Universität Stuttgart
Fakultät Informatik

Partitioning and Mapping Techniques for Distributed Multimedia Applications

M. Ashraf Iqbal, Alexander Hagin

CR-Klassification: C.2.4, C.4, G.1.6, G2.2, I6

Partitioning and Mapping Techniques for Distributed Multimedia Applications

Fakultätsbericht 14/1996
Technical Report
Dezember 1996

Fakultät Informatik
Institut für Parallele und
Verteilte Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstraße 20 - 22
D-70565 Stuttgart

Partitioning and Mapping Techniques for Distributed Multimedia Applications

M. Ashraf Iqbal[1]
Institute of Parallel & Distributed Systems (IPVR)
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
University of Engineering & Technology, Lahore 54890, Pakistan

Alexander Hagin
Institute of Parallel & Distributed Systems (IPVR)
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
St. Petersburg State Technical University, St. Petersburg, Russia

Abstract

Distributed Computer Systems have become competitive in providing large amounts of computational power at a very low cost. The system consisting of personal computers, mini or main frames, or high performance multiprocessors, integrated into a high speed computer network is capable of providing any organization the power of a super machine with only a small initial cost. Such a system has the further benefit of providing industry with an easy and modular upgrade path because increasing the power of the system simply involves increasing the number of networked computers. Distributed Multimedia Applications (DMA) are so rich in their diversity of methodology and so inherently computational intensive that they naturally require a very heterogeneous mix of distributed processors interconnected by an efficient interconnection network. A DMA can be represented by a precedence graph, where nodes represent components (or modules) of the application, interconnected by arcs representing data streams flowing between different components. In this paper we study the problem of partitioning and mapping Distributed Multimedia Application graphs onto a heterogeneous distributed computer system. Such applications require a compromise between quality of service and cost of the utilized resources of the distributed computer system. As the problem is difficult to solve, in general, we present an approximate scheme which optimally assigns task modules of the application onto the processors of the distributed system.

[1] This research was supported by a research grant of the University of Engineering & Technology, Lahore, Pakistan. Additional support was provided by the German Academic Exchange Service (DAAD).

1 Introduction

Recent research in parallel and distributed processing technology has resulted in many advances in all directions of computing technology, e.g., in device technology, networking capability, and in software engineering [1]. Research in device technology has resulted in faster and more powerful processors. Advances in computer network capability have introduced powerful, faster, and more reliable networks. Research in software has provided user friendly tools and environments. These advances have the potential to satisfy the ever growing computation needs of many scientific and engineering applications, e.g., now it has become economically possible to process digital audio and video signals in real time leading to the development of distributed multimedia systems.

Distributed Multimedia Applications (DMA) demands an efficient framework for its implementation over a Distributed Computer System (DCS). In fact DMA problems are so rich in their diversity of methodology and so inherently computational intensive that they naturally require a very heterogeneous mix of distributed processors interconnected by an efficient network. A DMA can be represented by a precedence graph, where nodes represent components (or modules), interconnected by arcs representing data streams flowing between different components of the application. Each node of the graph is weighted by computation requirements of the corresponding component and each arc is weighted by the channel capacity needed for the communication between adjacent components. Multimedia streams can originate at multiple sources, traverse a number of intermediate components and end at multiple sinks [2,3].

The Distributed Computer System (DCS) is a heterogeneous mix of mini, microcomputers, or workstations interconnected through a point to point physical link, Local Area Network (LAN) and or a Wide Area Network (WAN). The DCS is also represented by a graph where nodes represents individual machines while edges represents virtual channels of the DCS. Each node of this graph has a weight associated with the available computation capacity. Similarly there is a weight associated with each edge, it signifies the available capacity of the corresponding channel [3, 4, 5].

By partitioning the application task onto different machines that communicate over the network, components or stages of DMA can be executed simultaneously on the machines to which they are best suited in terms of cost and time constraints of execution [6, 7, 8]. Thus a network of distributed machines may be able to provide an optimal performance for such an applications. Efficient utilization of such an approach depends on a number of issues like modeling the DCS as well as the DMA for algorithm development, design of partitioning and mapping strategies, and integrating these strategies into existing programming systems to solve computationally intensive problems [1, 6, 7]. We would like to address some of these problems in this paper, in particular we shall attack the following problem: Given a set of M components of the multimedia application connected in some fashion, and a distributed computer system consisting of different machines, find an assignment of components to processors that minimizes the cost of using the computational as well as communicational resources such that the load on every machine is bounded by a fixed number while the total communication overhead on the network is also kept below its capacity. Note that the last constraint, i.e., the total communication requirement should be kept below the total capacity of the network, makes it possible for us to model the DCS graph as a completely connected structure. We should also try to keep the load on every machine in DCS below a certain level in order to provide an adequate quality of service for the multimedia applications.

If the number of processors are only two and we want to minimize the total cost of execution plus communication (i.e., without any other constraints) then it is possible to solve this problem using the network flow approach pioneered by Stone [7, 15]. If the interconnection structure of the DMA graph is chain or tree like, it is still possible to solve the problem for an arbitrary number of processors using a shortest tree algorithm designed by Bokhari [6, 7]. Towsley [7, 19] has shown how to find an optimal partitioning for a series-parallel graph using a series of graph transformations. It is important to notice that the general partitioning and mapping problem is very difficult to solve. Some of the researchers have thus solved the partitioning problem by putting constraints on the structure of the application graph while others have found an optimal solution by restricting the number of processors. In [3] an approach based on branch and bound method is proposed, however, the algorithm complexity restricts the dimensions of DMA and DCS that can be handled by the algorithm.

If, on the other hand, the problem is to minimize the load on the most heavily loaded processor in a conventional parallel processing environment then the problem, in general, is very difficult to solve using exact algorithms and this explains why most of the work in this field focused on heuristic techniques [5, 9]. If the structure of the application graph is as simple as a chain and the number of processors are confined to only two even then the problem is difficult to handle as reported in [10, 11, 13, 14]. A number of researchers have, however, attempted to solve the problem under certain restrictions on possible partitionings as described in [7, 10, 13]. Some of the graph theoretical research, conducted in the past was directed to find a general method for partitioning the vertices of a graph into two sets of prescribed sizes by the removal of minimum number of edges [16]. A number of researchers have also developed parallel algorithms for partitioning series-parallel and bandwidth-*k* graphs [17]. Solutions to such problems are also helpful in designing partitioning algorithms or approximate schemes for parallel and distributed computer systems.

Approximate techniques for partitioning chain and tree structured image processing tasks onto heterogeneous distributed computer systems have also been reported in [13, 14]. Iqbal [8, 11] have devised fully polynomial time approximation schemes in order to solve the partitioning and mapping problem approximately. The time complexity of such an scheme is polynomial in both the size of the problem as well as in *1/e,* where *e* is the relative error bound for the approximate scheme. In order to appreciate the usefulness of the approximate solutions, one should bear in mind that data for the problem being solved is often only approximately known as the procedures of code type profiling and analytical benchmarking are still in their infancy. Hence approximate solutions may be as meaningful as an exact solution for many of the practical problems where the extra accuracy of the exact solution is not needed and where the approximate solution can be obtained in a relatively short time [8, 11].

