3-30-2012

The Effect of Communication Time Delays in Parallel Computations

Chaouki T. Abdallah

J. D. Birdwell

J. Chiasson

Z. Tang

N. Alluri

See next page for additional authors

Authors
Chaouki T. Abdallah, J. D. Birdwell, J. Chiasson, Z. Tang, N. Alluri, and Tsewei Wang

The Effect of Communication Time Delays in Parallel Computations

C. T. Abdallah
ECE Dept, University of New Mexico
Alburquerque NM 87131-1356, USA
chaouki@eece.unm.edu

J. D. Birdwell, J. Chiasson*
Z. Tang, N. Alluri
ECE Dept, University of Tennessee
Knoxville TN 37996, USA
birdwell@utk.edu, chiasson@utk.edu
tang@hickory.engr.utk.edu, nalluri@utk.edu

Tsewei Wang
ChE Dept, University of Tennessee
Knoxville TN 37996, USA, twang@utk.edu

July 5, 2002

1 Introduction

Parallel computer architectures utilize a set of computational elements (CE) to achieve performance that is not attainable on a single processor, or CE, computer. A common architecture is the cluster of otherwise independent computers communicating through a shared network. To make use of parallel computing resources, problems must be broken down into smaller units that can be solved individually by each CE while exchanging information with CEs solving other problems.

The Federal Bureau of Investigation (FBI) National DNA Indexing System (NDIS) and Combined DNA Indexing System (CODIS) software are candidates for parallelization. New methods developed by Wang et al. [4][5][6][12][13] lead naturally to a parallel decomposition of the DNA database search problem while providing orders of magnitude improvements in performance over the current release of the CODIS software. The projected growth of the NDIS database and

*Corresponding author

in the demand for searches of the database necessitates migration to a parallel computing platform.

Effective utilization of a parallel computer architecture requires the computational load to be distributed more or less evenly over the available CEs. The qualifier "more or less" is used because the communications required to distribute the load consume both computational resources and network bandwidth. A point of diminishing returns exists.

Distribution of computational load across available resources is referred to as the *load balancing* problem in the literature. Various taxonomies of load balancing algorithms exist. Direct methods examine the global distribution of computational load and assign portions of the workload to resources before processing begins. Iterative methods examine the progress of the computation and the expected utilization of resources, and adjust the workload assignments periodically as computation progresses. Assignment may be either deterministic, as with the dimension exchange/diffusion [7] and gradient methods, stochastic, or optimization based. A comparison of several deterministic methods is provided by Willeback-LeMain and Reeves [14].

To adequately model load balancing problems, several features of the parallel computation environment should be captured (1) The workload awaiting processing at each CE; (2) the relative performances of the CEs; (3) the computational requirements of each workload component; (4) the delays and bandwidth constraints of CEs and network components involved in the exchange of workloads, and (5) the delays imposed by CEs and the network on the exchange of measurements. A queuing theory [10] approach is well-suited to the modeling requirements and has been used in the literature by Spies [11] and others. However, whereas Spies assumes a homogeneous network of CEs and models the queues in detail, the present work generalizes queue length to an expected waiting time, normalizing to account for differences among CEs, and aggregates the behavior of each queue using a continuous state model. The present work focuses upon the effects of delays in the exchange of information among CEs, and the constraints these effects impose on the design of a load balancing strategy. Preliminary results by the authors appear in [2]. However, new nonlinear models are developed here to obtain better fidelity and experimental results are presented and compared to that given by the models.

Section 2 presents our approach to modeling the computer network and load balancing algorithms to incorporate the presence of delay in communicating between nodes and transferring tasks. Section 3 contains an analysis of the stability properties of the linear models, while Section 4 presents simulations of the linear and nonlinear models. Section 5 presents experimental data from an actual implementation of a load balancing algorithm. Finally, Section 6 is a summary and conclusion of the present work and a discussion of future work.

