

Design of the Force Field Task Assignment Method and Associated Performance Evaluation for Desktop Grids

Edscott Wilson García^{1,*} and Guillermo Morales-Luna^{2,**}

¹ Instituto Mexicano del Petróleo
edscott@imp.mx

² Computer Science, CINVESTAV-IPN
gmoales@cs.cinvestav.mx

Abstract. In the case of desktop grids, a single hardware-determined latency and constant bandwidth between processors cannot be assumed without incurring in unnecessary error. The actual network topology is determined not only by the physical hardware, but also by the instantaneous bandwidth availability for parallel processes to communicate. In this paper we present a novel task assignment scheme which takes the dynamic network topology into consideration along with the traditionally evaluated variables such as processor availability and potential. The method performs increasingly better as the grid size increases.

Keywords: task assignment, load distribution, desktop grids.

1 Introduction

The complexity of contemporary scientific applications with increased demand for computing power and access to larger datasets is setting a trend towards the increased utilisation of grids of desktop personal computers [1]. In the quest for load balance, an important consideration to maximise the utilisation of a desktop grid is the question of optimal task assignment to processors in a parallel program.

As pointed out by Zhuge [2], in the future interconnection environment, resources will flow from high to low energy nodes. The basic laws and principles governing this field require more investigation. The work presented in this paper is a step in that direction.

A promising parallel computing model suggested by Valiant [3] is the *Bulk Synchronous Parallel Computer* (BSP) which emulates the von Neumann machine with respect to simplicity. This model is characterised by dividing the parallel program into a sequential run of *super-steps*. BSP might not be the most efficient model for each and every problem to solve, but several advantages are evident: scalability, predictability, and portability are the main promises of

* Supported by the graduate studies fund of the *Instituto Mexicano del Petróleo*.

** Partially supported by the *Consejo Nacional de Ciencia y Tecnología*.

the BSP model. A further characteristic, which is used in the force field task assignment method, is that upon barrier synchronisation the system reaches a known state. In the canonical BSP model, all processors are seen as equally costly to communicate with. For the force field task assignment, the BSP cost model must be extended with the considerations made by the LogP model.

The LogP model proposed by Culler *et al.* [4] is a generalisation of BSP, providing asynchronous communication, and accounting for latency and bandwidth costs of implementation. Using these cost features to extend the BSP model, but retaining barrier synchronisation, force field task assignment can be applied to the environment defined by a desktop grid.

The BSP model has great potential in its application towards desktop grids because the model provides for scalability. One of major features of desktop grids is precisely the ability to grow, and so scalability is an important issue. In 2001 Tiskin[5], developed a divide-and-conquer method under the BSP model and suggested that load balancing can be done in an efficient manner under the BSP model. With the bandwidth-aware BSP, not only this is possible, but also an error recovery algorithm can be implemented with very little extra computational overhead.

In order to harness the potential of networked computers, systems such as Globus [6] and Condor [7] distribute work tasks among a computer grid. These systems address in great detail issues such as job scheduling, access control and user/resource management. The mechanism provided by Condor [7] for sharing resources —by harnessing the idle-cycles on desktop machines—, enables high throughput using off-the-shelf components. A disadvantage arises from the organisation of the grid under the administration of a single central manager, statically configured. Details of job scheduling and migration are exclusive to the central manager. All tasks wait in a queue until the appropriate resource for execution can be located within the grid by the centralised server. Failure of this node brings down the entire Condor grid. Furthermore, there is no provision to share the workload of grid management: degraded system performance may arise with an overloaded server environment.

With respect to sharing resources amongst multiple Condor pools, recent work [8] has provided algorithms for automatising the discovery of remote Condor pools across administrative domains. In this work a p2p scheme based on a proximity-aware routing substrate [9] is used in conjunction with the Condor flocking facility. But within the Condor flock, as with most load distributing schemes, there is no mechanism for resolving the proximity-aware issue and take this factor into consideration while doing task assignment or migrations.

