Optimizing Deployment Configurations for LLM Inference

Retrospective Analysis on Challenges and Insights

Sung Min Cho / Meta

Meta

LLM Inference Is Extremely Expensive

Metric (per query)	Llama3 70B (Dense)	Recommendation Models	Gap
Workload	2K prompt, 150 output, Batch Size 64	Rank 4K items	-
FLOPs	8 EFLOPs+	< 2.5 TFLOPs	**3,000x+**
Memory traffic	10 TB+	< 400 GB	**25x+**
System time	Few seconds	< 100 ms	**30x+**

Llama served nearly **1 billion** monthly active users at Meta. System design was a business-critical decision.

Even small deployment inefficiencies translated to enormous cost.

Why Is Deployment Configuration So Hard?

1. Workload Challenges

Diverse applications with vastly different SLOs, input/output profiles, and model sizes

- SLO: Online vs. Offline
- Input: 50 – 32K+ tokens
- Output: 100 - 1K+ tokens
- Model size: 10B – 500B+ params

2. Infrastructure Challenges

Hardware platforms, runtime architectures, and parallelism strategies — each with trade-offs

- Scale-out, scale-up
- Parallelisms (TP, PP, EP, CP, DP)
- Continuous vs. Disagg

3. Optimization Challenges

Rapidly evolving techniques alter performance profiles, requiring constant re-evaluation

- Architectural (MoE, Mamba, ...)
- Compression (Quantization, Distillation, ...)
- System (KV cache, Speculative, ...)

Manual configuration is no longer feasible given the combinatorial design space.

Workload Challenges

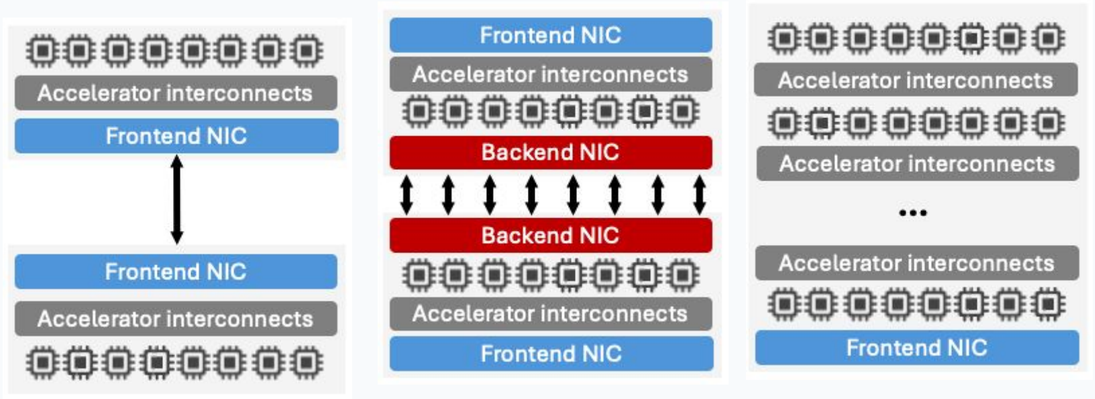
Service	Input	Output	TTFT	TTIT	Model Size
Text Agent	500-2K	100-500	200-500ms	<40ms	10-500B
Multi-modal Agent	MM+500-2K	100-500	300-800ms	<40ms	10-500B
Real-time Agent	<50	50-100	<200ms	<30ms	10-100B
Safety / Query Gen	<8K	Few / 100-500	<200ms	<10ms	10B
Data Augmentation	<8K	<1K	Few sec	<300ms	100-500B
Data Scoring	<32K	Few	N/A	N/A	100-500B

Caveat: These are just retrospective examples

Each workload demands a fundamentally different system configuration.

