минобрнауки россии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Электронные методические указания**

**к выполнению лабораторных работ по дисциплине**

**«Архитектура параллельных вычислительных систем»**

Учебно-методическое пособие

Санкт-Петербург

СПбГЭТУ «ЛЭТИ»

2019

Автор: С.В. Костичев

Электронные методические указания к выполнению лабораторных работ по дисциплине «Архитектура параллельных вычислительных систем»: учебно-методическое пособие. СПбГЭТУ «ЛЭТИ», 2019. 15 с.

Описывается цикл лабораторных работ.

Предназначено для студентов, обучающихся по направлению 09.04.01 «Информатика и вычислительная техника» по программам «Распределенные интеллектуальные системы и технологии» и «Информационное и программное обеспечение систем автоматизированного проектирования»

Одобрено

Методической комиссией факультета компьютерных технологий и

информатики

в качестве учебно-методического пособия

© СПбГЭТУ «ЛЭТИ», 2019

Оглавление

[Лабораторная работа №1. 3](#_Toc175110083)

[1.1. Предварительная подготовка к работе 3](#_Toc175110084)

[1.2. Порядок выполнения работы 3](#_Toc175110085)

[1.3. Содержание отчёта 6](#_Toc175110086)

[1.4. Контрольные вопросы 7](#_Toc175110087)

[Лабораторная работа №2. 7](#_Toc175110088)

[2.1. Предварительная подготовка к работе 7](#_Toc175110089)

[2.2. Порядок выполнения работы 8](#_Toc175110090)

[2.3. Содержание отчёта 8](#_Toc175110091)

[2.4. Контрольные вопросы 8](#_Toc175110092)

[Лабораторная работа №3. 9](#_Toc175110093)

[3.1. Предварительная подготовка к работе 9](#_Toc175110094)

[3.2. Порядок выполнения работы 9](#_Toc175110095)

[3.3. Содержание отчёта 10](#_Toc175110096)

[3.4. Контрольные вопросы 10](#_Toc175110097)

[Лабораторная работа №4. 10](#_Toc175110098)

[4.1. Предварительная подготовка к работе 11](#_Toc175110099)

[4.2. Порядок выполнения работы 11](#_Toc175110100)

[4.3. Содержание отчёта 11](#_Toc175110101)

[4.4. Контрольные вопросы 12](#_Toc175110102)

[Лабораторная работа №5. 12](#_Toc175110103)

[5.1. Предварительная подготовка к работе 12](#_Toc175110104)

[5.2. Порядок выполнения работы 12](#_Toc175110105)

[5.3. Содержание отчёта 13](#_Toc175110106)

[5.4. Контрольные вопросы 13](#_Toc175110107)

[Приложения. 14](#_Toc175110108)

[1. Последовательность действий при настройке Microsoft MPI 14](#_Toc175110109)

[2. Подключение openMP Visual Studio 2017 16](#_Toc175110110)

# Лабораторная работа №1.

**ОПЕРАЦИИ НАД ЭЛЕМЕНТАМИ ВЕКТОРОВ И МАТРИЦ НА СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ.**

Цель работы: дать знания о конструировании простых параллельных алгоритмов на системах с общей памятью; дать общее представление о масштабируемости задач; практическое освоение основных директив OpenMP и mpi; способах распределения вычислений между потоками; способах распределения вычислений итерационных циклов между потоками

## 1.1. Предварительная подготовка к работе

1. Ознакомиться с различными методами умножения матрицы на вектор, матрицы на матрицу, суммирования элементов векторов, вычисления скалярного произведения векторов.

2. Проверить, что подключены библиотеки OpenMP и mpi

## 1.2. Порядок выполнения работы

1. В зависимости от номера варианта задания разработать соответствующие алгоритмы операций над элементами векторов и матриц.

2. Написать и отладить программы на языке С++, реализующие разработанные алгоритмы последовательных и параллельных вычислений с использованием библиотек OpenMP и mpi.

