

Reliability-aware Optimal K Node Allocation of Parallel Applications in Large Scale High Performance Computing Systems

Narasimha R. Gottumukkala^{1,2}, Chokchai Box Leangsuksun², Raja Nassar², Mihaela Paun^{2,3}, Dileep Sule², Stephen L. Scott⁴

¹Centre for Business and Information Technologies, University of Louisiana at Lafayette, Lafayette, LA 70504, USA

²College of Engineering & Science, Louisiana Tech University, Ruston, LA 71270, USA

³Faculty of Mathematics and Informatics, Spiru Haret University, Bucharest, Romania

⁴Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA

raju@louisiana.edu, {box, nassar, mpaun, sule}@latech.edu,scottsl@ornl.gov

Abstract

The growing demand for more computational power to solve complex scientific problems is driving the physical scale of the system to hundreds and thousands of nodes. Ideally, scaling up the number of nodes should minimize the completion time, but algorithmic and system environment factors limit the scalability of parallel applications. Reliability is one of the major factors that limit performance, especially when applications are scaled to thousands of nodes. In this paper, we propose a reliability-aware optimal k-node allocation algorithm and compare it with Round Robin and reliability-aware resource allocation algorithms. Our simulation results indicate that giving flexibility to the resource manager in choosing the optimal number of nodes for large scale parallel applications based on the reliability of compute nodes minimizes the overall completion time and waste time

1. Introduction

There are several factors based on the application scalability, and system environment that limit the performance of parallel applications. First, there are the algorithmic limitations such as the sequential fraction of code that cannot be parallelized, the overhead of synchronized computation from multiple processors, and the overhead of communication delay. Second, there are the system environment factors such as the processing speed, communication bandwidth, scheduling techniques, and software. Failures and downtimes become major performance hindrance factors for large scale parallel applications that span thousands and hundreds of thousands of nodes.

Minimizing the performance loss in the presence of failures is one of the major challenges of large scale HPC systems. Checkpointing and reliability-aware scheduling are well known approaches for minimizing the impact of failures.

With checkpointing, the application state is saved at regular intervals to avoid restarting the application in the event of a failure. Reliability-aware resource allocation aims to allocate parallel applications to the most reliable nodes in order to minimize the impact of failures. The performance considerations of a parallel application include scalability, efficiency or job completion time. In this paper, we concentrate on the performance aspect relevant to job run-length. Job run-length is the most important performance metric for either scalability, efficiency or performance [15]. Reliability is an important challenge for large scale computational applications. Also, failures and reliability are important resource allocation attributes [9]. However, reliability-aware resource allocation in the context of scalability has not been given much attention. Considering reliability as an important performance metric for resource managers in selecting the number of nodes implies that one could minimize waste time and overall completion times. In this paper, we study how reliability affects job completion time with the increase in number of nodes and propose a reliability-aware k node allocation algorithm based on the expected completion time of a parallel program.

The rest of the paper is organized as follows. Section 2 discusses related work, Section 3 discusses the expected completion time of a parallel application, the reliability-aware optimal k node selection algorithm and the comparison results with various other resource allocation algorithms for various types of jobs. Section 4 discusses the conclusions and future work.

2. Related Work

For a given parallel architecture and problem size, the application speedup cannot continue to increase with the increase in the number of processors and saturates after a certain limit. Various aspects of scalability have been studied

in [2][3][4][14]. Amdahl's law [2] suggests that there is a serial part of the program that limits scalability. But according to Gustafson [17] there are other parameters in computation that can be overlapped while executing the serial portion. A survey of scalability models for parallel architectures and algorithms for a given parallel architecture and problem size is given by Kumar et al.[14]. The optimal selection of the number of processors for a numerical approximation problem and architecture has been discussed in [15]. The effect of reliability for large scale parallel applications is not new and has also been addressed recently [18][19]. The effect of reliability on the completion time of parallel programs is discussed in [10], and the effect of coordinated checkpointing on large scale parallel applications due to failures is discussed in [6][19]. Reliability-aware resource allocation of parallel applications using the reliability of nodes and workload properties is observed to minimize the waste time due to failures. Plank [1] discusses the importance of considering the number of processors as an important performance attribute for checkpointing. In this paper, we propose a reliability-aware optimal k node selection algorithm to minimize the completion time and waste time.

