МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА
Рецензент
к.т.н., доцент кафедры прикладной математики ЮУрГУ
_____ Т.Ю. Оленчикова
“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___”_____ 2017 г.

Q-ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ МЕТОДА ЯКОБИ ДЛЯ РЕШЕНИЯ СЛАУ НА СУПЕРКОМПЬЮТЕРЕ «ТОРНАДО ЮУРГУ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2017.115-045.ВКР

Научный руководитель
к.ф.-м.н., доцент кафедры СП
_____В.Н. Алеева

Автор работы,
студентка группы КЭ-402
_____Ю.С. Лаптева

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____ 2017 г.

Челябинск-2017

Оглавление

ВВЕДЕНИЕ

Актуальность

Параллельные вычисления – способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно). Термин охватывает совокупность вопросов параллелизма в программировании, а также создание эффективно действующих аппаратных реализаций. Теория параллельных вычислений составляет раздел прикладной теории алгоритмов.

В настоящее время развитие параллельных вычислений необходимо для того, чтобы программы могли в полной мере использовать ресурсы высокопроизводительных вычислительных систем.

Современный подход к распараллеливанию линейных алгоритмов не подразумевает решения этой задачи в общем виде. Однозначного и общепринятого критерия для оценки степени параллелизма программы или ресурса распараллеливания алгоритма не существует. И исследования в этих областях могут помочь в решении ряда практических задач. Например, зная ресурс распараллеливания конкретного алгоритма и характеристики и ограничения вычислительной системы, на которой этот алгоритм будет выполняться, будет возможно получить максимально быструю реализацию алгоритма.

В настоящее время активно предпринимаются попытки автоматизировать процесс распараллеливания программ [6], анализируется масштабируемость параллельных решений [15].

Именно поэтому очень важно исследовать возможные способы системного подхода к решению задачи максимально-возможного распараллеливания алгоритмов.

Одним из таких подходов является концепция Q-детерминанта [1, 4, 5, 9, 12-14].

Программирование и построение параллельных программ на основе

концепции Q-детерминанта позволяет использовать ресурс параллелизма алгоритма полностью, поэтому исследование производительности Q-эффективных программ на параллельных вычислительных системах может помочь решить ряд вопросов, которые в рамках существующей теории в области распараллеливания алгоритмов и их реализации на параллельных вычислительных системах, пока остаются без ответов [3, 8].

Цель и задачи исследования

Целью данной работы является создание Q-эффективных реализаций метода Якоби для решения СЛАУ для дальнейшего исследования их производительности на суперкомпьютере «Торнадо ЮУрГУ».

Для достижения поставленной цели было необходимо решить следующие задачи.

1. Изучение подхода к максимальному распараллеливанию алгоритмов, основанному на представлении алгоритмов в форме Q-детерминанта.
2. Изучение метода Якоби для решения СЛАУ.
3. Получение навыков работы с технологией OpenMP.
4. Получение навыков работы с технологией MPI.
5. Построение Q-детерминанта метода Якоби для решения СЛАУ.
6. Создание Q-эффективных реализаций алгоритма для общей и для распределенной памяти с использованием технологий OpenMP и MPI для распараллеливания.
7. Анализ зависимости эффективности и ускорения (относительно последовательной реализации) программ для общей и распределенной памяти от количества используемых процессоров.

Структура и объем

Работа включает в себя введение, четыре основных раздела, заключение и список литературы. Объем работы составляет 30 страниц, объем библиографии – 16 наименований.

Содержание работы

Первый раздел «Q-эффективная реализация алгоритма» содержит описание концепции Q-детерминанта и подхода к проектированию параллельных программ на основе Q-детерминанта алгоритма.

Второй раздел «Технологии параллельного программирования» содержит краткую информацию о технологиях OpenMP и MPI, используемых в рамках работы для построения параллельных программ.

Третий раздел «Проектирование и создание Q-эффективной реализации метода» содержит в себе построенный в ходе работы Q-детерминант метода и описание Q-эффективной реализации алгоритма, спроектированное на основе этого Q-детерминанта.

Четвертый раздел «Тестирование и эксперименты» содержит описание процесса тестирования написанных программ, и полученные данные об их быстродействии на суперкомпьютере «Торнадо ЮУрГУ»

В заключении подводятся итоги проделанной работы.

1. Q-ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА

Целью данной работы является разработка программы, выполняющей Q-эффективную реализацию алгоритма.

1.1. Концепция Q-детерминанта

Пусть α алгоритм для решения алгоритмической проблемы $\bar{y} = F(N, B)$, где $N = \{n_1, \ldots, n_k\}$ множество параметров размерности проблемы, N может быть пустым, B множество входных данных (счетное множество переменных), $\bar{y} = (y_1, \ldots, y_m)$ множество выходных данных, m является вычислимой функцией параметров N. $\bar{N}$ вектор $(\bar{n}_1, \ldots, \bar{n}_k)$, где $\bar{n}_i$ некоторое заданное значение параметра n_i $(1 \le i \le k)$. $\{\bar{N}\}$ множество всевозможных векторов $\bar{N}$. Q множество операций, используемых алгоритмом α. Любое однозначное отображение $w : \{\bar{N}\} \to V$, где V множество всех выражений над B и Q, называется безусловным Q-термом. Если при любом $\bar{N} \in \{\bar{N}\}$ и любой интерпретации переменных B $w(\bar{N})$ принимает значение логического типа, то w называется безусловным логическим Q-термом. Пусть $u_1, \ldots, u_l$ безусловные логические Q-термы, $w_1, \ldots, w_l$ безусловные Q-термы. Множество пар (u_i, w_i), где $i = 1, \ldots, l$, обозначается $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ и называется условным Q-термом длины l. Счетное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ называется условным бесконечным Q-термом, если $\{(u_i, w_i)\}_{i=1,\ldots,l}$ является условным Q-термом для любого $l < \infty$. Если не имеет значения, является Q-терм безусловным, условным или условным бесконечным, то его можно называть Q-термом.

Под вычислением безусловного Q-терма w при интерпретации B следует понимать вычисление выражения $w(\bar{N})$ при некотором $\bar{N} \in \{\bar{N}\}$. Для вычисления при заданной интерпретации B и некотором $\bar{N} \in \{\bar{N}\}$ условного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ необходимо, вычисляя выражения $u_i(\bar{N}), w_i(\bar{N})$, найти $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ такие, что $u_{i_0}(\bar{N})$ принимает значение *true*,

а значение $w_{i_0}(\overline{N})$ определено. В качестве значения $(\overline{u},\overline{w})$ нужно взять $w_{i_0}(\overline{N})$. Если установлено, что выражений $u_{i_0}(\overline{N}), w_{i_0}(\overline{N})$ не существует, то значение $(\overline{u},\overline{w})$ для данной интерпретации B и данного $\overline{N}$ не определено. Для вычисления при заданной интерпретации B и некотором $\overline{N} \in \{\overline{N}\}$ условного бесконечного Q-терма $(\overline{u},\overline{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ необходимо, вычисляя выражения $u_i(\overline{N}), w_i(\overline{N})$, найти $u_{i_0}(\overline{N}), w_{i_0}(\overline{N})$ такие, что $u_{i_0}(\overline{N})$ принимает значение *true*, а значение $w_{i_0}(\overline{N})$ определено. В качестве значения $(\overline{u},\overline{w})$ нужно взять $w_{i_0}(\overline{N})$. Если установлено, что выражений $u_{i_0}(\overline{N}), w_{i_0}(\overline{N})$ не существует, то значение $(\overline{u},\overline{w})$ для данной интерпретации B и данного $\overline{N}$ не определено.

Предположим, что I_1, I_2, I_3 подмножества множества $I = (1,\ldots,m)$ такие, что:

$I_1 \cup I_2 \cup I_3$;

$I_i \bigcap I_j = \varnothing$ $(i \neq j; i, j = 1,2,3)$;

одно или два из множеств I_i $(i = 1,2,3)$ могут быть пустыми.

Множество Q-термов $\{f_i\}_{i \in I}$ удовлетворяет условиям:

f_{i_1} $(i_1 \in I_1)$ – безусловный Q-терм, $f_{i_1} = w^{i_1}$;

f_{i_2} $(i_2 \in I_2)$ – условный Q-терм, $f_{i_2} = \{(u_j^{i_2}, w_j^{i_2})\}_{j=1,\ldots,l_{i_2}}$, l_{i_2} является вычислимой функцией параметров N;

f_{i_3} $(i_3 \in I_3)$ – условный бесконечный Q-терм, $f_{i_3} = \{(u_j^{i_3}, w_j^{i_3})\}_{j=1,2,\ldots}$.

Если алгоритм α состоит в том, что для определения y_i $(i \in I)$ требуется вычислить Q-терм f_i, то множество Q-термов y_i $(i \in I)$ называется Q-детерминантом алгоритма α, а представление алгоритма в виде $y_i = f_i$ $(i \in I)$ представлением в форме Q-детерминанта.

Реализацией алгоритма α, представленного в форме Q-детерминанта $y_i = f_i$ $(i \in I)$, называется вычисление Q-термов f_i $(i \in I)$ при заданной интер-

претации B и некотором $\overline{N} \in \{\overline{N}\}$. Если реализация такова, что выражения

$$W(\overline{N}) = \{w^{i_1}(\overline{N})\ (i_1 \in I_1);\ u_j^{i_2}(\overline{N}), w_j^{i_2}(\overline{N})\ (i_2 \in I_2,\ j = 1,\ldots,l_{i_2});\ u_j^{i_3}(\overline{N}), w_j^{i_3}(\overline{N})\ (i_3 \in I_3,\ j = 1,2,\ldots)\}$$

вычисляются одновременно (параллельно) и при вычислении каждого из выражений операции выполняются по мере их готовности, то реализация называется Q-эффективной. Q-эффективная реализация алгоритма с формальной точки зрения является максимально быстрой. Реализация алгоритма α называется выполнимой, если одновременно необходимо выполнять конечное число операций. Существуют алгоритмы, для которых Q-эффективная реализация не является выполнимой.

1.2. Проектирование параллельных программ на основе Q-детерминанта

Подход к разработке программы, выполняющей Q-эффективную реализацию алгоритма, основан на следующих утверждениях:

- Q-детерминант можно построить для любого численного алгоритма;
- Q-детерминант позволяет описать Q-эффективную реализацию алгоритма;
- если Q-эффективная реализация алгоритма является выполнимой, то можно разработать программу для ее выполнения.

Процесс разработки программы состоит из следующих этапов:

- построение Q-детерминанта алгоритма;
- описание Q-эффективной реализация алгоритма;
- если Q-эффективная реализация выполнима, то для нее разрабатывается программа.

Разработанную программу будем называть Q-эффективной, а процесс ее разработки Q-эффективным программированием. Q-эффективная программа полностью использует ресурс параллелизма алгоритма, т.к. выполняет его Q-эффективную реализацию. В связи с этим она не допускает дальнейшего распараллеливания.

На этапе 3 для разработки программы должны использоваться средства параллельного программирования. При использовании распределенной памяти исследования ограничиваются вычислениями по принципу «Мастер – Рабочие», который часто применяется на кластерных вычислительных системах. Для дальнейшего изложения вычислительный узел «Мастер» обозначим M, а узел «Рабочий» P.

2. ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Существуют различные технологические решения для создания параллельных программ, в рамках данной работы использовались технологии OpenMP и MPI.

2.1. Технология программирования OpenMP

OpenMP – это механизм написания параллельных программ для систем с общей памятью. [2,9,10]

Состоит из набора директив компилятора и библиотечных функций.

Позволяет достаточно легко создавать многопоточные приложения на C/C++.

Основной поток порождает дочерние потоки по мере необходимости.

Программирование с помощью технологии OpenMP происходит путем вставки в ключевые места программы директив компилятора. Компилятор интерпретирует эти директивы и вставляет библиотечные вызовы для распараллеливания участков кода.

Например, директива *#pragma omp parallel for* перед циклом указывает на то, что цикл следует разделить между потоками по итерациям.

Количество потоков можно контролировать из программы, или через среду выполнения программы – переменную окружения OMP_NUM_THREADS.

Для того, чтобы скомпилировать программу с поддержкой OpenMP компилятору следует указать дополнительный ключ:

mpicc -openmp prog.cpp

Программа, распараллеленная при помощи OpenMP чаще всего состоит из цепочки последовательных и параллельных участков (регионов) кода, как это изображено на рис. 1. На этом рисунке можно увидеть основной поток («master thread»), который в точке начала параллельного участка кода порождает дочерние потоки («threads»). В точке завершения параллельного региона происходит ожидание завершения исполнения всех запущенных потоков, и только после этого наступает момент завершения

параллельного региона. При этом в каждом параллельном регионе может оказаться различное количество параллельных потоков в зависимости от необходимости и ограничений системы.

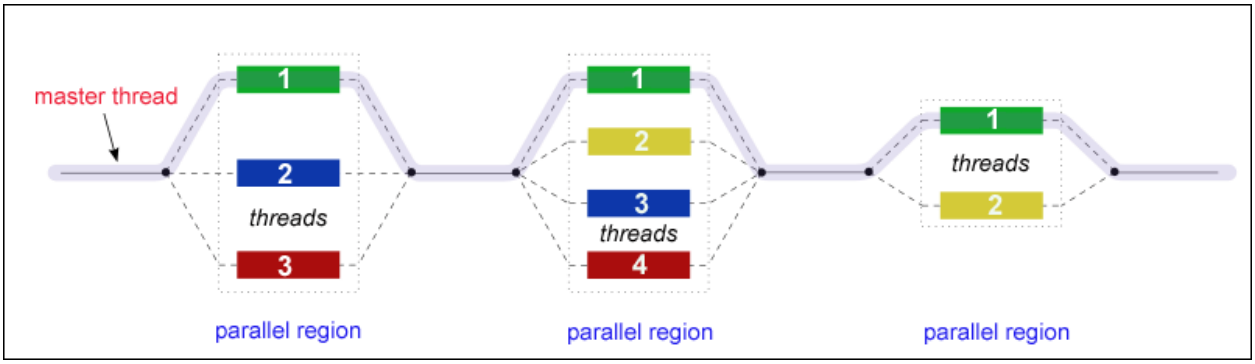

Рис. 1. Ход исполнения программы OpenMP

2.2. Технология программирования MPI

MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу.

MPI позволяет писать параллельные программы для систем с распределенной памятью.

Под параллельной программой в рамках MPI понимается множество одновременно выполняемых процессов. Процессы могут выполняться на разных процессорах, но на одном процессоре могут располагаться и несколько процессов (в этом случае их исполнение осуществляется в режиме разделения времени). В предельном случае для выполнения параллельной программы может использоваться один процессор – как правило, такой способ применяется для начальной проверки правильности параллельной программы.

Поскольку для коммуникации между процессами в MPI используется

передача сообщений на кластерных системах для написания эффективных параллельных программ чаще всего используется гибридное программирование MPI+OpenMP, т.е. для пересылки данных между узлами и взаимодействия между ними используется технология MPI, а для более быстрого взаимодействия между потоками на самих узлах (каждый из которых является системой с общей памятью) используется технология OpenMP [7, 16].

3. ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ Q-ЭФФЕКТИВНОЙ РЕАЛИЗАЦИИ МЕТОДА ЯКОБИ ДЛЯ РЕШЕНИЯ СЛАУ

3.1. Постановка задачи

Пусть дана система

$$\boldsymbol{A\vec{x} = \vec{b}}, \text{ где } \boldsymbol{A} = \begin{pmatrix} \boldsymbol{a_{11}} & \cdots & \boldsymbol{a_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a_{n1}} & \cdots & \boldsymbol{a_{nn}} \end{pmatrix}, \boldsymbol{\vec{b}} = \begin{pmatrix} \boldsymbol{b_1} \\ \dots \\ \boldsymbol{b_n} \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x_1} \\ \dots \\ \boldsymbol{x_n} \end{pmatrix}$$

Примем:

$$\boldsymbol{c_{ij} = -\frac{a_{ij}}{a_{ii}},\ d_i = \frac{b_i}{a_{ii}}}$$

Тогда итерационный процесс метода Якоби примет вид:

$$x^{(1)} = cx^{(0)} + d$$

$$x^{(2)} = cx^{(1)} + d$$

$$\dots$$

$$x^{(k+1)} = cx^{(k)} + d$$

где:

$$c = \begin{pmatrix} 0 & c_{12} \cdots c_{1n} \\ c_{21} & 0 \cdots c_{2n} \\ \dots & \cdots\cdots \dots \\ c_{n1} c_{n2} \cdots & 0 \end{pmatrix}, d = \begin{pmatrix} d_1 \\ d_2 \\ \dots \\ d_n \end{pmatrix}.$$

Критерий окончания итераций при достижении требуемой точности имеет вид:

$$\left\| x^{(k+1)} - x^{(k)} \right\| < \varepsilon.$$

3.2. Проектирование параллельной программы для Q-эффективной реализации алгоритма

Теперь построим Q-детерминант метода Якоби для решения СЛАУ и спроектируем параллельную Q-эффективную реализацию метода.

3.2.1. Этап 1

Количество Q-термов в Q-детерминанте алгоритма зависит от количества выходных данных. В случае с методом Якоби ответом будет решение системы, а значит вектор x из n элементов. Т.е. Q-детерминант метода

Якоби состоит из n условных бесконечных Q-термов.

Представление метода Якоби в форме Q-детерминанта имеет вид:

$$x_i = \left\{(\|x^1 - x^0\| < \varepsilon, x_i^1), \dots, \left(\left\|x^k - x^{k-1}\right\| < \varepsilon, x_i^k\right), \dots\right\} (i = 1, \dots, n)$$

3.2.2. Этап 2

Для упрощения описания Q-эффективной реализации введем обозначения:

$$u^l = \left\|x^l - x^{l-1}\right\| < \varepsilon \ (l = 1,2, \dots)$$

Тогда Q-детерминант имеет вид:

$$x_i = \left\{(u^1, x_i^1), \dots, \left(u^k, x_i^k\right), \dots\right\} (i = 1, \dots, n)$$

В соответствии с определением Q-эффективной реализации все безусловные Q-термы $\left\{u^l, x_i^l\right\} (i = 1, \dots, n; l = 1,2, \dots)$ должны вычисляться одновременно.

Сначала должны быть вычислены одновременно Q-термы $x_i^1 (i = 1,\dots,n)$, так как их операции готовы к выполнению. Затем вычисляются одновременно Q-термы $u^1, x_i^2 (i = 1,\dots,n)$. Если u^1 имеет значение *true*, то вычисления заканчиваются, а решением системы линейных уравнений является $x_i = x_i^1 (i = 1,\dots,n)$. Если вычисление будет продолжаться, то Q-термы $u^k, x_i^{k+1} (i = 1,\dots,n)$ будут вычисляться одновременно при любом значении $k \geq 2$. Если значение u^k – *true*, то вычисления заканчиваются, а решением системы линейных уравнений является $x_i = x_i^k (i = 1,\dots,n)$.

Q-эффективная реализация метода Якоби выполнима.

3.2.3. Этап 3

При использовании распределенной памяти каждая компонента вектора $\bar{x}^k (k = 1,2,\dots)$ вычисляется на своем вычислительном узле P. Если количество узлов P меньше n, то они должны выполнять вычисления для нескольких компонент вектора $\bar{x}^k (k = 1,2,\dots)$.

Сначала узел M посылает на узлы P необходимую информацию для вычисления Q-термов $x_i^1 (i = 1,\dots,n)$. Результаты вычисления передаются на

узел M. Узел M посылает на узлы P значения $x_i^1 (i = 1,...,n)$. Узел M вычисляет Q-терм $\left\| \overline{x}^1 - \overline{x}^0 \right\| < \varepsilon$. Вместе с этим на узлах P вычисляются $x_i^2 (i = 1,...,n)$ одновременно.

Следующие итерации выполняются аналогично.

3.3. Программирование Q-эффективной реализации алгоритма

Для выполнения поставленной задачи было решено разработать три независимые подсистемы с единственной функцией решения СЛАУ методом Якоби (рис. 2)

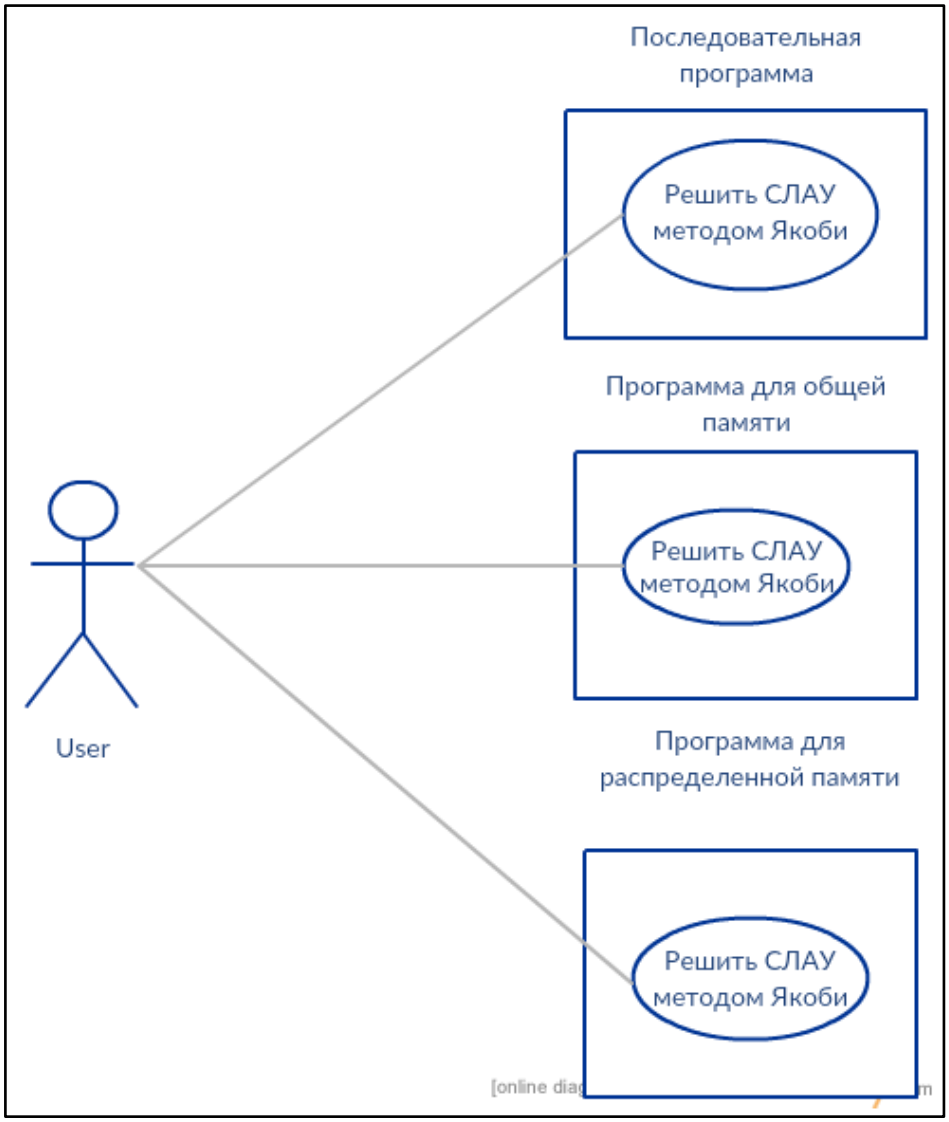

Рис. 2. Диаграмма вариантов использования

Также для тестирования каждого алгоритма была создана отдельная программа, включающая в себя возможность ввода данных, как с клавиатуры, так и из файла и вывода их в файл.

По плану выполнения, полученному в п. 3.2. данной работы, были разработаны Q-эффективная параллельная программа для общей памяти и

Q-эффективная параллельная программа для распределенной памяти. Для написания программ использовался язык программирования C++. Для общей памяти применялась технология OpenMP, для распределенной MPI и OpenMP. Также для повышения скорости выполнения разработанных программ была использована возможность компилятора генерировать векторные операции.

При вычислении каждого из Q-термов на каждом такте работы программы выполняются все готовые к выполнению операции. Если готовы к выполнению несколько операций цепочки, то они выполняются по схеме сдваивания (рис. 3).

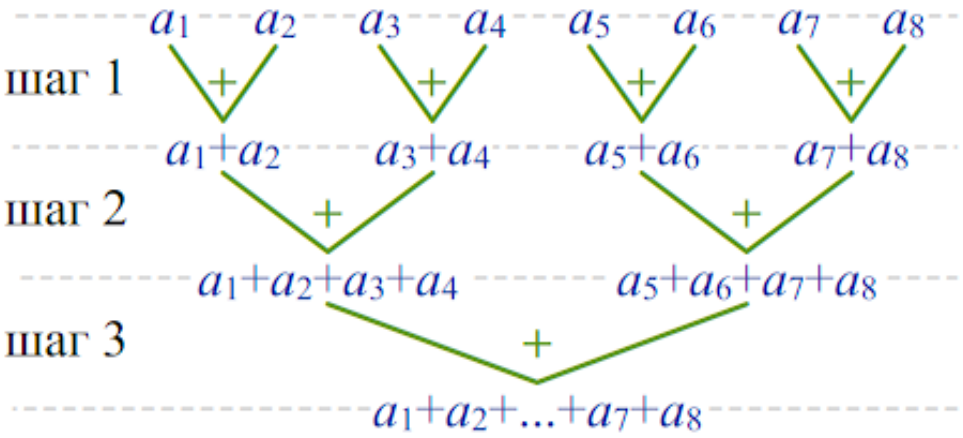

Рис. 3. Схема алгоритма сдваивания

Рассмотрим работу программы для СЛАУ из четырех уравнений. Так, например, граф выполнения Q-терма x_1^1 для такой СЛАУ можно увидеть на рис. 4.

Учитывая, что на первом такте работы программы вычисляются одновременно n Q-термов, можно посчитать, что операций для выполнения будет готово n(n-1). На последующих тактах это количество будет ниже, а значит не все процессоры будут загружены полностью, за счет чего эффективность программы будет ниже.

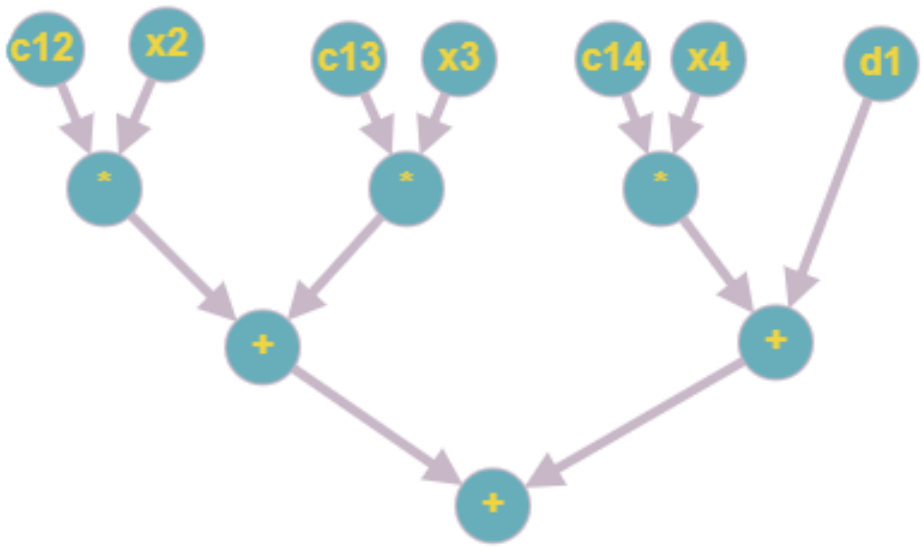

Рис. 4.Граф выполнения Q-терма $\boldsymbol{x_1^1}$ для СЛАУ размерности 4

4. ТЕСТИРОВАНИЕ И ЭКСПЕРИМЕНТЫ

4.1. Тестирование

Для проверки корректности работы разработанных программ было проведено функциональное тестирование.

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

В рамках данной работы были разработаны программы с единственной функциональной возможностью: решить СЛАУ методом Якоби. Для тестирования работы алгоритмов в программы были добавлены функции ввода данных с клавиатуры или из конфигурационного файла и вывода полученного решения в результирующий файл.

Также были найдены СЛАУ малых размерностей с заранее известными решениями. И сгенерированы единичные матрицы для проверки работы системы на больших размерностях систем. Найти СЛАУ больших размерностей с известными решениями не удалось.

После верного решения достаточно большого количества тестовых СЛАУ, работу программ можно считать корректной и для любых других СЛАУ.

4.2. Эксперименты

Одной из основных задач работы было получить данные о выполнении спроектированных программ на суперкомпьютере «Торнадо ЮУрГУ». Так как показатель чистого времени работы программы t на разных параллельных вычислительных системах будет зависеть от характеристик конкретной системы, а для исследования нужно было получить данные в более общем виде, было решено использовать такие показатели как S_p– ускорение работы параллельной программы ($S_p = T_1/T_p$, где T_p– время работы

параллельной программы, T_1– время работы последовательной программы) и E_p– эффективность параллельной программы ($E_p = S_p/p$)

На рисунках 3-7 представлены графики зависимостей полученные в ходе проведения экспериментов. Каждое значение времени, использованное для вычисления ускорения и эффективности, являлось средним временем на основе 10 экспериментов (для повышения точности экспериментальных данных).

4.2.1. Характеристики системы

Таблица 1. Технические характеристики суперкомпьютера «Торнадо ЮУрГУ»

Характеристики	Процессор
Число вычислительных узлов/процессорных ядер:	480/29184
Тип процессора:	Intel Xeon X5680 (Gulftown, 6 ядер по 3.33 GHz) – 960 шт.
Оперативная память:	16.9 TB
Дисковая память:	300 TB, твердотельные накопители SSD Intel, параллельная система хранения данных Panasas ActiveStor 11
Тип управляющей сети:	Gigabit Ethernet
Пиковая производительность комплекса:	473.6 TFlops
Производительность комплекса на тесте LINPACK:	288.2 TFlops
Операционная система:	Linux CentOS 6.2

Для вычислений (как единственно доступная параллельная вычислительная система с достаточно большим доступным количеством вычисли-

тельных узлов) был выбран суперкомпьютер «Торнадо ЮУрГУ», входящий в рейтинги TOP500, Green500 и СНГ TOP50.

Характеристики системы можно увидеть в таблицах 1-2.

Таблица 2 Характеристики вычислительного узла суперкомпьютера «Торнадо ЮУрГУ»

Характеристики	Процессор
Модель	Intel Xeon X5680
Количество ядер	12
Частота ядер, GHz	3.33
Количество потоков на ядро	2
Производительность, Тфлопс	0.371

4.2.2. Результаты запуска программы для общей памяти

На рис. 5 можно увидеть, как зависит ускорение работы программы для общей памяти от количества используемых вычислительных ресурсов.

Рис. 5. Зависимость ускорения работы программы для общей памяти от количества используемых процессоров

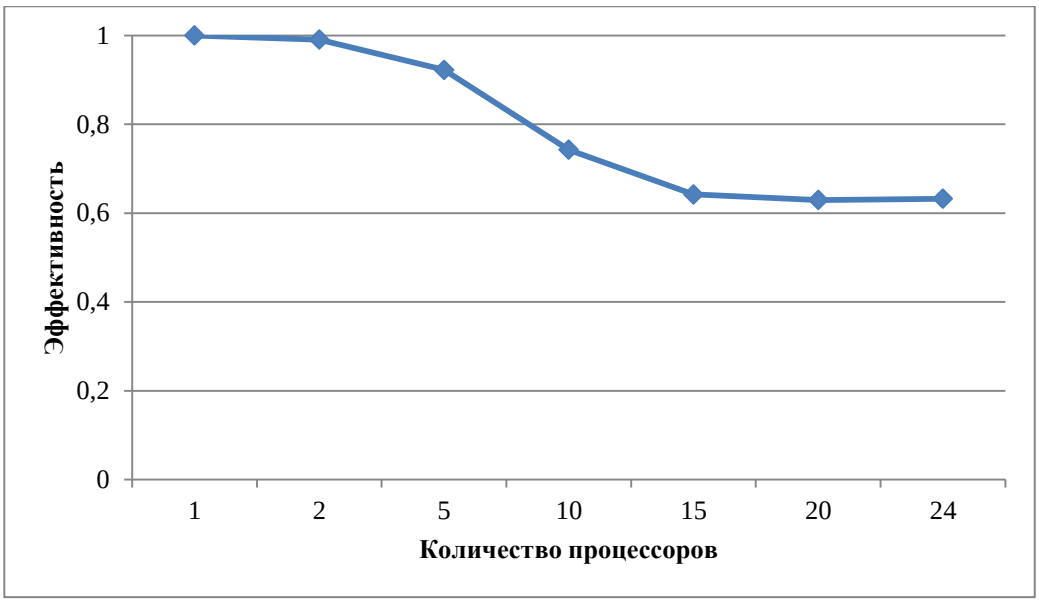

Рис. 6. Зависимость эффективности работы программы для общей памяти от количества используемых процессоров

Эффективность работы Q-эффективной реализации алгоритма для общей памяти на средних размерностях СЛАУ достаточно велика, что можно увидеть на рис. 6.

4.2.3. Результаты запуска программы для распределенной памяти

Для распределенной памяти последовательная программа обладает меньшей эффективностью (рис.8), чем при работе с общей памятью, скорее всего это связано с тем, что узел Мастер на каждой итерации метода должен получать результаты работы всех рабочих узлов и рассылать объединенные данные обратно на каждый узел. Эти пересылки являются дополнительными накладными расходами.

Помимо этого на эффективность работы программы влияет масштабируемость самого алгоритма. Легче всего проследить это по графику зависимости ускорения работы программы от количества процессоров. На рис. 7 видно, что ускорение работы программы достигает 45 и дальше пе-

рестает расти, это связано с тем, что доля последовательных вычислений алгоритма не позволяет достичь большего ускорения.

Рис. 7. Зависимость ускорения работы программы для распределенной памети от количества используемых процессоров

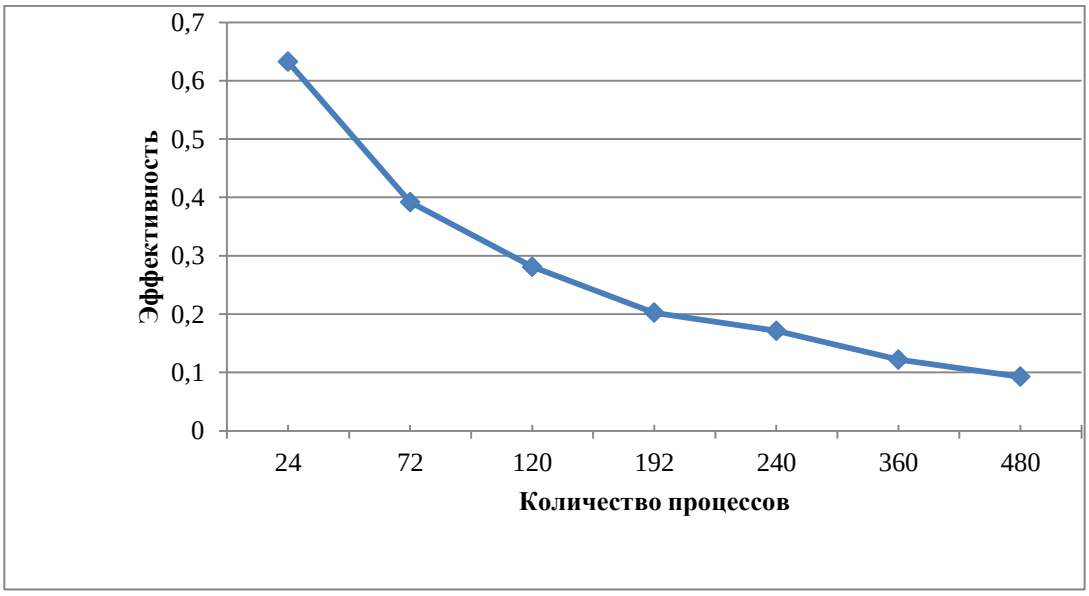

Рис. 8. Зависимость эффективности работы программы для распределенной памяти от количества используемых процессоров

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были разработаны Q-эффективные реализации алгоритма метода Якоби для решения СЛАУ для общей и распределенной памяти. Было проведено сравнение быстродействия полученных алгоритмов с последовательной реализацией этого метода и построены графики зависимостей ускорения и эффективности от количества используемых узлов при фиксированной размерности системы.

Были решены следующие задачи:

- Изучен подход к максимальному распараллеливанию алгоритмов, основанному на представлении алгоритмов в форме Q-детерминанта.
- Изучен метода Якоби для решения СЛАУ
- Получены навыки работы с технологией OpenMP
- Получены навыки работы с технологией MPI
- Построен Q-детерминанта метода Якоби для решения СЛАУ
- Созданы Q-эффективные реализации алгоритма для общей и для распределенной памяти
- Получены экспериментальные данные о работе спроектированных и реализованных на основе концепции Q-детерминанта программ на параллельной вычислительной системе суперкомпьютере «Торнадо ЮУрГУ»

Основной целью работы было реализовать Q-эффективную реализацию численного алгоритма, чтобы проверить ее поведение на реальной параллельной вычислительной системе, в данном случае суперкомпьютере «Торнадо ЮУрГУ».

Следует заметить, что Q-эффективная программа полностью использует ресурс параллелизма алгоритма, т.к. выполняет его Q-эффективную реализацию, поэтому дальнейшее распараллеливание такой программы невозможно.

В дальнейшем планируется использование полученных данных для составления расширенной математической модели концепции, учитывающей влияние накладных расходов на время выполнения Q-эффективной программы.

Также планируется провести сравнение Q-эффективных программ с другими существующими параллельными программами, реализующими данный метод.

ЛИТЕРАТУРА

1. Aleeva V.N., Sharabura I.S., Suleymanov D.E. Software System for Maximal Parallelization of Algorithms on the Base of the Conception of Q-determinant, 13th International Conference on Parallel Computing Technologies, PaCT 2015, Petrozavodsk, Russian Federation, 31 August - 4 September 2015; Proceedings. Lecture Notes in Computer Science, Springer, 2015. – Vol. 9251. P. 3-9.

2. Chandra R., Menon R., Dadum L., Kohr D. Parallel Programming in OpenMP Kindle Edition // UK: Morgan Kaufmann, 2001. – 163 c.

3. Voevodin V.V., Voevodin Vl.V. The V-Ray Technology of Optimizing Programs to Parallel Computers // Proc. of the 1st workshop on numerical analysis and applications, Russe, Bulgary, 24-27 June, 1996.

4. Алеева В.Н. Анализ параллельных численных алгоритмов: Препринт № 590 – Новосибирск: ВЦ СО АН СССР, 1985. – 23 с.

5. Алеева В.Н., Сулейманов Д.Э., Шарабура И.С., “Разработка программной системы QStudio для получения максимально быстрой реализации алгоритма”, Параллельные вычислительные технологии (ПаВТ’2015): труды международной научной конференции (31 марта – 2 апреля 2015 г., г. Екатеринбург), Издательский центр ЮУрГУ, Челябинск, 2015. – 523 с.

6. Бахтин В.А., Бородич И.Г., Катаев Н.А., Клинов М.С., Ковалева Н.В., Крюков В.А., Поддерюгина Н.В. Диалог с программистом в системе автоматизации распараллеливания САПФОР // Вестник ННГУ. 2012. №5-2. URL:http://cyberleninka.ru/article/n/dialog-s-programmistom-v-sisteme-avtomatizatsii-rasparallelivaniya-sapfor (дата обращения: 10.06.2017).

7. Бахтин В.А. Гибридная модель параллельного программирования DVM/OpenMP: диссертация кандидата физико-математических наук: 05.13.11 / Институт прикладной математики им. М.В. Келдыша РАН. – Москва, 2008. – 122 с.

8. Воеводин, Вл.В. Параллельные алгоритмы под микроскопом / Вл.В. Воеводин // Доклад на международной конференции «Параллельные

вычислительные технологии (ПаВТ’2016)» (28 марта - 1 апреля 2016 г., г. Архангельск).

9. Игнатьев С.В. Определение ресурса параллелизма алгоритмов на базе концепции Q-детерминанта: Вып. квалиф. работа магистра прикладной математики и информатики: 010500.68 /Южно-Уральский государственный университет. – Челябинск, 2009. – 75 с.

10. Левин М.П. Параллельное программирование с использование OpenMP. // Учебное пособие. М.: БИНОМ. Лаборатория знаний, Интернет-Университет Информационных Технологий (ИНТУИТ), 2008. – 120 с.

11. Официальный сайт проекта OpenMP. URL: http://www.openmp.org (дата обращения: 15.01.2017)

12. Свирихин Д.И., Алеева В.Н. Определение максимально эффективной реализации алгоритма на основе концепции Q-детерминанта // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (1–5 апреля 2013 г., г. Челябинск). – Челябинск: Издательский центр ЮУрГУ, 2013. – С. 617.

13. Свирихин, Д.И. Определение ресурса параллелизма алгоритма и его эффективного использования для конечного числа процессоров // Научный сервис в сети Интернет: поиск новых решений: труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). – М.: изд. МГУ, 2012. – С. 257–260.

14. Свирихин Д.И. Построение эффективной реализации алгоритма на основе концепции Q-детерминанта: Вып. квалиф. работа магистра: 010300.68 – Челябинск: Южно-Уральский государственный университет, 2013. – 47 с.

15. Теплов А.М. Анализ масштабируемости параллельных приложений на основе технологий суперкомпьютерного кодизайна: автореферат дис. кандидата физико-математических наук: 05.13.11 / Московский государственный университет им. М.В. Ломоносова

16. Шамов Е. А., Лукьянов В. С., Жариков Д. Н., Попов Д. С. Техно-

логии достижения параллелизма MPI, CUDA, OpenMP и моделирование динамики электронного потока в скрещенных полях с применением гибрида технологий MPI и OpenMP // Известия ВолгГТУ. 2010. №8.