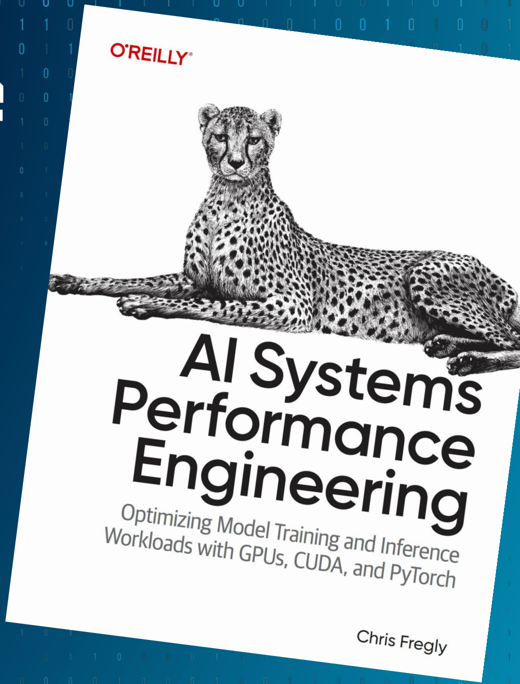
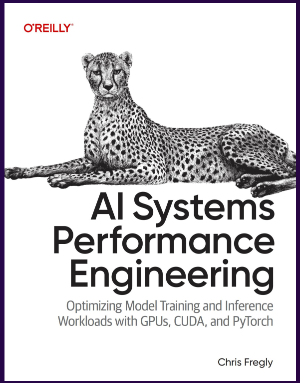

# Dynamic & Adaptive Inference Tuning (Reinforcement Learning & Other Methods)




***By Chris Fregly***

https://www.amazon.com/Systems-Performance-Engineering-Optimizing-Inference/dp/B0F47689K8/

# Why focus on inference tuning?




Faster time-to-response

Better end-user experience

Serve more users with same hardware

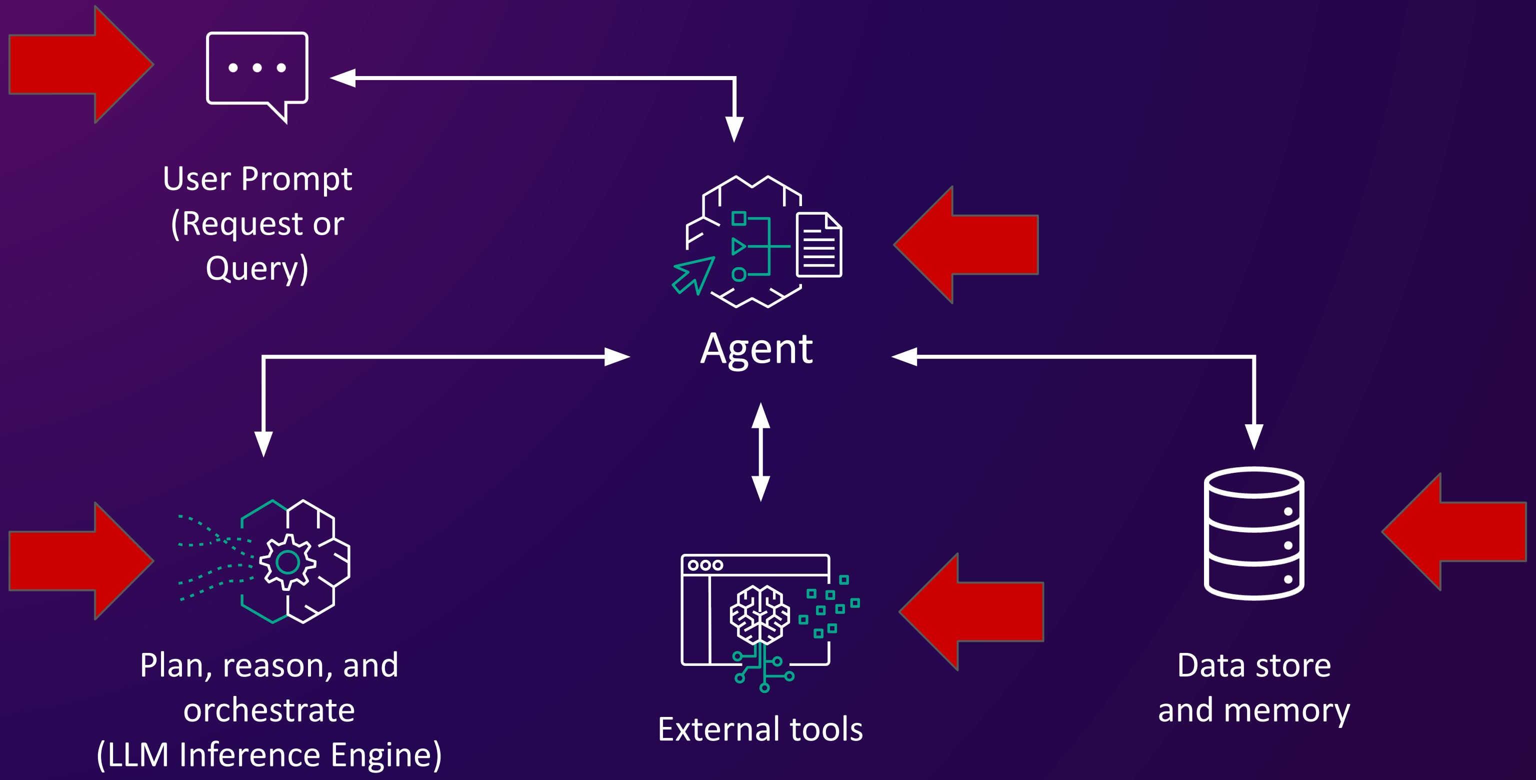
GPU cost savings ($$$ => $)

