CAEBeans Server: среда выполнения проблемно-ориентированных оболочек над инженерными пакетами*

Р.С. Федянина

В работе представлена организация CAEBeans Server – среды выполнения проблемно-ориентированных оболочек над инженерными пакетами, разработанной в рамках реализации технологии CAEBeans. Описаны структура и алгоритмы работы среды, структура CAE-проекта, метод организации и управления распределенными вычислениями.

1. Введение

В настоящее время применение систем класса CAE (Computer Aided Engineering) обуславливает успех в производстве сложной технологической продукции, а так же научно-исследовательской работе, связанной с проведением экспериментов, реализовать которые в реальности практически невозможно. Точность расчетов производимых в таких системах часто во многом зависит от степени детализации сеток расчетных моделей. На сегодняшний день размер сетки может составлять десятки миллионов элементов [1]. В связи с этим задачи инженерного моделирования характеризуются высокой вычислительной сложностью и требуют значительные вычислительные ресурсы.

Решить данную проблему позволяет использование мощных многопроцессорных систем, поскольку современные CAE-пакеты имеют параллельные реализации. Современный опыт использования суперкомпьютерных систем показывает, что максимальная эффективность использования вычислительных ресурсов может быть достигнута при объединении таких систем в вычислительные грид-сегменты [2, 3].

Что касается организации грид сегодня, для этого, разработано несколько видов программного обеспечения промежуточного слоя. Но конечный пользователь CAE-систем, не являясь специалистом в данной области, при попытке организации взаимодействия с компонентами грид, сталкивается с рядом проблем связанных с получением сертификатов, регистрацией в виртуальной организации, сложным низкоуровневым интерфейсом и т.д. Кроме этого, программное обеспечение данного класса не обладает гибкими средствами манипулирования ходом решения конкретной задачи (подзадачи).

Решение проблем интеграции CAE-систем в среду грид предлагает технология CAEBeans [4, 5]. В основе технологии лежит обеспечение сервис-ориентированного предоставления программных ресурсов базовых компонентов CAE-систем и формирование иерархий проблемно-ориентированных оболочек, инкапсулирующих процедуру постановки и решения определенного класса задач. Технология CAEBeans регламентирует процесс декомпозиции задачи в иерархию подзадач; поиск вычислительных ресурсов; сопоставление задачам соответстующих базовых компонент CAE-систем; мониторинг хода решения задач; передача результатов решения задач пользователю [6].

CAEBeans Server – одна из базовых компонент системы, разработанной для поддержки/реализации технологии CAEBeans. CAEBeans Server отвечает за решение инженерной задачи, представленной в виде CAE-проекта (комплекс взаимосвязанных проблемных, потоковых и компонентных оболочек CAEBeans), в распределенной вычислительной среде.

В работе описана архитектура CAEBeans Server, метод организации и управления грид-вычислениями с использованием программных систем класса CAE, а также роль и место CAE-Beans Server в комплексе CAEBeans.

* Работа выполняется при финансовой поддержке программы СКИФ-ГРИД (договор № 2009-СГ-03).

2. CAEBeans Server

CAEBeans Server – это часть системы CAEBeans, на которой хранятся проблемно-ориентированные оболочки, результаты их исполнения, а также осуществляется генерация и управление процессом исполнения данных оболочек в распределенной вычислительной среде.

При проектировании к CAEBeans Server предъявлялись следующие требования: независимость организации системы от используемой вычислительной платформы; обеспечение безопасного доступа к вычислительным ресурсам через сеть интернет; функционал среды должен быть доступен посредством некоторого набора сервисов со стандартизированными интерфейсами; поддержка стандартов SOA и WSRF.

Исходя из предъявляемых требований, было принято решение о том, что CAEBeans Server будет реализован на языке java в виде грид-службы.

