Global Innovation Institute

GII Research | Physical AI and Commercialization | Episode 2

The Inference Economy

A sequel to The Restaurant as a Physical AI Platform | SGLang Summit 2026

Inference becomes commercially useful when reliable systems turn model capability into measurable outcomes.

GII Independent Research | AI Infrastructure and Operating Economics
November 2026 · Standalone edition 2.0 · Field record: November 12–13, 2026

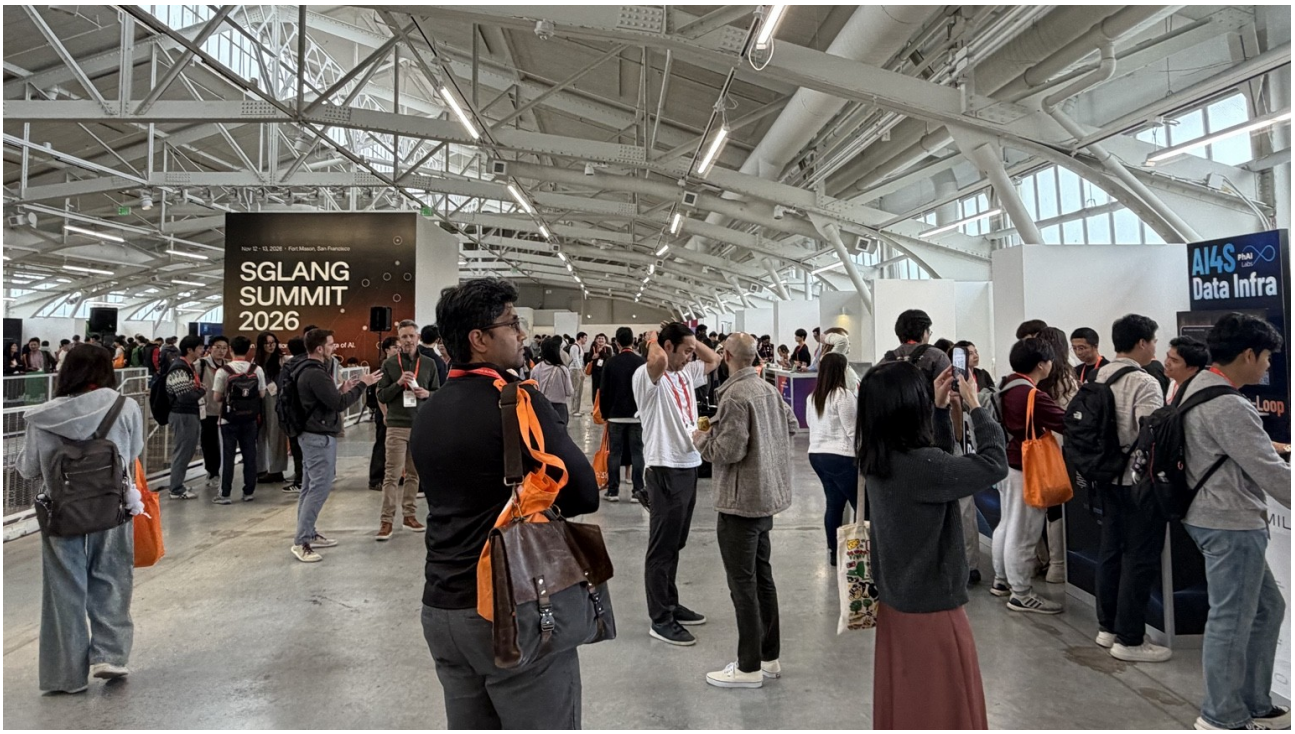
Dr. Cervantes Lee | Global Innovation Institute | clee@giinow.org
Hospitality Business Professor · Principal Real Estate Investor · AI Infrastructure Research

Original field reporting and analysis for technology executives, restaurant and retail operators, and AI infrastructure readers.

The argument

A fourfold improvement in an answer's streaming speed can leave system throughput unchanged. A restaurant campaign can attract more purchases yet reduce contribution. In both cases, the important question is what improved—and whether that improvement reaches the outcome the business values. This second episode follows that question from SGLang Summit's presentations and exhibition floor to the customer database, the serving engine and the data-center rack.

Drawing on my field photographs, session records and supporting public documents, I explain how inference becomes a working service. The practical bridge is a restaurant business with a hypothetical 300,000 customers and 100,000 identifiable profiles: how can it understand needs, estimate customer value and make better decisions about whom to reach?



Picture 1 | SGLang Summit exhibition floor, Fort Mason. Author's field photograph.

Reader Navigation — From the Conference Floor to the Customer Decision

The executive summary states the argument. Chapters 1–2 establish the connection to Episode 1 and explain the industry's participation. Chapters 3–6 follow the work through serving, hardware, applications and learning. Chapter 7 examines the economics; Chapter 8 develops the restaurant customer-intelligence example. Chapter 9 draws the implications for operations, robotics and data centers. The glossary and endnotes support independent reading of this volume.

Table 1 | Research chapters and recommended reading path

Read for	Start here	Decision supported
The central argument	Executive summary; Chapters 1–2	Why inference matters to operating businesses
Serving mechanics	Chapter 3	Which clock, workload and constraint improved?
Hardware and memory	Chapter 4	What configuration delivers the required service?
Applications and learning	Chapters 5–6	What counts as a useful, accepted result?
Commercial economics	Chapter 7	Who pays, and which costs and risks remain?
Customer intelligence	Chapter 8	How do profiles become measurable customer decisions?
Cross-industry implications	Chapter 9	What evidence would justify the next deployment?
Definitions and evidence	Appendices A–B; endnotes	How to read the terms, claims and public references

Read Chapters 3 and 8 together for the bridge from inference engineering to customer decisions. Hardware readers can begin with Chapter 4; operators and commercial readers can begin with Chapters 7–9.

Table 2 | How to interpret report evidence

Evidence type	Treatment in this report
Field photograph	Documents what was visible; it does not establish deployment scale or financial success.
Speaker or vendor claim	Attributed to the presentation and kept within its stated scope.
GII interpretation	Identifies the commercial implication and the evidence needed to test it.
Illustrative calculation	States assumptions; it is not a measured benchmark or a company forecast.
Public background documents	Original company disclosures and technical documents support dated background facts.

Table of Contents

Executive Summary — The Inference Economy....8
Technical progress matters when it changes an operating result....8
Author and Field Reporting Record....9
On site at SGLang Summit....9
1 From the Restaurant as a Physical AI Platform to the Inference Economy....10
1.1 The operating question carries forward....10
1.2 What the conference floor made visible....10
1.3 The decision connects the model to the business....11
2 Why the Industry Gathered Around Inference....12
2.1 A shared engine supports different businesses....12
2.2 Intel, AMD and NVIDIA: participation has an operating logic....12
2.3 Matcha, a long line and developer attention....13
2.4 Intel: software commitments give leadership presence meaning....13
2.5 AMD: the developer is part of the route to deployment....14
2.6 Hugging Face: an optimization must reach the application....15
2.7 IBM Spyre: integration makes new hardware usable....16
3 What Happens Between a Question and an Answer....17
3.1 What inference means....17
3.2 Prefill: processing the input....17
3.3 Decode: generating the output....18
3.4 Horace He: ask what the percentage measures....18
3.5 Four times faster can mean unchanged throughput....20
3.6 A conditional formula for the scheduling benefit....20
3.7 What PD separation must pay for....21
3.8 Four ways an inference workload can grow....21
3.9 Baseten: measure the stack of changes....22
3.10 Traction Layer: predictable service is a product....23
4 Why the Whole System Matters....24
4.1 Memory capacity and movement shape the service....24
4.2 Tencent TACO: managing state across the serving system....25
4.3 Netpreme: extending the memory hierarchy....26
4.4 WiCi InferSSD: capacity with a bandwidth tradeoff....27
4.5 NVIDIA: reading the rack from infrastructure to silicon....28
4.6 AMD: from accelerator board to complete server....32
4.7 Uni³Sim: test the system design before buying it....35
5 From Fast Tokens to Useful Work....36
5.1 Measure the complete workflow....36
5.2 Arcee: the agent still has work to finish....37
5.3 Palo Alto Networks: a late security answer can miss its purpose....38
5.4 Phala: privacy must survive the execution path....39
5.5 Core Automation: make research experiments easier to run....40
6 Inference That Searches, Learns and Verifies....41
6.1 SGLang and Miles: serving feeds the learning loop....41
6.2 Miles: the learning loop needs an operating record....42
6.3 Heterogeneous RL: coordinate the devices and the policy....42
6.4 ApprenticeBench: evaluate a continuing job....44
6.5 Axiom Math: generation joined to formal verification....45
6.6 Axiom's financing and the commercial test....46
6.7 Apodex: investigate the evidence behind agreement....47
6.8 Krea: the reward must preserve the intended image....48

6.9 MiniMax: coordinated tools expand the output....49
7 Where the Economics Work....50
7.1 The bottleneck can move—and so can the margin....50
7.2 SF Compute: value capture depends on the scenario....51
7.3 Hyperbolic: match committed capacity to useful work....52
7.4 SkyPilot: coordination determines productive use....53
7.5 Capacity, utilization and contribution are different measures....53
7.6 Make the revenue assumptions visible....54
7.7 Trust and reliability belong in the commercial offer....54
8 Understanding 100,000 Profiles in a 300,000-Customer Business....55
8.1 Reconcile the customer population....55
8.2 Build a customer evidence table....55
8.3 Understand needs before choosing an algorithm....56
8.4 Estimate customer value with an explicit horizon....56
8.5 Learn from the past without leaking the future....57
8.6 Where language-model inference adds value....57
8.7 Reach customers without confusing response and impact....58
8.8 Calculate the campaign economics....58
8.9 Match the serving design to the decision....59
8.10 A ninety-day research sequence....59
8.11 The complete customer-intelligence loop....60
9 Implications for Restaurants, Robotics and Data Centers....61
9.1 What changes for a restaurant operator....61
9.2 Generalist: inference meets a physical action....61
9.3 Data-center capacity must become a defined service....62
9.4 The questions I would take into the next deployment....62
9.5 The next research agenda....62
Appendix A Working Glossary....63
Working glossary....63
Working glossary continued....63
Appendix B Reporting Method and Evidence....64
Editorial and publication record....64
Endnotes and Public References....65

List of Tables

Table 1 | Research chapters and recommended reading path....2
Table 2 | How to interpret report evidence....2
Table 3 | The operating stack....12
Table 4 | Four questions, four methods....17
Table 5 | The speed vocabulary....18
Table 6 | Idealized allocation in the 75/25 example....19
Table 7 | Same output, different user experience....20
Table 8 | Sensitivity to the original prefill time share....21
Table 9 | PD deployment decision record....21
Table 10 | Four growth axes....22
Table 11 | Illustrative end-to-end task....36
Table 12 | Axiom’s disclosed financing rounds....46
Table 13 | Hypothetical monthly gross revenue: 100 accelerators × 720 hours....54
Table 14 | Population reconciliation....55
Table 15 | Minimum evidence schema....56
Table 16 | Illustrative twelve-month contribution....57
Table 17 | A practical model development sequence....57
Table 18 | Illustrative campaign comparison....58
Table 19 | Illustrative contribution bridge....59
Table 20 | Workload-to-architecture mapping....59
Table 21 | Glossary A Serving and hardware....63
Table 22 | Glossary B Learning trust and customer analysis....63
Table 23 | Evidence distinctions....64

List of Pictures

Picture 1 | SGLang Summit exhibition floor, Fort Mason....1
Picture 2 | Dr. Cervantes Lee at the summit sponsor wall....9
Picture 3 | Participation badge — EXPO PASS, QR concealed....9
Picture 4 | Conversations around the conference stage....11
Picture 5 | Nebius / Token Factory matcha counter....13
Picture 6 | Lip-Bu Tan on an on-site screen relay....13
Picture 7 | The audience watching Intel leadership....13
Picture 8 | Vamsi Boppana — official summit portrait....14
Picture 9 | Adarsh Shirawalmath on kernels....15
Picture 10 | Kernel portability and distribution....15
Picture 11 | Antoni Viros i Martin on IBM Spyre....16
Picture 12 | IBM Spyre software integration....16
Picture 13 | Horace He explaining inference scheduling....18
Picture 14 | Prefill and decode allocation across four engines....19
Picture 15 | Two models further along the engine timeline....20
Picture 16 | Tri Dao and Alexander Korte presenting Baseten....22
Picture 17 | The Baseten optimization stack....22
Picture 18 | Presenters at the Ghirardelli Square companion event....23
Picture 19 | Upstream releases and enterprise approval....23
Picture 20 | Xibo Sun presenting Tencent TACO....25
Picture 21 | Tencent TACO resource management....25
Picture 22 | Nikita Lazarev on memory tiering....26
Picture 23 | Memory tiering beyond HBM....26
Picture 24 | Zili Meng presenting WiCi InferSSD....27
Picture 25 | SSD capacity and its bandwidth gap....27
Picture 26 | The other side of the NVIDIA rack....28
Picture 27 | Front compartments perform different jobs....28
Picture 28 | GB300 compute tray with its identifying sign....29
Picture 29 | Power wires, liquid fittings and fans in GB300....29
Picture 30 | Rubin compute tray and its covered processing zones....30
Picture 31 | Rubin switch tray with management and cooling regions....30
Picture 32 | Vera CPU tray with eight numbered processor positions....31
Picture 33 | Vera CPU packages and surrounding memory arrays....31
Picture 34 | Eight AMD accelerator modules above the shared baseboard....32
Picture 35 | The MI350X specification sign establishes the product family....32
Picture 36 | Inside the opened silver server....33
Picture 37 | M.2 storage and the two unresolved wired modules....33
Picture 38 | Motherboard wiring, memory latches and local cooling....34
Picture 39 | The heat-sink assembly above an AMD accelerator....34
Picture 40 | Zhenhua Zhu on Uni³Sim....35
Picture 41 | Uni³Sim system configurations....35
Picture 42 | Lucas Atkins at the lectern....37
Picture 43 | Fast Tokens, Slow Agents....37
Picture 44 | Why language models for security inference....38
Picture 45 | Hang Yin presenting confidential inference....39
Picture 46 | Confidential execution and attestation....39
Picture 47 | Francis Zhang on research infrastructure....40
Picture 48 | Inference and research....40
Picture 49 | Ying Sheng — official summit portrait....41
Picture 50 | Serving and reinforcement-learning feedback....41

Picture 51 | Miles session title naming Zhichen Zeng....42
Picture 52 | Environments for Miles post-training....42
Picture 53 | Train on one chip, roll out on any — session title....42
Picture 54 | Every inference server as a rollout server....43
Picture 55 | ApprenticeBench title naming Xiang Deng....44
Picture 56 | ApprenticeBench learning across work....44
Picture 57 | Tobias Gessler and Rishi Malhotra presenting Axiom Math....45
Picture 58 | Axiom Math training and proof infrastructure....45
Picture 59 | Generation, formal verification and feedback....46
Picture 60 | Simon Shaolei Du presenting Apodex....47
Picture 61 | Reasoning as a team sport....47
Picture 62 | Sangwu Lee presenting Krea....48
Picture 63 | Krea reinforcement-learning update....48
Picture 64 | Ryan Lee on the Open Stage....49
Picture 65 | Modalities and bandwidth of expression....49
Picture 66 | Ethan Anderson on value capture....51
Picture 67 | Where will margin live — the value-capture matrix....51
Picture 68 | Jasper Yue Zhang on the compute grid....52
Picture 69 | The compute-grid architecture....52
Picture 70 | Zhanghao Wu presenting SkyPilot....53
Picture 71 | A simulated control workload in SkyPilot....53
Picture 72 | The restaurant customer-intelligence operating loop....60
Picture 73 | Generalist GEN-1.5 session title....61
Picture 74 | Physical prompting and learned model weights....61

Executive Summary — The Inference Economy

Technical progress matters when it changes an operating result

A fourfold improvement can leave the headline business measure unchanged. In Horace He's simplified prefill/decode example, each user's answer streams four times faster while aggregate output remains 160 tokens per second. In my restaurant campaign example, a higher purchase rate produces lower contribution after discounts and communication costs. Both results are internally consistent. The lesson is to identify the constraint that improved and follow its effect through the complete workflow. [1]

This is Episode 2 of GII's Physical AI and Commercialization research sequence. The Restaurant as a Physical AI Platform asked how AI capability earns a place in a real business. At SGLang Summit 2026, I followed the next question: what must happen between a trained model and a reliable service for a customer, employee or machine? Inference is where a model's capability is repeatedly applied—and where waiting time, cost and failure become operating facts. [2]

The summit's technical discussions point toward a wider commercial contest. Hardware suppliers need software that makes their equipment usable. Serving platforms need to turn specialized engineering into repeatable service. Application builders need useful results within a deadline and a budget. Open-source infrastructure can support all three groups while leaving substantial room for competition in integration, support and execution.

Prefill and decode expose this contest in miniature. Processing a prompt and generating an answer place different demands on the system. Separating those stages can improve the user's experience, but the result depends on workload, memory, data transfer and scheduling. "Faster inference" therefore needs a denominator: faster first response, faster streaming, more completed work, or lower cost at a specified quality.

The restaurant application makes the distinction concrete. A business with a hypothetical 300,000 customers and 100,000 identifiable database profiles needs sound records, predictive models, selective language-model analysis and controlled experiments. Its goal is to understand needs and improve customer decisions. No customer dataset was supplied or analyzed for this report; the worked examples show how an operator could design that investigation. The industry cases and explanations that support the argument are included here, with session attribution and public-source notes at the end.

Author and Field Reporting Record

Dr. Cervantes Lee
Global Innovation Institute

I approach inference through the operating question developed in The Restaurant as a Physical AI Platform: when does an impressive capability become a reliable, economically useful part of a business? My work in restaurants, education, real estate and investment makes that question hard to separate from the people, buildings and systems required to deliver the result.

At Fort Mason, I followed that question through talks, demonstrations and equipment displays. This publication brings those observations into one account of the inference economy. The photographs place the argument in its reporting context; the technical explanations and calculations make it useful beyond the conference.



Picture 2 | Dr. Cervantes Lee at the summit sponsor wall. Author's field photograph.

On site at SGLang Summit



Picture 3 | Participation badge — EXPO PASS, QR concealed. AI-assisted rotation and QR masking; original badge photograph retained.

The badge records the author's name, Global Innovation Institute, SGLang Summit 2026, November 12–13 and Fort Mason, San Francisco. It is reproduced as a participation record, with the QR credential concealed.

Reporting, photography and analysis: Dr. Cervantes Lee, Global Innovation Institute. AI tools assisted with organization, drafting and document production. Session statements and company disclosures remain attributed to their originators; analytical judgments and illustrative calculations are identified as my work. The endnotes describe the evidence behind each subject, and Appendix B explains the reporting method.

1 From the Restaurant as a Physical AI Platform to the Inference Economy

1.1 The operating question carries forward

My starting question is practical: what becomes possible for an operator when AI inference becomes faster, more reliable and less expensive? My work connects restaurants, retail, physical AI and infrastructure. These domains meet whenever a company turns observations into decisions and must support those decisions with equipment, software, people and capital.

Episode 1, The Restaurant as a Physical AI Platform, established three units of commercialization: the site as the unit of economics, the task as the unit of learning, and the network as the unit of scale. Episode 2 carries that framework into inference. The site bears service costs and operating constraints; the task defines what the model must accomplish; the network determines whether a useful result can be reproduced across locations and customers. [2]

A restaurant executive does not need to manage the business in tokens. The executive needs to know whether analysis improves retention, whether a customer can be reached through an authorized channel, whether an offer adds contribution after discounts, and whether a system can complete its work before a decision is due. Tokens, batches and GPU memory become relevant when they explain the cost or feasibility of those outcomes.

This is also why an infrastructure conference is relevant to a researcher with a real estate and capital allocation background. A rack is an investment in a particular operating capability, subject to power, cooling, network and software constraints. More installed equipment is useful only if it can perform saleable or internally valuable work. The route from hardware to contribution is the subject of this report.

My interpretation is that inference belongs in the study of operations because it connects information to a repeated decision or action. The presentations approached that connection from different directions. Their disagreements and tradeoffs are useful: they show why no single chip specification or model score can explain the whole business.

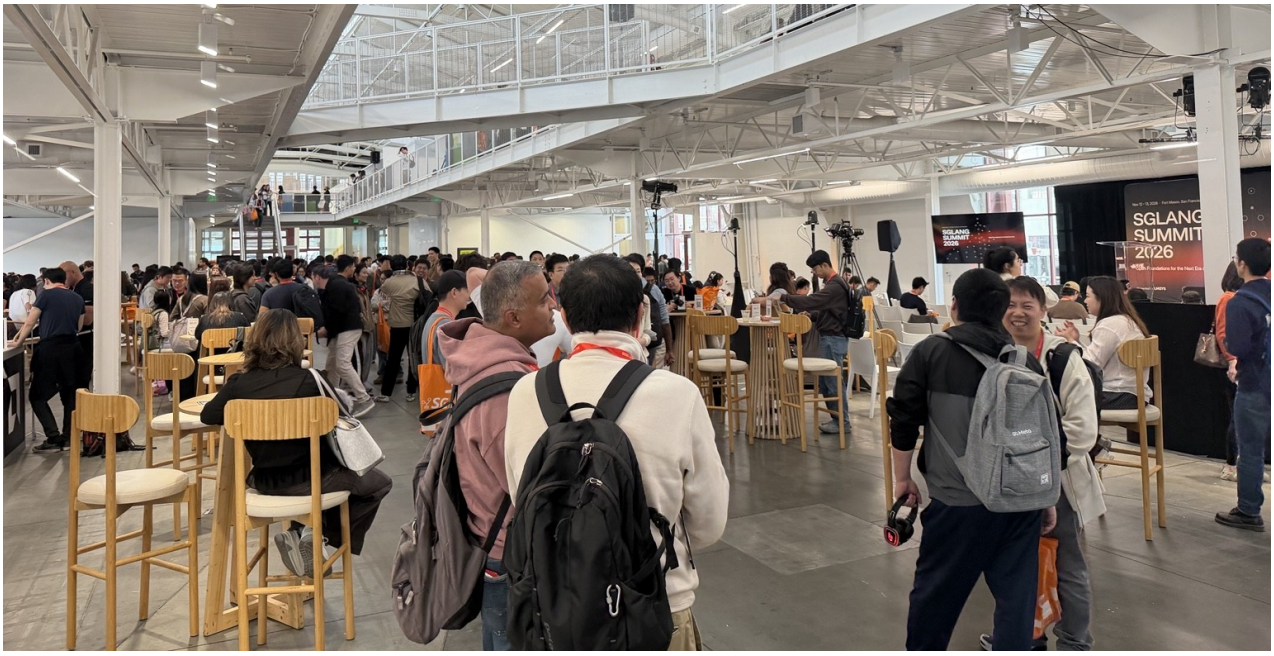
1.2 What the conference floor made visible

Inside Fort Mason, groups gathered around equipment displays, demonstration screens and conversation areas. The exhibition floor made an abstract subject tangible. Behind the discussion of tokens were engineers trying to make a service run, vendors trying to win those engineers' attention, and racks whose pipes and cables made the physical demands visible.

The photographs serve as a guide through that setting. A presenter beside a title establishes the topic under discussion. A photographed slide shows the claim or mechanism being explained. A hardware close-up lets us distinguish a compute tray from a switch and follow the connections that support both.

That visual record also sets limits. An obscured label cannot establish a product generation, and a busy room cannot establish market share. Where an identification matters, I pair the photograph with the relevant display label or public technical document and retain any uncertainty that remains.

The field record gives this report its starting point. The following chapters ask what the observations mean for companies buying, building and operating AI.



Picture 4 | Conversations around the conference stage. Author's field photograph.

1.3 The decision connects the model to the business

The report follows three linked questions. What must the system do? What prevents it from doing that reliably and economically? What evidence would show that the constraint has improved? These questions organize the company examples across the chapters, rather than treating each presentation as a separate destination.

For a restaurant, the answer may be a more useful customer decision. For a research team, it may be a checked proof or an experiment completed sooner. For an infrastructure buyer, it may be a defined service delivered within a power, latency and cost envelope. In every case, the task and its acceptance test come before a persuasive performance headline.

2 Why the Industry Gathered Around Inference

The industry's presence around a shared serving engine raises a commercial question: what does each participant gain when inference becomes easier to deploy? The answer differs for a chip supplier, an open-source maintainer, a cloud operator and an application builder. Their common interest is a working path from model to service; their opportunities to earn revenue lie at different points along that path.

2.1 A shared engine supports different businesses

A model contains learned numerical parameters. An application supplies instructions, inputs and a purpose. A serving system organizes the calculations needed to answer requests. A learning system can later update the model using data and feedback. These layers are connected, but they do not perform the same job.

SGLang handles model serving; Miles organizes reinforcement-learning work that repeatedly calls on that serving capacity. A model generates attempts, an environment or verifier evaluates them, and training changes the model. Release to production requires a separate evaluation decision. The loop is valuable when its feedback measures behavior worth learning. [3]

This helps explain the range of companies around the summit. Hardware suppliers need their processors supported by useful software. Model companies need reliable paths from their weights to applications. Cloud platforms need workloads that can be scheduled and sold. Application builders need the layers to work together without making every customer become a systems engineer.

