Implementing Apache Spark Jobs Execution and Apache Spark Cluster Creation for Openstack Sahara[1]

[1]A. Aleksiyants <aleksiyantsa@gmail.com>
[1]O. Borisenko <al@somestuff.ru>
[1,2,4]D. Turdakov <turdakov@ispras.ru>
[1]A. Sher <sher-ars@ispras.ru>
[1,2,3]S. Kuznetsov <kuzloc@ispras.ru>
[1]Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia.
[2] Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
[3] Moscow Institute of Physics and Technology (State University)
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
[4] National Research University Higher School of Economics (HSE)
11 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. In this paper the problem of creating virtual clusters in clouds for big data analysis with Apache Hadoop and Apache Spark is discussed. Both clouds and MapReduce models are popular nowadays for a bunch of reasons: cheapness and efficient big data analysis respectively. For these thoughts, having an open source solution for building clusters is important. The article gives an overview on existing methods for Apache Spark cluster creation in clouds. We consider two open source cloud engines OpenStack and Eucalyptus and the most popular proprietary cloud service Amazon Web Services and examine cloud related features presented by these systems. Afterwards, we regard possible ways of creating virtual clusters for big data processing in OpenStack and describe their pros and cons. In the second part we describe in details one of these solutions that uses service Sahara. Sahara represents a cluster management system for OpenStack and it is used for setting up virtual clusters and executing MapReduce jobs. Sahara did not support contemporary versions of Apache Spark. The article introduces the results of our work that led to a Sahara modification, describes its idea and implementation details. By virtue of our modification, Sahara is able to create and use virtual clusters with contemporary versions of Apache Spark in OpenStack clouds.

Keywords: Apache Spark, Openstack, Openstack Sahara, IaaS, PaaS

DOI: 10.15514/ISPRAS-2015-27(5)-3

[1] The work is supported by the RFBR, grant No 14-07-00602

For citation: Aleksiyants A., Borisenko O., Turdakov D., Sher A., Kuznetsov S. Implementing Apache Spark Jobs Execution and Apache Spark Cluster Creation for Openstack Sahara. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 35-48 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-3.

1. Introduction

Lots of modern computational tasks are implemented using MapReduce[1] parallel paradigm. The computational process in this paradigm consists of two stages: Map and Reduce. Before task execution all the data is distributed over nodes of the cluster. On the Map stage, the master sends the executable task to the other nodes. Then every worker (slave) processes its data. The next step is called Reduce which means the master gets all results from workers and returns the final results which depend on worker's results.

Since MapReduce paradigm is very popular there are a lot of frameworks implementing this model of computations, such as Apache Hadoop, Apache Spark, Infinispan, Gridgain, Riak and other solutions that provide Big Data Processing abilities.[2][3][4][5] Moreover, such frameworks are not usually used as a standalone system; there exists huge stack of related solutions and frameworks that provide another abstraction levels such as jobs isolation, data distribution, task scheduling and so on. Lots of efforts are needed to make these levels work together. Some software companies (such as Cloudera, HortonWorks, MapR etc.) provide their own distributions that include all the components for the whole stack of related technologies.

The most popular MapReduce for today is Apache Hadoop framework. The main goal of this project is implementation of MapReduce. Hadoop uses its file system called HDFS (Hadoop Distributed File System). HDFS is a distributed, fault tolerance block storage system that is the main priority in Apache Hadoop project now.

Another example of MapReduce implementation is Apache Spark[6] that is a successor for Apache Hadoop project. This system is probably the fastest implementation of MapReduce[7][8]. In contrast to Hadoop, Spark performs most computations in main memory boosting the performance. Apache Spark supports most of the common big data stack technologies (primarily – Apache technology stack). It seems that Apache Spark stands for replacement of Apache Hadoop since it's much faster and it was designed to fix Apache Hadoop issues.

As mentioned before, nowadays a software engineer should know not only how to develop applications for Spark but also how to configure this framework with all the needed related technologies for his particular task. It seems to be a problem since configuration part is not really related to developing process.

