International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 2, February- 2013

An AOP Solution for Checkpoint-Based Fault Tolerance in Distributed Computing

MohMohKhaing
Dept. of Information and Communication Technology, University of Technology (Yatanarpon Cyber City)

Abstract

Fault-tolerance is an important and critical issue in multiple DNA (Deoxyribonucleic Acid) sequence alignment on parallel computing using genetic algorithm. This system consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource to produce aligned sequences. While the larger size and computational works of parallel structure, there are more chances to become node failure and omission failure. Currently used fault tolerance approaches are high overhead and less dynamic in parallel computing environments. So, we propose fault tolerance system that combined with Synchronous Checkpoint/Rollback recovery approach and Aspect-Oriented Programming as single phase non-blocking. The proposed system is designed for the node failure and omission failure of distributed multiple sequence alignment application. This system avoids domino effect and less control messages between nodes. This system improves performance and accuracy of multiple sequence alignment computing. It supports code modularization away from code-tangling and scattering problems. It reduces the time complexity and overhead cost. The proposed system will be implemented on Linux platform by java language.

1. Introduction

Distributed and Parallel Computing uses many geographically independent computers and solves computationally intensive tasks efficiently [1].There are certain areas like air traffic control, railways signaling control, online banking and distributed disaster system, protein or DNA sequence computing.

DNA (Deoxyribonucleic Acid) is the basic unit of an organism, and its length ranges from a few hundred to several billions of nucleotides for different species [19]. The nucleotides are distinguished by a nitrogen base that can be of four kinds: adenosine (A), cytosine(C), guanine (G) and thymine (T).

Multiple DNA sequence alignment system is a powerful application for investigating the relationship between structures and function in bimolecular. Such alignment represents the evolutionary history of the group of sequences and used in mutagenesis experiments carried out by nature on a family of macromolecules.

In multiple DNA sequence alignment on parallel, given multiple DNA sequences (1…n) to be aligned, each sequence is broken into a number of substrings and sends these substrings to all worker nodes until all given n sequences are divided and send to workers by head node. Each worker works simultaneously align the subsequences using Genetic Algorithm (GA). GA computes alignment of input subsequences based on a evolutional approach that explores all the possible alignments between sub sequences by using genetic operators such as generating popularization ,selection, crossover and mutation of given subsequence. All subsequences are aligned simultaneously each on a different processor. The partial sub aligned result of each worker node are sent to head node and then that partial results are combined to produce as the new aligned sequences by head node and display the user [19].

In parallel DNA sequences alignment system, fault-tolerance is an important and critical issue [2]-[4]. As the size of computing sequence length increases, mean time increases failures such as node failure and omission failure. In absence of sufficient fault tolerance, huge human lives and money could be lost. Hence there is a strong need for improved algorithms for multiple fault tolerance with performance [5]-[6]-[7]. There are many fault tolerance approaches in parallel and distributed system: Replication-Based Recovery system, Fusion Based approach, Efficient mpich including checkpoint library compiler which has high overhead, and less dynamic in parallel computing environments and rarely used [16]-[13]. So, efficient single- phased non-blocking coordinated Checkpoint/Rollback fault tolerance system based on Aspect-Oriented Structure is proposed for parallel DNA sequences alignment using genetic algorithm

The organization of this paper is as follows. Section 2 presents the related works of fault-tolerance in parallel computing. Section 3 presents background theory and proposed system is followed in section 4 and section 5 is expected outcomes and conclusion of this paper.

2. Related Works

Y. Tang proposed checkpoint and rollback recovery based fault tolerance system [11]. However, this system's main drawback is expensive in terms of time and overhead. Their system relies on processor virtualization to achieve migration transparency. J.P. Walters proposed replication-based fault tolerance for MPI application [12]. However related issues are consistency among such replicas and need to be addressed the overhead carefully. Major problem of this system is number of backups so that fusion based approach is appeared [13]. That is emerging as a popular technique to handle multiple faults but requires fewer backup machines than replication based approaches. In fusion based fault tolerance technique, backup machines are used which is cross products of original computing machines. [14]. But there is high overhead during failure recovery. Hence this technique is suitable if the percentage of fault is low. S. Bansai [3] projected Dynamic Rank-Based fault tolerance algorithm for load redistribution works as a sequential restoration algorithm and reassignment algorithm for distribution of failure nodes to least loaded computing nodes works as a concurrent recovery reassignment algorithm. M.Tripathy [15] planned Hierarchical Checkpointing that includes 3 types of checkpoint: Local disk, Mirrored, Storage and three types of recovery for transient failure, permanent failure and storage failure. S.Kumar [16] anticipated Two Phase Coordinated Checkpointing algorithm which has first phase is each process takes a tentative checkpoint and second phase is tentative checkpoint was replaced by the permanent one. It takes the waiting time both tentative checkpoint and permanent checkpoint.

