Метод использования параллельных вычислений при обработке большого количества однотипных файлов

Е.Н. Кинчаров

Поволжский государственный университет телекоммуникаций и информатики, Самара
kincharov@ya.ru

В статье рассмотрена применимость параллельных вычислений при однотипной обработке больших массивов файлов.

Несмотря на высокий темп развития вычислительной техники, остаётся круг задач для которых необходимы вычислительные мощности которые могут обеспечить только суперкомпьютеры. С другой стороны, в современном мире на данный момент накопился огромный парк машин по производительности не уступающий машинам из международного рейтинга суперкомпьютеров "Top500" 1993 года. Для решения проблемы нехватки вычислительной мощности применяются 2 метода:

1. применение суперкомпьютеров;

2. создание из имеющегося парка компьютеров кластера и запуск на нём имеющихся задач.

Оба метода имеют свои достоинства и недостатки. Главный минус второго метода - сложность организации взаимодействия между компьютерами различных операционных систем (ОС) и архитектур - до недавнего времени не имел эффективного решения. С появлением ОС Linux, ядро которой имеет реализацию под большинство современных микропроцессоров, эта проблема перестала быть актуальной. Портирование программы, написанной для одной архитектуры, можно выполнить простой перекомпиляцией.

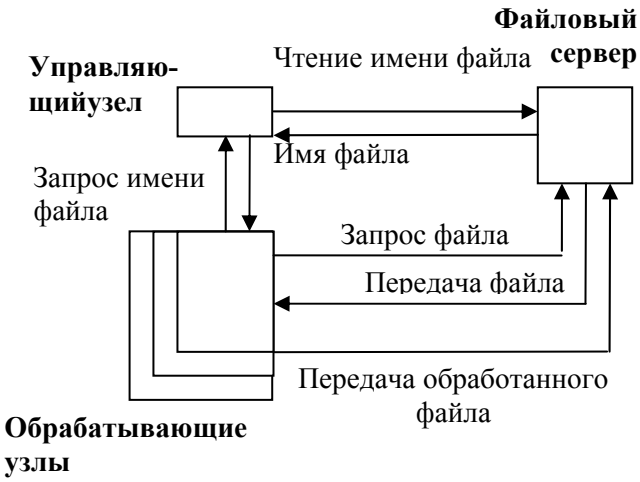

Рисунок 1. Блок-схема одной итерации на кластере

В таблице 1 перечислены шаги, производимые при выполнении одной итерации. Временем выполнения шагов с 1 по 6 можно пренебречь.

Время x зависит от времени ожидания задания в очереди, скорости сетевого интерфейса файлового сервера и скорости HDD.

В современных ОС скорость чтения файла по сети зависит от пропускной способности сетевого интерфейса, а также от времени, затраченного на его поиск на HDD (шаг 7).

Таблица 1. Шаги, выполняемые при одной итерации

№ шага	Источник	Ресурс	Время
1	Обрабатывающая ЭВМ	Сетевой интерфейс	~0
2	Управляющий ПК	Сетевой интерфейс	~0
3	Файловый сервер	HDD	~0
4	Файловый сервер	Сетевой интерфейс	~0
5	Управляющий ПК	Сетевой интерфейс	~0
6	Обработчик	Сетевой интерфейс	~0
7	Файловый сервер	Сетевой интерфейс, HDD	x
8	Обработчик	CPU	y
9	Обработчик	Сетевой интерфейс, HDD	z

При увеличении числа обрабатывающих узлов пропускная способность сети будет делиться между ними, т.е. чем больше машин участвуют в обработке, тем больше вероятность одновременного считывания файлов.

Предположим, что пропускная способность сетевого интерфейса - *a* Мбит/сек. В среднем файлы имеют размер *b* Мб. Отсюда время передачи при незагруженном канале *b/a* сек. Время обработки положим равным *c* сек. Если предположить, что итоговый файл по размеру такой же как и исходный (в результате обработки размер файла меняется незначительно), то получим, что время обработки одного файла *t* будет равным

$$t = \frac{2b}{a} + c \qquad (1)$$

При этом загрузка сетевого интерфейса (наиболее активно используемого узла) μ составит

$$\mu = \frac{\frac{2b}{a}}{\frac{2b}{a} + c} = \frac{2b}{2b + ca} \qquad (2)$$

где *b* - размер файла,

c - время обработки.

