УДК 004.415.2, 004.588

АРХИТЕКТУРА И ПУТИ РЕАЛИЗАЦИИ СИСТЕМЫ ЛОКАЛЬНОГО МОНИТОРИНГА РЕСУРСНОГО ЦЕНТРА

Кореньков Владимир Васильевич[1], Дмитриенко Павел Владимирович[2]

[1]Кандидат физико-математических наук, профессор;
ГОУ ВПО «Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: korenkov@cv.jinr.ru.

[2]Инженер-программист;
Объединенный институт ядерных исследований;
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, 6;
e-mail: orelnotre@mail.ru.

Функционирование сложной распределенной вычислительной системы и входящих в ее состав ресурсных центров невозможно без качественной системы мониторинга, дающей подробную картину функционирования и производительности её элементов, своевременно оповещающей о сбоях и позволяющей проводить комплексный анализ работы системы. В статье рассмотрена информационная модель взаимодействия компонент системы мониторинга, предложена архитектура, реализующая принципы универсальности, расширяемости и адаптивности; рассмотрено решение задач, возникших в процессе построения системы мониторинга ЦИВК ОИЯИ.

Ключевые слова: распределенные вычислительные системы, мониторинг, Nagios.

ARCHITECTURE AND WAYS OF IMPLEMENTING THE SYSTEM FOR MONITORING A LOCAL RESOURCE CENTER

Korenkov Vldimir[1], Dmitrienko Pavel[2]

[1]Candidate of Science in Physics and Mathematics, professor;
Dubna International University of Nature, Society, and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: korenkov@cv.jinr.ru.

[2]Engineer-programmer;
Joint institute for nuclear researches;
141980, Moscow reg., Dubna, Joliot-Curie, 6;
e-mail:orelnotre@mail.ru.

Operation of complex distributed computing system and its member resource centers is impossible without quality monitoring system, which gives a detailed picture of the functioning and performance of its elements, timely warns of failure and allows a comprehensive analysis of the system. The article presents the information model of the interaction of monitoring system components and proposes architecture, which implements the principles of universality, extensibility and adaptability. The solving of problems encountered in constructing a system for monitoring the CICC JINR also discussed.

Keywords: distribited computing systems, monitoring, Nagios.

Развитие современных научных исследований во многих областях человеческого знания требует совместной работы многих организаций по обработке большого объема данных в относительно короткие сроки. Для этого может быть недостаточно вычислительных мощностей одного суперкомпью-

терного центра, что подразумевает необходимость объединения мощности множества географически распределенных вычислительных систем (РВС). Наиболее динамично развивающиеся виды таких систем:

- грид, компьютерная инфраструктура нового типа, обеспечивающая глобальную интеграцию информационных и вычислительных ресурсов;
- облачные вычисления (компьютерные ресурсы и мощности, предоставляемые пользователю как интернет-сервис);
- вычислительные и отказоустойчивые кластеры.

Функционирование сколько-нибудь сложной РВС невозможно без качественной системы мониторинга, дающей подробную картину функционирования и производительности её элементов, своевременно оповещающей о сбоях и позволяющей проводить комплексный анализ работы системы. Можно выделить несколько «точек зрения» на объекты мониторинга РВС. Так, для грид-систем наиболее широкий взгляд свойственен глобальному мониторингу с отображением состояний и взаимодействия всех ресурсных центров, активности пользователей в реальном времени (например, на географической карте). Более локализованный с функциональной точки зрения информационный «срез» предлагает мониторинг отдельной виртуальной организации (динамичного целевого объединения пользователей, ресурсов и служб) – здесь важна информация об исполнении текущих задач, их распределении между отдельными сайтами, взаимоотношениях отправителей и исполнителей. Максимально локализованный (уже и с географической точки зрения) «срез» – это локальный мониторинг, предоставляющий данные о состоянии служб, поставщиков и ресурсов, входящих в состав отдельного сайта (стабильного уникально идентифицируемого набора служб, поставщиков и ресурсов).

В соответствии со спецификой контроля и обслуживания объекты локального мониторинга можно распределить по трём уровням. На нижнем уровне осуществляется сбор и отображение данных об отдельных узлах, их аппаратном обеспечении и операционных системах; доступность их по сети, загруженность процессоров и заполненность дисковых разделов; источники бесперебойного питания (ИБП); температурный режим указанных объектов. На сетевом уровне рассматривается структура локальной сети, состояние памяти и загрузки процессоров коммутаторов, характеристики их портов; состояние внешнего канала и доступность важнейших сетей. Верхний уровень – это контроль служб, предоставляемых конечным пользователям.

Существует ряд частных решений данной задачи для отдельных классов систем, различающихся по области охватываемых ресурсов (сетевой мониторинг, анализ протоколов, диагностика и управление конкретными устройствами, grid-специфичный мониторинг); по используемым технологиям (SNMP-мониторинг, агентный мониторинг); по условиям использования (коммерческие, условно-бесплатные, GPL). Развитые коммерческие продукты, как правило, не рассматриваются в качестве решений для крупных научных, национальных, а также и учебных проектов, в силу дороговизны и закрытости программного кода.

Актуальной представляется задача разработки открытой архитектуры, методики построения и рабочего прототипа универсального инструмента мониторинга вычислительных ресурсов, не ограниченного некоторой однотипной системой или сетевыми параметрами, охватывающего все три уровня мониторинга и учитывающего зависимости между ними, базирующегося на свободном программном обеспечении. Такой программный комплекс должен осуществлять общее наблюдение за интересующими объектами различной природы, извещать о сбоях и предоставлять средства для их анализа, в удобной форме представлять данные о функционировании системы. Изложенные в статье положения были использованы при разработке прототипа системы локального мониторинга Центрального информационно-вычислительного комплекса (ЦИВК) Объединенного Института Ядерных Исследований (ОИЯИ), а затем и рабочей версии системы.