3. Background Theory

3.1. DNA Sequence Alignment

Multiple Sequence Alignment (MSA) [23] refers to the problem of optimally aligning three or more sequences with inserting gaps between the genes. The objective is to the number of matching gene symbols between the sequences and also use only minimum gap insertions. Identical sub sequences are eventually aligned in a one-to-one correspondence naturally; gaps are not inserted in both sequences at the same position. In the end, the sequences end up with the same size. MSA helps to design new protein or genes and MSA belongs to a class of optimization problems with exponential time complexity, called combinatorial problems. Figure.1 illustrates an alignment between two DNA sequences.

```
A  =  ACAAGACAG-CGT
      | || | || |||
B  =  AGAACA-AGGCGT
```

Figure 1. Alignment of two sequences

3.2. Genetic Algorithm

Genetic Algorithm (GA) [8]-[9] has been proven to be robust, flexible and efficient in vast complex spaces including NP_ complete problems. The genetic algorithm [10] is used in the sciences, engineering and the business world. This algorithm has now been successfully applied to optimization problems, scheduling, data fitting, clustering trend spotting and path finding. It is an evolutionary system with the operators of selection, crossover and mutation. Firstly, genetic algorithms produces number of population consists of chromosomes including genes and then its three operators are applied on each population set in Figure 2.

```
Begin
Generate the initial population P;
for i=2 to genSize do
  for j=1to (popSize/2) do
    Select two chromosomes X and Y from P by Selection operation;
    Crossover(X,Y) and let X' and Y' be the offspring;
    Mutate(X');
    Mutate(Y');
    Put X' and Y' into mating pool M
End
Let M be the new population P
```

Figure 2. Step of gentic algorithm

The three operators in Genetic Algorithm are:
(1) Selection operation: The selection operator chooses better chromosomes in a population according to the fitness values of chromosomes. The roulette wheel

selection method and the ranking selection method are commonly used in selection methods.
(2) Crossover operation: The system chooses a pair of chromosomes after the selection operation is done, and then randomly generates a crossover point. A crossover rate must be given by the user for determining whether the pair of chromosomes will crossover or not and produces the local optimum solution. The crossover methods exists single-point crossover, the double-point crossover, and the uniform crossover.
(3) Mutation operation: The system uses the local optimum solution which came from the crossover operation and transforms to the global optimum by using mutation operators. The mutation rate must be given by the user for determining whether a chromosome will mutate or not. The binary mutation and the real value mutation are commonly used mutation methods.

3.3. Checkpointing-Based Rollback Recovery

Checkpointing technique has been used in distributed system as fault tolerance mechanism. A checkpoint is a snapshot of the current state of a process and saves enough information in non-volatile stable storage. If the volatile storage contents are lost due to process failure, one can reconstruct the process state from the information saved in the non-volatile stable storage. Checkpoint-Based Rollback Recovery techniques can be divided into three subcategories: asynchronous or uncoordinated checkpointing, synchronous or coordinated checkpointing and quasi-synchronous or communication induced checkpointing. In coordinated or synchronous checkpointing, processes take checkpoints independently and then the local checkpoints are made to get global consistent state as two-phase commit structure [17]. In the first phase, processes take tentative checkpoints and these are made to be permanent checkpoint state in the second phase. In case of failure, faulty processes rollbacks to last checkpoint state and make failure recovery. The coordinated checkpointing protocols can be classified into two types: blocking and non-blocking. In blocking algorithms, some processes take place in blocking during taking checkpoint and solving the failures. In non-blocking algorithms, no blocking processes are required for taking checkpoint and recovery of failures [18].So, running process will not exist in wait state.

3.4. Aspect-Oriented Programming

Aspect-oriented software development (AOSD) is a new technique that improves separation of concerns in software development [20]. AOSD makes it possible to modularize cross-cutting concerns of a software system to maintain and evolve easily. An aspect-oriented design can lead to be better system architecture, and an aspect-oriented programming language enforces a disciplined coding style.