2.1 Структура CAEBeans Server

CAEBeans Server (далее Server) имеет модульную структуру и включает в себя следующие компоненты:

- **Хранилище CAE-проектов (Storage Project)** отвечает за хранение CAE-проектов [7] (в формате xml) и библиотеки компонентных CAEBeans.
- **Хранилище результатов исполнения CAE-проектов (Storage Result)** отвечает за хранение промежуточных и конечных результатов, полученных в ходе исполнения CAE-проекта в распределенной среде грид. Данные хранятся пока не удалено соответствующее CAE-задание [7].
- **Целевое хранилище (Storage Target)** отвечает за хранение: URL доступных целевых систем с соответствующими сертификатами доступа к ЦС; параметров (имена) лицензий и key-файлов, необходимых при постановки задания конкретному вычислительному ресурсу.
- **Интерпретатор CAE-проекта (Executor)** – активный класс, управляющий исполнением CAE-проекта, соответствующего уникальному CAE-заданию с уникальным набором пользовательских параметров.
- **Подсистема Client Broker** отвечает за взаимодействие с подсистемой CAEBeans Broker.
- **Подсистема Client Unicore** отвечает за взаимодействие с вычислительными узлами грид, на которых производится расчет.
- **Служба создания и управления CAE-заданиями (Task Manager)** отвечает за создание, уничтожение и мониторинг экземпляров сервиса Task.
- **Служба исполнения CAE-задания (Task)** отвечает за исполнение конкретного CAE-проекта.
- **API** – интерфейсная часть грид-служб. Включает в себя интерфейсы служб Server и Task, а так же функции администрирования Server.

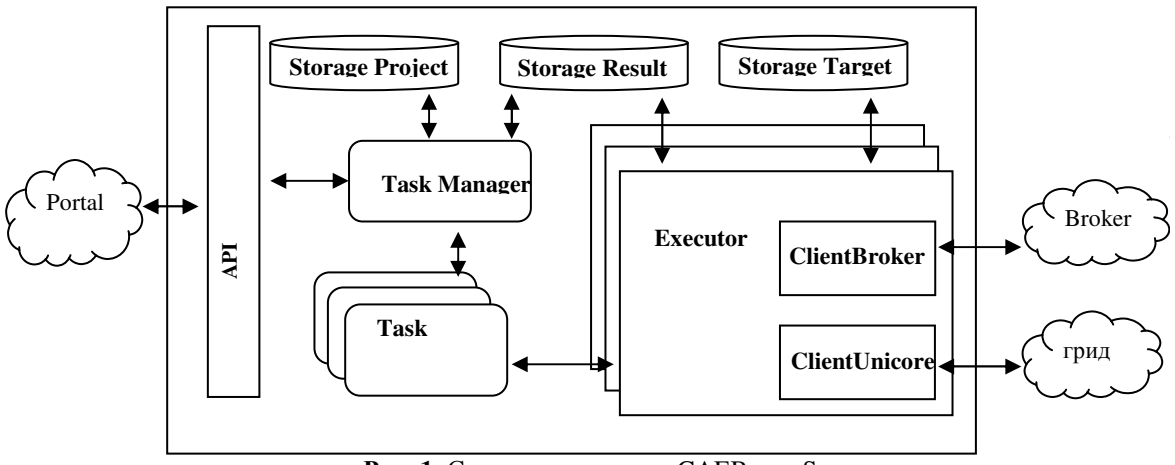

Рис. 1. Схема архитектуры CAEBeans Server.

На рисунке 1 показана схема взаимодействия подсистем Server, а также роль и место Server в комплексе CAEBeans. Рассмотрим общую схему работы Server в среде CAEBeans.

