Математические вопросы кибернетики и вычислительной техники 23, 2004, 154 – 162.

Система управления заданиями на многомашинном вычислительном комплексе*

Микаел К. Гюрджян

Институт проблем информатики и автоматизации НАН РА

Аннотация

Для эффективной организации высокопроизводительных кластерных вычислений разработана программная система для управления заданиями на многомашинном вычислительном комплексе(кластере). Целью системы является оптимизация управления заданиями в соответствии с вычислительными ресурсами и архитектурными особенностями многомашинного вычислительного комплекса, предоставление пользователю удобного сервиса для использования доступных вычислительных ресурсов и обеспечение безопасного доступа к вычислительной среде кластера.

1. Требования к системе.

Рост интереса к высокопроизводительным вычислениям и расширение возможностей доступа к высокопроизводительным вычислительным ресурсам, прежде всего к параллельным Linux-кластерам, определяет необходимость создания средств поддержки для разработчиков параллельных программ [16,17,19,22,23]. Способы и средства организации управления ресурсами многомашинного вычислительного комплекса (кластера) обуславливаются особенностями его программно-аппаратной архитектуры.

Для эффективной организации высокопроизводительных кластерных вычислений нами разработана программная система организации и управления заданиями на многомашинном вычислительном комплексе. Целью данной программной системы является оптимизация управления заданиями в соответствии с вычислительными ресурсами и архитектурными особенностями кластера, предоставление пользователю удобного сервиса для использования доступных вычислительных ресурсов и обеспечение безопасного доступа к вычислительной среде кластера.

При разработке программной системы учитывалось целое множество требований, среди которых можно отметить следующие[14,15,16,17,18,20,22]:

- наличие простого и удобного способа для удаленного доступа к кластеру, не требующего установки специального программного обеспечения;
- обеспечение безопасности доступа к вычислительной среде кластера;
- создание собственной системы авторизации пользователей, не связанной напрямую с системой авторизации операционной системы;
- реализация достаточного набора операций для выполнения вычислительных заданий на кластере (добавление задачи в очередь, удаление задачи из очереди, получение текущего статуса задачи, получение результатов вычислений, получение данных о работающих задачах);
- запоминание результатов выполнения вычислительных заданий;

* Работа выполнена в рамках работ по проекту МНТЦ А-823.

- возможность оповещения пользователя о запуске и/или завершении его задач;
- предоставление удобных утилит для компиляции и линковки параллельных приложений, не требующих составления специальных make-файлов;
- организация системы очередей заданий;
- оптимальное управление прохождением заданий на кластере;
- удобство интерфейса пользователя при работе с системой очередей;
- обеспечение прозрачности выполнения пользовательских заданий (система мониторинга заданий);
- наличие удобного WEB-интерфейса для мониторинга прохождения заданий, установки заданий в очередь, удаления задания из очереди и просмотра результатов вычислений .

Часть этих задач могут выполнять существующие системы. Например, задачи мониторинга могут решать пакеты Big Sister (Big Brother) [7], autocheck [5], Ganglia [2], mon [8] и другие. Для управления заданиями существуют такие пакеты как PBS (OpenPBS) [1], NQS [4], DQS [3], CLEO [9] и др. Для компиляции и линковки и запуска приложений можно воспользоваться утилитами программной среды MPICH [6]. Однако эти пакеты не ориентированы на решение всего вышеперечисленного набора задач и, поэтому, возникает потребность создания программного комплекса, который являлся бы надстройкой над уже существующими системами и удовлетворял указанным требованиям.

2. Условия работы системы.

Для функционирования программной системы на многомашинном вычислительном комплексе необходимо наличие следующих компонент: многомашинного вычислителя, управляющей ЭВМ, сервера (серверов) доступа и файлового сервера.

Введем несколько определений:

Вычислительный модуль – компьютер под управлением единой ОС Linux. При этом модуль может быть многопроцессорной SMP-системой.

Вычислитель – совокупность нескольких вычислительных модулей, соединенных локальной сетью и/или иной коммуникационной средой.

Параллельный ресурс – подсистема из одного или нескольких связанных процессоров вычислителя.

Управляющая ЭВМ – специально выделенный компьютер, соединенный с вычислителем локальной сетью, на котором ведется *очередь* задач к вычислителю. На управляющей ЭВМ функционирует планировщик задач. Возможно объединение функций файлового сервера и управляющей ЭВМ на одной рабочей станции.

