АЛГОРИТМЫ РАЗМЕЩЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА РЕСУРСАХ НЕОДНОРОДНЫХ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Е.А. Киселёв, О.С. Аладышев

Для современных многопроцессорных вычислительных систем (МВС) характерна неоднородность вычислительных ресурсов и коммуникационной инфраструктуры, что обуславливает зависимость времени выполнения параллельной программы от размещения ее отдельных ветвей по вычислительным устройствам МВС (вычислительным узлам, микропроцессорам, вычислительным ядрам микропроцессоров). Учет коммуникационных операций и вычислительной сложности ветвей при размещении параллельной программы на ресурсах МВС позволяет сократить время ее выполнения.

Задача размещения параллельной программы на вычислительных устройствах (ВУ) неоднородной МВС сводится к задаче отображения информационного графа программы на граф МВС.

Пусть задан граф $S = (C, L)$ МВС, содержащей вычислительные устройства различного типа. Множество вершин C графа S соответствуют ВУ МВС, а множество ребер L – каналам связи, соединяющим ВУ МВС. Для каждой вершины графа МВС задается пара значений (τ_i, ρ_i), где τ_i – тип вычислительного устройства i, а ρ_i – производительность вычислительного устройства i. Каждому ребру графа МВС ставится в соответствие пара значений (β_{ij}, θ_{ij}), где β_S – пропускная способность канала связи между ВУ i и j, а θ_S – его латентность.

Пусть задан информационный граф программы $I = (V, E)$. Вершины графа I соответствуют ветвям параллельной программы, а ребра E – информационным обменам между ними. Каждой вершине информационного графа ставится в соответствие пара значений (τ_j, ρ_j), где τ_j – тип ветви параллельной программы (т.е. тип вычислителя, на котором данная ветвь может выполняться), а ρ_j – количество выполняемых ветвью операций. Каждому ребру информационного графа ставится в соответствие пара (d_{ji}, δ_{ji}), где d_{ji} – объем переданных данных между ветвями i и j программы, а δ_{ji} – количество операций приема/передачи данных между ветвями i и j.

Граф МВС и информационный граф программы могут быть представлены в виде «коммуникационных матриц» M_S и M_I соответственно. Под «коммуникационной матрицей» будем понимать симметричную квадратную матрицу, на главной диагонали которой размещаются весовые коэффициенты вершин графа, а остальные элементы соответствуют весовым коэффициентам ребер. Примеры «коммуникационных матриц» графа МВС и информационного графа программы представлены на рисунке 1.

	c_0	c_1	c_2	c_3	c_4
c_0	(τ_0,ρ_0)	(β_{01},θ_{01})	(β_{02},θ_{02})	(β_{03},θ_{03})	(β_{04},θ_{04})
c_1	(β_{01},θ_{01})	(τ_1,ρ_1)	$(\beta_{12}\theta_{12})$	$(\beta_{13,13})$	(β_{14},θ_{14})
c_2	(β_{02},θ_{02})	(β_{12},θ_{12})	(τ_2,ρ_2)	(β_{23},θ_{23})	(β_{24},θ_{24})
c_3	(β_{03},θ_{03})	(β_{13},θ_{13})	(β_{23},θ_{23})	(τ_3,ρ_3)	(β_{34},θ_{34})
c_4	(β_{04},θ_{04})	(β_{24},θ_{24})	(β_{24},θ_{24})	(β_{34},θ_{34})	(τ_4,ρ_4)

(а)

	v_0	v_1	v_2	v_3	v_4
v_0	(τ_0,ρ_0)	(d_{01},δ_{01})	(d_{02},δ_{02})	(d_{03},δ_{03})	(d_{04},δ_{04})
v_1	(d_{01},δ_{01})	(τ_1,ρ_1)	(d_{12},δ_{12})	(d_{13},δ_{13})	(d_{14},δ_{14})
v_2	(d_{02},δ_{02})	(d_{12},δ_{12})	(τ_2,ρ_2)	(d_{23},δ_{23})	(d_{24},δ_{24})
v_3	(d_{03},δ_{03})	(d_{13},δ_{13})	(d_{23},δ_{23})	(τ_3,ρ_3)	(d_{34},δ_{34})
v_4	(d_{02},δ_{02})	(d_{14},δ_{14})	(d_{24},δ_{24})	(d_{34},δ_{34})	(τ_4,ρ_4)