2 Models of Load Balancing Algorithms

In previous work [2][1][3], the authors have developed both linear and nonlinear time delay models for load balancing. To introduce the basic approach to load balancing, consider a computing network consisting of n computers (nodes) all of which can communicate with each other. At start up, the computers are assigned an equal number of tasks. However, when a node executes a particular task it can in turn generate more tasks so that very quickly the loads on various nodes become unequal. To balance the loads, each computer in the network sends its queue size $q_j(t)$ to all other computers in the network. A node i receives this information from node j *delayed* by a finite amount of time τ_{ij}, that is, it receives $q_j(t-\tau_{ij})$. Each node i then uses this information to compute its local estimate[1] of the average number of tasks in the queues of the n computers in the network. In this work, the simple estimator $\left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n$ $(\tau_{ii}=0)$ which is based on the most recent observations is used. Node i then compares its queue size $q_i(t)$ with its estimate of the network average as $\left(q_i(t) - \left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n\right)$ and, if this is greater than zero, the node sends some of its tasks to the other nodes while if it is less than zero, no tasks are sent. Further, the tasks sent by node i are received by node j with a delay h_{ij}. The controller (load balancing algorithm) decides how often and fast to do load balancing (transfer tasks among the nodes) and how many tasks are to be sent to each node.

As just explained, each node controller (load balancing algorithm) has only *delayed* values of the queue lengths of the other nodes, and each transfer of data from one node to another is received only after a finite time delay. An important issue considered here is to study the effect of these delays on system performance. Specifically, the continuous time models developed here represent our effort to capture the effect of the delays in load balancing techniques and were developed so that system theoretic methods could be used to analyze them.

2.1 Basic Model

The basic mathematical model of a given computing node for load balancing is given by [1][3]

[1] It is an estimate because at any time, each node only has the delayed value of the number of tasks in the other nodes.

$$
\begin{aligned}
\frac{dx_i(t)}{dt} &= \lambda_i - \mu_i + u_i(t) - \sum_{j=1}^{n} p_{ij}\frac{t_{p_i}}{t_{p_j}}u_j(t-h_{ij}) \\
y_i(t) &= x_i(t) - \frac{\sum_{j=1}^{n} x_j(t-\tau_{ij})}{n} \\
u_i(t) &= -K_i \text{sat}\left(y_i(t)\right) \\
p_{ij} &\geqslant 0, p_{jj} = 0, \sum_{i=1}^{n} p_{ij} = 1
\end{aligned} \tag{1}
$$

where

$$
\begin{aligned}
\text{sat}\,(y) &= y \text{ if } y \geqslant 0 \\
&= 0 \text{ if } y < 0.
\end{aligned}
$$

In this model we have

- n is the number of nodes.
- $x_i(t)$ is the *expected waiting time* experienced by a task inserted into the queue of the i^{th} node. With $q_i(t)$ the number of *tasks* in the i^{th} node and t_{p_i} the average time needed to process a task on the i^{th} node, the expected (average) waiting time is then given by $x_i(t) = q_i(t)t_{p_i}$. Note that $x_j/t_{p_j} = q_j$ is the number of tasks in the node 1 queue. If these tasks were transferred to node i, then the waiting time transferred is $q_j t_{p_i} = x_j t_{p_i}/t_{p_j}$, so that the fraction t_{p_i}/t_{p_j} converts waiting time on node j to waiting time on node i.
- λ_i is the rate of generation of waiting time on the i^{th} node caused by the addition of tasks (rate of increase in x_i)
- μ_i is the rate of reduction in waiting time caused by the service of tasks at the i^{th} node and is given by $\mu_i \equiv (1 \times t_{p_i})/t_{p_i} = 1$ for all i.
- $u_i(t)$ is the rate of removal (transfer) of the tasks from node i at time t by the load balancing algorithm at node i. Note that $u_i(t) \leq 0$.
- $p_{ij}u_j(t)$ is the rate that node j sends waiting time (tasks) to node i at time t where $p_{ij} \geqslant 0, \sum_{i=1}^{n} p_{ij} = 1$ and $p_{jj} = 0$. That is, the transfer from node j of expected waiting time (tasks) $\int_{t_1}^{t_2} u_j(t)dt$ in the interval of time $[t_1, t_2]$ to the other nodes is carried out with the i^{th} node being sent the fraction $p_{ij}\frac{t_{p_i}}{t_{p_j}}\int_{t_1}^{t_2} u_j(t)dt$ where the fraction t_{p_i}/t_{p_j} converts the task from waiting time on node j to waiting time on node i. As $\sum_{i=1}^{n}\left(p_{ij}\int_{t_1}^{t_2} u_j(t)dt\right) =$