Shared software can expand the market for several competing suppliers at once. It lowers some adoption barriers while moving competition toward deployment quality, availability, support, integration and workload-specific performance. Open collaboration is compatible with commercial rivalry.

The SGLang Consortium offers another form of coordination: participants can improve a common technical foundation while maintainers retain technical acceptance decisions. A sponsor's logo identifies participation, not ownership of the code or control of the project. The commercial interests around a shared engine can remain distinct. [3]

Table 3 | The operating stack

Layer	Principal job	Business question
Application	Select a useful task	Who benefits from the answer?
Model	Map inputs to outputs	Is capability sufficient?
Serving	Schedule execution and reuse	Can it meet the deadline?
Infrastructure	Provide compute and movement	Can useful work be delivered reliably?
Evaluation	Test outputs and outcomes	Did the process improve?

2.2 Intel, AMD and NVIDIA: participation has an operating logic

The question of why senior hardware leadership and competing chip suppliers would gather around inference software has a straightforward economic answer. Hardware demand depends on what customers can run successfully. A processor becomes commercially useful through models, kernels, compilers, serving engines, networks and support arrangements that enable deployment.

I photographed Lip-Bu Tan on an on-site screen relay, with an audience following the conversation. Intel's public announcements connect that presence to collaboration with RadixArk on SGLang optimization and participation in the SGLang Consortium. Those commitments explain the relevance of the appearance; I do not have a complete transcript of Tan's remarks from which to reconstruct his argument. [4]

AMD's developer-program display brings the adoption question into view: who will build, test and maintain applications on the hardware? Its participation combined leadership visibility, engineering activity and developer outreach. The equipment photographs, examined in Chapter 4, show why an accelerator board, a CPU server and a complete service are different layers of the purchase. [5, 6]

At NVIDIA's display, the system proposition was physically visible. Compute trays, switches, power components, cooling connections and rack structure performed different jobs. My reading is that the commercial product extends to the system's useful service. A switch tray and a GPU compute tray can look similar while doing fundamentally different work. [7]

All three companies have reasons to make deployment easier. NVIDIA's display emphasizes integration; AMD's material links hardware with developer adoption; Intel's announcements connect leadership visibility with software collaboration. These observations explain participation. Ranking the alternatives would require matched workloads, delivered pricing and operating data.

2.3 Matcha, a long line and developer attention

The Nebius-branded matcha counter brought out the restaurant researcher in me. I loved the drink, although the very long line tested my patience. Apparently, even at an AI conference, I cannot switch off the part of my brain that notices the customer experience. Beneath the Nebius / Token Factory signage, a neo-cloud infrastructure company was introducing itself through a familiar hospitality ritual. The cup gave me a pleasant reason to remember the brand. Its inference performance and pricing would need their own evaluation.



Picture 5 | Nebius / Token Factory matcha counter. Author's field photograph.

Technical adoption also has a human side. Engineers must notice a product, understand its interfaces, obtain access, test a workload and decide whether to depend on it. A conference brings those steps closer together. A good drink can open a conversation; the engineering trial must earn the next one.

For a restaurant researcher, the scene is a reminder that attention, service and customer understanding travel across industries. Chapter 8 takes that curiosity into the database: how can a company learn more about its customers and make a better decision about whom to reach, with what offer and at what cost?

The drink makes an inviting introduction. The deeper commercial question begins when a developer tries the service: how much work is required to install it, measure it, recover from failure and keep it running? The next cases show why distribution and compatibility matter alongside chip performance.

2.4 Intel: software commitments give leadership presence meaning



Picture 6 | Lip-Bu Tan on an on-site screen relay. Author's conference photograph. [4]

The on-site screen-relay photograph establishes Lip-Bu Tan's visible participation and the audience setting. A complete speech transcript is not available in my record. Intel's public collaboration announcements therefore provide the firmer basis for discussing its engineering commitments. [4]

The commercial interpretation is that hardware suppliers have an interest in the software ecosystems that make their products deployable. Inference workloads can run across different processor types and system designs; practical adoption depends on compatibility, performance and support for the intended application.

The significance of leadership participation is the strategic attention it signals, not a license to attribute unrecorded promises. A serious follow-up would ask which workloads Intel targets, what software path supports them and how the delivered result compares under a customer-defined test.

The photographs document a screen relay, not an audio transcript. Captions on the screen are not treated as verified verbatim quotations. The moderator is not named here because the cited agenda does not identify that person. [4]



Picture 7 | The audience watching Intel leadership. Author's conference photograph. [4]

Verified context: Intel is a Platinum sponsor. The agenda also includes Mingfei Ma's technical session on SGLang for Intel GPUs and Xeon CPUs. This links corporate visibility to software engineering work. [4]

Intel announced a RadixArk collaboration on November 12 at 3:30 p.m. Eastern, equivalent to 12:30 p.m. Pacific. It targets SGLang optimization on Intel platforms and contributions to the shared project. The announcement preceded the scheduled fireside chat. [4]

Intel is also a founding member of the SGLang Consortium announced that day. NVIDIA, AMD, Google, Meta, Cloudflare, Modal and RadixArk also joined. Members contribute expertise and roadmap feedback; maintainers retain technical authority. Membership does not give Intel control of SGLang. [4]

The audience includes people who select serving software and influence hardware purchases. A CEO can signal sustained engineering support and attract partners. The commercial goal is to move from “the model runs” to “the customer can operate it reliably and affordably.” No source establishes his private motives or a purchase commitment from attendees. [4]

Tan became Intel CEO effective March 18, 2025, after serving as Cadence CEO from 2009 to 2021. That background connects chip-design software with semiconductor execution, but it does not by itself establish success in AI serving. [4]

2.5 AMD: the developer is part of the route to deployment



Picture 8 | Vamsi Boppana — official summit portrait. Official summit material. [5]

AMD’s summit coverage combines sponsor participation, senior leadership context and developer outreach. Its visible developer-program material points to the practical work needed to turn available hardware into an application platform.

The Radeon workstation photograph and the Instinct data-center hardware belong to different deployment contexts. A Google TPU image elsewhere in the collected material is Google’s, and is not used as an AMD benchmark. Keeping those identities clear is essential to a useful comparison. [5]

AMD’s opportunity depends on useful model support, stable software and delivered performance, as well as chip specifications. The competitive question is what a customer can deploy and maintain at an attractive total cost. The official portrait used here documents the agenda’s speaker identity; it is not an author-taken photograph of a speech.

AMD is a Platinum sponsor. Its participation includes senior leadership, engineering talks, and a hands-on reinforcement-learning workshop. This is evidence of ecosystem investment; sponsorship alone does not establish product superiority. [5]

The commercial interpretation is that developers influence hardware purchases. If engineers can deploy a model reliably on AMD, their organizations can consider AMD for production. Training, cloud access and software support reduce the effort required to evaluate that option. [5]

No complete AMD session recording is available in this reporting record. Photographs and official documents support this account; statements about AMD’s commercial objectives are the author’s interpretation, not reconstructed quotations. [5]

Illustrative weight memory in decimal GB: parameters × bits per parameter ÷ 8. An 8-billion-parameter model needs about 16 GB at 16-bit precision or 4 GB at 4-bit precision. A 32-billion model needs 64 GB or 16 GB respectively; a 70-billion model needs 140 GB or 35 GB. [5]

These estimates exclude runtime memory, quantization metadata and KV cache. KV cache retains intermediate attention information from the conversation. Longer conversations and more concurrent users require additional space. Four-bit compression can change quality, depending on the model and method. [5]

Instinct MI355X is a data-center accelerator with CDNA 4 architecture, 288 GB HBM3E and 8 TB/s memory bandwidth per GPU. These figures describe a different product class from the R9700; nine times the memory does not imply nine times the response speed. [5]

FLOPS measures floating-point operations per second. Tokens per second measures language-model output rate. There is no universal conversion: model size, precision, memory traffic, batching and software all affect actual speed. Latency measures waiting time; throughput measures total work completed over time. [5]

2.6 Hugging Face: an optimization must reach the application

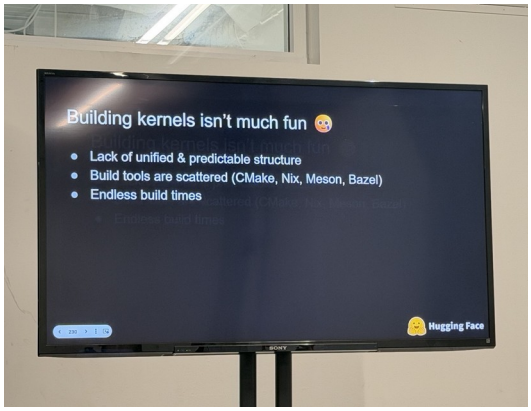
Adarsh Shirawalmath's kernels presentation asks how optimized code reaches an application reliably. A kernel is a low-level implementation of a particular operation. Its practical value depends on whether it can be built, loaded and used correctly with the customer's hardware and software versions.



Picture 9 | Adarsh Shirawalmath on kernels. Author's conference photograph. [8]

The Kernel Hub and related tooling address packaging, discovery and compatibility. An application may fall back to a slower implementation when an optimized dependency is missing or incompatible. The presentation's 7.5× comparison uses a particular fallback baseline; it does not show that every already-optimized installation becomes 7.5 times faster. [8]

Reducing deployment friction can be as commercially significant as inventing a faster operation. It broadens the population of developers who can obtain the intended performance. Reproducibility and version control remain essential, particularly when kernels participate in reinforcement-learning calculations where numerical differences can matter.



Picture 10 | Kernel portability and distribution. Author's conference photograph. [8]

Projects differ in directory structure and tools such as CMake, Nix, Meson, and Bazel. Compilation can take substantial time, and support matrices multiply maintenance work. Illustrative example: three PyTorch versions times two CUDA versions times two processor architectures create twelve combinations before other constraints. Someone must produce, validate, and maintain the combinations users actually need. [8]

The project offers tooling to build, distribute, and use compute kernels under a predictable structure. Reproducibility means recording dependencies and build conditions well enough to recreate intended artifacts consistently. It reduces reliance on an engineer's undocumented laptop configuration. standardized packaging can make specialist optimizations accessible to a much wider developer population. [8]

Kernel-builder serves the producer by preparing compiled artifacts. The kernels library serves the consumer by finding and loading an appropriate artifact. One resembles a factory; the other resembles delivery and installation. Keeping these responsibilities distinct lets users consume optimized code without reproducing the entire production process. [8]

A binary is compiled software. A component built for one PyTorch interface can fail under another version if low-level expectations change. The slide illustrates separate binaries for PyTorch 2.10, 2.11, and 2.12. This maintenance burden is about compatibility between components, not a change in the underlying mathematical definition of ReLU or attention. [8]

ABI means Application Binary Interface, the low-level agreement compiled components use to communicate. The deck proposes one binary targeting PyTorch's stable ABI across supported releases and states at least two years of ABI stability. Like keeping a socket specification unchanged, this can preserve compatibility. It requires use of the supported stable interfaces; it does not remove hardware, CUDA, or operating-system constraints. [8]

The example tries Hub FlashAttention first, then a locally installed implementation. If both fail, it marks the function unavailable. Later code must still choose another implementation or raise an error. Fallback improves resilience, but silent fallback can hide a major performance change; production systems benefit from logging which route actually ran. [8]

2.7 IBM Spyre: integration makes new hardware usable



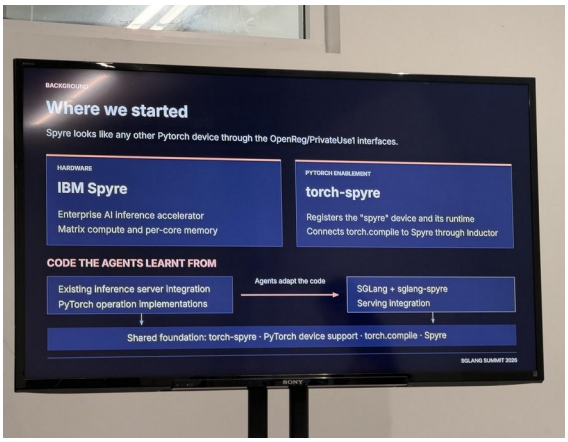
Picture 11 | Antoni Viros i Martin on IBM Spyre. Author's conference photograph. [9]

IBM's case addresses a barrier that raw chip specifications cannot solve: a model-serving framework must be connected to a hardware runtime in a way developers can actually use. Antoni Viros i Martin's session describes adapting existing Spyre support to SGLang, with agents assisting the engineering work.

The "under a week" result concerns this integration effort. The prior hardware, compiler and runtime were not created during that week. Reusing existing interfaces and implementation patterns is part of the explanation, not a detail to omit from the headline.

AI-assisted porting could shorten the path from hardware capability to framework support. Production use still requires tests, supported shapes and human review. A Hugging Face kernel benchmark is a separate result and is not attributed to IBM. [9, 8]

Spyre hardware and torch-spyre already existed. Matrix engines process numerical arrays; per-core memory keeps data close to computation. [9]



Picture 12 | IBM Spyre software integration. Author's conference photograph. [9]

Data must arrive before a calculation can run. Memory placement can matter as much as arithmetic capability. [9]

Torch-spyre registers a PyTorch device, manages runtime operations and connects torch.compile through Inductor. Runtime handles allocation and transfers; compilation translates operations into executable steps. OpenReg and PrivateUse1 are extension mechanisms, not new AI models. [9]

SGLang manages requests and reuse policy; sglang-spyre adapts interfaces; torch-spyre provides tensors, layouts, compilation and execution. [9]

Platform registration identifies the device. Attention backend implements attention calculations. KV pool manages stored keys and values. Operator dispatch sends each operation to its implementation. [9]

Two students can submit the same passage with different questions. Prefix caching reuses calculations for the identical opening, while different questions and answers still require work. This is calculation reuse, not copying a finished answer. [9]

Independent serving policy can remain unchanged while hardware execution changes. Logical tensor shape and physical memory layout are different concepts. [9]

The slide reports two developers, under one week and about 1,500 lines for the plugin. It lists attention, KV cache, model operations and prefix sharing. [9]

SDPA compares queries and keys and combines values. Existing Spyre attention and size handling were reused. KV allocation retains previous attention information; it still consumes real memory. [9]

Model operations: RMSNorm rescales values. SiLU and GeLU provide nonlinear transformations. Rotary embeddings encode position. Linear layers multiply by learned weights. These are different mathematical building blocks. [9]

Small integration code can depend on substantial existing infrastructure. Line count alone does not measure development productivity. [9]

Together, these examples show why an inference conference can bring competitors into the same room. A common software path can enlarge the set of usable machines and applications. The next question is what happens inside that path—and which improvement a buyer is actually being offered.

3 What Happens Between a Question and an Answer

The word inference can make the work sound like a single event. In a language-model service, it is a sequence of operations with different resource demands. The summit's recurring discussion of prefill, decode and their separation becomes easier to follow once those operations—and the clocks used to measure them—are made explicit.

3.1 What inference means

In machine learning, inference means applying a trained model to new inputs. A customer-retention model may return a probability. A language model may generate a sequence of words. A robot policy may produce an action representation. Calling all of these inference is reasonable, but their costs, validation methods and failure modes differ.

A language model's generated explanation can look like human reasoning. That appearance does not establish that each statement is true or that the reasoning process is reliable. For business use, the relevant question is whether the output satisfies a defined task and can be checked against data or outcomes. A fluent customer narrative can be less valuable than a well-calibrated probability with a clear decision rule.

Training and inference interact without becoming identical. Training changes model parameters; ordinary inference uses them. An application can alter behavior by adding context without updating those parameters, as Generalist's physical-prompting demonstration illustrates. Reinforcement learning can also use many inference attempts to generate feedback for later parameter updates. [10, 11]

The summit collection is therefore not simply about answering chat messages faster. It covers an expanding family of computational activities: generating candidates, processing large contexts, calling tools, checking proofs, creating media, interpreting feedback and coordinating hardware. Each activity can make different demands on the same infrastructure.

For the restaurant example, inference should be chosen after the decision is defined. Estimating repeat-purchase probability may call for a conventional predictive model. Extracting service issues from written feedback may benefit from a language model. Computing historical contribution requires accurate accounting and aggregation. Testing whether an offer caused additional visits requires an experimental design. One method does not replace all the others.

Table 4 | Four questions, four methods

Question	Suitable starting method	Validation
What happened?	Queries and descriptive analysis	Reconcile to source records
What is likely?	Predictive model	Future-period calibration
What does this text mean?	LLM extraction with a schema	Human-labeled sample
What did our action cause?	Randomized experiment	Treatment–control difference

3.2 Prefill: processing the input

Prefill processes the input sequence: instructions, the user's request, retrieved documents and any included history. In a typical transformer language model, the system computes representations for those input tokens and builds key/value state that later attention operations can reuse. That state is usually called the KV cache.

A long prompt can contain much more than the user's visible question. A research agent may include tool descriptions, search results, previous tool outputs and intermediate discussion. When the next step resends that accumulated material, input processing can become a large part of the workload. Reusable prefixes may reduce repeated work if the serving system can recognize and safely reuse compatible cached state.

Prefill often exposes large matrix operations that can use a GPU efficiently, especially with substantial input work. "Often" is important: prompt length, model architecture, batching and hardware affect the actual bottleneck. A short request is not guaranteed to be compute-bound, and a cache hit can change the work dramatically.

Time to first token includes more than prefill. It can include queuing, routing, input preparation and other overhead. A service may accelerate prefill but still leave the user waiting because requests queue behind a busy worker. A credible comparison therefore measures the whole first-response path as well as the engine operation.

In the customer-intelligence application, the most expensive prompt is not necessarily the most useful. Passing every transaction and every customer profile into one context is usually an inefficient way to ask a population-level question. Structured aggregation, a predictive scoring pipeline and retrieval of relevant supporting records can reduce unnecessary input while making the result easier to verify. [1, 12, 13]

3.3 Decode: generating the output

Decode generates additional tokens using the existing context and previously generated tokens. In ordinary autoregressive generation, later output depends on earlier output. This creates a sequential component: the system cannot simply calculate an arbitrary full answer in the same way it processes a fixed input sequence.

At small batches, repeated weight access and other memory operations can limit decode efficiency. With larger batches or different model structures, compute and communication can become more important. “Decode is memory-bound” is a useful starting intuition in some settings, not a universal law. Horace He’s presentation asks us to identify the actual operating point. [1]

The inter-token interval describes the gap between successive output tokens. A short interval makes a visible response feel smooth. Aggregate output tokens per second counts all requests together. These are different metrics. A service can stream one answer rapidly while serving relatively few users, or produce high total throughput while each user experiences slower output.

The KV cache grows with sequence state. Longer contexts, more simultaneous sequences and architecture-specific state requirements can consume memory even when the model's parameters are unchanged. Memory available for active requests can therefore constrain the batch size and the number of users that fit on a worker.

Speculative decoding changes how output candidates are proposed and verified. Accepted candidates can advance generation by several tokens in a step, but the gain depends on proposal quality, verification cost and the workload. A milliseconds-per-step figure cannot automatically be inverted to obtain user tokens per second without knowing how many tokens are accepted. The Baseten presentation makes this distinction particularly clear. [14]

Table 5 | The speed vocabulary

Metric	Meaning	Frequent mistake
TTFT	Time until first output token	Calling it pure prefill time
ITL	Gap between output tokens	Treating its inverse as total service output
Throughput	Completed work per unit time	Ignoring quality and delay
Goodput	Useful output meeting stated targets	Omitting the target definition
End-to-end time	Whole task completion time	Ignoring tools, retries and verification

3.4 Horace He: ask what the percentage measures

Horace He’s explanation is central to this report because it separates individual streaming speed from aggregate throughput. The prefill/decode example shows how eliminating interruptions can accelerate one request’s decode stream without increasing the system’s total output.

When a presentation says “75% prefill, 25% decode,” the first question is: 75% of what? It might mean engine time, total latency, hardware allocation, cost or token counts. These denominators are not interchangeable. Horace He's worked scheduling example concerns time spent by a shared engine on the two kinds of work. [1]

Suppose four identical mixed workers each spend three quarters of their time on prefill and one quarter on decode. Across the group, this is equivalent to three worker-units of prefill activity and one worker-unit of decode activity. A proposed split assigns three workers to prefill and one to decode. The total allocation is unchanged in this idealized accounting.



Picture 13 | Horace He explaining inference scheduling. Author’s conference photograph. [1]

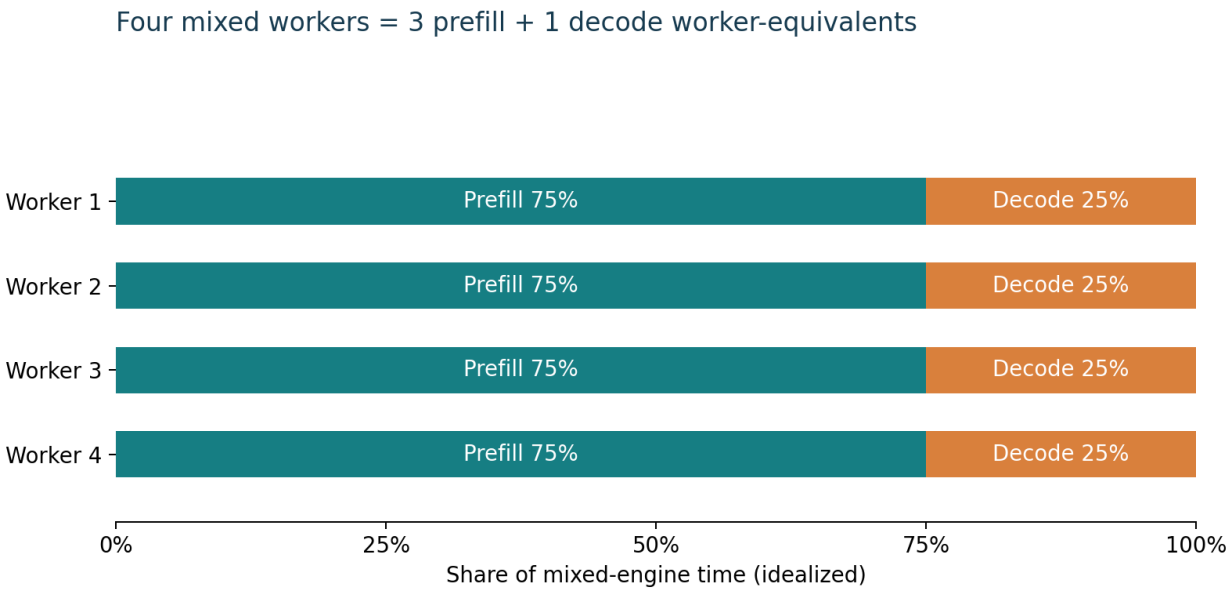
A request on a mixed worker receives decoding opportunities only part of the time. A dedicated decode worker can give its active sequences more regular opportunities. If it preserves the same decode-batch operating point, removing prefill interruptions can make a given sequence stream four times faster. That does not create four times as much total decode capacity. The original four workers already supplied the equivalent of one full-time decode worker.

The thought experiment is deliberately simplified. It assumes balanced arrivals, compatible batch sizes, sufficient memory and no additional cost for moving state between workers. Real deployment must test each assumption. It is especially dangerous to read the percentage as “75% of the tokens are input,” because one input token and one output token need not consume equal engine time.

This is the central distinction. PD disaggregation can improve the experience of a request by changing when work occurs, even before it changes the amount of work the whole service can perform. Whether it also improves cost or capacity is a separate measurement.

Table 6 | Idealized allocation in the 75/25 example

Design	Prefill worker-equivalents	Decode worker-equivalents
4 mixed workers	4 × 0.75 = 3	4 × 0.25 = 1
3 prefill + 1 decode	3	1
Change in aggregate allocation	None	None



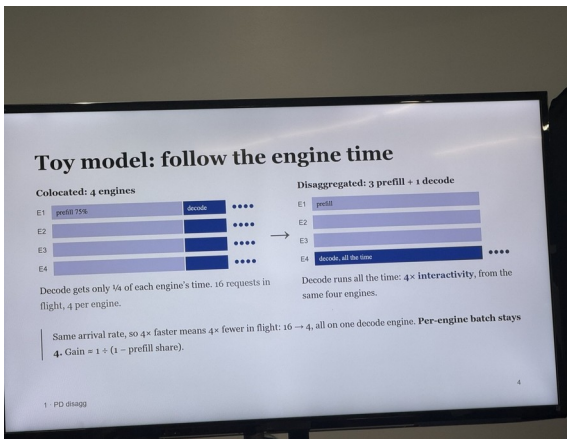
Picture 14 | Prefill and decode allocation across four engines. GII explanatory diagram; assumptions are stated in the text.

Each horizontal bar represents one mixed worker over the same interval. Three quarters of its time is allocated to prefill and one quarter to decode. Summing across four workers produces three prefill worker-equivalents and one decode worker-equivalent.

Separating the phases changes the schedule. It does not make the prefill work disappear. The dedicated decoder can stream more continuously, while three other workers continue processing inputs. The following section works through the arrival rate, active-request count and unchanged aggregate output.

