Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Легалов
подпись инициалы, фамилия
«___» _____ 2016 г.

БАКЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специализации

Система мониторинга комплекса высокопроизводительных вычислений СФУ
тема

Пояснительная записка

Руководитель	_____	к.т.н., доцент	Д. А. Кузьмин
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Д. С. Макаров
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	к.т.н., доцент	В. И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Эффективное управление сложной и динамичной системой, прогнозирование ее изменений возможно только на основе сбора и анализа непрерывного потока информации о ее состоянии, а также закономерностях всей совокупности процессов, протекающих в ней. Поэтому для действенного и качественного управления любыми процессами необходимо постоянно в режиме реального времени наблюдение и слежение за состоянием системы. Так, на сегодняшний день наиболее важным инструментом наблюдения, анализа и прогнозирования, а также фактором в принятии обоснованных и наиболее эффективных решений, является система мониторинга [1].

Задача комплексного мониторинга больших центров обработки данных является актуальной. Администраторы подобных комплексов должны получать и анализировать множество параметров, объектов и сущностей – от загрузки процессора до инженерных параметров состояния помещения комплекса, и комплекс высокопроизводительных вычислений СФУ не является исключением.

1 Оценка современного состояния исследуемой проблемы

На данный момент система мониторинга комплекса высокопроизводительных вычислений СФУ использует ряд независимых программных продуктов, каждый из которых обеспечивает решение своего круга задач. Так, для мониторинга загруженности узлов комплекса используется Ganglia, мониторинг состояния оборудования осуществляется ПО IBM Director, мониторинг инженерных параметров состояния серверных помещений комплекса ведется с помощью системы Rittal CMC-TC. Вместе все эти системы способны справляться со всеми задачами мониторинга, но при такой организации возникают проблемы в поддержке, изучении как минимум трёх систем мониторинга, что приводит к увеличению временных затрат на их обслуживание.

Для устранения этих проблем принято решение разработать единую систему, которая бы справлялась со всеми задачами мониторинга и имела точку входа для сотрудников комплекса, представляя данные о состоянии комплекса в удобном виде и позволяла осуществлять гибкую настройку всей системы и её отдельных модулей.

2 Постановка задачи

Была сформулирована следующая задача: ”Конструирование системы мониторинга”.

К системе мониторинга были предоставлены следующие требования:

1. Система должна уметь производить мониторинг эксплуатационных параметров ЦОДов;
2. Система должна уметь производить мониторинг физических параметров помещений;
3. Производить мониторинг различных параметров на уровне сервисов и пользователей;

4. Иметь удобный, интуитивно понятный(интерактивный) веб-интерфейс;

5. Иметь систему оповещения для критических ситуаций;

6. Иметь развитую систему отчётов для получения сводной статистики работы системы мониторинга;

7. Быть лёгкой в обслуживании.

3 Общие сведения о мониторинге

3.1 Понятие и сущность мониторинга

Мониторинг – (английское Monitoring – осуществление контроля, слежения) комплекс диагностических наблюдений, систематический сбор и обработка информации, которая может быть использована для выявления характера изменений в конкретном объекте за определенный промежуток времени [2].

Мониторинг представляет собой достаточно сложное и неоднозначное явление. В различных сферах он используется с различными целями, но при этом обладает общими характеристиками и свойствами. Однако, разные системы мониторинга, обладая общими чертами, существуют и развиваются достаточно изолированно в рамках той или иной науки или области управления [3].

3.2 Задачи мониторинга

Мониторинг несёт одну или более из трёх организационных функций:

1. выявляет состояние критических или находящихся в состоянии изменения явлений окружающей среды, в отношении которых будет выработан курс действий на будущее;

2. устанавливает отношения со своим окружением, обеспечивая обратную связь, в отношении предыдущих удач и неудач определенной политики или программ;

3. устанавливает соответствия правилам и контрактным обязательствам [2].

4 Обзор существующих систем и комплексных решений мониторинга

4.1 Ganglia

Ganglia – масштабируемая распределённая система мониторинга кластеров параллельных и распределённых вычислений и облачных систем с иерархической структурой. Позволяет отслеживать статистику и историю (загруженность процессоров, сети) вычислений в реальном времени для каждого из наблюдаемых узлов [4].

4.1.1 Назначение и применение

Система построена по иерархическому принципу для интеграции кластеров. Для мониторинга состояния кластеров и их объединения используется древовидная система основанная на P2P-соединениях и широковещательных протоколах. Использует такие технологии, как XML для представления данных, XDR для сжатия данных, RRDtool для хранения и визуализации данных. Для отображения страниц статистики используется шаблонизатор TemplatePower.

Система портирована на широкий спектр операционных систем и процессорных архитектур, известно об её использовании более чем 500 кластерах по всему миру. Существуют сборки для следующих операционных систем: Linux (i386, x86-64, SPARC, DEC Alpha, powerpc, m68k, MIPS, ARM,

PA-RISC, S390),FreeBSD, NetBSD, OpenBSD, DragonflyBSD, Mac OS X, Solaris (SPARC), AIX, IRIX, Tru64, HP-UX и Windows NT/XP/2000/2003/2008.

Используется для связи кластеров в университетских кампусах по всему миру и может масштабироваться для обработки кластеров имеющих до 2000 узлов в своем составе.

Необходимые пакеты для установки Ganglia присутствуют в большинстве репозиториев современных дистрибутивов Linux [4].

4.1.2 Демоны

Демон gmetad («Ganglia Meta Daemon») используется для сбора информации и её отображения на стороне пользователя. По умолчанию для получения данных от других клиентов используется TCP-порт 8651.

Демон gmond («Ganglia monitoring daemon») запускается на всех узлах для которых необходимо собирать статистику.

Gmond представляет собой многопоточный демон, который работает на каждом узле кластера, который нужно мониторить. Установка не требует наличия файловой системы NFS или базы данных, установкиотдельных учётных записей или обслуживания файлов конфигурации.

Gmond имеет четыре основные обязанности:

- отслеживать изменения в состоянии узла;
- показывать соответствующие изменения;
- наблюдать за состоянием всех других узлов через одноадресный или многоадресный канал передачи;
- отвечать на запрос XML-описания состояния кластера.

Каждый gmond передает информацию двумя различными способами:

- в канал групповой рассылки в формате external data representation (XDR) с использованием UDP-сообщений;
- отправка XML через соединение TCP [5].

4.1.3 Модули

Модуль *gstat* (англ. *GangliaClusterStatusTool*) – утилита командной строки, позволяющая импортировать информацию из Ganglia в другие приложения.

Для ввода данных из сторонних источников используется модуль gmetric (gexecd) – масштабируемая система удалённого выполнения задач (программ) в кластерах, которая может работать совместно с системой Ganglia. Для удалённого выполнения параллельных (распределённых) заданий используется RSA-аутентификация (демон authd). Система прозрачно перенаправляет программные потоки (stdin, stdout, stderr) и события между распределёнными процессами, что позволяет создавать распределённую среду переменных окружения и масштабировать систему до более чем 1000 узлов в составе без потери надёжности. Механизм работы основывается на создании древовидного массива всех TCP-сокетов между узлами и распространении управляющей информации по всему дереву. С помощью иерархической системы управления gexecраспределяет как и вычислительные задания, так и ресурсы. Это позволяет устранить проблемы, связанные с ограничениями каждого из узлов, например, ограничение на количество открытых дескрипторов файлов. В gexec интегрирована возможность распределения нагрузки в кластере. Информация о загруженности узлов запрашивается у gmond.

Для хранения и визуализации данных в системе используется инструмент RRDtool [5].

4.1.4 Веб-интерфейс

Веб-интерфейс Ganglia, который изображен на рисунке 1, обеспечивает отображение собранных данных монитринга через динамические веб-страницы. Наиболее важным является то, что оно тображает данные информативным способом, удобным для администраторов и пользователей. Хоть и изначально

веб-интерфейс был простой HTML-страницей, но после он стал целой системой, хранящей подробный и разнообразный архив из собранных данных.

Веб-интерфейс ганглии нацелен на системных администраторов и пользователей. Например, они могут наблюдать статистику использования процессора напротяжении прошлого часа, дня, недели, месяца или года. Веб-интерфейс также отображает похожие графики для использования памяти, диска, сети, числа работающих процессов и множества других метрик.

Веб-интефейс зависит от наличия модуля *gmetad*, который снабжает его данными с разных источников. По умолчанию он работает на порту 8651 и ожидает получения XML-древовидного документа. Сами веб-страницы чрезвычайно динамичны: любое изменение в данных немедленно отражается на веб-сайте. Из такой организации работы получается очень «отзывчивый» сайт, но требующий парсинга страницы при каждом посещении страницы. Следовательно, веб-интерфейс следует запускать на достаточно мощном, распределенном оборудовании, если он будет отображать большое количество данных.

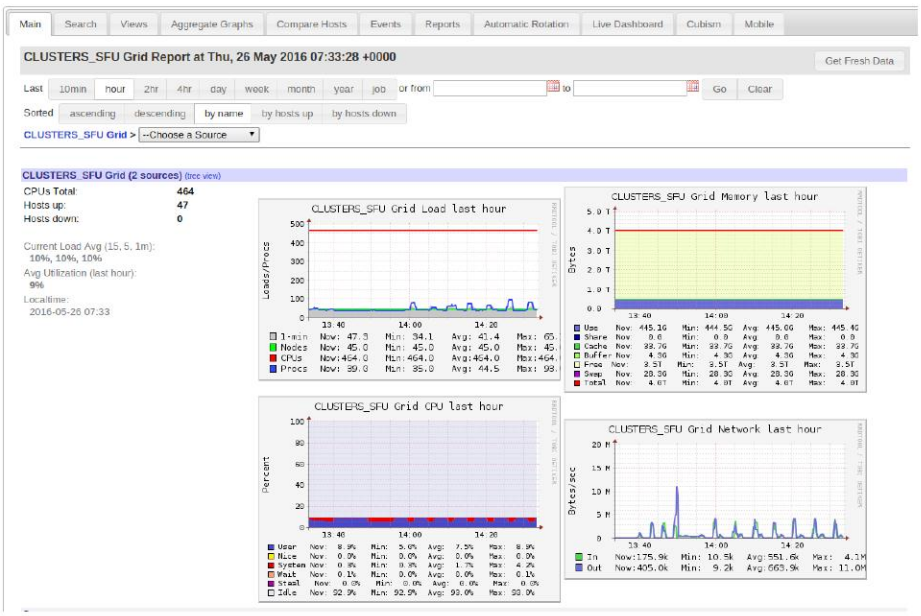

Рисунок 1 – Веб-интерфейс Ganglia

Веб-интерфейс написан на PHP и для отображения данных использует графики, сгенерированные модулем gmetad. Он тестировался на многих системах семейства UNIX наряду с веб-сервером Apache и модулем для PHP [5].

4.2 IBM-Director

IBM Director – это система типа element management system (EMS) (иногда используется альтернативный термин workgroup management system (система управления рабочими группами), впервые представленная корпорацией IBM в 1993 г. под названием NetFinity Manager. Этот программный продукт первоначально был написан в расчете на исполнение под управлением операционной системы OS/2 2.0. Впоследствии он прошел через серию последовательных переименований. В 1996 г. он получил имя IBM PC SystemView. В том же году он был переименован в TME 10 NetFinity. В следующем году он получил несколько измененную версию своего первоначального имени: IBM NetFinity Manager.

В 1999 г. корпорация IBM представила новый продукт NetFinity Director, который базировался на продукте Tivoli IT Director. Вышеуказанный продукт был предназначен для замены продукта IBM NetFinity Manager. После того, как IBM сменила название своей линейки корпоративных серверов с NetFinity на xSeries, описываемый продукт получил имя IBM Director.

С выходом версии 6.1 продукт IBM Director получил новое имя: IBM Systems Director. Данное решение доступно для работы с любыми серверами System x и BladeCenter, включая специальные конфигурации для СМБ Express Seller.

4.2.1 Компоненты

Продукт IBM Director состоит из следующих трех компонентов: агент, консоль и сервер. Для полного использования всех возможностей продукта IBM Director на систему, подвергаемую мониторингу, должен быть установлен компонент IBM DirectorAgent. Инвентаризационные данные и данные управления хранятся в базе данных с поддержкой SQL (Oracle, SQL Server, IBM DB2UniversalDatabase или PostgreSQL), которая может находиться на отдельной системе или на том же сервере, где размещается репозиторий IBM DirectorServer. В менее масштабных развертываниях также может быть использован продукт MicrosoftJet или MSDE. Конфигурирование сервера и управление им осуществляется с помощью инструмента IBM DirectorConsole, установленного на любой рабочей станции под управлением Linux или MicrosoftWindows [6].

