Transformers, Profiling, and Performance Modeling

Abhinav Bhatele, Daniel Nichols

UNIVERSITY OF MARYLAND

Announcements

- Assignment 1
  - Due Feb. 25th at midnight
- Office hours (IRB 3119)
  - Tuesday (today) 2/18 3-4pm (zoom only)
  - Thursday 2/20 10-11am
- Groups due March 4th
- No class this Thursday

Sequence Modeling

- Sequence modeling
- Given a sequence of values (i.e. tokens or words) we want to model the probability they are in some language/text distribution

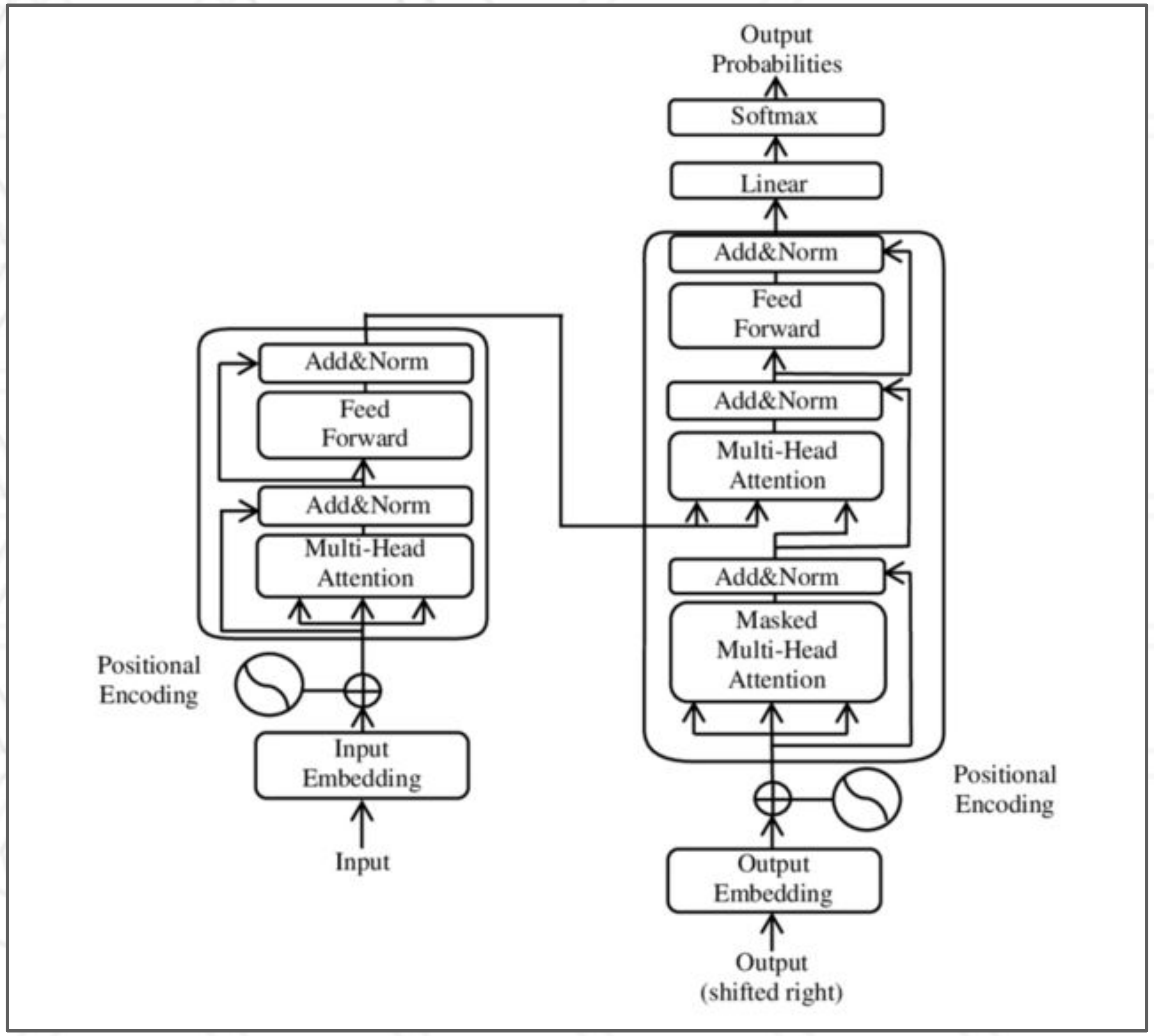
$$P(\text{"There's no place like home"}) = 0.1$$
$$P(\text{"There's is a place like home"}) = 0.08$$
$$P(\text{"There's no home place like"}) = 0.0001$$

- Instead of the joint distribution, we can model conditional probabilities

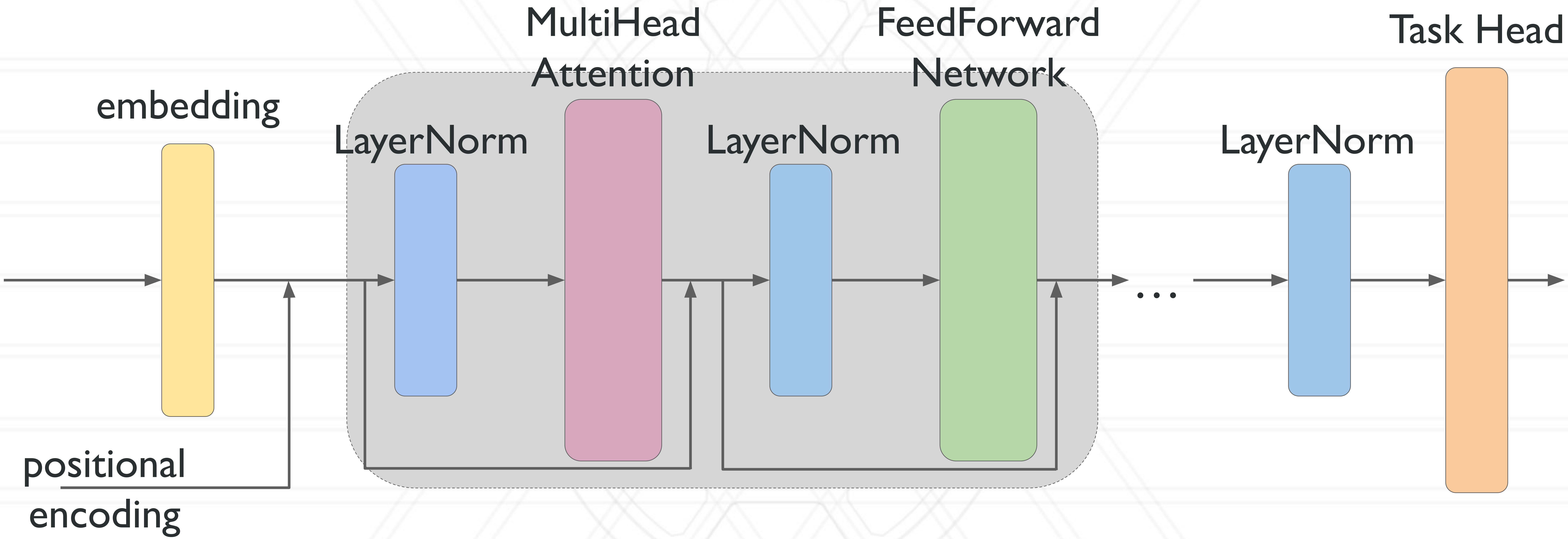
$$P(x_{1:i}) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \cdots P(x_i \mid x_{1:i-1})$$

Transformers

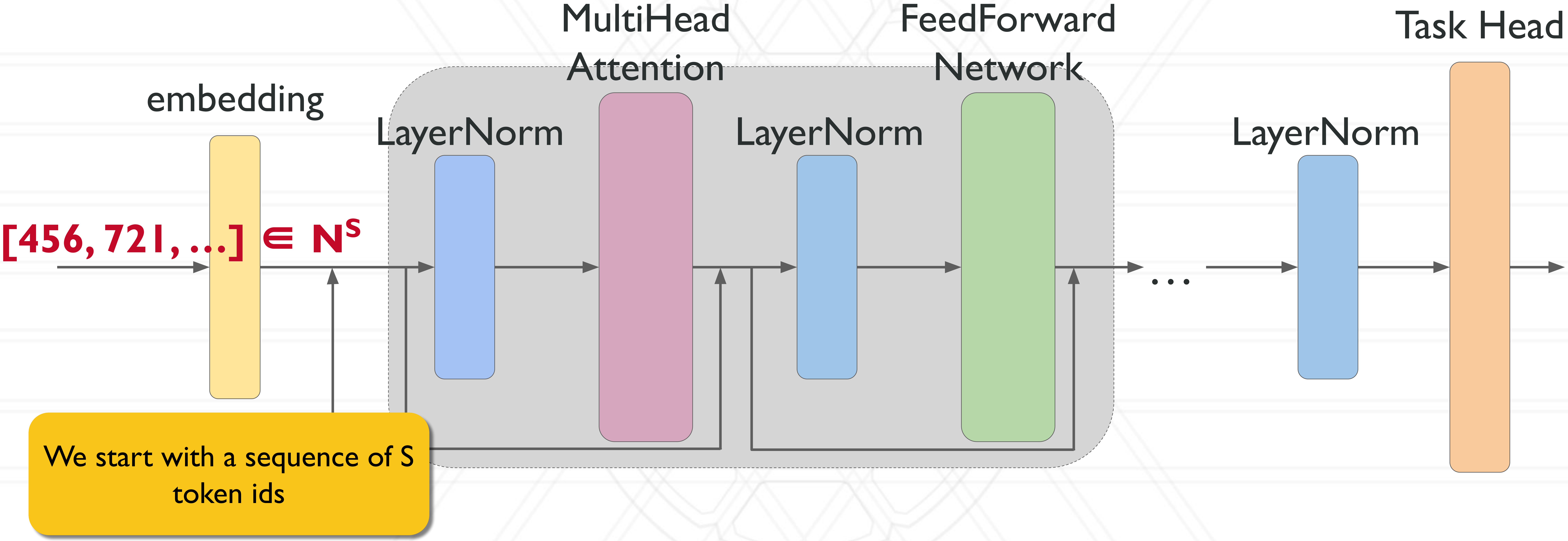
- Introduced in 2017, "Attention Is All You Need," Vaswani et. al.
  - Uses self-attention to model context



