Light-Weight Cloud-Based Virtual Computing Infrastructure for Distributed Applications and Hadoop Clusters

Vladimir Korkhov(✉), Sergey Kobyshev, Alexander Degtyarev, and Alexander Bogdanov

St. Petersburg State University,
7/9 Universitetskaya nab., St. Petersburg 199034, Russia
v.korkhov@spbu.ru

Abstract. Virtualized computing infrastructures are often used to create clusters of resources tailored to solve tasks taking into account particular requirements of these tasks. An important objective is to evaluate such requirements and request optimal amount of resources which becomes challenging for parallel tasks with intercommunication. In previous works we investigated how light-weight container-based virtualization can be used for creating virtual clusters running MPI applications. Such cluster is configured according to the requirements of particular application and allocates only necessary amount of resources from the physical infrastructure leaving space for co-allocated clusters running without conflicts or resource races. In this paper we investigate similar concepts for MapReduce applications based on Hadoop framework that use Cloudply virtualization tool to create and manage light-weight virtual Hadoop clusters on Amazon cloud resources. We investigate performance of several Hadoop benchmarks in different deployment scenarios and evaluate effects of resource sharing and limitation on application performance.

Keywords: Virtualization · Containers · Virtual cluster

1 Introduction

In this paper we explore possibilities of container-based virtual infrastructures to enable parallel and distributed applications. In previous research [1,2] we examined how flexible configuration of light-weight virtualized computing and networking resources can influence application performance and enable multi-tenancy with minimal impact of simultaneously running MPI applications on each other. Here we focus on deployment and execution of distributed data processing frameworks in virtual container-based clusters, namely we investigate the dependency of Hadoop benchmarking suite performance on resource restrictions and existence of other simultaneously running applications.

O. Gervasi et al. (Eds.): ICCSA 2017, Part V, LNCS 10408, pp. 399–411, 2017.
DOI: 10.1007/978-3-319-62404-4_29

The driving ideas for this investigation are the following:

- Enable efficient use of available resources (physical machines or VMs) by partitioning them into independent virtual clusters that can be used in parallel
- Allocate just as much resources as needed to solve particular problem
- Limit resource use by simultaneously running clusters with light-weight virtualization technologies

Platform and tools that are used for the experiments:

- Core infrastructure: Amazon cloud
- Fine-grained resource partitioning: Docker containers
- Container management: Docker Swarm
- Resource/application configuration and management: Cloudply
- Distributed computing and data-processing framework: Apache Hadoop
- Benchmarks: TestDFSIO, TeraSort benchmarking suite (TeraGen + TeraSort + TeraValidate), MRBench

Apache Hadoop [3] is a platform for building distributed computing and data processing applications that rely upon massive distributed data storage and distributed computing nodes. Hadoop file system uses data distribution and replication across hosts and racks of hosts to ensure data protection against failures and enable parallel processing of different data blocks located near different computing nodes thus minimizing overheads on sending data across the network.

There are several use-cases that might benefit from Hadoop virtualization. First, some virtualization platforms provide extra capabilities for high-availability and fault-tolerance which can be important to keep Hadoop master daemons alive. For example, such functionality in VMWare is examined in [6]. Another approach to build fault-tolerant frameworks for distributed applications with special attention to master-node fault-tolerance was presented and evaluated in [11,12], however it is not directly applicable to Hadoop clusters. Second, it might be required to deploy dynamic Hadoop clusters of particular size for particular periods of time to solve a particular problem. Such model of using resources on-demand is provided by cloud computing, and major cloud providers offer services of dynamic Hadoop cluster deployment, namely Microsoft Azure HDInsight or Amazon EMR. However, better control on resource utilization and configuration of the cluster according to target application requires involvement into such automatic services. Third, virtualization of the infrastructure helps sharing resources with other Hadoop clusters or other applications which helps better resource utilization. Even in the cloud model computing resources are issued as virtual machines of predefined capabilities which might not be fully utilized by a single application thus fine-grained resource utilization control with light-weight virtualization might be helpful. Some relevant theoretical background and analysis of approaches to build cloud middleware using message passing and scaling control along with scaling in distributed cloud application architecture are given in [13,14].

