УДК 681.3:621.3(62-52)

Ю.А. Тимошин, Т.Г. Шемсединов, В.П. Ярченко, А.И. Мороз

ТЕХНОЛОГИЯ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ДАННЫХ И ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКИ ИНТЕРПРЕТИРУЕМЫХ МЕТАМОДЕЛЕЙ

Аннотация: Статья посвящена разработке новой технологии, архитектуры и соответствующих программных средств обработки данных распределенных информационных систем, компоненты приложений и данные которых расположены в сетях различных организаций или компаний, образующих гетерогенную среду. Для этого предлагаются специальные новые компоненты- брокеры обработки и управления, прикладная виртуальная машина, включающая специализированный сервер Impress и СУБД MongoDB, которые используют программные средства динамического связывания компонентов и динамического построения моделей на базе интроспекции компонентов и методиках метапрограммирования.

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

Постановка задачи

Появление сервисных технологий с последующим включением их в качестве основы для сред с распределенной обработкой заявок удаленных пользователей, в том числе типа Cloud Computing, и породило множество проблем, связанных с организацией управления доступом к распределенной информации, управления процессами обработки и передачи данных по запросам этих пользователей, многоплатформенностью систем хранения и обработки данных, разнообразием формирование необходимых сервисов и отчетов, защитой персональных данных пользователей корпоративного уровня.

Переход от монолитных ИТ- систем к слабо связанной сервисной архитектуры требует пересмотра основ таких систем, в частности, пересмотра отношения к данным и процессов бизнес-логики, обрабатываемых в корпоративной среде, которое может включать такие базовые компоненты Cloud Computing, как фермы виртуальных серверов для загрузки, поиска и обработки данных. Такие системы должны учитывать состояние и характеристики представленной информации и ее назначение. Таким образом возникает потребность в интеллектуальной обработке данных на основе метаданных в среде распределенных сервисов, реализуемых в Cloud Computing.

Задача интеграции слабо структурированных данных еще недостаточно описана и не разработана с научной точки зрения в

 ISSN 1560-8956

среде Cloud Computing, а объем неструктурированных данных в мире растет со скоростью 200% в год. Форматы и структура данных, поступающих из различных источников, сильно различаются, и их необходимо согласовывать. Были проанализированы новые поколения IT-систем корпоративного уровня, действующих в режиме реального времени- RTE для реализации аналитики и мониторинга бизнес-активности, а также новые, такие как технологическое направление Application Oriented Network (AON), направление работы с данными- управление контентом предприятия ECM и интеграционные платформы EAI, которые оснащаются интеграционными адаптерами IBM, SAP, Oracle или Microsoft. Но они не поддерживают корпоративный поиск и еще более сложные функции, связанные с мультимедиа данными, и работают только со структурированными данными.

Свою специфику накладывают и технологии Cloud Computing, которые определяют разного рода информационные ресурсы, разделяемые в соответствии с запросом, форма которого зависит от особенностей того, как и каким Web-клиентом он выполняется, независимо от того, где расположены серверы обработки, ресурсы которых обязательно виртуализируются, что обеспечивает динамическое масштабирование и избавляет пользователей от привязки к конкретным физическим серверам и системам хранения, аппаратные и программные ресурсы которых реализуются в форме типовых облачных сервисов.

Для облачных приложений в среде Cloud Computing необходимо решать такие задачи, как управление состоянием модели предметной области и метамодели на сервере, управление сессиями пользователей, синхронизации и блокировки параллельных процессов и доступа к ресурсам, логирования данных и наладка среды, создание набора средств визуализации и построения пользовательских интерфейсов, кэширование данных и оптимизация выполнения бизнес-логики, отработка средств доступа к данным в различных СУБД - реляционных (например sql- совместимых), иерархических (например файловые системы), типа “ключ-значение” (например memcached) и т. п., создание средств сетевой коммуникации, межсерверного взаимодействия (сервис- сервис), а также создание средств для администрирования распределенных источников данных и систем распределенной обработки.

Для взаимодействия приложений обычно используются такие методы, как обмен файлами, общая база данных, удаленный вызов и асинхронный обмен сообщениями [1]. В этом списке нет прямого обмена данными между базами данных приложений: этот метод не ближе к интеграции приложений, а к перемещению данных. С точки зрения взаимодействия приложений важна возможность в процессе обмена данными выполнять какую-то содержательную обработку (например, при загрузке накладных перечис-

лять товарные остатки). Прямой обмен данными, который обычно выполняется средствами класса ETL (extract, transfer, load) или самодельными утилитами, обычно такой возможности не предоставляет.

