Benchmark Report: Univa Grid Engine, Nextflow, and Docker for running Genomic Analysis Workflows

Summary of testing by the Centre for Genomic Regulation (CRG) utilizing new virtualization technology

Benchmark Report: Univa Grid Engine, Nextflow, and Docker for running Genomic Analysis Workflows

Summary of testing by The Centre for Genomic Regulation utilizing new virtualization technology

Authors: Paolo Di Tommaso, Arnau Bria Ramirez, Emilio Palumbo, Cedric Notredame, Daniel Gruber

ABSTRACT

Computing environments frequently change in an effort to increase efficiency. Docker has emerged recently as a new type of virtualization technology that allows an organization to create a self-contained runtime environment. CRG tested the capabilities of Docker for the deployment of scientific data analysis pipelines on distributed clusters by incorporating Univa Grid Engine resource manager and CRG's Nextflow toolkit. The ability to virtualize a single process or the execution of a bunch of applications in a Docker container resulted in reduced configuration and deployment problems and produced an increase in task run times and ease of replication.

ABOUT CRG

The Centre for Genomic Regulation (CRG) is an international biomedical research institute of excellence, created in December 2000. It is a non-profit foundation funded by the Catalan Government through the Departments of Economy & Knowledge and Health, the Spanish Ministry of Economy and Competitiveness, the "la Caixa" Banking Foundation, and includes the participation of Pompeu Fabra University. The mission of the CRG is to discover and advance knowledge for the benefit of society, public health and economic prosperity.

The SIT Department is responsible for the administration of the HPC service at CRG. The computing facility consists of a 1,500 core cluster.

ABOUT UNIVA

Univa is the leading innovator of workload management solutions that optimize throughput and performance of applications, containers and services. Univa manages workloads automatically by maximizing shared resources and enabling enterprises to scale compute resources across on-premise, hybrid and cloud infrastructures. Univa's solutions help hundreds of companies to manage thousands of applications and run billions of tasks every day to obtain actionable insights and achieve faster time-to-results. Univa is headquartered in Chicago, USA, with offices in Canada and Germany. Contact us at www.univa.com.

INTRODUCTION

Running scientific workflows (aka pipelines) for discovering new information on genomic sequences is a standard operation at CRG. Due to their scientific nature, the results are required to be reproducible. Reproducibility, however, can be an issue when multiple third party software components are used to calculate intermediate results. Among other things, proper versioning is especially critical, considering that different releases of the same software can sometimes produce dramatically different results. At the same time, the complex publication process that can last over a year often involves the capacity to precisely re-generate older results, as a proof of principle, or to evaluate the effect of a precise parameter.

Running jobs in a Docker container combined with Univa Grid Engine resulted in minimal performance loss of execution time as well as in simplified and optimized Docker image deployments.

Maintaining such a production chain, with controlled versions, within a standard production environment is impossible in practice. This issue is precisely the one addressed by this white paper: the precise and unambiguous prototyping of a pipeline and the setting up of a technology solution that makes it simple and possible to re-generate such pipelines, with minimum maintenance overhead. This paper addresses both the issues of efficiency and reproducibility. We show how efficiency can be improved by incorporating tool-chains and Linux libraries within the application workflow while making Univa Grid Engine aware of this process to optimally schedule jobs to the compute nodes in which the required tool chains are already cached. This goal can be achieved using Univa Grid Engine as a standard scheduler for job submission, resource selection, and job execution.