$\int_{t_1}^{t_2} u_j(t)dt$, this results in a removing *all* the waiting time $\int_{t_1}^{t_2} u_j(t)dt$ from node j.

- The quantity $-p_{ij}u_j(t-h_{ij})$ is the rate of increase (rate of transfer) of the expected waiting time (tasks) at time t from node j by (to) node i where h_{ij} ($h_{ii} = 0$) is the time delay for the task transfer from node j to node i.
- The quantities τ_{ij} ($\tau_{ii} = 0$) denote the time delay for communicating the expected waiting time x_j from node j to node i.
- The quantity $x_i^{avg} = \left(\sum_{j=1}^{n} x_j(t-\tau_{ij})\right)/n$ is the estimate[2] by the i^{th} node of the average waiting time of the network and is referred to as the *local average* (local estimate of the average).

In this model, all rates are in units of the *rate of change of expected waiting time*, or *time/time* which is dimensionless). As $u_i(t) \leq 0$, node i can only send tasks to other nodes and cannot initiate transfers from another node to itself. A delay is experienced by transmitted tasks before they are received at the other node. The control law $u_i(t) = -K_i \mathrm{sat}(y_i(t))$ states that if the i^{th} node output $x_i(t)$ is above the local average $\left(\sum_{j=1}^{n} x_j(t-\tau_{ij})\right)/n$, then it sends data to the other nodes, while if it is less than the local average nothing is sent. The j^{th} node receives the fraction $\int_{t_1}^{t_2} p_{ji}u_i(t)dt$ of transferred waiting time $\int_{t_1}^{t_2} u_i(t)dt$ delayed by the time h_{ij}.

2.2 Linear Model

Model (1) is the basic model but one important detail remains unspecified, namely the exact form p_{ji} for each sending node i. One approach is to choose them as constant and equal

$$
\begin{aligned}
p_{ji} &= 1/(n-1) \text{ for } j \neq i \\
p_{ii} &= 0
\end{aligned}
$$

where it is clear that $p_{ji} \geqslant 0, \sum_{j=1}^{n} p_{ji} = 1$. If this were done, and the saturation functions removed, the following *linear time invariant* model results

$$
\begin{aligned}
\frac{dx_i(t)}{dt} &= \lambda_i - \mu_i + u_i(t) - \sum_{j \neq i} p u_j(t-h_{ij}) \\
y_i(t) &= x_i(t) - \frac{\sum_{j=1}^{n} x_j(t-\tau_{ij})}{n} \\
u_i(t) &= -K_i y_i(t) \\
p &= \frac{1}{n-1}
\end{aligned}
\tag{2}
$$

[2] This is an only an estimate due to the delays.

When $u_i(t) = -K_i y_i(t) < 0$, this operates as in (1) in that the tasks are immediately removed and sent to the other nodes where each of those nodes experiences a delay (h_{ij}) in getting these tasks. However, a fundamental problem with this linear model is that when $y_i(t) < 0$ the controller (load balancing algorithm) $u_i(t) = -K_i y_i(t) > 0$ so that the node is *instantaneously* taking on waiting time (tasks) from the other nodes before those tasks are removed from the other nodes' queues. That is, it is accepting the waiting times (tasks) $pu_j(t)$ from each of the other nodes. There is a finite time delay associated with this transfer of tasks, and this model ignores this fact. In spite of this fact, it is still of value to consider the system (2) because it can be completely analyzed with regards to stability, and it does capture the oscillatory behavior of the $y_i(t)$.

