International Journal of Foundations of Computer Science

ON HIERARCHICAL CONFIGURATION OF DISTRIBUTED SYSTEMS ON MESH AND HYPERCUBE*

DAJIN WANG
Department of Computer Science, Montclair State University
Upper Montclair, NJ 07043, USA
wang@pegasus.montclair.edu

and

JIANNONG CAO
Department of Computing, The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
csjcao@comp.polyu.edu.hk

Received 8 June 2003
Revised 28 January 2004
Communicated by Jie Wu

ABSTRACT

We study hierarchical configuration of distributed systems for achieving optimized system performance. A distributed system consists of a collection of local processes which are distributed over a network of processors, and work in a cooperative manner to fulfill various tasks. A hierarchical approach is to group and organize the distributed processes into a logical hierarchy of multiple levels, so as to coordinate the local computation/control activities to improve the overall system performance. It has been proposed as an effective way to solve various problems in distributed computing, such as distributed monitoring, resource scheduling, and network routing. The optimization problem considered in this paper is concerned with finding an optimal hierarchical partition of the processors, so that the total traffic flow over the network is minimized. The problem in its general form has been known to be NP-hard. Therefore, we just focus on distributed computing jobs which require collecting and processing information from all processors. By limiting levels of the hierarchy to two, we will establish the analytically optimal hierarchical configurations for two popular interconnection networks: mesh and hypercube. Based on analytical results, partitioning algorithms are proposed to achieve minimal communication cost (network traffic flow). We will also present and discuss heuristic algorithms for multiple-level hierarchical partitions.

Keywords: Distributed processing, Hierarchical architecture, Hierarchical configuration, Hypercube, Interconnection networks, Mesh.

*This work was partially supported by Hong Kong Polytechnic University under HK PolyU Research Grant A-PA24.

1. Introduction

The topologies of many distributed systems are more or less hierarchical. If distributed functions are performed in such a way as to reflect the underlying hierarchical topology, the design can be simplified. A hierarchical architecture may also improve scalability of the distributed functions and optimize their performance by increasing parallelism and reducing information flow. As a matter of fact, the hierarchical approach of letting distributed processes operate on a logical structure is a well-known design methodology, and has been used in a variety of forms for solving different distributed control problems, such as distributed monitoring, resource scheduling, and network routing, either to effectively coordinate the local control activities or to enhance the overall performance [1, 2, 5, 6, 8, 20].

Although a lot of work has been done in developing a variety of hierarchical distributed functions, most of them only proposed solutions based on existing hierarchical structures of processes. The important issue of how to form the hierarchy has not been adequately addressed. In this paper, we study the algorithms for optimally configuring a distributed system into a hierarchy of multiple levels of groups. The objective of the optimization algorithms is to find such a configuration for a given network topology that the total computation cost in terms of processing and communication as mapped into the hierarchy will be minimum.

In our previous works, we have developed a hierarchical monitoring system for a distributed environment [4, 18]. Conventional monitoring systems use a simple scheme: A collection of resident monitoring units on each processor, which collect and pre-process as much as possible local information, and a logically centralized unit, which correlates and stores distributed information. A hierarchical distributed monitoring system structures the monitoring units into groups at multiple levels. Each group has a leader, and group leaders at the same level may form groups at a higher level. Monitoring information is processed starting at individual monitoring units and integrated by group leaders. The proposed hierarchical architecture is unique in that it lends itself to parallel processing and allows complex, topology-specific events of different types of systems to be monitored and evaluated in a natural and efficient way. It reduces the complexity of distributed monitoring caused by such factors as spatial distribution of the distributed software, recognizing, collecting and processing of the large quantities of monitoring data, and heterogeneity of the computers in the distributed system.

We have performed experimental study of the performance of hierarchical monitoring system as compared with a conventional distributed monitoring system. Tests were designed for various sizes of monitoring groups and for different grouping strategies in order to find useful observations on forming an optimal configuration. Obtaining an optimal configuration means the minimization of the processing cost (time, memory space, and inter-process communication). The experiments showed a significant performance improvement made by the hierarchical monitoring system [18]. It was also found that the performance of the monitoring system was a function of the number and size of groups at each level. As a matter of fact, the solution to the optimization problem is both application and network topology dependent and

it can be shown that this problem is NP-complete. In [4], we identified the factors that are significant in determining the cost and performance of a hierarchical distributed monitoring system, and described various heuristics of finding an optimal or near-optimal hierarchical configuration of the monitoring system. In addition to empirical studies, we have also proposed an efficient algorithm for configuring the hierarchical monitoring units to optimality in a tree-structured system [23].

In this paper, we will focus on two regular network topologies: mesh and hypercube. Both are very popular networks, have been extensively studied, and commercial parallel computers using them have been available for a long time. We will present hierarchy schemes on mesh and hypercube that greatly reduces the traffic flow. As in most cases of optimization, a hierarchical configuration optimizing all factors is impossible to achieve. Therefore it is important to characterize the applicability of the proposed scheme. Generalizing the hierarchical monitoring system in our previous works [4, 18], the hierarchical configuration we present in this work is best applicable to those tasks that need to process data collected from all processors of the network, and the nature of the task allows "partial preprocessing" of data before they reach their final destination. There are many such data in both computational and managerial tasks. The monitoring data we have mentioned is of such nature. Another simple example is to get the sum of certain value from all processors: it is not necessary for the master adder of the network to collect all addends before it performs the addition — partial sums can be obtained by some "submasters," and sent to the master adder. That will prevent many pieces of data from traveling all the way to the master, reducing the traffic flow in the network.

Under the above context, by first limiting the levels of the hierarchy to two, we study optimal hierarchical configurations for mesh and hypercube. Based on analytical results, partitioning algorithms are presented which are optimal in terms of total communication cost. We will then discuss and present heuristic algorithms for multiple-level hierarchical partitions.