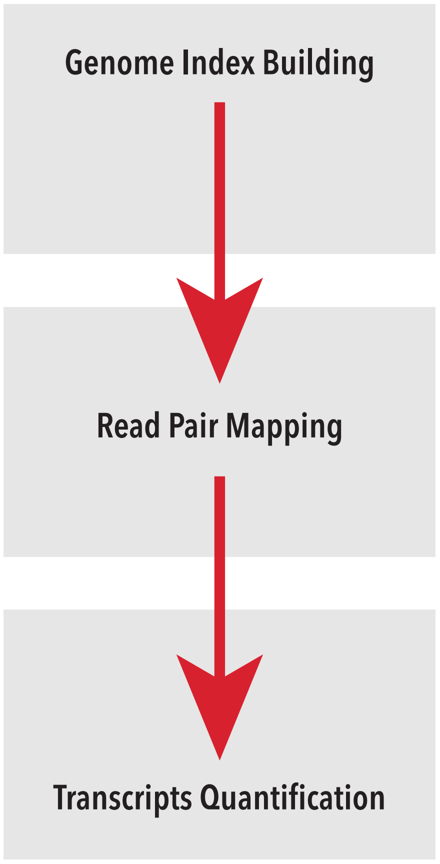
We also show how the reproducibility problem of Genomic analysis workflows can be efficiently addressed using the newly emerging container technology available for the Linux operating system. We discuss configuration options for Univa Grid Engine that allows scheduling the workflows optimally on the compute clusters to reduce the software image download overhead. Finally, we detail how a tight integration between workflows and Univa Grid Engine can improve the overall application runtime and use resource isolation for minimizing inter-job influence.

THE GENOMIC SEQUENCING WORKFLOW

The computation activity of the Comparative Bioinformatics research group at CRG mainly consists of different types of analyses carried out on Next Generation Sequencing data. The workflow is composed of computational tasks operating on short reads — character snippets coming from DNA/RNA sequencing machines — or specific data formats coming from other processing tasks. The most common tasks include:

- Reference assembly using short reads
- Alignment of short reads to a reference sequence
- Multiple sequence alignment
- Gene models building
- Gene and isoform expression quantification
- Computation of exon inclusion ratios and splicing indices

All of these tasks are computationally intensive and involve a wide range of hardware requirements in terms of consumable resources (RAM and disk memory, CPU). They require specific software tools which are optimized for the assigned task. The programs are in constant development and updated versions are released on a regular basis. Different versions of the same software can potentially produce different results, which in turn can change the whole outcome of a data analysis.



DATA CENTER MANAGEMENT USING UNIVA GRID ENGINE

Univa Grid Engine is a feature rich, highly scalable and adaptable distributed resource management system. It provides facilities for submitting, queueing, starting, and controlling jobs. The scheduler can be configured to guarantee fair usage of resources based on various metrics. An extended resource quota system can limit resource usage. Job limitations can be applied on the compute nodes, like assigning jobs to certain compute cores, in order to isolate workloads from each other. This isolation minimizes interference between different jobs scheduled to the same compute node.

CRG uses Univa Grid Engine to manage and optimize workloads executed in their distributed data center. The compute cluster consists of 150 heterogeneous compute nodes:

124 Proliant BL460c Gen8 (16 CPUs and 128 GB of RAM)
12 BladeCenter HS22 -[7870YJT]- (8 CPUs and 96 GB of RAM)
1 IBM 3850 M2 / x3950 M2 -[7233WC6]- (48 CPUs and 512 GB of RAM)
1 System x3850 X5 -[71453RG]- (24 CPUs and 256 GB of RAM)
10 PowerEdge C6145 (64 CPUs and 512 GB of RAM)

The Proliant nodes are connected using a 10Gb network and the rest of the nodes use 1Gb network. The compute cluster is connected to a 2.8PB NAS Storage accessed using NFS v3 protocol. The NAS Storage is divided in to:

- **EMC Isilon providing 1.9PB of disk space for user data and shared software (with snapshots and data replication)**
- **DDN SFA1000 providing 800TB of disk space for user data (with snapshots but no data replication)**

CRG is using several advanced Univa Grid Engine features to help fulfill the requirements of a stable cluster that uses available compute resources optimally. Many user applications create internal threads and / or processes. In order to prevent processing tasks interfering with applications from other users running on the same compute nodes, Univa Grid Engine's Control Group (cgroup) integration is used. It allows CRG to configure and automatically set job isolation parameters for any job started by Univa Grid Engine. The most important cgroup features are cpuset creation and main memory limitation.