Структура системы локального мониторинга

Применение метода функциональной декомпозиции позволило выделить следующие составные части, в той или иной степени характерные для любой системы мониторинга:

1) *Подсистема сбора данных* – осуществляет с заданной регулярностью опрос объектов, подлежащих мониторингу, для получения исследуемых значений. Может также включать в себя первчиный анализ полученных данных с целью, например, квалификации полученных значений как нормальных, требующих вмешательства оператора либо критических.

2) *Подсистема хранения* – отвечает за накопление, хранение, архивацию данных о результатах проверок. Включает компоненты для работы с базами данных (БД) или иными репозиториями, программные средства усреднения значений за некоторый отрезок времени для уменьшения объема хранимой информации и т.п.

3) *Подсистема анализа данных* – включает компоненты, производящие исследования данных, накопленных системой, поиск закономерностей, сбор статистики и тому подобные операции.

Эти подсистемы решают задачи, относящиеся к *фоновому мониторингу*, подразумевающему систематическое долговременное накопление и анализ данных о работе объектов.

4) *Подсистема оповещения* – отвечает за уведомление лиц, ответственных за функционирование проверяемых объектов о нештатных ситуациях и иных значимых событиях, возникающих в системе.

5) *Подсистема вывода* – отвечает за представление информации о работе системы и результатов проверок в виде, удобном для восприятия пользователем. Для взаимодействия с конечным пользователем, безусловно, необходим развитый веб-интерфейс, предоставляющий удобную навигацию между различными типами отчетов и сводок о состоянии обследуемых объектов.

6) *Подсистема коррекции* – в том случае, если предусмотрена возможность выполнения системой некоторых действий для устранения возникших нештатных ситуаций, данная подсистема будет включать компоненты для выбора и осуществления подходящих действий в соответствии с типом проблемы и другими параметрами. Логично тесное взаимодействие ее с подсистемой анализа данных.

Эти подсистемы ориентированы на *оперативный мониторинг*, направленный на оценку текущей работоспособности и эффективности работы исследуемых объектов и немедленную реакцию на разнообразные внештатные ситуации.

Анализ поставленной задачи приводит к следующим выводам. С одной стороны, ключевой функционал мониторинга (обеспечение указанных информационных потоков; механизмы передачи данных, запуска процедур сбора, оповещения, сохранения и архивации данных) должен быть реализован максимально просто и универсально и инкапсулирован в ядре системы (на ее серверной стороне), предоставляя сотруднику, сопровождающему систему, прозрачный набор механизмов для построения оптимального решения стоящих перед ним задач. С другой стороны, система должна позволять легко менять конкретную реализацию указанных выше процедур в соответствии с нуждами пользователя и позволять ему реализовывать и использовать в системе собственные модули оповещения, вывода, сбора и анализа данных, работающие в рамках поддерживаемой и регламентируемой ядром вышеуказанной схемы. Лучшим вариантом архитектуры ядра в таком случае может служить вариант «главная программа (core program) + подключаемые модули/агенты (plugins)», характерный, например, для Nagios или Munin [2, 3]. Подобные системы позволяют осуществлять мониторинг чего угодно – при условии разработки программного компонента, поставляющего интересующие данные о необходимом объекте, и выполнять в критической ситуации какие угодно действия – в случае разработки компонента, реализующего при выполнении некоторого условия требуемую последовательность действий.

На рисунке 1 приведена примерная архитектура системы, удовлетворяющей этим требованиям.