At the beginning of our project, there were Spark deployment applications but none of them were able to deploy Spark in cloud environments [9][10] with full flexibility of related technology stack. There are different definition of cloud computing but we will use the following: "Cloud is a parallel and distributed

computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLA) established through negotiation between the service provider and consumers."[11]

There are 3 most common types of cloud services provided nowadays:

1. Infrastructure as a Service (IaaS) provides access to the whole virtual machine. A client is able to use all functionality and memory given to the machine. In other words, a client maintains the operating system and software of each particular machine himself.
2. Platform as a Service (PaaS). This service provides a platform allowing clients to run and manage certain types of applications supported by the provider. Client can deploy his applications and he doesn't really concern about virtual machine configuration.
3. The last type is Software as a Service (SaaS). This type of service stands for client getting access to applications that are already deployed in the cloud just as a user.

The most common IaaS systems that could be used for clusters creation for big data processing are OpenStack, Amazon EC2 and Eucalyptus. Our project has the following goals:

1. Automated Apache Spark clusters construction in cloud environment.
2. Provide user an ability to choose what version of Apache Spark framework to deploy in a cloud.
3. Provide user an ability to use Apache Spark in couple with related technologies within the same virtual cluster.
4. Provide PaaS access for constructed cluster which stands for simple Spark jobs execution without any knowledge about the real cluster configuration (just click&run).
5. Provide IaaS access for constructed clusters which stands for user ability to login to any virtual machine in cluster and get full control for managing his cluster if one wants to.

The next section is a brief overview of modern IaaS systems that we have mentioned and their main capabilities.

2. Platform overview

2.1 Openstack

OpenStack is an open source cloud platform with modular pluggable architecture and multitenancy support. OpenStack consists of several services that communicate via REST API. Let's take a look at the main OpenStack services.

Nova

Nova is the main module of OpenStack. Nova is responsible for running and maintaining VMs that are controlled by hypervisor. Nova consists of independent daemons responsible for different tasks. Daemons use reliable message queue for communications. The main daemons are Scheduler and Compute. At the initialization step Scheduler decides which node should host new VM. Compute provides communication between hypervisors and central component that manages requests for VMs creations/deletions/etc. All the states changes are sent to the central component that also manages primary database. Nova also stores all the supplementary data about VMs in the cloud.

Keystone

Keystone is the second major subsystem. Keystone is responsible for user authentication and providing lists of services with their endpoints. Any interaction of services and users happens through Keystone. When a client (service or user) wants to interact with any service it sends its credentials to Keystone and if they are correct Keystone subsystem provides a token that is valid for some time span. After that the client can send requests to services with the given token.

Glance

Glance is an OS image management service. Glance can use various storage backends for images: file systems or object storage systems like Swift. The main task of the project is providing boot images for VMs in Nova.

Neutron

Neutron is a service for network management. Neutron is a rather complicated service which provides virtual network connectivity to VMs in Nova. Neutron allows users to organize multilevel networks with complicated topologies, complicated security in- and outbound rules. It provides tenants networks isolation and external network services such as floating IP pools and traffic balancing. Neutron can use various network hardware resources directly through specially designed for each hardware resource plugins.

Cinder

Cinder is a block storage service. Cinder is responsible for providing persistent block storage to VMs on-demand. These block volumes are attached to compute nodes via iSCSI protocol and then they could be used by VMs as ~~a~~ local storage volumes that don't disappear after VM shutdown.

Swift

Swift is a highly available, distributed, eventually consistent object storage [12]. Object stands for file plus unstructured user-defined metadata for each file. Swift has support for stored objects compressing. Swift provides access via REST API much common to Amazon S3 which makes them easily interchangeable.

Horizon

Horizon is a Web GUI application for OpenStack services. It works as a client for each Openstack service and aggregates the capabilities of the services in one place for common usage of all the subsystems. Horizon is based on Django framework and uses native client libraries for each Openstack project.

Heat

Heat is a configuration management tool for OpenStack. For node management Heat uses templates which represent some sort of scenarios. These templates should be written in AWS format (Amazon Web Services) or HOT (Heat Orchestration Template). It also provides its own metadata server which tries to convert Nova metadata in terms of Amazon EC2 metadata but there is no full compatibility and it provides not all of the Amazon EC2 metadata terms.

Sahara

Sahara (formerly «Savanna») provides PaaS access for most common big data tools using Heat or its own engine for VM configuration. This project will be observed later in more detailed way.

Ceilometer

Ceilometer is a resource usage monitoring system. This system is used for gathering information about using VMs. The service is mostly used in business projects for controlling VMs of clients and billing.