4.2.2. Применение

IBM Director поддерживает следующие основные задачи

- Идентификация объектов;
- Система управления BladeCenter;
- Обозреватель CIM;
- Настройка агента SNMP;
- Администрирование базы данных с политикойзахвата;
- Планирование действий для проведения мероприятий;
- Журналирование событий;
- Запуск внешнего приложения;
- Передача файла;
- Определение статуса оборудования;
- Инвентаризация;
- Конфигурирование сети;

- Управление процессами;
- Система управления стойками;
- Дистанционное управление;
- Удаленная сессия;
- Розничное управление периферийными устройствами;
- Распространение программного обеспечения RMA;
- Мониторинг ресурсов;
- Планировщик;
- Диспетчер конфигурации сервера;
- SNMP-браузер;
- Системные учетные записи;
- Менеджер обновлений;
- Администрирование пользователей [6].

4.3 Nagios

4.3.1 Nagios – это приложение, предназначенное для выполнения мониторинга систем и сетей. Оно следит за приложениями и службами и генерирует оповещения в зависимости от поведения наблюдаемых служб. Nagios это рекурсивный акроним, расшифровывающийся как Nagios Ain't Gonna Insist On Sainthood (Nagios не собирается настаивать на святости – намек на прежнее название проекта, NetSaint – сетевой святой). Nagios чрезвычайно мощный и гибкий инструмент, позволяющий настроить мониториг для окружения любого уровня сложности и масштаба [7].

4.3.2 Возможности Nagios:

- Мониторинг сетевых служб (SMTP, POP3, HTTP, NNTP, ICMP, SNMP);
- Мониторинг состояния хостов (загрузка процессора, использование диска, системные логи) в большинстве сетевых операционных систем;

- Поддержка удаленного мониторинга через шифрованные туннели SSH или SSL;
- Простая архитектура модулей расширений (плагинов) позволяет, используя любой язык программирования по выбору (Shell, C++, Perl, Python, PHP, C# и другие), легко разрабатывать свои собственные способы проверки служб;
- Параллельная проверка служб;
- Возможность определять иерархии хостов сети с помощью «родительских» хостов, позволяет обнаруживать и различать хосты, которые вышли из строя, и те, которые недоступны;
- Отправка оповещений в случае возникновения проблем со службой или хостом (с помощью почты, пейджера, смс, или любым другим способом, определенным пользователем через модуль системы);
- Возможность определять обработчики событий произошедших со службами или хостами для проактивного разрешения проблем;
- Автоматическая ротация лог-файлов;
- Возможность организации совместной работы нескольких систем мониторинга с целью повышения надёжности и создания распределенной системы мониторинга [7].

4.4 Zenoss

4.4.1 ZenossCore является приложением управления сетью и системами предприятия написана на Python/Zope. Zenoss предоставляет комплексный продукт для мониторинга доступности, производительности, событий и конфигурации [8].

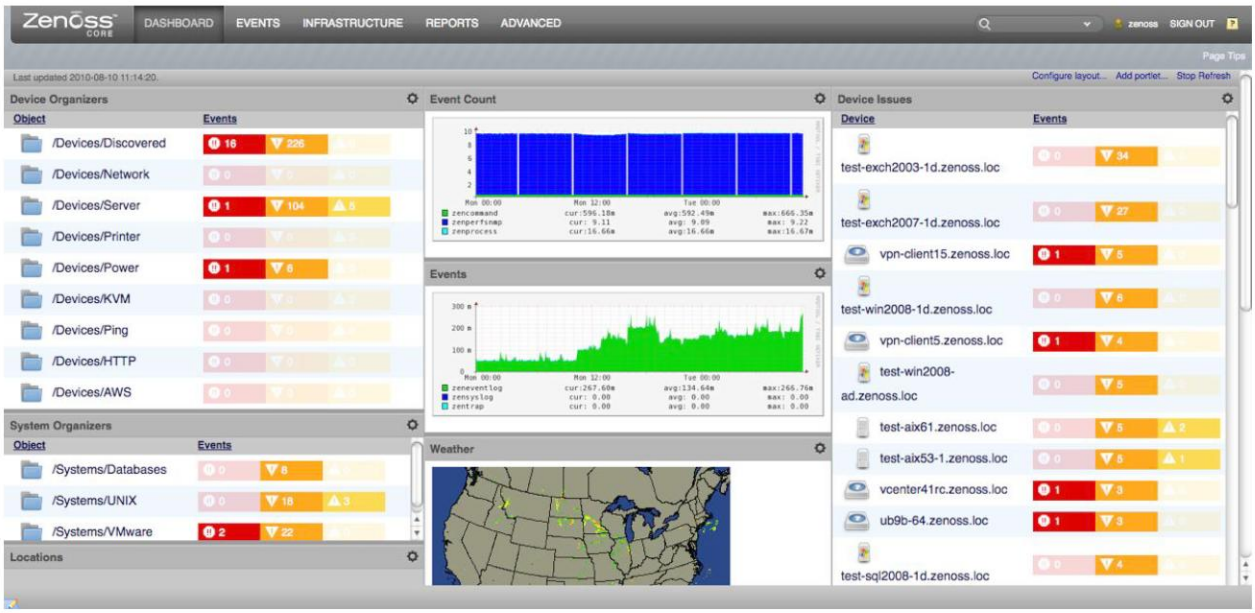

Рисунок 2 – Веб-интерфейс Zenoss

4.4.2 Характеристики ZenossCore:

- мониторинг сети, серверов и приложений посредством одного продукта.
- мониторинг устройств с исползованием SNMP протокола.
- оповещение в режиме реального времени при сбоях и снижении производительности.
- интерфейс для слежения за событиями EventConsole, куда входят оповещения Zenoss, SNMP-мониторинг и журнал событий.
- благодаря открытой архитектуре, Zenoss легко взаимодействует со сторонними приложениями (возможен как импорт, так и экпорт данных в другие продукты).

4.4.3 Система, построенная на ZenossCore, обеспечивает следующие возможности:

- мониторинг сетевых устройств при помощи SNMP, SSH, WMI, JMX, Ping/ICMP и Syslog
- мониторинг сетевых сервисов – HTTP, POP3, NNTP, SNMP, FTP
- мониторинг системных ресурсов популярных операционных систем

- мониторинг производительности устройств
- система оповещения с настраиваемыми событиями, реакцией и обнаружением взаимосвязи
- возможность расширения функциональности за счет плагинов собственной разработки ZenPack и плагинов системы мониторинга Nagios

Функция автообнаружения позволяет быстро собрать информацию обо всех активных системах в сети. При этом ядро Zenoss умеет анализировать среду, что дает возможность быстро разобраться с большим количеством специфических устройств. Параметры собранные разными способами нормализуются с использованием шаблонов и приводятся к единому виду. В случае обнаружения проблем, Zenoss может не только отправить сообщение администратору, но и например выполнить команду на перезапуск сервиса [8].

4.4.4 Другие преимущества:

- возможность изменения исходного кода программы.
- шаблоны отчетов и администрирования.
- мониторинг сетевых протоколов (таких как FTP). Стандартизированная структура расширений для сетевого управления.
- это ПО реализует расширяемую архитектуру, позволяющую размещать серверы ZenossCore вблизи тех устройств, мониторинг которых они будут осуществлять, и добавлять эти серверы по мере необходимости [8].

4.5 Zabbix

Zabbix – свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования. Для хранения данных используется MySQL, PostgreSQL, SQLite или Oracle. Веб-интерфейс, изображенный на рисунке 3, написан на PHP. Zabbix поддерживает несколько видов мониторинга:

- Simplechecks – может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP, без установки какого-либо программного обеспечения на наблюдаемом хосте.
- Zabbix agent – может быть установлен на UNIX-подобных или Windows-хостах для получения данных о нагрузке процессора, использования сети, дисковом пространстве и т. д.
- Externalcheck – выполнение внешних программ. ZABBIX также поддерживает мониторинг через SNMP [9].

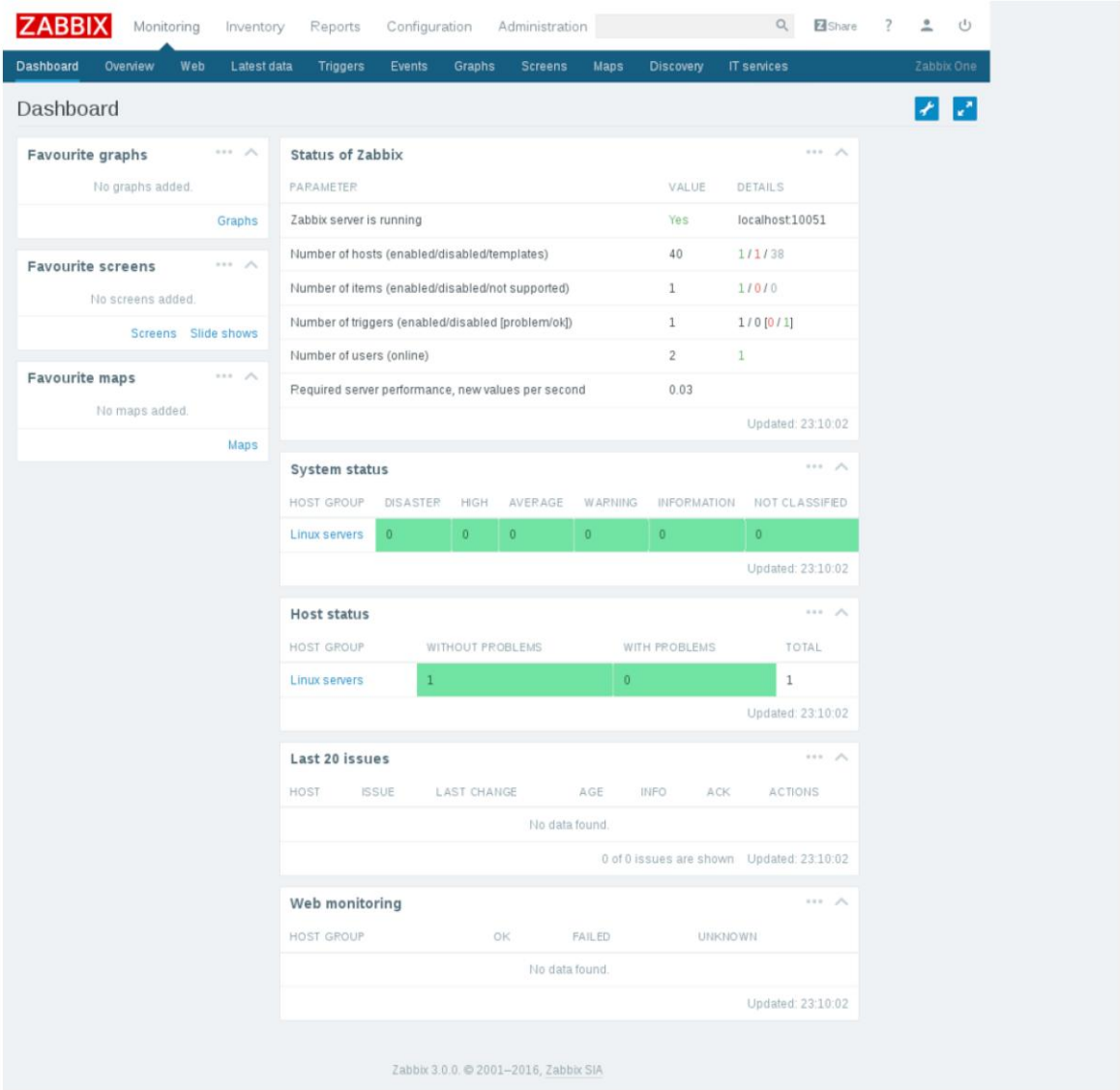

Рисунок 3 – Веб-интерфейс Zabbix

4.5.1 Возможности Zabbix

Zabbix - высоко интегрированное решение мониторинга сети, которое предлагает множество возможностей в одном пакете.

- Сбор данных:
    - проверки доступности и производительности;
    - поддержка мониторинга по SNMP, IPMI, JMX;
    - пользовательские проверки;
- Сбор желаемых данных за выборочные интервалы;
- Широкие возможности визуализации:
    - Графики в режиме реального времени;
    - Карты сети;
    - Пользовательские экраны и слайд шоу;
    - Отчеты;
- Хранение истории;
- Гибкая настройка:
    - Определение порогов;
    - Настраиваемые оповещения;
    - Автоматические реакции на события, в том числе удаленные команды;
    - Шаблонизация;
    - Система прав доступа;
- Возможности web-мониторинга;
- Веб интерфейс;
- Zabbix API;
- Наличие нативных клиентов под разные ОС;
- Готовое решение Zabbix, основанное на Open SUSE [10].

4.5.2 Архитектура и основные понятия Zabbix

4.5.2.1 Zabbix состоит из нескольких важных компонентов программного обеспечения, функции которых изложены ниже.

4.5.2.2 Zabbix сервер является главным компонентом, которому агенты сообщают информацию и статистику о доступности и целостности. Сервер является главным хранилищем, в котором хранятся все данные конфигурации, статистики, а также оперативные данные. Сервер выполняет опрос и захват данных, он вычисляет триггеры, отправляет оповещения пользователям. Это главный компонент которому Zabbix агенты и прокси отправляют данные доступности и целостности системы. Сервер может самостоятельно удаленно проверять сетевые устройства (так же как и веб сервера и почтовые сервера) используя простые проверки сервиса.

