Parallelism

Extracting more parallelism is arguably main goal of most systems research for the last decade.

(Pervasive Parallelism lab @ Stanford)

1. The main trends (from 5 years ago...) (Slides)
2. What a modern training setup looks like
3. Data, model & pipeline parallel

Goal: keep it high level, give you a taste

Disclaimer: I was active in this area in research
→ Biased slides
have cofounded, funded companies in space

⇒ to the slides!

What does a modern large training setup look like? (Terms)

Single GPU, TPU, ADU whatever

5 years ago, we were excited about
125 TFLOP → 1000 TFLOP (1 PF)
(this year, half precision)

Largest supercomputer in Canada is 5.6 PF
Germany is 44 PF

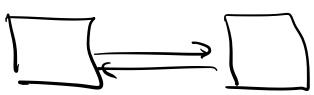
NOTE: NOT comparable (Built on 5 yo tech, limited $)

⇒ Specialization & Rapid Growth
⇒ Also creating: many GPUs contain multiple chips inside (soon)
⇒ Contain lots of high bandwidth memory (16, 32, ...) (HBM) 2016 AMD

Many GPUs → Box. (8 or 16) Nvidia calls DGX.

→ Plug into PCI-E

NVLink 200 GB/s
L2 P2P network.



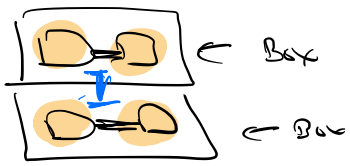
↑ ~60 GB/s (4 TB/s with HBM)
per lane.
16 lanes ~ 1 TB/s (whole device)

Boxes connected into racks. (4 per rack)

Racks connected into "bays" → Datacenter depended.

TERMS

CHIP
↓ MANY (8-16x)
BOX
↓ MANY (4x)
RACK
↓ MANY
POD



Key Challenges

1. Compute (How much can you use?) ~ 10-15% safe!
2. Memory (Does your model fit HBM → Spread out)
3. Network (loosely at all levels)

We've codesigned models to extract performance.

Example Matrix multiply is highly parallel

⟹ can get 80% utilization or more!

→ teams of folks took years for attention.

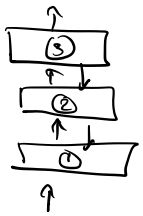
"typical code" runs at 10-15% out of the box.

MAJOR TECHNIQUES

→ Data Parallel

→ Model & / Pipeline Parallel

Data Parallel

INPUT 

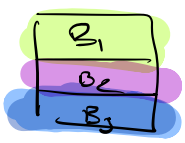
model w/ 3 "layers"

computes $\ell(w, x)$ — x: DATA, w: WEIGHTS

AND 3 NODES (CHIPS, BIKES, WHATEVER) think CUDA for cooking

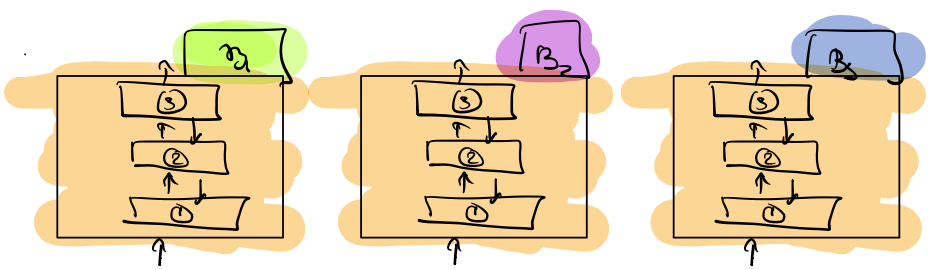
Goal

Compute a Batch of data B $\sum_{x \in B} \ell(w, x)$



And it's gradient wrt w.

$$\frac{\partial L(w, B)}{\partial w} = \sum_{x \in B} \frac{\partial \ell(w, x)}{\partial w}$$



Straightforward! $L(w, B) = L(w, B_1) + L(w, B_2) + L(w, B_3)$

$$\frac{\partial L}{\partial w}(w, B) = \frac{\partial L}{\partial w}(w, B_1) + \frac{\partial L}{\partial w}(w, B_2) + \frac{\partial L}{\partial w}(w, B_3)$$

