are BU-specific supply.

Market weight

A BU's share of total purchasing power, set by Google (and paid for by the BU).

Market weights: every BU mints its own currency

Setup. Business units k, each with teams $i \in k$.

Definition (market weights). Google sets each BU's **market weight** W_k, its share of total purchasing power ($\Sigma_k W_k = 1$). BU admins **mint** credits freely as incomes for their teams:
: $C_k = \Sigma_{i \in k} w_i$ per unit time.

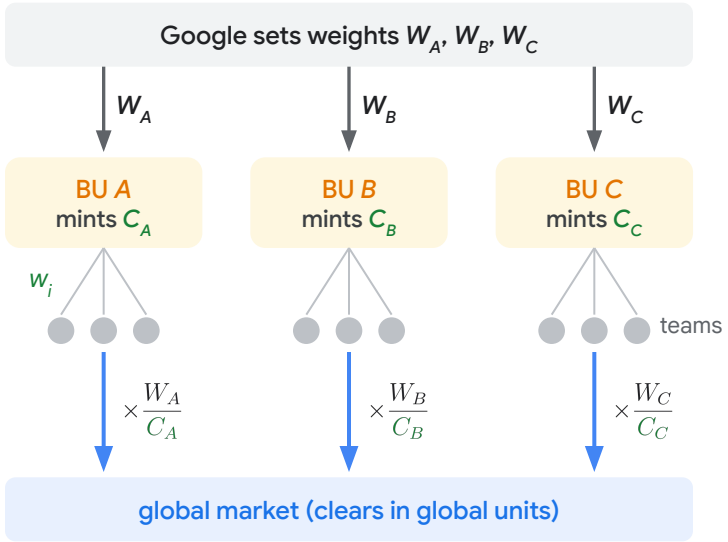
Exchange rate. At clearing, credits convert to a global unit:

$$1 \text{ credit of BU } k \;=\; \frac{W_k}{C_k} \text{ global units}$$

(i) Separation: across-BU priorities are set by Google (W_k); within-BU priorities by the BU (w_i/C_k).

(ii) Example: BU A doubles minting ($C_A \to 2C_A$) with W_A fixed ⇒ each A-credit is worth half; prices in A-credits double. Other BUs: no change.

(iii) The catch: inflation. Minting is free (new teams, raises), so local currencies inflate. It is contained: observed only inside that BU.



Each BU's credits enter the global market at its own exchange rate; Google controls only the weights.

Paper Sec. 2 (Market Weights).

BUs that own chips: two options

BYOC: Bring Your Own Chips

The BU puts its chips into the **shared pool**.

In exchange, it gets **extra market weight**.

Pro: one deep market; chips go to their highest-value use.

Con: no guaranteed access to "its" chips.

BU-first supply

Chips stay as **BU-specific supply**: the BU's teams get first claim, at no cost to the pool.

It buys whatever else it needs from the **shared market**.

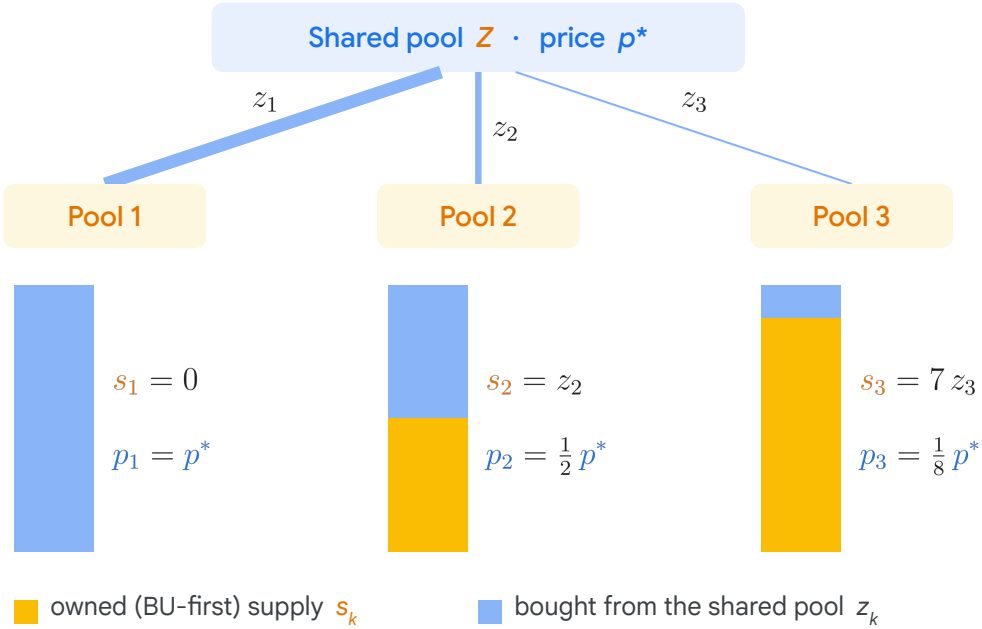
Pro: preferential access to hardware the BU paid for.