Empirical studies of desktop grids, such as the one done at the San Diego Supercomputing Center [10], provide the basis for elaborating theoretical evaluations for new load distribution schemes

The main contributions of the work in this paper are as follows:

- We describe an algorithm that uses proximity-aware along with cycle availability considerations for optimising task assignment on a desktop grid.

- We evaluate the proposed scheme by means of a simulator based on empirical characteristics of a desktop grid.

The rest of the paper is organised as follows. Section 2.1 gives an overview of the BSP model and task assignment problem on a desktop grid. Section 3 presents our proposed force-field scheme for task distribution on desktop grids, where the proximity-aware concept is developed. Section 4 presents an evaluation and analysis of the proposed scheme. Finally, section 5 provides concluding remarks.

2 Background

2.1 BSP Computational Model

Description. A *bulk synchronous parallel (BSP) computer* [11, 3] consists of a set of processor-memory pairs, a global communication network, and a mechanism for the efficient barrier synchronisation of the processors [12]. In the BSP model the parallel computation is divided into *super-steps*, each of which consists of a number of parallel-running threads that contain any number of operations. These threads perform only local communication until they reach a synchronisation barrier where all global communication takes place.

Task Assignment Overview. Consider the parallel application executing in the BSP machine composed of n parallel nodes, the application consists of m parallel tasks to be performed at super-step i . In order to assign task threads to processors at the beginning of a BSP super-step, a load balancing scheme should be used. Recent strategies [13] have been guided by the considerations of minimising communication costs [14, 15] and attaining load balance [16]. While this strategy will produce good results with small grids, the force field approach presented in this article will produce results which are closer to the minimum execution time as the grid size increases, as shown in section 4.

When dealing with the force field method, there are two considerations to take into account to determine the task assignment. The first —called the computational charge— is a combination of the memory cost for the task with the computational ability of the remote desktop computer. From the global desktop grid configuration, all nodes which lack the sufficient resources —mainly memory— to process the tasks will produce a net positive force, thus will repulsive and eliminated as candidates. The product of the amount of idle cycles per unit time that are available at the remote computer with the reciprocal of the cost in cycles which the memory utilisation of the task will determine the first main consideration for the force field method. This product is represented by the size of the circle in figure 1(a) and addressed in more detail in section 3.2. Note that the canonical *greedy idle-cycle* scheme will assign the first task to the machine with the most available cycles per unit time, which is *not necessarily* equivalent to largest circle in the figure since the memory cost is not taken into account in the same fashion.

The second main consideration is the cost in communications. This factor is dealt with an inverse-square formula. This takes both bandwidth availability and latency into consideration. In figure 1(a) the cost in communication is represented by the distance from the central assigning node to each of the remote computers —numbered 1 through 10. The canonical *greedy network-proximity* scheme will assign the first task to the machine with the least communication cost, *i.e.*, the closest circle.

The task assignment of the force field method is not evident from figure 1(a) because both the size and distance of the circles must be taken into account to determine the net effect. The novelty of the force field method is the way both the network-proximity and cycle-availability considerations are merged into a single force field to determine which node will receive the task. Figure 1(b) shows a three dimensional representation of the force field determined by the nodes in figure 1(a). The slope of the force field will determine in which direction the task will *roll* and reach a particular node, represented by a depression in the force field surface. The positive force indicates the node where the tasks are generated and distributed from.

