УДК 621.321

ОБОСНОВАНИЕ ПРИМЕНЕНИЯ КЛАСТЕРНЫХ ТЕХНОЛОГИЙ ДЛЯ МОДЕЛИРОВАНИЯ ВЛИЯНИЯ ТЕРРИТОРИАЛЬНО-РАСПРЕДЕЛЕННЫХ ОБЪЕКТОВ

Ю.В. КУНГУРИН
Костромской государственный университет имени Н.А. Некрасова, Кострома, Россия
E-mail: urak@ksu.edu.ru

Авторское резюме

Состояние вопроса: При проектировании территориально-распределенных объектов наиболее важной является задача автоматизации их размещения в пространстве территории с учетом их свойств, влияющих на окружающую среду. В процессе проектирования может участвовать большое количество объектов. Для быстрой обработки большого объема информации геоинформационной системы наиболее эффективно применение кластерных технологий.
Материалы и методы: Для анализа возможности и целесообразности применения кластерных технологий при решении задач моделирования пространственно-распределенных объектов используются выкладки из теории множеств и аналитические методы анализа быстродействия параллельных вычислительных систем.
Результаты: Приведен анализ эффективности использования кластерных технологий для решения задач моделирования пространственно-распределенных объектов.
Выводы: Приведенные аналитические исследования отражают эффективность применения кластерных технологий для моделирования влияния совокупности большого числа сложных территориально-распределенных объектов.

Ключевые слова: кластер, параллельные вычисления, территориально-распределенные объекты, геоинформационные системы.

REASONS FOR CLUSTERS TECHNOLOGIES APPLICATION FOR SIMULATION OF IMPACT OF GEOGRAPHICALLY DISTRIBUTED OBJECTS

J.V. KUNGURIN
Kostroma State University , Kostroma, Russia
E-mail: urak@ksu.edu

Abstract

Background: When designing geographically distributed objects the key task is to distribute them automatically taking into account the properties that influence the environment. A great amount of objects may be involved in designing. Clusters technologies are the most effective to process quickly great amount of geo information system data.
Materials and methods: Set theory basics and analytical methods of analysis of parallel computing system speed are used to analyze practicability of clusters technologies application when solving the problems of simulation of geographically distributed objects.
Results: The analysis of effectiveness of clusters technologies application when solving the problems of simulation of geographically distributed objects is presented.
Conclusions: Given analytical research shows the efficiency of clusters technologies application for simulation of impact of complex geographically distributed objects.

Key words: cluster, parallel calculations, geographically distributed objects, geoinformation systems.

Развитие средств вычислительной техники позволило применять для научных и прикладных исследований сложные многомерные и параметрические математические модели, требующие, в свою очередь, применения более мощных вычислительных средств. Производительность современных компьютеров во много раз превосходит производительность первых вычислительных систем и продолжает расти.

Однако уже сейчас технологический прогресс в области разработки микроэлектронных компонент сталкивается с ограничениями, обусловленными фундаментальными законами природы. Наблюдается уменьшение динамики развития вычислительных машин за счет усовершенствования их элементной базы. Для дальнейшего повышения быстродействия разработчикам пришлось вплотную заняться поиском новых архитектурных решений.

Традиционная архитектура ЭВМ в основе работы использовала принципы фон Неймана, т. е. предполагала последовательное расположение в памяти и выполнение всех команд одну за другой в порядке, определяемом программой. В результате поиска путей повышения быстродействия в 60-х годах XX века появилась векторная архитектура. Идея, положенная в основу новой архитектуры, заключалась в распараллеливании процесса обработки данных, когда одна и та же операция применяется одновременно к паре членов обрабатываемых массивов (векторов) значений. В этом случае

можно надеяться на определенный выигрыш в скорости вычислений. Идея распараллеливания программ оказалась плодотворной и нашла воплощение на разных уровнях функционирования компьютера. В соответствии с таксономией Флинна, можно выделить два уровня распараллеливания и их сочетание. Это распараллеливание потока команд и распараллеливание потока данных. Соответственно, выделяют четыре класса архитектур [1]:

1. ОКОД – вычислительная система с одиночным потоком команд и одиночным потоком данных Single Instruction stream over a Single Data stream (SISD). Распараллеливание отсутствует. Традиционная архитектура фон Неймана.

