КОМПЬЮТЕРНЫЕ НАУКИ

УДК 004.75

ПРОГРАММНЫЕ СРЕДСТВА СИСТЕМЫ УЧЕТА ИСПОЛЬЗОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ УКРАИНСКОЙ ГРИД-ИНФРАСТРУКТУРЫ

КЛИМЕНКО К.А., ЛЕВЧУК Л.Г., ПРИСТАВКА А.С., КУРОВ А.А.

Описываются программные средства, применённые при создании системы сбора и публикации информации об использовании вычислительных ресурсов грид-сети Украины NGI_UA. Система содержит актуальные данные о работе практически всех зарегистрированных в Европейской грид-инфраструктуре (EGI) сайтов NGI_UA. Созданный информационный ресурс характеризуется надлежащим уровнем наглядности и обеспечивает простой доступ к информации о работе вычислительных комплексов, с предоставлением визуальных отчетов в виде динамических графиков и диаграмм.

Ключевые слова: грид-инфраструктура, виртуальная организация, портал учета, средства визуализации.

Keywords: Grid infrastructure, virtual organization, accounting portal, visualization tools.

1. Введение

Одним из важнейших результатов бурного развития телекоммуникаций, произошедшего за последние два десятилетия, стало построение вычислительных инфраструктур принципиально нового типа с объединением территориально удаленных компьютерных ресурсов. Впечатляющим примером такого объединения являются созданные специализированные грид-инфраструктуры для решения задач в различных областях науки и техники, требующих трудоемких вычислений. Получение доступа к грид-ресурсу обеспечивается соответствующим пользовательским сертификатом, который должен быть зарегистрирован в так называемой виртуальной организации (ВО), предоставляющей данный ресурс с (возможно) установленным и сконфигурированным специализированным прикладным программным обеспечением. Созданные грид-инфраструктуры зачастую чрезвычайно широко географически распределены и даже глобальны. Ярким примером такой глобальной сети является всемирная грид-инфраструктура для обработки данных с Большого адронного коллайдера (LHC) – ‘Worldwide LHC Grid’ (WLCG) [1]. В настоящее время WLCG объединяет более 170 вычислительных центров в 42 странах и предоставляет ресурсы для хранения и обработки данных, получаемых в экспериментах на LHC в Европейской лаборатории ядерных исследований (ЦЕРН). При этом привлекаются вычислительные и информационные ресурсы национальных и международных грид-инфраструктур, таких как Европейская грид-инициатива (EGI) [2] и панамериканская академическая грид-сеть ‘Open Science Grid’ (OSG) [3].

EGI является федерацией, объединяющей национальные «грид-инициативы» (NGI) европейских стран, в том числе и Украины (NGI_UA). Украинский национальный грид (УНГ) при этом представлен более чем десятью ресурсными центрами научных и образовательных организаций Украины. Украинская грид-инфраструктура предоставляет свободный доступ к вычислительным ресурсам и ресурсам хранения данных в рамках, соответствующих ВО. Интеграция УНГ в европейскую грид-инфраструктуру обеспечивает возможность проведения совместных исследований с международными исследовательскими организациями и предоставляет доступ к европейским (EGI) и прочим международным (в частности, WLCG) ресурсам. В перспективе возможен переход к построению объединенной инфраструктуры для распределенных вычислений, состоящей из созданной грид-инфраструктуры и облачной инфраструктуры с возможностью обеспечения современного интерфейса доступа пользователей к различным информационно-вычислительным ресурсам. В частности, могут рассматриваться пути подключения NGI_UA к объединению ‘Partnership for Advanced Computing in Europe’ (PRACE) [4].

Инфраструктура NGI_UA в своей основе была создана за последние ~ 10 лет и находится в постоянном развитии. Любая грид-инфраструктура (в том числе и NGI_UA) функционирует с помощью так называемых базовых грид-сервисов. Таковыми службами являются, в частности, сертификационный (CA) и операционный центры (ОС), а также сервисы поддержки ВО (VOMS и VOMRS). Одними из важных базовых служб являются системы учета (‘accounting‘) использования грид-ресурсов (СУИГР). Такие системы могут работать (собирать, обрабатывать и публиковать необходимые данные) в рамках как крупных международных грид-сообществ, так и отдельных их участников. В частности, учет использования грид-ресурсов для EGI производиться с помощью портала [5], а для OSG роль СУИГР в настоящее время выполняет система GRACC [6]. Свои собственные СУИГР созданы и успешно функционируют также в ряде развитых национальных грид-инфраструктур (например, HLRmon [7] в Италии). СУИГР предоставляют ин-

формацию об использовании вычислительных ресурсов и позволяют точно оценить состояние соответствующих инфраструктур. Порталы СУИГР включают в себя отчеты (например, оценки активности тех или иных ресурсных центров, ВО и отдельных пользователей) и содержат обширный интерактивный инструментарий для анализа этой информации.

