УДК 681.3

В.А. Марьяновский, С.Д. Погорелый, Ю.В. Бойко, Д.Б. Грязнов, А.Д. Ломакин

Программное обеспечение *UACLUSTER*

Предложена методика проведения кластерных вычислений в организации без выделения дополнительных вычислительных ресурсов и создано программное обеспечение поддержи этой методики. Предусмотрен универсальный метод управления режимами работы кластерной системы. Создан механизм временной приостановки кластерных вычислений и их возобновления.

The methods of conducting the cluster calculations in an organization without dedicating the additional computing resources is suggested. The corresponding software is developed to support this technique. A universal method of the management of the cluster system operating mode is implemented. A mechanism of the temporary cluster calculation suspending and its further resuming is created.

Запропоновано методику проведення кластерних обчислень в організації без виділення додаткових обчислювальних ресурсів та створено програмне забезпечення підтримки цієї методики. Передбачено універсальний метод управління режимами роботи кластерної системи. Створено механізм тимчасової призупинки кластерних обчислень та їх відновлення.

Введение. Интерес к распределенным вычислениям растет с каждым днем, это объясняется тем, что на данный момент наблюдается уменьшение интенсивности роста вычислительной мощности микропроцессоров компьютеров, поэтому использование многоядерных процессоров или распределенных вычислений – это альтернатива возрастанию вычислительной мощности. В свою очередь это требует распараллеливания вычислительных задач и формирования фонда параллельных алгоритмов.

Кластерные системы могут быть построены с использованием различных операционных систем (ОС). Более 70% кластерных вычислений используют платформу *Linux*, что подтверждает статистика [1], а в роли альтернативной платформы, целесообразно использовать ОС *Windows* совместно с *Compute Cluster Server* (*CCS*). *Windows HPC Server* 2008 – последняя версия *CCS*, в работе использован *Windows Computer Cluster Server* 2003. Основные преимущества построение кластера на платформе *Windows CCS* следующие:

- распространенность и простота ОС *Windows*, не требующая изучения других ОС;
- упрощенная миграция между платформами для приложений, разработанных с использованием *MPI*, что на практике сводится к изменению заголовков в исходном коде программы, при миграции с *Linux* к *Windows*-кластеру. Для проверки этого утверждения использована программа, уже работающая на *Linux*-кластере [2] и запущенная на *Windows*-кластере, для чего потребовалось заменить заголовки исходного кода программы, связанные с подключением *MPI*-библиотек;
- специальная процедура лицензирования для высших учебных заведений (*MSDN AA*).

Учитывая перечисленные преимущества, использование кластера на основе *CCS* позволит достаточно быстро внедрить кластерные вычисления на обычных вычислительных ресурсах организации. Далее рассмотрим кластерные вычисления на платформе *Microsoft*.

Цель работы – создание программного обеспечения для реализации концепции построения гибкой гомогенной кластерной системы [4]. Квинтэссенция подхода – построение кластера на уже существующих ресурсах организации без выделения дополнительных [4].

Такой подход возможен при простое вычислительных ресурсов большую часть времени (ночью и в выходные дни). Поэтому они могут работать в двух режимах:

Режим 1. Функционирование в составе кластера.

Режим 2. Функционирование в качестве инструментальных средств для сотрудников организации.

Для реализации всех необходимых задач программное обеспечений *UACluster* разделено на две составляющие: панель администратора и загрузчик операционной системы (далее *bootstrap*). На рис. 1 представлена принципиальная схема связей между различными составляющими кластера. Кроме созданного програм-

много обеспечения были использованы следующие стандартные службы и протоколы:

• Служба управления доступом *Active Directory* (*AD*) – неотъемлемая часть *CCS*, также необходимая для удаленного управления узлами;

• Служба динамической настройки узлов по протоколу *DHCP* (*Dynamic Host Configuration Protocol*) – задает сетевые настройки узлов, а также передает параметры сетевой загрузки при использовании технологии *PXE* (*Preboot Execution Environment*);

• Служба передачи файлов по протоколу *TFTP* (*Trivial File Transfer Protocol*) – выбор этого протокола связан со сравнительной простотой использования при написании приложений.

