**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

Лабораторная работа №2

**по дисциплине**

**«Распределенные программные системы и технологии»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3308 |  | Сысоев В.Б. |
| Преподаватель |  | Пазников А.А. |

Санкт-Петербург

2018

# 1. Задание

1. Разработать MPI-программу, реализующую параллельную версию игры «Жизнь».
2. Проанализировать эффективность параллельной программы: построить графики зависимости ускорения и коэффициента эффективности от числа потоков, оценить масштабируемость, построить асимптотические оценки вычислительной и коммуникационной сложности параллельного алгоритма.

**Замечания:**

* Начальная конфигурация игры генерируется случайным образом на нулевом процессе. Размер игрового поля необходимо подобрать таким образом, чтобы минимальное время выполнения программы было не менее 10 секунд.
* При взаимодействии процессов в MPI-программе требуется использовать виртуальную декартовую топологию вида «двумерная решётка».
* Реализовать поддержку периодического и непериодического игрового поля.
* Состояние игрового поля необходимо сохранять в файл через заданное количество шагов игры.

# 2. Описание решения задачи

В основе данной реализации игры «Жизнь» лежит простейший алгоритм, подсчитывающий у каждой клетки поля количество соседей и принимающий на основании этих данных решение о том будет ли данная клетка «жить» или нет на следующем шаге.

Для реализации параллельного алгоритма, игровое поле разбивается на несколько одинаковых прямоугольных областей. Каждый процесс обрабатывает одну выделенную ему область. Перед каждым шагом игры процессы обмениваются состоянием клеток на границах своих областей. Для получения информации о ранках соседних процессов используется виртуальная двумерная топология. Игра может происходить на периодическом и непериодическом игровом поле, для этого двумерная топология конфигурируется либо с цикличностью, либо нет соответственно.

В приложении предусмотрен процесс сбора данных о состоянии игрового поля главным процессом и запись этой информации в файл через конфигурируемое количество шагов игры. Для просмотра сохраненного в файл состояния игрового поля создана дополнительная утилита, представляющая данные в удобной графической форме. Внешний вид данной утилиты представлен на рисунке 1.

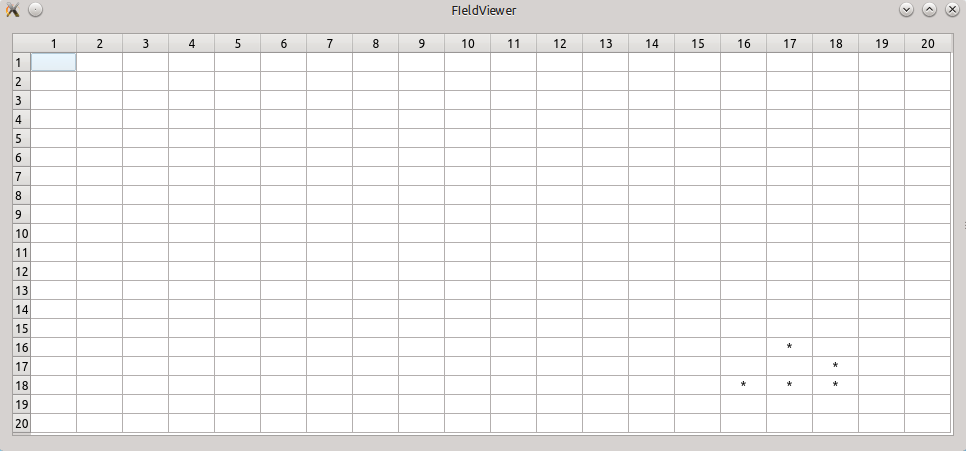


Рисунок 1. Утилита просмотра сохраненного состояния игрового поля

Описанный алгоритм будет иметь квадратичною вычислительную и линейную коммуникационную сложности, в зависимости от размера стороны игрового поля.

# 3. Организация экспериментов

Эксперименты проводились на персональном компьютере с процессором Intel Core i7 4700HQ. Для экспериментов использовалась виртуальная машина, которой было выделено 3 Гб оперативной памяти и все 4 ядра процессора. Основной ОС ПК являлась Windows 10, ОС на виртуальной машине являлась Kubuntu 14.04 64-bit. Для виртуализации использовалось ПО VMware Workstation 12.

Для компиляции использовался GCC версии 4.8.4, библиотека mpich версии 3.0.4-6ubuntu1. Основными ключами компиляции были -O3 -DNDEBUG и специфичные для mpich.

Для проведения экспериментов было взято квадратное поле со стороной 4200 клеток. Каждый запуск программы включал в себя 10 итераций по 10 шагов игры, в конце каждой итерации состояние игрового поля собиралось на 0-м процессе, однако запись в файл не производилась. Результаты экспериментов представлены ниже.

# 4. Результаты экспериментов

На рисунке 2 представлен график ускорения в зависимости от количества процессов для описанного ранее игрового поля.

Рисунок 2. График ускорения.

На рисунке 3 представлен график эффективности в зависимости от количества процессов для описанного ранее игрового поля.

Рисунок 3. График эффективности.

Для выявления равномерности загрузки, в приложении предусмотрен подсчет времени выполнения каждого из процессов. Результаты подсчета, для 4 процессов представлены ниже:

Starting /usr/bin/mpirun...

Main process 0/4 execution time: 18.887559

Process 1/4 execution time: 19.184901

Process 2/4 execution time: 19.184902

Process 3/4 execution time: 19.184905

/usr/bin/mpirun exited with code 0

Как видно из представленных результатов, дисбаланса загрузки не наблюдается.

# 5. Анализ результатов экспериментов и выводы

Итоги экспериментов говорят о том, что данная задача хорошо поддается распараллеливанию, а реализованный алгоритм обеспечивает сбалансированную загрузку процессорных ядер и неплохо масштабируется. Падение ускорения и снижение эффективности с ростом числа процессов объясняется линейно растущей коммуникационной сложностью и простоями процессов, связанными с необходимостью синхронизации после каждого шага игры.

# 6. Список использованных литературных источников

1. Wikipedia, Игра «Жизнь» [Электронный ресурс]. URL:  https://ru.wikipedia.org/wiki/Жизнь\_(игра) (дата обращения: 15.04.2018).
2. Пазников А.А. Параллельные вычислительные технологии: Практикум. – Новосибирск: Печатный Двор, 2016. – 59 с.