Con: fragments supply; needs a **pool-specific price**.

→ pool-specific prices: next slide

> The trade-off: one deep market, or preferential access to the chips a BU paid for.

Pools blend owned and shared supply



s_k supply that pool *k* owns (BU-first)
z_k what pool *k* buys from the shared pool
Z shared capacity: $\Sigma_k z_k \le Z$
p^* shared price; p_k pool *k*'s price

Each pool pays a blended price:

$$p_k = p^* \frac{z_k}{s_k + z_k} \le p^*$$

Own supply is 'free' to a pool's teams; imports cost p^*. The blend $p_k \le p^*$ honors ownership without walling capacity off.

Illustrative: three pools facing the same shared price p^*; the more of its supply a pool owns, the lower its price.

Clearing with BU-first supply: pool prices

Setup: pool k owns s_k chips and may buy $z_k \geq 0$ more from the shared market at price p^*.

1 · Pool clears (MC)

teams buy own supply plus imports

$$\sum_{i \in k} \min\left(d_i,\ \frac{b_i}{p_k}\right) = s_k + z_k$$

2 · Pool breaks even (BB)

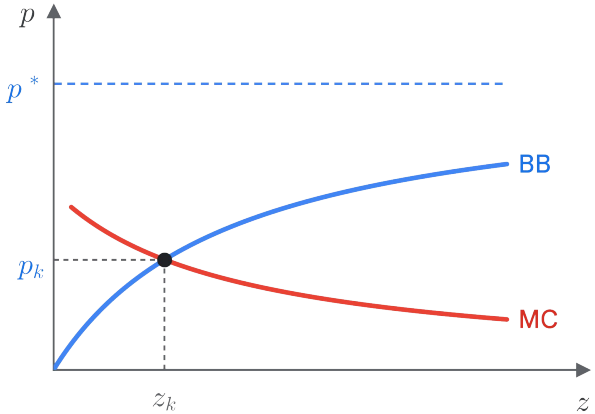
teams pay what the pool pays for imports

$$p_k\,(s_k + z_k) = p^*\,z_k \quad \Longrightarrow \quad p_k = p^*\,\frac{z_k}{s_k + z_k}$$

3 · Shared market clears

pools' remnant demands compete for Z

$$p^* = \min\{\, p \geq 0 \ :\ \textstyle\sum_k z_k(p) \leq Z \,\}$$



Given p^*, the pool equilibrium (z_k, p_k) is where BB meets MC.

- **Owned chips buy a discount:** $p_k \leq p^*$, with equality iff $s_k = 0$.
- **Fast:** $z_k(p^*)$ is decreasing, so nested bisection clears in near-linear time.
- **Allocation:** team i in pool k gets $x_i = \min(d_i, b_i/p_k)$.

Paper Sec. 2, Eqs. (1)–(3); bids normalized to global credits (currencies: next slide).

