КЛАРК Д., ЛАСТОВЕЦКИЙ А., РЫЧКОВ В.

Университетский колледж Дублина, Ирландия

РАЗРАБОТКА И РЕАЛИЗАЦИЯ АДАПТИВНЫХ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ НЕОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

В докладе описывается алгоритм оптимального распределения независимых единиц вычисления между неоднородными процессорами, скорость которых характеризуется априори неизвестными функциями объема вычислений, назначенных этим процессорам. Высокая эффективность алгоритма, позволяющая использовать его при расзработке адаптивных программ для выполнения на неоднородных вычислительных системах, демонстрируется на примере задач вычислительной линейной алгебры.

1. Введение

Традиционные алгоритмы распределения вычислений между неоднородными процессорами [1-4] основаны на константной модели производительности (КМП), которая представляет скорость процессора положительным константой. При этом вычисления между процессорами распределеяются пропорционально этим постоянным скоростям. Таким образом, традиционные алгоритмы предполагают, что скорость процессора не зависит от размера вычислительной задачи, решаемой ни этом процессоре. Это предположение, как правило, справедливое при решении вычислительных научных задач среднего размера на неоднородных кластерах рабочих станций, становится неверным в следующих случаях:

- В результате разбиения вычислений, некоторые задачи либо не помещаются главной памяти своего процессора, вызывая подкачку страниц с дисковой памяти, либо, наоборот, полностью умещаются в верхних, более быстрых, уровнях памяти.
- Некоторые вычислительные устройства, участвующие в выполнении программы, являются не традиционными процессорами общего назначения, а, например, ускорителями, такими как графические процессоры или специализированные ядра. В этом случае, относительная скорость традиционного и нетрадиционного процессоров может отличаться для двух разных размеров одной и той же задачи, даже если для этих размеров задача полностью умещается в основной памяти устройств.
- Разные процессоры используют разные программные для локального решения одной и той же вычислительной задачи.

Перечисленные случаи характерны для современных и, в особенности, для перспективных высокопроизводительных неоднородных вычислительных систем. В результате, применимость алгоритмов распределения вычислений, основанных на КМП, становится все более ограниченной. Действительно, чем выше уровень неоднородности и количество различных различных вычислительных устройств, тем уже становится диапазон рамеров вычислительной задачи, для которого их относительные скорости можно считать постоянными. Дла этих систем нужны новые алгоритмы, способные оптимально распределять вычисления для всего диапазона размеров вычислительных задач.

Функциональная модель производительности (ФМП) [5] намного реалистичней КМП, поскольку учитывает влияие на производительность процессоров многих важных факторов, таких как неоднородность архитектуры и ее прогрммно-аппаратной реализации, неоднородность структуры памяти, эффекты подкачки страниц и т.д. [6-7]. ФМП представляет скорость процессора функцией размера задачи. Скорость определяется как число единиц вычисления, выполняемых процессором в единицу.

Проблема распределения вычислений, использующая ФМП, была сформулирована и решена в [6-7] следующим образом. Общий размер задачи n представляет собой число единиц вычисления, которые нужно распределеить между p ($p < <n$) процессорами $P_1,\ldots,P_p$. Скорости процессоров представляются положительными непрерывными функциями размера

задачи $s_1(x),\ldots,s_p(x)$: $s_i(x) = x / t_i(x)$, где $t_i(x)$ – это время выполнения x единиц вычисления процессором i. Функции скорости определены в диапазоне [0, n]. Алгоритм решения этой проблемы возвращает распределение единиц вычисления, $d_1,\ldots,d_p$, между процессорами так, что $d_1 + d_2 + \ldots + d_p = n$. Нагрузка будет сбалансирована, если все процессоры завершат вычисления одновременно: $t_1(d_1) \approx t_2(d_2) \approx \ldots \approx t_p(d_p)$. Это можно выразить следующим образом:

$$\begin{cases} \dfrac{d_1}{s_1(d_1)} \approx \dfrac{d_2}{s_2(d_2)} \approx \ldots \approx \dfrac{d_p}{s_p(d_p)} \\ d_1 + d_2 + \ldots + d_p = n \end{cases}$$

Геометрически, решение этой системы уравнений, $d_1,\ldots,d_p$, можно представляется финкций скорости прямой линией, проходящей через начало системы координат (рис. 1).

Геометрический алгоритм решения этой проблемы, предложенный в [6-7], суммируется следующим образом. Любая прамая, выходящая из начала системы координат и пересекающая функции скорости представляет оптимальное распределение вычислений для некорого конкретного размера проблемы. Поэтому, пространство решений проблемы распределения вычислений состоит из всех таких прямых линий. Две прямые выбираются в качестве начального приближения. Верхняя прямая представляет оптимальное распределение $d_1^u,\ldots,d_p^u$ для проблемы размером $n_u < n$, $n_u = d_1^u + \ldots + d_p^u$, а нижняя прямая дает решение $d_1^l,\ldots,d_p^l$ для $n_l > n$, $n_l = d_1^l + \ldots + d_p^l$. Область между этими прямыми последовательно делится пополам. На шаге k, размер проблемы, соответствующий новой прямой, пересекающей функции скорости в точках $d_1^k,\ldots,d_p^k$ вычисляется как $n_k = d_1^k + \ldots + d_p^k$. В зависимиости от того, является ли n_k меньше или больше, чем n, эта прямая становится новой верхней или нижней границей. Приближая n_k к n, алгоритм находит оптимальное разбиение проблемы данного размера $d_1,\ldots,d_p$: $d_1 + \ldots + d_p = n$.