3. Reliability-aware Resource Allocation

In order to study the effect of reliability in selecting the optimal number of k nodes, we consider reliability and job completion time as an important metrics for space sharing jobs. In this section, we discuss existing resource allocation algorithms, and propose an optimal k node allocation algorithm based on the expected completion time of a parallel application. We compare the proposed optimal k node allocation algorithm with other resource allocation algorithms by simulating the resource allocation of parallel jobs on the failure data generated based on the failure properties of individual nodes from the ASCI White system.

3.1 Expected Completion time of a Parallel Program

The Expected Completion time of a parallel program running on k nodes is given by [24]

$$E(T_{c(k)}) = T_{c(k)} + (W + R) \left[\frac{F_k}{1 - F_k} \right] \quad (1)$$

Where, $T_{c(k)}$ is the estimated time of parallel application on k nodes. It is hard to analytically estimate the running time on k nodes. Therefore,

we may use the scalability models namely Amdahl's law and Gustafson's law for our study. W is the waste time due to failures, R the repair time and F_k the system failure probability, where

$$F_k = 1 - R_k, \text{ where } R_k = \prod_{i=1}^k R_i(t_i + x | t_i) \quad (2)$$

Here, in Equation (2), R_k is the system reliability, R_i is the reliability of node i, t_i is the failure free running time of node i, and x is the estimated job running time on k nodes.

3.2 Resource Allocation algorithms

The main objective of the resource allocation algorithm in the presence of failures is to select k nodes that minimize the waste time due to failures, and overall completion time of a parallel application. We consider the following resource allocation algorithms

1. All Nodes (ALL)

This technique selects all the available nodes in the system. In the absence of failures, selecting all the nodes should give the minimum completion time as this technique utilizes all the nodes. If any node fails before the job is completed, it has to be resubmitted (i.e job is restarted from the beginning) once the node is up again.

2. Round Robin Allocation (RR)

The Round Robin allocation technique allocates the job to k adjacent nodes based on the rotation policy of node ID's starting from the first node ID. When the last node number is reached, nodes are allocated beginning from the first node ID. In this technique, the k number of nodes required for the parallel application is fixed and given by the user. The RR policy does not take into account the node reliabilities or job run-length before allocating the job.

3. Reliability-Aware Allocation (RA)

Here, the k number of nodes for a job is given by the user, and the k value is fixed. The algorithm selects the k most reliable nodes available for every job. Reliability-aware resource allocation has been proposed in [9] and found to reduce the overall waste time. This policy considers the individual node reliabilities, and the k number of nodes required for the job is fixed by the user. The reliabilities are calculated using a Weibull distribution and the system reliability for k nodes is given in Equation (2).

For simulation purposes, we randomly generate the workloads with the number of nodes required by the user's applications. Further discussion on workload and failure data is provided in Section 3.4.

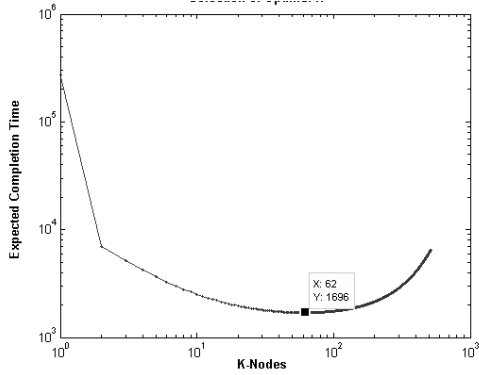


Figure 1(a)