# “Mechanical sympathy”

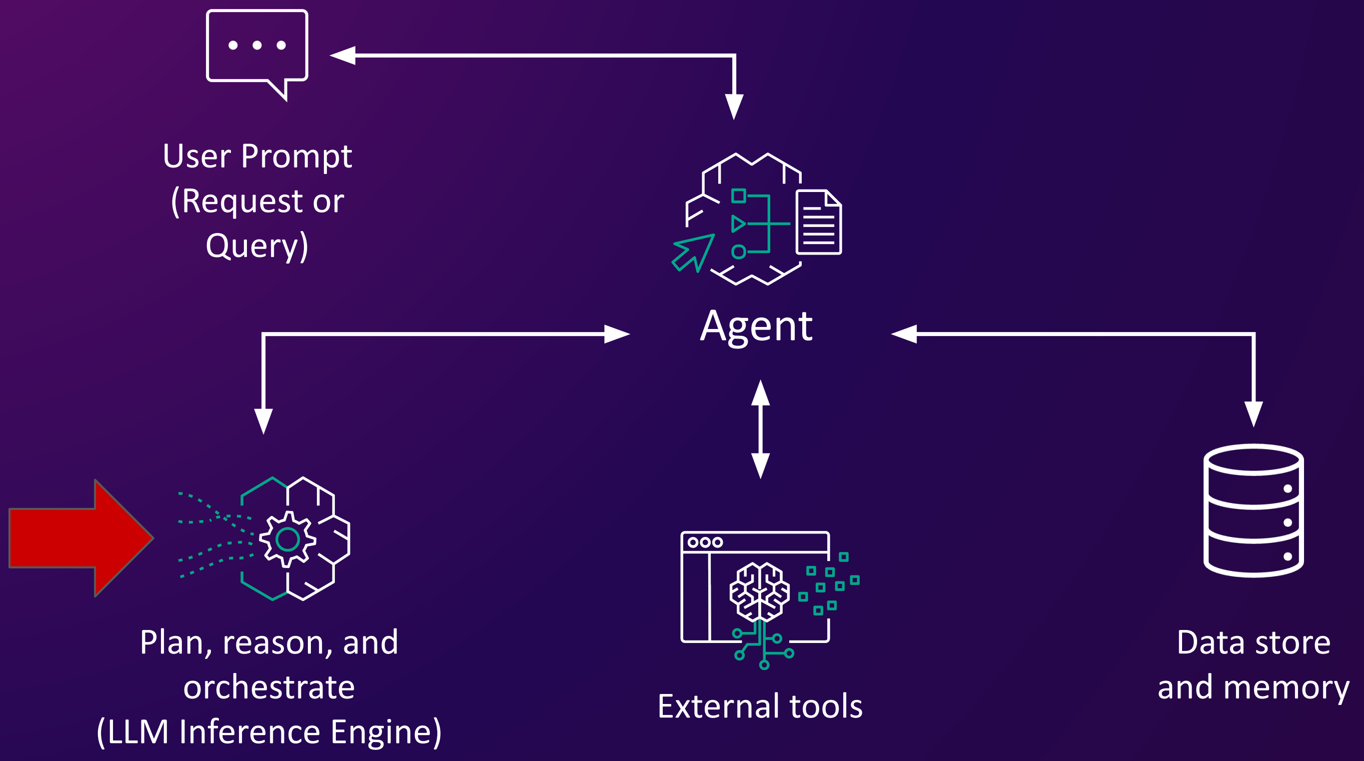
**Original reference**: Race car driver, Jackie Stewart, who was famously aware of the inner workings his race car.
*-- Martin Thompson (https://mechanical-sympathy.blogspot.com)*

