# High-Performance Machine Learning (HPML)

## Instructor

- 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 solve scientific and data-driven problems.
One of the key ingredients to the current success of ML is the ability to perform computations on vast amounts of training data. Today, applying HPC techniques to ML algorithms is a fundamental driver for the progress of Artificial Intelligence.
In this course, you will learn HPC techniques typically applied to supercomputing software and how they are applied to obtain the maximum performance from ML algorithms.

You will also learn about techniques for building efficient ML systems. This is especially becoming more critical in the era of large foundation models such as GPT and LLAMA that require massive amounts of computational power and energy. This course will introduce efficient AI computing techniques for both training and inference. Topics include model compression, pruning, quantization, knowledge distillation, neural architecture search, data/model parallelism, and distributed training.

The course is based on PyTorch and CUDA programming.

## 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 for efficient training.
- Use fast math libraries, CUDA, and C++ to accelerate High-Performance ML algorithms.
- Model compression techniques such as quantization, pruning, and knowledge distillation.
- Essential HPC techniques to handle large foundation models such as Large Language Models (LLMs)

## Prerequisites

- General knowledge of computer architecture and operating systems
- C/C++: intermediate programming skills
- Python: intermediate programming skills.
- Good understanding of Machine Learning concepts and Neural Network algorithms

The course is focused on system performance rather than algorithms, and a high-level review of the algorithms will be part of it. However, it is strongly recommended that you come to the course with a good understanding of the following algorithms: logistic regression, feed-forward (basic) neural networks, convolutional neural networks, recurrent neural networks, and transformer architectures.

## Course materials

The course does not follow a specific textbook; however, some books can be used as learning support. Pointers to 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
PyTorch performance profiling
Performance optimization in PyTorch
Parallel performance modeling
Intro to CUDA
Math libraries for ML (cuDNN)
Intro to MPI
Distributed ML
Distributed PyTorch algorithms, parallel data loading, and ring reduction
Hardware acceleration for ML and AI
Quantization and model compression
Neural Architecture Search
In-Memory Computing

## Course Information

- **Instructor**: Dr. Kaoutar El Maghraoui

- **Grading:** Homework (50%) + Final Project (30%) + Quizzes (15%) + Attendance & Participation (5%)

- **Homework**: There will be **five to six homework assignments,** mostly involving programming and experiments involving GPUs. Assignments will be based on C/C++, Python, and PyTorch.
- **Late Homework Policy:**
  - Quizzes and project submissions must be submitted on time. Zero credits will be given for late submissions.
  - The lowest grade out of your quizzes and homework will be dropped.

  - Throughout the semester, each student has an allocation of **6 'late days.'** These can be used only for homework submissions, allowing flexibility without penalty. However, once your total allowance of 6 late days is depleted, the following late submission penalties will apply:
    - **Original Due Time:** Assignments must be submitted on time for full credit.
    - **Counting Late Days:** Late days are calculated daily, with a new late day commencing at 11:59 pm ET.
    - **Penalty Post-Late Days Allowance:** After exceeding the 6 late days allowance, 20% of the total marks will be deducted per additional late day up to 5 days. Beyond that, the assignment will be given zero credits.

- **Course project**
  - Project proposals are due by the 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 concerning Roofline model.
- ***Week-3:*** **Gradient Descent Optimization Algorithms in PyTorch**
  PyTorch basics: tensors, variables, computation graph, Autograd;
  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-4:*** **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.
- ***Week-5:*** **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

- ***Week-6: CUDA Advanced Topics***
  Unified Virtual Memory (UVM); CUDA block and warp scheduling; CUDA streams. CUDA memory access: global memory, shared memory, caches; Matrix multiplication: simple, tiled; NVIDIA deep learning SDK; cuDNN: APIs and descriptors.
- ***Week-7:*** **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; Efficient transformers.
- ***Week-8:*** **Sparsity, Model Pruning/Compression**
  Activation sparsity, weight sparsity**,** Compression, Sparse Dataflow; Low-rank approximation, Knowledge distillation; Distilled architectures in convolutional and recurrent networks
- ***Week-9:*** **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-10:*** **Knowledge Distillation**
  Knowledge distillation, Distilled architectures in convolutional and recurrent networks, Knowledge distillation in vision transformers
- ***Week 11*****: Efficient Transformers and LLMs**
  Transformer basics, Encoder/Decoder architecture, KV Cache optimizations, Efficient inference algorithms for LLMs, FlashAttention, Switch Transformer, Efficient fine-tuning for LLMs: LoRA, Adapter and Prompt tuning, Attention Sparsity, Mixture of Experts
- ***Week 12*****: Efficient Vision Architectures and Implementations**
  Efficient CNNs, Vision Transformer
- ***Week 13*****: Designing Efficient DNNs with Neural Architecture Search**
  Improving efficiency in manual network design, Neural architecture search (NAS), and hardware-aware NAS.
- ***Week 14*****: In-Memory Computing**
  Analog AI, near-memory or In-memory computing, Neuromorphic computing
- ***Week 15-16*****: Final Project Presentations**