МЕТОД И СРЕДСТВА ПРОГНОЗИРОВАНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ НА ОСНОВЕ КЛАССИФИКАЦИИ ОПОРНЫХ ВЕКТОРОВ

А.И. Бакшеев, В.Ю. Воронов

Введение. Для современных многопроцессорных вычислительных систем характерно растущее число вычислительных узлов и ядер. Проблема разработки производительного и масштабируемого программного обеспечения для таких платформ становится все более актуальной. Один из важных этапов в процессе настройки производительности программы -- определение значений управляющих параметров алгоритмов в программе, выбор алгоритмов из некоторого множества доступных вариантов для учета специфики входных данных. На данном этапе «тонкая» настройка программы, выполняющейся на десятках тысяч вычислительных ядер, становится неприемлемой. Таким образом, необходимы средства автоматизации подобной настройки производительности программы.

В данной работе предложен метод автоматического выбора алгоритма и определения его параметров для заданных входных данных параллельной программы. В качестве класса параллельных программ, для которых предлагается метод, рассматриваются программы параллельного решения систем линейных уравнений с разреженной матрицей коэффициентов (разреженных СЛАУ). Данный алгоритм является вычислительным ядром во многих методах, поэтому от его производительности напрямую зависит производительность всей программы. Сложность проблемы заключается в том, что на время решения существенно влияет как выбор алгоритма решения, от свойств рассматриваемой СЛАУ, характеристик вычислительной системы. Кроме того, в ходе определения наилучшего алгоритма для рассматриваемой задачи необходимо исследовать влияние значений управляющих параметров, которые в общем случае могут быть различного типа. Подобная задача может быть сформулирована в терминах минимизации целевой функции по набору параметров смешанного типа, которая исследовалась в[1]. Предлагаемый в данной работе метод основан на сборе и анализе статистики решения некоторого набора СЛАУ на вычислительной системе и последующего формирования модели производительности параллельной программы решения СЛАУ. Предлагаемая модель основана на бинарной классификации и позволяет соотносить параметры алгоритма к одному из двух классов в соответствии с тем, насколько данный алгоритм хуже по времени работы чем самый лучший из рассматриваемых. Таким образом, рассматривается более слабая формулировка задачи, требующая меньших вычислительных затрат чем решение задачи смешанной целочисленной оптимизации. В качестве механизма классификации предложено использовать классификаторы опорных векторов (support vector classificator)[2].

Структура данной работы следующая: первый раздел посвящен анализу существующих разработок в области автоматической настройки производительности программ. Во втором разделе приведена формулировка задачи. В третьем разделе сформулирован предлагаемый метод. Вопросы программной реализации метода рассматриваются в четвертом разделе. Пятый раздел содержит результаты экспериментов с методом на параллельной платформе и их анализ. В заключении приводятся основные результаты работы и указаны пути дальнейшего развития.

1. Обзор существующих работ. Методы автоматической настройки производительности получили достаточно широкое распространение в разных видах программного обеспечения. Если ограничиться существующими работами в области анализа производительности параллельных программ и построения моделей прогнозирования их производительности можно отнести к двум подходам: построение модели функционирования программы на основе анализа исходного текста, трасс и другой статистики выполнения программы. Так, в[3] обосновывается модель выполнения параллельной программы на основе сетей Петри. Иллюстрацией работ второго типа является[4], в которой предложен алгоритм выбора значений параметров алгоритма решения СЛАУ на основе решения задачи классификации. Работа[5] представляет алгоритм прогнозирования характеристик масштабируемости параллельной программы с учетом особенностей обрабатываемых данных и характеристик вычислительной системы. Предлагаемый в работе алгоритм основан на построении модели производительности программы на основе регрессии.

Подход первого типа, ввиду сложности параллельных программ для решения прикладных задач, примененим к основном к программам модельного типа и часто применяется для верификации поведения программ на перпективных вычислительных системах. Дополнительную сложность составляет то, что исходные тексты не всех программ могут быть доступны пользователю, например в случае использования компонент-библиотек. К трудностям второго подхода относится необходимость сбора статистики производительности параллельной программы с различными комбинациями параметров алгоритма, что требует проведения серии вычислительных экспериментов. Поэтому данный подход актуально применять прежде всего для предметно-ориентированных прикладных параллельных программ, использование которых само по себе связано с выполнением множества серий вычислительных экспериментов с различными параметрами и для различных входных данных.

