УДК 62-82:658.512.011.56

ЛАДОГУБЕЦ В.В.,
КРАМАР А.В.,
ФИНОГЕНОВ А.Д.

РАСЧЕТ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПАРАЛЛЕЛЬНОГО МЕТОДА ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ НА ОСНОВЕ РАЗНОСТЕЙ ВЫСШИХ ПОРЯДКОВ

Рассмотрены способы определения времени выполнения параллельных методов на мультипроцессорных вычислительных системах. Получены основные соотношения, позволяющие априорно оценить время выполнения параллельного метода численного интегрирования для минимизации времени решения. Показана возможность использования полученных оценок для автоматического выбора метода.

The ways to define execution time of the parallel methods at the multiprocessor computing systems have been considered. The main relations have been obtained which allow estimating execution time of the numerical integration parallel method a priori to minimize solution time.

При разработке параллельных программ важную роль играет возможность предварительной оценки качественных и количественных характеристик реализуемого алгоритма. В теории параллельных вычислений принято выделять ряд основных количественных характеристик, таких как коэффициент ускорения параллельного алгоритма, коммуникационная трудоемкость, коэффициент масштабируемости и т.д. [1,2].

На практике для сложных вычислительных алгоритмов трудно оценить количество операций. Например, при решении задач численного интегрирования в пакетах схемотехнического проектирования, количество операций будет определяться как размерностью математической модели, так и моделями элементов схемы, видом задающих источников, наличием или отсутствием учета нестандартных ситуаций, связанных с демпфированием и т.д. При этом для одной и той же задачи при изменении начальных условий процесс решения может использовать различные ветви алгоритма. Эти факторы делают оценки, основанные на определении времени отдельных коммуникационных и вычислительных операций, применимыми лишь для задач, использующих «мелкозернистый» параллелизм. К таким задачам обычно относят решения систем линейных уравнений прямыми методами, операции умножения матриц и векторов и т.п.

Для задач, использующих «крупнозернистый» параллелизм, т.е. реализующих вычисления отдельных ветвей алгоритмов или использующих различные методы решения одной и той же задачи, целесообразно производить оценку по величинам, которые определяют тип передачи:

- $t_{1\text{-}1}(V)$ – время передачи данных объемом V по типу «один к одному»;
- $t_{p,1\text{-}\infty}(V)$ – время передачи данных объемом V по типу «один ко многим», где количество адресатов определяется значением p;
- $t_{p,\infty\text{-}1}(V)$ – время передачи данных объемом V по типу «многие к одному», где p – количество сообщений к одному адресату;
- $t_{p,\infty\text{-}\infty}(V)$ – время передачи данных объемом V по типу «многие ко многим», где p – количество принимающих/передающих адресатов.

Последнего типа передачи при создании параллельного алгоритма желательно избегать, т.к. подобные операции увеличивают нагрузку на средства коммуникации и усложняют контроль выполнения алгоритма.

Значения времени для различных типов передачи и различных объемов передаваемых данных могут быть определены до выполнения алгоритма, и использоваться для предварительной оценки качества реализации на конкретной мультипроцессорной вычислительной системе (МВС). Если объем передаваемых данных каким-либо образом зависит от размерности задачи (что чаще всего встречается на практике), целесообразно построить таблицу зависимости времени передачи от передаваемого объема данных. Наибольшее внимание необходимо уделять точности определения времени передачи не-

больших объемов данных, т.к. при их передаче значительными оказываются влияния особенностей работы сетевых протоколов, различных типов оперативной памяти и т.д. С увеличением объемов данных их влияние становится менее заметно и для практических оценок ими можно пренебречь.

Оценки времени, основанные на типе передачи данных, не только могут дать информацию о качестве реализации алгоритма, но и позволить модифицировать схемы обмена данными, выбрать оптимальные маршруты передачи, количество процессоров, которые целесообразно использовать для ускорения работы алгоритма или, при фиксированном количестве процессоров, выбрать метод решения и т.д.

