УДК 681.3.04

ПРОЕКТИРОВАНИЕ ИНСТРУМЕНТАРИЯ СЕТЕВОГО ПРОГРАММНОГО ПРОДУКТА. СОВРЕМЕННОЕ СОСТОЯНИЕ И ПЕРСПЕКТИВЫ РАЗВИТИЯ

Н Н. Глибовец, С.С. Гороховский

Национальный университет «Киево-могилянская академия»,
Украина, 04655, Киев, ул. Г. Сковороды, 2

Основной вопрос этой статьи – организация эффективного распределенного управления в современных распределенных системах с разнородными ресурсами. Описанная модель единого корпоративного управления WBEM способна содержать клиента, на базе браузера для мониторинга, контроля, конфигурации и диагностирования компонентов бизнес-процессов и транзакций.

The main topic of this article - the organization of effective distributed control in modern distributed systems with heterogeneous resources. Model a single corporate management WBEM is described. It can provide the client, browser-based monitoring, control, configuring, and diagnostics components of business processes and transactions.

Введение

Современный уровень развития распределенных систем, разнообразие программного обеспечения, разноплановость задач, которые решаются с помощью распределенных систем, отсутствие единой модели таких вычислений свидетельствуют о глубоких фундаментальных проблемах в управлении распределенными сетями и системами, решение которых остается актуальным и по сей день.

Напомним, что классическая полная программная модель распределенной системы состоит из двух частей: вычислительной модели и координационной модели. Вычислительная модель характеризуется единственной вычислительной активностью, т. е. вычисления проходят только с единственной ветвью дискретных вычислений. Координационная модель – средство соединения отдельных активностей в ансамбль, а координационные языки обеспечивают оперативное создание вычислительных активностей и поддержку взаимодействия между ними. Различают координационные модели и средства. Они могут интегрироваться в языки распределенного программирования, или оставаться независимыми от языка. Одной из первых, действующих и полных распределенных координационных систем была Linda [1], но концептуальная элегантность вызывала определенные трудности в реализации [2]. Linda до сих пор остается одной из лучших координационных моделей для распределенных вычислений, о чем свидетельствует как большое количество реализаций для разных языков и систем программирования, так и тот факт, что она остается отправной точкой для многих координационных систем следующих поколений. Под руководством авторов была реализована полная Linda для поддержки параллелизма и распределенности в языке Scala [Scala], который был объявлен как масштабируемый язык компонентного программирования, однако не содержащий явных механизмов параллельного и распределенного программирования. Тонкостям этой реализации будет посвящена отдельная статья (Scala – язык со статической типизацией).

Понятно, что невозможно создать универсальные инструментальные средства построения распределенных вычислений – слишком широкий спектр применений, но можно сформулировать общие требования к таким средствам. Они должны обеспечить возможность разработки масштабных, продуктивных, гибких, безопасных, координированных систем, которые разделяют динамические, большие по объему информационные и вычислительные ресурсы. Координированное владение общими ресурсами и решение сложных задач в динамических, короткоживущих средах хорошо выражается метафорой “виртуальной организации” [3].

Особое внимание в последнее время уделяется средствам распределенного программирования в Grid-системах. Здесь возникают тонкие проблемы, связанные с моделями программирования, динамической гетерогенной конфигурируемостью, надежностью вычислений, интероперабельностью, переносимостью, безопасностью, и адаптируемостью. Этот список проблем может быть продолжен. Выполняется масса проектов, цель которых – решить некоторые из проблем. Однако полного и удовлетворительного решения всех проблем достичь практически невозможно. Хотя в последнее время считается, что большинство из них может быть решено с помощью мобильных интеллектуальных агентов.

