Основы организации высокопроизводительных вычислений в Windows HPC Server 2008

Ижевск 2010

Министерство образования и науки Российской Федерации

ГОУВПО «Удмуртский государственный университет»

Основы организации высокопроизводительных вычислений в Windows HPC Server 2008

Учебно-методическое пособие

Ижевск 2010

УДК 681.3-181.48 (075)
ББК 32.973.26я7
О25

Рекомендовано к изданию
учебно-методическим советом УдГУ

Составитель М.А. Клочков

Рецензент: старший научный сотрудник
УНЛ «Параллельных вычислений»,
канд. ф.-м. наук, доцент Ю.С. Митрохин

О25 **Основы организации высокопроизводительных вычислений в Windows HPC Server 2008**: учеб.-метод. пособие / сост. М.А. Клочков. Ижевск: Изд-во «Удмуртский университет», 2010. 62 с.

Учебно-методическое пособие содержит задания, которые могут быть использованы для проведения практических занятий по учебным курсам «Операционные системы», «Системное администрирование», «Технология программирования и перепрограммирования», «Архитектура ЭВМ», «Параллельное программирование», «Распределенные системы обработки информации». В помощь студенту большинство примеров представлено в виде рисунков и листинга кода программ.

Пособие может быть рекомендовано студентам, а также преподавателям для проведения практических занятий и организации самостоятельной работы студентов.

УДК 681.3-181.48 (075)
ББК 32.973.26я7

ПРЕДИСЛОВИЕ

В предлагаемом пособии, предназначенном для студентов информационных направлений, изучающих курсы «Операционные системы», «Системное администрирование», «Технология программирования и перепрограммирования», «Архитектура ЭВМ», «Параллельное программирование», «Распределенные системы обработки информации», изложена основная информация, относящаяся к процедуре установки и настройке высокопроизводительной операционной системы, а также запуску и отладке программных приложений, использующих параллельные вычисления.

Параллельные вычисления имеют большое значение при решении прикладных задач. Особенно, если размерность задачи велика, то использование непараллельных вычислительных алгоритмов может потребовать слишком большого объема вычислений, и, следовательно, большого времени работы. Сократить это время можно, используя возможности современной вычислительной техники и программного обеспечения.

Достаточно подробное описание предлагаемых методов организации параллельных вычислений имеется на английском языке. Одной из основных целей создания данного пособия являлась адаптация применения англоязычного материала в учебном процессе.

Пособие состоит из трех разделов. Первый раздел содержит необходимую справочную информацию по подготовке к установке и конфигурированию системы. Основное условие успешного выполнения задания по первой части— это четкое выполнение описанных действий. Во втором разделе содержится описание запуска тестовых процедур, показывающих работоспособность системы в целом, как проверка условий выполнения задания по первому разде-

лу. В третьем разделе необходимо запрограммировать предложенный параллельный алгоритм и провести оценку производительности системы.

Для решения задач из раздела «Задания для самостоятельной работы», теоретического материала, помещенного в методическом пособии, недостаточно. Для их решения рекомендуем пользоваться источниками информации, указанными в конце пособия.

Предлагаемый к изучению материал можно использовать для проведения практических и лабораторных работ, а также при проведении учебных практик. Студентам предлагается установить на учебный компьютер высокопроизводительную операционную систему и запрограммировать различные типовые задачи в формате параллельных вычислений.

Использование данного пособия служит для формирования навыков установки и конфигурирования сложных программных систем, программирования параллельных алгоритмов.

ВВЕДЕНИЕ

Основными способами повышения производительности современных вычислительных систем являются: увеличение числа процессоров, количества ядер на процессор, разработка новых принципов построения коммуникационных сред и топологий с использованием различных сетевых интерфейсов.

В связи с этим для производителей операционных систем и прикладного программного обеспечения возникла необходимость эффективного использования новых возможностей, предоставляемых современными аппаратными архитектурами. Появились операционные системы (ОС), имеющие в своем составе программные инструменты для организации высокопроизводительных параллельных вычислений, как для решения узкоспециализированных исследовательских задач (например, *ОС Linux, http://www.linuxhpc.org/, QNX http://www.qnx.ru/, Barrelfish, http://www.barrelfish.org/*), так и для решения широкого класса прикладных задач с улучшенными возможностями управления вычислительным процессом через привычный и удобный графический интерфейс пользователя. К последнему типу относятся две операционные системы фирмы *Microsoft*—это *Windows Compute Cluster Server 2003*, http://www.microsoft.com/windowsserver2003/ccs/default.mspx?info=EXLINK/ и, рассматриваемая в данном пособии, *Windows High Performance Computing Server 2008*, *http://windowshpc.net/Pages/Default.aspx/*.

КРАТКОЕ ОПИСАНИЕ ТЕХНИЧЕСКИХ ХАРАКТЕРИСТИК И ОСНОВНЫХ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ В ОС *Windows HPC Server 2008*

Microsoft Windows HPC Server 2008 (HPCS)—это высокопроизводительная вычислительная система, предназначенная для работы на многопроцессорных вычислительных системах (ВС) с общей памятью и кластерных архитектурах. Благодаря централизованному интерфейсу управления и развертывания, эту систему можно легко установить и администрировать. В этой ОС используются простые и удобные средства управления, помогающие администраторам ВС работать наиболее эффективно. Также эта система позволяет ускорить выполнение вычислительных расчётов в научно-технических проектах.

HPCS предоставляет простой и удобный интерфейс для организации высокопроизводительных вычислительных систем, повышает эффективность работы пользователей и облегчает решение задачи администрирования кластера, также позволяет достичь максимальной продуктивности работы с высокопроизводительными системами за счет интеграции, а также увеличить скорость вычислений на 30% по сравнению с предыдущей версией ОС—*Windows Compute Cluster Server 2003*.

HPCS содержит программные инструменты поддержки высокопроизводительных вычислений, включающие планировщик заданий, реализацию *MS MPI*, программный менеджер для мониторинга работы вычислительных узлов и управления кластером. Работа *HPCS* основана на 64-х разрядной архитектуре центральных процессоров. На её основе возможна организация кроссплатформенного взаимодействия между высокопроизводительными архитектурами на базе ОС семейства *Windows* и *Linux*. Интерфейс

пользователя построен на основе технологии *Microsoft® System Center 2007* с поддержкой скриптов, написанных для *Windows PowerShell™*. Также *HPCS* поддерживает модульный подход к разработке программного обеспечения, основанный на использовании сервисов со стандартизированными интерфейсами (*SOA*).

При разработке HPCS фирма Microsoft сосредоточила свои усилия на четырёх направлениях:

1. Управление системой

- Разработана новая консоль администратора, включающая реализацию всех аспектов управления вычислительным кластером;
- Имеется возможность перегруппировки администратором вычислительных узлов по категориям и типу выполняемых вычислений;
- Возможно графическое отображение температурной карты для кластера или группы узлов, диаграммы для каждого узла, обзор производительности кластера;
- Имеется поддержка управления процессом вычислений через выполнение скриптов, использующих *Windows PowerShell™*— новая оболочка командной строки, поддерживающая более 130 средств и встроенный язык программирования;
- Имеются расширенные возможности управления многопроцессорными вычислительными узлами;
- Встроенная поддержка возможности обновления программного обеспечения, диагностики системы и аудита работы всего кластера в целом;
- Имеется возможность формирования расширенных отчётов для дополнительного анализа работы вычислительной системы с использованием служб аналитики, например *SQL Server Analysis Services*.

2. *Планировщик заданий*

- Поддерживается интеграция с *WCF (Windows Communication Foundation)*, что позволяет разработчикам приложений *SOA* использовать всю мощь высокопроизводительных вычислений;
- Планировщик заданий позволяет детализировать использование необходимых процессорных ядер и вычислительных узлов;
- ОС обладает улучшенной производительностью для больших кластерных систем;
- Поддерживает информационный обмен с внешними базами данных;
- Поддерживает *Open Grid Forum's HPC Basic Profile* интерфейс.

3. *Работа в локальной сети и поддержка стандарта MPI*

- Поддержка протоколов *RDMA (Remote Direct Memory Access)*—удалённого прямого доступа к памяти, при котором передача данных из памяти одного компьютера в память другого компьютера происходит без участия операционной системы, что позволило значительно повысить производительность выполнения *MPI* приложений;
- Улучшенный *Network Configuration Wizard*;
- Реализация нового стандарта *MS MPI*, поддерживающего работу на многоядерных серверах с общей памятью;

4. *Хранение данных*

- Поддержка протокола *iSCSI (Internet Small Computer System Interface)*— это протокол, который базируется на *TCP/IP* и разработан для установления взаимодействия и управления системами хранения данных, серверами и клиентами;

- Поддержка протокола *Server Message Block v2 (SMB v2)* для удаленного доступа к файлам, принтерам и другим сетевым ресурсам, а также для межпроцессного взаимодействия;
- Улучшенная файловая система.

Приведём общее описание конфигурации локальной сети под управлением HPCS. Рассмотрим основные компоненты, входящие в её состав, и опишем роль каждого из них.

Вычислительные узлы—это компьютеры, объединенные сетью в вычислительный кластер, на которые установлено специальное программное обеспечение, позволяющее пользователям запускать свои задания и контролировать их состояние. Среди узлов кластера выделяется ведущий (Мастер-узел), на который устанавливается управляющее программное обеспечение, которое с одной стороны предоставляет пользователям интерфейс доступа к кластеру, а с другой стороны осуществляет управление вычислительными узлами, посылая на них команды и запросы. Он может использовать интеллектуальную службу каталогов *Active Directory* для обеспечения безопасности всех выполняемых операций.

Узел-посредник *WCF*—это специализированный узел, который выполняет масштабирование кластера и действует как прокси-сервер, облегчающий связь между общедоступными сетевыми клиентами и вычислительными узлами в частных сетях.

Покажем схему типовой конфигурации локальной сети на следующем рисунке.

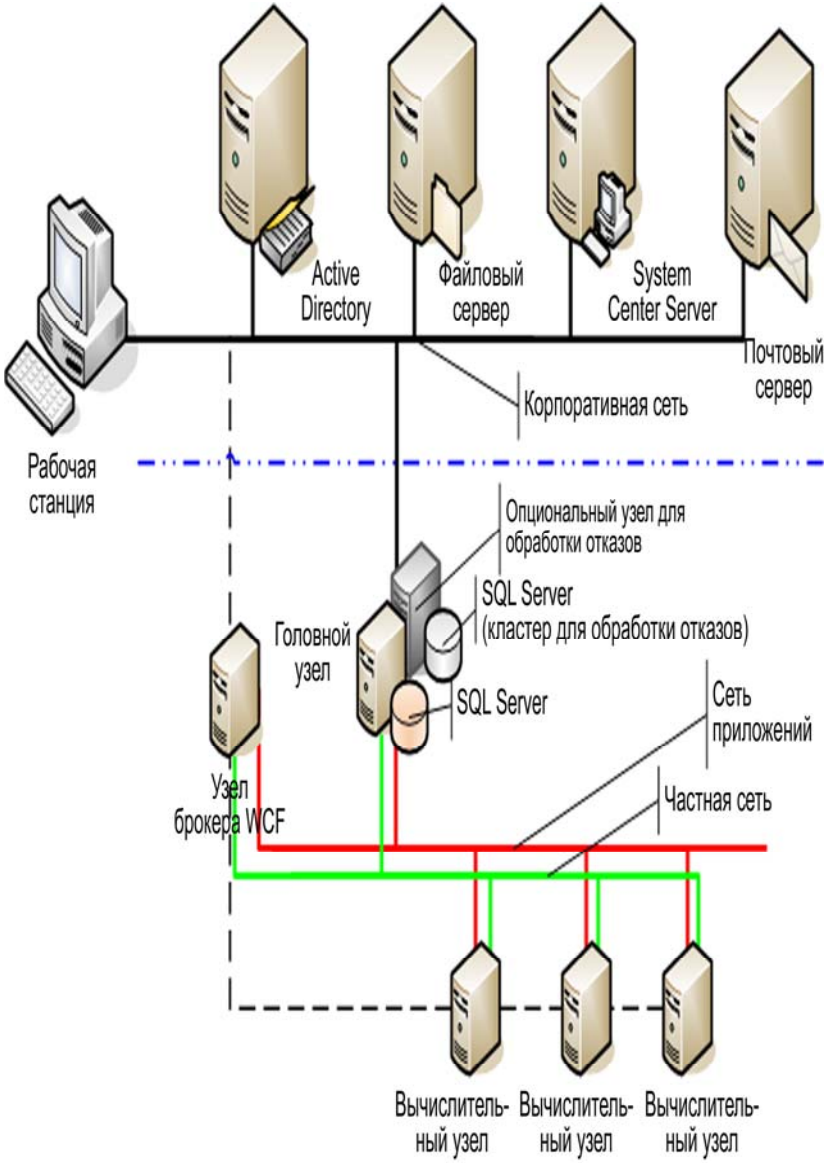

Рис. 1. Схема типовой конфигурации локальной сети

Минимальные технические требования для установки HPCS приведены в следующей таблице:

Таблица 1. Технические требования

Требование	***Windows HPC Server 2008***
Центральный процессор	64-разрядный компьютер с процессором семейства *Intel Pentium* или *Intel Xeon* с поддержкой технологии *Intel Extended Memory 64 (EM64T)*, либо с процессором семейства *AMD Opteron* или *AMD Athlon (AMD64)*, либо другим совместимым процессором. • минимум: 1.4 ГГц; • рекомендованный: 2 ГГц или быстрее
Объем оперативной памяти	• минимум: 512 Мбайт • рекомендованный: 2 Гбайт или больше • максимальный: 128 Гбайт
Поддержка многопроцессорной обработки	Поддерживает до четырех процессоров на сервер.
Привод	*DVD-ROM*
Сетевые адаптеры	На каждом узле должна быть хотя бы одна сетевая плата. Если планируется создание частной сети, то на ведущем узле необходимо, по крайней мере, две сетевых платы (для общедоступной и частной сети), в зависимости от выбранной топологии. Дополнительные сетевые платы на узлах могут потребоваться для подключения к общедоступной сети или для поддержки сети на основе *MPI*.