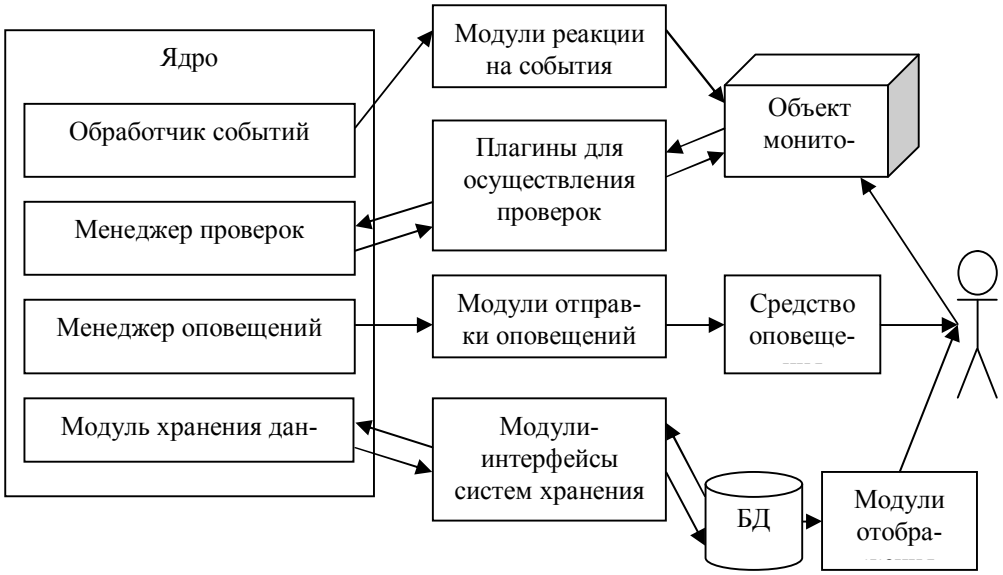

Рис. 1. Архитектура системы мониторинга

Характеристики системы Nagios

Сравнительный анализ существующего программного обеспечения, отвечающего подобным требованиям к ядру, позволил при разработке рабочего прототипа остановить выбор на системе Nagios, осуществляющей регламентный вызов модулей в соответствии с основной конфигурацией, описывающей всю систему – от узлов наблюдения до правил оповещения и расписаний. Модули Nagios являются исполняемыми файлами, возвращающими текстовое значение, реализуемы на любом удобном для этого языке (Perl, PHP, bash) и могут вызываться независимо от основного сервера, что упрощает их написание и отладку. Работа ядра Nagios заключается в запуске указанных для каждого сервиса проверочных команд с заданными интервалами и, в случае возвращения командой статуса Warning либо Critical, оповещения указанных для сервиса контактов, а также обеспечении срабатывания при выполнении заданных условий обработчиков событий. Таким образом, функционал Nagios полностью обеспечивает выполнение требований к подсистемам сбора данных и оповещения и частично – подсистеме коррекции. Крайне полезными являются также возможность определять иерархии объектов, а также возможность организации совместной работы нескольких систем с целью повышения надёжности и создания распределенной системы мониторинга [2].

Взаимосвязи между базовыми объектами, составляющими модель данных Nagios представлены на рисунке 2.

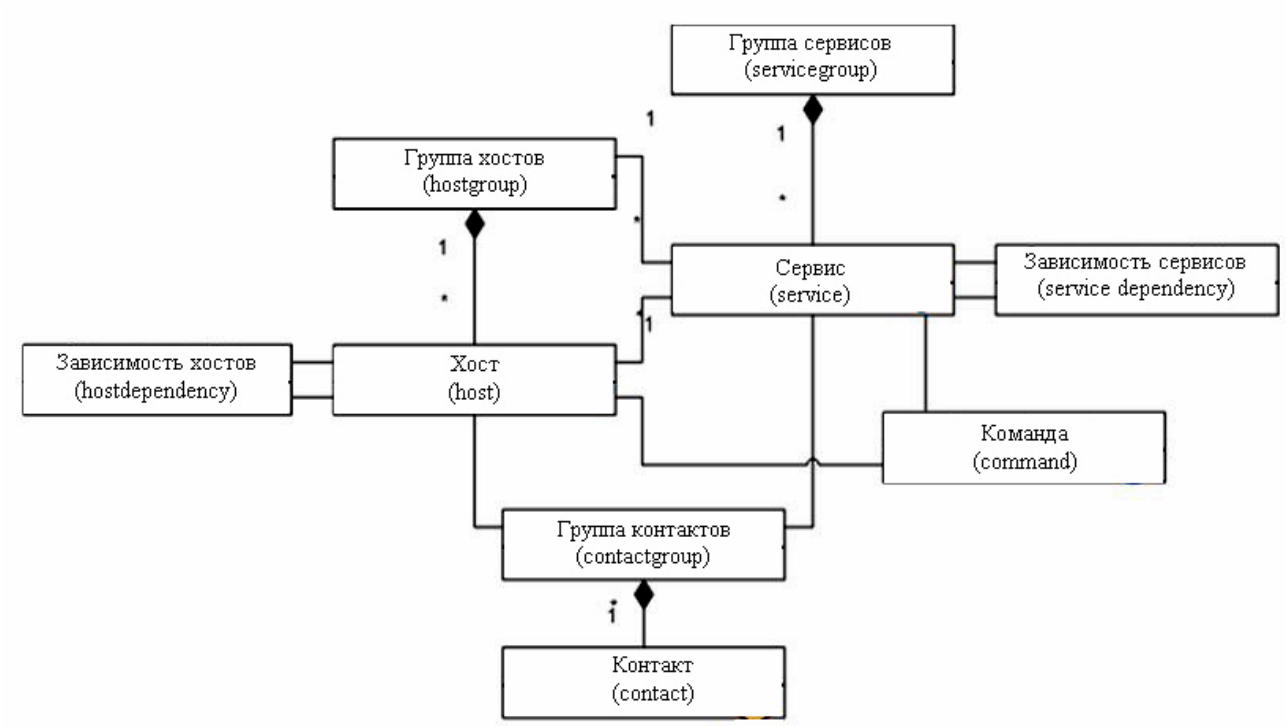

Рис. 2. Модель данных Nagios

Опишем представленные на схеме типы объектов:

Хост (host) – одно из устройств, за которыми осуществляется наблюдение. Определяющий параметр – IP-адрес; также определяется *команда* (см.ниже), проверяющая общую доступность устройства. Может входить в состав **Группы хостов (hostgroup).**

Сервис (service) – минимальный объект мониторинга, как правило, некоторый параметр хоста, подлежащий проверке. Определяются триггеры для следующих состояний: OK – данные получены, величина находится в границах оптимального интервала; Warning – данные получены, величина выходит за рамки оптимального интервала; Critical – данные получены, величина приняла критическое значение, что требует оперативного вмешательства. Unknown – данные не могут быть получены. Сервис может быть частью **Группы сервисов (servicegroup)**.

Контакт (contact) – некто, информируемый системой о событиях (для этого указываются соответствующие реквизиты, например адрес электронной почты); кроме того, используется для разграничения прав пользователей. Может входить в состав **Группы контактов (contact group).**

Команда (command) – описание вызова некоторой внешней программы с указанием необходимых параметров и порядка их передачи.

Зависимости (servicedependency, hostdependency) – используются для описания иерархических зависимостей между хостами или сервисами.

Для построения на основе Nagios системы, отвечающей сформулированным требованиям, необходимо в каждом конкретном случае решить следующие задачи:

1) организация получения данных от разнородных объектов мониторинга (сетевые устройства, сервера, службы);

2) организация хранения информации о состоянии объектов и работе системы в базе данных (Nagios в базовой комплектации не работает с БД);

3) создание удобных для конечного пользователя отображений состояния различных групп устройств и сервисов; построение отчётов и графиков.

