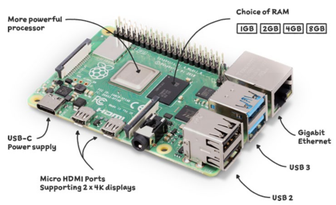
Unbeatable Website on a Pi: A Self-Adaptive Approach

ESASS 2024 Project Poster by Arthur Li & Daniel Almeida (Professor: Dr. Ladan Tahvildari) – Department of Electrical and Computer Engineering

Problem Statement

Reliability of Raspberry Pi Website (RPiWeb)

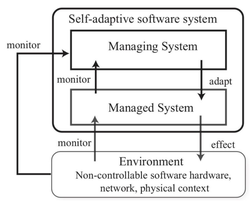
- Introduce dependable Raspberry Pi platform for personal web hosting
- Mitigate crashes, overheating, memory exhaustion
- Eliminate manual labour



Motivation

Raspberry Pi Utility

- Remote location, unreliable environments, resource-constrained
- Downtime, manual restarts, wasted resources/overprovisioning
- Ensure consistent uptime and user confidence
- No additional expenses
- **Deploy-and-forget**



MAPE-K

Monitoring

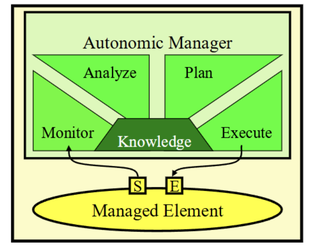
- Python monitoring daemon tracking system metrics
- Enable external weather metric monitoring

Analysis and Planning

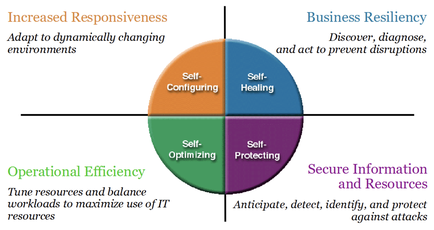
- Optimal metric thresholds
- Content degradation from dynamic to static
- CPU clock speed & voltage

Execution and Knowledge

- Watchdog timer reboots
- Historical data for adaptive anomaly detection
- HTML and reconfiguration adaptation



Self-CHOP



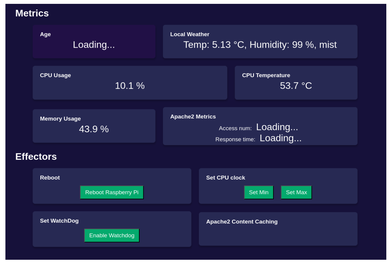
Self-Configuration
- Adjust CPU clock speeds dynamically

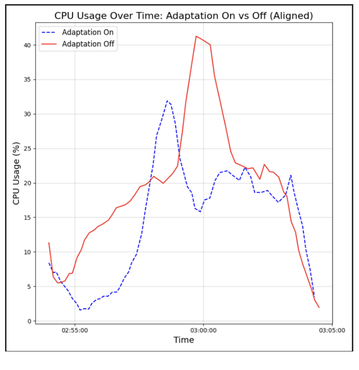
Self-Healing
- Crash recovery via watchdog mechanism

Self-Optimizing
- Content degradation

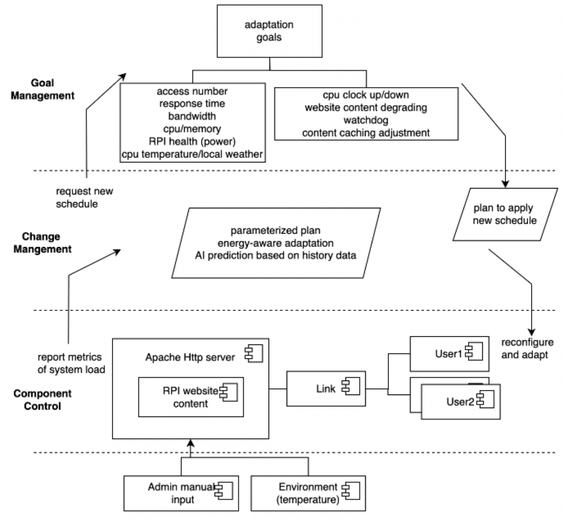
Self-Protection
- Priority-based fallback pages

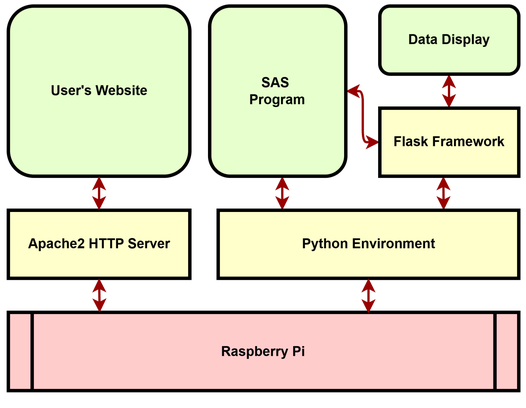
Case Study





Framework





Approach

Monitoring

- Leverage watchdog timers; system hang detection and recovery
- Voltage, CPU Temp, CPU Clock, Latency, CPU Usage, Memory Usage

Implementation

- Apache HTTP Server: Web content hosting and caching tools
- JMeter: User traffic simulation; varying traffic and bandwidth conditions
- Python: Daemon for resource monitoring and adaptation execution

Testing and Validation

- Traffic stress testing; simulating real-world scenarios
- Historical data collection
- Anomaly detection model training

Conclusions & Future Work

Outcome

- Adaptive framework improves RPiWeb hosting
- Dynamic adaptation optimizes performance under fluctuating traffic loads
- Automatic reboot mechanisms ensure minimal downtime and uninterrupted hosting
- Optimal thresholds reduce content oscillation and enable efficient resource allocation

Considerations

- ML improvements with richer historical data for model training
- Framework scalability to even more resource constrained environments