(б)

Рис. 1. Коммуникационные матрицы графа МВС (а) и информационного графа программы (б)

Отображение информационного графа параллельной программы $I = (V, E)$ на граф МВС $S = (C, L)$ обозначим как $\phi: V \to C$ и представим матрицей X_ϕ:

$$X_\phi = (x_{ij}), \text{ где } \quad x_{ij} = \begin{cases} 1, & \phi(v_i) = c_j \\ 0, & \phi(v_i) \neq c_j \end{cases} \quad v_i \in V, c_j \in C, |V| = |C|.$$

Таким образом, задачу размещения параллельной программы на ресурсах МВС можно свести к задаче нахождения матрицы X_ϕ, для которой время выполнения параллельной программы на заданной вычислительной системе наименьшее.

Ожидаемое время выполнения параллельной программы $T(X_\phi)$ при схеме размещения X_ϕ вычисляется в соответствии с функционалом (1) и включает в себя время выполнения вычислительных операций наиболее длительной ветви программы, время выполнения коммуникационных обменов между ветвями программы и время, затраченное на синхронизацию ветвей программы.

$$T(X_\phi) = \sum_{i=1}^{|I|}\sum_{m=1}^{|S|} x_{im}\cdot\frac{\rho(v_i)}{\rho(c_m)} + \sum_{i=1}^{|I|}\sum_{j=1}^{|I|}\sum_{m=1}^{|S|}\sum_{n=1}^{|S|} x_{im}\cdot x_{jn}\cdot\left(\vartheta(l_{mn})\cdot\delta(e_{ij}) + \frac{d(e_{ij})}{\beta(l_{mn})}\right) \to \min_{(x_{ij})} \quad (1)$$

при ограничении:

$$\tau(v_i) = \tau(c_m)\ \forall\ v_i \in V, c_m \in C \quad (2)$$

Ограничение (2) обусловливает назначение ветвей параллельной программы на ВУ МВС с учетом требуемого типа ВУ.

Искомому размещению программы соответствует отображение X_ϕ, для которого достигается минимум $T(X_\phi)$ функционала (1) при ограничении (2).

Задача размещения программ является NP-полной. Для МВС с большим числом ВУ (от 30 и более) применяются эвристические алгоритмы, позволяющие за приемлемое для практических случаев время определять рациональное размещение параллельной программы на МВС. На практике применяются более простые алгоритмы размещения, не учитывающие информационного графа программы и графа МВС [1]: *Processor Allocation Random* (*PAR*) – распределение ветвей программы по ВУ МВС случайным образом; *Processor Allocation Linear* (*PAL*) – выбор первых *M* свободных ВУ; *Processor Allocation Round Robin* (*PARR*) – распределение ветвей программы по ВУ системы из *N* вычислительных узлов в следующем порядке: первая ветвь программы размещается на первом ВУ первого по порядку вычислительного узла, вторая – на первом ВУ второго вычислительного узла, ветвь *N* – на первом ВУ вычислительного узла *N*, ветвь *N* + 1 – на втором ВУ первого вычислительного узла и т. д.

Анализ известных эвристических алгоритмов размещения параллельных программ, показал, что предпочтительными по точности получаемого результата и скорости работы являются алгоритмы, основанные на методе моделирования отжига [2]. Данный метод имитирует физический процесс отжига металла, состоящий в скачкообразном нагревании и охлаждении металла с постепенным понижением температуры нагрева. Алгоритм размещения, основанный на методе моделирования отжига, используется в системе управления прохождением параллельных заданий МВС (СУППЗ [3]). Однако, данный алгоритм ориентирован на однородные МВС и требует адаптации к случаю МВС с неоднородными ВУ.

