Name: Jiajin Pei
UPI: jpei190
Student ID: 136460163

A1 Assignment report

Implementation of Actor and CSP on solving the Longest Common Subsequence problem

Abstract

Parallel and concurrent programming are well-known difficult subjects. In the past few years, progress has been made in improving parallel and concurrent programming abstractions, techniques, and tools to simplify parallel and concurrent programming practices. This report will introduce two well known Parallel programming models: Actors and CSP (channel). By investigating these two models by setting up experiments on solving the Longest Common Subsequence (LCS) problem, and following by comparing their performance results, a conclusion can be made: the utilization of these two models could improve the computational performance on the LCS problem.

Background

Although the concept of concurrency and parallelism has existed since the first computer was introduced [13], multi-core processors became mainstream about 10 years ago, when AMD and Intel began selling dual-core processors for desktops. In order to make better use of multi-core technology, applications must be concurrent and parallel. A challenge was posted because it was well known that concurrent and parallel programming is difficult [12]. To reducing the burden of concurrent and parallel programming, great progress has been made including introducing new concurrent programming frameworks [8], models [6], and empirical researches. For instances, about how developers use/misuse concurrent and parallel programming constructs[21], as well as performance [2], satisfaction and error-propensities[10], and even energy assessments[11].

Concurrency vs Parallelism

Over the past decades, the CPU has been following the development of Moore's law, a single core frequency became faster and faster. However, in recent years, Moore's law has failed [1] because it is difficult for the CPU to balance between technological process and heat stability. Therefore, new strategy was to increase the number of cores. Quad-core home computers are very common nowadays where the server reached 32 cores and 64 threads. In order to make effective use of multicore CPUs, concurrency and parallelism should be considered at the code level.

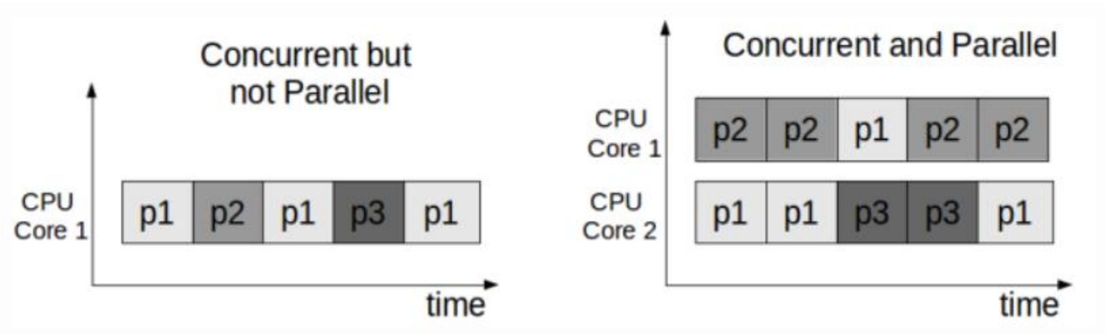
- Concurrency: When there are multiple threads operating, if the system has only one CPU, divide the CPU running time into several time periods and allocate them to each thread to execute. While the thread code of one time period is running, the other threads are suspended. This is called Concurrent.

  Concurrency = interval occurs

- Parallelism: When the system has more than one CPU, it is possible for threads to operate concurrently. When one CPU executes one thread, the other CPU can execute another thread. Two threads do not preempt the CPU resources and can do so simultaneously. This method is called Parallel.

Parallel = simultaneous occurs

- Distinction: parallelism refers to the occurrence of two or more events at the same time, while concurrency refers to the situation when two or more events occur at the same time interval. (1)



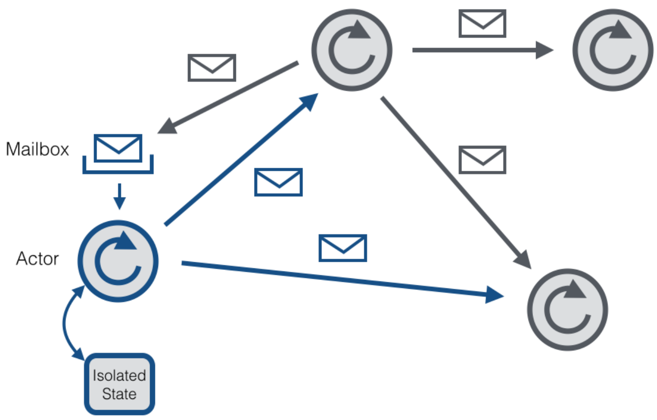
1. **Concurrency and Parallelism**

Actor

The Actor model was proposed in 1973 by Carl Hewitt, Peter Bishop and Richard Steiger [3]. An actor responds to incoming messages, makes local decisions, creates more actors, sends more messages and prepares to receive the next message. In Actor theory, everything is thought of as an Actor, much like in object-oriented programming (OOP) where everything is thought of as an object. For software, including object-oriented languages, usually executes sequentially, while the Actor model is essentially concurrent.