Данная работа относится ко второму подходу. Предлагаемый в работе метод выбора управляющих параметров алгоритма основан на формировании модели производительности параллельной программы и последующем ее использовании для выбора значений управляющих параметров алгоритма.

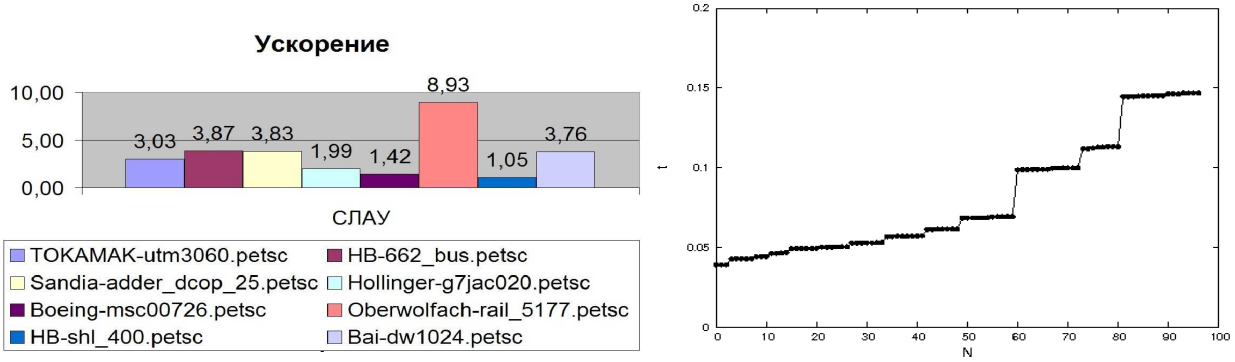
2. Модель производительности параллельной программы решения СЛАУ. Далее мы предполагаем, что рассматривается параллельная программа, в которой решается разреженная СЛАУ

$$Ax = b.$$

Критерием производительности программы будем считать время выполнения решения программой на заданной вычислительной системе для фиксированного числа процессоров. На время решения системы уравнений влияет вид матрицы СЛАУ, выбранное число процессоров и выбранный алгоритм решения. Подтверждением могут служить представленные на рис. 1(a) результаты вычислительного эксперимента по решению набора СЛАУ. В таблице указано отношение наибольшего времени решения данной СЛАУ к наименьшему для различных рассматриваемых алгоритмов. Это отношение далее будем называть для краткости «разбросом». Как видно, для различных СЛАУ значения разброса может существенно различаться. Чем больше значение разброса, тем большее влияние на производительность оказывает выбор значений параметров алгоритма. Введем следующие обозначения.

Пусть $F(A) = (f_1, f_2, \ldots, f_k)^T$ -- вектор признаков, которые вычисляются из матрицы A. Каждая компонента f_i -- скалярное значение, может быть вещественным или целочисленным признаком. Пусть также рассматривается некоторое множество алгоритмов решения разреженных СЛАУ $S \in \mathbf{S}$, каждый из которых характеризуется набором управляющих параметров $S = (s_1, s_2, \ldots, s_l)^T$. Тогда моделью производительности будет функция $f : (F(A), S) \to P_{S,A}$, где $P_{S,A}$ - производительность выполнения алгоритма S для СЛАУ A. Производительность алгоритма в данном случае будем определять как: $P_{S,A} = 1/t$, где t - время выполнения алгоритма S для решения СЛАУ A на заданной вычислительной системе. Таким образом, для определения наилучшего алгоритма необходимо построить такую функцию и определить ее минимум по параметру S.

Построить функцию f в аналитической форме нереалистично ввиду большого числа факторов, влияющих на производительность программы. Кроме того, дальнейшая минимизация функции для определения наилучшего алгоритма решения требует вычислительно сложного алгоритма. Вместо этого задача выбора алгоритма и его управляющих параметров рассматривается как задача определения некоторого подмножества «хороших» параметров алгоритма и отделения их от оставшихся «плохих». Для определения критерия выбора алгоритма рассмотрим рис. 1(b), на котором представлено время решения некоторой СЛАУ в зависимости от выбранных значений параметров алгоритмов.



