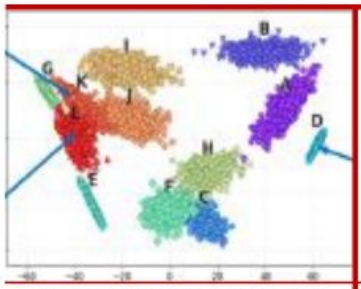
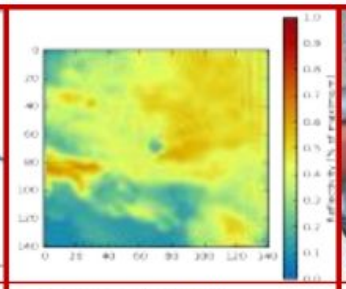
Scalable Data Management: A Key Driver for AI on HPC Systems

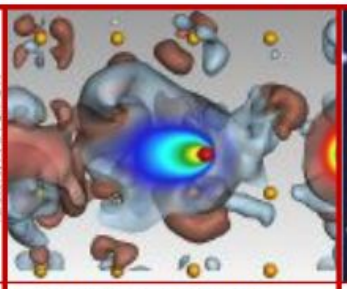
Bogdan Nicolae
Argonne National Laboratory, USA
Illinois Institute of Technology, USA

Argonne NATIONAL LABORATORY

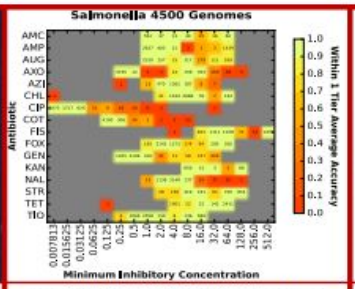
Deep Learning in HPC/Scientific Computing


Reduced order modeling of laser sintering

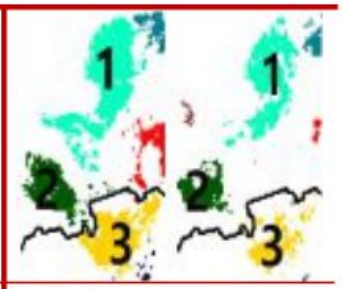

Nowcasting with convolutional LSTMs

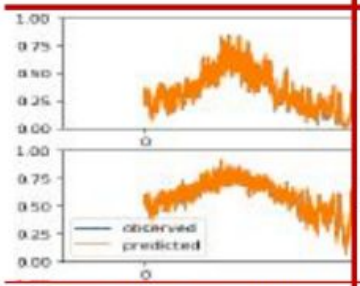

Prediction of radiation stopping power


Strong and weak lensing in sky survey data

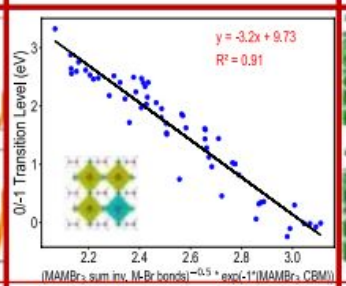


Prediction of antimicrobial resistance phenotypes

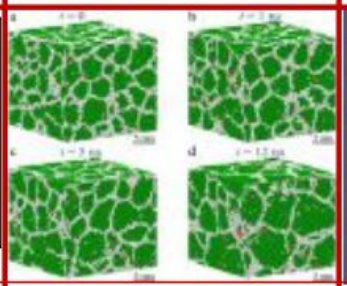

Identification and tracking of storms

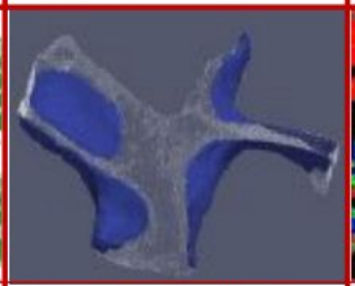


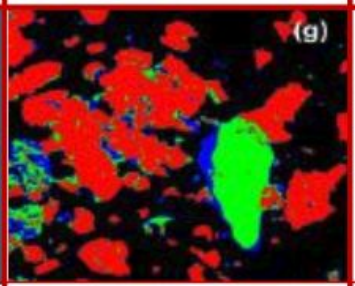
Efficient climate model emulators



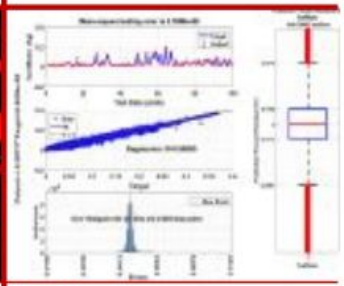
Defect-level prediction in seminconductors


Structure-property-process triangle in additive manufact.

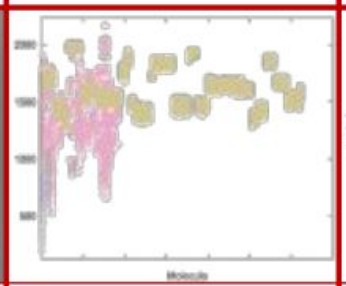

Parameter extraction in atom probe tomography

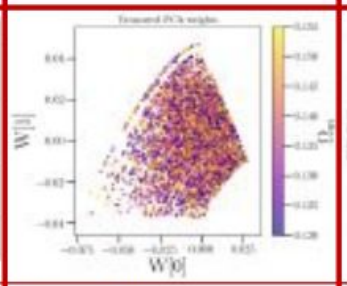


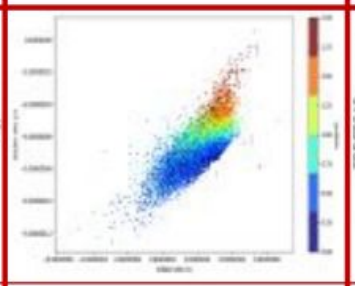
Learning for dynamic sampling in spectroscopy

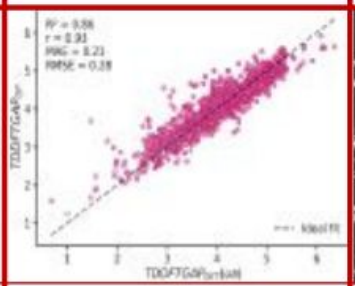

Vehicle energy consumption prediction

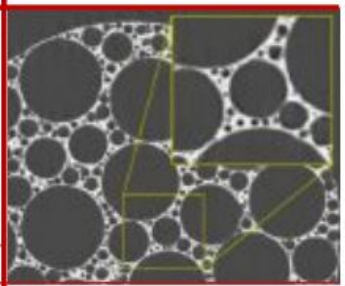

Flying object detector for edge deployment


Discovery of new energy storage materials


Cosmic Microwave Background emulation


Photometric red shift estimation

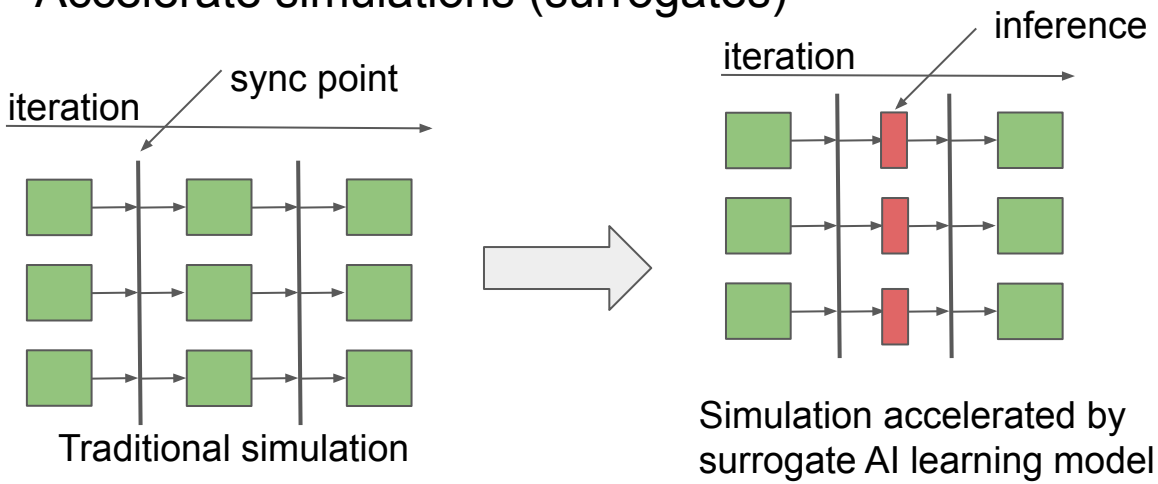

New materials for efficient solar cells


Enhancement of noisy tomographic images

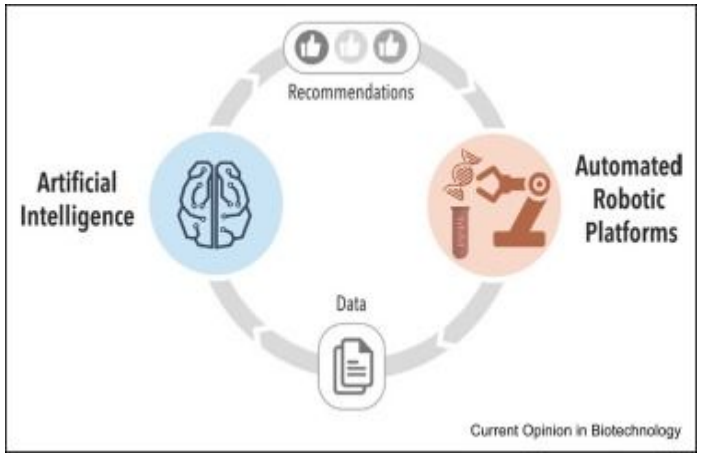
10 years ago: widespread use of “traditional” AI

Deep Learning in HPC/Scientific Computing

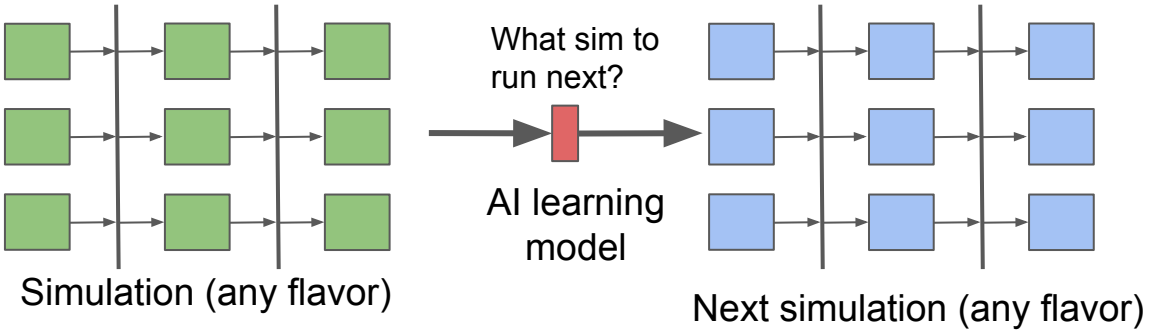
Accelerate simulations (surrogates)



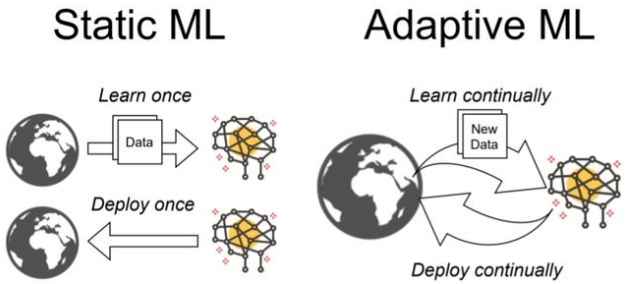
Self-driving laboratories



Planning and execution of ensembles



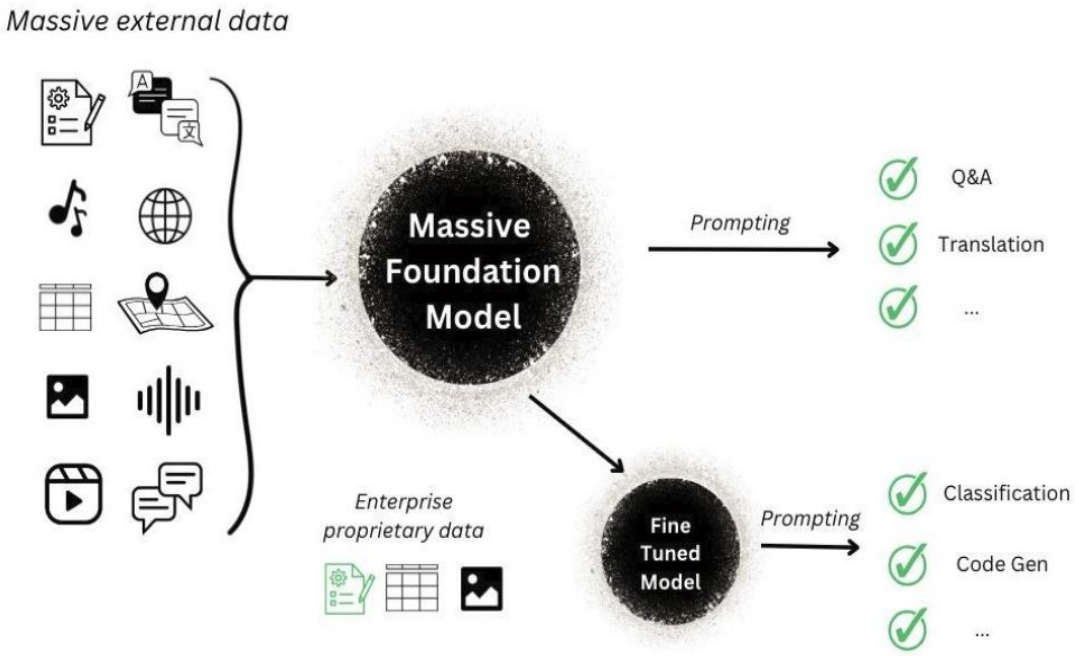
Real-time steering and acquisition



Edge Instruments + Supercomputers

5 years ago: more tighter integration with simulations and edge

AuroraGPT: Foundational Model for Science



AI assistants based on foundational models (e.g. LLMs) will be the norm in the future

