АНАЛИТИЧЕСКАЯ МОДЕЛЬ ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ МНОГОПРОЦЕССОРНОЙ ОБРАБОТКИ ДАННЫХ ДЛЯ НАБОРА ПАРАЛЛЕЛЬНЫХ АЛГОРИТМИЧЕСКИХ СТРУКТУР

К.В. Герценбергер,
младший научный сотрудник, Объединенный институт ядерных исследований, г. Дубна

Е.В. Чепин,
кандидат технических наук, доцент Национального исследовательского ядерного университета (МИФИ), Московский физико-технический институт (государственный университет) (МФТИ)

Адрес: Московская область, г. Дубна, ул. Жолио-Кюри, д. 6
E-mail: k.gertsenberger@gmail.com, evchepin@gmail.com

Эффективность параллельной обработки данных напрямую зависит от алгоритма распараллеливания и типа аппаратной архитектуры. Важно, чтобы оценка прогнозируемой эффективности распараллеливания задачи для конкретной аппаратной платформы происходила на наиболее раннем этапе разработки. Для этой цели предлагается разработанная аналитическая модель предсказания производительности будущей программно-аппаратной системы для трех основных параллельных алгоритмических структур – алгоритмов распараллеливания.

Ключевые слова: аналитическая модель, предсказание производительности, распределенные вычисления, параллельная аппаратная архитектура, параллельная алгоритмическая структура, параллельное программное обеспечение.

1. Введение

Традиционные низкоуровневые методы и инструменты параллельного программирования зачастую приводят к сложному, непереносимому, немасштабируемому программному коду, делают задачу создания программ узкоспециализированной, нетривиальной, трудоемкой с большими временными затратами, не защищают как от программных ошибок, так и низкой эффективности. В процессе развития программирования уровень абстракции неуклонно повышается. В настоящее время активно ведутся исследования в области создания высокоуровневых инструментальных средств параллельного программирования.

В контексте повсеместного распространения многоядерных процессоров и многопроцессорных систем, использование разработанной авторами статьи высокоуровневой методики, основанной на параллельных алгоритмических структурах, автоматизирующей основные этапы и позволяющей разработчику с невысокими знаниями параллельного программирования быстро реализовывать прикладные программы, успешно функционирующие в многопроцессорной среде, является крайне актуальным. Предложенный подход, представленный в работе [4], содержит следующие этапы разработки (курсивом выделены этапы, выполняющиеся полностью автоматически) параллельного программного обеспечения:

1. Совместное задание схемы многопроцессорной/многоядерной системы и цепочки функций обработки данных.

2. Выбор параллельных алгоритмических структур из набора для распараллеливаемых функций в соответствии с логикой их работы и зависимостью по данным.

3. *Проведение аналитического и/или имитационного моделирования планируемой системы*. В случае неудовлетворительных модельных характеристик или поиска оптимального решения корректировка схемы многопроцессорной системы, параллельного алгоритма и проведение нового моделирования.

4. Сопоставление исходного кода функций с блоками выбранных параллельных алгоритмов.

5. Проведение автоматической генерации параллельного кода функций.

6. Запуск полученного программного обеспечения на реальной системе при помощи планировщика и оценка эффективности решения.

В данном подходе разработчикам необходимо задать схему аппаратной и функциональной части, сопоставить параллельные алгоритмические структуры из заданного набора с распараллеливаемыми функциями, а остальные этапы автоматизированы. Такой подход освобождает программистов от разработки параллельных алгоритмов, выбора инструментария для используемой аппаратной платформы, написания параллельного исходного кода, отладки и ручного таймирования реализованной системы, сложного запуска на многопроцессорной системе.

Как и для других подходов разработки параллельного программного обеспечения, одним из наиболее важных моментов является начальный этап предсказания эффективности планируемой программно-аппаратной связки. Точный метод предсказания дает возможность проектировщику оценить производительность параллельного алгоритма до реализации и исследовать эффективность при выборе альтернативного алгоритма или изменения параметров аппаратной части, например, количества процессоров и пропускной способности сети, может помочь в поиске точек, критичных для производительности.