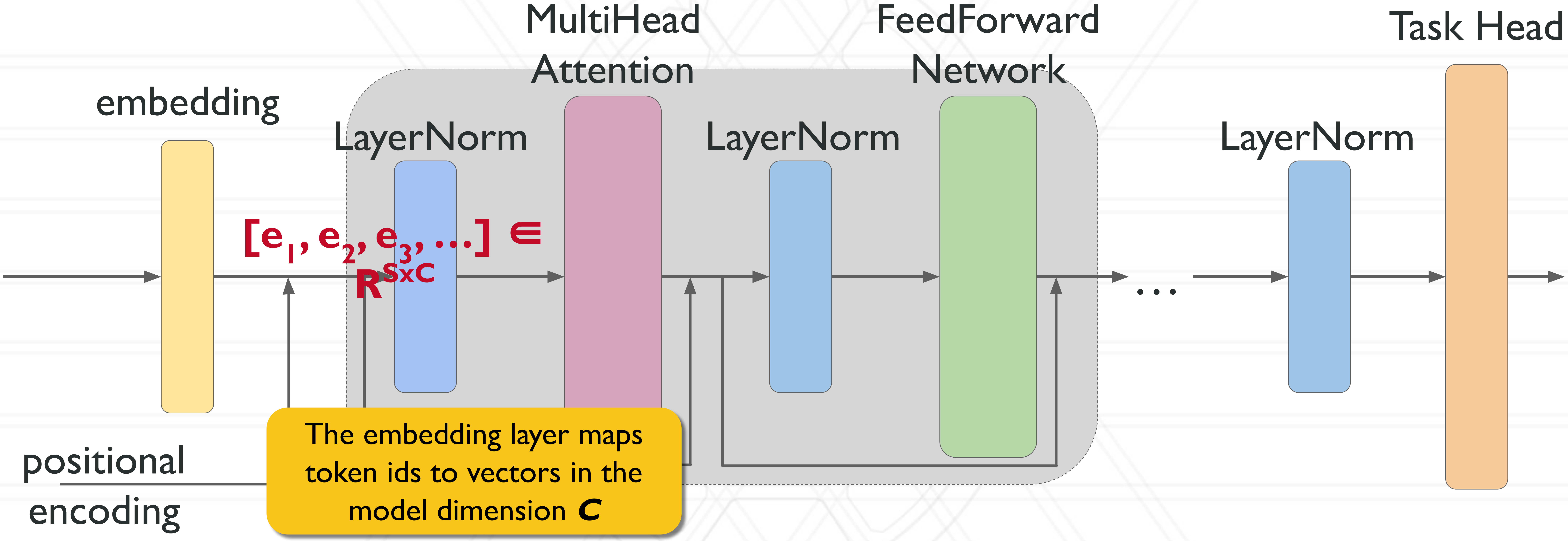
An Example Architecture



An Example Architecture

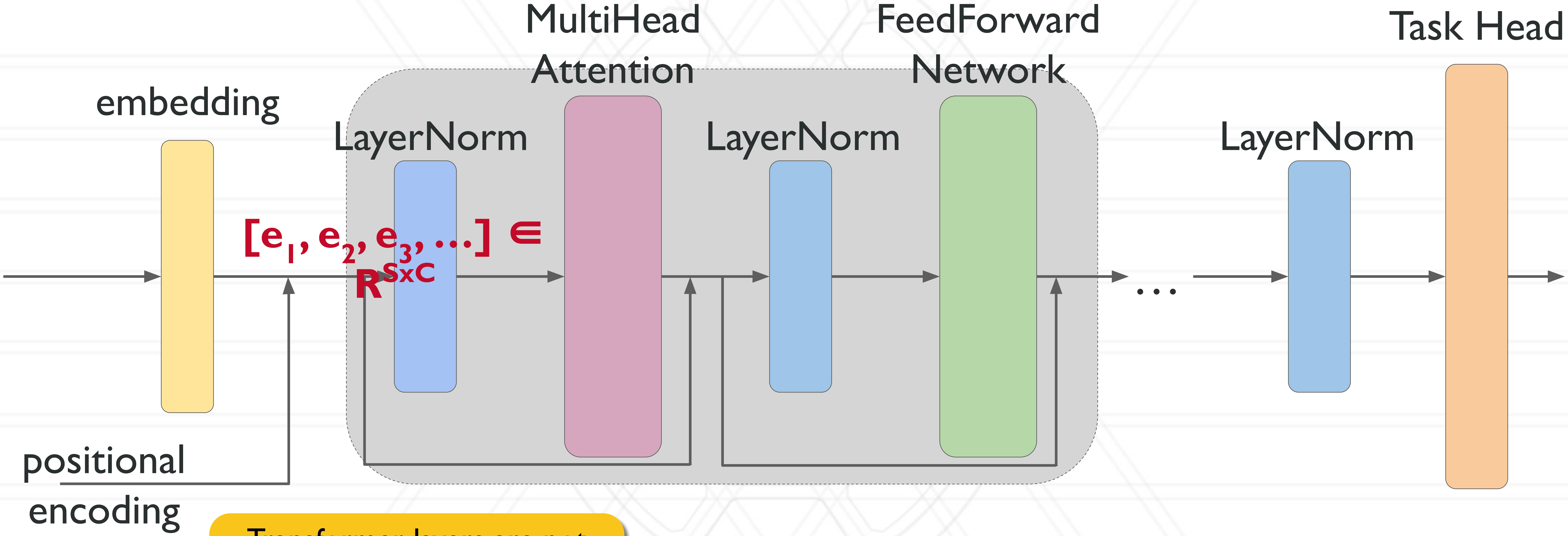


An Example Architecture



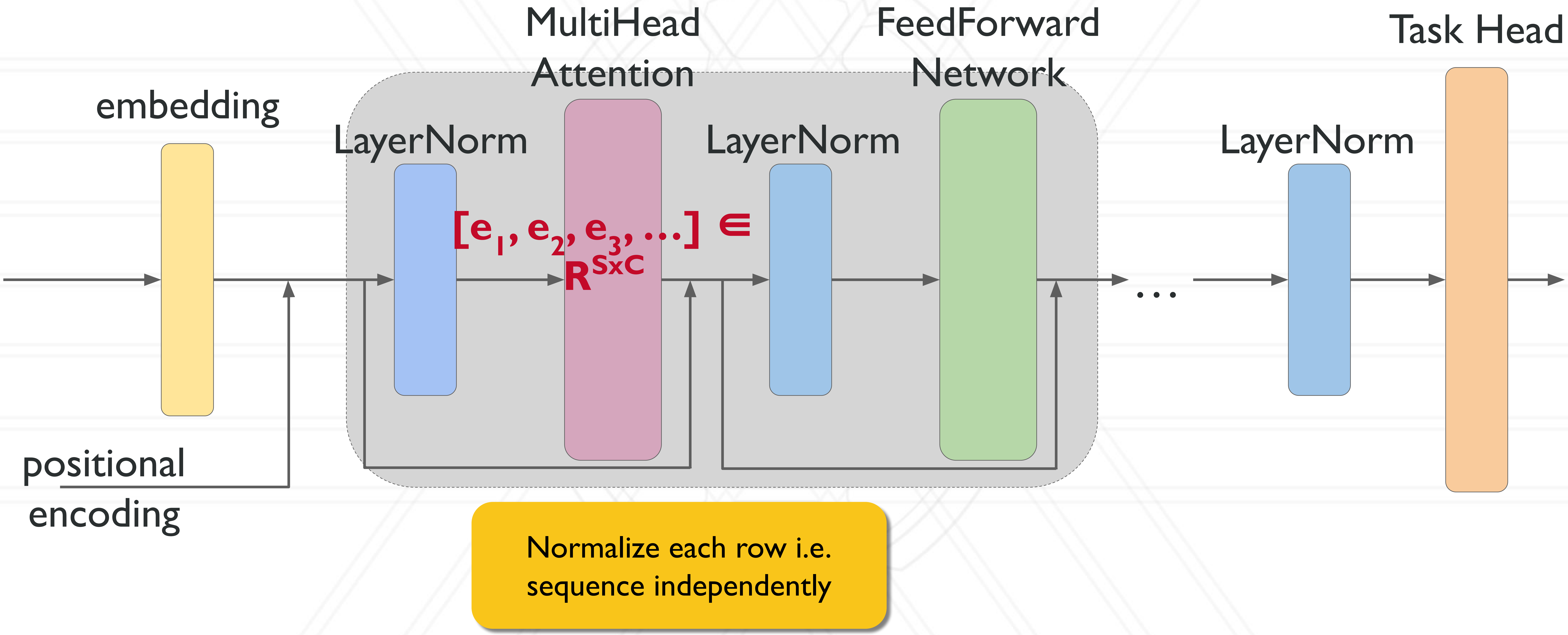
The embedding layer maps token ids to vectors in the model dimension ***C***

An Example Architecture



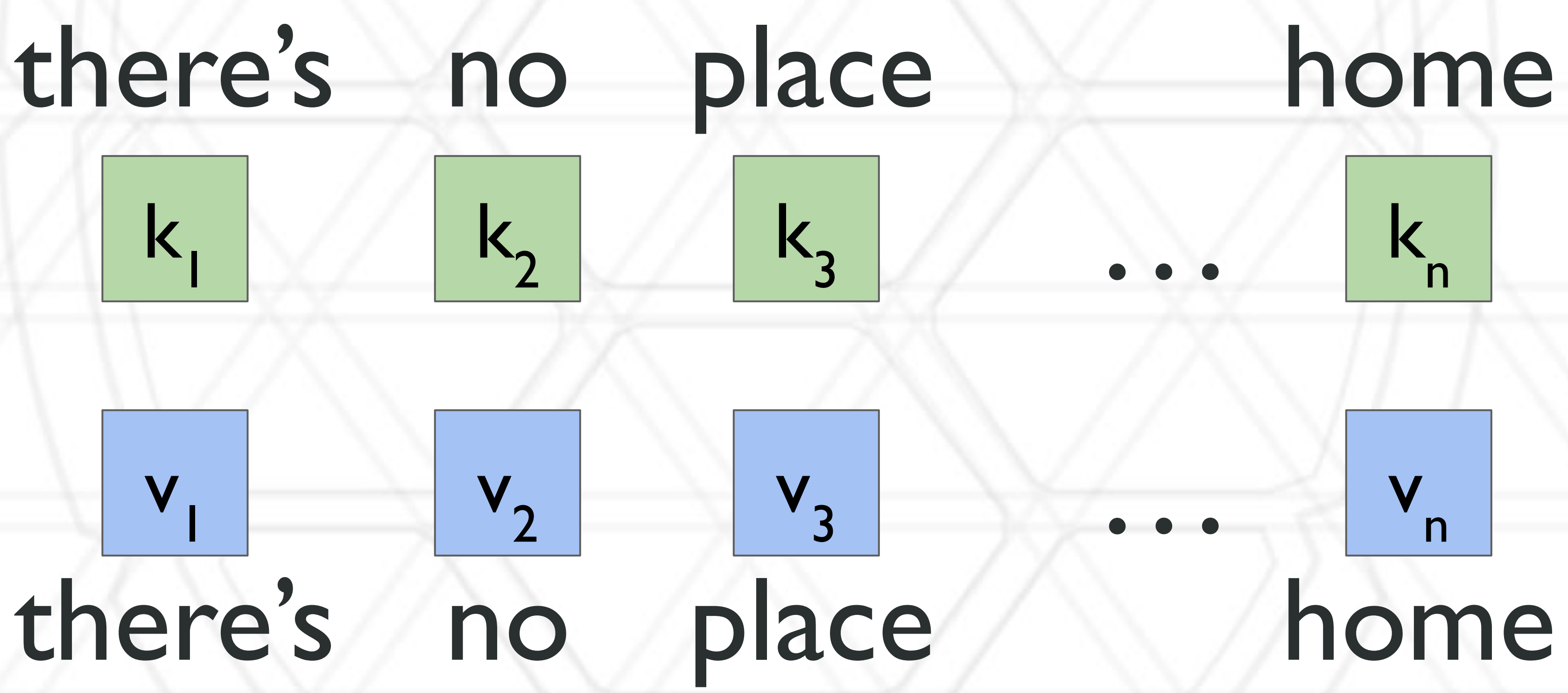
Transformer layers are not aware of position, so add encoded position information

An Example Architecture



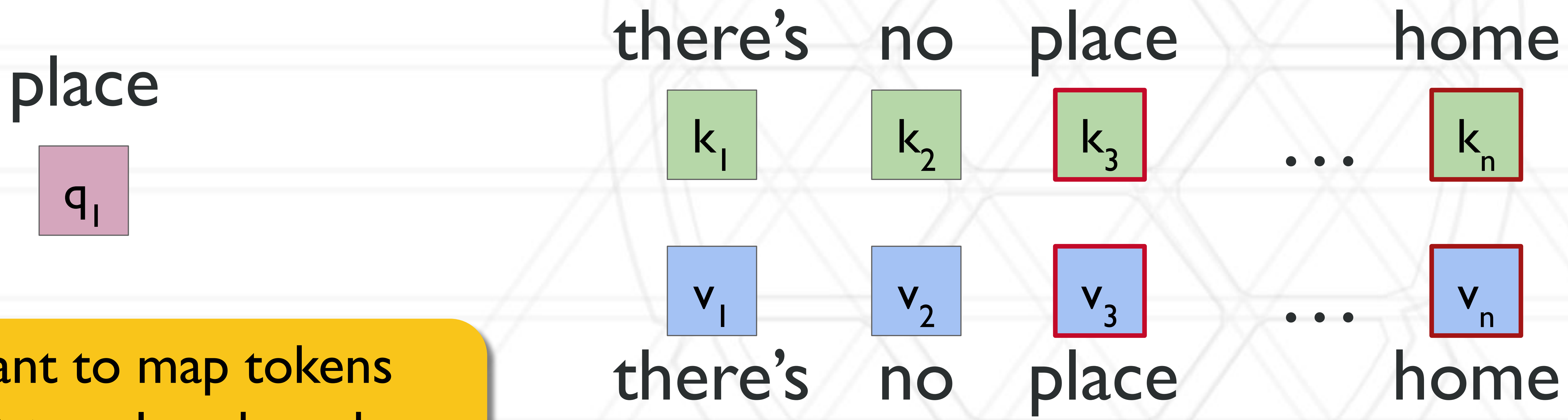
Attention

- Determine how much tokens should "attend" to other tokens
- Consider a hashmap of our tokens



Attention

- Determine how much tokens should "attend" to other tokens
- Consider a hashmap of our tokens

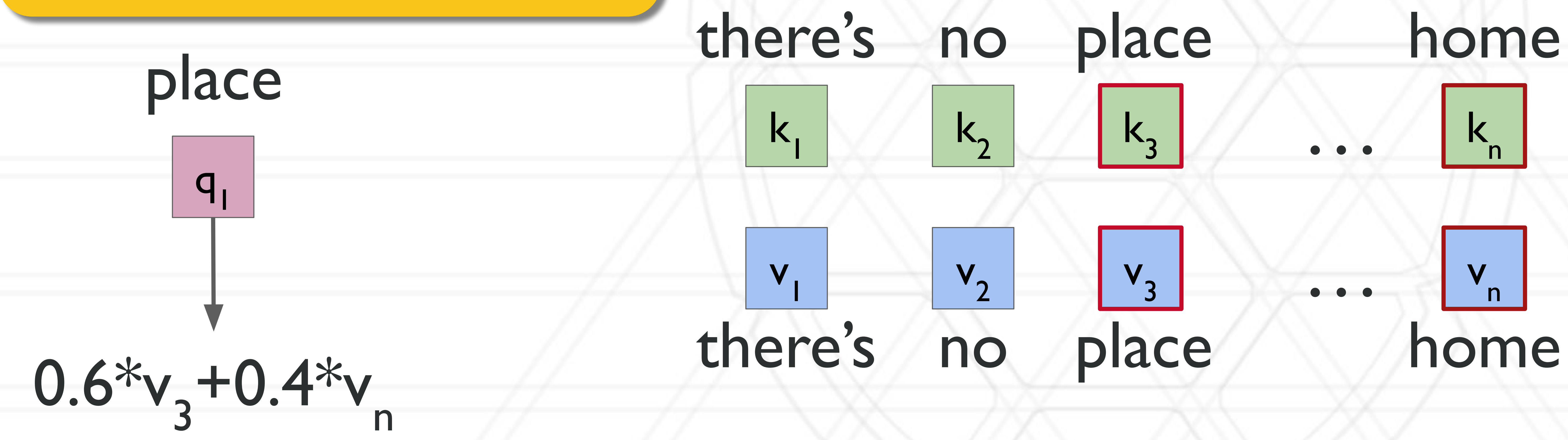


We want to map tokens (queries) to values based on the most similar keys

Attention

- Determine how much tokens should "attend" to other tokens
- Consider a hashmap of our tokens

We can answer this query with a scaled dot product of the similar values



But how much should we scale this dot product?

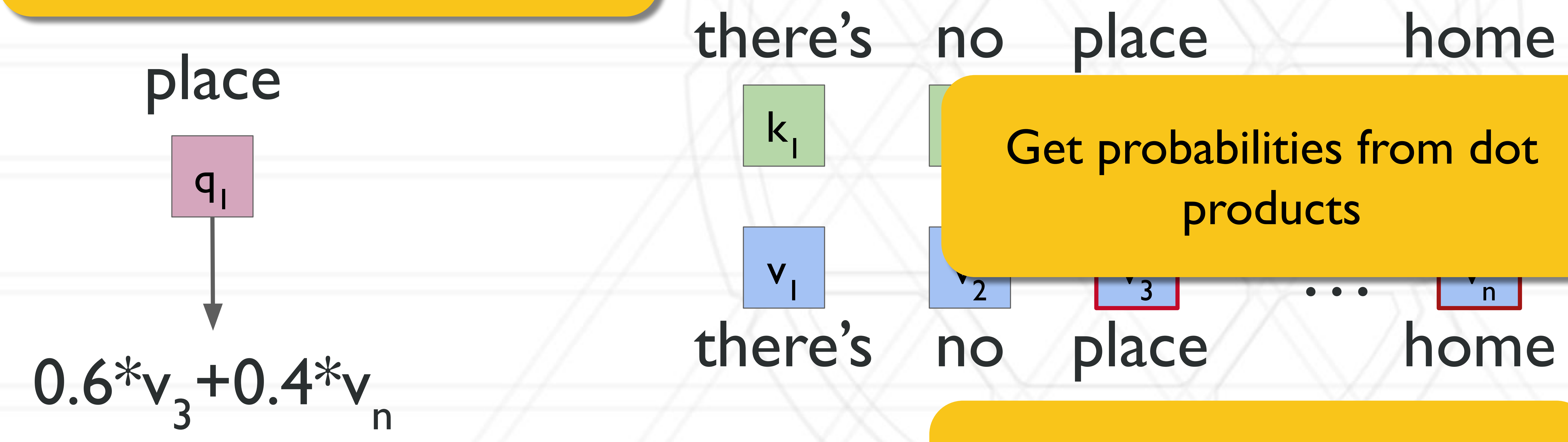
Attention

- Determine how much tokens should “attend” to other tokens
- Consider a hashmap of our tokens