На рис. 1 все необходимые службы запущены на главном узле, что не является обязательным условием и позволяет избежать необходимости в дополнительном оборудовании. В таком случае необходимо из настроек кластера исключить использование главного узла при расчете задач на кластере.

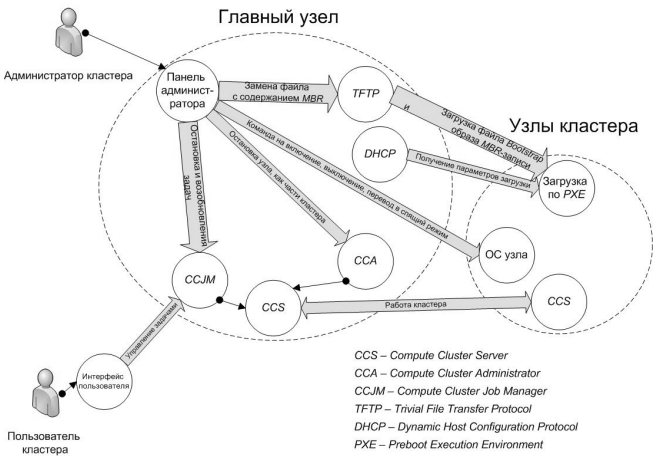

Рис. 1. Блок-схема работы кластера

Взаимодействие с *CCS* разделено на управление:

• задачами через *Compute Cluster Job Manager* (*CCJM*) – задание, отмена, проверка очереди и состояния задач на кластере, когда пользователи могут работать с кластером как через *CCJM*, так и через дополнительный интерфейс, требующий разработки или использования сторонних решений);

• узлами кластера как составляющей частью кластерной системы через *Compute Cluster Administrator* (*CCA*).

Изменения режимов работы узлов

В рабочее время компьютеры будут находиться в *режиме* 2, а в нерабочее – в *режиме* 1. В нужный момент времени необходимо менять режимы работы узлов кластера. Следует учитывать, что ОС в разных режимах будет разной, поэтому должен быть использован универсальный метод перевода компьютеров в нужный режим. Такой метод может быть основан на изменении *MBR*-записи жесткого диска перед загрузкой, необходимой ОС. Полный процесс загрузки будет состоять из следующих этапов [4]:

• использование сетевой загрузки узлами кластера с применением технологии *PXE*;

• по протоколу *DHCP* (*Dinamic Host Configuration Protocol*) узел получает конфигурацию стеки протоколов *TCP*/*IP* и параметры дальнейшей загрузки (имя *bootstrap*-файла и *IP*-адрес *TFTP*-сервера);

• узел выполняет загрузку *bootstrap*-файла с указанного *TFTP*-сервера и выполняет его;

• *Bootstrap*-файл загружает необходимый вид *MBR*-записи с *TFTP*-сервера (название файла на *TFTP*-сервере совпадает с *MAC*-адресом (физическим адресом сетевой карты) угла, так как *MBR*-запись для каждого узла своя) и записывает его в соответствующий сектор жесткого диска;

• *Bootstrap*-файл передает дальнейший процесс загрузки активному разделу жесткого диска, с которого в свою очередь загружается ОС.

Для сетевой загрузки узла необходимо настроить службу *DHCP* и *TFTP*-сервер. Кроме стандартных служб также используется файл *bootstrap*, который требует разработки.