На данный момент наиболее распространённым подходом при проектировании параллельных платформ является использование экспертных оценок. Этот подход, безусловно, позволяет в некоторой степени оптимизировать структуру проектируемых систем. Однако решения носят субъективный характер, поэтому для сложных распределённых систем полученные оценки и решения дают значительные отклонения от оптимальных значений. В случае ошибки выбора аппаратной платформы разработчикам приходится закупать другое оборудование, в случае неэффективного алгоритма распараллеливания – корректировать алгоритмическую часть и проводить разработку заново.

При проектировании и оптимизации вычислительных систем часто используют математическое моделирование. Моделирование планируемой системы дает возможность оценить максимальную производительность при имеющихся ресурсах, найти критичные точки аппаратной архитектуры, избежать ошибок проектирования, основанных на экспертной оценке. В ходе реализации методики, основанной на параллельных алгоритмических структурах, были разработаны аналитическая и имитационная модель параллельной обработки на многопроцессорной (многоядерной) системе.

Аналитическое моделирование предлагает более простой и формализованный подход для представления системы, более приспособлено к поиску оптимального решения. Имитационная модель, в свою очередь, не требует такого числа упрощений и допущений, позволяет получить более точную оценку работы сложной системы как единого целого. В данной статье представлена разработанная аналитическая модель. В качестве основного результата модели выбраны следующие характеристики: коэффициент ускорения, эффективность и масштабируемость. Так как представленные характеристики зависят от параллельного алгоритма, то

в статье проводится оценка эффективности параллельной обработки для каждого класса параллельных алгоритмов.

2. Основные типы параллельных алгоритмических структур

Разработчики в большинстве случаев используют существующие алгоритмы распараллеливания, которые давно известны, или их незначительные модификации. В связи с этим использование готового, но расширяемого набора параллельных алгоритмических структур для автоматизации является крайне актуальным и перспективным на сегодняшний день. В ходе анализа существующих алгоритмов распараллеливания был сформирован следующий набор параллельных алгоритмических структур для аналитической модели:

1. Точечный алгоритм реализует операцию над однородным множеством величин. Функция, применяемая к элементу множества, не зависит от остальных элементов. Неперекрывающиеся сегменты данных в этом случае распределяются по процессорам и независимо обрабатываются.

2. Локальный алгоритм (функция, применяемая к элементу множества, зависит от нескольких соседних элементов) аналогичен точечному алгоритму с тем отличием, что множество данных разбивается на перекрывающиеся фрагменты [2].

3. В алгоритме параллельной конвейеризации каждая ступень конвейера создается на отдельном процессоре, который производит обработку данных, полученных с предыдущей ступени конвейера. Конвейерная обработка сама по себе не обладает масштабируемостью, так как число ступеней фиксировано, однако в ряде случаев позволяет ускорить обработку для плохо распараллеливаемых задач.

4. Редукция использует ассоциативную операцию ко всем элементам множества, сводя множество к одному элементу. Алгоритм параллельной редукции заключается в распределении по процессорам неперекрывающихся сегментов данных, независимому выполнению редукции до одного элемента и сборке результатов для редукции на управляющем процессе (потоке).

5. Парадигма «разделяй-и-властвуй» осуществляет рекурсивную декомпозицию задачи на две или более подзадачи того же типа, но меньшего размера, с последующим комбинированием частичных результатов в конечный результат.

6. Параллельный алгоритм волновой схемы заключается в вычислении по каждому направлению волны одновременно различными процессорами. Зачастую данный алгоритм также сводится к параллельной рекурсивной схеме.

Предложенный набор содержит подходы распараллеливания как по данным (например, локальный алгоритм, параллельная редукция), так и по задачам (например, конвейер). Очевидно, что список не является полным, в будущем планируется его расширение новыми элементами. Стоит отметить, что, например, в области обработки изображений и сигналов он охватывает больше 80 % операций. Однако, первые три алгоритма из данного списка наиболее часто, на наш взгляд, встречаются в практических задачах, связанных с моделированием по методу Монте-Карло, с оптимизацией на основе генетических алгоритмов, с задачами обучения и распознавания. Примеры задач из различных прикладных областей, для которых целесообразно использование таких моделей для распараллеливания ПО, можно найти в статьях [6-8].

