Fault Tolerant Hierarchical Configuration of Mesh Networks

Jason Zurawski
Department of Computer Science, Montclair State University
Upper Montclair, NJ 07043, USA

Abstract

In an effort to improve the overall performance of distributed mesh systems, a hierarchical approach of dividing the processing units into logical levels has been proposed [7]. The scheme has been tested and shown to achieve substantial performance improvement in many situations. This hierarchical configuration divides the mesh into progressively smaller clusters, thus allowing for computation to occur in small local groups at the lower levels. After local operations, the results are passed to higher logical levels. Under certain assumptions, this method has been shown to produce an overall minimal total communication cost for the entire system [7].

As the distributed mesh systems attempt to achieve optimal performance with this hierarchical configuration, questions arise concerning the system's ability to handle node failure. When using a hierarchical configuration, certain elements in the mesh become more important to the overall system than others. It is important that the hierarchical system have a re-organizing mechanism in the case of node(s) failing to function, in such a way that the performance gain from hierarchical configuration is salvaged as much as possible.

The fault-tolerance problems considered in this project are concerned with minimizing the loss of performance in the system hierarchy due to the presence of failing nodes. We will propose schemes/algorithms for that purpose. The performance results will be compared to that of an ideal, fault-free system. The first phase of the project will be to design strategies to reconstruct the hierarchy, accommodating to the situation that some nodes in the original hierarchy are not functioning anymore. To that end, new local heads may be selected and local nodes regrouped. In the second phase of the project, simulation experiments will be developed to examine the effectiveness of the proposed schemes. Examples of both faulty and fault-free hierarchical mesh systems will be tested to quantify how good the proposed schemes are.

Index Terms — Distributed Processing, Fault tolerance, Hierarchical control, Interconnection networks, Mesh.

Contents

1 Introduction

In previous works, a hierarchical configuration for mesh was developed [7]. The proposed scheme divides a mesh network into uniform smaller clusters. Each of these clusters contains a leader to communicate with the other members of the group. Leaders are then required to communicate with other leaders to form groups at higher levels. It was shown that this method reduced the communication cost of distributed monitoring by minimizing the overall distance traveled by messages in the network[1][2][4][5][7].

Various experiments were conducted on the hierarchical configuration to simulate the different activities that it may be used for. Simulations were designed to test various sizes of the underlying mesh, as well as potential cluster sizes that may be utilized. In efforts to see if additional improvements could be made, a variety of throughputs of data were tested for the system.

Based on the previous work, questions have been raised regarding the robustness of such a hierarchically configured system. When certain processing units are chosen as leaders, they become very important to the overall health of the system. If a leader were to experience a failure, the localized cluster data may fail to reach the upper levels of the hierarchy. Failures at higher levels are even more dangerous to the effectiveness of the network system.

This M.S. Thesis research project explores possible techniques and algorithms to help a hierarchically configured mesh deal with failure of nodes. It will show that even though failures have occurred, it is still possible to achieve an overall good performance through the use of certain heuristic approaches. Individual processing unit locations will be examined to determine their overall impact to clusters as well as the system as a whole.

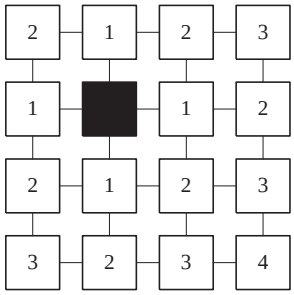
Two algorithms are proposed to deal with failure of cluster leaders. The first algorithm shifts the focus of individual clusters as needed, thus minimizing the cost within each of the clusters. The second algorithm involves shifting the focus of the entire configuration to better suited nodes, thus minimizing the costs at higher levels. These algorithms will be evaluated for speed, accuracy, and optimality through various empirical tests.

The rest of this thesis is organized as follows. Section 2 contains some of the mathematical background used throughout the paper. This background will be used extensively in section 3. In Section 3 processing unit failure will be formally defined and examined. Different strategic locations within the mesh will be presented, as well as techniques for dealing with their failure. A single level mesh will be examined first, to illustrate some basic concepts; this idea will then be extended to that of an optimal two-level hierarchy. Section 4 will present two different algorithms designed to help deal with the failure of processing units within a hierarchical mesh. Section 5 will analyze the empirical results given by the testing of these two algorithms. Different situations will be simulated to determine the strengths and weaknesses of these two approaches. Section 6 will summarize the work accomplished in this paper, and draw conclusions based on the findings.

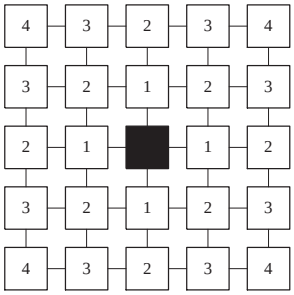
2 Preliminaries and previous works

Before dealing with the happenstance of faults within a mesh it is necessary to describe the concept of mesh in general, as well as the basics for a hierarchical configuration. We think of a square mesh in terms of size through the usage of the variable N. When $N = 5$, we

have a total mesh size of $N \times N$ with a total of 25 nodes. The same can be said for an even N. As shown in Figure 1a, an even mesh does not have a true central node. It is necessary to choose a node from the "pseudo center" area in the middle of the mesh. An odd mesh, as shown in Figure 1b, has a true central node.



(a) Even mesh of size $N = 4$



(b) Odd mesh of size $N = 5$

Figure 1: (a) A 4×4 mesh. The monitor node is at the "pseudo center" (dark). (b) A 5×5 mesh. The node at the center is the monitor (dark). The values in each node indicate the Hamming distance to the monitor.

