




PROJECT AND TEAM INFORMATION

Project Title

System Performance Analyzer

Student / Team Information

Team Name: Team Id:	Architechs SE(OS)-VI-T250
Team member 1 (Team Lead) Name - Harshit Jasuja Student id – 220211228 Email – harshitjasuja70@gmail.com	
Team member 2: Name - Shivendra Srivastava Student id – 220211349 Email – shivendrasri999@gmail.com	
Team member 3: Name - Yashika Dixit Student id – 22022577 Email – yashikadixit1611@gmail.com	

Motivation

Modern computing systems are becoming increasingly complex, with multiple applications running concurrently and competing for limited system resources. In such environments, efficient resource utilization is critical for optimal performance. However, identifying performance bottlenecks, resource contention issues, and inefficient resource usage patterns remains challenging for system administrators and developers alike.

Traditional system monitoring tools often provide fragmented views of system performance, making it difficult to correlate different performance metrics and identify the root causes of performance issues. Many existing tools either focus on a specific aspect of system performance (e.g., CPU or memory) or provide overwhelming amounts of data without actionable insights.

This project aims to develop a comprehensive System Performance Analyzer that addresses these limitations by providing an integrated view of system performance across multiple dimensions (CPU, memory, disk I/O). The tool will not only collect and display performance metrics in real-time but also analyze historical data to identify patterns, trends, and anomalies. By providing these insights, the System Performance Analyzer will enable system administrators and developers to proactively identify and resolve performance issues, optimize resource allocation, and improve overall system efficiency.

In resource-constrained environments like embedded systems or cloud instances, such optimization can lead to significant cost savings and improved user experience. Additionally, in development environments, the tool can help developers understand the resource requirements of their applications and optimize them accordingly.

State of the Art / Current solution

Modern computing systems are becoming increasingly complex, with multiple applications running concurrently and competing for limited system resources. In such environments, efficient resource utilization is critical for optimal performance. However, identifying performance bottlenecks, resource contention issues, and inefficient resource usage patterns remains challenging for system administrators and developers alike.

Traditional system monitoring tools often provide fragmented views of system performance, making it difficult to correlate different performance metrics and identify the root causes of performance issues. Many existing tools either focus on a specific aspect of system performance (e.g., CPU or memory) or provide overwhelming amounts of data without actionable insights.

This project aims to develop a comprehensive System Performance Analyzer that addresses these limitations by providing an integrated view of system performance across multiple dimensions (CPU, memory, disk I/O). The tool will not only collect and display performance metrics in real-time but also analyze historical data to identify patterns, trends, and anomalies. By providing these insights, the System Performance Analyzer will enable system administrators and developers to proactively identify and resolve performance issues, optimize resource allocation, and improve overall system efficiency.

In resource-constrained environments like embedded systems or cloud instances, such optimization can lead to significant cost savings and improved user experience. Additionally, in development environments, the tool can help developers understand the resource requirements of their applications and optimize them accordingly.

Project Goals and Milestones

General goals

1. Develop a lightweight, cross-platform system performance analyzer that monitors CPU, memory, and disk I/O metrics in real-time.
2. Create an intuitive visualization interface that presents performance data through interactive graphs and logs.
3. Implement data analysis features to identify performance patterns, anomalies, and potential bottlenecks.
4. Design a modular architecture that allows for future extension to monitor additional system resources or integrate with other tools.
5. Ensure minimal impact on system performance while collecting metrics.

Milestones

Milestone 1: Requirements Analysis and Design (Week 1)

- Define detailed functional and non-functional requirements
- Design system architecture and component interfaces
- Select appropriate technologies and libraries
- Create detailed project plan and timeline

Milestone 2: Data Collection Module (Week 2)

- Implement CPU performance metrics collection (/proc/stat, /proc/cpuinfo)
- Implement memory usage metrics collection (/proc/meminfo)
- Implement disk I/O metrics collection (/proc/diskstats, /sys/block)
- Create unified data model for storing and accessing performance metrics

Milestone 3: Data Visualization and User Interface (Week 3-4)

- Develop real-time graphing capabilities for CPU, memory, and disk metrics
- Implement historical data viewing and comparison features
- Create user-friendly dashboard for monitoring all system metrics
- Implement export functionality for reports and logs

Milestone 4: Data Analysis and Alerting (Week 5-6)

- Implement statistical analysis of performance data
- Develop anomaly detection algorithms for identifying unusual system behavior
- Create alerting system for predefined performance thresholds
- Implement resource usage prediction based on historical data

Milestone 5: Testing, Optimization, and Documentation (Week 7-8)

- Perform comprehensive testing across different system configurations
- Optimize tool performance to minimize system impact
- Create user and developer documentation
- Prepare final project deliverables

Project Approach

The System Performance Analyzer will be developed as a modular application with a clear separation of concerns between data collection, storage, analysis, and visualization components. This modular approach will facilitate maintenance, testing, and future extensions.

