УДК 004.272

РАЗРАБОТКА УСКОРЕННОГО АЛГОРИТМА ТРАССИРОВКИ ЛУЧЕЙ

Сереженко А.А., Мохаммад Юнис, Мальчева Р.В.
Донецкий национальный технический университет
E-mail: alexserezhenko@gmail.com

2

В данной статье рассматривается разработка ускоренного алгоритма трассировки лучей для архитектуры SIMD, результаты тестирования.

Общая постановка проблемы

Существует множество алгоритмов синтеза изображения. Все они делятся на два основных вида – алгоритмы, использующие метод растеризации при построении изображения, и алгоритмы, использующие метод трассировки лучей. Первая группа алгоритмов работает быстро, но имеет проблемы с синтезом фотореалистичных изображений. Группа алгоритмов, использующая технологию трассировки лучей, отличается от первой группы качеством синтезируемого изображения – оно высокофотореалистично. Но у этой группы алгоритмов основной проблемой является скорость построения изображения.

Постановка задач исследования

На текущем этапе задача состоит в реализации параллельного алгоритма трассировки лучей для xIMD архитектуры, выполнении его на многопроцессорной станции.

Анализ ускоренного алгоритма трассировки лучей, выбор архитектуры

MIMD (англ. Multiple Instruction stream, Multiple Data stream) — концепция архитектуры компьютера, используемая для достижения

параллелизма вычислений. Машины на многопроцессорных имеют несколько процессоров, которые функционируют асинхронно и независимо. В любой момент различные процессоры могут выполнять различные команды над различными частей данных. MIMD-архитектуры могут быть использованы в целом ряде областей, таких как системы автоматизированного проектирования / автоматизированное производство, моделирование, а также комуникатор связей (communication switches). MIMD машины могут быть либо с общей памятью, либо с распределяемой памятью. Эта классификация основана на том как организован MIMD-процессоров получают доступ к памяти [1].

SIMD (англ. single instruction, multiple data) — принцип компьютерных вычислений, позволяющий обеспечить параллелизм на уровне данных [2]. SIMD-компьютеры состоят из одного командного процессора (управляющего модуля), называемого контроллером, и нескольких модулей обработки данных, называемых процессорными элементами. Управляющий модуль принимает, анализирует и выполняет команды. Если в команде встречаются данные, контроллер рассылает на все процессорные элементы команду, и эта команда выполняется на нескольких или на всех процессорных элементах. Каждый процессорный элемент имеет свою собственную память для хранения данных. Одним из преимуществ данной архитектуры считается то, что в этом случае более эффективно реализована логика вычислений. До половины логических инструкций обычного процессора связано с управлением выполнением машинных команд, а остальная их часть относится к работе с внутренней памятью процессора и выполнению арифметических операций. В SIMD компьютере управление выполняется контроллером, а «арифметика» отдана процессорным элементам [2].

Алгоритм трассировки лучей относится к последовательным алгоритмам с N-повторяющимися действиями. Алгоритм работает с пикселями камеры, направленной в сцену. Это значит, что для каждого пикселя камеры выполняется набор одних и тех же

операций по формированию луча, поиска ближайшего объекта на пути следования луча, расчет преломлений, отражений, поиск источников света, в которые первоначальный луч попадает для вычисления теней.

Как видно из краткого описания алгоритма, алгоритм полностью повторяет концепцию Single Instructure – Multi Data – SIMD.

Для реализации алгоритма трассировки лучей был использован программный пакет MPI.NET. MPI.NET – высокопроизводительная, легкая в использовании реализация технологии Message Passing Interface в среде Microsoft .NET.

MPI является стандартом для программной реализации параллельных алгоритмов для систем, использующих распределенную память (вычислительные кластеры). MPI.NET обеспечивает поддержку всех языков .NET платформы. Библиотека включает в себя значительно расширенный функционал возможностей(например автоматическую сериализацию объектов), который позволяет гораздо легче строить «параллельные» программы для выполнения на вычислительных кластерах [3].

Построение SIMD-алгоритма трассировки лучей

SIMD алгоритм построен по классической схеме – мастер процесс раздает задания остальным процессам (Slave), и в последствии собирает результаты вычислений.