2.1 Single level mesh

The monitor cost for a single level of mesh is understood to be the total cost of communication for the system. All nodes must interact with the monitor. The communication cost for each node is the Hamming distance from itself to the monitor. This communication cost is demonstrated in Figure 1.

To minimize the monitoring cost for a single level of mesh it is necessary to place the monitor in a centralized location. With a centralized monitor, we are guaranteed an optimal system in comparison with that of a non-centralized monitor.

As calculated in [7], there are two different formulas that can be used to describe the total communication cost of a $N \times N$ mesh using a central monitor:

$$C_{1_{orig}}(N) = \begin{cases} \frac{N^3 - N}{2}, & N \text{ odd} \\ \\ \frac{N^3}{2}, & N \text{ even} \end{cases} \tag{1}$$

It can easily be shown that by using any other node as the monitor would produce a system costing more. It is possible to improve on this optimal performance through the usage of a logical hierarchy within the mesh. Because this hierarchy is logical, additional connections or nodes are not required.

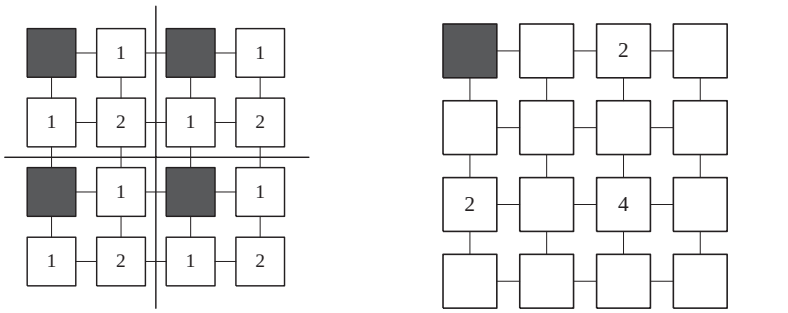
2.2 Hierarchical two level mesh

In a two-level partitioning, the entire mesh is divided into several identical submeshes. Each submesh has a local monitor which will join a group at a higher level as well as manage the

nodes at the lower level. A centralized node should act as a monitor [1][2][4][7].

The purpose of hierarchical monitoring is to reduce the overall system cost incurred by monitoring. Assuming two-level hierarchy, we need to find out the best way to divide the mesh so that the total communication cost is minimum. It has been shown that when choosing a sub mesh size μ such that μ is the closest integer to $\sqrt[3]{2N}$ that divides N evenly, we can achieve the minimum communication cost for the system [7].

At Level-one we divide the entire mesh into clusters of size $\mu \times \mu$. Each cluster will have a centralized node as a monitor. The monitor is responsible for collecting data in the submesh, and turning it in to a leader at the higher level. At Level-two each of the Level-one monitors will form a group of their own; a central monitor will be choosen for this level as well.



(a)Lower level of hierarchy **(b)Upper level of hierarchy**

Figure 2: A $N = 4$ mesh divided into clusters of size $\mu = 2$, the monitor nodes are in black. (a) There are $\left(\frac{N}{\mu}\right)^2$ clusters on the lower level. (b) The upper level is a single cluster of size $\left(\frac{N}{\mu}\right)^2$ by itself.

In order to calculate the total cost of the mesh we first divide the mesh into clusters of size $\mu \times \mu$ where μ is an integer and it divides N evenly. There are a total of $\left(\frac{N}{\mu}\right)^2$ clusters on the lower level to be included in the calculation. At the second level, note that all local monitors form a squared mesh by themselves (the black nodes in Figure 2a). So choosing the central or near-central node among them as the monitor (the darkest node in Figure 2b) will give the minimum communication cost. However, the cost of "one step" (i.e., passage of data from a node to its immediate neighbor) is μ instead of 1. The total cost for a hierarchical mesh system according to [7] is:

$$C_{2_{orig}}(N,\mu) = \begin{cases} = \frac{N^2\mu^3 - N\mu^2 - N^2\mu + N^3}{2\mu^2}, & \mu \text{ odd, } \frac{N}{\mu} \text{ odd} \\ = \frac{N^2\mu^3 - N^2\mu + N^3}{2\mu^2}, & \mu \text{ odd, } \frac{N}{\mu} \text{ even} \\ = \frac{N^2\mu^3 - N\mu^2 + N^3}{2\mu^2}, & \mu \text{ even, } \frac{N}{\mu} \text{ odd} \\ = \frac{N^2\mu^3 + N^3}{2\mu^2}, & \mu \text{ even, } \frac{N}{\mu} \text{ even} \end{cases} \quad (2)$$

Now that we have seen an ideal system, we are prepared to deal with the failure of node(s). These basic calculations will help describe the behavior of a faulty hierarchical mesh system.

3 Processing unit failure within mesh

In a true mesh there exist two distinct types of nodes. Leaf nodes exist on the edge of the mesh and can have either two or three connections to their neighbors. "Corner" nodes and "border" nodes are examples of this type of node. Internal nodes exist within the borders of the mesh, and have four connections to neighboring nodes. The three node instances are shown in Figure 3. These two different varieties of processing units will produce different results when they experience failure.

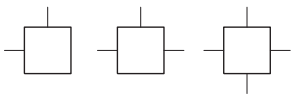


Figure 3: The three types of nodes within a system: corner nodes, edge nodes, and internal nodes.

