Оптимизация выполнения параллельных MPI-программ на мультикластерных вычислительных системах

М. Г. Курносов
Сибирский государственный университет
телекоммуникаций и информатики, Новосибирск

Введение

Одной из основных тенденций современной индустрии высокопроизводительной обработки информации является построение масштабных пространственно-распределенных вычислительных и GRID-систем. В архитектурном плане распределенная вычислительная система (ВС) представляется как множество элементарных машин (ЭМ), взаимодействие между которыми осуществляется через коммуникационную среду [1].

Коммуникационные среды современных распределенных вычислительных систем (ВС) имеют иерархическую организацию, каждый уровень которых характеризуется своими значениями показателей производительности. Например, в GRID-системах и пространственно-распределенных мультикластерных ВС нулевой уровень коммуникационной среды — сеть связи между вычислительными кластерами, первый уровень — сеть связи внутри кластеров, второй уровень — среда доступа процессоров вычислительного узла к общей памяти.

В настоящее время основным подходом к организации параллельных вычислений на таких системах является использование библиотек стандарта MPI. Параллельная программа, разработанная в модели передачи сообщений, может быть представлена информационным графом, вершинам которого соответствуют параллельные ветви программы, а ребрам — коммуникации между ними.

Многие параллельные программы характеризуются неоднородным информационным графом. Неоднородность выражается в объемах данных, передаваемых между ветвями, размерах используемых сообщений, а также в количестве смежных ветвей у каждого процесса.

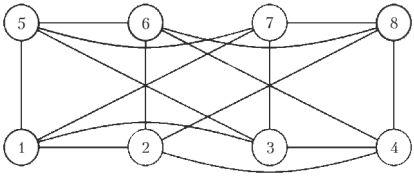


Рис. 1. Информационный граф теста HPL (NP=8, PMAP=1, BCAST=5)

На рис. 1 приведен информационный граф теста HPL (High-Performance Linpack), реализующего LU-разложение для решения системы линейных алгебраических уравнений. Время выполнения указанного теста на 4 узловом кластере, укомплектованном двухпроцессорными узлами на базе двухъядерных процессоров Intel Xeon 5150 и сетью связи стандарта Gigabit Ethernet, зависит от распределения ветвей по процессорным ядрам — распределения рангов MPI-процессов по процессорным ядрам. Например, время выполнения теста на указанном кластере с распределением ветвей (1, 2, 3, 4) (5, 6, 7, 8) ( ) ( ) составляет 118 с (44 GFLOPS, скобками обозначены узлы, а числа — номера ветвей). Очевидно, что целесообразнее распределять ветви так, чтобы интенсивно взаимодействующие ветви передавали сообщения через быстрые каналы связи, например, через общую память узла. Так время выполнения теста HPL с распределением ветвей (1, 3, 5, 7) (2, 4, 6, 8) ( ) ( ) составляет 100 с (53 GFLOPS).

Поэтому при выполнении параллельных MPI-программ на (мульти)кластерных ВС целесообразно учитывать производительности каналов связи коммуникационных сред и структуры информационных графов задач.

Задача оптимального назначения ветвей параллельной программы на процессорные ядра распределенной ВС с целью минимизации времени выполнения приложения ставится как задача дискретной оптимизации.

Постановка задачи

Пусть распределенная ВС состоит из H подсистем, объединенных каналами связи. Введем обозначения: $S=\{1, 2, ..., H\}$ — множество подсистем образующих распределенную ВС; n_i — количество процессорных ядер, входящих в состав подсистемы $i \in S$; N — количество процессорных ядер, входящих в распределенную

ВС; ω_i — производительность процессорного ядра i-ой подсистемы ($[\omega_i]$ = FLOPS, i ∈ S).

Коммуникационная среда i-ой подсистемы допускает представление в виде дерева высоты h_i. Для каждого элемента на уровне $k \in \{1, 2, ..., h_i - 1\}$ задано количество n_{ik} его дочерних элементов. Дочерними элементами нулевого уровня являются вычислительные подсистемы. Далее считаем $n_{i0} = H$.