Сервер является главным хранилищем, в котором хранятся все данные конфигурации, статистики, оперативные данные, а так же эта сущность в Zabbix, которая будет активно уведомлять администраторов в случае возникновения проблем в любой из наблюдаемых систем.

Функционал базового Zabbix сервера разделен на три отдельных компонента:

- Zabbix сервер;
- веб интерфейс;
- хранилище в базе данных [10].

4.5.2.3 Zabbix агенты разворачиваются на наблюдаемых целях для активного мониторинга за локальными ресурсами и приложениями (статистика жестких диски, памяти, процессоров и т.д.).

Агент собирает локальную оперативную информацию и отправляет данные Zabbix серверу для дальнейшей обработки. В случае проблем (таких как рабочий жесткий диск заполнен или упал процесс сервиса), Zabbix сервер может быстро уведомить администраторов конкретного сервера, который сообщил об ошибке.

Zabbix агенты чрезвычайно эффективны, потому что используют нативные системные вызовы для сбора информации статистики. Пассивные и активные проверки Zabbix агенты могут выполнять пассивные и активные проверки. В случае пассивной проверки агент отвечает на запрос данных. Zabbix сервер (или прокси) запрашивает данные, например, загрузку ЦПУ, и Zabbix агент возвращает результат. Активные проверки требуют более сложной обработки. Агент сначала получает список элементов данных для независимой обработки от Zabbix сервера. Далее он будет периодически отправлять новые значения серверу [10].

4.5.2.4 Zabbix прокси – это процесс, который может собирать данные мониторинга с одного или нескольких наблюдаемых устройств и отправлять эту информацию Zabbix серверу, в принципе прокси работает от имени сервера. Все собранные данные локально буферизуются и затем отправляются Zabbix серверу, которому принадлежит этот прокси.

Развертывание прокси опционально, но может быть очень полезна для распределения нагрузки на одиночный Zabbix сервер. Если данные собирают только прокси, то обработка этих данных на сервере значительно уменьшает загрузку ЦПУ и I/O диска.

Zabbix прокси – идеальное решение для централизованного мониторинга удаленных мест, филиалов и сетей без местных администраторов. Для Zabbix прокси требуется отдельная база данных [10].

4.5.2.5 В Zabbix 2.0 добавлена нативная поддержка для мониторинга JMX приложений введением нового демона Zabbix, называемого Zabbix Java gateway.

Zabbix Java gateway – это демон написанный на языке Java. Когда Zabbix сервер хочет знать значение конкретного JMX счетчика у узла сети, он опрашивает Zabbix Java gateway, который использует API управления JMX для опроса интересующего удаленного приложения [10].

4.6 Rittal CMC-TC

Rittal CMC-TC – полноценная система для контроля, настройки и сигнализации всех состояний по безопасности эксплуатации в сетевом шкафу передачи данных. Регулировка температуры, влажность, вибрация, а также персонифицированный контроль доступа осуществляется блоком CMC II. Имея обширную системную комплектацию, а также возможность использования различных технологий передачи, CMC II является прочным компонентом для безопасности информационных технологий.

Rittal предлагает разнообразные модульные решения для слежения и управления параметрами работы систем как в промышленности так и в IT сферах, включая видеонаблюдение, контроль доступа и присутствия, а также температуры, влажности, вибронагруженности, задымления.

Rittal CMC-TC постоянно наблюдает за всеми факторами, связанными с безопасностью систем шкафов или помещения, своевременно регистрирует все возможные опасности и уведомляет о них. Соответствующие действия можно предпринимать автоматически и при помощи центрального телеконтроля [11].

Рисунок 4 – Факторы, связанные с безопасностью систем шкафов или помещений

4.6.1 Преимущества:

- CMC II – аварийная сигнализация по TCP/IP и SMNP, серийно;
- CMC II-ISDN – решение по желанию заказчика. Контроль за оборудованием кондиционирования Rittal;
- Расширяемость входов и выходов через I2C/Power-I2C. 3-фазный контроль за напряжением через I2C;
- Техника "включай и работай";
- Персонифицированный контроль доступа;
- Блок питания с широким диапозоном 48 – 230 В DC /50 /60 Hz;
- Связь функций с выключателем SSC Мульти;
- Модульная система[11].

4.6.2 Особенности конструкции:

- Способен к работе в сети с TCP/IP;
- Протокол простого сетевого управления (SNMP). Telnet/RS 232;
- Упрощенный протокол передачи файлов (TFTP);
- Широкополосный блок питания 48 – 230 В пост.тока /50 /60 Гц;
- Открытая система/ программируема;
- Расширяемые входы/ выходы [11].

4.6.3 Области применения:

- Сетевые, серверные и электронные шкафы;
- Помещения для сетевых распределителей;
- Наружные системы;
- Область автоматизации/ промышленности;
- Банки, страховые учреждения;
- Фирмы с несколькими передвижными филиалами;
- Сферы высокой степени безопасности [11].

4.6.4 Эффективность использования:

- Контроль за всеми важными условиями окружающей среды компонентов сети или аналогичного актуального в отношении безопасности оборудования;
- Автоматическое уведомление в случае тревоги;
- Заблаговременная или автоматическая инициализация контромер;
- Оптимальный обзор и контроль в безопасных внутрифирменных локальных сетях [11].

5 Сравнение существующих систем

В ходе сравнения существующих систем было выявлено, что среди них нет той системы, которая бы полностью удовлетворила все потребности в мониторинге высокопроизводительного комплекса. Близкой к этому оказалась система Zenoss, так как у неё есть множество базовых возможностей. В случае их недостатка мы можем написать и подключить к ней новые плагины, так как эта система имеет открытый исходный код. Но, сожалению, с полным комплектом возможностей данная система является платной.

6 Организация мониторинга в комплексе высокопроизводительных вычислений СФУ

6.1 Общие сведения о комплексе

Центр обслуживает научные коллективы СФУ, взаимодействует с наукоемкими предприятиями Сибири, отделениями РАН, университетами России и Европы.

Установлены программные комплексы позволяющие проводить ресурсоемкие расчеты в различных областях науки. Среди установленного ПО – ANSYS, Matlab, GAMESS-US, CFOUR, MRCC, NWChem, ORCA, AMBER 11,

SigmaFlow., ПО геномных исследований – ABySS, FastQC, MaSuRCA. Все программное обеспечение имеет академические лицензии или является продуктами с открытым кодом.

Сотрудники центра специализируются в области организации управления высокопроизводительными комплексами. Силами центра реализуется проект «Высокопроизводительные вычисления как сервис». Проект обеспечивает создание универсальной инфраструктуры обработки данных, в которой функционирует множество сервисов, решающих конкретные прикладные задачи. Доступ пользователей к распределенным ресурсам центра осуществляется через единый Web-портал. Реализована комплексная система управления, позволяющая администраторам эффективно управлять вычислительными ресурсами, вводить новое специализированное ПО, осуществлять мониторинг ресурсов и задач пользователей [12].

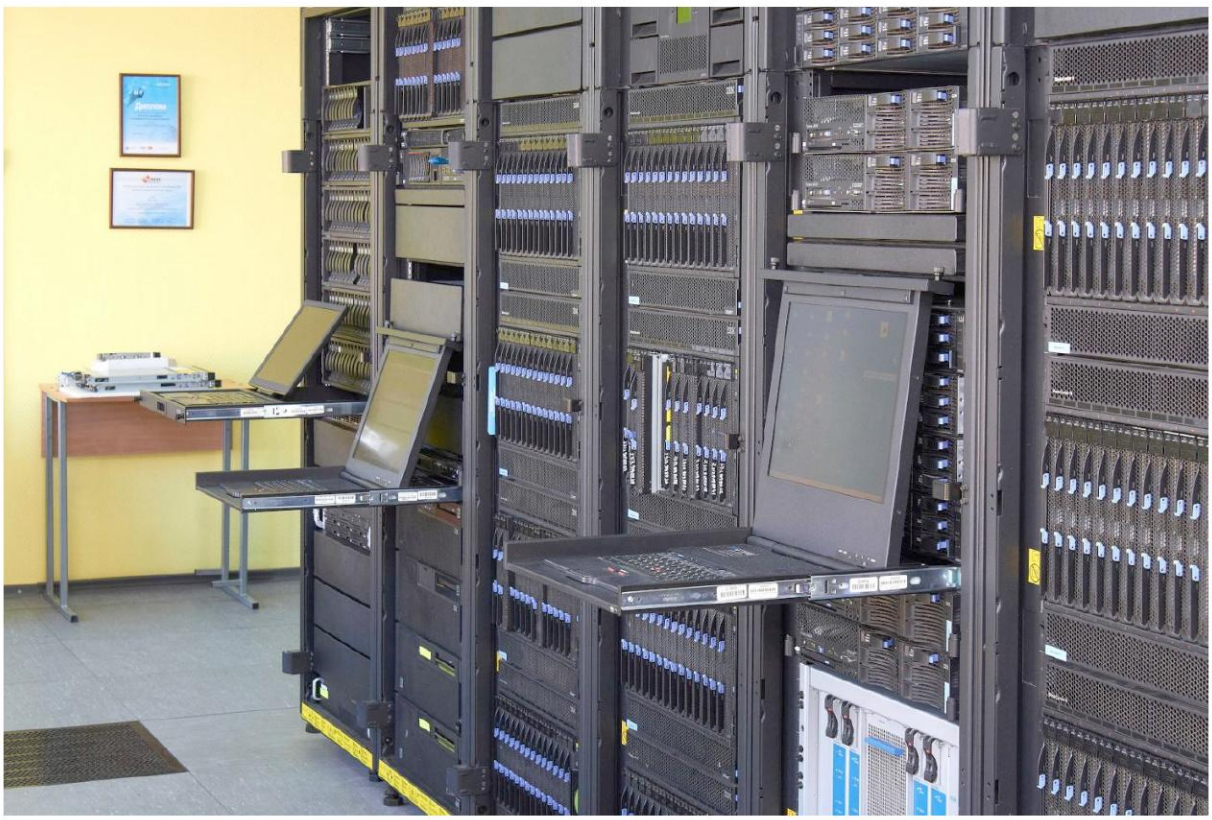

Рисунок 5 – Комплекс высокопроизводительных вычислений СФУ

6.2 Организация мониторинга

На данный момент система мониторинга комплексаиспользует ряд независимых программных продуктов, каждый из которых обеспечивает решение своего круга задач. Так, для мониторинга загруженности узлов комплекса используется Ganglia, мониторинг состояния оборудования осуществляется ПО IBM Director, мониторинг инженерных параметров состояния серверных помещений комплекса ведется с помощью системы Rittal CMC-TC.

6.3 Недостатки текущей организации

Вместе все эти системы способны справляться со всеми задачами мониторинга, но при такой организации возникают проблемы в поддержке и изучении как минимум трёх систем мониторинга, что приводит к увеличению временных затрат на их обслуживание. Однако, на сегодняшний день был выявлен ряд других недостатков:

- Отсутствие единой точки входа для конфигурирования, просмотра и управления мониторингом;
- Отсутствие гибкой системы отчетов, позволяющей выводить данные мониторинга в виде файлов;
- Невозможность гибкой настройки визуального отображения параметров, поскольку не все системы используют ПО с открытым исходным кодом;
- Отсутствие единого и удобного дизайна веб-интерфейса.

Для устранения этих проблем принято решение разработать единую систему, которая бы справлялась со всеми необходиммыми задачами мониторинга и имела точку входа для сотрудников комплекса, представляя данные о состоянии комплекса в удобном виде и позволяла осуществлять гибкую настройку всей системы и её отдельных модулей.

7 Разработка системы мониторинга

7.1 Формирование требований к системе

Изначально необходимо точно сформулировать основные идеи, которые определяют функционал системы мониторинга. Грамотно составленные требования к проекту являются гарантией того, что он будет выполнен максимально качественно и будет иметь весь необходимый функционал.

Таким образом, к разрабатываемой системе мониторинга предъявляются следующие требования:

1. Модульность – система должна состоять из отдельных модулей, отвечающих за отдельную задачу мониторинга;
2. Расширяемость – функционал системы должен быть легко расширяемым;
3. Наличие развитой системы отчетов – существует необходимость в автоматическом формировании различных видов отчетов по состоянию комплекса за период (его загруженность, детализация параметров мониторинга и т.д.);
4. Эффективность – система должна быть оптимизированной, быстродействующей и способной осуществлять мониторинг всего комплекса без задержек;
5. Возможность дальнейшего развития системы;
6. Наличие хранилища для данных мониторинга;
7. Возможность объединения единиц мониторинга по группам;
8. Наличие единой точки входа (веб-интерфейса), с помощью которой сотрудник комплекса может выносить суждение о состоянии всей системы в целом или её отдельных частей и производить общую и детальную настройку системы;
9. Возможность настройки отображения данных на веб-интерфейсе;
10. Мониторинг в режиме реального времени;

11. Простой и интуитивно понятный интерфейс, доступность информации.