3. Аналитическая модель предсказания производительности

Для построения моделей параллельной обработки на многопроцессорной системе используется тот факт, что любая аппаратная схема сводится к набору вычислительных узлов, выполняющих параллельную обработку данных, хранилища данных, а также коммутативной среды, связывающей вычислительные узлы между собой и хранилищем данных. В аналитической модели производится замена схемы гетерогенной сети аналогичной по структуре гомогенной сетью, у которой все вычислительные узлы и коммуникационные средства равны по производительности самому медленному узлу и самой медленной связи исходной схемы. В силу симметричности выбранных параллельных алгоритмических структур и закона Амдала в данном случае такие схемы эквивалентны для аналитического расчета. Исторически сложилось так, что время чтения/записи с жесткого диска заметно превышает время доступа к оперативной памяти, поэтому времена работы с оперативной памятью и кэшем не учитываются.

Выполнение обработки на многопроцессорной системе определяется параллельной алгоритми-

ческой структурой, поэтому анализируя все множество алгоритмов, можно построить модель для предсказания производительности, результатом работы которой будут выбранные характеристики. Используя формулу коэффициента ускорения можно спрогнозировать масштабируемость заданной системы следующим способом: для каждого числа процессоров от 1 до p рассчитывается соответствующий коэффициент ускорения. По полученной зависимости ускорения от числа процессоров аппаратной платформы можно оценить масштабируемость параллельной обработки на выбранной системе. Ниже подробно рассмотрены аналитические модели первых трех типов алгоритмов из приведенного выше списка.

3.1. Точечный алгоритм распараллеливания

а) Многопроцессорная система с распределенной памятью.

В структуре алгоритма можно выделить три этапа: чтение данных, независимая обработка сегментов на p-процессорах и запись полученного результата. Если обозначить количество узлов за P_{node}, среднюю скорость линейного чтения/записи носителя данных – W, пропускную способность сети – B, объем обрабатываемых данных – n, а T_1 – время обработки на одном процессоре, то можно математически представить коэффициент ускорения и эффективности для каждой параллельной алгоритмической структуры. Время последовательной обработки определяется следующим выражением:

$$T_{посл}(n)=T_{посл.чтение}(n)+T_1(n)+T_{посл.записи}(n), \qquad (1)$$

где $T_{посл.чтение}(n)=T_{посл.записи}(n)=n/W$.

Таким образом, время последовательного выполнения функции можно представить следующим образом:

$$T_{посл}(n)=2\cdot n/W+T_1(n). \qquad (2)$$

Согласно точечному алгоритму распараллеливания, время независимой обработки уменьшится в P_{node} раз и составит $T_1(n)/P_{node}$.

Скорость доступа к данным определяется соотношением пропускной способности сети и скоростью чтения/записи носителя информации. В случае, если пропускная способность сети выше скорости линейного чтения/записи, то производительность определяется медленным элементом: временем линейного чтения/записи W. В противном случае, носитель информации успевает считывать и передавать в сеть данные, пока последняя передает их рабочему узлу, поэтому пропускная способность сети B является определяющей. Таким образом, на первом шаге анализируется, какой компонент чтения/записи более медленный: сеть или носитель информации. В качестве скорости чтения/записи данных S выбирается минимальная характеристика: либо пропускная способность сети, либо средняя скорость линейного чтения/записи.

Время чтения определяется по приходу всех байт на все процессоры и равно:

$$T_{пар.чтения}(n)=T_{задержка1}+\frac{n}{S}\approx\frac{n}{S}, \qquad (3)$$

где $T_{задержка1}$ – время, через которое начинается передача первого байта данных. Это время мало по сравнению со временем чтения всего объема данных, поэтому в дальнейшем подобные характеристики учитываться не будут.

