Efficiently Scaling Transformer Inference

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury,
Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, Jeff Dean
Google
MLSys 2023 (Outstanding Paper Award)

Presented by Vignesh Suresh
Feb 27, 2024

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Goal of the work

Inference

- How to reduce latency for prefill and decode?

Transformer

- How to partition compute and memory?

Scaling

- How to scale to large batch sizes and sequences?

Efficiently

- How to ensure low chip cost and high utilization?

Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

Partitioning attention

Results from PaLM

Comparison with FasterTransformer

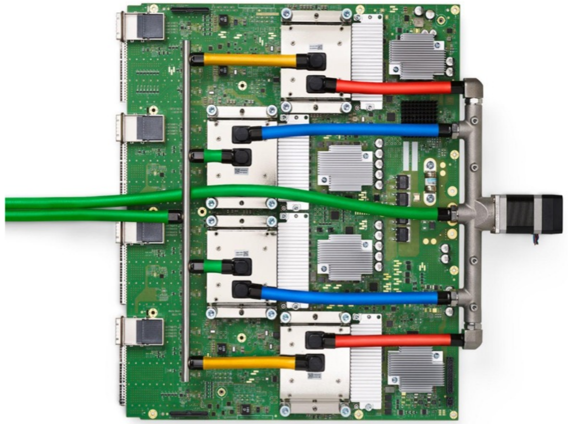
Discussion

Preliminaries

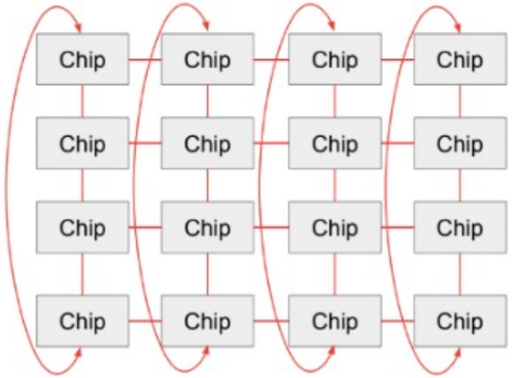
Key metrics for transfer inference

- Latency
- Throughput
- Model FLOPs utilization

System setup



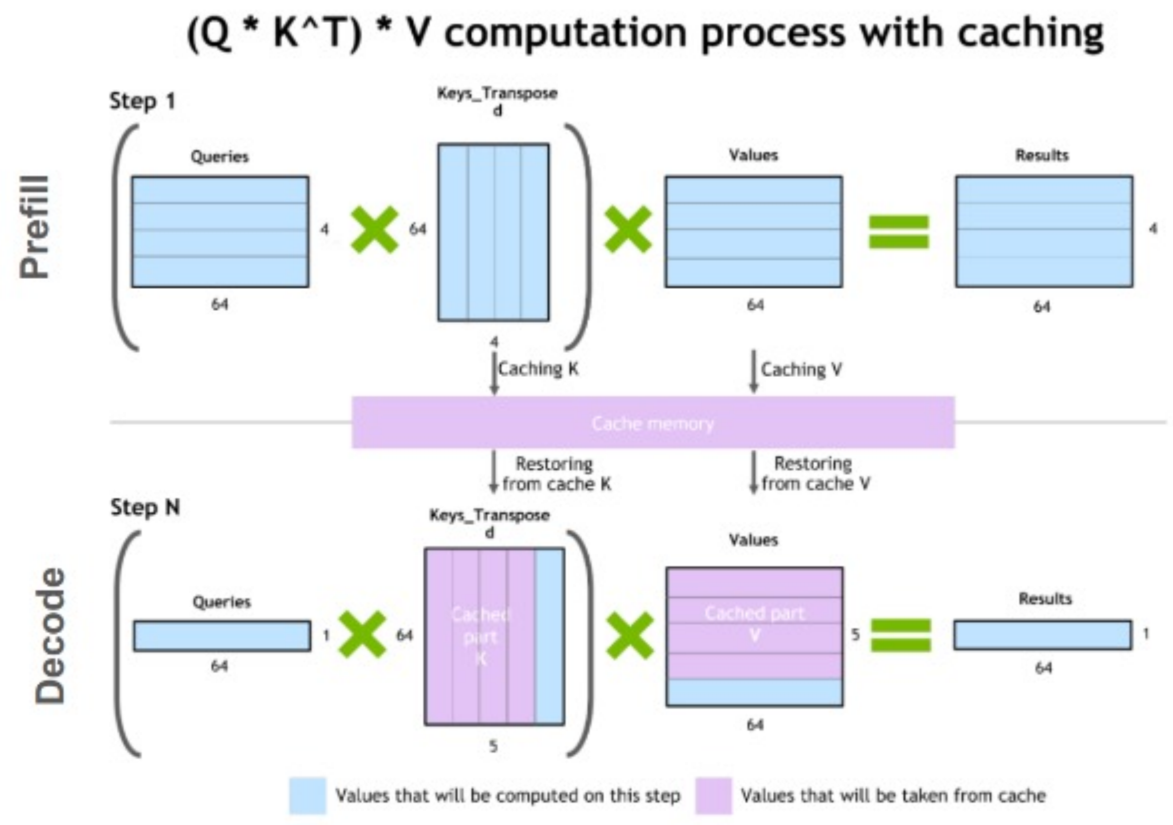
TPU v4[1]



3D Torus

[1] Jouppi, Norm, et al. "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings." ISCA 2023.