7.2 Выбор основного языка программирования

Для начала необходимо выбрать язык программирования, с помощью которого можно было бы реализовать систему, удовлетворяющую всем требованиям. Таким языком оказался Python.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ [6]. Он распространяется под свободной лицензией PythonSoftwareFoundationLicense, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Python на самом Python, что уменьшает затраты на изменения языка и постановку экспериментов над новыми возможностями.

Python – активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython [13].

Python – язык универсальный, он широко используется во всем мире для самых разных целее, в том числе для программирования Internet и Web приложений – серверных, клиентских, Web-серверов и серверов приложений. Python обладает богатой стандартной библиотекой, и еще более богатым набором модулей, написанных третьими лицами [14].

7.3 Преимущества и недостатки выбранного языка

Язык Python обладает рядом преимуществ и недостатков:

- Python – интерпретируемый язык программирования. С одной стороны, это позволяет значительно упростить отладку программ, с другой – обуславливает сравнительно низкую скорость выполнения;
- Динамическая типизация. В Python не надо заранее объявлять тип переменной, что очень удобно при разработке;
- Хорошая поддержка модульности. Вы можете легко написать свой модуль и использовать его в других программах;
- Встроенная поддержка Unicode в строках. В Python необязательно писать всё на английском языке, в программах вполне может использоваться ваш родной язык;
- Поддержка объектно-ориентированного программирования. При этом его реализация в python является одной из самых понятных;
- Автоматическая сборка мусора, отсутствие утечек памяти;
- Интеграция с C/C++, если возможностей python недостаточно;
- Понятный и лаконичный синтаксис, способствующий ясному отображению кода. Удобная система функций позволяет при грамотном

подходе создавать код, в котором будет легко разобраться другому человеку в случае необходимости. Также вы сможете научиться читать программы и модули, написанные другими людьми;

- Огромное количество модулей, как входящих в стандартную поставку Python, так и сторонних. В некоторых случаях для написания программы достаточно лишь найти подходящие модули и правильно их скомбинировать. Таким образом, вы можете думать о составлении программы на более высоком уровне, работая с уже готовыми элементами, выполняющими различные действия;
- Кроссплатформенность. Программа, написанная на Python, будет функционировать совершенно одинаково вне зависимости от того, в какой операционной системе она запущена. Отличия возникают лишь в редких случаях, и их легко заранее предусмотреть благодаря наличию подробной документации [15].

7.4 Архитектура системы мониторинга

Для удовлетворения всех требований разработана следующая архитектура системы мониторинга, изображенная на рисунке 6.

На данном рисунке представлено четыре составляющие системы мониторинга: ядро системы, объекты мониторинга, хранилище данных и точка входа (или веб-интерфейс). Стрелками указаны пути общению между составляющими системами. Веб-интерфейс, имея доступ на чтение, использует информацию из хранилища данных для отображения всей необходимой информации о мониторинге. Также веб-интерфейс имеет доступ на запись, поэтому с его помощью можно осуществлять настройку мониторинга. Ядро системы также имеет доступ к хранилищу для записи данных мониторинга и для получения необходимой информации, которая используется при организации процесса мониторинга. Между ядром системы и объектами

данных происходит обмен данными. Ядро запрашивает данные у объектов мониторинга, а затем получает их в ответ.

Рисунок 6 – Архитектура разработанной системы мониторинга

7.4.1 Ядро системы

Ядро системы располагается на отдельном сервере. Оно организует и осуществляет процесс мониторинга. Используя подключенные модули, оно получает данные с объектов мониторинга и записывает их в хранилище данных. К ядру можно подключать модули для решения различных задач мониторинга.

7.4.2 Модули

Отдельные компоненты системы, созданные для решения определенной задачи мониторинга, подключаются к системе, тем самым расширяя её функционал. В данной архитектуре в качестве примера представлено три модуля.

Каждый модуль состоит из демона-сборщика информации и агента, который устанавливается на счетные узлы. Агент устанавливается на объект мониторинга и отвечает на запросы демона, предоставляя необходимы данные по мониторингу.

7.4.3 Хранилище данных

Осуществляет хранение данных мониторинга и конфигурации системы. Оно предоставляет доступ на чтение и запись только ядру и веб-интерфейсу.

7.4.4 Объекты мониторинга

Объектами мониторинга в данном модуле являются сервера (или узлы), с которых можно получить информацию, используемую в качестве данных мониторинга. Узлы могут объединяться в группы.

7.4.5 Точка входа (веб-интерфейс)

Вычислительный кластер – сложный многокомпонентный объект, как в смысле состава оборудования, так и в плане используемого программного обеспечения. Количество внутренних состояний кластера определяется как моментальный снимок комбинаций компонентов, а также как последовательность изменений. В связи с этим очень важно иметь средства адекватной визуализации текущего состояния вычислительного кластера в виде компактной картинки, позволяющей быстро понять, что происходит. Это может быть полезно потребителям вычислительных ресурсов как для более точного планирования работы, так и непосредственно для обеспечения работоспособности.

С помощью веб-интерфейса наглядно и понятно представляются все данные мониторинга и осуществляется предоставление доступа к самой системе мониторинга для:

- наблюдения за текущим состоянием комплекса;
- конфигурации системы мониторинга;
- формирования сводных отчетов о состоянии комплекса по соответствующим параметрам мониторинга.

7.5. Проектирование системы мониторинга

7.5.1 Проектирование модуля загрузки узлов

7.5.1.1 Модуль загрузки узлов – это компонент системы мониторинга, который должен осуществлять мониторинг параметров по загрузке узлов. Для данного модуля был определен следующий список параметров, нуждающихся в мониторинге:

- Процент загрузки процессора;
- Процессорное время, затраченное на выполнение задач;
- Число процессоров;
- Статистические данные процессора;
- Использование оперативной памяти;
- Использование памяти подкачки;
- Разделы жесткого диска;
- Пространство диска;
- Статистика операций ввода/вывода на диске;
- Сетевой трафик;
- Сетевые сервисы;
- Сетевые интерфейсы;
- Время непрерывной работы сервера;

- Пользователи и их задачи;
- Загрузка процессов.

Как было сказано ранее, модуль должен состоять из демона-сборщика информации и агента. Из этого следует, что демон-сборщик должен обращаться к агенту с запросом данных по вышеперечисленным параметрам, а агент должен отвечать на эти запросы, предоставляя нужные данные.

7.5.1.2 Основу модуля мониторинга узлов составляют некоторые единицы мониторинга - параметры (метрики), например, загрузка процессора, загрузка сети и т.д. Так как список параметров является не конечным, то возможен вариант его дополнения новыми параметрами. Для удобства расширяемости, семантической определенности и поддержки структурности было решено создать плагиновую систему и разбить эти параметры на группы, названные плагинами. Каждый плагин отвечает за определённую область мониторинга. Плагины несёт в себе необходимую информацию и функционал как для демона-сборщика, так и для агента. Для агента это набор методов, позволяющих собирать данные с узла, а для демона-сборщика – конфигурационная информация по мониторингу, с помощью которой он осуществляет опрос данных у агента. Конфигурируя отдельный плагин, можно изменять модель обзора и опроса данных.

Основываясь на списке параметров, нуждающихся в мониторинге был создан список из следующих плагинов:

- CPU (плагин по мониторингу процессора);
- RAM (плагин по мониторингу системной памяти);
- HDD (плагин по мониторингу жесткого диска);
- NET (плагин по мониторингу сети);
- PROC (плагин по мониторингу процессов);
- USER (плагин по мониторингу пользователей).

Ниже приведена таблица, показывающая, в какой плагин входит тот или иной параметр мониторинга.

Таблица 1 – Параметры, входящие в плагины

Параметр	CPU	RAM	HDD	NET	PROC	USER
Процент загрузки процессора	+					
Процессорное время, затраченное на выполнение задач	+					
Число процессоров	+					
Статистические данные процессора	+					
Использование оперативной памяти		+				
Использование памяти подкачки		+				
Разделы жесткого диска			+			
Пространство диска			+			
Статистика операций ввода/вывода на диске			+			
Сетевой трафик				+		
Сетевые сервисы				+		
Сетевые интерфейсы				+		
Загрузка процессов					+	
Пользователи и их задачи						+

7.5.1.3 Для каждого плагина поставляются методы для сборки данных, реализованные с помощью модуля psutil. Их названия, выходные данные и принадлежность к плагину указана в таблице 2.

Таблица 2 – Методы сбора данных, поставляемые к плагинам

Метод	Плагин	Выходные данные	Описание метода
cpu_percent	CPU	*{'cpu_percent': data}*	Возвращает текущий процент загрузки процессора
cpu_count	CPU	*{'cpu_count': data}*	Возвращает число ядер процессора
cpu_stats	CPU	*{'cpu_stats': data}*	Возвращает статистику работы процессора с момента загрузки
swap_memory	RAM	*{'swap_memory': data}*	Возвращает информацию о swap-памяти: общей, доступной, использованной
disk_usage	HDD	*{'disk_usage': data}*	Возвращает информацию о памяти жесткого диска: общей, доступной, использованной
disk_partitions	HDD	*{'disk_partitions': data}*	Возвращает информацию о примонтированных разделах жесткого диска
disk_io_counters	HDD	*{'disk_io_counters': data}*	Возвращает статистику жесткого диска по операциям считывания и записи
net_io_counters	NET	*{'net_io_counters': data}*	Возвращает размер входящего и исходящего сетевого трафика с момента загрузки системы
net_if_addrs	NET	*{'net_if_addrs': data}*	Возвращает информацию об IP-адресе, соответствующего каждому сетевому интерфейсу
net_if_stats	NET	*{'net_if_stats': data}*	Возвращает информацию о каждой установленной сетевой карте
pids	PROC	*{'pids': data}*	Возвращает список идентификаторов всех работающих процессов

Окончание таблицы 2

Метод	Плагин	Выходные данные	Описание метода
procces_info	PROC	*{'procces_info': data}*	Возвращает информацию о процессе: идентификаторы процесса и его родителя, имя процесса, команда процесса, исполняемая директория и другие
users	USER	*{'users': data}*	Возвращает информацию о пользователях, работающих в системе

Благодаря плагиновой системе, мы можем:

- Подключить к системе мониторинга новую группу параметров мониторинга (например, связанные с пользователями);
- Обеспечить возможность отправки данных демону-сборщику (в плагине находятся методы, занимающиеся сбором информации с узла);
- Настроить отображение параметров в виде графиков, таблиц, схем (длявыполнение условия удобства использования системы настройка осуществляется через веб-интерфейс);
- Создать новый плагин для новой группы параметров или дополнить существующий.

Рисунок 7 – Плагиновая система

7.5.2 Выбор технологий и спецификации

При выборе технологий необходимо учитывать следующие вещи:

- Необходим двунаправленный канал связи между демоном сборщиком и агентом, то есть нужно создать интерфейс общения между ними – API;
- Данные, которые агент отправляет серверу должны быть специфицированы, то есть иметь определенный формат;
- Демон-сборщик запускается ядром системы и он опрашивает агента по каждому параметру с определенным периодом. Это значит, что необходимо осуществить асинхронные запросы и ответы, так как в один момент демон может запросить у агента сразу несколько параметров.

Учитывая вышеприведенные пункты было решено использовать протокол JSON-RPC.

JSON-RPC (JavaScriptObjectNotationRemoteProcedureCall – JSON-вызов удалённых процедур) – протокол удалённого вызова процедур, использующий JSON для кодирования сообщений. Это протокол, определяющий только несколько типов данных и команд. JSON-RPC поддерживает уведомления (информация, отправляемая на сервер, не требует ответа) и множественные вызовы.

JSON-RPC работает отсылая запросы к серверу, реализующему протокол. Клиентом обычно является программа, которой нужно вызвать метод на удалённой системе. Множество входных параметров может быть передано удалённому методу, как массив или объект. Метод также может вернуть множество выходных данных (это зависит от реализации). Удалённый метод вызывается отправлением запроса на удалённый сервер посредством HTTP или TCP/IP сокета.

Все передаваемые данные – простые объекты, сериализованные в JSON. Запрос – вызов определённого метода, предоставляемого удалённой системой.

Он должен содержать три обязательных свойства:

1. method – Строка с именем вызываемого метода;

2. params – Массив объектов, которые должны быть переданы методу, как параметры;

3. id – Значение любого типа, которое используется для установки соответствия между запросом и ответом;

Сервер должен отослать правильный ответ на каждый полученный запрос. Ответ должен содержать следующие свойства:

1. result – Данные, которые вернул метод. Если произошла ошибка во время выполнения метода, это свойство должно быть установлено в null.

2. error – Код ошибки, если произошла ошибка во время выполнения метода, иначе null.

3. id – То же значение, что и в запросе, к которому относится данный ответ [16].

К счастью, для Python существует сторонний модуль python-jsonrpc, позволяющий не только создать API для общения между сборщиком и агентом, но и организовать асинхронные вызовы, так как этот модуль использует tornado.

