In *Proc. 2nd Symp. Frontiers Massively Parallel Computation* (1988), pp. 491–494.

FAULT TOLERANCE OF ALLOCATION SCHEMES IN MASSIVELY PARALLEL COMPUTERS

Marilynn Livingston[*]
Department of Computer Science
Southern Illinois University
Edwardsville, IL 62026-1653

Quentin F. Stout[†]
Dept. of Elec. Eng. and Comp. Sci.
University of Michigan
Ann Arbor, MI 48109-2122

Abstract

This paper examines the problem of locating and allocating large fault-free subsystems in multiuser massively parallel computer systems. Since the allocation schemes used in such large systems cannot allocate all possible subsystems a reduction in fault tolerance is experienced. We analyze the effect of different allocation methods including the buddy and Gray-coded buddy schemes for the allocation of subsystems in the hypercube and in the 2-dimensional mesh and torus. Both worst case and expected case performance is studied. Generalizing the buddy and Gray-coded systems, we introduce a new family of allocation schemes which exhibits a significant improvement in fault tolerance over the existing schemes and which uses relatively few additional resources. For purposes of comparison, we study the behavior of the various schemes on the allocation of subsystems of 2^{18} processors in the hypercube, mesh, and torus consisting of 2^{20} processors. Our methods involve a combination of analytic techniques and simulation.

Keywords fault tolerance, allocation, hypercube computer, mesh, torus, buddy system.

1 Introduction

Parallel computers incorporating thousands of processors must be able to tolerate faulty processors and communication links if they are to achieve a usable mean-time-to-failure. In these large systems, processor allocation is needed for both multiuser environments, such as is provided with the NCUBE series of hypercubes, and for single user systems with multiple subtasking capabilities. In such a computing environment the problem of locating and allocating large fault-free subsystems is computation-intensive and, as a consequence, in practice some allocation scheme which recognizes only a subset of the existing subsystems is used. The allocation scheme often has a dramatic effect on the fault tolerance of the system, thus forcing a trade-off to be made between space and computation time devoted to the allocation scheme versus minimum acceptable level of fault tolerance of the system.

In this paper we examine allocation schemes for large subcubes of a hypercube and large subsquares of a two-dimensional mesh and torus, considering worst case and expected case fault tolerance of the interconnection network, and the reduction in fault tolerance caused by the fact that the allocation scheme used cannot allocate all subsystems. We focus on the allocation of large subsystems because we believe that most massively parallel machines are purchased in order to support large tasks as opposed to hundreds of simultaneous users with small tasks. For comparative purposes we analyze allocating a subcube or subsquare of 2^{18} processors in a machine of 2^{20} processors.

In a d-dimensional hypercube there are $\binom{d}{q}2^{d-q}$ subcubes of dimension q, called *q-subcubes*, and each processor belongs to $\binom{d}{q}$ of them. Thus in a hypercube of 2^{20} processors a faulty processor makes 190 of the existing 760 18-subcubes faulty. The smallest number of faulty processors which makes all subcubes containing m processors faulty in a hypercube containing n processors is denoted by $Q_w(n,m)$. Analogously, the expected number of processor faults which makes all subcubes of size m faulty is denoted by $Q_e(n,m)$, where we assume that faults are independent and uniformly distributed. We find that $Q_e(n,m)$ is significantly larger than $Q_w(n,m)$ and, in particular, $Q_w(2^{20},2^{18}) = 8$ while $Q_e(2^{20},2^{18}) \approx 24.5$. A discussion of these functions and their analogs for the 2-dimensional mesh and torus is included in Section 2.