Опишем требования к программному обеспечению HPCS. *Ведущий и вычислительные узлы* должны работать под управлением *Windows Server 2008 HPC Edition* или другом 64-битной изданием *Windows Server 2008* с дополнительно установленным *Microsoft HPC Pack 2008*.

Управляющие (клиентские) узлы должны работать под управлением следующих ОС:

- *Windows Server 2003 Service Pack 2 (SP2) or Release 2 (R2) (32-bit or x64 versions);*
- *Windows Server 2008 (32-bit or x64 versions);*
- *Windows® XP Professional SP3;*
- *Windows XP Professional, x64 Edition SP2;*
- *Windows Vista® Business, Enterprise, and Ultimate Editions SP1*.

Для организации процесса управления вычислительным процессом на ведущем и вычислительных узлах, также необходимо установить *Microsoft HPC Pack 2008* на управляющий узел.

HPCS поддерживает работу с пятью возможными вариантами организации топологии локальной сети, при этом каждый узел может иметь от одной до трех сетевых карт, а процесс создания требуемого варианта поддерживается специальным помощником *Network Wizard*. Правильный выбор используемой топологии необходим для оптимального функционирования вычислительного кластера.

При описании топологий используются следующие понятия:

- Общедоступная сеть (*Enterprise network*)— корпоративная сеть организации, соединенная с ведущим и (возможно) вычислительными узлами кластера. Через общедоступную сеть пользователи подключаются к ведущему узлу для управления выполнением их заданий. *MPI-*

трафик будет направлен через неё только в том случае, если нет частной или MPI-сети;

- Частная сеть (*Private network*)—выделенная сеть, предназначенная для коммуникации между узлами вычислительного кластера. Эта сеть (если она есть) будет использована для административного трафика (удалённый рабочий стол, установка вычислительных узлов). Кроме того, через частную сеть будет направлен *MPI*-трафик в том случае, если нет специальной *MPI*-сети;
- *MPI*-сеть (Application network)—выделенная сеть (предположительно, наиболее быстрая из трёх перечисленных), через которую идет трафик *MPI*-программ. В случае если программа не использует *MPI*-библиотеки для передачи сообщений по сети, то *MPI*-сеть не будет использоваться.

Перечислим пять различных вариантов сетевых топологий:

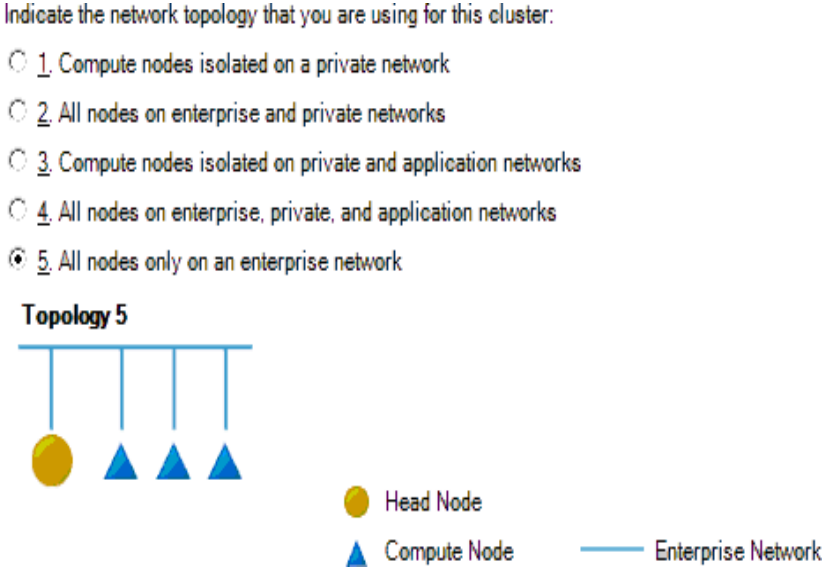

Рис. 2. Варианты сетевых топологий

Вычислительные узлы изолированы в частную сеть (Compute nodes isolated on private networks). В этом случае мастер-узел, имеющий две сетевые карты, обеспечивает трансляцию адресов (network address translation, *NAT*) между вычислительными узлами, каждый из которых по единственной сетевой карте подключен к частной сети (см. рис. 3)

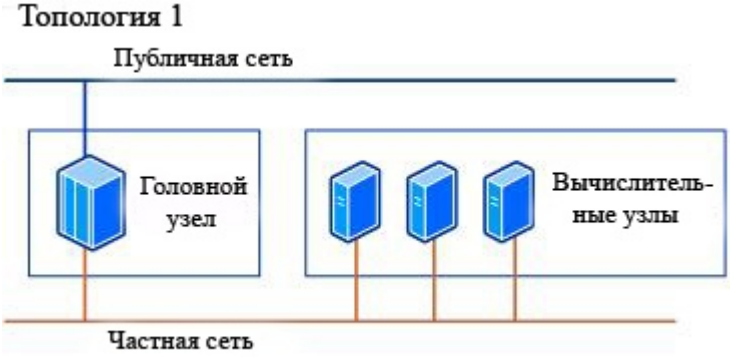

Рис. 3. Первый вариант подключения

Все узлы подключены и к публичной, и к частной сети (All nodes on enterprise and private networks), для этого используются по две сетевые карты на узел (см. рис. 4)

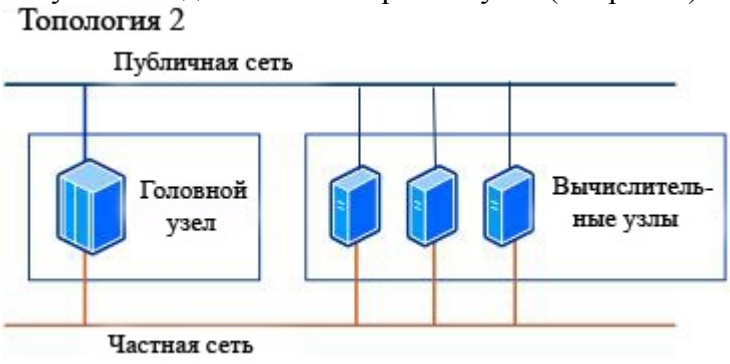

Рис. 4. Второй вариант подключения

Вычислительные узлы изолированы в частной сети и сети передачи по MPI (Compute nodes isolated on private and application networks). В данном случае мастер-узел использует три сетевые карты: для подключения к общедоступной, к частной сети и к сети *MPI*. Мастер-узел осуществляет *NAT* между общедоступной сетью и вычислительными узлами, каждый из которых подключен к частной сети и высокоскоростной сети *MPI* (см. рис. 5)

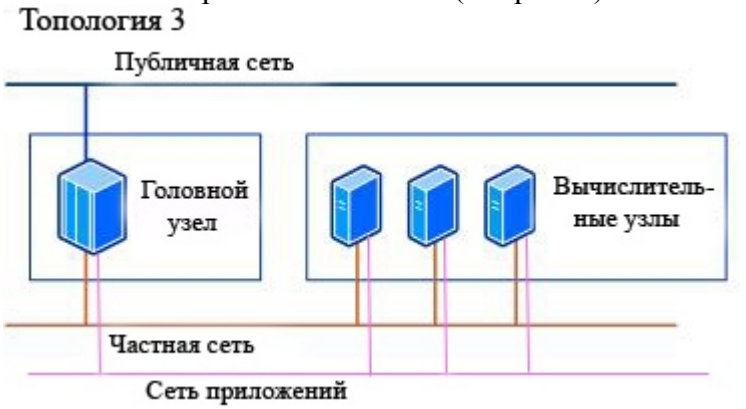

Рис. 5. Третий вариант подключения

Все узлы подключены к публичной и к частной сети, а также к сети MPI (All nodes on enterprise, private and application networks). В таком случае в каждом узле используются все три сетевые карты (см. рис. 6)

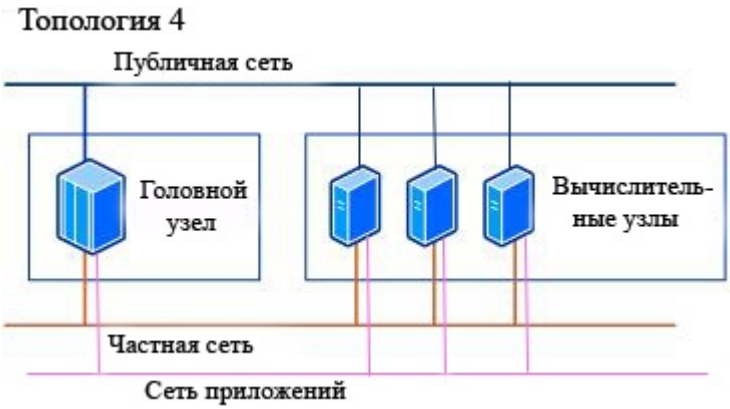

Рис. 6. Четвертый вариант подключения

Все узлы подключены только к общедоступной сети (All nodes only on an enterprise network)

Рис. 7. Пятый вариант подключения

Процесс настройки ВС начинается с установки и настройки ведущего узла. Каждый компьютер, работающий с кластером под управлением *HPCS*, должен принадлежать к одному из трех возможных типов узлов:

- Ведущий узел (*Head node*)—узел, поддерживающий очередь заданий кластера и распределяющий задания по вычислительным узлам;
- Вычислительный узел (*Compute node*)—узел, на котором происходит непосредственное выполнение заданий;

- Клиентский узел (*Client node*)—узел, на котором установлено специальное программное обеспечение для доступа к ведущему узлу для установки заданий в очередь, получения результатов вычислений и управления ходом выполнения заданий.

Если в одном кластере может быть только один ведущий узел, то число вычислительных и клиентских узлов неограниченно. При этом один компьютер может совмещать несколько функций. Так, ведущий узел может быть также вычислительным, а на вычислительном узле могут быть установлены клиентские программы.

Развертывание ВС следует всегда начинать с установки и настройки ведущего узла, состоящих из следующих этапов:

Этап 1. Установка *HPCS* на ведущем узле;

Этап 2. Добавление компьютера в состав какого-либо существующего домена *Active Directory*, либо создание нового домена и назначение ведущего узла контроллером домена;

Этап 3. Установка *Microsoft HPC Pack Server 2008* на ведущем узле.

Предлагается выполнить первое практическое задание по установке Microsoft HPC Pack 2008 на ведущем узле и его настройке. Работу необходимо выполнять поэтапно.