We can answer this query with a scaled dot product of the similar values

Compute dot product of the query with all the keys

$$O = [q_i k_1, q_i k_2, \ldots, q_i k_n]$$



Get probabilities from dot products

$$P = \mathrm{softmax}(O)$$

Scale the values based on probabilities

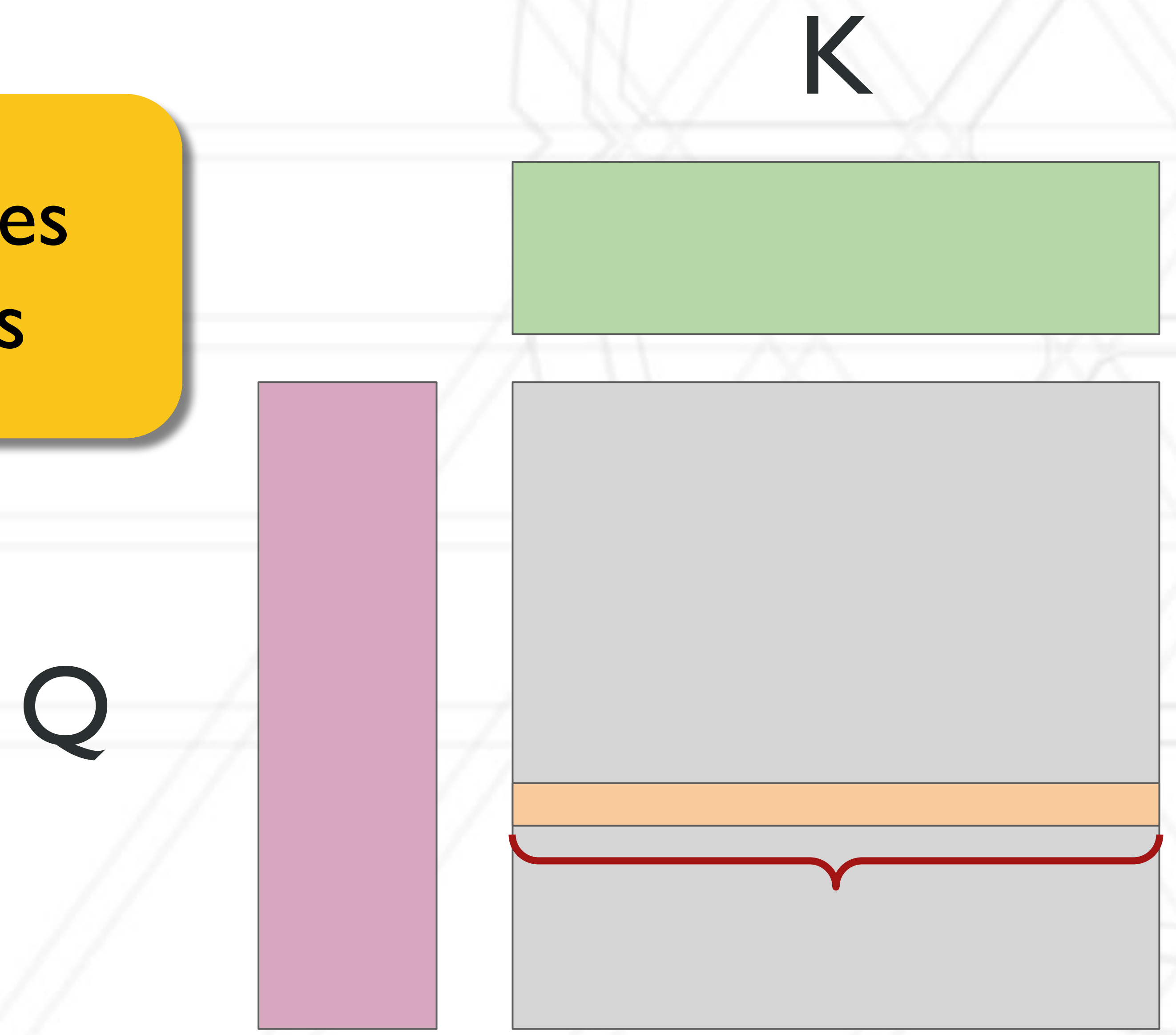
$$x_i = P[v_1, \ldots, v_n]^T$$

But how much should we scale this dot product?

Attention

- Determine how much tokens should "attend" to other tokens
- Can be done in batches quite efficiently
- Create weights (W_q, W_k, W_v) using linear transformations ($Q=XW_q$, $K=XW_k$, $V=XW_v$)

Matrix multiplication gives query-key dot products

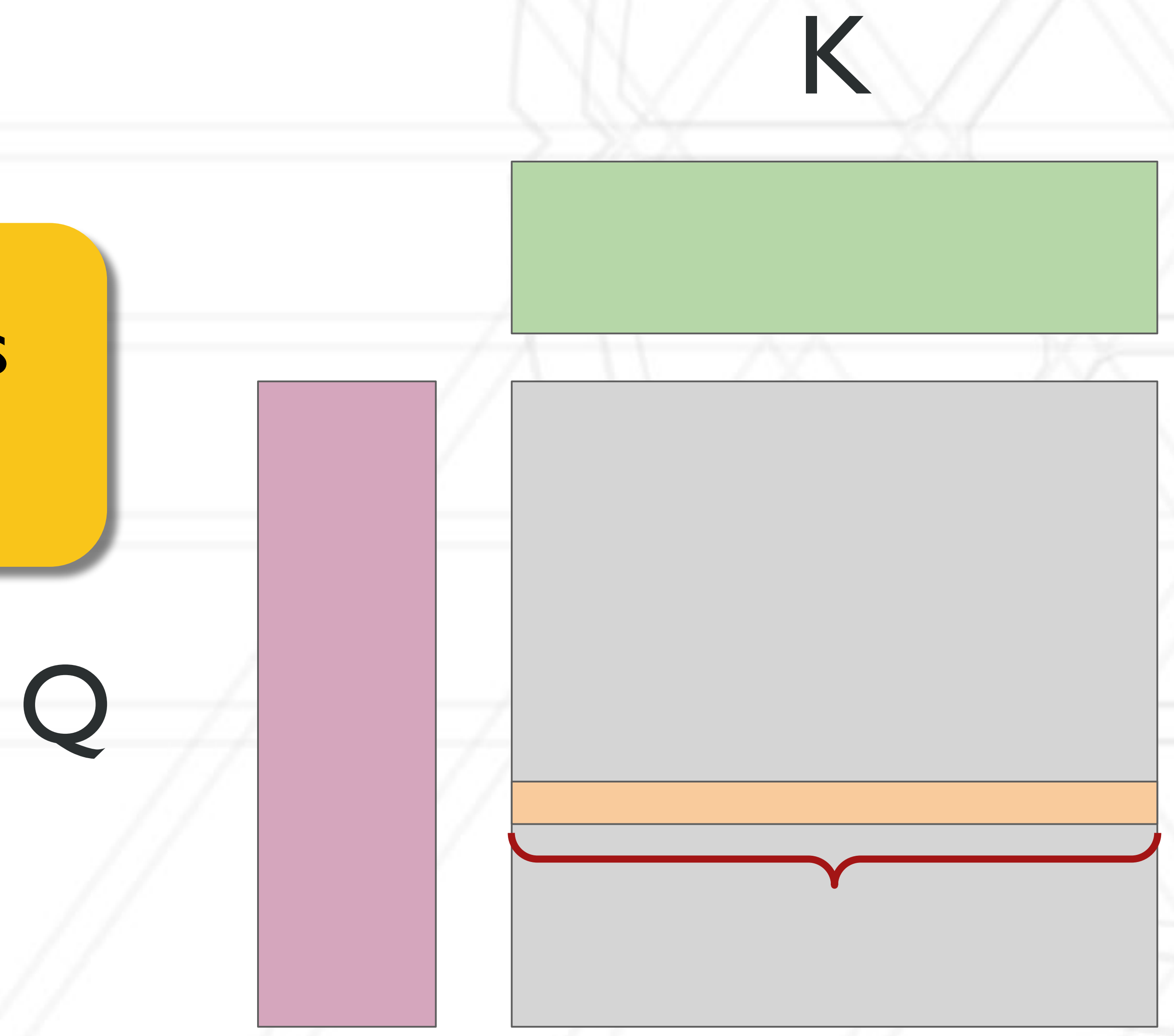


$$O = QK^T$$

Attention

- Determine how much tokens should "attend" to other tokens
- Can be done in batches quite efficiently
- Create weights (W_q, W_k, W_v) using linear transformations ($Q=XW_q$, $K=XW_k$, $V=XW_v$)

Softmax of each row yields probabilities

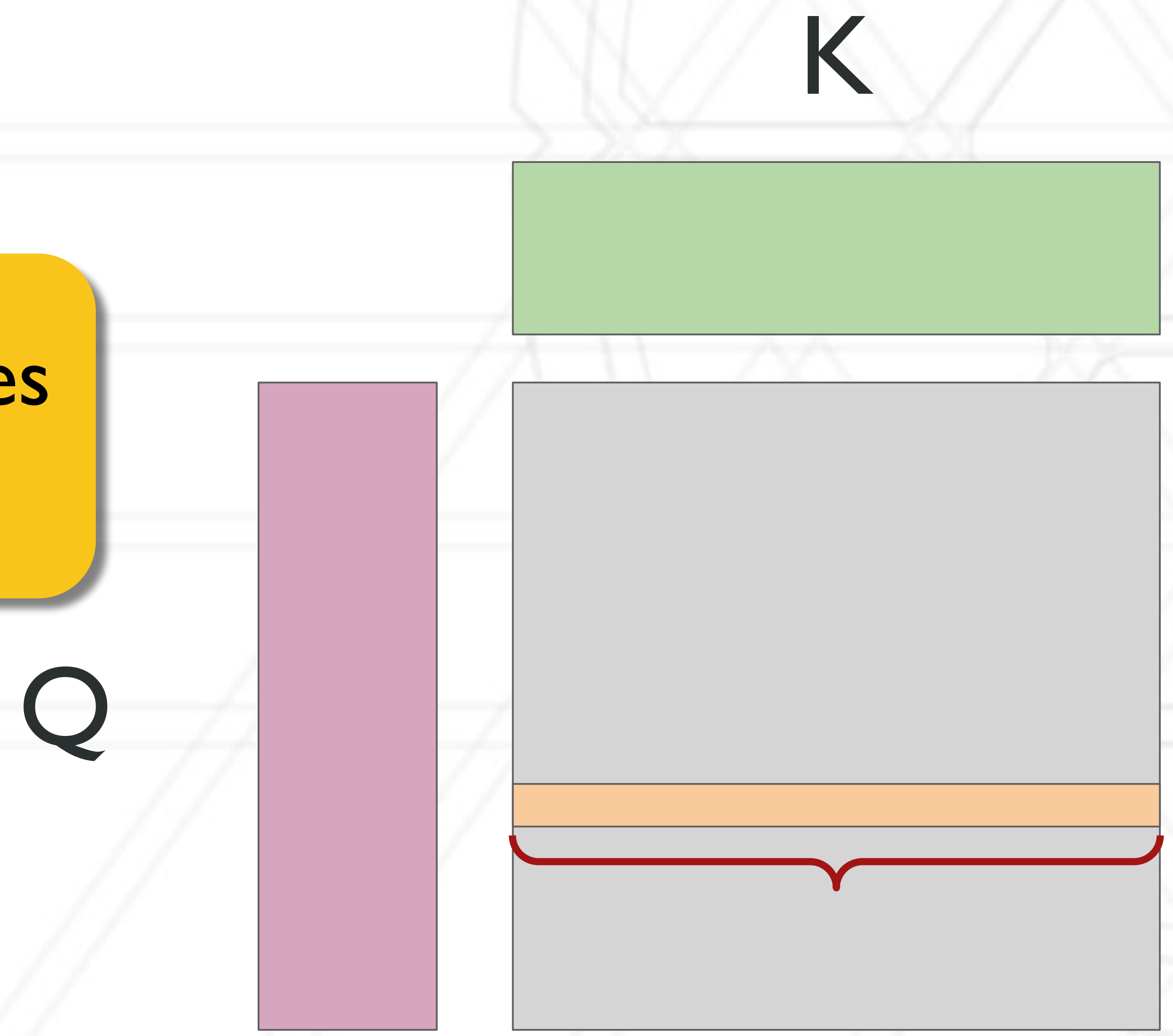


$$O = \mathrm{softmax}(QK^T)$$

Attention

- Determine how much tokens should "attend" to other tokens
- Can be done in batches quite efficiently
- Create weights (W_q, W_k, W_v) using linear transformations ($Q=XW_q$, $K=XW_k$, $V=XW_v$)