2.3 Nonlinear Model with Non Constant p_{ij}

The model (1) did not have the p_{ij} specified explicitly. For example, they can be considered constant as in the linear model. However, it is actually useful to use the local information of the waiting times $x_i(t), i = 1, .., n$ to set their values. Recall that p_{ij} is the fraction of $u_j(t)$ that node j allocates (transfers) to node i at time t, and conservation of the tasks requires $p_{ij} \geqslant 0, \sum_{i=1}^{n} p_{ij} = 1$ and $p_{jj} = 0$. The quantity $x_i(t - \tau_{ji}) - x_j^{avg}$ represents what node j estimates[3] the waiting time of node i is with respect to the local average of node j. If node i queue is above the local average, then node j does not send tasks to it. Therefore $\mathrm{sat}\big(x_j^{avg} - x_i(t - \tau_{ji})\big)$ is an appropriate measure by node j as to how much node i is *below* the local average. Node j then repeats this computation for all the other nodes and then portions out its tasks among the other nodes according to the amounts they are below the local average, that is,

$$p_{ij} = \frac{\mathrm{sat}\left(x_j^{avg} - x_i(t - \tau_{ji})\right)}{\displaystyle\sum_{i \ni i \neq j} \mathrm{sat}\left(x_j^{avg} - x_i(t - \tau_{ji})\right)}. \tag{3}$$

The p_{ij} are defined to be zero if the denominator $\displaystyle\sum_{i \ni i \neq j} \mathrm{sat}\big(x_i(t - \tau_{ji}) - x_j^{avg}\big) = 0$.

With the definition of the p_{ij} given by (3), a load balancing algorithm which portions out the tasks in proportion to the amounts they are below the local average, is given by the following nonlinear differential-delay system

[3] Again, the term "estimates" is used because node j does not know the current value of $x_i(t)$, but only its earlier value $x_i(t - \tau_{ij})$.

$$
\begin{aligned}
\frac{dx_i(t)}{dt} &= \lambda_i - \mu_i + u_i(t) - \sum_{j \neq i} p_{ij} u_j(t - h_{ij}) \\
x_i^{avg} &= \frac{\sum_{j=1}^{n} x_j(t - \tau_{ij})}{n} \\
y_i(t) &= x_i(t) - x_i^{avg} \\
u_i(t) &= -K_i \text{sat}\left(y_i(t)\right)
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
p_{ij} &= \frac{\text{sat}\left(x_j^{avg} - x_i(t - \tau_{ji})\right)}{\displaystyle\sum_{i \ni i \neq j} \text{sat}\left(x_j^{avg} - x_i(t - \tau_{ji})\right)} \text{ for } i \neq j \\
&= 0 \text{ for } i = j
\end{aligned}
$$

3 Stability Analysis of the Linear Model

In previous work, [2][1] a stability analysis was carried out for the linear system. There it was shown that a three node model with $K_1 = K_2 = K_3 = K$, $p = 1/2$, $\tau_{ij} = \tau, h_{ij} = 2\tau$ for $i \neq j$ for all $i, j = 1, 2, 3$ ($\tau_{ii} = h_{ii} = 0$) was stable if and only if $K < \frac{5\pi}{4\tau \sin(\pi/3)}$.

4 Simulations

Experimental procedures to determine the delay values are given in [8] and summarized in [9]. These give representative values for a Fast Ethernet network with three nodes of $\tau_{ij} = \tau = 200$ μsec for $i \neq j, \tau_{ii} = 0$, and $h_{ij} = 2\tau = 400$ μsec for $i \neq j, h_{ii} = 0$. The initial conditions were $x_1(0) = 0.6$, $x_2(0) = 0.4$ and $x_3(0) = 0.2$. The inputs were set as $\lambda_1 = 3\mu_1, \lambda_2 = 0, \lambda_3 = 0, \mu_1 = \mu_2 = \mu_3 = 1$. The t_{p_i}'s were taken to be equal.

