УДК 004.457

Никитин П.В.[1,2], Горохова Р.И.[1]

[1] Марийский государственный университет, г. Йошкар-Ола, Россия
[2] Межрегиональный открытый социальный институт, г. Йошкар-Ола, Россия

МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ОПТИМИЗАЦИИ МОНИТОРИНГА СОСТОЯНИЯ ПРОГРАММНО-АППАРАТНОГО ОБЕСПЕЧЕНИЯ*

Аннотация

В статье описывается модель системы мониторинга состояния программно-аппаратного обеспечения крупной компании, предоставляющей клиентам программное обеспечение как услугу (SaaS-решение). Рассмотрены основные функции и назначения систем мониторинга: предоставление актуальных данных для анализа состояния ИТ-инфраструктуры, быстрое выявление появившейся неисправности и эффективное её устранение, поддержание производительности ИТ-инфраструктуры. Описаны основные риски, связанные с предоставлением данных услуг, приведены сравнительные характеристики программного обеспечения, приведены авторские методики мониторинга состояния программно-аппаратного обеспечения. Разработанная система мониторинга позволяет повысить качество услуг, предоставляемых клиенту компанией, и выполняет сбор параметра свободного места на диске, проверку задач в планировщике заданий, проверку свободной оперативной памяти по контролю свободного места на диске, просмотр и анализ всевозможных логов. Предлагаемый продукт для мониторинга серверов и служб, а также автоматизации задач поможет значительно снизить время простоя сервисов, повысить качество услуг, предоставляемых клиенту компанией, а также значительно улучшить производительность, избежав, при этом, неоправданных вложений.

Ключевые слова

Моделирование; информационные системы; мониторинг; SaaS-решения; автоматизированные системы мониторинга.

Nikitin P.V.[1,2], Gorokhova R.I.[1]

[1] Mari State University, Yoshkar-Ola, Russia
[2] Interregional Open Social Institute, Yoshkar-Ola, Russia

MODELING OF INFORMATION SYSTEM FOR OPTIMIZATION OF MONITORING OF SOFTWARE AND SOFTWARE CONDITION

Abstract

The article describes a model of a hardware and software monitoring system for a large company that provides customers with software as a service (SaaS solution). The main functions and purposes of monitoring systems are considered: providing up-to-date data for analyzing the state of the IT infrastructure, quickly identifying the fault that has appeared and effectively eliminating it, maintaining the performance of the IT infrastructure. The main risks associated with the provision of these services are described, the comparative characteristics of the software are given, author's methods for monitoring the status of software and hardware are given. The developed monitoring system allows to improve the quality of services provided to the client by the company and collects the free disk space parameter, checks tasks in the task scheduler, checks free RAM to monitor free disk space, and examines and analyzes all kinds of logs. The proposed product for monitoring servers and services, as well as automating tasks, will significantly reduce the downtime of services, improve the quality of services provided to the client by the company, and significantly improve productivity, while avoiding unjustified investments.

* **Труды II Международной научной конференции «Конвергентные когнитивно-информационные технологии» (Convergent'2017), Москва, 24-26 ноября, 2017**

Proceedings of the II International scientific conference "Convergent cognitive information technologies" (Convergent'2017), Moscow, Russia, November 24-26, 2017

Keywords

Modeling, information systems; monitoring; SaaS-solutions; automated monitoring systems.

Введение

Программное обеспечение как услуга (SaaS) – это предоставление потребителю возможности использовать приложения поставщика, функционирующие на облачной инфраструктуре. Доступ к приложениям осуществляется с различных клиентских устройств через интерфейс тонкого клиента, например, веб-браузер. Потребитель не контролирует и не управляет облачной инфраструктурой, на которой запущено приложение, включая сеть, сервера, операционные системы, хранилища данных и даже параметры приложения. Возможное исключение – отдельные пользовательские настройки приложения.

Когда клиент потребляет ту или иную услугу в качестве сервиса, а не в качестве устанавливаемого у себя программного обеспечения, уже в этот момент, как правило, он становится потребителям SaaS. Простейший, и всеми используемый SaaS-сервис – это сервис электронной почты – тот же gmail.

Кроме преимуществ у «облачных» подходов реализации услуг клиентам есть и ряд рисков. Одни из них риски, связанные с корректной работоспособностью оборудования и программного обеспечения. Если клиент не имеет возможность получить услуги, то предприятия могут ожидать негативные последствия. Например, компания теряет лояльность клиента, что может сказаться на репутации. Компания может быть оштрафована по договору, в котором прописан SLA ИТ-услуг. Если компания имеет доход с предоставления услуг в количестве за время, то простой такой ИТ-услуги ведёт к прямым финансовым потерям. Следовательно, для уменьшения бизнес-рисков рекомендуется в ИТ-инфраструктуру компании внедрять систему мониторинга.