2. ОКМД – вычислительная система с одиночным потоком команд и множественным потоком данных Single Instruction, Multiple Data (SIMD). Происходит векторная обработка потока данных, т. е. наблюдается одновременное выполнение одной команды над множеством данных.

3. МКОД – вычислительная система со множественным потоком команд и одиночным потоком данных Multiple Instruction Single Data (MISD). Конвеерная обработка потока данных. Данный класс не получил признания.

4. МКМД – вычислительная система со множественным потоком команд и множественным потоком данных Multiple Instruction Multiple Data (MIMD). Происходит решение одной задачи на множестве процессоров с распараллеливанием данных и потока команд.

Класс МКМД включает в себя многопроцессорные системы, где процессоры обрабатывают множественные потоки данных.

В основе параллельного компьютера лежит идея использования для решения одной задачи одновременно нескольких процессоров, работающих сообща и связанных магистралями для обмена информацией. Расчет делается на то, что если одному процессору для выполнения задачи требуется время t, то p процессоров смогут решить эту задачу, разбитую на подзадачи, за время t/p. Однако это идеальное ускорение удается получить лишь в очень специальных ситуациях, когда подзадачи полностью независимы. На практике же целью программиста является построение алгоритмов, способных извлечь из наличия нескольких процессоров максимальную выгоду для данной задачи (в конечном итоге получить минимальное время выполнения).

Можно выделить несколько моментов подхода к распараллеливанию:

- обработкой данных управляет одна программа;
- пространство имен является глобальным, т. е. для программиста существует одна единственная память, а детали структуры данных, доступа к памяти и межпроцессорного обмена данными от него частично скрыты;
- слабая синхронизация вычислений на параллельных процессорах, т. е. выполнение команд на разных процессорах происходит, как правило, независимо и только лишь иногда, по желанию программиста, производится согласование выполнения циклов или других программных конструкций – их синхронизация. Нет гарантии, что в заданный момент времени на всех процессорах выполняется одна и та же машинная команда;
- параллельные операции над элементами массива выполняются одновременно на всех доступных данной программе процессорах.

В рамках данного подхода от программиста не требуется больших усилий по векторизации или распараллеливанию вычислений. Даже при программировании сложных вычислительных алгоритмов можно использовать библиотеки подпрограмм, специально разработанных с учетом конкретной архитектуры компьютера и оптимизированных для этой архитектуры.

Подход, основанный на параллелизме данных, базируется на использовании при разработке программ базового набора операций:

- операции управления данными;
- операции над массивами в целом и их фрагментами;
- условные операции;
- операции приведения;
- операции сдвига;
- операции сканирования;
- операции, связанные с пересылкой данных [4].

Для формализации задачи поиска информационно-независимых фрагментов программы в целях распараллеливания и ускорения работы производят анализ с помощью графов информационных зависимостей. Вершинами графов являются блоки программы. Если блоки программ являются информационно-зависимыми, то они соединяются направленными дугами (или ребрами). Поскольку информационная зависимость однонаправлена, то направление дуги определяет движение данных программы.

Если вершины графа информационных зависимостей отражают обработку только одного элементарного операнда программы, то такой граф называется информационной историей. Информационная история максимально подробно описывает алгоритм программы.

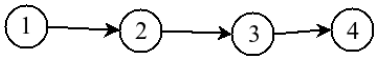

Рис. 1. Последовательный алгоритм

Наиболее простой способ выделения информационно-независимых частей программы – это разбиение на ярусы. Отнесем операнды, работающие только с входными переменными, к операндам первого яруса, операнды, работающие с данными операндов первого яруса, к операндам второго яруса и т. д. Такой вид называют ярусно-параллельной формой программы. Количество ярусов называется длиной критического пути. Например, длина критического пути графа, алгоритм которого показан на рис. 2, равна 6. Операции одного яруса являются гарантированно информационно независимыми и могут выполняться параллельно.

