An efficient approximation algorithm for load balancing with resource migration in distributed systems

Ravi Varadarajan
Computer and Information Sciences Department
University of Florida, Gainesville, FL 32611

An efficient approximation algorithm for load balancing with resource migration in distributed systems

Ravi Varadarajan
Assistant Professor
Computer and Information Sciences Department
University of Florida, Gainesville, FL 32611
Tel. No. (904) 392 -1467
e-mail : ravi@cis.ufl.edu

Abstract

Resource migration in a distributed computer system can be performed for performance enhancement and we call this activity "load balancing with resource migration". We propose a general load balancing model with both job and resource migration and demonstrate how this model applies to the file migration problem in distributed databases. The intractability of this load balancing model suggests obtaining approximate solutions. In this paper, we propose an approximation approach that partitions the distributed system into regions so that resource migration takes place between the regions while job migration occurs within the regions. By using a good rule for partitioning the system into regions, good approximate solutions can be obtained. We present theoretical bounds and experimental results to indicate how our approximation approach can provide close to optimal solutions in many instances.

List of Symbols

d

F_J, F_R, c_J, c_R

$V, P(V), u, v, I, I'$

f, g, h, b_x, q, d_{xy}

$\delta(.), T, Q(I, g), R(I, f)$

Z^+

S_{xy}, E_{xy}

$W_1, W_2, \ldots, W_m$

$I'_1, I'_2, \ldots, I'_m$

$N_j, L_j, R_j, S'_j, E'_{jk}, Q'_j$

$z_{jk}, z_{kj}, , z_{jj}, t_{jk}$

$\lceil, \rceil, \lfloor, \rfloor, \neq, \in, \leq, \geq, \sum$

$m, n, p, O(.)$

$\Delta, \Delta', k', R'_j$

z^*_{jk}

$\mathcal{T}, l(t), D, T^*, L, U$

log

L_k, D_k, J

g_j

m^3, n^2

ϵ_1, ϵ_2

U_j

$s(x), S_j, S'_n, a_l, L, L'\ r, z'_{jk}, G_{jk}, H_{jk}, \beta_{jk}, y_{jk}, J^+$

Δ_j, x^*_{jk}

K

$\subseteq$

1 Introduction

A resource in a computer system is defined as any hardware or software entity required for the execution of a user job. Examples of resources include processors, memories, interconnection networks, system processes, data files, database relations and file servers. In a distributed computer system, some of these resources are distributed among the various nodes in the system. If the distribution of a resource among the nodes can vary with time, then we call this resource a 'migratable' resource. Examples of such resources include datafiles, processes and mobile hosts. Traditionally, the term 'load balancing' refers to the operation of distributing or redistributing the user tasks among the different nodes in a distributed system, to achieve a desirable performance level; typical performance measures include job response time, throughput and processor utilization. We extend this definition of load balancing to include the operation of distributing or redistributing the *migratable* resources of a computer system to achieve a desirable performance level. Redistribution of the user jobs and the migratable resources among the nodes in the system are respectively referred to as *job migration* and *resource migration* respectively. Job migration also includes *process migration* which involves the migration of a task while it is still being executed.

More generally, in load balancing without resource migration, the distribution of the resources over the nodes is fixed, and load balancing is accomplished by determining an assignment of the resource requests to the resources such that the cost in handling these requests will satisfy some required constraint. On the other hand, in load balancing with resource migration, load balancing is accomplished by first determining a new distribution of the resources over the nodes and then determining an assignment of the resource requests to the resources under the new distribution. Obviously, load balancing with resource migration can provide a higher opportunity for achieving a lower cost in handling the resource requests, provided that the cost of migrating the resources is affordable. In general, load balancing without resource migration would be performed more frequently than load balancing with resource migration. And the latter is performed only when the pattern of resource requests has changed dramatically and when this change is expected to last over a long period of time.

In the literature, many load balancing models have been proposed for distributed and multiprocessor systems. Casavant and Kuhl [5] give a survey of many of these models. In general, load balancing models can be classified in the following four ways : (a) static and dynamic, (b) centralized and decentralized, (c) deterministic and stochastic and (d) models for general distributed and multiprocessor systems and those for systems with specific topologies. Job allocation (e.g. [2, 28]) and resource allocation [6, 33] models belong to the static category while job and resource migration models belong to the dynamic category. In centralized load balancing models (e.g. [21, 4]) load balancing decisions are performed

at a central site. In decentralized models (e.g. [17, 18, 19, 22, 24, 26, 32]), the nodes or processors perform load balancing using distributed strategies that rely on some local information available to the nodes. Deterministic load balancing models usually involve graph-theoretic (e.g. [28]), integer programming or heuristic approaches such as those used in distributed strategies. Stochastic load balancing models are based on Markov chains and (e.g. queueing models (e.g. [10, 21, 32]), Bayesian decision models (e.g. [27]) or other probabilistic approaches (e.g. [23]). Load balancing models for distributed memory multiprocessor systems (e.g. [1, 9]) exploit the specific topologies of the interconnetion networks. The load balancing model we propose in this paper is a centralized graph-theoretic deterministic model that includes both job and resource migration.

First we give a few examples of applications where resource migration can also be used for load balancing. In a distributed database or file system, file migration is performed in order to maintain at all times, a desirable relation between the file access rates and the distribution of file copies among the nodes. In this case, we refer to the database files as migratable resources, the file migration activity as resource migration and the remote file access activity as job migration. We will discuss, in detail, the example of database file migration in a later section. Many file migration models exist in the literature (e.g. [15, 14, 30, 31]). But these models with the exception of [15] ignore the effect of balancing the file access requests evenly among the many copies of the file. A distributed heuristic is proposed in [15] to determine whether to migrate a process or a file, or whether to replicate a file and this decision is based on probablistic estimates of number of read and write accesses to files and the utilization of various resources in the system. On the other hand, the model we propose is a deterministic one and considers only one type of resource. Another example of resource migration in a distributed computer system can occur when a job in one host needs the services of a system process such as a file server, query processing program and editor process, running on a remote host. Here, instead of sending the request to the remote host (that is, migrating the job) and transferring the results back, the required process itself (which is the resource) can be migrated from the remote host.

When load balancing can be achieved with both job and resource migration, first there is a problem of determining *when* to migrate jobs or resources. This migration detection problem is **NP-Complete** (see [31]) even in the presence of complete information regarding the future resource requirements. Hence in many applications, these migration points are not determined beforehand but instead the load balancing problem is considered periodically or at time points of degradation of specific performance measures.

The other important issue in load balancing with both job and resource migration is the problem of deciding which jobs or resource units to migrate. In general, the decision of whether to migrate jobs or resources in a distributed computer system depends on the following: (i) job characteristics such as resource requirements and migration cost, (ii) re-

source characteristics such as capacity, storage requirements and migration cost and (iii) communication network characteristics such as topology and channel bandwidth. We refer to the above problem of deciding which jobs and resource units to migrate as the general load balancing problem with resource migration. In this problem, it is necessary to find a proper distribution of resources and jobs among the various nodes in the system so that the *desired* trade-off occurs between the *job migration cost* and the *resource migration cost*. In one formulation of this problem, the total migration cost (of jobs and resources) is minimized. When the resources can migrate in parallel, *bottleneck cost* criterion is more appropriate than the *total cost* criterion for resource migration cost. In our formulation of the load balancing problem, we use a bottleneck criterion.

Due to the intractability of the load balancing problems, determining the exact optimal solutions need either exhaustive search or heuristic search procedures, all of which are prohibitively expensive to be executed in real time. As a result, approximate solutions are usually needed for such problems. One approach which is common to all the existing techniques is to use heuristic rules for guiding the search to an approximate solution. However, the heuristic rules for obtaining approximate solutions are generally difficult to derive in many formulations. In this paper, we propose a new approximation approach to solve the general load balancing problem.

In the new approach, we separate the resource migration and job migration problems. We do this by partitioning the system (or network) into regions so that resource migration takes place between the regions while job migration takes place only within the regions. For the resource migration problem, we use local approximations of the network characteristics and the total resource requirements within a region. In this problem, we attempt to minimize the bottleneck resource migration cost but with a constraint on the job migration costs. This resource migration problem is shown to reduce to a well-known commodity distribution problem for which efficient polynomial-time algorithms already exist. By suitable partitioning, good approximate solutions to resource migration problem can be obtained. Once resource migration is determined, job migration problems are solved independently for each region. We also propose an efficient approximation algorithm for determining the job migrations.

The organization of the paper is as follows. In the next section, we present our general load balancing model with resource migration along with the motivation for such a model. We also explain how this model can be used to solve a load balancing problem that arises in the case of file migration in distributed databases. In Section 3, we mention the complexity of the problem and explain, in detail, the proposed approximation approach. In Section 4, we present our approximation algorithm based on this approach and analyze its time complexity. In Section 5, we present theoretical bounds for the performance of our approximation algorithm. In Section 6, we discuss the experimental results to demonstrate how our

approximation algorithm provides close to optimal solutions in many instances, especially when an appropriate partition scheme is chosen. Finally, we summarize and present our conclusions.

2 Proposed load balancing model with resource migration

In a general load balancing problem with resource migration, two types of costs need to be considered in deciding whether job or resource migration needs to be performed. One is the *job migration cost* and the other is the *resource migration cost*. Each of these costs has two components, a fixed part and a variable part. Typically, the fixed cost is associated with the overhead time involved in setting up the job or the resource in a new node or location and hence is independent of the migration distance. On the other hand, the variable cost depends on the migration distance and we assume here the use of an *unit distance migration cost*. Unit distance job migration cost is defined as a measure of the time to send the job to a remote node, unit distance away and receive the results back. Unit distance resource migration cost is a measure of the the time to migrate a resource unit to a node, unit distance away.

In our proposed load balancing model, we assume that the resources can migrate only at the time when this load balancing problem is solved. On the other hand, job migration [1] occurs frequently after resource migration but only according to the *resource access function* determined by the load balancing model; the resource access function indicates which resource copy should be accessed by a job from a given node. Our assumption is reasonable in the light of the fact that the cost of resource migration is usually much higher than the cost of job migration. The resource migration is thus necessitated by a large shift in the resource access pattern at which time the proposed load balancing problem will be solved.