Moreover, virtualization brings new possibilities to integrate Hadoop workloads into a datacenter or cloud infrastructure:

- Elasticity allows to grow or shrink clusters as needed in order to release resources to other applications or decrease costs;
- Multi-tenancy enables several virtual clusters share a single physical cluster (or VM-based cluster) and keep high level of isolation;
- In some cases, security requires to distribute computational and data parts of Hadoop (TaskTracker and DataNode) onto separate machines; however keeping them close (e.g. within a single host or VM) increases data locality and performance.

One of the main benefits of virtualized environment is the possibility to tune capacity of every node precisely to suit application needs. In contrast, in a native environment every node has fixed characteristics, and application must be tuned to fit resources. However, some applications are designed to be deployed on small nodes and are not able to use all capabilities of powerful hosts. Another case is using a heterogeneous infrastructure, e.g. mixture of two- and four-core processor hosts; in this case it is reasonable to virtualize four-core hosts as two two-core hosts to make clusters more homogeneous.

The benefits of Hadoop virtualization will only make sense when it will not hamper the performance much. Moreover, it is even more attractive in case it helps to distribute available resources more efficiently between multiple applications or Hadoop clusters: virtualization can provide higher hardware/VM utilization by consolidating multiple Hadoop clusters and other workload on the same physical/VM cluster. In this paper we will evaluate several scenarios of running Hadoop in light-weight container clusters over Amazon cloud virtual machines with a set of benchmarks: MRBench, TestDFSIO, TeraGen, TeraSort, TeraValidate. With these tests we will quantify performance and overheads of running Hadoop in Docker container clusters, especially in case of multiple containers running within a single VM. In our setup container clusters are managed by Docker Swarm, and overall control over infrastructure and application deployment is done by a Cloudply tool. Cloudply takes blueprints of cluster setup and application deployment and with help of Docker Swarm rolls out computing nodes, configures the application, monitors the execution and workload.

The paper is structured as follows: Sect. 2 gives an overview of related work in managing virtual Hadoop clusters; Sect. 3 describes our experimental setup and tools; Sect. 4 shows experimental results for several scenarios of virtual Hadoop cluster deployment; Sect. 5 discusses the results and Sect. 6 concludes the paper.

2 Related Work

There are a number of works that look into running distributed data processing frameworks, in particular Hadoop, in virtual environments. Detailed analysis of virtualized Hadoop performance with VMWare vSphere is presented in [5]. Apache presents discussion on strengths and weaknesses of virtual Hadoop in [4].

Some container-based deployments of Hadoop and their analysis, in particular based on Docker Swarm, appeared in publications recently [7–10]. Most of such works, however, do not focus on evaluation of how we can efficiently utilize available resources by their simultaneous use with several distributed applications, in particular Hadoop clusters, which is the focus of our research.

3 Deploying Hadoop in Virtual Container-Based DCI

In this section we describe the concept of our approach and architecture of our testbed – virtual container-based distributed computing infrastructure (DCI). For the basic infrastructure in our testbed we rely upon Amazon AWS: instances of t2.large and t2.medium virtual machines. Every virtual machine runs one or several Docker containers that are managed by Docker Swarm. Actual configuration of nodes, application deployment and general setup and management is performed by the Cloudply tool (see Fig. 1).