Классификация требований

Современный информационный мир можно рассматривать как сверхсложную сеть, которая состоит из подсетей, любую из которых можно рассматривать как некоторую распределенную сеть или подсистему. Оптимистические надежды 90-х годов на беспроблемную работу в Интернет-обществе постепенно забываются, и человечество начинает понимать сверхсложность проблемы управления и защиты информации в таких

системах. Примеры и результаты атак на порталы мирового значения свидетельствуют о фундаментальных трудностях организации эффективного управления распределенными сетями и системами. Исследования многих авторов [4, 5] свидетельствуют о том, что современная технология поддержки процесса проектирования инструментальных средств распределенного программирования (ИСРП) должна быть интеллектуальной. В свою очередь интеллектуальный инструментарий должен быть многоуровневый и иметь хотя бы три уровня: организационный, технологический и базовый.

Кроме удобства проектирования традиционных компонент (управление, поддержка интерактивного режима, идентификации и защиты), ядро ИСРП должно иметь обязательные компоненты обеспечения удобного интерфейса и визуализации процессов, тестирования и проектирования, автоматической архивации, сохранения и перерегистрации версий проектов, быстрого прототипирования, развитый аппарат построения и поддержки баз знаний в соответствии с предметными областями рассмотрения. Важным моментом здесь является возможность индивидуализации и специализации проблемно-ориентированной среды вычислений в применении к выбранной предметной области, а также в соответствии с квалификацией и личными вкусами проектировщика. Особенности указанных компонентов определяют соответствующую классификацию ИСРП.

Понятно также, что качественная система ИСРП должна естественным образом включать в вычислительную модель разнотипные технологические средства решения задач, которые базируются на адекватных математических моделях описания предметных областей [4]. Сюда можно отнести разные логики, компьютерную алгебру, средства порождения и преобразования динамических структур данных, которые отображают эволюционный характер развития объектов программирования, средства логического вывода, моделирование, пакеты прикладных программ решения вычислительных и оптимизационных задач.

Не вызовет возражений и тот факт, что нижний, базовый уровень компонентов ИСРП должен поддерживать простую реализацию систем управления базами данных или знаний, готовых базовых блоков проектирования более сложных систем, иметь развитые средства машинной графики, обязательно иметь собственный языковой процессор, и удобные системы документооборота и сопровождения. Он должен иметь также средства саморазвития и отладки инструментария.

Можно также рассматривать уровни инструментария относительно уровней инфраструктуры информационных систем [6]. Очевидно, что проблемно-ориентированная среда должна быть открытой для включения новых программных и технических средств и готового прикладного программного продукта без перегенерации и перетрансляции.

Многообразие компонент требует специальных средств интеграции и взаимодействия, детальной проработки интерфейсов и протоколов. Понятно, что это возможно на базе объектно-ориентированного подхода, который кроме известных преимуществ, обеспечивает довольно высокую ступень повторного использования программного кода, а также активного развития корпоративных средств управления на базе Web. Примером одного из таких факторов может быть фактор объема “зерна” вычислений (fine grain/coarse grain), который обеспечит применение принципа макроконвейера в системе обработки данных. Преимущество его применения состоит в том, что каждый отдельный процессор (подсистема), который принимает участие в решении определенной задачи, обеспечивается такими объемами информации и вычислений, которые позволяют процессору достаточно долго обрабатывать эту информацию без прерываний на обмены с другими процессорами и внешней памятью [7].

Наличие развитых структур управления в современных языках параллельного программирования (Ада [8], Маяк [9], Парус [10]), возможность планирования логической структуры вычислительных ресурсов в соответствии со структурами данных задачи; организация и синхронизация обменов между параллельными ветвями задачи; обмен сообщениями в разных режимах (без прерывания по типу почтового ящика или с прерываниями) требуют динамической поддержки системой ИСРП.

