# Benchmarking Containers on Phase 2 SOC's

**Author**: Casey Pancoast
**Date**: 29 June 2020

We compared performance between the Apollo 5 and a docker container (official CentOS image) installed on the Apollo using a few common benchmarks, as well as their CPU usage while running them.

The Apollo will be used in the future track-trigger, and putting containers onto it would be convenient. However, we need to know that the containers won't kneecap the Apollo in terms of its processing power or resource usage. To perform their function appropriately, the Apollos will need to network with other machines at the TIF, so it would be useful to compare networking benchmarks between the Apollo and the container on the Apollo. However, this is not possible at the time due to an unintentional restriction on the Apollo's networking capabilities that should be resolved shortly, but after this author is no longer afilliated with the CMS collaboration.

Running the non-networking benchmarks revealed no difference in performance or CPU usage between the host Apollo and the container running on it.

## Overview

### Goal

The goal of this study was to determine the difference in performance and resource usage between the Apollo 5 and a container running on it.

### Machines

**Apollo**: The host OS for the container is the Apollo 5 at the CMS TIF. This Apollo and others like it will be used in the proposed track-trigger algorithm, to be implemented in the Phase 2 CMS upgrade.

**Docker on Apollo**: The container on apollo was run from this image, which is the official CentOS image for armv7l. From brevity's sake, I'll be referring to the container run from that image as the *Docker-on-Apollo*.

### Methods

Benchmarks were run using the benchmarking script in this system-on-chip benchmarking repository. The benchmark information requested was from Coremark, Dhrystone, and Whetstone.

## Comparing the Machines by Benchmarks

The scores below show how the container and the host image compare on three common benchmarks.

### Coremark

The coremark scores for the Apollo and Docker-on-Apollo are within statistical uncertainty of one another.

**Results, in iterations per second:**

**Apollo Mean/St.dev:** (3.73 ± 0.03) * 10^3
**Docker-on-Apollo Mean/St.dev:** (3.68 ± 0.03) * 10^3

### Dhrystone

These scores are exactly the same due to the fact that Dhrystone collects its results by integer dividing a total number of operations by an integer number of seconds. It is yet unknown if this is a mistake in Dhrystone — this is unlikely, due to the pedigree of Dhrystone, but why it's done this way is a mystery.

Anyway, even with this integer division affecting the raw scores, the experimental uncertainty still accurately reflects what it would have been without integer division. Ten trials were run, and Dhrystone scores (with and without registers) are, to put it lightly, within experimental uncertainty of one another.

**Results in Dhrystones per second:**

#### With Registers

**Apollo Mean/St.dev:** (2.48 ± 0.03) * 10^6
**Docker-on-Apollo Mean/St.dev:** (2.48 ± 0.03) * 10^6

#### Without Registers

**Apollo Mean/St.dev:** (2.47 ± 0.03) * 10^6
**Docker-on-Apollo Mean/St.dev:** (2.47 ± 0.03) * 10^6

### Whetstone

For the same reason as Dhrystone results, and with the same consequences, Whetstone scores for the Apollo and the Docker-on-Apollo are identical.

**Results, in Whetstones per second:**

**Apollo Mean/St.dev:** (5.0 ± 0.1 * 10^3)
**Docker-on-Apollo Mean/St.dev:** (5.0 ± 0.1 * 10^3)

### CPU usage

A note here: unlike with Dhrystone and Whetstone, these scores were not exactly the same. They only became so after appropriate significant-figure rounding.

The gist is here, these are also the same.

**Apollo Mean/St.dev:** (110 ± 30)%, *2 cores*
**Docker-on-Apollo Mean/St.dev:** (110 ± 30)%, *2 cores*

## Discussion

In a nutshell, *all benchmarking scores comparing the Apollo to the Docker-on-Apollo were within experimental certainty of one another*. This implies that running processes in a container rather than on the host Apollo has no measurable effect on the performance of those processes. However, it doesn't rule out the fact that running the container itself may consume significant resources.

## Future Work

- Accurate networking benchmarks were not able to be taken before the conclusion of the author's afilliation with CMS — this is a natural next step.
- A CentOS container may not be the final container used. This study is not difficult to run, and should be replicated for the appropriate container.
- Memory scores were not taken, and they should be.
- CPU and memory readings should be taken for the container process itself, not just for the processes on it, to determine whether the container itself consumes significant resources.