Исполнение CAE-проекта инициируется внешним источником, таким как CAEBeans Portal (далее Portal). На первом этапе Portal формирует запрос на создание экземпляра грид-сервиса обработки задания и отправляет на Server, грид-сервису приема запросов Task Manager. Task Manager, в свою очередь, осуществляет проверку наличия в Storage Project соответствующего CAE-проекта. В случае, если необходимый CAE-проект найден, Task Manager создает экземпляр грид-сервиса обработки задания (Task) и запускает его. При этом идентификатор созданного экземпляра сервиса возвращается Portal-у (рисунок 2) для организации дальнейшего взаимодействия. На следующем этапе Portal формирует запрос, адресованный созданному экземпляру сервиса, на исполнение CAE-проекта, в котором содержится некоторая проблемная оболочка концептуального слоя CAE-проекта с заданными пользователем параметрами. При получении данного запроса сервисом Task создается процесс Executer, который отвечает за исполнение CAE-проекта. Для обеспечения полноценного взаимодействия Portal и Server предусмотрены еще три типа запросов: на удаление экземпляра сервиса; получение информации о текущем статусе и промежуточных результатах исполнения CAE-проекта; получение результатов исполнения CAE-проекта.

```
<?xml version="1.0" encoding="UTF-8"?>
<task:result xmlns:add="http://www.w3.org/2005/08/addressing" xmlns:tas
k="http://caeTask" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http: //schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <add:Address> http://ural.susu.ac.ru:7780/services/task?res=7894485a-
7a24-43f3-8a46-2935bc86eaf7</add:Address>
</task:result>
```

Рис. 2. Пример ответного сообщения на запрос создания экземпляра сервиса Task. Возвращает уникальный идентификатор созданного экземпляра сервиса.

При запуске процесса Executer в качестве параметра передаются адрес директории в Storage Result, созданной Task Manager при создании экземпляра сервиса Task. К моменту запуска процесса Executer в данную директорию копируются все необходимые элементы CAE-проекта из Storage Project, а так же переданная с Portal проблемная оболочка с заданными пользователем значениями параметров. В ходе исполнения CAE-проекта все промежуточные и заданные конечные результаты будут сохраняться в этой же директории.

Перед исполнением CAE-проекта Executer-ом формируется дескриптор задачи. Дескриптор задачи представляет собой xml-файл определенной структуры (рисуноке 3), в котором содержится информация о значения параметров на текущем шаге решения. Посредством данных параметров производится постановка инженерной задачи, а также взаимодействие частей логического плана.

```
<?xml version="1.0" ?>
 <ParamDesc>
      <parameter id="int_1" type="Integer" value="16" />
      <parameter id="length1" type="Float" value="45" />
            …
      <parameter id="exec_bean_1" type="String" value="exec -h" />
  </ParamDesc>
```

Рис. 3. Пример файла дескриптора задачи.

После формирования дескриптора задач, Executer приступает к исполнению логического плана решения CAE-проекта. Узлы логического плана такие как: начальный, конечный, условный, ветвления и слияния исполняются непосредственно на Server. Узел деятельности исполняется на вычислительном узле грид. На период исполнения каждого узла деятельности CAE-проекта Executer-ом формируется поддиректория, в которой исполняется соответствующий

данному узлу деятельности компонентный CAEBean. Сформированные после его исполнения файлы (в некоторых случаях могут отсутствовать) и строка для запуска (набор передаваемых параметров) клиента Unicore [8] используются для запуска системного CAEBean. Системный CAEBean исполняется на вычислительном узле грид и представляет собой проблемно зависимый слой системы, инкапсулирующий процесс решения задачи в CAE-пакете.

Для определения адреса оптимального вычислительного узла Server обращается к подсистеме Broker. Взаимодействие с данной подсистемой осуществляется по специальному протоколу [9], обеспечивающему гарантированный отклик релевантного содержания. В качестве параметров запроса к Broker передается информация о выполняемом системном CAEBean и предпочтительных характеристиках вычислительного ресурса: архитектура процессора, количество процессоров/ядер, максимальное время выполнения задания в часах, время таймаута сервера, объем оперативной памяти в Мб, операционная система, объем памяти на жестком диске.

После того, как адрес потенциального вычислителя получен, посредством подсистемы ClientUnicore, производится соединение с целевой системой Unicore соответствующего вычислительного узла, копирование необходимых файлов, запуск системного CAEBean, по завершению исполнения копирование полученных файлов в Storage Result. Необходимые для подключения к целевой системе данные извлекаются из Storage Target. Использование средств Unicore, при организации взаимодействия между вычислительными узлами, обеспечивает соблюдение всех необходимых стандартов организации грид, а также необходимого уровня безопасности.