Aspect-Oriented Programming (AOP) is a complementary programming technique to the object-oriented programming (OOP). It provides tools for managing so called cross-cutting concerns. Log writing, authorization and transaction management can be taken as examples of this kind of cross-cutting functionality. In aspect-oriented programming, the system is divided into a two halves: the base program and the aspect program. The base program contains the main functionality of the system and can be implemented using object-oriented programming. The aspect program consists of the cross-cutting functionality that has been modularized away from the base program. Concerns are implemented in AOP by using units of code called aspects. Aspects contain pieces of code called advices which are used to implement the crosscutting concern and the places where the advice should be applied to the OOP base-code are defined using joinpoints. A weaver is used to combine the aspect code with base-code so that the appropriate links can be inserted at the places within the base-code specified by the joinpoints to reference the appropriate aspect code. This naturally reduces the amount of duplicated code and decreases the error probability.AOP can also be used to add new functions or to modify existing functions in the base program [21]-[22].

4. Proposed System

4.1. Architecture

The proposed Checkpoint-Based fault tolerance system will be implemented on parallel DNA sequence alignment application. This proposed system architecture includes only one head node that can be called master node and several worker nodes or slaves. All head node and workers are connected with local network and they computed their jobs without sharing storage spaces and input/output streams. They execute their jobs independently.

The Head node accepts the multiple DNA sequences as input and divides each sequence into sub sequences until all input sequences are completed and sends them to all workers. Each worker node computes alignment of sub sequences applying genetic algorithm and sends new partial sequence result to head node. Head node combines the partial results and displays the new sequences to user. Node failures and omission failures can occur frequently when computing multiple long DNA sequences. To demand the fault tolerance, this paper is proposed the efficient checkpointing algorithm unlike traditional checkpointing. Fig. 3 shows the architecture of checkpoint based fault tolerance system.

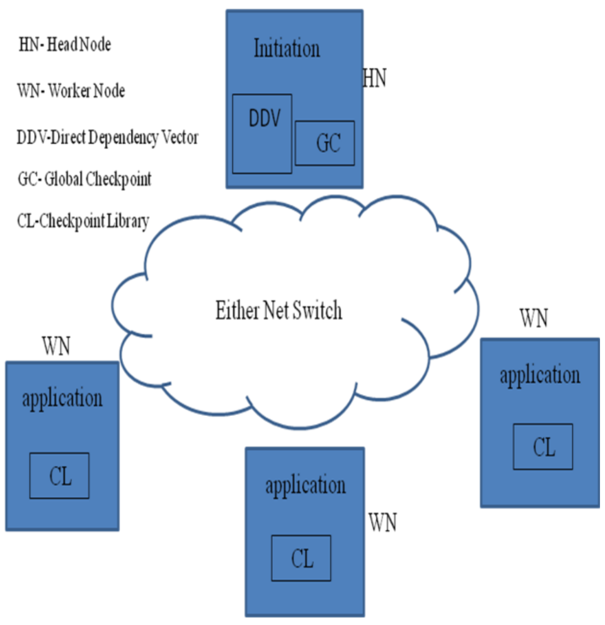

Figure 3. Architecture of proposed system and its component

To tolerance fault, Direct Dependency Vector (DDV), Global Checkpoint Storage (GL) and Local checkpoint Library (CL) are placed in head node (HN). Head Node creates the global checkpoint by taking the local checkpoint of each worker node. Direct Dependency Vector (DDV) is vector arrays that create the vector as the number of worker nodes to stores the state and content of each worker node at taking the global checkpoint. In each worker node, Local Checkpoint Library (CL) is placed and takes the checkpoint independently and periodically.

The proposed Checkpointing algorithm divides into two parts and applies initiator algorithm separately in head node at Fig. 4(a) and applies application algorithm Fig. 4(b) in every worker nodes .Only checkpoint initiator program will be run in head node and the checkpoint application program will be run in all worker nodes. Initiator process and application process are illustrated in Fig. 4(a) and 4(b).

```
Begin
  Check sending of message after its latest checkpoint
  Take Forced checkpoint and add Checkpoint sequence number
  Send control message to all processes
  Continue normal operation
End
```

(a)

```
Begin
  Check receive of control message from head node after its latest checkpoint
  Take Forced checkpoint and add Checkpoint sequence number
  Send control message to all processes
  Continue normal operation
End
```

(b)

Figure 4. (a) Initiator process in head node (b) Application process in each worker node

This paper presents checkpointing algorithm is a single phase algorithm because the initiator process interacts with the others processes only via the control message. Also a process after its participation in the algorithm does not wait for the algorithm to terminate before resuming its normal operation. This is called non-blocking and the proposed system differs from the conventional approach that is no need to take first temporary checkpoints and then converting them to permanent ones by processes. The proposed single phase checkpointing algorithm allows processes to take permanent checkpoints directly, without taking temporary checkpoints. The orphan messages are eliminated by sender processes and the in-transit messages are eliminated by checkpointing interval and retransmission mechanism.

