The impact of Docker containers on the performance of genomic pipelines

Paolo Di Tommaso*[1,2], Emilio Palumbo[1,2], Maria Chatzou[1,2], Pablo Prieto[1,2], Michael L Heuer[3], Cedric Notredame[1,2]

[1]Bioinformatics and Genomics Program,
Centre for Genomic Regulation (CRG), Dr. Aiguader 88, 08003 Barcelona, Spain
[2]Universitat Pompeu Fabra (UPF), 08003 Barcelona, Spain
[3]National Marrow Donor Program, Minneapolis, MN 55413-1753, USA
* Corresponding author: paolo.ditommaso@crg.eu

Abstract

Genomic pipelines consist of several pieces of third party software and, because their experimental nature, frequent changes and updates are commonly necessary thus raising serious distribution and reproducibility issues. Docker containers technology offers an ideal solution, as it allows the packaging of pipelines in an isolated and self-contained manner. This makes it easy to distribute and execute pipelines in a portable manner across a wide range of computing platforms. Thus the question that arises is to what extent the use of Docker containers might affect the performance of these pipelines. Here we address this question and conclude that Docker containers have only a minor impact on the performance of common genomic pipelines, which is negligible when the executed jobs are long in terms of computational time.

Introduction

Genomic pipelines usually rely on a combination of several pieces of third party research software. These applications tend to be academic prototypes that are often difficult to install, configure and deploy. Furthermore their experimental nature can result in frequent updates, thus raising serious reproducibility issues. In the past virtual machines were proposed as an answer to this issue. They are indeed very convenient but come along with a few major issues that include high latency and significant overhead.

Docker containers technology has been designed to address these issues. It has recently received an increasing level of attention throughout the scientific community because it allows applications to run in an isolated, self-contained package that can be efficiently distributed and executed in a portable manner across a wide range of computing platforms (Boettiger, 2015).

The first most obvious advantage of this approach is to replace the tedious installation of numerous pieces of software, with complex dependencies, by simply downloading a single pre-built ready-to-run image containing all the software and the required configuration.

The second strength of Docker is to run each process in an isolated container that is created starting from an immutable image. This prevents conflicts with any other installed program in the hosting computing environment, and guarantees that each process runs in a predictable system configuration that cannot change over time due to misconfigured software, system updates or programming errors.

Containers only require a few milliseconds to start and many instances can run in the same hosting environment. This is possible because it runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers.

A study from IBM Research showed that Docker technology introduces a negligible overhead for CPU and memory performance, and applications running in a container perform equally or better when compared to KVM virtualization in all tests (Felter et al., 2014).

In this work we assess the impact of Docker containers on the performance of genomic pipelines using a realistic computational biology usage scenario based on the re-computation of selected subsets of the ENCODE analysis.

Method

In order to evaluate the impact of Docker usage on the execution performance of bioinformatics tools we benchmarked three different genomic pipelines. A comparison of the execution times was made running them with and without Docker along with the same dataset. The tests were executed using a cluster node HP BL460c Gen8 with 12 cpus Intel Xeon X5670 (2.93GHz), 96 GB of RAM and running on Scientific Linux 6.5 (kernel 2.6.32-431.29.2.el6.x86_64).

Tests were executed using Docker 1.0 configured with "device mapper" as the storage driver. Docker images used for the benchmark were built starting from a Scientific Linux 6.5 base image. The compute node was reserved for the benchmark execution (this means that no other workload was dispatched to it), moreover to prevent any possible network latencies that could affect the execution times in an aleatory manner, all tests were executed using the node local disk as main storage.

All three pipelines are developed with Nextflow, a tool that is designed to simplify the deployment of computational pipelines across different platforms in a reproducible manner (Di Tommaso et al., 2014). Nextflow integrates the support for Docker allowing pipeline tasks to be executed transparently in Docker containers.

This allowed us to execute the same pipeline natively or run it with Docker without having to modify the pipeline code, but by simply specifying the Docker image to be used in the Nextflow configuration file.