Наконец, современные алгоритмы распределенных вычислений предопределяют специфические требования к практической реализации. Например, эффективность реализации повышается, если архитектура технических средств содержит: специализированные процессоры, среди которых можно выделить арифметические (с довольно значительной локальной памятью, большим набором вычислительных операций, в том числе векторных) и управляющие; средства поддержки развитых структур данных таких, как стеки, буфера, очереди; средства микропрограммной поддержки интерпретации языка очень высокого уровня; быструю реализацию широкого спектра прерываний; коммутационная сеть должна обеспечивать произвольные попарные соединения процессоров. Понятно также, что система ИСРП должна обязательно обеспечивать надежность обработки, своевременную реакцию на аварийные ситуации (продолжение вычислений при отказах отдельных процессоров или каналов связи), защиту информационной среды вычислений, удобный диалог пользователей с системой, мощные средства отладки и т. п.

Указанные требования являются “внешними” целями при проектировании систем ИСРП, и их успешное достижение существенно зависит от того, как достигаются “внутренние” цели проектирования, такие как простота интерфейсов между компонентами системы ИСРП, концептуальная целостность, максимальное использование реального параллелизма. Укажем попутно, что современная литература, посвященная проблемам распределенных вычислений, почти не использует опыт, накопленный в параллельном программировании и разработке многопроцессорных систем.

Опыт использования в многопроцессорных системах принципа распределения вычислений показывает, что для произвольной “чистой” стратегии найдется класс задач, для которого она становится неэффективной.

Поэтому для идеальной системы ИСРП (СИСРП) видится необходимым выделение двух типов компонентов. Первая группа должна настраиваться на каждую конкретную задачу, другая группа компонент вообще не должна зависеть от конкретных задач. Такого распределения при создании ИСРП можно достичь с помощью децентрализованного управления.

Итак, при проектировании ИСРП нужно выделять разные уровни управления вычислениями [11] и каждому уровню ставить в соответствие типовой модуль СИСРП. В этом процессе желательно придерживаться следующих принципов:

- поддерживать максимальную автономию каждого модуля после его активизации модулем более высокого уровня;
- выделять “универсальные”, независимые от задач, модули СИСРП;
- выделять модули СИСРП, которые настраиваются на решение конкретных задач.

Такая классификация модулей СИСРП и распределение функций сверху вниз на организационные, управляющие и функции обслуживания позволяют иметь четкое описание связей модулей СИСРП по данным и подчиненности. Одним из примеров реализации таких модулей определенного уровня иерархии является использование механизма сопрограмм. Симметричность отношения вызова можно реализовать с помощью единой структуры модулей через подпрограммы обработки внешних прерываний. В этом случае вызов интерпретируется передачей сообщения с прерыванием с фактическими параметрами. После активизации модуль СИСРП должен выполнить инициализацию своих внутренних структур данных и затем находиться в состоянии ожидания приема сообщений извне. Получив сообщение с прерыванием, другими словами, приказ “свыше”, модуль выполняет действия, связанные с получением сообщения, и возвращает управление в сопрограмму, которая его вызвала. После этого он может или продолжить работу, или перейти в режим ожидания. На время обработки сообщения с прерыванием процесс должен “закрываться” от внешних прерываний. Итак, каждый модуль СИСРП может иметь приблизительно такую структуру:

- подпрограмма внешнего прерывания, которая выполняет обработку сообщения от верхних уровней и от компонент своего уровня;
- подпрограмма внутреннего прерывания, которая выполняет обработку прерываний в середине процесса или компонента;
- инициализация внутренних структур и (возможно) настройка на ситуацию;
- оператор ожидания.

Отметим, что подход к инструментальным средствам распределенных вычислений значительно зависит от сферы использования конечного продукта. От этого очень зависит уровень выразительности языковых средств, обеспечение сетевой прозрачности модулей применений, стойкость применений в аварийных ситуациях и т.п.. Вдобавок, важное значение имеет среда применения, дисциплина координации, которая может варьироваться от модели клиент-сервер до коллектива равных партнеров (peer-to-peer). Есть уже достаточно примеров реализации сред распределенных вычислений на базе метафоры рынка (electronic marketplace) [12].