If a failure occurs to a leaf node it will not burden the system with additional processing cost, it can be simply thought of as being removed from the system. If an internal node fails it may either be removed from the system, or cause the system to incur additional processing by extending a potential path to the monitor. In the event that several nodes of close proximity fail, a limited number of paths to the monitor could remain. A situation like this could easily drive the total system cost well beyond acceptable levels.

A special case of internal nodes is a level monitor. If a level monitor experiences a failure it becomes necessary to elect a new one. Electing the new monitor can be generalized into two cases within a square mesh; having an N size that is even or odd. These situations are examined in the context of a single level mesh. These ideas will then be extended to a hierarchically configured system of two levels.

3.1 Monitor node failure within a single level mesh

As described earlier, a single level of mesh strives to use a central monitor to minimize the communication cost. It is necessary to treat an even and odd mesh differently due to the

design of the "central area". Because we treat the two mesh sizes differently we will see that level costs are not the same.

The ultimate goal is to keep the cost of a faulty mesh similar to that of the ideal mesh. When moving the monitor it is possible to either gain or lose cost, depending on the position of the new monitor. The strategies described below aim to maintain an optimal middle choice in order to minimize the cost.

3.1.1 N is even

When N is even a true central node does not exist. Any one of the four nodes in the central "area" of the mesh may be picked as the monitor. Because of this property, the top left node is usually assumed to be the standard choice for simplistic reasons. When the monitor fails, we should choose one of the three remaining nodes from the acceptable region to be the new monitor.

There are two distinct paths we can take at this point; choosing the node diagonal to the failed node, or choosing the nodes directly below or next to the failed node. The two choices are not at all equal; the first choice will benefit the system while the second choice causes the mesh to incur additional processing cost.

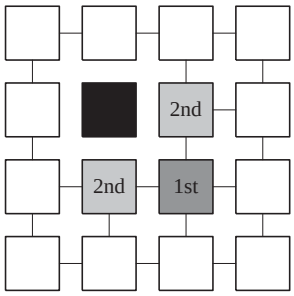


Figure 4: The "pseudo center" area in an even mesh allows for two possible categories of choices when a failure occurs.

The diagonal node is considered to be the best choice for our new monitor; choosing it will yield a net loss of 2 for the communication cost. This operation is shown in Figure 5. The communication cost assignment has not changed from that of an ideal system, with the exception of having to remove the single failed node. The diagonal choice is prized highly because it is able to retain at most four active connections. The second choices offer at most a section of 3 active connections.

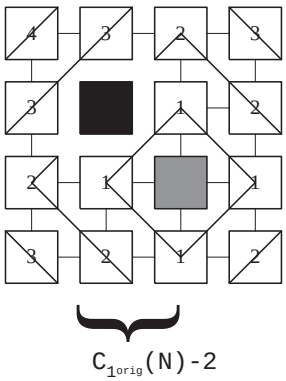


Figure 5: Cost breakdown for an even mesh of $N = 4$, when we choose the diagonal node in the event of a failure.

It is possible to utilize the original formula, Equation 1, for calculating the cost of an even mesh. The total cost for choosing the diagonal node, denoted as $C_{e_1}(N)$, is calculated as follows.

$$\begin{aligned} C_{e_1}(N) &= C_{e_{orig}}(N) - 2 \\ &= \frac{N^3}{2} - 2 \\ &= \frac{N^3 - 4}{2} \end{aligned}$$

Figure 5 contains a breakdown of cost by node position. The black node has experienced a failure causing the grey node to be named as the new monitor for the level. The concentric diamonds indicate the area of "relative cost". The only difference between this system and that of an ideal system is the loss of the old monitor.

If the diagonal node is not available it becomes necessary to pursue other options. The nodes directly below or to the right of the failed node will always produce the same result; a total gain of $N - 3$ to the system. The reason for the gain is due to having to pick a new "route" around the failed node(s). As shown in Figure 6, the concentric diamonds take on a new form. The failed node now sits between the new monitor and all other nodes above it vertically. All of these nodes will experience a gain of 2, and the system will lose 1, due to the failed node.

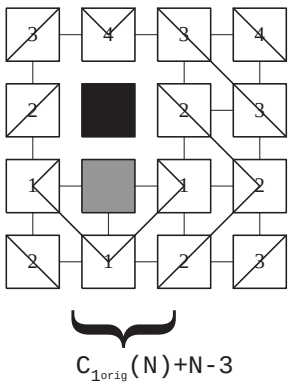


Figure 6: Cost breakdown for an even mesh of $N = 4$, when we choose the nodes next to, or below the failed node.

The mesh will experience a net gain of $2 \times (\frac{N}{2} - 1)$ as well as the loss of 1 when we choose this class of replacement. The new formula can be created using Equation 1 again and is denoted as $C_{e_2}(N)$:

$$\begin{aligned} C_{e_2}(N) &= C_{e_{orig}}(N) + 2\left(\frac{N}{2} - 1\right) - 1 \\ &= \frac{N^3}{2} + N - 3 \\ &= \frac{N^3 + 2N - 6}{2} \end{aligned}$$

To summarize the preceding discussion, the total communication cost $C(N)$ for an $N \times N$ mesh using a modified central monitor, when N is even is

$$C_e(N) = \begin{cases} \frac{N^3-4}{2}, & N \text{ even}_1 \\ \frac{N^3+2N-6}{2}, & N \text{ even}_2 \end{cases} \tag{3}$$

When moving the monitor in an even mesh we have seen that it is important to attempt to utilize the diagonal node first. It will be shown that the odd mesh is no different in its treatment of the diagonal node.