In our load balancing formulation, we determine how the resources should migrate as well as the resource access function so as to obtain the desired trade-off between the resource migration cost and the job migration cost. The job migration cost not only includes the cost of communicating the resource access request to the remote node and transferring the results back but also the processing delay at the remote site. The latter component of the job migration cost is reduced by balancing the resource access requests among the nodes that contain the resource copies. Either the total cost or the bottleneck cost criterion can be used for the job migration and resource migration costs. The total cost criterion is useful when the resources and the jobs have to be migrated one at a time as for example, when a single broadcast bus such as Ethernet is used for migrating the software resources and the jobs among the nodes. The bottleneck cost criterion is useful when the resources and the

[1] Here job migration also includes activities related to remote access of resources.

jobs can be migrated in parallel. Here, we give our formulation of the general load balancing problem with the bottleneck cost criterion. For this, we introduce the following notations:

V	— set of all nodes in the system
$P(V)$	— power set of V
I	— set of nodes with resource units after migration
I'	— set of nodes with resource units before migration
$f : (I - I') \rightarrow I'$	— migration function specifying how the resource units migrate $f(x)$ is the node from which the resource unit at x has migrated
$g : V \rightarrow I$	— job assignment function specifying where a resource request from a node needs to be processed $g(x)$ is the node that satisfies the resource access request from x
$h : I \rightarrow P(V)$	— indicates for a resource unit, the set of nodes whose resource requests need to be processed by the resource (h is the inverse function of g)
b_x	— resource requirements of the jobs submitted at the node x
q	— resource request processing capacity of a node (requests per unit time)
d_{xy}	— communication distance from node x to node y
$\delta : I \rightarrow Z^+$	— delay function indicating the delay in processing a resource request at a node
F_R, F_J	— fixed costs (distance-independent) of resource and job migration respectively
c_R, c_J	— cost per unit distance for resource and job migration respectively
T	— desired maximum job response time

We assume that the resource requests from all the jobs have the same processing time, independent of the nodes. This assumption is made only to simplify the discussion. The resource requirements of each job are specified as number of resource accesses required per unit time. We assume that if resource requests from a node need to be processed remotely, then all these requests are directed to the same remote node for processing. The communication cost for a resource request remotely processed depends only on the communication distance and is in particular, independent of the network traffic. This assumption is justified in networks with sufficient communication bandwidth. The processing delay is however dependent on the processing load at a node or a host and is assumed to be directly proportional to it.

The resource migration cost is denoted by $R(I, f)$ and the job migration cost is denoted by $Q(I, g)$; f and g are respectively resource and job migration functions. These costs are given as follows:

$$
\begin{aligned}
R(I, f) &= \max_{y \in I - I'} [F_R + c_R d_{f(y)y}] \\
Q(I, g) &= \max_{\{x \in V | b_x > 0\}} [F_J + c_J d_{xg(x)} + \delta(g(x))]
\end{aligned}
$$

Here $\delta(y) = \frac{\sum_{x \in h(y)} b_x}{q}$. Note that the delay here does not represent the average delay as given in a queueing model but rather gives a bound on the processing delay assuming no particular queueing discipline. A similar delay function is employed in the processor and link assignment problem studied in [3]. Thus the load balancing problem is one of determining I, g and f so as to achieve a *desirable* combination of $R(I, f)$ and $Q(I, g)$. In our formulation, we minimize $R(I, f)$ with a constraint on $Q(I, g)$, that is, $Q(I, g) \leq T$. In many applications, we also have the restriction that $|I| \leq |I'|$. The load balancing problem we wish to solve, which we call **Problem LBP**, can thus be stated as follows:

> **Given**: A network $G = (V, E)$ with distance matrix $\{d_{xy}\}$, set I' of initial locations of resource units, fixed cost F_R and unit distance cost c_R of resource migration, fixed cost F_J and unit distance cost c_J of job migration, resource request processing capacity q and the desired job response time T.
>
> **Required**: Set I of new locations for the resource units with $|I| \leq |I'|$, the resource migration function f and the job migration function g such that $R(I, f)$ is minimized with the constraint that $Q(I, g) \leq T$.

Now we will illustrate how our proposed model can be used to address the issue of file migration in distributed databases.

File migration in distributed databases

In a distributed relational database, relations are partitioned either vertically (grouping of attributes) [20] or horizontally (collections of tuples) [7] into possibly overlapping fragments which are referred to as database files. These files are distributed among the different sites or nodes in the network. To increase the availability as well as to improve the transaction response time, it is a common practice to replicate many of these files. The degree of redundancy is a design choice that often depends on the cost of performing "updates" on all the copies of a file. A transaction submitted by the user at a node translates into file access requests which can be either queries or updates. All the file migration and allocation models that had been proposed before (e.g. [6], [31]) assume that the file access requests are always sent to the nearest node containing a file copy; they ignore the effect of balancing the requests among the different file copy nodes. Our load balancing model attempts to minimize the query response time by balancing the work load among the different file copy nodes. As the pattern of file access requests changes, it may be necessary to migrate the file copies among the different nodes. The file migration problem we propose involves determining where to move the file copies to and also determining how to balance the workload among the new file copy locations so as to minimize the query response time as well as the file migration costs. The query response time consists of the following two components: (1) query communication time which includes the time for sending the request and receiving the results back and (2) query processing delay which includes the time for processing the file access request.

In formulating the file migration problem, we make the following reasonable assumptions:

1. A constant number of file copies is maintained at all times. In most applications, when there is a constant number of file copies, the cost of updating all the file copies and the cost of storage do not differ significantly among the different allocations of file copies. Thus the determination of number of file copies becomes a design choice that also needs to consider the availability of file copies in case of node failures. With this assumption, we do not have to consider update costs in our load balancing problem.

2. Due to the first assumption, whenever a new file copy is generated at a node, some other existing copy at another node needs to be deleted. We assume that the file access requests queued at the node (from which a copy is deleted) are directed to a node containing a new file copy. Thus the cost of migrating a copy from one node to another includes the following: (i) the cost of moving the file and (ii) the cost of redirecting the query requests already queued up at the node from which the file is migrated to some other nodes containing the file copies.

3. All file copies can migrate in parallel.

4. Query communication delay is independent of the query traffic and is dependent only on the communication distance. We assume that the nodes have a limited processing capacity (expressed in file access requests per unit time) and the processing capacity is the same for all the nodes. Moreover, we assume a query processing delay component that is directly proportional to the query traffic directed to the node containing the file copy.

With these assumptions, the file migration problem becomes identical in formulation to the load balancing problem (LBP) with *file copies* taking the place of resource units, *file access request rate* taking the place of resource requirements (i.e. b_x) and *query response time* taking the place of job response time.

3 Proposed approximation approach

The load balancing problem LBP can easily be shown to be **NP-hard** by a simple reduction from the Dominating set problem [12]. One approach to obtaining an optimal solution is to use heuristic search techniques such as branch-and-bound algorithms. We implemented one such branch-and-bound algorithm. Even for reasonable size problems with 12 nodes and 4 resource units, the branch-and-bound program took more than 2 CPU hours for some problem instances. This is undesirable since the load balancing problem needs to be solved in real time in almost all the applications. As a result, we seek efficient algorithms that

provide close to optimal solutions. One possible approximation approach is to use a branch-and-bound search that terminates with an approximate solution instead of with an optimal solution [30]. But the heuristic rules for obtaining these approximate solutions are generally difficult to derive for constrained optimization problems such as the one we had proposed. We propose an approximation approach that is efficient, simple to understand and yet can provide reasonably good approximate solutions.

In the approximation approach we propose, we partition the given system into regions so that resource migration takes place only *between the regions* while job migration takes place only *within* the regions. The partitioning helps to achieve local approximations for the underlying network characteristics so as to separate job and resource migration problems. After partitioning, we first focus on resource migration only but with a view to reducing the job migration costs. This problem will be shown to reduce to a problem known as the *bottleneck transportation problem* in the Operations Research literature. From the solutions of this problem, we can determine the number of resource units that should be transferred from one region to another but not the exact new locations for the resource units. We give a simple algorithm to determine the new locations for the resource units that are consistent with the optimal solution to the transportation problem. After resource migration is determined, we still have to determine where the remote resource access requests should be processed so as to reduce the job migration costs. Since the resource access requests are processed within the regions, we solve these job migration problems independently for each region. For this job migration problem, we ignore the job communication costs and focus on balancing the resource access requests evenly among the nodes that contain the resource units after resource migration. First we describe how the resource migration problem is solved using the partitioning approach. Later we address the problem of determining the job migrations.

3.1 Approximate resource migration problem among the regions

Let the set of nodes V be partitioned into m regions $W_1, W_2, \ldots, W_m$. Later we will explain how this partitioning can be done so as to obtain *good* approximate solutions. We require that the number of regions is at most the number of resource units to guarantee that each region has at least one resource unit after migration. Let $I'_1, I'_2, \ldots, I'_m$ be the corresponding sets of nodes in the regions containing file copies before file migration; that is, $I'_j = W_j \cap I'$, for $1 \leq j \leq m$. Additionally, we define the following quantities:

N_j = number of resource units in region W_j before migration
$= |I'_j|$
L_j = number of nodes in region W_j
$= |W_j|$
S'_j = average communication distance within region W_j

$= \frac{\sum_{x,y \in W_j} d_{x,y}}{(L_j^2 - L_j)}$

E'_{jk} = average communication distance for resource migration from region W_j to region W_k

$= \frac{\sum_{x \in I'_j} \sum_{y \in (W_k - I'_k)} d_{x,y}}{N_j \times (L_k - N_k)}$ if $j \neq k$ and 0 if $j = k$

Q'_j = total resource requirements in region W_j

$= \sum_{x \in W_j} b_x$

$F_J + c_J S'_j$ is the average cost of transferring a resource access request to a remote node within W_j and receiving the results back. $F_R + c_R E'_{jk}$ is the average cost of migrating a resource unit from region W_j to region W_k. It is reasonable to assume that a node will not send as well as receive a resource unit during migration. With these local approximations, the resource migration problem between the regions is formulated as follows:

$$\begin{aligned} \text{Minimize } & \max_{\{(j,k)|z_{jk}>0\}} (F_R + c_R.E'_{jk}) \\ \text{s.t. } & F_J + c_J S'_j + \frac{Q'_j}{(\sum_{k=1}^{m} z_{kj})\, q} \leq T, \text{ for all } 1 \leq j \leq m \\ & \sum_{k=1}^{m} z_{jk} \leq N_j, \text{ for all } 1 \leq j \leq m \\ & z_{jk} \in \mathcal{N}, \text{ for all } 1 \leq j,k \leq m. \end{aligned}$$