4.1 Linear Simulations

The figures below are plots of $y_1(t), y_2(t), y_3(t)$ using the linear simulation. Three sets of runs are shown. To compare with the experimental results given in Figure 4, Figure 1 shows the output responses of the *linear model* with the gains set as $K_1 = 6667, K_2 = 4167, K_3 = 5000$, respectively. In each of the plots, the effect of delay ($\tau = 200\mu$sec) coming into play at $t = 200\mu$sec is evident.

4.2 Nonlinear Simulations with constant p_{ij}

In this set of simulations, the model (1) is used. To compare with the experimental results given in Figure 4, Figure 2 shows the output responses with the gains set as $K_1 = 6667, K_2 = 4167, K_3 = 5000$, respectively.

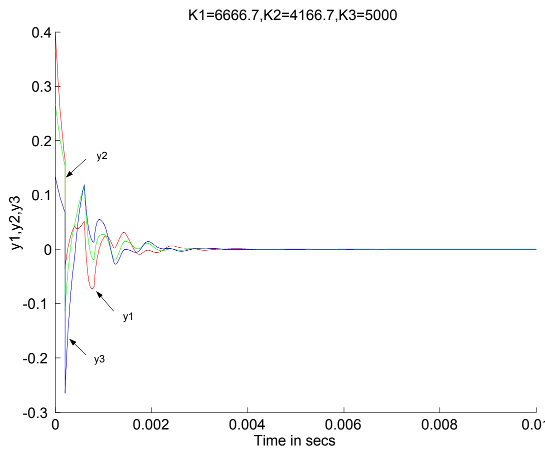

Figure 1: Linear simulation with $K1 = 6666.7; K2 = 4166.7; K3 = 5000$

4.3 Nonlinear Simulations

In this set, the model (4) is used. It is seen that the responses are faster for the $K = 1000$ case compared to the constant p_{ij} case. However, for $K = 5000$, the response is actually deteriorated compared to the constant p_{ij} case.

5 Experimental Results

Preliminary experimental work has been preformed on a computer network built at the University of Tennessee. To explain the connection between the control gain K and the actual implementation, recall that the waiting time is related to the number of tasks as $x_i(t) = q_i(t)t_{p_i}$ where t_{p_i} is the average time to carry out a task. The continuous time control law is

$$u(t) = -K\text{sat}\,(y_i(t))$$

where $u(t)$ is the rate of decrease of waiting time $x_i(t)$ per unit time. Consequently, the gain K represents the rate of reduction of waiting time per second in the continuous time model. Also, $y_i(t) = \left(q_i(t) - \left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n\right) t_{p_i} = r_i(t)t_{p_i}$ where $r_i(t)$ is simply the number of tasks above the estimated (local) average number of tasks and, as the interest here is the case $y_i(t) > 0$, consider $u(t) = -Ky_i(t)$. With Δt the time interval between successive executions of the load balancing algorithm, the control law says that a fraction of the queue $K_z r_i(t)$ $(0 < K_z < 1)$ is removed in the time Δt so the rate of reduction of *waiting time* is $-K_z r_i(t)t_{p_i}/\Delta t = -K_z y_i(t)/\Delta t$ so that

$$u(t) = -\frac{K_z y_i(t)}{\Delta t} \Longrightarrow K = \frac{K_z}{\Delta t}. \tag{5}$$