4.2 Processing Steps

In proposed Checkpoint/Recovery System has three processing steps: (1) Taking Global Checkpoint or Recovery Line (2) Failure detection in worker node (3) Failure Recovery for failed node.

4.2.1. Taking Global Checkpoint or Recovery Line

Head node takes forced checkpoint to make global checkpoint and send control messages to all worker nodes. All workers receive message and take forced checkpoint and send response to HN.HN computes the global consistent checkpoint for the whole checkpoint and store global checkpoint library are shown in Fig. 5.

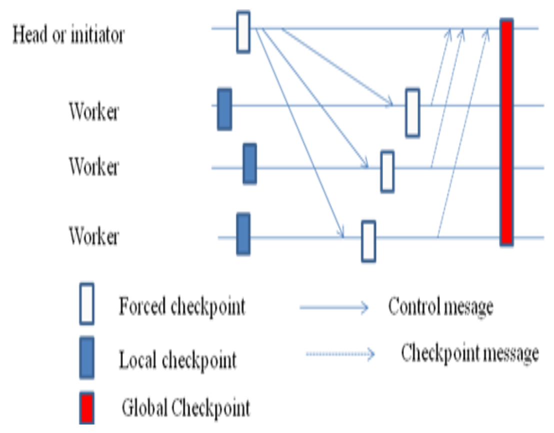

Figure 5. Recovery line

To reduce the orphan message and in-transit message, checkpointing interval is critical issue for recovery algorithm. In proposed algorithm, the value of the checkpointing interval was decided the time just larger than the maximum message passing time between any two processes of the system which differ from other conventional approach.

4.2.2. Failure Detection in Worker Node

Monitoring program in Head node watches the process of workers. If hardware or software failure occurs in worker node, monitor detects the failures and the head node transfers the load to another less load node.

4.2.3. Failure Recovery

The failure node is recovered by using the global checkpoint stored in head node. After recovery, it broadcasts alert message to head node and all worker node. All workers decide if they need to rollback or not by checking the corresponding entries in DDV vector upon receiving message. The recovery process is shown in Figure 6.

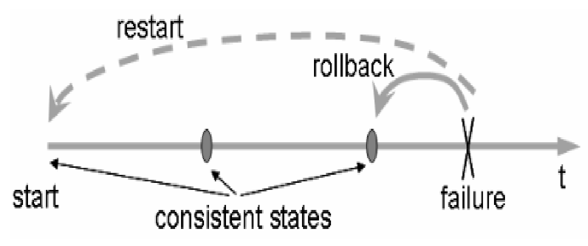

Figure 6. Failure recovery with rollback and restart

4.3. AOP Solution for Checkpoint based fault Tolerance System

In proposed checkpoint/recovery fault tolerance system, taking checkpoint, sending and receiving the control message are occurred between interactive of nodes. Monitoring the activities of workers and detection the error of workers are made periodically. Failure recovery makes placing the job from failure node to one node which has least load among other working nodes when failure occurs. These replacing jobs are occurred frequently and code tangling and crosscutting concerns occur between codes. So, the checkpoint/recovery system may be decreased performance. So, this system applies the program components in both initiator and application program into aspects and waves the base checkpointing algorithm. Figure 7 presents overview of the aspect oriented system design for checkpoint based recovery system.

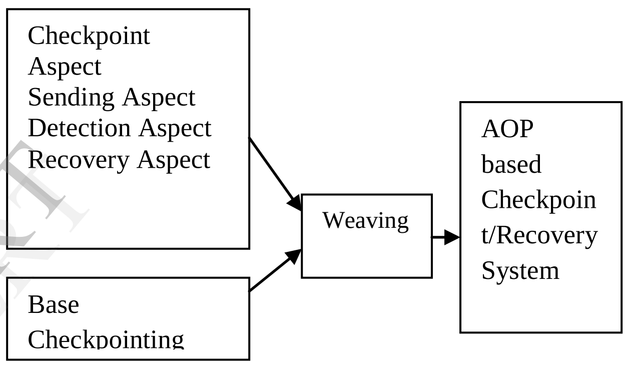

Figure 7. AOP design for checkpoint/recovery system

5. Conclusion

The proposed checkpointing algorithm offers the following important advantages. In DNA sequence alignment, the processes that have sent some messages after their latest checkpoints, take checkpoints during checkpointing period; thereby reducing the number of forced checkpoints to be taken. The algorithm is a synchronous one. However it differs from the classical synchronous approach in the following sense; it is just a single phase one unlike the two-phase classical approach, it does not need any exchange of additional (control) messages to coordinate the processes except only the request message, there is no synchronization delay. Another advantage of the proposed algorithm is that it is non-blocking which means that application processes are not suspended during checkpointing. This single phase non-blocking nature of the algorithm definitely contributes to its speed of execution. This proposed system improves code modularization by avoiding code-tangling and code-scattering and improves the performance of parallel computing system. The proposed fault tolerance in parallel DNA sequence computing will performs simultaneously well for multiple DNA sequences with a large number of genes with more reliable and more accurately.