Основные функции и назначения систем мониторинга

Главные функции системы мониторинга это: предоставление актуальных данных для анализа состояния ИТ-инфраструктуры, быстрое выявление появившейся неисправности и эффективное её устранение. Так же системы мониторинга могут следить за производительностью ИТ-инфраструктуры. Это позволяет администраторам идентифицировать «узкие места» в системе, а также заметить снижение производительности раньше пользователей. Непрерывный мониторинг помогает поддерживать основные ИТ-сервисы в исправном состоянии, помогает уменьшить количество простоев в их работе, спроектировать модернизацию инфраструктуры и поддерживать хороший уровень качества услуг. До автоматизированных систем мониторинга данными задачами занимались системные администраторы. Они вручную собирали данные о состоянии систем в разном неспециализированном программном обеспечении. А иногда информация о состоянии вообще нигде не хранилась и не обрабатывалась. При уходе специалиста все собранные данные об инфраструктуре, основанные на опыте, полностью исчезали.

Сейчас на рынке существует множество автоматизированных систем для мониторинга, они постоянно проводят анализ систем и агрегируют данные в хранилище. Из хранилища всегда можно получить необходимую информацию для изучения.

Очень интересным для ИТ-сервисов является мониторинг на основе сценариев взаимодействия пользователей с программным обеспечением. Такие сценарии запускаются через определенные интервалы на живой системе. Результат работы данного сценария посылается в систему мониторинга для дальнейшего анализа.

Для хранения поступившей информации как правило используется база данных. В отдельных её таблицах хранится описание объектов мониторинга, например, каждый сервер, сетевое устройство или ИТ-услуга. Данный вид хранения дает возможность интегрировать систему мониторинга с визуальными представлениями, например, диаграммами и графиками. Структура мониторинга заметно изменяется со временем, например, при широком внедрении виртуализации приходится обкладывать мониторингом как физический сервер, так и виртуальные машины, работающие на нем, ранее наблюдение велось только за физическими серверами. Иногда систему мониторинга настраивают на выполнение повторяющихся сервисных задач. Это задачи по автоматической очистке корзины, создание резервных копий важных данных, перемещение каких-либо файлов, когда не хватает дискового пространства.

При выборе, разработке, внедрении систем автоматизированного мониторинга для сначала нужно определиться с объектами, которые система будет наблюдать. Затем определить важные события и показатели, характеристиками которых являются частота сканирования и количество оповещений при переходе в нерабочее состояние.

Внедрение подобных решений особенно важно при использовании сервисного подхода к деятельности ИТ-подразделений, когда все процессы пересматриваются с точки зрения предоставляемых подразделением ИТ-сервисов. Каждый бизнес-сервис корпоративной системы по возможности интерпретируется как ИТ-сервис, задается определенный уровень качества его предоставления. Далее он описывается в системе мониторинга как набор взаимосвязанных компонентов ИТ-инфраструктуры.

В рамках данного подхода устанавливается соглашение об уровне качества сервисов (Service Level Agreement, SLA). Система мониторинга ведет сбор и хранение данных о качестве оказании ИТ-сервисов. Затем составляется отчет за определённый период времени, на базе накопленных данных. Анализ отчетной информации помогает осуществлять:

- модернизацию ИТ-инфраструктуры;
- переоценку уровня предоставления ИТ-сервисов;
- преобразование деятельности ИТ-подразделения.

Для потребителей разного уровня существуют разные типы систем мониторинга. Для небольших систем, как правило, хватает отправки уведомлений и обобщенного анализа узлов. Основными функциями систем мониторинга, обычно, выделяют: слежение; построение отчётов; хранение данных; поиск «узких» мест; визуализация; автоматизация сценариев. Если в системе мониторинга реализованы большинство, выделенных выше, функций, то она освобождает ресурсы администратора, так как поломки устраняются быстрее, диагностика производится точно и многомерно. Так же не нужно проверять состояние каждого узла вручную и теперь администратор может легко заняться планированием изменений в инфраструктуре.

Обзор существующих автоматизированных систем мониторинга

От множества факторов зависит выбор способов и объектов мониторинга. К таким факторам можно отнести: конфигурации серверов и ПО, установленное на них, конфигурация сетей, работающих в данных сетях служб и сервисов, используемого мониторинга и возможностей ПО для мониторинга. В общих чертах можно выделить основные проверки:

- проверка физической доступности оборудования (проверка состояния оперативной памяти и длины дисковой очереди);
- проверка характеристик, специфических для служб и сервисов в данном конкретном окружении (состояние служб и назначенных заданий, доступность IIS-сайтов);
- подробная проверка не критичных, но важных характеристик функционирования: загрузки, производительности, и т.п. (доступность виртуальной машины в сети хоста);
- проверка состояния выполняемых сервисов и служб (сбор характеристик ОС как Windows, так и на Linux).

Рассмотрим автоматизированные системы, предоставляемые на рынки ИТ.