Although the scheme of this paper is presented in a specific context, the approaches may be applied to a broader range of hierarchical control/computational problems in distributed processing. However, in general, different distributed functions have different objectives and there are various forms of hierarchical configuration problems in different applications. Therefore, the grouping strategy and the semantics of the operations may be different, e.g., the control function may be initiated top down or bottom up, and thus the information flow can be either downward or upward towards the group leader. For example, a key challenge of distributed multi-agent systems is to achieve group-level coherence. Such coherence is usually guaranteed by top-down control where a central controller must maintain updated information about the entire group, perform optimizations over the global state space, and send commands to the group. In addition to the high computation overhead required, the communication imposes a significant bottleneck that scales poorly with increased group size [13]. As another example, hierarchical network routing schemes have been proposed to reduce the volume of routing information that needs to be handled by the nodes [5, 20]. This is done by aggregating nodes

into clusters and clusters into superclusters, and so on. Each cluster has an address server which keeps track of the membership and address of the nodes in its cluster. Address servers cooperate to determine a routing address in the entire network. This hierarchical architecture makes the routing protocol scalable because, even if the size of the network becomes large, the amount of link state information can be reduced much less than that of the existing link state routing protocol. Study of algorithms for different forms of hierarchical configurations will be our future work.

The rest of this paper is organized as follows. In Section 2, we briefly describe the hierarchical architecture systems, formulate the optimal configuration problem, and define the terminology. Section 3 presents the analytical results for optimal two-level partitioning for mesh, and based on that proposes an algorithm to obtain the hierarchical structure. In Section 4, analysis for optimal hypercube two-level partitioning is done, and the hierarchical structuring algorithm is proposed. In Section 5 we briefly summarize the work of this paper and outline directions for extension.

2. Hierarchical configuration of distributed systems

The problem of optimal hierarchical configuration of a distributed system is concerned with finding an optimal partition of the processes/processors in a given network environment. A configuration C of a system consists of three parts: (1) a hierarchical partition of the nodes (processes/processors), defined by the levels of the hierarchy, (2) the grouping of nodes at each level, and (3) the location of the leader at each group. Optimization means to minimize the total processing cost. The three costs that are of primary consideration are the amount of memory required, the amount of communication between units, and the time required for processing. In this paper, we focus on reducing the amount of communication over the network.

In order to formulate such a problem quantitatively for analysis, we assume a simple model for the process of collecting information and perform some processing (e.g. combining information, making decisions) based on it. In the context of this paper, a hierarchical organization consists of local processes which aggregate information from other processes in the hierarchy, passing that information through the hierarchy. During this process, information can be condensed as it passes through the hierarchy. We are mainly concerned with the minimization of the total communication cost. In mesh-connected or hypercube computers, a node sending data to its group leader has to pass all intermediate nodes. Processors that are far apart have to travel long way to communicate, incurring great amount of data flow in the network. We normalize a node's cost for traveling to its immediate neighbor to "one step." The number of steps that a node travels to its leader is said to be its *communication cost*. For example, if there are 4 intermediate nodes between a node and its data collector, the communication cost for collecting this node's data is 5. The total communication cost is the total number of steps that all nodes have to travel in the processing. The target of this work is to find a hierarchical structure of nodes so that the total communication cost is minimized.

It is worth pointing out that assuming a normalized "1" cost for each hop of communication is for the purpose of mathematical tractability. Once the analytical hierarchy is obtained with that assumption, it should be useful to all kinds of (fixed or variable) cost incurred in communication between nodes.

We study the optimization problem for two specific networks: mesh and hypercube. It was discovered in empirical study that due to the large variety of the parallel and distributed systems and applications, it is impossible to find out a general solution to the optimization problem for all sorts of networks/applications. We choose these two networks because of their popularity in multicomputer systems. Furthermore, in this paper we restrict our treatment only to squared meshes for tractability. Meshes of more general dimensions will be considered in future work. Finally, in this paper, "processor," "process," and "node" will be used interchangeably.

3. Hierarchical configuration for mesh

Let the squared mesh contain N^2 processors, with dimensions $N \times N$. If the whole mesh is viewed as one hierarchy (one-level), then choosing the center node as the master node (i.e., the group leader with the entire mesh as the group) would obviously minimize the total communication cost. Depending on whether N is an odd or even number, the cost can be calculated as follows. See Figure 1.

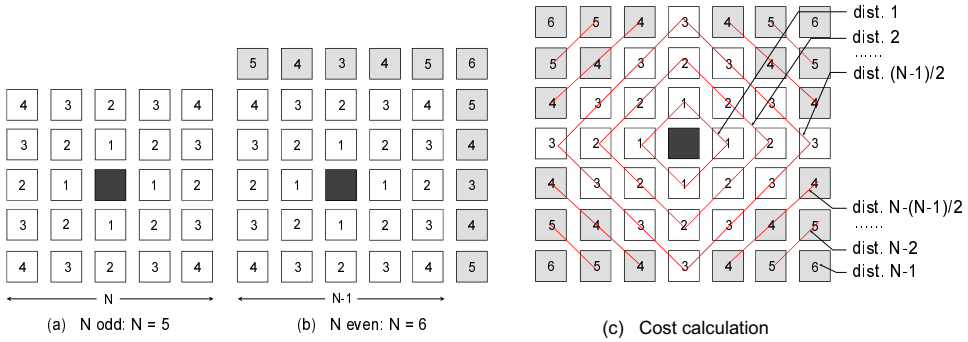


Fig. 1. (a) A 5×5 mesh. At center is the master node (dark). Numbers in all other nodes represent their communication costs. (b) A 6×6 mesh. The dark node at the "pseudo center" is the master. (c) Illustration for cost calculation.

