Inicio de Servidores desde Contenedores y desde Máquinas Virtuales. Análisis Comparativo

Jorge Alejandro Kamlofsky[1,2]

[1]CAETI – Universidad Abierta Interamericana – Facultad de Tecnología Informática
Av. Montes de Oca 745, Ciudad de Buenos Aires, Argentina.
jorge.kamlofsky@uai.edu.ar
[0.3cm] [2]GIA: Grupo de Investigación en Inteligencia Artificial – Universidad Tecnológica Nacional
Facultad Regional Haedo. Paris 532, Haedo, Provincia de Buenos Aires, Argentina.
jkamlofsky@frh.utn.edu.ar

Enero-2026

Resumen

Este artículo presenta un análisis comparativo entre la virtualización basada en hipervisores (Máquinas Virtuales) y la virtualización a nivel de sistema operativo (Contenedores), explorando sus arquitecturas, ventajas y casos de uso en el desarrollo de software moderno. Para validar estas diferencias, se presenta una Prueba de Concepto (POC) que realiza una comparación técnica de los tiempos de inicialización y acceso entre un servidor virtualizado en máquina virtual y otro en contenedor, ambos brindando los mismos servicios. Los resultados permiten cuantificar la eficiencia operativa de ambas tecnologías en entornos de despliegue real.

1. Introducción

1.1. Presentación del Tema

La principal diferencia entre VM (siglas en inglés de Máquinas Virtuales) y contenedores radica en la capa de abstracción. Mientras que las VM emulan hardware completo, los contenedores comparten el núcleo del sistema operativo anfitrión.

Esta distinción, tan radical, puede entenderse haciéndose el siguiente cuestionamiento: Si se dispone de una VM en un host que ya está encendido, y se desea iniciar el servidor cuya imagen está en la VM, se tiene que esperar que inicie el hardware emulado en la VM (que insume mucho tiempo). Si el mismo servicio se encuentra en un contenedor, al estar el host ya encendido, el contenedor puede iniciarse casi instantáneamente, ya que no es necesario emular hardware alguno.

1.2. Objetivos

Objetivo general: Comparar el tiempo de inicio y acceso a datos de un servidor virtualizado en una máquina virtual (VM) tradicional versus el mismo servidor ejecutado en un contenedor.

Objetivo particular: Analizar el uso de contenedores para servidores SCADA críticos

redundantes en lugar de utilizar máquinas virtuales, de modo de presentar una recuperación de sistemas más rápida y a menor costo computacional.

1.3. Estructura del trabajo

El resto de este documento se organiza de la siguiente manera:

- La **Sección 2** desarrolla el Marco Teórico sobre las tecnologías en estudio.
- La **Sección 3** detalla el Desarrollo Experimental: la Prueba de Concepto basada en Apache y MySQL.
- La **Sección 4** expone el Análisis de Resultados de las mediciones realizadas.
- La **Sección 5** presenta las Conclusiones obtenidas de la comparativa.

2. Estado del Arte

2.1. Máquinas Virtuales

Las Máquinas Virtuales (VM) representan el enfoque clásico de la virtualización basada en hardware, fundamentado en las exigencias formales de arquitecturas virtualizables que permiten la ejecución de múltiples sistemas operativos sobre un mismo recurso físico [5]. Esta tecnología opera mediante una capa de software crítica denominada hipervisor o Monitor de Máquinas Virtuales (VMM), la cual se encarga de interceptar las instrucciones sensibles y gestionar la asignación de recursos de CPU, memoria y dispositivos de entrada/salida de forma aislada [7].

En su aplicación práctica, si se dispone de una VM en un host que ya está encendido y se desea iniciar un servidor cuya imagen reside en dicha VM, es necesario esperar a que el hardware emulado complete su proceso de arranque (booteo). Cada VM es una entidad autónoma que incluye una copia completa de un sistema operativo invitado (que debe instalarse en el hardware emulado), garantizando un aislamiento total pero con un alto consumo de recursos de memoria y almacenamiento.

2.2. Contenedores

Los contenedores implementan una virtualización ligera a nivel de sistema operativo, compartiendo el núcleo (kernel) del anfitrión en lugar de emular hardware completo. Esta arquitectura se basa en primitivas de aislamiento específicas de Linux como los *Namespaces* y los *Control Groups* (cgroups), que regulan y limitan el consumo de recursos físicos de forma lógica [1].

