Tree Ensemble Classifiers on heterogeneous platforms: Performance & Scalability Challenges

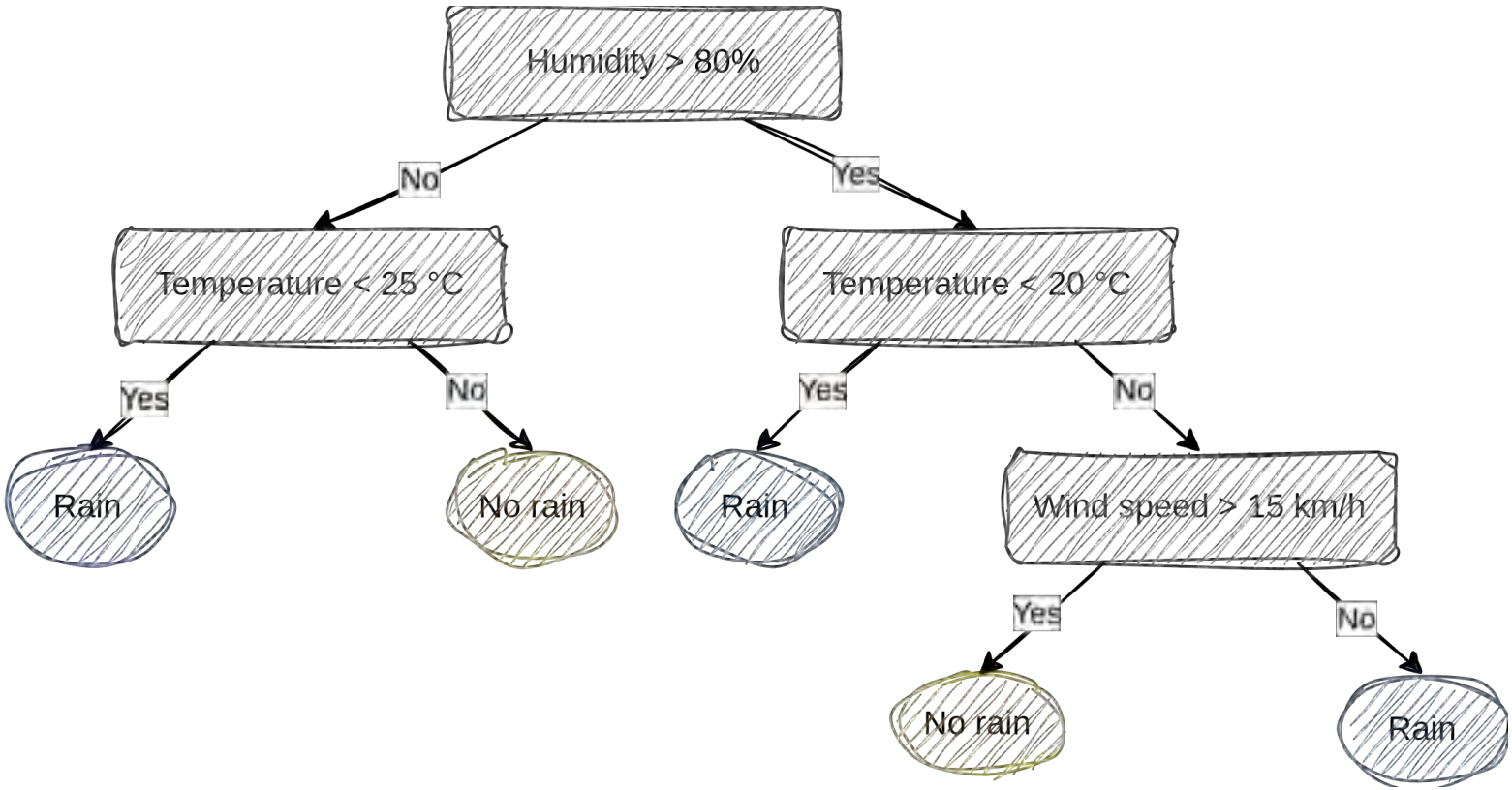
Andrei Stroganov
savthe@gmail.com

Heterogeneous CPUs

Heterogeneous CPUs, particularly those utilizing big.LITTLE architecture combines high-performance cores (big) with energy-efficient cores (little) within a single processor. Initially designed for mobile devices, where power efficiency is crucial, heterogeneous CPUs are now gradually making their way into the server segment.

