Splitwise
Efficient Generative LLM Inference Using Phase Splitting

Pratyush Patel

Esha Choukse  Chaojie Zhang  Aashaka Shah  Íñigo Goiri
Saeed Maleki  Ricardo Bianchini
Microsoft places huge cap-ex bets on datacenters for cloud and AI

Google Cloud braces for AI compute costs, ramps up data center investment

Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs

In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.
LLM clusters are very expensive and power hungry

https://cloud-gpus.com/
Inference demand far outweighs that of training
Inference demand far outweighs that of training

75-90% of the compute demand at scale
Splitwise optimizes LLM serving at scale

Characterize generative LLM inference and identify distinct prompt and token phases

Split inference onto different servers for phase-specific resource management

Design clusters using Splitwise, which improves efficiency across various metrics
Anatomy of a generative LLM inference
Anatomy of a generative LLM inference
User submits a prompt to the LLM

User

User prompt

Which is better, pizza or burger?

GPT-4 model
Forward pass 1:
LLM processes the prompt to generate first output token

User

User prompt

GPT-4 model

Which is better, pizza or burger?

Pizza

KV cache

Which is better, pizza or burger?
Forward pass 1:
LLM processes the prompt to generate first output token

User prompt: Which is better, pizza or burger?

Generated output tokens:
Pizza

User

Diagram:
- GPT-4 model
- KV cache
- Pizza

Note: The diagram shows the flow of information from the user prompt to the generated output tokens, with the GPT-4 model and KV cache as intermediaries.
Forward pass 2:
LLM generates next token using KV cache and previous token

User prompt: Which is better, pizza or burger?

Generated output tokens:
Pizza

Diagram:
- User
- User prompt: Which is better, pizza or burger?
- GPT-4 model
- KV cache
  - Pizza
- Pizza
Forward pass 2:
LLM generates next token using KV cache and previous token

User
User prompt
Which is better, pizza or burger?

GPT-4 model

KV cache
is

Pizza
is

Generated output tokens
Forward pass 3:
LLM generates next token using KV cache and previous token

User

Which is better, pizza or burger?

User prompt

Pizza

is

Generated output tokens
Forward pass 3:
LLM generates next token using KV cache and previous token

User prompt: Which is better, pizza or burger?

Generated output tokens: Pizza is better
Forward pass 4:
LLM generates next token using KV cache and previous token

Which is better, pizza or burger?
Forward pass 4:
LLM generates next token using KV cache and previous token

GPT-4 model

KV cache

EOS

User

User prompt

Which is better, pizza or burger?

Pizza

is

better

.
Latency metrics for LLM inference

User prompt: Which is better, pizza or burger?

Time to first token (TTFT)

Time between tokens (TBT)

Generated output tokens: Pizza is better.

End-to-end response time
Prompt computation vs. token generation

Which is better, pizza or burger?

User prompt: Pizza

First token: is

Rest of the output tokens: better

<table>
<thead>
<tr>
<th>Prompt phase</th>
<th>Token phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>User input processed in parallel</td>
<td>Serialized token generation</td>
</tr>
<tr>
<td>Compute intensive</td>
<td>Memory intensive (relies on KV cache)</td>
</tr>
</tbody>
</table>

Pizza is better.
Prompt phases hit a throughput bottleneck

Profiled for BLOOM-176B on vLLM with 512 input size per request
Token phase batches are memory constrained

Token phases use the KV cache, which can take up hundreds of GBs!

Profiled for BLOOM-176B on 8 A100 GPUs
Prompt phases are power intensive

3x BLOOM-176B inference requests on 8 GPUs
Prompt computation vs. token generation

<table>
<thead>
<tr>
<th>Prompt phase</th>
<th>Token phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute and power intensive</td>
<td>Memory intensive</td>
</tr>
<tr>
<td>Limited batching benefits</td>
<td>Batching improves throughput</td>
</tr>
</tbody>
</table>

User prompt: Which is better, pizza or burger?
First token: Pizza
Rest of the output tokens: is better.

Pizza is better.
Inefficient to run both on the same hardware

User prompt: Which is better, pizza or burger?
First token: Pizza
Rest of the output tokens: is better.

Server
GPUs
Splitwise splits phases onto different servers

Which is better, pizza or burger? 
Pizza

GPUs

User prompt  First token

Prompt server
Small batches, maximum power

is  better

Rest of the output tokens

Token server
Large batches, power capped

GPUs
Different trade-offs on different GPUs

<table>
<thead>
<tr>
<th>Spec</th>
<th>H100 : A100 ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2.15x</td>
</tr>
<tr>
<td>Max. power</td>
<td>1.75x</td>
</tr>
<tr>
<td>TFLOPs</td>
<td>3.43x</td>
</tr>
</tbody>
</table>
GPU memory scales slower than compute

<table>
<thead>
<tr>
<th>Spec</th>
<th>H100 : A100 ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2.15x</td>
</tr>
<tr>
<td>Max. power</td>
<td>1.75x</td>
</tr>
<tr>
<td>TFLOPs</td>
<td>3.43x</td>
</tr>
<tr>
<td>HBM capacity</td>
<td>1.00x</td>
</tr>
<tr>
<td>HBM bandwidth</td>
<td>1.64x</td>
</tr>
</tbody>
</table>
Phase preference for different GPUs

For BLOOM-176B on the mean request size in production

A100 is better

H100 is better
Splitwise splits phases onto different servers

Which is better, pizza or burger?