Al estar el sistema anfitrión ya encendido, un contenedor puede iniciarse de manera casi instantánea (en cuestión de segundos), ya que no requiere la carga de un kernel completo. Esta eficiencia es fundamental para el despliegue de servidores críticos, permitiendo una recuperación de sistemas mucho más rápida y a un menor costo computacional que las máquinas virtuales tradicionales. Además, su portabilidad y baja huella de memoria facilitan la implementación de arquitecturas modernas de microservicios con una densidad de despliegue superior [4]

2.3. Arquitecturas Comparadas

La distinción fundamental entre ambas tecnologías reside en la ubicación de la capa de abstracción y el manejo del sistema operativo. Mientras que las Máquinas Virtuales (VM) emulan hardware completo para ejecutar un núcleo independiente, los contenedores operan compartiendo el núcleo del sistema operativo anfitrión, aislándose únicamente a nivel de procesos mediante el motor de ejecución. Esta diferencia estructural explica por qué las VM presentan un mayor consumo de recursos y tiempos de arranque prolongados, en contraste con la ligereza y agilidad de los contenedores, que prescinden de la duplicación del kernel. Esta convergencia hacia la contenedorización permite una densidad de despliegue significativamente superior en infraestructuras de nube modernas [1].

En la figura 1 se ilustra esta comparación arquitectónica entre ambos enfoques.

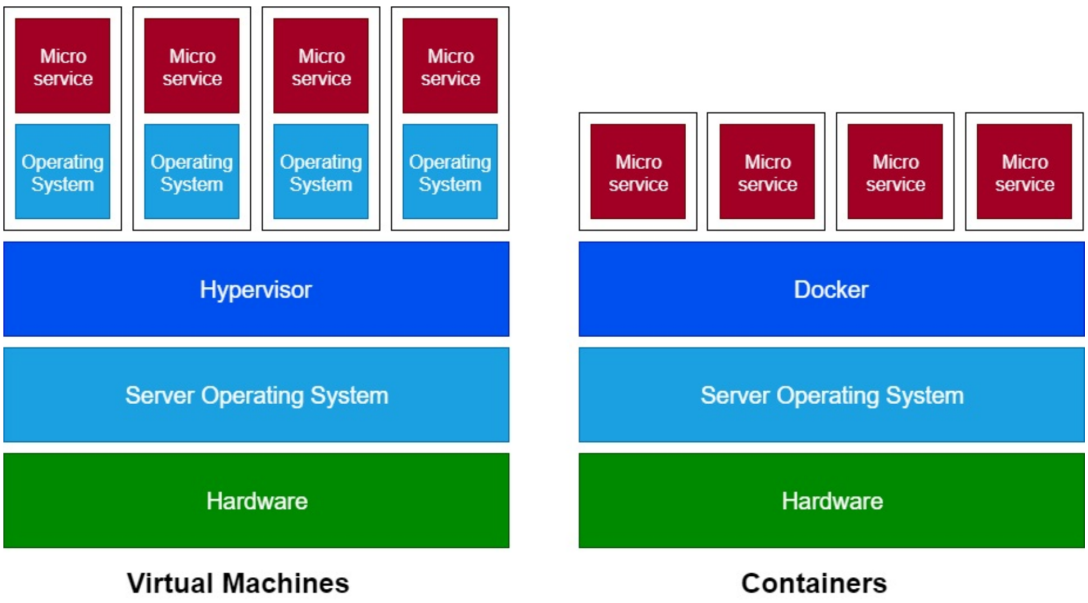

Figura 1: Comparativa Arquitectónica: Máquinas Virtuales vs. Contenedores.[1]

2.4. Trabajos Relacionados

El rendimiento comparativo entre estas tecnologías ha sido documentado extensamente. Felter et al. [2] analizan cómo los contenedores igualan el rendimiento de Linux nativo en CPU y memoria, superando a las VM que requieren una capa adicional de abstracción de hardware.

Por otro lado, Joy [3] destaca la portabilidad y densidad de los contenedores, señalando que permiten una escalabilidad superior al consumir menos recursos de memoria y CPU que una VM equivalente.

Finalmente, Shah et al. [6] exploran el despliegue de microservicios, subrayando que la simplicidad en el despliegue y la gestión de dependencias son factores determinantes para elegir la contenedorización en arquitecturas modernas de software.

[1]Fuente: `https://comparacloud.com/servicios-clouds/precios-cloud/`

3. Desarrollo Experimental

3.1. Resumen de la Prueba de Concepto (POC)

Se instaló un servicio Apache sobre un Ubuntu Server con una base de datos MySQL tanto en una máquina virtual sobre Virtualbox como en un contenedor docker. Se inició ambos servicios y se midió el tiempo de respuesta de ambos.

