Нижегородский государственный университет им. Н.И.Лобачевского

Межфакультетская магистратура по системному и прикладному программированию
для многоядерных компьютерных систем

Учебный курс «Технологии построения и использования кластерных систем»

Раздел Обзор системы управления кластерами «Метакластер»

Разработчик: А.В. Сенин

Нижний Новгород

2007

Оглавление

1. Введение

Для эффективного использования вычислительных ресурсов необходимо предоставить круглосуточный доступ к кластеру через Интернет, обеспечить своевременный запуск заданий пользователя в соответствии с принятой политикой планирования, включающей, например, ограничения на одновременный запуск нескольких заданий на одной машине и остановку программ, исчерпавших отведенный им лимит времени. Также необходимо вести постоянный учет времени работы программ, протоколирование всех выполненных действий, предоставление инструментов контроля и администрирования кластера.

Все это обуславливает необходимость создания некой программной среды, посредством которой будет происходить запуск заданий на кластере.

На сегодняшний день известно достаточно много программных систем, предоставляющих удаленный доступ к вычислительным ресурсам и обеспечивающих планирование запуска задач. Большинство таких систем разрабатываются для ОС семейства UNIX. Однако в последнее время подобные системы появляются и для ОС семейства Windows. К числу таких относятся, например, Microsoft CCS 2003 (Microsoft Compute Cluster Server 2003, [1]), LSF (Load Sharing Facility, [2]), Cluster CoNTroller ([3]).

Но использование готовых систем осложняется, с одной стороны, их высокой стоимостью, достигающей нескольких десятков тысяч долларов, с другой стороны, закрытостью исходного кода, делающей затруднительным проведение исследовательских работ. Дело в том, что кроме обеспечения бесперебойной работы вычислительного кластера система управления может быть использована как база для апробации алгоритмов планирования запуска задач на выполнение и мониторинга вычислительных ресурсов. Эффективное планирование распределения вычислительной нагрузки является практически важной задачей в силу частой неоднородности компьютерных ресурсов кластерных вычислительных систем.

Отмеченные факторы приводят к целесообразности создания собственных средств организации высокопроизводительных вычислений (в частности и на базе кластера Нижегородского университета).

2. Требования к системе управления кластерами Метакластер

В ННГУ имеется несколько мощных вычислительных ресурсов, среди которых: 128 процессорный кластер, приобретенный в рамках национального проекта с пиковой производительнстью 3 Tflops, кластер из 15 узлов на базе процессоров Intel Xeon, мини-кластер T-Forge Mini, состоящий из 8 процессоров AMD Opteron 2.2 GHz, ряд компьютерных лабораторий, используемых в учебном процессе и для проведения научно-исследовательских экспериментов. Для повышения эффективности использования этих компьютерных ресурсов, обеспечения отказоустойчивости даже в случае временной недоступности отдельных ресурсов, улучшения контроля и администрирования полезно объединить вычислительный кластер и компьютерные лаборатории (далее для простоты мы их также будем называть «кластерами») в единую вычислительную инфраструктуру под управлением одной системы управления кластерами, требования к которой могут быть сформулированы следующим образом.

Интегрированная система управления кластерами должна обеспечивать:

1. Управление несколькими кластерами на базе различных операционных систем: Windows, Linux;
2. Интеграцию с системами управления третьих разработчиков (в случае использования таковых в компьютерных лабораториях и невозможности полного перехода на новую систему): Microsoft Compute Cluster Server 2003, PBS, Condor и др.;
3. Простой и удобный доступ, не требующий установки на рабочих станциях пользователей какого-либо специального программного обеспечения, позволяющий получить доступ к системе из любой точки (веб-интерфейс);
4. Командный и графический интерфейсы;
5. Администрирование через специальный интерфейс, управляющий ходом работы;
6. Программный интерфейс (веб-сервис, COM-интерфейс и др);
7. Авторизацию пользователей через собственную систему, не связанную напрямую с системой авторизации операционных систем;

8. Реализация необходимого набора операций для выполнения задач на одном из кластеров, находящихся под управлением системы:
   8.1. Загрузка задачи на сервер;
   8.2. Добавление задачи в очередь задач (под очередью задач мы понимаем множество всех задач, ожидающих выполнения на кластерах);
   8.3. Получение текущего статуса очереди задач и задач, находящихся в этой очереди;
   8.4. Получение результатов вычислений (перехваченный поток вывода и файлы, созданные программой);
9. Сохранение задач пользователя после их выполнения и возможность их повторного запуска;
10. Автоматическое распределение задач по кластерам и далее – по узлам выбранного кластера;
11. Автоматическое сохранение результатов работы;
12. Устойчивость к отказам отдельных кластеров и отдельных узлов каждого кластера;
13. Ведение статистики использования компьютерных ресурсов, автоматическая генерация отчетов.
14. Надежность и переносимость системы для использования ее на кластерах широкого типа (при проектировании системы не должно приниматься решений, ограничивающих ее использование кластером ННГУ).

3. Структура системы управления кластерами Метакластер

В данном разделе приведено общее описание системы управления кластерами Метакластер. Система состоит из 3 основных компонент: менеджер удаленного доступа, интегратор кластеров (сервис, отвечающий за объединение кластеров в единую вычислительную инфраструктуру) и менеджеры заданий.

3.1. Общее описание

Серверная часть (интегратор кластеров и менеджер заданий) системы Метакластер написана на языке программирования C++, обеспечивающем высокую производительность вместе с богатыми возможностями объектно-ориентированного языка программирования. Основная среда разработки – Microsoft Visual Studio 2005. Веб-интерфейс написан на базе платформы .NET 2.0, на ASP.NET и C#. Функциональность веб-интерфейса дублируют веб-сервисы, делающие возможным доступ к системе из любой программы. В качестве сервера базы данных используется Microsoft SQL Server 2005.

3.1.1. Абстракция задачи

Базовое понятие системы управления кластерами – понятие вычислительной *задачи*.

Задача в системе Метакластер представляет собой исполняемый файл программы и набор дополнительных параметров:

1. Имя задачи (может не совпадать с именем исполняемого файла).

Параметр задает пользователь, поставивший задачу в очередь заданий. Имя задачи необходимо для упрощения работы конечного пользователя с системой через интерфейс менеджера удаленного доступа.

2. Уникальный идентификатор задачи в системе.

Параметр необходим для однозначной идентификации задачи менеджером заданий, интегратором кластеров и веб-сервисами.

3. Командная строка, передаваемая программе при запуске.
4. Число и тип необходимых вычислительных ресурсов.