Алгоритм SA*(I,S,T0, Δ)* имитации отжига для решения задачи отображения информационного графа программы *I* на граф ВС *S*, с заданной начальной температурой T_0 и областью сходимости Δ можно описать следующим образом.

Шаг 1. Установить начальную температуру $T = T_0$.

Шаг 2. Выбрать начальное произвольное отображение X_ϕ (с помощью алгоритма PAR) и вычислить значение $T(X_\phi)$ функционала (1).

Шаг 3. Выбрать произвольную вершину v_i информационного графа.

Шаг 4. Выполнить в цикле по всем вершинам c_j графа *S* шаги с 4.1 по 4.4.

Шаг 4.1. Поместить вершину vi на очередную вершину cj графа S с учетом (2).

Шаг 4.2. Вычислить значение $T(X_\phi)$ функционала (1).

Шаг 4.3. Вычислить приращение функционала $\Delta T(X_\phi) = T(X_\phi) - T'(X_\phi)$, где $T'(X_\phi)$ - значение функционала (1) на предыдущей итерации алгоритма. Если $\Delta T(X_\phi) \le 0$, то вершина v_i закрепляется за вершиной c_j. Если $\Delta T(X_\phi) > 0$, то закрепление вершины v_i за вершиной c_j осуществляется с вероятностью $e^{-\frac{\Delta T(X\phi)}{T}}$.

Шаг 4.4. Если все вершины c_j были пройдены, понизить температуру *T* по закону $T = \alpha \cdot T$, где α – параметр, влияющий на скорость понижения температуры, и перейти к шагу 3. Если зафиксирован выход функционала $T(X_\phi)$ на стационарное значение (т.е. в течение заданного числа шагов Δ , значение функционала $T(X_\phi)$ не изменяется) или текущее значение $T = 0$ – завершить работу алгоритма. В остальных случаях – увеличить значение *j* на 1 и перейти к шагу 4.1.

Вычислительная сложность алгоритма *SA* оценивается как $O(|C||I|)$.

В докладе рассматриваются два новых алгоритма *Parallel Simulated Annealing* (*PSA*) и *Parallel Cut Simulated Annealing* (*PCSA*) размещения параллельной программы на ВУ МВС [4], основанные на методе моделирования отжига, учитывающие характеристики информационных обменов между ветвями параллельной программы, а также неоднородность вычислительных и коммуникационных ресурсов МВС.

Пусть заданы коммуникационные матрицы M_I размера *m* и M_S – размера *n*, где $m << n$. В процессе выполнения алгоритма *PSA* производится распределение компонентов графа МВС по *k* ВУ в соответствии с разбиением матрицы M_S на *k* частей. Обозначим за $M_S^i, i \in [1;k]$ – часть матрицы M_S, размещенную в памяти *i*-го ВУ. Параллельная реализация алгоритма *SA*(*I,S,T0,* Δ) отображения информационного графа *I* на граф МВС *S* для *k* ВУ (алгоритм *PSA*) может быть описана следующим образом.

Шаг 1. Выполнить в цикле шаги с 1.1 по 1.5.

Шаг 1.1. Сформировать новое допустимое сочетание $(C_n^m)_i$ элементов коммуникационной матрицы M_S графа ВС. Допустимым является сочетание, соответствующее назначению ветви параллельной программы на один процессор или вычислительное ядро.

Шаг 1.2. Передать значения элементов $(C_n^m)_i$ очередному ВУ *i* в качестве элементов матрицы M_S^i для выполнения алгоритма моделирования отжига.

Шаг 1.3. На каждом процессоре *i*-го ВУ выполнить алгоритм моделирования отжига $SA(I,Si,T_0, \Delta)$, где S_i – подграф графа МВС, заданный матрицей M_S^i .

Шаг 1.4. Для каждого ВУ выбрать решение с минимальным значением $T(X_\phi)$ функционала (1) из найденных на всех процессорах.