Tornado – расширяемый, неблокирующий веб-сервер и фреймворк. Tornado был создан для обеспечения высокой производительности и является одним из веб-серверов, способных обеспечить обслуживание порядка 10 тыс. соединений одновременно [17].

И, наконец, была разработана следующая простая спецификация для данных, отправляемых агентом и данных, которые записываются демоном-сборщиком.

Формат данных, отправляемых агентом – это JSON-объект, состоящий из одного ключа и значения:

```
{
   'data': param_data
}
```

‘data’ – это ключ, по которому можно получить param_data, который в свою очередь является сериализованными JSON-объектом. Он может иметь

любой формат: число, строка, массив чисел, массив строк, объект, массив объектов и т.д.

Формат данных, который записывается в базу выглядит следующим образом:

```
{
   'node': node_name,
   'plugin': plugin_name,
   'param': param _name,
   'data' : param_data,
   'timestamp': date_time
}
```

Это тоже JSON-объект, в котором:

- node_name – имя узла, с которого пришли данные;
- plugin_name – имя плагина, реализовавший возможность отправки этих и данных и показывающий принадлежность этих данных к определенной группе параметров;
- param_name – имя параметра, которым названы полученные данные;
- param_data – сами данные;
- timestamp – дата и время получения этих данных.

7.5.3 Организация хранилища данных

Хранилище данных представляет собой базу данных, в которой хранится вся необходимая информация. Для создания системы мониторинга, удовлетворяющей требованиям, необходимо:

- Выбрать СУБД;
- Разработать модель базы данных.

7.5.3.1 Выбор СУБД

В качестве основной СУБД используется MongoDB.

MongoDB - документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц.

MongoDB была выбрана по ряду причин:

- Простота использования.
- Эффективное хранение двоичных данных больших объёмов, например, фото и видео
- У MongoDB есть множество официальных драйверов для различных языков (в том числе Python).
- Динамичность и бесструктурность. Часто рекламируемым преимуществом документ-ориентированных баз данных является то, что они бесструктурны. Это делает их гораздо более гибкими, нежели традиционные реляционные базы данных
- Запись. Область, для которой MongoDB особенно подходит, – это логгирование. Есть два аспекта MongoDB, которые делают запись быстрой. Во-первых, можно отправить команду записи и продолжить работу, не ожидая её возврата и действительной свершившейся записи. Во-вторых, с появлением в версии 1.8 журналирования и некоторыми улучшениями, сделанными в версии 2.0, стало возможно контролировать поведение записи с учётом целостности данных. Эти параметры, в дополнение к тому, сколько серверов должны получить ваши данные, прежде чем запись будет считаться успешной, настраиваются на уровне отдельной записи, что даёт вам большую степень контроля над выполнением записи данных и их долговечностью.
- Ограниченная коллекция. Мы можем создать ограниченную коллекцию, и когда она достигнет указанного размера, старые документы начнут автоматически удаляться. Это избавляет нас от лишней работы, связанной с освобождением дискового пространства;
- Устойчивость. Одной из самых важных функций, добавленных в MongoDB 1.8, стало журналирование, благодаря чему стало возможным добиться устойчивости хранилища в пределах одного сервера.

- Обработка данных. Для большинства задач обработки данных MongoDB использует MapReduce. Одно из преимуществ MapReduce в том, что для работы с большими объёмами данных он может выполняться параллельно.

MongoDB в большинстве случаев способна стать заменой реляционной базе данных. Она намного проще и понятнее; быстрее работает и имеет меньше ограничений для разработчиков приложений [18].

В MongoDB хранится следующая информация:

- Настройки мониторинга;
- Информацию о группах узлов;
- Информацию об узлах;
- Информацию о доступных плагинах и их параметрах;
- Информацию о пользовательских настройках;
- Данные мониторинга;
- Данные сервера мониторинга.
- Для хранения информации об администраторах системы используется СУБД SQLite3, так как:
- Эти данные, в основном, статичны и их можно уместить в один файл;
- Благодаря использованию этой СУБД автоматические решается проблема с разграничением доступа к веб-интерфейсу.

7.5.3.2 Модель базы данных

Для составления моделей базы данных предоставляется модуль mongoengine, который позволяет описать модель. Модель базы данных MondoDB не является реляционной, то есть между документами нет связи, они изолированы друг от друга. Модель базы данных SQLite3 представлена на рисунке 8. Модель базы данных MongoDB представлена на рисунке 9.

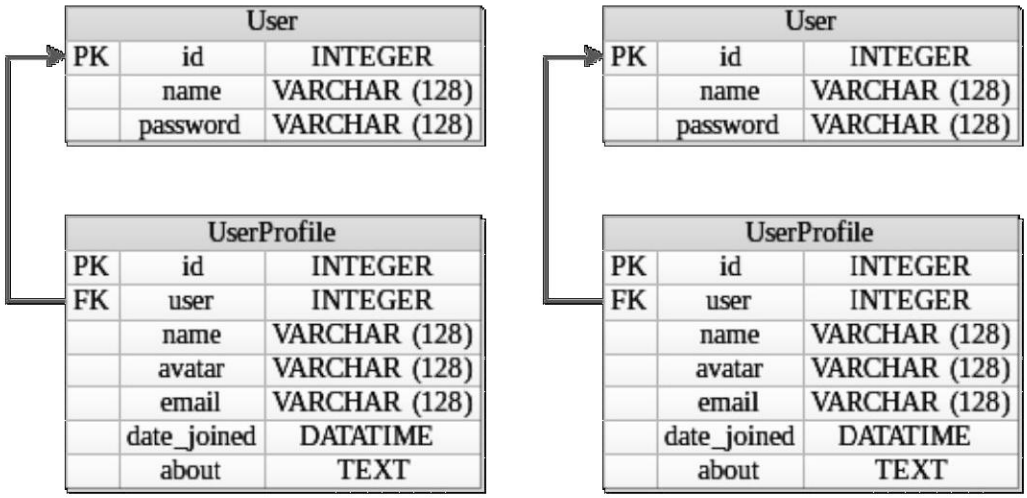

Рисунок 8 – Модель базы данных SQLite3

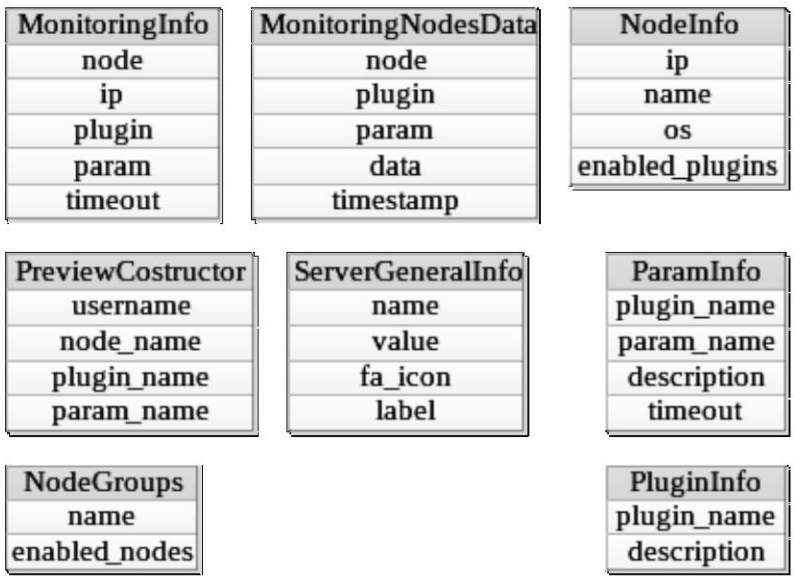

Рисунок 9 – Модель базы данных MongoDB

7.5.3.2.1 Конфигурационные данные мониторинга представлены в виде MongoDB-документа, в котором есть следующие поля:

- node – имя узла;
- ip – IP-адресс узла;
- plugin – имя плагина;
- param – имя параметра;
- timeout – период запроса данных.

7.5.3.2.2 Информация о группах узлов представлена в виде MongoDB-документа, в котором есть следующие поля:

- name – имя группы;
- enabled_nodes – узлы, входящие в группу с названием name.

7.5.3.2.3 Общая информация о сервере мониторинга представлена в виде MongoDB-документа, в котором есть следующие поля:

- name – имя единицы информации;
- value – соответствующая информация;
- fa_icon – иконка;
- label– метка.

7.5.3.2.4 Информация об узлах, подключенных к системе мониторинга, представлена в виде MongoDB-документа, в котором есть следующие поля:

- ip –IP-адрес узла;
- name – имя узла;
- os – операционная система узла;
- enabled_plugins – подключенные на узле плагины.

7.5.3.2.5 Информация о плагинах представлена в виде MongoDB-документа, в котором есть следующие поля:

- plugin_name – название плагина;
- description – описание плагина.

7.5.3.2.6 Информация о параметрах мониторинга представлена в виде MongoDB-документа, в котором есть следующие поля:

- plugin_name – название плагина;
- param_name –название параметра;
- description – описание параметра;
- timeout – период запроса данных по умолчанию для параметра;

7.5.3.2.7 Данные мониторинга представлены в виде MongoDB-документа, в котором есть следующие поля:

- node – узел;

- plugin – плагин;
- param – параметр;
- data – данные по данному параметру в данном узле;
- timestamp – время получения данных.

7.5.3.2.8 Информация о пользовательских графиках представлена в виде MongoDB-документа, в котором есть следующие поля:

- username – имя пользователя;
- node_name –узел;
- plugin_name – плагин;
- param_name – параметр

7.5.4 Веб-интерфейс

Веб-интерфейс является ключевым компонентом системы мониторинга, так как непосредственно через него осуществляется связь и общение между администратором и системой мониторинга. Именно поэтому реализация веб-интерфейса имеет высокую степень важности.

7.5.4.1 Назначение

На данный момент используя веб-интерфейс администратор имеет ряд возможностей.

Администратор может осуществлять настройки мониторинга:

- Добавить узел для мониторинга;
- Создавать группы;
- Добавить узел в группу;
- Подключить или отключить определенные плагины к узлу;
- Отключать или включать мониторинг определенных параметров узла;
- Остановить или возобновить мониторинг определенных параметров;

- Изменять период запроса данных о параметре (timeout).

Администратор может осуществлять настройки:

- Добавлять или удалять показ желаемых графиков на информационной панели системы (Dashboard);
- Осуществлять пользовательские настройки (например, смена пароля)

Администратор может видеть следующую информацию:

- Информационную панель с самой важной информацией (состояние мониторинга, оповещения и т.д.);
- Список плагинов и их описания;
- Список узлов и их описания;
- Графики, разделенные по узлам и плагинам или просто любой график;
- Таблицу мониторинга, на которой показано, что мониторится на данный момент;
- Таблицу групп серверов, на которой показано, какой узел к какой группе принадлежит.

7.5.4.2 Выбор технологий и инструментов

7.5.4.2.1 Для разработки веб-интерфейса был выбран веб-фреймворк Django.

Django – свободный программный каркас для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation [19].

7.5.4.2.2 Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Djangoуровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV) [19].

7.5.4.2.3 Некоторые возможности Django:

- ORM, API доступа к БД с поддержкой транзакций;
- встроенный интерфейс администратора, с уже имеющимися переводами на многие языки;
- диспетчер URL на основе регулярных выражений;
- расширяемая система шаблонов с тегами и наследованием;
- система кеширования;
- интернационализация;
- подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты;
- «generic views» – шаблоны функций контроллеров;
- авторизация и аутентификация, подключение внешних модулей аутентификации: LDAP, OpenID и проч.;
- система фильтров («middleware») для построения дополнительных обработчиков запросов, как например включённые в дистрибутив фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий;
- библиотека для работы с формами (наследование, построение форм по существующей модели БД);
- встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение [19].

7.5.4.2.4 Django проектировался для работы под управлением Apache с модулем modpython и с использованием PostgreSQL в качестве базы данных.

С включением поддержки WSGI, Django может работать под управлением FastCGI, mod_wsgi, uwsgi или SCGI на Apache и других серверах (lighttpd, nginx,…).

В настоящее время, помимо базы данных PostgreSQL, Django может работать с другими СУБД: MySQL, SQLite, MongoDB, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle.

В составе Django присутствует собственный веб-сервер для разработки. Сервер автоматически определяет изменения в файлах исходного кода проекта

и перезапускается, что ускоряет процесс разработки на Python. Но при этом он работает в однопоточном режиме и пригоден только для процесса разработки и отладки приложения [19].

7.5.4.2.5 Данный инструмент хорошо подходит для построения пользовательского интерфейса, который бы удовлетворил все требования к веб-интерефейсу.

7.5.4.3 Структура проекта

Структура проекта показана на рисунке 10.

