ОГЛАВЛЕНИЕ

[1 Использование среды разработки eclipse 2](#_Toc422429055)

[2 Систолическое матричное умножениЕ 11](#_Toc422429056)

# 1 ИСПОЛЬЗОВАНИЕ СРЕДЫ РАЗРАБОТКИ ECLIPSE

Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation. Гибкость Eclipse обеспечивается за счёт подключаемых модулей, благодаря чему возможна разработка не только на Java, но и на других языках, таких, как C/C++, Perl, Groovy, Ruby, Python, PHP, Erlang, Компонентного Паскаля, Zonnon, и прочих.

Платформа параллельных инструментов (PTP) обеспечивает портируемую, масштабируемую, основанную на стандартах платформу параллельных инструментов, которая позволит облегчить интеграцию инструментов, специфических для параллельных компьютерных архитектур.

В данной работе используется Eclipse PTP Luna. Интерфейс данной программы представлен на рисунке 1.

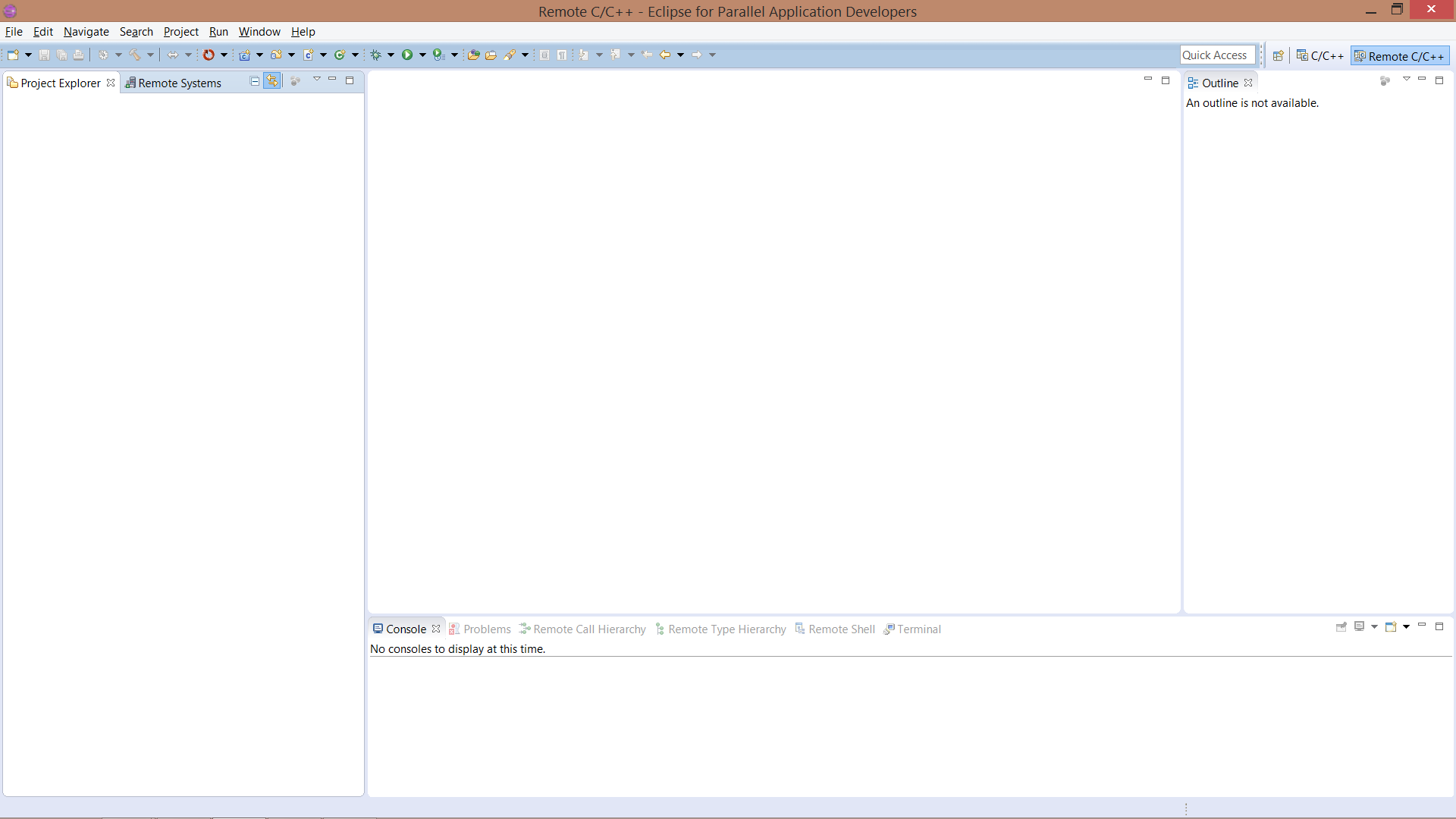


Рисунок 1 – Интерфейс Eclipse PTP Luna

Создадим новый синхронизируемый проект (File->New->Other…), как показано на рисунке 2.

