Прогнозирование характеристик эффективности выполнения DVM-программ на кластере*

М.С. Клинов

Прогнозирование основано на моделировании параллельного выполнения программы на кластере с заданными характеристиками. Помимо характеристик кластера входной информацией для прогнозирования является последовательность обращений к системе поддержки выполнения DVM-программ. Это позволяет прогнозировать характеристики как по трассе выполнения DVM-программы на инструментальной ЭВМ, так и в процессе автоматического распараллеливания последовательной программы еще до генерации параллельной программы. Кроме того, в данной работе описывается алгоритм поиска оптимальной конфигурации процессоров, на которой спрогнозированное время работы DVM-программы будет минимальным.

1. Введение

При разработке программы программист, как правило, преследует одну из двух целей – обеспечить решение задачи в приемлемые сроки, либо создать программу, способную эффективно решать на различных высокопроизводительных ЭВМ задачи определенного класса. Для получения эффективной программы необходима отладка эффективности, которая состоит из анализа эффективности уже написанной программы, внесения правок в программу, и анализа эффективности модифицированной программы. Автоматизации поддается этап анализа эффективности выполнения программ на интересующих пользователя ЭВМ. Для анализа эффективности необходимо предоставить пользователю набор характеристик выполнения (не только время выполнения), кроме того, для сложных программ недостаточно иметь характеристики выполнения всей программы целиком, а требуется детализировать их применительно к отдельным частям программы.

Вычисление характеристик эффективности необходимо не только для отладки эффективности программ пользователя, но и для выбора лучших вариантов распараллеливания программы (в системах автоматического или автоматизированного распараллеливания программ).

Вычисление характеристик эффективности может производиться как по результатам реальных запусков программ на целевой ЭВМ или инструментальной ЭВМ похожей архитектуры, так и путем прогнозирования. Под целевой ЭВМ будем понимать машину, на которой планируется запускать программу на счет. Инструментальная машина используется для отладки программы (сбор необходимой информации и запуски инструментов отладки). Анализ эффективности по результатам реальных запусков имеет ряд недостатков: для сбора необходимой информации используется высокопроизводительная ЭВМ, запуски программ обычно связаны с большими временами ожидания начала выполнения, характеристики выполнения реальных запусков нестабильны (различаются от запуска к запуску, особенно при использовании большого количества узлов кластера). Для прогнозирования используется рабочая станция и при необходимости незначительное количество узлов доступной ЭВМ (например, когда программе не хватает памяти для запуска на одном процессоре). Путем прогнозирования получаются более стабильные характеристики, чем при реальных запусках, что позволяют быстрее оценить влияние модификаций, направленных на повышение эффективности программы.

При анализе эффективности программы пользователь не обязательно должен запускать ее с тем большим объемом вычислений, который будет характерен для использования программы при решении реальных задач. Он может ограничить количество регулярно повторяющихся внешних итераций, например, до одной или двух итераций, предполагая, что эффективность выполнения этого

* Работа поддержана грантом Президента РФ № НШ-383.2006.9 для ведущих научных школ и грантами РФФИ № 05-01-00678 и № 05-07-90026.

цикла будет оставаться неизменной при росте числа итераций. Такой метод применим в обоих подходах к анализу эффективности.

Метод прогнозирования можно использовать еще до получения текста программ, что позволяет оперировать с большим количеством вариантов и разными размерами решаемой задачи, избежав при этом многочисленных компиляций, запусков и ожиданий результатов. В системах автоматического распараллеливания программ возникает огромное количество вариантов, поэтому в этом случае трудно обойтись без применения методов прогнозирования.

Варианты программ необходимо сравнивать по минимальным временам их выполнения на выделенном количестве процессоров, и, следовательно, необходимо находить оптимальное количество используемых процессоров. В случаях использования в программе многомерных распределений данных возникает понятие многомерной решетки процессоров. В таких случаях все процессоры логически образуют решетку, и требуется определять не только оптимальное количество процессоров, но и конфигурацию этой решетки.