6. References

[1] G. Georgina, Summary of parallelization and control approaches and their exemplary application for selected algorithms or applications, LarKC, 2008.

[2] R.Buyya, *High performance cluster computing: architectures and systems* (Upper Saddle River, N.J., USA :Prentice Hall, 1999).

[3] S.Bansai, S.Sharma, An Improved Multiple Faults Reassignment based Recovery in Cluster Computing, *Journal of Computing*, *Volume2*, November 2010, 2151-9617

[4] Tina Riedel, *Introduction to cluster and parallel computing* (Version 1.3: Training and Documentation Tufts University, 2003).

[5] S. Monnet, C. Morin, R.Badrinath, Hybrid checkpointing for parallel applications in cluster federations, *IEEE International Symposium on Cluster Computing and the Grid*, 2004, 773-782.

[6] B.Gupta, S. Rahimi, R. Ahmad, A New Roll-Forward Checkpointing Recovery Mechanism for Cluster Federation, *International Journal of Computer Science and Network Security,volume 6*, 2006, 292-298

[7] B. Gupta, S.Rahimi, Y.Yang, A Novel Roll-Backward Mechanism for Performance Enhancement of Asynchronous Checkpointing and Recovery, *Informatica (Slovenia),volume.31*, 2007, 1-13.

[8] Cheng. R, *Genetic algorithms and engineering design* (John Wiley & Sons, New York, U.S.A,1997).

[9] K.wong, Genetic algorithms: concepts and applications, *IEEE Transactions on Industrial Electronics*, 1996, 519-534.

[10] Michalewic.Z, *Genetic algorithms + data structure = evolution programs* (Springer, Berlin, Ger-many,1992).

[11] Y.Tang, G. Fagg and J. Dongarra, Proposal of MPI operation level checkpoint/rollback and one implementation, *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, 2006.

[12] J. P. Walters and V.Chaudhary, Replication-Based fault Tolerance for MPI Applications, *IEEE Transactions On Parallel and Distributed Systems, volume 20*, 2009.

[13] V. Ogale, B. Balasubramanian and V. K. Garg, Fusion-based Approach for Tolerating Faults in Finite State Machines, *IEEE International Symposium on Parallel & Distributed Processing*, 2009.

[14] V.K Garg, *Implementing fault-tolerant services using fused state machines*(Technical Report ECE-PDS-2010-001, Parallel and Distributed Systems Laboratory,ECE Dept, University of Texas at Austin ,2010).

[15] M.Tripathy, C.R.Tripathy, A Hierarchical Shared Memory Cluster Architecture with Load Balancing and Fault Tolerance, *International Journal of Computer Applications ,Volume 25*, June 2011.

[16] S. Kumar, P. Kumar, Hierarchical Non-blocking Coordinated Checkpointing Algorithms for Mobile Distributed Computing, *International Journal of Computer Science and Security (IJCSS), Volume (3*), 2002.

[17] P.Kumar, R.Setiya, and P.Gahlan ,Checkpointing Algorithms for Distributed Systems, *International Journal of Computing Science and Communication Technologies, Volume 2*, July 2009.

[18] L. Guoliang, C. Shuyu, Z. Xiaoqin, A Non-blocking Checkpointing Algorithm for Distributed Systems, *International Journal of Digital Content Technology and its Applications, Volume 5*, July 2011.

[19] J. D. Gradecki, N. Lesiecki, *Mastering aspectJ* (Wiley Publishing, Inc., Indianapolis, Indiana, 2008).

[20] S.Chen, C.Lin, Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer , *International Journal of Applied Science and Engineering ,Volume 3*, 2005, 89-100.

[21] J. Laukkanen, *Aspect-oriented programming* (University OF Helsinki, Department of Computer Science, 2008).

[22] I. Kiselev, *Aspect-oriented programming with aspectJ* (Sams Publishing ,2003).

[23]http://en.wikipedia.org/wiki/SmithWaterman_algorithm