Сервер доступа – специально выделенный компьютер, который служит для доступа пользователей и инициации всех команд пользователей. Сервер доступа является шлюзом между открытой сетью и вычислительной установкой. Серверов доступа может быть несколько. На одном из них может быть установлен WEB сервер. На сервере доступа должны быть устранены наиболее узкие звенья в системе безопасности.

Файл сервер – специально выделенный компьютер, на котором хранятся пользовательские директории в виде */home/имя_пользователя*. Сервер служит для подготовки и хранения исходных текстов программ пользователей, данных для пользовательских задач, результатов расчетов, для компиляции и подготовки самих задач.

Предполагается также полная симметрия возможностей доступа всех процессов (а значит, выполняющих их процессоров) к единой файловой системе. Это достигается использованием NFS [10] на базе сетевых интерфейсов управляющей сети. Все вычислительные модули, управляющая ЭВМ и серверы доступа монтируют директорию пользователей */home* под тем же именем в качестве NFS – клиентов.

Доступ пользователей к шлюзу осуществляется через протокол и систему аутентификации SSH [11], по этому же протоколу происходит поступление входных и выходных, а также управляющих данных. Протокол SSH [11] обеспечивает хорошую защищенность передаваемых данных. Возможность доступа определяется физической доступностью вычислительных ресурсов,

регистрацией пользователя на системе, наличием прав у пользователя для доступа к желаемому ресурсу.

При входе через SSH [11] или через WEB-интерфейс(с поддержкой SSL) пользователь получает доступ не ко всем приложениям, а только к программной системе для организации и управления заданиями на кластере. Доступ к управляющей ЭВМ разрешен только через программную систему для организации и управления заданиями. Пользовательский доступ на узлы кластера запрещен. Разрешен только SSH [11], и только в то время, когда соответствующий модуль занят процессом данного пользователя. Соответственно, все действия на вычислительных модулях выполняет от имени пользователя система запуска, а пользователь выдает ей команды, находясь в сеансе на сервере доступа через управляющую ЭВМ. Пользователю *root*, однако, разрешены любые действия и на управляющей ЭВМ и на узлах, что и позволяет администрировать установку.

3. Утилиты компиляции и линковки параллельных приложений.

После написания параллельной программы возникает вопрос о его компиляции и линковке. В программный пакет MPICH [6] включены команды *mpicc* (для программ на C), *mpiCC* (для программ на C++), и *mpif77/mpif90* (для программ на Фортране 77/90) для компиляции и линковки параллельных приложений, написанных с использованием стандарта передачи сообщений (MPI [13]). Но при использовании соответствующих команд можно получить исполняемый файл, который не будет оптимизирован для данной среды и для расчета задания потребуется больше времени. Для сборки многомодульных программ возникает необходимость написания соответствующих make-файлов, что в свою очередь является не простой задачей.

Для этих целей разработаны универсальные Makefiles (*Makefile*, *Makefile.rules*, *Makefile.globals*) для компиляции программ на высокопроизводительных системах (для языков Фортран, C, C++).

- *Makefile* – задается политика компиляции приложений;
- *Makefile.rules* – описываются правила для компиляции приложений;
- *Makefile.globals* – задаются все необходимые флаги.

Makefile написаны с использованием всех оптимизирующих флагов для данной архитектуры. Параметры для Makefile-ов передаются с помощью разработанных утилит *mpicc*, *mpiCC* и *mpif77/mpif90*.

Основными опциями вышеуказанных утилит являются:

- *mcs_arch* – архитектура вычислительной системы;
- *prog_path* – директория программы;
- *prog_name* – имя программы;
- *prog_type* – тип программы (исполняемая программа, статическая или динамическая библиотека.
- *include_path* – директории использованных **.h* файлов;
- *library_path* – используемые пользовательские библиотеки.

Процесс компиляции и линковки параллельной программы происходит на сервере доступа.

4. Конфигурационный файл задачи.

После компиляции программы и подготовки ее выполняемых модулей пользователь должен выдать команду для постановки этой программы в очередь задач, готовых к выполнению на вычислителе. В процессе выполнения этой команды системой готовится *конфигурационный файл задачи*, в который записываются параметры параллельной задачи.