Предположим, что передача файла по интерфейсу в любом направлении блокирует канал, тогда при увеличении числа обрабатывающих ЭВМ получаем удвоение загруженности канала

$$\mu = \frac{n * 2b}{2b + ca} \qquad (3)$$

Ввиду того, что загруженность сетевого интерфейса не может быть больше единицы (иначе возможно будут теряться пакеты), получаем неравенство

$$1 > \frac{n * 2b}{2b + ca} \qquad (4)$$

Из него для случая с блокирующимся каналом число компьютеров должно быть не более

$$n < 1 + \frac{ca}{2b} \qquad (5)$$

Необходимый инструментарий для проведения эксперимента

Для работы кластера в Linux необходимо:

1. Наличие соединения по протоколу TCP/IP между узлами кластера.
2. Наличие операционной системы Linux с ядром версии не раньше 2.2 на всех компьютерах, участвующих в обработке информации.
3. Наличие Mpich (интерфейса для распределённой обработки информации), ssh (демона Secure Shell на главном компьютере и клиентов ssh на подчинённых компьютерах).
4. Наличие демона NFS или SMB на центральном компьютере и соответствующих клиентов на подчинённых компьютерах.
5. Наличие утилит для выполнения действий над обрабатываемыми файлами.

В качестве примера возьмём задачу обработки большого массива фотографий в формате JPEG. Будем использовать сдвиг изображения на 45 градусов против часовой стрелки. Данную задачу можно разбить на 3 подзадачи:

1. Декодирование первоначального изображения (перевод его в формат BMP)
2. Сдвиг полученной матрицы
3. Упаковка результирующего изображения в формат JPEG.

Протестируем решение на одном компьютере:

Тестовая конфигурация:

1. Процессор AMD Athlon XP 1700+ (реальная частота 1466 МГц), производительность 2908.16 BogoMIPS,
2. 256 Мб ОЗУ DDR DIMM.
3. HDD 160 Гб, Samsung SP1604N. Файловая система ext3.
4. ОС Mandriva Linux 2006 ядро 2.6.12-12mdk.

Тестирование проводится на массиве из 16 фотографий размером 2048x1536 в формате JPEG.

Результаты тестирования на одном компьютере представлены в таблице 2.

Таблица 2. Результат обработки группы файлов

№ файла	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Время, с	4,62	5,24	5,73	6,09	5,22	4,17	4,19	4,15	4,12	5,8	5,27	4,22	4,23	4,17	4,23	5,37

В результате расчетов получено, что среднее время обработки одного изображения 4,8 сек, суммарное время 76,82 сек.

Исходные файлы имеют размер в среднем 650 Кб (5200Кбит) каждый, то есть по сети с пропускной способностью 100 Мбит/с они будут передаваться за 0,1 с.

Предположим, в сети имеется 2 подчинённых компьютера, значит передача двух файлов будет длиться примерно 1/5 сек, обработка около 4,8 сек, передача файла обратно 1/5 сек. Таким образом, в сумме один цикл будет продолжаться приблизительно 5 сек. В итоге получаем, что при использовании трёх компьютеров время обработки 16 файлов составляет около 40 сек.

Если допустить, что используется сеть с пропускной способностью 10 Мбит/с, то передача двух файлов займёт в одном направлении 1 сек, т. е. время, затраченное на обработку двух файлов, составит примерно 7 сек. Суммарное время обработки в этом случае будет равно 56 сек, что меньше времени обработки без применения кластера.

Результаты моделирования

Расчет выполнялся для системы с пропускной способностью сетевого интерфейса файлового сервера 10 Мбит (полудуплекс), размер обрабатываемых файлов 650 Кб. Среднее время обработки одного файла без оптимизации составило 4,8 сек.

Рассмотрим работу вычислительных узлов с производительностью *n* операций в секунду, тогда производительность всего кластера будет равна числу процессоров, умноженному на n. Производительность при каждом измерении различается, т. е. наступит момент, когда в данных условиях задачи суммарная производительность не будет повышаться с повышением числа узлов.