The cluster is a shared resource and is used simultaneously by many users, hence a fair-share scheduling policy is configured on a subset (partition) of the cluster. This guarantees a fair usage of resources by, and for, all users. The fair-share configuration respects historical usage of CPU, memory, and I/O. Backfilling is also used in the cluster and is enabled automatically in order to exploit reserved resources for smaller jobs to avoid conflicts with existing resource reservations. In the latest versions of Univa Grid Engine, the reserved resources can be displayed easily by the owner of a job, and the backfilling behavior can be controlled. For example backfilling can be allowed only for jobs that request hard runtime limits (which are guaranteed by Univa Grid Engine), while backfilling can be disallowed for jobs that provide a run-time estimation to the Univa Grid Engine scheduler.

Univa Grid Engine job arrays are often used because they greatly reduce the amount of internal resources required in the Univa Grid Engine scheduler for a large amount of similar jobs. Using job arrays can considerably reduce the submission, administration and job tracking overhead.

Computational resources are not free at CRG therefore the detailed accounting and billing features of Univa Grid Engine are used in order to calculate usage by job owners.

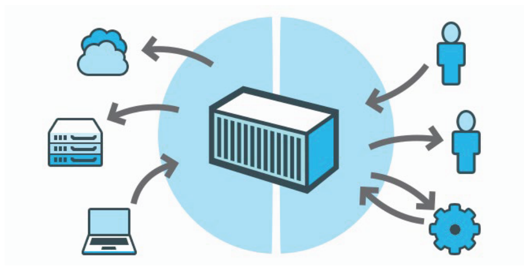
ABOUT THE DIFFICULTIES OF APPLICATION WORKFLOWS AND THEIR DEPENDENCIES

Recent progress in high throughput data production — mostly sequencing — has made biologists increasingly dependent on complex analysis pipelines. These pipelines consist of scientific workflow of dependent data processing tasks. The technology produces a prodigious stream of data that can only be processed in a timely manner by parallelizing and distributing the workload in a high performance and distributed data center.

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.

These pipelines usually rely on a combination of several pieces of third party research software. This software is normally difficult to install and configure as it may depend on outdated or conflicting operating system libraries, and will need to be updated frequently due to the short-lived nature of research software.

For all these reasons, genomics pipelines can be difficult to deploy and configure, even more so in a distributed cluster of computers. Moreover, their experimental nature



can result in frequent updates that can raise serious reproducibility issues meaning the results of the workflow may vary.

USING NEXTFLOW™ AND DOCKER™ CONTAINERS WITH UNIVA® GRID ENGINE™

To cope with the increasing complexity in genomics pipelines, CRG developed Nextflow, a toolkit that simplifies the writing of parallel and distributed computational workflows in a reproducible manner. Docker is an open-source software solution that manages the creation and startup of containers under the Linux operating system using Linux kernel features like namespaces, cgroups, and chroot. Docker allows independent “containers” to run within a single Linux instance.

Nextflow allows developers to define pipeline tasks in a declarative manner, using any scripting language of their choice (BASH, Perl, Python etcetera) and to re-use existing scripts and tools. The framework takes care of handling task dependencies and parallelism in an automatic manner. It manages the submission of pipeline tasks to the

Univa Grid Engine master for execution. In addition, it integrates the support for Docker containers technology, which allows tasks to be executed transparently in an isolated, well-defined and reproducible environment.

In other words, Nextflow provides an abstraction layer between the application logic and the underlying processing system. Thus it is possible to write your pipeline once and have it running on your computer or a cluster resource manager without modifying it, making it possible to replicate the result in a predictable manner.

To be more specific, Nextflow coordinates the pipeline tasks' parallelization and submits them to Grid Engine for execution. Parallelism is managed in a "single program multiple data" fashion (SPMD), meaning the same task can be executed multiple times when multiple input data is provided. For this reason, Nextflow creates a unique working directory each time a task is executed. This guarantees that multiple outputs of parallel tasks do not overwrite each other.

Each task script is wrapped by a launcher script that defines the cluster job directives, sets the required environment variables, and stages-in the input files and stages-out the output files from the cluster node local storage.

Nextflow also takes care of virtualizing the execution of the tasks with Docker containers. In order to do this, on each node that makes up the Univa Grid Engine cluster, the Docker engine needs to be installed. In simple terms, Nextflow turns each task execution into a cluster job that runs a Docker container that will carry out the user provided task script.