Структура информационных объектов

Информационные объекты, требующие обработки и трансляции в гетерогенной среде распределенных источников данных, включают следующие классы, виды и типы, которые создают все множество объектов обработки для технологий Cloud Computing [2,3].

1. Классы:
– Объекты обработки
– Процессы обработки и трансляции
2. Классы создают виды:
а) Запросы;
б) События;
в) Сервисы.
3. Виды делятся на типы, которые создают группы:
а) Запросы к Web- сервисам
б) Отложенные запросы
в) Сообщения
г) Трансляции и маршрутизации событий и запросов
д) Прослушивания портов и восстановления соединений
е) Управления очередями на обработку и трансляцию.

Согласно изложенной выше классификации была разработана общая структура указанных информационных объектов. На рис. 1 приведена следующая структура информационных объектов для обработки брокера запросов и трансляции событий в системе.

Интеграция гетерогенных систем внутри предприятия

ИТ-инфраструктура предприятия всегда состоит из многих компонентов, включающих системное и прикладное программное обеспечение, которое создано в разное время, на разных платформах и с применением различных архитектурных подходов. Это приводит к тому, что задача интеграции этих составляющих частей оказывается гораздо сложнее, чем задача разработки их функционала.

Существует несколько подходов к интеграции, которые разделяются по уровням реализации на три следующие: интеграция на уровне данных (когда в системе выделяется ядро базы данных и обеспечивается доступ к ней из всех модулей; интеграция на уровне вызовов (когда различные модули обращаются к серверу приложений) и ручная интеграция (когда нагрузка приходится

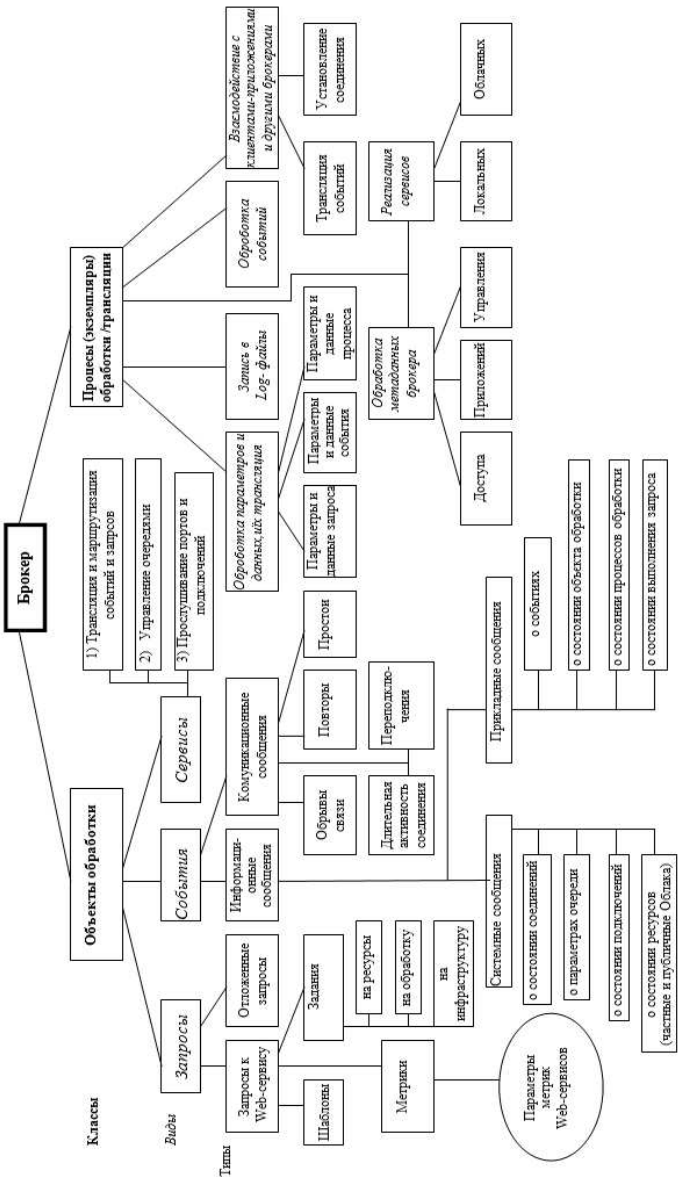

Рисунок 1. - Структура информационных объектов и процессов обработки брокера запросов и трансляции событий

на пользователя) [4,5]. Кроме этого, возможна интеграция с помощью динамической интерпретации метаданных [5,6]. Это не отдельный уровень, потому физическое взаимодействие при этом может быть реализовано на уровне данных или вызовов, это отдельный мета-уровень, когда решение об обращении по данным принимаются на основе интерпретации этих самых данных. При этом в системе не должно быть центральной базы данных или центральной сервисной шины, к которым обращаются все компоненты [5,7], зато нужен центральный брокер, что обеспечивает распространение по компонентам системы одной метамодели. Такая архитектура предполагает, также, фиксацию протоколов взаимодействия и форматов передачи данных, но функциональный набор сетевых вызовов не должен быть фиксированным.