По завершению расчета текущего шага решения, Server уведомляет Broker об освобождении вычислительного ресурса. Как только CAE-проект исполнен полностью, либо вследствие возникновения критической ошибки остановлен, об этом оповещается Portal. В случае, если проект был выполнен успешно, Portal может сформировать запрос на выгрузку результатов. В противном случае, Portal в ответ на тот же запрос получит промежуточные результаты, полученные на завершенных этапах решения задачи и лог исполнения CAE-проекта, который позволит определить проблемы, возникшие при исполнении CAE-проекта.

3. Файловая структура CAE-проекта

Для разработки CAE-проекта используется среда CAEBeans Constructor (далее Constructor) [6], разработанная в рамках реализации технологии CAEBeans. При компиляции CAE-проекта в Constructor для экспорта в Server создаются следующие файлы:

1. Project.xml содержит описание логического, физического слоя и описание дерева проблемных оболочек. Пример описания соответствующих слоев для задачи, логический план которой состоит из трех узлов (начального, узла деятельности и конечного), приведен на рисунке 4.
2. <Имя проблемной оболочки>.xml содержит описание концептуального слоя. Количество таких файлов соответствует, количеству проблемных оболочек в дереве и характеризуется уникальным идентификатором. Сведения о соответствие уникальных идентификаторов именам проблемных оболочек содержится в файле Project.xml.
3. <Имя компонентного CAEBean>.jar – исполняемый java архив соответствующий определенному компонентному CAEBean либо узлу управления.

```
  <logical>
    <node id="begin01" type="begin" condition="condition/>
    <node id="ansys" type="active" condition="condition" execute="an1"/>
    <node id="end01" type="end" condition="condition"
    <edge source="begin01" target="ansys" when="True" />
    <edge source="ansys" target="end01" when="True" />
  </logical>
  <physical>
      <componentCaebean name="an1">
            <input id="length1" type="integer"/>
            <input id="length2" type="integer"/>
            <input id="file1.db" type="file"/>
            <exec id="Ansys_01"/>
       </componentCaebean>
  </physical>
</conceptual>
```

Рис. 5. Фрагмент xml-описания логического и физического слоев CAE-проекта.

4. Алгоритм выполнения CAE-проекта.

За выполнение CAE-проекта в Server отвечает подсистема Executer. В процессе выполнения CAE-проекта Executer задействует концептуальный, логический и физический слои по мере прохождения логического плана решения. Весь процесс документируется в файле <IdTask>.out. Таким образом, в нем содержится подробная информация о том, на каком узле производился расчет конкретного шага решения и сколько времени он занял, какие ресурсы были задействованы, время постановки и старта расчета. Частично данная информация используется подсистемой Task Manager при осуществлении мониторинга, а также может быть предана пользователю для анализа хода решения задачи. Возникающие в ходе исполнения проекта ошибки записываются в < IdTask >.err.

4.1 Обработка концептуального слоя (структура дескриптора задач).

Концептуальный слой содержит параметрическое описание задачи. По его XML-описанию строится дескриптор задачи. Дескриптор задачи используется при исполнении CAE-проекта на каждом шаге логического плана для извлечения и размещения текущих значений параметров. В качестве начальных значений параметров устанавливаются значения, введенные пользователем через интерфейс Portal, либо значения указанные в проблемной оболочке как значения по умолчанию. Допустимы параметры с пустым значением (атрибут value=“”), это свидетельствует о том, что значение данного параметра будет установлено на определенном этапе в ходе решения задачи. После каждого шага решения дескриптор задачи обновляется. Структура дескриптора задач приведена на рисунке 3.

4.2 Обработка логического слоя

Данный слой содержит описание логического плана решения задачи, по которому исполняется CAE-проект. Логический план CAE-проекта представляет собой ориентированный граф. В вершинах графа могут находиться блоки двух типов: подзадачи, выполняемые отдельными базовыми компонентами, и специальные служебные операции управления потоком решения задачи (такие, как ветвление, распараллеливание и т.п.). Блоки логического плана решения задачи представляют собой набор узлов шести видов (начальный узел, конечный узел, узел деятельности, узел управления, узел ветвления, узел слияния). По их xml-описанию в памяти строится дерево, корнем которого является начальный узел, и вызывается метод execute данного узла. Далее рассмотрим, как обрабатываются узлы каждого типа.

