УДК 004.75

БУЛАХ Б.В.,
ЧЕКАЛЮК В.В.,
ЛАДОГУБЕЦ В.В.,
ФИНОГЕНОВ А.Д.

МОДИФИКАЦИЯ СПОСОБА УЧЕТА БАЛАНСА ЗАГРУЗКИ ПРИ ВЫПОЛНЕНИИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ

В статье рассматриваются вопросы адаптации параллельных программ, использующих библиотеку PVM, для запуска на кластере НТУУ «КПИ» через планировщик PBS. Разработана модификация способа баланса загрузки, учитывающая архитектуру вычислительных узлов кластера. Даны рекомендации по организации блока анализа конфигурации PVM для параллельных программ.

The article deals with the adapting of parallel PVM programs to run on a cluster of NTUU "KPI" with PBS scheduler. A modification of the load balancing mechanism that takes into account the architecture of computing nodes was developed. Recommendations on the organization of the PVM configuration analysis unit for parallel programs were provided.

1. Введение

Библиотека PVM [1], как средство поддержки параллельных вычислений, позволяет использовать параллельные вычисления на наиболее трудоемких этапах решения задач. Наличие разнообразной техники различной архитектуры и производительности, работающей под управлением различных операционных систем, обусловили популярность использования библиотеки PVM в научной среде при объединении гетерогенных вычислительных средств в единую виртуальную многопроцессорную машину.

В статье [2] рассматривались вопросы разработки параллельных алгоритмов в рамках существующих пакетов проектирования на примере адаптации одного из методов блока параметрической оптимизации – метода случайного поиска с уменьшением интервала поиска (СПУИП). В качестве многопроцессорной системы рассматривались три персональных компьютера, объединенных локальной сетью.

С появлением в НТУУ «КПИ» центра суперкомпьютерных вычислений [3], была проведена модификация пакета NetALLTED [4] и сопутствующей параллельной реализации метода СПУИП [5]. Однако данная реализация рассчитана на запуск в составе предварительно настроенной виртуальной машины. При этом количество slave-программ задавалось статически, что было оправдано в виду однотипности узлов.

В случае запуска параллельных программ с использованием планировщика задач PBS, установленном на кластере НТУУ «КПИ», возникает ряд технических особенностей, которые не позволяют напрямую запускать параллельные программы, использующие библиотеку PVM.

2. Постановка задачи

Запуск программ с использованием планировщика PBS, по сравнению с «ручной настройкой» виртуальной машины, отличается тем, что:
– отсутствует возможность интерактивной работы в консоли PVM;
– заранее неизвестны имена узлов, которые будут предоставлены для решения задачи;
– узлы кластера отличаются по архитектуре и производительности процессоров, а также по количеству доступной оперативной памяти;
– при указании в запросе только количества требуемых ядер отсутствует информация о количестве ядер, выделяемых на узле.

Кроме того, структуры данных конфигурации виртуальной машины хранят только сетевые имена хостов (в качестве хостов рассматриваются машины или узлы, которые имеют IP-адрес), их архитектуру, относительную скорость и идентификатор демона pvmd на данном хосте. Эти данные не позволяют на основании лишь конфигурации виртуальной машины определить количество ядер, выделенных на хосте для решения задачи и запуск соответствующего количества slave-программ.

Необходимость запуска параллельных модулей обусловлена функциональностью междисциплинарного комплекса опти-мального математического моделирования в грид-среде с автоматическим формированием и решением уравнений соответствующих математических моделей (НТР № ДП / 310 – 2011 от 26 июля 2011 г., номер государственной регистрации 0111U005434).

Целью статьи является решение вопросов, связанных с запуском и настройкой PVM с использованием планировщика задач, а также модификация алгоритма баланса загрузки и запуска slave-программ с учетом количества выделенных ядер на узле.

3. Конфигурирование среды выполнения

При описании задачи для планировщика наиболее эффективно указывать лишь количество необходимых ядер, т.к. это увеличивает количество возможных конфигураций и уменьшает время простоя задачи в очереди.

Положим, что для решения задачи необходимо p ядер. Планировщик задач может выделить их как на одном узле, так и на нескольких, в зависимости от текущей загруженности системы. Так для кластера НТУУ «КПИ» при запросе p=10 был получен следующий перечень узлов, хранящийся в файле, указанном в переменной окружения `PBS_NODEFILE`:

```
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n003.kpi
n003.kpi
```

Таким образом, планировщик выделил 8 ядер на узле `n004.kpi` и 2 на узле `n003.kpi`.

Каждая строка конфигурационного файла PVM выглядит следующим образом:

```
host <id=n> <sp=value>
```

где `id` – идентификатор (уникальный номер) хоста, `sp` – относительная вычислительная мощность хоста по сравнению с другими хостами в PVM, которая может принимать значение от 1 до 10^6. По умолчанию `sp=1000`. Используя дополнительные ключи можно указать логин на удаленном хосте (если он отличается от значения логина на текущем), месторасположение демона PVM и исполняемых файлов (slave), если пути отличаются от принятых по умолчанию и т.д.