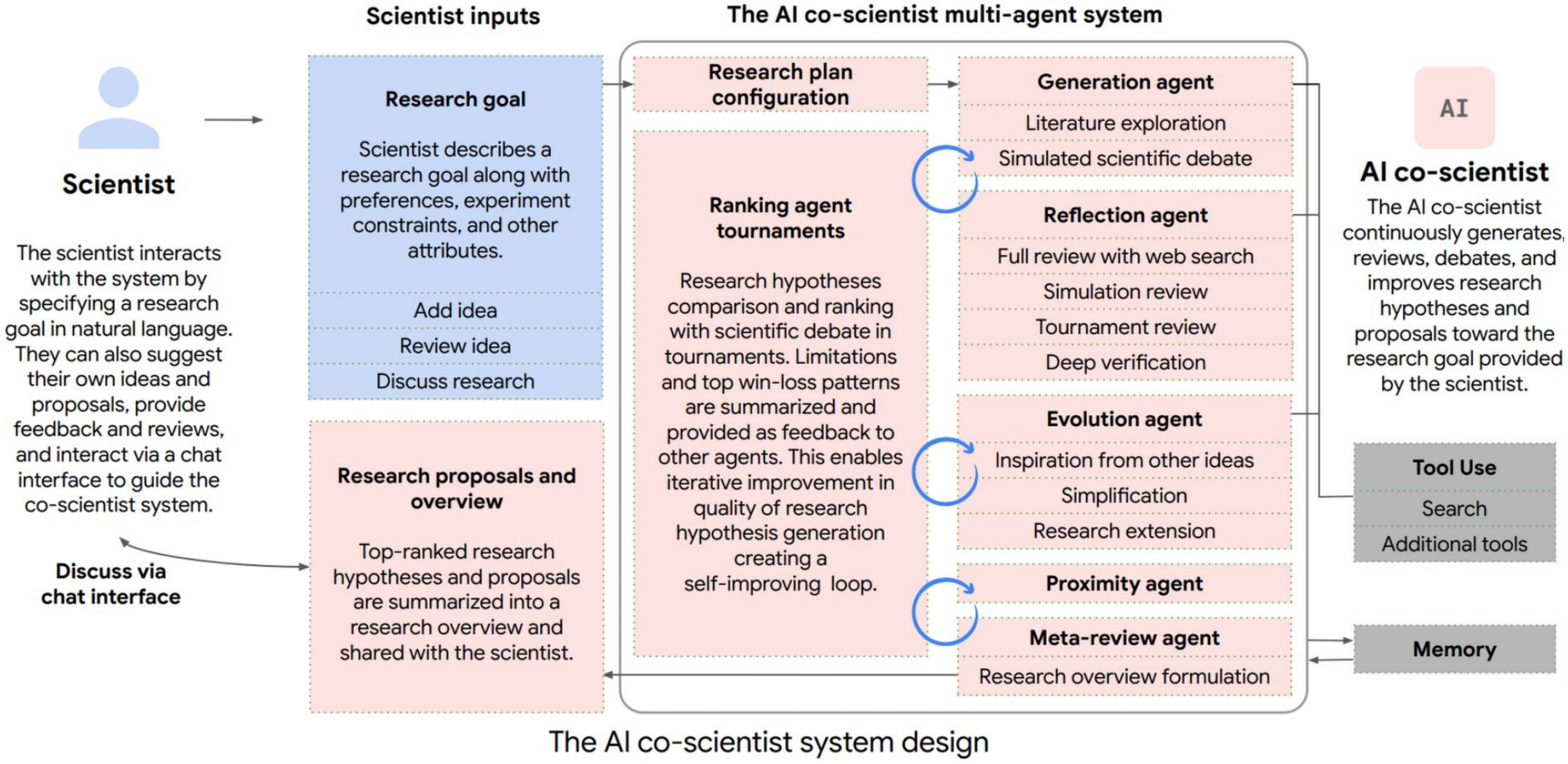


https://auroragpt.anl.gov/



2 years ago: The Age of LLMs and Transformers

Agentic AI Workflows For Science (1)

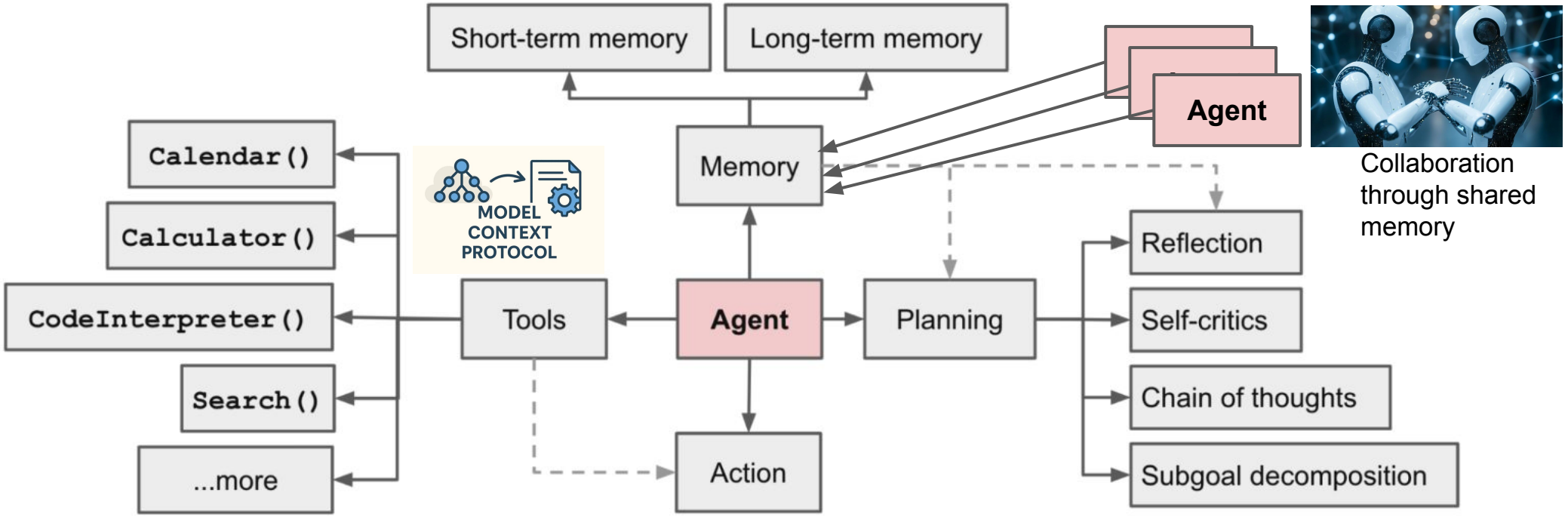


The AI co-scientist system design

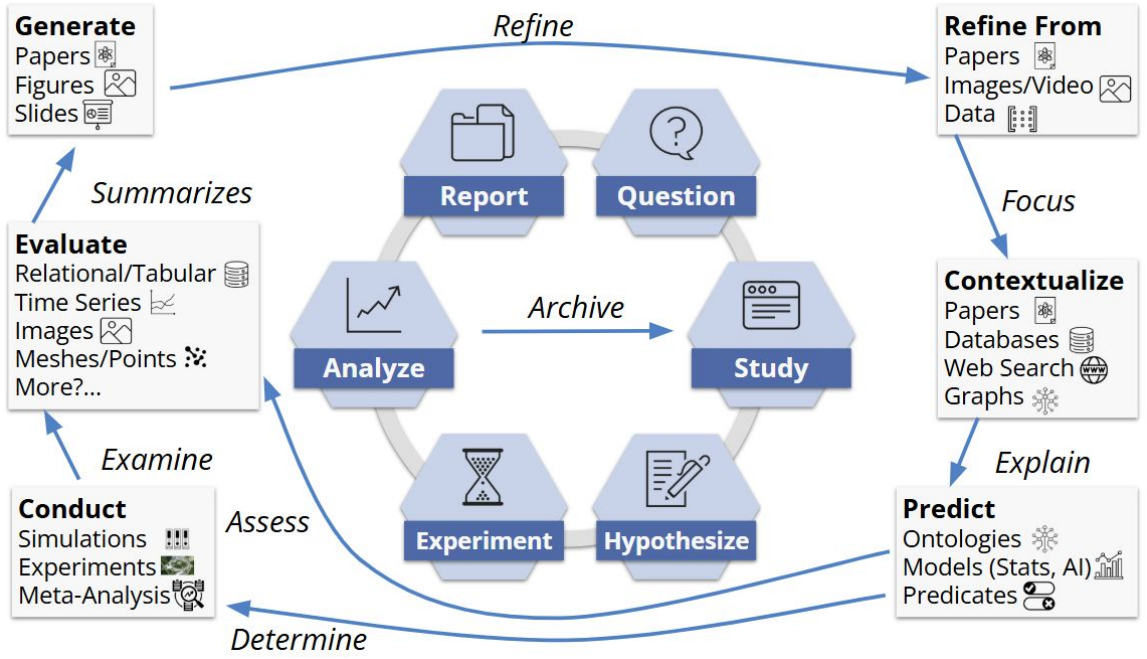
*Gottweis, Juraj et al. “Towards an AI co-scientist.” *ArXiv* abs/2502.18864 (2025)

Now: The Age of Agentic AI Workflows

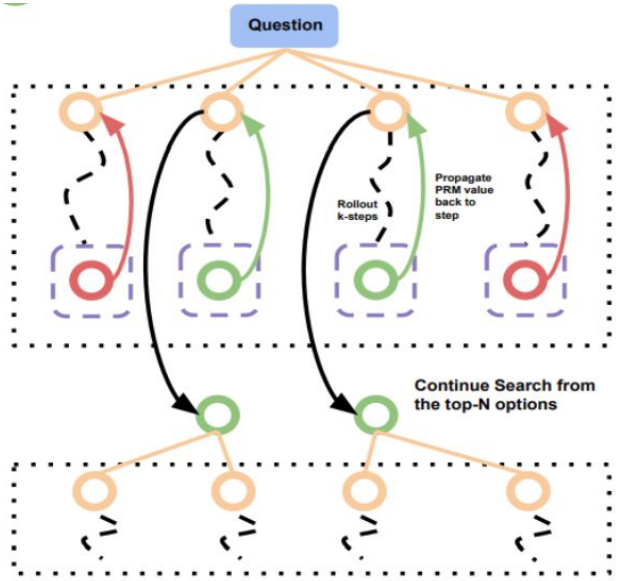
Agentic AI Workflows For Science (2)



Collaboration through shared memory



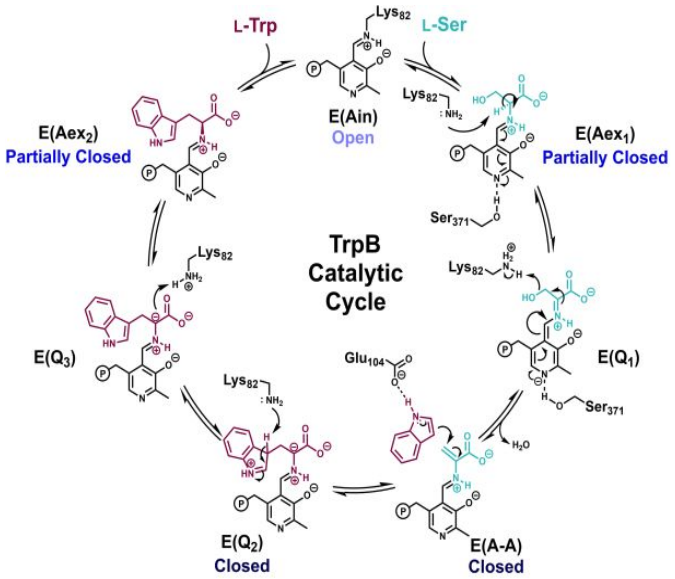
Example of lookahead ToT reasoning strategy



Agentic AI Examples

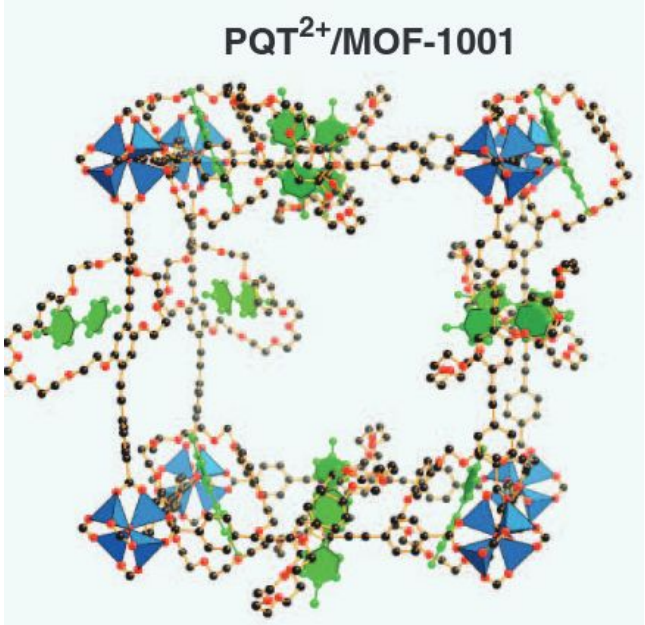
PeptideAgents (BioInformatics)

- PeptideAgents automates the search for hard-to-find peptides (high specificity, low toxicity, and ability to target bacteria, fungi, and viruses precisely)
- Seven agents employ specialized reasoning strategies that explore many overlapping variations
- Multi-modal data with separate embeddings: proteins from databases, published literature, past simulation data, new simulation data, experimental measurements



MOFA (Material Design)

- Metal Organic Frameworks (MOFs) are crucial for gas absorption (e.g. carbon capture)
- Five agents automate the design of MOFs: (1) generate candidate ligands; (2) combine ligands with metal clusters to create MOFs; (3) validate candidates using molecular dynamics simulations; (4) calculate fitness; (5) retrain the generator
- Multi-modal data: massive density function theory (DFT) datasets from simulations, published experimental findings, semantic graphs