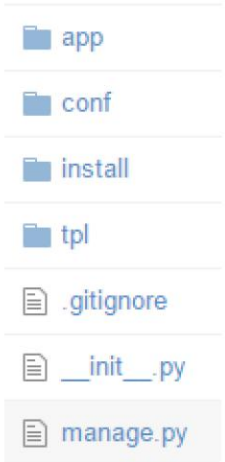

Рисунок 10 – Структура проекта

В данной структуре:

- app – директория, в которой находятся все приложения, их исходные коды, логика отображения страниц, диспетчеризация URL, директория со статичными файлами, а также другие файлы;
- conf – директория, в которой находятся файл *settings.py* с настройками всего django-проекта;
- tpl – директория, в которой находятся все файлы HTML-шаблонов, написанных на языке шаблонизации Django;

- install – директория, содержащая установочный скрипт и зависимости(модули, используемые в проекте);
- файл *manage.py*, с помощью которого исполняются различные команды (например, запуск сервера для разработки, создание и применение миграций, создание суперпользователя и другие).
- файлы*__init__.py* и *.gitignore* являются служебными.

В директории app, изображенной на рисунке 11, содержатся следующие приложения:

- user – приложение, которое организует пользовательскую систему и пользовательские страницы;
- api – приложение, которое осуществляет API для получения информации из хранилища данных;
- home – приложение, которое создает главные (домашние) страницы сайта (например, страница с информационной панелью);
- modules – приложение, содержащее в себе все подключенные модули, которые, в свою очередь, содержат приложения для показа графиков, таблиц, приложения для осуществления настроек и т.д.
- static – директория со статичными файлами (JavaScript-файлы, CSS-файлы, шрифты и картинки);
- файл *mongo_models.py*, в котором описаны модели базы данных, используемые в MongoDB (через эти модели осуществляется выборка, вставка, удаление и изменение данных);
- файл *urls.py* – корневой файл диспетчеризации URL-ов;
- файл wsgi.py – файл, осуществляющий запуск сервера в боевом режиме с использованием веб-серверов, таких как Apacheили Nginx;
- файл*__init__.py* является служебным.

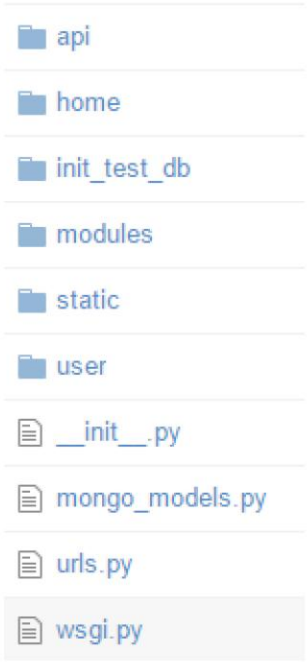

Рисунок 11 – структура приложения app

7.5.4.4 Описание приложений проекта

7.5.4.4.1 Приложение home

7.5.4.4.1.1 Приложение home имеет следующиеURL:

- /login/ – страница для авторизации пользователя. Обработку данной страницы осуществляет класс LoginFormView;
- /logout/ – страница для завершения сессии пользователя. Обработку данной страницы осуществляет класс LogoutView;
- / – домашняя страница сайта, на которой расположена информационная панель системы мониторинга. Обработку данной страницы осуществляет класс DashboardView;

7.5.4.4.1.2 Приложение homeимеет следующие классы отображения и логики страниц:

- LoginFormView – класс, унаследованный от стандартного класса Django“FormView” из пакета django.views.generic. Следуя из названия, этот

класс реализует страницу с формой отправки данных на сервер. В качестве формы указана форма “AuthenticationForm”, описывающая поля для ввода логина и пароля, из пакета django.contrib.auth.forms;В классе указан шаблон для отображения страницы “site/index/login.html”. Данный шаблон, как и любые другие, написан с использование шаблонизатора Django “DjangoTemplateLanguage”. Функция get, унаследованная от FormView, отвечает за формирование и отправку html-страницы клиенту. Эта функция срабатывает при GET-запросе на URL /login/.

- LogoutView – класс, унаследованный от стандартного класса Django “View” из пакета django.views.generic.base В данном классе переопределен метод get, который принимает на вход аргумент request, в котором находится вся информация о запросе. Данный метод осуществляет выход пользователя из системы и перенаправляет пользователя на URL /login/;
- DashboardView – класс, унаследованный от стандартного класса Django“TemplateView” из пакета django.views.generic. Согласно названию, данный класс отображает страницу по указанному шаблону, а именно “site/index/new_dashboard.html”. В данном классе переопределен метод get_context_data, в котором задаются переменные контекста, которые используются в шаблоне посредством языка шаблонов Django. Среди этих переменных присутсвуют: server_info (массив из словарей, информация о сервере мониторинга), monitoring_states (массив из словарей, информация о состоянии мониторинга), notifications(массив из словарей, уведомления о различных событиях), plugins(массив из словарей, информация о плагинах), nodes (массив из словарей, информация об узлах), groups(массив из словарей, информация о группах серверов).

7.5.4.4.1.3 Приложение homeимеет следующие шаблоны:

- /site/index/login.html – шаблонсформойввода логина и пароля для прохождения авторизации;
- /site/index/dashboard.html – шаблон с информационной панелью системы мониторинга.

7.5.4.4.2 Приложение user

7.5.4.4.2.1 Приложение userимеет следующиеURL:

- /user/ и /user/profile/ – страницы с профилем пользователя, на которой расположена информация о пользователе и меню для доступа к настройкам. Обработку данной страницы осуществляет класс UserProfileView;
- /user/settings/user/ – страница для пользовательских настроек (смена пароля, адреса электронной почты, аватара). Обработку данной страницы осуществляет класс UserSettingsFormView;
- /user/settings/dashboard/ – страница для настроек персонального отображения (выбор палитры цветов, выбор графиков, отображающихся на информационной панели) Обработку данной страницы осуществляет класс DashboardSettingsFormView;
- /user/settings/monitoring/ – страница для настроек мониторинга (создание нового узла, создание новой группы, удаления узла, удаление группы, изменение времени повторного опрашивания параметра мониторинга и другие). Обработку данной страницы осуществляет класс MonitoringSettingsFormView.

7.5.4.4.2.2 Приложение userимеет следующие классы отображения и логики страниц:

- UserProfileView – класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является “site/user/profile.html”. Также к этому классу добавлен класс-примесь UserProfileMixin, который устанавливает контекстную переменную user_profile. В этой переменной хранится пользовательская информация.
- MonitoringSettingsFormView – класс, унаследованный от стандартного класса Django“FormView” из пакета django.views.generic. Этот класс реализует страницу с формой отправки данных на сервер. В качестве формы указана форма “MonitoringSettingsForm”, описывающая поля с операциями по настройке мониторинга, из пакета app.user.config.forms. В классе указан шаблон

для отображения страницы “site/user/settings.html”. На данной странице пользователь может выбрать одну из следующих операций:

1. add_node – создать узел, подключая к нему нужные плагины и назначая ему имя и IP-адрес;
2. add_server_group – создать группу узлов;
3. del_server_group – удалить группу узлов;
4. add_node_to_group – добавить выбранный узел к группе узлов;
5. del_node_from_group – удалить выбранный узел из группы узлов;
6. add_node_to_monitor – начать мониторинг выбранного узла;
7. stop_to_monitor_node – прекратить мониторинг выбранного узла;
8. stop_to_monitor_param – прекратить мониторинг выбранного параметра;
9. change_param_timeout – изменить время повторного запроса данных о выбранном параметре.

- UserSettingsFormView – класс, унаследованный от стандартного класса Django“FormView” из пакета django.views.generic. Этот класс реализует страницу с формой отправки данных на сервер. В качестве формы указана форма “UserSettingsForm”, описывающая поля с операциями по настройке пользовательских данных, из пакета app.user.config.forms. В классе указан шаблон для отображения страницы “site/user/user_settings.html”. На данной странице пользователь может выбрать одну из следующих операций:

1. change_password – изменить пароль;
2. change_email – изменить адрес электронной почты;
3. change_avatar – изменить аватар.

- DashboardSettingsFormView – класс, унаследованный от стандартного класса Django “FormView” из пакета django.views.generic. Этот класс реализует страницу с формой отправки данных на сервер. В качестве формы указана форма “DashboardSettingsForm”, описывающая поля с операциями по настройке данных отображения веб-интерфейса, из пакета app.user.config.forms. В классе

указан шаблон для отображения страницы “site/user/dashboard_settings.html”. На данной странице пользователь может выбрать одну из следующих операций:

1. change_pallete – изменить цветовую схему веб-интерфейса;
2. add_preview_graph – добавить график выбранного параметра для его отображения на информационной панели;
3. del_preview_graph – убрать график выбранного параметра из отображения на информационной панели.

7.5.4.4.2.3 Приложение userимеет следующие шаблоны:

- profile.html – шаблонспрофилем пользователя;
- user_settings.html – шаблон с формой для пользовательских настроек.
- monitoring_settings.html – шаблон с формой для настроек мониторинга;
- dashboard_settings.html – шаблон с формой для настроек отображения веб-интерфейса.

7.5.4.4.3 Приложение modules

7.5.4.4.3.1 Приложение modules имеет URL /modules/ – страница со списком подключенных модулей системы мониторинга. Обработку данной страницы осуществляет класс ModulesView.

7.5.4.4.3.2 Приложение modules имеет класс отображения и логики страниц ModulesView. Это класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является “site/modules/index.html”.

7.5.4.4.3.3 Приложение modules имеет шаблон index.html – шаблон со списком модулей системы мониторинга.

7.5.4.4.4 Приложение modules.monitoring_nodes

7.5.4.4.4.1 Приложениеmodules.monitoring_nodes имеет URL /modules/monitoring-nodes/ – страница с полной информацией по модулю

мониторинга узлов. На ней содержатся списки плагинов, узлов и групп узлов. Обработку данной страницы осуществляет класс MonitoringView;

7.5.4.4.4.2 Приложение modules имеет класс отображения и логики страниц MonitoringView – класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является “site/modules/monitoring_nodes/index.html”.В данном классе переопределен метод get_context_data, в котором задаются переменные контекста, которые используются в шаблоне посредством языка шаблонов Django. Среди этих переменных присутствуют: plugins (массив из словарей, информация о плагинах), nodes (массив из словарей, информация об узлах) и groups (массив из словарей, информация о группах серверов).

7.5.4.4.4.3 Приложение modules имеет шаблон index.html – шаблон со страницей с подробной информацией о модуле загрузки узлов. На ней содержатся таблицы с плагинами(имя плагина, список параметров и описание), узлов (имя узла, описание, подключенные плагины) и групп узлов (имя группы и входящие в неё узлы).

7.5.4.4.5 Приложение modules.monitoring_nodes.graphs

7.5.4.4.5.1 Приложение modules.monitoring_nodes.graphs имеет следующие URL:

- /modules/monitoring-nodes/graphs/node-graphs/<node_name> – страницасовсемиграфикаминавыбранномузле. Обработку данной страницы осуществляет класс NodeGraphsView;
- /modules/monitoring-nodes/graphs/plugin-graphs/<plugin_name> – страница со всеми графиками по выбранному плагину. Обработку данной страницы осуществляет класс PluginGraphsView;
- /modules/monitoring-nodes/graphs/param-graphs/<param_name> – страница со всеми графиками с выбраннымпараметром. Обработку данной страницы осуществляет класс ParamGraphsView;

- /modules/monitoring-nodes/graphs/node-graphs – страницасвыборомграфиком. Обработку данной страницы осуществляет класс GraphsView;
- /modules/monitoring-nodes/graphs/node-graphs/<node_name>/<plugin_name>/<param_name>/ – страница с одним графиком выбранного параметра по фильтрам node_name, plugin_name и param_name. Обработку данной страницы осуществляет класс SingleGraphView.