Для *установки Microsoft HPC Pack 2008* на ведущий узел кластера выполните следующие пошаговые действия (в случае, если данный пакет на компьютере еще не установлен):

1) запустите программу установки «**setup.exe**» в папке, где находится дистрибутив *Microsoft HPC Pack 2008*. В открывшемся окне нажмите кнопку «**Next**»;

2) внимательно прочитайте лицензионное соглашение. Выберите пункт «**I accept the terms in the license**

agreement», что дает подтверждение вашего согласия с текстом лицензионного соглашения об использовании системы и нажмите кнопку «**Next**»;

3) в открывшемся окне выберите «**Create a new HPC cluster by creating a head node**» («Создать новый HPC кластер с этим сервером в качестве ведущего узла»). Поставьте флажок в пункт «**Use this head as a compute node**» («Использовать этот узел также и как вычислительный узел"), если Вы хотите использовать ведущий узел также в качестве вычислительного узла. Нажмите кнопку «**Next**» (см. рис. 8)

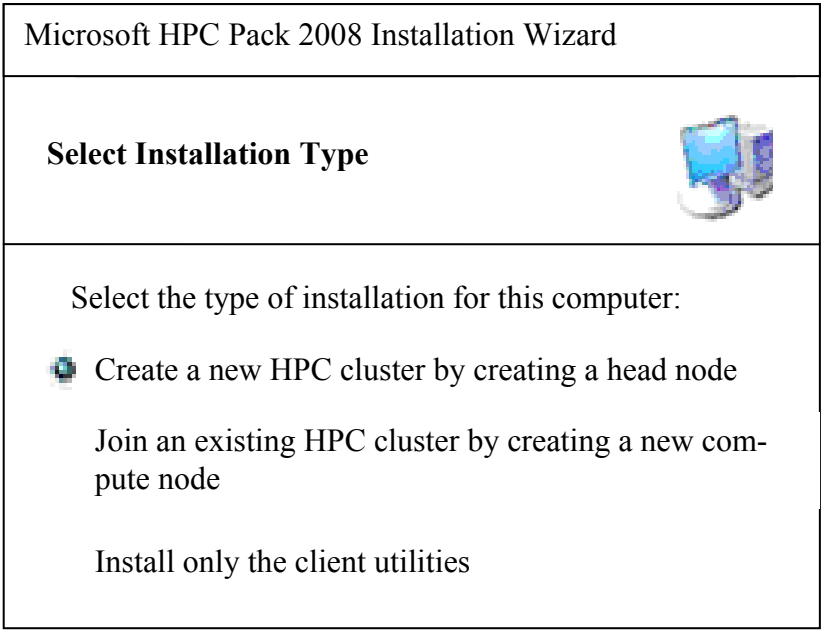

Рис. 8. Список вариантов начальной установки

4) выберите директорию, в которую будет установлен *Microsoft HPC Pack 2008*. Для изменения стандартной ди-

ректории нажмите кнопку «**Browse…**». Нажмите кнопку «**Next**»;

5) последовательно установите все программное обеспечение, указанное в списке открывшегося окна. Для того чтобы начать установку, следует выделить соответствующий пункт списка и нажать на кнопку «**Install**» («Установить»). Установка проводится в соответствии с документацией к устанавливаемой программе. Установку рекомендуется проводить именно в том порядке, который указан в списке. Последним устанавливается *Microsoft HPC Pack 2008*;

6) дождитесь, пока инсталлятор скопирует требуемые файлы;

7) по окончании копирования необходимых файлов нажмите кнопку «**Finish**»,

8) на этом установка программного обеспечения для ведущего узла завершена и можно перейти к его настройке.

Для *настройки* ведущего узла необходимо запустить программу **HPC Cluster Manager** через **Start → Programs → Microsoft HPC Pack → HPC Cluster Manager** и выполнить следующие пошаговые инструкции.

После запуска программы автоматически откроется окно «**To-do List**» («Список работ для выполнения») со списком возможных настроек системы (см. рис. 9):

To-do List

Configuration (Help)

Complete these four configuration steps first:

- Configure your network

  Choose one of five network topologies for your cluster.

- Provide installation credentials

  Specify the user name and password to use for system configuration, when adding compute nodes to the cluster, and when running certain diagnostic tests.

- Configure the naming of new nodes

  Specify the naming convention to use when generating names automatically for new compute nodes.

- Create a node template

  Create a template that defines the steps to follow when configuring a compute node.

Add an operating system image

Create a new image or load an existing image to use with your node templates when deploying compute nodes.

Add or remove users

Add or remove users or administrators for your cluster.

Add compute nodes

Add compute nodes to the cluster by choosing one of three supported options.

Manage drivers

Add device drivers to the operating system images.

Join the Customer Experience Improvement Program

Help Microsoft identify which HPC features to improve.

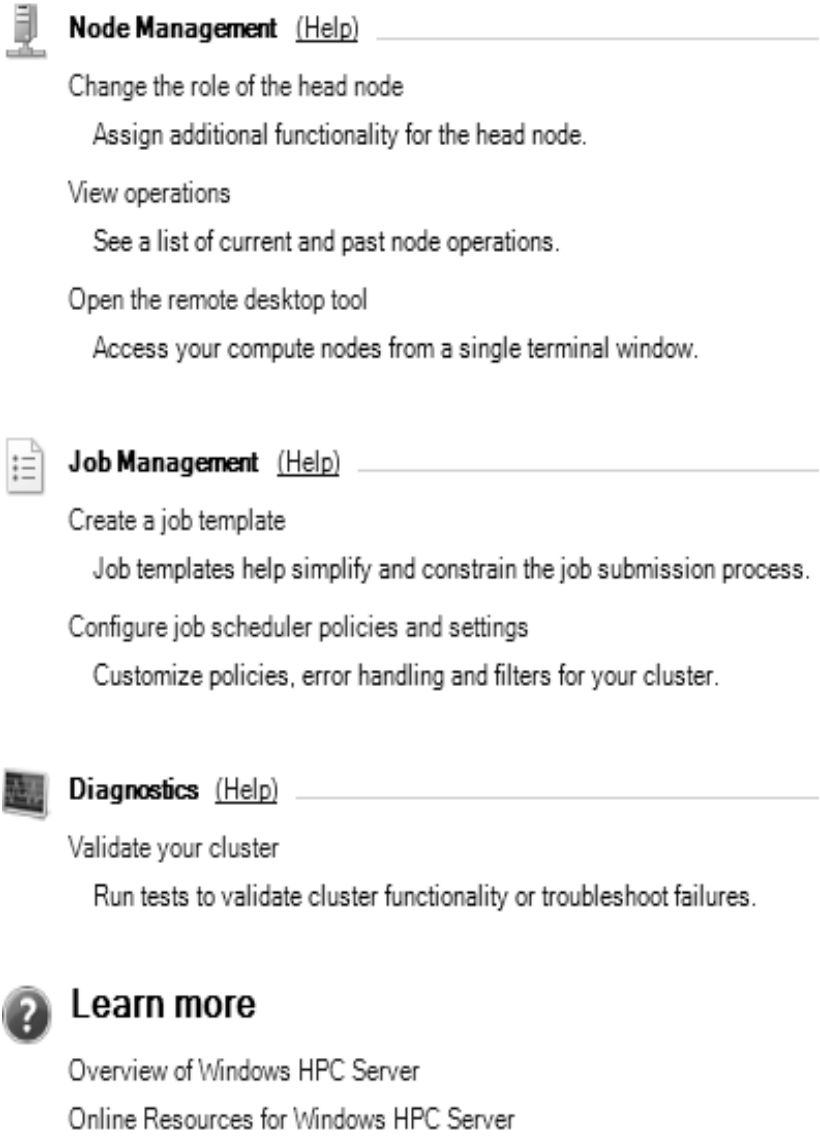

Рис. 9. Список работ для выполнения

1. Настройка сети кластера (пункт «**Configure your network**»)—позволяет задать сетевую топологию кластера для достижения оптимальной производительности;

2. Управление вычислительными узлами (пункт «**Node Management**»)—добавление и удаление из кластера вычислительных узлов;

3. Управление заданиями (кнопка «**Job Management**»)—создание шаблонов заданий и настройка планировщика;

4. Управление процессом диагностики (кнопка «**Diagnostics**»)—автоматически запускает процесс тестирования вычислительного кластера.

Настройка ведущего узла начинается с выбора администратором одной из пяти предопределенных сетевых топологий, описание которых предоставлено выше и зависит от физической организации локальной сети. Правильный выбор топологии позволит оптимизировать работу *MPI*-программ и системы в целом.

Предлагается выполнить следующие действия:

1) если программа **HPC Cluster Manager** еще не открыта на ведущем узле, откройте её, **Start → Programs → Microsoft HPC Pack → HPC Cluster Manager**;

2) в окне «**To-do List**», щелкаем по пункту «**Configure your network**»;

3) на странице **Network Topology Selection**, щелкните по топологии, которую Вы выбрали для кластера, и затем щелкните «**Next**», например можно выбрать второй или пятый пункт (см. рис. 2);

4) на странице **Enterprise Network Adapter Selection** в списке **Network adapter** щелкните по имени сетевого адаптера, который физически связан с общедоступной сетью и затем щелкните «**Next**»;

5) если Вы выбрали топологию номер пять, то перейдите к пункту девять. Иначе, повторите шаг номер четыре для сетевого адаптера, подключенного к частной сети;

6) на странице **Private Network Configuration**, напечатайте статический *IP*-адрес и маску подсети для ведущего узла. Произвольно выберите сетевое обслуживание для этой сети:

a) чтобы предоставить установить доступ к сетевым ресурсам для вычислительных узлов, установите флажок в пункте *Enable network address translation (NAT) on the head node*;

b) чтобы позволить службе *DHCP* автоматически выдавать *IP*-адреса для узлов кластера, установите флажок *Enable DHCP and define a scope*, затем задайте диапазон допустимых *IP*-адресов для области видимости *DHCP*;

c) если настройки шлюза и *IP*-адреса *DNS* сервера не были автоматически обнаружены, задайте их вручную;

7) нажмите «**Next**» после того, как Вы завершите настраивать частную сеть;

8) повторите шаги 4, 6, и 7 для сетевого адаптера, подключенного к общедоступной сети. Щелкните «**Next**» после того, как Вы закончите настраивать общедоступную сеть.

9) на странице **Firewall Setup** укажите настройки межсетевой защиты для узлов:

a) чтобы применить параметры автоматической настройки межсетевой защиты ведущего узла и вычислительных узлов в каждой сети, нажмите «**ON**»;

b) чтобы отключить поддержку межсетевой защиты в сети, нажмите «**OFF**»;

c) если Вы не хотите изменять какие-нибудь параметры настройки межсетевой защиты, нажмите «**Do not manage firewall settings**»;

10) на странице **Review** проверьте свои параметры настройки и щелкните «**Configure**». Если Вы хотите изменить любой из параметров настройки, передвигайтесь к соответствующей странице мастера, щелкая в навигационной области окна или щелкая «**Previous**»;

11) после того, как процесс настройки сети закончен, на странице **Configuration Summary** просмотрите список элементов конфигурации. Если Вы хотите сохранить настройки данной сетевой конфигурации, щелкните по пункту «**Save the configuration report**»;

12) поздравляем, настройка закончена, щелкните «**Finish**».

Опишем необходимую последовательность действий по конфигурированию дополнительных настроек системы. Начнем с *установки инсталляционных сертификатов*. Они необходимы для конфигурации новых вычислительных узлов. Эти сертификаты используются при установке операционных систем, приложений и для добавления узлов в домен под управление службы Active Directory. Кроме того, эти те же самые сертификаты используются для выполнения диагностических тестов на узлах кластера. Последовательность действий такова:

1) в **To-do List**, щелкните **Provide installation credentials**. Появится диалоговое окно **Installation Credentials**;

2) напечатайте имя пользователя, включая название домена (*DOMAIN\User*) и затем введите пароль для учетной записи пользователя домена, которую Вы будете использовать, чтобы устанавливать и настраивать вычислительные узлы и выполнять запуск диагностических тестов;

3) чтобы сохранить указанные сертификаты, нажмите «**OK**».

Далее понадобиться *настроить автоматическую систему создания имён узлов кластера*. *HPCS* имеет механизм автоматической генерации имён для новых узлов. Необхо-

димо описать порядок именования, используя ряд обозначений, которые задаются выбором названия корня и стартового числа, которое будет сопровождать то название. Стартовое число вкладывается в обозначения символа процент (%). Например: %ClusterNode%1000. Если необходимо установить три узла, то после задания обозначения: %ClusterNode-100% эти узлы будут иметь следующие названия: ClusterNode-100, ClusterNode-101, ClusterNode-102. Имя вычислительного узла должно не превышать 15 символов.