2.2 Eucalyptus

Yet another open source cloud platform. Like OpenStack, Eucalyptus consists of several loosely coupled components communicating over WSDL (Web Service Description Language) [13]. Let's take a look at the Eucalyptus architecture and its main components.

Node controller

Node Controller is deployed on every physical node that is intended to be used for VM hosting. It is responsible for launching, maintaining and destroying VMs. The service represents a communication point between Cluster Controller and operating systems on VMs. Also Node Controller knows all the configurations of VMs on its

host. When one wants to create a VM the Cluster Controller sends a signal to some Node Controller and after that the Node Controller creates a VM from image on the node and passes control to hypervisor of VM.

Cluster controller

This component gathers information about running VMs and manages the network. Cluster Controller could be run on any machine that is connected to the Cloud Controller and the network of Node Controllers. Cluster controller gets commands from high level controllers and transmits them to specific Node Controllers. Gathered cluster data sends back to Cloud Controller.

Virtual network overlay

This component represents the network for Eucalyptus components. All VMs should be interconnected and at least one of them should have a connection to the Internet. At the start VMs interfaces are attached to software Ethernet bridge of a physical machine. Then VMs get IP and MAC addresses. VMs can communicate freely inside their cluster but if it is needed to send a message to another node then Virtual network overlay works as a router.

Storage controller (Walrus)

Walrus is a storage with Amazon S3 compatible API. It is used for storing user data and VM images.

Cloud controller

Cloud Controller is an entry point to the system for users. Cloud Controller is responsible for resource provisioning and monitoring, managing user and system data, GUI and authentication.
Eucalyptus does not have any tools for deploying Hadoop or Spark on clusters[14]. There are articles about Hadoop performance on Eucalyptus clusters but tuning and configuring were done by scripts[15].

2.3 Amazon Elastic Cloud (EC2)

Amazon EC2 is the most popular proprietary cloud platform. EC2 uses hypervisor Xen for creating VMs. At the same time EC2 has a lot of options for configuring. This platform has a MapReduce service called Amazon Elastic MapReduce (Amazon EMR) for big data processing which can deploy and run Spark as PaaS. This service can create clusters with different extensions like Hive, Pig, Impala and others. Since this project is proprietary we can use it just as it is provided without any control.

2.4 Conclusions

We have observed the most popular modern IaaS systems that are suitable for big data processing clusters creation: Eucalyptus, Amazon EC2 and OpenStack. Amazon EC2 is a proprietary platform with its own closed-source MapReduce service. Eucalyptus platform has no special support for creating big data processing clusters at all, unlike Openstack, which has Sahara project. As a result, we decided to concentrate on the latter. In the rest of this paper we describe existing Apache Spark clusters deployment existing approaches, our proposed solution and its implementation.

3. Apache Spark clusters deployment existing approaches

At the start of our project, OpenStack had a subsystem called Heat[16]. One of the goals of Heat project was to make OpenStack behave similar to Amazon EC2. This would have made possible to use Amazon cluster deploying scripts (bundled with Spark sources) for OpenStack. In the last-year work[17] we have managed to adapt it for Openstack environment but it's still not flexible, doesn't provide an ability to use other components of big data processing tools they need to use specially prepared images for each Apache Spark release.

At the present time there are three ways of building Apache Spark clusters.

3.1 Manual configuration

The first way is to configure all the components manually which means that we need to deploy all the needed components separately and for each VM we use. In the minimal configuration that means that we should install JVM, Scala, Apache Spark on each node; manually set connectivity and roles in that cluster and run all the nodes hoping that no errors were made during configuration. If we need more than just Apache Spark, we should do all the steps for each separate component.

NB: some helper tools such as Cloudera manager exist for that task, but that still means that you should setup all the base requirements for Cloudera manager on each VM in your cluster and then manually provision the cluster with Cloudera web-interface which is still complicated and long process.

3.2 Amazon EC2 IaaS approach

We have already mentioned the PaaS for big data processing that Amazon provides (EMR). Nevertheless there is another approach which is used by Apache Spark developers team and which we have used earlier [17].

This approach stands for deploying clusters in Amazon EC2[18][19] (or using similar adapted scripts in Openstack). In this case one should use pre-configured images with installed components and configuring scripts. To support HDFS, the images should use two types of storage devices for storing persistent and temporary data.