- Dynamically allocate workloads to the appropriate cores
- Applications can range from lightweight microservices to resource-intensive data processing tasks
- Attractive option for data centers striving to lower operational costs and minimize their environmental impact.

Quick recap. A decision tree



Random forest model is collections of decision trees, where each tree is trained on a random subset of a training dataset.

Gradient boosting model is a sequence of small decision trees, where each tree aims to compensate prediction inaccuracy of previous trees.

Tree ensemble models

Tree ensemble classifiers are a powerful class of machine learning algorithms that combine the predictions of multiple decision trees to improve accuracy and robustness.

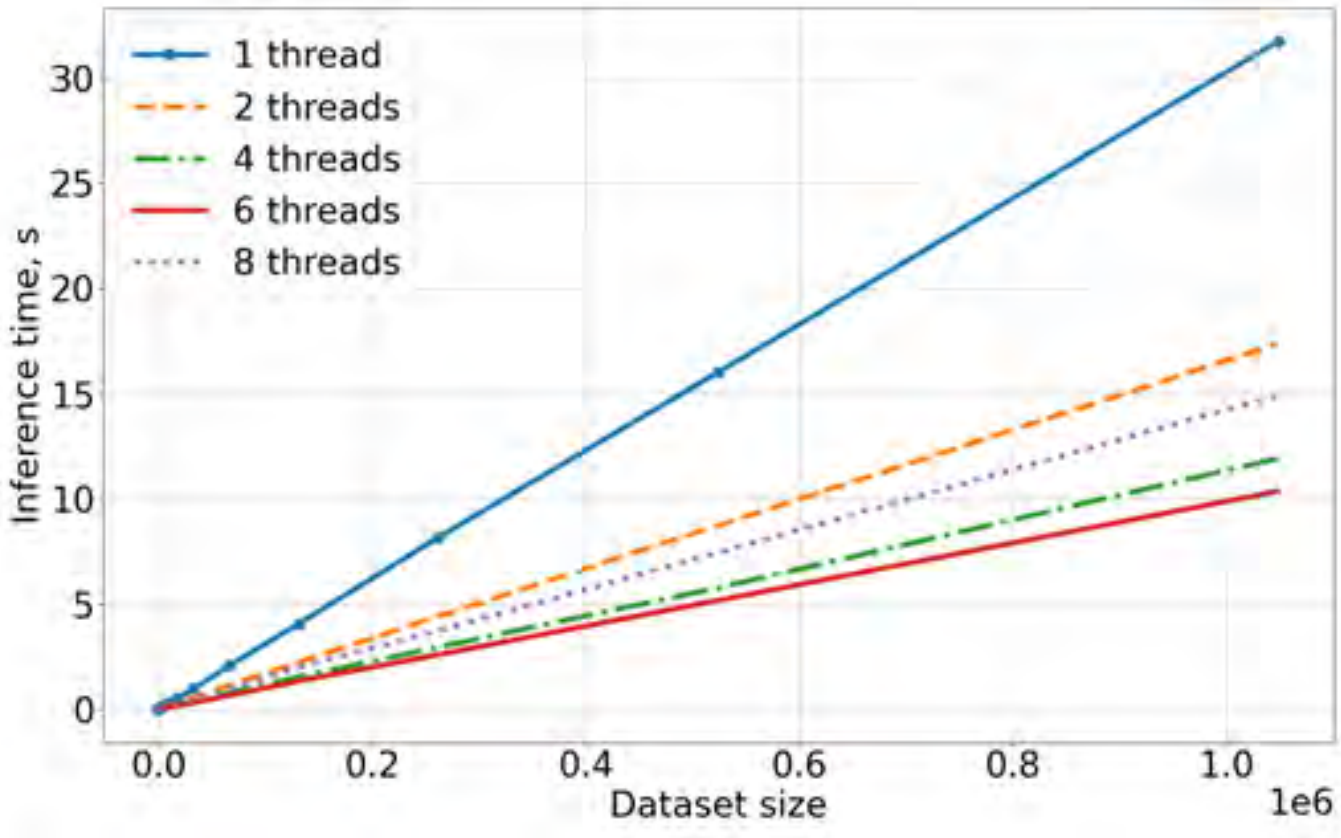
Random forest and Gradient boosting are two of the most popular tree ensemble methods, including implementations like ONNX, XGBoost and LightGBM. These algorithms are widely used across various domains, including finance, healthcare, and marketing, due to their ability to handle large datasets, manage high-dimensional feature spaces, and provide interpretable results.