Several New AI Supercomputers at Argonne

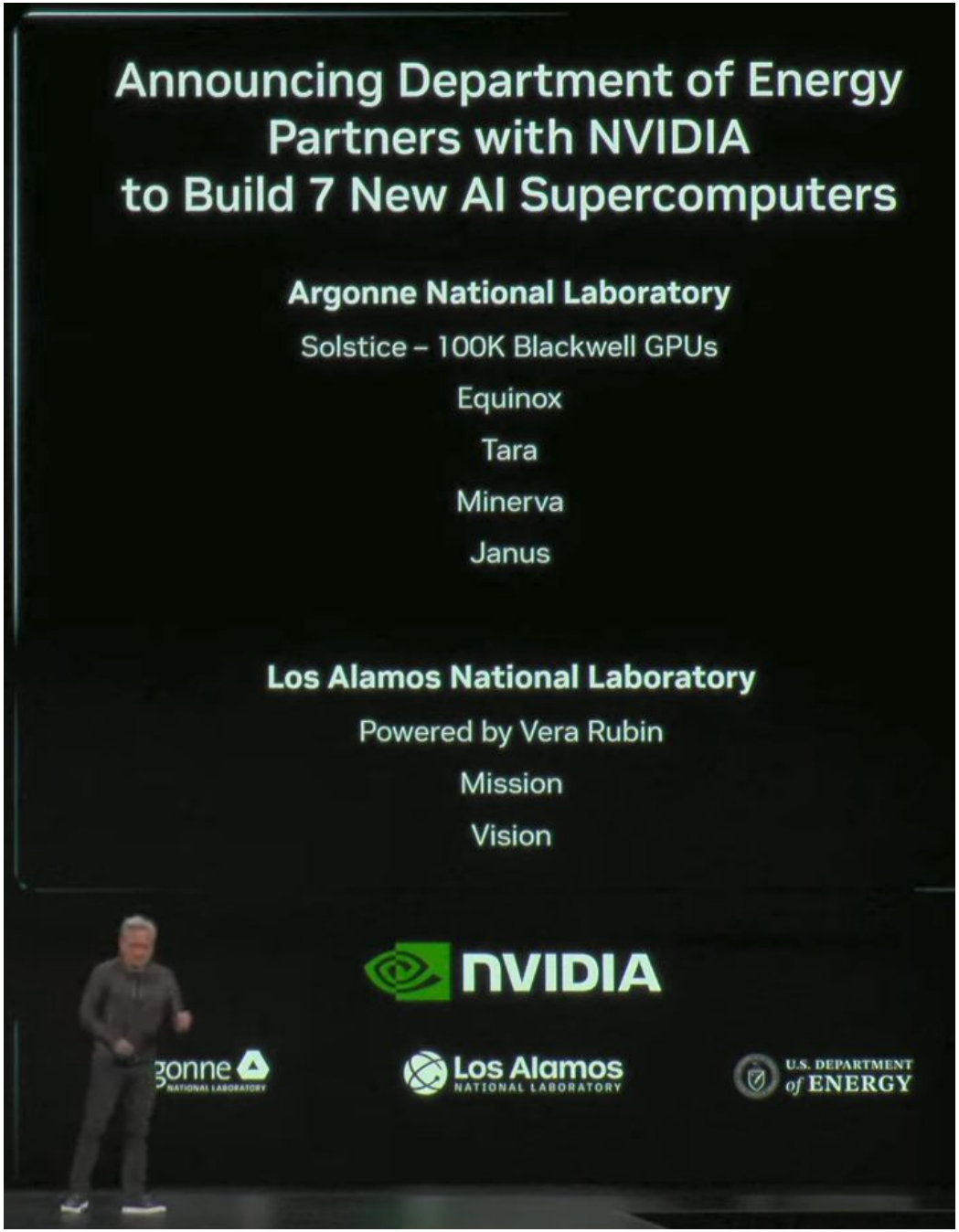


First Phase: Equinox

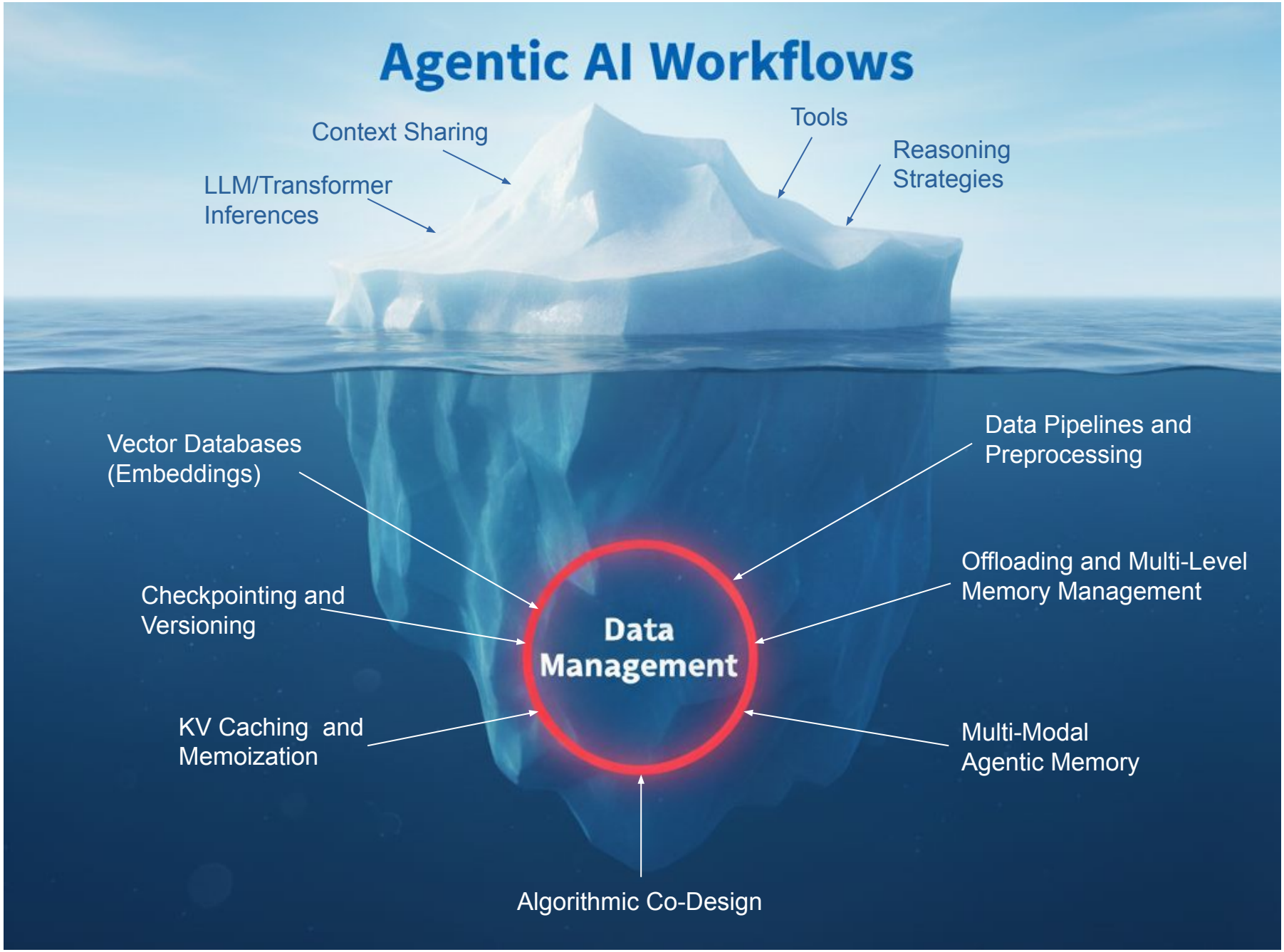
- 10K Blackwell GPUs
- Delivery planned 2026

Second Phase: Solstice & Co

- 100K+ Blackwell GPUs (200 MW)
- 2000 FP4 ExaFLOP
- Support 3T parameter LLMs/Transformers with agentic AI workflows

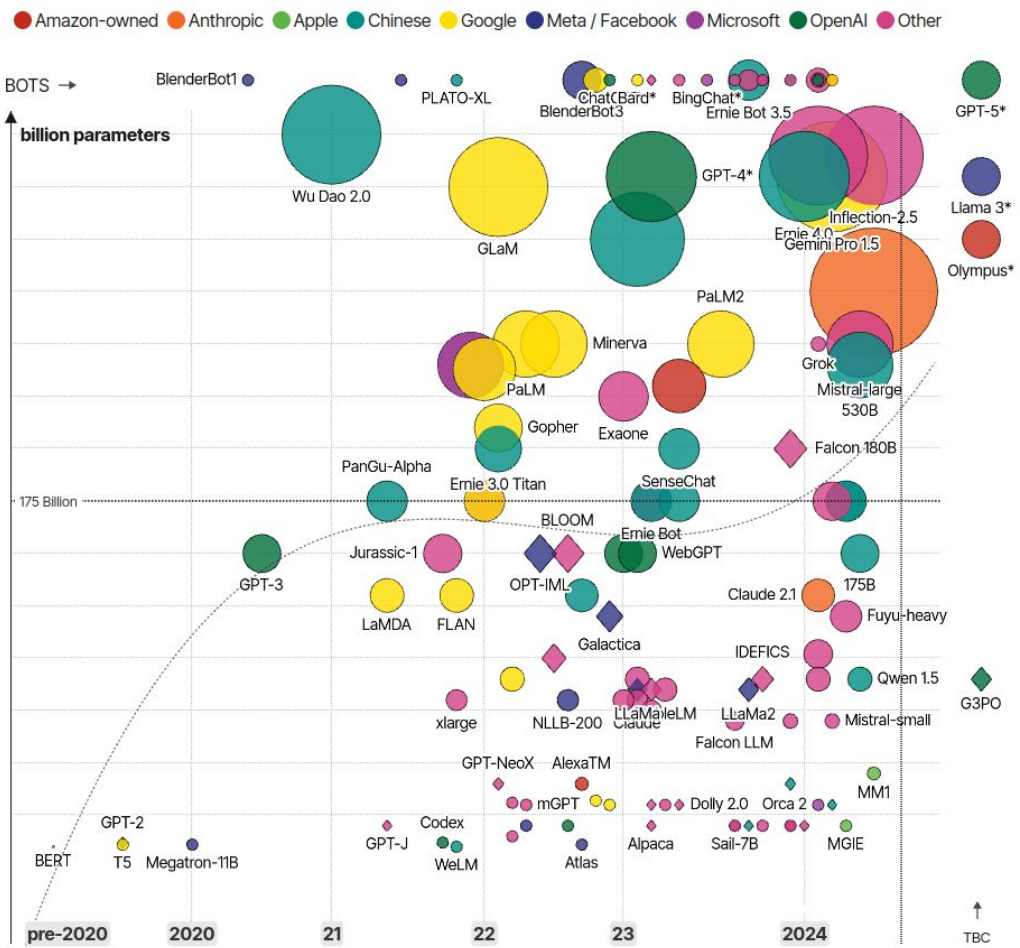


Scalable Data Management Foundation



Checkpointing of LLMs and Transformers

LLMs are Getting Bigger and More Expensive to Train



David McCandless, Tom Evans, Paul Barton
Information is Beautiful // UPDATED 20th Mar 24

source: news reports, LifeArchitect.ai
* = parameters undisclosed // see the data

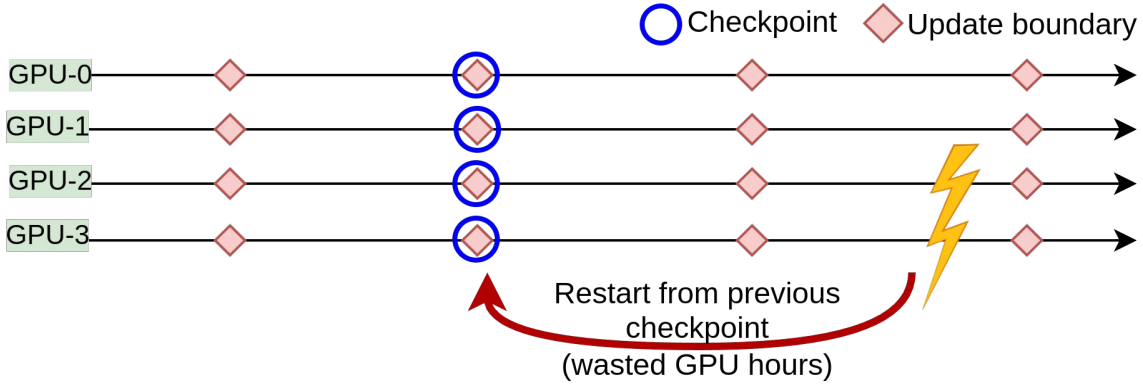
Model	Number of GPUs	Duration
GPT-3 (175B)	10,000	34 days
GPT-4	25,000	Several months
PaLM (540B)	6,144	2 months
Turing NLG	560	Several months
Bloom (176B)	384	3 months
Chinchilla (70B)	4,096	1 month
T5 (11B)	1,024	1 month

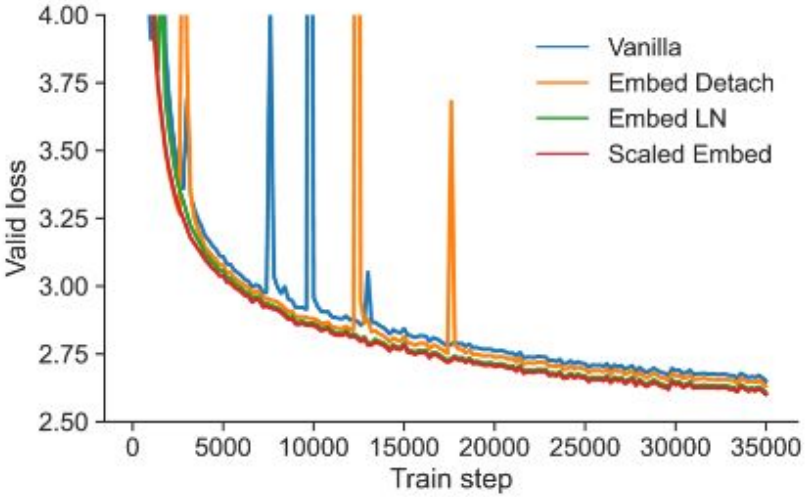
AuroraGPT: How much does pre-training cost?