Использование многомерных распределений позволяют более эффективно решать многие задачи. Во-первых, появляется возможность использовать больше процессоров, чем при одномерном распределении, при котором количество процессоров ограничено размером одного измерения массива. Во-вторых, для многих программ при многомерном распределении повышается отношение количества вычисляемых на процессоре элементов (объема многомерного параллелепипеда) к количеству пересылаемых с других процессоров граничных элементов (площади поверхности многомерного параллелепипеда).

Для разработки программ, способных выполняться на современных высокопроизводительных ЭВМ, можно использовать множество разных языков параллельного программирования, и созданные для этих языков различные инструменты. Одной из таких систем разработки программ является система DVM [1], разработанная в ИПМ им М.В.Келдыша РАН. В основу этой системы положена языковая модель, которая позволяет легко выражать функциональный параллелизм и параллелизм данных в научных и инженерных приложениях для ЭВМ с массовым параллелизмом. Модель объединяет в себе многие черты моделей OpenMP и HPF и ориентирована на создание мобильных приложений, которые могут эффективно выполняться на ЭВМ с распределенной памятью и ЭВМ с разделяемой памятью. Языки параллельного программирования Fortan-DVM и C-DVM являются расширениями стандартных языков Fortran и C директивами параллелизма. Программы, написанные на этих языках, будем называть DVM-программами. В составе системы имеются инструменты отладки DVM-программ (динамический контроль корректности и сравнительная отладка).

Целью данного исследования была разработка алгоритмов прогнозирования характеристик выполнения DVM-программ на кластерах, а также разработка алгоритмов автоматизированного поиска оптимальной конфигурации процессоров.

2. Алгоритмы прогнозирования характеристик выполнения

Требуется разработать алгоритмы прогнозирования, которые были бы применимы как для отладки эффективности DVM-программ, так и для оценки разных вариантов распараллеливания (до получения текста DVM-программ). Прогнозирование должно осуществляться путем моделирования на инструментальной ЭВМ параллельного выполнения программы на кластере, для которого известны основные его характеристики (производительность процессоров и параметры коммуникационной среды).

2.1 Особенности предлагаемого подхода

Для решения этой задачи предлагается рассматривать параллельное выполнение программы не с точностью до выполнения операторов, а в виде последовательности работы более крупных частей программы.

Параллельное выполнение DVM-программы организуется с помощью функций системы поддержки выполнения, которая является частью DVM-системы. Вызовы этих функций вставляются при компиляции DVM-программы. Участки программы от одного вызова такой функции до другого

считаются с точки зрения прогнозирования единым целым, детали устройства таких участков программы при моделировании не важны. В таком случае участки программы между вызовами функций поддержки характеризуются только временем их выполнения. Последовательность вызовов и времена выполнения участков программы между вызовами можно технически представить в виде последовательности вызовов, в которой каждый вызов будет характеризоваться еще и временем выполнения участка программы перед ним. Непосредственно перед выходом из программы всегда есть вызов функции поддержки, поэтому описанное действие является правомерным. Таким образом, выполнение DVM-программы на каждом процессоре кластера можно представить в виде последовательности вызовов функций поддержки, которые характеризуют не только вызовы, но и участки программы между вызовами.

Функции системы поддержки устроены таким образом, что последовательности вызовов функций поддержки на одном количестве процессоров (даже на одном процессоре) достаточно для получения последовательности вызовов на любом другом количестве процессоров. Выполнение программы можно представить в виде одной последовательности вызовов, и прогнозировать выполнение этой последовательности на любом количестве процессоров.

Последовательность вызовов функций поддержки можно получить не только при выполнении имеющейся DVM-программы, но и при автоматическом распараллеливании последовательной программы еще до получения готовой DVM-программы – по DVM-директивам каждого варианта распараллеливания. Для этого необходимо знать размеры массивов, параметры циклов (начальное значение, конечное значение, шаг цикла) и свойства условных операторов (вероятность выполнения каждой ветки или другие свойства).

Моделирование параллельного выполнения на кластере должно опираться на модель оценки времен вычислений и модель оценки времен коммуникаций.

