High Performance Machine Learning (HPML)

Instructors

- Dr. Parijat Dube, Adjunct Associate Professor of Computer Science and Senior Research Scientist, IBM T.J. Watson Research Center, NY
- Dr. Kaoutar El Maghraoui, Adjunct Professor of Computer Science, and Principal Research Scientist, IBM T.J. Watson Research Center, NY

Course Description

During the past decades, the field of High-Performance Computing (HPC) has been about building supercomputers to solve some of the biggest challenges in science. HPC is where cutting edge technology (GPUs, low latency interconnects, etc.) is applied to the solution of scientific and data-driven problems.

One of the key ingredients to the current success of ML is the ability to perform computations on very large amounts of training data. Today, the application of HPC techniques to ML algorithms is a fundamental driver for the progress of Artificial Intelligence.

In this course, you will learn HPC techniques that are typically applied to supercomputing software, and how they are applied to obtain the maximum performance out of ML algorithms. You will also learn about techniques for building efficient ML systems. The course is based on PyTorch, CUDA programming, MPI.

Objectives

At the end of the course, you will be able to:

- Use HPC techniques to find and solve performance bottlenecks
- Do performance measurements and profiling of ML software
- Evaluate the performance of different ML software stacks and hardware systems
- Develop high performance distributed ML algorithms
- Use fast math libraries, CUDA and C++ to accelerate High-Performance ML algorithms
- Model compression

Prerequisites

- Knowledge of computer architecture and operating system
- C/C++: intermediate programming skills
- Python: intermediate programming skills.
- Understanding of Machine Learning concepts and Neural Networks algorithms:

The course is focused on the system performance rather than the algorithms, and a basic explanation of the algorithms will be part of the course. However, it is strongly recommended to start the course with a good understanding of the following algorithms: logistic regression, feed forward (basic) neural networks, convolutional neural networks, recurrent neural networks.

Course materials

The course does not follow a specific textbook; however, some parts of the following books can be used as a learning support. Pointers to specific literature/web links will be provided in class.

Introduction to High Performance Computing for Scientists and Engineers

Authors: Georg Hager, Gerhard Wellein Editor: CRC Press
ISBN: 9781439811924

Introduction to High Performance Scientific Computing (ONLINE)
Authors: Victor Eijkhout with Edmond Chow, Robert van de Geijn

Computer Architecture 5th Edition - A Quantitative Approach
Authors: John Hennessy, David Patterson Editor: Morgan Kaufmann
ISBN: 9780123838728

Efficient Processing of Deep Neural Networks
Authors: Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer
Morgan & Claypool Publishers ISBN-13: 978-1681738352

Topics covered

ML/DL and PyTorch basics
PyTorch performance
Performance optimization in Pytorch
Parallel performance modeling
Intro to CUDA
Math libraries for ML (cuDNN)
DNNs architectures (CNN, RNN, LSTM, Attention, Transformers) in Pytorch
Intro to MPI
Distributed ML
Distributed PyTorch algorithms, parallel data loading, and ring reduction
Hardware acceleration for ML and AI
Quantization and model compression

Course Information

- **Instructors**: Dr. Parijat Dube and Dr. Kaoutar El Maghraoui

- **Grading:** Homework (40%) + Final Project (20%) + Final Exam (20%) + Quizzes (15%) + Attendance & Participation (5%)

- **Homework**: There will be **five homework assignments** mostly involving programming and experiments involving GPUs. Assignments will be based on C/C++, Python, and PyTorch

- **Course project**
  - Project proposals are due by midterm
  - Final presentations of all projects towards the end of the course.

