

Table of Contents

Preface

Section 1 – Data Parallelism

1

Splitting Input Data

Single-node training is too slow	4	Stochastic gradient descent	13
The mismatch between data loading bandwidth and model training bandwidth	5	Model synchronization	14
Single-node training time on popular datasets	6	Hyperparameter tuning	15
Accelerating the training process with data parallelism	8	Global batch size	16
		Learning rate adjustment	16
		Model synchronization schemes	17
Data parallelism – the high-level bits	9	Summary	18

2

Parameter Server and All-Reduce

Technical requirements	20	Defining the parameter server	27
Parameter server architecture	21	Defining the worker	28
Communication bottleneck in the parameter server architecture	22	Passing data between the parameter server and worker	30
Sharding the model among parameter servers	24	Issues with the parameter server	32
Implementing the parameter server	26	The parameter server architecture introduces a high coding complexity for practitioners	
Defining model layers	26		

All-Reduce architecture	34	Broadcast	40
Reduce	34	Gather	41
All-Reduce	36	All-Gather	42
Ring All-Reduce	37	Summary	43
Collective communication	40		

3

Building a Data Parallel Training and Serving Pipeline

Technical requirements	46	Single-machine multi-GPU	52
The data parallel training pipeline in a nutshell	46	Multi-machine multi-GPU	56
Input pre-processing	48	Checkpointing and fault tolerance	64
Input data partition	49	Model checkpointing	64
Data loading	50	Load model checkpoints	65
Training	50	Model evaluation and hyperparameter tuning	67
Model synchronization	51	Model serving in data parallelism	71
Model update	52	Summary	73
Single-machine multi-GPUs and multi-machine multi-GPUs	52		

4

Bottlenecks and Solutions

Communication bottlenecks in data parallel training	76	Tree All-Reduce	85
Analyzing the communication workloads	76	Hybrid data transfer over PCIe and NVLink	91
Parameter server architecture	77	On-device memory bottlenecks	93
The All-Reduce architecture	80	Recomputation and quantization	94
The inefficiency of state-of-the-art communication schemes	83	Recomputation	95
Leveraging idle links and host resources	85	Quantization	98
		Summary	99

Section 2 – Model Parallelism

5

Splitting the Model

Technical requirements	104	BERT	119
Single-node training error – out of memory	105	GPT	121
Fine-tuning BERT on a single GPU	105	Pre-training and fine-tuning	122
Trying to pack a giant model inside one state-of-the-art GPU	107	State-of-the-art hardware	123
ELMo, BERT, and GPT	110	P100, V100, and DGX-1	123
Basic concepts	110	NVLink	124
RNN	114	A100 and DGX-2	125
ELMo	117	NVSwitch	125
		Summary	125

6

Pipeline Input and Layer Split

Vanilla model parallelism is inefficient	128	Advantages of pipeline parallelism	141
Forward propagation	130	Disadvantages of pipeline parallelism	142
Backward propagation	131	Layer split	142
GPU idle time between forward and backward propagation	132	Notes on intra-layer model parallelism	145
Pipeline input	137	Summary	145
Pros and cons of pipeline parallelism	141		

7

Implementing Model Parallel Training and Serving Workflows

Technical requirements	148	Fine-tuning transformers	162
Wrapping up the whole model parallelism pipeline	149	Hyperparameter tuning in model parallelism	163
A model parallel training overview	149	Balancing the workload among GPUs	163
Implementing a model parallel training pipeline	150	Enabling/disabling pipeline parallelism	164
Specifying communication protocol among GPUs	153	NLP model serving	164
Model parallel serving	158	Summary	165

8

Achieving Higher Throughput and Lower Latency

Technical requirements	169	Exploring memory and storage resources	177
Freezing layers	169	Understanding model decomposition and distillation	180
Freezing layers during forward propagation	171	Model decomposition	180
Reducing computation cost during forward propagation	173	Model distillation	183
Freezing layers during backward propagation	174	Reducing bits in hardware	184
		Summary	184

Section 3 – Advanced Parallelism Paradigms

9

A Hybrid of Data and Model Parallelism

Technical requirements	189	Cross-machine for data parallelism	200
Case study of Megatron-LM	189	Implementation of Megatron-LM	201
Layer split for model parallelism	189	Case study of Mesh-TensorFlow	203
Row-wise trial-and-error approach	192		
Column-wise trial-and-error approach	196		

Implementation of Mesh-TensorFlow	204	Pros and cons of Megatron-LM and Mesh-TensorFlow	204
		Summary	205

10

Federated Learning and Edge Devices

Technical requirements	209	Communicating gradients for collaborative learning	212
Sharing knowledge without sharing data	209	Case study: TensorFlow Federated	217
Recapping the traditional data parallel model training paradigm	210	Running edge devices with TinyML	219
No input sharing among workers	211	Case study: TensorFlow Lite	219
		Summary	220

11

Elastic Model Training and Serving

Technical requirements	223	Traditional model-parallel model training paradigm	231
Introducing adaptive model training	223	Adaptive model training in model parallelism	232
Traditional data parallel training	224	Implementing adaptive model training in the cloud	235
Adaptive model training in data parallelism	226	Elasticity in model inference	236
Adaptive model training (AllReduce-based)	226	Serverless	238
Adaptive model training (parameter server-based)	229	Summary	238

12

Advanced Techniques for Further Speed-Ups

Technical requirements	241	Job migration and multiplexing	249
Debugging and performance analytics	241	Job migration	250
General concepts in the profiling results	243	Job multiplexing	251
Communication results analysis	245	Model training in a heterogeneous environment	251
Computation results analysis	246	Summary	252

Index

Other Books You May Enjoy