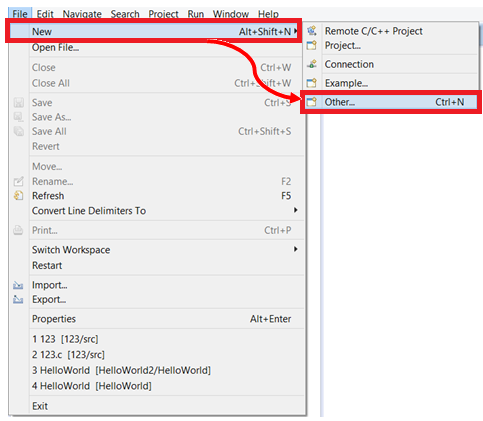


Рисунок 2 – Создание нового проекта

В появившемся окне необходимо выбрать тип проекта. В нашем случае выберем Synchronized C/C++ Project, как показано на рисунке 3.

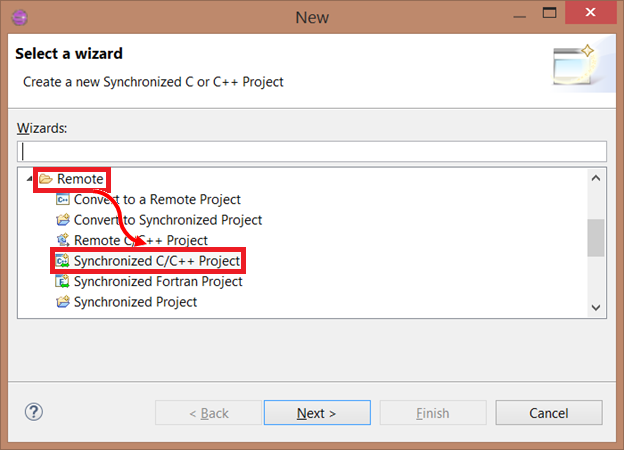


Рисунок 3 –Выбор типа проекта

В появившемся окне необходимо ввести название бедующего проекта в графе Project name, а также в случае необходимости создать новое подключение нажав на кнопку Remove directory -> New, как показано на рисунке 4.

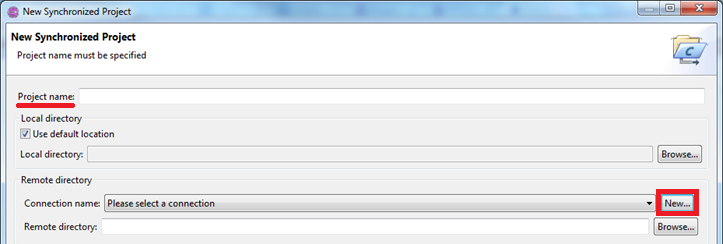


Рисунок 4 – Создание нового проекта

В появившемся окне необходимо ввести имя соединения (Connection name), адрес сервера (Host), а также имя пользователя (User) и пароль (Password), как показано на рисунке 5.

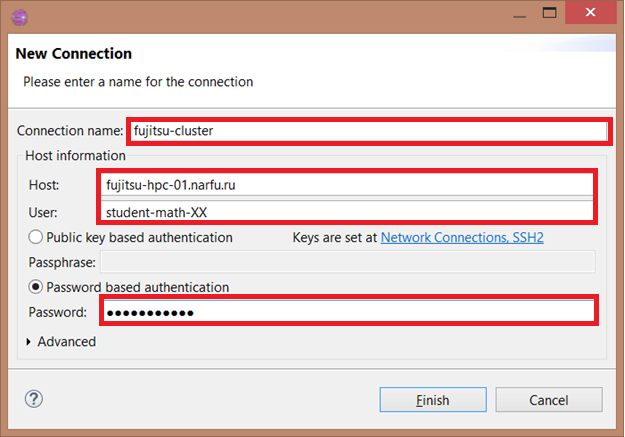


Рисунок 5 – Создание нового подключения

После создания подключения необходимо выбрать тип проекта и компилятор, как показано на рисунке 6.