Technologies and Platforms

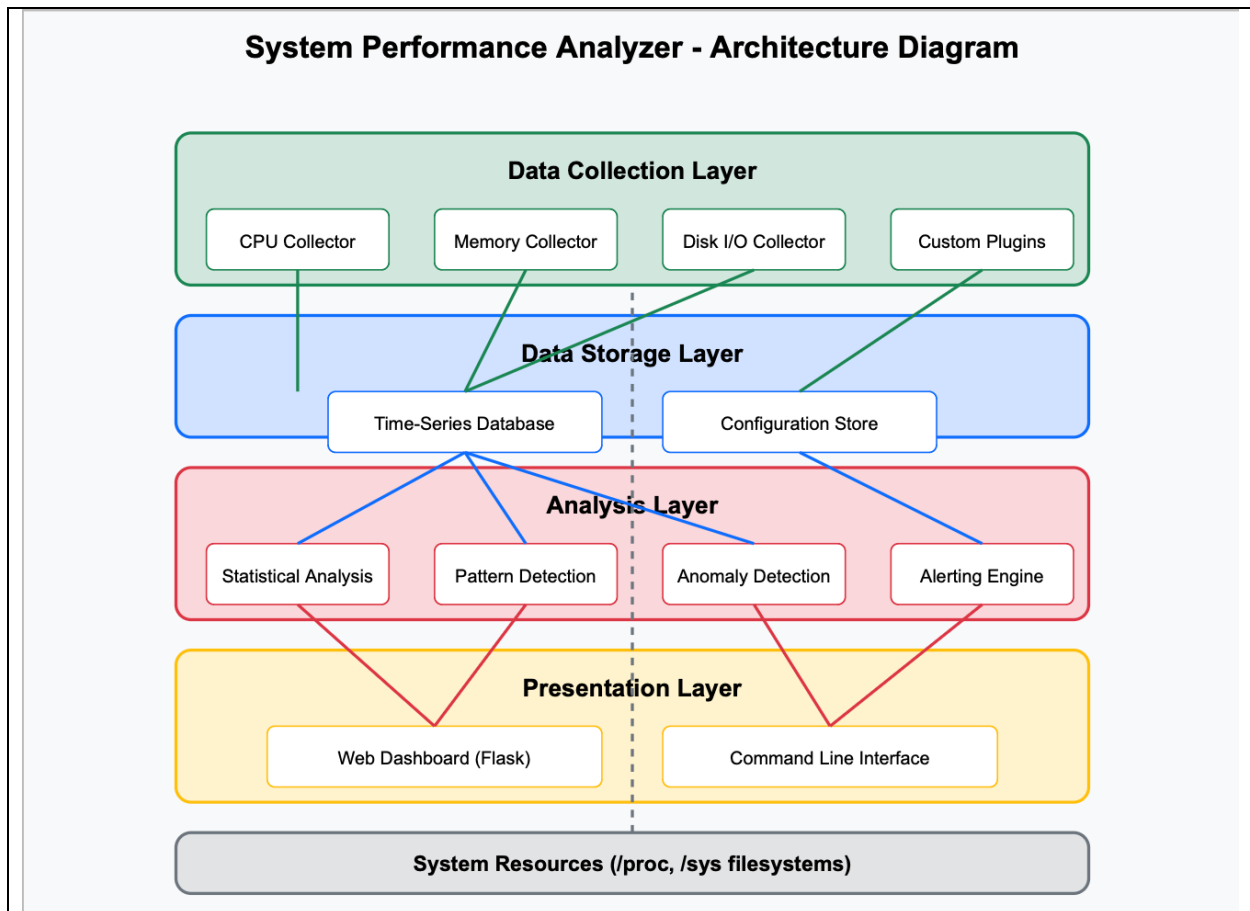
- **Primary Programming Language:** Python will be used for the main application logic, data analysis, and visualization components due to its rich ecosystem of libraries for data processing and visualization.
- **Performance-Critical Components:** C will be utilized for performance-critical data collection modules to minimize overhead and ensure efficient resource utilization.
- **Data Collection:** Direct interaction with Linux's /proc and /sys filesystems will be used to gather system metrics, providing low-level access to system information without requiring additional dependencies.
- **Data Storage:** Time-series data will be stored using SQLite for simplicity and portability, with an option to use InfluxDB for larger deployments.
- **Data Visualization:** Matplotlib and Plotly libraries will power the visualization components, offering interactive graphs and charts.
- **User Interface:** The application will provide both a command-line interface for scripting and automation, and a web-based dashboard built with Flask for visual monitoring and analysis.