**Today’s computing reference**: Co-designing software and algorithms hand-in-hand with hardware capabilities to maximize performance.

# Where are the bottlenecks?



# Where are the bottlenecks that *I will cover today?*



# Agenda

1. Dynamic and Adaptive AI System Optimizations

2. RL Agents for Production Systems Tuning and Operations

3. AI-Assisted Kernel and System Tuning

# Dynamic and Adaptive AI System Optimizations

**Adaptive Parallelism Switching** =>

| Inference Traffic Pattern | Recommended Parallelism | Rationale |
|---|---|---|
| Many short requests (<256 tokens, high QPS) | Data Parallel / Replica Scaling | Minimizes inter-GPU communications, each GPU runs replicas handling independent requests (assuming the model fits into a single GPU's memory) |
| Few long requests (≥8k tokens, low concurrency) | Pipeline Parallelism (with micro-batches) | Reduces per-request latency by splitting layers across GPUs |
| Mixed load (short + some long) | Hybrid Dynamic (auto-switching) | Runs small chats on single GPUs, pipelines long ones to meet latency SLAs |
| Extremely large model (>1 GPU memory) | Tensor + Pipeline Hybrid | Required to fit model; balances compute and memory across both dimensions |
| MoE model inference (sparse expert selection) | Expert Parallelism | Distributes individual experts across GPUs; each request only invokes a subset of experts, reducing per-device memory and compute load |

**Dynamic Precision Changes**

**Adaptive Batching and Scheduling (Chunked Prefill configuration, etc.)**

**Adjustable Speculative Decoding and KV-Cache Prefetching Configuration (Draft Model)**

**Occupancy-Aware CUDA Kernel Launch Parameters**

**Hot-Swappable CUDA Kernel Implementations** =>

Python
```python
import new_flash_attn_lib

# Monkey-patch the model's attention forward to use the new library
old_attn_forward = model.transformer.self_attn.forward

def new_attn_forward(self, *args, **kwargs):
    return new_flash_attn_lib.forward(*args, **kwargs)

model.transformer.self_attn.forward =
        new_attn_forward.__get__(
                model.transformer.self_attn,
                type(model.transformer.self_attn))
```

# RL Agents for Production Systems Tuning and Operations

**RL-based, Adaptive System Tuning =>**

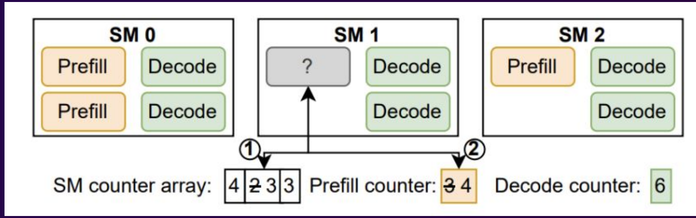
**Action 1:** choose parallelism mode: single, TP, PP, and hybrid
**Action 2:** choose precision: full FP8 versus mixing FP8 and FP4
**Action 3:** adjust batch size or batch-waiting time
**Action 4:** enable or disable cache compression
**Action 5:** enable or disable speculative decoding
**Action 6:** select a smaller draft model for speculative decoding
**Action 7:** select a larger draft model for speculative decoding
**Action 8:** enable or disable speculative kv prefetching

