Optimal Recovery Schemes in Distributed COMPUTING

R. Delhi Babu1 and P. Sakthivel2

[1]Department of Computer Science and Engineering,
Sri Sivasubramaniya Nadar College of Engineering, Chennai-603 110, India
[2]Department of Electronics and Communication Engineering,
Anna University Chennai, Chennai 600025, India

Abstract
Clusters and distributed systems offer fault tolerance and high performance through load sharing, and are thus attractive in real-time applications. When all computers are up and running, we would like the load to be evenly distributed among the computers. When one or more computers fail this must be redistributed. The redistribution is determined by the recovery scheme. The recovery scheme should keep the load as evenly distributed as possible even when the most unfavorable combinations of computers break down, i.e. we want to optimize the worst-case behavior. In this paper we compared all schemes (Modulo ruler, Golomb ruler, Greedy Sequence, Sloane Sequence, Log Sequence) with worst-case behavior. Finally we conclude our scheme (Sloane schemes) performs better than all the other schemes.

Key Words:
Fault tolerance, High performance computing, Cluster technique, Recovery schemes, Sloane sequence.

1. Introduction

One way of obtaining high availability and fault tolerance is to execute an application on a cluster or distributed system. There is a primary computer that executes the application under normal conditions and a secondary computer that takes over when the primary computer breaks down. There may also be a third computer that takes over when the primary and secondary computers are both down, and so on. Another advantage of using distributed system or cluster, besides fault tolerance, is load sharing between the computers. When all computers are up and running, We would like the load to be evenly distributed. The load on some computer will, however, increase when one or more computers are down, but under these condition we would like to distribute the load as evenly as possible on the remaining computers. The
distributed of the load when a computer goes
down is decided by the *recovery list* of the processes running on the fault computer.

The set of all recovery lists is referred to as the *recovery schemes*. Load balancing and availability or specially important in fault tolerant distributed system, where it is difficult to predict on which computer the process should be executed. This problem NP-complete for the large number of computers. Most cluster vendors support this kind of error recovery, e.g. the node list in Sun Cluster [13] the priority list in MC/Service Guard (HP) [4] the placement policy in TruCluster (DEC) [5], Cascading resource groups in HACMP (IBM) [4], and node preference list in Windows Server 2003 cluster (Microsoft, earlier called MSCS) [10].

We consider that the computers are connected in a ring in this work. A recovery scheme specifies where to transfer a process when the
computer in which it is running goes down. We call a transfer of a process from one computer to another as a jump. A jump is specified by a number that gives which computer to resume the process. The jump is the number of computers to pass by in the ring. Hence, the jumps are the same wherever in the ring we start. The jump is only dependent on the number of previous jumps of the process: i.e. on the number of transfers for the process. If a process is transferred from computer A to computer B, and also computer B is down, the next jump in the recovery scheme is used, counting from computer B. We use the term
"Wrap-around" when the total sum of jumps for a process exceeds the number of computers. We would like to do an optimal recovery process. Here optimal means that the maximal number of processes on the same computer after k crashes is BV (k) (Bound Vector). The function BV (k) provides a lower bound for any recovery scheme [9].

`Greedy', `Golomb' and `Modulo' schemes [6][7] are optimal for a larger number of computers than `Log'. The `Modulo' rule gives better optimal result for a larger number of computers down than the Golomb schemes and Greedy scheme. Both (Golomb and Greedy) recovery schemes consider the formulation where wrap-around is not taken into account whereas in this paper we use it as in `Modulo' scheme. In this paper we use a sharper mathematical formulation of the computer science problem, and give a new recovery schemes called Sloane scheme. These are

Manuscript received July 5, 2009
Manuscript revised July 20, 2009

optimal for a larger number of computers down in the original computer science problem. i.e. these results represent state-of-the-art in the field. The paper is organized as follows. In Section [3] we formulate and explain the problem. We briefly review the existing all work and general results in Section [4]. The General Theorem is defined in Section [5] and its performance in Section [6]. In Section [7] we conclude the paper..