Актуальность разработки СУИГР для национальной грид-сети Украины обусловлена быстрым развитием и расширением последней. Возникает необходимость определенной автоматизации при выполнении сравнительного анализа активности сайтов, входящих в УНГ. Основным и определяющим фактором такой активности является именно интенсивность использования вычислительных ресурсов. Первый прототип СУИГР для УНГ был создан в ННЦ ХФТИ в 2012 г. и работал в рамках специализированного вычислительного комплекса для обработки данных с LHC [8]. Изначально работа системы была основана на использовании мета-пакета 'Accounting Processor for Event Logs' (APEL) [9], входящего в состав промежуточного программного обеспечения (ППО) грид-платформы gLite [10], а в настоящее время этот пакет является частью сервиса 'Computing Element' (CE) ППО UMD [11]. Несмотря на успешную работу в течение нескольких лет созданного прототипа СУИГР, он в то же время имел ряд недостатков. Одним из таких ограничивающих факторов явилась зависимость от обновлений ППО на грид-сайтах УНГ с возможной перестройкой баз данных MySQL, на которых основана работа службы APEL. В связи с этим в 2015 году система была полностью переконфигурирована. В разработанной новой версии СУИГР источником информации о работе грид-сайтов УНГ является портал 'EGI Accounting Portal' (EGIAP) [4]. В настоящее время информация, представленная на портале СУИГР УНГ, охватывает практически все активные сайты NGI_UA. Созданная система является открытой и доступна по веб-адресу: https://grid-accounting.kipt.kharkov.ua.

В настоящей статье представлены детали работы СУИГР УНГ. В разделе 2 описано взаимодействие с порталом [4] для получения необходимой исходной информации. Раздел 3 содержит детали обработки и представления полученной статистической информации. Основные результаты работы кратко сформулированы в разделе 4.

2. Получение данных об использовании вычислительных ресурсов

Основным источником информации для работающей в настоящее время системы учета использования грид-ресурсов являются данные, публикуемые на портале [4]. Для ее получения формируется запрос с указанием в качестве параметров названия страны (Украина), полного названия вычислительного ресурса, категории данных, периода, за который необходимо получить данные, тип группировки данных и перечня активных ВО. Формат запроса отвечает принятому в WLCG перечню параметров, характеризующих работу вычислительных ресурсов: количество выполненных за определенный промежуток времени задач, использование процессорного времени в ненормированных ('Used_CPU hours') и нормированных ('Norm_CPU kSI2K-hours', 'Norm_CPU HEPspec06-hours') единицах. Нормировка затраченного процессорного времени производится по результатам тестирования производительности различных вычислительных систем в рамках стандарта SPEC [12] ('kSI2K' является сокращением от 'kiloSPECint2000'). Стандарт SPEC был разработан с целью обеспечить сравнительную оценку вычислительной производительности для широкого диапазона процессоров, используя рабочие нагрузки, разработанные на основе пользовательских задач. Единица 'HEPspec06' [13] определена на основе выборочного выполнения тестов производительности (для операций как с целочисленными переменными, так и с переменными с плавающей точкой) стандарта SPEC 2006 года (SPEC06) с наилучшим соответствием типу задач, характерных для физики высоких энергий.

Результат выполнения каждого запроса представлен в виде JSON-документа со строгой структурой. Для работы с данными в формате JSON применяется утилита JQ [14]. Она является «фильтром»: принимает данные на вход и формирует выходные данные. Существует множество встроенных фильтров для извлечения определенного поля, объекта, преобразования числа в строку и других стандартных задач, связанных с фильтрацией и форматированием данных. Утилита также позволяет комбинировать результаты работы фильтров в зависимости от задачи. Основным достоинством JQ является возможность реализации сложных итераций с данными в простой форме конвейера. Например, можно передавать выходные данные одного фильтра в другой фильтр или собирать выходные данные фильтра в массив. Некоторые фильтры могут формировать последовательности выходных данных, которые могут быть переданы как массив входных данных следующему

фильтру, который в свою очередь выполняет фильтрацию для каждого элемента массива. Как правило, подобные цепочки действий выполняются с помощью циклов и итераций. Данные в JQ представлены как потоки значений JSON – каждое выражение JQ выполняется для каждого значения в его входном потоке и может выводить любое количество значений в его выходной поток.

