УДК 004.9:004.75:004.451.82:004.738.52: 004.823

А.П. Лозинский, В.М. Симахин, А.А. Урсатьев

Моделирование технологий обработки больших данных на локальной облачной платформе

Предложена модель локальной облачной платформы с гибким перераспределением мощностей между рабочими нагрузками. Рассмотрена служба терминального доступа к рабочим столам и моделирование технологий обработки больших данных, развертывание архитектуры служб *Apache Hadoop* на виртуальном кластере, а также моделирование отдельных компонент поисковых систем.

Запропоновано модель локальної хмарної платформи з гнучким перерозподіленням потужностей між робочими навантаженнями. Розглянуто службу термінального доступу до робочих столів та моделювання технологій обробки великих даних, розгортання архітектури служб *Apache Hadoop* на віртуальному кластері, а також моделювання окремих компонент пошукових систем.

Введение. В развитие работ по созданию многоцелевого комплекса обработки научных данных и исследований [1] разработана действующая *Модель локальной облачной платформы* (далее Модель) и приведены результаты ее исследования на примерах выполнения прикладных задач. Ввиду многообразия возможных вариантов решений, рассмотрена реализация двух ресурсоемких прикладных задач. Первая относится к оптимизации организации рабочих мест. Вторая – реализует модель среды обработки больших данных. Основной вопрос состоит в разрешении проблемы разнородности решаемых задач и перераспределения между ними вычислительных ресурсов. Такая гибкость по определению свойственна облачным вычислениям [2]. Использование Модели в качестве основы построения позволяет заложить в разработку присущие облачным вычислениям свойства [3].

При существующих ограничениях доступности и безопасности облака посредством Интернета [4], использование публичных облаков на постоянной основе нецелесообразно. Требуется развертывание локальной частной облачной платформы, прототипом которой является рассматриваемая Модель. Цель исследований – выявление методов облачных вычислений, позволяющих автоматизировать, масштабировать, перераспределять, прозрачно защищать от сбоев рабочие нагрузки. В ходе исследований выявлен ряд ключевых характеристик узлов и компонентов, используемых в развертывании Модели, критичных для обеспечения указанных свойств, что будет способствовать обоснованному выбору требований к проекту.

Многоуровневая модель платформы

Существует противоречие во взглядах на облачные решения, состоящее в том, что присущие им свойства, с точки зрения пользователей, полностью абстрагированы от реальной физической аппаратуры. Это абстрактное восприятие создает впечатление, что оперативная память, дисковое пространство, сетевые соединения предоставляются будто «из ниоткуда», «из воздуха». В этом смысле термин *облако* вполне отображает восприятие пользователя. Данный подход можно наблюдать в использовании услуг таких широко известных публичных облачных платформ, как *AWS Amazon* [5], *Microsoft Azure* [6], *Google Cloud Platform* [7]. Действительно, забота о ресурсах пользователя облака сводится к своевременному пополнению финансового баланса на оплату используемых мощностей.

Вопреки восприятию пользователей, облако на самом деле развернуто на реальных физиче-

ских аппаратных узлах. Предоставляемые ресурсы входят в состав реальных аппаратных серверов, систем хранения данных и сетевых устройств. Абстрагирование достигается путем создания над физическим уровнем логической программной структуры, позволяющей реализовать свойства динамики в процессе предоставления ресурсов прикладным задачам пользователя.

Для упрощения понимания происходящих внутри Модели процессов предлагается рассматривать их в виде многоуровневой иерархической структуры (рис. 1). Основной принцип устройства структуры Модели состоит в том, что каждый нижележащий уровень служит платформой для построения вышестоящего.

Первый уровень представлен аппаратной инфраструктурой. Как показал опыт развертывания действующей Модели, ресурсоемкость, функциональность, характеристики и топология соединения узлов физического уровня критически влияют не только на ресурсоемкость вышестоящих уровней, но и обуславливает собственно возможность реализации в них таких базовых свойств облачных вычислений, как масштабируемость, производительность и гибкость перераспределения ресурсов между рабочими нагрузками.

В качестве сетевого ядра Модели используется коммутатор с портами 1 Гбит/с. Это обстоятельство демонстрирует ограничение вышестоящих уровней низкой скоростью обмена данными между физическими серверами на первом уровне. В совокупности с использованием стандартных дисковых накопителей в серверах систем виртуализации операции перераспреде-

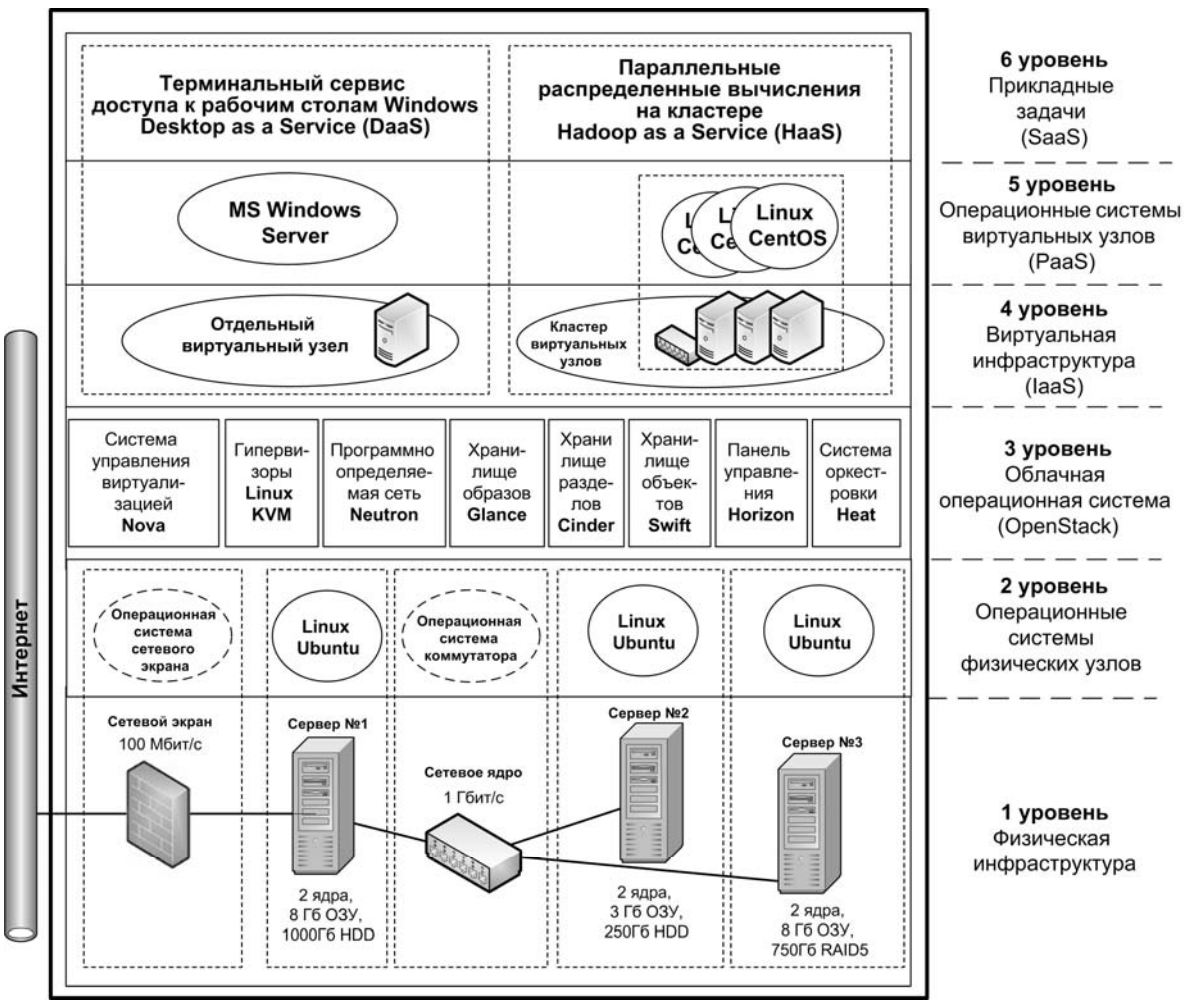

Рис. 1. Структура Модели облачной платформы

ления ресурсов длятся неприемлемо долго. К примеру, создание слепка системного раздела виртуального узла терминального сервера размером 20 Гб длится порядка 40 мин.