7.5.4.4.5.2 Приложение modulesимеет следующие классы отображения и логики страниц:

- GraphsView – класс, унаследованный от стандартного класса Django“FormView”из пакета django.views.generic. Этот класс реализует страницу с формой отправки данных на сервер. В качестве формы указана форма “ ShowGraphForm”, описывающая поля-фильтры по узлу, плагину и параметру, из пакета app.modules.monitoring_nodes.graphs.forms. Шаблоном для данного класса является "site/modules/monitoring_nodes/graphs/index.html". На данной странице пользователь может не только выбрать один график параметра для отображения, но и может выбрать просмотр графиков по узлу, плагину и параметру;
- SingleGraphView – класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является 'site/modules/monitoring_nodes/graphs/single_graph.html'. В данном классе переопределен метод get_context_data, в котором задаются переменные контекста, которые используются в шаблоне посредством языка шаблонов Django. Среди этих переменных присутствуют: id (идентификатор графика), node_name (имя узла), plugin_name (имя плагина) и param_name (имя параметра). По этим данным на странице происходит отрисовка графика при помощи JavaScript.
- NodeGraphsView – класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является "site/modules/monitoring_nodes/graphs/graphs_node.html”. В

данном классе переопределен метод get_context_data, в котором задается переменная контекста graphs_info (массив из словарей, информация о графиках). Внутри находятся словари с данными графика (идентификатор, имя узла, плагина и параметра);

- PluginGraphsView – класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является "site/modules/monitoring_nodes/graphs/graphs_plugin.html”. В данном классе переопределен метод get_context_data, в котором задается переменная контекста graphs_info (массив из словарей, информация о графиках). Внутри находятся словари с данными графика (идентификатор, имя узла, плагина и параметра);
- ParamGraphsView – класс, унаследованный от стандартного класса Django “TemplateView” из пакета django.views.generic. Шаблоном для данного класса является "site/modules/monitoring_nodes/graphs/graphs_param.html”. В данном классе переопределен метод get_context_data, в котором задается переменная контекста graphs_info (массив из словарей, информация о графиках). Внутри находятся словари с данными графика (идентификатор, имя узла, плагина и параметра);

7.5.4.4.5.3 Приложение modules.monitoring_nodes.graphsимеет следующие шаблоны:

- index.html – шаблон со страницей, на которой пользователь может перейти к просмотру графиков мониторинга со следующими фильтрами: по узлу, по плагину, по параметру и по всем трём сразу;
- single_graph.html – шаблон, накотором отображается график с фильтром по узлу, плагину и параметру. К этому шаблону подключены js-скрипты, которые осуществляют отрисовку графиков;
- graphs_node.html – шаблон, на котором отображаются графики с фильтром по узлу. К этому шаблону подключены js-скрипты, которые осуществляют отрисовку графиков;graphs_plugin.html – шаблон, на котором

отображаются графики с фильтром по плагину. К этому шаблону подключены js-скрипты, которые осуществляют отрисовку графиков;

- graphs_plugin.html – шаблон, на котором отображаются графики с фильтром по плагину. К этому шаблону подключены js-скрипты, которые осуществляют отрисовку графиков;
- graphs_param.html – шаблон, на котором отображаются графики с фильтром по параметру. К этому шаблону подключены js-скрипты, которые осуществляют отрисовку графиков.

7.5.4.5 Визуализация веб-интерфейса

7.5.4.5.1 Вид страницы с информационной панелью изображен на рисунке 12.

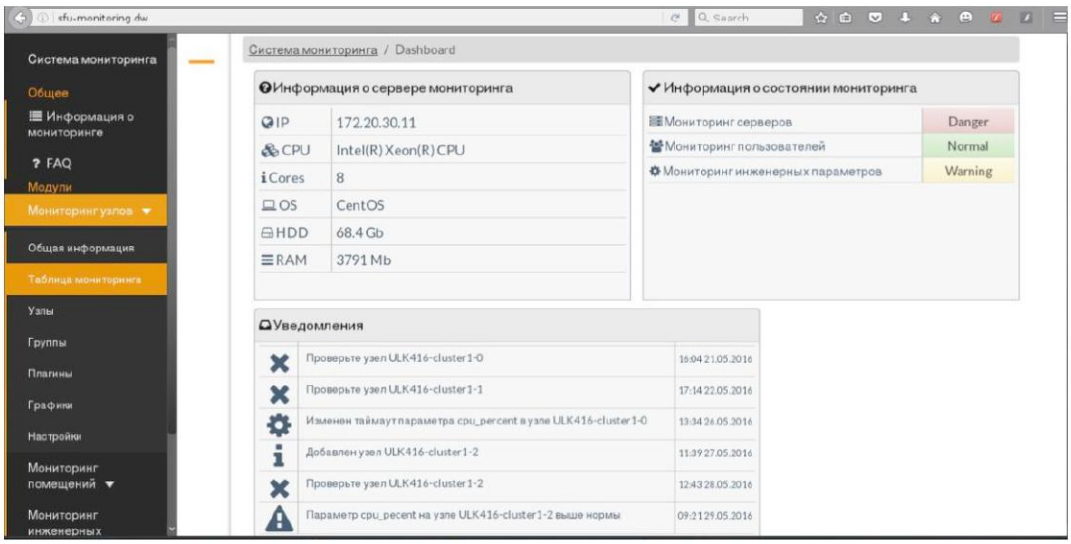

Рисунок 12 – Вид страницы с информационной панелью

7.5.4.5.2 Вид страницы с общей информацией по модулю мониторинга узлов изображен на рисунке 13.

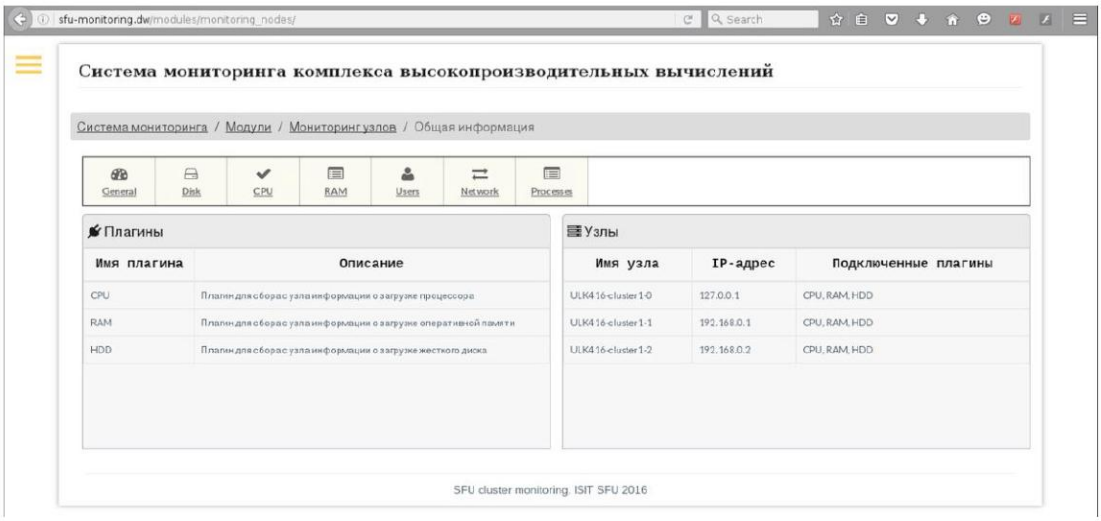

Рисунок 13 – Вид страницы с общей информацией по модулю мониторинга узлов

7.5.4.5.3 Вид страницы с древовидной таблицей с информацией о текущем производимом мониторинге. Из данной страницы можно перейти к нужному графику.

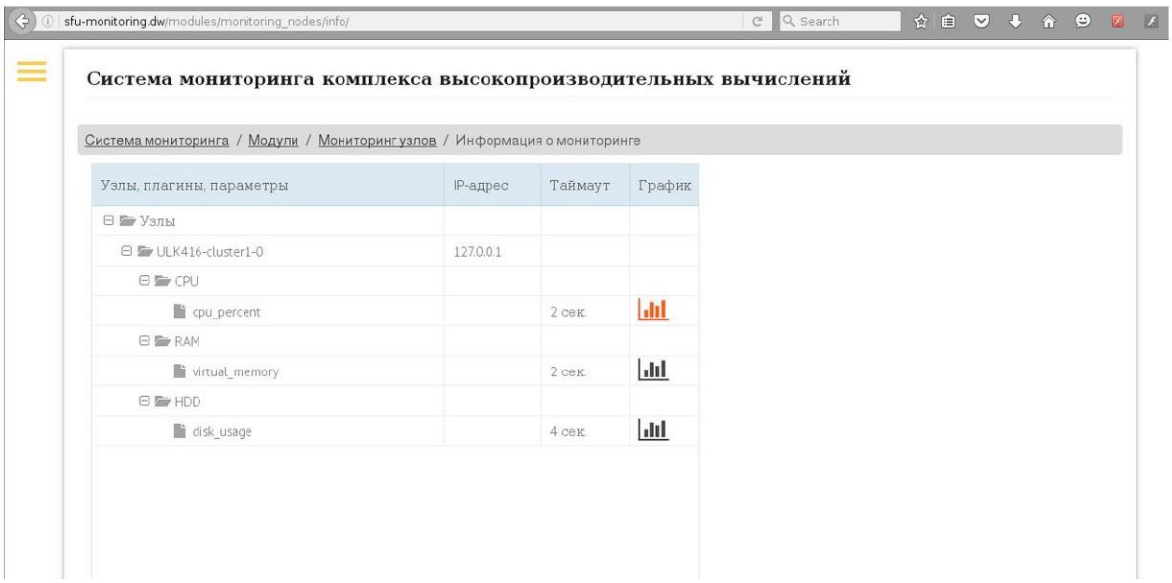

Рисунок 14 – Вид страницы с древовидной таблицей с информацией о текущем производимом мониторинге

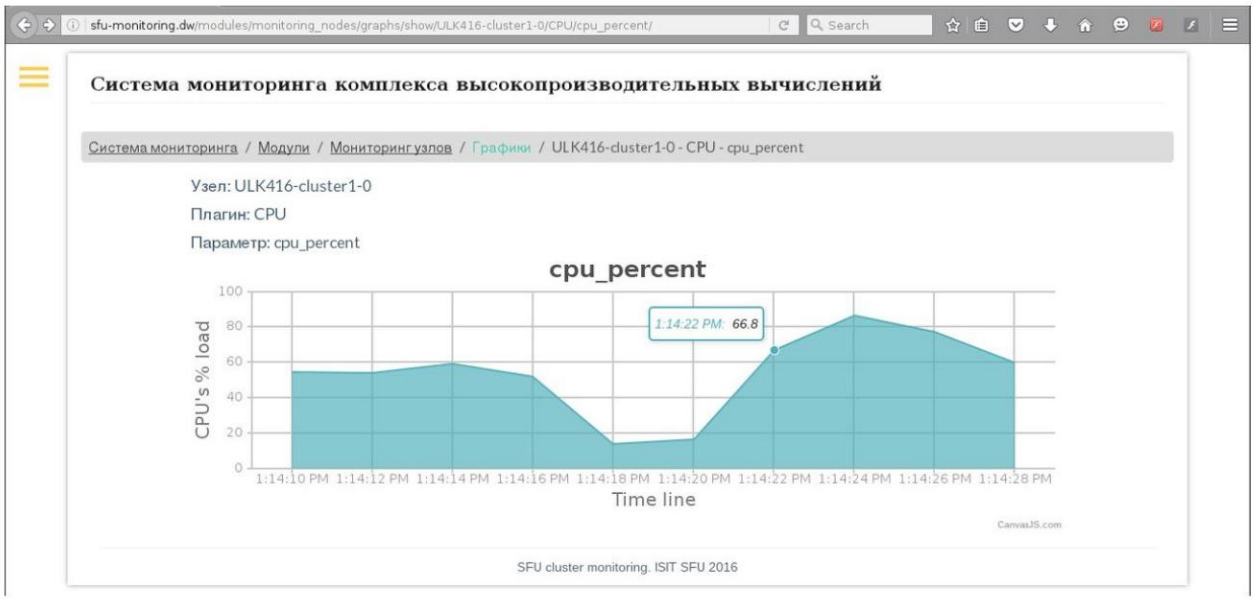

Рисунок 15 – Вид страницы с графиком параметра, выбранного в таблице с информацией о текущем производимом мониторинге

7.5.4.5.4 Вид страницы с выбором просмотра графиков модуля мониторинга узлов.

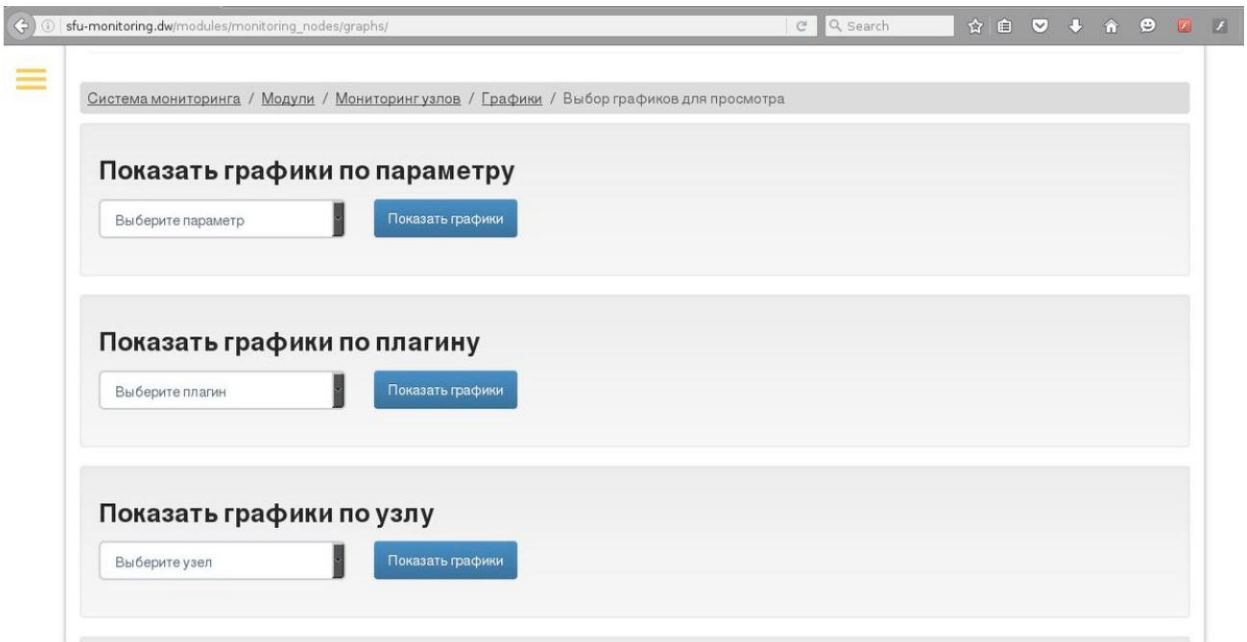

Рисунок 16 – Вид страницы с выбором просмотра графиков модуля мониторинга узлов

7.5.4.5.4 Вид страницы с выбором настроек модуля мониторинга узлов представлен на рисунках 17-19.