Шаг 1.5. Выбрать решение с минимальным значением $T(X_\phi)$ функционала (1) из найденных на всех ВУ. Перейти к шагу 2.

Шаг 2. Если все сочетания рассмотрены – завершить выполнение алгоритма и принять в качестве искомого – лучшее из найденных решений в процессе выполнения алгоритма. Иначе перейти к шагу 1 алгоритма.

Вычислительная сложность алгоритма *PSA* оценивается как $O(|V^2|)$.

Идея алгоритма *PCSA* состоит в распределении ветвей по «близко расположенным» процессорам с целью локализации коммуникационного трафика интенсивно обменивающихся ветвей программы в рамках заданного сегмента сети МВС и сокращения его влияния на обмены других ветвей программы. Интенсивность взаимодействия ветвей определяется количеством операций приема-передачи данных.

Выделение интенсивно взаимодействующих ветвей в программе сводится к задаче декомпозиции информационного графа программы на подграфы максимального веса. Процесс декомпозиции графа программы состоит из трех этапов.

1. *Этап сжатия графа*. Исходный граф G_0 преобразуется в последовательность графов $G_1, G_2, ..., G_m$ меньших размерностей, таких, что $|V_0|>|V_1|>|V_m|$. Графы меньших размерностей формируются объединением смежных вершин, инцидентных ребру максимального веса. Сжатие выполняется до тех пор, пока размеры графа G_m не позволят применить к нему алгоритм рекурсивной бисекции *GGP (Graph Growing Partitioning)*. Сжатие графа $G_i = (V_i, E_i)$ осуществляется стохастическим алгоритмом *HEM (Heavy Edge Matching)*. Вычислительная сложность алгоритма *HEM* составляет $O(|E_i|)$.
2. *Этап начального разбиения*. Граф $G_m = (V_m, E_m)$ малой размерности разбивается на *k* подмножеств алгоритмом *GGP*. Для того, чтобы разбить граф на *k* подмножеств, алгоритму *GGP* требуется выполнить порядка $O(|E| \cdot \log_2 k)$ операций. Поскольку, граф G_m имеет небольшую размерность, то время реализации такого разбиения незначительно.
3. *Этап улучшения разбиения*. Полученное на втором этапе разбиение улучшается нетрудоемким алгоритмом *BKL (Boundary Kernighan-Lin)*, основанным на эвристике Кернигана-Лина. Вычислительная сложность алгоритма *BKL* составляет $O(|E_i|)$.

После распределения ветвей программы по процессорам МВС на каждом процессоре выполняется одновременный запуск алгоритма *SA*. В качестве информационного графа и графа МВС на вход алгоритма *SA* подаются подграфы, описывающие связь ядер внутри процессора и часть параллельной программы, которая была распределена на данный процессор.

Вычислительная сложность алгоритма составляет $O(c^2)$, где *c* - равно максимальному количеству ядер в процессоре МВС.

Оценка эффективности предложенных алгоритмов *PSA* и *PCSA* и разработанных программных средств производилась на тестовом наборе программ: NAS Parallel Benchmarks, MV_Orig и MV_Allreduce.

Программный пакет NAS Parallel Benchmarks представляет собой набор синтетических программ, эмулирующих решение распространенных научных задач, и является стандартным средством оценки производительности МВС. Программы MV_Orig и MV_Allreduce представляют собой две различные реализации умножения разреженной матрицы на вектор в алгоритме Ланцоша.

Сравнение предложенных автором алгоритмов *PSA* и *PCSA* с применяемыми в настоящее время алгоритмами размещения программ *PAL*, *PAR*, *SA* проводилось на вычислительных системах Tilempower (ВС с общей памятью, построенной на базе 64-ядерного микропроцессора TilePro64 с архитектурой вида «сеть на кристалле») и «МВС-100К» (универсальная неоднородная вычислительная система с распределенной памятью, в состав которой входят многоядерные микропроцессоры Intel Xeon и графические ускорители Nvidia Tesla).