3. Запустить программы для следующих значений размерности матрицы и вектора: 10, 100, 500, 1000, 5000, 10000

4. Оценить размерности матрицы и вектора, при которых эффективнее использовать алгоритмы последовательного и параллельного вычислений для разного числа потоков (по крайней мере, для меньшего, равного и большего, чем число процессоров). Под эффективностью понимается время работы программы на матрице.

Варианты заданий приведены в таблице 1.1

Табл.1.1

|  |  |  |
| --- | --- | --- |
| 1 | Умножение матрицы на вектор с использованием директивы распараллеливания параметрических циклов #pragma omp for и с использованием “ручного” задания работ (распараллеливания циклов без директивы for) | OpenMP |
| 2 | Умножение матрицы на вектор | MPI |
| 3 | Умножение матрицы на матрицу с использованием директивы распараллеливания параметрических циклов #pragma omp for и с использованием “ручного” задания работ (распараллеливания циклов без директивы for) | OpenMP |
| 4 | Умножение матрицы на матрицу | MPI |
| 5 | Скалярное произведение векторов с использованием директив critical{…} или atomic | OpenMP |
| 6 | Скалярное произведение векторов с использованием директивы reduction | OpenMP |
| 7 | Скалярное произведение векторов с использованием редуцированных операций (MPI\_Reduce(…), MPI\_Allreduce(…)) | MPI |
| 8 | Скалярное произведение векторов с использованием парных взаимодействий (MPI\_Send(…), MPI\_Recv(…)). | MPI |
| 9 | Суммирования элементов векторов | OpenMP |
| 10 | Суммирования элементов векторов с использованием редуцированных операций (MPI\_Reduce(…), MPI\_Allreduce(…)). | MPI |
| 11 | Поиск максимального значения в векторе. Использовать следующие размеры вектора: 10, 1000, 100000, 10000000, 100000000 | OpenMP |
| 12 | Трансформация вектора - применении заданной операции ко всем элементам вектора.  В качестве заданной операции над вектором возьмите следующую функцию:  х=х∙х+х/2, где х — текущий элемент вектора.  Использовать следующие размеры вектора: 10, 1000, 100000, 10000000, 100000000 | OpenMP |

**Примечание**: Отличие использования директивы распараллеливания параметрических циклов #pragma omp for и “ручного” задания работ иллюстрируется примером.

Пусть m1, m2 –квадратные матрицы размером size и число потоков NUM\_THREADS=6.

**Последовательный алгоритм умножения матриц выглядит так:**

mult\_matrix\_serial(m1, m2, size){

for (int i=0; i<size; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

return result;

}

**Распараллеливания циклов с директивой for**

#define NUM\_THREADS 6

mult\_matrix\_parallel\_for(m1, m2, size){

#pragma omp parallel for num\_threads(NUM\_THREADS)

for (int i=0; i<size; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

return result;

}

**Распараллеливания циклов без директивы for (использование директивы sections) -** “ручное” задание работ

#define NUM\_THREADS 6

mult\_matrix\_parallel\_manual(m1, m2, size){

double \*\*result = allocate\_matrix(size);

#pragma omp parallel sections

{

#pragma omp section

{

for (int i=0; i<size/6; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

}

#pragma omp section

{

for (int i=size/6; i<2\*size/6; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

}

#pragma omp section

{

for (int i=2\*size/6; i<3\*size/6; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

}

#pragma omp section

{

for (int i=3\*size/6; i<4\*size/6; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

}

#pragma omp section

{

for (int i=4\*size/6; i<5\*size/6; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

}

#pragma omp section

{

for (int i=5\*size/6; i<size; i++){

for (int j=0; j<size; j++){

result[i][j] = 0;

for (int k=0; k<size; k++){

result[i][j] += m1[i][k] \* m2[k][j];

}

}

}

}

}

return result;

}

## 

## 1.3. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы

- описание метода

- блок-схему алгоритмов с пояснениями;

- простой пример, подтверждающий правильность работы программы;

- тексты программ;  
 - примеры запуска программ;

- сравнительную оценку эффективности программ для различных значений размерности матрицы и вектора для разного числа потоков в виде таблицы

- выводы

## 1.4. Контрольные вопросы

1. Какие схемы разделения данных могут быть использованы при параллельном выполнении умножения матрицы на вектор?
2. Какие возможности технологии OpenMP были использованы при разработке параллельной программы матрично-векторного умножения?
3. Как обеспечивается правильность использования общих данных в разработанной параллельной программе?
4. Получилось ли ускорение при матрице размером 10 на 10? Почему?
5. Что понимается под параллельной программой в рамках технологии OpenMP?
6. Как определяется критическая секция?
7. Опишите модель исполнения OpenMP программы.

