Chapter 33
Scalable Fault Tolerance for Large-Scale Parallel and Distributed Computing

Zizhong Chen
Colorado School of Mines, USA

ABSTRACT

Today's long running scientific applications typically tolerate failures by checkpoint/restart in which all process states of an application are saved into stable storage periodically. However, as the number of processors in a system increases, the amount of data that need to be saved into stable storage also increases linearly. Therefore, the classical checkpoint/restart approach has a potential scalability problem for large parallel systems. In this chapter, we introduce some scalable techniques to tolerate a small number of process failures in large parallel and distributed computing. We present several encoding strategies for diskless checkpointing to improve the scalability of the technique. We introduce the algorithm-based checkpoint-free fault tolerance technique to tolerate fail-stop failures without checkpoint or rollback recovery. Coding approaches and floating-point erasure correcting codes are also introduced to help applications to survive multiple simultaneous process failures. The introduced techniques are scalable in the sense that the overhead to survive k failures in p processes does not increase as the number of processes p increases. Experimental results demonstrate that the introduced techniques are highly scalable.

INTRODUCTION

The unquenchable desire of scientists to run ever larger simulations and analyze ever larger data sets is fueling a relentless escalation in the size of supercomputing clusters from hundreds, to thousands, and even tens of thousands of processors (Dongarra, Meuer & Strohmaier, 2004). Unfortunately, the struggle to design systems that can scale up in this way also exposes the current limits of our understanding

DOI: 10.4018/978-1-60566-661-7.ch033

of how to efficiently translate such increases in computing resources into corresponding increases in scientific productivity. One increasingly urgent part of this knowledge gap lies in the critical area of reliability and fault tolerance.

Even making generous assumptions on the reliability of a single processor, it is clear that as the processor count in high end clusters grows into the tens of thousands, the mean time to failure (MTTF) will drop from hundreds of days to a few hours, or less. The type of 100,000-processor (Adiga, et al., 2002) machines projected in the next few years can expect to experience a processor failure almost daily, perhaps hourly. Although today's architectures are robust to enough incur process failures without suffering complete system failure, at this scale and failure rate, the only technique available to application developers for providing fault tolerance within the current parallel programming model checkpoint/restart has performance and conceptual limitations that make it inadequate to the future needs of the communities that will use these systems. Alternative fault tolerance techniques need to be investigated.

In this chapter, we present some scalable techniques to tolerate a small number of process failures in large scale parallel and distributed computing. The introduced techniques are scalable in the sense that the overhead to survive k failures in p processes does not increase as the total number of application processes p increases. We introduced several encoding strategies into diskless checkpointing to improve the scalability of the technique. We present an algorithm-based checkpoint-free fault tolerance approach, in which, instead of taking checkpoint periodically, a coded global consistent state of the critical application data is maintained in memory by modifying applications to operate on encoded data. Because no periodical checkpoint or rollback-recovery is involved in this approach, process failures can often be tolerated with a surprisingly low overhead. We explore a class of numerically stable floating-point number erasure codes based on random matrices which can be used in the algorithm-based checkpoint-free fault tolerance technique to tolerate multiple simultaneous process failures. Experimental results demonstrate that the introduced fault tolerance techniques can survive a small number of simultaneous processor failures with a very low performance overhead.

BACKGROUND

Current parallel programming paradigms for high-performance distributed computing systems are typically based on the Message-Passing Interface (MPI) specification (Message Passing Interface Forum,1994). However, the current MPI specification does not specify the behavior of an MPI implementation when one or more process failures occur during runtime. MPI gives the user the choice between two possibilities on how to handle failures. The first one, which is the default mode of MPI, is to immediately abort all survival processes of the application. The second possibility is just slightly more flexible, handing control back to the user application without guaranteeing that any further communication can occur.

FT-MPI Overview

FT-MPI (Fagg, Gabriel, Losilca, Angskun, Chen, Pjesivac-Grbovic, et al., 2004) is a fault tolerant version of MPI that is able to provide basic system services to support fault survivable applications. FT-MPI implements the complete MPI-1.2 specification and parts of the MPI-2 functionality, and extends some of the semantics of MPI to support self-healing applications. FT-MPI is able to survive the failure of $n - 1$ processes in an n-process job, and, if required, can re-spawn the failed processes. However, fault