**Congestion-Aware Disaggregated Prefill-Decode Request Routing**

**RL Agent for Real-Time Monitoring, Ops, and Issue Resolution**

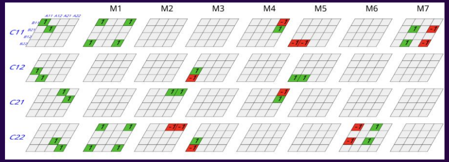
**RL-based Mixture-of-Experts (MoE) Expert Routing**

**GPU-aware Prefill-Decode Kernel Placement =>**

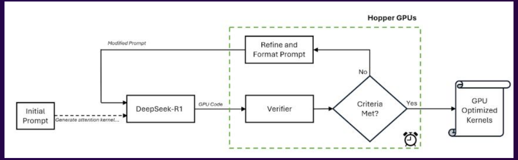


# AI-Assisted Kernel and System Tuning

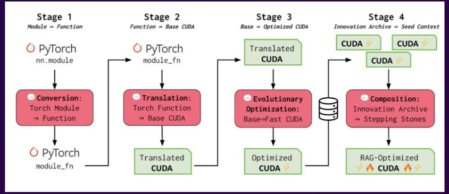
**Alpha Tensor Project (Google DeepMind).** Rediscovered Strassen’s sub-quadratic GEMM algorithm. Optimized CUDA kernel (better than NVIDIA).




**Optimize Attention CUDA kernel (NVIDIA).** Used DeepSeek-R1 reasoning model to create optimized Attention CUDA kernel implementation.




**“AI CUDA Engineer” (Sakana.ai).** Evolutionary strategies to iteratively refind CUDA kernel code. Optimized CUDA kernel (5x NVIDIA performance). Also, was able to process 230 of 250 primitive PyTorch ops.



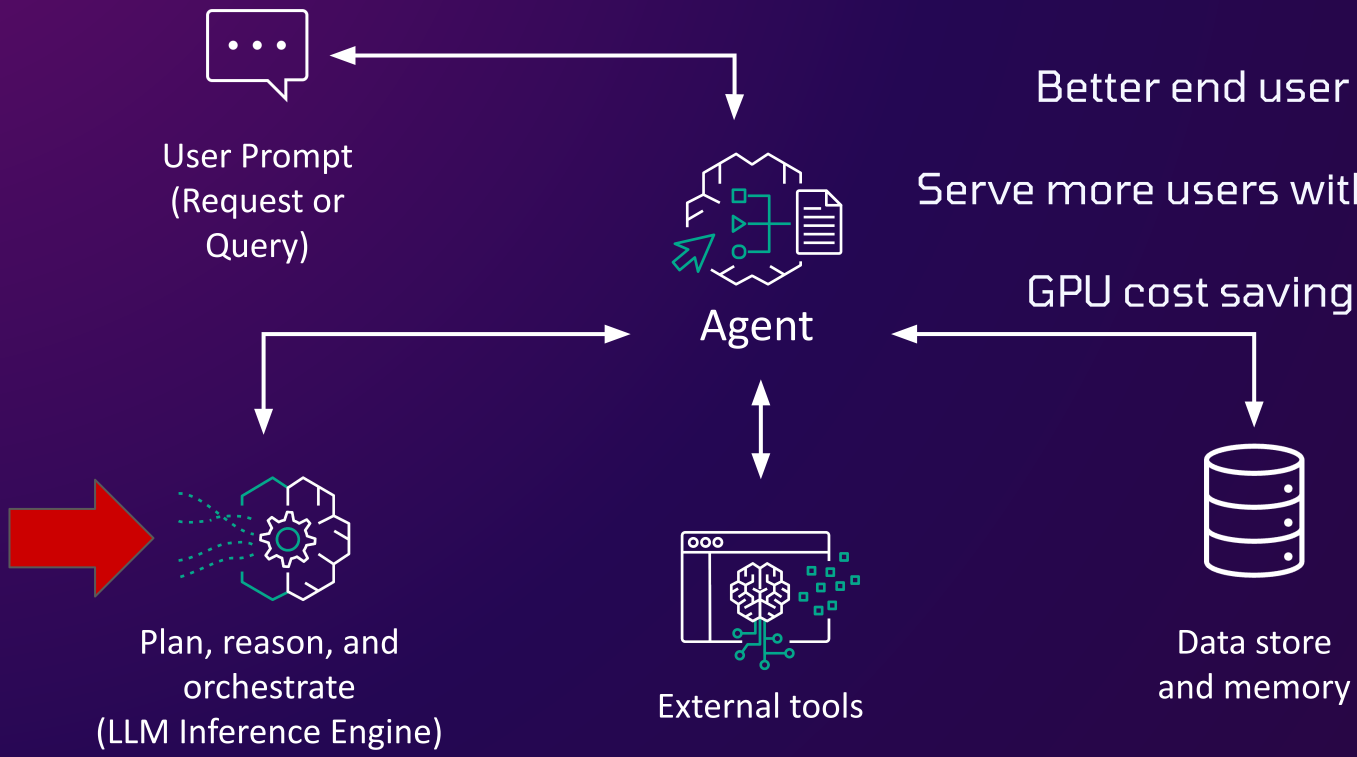
# Embrace Adaptive, AI-Assisted System Optimizations