Model Size (B)	Tokens (Trillion)	Aurora Time (h)	Polaris Time (h)	Aurora Time (Days)	Polaris Time (Days)	Cloud Cost ($3 GPU/hr)
7	2	2.29	333	0.10	14	$437K
7	3	3.34	500	0.14	21	$656K
70	2	22.88	3,333	0.95	139	$4,374K
70	3	34.31	5,000	1.43	208	$6,561K
200	6	196.08	28,571	8.17	1,190	$37,496K
200	10	326.80	47,619	13.62	1,984	$62,494K
1000	10	1633.99	238,095	68.08	9,921	$312,470K
1000	20	3267.97	476,190	136.17	19,841	$624,941K

Can afford very few tries

Why Do We Need Checkpointing?





What can go wrong during one (but long) pre-training:

Failures

- NCCL timeout
- NVLink error
- Invalid DMA mapping
- Task hung up
- Link flapping

Impact one or more processes

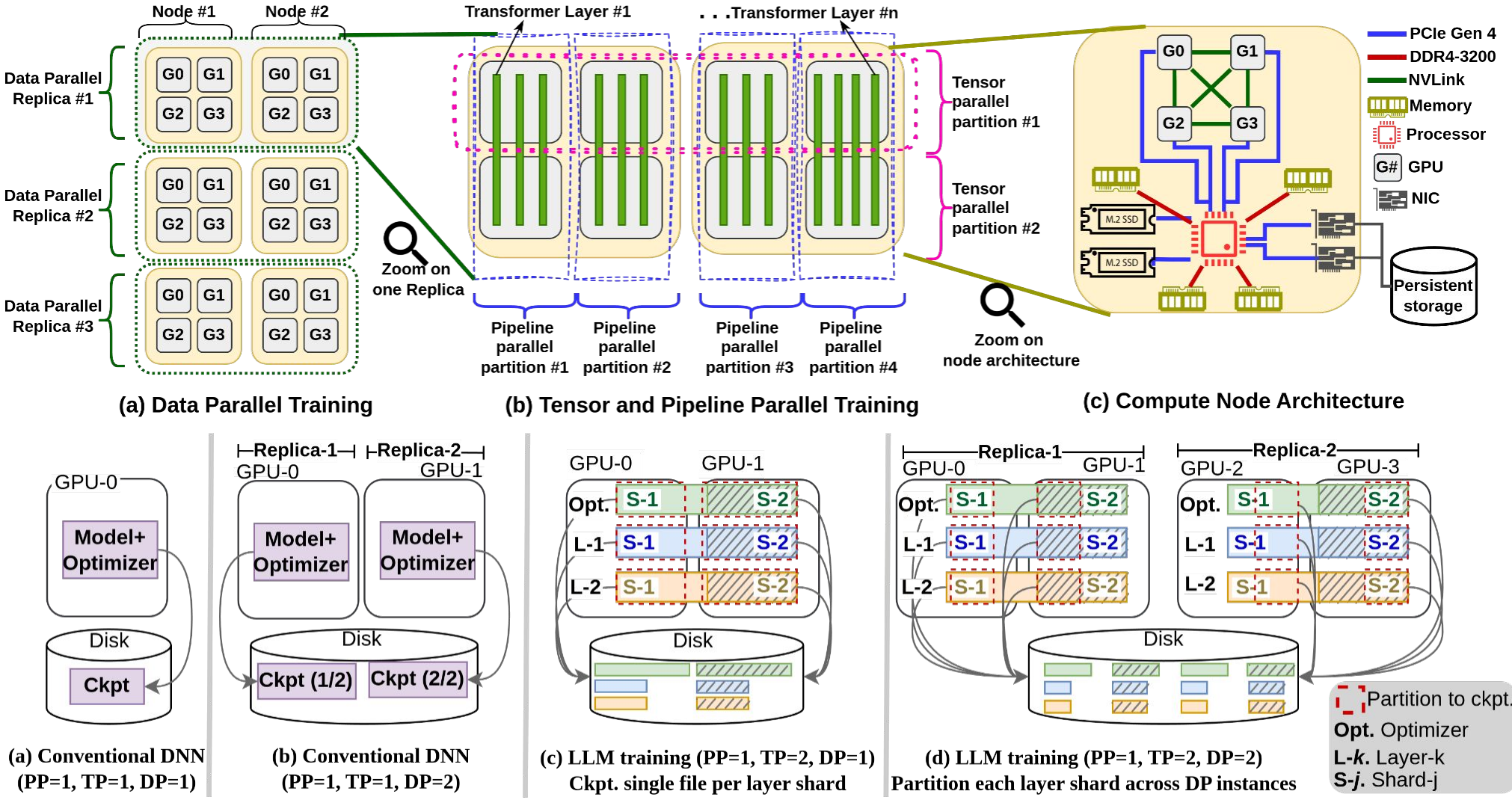
Undesirable training trajectories

- Reason: loss spikes, biases, unreliable training samples
- Complex recovery: mini-batch skipping, gradient clipping, alternative optimizers
- More difficult than fault tolerance: last checkpoint may be unreliable
- Need to keep a history of checkpoints and backtrack as needed further in the history

Administrative Interruptions

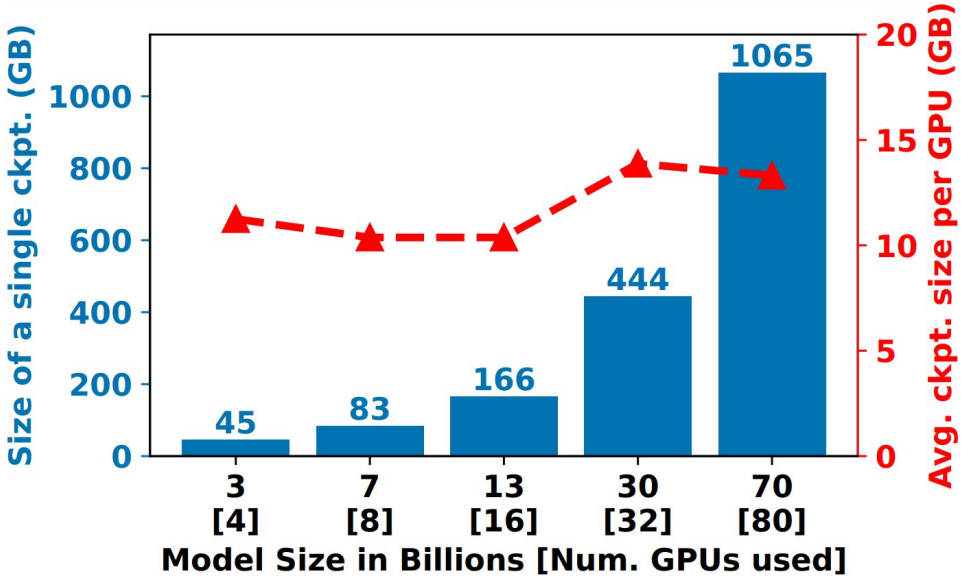
- Understanding model evolution
- Forensics, biases & ethics: periodic evaluation
- Suspend-resume (e.g. every 6h) to allow other HPC jobs to run
- Elastic training: Vary number of GPUs

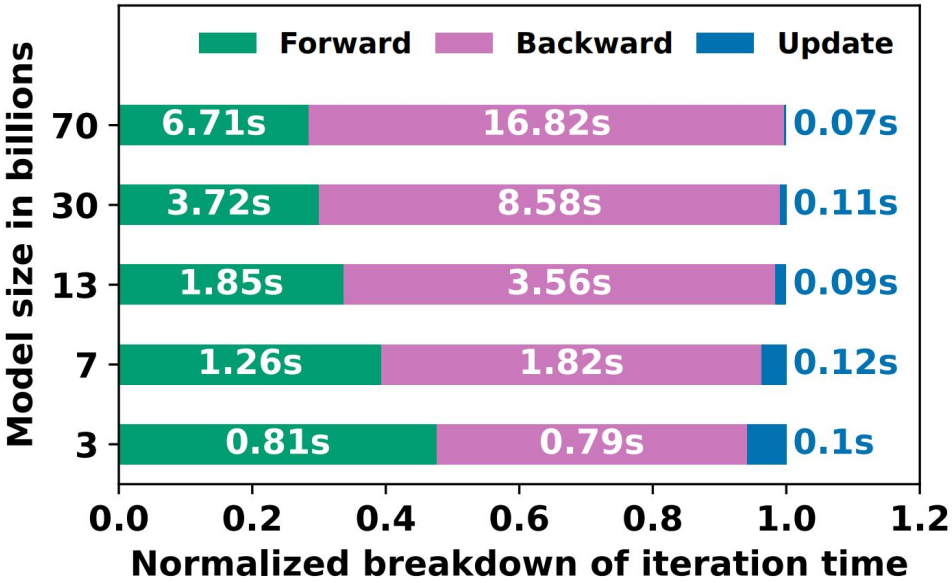
LLM Training Scales Using 4D Parallelism



- 4D Parallelism is commonly used: data, pipeline, tensor, sequence
- What do we checkpoint: model parameters, optimizer state (much larger)
- Many transformer blocks, each holds many distributed layers => large number of shards

Key Observations Driving Our Research

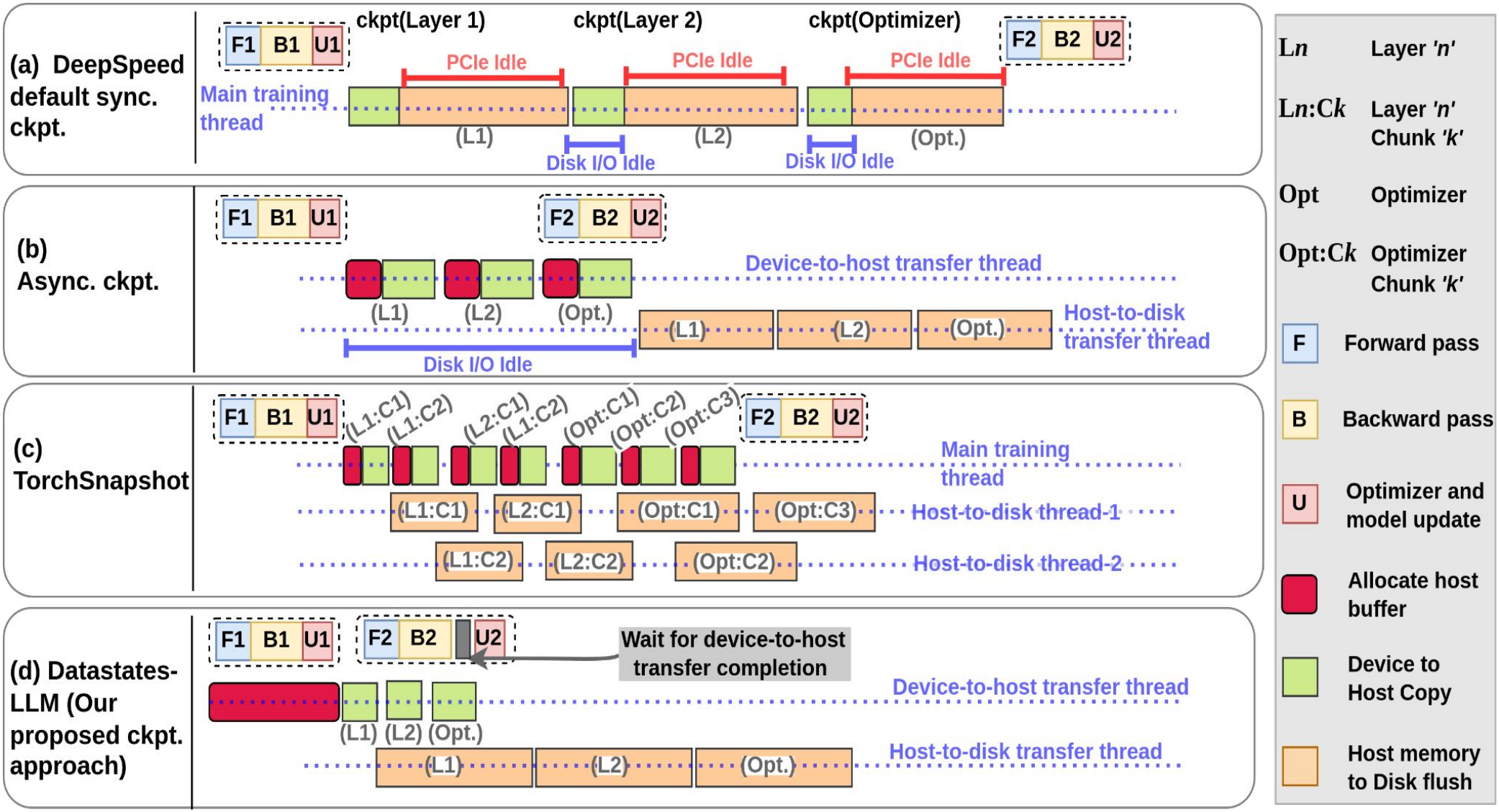




- Checkpoint size increases linearly with the number of model parameters
- Checkpoint **shard per GPU is load balanced** and remains the similar for different model sizes