It should be noted that when the pipeline is executed with Docker support it does not mean that the overall pipeline execution runs "inside" a single container, but that each task spawned by the pipeline runs in its own container. This approach allows a Docker based pipeline to use a different image for each different task in the computational workflow, and therefore scale seamlessly in a cluster of computers (which wouldn't be possible using the single container approach).

The overhead introduced by containers technology on the pipelines performance was estimated by comparing the median execution time of 10 instances running with and without Docker. As the pipeline ran parallel tasks, the execution time was normalized summing up the execution time of all the tasks in each instance.

Benchmark 1

The first performance evaluation was carried out using a simple pipeline for RNA-Seq data analysis (15).

The pipeline takes raw RNA-Seq sequences as input and first maps them to a reference genome and a transcript annotation by sequence alignment. The mapping information is then used to quantify known transcripts using the reference transcript annotation. For each processed sample, the pipeline produces as output a table of relative abundances of all transcripts in the transcript annotation.

The pipeline was run 10 times using the same dataset with and without Docker. The RNA-Seq data was taken from the ENCODE project and contained randomly sampled (10% of the original) Illumina paired-end sequences from brain samples (CNS) of mouse embryos at day 14 and day 18, in 2 bioreplicates. Each run executed a first *index* task using Bowtie, then a *mapping* task using Tophat2 and finally a *transcript* task using the Cufflinks tool. The following versions of these tools were used: Samtools 0.1.18 (Li et al., 2009), Bowtie2 2.2.3 (Langmead et al., 2012), Tophat-2.0.12 (Kim et al., 2013), Cufflinks 2.2.1 (Trapnell et al., 2010).

Each run executed 9 tasks. The median pipeline execution time in the native environment was 1,158.4 minutes (19h 18m 23s), while the median execution time when running it with Docker was 1,157.6 minutes (19h 17m 35s). Thus, the use of Docker containers didn't add any time overhead to the pipeline execution, on the contrary the median execution time was a few seconds faster (0.1%).

Pipeline	Tasks	Mean task time (mins)		Median execution time (mins)		Slow down
		native	docker	native	docker	
RNA-Seq	9	128.7	128.6	1,158.4	1,157.6	**0.999**
Variant call.	48	26.1	26.7	1,252.6	1,283.6	**1.025**
Piper	98	0.6	1.0	58.5	97.1	**1.659**

Table 1

Benchmark 2

The second benchmark was executed using an assembly-based variant calling pipeline, part of a Minimum Information for Reporting Next Generation Sequencing Genotyping (MIRING)-compliant genotyping workflow for histocompatibility, immunogenetic and immunogenomic applications (Mack, 2015).

Paired-end genomic reads from targeted human leukocyte antigen (HLA) and killer-cell immunoglobulin-like receptors (KIR) genes are assembled into consensus sequences. Reads and consensus sequences are then aligned to the human genome reference and used to call variants.

The pipeline was launched 10 times using Illumina paired end genomic reads targeted for major histocompatibility complex (MHC) class I HLA-A, HLA-B, and HLA-C genes and MHC class II gene HLA-DRB1 from 8 individuals. The following versions of these tools were used both in the native and Docker environment: ngs-tools 1.7, SSAKE 3.8.2 (Warren et al., 2007), BWA 0.7.12-r1039 (Li et al., 2010), Samtools 1.2 (Li et al., 2009).

Each run executed 48 tasks, and the maximum number of tasks that could be executed in parallel was set to 10. Most of the tasks completed in a few seconds, with the exclusion of the SSAKE stage which needed from 2 to 3.5 hours to complete (see fig. 2).

The median pipeline execution time in the native environment was 1,252.6 minutes (20h 52m 34s), while the median execution time when running it with Docker was 1,283.6 minutes (21h 23m 38s). This means that when running with Docker the execution was slowed down by 2.5% (see table 1).

Benchmark 3

The last benchmark was carried out using Piper-NF, a genomic pipeline for the detection and mapping of long non-coding RNAs.

The pipeline takes as input cdna transcripts sequences in FASTA format which are blasted against a set of genomes also provided in FASTA format. Homologous regions on the target genomes are used as anchor points and the surrounding regions are then extracted and re-aligned with the original query. If the aligner can align these sequences and the alignment covers a required minimal region of the original query, the sequences are used to build a multiple sequence alignment that is then used to obtain the similarity between each homologous sequence and the original query.