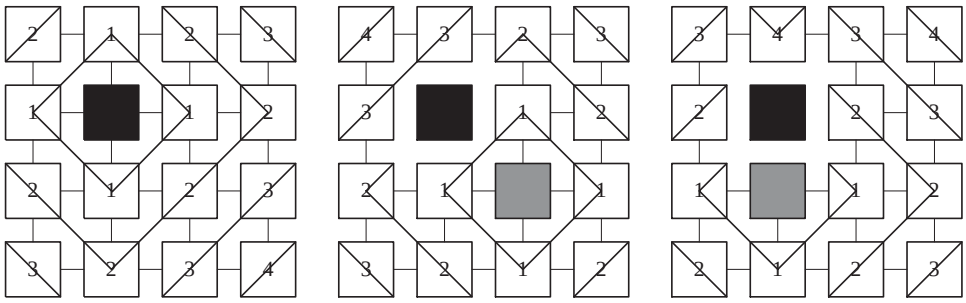


Figure 7: Three of the possible outcomes for cost analysis in even mesh. The ideal is on the far left, while the two failure cases are on the right.

3.1.2 N is odd

When N is odd, there exists a true central processing unit to serve as the monitoring node. If this node fails, we have more choices available for a replacement. Any of the nodes that surround the failed node can be used as a replacement monitor, yielding a net gain of $N + (N - 2)$ for the entire system. We experience the gain due to having to find new routes around the failed node(s). Although any of the nodes can be picked to get the same result, there are two distinct categories to choose from; nodes to the diagonal of the failed node, and nodes next to the failed node. We will first examine the diagonal nodes case, due to its straight forward calculation as well as its previous reputation as being a good choice.

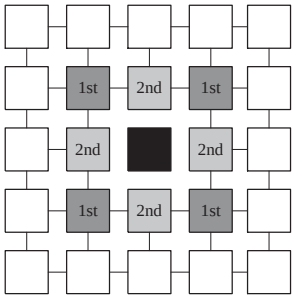


Figure 8: The "true center" of an odd mesh allows for two possible categories of choices when a failure occurs. Any of the choices will produce the same final result.

First it is necessary to split the odd mesh into an even mesh of size $N - 1$, and then add an extra "gray area" to the previous total as seen in Figure 9. The concentric diamonds that surround the monitor are un-broken, but still additional cost is incurred by the system due to the extra distances added by the "gray area" nodes.

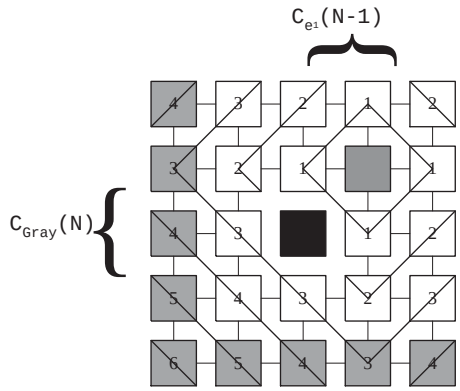


Figure 9: Cost breakdown for an odd mesh of $N = 5$, when we choose the diagonal node in the event of a failure.

We are able to use the $C_{e_1}(N-1)$ formula that is a part of Equation 3, which was calculated above. The formula, $C_{o_1}(N)$, can be calculated as follows.

$$C_{o_1}(N) = C_{e_1}(N-1) + C_{Gray}(N)$$

$$\begin{aligned} C_{Gray}(N) &= (1 \cdot (N+1)) + \left(2 \cdot \frac{N+1}{2}\right) + (2 \cdot N) + \left(4 \sum_{i=\frac{N-1}{2}-1}^{N-1} i\right) \\ &= 4N + 2 + \frac{3N^2 - 8N - 3}{2} \\ &= \frac{8N}{2} + \frac{4}{2} + \frac{3N^2 - 8N - 3}{2} \\ &= \frac{3N^2 + 1}{2} \end{aligned}$$

$$\begin{aligned} C_{o_1}(N) &= C_{e_1}(N-1) + C_{Gray}(N) \\ &= \left(\frac{(N-1)^3 - 4}{2}\right) + \frac{3N^2 + 1}{2} \\ &= \left(\frac{N^3 - 3N^2 + 3N - 5}{2}\right) + \left(\frac{3N^2 + 1}{2}\right) \\ &= \frac{N^3 + 3N - 4}{2} \end{aligned}$$

When choosing the nodes that lie in a straight line from the failed node, we use a similar calculation to that of $C_{o_1}(N)$. In this formula, we need to use $C_{e_2}(N-1)$ from Equation 3 to serve as the odd mesh, as well as add in another "gray area" that surrounds it, as shown in Figure 10.

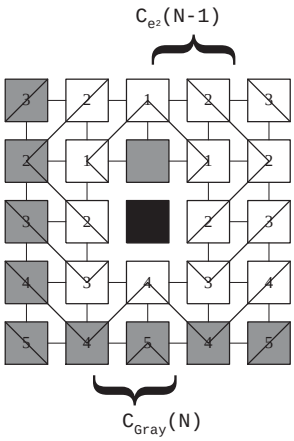


Figure 10: Cost breakdown for a odd mesh of $N = 5$, when we any of the other nodes in the event of a failure.