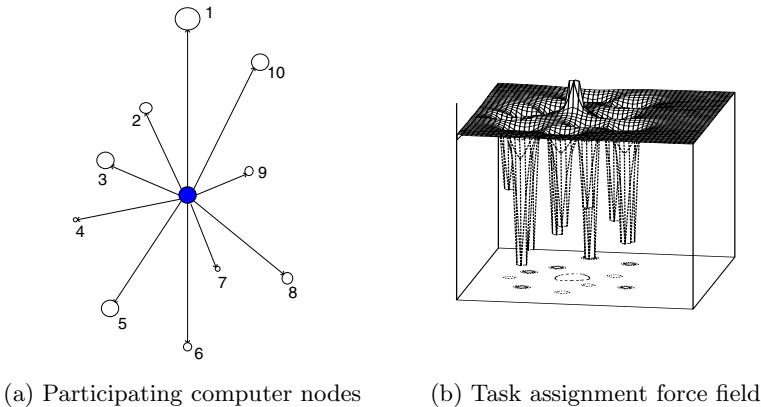


Fig. 1. Task assignment considerations

The force field method for synchronisation operates as follows. Upon reaching the synchronisation barrier, each node sends a completion message to all other processors participating in the super-step. Of these, one will currently be acting as a synchronisation master or sync-master (which will be the node in charge of distributing the tasks). Summing up, the sync-master will determine the force field and assign tasks to where the force field is strongest. To calculate the forces, the sync-master needs to know the available idle cycles and available memory at each node, besides the effective communication cost. The first two parameters are readily known and used for task assignment by systems such as Condor. For the third parameter evaluation, the BSP cost model [17] is extended with the full bandwidth and latency cost considerations of the LogP [4] model.

At the barrier all communications are completed and the system reaches a known state. The resulting force field values are ordered and nodes with the most negative values (attraction) receive the tasks. Positive force field values indicate a net repulsion for the assignment of new tasks, and are generated when the available memory of the node is inferior to the net memory requirements of the task.

The ubiquitous desktop computer suggests that desktop grids will continue growing. Our results indicate that the force field method yields even better results as the size of desktop grid increases.

Design Issues. On applying the BSP computational model there is an important aspect not to be overlooked. All tasks must synchronise at the BSP barrier and all communication takes place at this point. The cost of communication relative to the cost of computation is important. As processor speed increases at a rate faster than that of communication hardware, the ratio between the two will also increase. Parallel applications such as data distribution [13] already have a greater communication cost and can readily profit from the force field task assignment method. This is due to the increasingly better performance as the communication to computation cost ratio increases and the desktop grid size grows.

2.2 Desktop Grids

Network. While the opportunity to perform large computations at low-cost is a clear motivation for using desktop resources for parallel processing, the *uncertainty* of the resources which will be available at a given moment is the challenge in designing an effective task assignment strategy. The main points which characterise a desktop grid are the real-time network topology and the individual computational capacities of each desktop resource. The real-time network topology is determined in an instantaneous manner from the latency and available bandwidth characteristics of the desktop grid. While this topology is limited by the hardware with which the grid is built, the usage pattern by applications outside the BSP context will determine the effective topology available to the BSP machine and determines the *network cost* between any two points in the desktop grid.

Network proximity. The network proximity —or latency— is the amount of computation cycles taken to establish communication between any two nodes. This parameter is determined by the communications hardware.

Network capacity. The network capacity —or bandwidth— is the amount of information that can be transmitted between hosts once communication has been established. Since the network utilisation is not uniform nor static, the network capacity between any two nodes is not necessarily uniform nor constant through time.

3 Design

An important consideration with regard to desktop grids is that we can assume that the number of parallel tasks to be completed during a super-step is less than or equal to the number of available processors in the grid. We can make this assumption because increasing the grid size with off-the-shelf equipment is relatively easy and cheap.

In the design of an improved task assignment scheme under the bandwidth-aware BSP computational model, we borrow from the concept of force field to elaborate our proposed algorithm. On doing so, we part from an important fact: during the last years, computational speed has been increasing geometrically, while the speed in communication has only experienced a linear increase. This indicates that the trends in future parallel computing will be defined by communications. The question is not whether this will happen, but rather *when* the turning point will be. This is besides the fact that many data intensive parallel computations are already governed by the communications.

Consider that the costs in computational cycles for each task in a BSP super-step need not be equal. If C_i is the cost in cycles to solve a particular task i , and ξ_j is the amount of available cycles per unit time at node j , then the time to solve the task i can be written as C_i/ξ_j .