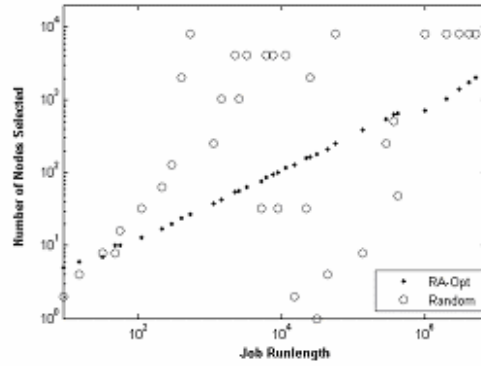


Figure 1(b)

Figure 1. Optimal k node selection. Figure 1(a) shows the optimal k nodes at the point where the expected completion time is minimal. Figure 1(b) shows the comparison of optimal k node selection with k selected randomly

3.3 Optimal K Node allocation

In an HPC system, the reliability of each individual node is calculated based on the failure parameters obtained from the failure history of the nodes. Each node may have different reliabilities and an optimal k node allocation algorithm considers the following three key factors

(1) *Number of Processors:* Increasing the number of processors reduces the overall completion time. On the other hand, increasing the number of nodes increases failure probability requiring resubmission of jobs (i.e. job is restarted from the beginning) which increases completion times and waste times.

(2) *Job Run-length:* A longer job has a higher chance of encountering failures as compared to a shorter job. Therefore, for a given set of nodes a longer job may have more waste time as compared to a shorter job.

(3) *Reliability of k node:* The reliability of a selected node affects the chances that the node will fail in the future. Also, failures increase the waste time and completion time of a job.

The scheduling problem may be defined as follows:

“Given a parallel application, and an HPC system that contains m nodes $n_1, n_2, n_3 \dots n_m$ with reliabilities $R_1, R_2, R_3 \dots R_m$, find k out of m nodes such that the overall completion time is minimized”

The RA-Optimal algorithm selects k nodes out of m nodes such that the expected completion time is minimal.

Algorithm: Optimal K-Node allocation

1. $k=0$ // The number of nodes selected
- $N[]$ //contains all the node id's
- J //contains the job runlength
2. for $i=1:\max(\text{size}(N))$ nodes
3. calculate $R(i)$ //Equation (2)
4. end
5. $M[] = \text{sortdesc}(N,R)$ //sort the nodes based on descending order //of reliabilities
6. $KNList.add(M(1));$ //add the node with highest reliability
7. while $E(T_{opt}, D) \leq E(T_{opt}, D)$ //Equation (1)
8. $k=k+1$
9. $KNList.add(M(k+1))$
10. end while
11. $opt_k=k$
12. allocate application to $KNList$ nodes
- allocate (k) Nodes to the Application

Figure 1(a) shows an example case, where the number of k nodes is selected based on the minimum expected completion time. The expected completion time (X-axis) versus the number of nodes (Y-axis) is plotted and the minimum expected completion time and the corresponding k nodes ($K=62$) are shown in Figure 1(a). Also, in Figure 1(b) we show the selection of the number of nodes (Y-axis) based on job run lengths (X-axis) when the number of nodes are chosen randomly and with the RA-Optimal algorithm. We observe in Figure 1(b) that the k node selection increases with the increase in the job run lengths. The effectiveness of various resource allocation techniques can be justified only when compared with completion times and waste times. The simulation results for various resource allocation techniques are discussed in Section 3.5.

3.4 Simulation Framework

The system failure logs and parallel job workloads are inputs to the simulator. Each job has a *job id*, *job run-length*, and the *number of*

nodes required for the job. The failure logs have *node ids*, *failure times*, *down times*, and *reliability* of the nodes. We simulate a 10,000 node system using the failure properties of compute nodes obtained from ASCI White system logs. The failure data and time to failure distributions of White are discussed in further detail in [9]. The failure data has the failure (Wall clock times when the nodes have failed), and the down time (time needed to become operational).

The processing times of each node are identical; however, the reliability of individual nodes may differ. We generate the failure data for the 10,000 system by using the ASCI White failure properties. In this study we are interested in a large scale system. However, the parallel workloads available at the website [21] are not suitable for our purpose. Therefore, a synthetic workload was generated using the distribution of the number of processors and the job run-lengths given in [11]. We use the uniform-log distribution to generate the number of processors, and two stage hyper exponential distribution to generated job run-lengths [11][22][23]. In addition to the actual workload, we also injected some jobs with very long run-lengths to test the effectiveness of our technique. We consider the following performance metrics for our study:

Mean Completion Time (MCT) is the ratio of the total completion time to the unit job run-length (unit job run-length = job-run-length/number of processors).