The equation can be described as follows:

$$C_{o_2}(N) = C_{e_2}(N-1) + C_{Gray}(N)$$

$$\begin{aligned} C_{Gray}(N) &= (N-1) + \left(\frac{N-1}{2}\right) + (2 \cdot N) + \left(\frac{N+1}{2} + 2\right) + \left(\sum_{i=\frac{N+1}{2}+1}^{N-1} 2i - 1\right) \\ &= \frac{8N+2}{2} + \frac{3N^2 - 10N + 3}{2} \\ &= \frac{3N^2 - 2N + 5}{2} \end{aligned}$$

$$\begin{aligned} C_{o_2}(N) &= C_{e_2}(N-1) + C_{Gray}(N) \\ &= \left(\frac{(N-1)^3 + 2(N-1) - 6}{2}\right) + \left(\frac{3N^2 - 2N + 5}{2}\right) \\ &= \left(\frac{N^3 - 3N^2 + 5N - 9}{2}\right) + \left(\frac{3N^2 - 2N + 5}{2}\right) \\ &= \frac{N^3 + 3N - 4}{2} \end{aligned}$$

To summarize the preceding discussion, the total communication cost $C(N)$ for an $N \times N$ mesh using a modified central monitor, when N is odd is

$$C_o(N) = \frac{N^3 + 3N - 4}{2} \tag{4}$$

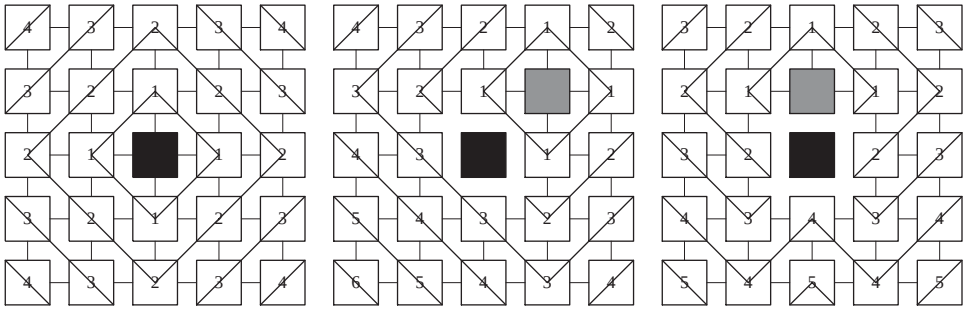


Figure 11: Three of the possible outcomes for cost analysis in odd mesh. The ideal situation is pictured on the far left, the two situations of failure are on the right.

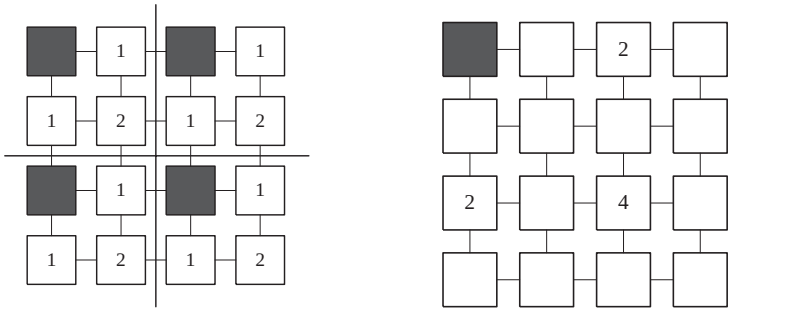
As shown in the even and odd calculations, the nodes that are diagonal to a failed node are the most sought after. This is due to the fact they have more active connections (i.e., a maximum of 4) than that of the other nodes (i.e., a maximum of 3). Future algorithms will attempt to take advantage of this property.

3.2 Hierarchically configured faulty mesh

Performing a hierarchical configuration in a faulty system is similar to that of an ideal system; the exception being that we must use the new strategies described above. All of the calculations for a faulty mesh are based on the calculations of [7].

The idea, as stated previously, is to divide the entire mesh into identical submeshes. The submeshes will each contain a monitor, and each monitor will be a member of the upper level. The upper level will have a monitor of its own.

We again choose a sub mesh size of μ, so we can achieve the minimum communication cost within the system [7].



(a)Lower level of hierarchy **(b)Upper level of hierarchy**

Figure 12: A $N = 4$ mesh having cluster size of $\mu = 2$, the monitor nodes are in black. (a) There are $\left(\frac{N}{\mu}\right)^2$ clusters on the lower level. (b) The upper level is a single cluster of size $\left(\frac{N}{\mu}\right)^2$ by itself.

Let the sub mesh be of dimension $\mu \times \mu$, so that μ is an integer and it divides N. Then by equations 3 and 4 the cost for local monitoring will be:

$$C(\mu) = \begin{cases} \frac{\mu^3+3\mu-4}{2}, & \mu \text{ odd} \\ \frac{\mu^3-4}{2}, & \mu \text{ even}_1 \\ \frac{\mu^3+2\mu-6}{2}, & \mu \text{ even}_2 \end{cases}$$

Therefore the total cost for all $(\frac{N}{\mu})^2$ level-one sub meshes is calculated by applying using equations 3 and 4:

$$C_I(N,\mu) = \begin{cases} \frac{\mu^3+3\mu-4}{2} \cdot \frac{N}{\mu}, & \mu \text{ odd} \\ \frac{\mu^3-4}{2} \cdot \frac{N}{\mu}, & \mu \text{ even}_1 \\ \frac{\mu^3+2\mu-6}{2} \cdot \frac{N}{\mu}, & \mu \text{ even}_2 \end{cases} \tag{5}$$