Последовательность действий по настройке имён следующая:

1) в **To-do List**, щелкните **Configure the naming of new nodes**. Появится диалоговое окно **Specify Compute Node Naming Series**;

2) задайте шаблон описания имён, который вы хотите использовать. Предварительный просмотр помогает посмотреть результат применения этого шаблона. Примечание: нельзя описать имя вычислительного узла, состоящего только из чисел;

3) для завершения настройки нажмите «**OK**».

Опишем порядок *задания шаблонов для вычислительных узлов*. Шаблоны узла задают порядок конфигурирования и добавления вычислительных узлов к развёртываемому вычислительному кластеру. Через шаблон мы можем установить образ операционной системы, добавлять определенные драйверы и программное обеспечение к вычислительным узлам или просто добавить уже сконфигурированный узел к нашему кластеру. Можно создавать различные типы шаблонов относительно необходимости конфигурирования различных узлов и вариантов функционирования системы.

Возможно создание двух видов шаблонов:

• **С образом операционной системы**—этот тип шаблона включает этап установки операционной системы на вычислительных узлах;

• **Без образа операционной системы**—этот тип шаблона используется, чтобы только добавить предварительно сконфигурированный вычислительный узел к кластеру, или обновить настройки существующих узлов.

Последовательность действий по настройке шаблонов следующая:

1) в **To-do List**, щелкните **Create a node template**. Появляется помощник создания шаблона узла;

2) на странице **Specify Template Name**, введите имя для шаблона и затем щелкните «**Next**»;

3) если Вы будете добавлять вычислительные узлы без предварительно установленной ОС к нашему кластеру, то необходимо:

а) на странице **Select Deployment Type** щелкните **With operating system** и затем щелкаем «**Next**» (см. рис. 10)

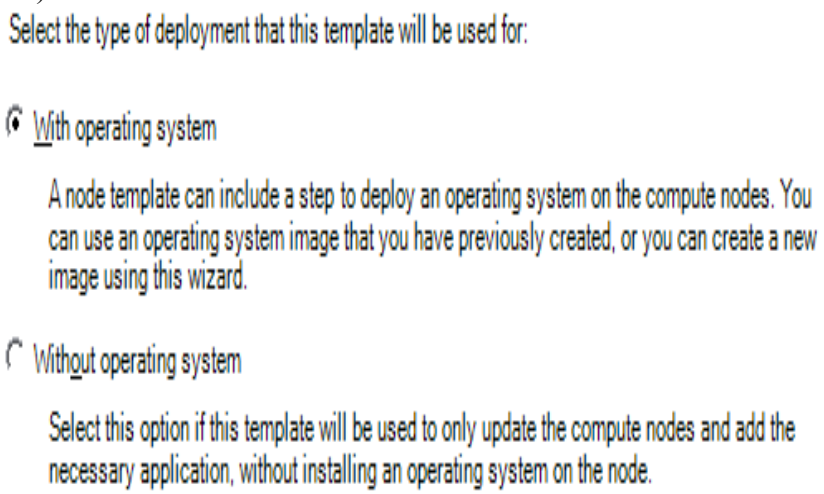

Рис. 10. Выбор типа установки ОС

b) если образ операционной системы, которую мы хотим использовать для установки на вычислительные узлы, уже перечислен в списке **Image Name**, то щелкните по нему и затем переходите к шагу **3.f**. Если же необходимо использовать различные образы операционных систем, то на странице **Select Operating System Image** щелкните «**Add Image**»;

c) в окне **Add Operating System Image** щелкните **Create a new operating system image** и затем напечатайте или через обзор укажите путь к местоположению файла установки ОС *Windows* для одного из 64-битовых выпусков *Windows Server 2008*;

d) задайте имя для нового образа операционной системы, затем нажмите «**OK**»;

e) после того, как образ добавлен, в списке **Image Name** щелкните по нему;

f) нажмите «**Next**» для продолжения;

g) на странице **Specify Local Administrator Password for Compute Node** щелкните **Use a specific password** и затем введите и подтвердите пароль администратора;

h) нажмите «**Next**» для продолжения, затем переходите к шагу 5;

4) если Вы будете добавлять предварительно сконфигурированный вычислительный узел к создаваемому кластеру, то на странице **Select Deployment Type** щелкните **Without operating system**, и далее «**Next**»;

5) на странице **Specify Windows Updates** определите выбор службы установки обновлений *Microsoft Update* или *Windows Server Update Services (WSUS)*, далее «**Next**»;

6) на странице **Review** щелкните «**Create**».

Опишем *порядок регистрации пользователей для работы с вычислительным кластером*:

1) в **To-do List** щелкните **Add or remove users**;

2) для выполнения операции добавления учётной записи пользователя кластера необходимо:

a) в области окна **Actions** щелкните **Add User**. Появляется диалоговое окно **Select Users or Groups**;

b) введите имя пользователя, которого вы хотите добавить, затем щелкните **Check Names**. Для получения дополнительной информации в окне **Select Users or Groups** щелкните **examples**;

c) повторите предыдущий шаг для всех добавляемых пользователей;

d) после того, как вы добавили всех пользователей, нажмите «**OK**»;

3) для выполнения операции добавление администратора кластера необходимо:

a) в области окна **Actions** щелкните **Add Administrator**. Появляется диалоговое окно **Select Users or Groups**;

b) задайте имя администратора, которого вы хотим добавить, затем щелкните **Check Names**. Для получения дополнительной информации в окне **Select Users or Groups** щелкните **examples**;

c) повторите предыдущий шаг для всех добавляемых администраторов;

d) после того, как мы добавили всех необходимых учётных записей администраторов кластера, нажмите «**OK**»;

4) чтобы удалить учётную запись пользователя или администратора, выберите его в списке **Users** и затем щелкните **Remove**.

ОРГАНИЗАЦИЯ ЗАПУСКА ПРОГРАММ НА СОЗДАННОМ ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ ПОД УПРАВЛЕНИЕМ ОС *HPCS 2008*

После того, как мы в предыдущем разделе настроили ведущий узел и добавили вычислительные узлы, необходимо *запустить диагностические тесты*, чтобы проверить функциональные возможности кластера и правильность его настройки. Для этого выполните следующие действия:

1) откройте **HPC Cluster Manager** на ведущем узле **Start → Programs → Microsoft HPC Pack → HPC Cluster Manager**;

2) в **Configuration**, в **Navigation Pane**, щелкните **To-do List**;

3) в **To-do List** щелкните **Validate your cluster** (под **Diagnostics**);

4) в диалоговом окне **Run Diagnostics** выберите опции **Run all functional tests** и **All nodes** и щелкните **Run**;

5) для просмотра результата выполнения диагностических тестов в **Diagnostics**, в **Navigation Pane** щелкните **Test Results** (см. рис. 11);

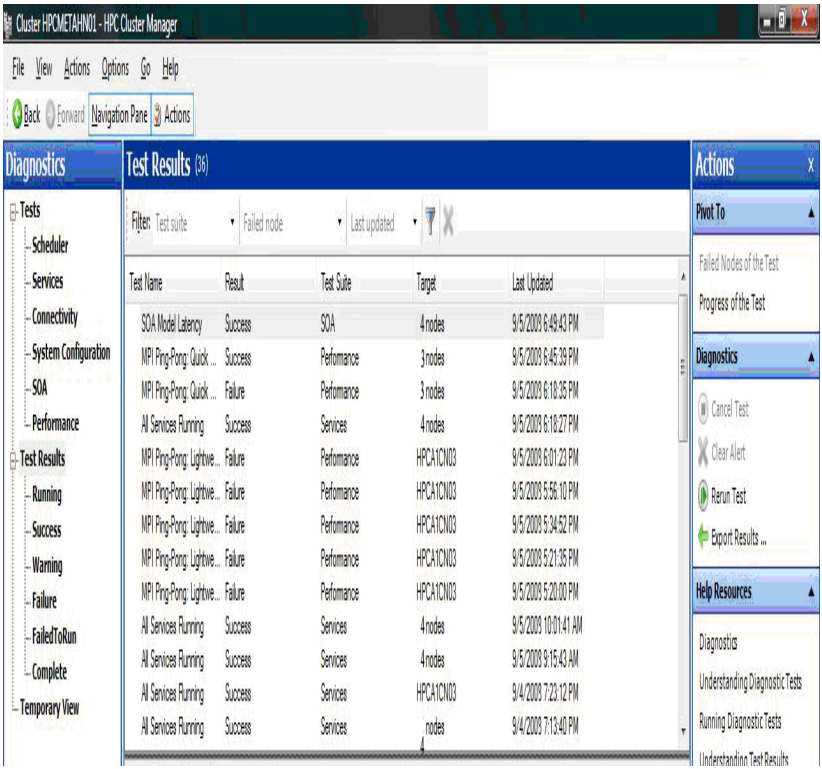

Рис. 11. Результаты тестирования

6) чтобы посмотреть подробную информацию о тесте, дважды щелкните по названию теста. Чтобы расширить информацию в разделе результатов, щелкните стрелкой «вниз» для выбранного раздела.

Для эффективного использования вычислительного ресурса кластера необходимо обеспечить не только непосредственный механизм запуска заданий, но и предоставить среду управления процессом их выполнения с учётом эффективного распределения вычислительных ресурсов.

Принципы работы с заданиями в *HPCS* базируются на трех важных понятиях:

- *Job Submission* (представление задания);
- *Job Scheduling* (планирование задания);
- *Job Execution* (выполнение задания).

Непосредственным запуском задач занимается планировщик, а пользователь может лишь добавить задачу в очередь, так как время ее запуска выбирается системой автоматически в зависимости от того, какие вычислительные ресурсы свободны и какие задания находятся в очереди. Таким образом, *для запуска программы, необходимо выполнить следующие действия*:

1) создать задание с описанием вычислительных ресурсов, необходимых для его выполнения;

2) создать задачу. Задача определяется при помощи той или иной команды, выполнение которой приводит к запуску на кластере последовательных или параллельных программ. Параллельная задача описывается при помощи команды mpiexec.exe с соответствующими параметрами (список узлов для ее запуска, имя параллельной программы, аргументы командной строки программы и др.);

3) добавить задачу к созданному ранее заданию.

Планировщик заданий работает как с последовательными, так и с параллельными задачами. Последовательной называется задача, которая использует ресурсы только одного процессора. Параллельной же называется задача, состоящая из нескольких процессов (или потоков), взаимодействующих друг с другом для достижения одной цели. Как правило, параллельным задачам для эффективной работы требуется сразу несколько процессоров. В этом случае при использовании *MPI* в качестве интерфейса передачи сообщений, процессы параллельной программы могут выполняться на различных узлах кластера. *Microsoft HPC Pack Server 2008* включает собственную реализацию стандарта *MPI2*: библиотеку *Microsoft MPI (MS MPI)*. В случае использования *MS MPI* в качестве интерфейса передачи

сообщений необходимо запускать параллельные задачи с использованием специальной утилиты *mpiexec*, осуществляющей одновременный запуск нескольких экземпляров параллельной программы на выбранных узлах кластера.

