УДК 004.75

Д.В. Жевнерчук, П.А. Родионов, А.С. Захаров

ИССЛЕДОВАНИЕ ИНТЕРОПЕРАБЕЛЬНОСТИ СИСТЕМ МОНИТОРИНГА ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ И РЕСУРСОВ

Нижегородский государственный технический университет им. Р.Е. Алексеева

Определен круг задач анализа и прогнозирования загрузки вычислительного ресурса неоднородными процессами. Предлагается методика самообследования процессов в многозадачной вычислительной среде, которая включает хронометраж алгоритмов, методов, групп методов, открытую систему мониторинга вычислительного ресурса и процессов, хранилище данных OLAP и модели data mining.

Ключевые слова: интероперабельность, открытая система, мониторинг, olap, data mining, большие массивы данных, java.

Введение

Современные вычислительные системы обладают свойствами, которые еще 10–15 лет назад относились к разряду второстепенных или вообще не рассматривались. Высокая динамика автоматизируемых бизнес-процессов и скорость адаптации к возможным структурным и функциональным изменениям, виртуализация приложений и рабочих мест, масштабируемость от уровня локальной машины до глобальных сетей, режим 24x7 – все это диктует новые требования к технологиям, методам и средствам проектирования и развертывания вычислительных систем.

Одним из ключевых параметров является интервал времени между постановкой задачи на создание системы и выходом конечного продукта с заданными свойствами, готового к эксплуатации. Для минимизации этого времени необходимо сконцентрировать внимание на процессах автоматизации, которые включают:

- повышение интероперабельности моделей вычислительных систем;
- стандартизацию баз знаний по технологиям и методам проектирования и алгоритмизации;
- генерацию одних моделей вычислительных систем на основе других;
- генерация кода систем и их сборка;
- мониторинг и исполняемого кода системы в однозадачном и многозадачном режимах.

Все выделенные процессы позволяют обеспечить свойства открытости [1, 2] для создаваемой вычислительной системы.

Современные аппаратные облачные системы формируют основу для создания гетерогенных вычислительных сред, в которых могут функционировать множество программных систем. Другими словами, различные вычислительные системы могут иметь пересекающиеся множества аппаратных и программных ресурсов.

Настоящая публикация посвящена проблеме обеспечения масштабируемости такой сложной динамической системы и является дальнейшим развитием методов и средств мониторинга сложных гетерогенных вычислительных систем.

Постановка задачи

Необходимо разработать комплекс методов и средств обеспечения мониторинга системы, которые могут обеспечить следующий результат:

- оценить среднее время выполнение произвольного фрагмента кода системы в однозадачном и многозадачном режимах под управлением произвольных программно-аппаратных платформ с поддержкой виртуализации;
- сформировать группы вычислительных систем, использующих один и тот же ресурс на общем интервале времени;

- построить функциональную зависимость эффективности функционирования вычислительной системы от множества вычислительных систем, разделяющих общий программно-аппаратный ресурс и от приложенной общей нагрузки;
- классифицировать новую вычислительную систему;
- выполнить первичную кластеризацию вычислительных систем.

Для решения задачи необходимо разработать комплексную методику и систему сбора и аналитической обработки параметров функционирования вычислительных систем при различных условиях.

Среди систем мониторинга, наиболее близко отвечающих требованиям, известные следующие:

- Munin,
- PandoraFMS,
- AggreGate Network Manager.

Munin – это приложение для мониторинга серверов и обычных клиентских компьютеров под управлением Linux, написанное на языке Perl. Программа создает вывод изменений характеристик системы в виде графиков, встроенных в html страничку. По умолчанию осуществляется мониторинг использования файловой системы, памяти, процессора, активности сетевых служб и др.

Система состоит из двух независимых частей. Первая часть – это сервер, устанавливаемый на машину администратора. Именно на этот компьютер и будут собираться все данные. Вторая часть – это небольшой агент munin-node, настраиваемый на машинах, которые будут анализироваться. Сам этот демон представляет собой, как это было описано ранее, небольшой Perl-скрипт, который слушает 4949 порт с помощью Net::Server.