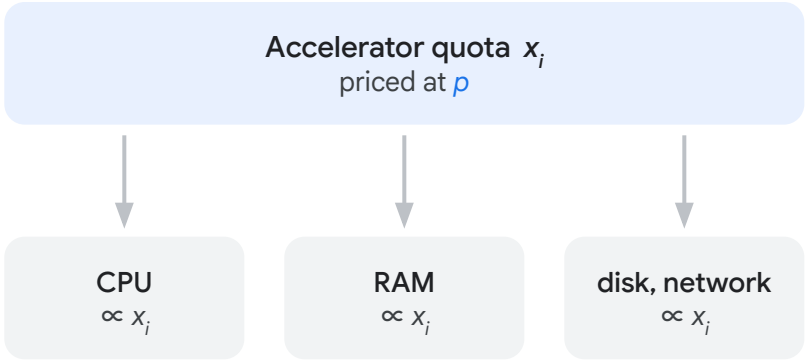
4

Auxiliary resources

CPU and RAM: from attach rates to a joint market with a price vector

Auxiliary resources today: price only the accelerator

- **Principal-resource pricing:** the Quota Marketplace prices only the ML accelerators.
- **Auxiliary resources** (CPU, RAM, disk, network) are allocated **in proportion** to the accelerator quota, via fixed attach rates.
- **Why:** accelerators are the dominant bottleneck for ML training; no combinatorial market; one price signal keeps users' cognitive load low.



Attach rates: fixed ratios, no price.

Trade-off: a combinatorial market could be more efficient; we chose **operability**.

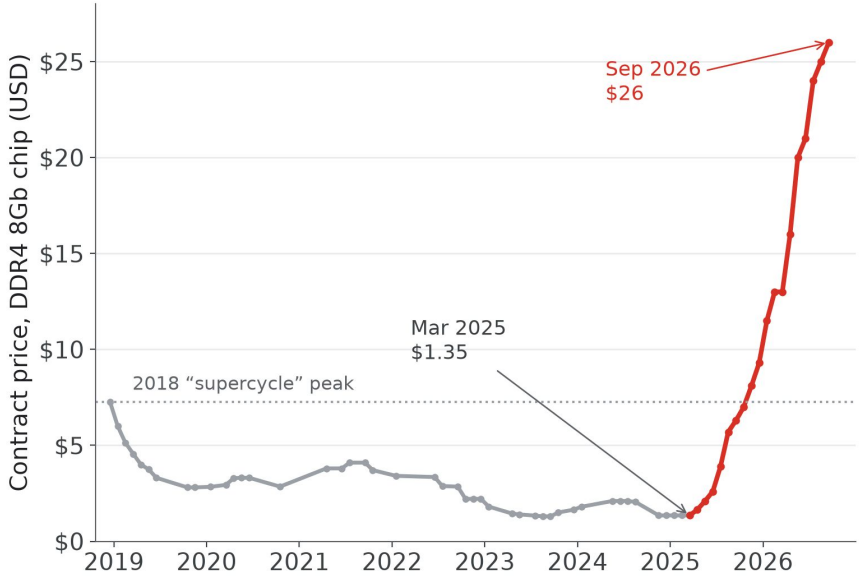
Paper Sec. 2 (Auxiliary Resource Requirements).

CPU and RAM are no longer “free”

DRAM contract price

DDR4 8Gb chip · $1.35 (Mar 2025) ⟶ $26 (Sep 2026)

~19×



- **Treated as abundant:** for years, CPU and RAM were allocated by attach rate, effectively free.
- **AI demand absorbs memory:** DRAM contract prices are up **~19× since March 2025**

Next: price them in the market.

Data: DRAMeXchange/TrendForce, DDR4 8Gb (1Gx8) PC DRAM avg. monthly contract price (USD/chip), as reported in Korean press; months without a verified report omitted.

Many resources: bundles and a price vector

Setup. Resources $k \in K$ (accelerators, CPU, RAM, …) with supply $s \in \mathbb{R}_+^K$. Team i bids a **scalar** b_i credits for a **bundle** $d_i \in \mathbb{R}_+^K$: its chips plus the CPU and RAM its jobs need.

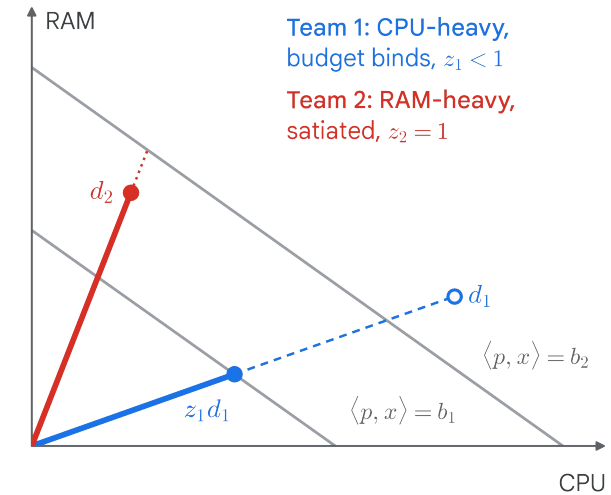
Market equilibrium. A price vector $p \in \mathbb{R}_+^K$ and fractions $z_i \in [0, 1]$ with

$$z_i = \min\left(1,\ b_i \big/ \langle p, d_i \rangle\right) \qquad \text{spend the bid unless satiated}$$

$$\textstyle\sum_i z_i\, d_i^k \le s^k, \quad \text{with "=" if } p^k > 0 \qquad \text{every resource clears}$$

