CSC4160: Cloud Computing - Final Project (Final Report)
Title: Analyzing public cloud traces using machine learning to improve resource allocation
Team member:
Chua Qin Di, 125400010, 125400010@link.cuhk.edu.cn

Github Repo: https://github.com/chuaqindi/cloud-computing-project
Video Link:
https://drive.google.com/file/d/1RUlf4Cxp_8FqwlVIPA-LYey4AZBJDK9U/view?usp=sharing

1. Introduction

Modern large-scale cloud providers such as Google Cloud, AWS, and Microsoft Azure execute millions of containerized workloads daily on shared compute clusters. Each submitted job requires users to specify CPU and memory requirements before execution. However, these user-provided resource requests are frequently inaccurate and overly conservative. To avoid failures, users tend to over-request resources, resulting in substantial underutilization and inflated operational costs. Conversely, aggressive under-requesting increases the risk of out-of-memory (OOM) failures, job evictions, and service-level agreement (SLA) violations.

Building on the midterm work, this final report presents a complete proactive resource prediction and right-sizing framework using the Google Cluster Trace 2019 dataset. The report describes the finalized data processing pipeline, the construction of a scheduler-compatible feature set, and the optimized LightGBM model used for resource demand prediction. The approach is evaluated through measuring predictive accuracy, analyzing feature importance, and quantifying potential reductions in resource over-provision. Finally, the report discusses how predictive right-sizing can be integrated into existing scheduler policies, identifies current limitations, and outlines directions for future extensions.

2. Dataset and Data Extraction

2.1 Google Cluster Trace 2019

The Google Cluster Trace 2019 dataset is a large-scale, publicly released trace collected from Google's production data centers. It captures how containerized workloads are scheduled, executed, and monitored across thousands of machines in a real cluster environment. At a high level, the trace includes:

- CPU and memory usage sampled at regular intervals for each container instance.
- User-requested CPU and memory specifications.
- Instance-level lifecycle events, including start, finish, failure, and eviction.
- Collection-level metadata such as priority, scheduling class, and workload type.
- Machine identifiers and scheduling-related attributes.

Because the trace reflects real production workloads rather than simulated data, it is well suited for studying workload behavior, inefficiencies in user resource requests, and the potential benefits of prediction-based resource management.

2.2 Use of Google BigQuery

The raw Google Cluster Trace data is stored across multiple Google BigQuery tables and is too large to download and process directly on a local machine. Google BigQuery is a serverless, petabyte-scale SQL analytics engine that enables efficient querying over large datasets without full materialization.

In this project, BigQuery is used to select, join, and aggregate relevant information from multiple tables:

- **instance_usage:** Runtime CPU and memory usage samples.
- **instance_events:** Instance lifecycle events (start, stop, fail, evict).
- **collection_events:** Job-level metadata such as scheduling class, priority, and user.

To make the dataset manageable for local analysis and modeling, a representative sample of 500,000 rows is extracted. The resulting dataset contains both pre-execution metadata and post-execution usage statistics suitable for exploratory data analysis and machine learning. Nested JSON fields are flattened into structured columns.

3. Exploratory Data Analysis

Exploratory analysis conducted during the midterm report revealed pervasive resource underutilization and strong structure in workload behavior. CPU usage across the cluster is extremely skewed: most workloads consume well under 2–3% CPU on average, with only short-lived bursts reaching higher utilization. In contrast, memory usage is comparatively stable over time, yet user-requested memory often exceeds observed peak usage by multiple factors, indicating systematic over-provisioning.

Reliability-related patterns also exhibit clear trends. Failure and eviction rates increase for both very small and very large memory requests. Additionally, workloads belonging to lower scheduling classes experience significantly higher disruption due to preemptive scheduling policies. These observations motivate the use of predictive models to estimate realistic resource requirements and improve scheduling efficiency.

4. Predictive Modeling

The predictive modeling experiments evaluate how effectively machine learning can estimate true resource usage relative to user-provided requests. Multiple target variables were considered, including average CPU usage, average memory usage, and peak memory usage. Models were evaluated under two distinct settings:

- **Idealized Prediction:** Incorporates post-execution and runtime-derived features
  Real-time Scheduling Prediction: Restricted strictly to submission-time metadata available to production schedulers.

This separation allows us to quantify the performance gap between offline models and realistic, scheduler-compatible predictors.

4.1 Findings

The comparison between the two modeling settings highlights a fundamental challenge in cloud resource management. While workload resource usage is highly predictable when runtime information is available, predictive performance degrades when models are limited to submission-time features. This reflects the inherent uncertainty faced by production schedulers, which must operate before any execution data is observed.