Конфигурационный файл представляет собой файл, состоящий из двух секций: [*General*] и [*Batch*]. Основными параметрами секции [*General*] *конфигурационного файла* являются следующие:

- *task_name* – имя параллельной задачи. При постановке задачи в очередь ей, помимо символьного имени, будет присвоен уникальный номер. Полное имя задачи будет

складываться из значения параметра *task_name* и, через точку - уникального номера. Такой порядок формирования полного имени задачи необходим для обеспечения уникальности имен.

- *host_directory* – имя *каталога стандартного вывода* для параллельной задачи. В данном каталоге при постановке задачи в очередь будет создан подкаталог с именем, совпадающим с полным именем задачи. В этом подкаталоге размещается файлы стандартного вывода параллельной задачи. При одновременном запуске двух экземпляров задачи с одним и тем же конфигурационным файлом, благодаря уникальности полных имен задач, в каталоге стандартного вывода будут созданы разные подкаталоги для разных экземпляров задачи.
- *stdout* – имя файла стандартного вывода задачи.
- *stdin* – имя файла стандартного ввода задачи.
- *stderr* – имя стандартного вывода ошибок задачи.
- *cpu_count* – число процессоров для выполнения задачи
- *user* – имя пользователя владельца параллельной задачи.
- *time_request* – время (по максимуму), необходимое для выполнения задачи.

Секция [*Batch*] содержит текст командного файла, который будет выполнен системой.

5. Схема взаимодействие процессов в системе.

Система для организации и управления заданиями на многомашинном вычислительном комплексе представляет собой совокупность взаимодействующих процессов на серверах доступа и управляющей ЭВМ, основные из которых следующие:

- **mpimcs** – клиент, работающий на сервере доступа, обеспечивает связь пользователя с системой управления прохождением задач. Именно этой программе передаются все пользовательские команды.
- **mpimcsd** – сервер запросов, отвечающий за связь с клиентом и обработку клиентских запросов. Функционирует на управляющей ЭВМ.
- **scheduler** – сервер очередей параллельных задач, функционирует на управляющей ЭВМ. Сервером очередей может быть любой доступный планировщик (maui[12], OpenPBS [1],CLEO [9] и д. т).

Рассмотрим схему взаимодействия процессов в системе на примере запуска параллельного приложения. Задачи, использующие MPI [13], запускаются через специальный командный файл **mpirun**, формирующий конфигурационный файл задачи и передающий его системе. В начале файла **mpirun** переменной *MPIR_HOME* присваивается имя корневой директории с файлами MPI [13]. После вызова файла *mpirun.args*, в котором обрабатываются параметры командной строки, формируется секция [*General*] конфигурационного файла задачи на основе этих параметров. Далее формируется секция [*Batch*] конфигурационного файла задачи.

Командный файл, формируемый в секции [*Batch*] конфигурационного файла задачи, служит для восстановления окружения, существующего при постановке задачи в очередь. Для этого в командном файле формируются инструкции перехода в директорию, из которой запускается задача, и восстановление переменных окружения, существовавших в момент постановки задачи в очередь. После формирования команд запуска программы на выполнение генерация секции [*Batch*] заканчивается, и сформированный конфигурационный файл задачи передается в очередь задач через программу **mpimcs**.

При наборе пользователем любой из команд системы вызывается клиент **mpimcs** с соответствующими набранной команде ключами. Клиент **mpimcs** соединяется с сервером запросов **mpimcsd**, проходит авторизацию и передает ему на исполнение пользовательскую команду. Общение между клиентом и сервером происходит через протокол TCP/IP, что позволяет выполнять клиент и сервер на разных ЭВМ. Все команды клиента сервер **mpimcsd** выполняет самостоятельно, за исключением команд на запуск и постановку в очередь параллельных задач, а также команды удаления задания из очереди, которые передаются для исполнения серверу очередей.

Процесс авторизации пользователя происходит следующим образом. После установления соединения с клиентом **mpimcsd** ожидает ввода имени пользователя от клиента **mpimcs**. После того как клиентом передано пользовательское имя, **mpimcsd** генерирует случайное 64-байтное число (ключ) и сообщает его клиенту. Для аутентификации клиент должен прочитать 64-байтный ключ сообщить его серверу. Сервер проверяет подлинность ключа и в случае несовпадения доступ запрещается. При этом сервер выдаст сообщение "*Login incorrect*" после чего соединение разрывается. При совпадении ключей сервер выдает приглашение "*OK*". Данная последовательность символов является ключевой и означает, что сервер готов к вводу очередной команды. После ввода и выполнения команды сервер выдает ответ в следующем формате. Первая строка ответа – код завершения, 0 – успешно, не 0 – ошибка. Далее могут идти несколько строк сообщения (в случае нормального завершения - информационного, в случае ошибки – сообщения об ошибке).