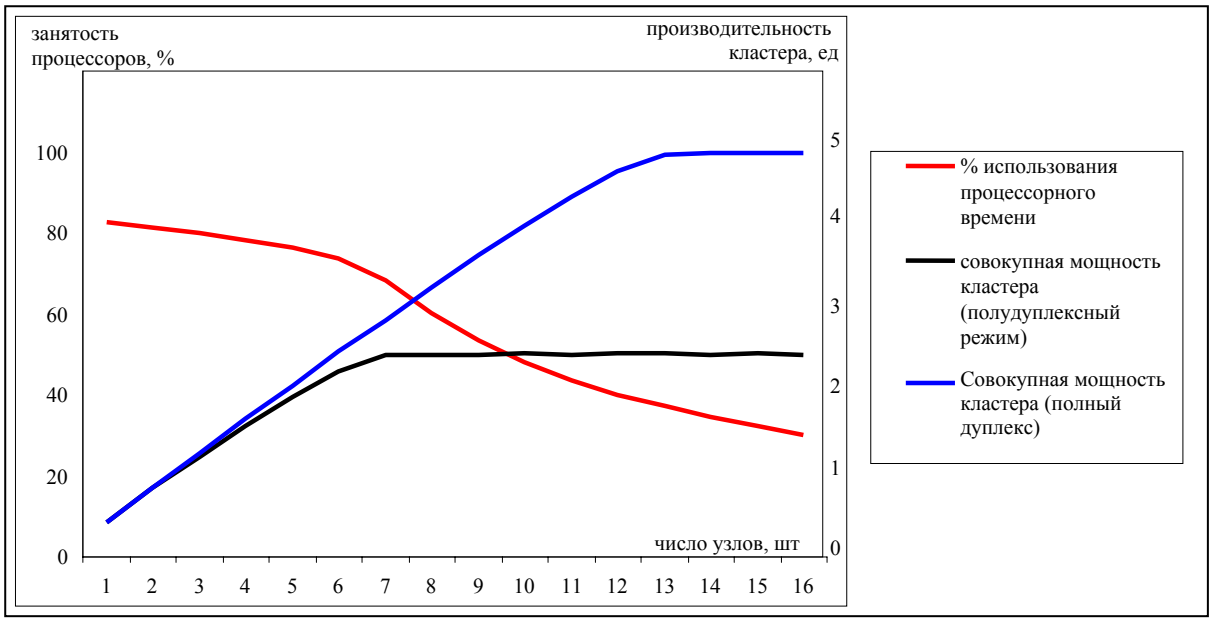

Рисунок 2. График зависимости времени обработки от числа узлов кластера.

Таблица 3. КПД идеального кластера в зависимости от числа узлов.

Число компьютеров	Занятость процессоров, %	КПД, %	Производительность вычислительного кластера, ед
1	82,8	100	0,828
2	81,4	97	1,628
3	79,9	94	2,397
4	78,2	92	3,128
5	76,3	87	3,815
6	73,5	86	4,41
7	68,3	81	4,781
8	60,2	70	4,816
9	53,5	63	4,815
10	48,2	57	4,82

В таблице 3 представлены данные, полученные из расчетов модели в среде GPSS. Процент занятости процессоров показывает, какую часть времени процессоры простаивают. К примеру, если 2 процессора работают 80% времени, то суммарная производи-

тельность составляет 80%, т. е. 20 % времени каждый процессор простаивает (или один простаивает 40% времени, а второй работает со стопроцентной загрузкой).

При увеличении числа процессоров больше 12, максимальное число одновременно занятых заданий остается равным 12, т. е. использование для данной задачи числа компьютеров более 12 нецелесообразно. Максимально достигнутая производительность равна 4,82 ед. Полученный кластер по производительности соответствует 5 компьютерам работающим автономно (без затрат времени на коммуникацию узлов кластера).

Моделирование проводилось в среде GPSS – признанном лидере среди программ для моделирования объектов теории массового обслуживания.

Исходный текст на языке GPSS:

```
10   GENERATE     ,,,100
20   QUEUE    ReadQ
21   COMP     STORAGE 15  //Число компьютеров
22   CPU1     STORAGE 15  //Число компьютеров
23   ENTER    COMP
25   SEIZE    NETIN
30   ADVANCE      53,1
35   RELEASE NETIN
40   DEPART   ReadQ
45   ENTER    CPU1
50   ADVANCE      511,99
55   LEAVE    CPU1
60   QUEUE    WriteQ
70   SEIZE    NETIN
80   ADVANCE      53,1
90   RELEASE NETIN
100  DEPART   WriteQ
105  LEAVE    COMP
110  TERMINATE 1
```

В результате статистического моделирования теоретического эксперимента на идеальном кластере получается почти двукратный прирост производительности. При запуске задачи на реальном кластере этот результат будет несколько хуже, однако увеличение числа обрабатываемых файлов и использование большего количества компьютеров, позволяет повысить эффективность исследуемого метода.

Литература

1. М. Митчелл “Программирование для Linux. Профессиональный подход” М. 2003
2. Д. Элджер “Библиотека программиста C++”. М.:"Питер", 2001
3. Н. Джосьютис “C++. Стандартная библиотека”. М.:"Питер", 2004.
4. Методические указания по выполнению практических работ по дисциплине “Параллельное программирование”, под общей редакцией д.т.н., проф. Солодянникова Ю.В. ПГАТИ, Самара 2004.