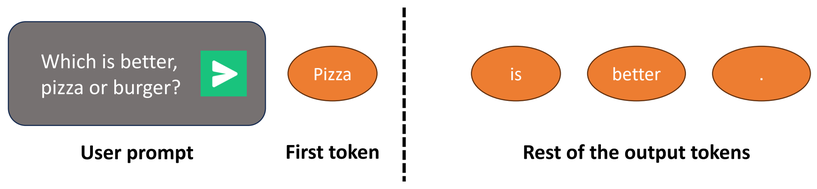
Splitwise: Efficient Generative LLM Inference Using Phase Splitting

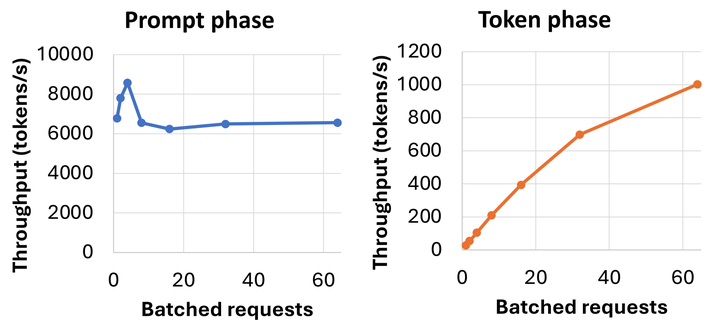
Each LLM inference request has two distinct phases with different resource requirements

Prompt computation and token generation phases

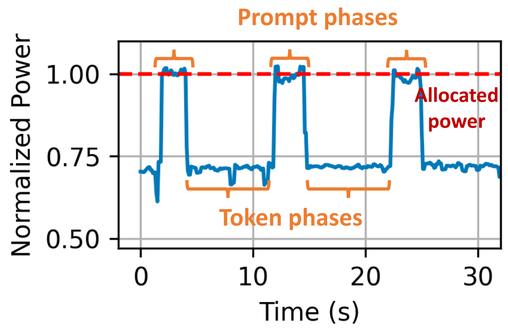


Prompt phase	Token phase
User input processed in parallel	Serialized token generation
Compute and power intensive	Memory intensive (relies on KV cache)
Limited batching benefits	Batching improves throughput

