УДК 681.3:621.3(62-52)

А.А. Стенин, Ю.А. Тимошин, Т.Г. Шемсединов, Н.В. Маленко

ПРИКЛАДНАЯ ВИРТУАЛЬНАЯ МАШИНА

Аннотация: Рассматриваются проблемы создания прикладной виртуальной машины, построения на ее основе инфраструктуры приватного облака и развертывания в нем информационных систем и приложений с динамической интерпретацией метамоделей, интерактивностью и высокой нагрузкой. Эта машина создана на базе методик метапрограммирования и унифицированных компонентов с динамической интерпретацией метамоделей, брокеров запросов и трансляции событий, асинхронного программирования и неблокирующего ввода/вывода. Предлагаемый подход позволяет гибко адаптировать ИС между собой, строить высоконагруженные API с состоянием и динамически модифицировать процессы обработки и программные компоненты без остановки обслуживания.

Ключевые слова: динамическая интерпретация, брокер обработки, метамодель, метаданные, машина состояний, информационные системы.

Введение

Прикладная виртуальная машина позволяет решать задачи интеграции распределенных источников данных, обработки слабосвязанных и слабоструктурированных данных в межкорпоративных системах, а также задачи, связанные с интенсивным обменом сообщений в режиме времени, приближенном к реальному. Такие системы и задачи получают широкое распространение в связи с развитием и распространением облачных технологий и соответствующих сервисов, предоставляемых клиентам.

Прикладная виртуальная машина состоит из машины состояний, брокера обработки запросов и трансляции событий, а также Web – сервисов, которые обрабатывают состояния информационных объектов в машине состояний, и реализуется в виде специализированного сервера Impress с динамической интерпретацией метамоделей.

Динамическая интерпретация предусматривает два режима обработки данных:

1. С изменением структуры, когда вместе с данными поступают новые метаданные и нужно повторно провести связывание абстрактных программных компонентов, преобразование их в машинный код и разворачивание в оперативной памяти соответствующих структур данных.

2. Без изменения структуры, когда программные компоненты метамодели уже закешированы, подготовлены к работе и динамический компилятор проводит лишь статистическую оптимизацию типов данных и классов сценария.

 ISSN 1560-8956

Первый режим используется, как показывает практика, на порядок реже второго, т.е. только в от 1% до 15% операций с данными (в зависимости от специфики предметной области и динамичности ее модели). В большинстве случаев возможна оптимизация с выполнением первого режима не в момент обработки запроса, а в момент изменения метаданных в системе или по расписанию, то есть, код и данные перестраиваются и кешируются не в момент запросов к сервисам, а сразу при получении каждым потоком обработки события о таких изменениях. Прикладной программный код метамодели может быть так же изменен и динамически подгружен в оперативную память без остановки всего сервера или отдельных процессов. Это достигается благодаря мониторингу файловой системы и обработке изменений файлов, содержащих метамодель и метаданные. При выявлении изменений, создании, удалении, переименовании прикладных программных модулей, а так же, файлов со схемами данных предметной области и конфигурации сервера, новые файлы загружаются в память и заменяют их место в кеше, а старые освобождаются. Однако, процессы, которые во время изменений уже начались, должны завершиться в штатном режиме, поэтому в памяти могут содержаться одновременно несколько версий метамодели и прикладного кода. Старые версии не выгружаются из памяти, а просто теряют ссылки в хеш-таблицах, по которым происходит инициализация новых запросов обработки. После завершения обработки данных, старыми экземплярами классов, программный код и структуры данных, с ним связанные, постепенно вытесняются из оперативной памяти и удаляются сборщиком мусора виртуальной машины.

Базовые компоненты прикладной виртуальной машины

Для реализации двух режимов обработки, приведенных выше, необходимы следующие функциональные модули - интерпретатор метамодели, средства синтаксического и структурного анализа метаданных, машина состояний, сигнальный протокол и транспорт для распространения событий изменения метаданных, кеш программного кода (подготовленного для исполнения), кеш метамоделей, шаблонов и интерфейсов (для построенных динамически программных компонентов). Реализация функциональных модулей представляет собой три основных компонента архитектуры сервера с динамической интрепретацией метамодели:

1. Брокер трансляции событий реализован на базе двух протоколов, механизма IPC (Inter-Process Communication / межпроцессовое взаимодействие), встроенного в операционные системы (для взаимодействия нескольких потоков обработки в рамках одного сервера) и протокола 0MQ (ZeroMQ), обеспечивающего паттерны быстрого обмена сообщениями по TCP и предоставляющего унифициро-

ванный интерфейс IPC+TCP, когда прикладной код не изменяется от выбора транспортного уровня. В частности, ZeroMQ обеспечивает паттерн PUB/SUB (публикация/подписка), который используется для быстрого распространения широковещательных уведомлений (несколько подписчиков к одному публикатору) об изменении метамодели и метаданных в облачной инфраструктуре процессов, запускаемых на технологии Impress (сервера приложений для прикладных программных систем с динамической интерпретацией).

2. Машина состояний используется для хранения сессий и состояния объектов информационной модели предметной области, для чего предлагается использовать непосредственно оперативную память V8. Машина состояний содержит и данные и метаданные, связанные на текущий момент времени с программным кодом. Их обработку осуществляют Web-сервисы, которые обращаются к машине состояний, а результаты передаются соответствующему брокеру обработки и трансляции.

3. Web-сервис обработки данных обеспечивает взаимодействие между распределенными приложениями и между клиентскими и серверными частями прикладных приложений по протоколам HTTP, HTTPS, WebSockets и SSE. Для тонких клиентов (Web-интерфейсов к прикладному ПО), Web-сервис предусматривает обмен данными в формате JSON (формат сериализации объектов JavaScript) по технологии AJAX (API асинхронных запросов при помощи XMLHttpRequest, сокращенно XHR-запросов).

Таким образом, три приведенные компонента покрывают полный спектр задач, необходимых для разработки как back-end, так и front-end прикладных информационных систем, а также их интеграции между собой и со смежными системами с применением сетевого HTTP API на базе Web-сервисов по подписке для обеспечения межсерверного взаимодействия внутри компании или при интеграции с внешними системами (например, партнерских организаций или филиалов). Для повышения гибкости передачи, обработки и хранения данных, применяются методики метапрограммирования: интраспекция, метанотации, динамическая интерпретация метамоделей, скафодлинг, динамическое и позднее связывание кода, специализированные метаязыки с декларативными и императивными синтаксическими структурами и соответствующие форматы сериализации.

Машина состояний

Машина состояний: это часть оперативной памяти процесса обработки, а именно, глобальные кеши, содержащие таблицы вызовов динамически созданных методов и объектов. Такое решение имеет преимущество перед использованием серверов состояний класса “ключ-значение” (например, memcached и redis), т.к. позволяет хранить сложные структуры данных и экономит вре-

мя для обращения из приложений к межкорпоративному серверу состояний, и не требует дополнительных преобразований данных (например, сериализации и десериализации) при сохранении и восстановлении объектов. Вопрос распределенности хранения решается брокером запросов сервера приложений Impress.

На уровне виртуальной машины V8 есть еще один уровень оптимизации, содержащий виртуальные классы (построенные на основе статистического анализа экземпляров объектов) и скрытые от программиста. Они используются для перехода от динамической к статической типизации, на период времени, пока структура известного множества объектов остается стабильной, т.е. в промежутке между выполнением операций динамической интерпретации. Изменения кода, должны быть синхронизированы с изменением данных и метаданных, т.к. код “ожидает” определенных типов и структур данных.

В таких условиях, машина состояний, превращается из простого кеша индексированных указателей и буферов памяти в сложную систему, которая должна поддерживать версионность, параллельное существование нескольких копий одних и тех же методов, объектов и структур данных. При этом, к банальному выделению и освобождению памяти под состояния, к операциям чтения и записи, добавляются операции мультиплексирования доступа по одинаковым идентификаторам для разных экземпляров программного кода, а так же конвертация данных из старых структур в новые в отложенном режиме.