Team i receives $z_i d_i$ and pays $z_i \langle p, d_i \rangle$.

- **Bids stay scalar:** teams do not split credits across resources; the market prices the bundle at $\langle p, d_i \rangle$.
- **$|K| = 1$** is the basic model.



Each team buys along its bundle's ray until its budget or its demand runs out.

Attach rates ≡ pricing only the accelerator ($p^{\text{CPU}} = p^{\text{RAM}} = 0$): right only if CPU and RAM never bind. In equilibrium, abundant resources are free automatically.

Revisiting Eisenberg–Gale with many resources

Proposition (Eisenberg–Gale). (p, z) is an equilibrium iff z solves

$$\max_{z} \ \sum_i b_i \log z_i \qquad \text{s.t.} \qquad \sum_i z_i d_i \le s, \quad 0 \le z_i \le 1,$$

and $p \in \mathbb{R}_+^K$ is the vector of multipliers on the $|K|$ supply constraints.

Dual. Prices minimize a convex potential whose gradient is excess supply:

$$\min_{p \ge 0} \ g(p) = \langle p, s \rangle + \sum_i h_i\big(\langle p, d_i \rangle\big), \qquad \nabla g(p) = s - D(p),$$

with $h_i(u) = \max_{0 \le z \le 1} \{ b_i \log z - z u \}$

and aggregate demand $D(p) = \sum_i z_i(p)\, d_i$.

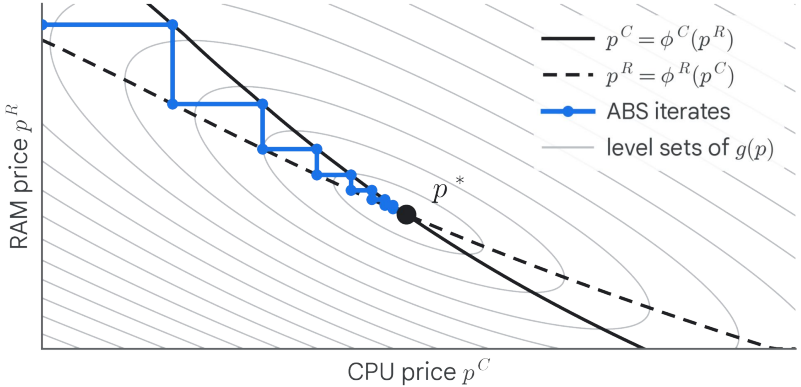
Remarks

- **Same objective** as the basic model: **budget-weighted Nash welfare**; the allocation is Pareto efficient and proportionally fair.
- The equilibrium **exists** and the allocation is **unique**.
- **Clearing the market** = finding $\nabla g(p) = 0$ (with $p^k = 0$ where supply is slack).

	bid	demand	supply	price	cost of demand	allocation	clearing
Basic model	b_i	$d_i \in \mathbb{R}_+$	$s \in \mathbb{R}_+$	$p \in \mathbb{R}_+$	$p\, d_i$	$\min(1, b_i/(p\, d_i))\, d_i$	binary search
Many resources	b_i (still a scalar)	$d_i \in \mathbb{R}_+^K$	$s \in \mathbb{R}_+^K$	$p \in \mathbb{R}_+^K$	$\langle p, d_i \rangle$	$\min(1, b_i/\langle p, d_i \rangle)\, d_i$	alternating binary search

Computing prices: alternating binary search

Problem: clear $|K|$ markets at once, $D^k(p) = s^k$ for all k. With $|K| \geq 2$ there is no single monotone variable to search over.



Each move clears one resource holding the other price fixed; iterates zig-zag to p^*.