1. Система мониторинга Nagios (изначально Netsaint) – система мониторинга в основном предназначена для наблюдения за сетями и системами [1]. Свободно распространяемая. Система позволяет обозначить проблемы сразу после их обнаружения, производит комплексное наблюдение за всей ИТ-инфраструктурой, может поставлять заинтересованным лицам данные полученные при наблюдении, имеет возможность наблюдать за безопасностью системы Данные возможности ведут к сокращению времени простоя и как следствие, уменьшению коммерческих потерь. В самом начале создана для операционных систем на базе Linux, сейчас в равной мере хорошо работает также и под Sun Solaris, FreeBSD, AIX и HPUX.

Возможности:

- мониторинг состояния хостов (использование диска, системные журналы, загрузка процессора) в большинстве сетевых операционных систем;
- мониторинг сетевых служб (SMTP, POP3, HTTP, NNTP, ICMP, SNMP);
- поддержка удаленного мониторинга через шифрованные туннели SSH или SSL;
- параллельная проверка служб;
- 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;
- возможность определять иерархии хостов сети с помощью «родительских» хостов, позволяет обнаруживать и различать хосты, которые вышли из строя, и те, которые недоступны;
- возможность определять обработчики событий, произошедших со службами или хостами для проактивного разрешения проблем;
- отправка оповещений в случае возникновения проблем со службой или хостом (с помощью почты, пейджера, смс, или любым другим способом, определенным пользователем через модуль системы);
- автоматическая ротация лог-файлов;
- включает в себя утилиту nagiostats, которая выводит общую сводку по всем хостам, по которым ведется мониторинг;
- возможность организации совместной работы нескольких систем мониторинга с целью повышения надёжности и создания распределенной системы мониторинга.

К минусам данной системы можно отнести: «общий» характер мониторинга показателей; проблема взаимодействия с серверами под управлением Windows; «сетевая» направленность мониторинга.

2. Автоматизированная системы мониторинга Zabbix – система мониторинга в основном предназначена для наблюдения за сетями и системами [2]. Свободно распространяемая. Состоит из нескольких частей.

Сервер мониторинга (ядро) – опрашивает по заданному интервалу характеристики, проводит их анализ, обрабатывает итоговые данные, если данные выходят за преднастроенные пороговые значения, запускает скрипты инициирующее рассылку уведомлений. Имеет возможность проверять сетевые сервисы удаленно. Хранит все конфигурационные настройки, собранные исторические данные, статистическую информацию в базе данные, обычно это СУБД MySQL или PostgreSQL. На сервера под управлением ОС Windows и OpenBSD не устанавливается. Обычно данное ядро именуется как Zabbix-сервер.

Прокси – занимается сбором характеристик на серверах тем же протоколом что и Zabbix. Характеристики, собранные данным узлом, хранятся в собственном хранилище прокси агента на локальном уровне, как только появится возможность данные передать Zabbix-серверу, агент их передает как можно быстрее. Прокси агент – это хорошее решение для мониторинга удаленных узлов, находящихся в филиалах, в которых нет системного администратора. Часто агента используют для уменьшения нагрузки на основного Zabbix-сервера, так прокси агент занимается сбором всех данных вместо Zabbix-сервера, тем самым освободив ресурсы сервера на выполнение других важных задач. Прокси агентов можно настроить очень много, а Zabbix-сервер может работать только в одном экземпляре.

Агент – специализированная фоновая задача, которую устанавливают и запускают на объектах наблюдения, она предоставляет доступ Zabbix-серверу или прокси агенту к характеристикам объекта. С помощью агента Zabbix контролирует параметры локальных ресурсов объекта (память процессор, жесткие диски и т.д.) и выполняемые на нем приложения. Обычно любой объект наблюдения из сетевых систем, работает с запущенным Zabbix-агентом. Не всегда запуск агента удобен, тогда для мониторинга можно использовать простой протокол сетевого управления SNMP. Zabbix умеет работать с тремя версиями данного протокола. Собирать характеристики через протокол SNMP можно запуская собственные скрипты отдающие данные. Еще можно запуская определенные встроенные проверки или отправив запрос по http, ssh, ftp протоколам, замеряя время ответа данных узлов. Так как Zabbix-агенты используют системные вызовы для получения информации о характеристиках, они являются очень эффективными. Одно из главного преимущества Zabbix является что Zabbix-агенты поддерживают не только *nix ОС, но и ОС Windows. Все поддерживаемые платформы указаны в таблице 1.

Таблица 1. Поддерживаемые платформы

Платформа	**ZABBIX-сервер**	**ZABBIX-агент**
AIX	Поддерживается	Поддерживается
FreeBSD	Поддерживается	Поддерживается
HP-UX	Поддерживается	Поддерживается
Linux	Поддерживается	Поддерживается
Mac OS X	Поддерживается	Поддерживается
Novell Netware	-	Поддерживается
Open BSD	Поддерживается	Поддерживается
SCO Open Server	Поддерживается	Поддерживается
Solaris	Поддерживается	Поддерживается
Tru64/OSF	Поддерживается	Поддерживается
Windows	-	Поддерживается

Веб-интерфейс – реализованное на PHP средство визуального представления для Zabbix, как и все приложения, написанн на языке программирования PHP. Для запуска нужно наличие веб-сервера.