Брокеры обработки запросов и трансляции событий

Брокеры обработки запросов и трансляции событий - это специализированные подсистемы в корпоративных и межкорпоративных информационных системах, которые обеспечиваются метаданными описания и управления операций индексации, хранения, получения и решения (resolving) для адресных пространств, индексов инфраструктуры, индексов данных и метаданных, индексов систем и сервисов [1]. Обращение к брокеру Web-сервиса предусматривает установление непосредственного взаимодействия между модулями в корпоративной информационной системе [1,2]. Запрос брокера может содержать логические идентификаторы объектов, интерфейсов и сервисов, понятные для системы запрашивающего приложения. Ответ брокера строится на фиксированных структурах данных, которые не меняются в процессе обработки и не зависят от продолжительности работы. Он содержит: идентификаторы низшего уровня абстракции (включая доменные имена, номера портов, системные имена служб, а так же индексы вопрос данных и метаданных). Ответ не может содержать IP или MAC адрес устройств серверов, потому что это уже другой, низкий уровень абстракции. Брокеры определяют эталонные, резервные и

основные системы, задающие направление репликации метаданных и приоритетность их объединения в метамодель, и могут образовывать иерархию.

Внутренние брокеры корпоративного уровня участвуют во всех взаимодействиях модулей прикладного программного обеспечения во внешнем и внутреннем периметре корпоративной сети. Таким образом, внутренний брокер является внутренним не только для топологической сети организации (и не ограничен рамками аппаратного обеспечения), а является логически внутренним для прикладной виртуальной сети. Такая сеть может активно использовать физические каналы общего доступа, личные технические и коммуникационные средства сотрудников, включая мобильные телефоны, смартфоны, персональные и домашние ПК, ноутбуки и планшетные устройства и т.п.

Внешний брокер координирует процесс интеграции уже на межкорпоративном уровне, включает в себя индексацию распределенных массивов данных, каталогизацию организаций, вступивших в соглашение об обмене данными, и взаимодействие с внутренними корпоративными брокерами. Брокеры могут образовывать иерархию как внутри, так и вне компании. При этом, внешняя иерархия первична по управлению совместным адресным пространством, а внутренняя - первична по управлению правами на доступ к ресурсам: сервисам, данным, метаданным, индексам и интерфейсам.

Корневой брокер управляет распределением в корне пространства имен и идентификаторов, устанавливает соответствие между диапазонами имен и идентификаторов для подчиненного брокера. Одной из основных функций брокера является кэширование индексов имен и идентификаторов, поступающих от высших брокеров при запросах от низших брокеров. Права же доступа, получаемые от нижестоящих брокеров, могут быть им не нужны, так как могут устареть в процессе взаимодействия.

Таким образом, в информационной системе, которая построена в технологиях с динамической интерпретацией метамоделей, предусмотрены динамические связи между слоями и подсистемами, а брокеры выполняют роль управляющей системы интерактивной связи, что и позволяет модифицировать другие связи, протоколы и структуры данных без остановки системы [3].

Брокер запросов служит для трансляции потока событий между разными информационными системами при подписке на эти события конечных пользователей. Брокер реализован на том же системном программном обеспечении и платформе, что и корпоративный брокер: виртуальная машина V8 и платформа node.js. Брокер имеет свой балансировщик нагрузки, который распределяет поток заявок на подключение по tcp сокетам на несколько нод (потоков обработки, поднятых при помощи одной из технологий:

виртуальных машин, виртуальных серверов VPS, облачных серверов или физических выделенных машин). Web-сервис так же имеет балансировщик нагрузки с возможностью подключать несколько серверов для обработки запросов, а для взаимодействия между ними используется технология межпроцессного взаимодействия ZeroMQ.

Прикладная виртуальная машина

Масштабирование прикладных ИС в гетерогенной структуре ограничено одной виртуальной машиной для выполнения одной логической функции, которая создана как программный модуль и скомпилирована в код для выполнения на определенной платформе. Эту машину действительно можно расширять динамически и удобно администрировать, но при этом требуется своя виртуальная машина на каждую функцию. Таких независимых функциональных модулей в ИС может быть максимум несколько десятков. Каждый функциональный блок имеет сильную связанность кода, и слабую внешнюю связанность с другими блоками. Даже в облаке, такой функциональный блок, имеет определенные границы масштабирования - не шире вычислительных возможностей одной виртуальной машины.

Выход за пределы одной машины невозможен без системы распределенных вычислений, что обеспечивает декомпозицию прикладных задач на сеть виртуальных серверов и их распределенное взаимодействие, централизованное управление и консолидацию результатов вычислений, как результат процесса распределенной обработки (в том числе с применением таких технологий, как Map - Reduce, ZeroMQ, Async I/O). В настоящее время, часто, эти задачи решаются отдельно для каждой ИС, т.е. требуют отдельных системных специализированных программных решений для каждого отдельного случая применения.