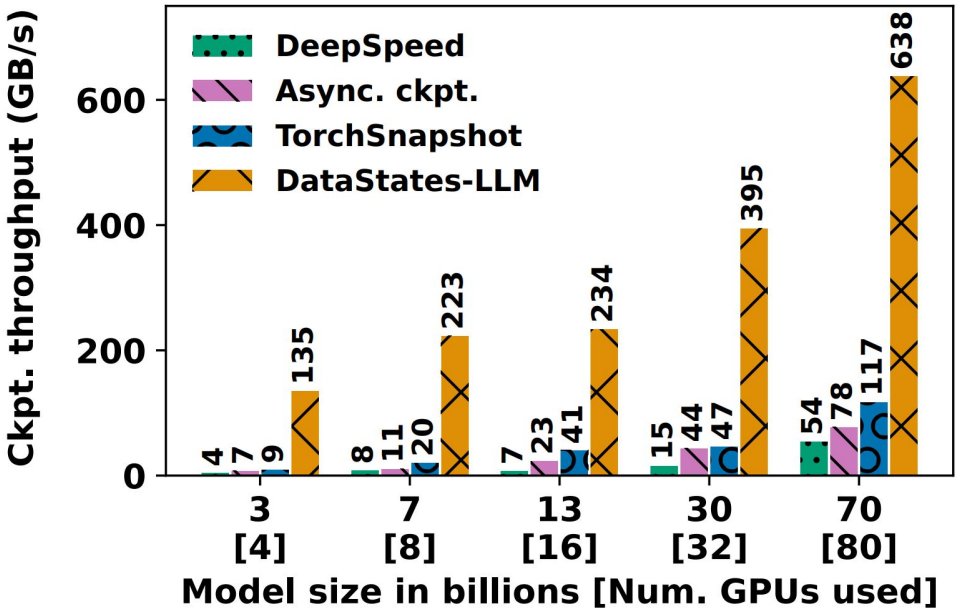
- Forward and backward pass consume majority of the iteration duration (>95%) during training
- **Model and optimizer states are immutable** during forward and backward passes

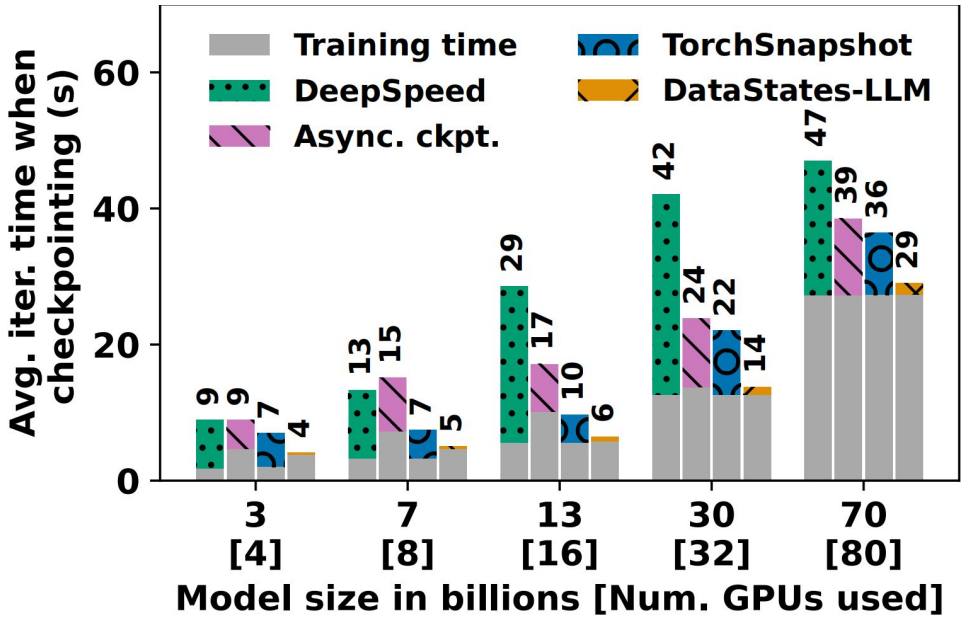
DataStates-LLM vs. State-of-Art



Maurya, A., Underwood, R., Rafique, M., Cappello, F. and Nicolae, B. 2024. DataStates-LLM: Lazy Asynchronous Checkpointing for Large Language Models. *HPDC'24: The 33nd International Symposium on High-Performance Parallel and Distributed Computing*. **Best Paper Award**

Results: Increasing Model Size





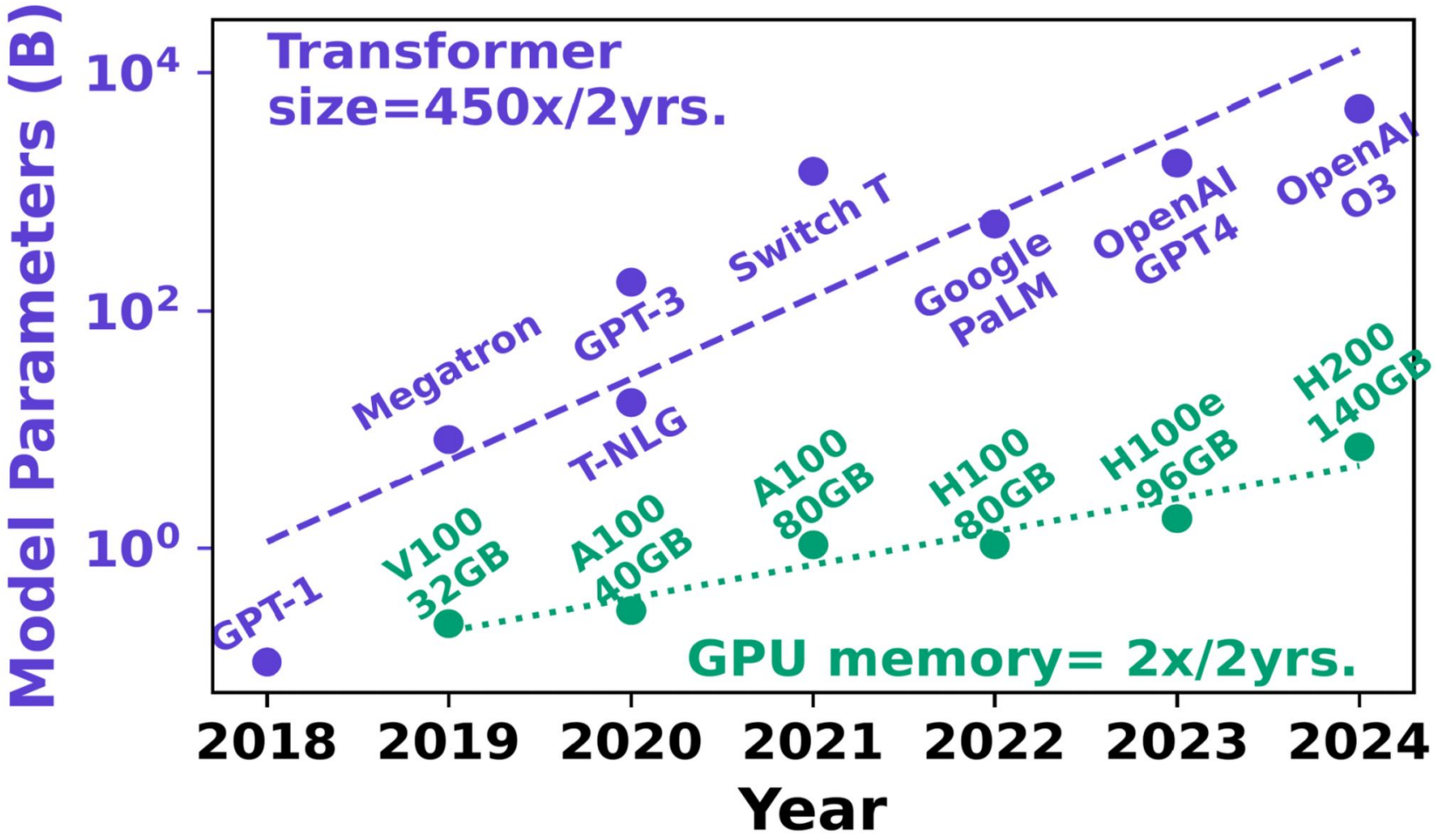
- DataStates-LLM achieves **4x – 34x faster checkpointing** throughput compared to state-of-art
- Checkpointing throughput increases for larger models because all GPUs flush in parallel

- DataStates-LLM achieves **1.3x – 4.5x faster iterations** compared to state-of-art
- DataStates-LLM shows negligible overheads on the training iteration when checkpointing

We contributed DataStates-LLM to DeepSpeed

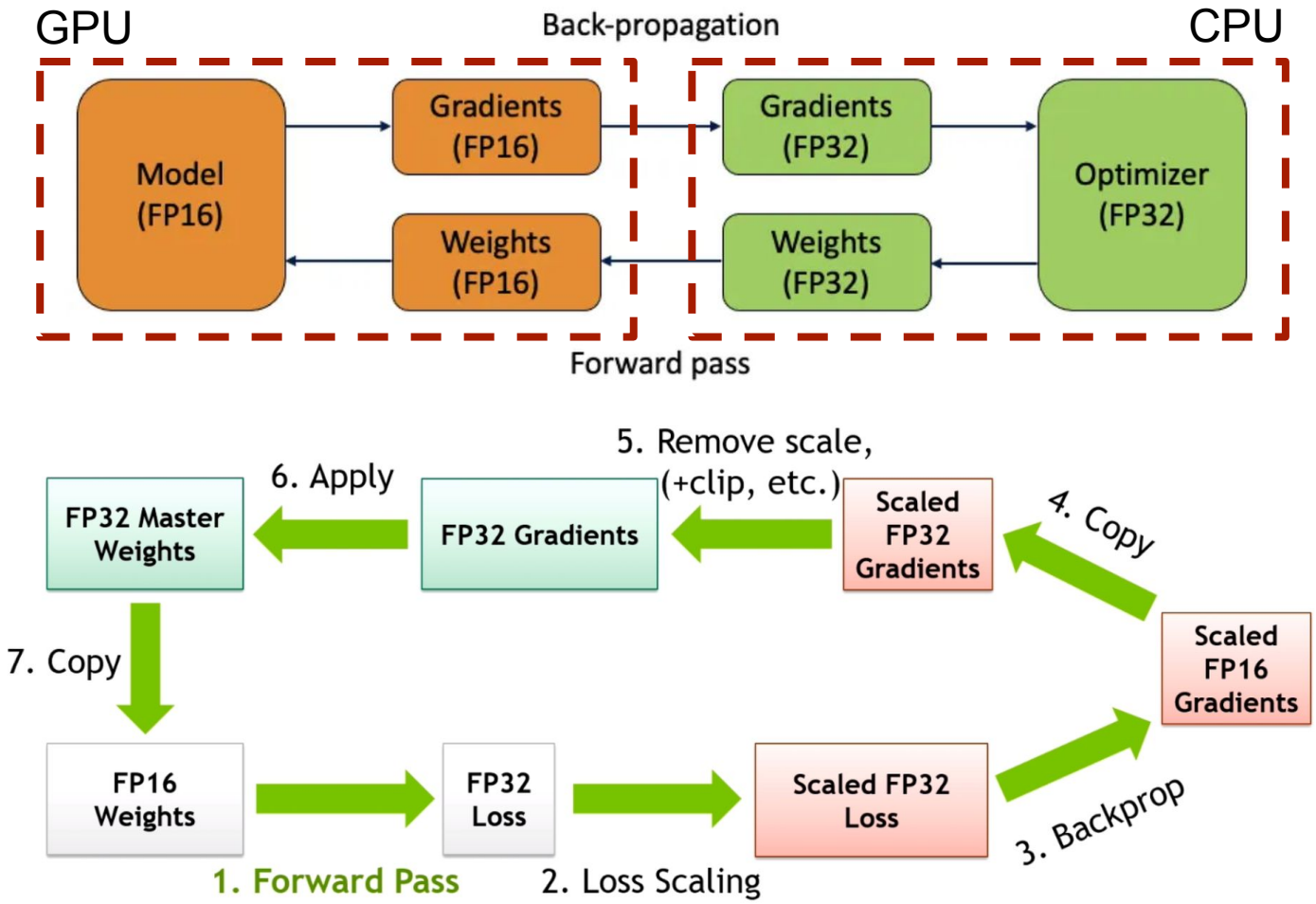
Breaking the GPU Memory Wall through Offloading

The GPU Memory Wall Hinders LLM Training



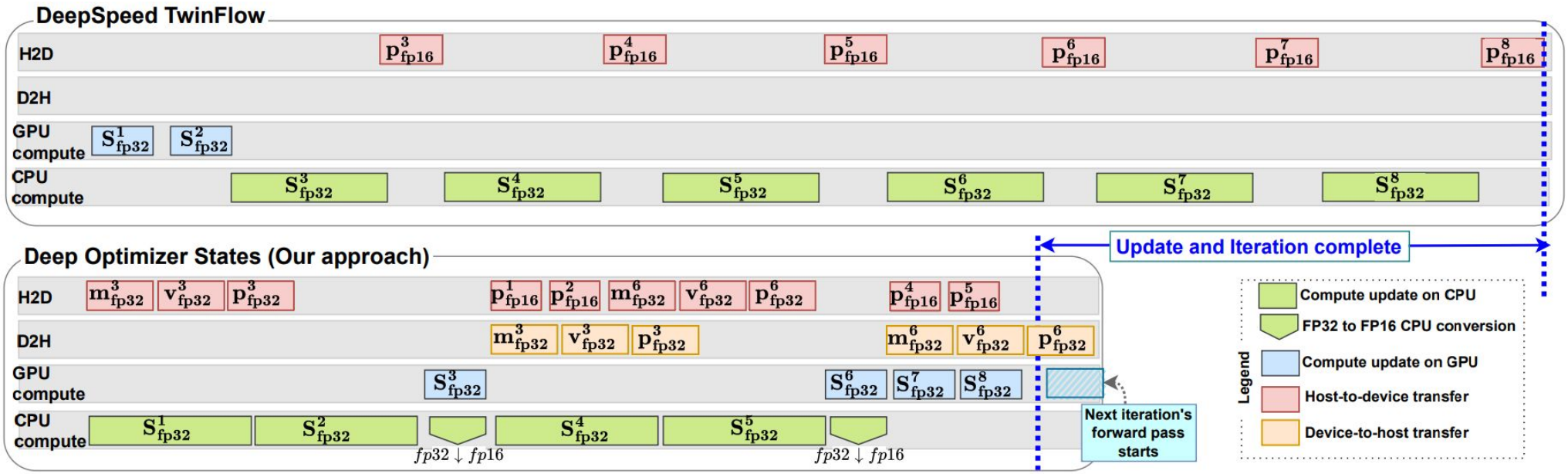
GPU memory capacity cannot keep up with the growth of LLMs