As in previous experiments the pipeline was run 10 times using the same dataset with and without Docker. We used as the input query a set of 100 RNA-Seq transcript sequences in FASTA format from Gallus gallus species. The input sequences were mapped and aligned towards a set of genomes consisting of Anas platyrhynchos, Anolis carolinensis, Chrysemys picta bellii, Ficedula albicollis, Gallus gallus, Meleagris gallopavo, Melobsittacus undulatus, Pelodiscus sinensis, Taeniopygia guttata, from Ensembl version 73. The following versions of the tools were used both in the native and Docker environment: T-Coffee 10.00.r1613 (Notredame et al., 2000), NCBI BLAST 2.2.29+ (Altschul et al., 1990), Exonerate 2.2.0 (Slater, Birney, 2005). Each run executed 98 jobs and the maximum number of tasks that could be executed in parallel was set to 10.

The median pipeline execution time in the native environment was 58.5 minutes, while the median execution time when running it with Docker was 97.1 minutes. In this experiment running with Docker introduced a significative slowdown of the pipeline execution time, around 66% (see table 1).

This result can be explained by the fact that the pipeline executed many short-lived tasks: the mean task execution time was 35.8 seconds, and the median execution time was 5.5 seconds (see fig. 3). Thus the overhead added by Docker to bootstrap the container environment and mount the host file system became significative when compared to the short task duration.

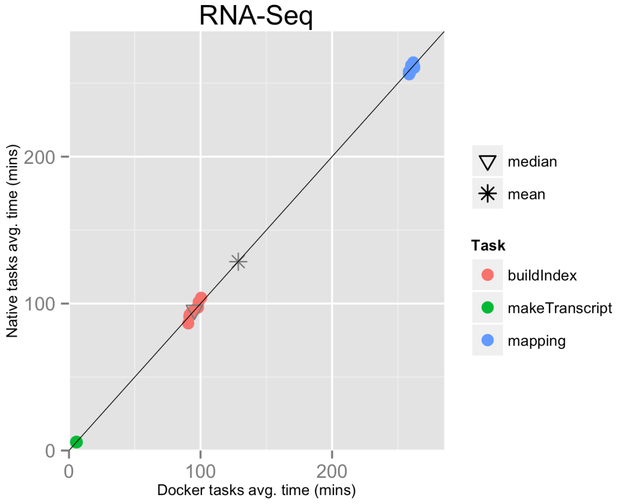

Figure 1

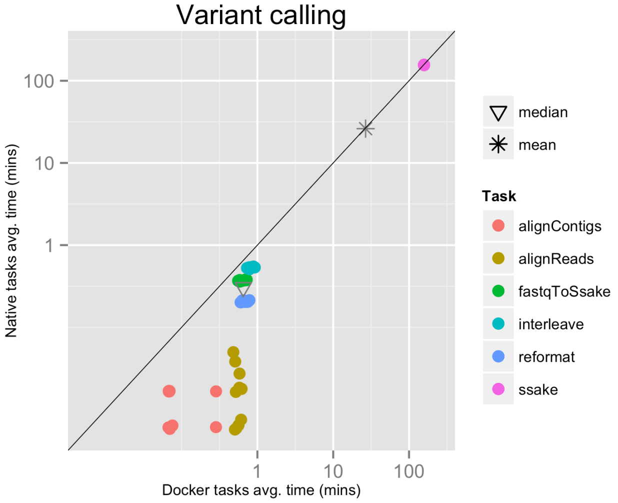

Figure 2

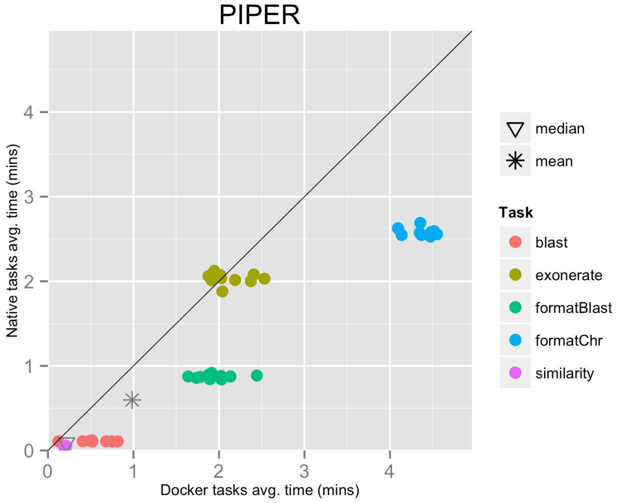

Figure 3

Results

In this work we assessed the impact of Docker containers technology on the performance of genomic pipelines. We showed that container "virtualization" has a negligible overhead on pipeline performance when it is composed by medium/long running tasks, which is the most common scenario in computational genomic pipelines. While the performance degradation is more significative for pipelines where most of the tasks have a fine or very fine granularity (few seconds or milliseconds).

Conclusion

The fast start-up time for Docker containers technology allows one to virtualize a single process or the execution of a bunch of applications, instead of a complete operating system. This opens up new possibilities, for example the possibility to "virtualize" distributed job executions in an HPC cluster of computers.

In this work we show that Docker containerization has a negligible impact on the execution performance of common genomic pipelines where tasks are generally very time consuming.

The minimal performance loss introduced by the Docker engine is offset by the advantages of running an analysis in a self-contained and precisely controlled runtime environment. Docker makes it easy to precisely prototype an environment, maintain all its variations over time and rapidly reproduce any former configuration one may need to re-use. These capacities guarantee consistent results over time and across different computing platforms.

References

Altschul, SF, Gish W, Miller W, Myers EW & Lipman DJ. 1990. Basic local alignment search tool. J Mol Biol. (1990) Oct 5; 215:403-410. PMID: 2231712

Boettiger C. 2015. An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, Special Issue on Repeatability and Sharing of Experimental Artifacts. 49(1), 71-79. doi:10.1145/2723872.2723882

Di Tommaso P, et al. 2014. Nextflow: A novel tool for highly scalable computational pipelines. Available at http://dx.doi.org/10.6084/m9.figshare.1254958

Felter W, Ferreira A, Rajamony R, Rubio J. 2014. An Updated Performance Comparison of Virtual Machines and Linux Contain. IBM Research. Available at http://ibm.co/V55Otq (accessed 1 Jun 2015)

Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL. 2013. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biol. 2013 Apr 25;14(4):R36. doi: 10.1186/gb-2013-14-4-r36.

Langmead B, Salzberg SL. 2012. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012 Mar 4; 9(4): 357–359. doi: 10.1038/nmeth.1923

Li H, et al. 2009. The Sequence Alignment/Map format and SAMtools. Bioinformatics 25 (16): 2078-2079. doi: 10.1093/bioinformatics/btp352

Li H, Durbin R. 2009. Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics (2009) 25 (14): 1754-1760. doi: 10.1093/bioinformatics/btp324

Notredame C, Higgins DG, Heringa J. 2000. T-Coffee: A novel method for fast and accurate multiple sequence alignment. J Mol Biol. (2000) Sep 8;302(1):205-17. doi:10.1006/jmbi.2000.4042

Slater GSC, Birney E. 2005. Automated generation of heuristics for biological sequence comparison. BMC Bioinformatics (2005) 6:31 doi:10.1186/1471-2105-6-31

Mack SJ, et al. 2015. Minimum Information for Reporting Next Generation Sequence Genotyping (MIRING): Guidelines for Reporting HLA and KIR Genotyping via Next Generation Sequencing. Available at http://biorxiv.org/content/early/2015/02/16/015230 (accessed 1 Jun 2015)

Trapnell C, et al. 2010. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nature Biotechnology 28, 511–515 doi:10.1038/nbt.1621

Warren RL, Sutton GG, Jones SJM, Holt RA. 2007. Assembling millions of short DNA sequences using SSAKE. Bioinformatics (2007) 23 (4): 500-501. doi: 10.1093/bioinformatics/btl629