- Message

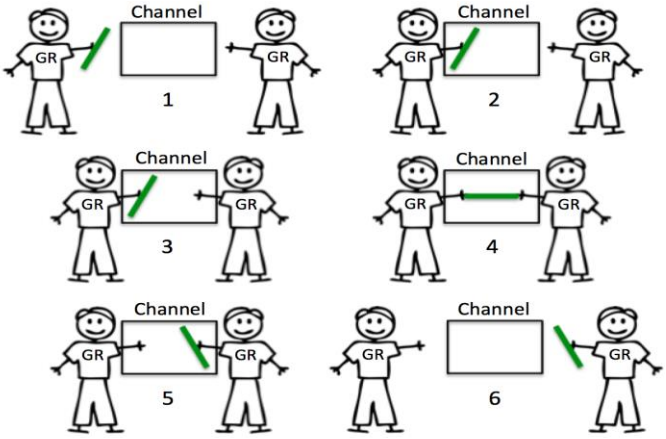
  In OOP, objects communicate with others through calling functions. Class A calls A function on class B, waits for the function to return, then class A can continue with the rest of the work. Actors communicate with other actors through messages. Messages are sent asynchronously, and it may take an arbitrarily long time to reach the recipient's mailbox. Moreover, the actor model does not guarantee the order of the messages (2). The message queuing and exit queues in the mailbox are atomic operations, so there is no race condition.



2. The bird view of Actor model

CSP(Channel)

The CSP version proposed by Hoare in his original paper in 1978 is essentially a concurrent programming language, not a process calculus [4]. A channel is a model for inter-process communication and synchronization through messaging. Messages can be sent through the channels. In the meantime, another process or thread can receive messages, such as streams, that are sent through the channel they referred (3). Different implementations of the channel can be buffered or unbuffered, either synchronous or asynchronous.



3. The conceptional picture of CSP

Longest Common Subsequence

The LCS problem is to find the longest subsequence shared by all sequences in a set of sequences (usually only two sequences). It differs from the longest common substring problem. Unlike substrings, substrings do not need to occupy consecutive positions in the original sequence.

For two sequences of n and m elements, the dynamic programming method runs for O(n×m) for any number of input sequences, and the solution given by the dynamic programming method is

$$O\left(N\prod_{i=1}^{N}n_i\right)$$

Literature review

The development of concurrent programming has gained significant popularity over the last decade. Dominik C et al. [7] designed and built CAF, a C++ Framework. They introduced a type-safe messaging interface to enhance the robustness of the actor program. They demonstrated the feasibility of CAF in several scenarios and found CAF continuously outperforms the competing actor environments Erlang, Charm++, SalsaLite, Scala, and ActorFoundry. Further improvement for CAF is to design effective monitoring systems and debugging facilities.

In terms of solving LCS problems, dynamic programming is the classic solution[5]. It is based on filling a scoring matrix through a scoring mechanism. The best score is the length of the LCS, and the subsequence of the LCS can be found through the trackback table. The time and space complexity of this dynamic programming approach is O(mn), where m and n are the lengths of the two compared strings.

Islam, Asha, and Ahmed proposed the Chemical Reaction Optimization (CRO), which is a new meta-heuristic method has widely used in solving optimization problems [9]. They found that the CRO has superior performance when compared with Dynamic LCS and Fast Dynamic LCS. However, the experiment test instances they used was small, the longest length of one of the strings was 360. It could underperform when given relatively larger length, for example, lengths of two strings are 30000.

Experiment

To compare the Actor and CSP model with sequential implementation, we can directly see how parallel programming boost the performance on computing the LCS.

First, we need a Sequential implementation (SEQ), used as a baseline to evaluate the parallel speedup.

Actor (ACT) implementation, used to evaluate the parallel speedup for the Actor model.

CSP (CSP) implementation, used to evaluate the parallel speedup for the CSP model.

In this experiment, I used dynamic LSC algorithm as my SEQ, the algorithm obeys the state transition equation, analyzing as below:

Set X = < x1, x2, x3, x4... , xm >, Y = < y1, y2, y3, y4... ,yn> is two sequences, Z=<z1,z2,z3,z4... Zk > is any common subsequence of them.

After analysis, we know that:

1. If *xm* = yn, zk = xm = yn and zk -1 is an LCS of xm -1 and yn -1
2. If xm! = yn and zk! Is equal to xm, then Z is an LCS of xm minus 1 and Y
3. If xm! = yn and zk! Is equal to yn, so Z is an LCS of X and yn minus 1.

$$C(i,j)\mathrm{f(x)} = \begin{cases} 0, & if\ i = 0\ or\ j = 0 \\ C[i-1, j-1] + 1, & If\ i,j > 0,\ x_i = y_i \\ max\{C[i, j-1], C[i-1, j]\}, & If\ i,j > 0,\ x_i \neq y_i \end{cases}$$

State Transition equation

Set: P1 is the length of the longest common subsequence of the first i -1 character of X and the first j character of Y