Current hypercube allocation schemes do not allocate all q-subcubes but instead employ some form of the *buddy system* approach, where the only q-subcubes allocated are those

[*]Partially supported by National Science Foundation grant CCR-8808839
[†]Partially supported by National Science Foundation grant DCR-8507851 and an Incentives for Excellence Award from Digital Equipment Corporation

consisting of all processors determined by arbitrarily fixing the high-order $d-q$ address bits. There are only 2^{d-q} such q-subcubes and each processor is in exactly one of them. Using $B_w(n,m)$ and $B_e(n,m)$ to denote the worst case and expected case number of faults needed to make all the buddy system subcubes of dimension $\lfloor \lg m \rfloor$ faulty, one has $B_w(2^{20}, 2^{18}) = 4$ and $B_e(2^{20}, 2^{18}) \approx 8.3$. Thus the use of the buddy system allocation scheme results in a considerable reduction in fault tolerance. In Section 3 we discuss the fault tolerance properties of the buddy system along with several variants including a *double buddy system* (DB), a *Gray-coded buddy system* (G), and a *double Gray-coded buddy system* (DG) defined for the hypercube, and the 2-dimensional mesh and torus.

In Section 3.3 we introduce a new family of allocation schemes which generalizes both the buddy and Gray-coded buddy systems and exhibits improved fault tolerance with relatively little increase in search time. The best of these, denoted DA^2, allocates only 48 subcubes of dimension 18 in a 20-dimensional hypercube, and yet $DA^2_e(2^{20}, 2^{18}) \approx 15$. Thus DA^2 achieves more than half of the fault tolerance of the hypercube in which all 760 subcubes of dimension 18 are allocable.

For purposes of comparison, we include in Section 4 simulation results for the hypercube with 2^{20} processors and the various schemes for allocation of subsystems possessing 2^{18} processors.

2 Subsystems

Throughout, $\lg$ means $\log_2$. We use n to denote the total number of processors in the system, and for convenience of notation, suppose n is an even power of two. Let $\mathcal{M}(n)$ denote a two-dimensional square mesh with grid points $\{(x,y) : 1 \le x, y \le \sqrt{n}\}$, where two grid points are connected if and only if their coordinates differ by one in exactly one coordinate position. The corresponding $\sqrt{n} \times \sqrt{n}$ two-dimensional torus will be denoted by $\mathcal{T}(n)$. It has the same grid points as $\mathcal{M}(n)$, and includes the connections of $\mathcal{M}(n)$, but in addition its boundary points $(x,1)$ and $(x,\sqrt{n})$ are adjacent as are $(1,y)$ and $(\sqrt{n},y)$ for $1 \le x,y \le \sqrt{n}$. We will denote by $\mathcal{Q}(n)$ the d-dimensional hypercube with $n = 2^d$ nodes which are the binary d-tuples and where two d-tuples are connected if and only if they differ in exactly one position.

The mesh $\mathcal{M}(n)$ has $(\sqrt{n} - \sqrt{m} + 1)^2$ subsquares of size $\sqrt{m} \times \sqrt{m}$, the torus $\mathcal{T}(n)$ has n such subsquares, while the hypercube $\mathcal{Q}(n)$ has $\binom{d}{q} \cdot 2^{d-q}$ subcubes of dimension q, where $d = \lg n$. Thus we see that both $\mathcal{M}(n)$ and $\mathcal{T}(n)$ have $\Theta(n^{1.5})$ square subsystems whereas $\mathcal{Q}(n)$ has $\Theta(3^d) = \Theta(n^{\lg 3})$ subcubes.

Let us now examine the worst case and expected case fault tolerance when all subsystems of a given size are allocable. Denote by $M_w(n,m)$ the smallest number of faulty processors which make all subsquares of $\mathcal{M}(n)$ with m processors faulty, and let $M_e(n,m)$ denote the *expected* number of faulty processors which must occur before all subsquares with m processors are faulty, assuming that faults are independent and uniformly distributed. The expressions $T_w(n,m)$ and $T_e(n,m)$ denote the corresponding quantities for the torus $\mathcal{T}(n)$ and $Q_w(n,m)$ and $Q_e(n,m)$ denote the corresponding quantities for the hypercube $\mathcal{Q}(n)$.

It is straightforward to establish that the functions $M(n,m)$, $T(n,m)$, and $Q(n,m)$ are monotone non-decreasing functions of n and monotone non-increasing functions of m. We state these results without proof in the following.