Решение задачи отображения данных является сильно зависимым от специфики конкретной разрабатываемой системы и требований к ней, однако для двух групп задач – получения/анализа и хра-

нения данных можно в рамках описанной архитектуры привести набор достаточно универсальных решений и рекомендаций.

Методы получения данных

Методы мониторинга (в особенности сбора и анализа данных) подразделяют на две группы: централизованные и децентрализованные. В рамках первого подхода сбор данных о происходящих в целевой системе событиях осуществляется в определённом узле сети и в дальнейшем обрабатывается локально. В случае децентрализованного метода предполагается как минимум распределение отдельных функций системы мониторинга (сбор, хранение, отображение и пр.) между несколькими узлами, а возможно, и осуществление распределённых вычислений в случае трудоемкости или высокой важности некоторого этапа мониторинга (анализ собираемых данных; верификация состояний исследуемых систем). В нашем случае нецелесообразным представляется высокая децентрализация системы с распределением между всеми ее локальными компонентами функций анализа данных и диагностики возможных неполадок. Верификация системы с использованием всех предоставленных агентами сведений о состоянии отдельных объектов, анализ накопленных данных о работе системы и возможное принятие решения остаются на серверной части системы мониторинга.

Наиболее универсальным методом извлечения данных об интересующих параметрах функционирования удаленных объектов является запрос по протоколу SNMP (англ. *Simple Network Management Protocol* – простой протокол управления сетью) [2] – протокол управления сетями связи на основе архитектуры TCP/IP. Переменные, доступные через SNMP, организованы в иерархии. Эти иерархии и другие метаданные (такие, как тип и описание переменной) описываются *Базами Управляющей Информации* (англ. *Management Information Bases (MIBs)*). Эта модель расширяема, поэтому производители могут определять свои собственные элементы мониторинга. SNMP может также использоваться и для установки значений MIB-переменных, допускающих это действие, и, таким образом, для настройки удалённых объектов. Основными взаимодействующими лицами протокола являются агенты и системы управления. Роль сервера выполняют агенты (устройства, для опроса состояния которых и был разработан протокол). Роль клиентов отводится системам управления - сетевым приложениям, необходимым для сбора информации о функционировании агентов. В модели протокола можно выделить ещё два объекта: управляющую информацию и сам протокол обмена данными.

Для объектов и сервисов, не поддерживающих SNMP либо не предоставляющих всю необходимую информацию по этому протоколу, необходимы другие средства связи с системой мониторинга – агенты, выполняющие первичный сбор данных на клиентской стороне. Для Nagios стандартным средством такого мониторинга считается NRPE (Nagios Remote Plugin Executor). Возможность создания защищенного соединения (SSL) между сервером мониторинга и обслуживаемой стороной, а также отсутствие необходимости открытия стороннего доступа к поддеревьям SNMP могут быть рассмотрены как важные преимущества этого и подобных решений даже в случае наличия всей необходимой информации в соответствующем MIB.

NRPE со стороны клиентской машины – это демон, слушающий на определённом порту запросы от центрального сервера мониторинга, при получении запроса запускающий объявленный в файле конфигурации скрипт и отсылающий его выдачу запросившему информацию хосту (серверу мониторинга). NRPE с серверной стороны – это стандартный плагин check_nrpe, посылающий запросы демонам на клиентских машинах. С точки зрения пользователя запрос к удалённому плагину через NRPE ничем внешне не отличается от запроса к локальному – что позволяет говорить о хорошей степени прозрачности данного механизма.

Другая концепция агентного мониторинга – это так называемые пассивные проверки (Passive Checks), при которых инициатива проверки принадлежит не ядру мониторинга, а самой подконтрольной системе, отсылающей на сервер мониторинга только результаты проверки и тогда, когда это необходимо. Для систем на базе Nagios данная возможность реализована в модуле NSCA (Nagios Service Check Acceptor), демоне, работающем на сервере мониторинга и принимающем результаты произведенных на удаленных объектах проверок.

Хранение данных

Для успешного решения таких задач, как поиск закономерностей в повторяющихся неисправностях, подготовка отчетов о работе объектов за указанный период времени, построение графиков с указанными параметрами – необходимо хранение результатов проверок в базе данных. С этой целью отработана схема реализации подсистемы хранения данных, осуществляющей взаимодействие с базой данных (MySQL, PostgreSQL либо другой СУБД, работающую через IP-порт и SQL запросы) на базе библиотеки NDO (Nagios Database Output) [5]. Основные ее компоненты – событийный брокер, осуществляющий экспорт данных демона Nagios (в текстовый файл либо в сокет (TCP/IP или Unix) с защитой от обрыва связи); утилита LOG2NDO, импортирующая информацию из лог-файлов Nagios; демон NDO2DB, осуществляющий запись в БД данных, поставляемых вышеуказанными модулями. Утилита позволяет параллельную, одновременную работу нескольких клиентов (брокеров), например, поставляющих данные разных систем Nagios-мониторинга на разных машинах для централизованного хранения в одной БД; в этом случае создается по одному экземпляру процесса NDO2DB для каждого соединения. Схема информационных потоков Nagios-БД с участием NDO2DB представлена на рисунке 3.

