1. Abstract

Nowadays, parallelism has become a new trend in software and architecture design, so that we usually consider a new solution for applications by using parallel patterns which include farm and pipeline pattern. And with cloud and edge deployments of parallel/distributed applications, the need for adaptivity in such applications has become important. For this, previous scientists have already conducted relative experiments to calculate the efficiency about running the stateful tasks on a parallel framework. However, balancing the computational load of multiple concurrent tasks on the parallel framework is also one of the critical requirements for efficient usage of such systems [1], and at the moment still no one knows what’s the efficiency if balancing the stateful tasks on a parallel framework. To address this problem, it’s necessary to measure how the ratio of sequence computing and parallel computing influence the running efficiency in a parallel pattern, and need to think about a new methodology or develop a new load-balancing task scheduler. Therefore, in this proposal, I have a plan to use C ++ generic and reusable parallel pattern interface ( GrPPI ) [2], developed by JavierLópez-Gómez ,JavierFernándezMuñoz, DaviddelRioAstorga, ManuelF.Dolz and J.DanielGarcia with a new MPI back end, it provides developers with a set of parallel programming patterns, such as Map, Task-Farm, and Pipeline. Finally, the expected experimental results that relative to the stateful pattern with adaptivity implementation built on GrPPI would be present, and I will predict the feasibility and efficiency of stateful pattern with adaptivity.

1. Introduction

With the development of processor technology, people can enjoy the benefits of high-performance processors. However, processor technology has reached the bottleneck [3], and it’s difficult for standard single-threaded code to increase more the computing speed. Therefore, in the recent years, everyone turned their attention to parallel programming. The theory of parallel computing is firstly dividing the code into each block, executes the blocks of code in parallel through multiple threads, and then integrates each sub-results together, which seamlessly extending computing power from a single processor to an unlimited number of processors. However, the implementation of parallel applications is a tortuous route. First of all, it is difficult to write explicit parallel programs because the programmer must specify how to divide the calculations on multiple processors, and must perform the necessary synchronization and data transfer operations, which not only has great requirements for programming ability, but also the program may contain some ignored error inside. In this case, programming using a parallel programming framework is particularly important, providing great convenience to developers. The popular mainstream parallel frameworks are OPENMP and MPI, etc., but these frameworks not totally suitable for exploring each kind of parallel patterns. C ++ generic and reusable parallel pattern interface ( GrPPI ) enables the execution of some streaming patterns on distributed platforms, which accommodates a unified layer of generic and reusable parallel patterns on top of existing execution environments and pattern-based frameworks.[4]

. However, in actual operation, the time spent on each task is different. The exact time required for each task depends on the task itself, and even base on the hardware environment at runtime. If you need to perform a lot of heavy tasks for a while, you may need more threads, and instead, you don't have to use so many threads when no many tasks. If in this way, our utilization efficiency of the CPU will be greatly increased. As we all know, stateful tasks can't get good parallel efficiency. Therefore, adaptive research on stateful tasks is also imperative and the current top priority. Once we understand their parallel operation efficiency through experiments and even try to improve their efficiency, it is possible to make an important contribution and breakthrough in the current field of parallel programming.

In this proposal, I propose a good idea about implementing an adaptive way with the stateful pattern and realize it in the GrPPI with a new MPI back end. My plan is evaluated by a series of benchmarks which include some synthetic models to verify the availability of the adaptivity

1. Methodology

I think GrPPI is a really good C++ framework to write parallel programs. The framework consists of many parallel patterns such as task-farm and pipeline. The task-farm pattern, for instance, is a composition by an emitter node, several worker nodes, and all nodes are implemented as individual OS threads. The nodes have input and output queues to connect to each other thus forming basic patterns node, with the help of an embedded load balancer, distributes tasks to the workers through their input queues. These workers, in principle, do the job concurrently, and the collector collects the results.[6]To implement the adaptivity function of this parallel framework, in the next few months, I need to learn how task scheduling works in the emitter of GrPPI. According to my previous experience, the basic idea of the adaptive implement is to focus on the number of active workers, i.e., during a certain period of time, the heavy tasks will use more workers, and easy tasks will use fewer workers. Then we must do the adaptive test with stateful tasks, which may not get much parallel speed up. I think the task-farm pattern can be used to implement the state access pattern. The task-farm contains n workers and each worker has a copy of the shared state. The emitter sends tasks to workers with the adaptive scheduling policy that we mentioned before. The scheduling policy makes sure tasks be sent in an adaptive way.

In order to perform a task in parallel in GrPPI and making it perfectly reflects its correctness, it is also necessary to make sure the correct order of the various small sub-tasks. That is to say, in what order the tasks in the emitter are input, the collectors must output the results in what order, and only in this way can the results be aggregated correctly.

1. Literature Review

This month I mostly focus my previous article "Adaptivity in Stateful Parallel Pattern" which makes me remember some research experience of the previous project and also has read 3 very good articles which closely relevant to this problem. The first which is the most important one written by Marco Danelutto, Peter Kilpatrick, Gabriele Mencagli and Massimo Torquati provides the idea of developing stateful parallel patterns by devising a classification of ways in which an application may access and update a state. The second written by Javier López-Gómez ,Javier Fernández Muñoz, David del Rio Astorga, Manuel F. Dolz and J. Daniel Garcia provide a good introduction to the GrPPI parallel pattern framework which I will use on this project. The third one provides what a parallel framework can do for real-world applications. All of these 3 articles are talking about the parallel pattern which is worthy of referencing in this research Project.

5.Expected Results and Discussion

I think in this project firstly I will measure the overhead of running the tasks on the parallel project and record their results, then adds some stateful tasks to the program and record their results. Finally, I will do the experiments with the adaptivity and get results, by comparing these results we may get some useful conclusion.

6.Conclusion and Recommendation

In this research proposal, I have presented some tools and learn the approach from my experience and some research articles. As we have seen the GrPPI with MPI framework that really could get a very good performance on parallel and there is still have lots of development space on the stateful access pattern. To this end, if this project could be chosen and as part of the future work, I will start to implement the stateful pattern on GrPPI, learn the GrPPI and MPI framework more, even if try to investigate the framework and improve its performance when using stateful access patterns on these frameworks.

1. References

[1] Md M, Kamran I, Michael R,Jose G, Adaptive Load-balancer for Task-scheduling Fastflow INFOCOMP 2015 : The Fifth International Conference on Advanced Communications and Computation page 6-12

[2] Javier L, Javier Fernández M, David del Rio A, Manuel F. Dolz and J. Daniel Garcia Exploring stream parallel patterns in distributed MPI environments Parallel Computing 84 (2019) 24–36

[3] Herb S ,The Free Lunch Is Over A Fundamental Turn Toward Concurrency in Software.Dr. Dobb's Journal, 30(3), March 2005

[4] Javier L, Javier Fernández M, David del Rio A, Manuel F. Dolz and J. Daniel Garcia Exploring stream parallel patterns in distributed MPI environments Parallel Computing 84 (2019) 24–36

[5] Marco D, Peter K, Gabriele M and Massimo T access patterns in stream parallel computations The International Journal of High Performance Computing Applications 2018, Vol. 32(6) 807–818

[6] Md M, Kamran I, Michael R,Jose G, Adaptive Load-balancer for Task-scheduling Fastflow INFOCOMP 2015 : The Fifth International Conference on Advanced Communications and Computation page 6-12