Proposition 2.1 *If $n' > n$, $d' > d$, $m' > m$, and $n \ge m'$ then*

(*i*) $M_w(n',m) \ge M_w(n,m)$ *and* $M_w(n,m') \le M_w(n,m)$,

(*ii*) $M_e(n',m) > M_e(n,m)$ *and* $M_e(n,m') < M_e(n,m)$,

(*iii*) $T_w(n',m) \ge T_w(n,m)$ *and* $T_w(n,m') \le T_w(n,m)$,

(*iv*) $T_e(n',m) > T_e(n,m)$ *and* $T_e(n,m') < T_e(n,m)$,

(*v*) $Q_w(d',m) \ge Q_w(d,m)$ *and* $Q_w(d,m') \le Q_w(d,m)$,

(*vi*) $Q_e(d',m) > Q_e(d,m)$, *and* $Q_e(d,m') < Q_e(d,m)$.

□

The values of $M_w(n,m)$ and $T_w(n,m)$ are relatively easy to determine. To illustrate for $m = n/4$, subdivide the $\sqrt{n} \times \sqrt{n}$ mesh into four $\sqrt{n/4} \times \sqrt{n/4}$ submeshes, designate a faulty processor p_0 in one of the submeshes and designate its translate as faulty in each of the other submeshes. This results in every square submesh of size $n/4$ being faulty, and the same fault pattern also makes every square submesh of the torus $\mathcal{T}(n)$ faulty. On the other hand, since the 4 subsquares are nonoverlapping, at least 4 faulty processors are required to cause them to be faulty. Using this reasoning, it follows that

$$M_w(n, n/4^i) = T_w(n, n/4^i) = 4^i$$

for any natural number i.

In the case of the hypercube the function $Q_w(n,m)$ is much more difficult to compute. However, it is known [1, 2] that $Q_w(n, n/4)$ is the minimum positive integer r such that $\binom{r-1}{\lfloor r/2 \rfloor - 1} \ge \lg n$ which yields

$$Q_w(n, n/4) = 8.$$

Thus, while in a square mesh or torus of 1024×1024 processors, as few as 4 faulty processors can make every 512×512 submesh faulty, in a 20-dimensional hypercube consisting of the same number of processors, at least 8 processors must become faulty before all 18-dimensional subcubes become faulty. As the number of processors increases, this difference becomes more pronounced since it has been shown [1, 2] that

$$Q_w(n, n/4) = \lg\lg n + \frac{1}{2}\lg\lg\lg n + O(1).$$

We used simulation to investigate the expected number of faulty processors that make the analogous subsystems faulty. The 1024×1024 mesh and torus was approximated by the continuous unit square, and the 512×512 submesh by a continuous $\frac{1}{2} \times \frac{1}{2}$ square. In each trial, faults were successively generated randomly and uniformly in the unit square until no fault-free subsquare of size $\frac{1}{2} \times \frac{1}{2}$ remained. The results of 100,000 trials yielded the following:

$$M_e(2^{20}, 2^{18}) \approx 13.54, \text{ and } T_e(2^{20}, 2^{18}) \approx 19.89 \,.$$

For the 20-dimensional hypercube we used a simulation with 10,000 trials and found that

$$Q_e(2^{20}, 2^{18}) \approx 24.50.$$

The program developed for this simulation, which will be described in Section 4, can also produce mean-time-to-failure values by incorporating a given probability distribution for the faults.

3 Allocation Schemes

In comparison with the 2-dimensional mesh and torus, the hypercube displays a high degree of fault tolerance with respect to large subsystems. However, this advantage is lost when only a small subset of the existing subsystems are allocable. In this section we will describe a few analytic results that indicate how the buddy system, and other allocation schemes, affect the fault tolerance properties of the hypercube, mesh, and torus.

3.1 Allocation Schemes for Hypercubes

Most allocation schemes for the hypercube employ the *buddy system* approach, where the only q-subcubes allocated in $\mathcal{Q}(2^d)$ are those of the form $a_1a_2\ldots a_{d-q} * \ldots *$, that is, the high-order $d-q$ address bits are fixed in each allocable q-subcube. There are 2^{d-q} of these subcubes and they form a partition of $\mathcal{Q}(2^d)$. We will use $BQ_w(n,m)$ and $BQ_e(n,m)$ to denote the worst case and expected case number, respectively, of faults needed to make all the buddy system subcubes of size m faulty.