Чтение байт массива данных процессорами происходит последовательно (выбрано для снижения числа коллизий). К моменту окончания обработки данных процессором, который получил байты последним, процессоры, принявшие свой сегмент информации раньше, в связи с тем, что чтение и запись происходят примерно с одной скоростью, уже успеют отослать свои результаты. Таким образом,

$$T_{пар.записи}(n)=\frac{n}{P_{node}\cdot S}. \qquad (4)$$

Подставив полученные выражения в формулу коэффициента ускорения, получается:

$$K_p(n)=\frac{2\cdot\frac{n}{W}+T_1(n)}{\frac{n}{S}+\frac{n}{S\cdot P_{node}}+\frac{T_1(n)}{P_{node}}}=\frac{S\cdot P_{node}\cdot(2\cdot\frac{n}{W}+T_1)}{n\cdot(P_{node}+1)+S\cdot T_1}. \qquad (5)$$

Видно, что при неограниченном росте числа процессоров коэффициент ускорения асимптотически стремится к величине

$$S\cdot\left(\frac{2}{W}+\frac{T_1}{n}\right).$$

Оценить масштабируемость системы можно следующим образом: пока коэффициент ускорения увеличивается, система масштабируется. Как

только ускорение начнет уменьшаться, масштабируемость перестает иметь место. Таким образом, определив число процессоров, соответствующее максимуму коэффициента ускорения, можно оценить масштабируемость программно-аппаратной связки. Экстремум определяется приравниванием первой производной коэффициента ускорения по числу процессоров к нулю:

$$\frac{dK_p}{dP}=0.$$

В данном случае, из формулы видно, что коэффициент ускорения с ростом числа процессоров будет постоянно расти. С другой стороны, эффективность параллельного выполнения будет уменьшаться, поэтому имеет смысл увеличивать количество процессоров до тех пор, пока эффективность будет выше некоторого предела. Эффективность для локального алгоритма определяется выражением:

$$E_p(n)=\frac{K_p(n)}{P_{node}}=\frac{S\cdot(2\cdot\frac{n}{W}+T_1)}{n\cdot(P_{node}+1)+S\cdot T_1}. \qquad (6)$$

Так как формулы коэффициента эффективности определяются путем простого деления ускорения на число вычислительных узлов или процессоров, поэтому далее они приводиться не будут.

Иногда при распараллеливании обработки данных используют подход, в котором запись осуществляется не параллельно, а обработанные данные передаются управляющему процессу, который производит последовательную запись результата в файл. В этом случае время параллельной записи увеличивается, но отсутствует необходимость синхронизации записи в один файл. Для точечного алгоритма на многопроцессорной системе с распределенной памяти время параллельной записи увеличится до величины

$$\frac{n}{P_{node}\cdot B}+\frac{n}{W_S},$$

где W_s – пропускная способность канала для записи в файл. Если файл сохраняется на узле, собирающем локальные данные, то $W_s=W$, в противном случае, $W_s=S$. Отсюда, для приведенного примера коэффициент ускорения составит:

$$K_p(n)=\frac{2\cdot\frac{n}{W}+T_1(n)}{\frac{n}{S}+\frac{n}{P_{node}\cdot B}+\frac{n}{W_S}+\frac{T_1(n)}{P_{node}}}=\frac{S\cdot P_{node}\cdot(2\cdot\frac{n}{W}+T_1)}{n\cdot\left(P_{node}+\frac{S}{B}+\frac{P_{node}\cdot S}{W_S}\right)+S\cdot T_1}. \qquad (7)$$

б) Многопроцессорная система с разделяемой памятью.

Расчет модели многопроцессорной системы с разделяемой памятью отличается от первого варианта отсутствием сетевого соединения между узлами и носителем данных:

$$K_p(n)=\frac{P\cdot(2\cdot n+W\cdot T_1)}{n\cdot(P+1)+W\cdot T_1}. \qquad (8)$$

3.2. Локальный алгоритм

Время последовательного выполнения аналогично определяется выражением (2).

а) Многопроцессорная система с распределенной памятью.