Development Methodology

The project will follow an iterative development approach with continuous integration and testing. Each milestone will produce a functional component that can be integrated and tested with existing components.

1. **Data Collection Strategy:** The system will use a hybrid approach for data collection, combining periodic polling of /proc and /sys filesystems with event-based monitoring where appropriate. This will balance comprehensive data collection with minimal system impact.
2. **Analysis Algorithms:** The tool will implement several analytical techniques, including:
 - Moving averages and trend analysis for baseline performance
 - Percentile-based analysis for identifying performance outliers
 - Correlation analysis between different metrics to identify resource contention
 - Machine learning-based anomaly detection for complex patterns
3. **Extensibility:** The system will use a plugin architecture for metrics collection, allowing users to add custom metrics or integrate with existing monitoring systems.
4. **Cross-Platform Considerations:** While initially focusing on Linux systems, the architecture will be designed to accommodate future expansion to other operating systems by abstracting platform-specific data collection.

System Architecture (High Level Diagram)



The diagram illustrates the four main layers of the System Performance Analyzer:

1. **Data Collection Layer** (Green) - Contains collectors for CPU, Memory, Disk I/O, and custom plugins
2. **Data Storage Layer** (Blue) - Includes the time-series database and configuration store
3. **Analysis Layer** (Red) - Features statistical analysis, pattern detection, anomaly detection, and alerting
4. **Presentation Layer** (Yellow) - Provides web dashboard and command-line interfaces

The bottom section represents the system resources (/proc and /sys filesystems) that provide the raw data. The connecting arrows show the data flow between components. Each layer is color-coded for clarity, with individual components displayed within their respective layers.

Project Outcome / Deliverables

The System Performance Analyzer project will yield the following deliverables:

1. **Core Application:** A fully functional system performance analyzer capable of monitoring CPU, memory, and disk I/O metrics in real-time, with minimal impact on system resources.
2. **Web-based Dashboard:** An interactive web interface for visualizing performance metrics through graphs, charts, and tables, allowing users to monitor system performance in real-time and analyze historical data.
3. **Command-line Interface:** A scriptable CLI tool for automation and integration with existing workflows and monitoring systems.
4. **Analysis Module:** Intelligent analysis components that can identify performance patterns, detect anomalies, and provide actionable insights for system optimization.
5. **Alerting System:** Configurable alerting mechanisms to notify users when system metrics exceed defined thresholds.
6. **Documentation:** Comprehensive user and developer documentation, including installation instructions, usage guides, API references, and examples.
7. **Performance Reports:** Templates for generating detailed system performance reports for documentation and stakeholder communication.
8. **Plugin Framework:** An extensible framework that allows users to develop custom metrics collectors and analysis modules.

The final product will be a valuable tool for system administrators, developers, and IT professionals who need to monitor, analyze, and optimize system performance in various environments, from development workstations to production servers and embedded systems.

Assumptions

- The initial implementation will focus primarily on Linux-based systems, leveraging the /proc and /sys filesystems for data collection, with potential future expansion to other operating systems.
- The tool will have minimal dependencies to ensure easy deployment across different environments.
- Users have basic system administration knowledge and understanding of system performance metrics.
- The tool's own resource consumption will be minimal, typically less than 5% of CPU and memory resources, to avoid significantly impacting the monitored system.
- The initial release will focus on local system monitoring, with network-based monitoring capabilities planned for future releases.
- The tool will be suitable for both continuous monitoring in production environments and ad-hoc analysis during development and troubleshooting.

References

- Linux Kernel Documentation: /proc filesystem
- <https://www.kernel.org/doc/Documentation/filesystems/proc.txt>
- Linux Kernel Documentation: /sys filesystem
- <https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt>
- Matplotlib Documentation - <https://matplotlib.org/stable/contents.html>
- Linux Programmer's Manual: proc(5) - <https://man7.org/linux/man-pages/man5/proc.5.html>
- Time Series Database Comparison - <https://db-engines.com/en/ranking/time+series+dbms>