In this paper we would consider the problem of partitioning DMA whose graphs are restricted to chain or tree like structures, and would present approximate solutions. The algorithms, that we present here are derived from the earlier work of Bokhari [4, 6, 7] and Iqbal [8, 10, 13, 14] who solved partitioning problems in sequential as well as in parallel processing environments using distributed machines with dedicated communication links. It is important to note that now we are concentrating on distributed computers interconnected using a general purpose LAN interconnection which, according to our opinion, would become a cost effective, efficient and commonly used resource in the coming future.

The paper is organized as follows. The next section formulates the partitioning and mapping model and defines the cost functions addressed in this paper. We also model the DMA and the DCS in this section. Section 3 addresses the partitioning problem. The results and conclusions are summarized in Section 4.

2 Problem Formulation

2.1 Model of the Distributed Multimedia Applications

Distributed multimedia applications are employed to generate, process, and consume continuous (e.g. audio, video) data streams. DMA topology can be constructed by specifying components interconnected via links. Components encapsulate processing of multimedia data, e.g., for generating (source components), consuming (sink components) or manipulating (filters and mixers) data. A component is an individually schedulable unit (e.g., by mapping to a thread). A link provides an abstraction from underlying communication mechanisms which may be used to perform the transport of data units.

To provide a uniform data access point for the components, ports are used that deliver data units to the component (input port) or take the data units from the component (output port). A component designer has to associate with each component port the streamtype to be used, thus making all related information available at the port.

A DMA can be represented by one or more precedence graphs [3]. In a DMA graph, nodes represent components that are interconnected by arcs representing data streams between components. Each component is associated with at least one device that produces (a source component) or processes (an intermediate component - filter or mixer) or consumes (a sink component) data streams. Media streams can originate at multiple sources, traverse a number of intermediate components and end at multiple sinks.

Before using an application, desired user QoS (Quality of Service) is specified with respect to output data generated by sink components (e.g. presented video frame size and rate). To guarantee the specified QoS requirements, corresponding resources for DMA components and links mapped to a distributed computer system have to be reserved. Thus, each node of the application graph is weighted by computational requirements of the corresponding component and each arc is weighted by the channel capacity needed for remote communication between adjacent components.

Let us consider a DMA graph and determine the values of node and arc weights. Every node is weighted by computational requirements of the corresponding component of the DMA. Let λ_i denote the arrival rate (messages per second) of input data streams to component i in the DMA graph. To process every message, component i needs V_i processor operations. Let C_i denotes the component computational requirement (operations per second). To exclude unlimited queue of the messages, it is necessary that $C_i > \lambda_i V_i$.

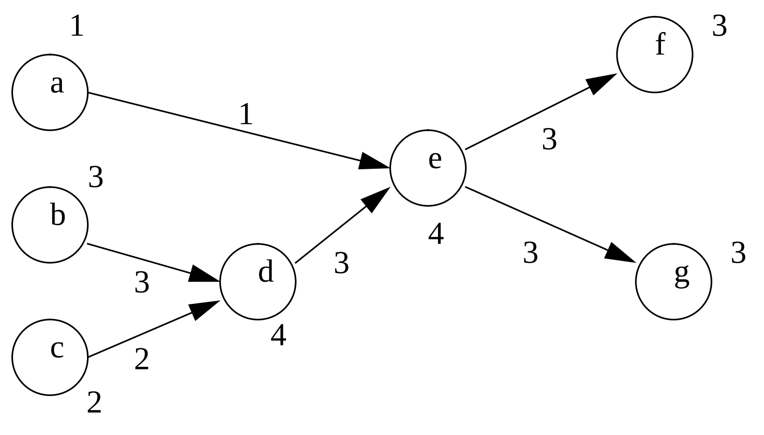

Figure 2.1. An example of DMA graph

Every arc (i, j) in the DMA graph is weighted by capacity requirement C_{ij} (bits per second) that must satisfy the inequality $C_{ij} > \lambda_i L_i$, where L_i is the length of the message (bits) arrived at the corresponding link of the DMA graph.

An example of a DMA graph is presented in Figure 2.1. The topology of DMA is composed of three source-components a, b and c connected to two mixing components d and e, last of which provides data streams to two sink components f and g. Weights at nodes and arcs denote computational and communication resource requirements respectively.

2.2 Model of the Distributed Computer System

A DMA graph can be arbitrarily distributed over several nodes of a distributed computer system. Generally, set of computers on which a component can be assigned depends on whether the computer configuration has devices and resources required to perform multimedia functions needed by a certain component type. On the other hand, some of the source-components and/or sink-components can only be assigned to certain computers in advance, these components are called pre-attached ones.

Let us consider a graph of a DCS. Every node n is weighted by available computational resource R_n (operations per second). If B_n is the total computational resource of computer n in the DCS and b_n is the computational resource already used by all other applications processed in the DCS, then the available computational resource of computer n is $R_n = B_n - b_n$

The graph representation of the DCS shows possible *virtual channel connections* (VC) between the computers of the DCS. A VC is a direct oriented logical connection between two computers (endsystems) with some assigned capacity. A VC is routed over one or more communication resources of the DCS (physical links, networks) to achieve sender-computer to receiver-computer connectivity. The available capacity of a VC is equal to minimum available capacities of all DCS communication resources over which the VC is routed. Let A_s be the available capacity of DCS communication resource s; ρ_{nm} be the set of DCS communication resources used by VC *(n,m)* from computer n to computer m of the DCS. Then the available capacity of the VC *(n,m)* is given by

$$R_{nm} = min\{A_s, s \in \rho_{nm}\} \tag{2.1}$$

Figure 2.2(a) illustrates an example of a DCS structure that is represented by the logical system graph shown in Figure 2.2(b). Here the capacities available for every computer pair connection are as follows:

$R_{12} = min\{A_1^{out}, A_2^{in}, A^{LAN1}\}$, $R_{13} = min\{A_1^{out}, A_3^{in}, A^{LAN1}, A^{WAN}, A^{LAN2}\}$, etc.,

where A_n^{out}, A_n^{ir} are available capacities of the output and input interfaces of computer n,

Further every arc *(n,m)* of the system graph represents corresponding VC *(n,m)* of the DCS and is weighted by available capacity R_{nm} (bits per second) of the VC (n, m).