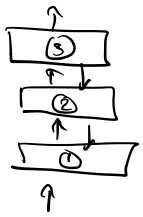
1. SEND BATCHES
2. EACH COMPUTE (IN PARALLEL)
3. SEND BACK GRADIENT (to ONE NODE OR HOST)
4. UPDATE MODEL WEIGHTS (take a step)
5. BROADCAST WEIGHTS BACK TO EACH WORKER

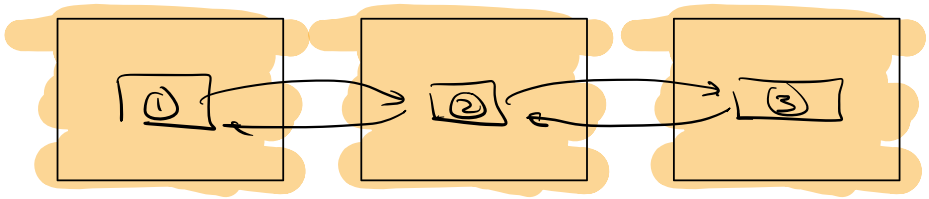
UTILIZATION VERY high if NODE IS (AMORTIZED WORK IN BATCHES)

Communication
ENTIRE MODEL SENT 3 TIMES (+3 TIMES for GRADIENT)

REQUIREMENT: Model fits in a NODE. GPT3-??! NO CHANCE!

Model "Parallelism"





Only each layer and it's weights need to fit.

Recall "Forward and Back"

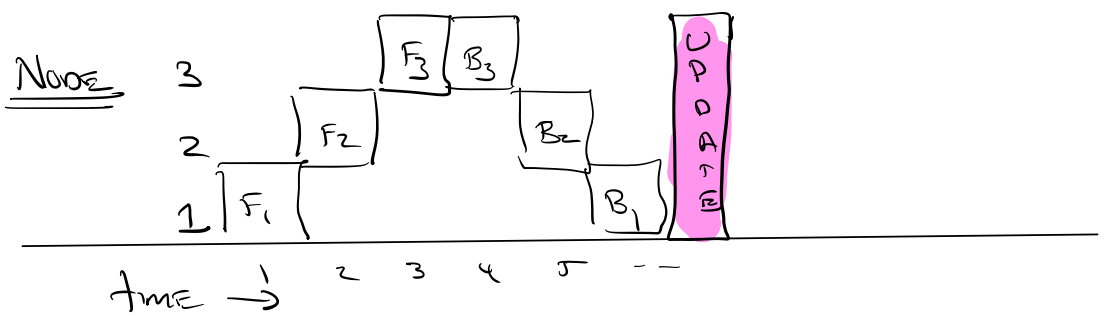
$z_1 = f_1(w_1, x)$

$z_2 = f_2(w_2, z_2)$ Forward pass easy to compute

$z_3 = f_3(w_3, z_2)$

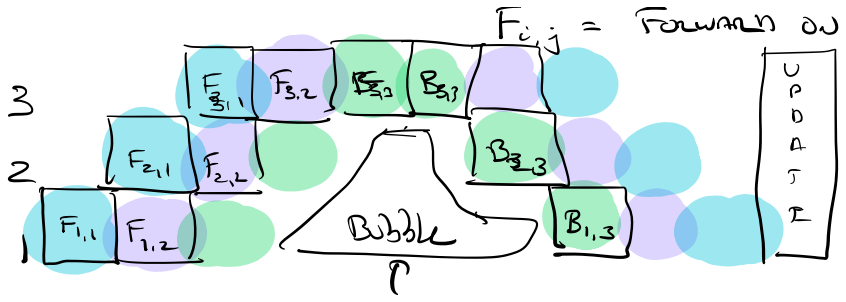
Backward (example) $\frac{\partial z_3}{\partial w_2} = \frac{\partial f_3(w_3, z_3)}{\partial} \cdot \frac{\partial z_2}{\partial w_2}$

↳ not computed until forward pass



Not parallel at all!

Idea Micro batches Smaller batches to overlap (pipe)



WHITE SPACE IS WASTED 12 WHITE SQUARES

3 · 3 · 2 18 SQUARES W/ WORK

$\Rightarrow \frac{18}{30}$ · PER NODE UTILIZATION

$\approx 60\%$ · ——

$\Rightarrow$ PIPEMARE, PIPEDREAM SHOW HOW TO IMPROVE.

KEY IDEA: ONE-SHOT, HOLISTIC!

Other Optimizations

→ Communication

→ Reduced Precision