Due to branching nature of both algorithms, they are usually executed on CPUs, where as we will see, the heterogenity should be considered during implementation or porting.

Experiment settings

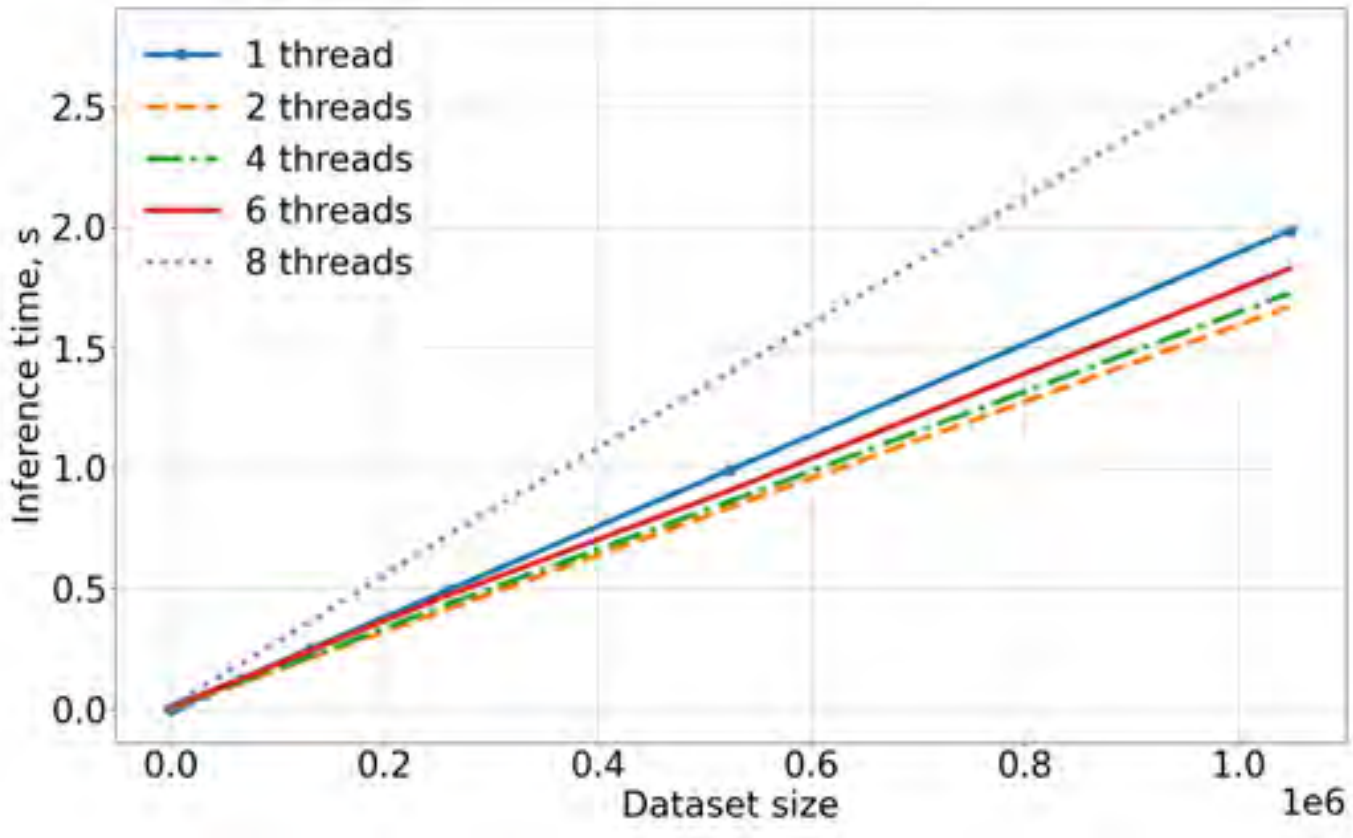
- **Dataset**. Weather forecasting dataset with 21 numerical features.
- **CPU**. Octa-core:
  - 2.84 GHz Cortex-A77 (1 core)
  - 2.42 GHz Cortex-A77 (3 cores)
  - 1.80 GHz Cortex-A55 (4 cores)
- **ONNX**. Runtime version: 1.20.1, graph optimization level: Full
- **Random forest**. 100 trees, max depth: 20.
- **Gradient boosting**. 100 trees, max depth: 3.

Random forest stock performance



Best performance achieved with 6 threads, 3.3x. Full concurrency penalty.