Without considering any communications cost when dealing with a computational grid of size m , the optimum for n parallel tasks is the minimum of the combinatorial set obtained by associating tasks to processors. If the cost in communication is considered, the complexity of the problem increases dramatically. The set of all super-steps in a BSP problem then becomes similar to a DAG determination, which is an NP-hard problem.

3.1 Fundamental Laws of Motion

A successful load balancing scheme using gravitational forces can be found in Hui and Chanson [18], although in this case the gravitational forces are masked behind the theory of hydrodynamics. This effective method of load balancing has found application with the PRAM parallel machine model. A computational model which is less dependent on system architecture, such as BSP, calls for a simpler expression of the laws governing task assignment.

A drawback to the hydrodynamic load balancing's use of Newton's Law of Universal Gravitation is the absence of repulsive forces. The hydrodynamic fluid cannot flow from a lower to a higher level. Only with repulsive forces may any particular node reject tasks where requirements surpass the node's resources.

On the other hand, the BSP model allows for direct application of Newton's laws. Nonetheless, gravitational forces are only attractive. Therefore, Coulomb's law is more appealing for this work.

How does this fit into the BSP model? At the synchronisation barrier the system reaches a well defined state upon which attractive and repulsive forces associated to emerging tasks can be determined. Network workstations participating in the BSP machine which temporarily are removed from the grid need

only to switch the sign on their computational capacity, with which all new tasks are repelled and sent to other available nodes on the desktop grid.

3.2 The Force Field Load Distribution

The Computational Distance. To find the physical equivalent for distance in the task assignment problem, we must calculate the cost in communications. Consider the case of performing task assignment on a BSP parallel computer composed of a network of workstations using shared communications resources. Available bandwidth between any two nodes will vary with time. What is important is not the instantaneous value at any particular moment but rather the average value during the particular time interval where the instantaneous value is located.

The Computational Charge. When dealing with the computational charges the force field method refers to two charges. The first lies within the remote computer and the second within the task to be assigned.

Remote computer charge. Represented by q_j , is the computational potential of node j , and is equivalent to the available computational cycles per unit time. This number is always positive or equal to zero. If node j sets $q_j = 0$, then neither attraction nor repulsion will exist. Whether or not the node actually receives a task depends on the force field values determined for other nodes on the grid.

Task charge. Represented by q_i , is a computational ease associated to the memory requirements of the task. Computational ease—inversely proportional to cost—is an important parameter which is often overlooked: the current bottleneck in the execution of most tasks is not in processor speed but rather the moving of data between memory storage and processor registers.

In other words, if the available memory on node j is less than that required for task i , then q_i is assigned a positive sign, ensuring a repulsive force. Otherwise the sign will be negative. If c_i is the cost in computation cycles entailed by the memory requirements of task i , then $q_i = \frac{1}{c_i}$ is the charge associated to the task¹. Note that the memory requirements of every task has to be known to any assignment scheme, otherwise the determination of whether the remote node has the potential resources to deal with the job would be impossible. The cost in cycles that will be required for the task is not used at any time. In our simulator we use the Large Numbers Law to obtain values for this parameter and randomly assign these values to tasks.

¹ A further distinction may be made whether the available virtual memory is fully in RAM or partly distributed in disk swap. This further refinement is not considered in the results presented in this paper.

4 Performance Evaluation

For each run the absolute minimum execution time—which considers all possible combinations of task/processor assignments—is also obtained for comparison with the task assignment algorithms being tested.

In the tests conducted, mean values were obtained from 1000 simulations for each data point. Thus, figures 2(a)–(c) show the results for an increasing amount of tasks in the parallel set to be completed, and where the grid size is equal to 1000 desktop computers. Computational availability, communications cost, task memory requirements and task computational costs are all determined by Gaussian distributions and randomly assigned to different identifiers.