Mean waste time (MWT) is the ratio of the total waste time to the unit job run-length.

Relative Percentage Difference
 $RPD(=100 * \left(\frac{T_1 - T_0}{T_0} \right))$, where T_0 is the

performance metric for the most optimal technique and T_1 is the performance metric for one of the three compared techniques (RR, RA or ALL). That positive value of percentage difference gives the percentage improvement of T_0 over T_1 and a negative value indicates the percentage improvement of T_1 over T_0 .

3.5 Experimental Results

We observe that MWT and MCT times are affected by the number of nodes, the job run-length and reliability of the selected k nodes. The All-nodes technique selects all the nodes which, in an ideal case should reduce the job completion time but increase the system failure probability.

Since failures happen multiple times, it is seen from Figure 2(a) that MCT is highest for the RR-technique. Figure 2(c) shows that the MWT is highest for the All-nodes technique. For the ALL technique, jobs fail more often contributing to waste time, however jobs also complete faster because we are using all the nodes. Therefore, we observe in Figure 2(a) that the MCT is lower for ALL technique as compared to RR, whereas in Figure 2(c) the MWT is higher for ALL technique than for RR.

The RR technique allocates nodes based on Round-Robin policy. Since the reliability of nodes is not considered, the MCT is higher for RR as compared to other techniques. The RA technique allocates the most reliable nodes, but the number of nodes is fixed by the user similar to RR. Therefore, although the RA technique performs better than RR and All-nodes techniques, it does not perform as well as the RA-Optimal technique. Figure 2(b) shows the percent improvement of the RA-optimal technique relative to the other techniques. We can observe that the MCT of RA-Optimal is 84.09% better than ALL 177.04% better than RR and 71.70% better than RA algorithms. Figure 2(d) shows the percentage difference of MWT when RA- Optimal is compared to other techniques. We observe that the MWT of RA-Optimal is 306.39% better than ALL, 156.64% better than RR, and 44.3% better than the RA technique. It is also important to compare the performance metrics with respect to job run-length and understand how well the algorithm performs with respect to job run-length.

4 Conclusions and Future Work

Increasing the number of nodes in HPC systems for solving ultra scale computational problems decreases reliability, presenting new challenges in resource management. In this paper, we propose reliability-aware optimal k node allocation algorithm based on the expected completion time and reliability of a system of k nodes. Our simulation results indicate that long jobs can especially benefit from the reliability-aware optimal k node allocation algorithm that steers away from failures, thereby minimizing the completion time and waste time for jobs.

This work has several scopes for improvement. First, the reliability-aware optimal k-node allocation can be combined with various

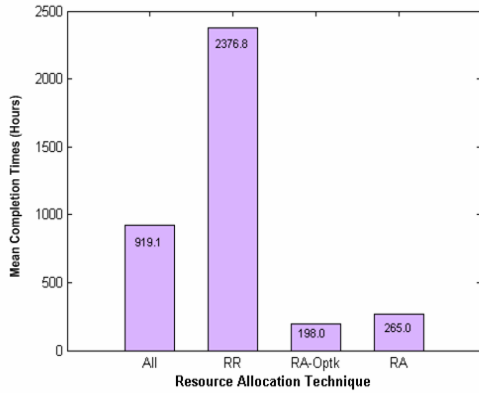


Figure 2(a)