Preliminaries - 2



Source: https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/

Run single parallel forwards pass for:
$B\ sequences * L_{input}\ tokens$

Run sequential (autoregressive) forwards pass for:
$L_{gen}\ tokens$

Question: are there use-cases **where prefill is more critical** to optimize and vice-versa?

Preliminaries - 3

Models get larger → Need to partition across chips

How does that impact compute and memory costs for inference?

- Compute time: not much change — time to perform matrix multiply
- Memory time:
  - Need to load weights and KV cache
  - Small batches: Weights dominate
  - Large batches: KV cache dominates

Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

Partitioning attention

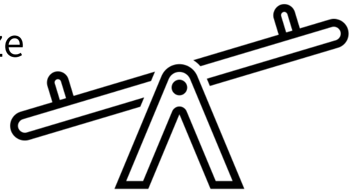
Results from PaLM

Comparison with FasterTransformer

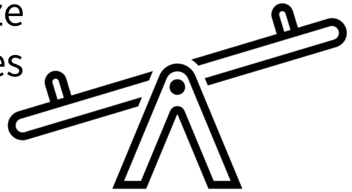
Discussion

Expected trade-offs

Trade-offs change with different use cases:



Low latency inference



Long context inference

Offline inference: Small and large batches require different partitioning strategies

Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

Partitioning attention

Results from PaLM

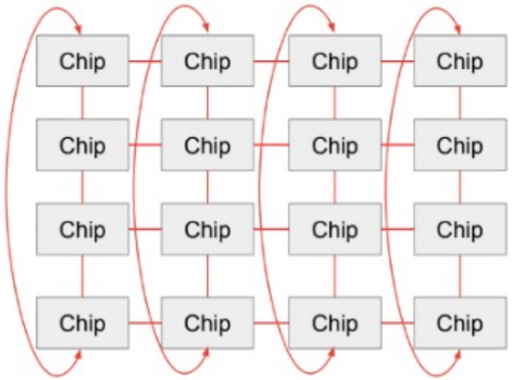
Comparison with FasterTransformer

Discussion

Partitioning Feedforward Layer

1D weight-stationary layout

- $E * F$ weight matrix stationary sharded along E or F axis.
- $B * L * E$ activation matrix also partitioned across all chips.



$\mathbf{BLE_{xyz}}$

Activations

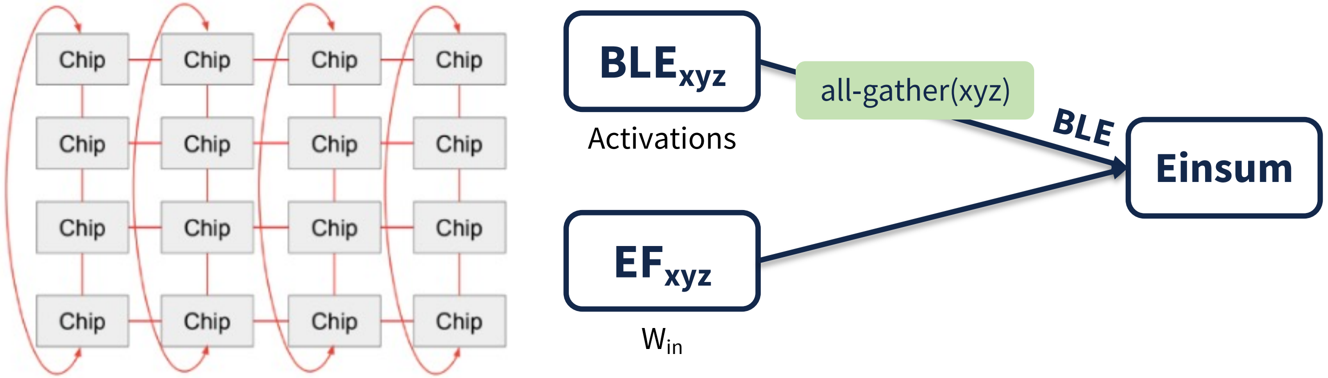
$\mathbf{EF_{xyz}}$

W_{in}

Partitioning Feedforward Layer

1D weight-stationary layout

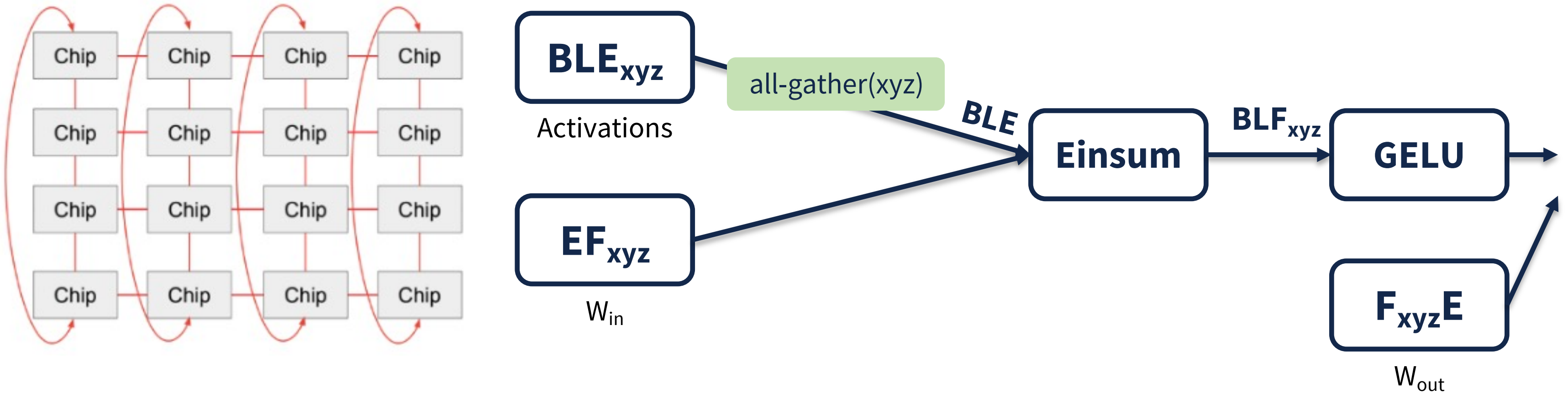
- $B * L * E$ activation matrix aggregated using all-gather.
- First matrix multiply performed.