Перечислим варианты использования команды *mpiexec*. Команда *job submit* является основным механизмом загрузки и запуска заданий на кластере под управлением *HPCS*. Она используется для передачи параметров управления запуском приложений и запускает на кластере программу *mpiexec.exe*, которая загружает *MS MPI*. Для проведения расчётов можно указать необходимое количество процессоров и перечень узлов. В свою очередь программа *mpiexec.exe* предоставляет полный контроль над выполнением заданий на кластере. Пример записи в командной строке для запуска приложения *MPI* с помощью планировщика заданий выглядит следующим образом:

job submit /numprocessors:8 /runtime:5:0 mpiexec myapp.exe

Эта команда предписывает планировщику заданий запустить приложение *MyApp.exe* на восьми процессорах с ограничением максимального времени выполнения пятью часами. В следующей таблице представлены наиболее распространенные аргументы для запуска команды *mpiexec*:

Таблица 2. Список аргументов команды mpiexec

Аргумент	**Результат**
mpiexec MyApp.exe *mpiexec -n * MyApp.exe*	Процесс *MyApp.exe* будет запущен на всех ядрах, выделенных заданию, либо (в случае использования пакета разработки *HPC Server 2008 SDK*) на каждом ядре локальной системы.
mpiexec -hosts 2 Node1 1 Node2 1 MyApp.exe	Экземпляр процесса *MyApp.exe* будет запущен на

Аргумент	Результат
	каждом из двух указанных вычислительных узлов.
mpiexec -cores 1 MyApp.exe	Процесс *MyApp.exe* будет запущен на каждом узле, выделенном для задания.
mpiexec -n 4 MyApp.exe	При использовании совместно с командой *job submit* процесс *MyApp.exe* будет запущен на четырех ядрах, выделенных для задания. При использовании пакета *HPC Server 2008 SDK* кластер будет эмулироваться на одном компьютере (на котором выполняется программа *mpiexec*) путем запуска четырех процессов *MyApp.exe*.
mpiexec -wdir \\headnode\MyFolder MyApp.exe	Указывает единый общий рабочий каталог для всех процессов *MyApp.exe* в задании.
mpiexec -wdir "%%USERPROFILE%%\My Folder" MyApp.exe	Указывает отдельный рабочий каталог (с пробелами в имени, из-за которых имя папки требуется заключить в кавычки) для каждого процесса *MyApp.exe* в задании. Будет использован домашний каталог пользователя на каждом узле (значение берется из переменной окружения *USERPROFILE*).
*mpiexec –n 1 master : -n * worker*	Запускается мастер-процесс, а на оставшихся ядрах, выделенных для задания, запускается рабочий процесс. При использовании пакета разработки *HPC Server 2008 SDK* используются

Аргумент	Результат
	свободные ядра локальной системы.
mpiexec –gmachinefile nodes.txt –n 1 master : worker	Запускается мастер-процесс и 31 рабочий процесс. В файле *nodes.txt* перечислены четыре сервера, на каждом из которых имеется по восемь ядер.
mpiexec -trace (pt2pt,coll,interconn) -n 2 app1	Отслеживание коллективных вызовов *MPI* и вызовов типа «точка-точка», которые осуществляются двумя процессами приложения, а также внутренних вызовов, которые осуществляются средствами *MS MPI*.
mpiexec -env MPICH_NETMASK 157.59.0.0/255.255.0.0 MyApp.exe	Перенаправление *MPI*-сообщений приложения *MyApp.exe* в подсеть кластера с адресами 157.59.x.x/255.255.0.0.
mpiexec –lines MyApp.exe	Приложение *MyApp.exe* запускается на всех ядрах, выделенных для задания, а ко всей выводимой информации добавляется префикс с рангом источника вывода.

С помощью планировщика заданий можно зарезервировать узлы для выполнения конкретных задач (с указанием учетных данных), а затем загрузить эти задачи непосредственно из среды разработки *Microsoft Visual Studio 2008* с помощью команды *mpiexec*. Эта функция значительно упрощает процесс отладки при разработке приложений. Также можно создавать небольшие отладочные кластеры и работать с ними с помощью *mpiexec* непосредственно в среде разработки *Microsoft Visual Studio*.

Для создания задания используйте *Microsoft HPC Pack 2008 Job Management* (см. рис. 12), выполните следующие действия:

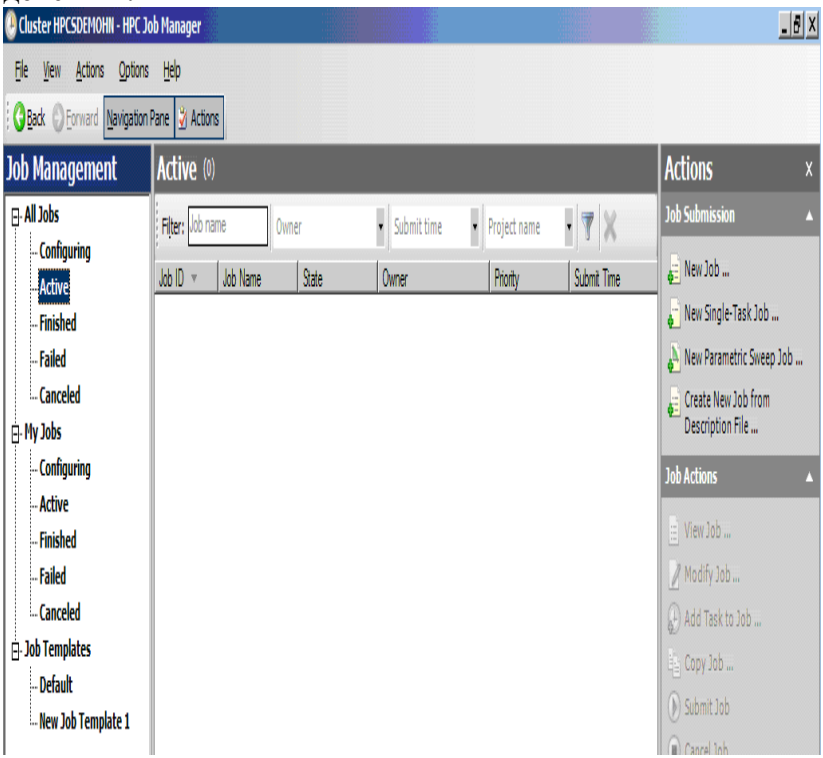

Рис. 12. Окно менеджера заданий

1) в открывшемся окне менеджера заданий выбирите пункт меню **Actions**, щелкните **Create New Job**, далее открывается окошко помощника **Create New Job Wizard**, где Вы определите имя задания, его приоритет, время исполнения, а также максимальное и минимальное число процессоров, необходимых для выполнения (см. рис. 13)

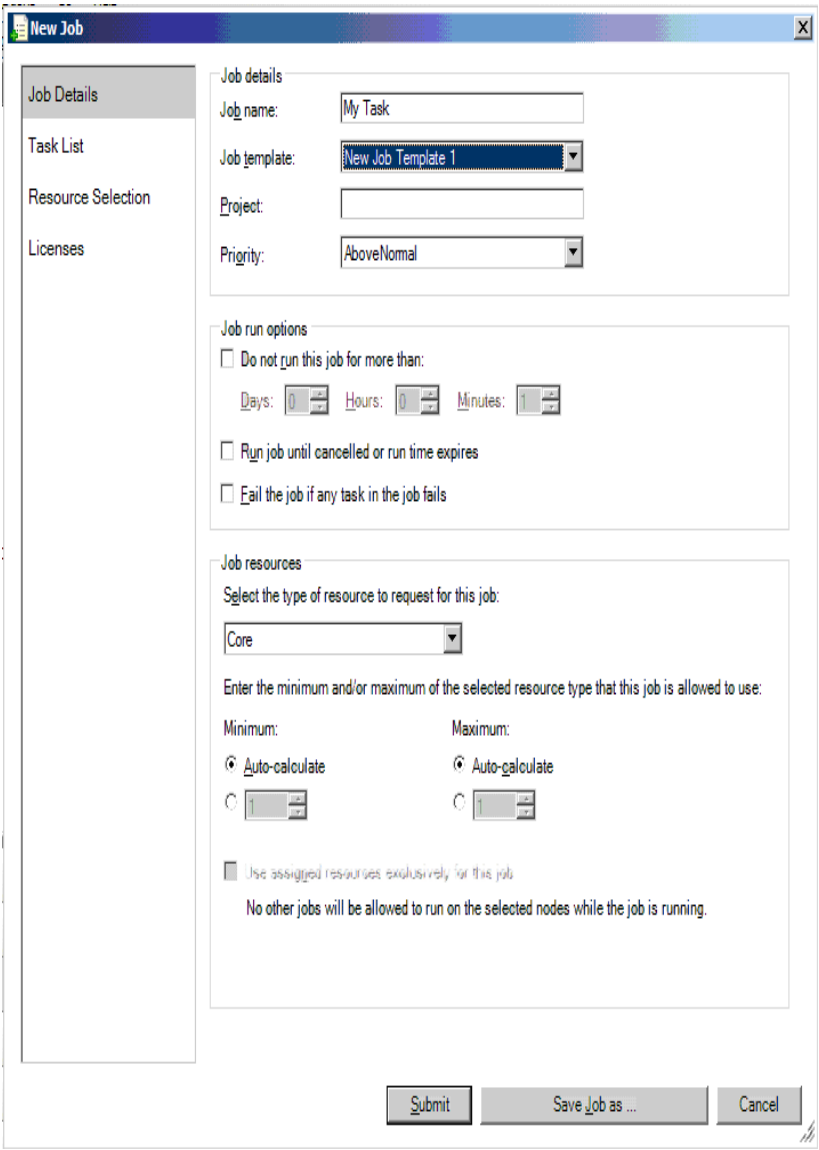

Рис. 13. Создание нового задания

2) далее добавьте новые задачи к заданию, для этого на странице **Create New Job Wizard** выберите **Task List** и

щелкните **Add**, после чего появляется окошко **Task Details and I/O Redirection**, где введите имя задачи (поле «**Task Name**») и команду, которую необходимо выполнить (поле «**Command Line**») (см. рис. 14), далее нажмите «**Save**»

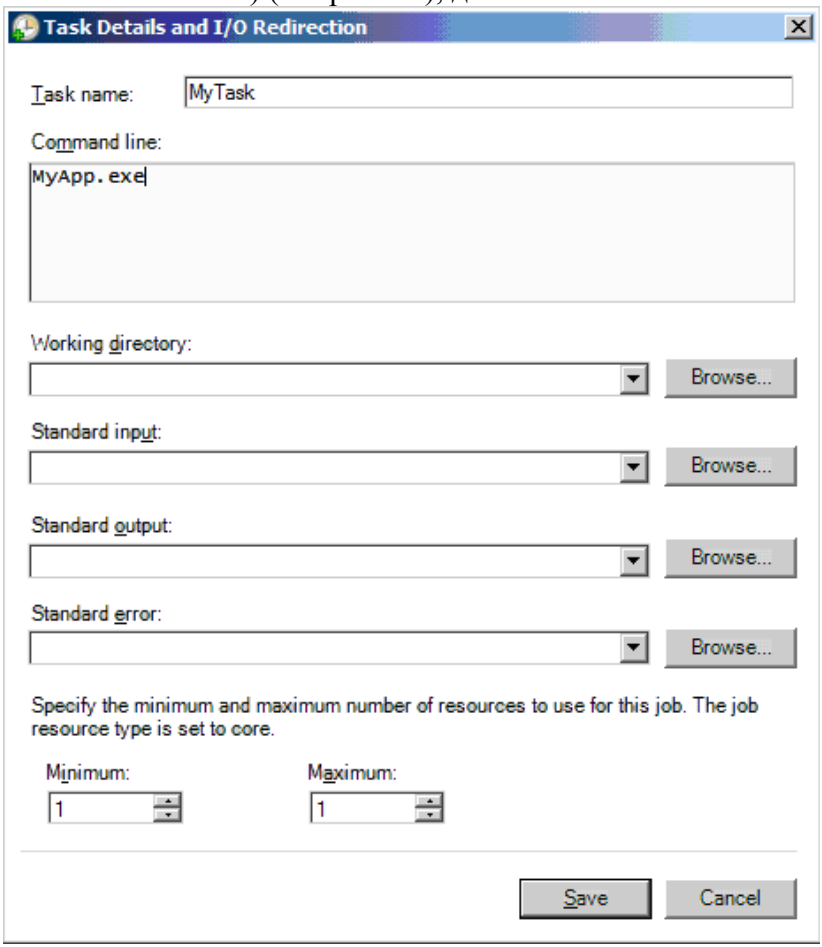

Рисунок 14. Добавление задачи к заданию

3) после того, как задание было полностью описано и сформировано, щелкните **Submit** для его отправки в очередь на выполнение.

Итак, прежде чем работать с планировщиком, необходимо установить соединение с кластером, а именно с ведущим узлом. После завершения работы с планировщиком нужно закрыть соединение. Задание представляет собой пакет задач, поэтому для каждой задачи пользователь должен задать все необходимые параметры (устанавливая соответствующие атрибуты интерфейса) и поместить их в задание, после чего нужно отправить задание в очередь на выполнение.

Для компиляции параллельных программ, работающих в среде MS MPI, необходимо установить SDK (Software Development Kit)—набор интерфейсов и библиотек для вызова *MPI*-функций.

Скачать *Microsoft HPC Pack 2008 SDK* можно перейдя по ссылке: http://www.microsoft.com/downloads/details.aspx?familyid=12887DA1-9410-4A59-B903-693116BFD30E&displaylang=en, после чего необходимо установить данную программу.