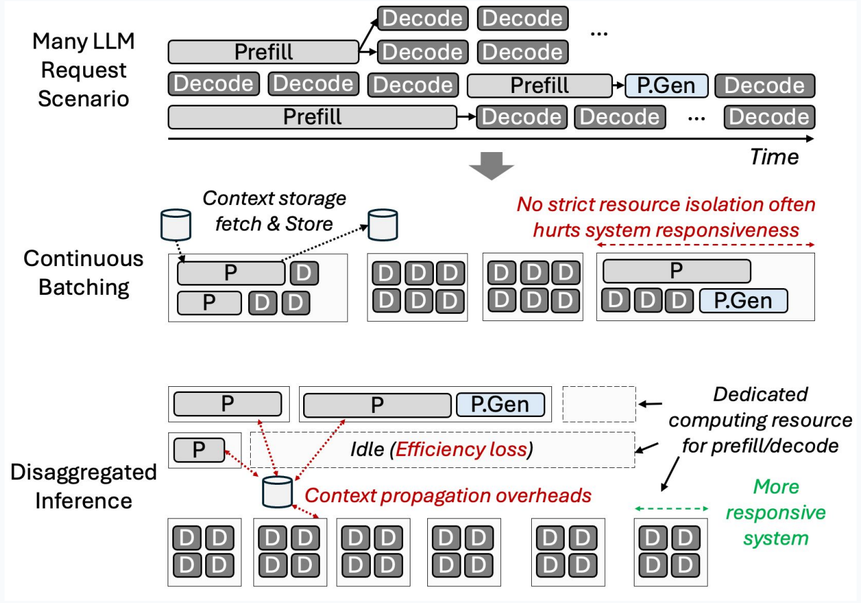
Infrastructure Challenges

Hardware Platform Types



- Weak scale-out (PCIe connected)
- Strong scale-out (NVLink intra-host)
- Scale-up (custom high-BW fabric)
- Wrong choice → 2-3x cost inefficiency

Runtime Architecture



Continuous batching: mixes prefill + decode

- Simple, but couples different phases

Disaggregated: separate prefill/decode pools

- Enables phase-specific optimization

Infrastructure Challenges

Tensor (TP) — Shard params across devices — low latency, needs high-BW

Pipeline (PP) — Layer chunks sequentially — good throughput, moderate comm.

Expert (EP) — Distribute MoE experts — All2All comm.

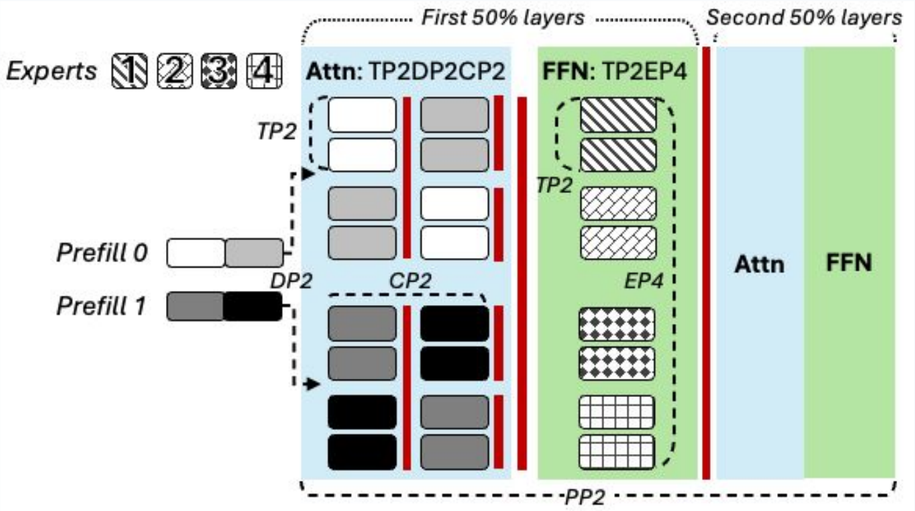
Context (CP) — Partition long sequences across devices.

Data (DP) — Replicate weights for independent batches.