Отмеченные возможности утилиты JQ существенно упрощают процедуру сбора и первичной обработки информации, получаемой для СУИГР, предоставляя в распоряжение API, а также обеспечивая гибкость посредством пользовательских сценариев. Фрагмент скрипта, представленный на рис. 1, демонстрирует формирование строки запроса к ресурсу [4] для получения необходимой информации (в данном случае для грид-сайта 'Kharkov-KIPT-LCG2') за текущий месяц в формате JSON и реализует цикл опроса по списку поддерживаемых ВО для получения данных по всем доступным метрикам.

```
#!/bin/bash
port="***" host="***.***.***.***"
user="***" password="***" db="accounting"
site='Kharkov-KIPT-LCG2'
month=`date +%m` year=`date +%Y`

for vo in "cms" "ops" "dteam"
do
  out=""
  value=""
  for metric in "njobs" "sumcpu" "sumcpu_days" "normcpu"
   "normcpu_days" "elap_processors" "elap_processors_days"
   "normelap_processors" "normelap_processors_days" "cpueff"
  do
  value=`curl -ks
'https://accounting-support.egi.eu/custom_xml.php?
&query="'$metric'"
&option=SITE&opval="'$site'"
&sYear="'$year'"&sMonth="'$month'"
&eYear="'$year'"&eMonth="'$month'"
&yrange=VO&xrange=DATE&groupVO=custom-"'
$vo'"&localJobs= localinfrajobs&tree=TIER2'
| jq '.[0] | .["'$year'-'$month'"]'`
  out="$out, $value";
  done
  echo "$year-$month: $vo$out"
  echo "Update test_accounting"
  mysql -P $port -h $host -u $user --password=$password $db << eof
REPLACE INTO accounting (date, site, lhc_vo, year, month, njobs,
sumcpu, sumcpu_days, normcpu, normcpu_days, elap_processors,
elap_processors_days, normelap_processors,
normelap_processors_days, cpueff)
VALUES (DATE_FORMAT(now(),'%Y-\'$month\'-\'$day\'),
"$site", "$vo", "$year", "$month" `echo "$out"`);
eof
done
```

Рис. 1. Формирование строки запроса к ресурсу [4] для получения данных

Результаты выполнения скрипта записываются в базу данных для дальнейшего использования средствами визуализации ‘Highcharts’ и ‘Highstock’ [15]. Дополнительно полученные данные публикуются в информационную панель ’Dashboard’ (подробности приведены в следующем разделе).

3. Визуализация статистических данных на портале

В течении длительного времени большинство решений, связанных с визуализацией данных, основывались на принципах сервер-клиент: графики и всевозможные диаграммы формировались на серверной стороне и передавались клиенту в виде статических изображений. Самым распространённым подходом был ‘Common Gateway Interface (CGI)’ [] и продукты, основанные на ‘JpGraph’ []. Такой подход имел ряд достоинств, а именно идеально подходил для задач автоматизации, например, при формировании различных отчетов и служебных уведомлений с вложенными статическими графиками и диаграммами. Формирование подобных отчетов (сравнительно небольшого объема) характеризовалось относительной простотой, и JavaScript при этом не использовался. К недостаткам можно отнести дополнительную нагрузку на стороне сервера, ограничения, связанные с возможностями языка HTML, отсутствие интерактивных элементов взаимодействия с данными на стороне клиента.

Следующим шагом развития решений для предоставления интерактивных возможностей для веб-приложений стало использование Java-апплетов на клиентской стороне. Это позволило выполнять программы в формате байт-кода непосредственно в браузере, существенно расширив возможности взаимодействия с пользователем. Это инициировало переход от традиционного серверного подхода к решениям, в которых основная работа проводится на стороне клиента. Java-апплеты в большинстве браузеров выполняются в изолированной среде без доступа к локальным данным. Код апплета загружается с веб-сервера, и браузер либо вставляет апплет в веб-страницу, либо открывает новое окно с собственным пользовательским интерфейсом апплета. В то же время применение Java-апплетов имело определенные недостатки: необходимость установки Java-расширения (plug-in), которое не во всех браузерах на тот момент было доступно по умолчанию, а также невозможность запуска без виртуальной машины Java.