Данная система поддерживает плагины, которые устанавливаются в папку сервера (/etc/munin/plugins) и запоминает их имена. Раз в 5 мин сервер munin подключается ко всем агентам, получает информацию от всех плагинов и сохраняет себе в базы rrdtool. Одним из условий работы клиентской части munin является наличие любого установленного web сервера.

Система Pandora FMS является системой мониторинга с открытым исходным кодом. Pandora FMS позволяет осуществлять мониторинг с визуализацией состояний и производительностью нескольких параметров из различных операционных систем, серверов приложений и аппаратных систем, таких как брандмауэры, прокси, баз данных, веб-серверов или маршрутизаторов.

Pandora FMS могут быть развернуты практически в любой операционной системе. Мониторинг осуществляется по средствам серверов (в том числе TCP, UDP, ICMP, HTTP) и агентов. Агенты доступны для каждой платформы. Из общего числа возможностей данной системы можно выделить;

- обнаружение новых систем в сети;
- создание в реальном времени отчетов и графиков;
- SLA отчетности;
- графики в реальном времени для каждого модуля;
- поддерживает до 2500 модулей на сервере;
- многопользовательские, многопрофильное, групповые;
- система событий с пользовательской проверкой для работы в группах;
- детализация доступа и пользовательские профили для каждой группы и каждого пользователя. Профили могут персонализировано использоваться с количеством атрибутов безопасности до 8, без ограничений по группам или профилям.

Система AggreGate Network Manager. разрабатывается и поддерживается российским R&D офисом компании Tibbo Technology. AggreGate Network Manager является единственной российской системой мониторинга сетей, составляющей конкуренцию продуктам крупнейших мировых производителей подобных систем.

В частности, серьезными плюсами являются мощные средства анализа SNMP и WMI-данных, интегрированный редактор отчетов, SDK с открытым исходным кодом и пр. Система включает сотни компонентов для сетевого управления (тревоги, отчеты, диаграммы, карты сети, инструментальные панели, выполняемые по расписанию задачи, и т.д.).

Особенности программы AggreGate Network Manager:

- простота интеграции с другими средствами автоматизации зданий;
- поиск сетевых устройств;
- сигналы тревоги в случае ошибок;
- построение графиков в режиме реального времени;
- фильтрация событий;
- AggreGate Network Manager имеет удобный интерфейс.

Общим недостатком всех рассмотренных систем является отсутствие механизмов мониторинга вызовов и хронометража исполняемого кода вычислительных систем.

Методика и средства

Вычислительные системы S_1, S_2, …, S_j назовем *пересекающимися по ресурсу R*, обозначим S^R, если существует интервал времени Δt, на котором S_i удерживает ресурс r_i, а S_j удерживает ресурс r_j, причем $r_i, r_j \in R$.

Обеспечивается поддержка опционального режима самообследования вычислительной системы, в котором выполняется фиксация меток времени начала t_0^{fc} и завершения t_n^{fc} фрагмента исполняемого кода *fc*. Очевидно, что разные замеры времени выполнения одного и того же фрагмента кода будут отличаться друг от друга и в общем случае зависят от конфигурации аппаратной платформы *A*, от множества вычислительных систем, являющихся пересекающимися по ресурсу *R* с исследуемой, причем $R \subseteq A$, от нагрузки *Q*, приложенной ко всем пересекающимся по *R* системам.

Таким образом, $\Delta t_k^{fc} = t_n^{fc} - t_0^{fc} = F(R,Q,S^R)$, и для комплексного анализа масштабируемости вычислительных систем, выполняющихся на платформе *A*, необходимо обеспечить сбор $t_n^{fc}, t_0^{fc}, R, Q, S^R$, хранение их в форматах, наиболее удобных для аналитической обработки, включая поддержку автоматического поиска зависимостей.