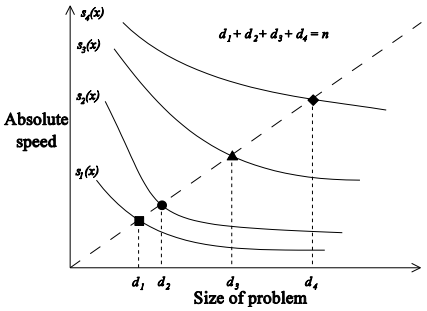

Рис. 1. Оптимальное распределение вычислений, демострирующее геометрическую пропорциональнсть числа единиц вычисления скорости процессора

Алгоритмы решения задач линейной алгебры на неоднородных вычислительных системах, использующие приведенный алгоритм в качестве основного строительного блока, опубликованы в [8-10].

Построение полной ФПМ может быть очень дорогой операцией, поскольку требует выполнения вычислительного ядра для большого числа размеров задачи. Проблема минимизации стоимости построения полных ФМП изучалась в [11], где предложено ее относительно эффективное приближенное решение. Однако, даже в случае оптимального решения этой проблемы стоимость построения полной ФМП будет слишком высокой для того, чтобы использовать алгоритмы распределения вычислений на основе полных ФМП в адаптивных научных прикладных программах.

В докладе описывается другое решение проблемы минимизации стоимости использования ФМП для распределении вычислений между неоднородными процессорами. Вместо использования полных ФМП, описываемый алгоритм строит и использует их частичные приближения, достаточные для оптимального распределения вычислений.

2. Алгоритм распределения вычислений, основанный на частичных ФМП

В математической форме, проблема распределения вычислений между неоднородными поцессорами, которую мы пытаемся решить, выглядит следующим образом:

Дано:

- Множество n независимых и одинаковых единиц вычисления;
- Множество p ($p << n$) процессоров $P_1,\ldots,P_p$, выполняющих x единиц вычисления за время $t_i(x)$;
- Требуемая относительная точность решения, ε.

Найти: разбиение n единиц вычисления между p процессорами, такое что $\sum_{i=1}^{p} d_i = n$ и $\max\limits_{1\le i,j\le p}\left|\dfrac{t_i(d_i)-t_j(d_j)}{t_i(d_i)}\right| \le \varepsilon$.

Алгоритм решения:

1. Все p процессоров параллельно выполняют n / p единиц вычисления. Время выполнения $t_1(n / p),\ldots,t_p(n / p)$ посылается процессору P_1.
2. IF $\max\limits_{1\le i,j\le p}\left|\dfrac{t_i(n/p)-t_j(n/p)}{t_i(n/p)}\right| \le \varepsilon$ THEN равномерное распределения вычислений является приемлемым решением и алгоритм останавливается;

ELSE процессор P_1 вычисляет абсолютные скорости всех процессоров, $s_i(n / p) = (n / p) / t_i(n / p)$ для $1 \le i \le p$, и строит первое приближение их ФМП в виде КМП, $s_i(x) = s_i(n / p)$.

3. Применяя геометрический алгоритм распределения вычислений к этому приближению ФМП, процессор P_1 вычисляет новое распределение, $d_1,\ldots,d_p$, и затем посылает каждому процессору P_i новое число единиц вычисления, d_i , которое тот должен выполнять.
4. Процессор P_i , параллельно с другими процессорами, выполняет d_i единиц вычислений, $1 \le i \le p$. Время выполнения $t_1(d_1),\ldots,t_p(d_p)$ посылается процессору P_1.
5. IF $\max\limits_{1\le i,j\le p}\left|\dfrac{t_i(d_i)-t_j(d_j)}{t_i(d_i)}\right| \le \varepsilon$, THEN текущее распределение, $d_1,\ldots,d_p$, решает проблему и алгоритм останавливается;

ELSE процессор P_1 вычисляет абсолютные скорости, демонстрируемые процессорами для этого распределения, $s_i(d_i) = d_i / t_i(d_i)$ $(1 \le i \le p)$, и использует эти вновь полученные точки функциональных моделей процессоров P_i, $(d_i, s_i(d_i))$, для построения их более точного кусочно-линейного приближения. А именно, пусть $\{(d_i^{(j)}, s_i(d_i^{(j)}))\}_{j=1}^{m}$, $d_i^{(1)} < \ldots < d_i^{(m)}$, - экспериментально полученные точки $s_i(x)$, использованные для построения ее текущего кусочно-линейного приближения. Тогда