Стандартная табличная структура NDO содержит 59 таблиц, агрегирующих информацию обо всех Nagios-объектах, начиная с хостов и сервисов и кончая описанными в конфигурационных файлах временными периодами. Однако и этот набор целесообразно расширить в соответствии с нуждами системы (разделение прав и авторизация пользователей, соответствующих контактам; ассоциированные с сервисами шаблоны для построения регулярных выражений, анализирующих «ответы» плагинов, и извлечения из них числовых данных). Последняя задача, решаемая в данной категории – это организация ротации базы данных и удаления избыточных записей для предотвращения снижения эффективности работы БД. Так, для ЦИВК ОИЯИ зафиксирована генерация данных системой мониторинга до 1 Гб в день, при выделенном дисковом пространстве 62 Гб. Простейший способ архивации – периодическое (раз в неделю либо чаще) снятие резервной копии базы данных, её архивация и последующая очистка базы; при необходимости анализа данных за некоторый период времени нужная копия может быть разархивирована и временно слита с текущей базой, а затем вновь осуществлен откат к точке начала анализа.

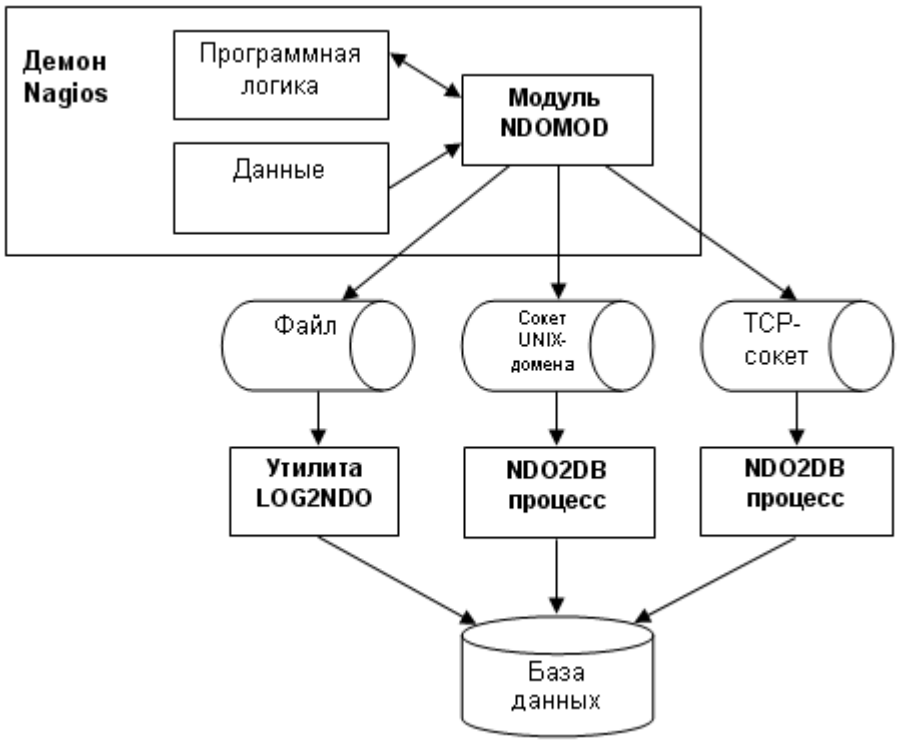

Рис. 3. Варианты экспорта данных Nagios в базу данных посредством NDO2DB

Построение системы мониторинга центрального информационно-вычислительного комплекса ОИЯИ

Примером ресурсного центра, входящего в состав глобальной распределенной вычислительной системы и требующего для корректного функционирования использование системы централизованного мониторинга, может служить центральный информационно-вычислительный комплекс ОИЯИ в Лаборатории информационных технологий (ЛИТ), объединяющий несколько вычислительных кластеров. WLCG-сайт ОИЯИ использовался экспериментами Большого Адронного Коллайдера на стадии подготовки экспериментов и активно используется на действующей фазе экспериментов как для массового моделирования физических событий, так и для целей физического анализа и хранения реплик данных больших объемов. Ресурсы ЦИВК ОИЯИ по состоянию на июнь 2011 года включают в себя около 350 машин (или 1584 вычислительных слота для выполнения задач), а также сервера пользовательских служб, таких, как batch, www, e-mail, DNS; сервера баз данных mysql и Oracle. Базовая операционная система, под управлением которой работают сервера – Scientific Linux (SL4.8 и SL5.4). К важнейшим ресурсам комплекса относятся две системы хранения данных dCache, включающие 12 серверов – основных интерфейсов системы dCache и 32 пула (системы хранения данных); а также три системы XROOTD (сервер обработки запросов к системе, сервера хранения данных). Для обеспечения этих систем предоставляется около 850 TB дискового пространства, организованного в RAID-массивы. Локальная сеть ЦИВК построена на базе агрегированных GigabitEthernet-соединений (транков), коммутаторов и маршрутизаторов HP Procurve и Cisco Catalyst [6]. Рассмотрим использованные управляющие алгоритмы и программные компоненты, решающие задачи оперативного и фонового мониторинга отдельных групп устройств и служб на каждом из трех уровней (аппаратном, сетевом и уровне служб).

Аппаратный уровень

Примером объектов этого уровня, подходящих для SNMP-мониторинга, выступают источники бесперебойного питания (ИБП). Идентификаторы переменных, хранящих необходимые величины, должны быть извлечены из базы управляющей информации (MIB), предоставляемой фирмой-производителем ИБП (в данном случае APC). Для контроля на каждом из рассматриваемых устройств были выбраны параметры, указанные в таблице 1, где OID – идентификатор соответствующего узла в дереве MIB.