5D Parallelism Trade-offs

Parallelism	Memory capacity		Latency	Throughput	Sync Overhead
	Weight	KV cache			
Tensor	1/N	1/N	1/N	N	High (AllReduce)
Pipeline	1/N	1/N	1	N	Low (Send/Recv)
Expert	1/N	N/A	1/N	N	Med (All2All)
Context	1/N	1/N	1/N	N	Med (Send/Recv)
Data	1	1	1	N	1

Example: TP2DP2CP2 (Attn) + TP2EP4 (FFN)



Navigating Millions of Configurations

Runtime Configurations

- Several workload targets
- Several models and hardware platforms
- Dozens of batch sizes
- Several optimization techniques
- Different execution phases (prefill/decode)

~1,000+ Combinations

Parallelizations

- 5D parallelism
- Separate parallelism per each module
- Scaling up to ~256 accelerators

~1,000+ Combinations

Total Search Space

The total number of combinations to explore often reaches the order of

~1M+ Combinations

Manual or heuristic approaches cannot navigate this scale

Lightweight Performance Simulator

Projection Framework

1. Inputs

Configurations

- Model Architecture
- Hardware Specs
- Runtime Settings
- SLOs & Requirements

2. Benchmarking

Performance Database

- Micro-benchmark Data
- Op/Module Perf DB
- Production Samples

3. Estimation

Performance Estimator

- Model Partitioning
- Operator Perf Model
- Graph Execution Simulator

4. Optimization

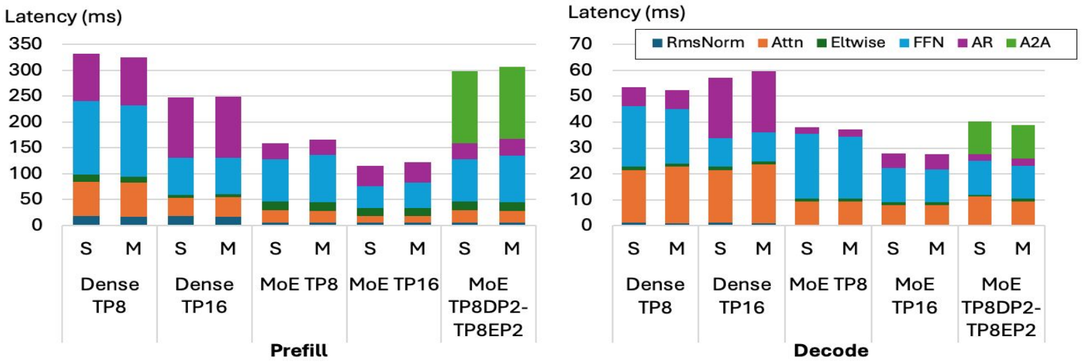
Objective: Max Throughput

- TTFT/TTIT estimation
- QPS Estimation
- Constraint Matching

Performance Estimator Details

- Micro-benchmark operators on each HW platform
- 100K+ measurements per platform
- Build piecewise-linear interpolation models
- Assemble into E2E latency per configuration
- Sweep millions of configs in minutes
- **Accuracy: ±5%** vs real hardware

Validation: Simulation (S) vs Measurement (M)



Insight 1: Disaggregated Runtime Wins for Online Inference

Online Inference (Strict SLOs)

Model	HW	Runtime	Dec. BS	QPS
70B	GPU-A	Disagg	112	356
70B	GPU-A	Cont.Batch	28	198
405B	GPU-B	Disagg	56	67
405B	GPU-B	Cont.Batch	8	37

Online Optimal Setups (selected results from paper)

- **Disagg:** 1.5–1.8x higher QPS (70B), 1.8–2.2x (405B)
- Independent prefill/decode optimization
- Much larger decode batches (e.g., 112 vs 28)

Offline Inference (Throughput-Only)

Model	HW	Runtime	Dec. BS	QPS
70B	GPU-A	Disagg	896	143
70B	GPU-A	Cont.Batch	512	143
70B	GPU-C	Disagg	2560	193
70B	GPU-C	Cont.Batch	1024	197

