ВОПРОСЫ ПРОГРАММНОЙ АРХИТЕКТУРЫ СИСТЕМЫ УЧЕТА ВРЕМЕНИ ИСПОЛЬЗОВАНИЯ КОМПЬЮТЕРНОЙ ТЕХНИКИ В УЧЕБНЫХ ЛАБОРАТОРИЯХ

В.В. Новый
Витебск, ВГУ имени П.М. Машерова

В рамках очередного этапа выполнения работ по теме государственной программы научных исследований и в продолжение [1] предыдущего этапа работ, связанных с автоматизацией анализа использования ресурсов компьютерной техники в учебных лабораториях университета, решалась проблема сбора необходимых данных с операционных систем семейства GNU/Linux и выполнялось проектирование взаимодействия клиентских модулей с сервером аналитики. Актуальность указанных проблем связана с расширяющимся использованием в современных условиях альтернативных семейству Microsoft Windows операционных систем и, в первую очередь, операционных систем на базе ядра Linux.

Таким образом, целью представленной работы является обсуждение вопросов, связанных с проектированием архитектуры клиентской части системы учета времени использования компьютерной техники в учебных лабораториях в операционных системах семейства GNU/Linux, а также анализ подходов к реализации сетевой подсистемы взаимодействия с серверной частью системы автоматизации с учетом работы этой системы в рамках сети университета.

Материал и методы. Объектом представленного исследования выступают подходы к получению, обработке и передаче данных, необходимых для учёта использования вычислительной техники под управлением операционных систем семейства GNU/Linux пользователями. При выполнении исследования используются как общенаучные методы исследования, такие как математическое моделирование, системный анализ, сравнение, так и частные методы программной инженерии и сетевого программирования.

Результаты и их обсуждение. Общая цель исследования соответствует описанной в [1] ранее, но распространяется на операционные системы, основанные на ядре Linux и применяемые длительное время в целом ряде дисциплин кафедры прикладного и системного программирования. При условии использования в ходе реализации клиентской части системы вызовов, совместимых со стандартом SUS v3, полученное решение также может быть легко перенесено на операционные системы Apple MacOS.

Первая из решаемых в рамках исследования задач – это определение источника данных для получения информации о работающих и работавших в системе пользователях. При этом необходимо отличать учетные записи обычных пользователей системы от учетных записей системных пользователей, используемых для работы системных демонов (сервисов). При этом фиксироваться должны успешные попытки входа в систему и выхода из нее.

Получение информации об использовании компьютерной системы под управлением операционной системы семейства GNU/Linux пользователем может быть выполнено несколькими способами. Рассматривались следующие варианты получения информации:

– прямое отслеживание создания сеанса пользователя в системе;

– использование для этого подсистемы auditd и вывода команды aureport;

– использование файлов протоколов /var/run/utmp (вошедшие в настоящий момент в систему пользователи) и /var/log/wtmp (история входа пользователей в систему);

– использование данных из /var/log/auth.log (или /var/log/secure для систем-потомков операционной системы Red Hat);

– файл протокола /var/log/lastlog.

Текущая реализация основывается на получении данных из файлов протоколов utmp/wtmp, как предоставляющих необходимую информацию в удобной как для программного разбора, так и для получения через разбор вывода соответствующих команд (last/lastb).

Второй решаемой задачей является определение способа реализации клиентской части. Ряд вариантов источников данных позволяют для этой цели использовать скрипты на одном из поддерживаемых языков или даже bash-скриптинг. Однако, после анализа требовании к клиентской части была выбрана разработка в виде демона (сервиса) операционной системы на основе нативного кода.

В качестве основных требований к клиентской части выступили:

– минимальное влияние на производительность системы, так как работа клиентской части предполагается все время, пока загружена соответствующая операционная система;

– независимость от вошедших в систему пользователей (в ряде случаев серверу аналитики будет нужна информация о предыдущих входах пользователей в систему в то время, когда нет активной сессии пользователя);

– относительно небольшой объем памяти, занимаемой работающим клиентом (в связи с тем, что демон должен выполняться на компьютерах с различной конфигурацией, в том числе на достаточно слабых конфигурациях);

– минимальные зависимости для развертывания в целевой системе (как с целью экономии дискового пространства, так как в большинстве случаев GNU/Linux установлена как вторая система и достаточно ограничена по дисковому пространству, так и с целью упрощения развертывания и обслуживания установленной клиентской части);

– возможность простого доступа к сетевой подсистеме для взаимодействия с сервером аналитики и получения сетевой идентификации текущего устройства (MAC- и IP- адрес).