N is odd

When N is odd, there exists a true central node, and it will be taken as the master node (the dark node in Figure 1(a) and (c)). The number in a node represents its communication cost for monitoring, i.e., the Hamming distance from itself to the monitor. The total communication cost, denoted as $C_o(N)$, can be calculated

as follows. Refer to Figure 1(c).

$$
\begin{aligned}
C_o(N) &= \underbrace{\overbrace{1\cdot 1\cdot 4}^{\text{nodes of dist. 1}} + \overbrace{2\cdot 2\cdot 4}^{\text{nodes of dist. 2}} + \overbrace{3\cdot 3\cdot 4}^{\text{nodes of dist. 3}} + \cdots + \overbrace{((N-1)/2)\cdot((N-1)/2)\cdot 4}^{\text{nodes of dist. (N-1)/2}}}_{\text{all white nodes in Figure 1(c)}} \\
&+ \underbrace{\overbrace{(N-1)\cdot 1\cdot 4}^{\text{nodes of dist. N-1}} + \overbrace{(N-2)\cdot 2\cdot 4}^{\text{nodes of dist. N-2}} + \cdots + \overbrace{(N-(N-1)/2)\cdot((N-1)/2)\cdot 4}^{\text{nodes of dist. N-(N-1)/2}}}_{\text{all grey nodes in Figure 1(c)}} \\
&= 4\sum_{i=1}^{\frac{N-1}{2}} i^2 + 4\sum_{i=1}^{\frac{N-1}{2}} (N-i)i \\
&= 4N\sum_{i=1}^{\frac{N-1}{2}} i \\
&= \frac{N^3-N}{2}
\end{aligned}
$$

N is even

Refer to Figure 1(b). When N is even, there is no true central node. Any one of the four nodes in the central "area" can be picked as the master, as shown in Figure 1(b). The total cost, denoted as $C_e(N)$, is $C_o(N-1)$ plus the cost of grey nodes.

$$
\begin{aligned}
C_e(N) &= \underbrace{\frac{(N-1)^3-(N-1)}{2}}_{C_o(N-1)} + \underbrace{2\cdot(N/2) + 4\sum_{i=\frac{N}{2}+1}^{N-1} i + 1\cdot N}_{\text{all grey nodes in Figure 1(b)}} \\
&= \frac{N^3}{2}
\end{aligned}
$$

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

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

It can be shown that using any non-central monitor would cost more, with the monitor at corner costing most.

In the following Section 3.1, we will obtain the optimal partition of processors assuming a two-level hierarchy. Using the result for two-level partitioning, in Section 3.2 we present the algorithm for optimal multiple-level partitioning.

3.1. Optimal two-level partitioning

In a two-level partitioning, the whole mesh is divided into several submeshes. Each submesh has a group leader. A group leader will collect data in its submesh, and then turn the data in to a leader at the higher level. As stated before, 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.

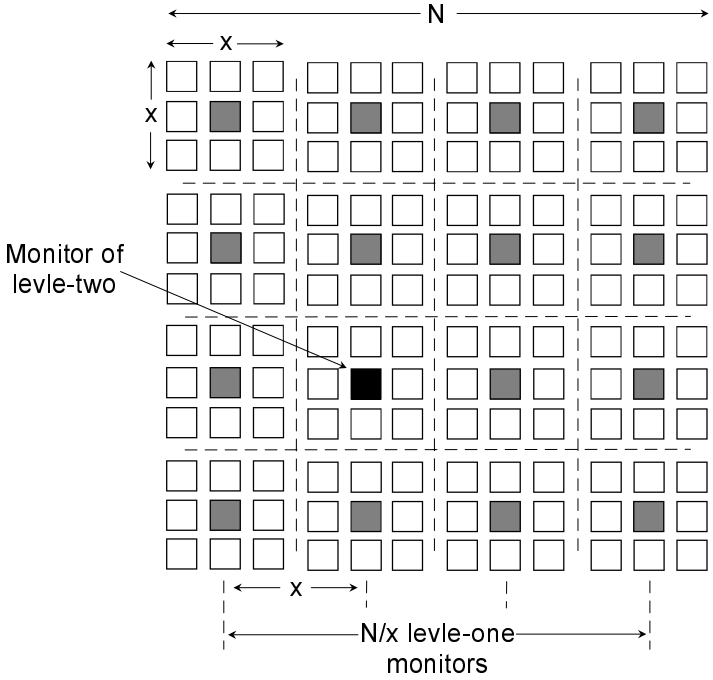


Fig. 2. Two-level partitioning. The $N \times N$ nodes are divided into $(\frac{N}{x})^2$ $x \times x$ submeshes. The grey nodes are level-one leaders, the darkest node is level-two leader.

See Figure 2. Let the submesh be of dimension $x \times x$, so that x divides N. Then by Eq. (1) the cost for local monitoring will be:

$$C(x) = \begin{cases} \frac{x^3-x}{2}, & x \text{ odd} \\ \\ \frac{x^3}{2}, & x \text{ even} \end{cases}$$

Therefore the total cost for all $(\frac{N}{x})^2$ level-one submeshes is given by:

$$C_I(N,x) = \begin{cases} \frac{1}{2}(x^3-x) \cdot (\frac{N}{x})^2, & x \text{ odd} \\ \\ \frac{1}{2}(x^3) \cdot (\frac{N}{x})^2, & x \text{ even} \end{cases} \quad (2)$$

At level-two, note that all local leaders form a squared mesh by themselves (the darker nodes in Figure 2). So choosing the central or near-central node among them

as the leader (the darkest node in Figure 2) 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 x instead of 1. Applying Eq. (1) again, the cost for level-two is given as follows:

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

Combining (2) and (3), we have the expression for total cost of the two-level hierarchical monitoring system:

$$C_{total}(N,x) = C_I(N,x) + C_{II}(N,x)$$

$$= \begin{cases} \frac{1}{2}(x^3 - x) \cdot (\frac{N}{x})^2 + \frac{1}{2}((\frac{N}{x})^3 - \frac{N}{x}) \cdot x &=& \frac{N^2x^3 - Nx^2 - N^2x + N^3}{2x^2}, & x \text{ odd}, \frac{N}{x} \text{ odd} \\ \frac{1}{2}(x^3 - x) \cdot (\frac{N}{x})^2 + \frac{1}{2}(\frac{N}{x})^3 \cdot x &=& \frac{N^2x^3 - N^2x + N^3}{2x^2}, & x \text{ odd}, \frac{N}{x} \text{ even} \\ \frac{1}{2}(x^3) \cdot (\frac{N}{x})^2 + \frac{1}{2}((\frac{N}{x})^3 - \frac{N}{x}) \cdot x &=& \frac{N^2x^3 - Nx^2 + N^3}{2x^2}, & x \text{ even}, \frac{N}{x} \text{ odd} \\ \frac{1}{2}(x^3) \cdot (\frac{N}{x})^2 + \frac{1}{2}(\frac{N}{x})^3 \cdot x &=& \frac{N^2x^3 + N^3}{2x^2}, & x \text{ even}, \frac{N}{x} \text{ even} \end{cases}$$

There is an optimal x to make the minimum $C_{total}(N,x)$. To obtain the optimal x, just take the derivative of $C_{total}(N,x)$ with respect to x, denoted $C_{total}(N,x)'_x$, and solve $C_{total}(N,x)'_x = 0$ for x.

$$C_{total}(N,x)'_x = \begin{cases} \left(\frac{N^2x^3 - Nx^2 - N^2x + N^3}{2x^2}\right)'_x &=& \frac{N^2}{2} + \frac{N^2}{2x^2} - \frac{N^3}{x^3}, & x \text{ odd}, \frac{N}{x} \text{ odd} \\ \left(\frac{N^2x^3 - N^2x + N^3}{2x^2}\right)'_x &=& \frac{N^2}{2} + \frac{N^2}{2x^2} - \frac{N^3}{x^3}, & x \text{ odd}, \frac{N}{x} \text{ even} \\ \left(\frac{N^2x^3 - Nx^2 + N^3}{2x^2}\right)'_x &=& \frac{N^2}{2} - \frac{N^3}{x^3}, & x \text{ even}, \frac{N}{x} \text{ odd} \\ \left(\frac{N^2x^3 + N^3}{2x^2}\right)'_x &=& \frac{N^2}{2} - \frac{N^3}{x^3}, & x \text{ even}, \frac{N}{x} \text{ even} \end{cases}$$

Solving $\frac{N^2}{2} + \frac{N^2}{2x^2} - \frac{N^3}{x^3} = 0$ and $\frac{N^2}{2} - \frac{N^3}{x^3} = 0$, respectively, for x, and only taking the real root, we get

$$x = \begin{cases} \frac{\sqrt[3]{27N + 3\sqrt[2]{3 + 81N^2}}}{3} - \frac{1}{\sqrt[3]{27N + 3\sqrt[2]{3 + 81N^2}}}, & x \text{ odd} \\ \\ \sqrt[3]{2N}, & x \text{ even} \end{cases}$$

The difference between $\sqrt[3]{2N}$ and $\left(\frac{\sqrt[3]{27N + 3\sqrt[2]{3 + 81N^2}}}{3} - \frac{1}{\sqrt[3]{27N + 3\sqrt[2]{3 + 81N^2}}}\right)$ is vanishingly small. So for all practical purposes, we can just use an integer close to $\sqrt[3]{2N}$ for the size of submesh to achieve the minimum total cost.

Theorem 1 *In a two-level hierarchical monitoring system, if the level-1 submesh is of dimension $x \times x$, so that*

1. *x is as close to $\sqrt[3]{2N}$ as possible*
2. *x divides N*

then the system's total communication cost is minimum.

Figure 3 illustrates the level-1, level-2, and total costs as function of submesh size x, for a two-level monitoring system, where the original mesh size is $N = 72$. $\sqrt[3]{2N} = \sqrt[3]{144} \approx 5.24$. According to Theorem 1, $x = 6$ will be chosen as the optimal submesh size. The total cost is 20736, which is minimum.

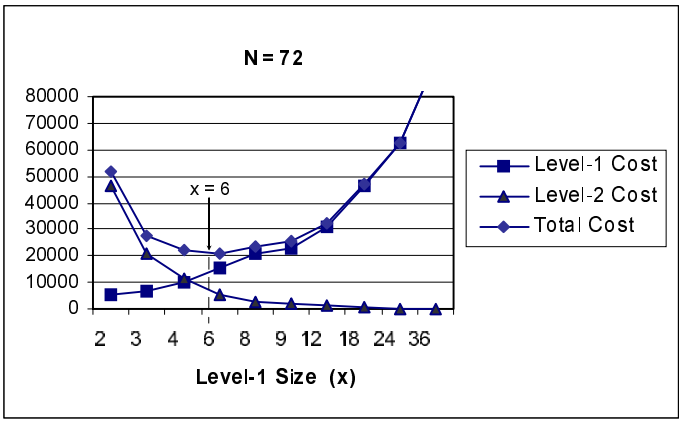


Fig. 3. Level-1, level-2, and total costs as function of submesh size x. The original mesh size is $N = 72$. It can be seen that there is a minimum total cost.

