GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding

Zhifeng Chen
Google Inc.

Hot Chips 2020 Tutorial on “Machine Learning Scaleout”
08/16/2020
Big models

#weights (log scale)
Google Neural Machine Translation
Our goal

Develop a universal machine translation model (i.e. one model for all languages and domains)

“Perhaps the way [of translation] is to descend, from each language, down to the common base of human communication -- the real but as yet undiscovered universal language -- and then re-emerge by whatever particular route is convenient.”

Warren Weaver (1949)
Motivation 1: Improve translation quality for all language pairs

Data distribution over language pairs

- **High-resource languages**
  - {Spanish, French, German, ...}
  - Approaching human quality (> 100M examples)

- **Low-resource languages**
  - {Yoruba, Sindhi, Hawaiian, ...}
  - Often not usable (< 1M examples)

25+ Billion Training Examples
Motivation 2: Expand language coverage

In the world, there are...

- **7,000+** Total languages
- **2,000+** African languages
- **700+** Native Am. languages

But Translate only supports...

- **103** Total languages
- **11** African languages
- **0** Native Am. languages

---

1. Estimate is 766 Native Am languages globally: 383 in South America (source); 176 in Central America (source); 207 in North America (source)
Motivation 3:
Neural network scaling and the new understanding of generalization
Motivation 3: Neural network scaling and the new understanding of generalization
Motivation 4:
This is a compelling test bed for ML research

Massive multilinguality requires advances in:
- Multi-task learning
- Meta-learning
- Continual learning

To achieve massive multilinguality, we need massive scale, requires advances in:
- **Model capacity**
- Trainability and optimization
- Efficiency improvements
Progress and Future
Number of Synapses

Fruit fly  Honey bee  Mouse  Cat  Macaque  Human

#synapses

[wiki]
Number of Synapses

- Fruit fly
- Honey bee
- Mouse
- Cat
- Macaque
- Human

NMT with Attention
ResNet50
[25-50M]

#synapses [wiki]
Number of Synapses

- Fruit fly
- Honey bee
- Mouse
- Cat
- Macaque
- Human

GNMT production model (2016)
200M

#synapses
[wiki]
Number of Synapses

Transformer [400M]

Fruit fly          Honey bee          Mouse          Cat          Macaque          Human

#synapses [wiki]
Number of Synapses

Transformer [400M]

Facebook ResNeXt101
Open AI GPT2
MSR ZeRO
NVidia Megatron-LM
Google-T5 [1-10B]

M4: Massively Multilingual Massive Machine Translation [80B]

Number of Synapses

Fruit fly
Honey bee
Mouse
Cat
Macaque
Human

#synapses [wiki]
Number of Synapses

- Fruit fly
- Honey bee
- Mouse
- Cat
- Macaque
- Human

GPT-3
- 170B

GShard
- [150B, 600B, 1.1T]

#synapses
- [wiki]
Big Data
Big Task
Big Model
Developer Complexity
Parallelism
Big Hardware

MoE Transformer
GShard API
Separation of Concerns
GShard Strategy
XLA extension
Transformer

- Powerful
  - Cores to many SOTA results.
- Simple
  - Easy to express in linear algebra.
  - Reproduced many many times.
- Originally proposed in the [paper](#)
Mixture of Experts (MoE)

- Sparsely gated
  - Cost-effective inference
- Embarrassingly parallelizable
  - Nice to accelerators
- Originally proposed in this [paper](#)
Mixture-of-Experts Transformer

Replace every-other FFN with an MoE layer
$x_s$ is the input token

$$G_{s,E} = \text{GATE}(x_s)$$

$$\text{FFN}_e(x_s) = w_{o_e} \cdot \text{ReLU}(w_{i_e} \cdot x_s)$$

$$y_s = \sum_{e=1}^{E} G_{s,e} \cdot \text{FFN}_e(x_s)$$

$E$ feed-forward networks $\text{FFN}_1 \ldots \text{FFN}_E$

$G_{s,E}$ is computed by a gating network.

$y_s$, is the weighted average
Algorithm details

- Gate function written in linear algebra
  - Easy to express in a sequential program
- Experts load balancing during training
  - Auxiliary loss helps
- Uniform routing during warming up phase
- Random second expert dispatch
- Flat beam search for inference
GShard Overview

- User **partially** annotates tensors in the TensorFlow graph
  - Called sharding annotations
  - Specifies which dimension to split, across how many devices
  - Specifies which tensor to be replicated
- The annotation is delivered to the XLA compiler (as HLO graph)
- XLA Propagates user-provided sharding information to the rest of the computation with some heuristics
  - Users are not required to annotate every tensor in the computation. Just need to give enough hints that the compiler can do this propagation well.
  - Weights need to be annotated if they need to be partitioned (default is replicate)
- Transforms each op in the XLA graph, following the determined input/output sharding decision
tf.einsum recap

# Matrix multiplication
```
einsum('ij,jk->ik', m0, m1)
```

# Transpose
```
einsum('ij->ji', m)
```
# Partition inputs along group (G) dim.
+ inputs = split(inputs, 0, D)

# Replicate the gating weights
+ wg = replicate(wg)

gates = softmax(einsum("GSM,ME->GSE", inputs, wg))
combine_weights, dispatch_mask = Top2Gating(gates)
dispatched_expert_inputs = einsum(
    "GSEC,GSM->EGCM", dispatch_mask, reshaped_inputs)
# Partition dispatched inputs along expert (E) dim.
+ dispatched_expert_inputs = split(dispatched_expert_inputs, 0, D)

h = einsum("EGCM,EMH->EGCH", dispatched_expert_inputs, wi)
...
...
Einsum Sharding Example

Matmul/Einsum: \( AB, BC \rightarrow AC \)

Global view

Parallel, B-partitioned matmul

All-reduce

Partition local view

Partial result for Partition 1

Full result
Einsum Sharding Example

Einsum: GSEC, GSM -> EGCM

Parallel, partitioned einsums

Reshard (all-to-all)
Quality vs. Cost

- Quality gain (ΔBLEU)
- Training wall time
- Computation cost
  - TPU v3 core years

- 37.5B weights
  - 128 TPU v3 cores
  - 6 years
- 150B weights
  - 512 TPU v3 cores
  - 15 years
- 600B weights
  - 2048 TPU v3 cores
  - 22 years

- Wall time (days)
- Quality gain (ΔBLEU)
HBM Profile

- Activations
- Weights

Memory usage in GB:

- MoE(128E, 12L)
- MoE(512E, 12L)
- MoE(2048E, 12L)
- MoE(2048E, 24L)
- MoE(2048E, 36L)
- MoE(2048E, 60L)
Execution Time Breakdown

MoE(128E, 36L)
MoE(512E, 36L)
MoE(2048E, 36L)

MoE fllayer  MoE dispatch and combine  Gate cumsum  Gate Einsums  Transformer fllayer
Transformer attention  Transformer projection
Communication Primitives for Sharding

- **AllReduce**: Performs elementwise reduction (e.g., summation) over the inputs from all participants.
- **AllGather**: Concatenates tensors from all participants following a specified order.
- **AllToAll**: Each participant splits its input along a dimension, then sends each piece to all other participants. On receiving data pieces from others, each participant concatenates the pieces.
- **Collective-Permute**: Given a list of source-destination pairs, the input data of a source device is sent to the corresponding destination device.
Conclusion

● Giant neural networks are awesome.

● Mixture-of-expert makes giant nets cost effective.

● Simple API makes building such networks feasible.
Thank you

- Paper
- Code

Fruit fly, Honey bee, Mouse, Cat, Macaque, Human