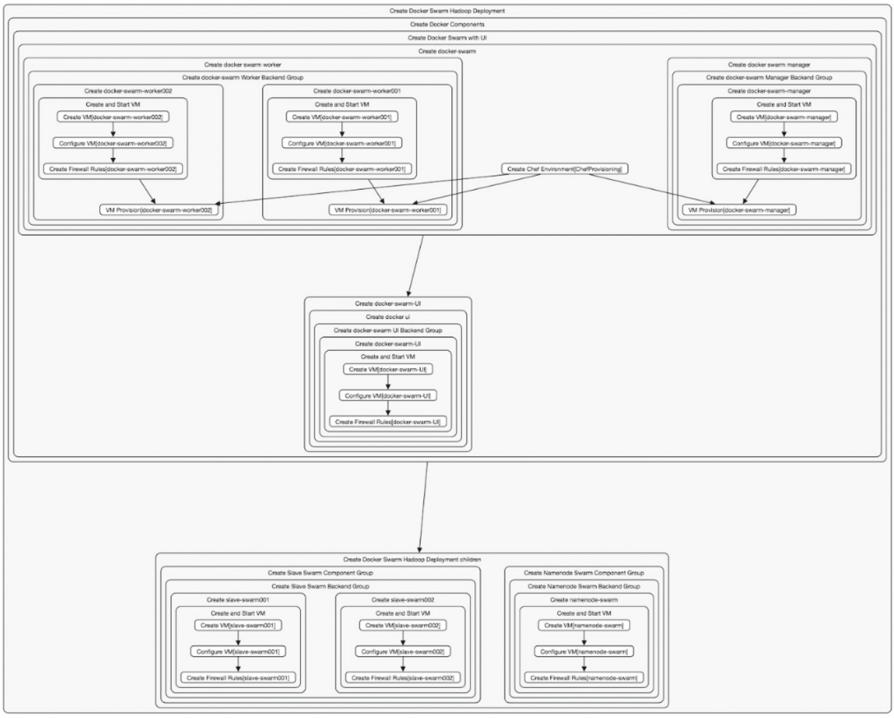

Fig. 1. Schematic view of the application deployment visualized by Cloudply

Cloudply accepts YAML-based descriptions of target infrastructure and applications: Network Blueprint to describe network structure, Security Blueprint to manage application secrets, Application Blueprint to describe application configuration:

```
ApplicationBlueprint:
 name: "DockerSwarmHadoop"
 description: "DockerSwarmHadoop description"
 applicationType: "GENERAL"
 componentGroups:
 - ComponentGroup:
       name: "Namenode Swarm Component Group"
       description: "hadoop namenode"
       producedServices:
       - Service:
           name: "hdfsUI"
           description: "HDFS web UI"
           portRange:
           - 50070
       - Service:
           name: "yarnUI"
           description: "Yarn web UI"
           portRange:
           - 8088
 - ComponentGroup:
     name: "Slave Swarm Component Group"
     description: "hadoop slave nodes"
```

Application Blueprint describes the high-level architecture of the application: components that will be used, services and their ports. In our case we will use only two component groups: one for Hadoop namenode and another one for Hadoop slave nodes. For the namenode we specify two ports: one for HDFS web interface (port 50070) and another one for YARN web interface (port 8088) Next we need to describe Infrastructure of our application. It contains all information about resources which application requires.

```
infrastructureBlueprints:
- InfrastructureBlueprint:
   name: "DockerSwarmHadoop Infra"
   description: "DockerSwarmHadoop infrastructure"
   agentName: "docker-swarm-agent"
   managedComponents:
   - DockerSwarm:
       name: "docker-swarm"
       agentName: "AWS"
       dockerAgentName: "docker-swarm-agent"
       etcd: "chef01.cloudply.org:2379"
       size:
         cpu: 2
         memory: 4096
         storage: 20
```

```
agentSettings:
  hosted_zone: "cloudply.org."
  subnet: "subnet-eff245a6"
  image: "ami-6d1c2007"
  keypair: "kobyshev.sergey"
  region: "us-east-1"
  instance_type: "t2.medium"
provisionings:
- ChefProvisioning:
    chefServer: "chef01.cloudply.org"
domain: "cloudply.org"
ssh: "kobyshev - centos"
publicIpRanges:
- "0.0.0.0/0"
workerCount: 2
servicePorts:
  docker: 12376
  docker-swarm: 12377
  dockerUI: 19000
  docker-swarm-data-plane: 4789
  docker-swarm-control-plane: 7946
```

In the infrastructure part we describe the Docker Swarm manage component. The manage component prepares templates for some often used components such as gateways, security applications, docker hosts and so on. In our case we need to create Docker Swarm which we will use for deploying a Hadoop cluster; for this we specify an agent which we will use to setup Docker Swarm. In our case we will use Amazon agent (AWS) which contains secrets to work with Amazon API. Next, we specify dockerAgentName. After uploading YAML to Cloudply engine an agent with this name will be created automatically and configured to work with Docker. For Docker Swarm we also specified etcd which contains all information about network using by Swarm. Then we specify hardware parameters for virtual machines: in our example we use t2.medium instances. In agentSettings part we specify parameters which are specific for Amazon cloud. Then we specified Chef server. Applications use Chef for provision, configuring and preparing nodes. We also can use Ansible provisioning system. Next, in Amazon we can specify our private domain, to make all nodes accessible from the same domain. Then we specified ssh key used in keypair. All ssh parameters should be created before. Then we specify Ip ranges that can access Docker Swarm resources. Than we specify the number of swarm workers. Finally, we describe all ports used by Docker Swarm.