Язык *Assembler* был использован для создания файла *bootstrap*. Рассмотрим реализацию основных действий этого файла:

• выполнить проверку наличия *PXE API* (*Application Programming Interface*). При загрузке сетевой карты в памяти формируется две структуры, описывающих загрузку экземпляра *PXE*

API. В старых реализациях *PXE* поддерживается только структура *PXENV+*, а в современных (начиная с версии 2.1 !*PXE*) существует несколько способов получения этих структур. В рассматриваемом случае использована функция 5650*h* (прерывания 1*Ah*). При ее успешном завершении в AX устанавливается значение 564*Eh*. При каких-либо проблемах предполагается, что *PXE* не поддерживается и программа завершается с формированием кода ошибки. При правильном результате пара регистров *ES*:*BX* содержит адрес структуры *PXENV+*, в последней со смещением 28*h* – адрес структуры !*PXE*. В свою очередь в структуре !*PXE* со смещением 10*h* содержатся функции *PXE API*.

```
      push es                       ;
резервирование ES
      push ds                       ;
резервирование DS
      mov ax, 5650h                 ; запись
номера фунции для определения наличия PXE
      int 1Ah                       ; вызов
прерывания 1Ah
      jc nopxe                      ; проверка
1. Флаг переноса не установлен?
      cmp ax,564Eh                  ; проверка
значения в AX
      jne nopxe                     ; выход из
программы, если PXE ек поддерживается

   pop ds                           ;
восстановление DS
      mov bx,es:[bx+28h]            ; !PXE
адрес заключен в BX, PXENV+ со смещением
BX+0A0h
      mov eax,es:[bx+10h]           ; Начальная
точка PXE API в EAX
      mov [PXEAPI_adr], eax         ; Начальная
точка PXE API в PXEAPI_adr
   pop es                           ;
восстановление ES

   ;--- Вызов PXE API найден, можно
проболжать работу
```

• Считается, что при работе с жестким диском он поддерживает механизм адресации и доступа к секторам *LBA* (*Logical Block Addressing*). Во избежание проблем повреждения данных необходимо проверить поддержку *LBA*. Рассмотрим пример работы с диском *Primary Master* (с номером 80*h*). Проверка выполняется функциями 55*AAh* (прерывание 13*h*). При правильной работе в регистр *BX* записывается значение 0*AA*55*h*.

```
   mov ah, 41h                      ; функция
проверки поддержки расширенных команд
   ; работы с диском
   mov bx, 55AAh                    ; входной
параметр для 41h
   mov dl, DRIVE                    ; входной
параметр, номер диска
   int 13h                          ; вызов
прерывания работы с диском
   jc nolba                         ; если CF
очищен, можно продолжать
   cmp bx, 0AA55h                   ; При
наличии поддержки LBA в BX записывается
0AA55h
   jne nolba                        ; если его
там нет – выход с ошибкой
```

• Получение файла – образа *MBR* с *TFTP*-сервера; для размещения его резервируется память переменной *diskbuf* размером 512 б.

```
   ; TFTP Read File Init structure
   ; структура чтения файла с TFTP-сервера
   lea ax, diskbuf                  ; адрес
рабочего буфера
   mov pxrf_v.bseg, ds              ; запись
сегмента в структуру
   mov pxrf_v.boff, ax              ; запись
смещения в структуру

   mov eax, SERVERIP          ; запись IP адреса
TFTP-сервера в EAX
   mov pxrf_v.sip, eax              ; запись IP
адреса сервера в струкутуру
   mov pxrf_v.gip, eax              ; запись IP
адреса шлюза в струкутуру

   mov ax,0                         ; входной
параметр AX=0
   push ds                          ;
резервирование DS
   lea di, pxrf_v                   ; запись в
DI ссылки на сруктуру
   push di                          ; помещение
DI в стек
   push 0023h                 ; помещение в стек
номера функции PXEAPI:TFTP Read File
   call  [PXEAPI_adr]               ; вызов
функции
   add sp,6                         ; откат на
6 позиций в стеке
   call printw                      ; вывод
статус-кода

   cmp ax, 0000h                    ; если
стетус код не 0
   jne nombr                        ; значит,
на сервере не найден требуемый mbr файл
```