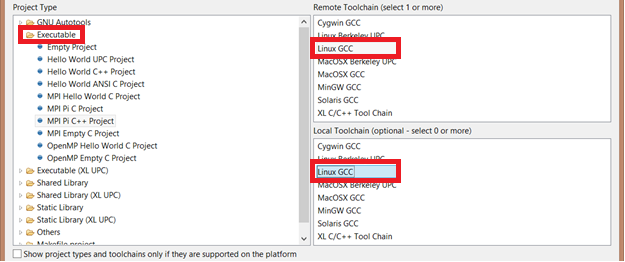


Рисунок 6 – Выбор типа проекта и компилятора

В дальнейшем следуя инструкциям рисунка 7 завершите формирование проекта.

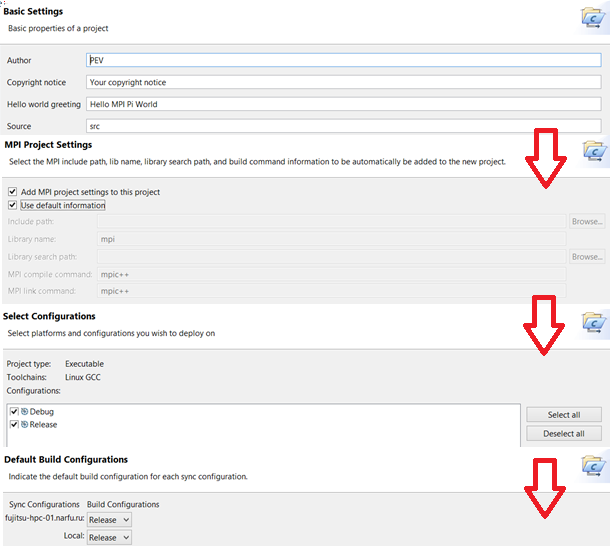


Рисунок 7 – Формирование проекта

После создания проекта его необходимо настроить синхронизацию как показано на рисунке 8.

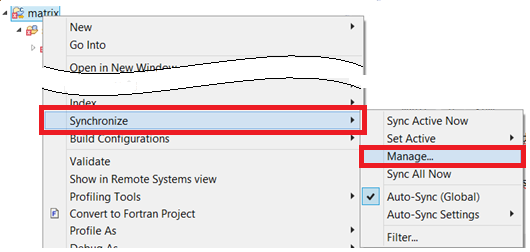


Рисунок 8 – Настройка синхронизации проекта

В появившемся окне необходимо добавить модуль openmpi-pbs, как показано на рисунке 9.

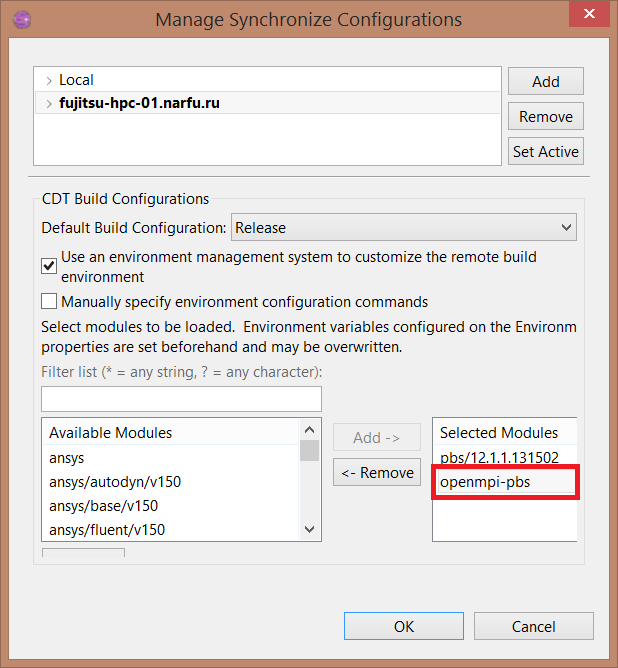


Рисунок 9 – Добавление модуля

После этого необходимо скомпилировать задачу в режиме Release, как показано на рисунке 10