Для оценки времен вычислений частей программы предлагается использовать одну из двух моделей. Первая модель основывается на замерах времен на инструментальной ЭВМ и пересчете их для целевой ЭВМ, считая при этом, что все операции станут в одинаковой степени быстрее или медленнее выполняться. Вторая модель использует аналитические оценки времен (например, по количеству операций, операторов и т.п.) и последующий их пересчет для целевой ЭВМ. Вторая модель более грубая, но не требует запуска программы, и поэтому работает быстрее. Замеры времен на инструментальной ЭВМ осуществляются путем накопления файлов трассы выполнения программы на любом количестве процессоров. В трассах собрана информация о последовательности вызовов функций системы поддержки, времена их работы, и времена работы участков программ между вызовами.

Оценка времен коммуникаций опирается в предлагаемом подходе на многоуровневую модель. На самом нижнем уровне находятся ядра одного процессора. Нижние уровни соединяются между собой через новый уровень, который представлен несколькими каналами связи с заданными для каждого латентностью и временными расходами на передачу каждого байта информации. Количество уровней не ограничено. Эта модель отражает тот факт, что коммуникации между ядрами одного процессора производятся быстрее, чем между процессорами; коммуникации между процессорами одного узла ЭВМ происходят быстрее, чем между разными узлами; и т.д.

Для ускорения моделирования параллельного выполнения вариантов распараллеливания было решено воспользоваться предположением (эвристикой), что значения соответствующих характеристик параллельного выполнения каждой итерации непараллельного цикла, начиная со второй итерации, будут близкими по величине, и их можно считать одинаковыми. В таком случае можно выполнить первую итерацию цикла (внутри нее могут быть параллельные и непараллельные циклы), запомнить текущие значения всех характеристик, выполнить вторую итерацию цикла и вычислить разность, которая говорит о характеристиках, получаемых при каждой последующей итерации непараллельного цикла. Полученную разность умножить на недостающее количество итераций и сложить со значениями характеристик первой итерации.

2.2 Алгоритмы моделирования

DVM-программа или вариант распараллеливания представлены в виде последовательности

вызовов функций системы поддержки выполнения DVM-программ. Эта последовательность характеризует не только сами вызовы, но и участки программы между вызовами. Функции поддержки устроены так, что по имени функции всегда можно понять, будут ли распределены вычисления в предшествующем вызову участке программы, и точно определить, на каких процессорах будет выполнены те или иные вычисления и коммуникации.

Моделирование выполнения непараллельных участков заключается в добавлении времени работы этого участка к временам работы всех процессоров, выполняющих данных участок. Наличие непараллельных участков в программе приводит к увеличению значению характеристики ”недостаточный параллелизм”, поскольку одни и те же вычисления производятся на многих процессорах.

При работе алгоритма моделирования выполнения параллельных циклов определяется количество итераций цикла, выполняемое каждым процессором, и осуществляется соответствующее разделение общего времени работы цикла по процессорам. В случаях дублирования вычислений увеличивается значение характеристики недостаточного параллелизма.

В DVM-программе могут встретиться несколько видов коммуникационных операций: обмен данными через теневые грани массивов (расширения локальной части массива для каждого процессора, которые служат буферами для приема данных с соседних процессоров), редукционные операции, копирование секций массивов, удаленный доступ к секциям массива, обмен данными при организации конвейерного выполнения витков цикла.

Для моделирования коммуникаций используются следующие алгоритмы.

Для операций обновления теневых граней определяется размер грани (в байтах), заполняется матрица пересылок, элементы которой задают объем пересылки с одного процессора на другой. Затем моделируется запуск пересылок данных по всей матрице, определяются загрузки коммуникационных каналов кластера и времена завершения операции обновления теневых граней для каждого процессора в отдельности. В момент завершения операции обновления теневых граней будут вычислены для всех процессоров времена, потраченные на ожидания завершения коммуникаций, и времена совмещения этих коммуникаций с вычислениями.

Для редукционных операций определяется количество байт, которые занимает редукционная переменная. Сбор значений от всех процессоров, участвующих в операции редукции, осуществляется через главный процессор (с минимальным номером из участвующих в операции). Вычисление редукционного значения по собранным значениям занимает незначительное время (и считается нулевым). После этого запускается рассылка полученного значения обратно процессорам, и вычисляется время завершения данной редукционной операции. По нему вычисляется время ожидания завершения редукционной операции на каждом процессоре.