Для описания прикладных объектов применяют сериализацию в распространенных декларативных форматах, таких, как JSON или XML. При этом нужно вводить дополнительные ограничения, чтобы четко фиксировать структуры данных в тех случаях, когда декларативные форматы имеют несколько вариантов описания одинаковых объектов.

Для динамической обработки распределенных данных и приложений была разработана прикладная виртуальная машина,

включающая специализированный сервер Impress и СУБД MongoDB, которые используют программные средства динамического связывания компонентов и динамического построения моделей на базе интроспекции компонентов и методиках метапрограммирования.

Для реализации приведенной архитектуры были разработаны и протестированы следующие программные компоненты:

Брокер корпоративной прикладной информационной системы- модуль, обеспечивающий реализацию протокола передачи метаданных на базе протоколов HTTP и HTTPS с поддержкой сессий соединения и реализацией машины состояния непосредственно в оперативной памяти сервера на базе бездисковых СУБД типа ключ-значение.

Библиотека для создания сетевых интерфейсов с динамической структурой- инструментарий, позволяющий быстро реализовать прикладные сетевые интерфейсы API как наборы функций, и централизованно обеспечивает многопоточную обработку запросов, передачу параметров, экранирование от неразрешенных типов данных, безопасность связи, блокирующие и асинхронные вызовы, т.д.

Средства сериализации прикладных объектов на уровне вызовов- встроенные в платформу средства на клиентской и серверной сторонах, обеспечивающих трансляцию и восстановления прикладных объектов (в том числе при взаимодействии между различными компонентами и подсистемами).

Средства обработки объектов на уровне СУБД- включают системную часть базы данных для хранения метаданных о структуре и отношение модулей системы, прав пользователей и сетевых прикладных API, средства выбора и обработки данных из базы и построения динамических запросов.

Средства интерпретации метамоделей- функционал, позволяющий формировать и модифицировать метамодели в оперативной памяти, интерпретировать параметры и превращать модели в сетевые вызовы, экранные формы, запросы в СУБД и т.д.

В течение жизненного цикла информационных систем, неоднократно возникает потребность в переформировании связей между программными компонентами, включении новых модулей или изменении бизнес-логики. Поэтому, одной из важных задач брокера, является обновление модели в компонентах системы. В архитектуре с динамической интерпретацией, прикладные модули получают модель вместе с данными, но кэшируют ее в оперативной памяти для уменьшения объема передаваемых данных. Обновление кэша происходит при каждом обращении после сравнения хэш - ключа (контрольной суммы модели). Таким образом, система может без перекомпиляции, замены программных модулей, и даже без перезагрузки, обновлять интерфейс, функциональность, структуру и параметры объектов, ограничения в полях форм и прочее. Но некоторые операции в системе могут быть растянуты во времени, например, вызов процедуры мог происходить при наличии одной модели, а результаты вернулись уже после ее изменения, поэтому, все компоненты по пути вызовов и передачи данных должны хранить не только последнюю модель, но и ту, которая была актуальна при инициировании предыдущей операции. Это аналог транзакционной целостности для операций динамической интерпретации метамоделей.

Другая особенность брокеров заключается в наличии “состояния” в каждом звене системы и его синхронизации через брокеры и специфической обработке в системах с динамической интерпретацией метамоделей. Состояние определяется данными, хранящимися в модуле системы после выполнения вызова, и влияет на последующие вызовы. Состояние отличается от “сессионных переменных” тем, что он хранится в оперативной памяти модуля долгое время и с ним не происходит операций хранения / восстановления при каждом вызове, как это предусмотрено архитектурой REST (Stateless или Representational State Transfer). Наличие положения предполагает возможность перехода системы в различные режимы работы, но состояние распространяется не на весь модуль, а на “контексты”, определяющими эти обособленные рабочие зоны в оперативной памяти приложений, за которыми закреплен определенный пользователь, и на один управляющий канал с которого система получает команды и куда отсылает результаты. Та-

ким каналом может быть интерфейс, подключенный к сервисной шине удаленный клиентский терминал или приложение, другая информационная система, транслирующая запросы пользователя через несколько слоев корпоративной инфраструктуры. Порождение контекстов происходит иерархически, начиная от пользователя, и заканчивая слоем данных, а разветвление может происходить на каждом слое, порождая дочерние контексты, уничтожаются при закрытии родительской. Задача брокера, синхронизировать данные контекстов, выделять и освобождать оперативную память для хранения "состояния" и связывать контексты с соответствующими версиями модели при интерпретации метаданных.