This picture should not be read as a universal prescription for three prefill GPUs and one decode GPU. A “worker” or “engine” can itself use multiple devices. Real allocations depend on the model, memory, network and workload. The example isolates one scheduling mechanism so that its consequences can be understood.

3.5 Four times faster can mean unchanged throughput



Picture 15 | Two models further along the engine timeline. Author's conference photograph. [1]

Consider the illustrative operating point used to explain Horace He's scheduling example: 1.6 arriving requests per second, each generating 100 output tokens. Required output is therefore 160 tokens per second. Before separation, each request streams at 10 tokens per second and spends approximately 10 seconds decoding. This is explanatory arithmetic, not a production benchmark. [1]

At steady state, arrival rate multiplied by mean time in the stage gives the average number of active requests in that stage. Here that is 1.6 × 10 = 16 active decoding requests across the service. Spread across four mixed workers, the example has four active sequences per worker. This is an application of flow accounting, not a universal batch-size rule.

After the idealized split, one dedicated decoder streams each of four active sequences at 40 tokens per second. Each 100-token output now takes 2.5 seconds to decode. At the same arrival rate, 1.6 × 2.5 = 4 active decoding requests. The dedicated worker can retain a batch of four while producing the same 160 output tokens per second.

This resolves an apparent contradiction. Per-request streaming is four times faster, but total output remains unchanged. The faster service holds fewer requests in the decoding stage at once. The result is not obtained by assuming the dedicated decoder must preserve all 16 simultaneous sequences.

The example excludes queueing, first-token work, transfers, warm-up and variability. It teaches the accounting needed to ask better questions of a benchmark. A real service might use the improvement to reduce waiting, accommodate a tougher service target or change hardware allocation. None of those consequences should be claimed until the measured system demonstrates them.

Table 7 | Same output, different user experience

Quantity	Mixed workers	Dedicated decode
Arrival rate	1.6 requests/s	1.6 requests/s
Output per request	100 tokens	100 tokens
Streaming speed	10 tokens/s	40 tokens/s
Decode duration	10 s	2.5 s
Active decode requests	16 across 4 workers	4 on 1 worker
Aggregate output	160 tokens/s	160 tokens/s

3.6 A conditional formula for the scheduling benefit

Let p be the share of a mixed engine's time spent on prefill. If all other conditions stay unchanged, removing prefill interruptions gives the decode stream an idealized scheduling multiplier of 1 / (1 − p). The 75% example gives 4.0. At 50%, the multiplier is 2.0. At 10%, it is only about 1.11. [1]

This formula is a way to reason about one mechanism. It is not a performance prediction for an arbitrary model. Changing the worker layout can change batching, memory fit, communication and queueing. A service that already devotes most engine time to decode has little interruption time to recover through this mechanism alone.

A system with very fast prefill and demanding decode may therefore prefer a mixed design, or may use separation for another reason: different hardware, operational isolation, or independent scaling. Conversely, long contexts and repeated input processing can make prefill pressure substantial. The actual workload distribution matters more than a universal architectural slogan.

Chunked prefill offers another approach. Instead of allowing a long input-processing job to monopolize an engine, the scheduler can divide it into smaller chunks and interleave them with decode work. This may reduce interruptions, but the chunk size and scheduling policy create their own tradeoffs. A 512-token scheduling budget shown in a talk is a configuration in an example, not a value every system should adopt.

For an operator, the correct procurement question is: under my prompt lengths, output lengths, burst pattern and latency targets, which layout provides the best useful service per dollar? A vendor should supply distributions and target-compliance rates, not only the most attractive average.

Table 8 | Sensitivity to the original prefill time share

Prefill time share p	Decode share 1 − p	Ideal scheduling multiplier
10%	90%	1.11×
25%	75%	1.33×
50%	50%	2.00×
75%	25%	4.00×

3.7 What PD separation must pay for

Separating prefill and decode creates a handoff. The decoding worker needs the state produced during input processing. Moving a KV cache consumes bandwidth, takes time and requires coordination. A design that looks attractive in a compute-only diagram may lose its advantage when transfer, scheduling and queueing are included.

The relevant transfer quantity is not just the original text prompt. It is the model's derived state, whose size depends on architecture, sequence length, numerical format and parallelism. Moving that state across machines can be a different engineering problem from keeping it within a tightly connected system.

Hang Yin’s Phala presentation adds a further complication. The protected environment changes available communication paths and can limit direct transfer mechanisms. Fast matrix multiplication in that environment does not guarantee equally fast end-to-end serving. Security requirements belong in the architecture decision from the start. [15]

Baseten's stack combines PD separation with cache-aware routing and peer-to-peer state sharing. These are complementary pieces: routing determines where requests go, reuse avoids repeated work, transfer makes state available, and scheduling determines which phase runs. A production result belongs to that combined system, not automatically to one isolated feature. [14]

A useful experiment compares mixed and separated configurations with the same model version, precision, total accelerator count, input/output distributions and service target. It also reports cache policy, network topology and warm-up. If the separated configuration uses more hardware, the comparison should disclose the extra resources. If quality changes, that belongs beside the performance result.

Table 9 | PD deployment decision record

Potential benefit	Associated cost or condition
Independent phase scaling	Risk of imbalance and idle workers
Smoother decode intervals	Cache-transfer and handoff delay
Hardware specialization	More complex placement and operations
Workload isolation	Separate queues and capacity planning
Reuse across requests	Correct cache identity and lifecycle control

3.8 Four ways an inference workload can grow

A model can grow in parameter count. A request can grow in context length. A service can grow in concurrent users. An agent can grow in the number of attempts or steps it performs. These changes are often discussed together, but they place different demands on infrastructure.

More parameters generally mean more weight storage and more computation, although the relationship depends on architecture. A mixture-of-experts model can contain many total parameters while activating only a subset for a token. That does not make the remaining parameters irrelevant: they must still be stored, selected and made available when needed. WiCi's SSD work addresses this capacity-and-movement problem. [16]

Longer context increases the amount of information a request carries. Depending on the model and attention implementation, it can increase prefill work, state storage and subsequent attention cost. More concurrent sequences multiply the amount of live state. A model that fits in memory for one user may not support the intended production concurrency.

More agent steps create repeated inference and tool activity. A relatively small model used many times may consume considerable total resources. Verification can add cost while improving useful success. The denominator should therefore be a completed, validated task or a defined operating result, not merely a model invocation.

The growth question becomes an architectural question: which resource runs out first? The answer may be memory capacity, memory bandwidth, compute, interconnect, software scheduling or an external tool. Buying more arithmetic capacity will not solve every one of these constraints. The memory-tiering, orchestration and agent talks are valuable because they identify different limits.

Table 10 | Four growth axes

Growth axis	Likely pressure	First measurement
Parameters	Weights, active computation	Memory fit and per-step cost
Context length	Input processing and KV state	TTFT and state bytes
Concurrency	Live state and scheduling	Tail latency under load
Agent steps	Repeated calls and tools	Validated task time and cost

3.9 Baseten: measure the stack of changes

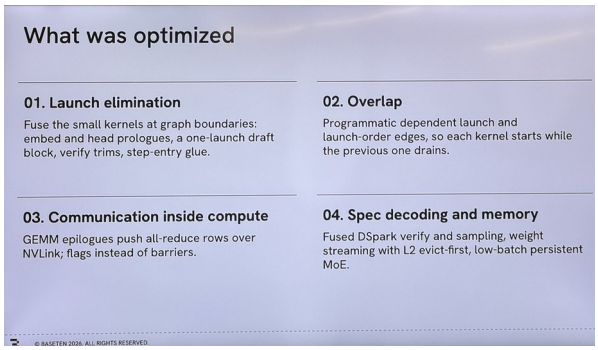
Tri Dao and Alexander Korte's Baseten session presented an optimized serving path and the engineering process behind it. Dao is the Baseten engineer identified in the session, distinct from the FlashAttention researcher of the same name. Profiling, attribution and validation gates connect performance work to correctness, model quality and behavior under load. [14]



Picture 16 | Tri Dao and Alexander Korte presenting Baseten. Author's conference photograph. [14]

The presentation uses several clocks: a synthetic layer result in microseconds, decode-step timing, user token speed and a live application's first-token and completion times. They measure different scopes. The 43-microsecond layer result is not whole-answer latency, and a speculative step can accept more than one token. [14]

The most valuable lesson is the disciplined optimization process. A faster result needs a precisely defined baseline and a test that protects useful behavior. The stack's cache-aware routing, state sharing and PD separation also show why a production result cannot automatically be assigned to a single feature.



Picture 17 | The Baseten optimization stack. Author's conference photograph. [14]

The slide groups work into launch elimination, overlap, communication inside compute, and speculative decoding plus memory. A kernel is a GPU function executed by many threads. [14]

Tiny operations can spend a meaningful share of time starting, waiting, moving data, or coordinating. Making the arithmetic faster alone may leave those delays untouched. [14]

Fusion combines small operations at graph boundaries. PDL overlaps eligible work. GEMM epilogues—the final steps of matrix multiplication—push all-reduce data over NVLink, using flags where appropriate. DSpark verification and sampling are fused; L2 evict-first hints help manage streamed weights; persistent MoE work targets low batches. [14]

These techniques reduce different overheads and must work together. A flag still requires correct ordering; a cache hint changes retention policy, not model knowledge or numerical precision. [14]

The workflow is profile, attribute, and gate. Nsight Systems and the PyTorch profiler show the step timeline; Nsight Compute studies individual kernels; CUDA events time A/B comparisons. [14]

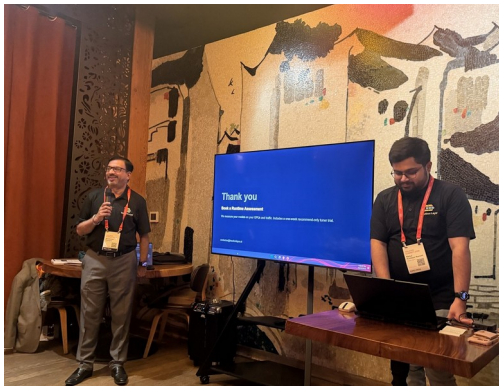
A blank gap in a GPU trace does not explain itself. The team must determine whether the cause is launch overhead, a data dependency, communication, or underused execution resources. [14]

The slide says to assign each gap or underused SM to a kernel or group and attack the biggest first. An SM is a streaming multiprocessor, one of the GPU's compute units. Bit-exact changes require bitwise checks; numerical changes must pass log-probability KL checks and reasoning-mode GSM8K. [14]

Measure before optimizing and compare under matched conditions. Warm-up, context length, output length, concurrency, and model settings can otherwise move the result independently of the code change. [14]

Three safeguards are shown: bitwise comparison, model-quality checks, and hang or stress testing. Large code-diff counts on the right illustrate review volume, not verified productivity. [14]

3.10 Traction Layer: predictable service is a product



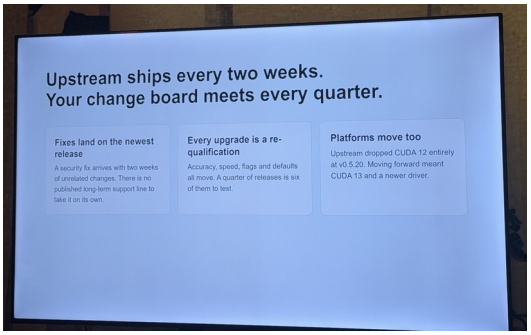
Picture 18 | Presenters at the Ghirardelli Square companion event. Author's conference photograph. [17]

At the Dim Sum and Demo companion gathering at Ghirardelli Square, Shankaran Sitarama and Rahul Kundapur Ramesh presented Traction Layer's supported SGLang path, hardened builds and operating controls. The setting was separate from the summit stage. The product proposition was more predictable production behavior. [17]

The commercial problem is familiar to enterprise operators: rapidly changing upstream software can be difficult to adopt continuously. Maintaining a known branch, backporting fixes and controlling upgrades may reduce operational disruption, but support scope and compatibility must be explicit.

The performance discussion also involves tradeoffs. Smoother decode streaming can require delaying other work or changing throughput. A controller that improves one tail-latency measure should be judged alongside first-token latency, total useful output and failure behavior. stability is a service attribute customers may pay for, provided it is defined and demonstrated rather than implied by the word "hardened."

The slide contrasts frequent upstream releases with slower enterprise change approvals. A change board is a group that evaluates operational risk before production changes. Qualification means checking that the actual application still works: correct answers, acceptable latency, secure access and successful rollback. It is more than confirming that the software starts. [17]



Picture 19 | Upstream releases and enterprise approval. Author's conference photograph. [17]

The transcript mentions CUDA changes. CUDA is NVIDIA's GPU computing platform; moving between major software environments can require compatible libraries and testing. This is a speaker's example, not proof that every minor SGLang update forces a CUDA upgrade. The maintenance product is aimed at reducing repeated qualification work. [17]

A backport takes a fix developed for newer code and adapts it to an older supported branch. It is not necessarily copying a file: surrounding code can differ, so engineers must adapt and test it. The slide includes security and correctness fixes while excluding new features, changed defaults and other behavioral changes. [17]

A repaired lock is a useful analogy: you can replace the faulty lock in an existing classroom without rebuilding the school. But installing an entirely new classroom layout is a bigger change. Ask which fixes are included, how quickly they arrive, how long the branch lives and what happens when a fix cannot be adapted safely. [17]

This transition introduces the tuning demonstration, attributed on the slide to Rahul Kundapur Ramesh. The problem is mixed traffic: short chats, long prompts and long answers share a serving engine. A system can generate many tokens overall while a particular person sees a frozen answer. [17]

The engineering lesson is to keep the clocks separate and then reconnect them. Prefill, decode, queues and transfers each explain part of the delay. The useful architecture is the one that meets the complete workload's requirements with measured quality, reliability and cost.

4 Why the Whole System Matters

The previous chapter explains when work runs. This chapter follows where it lives and how it moves: from model weights and cached attention state to memory tiers, accelerator trays, switches and cooling connections. The photographs are useful because they make those dependencies visible. Software efficiency and physical infrastructure meet inside a working configuration.

4.1 Memory capacity and movement shape the service

Memory capacity answers how much information can be held. Bandwidth answers how quickly it can move. Latency answers how long an individual access takes. Locality describes whether the needed data is already near the processor. All four matter, and a large number for one does not substitute for the others.

Netpreme's presentation discusses a hierarchy that uses external memory and selective access. Its reported 24 TB capacity, a prototype bandwidth figure and an architecture-level aggregate figure describe different scopes. They should not be collapsed into a statement that external memory behaves exactly like on-package HBM. The software must identify and move the useful subset at the right time. [18]

WiCi applies a related principle to large mixture-of-experts models on consumer hardware. SSD storage expands what can be held, but repeated uncached reads can limit output speed. Packing data for access patterns, avoiding unnecessary transfers and managing reuse are central to the result. A large advertised parameter count alone says little about interactivity. [16]

Xibo Sun’s Tencent TACO presentation approaches state management through interfaces, compatibility and shared ownership. Greater KV capacity is not itself a throughput result. The service also needs correct routing, synchronization, lifecycle handling and fallback paths. [19]

Useful inference capacity is created partly by deciding what does not need to move or be recomputed. This is why memory software can carry commercial importance even though it is less visible than the accelerator. It can expand the feasible workload, improve utilization or reduce unnecessary hardware duplication. The economic benefit remains conditional on engineering cost and measured production performance.

4.2 Tencent TACO: managing state across the serving system

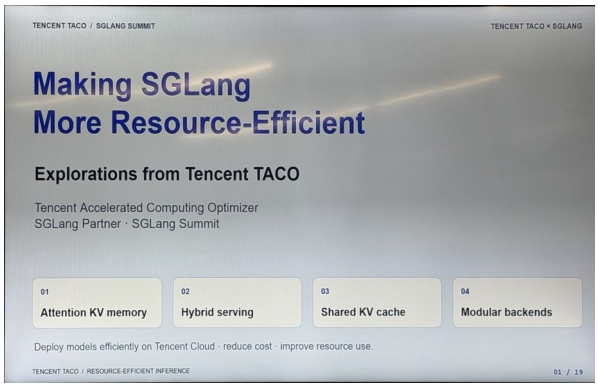
Xibo Sun's Tencent presentation brings attention to the software contracts around distributed inference. Different components need agreed representations and rules for looking up, sharing, moving and retiring state. Physical layouts can differ while interfaces define what is compatible.



Picture 20 | Xibo Sun presenting Tencent TACO. Author's conference photograph. [19]

TACO's cache and parallelism work includes shared-memory and state-management ideas. The operational test is whether the state can be reused and moved correctly under a real serving workload, rather than simply whether a larger quantity can be stored. [19]

Standardizing boundaries can reduce integration cost and make alternative backends easier to use. It can also reveal unsupported cases that must be rejected or sent through a fallback path. A capacity increase such as the presentation's CP8 example should not be described as a throughput increase without a measured workload result.



Picture 21 | Tencent TACO resource management. Author's conference photograph. [19]

Tencent TACO presents four ways to improve the machinery that runs an AI model: reduce duplicated memory, use serving capacity more flexibly, share cached work, and simplify backend integration. This is a talk about inference—the process of using a trained model to produce answers. [19]

Think of a school library. Buying more desks does not solve every problem if books are duplicated unnecessarily, some desks sit empty, and students repeatedly carry the same books between rooms. The talk examines comparable waste inside AI serving systems. [19]

The four workstreams operate at different levels: attention memory inside a distributed engine; routing between prefill and decode workers; cache management across machines; and interfaces between software modules. Later slides connect each level to a specific mechanism. [19]

Resource efficiency means doing more useful work with available resources. The slides contain both implemented experiments and future directions; the report keeps those categories separate. [19]

The roadmap organizes the presentation into FSCP, P+PD, FlexKV with RadixShmem, and framework–backend decoupling. FSCP addresses duplicated KV storage; P+PD addresses uneven serving demand; FlexKV addresses cache reuse; backend interfaces address integration dependencies. [19]

A single utilization percentage hides different problems. A GPU may have unused computing capacity but insufficient memory. A decode server may be busy generating answers while its CPU memory remains available. Each resource needs its own policy. [19]

Read the four sections as complementary proposals, not four multipliers that can be multiplied together. Their benefits may overlap, and a change that saves memory can add communication or scheduling work. [19]

First identify the constrained resource, then evaluate the matching technique. Memory capacity, transfer cost and scheduling flexibility need separate measurements. [19]

4.3 Netpreme: extending the memory hierarchy



Picture 22 | Nikita Lazarev on memory tiering. Author's conference photograph. [18]

Nikita Lazarev's memory-tiering case starts with a constraint: fast on-package memory is valuable and difficult to expand indefinitely. External memory can provide more capacity, but the system must move the right information fast enough to preserve useful performance.

The presentation combines a memory hierarchy, selective access and software mechanisms for data movement. The reported 24 TB capacity, per-GPU prototype bandwidth and broader architecture bandwidth are different measurements. They should remain attached to their configurations and scopes. A larger memory pool is not equivalent to the same quantity of local HBM.

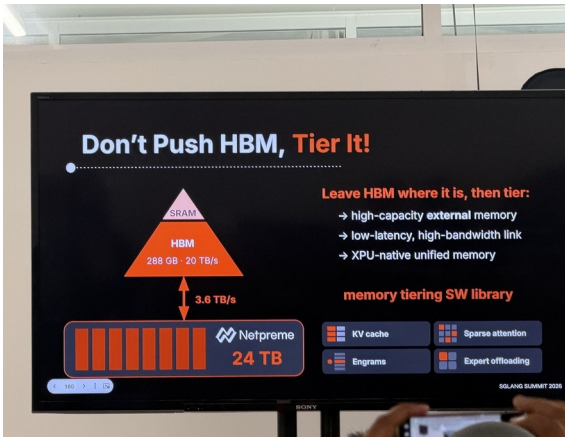
The opportunity is to increase useful state capacity without paying for unnecessary compute duplication. The risk is that access patterns and transfers erase the benefit. Commercial evaluation therefore requires an end-to-end workload, including cache behavior, network contention and integration effort. The strongest comparison is useful service at a fixed quality and latency target.

The pyramid places tiny SRAM above HBM labeled 288 GB and 20 TB/s, with a 24 TB external pool connected by a 3.6 TB/s link. These are architecture labels. [18]

Not every byte needs the fastest memory at every moment. External capacity can hold a much larger history while fast memory holds the active working set. [18]

A software library manages KV caches, sparse attention, Engrams and expert offloading. Unified memory describes a programming and management goal; it does not mean every location has identical access time. [18]

24 TB is about 83 times 288 GB using decimal units, but this is a capacity ratio only. Do not treat the slide as a shipping B300 bandwidth specification. [18]



Picture 23 | Memory tiering beyond HBM. Author's conference photograph. [18]

Dense attention reads and stores the full cache in HBM. Ordinary sparse attention reads selected records but may still store everything there. HiSparse adds tiering so HBM holds a hot set. [18]

Reading fewer records does not automatically free capacity. Full history must remain available because later queries can select different records. [18]

The two way arrow connects HBM with CPU DRAM through PCIe or chip to chip links. Cache misses bring needed records back. A KV cache stores intermediate model representations rather than literal text pages. [18]

The 4.7 times peak decode claim belongs to the cited HiSparse work across its tested settings. It is not the same result as Netpreme's later 2.4 times comparison. [18]

Prefill GPUs process the initial prompt and publish the full KV cache. Decode GPUs generate new tokens using a hot KV buffer. X Mem holds a shared 24 TB cache pool. [18]

Separating a large history from the active buffer could support more requests without requiring every decode GPU to hold every cached record locally. [18]

4.4 WiCi InferSSD: capacity with a bandwidth tradeoff



Picture 24 | Zili Meng presenting WiCi InferSSD. Author's conference photograph. [16]

Zili Meng's WiCi session explores running very large mixture-of-experts models with SSD-backed capacity on consumer hardware. The system must make selected expert weights available efficiently rather than load every parameter into scarce fast memory at once.

The InferSSD presentation explains data layout, explicit reads, memory budgets and different prefill/decode policies. The photographed 7.27-token-per-second result belongs to the displayed configuration. It should not be combined with specifications from other product descriptions to create a more favorable composite result. [16]

The approach can expand what is technically possible on a local machine, but buyers should evaluate time to first token, sustained output, storage behavior and complete ownership cost. "No additional quantization" does not automatically mean full FP16 weights. The model format and baseline must be stated.

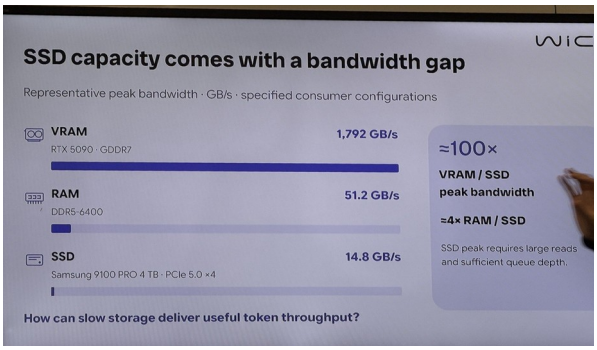
The displayed peaks are 1,792 GB/s for VRAM, 51.2 GB/s for RAM and 14.8 GB/s for SSD. Their ratios are about 121 and 3.46, respectively, relative to SSD. [16]

A fast GPU can sit idle while waiting for weights. Peak device bandwidth also differs from useful end-to-end bandwidth. [16]

Illustration: reading 10 GB at 14.8 GB/s takes at least about 0.68 seconds before other costs. Larger contiguous requests, sufficient outstanding reads and reuse help; RAM bandwidth depends on channel configuration. [16]

The stack runs from inference engine and operators through expert runtime, filesystem, NVMe driver and SSD controller. [16]

Each layer can add waiting or extra copying. Improving one stage leaves the slowest remaining stage in control of throughput. [16]



Picture 25 | SSD capacity and its bandwidth gap. Author's conference photograph. [16]

The runtime maps selected experts to locations, reads missing data and places work on CPU or GPU. Filesystem and driver co-design is a broader system effort than the open runtime alone. [16]

The university logos on the slide indicate a research context, not evidence of corporate ownership or financial investment. [16]

During ordinary inference, weights remain fixed even though activations and KV state change. Read-only applies to weights, not the entire program. [16]

Stable weights can be laid out in advance, cached and distributed across drives. Flash reads generally consume far less write endurance than program/erase cycles. [16]

Four drives can hold different model portions and read concurrently if lanes and controllers support it. Four drives sharing one narrow link may remain bottlenecked there. [16]

Read/write consistency is not the only reason HBM or DDR capacity is difficult to scale. Packaging, power, cost and interconnect also matter. [16]

Direct GGUF indexing locates existing weight ranges. Optional packing places one expert's gate, up and down matrices together. [16]

4.5 NVIDIA: reading the rack from infrastructure to silicon

The rack's rear view changes the story. The front suggests rows of computing equipment; the back reveals the connections and service access that make those rows usable. This is why the infrastructure chapter belongs in the sequel to a restaurant operating-platform report: deployment depends on the surrounding system, not simply on acquiring a capable chip. [7, 2]

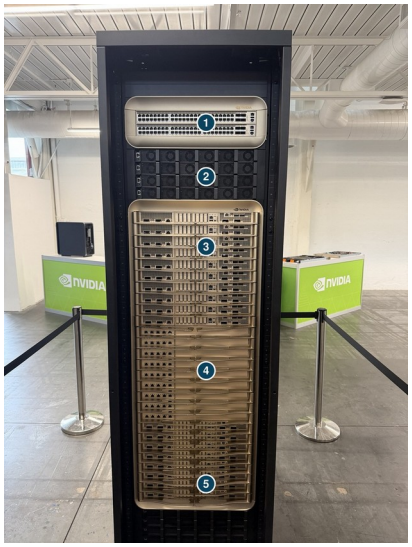
Read the rear markers. 1 identifies the large upper pipe fittings. 2 identifies the dense interconnect and service region. 3 identifies cable-routing structures. The horizontal black runs are not all water pipes. Their appearance does not establish hidden electrical ratings, valve arrangements or the exact rack generation. [7]

The practical question is access. A technician needs room to isolate power and coolant, reach connections, and remove a tray without damaging adjacent cables. A rack that fits into a floor plan may still be difficult to maintain. These requirements connect AI equipment to usable building capacity, maintenance time and the economics of reliable service. [7]



Picture 26 | The other side of the NVIDIA rack. Author's photograph; numbered annotations. [7]

Read the front as a map of jobs



Picture 27 | Front compartments perform different jobs. Author's photograph; numbered annotations. [7]

Marker 1 — upper network equipment. The visible chassis resembles a network device; its precise management role must be checked against the bill of materials. A network box is not another GPU tray.