After describing Docker Swarm we need to describe Components groups. Most fields the same. As before we describe two component groups, one for Hadoop namenode, second for Hadoop slaves. The last part of application description is deployment part, which integrates all pieces of application together. In our case application is simple and it contains only one infrastructure blueprint.

4 Evaluation of Virtual DCI

For evaluation of the deployment of Hadoop on virtual container-based cluster over Amazon cloud resources we execute a number of standard Hadoop benchmarks: TestDFSIO, TeraSort (including TeraGen and TeraValidate), MRBench.

TestDFSIO benchmark is a read and write storage throughput test for HDFS. It performs stress-testing of the distributed filesystem, discovers performance bottlenecks, in particular in networking as the write test does twice as much I/O as the read test and generates substantial networking traffic. This benchmark gives an overall estimation of how fast the cluster is in terms of I/O.

TeraSort is a benchmark that sorts a large amount of data as fast as possible. It combines testing the HDFS and MapReduce layers of a Hadoop cluster. TeraSort sorts a large number of 100-byte records. It performs significant computation, networking, and storage I/O workloads; it is often considered to be representative of real Hadoop workloads. The benchmark is divided into three parts: generation, sorting, and validation. TeraGen creates the data and is similar to TestDFSIO-write except that large computation is done during generation of random data. The map tasks write directly to HDFS, and there is no reduce phase. TeraSort does the actual sorting and writes sorted data to HDFS in a number of partitioned files. TeraValidate reads all the sorted data to verify that it is in order.

MRBench runs small jobs a number of times and checks whether small jobs are responsive. It is a complimentary benchmark to the large-scale TeraSort benchmark suite to check whether small job runs are running efficiently on the cluster. The test focuses on the MapReduce layer as its impact on the HDFS layer is very limited.

The infrastructure and Hadoop clusters are deployed according to several tests scenarios:

Scenario1: The cluster is composed a set of t2.large VMs (2 vCPUs, 8 GB RAM); every VM runs a single Docker container that uses full VM resources without constraints; Hadoop is deployed with 1 namenode and 2 worker nodes;

Scenario2: The cluster is composed a set of t2.large VMs; every VM runs a single Docker container constrained to use only 4 GB RAM; Hadoop is deployed with 1 namenode and 2 worker nodes;

Scenario3: The cluster is composed a set of t2.large VMs; every VM runs two Docker containers, each constrained to use only 4 GB RAM; two Hadoop clusters

are deployed in parallel on containers 1 namenode and 2 worker nodes; thus every VM is shared between two simultaneously running Hadoop clusters.

Figures 2, 3, 4, 5 and 6 illustrate experimental results of running the benchmarks in all 3 scenarios. The following section discusses the obtained results.