Finally, we can scale the values to get the final result

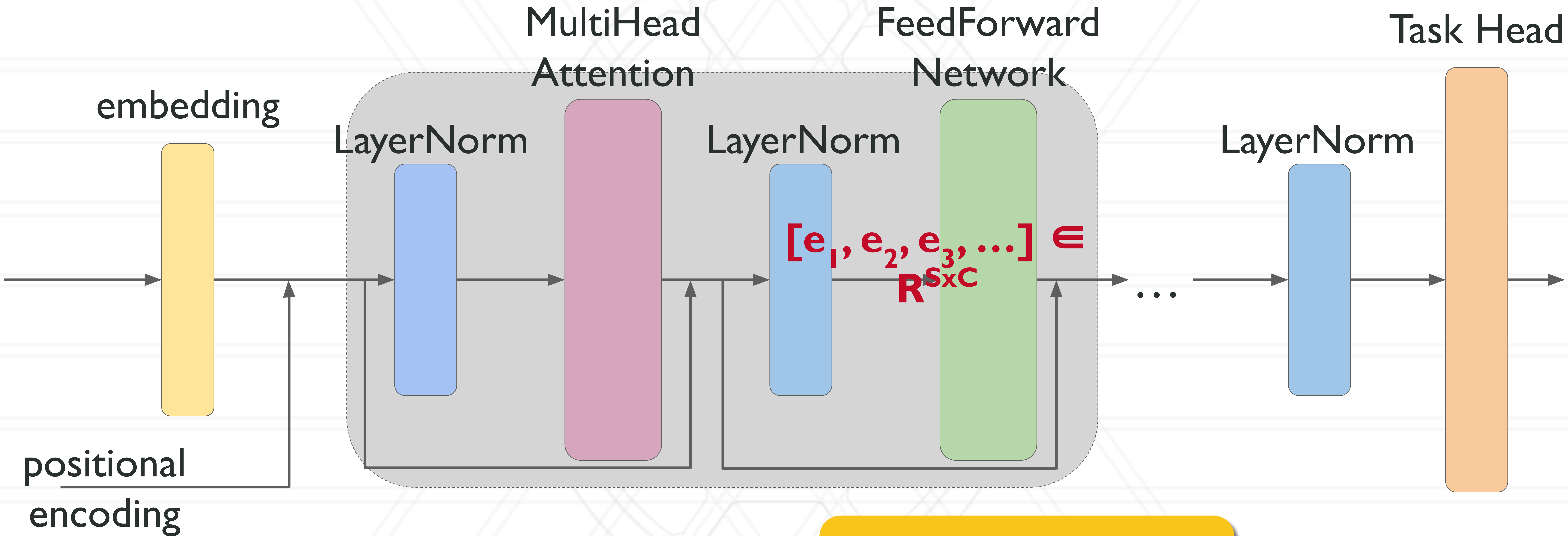


$$\text{Attn} = \text{softmax}(QK^T)V$$

Multihead Attention

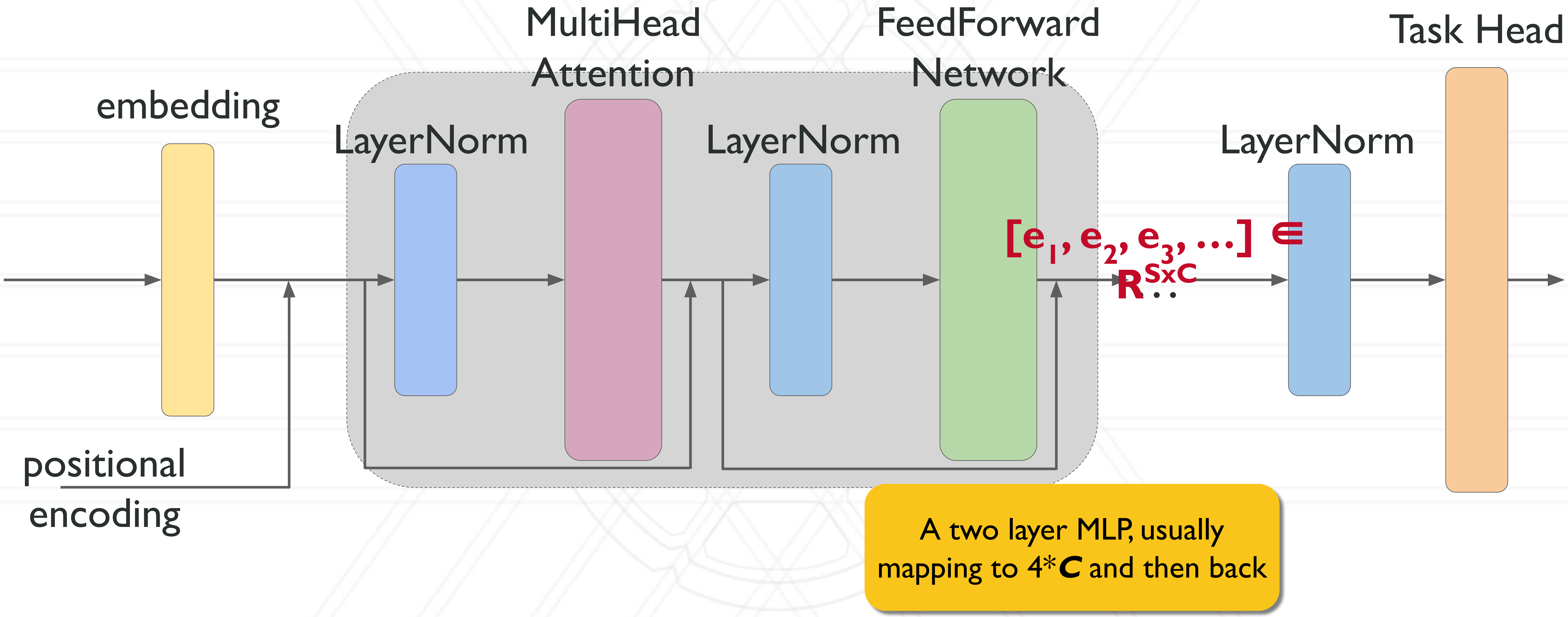
- Determine how much tokens should "attend" to other tokens
- Can be done in batches quite efficiently
- Compute multiple attentions and concatenate them
  - A projection matrix is used to project back to the model's embedding dimension

An Example Architecture

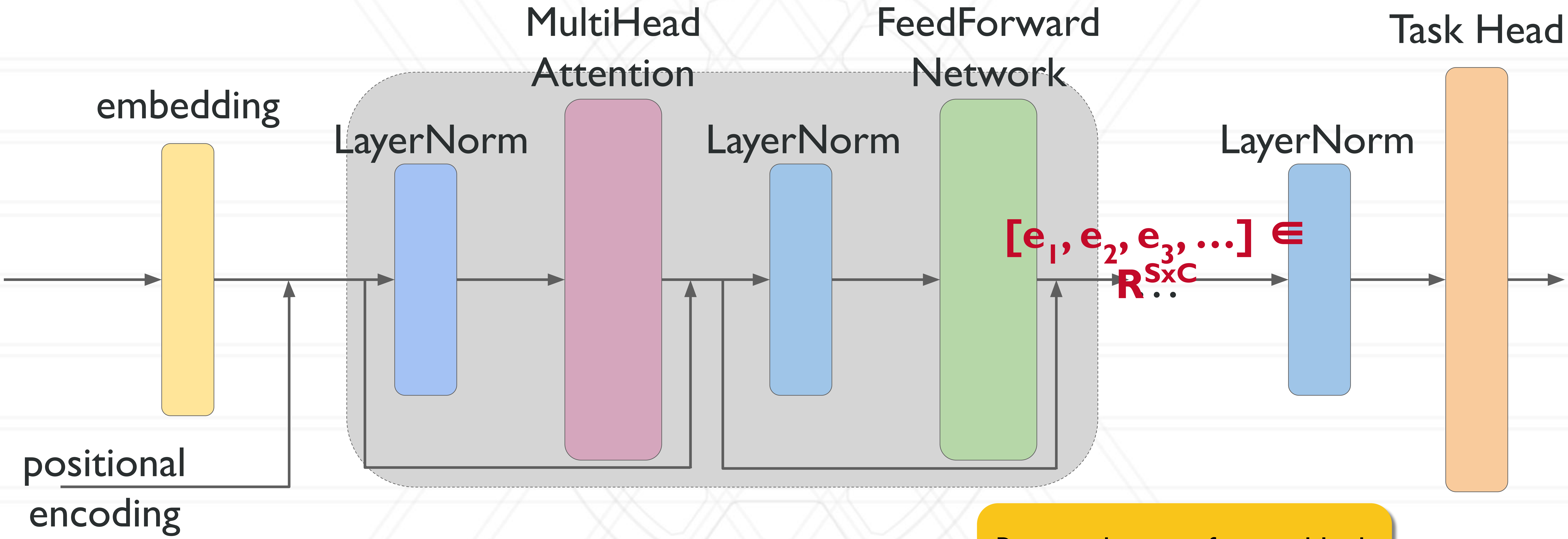


Normalize each row i.e. sequence independently

An Example Architecture

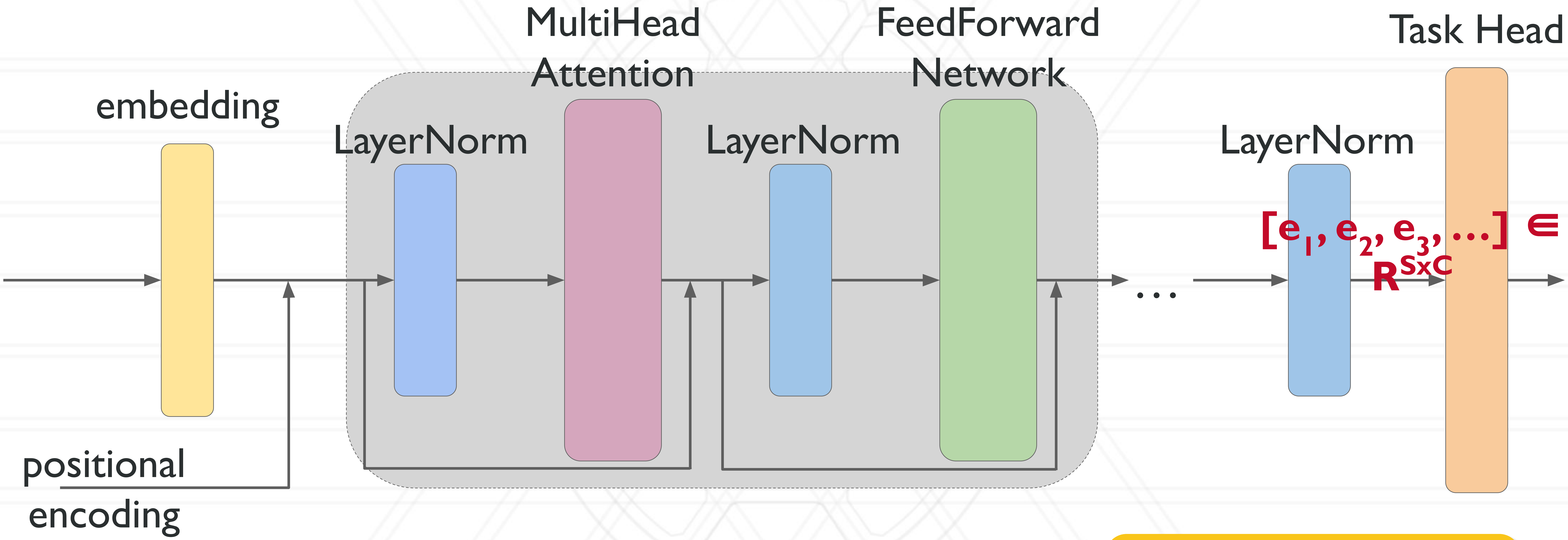


An Example Architecture



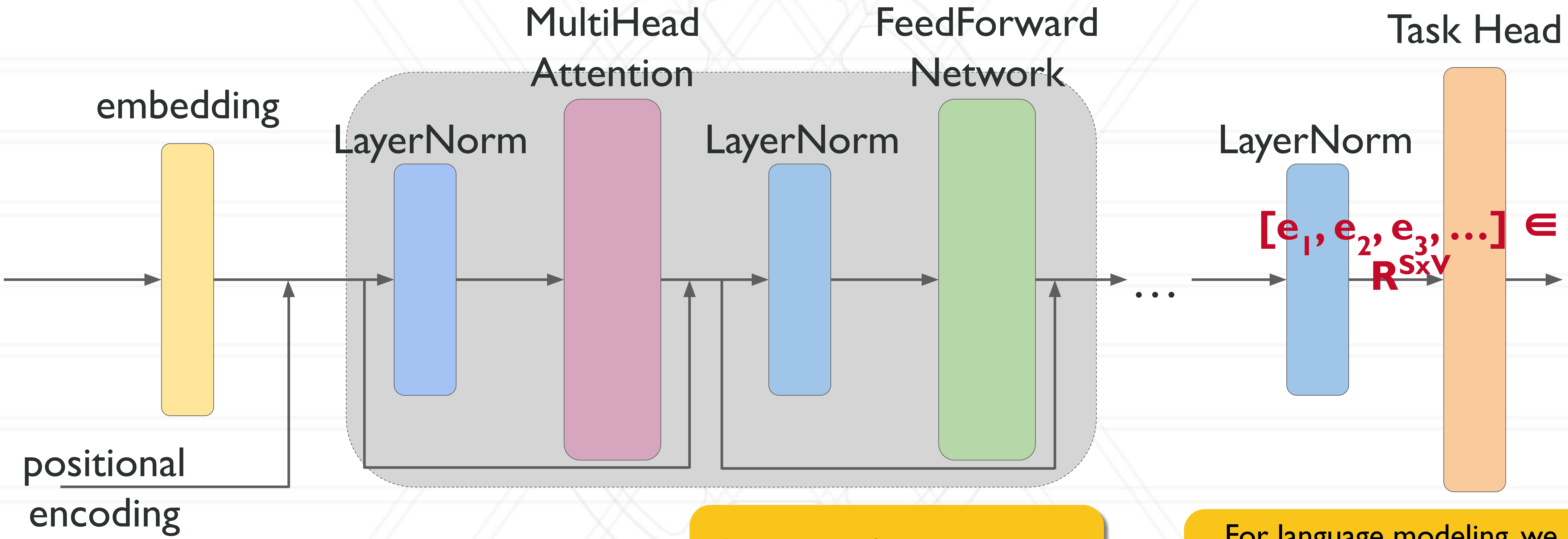
Repeat the transformer block many times

An Example Architecture



Normalize each row i.e. sequence independently

An Example Architecture



A task specific linear layer at the end

For language modeling, we project back to size of the vocab ***V***

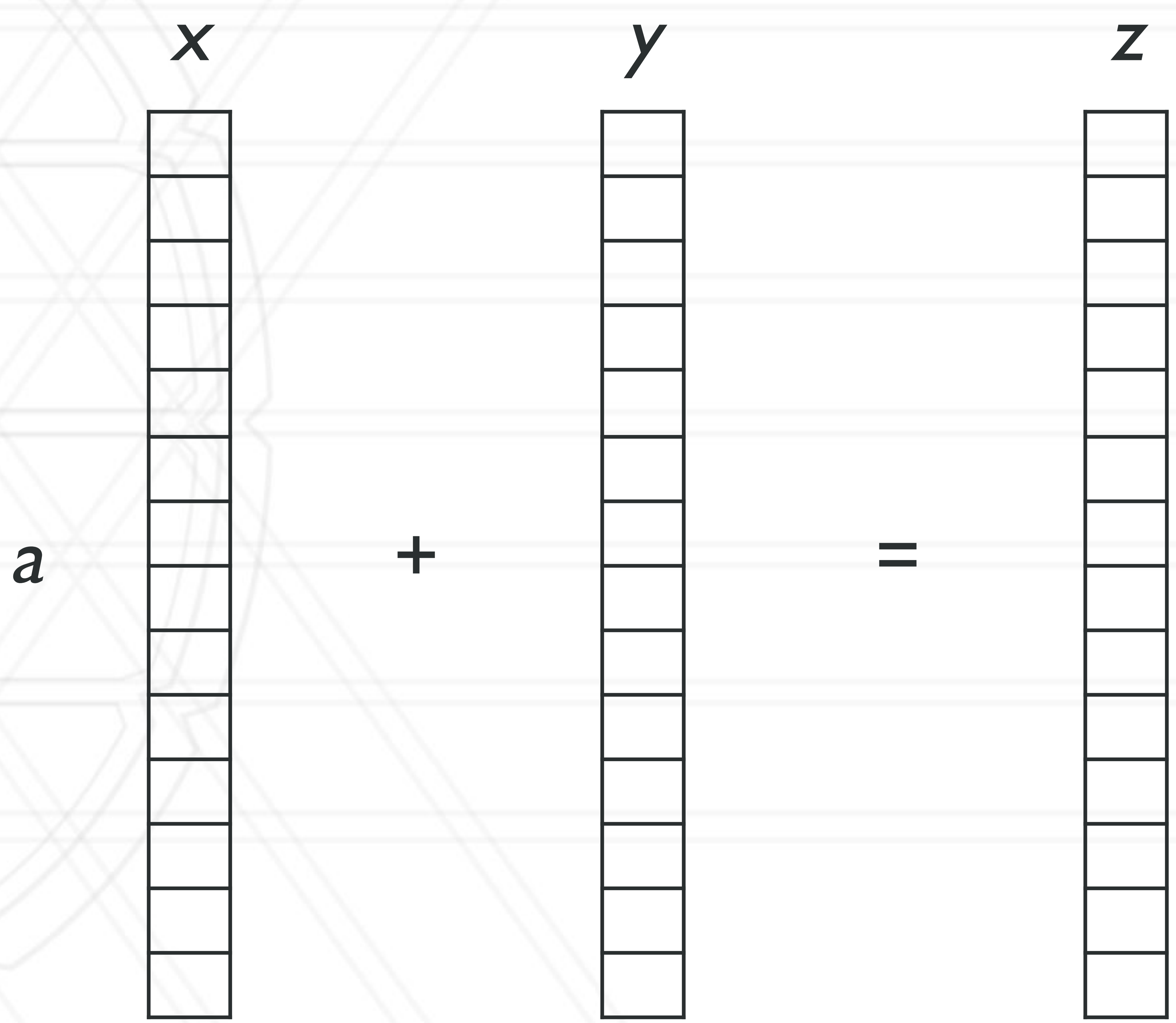
Performance Metrics

- Wall time
  - Fixed time-to-solution
- Time per batch
- Throughput
- Utilization
- Flops/s
- Peak memory used
- Parallel speedup and efficiency