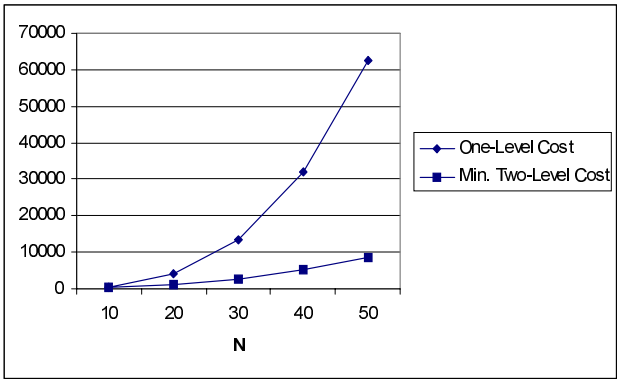
The saving of communication cost gained by this two-level hierarchical scheme is substantial. The ratio of min.-two-level-cost/one-level-cost is (assuming even N, even x)

$$\frac{\left.\frac{N^2x^3+N^3}{2x^2}\right|_{x=\sqrt[3]{2N}}}{\left(\frac{N^3}{2}\right)} = \frac{3}{2}\sqrt[3]{\frac{2}{N^2}}$$

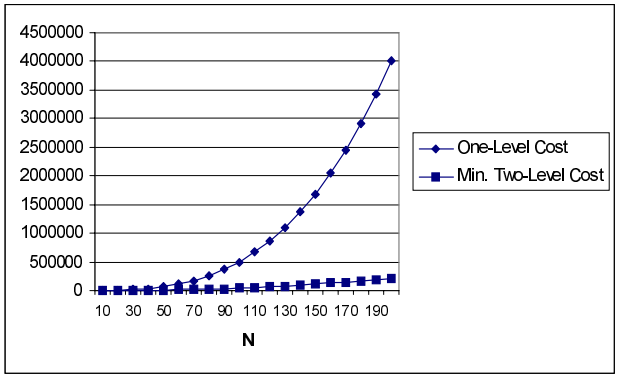
which is ever decreasing as N grows. Figure 4(a) shows a comparison between minimum two-level cost and one-level cost, and Figure 4(b) shows the same comparison for a broader range of N. When $N = 10$, the min.-two-level-cost/one-level-cost ratio is about 40%; when $N = 100$, less than 9%; when $N = 200$, less than 6%.

3.2. Multiple-level partitioning

If the levels of partition are more than 2, cost can be further reduced. Extending the approach for two-level partition, for a partition of m levels, let x_1 be the submesh size at level-1 (the bottom level), x_2 the submesh size at level-2, ..., so that $N = x_1 \times x_2 \times \cdots \times x_m$. Ideally, an optimal total cost can be obtained by finding the



(a)



(b)

Fig. 4. Comparison of minimum two-level cost and one-level cost.

minimum value of a multi-variable cost function $C(x_1, ..., x_m)$. However, even for a modest m, that is a mathematically burdensome task. So instead of computing the absolutely optimal partition for a larger m, we resort to repeatedly applying the result for two-level partition, until the desired level number is reached or desired total cost obtained.

It is observed that in optimal two-level partition, the cost of level-one (bottom level) is twice as large as that of level-two. Assuming even N, even x:

$$C_I(N, x = \sqrt[3]{2N}) = \frac{1}{2}(x^3) \cdot (\frac{N}{x})^2 \Big|_{x=\sqrt[3]{2N}} = \frac{1}{2}\sqrt[3]{2N^7}$$

and

$$C_{II}(N, x = \sqrt[3]{2N}) = \frac{1}{2}(\frac{N}{x})^3 \cdot x \Big|_{x=\sqrt[3]{2N}} = \frac{1}{4}\sqrt[3]{2N^7} = \frac{C_I}{2}$$

So, for three-level partition, we can do a second two-level partition for all sub-meshes at the bottom level, further reducing the total cost. As the number of levels increases, it is not always the bottom level that has the largest cost. However, keeping track of the level of largest cost is not a difficult job. All we need to do is to compare the current largest cost with the cost of newly obtained level. The algorithm for multiple-level partition is described below.

Do an optimal two-level partition /* initial step */

levelCount $\leftarrow 2$

levelOfLargestCost $\leftarrow 2$

repeat

{

At *levelOfLargestCost*, do an optimal two-level partition

Update *levelOfLargestCost*

levelCount++

}

until (*levelCount* = *targetLevelNumber* || *totalCost* $\leq$ *targetCost*)

An important observation is that the magnitude of cost reduction, in terms of percentage, drops rather quickly as each new level is added, i.e, the most substantial saving occurs when the system goes from one-level to two-level, but much less substantial going from two-level to three-level, and so on. Take the example of $N = 72$ again. The ratio of min.-two-level-cost/one-level-cost $\frac{3}{2}\sqrt[3]{\frac{2}{N^2}}\Big|_{N=72} \approx 11\%$, with the optimal level-one submesh size 6, and total cost 20736. The costs at level-1 and level-2 are 15552 and 5184, respectively. Another two-level partition at bottom-level will reduce the cost from 15552 to 8640. So the reduction rate from two-level to three-level is $(8640 + 5184)/20736 \approx 67\%$.

What this fact suggests is that after a few partitions, there will be no much significant gain in cost reduction. Therefore, it is a good idea to set the number of levels to a modest value, such as 3, 4, or 5, for most meshes, directly reducing the time complexity of the algorithm.

4. Hierarchical configuration for hypercube

An n-cube consists of 2^n nodes, addressed (numbered) from $\underbrace{00...0}_{n}$ to $\underbrace{11...1}_{n}$. A link exists between two nodes if their addresses differ in one and only one bit. Figure 5 shows a 4-cube.

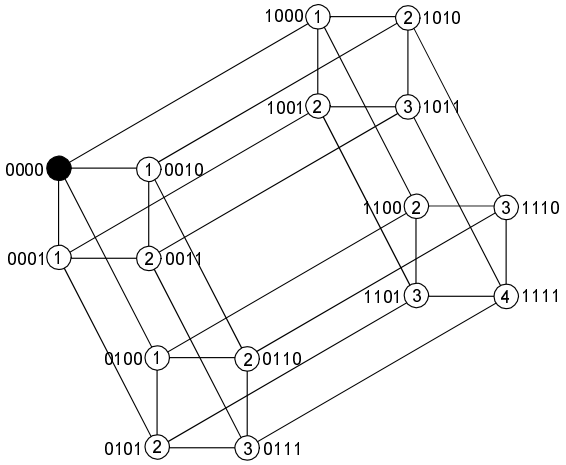


Fig. 5. A 4-cube. Node 0000 is the monitor. The number inside a node represents its communication cost.