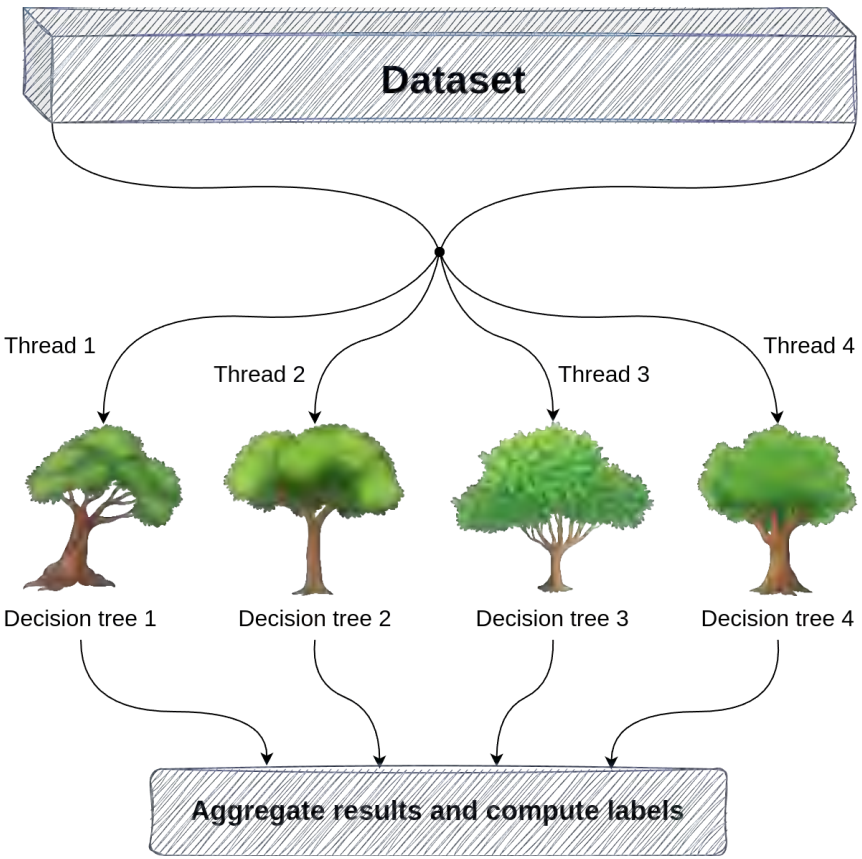
Gradient boosting stock performance



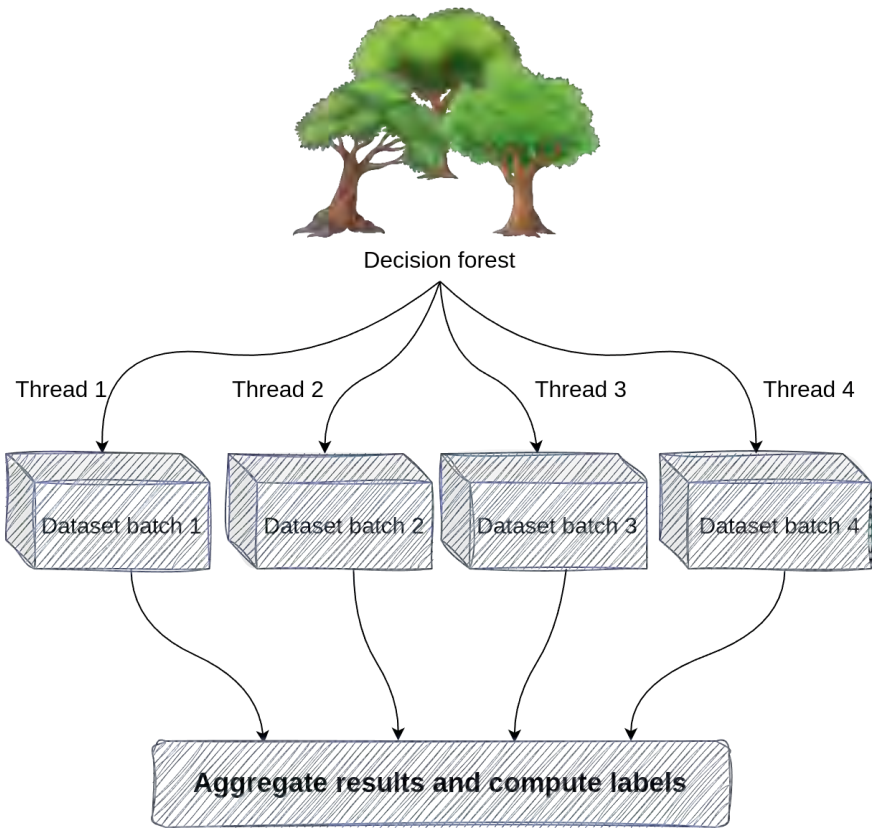
Best performance achieved with 2 threads, 1.2x. Full concurrency penalty.