Прикладная виртуальная машина с динамической интерпретацией метамоделей и обработкой данных в асинхронном режиме (без блокировки) может снять эти вопросы по прикладным задачам ИС, решив их системно, что составляет целостное архитектурное решение, упрощает интеграцию, удешевляет масштабирование и поддержку прикладных информационных систем [4].

Для достижения максимальной производительности разработки прикладных программных систем с использованием современных облачных технологий надо:

- абстрагировать прикладные решения от системных, унифицировать их использование, сделать его прозрачным для прикладных разработчиков,
- сделать разработку моделей и алгоритмов обработки данных в высоконагруженных и масштабируемых системах прозра-

чной, т.е. такой, чтобы процесс разработки не отличался от не облачных систем.

Стиль кода при этом может отличаться, например, асинхронное программирование требует оформления всех результатов внешних запросов в обратные вызовы (callbacks) с неблокирующим вводом / выводом, но уровень абстракции кода не должен уменьшаться, (кода не должно быть больше, чем при обычном объектно-ориентированном или процедурном программировании), т.е. прикладной код должен быть отделен от системного.

Применение методов метапрограммирования и динамической интерпретации метамоделей предметной области, наоборот, существенно повышает уровень абстракции прикладного кода, и способствует уменьшению синтаксических конструкций, в частности для облачной инфраструктуры. При этом, прикладная виртуальная машина со всеми приведенными технологиями и архитектурными принципами построения кода дает возможность запустить отдельную прикладную задачу (функциональный модуль с высокой связанностью кода) на ограниченном количестве виртуальных машин и обеспечить прозрачность для этой задачи, если не известно - на каких виртуальных машинах задача обрабатывается. Таким образом, задача кластеризуется в облачной инфраструктуре при наличии распределенных ресурсов и сервисов как “частного облака”, так и “облака” внешнего провайдера.

Область применения

Системы с динамической интерпретацией метамоделей на базе методик метапрограммирования могут найти применение при разработке прикладных информационных систем, в том числе для:

- складских и логистических IT-систем с большим количеством вовлеченных организаций и информационных объектов;
- систем электронной коммерции с межкорпоративным обменом данными в режиме времени, приближенном к реальному;
- автоматизированных систем обработки слабосвязанных данных и коммуникационных систем обмена данными в гетерогенной среде всемирной сети;
- высоконагруженных систем социальных коммуникаций и IT-систем для исследования общественного мнения.

Выводы

Использование прикладной виртуальной машины в качестве средства для динамической обработки запросов приложений и трансляции событий в среде распределенных источников данных и сообщений дает возможность упростить интеграцию информационных систем для гетерогенной архитектуры IT-систем обработки.

За счет размещения базовых компонентов прикладной виртуальной машины и выполнения обработки в оперативной памяти сервера с динамической интерпретацией метамоделей, удается сократить число слоев виртуализации до одного, что позволяет реализовать высоконагруженные прикладные серверы различной функциональности и значительно повысить их производительность (по нескольким различным тестам производительности, которые проводились авторами, зафиксировано ее повышение на 2-4 порядка).

Библиографический список

1. Шемсединов Т.Г. Слой ИС с динамической интерпретацией метаданных. http://blog.meta-systems.com.ua/2011/01/blog-post_28.html
2. Л.Е.Карпов, В.Н.Юдин. Адаптивное управление по прецедентам основанное на классификации состояний управляемых объектов.- Электронный ресурс: http://cyberleninka.ru/article/n/adaptivnoe-upravlenie-po-pretsedentam
3. Стенін О.А., Тимошин Ю.А., Шемсединов Т.Г., інші. Розробка архітектури та технології обробки корпоративних розподілених джерел даних у середовищі Cloud Computing на основі метамоделей з їх динамічною інтерпретацією. – Звіт про н/т. роботу, НТУУ “КПІ”, № держреєстрації 0112U001455, 2012р., 215 стор.
4. Стенин А.А., Тимошин Ю.А., Шемсединов Т.Г. Метод динамической интерпретации метамоделей в разработке прикладных информационных систем. – Материалы международной научно- практической конференции “Академическая наука – проблемы и достижения”.-М.-2012. – сс. 186–192.

Отримано 04.10.2013 р.