The purpose of the evaluation is to determine which strategy is *better* for task assignment on a desktop grid. Every process can be qualified with the following considerations:

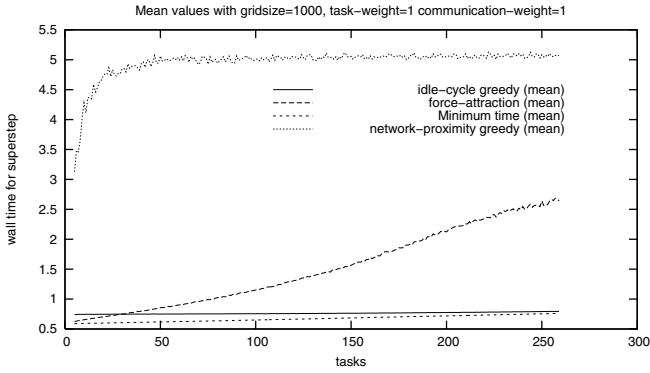
- The amount of memory required for data storage.
- The amount of computation cycles that are required to complete all programmed operations.
- The time units consumed by the communication requirements.

Strictly speaking, the second point will depend on the hardware and can be associated among different platforms by linear proportionality. On the other hand, computation cycles can be dealt with in an abstract manner—architecture independent—such as Papadimitriou and Yannakakis [19] do in the presentation of the directed acyclic graphs model. In this paper the same approach is used.

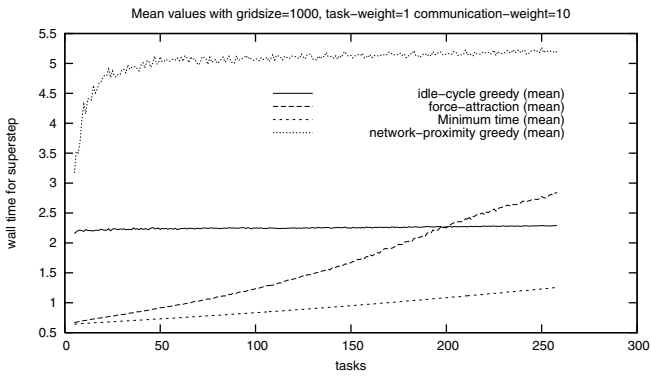
With these qualities the merits of any particular task assignment scheme may be evaluated. The first point—memory requirements—is necessary data to determine if the remote computer has the necessary resources to complete the job. The second point—the cycles required by the task and the cycle available per unit time at the remote node—allows the execution wall clock time to be obtained. Nonetheless, the knowledge of the amount of cycles each task will require is not generally known, except for the most elemental applications. Besides, in the tabulation of a parallel job, the time associated to communication costs must also be considered.

For the optimum calculation, it is necessary to know the amount of cycles that each task will require. This will be used to evaluate the performance of the different algorithms compared (these algorithms, of course, may not use this information). An algorithm will be *better* than another if the times obtained are closer to the optimum values.

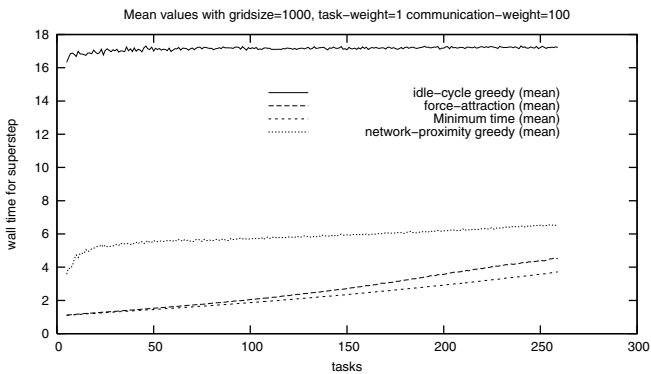
Figure 2(a) shows the results when the cost in communication is low with respect to the cost in computation. In this graph the greedy computation cost quickly becomes asymptotic with the minimum attainable value when the number of simultaneous tasks in the BSP super-step reaches 25% of the configured processors in the desktop grid. For super-steps with simultaneous tasks occupying less than 2.5% of the configured processors, the force field task assignment