Solution: Offloading + Mixed Precision (MP) Training



- Reduced precision on GPUs accelerates forward and backward pass
- Update phase on CPUs in full precision controls rounding errors
- Optimizer state only needed by CPUs (updates), freeing up more GPU memory
- However, running the update phase on CPUs is slower

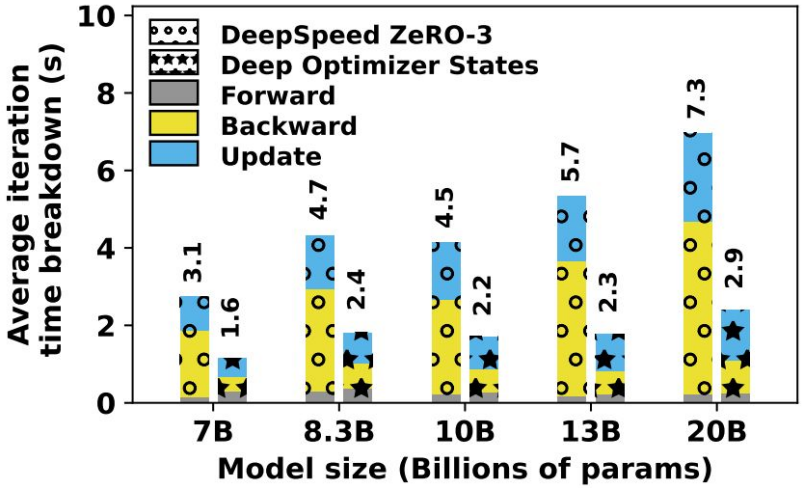
Deep Optimizer States: GPU-Accelerated Offload



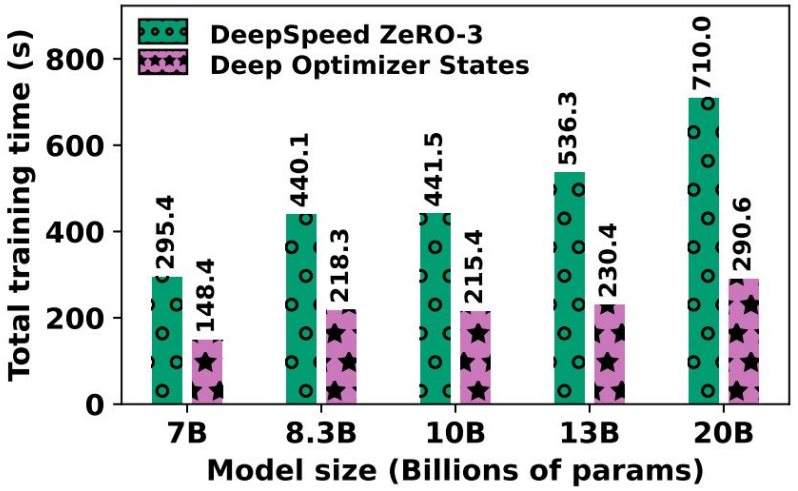
- Update N subgroups on the CPU for every one subgroup on GPU
  - Optimal N can be obtained through performance modeling (details in paper)
- When CPU is processing the n=2 subgroups
  - Flush out the previously updated GPU subgroup (parameters, momentum, variance)
  - Prefetch the next subgroup to be updated on the GPU (p, m, v, gradients)
  - Parallel bidirectional transfers use full-duplex D2H and H2D bandwidth of PCIe interconnect
- When GPU is processing update of a subgroup, CPU performs FP32-> FP16 downscaling of parameters for next iterations on GPU with mixed-precision

Maurya, A., Ye, J., Rafique, M.M., Cappello, F. and Nicolae, B. 2024. Deep Optimizer States: Towards Scalable Training of Transformer Models using Interleaved Offloading. In Middleware'24

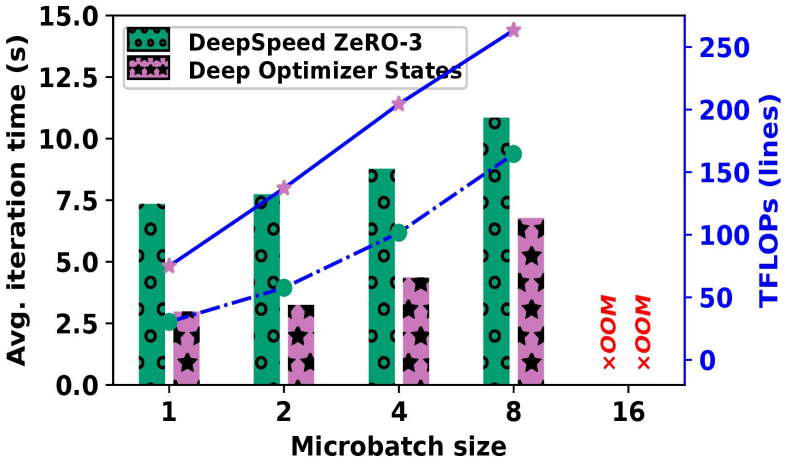
Results vs. DeepSpeed Zero-3 (State of Art)



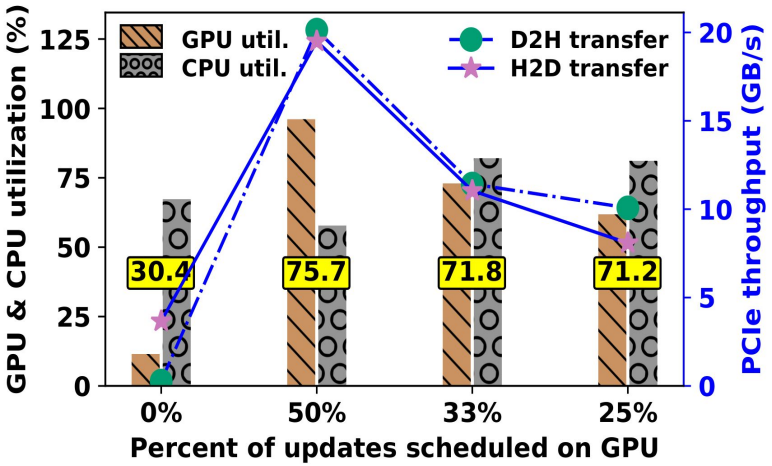
2x faster iterations across different model sizes



2x faster end-to-end training (100 iterations)



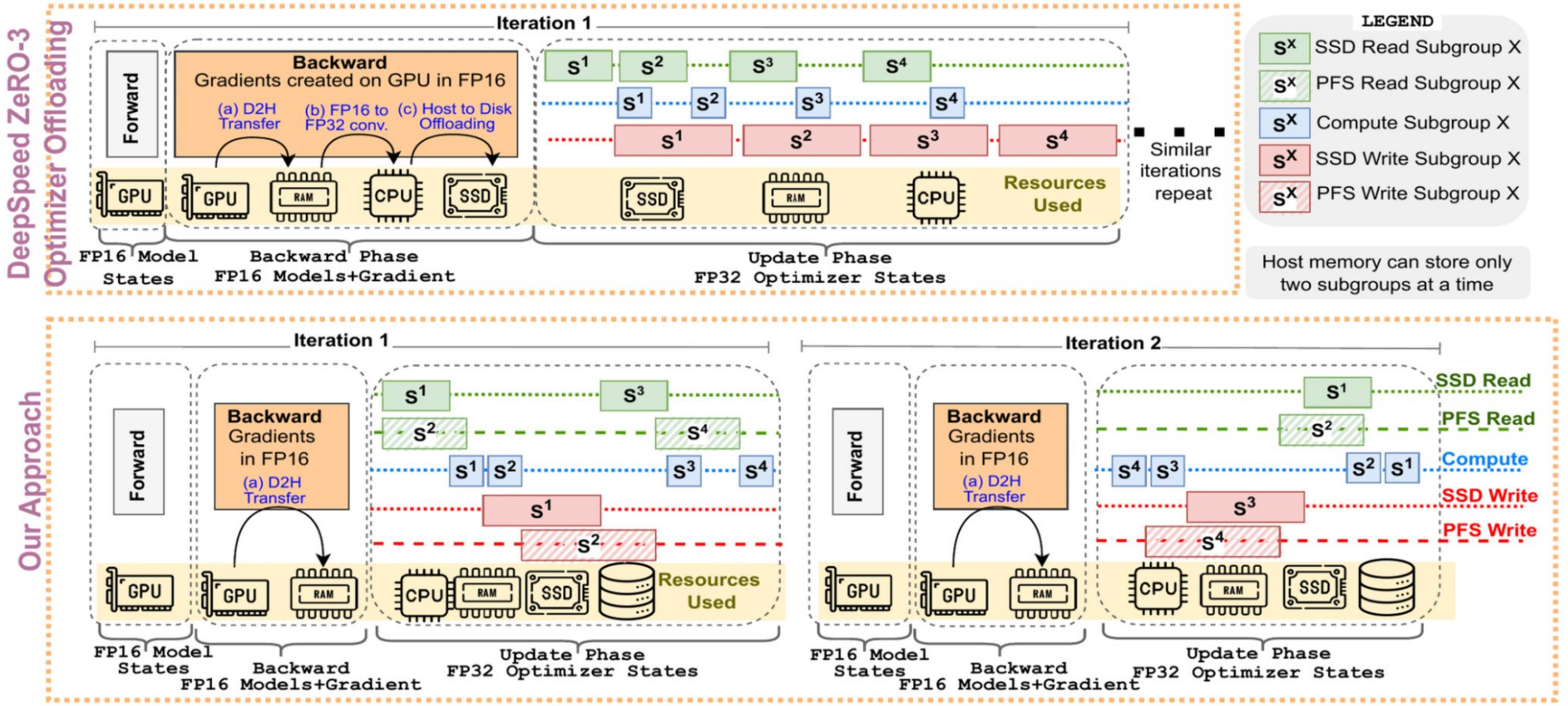
2x faster iterations for variable microbatch sizes



2x faster iterations for variable microbatch sizes

Our approach significantly outperforms existing offloading solutions by accelerating the backward and update phases

MLP-Offload: Multi-Path Accelerated Offload



- When host memory is not enough either, we can use local storage (DeepSpeed-Zero3)
- MLP-Offload: use local storage (SSD) + external storage (PFS)
- Several optimizations: cache-friendly reordering, in-place gradient conversion, etc.

Maurya, A., Rafique, M., Cappello, F. and Nicolae, B. 2025. MLP-Offload: Multi-Level, Multi-Path Offloading for LLM Pre-training to Break the GPU Memory Wall. To be presented at SC'25

MLP-Offload: Results

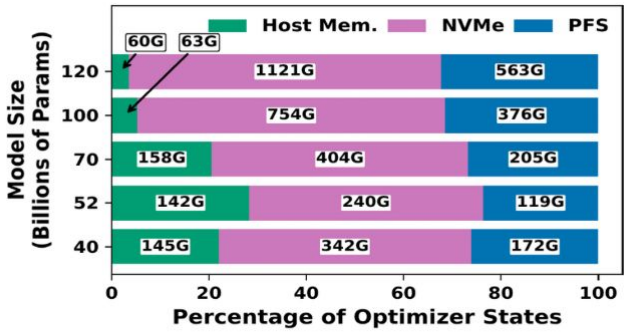


Figure 10: Distribution of optimizer states across different tiers.

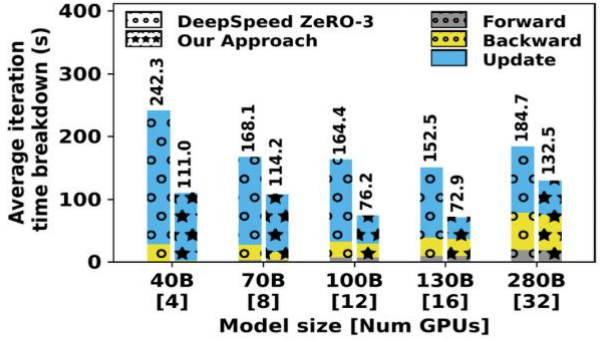


Figure 11: Weak scaling: Iter. time for increasing model sizes with # GPUs.

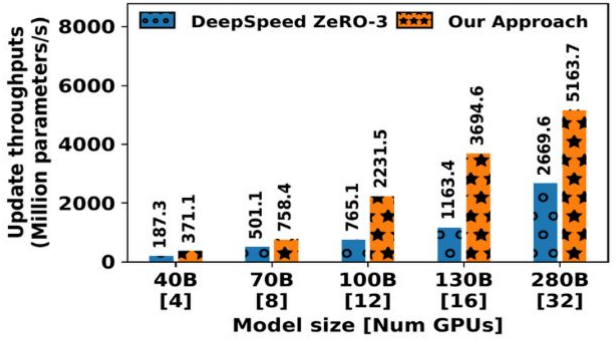


Figure 12: Weak scaling: Update thru. for increasing model sizes with # GPUs.

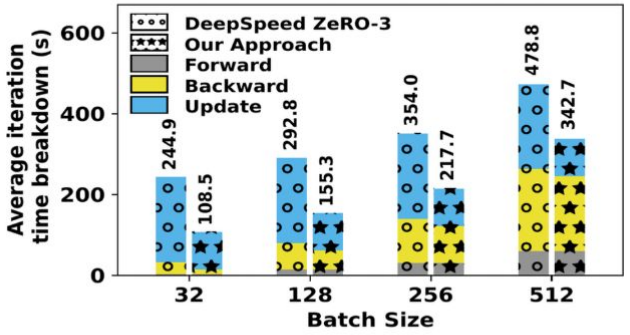


Figure 13: Average iteration time of different batch sizes for the 40B model.