It is important to note, that communication resources in the DCS can be shared between different VCs. For example, the capacity of the LAN Ethernet does not belong to any pair of computers but is distributed among all computers of the LAN. The LAN provides a virtual channel between any pair of computers. Therefore, a system graph for the LAN is a logical graph representing all possible VCs between computers connected to the LAN (see Figure 2.3). Available capacity A of the LAN transmission line is distributed among all data-exchanging computer pairs. Therefore the following inequality has to be satisfied:

$$0 \le \sum_{(n, m)} R_{nm} \le A \tag{2.2}$$

The capacity A is available for every possible VC in the system graph. It means that if the available capacity of the LAN, for example, is decreased by a, then the available capacity of every possible virtual channel with $R_{nm} = A$ decreases by the same amount. Equations (2.1) and (2.2) specifies the relationship between capacities of the communication resources of the DCS virtual channels, represented by arcs in the system graph.

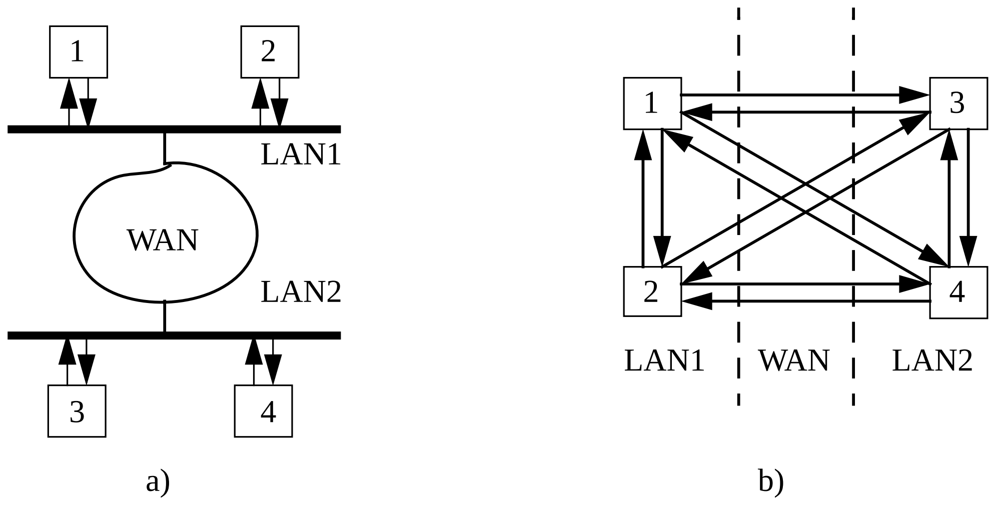

Figure 2.2, a) Communications in the DCS take place through various networks,
b) Representation of computer communications through VCs in the system graph

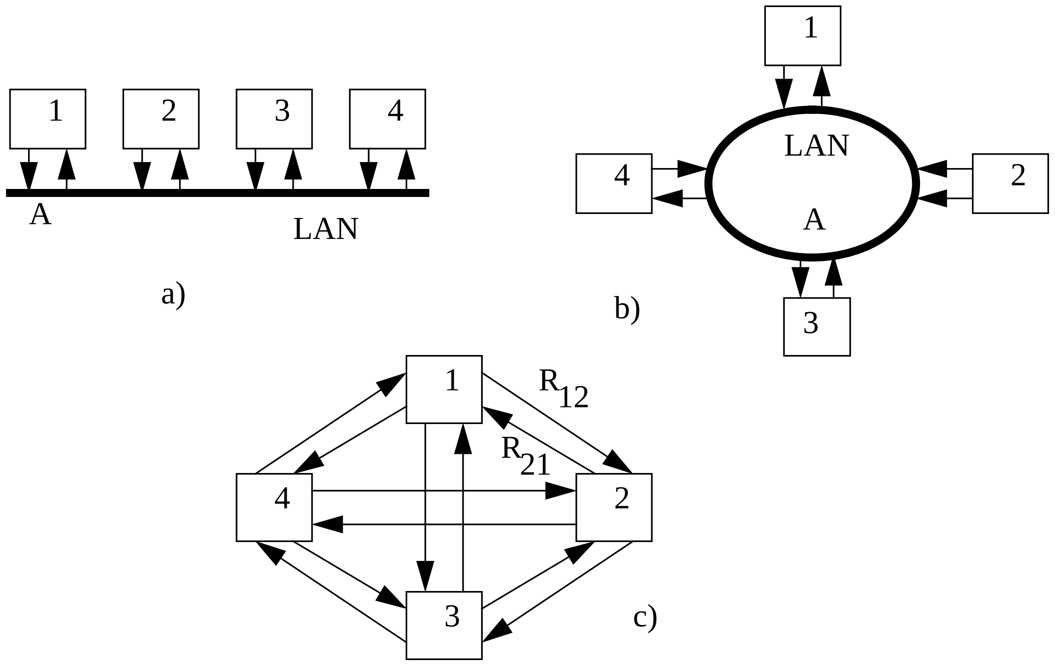

Figure 2.3,a,b) Connections in the DCS through local area network,
c) Representation of computer communications through VCs in the system graph

2.3 Cost functions

There are different ways to partition and map a DMA graph over the DCS graph. We should select the one that meets QoS requirement at minimal cost. In order to calculate the cost functions we must take into account the following:

1. The DCS is heterogeneous, i.e., the computers can differ with respect to their power and capacity of available resources, the virtual channels between the computers can be provided by various mediums of communication in the DCS.
2. Each component of the DMA can be implemented in different ways on different computers, e.g., by hardware, software, or a mixture of the two. Moreover different kinds of components can differ from each other in types and sizes of required computer resources (CPU slots, catche and disk memory, bus capacity, etc.)

Thus cost of different permissible component allocations to computers will be represented by cost matrix $f = \{f_n^i\}$ with entries f_n^i denoting cost of allocation of component *i* to computer *n*.

Suppose a cost function $g_s(x)$ for every communication resource *s* of the DCS is given. Then the cost of mapping a DMA link *(i,j)* to virtual channel *(n,m)* of the DCS can be computed by the formula:

$$g_{nm}^{ij} = \sum_{s \in \pi_{nm}} g_s(d_{ij})$$

where π_{nm} is set of communication resources, which channel *(n,m)* is routed over; d_{ij} is required communication capacity of the DMA link *(i,j)*.