To obtain a good upper bound for $BQ_e(n,m)$, we consider the following model: balls are tossed at random into x identical boxes until each box contains at least one ball. Suppressing details, we can show that the expected number of balls required is $\sum_{j=0}^{x-1} \frac{x}{x-j}$. The balls correspond to faults, and the boxes represent the disjoint subcubes allocated by the buddy system. Thus $x = n/m$. The number of balls required is a slight overestimate of the number of faults required because a subcube with one or more faults is slightly less likely to acquire another fault. From this model we have the following.

Theorem 3.1 *For $n \geq m \geq 2$ and both n and m even powers of 2,*

(*i*) $BQ_w(n,m) = n/m,$

(*ii*) $BQ_e(n, n/2) \leq 3$ *and* $\lim_{d\to\infty} BQ_e(n,n/2) = 3,$

(*iii*) $BQ_e(n,m) \leq \frac{n}{m}\left[\ln 2 \,\lg(\frac{n}{m}) + O(1)\right].$

□

Several allocation schemes which are more fault tolerant than the buddy system have been proposed. For example, two "orthogonal" buddy systems, which we call *double buddy systems* (DB), can be used to allocate m-subcubes of the form $a_1a_2\ldots a_{d-m} * \ldots *$ and $* \ldots * \, a_{m+1}\ldots a_d$. With twice the number of subcubes allocated and roughly twice the overhead in the allocation algorithm we shall see an increase of approximately 25% in fault tolerance. In general, multiple buddy systems use multiple permutations of the index set $\{1, 2, \ldots, d\}$, and for each permutation π they allocate m-subcubes by fixing index positions $\pi(1)\ldots\pi(d-m)$ and varying the remaining m indices.

Another class of allocation schemes for the hypercube involve the use of the Gray code numbering of the nodes of $\mathcal{Q}(n)$. Let g_d denote the binary reflected Gray code map from $\{0\ldots 2^d-1\}$ to d-bit strings. A single *Gray-coded buddy system*, denoted by G here, allocates q-subcubes that arise as pairs of $(q-1)$-subcubes of the form $\{a_1\ldots a_{d-q+1} * \ldots *, b_1\ldots b_{d-q+1} * \ldots *\}$, where $g^{-1}_{d-q+1}(a_1\ldots a_{d-q+1})$ and $g^{-1}_{d-q+1}(b_1\ldots b_{d-q+1})$ are consecutive mod 2^{d-q+1}. An approximate analysis of the behavior of the Gray-coded buddy system can be obtained through the use of a model in which x identical boxes are arranged in a ring and balls are tossed at random into the boxes until no two adjacent boxes are empty. Here $x = 2^{d-q+1}$, which is the number of q-subcubes allocable by scheme G. While it is straightforward to show that $\lim_{d\to\infty} GQ_e(2^d, 2^{d-k})$ is trapped between $2^k k \ln 2$ and $2^{k+1}(k\ln 2 + O(1))$, we can obtain a more exact description. Let $p(x,i)$ denote the conditional probability that there are two adjacent empty boxes, given that i balls have been tossed at random into the boxes. We have evaluated $p(x,i)$ which, in turn, gives an explicit expression for the limiting behavior of $GQ_e(n,m)$ as stated in the following.

Theorem 3.2 *For $n \geq m \geq 2$ and both n and m even powers of 2,* (*i*) $GQ_w(n, n/4) = n/4$,

(*ii*) $GQ_e(n, n/2) \leq 3\frac{2}{3}$ *and* $\lim_{n\to\infty} GQ_e(n, n/2) = 3\frac{2}{3}$,

(*iii*) $GQ_e(n,m) \leq \sum_{j=0}^{x-1} p(x,i)\frac{x}{x-j}$, *where* $x = n/m$ *and* $p(x,i)\binom{x}{i} = \binom{x}{i} - \binom{i}{x-i} - \binom{i-1}{x-i-1}$, *and* $\lim_{n\to\infty} GQ_e(n, n/x) = \sum_{j=0}^{x-1} p(x,i)\frac{x}{x-j}$.