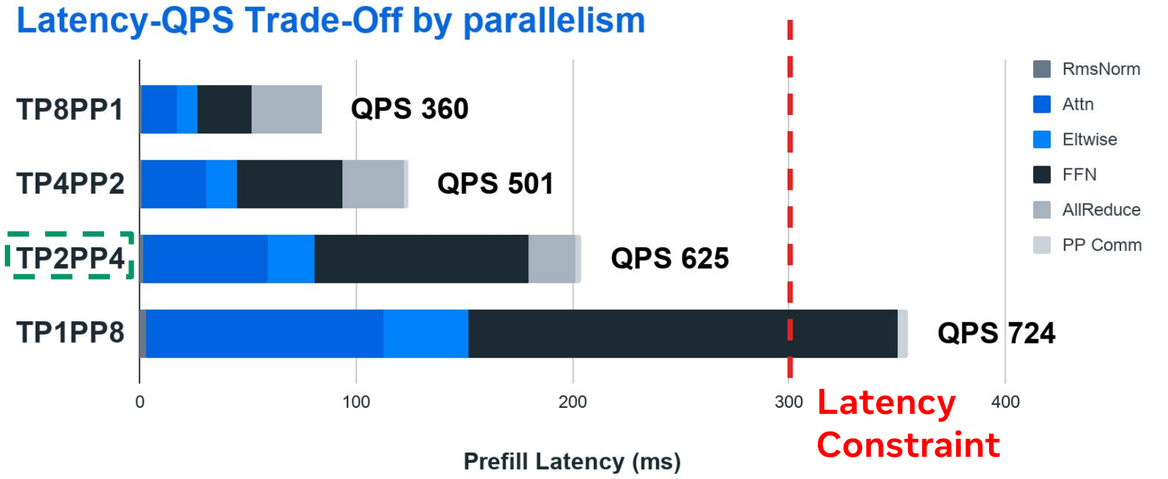
Offline Optimal Setups (selected results from paper)

- Gap disappears for offline workloads
- Both converge to deep PP + large batches
- Cont.batch simpler operationally

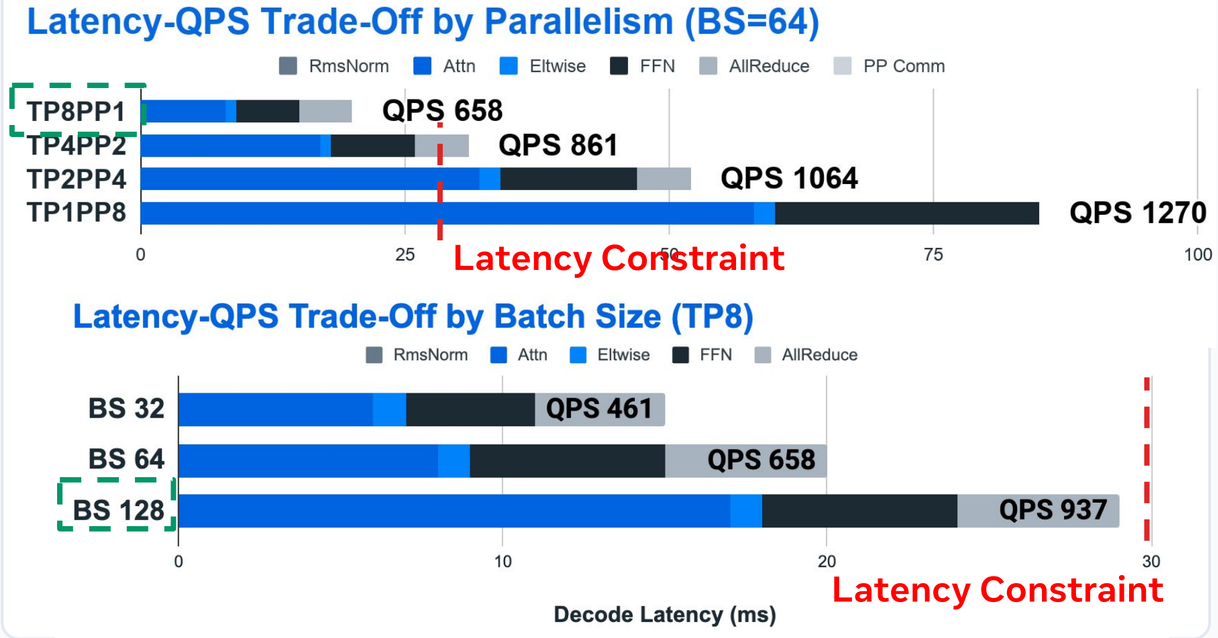
Impact: ~30% capacity savings by migrating online services to disaggregated runtime

