Computers in Biology and Medicine 48 (2014) 28–41

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

Automating fault tolerance in high-performance computational biological jobs using multi-agent approaches

Blesson Varghese [a,*], Gerard McKee [b], Vassil Alexandrov [c]

[a] School of Computer Science, University of St. Andrews, UK
[b] Faculty of Computing and IT, Baze University, Nigeria
[c] Barcelona Supercomputing Centre, Spain

ARTICLE INFO

Article history:
Received 9 December 2013
Accepted 12 February 2014

Keywords:
High-performance computing
Fault tolerance
Biological jobs
Multi-agents
Seamless execution
Checkpoint

ABSTRACT

Background: Large-scale biological jobs on high-performance computing systems require manual intervention if one or more computing cores on which they execute fail. This places not only a cost on the maintenance of the job, but also a cost on the time taken for reinstating the job and the risk of losing data and execution accomplished by the job before it failed. Approaches which can proactively detect computing core failures and take action to relocate the computing core's job onto reliable cores can make a significant step towards automating fault tolerance.

Method: This paper describes an experimental investigation into the use of multi-agent approaches for fault tolerance. Two approaches are studied, the first at the job level and the second at the core level. The approaches are investigated for single core failure scenarios that can occur in the execution of parallel reduction algorithms on computer clusters. A third approach is proposed that incorporates multi-agent technology both at the job and core level. Experiments are pursued in the context of genome searching, a popular computational biology application.

Result: The key conclusion is that the approaches proposed are feasible for automating fault tolerance in high-performance computing systems with minimal human intervention. In a typical experiment in which the fault tolerance is studied, centralised and decentralised checkpointing approaches on an average add 90% to the actual time for executing the job. On the other hand, in the same experiment the multi-agent approaches add only 10% to the overall execution time.

1. Introduction

The scale of resources and computations required for executing large-scale biological jobs are significantly increasing [1,2]. With this increase the resultant number of failures while running these jobs will also increase and the time between failures will decrease [3–5]. It is not desirable to have to restart a job from the beginning if it has been executing for hours or days or months [6]. A key challenge in maintaining the seamless (or near seamless) execution of such jobs in the event of failures is addressed under research in fault tolerance [7–10].

Many jobs rely on fault tolerant approaches that are implemented in the middleware supporting the job (for example [6,11–13]). The conventional fault tolerant mechanism supported by the middleware is checkpointing [14–17], which involves the periodic recording of intermediate states of execution of a job to which execution can be returned if a fault occurs. Such traditional fault tolerant mechanisms, however, are challenged by drawbacks such as single point failures [18], lack of scalability [19] and communication overheads [20], which pose constraints in achieving efficient fault tolerance when applied to high-performance computing systems. Moreover, many of the traditional fault tolerant mechanisms are manual methods and require human administrator intervention for isolating recurring faults. This will place a cost on the time required for maintenance.

Self-managing or automated fault tolerant approaches are therefore desirable, and the objective of the research reported in this paper is the development of such approaches. If a failure is likely to occur on a computing core on which a job is being executed, then it is necessary to be able to move (migrate) the job onto a reliable core [21]. Such mechanisms are not readily available. At the heart of this concept is mobility, and a technique that can be employed to achieve this is using multi-agent technologies [22].

* Corresponding author.
E-mail addresses: varghese@st-andrews.ac.uk (B. Varghese), gerard.mckee@bazeuniversity.edu.ng (G. McKee), vassil.alexandrov@bsc.es (V. Alexandrov).
URL: http://www.blessonv.com (B. Varghese).

http://dx.doi.org/10.1016/j.compbiomed.2014.02.005
0010-4825

Two approaches are proposed and implemented as the means of achieving both the computation in the job and self-managing fault tolerance; firstly, an approach incorporating agent intelligence, and secondly, an approach incorporating core intelligence. In the first approach, automated fault tolerance is achieved by a collection of agents which can freely traverse on a network of computing cores. Each agent carries a portion of the job (or sub-job) to be executed on a computing core in the form of a payload. Fault tolerance in this context can be achieved since an agent can move on the network of cores, effectively moving a sub-job from one computing core which may fail onto another reliable core.

In the second approach, automated fault tolerance is achieved by considering the computing cores to be an intelligent network of cores. Sub-jobs are scheduled onto the cores, and the cores can move processes executed on them across the network of cores. Fault tolerance in this context can be achieved since a core can migrate a process executing on it onto another core.

A third approach is proposed which combines both agent and core intelligence under a single umbrella. In this approach, a collection of agents freely traverse on a network of virtual cores which are an abstraction of the actual hardware cores. The agents carry the sub-jobs as a payload and situate themselves on the virtual cores. Fault tolerance is achieved either by an agent moving off one core onto another core or the core moving an agent onto another core when a fault is predicted. Rules are considered to decide whether an agent or a core should initiate the move.