3.2. Архитектура системы

На Рис. 1 показана диаграмма процессов системы Метакластер. Каждый элемент диаграммы представляет собой процесс операционной системы; взаимодействие между процессами показаны стрелками. Ниже приведено краткое описание процессов системы.

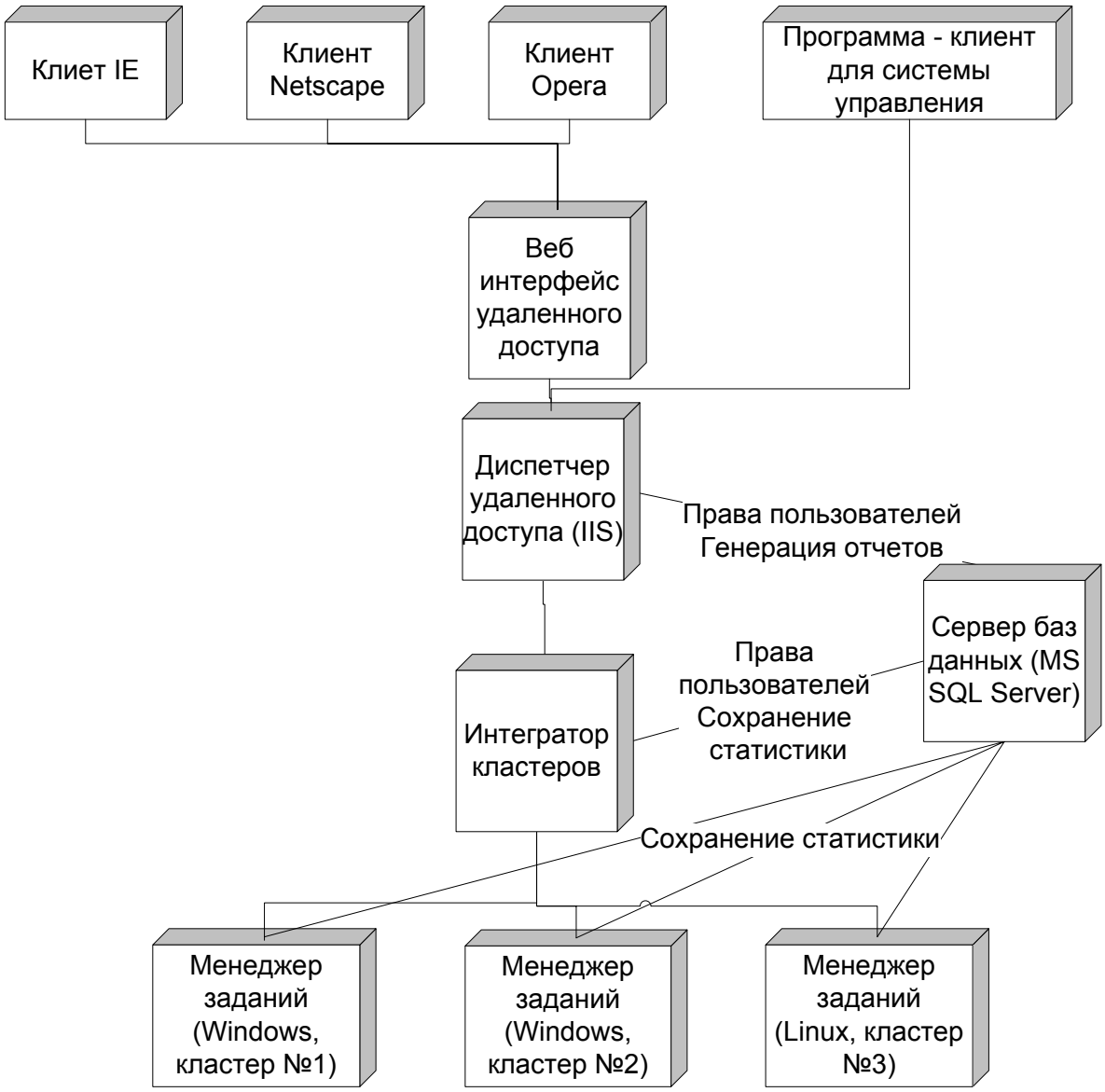

Рис. 1. Диаграмма процессов системы Метакластер

3.2.1. Диспетчер удаленного доступа

Диспетчер удаленного доступа представляет собой независимую компоненту, реализованную в виде .NET веб-сервиса, взаимодействующего с сервером системы через механизм сокетов по специально разработанному XML-протоколу (описание протокола взаимодействия см. в пункте «Формат сообщений между компонентами систем»). Как показано на Рис. 1, клиентами диспетчера удаленного доступа могут быть как обычные приложения, так и веб-интерфейс. В случае взаимодействия через веб-интерфейс добавляется еще один слой - ASP.NET страница, которая тоже взаимодействует с системой управления кластерами через веб-сервисы. Использование веб-сервисов необходимо для упрощения взаимодействия с системой управления кластерами клиентских программ и, в то же время, обеспечения кроссплатформенности (Windows, мобильные устройства на базе Windows Mobile, Linux, где к веб-сервисам можно обращаться с использованием Mono – открытой реализации .NET).

Основные функциональные возможности диспетчера удаленного доступа:

1. **Аутентификация пользователей** (с использованием собственной базы данных).

Каждый пользователь обладает уникальным буквенно-цифровым идентификатором («Имя пользователя»), а так же секретной кодовой последовательностью («Пароль»). Эта информация позволяет системе однозначно идентифицировать пользователя и предотвратить доступ посторонних к пользовательской информации (файлы программ, результаты экспериментов). Процедура регистрации новых пользователей осуществляется администратором системы через административный интерфейс.

2. **Выделение независимого файлового пространства**.

Каждому пользователю выделяется индивидуальное файловое пространство для хранения файлов программ и результатов эксперимента. Доступ к хранилищу осуществляется через веб-интерфейс, предоставляющий возможности как загрузки файлов на сервер, так и сохранения их на локальном жестком диске (а также чтения текстовых файлов без промежуточного их сохранения). Имеется возможность настройки индивидуальных для каждого пользователя ограничений на максимальный размер файлов, хранимых на сервере.

3. **Система именованных задач**.

После загрузки программы на сервер пользователь имеет возможность создать на ее основе задачу, указав уникальное имя задачи, имя файла с результатами (для сохранения перехваченного потока вывода), необходимое количество и тип вычислительных мощностей, параметры командной строки, а также некоторую дополнительную информацию. Затем пользователь может ставить и снимать задачу с выполнения, обращаясь по заданному идентификатору.