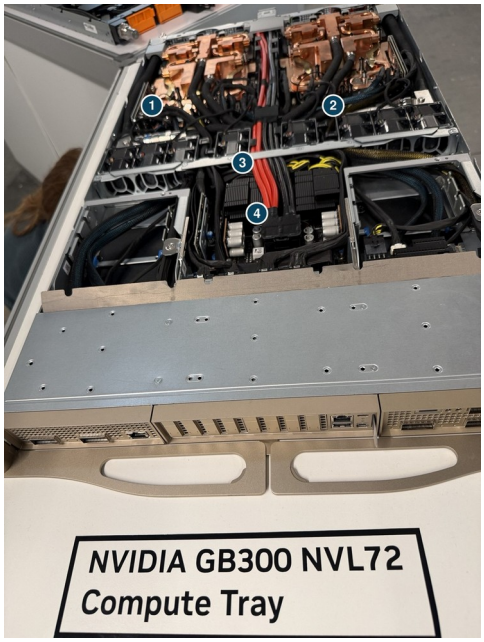
Marker 2 — power-shelf-style region. The repeated small modules are consistent with power infrastructure. Counting each module as a GPU would confuse electrical supply with computation.

Markers 3 and 5 — compute-tray-style regions. These removable units provide the server structure for CPU and GPU work. The display photograph alone does not verify installed counts or whether the rack was powered.

Marker 4 — switch-tray-style region. Its job is to move data among accelerators. Communication capacity is a distinct component of the system, even when the tray resembles a compute server. The exact product generation remains unconfirmed from this rack photograph. [7]

My interpretation is that a buyer should specify the workload and service level before comparing racks. Hardware, networking, cooling and support need to be priced as a working configuration. Neither the display nor the component count supplies a complete purchase quotation. [7]

Inside GB300 — computation, electricity and heat

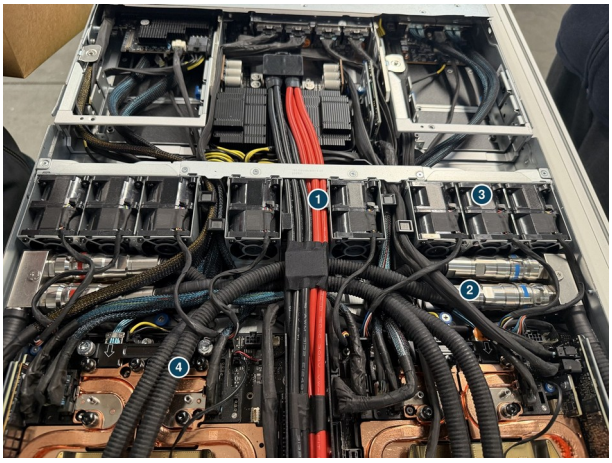


Picture 28 | GB300 compute tray with its identifying sign. Author's photograph; numbered annotations. [7]

The sign is readable: GB300 NVL72 Compute Tray. It provides firmer identification than proximity to a sponsor logo. NVIDIA's reference architecture describes the standard tray as two Grace CPUs and four Blackwell Ultra GPUs. Those counts come from the technical documentation; the photograph shows cooling assemblies covering the silicon. [7]

Markers 1 and 2 identify the two compute-board regions. Each standard board combines one Grace CPU with two GPUs. The CPU coordinates host work; the GPUs carry out parallel model computation. A single visible cooling assembly should not be treated as a direct photograph of every chip underneath it. [7]

Marker 3 identifies the fan row; marker 4 identifies power distribution and cabling. Liquid cooling and fans can coexist because components within the same chassis have different cooling needs. The tray is a server building block, not the entire NVL72 rack. It depends on compatible switching, power distribution and liquid connections. [7]



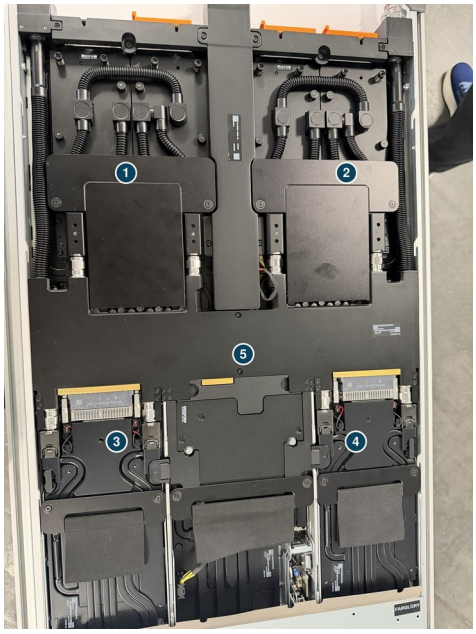
Picture 29 | Power wires, liquid fittings and fans in GB300. Author's photograph; numbered annotations. [7]

Follow the connections, not the colors. In the second photograph, 1 marks thick red/black electrical conductors; 2 marks metal liquid couplings; 3 marks fans; and 4 marks a covered hose or cooling route. Braided signal bundles are also visible. Red and blue rings on a liquid fitting do not turn it into an electrical connection. [7]

These systems fail differently. A data connection can lose bandwidth or introduce errors; a restricted cooling path can cause overheating; an electrical fault can interrupt service. The corresponding installation checks concern network, hydraulic and electrical systems before combined operation. This is the physical counterpart to checking model quality, latency and reliability separately in a serving benchmark. [7]

For an operator, the relevant outcome is sustained useful work. The cost of the rack must include the conditions that keep it operating: installation, cooling capacity, replacement procedures and support. A peak arithmetic figure cannot answer those questions. [7]

Rubin — a compute tray and a switch tray are different purchases

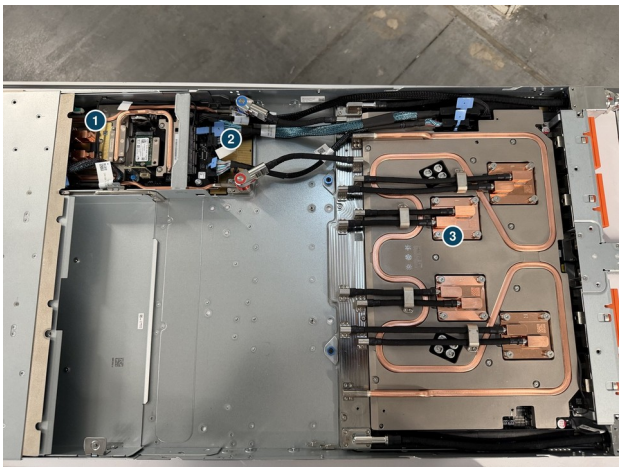


Picture 30 | Rubin compute tray and its covered processing zones. Author's photograph; numbered annotations. [7]

Readable exhibition signs and wider views distinguish the Rubin compute tray from the Rubin switch tray. The standard compute tray contains two Vera CPUs and four Rubin GPUs. Black covers and cold plates conceal package boundaries, so the numbered zones are not four individually exposed GPU packages. [7]

Read markers 1–4 as locations of major covered assemblies. Marker 5 identifies the central structural and service region. The broadly symmetric layout helps explain modular construction, but a handle alone does not prove that a module can be replaced while the system is powered. [7]

The commercial question is what replacement interrupts: one component, a tray, or a wider job. Service procedures and supported replacement units matter because an inference service can be affected by the work needed to maintain its hardware. [7]



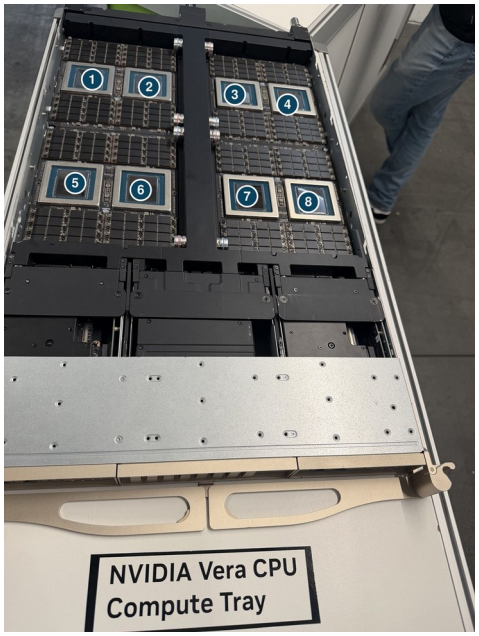
Picture 31 | Rubin switch tray with management and cooling regions. Author's photograph; numbered annotations. [7]

The switch photograph tells a different story. Marker 1 identifies the management-board area and small storage-module region. Marker 2 identifies red/blue-ringed liquid connections. Marker 3 identifies the cooling assembly associated with four NVSwitch ASICs in the reference design. These specialized chips move data rather than performing the role of the rack's AI GPUs. [7]

A switch still needs firmware, boot support, monitoring and local control. Storage and a battery inside it do not make it a compute tray. The wider view places the M.2 module and liquid connections in the Rubin switch tray, resolving the ambiguity of an isolated close-up. [7]

Interconnect is part of delivered model performance. Buying accelerators without the appropriate communication fabric does not recreate the behavior of an integrated rack. Compare configurations at system level, with the model, precision, concurrency and service objectives held explicit. [7]

Vera — eight CPU packages are not eight GPUs



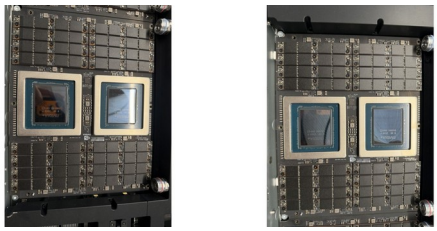
Picture 32 | Vera CPU tray with eight numbered processor positions. Author's photograph; numbered annotations. [7]

The display label reads Vera CPU Compute Tray. Eight exposed large processor packages are visible. The numbered markers locate those packages; they do not identify eight GPU accelerators or eight individual CPU cores. The GPU count in the neighboring Rubin compute tray is a separate question. [7]

A package is a physical processor component; a core is an execution resource inside it. Eight CPUs multiplied by the documented 88-core configuration would give 704 CPU cores. The arithmetic applies if that configuration is installed; it does not establish the exhibit's clock settings, memory population or operational state. [7]

This distinction matters when interpreting hardware marketing. CPU cores, GPU devices, memory capacity and floating-point operations measure different things. None can be substituted directly for the number of simultaneous customers a model-serving system can support. [7]

The close-up shows two processor packages and neighboring memory-module arrays. They are part of the same eight-CPU system, not two more CPUs to add to the inventory. Installed memory capacity depends on configured parts and population, not on counting dark rectangles in the photograph. [7]



Picture 33 | Vera CPU packages and surrounding memory arrays. Author's photograph; numbered annotations. [7]

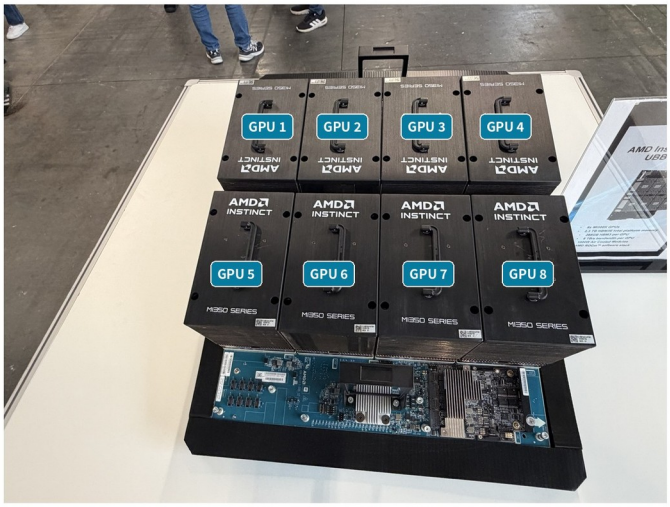
For the inference research agenda, the useful chain runs from workload to configuration: what must the model do, what state must remain in memory, how quickly must data move, and what does the full system consume while delivering the required service? The rack photographs give that chain a physical setting. They do not replace an acceptance test. [7]

Questions for the next field interview. Which exact configuration is being offered? Which components are included in the price? What quality and response-time targets were measured, and under what concurrent load? What happens during a component failure or planned maintenance? [7]

4.6 AMD: from accelerator board to complete server

The AMD photographs distinguish an eight-accelerator platform from an opened rack-style server. They should not be combined into one invented configuration. The exhibit sign identifies the accelerator board as MI350X; the precise CPU and GPU population inside the separate silver chassis remains unresolved. [6]

Eight MI350X GPU modules
八組 MI350X GPU 模組



8 GPUs
八組 GPU 模組

2 rows × 4 modules
兩排 每排四組

Black blocks are heat sinks
黑色大方塊是散熱器

GPU packages are underneath
GPU 封裝在散熱器下方

UBB is the common baseboard
UBB 是共用底板

No host CPU is identified here
此圖未辨識出主機 CPU

IMG_8515.jpeg | Numbered explanation only; original retained | 編號解說 原始照片保留

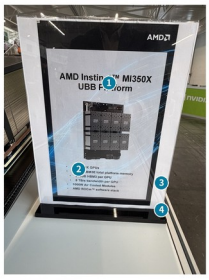
Picture 34 | Eight AMD accelerator modules above the shared baseboard. Author's photograph; numbered annotations. [6]

Count two rows of four. The eight large black assemblies are heat sinks above accelerators; the GPU packages and high-bandwidth memory are underneath. The blue baseboard supplies common power, signal and management connections. A complete server also needs a host processor board, system memory, storage and networking. [6]

The exhibit identification and AMD's product documentation supply the specification context: 288 GB of HBM per GPU and an eight-GPU aggregate of 2,304 GB, approximately 2.3 TB in the displayed units. This sums local memories; it does not promise one uniform access speed to every byte. Rated bandwidth and power also need to be distinguished from measured application results. [6]

The purchase question is how a chosen model uses those resources. Model replicas and a model divided across accelerators create different communication and latency demands. The memory total alone cannot select the best serving strategy, and the photograph does not establish that all eight modules were running. [6]

MI350X specification card
MI350X 規格牌



1
MI350X platform
MI350X 平台名稱

2
8 GPUs and 2.3 TB
8 顆 GPU 與總容量

3
1000 W air cooled
每模組 1000 W 風冷

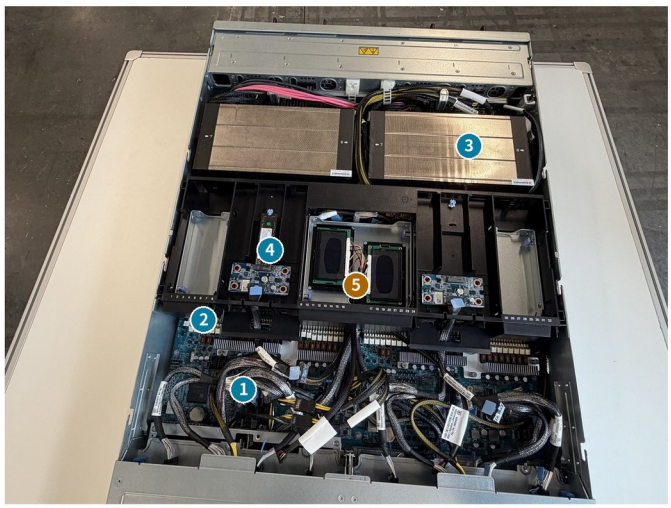
4
ROCm software
ROCm 軟體平台

IMG_8514.jpeg | Numbered explanation only; original retained | 編號解說 原始照片保留

Picture 35 | The MI350X specification sign establishes the product family. Author's photograph; numbered annotations. [6]

Inside the AMD exhibit — memory, storage and unidentified processing regions

Open server layout
拆開的伺服器內部



1
System board and cables
主機板與線材

2
Numbered DIMM positions
DIMM 記憶體位置編號

3
Two large fin assemblies
兩組大型散熱鰭片

4
M.2 drive area
M.2 儲存裝置區

5
Unidentified wired modules
兩個帶線模組 功能待確認

IMG_8510.jpeg | Numbered explanation only; original retained | 編號解說 原始照片保留

Picture 36 | Inside the opened silver server. Author's photograph; numbered annotations. [6]

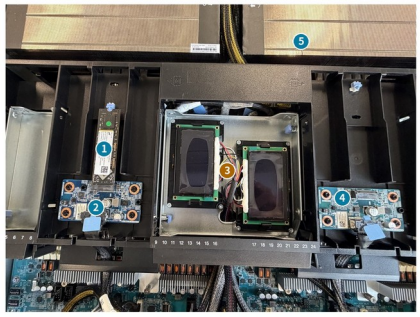
The blue printed circuit board is the system board. The repeated sockets and white latches identify DIMM memory positions; numbering running to 32 denotes positions, not 32 CPUs or GPUs. Black partitions guide airflow, while braided bundles carry power or signals. The two large copper-colored fin assemblies remove heat from components beneath them. Their size does not establish the CPU model or count. [6]

The long narrow module with its white label is an M.2 solid-state drive; M.2 describes its physical format. The two central wired modules remain unclassified in these photographs. A board manual or exhibitor identification is needed before assigning them a processor role. [6]

The server's management label is also not its CPU model. BMC means Baseboard Management Controller: the small management computer used for functions such as remote power control and health monitoring. Asset identifiers help manage equipment, but they do not disclose installed GPU count or peak arithmetic performance. [6]

A useful follow-up is specific: request the server model plate, configured component inventory and a diagram locating each processor and accelerator. Preserve the distinction between what the photograph shows and what a configuration sheet would establish. [6]

Storage and unidentified modules
儲存裝置與待辨識模組



1
M.2 SSD
M.2 固態硬碟

2
M.2 adapter board
M.2 轉接小板

3
Two wired modules
兩個帶線模組 不判定為 CPU

4
Adapter without visible SSD
轉接板 未見長條 SSD

5
Large heat sinks
大型散熱器 下方零件未確認

IMG_8513.jpeg | Numbered explanation only; original retained | 編號解說 原始照片保留

Picture 37 | M.2 storage and the two unresolved wired modules. Author's photograph; numbered annotations. [6]

AMD detail — the small supporting components make the large accelerators usable



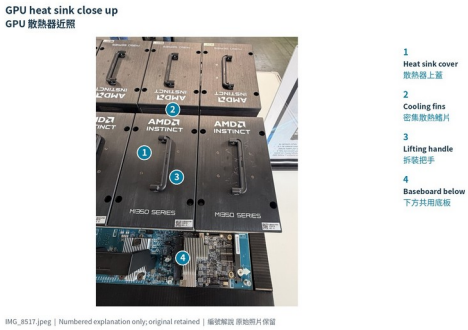
Picture 38 | Motherboard wiring, memory latches and local cooling. Author's photograph; numbered annotations. [6]

The motherboard close-up reveals the work surrounding the processors. White latches secure memory modules. Small silver fin blocks cool local circuits, potentially controllers or power regulation. The coin-cell battery supports functions such as the real-time clock; it does not power the server. Cable sleeves protect bundles, while connector functions still require the board documentation. [6]

The large black blocks are cooling assemblies. Fins increase surface area, allowing server-level airflow to remove heat. The path runs from the semiconductor through thermal-interface materials into the heat sink and then the air. No small fan needs to be visible on each block for an air-cooled server design to use it. [6]

MI350 SERIES names the family; the nearby MI350X sign supplies the more specific exhibit identification. This close-up covers only part of the array. It does not replace the eight-module count in the wide photograph, and it does not expose the GPU die or HBM directly. [6]

The connection to SGLang is operational. Storage holds model files; host processing and system memory coordinate loading and requests; accelerator memory holds data used for model execution. ROCm, frameworks and the serving layer must match the actual devices and software versions. Efficient software can improve utilization, but it cannot add physical GPUs. [6]



Picture 39 | The heat-sink assembly above an AMD accelerator. Author's photograph; numbered annotations. [6]

Episode 2 conclusion. The restaurant site in Episode 1 and the server here share a commercial constraint: the whole operating system must work. A buyer needs supported configurations, measured service quality and full-system costs. A photograph is valuable evidence of components and context, but it cannot supply the missing benchmark or configuration inventory. [2, 6]

4.7 Uni³Sim: test the system design before buying it

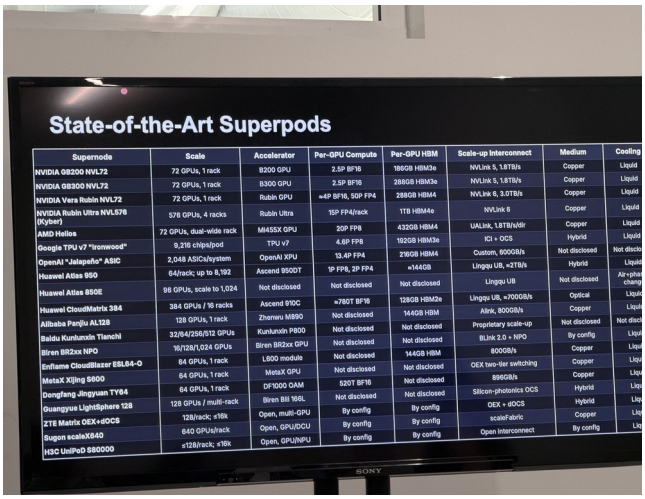


Picture 40 | Zhenhua Zhu on Uni³Sim. Author's conference photograph. [20]

Zhenhua Zhu's Uni³Sim case belongs to AI infrastructure design. It connects scheduling behavior with models of computation, memory and interconnect. The purpose is to explore how system choices interact, rather than optimize a single component in isolation.

Zhenhua Zhu presented Uni³Sim as a university research project associated with Tsinghua Shenzhen International Graduate School. The evidence supports that research identity, not a separately funded venture or an established commercial customer base. I have not audited its repository. [20]

A calibrated simulator can help eliminate poor configurations before hardware procurement. Its usefulness depends on whether predictions match measured systems and whether that accuracy extends beyond the calibration cases. A visually persuasive simulation is not a substitute for validation, and predicted and measured latency must be compared in consistent units.



Picture 41 | Uni³Sim system configurations. Author's conference photograph. [20]

The table compares accelerators, scale, memory, links, transmission media and cooling. It illustrates a large design space. [20]

Read each column with units. BF16, FP8 and FP4 performance are not directly interchangeable. Per-chip and per-rack numbers differ; the slide contains a per-rack entry beneath a per-GPU heading. Product specifications and availability have not been independently checked. [20]

Electronic design automation helps design and verify chips. System-level EDA expands attention to packages, boards, racks and networks. [20]

Locally optimal parts may form a poor system. Five chips, four memory designs, three networks and six deployment policies produce 360 combinations in an illustrative example. Simulation screens combinations before physical tests. [20]

Requests enter a batch; model and parallelism settings generate kernel tasks; execution changes memory occupancy and elapsed time; these states return to the scheduler. Longer execution lets more requests accumulate. H2D means host to device, D2H the reverse, and D2D device to device. [20]

A kernel is an executable device program, such as matrix multiplication. Peak hardware specifications do not describe every workload. [20]

First estimate operations, execution waves, cache misses, memory traffic and concurrency. Then convert resource demand into service time using hardware equations. Sequential work adds time; independent work may overlap when resources permit. White-box means assumptions are inspectable. [20]

The team reports 1027 cases: 221 GEMM, 334 attention, 176 MoE and 296 vector cases. These measure timing errors, not answer correctness. [20]

Uni³Sim mean absolute percentage errors are 7.6% on A800, 7.1% on MetaX C500 and 12.9% on Ascend 910B. Scatterplot horizontal axes are measured latency; vertical axes are predicted latency. Points on the diagonal agree; logarithmic axes cover many scales. [20]

The rack and memory cases change the meaning of capacity. A device count is an inventory measure. Useful inference capacity emerges only after memory, data movement, power, cooling, scheduling and software work together. That is the system the application team must ultimately buy or operate.

5 From Fast Tokens to Useful Work

A user wants a completed result. The serving engine supplies one part of it. The remaining work may include a search, a tool call, a security decision, a verification step or a human review. These cases show why a fast model and a useful application require different measurements—and why the boundary around the task matters.

