MARK ANGELO D. SIAZON ***IV-ACSAD*** ***ELEC-3***

Narrative Report:

A Comparative Analysis of Virtualization and Containerization for Edge Computing

Based on the article: "Virtualization vs. Containerization, a Comparative Approach for Application Deployment in the Computing Continuum Focused on the Edge" Sturley et.al.)

1. Introduction: The Deployment Challenge

As digital transformation accelerates across various sectors like Industry 4.0 and Smart Cities, the underlying infrastructure for software applications is under critical examination. A key decision for modern systems, particularly in the growing field of **Edge Computing**, is how to deploy applications. Edge environments often operate with constrained resources, making the choice between deployment strategies a vital factor for performance and efficiency.

This report summarizes a research study that directly compares the two most prevalent deployment technologies: **virtualization**, using Virtual Machines (VMs), and **containerization**, using tools like Docker. The study's objective was to evaluate both methods on criteria including performance, scalability, and energy efficiency to determine the optimal environment for hosting applications, especially in the Edge Computing context.

---

2. Background: Core Technologies Compared

The fundamental difference between the two methods lies in their architecture.

- **Virtual Machines (VMs)** are described as a well-established technology. They allow multiple operating systems to run concurrently on a single physical machine by abstracting the hardware layer. This means a VM is a "completely autonomous system" that emulates low-layer components. This approach provides robust isolation and security but also introduces significant overhead, which can impact performance.
- **Containers** offer a more lightweight approach. They isolate applications within containers that all share the host operating system's kernel. This method leads to faster deployment times and more efficient resource utilization. Because they only package the application and its code, they are lighter and more flexible than VMs.

---

3. Methodology and Experimental Setup

To provide a comprehensive comparison, the researchers designed a series of benchmarks using specific hardware and software tools.

- **Hardware Testbeds:** The experiments were conducted on two distinct platforms.
    1. An **ARM-based Edge Device** (a Raspberry Pi 4B+) was selected to represent ARM-based Edge Computing infrastructure.
    2. An **x86 Server** (using a Proxmox virtual environment) was used to represent a traditional, more powerful host system.
- **Measurement Tools:** To gather data, two key tools were employed.
    1. **stress-ng:** This benchmarking tool was chosen to apply a "controlled and reproducible workload" to the systems, testing the CPU, memory, and I/O subsystems. Performance was measured in "Bogo Ops" (bogus operations) to quantify the amount of "work" done.
    2. **UM24C Power Meter:** This hardware logger was used to measure the system's sustainability by tracking actual power consumption in watts.
- **Software Configurations:** The study tested several popular technology combinations, including different container runtimes (Docker Engine, Podman) and orchestrators (Docker Compose, Kubernetes), as well as QEMU-based virtual machines.

---

4. Analysis of Key Findings

The test results revealed significant performance differences, which were often dependent on the duration and nature of the workload.

- **Finding 1: Short-Term Job Performance (1-10 minutes)** For tasks that start, run for a short time, and then stop, simple container setups were found to be superior. The paper notes that the combination of **Docker Compose and Podman** performed better "on all levels" in this context. In contrast, **Kubernetes** was found to be less effective for these short jobs. The paper explains this is due to Kubernetes' complexity, as it requires many components to run (like an API server and controller manager), which creates overhead and consumes more resources. For example, in 1-minute I/O disk tests, Docker Compose setups were on average 40% more efficient.
- **Finding 2: Long-Running Service Performance (10-20+ minutes)** The performance landscape changed when the tests were run for longer periods. The study observed that the performance gap between Kubernetes and other solutions **decreased over time**. This trend supports the hypothesis of its strong long-term efficiency. Most notably, in 20-minute power efficiency tests, the data shows that **Kubernetes' efficiency actually**

surpassed that of Docker. This indicates that while Kubernetes has a higher initial resource cost, it is highly optimized for sustained, continuous workloads.

- **Finding 3: Efficiency on the Raspberry Pi (Edge Device)** The tests on the ARM-based Raspberry Pi provided clear insights for Edge computing. The performance measurements for the native Raspberry Pi, a Docker container, and a properly configured ARM (aarch64) virtual machine were all **"highly similar"**. This important finding demonstrates that both modern containers and proper virtualization (using extensions like KVM) can be highly efficient. In sharp contrast, the emulated x86 virtual machine showed **"extremely poor performance"**. This result highlights the significant inefficiency and difficulty of emulating different hardware architectures.

---

5. Conclusion and Recommendations

The paper concludes that there is no single best solution. The choice **depends on the requirements** of the project, and selecting the right method is crucial for its success. Based on the evidence, the study provides the following guidance:

- **Containers (like Docker)** should be used when flexibility and resource efficiency are critical. They are ideal for resource-constrained environments like Edge Computing and for separating applications on the same host without the "overload" of a virtual OS. For one-time, short-term tasks, the **Docker Compose and Podman** stack was found to be the most resource-efficient.
- **Kubernetes** is recommended for complex architectures that need redundancy, error management, or must run as a long-term service. It is designed for automated scalability, load balancing, and managing container failures. Its higher initial overhead is justified by its superior long-term performance and power efficiency.
- **Virtual Machines (VMs)** remain the relevant choice when complete isolation or specific OS configurations are necessary. A VM is also required when an application needs a graphical user interface for monitoring or must interact with low-layer hardware. The paper specifically notes that accessing devices like **USB sticks** is "an impossible thing to do with containers" but is possible with VMs.