□

Multiple Gray-coded buddy systems combine Gray codes with the multiple index permutations of multiple buddy systems. We will use DG to denote the double Gray-coded buddy system which allocates q-subcubes from pairs of $(q-1)$-subcubes in which the first $d-q+1$ index positions have been fixed together with the pairs of $(q-1)$-subcubes in which the last $d-q+1$ index positions have been fixed. Chen and Shin [4] have suggested DG as an improved allocation scheme for $\mathcal{Q}(n)$.

To obtain corresponding analytic results for DB and DG, we can use the same model but the analysis involves the consideration of many special cases. We have resorted to simulation to help us understand the performance of these schemes and will report the results in Section 4.

3.2 Allocation Schemes for the Mesh and Torus

A buddy system for the allocation of square submeshes containing m nodes in $\mathcal{M}(n)$ is analogous to that described for hypercubes. There are n/m subsquares of dimensions $\sqrt{m} \times \sqrt{m}$ allocated and these are of the form $\{(x,y) : (j-1)\sqrt{m}+1 \leq x \leq j\sqrt{m}, (k-1)\sqrt{m}+1 \leq y \leq k\sqrt{m}\}$ for $1 \leq j,k \leq \sqrt{n/m}$. So, for example, if $m = n/4$, there are only 4 allocable square submeshes containing $n/4$ processors. In the worst case we see that only 4 faulty processors are needed to make every buddy system submesh of size $n/4$ faulty. Assuming the faults are randomly and uniformly distributed, the expected number of faulty processors needed to make every buddy system submesh of size $n/4$ faulty can be found by considering the same model as that used for the buddy system for hypercubes. This gives an expected value of approximately $8\frac{1}{3}$. The same arguments hold for the torus, as we state in the following.

Theorem 3.3 *For $n \geq m \geq 2$, and both n and m even powers of 2,*

(*i*) $BM_w(n,m) = BT_w(n,m) = n/m$,

(*ii*) $BM_e(n,m) = BT_e(n,m) \leq \frac{n}{m}\left[\ln(\frac{n}{m}) + O(1)\right]$.

□

The Gray-coded buddy system for $\mathcal{M}(n)$ and $\mathcal{T}(n)$ is the same as the double buddy system DB here. $\mathcal{M}(n)$ is partitioned into square submeshes consisting of $m/4$ nodes each and G allocates a square submesh of size m when that submesh arises as a 2×2 array of the smaller $m/4$ submeshes in the partition. Thus G allocates a total of $(2\sqrt{n/m}-1)^2$ m-node square submeshes for $\mathcal{M}(n)$ and a total of $4n/m$ for $\mathcal{T}(n)$.

When n/m is an integral power of 4 the buddy systems are already worst-case optimally fault tolerant for the mesh and torus, and hence also $GM_w(n,m) = GT_w(n,m) = n/m$. To obtain analytic estimates of $GM_e(n,m)$ and $GT_e(n,m)$, the same probability model as used in the case of the hypercube can be utilized here. However, the number of configurations to handle becomes large and we have not carried the computations out but rather have resorted to simulation studies instead. In the study, we approximated $\mathcal{M}(n)$ and $\mathcal{T}(n)$ by the continuous unit square, and the $\sqrt{n/4} \times \sqrt{n/4}$ submesh by a continuous $\frac{1}{2}\times\frac{1}{2}$ square as we described in Section 2. The results of 100,000 trials gave

$$GM_e(n, n/4) \approx 10.0 \text{ and } GT_e(n, n/4) \approx 11.7.$$

3.3 A New Family of Allocation Schemes

We will describe here a new family of allocation schemes for the hypercube and note that an analogous family can be described for the 2-dimensional mesh and torus.

