Scalable Computing: Practice and Experience
Volume 11, Number 3, pp. 221–237. http://www.scpe.org
ISSN 1895-1767

OPTIMIZING IMAGE CONTENT-BASED QUERY APPLICATIONS OVER HIGH LATENCY COMMUNICATION MEDIA, USING SINGLE AND MULTIPLE PORT COMMUNICATIONS

GERASSIMOS BARLAS*

Abstract. One of the earliest applications that explored the power and flexibility of the grid computing paradigm was medical image matching. A typical characteristic of such applications is the large communication overheads due to the bulk of data that have to be transferred to the compute nodes.

In this paper we study the problem of optimizing such applications under a broad model that incorporates not only communication overheads but also the existence of local data caches that could exist as a result of previous queries. We study the cases of both 1- and N-port communication setups. Our analytical approach is not only complimented by a theorem that shows how to arrange the sequence of operations in order to minimize the overall cost, but also yields closed-form solutions to the partitioning problem.

For the case where large load imbalances (due to big differences in cache sizes) prevent the calculation of a closed-form solution, we propose an algorithm for optimizing load redistribution.

The paper is concluded by a simulation study that evaluates the impact of our analytical approach. The simulation, which assumes a homogeneous parallel platform for easy interpretation of the results, compares the characteristics of the 1- and N-port setups.

Key words: parallel image registration, divisible load, high performance

1. Introduction. In the past five years there has been a big drive towards harnessing the power of parallel and distributed systems to offer improved medical services in the domain of 2D and 3D modalities. Content-based queries are at the core of these services, allowing physicians to achieve higher-accuracy diagnoses, conduct epidemiological studies or even acquire better training among other things [1].

In [2], the authors present a high-level overview of the methodologies used for medical image matching. The authors identify two broad types of approaches: *image retrieval* that utilizes similarity metrics to offer suitable candidate images and *image registration* that tries to fit the observed data onto fixed or deformable models. Finally, the authors suggest an integrated system architecture that could combine the advantages of the two approaches. A comprehensive review and classification of current medical image handling systems is published in [3].

Apart from the classification mentioned in [2], image registration techniques are also classified based on whether:

- Image features are used (control-point based) or the whole (or an area of interest) image (global registration).
- Work is done at the spatial or frequency domain.
- Global (rigid) or local (non-rigid) geometrical transformations are used.

The key problem is determining the optimum geometrical transformation. A brute-force approach entails huge computational requirements, leading researchers to either perform the search in several refinement steps [4, 5], or switch to heuristic techniques such as genetic/evolutionary algorithms and simulated annealing [6, 7]. Domain specific techniques have been also suggested [8].

A domain which has been enjoying early success is mammography [9, 1, 10]. Many projects that seek to harness the power of Grids [11] to offer advanced medical services have spawned over the last 8 years. A typical example is the MammoGrid project. Amendolia et.al present an overview of its service architecture design in [1]. On the other side of the Atlantic, the National Digital Mammography Archive Grid is a similar initiative [10]. A P2P system that seeks to address scalability issues that arise with the operation of typical client-server systems has been also proposed in [12].

While the problem of image registration is inherently 'embarrassingly' parallel, the domain has seen little work on performance optimization especially over heterogeneous platforms. In [5] the authors use wavelets to perform global registration in increasing refinement steps that allows them to reduce the search space involved. Zhou et al. also evaluate four parallelization techniques and derive their complexity in big-O notation by

*Department of Computer Science & Engineering, American University of Sharjah, P.O.B. 26666, Sharjah, UAE, gbarlas@aus.edu

implicitly assuming a homogeneous platform. However, they fail to take into account the communication overheads involved and use their analysis to optimize the load partitioning of their strategies.

Ino et al. propose a uniform inter-image 2D partitioning for performing 2D/3D registration, i. e. estimate the spatial location of a 3D volume from its projection on a 2D plane [13]. While Ino et al. discuss other possible distributions, they do not use an appropriate model that would allow for optimization. Subsequently, in [14] the authors compare very favourably a GPGPU approach with their parallel implementation on 2D/3D registration.

De Falco et al. have employed a differential evolution mechanism for estimating the parameters of an affine transformation for global registration [6]. The load distribution is performed on the population level, while at regular intervals, individuals are exchanged between neighboring nodes on the torus architecture used.

One of the early systems is the one described in [9]. Montagnat et.al use an array of high run-time cost, pixel-based, image retrieval algorithms to answer image similarity queries. As described in [15], the homogeneous system that is used to run the queries employs equal size partitioning, e.g. the M images that need to be compared against a new one, are split into k jobs of size $\frac{M}{k}$. In [15] the authors develop empirical cost models for each of the similarity metrics used to answer a content-based query. These are complemented by a study of the scheduling and data replication costs that are incurred upon submitting a job to a Grid platform.

While the models shown in [15] capture much of the inner workings of the algorithms used, they are not the most suitable for developing a strategy or criteria for optimizing the execution of content-based queries. Instead, they focus on estimating the optimum number of jobs to spawn, given the high associated cost of task/resource scheduling on Grids.

A particular problem in deriving an analytical partitioning solution is that upon performing a sequence of queries, the system is in a state where local image caches can reduce the communication cost. This is of course true as long as they refer to images of the same modality and type of content. To our knowledge, this paper is the first attempt to treat this problem in an analytical fashion that incorporates all the aforementioned system/problem parameters.

Our analytical approach belongs to the domain of Divisible Load Theory [16], which since its inception in the late 80s, has been successfully employed in a multitude of problems [17]. In [17] the problem of optimally partitioning and scheduling operations for two classes of problems identified as *query processing* and *image processing* respectively, has been studied. The problem characterizations were based on the communication characteristics and more specifically, the relation between the communication cost and the assigned load. This paper fills a gap left by that work by proposing a model and an analytical solution to *image-query processing* applications.

The contribution of our work is that for the first time a fully analytical model is employed to devise an optimizing strategy for the total execution time, given communication costs and the state (and not just the capabilities) of the parallel platform. Our simulation study shows that the benefits of the proposed framework are significant, in both a single-shot and a series of queries scenarios. Also, by isolating the specifics of the matching algorithms, our proposed solution is more adept to easy implementation and deployment, given the few system parameters that need to be known/estimated.

The organization of the paper goes as follows: in section 2 the cost model used in our analysis is introduced and explained within a broader context. Section 3.1 contains a study of the two-node scenario that cultivates to Theorem 3.1 for the optimum sequence of operations. The closed-form solutions to the partitioning problem for N nodes in 1-port configuration, is given in 3.2, while the N-port problem is solved in Section 4. An algorithm for managing the cache size of the compute nodes towards minimizing the execution time, is given in section 5. Finally, the simulation study in Section 6 highlights the benefits and drawbacks of our analytical approach and brings-up interesting facts about the different communication setups.

2. Model Formulation. The architecture targeted in this paper consists of N heterogeneous computing nodes that receive image data from a load originating node and return the results of the image matching process to it. The network architecture is a single-level tree or a bus-connected one. Because this can be a repetitive process, each node can build up a local image cache that can be reused for subsequent queries. Hence the load originating node has to communicate to the computing nodes only what they are missing, either because of the incorporation of new images or because of the departure of nodes from the computing pool.