At the higher level, note that all local monitors form a mesh by themselves. Choosing the central or near-central node among them as the monitor will give the minimum communication cost. However, the cost of "one step" (i.e., passage of data from a node to its immediate neighbor) is μ instead of 1. Applying equations 3 and 4 again, the cost for level-two is given as follows:

$$C_{II}(N,\mu) = \begin{cases} \frac{\frac{N}{\mu}^3+3\frac{N}{\mu}-4}{2} \cdot \mu, & \frac{N}{\mu} \text{ odd} \\ \frac{\frac{N}{\mu}^3-4}{2} \cdot \mu, & \frac{N}{\mu} \text{ even}_1 \\ \frac{\frac{N}{\mu}^3+2\frac{N}{\mu}-6}{2} \cdot \mu, & \frac{N}{\mu} \text{ even}_2 \end{cases} \tag{6}$$

Combining equations 5 and 6, we have the expression for total cost of the two-level hierarchical monitoring system:

$$C_{total}(N,\mu) = C_I(N,\mu) + C_{II}(N,\mu) \tag{7}$$

$$= \begin{cases} \frac{\mu^3(N^2-4)+3N\mu^2+3N^2\mu+N^2(N-4)}{2\mu^2}, & \mu \text{ odd}, \frac{N}{\mu} \text{ odd} \\ \frac{\mu^3(N^2-4)+3N^2\mu+N^2(N-4)}{2\mu^2}, & \mu \text{ odd}, \frac{N}{\mu} \text{ even}_1 \\ \frac{\mu^3(N^2-4)+2N\mu^2+3N^2\mu+N^2(N-4)}{2\mu^2}, & \mu \text{ odd}, \frac{N}{\mu} \text{ odd}_2 \\ \frac{\mu^3(N^2-4)+3N\mu^2N^2(N-4)}{2\mu^2}, & \mu \text{ even}_1, \frac{N}{\mu} \text{ odd} \\ \frac{\mu^3(N^2-4)+N^2(N-4)}{2\mu^2}, & \mu \text{ even}_1, \frac{N}{\mu} \text{ even}_1 \\ \frac{\mu^3(N^2-6)+2N\mu^2+N^2(N-4)}{2\mu^2}, & \mu \text{ even}_1, \frac{N}{\mu} \text{ even}_2 \\ \frac{\mu^3(N^2-4)+3N\mu^2+2N^2\mu+N^2(N-6)}{2\mu^2}, & \mu \text{ even}_2, \frac{N}{\mu} \text{ odd} \\ \frac{\mu^3(N^2-4)+2N^2\mu+N^2(N-6)}{2\mu^2}, & \mu \text{ even}_2, \frac{N}{\mu} \text{ even}_1 \\ \frac{\mu^3(N^2-6)+2N\mu^2+2N^2\mu+N^2(N-6)}{2\mu^2}, & \mu \text{ even}_2, \frac{N}{\mu} \text{ even}_2 \end{cases}$$

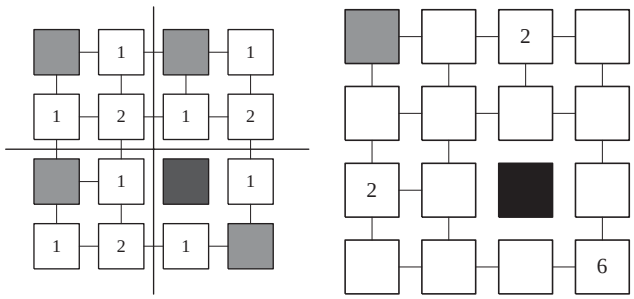
4 Failure recovery methods

When dealing with hierarchical levels it becomes necessary to worry about failure within an individual cluster as well as failure in the higher levels. A single failed node can have a great impact upon both logical levels in the hierarchy. When our primary concern is failure within a cluster, the first proposed strategy is ideal: *singular shifting*. If keeping optimality at the higher levels is more highly desired the second proposed strategy may be better: *total shifting*. These two methods will be examined in depth, in an effort to find which method is best for a given situation.

4.1 Singular shifting

It has been shown that to keep optimality within the hierarchical system it is necessary to keep optimality within each cluster [7]. This greedy principal prompted the creation of the first algorithm for dealing with fault tolerance; *singular shifting*.

Within any cluster it is always necessary to communicate with the monitor node. The different failure situations discussed previously can be put to use in each cluster. If a leaf fails it is ignored; if an interior node fails it can be ignored, or it can add additional processing to the system. The most important situation of all occurs when the middle fails. If the middle fails, the optimal choice for a new middle should always be chosen. If we always make an optimal choice, the system will always remain optimal.



(a) Lower level **(b) Upper level**

Figure 13: Faulty $N = 4$ mesh with cluster size of $\mu = 2$. The failed node is pictured in black, the monitor nodes are in gray. Communication costs are listed for each cluster on the lower level (a) as well as for the monitors on the upper level(b).

This process is repeated for each cluster in the mesh as well as on the upper level. An ideal system is shown in Figure 12, a faulty system can be seen in Figure 13. The following algorithm can be used to describe the behavior of *singular shift*.

```
Do an optimal two-level partition
for each (cluster on lower level)
{
    if (cluster head has failed)
        Move cluster head to optimal position
    Calculate total cluster cost
}
Calculate upper level cost
```

As it was discussed earlier, this algorithm aims to keep optimality at the cluster level. The upper level therfore may experience sub-optimal performance due to the haphazard placement of monitors. *Singular shifting* does not make an attempt to meet a uniform standard.