Рассмотрим расчет времени выполнения параллельного метода численного интегрирования на основе разностей высших порядков с использованием выборки всех доступных порядков и набора размеров шагов на каждом шаге интегрирования [3], реализованного в составе пакета схемотехнического САПР ALLTED [4] на кластере НТУУ «КПИ» [5]:

$$T_p = T_{p,comp} + T_{p,comm.}, \quad (1)$$

где T_p – общее время реализации параллельного алгоритма на параллельной архитектуре;

$T_{p,comp}$ – время решения задачи заданного размера с использованием параллельного алгоритма на компьютере из p-процессоров без учета операций на обмен (время реализации вычислений); $T_{p,comm.}$ – коммуникационная трудоемкость или время выполнения межпроцессорных операций обмена.

Время решения задачи без учета операций обмена $T_{p,comp}$ включает в себя этапы:

– трансляцию файла задания и формирования мат. модели $T_{мат.модели}$;
– численного интегрирования $T_{интегр.}$;
– обработки результатов и генерации выходных файлов $T_{рез.}$.

$$T_{p,comp} = T_{мат.модели} + T_{интегр.} + T_{рез.} \quad (2)$$

Основная идея параллельного метода заключается в использовании на каждом шаге выборки $[K \times h]$ шагов и порядков и выборе лучших в качестве новых текущих значений. В этом случае, время, затрачиваемое параллельным алгоритмом на процесс численного интегрирования, зависит от времени выполнения следующих операций:

– генерации выборки шагов и порядков $T_{ген}$;
– формирования системы линейных уравнений $T_{фор}$;
– выполнения итераций Ньютона $T_{ин}$;
– оценки результатов и определения следующего базового размера шага $T_{оц}$.

Количество этапов формирования линейных систем, выборки порядков и размеров шага, а также количества оценок лучшего зависит от количества шагов интегрирования. Время выполнения итераций Ньютона зависит от количества итераций на шаге, т.е. качества определения прогнозированного значения.

$$T_{интегр.} = H(T_{фор} + T_{ген} + T_{оц}) + NT_{ин}, \quad (3)$$

где H – количество временных шагов, N – количество итераций Ньютона.

С ростом размера выборки время оценки $T_{оц}$ увеличивается незначительно, т.к. варианты могут быть отброшены еще на первом этапе сравнения с лучшим результатом. Время генерации выборки $T_{ген}$ так же незначительно, а время формирования $T_{мат.модели}$ и время генерации файлов результата $T_{рез.}$ является для конкретной задачи константой.

Таким образом, можно определить время, которое практически не влияет на суммарное время решения или является постоянным, т.е. не зависит от размерности выборки:

$$T_{p,const} = T_{мат.модели} + T_{рез.} + H(T_{ген} + T_{оц}) \quad (4)$$

С учетом (3,4), время реализации параллельного алгоритма $T_{p,comp}$ (2) будет определяться как:

$$T_{p,comp} = T_{p,const} + HT_{фор} + NT_{ин} \quad (5)$$

Т.к. в случае параллельного алгоритма значение N для каждого компонента выборки может лежать в пределах от 1 до $MAXIT$ (максимально допустимое количество итераций Ньютона на шаге), то в качестве N – необходимо учитывать максимальное количество итераций Ньютона, которое было получено на компонентах выборки. Пусть $isi1_n$ – количество итераций Ньютона, полученных на i-й компоненте выборки $[K \times h]$ на n-м шаге интегрирования, при этом размерность выборки равна p – количеству доступных процессоров. В этом случае, суммарное количество итераций Ньютона, выполняемых не одновременно, будет равно:

$$N_{max} = \sum max_n(isi1_i),\ i=1(1)p,\ n=1(1)H \quad (6)$$

С учетом (6), выражение (5) примет вид:

$$T_{p,comp} = T_{p,const} + HT_{фор} + N_{max}T_{ин} \quad (7)$$

Для определения коммуникационной сложности алгоритма $T_{p,comm}$, рассмотрим данные, которые необходимо передавать при параллельной реализации. Для этого необходимо выделить три типа данных, которые могут использоваться при обмене.