How to parallelize Random Forest?

Each tree is processed in a dedicated thread



Dataset is split into batches. Each batch is processed by decision forest in dedicated thread



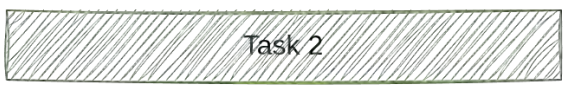
Parallel task processing

Let's give 4 equally demanding tasks to 4 cores with different performance.

Case 1. Cores won't help each other

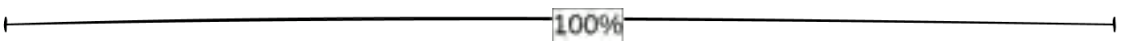
Core A: the fastest, 4 times faster than slowest core

Core B: medium, 2 times faster than slowest core



Core C: the slowest

Core D: the slowest



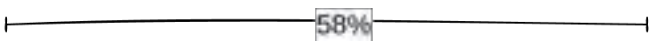
Case 2. Fastest cores take large tasks from slower ones

Core A

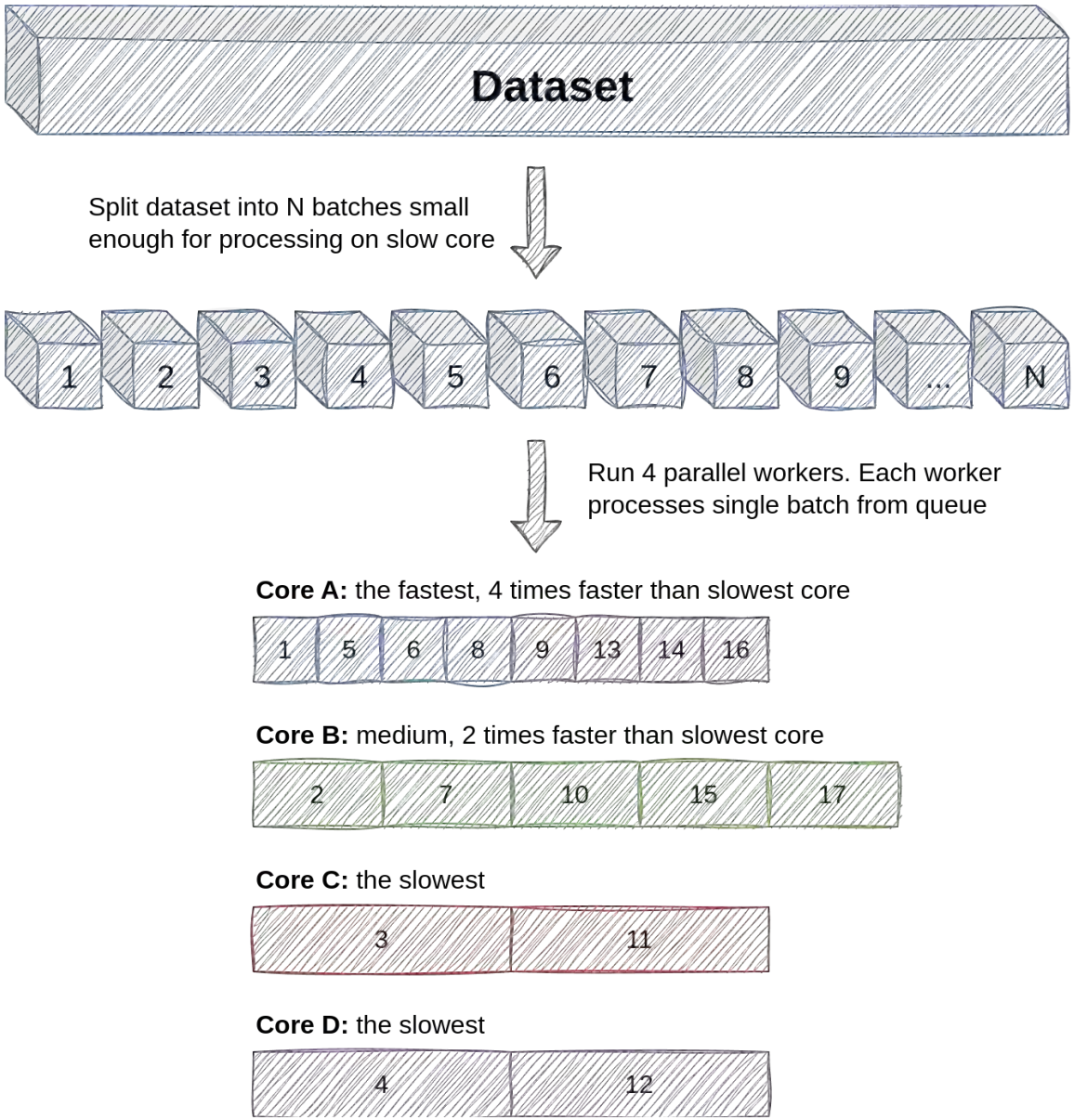
Core B

Core C

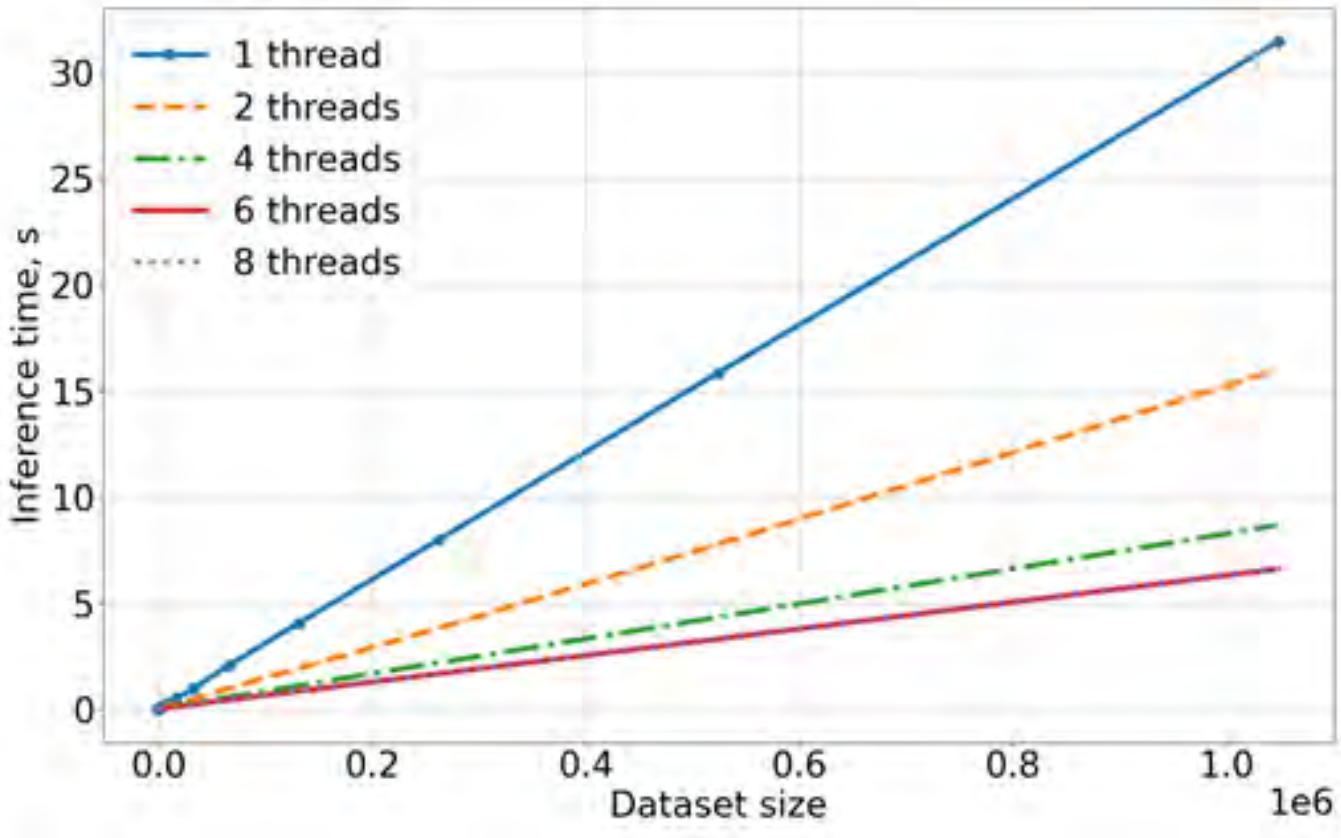
Core D



Worker pool