The variable z_{jk} denotes the number of resource units that need to migrate from the region W_j to the region W_k; z_{jj} is the number of resource units that remain in their locations within region W_j. We allow the possibility of some resource units not being utilized after resource migration and hence we have the inequality in the second set of constraints. Let the quantity $R_j = \left\lceil \frac{Q'_j}{(T - F_J - c_J.S'_j)q} \right\rceil$ denote the minimum resource capacity (expressed in number of resource units) needed to meet the requirements of jobs in region W_j. Also, let $t_{jk} = (F_R + c_R d_{jk})$ denote the cost of migrating a resource unit from the region W_j to the region W_k. Then the approximate resource migration problem we wish to solve is formulated as follows:

$$\begin{aligned} \text{Minimize } & \max_{\{(j,k)|z_{jk}>0\}} t_{jk} \\ \text{s.t. } & \sum_{k=1}^{m} z_{kj} \geq R_j, \text{ for all } 1 \leq j \leq m \\ & \sum_{k=1}^{m} z_{jk} \leq N_j, \text{ for all } 1 \leq j \leq m \\ & z_{jk} \in \mathcal{N}, \text{ for all } 1 \leq j,k \leq m. \end{aligned}$$

The above formulation is known as the *bottleneck transportation problem* in the Operations Research literature. In the general bottleneck transportation problem, there are m suppliers each of which can supply units of a certain commodity to n destinations and there is a shipping cost per unit of commodity for each supplier-destination pair. There is a maximum

quantity that a supplier can provide and a minimum quantity demanded by each destination. In our formulation, the resource units are the "commodities" and the regions we had defined serve as both "suppliers" and "destinations". Note that the bottleneck transportation problem is *infeasible* if and only if $\sum_{k=1}^{m} R_k > \sum_{k=1}^{m} N_k$. We will explain in Section 3.2 how this case is addressed in our approach. In the following discussion, we will assume that the problem is feasible.

In the above formulation of our problem, we have implicitly assumed that the regions either supply resource units or receive resource units but do not do both. Since $E'_{jj} = 0$, it follows that there always exists an optimal solution such that for a region W_j, when $R_j \leq N_j$, region W_j serves only as a supply region and when $R_j \geq N_j$, region W_j serves only as a destination region. Thus we can eliminate from consideration those regions W_j for which $R_j = N_j$ and divide the remaining regions into "supply regions" and "destination regions". For a supply region W_j, the supply quantity is given by $N_j - R_j$, while for a destination region W_j, the demand quantity is given by $R_j - N_j$.

Efficient algorithms (e.g. [13], [25]) had been proposed in the literature to solve $m \times n$ bottleneck transportation problems in which there are m suppliers and n destinations. Of these, the algorithm due to Garfinkel and Rao [13] has a polynomial time complexity given by $O((m+n)^3 \log(mn))$. The special case of $m = 2$ (or $n = 2$) can be solved optimally in $O(n)$ (or $O(m)$) time using the algorithm proposed in [29].

3.2 Reducing regional resource requirements

Now we address the case when the bottleneck transportation formulation is *infeasible*, that is, when $\sum_{k=1}^{m} R_k > \sum_{k=1}^{m} N_k$. This does not necessarily mean however that the load balancing problem **LBP** is infeasible. In this case, we reduce the total resource requirement so as to make it equal to the number of resource units available. Each region has its resource requirements reduced with the restriction that there is at least one resource unit required in each region after this reduction. The latter restriction is required so as to ensure that job migration takes place only within the regions. Before we use the following procedure, we first find the set of regions W_j for which $R_j > |W_j|$, the number of nodes in the region. For each region W_j in this set, let $R_j \leftarrow |W_j|$ so as to ensure that there is at most one resource unit per node. Then we apply the following *heuristic* procedure to reduce the resource requirements further if necessary.

Procedure ReduceDemand;
(* Reduces the resource requirements for the regions so that the total requirement is equal to p, the number of resource units available. This is done so that the reduction is somewhat equally distributed among the regions and there is at least one resource unit in each region

after this reduction process. Note that $p \geq m$, the number of regions *)

1. Sort the regions in non-increasing order of their resource requirements. Let $R_1 \geq R_2 \ldots \geq R_m \geq 1$.
2. $\Delta \leftarrow \sum_{i=1}^{m} R_i - p$; $k \leftarrow m$;
3. While $(\Delta > 0)$ do steps 4-11.
    4. Let k' be the largest $1 \leq j \leq k$ such that $R_j > 1$ and $R_{j+1} = 1$.
       If there is no such j then let $k' \leftarrow k$.
    5. $\Delta' \leftarrow \min\left(\lfloor \frac{\Delta}{k'} \rfloor, R_{k'} - 1\right)$.
    6. Let $R_i \leftarrow R_i - \Delta'$ for all $1 \leq i \leq k'$.
    7. If $R_{k'} > 1$ and $\Delta \bmod k' \neq 0$ then do steps 8-9 and exit While loop.
        8. For all $(k' - \Delta \bmod k' + 1) \leq j \leq k'$, let $R_j \leftarrow R_j - 1$.
        9. $\Delta \leftarrow 0$.
    10. $\Delta \leftarrow \Delta - k'\Delta'$.
    11. $k \leftarrow k' - 1$.

end while;

End **ReduceDemand**;

We claim that the above procedure terminates with the new resource requirements $R'_1 \geq R'_2 \ldots \geq R'_m \geq 1$ such that $\sum_{i=1}^{m} R'_i = p$. The termination of the procedure follows from the fact that Δ strictly decreases during each iteration. To prove correctness, we easily prove by induction the following loop invariants : (a) $\sum_{i=1}^{m} R_i - p = \Delta$ and $\Delta \leq \sum_{i=1}^{k}(R_i - 1)$, (b) $\Delta > 0$ implies that $k' > 0$ and (c) $R_1 \geq R_2 \geq \ldots R_m \geq 1$ for all $1 \leq i \leq m$. The procedure has a time complexity of $O(m^2)$, since the first step has $O(m \log m)$ complexity and there are $O(m)$ iterations of the While loop with each iteration having $O(m)$ complexity.

With this reduction in resource requirements, the average job response time is increased to $T' \leq T + \max_{1 \leq j \leq m} \frac{Q'_j}{q}\left(\frac{1}{R'_j} - \frac{1}{R_j}\right)$. In cases when the response time increases beyond a certain threshold set by the system designer, we recommend the alternative of increasing the number of resource units if possible. Since it is not desirable for the load balancing algorithm to increase the number of resource units automatically, we will not address that issue here. After we solve the bottleneck transportation problem, we need to determine the exact nodes the resource units migrate to, in accordance with the optimal solution to the transportation problem. We discuss this issue in the next section.

3.3 Determining new locations for resource units

The optimal solution to the bottleneck transportation problem determines the number of resource units that need to migrate from one region to another. As discussed in Section 3.1, the regions either supply excess resource units or receive deficit resource units. For those regions for which resource requirements equal resource units available, we keep the resource

units in their original locations. For the remaining regions, we have to determine the new locations for the resource units such that the number of units that migrate from one region to another corresponds to the optimal solution of the bottleneck transportation problem. We use the following *greedy random assignment* procedure to determine the new locations for the resource units. In a later section, we will show how certain partitioning schemes can guarantee that any arbitrary random assignment yields a bottleneck resource migration cost that is close to the optimal bottleneck resource migration cost of the transportation problem. A word on notation. For a region W_j, I'_j and I_j respectively denote the set of initial and new locations of the resource units in region W_j.

Procedure FindNewResourceLocation;
(* Given the optimal solution $\{z^*_{jk}\}_{1 \leq j,k \leq m}$ to the bottleneck transportation problem, finds the set I of new locations of the resource units as well as the resource migration function f using a greedy random assignment *)

1. For all $1 \leq j \leq m$, let $I_j \leftarrow I'_j$.
2. For $j \leftarrow 1$ to m do
3. If $N_j > R_j$ then do the following:
    4. For each k such that $z^*_{jk} > 0$, repeat steps 5-6 z^*_{jk} times.
    5. *Randomly* select a pair (u, v) from the set $I_j \times (W_k - I_k)$.
    6. $I_j \leftarrow I_j - \{u\}$; $I_k \leftarrow I_k \cup \{v\}$; $f(v) \leftarrow u$.

End **FindNewResourceLocation**;

This procedure takes $O(p)$ time where p is the number of resource units initially available. After we determine how the resource units migrate, it only remains to determine how the resource access requests are assigned to the resource units within the regions. This problem (which we call the *job migration* problem) is solved independently for each region. It is the subject of discussion in the next section.

3.4 Determining job migrations

The problem of determining optimal job migration within each region can be posed as follows:

> **Given**: A graph $G = (V, E)$ with distance matrix $\{d_{xy}\}$, set I of locations of resource units, resource requirements $\{b_x\}_{x \in V}$ and desired response time T.
> **Required**: Job migration function $g : V \rightarrow I$ such that $Q(I, g) = \max_{\{x \in V | b_x > 0\}}[d_{xg(x)} + \delta(g(x))] \leq T$, where $\delta(y) = \sum_{\{x | g(x) = y\}} b_x$.

Note that for ease of discussion, we have absorbed the resource processing capacity q into the resource requirements and similarly the unit distance job migration cost c_J into the

inter-node distances; we have also substracted the fixed cost of job migration F_J from the desired response time. This job migration problem is **NP-hard** since a special case of this problem with equal values of d_{xy} for all $x, y \in V$, becomes identical to the multiprocessor scheduling problem that is already known to be **NP-Complete** [12]. For determining job migration within a region, we ignore the communication costs and attempt to balance the resource access requests evenly among the nodes that contain the resource units after resource migration. In other words, we attempt to determine $g : V \rightarrow I$ so as to minimize $\max_{y \in I} \sum_{\{x \in V | g(x)=y\}} b_x$. Since this problem is identical to the multiprocessor scheduling optimization problem, we use one of the approximation algorithms that had been proposed for the latter problem. The heuristics behind these approximation algorithms are quite similar to the heuristics proposed for the **bin-packing** problem [12]. The only difference between the bin-packing problem and the multiprocessor scheduling problem is that the latter can be viewed as the problem of packing a fixed number of bins as evenly as possible while the former problem is that of packing the items using a minimum number of bins of fixed capacity.