3.2.2. Интегратор кластеров

Компонент, являющийся прослойкой между веб – сервисом и менеджерами заданий. Присутствие этого уровня необходимо для достижения следующих целей:

1. Возможность автоматического распределения запуска задач по различным кластерам;
2. Обеспечение бесперебойной работы «Метакластера» в случае временной недоступности одного из кластеров. Так, в случае временных проблем на линии связи между интегратором кластеров и одним из менеджеров заданий интегратор может переадресовать особо приоритетные задачи с недоступного на доступный в настоящее время кластер;
3. Интегратор кластеров может быть установлен на той же машине, что и веб – сервисы, что обеспечивает более быстрый доступ к информации о состоянии задач в очереди заданий или другой статистической информации о работе «Метакластера», которую необходимо запрашивать не из базы данных (например, перехваченный поток вывода).

Функциональность интегратора кластеров:

1. Хранение общей для всех кластеров очереди заданий;
2. Предоставление статистической информации по запросу веб-сервиса (как минимум, ход выполнения заданий, информация о загруженности кластера, результаты вычислений);
3. Сохранение статистической информации в базу данных (в том числе, подробный список действий, выполненных системой);
4. Добавление/удаление задач по запросу веб – сервиса;
5. Планирование распределения задач по кластерам. Пользователь должен иметь возможность указать, на каком именно кластере он хочет посчитать свою задачу. Но в случае отсутствия пользовательских указаний система должна принять самостоятельное решение на основе имеющейся информации о загруженности кластеров, предыдущем опыте запуска и правах конкретного пользователя (информацию о том, на каких кластерах пользователь имеет право производить запуск, сообщается интегратору веб – сервисом);
6. Миграция заданий между кластерами (в перспективе).

3.2.3. Менеджер заданий

Компонент «Метакластера» (см. Рис. 2), осуществляющий непосредственный запуск заданий на кластере и мониторинг вычислительных узлов кластера. Компонент устанавливается на головной узел каждого кластера под управлением «Метакластера».

Функциональность менеджера заданий:

1. Поддержание локальной очереди задач, запущенных на том кластере, на котором менеджер заданий выполняется. Добавление задач осуществляется интегратором кластеров;
2. Планирование порядка запуска заданий на выполнение;
3. Запуск вычислительных задач на выполнение в соответствии с принятой локальной стратегией запуска;
4. Возможность принудительного снятия с выполнения задач по запросу интегратора кластеров;
5. Мониторинг вычислительных узлов кластера;
6. Сбор и сохранение результатов работы запущенных задач (перехваченный поток вывода и файлы результатов);
7. Сохранение в базе данных статистической информации о загруженности кластера (информация монитора вычислительных ресурсов), ходе выполнения заданий, результатах вычислений.

Важно отметить, что менеджеров заданий в системе может быть несколько (по одному на каждый имеющийся кластер). Каждый менеджер разрабатывается с учетом особенностей кластера, на котором он будет работать (операционная система, установленное программное обеспечение, возможно, аппаратные характеристики). Интегратор кластеров в системе предполагается только один.

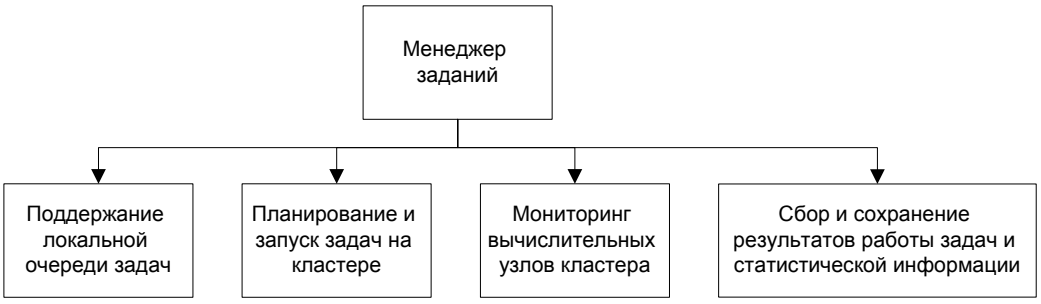

Рис. 2. Основные функции менеджера заданий системы «Метакластер»

3.3. Формат сообщений между компонентами системы

Веб-сервис посылает команды интегратору кластеров, а тот в свою очередь – менеджерам заданий, с использованием механизма сокетов – интерфейс обмена сообщений между процессами. Для этого был разработан специальный XML – формат команд, пример которого приведен ниже:

```
<?xml version="1.0" ?>
<Requests                    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="ServInegrMessages.xsd">
  <Command Type="AddTask">
    <TaskState>WAITING</TaskState>
    <TaskId>10</TaskId>
    <ModuleName>RunMe.exe</ModuleName>
    <Priority>1</Priority>
    <Name>Something special</Name>
    <OutputRedirection>OutPut.txt</OutputRedirection>
    <NumberOfProcesses>2</NumberOfProcesses>
    <CommandLine>1 2 3</CommandLine>
    <ActiveDir>\\srv114-1-5\temp</ActiveDir>
    <TypeOfProcessors>Any</TypeOfProcessors>
    <UserLogin>SeninAndrew</UserLogin>
    <UserId>1</UserId>
    <SessionId>1</SessionId>
  </Command>
</Requests>
```

Тег **Requests** содержит один или несколько подтегов **Command**. Тип каждой команды определяет атрибутом **Type** тега **Command**. Таким образом, в одном xml-файл мы можем передать неограниченное число команд за один раз. Допустимыми значениями атрибута **Type** являются следующие:

1. **AddTask** – добавить задачу в очередь (в случае интегратора кластеров – в общую очередь задач на всех кластерах в системе, в случае менеджера заданий – в локальную очередь кластера);
2. **DeleteTask** – удалить задачу из очереди;
3. **TaskState** – получить состояния задач.

В **Ошибка! Источник ссылки не найден.**приведено описание основных параметров команды.

Табл. 1. Описание параметров команды добавления задачи в очередь задач

Название параметра	***Описание параметра***
TaskState	Состояние задачи (см. пункт «абстракция задачи»)
TaskId	Уникальный идентификатор задачи
ModuleName	Имя исполняемого модуля (exe – файла)
Priority	Приоритет задачи (в настоящий момент параметр игнорируется)
Name	Имя задачи
OutputRedirection	Файл результата (сюда помещается перехваченный поток вывода параллельной программы)
NumberOfProcesses	Необходимое число машин
CommandLine	Командная строка
ActivDir	Директория, устанавливаемая программе по умолчанию
TypeProcessors	Необходимый тип компьютера для запуска задачи: – Any – тип компьютера не имеет значения – 1 - процессорная машина – 2-х процессорная машина – 4-х процессорная машина
UserLogin	Имя пользователя (логин в системе)
UserId	Уникальный идентификатор пользователя в системе
SessionId	Зарезервированный параметр (в настоящее время не используется)

Ответом на каждый запрос служит команда такого же формата. Например, в ответ на каждую команду типа **TaskState** интегратору будет возвращено сообщение, в котором тег **Command** имеет такое же значение атрибута и заполненное значение тега состояния задачи.

