УДК 62-82:658.512.011.56

ЛАДОГУБЕЦ В.В.,
КРАМАР А.В.,
ФИНОГЕНОВ А.Д.

АДАПТАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА СПУИП ДЛЯ КЛАСТЕРА НТУУ «КПИ»

Описаны проблемы адаптации параллельного алгоритма СПУИП, реализованного в составе пакета Allted, для работы на кластере НТУУ «КПИ» и пути их решения. Выполнена модификация алгоритма баланса загрузки для использования возможностей аппаратной части. Приведены результаты решения тестовой задачи и сделаны выводы об эффективности предложенной реализации.

The problems of adaptation of SPUIP parallel algorithm implemented in ALLTED software to work on NTUU KPI's cluster and the ways to solve them are described. To make use of hardware possibilities, a modification of the load balance algorithm has been done. The results of a test task solution are given, and the conclusions regarding efficiency of the suggested realization are made.

С развитием многопроцессорных вычислительных систем роль параллельных вычислений неуклонно растет. Практически, параллельные вычисления стали единственным средством, позволяющим использовать современную компьютерную технику. Однако с усовершенствованием техники появилась необходимость адаптации разработанных параллельных алгоритмов к текущим реалиям.

Так в [1] был описан параллельный алгоритм оптимизации метода случайного поиска с уменьшением интервала поиска (СПУИП) [2], реализованного в составе ППП Allted [3]. Целью данной статьи является описание проблем и их решение при адаптации пакета и алгоритма для работы в составе кластера НТУУ «КПИ» [4].

В первую очередь рассмотрим отличия мультипроцессорной вычислительной среды от используемой ранее (рис.1).

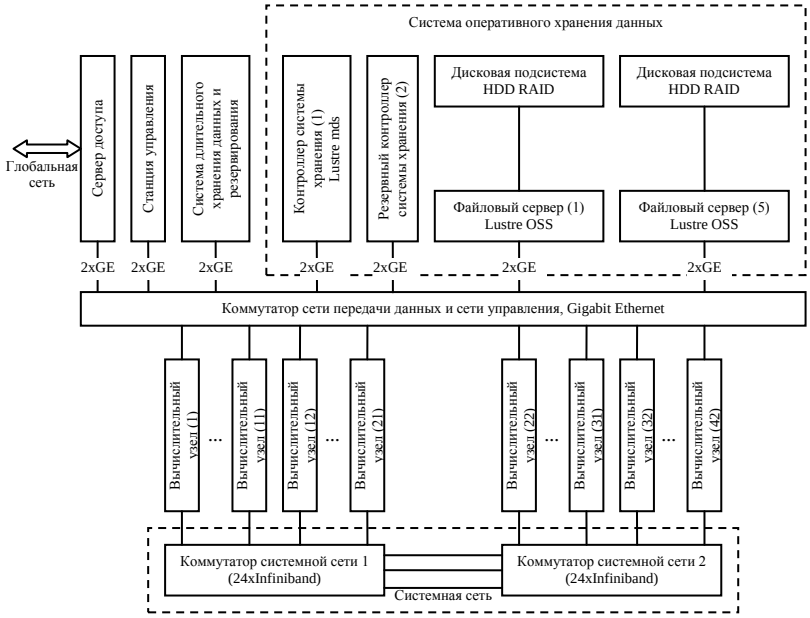


Рис.1. Структура кластера НТУУ «КПИ»

Каждый вычислительный узел кластера представляет собой два 2-х ядерных процессора Intel Xeon 5160 2x3.0 ГГц. Системная сеть построена на технологии InfiniBand/Gigabit Ethernet. В качестве операционной системы используется одна из версий ОС UNIX. Необходимо также отметить, что процессоры кластера имеют 64-х разрядную архитектуру.

Таким образом, можно выделить следующие этапы, необходимые для реализации алгоритм СПУИП в составе Allted на кластере НТУУ «КПИ»:

1)Адаптировать 32-х разрядный пакет для работы на 64-х разрядной платформе

2)Установить библиотеку PVM [5] и адаптировать ее к правилам безопасности, принятым на кластере.