# Лабораторная работа №2.

**СОРТИРОВКА НА СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ**

Цель работы: изучить организацию параллельных вычислений для не вычислительных задач на системах с общей памятью.

## 2.1. Предварительная подготовка к работе

1. Ознакомиться с различными типами сортировок.
2. Проверить, что подключены библиотеки OpenMP и mpi
3. Вспомнить поразрядные логические операции в языке С++, указатели и операции над ними, структуры типа union и функции работы с файлами.

Для ввода данных для сортировки использовать файл, содержащий набор символов для сортировки или использовать генератор случайных чисел (количество чисел вводится с клавиатуры).

Отсортированные данные выводятся на экран и в файл.

## 2.2. Порядок выполнения работы

1. В зависимости от номера варианта задания разработать алгоритмы сортировки для последовательных и параллельных вычислений.

2. Написать и отладить программы на языке С++, реализующие разработанные алгоритмы последовательных и параллельных вычислений с использованием библиотек OpenMP и mpi.

3. Запустить программы для следующих значений размерности цепочек данных: 10, 100, 500, 1000, 5000, 10000

4.Оценить размерность цепочек данных, при которых эффективнее использовать алгоритмы последовательного и параллельного вычислений для разного числа потоков (по крайней мере, для меньшего, равного и большего, чем число процессоров)

Варианты заданий приведены в таблице 2.1

Табл.2.1

|  |  |  |
| --- | --- | --- |
| 1 | Сортировка методом пузырька | MPI |
| 2 | Сортировка методом Шелла | OpenMP |
| 3 | Сортировка выбором(SelectSort) | MPI |
| 4 | Сортировка простыми вставками(InsertSort) | OpenMP |
| 5 | Пирамидальная сортировка (HeapSort) | MPI |
| 6 | Быстрая сортировка (QuickSort) | OpenMP |
| 7 | Сортировка слиянием | MPI |
| 8 | Шейкер-сортировка | OpenMP |

## 2.3. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы

- описание метода

- блок-схему алгоритмов с пояснениями;

- простой пример, подтверждающий правильность работы программы;

- тексты программ;  
 - примеры запуска программ;

- сравнительную оценку эффективности программ для различных размерностей цепочек данных для разного числа потоков в виде таблицы

- выводы

## 2.4. Контрольные вопросы

1. В чем состоит постановка задачи сортировки данных?

2. Приведите несколько примеров алгоритмов сортировки? Какова вычислительная сложность

приведенных алгоритмов?

3. Какая операция является базовой для задачи сортировки данных?

4. В чем суть параллельного обобщения базовой операции задачи сортировки данных?

5. Что представляет собой алгоритм чет-нечетной перестановки?

6. В чем состоит параллельный вариант алгоритма Шелла? Какие основные отличия этого варианта параллельного алгоритма сортировки от метода чет-нечетной перестановки?

7. Что представляет собой параллельный вариант алгоритма быстрой сортировки?

8. Какой формат записи директив OpenMP?

# Лабораторная работа №3.

**РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАНЕНИЙ ИТЕРАЦИОННЫМИ МЕТОДАМИ НА СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ.**

Цель работы: практическое освоение методов решения систем линейных алгебраических уравнений (СЛАУ) итерационными методами на вычислительных системах с общей памятью.

## 3.1. Предварительная подготовка к работе

1. Ознакомиться с итерационными методами решения СЛАУ.

2. Проверить, что подключены библиотеки OpenMP и mpi

## 3.2. Порядок выполнения работы

1. В зависимости от номера варианта задания разработать алгоритмы решения СЛАУ для последовательных и параллельных вычислений.