5.1 Measure the complete workflow

Arcee's Fast Tokens, Slow Agents session is an essential counterweight to throughput marketing. An agent is a system that may retrieve information, call a model, execute a tool, observe the result and repeat. The model is one component inside that sequence. [12]

Imagine a hypothetical research workflow taking 60 seconds: 15 seconds generating text, 30 seconds waiting for searches and tools, and 15 seconds validating and assembling the result. Making generation three times faster reduces its contribution to five seconds. The total becomes 50 seconds, a 1.2× speedup, not 3×. This arithmetic is an original teaching example, not a reported Arcee benchmark.

This does not make inference optimization unimportant. It clarifies where its benefit appears. If generation dominates, it can matter greatly. If the tool path dominates, the next improvement may be caching, parallel retrieval, better queries, fewer retries or a simpler task design. Quality failures can also dominate cost when an answer must be discarded and regenerated.

The same logic applies to customer analysis. A language model may summarize complaints quickly while the project remains blocked by fragmented customer identifiers, missing consent records or inconsistent transaction histories. Improving the model will not reconcile those records automatically. The system should measure each stage so investment follows the actual constraint.

The research implication is to report two levels of performance: the component result and the business-task result. The former helps engineers reproduce a technical improvement. The latter tells the buyer whether the improvement matters. Both are needed for a credible account of the inference economy.

Table 11 | Illustrative end-to-end task

Stage	Before	After 3× faster generation
Model generation	15 seconds	5 seconds
Search and tools	30 seconds	30 seconds
Validation and assembly	15 seconds	15 seconds
Total	60 seconds	50 seconds

5.2 Arcee: the agent still has work to finish

Lucas Atkins' Fast Tokens, Slow Agents session places the model inside a larger software system. Prompt construction, retrieval, tools, context reuse, sampling, evaluation and environment design can all determine task completion time and quality.



Picture 42 | Lucas Atkins at the lectern. Author's conference photograph. [12]

This changes how a buyer should interpret model performance. A faster token stream may be valuable, but it does not eliminate tool latency or correct a poor workflow. Repeatedly adding context can also increase input work and make failures harder to diagnose.

Agent systems should be profiled as complete processes. Measure where time is spent, why retries happen and which outputs pass validation. This is directly relevant to the restaurant case: the model should assist a reproducible analytical pipeline, not conceal missing data or undefined decision rules beneath a fluent narrative.



Picture 43 | Fast Tokens, Slow Agents. Author's conference photograph. [12]

The slide compares Generate → Read → Decide with Generate → Validate → Execute. The output consumer changes from a person to a program. [12]

Humans need comprehension time. Once output exceeds comfortable reading speed, extra speed can have diminishing value. Machines can act on a valid instruction immediately. [12]

For a spreadsheet task, a person reads directions and clicks menus. An agent validates a structured command and calls a function. Validation must check meaning and permissions, not just syntax. [12]

Machine consumption reduces the human reading bottleneck, but network, startup and execution delays remain. [12]

Both bars contain time to first token (TTFT), usable-output generation and tool execution. Decoding generates output tokens. [12]

A tool often cannot execute half a command. It needs the complete filename, arguments or patch. The first visible word is not necessarily usable work. [12]

Shortening the orange generation segment moves the dotted handoff earlier. The blue execution segment starts earlier but retains its duration. No numerical time scale is provided. [12]

T(s) = D/s + R. D is original decode time; s is its speedup; R is time not shortened by that particular optimization. [12]

As generation becomes faster, queues, sandbox setup, file operations and tools dominate a larger share of the job. [12]

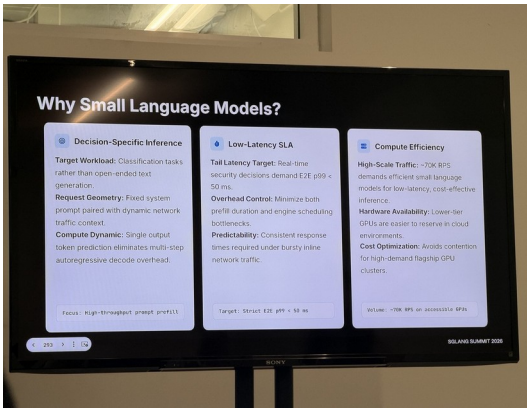
5.3 Palo Alto Networks: a late security answer can miss its purpose

Matthew Schulz and Zhemin Su’s Palo Alto Networks session examines security inference under a demanding latency target. Model execution serves a decision for which a late answer may have little operational value. Target-compliant output therefore matters more than a high average throughput figure alone. [21]

The displayed comparison reduces p99 latency from 53.1 to 44.2 milliseconds, approximately 16.8%. It also increases the number of concurrent clients passing the target from two to three. These are different denominators, and neither permits a service-wide request-volume figure to be assigned to one GPU. [21]

The case illustrates how to assess latency-sensitive AI: specify the workload, hardware, model and pass criterion, then count failures as well as successes. The slide and speech record differ between 7B and 8B when describing model size; that discrepancy remains unresolved. [21]

Illustrative example: if 10,000 requests have a p99 near 50 ms, roughly 100 requests can lie in the slower 1% tail, depending on percentile calculation and ties. A safety-critical business still needs a policy for those delayed requests. This example is not a count from the talk. [21]



Picture 44 | Why language models for security inference. Author’s conference photograph. [21]

Prefill processes the supplied input and produces the information needed to select the first output token. A one-token classification avoids repeated autoregressive decode steps for subsequent tokens. It does not eliminate input attention, output selection, or all runtime scheduling work. The later scheduler bug illustrates why even a one-token task can incur unwanted extra execution. [21]

SLO is a service-level objective, an operational target. SLA generally refers to a service-level agreement, potentially contractual. The transcript uses SLO while the slide says Low-Latency SLA. No contractual terms were supplied, so this report calls 50 ms a target. [21]

The speaker describes support for different model families and inference frameworks, serving both online security decisions and offline batch processing. Unclear names in the speech record remain unresolved rather than being assigned to products. [21]

The architecture discussion covers decoder and encoder models, several serving frameworks and heterogeneous accelerators. The design principle is to choose compatible software and hardware for each model. It does not establish that every named framework supports every model on every device; unclear words in the speech record are left unresolved. [21]

SGLang is the computation-serving layer for the tested language model. Ray Serve and Kubernetes address higher-level application and infrastructure concerns. A slow request may originate in waiting for a replica, CPU preprocessing, GPU execution or transfer back to the client. Optimizing one kernel cannot remove every other delay. [21]

5.4 Phala: privacy must survive the execution path



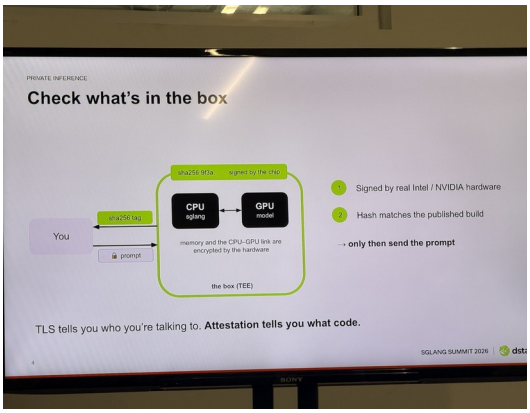
Picture 45 | Hang Yin presenting confidential inference. Author's conference photograph. [15]

Hang Yin's Phala session addresses confidential inference: protecting data during execution and giving a client evidence about the protected environment. He describes attestation, client verification and the difficulty of observing and tuning a system without exposing private content. [15]

The performance details show why security and serving architecture interact. Matrix operations can remain fast while transfers or CPU scheduling become bottlenecks. The availability of direct GPU communication mechanisms depends on the protected configuration described in the talk.

Confidential computing can expand the set of workloads a customer is willing to deploy, but it is not a universal guarantee of model correctness or data governance. The trust model, software identity, key handling and remaining exposed metadata must be explicit. The title's 200-billion-tokens-per-day figure remains a vendor scale claim.

Attestation is evidence about a protected environment and its measured software state. The slide shows a hardware-signed measurement being returned before a prompt is sent. [15]



Picture 46 | Confidential execution and attestation. Author's conference photograph. [15]

A provider could display a privacy label without actually running the expected configuration. Verification makes that claim testable. [15]

The client checks the evidence's authenticity, acceptable measurements and connection to the endpoint receiving its secret. A SHA-256 hash is a digital fingerprint; it is not encryption or a complete code review. [15]

A valid measurement can establish identity and configuration under the verification policy. It does not prove the application has no bugs, cannot leak outputs, or gives correct AI answers. [15]

The next step connects public source code to the measured build running on the server. That link adds evidence about what software is actually deployed. [15]

Publishing code does not prove a service runs that code. The service could deploy a modified version. [15]

A reproducible build aims to let another person use the same inputs and tools to obtain the same output. The verifier compares the expected build identity with the attested identity. [15]

The chain is source review, controlled build, measured deployment and verified connection. Checking only a website's claim or only a repository leaves gaps. [15]

A production service has a changing fleet of workers. Workers are processes or machines that perform inference, not necessarily one GPU each. [15]

Nodes restart, expand or move. Verifying one machine yesterday does not establish the identity of whichever machine receives today's request. [15]

The diagram asks which workers were checked. Verification must follow membership changes rather than remain a one-time manual task. [15]

5.5 Core Automation: make research experiments easier to run

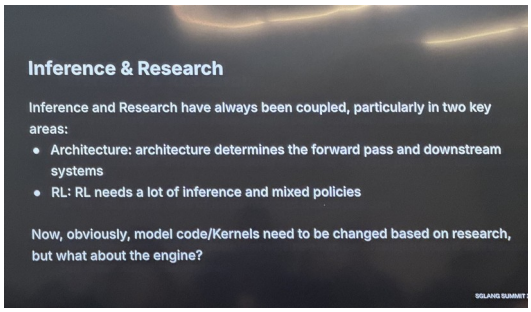
Francis Zhang considers inference as a research tool whose behavior needs to be predictable. His session examines session identity, cache behavior and separate memory pools as ways to organize work and avoid unwanted interactions between experiments or request classes. [13]



Picture 47 | Francis Zhang on research infrastructure. Author's conference photograph. [13]

Caching is useful only when the system knows exactly which prior state can be reused. A hidden change in context, model configuration or session handling can invalidate an apparently comparable run. For a researcher, reproducibility can matter as much as a peak speed result.

Inference infrastructure should make experimental controls visible. The presentation does not establish a universal measured speedup from the proposed architecture. Its relevance is the disciplined management of state and request boundaries, which can improve the interpretability of research results. [13]



Picture 48 | Inference and research. Author's conference photograph. [13]

The slide identifies two connections: architecture determines the forward pass, and reinforcement learning needs many inference runs. Architecture means the model's internal arrangement of computations. Reinforcement learning, or RL, improves behavior using outcomes and rewards. [13]

If a researcher changes the attention mechanism, the stored state and memory needs may change. If RL needs thousands of attempted answers, slow generation can delay the next learning step. The engine therefore affects which experiments are practical. [13]

A simplified RL loop has four steps: the model attempts a task; an evaluator or environment supplies a result; a training procedure updates the model; the updated model tries again. The attempt is often called a rollout. Inference produces that experience; the subsequent training step changes the weights. [13]

Technical background: a token is a piece of text, sometimes a whole word and sometimes part of one. A log probability is the logarithm of a probability. It makes products of many token probabilities easier to handle as sums. It is a model score, not a guarantee that an answer is true. [13]

Illustrative example: a math tutor generates ten possible solutions, checks them, and uses the feedback in a later learning update. Generating and scoring those attempts consumes inference even though no customer is waiting in a chat window. [13]

Research infrastructure must make repeated model execution dependable and reasonably efficient. "Mixed policies" also means the system must track which model version produced which data; the talk does not specify its exact policy mixing algorithm. [13]

An abstraction is a useful interface that hides details. Isolation means keeping an unrelated change from forcing you to understand the whole system. A researcher should be able to describe a new attention method without also redesigning every memory allocation rule. [13]

These applications require different acceptance tests. An agent must finish a valid task; a security service must answer within its decision window; confidential inference must preserve its trust boundary; a research engine must support a reproducible experiment. Faster tokens can help each, but they do not substitute for those tests.

6 Inference That Searches, Learns and Verifies

Inference can produce more than a final answer. It can generate an attempted proof, an action, a candidate image or a research hypothesis that another process checks. Those attempts may become feedback for the next model update. The commercial opportunity expands when additional computation produces more reliable results; it weakens when the extra work merely repeats an untested assumption.

6.1 SGLang and Miles: serving feeds the learning loop

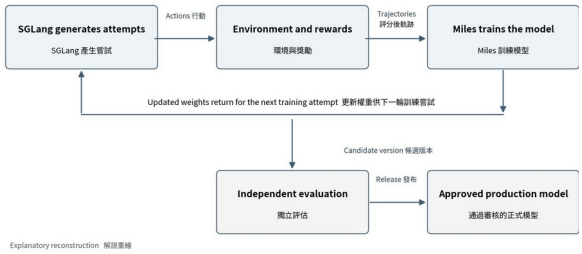
The opening SGLang and Miles discussion placed serving and post-training in the same engineering cycle. Reinforcement learning calls on inference to generate experiences; the training system evaluates them, updates weights and returns a new model version to the generation side. [3]



Picture 49 | Ying Sheng — official summit portrait. Official summit material. [3]

There are several organizational relationships here. SGLang is an open-source project hosted by LMSYS Org. Miles is an open-source framework maintained in RadixArk's GitHub organization, with acknowledged origins in slime. RadixArk offers commercial infrastructure around both. This is a relationship among projects, maintainers and commercial services; the evidence does not establish a merger between two companies called SGLang and Miles. [3]

Integration can reduce compatibility and operational work, but the buyer must still examine supported configurations, exit paths and the quality of the learning loop. Open code does not make deployment free, and a managed service does not remove the need for independent evaluation.



Picture 50 | Serving and reinforcement-learning feedback. GII explanatory diagram. [3]

Maintainers' technical decisions and a sponsor's commercial interests should be distinguished. Open code gives users access to implementation and an alternative path to operation, while packaging, maintenance and managed service can remain valuable paid work. [3]

The combined offering can reduce the work required to turn an open model into a tailored, operating service. Customers may pay for deployment and operations even when the underlying code is freely available. However, adoption, paid conversion, customer economics and provider profitability are different measurements. None can be inferred from GitHub stars or a page of company logos. [3]

The wider SGLang Consortium is a separate form of collaboration. It brings hardware and infrastructure participants into a shared technical forum while maintainers retain technical decision authority. Its economic logic is to improve a common foundation that participants use in different businesses. It is not evidence that those businesses share revenue, ownership or customer data. [3]

The loop illustrates the relationship in Miles documentation. Production release is shown as a separate evaluation decision rather than an automatic consequence of each training update. [3]

An agent makes this dependency stronger. One task may require reading a file, generating code, running a test, interpreting an error and trying again. Each generation consumes inference resources; each external tool can introduce unpredictable waiting time. Faster generation is useful, but the total experiment also depends on environments, reward calculation, training and synchronization. Improving one component alone may expose a different bottleneck. [3]

6.2 Miles: the learning loop needs an operating record



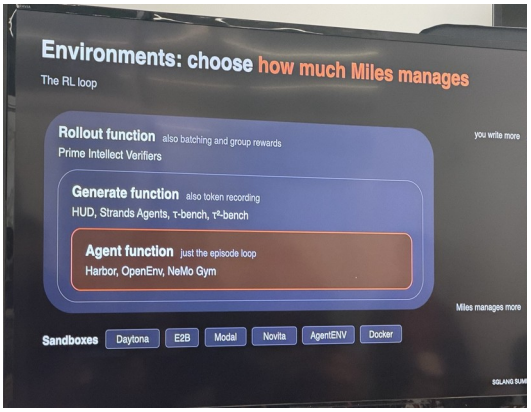
Picture 51 | Miles session title naming Zhichen Zeng. Author's conference photograph. [11]

Zhichen Zeng's Miles session focuses on production-level post-training. The system must coordinate generated trajectories, rewards, model updates and recovery while preserving the information needed for valid learning calculations.

Zhichen Zeng identifies several ways rollout and training can disagree: token handling, expert routing, sparse attention, kernels and asynchronous staleness. These differences affect the relationship between the policy that generated an attempt and the policy being trained. [11]

Scaling reinforcement learning is not just a matter of generating more attempts. A production system needs consistent records, versioning and checks on the data it consumes. Faster rollout production is useful only when the resulting learning remains valid and improves an independent evaluation.

The agent interacts with an environment over several turns. Its complete history and reward enter a versioned, asynchronous buffer. The trainer takes a batch, calculates updates, and publishes a new weight version. Version labels record which model produced which data. The slide lists Megatron-LM, FSDP2, and TorchTitan as training choices and several transports for returning weights. [11]



Picture 52 | Environments for Miles post-training. Author's conference photograph. [11]

For a coding exercise, the agent edits a program, runs tests, reads errors, and edits again. Passing the tests produces a reward. The trainer uses many such histories to adjust the model. A single successful attempt is training material; it does not by itself establish general coding ability. [11]

The slide offers three integration levels. A rollout function gives you more control over the whole process. A generate function includes responsibility for token recording. An agent function focuses on the episode loop while Miles manages more of the surrounding work. [11]

The nested boxes show a responsibility boundary. At the outer level, users handle more, including batching and group rewards. At the inner level, they mainly write the agent's episode loop. Sandboxes provide the controlled computers where tool actions run. An environment defines the task and feedback; a sandbox supplies the execution space. [11]

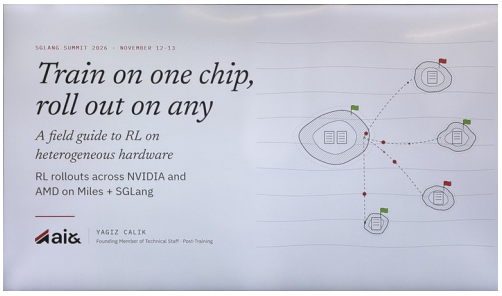
A session server sits between an agent harness and SGLang. The harness manages messages and tools. The server records the exact tokens generated by the model and assembles histories for the trainer. [11]

6.3 Heterogeneous RL: coordinate the devices and the policy

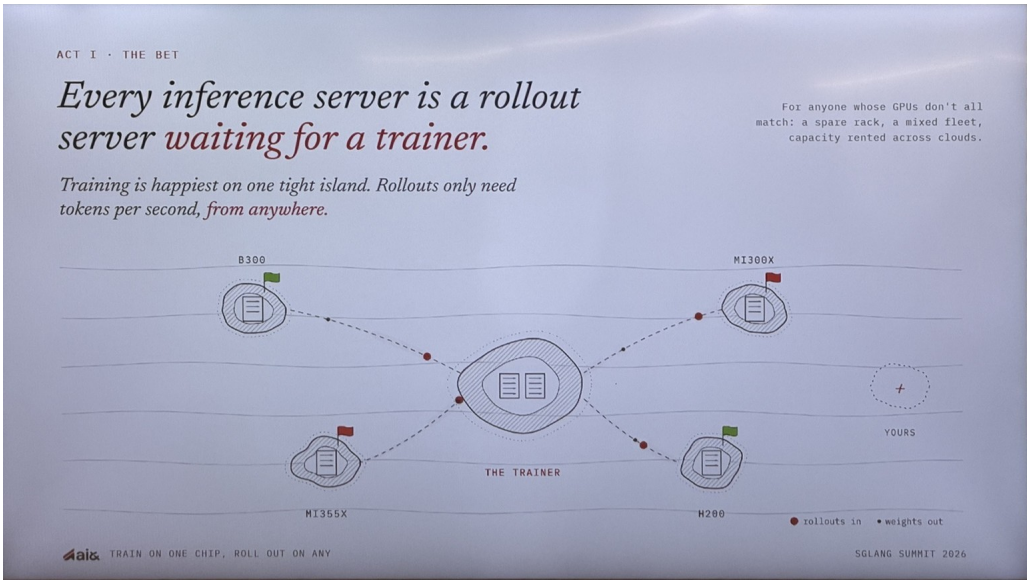
Yağız Çalık's session describes reinforcement learning across multiple GPU types and vendors. The presentation reports a 48-GPU setup with four device types and two vendors, separating training from rollout generation. [22]

Its reported throughput and trainer-idle-time improvements describe a particular engineering configuration. A smaller adapter update can be easier to distribute than an entire large base model, but the system still needs compatibility and numerical consistency. Increased rollout throughput does not independently establish better model capability.

Heterogeneous capacity can become more useful when software assigns work to appropriate devices. This may improve flexibility and reduce stranded resources, but integration effort, communication and reliability belong in the cost comparison.



Picture 53 | Train on one chip, roll out on any — session title. Author's conference photograph. [22]



Picture 54 | Every inference server as a rollout server. Author's conference photograph. [22]

The central island is a coordinated training system. The surrounding islands run model copies on B300, H200, MI355X and MI300X hardware. The proposal draws on useful capacity wherever a compatible engine is available. The "Yours" island is an invitation to integrate another compatible deployment, not a claim of automatic support. [22]

Rollout data returns toward the trainer; updated model settings travel outward. GPUs within a distributed trainer communicate frequently. Separate rollout replicas can attempt different problems more independently, although each replica may itself need several closely connected GPUs. [22]

"One trainer" does not mean one GPU. A node is a server or computing unit; an engine is a serving process or deployment; a GPU is a physical accelerator. Those counts must not be substituted for one another. [22]

The speaker depicts NVIDIA capacity as scarce and shows alternatives including AMD Instinct, Google TPU, AWS Trainium, Etched, Tenstorrent and Rebellions. This is an opportunity map, not a test result for every device on it. Only the four NVIDIA and AMD models are shown as passing this experiment's gates. [22]

If inference engines generate much of the practice material in an RL job, an organization may be able to use compatible spare inference capacity for that work. Availability, memory, numerical behavior and software support all matter; a chip's brand alone is insufficient. [22]

Correction and The transcript's inference-only description is too broad: Google TPU and AWS Trainium also support training. The session's scarcity framing is a presenter claim, not a live market inventory. [22]

The "contract" is a technical interface agreement. The engine must provide compatible numerical behavior, RL-specific features and a supported update mechanism. The vendor name is deliberately not a requirement. The four "passed gates" stamps are claims about the tested configurations. [22]

6.4 ApprenticeBench: evaluate a continuing job



Picture 55 | ApprenticeBench title naming Xiang Deng. Author's conference photograph. [23]

Xiang Deng's ApprenticeBench presentation asks whether an agent can improve across a continuing stream of related work. Its simulated construction-accounting setting includes historical records, company instructions, mentoring and changing conditions. That evaluates continuity in a job rather than independent questions with a reset after every answer. [23]

The work is associated with NeoCognition. The presented comparisons examine configurations with different access to history and feedback. They are benchmark evidence about learning from experience, not proof that autonomous accounting can be deployed reliably without oversight. [23]

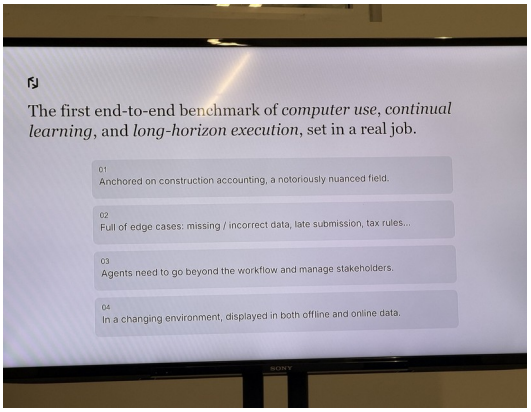
Workplace AI should be evaluated on continuity, error recovery and appropriate use of prior experience. A rising score can also result from easier later tasks, so comparisons should control task difficulty and access to feedback. For the research agenda developed here, the question is whether experience improves organizational capability across continuing work.

Can an agent learn a whole role? A good exam score does not guarantee workplace competence. Evaluate performance across related assignments, rather than reset after each question. Measure growth as well as initial ability. [23]

Computer use, continual learning and long-horizon execution in one benchmark. Missing data, late vendors and policy changes disrupt fixed workflows. Use construction accounting tasks with exceptions and stakeholders. Following a script is not enough. [23]

An agent navigates business software and compares records. Processing one bill includes searches, checks and decisions. Inspect records, consult precedent and enter the appropriate information. One business task can contain hundreds of interface actions. [23]

Video playback time, the internal task timer and cost counters measure different quantities. They should be interpreted separately when evaluating the demonstration. [23]



Picture 56 | ApprenticeBench learning across work. Author's conference photograph. [23]

Historical examples precede hands-on and hands-off mentoring. A new employee needs company-specific knowledge. Provide a handbook, Odoo tutorial, posted bills and completed work. Treat learning conditions as part of the evaluation. [23]

The slide labels 35 tasks over May–July; results later use 100 tasks. Simulated business months do not imply a year of wall-clock runtime. [23]

Observe, decide, click or type, and check the result. Each screen adds information and each action can fail. An agent repeatedly interprets the interface while retaining relevant context. Good reasoning must survive a long chain of small operations. [23]

An API is a software entrance for structured requests; a GUI is the screen and buttons a person uses. One API call is not a universal substitute for a whole workflow. [23]

6.5 Axiom Math: generation joined to formal verification



Picture 57 | Tobias Gessler and Rishi Malhotra presenting Axiom Math. Author's conference photograph. [24]

Tobias Gessler and Rishi Malhotra's presentation links mathematical generation with execution, retrieval and learning. AXILE supports proof execution and reuse; Atlas addresses mathematical search; Snorlax supports training infrastructure. Together, these components explain why a mathematics company needs substantial inference engineering.