Containers are managed in such a way that the task result files are created in the hosting file system such that it behaves in a completely transparent manner without requiring extra steps, or affecting the flow in the pipeline.

The Docker image required by the jobs execution is included as a computing resource in the job request directives. Notably, the image is requested as a soft resource. In doing this, the Univa Grid Engine scheduler will first try to run the job on a node where the image has already been downloaded to speed up the execution. If such a node is not available, the job is given a lower priority and it will be executed by another node which will then pull the required image from the registry at the time of the job execution.

In order to reduce the time overhead and save network bandwidth when pulling Docker images required by workload runs, a Docker private registry mirror needs to be installed in the cluster local network.

ENTERPRISE-CLASS WORKLOAD OPTIMIZATION FOR HIGH-PERFORMANCE DATA CENTERS

Univa has developed the industry's most powerful and integrated solution suite for mission-critical and dynamically shared enterprise clusters.

The ability to scale to large clusters, increase throughput workload, and integrate applications easily, makes Univa the choice for hundreds of enterprises, academic institutions and government agencies.

ADAPTING UNIVA GRID ENGINE CONFIGURATION TO BE DOCKER AWARE

The process of adapting Univa Grid Engine to be Docker aware has been possible thanks to the Univa Grid Engine resource (complex) concept and the load sensor facility. Complexes are an abstract concept for configuring and denoting resources in Univa Grid Engine. A load sensor is a script or executable binary which feeds into Univa Grid Engine the state of resources. Load sensors can be written in any scripting or programming language. They must simply follow a very simple pre-defined input and output protocol. Load sensors are managed by Univa Grid Engine execution daemons in each computing host and reports resource / complexes values.

The "docker image" complex is declared in the complex configuration as:

```
name: docker_images
shortcut: docker
type: CSTRING
relop: ==
requestable: YES
consumable: NO
default: NONE
urgency: 0
```

Each computing node is running a load sensor that reports the images available on each node. The load sensor is defined in the global Univa Grid Engine configuration:

```
/usr/local/sbin/docker_load_sen
sor.sh
```

The docker_load_sensor.sh load sensor is a bash script. It is saved locally at /usr/local/sbin/docker_load_sensor.sh in each computing nodes and the output is generated querying the local docker daemon with the command:

```
$(/usr/bin/docker
images|/bin/grep -v
'^REPOSITORY'|/bin/awk {'print
$1'} OFS=":" ORS=',')
```

The load sensor generates an output like:

```
node-hp0511.linux.crg.es:docker_i
mages:none,d.reg/nextflow/rnatoy,
```

This complex is reported for each computing node in the format:

```
# qstat -F docker_images -q
*@node-hp0511
queuename   qtype resv/used/tot.
np_load  arch         states
---------------------------------------------
short-sl65@node-hp0511.linux.c
BIP   0/22/24    0.56    lx-amd64
hl:docker_images=none,d.reg/
nextflow/rnatoy,
```

In order to request the Docker image for an application, a soft request for this resource needs to be added on the job submission (qsub) parameters. Soft requests are resource requests that the Univa Grid Engine scheduler should fulfill for a job. If this is not possible, a job can also be scheduled to a host where the soft request is not fulfilled. The key point of using soft requests is to keep the number of hosts where certain Docker images are installed low and not to distribute all Docker images on all hosts. After the job is scheduled to a host where the requested Docker image resides the startup times are minimal. When a job is scheduled to a host without the requested Docker image, then the image must be downloaded from the local repository, which causes a delay in starting up the job.

When processes are started in Docker containers one important feature of containers is that they provide mechanisms for isolating their resources requests from each other. Docker is using the cgroups Linux kernel feature for doing this. Univa Grid Engine supports Linux Control Groups (cgroups) by fencing in all processes and child processes of the job. Since Docker works in a client server way, processes started by Docker are not direct child processes of the Univa Grid Engine job. Hence the resource request needs to be forwarded to the Docker system. The following requests are passed:

Main memory limitations
Main memory limits are passed to the system by setting an environment variable containing the memory request. The environment variable is created by a Univa Grid Engine JSV script automatically out of the requested memory limit.