Сравнение алгоритмов *SA*, *PSA* и *PCSA* проводилось при одинаковых значения входных параметров (T_0=2000 и Δ=30). Отмечается, что при таких параметрах T_0 и Δ достигается приемлемая точность отображения информационного графа программы на граф МВС с помощью алгоритма *SA* при небольшом времени работы. Точность отображения информационного графа программы на граф МВС оценивается процентом совпадения ребер графов.

Для оценки эффективности размещения на ресурсах МВС программы i алгоритмом размещения j автором введен показатель эффективности $\varepsilon(i,j)$.

$$\varepsilon(i,j) = \frac{T_i + T_i^{/}}{T_{ij} + T_{ij}^{/}} \quad , \qquad (3)$$

где T_i – время выполнения параллельной программы при использовании алгоритма размещения *PAL,* $T_i^{/}$ - время построения схемы размещения программы i алгоритмом *PAL,* T_{ij} - время выполнения программы с использованием алгоритма размещения j, $T_{ij}^{/}$ - время построения схемы размещения программы i алгоритмом j.

Эффективность размещения набора из m программ алгоритмом j может быть оценена по формуле (4).

$$\varepsilon(j) = \frac{\sum_{i=1}^{m} \varepsilon(i,j)}{m} \quad , \qquad (4)$$

где m – количество программ в тестовом наборе.

Определен критерий эффективности алгоритма j (5).

$$j : \varepsilon(j) = \max_{l \in \{PAR,SA,PSA,PCSA\}} \{\varepsilon(l)\} \text{ , при } T_0 = 2000 \text{ , } \Delta = 30 \qquad (5)$$

Эффективным считается алгоритм j с максимальным значением показателя эффективности (4) для заданных параметров T_0 и Δ .

Оценка эффективности алгоритмов размещения *PSA* и *PCSA*, производилась с точностью до 0,01 в соответствии со следующими пунктами методики.

Обозначим n – число ядер в МВС, на которых выполняется запуск параллельных программ, а m – число параллельных программ в тестовом наборе.

1. Выполнить в цикле по всем m программам из тестового набора пункты методики с 1.1 по 1.3 включительно.

1.1. На n вычислительных ядрах свободной от вычислений МВС запустить на выполнение программу $i \in [1;m]$ из тестового набора с использованием алгоритма PAL. Измерить время T_i выполнения программы i с требуемой точностью.

1.2. Выполнить в цикле по всем алгоритмам *PAR*, *SA*, *PSA* и *PCSA* пункты методики с 1.2.1 по 1.2.2 включительно.

1.2.1. Сформировать с помощью алгоритма $j \in \{PAR, SA, PSA, PCSA\}$ схему размещения программы i на n ядрах МВС. Измерить время $T_{ij}^{/}$ построения алгоритмом j схемы размещения программы i с требуемой точностью.

1.2.2. На свободной от вычислений МВС запустить на выполнение программу i из рассмотренного выше тестового набора с использованием алгоритма j на N вычислительных ядрах. Измерить время выполнения T_{ij} программы i с требуемой точностью.

2. Выполнить в цикле по всем алгоритмам *PAR*, *SA*, *PSA* и *PCSA* пункты методики с 2.1 по 2.2 включительно.

2.1. Для всех программам из тестового набора вычислить значение показателя эффективности $\varepsilon(i,j)$ алгоритма j при размещении программы i в соответствии с (3).

2.2. Для алгоритма размещения j в соответствии с (4) вычислить среднее значение показателя эффективности $\varepsilon(j)$ по всем программам из тестового набора.

3. Выбрать алгоритм размещения j в соответствии *с* критерием (5).

Результаты оценки эффективности алгоритмов *PSA, PCSA, SA* и *PAR* приведены в таблице 1.

Согласно представленным в таблице 1 результатам и в соответствии с критерием (5) эффективным является алгоритм *PCSA*.