С помощью Zabbix можно осуществлять распределённый мониторинг до 100000 узлов, где конфигурация младших узлов контролируется старшими в иерархии. Также продукт включает централизованный мониторинг лог-файлов, возможность создавать карты сетей (вручную по шаблону), выполнение запросов в различные базы данных, генерацию отчётов и тенденций, выполнение сценариев на основе мониторинга, поддержку интеллектуального интерфейса управления платформами (IPMI).

В Zabbix существуют условия-триггеры, они включаются если что-то в системе работает не корректно. Сам Zabbix предоставляет универсальный функционал для гибкой настройки триггеров. Они уведомляют о состоянии системе в веб-интерфейсе, или в другие настроенные источники уведомлений.

Zabbix достаточно самостоятелен и сможет отправить уведомление на почту, в jabber или sms с помощью gsm-модема, или даже попытаться самостоятельно поднять упавший сервис, выполнив заранее определенные действия, которые запускаются при срабатывании определенных триггеров [3].

Автоматическое обнаружение:

• автоматическое обнаружение по диапазону IP-адресов, доступным сервисам и SNMP проверка;

• автоматический мониторинг обнаруженных устройств;
• автоматическое удаление отсутствующих хостов;
• распределение по группам и шаблонам в зависимости от возвращаемого результата.

В запасе у Zabbix есть еще полдесятка функций, которые позволяют еще больше упростить наблюдение за сетью, такие как мониторинг состояния веб-сайта с помощью автоматического выполнения сценария вроде имитации пользовательских действия на сайте. В итоге это одна из мощнейших и обширнейших систем мониторинга.

В итоге в основу автоматизированной системы мониторинга была выбрана автоматизированная системы мониторинга Zabbix.

Проектирование модели автоматизированной системы мониторинга

Предположим, что центр отправки данных (ЦОД) предприятия находится в двух дата-центрах (ДЦ), в каждом из них около 100 виртуальных машин, которые будут покрываться мониторингом. Zabbix-сервер одновременно может работать только в одном экземпляре, соответственно нужно обеспечить доступность потока данных из одного ДЦ в другой. Принято решение поднять прокси-агента в каждом из ДЦ. База данных для прокси-агентов будет находится на той же машине что и сам агент. Таким образом при потере связи между ДЦ агенты продолжать собирать показатели каждый в своем ДЦ, и хранить их в локальной базе данных (БД). Как только у прокси-агентов появляется возможность отправить данные к Zabbix-серверу, они отправляют ему всю собранную информацию с последнего сеанса связи.

Как сказано выше, Zabbix-сервер одновременно может работать только в одном экземпляре, соответственно он будет запущен в одном из ДЦ, и все данные логично хранить в БД в том же ДЦ, так как считается, чем ближе обработчики данных и хранилище, тем система получится производительнее. Но оставив данные в одном экземпляре в одном ДЦ, есть большой риск потерять их целиком или доступ к ним не будет ограничен на момент аварии. Так как система мониторинга должна помогать во время аварии, а не наоборот, данная ситуация не приемлема. Одно из решений — это настроить отказоустойчивый кластер БД.

Уровень взаимодействия с базой данных является последним уровнем архитектурной схемы компонентов системы распределенного удаленного хранения данных. Данный уровень отвечает за обеспечение надежного хранения данных, высокий показатель их доступности, а также за защиту пользовательской информации. Требование повышенной отказоустойчивости в полной мере распространяется на данный компонентный слой.

Для обеспечения отказоустойчивости кластера из серверов MySQL используется репликация – одна из техник масштабирования баз данных. Состоит эта техника в том, что данные с одного сервера базы данных постоянно копируются (реплицируются) на один или несколько других (называемые репликами). Для приложения появляется возможность использовать не один сервер для обработки всех запросов, а несколько. Таким образом появляется возможность распределить нагрузку с одного сервера на несколько. СУБД MySQL предоставляет несколько вариантов конфигурирования кластера для осуществления репликации данных:

• Master-Slave. В этом подходе выделяется один основной сервер базы данных, который называется Мастером. На нем происходят все изменения в данных (любые запросы MySQL INSERT/UPDATE/DELETE). Slave -сервер постоянно копирует все изменения с Мастера. С приложения на Slave-сервер отправляются запросы чтения данных (запросы SELECT). Таким образом Мастер сервер отвечает за изменения данных, а Slave за чтение.

• Master-Master. В этой схеме, любой из серверов может использоваться как для чтения, так и для записи. Схема работы кластера MySQL в конфигурации Master-Master представлена на рисунке 1.

Схема реплицирования для MySQL была выбрана Master-Slave, как более надежная, поскольку надежность – одно из требований к архитектуре автоматизированной системы мониторинга.

В рамках реализации предложенной архитектуры информационной системы был развернут MySQL кластер, состоящий из двух машин по схеме Master-Slave. Данная схема позволяет организовать отказоустойчивое хранилище данных MySQL. Кроме этого, она позволяет распределить нагрузку на базу данных между несколькими серверами. Использованная схема кластера представляет из себя три машины, на которых запущен MySQL. Одна из них является Master-сервером, две другие – Slave-серверами. Master-сервер принимает и обрабатывает входящие запросы, а также ведет бинарный журнал, который синхронизируется со Slave-серверами. После получения новых записей бинарного журнала Slave-серверы выполняют указанные в нем команды и тем самым синхронизируются с Master-сервером.