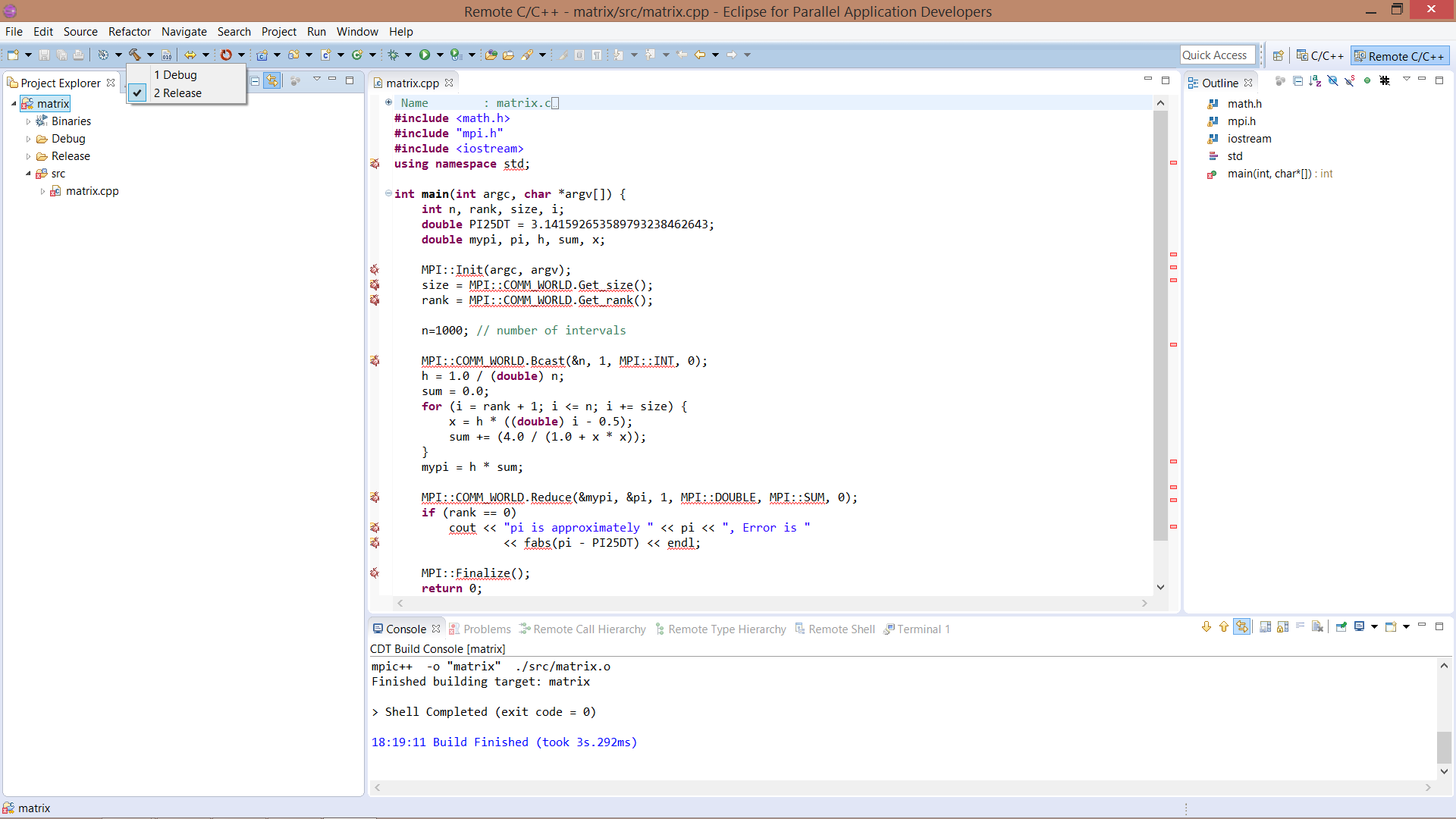


Рисунок 10 – Компиляция приложения

Необходимо выбрать конфигурации запуска проекта, как показано на рисунке 11

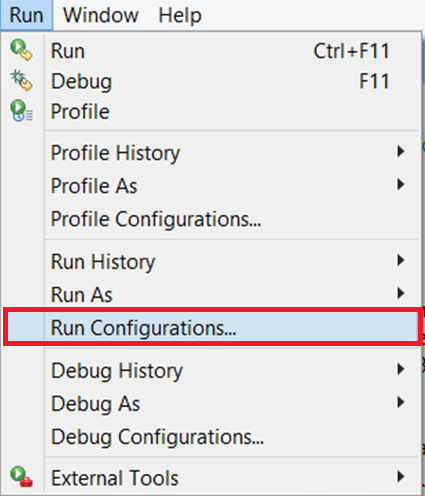


Рисунок 11 – Выбор конфигурации запуска

В меню Parallel Application -> Resources -> Target System Configuration необходимо выбрать Generic PBS Batch, как показано на рисунке 12.