Assuming a one-level scheme, since the n-cube is a completely symmetric structure, whichever node is picked as the monitor, the total communication cost is the same. In Figure 5, node 0000 is the monitor. The number inside a node represents its communication cost. The total communication cost $C(n)$ is:

$$\begin{aligned} C(n) &= 1 \cdot \binom{n}{1} + 2 \cdot \binom{n}{2} + \cdots + n \cdot \binom{n}{n} \\ &= \sum_{i=1}^{n} i \cdot \binom{n}{i} = n \sum_{i=0}^{n-1} \binom{n-1}{i} \qquad (4) \\ &= n \cdot 2^{n-1} \end{aligned}$$

4.1. Optimal two-level partitioning

In a two-level scheme, we divide an n-cube into 2^{n-x} x-cubes, where $x \leq n$. The monitoring data in each subcube is collected by a local group leader, represented by the grey nodes in Figure 6. All level-1 leaders themselves form a $(n-x)$-cube,

connected by the bold links in Figure 6, and report to a master node, which is represented by the darkest node in Figure 6. In Figure 6(a), $n = 4$ and $x = 2$. Applying Eq. (4), the local cost is $2 \cdot 2^{2-1} = 4$. There are $2^{4-2} = 4$ subcubes, giving total level-1 cost of 16. Level-1 monitors form a 2-cube at level-2, giving cost

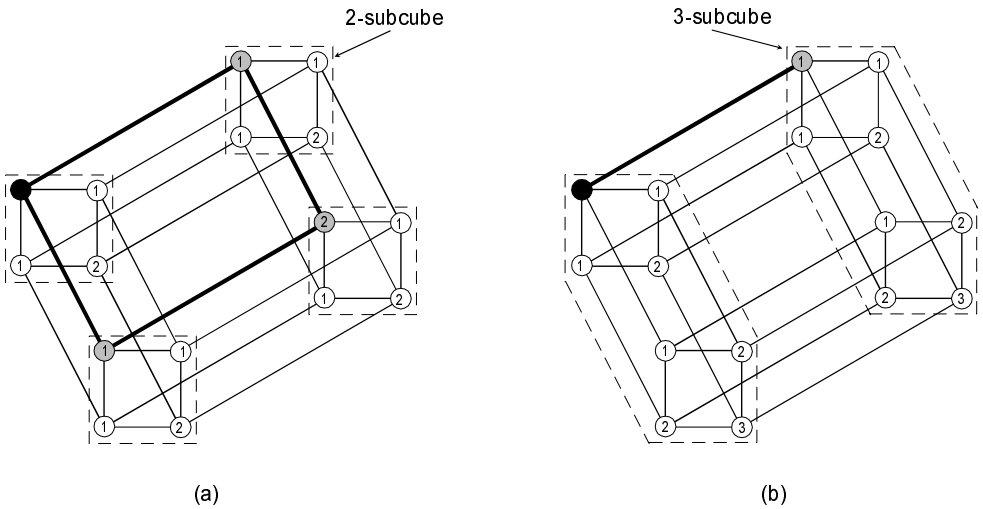


Fig. 6. (a) Subcube size $x = 2$. (b) Subcube size $x = 3$.

To summarize, the total cost for a two-level n-cube monitoring system, denoted $C(n, x)$ with x being the subcube dimension, is

$$\begin{aligned} C(n,x) &= \underbrace{x \cdot 2^{x-1} \cdot 2^{n-x}}_{\text{level-1 cost}} + \underbrace{(n-x) \cdot 2^{(n-x)-1}}_{\text{level-2 cost}} \\ &= x \cdot 2^{n-1} + (n-x) \cdot 2^{n-x-1} \end{aligned} \tag{5}$$

To obtain the minimum value of $C(n, x)$, we take the derivative of $C(n, x)$ with respect to x, denoted $C(n, x)'_x$, and solve $C(n, x)'_x = 0$ for x.

$$\begin{aligned} C(n,x)'_x &= \left(x \cdot 2^{n-1} + (n-x) \cdot 2^{n-x-1}\right)'_x \\ &= 2^{n-1} - 2^{n-x-1} - \ln 2 \cdot (n-x) \cdot 2^{n-x-1} \end{aligned} \tag{6}$$

Solving $2^{n-1} - 2^{n-x-1} - \ln 2 \cdot (n-x) \cdot 2^{n-x-1} = 0$ for x, we have

$$x = \frac{-LambertW(e^{\ln 2 \cdot n + 1}) + \ln 2 \cdot n + 1}{\ln 2} \tag{7}$$

where $LambertW$ is a special function that satisfies

$$LambertW(x) \cdot e^{LambertW(x)} = x$$

The numerical values of x as a function of n is given in Figure 7. To see the trend of the function we chose the range of n from 1 to 64. But given current

technology, 64 is too big a number for the dimension of hypercube. A more practical range of n is from 5 (32 nodes) to 20 (1,048,576 nodes). Since the calculation of $x = (-LambertW(e^{\ln 2 \cdot n+1}) + \ln 2 \cdot n + 1)/\ln 2$ is not a light undertaking, for implementation we could pre-store the table in Figure 7.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	0.459	0.8469	1.177	1.463	1.713	1.933	2.13	2.307	2.467	2.614	2.749	2.873	2.989	3.101	3.199	3.294

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
3.391	3.463	3.549	3.621	3.694	3.766	3.838	3.896	3.968	4.025	4.083	4.126	4.184	4.242	4.285	4.328

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
4.386	4.429	4.473	4.516	4.559	4.588	4.631	4.675	4.704	4.747	4.776	4.819	4.848	4.877	4.92	4.949

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
4.978	5.006	5.035	5.064	5.093	5.122	5.151	5.18	5.208	5.237	5.252	5.281	5.309	5.338	5.353	5.382

