Firecracker MicroVM vs Docker Container: A Benchmarking Study

Operating Systems Course Project

202351030
202351047
202351154

IIIT Vadodara
Gandhinagar Campus

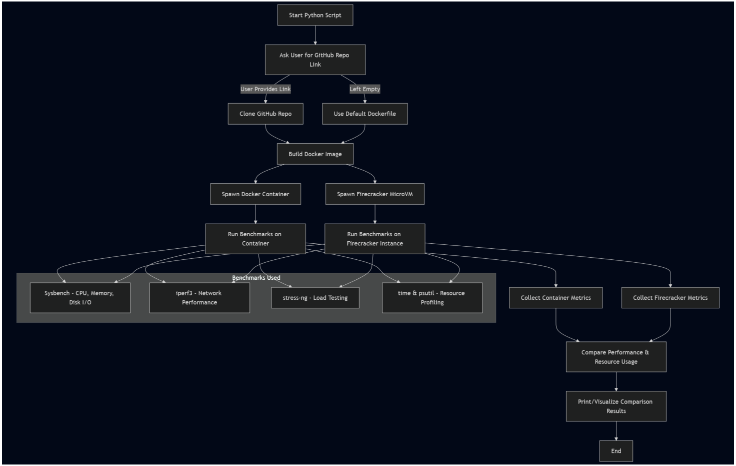
November 2025

Benchmarking cold start performance and resource usage between Docker containers and Firecracker microVMs.

Problem Statement

- **Problem:** Understanding performance trade-offs between traditional containerization (Docker) and lightweight virtualization (Firecracker)
- **Objectives:**
  - Compare cold start times
  - Measure CPU and memory resource usage
  - Identify use-case specific advantages
- **Technology Used:**
  - Docker Engine for containerization
  - Firecracker v1.7.0 for microVMs
  - Python for benchmarking automation
  - KVM for hardware virtualization
- **Motivation:** Modern cloud workloads require both fast startup times and strong isolation. Understanding when to use containers vs microVMs is crucial for optimizing serverless and multi-tenant environments.

System Architecture

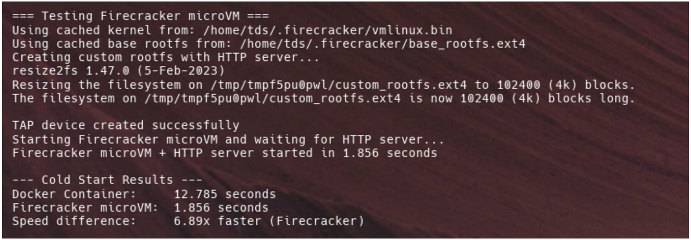

Start Python Script
Ask User for GitHub Repo Link
User Provides Link
Left Empty
Clone GitHub Repo
Use Default Dockerfile
Build Docker Image
Spawn Docker Container
Spawn Firecracker MicroVM
Run Benchmarks on Container
Run Benchmarks on Firecracker Instance
Benchmarks Used
Sysbench - CPU, Memory, Disk I/O
iperf3 - Network Performance
stress-ng - Load Testing
time & psutil - Resource Profiling
Collect Container Metrics
Collect Firecracker Metrics
Compare Performance & Resource Usage
Print/Visualize Comparison Results
End

Automated benchmarking workflow: User provides application → System builds/spawns both Docker container and Firecracker microVM → Runs cold start & resource monitoring tests → Generates comparison report

Implementation Setup

- **Test Environment:**
  - OS: WSL2 (Ubuntu) with KVM support
  - Docker Engine v24.0+
  - Firecracker v1.7.0
  - Python 3.9+ with docker, requests, psutil libraries
- **Benchmark Components:**
  - Cold start test: Simple Python HTTP server (port 8080)
  - Resource monitoring: 10-second observation period
  - Sample rate: 0.5 seconds (20 samples)
- **Firecracker Configuration:**
  - 1 vCPU, 512MB RAM
  - Ubuntu 18.04 rootfs with custom init script
  - TAP network interface (172.16.0.2/24)