3) Проанализировать конфигурацию вычислительной сети и адаптировать баланс загрузки.

Адаптация пакета Allted

Проблемы адаптации пакета в первую очередь сводятся к выбору одного или набора компиляторов, которые дают возможность произвести корректную сборку исходных файлов пакета. Исходные файлы пакета написаны на языке FORTRAN, однако многие дополнения к пакету описывались в более позднее время и написаны на языке Си (в том числе и реализация параллельного алгоритма СПУИП). Компиляция производилась в два этапа: в первом случае с помощью компилятора f2c[6] из исходных кодов на FORTRAN получались объектные файлы, на втором из файлов Си получались объектные файлы с помощью компилятора gcc 3.4.6. Компиляция исполняемых файлов также производилась с помощью компилятора gcc. Для эмуляции 32-х разрядной среды в команды компиляции была включена опция «–m32».

Установка библиотеки PVM.

При установке PVM также необходимо установить опцию эмуляции «-m32» для совместимости с пакетом. Кроме этого необходимо учитывать следующую особенность PVM. Передача сообщений происходит с использованием утилиты rsh (Remote shell). Ее недостатком является незащищенность, поэтому на большинстве систем в целях обеспечения безопасности ее использование ограничено. Разработчиками была предусмотрена такая возможность, и для передачи можно воспользоваться стандартным протоколом – ssh (secure shell), который обеспечивает шифрование передаваемых данных. Для его использования необходимо указать путь к файлу ssh – в файлах конфигурации PVM. Недостатком использования ssh является увеличение времени обмена между процессами.

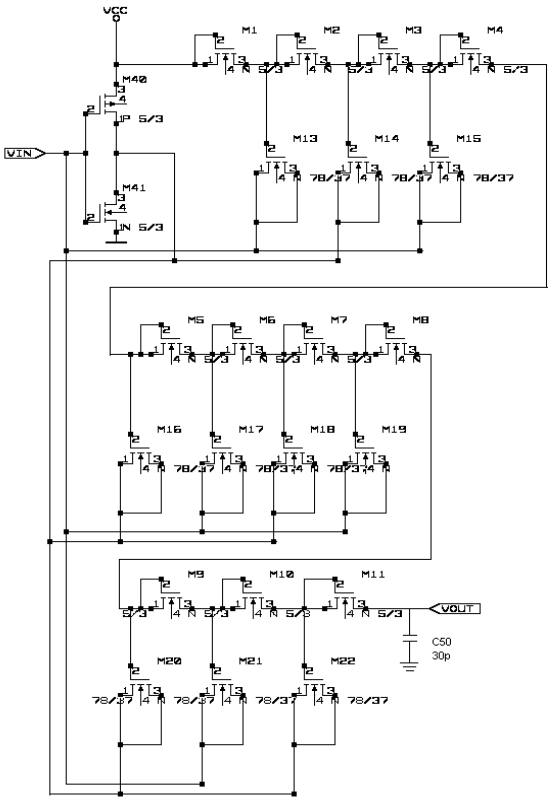
Адаптация алгоритма баланса загрузки

Описанный в [7] алгоритм баланса загрузки не предусматривал наличие в составе кластера многопроцессорных (а тем более многоядерных) вычислительных систем. Т.к. каждый узел кластера суммарно содержит 4-е вычислительных ядра, которые одинаковы по своим параметрам, то очевидно, что возможен запуск до 4-х slave процессов на одном узле одновременно. Т.к. относительные мощности всех ядер равны, то каждый slave процесс будет выполнять:

$$n_i = \frac{N}{i};$$

где N – количество вычислений целевой функции выполняемых узлом, i – количество запускаемых slave-процессов на одном узле.