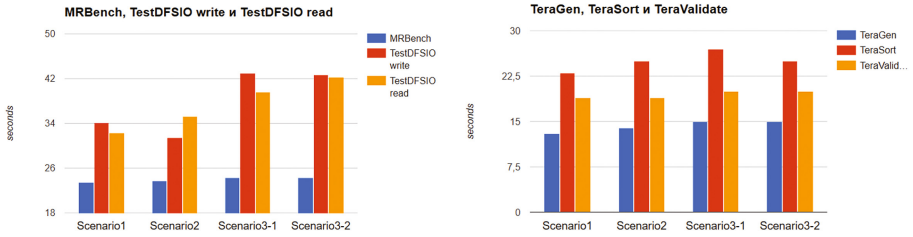

Fig. 2. Benchmark execution time for different scenarios

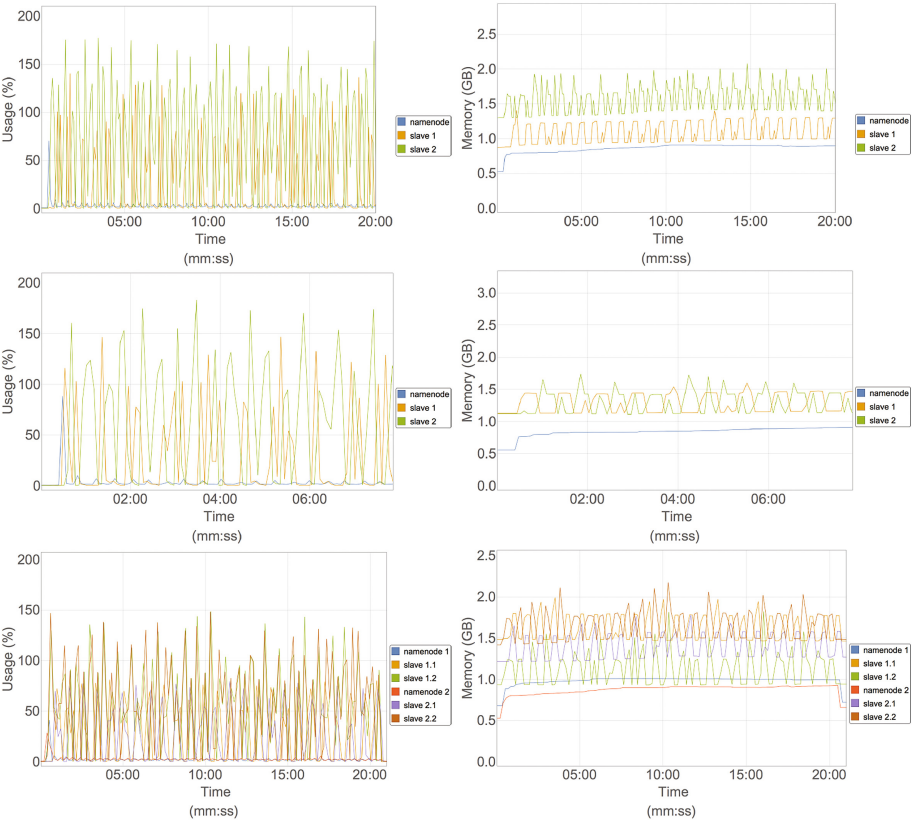

Fig. 3. MRBench: scenario 1, scenario 2, scenario 3

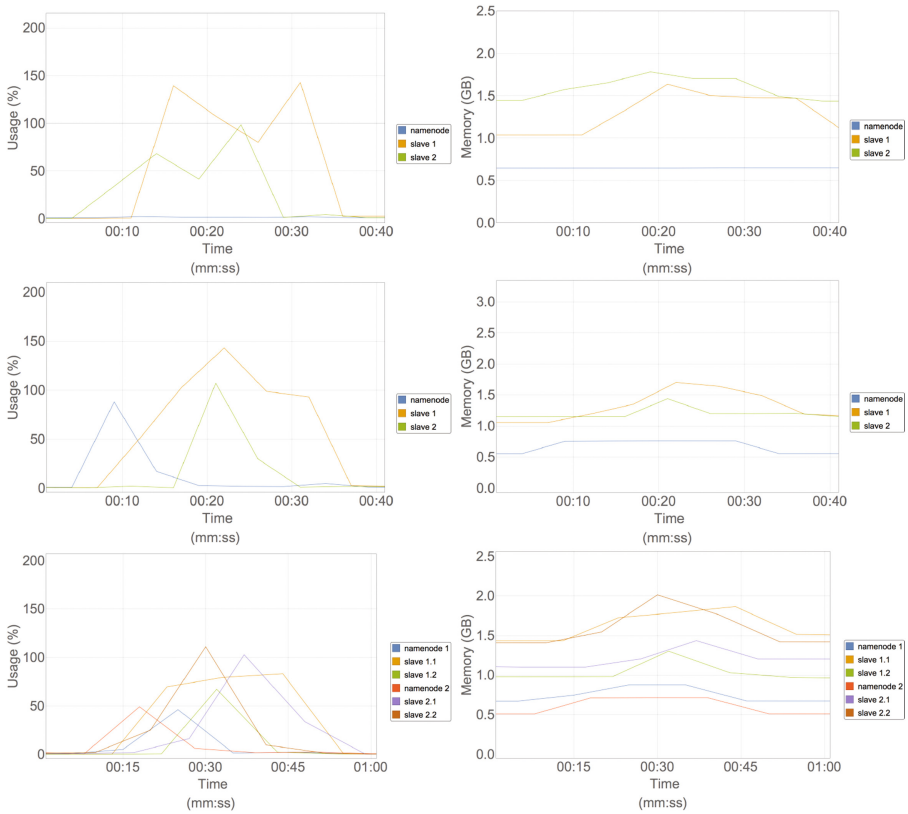

Fig. 4. Terasort: scenarios 1, 2, and 3.