Широкое распространение JavaScript и HTML5 привело к новым подходам при разработке пользовательских веб-интерфейсов. JavaScript обычно ис-

пользуется в качестве встраиваемого языка для программного доступа к объектам приложений, а наиболее широкое применение получил как язык сценариев для придания интерактивности веб-страницам. К основным архитектурным особенностям JavaScript можно отнести динамическую типизацию и прототипное программирование. Новый стандарт HTML5 содержит элемент ‘Canvas’, часто используемый при создании интерактивных графиков с помощью JavaScript. Этот элемент объявляет область графического вывода и использует API JavaScript для пиксельного отображения линий и форм. ‘Canvas’ не имеет встроенной анимационной процедуры, поэтому для имитации анимации используются API-вызовы с временными последовательностями. Кроме того, в технологии ‘Canvas’, отсутствует поддержка обработки событий, поэтому при разработке необходимо вручную привязать обработчики событий к области графического вывода. Более прогрессивным подходом явилось внедрение масштабируемой векторной графики (SVG) и соответствующего стандарта, позволившего существенно расширить возможности графического отображения и упростить процедуру настройки атрибутов графики.

При разработке СУИГР были применены различные технологии визуализации статистический данных. В частности, использовалась библиотека ‘Highcharts’ [15], разработанная с использованием JavaScript. Ее реализация позволяет применять программные платформы различных сторонних разработчиков. В частности, эта библиотека может взаимодействовать с ‘MooTools’ [16], ‘Prototype’ [17] и ‘jQuery’ [18], что позволяет без ущерба интегрировать ее к готовым веб-приложениям и использовать ту платформу, которая лучше всего подходит для управления и разработки. ‘Highcharts’ обладает широким спектром типов визуализации и представления данных от простых графиков до сложно комбинированных диаграмм, а также совместима со многими веб-браузерами и может быть использована в большинстве современных мобильных платформ. Кроме того, эта библиотека имеет простую модель API с хорошо оформленной онлайн документацией (каждое определение содержит описание и онлайн демонстрацию для настройки).

‘Highcharts’ состоит из пяти классов (см. рис. 2): ‘Chart’, ‘Axis’, ‘Series’, ‘Point’ и ‘Renderer’, в некотором смысле связанных между собой (например, класс ‘Point’ имеет ряд свойств, наследуемых от класса ‘Series’). В то же время каждый класс имеет набор уникальных методов управления и отображения для собственного слоя. ‘Chart’ является классом самого высокого уровня, представляющим все объекты и содержащим все методы вызова графиков – например, экспорт графика в SVG или другие графические форматы, настройки размера диаграммы. Класс ‘Chart’ имеет несколько массивов для определения осей графиков и серий объектов, т.е., график может иметь одну или более осей и множество серий. Класс ‘Renderer’, обеспечивает общий интерфейс для отображения в SVG. Класс ‘Point’ является простым объектом, содержащим координаты на осях и обратную ссылку на соответствующий объект серии.

Следует отметить, что метод ‘Highcharts.Chart’ создает и возвращает объект графики, а также имеет второй необязательный параметр, называемый обратным вызовом. Внутри функции обратного вызова можно получить либо методы компонента, либо доступ к свойствам внутри объекта диаграммы. Созданный объект диаграммы передается через единственный параметр функции обратного вызова. Существует несколько способов перемещения по объекту диаграммы – через ее модель иерархии или путем извлечения компонента непосредственно с помощью метода ‘Chart.get’. На практике допускается совмещение этих подходов.

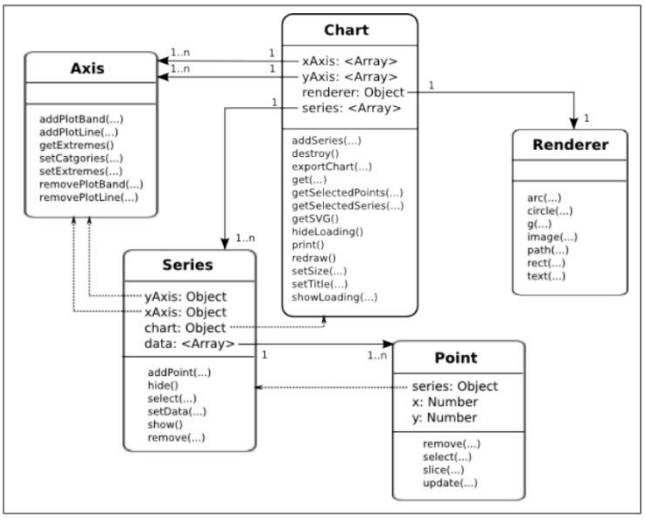

Рис. 2. Основные классы ‘Highcharts’

Существует четыре метода обновления наборов данных: ‘Series.setData’, ‘Chart.addSeries’, ‘Point.update’ и ‘Series.addPoint’. При обновлении данных с помощью ‘Series.setData’ передается новый набор данных к уже существующему. Данные могут быть переданы в виде одномерного массива, массива пар значений *x* и *y* или массива объектов. Метод является простейшей формой всех подходов и не предоставляет каких-либо возможностей анимации.