В качестве теста использовался пример, который аналогичен рассмотренному в [1]: задача оптимизации схемы подкачки рис.2. с увеличенным диапазоном изменения варьируемых параметров (время решения такой задачи на «машина1», которая использовались в экспериментах из [1] составляет более 1 часа, что делало весьма затруднительным проведения «чистого» эксперимента в виду загруженности техники). Напомним, что для этой схемы необходимо так поднять выходное напряжение (VC80) что бы в момент времени 2 msec напряжение ровнялось бы 15 V. В качестве варьируемых параметров можно синхронно изменять W, L, и толщину окисла (TOX) транзисторов M13-M22. Время моделирования – 3ms. В качестве моделей транзисторов используется BSIM1 модель (аналогичная Level13 в HSPICE).



Параллельная виртуальная машина конфигурировалась с различным количеством узлов (от 1 до 5) и различным количеством запускаемых задач на каждом узле (от 1 до 4). Результаты времени расчетов (приведены в секундах), а также соответствующие им коэффициенты ускорения (K_y) представлены в табл. 1.

Отметим, что в качестве времени решения указывается полное время работы пакета, а не время работы алгоритма. Т.е. полученные временные значения включают в себя не только затраты на обмен данными, но и время трансляции и анализа файла задания, формирование математической модели и запись файла результатов. Во всех экспериментах относительные мощности всех узлов были равны.

Рис.2. Схема подкачки

Таблица 1. Решение задачи оптимизации при различных конфигурациях PVM*

Количество запускаемых процессов	Количество узлов в составе PVM				
	N=1 (K_y)	N=2 (K_y)	N=3 (K_y)	N=4 (K_y)	N=5 (K_y)
1	1777 (1)	916 (1.94)	639 (2.78)	493 (3.60)	405 (4.39)
2	912 (1.95)	486 (3.66)	345 (5.15)	275 (6.46)	231 (7.69)
3	643 (2.76)	363 (4.90)	254 (7.00)	208 (8.54)	180 (9.87)
4	484 (3.67)	273 (6.51)	201 (8.84)	186 (9.55)	145 (12.26)

* - приведенное время является лучшим из 10 испытаний при каждой конфигурации параллельной виртуальной машины.

Приведенные результаты свидетельствуют о высокой эффективности реализованного алгоритма. Коэффициент ускорения для параллельной части алгоритма будет равен:

$$K_y = \frac{T_{послед.} - T_{форм.}}{T_{парал.} - T_{форм.}} \qquad (1)$$

где $T_{послед.}$ – время решения задачи пакетом, с использованием последовательного алгоритма; $T_{парал.}$ – время решения задачи пакетом, с использованием параллельного алгоритма; $T_{форм.}$ – время, включающее трансляцию файла задания, установки начальных значений, формирования математической модели и выходного файла результатов.

Время, затрачиваемое на последовательную часть работы пакета, составляет для данной задачи $T_{форм} \approx 40$ секунд. Например, для конфигурации параллельной виртуальной машины с пятью узлами и запуском 4-х slave-процессов на каждом, в соответствии с (1), получим:

$$K_y = \frac{1777 - 40}{145 - 40} = \frac{1737}{105} \approx 16.54$$

что составляет 82.7% от теоретически возможного ускорения.

Рост коэффициента ускорения и времени решения для некоторых из конфигураций может носить нелинейный характер. Это связано с тем обстоятельством, что в ряде случаев невозможно получить равномерную загрузку для всех slave-процессов. Так в тестовой задаче размер выборки для алгоритма СПУИП был равен N=200. Соответственно, что в ряде случаев, например для эксперимента, с 3 узлами и 3 запускаемыми slave-процессами, время решения будет определяться временем работы процесса с наибольшей загрузкой. Суммарное количество рассчитанных целевых функций равнялось 790.

При анализе времени решения задачи необходимо учитывать, что эксперименты проводились на узлах кластера, которые могли быть задействованы для решения задач других пользователей. Часто, получаемые результаты отличались от лучших на 30-70%.

Необходимо отметить еще одну возможность сокращения времени решения. Предложенный алгоритм разрабатывался для случая систем с разделенной оперативной памятью, поэтому запуск slave-процессов и обмен данными с ними по аналогии с [1] производился для каждого slave-процесса в отдельности (рис.3).