Partitioning Feedforward Layer

1D weight-stationary layout

- Output $B * L * F_{xyz}$ matrix input to GELU activation.
- Second $E * F$ weight matrix sharded along second axis.

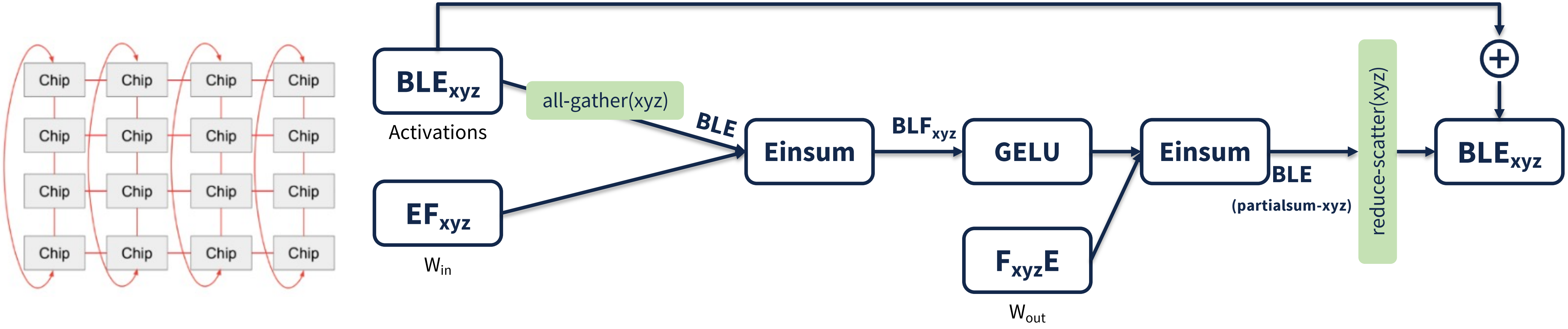


Output and input axis flip "trick" to reduce communication

Partitioning Feedforward Layer

1D weight-stationary layout

- Second matrix multiply performed.
- Partial sum is reduce-scatter-ed to all chips

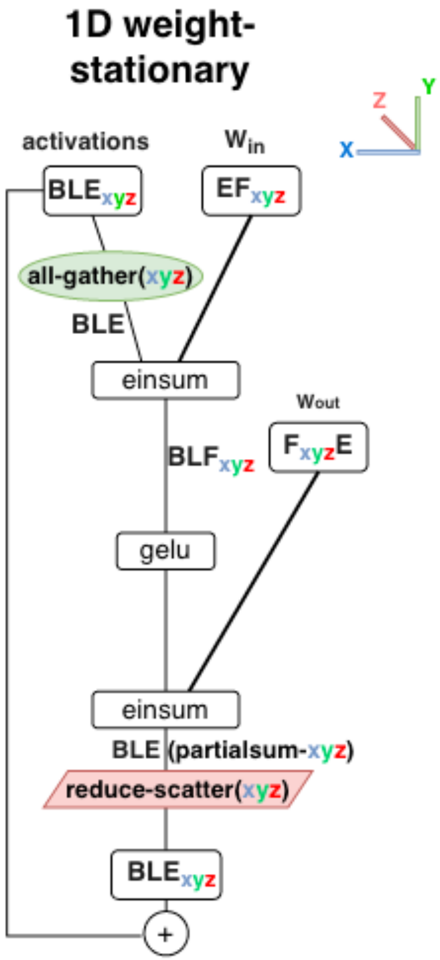


Partitioning Feedforward Layer

1D weight-stationary layout

- As the chips increase:
  - Compute and memory time decrease
  - Communication time constant (eventually bottleneck)
- Communication cost (all-gather + reduce-scatter):