3.4. Файлы конфигурации

Для загрузки параметров системы разработан основанный на XML формат файлов. Всего существует 4 вида конфигурационных файлов:

1. **Clusters.xml** – задает список менеджеров кластеров, к которым будет подключаться интегратор, для каждого менеджера: имя компьютера, на котором он установлен, номер порта, тип операционной системы;
2. **Nodes.xml** – задает список узлов, включенных в соответствующий кластер;
3. **IntegratorParameters.xml** – задает параметры интегратора кластеров (порт, на котором интегратор будет ожидать команды от веб-сервиса, рабочая директория по умолчанию для запускаемых программ);
4. **ManagerParameters.xml** – задает параметры менеджера кластеров (параметры аналогичны параметрам интегратора).

3.5. Основные классы системы

Сервисная часть «Метакластера» (Интегратор и Менеджеры заданий) состоит более чем из 30 классов, основные из которых перечислены ниже:

1. **CTask** – абстракция задачи (задача в системе Метакластер представляет собой исполняемый файл программы (полный путь до программы) и набор дополнительных параметров);
2. **CTaskPool** – очередь задач;
3. **CNode** – абстракция вычислительного узла. Содержит наиболее актуальную информацию о доступности и загруженности узла, объеме оперативной памяти, свободного дискового пространства и др. информации, которая может быть использована при принятии решения о запуске задач на том или ином узле;
4. **CNodePool** – список вычислительных узлов, заполняемый монитором;
5. **CEvent** – абстракция события в системе (добавление, удаление, запуск, остановка задачи и др.);
6. **CCluster** – абстракция вычислительного кластера;
7. **CClusterPool** – список вычислительных кластеров, работающих под управлением интегратора кластеров;
8. **CEventPool** – список событий. В структуре хранятся только те события, которые еще не были сохранены в базе данных;
9. **CLauncher** – абстрактный класс активатора заданий, осуществляющего непосредственный запуск программ в операционной системе и обеспечивающего перехват потока вывода в файл);
10. **CCommunicator** – абстрактный класс высокоуровневой оболочки сокетов для приема и передачи сообщений по сети;
11. **CGlobalParameters** – абстрактный класс для загрузки/сохранения параметров в файл или базу данных;
12. **CControl** – абстрактный управляющий класс (в обязанности класса входит создание, удаление всех объектов, управление ходом работы всех классов);
    - 12.1. **CClusterControl** – наследник CControl для менеджера заданий;
    - 12.2. **CIntegratorControl** – наследник CControl для интегратора кластеров;
13. **CMonitor** – абстрактный класс для мониторинга вычислительных узлов кластера. Класс имеет наследников, реально выполняющих действия по сбору информации. Реализация мониторов для различных операционных систем может отличаться;
14. **CStatistics** – класс, отвечающий для сохранения статистической информации (информация из структур CEventPool и CNodePool) в базу данных;
15. **CScheduler** – абстрактный класс, отвечающий за планирование распределения вычислительных заданий по имеющимся ресурсам (ресурс – это либо кластер в случае интегратора кластеров, либо вычислительная машина в случае менеджера заданий). Конкретные алгоритмы планирования реализуются классами – наследниками.

3.6. Описание рабочих потоков менеджера заданий

В каждый момент времени в процессе менеджера заданий работает не менее 4 потоков. При поступлении команды от интегратора создается еще один поток, обрабатывающий поступающий запрос. Описание рабочих потоков менеджера заданий:

1. Управляющий поток

Основной поток, синхронизирующий работу всех остальных потоков в системе. Управляющий поток первым начинает работу, производит загрузку параметров из конфигурационных файлов, выделяет память под совместно используемые структуры данных, запускает все рабочие потоки. В управляющем потоке выполняется код класса CClusterControl.

2. **Поток планировщика**

Поток в цикле просматривает очередь заданий (CTaskPool) и принимает решение о времени запуска задания и узлах, на которых задание будет запущено. Решение принимается исходя из информации, предоставляемой монитором, и ожидаемом времени выполнения задания. При принятии решения о запуске задания поток планировщика передает указатель за задачу объекту класса CLauncher, выполняющему непосредственный запуск пользовательских заданий. В потоке выполняется код класса CScheduler.

3. **Поток монитора**

Поток монитора собирает информацию о загруженности узлов кластера и записывает ее в объект класса CNodePool, к которому имеет доступ поток планировщика. В настоящее время используется монитор GPM (Grid Performance Monitoring), разрабатываемый проектом «Методы и инструменты Grid» лаборатории «Информационные Технологии» ([6]). Несмотря на то, что GPM разрабатывается, преимущественно, для Grid систем, он вполне применим и для нашего случая. Особенностью данного монитора является необходимость устанавливать сервис на каждый узел кластера. Помимо этого, был разработан собственный монитор, способный получать данные удаленно, без установки ПО на узел.

4. **Поток коммуникатора**

Поток коммуникатора отвечает за прием входящих сетевых соединений от компоненты удаленного доступа и создания клиентских потоков для обработки клиентских запросов.

5. **Клиентские потоки**

Поток (или группа потоков), отвечающий за обработку запросов компоненты удаленного доступа. Создается по одному потоку для каждого принимаемого запроса. По окончании обработки запроса клиентский поток отсылает ответ согласно принятому протоколу общения компонент.

4. Описание интеграции с Microsoft Compute Cluster Server 2003

CCS (Microsoft Compute Cluster Server 2003) предоставляет сразу несколько интерфейсов доступа: графический интерфейс, интерфейс командной строки и программный интерфейс (API – Application Programming Interface). Пользовательское приложение может использовать API через динамическую библиотеку cppapi.dll, содержащуюся в поставляемом с Microsoft Compute Cluster Pack SDK (Software Development Kit – набор средств для разработки ПО). В данной библиотеке содержатся COM-интерфейсы, предоставляющие методы работы с CCS. Информация о предоставляемых библиотекой методах может быть найдена в разделе «Система управления Microsoft Compute Cluster Server 2003». В данном разделе мы рассмотрим как при помощи данного API была произведена интеграция Microsoft CCS с системой Метакластер.