(a) (b)

Рис. 1. Разброс времени решения для различных СЛАУ с помощью рассматриваемых алгоритмов(a), нормализованное время решения СЛАУ в зависимости от значений параметров алгоритма решения (b).

На представленном графике видно, что имеются скачки производительности, отделяющие одно множество параметров алгоритмов от другого. Это может быть использовано для разделения на классы. С другой стороны, в случае когда разделение классов не может быть произведено настолько же ясно, предлагается для заданного порогового значения $\alpha \in [0,1]$ относить к классу «хороших» алгоритмов те, которые отличаются от самого производительного не более чем в α раз, а к «плохому» -- остальные.

Отметим, что предложенная модель может быть обобщен на достаточно широкое семейство параллельных программ, отличных от решения СЛАУ. Действительно, для его применения необходимо сформировать множество алгоритмов и их параметров S, признаковое пространство входных данных программы F, соответствующие рассматриваемой задаче. После этого становится возможно сформулировать

идентичную рассмотренной модель производительности программы и применять метод выбора параметров алгоритма программы, указанный далее.

3. Метод выбора алгоритма решения СЛАУ и значений его параметров. Предложенный метод состоит из двух стадий, соответствующих построению классификатора на обучающем наборе задач и применению построенного классификатора для входного набора задач:

1 Этап. Обучение.

- Выбор обучающего множества СЛАУ Tr.
- Создание различных комбинаций параметров алгоритмов S_{tr}.
- Измерение производительности P выполнения алгоритмов S_{tr} на входных данных из множества Tr. Составление множества результатов $(P_{s,A}, s, F(A))$, где $P_{S,A}$ - производительность выполнения алгоритма $s \in S_{tr}$ для СЛАУ $A \in Tr$.
- Вычисление признаков $F(A)$ для всех СЛАУ $A \in Tr$.
- Разбиение множества $(P_{s,A}, s, F(A))$ на классы $Y = -1,1$ по следующему принципу: для заданного порогового значения $\alpha \in [0,1]$ относим к классу 1 («хороших») те наборы $(P_{s,A}, s, F(A))$, которые для фиксированного СЛАУ A отличаются от самого производительного не более чем в α раз, а к -1 («плохому») -- остальные.
- Построение классификатора f_α на основе обучающей выборки $\{s, F(A); y\}$, $y \in Y$

2 Этап. Применение.

- Вычисление признаков $F(\hat{A})$ для заданной СЛАУ $\hat{A}$.
- Классификация наборов $(s, F(\hat{A}))$, $s \in S_{tr}$, с помощью классификатора f_α на классы Y. Выбор произвольного набора $(\hat{s}, F(\hat{A}))$, $\hat{s} \in S_{tr}$, заданного класса $y \in Y$.
- Использование алгоритма $\hat{s}$ для решения заданной СЛАУ $\hat{A}$.

4. Реализация предложенного метода. Для анализа предложенного метода и его применения в решении практических задач на параллельных вычислительных системах была разработана программная реализация. Данный инструмент состоит из следующих элементов: (1)*MLSolver* -- основной компонент, управляющий работой других модулей; (2) *Features extractor* – модуль, вычисляющий набор признаков для заданной СЛАУ; (3) *Commands runner* – модуль, осуществляющий постановку заданий на выполнение на вычислительной системе Blue Gene/P и анализирующий ход выполнения задач; (4) *Commands creator* – модуль, создающий случайное множество наборов управляющих параметров для построения обучающей выборки; (5) *Results analyzer* – модуль, анализирующий результаты выполнения заданий; (6) *Classifier creator* – модуль, формирующий классификатор на основе данных, предоставляемых модулями *Features extractor*, *Results analyzer; (7) Command selector* – модуль, классифицирующий множество наборов управляющих параметров.

Схема компонентов разработанной инструменатльной системы представлена на рис. 2.