2.4 Problem Statement

The general problem formulation of partitioning and mapping a DMA graph to a DCS graph is as follows. We are given the following information [2]:

1. A DMA graph with

   η - a set of nodes (components or modules),

   λ - a set of directed arcs connecting components with each other, $\lambda = \{(i, j), i, j \in \eta\}$,

   d_i - a required computational resource for every component $i \in \eta$

   d_{ij} - a required communication capacity for every link $(i, j) \in \lambda$,

2. A DCS graph with

   ζ - a set of nodes (each node represent a computer) ,

   π - a set of VCs (or simply channels) connecting computers with each other, $\pi = \{(n, m), n, m \in \zeta\}$,

R_n - an available (vacant) computational resource of every computer $n \in \zeta$,

ρ - a set of communication resources in the DCS[1],

ρ_{nm} - a set of communication resources of the DCS used by channel (n, m) , $\rho_{nm} \in \rho$, $\bigcup_{(n, m) \in \mu} \rho_{nm}$,

π_s - a set of channels routed over shared communication resource $s \in \rho$, $\bigcup_{s \in \rho} \pi_s = \pi$,

A_s - a capacity of a communication resource s available to the mapped DMA, $s \in \rho$,

ζ_i - a set of acceptable locations of every component $i \in \eta$ in the DCS ,

3. Cost functions

f - a cost matrix, an element f_n^i of f specifies the cost of mapping component i to computer n,

g - a cost matrix, an element g^{ij} of g specifies the cost of mapping DMA link *(i,j)* to virtual channels *(n,m)* of the DCS.

The solution variables are x_{in} such that $x_{in} = 1$, if component i is assigned to computer n, and $x_{in} = 0$, otherwise.

The Partitioning and Mapping Problem can then be defined as:

$$F(x_{in}) = min_{x_{in}} \left\{ \sum_{i \in \eta} \sum_{n \in \zeta_i} x_{in} f_n^i + \sum_{(i, j) \in \lambda} \sum_{(n, m) \in \pi} x_{in} x_{jm} g_{nm}^{ij} \right\} \tag{2.3}$$

subject to

$$\sum_{n \in \zeta_i} x_{in} = 1, \forall i \in \eta \tag{2.4}$$

$$\sum_{i \in \eta} x_{in} d_i \le R_n, \forall n \in \zeta \tag{2.5}$$

$$\sum_{(n, m) \in \pi_s} \sum_{(i, j) \in \lambda} x_{in} x_{jm} d_{ij} \le A_s, \forall s \in \rho \tag{2.6}$$

where g_{nm}^{ij}=0 if $n = m$; $g_{nm}^{ij} \ge 0$ if $n \ne m$ and $(n, m) \in \pi$; $g_{nm} = \infty$ if $(n, m) \notin \pi$.

In this formulation, objective function F minimizes the total cost of computational and communication resources used for the DMA assignment onto the DCS. The first term in the objective function identifies the cost of computer resources that are used to execute components of the DMA. The second term represents the cost of communication resources of channels on which DMA arcs are placed.

Constraint set (2.4) guarantees that every component $i \in \eta$ will be placed into the DCS and only onto one computer. Constraint set (2.5) guarantees that resources used by components assigned to a computer do not exceed the available resource of the computer.

[1] Computer interfaces in the DCS can also be included into the set ρ .

Constraint set (2.6) guarantees that capacity of communication resource s in the DCS used by all DMA arcs placed on resource s do not exceed the available capacity of the resource.

3 Partitioning & Mapping Schemes

In this section we would describe efficient partitioning and mapping schemes which can be used for partitioning chains or tree structured distributed multimedia applications. We intend to extend the approach of our algorithm for series-parallel graphs which is another important class of applications in multimedia systems. The algorithm is derived from the earlier work of Bokhari [4, 5, 6, 7] and Iqbal [8, 10, 13, 14], who solved partitioning problems in sequential as well as parallel processing environments using distributed machines with dedicated communication links. Here we are concentrating on distributed computers interconnected using a general purpose LAN interconnection. As discussed before our objective is to minimize the sum of execution and communication costs with the constraint that the total communication requirement over the network is bounded. We work under the assumption that the available computing resources of every machine in the DCS are enough to satisfy the requirements, we provide an approximate solution to the partitioning and mapping problem taking into account only the communication constraint and minimizing the total cost. It is possible to design efficient heuristics based on this approach which can take into account the additional resource constraint on the load assigned to machines in the distributed computer system.

3.1 Partitioning Chain Structured Applications

We show a chain structured application graph in Fig. 3.1. It consists of a source, a sink, and one or more intermediate components. Every data unit (video frame or audio sample) of media stream is generated, processed, and consumed by source, intermediate (e.g., a filter), and sink components respectively. Should a module resident on one machine communicates with a module resident on another computer, there will be an overhead of inter computer communication through the network. This will not only burden the communication resource but would also incur a cost proportional to the amounts of data transmitted between the two computers. The cost of transmitting data between two coresident modules is assumed to be negligible. This assumption can also be relaxed to take into account nonnegligible intracomputer communication costs.

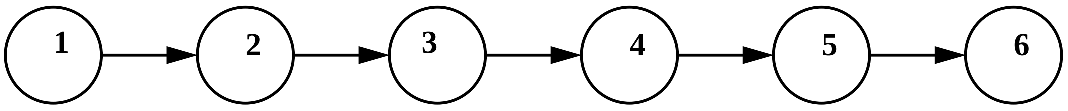

Fig. 3.1 A chain structured Distributed Multimedia Application.

The Assignment Graph

Given the invocation chain of Fig. 3.1, and the execution and communication costs, we can draw an assignment graph as shown in Fig. 3. 2. This figure assumes a three processor system. It is important to note that the following observations apply to the assignment graph:

1. There is one special node called the start node, denoted by *s*, and there is an end node denoted by *t.*

2. In addition to the start and the end nodes there are $M \times N$ further nodes in the assignment graph, where each node is labelled with a pair of numbers *(i,p)*, and represents the assignment of module *i* to processor *p*.

3. A directed path from the start node to the end node in this graph corresponds to a partitioning of the chain structured program over the distributed computer system. Similarly any partitioning of the chain can also be represented by a path in this graph as shown in Fig. 3.2 in bold.

4. There is an ordered pair, *<a,b>* of weights attached with each edge.

5. All edges incident on the end node have zero weights on them, i.e., both elements of the ordered pair are zero.

6. The edge joining the start node to node *(1,p)* have an ordered pair, *<a,b>*, associated with each edge where $a = f_p^1$ and $b = 0$.