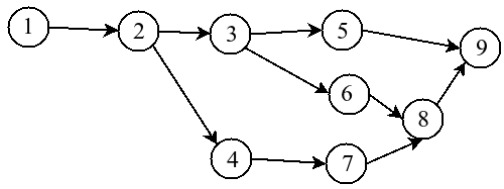

Рис. 2. Сложный алгоритм

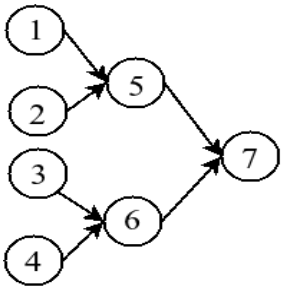

Рис. 3. Алгоритм сдваивания

Для оценки эффективности алгоритма используется понятие степени параллелизма. Степенью параллелизма яруса будем называть количество его независимых операций. В нашем примере степень параллелизма первого яруса равна 4, второго – 2, третьего – 1. В общем случае степень параллелизма q-го яруса сложения n переменных методом сдваивания составит $n/2^q$. Степенью параллелизма всего алгоритма называется отношение количества всех операций к числу ярусов. Для алгоритма сдваивания n переменных степень параллелизма можно вычислить по формуле

$$S = \frac{n-1}{\log_2(n)}. \quad (1)$$

Чем выше степень параллелизма задачи, тем на больших числах процессоров ее можно эффективно запустить. При сложении 8-ми операндов степень параллелизма составит 7/3 (или 2,3(3)). Максимальная степень параллелизма наблюдается на 1-м ярусе и равна 4, т. е. наиболее быстро задача будет решена при использовании 4-х процессоров, но на 2-м этапе будут простаивать 2 процессора, на третьем 3, что с точки зрения экономии электроэнергии не является хорошим результатом. На практике число процессоров может быть и меньше максимальной степени параллелизма какого-либо яруса. Поэтому для написания программ используют различные распределения инструкций по процессорам. Количество ярусов в этом случае увеличивается в целях уменьшения неравномерности загрузки. Незначительно увеличивается время работы алгоритма, но при этом уменьшается простой вычислительных мощностей системы.

Для оценки эффективности параллельных вычислений используют и другие показатели. Ускорением параллельного алгоритма называют отношение времени выполнения задачи на одном узле к времени выполнения на p узлах:

$$S(p) = \frac{T_1}{T_p}, \quad (2)$$

где T_1 – время выполнения задачи на одном процессоре; T_p – время выполнения на p процессорах; $S(p) \le p$. Также используют не зависимую от числа узлов величину эффективности:

$$E(p) = \frac{S(p)}{p}. \quad (3)$$

Теоретически $E(p) \le 1$, но на практике $E(p)$ иногда превышает значение 1 из-за аппаратных особенностей узлов.

На практике крайне редко встречаются программы, состоящие из полностью информационно независимых операций. Часть операций остается информационно зависимой и может выполняться только последовательно. Пусть доля программы, выполняемая на одном процессоре (остальные в этот момент простаивают), составляет α, тогда в формуле (2) $T_p = T_2 + T_3$, где T_2 – время выполнения на одном процессоре; T_3 – время выполнения на p процессорах. Поскольку скорость выполнения на одном процессоре не меняется, то $T_2 = T_1\alpha$, время выполнения параллельного кода на p процессорах $T_3 = \frac{(1-\alpha)T_1}{p}$. Подставляя полученные данные в (2), получаем закон Амдаля:

$$S(p) = \frac{1}{\alpha + (1-\alpha)/p}. \quad (4)$$

Соответственно,

$$E(p) = \frac{1}{\alpha(p+1)+1}. \quad (5)$$

При увеличении числа узлов $S(p) \to \frac{1}{\alpha}$, $E(p) \to 0$. Таким образом, для каждой программы существует свое оптимальное количество процессоров, на котором она будет наиболее эффективно выполняться. Оно зависит от количества информационно независимых входных переменных [1, с. 1].