Второй уровень представлен платформой *операционных систем* (ОС). Под платформой подразумевается не одна, а совокупность всех ОС, установленных в физические узлы (т.е. сервера и коммутатор сетевого ядра). В рассматриваемой действующей Модели на всех физических серверных узлах развернута операционная система *Linux Ubuntu*. В узле неуправляемого коммутатора такая система отсутствует.

Отсутствие ОС в неуправляемом коммутаторе сетевого ядра ограничивает гибкость вышестоящих уровней. Встроенная операционная система управляемого коммутатора должна позволить обрабатывать заголовки фреймов сетевого трафика (например, теги виртуальных сетей) и перенастраивать порты в соответствии с формируемыми на вышестоящих уровнях структурами *программно-определяемых сетей* (*SDN*).

Третий уровень представлен облачной операционной системой *OpenStack* [8, 9], основными структурными блоками которой являются сервисы ОС второго уровня. В совокупности взаимодействия они формируют сервис-ориентированную архитектуру [10]. Назначение каждого из сервисов приведено в [11].

Третий уровень обеспечивает характерные свойства облачных вычислений. Он позволяет реализовать абстрагирование вышестоящих уровней от расположенных ниже. Состав и параметры взаимодействия сервисов обуславливают функциональные возможности вышестоящего, четвертого уровня.

Четвертый уровень – логическая инфраструктура, абстрагированная от первых трех уровней. Реализуется пользователем облака путем использования сервисов третьего уровня, в облачных вычислениях известен как *IaaS* (инфраструктура как услуга). Логическая инфраструктура кардинально отличается от физического уровня гибкостью распределения ресурсов. На этом уровне создаются логические (виртуальные) узлы, которым свойственна возможность изменения ресурсоемкости (масштабирование), а также перемещения между физическими серверами (мобильность). Таким образом, Модель позволяет изменять ресурсоемкость логического узла (количества ядер, оперативной памяти и дискового пространства) без остановки его работы.

Масштабирование происходит автоматически, с использованием сервисов нижележащего (третьего) уровня, и состоит из следующей последовательности действий: создается копия виртуального узла с целевыми параметрами ресурсоемкости. После полной синхронизации дисковых разделов и процессов в оперативной памяти существующие сетевые соединения перенаправляются на новую копию узла. По завершении переключения исходная копия уничтожается.

Так, в качестве масштабируемой рабочей нагрузки может служить сервис дистанционного доступа к рабочим столам с открытыми сессиями пользователя. В целях оценки прозрачности переключения, с точки зрения пользователя, на масштабируемом узле внутри открытой терминальной сессии запускается браузер *Mozilla*, воспроизводящий видеопоток с сайта *YouTube.com*. Масштабирование происходит прозрачно, без прерывания существующего сеанса терминального подключения и воспроизведения видеопотока в запущенном браузере. Пользователь не ощущает процесса переноса виртуального узла четвертого уровня между физическими серверами первого уровня.

Пятый уровень представлен платформой операционных систем, подобной платформе второго уровня. Отличие в том, что ОС установлены в виртуальных узлах четвертого уровня. В облачных вычислениях известен как *PaaS* (платформа как услуга).

Гибкость формирования данного уровня Модели обусловлена механизмом развертывания ОС. Узлы четвертого уровня в процессе создания снабжаются предустановленными системными разделами (образами) из хранилища образов и слепков (*Glance*). По сути, каждый такой образ представляет собой единичный файл фор-

мата *QCOW*2 [12]. В хранилище *Glance* Модели загружены образы системных разделов с предустановленными операционными системами *Linux CentOS*, *Linux Ubuntu*, *Windows Server*.

В рассматриваемой Модели (см. рис. 1) развернуты и сосуществуют две отдельные платформы пятого уровня для каждой из задач. Платформа, используемая для задачи дистанционного доступа к рабочим столам, содержит один экземпляр *Windows Server*. Платформа для задачи моделирования технологий обработки больших данных состоит из трех экземпляров *Linux CentOS*. Однако в случае необходимости платформа пятого уровня может быть разнородной, в ее состав могут входить экземпляры ОС разных версий и производителей.

Шестой уровень представлен прикладными задачами и известен как (*SaaS*). По сути, он непосредственно воспринимается пользователями как облако и является надстройкой над нижележащими уровнями, прозрачно предоставляющими прикладным задачам все, присущие облачным вычислениям свойства. Нижележащие уровни скрыты от пользователя. Для управления и контроля к первым пяти уровням Модели у пользователя нет прямого доступа.

Реализация прикладных задач

Как изложено, в рассматриваемой действующей Модели реализованы и одновременно сосуществуют две прикладные задачи. Одна из них – предоставление дистанционного доступа к рабочим столам – широко применяется в современной практике, хорошо известна и не требует особого рассмотрения в рамках настоящей статьи. Единственное, что заслуживает внимания в ее реализации, это необходимость обеспечения данного сервиса достаточным количеством ресурсов, необходимых для обслуживания пользователей. Исходя из практики использования сервиса в рамках рассматриваемой Модели, терминальная сессия одного пользователя, использующего приложения *MSOffice* и десяток закладок в браузере *Mozilla* расходует порядка одного гигабайта оперативной памяти.

Другая прикладная задача – освоение, моделирование программных сред обработки больших данных и их последующее использование в повседневной работе. Широко известная среда – де-факто стандарт технологий для работы с *Big Data* – программная платформа с открытым исходным кодом *Hadoop*. Ею обеспечивается совместное распределенное хранение и массовая параллельная обработка данных в кластере стандартных серверов.