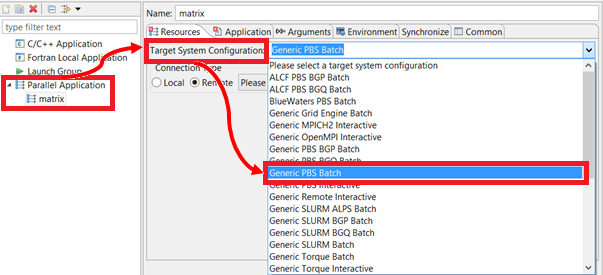


Рисунок 12 –Настройка конфигураций запуска

В появившемся окне выберите созданное ранее подключение, впишите имя пользователя, номер доступных узлов и число процессов, как показано на рисунке 13. В графе 1 необходимо указать адрес вычислителя, на который будут посылаться задачи. В графе 2 необходимо указать аккаут из под которого будут выполняться задачи. В графе 3 необходимо указать количество доступных усзов в виде 2:ppn=20 (2 – количество узлов, 20 – количество доступных процессоров на 1 узле (в САФУ максимум 20)). В графе 4 необходимо указать сколько MPI процессов необходимо запустить на вычислителе. Для продолжения настройки нажмите на кнопку в графе 5 Configure.

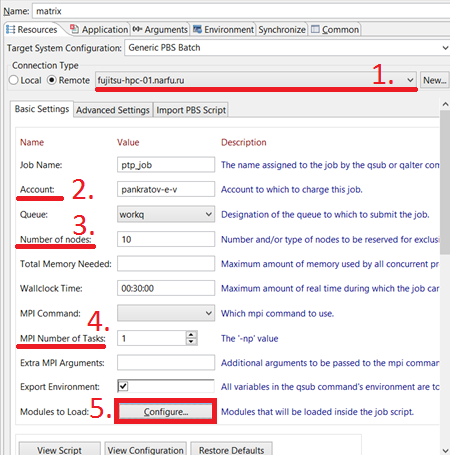


Рисунок 13 –Настройка конфигураций запуска

После этого в появившемся окне необходимо добавить модуль модуль openmpi-pbs, как показано на рисунке 14.

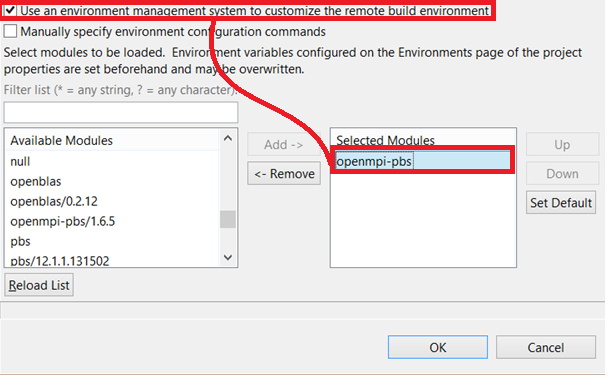


Рисунок 14 – Добавление модуля

На вкладке Application необходимо выбрать скомпилированный файл. Как показано на рисунке 15

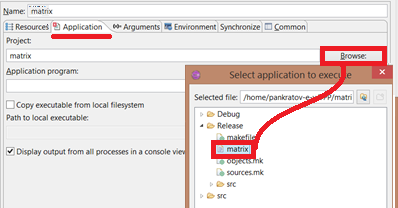


Рисунок 15 – Выбор файла

После этого необходимо запустить расчет нажав кнопку Run. При запуске расчета должно появится окно как на рисунке 16.