5 Discussion

The goal of the experimental evaluation was to check the efficiency of using resources in distributed virtual Hadoop cluster. We compared several scenarios of infrastructure and application deployment along with running a number of standard benchmarks. Scenarios are explained in the previous section (Scenarios 3-1 and 3-2 in figures mean results for each of Hadoop clusters running in parallel).

Figure 2 shows the runtime of different benchmarks executed in different scenarios. We can see that MRBench performance does not depend on the scenario – indeed, it focuses on MapReduce without much use of HDFS, thus it relies mostly on CPU. In our setup every VM has two vCPUs, thus even in scenario 3 each container gets its own CPU. In turn, we see that the performance if TestDFSIO significantly depends on the scenario: in Scenario 3 both read and write tests perform significantly slower than in Scenarios 1 and 2 – though not

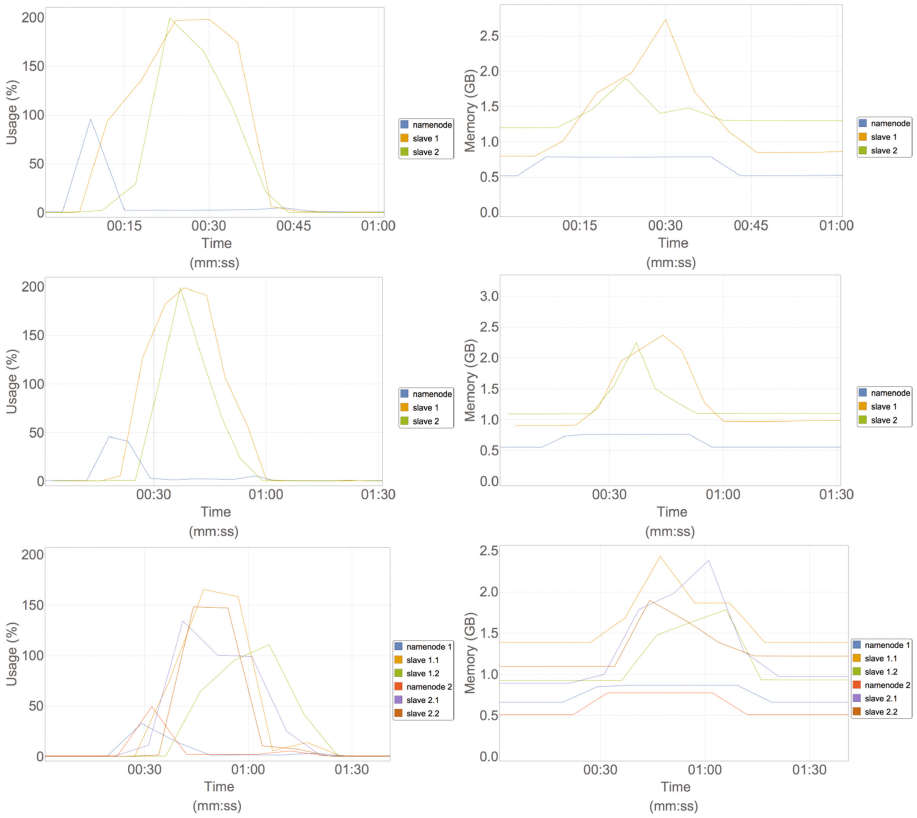

Fig. 5. TestDFSIO read: scenarios 1, 2, and 3.

twice as slow but only about 1.5 times slower, which supports the statement about efficiency of using parallel clusters. TeraSort benchmark shows only a slight decrease of performance in Scenario 3. Again, this is a good evidence that using parallel virtual clusters on a single set of resources (physical hardware or VMs) can increase efficiency of using resources and decrease costs caused by using paid cloud resources for processing workloads. In this case we managed to process twice as much as the original TeraSort workload increasing the overall processing time just for about 15%.

