Table of Contents

Abstract. The implementation of parallel computing the integral of a function from Monte Carlo method is described in this assignment. Particularly, the assignment exploits the FastFlow parallel framework and C++ threads for building stream parallelism including pipeline and farm. The experiments illustrate the comparison of time consuming of these ways.

1 Introduction

With the development of hardware technologies and a plenty of data, the requirement of time consuming to process or compute some tasks have been noticed. Therefore, the parallelism is considered and utilized. The parallelism tasks are divided into some parts such as: stream parallelism, data parallelism, task parallelism and so on.
This assignment is about exploiting the stream parallelism on computing integral of a given function with Monte Carlo method. Because of a stream of interval numbers from the requirement of the project, the stream parallelism with pipeline and farm is utilized to parallel computing the Monte Carlo method for each interval number. To create a stream parallelism structure, the FastFlow (FF) framework and C++ threads are mentioned in this assignment. The remaining of the assignment is constructed: the Section 2 describes about the implementation of computing the integral of a given function with Monte Carlo. The results and experiments using FF and C++ threads are represented in Section 3, while Section 4 concludes the assignment.

2 Implementation

In this part of the report, the general idea of the Monte Carlo in computation of a integral function is design with FF and C++ threads. The basic idea is about forming an interval number object with basic fields and methods. Hence, each interval number goes through each stage which executes specific tasks in stream parallelism built by FF and C++ threads.

2.1 FastFlow

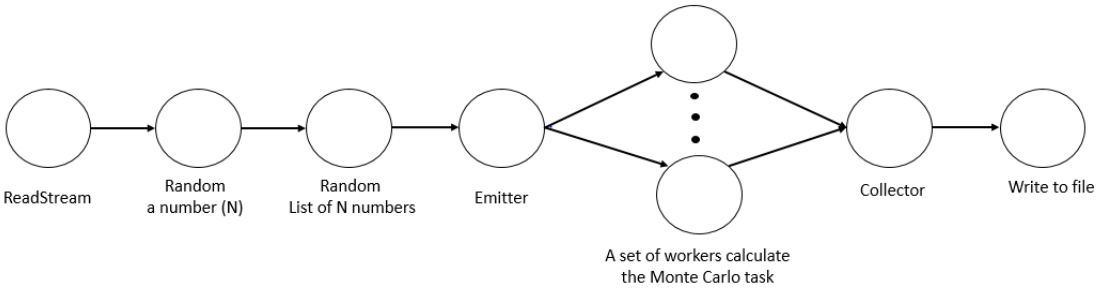

Fig. 1: The diagram of FF structure

To build a parallelism of the Monte Carlo method for computing integral of a function, the FF a structured parallel programming framework is utilized because of plenty of advantages such as highly efficient stream parallel and a set of ready-to-use for programmers. In addition, FF can be flexible for creation complex parallelism with a lot of different nodes.

With the Monte Carlo task, the FF framework is used to build a structure stream parallelism with pipeline and farm. Particularly, from the Fig 1, it can be clear that the structure is built based on a set of stages including: reading a set of interval numbers, choosing a random number (N), choosing a list of N numbers, setting Monte Carlo number calculations to threads and writing to a file.

Generally, each stage has a specific task to resolve. Because of the convenience of FF, each of stage or task or node is implemented as a function. However, if the a stage has more than one input or out put, it can be redefined. In more detail, the tasks of stages is described:

- readStream: takes a couple numbers at each time from a stream of file.
- random a number (N): randomly selects a number.
- random a list of N numbers: is generated from this stages.
- emitter: split interval numbers to free workers.
- a set of workers: compute Monte Carlo method for set of interval numbers.
- collector: takes bunch of interval numbers and send to the next stage.
- write to file: write the Monte Carlo to a file.

2.2 C++ Thread

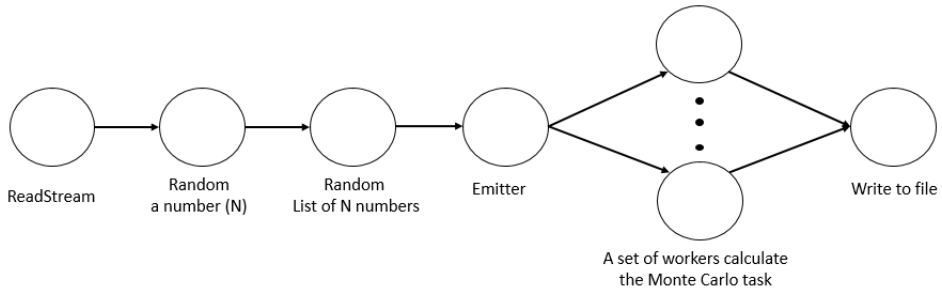

Fig. 2: The diagram of C++ threads structure

In case of using C++ threads, the similar construction is built with pipeline and farm on stream parallelism (in the Fig 2). However, without supporting from FF, the communication between stages is executed through queues and combination of mutex and condition variable in C++. In more detail, the queues run a role like containers which hold interval numbers executed from previous stage and waiting the next call of the next stage. Meanwhile, the mutex and condition variable are similar to doors and bells, respectively. Particularly, to start a stage by a thread, the mutex is used to lock the thread. After a stage has completed its task, the thread is unlocked by the mutex and sends a notify to next threads waiting for their turns. Based on some conditions, the notified threads can be waken up for their tasks or still sleep. In addition, each stage in case of C++ threads is implemented by a thread with a specific task like FF construction. The idea of stream parallelism in C++ threads construction is also similar the way of FF construction. Nevertheless, at the last stage, the C++ threads construction uses a queue to contain results of workers, in stead of using a collector to gather output of workers from farm. Then the final stage pops each result from queue to a file.

3 Experiments

The experiments of the Monte Carlo task with FF framework and C++ threads are mentioned in this section. Particularly, a set of different functions is tried with different of power. However, a fixed of interval numbers is considered with ... couples.

3.1 FastFlow

3.2 C++ thread

4 Conclusion

The computation of integral of a given function with Monte Carlo method is implemented with stream parallelism. In more detail, the implementation is built based on the combination of pipeline and farm with FF framework and C++ threads. The comparison of these techniques are

5 Ackowledgements

I would like to show my gratitude to Prof. Marco Danelutto and his Assistant Dr. Massimo Torquati about lessons in Distributed systems: paradigms and models course at University of Pisa. Additionally, I say thank to my friends who support and encourage me in this course.