1) *Входные* или *константные данные*, т.е. данные, описывающие параметры решаемой системы и не изменяющиеся на протяжении всей работы алгоритма.

2) *Данные процесса решения*. К ним относятся данные, которые в пределах параллельных блоков не изменяются, но изменяются при анализе лучшего варианта и в дальнейшем необходимы для пересчета ряда параметров.

3) В*ыходные* или *результирующие данные*, к которым относятся результаты работы параллельных блоков и использующиеся для анализа лучшего варианта и расчета данных на следующем временном шаге.

В соответствии с выделенными типами, представим объем этих данных как:

– $V_{ВД}$ – объем входных данных;
– $V_{ДПР}$ – объем данных процесса решения;
– $V_{РД}$ – объем результирующих или выходных данных.

Очевидно, что входные данные нужны для начала работы алгоритма, однако в последствии их передача не требуется, т.е. обмен этими данными производится один раз перед первым шагом работы алгоритма. Соответственно, данные процесса решения требуют только передачи на каждом временном шаге, т.к. их значения изменяются вне параллельных блоков. Для результирующих данных должен происходить только процесс приема на каждом шаге.

Рассчитаем объем данных, которые необходимо будет передавать при решении задачи численного интегрирования. На первом шаге интегрирования параллельным блокам необходимо передать входные данные и данные процесса решения.

$$V_{m,1} = V_{ВД} + V_{ДПР} \quad (8)$$

Начиная со второго и для последующих шагов, объем передаваемых данных в параллельные блоки будет составлять:

$$V_{m,2-\infty} = (H-1) \cdot V_{ДПР} \quad (9)$$

Исключение составляет последний шаг. В случае достижения предела интегрирования, необходима передача сигнала о завершении работы параллельных блоков. Однако, в этом случае, головной программе нет необходимости ожидания, поэтому данная передача практически не будет влиять на время выполнения. Объем возвращаемых данных будет всегда одинаков:

$$V_s = HV_{РД} \quad (10)$$

В данном случае не учитывается неполная нагрузка на процессоры на начальном этапе. Очевидно, что полная выборка $[K \times h]$ будет сформирована по достижению значения порядка K_{max}, однако на практике это чаще всего происходит уже на первых десяти шагах и практически не влияет на оценку времени выполнения или коэффициента эффективности в виду большого (10^3 и более) количества шагов для решения.

В качестве архитектуры параллельной программы удобно выбрать архитектуру «ведущий (master) –подчиненный (slave)», где на slave возлагаются этапы формирования линейной системы уравнений $T_{фор}$ и проведение итераций Ньютона $T_{ин}$. Функции master включают в себя трансляцию входного файла задания и формирование математической модели $T_{мат.модели}$, генерацию выборки $T_{ген}$, оценку результатов для выбора лучшего $T_{оц}$ и формирование выходных файлов результата $T_{рез.}$.

Для оценки времени решения, рассмотрим схему обмена данными, которая не учитывает особенности архитектуры (рис. 1), и проведем ее модификацию для систем с распределенной памятью.

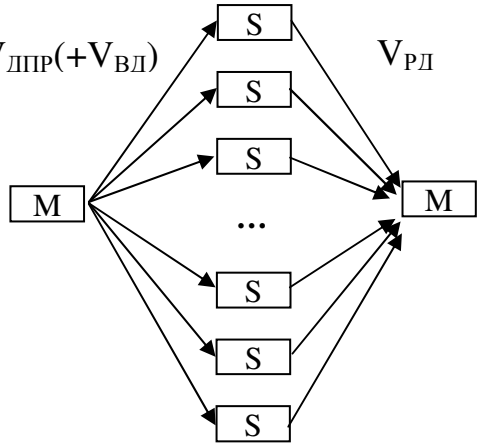

Рис. 1. Схема обмена данными

Для данной схемы обмена данными необходимо определить время выполнения такого типа обмена, как «один к одному», т.е. многократную пересылку данных процесса решения от master к slave и тип передачи «многие к одному», т.е. сбор результирующих данных.