Para atestiguar que el servicio se haya iniciado efectivamente, al iniciar cada servicio se realizó una consulta a una base de datos presente en el servidor.

Para la realización de esta POC se utilizaron los siguientes servicios: Virtualbox como gestor de VM, Docker para la gestión de contenedores, Apache como servidor web y MySQL como sistema gestor de bases de datos. El IDE de desarrollo de la POC fue Visual Studio Code.

3.2. Escenario: Contenedores Docker

3.2.1. Inicialización del Contenedor

Se creó un contenedor Docker basado en la imagen oficial de Ubuntu Server, instalando Apache y MySQL mediante un Dockerfile personalizado. El contenedor se configuró para iniciar automáticamente los servicios al arrancar.

En la carpeta "test-virtualizacion" se presentan los archivos de programación de la POC. Los contenedores se encuentran en una única red, como se muestra en la figura 2 (la figura se obtuvo de VS-Code, extensión Docker). La red contiene: la parte del sitio, y la parte del servidor y base de datos.

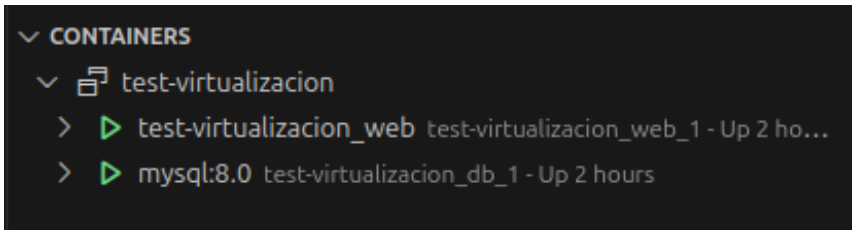

Figura 2: Red de Contenedores Docker utilizada en la POC.

3.2.2. Acceso al Sitio

Para acceder al sitio web alojado en el contenedor, se utilizó un navegador web apuntando a http://localhost:8080. La consulta a la base de datos se realizó mediante un script PHP alojado en el servidor Apache, que se ejecutó automáticamente al acceder a la página principal. La figura 3 muestra la página web servida desde el contenedor Docker.

3.2.3. Gestión de Servicios con Docker Compose

Para la administración del ciclo de vida de los servicios en la Prueba de Concepto, se utilizaron los siguientes comandos:

- **Construcción e inicio:** `docker-compose up --build -d` (construye las imágenes y levanta los servicios en segundo plano).

Figura 3: Página web servida desde el contenedor Docker.

- **Apagado y eliminación:** `docker-compose down` (detiene los contenedores y elimina las redes creadas).
- **Solo apagado:** `docker-compose stop` (detiene los servicios manteniendo las imágenes y el estado actual).
- **Solo inicio:** `docker-compose start` (inicia los servicios a partir de imágenes ya existentes).

Es importante destacar que, gracias al uso de volúmenes de datos, la información de la base de datos persiste incluso después de reiniciar los servicios.

3.3. Escenario: Máquina Virtual con VirtualBox

3.3.1. Resumen

Se creó una máquina virtual en Virtualbox basada en Linux, distribución Ubuntu versión 25.04. En la misma se instalaron los servicios: Apache, PHP y base de datos MySQL. Se incluye una página similar a la presentada, que se muestra localmente en http://localhost:8080/.

3.3.2. Instrucciones para Acceder al Sitio

La máquina virtual instalada incluye interfaz gráfica. Por lo tanto, puede accederse a un navegador (Mozilla Firefox). Abrir un navegador y navegar a http://localhost:8080/, igual a el ejemplo de contenedores. Aparece una imagen similar a la presentada en el otro ejemplo. Al aparecer la tabla completa, se evidencia el correcto inicio de los servicios: Apache, PHP, Mysql.

3.3.3. Ejecución del Experimento

En Visual Studio Code se instalaron los siguientes complementos: Docker (containers) y Jupyter para el armado de una POC interactiva.

Se compararon los tiempos de acceso efectivo al sitio operando con contenedores y máquinas virtuales, en similares condiciones.

Se estudió la disponibilidad del servidor desde power-off: en caso de corte de energía general, se afecta al scada-server y a su redundancia, si está en la misma línea de alimentación,

o si no se dispone de redundancia y se requiere un apagado forzado. Se analizaron dos casos:

1. **Acceso a la información con servidores encendidos y conectados:** esto es de utilidad para cuando se dispone de servidores con redundancia, conectados en un hot-standby.
2. **Disponibilidad del servidor desde power-off::** en caso de corte de energía general, se afecta al scada-server y a su redundancia, si está en la misma línea de alimentación, o si no se dispone de redundancia y se requiere un apagado forzado.

sectionAnálisis de Resultados Se midieron los tiempos de inicio y acceso a datos en ambos escenarios. Los resultados obtenidos se resumen en la Tabla 1.