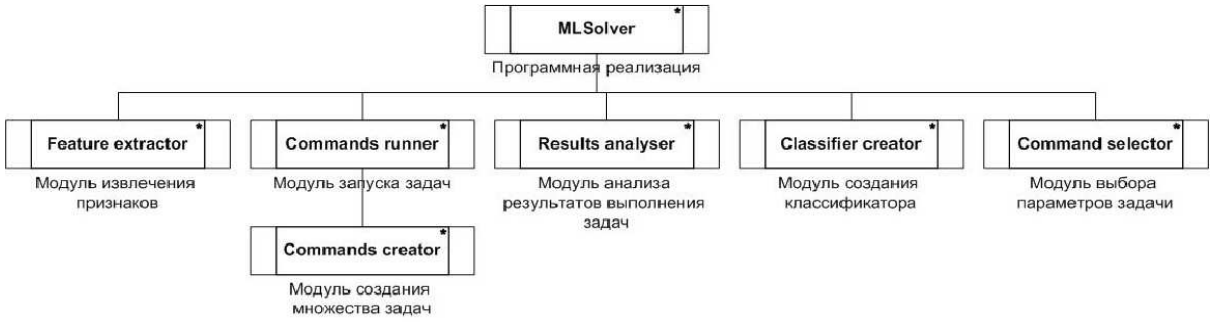


Рис. 2. Компоненты разработанной программной реализации.

Предложенная реализация была выполнена для поддержки вычислительного эксперимента на платформе Blue Gene/P. На этапе генерации обучающей выборки в модуле *Commands creator* выполняется случайная генерация множества наборов управляющих параметров $s \in S$, для которых выполняется вычислительный эксперимент. В качестве источника методов решения СЛАУ в программной реализации поддерживаются реализации итерационного решения с предобусловливанием из пакета PETSc[6]. В работе представлены результаты экспериментов, в которых множество алгоритмов состоит из одного типа итерационного метода решения GMRES с блочным предобусловливателем типа Якоби. Рассматриваемые типы и значения управляющих параметров, для которых поддерживается метод, представлены в таблице 1.

Таблица 1. Управляющие параметры алгоритма решения СЛАУ, их тип и допустимые значения.

Название параметра	Тип значений параметра	Диапазон значений параметра
Тип итерационного метода	перечислимый	GMRES
Параметр перезапуска для метода GMRES	целочисленный	[5,100]
Использовать преаллокацию памяти для полного построения подпространства Крылова	бинарный	{0, 1}
Использовать модифицированный метод Грамма-Шмидта	бинарный	{0, 1}
Использовать алгоритм уточнения для ортогонализации Грамма-Шмидта	перечислимый	{Нет, при необходимости, всегда}
Тип переобусловливателя	перечислимый	Блочный метод Якоби
Число блоков предобусловливателя Якоби	целочисленный	[4, 16]
Тип разложения блока предобусловливателя	перечислимый	ILU-разложение
Значение для заполнения элементов блока при ILU-разложении	вещественный	[0.1, 5.0]
Тип алгоритма переупорядочивания строк и столбцов в блоках	перечислимый	Метод вложенных делений, обратный метод Катхилла-Макки

Отметим, что ряд параметров имеет вещественный тип. Для них выбраны ограничения диапазона значений.

Для извлечения признаков из СЛАУ в модуле *Features extractor* используется пакет AnaMod[7], который вычисляет набор вещественных характеристик матрицы СЛАУ. В данной программной реализации используются 52 признака, включая значения норм, распределение ненулевых элементов, спектральные характеристики и др. Для построения классификаторов в модуле *Classifier creator* используются реализации методов опорных векторов из пакета LibSVM[8]. На рис. 3 представлены последовательности использования модулей для каждой части алгоритма в программной реализации предложенного метода.