Формула Амдаля не учитывает времени на подготовку данных и времени на пересылку. Если принять, что время на подготовку данных равно t_d, а время на пересылку данных от одно-

го узла к другому составляет t_c и общее время пересылки линейно растет с увеличением числа узлов (например, для пересылки используется общая среда) [2], то формула (4) принимает вид

$$S(p) = \frac{T_1}{\left(\alpha + \frac{(1-\alpha)}{p}\right)T_1 + t_d + t_c p}, \quad (6)$$

где T_1 – время выполнения задачи на одном процессоре; α – доля операций, выполняемых на одном процессоре; t_d – время, необходимое на подготовку данных; t_c – время, затрачиваемое на пересылку данных между процессорами.

На рис. 4 представлены графики функций (4) и (6). Функция (6) имеет максимум.

Проведя операцию дифференцирования и нахождения максимума, получим

$$p_{\max} = \sqrt{\frac{T_1}{t_c}(1-\alpha)}. \quad (7)$$

Чем больше составляет доля времени пересылки данных от времени выполнения всей программы, тем на меньшем количестве процессоров наступает максимум и увеличение числа узлов только замедлит программу. Вместе с тем необходимость увеличения числа процессоров сильно зависит от доли параллельно выполняющихся инструкций. Чем она больше, тем при большем количестве узлов наступит момент, когда их увеличение замедлит производительность.

Анализ показал, что эффективность применения запуска программы на кластерах сильно зависит от производительности оборудования, в частности от инфраструктуры сети данных, и от структуры программы данных. Кластерные вычислительные системы, даже достаточно мощные, не всегда эффективно использовать для расчетов широкого круга задач. Их оптимальное применение выгодно при решении задач с большими наборами независимых данных, оказывающих слабое влияние друг на друга. При этом результат вычислений получается тем быстрее, чем больше независимых данных обрабатывает задача. Рост числа узлов оптимален, если число входных параметров соизмеримо с числом процессоров или незначительно больше. Из числа оптимальных задач можно выделить задачи обработки изображений, 3D моделирования, обработки видеоинформации, обработки данных геоинформационных систем. В этих направлениях наиболее целесообразно применение распределенных вычислительных систем с большим количеством узлов и быстрой инфраструктурой пересылки потоков данных.

Из данного ряда задач необходимо выделить важное направление, основанное на технологии построения и обработки пространственных моделей территориально-распределенных объектов для автоматизации решения задач оценки и формирования энергетической обеспеченности территории на основе геоинформационных систем. Такие задачи требуют выполнения большого объема односложных операций, которые в большинстве случаев являются информационно независимыми.

Входными данными геоинформационных систем, как правило, является некоторое конечное множество объектов $F = \{f_i\}$, $i \in I$.

Графическое представление модели (либо в виде поверхности, либо в виде таблицы), отражающее обобщенное влияние на территорию объектов одного подмножества F_i, называется монорельефом, который является трехмерным координатно-локализованным графиком, отражающим изменение величины влияния рассматриваемого фактора на всей протяженности территории.

Монорельеф, по сути, является результатом влияния рассматриваемых свойств объектов с учетом территориальной распределенности. Часто в одной точке пространства пересекаются сферы влияния различных объектов. Совокупное влияние в это случае определяется формулой

$$S^0_{x.y} = \Phi_{x,y,j}, \quad j = \overline{1, N} \quad (8)$$

где $S^o_{x,y}$ – общее влияние объектов на точку с координатами x, y; $s_{x,y,j}$ – влияние j-го объекта на точку с координатами x, y; N – число элементов, оказывающих влияние на точку с координатами x, y; Φ – функционал, определяющий взаимодействие между объектами [6, с. 10].

Требуемая производительность вычислительной системы напрямую зависит от функционала, определяющего взаимодействие между объектами. Объекты являются чаще всего информационно независимыми, поэтому применение кластерных технологий для анализа монорельефа наиболее целесообразно.