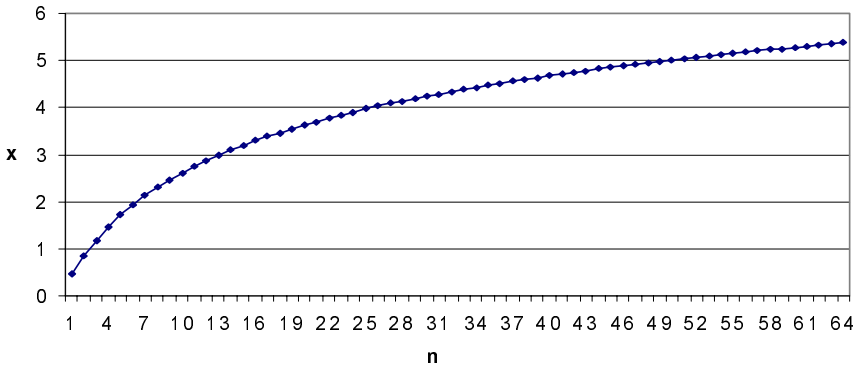


Fig. 7. Values of x as a function of n. Note that x grows very slowly.

It is worth pointing out that the optimal subcube size grows very slowly as the n-cube grows. More specifically, for all practical cubes (e.g., from 5-cube to 20-cube), the optimal subcube dimension can only be 2, or 3, or 4. This property can be used when constructing multiple-level hierarchical monitoring systems.

The saving of communication cost gained by the optimal two-level scheme is shown in Figure 8, which depicts the ratio of min-two-level-cost and one-level-cost. It shows a pattern of behavior similar to the case of mesh: the ratio is ever decreasing as the cube size is increasing. When $n = 5$, ratio is about 55%; when $n = 10$, about 39%; when $n = 20$, about 25%.

4.2. Multiple-level partitioning

Figure 9 shows the ratio of 1st-level and 2nd-level costs, given the optimal subcube size.

Like in the case of mesh, the cost of 1st-level is always bigger than that of 2nd-level. (The difference is, however, that the ratio is not a fixed value as in the case of mesh: it grows as n grows.) To construct a multiple-level hierarchical monitoring system, we could repeatedly do two-level partitions, as we did for mesh.

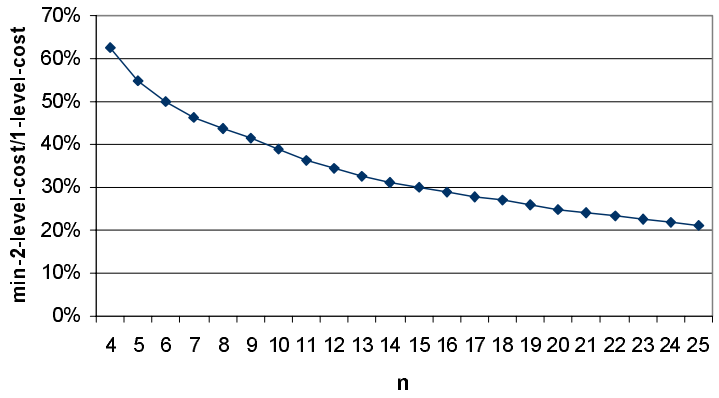


Fig. 8. Ratio of minimum two-level cost and one-level cost for hypercube.

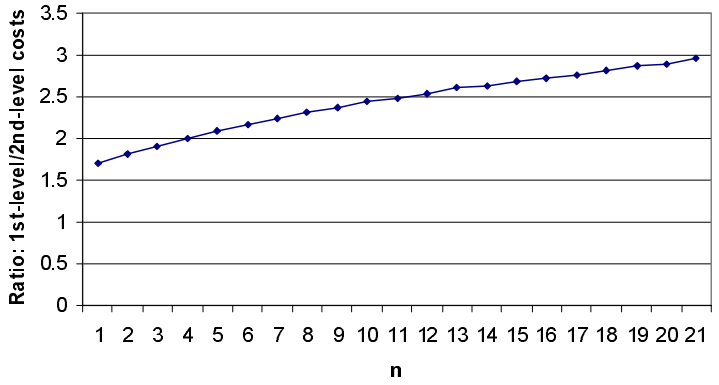


Fig. 9. Ratio of 1st-level and 2nd-level costs given the optimal subcube size.

An alternative approach is to make use of the property that the optimal subcube size can only take value of 2, 3, or 4 for any practical cube. Since the set of possible subcube sizes is small, an "exhaustive" trial can be carried out to find an optimal partition. In the following algorithm, m is the number of levels one wants to implement.

$minCost \leftarrow \infty$ /* initial $minCost$ */
for ($c_1 = 2, 3, 4$)
  for ($c_2 = 2, 3, 4$)
    ·····
      for ($c_{m-1} = 2, 3, 4$) {
        if ($c_1 + c_2 + \cdots + c_{m-1} < n$) {
          compute *currentTotalCost* with c_i-cube at level-i, $1 \leq i \leq m-1$, and $(n - (c_1 + c_2 + \cdots + c_{m-1}))$-cube at level-$m$
          if (*currentTotalCost* $<$ *minCost*) {
            $minCost \leftarrow$ *currentTotalCost*
            keep track of current $(c_1, c_2, \cdots, c_m)$
          }
        }
      }

return *minCost*, $(c_1, c_2, \cdots, c_m)$

The observation made for mesh also holds here: the rate of cost reduction drops significantly as each new level is added. Therefore, for all practical n-cubes, we have a hierarchical system with a small number of levels. That means the for-loops in the algorithm will nest only a small number of times (say 3, 4, or 5), making the algorithm complexity tractable.

