Parallel and Distributed Computing

Course Overview

Prerequisites: Python Programming, Data Structures in Python, Pandas, NumPy, Machine Learning Fundamentals, Operating Systems, Architecture/COAL, and Computer Networks
Primary Language: Python (with minimal exposure to configuration files)

Course Description

This course introduces parallel and distributed computing concepts exclusively through Python-based tools and frameworks, with special emphasis on machine learning applications. Students will learn to scale data science and machine learning workflows using Python libraries and frameworks, focusing on both parallel and distributed machine learning techniques.

Module 1: Python Parallel Computing Fundamentals (Lectures 1-7)

Lecture 1: Introduction to Parallel Computing in Data Science

- Why parallel computing matters for data science and ML
- Types of parallelism: data, task, and model parallelism
- Python's role in parallel and distributed computing
- Performance bottlenecks in single-threaded data processing
- Overview of Python tools ecosystem for parallelism

Lecture 2: Python's Global Interpreter Lock (GIL) and Workarounds

- Understanding Python's GIL limitations
- CPU-bound vs I/O-bound tasks in data science
- When to use threading vs multiprocessing
- Practical examples with pandas and NumPy operations
- Profiling Python code for performance bottlenecks

Lecture 3: Multiprocessing in Python

- multiprocessing module deep dive
- Process pools for parallel data processing
- Sharing data between processes (queues, pipes, shared memory)
- Parallel pandas operations using multiprocessing
- Error handling and debugging in parallel processes

Lecture 4: Concurrent Programming with Python
- concurrent.futures module for high-level parallelism
- ThreadPoolExecutor vs ProcessPoolExecutor
- Asynchronous programming with asyncio basics
- Parallel web scraping and API calls
- Best practices for concurrent data collection

Lecture 5: NumPy and SciPy Parallel Operations
- Understanding NumPy's built-in parallelism
- BLAS and LAPACK backend optimization
- Parallel linear algebra operations
- SciPy parallel computing capabilities
- Memory management for large arrays

Lecture 6: Joblib for Scientific Computing
- Joblib library for parallel computing
- Parallel for-loops with joblib.Parallel
- Memory mapping for large datasets
- Caching expensive computations
- Integration with scikit-learn parallel operations

Lecture 7: Introduction to Dask - Part 1
- Dask ecosystem overview
- Dask arrays for out-of-core NumPy operations
- Lazy evaluation and task graphs
- Basic Dask array operations and performance
- Setting up Dask local clusters

Module 2: Parallel Machine Learning (Lectures 8-15)

Lecture 8: Parallel Machine Learning Concepts
- Data parallelism vs model parallelism in ML
- Embarrassingly parallel algorithms
- Parallel cross-validation and hyperparameter tuning
- Batch processing vs mini-batch processing
- Communication overhead in parallel ML

Lecture 9: Scikit-learn Parallel Computing
- Built-in parallelism in scikit-learn algorithms
- n_jobs parameter across different estimators
- Parallel model selection with GridSearchCV
- Parallel ensemble methods (Random Forest, Extra Trees)
- Custom parallel estimators and transformers

Lecture 10: Dask-ML for Scalable Machine Learning
- Dask-ML ecosystem overview
- Scaling scikit-learn with Dask-ML
- Incremental learning algorithms (SGD, Passive-Aggressive)
- Parallel feature selection and dimensionality reduction
- Model evaluation on large datasets

Lecture 11: Advanced Dask-ML Applications
- Parallel clustering algorithms (K-means, DBSCAN)
- Scalable preprocessing and feature engineering
- Hyperparameter optimization at scale
- Model persistence and serialization
- Integration with existing ML pipelines

Lecture 12: Ray for Machine Learning
- Ray architecture and core concepts
- Ray tasks and actors for ML workflows
- Ray Tune for distributed hyperparameter optimization
- Ray Train for distributed training
- Scaling popular ML libraries with Ray

Lecture 13: Parallel Deep Learning Fundamentals
- Challenges in parallel deep learning
- Data parallelism strategies
- Gradient synchronization methods
- Communication patterns in distributed training
- Memory management for large models

Lecture 14: TensorFlow Distributed Training
- TensorFlow distributed strategies
- MirroredStrategy for multi-GPU training
- MultiWorkerMirroredStrategy for multi-machine training
- Custom training loops with distribution

- Performance optimization techniques

Lecture 15: PyTorch Distributed Training

- PyTorch distributed package (torch.distributed)
- DistributedDataParallel (DDP) implementation
- Data loading strategies for distributed training
- Gradient accumulation and synchronization
- Fault tolerance and checkpointing

Module 3: Distributed Computing and Machine Learning (Lectures 16-23)

Lecture 16: Introduction to Distributed Machine Learning

- Distributed ML system architectures
- Parameter servers vs peer-to-peer approaches
- Consistency models (BSP, ASP, SSP)
- Communication topologies and patterns
- Fault tolerance in distributed ML systems

Lecture 17: Apache Spark and PySpark Fundamentals

- Spark architecture for distributed ML
- Installing and configuring PySpark locally
- SparkContext and SparkSession
- Resilient Distributed Datasets (RDDs) in Python
- Basic RDD operations for data processing

Lecture 18: PySpark DataFrames for ML Data Processing

- Spark DataFrame API in Python
- Creating DataFrames from various sources
- Data preprocessing and feature engineering
- Handling missing data and outliers
- Data partitioning strategies for ML