Before we discuss two of the heuristics proposed for the multiprocessor scheduling problem, we have a few words on bounds on the optimal value. Let T^* be the optimum value given by $T^* = \min_g T(g) = \max_{y \in I} \sum_{\{x \in V | g(x)=y\}} b_x$. It has been proved [8] that $L \leq T^* \leq U$ where $L = \max\left(\frac{\sum_x b_x}{|I|}, \max_x b_x\right)$ and $U = \max\left(2\frac{\sum_x b_x}{|I|}, \max_x b_x\right)$.

In the *best-fit decreasing* heuristic, a bin capacity of L is initially assumed and the items (jobs) are considered in non-increasing order of their sizes. At each step, an item is attempted to be fitted into the bin that has the smallest capacity left after the item is put into the bin; in case more than one bin has the same smallest capacity, then the tie is broken arbitrarily. When no bin can fit the item with the current capacity, the capacity of all the bins is increased by the *smallest amount* that is necessary to fit the item into a bin. The worst-case time complexity of the best-fit decreasing algorithm is given by $O(|V| \log |V| + |V||I|)$.

In the *multi-fit* algorithm proposed by Coffman, Garey and Johnson [8], binary search is employed on the interval $[L, U]$ to arrive at the smallest bin capacity for which the first-fit decreasing heuristic can fit all the items into the given bins without increasing their capacities. Though the number of iterations is bounded by $O(\log(\max_x b_x))$, in practice it is not allowed to exceed a predetermined constant k. The time complexity of the multi-fit algorithm is $O(|V| \log |V| + k|V| \log |I|)$; we assume that b_x's are integers. The multi-fit algorithm has been shown to give solutions that are no more than $\frac{6}{5}$ times the optimal value T^* and can be as much as $\frac{13}{11}$ times T^* for some instances [11]. Friesen and Langston show [11] that a modification of the multi-fit algorithm provides solutions that are no more than $\frac{72}{61}$ times T^*.

We use the *best-fit decreasing* heuristic for our job migration problem due to its simplicity and its potential for providing good approximate solutions on the average. The algorithm

with time complexity $O(|V|\log|V| + |V||I|)$ is given below.

Procedure FindJobMigration$(V, I, \{b_x\}_{x\in V}; g, \{\delta(y)\}_{y\in I})$;
(* Given the set of V of nodes along with their resource access request rates b_x's and the set I of nodes that has resource units, finds a job assignment function $g : V \to I$ using the best-fit decreasing heuristic and the corresponding processing delays $\delta(y)$'s *)
1. Sort V in non-increasing order of their request rates (b_x's). Let $b_1 \geq b_2 \geq \cdots \geq b_l$ where $l = |V|$.
2. Let the capacity $C \leftarrow \left\lceil \frac{\sum_{x\in V} b_x}{p} \right\rceil$ where $p = |I|$. Also let $L_k \leftarrow 0$ and $D_k \leftarrow \emptyset$ for all $k = 1, 2, \ldots, p$.
3. For each $i = 1, 2, \ldots, l$ do steps 4-9.
    4. Let $J \leftarrow \{1 \leq k \leq p | C - (L_k + b_i) \geq 0\}$.
    5. If $J \neq \emptyset$ then
        6. find $h \in J$ such that $C - (L_h + b_i) = \min_{k\in J} C - (L_k + b_i)$.
        7. $D_h \leftarrow D_h \cup \{v_i\}$ and $L_h \leftarrow L_h + b_i$.
        else
            8. find $1 \leq h \leq p$ such that $(L_h + b_i) - C = \min_{1\leq k\leq p}(L_k + b_i - C)$.
            9. $C \leftarrow L_h + b_i$; $D_h \leftarrow D_h \cup \{v_i\}$ and $L_h \leftarrow C$.
10. We randomly assign the nodes in I to integers in the set $\{1, 2, \ldots, p\}$. For each integer $1 \leq i \leq p$, if i is assigned to node $u \in I$, then let $g(x) \leftarrow u$ for all $x \in D_i$ and let $\delta(u) \leftarrow L_i$.
End **FindJobMigration**.

4 Approximation algorithm

In this section, we give the approximation algorithm for solving the load balancing problem based on the ideas and procedures discussed in the last section.

Procedure Approximation algorithm;
(* Given all the necessary inputs to the LBP, determines an approximate solution (I, f, g), where I is the set of new locations of the resource units, f is the resource migration function and g is the job migration function. It also computes the bottleneck resource migration cost $R(I, f)$ and the job response time $Q(I, g)$ *)

1. Partition the set of nodes V into m regions where m is at most equal to p, the number of resource units initially available. Some suitable partitioning criteria are discussed in Section 5. Let $W_1, W_2, \ldots, W_m$ be the regions.

2. Compute the following quantities, the definitions of which are given in Section 3.1 : (a) average communication distance S'_j for all $1 \leq j \leq m$, (b) average communication

distance for resource migration from region W_j to region W_k for all $1 \leq k \neq j \leq m$ and (c) total resource requirements in region W_j for all $1 \leq j \leq m$.

3. For each region W_j, determine the number of resource units required based the desired response time T as follows: $R_j = \lceil \frac{Q'_j}{(T - F_J - c_J.S'_j)q} \rceil$.

4. For each region W_j for which $R_j > |W_j|$, decrease R_j to $|W_j|$.

5. If $\sum_{j=1}^{m} R_j > p$ (the number of resource units available), then apply the procedure **ReduceDemand** given in Section 3.2.

6. Solve the bottleneck transportation problem discussed in Section 3.1.

7. Apply the procedure **FindNewResourceLocation** given in Section 3.3 to determine the set I of new locations of the resource units as well as the resource migration function f.

8. For each region W_j, let I_j be the set of locations of resource units in W_j after migration. For each $1 \leq j \leq m$, invoke procedure **FindJobMigration**$(W_j, I_j, \{\frac{b_x}{q}\}_{x \in W_j}; g_j, \{\delta(y)\}_{y \in I_j})$, where $g_j : W_j \rightarrow I_j$ is the job migration function restricted to W_j. This procedure is given in Section 3.4.

9. Determine

$$R(I, f) = \max_{y \in I - I'} [F_R + c_R.d_{f(y)y}]$$

and

$$Q(I, g) = \max_{1 \leq j \leq m} \max_{\{x \in W_j | b_x > 0\}} F_J + c_J.d_{xg(x)} + \delta(g(x)).$$

End **Approximation algorithm**.

Complexity analysis: We assume that the partition information is provided by the user and hence step 1 takes $O(m)$ time. Step 2 takes $O(n^2)$ time where $n = |V|$ is the number of nodes in the network. Steps 3 and 4 take $O(m)$ time. Step 5 takes $O(m^2)$ time. Step 6 takes $O(m^3 \log m)$ time if we employ the algorithm of [13] to solve the $m \times m$ bottleneck transportation problem. Step 7 takes $O(p)$ time where p is the number of resource units and $m \leq p \leq n$. Step 8 takes $O(n \log n + nm)$ time. Step 9 takes $O(n)$ time. Thus the worst-case time complexity of the whole algorithm is given by $O(m^3 \log m + n^2)$ with the complexity of steps 2 and 6 being dominant.

5 Performance bounds

In this section, we show how our approximation algorithm can guarantee theoretical bounds for the solutions it provides under certain conditions that are satisfied by the partition scheme. Before we present the results, we define the notion of an (ϵ_1, ϵ_2) partition. An (ϵ_1, ϵ_2) partition (with $\epsilon_1, \epsilon_2 \geq 0$) is a partition of the network into regions $W_1, W_2, \ldots, W_m$ such that the following are satisfied: (i) for each pair of regions W_j and W_k with $j \neq k$, $\frac{|d_{xy} - E'_{jk}|}{E'_{jk}} \leq \epsilon_1$ for all $x \in W_j$ and $y \in W_k$ and (ii) for each region W_j, $\frac{|d_{xy} - S'_j|}{S'_j} \leq \epsilon_2$ for all $x, y \in W_j$. Note that $S'_j = \frac{\sum_{x,y \in W_j} d_{xy}}{|W_j|(|W_j|-1)}$ whereas $E'_{jk} = \frac{\sum_{x \in I'_j} \sum_{y \in (W_k - I'_k)} d_{x,y}}{|I'_j|(|W_k| - |I'_k|)}$; I'_j is the set of initial locations of resource units in region W_j. Our first result relates the approximate resource migration problem discussed in Section 3.1 to the load balancing problem LBP defined in Section 2.

Theorem 1 *For an (ϵ_1, ϵ_2) partition scheme, let the approximate resource migration problem among the regions be formulated as a (feasible) bottleneck transportation problem with desired job response time T and let the optimal solution to that problem have E as its value. Then our approximation algorithm provides a solution (I, f, g) to the LBP such that (i) $E(1-\epsilon_1) \leq R(I,f) \leq E(1+\epsilon_1)$ and (ii) $Q(I,g) \leq \max(2, 1+\epsilon_2)T$ provided that $2\frac{Q'_j}{|W_j|} \geq \max_{x \in W_j} b_x$ for all the regions W_j.*

Proof: Suppose the bottleneck transportation problem has an optimal solution $\{z_{jk}\}$ with optimal value E. Using the **FindNewResourceLocation** procedure discussed in Section 3.3, we can determine a resource migration function $f : I - I' \rightarrow I'$ that is consistent with the solution to the transportation problem. Then

$$\begin{aligned} R(I,f) &= \max_{y \in I - I'} (F_R + c_R d_{f(y)y}) \\ &= F_R + \max_{\{(j,k)|z_{jk}>0\}} \max_{\{y \in W_k, f(y) \in I'_j\}} c_R d_{f(y)y} \\ &\geq F_R + c_R(1-\epsilon_1) \max_{\{(j,k)|z_{jk}>0\}} E'_{jk} \end{aligned}$$

The last inequality follows from the definition of an (ϵ_1, ϵ_2) partition. We can prove in a similar fashion that $R(I,f) \leq F_R + c_R(1+\epsilon_1) \max_{\{(j,k)|z_{jk}>0\}} E'_{jk}$. Since $E = F_R + c_R \max_{\{(j,k)|z_{jk}>0\}} E'_{jk}$, we thus show that $E(1-\epsilon_1) \leq R(I,f) \leq E(1+\epsilon_1)$.