ВЫПОЛНЕНИЕ И ОТЛАДКА ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ ДЛЯ ВЫЧИСЛЕНИЯ ЧИСЛА ***π*** НА ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ ПОД УПРАВЛЕНИЕМ ОС *Windows HPC Server 2008*

Предлагается выполнить следующее задание по программированию и запуску параллельной программы на сконфигурированном кластере. Оригинал описания последовательности необходимых этапов по реализации вычислений находиться по адресу http://www.windowshpc.net/Resources/Programs/HPCLAB.zip.

Для нахождения числа ***π*** используется численный метод вычисления определённого интеграла

$$\int_0^1 \frac{4}{1+x^2}\,dx = \pi\,,$$

который заключается в разбиении отрезка [0,1] на очень маленькие равные отрезки, являющиеся основанием прямоугольников с высотой равной значению $f(x) = 4/(1+x^2)$ в точке середины отрезка, это, так называемый, метод средних прямоугольников. Таким образом, суммируя площади получаемых прямоугольников, мы получаем число, стремящееся в пределе к значению величины числа ***π*** при большем количестве разбиений исходного отрезка. Процесс построения разбиения отрезка [0,1] показан на следующем рисунке

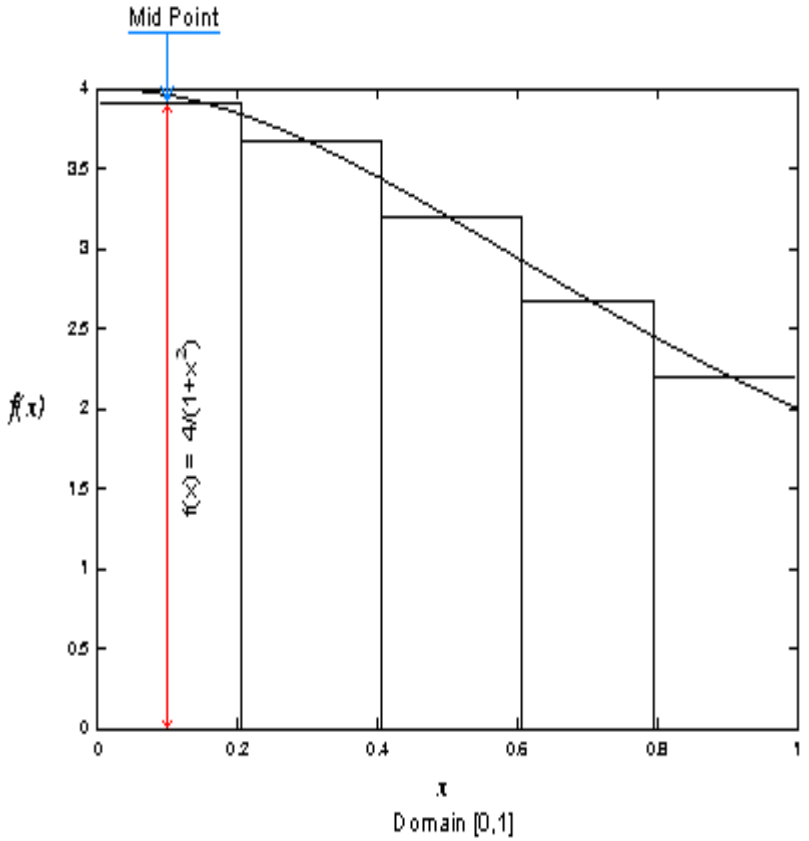

Рис. 15. Разбиение отрезка [0,1]

Пример алгоритма вычислений на псевдокоде может выглядеть следующим образом:

```
//количество интервалов разбиения, например 5, как на
//рисунке
NumIntervals = n

//вычисление ширины интервала, например 0.2, как на
//рисунке
IntervalWidth = 1.0 / NumIntervals

//вычисление интеграла по средней точке
//f(x) = 4/(1 + x^2)
```

```
For (Interval = 1 to NumIntervals)
Begin
IntervalMidPoint = (Interval - 0.5) * IntervalWidth
IntervalLength = IntervalLength + (4.0 / (1.0 + Inter-
   valMidPoint^2)) //заданная функция
End

//вычисление площади всех прямоугольников
Area = IntervalWidth * IntervalLength

Print ("Approximation of PI is ", Area)
```

Опишем необходимую последовательность шагов для создания и запуска параллельной версии алгоритма нахождения числа π.

Для большей наглядности воспользуемся программным кодом на языке C++ последовательной версии программы для решения нашей задачи, как основой для написания параллельной версии, а именно:

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

void main(int argc, char *argv[])
{
   int     NumIntervals = 0;   //число интервалов
   разбиения отрезка [0,1] для F(x)= 4 / (1 + x*x)
   double  IntervalWidth= 0.0;
   //ширина интервала
   double IntervalLength  = 0.0;
   //длина интервала
   double  IntrvlMidPoint      = 0.0;
   //координата средней точки
   int            Interval           = 0;
   //количество интервалов
   int            done               = 0;   //флажок
   double  MyPI               = 0.0;
   //результат вычислений
   double  ReferencePI        =
   3.14159265358979323846264338327950288419716939937510;
   //точное значение числа Pi

   while (!done) //цикл пока done == 0
   {
```

```
      IntervalLength = 0.0;
      printf("\nEnter the number of intervals: (0 quits)
    ");fflush(stdout);
      scanf_s("%d",&NumIntervals);

      if (NumIntervals == 0)
            done = 1;
    //условие выхода из цикла NumIntervals = 0
      else
      {
            //approximate the value of PI
            IntervalWidth   = 1.0 / (double)
    NumIntervals;
            for (Interval = 1; Interval <=
    NumIntervals; Interval++){
                  IntrvlMidPoint = IntervalWidth *
    ((double)Interval - 0.5);
                  IntervalLength += (4.0 / (1.0 +
    IntrvlMidPoint*IntrvlMidPoint));
            }

            MyPI = IntervalWidth * IntervalLength;

      }
    }
}
```

Добавьте в код программы библиотеку «**mpi.h**» для подключения возможности вызова функций *MS MPI*:

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <mpi.h>
```

Добавьте следующие выделенные цветом строки, в которых описаны дополнительные переменные

```
int         NumIntervals        = 0;
double      IntervalWidth       = 0.0;
double      IntervalLength      = 0.0;
double      IntrvlMidPoint      = 0.0;
int         Interval            = 0;
int         done                = 0;
double      MyPI                = 0.0;
```

```
   double     ReferencePI         =
3.14159265358979323846264 3;
   //все параллельные процессы сохраняют результаты
   //вычисления числа Pi с использованием следующих
   //переменных
   double PI;
   //переменная для хранения имени процессора, на котором
   //будут проводиться вычисления, константа
   //MPI_MAX_PROCESSOR_NAME описана в MPI.H
   char processor_name[MPI_MAX_PROCESSOR_NAME];
   //массив всех имён процессоров
   char (*all_proc_names)[MPI_MAX_PROCESSOR_NAME];
   //количество процессов
   int  numprocs;
   //ранг процесса
   int  MyID;
   //длина имени процессора MPI
   int    namelen;
   //переменная цикла
   int    proc = 0;
```

Далее добавьте в листинг функцию *MPI_Init()* для инициализации среды выполнения вызовов функций *MS MPI*

```
   double  PI;
   char    processor_name[MPI_MAX_PROCESSOR_NAME];
   char (*all_proc_names)[MPI_MAX_PROCESSOR_NAME];
   int   numprocs;
   int   MyID;
   int   namelen
   int   proc = 0;

   MPI_Init(&argc,&argv);
```

Определим число процессов в группе, связанной с коммуникатором *MPI*, через функцию *MPI_Comm_size()*

```
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
```

Определим ранг запущенного процесса через вызов функции *MPI_Comm_rank()*, каждый процесс имеет свой номер в диапазоне от нуля до *numprocs - 1*

```
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
```

```
MPI_Comm_rank(MPI_COMM_WORLD,&MyID);
```

Используя функцию *MPI_Get_processor_name()* можно получить имя процессора, на котором проводятся расчеты и запуск параллельных процессов, результат возвращенного значения имени совпадает с результатами выполнения команды *hostname* или *host* в командной строке

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&MyID);
MPI_Get_processor_name(processor_name,&namelen);
```

Сохраните в массиве все имена используемых процессоров с помощью функции *MPI_Gather()*. Все рабочие процессоры должны будут выполнить эту функцию, но только один из них, с указанным номером внутри параметров вызова, получит результат

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&MyID);
MPI_Get_processor_name(processor_name,&namelen);
all_proc_names = malloc(numprocs *
    MPI_MAX_PROCESSOR_NAME);
MPI_Gather(processor_name,
           MPI_MAX_PROCESSOR_NAME,
           MPI_CHAR,
           all_proc_names,
           MPI_MAX_PROCESSOR_NAME,
           MPI_CHAR,
           0,
           MPI_COMM_WORLD);
```

Добавьте фрагмент для печати сформированного списка рабочих процессоров

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&MyID);
MPI_Get_processor_name(processor_name,&namelen);
```

```
all_proc_names = malloc(numprocs *
    MPI_MAX_PROCESSOR_NAME);
MPI_Gather(processor_name,
           MPI_MAX_PROCESSOR_NAME,
           MPI_CHAR,
           all_proc_names,
           MPI_MAX_PROCESSOR_NAME,
           MPI_CHAR,
           0,
           MPI_COMM_WORLD);
for (proc=0; proc < numprocs; ++proc)
printf("Process %d on %s\n", proc,
    all_proc_names[proc]);
```

Для главного процесса добавьте фрагмент для ввода количества интервалов разбиения отрезка [0,1]

```
if (MyID == 0){
    printf("\nEnter the number of intervals: (0
    quits) "); fflush(stdout);
    scanf_s("%d",&NumIntervals);
}
```

Далее необходимо всем рабочим процессам разослать введенное с клавиатуры значение величины *NumIntervals*

```
if (MyID == 0){
    printf("\nEnter the number of intervals: (0
    quits) ");fflush(stdout);
    scanf_s("%d",&NumIntervals);
}
MPI_Bcast(&NumIntervals, 1, MPI_INT, 0,
          MPI_COMM_WORLD);
```

Например, можно распределить вычисления между параллельными процессами в соответствии со следующим рисунком

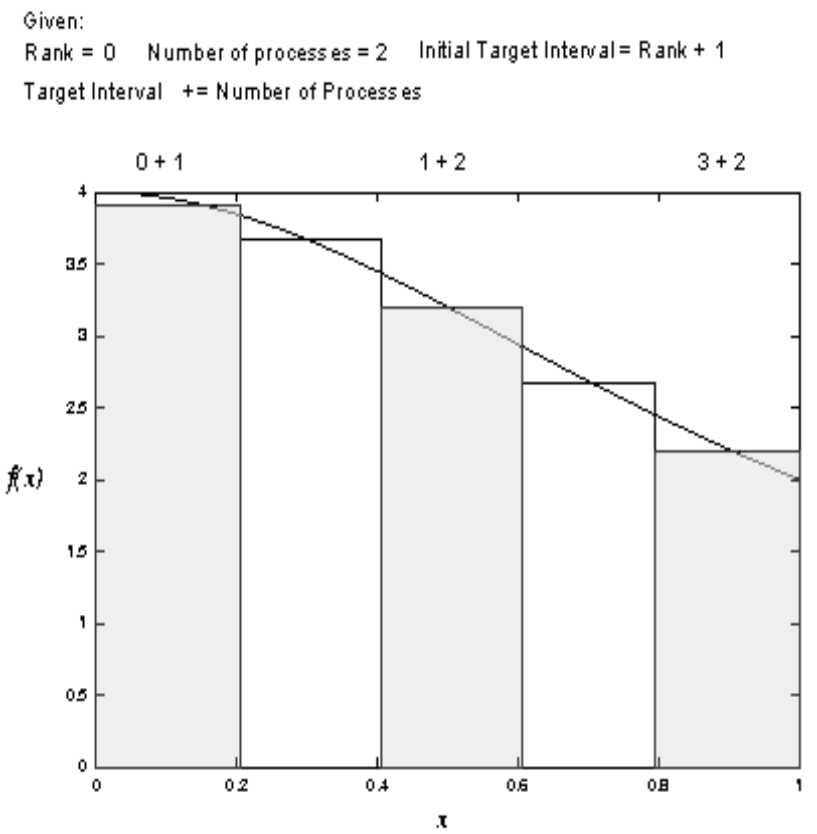

Рис. 16. График распределения вычислений

Всего используются два процесса: главный и вспомогательный. Здесь главный процесс ранга ноль будет вычислять значение площадей закрашенных прямоугольников с нечётными номерами 1, 3 и 5, а вспомогательный процесс ранга один будет рассчитывать прямоугольники с чётными номерами 2 и 4. Измените общий цикл вычисления определённого интеграла в последовательном варианте программы следующим образом