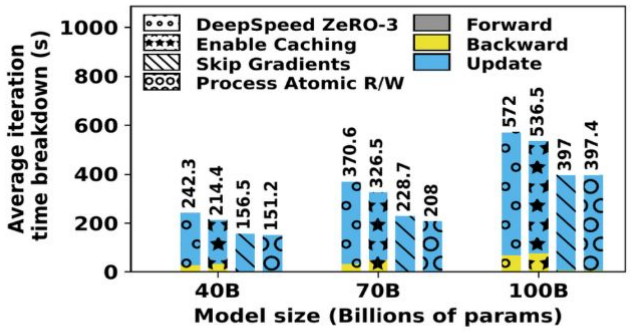


Figure 14: Performance ablation on node-local NVMe.

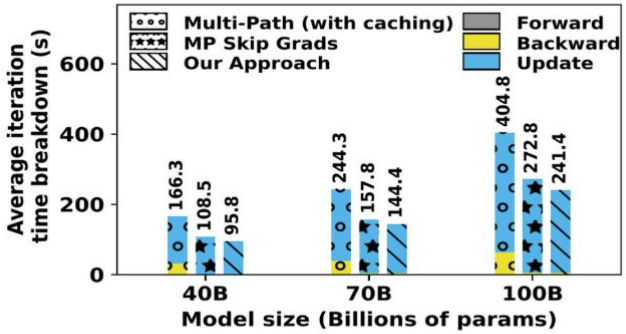
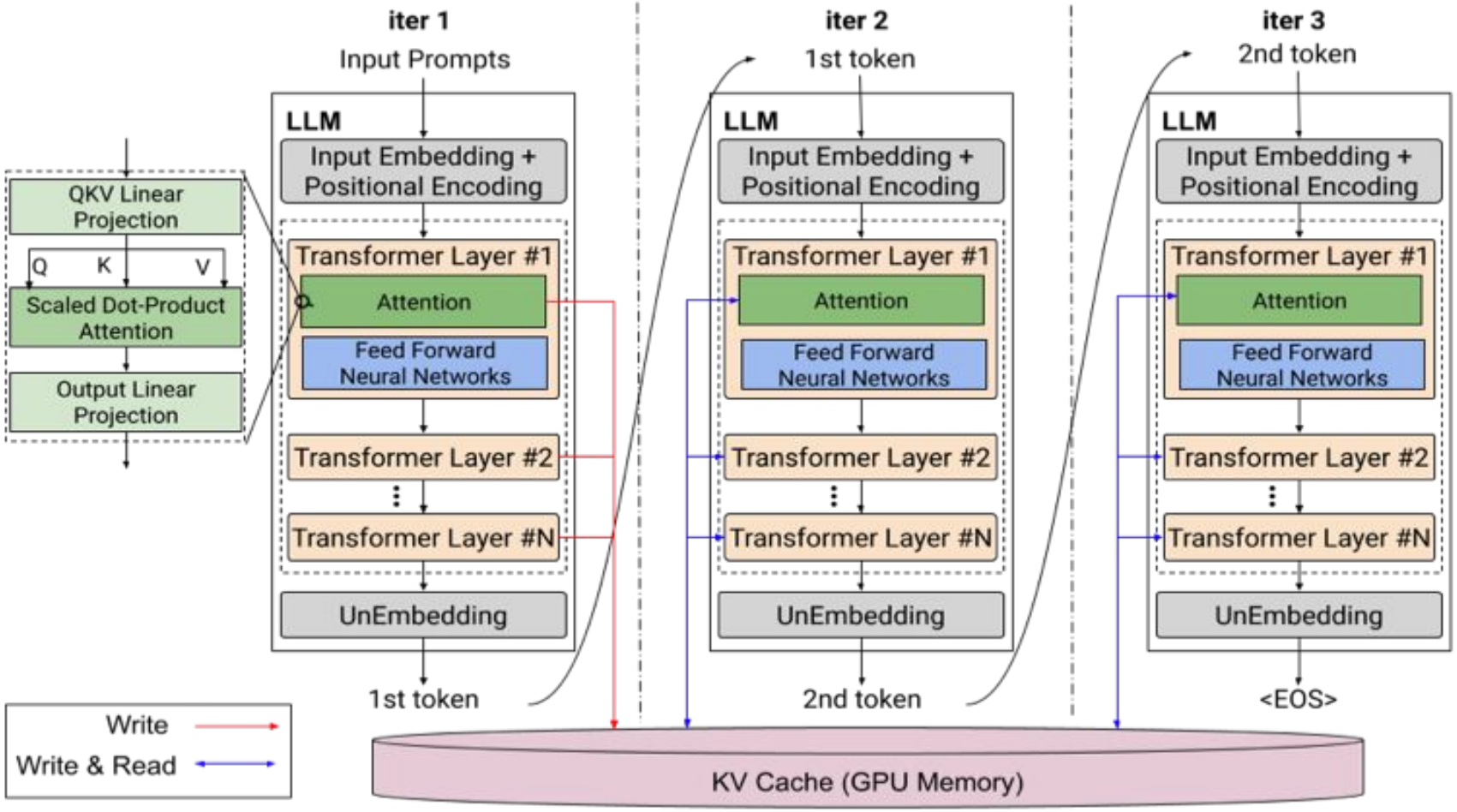


Figure 15: Performance ablation on node-local NVMe and PFS.

Our approach achieves 2.5× faster training iterations compared with DeepSpeed ZeRO-3 for models various scales (40B-280B LLama models)

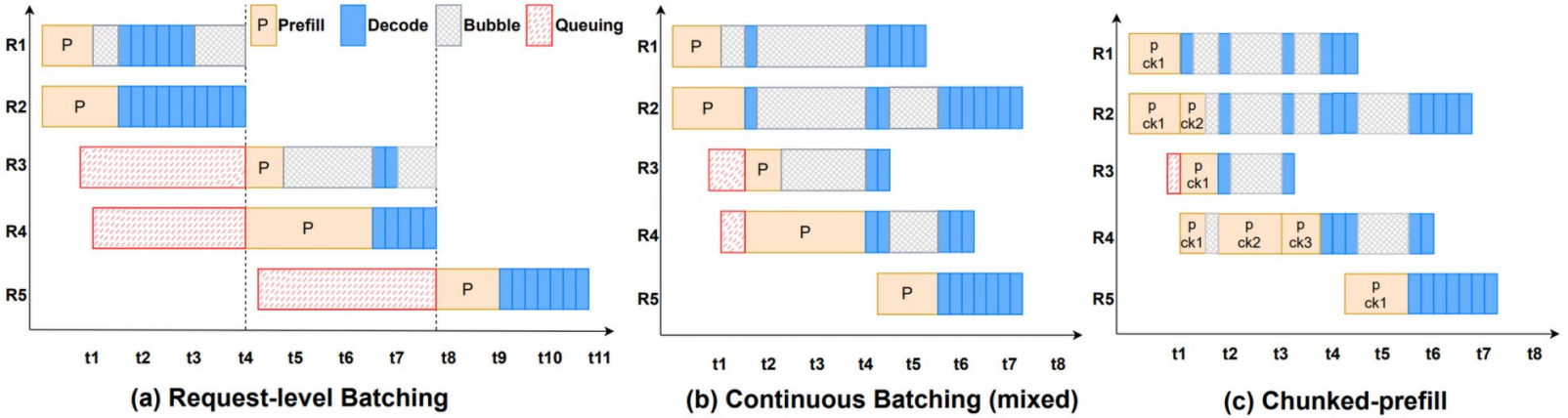
LLM and Transformer Inferences

LLM/Transformer Inferences

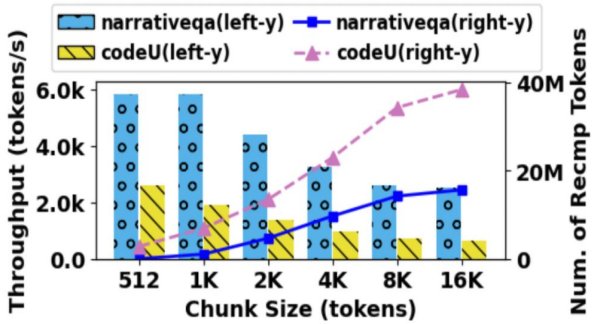


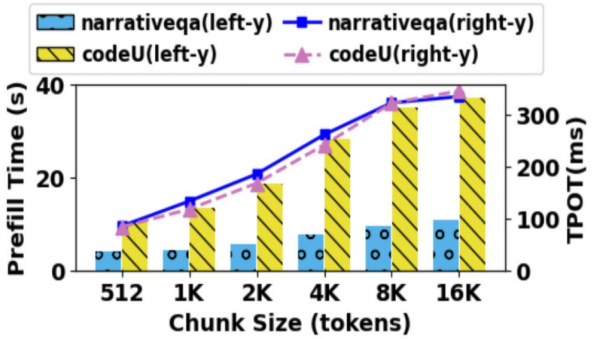
- Generative LLM inferences involve two phases:
  - **Prefill Phase:** Process the initial prompt using as many tokens as possible in parallel in the same forward pass
  - **Decode Phase:** Generate subsequent tokens one-by-one in separate forward passes
- Intermediate attention results (KV tensors) can be cached and reused
  - **Efficient KV Cache management** is a key data problem and performance challenge

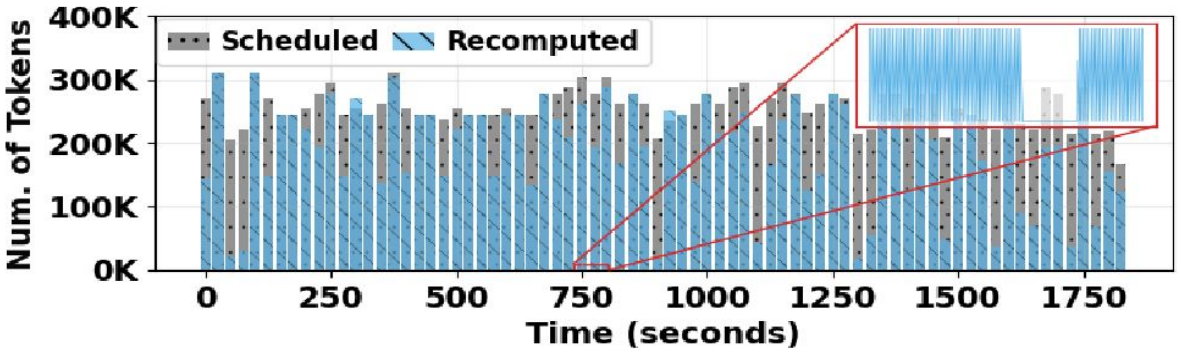
Optimizing GPU Utilization: Request Batching



- Without batching GPUs will be underutilized (especially during decode)
- Modern LLM inference frameworks (e.g. vLLM) use chunked prefill



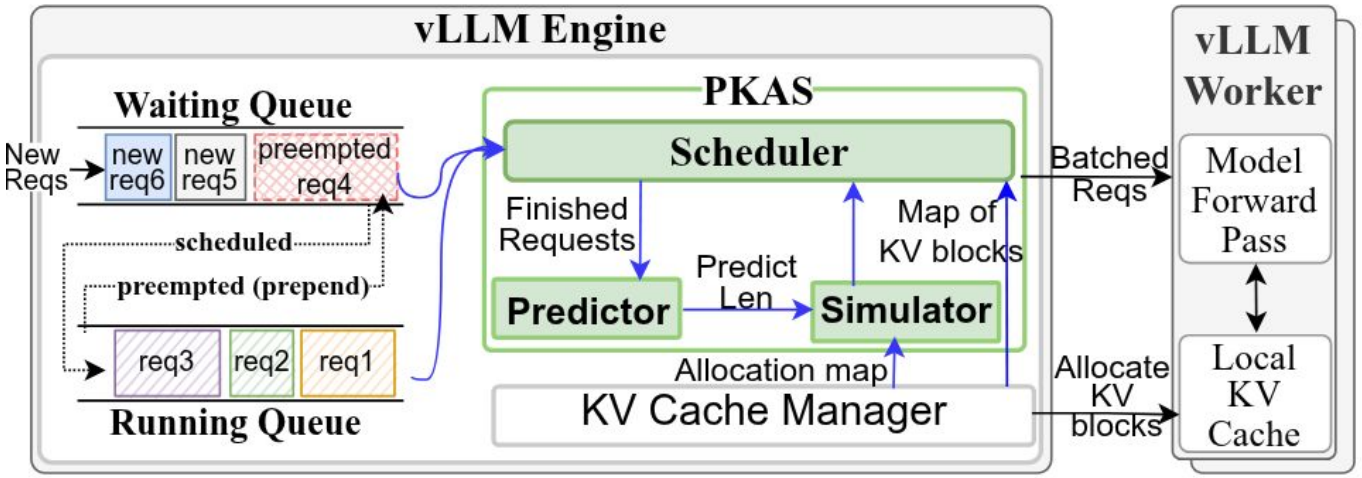




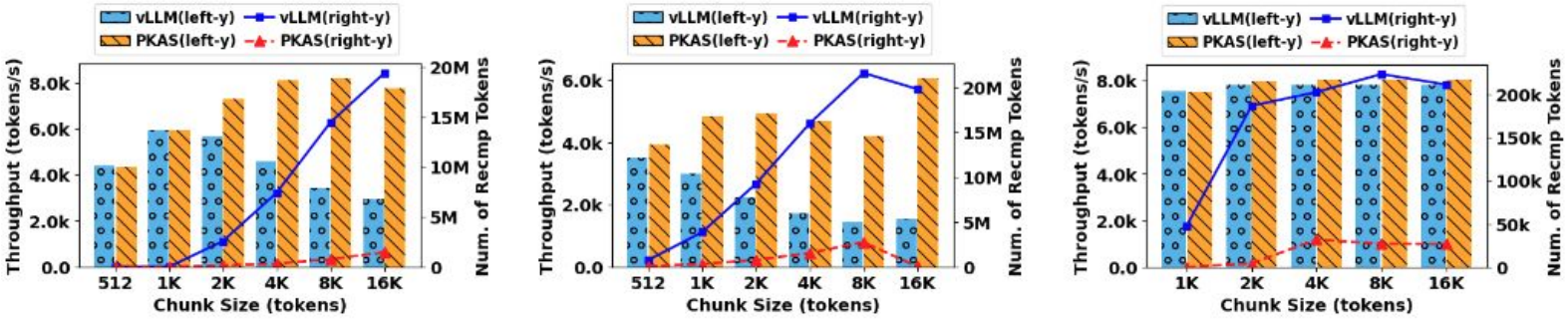
- Most batching strategies are **speculative**
- Speculations are difficult:
  - KV Cache overflows
  - Requests are Preempted and Recomputed Later
  - Large increase in TTFT and TPOT
  - Large waste GPU cycles
  - “Ping-pong” pattern