При копировании определяется объем передаваемых данных и заполняется матрица пересылок. Дальнейшее моделирование похоже на моделирование обновления теневых граней. Моделирование удаленного доступа к секциям массива похоже на моделирование копирования.

Конвейер используется для распараллеливания циклов с регулярными (длина зависимости не более некоторой константы) зависимостями витков цикла по данным. Алгоритм моделирования конвейера начинается с определения количества витков цикла (кванта конвейера), которые будут выполняться процессорами на каждом шаге конвейера. Размеры кванта конвейера определяются по той же схеме, что используется в системе поддержки выполнения DVM-программ для расчета параметров конвейера. Характеристики выполнения таких циклов получаются путем упрощенного моделирования работы конвейера. Все процессоры представлены в виде логической решетки процессоров, которая используется для многомерных распределений данных. Решетка задает, на сколько процессоров распределяется каждое измерение данных. Для упрощенного моделирования работы конвейера по каждому измерению процессорной решетки берется только по 4 процессора: 3 первых и последний. Загрузка других процессоров будет экстраполирована линейно (по второму и третьему процессору). Помимо такого упрощения для каждого процессора конвейера детально моделируются вычисление только 4 квантов конвейера (3 первых и последний). По разнице характеристик между вторым и третьим квантом определяются затраты, которые будут при обработке каждого последующего кванта от четвертого до предпоследнего. Первый и последний квант могут быть меньше, чем средние, и поэтому рассматриваются отдельно. Также на первом кванте идет в

определенном порядке ожидание освобождения каналов связи, и этот порядок может отличаться от ожидания этих же каналов при обработке второго кванта. На средних процессорах или при обработке на одном процессоре средних квантов работы конвейера разбросы времен ожидания освобождения каналов связи менее значительные, и для упрощенного моделирования они считаются одинаковыми. Таким образом, упрощенное моделирование конвейера идет быстрее, чем полное его моделирование, а погрешность упрощенного моделирования по сравнению с полным представляется совсем небольшой.

2.3 Алгоритм поиска оптимальной решетки

Система DVM и ее языки (C-DVM и Fortran-DVM) поддерживают многомерные распределения данных по процессорам ЭВМ (при этом массивы данных разрезаются по нескольким своим измерениям). При запуске DVM-программы на выполнение пользователь должен указывать конфигурацию процессоров, т.е. размерности процессоров по каждому измерению многомерного куба. Но для многомерных распределений пользователь чаще всего не знает, сколько процессоров ему следует использовать на данной ЭВМ (чтобы задача выполнялась и быстро, и эффективность параллельного выполнения была достаточно высокой), и какую конфигурацию процессоров при этом задать. Чтобы несколько раз не запускать прогнозирование с разными конфигурациями процессоров, разработан специальный и более эффективный режим поиска оптимальной конфигурации.

Алгоритм поиска использует прогнозирование характеристик программы на одной конфигурации процессоров, и сокращает время поиска за счет сокращения количества конфигураций процессоров. Оптимальность конфигурации определяется по времени выполнения программы при условии, что эффективность распараллеливания не ниже некоторой заданной границы. Общее количество процессоров при этом не превышает общего количества процессоров заданного пользователем кластера.

Множество возможных конфигураций представляют собой некоторую область для перебора. Процедура поиска оптимума в ней основана на построении некоторой сетки при помощи функции оценки равномерности распределения массивов по процессорам. Непросто оценить, как влияет неравномерность распределения разных массивов на время выполнения программы, поэтому приоритет отдается равномерности распределения распределенного массива, который имеет максимальное количество элементов. Сетка задается максимальными значениями оценочной функции. Рассмотрев максимальные значения функции (делители размерности или наиболее близкие к делителям величины), алгоритм сокращает область перебора и переходит к рассмотрению менее близких к делителям величин.