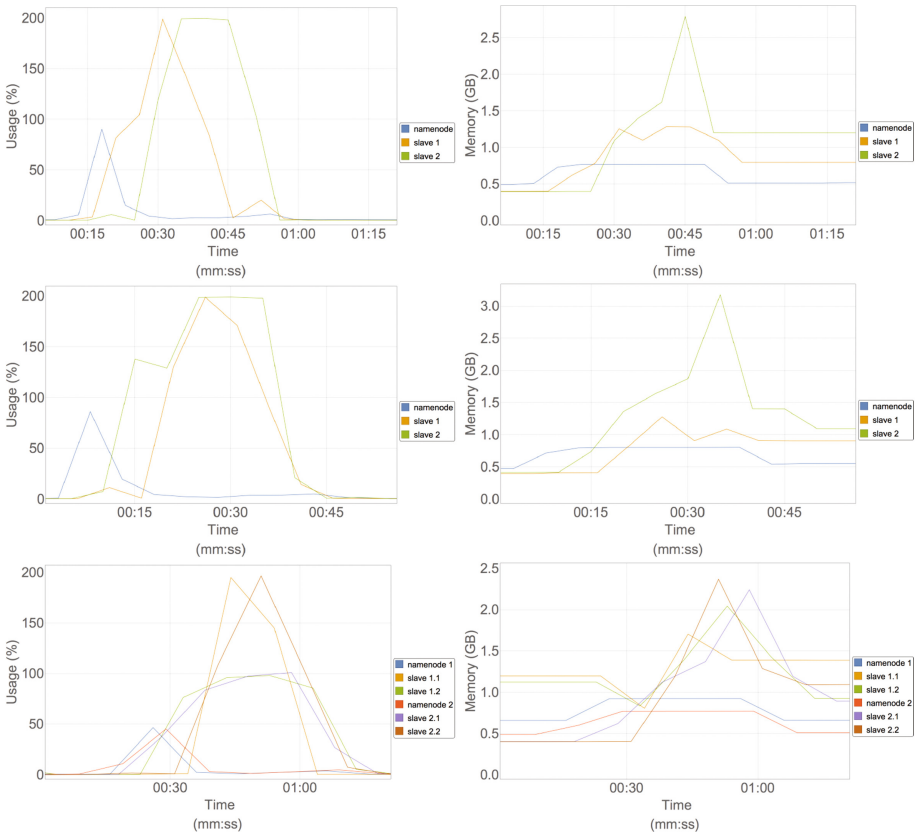

Fig. 6. TestDFSIO write: scenarios 1, 2, and 3.

6 Conclusions and Future Work

In this paper we presented Cloudply – an environment for creating light-weight virtual infrastructures on top of physical or cloud resources. With help of this tool we deployed Hadoop clusters in containers over Amazon VMs and performed experiments to check efficiency of using resources by particular Hadoop workloads. For the workloads we used well-known benchmarks: MRBench, TestDFSIO, TeraSort.

The goal of our experiments was to demonstrate that we can increase efficiency of using distributed resources – even in case of utilizing cloud resources – by simultaneous execution of light-weight virtual clusters. As long as individual requirements of applications are taken into account, we can increase the number of applications occupying a single hardware node or VM by splitting it with help of light-weight virtualization tools. These tools help to control fair resource distribution between parallel applications within a VM (Hadoop clusters in our case) and ensure no significant performance breakdowns for the applications.

We have demonstrated that for some benchmarks (e.g. MRBench) execution in parallel Hadoop clusters have not caused any performance decrease for each cluster. Other benchmarks (e.g. TeraSort, TestDFSIO) have shown slight or significant slowdown of each cluster in the scenario with parallel Hadoop clusters, however the amount of overall processed workload divided by the total wallclock time taken for processing showed good efficiency of this approach.

In future we plan to investigate the influence of infrastructure parameters (CPU share, memory, network bandwidth and latency) for the test Hadoop workloads to be able to automatically configure virtual light-weight clusters on top of available hardware of cloud resources according to particular application requirements. Potential possibility of using single cloud-based VM for several applications running in parallel without hampering each other can help to increase efficiency of using cloud resources and decrease overall costs. We also plan to look into porting new applications onto the infrastructure, in particular tools for numerical modeling of dangerous convective phenomena [15,16] and distributed visualization and rendering [17].

Acknowledgments. The research was supported by Russian Foundation for Basic Research (projects N 16-07-01111, 16-07-00886, 16-07-01113).