Once we determine f and I, we can determine for each region W_j, the job migration function $g_j : W_j \rightarrow I_j$ using the **FindJobMigration** procedure given in Section 3.4. This procedure uses the *best-fit decreasing* heuristic that can easily be shown (using an argument identical to the one employed in [8] to show a similar bound for the multi-fit heuristic) to give a solution with the delay $\delta(y) \leq U_j$ for all $y \in I_j$. Here $U_j = \max\left(2\frac{Q'_j}{q|I_j|}, \max_{x \in W_j} \frac{b_x}{q}\right)$;

note that $Q'_j = \sum_{x \in W_j} b_x$. Thus for a region W_j and a node $x \in W_j$,

$$\begin{aligned} F_J + c_J d_{x g_j(x)} + \delta(g_j(x)) &\leq F_J + c_J(1+\epsilon_2)S'_j + \delta(g_j(x)) && (1)\\ &\leq F_J + c_J(1+\epsilon_2)S'_j + U_j && (2)\\ &\leq F_J + c_J(1+\epsilon_2)S'_j + \frac{2Q'_j}{q|I_j|} && (3)\\ &\leq \max(2, 1+\epsilon_2)\left(F_J + c_J S'_j + \frac{Q'_j}{q|I_j|}\right) && (4)\\ &\leq \max(2, 1+\epsilon_2)T && (5) \end{aligned}$$

Here equation 1 follows from the definition of an (ϵ_1, ϵ_2) partition scheme. Equation 2 is a consequence of the bound (discussed above) given by the best-fit decreasing heuristic which is used by our algorithm. Equation 3 uses the assumption that $2\frac{Q'_j}{|W_j|} \geq \max_{x \in W_j} b_x$ and the fact that $|I_j| \leq |W_j|$. Equation 5 follows from the fact that $|I_j| = \sum_{k=1}^{m} z_{kj} \geq R_j \geq \frac{Q'_j}{(T - F_J - c_J S'_j)q}$. ■

Remarks: The constant $\max(2, (1+\epsilon_2))$ used in the bound on $Q(I, g)$ can be improved to $\max\left(\frac{72}{61}, (1+\epsilon_2)\right)$ if the modified multi-fit algorithm suggested in [11] is used in the **FindJobMigration** procedure instead of the best-fit decreasing fit heuristic. But we prefer the latter due to its simplicity.

In many practical applications, the network can be naturally partitioned according to an (ϵ_1, ϵ_2) partition scheme with ϵ_1 and ϵ_2 very small and usually less than 1. Theorem 1 needs the constraint that the resource requirements at a node in a region are *no more than twice* the average resource requirements within the region. We expect this constraint to be satisfied in most instances. Next we show how a feasible solution for the LBP is related to a feasible solution for the bottleneck transportation formulation which is the basis of our approximate resource migration problem among the regions. The proof of this result makes use of the following lemma.

Lemma 1 *Consider a bin-packing problem in which each item x has a size $s(x) \in \Re$ and the items are grouped into l classes $S_1, S_2, \ldots, S_l$ along with the capacity limits for these classes given by $a_1 \leq a_2 \leq \cdots \leq a_l$. Let L be the minimum number of bins needed to pack these items with the restriction that for all $1 \leq j \leq l$, and for all items $x \in S_j$, the sum of the sizes of the items packed in the bin that contains x should not exceed a_j. Then*

$$L \geq \sum_{j=1}^{l} \left(\frac{\sum_{x \in S_j} s(x)}{a_j} \right).$$

Proof: We omit the proof which works by induction on the number of classes l. ■

Now we are ready to state our second main result regarding the performance of our algorithm.

Theorem 2 *Consider an (ϵ_1, ϵ_2) partition scheme with $0 \leq \epsilon_1, \epsilon_2 < 1$, that also satisfies the following : for each pair of regions W_j and W_k with $j \neq k$, $(1-\epsilon_2)S'_j \leq (1-\epsilon_1)E'_{jk}$. Let (I, f, g) be a feasible solution to the LBP with $R(I, f) = E$ and $Q(I, g) \leq T$. Then there exists a feasible solution to the bottleneck transportation problem with the desired response time of no more than $\frac{rT}{(1-\epsilon_2)}$ provided that the solution to the LBP satisfies the following : $|I_j| \geq \frac{2r}{(r-1)}$ for all $j = 1, 2, \ldots, m$.*

Proof: The idea behind the proof is to transform the given feasible solution for the LBP to a feasible solution for the bottleneck transportation problem formulated with possibly higher desired response time. Notice that a solution for the transportation problem only needs to specify the number of resource units to move from one region to another. The transportation problem requires that after resource migration, all the resource access requests within a region be satisfied by the resource units that exist within the same region. Since this requirement is not necessarily satisfied by the given solution for the LBP, we move the resource units among the regions so as to meet this requirement. This task forms the main ingredient of the proof.

Let (I, f, g) be the given feasible solution to the LBP that satisfies $|I_j| \geq \frac{2r}{(r-1)}$ for some r. Note that this requires that $|I_j| > 2$ for all $j = 1, 2, \ldots, m$. For all pairs of regions W_j and W_k with $j \neq k$, let z'_{jk} indicate the number of resource units that need to move from W_j to W_k according to the solution (I, f, g). Thus

$$\sum_{k \neq j} z'_{kj} - \sum_{k \neq j} z'_{jk} = |I_j| - N_j \tag{6}$$

for all $j = 1, 2, \ldots, m$; here N_j is the number of resource units that are initially (before migration) in the region W_j. For all $1 \leq j, k \leq m$, we define G_{jk} to be the set of nodes in region W_j that access the resource units in region W_k if resource and job migration take place according to the solution (I, f, g). In other words, $G_{jk} = \{x \in W_j | g(x) \in W_k\}$. Also let $H_{jk} = \sum_{x \in G_{jk}} b_x$. For a region W_j, we also define the following quantities:

$$\beta_{jk} = \frac{1}{\left(rT - F_J - c_J(1-\epsilon_1)E'_{jk}\right) q}, \text{ for } k \neq j \text{ and} \tag{7}$$

$$\beta_{jj} = \frac{1}{\left(rT - F_J - c_J(1-\epsilon_2)S'_j\right) q} \tag{8}$$

β_{jk} indicates the *minimum* processing capacity required for a resource access request from a node in region W_j to a node in region W_k if the desired response time is no more than rT. Since $(1-\epsilon_2)S'_j \leq (1-\epsilon_1)E'_{jk}$, it is easy to see that $\beta_{jj} \leq \beta_{jk}$.

For a pair of regions W_j and W_k with $j \neq k$, let

$$y_{jk} = z'_{jk} + \beta_{kj} H_{kj} \tag{9}$$

The second term in the above equation indicates the resource capacity in W_j that is used by the nodes in region W_k if resource and job migration occur according to (I, f, g). Let Δ_j indicate the excess resource capacity available in region W_j given by

$$\Delta_j = \sum_{k \neq j} y_{jk} - \sum_{k \neq j} y_{kj} \tag{10}$$

Let $J^+ = \{j | \Delta_j > 0\}$. Then it is clear that

$$\begin{aligned}
\sum_{j \in J^+} \lceil \Delta_j \rceil - \sum_{j \notin J^+} \lfloor -\Delta_j \rfloor \quad &\geq \quad \sum_{j \in J^+} \left(\sum_{k \neq j} y_{jk} - \sum_{k \neq j} y_{kj} \right) \\
&\quad - \sum_{j \notin J^+} \left(- \sum_{k \neq j} y_{jk} + \sum_{k \neq j} y_{kj} \right) \text{ (by equation 10)} \qquad (11) \\
&= \quad \sum_{j=1}^{m} \sum_{k \neq j} y_{jk} - \sum_{j=1}^{m} \sum_{k \neq j} y_{kj} \qquad (12) \\
&= \quad 0 \qquad (13)
\end{aligned}$$

Hence the following transportation problem has a feasible solution.

$$\begin{aligned}
\text{Minimize} \quad & \max_{\{(j,k) | x_{jk} > 0\}} (F_R + c_R E'_{jk}) \\
\text{s.t.} \quad \sum_{k \notin J^+} x_{jk} \quad &\leq \quad \lceil \Delta_j \rceil, \text{ for all } j \in J^+ \qquad (14) \\
\sum_{j \in J^+} x_{jk} \quad &\geq \quad \lfloor -\Delta_k \rfloor, \text{ for all } k \notin J^+ \qquad (15)
\end{aligned}$$

Let $\{x^*_{jk}\}_{j \in J^+, k \notin J^+}$ be an optimal solution to this transportation problem. We claim that the following solution is feasible for the transportation problem defined in Section 3.1 for a desirable response time of $\frac{rT}{(1-\epsilon_2)}$.

$$z_{jk} = \begin{cases} x^*_{jk} & \text{if } j \in J^+ \text{ and } k \notin J^+ \\ R_j & \text{if } j = k \text{ and } j \in J^+ \\ N_j & \text{if } j = k \text{ and } j \notin J^+ \\ 0 & \text{otherwise} \end{cases}$$

Note that for all $j = 1, 2, \ldots, m$, $R_j = \left\lceil \frac{Q'_j}{(\frac{kT}{(1-\epsilon_2)} - F_J - c_J S'_j) q} \right\rceil$. Thus it is only required to show the following:

$$\begin{aligned}
N_j - \sum_{k \notin J^+} x^*_{jk} \quad &\geq \quad R_j, \text{ for all } j \in j^+ \\
N_j + \sum_{k \in J^+} x^*_{kj} \quad &\geq \quad R_j, \text{ for all } j \notin J^+
\end{aligned}$$

By equations 14 and 15, the above inequalities are implied by the following equations:

$$\begin{aligned} N_j - \lceil \Delta_j \rceil &\geq R_j, \text{ for all } j \in J^+ \\ N_j + \lfloor -\Delta_j \rfloor &\geq R_j, \text{ for all } j \notin J^+ \end{aligned}$$

Thus the rest of the proof is designed to show that for all $j = 1, 2, \ldots, m$, $N_j - \Delta_j - 1 \geq R_j$.