Для достижения максимальной производительности разработки прикладных программных систем с применением современных облачных технологий, нужно абстрагировать прикладные решения от системных, и унифицировать их использование, сделать его прозрачным для прикладных разработчиков, а разработку моделей и алгоритмов обработки данных в высоконагруженных и масштабируемых системах такими, чтобы процесс разработки не отличался от обычной разработки. Стиль кода 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 отделен от системного.

Применение методов метапрограммирования и динамической интерпретации метамоделей предметной области, наоборот, существенно повышает уровень абстракции прикладного кода, и способствует уменьшению синтаксических конструкций. Таким образом, 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 облачных виртуальных машин и совсем прозрачно для этой задачи, если не известно на каких виртуальных машинах реально задача обрабатывается. Таким образом, задача кластеризуется в облачной инфраструктуре при наличии распределенных ресурсов и сервисов "частного облака" и "облака" внешнего провайдера.

Разработка инфраструктуры межоблачного взаимодействия

При взаимодействии двух и более систем через сетевые прикладные интерфейсы на основе интерпретации метамодели позволят взаимодействовать прикладным информационным системам, которые связаны между собой. Метамодели передаваемых со сто-

роны сети, не знают заранее структуру и параметры информационных объектов, и не привязаны жестко к именам функций и наборов параметров, при осуществлении межсистемных вызовов. Вместо этого, стороны знают язык мета- описания, что позволяет динамически интерпретировать данные и осуществлять вызовы, формируя параметры и интерпретируя ответы удаленной стороны.

Инфраструктура межоблачного взаимодействия условно разделена на следующие зоны:

1. Внешние сервисы – это системы, поддерживающие функционирование глобальной сети, средств связи общего пользования и такие внешние сервисы, как облачные хранилища данных CDN (сети доставки данных), система разрешения доменных имен DNS и т.п.

2. Корпоративная информационная система – это система, которую составляют Облачный Датацентр, что создает ядро прикладной сети компании, а также защищен периметр внутри организации. Датацентр содержит сервера и оборудования для обеспечения распределения нагрузки, хранения и выполнения бизнес-приложений, которые создают разнофункциональные кластеры. Периметр составляют сервера, которые определяют ресурсы для реализации облачных сервисов IaaS, PaaS, SaaS или DaaS и организуют очереди на выполнение этих сервисов и их управления.

3. Облачный контроллер прикладной виртуальной машины, серверы балансировки нагрузки и серверы службы интеграции, находящихся в Датацентре и под управлением системного интегратора или одной из компаний, входящих в групповые договоренности по интеграции информационных систем (как следствие плотной интеграции бизнес - процессов).

Технология обработки брокерами запросов и трансляции событий в среде Cloud Computing потребовали разработки новой топологии взаимодействия базовых функциональных компонентов для структурирования корпоративного Датацентра, веб- сервисов, СУБД, машины состояний, компонентов балансировки нагрузки и для обработки отложенных заданий, а также специализированных серверов - сервера распределения задач, сервера распределения данных и сервера для динамической интерпретации метаданных, что и было успешно сделано авторами.

Архитектура системы с динамической интерпретацией метамоделей

Для реализации системы с динамической интерпретацией метамоделей используется следующий стек технологий:

Операционная система CentOS 6.4 (64bit)

Виртуальная машина V8 с node.js

СУБД (сервера баз данных): Postgress, Oracle, MongoDB (собственной разработки)

Веб сервер: node.js (HTTP, HTTPS)

Сервер приложений: Impress (собственной разработки).

Предлагаемая архитектура системы включает:

1. Кластер обработки, который разворачивается на базе node.js и Impress с двумя каскадами балансировки: первый уровень обеспечивается аппаратным решением и маршрутизирует запросы с одной точки входа по серверам обработки, а второй уровень производит балансировку в стеке процессов node.js, привязанных к логическим ядрам CPU в рамках одного сервера. Связывание соединения с потоком обработки происходит через служебный cookie с именем "node в формате "C"<ClusterID>"N"<NodeProcessID>. Если cookie не найден, то обработка запроса передается в случайный процесс по алгоритму RoundRobin процесс маркирует соединение своим идентификатором и следующие запросы попадают в этот же процесс.