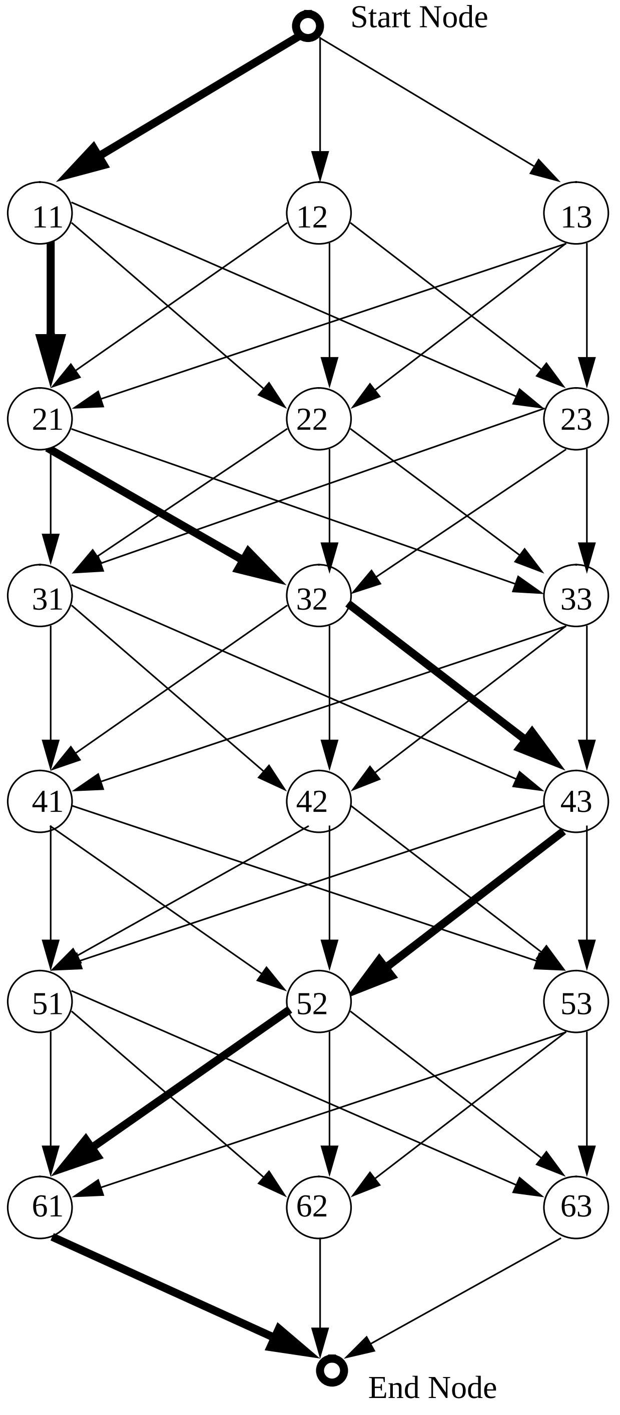

Fig. 3.2 An assignment graph for the chain of Fig. 3.1 and a three computer system.

7. The edge joining node *(i,p)* to node *(j,q)* has an ordered pair, <*a,b*>, where $a = f_q^j + g_{pq}^{ij}$ and $b = d_{ij}$.

8. It is important to note that each path from the start node to a node *(i,p)* in the assignment graph corresponds to an assignment of module 1 to module *i* onto the machines of the distributed computer system. The path is composed of a number of edges and if we now sum up the first elements (i.e. *a*'s) of the ordered pair associated with each edge in the path then it would specify the total cost of assigning the first *i* modules of the chain structured application, this will include execution as well as communication costs. We denote this sum by *F.*

9. If we sum up the second elements (i.e. *b*'s) of the ordered pairs associated with each edge in the path then the corresponding sum would specify the total communication requirements of assigning the first *i* modules of the application onto the distributed network system. We denote this sum by *D*.

10. Consider two distinct but partial assignments (or two paths between two nodes in the assignment graph) of the chain structured application. Let the two assignments are denoted by *P1* and *P2*. If *D(P1)* as well as *F(P1)* is less than or equal to the respective values of *D(P2*) and *F(P2)* then the path *P2* is just redundant and there is no need to consider the assignment corresponding to this path. So we just ignore the so called redundant path *P2*. Note that we are in a position to reject a number of partial paths which satisfy the above constraint without extending these paths to the start or the end nodes. This facility provides us to cut down the complexity of the problem and reduce our search from an exponential number of paths to a clean polynomial expression as described in the partitioning scheme.

11. Consider the possibility that *D(P1)* is less than *D(P2)* but *F(P1)* is larger than *F(P2)* then it is not possible to reject one path or the other as we are not sure which assignment would provide us best results, i.e., minimal cost of assignment such that the total communication requirement is bounded. We have to extend these paths to the start node on one side and the end node on the other side to make a final judgement, the information available until now is not sufficient to make a comparison.

The Partitioning Scheme

Let C_T represents the maximum possible value of the communication requirement of assigning the chain structured application on the distributed computer network. It is evident that it would be equal to the sum of all the d_{ij} 's in the chain structured application. We now resolve C_T to an accuracy of *e,* i.e., two adjacent levels for the communication requirement are separated by *e.* In other words the total communication requirement is restricted to have only C_T/e distinct values in the range of zero to C_T. Now consider two paths *P1* and *P2* arriving at a node *(j,q)* in the assignment graph. Two possibilities exist:

1. The total communication requirements of the two paths lie into two distinct levels. *If D(P1*) as well as *F(P1)* is less than or equal to the respective values of *D(P2)* and *F(P2)* then path *P2* is just redundant and there is no need to consider the assignment corresponding to this path. If, however, *D(P1)* is less than *D(P2)* but *F(P1)* is larger than *F(P2)* then it is not possible to reject one path or the other as we are not sure which assignment would provide us best results, i.e., a minimal cost of assignment such that the total communication requirement is bounded.

2. The total communication requirement of the two paths lie in between two successive permissible levels. If *D(P1)* as well as *F(P1)* is less than or equal to the respective values of *D(P2)* and *F(P2)* then path *P2* is just redundant and there is no need to consider the assignment corresponding to this path. But if *D(P1)* is less than *D(P2)* and *F(P1)* is larger than *F(P2)* even then it is possible to reject one path as follows. Here we reject the path *P1* as the cost of assignment of *P1* would always be larger than the cost of assignment corresponding to *P2* with a guarantee that the error in the communication requirement of the selected path, i.e., *P2* would be bounded by *e.*

With the rejection techniques described above, the number of outgoing paths from a node would be restricted to C_T/e. Thus the maximum number of incoming paths would also be restricted to C_T/e from each processor. The total comparisons to be made at each node *(j,q)* of the assignment graph in order to reject unnecessary paths would be proportional to $O(NC_T/e)$ with the surety that the outgoing paths from the node would be restricted to at the most C_T/e under worst case circumstances.

As the number of modules in the chain structured application are equal to *M* then the total time complexity of the approximate scheme would be bounded by $O(N^2M\, C_T/e)$ with the guarantee that the maximum difference between the total communication requirement of the partitioning we found and the actual communication requirement of the optimal partitioning is at the most equal to *Me.* If the relative error bound for the user interested to use the approximate scheme is $\varepsilon = Me$ then the time complexity of the algorithm would be equal to $O(N^2M^2(C_T/\varepsilon))$. The approximate technique that we have described is thus a fully polynomial time approximation scheme in which the time complexity is a polynomial function of the size of the problem as well as the reciprocal of the total error ε.