Таблица 1. Оценка эффективности алгоритмов *PSA, PCSA, SA* и *PAR* на вычислительных системах Tilempower и «МВС-100К»

Название программы из тестового набора	**Значение показателя эффективности $\varepsilon(i,j)$ размещения программы *i* алгоритмом *j***			
	PAR	**SA**	**PSA**	**PCSA**
Tilempower				
MV_Allreduce	1,06	1,12	1,24	1,34

Таблица 1. Оценка эффективности алгоритмов *PSA, PCSA, SA* и *PAR* на вычислительных системах Tilempower и «МВС-100К»

Название программы из тестового набора	**Значение показателя эффективности $\varepsilon(i,j)$ размещения программы *i* алгоритмом *j***			
MV_Orig	0,77	4,16	5,03	5,02
NPB SP	0,92	1,02	1,12	1,14
NPB LU	1,09	1,89	2,05	2,42
NOB IS	1,05	1,12	1,26	1,34
NPB FT	1,07	1,13	1,32	1,3
NPB CG	1,21	1,65	2,08	2,36
NPB BT	0,89	1,10	1,14	1,27
NPB MG	0,81	1,88	3,72	3,74
NPB DT BH	1,43	2,35	4,79	5,31
NPB DT WH	1,32	2,61	4,46	5,33
NPB DT SH	0,79	1,23	1,32	2,14
NPB EP	1,01	0,69	0,72	0,92
Показатель эффективности $\varepsilon(j)$ алгоритма размещения *j*	**1,03**	**1,69**	**2,33**	**2,59**
«МВС-**100К**»				
MV_Allreduce	1,02	1,04	1,07	1,09
MV_Orig	1,02	1,02	1,04	1,06
NPB SP	1,00	1,00	1,02	1,02
NPB LU	1,00	0,8	0,8	0,8
NOB IS	1,00	1,02	1,03	1,04
NPB FT	1,03	1,04	1,04	1,05
NPB CG	1,00	1,00	1,04	1,04
NPB BT	1,01	1,02	1,02	1,03
NPB MG	1,00	1,02	1,04	1,05
NPB DT BH	1,00	1,02	1,03	1,03
NPB DT WH	1,01	1,02	1,03	1,03
NPB DT SH	1,00	1,04	1,05	1,06
NPB EP	1,00	1,01	1,03	1,04
Показатель эффективности $\varepsilon(j)$ алгоритма размещения *j*	**1,01**	**1,00**	**1,02**	**1,03**

Представленные в таблице 1 результаты получены путем вычисления средних значений показателя эффективности $\varepsilon(j)$. Точности полученных измерений оценивалась с доверительной вероятность равной 0,95 и доверительным интервалом равным 0,01.

Анализ результатов экспериментов показал, что предложенные алгоритмы *PSA* и *PCSA* в большинстве случаев обеспечивают более эффективное использование вычислительных и коммуникационных ресурсов МВС по сравнению с известными алгоритмами размещения параллельных программ на ресурсах МВС.

ЛИТЕРАТУРА:

1. М.Г. Курносов. Алгоритмы распределения ветвей параллельных программ по процессорным ядрам мультикластерных вычислительных систем. Материалы Всероссийской научной конференции «Наука. Технологии. Инновации». – Новосибирск: Изд-во НГТУ, 2007. – С. 258-260.
2. О.Г. Монахов, Э.А. Монахова. Параллельные системы с распределенной памятью: управление ресурсами и заданиями. – Новосибирск: Изд-во ИВМиМГ СО РАН, 2001. – 168 с.
3. А.В. Баранов, Д.М. Голинка. Система управления прохождением задач и планировщик Maui для МВС-100К. Труды Всероссийской суперкомпьютерной конференции «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность» (21-26 сентября 2009 г., г. Новороссийск). М.: Изд-во МГУ. 2009. С. 314-315. ISBN 978-5-211-05697-8.
4. Е.А. Киселёв, О.С. Аладышев. Алгоритм эффективного размещения программ на ресурсах многопроцессорных вычислительных систем. Программные продукты и системы. – Тверь, 2012 – Вып. 4. – с. 18-25.