Очень интересные решения инструментария распределенных вычислений можно найти в современных многоагентных системах [13]. Агенты существуют и действуют в средах, которые распределены пространственно (места существования), семантически (разные онтологии и языки), во времени и функционально. Для программирования поведения мобильных агентов, которые могут двигаться от хоста к хосту, селиться в новых средах и выполнять свои функции, применяются весьма разнообразные инструментальные средства.

Модель единого корпоративного управления WBEM

Вернемся к основному вопросу этой статьи – организации эффективного распределенного управления в современных распределенных системах с разнородными ресурсами.

Понятно, что типичным представителем таких систем является Internet/intranet. Несколько лет тому в Internet существовало четкое разделение труда между подсистемами жизнеобеспечения сети. Так, за отображение информации на веб-странице «отвечал» язык HTML, для наполнения страниц какой-то “интеллектуальностью” использовались специализированные приложения, например, Java-апплеты. Различалась работа на сервере и клиентском месте. Сценарий обработки некоторого события на сервере описывался на каком-нибудь скриптовом языке, например Perl, а на клиентском месте – JavaScript. Появилось много языков и инструментальных средств с одинаковой или подобной функциональностью реализации этих действий.

Весь процесс обработки информации в таких системах можно подать в виде такой тройки “получение (обработка) – передача – отображение (обработка)”. Например, чтобы некоторая информация появилась в окне браузера, нужно первичные данные, обработанные на серверах приложений Веб-серверов, выбрать из источников, передать на локальную машину клиента и на ней уже выполнить финальную обработку. Появилась и задача оптимизации такой обработки.

Для многих специалистов было понятно, что возможность поддержки общепризнанного тезиса об отделении содержания от формы (содержание информации, которое формируется на сервере, и передается клиенту, должно быть независимым от формы ее отображения) с помощью HTML реализовать естественно невозможно. Они также ощущали, что желательно было бы разработать такое унифицированное средство представления данных, которое обеспечивало возможность их обработки на разных стандартных Web-серверах

и позволяло передавать их стандартными протоколами и при этом сохранять структурированность. Иначе говоря, при приведении данных к унифицированному виду их общая структура и взаимосвязи между элементами должны быть сохранены. Должна быть возможность описания произвольных новых данных без изменения способа описания. В конце концов, этот унифицированный способ описания данных должен ориентироваться на использование в Интернете уже признанных протоколов обмена.

В середине 90-х годов появилась идея создания единой модели корпоративного управления WBEM (Web-based Enterprise Management Interface). Идея состояла в использовании для управления системами и сетями стандартных технологий взаимодействия и защиты Web. Можно выделить три ее основных компонента: общая информационная модель CIM (Common Information Model), в виде объектно-ориентированных схем для управляющей информации; универсальный транспортный протокол передачи информации HTTP; расширенный язык разметки XML (Extensible Markup Language). Последний компонент является простым и в то же время мощным дополнением к протоколу HTTP создание семантических дополнений, которые передаются между приложениями, из браузера в приложения и из браузера в объект управления.

CIM – это абстрактная схема данных. Схемы CIM могут задаваться в виде текстовых файлов, структурированных в соответствии с форматом MOF (Managed Object Format), и графически в файлах Visio или подобной графической программы. Эту схему иногда еще называют словарем данных, которые используются для управления системами и сетями. Словарь имеет средства выделения объектов, атрибутов, отношений и действий и средства документирования, как эти свойства взаимно относятся между собой.

Для реализации приложений управления с использованием данных CIM традиционно используется менеджер объектов (CIM Object Manager, CIMOM). CIMOM можно рассматривать или как процесс диспетчеризации, или как вспомогательный процесс, который выполняет функции посредника между приложениями управления или консолями и местом сохранения данных и индивидуальными источниками данных.