$$\begin{aligned} N_j - 1 - \Delta_j &= N_j - 1 - \sum_{k \neq j} y_{jk} + \sum_{k \neq j} y_{kj} \text{ (by equation 10)} && (16) \\ &= N_j - 1 - \sum_{k \neq j} \left(z'_{jk} + \beta_{kj} H_{kj} \right) + \sum_{k \neq j} \left(z'_{kj} + \beta_{jk} H_{jk} \right) \text{ (by eqn. 9)} && (17) \\ &= \left(N_j - \sum_{k \neq j} z'_{jk} + \sum_{k \neq j} z'_{kj} \right) - 1 + \sum_{k \neq j} \beta_{jk} H_{jk} - \sum_{k \neq j} \beta_{kj} H_{kj} && (18) \\ &= |I_j| - 1 + \sum_{k \neq j} \beta_{jk} H_{jk} - \sum_{k \neq j} \beta_{kj} H_{kj} \text{ (by eqn. 6)} && (19) \end{aligned}$$

Since the given solution to the LBP is feasible, $F_J + c_J d_{xg(x)} + \delta(g(x)) \leq T$ for all nodes x. For a node $x \in G_{kj}$ with $k \neq j$, $d_{xg(x)} \geq (1 - \epsilon_1) E'_{kj}$ while for a node $x \in G_{jj}$, $d_{xg(x)} \geq (1 - \epsilon_2) S'_j$; these follow from the definition of the (ϵ_1, ϵ_2) partition scheme. Thus for a node $x \in G_{kj}$, the following is true:

$$\delta(g(x)) = \frac{\sum_{\{y | g(y) = g(x)\}} b_y}{q} \leq \begin{cases} T - F_J - c_J (1 - \epsilon_1) E'_{kj} & \text{if } k \neq j \\ T - F_J - c_J (1 - \epsilon_2) S'_j & \text{if } k = j \end{cases}$$

Now we apply Lemma 1 to each region W_j. Here the resource units in the set $|I_j|$ are the bins and the items are the resource access requests. There are m classes of items G_{kj} where $k = 1, 2, \ldots, m$ with the capacity of class G_{kj} defined by $\left(T - F_J - c_J (1 - \epsilon_1) E'_{kj} \right) q$ when $k \neq j$ and $\left(T - F_J - c_J (1 - \epsilon_2) S'_j \right) q$ when $k = j$. Then by the result of Lemma 1, we conclude that

$$\begin{aligned} |I_j| &\geq \sum_{k \neq j} \left(\frac{H_{kj}}{\left(T - F_J - c_J (1 - \epsilon_1) E'_{kj} \right) q} \right) + \frac{H_{jj}}{\left(T - F_J - c_J (1 - \epsilon_2) S'_j \right) q} && (20) \\ \frac{|I_j|}{r} &\geq \sum_{k \neq j} \left(\frac{H_{kj}}{\left(rT - F_J - c_J (1 - \epsilon_1) E'_{kj} \right) q} \right) + \frac{H_{jj}}{\left(rT - F_J - c_J (1 - \epsilon_2) S'_j \right) q} && (21) \\ \frac{|I_j|}{r} &\geq \sum_{k=1}^{m} \beta_{kj} H_{kj} && (22) \end{aligned}$$

The last but one inequality requires that $\epsilon_1, \epsilon_2 < 1$ and the last inequality follows from equations 7 and 8. But for all $j = 1, 2, \ldots, m$, $|I_j| \geq \frac{2r}{(r-1)}$ implies that

$$|I_j| - 2 \geq \frac{|I_j|}{r} \geq \sum_{k=1}^{m} \beta_{kj} H_{kj} \tag{23}$$

Hence from equations 19 and 23, we can show that

$$
\begin{aligned}
N_j - 1 - \Delta_j &\geq 1 + \sum_{k=1}^{m} \beta_{kj} H_{kj} - \sum_{k \neq j} \beta_{kj} H_{kj} + \sum_{k \neq j} \beta_{jk} H_{jk} && (24)\\
&= 1 + \beta_{jj} H_{jj} + \sum_{k \neq j} \beta_{jk} H_{jk} && (25)\\
&\geq 1 + \beta_{jj} \sum_{k=1}^{m} H_{jk}, \text{ since } \beta_{jj} \leq \beta_{jk} && (26)\\
&= 1 + \frac{Q'_j}{(rT - F_J - c_J(1-\epsilon_2)S'_j)q} && (27)\\
&\geq 1 + \frac{Q'_j}{\left(\frac{rT}{(1-\epsilon_2)} - F_J - c_J S'_j\right) q}, \text{ since } \epsilon_2 < 1 && (28)\\
&\geq \left\lceil \frac{Q'_j}{\left(\frac{rT}{(1-\epsilon_2)} - F_J - c_J S'_j\right) q} \right\rceil && (29)\\
&\geq R_j && (30)
\end{aligned}
$$

■

Remarks: In Theorem 2, tighter bounds are obtained by decreasing the value of r but r must be strictly greater than 1. Smaller the value of r, higher should be the number of resoure units in each region in an optimal solution to the LBP. The feasible solution to the bottleneck transportation problem derived in the proof of Theorem 2 has its bottleneck cost given by the optimal value of the transportation problem formulated through equations 14 and 15. Though a good bound for this value cannot be estimated theoretically, we expect that it will be between $E(1-\epsilon_1)$ and $E(1+\epsilon_1)$ in most cases, where E is the bottleneck cost of the given feasible solution to the LBP.

Theorems 1 and 2 guarantee only loose accuracy bounds for our approximation algorithm but we can obtain much better solutions in practice. This has been confirmed by our experiments which will be discussed in the next section.

6 Experimental Results

We conducted experiments to evaluate the performance of our approximation algorithm and in particular, the effectiveness of the partitioning criteria suggested in Section 5. Since we need to compare the approximate solutions obtained by our algorithm to the optimal solutions, we implemented an algorithm based on *branch-and-bound* search to find the optimal solutions to the LBP. In this section, first we will explain how the branch-and-bound search was employed in the algorithm to find optimal solutions. Then we will discuss how the ex-

periments were performed to evaluate our approximation algorithm and then interpret the results obtained through these experiments.

Since the LBP is a constrained optimization problem, we require a two-level branch-and-bound search, the first level for determining resource unit migrations and the second level for determining job migrations. Due to memory limitations, we used the depth-first order for exploring the nodes in the branch-and-bound tree at both levels. Each node in the tree at the first level corresponds to a partial resource migration function while each node in the tree at the second level corresponds to a partial job migration function. At the first level, a node branches into two children nodes that correspond to the two choices of including and excluding the migration of a resource unit between two nodes in the network. network that does not have a resource unit. At the second level, branching of a node is based on the two choices of including and excluding the assignment of resource access requests from a node to a node that has a resource unit. A complete solution is one in which every resource unit is assigned to a new location. Whenever we reach a complete solution node at the first level, we conduct the branch-and-bound at the second level to determine a *feasible assignment* of resource access requests from the nodes in the network to the new resource locations. In a feasible assignment, the response time constraint is satisfied for every resource access request. We terminate the search at the second level, whenever we find a feasible assignment. In this case, the node at the first level where the second level search originated is said to be a *feasible solution node.*

Pruning of the nodes in the branch-and-bound tree at both levels was based on *bounding functions* that were used to estimate the lower bounds for the partial solution nodes at the two levels. The lower bound functions are based on the *bottleneck cost* of partial solution and the costs of remaining candidate edges for completing the solution. In our experiments, we observed that approximately 50% of the nodes were pruned on the average at both levels of the branch-and-bound tree. This indicates that the pruning power of our bounding heuristics is good in spite of their simplicity.

For our experiments, we selected 3 networks containing 8, 12 and 15 nodes respectively. In order to make the experiments conform with real life applications, we chose U. S. cities as nodes and distances between them (in units of 100 miles) were used as the basis for internode distances. The corresponding distance matrices are given in Tables 1, 2 and 3. Since the branch-and-bound algorithm required 2 CPU hours to solve some instances of the 15 node example, we did not include larger networks in our experimental evaluation. For each of these networks, we evaluated our approximation algorithm under four different partitions for various resource access rates. The intra- and inter-region metrics for these four partitions are given in Tables 4, 5 and 6. Though the relevant characteristic of a partition is difficult to be expressed by a single metric, we conjecture that the ratio of the *average intra-region distance* to the *average inter-region distance* is a good measure that can be employed for this

purpose. We also determine for each partition, the smallest values of ϵ_1 and ϵ_2 for which it is an (ϵ_1, ϵ_2) partition scheme. When the access rates are large and the resource units are scarce, another relevant characteristic of a partition is how evenly the nodes are distributed among the different regions. We chose the four partitions for each example so as to have variation in terms of these characteristics. For example, in the one extreme, we clustered the nodes according to their geographic proximities while in the other extreme, we clustered the nodes without regard to their proximities.

In each experiment of 100 trials, we used the following input data: (1) distance matrix of the given network, (2) number of resource units, (3) average resource access request rate with allowed deviations and (4) partition of network into regions. The number of resource units was chosen so as to be approximately one third of the number of nodes in the network. For simplicity, we let $F_R = F_J = 0$, $c_R = c_J = 1$ and $q = 1$. In each of these trials, the initial locations of the resource units were *randomly* chosen and the resource access request rates for the nodes were *randomly generated* based on an uniform distribution with the given average access rate and the allowed deviations from the average. For each such instance, an optimal solution to the LBP was determined using the branch-and-bound approach. The approximation algorithm of Section 4 was then employed for the given partition to determine an approximate solution to the LBP. The bottleneck transportation problem was solved using the algorithm of [25]. Though the branch-and-bound algorithm took as many as 2 CPU hours to solve some instances, the approximation algorithm gave solutions almost instantaneously. We then compared the approximate solution to the optimal solution with respect to (i) bottleneck resource migration cost and (ii) job response time. For the branch-and-bound (optimal) solution, we let the job response time be equal to the desired job response time T. In order to allow consistent comparisons among the results obtained under different partitions, we generated an identical sequence of access rates and initial resource locations for all the partitions of a given network.