При выходе из строя Master-сервера, один из оставшихся серверов принимает на себя его роль. В качестве системы хранения данных была использована InnoDB. Она выгодно отличается от MyISAM, так как предоставляет большую надежность при хранении данных. В итоге у нас получилось отказоустойчивая структура автоматизированной система мониторинга Zabbix, представленная на рисунке 2.

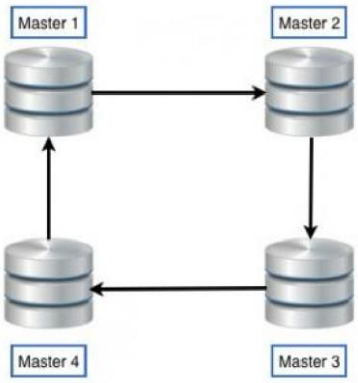

Рисунок 1. Схема работы кластера MySQL в конфигурации Master-Master

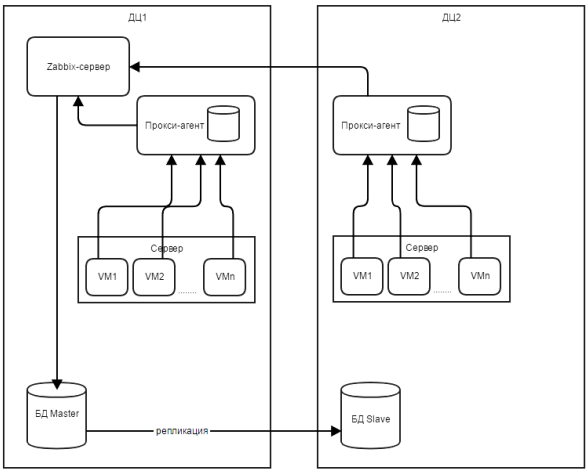

Рисунок 2. Структура автоматизированной системы мониторинга Zabbix в двух ДЦ

Система автоматизированного мониторинга Zabbix имеет очень широкий функционал, однако инструментарий для визуализации и анализа данных, очень сложный. Нужно создавать каждый график отдельно для их объединения на одни шкалы. Тяжело просматривать данные приходящие в реальном времени. Нет возможности объединить графики с разных серверов на одну шкалу для сравнения. Данный инструментарий очень важен для анализа как исторических данных, так и для данных обрабатываемых в реальном времени. Поэтому был найден удобный и легковесный инструмент для визуализации и анализа данных в разрезе времени Grafana. Grafana – прекрасная альтернатива информационным панелям Zabbix. Она как раз специализируется на визуальном представление данных собранных в разрезе времени. Позволяет работать с информационными панелями и графиками на основе данных из различных систем хранения, в том числе на основе Zabbix API.

Grafana — редактор графиков и информационных панелей, базирующийся на данных из Graphite, InfluxDB или OpenTSDB, специализирующийся именно на отображении и анализе информации. Он лёгок, относительно прост в установке. Функционально Grafana представляет собой набор пользовательских информационных панелей, разделённых на строки задаваемой пользователем высоты, в которых, в свою очередь, можно создавать функциональные элементы (графики, html-вставки и плашки-триггеры). Строки можно перемещать, переименовывать. Естественно, главными функциональными элементами информационных панелей являются графики. Создание нового графика-метрики в Grafana совершенно не требует каких-либо специальных знаний.

Для интеграции Grafana и Zabbix есть свободно распространяемый модуль. После его установки в источниках данных для отображения появляется Zabbix. Пример информационной панели в Grafana на основе данных из Zabbix можно увидеть на рисунке 3.

Объем необходимой памяти диска, очевидно, зависит от количества узлов сети и параметров, которые будут контролироваться. Если планируется сохранять историю контролируемых параметров длительный срок, то необходимо иметь ввиду, что по крайней мере нужно иметь несколько гигабайт для хранения данных истории в базе данных.

Рисунок 3. Пример информационной панели

Каждый процесс демона Zabbix требует несколько подключений к базе данных. Объем памяти, необходимой для подключения к базе данных, зависит от настроек базы данных. Zabbix и особенно база данных может потребовать значительных процессорных ресурсов в зависимости от количества контролируемых параметров и выбранной базы данных.

Основные возможности разработанной системы мониторинга

1. Сбор параметра свободного места на диске.