- IF $d_i < d_i^{(1)}$, THEN отрезок $(0, s_i(d_i^{(1)})) \to (d_i^{(1)}, s_i(d_i^{(1)}))$ этого приблидения заменяется парой отрезков $(0, s_i(d_{(i)})) \to (d_i, s_i(d_i))$ и $(d_i, s_i(d_i)) \to (d_i^{(1)}, s_i(d_i^{(1)}))$;
- IF $d_i > d_i^{(m)}$, THEN прямая $(d_i^{(m)}, s_i(d_i^{(m)})) \to (\infty, s_i(d_i^{(m)}))$ заменяется отрезком $(d_i^{(m)}, s_i(d_i^{(m)})) \to (d_i, s_i(d_i))$ прямой $(d_i, s_i(d_i)) \to (\infty, s_i(d_i))$;
- IF $d_i^{(k)} < d_i < d_i^{(k+1)}$, THEN отрезок $(d_i^{(k)}, s_i(d_i^{(k)})) \to (d_i^{(k+1)}, s_i(d_i^{(k+1)}))$ парой отрезков $(d_i^{(k)}, s_i(d_i^{(k)})) \to (d_i, s_i(d_i))$ и $(d_i, s_i(d_i)) \to (d_i^{(k+1)}, s_i(d_i^{(k+1)}))$.

6. GOTO 3.

Описанный алгоритм динамически строит функциональные модели процессоров в требуемом диапазоне и с требуемой точностью, и возвращает решение, совпадающее или близкое к решению, основанному на полных функциональных моделях. Данные затем распределяются в соответствии с найденным решением и программа затем выполняет вычисления над этими данными. В докладе приводятся примеры применения описанного алгоритма для реализации адаптиных научных прикладных программ для современных

неоднородных вычислительных систем, демонстрирущие эффективность предложенного решения. Другие приложения можно найти в [12-15].

Литература

1. A. Kalinov, A. Lastovetsky, Heterogeneous distribution of computations while solving linear algebra problems on networks of heterogeneous computers, in: HPCN Europe 1999, LNCS 1593 (1999) 191-200.

2. O. Beaumont, V. Boudet, F. Rastello, Y. Robert, Matrix Multiplication on Heterogeneous Platforms, IEEE Trans. Parallel Distrib. Syst. 12 (2001) 1033-1051.

3. A. Kalinov, A. Lastovetsky, Heterogeneous Distribution of Computations Solving Linear Algebra Problems on Networks of Heterogeneous Computers, J. Parallel Distrib. Comput. 61 (2001) 520-535.

4. M. Cierniak, M. Zaki, W. Li, Compile-Time Scheduling Algorithms for Heterogeneous Network of Workstations, Computer J. 40 (1997) 356-372.

5. A. Lastovetsky and J. Twamley, Towards a Realistic Performance Model for Networks of Heterogeneous Computers, in: IFIP TC5 Workshop, World Computer Congress, Springer, 2005, pp. 39-58.

6. A. Lastovetsky and R. Reddy, Data Partitioning with a Realistic Performance Model of Networks of Heterogeneous Computers, in: 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), 2004.

7. A. Lastovetsky, R. Reddy, Data Partitioning with a Functional Performance Model of Heterogeneous Processors, Int. J. High Perform. Comput. Appl. 21 (2007) 76-90.

8. A. Lastovetsky and R. Reddy, Data Distribution for Dense Factorization on Computers with Memory Heterogeneity, Parallel Computing 33 (2007) 757-779, 2007

9. A. Lastovetsky, R. Reddy, Two-dimensional Matrix Partitioning for Parallel Computing on Heterogeneous Processors Based on their Functional Performance Models, in: HeteroPar 2009, LNCS 6043, Springer, 2010, pp. 112-121.

10. D. Clarke, A. Lastovetsky, V. Rychkov, Column-based Matrix Partitioning for Parallel Matrix Multiplication on Heterogeneous Processors Based on Functional Performance Models, in: HeteroPar 2011, LNCS 7155, pp. 450-459, 2012.

11. A. Lastovetsky, R. Reddy, and R. Higgins, Building the Functional Performance Model of a Processor, in: 21st Annual ACM Symposium on Applied Computing (SAC 2006), ACM Press, pp. 746-753, 2006.

12. D. Clarke, A. Lastovetsky, V. Rychkov, Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms, Parallel Processing Letters. 21 (2011) 195-217.

13. A. Lastovetsky, R. Reddy, V. Rychkov, D. Clarke, Design and Implementation of Self-adaptable Parallel Algorithms for Scientific Computing on Highly Heterogeneous HPC Platforms. Arxiv preprint arXiv:1109.3074 (2011)

14. D. Clarke, A. Ilic, A. Lastovetsky, and L. Sousa, Hierarchical Partitioning Algorithm for Scientific Computing on Highly Heterogeneous CPU + GPU Clusters, 18th International European Conference on Parallel and Distributed Computing (Euro-Par 2012), 2012.

15. Z. Zhong, V. Rychkov, and A. Lastovetsky, Data Partitioning on Heterogeneous Multicore Platforms, in: 2011 IEEE International Conference on Cluster Computing (Cluster 2011), pp. 580-584, 2011.