The script executes the following steps:

1. The launching process is initiated from Amazon EC2 API. This process launches VMs with chosen OS from image based on Red Hat Enterprise Linux 5.3.
2. The script sets up security groups via API.
3. The script waits for 5 minutes.
4. The script gets nodes IP addresses via API.
5. Configuration files are created for all nodes.
6. A script is run on the master node over ssh. This script downloads configuration templates from repositories.
7. The master node sends a file containing master and worker addresses over ssh to other nodes.
8. The master node creates worker's configuration files from templates.
9. The script sends created configuration files to workers.
10. The master node runs HDFS and Apache Spark.

After these steps the cluster are ready to work.

Drawbacks for this approach are the following:

1. Pre-configured images for Amazon EC2 are private. We used to prepare images for Openstack based on CentOS but it takes a lot of time and Apache Spark releases appear too fast.
2. Our scripts need a lot of tune-up for particular Openstack environment: we should provide networks ids for internal and external networks, external IP address allocation method, we should know what networking system uses particular Openstack installation (it could be Quantum/Neutron or legacy Nova networking), we should implement separate adaptors for communications for each version of API.
3. This approach is not extendable and doesn't allow using other than Apache HDFS and Apache Spark frameworks.

These actions could be managed more easily with orchestration tools such as Ansible/Salt/Chef/Puppet but the main problems of this approach would remain the same.

3.3 Openstack Sahara approach

The third option is using Sahara project in OpenStack. The main goals of Sahara are creating clusters for big data processing in OpenStack, deploying MapReduce and running jobs on clusters.[20] Before our project succeeded, Sahara could create clusters with only vanilla Apache Spark 1.0 which was already obsolete at that moment and there was no choice of other frameworks for such clusters. Also it provided access to created clusters in IaaS terms only.

At the present time Amazon EC2 lets users to create clusters with Apache Hadoop or Apache Spark automatically. But this project is not open source and does not satisfy us.

Considering the last two approaches, the most attractive is direct integration with OpenStack because of Sahara existence.

At the start of the project all these solutions either did not satisfy our requirements or needed a lot of supplementary work during every cluster launch. Therefore there was no open source system that would create clusters with current Apache Spark version from different distributors.

We decided to implement the missing functionality in Openstack Sahara project.

4. Proposed solution

Sahara works with different distributions for big data. Spark support in Sahara was limited to creation of clusters with vanilla Spark 1.0 only with basic support for running jobs via Sahara REST API with custom methods for Spark.

At the same time Sahara could create clusters with other MapReduce frameworks like Hortonworks, Cloudera and MapR. These distributors frequently update versions for distributions components including Spark. Since Sahara tries to keep up with versions of these frameworks it made sense to add Spark support as a general adapter for all the distributions and to make one specific implementation as an example for others. We have chosen to implement Spark support for Cloudera and this choice is quite random – it could be done in the same way for the others.

MapReduce implementation is defined by plugin system in Sahara. A client can create clusters using quite simple interface. Moreover, it is possible to add and remove cluster nodes online. You can see module interaction in Figure 1.

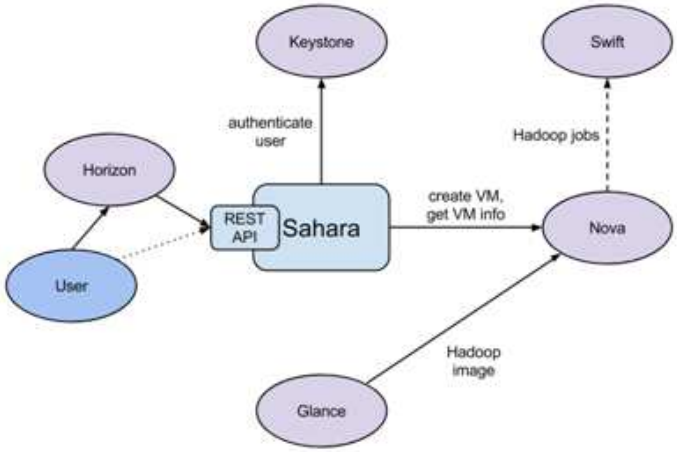

Figure 1

Cluster creation happens as follows:

1. Choose Hadoop version.
2. Choose image. Any image with cloud-init could be used.

3. Configure cluster and nodes. During this step, master and worker nodes are defined by choosing running processes for nodes.
    a. Create Node Group Template. Node group stands for group of VMs that use the same set of frameworks and tools onboard. You can assign roles to Node Groups such as HDFS Namenode, HDFS Datanode etc.
    b. Create Cluster Node Template. Cluster configuring stands for selecting Node Groups that should be included to cluster and the quantity of VMs for each Node Group.
4. Run cluster

At the start of the project you could run different MapReduce jobs (Pig, Hive, Java) through Horizon web-interface on running cluster depending on Hadoop version. Spark jobs were not on the list that time.

