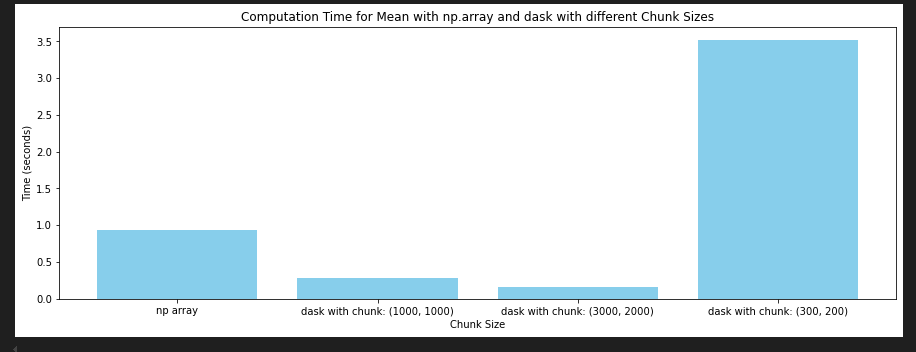
**CSYE7105 13969 Parallel Machine Learning & AI SEC 01 Fall 2024**

Submitted by: Anshul Chaudhary

NEU ID: 002820096

Professor: Handan Liu

**Part 1:**

****

**Analysis:**

* **Larger Chunks Improve Performance:** The fastest execution time was achieved with the largest chunk size (3000, 2000), which minimized the number of tasks Dask needed to manage, reducing task overhead significantly.
* **Medium-Sized Chunks as a Balanced Option:** The (1000, 1000) chunk size provided a balance between performance and memory usage, making it a good compromise for systems with moderate memory availability.
* **Increased Task Overhead with Smaller Chunks:** The smallest chunk size (300, 200) resulted in the slowest execution time, as the increased number of tasks created high overhead for task management, highlighting the downside of having too many small chunks.
* **Memory vs. Speed Trade-Off:** Larger chunks generally led to faster execution but also required more memory per chunk. Choosing an optimal chunk size depends on the system’s available memory and processing power.
* **Optimal Chunk Size Depends on Resource Constraints:** While (3000, 2000) was optimal here, selecting an ideal chunk size should consider memory limitations, data scale, and the system’s ability to handle larger blocks efficiently without running out of memory.

**Part 2:**

A graph with a line

Description automatically generated

**Training Time Decreases with More CPUs:** The plot shows a clear trend where the training time decreases as the number of CPUs increases. As you go from 1 CPU to 8 CPUs, the training time drops significantly. This indicates that the model is highly parallelizable and can efficiently leverage multiple CPUs to speed up the training process. The reduction in time between each step (e.g., from 1 to 2 CPUs, 2 to 4 CPUs, etc.) demonstrates how the task is scaled across different processors.

A screen shot of a graph

Description automatically generated

The results show two distinct execution times:

* Dask XGBoost: 6.864945888519287 seconds
* XGBoost with njobs=4: 20.394912004470825 seconds

Analysis:

* Efficiency: Dask XGBoost demonstrated superior efficiency, completing the task in roughly one-third of the time taken by regular XGBoost.
* Scalability: The results suggest that Dask XGBoost scales better with multiple CPUs, likely due to its distributed computing architecture.
* Resource Utilization: Dask XGBoost appears to make more effective use of the available CPU resources, possibly through better data partitioning and task distribution.

**Part 3:**

Dask Process:

A screenshot of a computer

Description automatically generated

* This graph shows the status of tasks being processed across multiple processors with four distinct bars, which represent four separate processing workers.
* Each bar is filled solidly in blue, indicating that all workers are fully engaged in processing tasks.

Dask Graph:

A screenshot of a computer

Description automatically generated

* Efficient Memory Management: Completed tasks are released from memory, freeing up resources and ensuring that only necessary intermediate results are retained, optimizing memory usage.
* Balanced Parallel Processing: Tasks are evenly distributed across four processors, which helps maximize parallel processing efficiency and reduces idle time.

Dask Task Stream:

A screenshot of a computer

Description automatically generated

* Task Duration and Sequence: The Task Stream plot shows tasks being executed over time, with different colors representing different task types. This allows us to see the order and speed of task completion, helping identify any slower stages in the process.
* Resource Usage and Optimization: The Bytes stored per worker chart and color-coded tasks indicate memory usage and potential bottlenecks. For example, red tasks may use more memory, highlighting areas where optimizing memory allocation could improve performance.