P2 is the length of the longest common subsequences of the first i characters of X and the first j - 1 characters of Y

P is the length of the longest common subsequences of the first i -1 character of X and the first j-1 character of Y

P0 is the length of the longest common subsequence of the first i characters of X and the first j characters of Y

If the ith character of X is the same as the JTH character of Y, then p0 = p + 1

If the ith character of X is not equal to the JTH character of Y, then p0 = Max (p1, p2)

ACT:

For the algorithm mentioned above perform in actor model, consider using multiple actors can compute small substring for 2 entire strings separately, a divide string mechanism was proposed, so that multiple smaller substring can sent to each actor. Divided substrings are the defined by bands. For instances, if the bands' number are 80, 80 of horizontal and vertical bands respectively, then there are 80 x 80 divided substring to be computed by actors. After computing the LCS of the smaller substring, Actors then sent the result to the mailbox of other actors. The waiting actors received the message and perform the same computation.

CSP:

The mechanism is the same as ACT. Instead of using sent message through mailbox, CSP allows each entity can sent message through channel. Other entities listen to the channel, then receive message and do further processing.

1. Experiment result:

The Lab computer runs the experiment has hardware support of:
Inter® Core ™ i5-6600 CPU @ 3.30GHz
Installed memory (RAM) 8.00GB (7.86 GB usable)
System type: 64-bit operating System, x64-based processor
Number of cores: 4
Two text files with great number of string length will be used as input. Length of file input 1: 100256, Length of file length 2: 100300.
10 sets of bands will be tested on both ACT and CSP. Measure the time duration of these models. Compare their result. The result is shown as follows:

Bands tested	ACT model time used (ms)	CSP model time used (ms)
1, 1	58381	58321
5, 5	24284	24491
10, 10	17010	17045
20, 20	14226	14404
30, 30	15085	13818
40, 40	17309	13680
50, 50	17548	13624
60, 60	18349	13529
80, 80	19994	13717
100, 100	23378	13800

We can see both ACT and CSP models decreased the time used as the horizontal and vertical bands increased. ACT model shows an increasing trend of duration after 30, 30 bands. Whereas, CSP continuously decreased duration.

Conclusion

The report introduced abstractions of ACT and CSP parallel programming models. The objective of this report is to present the implementation of these two models on solving the LCS problem. The result revealed that the models improved on the duration and outperformed the traditional dynamic algorithm. Also, the more bands used the less time duration to solve the LCS problem. Even the ACT and CSP are outperforming, in the practical situation, using actors or CSP are subject to changes in the whole application architecture mechanism and way of thinking. The concept of parallel programming can meaningful for computation, specifically when Moore's law is failing. Parallel programming could be the future trend of computer programming. Future work could focus on how the implementation of parallel programming becomes easier.

Reference:

[1] Kinnunen, Markku. (2015). Examining the limits of Moore's law - Possible influence of technological convergence on redefining the curriculum in ICT institutions.

[2] T. David, R. Guerraoui, and V. Trigonakis. Everything you always wanted to know about

[3] Carl Hewitt; Peter Bishop; Richard Steiger (1973). "A Universal Modular Actor Formalism for Artificial Intelligence". IJCAI.

[4] Hoare, C. A. R. (1978). "Communicating sequential processes". *Communications of the ACM*. **21** (8): 666–677. doi:10.1145/359576.359585.

[5] Hirschberg, D.S.: Algorithms for the Longest Common Subsequence Problem, Journal of the ACM, Vol. 24, No. 4, Vol. 18, NO. 6, 1975, pp. 341-343 1977, pp. 664-675

[6] A. Kulkarni, Y. Liu, and S. Smith. Task types for pervasive atomicity. In OOPSLA, 2010.
synchronization but were afraid to ask. In SOSP, 2013.

[7] Charousset, Dominik & Hiesgen, Raphael & Schmidt, Thomas. (2015). Revisiting Actor Programming in C++. Computer Languages, Systems & Structures. 45. 10.1016/j.cl.2016.01.002.

[8] D. Lea. A java fork/join framework. In Java Grande, 2000.

[9] M. Rafiqul Islam, Zarrin Tasnim Asha and R. Ahmed, "Longest Common Subsequence using Chemical Reaction Optimization," *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, Khulna, 2015, pp. 29-33.
doi: 10.1109/EICT.2015.7391917

[10] V. Pankratius, F. Schmidt, and G. Garreton. Combining functional and imperative programming for multicore software: An empirical study evaluating scala and java. In ICSE, 2012.

[11] G. Pinto, F. Castor, and Y. Liu. Understanding energy behaviors of thread management constructs. In OOPSLA, 2014.

[12] H. Sutter and J. Larus. Software and the concurrency revolution. Queue, 3(7):54–62, Sept. 2005. ISSN 1542-7730.

[13] J. von Neumann. First draft of a report on the edvac. IEEE Ann. Hist. Comput., 15(4):27–75, Oct. 1993. ISSN 1058- 6180.