Параметры t_n^{fc}, t_0^{fc} могут быть получены с помощью дополнительной функциональности в программном коде. Для этого используется сервисный класс или библиотека, содержащая методы (функции) разметки и выполнения хронометража кода, далее *Chronometer*. В каждом модуле, который содержит код, требующий хронометража, создается объект класса *Chronometer*. Участок исследуемого кода обрамляется методами *start_chron()* и *stop_chron()*. Вызовы этих методов порождают события *starttime_event*, *stoptime_event* и запускают на выполнение их обработчики, которые считывают текущее время, тип метки, идентификатор вычислительной системы, идентификатор *fc* и передают эти сведения в стандартный поток вывода. В среде разработки могут быть использованы дополнительно конструкторы режима самообследования, которые включают визуальные инструменты разметки *fc* и генерации соответствующих программных конструкций.

Общая нагрузка *Q* на систему S^R определяется множествами пользовательских и программных запросов.

Оценке пользовательской нагрузки посвящены работы [3–5]. Для отслеживания программной нагрузки предлагается на уровне кода ввести маркировку запросов и ответов.

Вычислительная система, выполняющая запрос, является клиентом, отвечающая на запрос – сервером. Маркер запроса включает информацию об идентификаторе клиента, запрашиваемого *fc*, идентификаторе сервера, момента времени, когда выполнялся запрос. Маркер ответа включает информацию об идентификаторе клиента, идентификаторе сервера, момента времени, когда был передан ответ. Маркеры передаются стандартным потоком вывода и могут быть обработаны системами мониторинга. Запрос и ответ передаются между

вычислительными системами. Формирование и маркера и вывод его в стандартный поток осуществляется сервисной библиотекой функций либо классом, далее *QueryResultMarker.*

Для управления процессом сбора данных локальной платформы либо сетевого ресурса применяется многоагентная система. Серверная часть включает агенты, установленные на компьютер для сбора информации. Агенты ожидают подключение клиента и по его команде начинают сбор данных, который проводится в режиме реального времени. Результаты сбора информации сохраняются в структурированные файлы отчетов.

Координатор процесса сбора параметров вычислительной системы выполняет следующие функции:

- активации/деактивации агентов;
- отображения информации о состоянии агентов;
- экспорта данных в структурированные файлы отчетов;
- сбора сведений о доступных утилитах;
- формирования расписания активации утилит.

Информация о ресурсах платформы, включая доступные и удерживаемые, собирается средствами операционных систем и системными утилитами.

Собираемые данные приводятся к формату гиперкубов [6], что позволит:

- естественным образом формировать аналитические структуры, поддающиеся автоматизированной обработке;
- разрабатывать модели Data Mining [7, 8], для построения классификаторов и неочевидных функциональных зависимостей в собираемых данных с целью формализованного описания вычислительных систем и выявления оптимальных S^R.

Современные системы OLAP и Data Mining ориентированы на хранение и обработку данных, которые изначально могут быть описаны в различных форматах, т.е. они обладают высоким коэффициентом интероперабельности.

Система мониторинга является комплексной, базирующейся на интероперабельности отдельных программных решений, и в общем случае может включать в свой состав следующие компоненты:

- сервисный класс *Chronometer;*
- сервисный класс *QueryResultMarker;*
- генераторы программного кода;
- конструкторы режима самообследования;
- координатор процесса сбора параметров вычислительной системы;
- утилиты сбора данных об использовании ресурсов узлов;
- временное хранилище собираемых параметров вычислительной системы;
- ETL утилиты;
- процессоры OLAP-системы;
- модели и процессоры Data Mining;
- OLAP хранилище данных;
- Data Mining хранилище данных;
- система визуализации аналитических моделей.

Концептуальная модель комплексной открытой системы мониторинга представлена на рис. 1.