Our treatment of the problem is based on the formulation of an affine model that describes the computation and communication overheads associated with the query data distribution, the image matching process and the

TABLE 2.1
Notations

Symbol	Description	Units
b	is the constant overhead associated with load distribution. It consists of the image to be matched in addition to any query specific data (e.g. matching thresholds).	B
d	is the constant overhead associated with result collection. Typically $d < b$.	B
e_X	is the part of the load which is resident at node X, i. e. a local image cache.	B
I	is the typical size of an image used for image matching.	B
L	the load that is has to be communicated to the computing nodes	B
l_X	is inversely proportional to the speed of the link connecting X and its load originating node.	sec/B
p_X	is inversely proportional to the speed of X.	sec/B
$part_X$	is the part of the load L assigned to X, hence $0 \leq part_X \leq 1$. The total load assigned to X is $part_X L + e_X$	NA

result collection phase. These models are closely related with the ones introduced in [17] although the semantics for some of the constants used here are different. Given a node X that is connected to a load originating node with a connection of (inverse) speed l_X, we assume that the load distribution t_{distr}, the computation t_{comp} and the result collection t_{coll} costs are given by:

$$t_{distr} = l_X \left(part_X L + b\right) \tag{2.1}$$

$$t_{comp} = p_X \left(part_X L + e_X\right) \tag{2.2}$$

$$t_{coll} = l_X d \tag{2.3}$$

The symbols used above, along with all the remaining ones to be introduced later in our analysis, are summarized in Table 2.1.

The total load to be processed by N nodes is

$$\sum_{i=0}^{N-1} \left(part_i L + e_i\right) \tag{2.4}$$

and for the communicated load parts we have:

$$\sum_{i=0}^{N-1} part_i = 1 \tag{2.5}$$

The contribution of the above components to the overall execution time of node X depends on how communication and computation overlap. We can identify two cases:

- **Block**-type computation: no overlap between communication and computation. Node X can start computing only after all data are delivered:

$$t_X = l_X \left(part_X L + b + d\right) + p_X \left(part_X L + e_X\right) \tag{2.6}$$

- **Stream**-type computation: node X can start using each local image cache immediately after receiving the query data. Computation can run concurrently with the communication of the extra data $part_X L$. There are two cases depending on the relative speed between communication and computation:
  - Communication speed is high enough to prevent X from going idle i. e.

$$p_X \left(part_X L + e_X - I\right) \geq l_X part_X L \tag{2.7}$$

where I is the size of the last image to be compared against the required one. Then:

$$t_X = l_X \left(b + d\right) + p_X \left(part_X L + e_X\right) \tag{2.8}$$

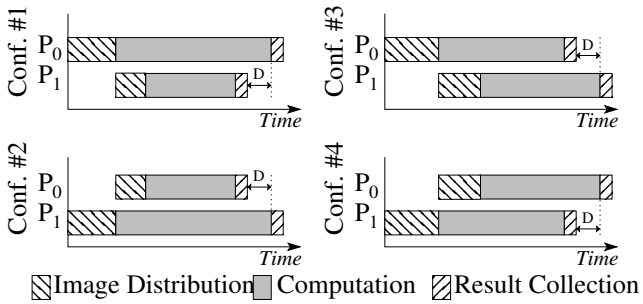

FIG. 3.1. *The four possible configurations of processing by two nodes when 1-port communications are used. Result collection is assumed to be separated by a constant delay D.*

– Node X has to wait for the delivery of data through a slow link, i. e. condition (2.7) is invalid. Then:

$$t_X = l_X\,(b+d) + l_X part_X L + p_X I \tag{2.9}$$

The additional parameter that controls the overall cost when N nodes are used, is whether single-port or N-port communications are employed, e.g. whether the load originating node can distribute L concurrently to multiple nodes.

In the remaining sections we focus on block-type tasks under both 1- and N-port communication setups. Our derivations are based on the assumptions of uniform communication media, i. e. $l_i = l\ \forall i$. A comparison between the two communication setups is performed in section 6.

It should be noted that the static model proposed in this paper, while not apparently suitable for a grid computing scenario, in which computation and communication costs change over time, it can form the basis for an adaptive scheduler that modifies load distribution over time given cost estimates. This goes beyond the scope of this paper and should be the topic of further research.

3. The 1-port Communication Case.

3.1. The two-node scenario. If we assume that there is a load originating node that distributes the load to two nodes, then if single port communications and a single installment [16] are used, the possible sequences of communication and computation operations are shown in Fig. 3.1, as imposed by the need to have no gaps between stages (otherwise, execution time is not minimized). For reasons that will become obvious in the rest of the section, we also assume that the two result collection phases are separated by a constant delay D.

The total execution time for configuration #1 is given by:

$$t_1 = l\,(part_0 L + b) + p_0\,(part_0 L + e_0) + ld \tag{3.1}$$

where

$$p_0(part_0 L + e_0) = l(part_1 L + b) + p_1(part_1 L + e_1) + l\ d + D \tag{3.2}$$

Eq. (3.2) coupled with the normalization equation $part_0 + part_1 = 1$ can provide a solution for $part_0$ and t_1. A similar procedure can produce the times for the three remaining configurations. Thus we can form the pairwise differences of running times:

$$t_3 - t_4 = \frac{l\,(e_1 p_1 - e_0 p_0) + (dl - bl + D)\,(p_1 - p_0)}{p_0 + p_1 + l} \tag{3.3}$$

$$t_3 - t_2 = \frac{l\,(e_1 p_1 - e_0 p_0 - b\,(p_1 - p_0) - dl - D)}{p_0 + p_1 + l} \tag{3.4}$$

$$t_3 - t_1 = \frac{(dl + D)(p_1 - p_0 - l)}{p_0 + p_1 + l} \tag{3.5}$$

$$t_1 - t_4 = \frac{l\,(e_1 p_1 - e_0 p_0 - b\,(p_1 - p_0) + d\;l + D)}{p_0 + p_1 + l} \tag{3.6}$$

$$t_1 - t_2 = \frac{l\,(e_1 p_1 - e_0 p_0 - (b + d)\,(p_1 - p_0))}{p_0 + p_1 + l} - \frac{D\,(p_1 - p_0)}{p_0 + p_1 + l} \tag{3.7}$$

$$t_4 - t_2 = -\frac{(d\;l + D)\,(p_1 - p_0 + l)}{p_0 + p_1 + l} \tag{3.8}$$

Clearly, the problem is too complex to have a single solution even for the simplest case of two nodes. We can however isolate a number of useful special cases that make a closed form solution to the N-node problem tractable:

- **No image caches** ($e_0 = e_1 = 0$). If we assume than $p_0 \leq p_1$ and given that $b > d$, we have:

$$t_3 - t_4 = \frac{(dl - bl + D)\,(p_1 - p_0)}{p_0 + p_1 + l} \tag{3.9}$$

$$t_3 - t_2 = \frac{l\,(-b\,(p_1 - p_0) - d\;l - D)}{p_0 + p_1 + l} \leq 0 \tag{3.10}$$

$$t_3 - t_1 = \frac{(d\;l + D)\,(p_1 - p_0 - l)}{p_0 + p_1 + l} \tag{3.11}$$

If $dl - bl + D \leq 0 \Rightarrow D \leq l\,(b - d)$, then Eq. (3.11) dictates that either configuration #3 or configuration #1 are optimum based on whether $p_1 - p_0 - l$ is negative or not. If we assume that the differences in execution speed are small relative to the communication cost l (i. e. $p_1 - p_0 \leq l$) then configuration #3 is the optimum one.
The execution time is given by

$$t_3^{(nc)} = l\left(part_0^{(nc)} L + b\right) + p_0 part_0^{(nc)} L + D + 2ld \tag{3.12}$$

where:

$$part_0^{(nc)} = \frac{p_1 L + l(L - d + b) - D}{L\,(p_0 + p_1 + l)} \tag{3.13}$$