Pizza

User prompt

First token

is better

Prompt server

Rest of the output tokens

Token server
Each phase could use preferred hardware

Which is better, pizza or burger?

Pizza

Prompt GPUs

User prompt

First token

Prompt server

is better

Token GPUs

Rest of the output tokens

Token server

Pizza is better.
Splitting inference requires fast state transfers

Which is better, pizza or burger?

Pizza

User prompt  First token

is  better

Rest of the output tokens

Prompt GPUs  KV

Token GPUs

Prompt server  Token server
Splitwise uses GPU Infiniband to ship state

Which is better, pizza or burger?

Pizza

First token

is better

Prompt GPUs

Rest of the output tokens

KV

Token GPUs

Prompt server

Token server

User prompt
Splitwise uses GPU Infiniband to ship state

Which is better, pizza or burger?

User prompt

First token

Pizza

Prompt GPUs

Prompt server

GPU IB

Rest of the output tokens

is

better

.

Token GPUs

Token server
Parallelize transfers for high bandwidth

Assuming 4-way tensor parallelism

Which is better, pizza or burger?

Pizza is better.

User prompt
First token

GPU
GPU
GPU
GPU

Prompt server

: 1/4\textsuperscript{th} of the KV cache

25GB/s

Rest of the output tokens

GPU
GPU
GPU
GPU

Token server

Total bandwidth: 100GB/s
Transfer overheads may still be large

KV cache sizes can be hundreds of GBs!
Splitwise overlaps transfer with prompt phase

Start shipping the KV-cache after the first prompt layer
Splitwise adds very little latency overhead

Less than ~0.8% overhead for a typical inference request

Implemented in vLLM
Splitwise: Phase Splitting for Generative LLMs

Characterize generative LLM inference

Split inference onto different servers

Design clusters using Splitwise
Splitwise partitions servers into three pools

- **Prompt pool**
- **Mixed pool**
- **Token pool**

Details in the paper
Servers are fungible across the pools

Details in the paper
Scheduler decides how to split LLM requests

Details in the paper
Servers implement phase-aware batching
Evaluation compares different cluster designs

Optimize for different metrics on two production traces

Traces available at: https://github.com/Azure/AzurePublicDataset
Simulated at scale using performance profiles

App. inputs → Hardware profiles → Simulated cluster

- Prompt sizes
- Output sizes
- Service-Level Objectives

- LLM performance
- KV-cache transfer

Global scheduler

Request trace

State transfer
**Baselines**

Run requests end-to-end on same server

- A100
- H100

**Splitwise homogeneous**

Use the same server type for prompt and token phases

- Prompt: A100
- Token: A100

**Splitwise heterogeneous**

Use H100s for prompt and A100s for token phases

- Prompt: H100
- Token: A100
Baseline
Run requests end-to-end on same server

Splitwise homogeneous
Use the same server type for prompt and token phases

Splitwise heterogeneous
Use H100s for prompt and A100s for token phases

More results in the paper
<table>
<thead>
<tr>
<th>Throughput optimized clusters</th>
<th>Baseline</th>
<th>Splitwise homogeneous</th>
<th>Splitwise heterogeneous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A100 x70</td>
<td>A100 x45  A100 x25</td>
<td>H100 x25  A100 x26</td>
</tr>
<tr>
<td>#Servers</td>
<td>1x</td>
<td>1x</td>
<td>0.73x</td>
</tr>
<tr>
<td>Cost</td>
<td>1x</td>
<td>1x</td>
<td>1.14x</td>
</tr>
<tr>
<td>Power</td>
<td>1x</td>
<td>1x</td>
<td>1x</td>
</tr>
<tr>
<td>Throughput</td>
<td>1x</td>
<td>2.4x</td>
<td>2.6x</td>
</tr>
<tr>
<td>Cost optimized clusters</td>
<td>Baseline</td>
<td>Splitwise homogeneous</td>
<td>Splitwise heterogeneous</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
<td>-----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>#Servers</td>
<td>1x</td>
<td>0.46x</td>
<td>0.34x</td>
</tr>
<tr>
<td>Cost</td>
<td>1x</td>
<td>0.46x</td>
<td>0.48x</td>
</tr>
<tr>
<td>Power</td>
<td>1x</td>
<td>0.46x</td>
<td>0.43x</td>
</tr>
<tr>
<td>Throughput</td>
<td>1x</td>
<td>1x</td>
<td>1x</td>
</tr>
</tbody>
</table>
LLM inference requests have distinct prompt and token phases.

Splitting inference enables phase-specific resource management.

Splitwise improves inference cluster efficiency across various metrics.

Thanks!
pratyush@cs.uw.edu

Paper, code, traces at aka.ms/splitwise