В Модели (см. рис. 1) этот сервис представлен в слое шестого уровня как услуга *Hadoop as a Service* (*HaaS*). Кластер виртуальных узлов *Hadoop*, установка на них ОС, объединение в сеть и назначение узлам *IP*-адресов проводится облачной ОС *OpenStack* [1, 9]. Образ операционной системы *Linux CentOS* – *CentOS*-7-*x*86_64-*GenericCloud*-1611.*qcow*2 – скачан с сайта разработчика [13] и загружен в хранилище образов (*Glance*) Модели под именем *cdh-node*. Формирование инфраструктуры создаваемого кластера (четвертый уровень) выполняется средствами управления *OpenStack*, такими как *web*-консоль, *API* или командная строка [14]. Последняя обеспечивает более гибкое задание параметров *виртуальной машины* (ВМ). Создание узлов кластера выполняется командой:

```
  nova boot cdh-manager --image cdh-node\\
  --nic net-id=dc4e73ee-000c-1a5d-a179-
9cb7e2fe0431,v4-fixed-ip=10.1.1.10 \\
  --flavor 'VCPU-2 RAM-3Gb HDD-65Gb' --key-
name mahout --availability-zone c1 ,
```

где *nova* – программа-клиент управления службой *Nova*, *boot* – команда создания виртуального узла с именем *cdh-manager*, *-image cdh-manager* – параметр, указывающий имя образа из хранилища *Glance*, *-nic net-id=fc*4*c*7*fee*-000*f*-4*a*7*d*-*a*159-7*af*7*f*2*ee*0370,*v*4-*fixed-ip*=10.1.1.10 – параметр идентификатора сети подключения и *IP*-адреса сетевого порта, *-flavor 'VCPU*-2 *RAM*-3*Gb HDD*-65*Gb*' – параметр шаблона выделения ресурсов, *--key-name mahout*– частный ключ авторизации *ssh*-подключения к узлу, *--availability-zone c*1 – параметр выбора зоны размещения. Список команд и параметров программы-клиента *nova* обширен, полный список можно получить командой *nova help*.

Создание отдельного узла кластера выполняется аналогичной командой, в которой изменяются значения параметров в соответствии

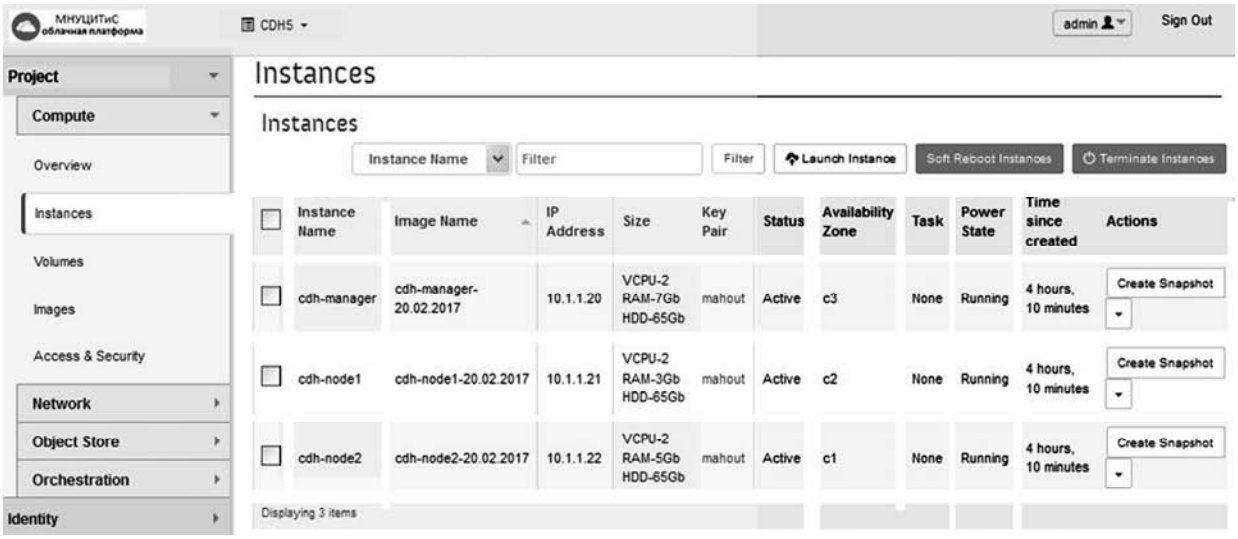

Рис. 2. *Web*-консоль панели управления *OpenStack*

с целевыми данными об имени, образе, *IP*-адресе, шаблоне выделения ресурсов и зоне размещения (физического расположения). Результат выполнения создания экземпляров узлов иллюстрирует *web*-консоль панели управления *OpenStack*. Приведенная на рис. 2 страница панели управления *Instances* (узлы), отображает перечень параметров и состояний трех узлов виртуального кластера.

OpenStack подключает созданные узлы к предоставляемому сервисом *Neutron* сегменту *SDN*-сети с *IP*-адресом 10.1.1.0/24, настраивая *IP*-адреса и сетевые маршруты, объединяя в виртуальный кластер.

Для корректного развертывания *CDH* необходимо на всех узлах кластера задать их имена. Имя своего узла *Linux CentOS* хранит в файле */etc/hostname*. На каждом узле следует задать в этом файле его уникальное имя, например *cdh-manager*. Преобразование имен других узлов кластера в *IP*-адреса обеспечивается изменением на узлах файлов */etc/hosts*. В рассматриваемом случае их содержание следует привести к следующему виду:

```
127.0.0.1 localhost
10.1.1.20 cdh-manager
10.1.1.21 cdh-node1
10.1.1.22 cdh-node2 .
```

Эта информация, по-видимому, должна доставляться в ОС ВМ в момент загрузки в виде метаданных, при наличии установленного пакета *cloud-init*. Однако эта операция может быть выполнена в командной строке подключением к каждой операционной системе[1] и внесением значений в указанные файлы.

Существует также вариант развертывания посредством сервиса оркестровки *Heat*. Развертывание осуществляется путем составления *HOT*-шаблона на языке *Heat orchestration template* [15]. Создание всего кластера автоматизируется предоставлением службе оркестровки одного шаблона. Фрагмент *HOT*-шаблона создания и настройки кластера выглядит так:

```
heat_template_version: 2013-05-23
description: Template to deploy 3 node for
Hadoop cluster
  resources:
  cdh-manager:
      type: OS::Nova::Server
      properties:
        name: cdh-manager
        key_name: mahout
        flavor: VCPU-2 RAM-3Gb HDD-65Gb
        image: cdh-manager
        networks:
          -   {   network:   production,
fixed_ip: 10.1.1.20 }
        user_data: |
          #!/bin/bash
  yum –y install ntpd
```

[1] Подключение к текстовой командной консоли ОС узлов виртуального кластера выполняется стандартным *ssh*-клиентом. Данные авторизации – имя учетной записи *centos* и частный ключ, который генерируется пользователем из *web*-консоли панели управления *Horison* (пункт меню *Compute - Access & Security - Key Pairs*).

```
        tee -a /etc/hosts <<EOF
        10.1.1.20 cdh-manager
        10.1.1.21 cdh-node1
        10.1.1.22 cdh-node2
      availability_zone: c1
cdh-node1:
    type: OS::Nova::Server
    properties:
      name: cdh-node1
  …
```

В приведенном фрагменте кроме параметров создаваемого узла *cdh-manager* задано выполнение нескольких команд (секция *user_data*) – установка пакета *yum –y install ntpd* и добавления в файл */etc*/*hosts* строк со значениями *IP*-адресов и имен узлов кластера. Таким образом, *Heat* обеспечивает создание узлов, автоматическую настройку параметров операционных систем, установку пакетов и настройку сервисов.

На базе созданного виртуального кластера (см. рис. 2) с ОС *Linux CentOS* была развернута сервис-ориентированная архитектура[2] программного продукта *Cloudera CDH* (*CDH*, *Cloudera Distribution including Apache Hadoop*). Благодаря интеграции *Hadoop* с рядом других актуальных проектов, *CDH* представляет законченную функционально развитую платформу для аналитики и управления данными.