Идентификатор `id` используется для формирования уникальных имен временных файлов демона PVM на каждом хосте в случае общей файловой системы. Номера могут задаваться произвольно, но должны быть уникальными в пределах текущей конфигурации.

При формировании файла конфигурации PVM сохранить информацию о количестве ядер, выделенных на узле, можно, используя часть цифр, отводимых для хранения относительной вычислительной мощности хоста. Т.к. на узле в ближайшее время не ожидается больше 99 ядер, а разброс вычислительной мощности между узлами значительно меньше, чем 1000 раз даже с учетом накладных расходов на пересылку данных, то формат можно представить следующим образом:

$$sp_i = sp_{Ri} \cdot 100 + nslaves_i,\ i = 1(1)nhost$$

где sp_i – относительная вычислительная мощность i-го узла; sp_{Ri} – предварительно рассчитанная относительная мощность i-го узла, который может входить в состав PVM из числа узлов кластера; $nslaves_i$ – количество ядер, выделенных на узле для данной задачи; `nhost` – количество выделенных узлов для данной задачи.

Параметры $nslaves_i$ и `nhost` определяются на основании анализа данных переменной окружения `PBS_NODEFILE`.

sp_{Ri} – можно определить с помощью предварительного решения тестовых задач для каждой архитектуры узла:

$$sp_{Ri} = \frac{T_{Rk}}{\max(T_{Rk})} \cdot 1000$$

где T_{Rk} – время решения тестовой задачи на k-й архитектуре узла.

В составе кластера НТУУ «КПИ» присутствуют 2 типа узлов, доступных локальному пользователю:

– узлы (n001-n044) с 2-мя четырехъядерными процессорами Intel Xeon E5440 @ 2.83ГГц и 8 Гб оперативной памяти;

– узлы (n045-n112) с 2-мя двухъядерными процессорами Intel Xeon 5160 @ 3.00ГГц и 4 Гб оперативной памяти.

Грубую оценку можно также проводить, сравнивая тактовую частоту процессоров:

$$sp_{Ri} = \frac{\varphi_{Rk}}{\max(\varphi_{Rk})} \cdot 1000$$

где φ_{Rk} – тактовая частота процессоров узлов. Для кластера НТУУ «КПИ» оценки относительной вычислительной мощности узлов выглядят следующим образом:

$$sp_{R1} = \frac{2.83}{3.00} \cdot 1000 = 943 \qquad (n001 - n044)$$

$$sp_{R2} = \frac{3.00}{3.00} \cdot 1000 = 1000 \qquad (n045 - n112)$$

Для описанной в пункте 3 тестовой задачи файл конфигурации будет выглядеть следующим образом:

```
n004.kpi id=1 sp=94308
n003.kpi id=2 sp=94302
```

Наиболее оптимальным вариантом представляется создание «карты узлов», т.е. конфигурационного файла, содержащего имена узлов и предварительно рассчитанные относительные вычислительные мощности. В этом случае при перестройке конфигурации кластера, его модернизации или расширении не требуется вмешательства в код программы формирования файла конфигурации для PVM.

Формат файла, который содержит вычислительные мощности узлов (hostmap), может иметь вид, аналогичный файлу конфигурации PVM, например:

```
n001.kpi 943
…
n044.kpi 943
n045.kpi 1000
…
n112.kpi 1000
```

4. Учет баланса загрузки

Учет баланса загрузки и распределение slave-программ по выделенным ядрам на основании строк конфигурационного файла необходимо реализовать в главном процессе параллельной программы

Т.к. выделенные ядра могут принадлежать разным по производительности процессорам, то в качестве алгоритма баланса загрузки (АБЗ) можно воспользоваться [6]. Для работы данного алгоритма необходимо определение следующих параметров:

$$p = \sum_{i=1}^{nhost} sp_i \%100$$

$$sp_{Rk} = sp_i / 100, \qquad k = 1(1)p$$

Рассмотрим условие

$$M \geq p$$

где M – количество однотипных групп расчетов, на которые можно разделить исходную задачу. Первым этапом АБЗ является вычеркивание «медленных процессоров», для которых выполняется условие

$$M \cdot sp_{Ri} \leq 1 \qquad (1)$$

Для узлов, которые могут войти в состав PVM при текущей архитектуре кластера НТУУ «КПИ», соотношение между «самым быстрым» процессором и «самым медленным» процессором таково, что данное условие (1) является неактуальным и шаг вычеркивания «медленных процессоров» можно опустить.