- **Homogeneous system** ($p_0 = p_1 = p$). If we assume that $e_0 \geq e_1$ then:

$$t_3 - t_4 = \frac{pl\,(e_1 - e_0)}{2p + l} \leq 0 \tag{3.14}$$

$$t_3 - t_1 = -\frac{l\,(d\;l + D)}{2p + l} \leq 0 \tag{3.15}$$

$$t_1 - t_2 = \frac{pl\,(e_1 - e_0)}{2p + l} \leq 0 \tag{3.16}$$

which again translates to having configuration #3 as the optimum one. It should be noted that the optimum order dictates that load is sent first to the node with the biggest cache, which is a counter-intuitive result! The execution time is given by

$$t_3^{(homo)} = l\left(part_0^{(homo)} L + b\right) + p\left(part_0^{(homo)} L + e_0\right) + D + 2ld \tag{3.17}$$

where:

$$part_0^{(homo)} = \frac{p(L + e1 - e0) + l(L - d + b) - D}{L\,(2p + l)} \tag{3.18}$$

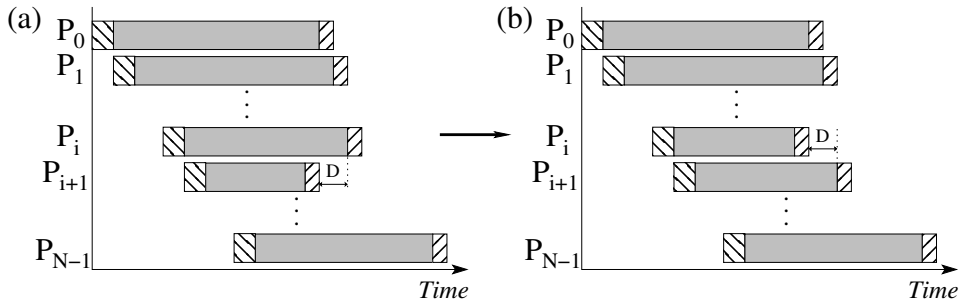

FIG. 3.2. *(a) A possible ordering of load distribution and result collection for N nodes. (b) Improving the execution time by ordering the operations of P_i and P_{i+1} in non-decreasing order of their speed (assuming $p_i \leq p_{i+1}$). Note: the phase durations are disproportionate to actual timings.*

The delay D that was introduced above allows us to extend our analytical treatment from 2 nodes to N. D is supposed to model the time taken by the result collection operations of other nodes. Hence, D is a multiple of $d \cdot l$ with a maximum value of $(N-2)\, d \cdot l$. For the case of no-caches, as long as $D \leq l\,(b-d) \Rightarrow N \leq \frac{b}{d} + 1$ and the differences in computation speed are smaller than the communication speed, configuration #3 is the optimum one as stated by the following theorem. Given the 2-3 orders of magnitude difference expected between b and d, the range of N that the theorem applies is quite broad.

THEOREM 3.1. *The optimum load distribution and result collection order for an image query operation performed by N nodes is given by:*

- ***No image caches**: distributing the load and collecting the results in non-increasing order of the nodes' speed (i. e. in non-decreasing order of the p_i parameters). The sufficient but not necessary conditions for this to be true is $N \leq \frac{b}{d} + 1$ and $|p_i - p_j| \leq l$ for any pair of nodes i, j.*
- ***Homogeneous system**: distributing the load and collecting the results in non-increasing order of the local image cache sizes.*

Proof. We will prove the above theorem for the no-caches case via contradiction. The proof for the homogeneous case is identical. Let's assume that the optimum order is similar to the one shown in Fig. 3.2(a). Without loss of generality we assume that the distribution order is $P_0, P_1, \ldots P_{N-1}$

For any two nodes P_i and P_{i+1} that do not satisfy the order proposed by Theorem 3.1, we can rearrange the distribution and collection phases so as the part of the load that is collectively assigned to them $(L\,(part_i + part_{i+1}))$ is processed in a shorter time frame (as long as $N \leq \frac{b}{d} + 1$), while occupying in an identical fashion the communication medium (see Fig. 3.2(b)). Thus, the operation of the other nodes is not influenced. At the same time the shorter execution time would allow additional load to be given to nodes P_i and P_{i+1} resulting in a shorter total execution time. The outcome is a contradiction to having the original ordering being an optimum one. The only ordering that cannot be improved upon by the procedure used in this proof, is the one proposed by Theorem 3.1. □

The above discussion settles the ordering problem, allowing us to generate a closed-form solution to the partitioning problem for N nodes.

3.2. Closed-form solution for N nodes.

3.2.1. No image caches. The following relation holds between every pair of nodes which are consecutive in the distribution and collection phases (without loss of generality we will again assume that the nodes' order is $P_0, P_1, \ldots, P_{N-1}$):

$$\begin{aligned} p_i part_i L + ld &= l\,(part_{i+1} L + b) + p_{i+1} part_{i+1} L \Rightarrow \\ part_{i+1} &= part_i \frac{p_i}{p_{i+1} + l} + \frac{l\,(d-b)}{L\,(p_{i+1} + l)} \end{aligned} \tag{3.19}$$

This can be extended to any pair of nodes P_i and P_j, where $i > j$:

$$part_i = part_j \prod_{k=j}^{i-1} \frac{p_k}{p_{k+1} + l} + \frac{l\,(d-b)}{L} \sum_{k=j+1}^{i} \left[(p_k + l)^{-1} \prod_{m=k}^{i-1} \frac{p_m}{p_{m+1} + l} \right] \tag{3.20}$$

Equipped with Eq. (3.20), we can associate each $part_i$ with $part_0$ and use the normalization equation:

$$\sum_{i=0}^{N-1} part_i = 1 \tag{3.21}$$

to compute a closed form solution for $part_0$:

$$part_0 = \frac{1 - \frac{l(d-b)}{L} \sum_{i=1}^{N-1} \sum_{k=1}^{i} \frac{\prod_{m=k}^{i-1} \frac{p_m}{p_{m+1}+l}}{(p_k+l)}}{1 + \sum_{i=1}^{N-1} \prod_{k=0}^{i-1} \frac{p_k}{p_{k+1}+l}} \tag{3.22}$$

Equations (3.22) and (3.20) solve the partitioning problem. The total execution time is:

$$t_{total}^{(nc)} = l\left(part_0 L + b\right) + p_0 part_0 L + N\ l\ d \tag{3.23}$$

The above constitute a closed form solution that can be computed in time $\frac{N^2-N}{2} + 3(N-1) + Nlg(N) = O\left(N^2\right)$, where $Nlg(N)$ is the node-sorting cost.