$$T_{\text{comm}} = \frac{2BLE}{\text{network bandwidth}}$$

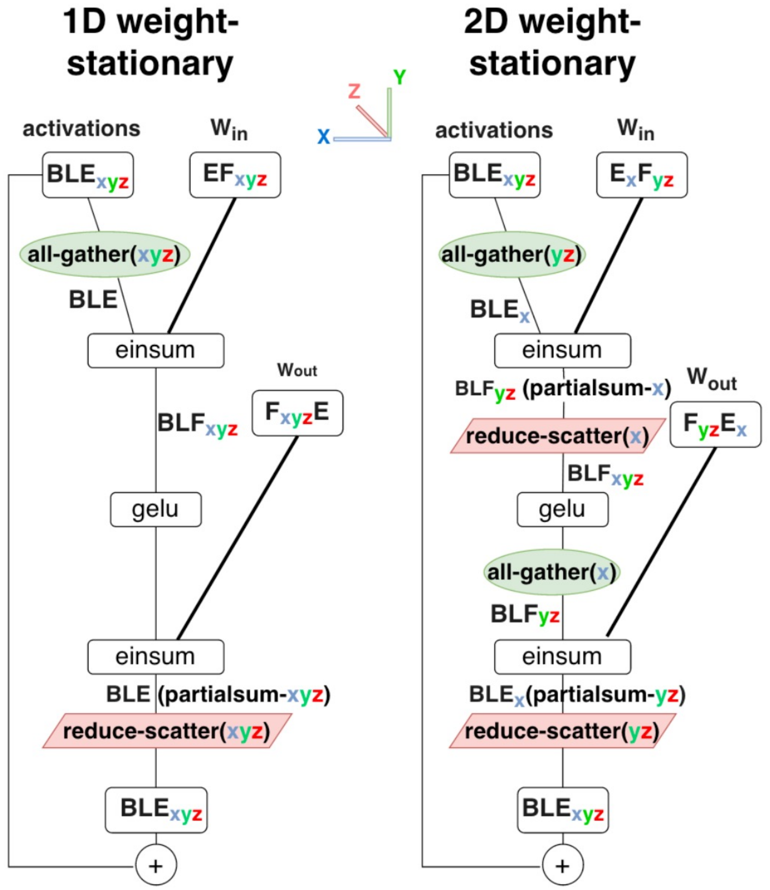


Partitioning Feedforward Layer

Extending to 2D weight-stationary layout:

- Partition weight across ***both*** E and F axes.
- Communication is more efficient:
  - Alternate axis to perform aggregation
  - Adds two more collective operations
  - Scales as $O\left(\frac{1}{\sqrt{n_{chips}}}\right)$ – more chips reduces latency!
- Communication cost:

$$T_{\text{comm}} = \frac{8BLE}{\sqrt{n_{\text{chips}}} \times \text{network bandwidth}}$$

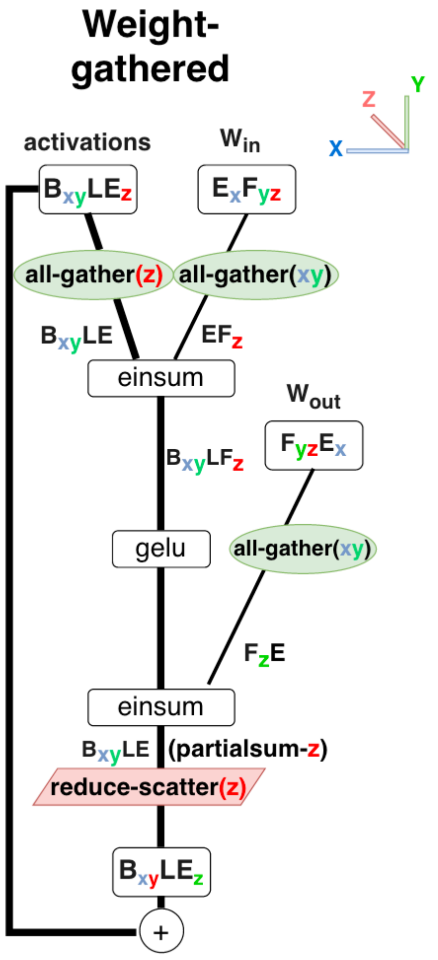


Partitioning Feedforward Layer

Extending to weight-gathered layout :

- As batch size increase
  - Keep activations stationary
  - Transfer weights between chips
- You could also have a hybrid approach:
  - Both are transferred across different axes
  - They propose XY-weight gathered used in prefill
  - Weight across X and Y; activations across Z
- Communication cost:

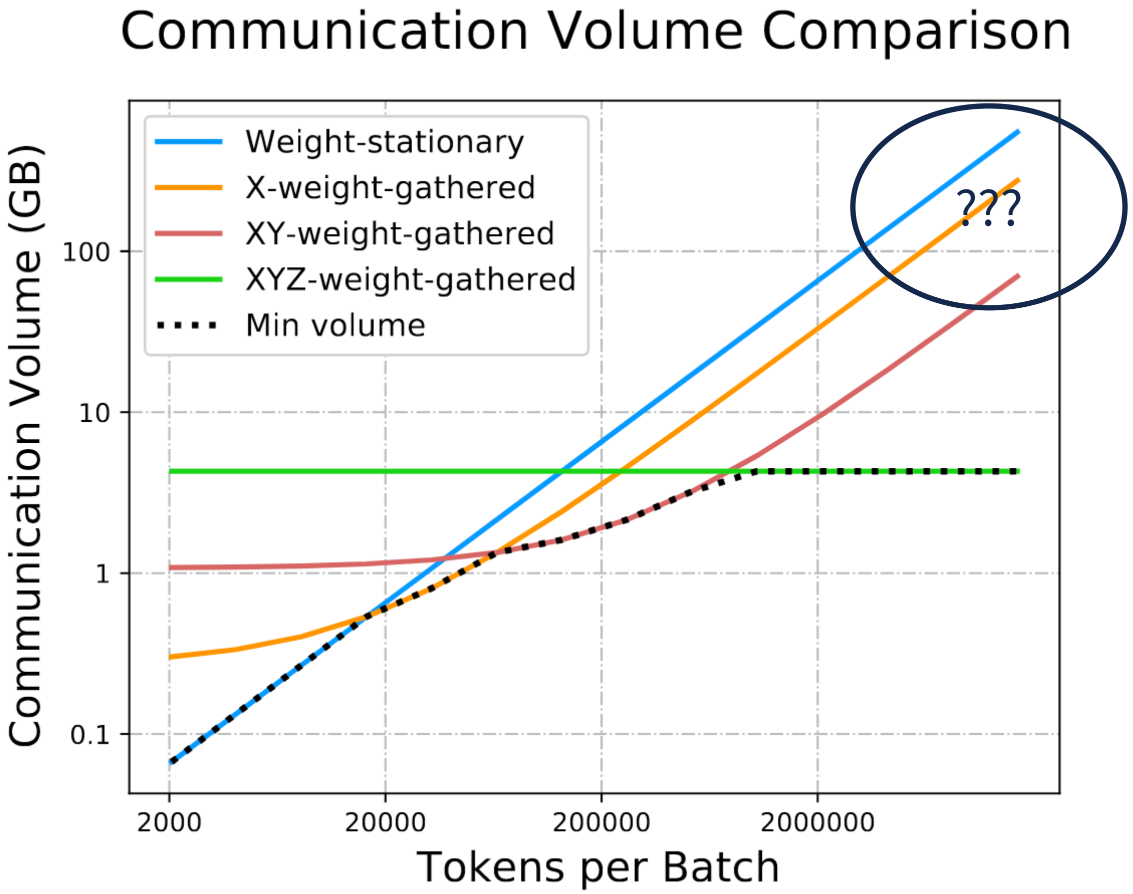
$$T_{\text{comm}} = 4E \frac{\sqrt{BLF}}{\sqrt{n_{\text{chips}}} \times \text{network bandwidth}}$$