4.1 Sahara Architecture Overview

Figure 2 shows Openstack Sahara architecture. We have marked with colors the components we have modified to achieve our goals.

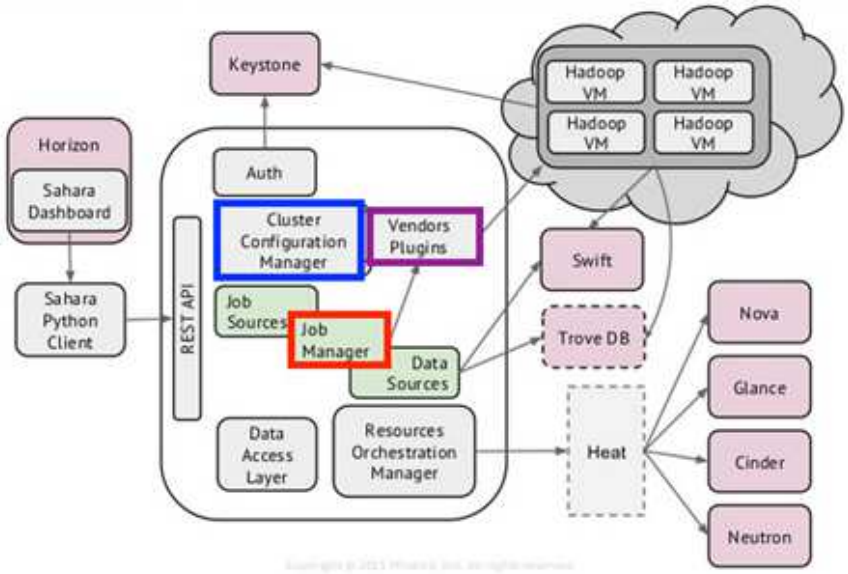

Figure 2

1. Auth component is responsible for client authentication and authorization, communicates with the OpenStack Identity service (Keystone).
2. DAL - Data Access Layer, persists internal models in DB.

3. Provisioning Engine is a component responsible for communication with the Openstack Compute (Nova), Orchestration (Heat), Block Storage (Cinder) and Image (Glance) services.
4. Vendor Plugins is pluggable mechanism responsible for configuring and launching data processing frameworks on provisioned VMs. Existing management solutions like Apache Ambari and Cloudera Management Console could be utilized for that purpose as well.
5. EDP - Elastic Data Processing is responsible for scheduling and managing data processing jobs on clusters provisioned by Sahara.
6. REST API - exposes Sahara functionality via REST HTTP interface.
7. Python Sahara Client - like other OpenStack components, Sahara has its own Python client
8. Sahara pages - a GUI for the Sahara is located in the OpenStack Dashboard (horizon).[21]

Sahara gets requests over REST API with web framework Flask. As mentioned before, support for different MapReduce distributions is implemented via plugins. Sahara uses SQLAlchemy ORM to be able to use different RDBMS for internal data handling (such as storing jobs and their state, clusters configurations etc).

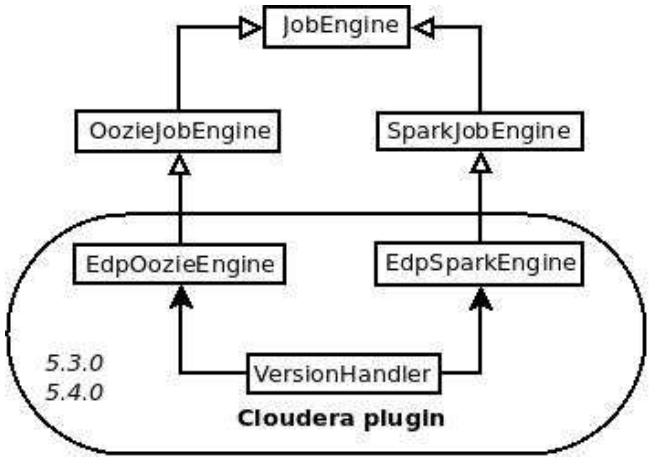

Figure 3

Sahara runs jobs on cluster through JobEngine class. This class is the base class for all types of jobs. Spark jobs are run from SparkJobEngine, which contains the most technical part. Every distribution with Spark support has its own implementation of this class. We have added another implementation for Cloudera called EdpSparkEngine. Previously Cloudera plugin could run Hadoop jobs only and they were run over Oozie.