Ряд компаний, таких как *Cloudera*[16], *Hortonworks* [17], *MapR* [18] предоставляют коммерческие услуги по поддержке инфраструктуры *Hadoop*, состоящей из *Apache Hadoop* и дополнительных ключевых проектов (*frameworks Apache Spark*, *Apache Storm, Apache Tez* и др.) с открытым исходным кодом. Эти фреймворки вносят в экосистему *Hadoop* эффективные средства оперативной обработки. Другим важным компонентом является *YARN*. Это абсолютно новая архитектура кластера *Hadoop*, изменившая способ реализации и выполнения распределенных приложений и взявшая на себя функции по управлению ресурсами и планированию заданий. *YARN* управляет кластером, как мультиарендной системой, несущей одновременно разнопрофильные и разнокалиберные нагрузки. Выигрыш от использования *YARN* получают как малые, так и большие кластеры. С появлением *YARN* подход *MapReduce*[3], используемый с *Hadoop*, превратился в распределенное приложение *MRv*2 – реализацию классического механизма *MapReduce* (*MRv*1), выполняемого поверх *YARN* [19, 20].

Программный продукт *Cloudera CDH* представляется для скачивания на сайте компании в трех вариантах [21] – ВМ *QuickStart* и программные продукты *Cloudera Manager* и *Cloudera Director*. ВМ *QuickStart* является ознакомительным продуктом, предоставляющим возможность тестировать *CDH* в рамках одной ВМ. Программный продукт *Cloudera Manager* позволяет развертывать *CDH* в составе кластера. Продукт *Cloudera Director* предназначен для развертывания *CDH* на основе публичных облаков *Amazon Web Services, Microsoft Azure* или *Google Cloud Platform*.

Развертывание кластерного решения *CDH* состоит из двух этапов. *Первый* – установка программного продукта *Cloudera Manager* на узел кластера. *Второй* – развертывание продуктов *CDH* на остальные узлы кластера посредством *web*-интерфейса *Cloudera Manager*.

На первом этапе в рассматриваемом случае для установки *Cloudera Manager* выбран узел с именем *cdh-manager* и *IP*-адресом 10.1.1.20 (см. рис. 2). Рекомендации установки приведены на сайте разработчика [22]. В результате установки на узле *cdh-manager* запускается *web*-интерфейс *Cloudera Manager*, доступный посредством браузера по адресу *http*://10.1.1.20:7180/.

Второй этап развертывания позволяет выполнять поэтапную установку продуктов *CDH* и назначение ролей узлов. Доступны три вари-

[2] Следует отметить, что используемые в интерфейсе и документации *Cloudera* термины *служба* и *роль* не соответствуют стандартам. Стандарт [*ISO*/*IEC* 18384-1:2016(*E*), *Information technology – Reference Architecture for Service Oriented Architecture* (*SOA RA*) – *URL*: *https*://*webstore.iec.ch*/*preview*/*info_isoiec*18384-1%7*Bed1*.0%7*Den.pdf*] определяет роли поставщика и потребителя простых (атомарных) служб и их композиций. В *Cloudera* атомарные службы именуются как роли, а их композиции – как сервисы.

[3] *MapReduce* – модель программирования, объединяющая формирование наборов из больших данных на узлах кластера с их обработкой, служит для организации параллельных распределенных вычислений.

анта развертывания: *Free* (ограниченный функционал), *Cloudera Enterprise Trial* (ознакомительный полнофункциональный на 30 дней), *Cloudera Enterprise* (полный функционал, требующий лицензии). Все варианты развертывания не ограничивают количество узлов кластера.

Имеется возможность установки продуктов *CDH* с помощью *packets* и *parcels* [23]. При *packets*-установке каждый узел кластера индивидуально скачивает пакеты *CDH* с сайта разработчика. В установке *parcels* программное обеспечение закачивает только *Cloudera Manager*, далее ПО дублируются на все узлы кластера. Таким образом, при большом их количестве значительно экономится внешний трафик. *Parcels*-установка позволяет также инсталлировать несколько версий *CDH* с последующей возможностью переключения между ними. На данном этапе в каждый узел кластера устанавливается полный перечень продуктов *CDH*. В состав *CDH* 5.10 входит более десятка продуктов, в том числе: *Apache Hadoop* (*Common*[4], *HDFS*, *MapReduce*, *YARN*), *HBase*, *Hive*, *Flume*, *Oozie*, *Spark*, *ZooKeeper* и др. [24]. Суммарный объем занятого дискового пространства продуктов *CDH* 5.10 и *Linux CenOS* составил 6,7 Гб (узлы *cdh-node*1 и *cdh-node*2). На узле *cdh-manager* с учетом установленного ранее *Cloudera Manager* занято 9,3 Гб.

Каждый продукт при развертывании формирует распределенную между узлами кластера композицию сервисов. Выбор и распределение сервисов (ролей) между узлами кластера, в том числе с возможностью выбора по усмотрению пользователя, выполняется на последних шагах процедуры развертывания. В рассматриваемом случае выбран запуск сервисов следующего состава: *Cloudera Management Service, HDFS, Yarn, Spark*. Ограниченность на узлах кластера ресурса оперативной памяти требует ручного распределения ролей. Композиция сервисов управления кластером *Cloudera Management Service* расходует для запуска 2,4 Гб ОЗУ. Соответственно запуск сервисов этой композиции назначается на узле *cdh-manager*, обладающем достаточным количеством оперативной памяти (см. рис. 2). Сервисы *HDFS*, *Yarn* и *Spark* распределены между узлами *cdh-node*1 и *cdh-node*2. Подробное распределение ролей и суммарный расход оперативной памяти на узлах кластера приведены в таблице.

Распределение архитектуры сервисов *CDH* по узлам кластера

Имя узла	Название сервиса	Название роли	Занимаемый объем ОЗУ
cdh-manager	*Cloudera Manager*	Ряд *java*-процессов	5 Гб
	Cloudera Management Service	*Alert Publisher*, *Event Server*, *Host Monitor*, *Reports Manager*, *Service Monitor*	
cdh-node1	*HDFS*	*Data Node*	700 Мб
	Yarn	*Node Manager*	
	Spark	*Gateway*	
cdh-node2	*HDFS*	*Balancer*, *Data Node*, *Name Node*, *Secondary Name Node*	1,6 Гб
	Yarn	*Job History Server*, *Node Manager*, *Resource Manager*	
	Spark	*Gateway*	

На случай выхода из строя узлов кластера в *HDFS* предполагается дублирование информации [19]. Для реализации этого алгоритма требуется присутствие *Data Node* на трех узлах кластера. Назначение данной роли только двум из них (см. таблицу) приводит к предупреждению в *web*-консоли о возникшем несоответствии, так как по умолчанию репликация считается успешной при создании трех экземпляров блоков данных, а имеется только два узла с *Data Node*. В рассматриваемом случае конфликт устраняется сокращением числа экземпляров репликации до двух.

В рамках парадигмы *MapReduce* на кластере *Apache Hadoop* были выполнены контрольные примеры: подсчет частоты слов в текстовом файле, более известный как *WordCount* [25], модифицированная версия *WordCount*, использующая дополнительные операции с файловой системой *HDFS* и сортировку получаемых результатов, а также поиск в файле по регулярным выражениям, например, адресов электронной почты заданного почтового сервиса. Фреймворк *Spark* был протестирован с помо-

[4] *Common* – это набор компонентов и интерфейсов для распределенных файловых систем и общего ввода–вывода.