PKAS: KV-Cache Aware Predictive Batching

- Accept new requests only if they are likely not to cause preemptions
- Fast algorithms to simulate forward passes and KV cache utilization
- On-the-fly output token prediction

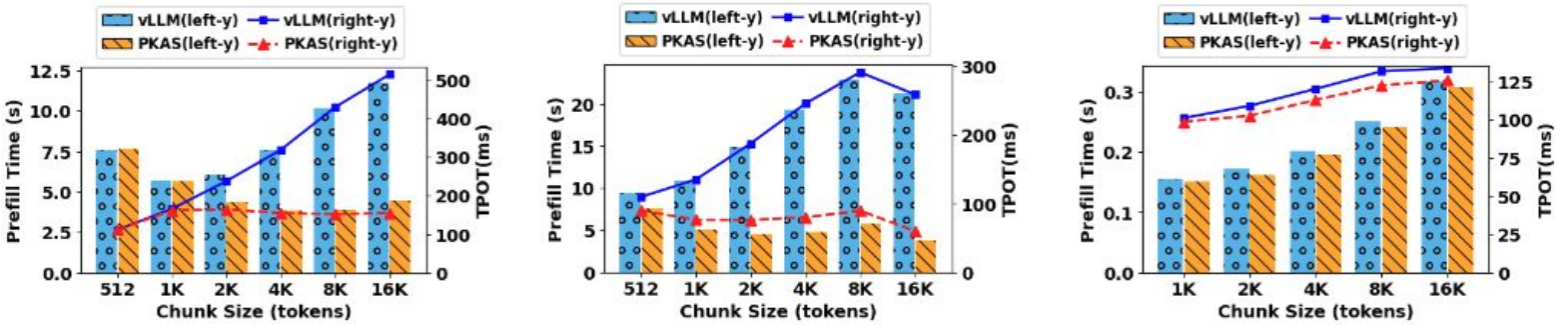


- Setup: Polaris (A100 GPUs)
- Comparison: vLLM vs PKAS
- Both large and small requests (3 datasets)
- PKAS improves throughput, reduces prefill duration, reduces TPOT several times (up to 5x)



(a) NarrativeQA dataset (b) codeU dataset (c) ShareGPT dataset

Fig. 8: Yi-Coder-9B-Chat-8x-MoE (4 GPUs): Throughput (higher, better) and recomputed tokens (lower, better).



(a) NarrativeQA dataset (b) codeU dataset (c) ShareGPT dataset

Fig. 9: Yi-Coder-9B-Chat-8x-MoE model (4 GPUs): Prefill time (lower, better), and TPOT (lower, better).

Data Management for Agentic AI Workflows

TIDES: Reusable Agentic Memory

Two Types of Agentic Memory

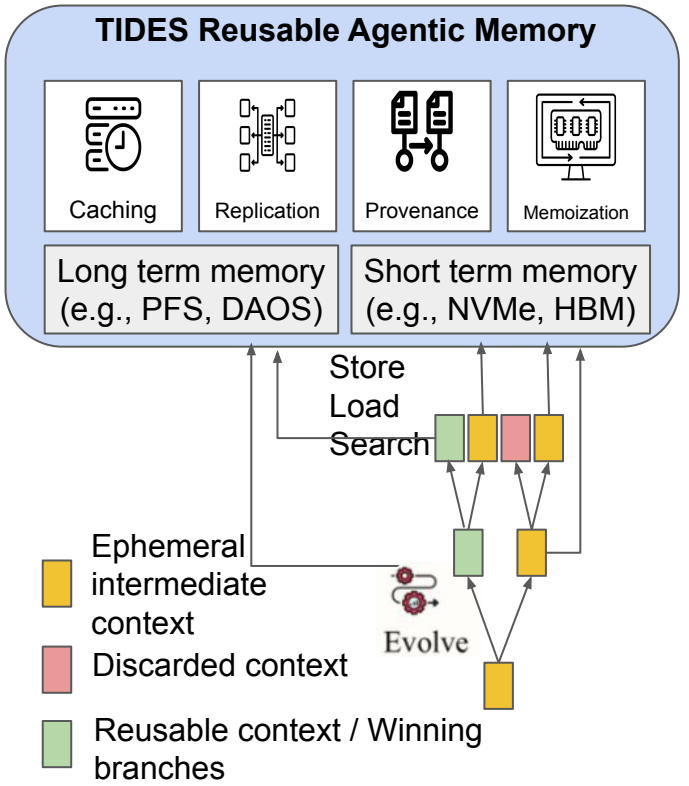
- Short-term: evolving context, ephemeral intermediate steps in a reasoning strategy (e.g., ToT nodes)
  - Metadata (e.g., scores), transformer outputs, cached results from tools
  - Typically “hot”: fast node-local tiers (GPU memory, host memory, NVMe)
- Long-term: persistent facts, key intermediate steps in past interactions reused across sessions, tasks, users
  - User prompts, valuable artifacts, winning reasoning branches/subtrees
  - Typically “cold”: durable but slower external repositories (PFS, DAOS)

State-of-the-Art

- Several popular approaches are described in the literature
- LangGraph (intermediate steps append/read/summarize flat memory)
- LlamaIndex (episodic + semantic memory fusion
- OpenAgents (memory graphs: objects → nodes, relationships → edges)

Key Advantages over State-of-Art

- Distributed Caching / Memoization: reuse past intermediate results to avoid duplication in RAG, inferences, tool invocations, overlapping reasoning paths
- Replication: improves data availability especially when agents live off-site
- Lineage of data: keeps track of provenance and data dependencies to enable roll back, branch off and to explain reasoning paths
- Scalable sharing and concurrency control to facilitate agentic collaboration



Integration of agentic memory with a ToT reasoning strategy: intermediate steps (evolving context, artifacts produced by agents) are stored to short term memory and discarded or committed to long-term memory if successful to improve future requests

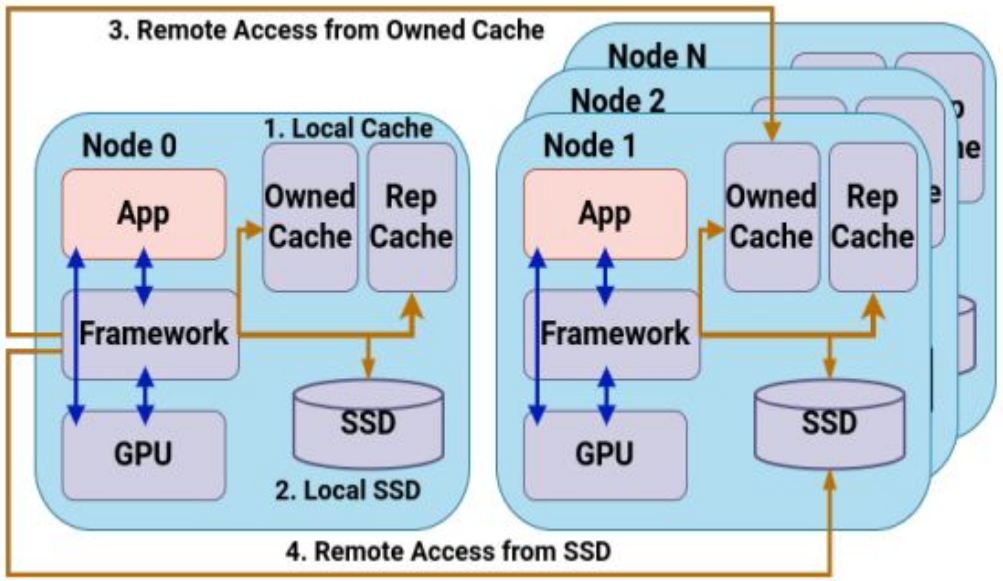
EvoStore: Lineage-Driven Tensor Repository

Challenges

- **Search queries:** What model in the repository is the best fit for specified features?
- **Ancestry queries:** How did the models derive from each other? What is the most recent ancestor of a frozen layer?
- **Fine-grain tensor-level access:** Efficient I/O of individual/groups of tensors
- **Space efficiency:** How do we apply incremental storage techniques at tensor-level for layers that were frozen during transfer learning?
- **Scalability:** Large number of workers evaluate model candidates in parallel, they need efficient access the repository under concurrency both in terms of metadata (queries) and data (tensors)

Our approach

- Lineage captures the evolution of transfer learning and addresses search queries and ancestry queries
- **Incremental storage** of modified parts of the data states enables space efficiency
- Decentralized data and metadata distributed enables **scalable queries** and **fine-grain access**
- **Popular tensor replication:** frequently accessed tensors are disseminated and reused close to the GPUs where they are needed

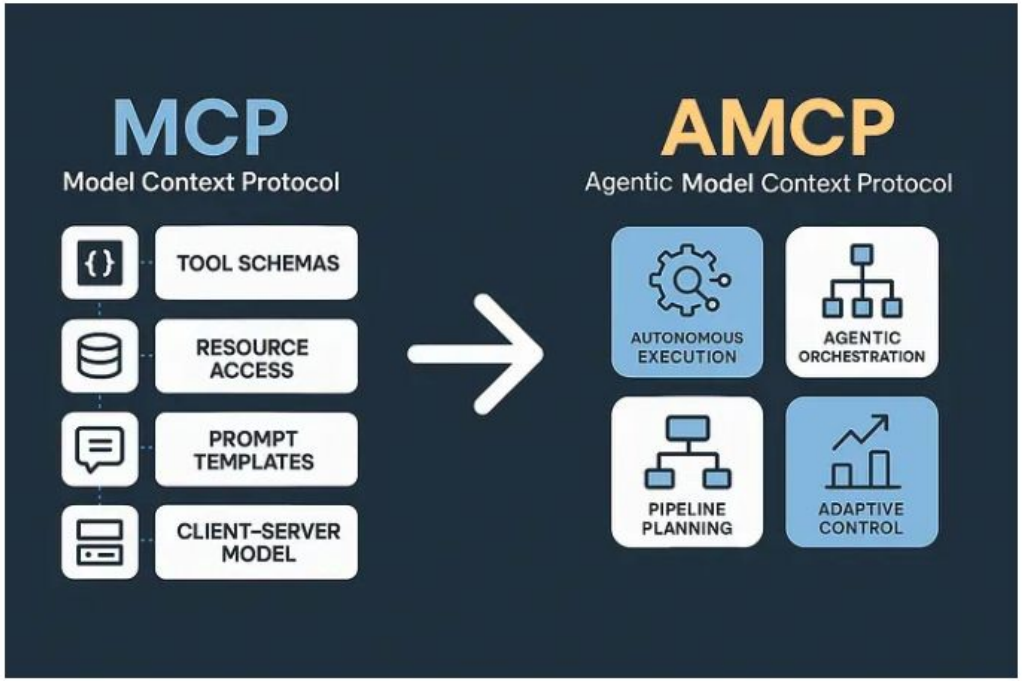


Underwood, R., Madhyastha, M., Burns, R. and Nicolae, B. EvoStore: Towards Scalable Storage of Evolving Learning Models. In HPDC'24 (Pisa, Italy, 2024).

Towards Resilient MCP for Tool Invocation

Scientific MCP Tools

- Long-running, may need checkpointing
- May stream partial results (need stateful RPCs)
- Alternatives: e.g., surrogate models



Resilient Orchestration:

- Many tools may need to run in parallel
- Automated decisions:
  - Do we need all results or is a subset enough?
  - On failure restart from checkpoint or switch to cheaper tool (and maybe less precise tool)?

- Need load balancing to avoid stragglers
- Need efficient async techniques to log interactions (MCP invocations, parameters) to handle restart

Acknowledgements

Co-authors and collaborators: *Avinash Maurya, Robert Underwood, Franck Cappello, Justin Wozniak, Rob Ross, Matthieu Dorier, Tom Peterka, Orçun Yıldız, Ian Foster, Kyle Chard, Tekin Bicer (ANL); Kathryn Mohror, Greg Kosinovsky (LLNL), Nigel Tan (Los Alamos), Michela Taufer (UTK), Mikaila Gossman, Jon Calhoun (Clemson), Jaiaid Mobin, Mustafa Rafique (RIT), Meghana Madhyastha, Randal Burns (JHU), Line Pouchard (BNL), Tanzima Islam (TSU), Jie Ye, Anthony Kougas, Xian-He Sun (IIT), A. Costan, G. Antoniu, E. Jeannot (INRIA), Kento Sato, Mohammed Wahib (RIKEN)*

Funding: *DataStates (DOE ECRP), RECUP (DOE ASCR), TIDES (DOE ASCR), DTIO (DOE ASCR), AI4S-TT(DOE ASCR), Diaspora (DOE ASCR), VLCC-States (NSF), MPI4AI (NSF)*

Conclusions

- From “traditional” deep learning to foundational models and agentic AI workflows, AI has rapidly transformed scientific applications and methodologies
- AI techniques needs to scale on large supercomputers and are co-designed with HPC systems research
- Data management is a key foundation for many aspects at the intersection of AI and HPC, especially modern agentic AI workflows
  - Checkpointing and Versioning
  - Offloading and Multi-Level Memory Management
  - KV Caching and Memoization
  - Multi-Modal Reusable Agentic Memory
  - Vector Databases
  - Data Pipelines and Preprocessing
- Many other exciting research directions, feel free to reach out!

Contact: bnicolae@anl.gov
Web: www.bnicolae.net