Заключение

В настоящее время при проектировании территориально-распределенных объектов наиболее важной является задача автоматизации их размещения в пространстве территории с учетом их свойств, влияющих на окружающую среду. Например, это очень актуально при организации инфраструктуры территории. В процессе проектирования может участвовать большое количество объектов, оказывающих значительное взаимное влияние на результат оценки и проектирования. Решение данных задач позволяет оптимизировать территориальную инфраструктуру с учетом энергозатрат и экологической обстановки, что в конечном итоге выльется в сэкономленные денежные средства и сохраненный природный потенциал. А применение кластерных техноло-

гий для решения подобных задач в свою очередь позволит оценить обстановку наиболее точно и с наименьшими затратами применения большего количества объектов.

Список литературы

1. **Многопроцессорные** вычислительные системы и параллельное программирование [электронных ресурс] / URL: http://oldunesco.kemsu.ru/mps/ (дата обращения: 15.05.2011).

2. **Олифер В., Олифер Н.** Компьютерные сети. Принципы, технологии, протоколы. – 4-е изд. – СПб.: Изд-во «Питер», 2010. – С. 87–92.

3. **Антонов А.С.** Параллельное программирование с использованием технологии OpenMP: учеб. пособие. – М.: Изд-во МГУ, 2009. – С. 54–56.

4. **Антонов А.С.** Введение в параллельные вычисления: учеб. пособие. – М.: Изд-во МГУ, 2002. – С. 69.

5. **Филиппенко П.Н.** Обзор в области построения и использования кластерных систем // Информатика, вычислительная техника и инженерное образование. – 2010. – № 1. – С. 19–27.

6. **Ершов В.Н.** Технология построения пространственных моделей для проектирования территориально-распределенных объектов (на примере энергетических систем): автореф. – Кострома, 2002.

7. **Ершов В.Н.** Автоматизация принятия управленческих решений на основе пространственного анализа // Актуальные проблемы науки в АПК: мат-лы межвуз. науч.-практич. конф. 3–4 февраля 2000 г.: в 2 т. – Кострома: Изд-во КГСХА, 2000. – Т. 2. – С. 168–170.

References

1. Mnogoprotsessornye vychislitel'nye sistemy i parallel'noe programmirovanie [Multiprocessor computing systems and parallel programming], URL: http://oldunesco.kemsu.ru/mps/

2. Olifer, V., Olifer, N. Komp'yuternye seti. Printsipy, tekhnologii, protokoly [Computer nets. Principles, technologies, protocols], Sankt-Petersburg: izdatel'stvo «Piter», 2010, pp. 87–92.

3. Antonov, A.S. Parallel'noe programmirovanie s ispol'zovaniem tekhnologii OpenMP [Parallel programming with OpenMP technology], Moscow: izdatel'stvo MGU, 2009, pp. 54–56.

4. Antonov, A.S. Vvedenie v parallel'nye vychis-leniya [Introduction to parallel calculation], Moscow: izdatel'stvo MGU, 2002, p. 69.

5. Filippenko, P.N. Obzor v oblasti postroeniya i ispol'zovaniya klasternykh system [Architecture and use of clusters systems review], in *Informatika, vychis-litel'naya tekhnika i inzhenernoe obrazovanie*, 2010, 1, pp. 19–27.

6. Ershov, V.N. Tekhnologiya postroeniya prostranstvennykh modeley dlya proektirovaniya territorial'no-raspredelennykh ob»ektov (na primere energeticheskikh sistem) [Technology design of spacial model to design geographically distributed objects (energy power system)], avtoreferet, Kostroma, 2002.

7. Ershov, V.N. Avtomatizatsiya prinyatiya upravlencheskikh resheniy na osnove prostranstvennogo analiza [Managerial solution automation on the basis of spatial analysis], in *Aktual'nye problemy nauki v APK: mat-ly mezhvuz. nauch.-praktich. konf. 3–4 fevralya 2000 g.*, v 2 t, t. 2, Kostroma: izdatel'stvo KGSKhA, 2000, pp. 168–170.

Кунгурин Юрий Валентинович,
Костромской государственный университет имени Н.А. Некрасова,
начальник управления информатизации,
телефон (4942) 39-16-11,
e-mail: urak@ksu.edu.ru