2. Написать и отладить программы на языке С++, реализующие разработанные алгоритмы последовательных и параллельных вычислений с использованием библиотек OpenMP и mpi.

3. Запустить программы для следующих значений размерности СЛАУ: 5, 10, 100, 500, 1000, 5000, 10000

4. Оценить размерность СЛАУ, при которой эффективнее использовать алгоритмы последовательного и параллельного вычислений для разного числа потоков (по крайней мере для меньшего, равного и большего, чем число процессоров). Под эффективностью понимается время работы программы на матрице.

Варианты заданий приведены в таблице 3.1

Табл.3.1

|  |  |  |
| --- | --- | --- |
| 1 | Решение СЛАУ Ax = b методом простой итерации (Якоби) | OpenMP |
| 2 | Решение СЛАУ Ax = b методом простой итерации (Якоби) | MPI |
| 3 | Решение СЛАУ Ax = b методом Зейделя (Гаусса-Зейделя) | OpenMP |
| 4 | Решение СЛАУ Ax = b методом Зейделя (Гаусса-Зейделя) | MPI |
| 5 | Решение СЛАУ Ax = b методом верхней релаксации | OpenMP |
| 6 | Решение СЛАУ Ax = b методом верхней релаксации | MPI |

## 3.3. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы

- описание метода

- блок-схему алгоритмов с пояснениями;

- простой пример, подтверждающий правильность работы программы;

- тексты программ;  
 - примеры запуска программ;

- сравнительную оценку эффективности программ для различных размерностей СЛАУ для разного числа потоков в виде таблицы

- выводы

## 3.4. Контрольные вопросы

1. Что представляет собой система линейных уравнений? Какие типы систем вам известны? Какие методы могут быть использованы для решения систем разных типов?
2. В чем состоит постановка задачи решения системы линейных уравнений?
3. В чем состоит идея параллельной реализации метода верхней релаксации?
4. В чем отличие прямых и итерационных методов решения СЛАУ?
5. Какие компьютерные платформы относятся к числу вычислительных систем с общей памятью?
6. Могут ли прямые и итерационные методы использоваться совместно?

# Лабораторная работа №4.

**РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАНЕНИЙ ПРЯМЫМИ МЕТОДАМИ НА СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ.**

Цель работы: практическое освоение методов решения систем линейных алгебраических уравнений прямыми методами на вычислительных системах с общей памятью.

## 4.1. Предварительная подготовка к работе

1. Ознакомиться с прямыми методами решения СЛАУ.

2. Проверить, что подключены библиотеки OpenMP и mpi

## 4.2. Порядок выполнения работы

1. В зависимости от номера варианта задания разработать алгоритмы решения СЛАУ для последовательных и параллельных вычислений.

2. Написать и отладить программы на языке С++, реализующие разработанные алгоритмы последовательных и параллельных вычислений с использованием библиотек OpenMP и mpi.

3. Запустить программы для следующих значений размерности СЛАУ: 5, 10, 100, 500, 1000, 5000, 10000

4. Оценить размерность СЛАУ, при которой эффективнее использовать алгоритмы последовательного и параллельного вычислений для разного числа потоков (по крайней мере для меньшего, равного и большего, чем число процессоров). Под эффективностью понимается время работы программы на матрице.

5. Недостаток прямых методов заключается в том, что погрешность накапливается в процессе вычисления. Оцените погрешность вычислений при различных размерностях СЛАУ.

Варианты заданий приведены в таблице 4.1

Табл.4.1

|  |  |  |
| --- | --- | --- |
| 1 | Решение СЛАУ Ax = b простым методом Гаусса. | MPI |
| 2 | Решение СЛАУ Ax = b простым методом Гаусса. | OpenMP |
| 3 | Решение СЛАУ Ax = b методом Гаусса-Жордана | MPI |
| 4 | Решение СЛАУ Ax = b методом Гаусса-Жордана | OpenMP |
| 5 | Решение СЛАУ Ax = b методом Крамера | MPI |
| 6 | Решение СЛАУ Ax = b методом Крамера | OpenMP |
| 7 | Решение СЛАУ Ax = b матричным методом | MPI |
| 8 | Решение СЛАУ Ax = b матричным методом | OpenMP |