щью той же задачи *WordCount* и базовых примеров работы с библиотекой машинного обучения *MLlib* [26]. Результаты проверок сопоставимы с данными [27].

Моделировалось создание компонент *поисковых систем* (ПС) различного назначения, так как задача наибыстрейшего поиска необходимой информации стоит очень остро. Как известно, ПС решают четыре базовые задачи: поиск в Интернете веб-страниц и извлечение из них информации, ее обработка и индексация для ускорения получения результатов и повышение релевантности, сохранение и пополнение индекса, поиск документов по индексу. ПС подразделяются на четыре типа [28]: системы, использующие поисковых роботов; системы, управляемые человеком; гибридные и мета-поисковые системы[5]. Из них гибридные ПС, более релевантные получаемым результатам, так как объединяют в себе функции систем, обеспечивающих автоматический поиск информации в Сети, и систем, управляемых человеком. При таком подходе базовый индекс поисковой системы изначально создается под воздействием ее целевой функции, затем проводится наполнение индекса *поисковым роботом* (ПР).

ПР («*web-crawler*») – программа, сканирующая веб-страницы в Интернете и сохраняющая их содержимое в виде, пригодном для работы следующих элементов ПС. Обход веб-страниц происходит автоматически и по истечении некоторого времени повторяется для сравнения и обновления данных на странице. ПР взаимодействуют с базой данных, которая содержит ссылки на веб-страницы, пути к сохраненным данным, текущий статус и другую необходимую информацию.

Модульный фреймворк *Apache Nutch* [29] – ПР, написанный на языке *Java*, совмещенный с распределенной системой хранения данных *Hadoop*, является одним из свободно распространяемых продуктов *Apache Software Foundation*. ПР поддерживает работу с большинством кодировок и языков (в том числе русский и украинский). *Apache Nutch* является надстройкой над *Hadoop* и представляет собой набор работ (*job*) вида *map* и *reduce* и правил их выполнения. Доступны две ветки фреймворка *Apache Nutch* – 1.x и 2.x. Основным отличием последней от 1.x, хранящей информацию в *HDFS*, является абстрагирование хранения информации от какого-либо конкретного базового хранилища данных с помощью фреймворка *Apache Gora* [30]. Это дает возможность реализовать гибкую модель для хранения всех данных в различных *NoSQL*-хранилищах [31].

Рассмотрен фреймворк версии 1.12. Его установка была выполнена путем сборки из исходного кода [32] при помощи утилиты *Apache Ant*. Фреймворк может работать как локально (с использованием вычислительных ресурсов одной ВМ), так и в распределенном кластере. Локальный режим наиболее приемлем для иллюстрации работы ПР, поскольку имеет упрощенный синтаксис команд. Особенности использования фреймворка *Nutch* на кластере *Hadoop* будут рассмотрены далее. Для локального запуска используется скрипт *nutch*, расположенный в папке *bin* директории установки:

```
nutch <команда> [требуемые параметры для запуска команды].
```

Например, команда *inject* для запуска в локальном режиме будет выглядеть так:

```
Nutch inject ~/localCrawl/crawldb
~/localCrawl/urls ,
```

где *~/localCrawl/crawldb* – путь к БД *crawldb*; *~/localCrawl/urls* – путь к каталогу, содержащему текстовый файл с добавляемыми ссылками.

Основные команды, поясняющие работу фреймворка *Apache Nutch* (рис. 3): *inject* – добавление в БД *crawldb URL*-адресов из источника – текстового файла; *generate* – генерация сегмента со списком *URL* из БД *crawldb*, которые необходимо обработать; *fetch* – получение исходного *HTML*-кода веб-страниц; *parse* – анализ и разбор полученных документов, извлечение текстовой

[5] Мета-поисковая система – это поисковый инструмент, посылающий запрос одновременно на несколько ПС. В собранных результатах удаляются дублированные ссылки и, в соответствии с алгоритмами, результаты объединяются/ранжируются в общем списке.

информации, метаинформации и ссылок на другие документы; *updatedb* – обновление статуса обработанных документов (скачан, недоступен, сервер не отвечает) и добавление извлеченных ссылок в БД *crawldb*.

Список всех команд скрипта, их подробное описание и требуемые параметры даны в [33, 34].

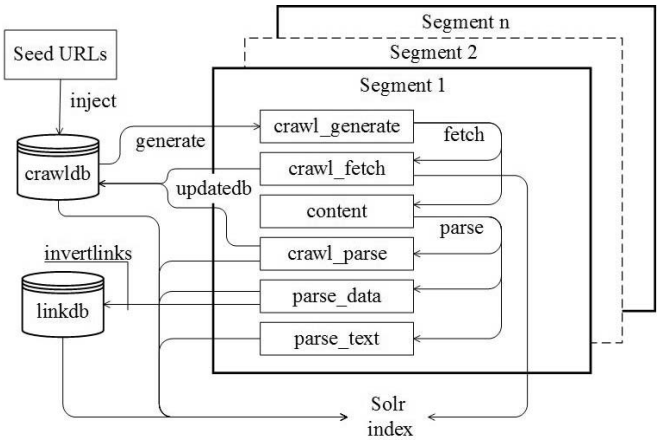

Рис. 3. Последовательность работ фреймворка *Apache Nutch*

На рис. 3 обозначены *Segments* (сегменты), каждый из которых состоит из набора ссылок, обрабатываемых как единое целое. Сегмент представляется в виде каталога, его подкаталоги создаются в результате выполнения различных этапов работы фреймворка: *crawl_generate* – *URL*-адреса для обработки командой *fetch*; *crawl_fetch* – статус выполнения. Например, будет отмечено, что доступ к серверу не удался; *content* – исходный *HTML*-код веб-страницы для каждого *URL*; *crawl_parse* – записи в формате БД *crawldb*, которые содержат исходящие ссылки обрабатываемой веб-страницы (одна запись – одна ссылка); *parse_text* – извлеченный текст для каждого *URL*; *parse_data* – заголовок страницы, список исходящих ссылок и метаинформация для каждого *URL*-адреса.

Для хранения данных в процессе работы также используются следующие структурные элементы: БД *crawldb*, которая содержит список всех *URL*-адресов и хранит состояние каждого адреса, и БД *linkdb*. Последняя представляет собой инвертированную БД ссылок, т.е. для каждого *URL* она содержит список *URL*-адресов, по которым на него ссылаются.

Подкаталоги сегментов представляют собой базу данных. Типы и форматы файлов БД *crawldb*, *linkdb* и сегментов нестандартны. Их описание доступно на странице документации [35].

Для работы в кластере *Hadoop* фреймворк запускается командой

```
Hadoop jar <jar> mainClass (-D
'http.agent.name'='<имя>') args ,
```

где *hadoop jar* – команда, используемая для запуска *.jar*-файлов, *<jar>* – путь к запускаемому *.jar* файлу, *mainClass* – класс, содержащий метод *main*, который необходимо выполнить, *–D 'http.agent.name'='<имя>'* – опциональный параметр для работы *Apache Nutch*, указывающий имя, которое будет передаваться обрабатываемому сайту, *args* – необходимые параметры для запуска *main*-метода.