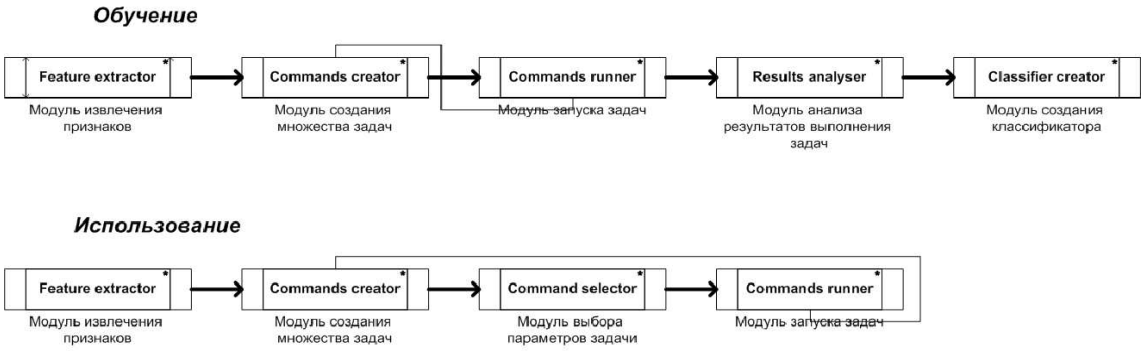


Рис. 3. Сценарии использования программной реализации при обучении и применении метода.

На этапе «обучения» выбирается произвольный набор СЛАУ. Далее для каждой СЛАУ из данного набора с помощью модуля *Features extractor* извлекаются признаки. На следующем шаге *Commands creator* генерирует случайное множество наборов управляющих параметров итерационного метода для дальнейшего решения СЛАУ алгоритмом из пакета PETSc. Далее модуль *Commands runner* выполняет решение со всеми наборами управляющих параметров для каждого СЛАУ из полученного случайного множества на Blue Gene/P. На следующем шаге *Results analyzer* производит анализ результатов выполнения метода, в частности, получает параметр, характеризующий производительность метода. На основе данных, полученных модулями *Features extractor* и *Results analyzer* строится классификатор с помощью *Classifier creator.*

На этапе «использования» для необходимого СЛАУ *Features extractor* извлекает признаки. Далее с помощью классификатора, полученного на этапе «обучения», и признаков модуль *Command selector* выбирает случайный набор управляющих параметров из множества, для которого производился вычислительный эксперимент и элементы которого классифицированы как «хороших». В заключении *Commands runner* выполняет итерационный метод с данными параметрами на платформе Blue Gene/P. Отметим, что предложенная программная реализация поддерживает расширение множества рассматриваемых алгоритмов.

5. Результаты применения метода на платформе Blue Gene/P. Представим результаты экспериментального анализа предложенного метода. Для оценки точности метода рассматривается выборка

СЛАУ из репозитория университета Флориды[9]. Данные СЛАУ соответствуют ряду практических задач, возникающих в различных физических процессах и моделях структурной механики, моделирования электрических сетей, задач оптимизации, процессов электромагнитного распространения и др.

Кол-во СЛАУ для формирования статистики и обучающей выборки составляет 7 элементов, размер тестового множества составляет 1 элемент. В таблице 3 указаны некоторые характеристики рассматриваемого набора СЛАУ и пределы их разброса. Отметим, что небольшой размер рассматриваемых выборок связан с имеющимися ограничениями на машинное время для проведения экспериментов.

Таблица 2. Основные характеристики рассматриваемого набора СЛАУ.

Тип характеристики	Минимальное значение	Максимальное значение
Количество строк	662	3060
Число ненулевых элементов	4712	42211
Оценка числа обусловленности	5.829713e+00	8.115051e+06
Модуль максимального собственного значения матрицы СЛАУ	4.992242e-05	4.219587e+09
Модуль минимального собственного значения матрицы СЛАУ	1.188631e-09	5.816462e+04

Эксперимент на платформе Blue Gene/P производится на 128 узлах в режиме запуска задач VN, что соответствует запуску программ для 512 ядер. В результате формируется обучающая выборка, содержащая 602 элементов, и тестовая выборка, содержащая 196 элементов. Каждый элемент выборок содержит 59 скалярных величин. На основе собранной обучающей выборки строится классификатор. В таблице 4 представлены результаты проведенного эксперимента.

Таблица 4. Характеристики точности и ошибки классификации в предложенном методе на тестовой выборки СЛАУ .