2. В качестве машины состояний для хранения сессий и состояния объектов информационной модели предметной области, предлагается использовать непосредственно оперативную память V8.

3. Для СУБД используется несколько машин, объединенных в кластер хранения. В зависимости от особенностей данных может быть выбрано или реляционное хранение или безсхемное или же, на логическом уровне, может быть сконструировано гибридное решение. Часть данных (например, изображения) могут храниться в файловой системе. Для гибридного решения могут быть разработаны соответствующие драйвера, которые обеспечат целостность данных, буферизацию и восстановление сессий при временных перебоях связи, транзакции с вовлечением нескольких СУБД и другие специфические и вспомогательные задачи, как ORM (object-relational mapping) и scaf olding (построение программных и визуальных компонентов динамически на основе структур данных).

4. Клиентская часть Web-сервера node.js реализуется как HTML5 Application (с возможностью работать в оффлайне и онлайне через App Cache и хранящее локальные данные в СУБД WebSQL или IndexedDB). Такое решение будет универсальным, унифицированным и кроссплатформенным, как для браузеров, так и для мобильных устройств, но не позволяет использовать все датчики и подсистемы на мобильных платформах. Если это необходимо, то можно создать дополнительно нативные мобильные приложения. Например, для Android приложение Java, и т.д. для платформ iOS (iPad, iPod и iPhone), Windows Phone и разных версий Android (для 2.x и 4.x может потребоваться разный код).

5.Сервер приложений Impress поддерживает двухуровневую каскадную маршрутизацию потоков событий в кластере брокеров на node.js и широковещательные события с группировкой клиентов по именованным каналам событий (по принципу подписки на каналы).

Протоколы сетевого взаимодействия клиента и сервера могут быть реализованы на базе HTTP и HTTPS, для Web-сервисов (сетевое API), или как асинхронные запросы AJAX с обменом данными в формате сериализации JSON. Для трансляции событий применяем архитектуру брокеров и протокол SSE (Server-sent events), который поддерживает однонаправленные потоки событий с кодированием структур данных в том же JSON.

Сетевые интерфейсы платформы Impress строятся при помощи отображения структуры файловой системы (каталогов со специфицированными именами) в URL, вызываемые со стороны клиента.

Выводы

Разработанная технология и архитектура позволяет решать широкий круг прикладных задач и проводить комплексную интеграцию приложений в информационных системах уровня предприятия и межкорпоративного уровня для ИТ-систем с динамической интерпретацией метамоделей.

Использование функциональных брокеров, прикладной виртуальной машины и других компонентов 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.

Список использованных источников

1. Эпштейн А. Асинхронная передача сообщений с помощью MSMQ “Windows IT Pro”, №08, – 2002, – [Электронный ресурс].– Режим доступа: http://www.osp.ru/win2000/2002/08/175133/
2. Шемсединов Т.Г. Слой ИС с динамической интерпретацией метаданных. [Электронный ресурс].– Режим доступа: http://blog.meta-systems.com.ua/2011/01/blog-post_28.html
3. Стенін О.А. Розробка фізичних і логічних метрик в задачі багатокритеріальної оптимізації інформаційного навантаження при структуризації корпоративного центру даних. // Стенін О.А., Тимошин Ю.А., Шемсединов Т.Г., Шуст С.О. Адаптивні системи автоматичного управління - Дніпропетровськ: ДНВП Системні технології, 2009-Вип.12(32). – С. 86–91
4. Боркус В. Методы и инструменты интеграции корпоративных приложений: Отчет/ RC Group.– М.: RC Group, 2006. – 13 с.
5. Аткин А., Интеграция ИТ: основные понятия и технологии. 2009г.[Электронный ресурс]. – http://www.citcity.ru
6. Taylor John. Thoughts from the Integration Consortium: Enterprise Information Integration: A New Definition

(Вести из Консорциума по интеграции. Интеграция корпоративной информации – новое определение). [Электронный ресурс].– Режим доступа: http://www.dmreview.com/article_sub.cfm?articleId=1009669.

7. Зауфер Гюнтер. Шаблоны для информационного сервиса (03.08.2007). //Гюнтер Зауфер, Мэй Сельваж, Эойн Лейн, Билл Мэтьюс.[Электронный ресурс].– Режим доступа: http://www.osp.ru/win2000/2002/08/175133/

Отримано 28.03.2014 р.