В распределенной файловой системе *HDFS* был создан каталог */hadoopCrawl* с подкаталогом */urls*, содержащим текстовый файл *seed.txt*. В него был записан один *URL*-адрес. Пример команд *inject* и *fetch* для запуска на *hadoop* кластере:

```
   hadoop jar ~/apache-nutch-
1.12/build/apache-nutch-1.12.job
org.apache.nutch.crawl.Injector
/hadoopCrawl/crawldb/ /hadoopCrawl/urls

   hadoop jar ~/apache-nutch-
1.12/build/apache-nutch-1.12.job
org.apache.nutch.fetcher.Fetcher -D
'http.agent.name'='crawler'
/hadoopCrawl/segments/20170228111909 ,
```

где */crawldb* – каталог с БД *crawldb*; */segments*/20170228111909 – каталог, содержащий полученный сегмент данных. Наличие опционального параметра *http.agent.name* зависит от взаимодействия с веб-сайтом: метод *fetch* обрабатывает веб-страницы и для корректного доступа через Интернет к веб-ресурсу требует обязательного указания данного параметра, а метод *inject* выполняет изменения в БД *crawldb* и не взаимодействует с Интернетом.

Помимо создания баз данных и индексов, которые впоследствии передаются в поисковый механизм *Apache Solr* или *Apache Lucene*, фреймворк позволяет просматривать полученные результаты работы путем чтения сегмен-

тов командой *readseg* для скрипта *nutch* либо запуска *main*-метода класса *org.apache.nutch.segment.SegmentReader*. Результаты чтения можно сохранить в текстовый файл для анализа или последующей обработки.

Приведем фрагмент информации, сохраненной в сегменте, после выполнения команды *fetch*:

```
    Recno:: 0
    URL::
  https://www.osp.ru/os/2016/04/13050983/
    CrawlDatum::
    Version: 7
    Status: 33 (fetch_success)
    Fetch time: Mon Mar 06 13:53:41 EET
  2017
    Modified time: Thu Jan 01 03:00:00 EET
  1970
    Retries since fetch: 0
    Retry interval: 2592000 seconds (30 days)
    Score: 1.0
    Signature: null
    Metadata:
            _ngt_=1488801004893
            pst_=success(1), lastModified=0
            _rs_=1318
            Content-Type=text/html
            nutch.protocol.code=200 .
```

Здесь параметр *Recno*:: указывает на номер прочитанной записи; *CrawlDatum*:: определяет формат данных *CrawlDatum* для последующих параметров. Помимо этого, в параметрах присутствует *URL*-адрес, над которым выполнялась команда, время выполнения, интервал повторения выполнения для заданной записи, количество уже выполненных повторений, а также метаданные и некоторая другая информация.

Для иллюстрации результатов выполнения команды *parse* представлены текстовые данные статьи из журнала [36]. *ParseText*:: указывает, что следующие данные представляют собой извлеченный текст:

```
  Recno:: 0
  URL::
https://www.osp.ru/os/2016/04/13050983/
  ParseText::
  Ускорители инноваций: «большая семерка» ОС, версия 2017 | Открытые системы. СУБД | Издательство «Открытые системы» Вход для зарегистрированных пользователей Логин Пароль Восстановить пароль Войти Регистрация Новости События Экспертиза Подписка Computerworld LAN Директор ИС Открытые системы Windows IT Pro Мир ПК DGL Тема номера Текущий номер Current Issue Архив Об издании About Issue Реклама Открытые системы. СУБД 2016 № 04 4094 прочтения Ускорители инноваций: «большая семерка» ОС, версия 2017 Год 2017-й ознаменуется развитием технологий третьей платформы и активным освоением новых «ускорителей инноваций»: искусственного интеллекта, дополненной реальности и ряда других, появившихся на обновленном ИТ-ландшафте. 16.11.2016 Наталья Дубова Ключевые слова / keywords: Блокчейн Blockchain Большие Данные Big Data Интернет вещей Internet of Things Искусственный интеллект Artificial Intelligence Прогноз ОС Журнал «Открытые системы. СУБД» традиционно завершает год обзором технологий, которые и по мнению западных аналитиков, и по мнению редакции «сделают» год грядущий. Бизнес сегодня уже неразрывно связан с технологическими революциями — именно они определяют способность компаний и организаций к проведению цифровой трансформации, без которой успех да и просто выживание на современном рынке невозможны. Защита корпоративных и пользовательских данных на всех этапах Однажды описав третью платформу как основу для цифрового бизнеса, аналитики IDC на годы вперед определили повестку обзоров ИТ-тенденций, которые регулярно включают в себя вариации на тему Больших Данных и предсказательной аналитики, мобильных и социальных технологий, облаков и Интернета вещей.
```

Nutch столь же безукоризненно извлекает текст на украинском и других языках. Правда, это не дает основания ожидать, что при индексировании будет поддержка особенностей языка. Для украинского, русского и других аналогичных необходимо использовать автоматические средства для обеспечения морфологического анализа слов как на этапе индексирования, так и поиска информации. Чаще в ПС используют варианты анализаторов для разных языков с поддержкой *Stemming* – приведения слов к нормальной форме.

Apache Nutch поддерживает такие форматы документов как *XML*, *PDF*, *Microsoft Word* и другие, предоставляет данные для вычисления индекса поискового механизма. Возможно подключение плагинов, которые позволяют добавлять или убирать функции, настроить взаимодействие с такими продуктами *Apache*, как *Solr*, *Lucene*, *Tika* или *Gora*. Так, *Solr* представляет собой поисковый сервер с возможностями полнотекстового поиска, подсветки результатов, динамической кластеризации, интеграции с ба-

зами данных и др. Он обладает масштабируемостью и отказоустойчивостью, обеспечивая распределенное индексирование, репликацию и балансировку нагрузки.

Известны также и другие библиотеки и фреймворки, позволяющие реализовать ПР. Например, фреймворк *Scrapy* [37] для языка *Python*, *Java* библиотека *crawler4j* [38] или *Frontera* – фреймворк на *Python,* содержащий примитивы для распределения и масштабирования работ [39].

Компоненты и структура ПР варьируются в различных реализациях зависимо от его функциональности. Так, в работе создателей ПС *Google* [40] поисковый робот представляется программой, которая обходит все *URL*-адреса, предлагаемые специальным *URLserver*'ом и загружает веб-страницы на сервер хранения. Другой подход – ПР, реализующие дополнительные возможности, преследуемые определенной целью, например, вертикального поиска [41]. Примерами этого подхода служат ПР [42, 43], которые фильтруют всю сканируемую информацию и определяют коэффициент научности текста, что позволяет сузить поисковый индекс путем исключения ненаучных материалов, или предполагают создание отдельных ПР для известных наукометрических баз данных, что упрощает извлечение необходимой информации.

Для сравнения авторами использован вариант ПР, состоящий из двух модулей. Первый – *web-scraper* – извлекает текстовые данные из веб-страницы и сохраняет их для последующей индексации. Второй – *web-spider* – анализирует *HTML*-код сохраненной страницы и извлекает ссылки на другие источники. Для хранения необходимой информации ПР использует *NoSQL* БД. В нее записывается основная информация об обработанных веб-страницах: *URL*-адрес, путь к связанным сохраненным данным, статус, время последнего доступа к сайту и другое, а также хранятся ссылки, найденные с помощью *web-spider*.