В общем случае выделенные компоненты принадлежат разным системам и могут выполняться под управлением различных платформ. Генераторы программного кода и конструкторы режима самообследования играют вспомогательную роль и являются элементами сред разработки, поэтому они не вынесены на схему.

ETL утилита предназначена для передачи данных из временного хранилища системам OLAP и Data Mining. Она может являться частью аналитических платформ либо подсистемой координатора сбора данных, либо самостоятельной программой. Временное хранилище существует на каждом узле, где происходит мониторинг исполняемого кода и ресурсов, потребляемых вычислительными системами, и реализовано множеством структурированных файлов, для которых выделены специальные системные папки. Для повышения интероперабельности разрабатываемого комплексного решения, не существует жестких ограничений на формат временного хранилища.

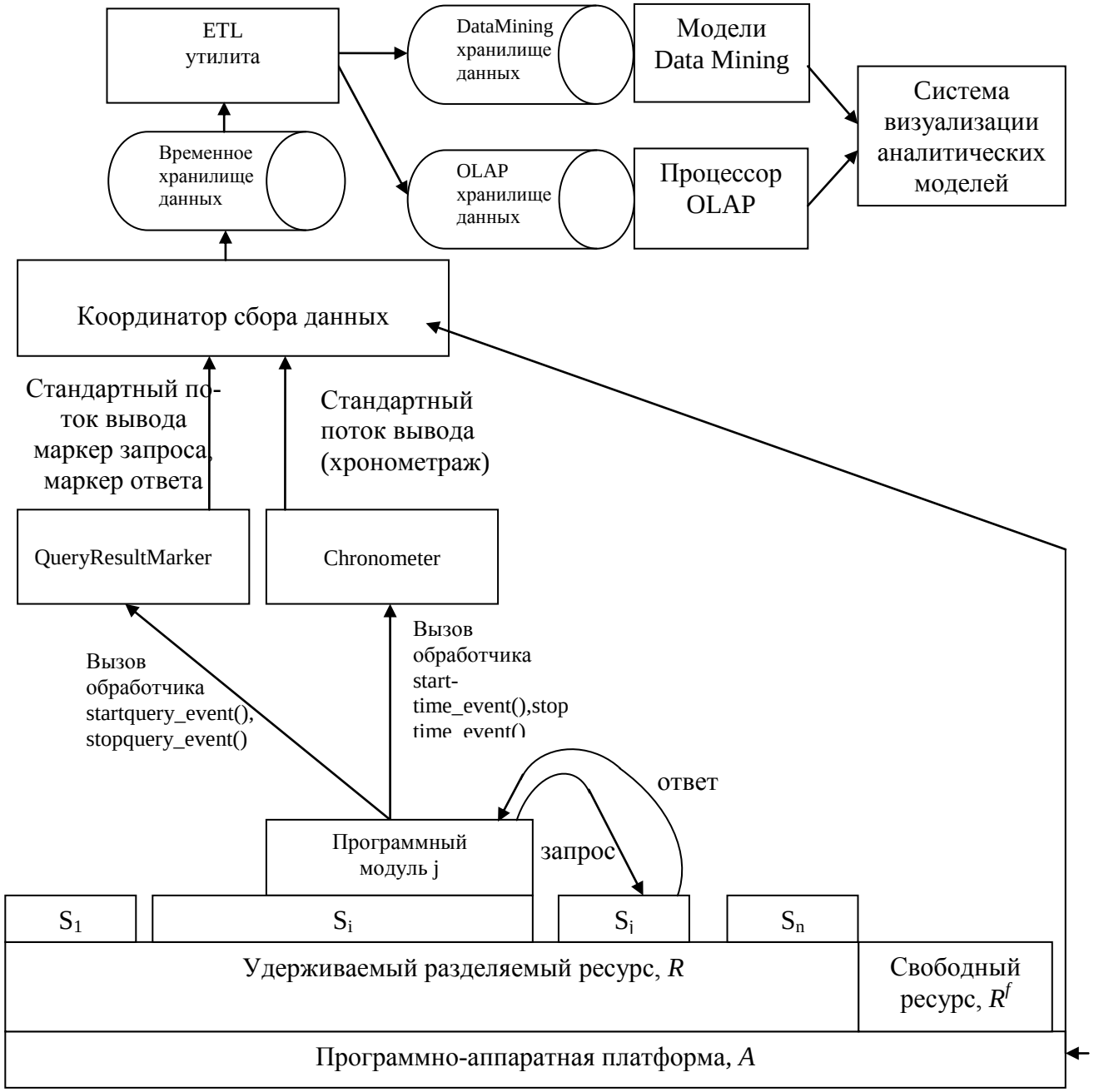

Рис. 1. Архитектура комплексной открытой системы мониторинга

В зависимости от реализации OLAP и Data Mining систем конечные форматы хранения данных и построенных на их основе аналитических моделей могут варьироваться. Модели Data Mining могут быть построены и на основе хранилища данных OLAP.

Системы визуализации данных, как и ETL утилиты, могут быть либо самостоятельными, либо являться частью аналитических компонент комплексной среды мониторинга.

Экспериментальная часть

Предложенная методика была использована при построении прототипа комплексной системы мониторинга исполняемого кода и потребляемых ресурсов вычислительных систем.

Разработаны классы хронометража и мониторинга запросов для Java. Тестовая вычислительная система включает три класса, каждый из которых включает два-три хронометрируемых метода. Отдельные экземпляры системы обмениваются данными по запросу.

Испытания системы проводились с использованием следующих программных и аппаратных платформ и компонентов:

- процессор: Intel Pentium 4 или аналогичный AMD Athlon;
- оперативная память: 128 МБ; 64 МБ для Windows XP (32-разрядная версия);
- жесткий диск: 200Мб свободного места;
- консоль 80x25 в текстовом режиме;
- периферия: наличие клавиатуры;
- Windows 8 (Настольные ПК), Windows 7, Windows Vista SP2, Windows XP SP3 (32-разрядная версия); Windows XP SP2 (64-разрядная версия), Windows Server 2008, Windows Server 2012 (64-разрядная версия);
- Linux Oracle Linux 5.5+, Oracle Linux 6.x (32-разрядная версия)*, 6.x (64-разрядная версия)**, Red Hat Enterprise Linux 5.5+, 6.x (32-разрядная версия)*, 6.x (64-разрядная версия)**, Ubuntu Linux* 10.04 и выше, Suse Linux Enterprise Server* 10 SP2, 11.x;
- Java Standard Edition 1.7,
- Process Explorer,
- Weka [8].

В режиме самообследования метки времени выполнения методов и ожидания отклика фиксируются в csv-файлах. Также фиксируются ресурсы, потребляемые каждым Java процессом.

Координатор сбора включает агенты сбора системной информации о потребляемых ресурсах и агенты, конвертирующие данные из csv файлов в arff файлы, с которыми работает система Data Mining Weka.

В ходе экспериментального исследования установлено, что прототип обладает высокой степенью интероперабельности:

- функционирование на программно-аппаратных платформах, поддерживаемых JVM;
- отсутствуют ограничения на форматы временного хранилища, которые определяются на уровнях координатора сбора данных и классов Chronmeter и QueryResultMarker;
- отсутствуют ограничения на интерфейсы взаимодействия с аналитическими системами;

Кроме того, среда хорошо расширяема по аналитическим моделям. В общем случае можно добавить любые методы обработки структурированных данных хронометража и потребления ресурсов.

Система является масштабируемой благодаря агентной архитектуре. Поддержка CLI упрощает процедуру управления агентами, функционирующими в распределенной вычислительной среде.