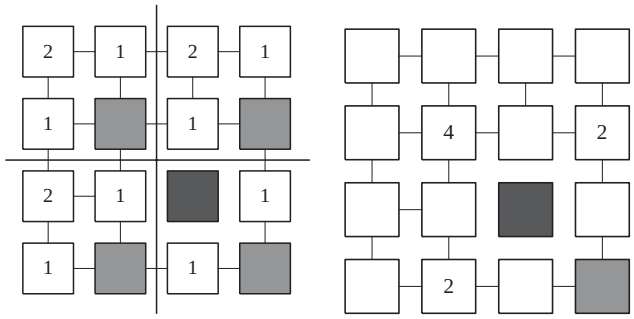
4.2 Total shifting

When a mesh experiences a significant amount of monitor failure, many cluster heads may need to be moved to allow for optimal performance. This indicates that not all of the cluster heads will be in the same relative position; causing the need for longer routes, particularly at the upper level.

Having a uniformity at the upper level is desirable, especially when large amounts of data must be transfered. It would be beneficial to keep the upper level uniform, as well as keeping distances to a minimum. This principal prompted the creation of the second algorithm for dealing with fault tolerance; *total shifting*.

In *total shifting* we examine each of the initial cluster heads for failure. If any of the monitors has failed we then perform the same operation on the potential cluster heads. It is necessary

to find the best "position" for all of the new cluster heads (i.e., find the "position" where the least number of monitors has failed). When a satisfactory position is found we move all monitors to this position. Subsequently, we move the upper level monitor as well to reflect the cluster level changes.



(a) Lower level **(b) Upper level**

Figure 14: A faulty $N = 4$ mesh with cluster size of $\mu = 2$. The failed node is in black, the monitor nodes are in gray. Communication costs are listed for each cluster on the lower level(a) as well as the monitors on the upper level (b).

With this process we aim to gain an optimal upper level, as well as semi-optimal clusters. An ideal system is shown in Figure 12, a faulty system can be seen in Figure 14. The following algorithm can be used to describe the behavior of *total shift*.

```
Do an optimal two-level partition
repeat
{
    count ← 0
    for each (cluster head)
        if (cluster head has failed)
            count++;
}
while ( count ≥ 0)
{
    Move all cluster heads
    for each (cluster on lower level)
        Calculate total cluster cost
    Calculate upper level cost
}
```

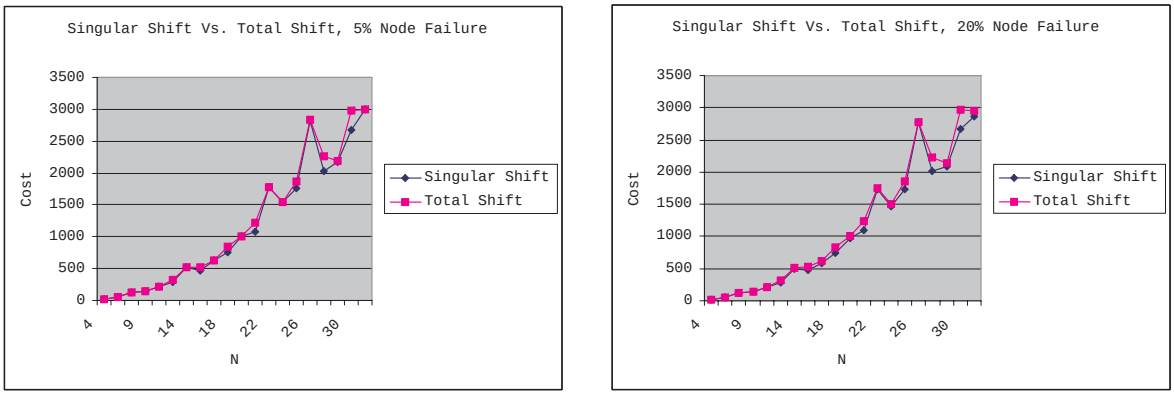
Both *singular shifting* and *total shifting* attempt to gain back some of the optimality lost when failure within the mesh has occured. Testing each algorithm for many different situations should illustrate where the strong and weak points of each algorithm lie.

5 Algorithmic analysis

Testing of the two algorithms was performed side by side with that of an ideal fault free system. The tests were conducted in this manner in order to reveal overall performance characteristics as well as relative optimality.

Simulations were constructed to test each algorithm on mesh systems between $N = 4$ and $N = 32$ processing units as well as with node failure rates between 5% and 20%. The simulations were performed several times in order to develop the average behavior.

The simulations have shown that there is usually a very small difference in the results of the two algorithms; *singular shifting* has been observed to have the lower cost much of the time. As the failure percentage increases *singular shifting* becomes an overwhelming winner, but by a very small margin. In situations where *total shifting* happens to outperform *singular shifting*, the difference between the two remains extremely small. Figure 15 shows two graphs; each having a different failure rate for nodes within the system. These graphs show that the algorithms have produced almost identical results.



(a) 5% Node Loss **(b) 20% Node Loss**

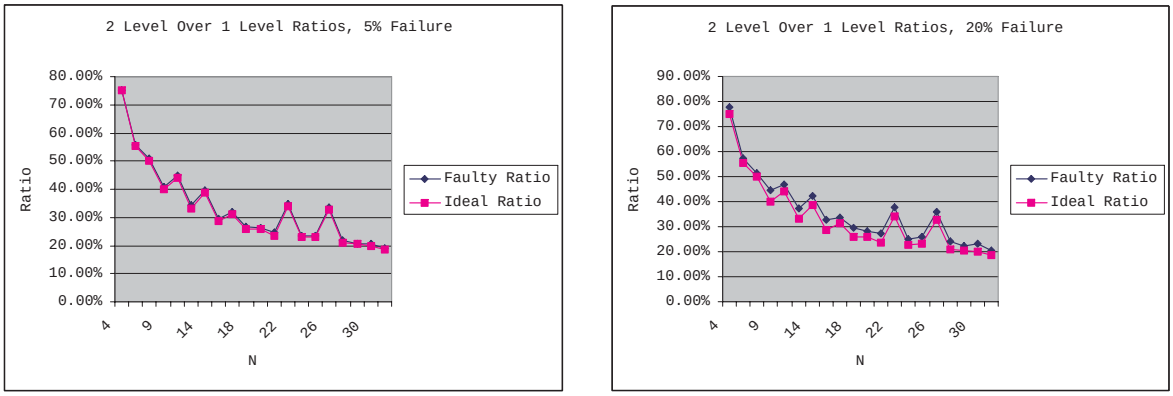
Figure 15: Graph showing mesh sizes $N = 4$ to $N = 32$ averaged over twenty five trials. *singular shifting* produces the lower costs most of the time.