Lecture 19: PySpark MLlib - Distributed Machine Learning

- MLlib architecture and algorithms
- Distributed feature engineering and selection
- Classification algorithms (Logistic Regression, Random Forest, SVM)
- Regression algorithms (Linear Regression, Decision Trees)
- Model evaluation and cross-validation

Lecture 20: Advanced PySpark MLlib
- Clustering algorithms (K-means, Gaussian Mixture)
- Collaborative filtering and recommendation systems
- Dimensionality reduction (PCA, SVD)
- ML pipelines and workflow automation
- Model persistence and deployment

Lecture 21: Distributed Deep Learning with PySpark
- Deep learning on Spark overview
- Horovod integration with Spark
- TensorFlow and PyTorch on Spark
- Distributed hyperparameter tuning
- Large-scale model training strategies

Lecture 22: Federated Learning Concepts
- Introduction to federated learning
- Privacy-preserving machine learning
- FedAvg algorithm implementation
- Challenges in federated learning
- Python frameworks for federated learning (PySyft basics)

Lecture 23: Ensemble Methods in Distributed Settings
- Parallel ensemble training strategies
- Bagging and boosting in distributed environments
- Voting classifiers across multiple nodes
- Stacking and blending techniques
- Distributed model averaging and aggregation

Module 4: Advanced Topics and Applications (Lectures 24-30)

Lecture 24: GPU-Accelerated Machine Learning
- Introduction to GPU computing for ML
- CuPy for GPU-accelerated NumPy operations
- RAPIDS cuML for GPU machine learning
- CuDF for fast data preprocessing
- GPU memory management and optimization

Lecture 25: Parallel Natural Language Processing
- Distributed text preprocessing pipelines
- Parallel feature extraction (TF-IDF, embeddings)

- Distributed topic modeling with Gensim
- Scaling transformer models
- Parallel sentiment analysis and classification

Lecture 26: Parallel Computer Vision and Image Processing
- OpenCV with multiprocessing for image processing
- Distributed feature extraction from images
- Parallel image classification workflows
- Batch processing of image datasets
- Video processing and analysis at scale

Lecture 27: Time Series Analysis and Forecasting at Scale
- Distributed time series preprocessing
- Parallel feature engineering for time series
- Scaling time series forecasting models
- Prophet for parallel forecasting
- Distributed anomaly detection in time series

Lecture 28: Parallel Optimization Algorithms
- Distributed gradient descent variants
- Parallel genetic algorithms
- Swarm intelligence algorithms in parallel
- Multi-objective optimization at scale
- Custom optimization with Ray and Dask

Lecture 29: Performance Monitoring and Model Serving
- Profiling distributed ML applications
- Memory and CPU optimization techniques
- Model serving with parallel inference
- Load balancing for ML services
- Monitoring model performance in production

Lecture 30: Integration and Future Trends
- Combining different parallel/distributed frameworks
- MLOps for parallel and distributed systems
- AutoML in distributed environments
- Quantum machine learning concepts
- Course review and final presentations

Assessment Methods

- Programming Assignments (25-30%): hands-on Python projects focusing on parallel/distributed ML
- Midterm Examination (30-35%): Theoretical concepts and Python code analysis
- Final Examination (40%): Comprehensive parallel/distributed ML application
- Lab Exercises and Participation (10%): Weekly hands-on exercises

Learning Outcomes

Upon completion of this course, students will be able to:

1. Implement parallel machine learning algorithms using Python libraries
2. Design and deploy distributed machine learning systems
3. Scale deep learning training across multiple GPUs and machines
4. Optimize performance of ML workflows using parallel computing techniques
5. Compare and select appropriate parallel/distributed frameworks for ML tasks
6. Troubleshoot and debug parallel ML applications
7. Understand theoretical foundations of distributed machine learning

Required Python Libraries and Tools

Core Libraries:
- multiprocessing, concurrent.futures, joblib
- NumPy, pandas, scikit-learn
- Dask, Dask-ML, Ray

Machine Learning Libraries:
- TensorFlow, PyTorch (with distributed modules)
- PySpark, PySpark MLlib
- XGBoost, LightGBM (with parallel support)

Specialized Libraries:
- CuPy, RAPIDS cuML (for GPU computing)
- Gensim (for NLP)
- OpenCV (for computer vision)
- NetworkX (for graph ML)

Development Environment:
- Jupyter Lab/Notebook
- Python 3.8+
- Docker (for containerization)
- Git (for version control)

Recommended Books

Books:
1. "High Performance Python" by Micha Gorelick and Ian Ozsvald
2. "Learning Spark, 2nd Edition" by Jules S. Damji et al.
3. "Distributed Machine Learning Patterns" by Yuan Tang
4. "Scaling Python with Dask" by Holden Karau
5. "Deep Learning with Python" by François Chollet

Research Papers:
- "Distributed Machine Learning: Trends, Challenges, and Opportunities"
- "Federated Learning: Challenges, Methods, and Future Directions"
- "Large Scale Distributed Deep Networks" (Dean et al.)

Online Resources:
- Dask-ML Documentation and Examples
- Ray MLlib Documentation
- PySpark MLlib Programming Guide
- TensorFlow/PyTorch Distributed Training Tutorials

Hardware Requirements

Minimum Setup:
- Multi-core CPU (4+ cores recommended)
- 8GB+ RAM
- Local cluster setup with Docker

Recommended Setup:
- Multi-core CPU (8+ cores)
- 16GB+ RAM
- GPU support (NVIDIA with CUDA)/ CoLAB
- Multiple machines for true distributed testing

Hadoop (Check Hadoop Library)
Low level Quantization