Система управления Метакластер позволяет управлять сразу несколькими кластерами. Система позволяет распределять задания как между кластерами, так и внутри кластера, по узлам. Однако, в том случае, когда на каком-то кластере уже установлена система управления, Метакластер может служить промежуточным звеном между локальной системой и пользователем. В последнем случае распределение заданий по узлам осуществляет система управления, с которой произведена интеграция. Под интеграцией в данном случае мы понимаем предоставление возможности выполнения следующих операций: постановка задач в очередь CCS, мониторинг их состояния, принудительное снятие с выполнения.

Метакластер состоит из трех основных компонент: диспетчер доступа, интегратор кластеров и менеджер кластера (см. Рис. 1). Интегратор кластеров осуществляет единообразное управление несколькими кластерами, для каждого из которых создается специфический менеджер, учитывающий особенности архитектуры кластера, установленной операционной системы, конкретной версии библиотеки MPI. Менеджер ответственен за планирование и запуск заданий на машинах кластера, на котором он функционирует.

В нашем случае необходимо создать менеджер, фактически представляющий оболочку (посредник) для взаимодействия с CCS. Его особенность состоит в том, что он сам не осуществляет планирование и запуск параллельных задач, как это происходит в случае с менеджерами кластеров без предустановленной системы управления сторонних разработчиков. Он лишь передает задачи от интегратора к системе CCS, которая самостоятельно производит планирование и запуск.

Для реализации требуемой функциональности был реализован специальный класс-наследник планировщика CSchedulerCCS, который в отличие от штатного планировщика системы Метакластер отправляет задачи на CCS сразу при их поступлении от диспетчера удаленного доступа. Такой подход выбран потому, что в случае использования Microsoft CCS, последняя система сама принимает решение о моменте запуска и используемых для этого узлах.

Так как CCS не только принимает решение о моменте запуска, но и осуществляет непосредственный запуск, то необходимо было написать наследник класса, отвечающего за запуск задания (CLauncher), такой, чтобы он не выполнял запуск задачи, а просто добавлял ее в очередь CCS. Этот наследник называется CLauncherWindowsCCS.

Класс CLauncherWindowsCCS содержит указатель на COM – интерфейс ICluster, определяющий кластер под управлением Windows CCS, с которым работает Менеджер заданий «Метакластера». Интерфейс создается в соответствии с настройками, содержащимися в конфигурационном файле «Метакластера».

В функции запуска задачи класса CLauncherWindowsCCS создается задание (Job) с помощью функции ICluster::CreateJob. После этого создается задача (Task) вызовом функции ICluster::CreateTask, устанавливаются параметры:

1. **Имя задачи** (ITask::Name) – имя, введенное пользователем при добавлении задачи в интерфейсе «Метакластера»;

2. **Команда** (ITask::CommandLine) – команда, которая будет выполнена CCS. Команда представляет собой строку вида:
   `mpiexec -hosts %CCP_NODES% <путь до исполняемого файла программы> <параметры командной строки>.`
   Данная команда указывает CCS, что исполняемый файл программы – параллельная программа, написанная с использованием MPI и что для запуска ее необходимо использовать стандартную утилиту mpiexec, входящую в состав Microsoft Compute Cluster Pack. Параметр «-hosts» задает список узлов, на которых будет запущена параллельная программа. Список узлов будет выбран CCS исходя из текущей загрузки кластера. Этот список будет записан в переменную окружения CCP_NODES, которая будет подставлена во введенную нами строку при запуске программы на выполнение;

3. **Файл перенаправленного стандартного потока вывода** (ITask::Stdout) – файл, куда будет перенаправлен стандартный поток вывода запускаемой программы. Имя файла задается пользователем при старте программы через интерфейс «Метакластера»;

4. **Рабочая директория** (ITaks::WorkDirectory) – рабочая директория запускаемой программы, совпадает с каталогом, где физически находится файл пользовательской программы;

5. **Минимальное и максимальное число процессоров**, необходимых задаче и заданию (ITask:: MinimumNumberOfProcessors, ITask:: MaximumNumberOfProcessors, IJob:: MinimumNumberOfProcessors и IJob:: MaximumNumberOfProcessors) – совпадают с числом необходимых пользователю процессов, которые он ввел в интерфейсе «Метакластера»;

После установки параметров задача (Task) добавляется в задание (Job) вызовом функции IJob::AddTask(ITask *).

Важно отметить, что в терминологии CCS понятия задания (Job) и задачи (Task) отличаются: одно задание может содержать несколько задач. Но в нашем случае эти понятия можно считать эквивалентными, так как при добавлении заданий через Метакластер все задания содержат только 1 задачу. Впоследствии возможно, что в системе Метакластер будет использован подобный принцип работы, поскольку он позволяет организовать пакетную обработку задач, которая может быть необходима пользователям.

Последний этап добавления задания в CCS – вызов функций ICluster::AddJob(IJob *) с указателем на созданную нами задачу в качестве параметра и вызов функции ICluster::SubmitJob(…), в которую передаются идентификатор задания (IJob::Id), имя пользователя и пароль, под которым будет запущено задание (берется из конфигурационного файла) и ряд других вспомогательных параметров.

Для получения состояния запущенных задач (при запросе диспетчера удаленного доступа) можно воспользоваться свойством ITask::Status. При этом хранить в памяти все запущенные задачи необязательно, достаточно сохранить уникальные идентификаторы заданий, в рамках которых задачи были запущены в CCS, (IJob::get_Id). Получить указатель на задачу можно, вызвав функцию
ICluster::GetTask(<идентификатор задания>, <идентификатор задачи внутри задания>).
Второй параметр в нашем случае всегда равен 1, так как во всех заданиях, запущенных «Метакластером», только 1 задача.

Для отмены задания по команде диспетчера удаленного доступа можно воспользоваться функцией ICluster::CancelTask(<идентификатор задания>, <идентификатор задачи>, <строка с пояснением для лога CCS>).

4.1. Описание пользовательского интерфейса

В настоящее время работа с «Метакластером» возможна через специально разработанный веб-интерфейс. Для работы с системой необходим компьютер, подключенный к сети Интернет и любой браузер, например Microsoft Internet Explorer (рекомендуется), Opera или Firefox.

4.1.1. Вход в систему (аутентификация)

Для входа в систему необходимо набрать в строке ввода адреса выбранного браузера адрес: http://cluster.software.unn.ru.