При поступлении команды на запуск или постановку в очередь параллельной задачи **mpimcsd** получает от клиента конфигурационный файл задачи, содержащий ее описание в терминах системы.

Сервер проверяет корректность данных задачи. В качестве ограничений могут быть заданы: количество процессоров на одну задачу, количество одновременно занятых процессоров (если одновременно запускается несколько задач), максимальное время счета задачи, максимальный приоритет задачи. Вся эта информация хранится в виде таблиц в базе данных MySQL, доступ на чтение и запись которого предоставлен только суперпользователю. В базе данных хранится также информация о запущенных (завершенных, стоящих в очереди) задачах вместе со всеми его параметрами (время запуска, время счета, время окончания задания, потребованное количество процессоров и т.д.).

После того как сервер проверяет корректность данных задачи, он обращается к серверу очередей. Сервером очередей может служить любой доступный планировщик заданий на кластере (CLEO [9], OpenPBS [1]). Серверы очередей ведут очередь параллельных задач согласно принципам планирования очередей. Наиболее типичными являются следующие схемы планирования и обслуживания заданий:

1. задания поступают на обслуживание в порядке поступления их в систему, при этом, учитываются все задания из очереди до первого, требующего больше узлов, чем имеется;
2. задания поступают на обслуживание в порядке поступления их в систему, при этом, задания в очереди перебираются до конца;
3. задания поступают на обслуживание в порядке числа требуемых узлов, при этом, задания в очереди перебираются до конца.

Система позволяет разбивать вычислитель на две и более логических систем. Вычислительные модули, отнесенные к той или иной логической системе, не могут пересекаться. Механизм логических систем позволяет работать одной управляющей ЭВМ сразу с несколькими вычислителями. Логические системы различаются по *именам*. Сервер запросов один и тот же для разных логических систем, т.к. именно на его уровне производится деление на системы. Логические системы определяются администратором через соответствующие конфигурационные файлы.

Благодаря логическим системам сервер **mpimcsd** может сообщаться с несколькими серверами очередей (планировщиками), зная конечно протокол обмена с сервером очереди. Каждая логическая система может иметь свой собственный сервер очередей. Часть кластера может работать с одним сервером очередей (например CLEO [9]), другая часть с другим сервером очередей (например OpenPBS [1]). Каждый сервер очереди (планировщик) может быть настроен на определенный параллельный ресурс, настройка которого задается в его конфигурационном файле. Исходя из параметров конфигурационного файла задачи сервер сам решает какому планировщику передавать соответствующее задание на обслуживание. Существенными характеристиками задачи из числа данных конфигурационного файла являются количество процессоров и время исполнения, необходимые для решения задачи. На основании этих данных, рассчитывается время ожидания задания до поступления его на обслуживание и передачи в связи с этим соответствующему планировщику. То есть задание передается тому планировщику, время ожидания в очереди которого до поступления на обслуживание – минимально.

Общая схема запуска заданий сервером очередей следующая:

- определение свободных вычислительных модулей;
- выделение требуемого параллельного ресурса для параллельной задачи;
- порождение с помощью команды *fork()* специального процесса-менеджера задачи, который запускает задачу и контролирует её;
- запись информации о запущенной задаче в базе данных;

Затем менеджер ждет завершения порожденных им процессов или сигнала на принудительное завершение задачи. Получив тот или иной сигнал, менеджер формирует код возврата и диагностическое сообщение, передает их серверу очередей и завершает работу.

Решение успешно запущенной задачи может быть завершено четырьмя различными способами:

- «естественным» путем, т.е. закончив счет;
- по истечении заказанного времени;
- принудительно пользователем;
- принудительно администратором системы.

В любом случае, при завершении выполнения задачи менеджер осуществляет следующие действия:

- производит очистку всех выделенных задаче вычислительных модулей;
- соединяется со своим сервером очередей и сообщает ему о завершении задачи и освобождении ресурсов, после чего завершает работу.