Lemma (monotonicity). Fixing the other prices p^{-k}, $D^k(\cdot, p^{-k})$ is continuous and non-increasing in p^k. So the best response $\phi^k(p^{-k})$ that clears resource k is found by binary search (0 if k is under-demanded).

Algorithm (ABS)

1. Start from p_0 (e.g., last cycle's prices).
2. For each $k \in K$, set $p^k \leftarrow \phi^k(p^{-k})$.
3. Repeat step 2; stop when $\|p_{t+1} - p_t\|_1 < \varepsilon$.

Theorem (convergence)

ABS is exact coordinate descent on the convex dual g and converges to the equilibrium p^*. With two resources the rate is linear, $\|p_t - p^*\| \leq C\,\rho^t$, where $\rho = \cos^2\theta$ and θ is the angle between the teams' (weighted) CPU and RAM demand profiles.

- **Diverse bundles** ⇒ small ρ ⇒ **fast convergence**.
- Each step is **one binary search**, $O(n \log(1/\varepsilon))$: the same primitive as today.

From attach rates to a joint market

For teams: nothing changes

Still one scalar bid. The auto-bidder walks down the same priority queue, but costs each job's bundle at the price vector, $\langle \hat{p}, d \rangle$, instead of $\hat{p} \times$ chips.

For the market: one search per resource

Alternating binary search, warm-started from last cycle's prices. CPU and RAM get a price only when they bind.

Ongoing: CPU & RAM for non-ML compute

The same design for best-effort CPU and RAM jobs, where neither resource dominates and attach rates do not apply.

> Same Fisher-market logic, but prices become vectors; resources that are not scarce stay free.

Conclusion: prices instead of quotas

1 · BASIC MODEL

Teams get credits, not chips

Every ~30 s a Fisher market (Eisenberg–Gale) sets prices; an auto-bidder over each team's priority queue means no one has to day-trade.

2 · EVIDENCE FROM DEPLOYMENT

Prices beat static quotas

Occupancy **75% → 93%**; opportunistic share **33.5% → 5.7%**; **≥ 25%** bonus capacity. New chips and new capacity are found by prices, not mandates.

3 · BUSINESS UNITS

Governance on top of one market

Market weights, BU-owned supply and pool prices keep BU autonomy without splitting the market.

4 · AUXILIARY RESOURCES

Prices become vectors

As CPU and RAM become scarce, Eisenberg–Gale still applies, and alternating binary search computes the price vector.

> A market without money can allocate scarce compute in minutes, not quarters, with priorities still set by the firm.

Lessons for market design, and open questions

LESSONS FROM PRACTICE

Simplicity drives adoption

One scalar bid, an auto-bidder, and one price per chip.

Funny money works

When budgets are scarce and keep arriving: credits as income, market weights as the policy lever.

Prices are a public signal

A company-wide price dashboard steers demand in time, place and chip type.

Governance as a layer

BUs keep autonomy through their own currencies, on top of one market.

OPEN QUESTIONS

Incentives

With credit banking, deadlines and auto-bidders, how close to truthful is the dynamic market?

Stability vs. speed

How fast should prices move, given switching costs and preemption?

Richer demand

Topology, complements, and bundles that are not rays.

Closing the loop

Can internal prices guide capex, i.e. which chips to buy and where?

Paper: arXiv:2607.09802

Thank you

Questions?

arXiv 2607.09802

Joint work with

Google Research + Google Cloud

Balasubramanian Sivan
Mihai Tiuca
Ian McFarlane
Vasilis Gkatzelis
Nehal Mehta
Soheil Hassas Yeganeh
Vahab Mirrokni
Amin Vahdat

...and every team that trusted a market with its compute.



Appendix

Where we stand: efficiency and fairness across mechanisms

APPENDIX

Where we stand

Efficiency and fairness of four allocation mechanisms, with equal vs. heterogeneous job values.

	Equal values		Heterogeneous values	
Mechanism	Pareto efficient	Max-min fair	Pareto efficient	Max-min fair
Static allocation	✗	✗	✗	✗
Static + opportunistic	✓	✗	✗	✗
Chip-hours (Karma)	✓	✗	✗	✗
Market with dynamic prices	✓	✓	✓	✓*

* Approximate max-min fairness, converging to exact as teams' average prices equalize (Paper Thm. 4).

A market with an artificial currency and dynamic prices is Pareto efficient (first welfare theorem) and approximately max-min fair.