Метод ‘Point.update’ позволяет обновлять отдельные элементы данных. Он имеет прототип ‘setData’, который принимает одно значение, значение пары *x* и *y* или объект элемента данных. Каждый вызов обновления отображает диаграмму с анимацией или без нее. При использовании ‘Point.update’ выполняется последовательное обращение к каждому элементу данных объекта и вызывается определяющая его функция. Для экономии процессорного времени параметру ‘redraw’ задается значение ‘false’, а после обновления последнего элемента данных вызывается метод ‘Chart.redraw’. ‘Point.update’ переопределяет каждый элемент данных вдоль выбранного направления. Это создает эффект «волнистости» при обновлении графика. Вместо обновления каждого отдельного элемента данных можно последовательно использовать методы ‘Point.remove’ для удаления массива данных и ‘Series.addPoint’ для добавления новых элементов в набор.

Обработка событий в ‘Highcharts’ производится в нескольких областях и может относиться к диаграммам, наборам данных и базовым осям. События инициализируются вызовами API или взаимодействием пользователя с диаграммой.

Для инициализации графики и передачи для визуализации полученных данных необходимо сконфигурировать ряд свойств, задав им соответствующие значения. Некоторые из этих свойств приведены ниже:

- chart: определяет конфигурацию верхнего уровня для графики (общий вид, размерность, содержимое, анимация и взаимодействие с пользователем);
- series: представляет массив объектов, содержащих единичные либо множественные наборы данных (возможно, с дополнительными их свойствами);
- xAxis/yAxis: конфигурирует свойства осей (стиль отображения, разбивка, интервалы, тип линии);
- tooltip: конфигурирует формат и стиль подсказок для набора данных;
- legend: представляет формат и стиль надписей графики;
- plotOptions: содержит все параметры графики (отображение анимации, взаимодействие с пользователем, определение типа графика);
- exporting: конфигурирует параметры печати и экспорта графики.

Для того что бы воспользоваться ‘Highcharts’, необходимо подключить соответствующий JavaScript-файл к HTML-документу. На рис. 3 приведен пример такого подключения с минимальной конфигурацией. При конфигурации ‘Highcharts’ на СУИГР мы следуем похожей схеме инициализации. При этом, однако, обращение к вспомогательным библиотекам осуществляется после полной загрузки HTML-документа, что ускоряет работу портала. Заметим, что, перед тем как загрузить ‘Highcharts’, мы используем службу публичных библиотек ‘Google’ для загрузки библиотеки ‘jQuery’ (в данном случае – версии 1.8.2).

```
<!DOCTYPE HTML>
<html>
 <head>
 <meta http-equiv="Content-Type"
 content="text/html; charset=utf-8">
 <title>Highcharts</title>
 <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js">
</script>
 <script
type="text/javascript" src="Highcharts-4.0.4/js/Highcharts.js">
</script>
```

Рис. 3. Подключение библиотеки ‘Highcharts’

Пример инициализации графика с помощью библиотеки ‘Highcharts’ представлен на рис.4. График сплайна создается как новый объект, который содержит (либо наследует от класса ‘Chart’) все необходимые свойства. После создания объекта соответствующий тип графики отображается в веб-браузере. Внутри объекта определяются параметры, характеризующие внешний вид и структуру графики. В приведенном примере параметр ‘renderTo’ указывает ‘Highcharts’место расположение вывода графики в HTML-документе (в данном случае это идентификатор 'container').

```
<script type="text/javascript">
var chart;
 $(document).ready(function() {
 Chart = new Highcharts.Chart({
 chart: {
 renderTo: 'container',
   type: 'spline'
 }
```

Рис. 4. Инициализация графика с помощью ‘Highcharts’

По умолчанию область вывода графика состоит из трех элементов: область маркировки, промежуточная область (или интервал) и участок. Область маркировки представляет собой место расположения меток (названия графика и осей, надпись) и находится между краем области графика и внутренним краем промежуточной области. Промежуточная – это область между границей контейнера и внешним краем области маркировки. Под площадью участка понимается область внутри прямоугольника, которая содержит только график.