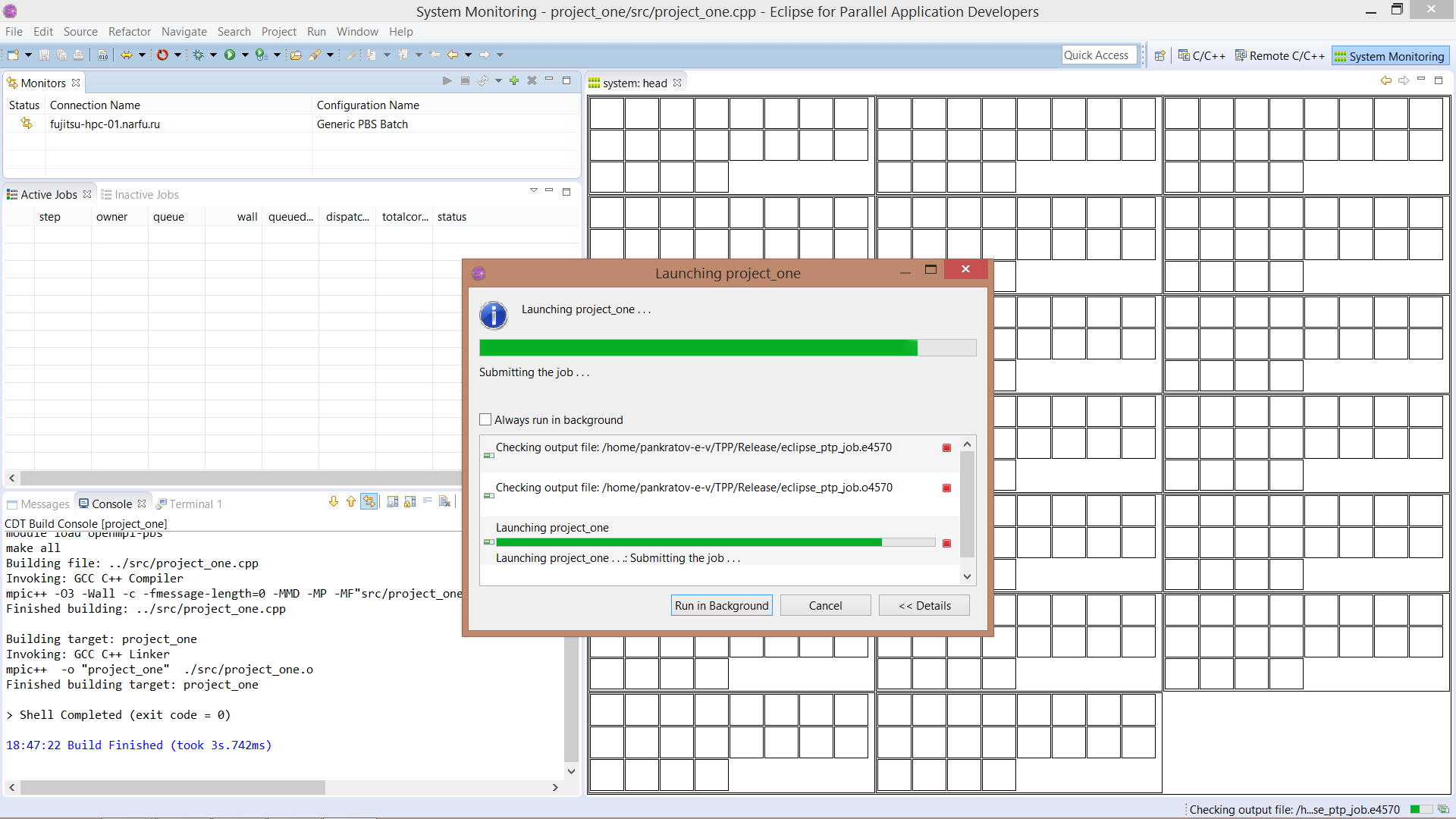


Рисунок 16 – Окно запуска расчета

Для мониторинга загрузки кластера следует воспользоваться System monitoring, как показано на рисунке 17