Таблица 1. Характеристики ИБП, подлежащие мониторингу

Параметр	OID
Ёмкость аккумулятора	.1.3.6.1.4.1.318.1.1.1.2.2.1.0
Внешняя температура	.1.3.6.1.4.1.318.1.1.10.2.3.2.1.4.1
Внутренняя температура	.1.3.6.1.4.1.318.1.1.1.2.2.2.0
Частота	.1.3.6.1.4.1.318.1.1.1.4.2.2.0
Сила проходящего тока	.1.3.6.1.4.1.318.1.1.1.4.2.1.0
Напряжение	.1.3.6.1.4.1.318.1.1.1.4.2.4.0
Нагрузка (%)	.1.3.6.1.4.1.318.1.1.1.4.2.3.0
Состояние батареи	.1.3.6.1.4.1.318.1.1.1.2.1.1.0

Для получения значений указанных переменных был использован стандартный плагин Nagios check_snmp, входящий в базовую комплектацию. Параметры, необходимые этому плагину, включают адрес опрашиваемого устройства, community-идентификатор с правами чтения, OID необходимой переменной, «жёлтую» и «красную» границы её значения. Точно так же осуществляется сбор данных и со всех остальных устройств, для которых достаточно SNMP-мониторинга. Сюда относятся также

вентиляционные блоки серверных стоек, для которых анализируются параметры, сведённые в таблицу 2.

Таблица 2. Характеристики вентиляционных блоков серверных стоек, подлежащие мониторингу

Параметр	OID
Проходящий воздух (куб.м./час)	.1.3.6.1.4.1.318.1.1.14.3.3.1.14.1
Скорость вращения вентилятора	.1.3.6.1.4.1.318.1.1.14.4.1.1.2.1.1
Температура (значения трёх датчиков)	.1.3.6.1.4.1.318.1.1.14.3.3.1.5.1 .1.3.6.1.4.1.318.1.1.14.3.3.1.6.1
Флаг состояния (может сигнализировать о неисправностях разных типов)	.1.3.6.1.4.1.318.1.1.14.3.3.1.10.1

Для эффективного мониторинга серверов протокола SNMP уже недостаточно. В общем случае система мониторинга должна получать информацию о таких параметрах, как загрузка процессора, памяти, свободное дисковое пространство на доступных разделах, а для отдельных серверов – специфические данные по работающим на них процессам (сервисам). Контроллеры дисковых массивов RAID могут служить примером серверных объектов, не поддерживающих SNMP либо не предоставляющих всю необходимую информацию по этому протоколу. Ситуация осложняется тем, что не существует единообразного для всех RAID-контроллеров средства (интерфейса) контроля и управления. Большинство фирм-поставщиков предлагают своё программное обеспечение для мониторинга и управления подконтрольными массивами дисков (3ware – утилита tw_cli; HighPoint – hptraidconf; Adaptec – arcconf). Процедуры проверки контроллеров любого типа работают по общему алгоритму:

1. Проверить наличие и доступность на данной машине raid-контроллеров требуемого производителя.
2. Получить список идентификаторов контроллеров.
3. Для каждого из полученных контроллеров получить общую информацию о текущем состоянии, список активных портов, информацию о подключенном диске и его состоянии (с использованием регулярных выражений для разбора данных, возвращаемых используемыми утилитами).

Определить статус проверки (корректное состояние, тревожное состояние, критическое состояние), сформировать отчет о проверке.

Для осуществления подобных проверок был реализован единый плагин, обнаруживающий на локальной машине контроллер той или иной фирмы и в зависимости от его типа применяющий соответствующие команды для проверки состояния. Другие задачи, связанные с мониторингом серверов (состояние процессора, памяти, дисков, работающих процессов) более тривиальны и могут быть решены с помощью сочетания стандартных nagios-плагинов, таких, как check_load, check_disk, check_procs с check_nrpe.

Сетевой уровень

Основные задачи на данном уровне – мониторинг коммутаторов HP Procurve и Cisco, агрегированных соединений (транков), а также внешнего канала. Мониторинг всех объектов сетевого уровня может быть осуществлен с помощью SNMP с привлечением некоторых дополнительных средств анализа на серверной стороне. Так, информация об отдельных интерфейсах (портах) коммутатора хранится в поддереве iso.0.8802.1.1.2.1.3.7.1. Поддерево iso.2.840.10006.300.43.1.1.2.1.1 содержит информацию об агрегировании отдельных интерфейсов в транки; так, если порты 269-272 образуют транк, идентифицируемый номером 297, соответствующая ему переменная MIB будет иметь значение: iso.2.840.10006.300.43.1.1.2.1.1.297 = STRING: "269, 270, 271, 272, ".

Для эффективного наблюдения за состоянием портов и каналов требуется построение графиков загрузки, обновляющихся в реальном времени. Для этого была применен набор утилит MRTG. Его

основной процесс должен читать указанный в качестве параметра конфигурационный файл и в соответствии с ним получать новые значения описанных в нем величин, графики которых строятся. Эти данные должны поставляться скриптом, который будет:

4. принимать в качестве входных параметров идентификатор устройства, наименование сервиса или нечто подобное, позволяющее однозначно определить необходимую строку лог-файла Nagios либо запись БД;
5. выполнять запрос к БД / осуществлять поиск в лог-файле Nagios последней записи, соответствующей данным критериям;
6. выводить извлеченное значение.

Данная схема использована не только для отображения состояния объектов сетевого уровня, но и везде, где необходимы графики общего вида (отчёт за день/неделю/месяц/год) без тонкой настройки отображения пользователем. Типичный скрипт извлечения значения последней проверки из лог-файла использует три параметра: идентификатор устройства, наименование сервиса и форма записи единицы измерения (для выделения из строки численного значения).

Уровень служб

Для массового хранения экспериментальных данных и результатов моделирования ЦИВК ОИЯИ использует файловую систему dCache, предоставляющую доступ к файлам, в том числе средствами GRID, которые логически упорядочены в одно дерево, но физически находятся на разных дисках. Система не содержит встроенных средств анализа эффективности работы, достаточного для принятия решений по программной или аппаратной реконфигурации системы хранения данных. Наиболее актуальной здесь является задача учета и анализа эффективности распределения ресурсов дисковой памяти. Набор параметров, используемых системой анализа заданий сайта и отображаемых соответствующей информационной системой, недостаточен для задач локального администрирования системы [7]. Для мониторинга были выбраны следующие дополнительные параметры:

7. объем и эффективность использования памяти; количество и объём файлов, которые были однажды записаны и с тех пор ни разу не читались;
8. количество одновременно выполняемых процессов передачи;
9. уровень ошибок – статистика успешных/неуспешных завершений процедур;
10. уровень нагрузки и трафика;
11. нагрузка на процессы-«двери» и «пулы».