Каждый уровень $k \in \{1, 2, ..., h_i\}$ коммуникационной среды i-ой подсистемы характеризуется своими значениями показателей производительности: $l_i^{(k)}(a,b,m)$ — зависимость латентности канала связи между элементами a и b на уровне k от размера m передаваемого сообщения ($a,b \in \{1,2,...,n_{ik}\}$, $\left[l_i^{(k)}(a,b,m)\right]$ = с). Аналогично $b_i^{(k)}(a,b,m)$ — пропускная способность канала связи ($\left[b_i^{(k)}(a,b,m)\right]$ = бит/с), а $l^{(0)}(a,b,m)$ и $b^{(0)}(a,b,m)$ определяют производительность каналов связи нулевого уровня коммуникационной среды.

Обозначим через $C = \{c_1, c_2, ..., c_N\}$ множество процессорных ядер, входящих в распределенную ВС. За g_r обозначим номер подсистемы, которой принадлежит процессорное ядро с номером $r \in \{1,2,...,N\}$. Каждый элемент $c_r = \left(e_0^r, e_1^r, ..., e_{h_{g_r}-1}^r\right)$, $c_r \in C$ определяет координату размещения процессорного ядра в распределенной ВС (номер кластера, узла, процессора и т. д., см. рис. 2).

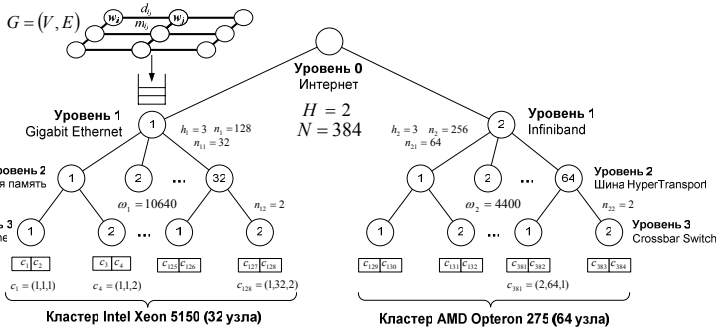


Рис. 2. Пример иерархической организации коммуникационной среды

Пусть процедура назначения ветвей параллельной программы на процессорные ядра осуществляется с подсистемы $s \in S$, а поступившая на выполнение параллельная программа представлена

ее информационным графом $G=(V,E)$. Где $V=\{1,2,...,M\}$ — множество параллельных ветвей программы; $E\subset V\times V$ — множество информационно-логических связей между ее параллельными ветвями. Обозначим: w_i — количество элементарных операций (арифметических и логических) выполняемых ветвью $i\in V$; d_{ij} — объем данных передаваемый по ребру $(i,j)\in E$ за время выполнения параллельной программы ($\lceil d_{ij} \rceil$ = байт); m_{ij} — средний размер сообщения передаваемого по ребру $(i,j)\in E$ за одну операцию приема/передачи ($\lceil m_{ij} \rceil$ = байт); z — размер файла и данных параллельной программы ($[z]$ = байт).

Требуется найти инъективную функцию $f: V\to C$, ставящую в соответствие ветвям параллельной программы процессорные ядра распределенной ВС. Искомая функция допускает представление в виде вектора:

$$X=\{x_{ij}: i\in V, j\in C\},\ x_{ij}=\begin{cases}1, & \text{если } f(i)=j;\\ 0, & \text{иначе.}\end{cases}$$

Качество назначения ветвей параллельной программы на процессорные ядра будем оценивать ожидаемым временем ее обслуживания, которое складывается из времени доставки программы на процессорные ядра и времени выполнения ветвей на них.