Для сбора параметра свободного места на диске был написан скрипт, предназначенный для слежения за свободным дисковым пространством: каждый сервер опрашивается по WMI для генерирования набора дисков, после чего, вновь с использованием WMI, запрашивается информация о свободном и занятом пространстве. Скрипт включает в себя метод Get-WmiCustom, который расширяет стандартный командлет Get-WMIObject дополнительным параметром – таймаут запроса. Если за время, которое ограничено этим параметром, ответ от сервера не был получен, то генерируется исключение, содержащее сообщение о недоступности машины и код выхода из скрипта изменяется на ошибочный. Такое решение позволяет избежать зависания скрипта при различных проблемах сети, а также одновременно проверять доступность хостов и работоспособность на них службы WMI. В методе создается ManagementObjectSearcher, который извлекает коллекцию управляющих объектов в соответствии с заданным запросом. Этот класс является одной из наиболее часто используемых исходных точек входа для извлечения сведений об управлении. Например, его можно использовать для перечисления всех дисководов, сетевых адаптеров, процессов и прочих управляющих объектов в системе, либо для запроса всех работающих сетевых подключений, приостановленных служб и т. д. После создания экземпляр этого класса принимает в качестве исходных данных запрос WMI, представленный объектом ObjectQuery или его производными, и при необходимости объектом ManagementScope, представляющим пространство имен WMI, в котором должен быть выполнен запрос. Экземпляр может также принять дополнительные параметры, которые содержатся в объекте EnumerationOptions. При вызове метода Get() для этого объекта ManagementObjectSearcher выполняет данный запрос в заданной области и возвращает коллекцию управляющих объектов, которые удовлетворяют запросу, в ManagementObjectCollection.

2. Проверка задач в планировщике заданий.

Для проверки настроек и состояния задач определённого типа в планировщике заданий Windows был написан скрипт, который в случае не выполнения задачи, принудительно запускает её.

Скрипт создаёт объект типа Schedule.Service, который предоставляет доступ к планировщику заданий для управления зарегистрированными задачами. Сначала используется метод Connect() для подключения к удалённой машине и ассоциации всех последующих вызовов на этом интерфейсе с этой удалённой сессией. Затем применяется метод GetFolder(), который получает все содержимое зарегистрированных заданий.

Для нужд проекта требуется, чтобы постоянно были запущены задания, у которых триггер срабатывает при включении сервера и зарегистрированный статус задачи не равен «Disabled».

Необходимо проверить, что задача, подходящая по этим параметрам, запущена и у нее отсутствует настройка автоматической остановки через трое суток после запуска. Для этого необходимо сравнить различные свойства объекта задачи (Task.Definition) с этими значениями.

Если же задача остановлена, то нужно изменить код выхода на ошибочный и попытаться запустить задание методом Run().

3. Проверка оперативной памяти.

Проверка свободной оперативной памяти производится подобно вышеописанному скрипту по контролю свободного места на диске: по выборке компьютеров из Active Directory проводится сбор

сведений с помощью WMI запроса. Результаты обрабатываются по определённым условиям. Однако, кроме пропуска машин из специальной переменной SkipHosts, необходимо также не проверять машины с установленной службой виртуализации Hyper-V, т.к. наличие оперативной памяти на хосте для виртуализации – не важный показатель, интерес представляют только значения на самих виртуальных машинах. Определение наличия службы виртуализации также осуществляется через wmi запрос. Если объём доступной оперативной памяти на сервере менее 10%, то об этом поступает сообщение в лог с использованием функции Write-Error, а затем переменная выхода из скрипта принимает значение равное 1, чтобы при окончании выполнения сообщить об ошибке при сборке этого шага конфигурации

4. Очистка логов.

Для просмотра и анализа всевозможных логов в проекте используется связка Logstash-Elasticsearch-Kibana.

Logstash – бесплатная open-source программа на java для сбора и нормализации логов. Может принимать логи либо с локальных файлов, либо через tcp/udp порты. Разных входных (input) фильтров насчитывается 26. Также есть входной модуль, для сбора сообщений из twitter'a или irc.

ElasticSearch – бесплатный open-source поисковый сервер, основанный на Apache Lucene. Быстрый, легко настраиваемый и очень масштабируемый.

Kibana – веб-интерфейс, написанный на ruby, для отображения данных из ElasticSearch. Простая настройка, но множество функций – поиск, графики, stream.

Мониторингу требуется просматривать датирование логов, и, если существуют устаревшие, то удалять их. Управление логами осуществляется через HTTP API. Для выдачи статистики логов достаточно отправки HTTP-GET запроса, для удаления – HTTP-DELETE. Для каждого из запросов в скрипте реализованы отдельные методы. Первый метод Execute-HTTPGetCommand(), в котором создаётся объект класса System.Net.WebRequest, который выполняет запрос к заданному URI (Uniform Resource Identifier). Затем запрашивается ответ на запрос, результат которого считывается символами из потока байтов в определённой кодировке (System.IO.StreamReader).

Для получения параметров на основе логов, находящихся в системе elasticsearch был написан скрипт, листинг которого приведен ниже.