Отдельным вопросом выступает архитектура сетевой подсистемы для сбора данных на сервере аналитики. В качестве решения этого вопроса могут рассматриваться различные варианты:

– ведущая роль принадлежит серверу, который выполняет опрос клиентских модулей для получения необходимой информации;

– ведущая роль принадлежит клиентскому модулю, который периодически передает требуемую информацию серверу.

В случае ведущего сервера варианты реализации предусматривают либо хранение списка опрашиваемых клиентов на сервере, либо их автоматическое обнаружение в процессе опроса. В виду того, что система ориентирована на работу с учебными компьютерными лабораториями, состав которых и используемого в них оборудования является относительно постоянным первый вариант выступает предпочтительным. Отсутствие отклика от клиентского модуля в данном случае будет означать что опрашиваемый компьютер отключен или есть проблемы с его сетевым соединением.

В случае если ведущая роль принадлежит клиенту, он должен либо хранить на своей стороне информацию о серверах, которым будут отправляться данные о работе компьютерной техники, либо использовать механизм автоматического обнаружения таких серверов. Первый случай будет вести к необходимости изменения настроек на всех развернутых в данный момент клиентах при перемещении сервера, что при большом парке вычислительной техники будет вести к нежелательным временным затратам. Второй вариант может быть относительно затруднен, например, при использовании отдельных VLAN для различных сегментов внутренней сети.

Важным является вопрос минимизации накладных расходов в виде избыточного сетевого трафика, создаваемого при работе системы. Главным критерием здесь является минимальное количество сетевых обменов между клиентом и сервером и минимизация размера передаваемых данных.

Второй подзадачей является вопрос выбора режима взаимодействия:

– с сохранением состояния и установкой соединения;

– без установки соединения и без сохранения состояния.

Для первого варианта характерной является большая нагрузка на сеть, так как необходимо отслеживать состояние соединения и более сложная логика серверной части, так как при перезагрузке компьютера из одной операционной системы в другую будет требоваться повторная установка соединения и, возможно, повторный опрос для получения данных.

В зависимости от периодичности опроса должен решаться вопрос кэширования извлеченной из локальной операционной системы информации о входе/выходе пользователей системы. При высокой частоте опроса и больших объемах файлов протоколов (например, в отсутствие ротации log-файлов) клиентский модуль может порождать нежелательную нагрузку на клиентский узел за счет повторного разбора данных из источника.

Заключение. Данный этап работ предусматривал проектирование и экспериментальную реализацию клиентского модуля для операционных систем семейства GNU/Linux, а также проектирование сетевой подсистемы для взаимодействия с сервером, предоставляющим данные для обработки модулю аналитики. В настоящее время выполняется разработка нескольких вариантов реализации клиентского модуля на базе различных технологии и языков программирования и оценка эффективности подходов к построению сетевой подсистемы коммуникации между клиентами и серверным модулем. Результаты удовлетворяют заявленным требованиям к проекту. Далее планируется тестовая эксплуатация и отладка сетевой подсистемы в режиме реальной эксплуатации.

1. Новый, В.В. Об учете времени использования компьютерной техники в учебных лабораториях / В.В. Новый // Наука – образованию, производству, экономике: материалы 74-й Региональной научно-практической конференции преподавателей, научных сотрудников и аспирантов, Витебск, 18 февраля 2022 г. – Витебск: ВГУ имени П.М. Машерова, 2022. – с. 33–34. – URL: https://rep.vsu.by/handle/123456789/31552 (дата обращения: 31.01.2024).

ОБ ОПТИМИЗАЦИИ РАБОТЫ С БАЗАМИ ДАННЫХ ПРИ ИСПОЛЬЗОВАНИИ ORM НА ПРИМЕРЕ ENTITY FRAMEWORK

М.Г. Семёнов
Витебск, ВГУ имени П.М. Машерова

Современные приложения повсеместно используют базы данных для хранения информации. Для доступа приложения к базе данных классически используются такие шаблоны проектирования, как Repository, Unit of Work, DAO, ORM и другие. Однако если раньше было целесообразно разрабатывать свою реализацию шаблона для конкретного решения, то современные приложения используют готовые реализации. Одной из таких реализаций, использующейся при разработке на стеке технологий .NET, является Entity Framework.

При работе с БД в целом и с Entity Framework в частности, разработчики программного обеспечения сталкиваются с неизбежной задачей оптимизации SQL запросов для улучшения производительности. *Целью* данной работы является систематизация и анализ методов оптимизации SQL запросов, направленных на уменьшение нагрузки на базу данных и повышение общей эффективности приложения, при работе современных ORM на примере Entity Framework.