Время взаимодействия $t^{(k)}(i,j,p,q)$ параллельных ветвей i и j будем оценивать через латентность и пропускную способность канала связи между процессорными ядрами p и q, на которые они назначены:

$$t^{(k)}(i,j,p,q)=\begin{cases}l^{(0)}\left(e_0^p,e_0^q,m_{ij}\right)\cdot\dfrac{d_{ij}}{m_{ij}}+\dfrac{d_{ij}}{b^{(0)}\left(e_0^p,e_0^q,m_{ij}\right)}, & \text{если } k=0,\\ l_{g_p}^{(k)}\left(e_k^p,e_k^q,m_{ij}\right)\cdot\dfrac{d_{ij}}{m_{ij}}+\dfrac{d_{ij}}{b_{g_p}^{(k)}\left(e_k^p,e_k^q,m_{ij}\right)}, & \text{иначе,}\end{cases}$$

где k — номер уровня, являющийся ближайшим общим предком для ядер p и q.

Учитывая инъективность функции f, получаем задачу (1)—(4) оптимального назначения ветвей параллельной программы на процессорные ядра распределенной ВС с целью минимизации времени ее выполнения.

$$F(X)=\max_{i\in V}\left\{\sum_{j=1}^{N}x_{ij}\cdot t^{(0)}\left(s,g_j,z\right)\right\}+$$
$$+\max_{i\in V}\left\{\sum_{j=1}^{N}x_{ij}\cdot\frac{\varpi_i}{\omega_{g_j}}+\sum_{j=1}^{M}\sum_{p=1}^{N}\sum_{q=1}^{N}x_{ip}\cdot x_{jq}\cdot t^{(k(p,q))}\left(i,j,p,q\right)\right\}\to\min_{(x_{ij})} \quad (1)$$

при ограничениях:

$$\sum_{j=1}^{N}x_{ij}=1,\ \ i=1,2,...,M \quad (2)$$

$$\sum_{i=1}^{M}x_{ij}\le 1,\ \ j=1,2,...,N \quad (3)$$

$$x_{ij}\in\{0,1\},\ \ i\in V, j\in C \quad (4)$$

Ограничения (2), (4) гарантируют назначение каждой ветви параллельной программы на единственное процессорное ядро, ограничения (3) обеспечивают назначение на ядро не более одной ветви.

Задача (1)—(4) представляет собой задачу дискретной оптимизации, мощность множества D допустимых решений которой оценивается величиной $|D|=A_N^M=N!/(N-M)!$. Для большемасштабных распределенных ВС отыскание точного решения связано со значительной вычислительной сложностью. Учитывая данное обстоятельство, целесообразно применение приближенных методов решения задачи.

Методы решения

На основе метаэвристики имитации отжига (Simulated Annealing) [2, 3] разработан стохастический алгоритм PTA_SA (Parallel Task Allocation Simulated Annealing), позволяющий отыскивать приближенные решения задачи за приемлемое время. Общая схема алгоритма выглядит следующим образом. На первом шаге алгоритма строится начальное допустимое решение $x^{(0)}\in D$. На k-ом шаге имеется текущее решение $x^{(k)}\in D$ и наилучшее на данный момент $x^*\in D$. Шаг начинается с выбора случайным образом соседнего решения $y\in N\left(x^{(k)}\right)$ из окрестности

текущего. Если значение целевой функции от соседнего решения меньше значения целевой функции от текущего, то соседнее решение принимается за текущее. Кроме того, даже если соседнее решение по целевой функции превосходит текущее, то с вероятностью

$$P\left(x^{(k)}, y, k\right) = \begin{cases} 1, & \text{если } F(y) \le F(x); \\ \exp\left(\left(F(x) - F(y)\right)/c_k\right), & \text{иначе.} \end{cases}$$

соседнее решение принимается за текущее. Данный шаг позволяет алгоритму имитации отжига не «застревать» в локальных оптимумах и исследовать больше количество допустимых решений.

Результаты экспериментов

Разработанные алгоритмы вложены в программный пакет mpitaskmap (см. рис. 3), в который входя средства анализа протоколов выполнения MPI-программ и построения их информационных графов, средства оценки производительности каналов связи, а также модуль распределения ветвей параллельных программ по процессорным ядрам.