Flops/s

- Flops/s – floating point operations per second
- Theoretical vs achieved
- Function of the algorithm, data, and hardware
- Example: saxpy – a*x+y

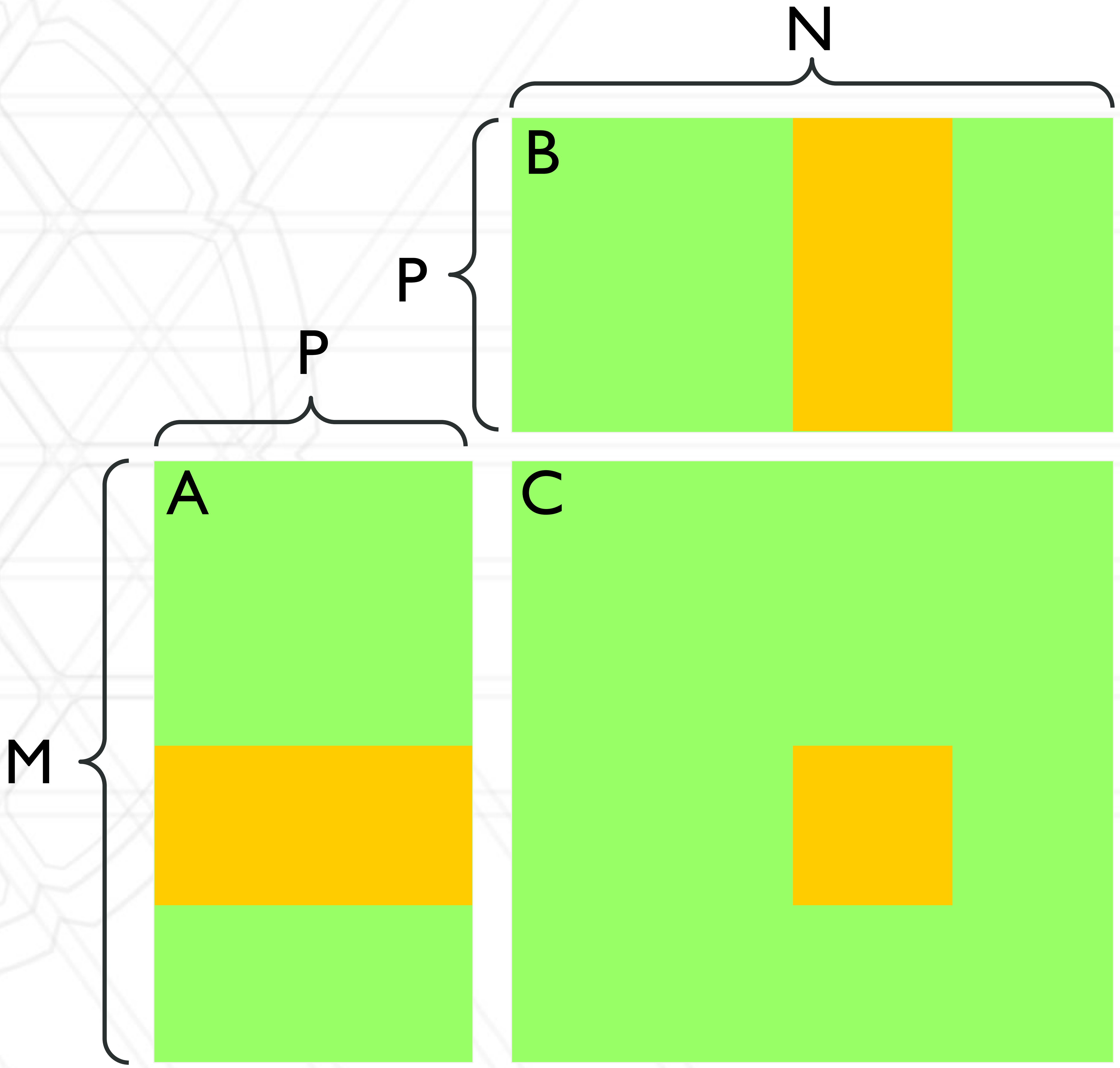
$$flops = 2N$$



Flops/s

- Flops/s – floating point operations per second
- Theoretical vs achieved
- Function of the algorithm, data, and hardware
- Example: matrix multiplication

$$flops = 2MPN$$

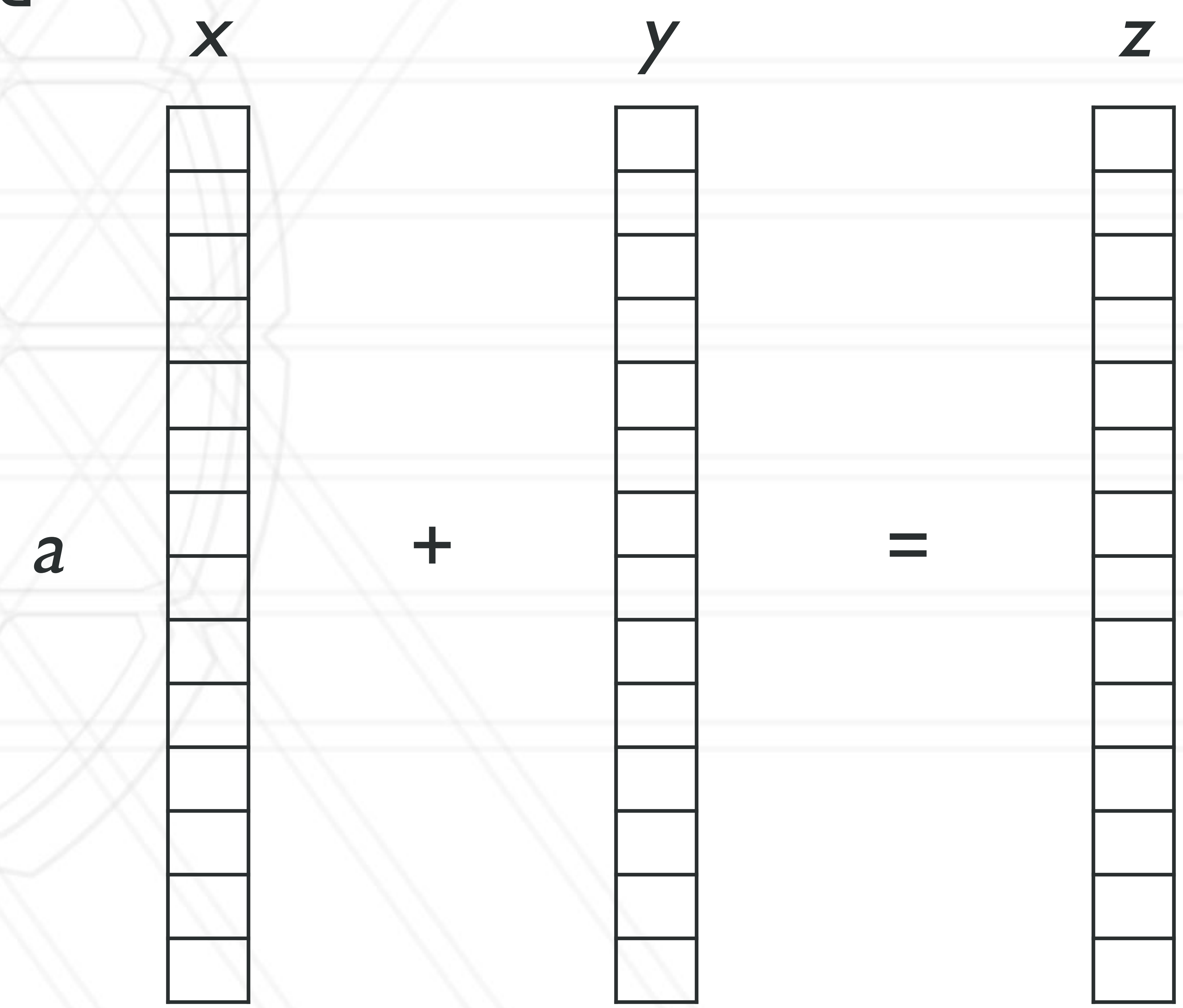


Compute vs Memory Bound

- Is code being bottlenecked by data loading or compute?
- Important for algorithm design, optimizations, and hardware selection
- Arithmetic intensity
  - Ratio of arithmetic instructions to bytes loaded
  - Property of algorithm

$$flops = 2N$$

$$\begin{aligned} traffic &= \left[(2 \text{ loads})\left(8\frac{\text{bytes}}{\text{load}}\right) + (1 \text{ store})\left(8\frac{\text{bytes}}{\text{store}}\right)\right] \cdot N \\ &= 24N \end{aligned}$$



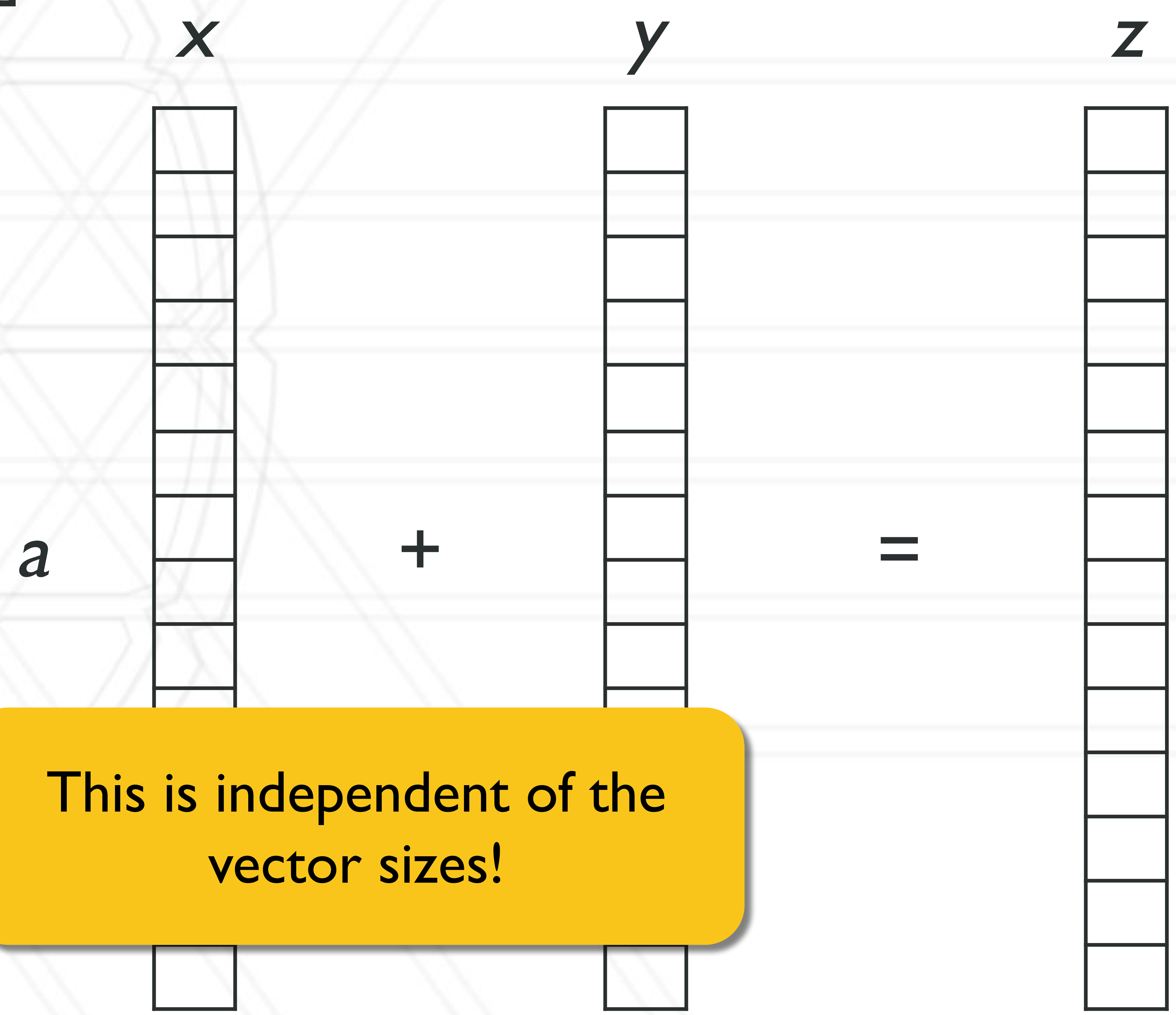
Compute vs Memory Bound

- Is code being bottlenecked by data loading or compute?
- Important for algorithm design, optimizations, and hardware selection
- Arithmetic intensity
  - Ratio of arithmetic instructions to bytes loaded
  - Property of algorithm

$$flops = 2N$$

$$traffic = 24N$$

$$AI = \frac{flops}{traffic} = \frac{1}{12}$$



This is independent of the vector sizes!

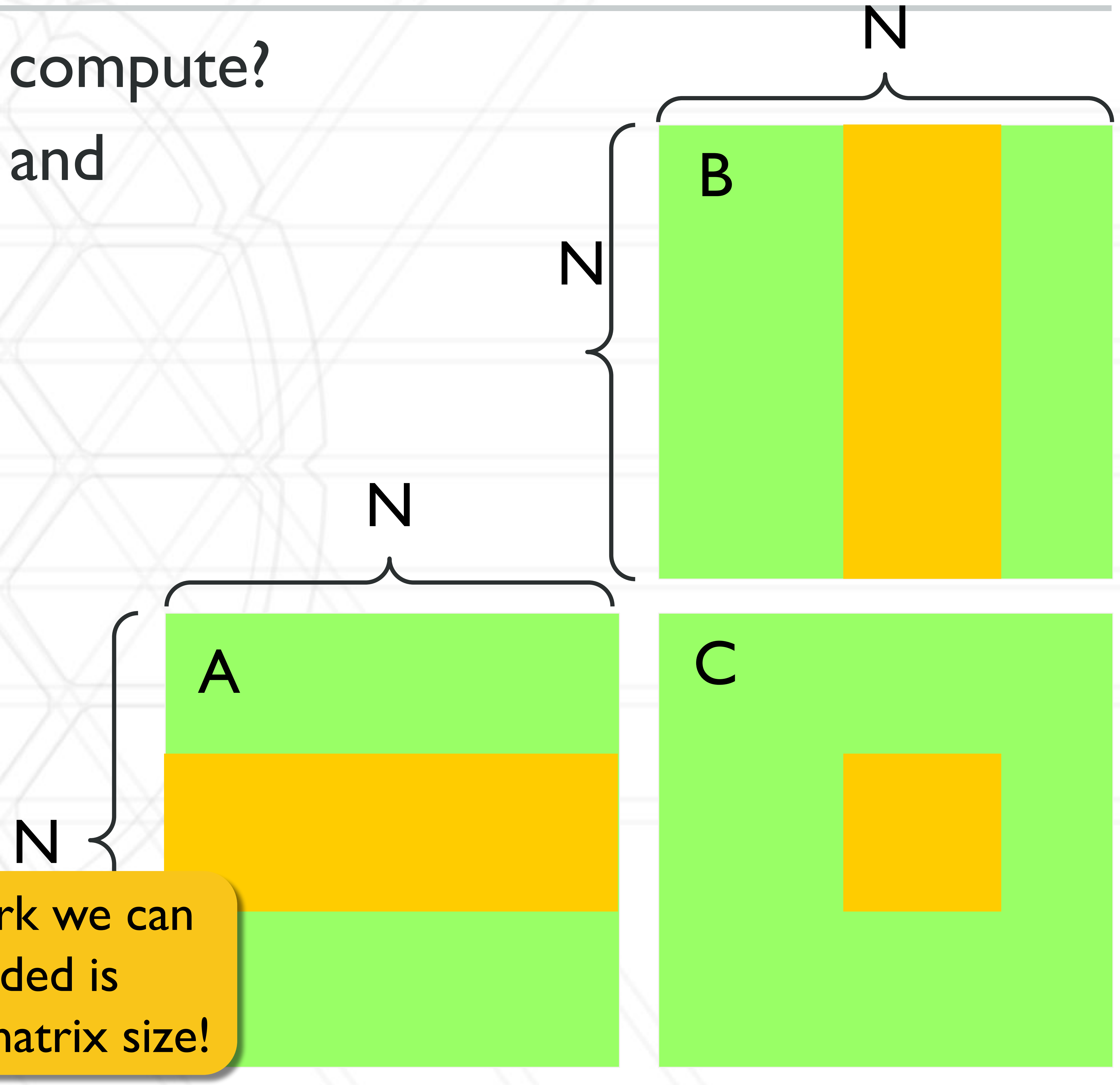
Compute vs Memory Bound

- Is code being bottlenecked by data loading or compute?
- Important for algorithm design, optimizations, and hardware selection
- Arithmetic intensity
  - Ratio of arithmetic instructions to bytes loaded
  - Property of algorithm

$$flops = 2N^3$$

$$traffic = 4 \cdot 8 \cdot n^2$$

$$AI = \frac{n}{16}$$



The amount of work we can do per data loaded is dependent on the matrix size!