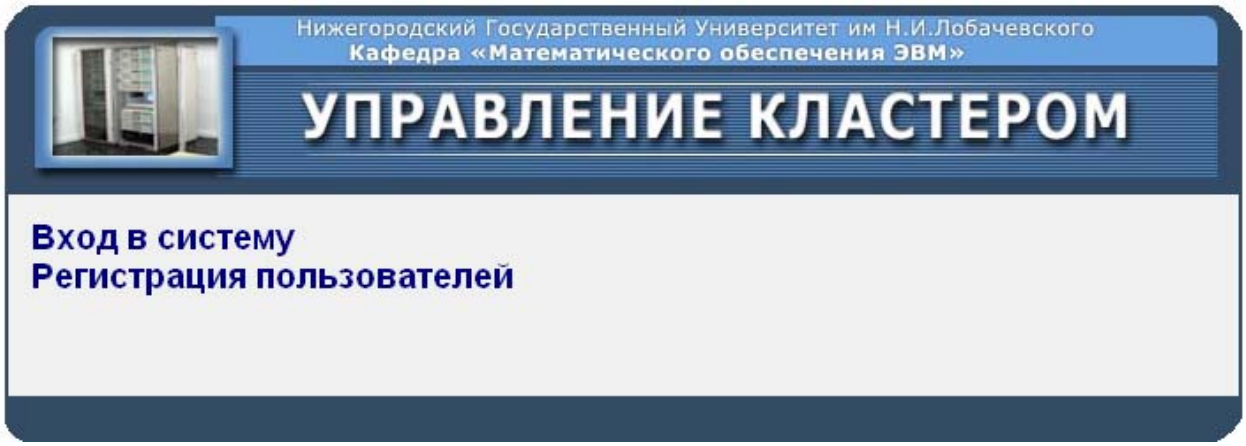

Рис. 3. Вход в систему

Затем необходимо кликнуть мышкой на ссылку *«Вход в систему»*.

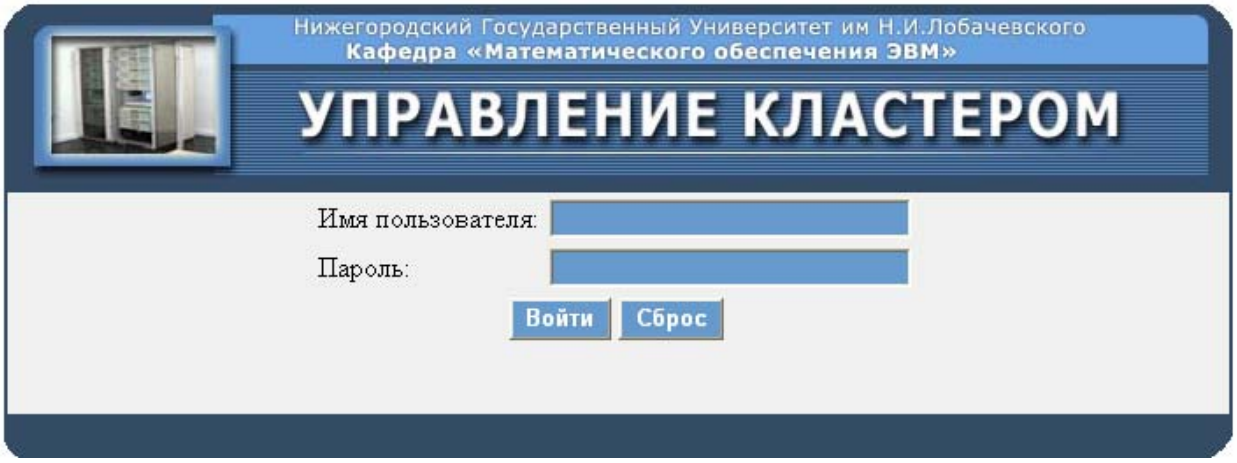

Рис. 4. Аутентификация пользователя

Поля *«***Имя пользователя***»* и *«***Пароль***»* необходимо заполнить данными, полученными при регистрации.

Кнопка *«***Сброс***»* позволяет очистить поля *«***Имя пользователя***»* и *«***Пароль***»*.

4.1.2. Работа с файлами

После успешного прохождения аутентификации в окне браузера появится страница с информацией о хранимых на сервере системы файлах пользователя и исполненных ранее задачах.