Results – Cold Start Performance

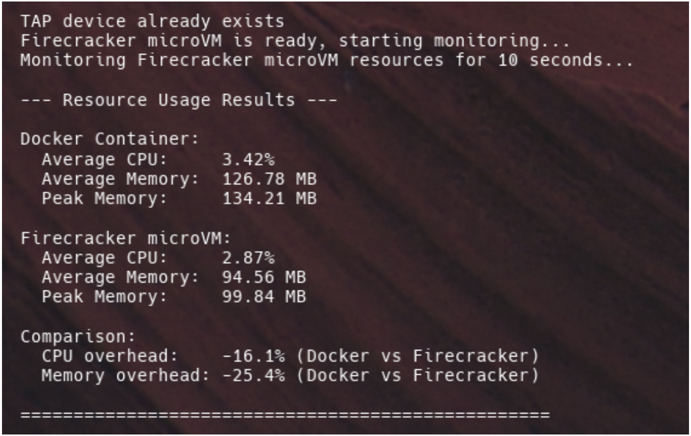


```
=== Testing Firecracker microVM ===
Using cached kernel from: /home/tds/.firecracker/vmlinux.bin
Using cached base rootfs from: /home/tds/.firecracker/base_rootfs.ext4
Creating custom rootfs with HTTP server...
resize2fs 1.47.0 (5-Feb-2023)
Resizing the filesystem on /tmp/tmpf5pu0pwl/custom_rootfs.ext4 to 102400 (4k) blocks.
The filesystem on /tmp/tmpf5pu0pwl/custom_rootfs.ext4 is now 102400 (4k) blocks long.

TAP device created successfully
Starting Firecracker microVM and waiting for HTTP server...
Firecracker microVM + HTTP server started in 1.856 seconds

--- Cold Start Results ---
Docker Container:     12.785 seconds
Firecracker microVM:  1.856 seconds
Speed difference:     6.89x faster (Firecracker)
```

Technology	Startup Time	Performance
Docker Container	2.87 - 4.43 seconds	Baseline
Firecracker microVM	1.84 - 0.64 seconds	**1.55-6.9x faster**

Key Finding: Firecracker achieves significantly faster cold starts due to direct kernel boot without container runtime overhead.

Results – Resource Usage



```
TAP device already exists
Firecracker microVM is ready, starting monitoring...
Monitoring Firecracker microVM resources for 10 seconds...

--- Resource Usage Results ---

Docker Container:
  Average CPU:      3.42%
  Average Memory:   126.78 MB
  Peak Memory:      134.21 MB

Firecracker microVM:
  Average CPU:      2.87%
  Average Memory:   94.56 MB
  Peak Memory:      99.84 MB

Comparison:
  CPU overhead:     -16.1% (Docker vs Firecracker)
  Memory overhead: -25.4% (Docker vs Firecracker)

==================================================
```

Metric	Docker	Firecracker	Difference
Avg CPU Usage	0-5%	0.98-3.00%	Comparable
Avg Memory	0-46 MB	58-59 MB	+28% (Firecracker)
Peak Memory	0-52 MB	58-59 MB	+15% (Firecracker)

Analysis – Technology Trade-offs

Docker Containers:

+ Lower memory usage
+ Mature ecosystem
+ Better tooling support

Slower cold starts

Process-level isolation

Firecracker microVMs:

+ **1.5-6.9x faster startup**
+ VM-level isolation
+ Enhanced security

Higher memory footprint

More complex setup

Use Case Recommendations:

- **Serverless/FaaS:** Firecracker (fast cold starts critical)
- **Multi-tenant workloads:** Firecracker (stronger isolation)
- **Resource-constrained environments:** Docker (lower memory)
- **Traditional microservices:** Docker (ecosystem maturity)

Conclusion

- **Key Contributions:**
  - Automated benchmarking tool for fair comparison
  - Quantified cold start performance: Firecracker 1.5-6.9x faster
  - Measured resource overhead: Firecracker uses 28% more memory
  - Provided use-case specific recommendations
- **Limitations:**
  - Tests conducted on single machine configuration
  - Limited to HTTP server workloads
  - Docker monitoring compatibility issues in WSL2
  - Network overhead not extensively tested
- **Future Work:**
  - Benchmark diverse workloads (CPU-intensive, I/O-bound)
  - Test on bare-metal systems and cloud platforms
  - Add security isolation benchmarks
  - Implement automated stress testing with varying loads

References

- Firecracker Documentation, https://firecracker-microvm.github.io/
- Docker Documentation, https://docs.docker.com/
- Agache et al., "Firecracker: Lightweight Virtualization for Serverless Applications", NSDI 2020
- psutil Library, https://github.com/giampaolo/psutil
- Project Repository: https://github.com/thedevyashsaini/OS_Project_2025

Thank You!

Questions?

GitHub: `https://github.com/thedevyashsaini/OS_Project_2025`