2. Problem definition

We consider a cluster with *n* identical computers with one process on each computer. The work is evenly split between these *n* processes. There is a recovery list associated with each process. This list determines where the process should be restarted if the current computer breaks down. The set of all recovery lists is referred to as the recovery scheme. Figure [1] shows such a system for $n = 4$. We assume that processes are moved back as soon as a computer comes back up again. In most cluster systems this can be configured by the user [3][13][14], i.e. in some cases one may not want automatic relocation of processes when a faulty computer comes back up again. The left side of the figure shows the system under normal conditions. In this case, there is one process on each computer. The recovery lists are also shown; one list for each process. The set of all recovery lists is referred to as the recovery scheme.

The right side of Figure [1] shows the scenario when computer zero breaks down. The recovery list for process zero shows that it should be restarted on computer one when computer zero breaks down. If computer one also breaks down, process zero will be restarted on computer two, which is the second computer in the recovery list. The first

computer in the recovery list for process one is computer zero. However, since computer zero is down, process one will be restarted on computer three. Consequently, if computers zero and one are down, there are two processes on computer two (processes zero and two) and two processes on computer three (processes one and three). If computers zero and one break down the maximum load on each of the
remaining computers is twice the normal load. This is a good result, since the load is as evenly distributed as possible. However, if computers zero and two break down, there are three processes on computer one (processes zero, one and two), i.e. the maximum load on the most heavily loaded computer is three times the normal load. Consequently, for the recovery scheme in Figure [1], the combination of computers zero and two being down is more unfavorable than the combination of computers zero and one being down. We are interested in the worst-case behavior.

Our results are also valid when there are *n* external systems feeding data into the cluster, e.g. one telecommunication switching center feeding data into each computer in the cluster.
If a computer breaks down, the switching center must send its data to some other computer in the cluster, i.e. there has to be a
"recovery list" associated with each switching center. The fail-over order can alternatively be handled by recovery lists at the communication protocol level, e.g. IP takeover [11]. In that
case, redirecting the communication to another computer is transparent to the switching center. We assume that the work performed by each of the *n* computers must be moved as one atomic
unit. Examples are systems where all the work performed by a computer is generated from one external system or when all the work is performed by one process, or systems where the external communication is handled by IP takeover.

3. Optimal Recovery Schemes

Here we review previous works in which algorithms that give recovery schemes for a number of crashed computers. In [8] the problem of finding a recovery scheme that can guarantee optimal worst-case load distribution when at most *x* computers are down is presented for the first time. The schemes algorithm generates the recovery schemes that should have as large *k* as possible. The *Log* any static recovery scheme. BV is by definition increasing and contains exactly *k* entries that equals x for all $x \leq 2$. The j-th entry in the vector BV (*k*) equals $\lfloor \sqrt{2(j+1)} + 1/2 \rfloor$. Hence, BV (*k*) = < 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, . . .>.

The paper [6] presents two other algorithms, the *Greedy* algorithm and the *Golomb* scheme. These algorithms generate the recovery schemes that give better optimality than the *Log* algorithm [8], (i.e. better load balancing). The Greedy algorithm is based on the mathematical problem of finding the sequence of positive integers such that all sums of subsequences are unique and minimal. It is easy to calculate the Greedy algorithm even for large *n*. The Golomb scheme is a special case of the Greedy algorithm.

