David Christian C. Olimberio

IV – ACSAD

ASSIGNMENT 4: NARRATIVE REPORT

Virtualization vs. Containerization

Executive Summary

This report summarizes a comparative study on virtualization and containerization technologies, focusing on their suitability for application deployment in Edge Computing environments. The research concludes that the optimal technology choice is highly dependent on the specific use case. For short-term, resource-intensive tasks, lightweight container solutions like **Docker Compose with Podman** offer superior performance. Conversely, for complex, long-running services requiring robust management and redundancy, **Kubernetes** is the recommended solution, as its efficiency improves over time, justifying its initial overhead.

1. Introduction: The Challenge at the Edge

The rise of the Computing Continuum, spanning from centralized Cloud servers to decentralized Edge/Fog environments, has created a critical need to choose the right deployment strategies. Edge Computing, which processes data closer to its source, is increasingly vital but often operates under significant resource constraints (e.g., lower power, limited processing).

This study addresses a central question: is traditional **virtualization** (running full virtual machines) or modern **containerization** (running lightweight, isolated applications) the better approach for these resource-constrained Edge environments? The paper evaluates various technology stacks to determine the optimal solution based on performance, scalability, and resource utilization, paying close attention to the differences between **ARM** architectures (common in Edge devices) and traditional **x86** architectures.

2. Methodology: How the Comparison Was Conducted

To provide a comprehensive comparison, the researchers designed an experimental setup to test different technology combinations under stress.

- **Hardware:** The tests were conducted on **ARM-based Raspberry Pi** devices, which are representative of common, low-cost Edge hardware. A comparison with an **x86-based virtual machine** was also included.
- **Technology Stacks Tested:**
    - **Container Runtimes:** Podman and Docker Engine.
    - **Orchestration/Management:** Kubernetes and Docker Compose.
    - **Virtualization:** KVM (Kernel-based Virtual Machine).
- **Benchmarking Tools:**
    - **stress-ng:** This was the primary tool used to apply controlled, reproducible workloads (stressors) to the system's **CPU**, **memory**, and **I/O** subsystems.
    - **top:** This system monitor was used to capture real-time performance data, such as CPU and memory consumption.
- **Performance Metrics:** The key metric for comparison was "Bogo Ops" (Bogus Operations) per second, a measure provided by stress-ng that indicates the amount of "work" a system can perform under a specific load. Tests were run for various durations (e.g., 1, 10, and 20 minutes) to observe both short-term and long-term performance trends.

3. Key Findings and Results

The experimental results revealed a clear trade-off between lightweight efficiency and complex orchestration.

- **Short-Term Performance Winner:** For short-duration tasks (1-10 minutes), the combination of **Docker Compose with Podman** consistently outperformed all other setups, especially in CPU and I/O performance. This is attributed to Podman's daemon-less architecture and the low overhead of Docker Compose, which consumes fewer resources than Kubernetes.
- **Long-Term Performance Winner:** As test durations increased (e.g., 10-20 minutes), **Kubernetes**'s performance became significantly more competitive, narrowing the gap. The findings suggest that Kubernetes is more power-efficient and performs well for long-running, stable services.

- **The "Kubernetes Overhead":** Kubernetes showed higher initial resource consumption (CPU, memory, I/O) due to its complex architecture, which includes an API server, controller manager, and other components. This overhead makes it less suitable for simple, on-demand tasks.
- **Virtualization Performance:** The x86-based virtual machine demonstrated "extremely poor performance" in the tests, highlighting the significant inefficiency and overhead of full hardware emulation compared to lightweight containerization on modern ARM-based Edge devices.

4. Discussion and Conclusion

The study concludes that there is **no single "best" solution**; the optimal choice depends entirely on the deployment scenario.

1. **Recommendation for Short-Term Tasks:** For applications that are resource-efficient, on-demand, or run for short periods (e.g., a one-time data processing job or training an AI model), a lightweight solution like **Docker Compose and Podman** is recommended.
2. **Recommendation for Long-Term Services:** For complex, long-term applications that require high availability, redundancy, and automated error management (e.g., a persistent web service), **Kubernetes** is the superior choice. Its initial resource overhead is justified by its robust orchestration capabilities and improved long-term efficiency.

The research demonstrates that containerization is a highly promising technology for the Edge, offering significant performance benefits over traditional virtualization. The choice between a simple manager like Docker Compose and a full-scale orchestrator like Kubernetes should be made based on the specific needs of the application.