Массив программы может распределяться по нескольким измерениям. Для каждого измерения массива имеется его размер dim и размер соответствующего измерения процессорной решетки pdim, на которое распределено это измерение массива. Среди всех процессоров, на которые распределялось это измерение массива (а их pdim), выбираются процессоры, на которые распределено наименьшее и наибольшее число элементов соответствующего измерения массива. Число этих элементов обозначим Emin и Emax соответственно. Если на какой-то процессор из pdim не попало элементов, то Emin будет нулевым.

В качестве функции оценки равномерности распределения измерения массива возьмем отношение Emin/Emax. Заметим, что Emax никогда не нуль, потому что это будет означать, что размерность какого-то измерения массива нулевая, чего быть не может. А отношение Emin/Emax максимально (т.е. равно 1) для делителей измерения массива.

Для многомерной конфигурации процессоров берется минимум оценочной функции по всем измерениям. Отметим, что выбранная функция равна нулю, тогда и только тогда когда хотя бы один из процессоров не включен в вычисления по этому массиву, потому что на нем нет элементов массива. Соответствующие этим случаям конфигурации процессоров заведомо будут считаться не лучше (по времени и эффективности) конфигураций без процессоров, не включенных в работу.

В начале поиска строится область перебора, состоящая из вектора всевозможных конфигураций (в соответствии с топологией процессоров, заданной при распределении массивов в DVM-программе, и ограниченных числом процессоров заданного кластера). Для каждой конфигурации из вектора

находится значение оценочной функции равномерности распределения по выбранному массиву. Отбрасываются конфигурации с нулевым значением функции. Далее выбирается максимальное значение оценочной функции среди всех конфигураций области перебора и строится сетка, узлы которой – это конфигурации процессоров с одним значением оценки.

Поиск производится среди узлов этой сетки по алгоритму деления пополам. На очередном шаге выбирается некоторое количество процессоров, и для него перебираются все конфигурации процессоров с заданным значением оценочной функции. После каждого обработанного числа процессоров выполняется сокращение области перебора, исходя из полученных результатов. После обработки всех значений с текущим значением оценочной функции выбирается новая сетка, узлы которой соответствуют менее равномерно распределенным массивам программы. Так происходит до тех пор, пока не переберутся все варианты из постоянно сокращающейся на каждом шаге области перебора.

При сокращении области используется принцип рассмотрения всех конфигураций процессоров по классам, элементы которых различаются между собой только по одному измерению. Изменение времени выполнения программы среди элементов одного класса более закономерное, чем среди элементов разных классов. В каждом классе ищется один локальный минимум, и лучший среди локальных минимумов всех классов считается искомым минимумом, а соответствующая ему конфигурация процессоров будет считаться оптимальной.

3. Особенности реализации алгоритмов прогнозирования для DVM-программ

Предложенные алгоритмы были реализованы в виде библиотеки прогнозирования характеристик выполнения функций системы поддержки выполнения DVM-программ. Разработанная библиотека прогнозирования используется для прогнозирования характеристик выполнения DVM-программ по файлам трасс [2] и для сравнения вариантов автоматического распараллеливания [3].

В системе DVM есть параметр, при задании которого в ходе выполнения DVM-программы на любой ЭВМ создаются файлы трасс. В файлах представлены последовательности вызовов функций системы поддержки, произошедшие в каждом из выполняющих программу параллельных процессов. Для каждого вызова записывается список значений фактических параметров, время работы функции и время, прошедшее между вызовом этой функции и завершением работы предыдущей.

В некоторых случаях невозможно запустить программу на одном процессоре, например, из-за недостаточного для этого запуска количества оперативной памяти. Как следствие, невозможно собрать трассу и получить последовательность вызовов функций по этому запуску. Но можно запустить программу на большем количестве процессоров, получить файлы трассы, а затем их объединить. Объединенная трасса будет представлять собой трассу непараллельного выполнения этой программы на одном процессоре. Такое объединение возможно и является достаточно простым, потому что имеется единая последовательность вызовов функций поддержки для всех процессоров.

Инструмент прогнозирования характеристик выполнения DVM-программ работает по файлам трасс и при необходимости производит объединение трасс в одну с тем, чтобы потом работать по объединенной трассе. Также на вход этому инструменту поступает файл с характеристиками кластера, для которого производится прогнозирование. В этом файле описывается многоуровневая система коммуникаций, производительность процессоров, для каждого канала передачи задается латентность и временные расходы на передачу каждого байта информации.