3.2.2. Homogeneous System. Following a similar procedure to the previous section, it can be shown that:

$$\begin{aligned} p\left(part_i L + e_i\right) + l\ d = l\left(part_{i+1} L + b\right) + p\left(part_{i+1} L + e_{i+1}\right) \Rightarrow \\ part_{i+1} = part_i \frac{p}{p+l} + \frac{l\left(d-b\right) + p\left(e_i - e_{i+1}\right)}{L\left(p+l\right)} \end{aligned} \tag{3.24}$$

This can be extended to any pair of nodes P_i and P_j, where $i > j$:

$$part_i = part_j \left(\frac{p}{p+l}\right)^{i-j} + \sum_{k=j}^{i-1} \frac{l\left(d-b\right) + p\left(e_k - e_{k+1}\right)}{L\left(p+l\right)} \left(\frac{p}{p+l}\right)^{i-k-1} \tag{3.25}$$

Again, Eq. (3.25), and the normalization equation can produce a closed form solution for $part_0$:

$$part_0 = \frac{d-b}{L} + \frac{l + \frac{l\ N(b-d)}{L}}{p + l - p\left(\frac{p}{p+l}\right)^{N-1}} - \frac{l \sum_{i=1}^{N-1} \sum_{k=0}^{i-1} \frac{(e_k - e_{k+1})}{L} \left(\frac{p}{p+l}\right)^{i-k}}{p + l - p\left(\frac{p}{p+l}\right)^{N-1}} \tag{3.26}$$

The total execution time can be then computed as:

$$t_{total}^{(homo)} = l\left(part_0 L + b\right) + p\left(part_0 L + e_0\right) + N\ l\ d \tag{3.27}$$

As with the previous case, the solution requires an $O\left(N^2\right)$ computational cost.

A special case needs to be considered if $L = 0$ as the above equations cannot be applied. The minimum execution can be achieved only if the local caches are appropriately sized to accommodate this. Similarly to Eq. (3.25) for two nodes P_i and P_j, where $i > j$ we would have:

$$\begin{aligned} p_i e_i + (j-i) l d = (j-i) l b + p_j e_j \Rightarrow \\ e_j = e_i \frac{p_i}{p_j} + \frac{(j-i) l (d-b)}{p_j} \end{aligned} \tag{3.28}$$

If the caches do not satisfy condition (3.28), the load must be reassigned/transferred between nodes. In this paper we assume that this is performed by the load originating node and not by a direct exchange between the compute nodes. Section 5 elaborates more on how we can treat this case.

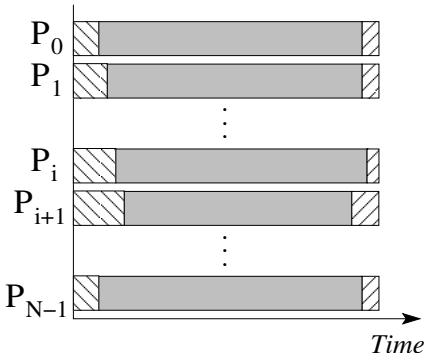

FIG. 4.1. *Optimum scheduling for a N-port communication setup.*

4. The N-port Communication Case.

4.1. Closed-form solution for N nodes. The N-port communication case is much simpler than the 1-port one since no explicit node ordering is necessary. It can be easily shown in this case that the optimum load partitioning has to produce identical running times on all the participating compute nodes, i. e. all nodes must start receiving data and finish delivering results at the same instant. Since all nodes must have the same starting and ending times as shown in Fig.4.1, for any two nodes i and j, the following has to hold:

$$\begin{aligned} l\left(part_i L + b\right) + p_i\left(part_i L + e_i\right) + ld = \\ l\left(part_j L + b\right) + p_j i\left(part_j L + e_j\right) + ld \Rightarrow \\ part_i L\left(p_i + l\right) + p_i e_i = part_j L\left(p_j + l\right) + p_j e_j \Rightarrow \\ part_i = part_j \frac{p_j + l}{p_i + l} + \frac{p_j e_j - p_i e_i}{L\left(p_i + l\right)} \end{aligned} \tag{4.1}$$

The normalization equation (3.21) can then be used to produce a closed-form solution for $part_0$ and subsequently all $part_i$:

$$\begin{aligned} \sum_{i=0}^{N-1} part_i = 1 \Rightarrow \\ part_0 \sum_{i=0}^{N-1} \frac{p_0 + l}{p_i + l} + \sum_{i=1}^{N-1} \frac{p_0 e_0 - p_i e_i}{L\left(p_i + l\right)} = 1 \Rightarrow \\ part_0 = \frac{1 + \sum_{i=1}^{N-1} \frac{p_i e_i - p_0 e_0}{L(p_i + l)}}{\sum_{i=0}^{N-1} \frac{p_0 + l}{p_i + l}} \end{aligned} \tag{4.2}$$

The total execution time is given by:

$$t_{total}^{(Nport)} = l\left(part_0 L + b\right) + p_i\left(part_0 L + e_i\right) + ld \tag{4.3}$$

4.2. Homogeneous System Solution. For a homogeneous system ($\forall p_i \equiv p$), the above equations are simplified to the following:

$$(4.1) \Rightarrow part_i = part_j + \frac{p\left(e_j - e_i\right)}{L\left(p + l\right)} \tag{4.4}$$

$$(4.2) \Rightarrow part_0 = N^{-1}\left(1 + \frac{p}{L}\sum_{i=1}^{N-1} \frac{e_i - e_0}{p + l}\right) \tag{4.5}$$

which translates to having differences in the local caches as the single cause of any imbalances in the split of the new load L. Otherwise the load should be evenly split.

5. Image Cache Management. Eq. (3.20), (3.25), (4.1) and (4.4) allow for negative values for $part_i$s. Such an event indicates that the corresponding node should not participate in the calculation, either because it is too slow or because the local cache size is too large for a node to process and keep up with the other nodes. In the latter case it is obvious that a node should use only a part of its cache. The load surplus should be transferred to other nodes. This situation can arise when following the initial distribution of load to the nodes, subsequent queries are no longer accompanied by big chunks of data, making the initial distribution a suboptimal one.

In this section we address this problem by proposing a algorithm for estimating the proper cache size that should be used, along with the corresponding load L that should be communicated to other nodes.

The algorithm presented below, is based on the assumption that the intersection of all caches is $\emptyset$. The key point of the algorithm is a re-assignment of load from the nodes with an over-full cache (identified as set S in line 8) to the nodes with little or no cache. This process reduces the total execution time as long as communication is faster than computation.

This algorithm has been also enhanced from the version presented in [18] to address the case when $L = 0$, i. e. when processing is based entirely on the nodes' local data. In that case, L can be initialized to a small value, e.g. $L = 1$ (lines 2-5), which would be subsequently subtracted once a redistribution is dimmed necessary (lines 32-36).

Set S does not change after line 8 as the subsequent increase in L due to a load shift (line 31) does not permit any other node from having a negative assignment. The loop of lines 13-46 is executed for as long as there is a negative $part_i$, or a load shift is necessary for balancing the node workload. In line 17 the size of the cache that should be used in a node with a negative assignment is estimated. Because the load is reassigned collectively in line 31, the cache size of each node in S can be under-estimated (by "bloating" the load L that should be communicated). This defeats the optimization procedure by forcing the communication of data that are already present at the nodes, and in order to guard against this possibility, lines 21-28 re-adjust any previous overestimation for nodes that subsequently got to have positive $part_j$. Lines 12 and 41-44 serve as sentinels against cases where the outer *while* loop does not converge. In that case, fixing the part assigned to the last node in the distribution sequence (smallest e) to 0, allows the convergence of the outer loop. A value for threshold $THRES$ that was found to yield good results in our experiments is 20. Threshold values that depend on the number of compute nodes did not provide any visible difference.