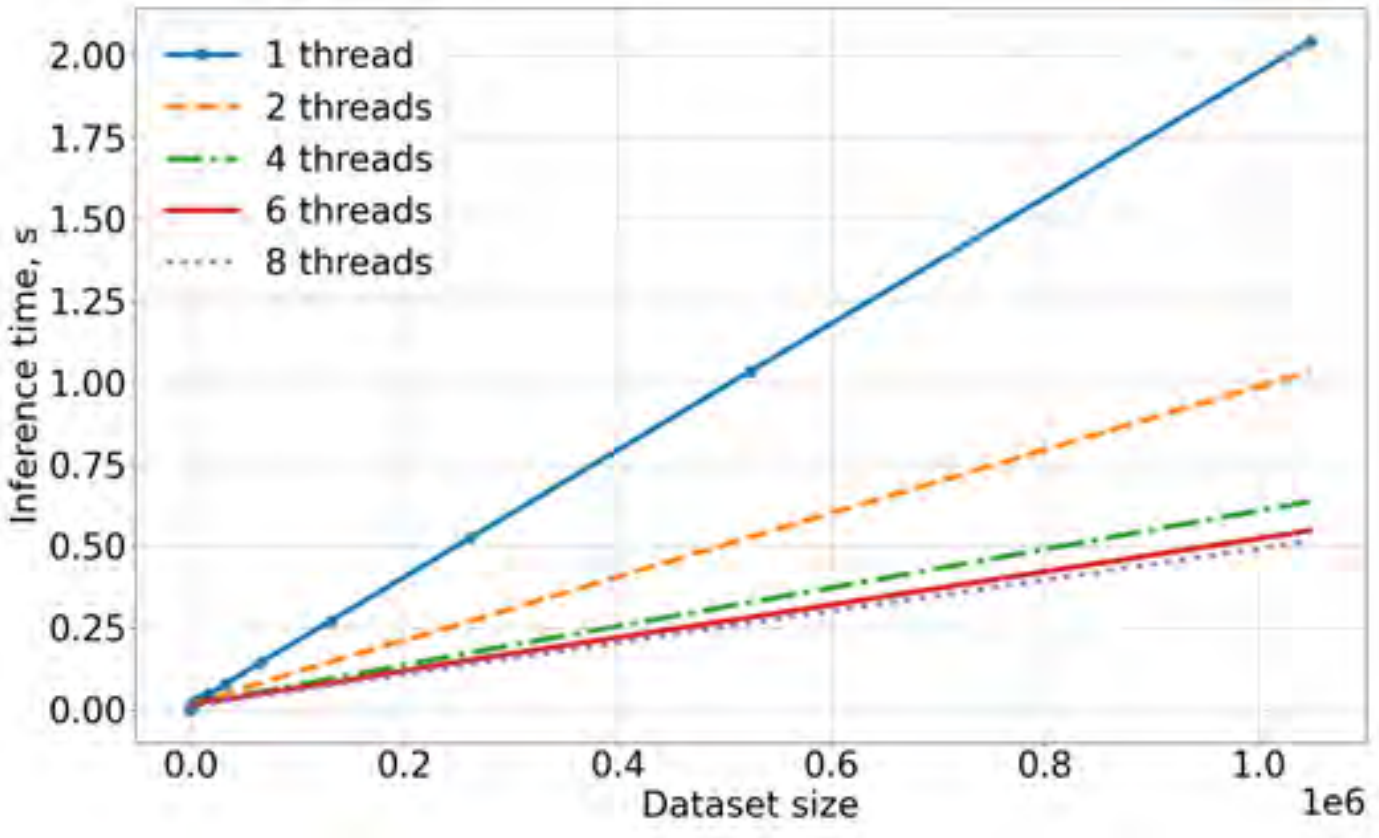

Dataset
Split dataset into N batches small enough for processing on slow core
1
2
3
4
5
6
7
8
9
...
N
Run 4 parallel workers. Each worker processes single batch from queue
Core A: the fastest, 4 times faster than slowest core
1
5
6
8
9
13
14
16
Core B: medium, 2 times faster than slowest core
2
7
10
15
17
Core C: the slowest
3
11
Core D: the slowest
4
12

