CP395 Weekly Report 1

Sufiya Rahemtulla – 169018559

Sunday January 18th, 2026

1. Week 1 Objectives and Completion Status

- Project Kickoff Meeting: attended
- GitHub repository created and initialized - Git repository link
- GitHub Project board created with Week 01–02 task - Git project board link
- Dataset Identification (inc description & baseline awareness) – *See Section 2*
- 1-page problem statement drafted – included in report section of Git repository
- (Week 1 Problem Statement Draft)

2. Decisions Made

Research Scope - Autoscaling:
Autoscaling was most interesting to me as it closely aligns with time-series analysis, forecasting, and performance evaluation, and because it is easier to reason about at a high level compared to scheduling, which often requires deeper systems-level knowledge. From what I understand, autoscaling also provides a clear way to evaluate trade-offs between **system performance and resource cost**, which makes it well suited for the problem statement.

Dataset Selection:
I chose cluster trace data over VM-level or serverless traces because cluster trace capture realistic, large-scale workload variability and resource usage patterns that are directly relevant to autoscaling decisions. Google's traces are widely used in academic research, making them a reliable starting point for the scope of this project. I'll be able to download JSON files (most cost effective) of the data which is stored on Google's cloud platform since I likely won't use all 8 clusters for this specific project (as that would be quite large to analyze).

- ***Selected Dataset:*** *Google Cluster-Usage Traces v3 (2019)*
- ***Source:*** *The dataset is the "Google cluster-usage traces v3," released by Google in 2020. It provides trace data from eight Google compute cells managed by the Borg cluster management system.*
- ***Citation:*** *Wilkes, J. et al. (2020). Google cluster-usage traces v3. Google. Available at: https://github.com/google/cluster-data.*
- ***Time Span:*** *The traces cover the full month of May 2019 (May 1st to May 31st).*
- ***Granularity:*** *Resource usage is reported in non-overlapping windows of typically 5 minutes (300 seconds). Within these windows, usage is sampled at approximately 1 Hz (once per second) to calculate averages and peaks. Timestamps are provided in microseconds*
- ***Available Signals:*** *The dataset provides rich signal data relevant to autoscaling in the InstanceUsage and TaskEvents tables:*
  - ***CPU Usage:*** *Average usage, maximum usage, and distribution percentiles (0th to 100th) measured in Normalized Compute Units (NCUs).*
  - ***Memory Usage:*** *Average memory consumption and maximum memory usage (normalized bytes), as well as the assigned memory limit.*
  - ***Lifecycle Events:*** *Timestamps for when tasks are SUBMIT, SCHEDULE, EVICT, or FINISH, allowing for the reconstruction of arrival rates and job durations.*
  - ***Constraints:*** *Machine attributes and placement constraints are also available.*

Suitability:
This dataset is directly aligned with the problem context because it captures the "bursty" and "rapidly changing" production workloads that cause simple threshold-based scalers to fail. Furthermore, the availability of both actual usage data and assigned resource limits allows for a realistic simulation of the research question's core trade-off: calculating "resource utilization" (efficiency) versus determining when usage would have exceeded capacity (simulated "SLA violations").

Baseline awareness: To represent the "reactive threshold-based scaling" described in the research question, the baseline will use the trace's average_usage CPU signal to trigger scaling actions. This heuristic will simulate the industry-standard approach by adding resources only when the 5-minute average usage crosses a high threshold (e.g., 80%) and removing them when it falls below a low threshold, serving as the control to measure the predictive model's improvements in performance and efficiency

3. Early Observations, Assumptions, and Risks

Early Observations:

- The selected Google dataset reports resource usage in 5-minute windows. While this is sufficient for general trending, it effectively smooths out sub-second spikes. This means my autoscaler will be reacting to "sustained" load changes rather than instantaneous bursts
- The Google traces rely on "Normalized Compute Units" (NCUs) rather than raw core counts. This observation confirms that my cost metrics must be relative (e.g., "utilization of available capacity") rather than absolute dollar amounts, as the specific hardware specs are obfuscated

Assumptions:

- **Fixed Provisioning Delay**: (e.g., 60 seconds), In reality, cloud startup times vary, but simulating variable delays would add unnecessary complexity for a 3-month timeline
- **Data Continuity**: The trace documentation notes that missing records can occur during overload or monitoring failures. I assume that short gaps in trace data can be linearly interpolated without significantly biasing the autoscaling evaluation
- **Independent Intervals:** I assume that scaling decisions made in one time interval do not impact the *incoming workload* in the next. The workload traces are "played back" regardless of system performance.

Risks:

- The Google traces are massive and split across multiple files. There is a technical risk that preprocessing the data into a usable format (cleaning, joining tables) may take longer than expected.
- There is a risk that the "Average Resource Utilization" reported in the traces might hide short-term violations that occurred within the 5-minute window. If the provided percentiles (CPU usage distribution) are difficult to parse, my "SLA Violation" metric might be too optimistic

Next Weeks Plan:
Week 2 will focus on literature review as well as identify research gaps.