5. Conclusion

We have studied the problem of hierarchical configuration for distributed systems on mesh and hypercube. Non-exact algorithms, which include approximate and heuristic algorithms, yield feasible solutions that cannot be guaranteed to be optimal. They are feasible alternatives if finding optimal solutions with acceptable cost is impossible. On the other hand, exact solutions may be feasible only under special conditions and assumptions. In this paper, we have proposed hierarchical configuration schemes for two popular interconnection networks, mesh and hypercube, based on analytical results for optimal two-level partition. The absolutely optimal multiple-level partition could be achieved by extending the results for two-level partition. However from an algorithmic point of view we resort to repeatedly

applying two-level partitions to obtain multiple-level hierarchy. An important property revealed by the analysis is that for most practical meshes/hypercubes, a modest number of levels will already achieve almost all the cost reduction, i.e., very soon we will reach the point where adding more levels will not gain any meaningful reduction. This fact is very useful in controlling the algorithm's complexity.

When we dealt with mesh in this paper, we assumed its shape squared. Although it is an assumption in many research works, meshes of rectangular but non-squared shape are commonplace. The optimal partition of non-squared meshes would be an interesting research topic, of which the results of this paper will be a special case. Another possible extention of this paper's work is to look into the meshes of large, prime-numbered sizes, e.g., a mesh of 31×31. Under the current scheme, we could just use equation $\sqrt[3]{2N}$ to get a set of submeshes of unequal sizes. But there might be an altogether new way to divide prime-numbered sizes that can achieve better reduction.

As we mentioned in the Introduction, although this paper is presented in specific contexts (e.g, in terms of distributed monitoring), the approaches developed here may be used to a broader range of hierarchical control/computational problems in distributed processing. Study of algorithms for different forms of hierarchical configurations could be a direction to which the current work can be extended. Another immediate future research can be the simulation study of the effect of assuming unit cost (for analytical tractability) against the more realistic variable traffic cost incurred in the network. Finally, this paper has considered the hierarchical configuration of networks only theoretically. Since optimizing network performance using hierarchical levels is already a commonly adopted approach in practice, experimental measurements of improvement on real machines/networks would be useful to show practical relevance of this strategy.

References

1. D. Agrawal and A. El Abbadi, "An Efficient and Fault-Tolerant Solution to Distributed Mutual Exclusion," *ACM Transactions on Computer Systems*, Vol. 9, No. 1, Feb. 1991, pp.1-20.
2. I. Ahmad, A. Ghafoor, and G. C. Fox, "Hierarchical Scheduling of Dynamic Parallel Computations on Hypercube Multicomputers," *Journal of Parallel and Distributed Computing*, Vol. 20 (1994), pp.317-329.
3. 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.
4. J. Cao, K. Zhang, and O. de Vel, "On Heuristics for Optimal Configuration of Hierarchical Distributed Monitoring Systems," *Journal of Systems and Software*, Elsevier Science Inc., New York. Vol. 43, No. 5, 1998, pp. 197-206.
5. J. Cao and F. Zhang, "Optimal Configuration in Hierarchical Network Routing," *Proc. 1999 IEEE Canadian Conference on Electrical and Computer Engineering*, Edmonton, Alberta, Canada, May, 1999. pp. 249-254.
6. C.-H. Edward Chow, "Resource Allocation for Multiparty Connections," *J. System*

Software, 1995, Vol. 28, pp. 253-266.

7. W.J. Dally "Performance Analysis of k-ary n-cube Interconnection Networks," *IEEE Transactions on Computers*, Vol. 39, No. 6, pp. 775-785, June 1990.
8. G. Feitelson and L. Rudolph, "Distributed Hierarchical Control for Parallel Processing," *Computer*, pp. 65-77, May 1990.
9. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.
10. D. Haban and D. Wybranietz, "A Hybrid Monitor for Behavior and Performance Analysis of Distributed Systems," *IEEE Transactions on Software Engineering*, Vol. 16, No. 2, pp. 197-211, February 1990.
11. J.K. Hollingsworth and B.P. Miller, "Dynamic Control of Performance Monitoring on Large Scale Parallel Systems," *Proc. International Conference on Supercomputing*, Tokyo, July 1993, pp. 235-245.
12. J. Joyce, G. Lomow, K. Slind, and B. Unger, "Monitoring Distributed Systems," *ACM Transactions on Computer Systems*, Vol. 5, No. 2, pp. 121-150, May 1987.
13. M. J. Mataric, "Using Communication to Reduce Locality in Distributed Multi-Agent Learning," *J. Experimental and Theoretical Artificial Intelligence*, Vol. 10, No. 3, 1998. pp.357-369.
14. B.P. Miller, C. Macrander, and S. Sechrest, "A Distributed Programs Monitor for Berkeley UNIX," *Software – Practice and Experience*, Vol. 16(2), pp. 183-200, February 1986.
15. O. Ogle, K. Schwan, and R. Snodgrass, "The Real-Time Collection and Analysis of Dynamic Information in Distributed and Parallel Systems," Technical Report, Computer and Information Science Research Center, The Ohio State University, August 1987.
16. C.-C. Shen and W.-H. Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems Using a Minimax Criterion," *IEEE Transactions on Computers*, Vol. C-34, No. 3, pp. 197-203, March 1985.
17. 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.
18. L. Shi, J. Cao, and O. de Vel, "A Hierarchical, Distributed Monitoring System For Interprocess Communications," *International Journal of Computer Systems: Science and Engineering*, CRL Publishing Ltd. (14), Sep. 1999. pp. 317-325.
19. 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.
20. M. Steenstrup, Routing in Communications Networks, Prentice Hall, Inc. 1995.
21. 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.
22. Y. Zhu, "Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers," *Journal of Parallel and Distributed Computing*, No. 16, pp. 328-337, 1992.
23. F. Zhang and J. Cao, "Hierarchical Configuration of Monitoring Units in a Tree-structured Distributed System," *Proc. 1999 IEEE International Conference on Systems, Man and Cybernetics*, Japan, Sep., 1999.