3. The output of our algorithm would provide C_T/e partitionings of the chain structured application over the distributed computer system with different costs and communication requirements. It will, for example, include the special case when the communication requirement is very small but then it might be very expensive, similarly it will also accommodate a case when the communication requirement is large but the total cost is relatively small. The user will have the freedom to select the partitioning which he can afford provided the communication requirement is below the capacity of the network. The accuracy of communication requirement resolution is determined by the relationship $e = \varepsilon / M$, where the relative error bound ε can be selected on the basis of acceptable quality of service degradation permissable by the user.

The Resource Constraint on Load

The additional resource constraint on the load assigned to every machine can easily be plugged into the approximate scheme but then it no longer remains a fully polynomial time approximation scheme but would become a heuristic. Consider the assignment graph once again as shown in Fig. 3.2. Suppose that the available computational resource for a computer x is R_x, i.e., we should not assign load to computer x more than R_x. Now consider only those paths in the assignment graphs between the start node and any intermediate node in which the load assigned to every machine is below or just equal to its capacity. There exists two possibilities:

1. If the number of paths, inspite of the resource constraint on load, grows exponentially then it justifies our assumption used in this paper that the available computing resources of every machine is adequate enough to satisfy the requirements. Under such conditions the number of paths or assignments can be limited by selecting the one with minimal cost (with some fast priority scheme) as described in the assignment scheme.

2. If, on the other hand, the number of paths which satisfies the load constraint does not grow exponentially then it is some thing to be happy about. Under such conditions we can easily select the assignemnt of minimal cost.

In practise, however, the number of paths would initially have the tendency to grow exponentially but then the number would be limited due to the resource constraint on load. The actual behaviour would be determined by the particular application and how the heuristic is actually implemented.

3.2 Partitioning Tree Structured Applications

We show a tree structured application graph in Fig. 3.3. Two media streams starts from source component 5 and 6. The first one is processed by filter 4 and then they are mixed by component 3 which provides the mixed stream through filter 2 to sink component 1. It is also possible to consider an application with more than one sink components, what is essential at this point is that the graph of the multimedia application should be a tree. It is, however, possible to extend the approach to series-parallel graphs [7] and perhaps to some other restricted structures.

The Assignment Graph

The assignment graph of the tree structured application of Fig. 3.3, is shown in Fig. 3.4, we assume a three processor distributed computer system. There are a number of similarities and a number of important differences between this assignment graph and the one shown in Fig. 3.2. We would concentrate on important similarities as well as differences:

1. There are multiple number of start or source nodes and similarly there are multiple sink or end nodes.

2. Each assignment tree corresponds to an application assignment and each application assignment corresponds to an assignment tree. One such assignment tree in the assignment graph is shown in bold in Fig. 3.4.

3. If we sum up the first elements (i.e. *a'*s) of the ordered pair associated with each edge in the assignment tree then it would specify the total cost of assignment of the tree structured application, this will include execution as well as communication costs. We denote this sum by *F*.

4. If we sum up the second elements (i.e. *b's*) of the ordered pairs associated with each edge in the assignment tree then the corresponding sum would specify the total communication requirement of the said assignment. We denote this sum by *D*.

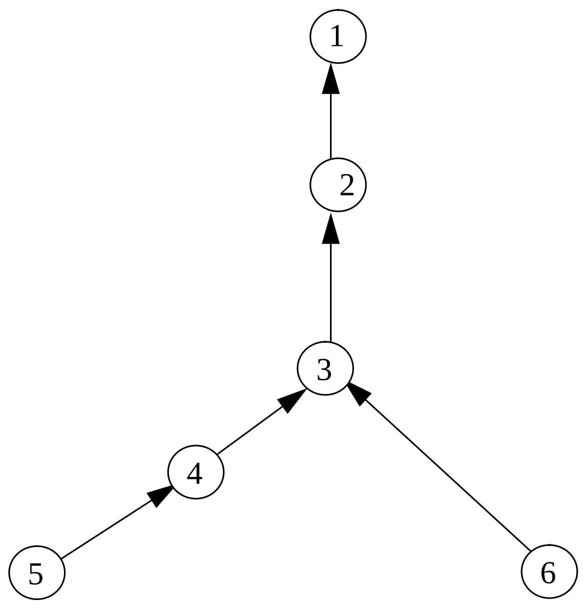

Fig. 3.3 A tree structured Distributed Multimedia Application.

The problem is then to find an assignment tree in the assignment graph with minimal cost in which the total communication requirement is bounded.

The Procedure Merge

We have already described how to partition a chain structured application over a distributed computer system such that the total cost of execution plus communication is minimal while the total communication requirement is within a certain bound. Here in this procedure we will discuss ways of utilizing the chain partitioning procedure to partition tree structured applications. Consider the tree shown in Fig. 3. 3, it is a tree and not a chain because the indegree of module 3 is 2. We will use the procedure Merge to handle such a situation. Consider the assignment graph again reproduced in Fig. 3.5. We can find approximate partitionings from the start node s_1 to every node in the layer, which corresponds to the module with an indegree more than 1. Here this would be layer 3, thus we have to find C_T/e paths or partitionings from s_1 to each