Insight 2: Parallelism Must Be Tuned Per Phase

Prefill best setup: TP4PP2 BS=1



Decode best configuration: TP8PP1 BS128



- Prefill adopts PP to minimize Comm overheads and maximize throughput
- Decode cannot afford PP due to latency constraints, but leverages batching to enhance throughput
- Highlights another benefit of disaggregated setup: Cont. batching cannot leverage this traits

Impact: Phase-specific parallelization improved throughput at given SLO constraints

Insight 3: Heterogeneous Hardware for Cost Efficiency

Heterogeneous Inference for 405B Model

GPU	Strength	Relative Cost
GPU-A	High FLOPs	$$
GPU-B	High FLOPs + BW	$$$
GPU-C	High HBM BW	$

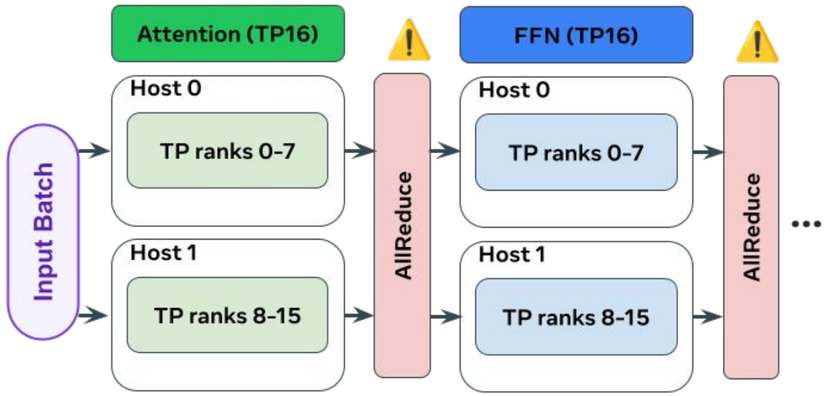
Prefill HW	Decode HW	Prefill QPS	Decode QPS	E2E QPS	Cost
GPU-A	GPU-A	**88**	**77**	**41**	**$$**
GPU-B	GPU-B	**88**	276	**67**	**$$$**
GPU-C	GPU-C	**85**	**277**	65	**$**
GPU-A *	**GPU-B ***	**88**	276	**67**	$$~$$$
GPU-A *	**GPU-C ***	**88**	**277**	**67**	**$~$$**

- ★ **Heterogeneous configs** match best homogeneous QPS with less TCO
- Estimated **15–25% TCO savings**