Порог α	Точность классификации	TP	FP	TN	FN	Вероятность выбора класса «хороших»
0.1	0.891304	0	10	82	0	0
0.2	0.771739	6	14	65	7	0,46154
0.3	0.782609	15	15	57	5	0,75
0.4	0.771739	25	14	56	7	0,78125
0.5	0.51087	25	24	22	21	0,54348

Обозначения: Точность классификации TP - число элементов правильно классифицируемых из класса «хороших», FP - число элементов неправильно классифицируемых из класса «хороших», TN - число элементов правильно классифицируемых из класса «плохие», FN - число элементов неправильно классифицируемых из класса «плохие».

Как видно из представленных результатов, при использовании предложенного метода и значений порогов $\alpha = 0.3,0.4$ с достаточно большой точностью выбираются управляющие параметры итерационного метода, на которых достигаются высокие значения производительности. В таблице 5 представлены результаты сравнения среднего времени решения СЛАУ для алгоритма, выбранного предлагаемым в работе методом, по сравнению с алгоритмом со средней и минимальной производительностью, определенных в результате вычислительного эксперимента.

Таблица 4. Характеристики точности и ошибки классификации в предложенном методе на тестовой выборки СЛАУ .

Порог α	t	$t_{average}$	t_{max}	$t_{average}/t$	t_{max}/t	Время обучения классификатора, сек.
0.2	0.05019	0.08024	0.14683	1.59872	2.92548	65.120
0.3	0.04649	0.08024	0.14683	1.72596	3.15831	70.864
0.4	0.06127	0.08024	0.14683	1.30961	2.39644	74.009
0.5	0.11225	0.08024	0.14683	0.71483	1.30806	78.077

Таблица 5. Сравнение производительности предложенного метода.

Здесь t -- время выполнения алгоритма, найденного данным методом для СЛАУ тестовой выборки. $t_{average}$ - среднее время выполнения алгоритмов для СЛАУ тестовой выборки. t_{max} - максимальное время выполнения алгоритмов для СЛАУ из тестовой выборки. Как видно из данных результатов, предложенный

метод выбора управляющих параметров позволяет увеличить производительность в среднем в 1.5 раза по сравнению со средними значениями и в 2.8 раз -- по сравнению с минимальными.

Заключение. Предложен метод автоматического выбора параметров алгоритма решения СЛАУ, для которого достигаются высокие значения точности прогнозирования. В дальнейшей работе по данной теме планируется: (1) в первую очередь существенное расширение масштабов вычислительного эксперимента для получения более детальных результатов (2) повышение точности классификации за счет более сложного алгоритма построения классификации; (2) снижение признакового пространства без потери качества обучения; (3) исследование точности метода при использовании мультиклассовой классификации.

Работа выполнена в рамках Федеральной целевой программы "Научные и научно-педагогические кадры инновационной России" (государственный контракт, заключенный по результатам конкурса НК-682П).

ЛИТЕРАТУРА:

1. V. Voronov, N. Popova. Automatic Performance Tuning Approach for Parallel Applications Based on Linear Solvers // Abstract Book of Int. Conf. on Parallel Computations (ParCo-2009). 2010.
2. Cortes, C. and Vapnik, V. N. Support-Vector Networks // Machine Learning. Vol. 20. P. 273-297. 1995.
3. H. Wabnig, G. Kotsis, G. Haring. Performance Prediction of Parallel Programs. // Tagebuch von Messung, Modellierung und Bewertung. P. 64-76. 1993.
4. I. Salawdeh, E. Cesar, A. Morajko, T. Margalef, E. Luque. Performance Model for Parallel Mathematical Libraries Based on Historical Knowledgebase // LNCS. Vol. 5168. P. 110-119. 2008.
5. B. Barnes et al. A regression-based approach to scalability prediction //Proc. of the 22nd Int. Conf. on Supercomputing. P. 368-377. 2008.
6. S. Balay et al. PETSc web page. http://www. mcs. anl. gov/petsc
7. V. Eijkhout, E. Fuentes. A proposed standard for numerical metadata // Innovative Computing Laboratory, University of Tennessee, Tech. Rep. ICL-UT-03-02. 2003.
8. C. Chang, C.Lin. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm
9. T. Davis. University of Florida sparse matrix collection // NA digest. Vol. 97. N. 23. P. 7. 1997.