Compute vs Memory Bound

- Is code being bottlenecked by data loading or compute?
- Important for algorithm design, optimizations, and hardware selection
- Arithmetic intensity
  - Ratio of arithmetic instructions to bytes loaded
  - Property of algorithm
- Attainable performance
  - Hardware has peak performance π flops/s
  - Hardware has peak memory bandwidth β bytes/s

$$\text{AttainablePerformance}(AI) = \min\{\pi, \beta \cdot AI\}$$

Compute vs Memory Bound

- Is code being bottlenecked by data loading or compute?
- Important for algorithm design, optimizations, and hardware selection
- Arithmetic intensity
  - Ratio of arithmetic instructions to bytes loaded
  - Property of algorithm
- Attainable performance
- Roofline model

Usually applied to individual kernels, but can be applied to groups of operations as well

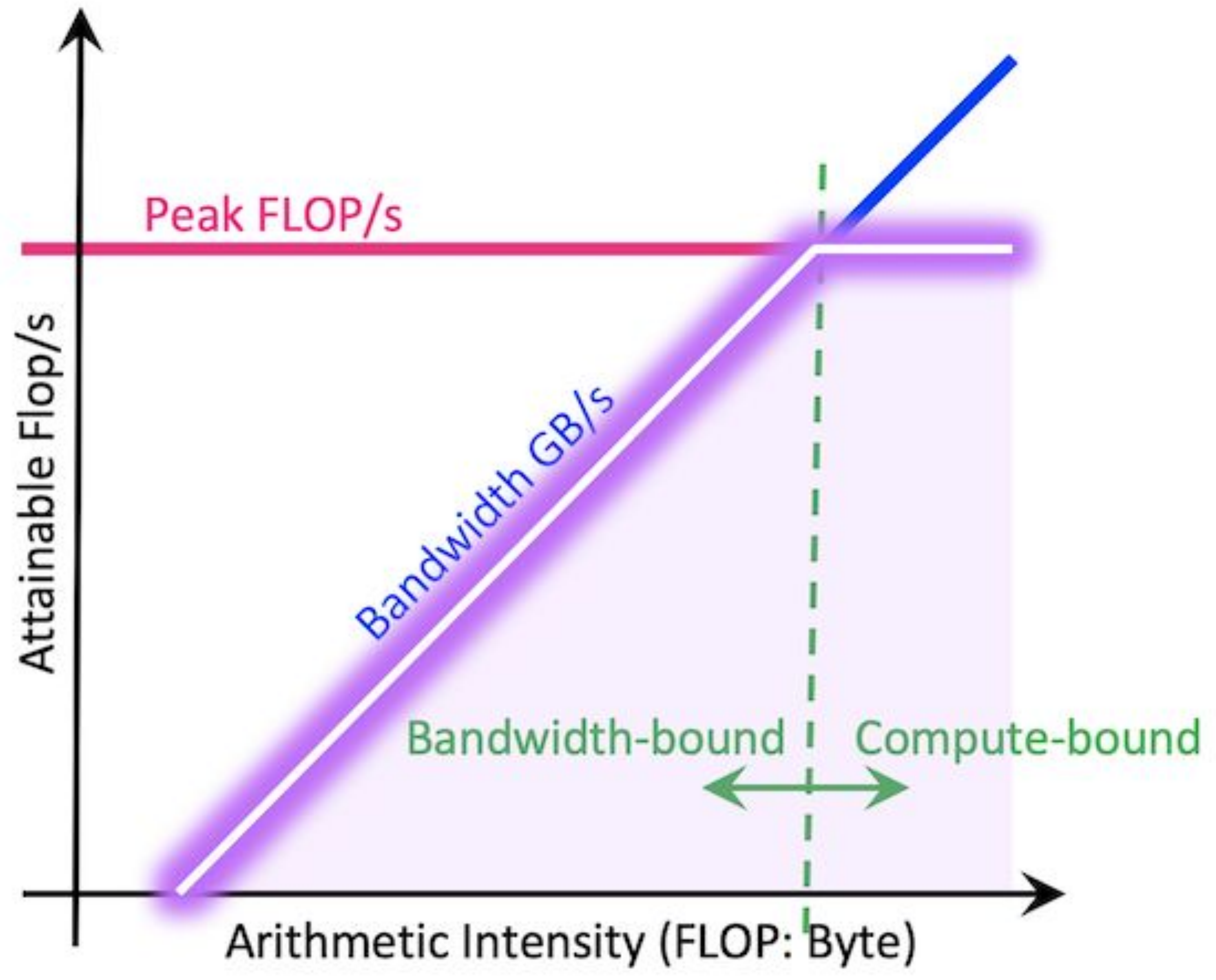
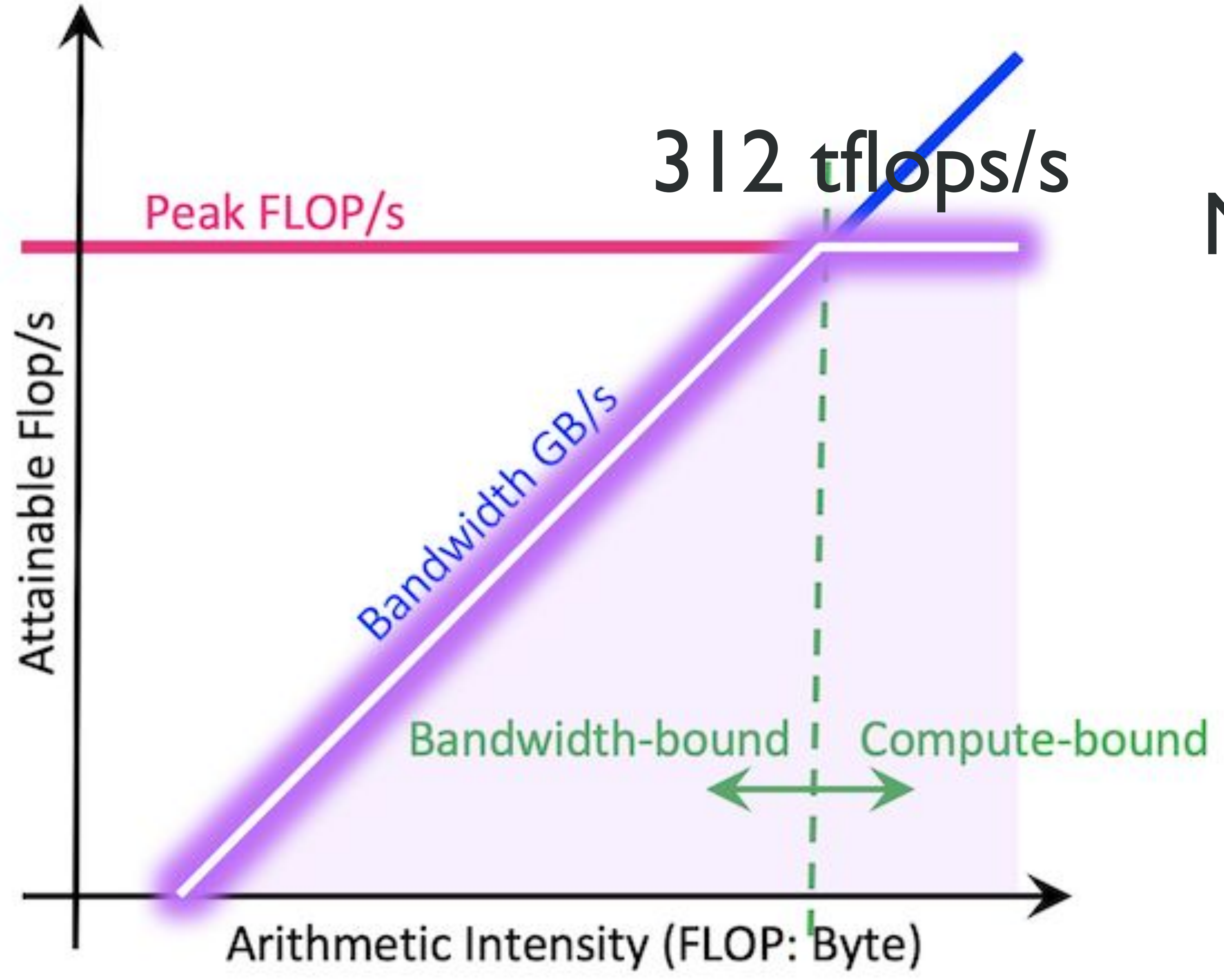


image: https://docs.nersc.gov/tools/performance/roofline/

An Example Roofline Model

- A100
  - 312 teraflops fp16 performance
  - 1555 GB/s bandwidth
- Matrix Multiplication

$$\text{AttainablePerformance}(AI) = \min\left\{312, \frac{n}{16} \cdot 1.5\right\}$$