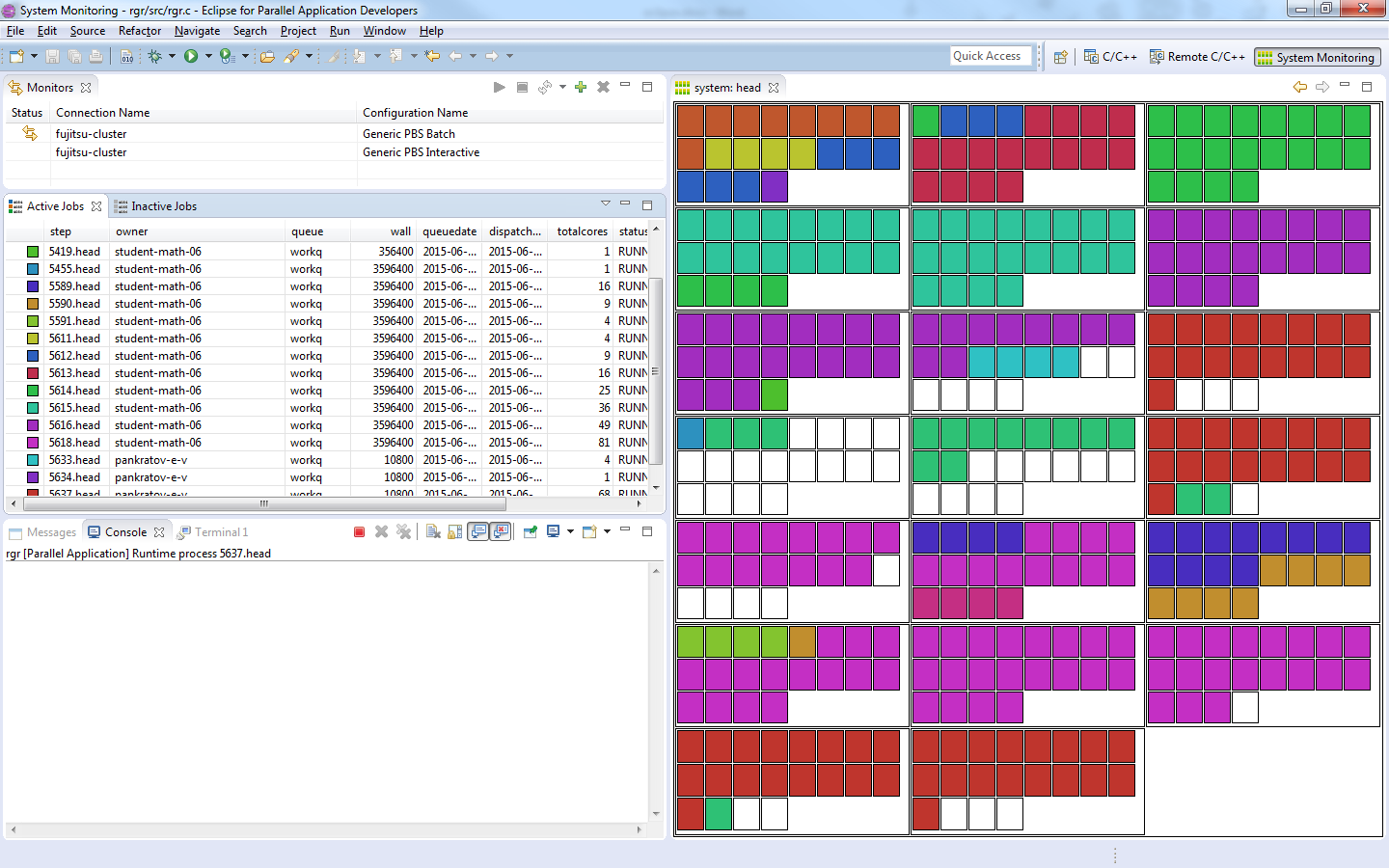


Рисунок 17 – Окно запуска расчета

Для проверки выходного файла необходимо нажать на кнопку Get Job Output как показано на рисунке 18.

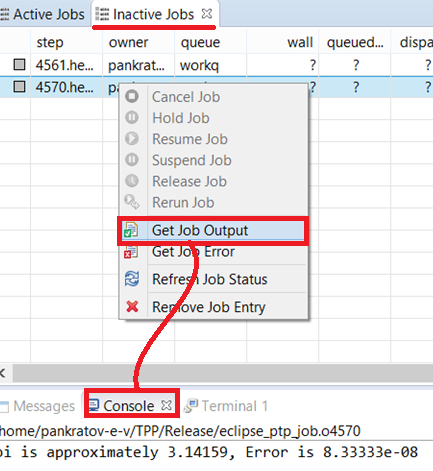


Рисунок 18 – Вывод результата

# 2 СИСТОЛИЧЕСКОЕ МАТРИЧНОЕ УМНОЖЕНИЕ

Матричное умножение (МУ) - это базовая операция линейной алгебры и доминирующая вычислительная часть многих научных приложений, таких как решение СЛАУ, интегрирование систем дифференциальных уравнений как обыкновенных, так и в частных производных. Существует множество традиционных параллельных алгоритмов вычисления матричного произведения для плотных матриц. Практически все алгоритмы или их варианты имеют приблизительно линейное ускорение для больших размерностей матриц, и не существует алгоритма, который был бы существенно лучше других. В этой работе анализируется эффективность систолического перемножения квадратных матриц.