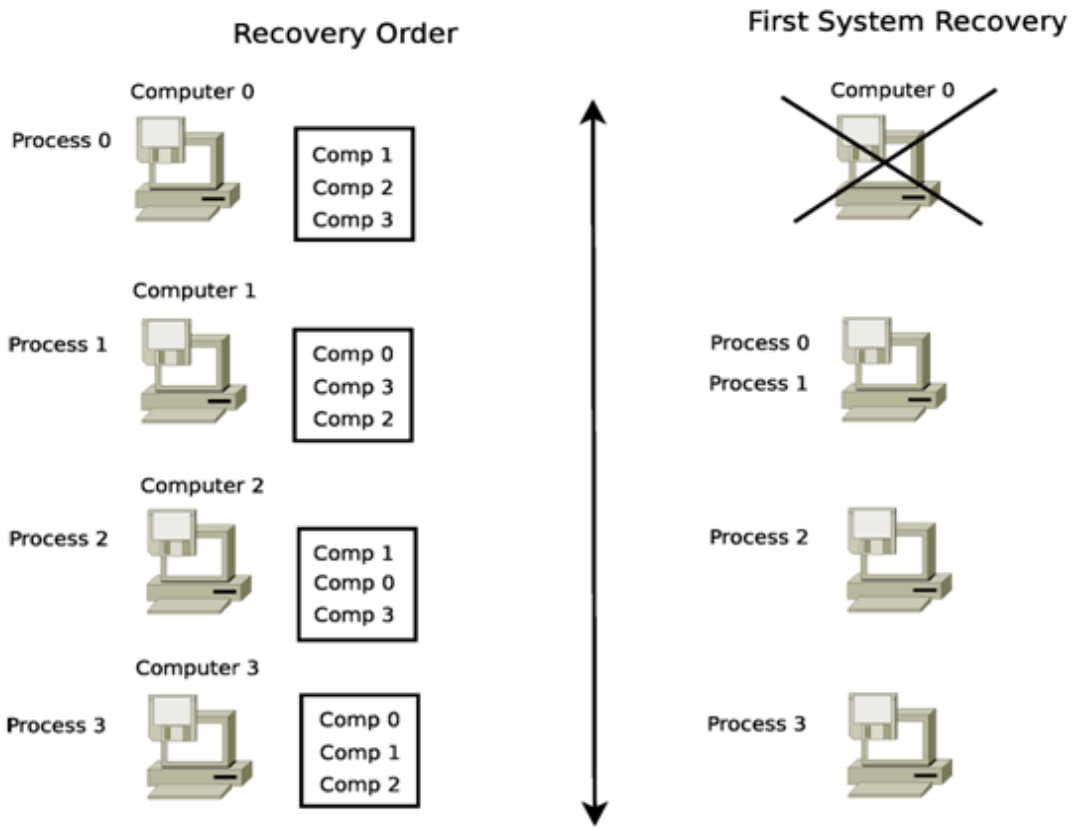

Figure 1 An application execution on a cluster with four computers.

However, finding (and proving) optimal Golomb schemes becomes exponentially more difficult as the number of computers (*n*) increases. Therefore, for large *n* one can easily calculate a sequence with distinct partial sums with the Greedy algorithm [14][16].

In [7] the problem is optimized by taking into account the *wrap-around* scenario: process being sent backwards or passing by the initial
guarantee optimality when at most $\lfloor \log_2 n \rfloor$ computers go down. Here optimal means that the maximal number of processes on the same

computer after *k* crashes is BV (*k*), where the function BV (*k*) provides a lower bound for
computer. This corresponds to the new mathematical problem of finding the longest sequence of positive integers for which the sum of all subsequences are unique modulo n. This mathematical formulation of the computer science problem gives new more powerful recovery schemes, called *modulo* schemes.

Our scheme [1] for the number of cluster computers n=140. "In worst case scenario our scheme gives better results".

In [8] the problem of finding recovery schemes for any number of crashed computers by an exhaustive search, where brute force testing is
avoided by a mathematical reformulation of the problem and a *branch-and-bound* algorithm. The search nevertheless has a high complexity. Optimal sequences and thus a corresponding optimal bound are presented for a maximum of twenty *one* computers in the distributed system or cluster.

4. General Proof

Theorem 1. VL(i) $\leq \lceil n/(n-i) \rceil$ VL(i) is entry number *i* in VL [9].

Proof: If *i* computers are down, there are *i* processes which must be allocated to the remaining *n-i* computers. The best one can hope for is obviously to obtain a load of

$\lceil n/(n-i) \rceil$ processes on the most heavily loaded computer. □

Theorem 2. BL *is consecutively smaller then than* VL [9].

Proof: Because of Theorem 1 we know that BV (n-1) < $n \leq$ VL (n-1).

Based on theorems 1 and 2, we define
B(i) = max (BV (i), $\lceil n/(n-i) \rceil$). Next we prove that our proposed scheme is optimal.