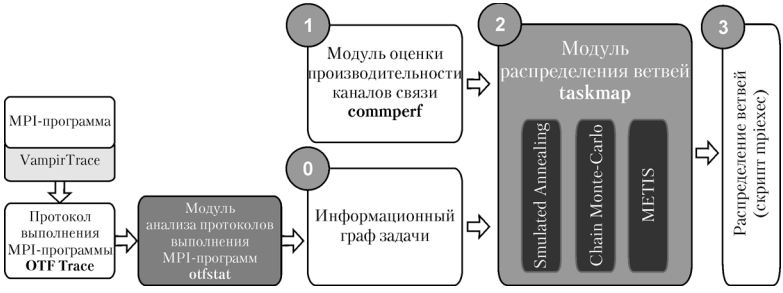


Рис. 3. Функциональная структура программного пакета mpitaskmap

На вычислительных кластерах Центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики (ЦПВТ СибГУТИ) и Лаборатории вычислительных систем института физики полупроводников СО РАН (ИФП СО РАН) проведено исследование разработанных алгоритмов и программного пакета. Использовались вычислительные кластеры следующих конфигураций:

- кластер ЦПВТ СибГУТИ: 4 узла: 2x Intel Xeon 5150, сети Gigabit/Fast Ethernet;
- кластер Лаборатории вычислительных систем ИФП СО РАН: 5 узлов: 2x AMD Opteron 248—252, сети Gigabit/Fast Ethernet.

В качестве тестовых задач использовались MPI-программы из пакетов NPB (NAS Parallel Benchmarks), HPL и SPEC MPI2007.

На рис. 4 и 5 приведены структуры информационных графов тестов NPB CG и HPL, а в табл. 1, результаты запуска тестов на различных сетях связи с распределением ветвей по ядрам различными алгоритмами.

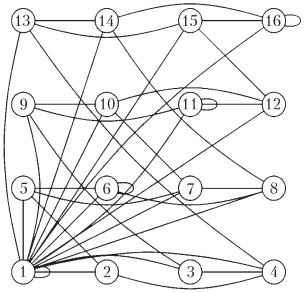


Рис. 4. Структура теста NPB CG

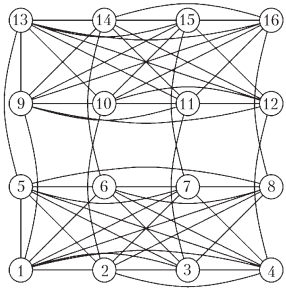


Рис. 5. Структура теста HPL

Таблица 1

Результаты экспериментов

Сеть	RR (mpiexec), с.	PTA_SA, с.	%	Xsubopt
Тест NPB CG, кластер: 5 узлов 2 x AMD Opteron 248				
Fast Ethernet	402.73	140.80	65	(2, 4) (6, 8) ( ) (5, 7) (1, 3)
Gigabit Ethernet	107.72	56.06	47	(2, 4) (6, 8) ( ) (5, 7) (1, 3)
Тест NPB CG, кластер: 4 узла 2 x Intel Xeon 5150				
Fast Ethernet	726.02	400.36	44	(1,2,3,4) (13,14,15,16) (5,6,7,8) (9,10,11,12)
Gigabit Ethernet	97.56	42.05	57	(1,2,3,4) (9,10,11,12) (5,6,7,8) (13,14,15,16)
Тест HPL, кластер 5 узлов 2 x AMD Opteron 248				
Fast Ethernet	1108.69	911.81	17	(5,6,13,14) (3,4,11,12) (7,8,15,16) (1,2,9,10)
Gigabit Ethernet	263.15	231.72	12	(5,6,13,14) (3,4,11,12) (7,8,15,16) (1,2,9,10)

Заключение

Результаты исследования разработанных алгоритмов и программного пакета показывают, что даже на сосредоточенных SMP-кластерах учет производительности каналов связи коммуникационной сред и структур информационных графов задач позволяет сократить время выполнения MPI-программ.

Литература

1. **Хорошевский В. Г.** Архитектура вычислительных систем. — М. : МГТУ им. Н. Э. Баумана, 2005.

2. **Береснев В. Л.** Дискретные задачи размещения и полиномы от булевых переменных. — Новосибирск : ИМ СО РАН, 2005.

3. **Кочетов Ю. А.** Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения. — М. : МГУ, 2001. — С. 87—117.