Let $k \geq 1$, and consider an allocation scheme A^k that, for a given d and q, will allocate q-subcubes whose nodes are d-tuples in which the last $q-k$ bits are arbitrary and the first $d-q+k$ bits are the nodes of a k-subcube in $\mathcal{Q}(2^{d-q+k})$. For example, A^2 will allocate 18-subcubes in $\mathcal{Q}(2^{20})$ of the form $a_1a_2a_3a_4 * \ldots *$ in which the *last* 16 components are $*$'s and where two of the a_i have values 0 and 1 and the other two a_i are equal to $*$. Thus, A^2 allocates 24 subcubes of dimension 18. The *double* A^2 family, denoted by DA^2, allocates a set of q-subcubes of $\mathcal{Q}(n)$ which consist of the set of q-subcubes allocated by A^2 together with a corresponding set in which the *first* 16 components are $*$'s and the last four components are chosen in an analogous way to the first four components for A^2. In general, A^k allocates $\binom{d-q+k}{k}2^{d-q}$ subcubes of dimension q in $\mathcal{Q}(2^d)$ and DA^2 allocates twice this number. In the families A^k and DA^k, increasing k clearly increases the number of allocable subcubes and hence increases the fault tolerance, at a cost of increased search time. Analytic results for these families, similar to those for the buddy and Gray-coded buddy systems has not been done as yet. However, simulation studies we have done on these schemes show them to yield a significant improvement in the fault tolerance of the hypercube as we will see in the next section.

	E	B	DB	G	DG	A^1	DA^1	A^2	DA^2
S_a	760	4	8	8	16	12	24	24	48
Q_w	8	4	4	4	4	4	4	5	5
Q_e	24.6	8.1	10.1	9.8	11.9	10.7	13.0	12.8	15.4

Table 1: Expected and Worst Case Behavior of Allocation Schemes

4 Simulation Results

We illustrate in Table 1 the results of simulation studies of the buddy (D), double buddy (DB), Gray-coded buddy (G), double Gray-coded buddy (DG), the new allocation families (A^1) and (A^2), and their doubles (DA^1) and (DA^2). We include the results for the scheme E in which every subcube is allocable, as well as including the worst case results and a listing for the number S_a of subcubes allocated by each scheme. These schemes were used in the allocation of subcubes in $\mathcal{Q}(2^{20})$ of size 2^{18}. The values for Q_e shown in the table were obtained from 1000 trials. In each trial, a list of random faults sufficient to make every 18-subcube faulty was generated. For each allocation scheme X, the list was scanned to identify the first fault on the list that resulted in all of the 18-subcubes allocated by X being made faulty. Although we have not done so, we could easily modify our program to compute mean-time-to failure for each of the allocation schemes once a probability distribution for faults was specified. The expected case values in the table are based on the assumption that faults are randomly and uniformly distributed with respect to the nodes of the hypercube $\mathcal{Q}(2^{20})$.

5 Conclusion

Our results show that simple allocation schemes such as those based on the buddy system lose much of the fault tolerance of the system, but that much of this loss can be regained by a more sophisticated allocation scheme such as the new family DA^k that we described. Results of our simulations indicate, for example, that by using the allocation scheme DA^2 on the hypercube of dimension 20, we can roughly double the fault tolerance of that provided by the buddy system. Moreover, very little overhead is involved in the implementation of DA^2. As a final note, we observe that the hypercube is significantly more fault tolerant than the mesh or torus, and that the use of the buddy system for large subcube allocation reduces the fault tolerance to that of the mesh.

References

[1] B. Becker and H. Simon, “How robust is the n-cube?”, *Proc. 27th Annual IEEE Symp. on Foundations of Comp. Sci.* (1986), pp. 283–291.

[2] N. Graham, F. Harary, M. Livingston, and Q. F. Stout, “Subcube fault tolerance in hypercubes”, *Information and Computation* **102** (1993), pp. 280–314.

[3] M. Livingston and Q. F. Stout, “Distributing resources in hypercube computers”, *Proc. Third Conf. on Hypercube Concurrent Computers and Applications, Pasadena, CA* (1988), pp. 222–231.

[4] M.-S. Shen and K. Shin, “Processor allocation in an n-cube multiprocessor using gray codes”, *IEEE Trans. Computers* **C-36** (1987), pp. 1396–1407.