Начальный и конечный узлы обрабатываются непосредственно на сервере. Начальный узел инициирует создание дескриптора задания. Конечный узел свидетельствует об успешном окончании исполнения CAE-проекта и возвращает управление.

Узел управления (if) содержит условие, проверяемое на сервере. Проверка условия осуществляется посредством исполнения метода java, описание которого содержится в соответствующем исполняемом java архиве, входящего в состав CAE-проекта. Таким образом, структура и формат проверяемого условия определяется CAE-программистом в момент создания CAE-проекта и не ограничена конкретным синтаксисом. Однако входными и выходными параметрами проверяемого условия могут быть только переменные, описанные в дескрипторе задачи. Дальнейшее движение по логическому плану происходит в направлении соответствующей результату проверки условия ветке.

Узел деятельности является ключевым. Исполнение данного узла осуществляется на вычислительном узле грид. Каждому узлу деятельности соответствует определенные компонентный и системный CAEBean. Компонентный CAEBean исполняется на сервере. Его основная задача заключается в подготовке входных и обработки выходных данных системного CAEBean. Системные CAEBeans размещены на вычислительных узлах взаимодействующей с Server грид-среды и реализованы как задания, описанные в целевой системе Unicore.

Узел ветвления отвечает за формирование параллельных веток логического плана. В Server заложены два алгоритма обработки параллельных веток: первый предполагает автоматическую генерацию произвольного количества веток одинаковой структуры, второй алгоритм позволяет обрабатывать заранее заданный набор произвольных веток возможно различной структуры. В первом случае CAE-программистом задается структура одной параллельной ветки, границы диапазона (два параметра проблемного CAEBean) входных значений параметра (-ов) и шаг изменения параметра. При исполнении CAE-проекта необходимое количество веток будет сформировано автоматически. Во втором случае для каждой параллельной ветки CAE-программисту необходимо описать ее структуру и задать входные параметры. Обработка данных параллельных веток осуществляется следующим образом. Для каждой параллельной ветки во временной директории узла ветвления создается временная поддиректория, в которую помещаются необходимые для исполнения файлы и локальная копия дескриптора задачи с соответствующими параметрами.

Узел слияния является парным узлу ветвления, поэтому все необходимые файлы копируются в директорию предшествующего узла ветвления. Узлу слияния соответствует свой компонентный CAEBean, который исполняется на сервере по тому же алгоритму, что и компонентный CAEBean узла деятельности. Его основная функция – анализ локальных копии дескрипторов задачи параллельных веток и формирование данных общего дескриптора задачи. Данный узел не предполагает наличие соответствующего системно CAEBean, т.е. исполнение данного шага завершается, как только завершил работу компонентный CAEBean.

4.3 Обработка физического слоя

Оболочками физического слоя являются компонентные CAEBeans [4]. Компонентный CAEBean, представляет собой java архив, который может быть запущен с одним из двух видов ключей: «-p» (от англ. prepare) для подготовки строки запуска системного CAEBean и всех необходимых для этого данных; и «-w» (от англ. work) для обработки полученных, в результате работы системного CAEBean данных. При запуске компонентного CAEBean с ключом «-p» могут передаваться параметры. Список входных параметров компонентного CAEBean определяется по xml-описанию физического (рисунок 5) слоя CAE-проекта, а значения извлекаются из дескриптора задачи.