Partitioning Feedforward Layer

Trade-offs between the approaches:

- How do they scale with batch size?



- Question: why linear?

Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

Partitioning attention

Results from PaLM

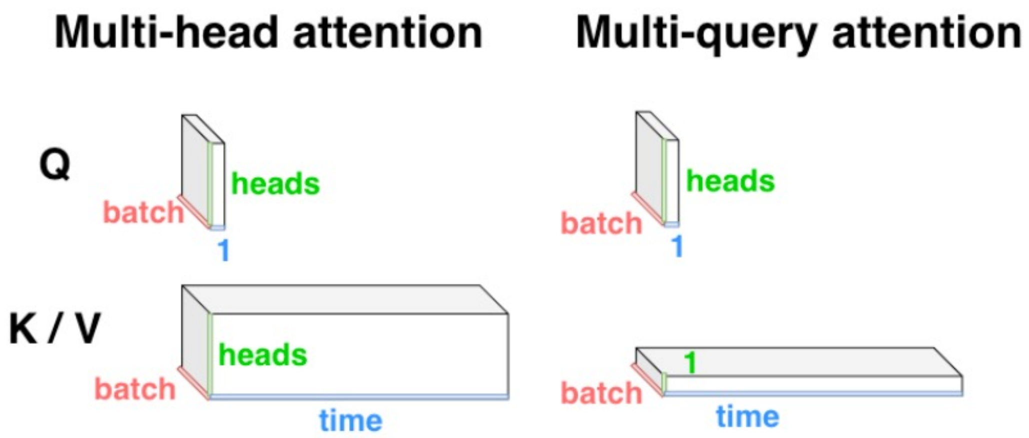
Comparison with FasterTransformer

Discussion

Partitioning Attention Layer

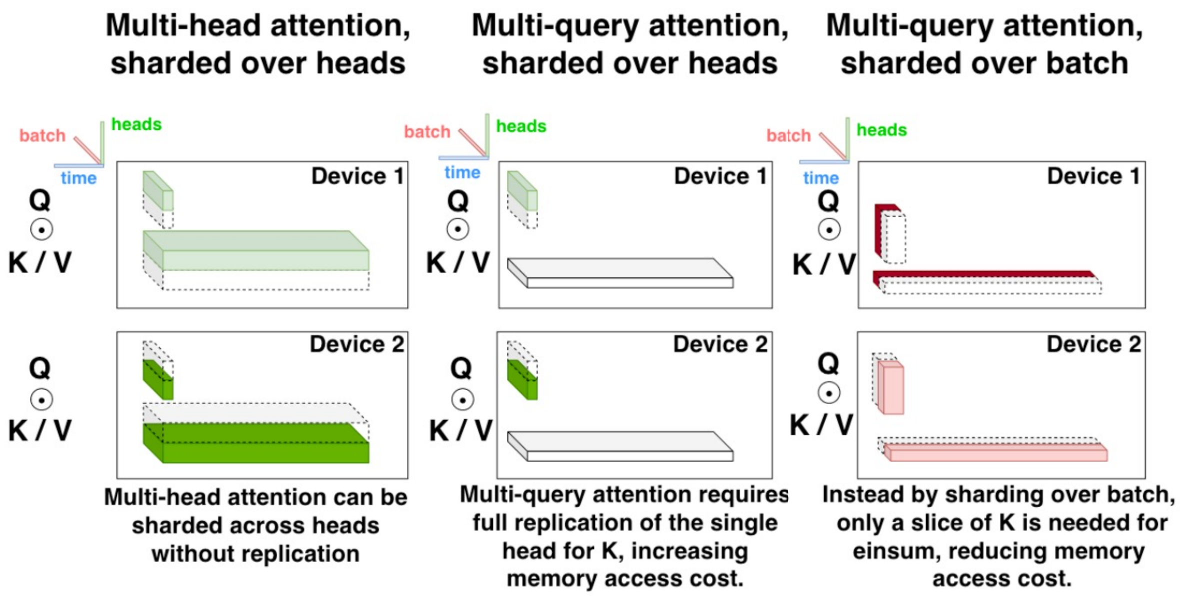
Changes to model architecture:

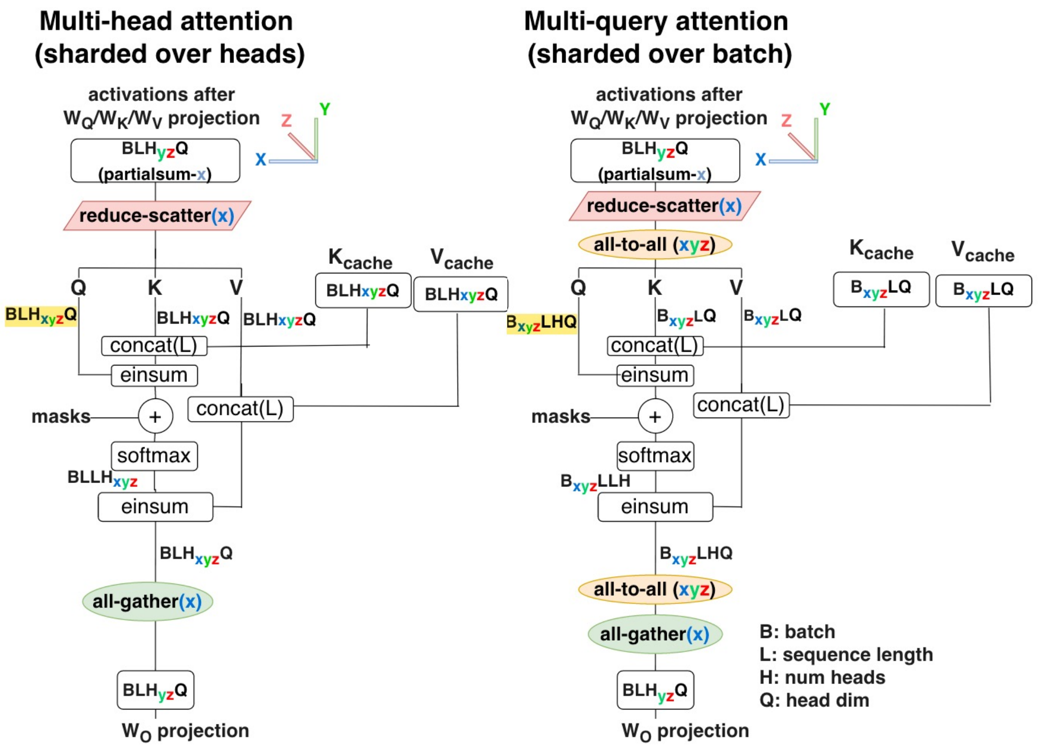
- **Multi-query attention** vs. multi-head attention
  - n_{heads} for the query, but one head for the key and value



- **Parallel formulation** vs. serialized formulation of transformer
  - Question: Megatron-style model parallel and multi-query?

Partitioning Attention Layer





Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

Partitioning attention

Results from PaLM

Comparison with FasterTransformer

Discussion

Case study – PaLM models

Large transformer model from Google:

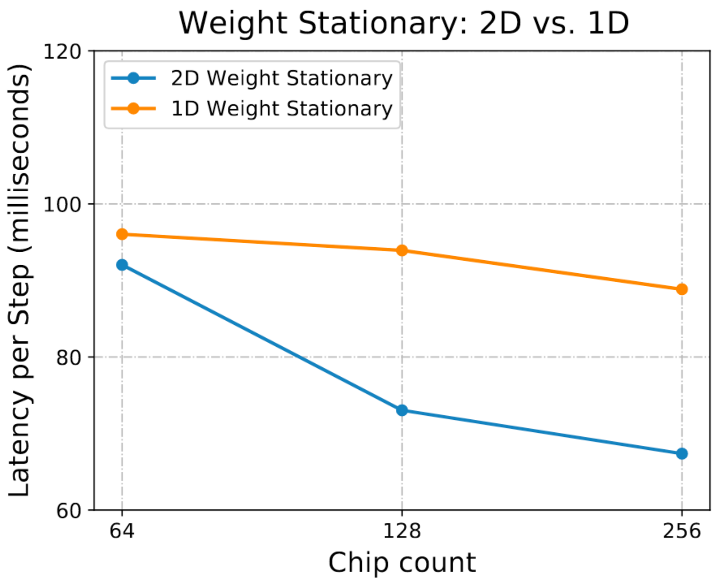
- Predecessor to the new Gemini model
- Incorporates multi-query attention and parallel transformer.
- Thought: case of model-system co-design

See Chowdhery, et al. "Palm: Scaling language modeling with pathways.”