We have made possible for Cloudera plugin to choose 'engine' class depending on the type of a job one wants to run. Now plugin's class VersionHandler returns either

Oozie or Spark engine. In addition to this we have changed common SparkJobEngine class so it could be used for all the plugins (Vanilla Spark and Cloudera were implemented during the project). This feature is supported in Cloudera 5.3.0 and 5.4.0.

Also we have changed the way Cloudera clusters are configured and made possible to create Spark clusters on Yarn using Cloudera plugin in Spark. That feature worked in standalone Cloudera version before (not in Sahara version).

Spark jobs are supposed to use storage for input and output. Originally Cloudera Spark could work with HDFS but not with Swift. We have modified classes responsible for configuring Cloudera cluster so now such clusters are able to work with Swift using Cloudera HDFS to Swift adapter.

You can check all the implementation details in Sahara official repository; all the code is available under open-source license.

6. Results

As a result of this project we have now full support for Apache Spark cluster creation in Openstack environments with support for running Spark jobs via web-interface. Also Swift object storage can be used by Spark applications transparently in Cloudera plugin for Sahara instead of HDFS. All the code is included in Openstack Liberty release and available in the official Openstack Sahara repository.

References

[1]. Jeffrey D., Sanjay G. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.
[2]. Official Hadoop homepage - http://hadoop.apache.org/
[3]. Official Infinispan homepage - http://infinispan.org/
[4]. Official Cloudera CDH Apache Hadoop homepage - http://www.cloudera.com/content/cloudera/en/productsand-services/cdh.html
[5]. Official BashoRiak homepage - http://basho.com/riak/
[6]. Official ApacheSpark homepage - http://spark.apache.org/
[7]. M. Chowdhury, M. Zaharia, I. Stoica. Performance and Scalability of Broadcast in Spark. 2010.
[8]. VMWare Serengeti page - http://www.vmware.com/hadoop/serengeti
[9]. Official Cloudera Manager homepage - http://www.cloudera.com/content/cloudera/en/products-andservices/cloudera-enterprise/cloudera-manager.html
[10]. Buyya R., Broberg J., Goscinski D. Cloud Computing: Principles and Paradigms. Wiley, 2011, 664 P.
[11]. Buyya R., Yeo C. S., Venugopal S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. CoRR, (abs/0808.3558), 2008
[12]. Swift Architectural Overview - http://docs.openstack.org/developer/swift/overview architecture.html
[13]. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language - http://www.w3.org/TR/wsdl20/

[14]. Nurmi, D. The Eucalyptus Open-Source Cloud-Computing System. Cluster Computing and the Grid. 2009. 10.1109/CCGRID.2009.93
[15]. Nilson J. Hadoop MapReduce in Eucalyptus Private Cloud. Bachelor's Thesis in Computing Science. Umea, Sweden, 2011
[16]. Official Openstack Heat homepage - https://wiki.openstack.org/wiki/Heat
[17]. O. Borisenko, D. Turdakov, S. Kuznetsov. Avtomaticheskoe sozdanie virtual'nўkh klasterov Apache Spark v oblachnoĭ srede Openstack. [Automating cluster creation and management for Apache Spark in Openstack cloud] Trudy ISP RAN [The Proceedings of ISP RAS], volume 26, issue 4, p. 33-43, 2014. (In Russian)
[18]. Official Amazon Elastic Compute Cloud (EC2) homepage - http://aws.amazon.com/ec2/
[19]. Creeger, Mache. Cloud Computing: An Overview. ACM Queue 7. 5. 2009
[20]. OpenStack Sahara roadmap - https://wiki.openstack.org/wiki/Sahara/Roadmap
[21]. OpenStack Sahara Architecture - http://docs.openstack.org/developer/sahara/architecture.html

Реализация сервиса для выполнения Apache Spark задач и создания Apache Spark кластеров на основе Openstack Sahara[1]