Существует два типа отображающейся на диаграмме информации – категории данных и собственно наборы этих данных (числовая информация). Для отображения имен и названий чаще всего

используют категории, которые представляют из себя массив строк. Каждая запись в массиве ассоциируется с определенным массивом упорядоченных данных. В качестве альтернативы могут быть сконфигурированы вложенные массивы отображаемой информации. ‘Highcharts’ извлекает данные, интерпретирует их тип, производит форматирование и соответствующим образом маркирует значения. Информация о формате данных и о том, как они должны быть представлены на диаграмме, определяется свойством ‘series’. При этом все значения данных на диаграмме передаются через поле ‘data’, которое может принимать массив в нескольких форматах: числовые значения, массив со значениями координат, а также объект со свойствами, описывающими группу данных. ‘Highcharts’ производит построение графиков и диаграмм даже в случае частичного отсутствия числовой информации (по умолчанию, соответствующие поля заполняются нулями).

Группировка столбчатой диаграммы позволяет объединять столбцы друг с другом вертикально. Программный код для количественной визуализации соответствующих значений (рис. 5) предоставляет возможность мгновенно наблюдать общие значения каждой категории и изменение отношений между рядами.

В примере, показанном на рис. 6, отображена статистика использования ресурсов сайтов NGI_UA задачами, которые поступали из грид-среды в 2018 году. Реализовать такое представление столбчатой диаграммы позволяет параметр ‘stacking’ с установленным значением ‘percent’ (см. рис. 5, где приведен код, формирующий данную диаграмму). При этом мы следуем принятым в EGI единицам измерения для использованных вычислительных ресурсов: процессорное время в часах (диаграмма 1), процессорное время в сутках (диаграмма 2), нормированное процессорное время в HEPspec06-часах (диаграмма 3), нормированное процессорное время в HEPspec06-сутках (диаграмма 4), астрономическое время в часах (диаграмма 5), астрономическое время в сутках (диаграмма 6), астрономическое время в HEPspec06-часах (диаграмма 7), астрономическое время в HEPspec06-сутках (диаграмма 8), количество выполненных задач (диаграмма 9). Отметим, что приведенные здесь диаграммы являются интерактивными, и при подведении курсора к определенному их фрагменту можно получить подробную информацию о соответствующем вкладе (в частности, о его абсолютной величине в соответствующих единицах измерения – так, как это показано на диаграмме 1 рис. 6).

В качестве вспомогательного средства визуализации данных (помимо ‘Highcharts’) на СУИГР также используется среда ‘Dashing’ [19]. С ее помощью формируются панели ‘Dashboard’ для сокращенного представления статистической информации. На рис. 7 показан пример таких панелей, отображающих информацию об использовании вычислительных ресурсов NGI_UA в целом и одного из входящих в NGI_UA грид-сайтов (фактически самого активного и загруженного) за текущий месяц.

```
<script type="text/javascript">
  $(function() {
    $.getJSON(
      './json/json_chart_report_accounting_ngi_ua_sites_percent.php',
      function(data) {
        var chart;
        $(document).ready(function() {
          chart = new Highcharts.Chart({
            chart: {
              renderTo: 'container_site_percent',
              type: 'column'
            },
            title: {
              text: 'NGI UA sites relative performance
              in percentage for current year'
            },
            subtitle: {
              text: 'Statistical calculations for
              <?php echo "".date("Y")."" ?> year'
            },
            xAxis: {
              categories: [
                'CPU Time hours',
                'CPU Time days',
                'CPU Work HS06/hours',
                'CPU Work HS06/days',
                'Wallclock Time/hours',
                'Wallclock Time days',
                'Wallclock Work HS06/hours',
                'Wallclock Work HS06/days',
                'Jobs'
              ]
            },
            yAxis: {
              min: 0,
              title: {
                text: 'Relative perfomance %'
              }
            },
            tooltip: {
              formatter: function() {
                return '' + 'Total: ' +
                  this.point.stackTotal +
                  '<br/>' + this.series.name +
                  ': ' + this.y + ' (' +
                  Math.round(this.percentage) +
                  '%)';
              }
            },
            plotOptions: {
              column: {
                stacking: 'percent'
              }
            },
            series: data
          });
        });
      });
  });
</script>
```

Рис. 5. Фрагмент кода, формирующего столбчатую диаграмму

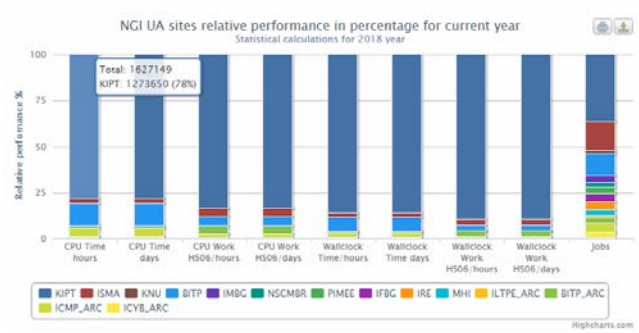

Рис. 6. Вклады отдельных сайтов NGI_UA в использование ресурсов в WLCG за 2018 год