The experimental results are given in Tables 7, 8 and 9. We let the number of resource units be respectively equal to $3, 4$ and 4 for the 8-node, 12-node and 15-node examples. In these tables, column 1 indicates three different (uniform) distributions $10 \pm 2, 10 \pm 6$ and 20 ± 4 used in our experiments. Column 2 indicates the desired response time T. Column 4 indicates the average increase in response time of our approximate solutions with respect to the desired response time. In all the 100 trials, the LBP has a solution with less than or equal to this response time. Column 6 gives the number of trials in which the approximate solution has a response time less than or equal to T and column 7 gives the average decrement in response time for these trials. Column 5 indicates the average increase in bottleneck resource migration cost of the approximate solution with respect to the optimal solution. We make the following observations from these results:

1. As far as response time is concerned, there is a strong correlation, as expected, between

Table 1: Distance matrix — 8 node example

$$\begin{pmatrix} 0 & 13 & 25 & 19 & 23 & 30 & 23 & 30 \\ 13 & 0 & 12 & 11 & 11 & 18 & 16 & 23 \\ 25 & 12 & 0 & 15 & 11 & 10 & 18 & 21 \\ 19 & 11 & 15 & 0 & 5 & 13 & 5 & 12 \\ 23 & 11 & 11 & 5 & 0 & 8 & 7 & 11 \\ 30 & 18 & 10 & 13 & 8 & 0 & 13 & 13 \\ 23 & 16 & 18 & 5 & 7 & 13 & 0 & 7 \\ 30 & 23 & 21 & 12 & 11 & 13 & 7 & 0 \end{pmatrix}$$

Table 2: Distance matrix — 12 node example

$$\begin{pmatrix} 0 & 4 & 2 & 8 & 11 & 15 & 15 & 19 & 22 & 23 & 25 & 28 \\ 4 & 0 & 5 & 6 & 11 & 12 & 14 & 16 & 20 & 23 & 24 & 27 \\ 2 & 5 & 0 & 7 & 10 & 14 & 14 & 17 & 21 & 21 & 24 & 26 \\ 8 & 6 & 7 & 0 & 5 & 7 & 8 & 11 & 14 & 17 & 17 & 21 \\ 11 & 11 & 10 & 5 & 0 & 6 & 4 & 9 & 11 & 13 & 14 & 17 \\ 15 & 12 & 14 & 7 & 6 & 0 & 4 & 4 & 8 & 14 & 11 & 17 \\ 15 & 14 & 14 & 8 & 4 & 4 & 0 & 6 & 7 & 10 & 10 & 14 \\ 19 & 16 & 17 & 11 & 9 & 4 & 6 & 0 & 5 & 13 & 8 & 14 \\ 22 & 20 & 21 & 14 & 11 & 8 & 7 & 5 & 0 & 9 & 3 & 9 \\ 23 & 23 & 21 & 17 & 13 & 14 & 10 & 13 & 9 & 0 & 9 & 5 \\ 25 & 24 & 24 & 17 & 14 & 11 & 10 & 8 & 3 & 9 & 0 & 8 \\ 28 & 27 & 26 & 21 & 17 & 17 & 14 & 14 & 9 & 5 & 8 & 0 \end{pmatrix}$$

Table 3: Distance matrix — 15 node example

$$\begin{pmatrix}
0 & 14 & 23 & 13 & 16 & 7 & 5 & 16 & 8 & 21 & 12 & 12 & 22 & 10 & 6 \\
14 & 0 & 11 & 7 & 8 & 8 & 14 & 7 & 23 & 7 & 11 & 5 & 9 & 6 & 19 \\
23 & 11 & 0 & 10 & 7 & 18 & 20 & 7 & 30 & 15 & 14 & 16 & 2 & 12 & 24 \\
13 & 7 & 10 & 0 & 4 & 9 & 10 & 3 & 21 & 12 & 4 & 9 & 8 & 3 & 14 \\
16 & 8 & 7 & 4 & 0 & 12 & 14 & 2 & 24 & 13 & 8 & 11 & 5 & 6 & 18 \\
7 & 8 & 18 & 9 & 12 & 0 & 8 & 12 & 15 & 13 & 10 & 5 & 16 & 6 & 13 \\
5 & 14 & 20 & 10 & 14 & 8 & 0 & 13 & 11 & 21 & 9 & 13 & 18 & 9 & 5 \\
16 & 7 & 7 & 3 & 2 & 12 & 13 & 0 & 23 & 14 & 7 & 11 & 6 & 5 & 17 \\
8 & 23 & 30 & 21 & 24 & 15 & 11 & 23 & 0 & 28 & 20 & 20 & 28 & 18 & 7 \\
21 & 7 & 15 & 12 & 13 & 13 & 21 & 14 & 28 & 0 & 18 & 9 & 13 & 12 & 26 \\
12 & 11 & 14 & 4 & 8 & 10 & 9 & 7 & 20 & 18 & 0 & 13 & 12 & 6 & 13 \\
12 & 5 & 16 & 9 & 11 & 5 & 13 & 11 & 20 & 9 & 13 & 0 & 14 & 7 & 19 \\
22 & 9 & 2 & 8 & 5 & 16 & 18 & 6 & 28 & 13 & 12 & 14 & 0 & 10 & 22 \\
10 & 6 & 12 & 3 & 6 & 6 & 9 & 5 & 18 & 12 & 6 & 7 & 10 & 0 & 14 \\
6 & 19 & 24 & 14 & 18 & 13 & 5 & 17 & 7 & 26 & 13 & 19 & 22 & 14 & 0
\end{pmatrix}$$

Table 4: Partitions for 8 node example

Partition	Avg. intra-region distance (1)	Avg. inter-region distance (2)	Ratio (1)/(2)	ϵ_1	ϵ_2
$\{1,6,8\},\{3,5,7\},$ $\{2,4\}$—(a)	17.14	14.43	1.19	0.60	0.50
$\{1,2,3\},\{4,5,6\},$ $\{7,8\}$—(b)	11.86	16.19	0.73	0.82	0.50
$\{1,2\},\{3,4,5,6\},$ $\{7,8\}$—(c)	10.25	17.05	0.60	0.68	0.52
$\{1,2\},\{3,5,6\},$ $\{4,7,8\}$—(d)	9.43	17.00	0.53	0.63	0.50

Table 5: Partitions for 12 node example

Partition	Avg. intra-region distance (1)	Avg. inter-region distance (2)	Ratio (1)/(2)	ϵ_1	ϵ_2
$\{1,6,11,12\},\{2,5,8,10\},$ $\{3,4,7,9\}$—(a)	14.44	12.21	1.15	1.05	0.77
$\{1,2,3\},\{4,5,6,7\},$ $\{8,9,10,11,12\}$—(b)	6.74	15.28	0.44	0.71	0.69
$\{1,2,3\},\{4,5,6,7,8\},$ $\{9,10,11,12\}$—(c)	6.21	15.49	0.40	0.65	0.72
$\{1,2,3,4\},\{5,6,7,8\},$ $\{9,10,11,12\}$—(d)	6.00	15.38	0.39	0.60	0.64

the ratio of average intra-region distance to the average inter-region distance for a partition and its performance. Smaller this ratio, better its performance. Thus for the 8 node example, partition (d) is the best while for the 12 node and 15 node examples, all the partitions (b), (c) and (d) give better results than partition (a).

2. When the access rates are very high and the resource units are scarce (close to number of regions), in addition to the above measure, the evenness of the number of nodes among the regions also plays a major role. This is because when the number of nodes is not evenly distributed among the regions, there is a large variation in the resource requirements among the regions that cannot be overcome by resource migration among the regions. Note that in the 12 node example, since one of the regions must have 2 resource units, partition (b) is the best in this respect having a node distribution closer to 6-3-3. Thus for the 12 node example, partition (b) is the best while for the 15 node example, partition (c) is the best. For the same reason as above, partition (c) for the 8 node example, partition (d) for the 12 node example and partitions (b) and (d) for the 15 node example give good results for small access rates but poor results for large access rates.

3. It is encouraging to note that under reasonable response time requirements, the approximate solutions for the best partition give close to optimal response time for all the access rates.

4. For the partitions with small ratio of intra-region to inter-region distance, the resource migration cost increases due to the increase in average inter-region distance. Thus there is a trade-off between the intra-region and inter-region distances that should reflect the trade-off between job migration and resource migration. Since resource migration occurs less frequently than job migration, smaller ratio of intra-region to inter-region distances is always preferable.

5. In our experiments, the initial locations of the resource units are randomly generated for each trial. This tends to give rise to resource migration between the regions in our approximate solution at every trial. In practical applications, resource units tend to remain within the same region for small fluctuations in the resource access rates and hence resource migration cost will be incurred very rarely.

In summary, we recommend a partition scheme with the smallest ratio of average intra-region to average inter-region distance and in case when the resource units are almost equal to the number of regions, equal number of nodes per partition is desirable.

Table 6: Partitions for 15 node example

Partition	Avg. intra-region distance (1)	Avg. inter-region distance (2)	Ratio (1)/(2)	ϵ_1	ϵ_2
$\{9,10,11,13\}, \{4,6,7,4\}, \{1,2,3,5\}, \{8,12,15\}$—(a)	13.81	12.19	1.13	1.14	0.75
$\{1,6,7,9,15\}, \{2,10,12,14\}, \{3,13\}, \{4,5,8,11\}$—(b)	7.00	14.06	0.50	0.87	0.76
$\{1,7,9,15\}, \{2,6,10,12\}, \{3,5,8,13\}, \{4,11,14\}$—(c)	6.24	14.08	0.44	0.80	0.66
$\{1,7,9,15\}, \{2,6,10,12\}, \{3,13\}, \{4,5,8,11,14\}$—(d)	6.04	14.33	0.42	0.73	0.67

Table 7: Results for 8 node example

Resource access rate	Response time T	Partition	$\Delta_{\text{resp. time}}$	$\Delta_{\text{mig. cost}}$	$\text{Freq}_{\leq T}$	avg. dec.
10 ± 2	46	(a)	13.98	9.74	0	0
		(b)	8.14	11.43	36	2.25
		(c)	7.46	13.51	0	0
		(d)	0.04	13.01	98	4.41
10 ± 6	50	(a)	10.43	9.8	3	2.0
		(b)	6.12	11.49	45	4.33
		(c)	7.5	13.57	35	4.31
		(d)	2.35	13.07	88	7.27
20 ± 4	73	(a)	16.7	8.66	0	0
		(b)	9.6	10.41	14	1.64
		(c)	20.17	12.53	0	0
		(d)	2.65	12.08	61	2.67