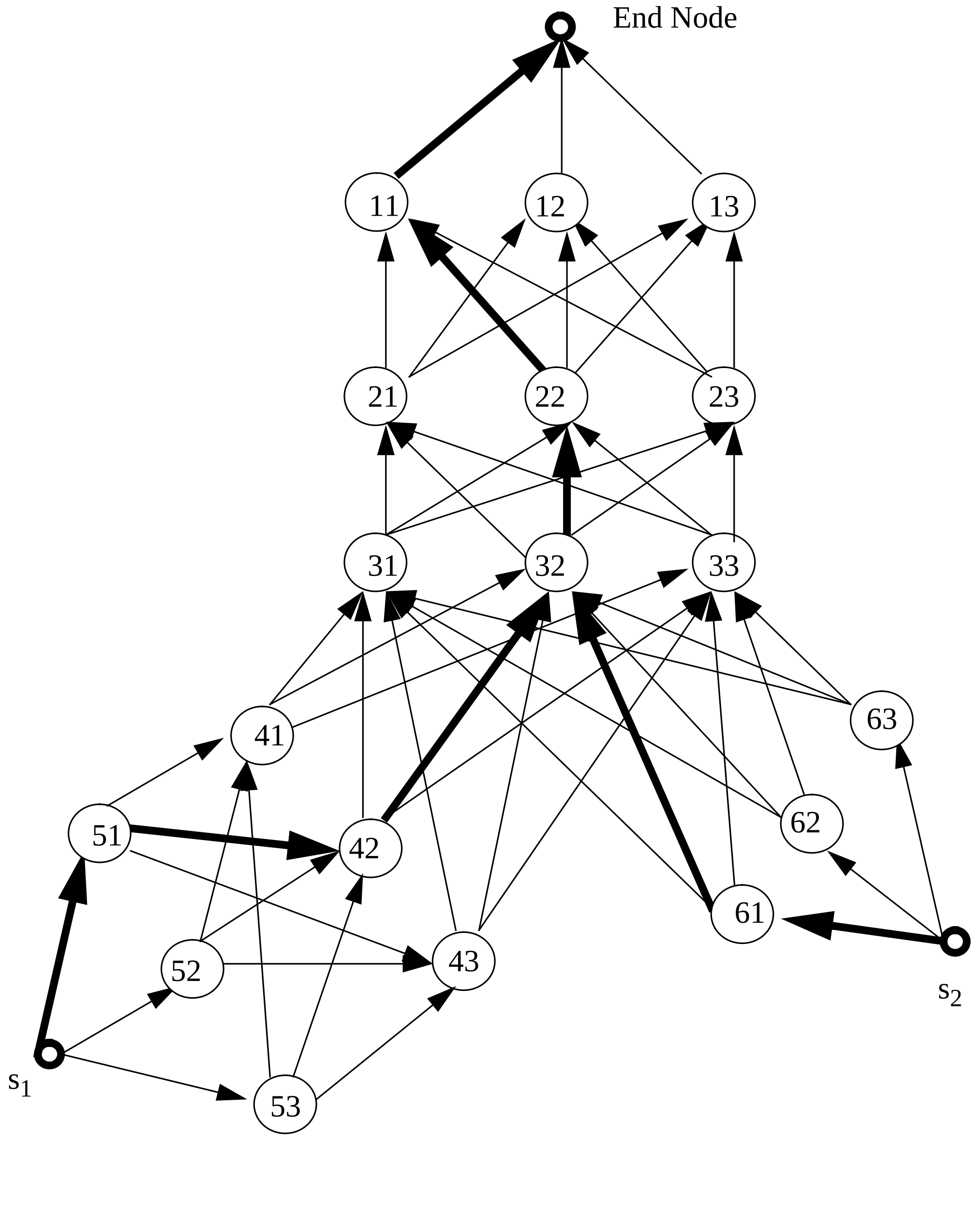

Fig. 3.4 An assignment graph for the tree of Fig. 3.3 and a three processor system.

node in the 3rd layer, i.e., node 31, 32, and to node 33 in the assignment graph. One such path or partitioning is shown, in bold, in Fig. 3.5 (top). We would have to make $N^2k(C_T/e)$ comparisons to find all the approximate partitionings from s_1 to every node in layer number 3 where k is the number of layers between s_1 and the layer number 3.

Similarly we can find C_T/e paths or partitionings from s_2 to each node in the 3rd layer, i.e., node 31, 32, and to node 33 in the assignment graph. One such path or partitioning is shown, in bold, in Fig. 3.5 (top). Still we have to make $N^2k(C_T/e)$ comparisons each time we find approximate partitionings from s_2 to node 31, 32, and to node 33 in the assignment graph. Node 32, for example, will have a number of C_T/e paths coming from s_1 and a similar number of approximate paths coming from start node s_2. Each path from s_1 to node 32 can be combined with a path from s_2 to node 32. By combination we mean that the two paths can be represented by an edge from a pseudo node *s* to node 32 as shown in Fig. 3.5 (bottom). Let *L1* represents a path from s_1 to node 32, and let us represent a path from s_2 to node 32 by *L2*. The path (it is only an edge) from the pseudo node *s* to node 32 is represented by *L3*. Then we have the following equations:

$$a(L3) = F(L1) + F(L2)$$

$$b(L3)) = D(L1) + D(L2)$$

The number of edges (or paths) between the pseudo node *s* and the node 32 would be equal to $(C_T/e)^2$, each edge corresponds to a combination of two paths, one path coming from s_1 and the other coming from s_2. We can reduce these $(C_T/e)^2$ paths into C_T/e by a technique very similar to the one we already used in the partitioning of chain structured programs. Let *P1* and *P2* are two edges (or two paths) between *s* and node 32. Assume that the total communication requirement of the two paths lie in between two successive permissible levels. If *a(P1)* as well as *b(P1)* is less than or equal to the respective values of *a(P2)* and *b(P2)* then the path *P2* is just redundant and there is no need to consider the assignment corresponding to this path. But if *b(P1)* is less than *b(P2)* and *a(P1)* is larger than *a(P2)* then also it is possible to reject one path as follows. Here we reject the path *P1* as the cost of assignment of *P1* would always be larger than the cost of assignment corresponding to *P2* with a guarantee that the error in the communication requirement of the selected path, i.e., *P2* would be bounded by *e.*

Each application of the procedure Merge will take time not larger than $O(MN^2 (C_T/e)^2)$ with the guarantee that the maximum difference between the total communication requirement of the partitioning we found and the actual communication requirement of the optimal partitioning is at the most equal to *Me.* If the relative error bound for the user interested to use the approximate scheme is ε then the time complexity of the algorithm would be equal to $O((C_T/\varepsilon)^2N^2M^3)$.