Рис. 7. Панели ‘Dashboard’с информацией об использовании вычислительных ресурсов за текущий месяц NGI_UA в целом (слева) и одного из входящих в NGI_UA грид-сайтов – Kharkov-KIPT-LCG2 (справа)

Для формирования пользовательского интерфейса на портале используется среда разработки ‘Bootstrap’ [20] с базовыми настройками HTML, CSS и JavaScript. ‘Bootstrap’ является популярным инструментом для создания интерфейсов веб-приложений с доступной документацией и относительной простотой настройки. Верстка с помощью ‘Bootstrap’ заключается в использовании встроенных компонентов, зачастую представляющих из себя уже готовые HTML-блоки с предопределенными стилями. Некоторые компоненты используют JavaScript. Существует также ряд сторонних компонентов, которые существенно расширяют базовые возможности ‘Bootstrap’ и при необходимости могут быть интегрированы в среду разработки. ‘Bootstrap’ позволяет быстро создавать рабочие прототипы, а также обеспечивает согласованность кода с наименованием селекторов и структурированием CSS-файлов в рамках стандартизованной конфигурации. В последних версиях ‘Bootstrap’ введен компонент ‘Cards’, во многом упрощающий структуру среды разработки и делающий ее более легкой для восприятия и использования. Кроме того, все плагины с JavaScript переписаны в соответствии со стандартом ES6, что позволяет использовать новейшие функции JavaScript при проектировании. Для создания макетов страниц применяется система сеток ‘Bootstrap Grid System’, упрощающая разработку адаптивных веб-страниц. Сетка разделена на 12 колонок и является основой макета страницы. При настройке размеров вложенных элементов указывается число столбцов базовой 12-колоночной сетки, которое должен занимать конкретный элемент. Для создания навигационных панелей (с указанием их расположения на веб-странице) и прочих элементов интерфейса в ‘Bootstrap’ предусмотрены соответствующие компоненты. Пример веб-страницы, сформированной с помощью ‘Bootstrap’ и содержащей хронологический график использования вычислительных ресурсов грид-сайта Kharkov-KIPT-LCG2, приведен на рис. 8.

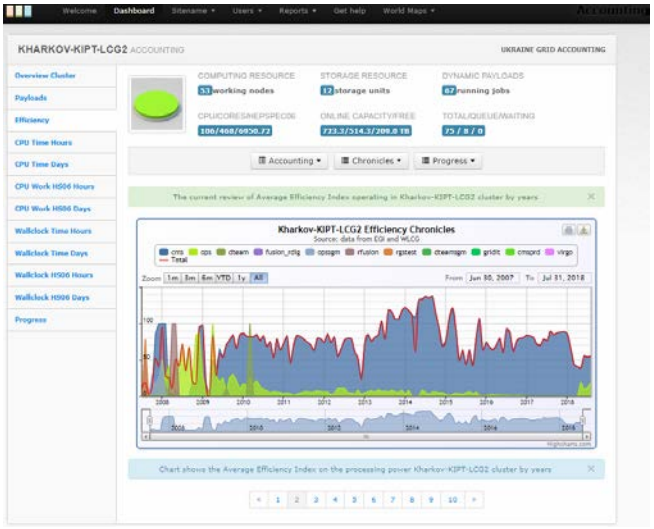

Рис. 8. Веб-страница, сформированная с помощью ‘Bootstrap’, с хронологическим графиком использования вычислительных ресурсов грид-сайта Kharkov-KIPT-LCG2

4. Заключение

Рассмотрен инструментарий, применённый при создании системы сбора и публикации информации об использовании вычислительных ресурсов грид-сети Украины NGI_UA (СУИГР). Система работает в рамках специализированного вычислительного комплекса ННЦ ХФТИ для обработки данных с Большого адронного коллайдера и содержит актуальные данные о работе всех зарегистрированных в EGI сайтов NGI_UA. Основным источником информации является портал [5] учета использования вычислительных ресурсов EGI. Процесс сбора данных о работе сайтов NGI_UA сводится к выполнению таких последовательных операций: формирование запроса к порталу [5] и получение информации в виде JSON-документа, извлечение данных, запись полученных значений в MySQL-базу пор-

тала СУИГР. Для визуализации данных используется библиотека ‘Highcharts’, которая позволяет работать с данными в различных форматах. В качестве вспомогательного средства также используется среда ‘Dashing’, с помощью которой формируются панели сокращенного представления статистической информации. Информационный ресурс характеризуется надлежащим уровнем наглядности и обеспечивает легкость доступа к статистическим данным, характеризующим работу вычислительных комплексов, с предоставлением визуальных отчетов в виде динамических графиков и диаграмм. Портал доступен по адресу https://grid-accounting.kipt.kharkov.ua и содержит широкий спектр инструментов для представления статистических данных с возможностью компоновки, сортировки, выборки и сравнения за произвольный период времени для всех ВО, поддерживаемых в NGI_UA.