The valuable unit is a checked mathematical artifact or a useful step toward one. A model can propose many candidates, but the formal environment determines whether a candidate satisfies the encoded statement. This creates feedback with a different character from a subjective text preference.

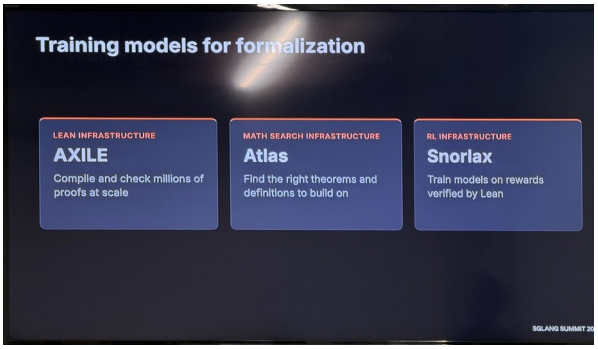
This case demonstrates how inference can support search with strong rejection criteria. The remaining boundary is specification. A valid proof under stated definitions and axioms is not a guarantee that the original human problem was formalized correctly. Financing, repository scale and reported problem-solving achievements should each be evaluated separately from customer revenue or commercial deployment.

Agents need existing knowledge, feedback and a training process. Otherwise they repeatedly rebuild foundations or accumulate errors. [24]

Treat the three as a proof workshop, library search and training coordinator. The model uses tools across this system. [24]

Autoscaling handles resources; cached artifacts share prior builds; persistent sessions reuse elaboration state; tools provide goals, diagnostics and final verification. Elaboration resolves source expressions into explicit structures. [24]

Reuse can reduce waste, but the talk does not quantify savings with a controlled benchmark. Cache validity must follow versions and dependencies. [24]



Picture 58 | Axiom Math training and proof infrastructure. Author's conference photograph. [24]

Retrieve the statement, assumptions, source and dependencies. The Dade isometry example shows linked results. A positive-number theorem cannot automatically justify an all-number claim. [24]

Search speed helps only if retrieval is relevant and assumptions match. No exact latency benchmark is supplied. [24]

An unsuccessful check identifies remaining goals or invalid steps. The agent revises or searches again. Final verification must preserve the original statement and accepted assumptions. [24]

The driver dispatches tasks to rollout workers. Workers manage sandboxed agent sessions, call AXILE and Atlas, and request model outputs through routers and SGLang samplers. Completed trajectories return for training. [24]

SGLang generates candidates; Lean-related verification checks proofs. The exact optimizer, reward weights, GPU count and cost are undisclosed. [24]

Illustration: prove a withdrawal function never creates a negative balance for valid inputs. The specification must account for actual rules, including fees or concurrency when relevant. [24]

Verification covers stated properties under assumptions; deployment still requires checks for risks outside the formal specification. [24]

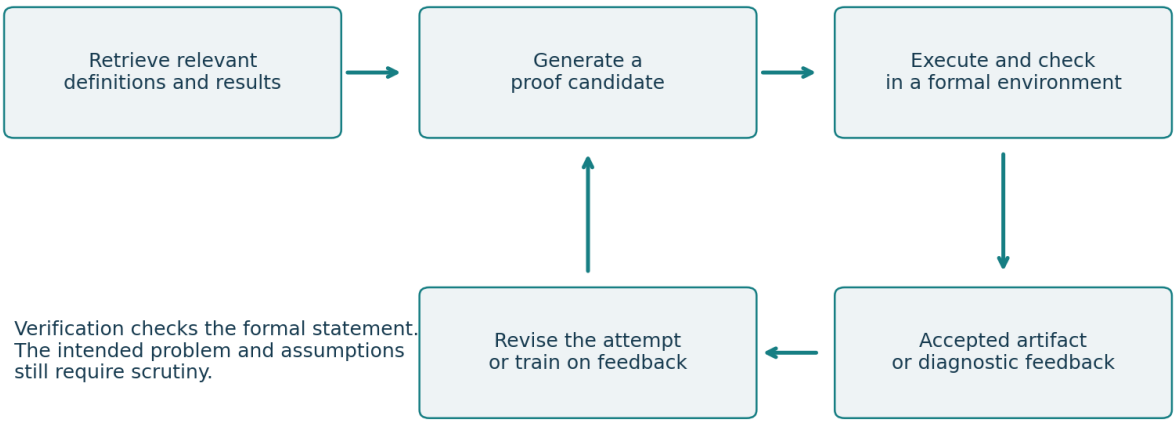
Axiom Math provides a particularly clear example of a valuable inference loop. A model proposes mathematical work; retrieval supplies relevant prior knowledge; an execution environment checks formal proof steps; feedback guides further attempts and learning. The result is more than a long text answer. It is a candidate artifact subjected to a defined verification process. [24]

Verification changes the economics of repeated attempts. If failed candidates can be rejected reliably, additional inference may be spent on search without requiring a human to manually inspect every generated sentence. But the verifier checks a formal statement under its definitions and assumptions. It does not guarantee that the statement captures the original human intention or that a software specification covers every real-world requirement.

The timing of a result changes its meaning. Axiom's account distinguishes eight Putnam problems solved within the competition window from four solved later. All twelve eventually is a different achievement from all twelve during the contest.

Formalizing known mathematics also differs from discovering a new theorem, and lines of code alone do not measure mathematical depth. [24]

Axiom's strategic importance is the combination of generation and checkable feedback. The generalizable question for other industries is: can we create a verifier or outcome measure strong enough to make repeated inference useful? For customer intelligence, that measure may be a calibrated prediction or a controlled campaign result rather than a formal proof.



Picture 59 | Generation, formal verification and feedback. GII explanatory diagram; assumptions are stated in the text.

The formal checker provides a way to reject invalid candidates. Retrieval supplies relevant statements and dependencies. The generator proposes a proof step or candidate, and diagnostic feedback can support another attempt. Training uses accumulated experience to change later behavior.

The commercial lesson is not that every business problem has a mathematical proof. It is that reliable feedback changes the usefulness of repeated inference. For code, a test suite may provide partial evidence. For customer targeting, a controlled experiment measures an outcome. The strength and coverage of the evaluator determine what conclusions can be drawn.

A checked artifact still has a scope. Its definitions, assumptions and intended use must be understood. I have not independently rebuilt the large formalization projects discussed at the conference; their reported completion remains attributed to Axiom. [24]

6.6 Axiom’s financing and the commercial test

Axiom’s public materials identify Carina Hong as founder and CEO, Shubho Sengupta as CTO and a founding partner, and Ken Ono as founding mathematician. Those are distinct roles; varied public wording does not establish legal cofounder status or equity ownership. Tobias Gessler and Rishi Malhotra presented the technical work at the summit. [24]

Axiom’s dated fundraising disclosures give commercial context to the research program. Capital raised indicates financing obtained, not cash currently remaining, customer revenue or proven product-market fit. The valuation of one round should not be added to the valuation of another. [24]

Table 12 | Axiom’s disclosed financing rounds

Recorded round	Capital raised	Valuation treatment
October 2025 seed	US$64 million	US$300 million; basis unspecified
March 2026 Series A	US$200 million	US$1.6 billion+ post-money
Disclosed total	US$264 million	Valuations are not additive

Axiom identifies B Capital as the seed lead and Menlo as the Series A lead. The disclosed rounds total $264 million raised. Public profile counts and company size bands do not establish an exact payroll headcount, so none is asserted here. [24]

Commercial evaluation should move from research artifacts to a customer's problem. A plausible route is a narrowly specified verification task integrated into a developer workflow, with measured reductions in error or review effort. That is the author’s analysis, not a confirmed product calendar or revenue forecast. The evidence needed next is a reproducible customer deployment, a clearly scoped specification and an economic comparison with the existing process. [24]

6.7 Apodex: investigate the evidence behind agreement

Simon Shaolei Du's Apodex presentation connects multi-agent research with evidence verification and future-event prediction. Multiple teams can investigate a question, but their agreement is useful only if the evidence and interpretation are sound.



Picture 60 | Simon Shaolei Du presenting Apodex. Author's conference photograph. [25]

The presentation's football example shows a majority of agent teams following a misleading interpretation while an aggregator selects an alternative. It illustrates the need to inspect the evidence path and task definition, rather than count votes alone. It does not establish that minority answers are usually correct. [25]

Agent orchestration adds value when it diversifies investigation, preserves provenance and tests counterevidence. Repeated access to the same weak source can create correlated error. The closing Apodex 2.0 statement is a roadmap announcement in the source record, not confirmation of a shipped model.



Picture 61 | Reasoning as a team sport. Author's conference photograph. [25]

Like a newsroom: reporters gather evidence, analysts interpret it, fact checkers audit claims and an editor integrates the report. Asynchronous teams return results when ready. [25]

Up to 150 sub-agents and over 15000 steps describe scale, not guaranteed quality or a requirement for every task. Verifiers can also fail. [25]

Nodes hold findings and tentative claims; edges show support or contradiction. Three websites repeating one source are not three independent confirmations. [25]

The slide claims gains of 14.8 and 18.4 points on two benchmarks. Baselines and additional cost are needed to judge efficiency. [25]

Five of eight trajectories selected three goals after reading a modal 3–0 score forecast. The task actually asks total-goal buckets. [25]

From the slide: P(3 goals)=P(over 2.5) minus P(over 3.5)=85% minus 63%=22%. P(5+)=P(over 4.5)=45%. The six buckets are 2%, 4%, 9%, 22%, 18%, 45%. [25]

45 percent is below half but the largest single bucket. Rebuilding probabilities from raw odds requires margin and timestamp checks. [25]

The aggregator selected 5+ against a five-of-eight majority. The slide reports a final score of Germany 7–1. [25]

The verifier reinterprets the task, uses the more relevant total-goal distribution and checks supporting context rather than simply counting answers. [25]

Evidence re-weighting beats vote-counting is the lesson of this case, not a theorem that overturning majorities always works. The match outcome remains a speaker-reported result here. [25]

The prediction harness asks each trajectory to challenge its initial candidate and propose up to two alternatives. [25]

Like preparing a debate, identify the strongest objection. Each path returns one to three candidates with confidence; aggregation reasons over evidence packages. [25]

Reflection is useful when it changes evidence and reasoning. Claimed confidence must be tested for calibration over many events. [25]

6.8 Krea: the reward must preserve the intended image

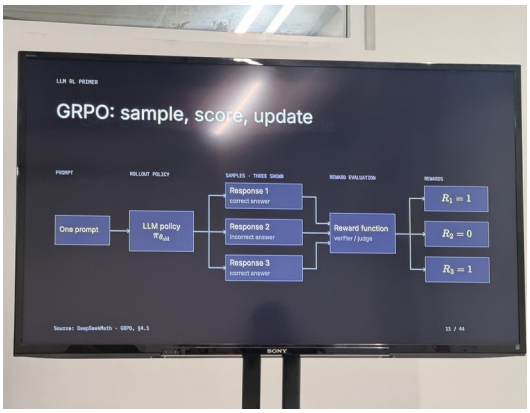


Picture 62 | Sangwu Lee presenting Krea. Author's conference photograph. [26]

Sangwu Lee's Krea presentation explains reinforcement learning for image generation and editing. The challenge is to evaluate outputs that combine instruction following, visual quality, text rendering and other task-specific requirements.

A narrow reward can encourage an undesirable shortcut. Correctly rendered text does not guarantee a good composition; penalizing one visual behavior can overcorrect and harm another. Krea's examples make reward design an operating problem with visible consequences. [26]

Evaluation must represent the customer's actual task. More generated candidates create opportunities to improve, but also opportunities to optimize the wrong objective. This lesson carries directly into retail and restaurant applications: a metric such as click rate, speed or redemption can be improved while the overall customer or business outcome deteriorates.



Picture 63 | Krea reinforcement-learning update. Author's conference photograph. [26]

GRPO means Group Relative Policy Optimization. Compare rewards within the group; an advantage expresses relative performance. The diagram shows scoring, not the full update equation. [26]

Noise becomes a clean image, the image is scored, then fresh noise is added for training. [26]

DiffusionNFT means Diffusion Negative-aware Fine-Tuning. The learning objective uses reward information and flow matching rather than requiring the full sampled trajectory. [26]

Separate preference from instruction compliance. A viewer may dislike a style even when the edit follows the brief exactly. [26]

If the judge misreads the image, its reward can punish a correct edit or approve an incorrect one. [26]

Compare the intended text with recognized characters and inspect uncertain cases. Noise here includes scoring errors, not only the random noise used during generation. [26]

Use separate criteria for what the text says and how it looks. Preserve the hierarchy between headline and supporting text. [26]

Add a typography judge alongside the text reader. Examine depth, style and integration with the paper, not just recognized characters. [26]

A perfect proxy score can coexist with an unwanted edit. This is an example of reward exploitation. [26]

6.9 MiniMax: coordinated tools expand the output



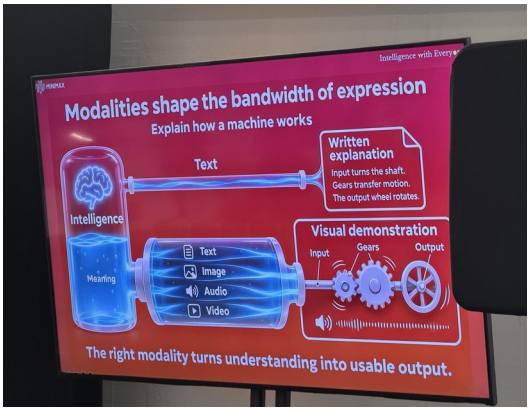
Picture 64 | Ryan Lee on the Open Stage. Author's conference photograph. [27]

Ryan Lee's MiniMax session examines the path from language-model orchestration to richer multimodal systems. A model can call separate tools for image, audio or code output; an end-to-end architecture seeks closer coordination across representations and outputs.

MiniMax connects the technical direction to developer services and consumer applications in China and international markets. English-language accessibility alone does not quantify U.S. revenue or establish local enterprise adoption. Geographic economics require their own disclosures and definitions. [27]

Multimodal capability becomes a business when it fits a repeated workflow with a customer willing to pay. Serving cost, latency, quality and retention must be assessed together. The current synthesis uses the presentation's technical and strategic analysis, while leaving detailed financial figures in the dated initial-coverage report rather than presenting them as newly verified results.

The slide contrasts a written machine explanation with a visual demonstration. A modality is an information form, such as text, image, audio or video. spatial relationships and timing can be easier to communicate visually. the same underlying concept is expressed through different output channels. choose the output that helps the user perform the task; expressive richness is not proof of factual accuracy. [27]



Picture 65 | Modalities and bandwidth of expression. Author's conference photograph. [27]

Illustration: an espresso-machine cleaning lesson can combine a close-up of the correct valve, narration and step timing. A generated video must still be checked against the actual machine and approved procedure. The slide's bandwidth metaphor concerns communication, not measured network speed. [27]

An LLM agent branches into model calling and code execution. a language model does not itself render every image, sound or game. it plans, sends requests to specialized generators, or executes HTML, Blender or game-engine code, checks results and iterates. The arrows converge on multimedia outputs. the whole workflow's reliability matters more than the coordinator's fluency. [27]

Illustration: a restaurant launch campaign requires copy, artwork, a video and a website. Different tools can disagree about colors, character appearance or tone. Review and regeneration increase the effective cost per accepted campaign. The slide demonstrates a system architecture, not measured production savings. [27]

Five handoff problems are contrasted with joint representations, streaming transfer, shared context, cross-modal learning and joint optimization. separate stages may discard tone, introduce waits, reinterpret intent, isolate knowledge and propagate errors. jointly trained representations can preserve more relationships between modalities. end-to-end is a design choice, not a guarantee that every deployment is faster or cheaper. [27]

Across these cases, inference becomes an input to a larger process of improvement. More attempts are valuable when their feedback is trustworthy and their cost is justified by better accepted results. For business applications, that shifts attention from the volume of generated material to the quality of the evaluator and the usefulness of what survives it.

7 Where the Economics Work

A technical improvement changes the opportunity to earn money, but does not determine who keeps it. Compute can remain heavily used while margins compress. A serving platform can simplify deployment while absorbing substantial operating work. The cases in this chapter ask where capacity, coordination and differentiated service may retain value—and which assumptions would reverse the conclusion.

7.1 The bottleneck can move—and so can the margin

The SF Compute presentation is useful because it treats value capture as conditional. Its scenarios ask how model capability, compute supply and demand evolve, rather than assuming that every layer benefits equally from AI growth. [28]

A cloud can sell more GPU hours while earning a lower margin per hour. A model provider can serve more tokens while reducing prices. An application can become more valuable to users even as its infrastructure suppliers face competition. Volume growth, revenue growth and profit growth are related through prices and costs; they are not the same outcome.

For an inference provider, a simplified revenue model is paid usage multiplied by realized price. Costs include hardware or capacity commitments, power, networking, software operations, support and idle time. The exact accounting varies by business. The key point is that high technical utilization and high paid utilization are different: a machine can be busy doing retries, internal tests or discounted work that does not support the intended economics.

For a buyer, the analogous measure is cost per useful result at the required quality and deadline. A provider may be attractive even with a higher nominal unit price if it reduces failures, engineering work or time to deployment. Conversely, a fast demonstration can be expensive to reproduce at the buyer's scale.

Durable value capture depends on which constraint remains scarce and who can solve it economically. That constraint may be chip supply, power-ready capacity, model capability, reliable serving, customer access or verified workflow integration. The conference collection supports examining these alternatives; it does not prove a single permanent winner.

7.2 SF Compute: value capture depends on the scenario

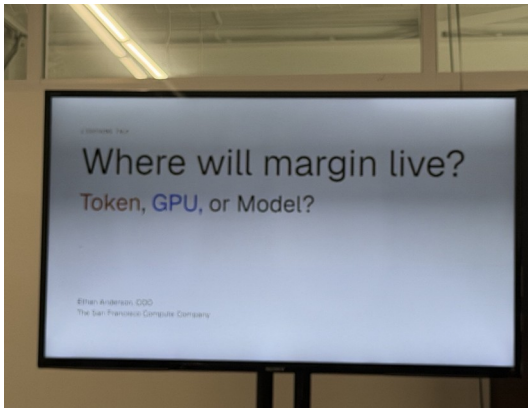
Ethan Anderson's presentation organizes industry economics around alternative assumptions about compute, models and demand. Its contribution is a conditional framework: different combinations can change where scarcity and margins remain.

Some photographed cells in the scenario matrix are obscured. I retain the framework's conditional logic without reconstructing missing text or treating each visible cell as a forecast. The useful question is how the answer changes when assumptions about demand, models and compute change. [28]

A company can retain a business while losing software-like margins. More demand does not guarantee that every supplier captures more profit. The relevant analysis separates revenue growth, pricing power, cost structure, differentiation and capital intensity. This framework is particularly useful for connecting conference technology to investment and infrastructure research.



Picture 66 | Ethan Anderson on value capture. Author's conference photograph. [28]



Picture 67 | Where will margin live — the value-capture matrix. Author's conference photograph. [28]

Three layers may capture value: GPU capacity, models, and token-serving services. A token is a unit of text, not cryptocurrency in this talk. [28]

A fast-growing industry does not guarantee attractive profits for every participant. Revenue can flow through a company to its suppliers. [28]

Ask which capability customers cannot easily replace and how much it costs to provide. The title introduces a question rather than choosing a winner. [28]

Chip type, cluster size, location, payment terms, and availability all affect prices. One index may conceal these differences. [28]

A long lease at a low hourly equivalent is not economically identical to flexible hourly access. Likewise, 1, 2, 4, 8 grows differently from 1, 2, 3, 4. Neither comparison proves future prices. [28]

Current prices describe present conditions. Long-run margin predictions need explicit assumptions about supply, demand, and competitive advantage. [28]

Illustration: demand rises from 100 to 150 while capacity rises from 100 to 120. Both grow, but scarcity increases. Reverse those growth rates and supply becomes relatively abundant. [28]

A company can correctly predict AI growth yet build the wrong quantity, generation, or location of infrastructure. The axis compares growth rates, not simply growth versus decline. [28]

Axis 2 asks whether open source takes a larger or smaller share of total AI value. [28]

Good-enough accessible models may perform ordinary tasks economically. Exclusive frontier abilities may unlock exceptionally valuable tasks. [28]

Illustration: open models create $5 of a $10 market, then $20 of a $100 market. Absolute contribution quadruples, but share falls from 50% to 20%. [28]

7.3 Hyperbolic: match committed capacity to useful work

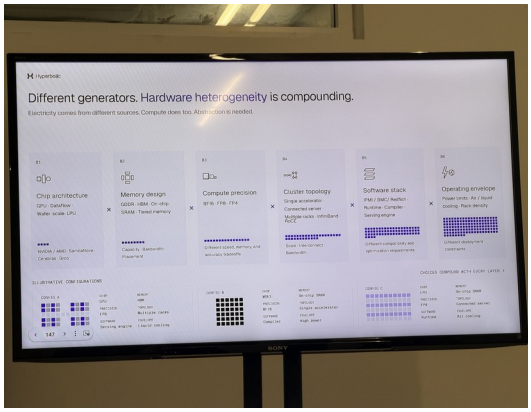
Jasper Yue Zhang's compute-grid presentation extends the discussion from a single cluster to coordination among providers. The proposed market connects customers needing flexible compute with suppliers holding capacity that may otherwise be underused.

The challenge is heterogeneity. Architecture, memory, precision, network, software and geography affect whether capacity can serve a workload. A marketplace must help match those characteristics rather than treat all accelerator-hours as identical commodities.

Better coordination may reduce some idle capacity and improve customer flexibility. It cannot guarantee resale demand, preserve every service target or eliminate contractual restrictions. The presentation's capacity-resale examples should be understood as economic scenarios. They become a business result only when compatible buyers transact at a price that covers the relevant costs.



Picture 68 | Jasper Yue Zhang on the compute grid. Author's conference photograph. [29]



Picture 69 | The compute-grid architecture. Author's conference photograph. [29]

Six dimensions differ: chip architecture, memory, numerical precision, network topology, software stack, and power or cooling limits. [29]

Match each workload with capabilities. Memory capacity controls what fits; bandwidth controls data movement. Lower precision can save resources but needs accuracy checks. Network topology controls communication. [29]

A request for eight compatible GPUs for six hours requires resource discovery, software checks, access, monitoring and billing. This is a functional explanation, not proof all features are deployed. [29]

Resell quiet periods or unused commitments while protecting the next scheduled job. Upgrades may pair resale of older hardware commitments with rental of newer capacity. Tenant isolation, permissions and metering are essential. [29]

Buyers must exist at the right time. Correlated peaks, lower old hardware prices and contractual restrictions can limit recovery. [29]

The slide's illustrative reservation is 1,000 B300 GPUs for seven days at $4.50 per committed GPU-hour. That is 1,000 × 168 = 168,000 GPU-hours and an original cost of $756,000. Assumed resale recovery of $268,800 reduces the net cost to $487,200, or $2.90 per committed GPU-hour—a 35.6% recovery of the original bill. These are scenario assumptions, not a quoted offer or realized customer saving. [29]

The denominator is committed hours, not own use hours. Actual savings depend on sale prices, fees, timing and demand; they are not guaranteed. [29]

In his closing remarks, Jasper Zhang proposed optimization across clouds: a customer specifies the workload, budget and deadline, and the platform chooses suitable resources. He described inference as an established activity and sandboxing and training services as gradually rolling out at the time of the talk. These are statements about the platform's direction and reported status, not an independent test of availability. [29]

For example, processing 10,000 images overnight requires checking compatible software, memory, expected speed, data transfer and total cost. A looser deadline may permit cheaper machines. This is an explanatory example rather than a tested product offer. [29]

Customer savings, supplier revenue and utilization can conflict. Higher prices benefit sellers but may hurt buyers. Relocation can improve hardware availability while increasing data transfer cost. Optimization therefore requires constraints and tradeoffs rather than promising to maximize every objective at once. [29]

7.4 SkyPilot: coordination determines productive use



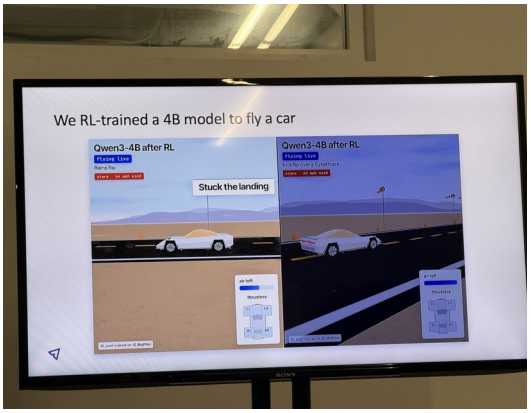
Picture 70 | Zhanghao Wu presenting SkyPilot. Author's conference photograph. [30]

Zhanghao Wu's SkyPilot session describes reinforcement learning as a coordination problem across different kinds of work. Rollouts, training and supporting services need resources at the right times. Some jobs require a group of resources to start together, making partial allocation wasteful.