Nevertheless, the results indicate that a non-trivial amount of predictive signal remains available at job submission. Historical workload behavior, scheduling metadata, and job context collectively enable meaningful approximation of future resource demand. These findings motivate the design of history-aware, submission-time prediction frameworks that balance predictive accuracy with strict scheduler constraints.

4.2 Final Model Performance (Pre-execution features only)

To evaluate the feasibility of resource prediction under realistic scheduler constraints, the final experiments restricted the model strictly to pre-execution features available at job submission time. This setting reflects the operational reality of production schedulers, which cannot rely on runtime measurements when making initial admission decisions.

Under this strict constraint, the LightGBM model achieved an **R^2 score of 0.6049** and an **RMSE of 0.00505** when predicting maximum memory usage. Although this performance is lower than models incorporating post-execution features, it demonstrates that approximately 60% of the variance in memory usage is predictable using only submission-time metadata and historical context.

Feature Importance Analysis Gain-based feature importance analysis provided insight into the sources of this predictive signal:

- **Assigned Memory (Score: 1256):** This remained the dominant feature, indicating that user requests, while conservative still encode meaningful information about workload scale.
- **Priority (419) and Scheduling Class (387):** These metadata features ranked highly, highlighting that the "importance" a user assigns to a job correlates strongly with its resource consumption stability.
- **Lagged Machine Load (350):** Recent activity on the hosting machine provided useful contextual cues regarding expected workload behavior.

Together, these results demonstrate that scheduler-compatible, pre-job prediction can support meaningful memory right-sizing, even in the absence of runtime observability.

4.3 Safety Margins and Right-Sizing

Predictive resource allocation inevitably involves uncertainty. In particular, under-prediction of memory demand may result in out-of-memory (OOM) failures. In production cloud systems, such failures are typically not catastrophic; schedulers address them through job retries with increased allocations or by migrating failed jobs.

To account for this reality, predictive right-sizing must balance resource efficiency against failure risk. We incorporated safety margins by scaling predicted memory usage by a configurable multiplicative factor (1.20x) before allocation.

Empirical results show that even modest safety margins allow the ML-based policy to significantly reduce average memory over-provisioning while keeping the rate of unsafe allocations within tolerable bounds. The safety rate (percentage of jobs not requiring retry) shifted from **72.70%** (baseline) to **49.79%** (ML-assisted), a trade-off accepted to achieve large efficiency gains. These findings align with the design of real-world schedulers, which accept occasional failures as part of a broader optimization strategy.

5. Key Findings

The experimental results lead to several key insights regarding predictive resource allocation in large-scale cloud environments:

- **User memory requests are highly conservative.** Analysis of the test set revealed pervasive over-estimation of resource needs. The median workload requested **1.63x** its actual peak memory usage, while the top 25% of most wasteful jobs (75th percentile) requested more than **2.62x** their actual usage. This confirms that significant reclaimable capacity exists within the cluster.
- **Runtime information dramatically increases predictability but is unavailable.** Experiments comparing "Idealized" vs. "Strict" settings confirmed that while runtime metrics (like current CPU usage) offer high predictive power, they are inaccessible at the scheduling stage.
- **Meaningful prediction remains possible using only pre-job features.** Despite strict information limitations, the submission-time model achieved an R^2 of ~0.60. Feature importance analysis identified **Assigned Memory** and **Priority** as dominant predictors, proving that a job's initial request and its scheduling metadata provide sufficient signal for effective approximation.
- **Prediction enables policy-level gains without perfect accuracy.** Even with imperfect predictions, the ML-based right-sizing policy demonstrated substantial efficiency gains. When applied to the test set, the proposed model reduced median over-provisioning by

40.1% (from 1.63x down to 0.98x) and reduced the waste of the worst offenders (75th percentile) by **27.5%**.

6. Conclusion

Overall, this study highlights both the promise and the limitations of predictive resource management, emphasizing that practical gains arise not from perfect forecasts, but from the careful integration of machine learning with systems-aware policy design.

The results demonstrate that while perfect prediction is unattainable without runtime information, a significant portion of memory demand remains predictable at job submission. When combined with safety margins, these predictions enabled substantial reductions in memory over-provisioning, reducing median waste by over **40%** while maintaining reasonable failure risk. Importantly, this approach aligns with the operational realities of production schedulers and requires no fundamental changes to existing execution models.

Future work could extend this framework in several directions. First, incorporating richer job lineage information or user-level behavioral patterns could further improve pre-job prediction accuracy. Second, integrating lightweight reactive mechanisms based on runtime monitoring could refine allocations during execution, combining proactive prediction with adaptive feedback. Finally, evaluating the approach within a full cluster simulation would allow direct measurement of system-wide performance impacts such as throughput, fairness, and tail latency.