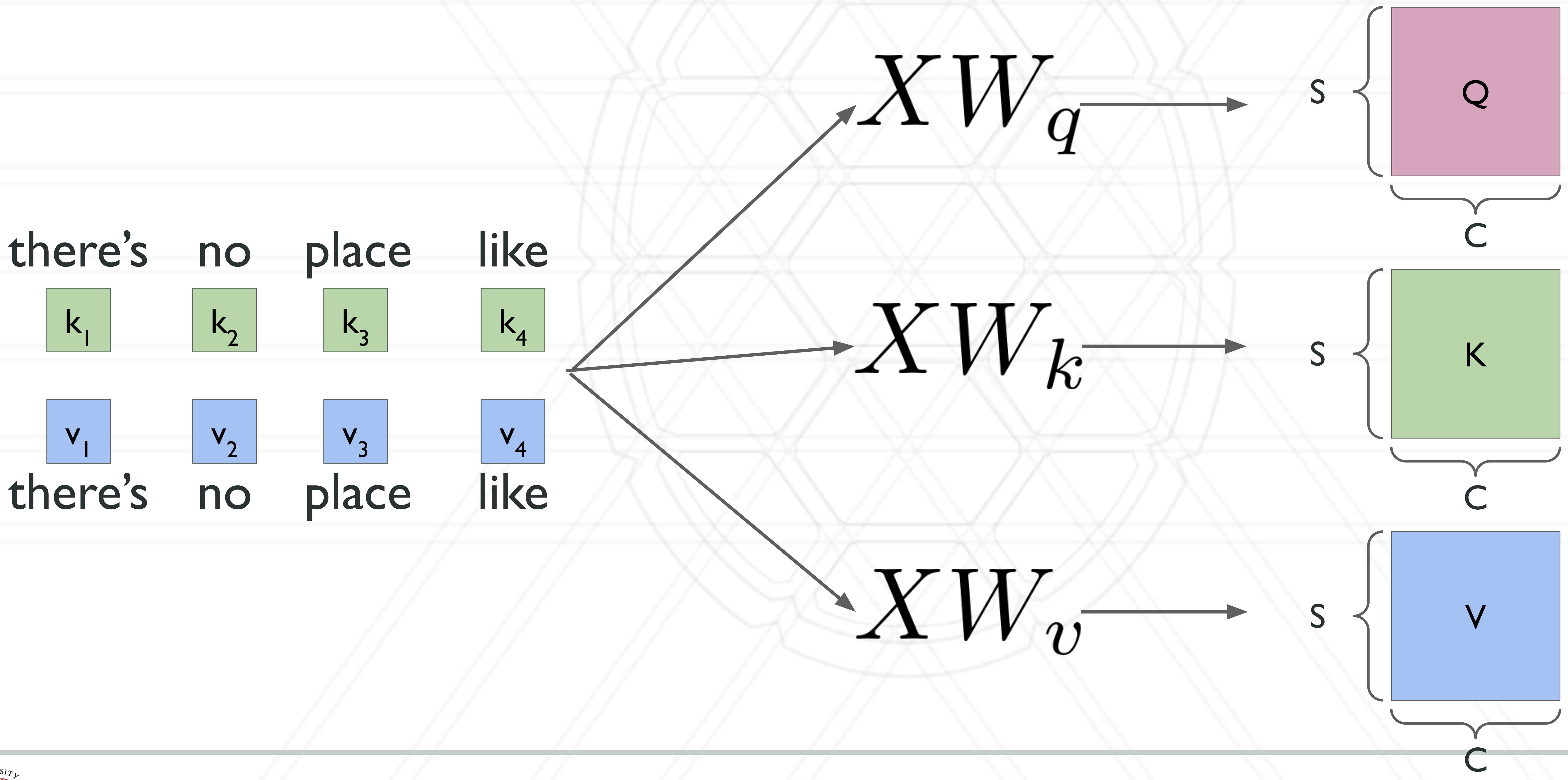


N=3328

Past N=3328 we are now compute bound

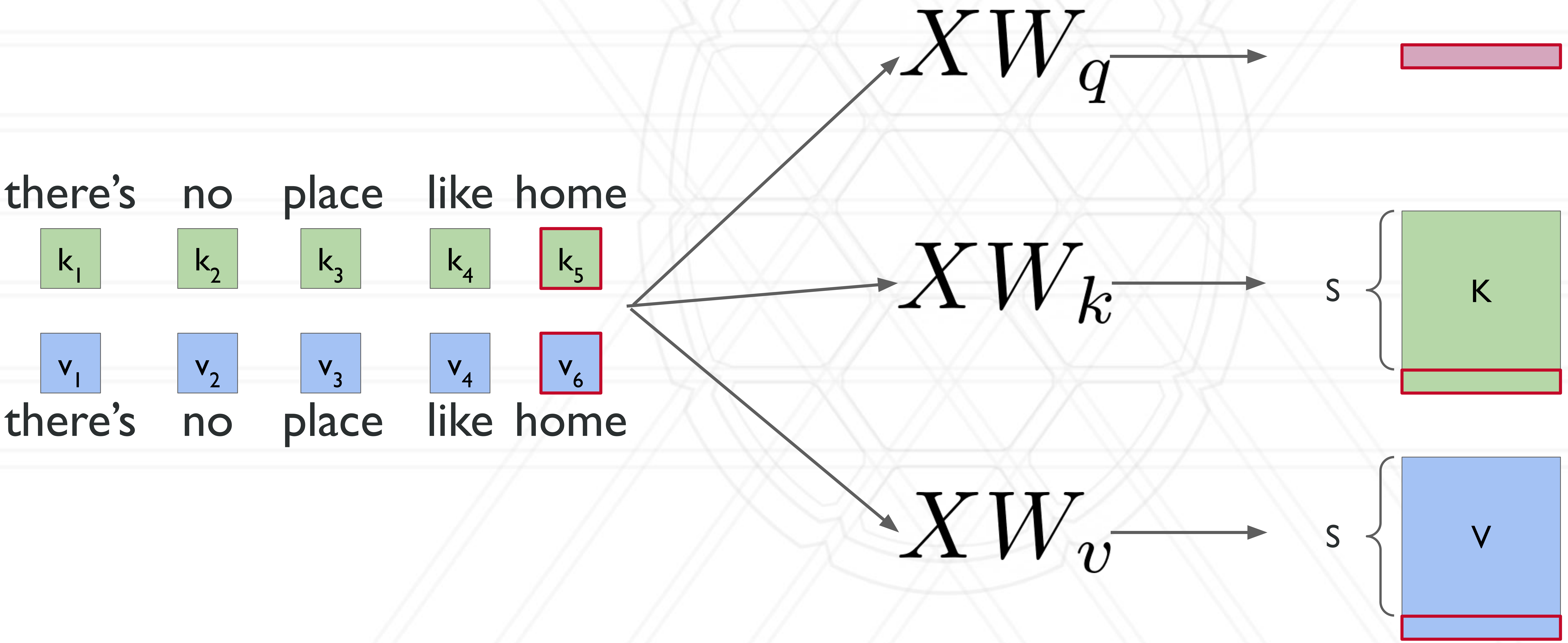
KV Caching and Performance Models

- We recompute KV values when we run models in an autoregressive manner



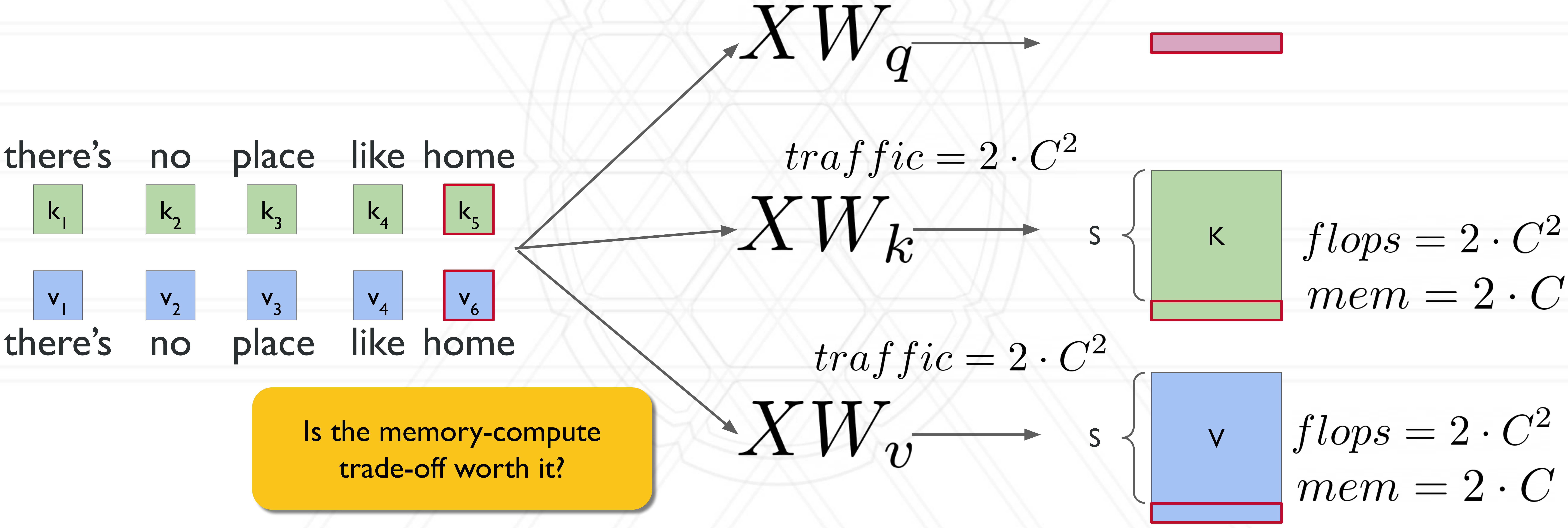
KV Caching and Performance Models

- We recompute KV values when we run models in an autoregressive manner



KV Caching and Performance Models

- We recompute KV values when we run models in an autoregressive manner



Is the memory-compute trade-off worth it?

KV Caching and Performance Models

- Arithmetic intensity of KV calculation for new token

$$flops = 2 \cdot C^2 \qquad traffic = 2 \cdot C^2$$

$$AI = 1$$

We're in the memory bound region

$$\text{AttainablePerformance} = \min\left\{312 \cdot 10^{12} \frac{\text{flops}}{\text{s}}, \left(1 \frac{\text{flops}}{\text{byte}}\right) \cdot \left(1.5 \cdot 10^{12} \frac{\text{bytes}}{\text{s}}\right)\right\}$$

$$\frac{\pi}{\beta} = 208$$

Ridge: computing KV for one token takes the same as 208 tokens

Chen, Carol "Transformer Inference Arithmetic"

GPU Utilization

- How effectively are you using your resources?
- Occupancy
  - Ratio of active warps to total per SM
- Utilization
  - Percentage of samples where kernel is running
  - torch.cuda.utilization()

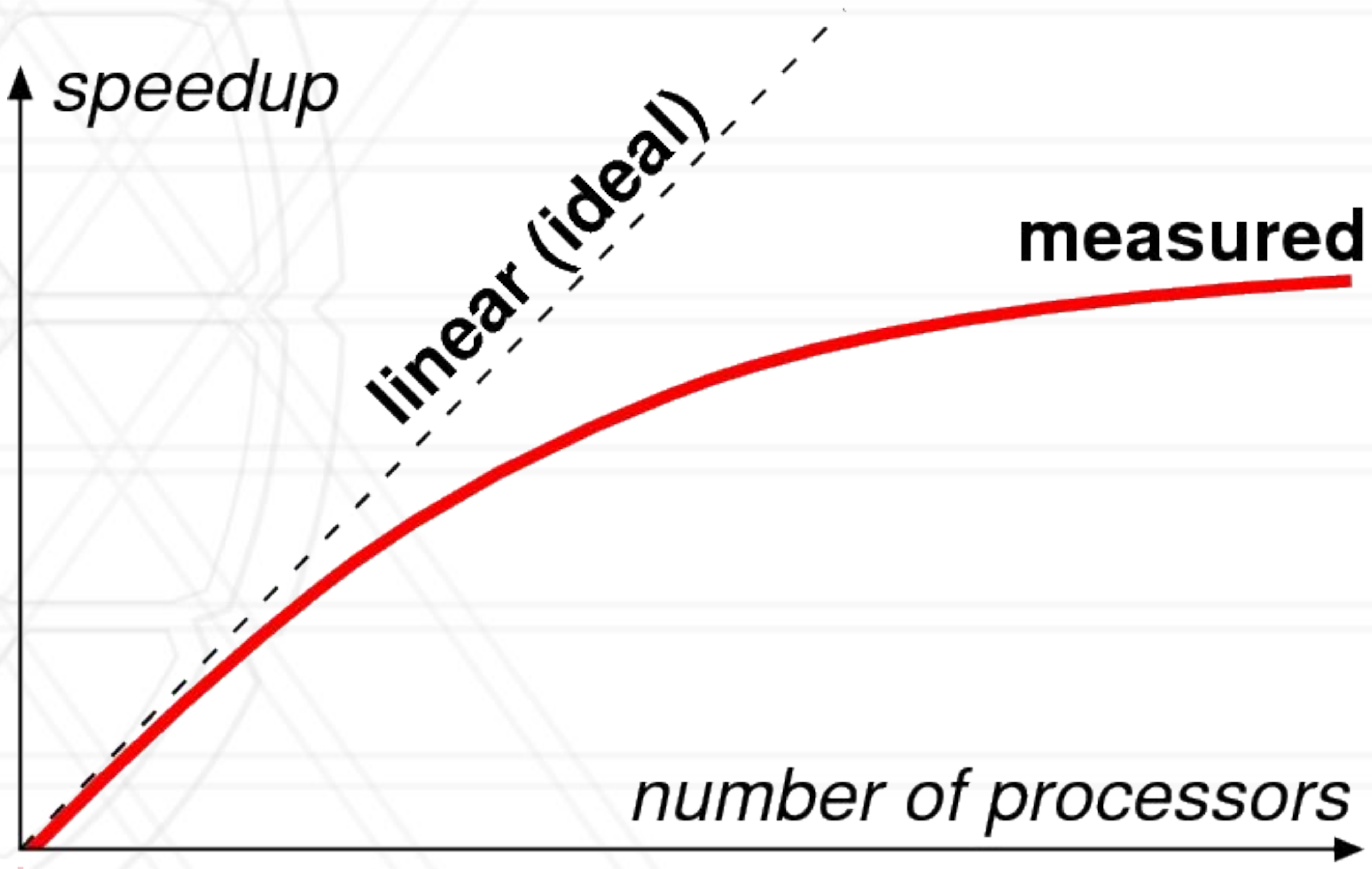
Parallel Scaling and Efficiency

- Speedup

$$S(p) = \frac{T(1)}{T(p)}$$

- Efficiency

$$E(p) = \frac{S(p)}{p}$$



Performance Space

- Performance depends on more than just processor count

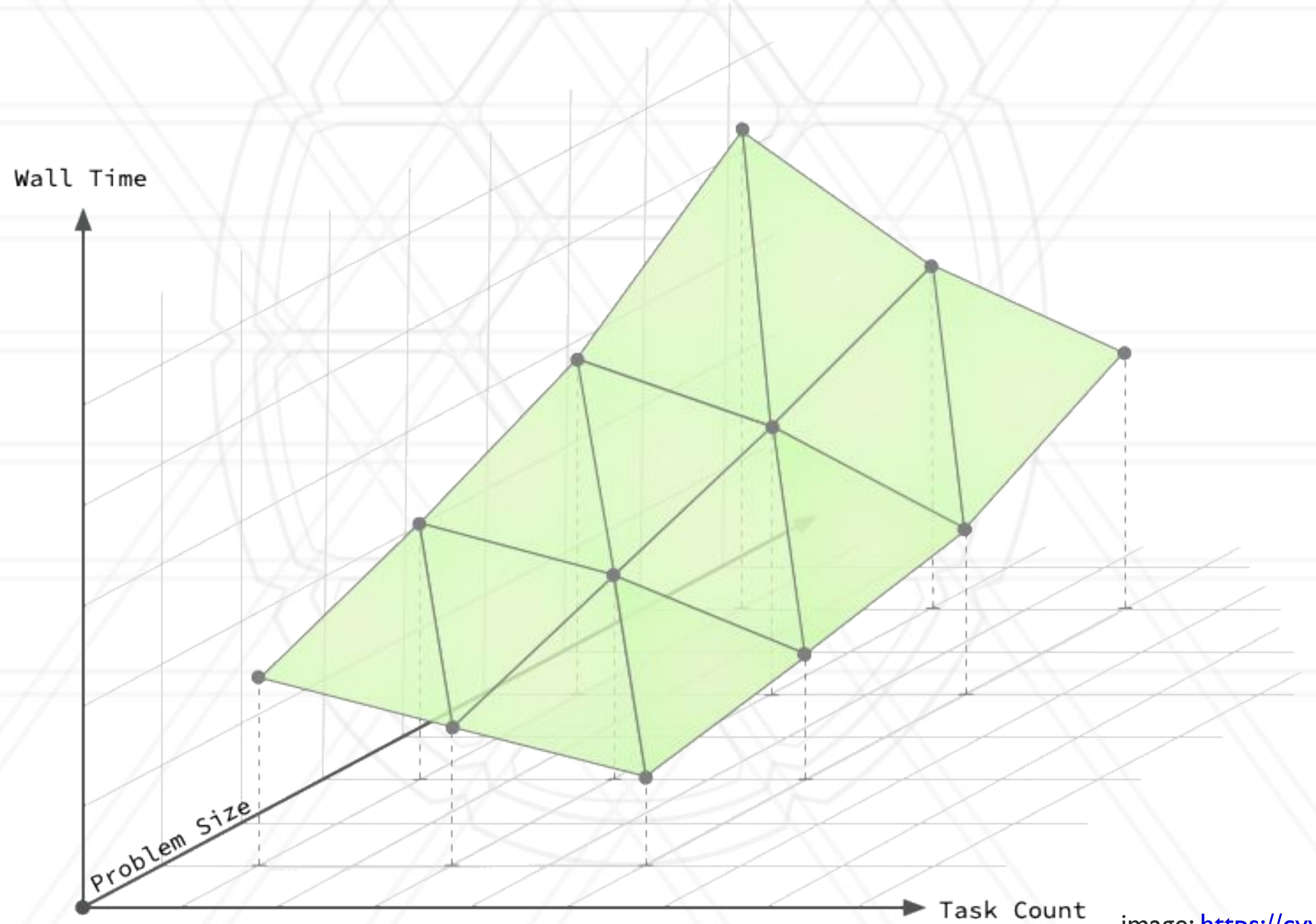
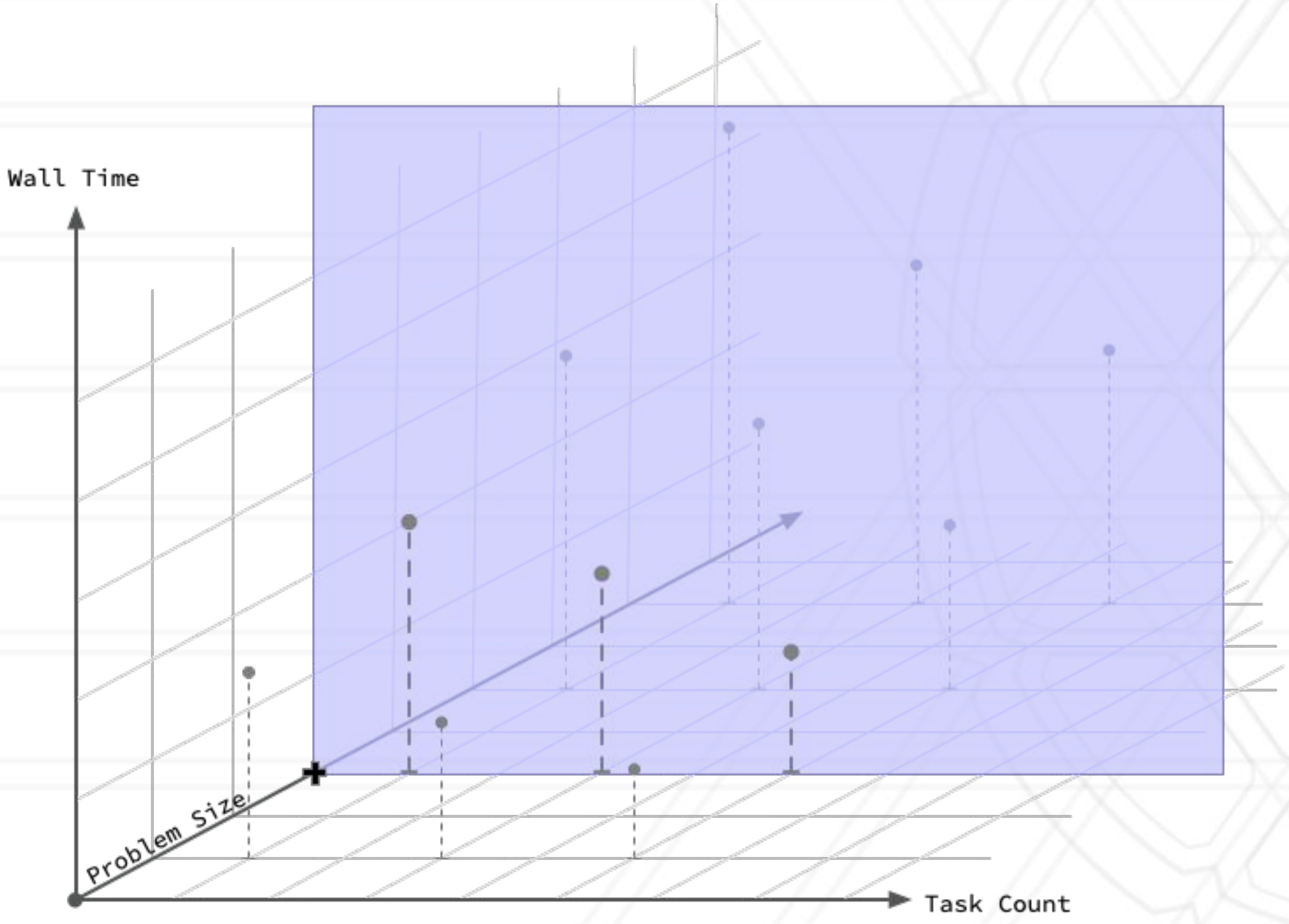


