Project Definition – Server Digital Twin

Authors: Wilson Santiago Bonilla Guevara and Harold Stiven Vargas Henao

Purpose and Problem Statement

Modern computing systems, especially servers, operate under dynamic and unpredictable workloads. Understanding their behavior under varying resource demands is crucial for ensuring performance, reliability, and energy efficiency.
However, conventional monitoring tools (such as Grafana or Prometheus) only provide descriptive information of the system in real time — they do not allow controlled experimentation or predictive modeling.
The purpose of this project is to build a Digital Twin of a computing server that can simulate different workloads (CPU, memory, and disk stress), record real-time system responses, and generate datasets that serve as the foundation for predictive models of server performance.

Background and Gaps

Existing monitoring solutions and observability platforms focus mainly on data visualization and alerting. While they capture real-time data, they lack experimentation control and do not produce reproducible datasets for simulation or learning.
Research on infrastructure twins and cloud performance modeling demonstrates the potential of combining telemetrywith simulation-based experimentation, but these approaches often require complex cloud infrastructure, external APIs, or vendor-specific sensors.
The gap lies in the lack of a lightweight, open-source framework that allows local data collection, experiment scheduling, and system behavior analysis without relying on third-party monitoring suites.

Contribution

This project contributes to a self-contained and reproducible Digital Twin architecture for a physical computing node (server or PC).
Our contribution includes:

- A modular Python system that can generate, execute, and log resource-intensive workloads through “stressors” (CPU, memory, disk).
- A telemetry logger (psutil_logger.py) that records real system metrics in real time, forming a structured dataset(system_metrics_dataset.csv) for later analysis.
- A framework for creating controlled experiments that allow correlation between planned events and measured system responses.
- The foundation for future integration with predictive analytics or machine learning models, enabling simulation of "what-if" scenarios (e.g., predicting system overload).

Methods and Tools

The system consists of three layers:

1. **Stress Generation Layer**
   cpu_stress.py, mem_stress.py and disk_stress.py simulate controlled loads on CPU, RAM, and disk.
2. **Experiment Control Layer**
   make_schedule.py, runner.py and runner_parallel.py generate and execute an event schedule (schedule.json) and record experiment annotations (annotations.csv).
3. **Data Layer –** The psutil_logger.py module acts as a sensor, recording system metrics (CPU %, RAM %, I/O, network, processes, etc.) at defined intervals.

All scripts are implemented in Python using:

- psutil for system telemetry,
- subprocess and multiprocessing for load generation,
- csv and json for data storage, and
- matplotlib / seaborn for analysis and visualization.

The collected data is then cleaned, validated, and analyzed through Exploratory Data Analysis (EDA) techniques (heatmaps, scatter plots, and correlations).

Feasible Goals

The project focuses on the Digital Twin of a single server (asset-level twin), with emphasis on data generation, synchronization, and behavioral analysis.
Real-time feedback control or predictive automation is beyond current scope, but the dataset and architecture are designed to enable these in future work.

Output – Deliverables and Outcomes

Deliverables:

- system_metrics_dataset.csv – dataset containing real-time system metrics during controlled stress tests.
- annotations.csv – experiment timeline with start and end times of each stress event.
- schedule.json – automatically generated experiment plan.
- Project documentation and short report (data description, cleaning, preparation, and EDA).

Expected Outcomes:

- A validated experimental framework for data-driven server monitoring.
- A reproducible dataset suitable for predictive modeling.
- A conceptual and practical demonstration of a Digital Twin for computing infrastructure, aligning with the “Descriptive → Predictive” twin maturity model.