Impact of partitioning feedforward layer

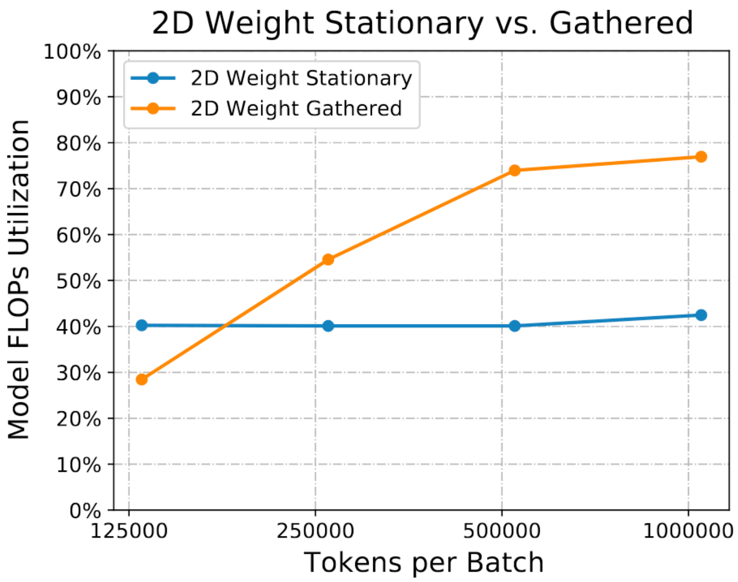
Performance of decoding

Latency Scaling with Chip Count



Performance of prefill

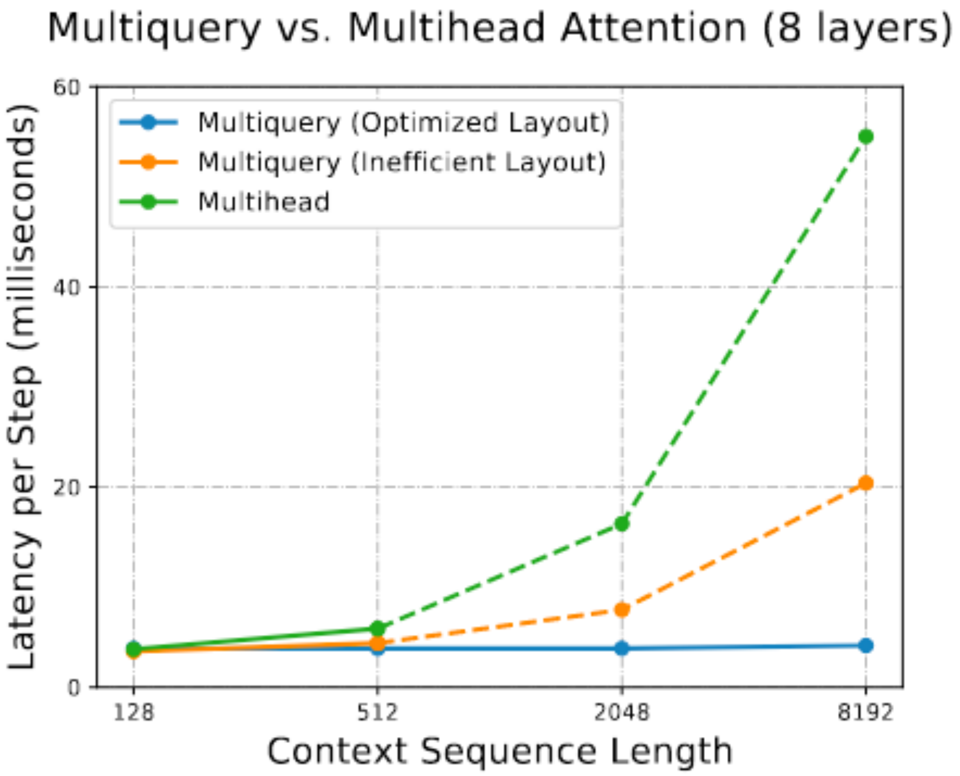
Utilization Scaling with Batch size



Impact of partitioning attention layer

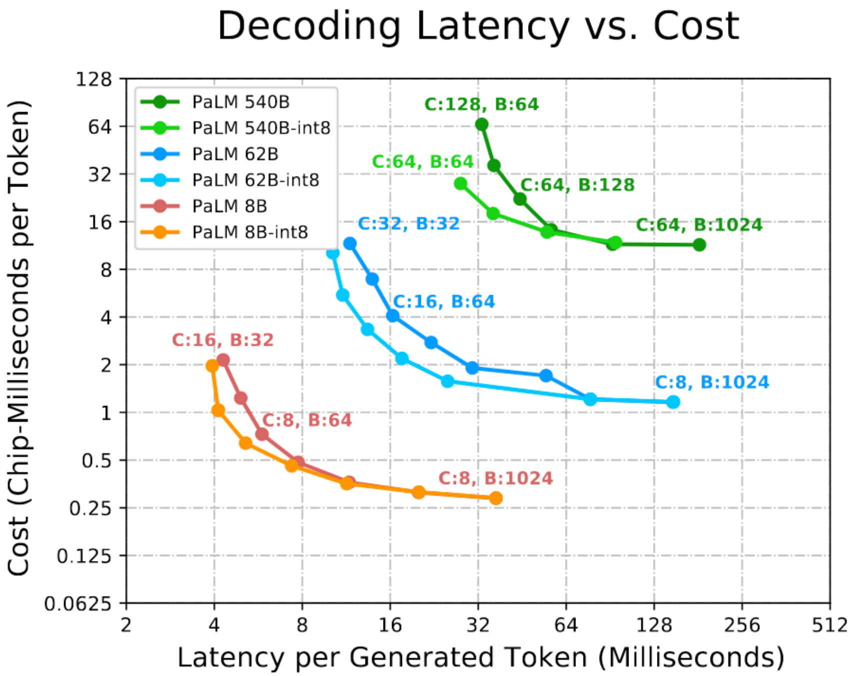
Performance of decoding

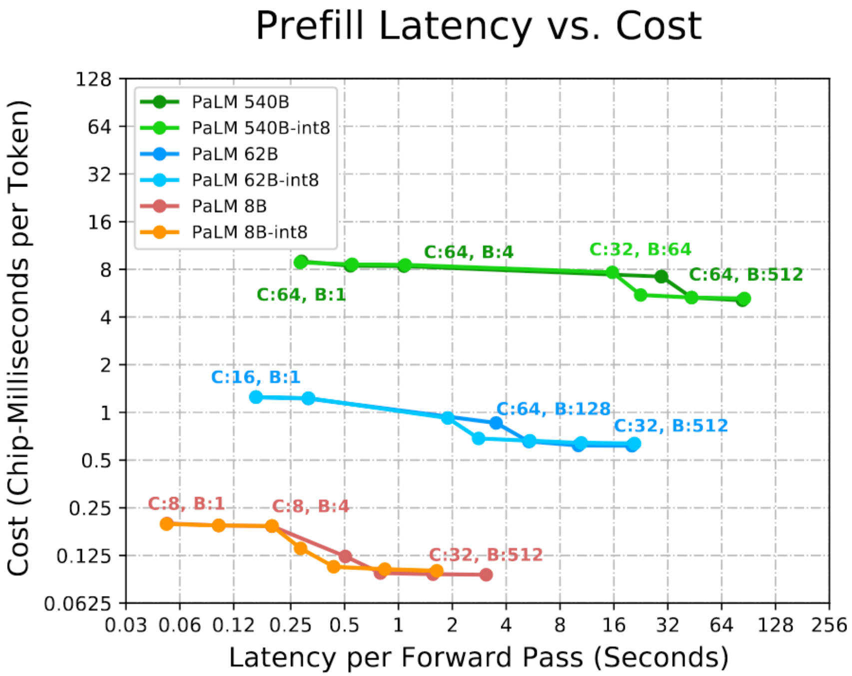
Latency Scaling with Sequence Length



Question: what about prefill?

End-to-End results





Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

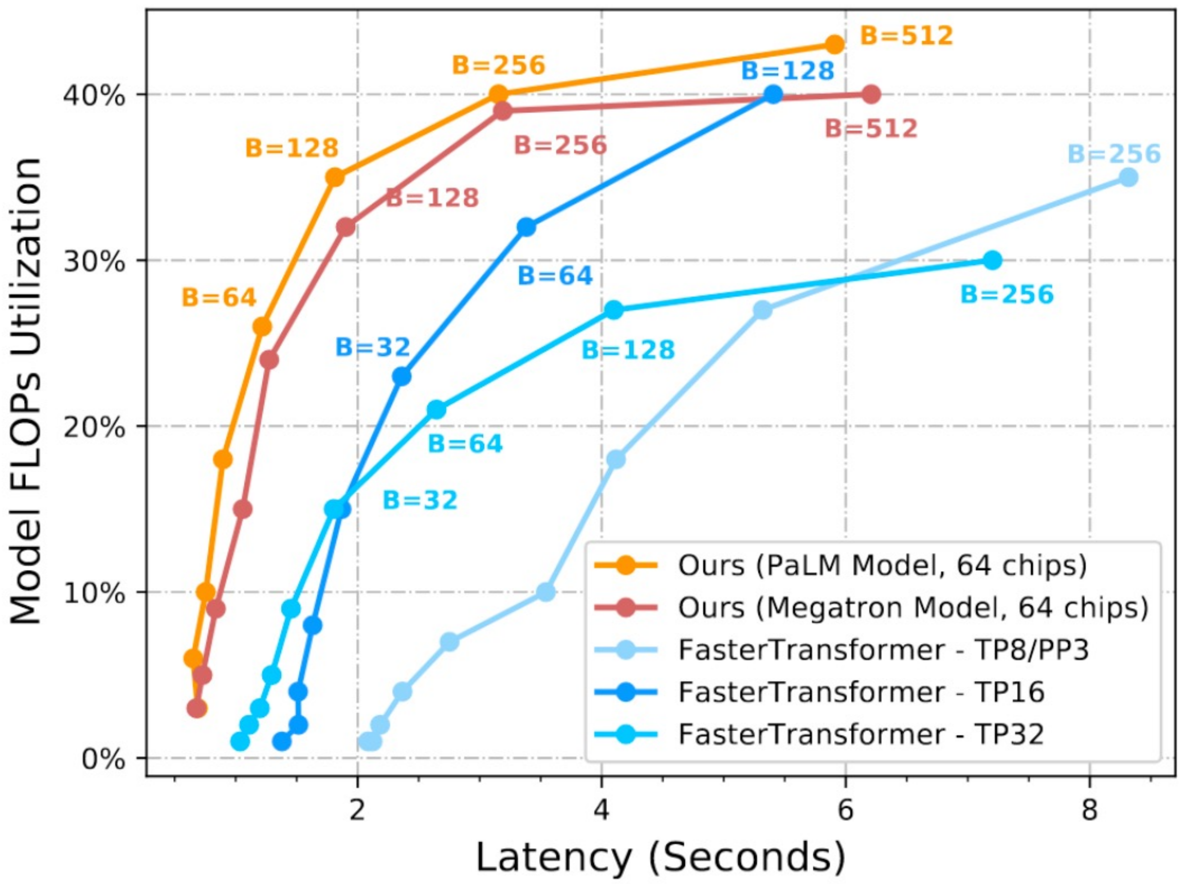
Partitioning attention

Results from PaLM

Comparison with FasterTransformer

Discussion

Comparison with FasterTransformer



Summary

Inference

- Prefill and decoding have different trade-offs

Transformer

- PaLM model with multi-query attention and parallel formulation

Scaling

- Partitioning strategies for feedforward and attention

Efficiently

- Different strategies are efficient for different use cases:
  - chip count/batch size/sequence length

Overview

Preliminaries

Expected trade-offs

Partitioning feedforward layer

Partitioning attention

Results from PaLM

Comparison with FasterTransformer

Discussion

Discussion

- Initial thoughts?
- What is a more generalized strategy for any transformer architecture?
- GPU vs TPU
  - This paper does not make a case to use TPU over GPU (they could have)
  - So, what is the case for TPU?
- How can we further improve decoding utilization? (~40% for PaLM)