Automated fault tolerance can be beneficial in areas such as molecular dynamics [23–26]. Typical molecular dynamics simulations explore the properties of molecules in gaseous, liquid and solid states. For example, the motion of molecules over a time period can be computed by employing Newton's equations if the molecules are treated as point masses. These simulations require large numbers of computing cores that run sub-jobs of the simulation which communicate with each other for hours, days and even months. It is not desirable to restart an entire simulation or to loose any data from previous numerical computations when a failure occurs. Conventional methods like periodic checkpointing keep track of the state of the sub-jobs executed on the cores, and helps in restarting a job from the last checkpoint. However, overzealous periodic checkpointing over a prolonged period of time has large overheads and contributes to the slowdown of the entire simulation [27]. Additionally, mechanisms will be required to store and handle large data produced by the checkpointing strategy. Further, how wide the failure can impact the simulation is not considered in checkpointing. For example, the entire simulation is taken back to a previous state irrespective of whether the sub-jobs running on a core depend or do not depend on other sub-jobs.

One potential solution to mitigate the drawbacks of checkpointing is to proactively probe the core for failures. If a core is likely to fail, then the sub-job executing on the core is migrated automatically onto another core that is less likely to fail. This paper proposes and experimentally evaluates multi-agent approaches to realising this automation. Genome searching is considered as an example for implementing the multi-agent approaches. The results indicate the feasibility of the multi-agent approaches; they require only one-fifth of the time compared to that required by manual approaches.

The remainder of this paper is organised as follows. Section 2 presents the three approaches proposed for automated fault tolerance. Section 3 highlights the experimental study and the results obtained from it. Section 4 presents a discussion on the three approaches for automating fault tolerance. Section 5 summarises the key results from this study.

2. Methods

Three approaches to automate fault tolerance are presented in this section. The first approach incorporates agent intelligence, the second approach incorporates core intelligence, and in the third a hybrid of both agent and core intelligence is incorporated.

2.1. Approach 1: fault tolerance incorporating agent intelligence

A job, J, which needs to be executed on a large-scale system is decomposed into a set of sub-jobs $J_1, J_2 \ldots J_n$. Each sub-job $J_1, J_2 \ldots J_n$ is mapped onto agents $A_1, A_2 \ldots A_n$ that carry the sub-jobs as payloads onto the cores, $C_1, C_2 \ldots C_n$ as shown in Fig. 1. The agents and the sub-job are independent of each other; in other words, an agent acts as a wrapper around a sub-job to situate the sub-job on a core.

There are three computational requirements of the agent to achieve successful execution of the job: (a) the agent needs to know the overall job, J, that needs to be achieved, (b) the agent needs to access data required by the sub-job it is carrying and

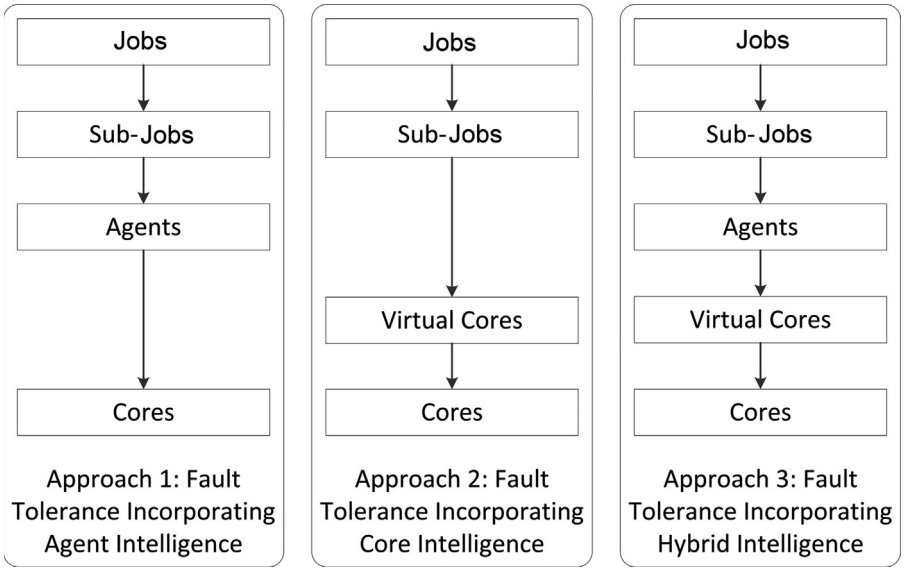

Fig. 1. The job, sub-jobs, agents, virtual cores and computing cores in the two approaches proposed for automated fault tolerance.