As the mesh size, and subsequently the cluster size, increases it is clear that *singular shifting* is the prefered algorithm. Because of the increasing size and amount within each cluster it is far better to achieve optimal performance at the lower level than at the upper level. Having optimal performance at a higher level would be valued, but not at the price of loosing optimality on the lower levels.

Total Shifting forces conformity at the lower level; this choice can sometimes move the monitoring task away from a perfectly good node. This loss of optimality is enough to cause the entire system to suffer. In addition the cost of trying to find the "ideal" position for the new monitor causes the algorithm to run a little slower to that of the "on demand" nature of *singular shifting*.

Both algorithms operate with an asymptotic running time of $O(n^3)$. This is due in part to the inclusion of the "all pairs shortest path" algorithm used to calculate distances between each node and the monitor. This dictates the overall speed, so the additional calculation needed to find the "ideal" monitor in the *total shifting* algorithm is negligible.

It is possible to compute a ratio of the hierarchical cost to that of the single level cost, (i.e., $\frac{Equation6}{Equation3 or Equation4}$) in an effort show how well the system performs. This ratio was calculated for the ideal system, as well as to that of the a faulty system utilizing the *singular shifting* method. Figure 16 shows the results of this calculation. Having a low ratio is desirable in this case; it indicates there is a large difference between that of the hierarchical configuration and the single level meshes.



(a) 5% Node Loss **(b) 20% Node Loss**

Figure 16: Graph showing mesh sizes $N = 4$ to $N = 32$ and the ratios between a single level mesh and a hierarchical two level system. The faulty systems produce a higher ratio to that of an ideal system.

Figure 16a shows a faulty system with 5% node loss; the results are almost identical to that of an ideal system. Figure 16b features a faulty system with 20% node loss; in comparison to the ideal system it performs a bit worse, but still with the same general performance curve.

6 Conclusion

We have proposed hierarchical distributed monitoring schemes for faulty systems, ultimately deciding that for moderate levels of failure (i.e., less than or equal to 20%), performance similar to that of an ideal system can be achieved. This conclusion is based on the results of our empirical experiments showing that optimal performance is easy to achieve though the use of the devised algorithms. The proposed solutions are easy to calculate although expensive, especially when the mesh size grows.

Additional interesting research topics can be built from this basic concept, namely for deciding when the proper time to use *singular shifting* vs. *total shifting* is warranted. It is hypothesized that as the system carries large amounts of data from the clusters to the higher level, *total shifting* would be highly desirable. If the amount of data is smaller at the higher levels, and larger within the clusters it singular shifting will produce better results.

References

[1] J. Cao, O. de Vel, and L. Shi, "Architecture Design of Distributed Performance Monitoring Systems: A Hierarchical Approach," *Proc. 7th International Conference on Parallel and Distributed Computing Systems*, Las Vegas, USA, October 1994, pp. 658-663.

[2] J. Cao, K. Zhang, and O. de Vel, "On Heuristics for Optimal Configuration of Hierarchical Distributed Monitoring Systems," *Journal of Systems and Software*, Vol. 45, No. 2, pp. 141-154, 1999.

[3] G. Feitelson and L. Rudolph, "Distributed Hierarchical Control for Parallel Processing," *Computer*, pp. 65-77, May 1990.

[4] L. Shi, O. De Vel, J. Cao, and M. Cosnard, "Optimization in a Hierarchical Distributed Performance Monitoring System," *Proc. First IEEE International Conference on Algorithms and Architectures for Parallel Processing*, Brisbane, Australia, April 1995, pp. 537-543.

[5] L. Shi, J. Cao and O. de Vel, "A Hierarchical, Distributed Monitoring System For Interprocess Communications," to appear in *International Journal of Computer Systems: Science and Engineering.*

[6] M. Spezialetti and J.P. Kearns, "A General Approach to Recognizing Event Occurrences in Distributed Computations," *Proc. IEEE 8th Int'l Conf. on Dist. Comput. Sys.*, 1988, pp. 300-307.

[7] D. Wang, J. Cao, "On optimal hierarchical configuration of distributed system on mesh and hypercube," *2003 Workshop on Advances in Parallel and Distributed Computational Models (APDCM'03), in conjunction with International Parallel and Distributed Processing Symposium (IPDPS)*, Nice, France, April 2003, pp 167.

[8] C.-Q. Yang and B.P. Miller, "Performance Measurement for Parallel and Distributed Programs: A Structured and Automatic Approach," *IEEE Transactions on Software Engineering*, Vol. 15, No. 12, pp. 1615-1629, December 1989.

[9] Y. Zhu, "Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers," *Journal of Parallel and Distributed Computing*, No. 16, pp. 328-337, 1992.