В этом случае $T_{p,comm}$ с учетом (8-10) будет определятся как:

$$T_{p,comm} = pt_{1\text{-}1}(V_{m,1}) + p(H\text{-}1)t_{1\text{-}1}(V_{m,2\text{-}\infty}) + H\, t_{p,\infty\text{-}1}(V_s)$$

или

$$T_{p,comm} = p(Ht_{1\text{-}1}(V_{ДПР}) + t_{1\text{-}1}(V_{ВД})) + H\, t_{p,\infty\text{-}1}(V_{РД}) \quad (11)$$

Недостатком такой схемы взаимодействия (рис. 1) является многократная пересылка практически одинаковых данных $V_{ДПР}$ каждому параллельному блоку (за исключением значений компонентов выборки $[K\times h]$). Поэтому на практике на этапе запуска slave-программ целесообразно определить уникальные идентификаторы каждой и в дальнейшем передавать весь массив значений выборки $[K\times h]$. Идентификатор slave-программы будет соответствовать номеру компоненты выборки и сделает весь объем передаваемых данных полностью идентичным для всех slave. В этом случае становится возможным использование на каждом шаге не множественной операции пересылки по типу «один к одному», а однократной пересылки «один ко всем».

Передача всего объема результирующих данных от всех slave также имеет ряд недостатков: с ростом размерности задач увеличивается объем требуемой памяти для хранения данных, большая часть из которых (за исключением данных, полученных на компоненте выборки признанной лучшей) в дальнейшем не используется. Кроме того, будет происходить рост объема передаваемых данных, что в свою очередь повлечет увеличение времени на их передачу. В связи с этим представляет интерес использование следующей схемы возврата значений расчетов slave: на первом этапе передаются только данные, которые необходимы для анализа лучшего варианта. После проведения анализа, master инициирует обмен всеми данными только с одним из slave, результаты которого признаны лучшими. Такая схема взаимодействия резко снижает объем передаваемых данных и объем требуемой памяти, но увеличивает количество требуемых обменов. Оценка эффективности по времени такой схемы обмена данными будет зависеть от соотношения времени передачи единицы данных и латентности сети.

Для большинства существующих систем соотношение латентности сети и времени передачи одного байта данных обычно лежит в пределах 10^2-10^3. Компенсация времени дополнительного обмена возможна при уменьшении объема предаваемых данных на 200-2000 байт, что возможно даже для небольших схем. Например, в зависимости от используемых моделей транзисторов, такое уменьшение будет наблюдаться для схем, состоящих из 4-5 транзисторов. В этом случае схема обмена данными примет вид, представленный на рис. 2.

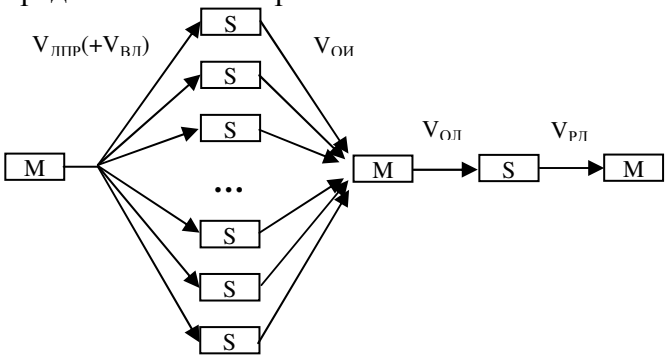

Рис. 2. Схема обмена данными для систем с распределенной памятью

Для такой схемы обмена необходимо определить следующие типы данных:

– $V_{ОИ}$ – объем информации для оценки лучшего (обычно не болем 100 байт);
– $V_{ОЛ}$ – признак того, что получивший данную информацию slave, должен передать все результирующие данные (1 байт).