Gang scheduling, placement and resource orchestration can improve the use of available capacity. The presentation's utilization figures are percentages, not multiplicative speedups. More productive scheduling does not establish improved model quality by itself. [30]

The software layer between a workload and available capacity can have significant economic value. Its benefit should be measured as useful job completion, reduced idle time and operational simplicity, with the engineering and service costs included.

Qwen3-4B controls a simulated vehicle with four thrusters. Four billion parameters are learned numerical settings inside the model, not four billion rules written by an engineer. [30]



Picture 71 | A simulated control workload in SkyPilot. Author's conference photograph. [30]

Lift, rotation, wind and landing must be coordinated. More lift is not always better: the vehicle must also return safely to the landing area. [30]

The control interval is 0.2 seconds, so 1 ÷ 0.2 = 5 decisions per second. Uneven forces can rotate the vehicle, like pushing under one edge of a book rather than lifting both sides evenly. The simulator determines the actual physics. [30]

An action must work now and leave a state from which the next action can also succeed. [30]

The chart separates decision quality from response latency. A paused simulator asks whether an action is good; a moving simulator asks whether it is good and timely. Latency is time for one response, while throughput is work completed per unit of time. [30]

A model can serve many requests collectively and still be too slow for one real-time controller. A late answer may respond to a state that no longer exists. [30]

Company-reported storm-kickflip success is 0% untrained, 84% after imitation, and 90% after imitation plus RL. The trained response is about 0.09 seconds. The 84-to-90 change is six percentage points. The selected training run and simulator conditions limit generalization. [30]

Targeted post-training can be valuable. These results do not establish that RL alone always wins. The displayed $13 training figure is not the total cost of research or original model pretraining. [30]

Inference generates actions; an environment executes them; observations and rewards record the outcome; the trainer changes model weights. The updated weights return to inference. [30]

7.5 Capacity, utilization and contribution are different measures

Hyperbolic's compute-grid vision concerns matching heterogeneous capacity with demand. Hardware differs in memory, interconnect, software support, location and power characteristics. A GPU hour is therefore not perfectly interchangeable with every other GPU hour. The customer buys a usable configuration, not an abstract count. [29]

Unused committed capacity can sometimes be resold or reassigned, but only if there is compatible demand and the commercial and technical arrangements permit it. A resale example is a scenario, not guaranteed revenue. Availability at the wrong time, in the wrong region or with the wrong network can be difficult to monetize.

SkyPilot's reinforcement-learning discussion focuses on coordinating a workload across resources. Some stages must start together; others can run independently. Gang scheduling means a group of required resources is allocated as a unit. Starting part of a tightly coupled job while the rest waits can waste capacity. [30]

These views connect finance and engineering. A long-term capacity commitment may reduce unit price but expose the buyer to idle time. Flexible capacity can reduce commitment risk but cost more or be less available. Orchestration software can improve usage, yet it does not create customers or remove every compatibility constraint.

The research question is therefore two-sided: how much productive work can the system schedule, and how much of that work is economically valuable or paid? Both should appear in a serious infrastructure business model. A utilization percentage without its definition is too ambiguous to support an investment conclusion.

7.6 Make the revenue assumptions visible

Consider a purely illustrative pool of 100 accelerators with 720 available hours in a thirty-day month. At 60% paid utilization, the pool sells 43,200 accelerator-hours. At $2 per hour, revenue would be $86,400. At 80% utilization, the same price produces $115,200. These are arithmetic scenarios, not market prices or a recommendation to buy hardware.

The sensitivity table shows why both price and paid utilization matter. A higher nominal rental price does little if demand is insufficient. A full machine at a deeply discounted price may not cover its total costs. The analysis must also distinguish committed capacity costs from variable expenses and include financing or depreciation as appropriate.

A useful investment model would add hardware acquisition or lease terms, useful life, residual value, power draw, electricity cost, cooling, network expense, staff, service credits and customer concentration. It would test ramp-up and downtime rather than assume immediate steady state. The example here isolates utilization and revenue; it is not an audited cost model for a specific facility.

For a restaurant buyer, these economics usually appear indirectly through provider pricing and availability. For GII's data-center research, they are central. The same inference workload that creates demand also determines the required service profile: latency-sensitive serving, batch analysis and RL rollouts can have different placement and scheduling needs.

The purpose of a sensitivity table is to expose assumptions. A responsible report should make it possible for a reader to replace them and see whether the conclusion survives.

Table 13 | Hypothetical monthly gross revenue: 100 accelerators × 720 hours

Paid utilization	$1.50/hour	$2.00/hour	$2.50/hour
40%	$43,200	$57,600	$72,000
60%	$64,800	$86,400	$108,000
80%	$86,400	$115,200	$144,000

7.7 Trust and reliability belong in the commercial offer

Phala’s confidential-inference work addresses protection of data while computation occurs. Attestation gives a client evidence about the environment and software identity under a stated trust model. That is valuable evidence about execution, while the correctness of the model’s answer remains a separate question. [15]

Palo Alto Networks addresses a different problem: running a model within a stringent latency target for a security service. The measured configuration matters. Increasing the number of concurrent clients that pass a target is not automatically a proportional increase in total throughput, and a service-wide request count is not a single-GPU benchmark. [21]

For customer intelligence, confidentiality may determine where data can be processed and which operators can access it. Accuracy determines whether extracted topics and forecasts are reliable. Safety and operating policy determine which actions are permitted. Each requirement needs its own controls and evidence.

This separation also improves commercial reporting. “Secure AI” is too broad to be informative without a threat model. “Accurate AI” needs a task and evaluation set. “Fast AI” needs a workload and a latency distribution. A vendor can make progress on one dimension while accepting a cost on another.

Trust is part of the delivered service. It can influence infrastructure choice, procurement review and the scope of applications a customer will deploy. It should be analyzed with the same specificity as memory, compute and price.

The operating implication is practical: identify the buyer, the task, the binding resource and the payment model together. A capacity seller, a software provider and an application operator can all benefit from growing inference demand while facing very different economics. A useful industry assessment makes those differences explicit.

8 Understanding 100,000 Profiles in a 300,000-Customer Business

My interest in inference ultimately returns to a business question. Suppose a restaurant company serves 300,000 customers but has 100,000 identifiable profiles in its database. How can it learn who those customers are, what they need, what future contribution they may generate and which contact would be useful? The numbers below are a hypothetical research design, not a description of a dataset already analyzed.

8.1 Reconcile the customer population

Consider a hypothetical restaurant company that estimates 300,000 unique customers over a defined twelve-month period. Its first-party database contains 100,000 identifiable customer profiles. These figures define the illustrative scenario used throughout this chapter; they are not observed company data.

The first task is to determine whether the two populations are comparable. A “customer” might mean a person, an account, a household, a payment token or a delivery-platform user. Duplicate accounts can inflate the database. Shared payment methods can combine several people. Visitors who never identify themselves may appear only in transaction totals. A precise analysis starts with definitions and reconciliation.

Even if all 100,000 profiles are unique, they are not automatically all reachable. Some have no valid contact channel; some have opted out; some records are stale; some contacts may be usable only for a specific purpose. Nor are those profiles automatically representative of the other 200,000 customers. Loyalty members can differ systematically from occasional visitors.

The objective is therefore not to infer invented individual profiles for everyone. It is to understand what is known, measure what can be estimated, identify uncertainty and improve decisions within those limits. Aggregate store patterns can inform planning for anonymous demand without pretending the company knows each person's identity.

The managerial questions are concrete. Which needs are recurring? Which customers are likely to return? Which relationships generate contribution after costs? Which customers might change their behavior because of an intervention? Which channel and timing are appropriate? Each question requires a different evidence path.

Table 14 | Population reconciliation

Population	Illustrative count	Interpretation
Total unique customers in period	300,000	Definition and estimation method required
Identifiable database profiles	100,000	Must be deduplicated and reconciled
Customers outside identified profiles	200,000	Only if populations are truly comparable
Reachable and eligible profiles	Unknown	Cannot infer from database size
Experiment-eligible profiles	Unknown	Depends on channel and design

8.2 Build a customer evidence table

A useful customer table is an organized record of evidence. It can include a stable internal identifier, observed visit dates, spending, location, channel, products, returns, offers redeemed, consent status and feedback. Each field should retain its source and refresh date. The analysis should be able to trace a conclusion back to the records that support it.

Identity resolution comes first. Deterministic matches, such as an authenticated account identifier, are usually easier to defend than probabilistic guesses. Where uncertain matching is necessary, preserve the confidence and the original records. Do not merge two people merely because they share an address or surname. A wrong merge can contaminate both predictions and communication.

A transaction table and a profile table serve different purposes. Transactions record events; profiles summarize the relationship. Aggregated features might include days since the last visit, number of visits in a trailing period, average contribution, daypart mix and channel mix. Those features need time windows. A frequency number without a period cannot be interpreted.

Missing values need meanings. “No recorded dietary preference” is different from “no dietary preference.” “No email” is different from “does not want email.” A language model should not fill such gaps with plausible personal attributes. For customer communication, uncertain inferences should be treated as hypotheses, with a route for correction.

The project can begin with a modest data model and a reproducible refresh process. The highest-value early result may be learning how many of the 100,000 profiles are actually usable for analysis and outreach. That is a business finding, even before any sophisticated model runs.

Table 15 | Minimum evidence schema

Field group	Example	Control
Identity	Internal customer ID	Deduplication and match provenance
Behavior	Visit dates and items	Defined observation window
Economics	Contribution after variable costs	Reconcile to accounting
Channel	Email / app / SMS eligibility	Current preference record
Text evidence	Complaint or review excerpt	Source and confidence
Model output	Predicted next-period contribution	Version and scoring date

8.3 Understand needs before choosing an algorithm

Observed behavior can suggest a need without proving it. A customer who visits mostly at weekday lunch may value speed or location convenience, but the transaction pattern alone does not tell us which. Written feedback, a short survey or a well-designed experiment can provide additional evidence.

A useful analysis combines several views. Geographic patterns identify where demand occurs. Daypart and occasion patterns describe when. Product and channel patterns describe what people select and how they transact. Text feedback can explain some reasons. The combined result is richer than a single demographic label, especially when demographic data has not actually been collected.

An LLM can help classify feedback into a controlled set of topics such as waiting time, order accuracy, portion consistency, value or hospitality. The output should include a source excerpt and an uncertainty flag. If the text does not support a topic, the model should return “insufficient evidence” rather than inventing a reason.

Population summaries should report denominators. If 12% of analyzed complaints mention waiting time, that does not mean 12% of all customers experienced excessive waits. People who leave feedback are a selected population. The report should say how many texts were analyzed, how they were sampled and whether multiple comments from one person were counted separately.

The strongest first use of language-model inference may be converting scattered qualitative material into a consistent, auditable dataset. It supplements structured analysis. It does not replace the transaction ledger or turn unobserved customer characteristics into facts. This is a practical connection between inference infrastructure and customer understanding.

8.4 Estimate customer value with an explicit horizon

Historical customer contribution measures what a relationship has already generated under a stated accounting definition. Predicted customer lifetime value estimates future contribution. The two should be displayed separately, because a formerly valuable customer may be unlikely to return, while a new relationship may have little history but substantial potential.

A manageable starting point is twelve-month expected contribution rather than an unbounded “lifetime” estimate. For each customer, estimate expected visits or purchases during the next twelve months, expected contribution per visit and relevant relationship costs. More advanced models can account for changing purchase behavior, uncertainty and discounting. A clear baseline is preferable to a sophisticated number whose definition cannot be explained.

Suppose a hypothetical customer is expected to make eight visits in the next year, with $9 contribution per visit before customer-specific marketing expense. Expected contribution is $72. If planned customer-specific communication costs $4, the resulting estimate is $68 before any additional adjustments. This is a teaching example; it is not a prediction for an actual customer population.

The contribution measure should reflect the decision. Discounts, channel commissions, refunds and other variable costs can change the value of an incremental visit. Allocating every fixed cost to a customer may obscure a marginal campaign decision, while ignoring fixed infrastructure costs would be wrong for a full investment appraisal. The report should state which question the metric answers.

A customer-value model should also produce uncertainty or at least distinguish stable estimates from thin-data estimates. A profile with one visit should not receive the same confidence as a profile with years of consistent observations. The aim is better prioritization and learning, not a false claim of precision.

Table 16 | Illustrative twelve-month contribution

Input	Assumption	Result
Expected visits	8	Forecast, not a guarantee
Contribution per visit	$9	$72 expected visit contribution
Relationship cost	$4	$68 net expected contribution
Uncertainty	Not estimated in this toy example	Required for an operational model

8.5 Learn from the past without leaking the future

To predict the next twelve months, construct historical examples using information available at an earlier date. Features might summarize the prior six or twelve months. The target then records the customer's actual outcome during the following period. This creates a supervised-learning dataset: prior observations paired with later outcomes.

A simple baseline could use recency, frequency and contribution groups. A more flexible model could predict return probability, expected purchase frequency or future contribution directly. Model choice should follow the target, data volume, missingness and need for explanation. A language model is not automatically the best numerical predictor for a structured transaction table.

Time separation is essential. If a feature accidentally includes a visit from the outcome period, the model is learning from the answer. Its backtest may look excellent and fail in production. Split training and evaluation chronologically, and evaluate on a period not used to choose features or tune the model. Repeat this process across several historical cutoffs if the data permits.

Measure calibration as well as ranking. If customers assigned a 30% return probability return roughly 30% of the time in a comparable evaluation set, the forecast is calibrated. A model can rank people reasonably well while systematically overstating their probabilities. That matters when a budget depends on expected contribution.

Finally, compare against a simple operating policy. If the model cannot improve decisions over a recency-frequency baseline after implementation costs, complexity has not yet earned its place. The initial deployment can run in shadow mode, producing scores without changing customer treatment, while the team checks data quality and stability.

Table 17 | A practical model development sequence

Step	Deliverable
Define target	Next-90-day return or next-year contribution
Freeze observation date	Only information available at that date
Build baseline	Recency / frequency / contribution rules
Train and validate	Chronological holdout with leakage checks
Evaluate business usefulness	Calibration, ranking and decision cost
Deploy cautiously	Versioned scores and ongoing monitoring

8.6 Where language-model inference adds value

Language models can contribute at several points: extracting topics from feedback, standardizing messy text, retrieving relevant operating information, explaining a model's inputs in plain language and drafting communications for review. Each role should have a bounded output and a testable quality standard.

For feedback extraction, request a structured record with fields such as topic, sentiment, supporting phrase and confidence. Provide an allowed category list and an abstention option. Evaluate the outputs against a human-labeled sample, including short, ambiguous and multilingual comments. Accuracy on easy examples should not conceal systematic errors in difficult groups.

For analytical explanation, the LLM should receive validated figures and definitions, not be asked to invent a calculation. A deterministic query or model computes the score; the language model describes the result and its limitations. If the evidence says a customer's visits declined, the explanation should not assert that the customer moved away or disliked the food without supporting information.

For outreach, use a controlled set of approved offers and claims. The system can vary tone or summarize a benefit, but eligibility, contact permissions, budget and offer rules should remain explicit controls. Human review may be appropriate for sensitive or unusual cases. Monitoring should include complaint and unsubscribe behavior, not just clicks.

This design creates a clear connection to the summit's infrastructure themes. Many short classification tasks may benefit from batch serving. Interactive staff assistance may need low latency. Shared instructions may benefit from caching. A

customer-data boundary may require confidential or locally controlled deployment. Architecture follows the actual use case rather than a generic preference for the largest model. [12, 15, 1]

8.7 Reach customers without confusing response and impact

A high response probability does not mean a promotion creates additional value. Some loyal customers would have visited without a discount. Giving them an offer may reduce contribution while producing an attractive redemption rate. The relevant causal question is how behavior changes because of the intervention.

A randomized holdout creates a useful starting point. Among eligible customers, randomly assign comparable groups to receive an intervention or continue under the existing policy. Define the evaluation window and outcome before launching. Measure incremental contribution, not only visits or gross sales. Include discounts, communication costs and relevant variable costs.

Suppose 20,000 eligible customers are randomly assigned: 10,000 receive an offer and 10,000 form a control group. During the same observation window, 600 treatment customers and 500 control customers each make one purchase; assume no repeat purchases in this teaching example. The observed customer purchase rates are 6% and 5%, a difference of one percentage point, or 100 purchases across equal-sized groups. This is the experiment’s point estimate; uncertainty still needs assessment. Crediting all 600 treatment purchases to the offer would overstate its effect.

An uplift model attempts to estimate differences in treatment effect across customers. It requires suitable experimental or otherwise defensible causal data. A standard purchase-probability model cannot be relabeled an uplift model simply because its scores are used for targeting.

Over time, the company can learn which needs, channels and offers produce incremental contribution for which groups. The learning objective should preserve a good customer relationship: repeated messages, discount dependence and attrition can erase a short-term gain. This is where customer value becomes a decision system rather than a single score.

Table 18 | Illustrative campaign comparison

Metric	Treatment	Control
Eligible customers	10,000	10,000
Customers making one purchase	600	500
Customer purchase rate	6%	5%

8.8 Calculate the campaign economics

Continue the hypothetical experiment. Assume each purchase contributes $12 before the promotional discount, every treatment purchase redeems a $2 discount, and the control group receives none. Treatment contribution is 600 × ($12 − $2) = $6,000; control contribution is 500 × $12 = $6,000. Before communication costs, the observed increase in purchases adds no contribution under these simplified assumptions.

If contacting the 10,000 treatment customers costs $500, the campaign's observed net contribution difference is negative $500. This example shows why a campaign with a higher purchase rate can still destroy value. The discount is paid to customers who would have purchased anyway as well as to the incremental customers.

The calculation deliberately omits longer-term effects, basket changes, capacity constraints and statistical uncertainty. A real experiment should include those effects when material. For example, a campaign might introduce a customer to a new habit that improves later contribution; it might also shift a visit forward from next week rather than create a new visit. The evaluation window and follow-up period matter.

The lesson for inference is that better prediction is useful only when the decision rule is economically sound. An accurate model that identifies likely buyers may intensify the wrong promotion. A less complex model paired with a well-designed experiment can produce more valuable learning.

For GII Research, this provides the bridge from GPU efficiency to restaurant outcomes. Infrastructure determines how cheaply and quickly analyses can run. Experimental measurement determines whether the resulting actions create value. The report should never present one as proof of the other.

Table 19 | Illustrative contribution bridge

Calculation	Treatment	Control
Purchases	600	500
Contribution before discount	$7,200	$6,000
Discount cost	$1,200	$0
Communication cost	$500	$0
Net contribution	$5,500	$6,000

8.9 Match the serving design to the decision

Not every customer-analysis task needs a subsecond response. A nightly profile-scoring job can optimize for cost and completion by a morning deadline. A staff member asking for an explanation during a customer interaction needs a different latency target. A monthly research report can tolerate more verification and longer computation than a live recommendation.

The 100,000-profile database should not imply 100,000 long generative prompts. Structured models can score rows efficiently. Text extraction should process the relevant new or changed records. Retrieval should provide only the evidence required for a question. Reusing stable instructions or validated intermediate results can reduce repeated work, subject to correctness and access controls.

A sensible first deployment may use a managed service, because the company is purchasing useful analysis rather than a GPU operations function. Self-hosting becomes worth evaluating when workload scale, data requirements, predictability or specialized capabilities justify the operating responsibility. Neither choice is universally cheaper.

Measure complete costs: data preparation, model calls, engineering, monitoring, review, failed jobs and downstream rework. Token charges are one input. A low price per token can be offset by unnecessarily long prompts, repeated retries or low-quality outputs that require manual correction.

The deployment decision should be revisited after observing real workloads. The summit's PD, caching, memory and orchestration lessons become actionable when the team knows prompt distributions, output lengths, concurrency, quality thresholds and deadlines. That measured workload is the link between a restaurant question and an infrastructure purchase.

Table 20 | Workload-to-architecture mapping

Workload	Primary target	Likely starting design
Nightly numerical scoring	Complete by deadline	Structured batch pipeline
Feedback classification	Accuracy and cost per accepted record	Batched LLM extraction
Manager research question	Grounded answer with citations	Retrieval plus validated tables
Live staff assistance	Latency and safe escalation	Interactive service with monitoring
Campaign learning	Causal contribution estimate	Experiment and analytical pipeline

8.10 A ninety-day research sequence

Days 1–30 should establish definitions, data access and a baseline. Reconcile the 300,000-customer estimate with the 100,000-profile database; identify duplicates and reachable customers; define contribution; document consent and channel status; and select one business question. A useful first question might be whether a service-recovery message improves return behavior after a documented complaint.

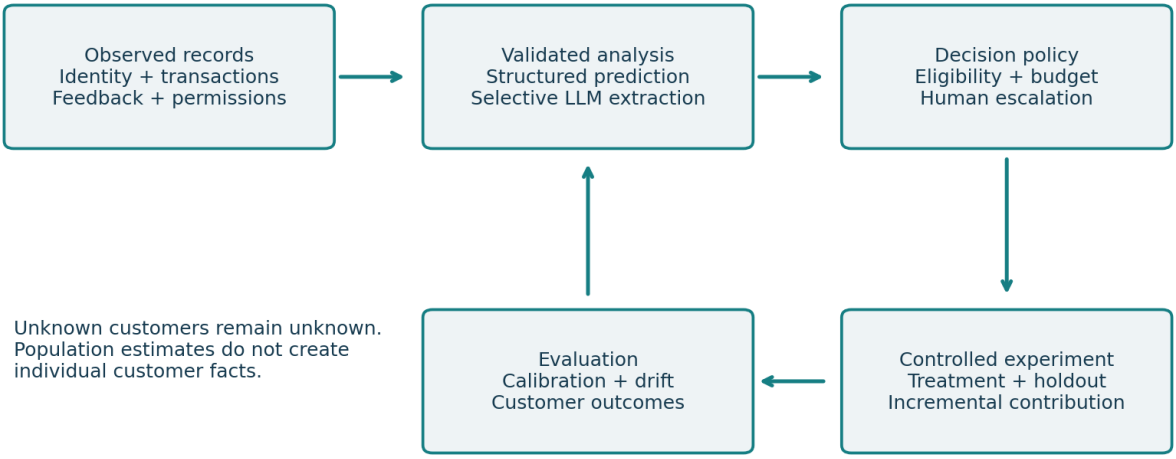
Days 31–60 should build a reproducible analytical pipeline. Create time-defined features, implement a simple prediction baseline, evaluate text classification against labeled examples and produce a scorecard with uncertainty. Run the pipeline in shadow mode. Staff should be able to explain where each number came from and identify cases where the system should abstain.

Days 61–90 should test one controlled intervention with a holdout. Predefine the target population, contact rule, offer, outcome and observation window. Review operational readiness: can stores honor the offer, can staff handle questions, and can the business record outcomes consistently? A technically elegant campaign can fail because the operating process is incomplete.

The deliverable is not merely a model. It is a reliable evidence path from records to a decision, with a measured result and a documented next question. If the first experiment shows no gain, the project can still succeed as learning if it prevents a costly broad rollout and identifies a better hypothesis.

This is my proposed application framework, informed by the conference’s system-level lessons. It is not a restaurant deployment claimed by a summit speaker. A real pilot would need the organization’s data, staffing capacity, decision calendar and operating constraints before budgets or performance targets could be set.

8.11 The complete customer-intelligence loop



Picture 72 | The restaurant customer-intelligence operating loop. GII explanatory diagram; assumptions are stated in the text.

The loop begins with observed records and ends with measured outcomes. The prediction stage estimates what may happen. The policy stage decides which eligible action to take. The experiment measures whether that action changed contribution or customer behavior.

Feedback returns to the analytical stage, but a model update should not be automatic. The team must check whether a change in results reflects customer behavior, data quality, a new offer, operational disruption or a genuine improvement in the model. Versioned records make that investigation possible.

The population boundary remains visible throughout: 100,000 identifiable profiles do not create individual knowledge about the remaining customers. Better analysis can improve population estimates and guide further data collection without pretending that uncertainty has disappeared.

9 Implications for Restaurants, Robotics and Data Centers

The conference broadens the question raised by Episode 1. A restaurant's physical operation, a robot's action loop and a data center's serving capacity each depend on a complete system working under local constraints. The useful comparison is how capability becomes repeatable work—and what evidence would justify expanding the commitment.

9.1 What changes for a restaurant operator

The first practical move is to define a customer decision that can be measured. A restaurant can begin with an identifiable cohort, a clean outcome window and a modest test of whether a prediction or message improves contribution. It can then choose the simplest analytical method that answers the question and add language-model capabilities where they produce useful, checkable work.

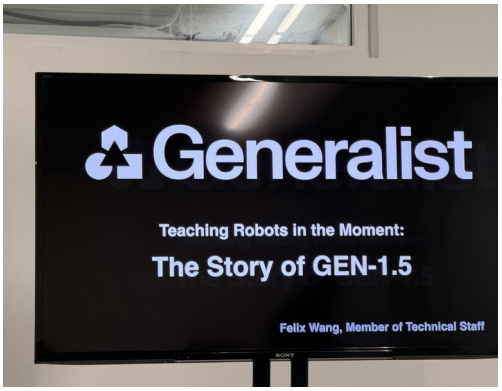
This is the connection to Episode 1. The site bears the operating consequences; the task defines the learning and acceptance test; the network determines whether a successful method transfers. A pilot that improves a local result may still need changes in data capture, staff workflow and permissions before it can be reproduced across a chain.