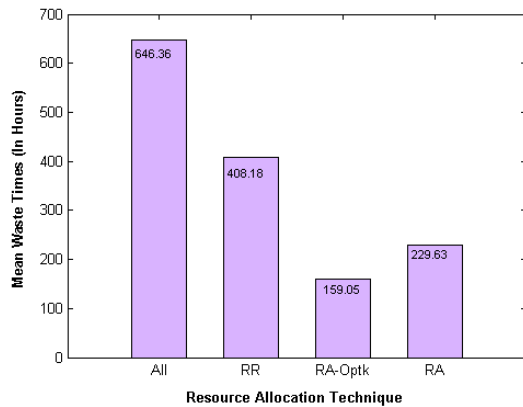


Figure 2(c)

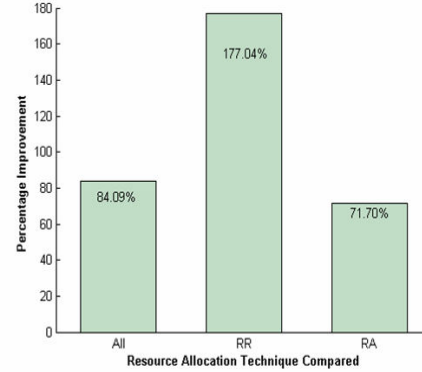


Figure 2(b)

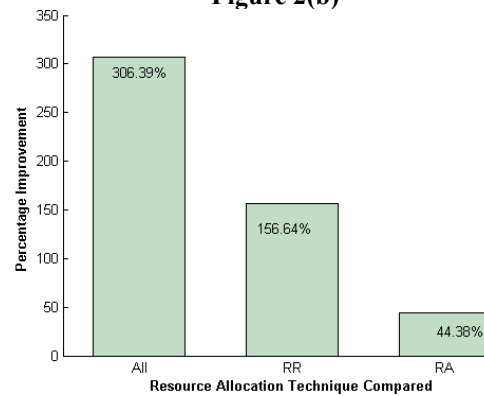


Figure 2(d)

Figure 2. Comparison of MCT and MWT, Figure 2(a) shows the MCT of the three techniques and Figure 2(b) shows the corresponding RPD. Figure 2(c) shows the MWT of each technique and Figure 2(d) shows the corresponding improvement of RA- Optimal over other techniques.

scheduling algorithms and checkpointing for further optimization to minimize the overall completion times.

This work is confined to malleable jobs (i.e jobs for which the number of nodes can be determined dynamically by the scheduler). However, it may not be the case for all types of jobs because it is hard to scale each and every application. The results would be more convincing if the reliability-aware algorithm is applied to an HPC system that has both the workload and failure data.

6. Acknowledgements

¹ This research is supported by the Department of Energy Grant no:DE-FG02-05ER25659, and

⁴ by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science,

U. S. Department of Energy, under contract No. DE-AC05- 00OR22725 with UT-Battelle, LLC.

5. References

- [1] James S. Plank and Michael G. Thomason, "The Average Availability of Parallel Checkpointing Systems and Its Importance in Selecting Runtime Parameters," *29th International Symposium on Fault-Tolerant Computing, Madison, WI, June, 1999*, pp. 250-259.
- [2] G. M. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proceedings of the AFZPS*, vol. 30, 1967, pp. 483-485.
- [3] N. J. Davies. "The Performance and Scalability of Parallel Systems", PhD thesis, Department of Computer Science, Faculty of Engineering, University of Bristol, UK, Dec. 1994.
- [4] D. Eager, J. Zahorjan, and E. Lazowska. "Speedup versus Efficiency in Parallel Systems", *IEEE Transactions on Computers*, 38(3):408 -- 423, March 1989.