Листинг 1. Скрипт для получения параметров на основе логов, находящихся в системе elasticsearch

```
    #UserParameter = es.iis_rs_ms.url.percentile[*],PowerShell.exe -executionpolicy bypass -file C:\Zabbix\es.iis_rs_ms.percentile.ps1 -Appname "$1" -Url "$2" -Percentile "$3" -MinuteAgo "$4"
    #UserParameter = es.iis_rs_ms.query.percentile[*],PowerShell.exe -executionpolicy bypass -file C:\Zabbix\\es.iis_rs_ms.percentile.ps1 -Query "$1" -Percentile "$1" -MinuteAgo "$2"
    param(
        [string]$Appname,
        [string]$Query,
        [string]$Url,
        [string]$Percentile,
        [string]$MinuteAgo
    )
    if([string]::IsNullOrEmpty($Query))
    {
      if([string]::IsNullOrEmpty($Appname))
      {
        $Query = "request_hash_raw:\""$Url\""";
      } else {
        $Query = "appname: $Appname AND request_hash_raw:\""$Url\""";
      }
    }
    if([string]::IsNullOrEmpty($Percentile))
    {
      $Percentile = "75";
    }
    if([string]::IsNullOrEmpty($MinuteAgo))
    {
      $MinuteAgo = "15m";
    }

    [string]$responseBody = @"
    {
```

```
  "size" : 0,
  "query" : {
    "filtered" : {
      "query" : {
        "query_string" : {
          "query" : "$Query",
          "analyze_wildcard" : true
        }
      },
      "filter" : {
        "bool" : {
          "must" : [{
              "range" : {
                "@timestamp" : {
                  "gte" : "now-$MinuteAgo"
                }
              }
            }
          ],
          "must_not" : []
        }
      }
    }
  },
  "aggs" : {
    "1" : {
      "percentiles" : {
        "field" : "response_time_ms",
        "percents" : [$Percentile]
      }
    }
  }
}
"@;
$apiUrl = 'http://localhost:9200/iis_*/iis/_search'
$client = new-object System.Net.WebClient
$client.Encoding = [System.Text.Encoding]::UTF8
$reponse = $client.UploadString($apiUrl, $responseBody)
$value = $reponse -match '.*"values":\{"[\d\.]*":(?<value>[\d\.]*)\}.*'
Write-Host $Matches.value
```

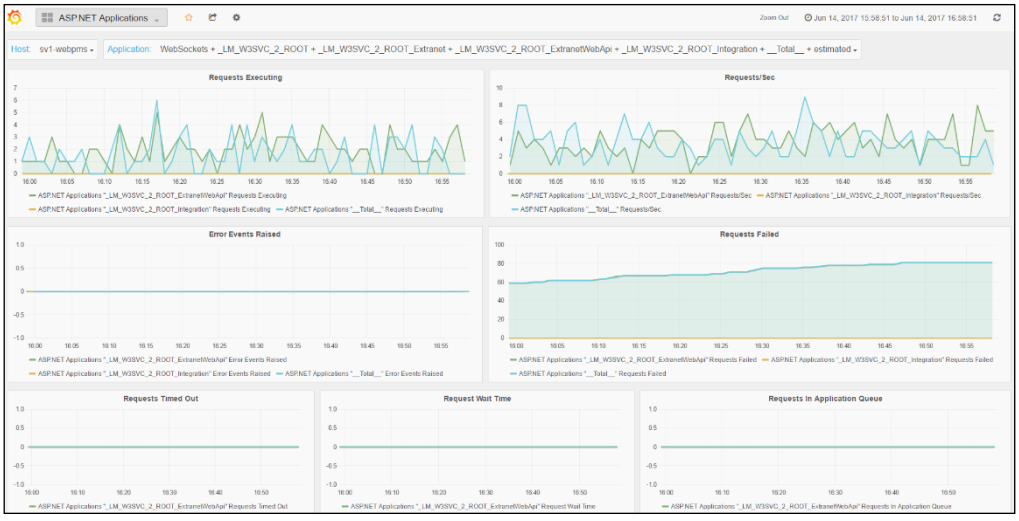

Рисунок 4. Информационная панель для просмотра ASP NET приложений, находящихся в IIS веб-сервере

Для внедрения данной системы мониторинга необходимо установить в дата-центре ПО Zabbix на

виртуальные машины с ОС Linux, после чего настроить через Zabbix веб-интерфейс виртуальные машины со стандартными шаблонами, зависящими от ОС. Также необходимо настроить уведомления о важных событиях по email и смс и затем настроить информационные панели в Grafana. На рисунках 4-7 приведены основные графики работы системы мониторинга.