Сервер очередей в свою очередь сообщает это сообщение серверу заданий, который записывает информацию о запущенных и завершенных задачах в базу данных.

6. Пользовательские утилиты для работы с системой.

6.1 Запуск на исполнение параллельной MPI-программы

Как было уже сказано выше, запуск на исполнение параллельной MPI-программы производится с помощью команды:

mpirun *[mpirun_args]* *<programm name>* *[programm_args]*.

Параметры команды **mpirun** следующие:

- *maxtime* – максимальное время счетамаксимальное время счета. После истечения этого времени задача принудительно заканчивается.
- *np* – число процессоров, требуемое программе.
- *stdiodir* – имя каталога стандартного ввода/вывода, в который будут записываться файл стандартного вывода и имена узлов, на которых запускалась задача.

Удачно запущенная задача получает определенный номер, который добавляется к имени задачи. Это позволяет пользователю запускать одновременно несколько экземпляров задачи с одним и тем же именем – система присвоит каждому экземпляру задачи уникальный номер. Для каждого экземпляра будет создан отдельный каталог стандартного ввода/вывода.

При удачном старте система выдаст пользователю следующую информацию:

- имена выделенных под задачу узлов;
- сведения о принятых системой установках по умолчанию;
- сообщение об удачном старте задачи, причем в сообщении указывается присвоенный задаче номер;
- статус задания в системе (пошло на обслуживание или стало в очередь).

6.2 Получение информации о запущенных задачах

Информацию о запущенных пользователем задачах можно получить с помощью команды:

mpistat *[task_name.id][stat_arg]*.

Параметром для команды служит имя задачи и – через точку – ее номер. При отсутствии параметра на экран будет выдан список всех запущенных пользователем и работающих или находящихся в очереди на момент выдачи команды задач. При задании параметра система выдаст информацию о задаче – время старта (и завершения, если задача уже завершилась), имена узлов, на которых выполняется задача.

Параметр команды **mpistat** следующий:

history – выдаст информация о всех запущенных, находящихся в очереди и уже выполненных заданиях. Данная информация хранится в базе данных.

6.3 Завершение запущенной или поставленной в очередь задачи.

Завершить запущенную или поставленную в очередь задачу можно командой:

mpikill *[task_name.id]*.

Параметром для команды служит имя задачи и – через точку – ее номер. Данная команда допускает задание в качестве параметра маски Unix-формата с использованием символов-джокеров. По этой команде будут завершены все задачи, имена которых удовлетворят заданной маске. При отсутствии параметра пользователю будет выдан список всех запущенных задач и предложено ввести номер (по списку) той задачи, которую нужно завершить.

6.4 Просмотр стандартного вывода задачи.

Стандартный вывод задачи можно посмотрть с помощью команды:

mpiout *[task_name.id][out_file]*.

Первым параметром для команды служит имя задачи и – через точку – ее номер. Если задание находится в очереди или на обслуживании системой, то вывод данной команды будет пустым.

Значения второго параметра команды следующие:

- *Out* - означает выдачу стандартного вывода;
- *Err* - означает просмотр стандартного вывода сообщений об ошибках.

При запуске задачи система создаст в указанном в конфигурационном файле задачи каталоге стандартного ввода/вывода подкаталог, имя которого будет совпадать с именем запущенной задачи (включая номер через точку). В этом подкаталоге и будут непосредственно размещены файлы стандартного ввода/вывода задачи. Система сама не уничтожает эти файлы и подкаталог. После завершения выполнения задачи данные файлы полностью доступны пользователю.

7. Заключение

Плюсами данной системы являются следующие возможности:

- Обеспечение доступа к системе с любого компьютера, подключенного к сети Интернет, использовав в качестве средств доступа ssh-клиент (что позволяет, в частности, не устанавливать на компьютерах пользователей какого-либо дополнительного программного обеспечения);
- Обеспечение безопасности от несанкционированных действий к вычислительной среде кластера;
- Замену или добавления нового планировщика и изменение стратегии распределения вычислительных ресурсов кластера;
- Трансляции своих программных комплексов без необходимости составления make-файлов и выбора опций компилятора;

- Предоставление пользователю удобных утилит для организации своих заданий на кластере.
- Сбор в базе данных и выдачу статистики использования задачами пользователей вычислительных ресурсов кластера.