CPU core selection
CPU cores are selected by the Univa Grid Engine scheduler automatically when the -binding env submission request is used. With the env option Univa Grid Engine is requested not to create the cgroup, instead it just sets an environment variable (SGE_BINDING) with the cores selected by the

Univa Grid Engine scheduler. This environment variable is passed to the docker command with the --cpuset option.

BENCHMARK RESULTS

In order to assess the impact of Docker usage on the execution performance of a genomic pipeline we benchmarked Piper-nf, a genomic pipeline for the detection and mapping of long non-coding RNAs. A comparison was made running it with and without Docker along with the same dataset.

The pipeline runs on top of Nextflow, which takes care of the tasks parallelization and submits the jobs for execution to Univa Grid Engine 8.2.1. Docker 1.0 was used in this evaluation. The pipeline was run 5 times by using the same dataset with and without Docker. Each run executed around 100 jobs using up to 10 computation nodes. We observed that the average execution time without Docker was 28.6 minutes, while the average pipeline execution time, running each job in a Docker container, was 32.2 minutes. Thus, by using Docker the overall execution time increased by 12.5%.

It is worth noting that this time includes the Docker bootstrap time; the time required to potentially download the image and the latency that is added to the task execution by the virtualization layer itself. For this reason we also wanted to measure the performance of the tasks real execution time i.e. without including the Docker bootstrap time and images download time overhead.

In this case, the mean task execution time was 35.07 seconds without Docker and 36.44 seconds when running the same tasks using Docker. Thus, the time overhead added by the Docker virtualization layer to the effective task run time can be estimated to around 4% in our pipeline benchmark.

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 a high performance cluster of computers.

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 capabilities guarantee results consistency over time and across different computing platforms.

The biggest impact on execution time when using Docker is when images need to be downloaded, and deployed. Installing each Docker image on each compute node requires long pre-setup times by the administrator and it adds major pre-configuration overhead. This approach does not scale with the amount of compute nodes and images, which is expected to increase over time. Local storage resources are wasted. Univa Grid Engine is a scalable solution that places jobs on compute nodes, that already have the required images installed resulting in reduced deployment and administration effort by the administrator. At the same time, the solution prevents job starvation in cases when Docker images are not yet deployed or are deployed on minimal compute nodes. They are automatically added but only on a subset of nodes, depending on the usage profile of the application.

ABOUT THE AUTHORS

Paolo Di Tommaso is a Research Software Engineer in the Comparative Bioinformatics research group at the Centre for Genomic Regulation (CRG), in Barcelona. He has developed algorithms and software architectures for more than 20 years and his main specialities are parallel programming, HPC and cloud computing. Paolo is B.Sc. in Computer Science and M.Sc. in Bioinformatics. He is the creator and project leader of the Nextflow pipeline framework.

Arnau Bria is a System Administrator at the Centre for Genomic Regulation (CRG), in Barcelona at CRG. Before joining CRG three years ago, he worked for more than 7 years at Port ‘dInformació Científica also as System Administrator managing a 4,000 cores cluster (Torque/ MAUI). He has been working with Batch Systems and GNU/-Linux for 15 years.

Emilio Palumbo is a Bioinformatics Software Developer in the Computational Biology of RNA Processing research group at the Centre for Genomic Regulation (CRG), in Barcelona. Among other tasks, he is in charge of the main development and maintenance of the standard RNA-seq and ChIP-seq pipelines used by the Computational Biology of RNA Processing research group. He has a B.Sc in Computer Engineering and a M.Sc in Computer Science.

Cedric Notredame is the senior leader of the Comparative Bioinformatics research group at the Centre for Genomic Regulation (CRG), in Barcelona. He is PhD in Bioinformatics from the European Bioinformatics Institute. He has developed algorithms for high throughput sequence comparison for more than 20 years. He has published over 70 scientific publications in peer-review journals that have received over 8,000 citations.

Daniel Gruber is Senior Software Engineer at Univa. Before joining Univa Corporation in 2011, he worked at Sun Microsystems in the Sun Grid Engine development team. Daniel has a B.Sc. and M.Sc. degree in Information Engineering and is co-chair of the Open Grid Forum DRMAA working group.