В этом случае необходимо определение времени выполнение типов передачи по типу $t_{p,1\text{-}\infty}$ – «один ко многим» и $t_{p,\infty\text{-}1}$ – «многие к одному» для p slave, и $t_{1\text{-}1}$ – время выполнения операции передачи данных типа «один к одному».

Очевидно, что время выполнения этих операций будет зависеть от конфигурации, технических характеристик, а также от размещения соответствующих slave-процессов на архитектуре МВС. Для предложенной схемы обмена данными, коммуникационная сложность алгоритма (11) будет определяться следующим образом:

$$T_{p,comm} = T^1 + (H\text{-}1)T^2 \quad (12)$$

где T^1 – время, затрачиваемое на обмен на первом шаге; T^2 – время, затрачиваемое на обмен на втором и последующих шагах, которые определяются как:

$$T^1 = t_{p,1\text{-}\infty}(V_{ВД}+V_{ДПР}) + t_{p,\infty\text{-}1}(V_{ОИ}) + t_{1\text{-}1}(V_{ОЛ}) + t_{1\text{-}1}(V_{РД});$$

$$T^2 = t_{p,1\text{-}\infty}(V_{ДПР}) + t_{p,\infty\text{-}1}(V_{ОИ}) + t_{1\text{-}1}(V_{ОЛ}) + t_{1\text{-}1}(V_{РД}).$$

В случае использования систем с общей памятью, время решения будет в большей мере зависеть от времени выполнения итераций Ньютона, т.к. отсутствует необходи-

мость в передаче данных. Основными затратами на организацию параллельных вычислений в этом случае будут выступать затраты на копирование экземпляров данных в памяти, т.к. каждый параллельный блок производит вычисления на своей паре *(K,h)*, и, соответственно, не может использовать единый набор данных процесса решения. Входные же данные наоборот остаются неизменными в процессе всего решения и могут использоваться всеми параллельными блоками совместно.

Процесс выбора лучшего в этом случае сводится к определению экземпляра результирующих данных, который необходимо скопировать остальным параллельным блокам (рис. 3).

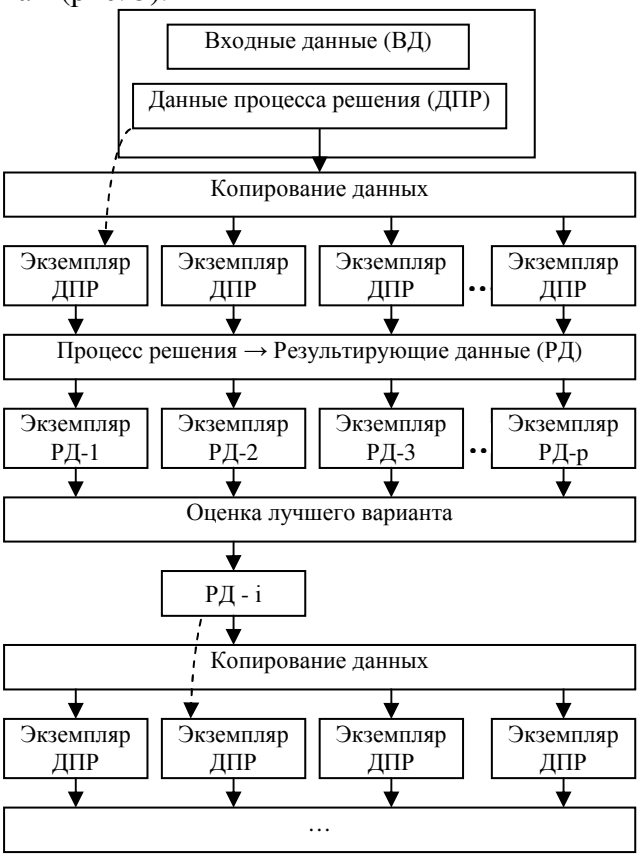

Рис. 3. Схема работы алгоритма с использованием общей памяти

Коммуникационная сложность данной схемы будет в основном определяться временем копирования экземпляров данных процесса решения для *p* параллельных блоков после определения лучшего, т.к. затратами времени на синхронизацию, при решении сложных задач большой размерности можно пренебречь:

$$T_{p,comm} = (H+1) \cdot t_{p,k}(V_{ДПР}) \qquad (13)$$

где $t_{p,k}$ – время копирования экземпляров данных $V_{ДПР}$, а дополнительное копирование связано с начальным заполнением массивов данных.

Оценку полученных соотношений проведем для двух методов, использующих на каждом шаге интегрирования выборку из всех доступных порядков и набор размеров шагов с уменьшением: $h=\{h,0.95h,0.9h,0.8h,0.7h\}$. «Метод 1» для выбора лучшего использует оценку, основанную на максимизации прогнозированного времени $max(T_n+h_{n+1})$ и является примером «задачи на максимум», т.е. для оценки необходимо получение результатов работы всех параллельных блоков. «Метод 2» для выбора лучшего использует оценку, основанную на минимальном количестве итераций Ньютона *min(isi1)* и является примером «задачи на минимум», т.к. для определения лучшего необходимы только данные от одного(нескольких) успешных блоков, закончивших работу с минимальным количеством итераций, что позволяет использовать в качестве количества неодновременно выполняемых итераций Ньютона значение *N*. Для проверки полученных соотношений времени использовались временные характеристики, полученные на кластере НТУУ «КПИ».

В качестве теста использовалась схема FADD32 из тестового набора, предложенного институтом Беркли [6], с параметрами, приведенными в табл. 1.

Табл. 1. Параметры схемы FADD32

Параметры	**FADD32**		
	Базовый	**Метод 1**	**Метод 2**
Количество временных шагов, *S*	17407	9413	11925
Количество итераций Ньютона, *N*	31092	17571	14466
Количество отказанных шагов, *R*	1019	522	399
Максимальное число итераций Ньютона, N_{max}	31092	22961	27743
Объем входных данных, байт	-	271100	
Объем данных процесса решения, байт	-	66080	
Объем результирующих данных, байт	-	66816	-
Объем данных для оценки лучшего, байт	-	32	-
Объем данных обмена с лучшим, байт	-	1	-
Среднее время выполнения одной итерации Ньютона, с		$1077*10^{-6}$	
Среднее время формирования, с		$100*10^{-6}$	

Для выборки размером $p = 35$ в качестве МВС с распределенной памятью использовались пять 8-ми ядерных узлов кластера с равномерным распределением параллельных блоков в узлах. Время выполнения операции «один к одному» производилось с помощью отправки сообщения объемом V байт и ожидания ответа от slave. Для определения времени операции «один ко всем» использовалась аналогичная система отправки сообщений различного объема к slave с ожиданием получения ответа. В отличие от операции «один к одному» в данном случае интерес представляет непосредственно среднее время обмена, т.к. при малых объемах возвращаемых данных (менее 1кб) время фактически не меняется. Таким образом, полученное значение времени можно использовать для определения сразу суммы времен: времени передачи данных процесса решения и времени возврата данных для оценки лучшего:

$$t_{ДПР+ОИ} = t_{p,1-\infty}(V_{ДПР}) + t_{p,\infty-1}(V_{ОИ}) \qquad (14)$$

Для определения времени копирования 35 экземпляров данных процесса решения использовался запуск 35 потоков на одном узле кластера (табл. 2).

Для 8-ми ядер кластера коммуникационная сложность алгоритма на основании (13) рассчитывается как:

$$T_{p,comm} = L(H+1) \cdot t_{p,k}(V_{ДПР}) \qquad (15)$$

где L – коэффициент, округленный к большему целому отношения p/C – размерности выборки p к числу задействованных ядер C.