- [5] Lawrence Livermore National Laboratory Trace Logs: [url: http://www.llnl.gov/asci/platforms/white/](http://www.llnl.gov/asci/platforms/white/)
- [6] Elmootazbellah N. Elnozahy, James S. Plank. "Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery," *IEEE Transactions on Dependable and Secure Computing*, vol. 01, no. 2, pp. 97-108, April-June, 2004.
- [7] Worlton, J. 1993. "Toward a taxonomy of performance metrics", In *Computer Benchmarks*, J. J. Dongarra and W. Gentsch, Eds. Elsevier Advances In Parallel Computing Series, vol. 8. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 3-22.
- [8] Yudan Liu; Leangsuksun, C.B.; Hertong Song; Scott, S.L., "Reliability-aware Checkpoint/Restart Scheme: A Performability Trade-off," *Cluster Computing, 2005. IEEE International*, vol., no., pp.1-8, Sept. 2005.
- [9] Narasimha Raju, Gottumukkala, Chokchai Leangsuksun, Raja Nassar, Stephen L Scott. "Reliability-Aware Resource Allocation in HPC Systems", *Proceedings of the IEEE International Conference on Cluster Computing 2007*, Austin Texas.
- [10] N. R. Gottumukkala, C. Leangsuksun, and S. L. Scott. "Reliability-aware approach to improve job completion time for large-scale parallel applications". In *Proceedings of 2nd Workshop on High Performance Computing Reliability Issues (HPCRI) 2006*, Austin, TX, USA, February 11-15, 2006.
- [11] Uri Lublin and Dror G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs". *Journal of Parallel & Distributed Computing*. 63(11), pp. 1105-1122, Nov 2003.
- [12] Vipin Kumar and Anshul Gupta. "Analyzing scalability of parallel algorithms and architectures". *Journal of Parallel and Distributed Computing*, 22(3):379--391, 1994.
- [13] Oliner, A.J.; Sahoo, R.K.; Moreira, J.E.; Gupta, M.; Sivasubramaniam, A., "Fault-aware job scheduling for BlueGene/L systems" *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, vol., no., pp. 64-, 26-30 April 2004.
- [14] Kumar, V. and Gupta, A. 1991. "Analysis of scalability of parallel algorithms and architectures: a survey". In *Proceedings of the 5th international Conference on Supercomputing (Cologne, West Germany, June 17 - 21, 1991)*. E. S. Davidson and F. Hossfield, Eds. ICS '91. ACM, New York, NY, 396-405.
- [15] Nicol, D. M. and Willard, F. H. 1988. "Problem size, parallel architecture, and optimal speedup." *Journal of Parallel & Distributed Computing*. 5, 4 (Aug. 1988), 404-420.
- [16] Sartaj Sahni, Venkat Thanvantri, "Performance Metrics: Keeping the Focus on Runtime," *IEEE Parallel and Distributed Technology*, vol. 04, no. 1, pp. 43-56, Spring, 1996.
- [17] Gustafson, J. L. 1988. "Reevaluating Amdahl's law". *Communications of the ACM* 31, 5 (May. 1988), 532-533.
- [18] Reed, D. A., Lu, C., and Mendes, C. L. 2006. "Reliability challenges in large systems". *Future Generation Computing. Systems*. 22, 3 (Feb. 2006), 293-302.
- [19] J.S. Plank, W.R. Elwasif, "Experimental Assessment of Workstation Failures and Their Impact on Checkpointing Systems," *FTCS*, p. 48, 1998
- [20] B. Schroeder and G.A Gibson, 2006. "A large-scale study of failures in high-performance computing systems". In *Proceedings of the international Conference on Dependable Systems and Networks*, June 2006.
- [21] D. G. Feitelson. Parallel workloads archive. <http://cs.huji.ac.il/labs/parallel/workload/index.html>, 2001.
- [22] A.B. Downey, "A Parallel Workload Model and its Implications for Processor Allocation," In *Proceedings of the 6th IEEE international Symposium on High Performance Distributed Computing*, High Performance Distributed Computing. IEEE Computer Society, Washington, DC, August 1997.
- [23] D.G. Feitelson, "Packing Schemes for Gang Scheduling," In *Proceedings of the Workshop on Job Scheduling Strategies For Parallel Processing* D. G. Feitelson and L. Rudolph, Eds. Lecture Notes In Computer Science, vol. 1162. Springer-Verlag, London, pp. 89-110, 1996.
- [24] Narasimha Raju, Gottumukkala, "Failure Analysis and Reliability Aware Resource Allocation of Parallel Applications in High Performance Computing Systems", PhD thesis, Department of Computer Science and Computational Analysis and Modeling, Louisiana Tech University, Ruston, LA February 2008.