Мастер процесс в зависимости от количества Slave – процессов вычисляет какие строки будет обрабатывать каждый Slave процесс, рассылает эти данные и начинает ожидание приема результатов вычислений [4].

Схема SIMD алгоритма приведена на рис. 1.

Схема выполнения и взаимосвязей процессов SIMD алгоритма изображена на рис. 2.

В нижеприведенном фрагменте кода приведено деление данных между Slave-процессами и их рассылка.

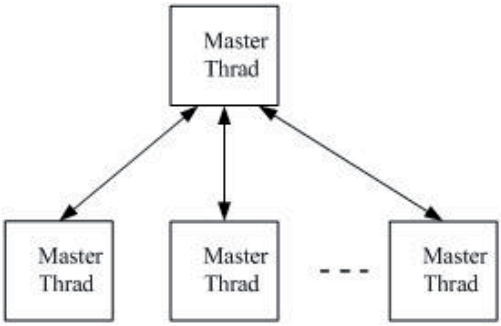

Рисунок 1 - Схема SIMD алгоритма

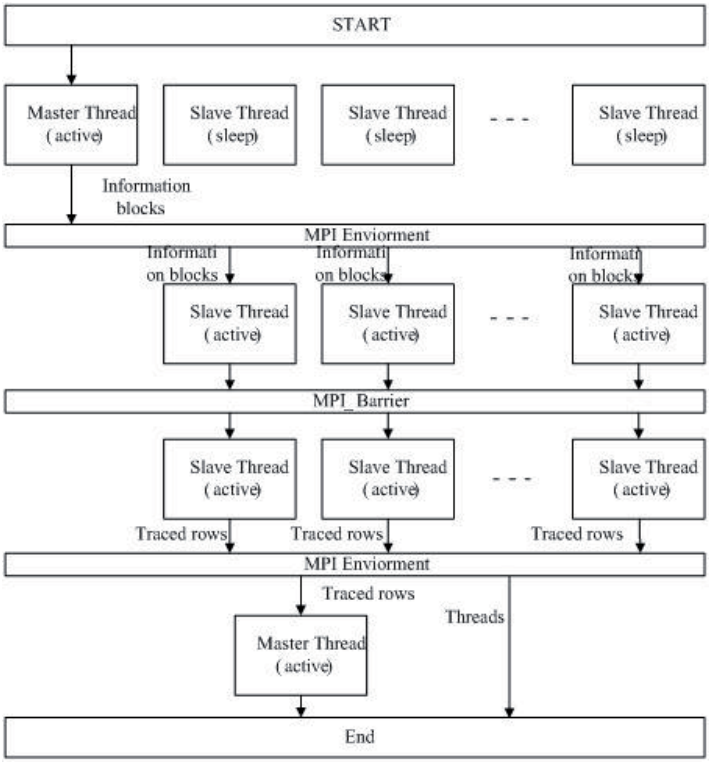

Рисунок 2 - Схема выполнения и взаимосвязей процессов SIMD алгоритма

```
for (int i = 1; i < commSize; i++)
 {
  sb = new sendingBlock(master.AA,0,0,master.Scene,
                        master.Step);
  sb.Begin = 300 / (commSize-1) * (i-1);
  sb.End = sb.Begin + 300 / (commSize-1);
  comm.Send<sendingBlock>(sb, i, 0);
 }
```

В следующем фрагменте приведен код, который обеспечивает прием данных в мастер-процессе.

```
if (rank == 0)
 {
  int row_id = -1;
  Drawing.Brush br = null;
  int[] status_array = new int[comm.Size];
  CompletedStatus cs;
  logging logcolor = new logging("logcolor");
  for (UInt16 cnt_rcv = 0; cnt_rcv < (300+(comm.Size-1));
       cnt_rcv++)
   {
    comm.Receive<int>(Communicator.anySource,
                      TAG_INFORMATION,
                      out row_id, out cs);
    comm.Receive<Color>(cs.Source, TAG_DATA,
    ref color_array, out cs);
    f.setBitmapRows(row_id, color_array);
   }
 }
```

Расчет времени выполнения трассировки сцены обеспечивает следующий код:

```
DateTime t1 = DateTime.Now;
……Some RT code ……
DateTime t2 = DateTime.Now;
TimeSpan dt = t2.Subtract (t1);
```