Работа поддержана грантами НАНУ в рамках целевой комплексной программы научных исследований НАН Украины «Грид-инфраструктура и грид-технологии для научных и научно-прикладных применений» (2014 – 2018).

Литература:

1. LHC Computing Grid http://wlcg.web.cern.ch.
2. European Grid Infrastructure http://www.egi.eu.
3. The Open Science Grid http://opensciencegrid.org
4. PRACE http://www.prace-ri.eu/
5. EGI Accounting Portal http://accounting.egi.eu.
6. GRACC https://gracc.opensciencegrid.org
7. HLRmon http://www.italiangrid.it/operations/accounting_portal
8. Levchuk L.G., Soroka D.V., Voronko M.V., Zub S.S. LCG middleware at the KIPT CMS Linux cluster // Visnyk Kharkivskogo natsionalnogo universitetu. Series“Matematychne modelyuvannia; informatsiini technologii, avtomatyzovani systemy upravlinnia”. 2005, № 5 (703), p. 74-86.
9. APEL https://apel.github.io
10. Middleware for Grid computing http://glite.cern.ch.
11. UMD http://repository.egi.eu/category/umd_releases
12. SPEC http://www.spec.org/
13. HEPspec06 https://www.spec.org/cpu2006
14. JQ JSON processor https://stedolan.github.io/jq
15. Highcharts http://www.Highcharts.com
16. Mootools https://mootools.net
17. Prototype http://prototypejs.org
18. jQuery https://jquery.com
19. Dashing dashboard framework http://dashing.io
20. Bootstrap https://getbootstrap.com
21. European Middleware Initiative http://www.eu-emi.eu.
22. Cloud Platform https://www.heroku.com.

Надыйшла до редколегії 21.11.2018
Рецензент: д-р техн. наук, проф. Хажмурадов М.А.

Клименко Ким Александрович, ИФВЭЯФ ННЦ ХФТИ. Научные интересы: информационные технологии. Адрес: Украина, 61108, Харьков, ул. Академическая, 1, ННЦ ХФТИ, тел. +38 (057) 335-35-30.
Левчук Леонид Геннадиевич, заведующий отделом экспериментальных исследований по физике элементарных частиц и ядерной физике высоких энергий на ускорителях ЛУЭ-2000 и ЛУЭ-300 МЭВ (ИФВЭЯФ) ННЦ ХФТИ. Научные интересы: физика элементарных частиц. Адрес: Украина, 61108, Харьков, ул. Академическая, 1, ННЦ ХФТИ, тел. +38 (057) 335-35-30.
Приставка Александр Сергеевич, ИФВЭЯФ ННЦ ХФТИ. Научные интересы: функциональное программирование. Адрес: Украина, 61108, Харьков, ул. Академическая, 1, ННЦ ХФТИ, тел. +38 (057) 335-35-30.
Куров Алексей Александрович, ИФВЭЯФ ННЦ ХФТИ. Научные интересы: операционные системы и сети. Адрес: Украина, 61108, Харьков, ул. Академическая, 1, ННЦ ХФТИ, тел. +38 (057) 335-35-30.
Kim Klimenko, National Science Center Kharkov Institute of Physics and Technology. Scientific interests: information technology. Address: 1, Akademicheskaya St., Kharkov, 61108, Ukraine, tel +38 (057) 335-35-30.
Leonid Levchuk, National Science Center Kharkov Institute of Physics and Technology. Scientific interests: elementary particle physics. Address: 1, Akademicheskaya St., Kharkov, 61108, Ukraine, tel +38 (057) 335-35-30.
Alexander Pristavka, National Science Center Kharkov Institute of Physics and Technology. Scientific interests: functional programming. Address: 1, Akademicheskaya St., Kharkov, 61108, Ukraine, tel +38 (057) 335-35-30.
Alexey Kurov, National Science Center Kharkov Institute of Physics and Technology. Scientific interests: operating systems and networks. Address: 1, Akademicheskaya St., Kharkov, 61108, Ukraine, tel +38 (057) 335-35-30.