Время параллельного чтения увеличивается на время, необходимое для дополнительного получения процессорами перекрывающихся частей. Возможно перекрытие слева, справа и с обеих сторон, поэтому время параллельного чтения определяется следующим выражением:

$$T_{пар.чтения}(n)=\frac{n+(P_{node}-1)\cdot(L_l+L_r)}{S}, \qquad (9)$$

где L_l – длина перекрытия слева, L_r – длина перекрытия справа (0, если нет). Если предположить, что время обработки прямо пропорционально длине обрабатываемого массива данных, то:

$$T_{пар.обработки}(n)=\frac{T_l(n)}{P_{node}}+T_1(n)\cdot\frac{(L_l+L_r)}{n}. \qquad (10)$$

Время параллельной записи, как у точечного алгоритма, равно

$$\frac{n}{P_{node}\cdot S}.$$

Таким образом,

$$K_p(n)=\frac{S\cdot P_{node}\cdot(2\cdot\frac{n}{W}+T_1)}{P_{node}\cdot n\cdot(1+(P_{node}-1)\cdot(M_l+M_r))+S\cdot T_l\cdot(1+P_{node}\cdot(M_l+M_r))+n}. \qquad (11)$$

где $M_l=\frac{L_l}{n}$ – коэффициент перекрытия слева,

а $M_r=\frac{L_r}{n}$ – перекрытия справа.

б) аналогично для многопроцессорной системы с разделяемой памятью получается:

$$K_p(n)=\frac{P\cdot(2\cdot n+T_1\cdot W)}{P\cdot n\cdot(1+(P-1)\cdot(M_l+M_r))+W\cdot T_1\cdot(1+P\cdot(M_l+M_r))+n}. \quad (12)$$

3.3. Алгоритм параллельной конвейеризации

На *рисунке 1* представлено обозначение ступеней конвейеров и массивов данных, пересылаемых ступенями.

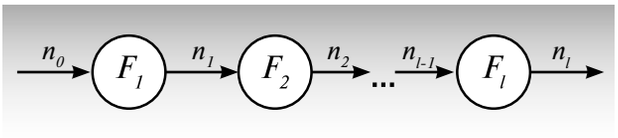

Рис. 1. Конвейерная обработка данных.

Время последовательной обработки данных размером N за $k=\frac{N}{n_0}$ операций можно представить следующим выражением:

$$T_{посл}(N)=k\cdot(T_{посл.чтение}(n_0)+T_1(n_0)+T_{посл.записи}(n_l)). \quad (13)$$

Времена последовательного чтения и записи равны n_0/W и n_l/W соответственно.

$$T_{посл}(N)=\frac{k\cdot n_0+k\cdot n_l}{W}+k\cdot T_1(n_0)\cdot \quad (14)$$

а) Многопроцессорная система с распределенной памятью.

Для оценки эффективности алгоритма предполагается, что одновременно могут приниматься и передаваться данные. В противном случае, пропускная способность коммуникационной среды будет примерно в два раза ниже. Так как время конвейерной обработки k операций складывается из времени одного прохода по конвейеру и последующему k-1 срабатыванию самой медленной ступени конвейера, тогда:

$$T_{пар}(N)=T_1(n_0)+\frac{(k-1)\cdot n_{x-1}+\sum_{m=0}^{l} n_m}{S}+(k-1)\cdot T_{fx}, \quad (15)$$

где x – номер самой медленной ступени конвейера, то есть

$$T_{fx}+\frac{n_{x-1}}{S}=max(T_{fi}+\frac{n_{i-1}}{S}),\ i=1,2...l.$$

Подставив выражения в формулу коэффициента ускорения, получается:

$$K_p(n)=\frac{\frac{n_0+n_l}{W}+T_l}{\frac{n_{x-1}-\frac{n_{x-1}}{k}+\sum_{m=0}^{l}\frac{n_m}{k}}{S}+T_{fx}+\frac{T_l-T_{fx}}{k}}. \quad (16)$$

б) Многопроцессорная система с разделяемой памятью.

В этом случае отсутствует задержка передачи по сети промежуточных данных, поэтому выражение (16) сводится к следующему:

$$K_p(n)=\frac{\frac{n_0+n_l}{W}+T_1}{\frac{n_0+n_l}{k\cdot W}+T_{fx}+\frac{T_1-T_{fx}}{k}}, \quad (17)$$

где $T_{fx}=max(T_{f1}+\frac{n}{S},T_{fi},T_{fl}+\frac{n_l}{S})$, $i=2,3,...,l-1$.