(a) Low communication cost



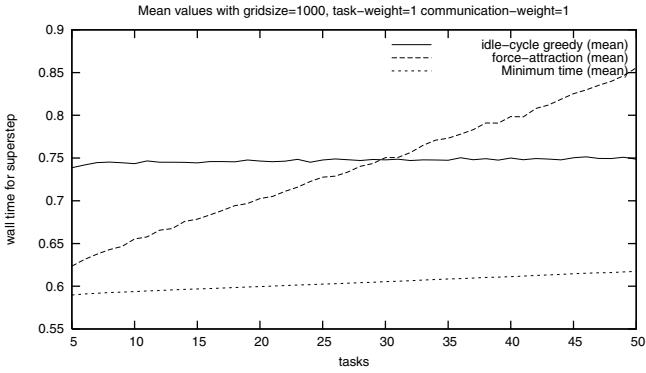
(b) Normal communication cost



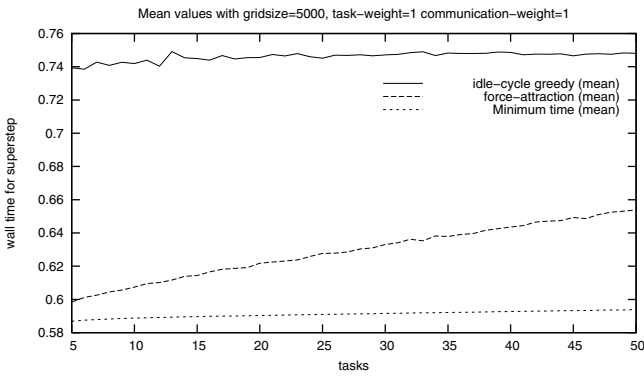
(c) High communication cost

Fig. 2. Mean values for different communication costs

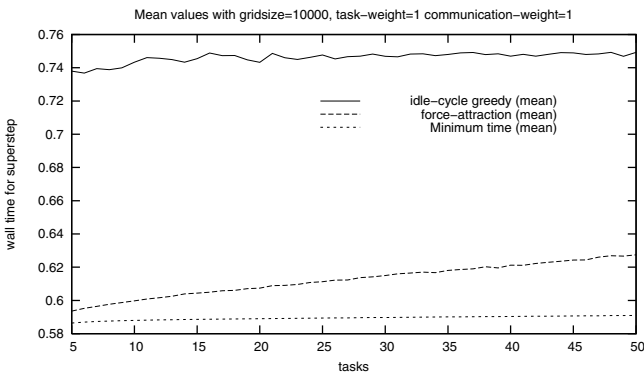
out performed the greedy computation cost scheme. When the grid size starts to increase, as seen *ut infra*, the situation changes dramatically in favour of the force field task assignment scheme.



(a) Low communication cost (detail)



(b) Low communication cost (detail 5X grid)



(c) Low communication cost (detail 10X grid)

Fig. 3. Mean values for low communication cost and growing grid size

In figure 2(b), are the results for a parallel program where a computation cost still exceeds the communication cost. Many parallel applications fall into this category. In this case, we can observe that the force field assignment scheme

exceeds the greedy computation cost scheme well up to the utilisation of 20% of the configured desktop grid for the simultaneous execution of a single BSP super-step.

In the third graph of the simulation results, figure 2(c), communication costs exceed computation costs. Distributed data applications fall into this category, as well as an increasing number of other parallel applications as computing speed increases geometrically while communication speed does so linearly. In this case the force field approach far exceeds both the greedy communication cost and greedy computation cost and remains quite close to the theoretical minimum performance that can be obtained.

In the foreseeable future the size of available desktop grids is bound to increase, and for this reason it is important to analyse how the force field task assignment will perform in relation to the greedy schemes as the grid grows in size. In figures 3(a)–(c) we can observe that, as the number of computers participating in the grid configuration increases, the force field task assignment outperforms the greedy schemes. As our results demonstrate, this is even with tasks involving a low communications cost. This points to the force field as an important option to consider when large desktop grids are to be used for parallel computing.