Входными данными для сравнения вариантов автоматического распараллеливания являются DVM-директивы вариантов распараллеливания и информация о программе (структура программы, времена выполнения ее частей, параметры циклов, размеры массивов). По этих данным строится последовательность вызовов функций поддержки, по которой будет производиться прогнозирование характеристик параллельного выполнения. По полученным характеристикам производится сравнение вариантов распараллеливания одной программы и выбор лучших вариантов.

Характеристики выполнения программы строятся не только для всей программы, но и для ее частей. Такие части будем называть интервалами выполнения программы. Их можно задавать в тексте программы и для этого они должны удовлетворять требованию одного входа в интервал и одного

выхода из него, а можно генерировать их во время компиляции DVM-программ (тогда интервалами могут стать все параллельные циклы, все непараллельные циклы или вообще все циклы программы)

Характеристиками выполнения интервалов является следующие характеристики [4]: время выполнения, эффективность параллельного выполнения, суммарное время работы всех процессоров (время выполнения, умноженное на количество процессоров), полезное время работы (предполагаемое время выполнения DVM-программы на одном процессоре), потерянное время (суммарное время работы всех процессоров минус полезное время), суммарная рассинхронизация перед вызовами коллективных операции и суммарная рассинхронизация после вызовов, суммарное время совмещений коммуникаций и вычислений. Среди потерянного времени выделяются время коммуникаций, время на дублирование одних и тех же вычислений на разных процессорах (недостаточный параллелизм) и время простоя процессоров. Коммуникационные операции разбиты на классы (операции ввода-вывода, редукционные операции, операции удаленного доступа, операции обмена теневыми гранями). Для каждого типа операции указывается количество таких операции, суммарное время коммуникаций операций этого типа, время совмещений коммуникаций с вычислениями, и рассинхронизация до и после коммуникационных операций.

В результате работы инструмента прогнозирования получаются HTML файлы с прогнозируемыми характеристиками выполнения всех интервалов программы, заданных программистом или автоматически выделенных при компиляции. В HTML файле характеристики представлены в виде цветных таблиц и со средствами навигации по интервалам программы.

4. Эксперименты

Проводились эксперименты по исследованию коммуникационной модели и реализованных алгоритмов.

4.1 Исследование модели коммуникаций

Проводилось сравнение времен коммуникаций на кластере с используемой для прогнозирования моделью коммуникаций. Исследование проводилось на кластере МВС-15К Межведомственного Суперкомпьютерного центра РАН [5]. В качестве исследуемой операции была взята операция пересылки некоторого количества байт от одного процессора другому. Организация и замер времени этой коммуникации был устроен таким образом, чтобы последующие замеры коммуникаций не мешали выполнению предыдущих обменов. В таблицах 1 и 2 приведены времена коммуникаций внутри и между узлов МВС-15К (два процессора в узле), прогноз по модели и погрешность (абсолютная и относительная) модели относительно реальных коммуникаций. Модель коммуникаций внутри узлов имела параметры: латентность $1*10^{-6}$ с, передача байта $1*10^{-9}$ с. Между узлами латентность $7*10^{-6}$ с, передача байта $4*10^{-9}$ с.

Таблица 1. Сравнение модели с реальными коммуникациями внутри узлов МВС-15К (времена в 10^{-6} с).

	0,5 Кбайт	1,5 Кбайт	5 Кбайт	30 Кбайт	40 Кбайт	100 Кбайт
Коммуникации	2	4	5	30	40	93
Прогноз	1,5	2,5	6	31	41	101
Абсолютная погрешность	-0,5	-1,5	+1	+1	+1	+8
Относительная погрешность	-25,0%	-37,5%	+20,0%	+3,3%	+2,5%	+8,8%

Таблица 2. Сравнение модели с реальными коммуникациями между узлами МВС-15К (времена в 10^{-6} с).