Примечание: Используйте директивы параллельных вычислений при **триангуляции матрицы (приведении матрицы к треугольному виду), если она (триангуляция) является частью алгоритма.**

## 4.3. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы

- описание метода

- блок-схему алгоритмов с пояснениями;

- простой пример, подтверждающий правильность работы программы;

- тексты программ;  
 - примеры запуска программ;

- сравнительную оценку эффективности программ для различных размерностей СЛАУ для разного числа потоков в виде таблицы

- выводы

## 4.4. Контрольные вопросы

1. В чем идея параллельного обобщения метода Гаусса?
2. Какие показатели эффективности для параллельного варианта метода Гаусса?
3. В чем состоит схема программной реализации параллельного варианта метода Гаусса?
4. В чем отличие прямых и итерационных методов решения СЛАУ?
5. В чем состоят основы технологии OpenMP?
6. Могут ли прямые и итерационные методы использоваться совместно?

# 

# Лабораторная работа №5.

**РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАНЕНИЙ**

**С МАТРИЦАМИ СПЕЦИАЛЬНОГО ВИДА НА СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ.**

Цель работы: практическое освоение методов решения СЛАУ с матрицами специального вида на вычислительных системах с общей памятью.

## 5.1. Предварительная подготовка к работе

1. Ознакомиться с различными методами решения СЛАУ.

2. Проверить, что подключены библиотеки OpenMP и mpi

## 5.2. Порядок выполнения работы

1. В зависимости от номера варианта задания разработать алгоритмы решения СЛАУ для последовательных и параллельных вычислений.

2. Написать и отладить программы на языке С++, реализующие разработанные алгоритмы последовательных и параллельных вычислений с использованием библиотек OpenMP и mpi.

3. Запустить программы для следующих значений размерности СЛАУ: 5, 10, 100, 500, 1000, 5000, 10000

4. Оценить размерность СЛАУ, при которой эффективнее использовать алгоритмы последовательного и параллельного вычислений для разного числа потоков (по крайней мере для меньшего, равного и большего, чем число процессоров). Под эффективностью понимается время работы программы на матрице.

Варианты заданий приведены в таблице 5.1

Табл.5.1

|  |  |  |
| --- | --- | --- |
| 1 | Решение СЛАУ Ax = b методом прогонки (параллельный алгоритм не эффективен) | OpenMP |
| 2 | Решение СЛАУ Ax = b методом прогонки(параллельный алгоритм не эффективен) | MPI |
| 3 | Решение СЛАУ Ax = b методом квадратного корня (разложение Холецкого) | OpenMP |
| 4 | Решение СЛАУ Ax = b методом квадратного корня (разложение Холецкого) | MPI |
| 5 | Решение СЛАУ Ax = b методом LU- разложения | OpenMP |
| 6 | Решение СЛАУ Ax = b методом LU- разложения | MPI |

**Примечание**: фраза «параллельный алгоритм не эффективен» для метода прогонки означает, что при указанных значениях размерности СЛАУ использование параллельного алгоритма для разного числа потоков может не привести к повышению эффективности. Если позволяют аппаратные ресурсы, попробуйте запустить программу для значений размерности СЛАУ больше чем 200000.

## 5.3. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы

- описание метода

- блок-схему алгоритмов с пояснениями;

- простой пример, подтверждающий правильность работы программы;

- тексты программ;  
 - примеры запуска программ;

- сравнительную оценку эффективности программ для различных размерностей СЛАУ для разного числа потоков в виде таблицы

- выводы

## 5.4. Контрольные вопросы

1. Что представляет собой система линейных уравнений? Какие типы систем вам известны? Какие методы могут быть использованы для решения систем разных типов?
2. В чем состоит постановка задачи решения системы линейных уравнений?
3. В чем отличие прямых и итерационных методов решения СЛАУ?
4. В чем состоят основные преимущества технологии OpenMP?
5. Почему надо распараллеливать программу?