Table 8: Results for 12 node example

Resource access rate	Response time T	Partition	$\Delta_{\text{resp. time}}$	$\Delta_{\text{mig. cost}}$	$\text{Freq}_{\leq T}$	avg. dec.
10 ± 2	49	(a)	12.26	10.86	0	0
		(b)	2.05	11.37	83	2.27
		(c)	2.22	11.94	57	1.77
		(d)	1.77	8.55	62	0.94
10 ± 6	52	(a)	11.69	10.45	7	2.71
		(b)	4.05	11.44	70	5.24
		(c)	2.83	11.33	70	4.84
		(d)	2.5	9.1	73	3.4
20 ± 4	78	(a)	24.47	10.76	0	0
		(b)	9.64	10.56	3	0
		(c)	11.08	11.83	1	2.0
		(d)	12.85	13.54	0	0

Table 9: Results for 15 node example

Resource access rate	Response time T	Partition	$\Delta_{\text{resp. time}}$	$\Delta_{\text{mig. cost}}$	$\text{Freq}_{\leq T}$	avg. dec.
10 ± 2	58	(a)	8.48	10.86	1	0
		(b)	5.87	13.41	18	1.28
		(c)	0.07	13.24	99	5.79
		(d)	1.61	13.67	71	2.14
10 ± 6	62	(a)	8.32	10.86	19	3.11
		(b)	7.43	13.41	48	4.62
		(c)	2.12	13.24	89	7.76
		(d)	3.95	13.67	65	6.12
20 ± 4	97	(a)	10.19	10.86	2	1
		(b)	15.52	13.41	1	0
		(c)	2.37	13.24	82	4.55
		(d)	10.36	13.67	4	1.75

7 Conclusions

In this paper, we have proposed a load balancing model with resource migration in distributed systems that can be employed in many practical applications such as file migration in distributed databases. To tackle the intractability of the model, we have proposed a novel approximation approach based on partitioning the network into regions. This partitioning approach has been shown to have a very low time complexity and yet has been demonstrated through experiments to provide reasonably close to optimal solutions. In contrast, the branch-and-bound algorithm that determines an optimal solution cannot be used to solve the load balancing problems in real time.

The performance of our approximation algorithm very much depends on how the partitioning is done. Our experiments demonstrate that the lower the ratio of the average intra-region distance to the average inter-region distance, better the performance of the algorithm in terms of job response time. Most distributed systems based on wide-area networks can naturally be partitioned into geographical regions. For this reason, the partitioning task is left to the system designer to specify. But if the partitioning needs to be performed by our load balancing algorithm, we can make use of clustering algorithms proposed in the literature (see [16] for a good review of clustering algorithms). In particular, the k-means clustering algorithms attempt to find a partition of k regions so as to minimize the sum of the squares of the deviations of the intra-region distances from the regional averages. Usually these algorithms produce suboptimal solutions. Though the clustering or partitioning problem is **NP-hard** in general [12], it needs to be solved only once at the time of system design and it can remain the same so long as the network remains the same.

It is interesting to explore local search techniques that give good approximate solutions on the average and can efficiently be parallelized. One such approach we are currently investigating is to map the problem directly onto Boltzmann machine models which are parallel networks with simple computing elements. This is the subject of our ongoing and future research.

Acknowledgement : We thank Injae Hwang at the University of Florida for his invaluable assistance in performing the tedious experiments required for this project. We also thank Dr. Eva Ma for providing some insight at the early stages of this project.

References

[1] K. Baumgartner and B. Wah, "GAMMON: A Load Balancing Strategy for Local Computer System with Multiaccess Networks," *IEEE Trans. Computers*, Vol. 38, no. 8, Aug. 1989, pp. 1098-1109.

[2] S. H. Bokhari, "Dual Processor Scheduling with Dynamic Reassignment," *IEEE Trans. Software Engineering*, Vol. 5, no. 4, July 1979, pp. 341-349.

[3] S. W. Bollinger and S. F. Midkiff, "Processor and Link Assignment in Multicomputers Using Simulated Annealing," *Proc. Intl. Conf. on Parallel Processing*, May 1988, Vol.I, pp. 1-7.

[4] F. Bonomi and A. Kumar, "Adaptive Optimal Load Balancing in a Nonhomogeneous Multiserver System with a Central Job Scheduler," *IEEE Trans. Computers*, Vol. 39, no. 10, Oct. 1990, pp. 1232-1250.

[5] T. L. Casavant and J. G. Kuhl, "A Taxonomy of Scheduling in General-purpose Distributed Computing Systems," *IEEE Trans. Software Engineering*, Vol. 14, no. 2, Feb. 1988, pp. 141-154.

[6] R. G. Casey, "Allocation of Copies of a File in an Information Network," *SJCC*, 1972, pp. 617-625.

[7] S. Ceri, M. Negri and G. Pelagatti, "Horizontal Data Partitioning in Database Design," *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 1982.

[8] E. G. Coffman, M. R. Garey and D. S. Johnson, "An Application of Bin-packing to Multiprocessor Scheduling," *SIAM Journal on Computing*, Vol.7, no. 1, Feb. 1978, pp. 1-17.

[9] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multiprocessors," *J. Parallel and Distributed Computing*, Vol.7, no.2, Oct. 1989, pp. 279-301.

[10] D. L. Eager, E. D. Lazowska and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," *IEEE Trans. Software Engineering*, Vol. 12, no. 5, May 1986, pp. 662-675.

[11] D. K. Friesen and M. A. Langston, "Evaluation of a MULTIFIT-Based Scheduling Algorithm," *Journal of Algorithms*, 7, 1986, pp. 35-59.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability — A Guide to the Theory of NP-Completeness*, W. H. Freeman Company, New York, 1979.

[13] R. S. Garfinkel and M. R. Rao, "The bottleneck Transportation Problem," *Nav. Res. Log. Quart. 18*, 1971, pp. 465-472.

[14] B. Gavish and O. R. Liu-Sheng, "Dynamic File Migration in Distributed Computer Systems," *Communications of the ACM*, Vol.33, no. 2, Feb. 1990, pp. 177-189.

[15] A. Hać, "A Distributed Algorithm for Performance Improvement Through File Replication, File Migration, and Process Migration" *IEEE Trans. Software Engineering*, Vol. 15, no. 11, Nov. 1989, pp. 1459-1470.

[16] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall Advanced Reference Series, New Jersey, 1988.

[17] F. C. H. Lin and R. M. Keller, "The Gradient Model Load Balancing Method," *IEEE Trans. Software Engineering*, Vol. 13, Jan. 1987, pp. 32-38.

[18] V. M. Milutinović, J. J. Crnković and C. E. Houstis, "A Simulation Study of Two Distributed Task Allocation Procedures," *IEEE Trans. Software Engineering*, Vol. 14, no. 1, Jan. 1988, pp. 54-61.

[19] R. Mirchandaney, D. Towsley and J. Stankovic, "Adaptive Load Sharing in Heterogeneous Distributed Systems," *J. Parallel and Distributed Computing*, Vol.9, no.4, Aug. 1990, pp. 331-346.

[20] S. B. Navathe, S. Ceri, G. Wiederhold and J. Dou, "Vertical Partitioning algorithms for Database Design," *ACM Trans. on Database Systems*, Vol.9, no. 4, Dec. 1984, pp. 680-710.

[21] L. M. Ni and K. Hwang, "Optimal Load Balancing in a Multiple Processor System with Many Job Classes," *IEEE Trans. Software Engineering*, Vol. 11, May 1985, pp. 491-496.

[22] L. M. Ni, C. W. Xu and Gendreau, "A Distributed Drafting Algorithm for Load Balancing," *IEEE Trans. Software Engineering*, Vol. 11, no. 11, Oct. 1985, pp. 1153-1161.

[23] E. Shamir and E. Upfal, "Probabilistic Approach to the Load Sharing Problem in Distributed Systems," *J. Parallel and Distributed Computing*, Vol.4, no.5, Oct. 1987, pp. 521-530.

[24] K. G. Shin and Y-C. Chang, "Load Sharing in Distributed Real-Time Systems with State-Change Broadcast," *IEEE Trans. Computers*, Vol. 38, no. 8, Aug. 1989, pp. 1124-1142.

[25] V. Srinivasan and G. L. Thompson, "Algorithms for Minimizing Total Cost, Bottleneck Time and Bottleneck Shipment in Transportation Problems," *Nav. Res. Log. Quart. 23*, 1976, pp. 567-595.

[26] J. A. Stankovic and I. Sidhu, "An Adaptive Bidding algorithm for Processes, Clusters and Distributed groups," *Proc. Intl. Conf. Distributed Comp. Systems*, May 1984.

[27] J. A. Stankovic, "An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling," *IEEE Trans. Computers*, Vol. 34, no. 2, Feb. 1985, pp. 117-130.

[28] H. S. Stone, "Multiprocessor Scheduling with the Aid of Network Flow Algorithms," *IEEE Trans. Software Engineering*, Vol. 3, no. 1, Jan. 1977, pp. 85-93.

[29] R. Varadarajan, "An Optimal algorithm for $2 \times n$ bottleneck transportation problems," to appear in *Operations Research Letters*, Vol.10, no. 9, Feb. 1992.

[30] B. W. Wah, "A systematic Approach to the Management of Data on Distributed Databases," *Ph.D. thesis, University of California, Berkeley, 1979.*

[31] B. W. Wah, "File Placement on Distributed Computer Systems," *IEEE Computer*, Vol.17, no. 1, Jan 1984, pp. 23-32.

[32] Y. T. Wang and R. J. T. Morris, "Load Sharing in Distributed Systems," *IEEE Trans. Computers*, Vol. 34, no. 3, Mar. 1985, pp. 204-217.

[33] C. M. Woodside and S. K. Tripathi, "Optimal Allocation of File Servers in a Local Network Environment," *IEEE Trans. Software Engineering*, Vol. 12, no. 8, Aug. 1986, pp. 844-848.

Biographical Sketch

Ravi Varadarajan received his Bachelor of Engineering (B.E.) (Honors) degree (1978) from the University of Madras, India and his Ph.d. (1987) degree in Computer Science from the University of Pennsylvania, Philadelphia. Since 1987, he has been an Assistant Professor in the Computer and Information Sciences department of the University of Florida, Gainesville. His research interests include design of efficient sequential and parallel algorithms, parallel architectures, distributed processing and theoretical computer science. He is a member of the IEEE Computer Society and ACM.