Theorem 3: The Sloane recovery scheme is optimal as long as x computes or less have crashed, where x = max(i), such that $R_0(i) < n$) be the heaviest loaded computer when x computers have crashed, where x = max(i), such that $R_0(i) < n$.[1]

Proof: When x computer have crashed, process z($0 \leq z < n$) will in the ith step end on computer $(z + \sum_{j=1}^{i} r(j))$ mod n($1 \leq i \leq x$). This means that a process ends up on computer y after i steps was originally allocated to computer $y - \sum_{j=1}^{i} r'(j) + n$ mod n:

1. $(y - \sum_{i=1}^{1} r'(i) + n)$ mod n.

2. $(y - \sum_{i=1}^{2} r'(i) + n)$ mod n,
   $(y - \sum_{i=2}^{2} r'(i) + n)$ mod n.

3. $(y - \sum_{i=1}^{3} r'(i) + n)$ mod n,
   $(y - \sum_{i=2}^{3} r'(i) + n)$ mod n.
   $(y - \sum_{i=3}^{3} r'(i) + n)$ mod n.

and so on. In general for x th

x: $(y - \sum_{i=1}^{x} r'(i) + n)$mod n,
$(y - \sum_{i=2}^{x} r'(i) + n)$ mod n,…,
$(y - \sum_{i=x}^{x} r'(i) + n)$ mod n.

5. Performance with All Schemes

In figure (2) compare Sloane schemes with all schemes. For example, in the case of n=100 computers in a cluster, modulo-m sequences guarantee optimal behavior in the case of 11 crashes, while Golomb rulers only guarantee optimality for 10 crashes, greedy scheme for 9
crashes and sloane scheme for 6 crashes. The advantage with the sloane and greedy algorithm compared to other schemes is that we can easily calculate a sequence with distinct partial sum.

Apart from this our scheme has the following advantages:

- Overall our scheme produced best results even under worst case scenario.
- When n>45 our scheme performs better than all the other schemes (see Figure (2) for n=100).

6. Conclusion

In many cluster and distributed systems, the designer must provide a recovery scheme. Such schemes define how the workload should be redistributed when one or more computers break down. The goal is to keep the load as evenly distributed as possible, even when the most unfavorable combinations of computers break down, i.e. we want to optimize the worst-case behavior which is particularly important in real-time systems. We consider n identical computers, which under normal conditions execute one process each. All processes perform the same amount of work. Recovery schemes that guarantee optimal worst-case load distribution, when x computers have crashed are referred to as optimal recovery schemes for the values n and x.

A contribution in this paper is that we have shown that the problem of finding optimal recovery schemes for a system with n computers corresponds to the mathematical problem of finding the longest sequence of positive integers such that the sum and the sums of all subsequences number are unique. No efficient algorithm that finds the longest have previously obtained recovery schemes that are optimal when a larger number of computers are down and they do or don't cover load balancing when wrap-around occurs. In this paper they don't consider. sequence with these properties is known. We cluster, modulo-m sequences guarantee.