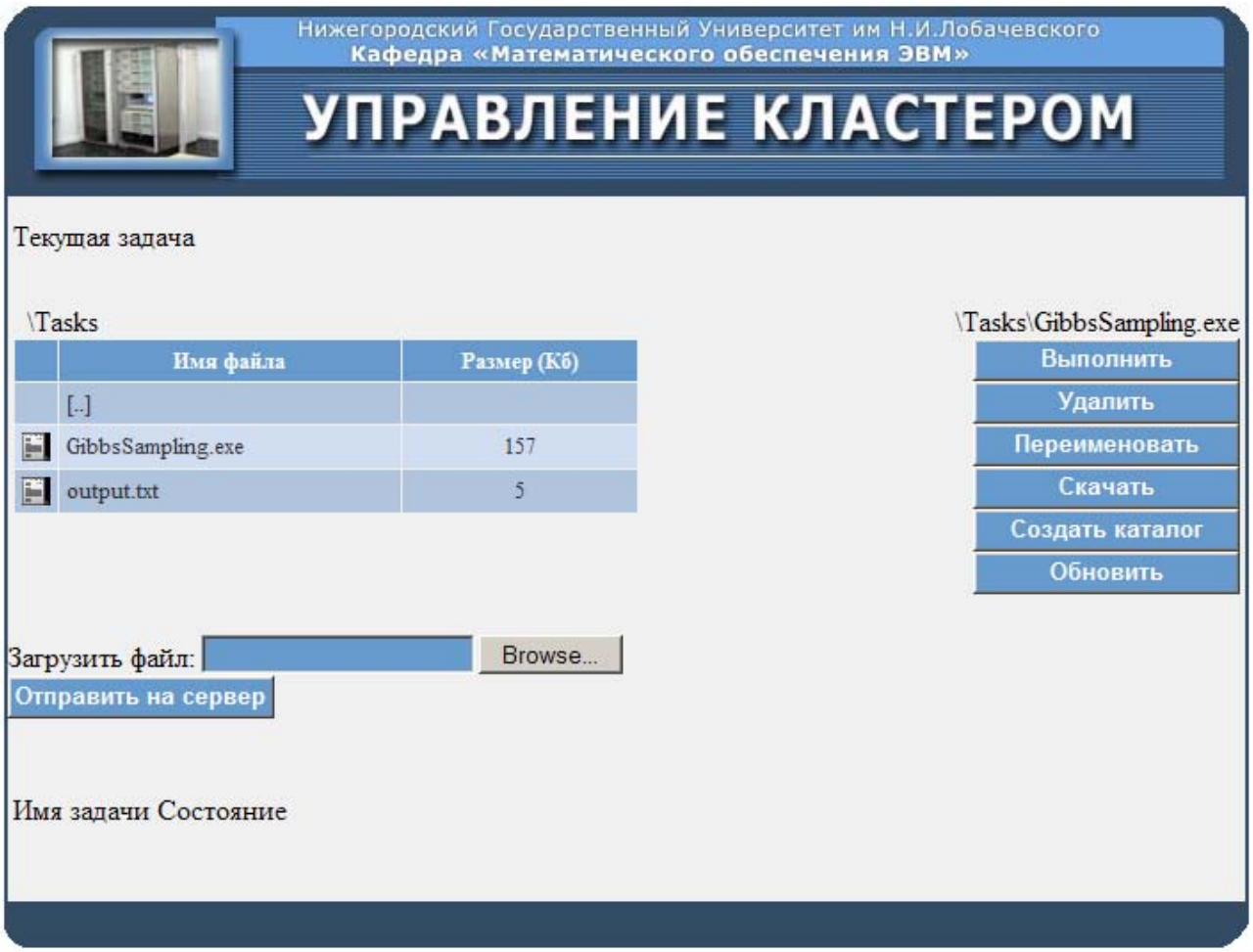

Рис. 5. Информация о хранимых на сервере файлах

В таблице представлена информация о хранимых на сервере системы файлах пользователя, которая включает в себя:

1. **Имя файла** – имя файла на сервере, которое может быть изменено;
2. **Размер** – размер хранимого файла в килобайтах.

Пользователь имеет возможность выполнить следующие операции над файлами:

1. **Загрузить файлы в виртуальный каталог***;*

Для того чтобы загрузить файл на сервер, сначала необходимо открыть рабочий каталог, кликнув на иконку каталога, нажать на кнопку «**Browse**…» и в открывшемся диалоговом окне выбрать необходимый файл. После этого нужно нажать кнопку «**Отправить на сервер**» для окончательной отправки файла.

2. **Удалить файлы из виртуального каталога** (кнопка «**Удалить**»);

3. **Переименовать файлы***;*

Чтобы переименовать файл на сервере необходимо кликнуть на иконку файла и нажать на кнопку «**Переименовать**». Затем нужно ввести новое имя файла в соответствующее поле и нажать кнопку «**Ok**».

4. **Просмотреть содержимое файлов***;*

Для того чтобы просмотреть содержимое файла, необходимо кликнуть мышью на имени интересующего файла и нажать кнопку *«***Скачать***»*. Затем необходимо выбрать директорию для сохранения на локальном компьютере или на внешнем носителе.

4.2. Работа с задачами

4.2.1. Установка задачи в очередь на исполнение

Для того чтобы поставить задачу в очередь на исполнение, необходимо выбрать исполняемый файл в списке «**Файлы**», нажать на кнопку «**Выполнить**».

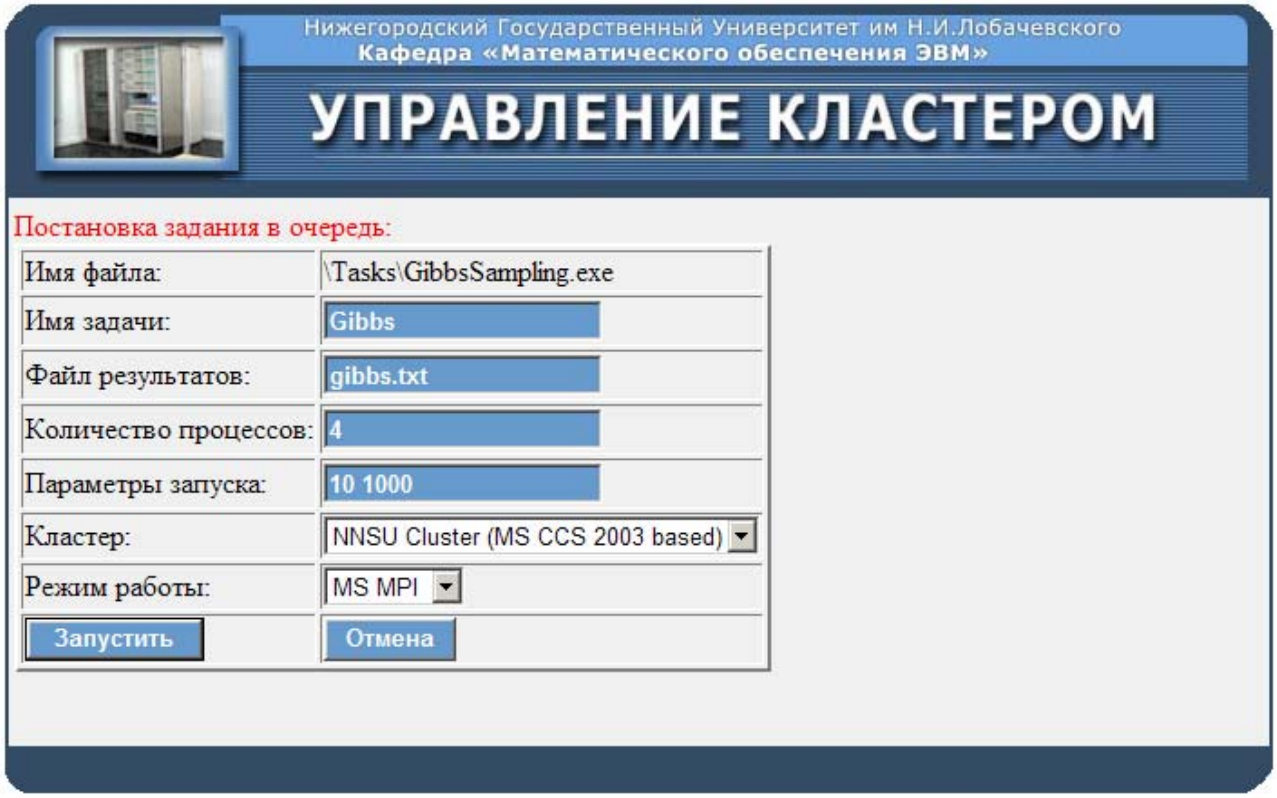

Рис. 6. Установка задачи в очередь на исполнение

В появившемся окне необходимо заполнить следующие поля:

1. **Имя задачи** – произвольное выбранное пользователем имя задачи, идентифицирующее в дальнейшем данную конкретную задачу;
2. **Файл результатов** – имя файла, в который будет перенаправлен стандартный поток вывода (stdout в C++) при исполнении задачи. Если имя файла результатов не указано, то поток вывода теряется при условии, что сама запускаемая программа не генерирует выходной файл;
3. **Количество процессов** – необходимое количество процессов (процессоров) для запуска задачи;
4. **Параметры запуска** – параметры командной строки запускаемой задачи;
5. **Кластер** – кластер, на котором пользователь предпочитает запустить задачу;
6. **Режим работы** – MPI-библиотека, с которой была скомпилирована программа (MS MPI или MPICH2).