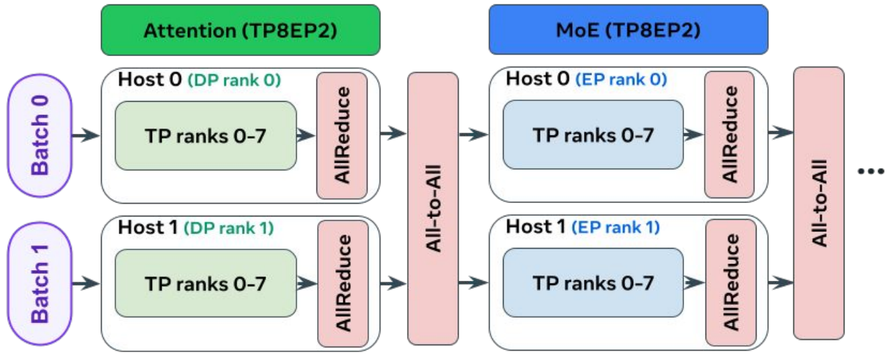
Impact: Deployed services to platforms that best fit the characteristics to maximize the fleet efficiency

Insight 4: Expert Parallelism Makes MoE Scaling Efficient

Dense (TP16) Architecture

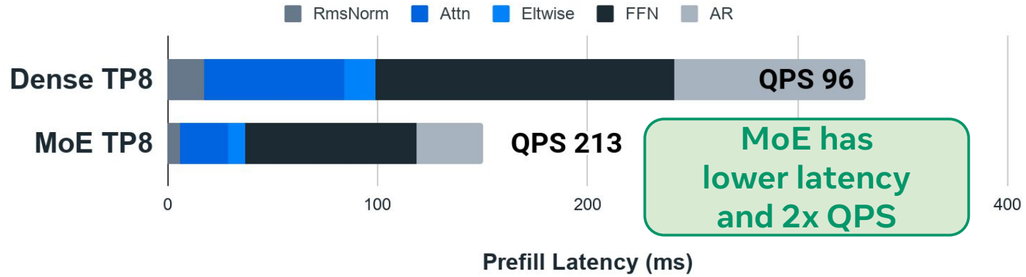


MoE (TP8DP2-TP8EP2) Architecture

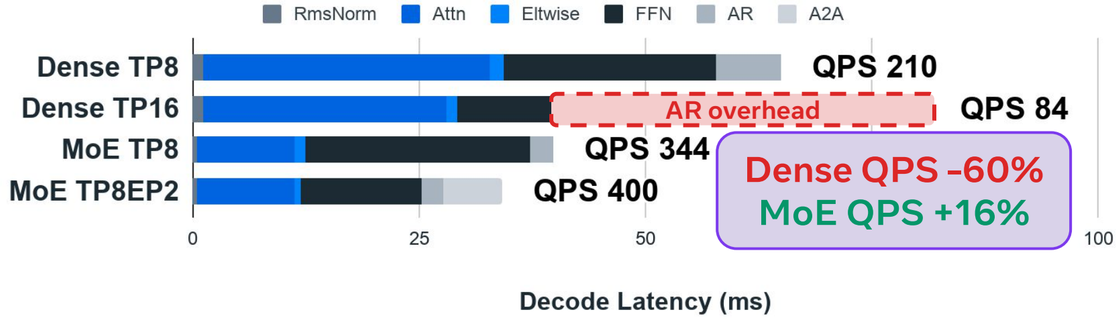


405B Dense vs MoE (iso-parameter)

Dense vs MoE (Prefill TP8)



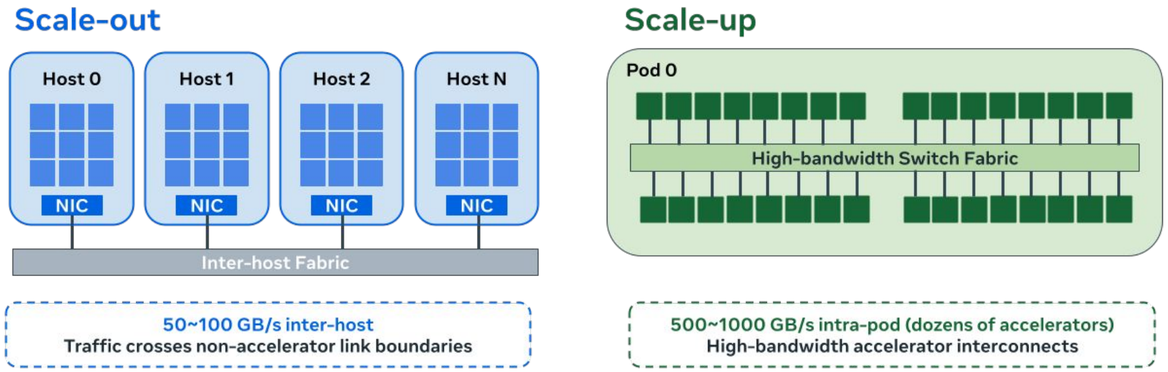
Dense vs MoE (Decode Scaling)