Эти данные собираются при помощи плагинов, написанных на языке perl и запускаемых на основном сервере dCache посредством Nrpe. Помимо общего мониторинга dCache к прикладному уровню относится также и анализ состояния RAID-массивов, обеспечивающих систему хранения, средства которого были рассмотрены ещё на физическом уровне. Для каждого из серверов указывается текущий статус, число проверенных дисков, дисковых массивов и контроллеров (например, 4 array(s), 23 drives checked on 2 controller(s)); в случае наличия проблем указывается их природа и максимально возможно локализуется её источник. Два основных класса таких проблем (некорректных состояний):

Проблемы, непосредственно относящиеся к raid-массивам, например

<контроллер> is DEGRADED-VRFY (выход из строя одного из приводов, производится проверка и восстановление).

Внешние проблемы (недоступность того или иного сервера в сети, ошибка плагина), например:

CHECK_NRPE: Socket timeout after <N> seconds (клиентская часть NRPE на требуемой машине не вернула ответ в установленный срок).

Относящийся также к прикладному уровню мониторинг таких сервисов, как SMTP, DNS, www (веб-ресурсы ЦИВК и ЛИТ) является тривиальным и осуществляется с помощью стандартных nagios-плагинов check_http, check_dns, check_smtp.

Представление данных

Web-интерфейс системы был реализован в основном на языке PHP с использованием для разметки страниц и представления данных XHTML и Javascript. Для автообновления графиков в реальном времени использована технология AJAX (jQuery). В основном режиме работы пользователю доступно дерево объектов, где на верхнем уровне находятся группы хостов, а «листьями» дерева являются сами хосты и в некоторых случаях – отдельные сервисы-проверки (рис. 4). Для каждого объекта динамически генерируется страница с данными по всем определенным для него сервисам – текущее состояние, график значений за прошедшие сутки и некоторые другие вспомогательные данные. Нажатием специальной кнопки можно вызвать всплывающее окно с отчетом по выбранному сервису, который по умолчанию содержит информацию обо всех нештатных ситуациях за минувшие сутки (данный интервал может быть изменён после открытия окна).

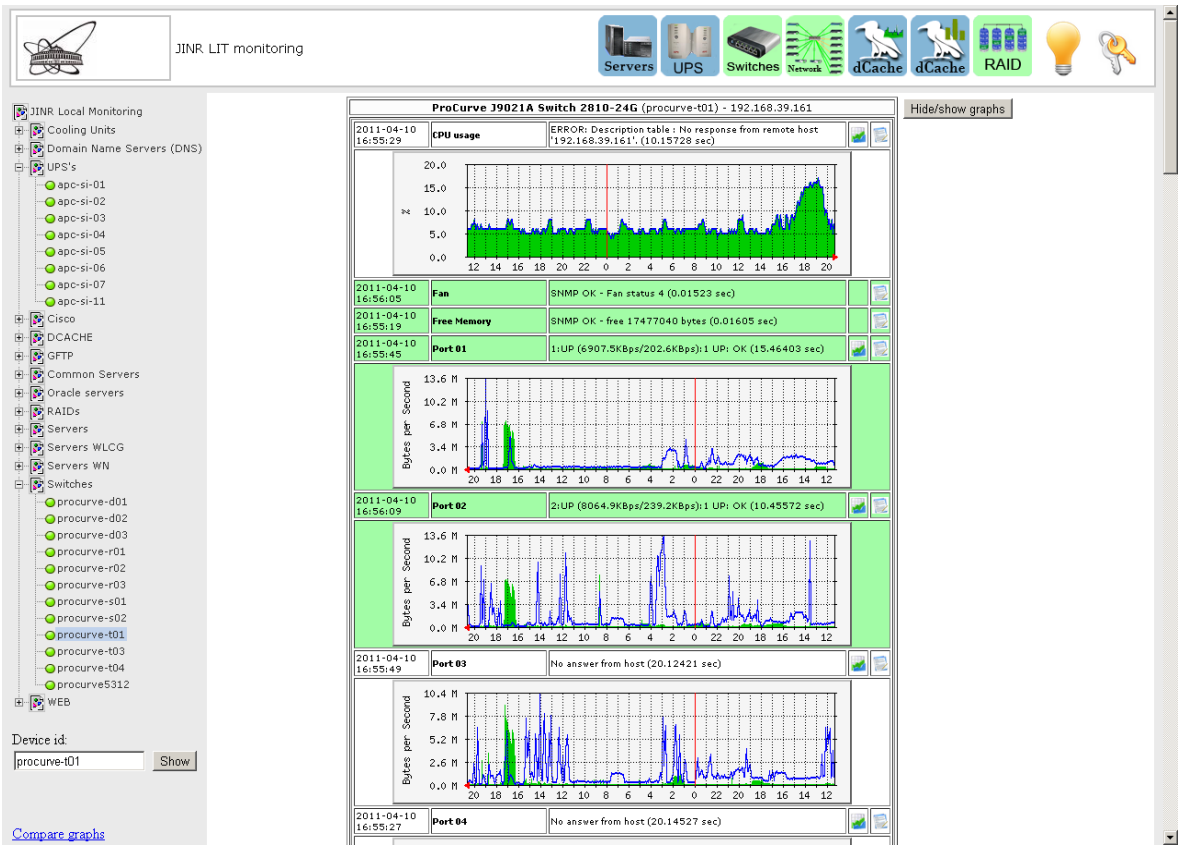

Рис. 4. Общий вид интерфейса системы: дерево объектов (http://litmon.jinr.ru)

