Распараллеливание вычислений в CFD-решателе F при расчете трехмерных течений вязкого газа в многоступенчатых турбомашинах

Сергей В. Ершов[1], Виктор А. Яковлев[1], Мария Н. Гризун[2], Дмитрий А. Козырец[2]

[1] *Институт проблем машиностроения НАН Украины, ул. Д.Пожарского 2/10, Харьков, Украина*
[2] *Национальный технический университет «ХПИ», ул.Фрунзе, 14, Харьков, Украина*

yershov@ipmach.kharkov.ua, yava@ipmach.kharkov.ua,
masha.grizun@gmail.com, trancefer17@rambler.ru

Аннотация. *Статья посвящена вопросам распараллеливания вычислений при расчетах трехмерных турбулентных течений вязкого газа в многоступенчатых турбинах и компрессорах. Рассматривается CFD решатель F, предназначенный для численного интегрирования осредненных по Рейнольдсу уравнений Навье-Стокса в трехмерной расчетной области, представляющей собой проточную часть турбомашины. Распараллеливание выполнено применительно к многопроцессорным вычислительным системам с распределенной и общей памятью. Программная реализация распараллеливания использует межпроцессорный обмен информацией, построенный на основе сокет-интерфейса. Для этой цели привлекаются функции Windows Socket API. Тестирование разработанного алгоритма показало его достаточно высокую эффективность при распараллеливании задачи на 8 потоков. Рассмотрены направления дальнейшей работы по совершенствованию алгоритмов распараллеливания.*

Ключевые слова

Распараллеливание, сокет-интерфейс, CFD решатель, многоблочная область, турбомашины.

1 Введение

Общей тенденцией развития современных CFD-решателей является использования параллельных вычислений для ускорения расчетов. Распараллеливание выполняется на вычислительных системах с общей и распределенной памятью, на центральных и графических процессорах. Каждый из этих видов распараллеливания имеет свои особенности и свою специфику реализации. Для распараллеливания обычно используются специальные библиотеки и языки, такие как MPI, PVM, Cuda, OpenCL и др. Можно найти большое количество публикаций на эту тему, например [1-6].

В настоящей статье рассмотрено распараллеливание вычислений в CFD-решателе F [7], разработанном на основе кода FlowER [8]. Распараллеливание выполнено для вычислительных систем с общей и распределенной памятью с помощью межпроцессорного обмена информацией, построенного на основе сокет-интерфейса. Библиотеки параллельных вычислений типа MPI при этом не применялись.

2 Описание CFD-решателя

Трехмерное турбулентное течение вязкого теплопроводного сжимаемого газа в проточной части многоступенчатой турбомашины: турбины или компрессора – описывается системой нелинейных уравнений Навье-Стокса, осредненных по Рейнольдсу. Используется семейство дифференциальных моделей турбулентности семейства k-ω [9]. Исходные уравнения интегрируются численно с помощью итерационной явно-неявной разностной схемы, явный оператор которой основан на ENO схеме Годунова [10], а неявная

аппроксимация реализована с помощью итерационного метода Ньютона [11] и схемы Бима-Уорминга-Стегера [12].

Расчетная область имеет многоблочную структуру. Каждый блок включает в себя область течения в одном межлопаточном канале одного венца – направляющего (спрямляющего) аппарата или рабочего колеса. Обмен информацией между блоками в исходном нераспараллеленном коде осуществляется на общих границах этих блоков одномерными массивами осредненных в окружном направлении газодинамических параметров (давление, плотность, компоненты скорости, параметры турбулентности).

При этом на каждой итерации расчета информация с выхода некоторого лопаточного венца передается на вход следующего венца и, наоборот, с входа текущего венца передается на выход предыдущего. Алгоритм реализован таким образом, что начало следующей итерации вычислительного процесса возможно только после завершения этого обмена данными. В противном случае на одном этапе вычислений использовались бы данные с различных итераций, что может негативно сказаться как на скорости сходимости решения, так и на его устойчивости.

Программные модули CFD-решателя F написаны на алгоритмическом языке Fortran-95 и включают в себя, кроме собственно решателя, препроцессор для подготовки данных, модули расчета координат сеточных узлов, характеристик сетки, построения начального приближения и постпроцессор для обработки результатов расчетов. Графические возможности программной оболочки реализованы с помощью средств GUI Windows API, в то время как сам решатель использует стандарт алгоритмического языка и может работать в других операционных системах.

3 Реализация распараллеливания вычислений для многоблочных областей

Расчетная область многоступенчатой турбомашины имеет многоблочную структуру и для каждого из блоков необходимо проводить однотипные вычисления. Как правило, количество таких блоков (лопаточных венцов) может варьироваться от одного до двух-трех десятков. В многопроцессорной системе логично вычисления для каждого блока поручить одному процессору (ядру или потоку). Причем, это может быть как система с общей памятью, так и с распределенной. Если доступных процессоров меньше, чем лопаточных венцов, то имеет смысл в каждый блок включать по несколько лопаточных венцов.

Так как время расчета для каждого блока будет определяться, в основном, двумя факторами – размерностью разностной сетки в блоке и производительностью процессора, то при распределении лопаточных венцов по блокам и выборе количества узлов в каждом из них, необходимо это учитывать таким образом, чтобы исключить неравномерную загруженность процессоров. В программной оболочке пакета F реализовано автоматическое распределение вычислительной работы между выбранными процессорами многопроцессорной системы.

Каждый процесс идентифицируется номером, который определяет, для какого блока расчетной области выполняются вычисления. Во избежание конфликтов при одновременном открытии файлов на чтение (перед выполнением вычислений) и запись (после окончания вычислений) эти операции для всех процессов выполняются последовательно путем передачи-приема разрешений.

В итерационном цикле перед началом вычислений каждый процесс отправляет данные с границ блока соответствующей расчетной области, являющихся общими с блоками других процессов, а затем принимает информацию для этих же границ, отправленную другими процессами. Так как операция получения информации по сети может быть реализована как блокирующая (т.е. выполнение программы приостанавливается до окончания этой операции), то при правильной организации такого обмена не требуется никакого другого согласования процессов, и возникновение ситуаций взаимного блокирования процессов или рассинхронизации вычислений невозможно.

Реализация параллельных процессов в решателе F выполнена с помощью дополнительных процедур, написанных на алгоритмическом языке Fortran-95 с привлечением функций WinSock интерфейса прикладного программирования API [13,14]. По мнению авторов настоящей статьи, непосредственное использование сокет-интерфейса вместо стандартных библиотек типа MPI позволит достичь более высокой эффективности распараллеливания.

4 Организация удаленного запуска параллельно выполняемых задач

Программный комплекс F позволяет расчетчику автоматически выполнить локальный запуск нераспараллеленного задания на процессоре того компьютера, на котором запущена управляющая оболочка.

Для распараллеленных заданий необходима возможность удаленного запуска. В частности, при расчетах на распределенной вычислительной системе отдельные ее компьютеры могут одновременно использоваться для выполнения других вычислительных задач. Если же параллельные вычисления проводятся на удаленной многопроцессорной системе с общей памятью, то работа через терминал не всегда удобна и возможна, если, например, удаленная система использует операционную систему Linux.

В связи с этим в комплексе программ F реализован удаленный запуск параллельно выполняющихся расчетных заданий. На каждом удаленном компьютере, подготовленном для такого запуска, работает служба ParallelService, которая предназначена для

- связи с программами-клиентами, с которых удаленно запускаются задания;
- передаче программам-клиентам информации об удаленном компьютере, а именно: количество процессоров (ядер или потоков), их производительность и доступность;
- передаче программам-клиентам информации о запущенных на удаленном компьютере процессах;
- копирования необходимых для расчетов данных (файлов базы данных) с компьютера программы-клиента на удаленный компьютер и обратно;
- выполнение команд запуска и останова заданий на удаленном компьютере.

Служба ParallelService и программы-клиенты написаны на языке Fortran-95 с использованием функций Windows Socket API.

5 Тестирование алгоритма

Тестирование разработанного алгоритма выполнялось на ПК с процессором Intel Core i7-3770K, который имеет 4 ядра и, благодаря технологии hyper-threading, может реализовать 8 потоков. Рассчитывалось течение в проточной части компрессора, содержащей восемь венцов. В расчетной области была построена трехмерная разностная сетка с 10 616 832 (8x96x96x144) ячеек – более 1 300 000 ячеек в венце.

Результаты расчета течения в компрессоре приведены на рис. 1 и 2. Распределение осредненного в окружном направлении давления по проточной части компрессора показано на рис. 1, а изолинии числа Маха в средних тангенциальных сечениях лопаточных венцов компрессора даны на рис. 2. Результаты расчетов, полученные как без распараллеливания, так и с распараллеливанием, практически идентичны.

Выполнено три теста для оценки эффективности распараллеливания. В первом из них проведено четыре расчета: для нераспараллеленной задачи и распараллеливания на два, четыре и восемь потоков. При этом процессор используемого ПК не был загружен никакими дополнительными заданиями. Полученные результаты приведены в таблице 1. Ускорение распараллеливания рассчитывалась как

$$S_p = \frac{T_1}{T_p},$$

где T_1 – время исполнения нераспараллеленной задачи, T_p – время исполнения задачи при распараллеливании на p процессорах. Идеальное время распараллеливания определялось как

$$T_{ид} = \frac{T_1}{p}.$$

Эффективность использования процессоров (загруженность распараллеливания) находилась как отношение идеального времени распараллеливания к реальному:

$$\eta_p = \frac{T_{ид}}{T_p} = \frac{T_1}{pT_p}.$$

Рассматривалось астрономическое время выполнения задач.

График эффективности распараллеливания для данного теста приведен на рис. 3. Видно, что с ростом количества процессов эффективность распараллеливания резко снижается. В то же время следует заметить, что во всех блоках расчетной области, отдаваемых на один процесс каждый, использовалась одинаковая сетка, откуда следует приблизительно одинаковая вычислительная работа на каждом блоке.

Объем данных, пересылаемых и получаемых отдельным процессом на одной итерации, не превышал 12 Кб, следовательно, время пересылки должно быть несоизмеримо меньше времени вычислений на сетке, состоящей из более, чем миллиона ячеек. Поэтому такое снижение эффективности распараллеливания на четыре и восемь потоков не может быть объяснено ни ожиданием процессов друг друга, ни затратами времени на обмен.

Табл.1. Эффективность распараллеливания (тест 1)

Количество процессов	*Время одной итерации, с*	*Ускорение*	*Эффективность*
1	53,4	1,00	1,00
2	28,5	1,87	0,94
4	15,7	3,40	0,85
8	12,2	4,38	0,55

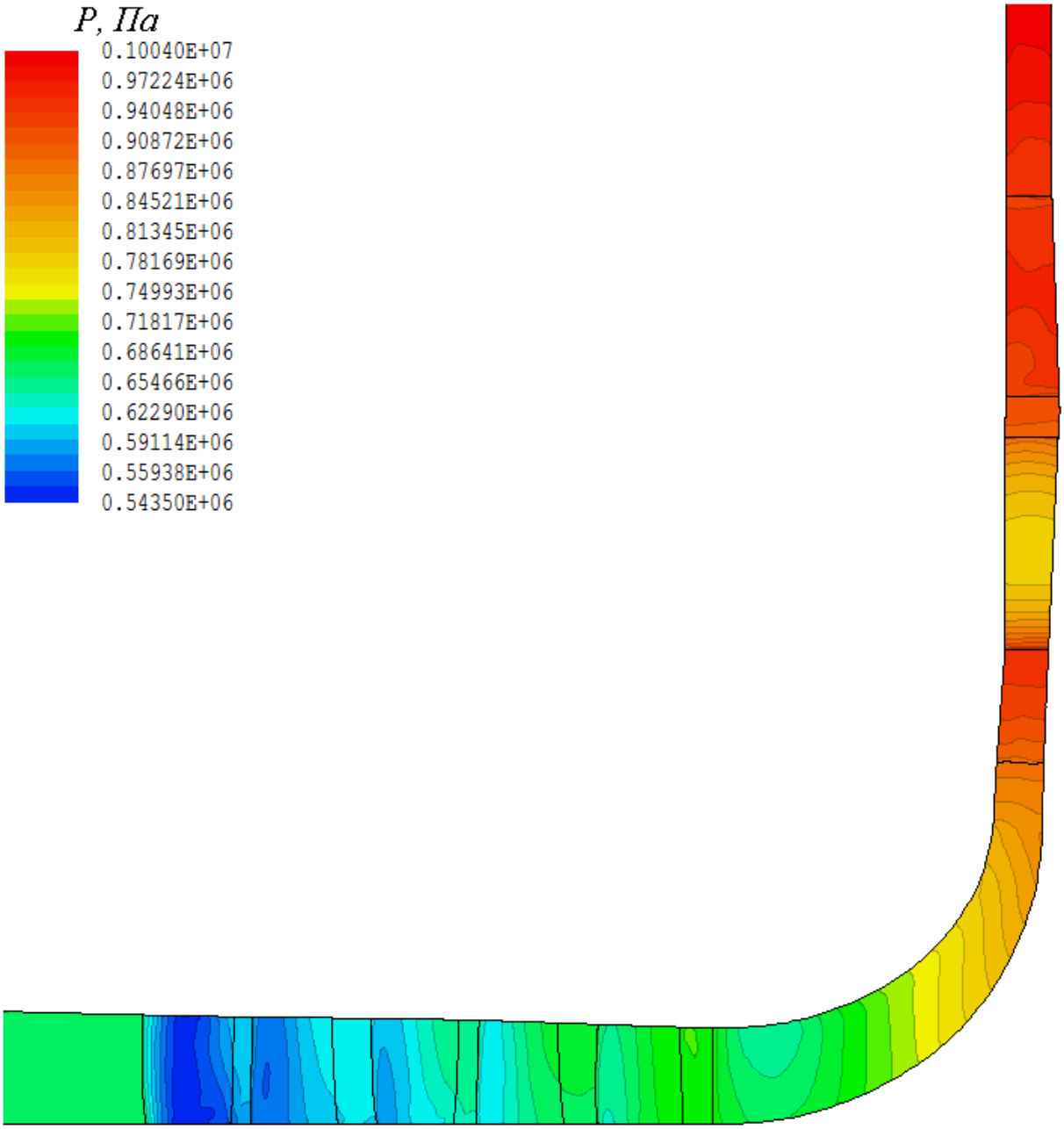

Рис. 1. Изолинии осредненного в окружном направлении давления в проточной части компрессора

В то же время было обнаружено, что время исполнения на процессоре Intel Core i7-3770K других задач, в том числе и не использующих межпроцессорный обмен данными, зависит от того, сколько заданий выполняется в данный момент на ПК. Поэтому в следующих тестах обеспечивалась полная загруженность всех потоков независимо от количества запускаемых параллельных процессов за счет запуска дополнительных «фантомных» процессов.

Во втором тесте нераспараллеленный расчет в одном потоке и распараллеливание на двух и четырех потоках проводились без использования технологии hyper-threading. В третьем тесте нераспараллеленный расчет и распараллеливание на два, четыре и восемь потоков были реализованы с использованием технологии hyper-threading..

Полученные результаты для обоих тестов приведены в таблицах 2 и 3, а график эффективности показан на рис. 4. Видно, что эффективность распараллеливания оказывается достаточно высокой. Потери на ожидание и межпроцессорный обмен данными оказываются незначительными по сравнению со временем расчета одной итерации. Следует заметить, что результаты теста 1, приведенные в таблице 1 и на рисунке 3, показывают, во сколько раз быстрее может быть получено численное решение при распараллеливании расчета, в то время как тесты 2 и 3 говорят именно о потерях при распараллеливании.

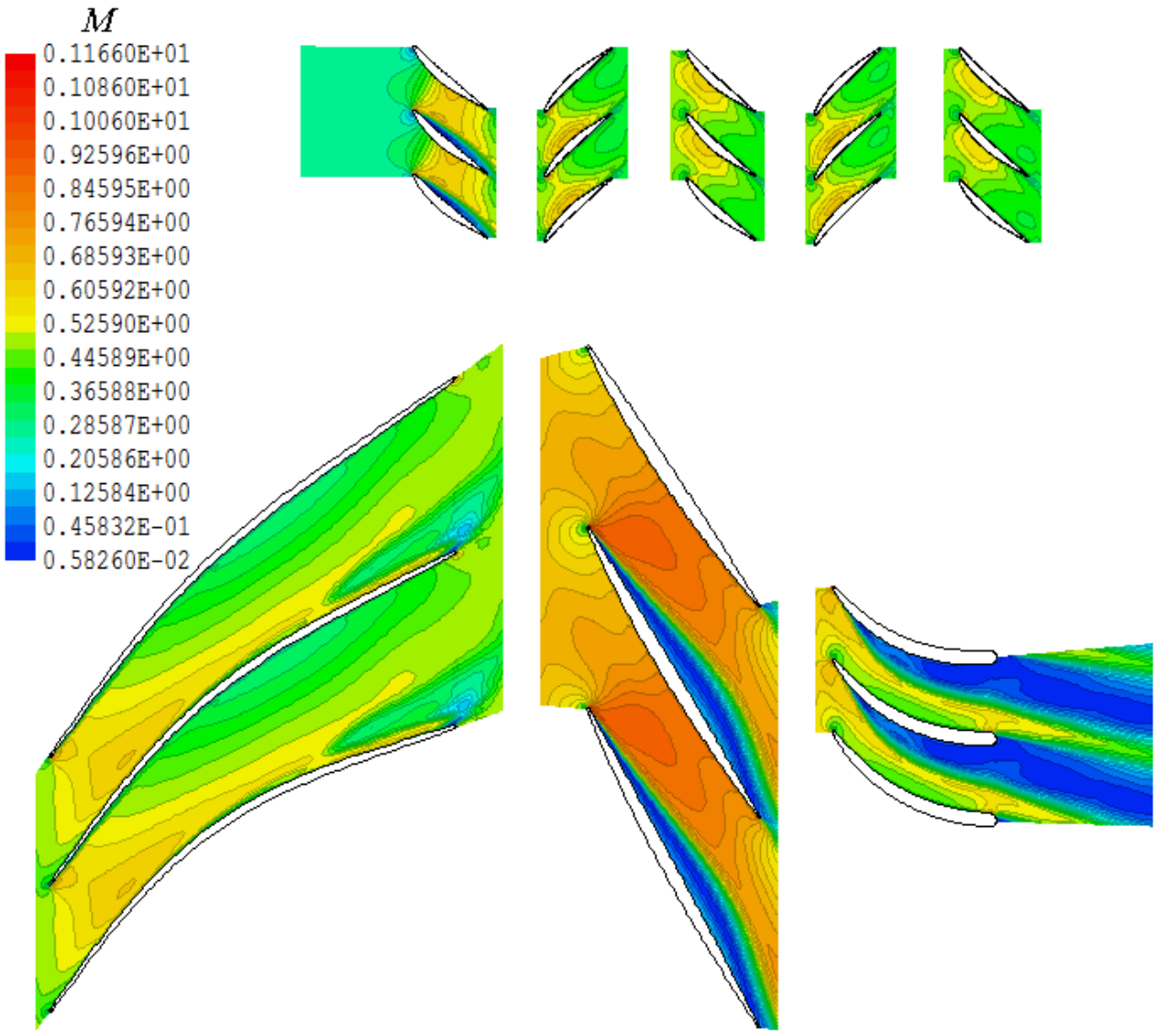

Рис. 2. Изолинии чисел Маха в среднем тангенциальном сечении проточной части

Табл.2. Эффективность распараллеливания (без hyper-threading)

Количество процессов	*Время одной итерации, с*	*Ускорение*	*Эффективность*
1	60,3	1,00	1,00
2	30,9	1,95	0,98
4	15,9	3,79	0,95

Табл.3. Эффективность распараллеливания (с hyper-threading)

Количество процессов	*Время одной итерации, с*	*Ускорение*	*Эффективность*
1	94,4	1,00	1,00
2	48,0	1,97	0,98
4	24,2	3,91	0,98
8	12,2	7,73	0,97

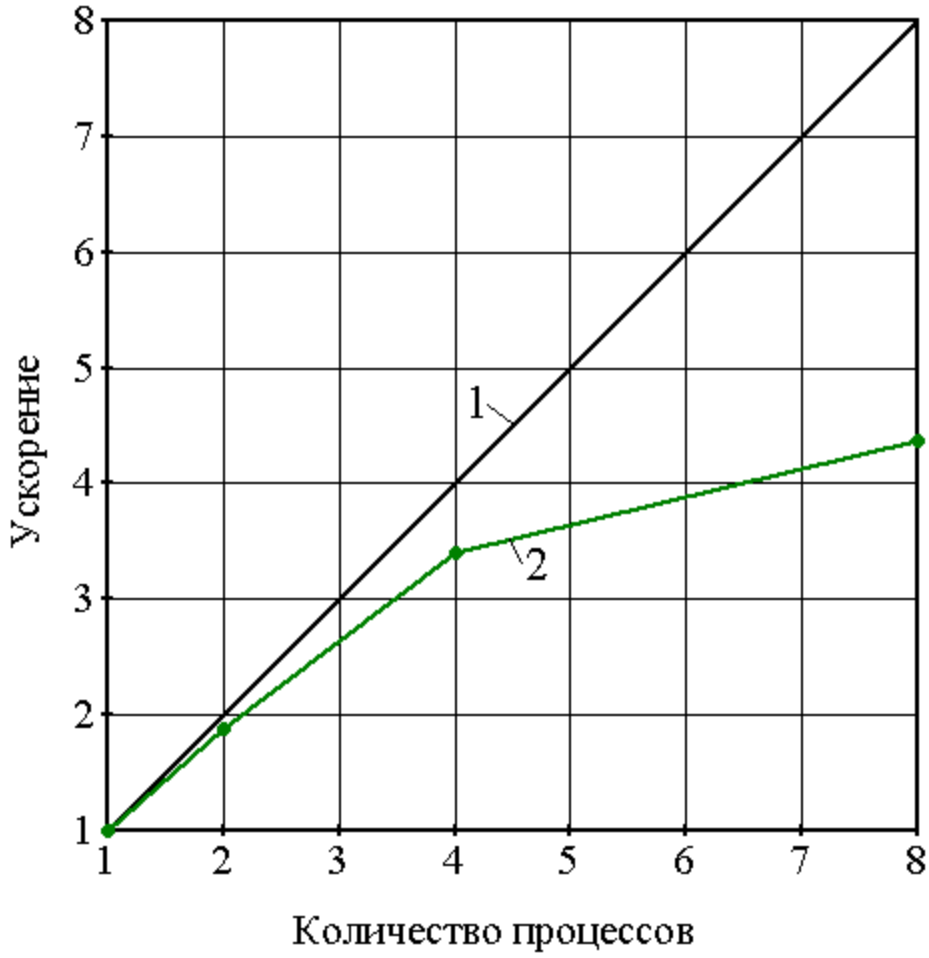

Рис. 3. Эффективность распараллеливания (тест 1)
1 – идеальное распараллеливание; 2 – реальное распараллеливание

6 Перспективные направления дальнейшей работы

Ближайшим заданием работы в данном направлении является адаптация разработанного программного обеспечения для функционирования в ОС типа Linux. В этом случае, кроме реализации удаленного запуска параллельных заданий на вычислительной станции с такой системой, появляется возможность полнофункциональной работы комплекса программ F на ПК с ОС Linux.

При этом оболочка, реализованная с использованием Windows API функций, может работать под эмулятором Windows, а запуск заданий в среде Linux будет осуществляться из этого эмулятора передачей соответствующих команд через сокет-интерфейс службе ParallelService, запущенной в среде Linux. Учитывая тот факт, что программирование сокетов в разных операционных системах выполняется подобным образом, то очевидно, что доработка разработанных программ для функционирования в ОС Linux не вызовет принципиальных трудностей.

Весьма перспективным представляется применение внутреннего многопотокового распараллеливания для использования возможностей современных графических процессоров. Исходя из опыта других авторов [4, 5, 15 и др.], решение такой задачи, требующее серьезной реорганизации программного кода, должно позволить достичь ускорения получения решений на один-два порядка, что является необходимым для включения CFD решателя в программные системы оптимизации проточных частей турбомашин. Реализация данной задачи может быть выполнена с помощью технологии CUDA.

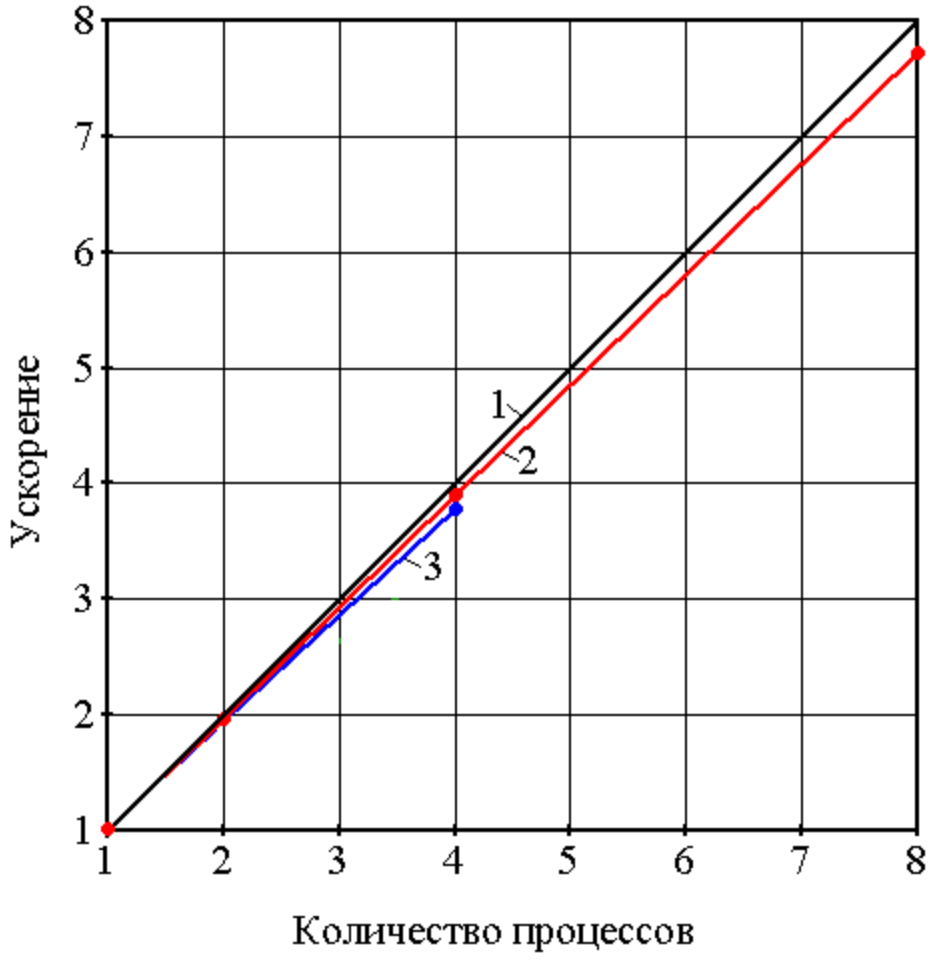

Рис. 4. Эффективность распараллеливания (тесты 2 и 3)
1 – идеальное распараллеливание; 2 – с hyper-threading;
3 – без hyper-threading

7 Заключение

Для CFD-решателя F реализовано распараллеливание вычислений на многопроцессорных системах с распределенной и общей памятью. Разработанный алгоритм основан на использовании функций Windows Socket интерфейса прикладного программирования API.

Тестирование показало, для задач течения вязкого газа в многоступенчатых турбомашинах при распараллеливании вычислений на ПК с процессором Intel Core i7-3770K на 4 или 8 потоков можно достичь уменьшения астрономического времени расчета в 3,4 и 4,4 раза соответственно. При этом эффективность распараллеливания достаточно высокая и потери на ожидание процессов и межпроцессорный обмен данными не превышают 5 % времени расчета.

8 Благодарность

Авторы выражают благодарность Д.С.Ершову за консультации по вопросам организации обмена данными между процессами с помощью сокет-интерфейса и д-ру В.В.Шмелеву за интерес, проявленный к нашим работам и приглашение к участию в конференции.

Литература

[1] X. Gao, C. P. T. Groth: Parallel Adaptive Mesh Refinement Scheme for Three-Dimensional Turbulent Non-Premixed Combustion. AIAA Paper, 2008-1017, 1-18, 2008.
[2] R. H. Bush, G. D. Power, C. E. Towne: WIND: The Production Flow Solver of the NPARC Alliance. AIAA Paper, 98-0935, 1-14, 1998.
[3] A. S. Antoniou, K. L. Karantasis, E. D. Polychronopoulos, J. A. Ekaterinaris: Acceleration of a Finite-Difference WENO Scheme for Large-Scale Simulations on Many-Core Architectures. AIAA Paper, 10-0525, 1-12, 2010.
[4] K. E. Niemeyer, C.-J. Sungy: Accelerating reactive flow simulations using graphics processing units. 51st AIAA Aerospace Sciences Meeting, Grapevine, Texas, USA. 6–10 January, 1-13 2013
[5] T. Brandvik, G. Pullan: Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware. Journal of Mechanical Engineering Science, 221(12):1745–1748, 2007.
[6] J. Schalkwijk, E. J. Griffith, F. H. Post, H. J. J. Jonker: High-Performance Simulations of Turbulent Clouds on a Desktop PC. Exploiting the GPU. Bulletin of the American Meteorological Society, 93(3):307-314. 2012
[7] С. В. Ершов, В. А. Яковлев, А.И.Деревянко, М.Н.Гризун, Д.А.Козырец: Развитие комплекса программ расчета трехмерных течений вязкого сжимаемого газа в лопаточных аппаратах турбомашин // Энергетические и теплотехнические процессы и оборудование. Вестник национального технического университета «ХПИ», (5): 25-32, 2011.
[8] С. В. Єршов, А. В. Русанов: Комплекс програм розрахунку тривимірних течій газу в багатовінцевих турбомашинах «FlowER»: свідоцтво про державну реєстрацію прав автора на твір, ПА № 77; 19.02.96 – Державне агентство України з авторських та суміжних прав, 1 с., 1996.
[9] D. C. Wilcox: Turbulence Modeling for CFD. Second Edition. Palm Drive: DCW Industries, 540 p., 2004.
[10] С. В. Ершов: Квазимонотонная схема повышенной точности для интегрирования уравнений Эйлера и Навье–Стокса. Мат. моделирование, 6(11):63–75, 1994.
[11] С. В. Ершов, А. И. Деревянко, М. Н. Гризун: Метод Ньютона для неявной схемы численного интегрирования уравнений газовой динамики. Проблемы машиностроения, – 13(4):27–36, 2010.
[12] R. M. Beam, R. F. Warming: An implicit factored scheme for the compressible Navier-Stokes equations. AIAA Journal, 16(4):393-402, 1978.
[13] N .Lawrence: Compaq Visual Fortran: A guide to creating windows applications, Elsevier Digital Press, 534 p., 2002.
[14] P. Bonner: Network Programming with Windows Sockets, Prentice Hall, Englewood Cliffs, New Jersey, 328 p., 1995.
[15] X. S. Trompoukis: Numerical solution of aerodynamic-aeroelastic problems on graphics processing units, National Technical University Of Athens, PhD Thesis, 242 p., 2012.