Lines 32-36 cancel the addition of 1 load unit that is done when $L = 0$. Finally, if L remains 0 after load redistribution is examined, cache sizes satisfy condition (3.28) for a homogeneous system and nothing more needs to be done (lines 37-40).

A key point that should be made here is that Algorithm 1 produces a sub-optimum solution when a *series* of query operations are to be scheduled. Designing an optimum algorithm for this scenario is beyond the scope of this paper.

6. Simulation Study. Single-port communication is surely not a contemporary technology limitation. It is rather a design feature whereas the load originating node dedicates its attention to a single node at a time, with the objective of minimizing the message exchange cost between itself and the corresponding node. In this section we explore the impact of the two alternative design choices with the assistance of our analytical framework. Also, we evaluate the performance achieved by the use of Algorithm 1 for managing the image caches through a battery of image queries.

We base the bulk of our simulations on the assumption of a homogeneous platform. While the requirement of a homogeneous system may seem unrealistic, it can be typical of many large scale installations in big organizations.

The key points of our simulation scenario which consists of a series of image query operations, are the following:

- The image DB[1] consists originally of 10000 images of size 1MB each. This is a small number relevant to the yearly "production" of mammograms generated at a national level. Additionally, the image size matches real data only in the order of magnitude as high resolution mammograms can be much larger (e.g. 8MB).
- Each new image that is matched against the DB is also 1MB in size, hence $b = 1MB$.

[1]We use the term DB to loosely refer to the collection of available, tagged, medical images, and not to an actual DBMS system. Storage services are offered in MammoGrid [1] by MySQL and in NDMA by IBM's DB2 [10]

Algorithm 1 Estimating the local image cache sizes that yield the minimum execution time for the next query operation

1: $load\_shift \leftarrow 0$
2: **if** $L = 0$ **then**
3: $added \leftarrow TRUE$
4: $L \leftarrow 1$
5: **end if**
6: In the case of 1-port communication and a homogeneous system, sort the nodes in descending order of their e_i parameters.
7: Calculate the load part for each node P_i via Eq. (3.26), (3.25) or (4.2), (4.1)
8: Let S be the set of nodes with $part_j < 0$
9: **if** $S \neq \emptyset$ **then**
10: Copy the cache sizes of all nodes in temporary variables $e_i^{(orig)}$
11: **end if**
12: $iter \leftarrow 0$
13: **while** $S \neq \emptyset$ OR $load\_shift \neq 0$ OR $added = TRUE$ **do**
14: $load\_shift \leftarrow 0$
15: **for** each $P_j \in S$ **do**
16: **if** $part_j < 0$ **then**
17: $aux \leftarrow part_j L + e_j$
18: $load\_shift \leftarrow load\_shift + e_j - aux$
19: $e_j \leftarrow aux$
20: **else**
21: $aux \leftarrow part_j L + e_j$
22: **if** $aux > e_i^{(orig)}$ **then**
23: $diff \leftarrow e_j^{(orig)} - e_j$
24: **else**
25: $diff \leftarrow aux - e_j$
26: **end if**
27: $load\_shift \leftarrow load\_shift - diff$
28: $e_j \leftarrow e_j + diff$
29: **end if**
30: **end for**
31: $L \leftarrow L + load\_shift$
32: **if** $added = TRUE$ **then**
33: $added \leftarrow FALSE$
34: $L \leftarrow L - 1$
35: $load\_shift \leftarrow 1$
36: **end if**
37: **if** $L = 0$ **then**
38: Set for all nodes P_j, $part_j \leftarrow 0$
39: BREAK
40: **end if**
41: $iter \leftarrow iter + 1$
42: **if** $iter > THRES$ **then**
43: Fix the $part_k$ assigned to the node with the smallest e_k to 0
44: **end if**
45: Calculate the load part for each P_i, other than the nodes fixed in step 43.
46: **end while**

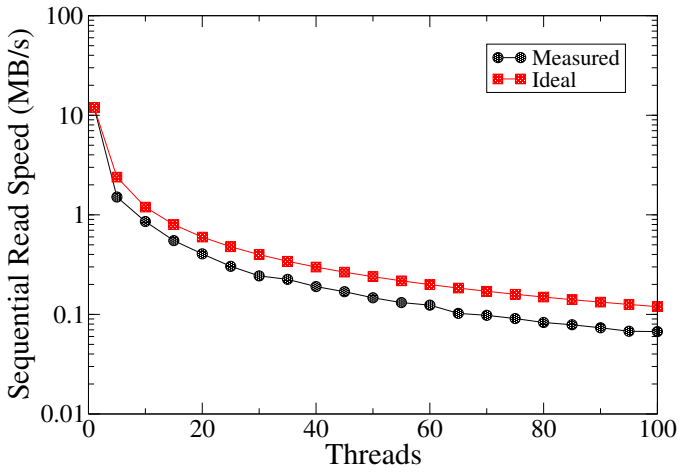

FIG. 6.1. *Average sequential disk read speed per thread. The ideal curve represents the case where the total bandwidth is evenly divided between the threads without losses.*

- Every 100 queries, 100 appropriately tagged images are incorporated in the image DB, hence the resident load increases gradually.
- The data collected from each node consist of the best 10 matches, along with the corresponding image IDs and objective function values, assumed in total to be of size $d = 10 \cdot (2 + 4) = 60B$.
- The *tiobench* utility [19] was used to estimate realistic values for the data rates between the load originating nodes and the compute nodes. A variable number of threads were used to represent simultaneous access from multiple clients. The results which were collected on a Linux laptop machine, equipped with a ATA 100 100GB hard disk spinning at 4200rpm, formatted using the ReiserFS filesystem, are shown in Fig. 6.1. The effect of the disk cache was minimized by using a 3GB file size. These speeds were used in the 1- and N-port simulations that are reported in this paper. For 1-port communications in particular, l was set equal to $0.00997 sec/Mb$, which translates to $0.0837 sec/image$.

The first question we would like to answer, is what would be the improvement of using our analytical approach over an Equal load Distribution (ED) strategy that is traditionally used in homogeneous systems [15], in a single-shot scenario, i. e. when only one query operation is performed. For this purpose, we tested both 1- and N-port approaches, where the computing speed of all nodes was set to be one of the following values $\{0.08, 0.17, 0.33, 0.67, 1.34\} sec/image$, roughly corresponding to 1x, 2x, 4x, 8x and 16x the time required to communicate a single image when 1-port communication is used. In the remainder of this section we will refer to these processing speeds as $1l$, $2l$, $4l$, $8l$ and $16l$ respectively. Such a selection of processing speeds/costs matches closely the running times reported in [15] for real-life tests and they are supposed to help us probe the effects of different computation/communication ratios and the use of different image registration algorithms.