	0,5 Кбайт	1,5 Кбайт	5 Кбайт	30 Кбайт	40 Кбайт	100 Кбайт
Коммуникации	7	7,5	20,5	122	208	458
Прогноз	9	13	27	127	167	407
Абсолютная погрешность	+2	+5,5	+6,5	+5	-41	-51
Относительная погрешность	+28,6%	+73,3%	+31,7%	+4,0%	-19,7%	-11,1%

4.2 Исследование алгоритмов прогнозирования

Сравнение спрогнозированных времен выполнения программы с реальными проводилось на примере задачи mhpdv и с использованием машины МВС-100К [6]. mhpdv – это программа трехмерного моделирования сферического взрыва во внешнем магнитном поле с помощью решения уравнений идеальной магнитной гидродинамики [7]. Результаты представлены в абсолютных временах (таблица 3) и относительных временах (таблица 4).

Таблица 3. Прогнозирование времен выполнения разных вариантов распараллеливания задачи mhpdv для МВС-100К (времена в секундах).

	Размерность распределения данных	1 процессор реальное (прогноз)	8 процессоров реальное (прогноз)	64 процессора реальное (прогноз)
Вариант 1	3	3602,66 (3807,52)	484,92 (478,62)	87,13 (61,56)
Вариант 2	2	3580,51 (3807,52)	491,37 (481,79)	99,39 (65,61)
Вариант 3	2	3707,17 (3807,52)	494,34 (481,79)	99,51 (65,80)
Вариант 4	1	3691,74 (3807,52)	509,41 (499,84)	102,97 (76,90)
Вариант 5	2	3697,28 (3807,52)	575,08 (481,79)	121,56 (65,78)
Вариант 6	1	3701,79 (3807,52)	823,33 (499,74)	204,06 (79,17)

Спрогнозированные времена можно рассматривать как времена работы программы на некотором идеальном кластере. Реальные кластеры отличаются и от этого идеального кластера, и друг от друга. При написании своей программы программист исходит из своих представлений о работе программы на кластере, т.е. в некотором роде тоже идеальном кластере. Прогнозирование же позволяет программисту количественно измерить эффективность параллельного выполнения для разных вариантов его программы на идеальном кластере, а также количественно сравнить поведение одного варианта на разных идеальных кластерах.

Таблица 4. Прогнозирование времен выполнения разных вариантов распараллеливания задачи mhpdv для МВС-100К (времена в процентах от наименьшего времени на данном количестве процессоров).

	Размерность распределения данных	1 процессор реальное (прогноз)	8 процессоров реальное (прогноз)	64 процессора реальное (прогноз)
Вариант 1	3	100,6% (100%)	100% (100%)	100% (100%)
Вариант 2	2	100% (100%)	101,3% (100,7%)	114,1% (106,6%)
Вариант 3	2	103,5% (100%)	101,9% (100,7%)	114,2% (106,9%)
Вариант 4	1	103,1% (100%)	105,1% (104,4%)	118,2% (124,9%)
Вариант 5	2	103,3% (100%)	118,6% (100,7%)	139,5% (106,8%)
Вариант 6	1	103,4% (100%)	169,8% (104,4%)	234,2% (128,6%)

Если сравнивать последовательности, в которых упорядочены варианты распараллеливания согласно реальным временам выполнения и спрогнозированным, то ошибка прогнозирования состоит только в пятом варианте. Максимальная ошибка прогнозирования на данной программе была на варианте 5 и 64 процессорах, где прогнозировалось не ухудшение на 21,3% относительно варианта 4, а улучшение на 18,1%, что в сумме можно считать погрешностью в 39,4%.

4.3 Исследование алгоритма поиска оптимальной конфигурации процессоров

На примере задачи JAC (решение системы линейных уравнений итеративным методом Якоби, размер массива 10000x10000, 10 итераций) посмотрим, насколько сокращается количество вариантов при поиске оптимальной конфигурации процессоров (таблица 5). Поиск построен на применении эвристик. Первая сокращает область перебора еще до рассмотрения каких-либо конфигураций, вторая эвристика сокращает область перебора во время поиска путем отсечения вариантов внутри некоторых классов конфигураций (внутри класса все конфигурации процессоров отличаются значениями только

по одному измерению).