image: https://cvw.cac.cornell.edu/parallel/efficiency/scaling

Strong Scaling

- Increase resources, fix problem size



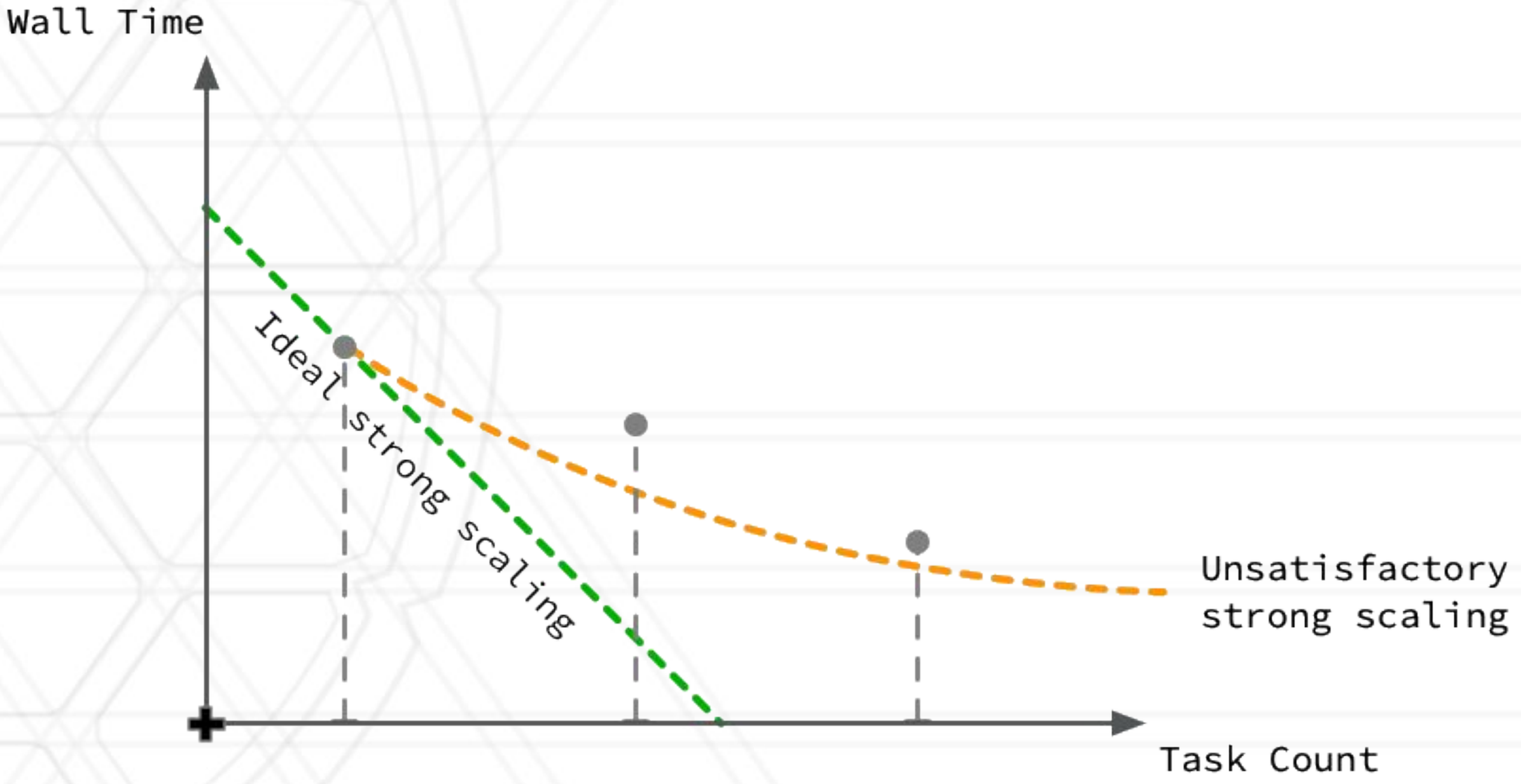
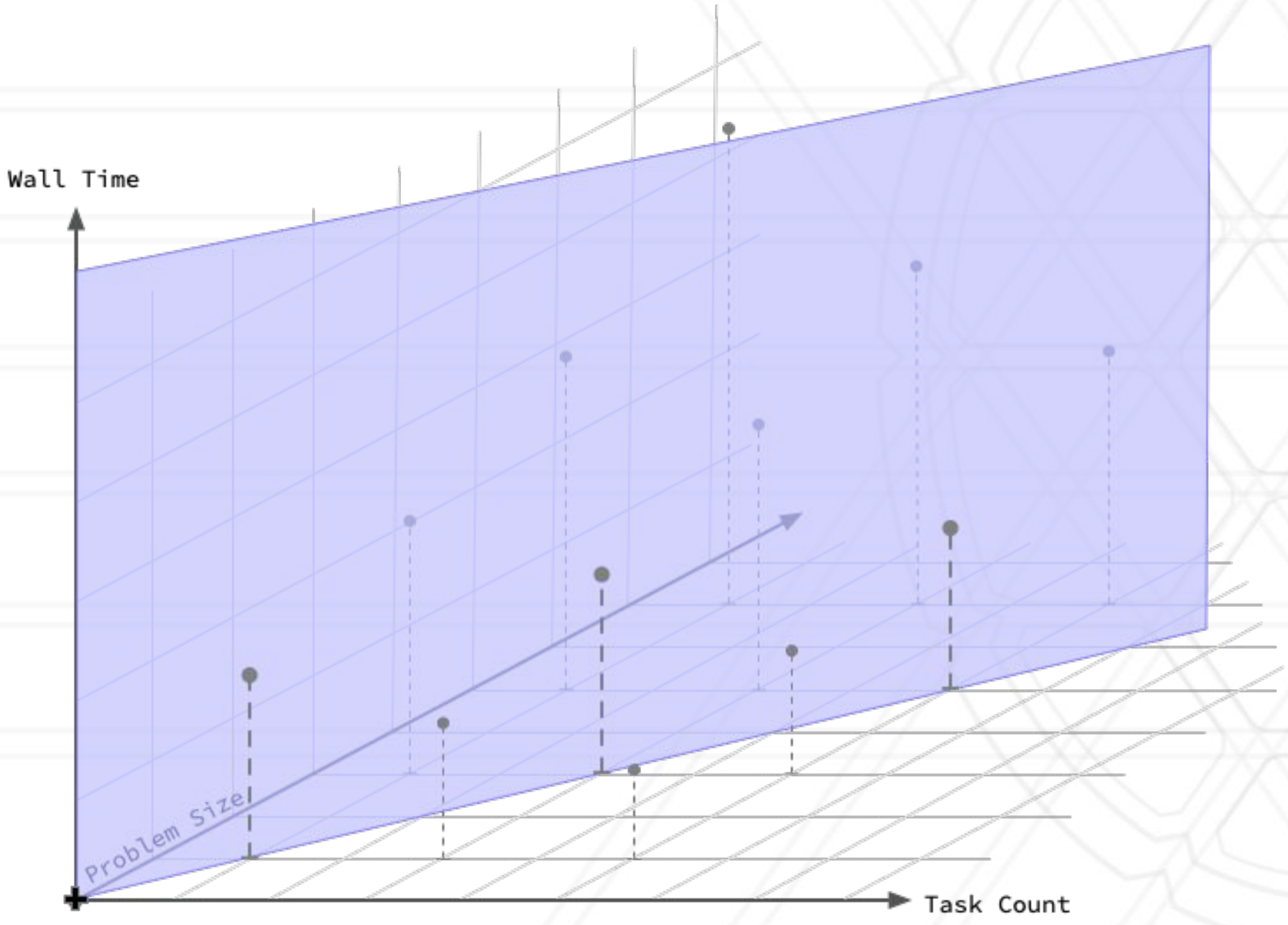


image: https://cvw.cac.cornell.edu/parallel/efficiency/scaling

Weak Scaling

- Increase problem size proportional to resources



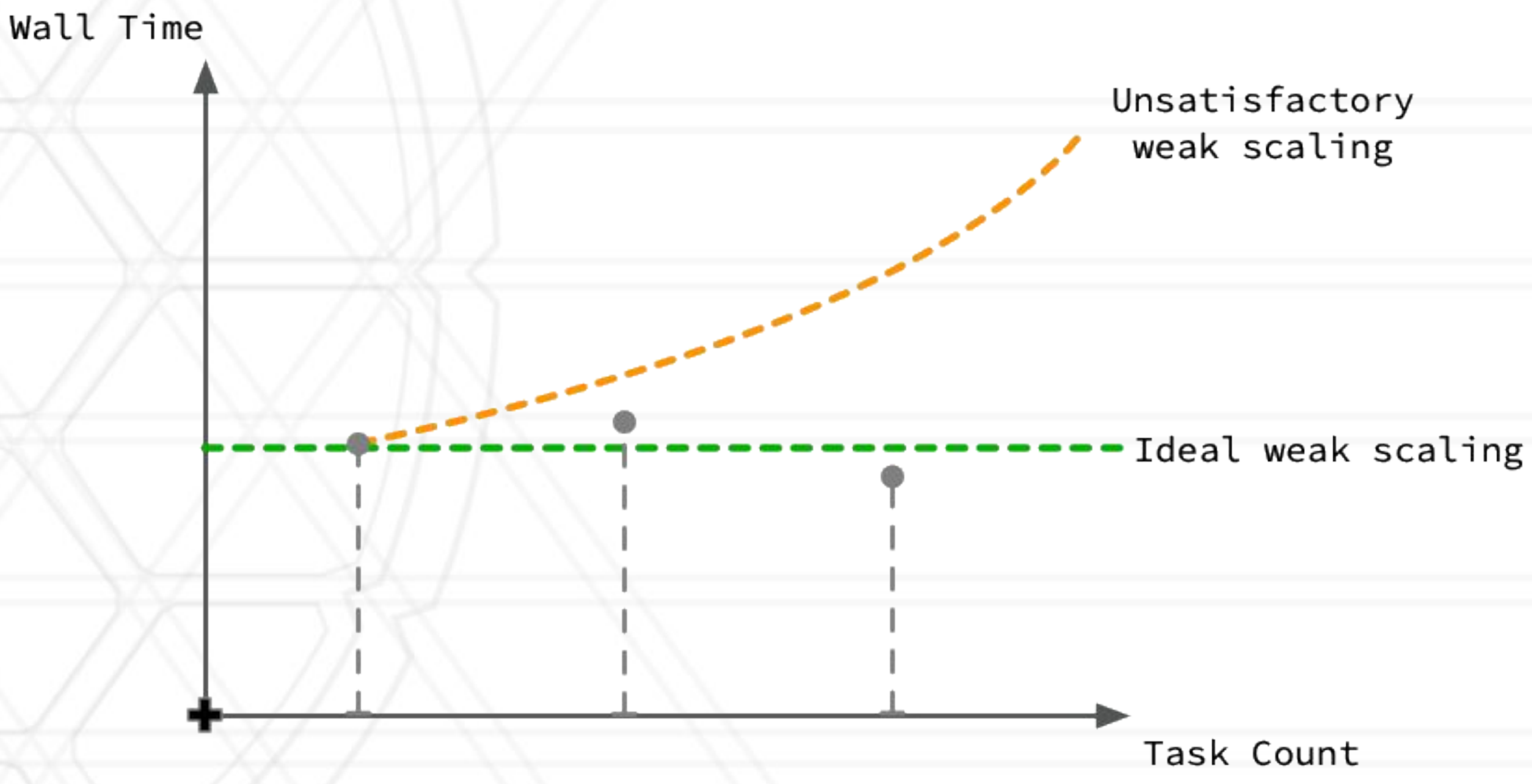


image: https://cvw.cac.cornell.edu/parallel/efficiency/scaling

Profiling

- PyTorch provides a profiler internally for measuring performance
- Simple python context
- https://pytorch.org/docs/main/profiler.html

```
with profile(activities=[ProfilerActivity.CPU], record_shapes=True) as prof:
    with record_function("model_inference"):
        model(inputs)
```

Profiling

- PyTorch provides a profiler internally for measuring performance
- Simple python context
- https://pytorch.org/docs/main/profiler.html

```
with profile(activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA],
             record_shapes=True,
             on_trace_ready=torch.profiler.tensorboard_trace_handler(dir_name)
            )as prof:
    with record_function("model_inference"):
        model(inputs)
```

Output for inspection in tensorboard

UNIVERSITY OF
MARYLAND