Выражаю глубокую благодарность моему научному руководителю В.Г.Саакяну за ценные советы и помощь в процессе работы над статьей.

Литература

1. «Portable Batch System» www.openpbs.org
2. «Ganglia» http://ganglia.sourceforge.net
3. «DQS-Distributed Queuing System» http://www.scri.fsu.edu/~pasko/dqs.html
4. «NQS-Network Queuing System» http://umbc7.umbc.edu/nqs/nqsmain.html
5. «Autocheck» http://sourceforge.net/projects/autocheck/
6. «MPICH–Portable Implementation of MPI» http://www-unix.mcs.anl.gov/mpi/mpich/
7. «Big Brother» http://quest.com/bigbrother/
8. «MON» http://www.kernel.org/software/mon/
9. «CLEO» Руководство системного программиста.
10. "NFS-Network File System" http://nfs.sourceforge.net/
11. «OpenSSH» http://www.openssh.com/
12. «Maui Scheduler» http://supercluster.org/mauidocs/
13. «The Message Passing Interface (MPI) standard» http://www-unix.mcs.anl.gov/mpi/
14. С.А.Жуматий, А.А.Кальянов. Комплекс мониторинга распределённых информационно-вычислительных систем. // Труды Всероссийской научной конференции . 2002 г., Изд-во МГУ. С. 47-49.
15. С.А.Жуматий. Средства мониторинга и управления параллельными вычислительными системами. // Труды Всероссийской научно-методической конференции «Телематика 2003».
http://tm.ifmo.ru/tm2003/db/doc/get_thes.php?id=281
16. В.П.Гергель, А.Н.Свистунов. Система организации высокопроизводительных вычислений для кластера нижегородского университета. // Труды Всероссийской научно-методической конференции «Телематика 2003». http://tm.ifmo.ru/tm2003/db/doc/get_thes.php?id=133
17. А.А. Букатов, Г.М.Хачкинаев. Система управления пакетными заданиями в гетерогенной вычислительной сети ЦВВ РГУ. // Труды Всероссийской научно-методической конференции «Телематика 2002». С. 261-263.
18. А Шевель. Визуализация состояний вычислительного кластера. // Открытые Системы, №1, 2003, С. 10-12.
19. А.А.Андреев, Вл.В.Воеводин, С.А.Жуматий. Кластеры и суперкомпьютеры – близнецы или братья? // Открытые Системы, № 5-6, 2000. http://www.osp.ru/os/2000/05-06/009.htm
20. В. Коваленко, Е. Коваленко. Пакетная обработка заданий в компьютерных сетях. // Открытые системы, 2000, № 7-8. С. 1-19.
21. Коваленко В.Н., Коваленко Е.И., Корягин Д.А, Любимский Э.З., Хухлаев Е.В., Управление заданиями в распределенной вычислительной среде. // Открытые системы, 2001. № 5-6, С. 22-28, http://www.osp.ru/os/2001/05-06/022.htm
22. W.Saphir, L.A.Tanner, B.Traversat. Job Management Requirements for NAS Parallel Systems and Clusters. NAS Technical Report NAS-95-006 February 1995. http://www.nas.nasa.gov/Research/Reports/Techreports/1995/nas-95-006-abstract.html
23. Rajkumar Buyya. High Performance Cluster Computing. Volume1: Architectures and Systems. Volume 2: Programming and Applications. Prentice-Hall Inc., 1999. P. 664.

Խնդիրների ղեկավարման համակարգ բազմամեքենայական հաշվողական համալիրում

Մ. Ղ. Գյուրջյան

Ամփոփում

Բազմամեքենայական հաշվողական համալիրում (կլաստերում) մշակված է ծրագրային համակարգ բարձր արտադրողականությամբ կլաստերային հաշվարկների էֆեկտիվ կազմակերպման համար: Համակարգի նպատակն է բազմամեքենայական հաշվողական համալիրի խնդիրների ղեկավարման օպտիմալացումը ըստ հաշվողական ռեսուրսների և ճարտարապետական հատկությունների, օգտագործողի համար հասանելի հաշվողական ռեսուրսների օգտագործման հարմարավետ միջոցների ստեղծումը և կլաստերի հաշվողական միջավայրին անվտանգ հասանելիության ապահովումը: