УДК 004.051

ПРОГНОЗИРОВАНИЕ ХАРАКТЕРИСТИК ЭФФЕКТИВНОСТИ ВЫПОЛНЕНИЯ DVM-ПРОГРАММ НА КЛАСТЕРЕ*

М.С. Клинов

Институт прикладной математики им. М.В. Келдыша РАН

klinov@keldysh.ru

Поступила в редакцию 31.03.2009

Рассматриваются алгоритмы моделирования параллельного выполнения программы на кластере с заданными для него характеристиками, и описывается алгоритм поиска оптимальной конфигурации процессоров, на которой спрогнозированное время работы программы будет минимальным. При моделировании программа представляется в виде последовательности вычислительных блоков и вызовов функций системы поддержки выполнения DVM-программ.

Ключевые слова: высокопроизводительные вычисления, кластер, анализ эффективности, прогнозирование, моделирование выполнения.

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 в целом, 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 может осуществляться как по результатам реальных запусков программ на целевой ЭВМ или на 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: для сбора необходимой информации и запусков 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 их 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, предполагая, что эффективность выполнения этого цикла будет оставаться неизменной при росте

* Статья рекомендована к печати программным комитетом Международной научной конференции «Параллельные вычислительные технологии 2009» (http://agora.guru.ru/pavt).

числа итераций. Такой метод применим в обоих подходах к анализу эффективности.

Подход с прогнозированием можно использовать еще до получения текста программ, что позволяет оперировать 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 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 позволяет более эффективно решать многие задачи. Во-первых, появляется возможность задействовать 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], созданная 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 и 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. Для их отладки в составе системы имеются иументы: динамический контроль корректности и сравнительная отладка.

Целью данного исследования была разработка алгоритмов прогнозирования характеристик выполнения DVM-программ на кластерах, а также алгоритмов автоматизированного поиска оптимальной конфигурации процессоров. Характеристики выполнения представляют собой целый набор характеристик [2], который используется в DVM-системе для анализа эффективности.

1. Алгоритмы прогнозирования характеристик выполнения

Алгоритмы прогнозирования должны быть применимы как для отладки эффективности DVM-программ, так и для оценки разных вариантов распараллеливания исходной программы в DVM-программу. Само прогнозирование должно осуществляться путем моделирования параллельного выполнения программы. При этом известны параметры коммуникационной среды целевого кластера и производительность его процессоров.

1.1. Особенности предлагаемого подхода. Для решения задачи прогнозирования предлагается рассматривать 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 и характеризуются только временем их выполнения. Последовательность вызовов и времена выполнения участков программы между вызовами технически можно представить в виде единой последовательности вызовов, но в которой каждый вызов функции поддержки будет характеризоваться еще и временем выполнения участка программы перед ним. Непосредственно перед выходом из программы всегда есть вызов функции поддержки, поэтому описанное действие правомерно.

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. Это упрощает сбор необходимой информации для прогнозирования.

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 программы на кластере должно опираться на модель оценки времен вычислений частей программы и на модель оценки времен коммуникаций.

Для оценки времен вычислений предлагается использовать одну из двух моделей. Первая модель основывается на замерах времен на инструментальной ЭВМ и пересчете их для целевой ЭВМ, при этом считается, 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 представлены в трассах выполнения программы на некотором количестве процессоров. В них собраны 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 неограничено. 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.

1.2. Алгоритмы моделирования. На вход алгоритмам прогнозирования поступает последовательность вызовов функций системы поддержки выполнения DVM-программ. Эта последовательность характеризует не только вызовы, но и участки программы между ними. Функции поддержки устроены так, что по имени функции всегда можно определить, как распределены вычисления в предшествующем вызову участке программы и какие возникают при этом коммуникации.

Моделирование выполнения непараллельных участков заключается в добавлении времени работы этого участка к временам работы всех процессоров, которые выполняют данный участок. Наличие непараллельных участков в программе приводит к увеличению значения характеристики «недостаточный параллелизм», поскольку одни и те же вычисления выполняются на многих процессорах.

В алгоритме моделирования выполнения параллельных циклов определяется количество итераций цикла, выполняемое каждым процессором, и осуществляется соответствующее разделение общего времени работы цикла между процессорами. В случаях дублирования вычислений увеличивается значение характеристики недостаточного параллелизма.

В DVM-программе могут использоваться 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 этих операций 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 ожидания завершения обновления теневых граней будут вычислены для каждого процессора время, затраченное собственно на ожидание, и время совмещения коммуникаций с предшествующими вычислениями.

Для редукционных операций определяется количество байт, которые занимает редукционная переменная. Моделируемый сбор значений от всех процессоров, участвующих в операции редукции, осуществляется через главный процессор (с минимальным номером из участвующих в операции процессоров). Вычисление редукционного значения по собранным значениям занимает незначительное время, которое при моделировании считается нулевым. После этого моделируется запуск рассылки 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 при моделировании определяются по той же схеме, что используется в системе поддержки выполнения DVM-программ. Характеристики выполнения таких циклов получаются путем упрощенного моделирования работы конвейера. Для этого 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 моделируется 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 кванты могут быть меньше, чем средние, и поэтому рассматриваются отдельно. Также на первом кванте время ожидания освобождения каналов связи часто сильно отличается от соответствующих времен при обработке последующих квантов. 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 такого упрощения представляется совсем небольшой.

2. Алгоритм поиска оптимальной конфигурации процессоров

DVM-система 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. Для многомерных распределений пользователю сложнее определять оптимальную конфигурацию процессоров для каждой ЭВМ (чтобы задача выполнялась и быстро, и достаточно эффективно). Чтобы избежать многочисленных запусков прогнозирования с разными конфигурациями процессоров, разработан специальный и более эффективный режим поиска оптимальной конфигурации.

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

Процедура поиска оптимума среди возможных конфигураций основана на построении некоторой сетки при помощи функции оценки равномерности распределения массивов программы по процессорам. Сначала перебираются конфигурации процессоров, на которые более равномерно распределяются массивы программы, затем конфигурации с менее равномерно распределенными массивами. Функция оценки

равномерности распределения массива равна минимуму среди функций оценки для каждого измерения массива. Каждая такая функция задается отношением минимального количества элементов измерения, которые распределены на некоторый процессор, к максимальному количеству элементов среди процессоров, на которые распределялось данное измерение массива. Значение функции может быть нулевым (например, когда процессоров больше, чем элементов массива) – на этом основана первая эвристика поиска. Соответствующие этим случаям конфигурации процессоров заведомо будут считаться не лучше (по времени и эффективности) конфигураций, которые не имеют процессоров, не включенных в вычисления по массиву.

Для сокращения области перебора 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 (ненужные конфигурации отбрасываются – вторая эвристика поиска), лучший среди локальных минимумов всех классов считается искомым минимумом, а соответствующая ему конфигурация процессоров будет считаться оптимальной.

3. Эксперименты

Предложенные алгоритмы были реализованы в виде библиотеки прогнозирования характеристик выполнения функций системы поддержки выполнения DVM-программ. Разработанная библиотека используется для прогнозирования характеристик выполнения DVM-программ по файлам трасс [3] и для сравнения вариантов автоматического распараллеливания [4].

Проводились эксперименты по исследованию коммуникационной модели, используемой при прогнозировании, и самих алгоритмов прогнозирования.

3.1. Исследование модели коммуникаций. Сравнение времен коммуникаций на кластере с используемой для прогнозирования моделью коммуникаций проводилось на кластере МВС-15К Межведомственного суперкомпьютерного центра РАН [5]. В качестве исследуемой была взята операция пересылки некоторого количества байт от одного процессора другому. Замер времени этой коммуникации был устроен таким образом, чтобы предыдущие и последующие коммуникации ему не мешали. В таблицах 1 и 2 приведены времена коммуникаций внутри узлов и между узлами 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 – $1 \cdot 10^{-6}$ с, передача байта – $1 \cdot 10^{-9}$ с. Между узлами латентность – $7 \cdot 10^{-6}$ с, передача байта – $4 \cdot 10^{-9}$ с.

3.2. Исследование алгоритмов прогнозирования. Исследование алгоритмов проводилось путем сравнения спрогнозированных времен выполнения программы с реальными на примере задачи MHPDV и с использованием машины МВС-100К [6]. MHPDV – это программа трех-

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%

мерного моделирования сферического взрыва во внешнем магнитном поле с помощью решения уравнений идеальной магнитной гидродинамики [7]. Результаты представлены в абсолютных (табл. 3) и в относительных (табл. 4) временах.

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 и поведение одного варианта на разных идеальных кластерах.

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%.

3.3. Исследование алгоритма поиска оптимальной конфигурации процессоров. На примере задачи JAC (решение системы линейных уравнений итеративным методом Якоби, размер массива 10000×10000, 10 итераций) посмотрим, насколько сокращается количество конфигураций при поиске оптимальной конфигурации процессоров (табл. 5). Поиск построен на применении эвристик. Первая сокращает область перебора еще до рассмотрения каких-либо конфигураций, вторая эвристика сокращает область

Таблица 3

Прогнозирование времен выполнения разных вариантов распараллеливания задачи MHPDV для МВС-100К (времена в секундах)

Вариант	Размерность распределения данных	1 процессор реальное (прогноз)	8 процессоров реальное (прогноз)	64 процессора реальное (прогноз)
1	3	3602.66 (3807.52)	484.92 (478.62)	87.13 (61.56)
2	2	3580.51 (3807.52)	491.37 (481.79)	99.39 (65.61)
3	2	3707.17 (3807.52)	494.34 (481.79)	99.51 (65.80)
4	1	3691.74 (3807.52)	509.41 (499.84)	102.97 (76.90)
5	2	3697.28 (3807.52)	575.08 (481.79)	121.56 (65.78)
6	1	3701.79 (3807.52)	823.33 (499.74)	204.06 (79.17)

Таблица 4

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

Вариант	Размерность распределения данных	1 процессор реальное (прогноз)	8 процессоров реальное (прогноз)	64 процессора реальное (прогноз)
1	3	100.6 (100)	100 (100)	100 (100)
2	2	100 (100)	101.3 (100.7)	114.1 (106.6)
3	2	103.5 (100)	101.9 (100.7)	114.2 (106.9)
4	1	103.1 (100)	105.1 (104.4)	118.2 (124.9)
5	2	103.3 (100)	118.6 (100.7)	139.5 (106.8)
6	1	103.4 (100)	169.8 (104.4)	234.2 (128.6)

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

перебора во время поиска путем отсечения вариантов внутри некоторых классов конфигураций (внутри класса все конфигурации процессоров отличаются только по одному измерению процессорной решетки).

Для одномерного распределения оптимальное количество процессоров составляло по прогнозированию 110 процессоров; результаты эвристического поиска и полного перебора совпали. Для двумерного распределения оптимальная конфигурация прогнозировалась как 7×16 (112 процессоров), а полный перебор дал 7×18 (126 процессоров). Эта конфигурация (7×18) была отсечена из рассмотрения второй эвристикой (на основе классификации).

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 оценивать его решения на примере некоторого идеального кластера.

При автоматическом распараллеливании программ возникает огромное количество вариантов распараллеливания, многие из которых вообще не ускоряют выполнения программы – поэтому необходима также количественная оценка 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 их количестве. 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 исследования является также автоматическое определение параметров коммуникационной среды. В системе DVM такие автоматические средства отсутствуют, а задание параметров для каждого кластера является трудоемким.

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

Список литературы

1. DVM система: сайт. – URL: http://www.keldysh.ru/dvm (дата обращения 26.01.2009).

2. Денисов В.Е., Ильяков В.Н., Ковалева Н.В., Крюков В.А. Отладка эффективности DVM-программ: Препр. ИПМ РАН. 1998. № 74.

3. Клинов М.С., Крюков В.А. Прогнозирование характеристик параллельного выполнения DVM-программ // Труды Всерос. науч. конф. «Научный сервис в сети Интернет: технологии параллельного программирования», Новороссийск, 18–23 сентября 2006. М.: Изд-во МГУ, 2006. С. 113–114.

4. 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», Новороссийск, 24–29 сентября 2007. М.: Изд-во МГУ, 2007. С. 95–97.

5. Учреждение Российской академии наук Межведомственный суперкомпьютерный центр РАН: сайт. – URL: http://www.jscc.ru (дата обращения 17.02.2009).

6. Суперкомпьютер «МВС-100К»: сайт. – URL: http://www.jscc.ru/hard/mvs100k.shtml (дата обращения 26.01.2009).

7. Ustyugov S.D. Three Dimensional Numerical Simulation of MHD Solar Convection on Multiproccesors Supercomputer Systems // Proceedings of the NSO 23 Workshop «Solar MHD: Theory and Observations – a High Spatial Resolution Perspective», Sunspot, New Mexico, USA, 18–22 July, 2005.

PERFORMANCE PREDICTION OF DVM PROGRAM EXECUTION ON A CLUSTER

M.S. Klinov

The prediction is based on modeling the parallel program execution on a cluster. Besides cluster's characteristics, input data for prediction contain a sequence of calls to the DVM program runtime support system. This allows one to predict the performance characteristics both along the execution traces of the DVM program on the instrumental computer and during the automatic parallelization of the sequential program before the parallel program generation. In addition, a search engine for the optimal configuration of processors, in which the predicted DVM program execution time is minimal, is described.

Keywords: high-performance computing, cluster, performance analysis, performance prediction, execution modeling.