Impact: EP-based scaling informed the shift toward MoE architectures for model scaling

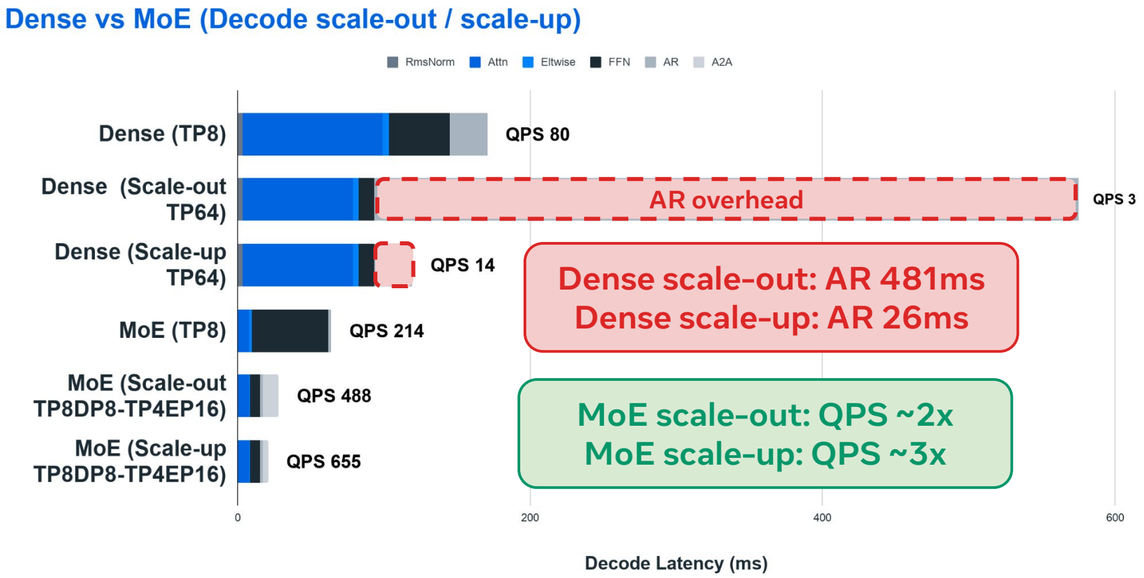
Insight 5: Scale-out vs Scale-up Platform Trade-offs

Scale-out vs Scale-up



- Which scenario justifies **scale-up cost**?

1.8T Dense vs MoE (iso-parameter)



- Dense models require scale-up: scale-out TP is bottlenecked by inter-host AllReduce
- MoE models scale better on scale-out via EP; but scale-up still maximizes performance
- Right platform choice depends on interplay of model architecture, size, SLOs, parallelism, etc.

Impact: Simulation-based projections enabled data-driven infrastructure planning ahead of HW availability

Conclusion & Key Takeaways

01 **Disaggregated Runtimes:** 1.5–2.2x throughput for online; both converge for offline

02 **Phase-Specific Parallelism:** Prefill and decode require fundamentally different strategies

03 **Hardware Heterogeneity:** Matching phases to accelerator strengths saves 15–25% TCO

04 **Expert Parallelism:** EP makes MoE a compelling architecture for model scaling

05 **Platform Choice:** Scale-out vs. scale-up depends on model architecture, size, and SLOs


Meta