Example 1: Batching effects

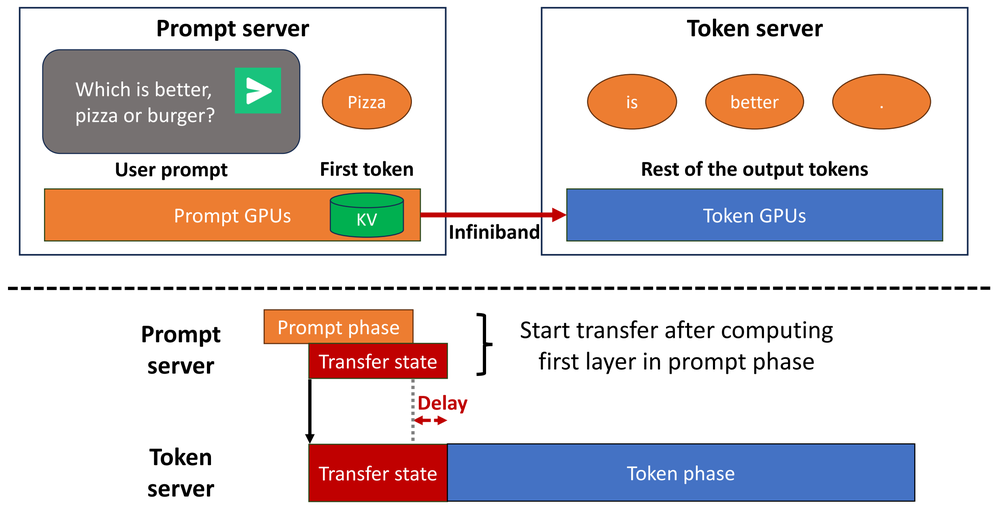


Example 2: Power usage

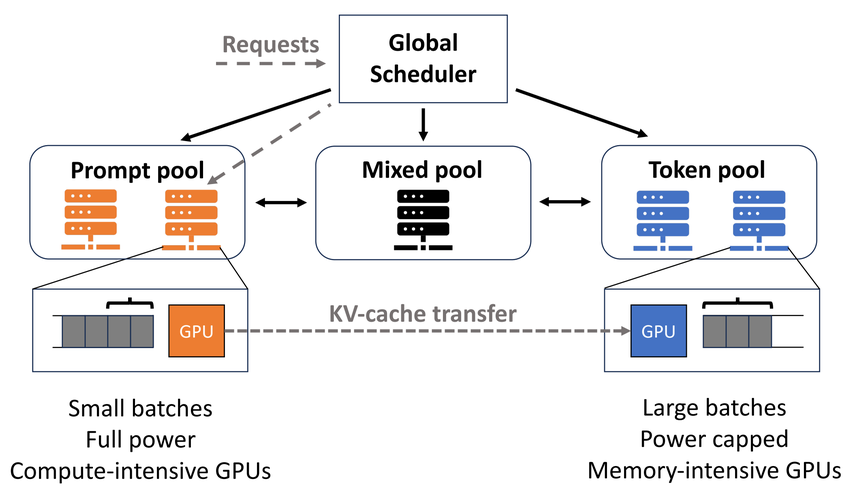


Splitwise splits inference across different servers to enable phase-specific resource management

1 Transfer request state over P2P GPU Infiniband; optimize with parallel and overlapped transfers

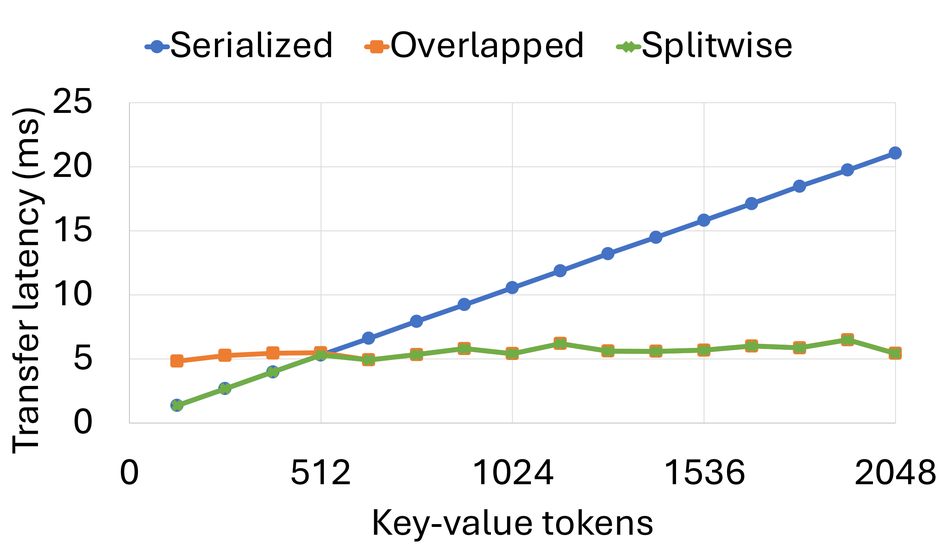


2 Split cluster into three server pools and use phase-specific resource management at scale



Splitwise clusters are much more resource efficient than existing clusters

Result 1: Splitwise transfers request state with less than ~0.8% end-to-end overhead on average



Result 2: Clusters designed using Splitwise provide much higher throughput than existing clusters

Throughput optimized clusters	Baseline (A100 x70)	Splitwise homogeneous (A100 x45 → A100 x25)	Splitwise heterogeneous (H100 x25 → A100 x26)
#Servers	1x	1x	0.73x
Cost	1x	1x	1.14x
Power	1x	1x	1x
Throughput	1x	2.4x	2.6x

Paper, code, traces at aka.ms/splitwise

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, Ricardo Bianchini