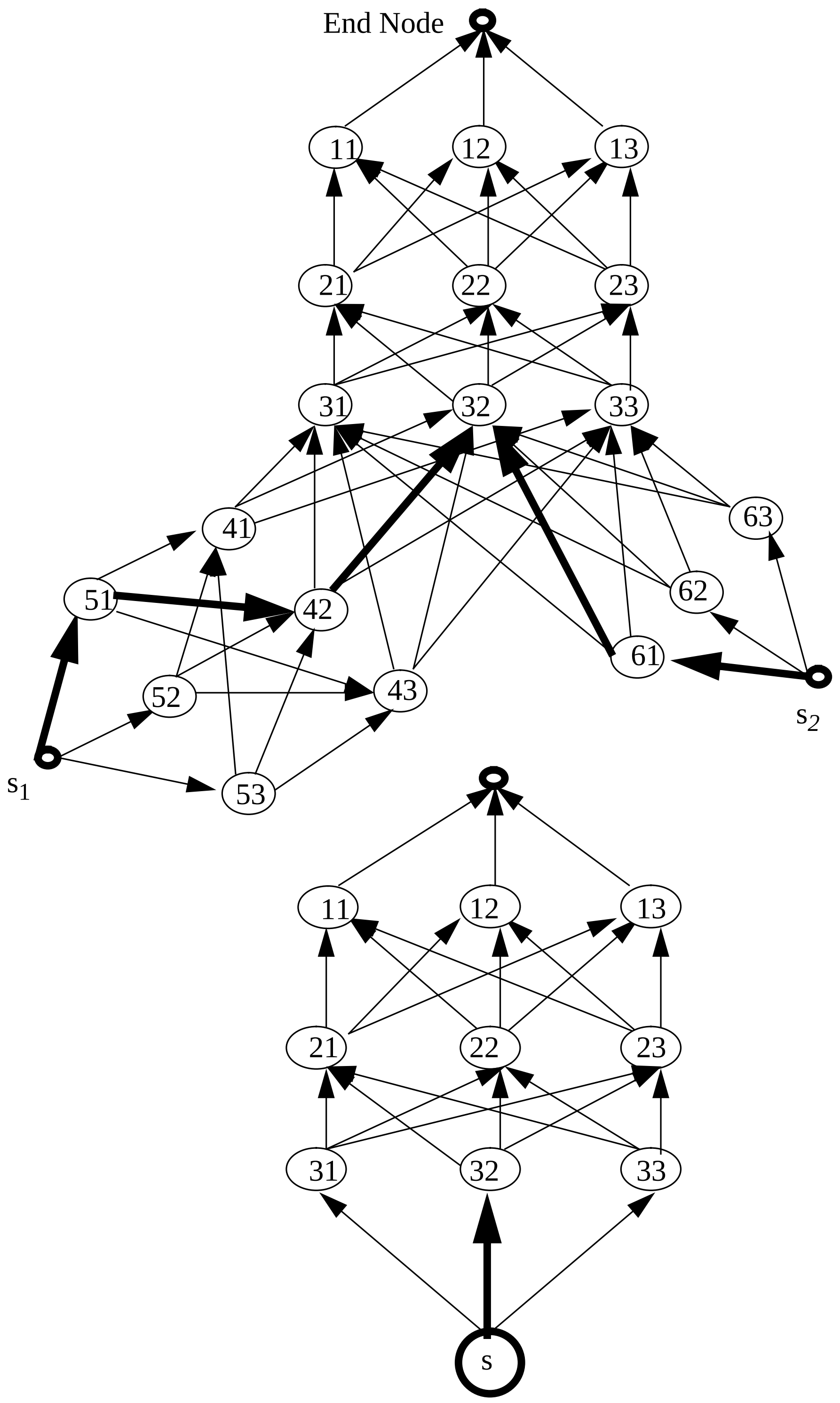

Fig. 3.5 A path from s_1 to node 32 and a path from s_2 to node 32 (top) are combined into a single path or edge from s to node 32 (bottom).

The approximate technique that we have described is thus a fully polynomial time approximation scheme in which the time complexity is a polynomial function of the size of the problem as well as the reciprocal of the total error ε.

4 Conclusions

Distributed Computing offers a solution where a network of heterogeneous computers can be used to solve large scale scientific and engineering problems. In theory, this approach can dramatically improve the performance because each component of the application is executed on an architecture that is best suited for it. In reality, however, a number of problems should be solved in order to utilize the full potential of the distributed system. The distributed computer system and the distributed application should be modeled in such a fashion that analytical research can provide solutions to the fundamental question of how to optimally partition the application across the machines in the distributed computer system.

In this paper we have presented efficient schemes which can partition chain or tree structured Distributed Multimedia Applications consisting of several heterogeneous components across the distributed machines in the network. We have provided approximate solutions to the problem of partitioning chain and tree structures taking into account only the communication constraint and minimizing the total cost. It is possible to extend these techniques to less restricted structures and currently we are working on partitioning series-parallel structures which belongs to a yet another useful class of distributed applications. We also intend to find exact or approximate solutions to the general partitioning problem under further restrictions on the application problem. It is also possible to design efficient heuristics based on this approach which can take into account the additional resource constraint on the load assigned to machines in the distributed computer system.

Acknowledgment: We acknowledge the motivation, encouragement and support provided by Asima Ashraf, Gabriel Dermler, Thomas Braunl, and Kurt Rothermel.

5 References

[1] A. Khokhar, V. K. Prasanna, M. Shabaan, and C. Wang, “Heterogeneous Supercomputing: Problems & Issues,’’ IEEE Computer, June 1993..

[2] Rothermel K., Barth I., Helbig T., “CINEMA - an architecture for configurable distributed multimedia applications,’’ Tech. Report 3/1994, Universität Stuttgart, Fakultät Informatik.

[3] Hagin A., Dermler G., Rothermel K., “Problem formulations, models and algorithms for mapping distributed multimedia applications to distributed computer systems,’’ Tech. Report 3/1996, Universität Stuttgart, Fakultät Informatik.

[4] S. H. Bokhari, “Dual processor scheduling with dynamic reassignment,” IEEE Transactions on Software Engineering,” July 1979, Pages = 341-349.

[5] S. H. Bokhari, “On the mapping problem”, IEEE Transactions on Computers, March 1981, Pages 207-214.

[6] S. H. Bokhari, “A shortest tree algorithm for optimal assignments across space and time in a distributed processor system”, IEEE Transactions on Software Engineering, November 198 1, Pages=583-589.

[7] S. H. Bokhari, “Assignment problems in parallel and distributed computing”, Kluwer Academic Publishers, 1987.

[8] M. Ashraf Iqbal,“Approximate algorithms for partitioning and assignment problems”, ICASE Report Number = “86-40”, NASA Contractor Report 178130, June 1986.

[9] M. Ashraf Iqbal, Joel H. Saltz and S. H. Bokhari, “A comparative analysis of static and dynamic load balancing strategies”, Proceedings of the 1986 International Conference on Parallel Processing, August 1986.

[10] M. Ashraf Iqbal and S.H. Bokhari, “Efficient Algorithms for a class of Partitioning Problems,’’ IEEE Trans. on Parallel & Distributed Systems, 1995.

[11] M. Ashraf Iqbal, “Approximate Algorithms for Partitioning Problems’’, International Journal of Parallel Programming, October 1991.

[12] M. Ashraf Iqbal, “Efficient Algorithms for Dilated Mapping of Binary Trees’’, Journal of Parallel & Distributed Computing (JPDC), 1992.

[13] M. Ashraf Iqbal, Saeed Iqbal, and M. E. Shabaan, “Partitioning Image Processing Tasks on Heterogeneous Computer Systems, ’’Proceedings of the Workshop on Heterogeneous Processing, April 1994.

[14] M.Ashraf Iqbal & M.E. Shabaan, “Heterogeneous Partitioning of Chain Structured Image Processing Tasks,’’Worshop on Computer Architectures for Machine Perception (CAMP’93), NewOrleans, Louisiana.

[15] Harold S. Stone, “Multiprocessor scheduling with the aid of network flow algorithms”, IEEE Transactions on Software Engineering, January 1977, Pages = 85-93.

[16] T.N. Bui & C. Jones, “Parallel Algorithms for Partitioning Simple Classes of Graphs,’’ International Conference on Parallel Processing, August 1990.

[17] R.M. MacGregor, “On Partitioning a Graph: a theoretical & empiric study’’, Ph.D Thesis, University of California, Berkeley, 1978.

[18] D. M. Nicol and D. R. O’Hallaron, “Improved Algorithms for Mapping Pipelined and Parallel Computations,’’ IEEE Trans., Computers, vol. 40, No. 3, 1990.

[19] D. F. Towsley, “Allocating programs containing branches and loops within a multiple proccesser system, ’’ IEEE Trans. on Software Engineering, Oct., 1986, pp. 272-277.