4. Análisis de Resultados

En esta sección se presentan y analizan los datos recolectados durante las experiencias, organizados por el tipo de prueba realizada.

4.1. Comparativa de Tiempos de Inicialización

En la Tabla 1 se detallan los tiempos obtenidos desde el estado de *power-off* hasta la disponibilidad total de los servicios[cite: 36, 40]. Se observa una diferencia drástica en el tiempo total, donde los contenedores resultan un orden de magnitud más rápidos que las máquinas virtuales.

Entorno	**Etapa de Inicialización**	**Tiempo (s)**
Contenedores	Inicio total de servicios	4,80
Máquina Virtual	Inicio de VM	1,10
	Activación de Apache (Puerto 8080)	37,30
	Operatividad de MySQL y DB	6,36
	Tiempo TOTAL de inicio completo	**44,76**

Cuadro 1: Desglose de tiempos de inicio desde estado Power-off.

4.2. Comparativa de Tiempos de Acceso

La Tabla 2 muestra los tiempos de acceso medidos con los servidores ya encendidos y conectados. En este caso, los valores son similares para ambas tecnologías, manteniéndose en el mismo orden de magnitud.

Tecnología de Virtualización	**Tiempo de Acceso (s)**
Servidores Contenedorizados	0,05
Servidores en Máquina Virtual	0,03

Cuadro 2: Tiempos de acceso con servidores en estado operativo.

4.3. Interpretación de los Resultados

Para el caso de recuperación tras un fallo total de energía, la tecnología de contenedores presenta una ventaja crítica, reduciendo el tiempo de inactividad de 44,76 a solo 4,80 segundos. Por otro lado, la diferencia en los tiempos de acceso (30 ms vs 50 ms) resulta marginal para la mayoría de las aplicaciones críticas, aunque favorece levemente a la máquina virtual.

5. Conclusiones

El presente trabajo permitió contrastar empíricamente las diferencias de rendimiento entre la virtualización tradicional y la contenedorización. Los resultados obtenidos en la Prueba de Concepto validan las hipótesis planteadas respecto a la agilidad y el bajo costo computacional de los contenedores frente a las Máquinas Virtuales.

En términos de disponibilidad desde un estado de apagado total (*power-off*), la tecnología de contenedores demostró ser superior por un orden de magnitud, logrando la operatividad total en 4,80 segundos, comparado con los 44,76 segundos requeridos por la VM. Esta diferencia de casi el 832 % es crucial para sistemas críticos redundantes, donde la velocidad de recuperación ante un fallo de energía determina la continuidad del servicio y la integridad de los datos industriales.

Por otro lado, el análisis de los tiempos de acceso con servidores ya operativos reveló una paridad técnica (30 ms para VM vs. 50 ms para contenedores). Si bien la máquina virtual presenta una ventaja mínima, ambos resultados se encuentran en el mismo orden de magnitud, lo que confirma que el uso de contenedores no introduce una penalización de latencia significativa una vez que el sistema está en ejecución.

Como conclusión final, se recomienda la adopción de arquitecturas contenedorizadas para servidores críticos que requieran alta disponibilidad y redundancia rápida. Si bien los valores absolutos obtenidos corresponden a la aplicación específica de esta POC, se espera que en otras condiciones de despliegue los resultados mantengan una proporcionalidad similar, favoreciendo la eficiencia operativa y la optimización de recursos de hardware.

Referencias

[1] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[2] Wes Felter, Alexandre Ferreira, Ramakrishnan Rajamony, and Juan Rubio. An updated performance comparison of virtual machines and linux containers. In *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pages 171–172. IEEE, 2015.

[3] Amit M Joy. Performance comparison between linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 342–346. IEEE, 2015.

[4] Claus Pahl. Containerization and the paas cloud. In *2015 IEEE International Conference on Cloud Engineering*, pages 363–365, 2015.

[5] Gerald J Popek and Robert P Goldberg. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421, 1974.

[6] Jignesh Shah and Dhara Dubaria. Performance comparison of container-based virtualization and virtual machine-based virtualization. *International Journal of Computer Applications*, 183(1):20–25, 2021.

[7] James Edward Smith and Ravi Nair. *Virtual machines: versatile platforms for systems and processes*. Elsevier, 2005.