4. Пример сравнения результата моделирования с практическими значениями

Адекватность разработанной модели была практически проверена в ряде проектов для алгоритмов обработки изображений и сигналов. В работе [5] приводится сравнение модельных и практических значений коэффициента ускорения для КИХ-фильтра цифрового сигнала. КИХ-фильтр был реализован с помощью секционированной свертки и ее вычисления через быстрое преобразование Фурье. Эта операция по степени зависимости данных относится к локальной алгоритмической структуре. Для оценки эффективности алгоритма проводилось таймирование фильтра обработки сигнала размером 227 мегабайт на кластерной системе. Зависимость модельного и практического коэффициента ускорения от количества процессоров показало, что максимальное отклонение аналитической модели от практических значений составляет в данном случае 6,4 %.

5. Заключение

В данной статье кратко представлена методика автоматизированного получения параллельного программного обеспечения. Одним из ключевых моментов, как предложенной высокоуровневой методики, так и других инструментов разработки параллельных решений, должен являться этап предсказания производительности обработки данных для планируемой программно-аппаратной

системы. В данной статье представлена разработанная аналитическая модель для оценки эффективности программно-аппаратной платформы до ее реализации. Для каждой параллельной алгоритмической структуры показан вывод основных характеристик производительности. При добавлении нового параллельного алгоритма в набор необходимо разработать оценку в соответствии с приведенной схемой.

Разработанная аналитическая модель успешно применена в ряде проектов [1,3,5]. Прогнозирование эффективности программно-аппаратной связки позволило не только оценить ускорение и масштабируемость будущей системы до реализации, но и определить узкие места аппаратной платформы, сделать выбор оборудования до реализации. Максимальное отклонение результатов аналитической модели в проектах составило 11 %. В будущем планируется расширить аналитическую модель новыми параллельными алгоритмическими структурами, отказаться от ряда упрощений для более точного расчета характеристик производительности. ■

Литература

1. Богин И.В., Герценбергер К.В., Чепин Е.В. Реализация параллельной обработки цифровых изображений для библиотеки IPPLab MSCo. // Науч. сессия МИФИ-2006: Сб. науч. Тр. М.: МИФИ, 2004 - Т. 12 - С.139-140.
2. Герценбергер К.В. Методы параллельной обработки изображений и сигналов в зависимости от локальности вычислений // Сб. Тр. Седьмой международной научно-практической конференции «Исследование, разработка и применение высоких технологий в промышленности». Т. 2. СПб.: Изд-во Политехн. ун-та, 2009 – С.49-50.
3. Gertsenberger K.V., Chepin E.V. Implementation parallel processing of digital signals on cluster system // In: Proc. of the International Workshop on Computer Science and information technologies (CSIT'2007), Bashkortostan, 2007. Vol. 1. Ufa State Aviation Technical University, 2007 – P.152-155.
4. Gertsenberger K.V., Chepin E.V. Using a CASE-oriented approach for parallel software development. In: Proc. of the International Workshop on Computer Science and information technologies (CSIT'2008), Antalya, Turkey, 2008. Vol. 1. Ufa State Aviation Technical University, 2008 – P/63-68.
5. Герценбергер К.В., Чепин Е.В. Использование параллельных алгоритмических структур для автоматизации процесса разработки параллельного программного обеспечения. Труды МФТИ.- 2011 (находится в печати).
6. Ошмарин Д. В. Распределение канальных ресурсов в сетях когнитивного радио на основе теории игр. // Бизнес-информатика. - 2010. № 4. С. 38-45.
7. Голубев С. В. Распознавание структурированных документов на основе машинного обучения. // Бизнес-информатика. - 2011. № 2. С. 48-55.
8. Силантьев Д. А. Оценка необходимого размера свертки биометрического образца для обеспечения заданных параметров надежности биометрической системы идентификации. // Бизнес-информатика. - 2009. № 3. С. 21-23.