Course Outline

Parallelizing Computation with Dask Delayed and Futures

Course Summary

Description

This class module focuses on using the Dask scheduler to empower custom parallel computation. Dask Delayed and Futures represent lightweight mechanisms for building and running custom task graphs, while staying within traditional Python coding patterns. This combination — regular Python code with a powerful distributed scheduler — enables all kinds of industry or discipline-specific workloads to be parallelized for fast, large-scale computation.

Objectives

At the end of this course, students will understand:

- How to parallelize algorithms with minimal changes to your code
- Patterns for scheduling compute against data that may not yet be available
- Maximizing parallelism and minimizing bottlenecks

Topics

- Introduction
- Building and Running Graphs with Delayed
- Running and Managing Work with Futures
- Review and Q & A

Audience

This course is intended for engineers or data scientists who typically work with large volumes of workloads for computation.

Prerequisites

Students should have experience in Python at a basic to intermediate level.

Duration

One day

Course Outline

Parallelizing Computation with Dask Delayed and Futures

Course Outline

I. Introduction
- A. Local parallel programming with ThreadPoolExecutor.map
- B. Async with submit () and concurrent.futures
- C. Generalizing async code with task graphs
- D. Scaling task graphs: basic scheduler idea

II. Building and Running Graphs with Delayed
- A. Creating and running Delayed objects
- B. Decorator and explicit Delayed
- C. Compute and Persist
- D. Inspecting dependencies and task graphs programmatically and visually
- E. Allowed operations with Delayed
- F. What to do about prohibited operations with Delayed, and why
- G. Aggregating data with Delayed
- H. Best practices for Delayed

III. Running and Managing Work Futures
- A. Review of Future/Promise model
- B. Launching tasks with submit, map
- C. Getting results
- D. Inspecting Future status
- E. Graphs: Composing Futures
- F. Managing Future lifetimes and task scheduling: errors, cancel, wait, fire_and_forget, del
- G. Dynamic graphs: as_completed

IV. Review and Q & A
- A. Comparison: Futures vs. Delayed
- B. Using scatter
- C. Debugging