В последнее время появились конкретные программные реализации CIMOM для разных операционных систем и платформ: WMI (Windows Management Instrumentation) для 32-разрядных сред Windows; SWBEMS (Solaris WBEM Services) для среды Solaris платформ SPARC і Intel. Например, программное обеспечение реализованного SWBEMS включает в себя: CIMOM; компилятор MOF, способный проводить грамматический анализ операторов ASCII MOF и прибавлять объектные классы в место сохранения CIM; Solaris Schema, что состоит из классов Java описания объектов управления Solaris Operating Environment; Solaris Provider обеспечивает взаимодействие с Solaris Operating Environment и CIMOM.

Спецификация CIM не описывает конкретизацию данных в приложениях, обмен данными между подсистемами управления. Первая задача решается с помощью языка разметки XML и языка преобразования XSL, а вторая – HTTP. Международная организация по стандартизации спецификаций в Интернет W3C (World Wide Web Consortium) определяет XML (eXtensible Markup Language) как язык предназначенный для описания в текстовом виде структурированных данных. Данные, описанные в XML называются XML-документами. XSL (eXtensible Stylesheet Language) является языком трансформации и форматирования XML-документов.

Фактически XML является набором правил определения возможности вставки тегов в поток текста для структурирования этого документа [14]. Этот язык с помощью своих соглашений дает структуру, которая отображает теги идентификации объектов или атрибутов на некоторую внешнюю семантическую структуру.

Проблему стандартизации словарей и структур данных для корректного применения в разных приложениях взяла на себя рабочая группа по распределенному управлению DMTF (Distributed Management Task Force). С помощью механизмов определения типов документов DTD (Document Type Definition) описывается грамматика документа XML путем перечисления всех допустимых в нем элементов и определения их структуры. DTD размещаются в месте, доступном всем приложениям, которые будут использовать этот документ (они могут быть размещены и в самом документе).

Документы XML содержат также и характеристики отображения. Стандартом таблиц стилей документов является язык спецификации семантики и стилей документов DSSSL (Document Style Sheet and Semantics Specification Language), предшественника XML стандартного обобщенного языка разметки SGML (Standard Generalized Markup Language). Форматироваться и отображаться XML-документы могут и с помощью каскадных таблиц стилей CSS (Cascading Style Sheet). Наилучшие возможности этих двух языков форматирования вобрал в себя язык XSL. С помощью последнего можно организовать разнообразное отображение элементов данных из XML-документов: таблицы свойств, графики, диаграммы, разные типы графики, интерфейс стандартной командной ленты, стандартные типы данных языков программирования и тому подобное.

Обработка XML-документа проходит в два этапа [14].

На первом этапе документ обрабатывает специализированная программа-анализатор (C++ -анализатор, java-анализатор) согласно следующему алгоритму:

1. Проводится сбор XML-данных из разных источников (локальные файлы, документы веб-серверов, COM-потоки и т.п.).
2. Проводится проверка синтаксической правильности документа, синтаксический анализ.
3. Если документ содержит элементы определения типов данных DTD, то проверяется семантическая верность документа.
4. Строится древовидная структура элементов данных.
5. Для доступа к данным извне последняя преобразуется в дерево объектов.

На втором этапе обработки XML-документа данные становятся доступными всем внешним приложениям. Дальнейшая работа с данными организуется через объектную модель, в которой документ реализуется с помощью древовидной структуры (object-document).

Итак, общедоступные данные уже готовы для обработки. Для того чтобы они были обработаны в сети, нужно средство их передачи между модулями сети (приложениями сети). Естественно, это средство должно удовлетворять таким основным требованиям: реализовано на всех Веб-браузерах; его реализация должна требовать незначительных ресурсов операционной системы; информация, которая передается, должна быть надежно защищена от несанкционированного доступа. Этим требованиям удовлетворяют модификации протокола HTTP типа Secure HTTP и Secure Sockets Layer (SSL). Последние позволяют организовать в WBEM защищенные сеансы управления как между приложениями, так и между пользователем и управляемым объектом.