```
IntervalWidth   = 1.0 / (double) NumIntervals;
for (Interval = MyID + 1; Interval <=
    NumIntervals; Interval += numprocs)
```

```
{
  IntrvlMidPoint = IntervalWidth *
    ((double)Interval - 0.5);
  IntervalLength += (4.0 / (1.0 +
    IntrvlMidPoint*IntrvlMidPoint));
}
```

Итак, каждый процесс выполнит свою часть вычислительной работы и результаты вычислений вернёт главному процессу

```
for (Interval = MyID + 1; Interval <=
    NumIntervals; Interval += numprocs)
{
  IntrvlMidPoint = IntervalWidth *
    ((double)Interval - 0.5);
  IntervalLength += (4.0 / (1.0 +
    IntrvlMidPoint*IntrvlMidPoint));
}

MyPI = IntervalWidth * IntervalLength;

MPI_Reduce(&MyPI, &PI, 1, MPI_DOUBLE, MPI_SUM,
           0, MPI_COMM_WORLD);
```

Остаётся добавить фрагмент печати результата проведённых вычислений на экран

```
MPI_Reduce(&MyPI, &PI, 1, MPI_DOUBLE, MPI_SUM,
           0, MPI_COMM_WORLD);
if (MyID == 0) {
  printf("PI is approximately %.16f, Error is
    %.16f\n", PI, fabs(PI - ReferencePI));
}
```

В заключении, не забудьте корректно завершить запуск параллельной программы через вызов *MPI_Finalize()*

```
if (MyID == 0) {
  Printf("PI is approximately %.16f, Error is
    %.16f\n", PI, fabs(PI - ReferencePI));
  }
}}
MPI_Finalize();
} //конец программы
```

Итак, полный листинг параллельной версии программы вычисления числа π должен выглядеть следующим образом

```
   #include <stdlib.h>
   #include <stdio.h>
   #include <math.h>
   #include <mpi.h>

   void main( int argc, char *argv[] )
   {
   int NumIntervals      = 0 ;
   double    IntervalWidth     = 0.0 ;
   double  IntervalLength  = 0.0 ;
   double    IntrvlMidPoint    = 0.0 ;
   int       Interval          = 0 ;
   int       done              = 0 ;
   double    MyPI              = 0.0 ;
   double    ReferencePI       =
3.14159265358979323846264338 ;
   double PI ;
   char      processor_name[MPI_MAX_PROCESSOR_NAME]
;
   char
       (*all_proc_names)[MPI_MAX_PROCESSOR_NAME] ;
   int       numprocs ;
   int       MyID ;
   int       namelen ;
   int       proc = 0 ;

   MPI_Init( &argc, &argv ) ;
   MPI_Comm_size( MPI_COMM_WORLD, &numprocs ) ;
   MPI_Comm_rank( MPI_COMM_WORLD, &MyID) ;
   MPI_Get_processor_name( processor_name,
&namelen) ;

   all_proc_names = malloc( numprocs *
MPI_MAX_PROCESSOR_NAME ) ;

   MPI_Gather( processor_name,
               MPI_MAX_PROCESSOR_NAME, MPI_CHAR,
               all_proc_names,
```

```
                MPI_MAX_PROCESSOR_NAME, MPI_CHAR, 0,
                MPI_COMM_WORLD ) ;
   for ( proc = 0; proc < numprocs; ++proc )
       printf("Process %d on %s\n", proc,
all_proc_names[proc] ) ;

   while ( !done )
   {
       IntervalLength = 0.0 ;
       if (MyID == 0){
             printf( "\nEnter the number of
intervals: (0 quits) " ) ; fflush( stdout ) ;
             scanf_s( "%d", &NumIntervals ) ;
       }

       MPI_Bcast( &NumIntervals, 1, MPI_INT, 0,
                  MPI_COMM_WORLD ) ;

       if (NumIntervals == 0) done = 1;
       else
       {
             IntervalWidth    = 1.0 / (double)
NumIntervals ;

             for (Interval = MyID + 1; Interval <=
NumIntervals ; Interval += numprocs ){
                   IntrvlMidPoint = IntervalWidth *
((double)Interval - 0.5) ;
                   IntervalLength += (4.0 / (1.0 +
IntrvlMidPoint * IntrvlMidPoint )) ;
             }

             MyPI = IntervalWidth * IntervalLength;

             MPI_Reduce( &MyPI, &PI, 1, MPI_DOUBLE,
MPI_SUM, 0, MPI_COMM_WORLD ) ;

             if (MyID == 0){
                   printf("PI is approximately
%.16f, Error is %.16f\n", PI, fabs(PI -
ReferencePI));
```