References

1. Korkhov, V., Kobyshev, S., Krosheninnikov, A.: Flexible configuration of application-centric virtualized computing infrastructure. In: Gervasi, O., Murgante, B., Misra, S., Gavrilova, M.L., Rocha, A.M.A.C., Torre, C., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2015. LNCS, vol. 9158, pp. 342–353. Springer, Cham (2015). doi:10.1007/978-3-319-21410-8_27
2. Korkhov, V., Kobyshev, S., Krosheninnikov, A., Degtyarev, A., Bogdanov, A.: Distributed computing infrastructure based on dynamic container clusters. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Torre, C., Taniar, D., Apduhan, B.O., Stankova, E., Wang, S. (eds.) ICCSA 2016. LNCS, vol. 9787, pp. 263–275. Springer, Cham (2016). doi:10.1007/978-3-319-42108-7_20
3. Apache Hadoop Project. http://hadoop.apache.org/
4. Apache Wiki: Virtual Hadoop. https://wiki.apache.org/hadoop/Virtual
5. Buell, J.: Virtualized Hadoop Performance with VMware vSphere 5.1. Performance Study. http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/vmware-virtualizing-apache-hadoop-white-paper.pdf
6. Buell, J.: Protecting Hadoop with VMware vSphere 5 Fault Tolerance. VMware Inc. (2012). http://www.vmware.com/resources/techresources/10301
7. Zhang, R., Li, M., Hildebrand, D.: Finding the big data sweet spot: towards automatically recommending configurations for Hadoop clusters on Docker containers. In: 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, pp. 365–368 (2015)
8. Rey, J., Cogorno, M., Nesmachnow, S., Steffenel, L.A.: Efficient prototyping of fault tolerant map-reduce applications with Docker-Hadoop. In: 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, pp. 369–376 (2015)

9. Qiao, Y., Wang, X., Fang, G., Lee, B.: Doopnet: an emulator for network performance analysis of Hadoop clusters using Docker and Mininet. In: 2016 IEEE Symposium on Computers and Communication (ISCC), Messina 2016, pp. 784–790 (2016)
10. Ivanov, T., Zicari, R., Izberovic, S., Tolle, K.: Performance evaluation of virtualized hadoop clusters. Technical report No. 2014-1, Frankfurt Big Data Lab oratory. https://arxiv.org/ftp/arxiv/papers/1411/1411.3811.pdf
11. Gankevich, I., Tipikin, Y., Korkhov, V., Gaiduchok, V., Degtyarev, A., Bogdanov, A.: Factory: master node high-availability for big data applications and beyond. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Torre, C., Taniar, D., Apduhan, B.O., Stankova, E., Wang, S. (eds.) ICCSA 2016. LNCS, vol. 9787, pp. 379–389. Springer, Cham (2016). doi:10.1007/978-3-319-42108-7_29
12. Gankevich, I., Tipikin, Y., Korkhov, V., Gaiduchok, V.: Factory: non-stop batch jobs without checkpointing. In: 2016 International Conference on High Performance Computing and Simulation, HPCS 2016, Art. no. 7568441, pp. 979–984 (2016)
13. Iakushkin, O.: Cloud middleware combining the functionalities of message passing and scaling control. In: EPJ Web of Conferences, vol. 108, Art. no. 02029 (2016). doi:10.1051/epjconf/201610802029
14. Iakushkin, O.: Intellectual scaling in a distributed cloud application architecture: a message classification algorithm, In: Proceedings of International Conference on Stability and Control Processes in Memory of V.I. Zubov, SCP 2015, art. no. 7342245, pp. 634–637 (2015). doi:10.1109/SCP.2015.7342245
15. Raba, N.O., Stankova, E.N.: On the problem of numerical modeling of dangerous convective phenomena: possibilities of real-time forecast with the help of multi-core processors. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011. LNCS, vol. 6786, pp. 633–642. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21934-4_51
16. Raba, N., Stankova, E., Ampilova, N.: On investigation of parallelization effectiveness with the help of multi-core processors. In: Proceedings of 10th International Conference on Computational Science (ICCS) 2010. Procedia Computer Science, vol. 1(1), pp. 2763–2768 (2010). doi:10.1016/j.procs.2010.04.310
17. Bogdanov, A., Ivashchenko, A., Belezeko, A., Korkhov, V., Kulabukhova, N., Khmel, D., Suslova, S., Milova, E., Smirnov, K.: Building a virtual cluster for 3D graphics applications. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Torre, C., Taniar, D., Apduhan, B.O., Stankova, E., Wang, S. (eds.) ICCSA 2016. LNCS, vol. 9787, pp. 276–291. Springer, Cham (2016). doi:10.1007/978-3-319-42108-7_21