Weekly Lesson Plan

- *Week-1:* **Introduction to HPC and ML**

Course introduction and organization; HPC and ML technology; ML/DL success drivers; HPC for ML; hardware overview: CPUs, accelerators, high speed networks; software overview: algorithms, math libraries, frameworks

- *Week-2:* **ML performance optimization**
  Factors affecting ML performance; software performance optimization for ML; Performance optimization methodology: measurement, analysis, optimization; Measurement: metrics, benchmarking workloads, time/resources, throughput, time to accuracy (TTA), profiling, tracing; Analysis: Amdahl's law, critical path, bottleneck, data movement locality principle, Roofline model; Optimization in relation to Roofline model
- *Week-3:* **Basic ML in PyTorch**
  Overview of basic ML: supervised vs unsupervised, linear and logistic regression; Deep learning basics: artificial neural network, activation functions, loss functions, gradient descent, forward and backward propagation, SGD, hyperparameters; PyTorch basics: tensors, variables, computation graph, Autograd; Neural network in PyTorch: define and train; PyTorch examples: Autograd, linear regression, neural network
- *Week-4:* **Gradient Descent Optimization Algorithms and PyTorch**
  PyTorch Optimizer: momentum, Nesterov momentum, Adagrad, Adadelta, Adam; PyTorch Multiprocessing: concurrency vs parallelism, forking, spawning, shared memory; PyTorch data loading: Dataloader class, data prefetching, disk I/O performance, PyTorch CUDA
- *Week-5:* **PyTorch Performance**
  Python performance: interpreter inner workings, CPython, memory management, dynamic typing; PyTorch performance: computation graph evaluation approach, Just in Time compilation, profiling, benchmarking; Declarative vs imperative approach for computation graph; JIT compilation optimization; PyTorch profiling: cprofile/profile, profiling a PyTorch neural network, visualization; PyTorch benchmarking using timeit module.
- *Week-6:* **Deep Neural Networks and Pytorch (CNN, RNN, LSTM, Attention, Transformer, VisionTransformer)**
  Overview of standard DNN architectures: CNN, RNN, LSTM and their implementation in PyTorch; Standard CNN architectures: LeNet, AlexNet, VGG, Inception, ResNet; Performance comparison of different CNN architectures; Attention mechanism; Transformers, BERT, and GPT; Bert-base, Bert-large, Roberta; Vision Transformer; Implementation of encoder and decoder in PyTorch.
- *Week-7:* **CUDA Basics**
  Heterogeneous architectures motivations; NVIDIA GPUs and CUDA: compute capability; CUDA compilation and runtime: CUDA runtime, CUDA driver, AoT and JIT compilation; CUDA Programming Model: grid, block, thread, Unified Virtual Memory (UVM); CUDA block and warp scheduling; CUDA streams
- *Week-8:* **CUDA and CNN**
  CUDA memory access: global memory, shared memory, caches; Matrix multiplication: simple, tiled; NVIDIA deep learning SDK; cuDNN: APIs and descriptors; Convolution algorithms in cuDNN; Benchmarking through cuDNN, algorithm performance, workspace; cuBLAS
- *Week-9:* **Message Passing Interface (MPI) and Distributed Deep Learning (DDL)**

DL hardware trends: GPU and communication trends; MPI: interface, communication primitives, API semantic; DDL performance modeling: work depth model, alpha beta model, LogP model

- *Week-10:* **Distributed Deep Learning Algorithms and PyTorch**
  Model, data, hybrid parallelism; Synchronous and asynchronous DDL; Stragglers and stale gradients; Centralized and decentralized DDL; PyTorch DDL: modules for single and multi-node distributed training, available collectives; All-Reduce algorithm; NCCL
- *Week-11:* **Sparsity and Model Pruning/Compression**
  Activation sparsity, weight sparsity**;** Compression; Sparse Dataflow; Low-rank approximation; Knowledge distillation; Distilled architectures in convolutional and recurrent networks
- *Week-12:* **Reduced Precision and Quantization**
  Determining bit-width; Mixed and varying precision; Quantization: post-training quantization, static vs dynamic quantization, quantization aware training, graph mode quantization; hardware aware quantization
- *Week 13*: **Designing Efficient DNNs**
  Improving efficiency in manual network design; Neural architecture search (NAS), hardware-aware NAS; Near memory and In-memory processing; Analog AI