Worker pool random forest performance



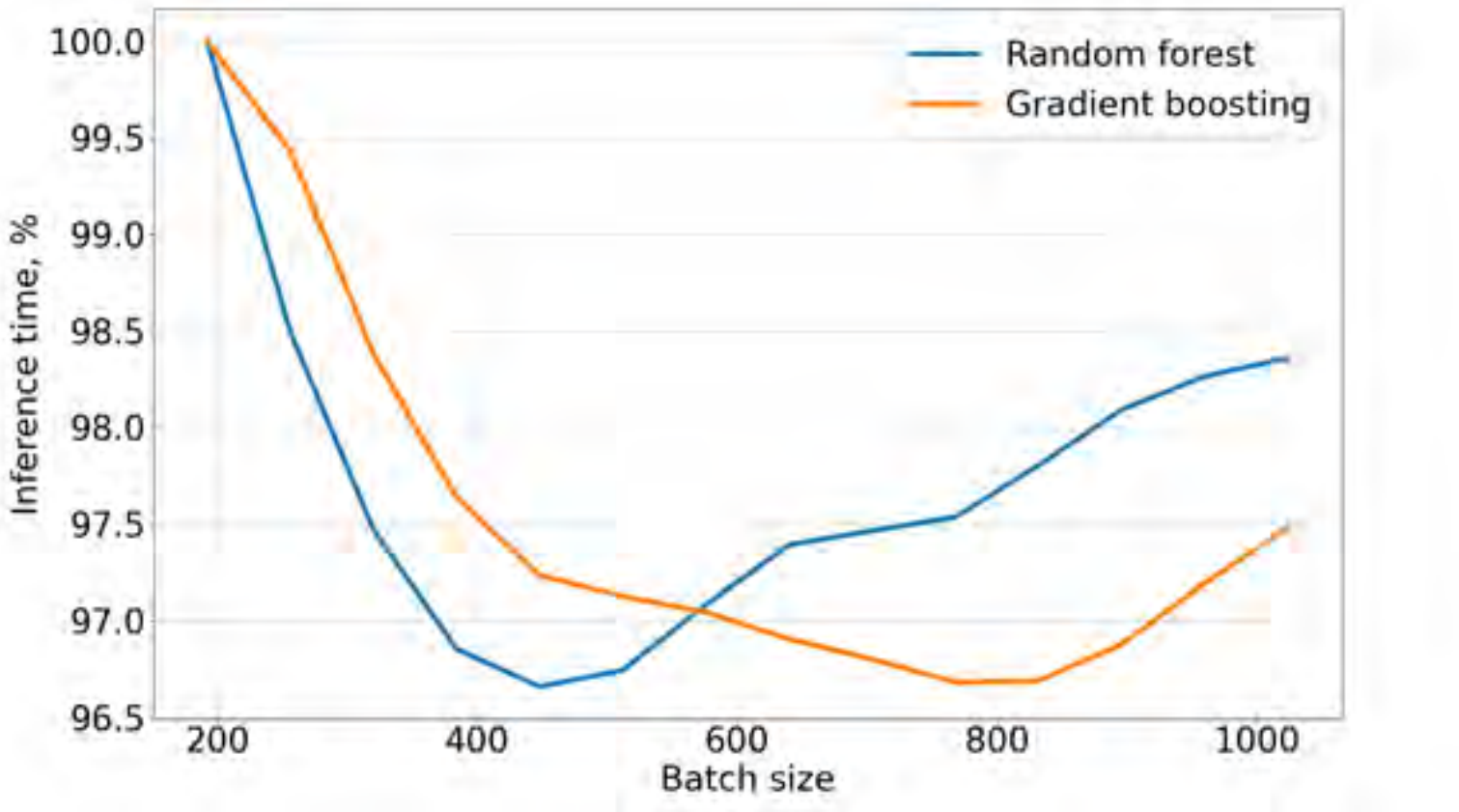
Best performance achieved with 6, 7, 8 threads, 4.8x. No concurrency penalty.

Worker pool gradient boosting performance



Best performance achieved with 8 threads, 4x. No concurrency penalty.

Batch size



Batch size of approximately 400 — 800 samples provides best performance in worker pool approach.

Conclusion

- Porting software to heterogeneous CPUs may require some architectural changes to fully utilize computing potential.
- Demonstrated that stock multithreading architecture may inefficiently utilize heterogeneous CPU and discussed reasons for this.
- Demonstrated how to solve this problem using worker pool pattern.
- Worker pool solution gives 4.8x and 4x performance boost for Random forest and Gradient boosting respectively, while stock performance boost is 3.3x and 1.2x.