Систолический алгоритм умножения матриц является наиболее эффективным для параллельных компьютеров SIMD архитектуры. Рассмотрим модификацию этого алгоритма с целью использования его на кластерных и MIMD системах. В качестве модели высокопроизводительной параллельной вычислительной системы будем использовать мультикомпьютер с распределенной памятью и топологией 2D-тор.

Вычислительная схема алгоритма основана на блочном разбиении матриц, и использует как преимущества последовательного умножения матриц, так и систолического алгоритма. Этап предварительной подготовки состоит из разбиения перемножаемых матриц на блоки и отображения блоков на виртуальную топологию тор. Далее над блоками выполняются в точности те же действия, которые выполняются систолическим алгоритмом над элементами матриц, причем сложение и умножение матричных блоков выполняются последовательно одним процессором.

Пусть исходные матрицы A, В имеют размерность m\*m, и разбиваются на квадратные блоки порядка k=[m/p]. Для простоты рассуждений пусть и . Тогда, , где - это размер ширины блока, а - количество блоков.

Операцию умножения матриц A и B в блочном виде можно представить следующим образом:

где каждый блок матрицы определяется в соответствии с выражением:. На первом этапе алгоритма блочного систолического умножения матриц каждый процессор решетки с номером содержит два блока матриц исходных данных: и . В процессе выполнения алгоритма процессор отвечает за вычисление блока матрицы результата

Далее блоки матрицы косо сдвигаются влево по строкам:

|  |  |  |
| --- | --- | --- |
|  |  |  |

а блоки матрицы В косо вверх по столбцам:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Затем, над блоками выполняются в точности те же действия, которые выполняются систолическим алгоритмом над элементами матриц, причем сложение и умножение матричных блоков выполняются последовательно одним процессором. То есть на каждом из шагов выполняется умножение блоков матриц и на каждом процессоре по простому последовательному стандартному алгоритму. Затем производится одиночный сдвиг влево для блоков матрицы и вверх для блоков матрицы , и описанные действия повторяются. В результате на каждом процессоре с номером получается блок матрицы результата . Вычислительная схема описанного алгоритма для мультикомпьютера из процессоров приведена на рисунке 1.

Пусть время выполнения одной операции умножения/сложения чисел составляет . Поскольку для большинства RISC архитектур справедливо соотношение:, вводится понятие флопа, как произвольной операции с плавающей точкой, и далее все временные характеристики приводятся к флопу.

Время реализации арифметических операций для блочного систолического алгоритма составляет:

|  |  |  |
| --- | --- | --- |
|  |  |  |

где – время, необходимое для реализации операций умножения/сложения блоков матриц размерности: .

В свою очередь, время вычисления произведения блоков матриц равно:

|  |  |  |
| --- | --- | --- |
|  |  |  |

где - количество элементов блока, - время, необходимое для вычисления одного элемента блока.

Время на реализацию сложения блоков матриц составляет:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Таким образом, общее время вычислений блочного систолического алгоритма умножения матриц составляет:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Время на обменные операции по блочному систолическому алгоритму равно:

|  |  |  |
| --- | --- | --- |
|  |  |  |

где - Длительность подготовки сообщения для передачи; - Время передачи одного слова; - Объем передаваемых данных в словах.

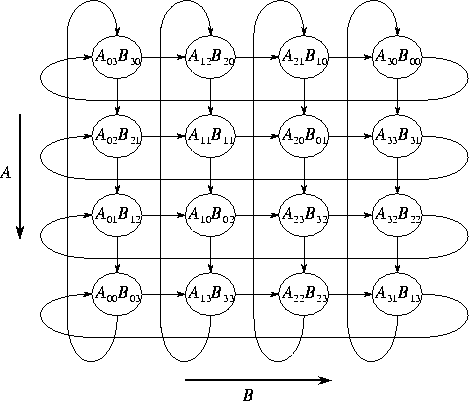


Рисунок 19 – Вычислительная схема блочного систолического умножения квадратных матриц

Динамическими характеристиками, определяющими качество полученного параллельного алгоритма, являются коэффициенты ускорения и эффективности:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Заметим, что везде речь идет о квадратных матрицах, так как интегрирование линейных СОДУ экспоненциальным методом предполагает работу с квадратными матрицами. В произвольном случае при наличии прямоугольных матриц преобразование в квадратные происходит окаймлением нулевыми элементами по строкам или по столбцам