5 Conclusions

Although the physical motivation behind the force field task assignment algorithm may not be a consequence of formal computer science, in practise the analogies made with the laws of physics converge to a good strategy for task assignment in the case of a dynamic network bandwidth topology and growing grid size configurations.

The effect that communications has on the time to complete a BSP super-step is a factor which should not be disregarded when dealing with desktop grids of growing size. The effect of communications in performing parallel computation will have an increasingly greater effect on parallel programs as growth in computation speed out paces growth in communication speed.

Task assignment strategies which take both factors into consideration, such as the force field algorithm, should be preferred when there is reason to assume the performance shall be better than the respective greedy strategies.

In the development of this algorithm we have considered the force field as a scalar field. Notwithstanding, the analysis as a vector field could produce better results allowing for a more homogeneous load balancing of the entire desktop grid. More work remains to be done in this direction.

References

1. Foster(Ed.), I., (Ed.), C.K.: The GRID: Blueprint for a New Computing Infrastructure. Morgan Kauffmann Publishers (1999)
2. Zhuge, H.: The future interconnection environment. *IEEE Computer* 4 (2005) 27–33

3. Valiant, L.G.: A bridging model for parallel computation. *Communications of the ACM* **8** (1990) 103–111
4. Culler, D., Karp, R., Patterson, D., Sahay, A., Schauer, J., Santos, E., Subramanian, R., von Eicken, T.: Logp: Towards a realistic model of parallel computation. In: *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. (1993) 1–12
5. Tiskin, A.: A new way to divide and conquer. *Parallel Processing Letters* **4** (2001)
6. Foster, I., Kesselman, C.: Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputing Applications and High Performance Computing* **2** (1997) 115–128
7. Litzkow, M., Livny, M., Mutka, M.: Condor - a hunter of idle workstations. In: *Proceedings 8th International Conference on Distributed Computing Systems (ICDCS 1988)*. (1988) 104–111
8. Butt, A.R., Zhang, R., Hu, Y.C.: A self-organizing flock of condors. In: *Proceedings Super Computing 2003, ACM* (2003) 15–21
9. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Exploiting network proximity in peer-to-peer overlay networks. Technical report, Microsoft Research (2002) Technical Report MSR.TR-2002-82.
10. Kondo, D., Taufer, M., Brooks, C.L., Casanova, H., Chien, A.A.: Characterizing and evaluating desktop grids: An empirical study. Technical report, San Diego Supercomputer Center and University of California, San Diego (2004) Work supported by the National Science Foundation under Grant ACI-0305390.
11. Gibbons, A.M., Spirakis, P., eds. In: *General Purpose Parallel Computing*. Cambridge University Press (1993) 337–391
12. van Leeuwen, J., ed. In: *Scalable Computing*. Springer-Verlag (1995) 46–61
13. Sujithan, K.R.: Towards a scalable parallel object database - the bulk synchronous parallel approach. Technical report, Wadham College Oxford (1996) Technical Report PRG-TR-17-96.
14. Adler, M., Byers, J.W., Karp, R.M.: Scheduling parallel communication: The h-relation problem. Technical report, International Computer Science Institute, Berkeley (1995) Technical Report TR-95-032.
15. Goodrich, M.T.: Communication-efficient parallel sorting. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ACM (1996) 247–256
16. Shi, H., Schaeffer, J.: Parallel sorting by regular sampling. *Journal of Parallel and Distributed Computing* **4** (1992) 361–372
17. Baumker, A., Dittrich, W., Heide, F.M.: Truly efficient parallel algorithms: 1-optimal multisearch for an extension of the bsp model. *Theoretical Computer Science* (1998)
18. Hui, C.C., Chanson, S.T.: Hydrodynamic load balancing. *IEEE Trans. Parallel and Distributed Systems* **10** (1999)
19. Papadimitriou, C., Yannakakis, M.: Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput.* **19** (1990) 322–328