• Запись полученного образа на диск осуществляется с помощью функции 43*h* (запись сектора в режиме *LBA*) прерывания 13*h*. *dap_v* – структура входящих параметров функции, в ней записан сегмент и смещение с образом *MBR*.

```
   lea ax, diskbuf                    ; адрес
рабочего буфера

   mov dap_v.boff, ds                 ; запись
сегмента в структуру
   mov dap_v.bseg, ax                 ; запись
смещения в структуру


   mov ah, 43h                        ; запись
сектора в режиме LBA
   mov al, 0                          ; запись по
умолчанию
   mov dl, DRIVE                      ; DRIVE =
80h по умолчанию

   lea si, dap_v                      ; запись в
SI ссылки на сруктуру
   int 13h                            ; вызов
прерывания работы с диском
   jc lbareaderr                      ; Что-то не
так, ошибка в AH
```

• Корректное завершение и выход из программы *PXE* – важное условие функционирования программ для различных реализаций аппаратных платформ.

```
   ;--- PXENV_STOP_BASE
   lea di, pxst_v                     ; рабочая
структура выгрузки PXE
   push di                            ; помещение
DI в стек
   push 0076h                ; помещение в стек
номера функции выгрузки PXE

   call  [PXEAPI_adr]                 ; вызов
функции
```

Полный объем программы манипуляции *MBR*-записью составляет около шести сотен строк с комментариями (полный код программного обеспечения) [3].

Программные средства управления узлами кластера

Кроме смены режимов работы узлов кластера, необходимо также реализовать следующую функциональность:

• автоматическое изменение режимов работы узлов кластера;

• останавливать роботу узлов кластера – «замораживание» задач на кластере для их возобновления после перезагрузки;

• сохранение *MBR*-записи для каждого узла.

Для реализации этого и была использована среда разработки *Visual Studio* 2005 и язык программирования *C#*. Интерфейс пользователя представлен на рис. 2, где показан список всех узлов и их режим, *IP*-адрес и *MAC*-адрес. Он позволяет проверять и задавать режим работы для каждого узла. Конфигурация узлов сохраняется в конфигурационном файле (язык *XML*). Работа с объектами конфигурационного файла выполняется с использованием пространства имен *System. Xml.Serialization*.

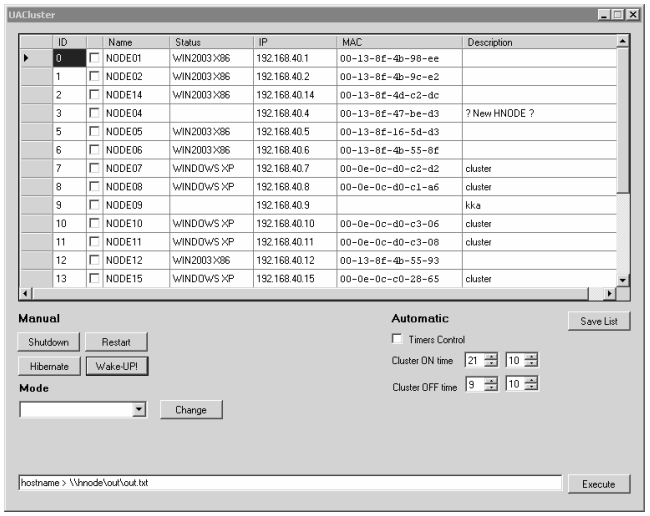

Рис. 2. Интерфейс администратора

Проверка состояния (включен или выключен) узла кластера может быть выполнена с использованием протокола *ARP* (*Address Resolution Protocol*). В сети *Ethernet* перед началом передачи данных при известном *IP*-адресе узла необходимо также определить *MAC*-адрес узла. Для определения *MAC*-адреса станции посылается *ARP*-запрос и при получении ответа можно утверждать, что узел подключен к сети и активен (включен). Активным узел будет в режиме, совпадающем с указанным в панели администратора.

Основная функция интерфейса администратора – управление режимами, в которых загружаются узлы. Как было указано, режим работы узла управляется файлом *bootstrap*, загружающим необходимый файл с *TFTP*-сервера. Путем манипуляции этим файлом достигается управление режимами работы узла. Для этого необходимо сохранять все варианты *MBR*-записи для каждого узла.

Важная функция интерфейса администратора состоит в возможности выключения и загрузки узлов в необходимый момент времени.