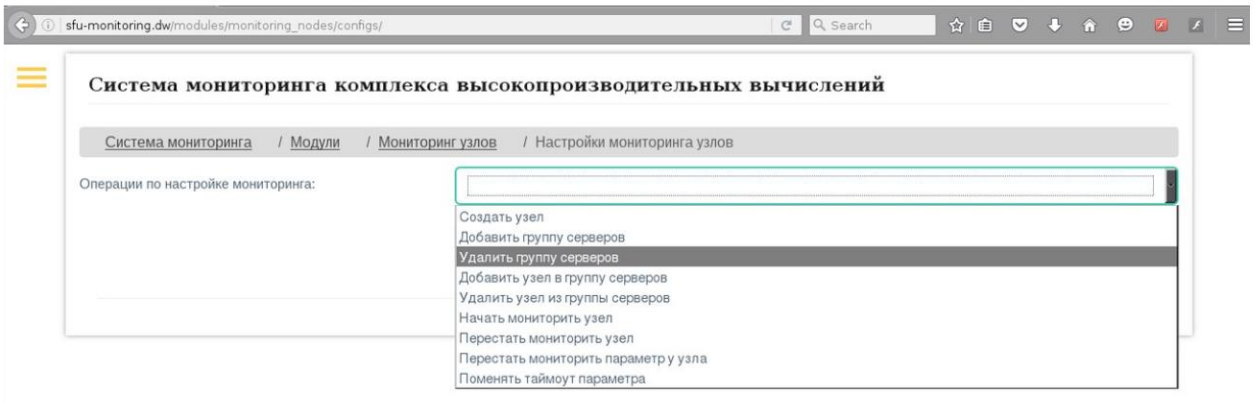

Рисунок 17 – Вид страницы с выбором настроек модуля мониторинга узлов (все настройки)

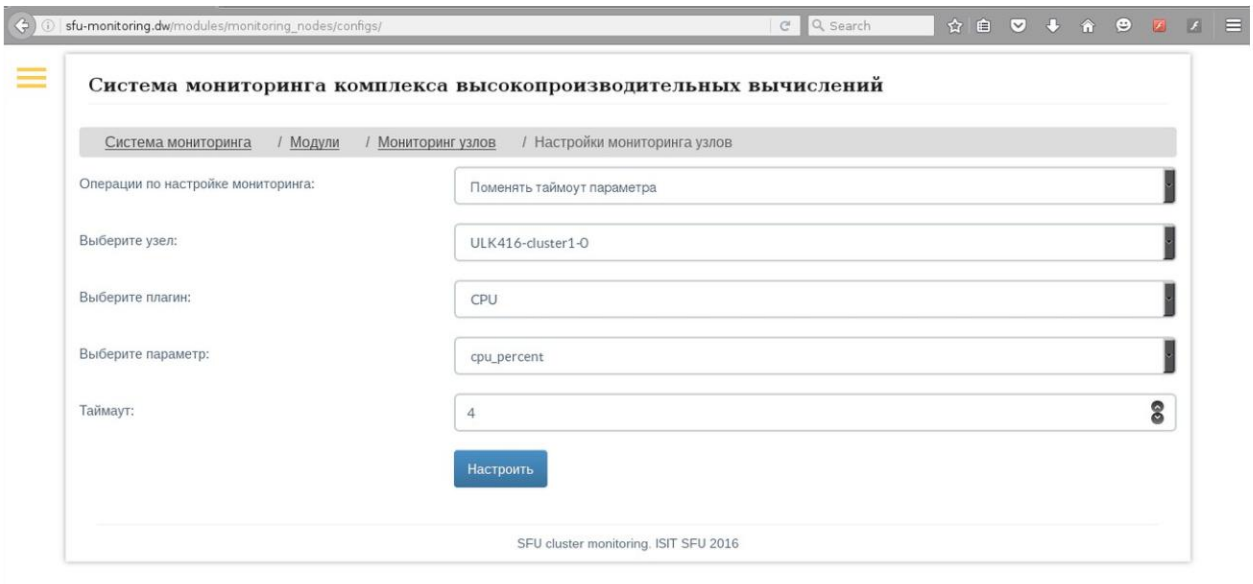

Рисунок 18 – Вид страницы с выбором настроек модуля мониторинга узлов (настройка смены периода опрашивания узла)

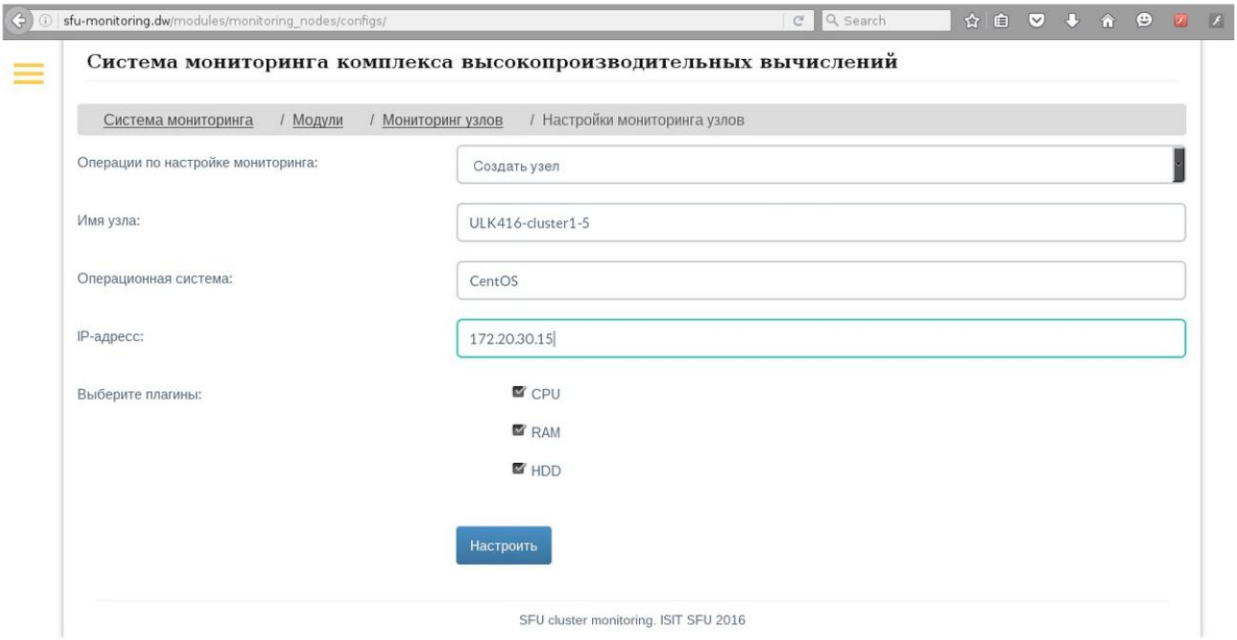

Рисунок 19 – Вид страницы с выбором настроек модуля мониторинга узлов (настройка по созданию узла)

ЗАКЛЮЧЕНИЕ

В настоящее время система проходит этап тестирования и проверки работоспособности на оборудовании суперкомпьютерного комплекса СФУ. Система имеет всего лишь один модуль по мониторингу загрузке узлов и обеспечивает весь его необходимый функционал. В дальнейшем функционал данного модуля будет улучшаться и расширяться, будут реализованы новые модули. Исходный код системы расположен в репозитории на github.com, благодаря чему система может получить поддержку open-source сообщества и в стезе применения выйти за рамки комплекса высокопроизводительных вычислений СФУ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Масленникова, Н. Ю. Понятие и сущность мониторинга с позиции системного подхода / Масленникова Н. Ю., Слинкова О.К. // ScienceTime. — 2014. — № 6 — С.110-121.

2. Мониторинг [Электронный ресурс] : Материал из Википедии — свободной энциклопедии, Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: http://ru.wikipedia.org/?oldid=77804864

3. Боровкова, Т. И. Мониторинг развития системы образования : учеб. пособие / Боровкова Т. И., Морев И. А. — Владивосток: Изд-во Дальневосточного университета, 2004. — 150 с.

4. Ganglia [Электронный ресурс] : Материал из Википедии — свободной энциклопедии, Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: http://ru.wikipedia.org/?oldid=77839373

5. Ganglia (Software) [Электронный ресурс] : Материал из Википедии — свободной энциклопедии, Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: https://en.wikipedia.org/w/index.php?title=Ganglia_(software)&oldid=714365240

6. IBM Director [Электронный ресурс] : Материал из Википедии — свободной энциклопедии, Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: http://ru.wikipedia.org/?oldid=69561560

7. Nagios – мониторинг сети [Электронный ресурс]: Материал из Блога "AmigosTeam" // Команда "AmigosTeam" — Режим доступа: http://amigosteam.ru/blog/item/10-nagios

8. ZenossCore – мониторинг ИТ-инфраструктур [Электронный ресурс]: Материал с портала «pro-spo.ru» про новости информационных технологий // Портал «pro-spo.ru» — Режим доступа: http://pro-spo.ru/inetl/872-zenoss-core

9. Zabbix [Электронный ресурс] : Материал из Википедии — свободной энциклопедии, Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: http://ru.wikipedia.org/?oldid=78444354

10. Zabbix – мониторинг сети [Электронный ресурс]: Материал из Блога "AmigosTeam" // Команда "AmigosTeam" — Режим доступа: ;amigosteam.ru/blog/item/11-zabbix

11. Система контроля распределительных шкафов CMC-TC [Электронный ресурс] : Материал с портала «pns.by» // Портал «pns.by» — Электрон.дан. — г. Минск: ЗАО «Профессиональные сетевые системы», 2016. — Режим доступа: http://pns.by/products/brand/rittal/cmc/

12. Центр высокопроизводительных вычислений СФУ [Электронный ресурс] : Материал с сайта кафедры «Высокопроизводительные вычисления» // Сайт кафедры «Высокопроизводительные вычисления» — Электрон. дан. — г. Красноярск: ФГАОУ ВПО СФУ, 2016. — Режим доступа: //cluster.sfu-kras.ru/page/supercomputer/

13. Python [Электронный ресурс] : Материал из Википедии — свободной энциклопедии, Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: http://ru.wikipedia.org/?oldid=77643549

14. Язык программирования Python / Г. Россум, Ф.Л.Дж. Дрейк, Д.С. Откидач, М. Задка, М. Левис, С. Монтаро, Э.С. Реймонд, А.М. Кучлинг, М.-А. Лембург, К.-П. Йи, Д. Ксиллаг, Х.Г. Петрилли, Б.А. Варсав, Дж.К. Ахлстром, Дж. Роскинд, Н.Шеменор, С. Мулендер — Д.С. Откидач, 2001 – 454 с.

15. Язык программирования Python 3 для начинающих и чайников[Электронный ресурс] : Материал с портала «Python 3 для

начинающих» // Портал «Python 3 для начинающих» — Электрон.дан. — 2012-2016. — Режим доступа: http://pythonworld.ru/

16. JSON-RPC [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 74984573, сохранённая в 11:08 UTC 8 декабря 2015 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: http://ru.wikipedia.org/?oldid=74984573

17. Tornado [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 75657885, сохранённая в 20:48 UTC 9 января 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: http://ru.wikipedia.org/?oldid=75657885

18. Karl Seguin, The Little MongoDB Book: Электроннаякнига / Karl Seguin. — 2011-2016. — 29 с.

19. Django [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 78442098, сохранённая в 02:03 UTC 19 мая 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон.дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: http://ru.wikipedia.org/?oldid=78442098

ПРИЛОЖЕНИЕ А

Сертификат участия на Международной научной конференции студентов, аспирантов и молодых ученых «Молодежь и наука: проспект свободный»


Институт космических и информационных технологий СФУ

Сертификат участия

выдан

Макарову Дмитрию Сергеевичу,
студенту гр. КИ12-09Б

на Международной научной конференции студентов,
аспирантов и молодых ученых
«Молодежь и наука: проспект свободный»

Секция «Вычислительная техника и информационные сети и системы»

Директор ИКИТ Г.М. Цибульский

20 апреля 2016 года