

A Comparative Analysis of Ray and Dask: Scaling Frameworks for Big Data Analysis and Machine Learning in Python

Professor : Dimitrios Tsoumakos

Varis
Athanasios

Vlachopoulos
Georgios

Nikiforidis
Ioannis



Table of Contents



-
- I. Introduction
 - II. Installation and Setup Process
 - III. Subjects of Analysis
 - IV. Data Generation and Database Loading
 - V. Setting up the Clusters
 - VI. Performance Measuring Code
 - VII. Results
 - VIII. Conclusion and Use Cases
-





I. Introduction



- Virtual Machines
- Python
- Dusk
- Ray





Virtual Machines



- Consistent Environments
- Resource Efficiency
- Cost Savings
- Scalability
- Isolation





Python



- Machine Learning Frameworks
- Scalability with Dask and Ray
- Community Support
- Extensive Libraries
- Data Visualization
- Accessibility



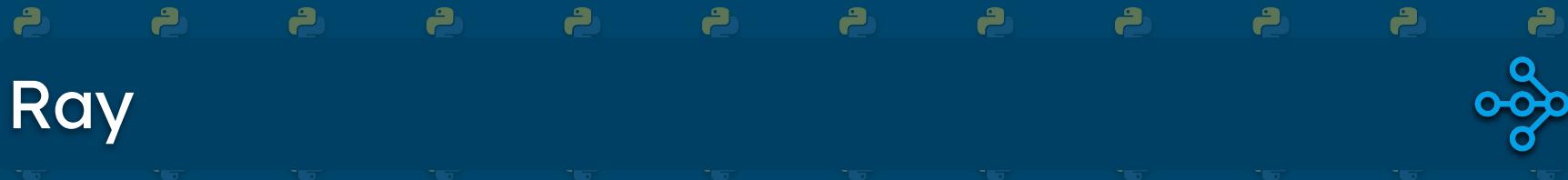


Dask



- Support for Out-of-Core Computing
- Integration with Existing Libraries
- Dynamic Task Scheduling
- Parallel Data Structures
- Scalability





Ray



- Distributed Data Processing
- Dynamic Task Graphs
- Distributed Memory
- Task Parallelism
- Ray Libraries
- Actor Model



II. Installation and Setup Process



- Virtual Machines
- Python
- Ray Framework
- Dask Framework
- Networking





III. Subjects of Analysis



- Classification
- Grid Search
- Clustering
- Principal Component Analysis





IV. Data Generation and Database Loading



- Dataset for Classification and Grid Search



- Data points
 - Features
 - Labels



- Dataset for Clustering and PCA

- Data points
 - Noise





V. Setting up the Clusters



Ray

- The head node runs on the master, that provides a dashboard and an IP address for the workers to connect to.
- Multiple worker processes that connect to the IP address of the head node and get assigned tasks.

Dask

- A scheduler process that provides an IP address for the workers to connect to, and a dashboard to monitor the cluster.
- Multiple worker processes that connect to the IP address of the scheduler and get assigned tasks.





VI. Performance Measuring Code



- **Code**

- Parsing data into dataframes / numpy arrays
- **Implementing Classification**
- Implementing Grid Search
- Implementing Clustering
- **Implementing PCA**

- **Making the clusters**

- **Ray**
- **Dask**





VII. Results



- Hinderancies
- Performance Evaluation
- Efficiency Evaluation
- Classifiers
- Clustering





VIII. Conclusion and Use Cases



- Further Research Required
- Similar Performance
- Varied Efficiency
- Necessity of distributed computing



The End



Thank You!