Таблица 5. Различные характеристики поиска оптимальной конфигурации процессоров для задачи JAC.

Размерность распределения данных	Количество конфигураций при разном количестве процессоров (8, 64, 256)	После применения первой эвристики	После применения второй эвристики	Выигрыш от применения обеих эвристик (в процентах)
1	8, 64, 256	8, 64, 160	6, 13, 16	25, 80, 94
2	20, 280, 1466	20, 280, 1260	15, 74, 123	25, 74, 92

Для одномерного распределения оптимальное количество процессоров составляло по прогнозированию 110 процессоров, и результаты эвристического поиска и полного перебора совпали. Для двумерного распределения оптимальная конфигурация прогнозировалась как 7x16 (112 процессоров), а полный перебор дал 7x18 (126 процессоров). Эта конфигурация (7x18) была отсечена из рассмотрения второй эвристикой (на основе классификации).

5. Заключение

При отладке эффективности программ очень важно иметь представления о том, как программа будет выполняться на кластере. Измерять эффективность программы по результатам реальных запусков на ЭВМ является дорогим и затратным по времени решением (из-за нестабильности запусков). К тому же вполне может получиться, что на разных кластерах самыми эффективными окажутся разные варианты. Как правило, программист не так часто измеряет эффективность программ, а чаще исходит из своих оценок: какие циклы стоит распараллелить, выгодно ли использовать конвейер и т.п. Инструменты прогнозирования позволяют программисту количественно оценить его решения на примере некоторого идеального кластера.

При автоматическом распараллеливании программ возникает огромное количество вариантов распараллеливания, многие из которых вообще не ускоряют выполнение программы – поэтому необходима количественная оценка характеристик выполнения каждого варианта. В этом случае трудно обойтись без методов прогнозирования.

Важной особенностью, повлиявшей на возможность прогнозирования для DVM-программ, является подход к организации системы поддержки выполнения DVM-программ, который позволяет по трассам выполнения программ на одном количестве процессоров определять характеристики выполнения программы на другом количестве процессоров. Для любого количества процессоров и параметров многомерной процессорной решетки последовательность функций системы поддержки одна и та же, что упрощает процесс прогнозирования.

Наиболее важным направлением дальнейшего развития является поиск путей повышения точности результатов прогнозирования, особенно в области прогнозирования коммуникаций. Вторым направлением является сокращение времени работы алгоритмов прогнозирования при незначительном снижении точности прогнозирования.

Предметом для исследования является также автоматическое определение параметров коммуникационной среды. В системе DVM такие автоматические средства отсутствуют, а задание параметров для каждого кластера требует отдельного трудоемкого исследования.

Литература

1. DVM система: [www.keldysh.ru/dvm].
2. Клинов М.С., Крюков В.А. Прогнозирование характеристик параллельного выполнения DVM-программ. // Труды Всероссийской научной конференции “Научный сервис в сети Интернет: технологии параллельного программирования”, сентябрь 2006, г. Новороссийск. –М.: Изд-во МГУ, 2006. –С. 113-114.

3. Клинов М.С., Крюков В.А. Система автоматизированного распараллеливания программ на языке Фортран. DVM-эксперт. // Труды Всероссийской научной конференции “Научный сервис в сети Интернет: многоядерный компьютерный мир”, сентябрь 2007, г. Новороссийск. –М.: Изд-во МГУ, 2007, –С. 95-97.
4. Денисов В.Е., Ильяков В.Н., Ковалева Н.В., Крюков В.А. Отладка эффективности DVM-программ. –М., 1998 (Препр. ИПМ РАН; N 74).
5. Учреждение Российской академии наук Межведомственный суперкомпьютерный центр РАН: [http://www.jscc.ru].
6. Суперкомпьютер "МВС-100К": [http://www.jscc.ru/hard/mvs100k.shtml], 24.09.2008.
7. Устюгов С.Д., Чечеткин В.М. Взрыв сверхновой при крупномасштабной конвективной неустойчивости вращающейся протонейтронной звезды. // Астрономический журнал, 1999, –Т. 76, №11, –С. 816-824.