Выводы

Предложенная методика является эмпирической, не содержит допущений или ограничений, которые могут повлиять на точность конечного результата. Экспериментально установлено, что созданный прототип является открытой вычислительной системой: *переносимый (кроссплатформенный)* между широким кругом программно-аппаратных платформ, *расширяемый* по аналитическим компонентам, утилитам, поддерживаемым форматам хранения данных, *масштабируемый* благодаря агентной архитектуре, интероперабельный, благодаря чему практически сняты ограничения на организацию внутреннего межкомпонентного взаимодействия, так и взаимодействия с внешними системами.

Методика может быть применена:

- для тестирования программного обеспечения, функционирующего в монопольном и многозадачном режимах;
- балансировки нагрузки серверного ресурса, обрабатывающего произвольные группы процессов;
- построения моделей прогнозирования поведения вычислительных систем.

В перспективе развитие прототипа комплексной системы мониторинга исполняемого кода и потребляемых ресурсов вычислительных систем будет проходить в направлениях:

- создания генераторов кода для режима самообследования;
- создания конструкторов для поддержки режимов самообследования: визуализация хронометрируемых участков кода и форматов временного хранилища;
- портирования классов Chronometer и QueryResultMarker в другие среды разработки программного обеспечения.

Библиографический список

1. **Сухомлин, В.А.** Методологический базис открытых систем // Открытые системы. 1996. № 4.
2. Технология открытых систем / под ред. А.Я. Олейникова. – М.: Янус-К, 2004.
3. **Жевнерчук, Д.В.** Моделирование трафика открытой информационной системы с трансляцией GUI в потоковом видео // Инфокоммуникационные технологии. 2011. №4. С 46–52
4. **Жевнерчук, Д.В.** Методика моделирования нагрузки на сервер в открытых системах облачных вычислений / Д.В. Жевнерчук, А.В. Николаев // Информатика и ее применения. 2012. Т. 6. Вып. 2. С. 99–106.
5. **Жевнерчук, Д.В.** Программный генератор трафика пользователей ресурса виртуальных лабораторий, Программные продукты и системы/ Д.В. Жевнерчук, А.В. Николаев // Программные продукты и системы. 2012. №3. С 31–38.
6. **Паклин, Н.Б.** Бизнес-аналитика: от данных к знаниям (+ CD) / Н.Б. Паклин, В.И. Орешков. – СПб.: Питер, 2009. – 624 с.
7. **Тоби**, **Сегаран**. Программируем коллективный разум / Тоби Сегаран. – М.: Символ-Плюс, 2012. – 368 с.
8. **Ian H. Witten**. Data Mining: Practical Machine Learning Tools and Techniques / Ian H. Witten, Eibe Frank, Mark A. Hall // Third Edition The Morgan Kaufmann Series in Data Management Systems, MK Morgn Laufmann, 2011. – 421 p.

Дата поступления
в редакцию 01.02.2014

D.V. Zhevnerchuk, P.A. Rodionov, A.S. Zaharov

RESOURCE AND COMPUTING PROCESSES MONITORING SYSTEMS INTEROPERABILITY RESEARCH

Nizhny Novgorod state technical university n.a. R.E. Alexeev

Subject: The subject of this study is the interoperability of resource and computing processes monitoring systems.
Purpose: The aim is to create methods and instrumental environment for computing processes and resources monitoring with high valued interoperability parameter.
Design/methodology/approach: A theoretical framework is proposed based on methodology of open systems in computer science and design of network software systems, OLAP and Data Mining, Software engineering.
Findings: The results can be applied to the design and development of scalable, extensible integrated monitoring systems whose components operate in heterogeneous distributed computing environment.
Research limitations/implications: The present study provides a foundation for interoperable monitoring systems development.
Originality/value: Monitoring systems have two components: components, embedded in applications for collecting temporal parameters and components for the statistics collection and analytical models construction. The study defined technical basis to ensure the interoperability of monitoring systems which can collect and process temporal parameters for application's different executable parts.

Key words: interoperability, open system, monitoring, OLAP, Data Mining, large data sets, java.