The results for the 1-port case are shown in Figure 6.2 in the form of the improvement achieved over the ED approach. In all the comparative results reported in this section, we use the execution time provided by the 1-port non-uniform proposed distribution strategy (as given by Eq.(3.27) and denoted below as t_{SP}) as the baseline. The improvement is defined as:

$$\frac{t_{ED} - t_{SP}}{t_{SP}} \tag{6.1}$$

which is basically the percent overhead that ED (t_{ED}) is causing over the proposed analytical solution. All initial caches were set equal to 0 which is a typical initialization scenario. It should be noted that all the results reported in Fig. 6.2 and the remaining graphs of this section, correspond to cases where all available nodes can be utilized, hence the lack of data points for big values of N when p is relatively small. This qualification was imposed to avoid skewed results.

As can be observed in Fig. 6.2, the improvement is even higher when the computational cost is proportionally higher than the communication, topping around 28% for the $p = 16l$ case. In the majority of the tested cases, the gain is above 10%.

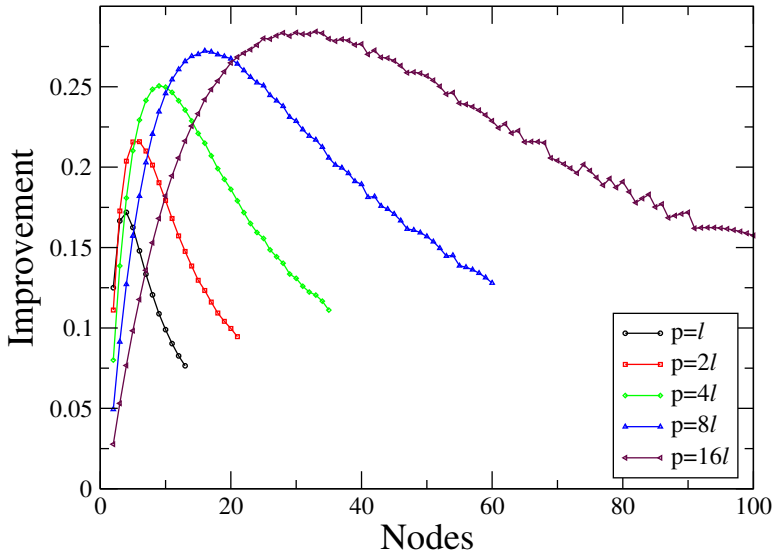

FIG. 6.2. *Execution time improvement offered by the proposed scheme over the ED strategy, for a single-shot scenario.*

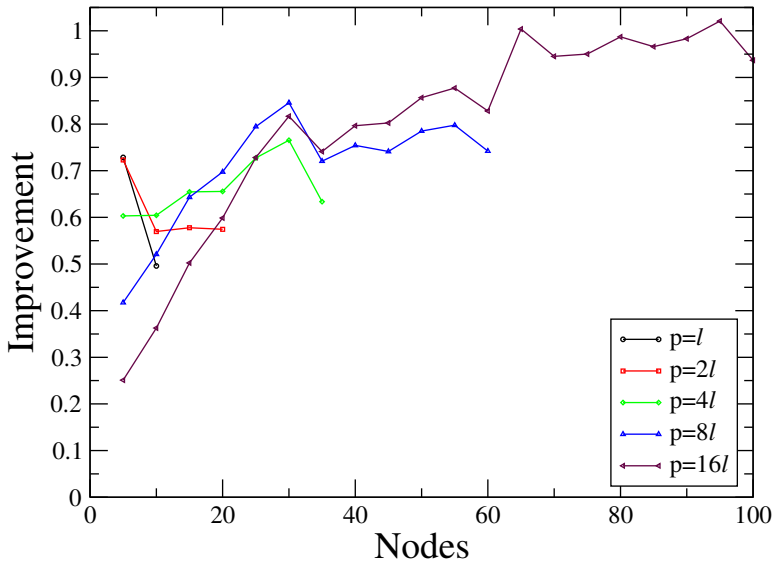

FIG. 6.3. *Execution time improvement offered by the 1-port over the N-port approach, for a single-shot scenario.*

Comparing the 1-port and N-port cases is less straightforward as there is a question of whether the N-port communication setup is accomplished by sharing the same medium - as is usually the case in non-dedicated platforms such as Networks of Workstations (NoW) -, or the load originating node is having a dedicated link for each worker. In the following paragraphs we assume that the former setup is applicable.

The improvement offered (!) by the 1-port over the N-port case is shown in Figure 6.3, where improvement is now computed by Eq. (6.1) by replacing t_{ED} with the execution time of the N-port arrangement t_{NP}. It comes as a surprise that the N-port arrangement can be such a poor performer! The reasons can be summarized as follows: (a) sharing the communication medium causes the computation phase to be overly delayed while data are being downloaded and (b) the cost of switching is taking a heavy toll on the available bandwidth, as observed in Figure 6.1 if one compares the measured against the ideal curves. In summary, the 1-port setup allows -some of- the compute nodes to start processing the load a lot sooner. Of course this result has to be seen in the *proper context*, i. e. we have block-type tasks and the nodes have no image cache. As it will be shown below, this picture is far from the truth for a sequence of query operations.

In order to test what would be the situation if a sequence of queries were performed, we simulated the successive execution of 1000 queries. The corresponding improvement for the 1-port scheme is shown in Fig. 6.4 (a). As can be observed, the ED strategy is not worst in every case due to the cost of cache redistribution that Algorithm 1 is causing. Actually for fast computation ($p = l$) and a relatively small number of nodes, ED is faster. For the majority of the other cases, the gains seems insignificant (in the order of 1%) as the constant shuffling of the caches slows down the whole process. These effects can be minimized if queries are run in batches as can be clearly seen in Fig.6.4 (b) and (c), for moderate (10 queries) and extreme batch sizes (100 queries) respectively. For batch processing the same analytical models can be applied, if we multiply the constants b, d and p by the batch size. Batching requests together does not come close to optimizing a sequence of them as performed in [20], but as it is shown in Fig.6.4, boosts performance substantially. Under such conditions the proposed strategy is consistently better than the ED one, although the actual gains depend on the ratio between computation and communication costs. If the former are dominant (e.g. as in the $p = 16l$ case), any benefits made by effectively scheduling the communication operations is marginalized.

Fig. 6.4 does not convey the complete picture though, as the gains seem insignificant. However, when the running times are as high as shown in Fig. 6.5 even small gains translate to big savings in time.

For the N-port case, batching requests produces small absolute savings as shown at the bottom of Fig. 6.5 (b), (c). While the gain barely reaches 1 hour overall, the real benefit comes from increased scalability, i. e. the ability to use bigger sets of processors for the task. For example, for $p = l$ batches of 100 queries can run on 100 nodes, while individually queries are limited to 13 nodes.

The picture is completely reversed for the N-port case when multiple queries are considered, as can be observed in Fig. 6.6. Even with the reduced bandwidth available to each compute node and the deterioration of the total available bandwidth, the N-port approach is a hands-down winner. This is especially true when the number of nodes grows beyond a limit, making this the most scalable strategy, despite the bandwidth loss identified in Fig. 6.1. Additionally, batching queries together benefits the N-port approach even more than the 1-port, non-uniform one.

7. Conclusion. In this paper we present an analytical solution to the problem of optimizing content-based image query processing over a parallel platform under communication constraints. We solve the problem analytically for both the single and N-port cases and we also prove an important theorem for the sequence of operations that minimize the execution time. Our analytical solution is accompanied by an algorithm for the cache management of the nodes of a system, either 1-port homogeneous or N-port heterogeneous. Our closed-form solution for the 1-port heterogeneous case with no image caches, can be employed when a single-shot operation is preferred.

The extensive simulations that were conducted were able to reveal the following design principles, as far as homogeneous platforms are concerned:

- If a single-shot execution is desired, a 1-port non uniform distribution as highlighted in Section 3.2.2 is the best one.
- For a sequence of operations, the N-port strategy is the best performer, especially if the computational cost is proportionally higher, or the number of nodes is high.

Future research directions could include:

- Using the proposed methodology as a part of a Grid middleware scheduler. It is possible that the high overhead of typical grid schedulers compromises the benefits shown in this paper, requiring further optimizations.
- Devising a solution for a heterogeneous system with local cache.
- Examine the case of multiple image sources instead of a single load originating node. Although current generation systems rely mostly on a single image repository, next generation ones are moving away from this paradigm [1].

REFERENCES

[1] S. R. Amendolia, F. Estrella, R. McClatchey, D. Rogulin, and T. Solomonides, "Managing pan-european mammography images and data using a service oriented architecture," in *Proc. IDEAS Workshop on Medical Information Systems: The Digital Hospital (IDEAS-DH'04)*, September 2004, pp. 99–108.

[2] M. M. Rahman, T. Wang, and B. C. Desai, "Medical image retrieval and registration: Towards computer assisted diagnostic approach," in *Proc. IDEAS Workshop on Medical Information Systems: The Digital Hospital (IDEAS-DH'04)*, September 2004, pp. 78–89.

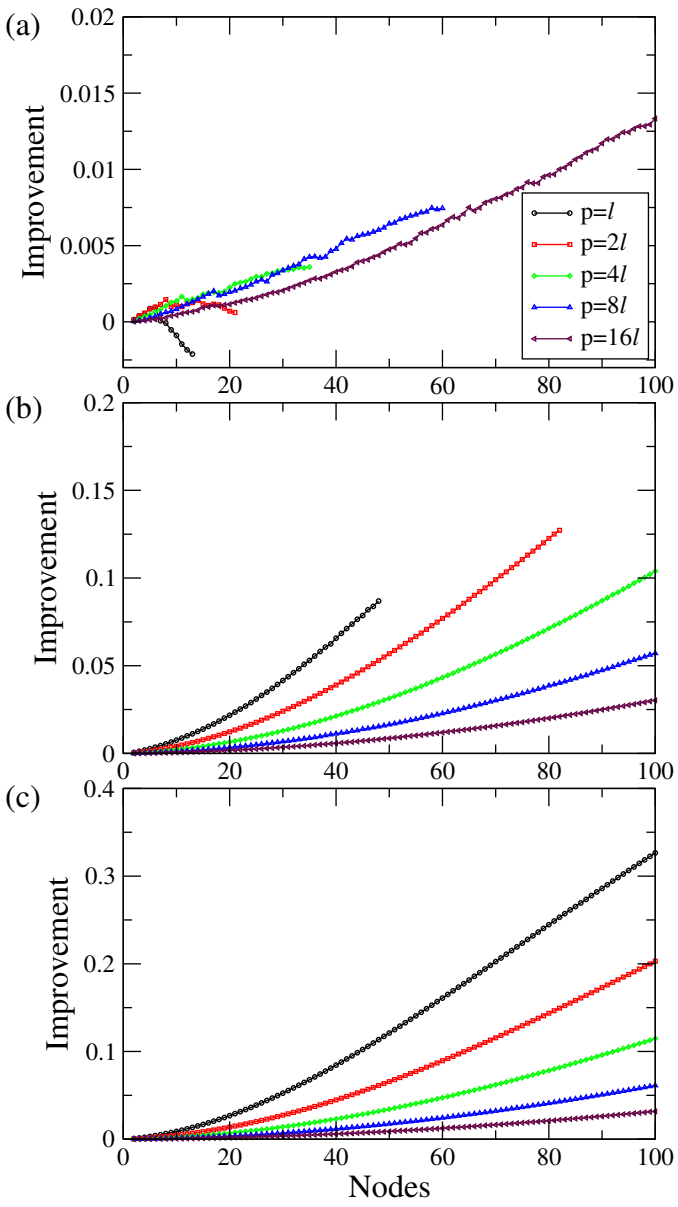

FIG. 6.4. *Execution time improvement offered by the 1-port scheme over the ED strategy, for a sequence of 1000 queries: (a) when each query is run individually, (b) when queries are run in batches of 10, and (c) when queries are run in batches of 100.*

[3] S. Sneha and A. Dulipovici, "Strategies for working with digital medical images," in *Proc. 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, January 2006, p. 100a.

[4] Y. Kawasaki, F. Ino, Y. Sato, S. Tamura, and K. Hagihara, "Parallel adaptive estimation of range of motion simulation for total hip replacement surgery," *IEICE Transactions on Information and Systems*, vol. E90-D, no. 1, pp. 30–39, 2007.

[5] H. Zhou, X. Yang, H. Liu, and Y. Tang, "First evaluation of parallel methods of automatic global image registration based on wavelets," in *2005 International Conference on Parallel Processing (ICPP'05)*, July 2005, pp. 129–136.

[6] I. D. Falco, D. Maisto, U. Scafuri, E. Tarantino, and A. D. Cioppa, "Distributed differential evolution for the registration of remotely sensed images," in *15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'07)*, February 2007, pp. 358–362.

[7] S. Ait-Aoudia and R. Mahiou, "Medical image registration by simulated annealing and genetic algorithms," in *Geometric Modelling and Imaging (GMAI '07)*, July 2007, pp. 145–148.

[8] Y. Bentoutou, N. Taleb, K. Kpalma, , and J. Ronsin, "An automatic image registration for applications in remote sensing," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 43, no. 9, pp. 2127–2137, September 2005.

[9] J. Montagnat, H. Duque1, J. Pierson, V. Breton, L. Brunie, and I. E. Magnin, "Medical image content-based queries using the grid," in *Proc. of HealthGrid 03*, 2003.

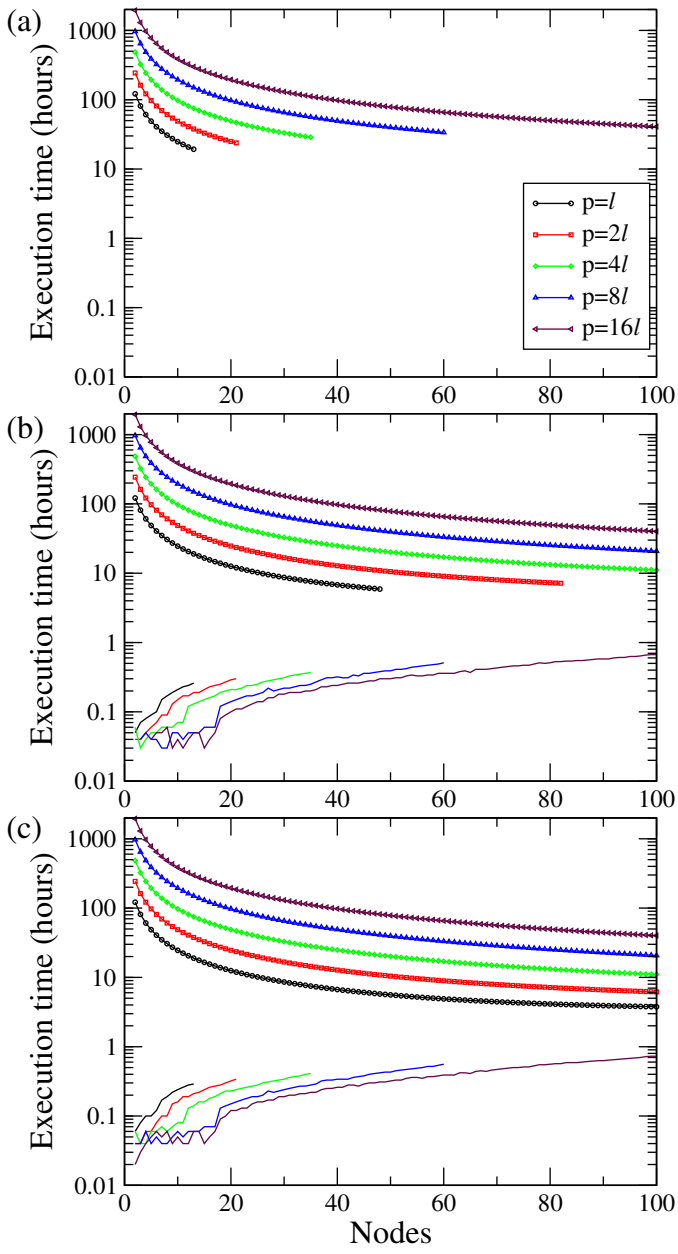

FIG. 6.5. *Total execution time offered by Algorithm 1 for a sequence of 1000 queries, under 1-port configuration: (a) when each query is run individually, (b) when queries are run in batches of 10, and (c) when queries are run in batches of 100. The plain lines at the bottom of (b) and (c) show the corresponding absolute time gain over (a).*

[10] University of Pennsylvania Consortium and National Digital Mammography Archive Grid: http://www.ibm.com/e-business/ondemand/us/innovation/univofpa.shtml [Online]. Available: http://www.ibm.com/e-business/ondemand/us/innovation/univofpa.shtml

[11] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," DRAFT document available at http://www.globus.org/research/papers/ogsa.pdf. [Online]. Available: http://www.globus.org/research/papers/ogsa.pdf

[12] I. Blanquer, V. Hernandez, and F. Mas, "A peer-to-peer environment to share medical images and diagnoses providing context-based searching," in *13th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'05)*, February 2005, pp. 42–48.

[13] F. Ino, Y. Kawasaki, T. Tashiro, Y. Nakajima, Y. Sato, S. Tamura, and K. Hagihara, "A parallel implementation of 2-d/3-d image registration for computer-assisted surgery," in *11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS'05)*, July 2005, pp. 316–320.

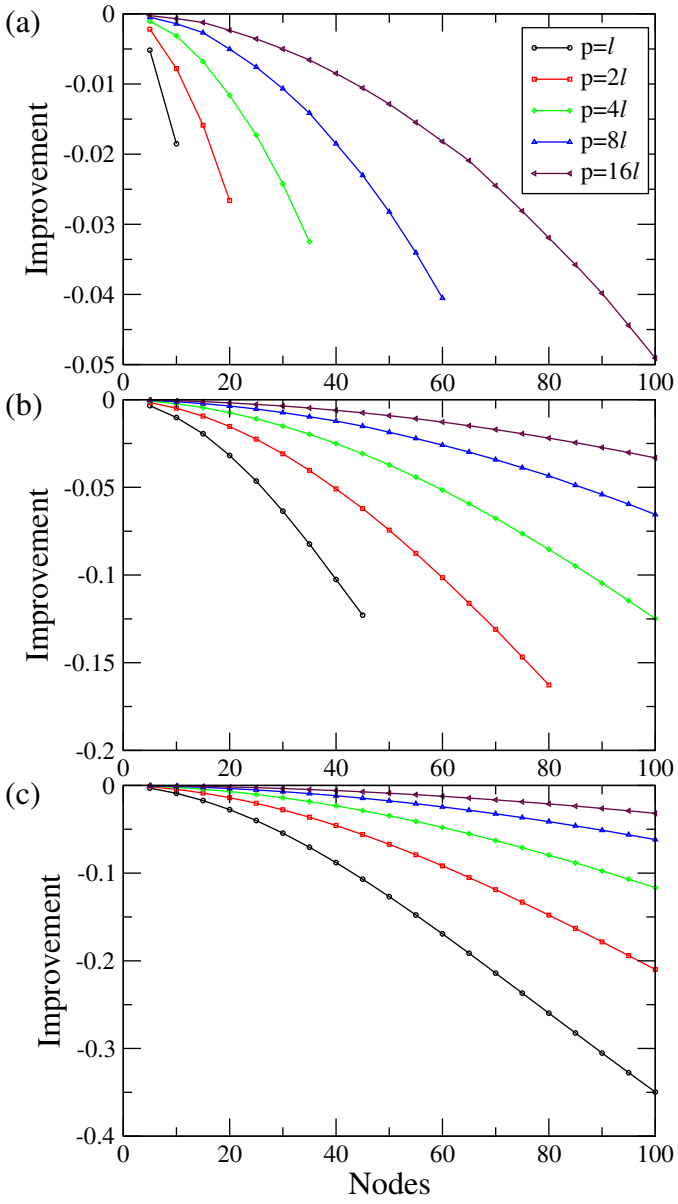

FIG. 6.6. *Execution time improvement offered by the 1-port scheme over the N-port setup, for a sequence of 1000 queries: (a) when each query is run individually, (b) when queries are run in batches of 10, and (c) when queries are run in batches of 100. Negative scores indicate a deterioration in performance, i. e. the 1-port setup is slower.*

[14] F. Ino, J. Gomita, Y. Kawasaki, and K. Hagihara, "A gpgpu approach for accelerating 2-d/3-d rigid registration of medical images," in *4th International Symposium on Parallel and Distributed Processing and Applications (ISPA 2006)*, ser. Lecture Notes in Computer Science 4330. Sorrento, Italy: Springer-Verlag, 2006, pp. 939 – 950.

[15] J. Montagnat, V. Breton, and I. Magni, "Medical image databases content-based queries partitioning on a grid," in *Proc. of HealthGrid 04, also available at http://www.i3s.unice.fr/~johan/publis/HealthGrid04.pdf*, 2004.

[16] B. Veeravalli, D. Ghose, V. Mani, , and T. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems.* Los Alamitos and California: IEEE Computer Society Press, 1996.

[17] G. D. Barlas, "Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of a divisible load on arbitrary processor trees," *IEEE Trans. on Parallel & Distributed Systems*, vol. 9, no. 5, pp. 429–441, May 1998.

[18] G. Barlas, "Optimizing image content-based query applications over high latency communication media," in *Proc. of 15th Euromicro Conf. on Parallel, Distributed and Network-Based Processing, PDIVM 2007*, Napoly, Italy, February 2007, pp. 341–348.

[19] Threaded I/O bench for Linux: http://tiobench.sourceforge.net/.

[20] V. Bharadwaj and G. D. Barlas, "Efficient strategies for processing multiple divisible loads on bus networks," *Journal of Parallel and Distributed Computation*, no. 62, pp. 132–151, 2002.

Edited by: Pasqua D'Ambra, Daniela di Serafino, Mario Rosario Guarracino, Francesca Perla
Received: June 2007
Accepted: November 2008