Faster time-to-response

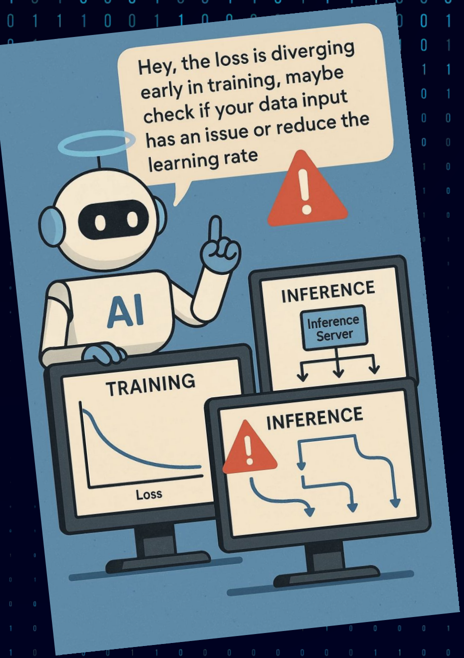
Better end user experience

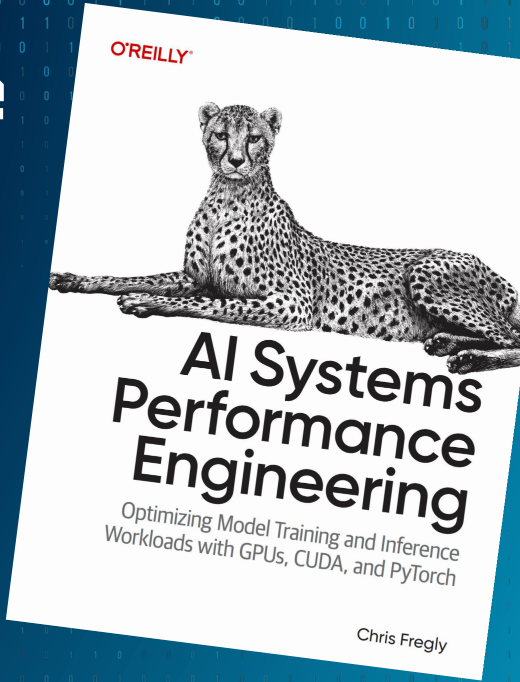
Serve more users with same hardware

GPU cost savings ($$$ => $)



# Thank You!
# Dynamic & Adaptive Inference Tuning (Reinforcement Learning & Other Methods)




## *By Chris Fregly*

https://www.amazon.com/Systems-Performance-Engineering-Optimizing-Inference/dp/B0F47689K8/