В общем случае для узлов, в состав которых входят многоядерные процессоры, необходимо учитывать следующие особенности:

– из перечня процессоров «вычеркивается» не процессор, а ядро, что требует контроля количества «не вычеркнутых» ядер на узле, которое будет соответствовать количеству запускаемых на узле slave-процессов;

– проверка ядер, принадлежащих одному узлу, на условие исключения должна идти последовательно по всем ядрам узла, начиная с узла с наименьшим количеством выделенных ядер. В этом случае при равных относительных вычислительных мощностях, будет минимизироваться количество задействованных узлов, что уменьшает количество необходимых затрат на установление обмена.

5. Адаптированный механизм выполнения параллельных программ

Запуск параллельной программы осуществляется путем постановки в очередь планировщика задач файла сценария (start.sh), который содержит служебную информацию о количестве запрашиваемых ресурсов, вызов программы генерации конфигурационного файла PVM с учетом полученных от планировщика имен узлов, выбор мастер-узла для запуска PVM, и вызов непосредственно требуемой параллельной программы.

В случае если пользователю предоставляется возможность задавать количество требуе-

мых процессоров для параллельного решения, то файл сценария также можно генерировать с подстановкой количества ядер (N) указанных пользователем.

Общая схема запуска представлена на рис.1.

6. Выводы

Разработанная схема выполнения параллельных программ, использующих библиотеку PVM, обеспечивает возможность корректного конфигурирования виртуальной машины при отложенном запуске через планировщик задач PBS.

Использование «карты» вычислительной мощности узлов, а также механизм учета количества предоставленных на узле ядер для решения, позволяет осуществлять баланс загрузки не только в случае использования кластера НТУУ «КПИ», но и при создании параллельной виртуальной машины из разнородных ЭВМ.

Дальнейшее направление работ связано с предоставлением возможности включения параллельной версии пакета ALLTED в маршрут проектирования в рамках проекта создания междисциплинарного комплекса математического моделирования в грид-среде.

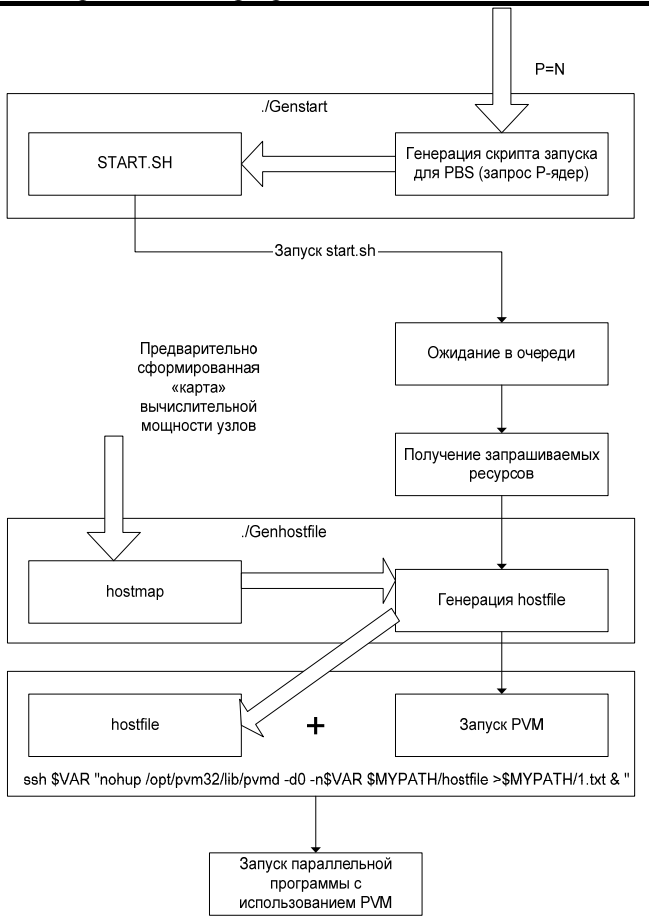

Рис. 1 – Подготовка выполнения параллельной программы

Список литературы

1. Официальный сайт разработчиков библиотеки PVM. – Режим доступа : http://www.csm.ornl.gov/pvm/pvm_home.html. – Дата доступа : 19.02.2012.
2. Ладогубец В.В. Особенности реализации параллельных алгоритмов для однопроцессорных пакетов / Финогенов А.Д., Ладогубец В.В. // Электроника и связь. – 2005. – № 25. – С. 95–98.
3. Официальный сайт центра суперкомпьютерных вычислений НТУУ «КПИ». – Режим доступа : http://hpcc.org.ua. – Дата доступа : 19.02.2012.
4. Сайт разработчиков САПР ALLTED. – Режим доступа : http://allted.kpi.ua. – Дата доступа : 19.02.2012.
5. Ладогубец В.В. Адаптация параллельного алгоритма СПУИП для кластера НТУУ «КПИ» / Ладогубец В.В., Крамар А.В., Финогенов А.Д. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – 2008. – № 48. – С. 99–103.
6. Ладогубець В.В. Алгоритм оптимального балансу загрузки / Ладогубець В.В. // Электроника и связь. – 1999. – Т. 1, № 6. – С. 74–76.