Выключение узла в *режиме* 1 можно выполнить дистанционно, используя утилиты *Windows Sysinternal* [5]. Команда на перезагрузку может иметь следующий вид

```
psexec.exe -d \\node_name
  shutdown -r -t 00 -f
```

В этой команде *node_name* – имя или *IP*-адрес узла.

Выключение узла в *режиме* 2 выполняется различными методами в зависимости от ОС, которая используется в этом режиме, поэтому у администратора есть возможность указать сценарий для выключения узла в этом режиме.

Сетевые адаптеры узлов кластера должны иметь встроенную функцию *WOL* (*Wake-On-Lan*), использование которой позволяет включить узел дистанционно. Для этого необходимо послать специальную последовательность байтов, инкапсулируемую в *TCP*- или *UDP*-пакет. Последовательность должна состоять из шести байтов со значением *FF* и *MAC*-адреса сетевой карты узла, который повторяется 16 раз. Код функции для включения узла с использованием языка *C#* имеет следующий вид:

```
   string  MAC_ADDRESS = "001A4B4BA1A1";
   UdpClient client = new UdpClient();

   // Создание соединения на широковещательный адрес с номером порта 9 по протоколу UDP
   client.Connect(new IPAddress(0xffffffff), 0x0009);

   if( client. Active )
   this.Client.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast,0);

   int counter = 0;
   byte[] bytes = new byte[1024];

   // В массив записать шесть байт со значением 0xFF в шестнадцатеричном виде
   for (int y = 0; y < 6; y++)
   bytes[counter++] = 0xFF;

   // Добавить к массиву bytes значение MAC-адреса шестнадцать раз
   for (int y = 0; y < 16; y++) {
   int i = 0;
   // Прохождение по всем байтам MAC-адреса (всего их шесть)
               for (int z = 0; z < 6; z++) {
                     bytes[counter++] = byte.Parse(  MAC_ADDRESS.Substring(i, 2),

       NumberStyles.HexNumber);
       // один байт в шестнадцатеричном виде – это два символа
                  i += 2;
   }
   }
   // Отправка массива байт с переменной bytes
   int returned_value = client.Send(bytes, 1024);
```

Как показано в [4], проблема большинства кластерных систем заключается в отсутствии возможности остановки кластера на определенное время, так как при этом нет возможности остановки кластерных вычислений. Такая же проблема наблюдалась и у кластера, построенного на *CCS*. Для решения этой проблемы перед остановкой кластера необходимо все процессы, связанные с вычислениями переводить в режим паузы (*syspend*), а сами узлы – в режим сна (*hibernate*). Эти действия можно выполнить дистанционно с использованием утилиты *Sysinternals*.

Заключение. Предложенная методика проведения кластерных вычислений позволяет решать проблему проведения в организации ресурсоемких расчетов с использованием существующих вычислительных ресурсов без создания *стационарного* кластера. Существенной отличительной особенностью созданной методики является формирование универсального подхода к управлению кластером, позволяющим выполнить приостановку кластерных вычислений с возможностью их возобновления в нужный момент. Разработанное программное обеспечение *UACluster* реализует предложенную методику. В настоящее время осуществляется опытная эксплуатация созданного кластера в Киевском национальном университете имени Тараса Шевченко, давшая положительный результат.

1. *Top* 500 Supercomputers. – http://www.top500.org/
2. *Обчислювальний* кластер Інформаційно-обчислювального центру КНУ ім. Тараса Шевченка. – http://www.cluster.kiev.ua/
3. *Программное* обеспечение *UACluster* – http://codeplex. com/UACluster
4. *Концепція* створення гнучких гомогенних архітектур кластерних систем / С.Д. Погорілий, Ю.В. Бойко, Д.Б. Грязнов та ін. // Проблеми програмування. – 2008. – № 2–3. – С. 84–90.
5. *Windows* Sysintertals. – http://technet.microsoft.com/en-us/sysinternals/default.aspx

Поступила 06.01.2009
Тел. для справок: (044) 521-3347 (Киев)