# Приложения.

## 1. Последовательность действий при настройке Microsoft MPI

Источник: статья «Подключение MPI в Visual Studio 2015» (<https://pro-prof.com/forums/topic/подключение-mpi-в-visual-studio> )

1.1.Скачать и установить:

* + Microsoft Visual Studio 2019,
  + Microsoft HPC Pack 2012 Client Utilities Redistributable Package (***HpcClient\_x64.Msi***) (эта программа должна быть установлена перед sdk\_x64.msi)
  + Microsoft HPC Pack 2012 SDK SP1 (***sdk\_x64.msi***) , (устанавливается в папку *C:\Program Files\Microsoft HPC Pack 2012 SDK\)*
  + Microsoft MPI (***msmpisetup.exe*** – добавляет dll). -взят из Microsoft MPI v.10.0 (устанавливается в папку *C:\Program Files\Microsoft MPI\)*
  + Microsoft SDKs (***msmpisdk.msi*** – устанавливает MPI SDK) , -взят из Microsoft MPI v.10.0 (устанавливается в папку *C:\Program Files (x86)\Microsoft SDKs\MPI\)*

1.2 В VS 2019 создать консольный проект

1.3 Проект – Свойства – Свойства конфигурации – Каталоги VC++

Включаемые каталоги (Include Directories):

- *C:\Program Files (x86)\Microsoft SDKs\MPI\Include (*путь к файлу mpi.h )

- *C:\Program Files\Microsoft HPC Pack 2012 SDK\Include*

Каталоги библиотек:

*C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x64* (путь к файлу msmpi.lib)

*C:\Program Files\Microsoft HPC Pack 2012 SDK\Lib\amd64*

1.4 Проект – Свойства – Свойства конфигурации – Компановщик (Linker) – Ввод (Input) Дополнительные зависимости (Additional Dependencies) : *msmpi.lib*

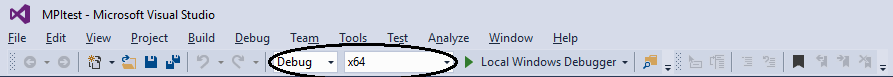
1.5 Проект – Свойства – Свойства конфигурации – Отладка (Debugging) (загружаемый отладчик: локальный отладчик Windows)

Команда: <обзор>

*C:\Program Files\Microsoft MPI\Bin\* *mpiexec.exe* (путь к файлу mpiexec.exe)

Аргументы команды (Command Arguments) указать, например

-*n 4 $(TargetPath)*

1.6 перед запуском программы не забудьте указать её разрядность (установить платформу x64):  


В противном случае повляются ошибки и предупреждение типа

ConsoleApplication1.obj : **error LNK2019**: ссылка на неразрешенный внешний символ \_MPI\_Comm\_size@8 в функции \_main

C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x64\msmpi.lib **: warning LNK4272**: тип компьютера библиотеки "x64" конфликтует с типом целевого компьютера "x86"

Иногда против последней ошибки помогает следующее [1] (смотри ниже)

* 1. Пример запуска программы с MPI:

*#include<mpi.h>*

*#include<iostream>*

*using namespace std;*

*int main(int argc, char \*\*argv)*

*{*

*int rank, size;*

*MPI\_Init(&argc, &argv);*

*MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);*

*MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);*

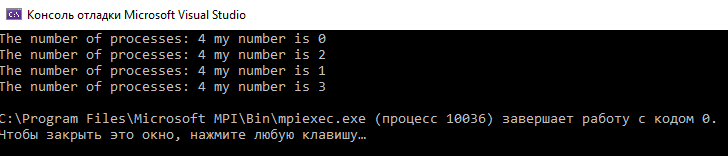
*cout << "The number of processes: " << size << " my number is " << rank << endl;*