[1]А.В. Алексиянц <aleksiyantsa@gmail.com>
[1]О.Д. Борисенко < al@somestuff.ru >
[1,2,4]Д. Ю. Турдаков <turdakov@ispras.ru>
[1]А. В. Шер <sher-ars@ispras.ru>
[1,2,3]С. Д. Кузнецов <kuzloc@ispras.ru>
[1] Институт системного программирования РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25
[2] Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1.
[3]Московский физико-технический институт (государственный университет), 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9
[4] Национальный исследовательский университет «Высшая школа экономики» 101000, Россия, Москва, ул. Мясницкая, д.20

Аннотация. В работе рассматривается задача создания виртуальных Apache Spark и Apache Hadoop кластеров для обработки больших данных в облачных средах. Проведен обзор существующих методов создания Apache Spark кластеров. Также описывается реализованный способ создания Apache Spark кластеров и сервиса для выполнения Apache Spark задач в среде OpenStack. Предложенное решение включено в проект OpenStack Sahara и доступно начиная с релиза OpenStack Liberty.

Ключевые слова: Apache Spark, Openstack, Openstack Sahara, IaaS, PaaS
DOI: 10.15514/ISPRAS-2015-27(5)-3
Для цитирования: Алексиянц А.В., Борисенко О.Д., Турдаков Д. Ю., Шер А.В., Кузнецов С. Д. Реализация сервиса для выполнения Apache Spark задач и создания Apache Spark кластеров на основе Openstack Sahara. Труды ИСП РАН, том 27 (выпуск 5), 2015 г. стр. 35-48. DOI: 10.15514/ISPRAS-2015-27(5)-3.

[1] Работа поддержана грантом РФФИ 14-07-00602 А Исследование и разработка методов автоматизации масштабирования и разворачивания виртуальных кластеров для обработки сверхбольших объёмов данных в облачной среде Openstack

Список литературы

[1]. Jeffrey D., Sanjay G. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.
[2]. Официальная страница проекта Hadoop - http://hadoop.apache.org/
[3]. Официальная страница проекта Infinispan - http://infinispan.org/
[4]. Официальная страница Cloudera CDH Apache Hadoop - http://www.cloudera.com/content/cloudera/en/productsand-services/cdh.html
[5]. Официальная страница BashoRiak - http://basho.com/riak/
[6]. Официальная страница Apache Spark - http://spark.apache.org/
[7]. M. Chowdhury, M. Zaharia, I. Stoica. Performance and Scalability of Broadcast in Spark. 2010.
[8]. Официальная страница VMWare Serengeti - http://www.vmware.com/hadoop/serengeti
[9]. Официальная страница Cloudera Manager - http://www.cloudera.com/content/cloudera/en/products-andservices/cloudera-enterprise/cloudera-manager.html
[10]. Buyya R., Broberg J., Goscinski D. Cloud Computing: Principles and Paradigms. Wiley, 2011, 664 P.
[11]. Buyya R., Yeo C. S., Venugopal S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. CoRR, (abs/0808.3558), 2008
[12]. Обзор архитектуры Swift - http://docs.openstack.org/developer/swift/overview architecture.html
[13]. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language - http://www.w3.org/TR/wsdl20/
[14]. Nurmi, D. The Eucalyptus Open-Source Cloud-Computing System. Cluster Computing and the Grid. 2009. 10.1109/CCGRID.2009.93
[15]. Nilson J. Hadoop MapReduce in Eucalyptus Private Cloud. Bachelor's Thesis in Computing Science. Umea, Sweden, 2011
[16]. Официальная страница Openstack Heat - https://wiki.openstack.org/wiki/Heat
[17]. О. Д. Борисенко, Д. Ю. Турдаков, С. Д. Кузнецов. Автоматическое создание виртуальных кластеров Apache Spark в облачной среде OpenStack. Труды Института системного программирования РАН, том 17, 2009 г. Стр 31-50.
[18]. Официальная страница Amazon Elastic Compute Cloud (EC2) - http://aws.amazon.com/ec2/
[19]. Creeger, Mache. Cloud Computing: An Overview. ACM Queue 7. 5. 2009
[20]. OpenStack Sahara roadmap - https://wiki.openstack.org/wiki/Sahara/Roadmap
[21]. Архитектура OpenStack Sahara - http://docs.openstack.org/developer/sahara/architecture.html