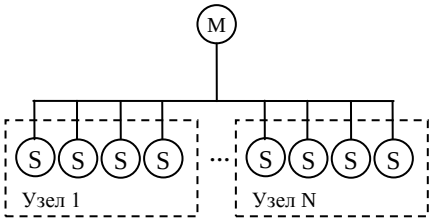


Рис.3 Организация обмена «master – каждый slave»

Т.к. на кластере НТУУ «КПИ» каждый узел имеет общую оперативную память, то, возможна организация взаимодействия по типу дерева, когда master-процесс производит запуск только одного slave-процесса на каждом узле (slave-процесс первого уровня), а тот в свою очередь запускает slave-процессы второго уровня.

Такая организация обмена предпочтительней для тех конфигураций мультипроцессорных вычислительных систем, где временные затраты на передачу данных значительны. Это характерно для систем, где используются низкоскоростные сети или сама сеть загружена. Уровней дерева может быть и больше, однако в этом случае необходимо учитывать необходимость передачи данных для уровней без общей памяти, что отразится на времени решения.

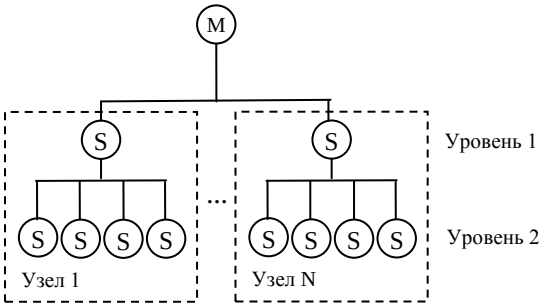


Рис.4 Организация обмена в виде дерева

Анализ времени передачи и приема данных на кластере показал, что основные затраты времени связаны не с передаваемым объемом данных, а временем согласования процессов. Напомним, что в соответствии с [1] при запуске slave-процессов осуществляется передача начальных данных (для данной задачи объем данных составлял $\approx 300Кб$). В дальнейшем передаются только данные, необходимые для расчетов значений целевых функций ($< 5Кб$ для задачи с двумя slave-процессами). В этом случае организация передачи между master- и slave-процессами в виде дерева может негативно сказаться на общем времени решения из-за необходимости обеспечения двух уровней связывания процессов.

Преимуществом организации обмена в виде дерева может стать возможность использование, отличных от других уровней, средств поддержки параллельных вычислений.

Выводы

Параллельный алгоритм СПУИП, реализованный в составе пакета Allted, показал высокую эффективность при реализации на кластере НТУУ «КПИ». Результаты тестирования показали постепенное замедление роста коэффициента ускорения, что говорит о достижении предела эффективного увеличения количества процессов для тестовой задачи. Конфигурация взаимодействия master- и slave-процессов должна выбираться в соответствии, как с анализом технических характеристик мультипроцессорной вычислительной системы, так и анализом особенностей реализуемого алгоритма.

Список литературы:

1. *Ладогубец В.В.*, Финогенов А.Д.: «Особенности реализации параллельных алгоритмов для одно-процессорных пакетов», Киев, Электроника и связь, №25, 2005, с.95-98.
2. *А.И. Петренко*, В.В. Ладогубец, В.В. Чкалов Оптимальное схемотехническое проектирование в машиностроении. - Киев,1989. – 164 стр.
3. *Petrenko A.*, Ladogubets V., Tchkalov V., Pudlowski Z.ALLTED - a computer-aided engineering system for electronic circuit design.(монографія) Melbourne: UICEE, 1997
4. Официальный сайт кластера НТУУ «КПИ»: http://hpcc.org.ua
5. Официальный сайт PVM: http://netlib.org/pvm
6. *S. I. Feldman*, David M. Gay, Mark W. Maimone, and N. L. Schryer, "A Fortran to C Converter," AT&T Bell Laboratories technical report, 1990
7. *Ладогубець В.В.* Алгоритм оптимального балансу загрузки Электроника и связь, №6, 1999