*MPI\_Finalize();*

*return 0;*

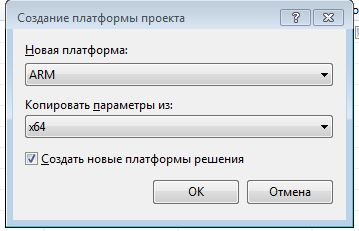
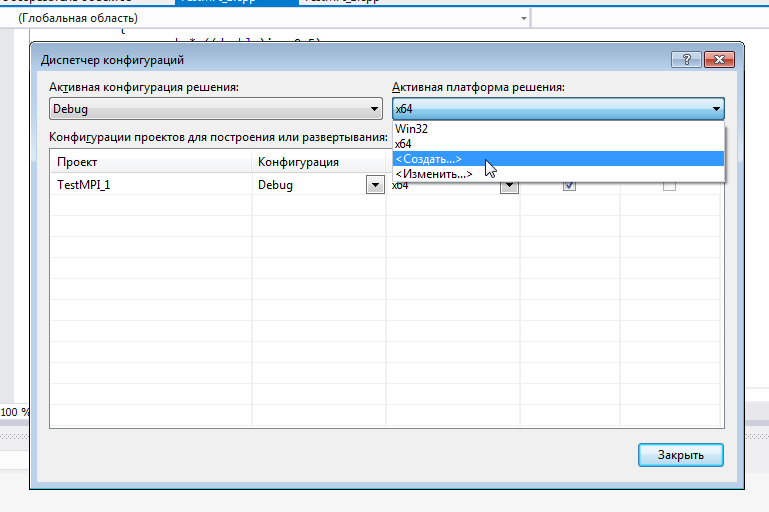
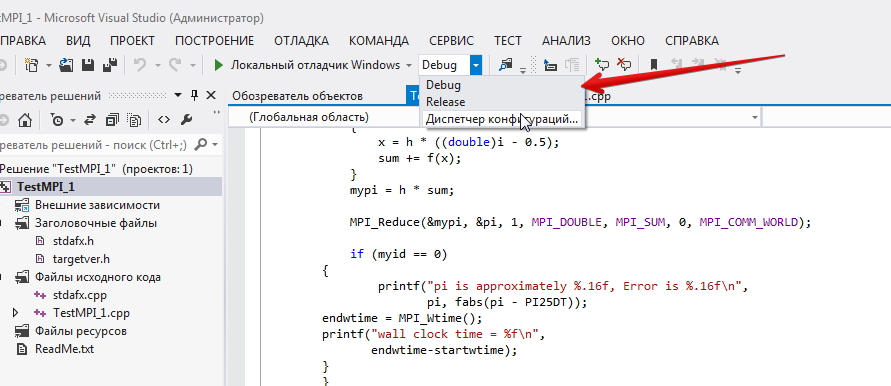
*}*

Работа программы на 4-х узлах



[1] <http://www.cyberforum.ru/cpp-beginners/thread792097.html>

mpich error LNK2019: ссылка на неразрешенный внешний символ



## 2. Подключение openMP Visual Studio 2017

2.1 Установить в свойствах проекта

* свойства проекта –> свойства конфигурации –>C/ С++ –> язык –> Поддержка openMP - "Да"
* свойства проекта –> свойства конфигурации –>C/ С++ –> командная строка –>Изменить поле Дополнительные параметры, добавив туда /Zc:twoPhase- и нажать OK (если этого не сделать, то при компиляции появляется ошибка c1xx: error C2338: двухэтапный поиск имен не поддерживается для C++/CLI, C++/CX или OpenMP. Используйте /Zc:twoPhase- )

См. <https://docs.microsoft.com/ru-ru/cpp/build/reference/zc-twophase?view=vs-2017>

/Zc:twoPhase-(отключение двухэтапного поиска имени)

Используйте /Zc:twoPhase- Если код требует старое поведение для правильной компиляции. Настоятельно рекомендуется обновлять код, чтобы соответствовать стандарту.

2.2 Подключить библиотеку #include <omp.h>

Visual Studio 2017 поддерживает только OpenMP2

Учебно-методическое пособие

Костичев Сергей Валентинович

Электронные методические указания к выполнению лабораторных работ по дисциплине «Архитектура параллельных вычислительных систем»

Издание публикуется в авторской редакции

СПбГЭТУ «ЛЭТИ»

197376, Санкт*-*Петербург, ул. Проф. Попова, 5