Алгоритм работы модуля *web-scraper*:

- добавление всех необработанных адресов из БД в очередь;
- обработка веб-страницы с возвращением переменной, которая содержит извлеченный основной текстовый фрагмент в строковом формате;
- сохранение текстового фрагмента в файловой системе и запись информации о расположении сохраненного файла и текущего времени в БД;
- изменение статуса обрабатываемой записи в БД указывающего, что ссылка была обработана *web-scrapper'*ом;
- повторение предыдущих шагов для всех ссылок из очереди;
- Алгоритм работы модуля *web-spider*:
- добавление в очередь всех необработанных ссылок из БД;
- извлечение *HTML*-кода веб-страницы по обрабатываемому *URL*-адресу;
- определение границ извлеченного текстового фрагмента в *HTML*-коде; запоминаются позиции начала и конца фрагмента в коде;
- извлечение *URL*-адресов из *HTML*-тегов *<a href=\>* во фрагменте, определенном на предыдущем этапе;
- исключение неподходящих адресов, например ссылающихся на изображения, видео- и аудиофайлы, архивы;
- добавление проверенных *URL*-адресов в БД со статусом необработанных;
- изменение статуса обработанной записи БД.

Рассмотрим используемые при реализации алгоритмов средства. Авторами была использована комбинация двух *Java*-библиотек – *boilerpipe* и *jsoup*. Работа библиотеки *boilerpipe* [44] показана на рис. 4. Библиотека предоставляет методы для определения основного, превалирующего фрагмента текста путем удаления так называемых «шаблонов» (*boilerplate*) веб-страниц, определяемых методом неглубокого анализа текста (*Shallow text features*) [45]. Метод основан на квантитативной лингвистике (*quantitative linguistics*) и учитывает такие показатели, как средняя длина слова, средняя длина предложения, абсолютное количество слов текста (в данном случае – веб-страницы), абсолютное и отно-

сительное положение фрагмента текста в теле веб-страницы, коэффициент частоты заглавных букв и разделительных знаков. С их помощью из *HTML*-кода выделяется раздел веб-страницы, содержащий основную текстовую информацию. Библиотека располагает конкретными стратегиями для общих задач, а также может быть расширена для индивидуальных задач. В качестве входных данных используется лишь *URL*-адрес. Выходные данные доступны в *TXT*-формате, *HTML* или *JSON*. Демонстрационная веб-версия [46] позволяет проверить и сравнить результаты извлечения.

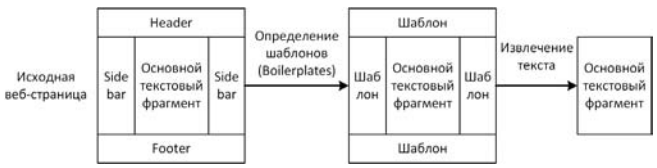

Рис. 4. Схема извлечения текста с помощью библиотеки *boilerpipe*

Библиотека предусматривает четыре стратегии извлечения, которые различаются количеством удаляемых *шаблонных* элементов.

• *ArticleExtractor* – извлекает всю основную текстовую информацию, настроен на более точную работу со статьями в новостных ресурсах;

• *DefaultExtractor* – извлекает всю основную текстовую информацию, но возможно худшее качество определения, чем у *ArticleExtractor*;

• *LargestContentExtractor* – работает по принципу *DefaultExtractor*, но извлекает именно наибольший фрагмент, хорошо работает с текстами, состоящими из одного основного блока;

• *KeepEverythingExtractor* – извлекает все содержимое веб-страницы, полезно для определения ошибок *XML*-парсинга.

При использовании библиотеки код с процедурой извлечения текста приведен ниже:

```
public String getDefaultText (ULR url) {
    String output = “”;
    try {
        output =
ArticleExtractor.INSTANCE.getText(url);
    } catch (BoilerpipeProcessingException
e) {
        e.printStackTrace();
    }
    return output;
}
```

Применение библиотеки позволило вместо сохранения веб-страницы целиком (сравнить с фрагментом текста, полученного при работе фреймворка *Apache Natch*), извлечь и сохранить лишь необходимую для последующего анализа информацию:

```
#URL:
https://www.osp.ru/os/2016/04/13050983/
#Page_title: Ускорители инноваций: «большая семерка» ОС, версия 2017 | Открытые системы. СУБД | Издательство «Открытые системы»
Прогноз ОС
Журнал «Открытые системы. СУБД» традиционно завершает год обзором технологий, которые и по мнению западных аналитиков, и по мнению редакции «сделают» год грядущий. Бизнес сегодня уже неразрывно связан с технологическими революциями — именно они определяют способность компаний и организаций к проведению цифровой трансформации, без которой успех да и просто выживание на современном рынке невозможны.
Защита корпоративных и пользовательских данных на всех этапах
Однажды описав третью платформу как основу для цифрового бизнеса, аналитики IDC на годы вперед определили повестку обзоров ИТ-тенденций, которые регулярно включают в себя вариации на тему Больших Данных и предсказательной аналитики, мобильных и социальных технологий, облаков и Интернета вещей.
```

Работа с *HTML*-кодом веб-страницы выполняется при помощи библиотеки *jsoup* [47]. Данная библиотека – одна из лучших в классе парсеров, что подтверждается использованием в крупных проектах, таких как *OpenRefine* [48] или *Cucumber* [49]. Методы, предоставляемые библиотекой, позволяют извлечь необходимые данные, используя *DOM*, *CSS* и методы в стиле *jQuery*.

Таким образом проверенные решения ПР, могут быть использованы в разработке предметно-ориентированных ПС при соответствующей их адаптации компонентов под конкретные требования задач пользователя. Моделирование задачи на кластере позволило выявить особенности настройки и применения программной среды с открытым исходным кодом – инструментария проектирования распределенных систем для хранения и обработки больших данных на базе проекта *Apache Hadoop*, эффективность которого проявляется только при решении трудоемких задач, требующих одно-

временно и большого объема вычислений, и хранения больших объемов данных.

Заключение. Рассмотренные платформы являются частными случаями возможных целевых решений. Благодаря технологии виртуализации Модель предоставляет гибкий механизм моделирования и развертывания платформ широкого спектра архитектур, целевого назначения и производства. Она предлагается в качестве общего решения для развертывания частных-локальных облачных решений на предприятиях.