Свойства ЦП	
Тип ЦП	Mobile DualCore Intel Core 2 Duo T5500, 1666 MHz (10 x 167)
Multi CPU	
CPU #1	Intel(R) Core(TM)2 Duo CPU T5450 @ 1.66GHz, 1666 МГц
CPU #2	Intel(R) Core(TM)2 Duo CPU T5450 @ 1.66GHz, 1666 МГц

Рисунок 3 – Параметры тестовой станции

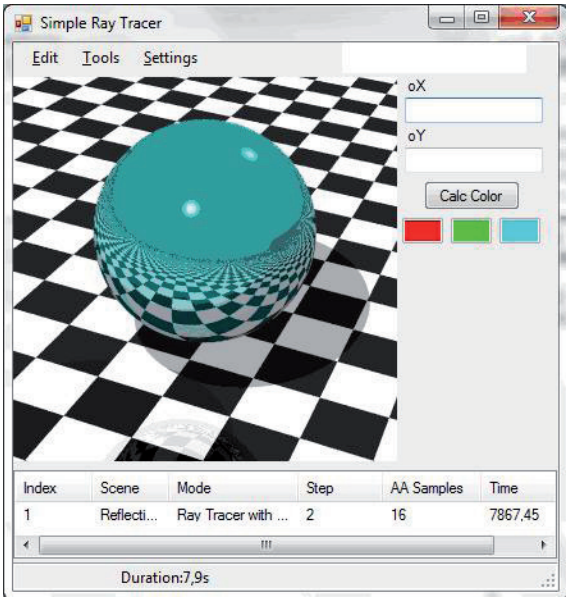

Рисунок 4 - Результаты выполнения алгоритма на 1 ядре (7.9 секунд)

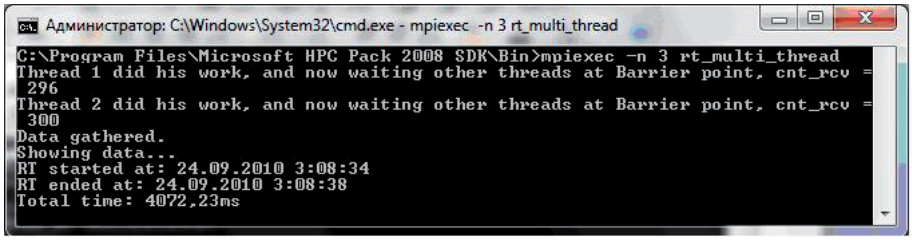

Рисунок 5 - Результаты выполнения алгоритма на 2 ядрах (4 секунды)

Анализ результатов распараллеливания алгоритма

Тестирование алгоритма проводилось на двуядерной системе, параметры системы приведены на рис 3.

Ниже праведны скриншоты результатов выполнения алгоритма на одном ядре (рис. 4) и на двух ядрах (рис. 5).

Прирост быстродействия на сцене размером 300 на 300 пикселей составил 100% (4 секунды против 7.9 секунды). С ростом количества процессоров без увеличения размера изображения прирост быстродействия остановится и со временем будет падать в точке, в которой мастер процесс не сможет своевременно принимать и обрабатывать результаты вычислений.

Выводы

Данная статья освещает исследование и реализацию параллельного ускоренного алгоритма трассировки лучей, реализованного с помощью технологии Message Passing Interface for .NET. Так же в статье описана основная проблема взаимосвязи количества процессоров и размера изображения. В дальнейшем планируется оптимизация алгоритма работы мастер процесса, направленная на ускорение обработки данных.

Литература

[1] SIMD (Wikipedia) – [электронный ресурс] – http://ru.wikipedia.org/wiki/SIMD.

[2] MIMD (Wikipedia) – [электронный ресурс] – http://ru.wikipedia.org/wiki/MIMD.

[3] Message Passing Interface for .NET – [электронный ресурс] - http://osl.iu.edu/research/mpi.net/

[4] Codermind team A ray tracer in C++ – [электронный ресурс] http://www.codermind.com/articles/Raytracer-in-C++-Introduction-What-is-ray-tracing.html