Было также создано несколько специальных отображений, позволяющих более наглядно оценить состояние той или иной группы сервисов и проследить взаимосвязь между объектами. Для этого был задействован NagVis – обладающее широкими возможностями дополнение для визуализации состояния узлов и сервисов, позволяющее размещать объекты на графических «картах» и визуализировать IT процессы. Так, с его помощью создано обзорное отображение, где на структурную схему сети наложены маркеры, характеризующие состояние соответствующих объектов. Для группы сервисов, относящихся к dCache, создано отображение, позволяющее сравнить наблюдаемые значения проверяемых параметров для экспериментов Atlas и CMS. Ряд отображений создан без помощи NagVis – это сводки по серверам (с распределением их по стойкам, ИБП, администраторским группам) и отчётная таблица по дисковым массивам с указанием состояния каждого контроллера и диагностикой типа проблемы в случае её наличия (рис. 5).

В случае возникновения нештатной ситуации (а также и выхода из подобной ситуации) система рассылает лицам, ответственным за проблемные сервисы соответствующие оповещения посредством электронной почты или SMS.

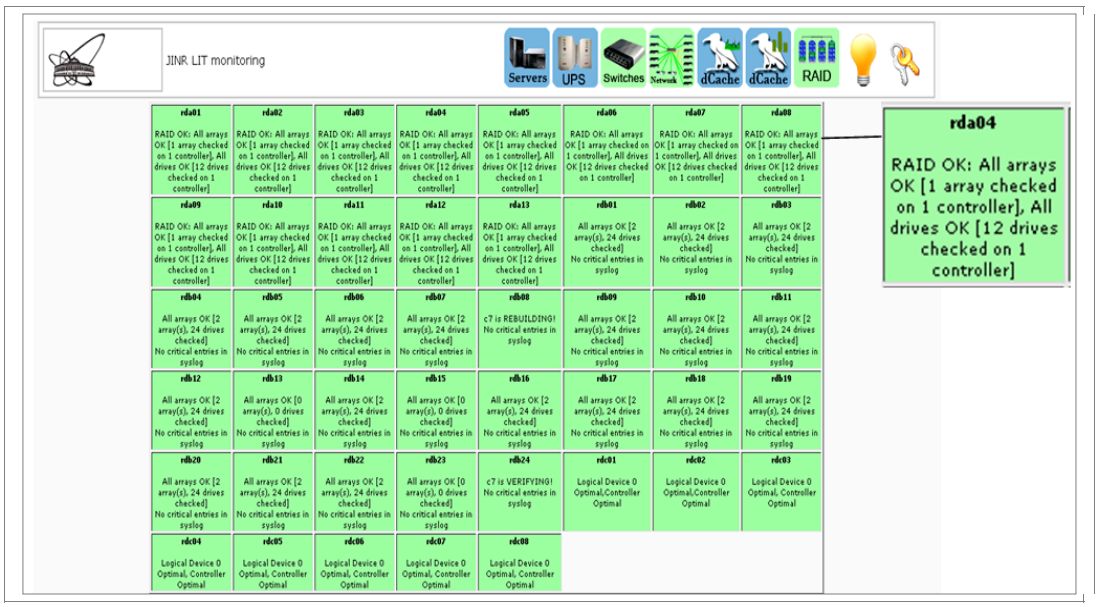

Рис. 5. Отчет о текущем состоянии RAID-серверов (http://litmon.jinr.ru/index.php?view=raids)

Заключение

Представленная архитектура системы локального мониторинга ресурсного центра формулировалась с целью соответствия требованиям универсальности, расширяемости, централизованности и защищённости. Опирающиеся на нее методы организации базовых информационных потоков системы мониторинга и её подсистем, позволяют создать удовлетворяющую данным требованиям систему на основе подходящих открытых программных продуктов либо путём реализации необходимых модулей с нуля. Изложенные положения были использованы при разработке прототипа системы локального мониторинга ЦИВК ОИЯИ, применённые в которой методы мониторинга отдельных устройств и сервисов, распределённых по трём уровням – аппаратному, сетевому и прикладному, также рассмотрены в статье.

Литература

1. Comparison_of_network_monitoring_systems. – [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems
2. Barth W. Nagios System and Network Monitoring / No Starch Press, 2006.
3. Falko Timme: Server Monitoring With munin And monit. – [Электронный ресурс]. URL: http://www.howtoforge.org/server_monitoring_monit_munin.
4. Благодарный Е. SNMP протокол – принципы, безопасность, применение. – [Электронный ресурс]. URL: [http://www.linuxrsp.ru/artic/snmp.html].
5. Galstad E. NDOUTILS Documentation Version 1.4 / Etien Galstad, NDOUtils Documentation. [Электронный ресурс]. URL: http://nagios.sourceforge.net/docs/ndoutils/NDOUtils.pdf.
6. Кореньков В.В. Статус и перспективы развития сетевой и компьютерной инфраструктуры ОИЯИ / Сетевая и информационно-вычислительная инфраструктура ОИЯИ. – [Электронный ресурс]. URL: [http://www.d-instruments.ru/materials/Korenkov_Infrastructure.pdf].
7. Трофимов В., Дмитриенко П. Об одном подходе к мониторингу и последующей оптимизации элемента памяти грид-структуры, реализованного на основе системы dCache / Научный отчет Лаборатории информационных технологий ОИЯИ 2008-2009. – Дубна: ОИЯИ, 2009.