1. *Гриценко В.И.*, *Урсатьев А.А.*, *Лозинский А.П.* Облачные технологии Многоцелевых комплексов геораспределенных систем // УСиМ, 2015,№ 2, С. 4–17.
2. *Гриценко В.И*, *Урсатьев А.А*. *Cloud Computing* и облачная модель предоставления ИТ-услуг // КВТ, 2013, **171**, С. 5–19.
3. *ISO/IEC* 17788:2014 Information technology – Cloud computing – Overview and vocabulary – impl. 15.10. 2014. – Brussels: European Committee for Electrotechnical Standardization, 2014, 16 p.
4. *Cloud* Computing Synopsis and Recommendations. Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800–146 / L. Badger, T. Grance, R. Patt-Corner et al. – URL: http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf
5. *AWS* Amazon. – URL: https://aws.amazon.com/ru/
6. *Microsoft* Azure. – URL: https://azure.microsoft.com/ru-ru/
7. *Google* Cloud Platform. – URL: http://cloud.google.com/?hl=ru
8. *Openstack* open source cloud computing software. – URL: https://www.openstack.org/
9. *Лозінський А.П.* Огляд функціональних можливостей програмного забезпечення хмарної платформи OpenstackIcehouse // Наук. записки, 2014, № 122, С. 84–93. – URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM = 2&I21DBN = UJRN&P21DBN = UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name= PDF/Nzped_2014_122_13.pdf
10. *ISO/IEC* 18384-1:2016(E), Information technology – Reference Architecture for Service Oriented Architecture (SOA RA). – URL: https://webstore.iec.ch/preview/info_isoiec18384-1%7Bed1.0%7Den.pdf
11. *What* is Open Stack? – URL: http://www.openstack.org/software/
12. *Формат* дискового образа программы QEMU. – https://ru.wikipedia.org/wiki/Qcow2
13. *Linux* CentOS images download. – URL: http://cloud.centos. org/centos/7/images/
14. *Open Stack* Docs. – URL: https://docs.openstack.org/
15. *Heat* Orchestration Template (HOT) Guide – URL: http://docs.openstack.org/ developer/heat/template_guide/hot_guide.html
16. *Cloudera* Enterprise Solution. – URL: http://www.cloudera.com/
17. *What* is Apache Hadoop? – URL: http://hortonworks.com/hadoop/
18. *Hadoop&BigData*. – URL: https://www.mapr.com/products/apache-hadoop
19. *Урсатьев* А.А. Некоторые программные среды аналитики больших данных // УСиМ, 2016, № 3, С. 29–42.
20. *Урсатьев* А.А. Некоторые программные среды аналитики больших данных и машинного обучения // Там же, 2016, № 5, С. 63–75.
21. *Cloudera* Enterprise Download. – URL: http://www.cloudera.com/downloads.html
22. *Installing* Cloudera Manager and CDH. – URL: http://www.cloudera.com/documentation/enterprise/latest/topics/ installation.html
23. *Hadoop*, Ч. 1: развертывание кластера. – URL: https://habrahabr.ru/company/selectel/blog/198534/
24. *CDH* 5 Packaging and Tarball Information. – URL: https://www.cloudera.com/documentation/enterprise/release-notes/topics/cdh_vd_cdh_package_tarball.html
25. *Apache* Hadoop 2.7.2 – MapReduce Tutorial. – URL: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduTutorial.html#Example:_WordCount_v.2.0
26. *Machine* Learning Library (MLlib) Programming Guide – Spark 1.2.0. Documentation. – https://spark.apache.org/docs/1.2.0/mllib-guide.html
27. *Глибовец А.Н.*, *Дмитрук Я.О.* Эффективность применения языков программирования в фреймворке *Apache Hadoop* с использованием *MapReduce* // УСиМ, 2016, № 5, С. 84–92.
28. *Tarakeswar K.*, *Kavitha D*. Search Engines: A Study // J. of Comp. Appl. (JCA) ISSN: 0974-1925, **IV**, Issue 1, 2011. – URL: http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.300.4896&rep=rep1&type=pdf
29. *Apache* Nutch™ – URL: https://nutch.apache.org
30. *Apache* Gora™ – URL: https://gora.apache.org/
31. *Front* Page – Nutch Wiki. – URL: https://wiki.apache.org/nutch/FrontPage#What_is_Apache_Nutch.3F
32. *Nutch* Tutorial – Nutch Wiki. – URL: https://wiki.apache.org/nutch/NutchTutorial
33. *Nutch* Command Line Options of bin/nutch – Nutch Wiki. – URL: https://wiki.apache.org/nutch/CommandLineOptions
34. *Laboratorio* de Investigación Aplicada – Отчет по Apache Nutch. – URL: http://nitec.wikidot.com/
35. *NutchFileFormats* – Nutch Wiki. – https://wiki.apache.org/nutch/NutchFileFormats
36. *Дубова Н*. Ускорители инноваций: «большая семерка» // Открытые системы, 2016, № 4. – https://www.osp.ru/os/2016/04/13050983

37. *Scrapy* | A Fast and Powerful Scraping and Web Crawling Framework. – https://scrapy.org
38. *Github* – yasserg/crawler4j. – https://github.com/yasserg/crawler4j
39. *Github* – scrapinghub/frontera. – https://github.com/scrapinghub/frontera
40. *Brin S.*, *Page L.* The Anatomy of a Large-Scale Hypertextual Web Search Engine // Comp. Networks and ISDN Syst., April 1, 1998, **30** Issue 1–7, P. 107–117. – http://dx.doi.org/ 10.1016/S0169-7552(98)00110-X
41. *Croft W.B.*, *Metzler D.*, *Strohman T.* Search Engines Information Retrieval in Practice, 2015, 518 p.
42. *Глибовець А.М.*, *Шабінський А.С.*, *Ольшевський Р.Я.* Побудова пошукового робота україномовних наукових матеріалів // Наук. праці, **130**, Т. 143. – http://lib.chdu.edu.ua/pdf/naukpraci/computer/2010/143-130-13.pdf
43. *Коляда А.С.*, *Гогунский В.Д.* Автоматизация извлечения информации из наукометрических баз данных // Управління розвитком складних систем, 2013, **16**, С. 96–99. – http://journals.uran.ua/urss/artocle/view/38927/35236
44. *Github* – kohlschutter/boilerpipe. – https://github.com/kohlschutter/boilerpipe
45. *Kohlschütter C.*, *Fankhauser P.*, *Nejdl W.* Boilerplate Detection using Shallow Text Features – http://www.l3s.de/~kohlschuetter/publications/wsdm187-kohlschuetter.pdf
46. *Boilerpipe* Web API. – https://boilerpipe-web.appspot.com
47. jsoup: Java HTML Parser. – https://jsoup.org
48. *OpenRefine*. – http://openrefine.org
49. Cucumber Simple, human collaboration – https://cucumber.io

Поступила 19.05.2017
Тел. для справок: +38 044 526-4159 (Киев)
E-mail: *aleksei@irtc.org.ua*

UDC 004.9:004.75:004.451.82:004.738.52: 004.823

A.P. Lozinskiy[1], V.M. Simakhin[2], A.A. Oursatyev[3]

Technologies Modeling for Processing Large Data on the Local Cloud Platform

[1] Junior Research Associate, International Research and Training Centre of Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03680, Ukraine, loza@irtc.org.ua
[2] Engineer, International Research and Training Centre of Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03680, Ukraine, sima@irtc.org.ua
[3] PhD in Techn. Sciences, Leading Research Associate, International Research and Training Centre of Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03680, Ukraine, aleksei@irtc.org.ua

Introduction. The implementation of the operational local cloud platform model is considered which provide two services at the SaaS level. The first one relates to the issue of optimizing the organization of workplaces in the organization. The second one implements the model of the large data processing environment. The main issue is to solve the problem of combining heterogeneous tasks within a common environment and redistributing computing resources between them.

Purpose. The purpose of this article is to build a model of a multi-purpose local cloud platform with a flexible redistribution of power between workloads and to consider the usages of two applications for the model: the service of terminal access to desktops and the Hadoop software platform for big data analysis. In addition, the modeling of the search engine element – the search robot – as the task of big data analysis.

Methods. Methods of modeling and abstraction were used.

Results. A functioning model of the local cloud platform, which provides a flexible mechanism for modeling and deploying platforms of a wide range of architectures, purpose and production, is proposed. The possibilities of using existing solutions of search robots are analyzed and our own experimental development is created. Both variants of search robots provide the necessary information in a suitable form for the work of consequential elements of search engines.

Conclusion. The model is proposed as a general solution for the deployment of private local clouds in enterprises. One of the possible applications implemented based on a platform for big data analytics is the creation of a search engine.