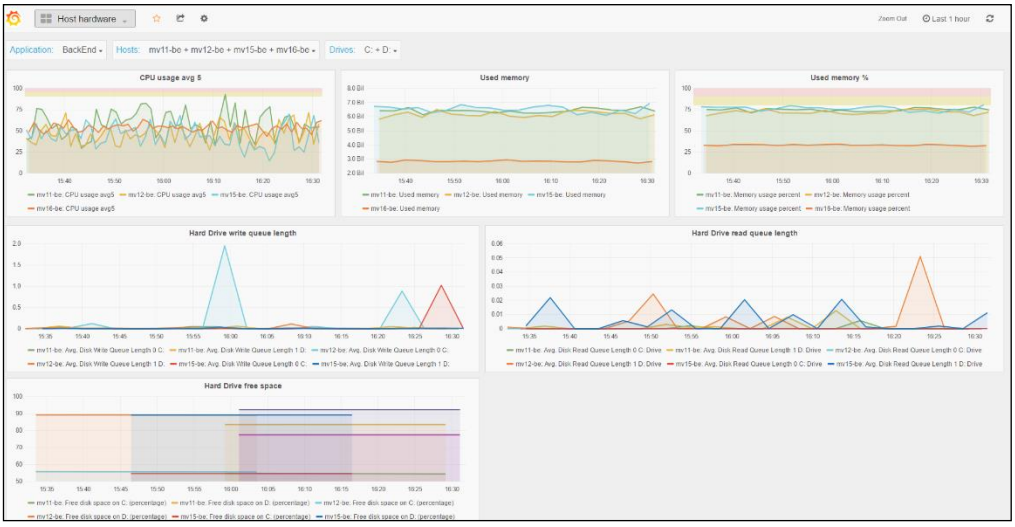

Рисунок 5. Информационная панель для просмотра характеристик виртуальных машин

Рисунок 6. Информационная панель для просмотра характеристик базы данных на основе СУБД MySQL

Рисунок 7. Информационная панель для просмотра характеристик сервера отправки почты

Отметим, что администратор сам может настроить любые панели просмотров характеристик мониторинга состояния программно-аппаратного обеспечения. Это могут быть: информационная панель для просмотра характеристик очередей продукта компании; информационная панель для просмотра программных характеристик продукта компании; информационная панель для просмотра бизнес характеристик продукта компании; информационная панель для просмотра характеристик IIS App pools и т.п.

Заключение

Бизнес в любой сфере сильно зависит от доступности и работоспособности его ИТ-инфраструктуры 24/7/365. Чтобы обеспечить эту работоспособность, необходимо заранее выявлять узкие места в конфигурации систем и сетей, а также быстро узнавать о наличии поломки и её причине. Для этих нужд в компаниях, где подобное слежение неосуществимо за счёт только специалистов, принято использовать системы мониторинга.

Предложенная система мониторинга разработана на основе Zabbix. Для анализа и визуализации данных в разрезе времени к Zabbix был интегрирован удобный и легковесный инструмент Grafana. Для получения сбора данных из источников ЦОД предприятия разработаны частные скрипты.

Использование данной системы позволит быстро среагировать на любую неисправность за счет уведомлений, а преднастроенные панели визуализации характеристик систем позволяют довольно точно определить источник проблемы и метку времени начала ухудшения характеристик. Благодаря всему этому бизнес-риск простоя сервисов услуг компании значительно снизится. Например, многие проблемы, которые в итоге приводили к неработоспособности сервисов, можно определить заранее, еще до того, как об этом узнают пользователи. Предлагаемый продукт для мониторинга серверов и служб, а также автоматизации задач поможет значительно снизить время простоя сервисов, повысить качество услуг, предоставляемых клиенту компанией, а также значительно улучшить производительность, избежав, при этом, неоправданных вложений. В будущем можно улучшить качество мониторинга за счет постоянного выполнения популярных сценариев использования сервисов.

Литература

1. Nagios [Электронный ресурс]: Nagios Documentation // URL: http://www.nagios.org/ (дата обращения 1.10.2017).
2. Zabbix Documentation [Электронный ресурс]: Zabbix documentation for version 2.0. // URL: http://www.zabbix.com/documentation/2.0 (дата обращения 1.10.2017).
3. Далле Вакке А. Zabbix. Практическое руководство / пер. с англ. А. Н. Киселева – М.: ДМК Пресс, 2017. – 365 с.

References

1. Nagios [Electronic resource]: Nagios Documentation // URL: http://www.nagios.org/ (the date of reference 1.10.2017).
2. Zabbix Documentation [Electronic resource]: Zabbix documentation for version 2.0. // URL: http://www.zabbix.com/documentation/2.0 (the date of reference 1.10.2017).
3. Dalle Vakke A. Zabbix. Code of practice / пер. with англ. A. N. Kiseleva, is M.: ДМК Press, 2017. – 365 с.

Об авторах:

Никитин Петр Владимирович, кандидат педагогических наук, доцент кафедры прикладной математики и информатики и кафедры методики обучения математики и естественнонаучных дисциплин, Марийский государственный университет; доцент кафедры информационной безопасности, Межрегиональный открытый социальный институт, petrvlni@rambler.ru

Горохова Римма Ивановна, кандидат педагогических наук, доцент, доцент кафедры методики обучения математики и естественнонаучных дисциплин, Марийский государственный университет, gorokhova-ri@yandex.ru

Note on the authors:

Nikitin Petr V., Candidate of Pedagogic Sciences, Assist. Prof. Dr, Mari State University; associate Professor of information security, Interregional Open Social Institute, petrvlni@rambler.ru

Gorokhova Rimma I., Candidate of Pedagogic Sciences, Prof. Dr, Mari State University, gorokhova-ri@yandex.ru