9.2 Generalist: inference meets a physical action

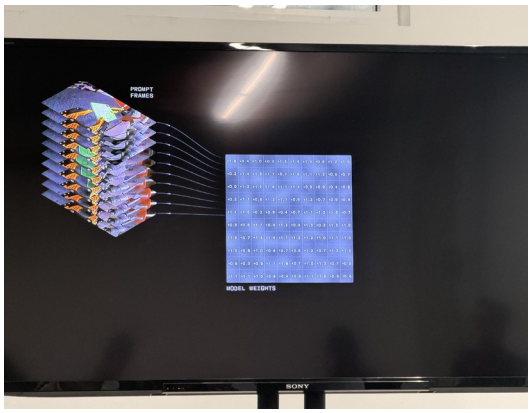
Felix Wang's Generalist session describes physical prompting: short demonstrations enter the model's context and influence robot behavior without a weight update. The mechanism is distinct from fine-tuning a policy for every new task.

The presentation combines experiment descriptions, demonstration frames and an analogy with language-model development. A temporary context can change the robot's next action while underlying parameters remain fixed. That distinction is central to understanding the demonstration. [10]

Rapid adaptation could make robots more flexible, but a short demonstration does not establish reliability across a restaurant, warehouse or retail environment. Task distribution, recovery behavior, hardware durability and safe operating boundaries still require measurement. The "GPT-3 moment" framing remains a speaker analogy, not an independently established adoption timetable.



Picture 73 | Generalist GEN-1.5 session title. Author's conference photograph. [10]



Picture 74 | Physical prompting and learned model weights. Author's conference photograph. [10]

A stack of prompt frames connects to a block labeled model weights. The numbers are an explanatory drawing, not a published network specification. [10]

The distinction between input and stored parameters is essential. A robot can change what it does because its input changed, even when its weights remain fixed. [10]

Think of weights as practiced skills and prompt frames as the example currently on the desk. Context temporarily guides those skills. [10]

Physical prompting changes the information supplied to the model. Fine-tuning changes the model itself. These are distinct adaptation mechanisms. [10]

Prompt frames and current input frames share a context window. Model outputs feed robot trajectories. The earlier animation frame has an empty trajectory panel; the later frame shows paths. [10]

The robot must relate the example to the present scene. Copying exact coordinates would fail when an object moves. [10]

The model conditions its action prediction on the example and present observations. A trajectory is a sequence of positions or actions over time, which the physical system must execute. [10]

The diagram explains an information flow, not every software component. It does not specify model size, attention layout, control-loop timing, or safety mechanisms. [10]

The question is whether a robot can adapt using less than a minute of demonstration and zero gradient updates. This is labeled in-context learning or physical prompting. [10]

Avoiding per-task optimization could shorten setup and allow behavior to change as soon as a new example is supplied. [10]

Place the example into model memory rather than using it to rewrite weights. The pretrained model interprets the context during inference. [10]

Zero updates does not mean zero prior training, zero computation, or zero task experience in the training corpus. It refers to adaptation at this stage. [10]

9.3 Data-center capacity must become a defined service

The hardware photographs make inference tangible: dense systems require power delivery, heat removal, data movement and service access. Facility and network design determine whether the processors can operate as intended. [7, 6]

Installed megawatts and accelerators measure physical capacity. Saleable inference capacity also depends on workload, software, memory, interconnect, service targets and availability. Facilities with similar electrical capacity can therefore support different commercial profiles.

Procurement requires a dated configuration covering compute, network, cooling, facility work, support and deployment. The equipment photographs in Chapter 4 show why those items belong in the same discussion. An illustrative purchasing budget is not a vendor quotation. [7, 6]

Uni³Sim models interactions among compute, memory, interconnect and scheduling. It can screen configurations before deployment, provided its assumptions and predictions are calibrated against measured systems. [20]

For GII, the underwriting chain runs from physical capacity to workload-level service and then to revenue or internal value. It connects site selection, engineering and commercial diligence.

9.4 The questions I would take into the next deployment

Begin with the customer's task: what output, at what quality, by what deadline, for how many users? Those requirements guide model choice, serving layout, hardware and cost.

For performance claims, request the model version, precision, input and output distributions, concurrency, cache policy, hardware count and service target. Record failures and timeouts; distinguish averages, percentiles and best runs.

For commercial claims, trace the improvement to less engineering work, more useful requests, stronger retention or a previously impractical application. The buyer needs that connection to value.

For adoption claims, separate demonstrations, trials, paid deployments and scaled recurring use. Record funding as financing, participation logos as participation, and roadmaps as future intent.

For learning systems, identify the trusted feedback and evaluation outside the training loop. When generated outputs become future training material, more computation can amplify both useful search and systematic mistakes.

These questions make technical reporting accessible while preserving the details an expert needs to challenge its conclusions.

9.5 The next research agenda

Inference is a systems and operating-economics question. Model capability depends on scheduled execution, correctly managed state, reliable tools and evaluation that recognizes useful output.

The prefill/decode example and the customer campaign share a lesson: retain the denominator. Faster individual streaming can coexist with unchanged throughput; a higher purchase rate can coexist with lower contribution. The headline improvement and the business outcome must both be measured.

The hardware and platform cases explain why major suppliers participate. Shared software can expand adoption for competitors, while integration, support and operating performance remain sources of differentiation.

Axiom Math, Krea, Apodex, Miles and Generalist extend inference into search, verification, learning, creation and action. Each needs an appropriate acceptance test: a formal verifier, an independent evaluation, a controlled experiment or a physical success criterion.

My next GII research priority is to connect these mechanisms to operating evidence: customer-value analysis with an actual dataset, workload-based infrastructure economics and physical-AI deployments measured in real environments. The question that began at the restaurant site remains the right one at the server rack: does the complete system deliver useful work at an acceptable cost?

I came to the summit with an operator's interest in what AI could make possible. I leave with a more precise research question: what must be measured before a promising capability earns a place in a working business? The path runs through customer evidence, model behavior, system engineering and operating economics. Following that path is how GII can connect the restaurant floor to the inference economy.

Appendix A Working Glossary

Working glossary

Table 21 | Glossary A Serving and hardware

Term	Meaning
Inference	Applying a trained model to new inputs.
Prefill	Processing the input sequence and preparing state for generation.
Decode	Generating additional output tokens using the available context.
KV cache	Stored key/value state used by attention; not a database of finished answers.
PD disaggregation	Placing prefill and decode work in separate serving resources.
TTFT	Time to first output token, including relevant non-engine delays.
ITL	Inter-token latency, the interval between generated tokens.
Throughput	Work completed per unit of time across the measured system.
Goodput	Useful completed work satisfying explicitly stated quality and service targets.
Batch	A group of work items processed together; its meaning depends on the engine.
Concurrency	The number of requests or activities in progress at once.
SLO	A defined service-level objective; distinct from a contractual SLA.
HBM	High-bandwidth memory located close to an accelerator.
Memory tiering	Managing state across storage levels with different capacity and speed.
MoE	Mixture of experts; a model architecture that activates selected expert components.
Kernel	A device program implementing a computational operation.

Working glossary continued

Table 22 | Glossary B Learning trust and customer analysis

Term	Meaning
Rollout	An attempted task or trajectory generated for evaluation or learning.
Trajectory	A sequence of observations, actions and associated information.
Post-training	Training after a base model has been developed to change its behavior or capability.
Reward	A signal used to evaluate an attempt in a learning process.
Attestation	Evidence about an execution environment and its measured software state.
Formal verification	Checking stated properties within a formal system and its assumptions.
Calibration	Agreement between predicted probabilities and observed frequencies.
Customer contribution	Revenue less the costs included in a stated contribution definition.
Predicted CLV	An estimate of future customer contribution over a defined or modeled horizon.
Incrementality	The difference in outcome caused by an intervention relative to its counterfactual.
Uplift	A treatment-effect concept; not simply purchase or response probability.
Holdout	A comparison group or evaluation set kept separate for a defined purpose.
Data leakage	Use of information that would not be available at the real prediction time.
Drift	A change in data, relationships or performance over time.
Paid utilization	The share of defined capacity that generates paid usage; distinct from technical activity.

Appendix B Reporting Method and Evidence

This is an original field report by Dr. Cervantes Lee, based on photographs, session records and notes collected around SGLang Summit at Fort Mason, San Francisco, November 12–13, 2026. The Traction Layer gathering at Ghirardelli Square is identified separately as a companion event. Official agenda material and publicly issued company and technical documents support session identities, product specifications and selected company facts.

The report is self-contained. Its technical explanations, relevant cases and worked examples are included in the chapters. The author's underlying recordings, notes and production documents remain a private reporting archive. The numbered endnotes identify the session evidence and original public documents relevant to each subject; no unpublished working report is required reading.

Attribution determines what a statement establishes. A photograph documents what was visible. A presenter's result remains a presenter-reported result unless independently tested. Product documentation describes a reference configuration. The analysis draws conclusions from those materials without converting a sponsorship, funding round, demonstration or roadmap into evidence of commercial performance.

The technical and public-source record reflects materials collected for the November 2026 reporting period. This standalone edition reorganizes and edits that record; it does not claim a new round of external verification or replay of every original recording. Live documentation may change after publication. Where a specification, identity or measurement remains unresolved, the text keeps that limitation attached to the claim.

The restaurant populations, customer-value exercise, campaign comparison, end-to-end timing example and utilization sensitivities are illustrative GII calculations. No actual restaurant customer database was supplied or analyzed. Tables state their assumptions so that a reader can replace the teaching values with measured operating data.

Photographs retain their documentary content. Cropping, orientation adjustments and numbered annotations improve legibility; they do not reconstruct hidden equipment or missing slide text. Official portraits and original explanatory diagrams are labeled. The participation badge's QR credential is concealed. Some slides remain partially obscured because that is what the field photograph records.

These distinctions determine how much weight a reader can place on the evidence. They apply throughout the report.

Table 23 | Evidence distinctions

Item	Treatment in this edition
Intel leadership photograph	Identified as an on-site screen relay; no reconstructed speech quotation.
AMD official portrait	Identified as an agenda portrait rather than an author-taken stage image.
ApprenticeBench portrait	Session-title image used instead of making an additional portrait-identity assertion.
Traction Layer	Identified as a Ghirardelli Square companion event.
Uni³Sim	Treated as academic AI infrastructure simulation.
NVIDIA tray identification	Switch and compute trays remain distinct.
Axiom Putnam result	Eight within the contest window; twelve eventually, as reported by Axiom.
PD 75/25 example	Engine-time allocation, not a token ratio or universal hardware prescription.
Customer populations	Hypothetical 300,000 total and 100,000 database profiles; no real dataset analyzed.
Customer value examples	Illustrative arithmetic with explicit assumptions, not company forecasts.
Benchmarks	Presenter- or company-reported unless explicitly identified as GII teaching arithmetic.

Editorial and publication record

Standalone edition 2.0, November 2026. Reporting, photography and analysis by Dr. Cervantes Lee, Global Innovation Institute. This edition integrates the industry cases into the argument and replaces references to private working documents with session attribution and public-source notes. AI tools assisted with organization, drafting and document production. Corrections should identify the passage and supporting evidence; later developments should be dated separately from the November 12–13 field record.

Endnotes and Public References

Numbers in brackets refer to the records below. Conference claims are supported by the author's field photographs and session notes; selected original public documents support identities, specifications and background. A public documentation link supplies context and does not independently verify a presenter's benchmark. The official summit agenda provides the common event reference.

SGLang Summit 2026 — official agenda and speaker roster

1. Horace He: Two Mental Models About Inference

SGLang Summit, November 13, 2026. Author's photographs and speech record. The 75% prefill / 25% decode example concerns engine-processing time, not token shares. Its fourfold per-user streaming result assumes balanced pools and compatible decode batches; total output remains 160 tokens per second in the explanatory operating point. The formula and arithmetic in Chapter 3 are teaching interpretations, not independently reproduced benchmarks.

SGLang PD disaggregation

vLLM disaggregated prefilling

Modular inference handbook — PD disaggregation

Google SRE — service-level objectives

2. GII research sequence and original analytical work

Dr. Cervantes Lee, The Restaurant as a Physical AI Platform, Global Innovation Institute, October 2026. Episode 1 supplies the site / task / network framework and this volume's visual identity. The framework is explained in Chapter 1, so reading the earlier volume is optional. Restaurant populations, campaign calculations, revenue sensitivities and other examples explicitly labeled illustrative are original analytical exercises in this volume.

3. SGLang, Miles and the opening-session ecosystem

SGLang Summit opening material identifies Lianmin Zheng, Ying Sheng and Banghua Zhu. Official summit portraits are credited as such in the picture captions. Author's session record and original project documentation support the relationship between serving and reinforcement learning. Open-source projects, commercial services and consortium governance are distinguished.

SGLang repository

Miles repository

RadixArk — Miles v0.1

LMSYS — SGLang Consortium

4. Intel: leadership presence and open inference commitments

Author's screen-relay and audience photographs, November 12, 2026, Fort Mason. They document Lip-Bu Tan's visible participation; a full transcript of his remarks is not available in this reporting record. Intel's public announcements supply the collaboration statements. Motives and commercial implications in the text are the author's interpretation.

Intel and RadixArk collaboration announcement

Intel — open infrastructure for agentic AI

LMSYS — SGLang Consortium

5. AMD participation and developer adoption

SGLang Summit, November 12–13, 2026. Author's exhibit photographs, official sponsor roster and agenda support this account. Vamsi Boppana is identified through official summit material; his portrait is not presented as an author-taken stage photograph. The agenda records leadership, engineering and workshop participation. No full recording of these AMD sessions was available for reconstructing remarks.

Official SGLang Summit agenda

AMD AI Developer Program

AMD SGLang documentation

SAPPHIRE Radeon AI PRO R9700

6. AMD exhibition hardware and reference specifications

Author's hardware photographs, November 12, 2026. The MI350X exhibit is an eight-module accelerator baseboard. The separate MI355X sign describes a liquid-cooled platform. The opened silver chassis has an unresolved CPU model and accelerator population. Product documentation supplies reference specifications; the photographs are not a running-system test or a complete inventory. Numbered overlays identify visible regions without reconstructing hidden hardware.

AMD Instinct MI350X platform

AMD Instinct MI355X platform

AMD ROCm MI350 architecture

AMD SGLang documentation

7. NVIDIA exhibition hardware and reference configurations

Author's hardware photographs at Fort Mason during SGLang Summit, November 12–13, 2026. Numbered overlays are explanatory annotations. Readable tray labels support GB300, Rubin and Vera identifications; the overall front-rack photograph does not conclusively establish its generation. CPU/GPU counts describe the cited reference configurations, not hidden packages independently inspected. No purchase quotation or running-system benchmark was obtained.

NVIDIA GB300 NVL72

NVIDIA DGX SuperPOD GB300 reference architecture

NVIDIA Vera Rubin NVL72

NVIDIA DGX Vera Rubin hardware guide

8. Adarsh Shirawalmath, Hugging Face: Beyond the Brrr

SGLang Summit, November 13, 2026, Decode Room. Author's session photographs, speech record and the speaker presentation support the discussion of kernel packaging and fallback behavior. The stated 7.5× comparison applies to its shown baseline. Kernel Hub documentation supplies the packaging and compatibility explanation.

Hugging Face kernels documentation

Hugging Face kernels repository

Kernel builder design overview

9. IBM Spyre: integration into SGLang

SGLang Summit, November 13, 2026. Author's photographs and session record support the integration account. The under-a-week claim concerns connecting existing hardware support to SGLang; it is not the development time for the chip or the entire compiler stack. The PyTorch article and implementation repository provide the original technical context.

PyTorch — Integrating SGLang into Spyre in Under a Week with AI

PyTorch — Building Spyre as a Native PyTorch Device

torch-spyre / sglang-spyre

10. Felix Wang, Generalist: GEN-1.5 and physical prompting

SGLang Summit, November 13, 2026, 1:00–1:15 p.m. Pacific, Decode Room. Author's photographs and speech record; speaker identified as a Generalist technical staff member. Demonstration results and the analogy to language-model development are presenter claims. Context conditioning and parameter updates are distinguished. No restaurant deployment is inferred from the robotics examples.

Generalist — GEN-1.5

Generalist — Beyond World Models

Generalist — GEN-1

11. Zhichen Zeng, RadixArk: production-level post-training with Miles

SGLang Summit, November 13, 2026, 11:10–11:25 a.m. Pacific, HiCache Room. Author's photographs and speech record. The technical discussion concerns rollouts, trajectory history, synchronization, consistency and recovery. Use of an application harness does not establish that Miles trains the proprietary model associated with that harness.

Miles documentation

Miles repository

Miles paper, arXiv:2609.08368

12. Lucas Atkins, Arcee AI: Fast Tokens, Slow Agents

SGLang Summit, November 13, 2026, Decode Room. Author's photographs and speech record; Atkins is identified as CTO and cofounder. The session distinguishes generation speed from validated task completion. The 60-second workflow and its 1.2× overall speedup are original explanatory arithmetic in this report, not an Arcee benchmark.

Arcee AI company background

Lucas Atkins speaker biography

13. Francis Zhang, Core Automation: inference systems for research

Designing Inference Systems for the Age of Research, SGLang Summit, November 13, 2026, 10:45–11:00 a.m. Pacific, HiCache Room. Author's photographs and speech record. The proposed cache interfaces and memory pools are an engineering argument; the presentation does not establish a measured universal speedup.

SGLang — unified radix cache

vLLM hybrid KV-cache manager

Transformers cache explanation

14. Tri Dao and Alexander Korte, Baseten: Ultrafast Inference on SGL

SGLang Summit, November 13, 2026, 10:30–10:45 a.m. Pacific, HiCache Room. Author's photographs and speech record. Dao is the Baseten engineer identified by the event and professional profile, distinct from the FlashAttention researcher of the same name. Microsecond layer results, milliseconds per decode step and user token rates describe different measurements. Performance claims remain attributed to the presentation.

Baseten — Flash-model inference optimization

Tri Dao professional profile

Official summit speaker roster

15. Hang Yin, Phala: confidential AI inference and dstack

SGLang Summit, November 13, 2026. Author's photographs and speech record; Yin is identified as cofounder and CTO. Attestation, reproducible builds and protected operation address a stated trust model. They do not verify the correctness of an AI answer or remove every application-level risk.

dstack repository

Private AI gateway

Microsoft confidential GPU documentation

16. WiCi InferSSD: local inference and expert streaming

SGLang Summit 2026. Author's slide photographs and speech record support the account of streaming selected mixture-of-experts weights from SSD. The displayed 7.27-token-per-second result is configuration-specific. Capacity expansion does not equate SSD with GPU memory or establish universal cloud parity.

WiCi InferSSD technical article

InferSSD repository

WiCi inference product

17. Traction Layer: Dim Sum and Demo

Companion gathering at Ghirardelli Square, San Francisco, November 12, 2026; separate from the SGLang Summit stage program. Author's photographs and conversation record include Shankaran Sitarama and Rahul Kundapur Ramesh. Scheduling results are demonstrations with tradeoffs, not independently repeated benchmarks. The commercial offer and tuner description come from Traction Layer's materials.

Traction Layer product

Traction Layer performance

SGLang scheduler implementation

18. Nikita Lazarev, Netpreme: GPU memory tiering

SGLang Summit, November 13, 2026, 2:10–2:25 p.m. Pacific, Decode Room. Author's slide photographs and speech record. Lazarev is identified as cofounder and CTO. Capacity and bandwidth figures are presenter-reported and retain their individual scopes; external capacity is not treated as equivalent to local HBM.

Netpreme — XM-Sparse, HiSparse and XMem

Netpreme

Research paper, arXiv:2405.04434

19. Xibo Sun, Tencent TACO: resource-efficient inference

SGLang Summit 2026. Author's slide photographs and speech record support the account of resource management, cache interfaces and compatibility. Product descriptions and software documentation are supporting technical context. Enlarged state capacity, correctness of sharing and measured serving performance remain separate claims.

Tencent TACO product

FlexKV repository

SGLang unified-cache components

20. Zhenhua Zhu: Uni³Sim

SGLang Summit, November 13, 2026, 3:30–3:45 p.m. Pacific, Decode Room. Author's photographs and speech record identify a research project associated with Tsinghua Shenzhen International Graduate School. Simulation results are presenter-reported. The project repository was not audited and its forecasts were not independently reproduced. The project concerns AI infrastructure, not a separately established robotics venture.

Academic biography

Official summit agenda

21. Palo Alto Networks: small language models for real-time security

Matthew Schulz and Zhemin Su; SGLang Summit, November 13, 2026, 3:45–4:00 p.m. Pacific, Decode Room. Author's slide photographs and speech record. The displayed single-GPU comparison reports p99 latency of 53.1 versus 44.2 ms at three concurrent clients, against a 50 ms target. It does not measure security accuracy or fleet-wide throughput. Issue records provide implementation context, not independent validation of the full benchmark.

SGLang issue 43930

SGLang issue 43929

RayService documentation

22. Yağız Çalık, ai&: reinforcement learning across heterogeneous GPUs

SGLang Summit, November 13, 2026, 10:15–10:30 a.m. Pacific, HiCache Room. Author's photographs and speech record. The 48-GPU configuration, four device types and two vendors are presenter-reported. Throughput and idle-time improvements retain their test context; they do not independently establish improved model capability.

ai& training

Miles pull request 3690

SGLang pull request 48241

PyTorch — Miles post-training stack

23. Xiang Deng, NeoCognition: ApprenticeBench

SGLang Summit, November 13, 2026. Author's photographs and speech record. The simulated construction-accounting task tests continuing work with history, instructions and feedback. Demonstration playback time, internal task time and cost counters are distinct. Benchmark improvement does not establish autonomous production accounting.

ApprenticeBench

NeoCognition

Xiang Deng biography

24. Tobias Gessler and Rishi Malhotra, Axiom Math

Training Systems for Formalizing All of Mathematics, SGLang Summit, November 13, 2026. Author's slide photographs and speech record support the account of proof search, Lean and formalization. Axiom's project announcements and fundraising page supply company disclosures. Eight Putnam problems within the competition window and four later are retained as separate timing claims. Funding and round valuations are dated disclosures, not revenue, current cash or evidence of commercial adoption.

Axiom — fundraising disclosures

Axiom — The Stacks Project Formalized

Axiom — The Enormous Theorem Formalized

Lean language and proof assistant

25. Simon Shaolei Du, Apodex: deep discovery and prediction

SGLang Summit, November 13, 2026. Author's photographs and speech record. The multi-agent reasoning examples, including the football example, are attributed to the presentation; they are not independent forecasts reconstructed after the event. Selection of a minority answer in one case does not establish a general rule.

Apodex introduction

Apodex 1.0

Apodex models

Simon Du academic biography

26. Sangwu Lee, Krea: Scaling RL for Diffusion Models

Official agenda: SGLang Summit, November 13, 2026, 5:10–5:25 p.m. Pacific, Decode Room. Session identification follows agenda and topic, not facial identification. Author's photographs and speech record support the reward-design examples. Product and technical reports supply company context; a high reward score is not treated as proof of an acceptable finished image.

Krea 3 technical report

Krea 3

Official summit agenda

27. Ryan Lee, MiniMax: multimodal systems and agent workflows

SGLang Summit, November 2026; session identity follows agenda and topic. Author's slide photographs support the workflow account. MiniMax's original product and technical materials supply company context. An English-language product or international availability does not by itself establish a quantified U.S. revenue contribution.

MiniMax company profile

MiniMax M3.1F

MiniMax H3

SGLang — MiniMax M3.1 Flash support

28. Ethan Anderson, SF Compute: where AI profits may settle

SGLang Summit, November 13, 2026, 2:40–2:55 p.m. Pacific, Decode Room. Author's photographs and speech record. The scenario matrix examines combinations of compute demand and model supply; it is a conditional framework, not an unconditional forecast. Obscured slide text has not been reconstructed.

SF Compute

SF Compute documentation

SF Compute company background

29. Jasper Zhang, Hyperbolic: the global AI compute grid

SGLang Summit, November 13, 2026. Author's photographs and speech record. Zhang is identified as cofounder and CEO. The 1,000-GPU reservation and resale-recovery arithmetic explains an illustrative slide scenario, not a quoted offer or a verified customer saving. Descriptions of service availability and rollout are attributed to the speaker.

Hyperbolic — company direction

Hyperbolic — financing announcement

30. Zhanghao Wu, SkyPilot: reinforcement-learning infrastructure

SGLang Summit, November 13, 2026. Author's photographs and speech record. The discussion concerns coordination of rollouts, training and supporting services. Utilization percentages and test results remain tied to the displayed configuration. Company technical articles and documentation supply additional workflow detail.

SkyPilot — RL Is Everything Everywhere All at Once

SkyPilot — RL Is Bottlenecked by Inference

SkyPilot — Grace Blackwell needs a new scheduler

Job Groups documentation