```
            }
        }
    } //end while
    MPI_Finalize();
    } //end program
```

После набора данного программного кода в редакторе Microsoft Visual Studio, выполните компиляцию и сборку программы **Build → Build Solution (F7)**. Если появятся ошибки, исправьте их самостоятельно, либо обратитесь за консультацией к преподавателю. Далее, для выполнения проекта необходимо загрузить командную строку **Start → Run**

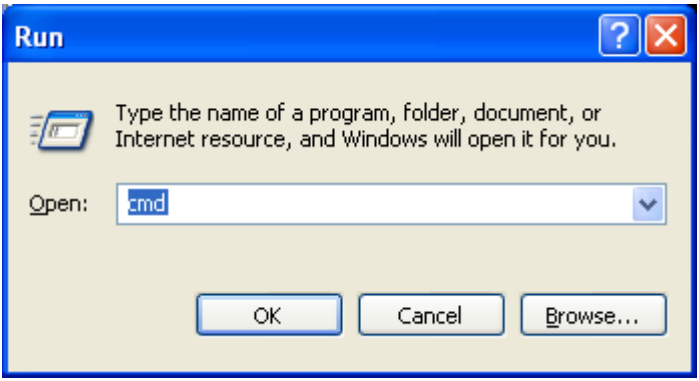

Рис. 17. Запуск командной строки

После нажатия кнопки «**OK**» откроется новое окно с чёрным фоном, в нём нужно будет выполнить команду ОС cd для смены текущего каталога на рабочий каталог, хранящий исполняемый файл нашей программы, например

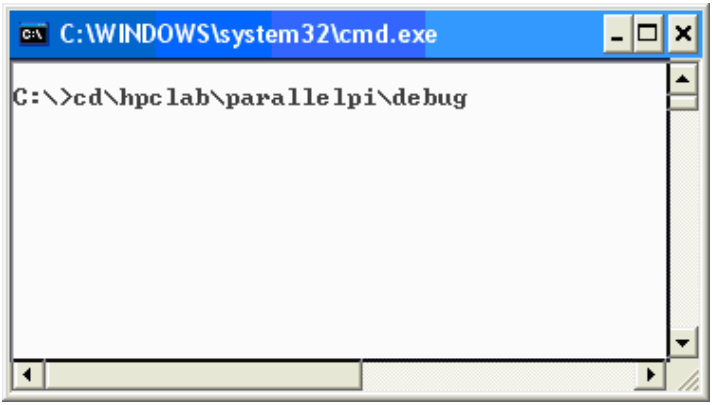

Рис. 18. Пример набора команды

Для запуска программы используйте команду mpiexec с параметрами, указанными в таблице 2, наберите её в командной строке следующим образом

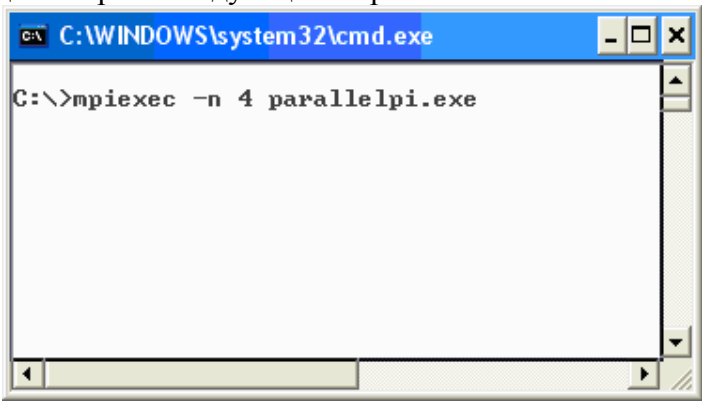

Рис. 19. Пример набора команды

Предлагается выполнить следующее задание по изучению процедуры отладки параллельной программы.

Для этого необходимо установить «свойства отладки» для нашего проекта в среде Microsoft Visual Studio как показано на следующих рисунках

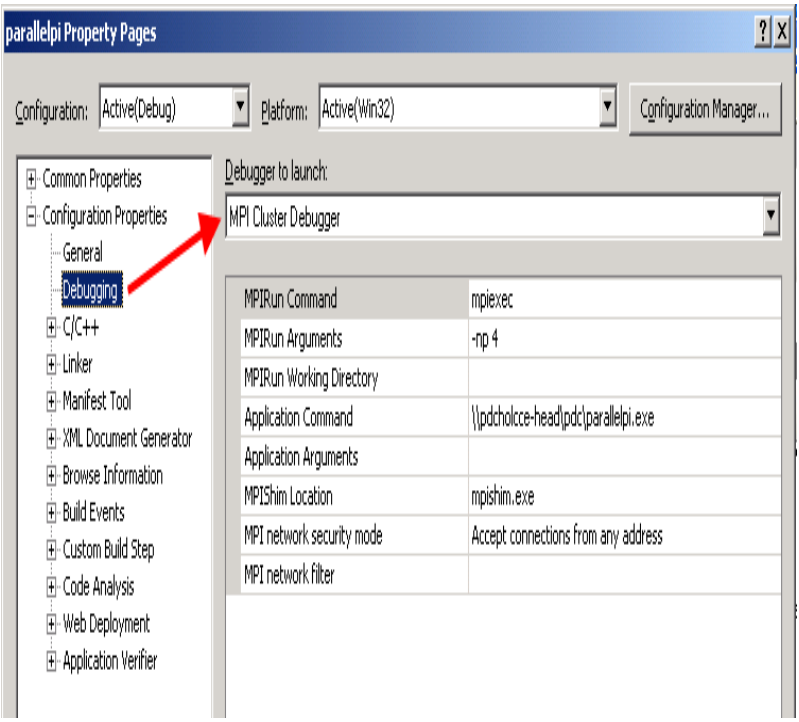

Рис. 20. Окно свойств отладки

В данной вкладке необходимо установить следующие значения для полей:

MPIRun Command	mpiexec.exe
MPIRun Arguments	-np 4
MPIRun Working Directory	
Application Command	\\pdcpath-head\PDC\parallelpi.exe
Application Arguments	
MPIShim Location	mpishim.exe
MPI network security mode	Accept connections from a local subnet only
MPI network filter	

здесь также необходимо указать сетевой путь к отлаживаемой параллельной программе в поле Application Command, далее выполнить следующие установки

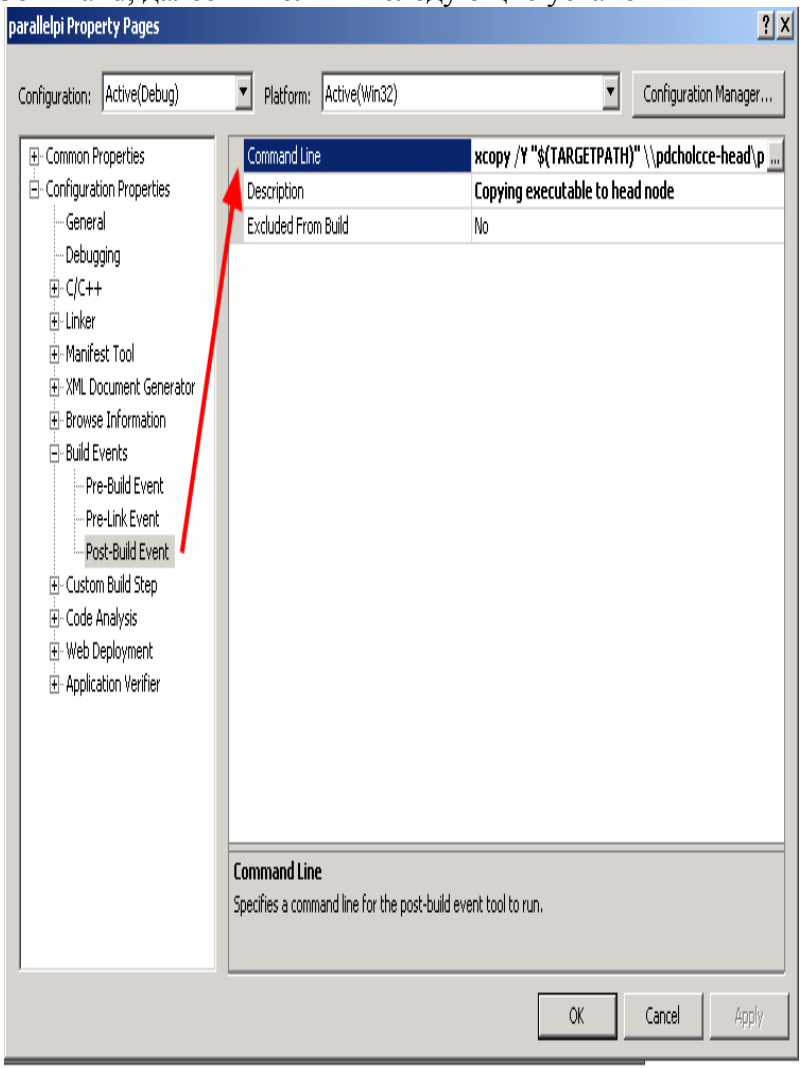

Рис. 21. Задание параметров

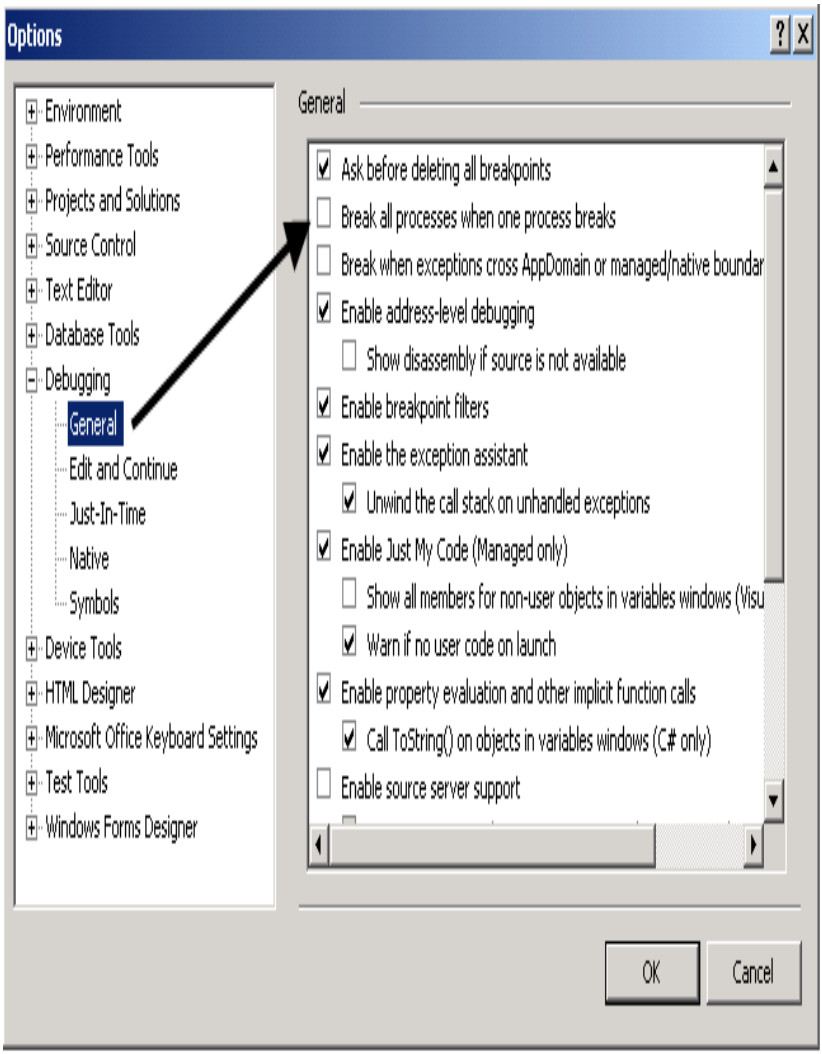

Рис. 22. Задание параметров

Далее, уже в тексте отлаживаемой программы устанавливаем точку остановки, например как показано на следующем рисунке

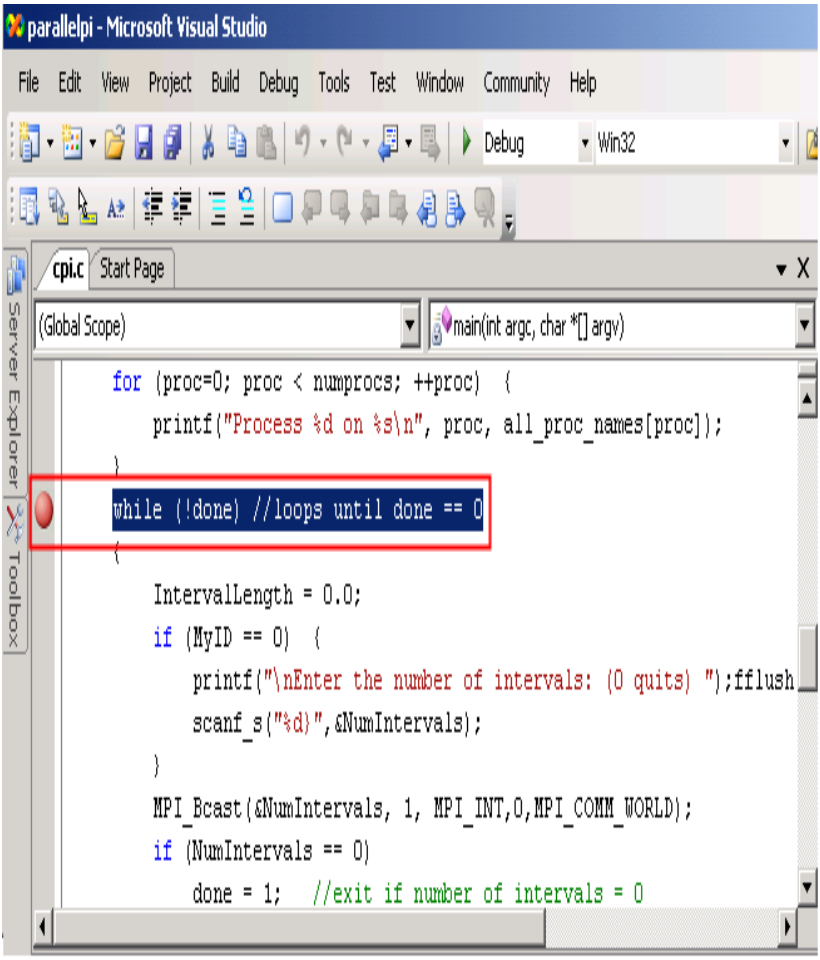

Рис. 23. Задание точки остановки

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Общие требования: *для всех вариантов необходимо написать параллельный и последовательный вариант программы, выполнить отладку, провести анализ результатов выполнения программы в зависимости от объема входных данных и числа использованных процессоров*. Все полученные результаты предоставляются студентами в виде отчета по лабораторной работе установленного образца.

Вариант 1. Программа выполняет транспонирование матрицы $N \times N$ на M процессорах. Каждому процессу передаются N / M строк, а после транспонирования каждый процесс возвращает N / M столбцов.

Вариант 2. Напишите программу решения любого дифференциального уравнения сеточным методом, используя разные шаблоны, например «крест».

Вариант 3. Напишите программу вычисления максимального (минимального) элемента массива вещественных значений.

Вариант 4. Напишите программу перемножения двух матриц.

Вариант 5. Напишите программу вычисления скалярного произведения векторов.

Вариант 6. Даны матрицы A и B. Напишите программу вычисления $C = AB - BA$.

Вариант 7. Дана матрица A и векторы a и b. Напишите программу вычисления $p = (a, Ab)$, ( )—скалярное произведение векторов.

Вариант 8. Дана матрица A и векторы a и b. Напишите программу вычисления $c = a - Ab$.

Вариант 9. Необходимо смоделировать одномерное движение частиц в определенной области. Каждая частица характеризуется двумя значениями—координатой и скоростью. В начальный момент времени информация обо всех частицах находится на одном процессе, а затем распределяется поровну между другими процессами, которые моделируют движение своей группы частиц.

Вариант 10. Напишите программу вычисления сумм всех элементов в строке двумерного массива.

Вариант 11. Напишите программу, реализующую схему Горнера вычисления значения многочлена в точке.

Вариант 12. Даны матрицы A и B—квадратные матрицы. Напишите программу вычисления $C = A^T (B + B^T) A$.

Вариант 13. Напишите программу для вычисления рекуррентных значений $u_{i,j,k} = F(u_{i-1,j,k}, u_{i,j-1,k}, u_{i,j,k-1})$ для произвольной функции $F(x, y, z)$

Вариант 14. Напишите программу вычисления определенного интеграла методом Монте-Карло.

Вариант 15. Вычислить сумму ряда с точностью EPS:

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{(n+1)(n+2)} * f(n), \quad \text{где } f(n) = \sum_{i=1}^{n} \frac{1}{(2*i)!+2}.$$

Вариант 16. Вычислить сумму ряда с точностью EPS:

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{2n(n+1)} * f(n), \quad \text{где } f(n) = \sum_{i=1}^{n} \frac{1}{4^i}.$$

Вариант 17. Вводится X. Получить значение Y=(1+X*(2+X*(3+X*(...(9+X*10)...)))).

Вариант 18. Напишите программу вычисления функции $n!$ с сохранением всех цифр разрядов.

Вариант 19. Дано натуральное число N. Получить все такие натуральные q, что N делится на q^2 и не делится на q^3. Полученные значения q записать в одномерный массив.

Вариант 20. Даны числа A_1, A_2, ..., A_m. Известно, что $A_1 > 0$ и что среди A_2, A_3, ..., A_m есть хотя бы одно отрицательное число. Пусть A_1, ..., A_n – члены данной последовательности, предшествующие первому отрицательному члену (n заранее неизвестно).

Получить $\max\left(A_1^2,\ A_2^2,\ \ldots, A_n^2\right)$. Числа A_1, A_2, ..., A_m представлены в виде одномерного массива.

Вариант 21. Даны числа A_1, A_2, ..., A_m. Известно, что $A_1 > 0$ и что среди A_2, A_3, ..., A_m есть хотя бы одно отрицательное число. Пусть A_1, ..., A_n – члены данной последовательности, предшествующие первому отрицательному члену (n заранее неизвестно).

Получить $\min\left(A_1 + A_2,\ A_2 + A_3,\ \ldots,\ A_{n-1} + A_n\right)$. Числа A_1, A_2, ..., A_m представлены в виде одномерного массива.

Вариант 22. Написать программу генерации простых чисел.

Вариант 23. Заданы некоторые символы алфавита, длина слова. Написать программу, генерирующую все возможные слова с заданной длиной и алфавитом

Вариант 24. Дана функция $z = f(x, y)$, x,y находятся в единичном квадрате. Написать программу поиска максимального (минимального) значения этой функции.

Список полезных источников информации

1. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Петербург, 2002.
2. URL: http://windowshpc.net (дата обращения: 29.11.10).
3. URL: http://www.microsoft.com/hpc/ (дата обращения: 29.11.10).

СОДЕРЖАНИЕ

Учебное издание

Составитель Михаил Аркадьевич Клочков

Основы организации высокопроизводительных вычислений в Windows HPC Server 2008

Учебно-методическое пособие

Отпечатано с оригинал-макета заказчика
Подписано в печать 1.12.10. Формат 60x84$^{1/16}$.
Печать офсетная. Усл. печ. л. 3,6 Уч.-изд. л. 1,35
Заказ № Тираж 25 экз.
Издательство «Удмуртский университет»
426034, Ижевск, Университетская, 1, корп. 4.