Нажмите кнопку «**Запустить**». После этого в нижней части страницы появится новая строка, соответствующая только что запущенной задаче.

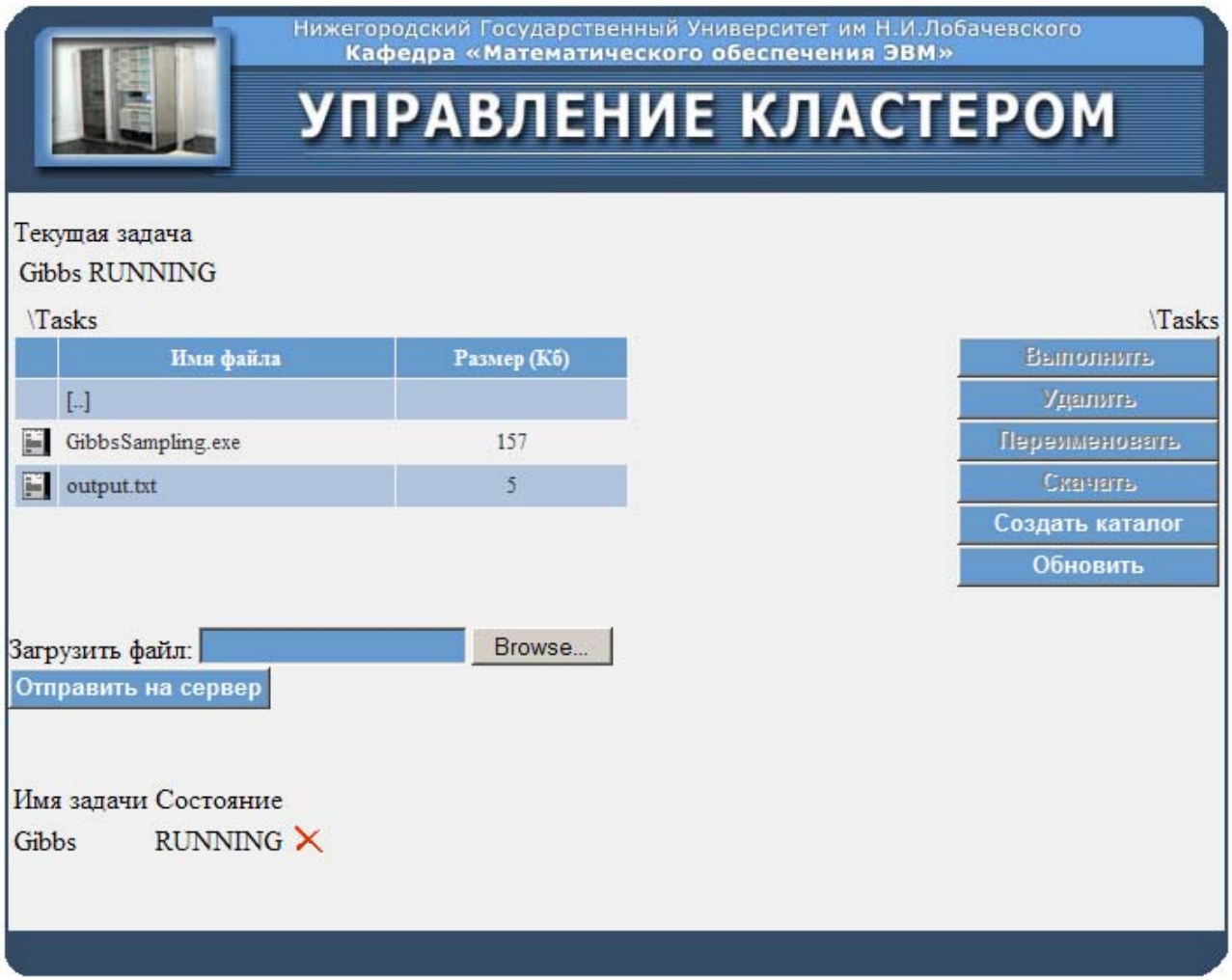

Рис. 7. Изменение диалогового окна после установки задачи в очередь заданий

4.2.2. Информация о созданных задачах

В таблице представлена информация о созданных на сервере системы задачах:

1. **Имя задачи** – введенное пользователем имя задачи при постановке ее в очередь на исполнение;
2. **Состояние** – состояние задачи в очереди:
   - **NOT DISTRIBUTED** – задача находится в общей очереди системы, т. е. задача ещё не передана какому-либо кластреу для выполнения;
   - **WAITING** – задача находится в очереди (распределена на кластер, но ожидает запуска);
   - **RUNNING** – задача выполняется в настоящий момент. Нажатие на иконку, стоящую напротив состояния задачи «RUNNING», позволяет остановить задачу в процессе её работы, в результате чего состояние задачи меняется на **STOPPED**;
   - **FINISHED** – задача выполнена.

5. Заключение

Система управления Метакластер в настоящее время распределяет задания по 3 вычислительным установкам: кластер ННГУ, мини-кластер (оба под управлением Microsoft CCS 2003) и 15 узловый кластер на базе Intel Xeon под управлением Менеджера задач «Метакластера». Вычислительные ресурсы, доступ к которым предоставляется через систему управления Метакластер, используются широким кругом исследовательских проектов как из ННГУ, так и из других университетов. В числе проектов можно упомянуть проект по моделированию работы сердца кафедры ТУДМ факультета ВМК ННГУ, проектами ПараЛаб и

Абсолют, выполняемых на кафедре МО ЭВМ, проектом Финансовая математика лаборатории Информационные технологии. Метакластер использовался для проведения трех школ по параллельным вычислениям Нижегородской лаборатории Информационные технологии при поддержке компании Интел. Все параллельные задачи, выполняемые на кластере в рамках семинаров по параллельным вычислениям, запускались через систему Метакластер.

За время использования система показала свою надежность, расширяемость, удобство в эксплуатации.

6. Литература

1. High-Performance Computing with Windows Compute Cluster Server 2003, http://www.microsoft.com/windowsserver2003/ccs/default.aspx.
2. http://www.platform.com/.
3. http://www.mpi-softtech.com/.
4. Гергель В.П. Теория и практика параллельных вычислений. (http://www.software.unn.ac.ru/ccam/kurs1.htm).
5. Baker M. et al. Cluster Computing White Paper. Final Release, Version 2.0.
6. Боголепов Д., Алехин А. Архитектура системы мониторинга GPM. (http://www.software.unn.ac.ru/grid/gpm).