Соответственно, время выполнения параллельного метода (1) с учетом выражений (4), (13) и (15) для «Метода 2» примет вид:

$$T_p = T_{p,const} + L\,(HT_{фор} + NT_{ин} + (H+1)t_{p,k}(V_{ДПР})) \qquad (16)$$

где $H=S+R$.

Погрешность определения времени выполнения с использованием соотношений (7), (12) и (14) для систем с распределенной памятью и (16) для систем с общей памятью не превышает 20%, что делает приведенные соотношения применимыми для практической оценки времени выполнения параллельных алгоритмов (табл. 3).

Табл. 2. Время выполнения передачи данных

Время трансляции файла задания, генерации мат. модели и т.д., с	$T_{p,const} = 0.51$
Среднее время передачи данных для оценки лучшего, с	$t_{1-1}(V_{ОЛ})=36.735\cdot10^{-6}$
Среднее время передачи результирующих данных, с	$t_{1-1}(V_{РД})=0.48\cdot10^{-3}$
Среднее время передачи входных данных и возврат данных для оценки лучшего (первый шаг), с	$t_{ДПР+ОИ}(V_{ДПР}+V_{ВД})=19.1\cdot10^{-3}$
Среднее время передачи входных данных и возврат данных для оценки лучшего (кроме первого шага), с	$t_{ДПР+ОИ}(V_{ДПР})=4.029\cdot10^{-3}$
Время копирования данных процесса решения, с	$t_{35,k}(V_{ДПР})=0.47\cdot10^{-3}$

Табл. 3. Ожидаемое и реальное время выполнения (p=35)

Схема	Тип памяти	Распределенная память		Общая память		
	Метод	$t_{ожидаемое}$ C=35, с	$t_{реальное}$ C=35, с	$t_{ожидаемое}$ C=8, с	$t_{ожидаемое}$ C=35, с	$t_{реальное}$ C=8, с
Fadd32	Базовый	35.3	36.35	35.3	35.3	36.35
	Метод 1	70.9	88.6	152.46	36.47	182.37
	Метод 2	87.09	107.34	113.52	26.19	130.96

Полученные оценки времени выполнения параллельных алгоритмов численного интегрирования позволяют выбрать наиболее эффективный алгоритм решения. Базовые параметры решаемой задачи (табл. 2), необходимые для расчета времени выполнения, можно оценить как на основании априорного анализа математической модели задачи, так и в ходе работы алгоритма при видах анализа, использующих многократный расчет, например мультивариантном анализе или оптимизации параметров схемы во временной области.

Список литературы

1. Гергель В.П. Теория и практика параллельных вычислений / Гергель В.П. – М. : Бином. Лаборатория знаний, 2007. – 423 с.
2. Молчанов И.Н. Введение в алгоритмы параллельных вычислений / Молчанов И.Н. – К. Наукова думка, 1990. – 128 с.
3. Фіногенов О.Д Паралельний метод чисельного інтегрування жорстких систем диференційних рівнянь з визначенням кращого порядку на кроці / Фіногенов О.Д. // Комп'ютерні технології друкарства: Зб. наук. праць. – Львів : МОНУ Українська академія друкарства, 2008. – С. 108–113.
4. Petrenko A. ALLTED – a computer-aided engineering system for electronic circuit design / Petrenko A., Ladogubets V., Tchkalov V., Pudlowski Z – Melbourne: UICEE, 1997, 205 p.
5. Центр суперкомпьютерных вычислений НТУУ «КПИ». – Режим доступа: http://hpcc.org.ua/index.php
6. Набор тестовых схем института Беркли. CircuitSim90 Benchmark Information. – Режим доступа: http://www.cbl.ncsu.edu/CBL_Docs/csim90.htm

Поступила в редакцию 11.12.2009