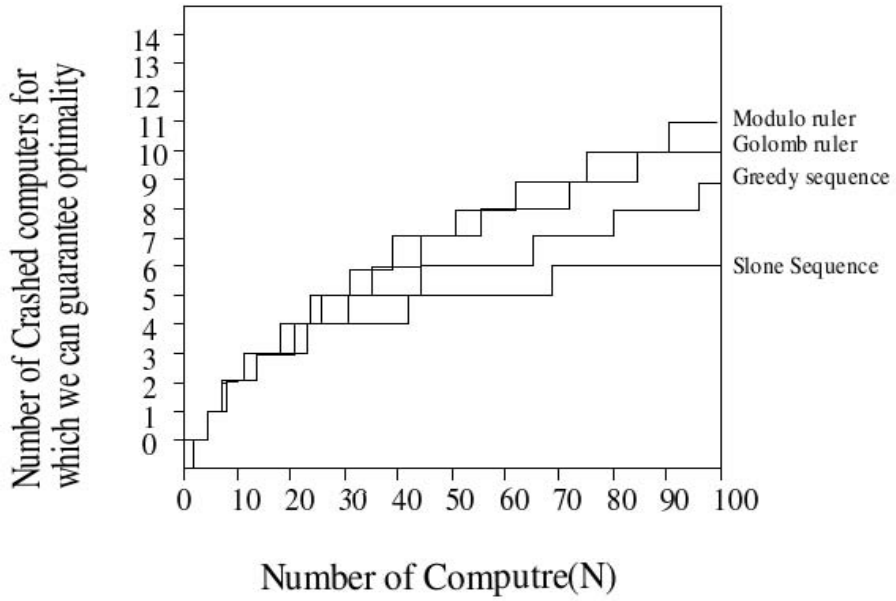

Figure 2. Performance with Sloane Schemes

In this paper we present the sloane sequences that minimize the maximum load. Modulo sequences allow optimal behavior for a larger number of crashed computers than Golomb, greedy and sloane sequences. For example, in the case of n=100 computers in a optimal behavior in the case of 11 crashes, while Golomb rulers only guarantee optimality for 10 crashes, greedy scheme for 9 rashes and sloane scheme for 6 crashes. Golomb rules are known or lengths upto 41912 (with 211 marks). Of these the first 373 (with 23 marks) are known to be optimal while modulo rulers, are known only for 13 marks. Modulo sequence are known up to 92 computers in the cluster and the Sloane schemes are known only up to 140 computers in a cluster.

Our recovery schemes can be immediately used in commercial cluster systems, e.g. when defining the list in Sun Cluster using the scconf command. The results can also be used when a number of external systems, e.g. telecommunication switching centers, send data to different nodes in a distributed system (or a cluster where the nodes have individual network addresses). In that case, the recovery lists are either implemented as alternative destinations in the external systems or at the communication protocol level.

References

[1] Delhi Babu, R and Karthigeyan, S, Using Sloane Rulers for optimal recovery schemes in distributed computing, Journal of Computer Science and Technology(under review).

[2] Delhi Babu, R and Karthigeyan, S, Optimal Recovery Schemes in Distributed Computing a Short Survey"- International Conference on Mathematics and Computer Science 2009 (ICMCS 2009).Vol 2.PP 458 to 461.

[3] Hewlett-Packard Company, TruCluster Server – Cluster Highly Available Applications, Hewlett-Packard Company, September 2002.

[4] Hewlett-Packard, Managing MC / ServiceGuard, Hewlett- Packard, March 2002.

[5] IBM, HACMP, Concepts and Facilities Guide, IBM, July 2002.

[6] Klonowska, K Lundberg L and Lennerstad H, Using Golomb Rulers for Optimal Recovery Schemes in Fault Tolerant Distributed Computing, In Proc. of the 17th International Parallel and Distributed Processing Symposium IPDPS 2003, Nice, France, April 2003, pp. 213.

[7] Klonowska, K., Lundberg, L., Lennerstad, H., Svahnberg, C. Using modulo rulers for optimal recovery schemes in distributed computing. 10th International Symposium PRDC 2004, Papeete, Tahiti, French Polynesia, March 2004, Proceedings, pp. 133-142

[8] Klonowska, K., Lundberg, L., Lennerstad, H., Svahnberg, C.Optimal recovery schemes in fault tolerant distributed computing. Acta infomatica 41.314-365,May 2005.

[9] Lundberg,L and Svahnberg,C, Optimal Recovery Schemes for High-Availability Cluster and Distributed Compute, JPDC, 2001

[10] Microsoft Corporation, Server Clusters: Architecture Overview for Windows Server 2003, Microsoft Corporation, March 2003.
[11] Pfister,G.F, In Search of Clusters, Prentice-Hall, 1998.
[12] Sloane,N.J.A, and Plouffe,S The encyclopaedia of interger sequence, Academic Press,1995.
[13] Sun Microsystems, Sun Cluster 3.0 Data Services Installation and Configuration Guide, Sun Microsystems, 2000.
[14] TruCluster, Systems Administration Guide, Digital Equipment Corporation,
[15] http://www.distributed.net/ogr/index.html
[16] http://www.research.ibm.com/people/s/shearer/grtab.html