В результате работы компонентного CAEBean, запущенного с ключом «-p», генерируется файл <Имя системного CAEBean>.u и файлы необходимые для работы системного CAEBean, которые требуется передать на вычислительный узел. В файле <Имя системного CAEBean>.u (рисунок 6) содержится информация регламентирующая выполнение системного CAEBean: количество, порядок и значения передаваемых при запуске параметров; файлы, которые необходимо передать на вычислительный узел перед запуском системного CAEBean; файлы которые требуется скачать с вычислительного узла, после того как системный CAEBean завершит

исполнение. Далее Executer, посредством подсистемы Client Unicore последовательно производит соединение с целевой системой Unicore, старт системного CAEBean и переход в режим ожидания равный заранее заданному значению (данный параметр может передаваться как один из параметров CAE-проекта). Выход из режима ожидания может быть досрочным в случае получения оповещения о завершении исполнения задания.

```
{ ApplicationName: Ansys_01,
  ApplicationVersion: 1.0,
# environment settings
Environment: [ "SOURCE=antstat.bat",],
Imports: [
    { File: antstat.bat, To: antstat.bat },
    { File: p.lgw, To: p.lgw },],
Exports: [
    { File: file000.png, To: file000.png },
  ],}
```

Рис. 6. Пример файла запуска задания Unicore.

На следующем этапе исполнения узла деятельности компонентный CAEBean запускается с ключом «-w». В результате работы компонентного CAEBean в этом режиме обрабатываются полученные в ходе работы системного CAEBean данные, обновляется дескриптор задания, необходимые файлы (необходимыми являются те файлы, имена которых являются значениями параметров дескриптора задач и были сформированы/изменены на текущем шаге решения) копируются из временной директории текущего шага решения задачи в корневую директорию CAE-задания. После этого временная директория текущего шага решения удаляется. Executer переходит к следующему узлу логического плана.

Заключение

В статье представлена архитектура среды CAEBeans Server, исполняющей функции: хранилища проблемно-ориентированных оболочек и результатов их исполнения; генерации и управления процессом выполнения данных оболочек в грид-среде. Описана файловая структура CAE-проекта и алгоритм выполнения проблемно-ориентированных оболочек, описан процесс взаимодействия компонент Portal и Broker системы CAEBeans со средой CAEBeans Server.

В качестве дальнейших направлений работы можно выделить следующие: добавление в алгоритм обработки логического плана решения, нового типа узла деятельности, отвечающего за подключение системы многокритериальной оценки; реализация сервиса загрузки проблемно-ориентированных оболочек из CAEBeans Constructor; исследование и реализация подходов к организации распределенного хранилища проблемно-ориентированных оболочек и результатов их исполнения.

Литература

1. Бегунов А.А. Применение результатов моделирования для оптимизации и управления технологическими процессами // Параллельные вычислительные технологии: Тр. Междунар. науч. конф. (28 янв. – 1 февр. 2008 г., г. Санкт-Петербург). -2008. -С. 31-38.

2. Foster I., Kesselman C. The Grid 2: Blueprint for a New Computing Infrastructure. Second edition. -San Francisco: Morgan Kaufmann, 2003. -750 p.

3. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International J. of Supercomputer Applications and High Performance Computing. -2001. -T. 15, No 3. -P. 200-222.

4. Радченко Г.И. Технология построения проблемно-ориентированных иерархических оболочек над инженерными пакетами в грид-средах // Системы управления и информационные технологии. 2008. № 4(34). С. 57-61.

5. Радченко Г.И., Соколинский Л.Б. Технология построения виртуальных испытательных стендов в распределенных вычислительных средах // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. № 54. 2008. С. 134-139.

6. Радченко Г.И. Грид-система CAEBeans: интеграция ресурсов инженерных пакетов в распределенные вычислительные среды // Параллельные вычислительные технологии (ПаВТ'2009): Труды международной научной конференции (Нижний Новгород, 30 марта – 3 апреля 2009 г.). Челябинск: Изд-во ЮУрГУ, 2009. С. 281-292.

7. Проект CAEBeans. URL: http://sites.google.com/site/caebeans/ (дата обращения: 27.11.09).

8. Официальный сайт проекта Unicore. URL: http://www.unicore.eu (дата обращения: 27.11.09).

9. Шамакина А.В. Протокол взаимодействия с брокером ресурсов в системе CAEBeans // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийск. науч. конф. (21-26 сентября 2009 г., Новороссийск). М.: Изд-во МГУ, 2009. С. 400-402.