Описанная модель уже нашла свое практическое воплощение. Много признанных компаний предложили свои версии реализации соответствующей идеологии WBEM. Например, фирма Manage.Com разработала такой продукт как Frontline e.M, который содержит клиента, на базе браузера для мониторинга, контроля, конфигурации и диагностирования компонентов бизнес-процессов и транзакций. Серверное программное обеспечение e.M выполняется на Web-сервере и способно: автоматически находить сервисы, приложения и устройства, контролировать IP-протокол; конфигурировать и сопоставлять данные от объектов управления; отвечать за сообщение об аварийных ситуациях; помогать составлять отчеты и реализовывать правила; обеспечивать использование идей агентных технологий за счет использования специализированного инструментария e.Agents и e.Boots; сохранять схемы описания объектов управления, правила конфигурирования и информацию о диагностике, производительности и загруженности сервисов за счет использования портала управления e.Registry.

Заключение

Последние договоренности между основными разработчиками как программного обеспечения, так и сетевого оборудования в рамках международной организации стандартизации спецификаций для Интернета World Wide Web Consortium (W3C) по практической реализации подхода WBEM, в конце концов, могут обеспечить осуществление заветной мечты многих поколений программистов, решение вечной проблемы – эффективного управления гетерогенными сетевыми средствами; заложить единую базу построения специализированного инструментария, ориентированного на эффективное решение многих прикладных задач, связанных с работой в сетях и упростить переход на новые версии программных продуктов.

1. http://www.cs.yale.edu/HTML/YALE/CS/Linda/linda-lang.html
2. *Ciancarini P, Guerrini N.* Linda Meets Minix // ACM SIGOPS Operating Systems Review. – 1993. – **27**. N 4, October.
3. *Foster I., Kesselman C., Tuecke S.* The Anatomy of the Grid. Enabling Scalable Virual Organizations // Intl. J. Supercomputer applications, 2001.
4. *Капитонова Ю.В., Летичевский А.А.* Математическая теория проектирования вычислительных систем. – М., Наука, 1986 – 206 с.
5. *Анисимов А.В.* Технология программирования ПАРУС // Разработка ЭВМ нового поколения архитектура, программирование, интеллектуализация, Новосибирск, 1986, С. 104–109.
6. *Інформаційна* система Національного університету “Києво-Могилянська академія” / М.М Глибовець., С.С Гороховський. О.Л. Синявський та ін. // Наукові Записки НаУКМА. – 1999. –**16**. – С. 4–16.
7. *Глушков В. М.* Об архитектуре высокопроизводительных ЭВМ. – Киев, 1978. – 16 с. – (Препр. / ИК АН УССР; 78-65).
8. *Бар Р.* Язык Ада в проектировании систем: Пер. с англ. – М.: Мир, 1988. – 320 с.
9. *Алгоритмический* язык Маяк / С.С. Гороховский, Ю.В. Капитонова, А.А. Летичевский и др. // Кибернетика. – 1984. – № 3. – С. 54–74.
10. *Анисимов А.В.* Технология программирования ПАРУС // Разработка ЭВМ нового поколения архитектура, программирование, интеллектуализация. Новосибирск: Изд-во ВЦ СО АН СССР, 1986. – С. 104–109.
11. *Гороховский С.С.* Распределенная операционная система МВК ЕС 1766 // Докл.2-ого Всесоюз. семинара "Программное обеспечение многопроцессорных систем. – 1988, Калинин. – С. 5–9.
12. *James E.* White, Mobile Agents // in Jeffrey M. Bradshow, ed. Software Agents. – MIT Press, 1997. – P. 437–472.
13. *Multiagent* Systems // A Modern Approach to Distributed Artificial Intelligence. – G. Weiss ed. MIT Press, 2000. – P. 619.
14. Extensible Markup Langage, http://www.w3.org/XML/