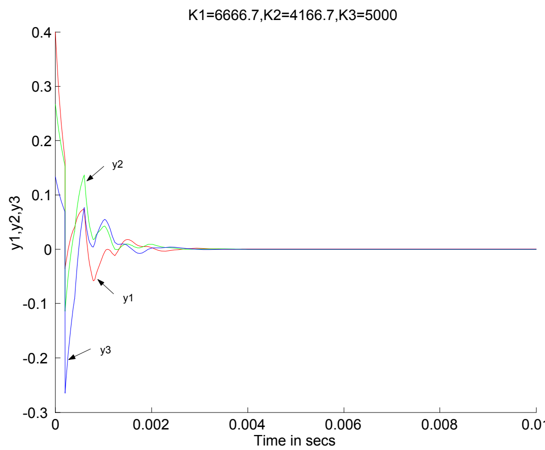

Figure 2: Nonlinear simulation with constant p_{ij} and $K1 = 6666.7; K2 = 4166.7; K3 = 5000$

This shows that the gain K is related to the actual implementation by how fast the load balancing can be carried out and how much (fraction) of the load is transferred. In the experimental work reported here, Δt actually varies each time the load is balanced. As a consequence, the value of Δt used in (5) is an average value for that run. The average time t_{p_i} to process a task is the same on all nodes (identical processors) and is equal $10\mu\sec$ while the time it takes to transfer of load is about $50\mu\sec$. The initial conditions were taken as $q_1(0) = 60000, q_2(0) = 40000, q_3(0) = 20000$ (corresponding to $x_1(0) = q_1(0)t_{pi} = 0.6, x_2(0) = 0.4, x_3(0) = 0.2$). All of the experimental responses were carried out with constant $p_{ij} = 1/2$ for $i \neq j$.

Figure 4 is a plot of the responses $r_i(t) = q_i(t) - \left(\sum_{j=1}^{n} q_j(t - \tau_{ij})\right)/n$ for $i = 1, 2, 3$ (recall that $y_i(t) = r_i(t)t_{p_i}$). The (average) value of the gains were $(K_z = 0.5)$ $K_1 = 0.5/75\mu\sec = 6667, K_2 = 0.5/120\mu\sec = 4167, K_3 = 0.5/100\mu\sec = 5000$. This figure compares favorably with Figures 1 (linear model) and 2 (nonlinear model) except for the time scale being off, that is, the experimental responses are slower. The explanation for this it that the gains here vary during the run because Δt (the time interval between successive executions of the load balancing algorithm) varies during the run. Further, this time Δt is *not* modeled in the continuous time simulations, only its average effect in the gains K_i. That is, the continuous time model does not stop processing jobs (at the average rate t_{p_i}) while it is transferring tasks to do the load balancing. An important point is that the actual delays experienced by the network traffic in the parallel machine are *random*. Work has been performed to characterize the bandwidth and delay on unloaded and loaded network switches, in order to identify the delay parameters of the analytic models and is reported in [8][9].

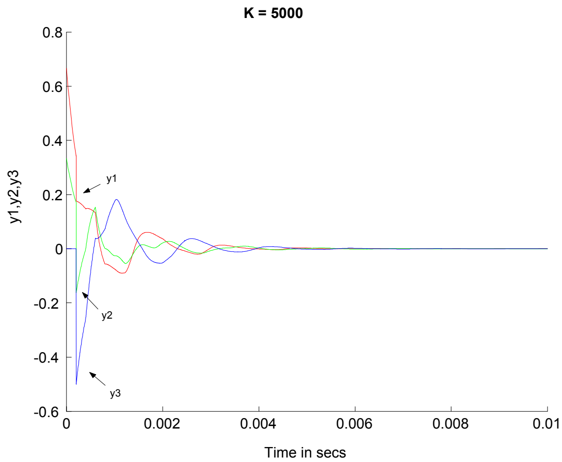

Figure 3: Nonlinear output responses with $K = 5000$.

The value $\tau = 200\ \mu$sec used for simulations represents an average value for the delay and was found using the procedure described in [9]. The interest here is to compare the experimental data with that from the three models previously developed.

6 Summary and Conclusions

By the time of the conference, the authors expect to have experimental results for the nonlinear model with non constant p_{ij}. The decision to use constant or non constant p_{ij}'s may depend on the network size. Preliminary work with only three nodes indicates that the constant p_{ij}'s seem to outperform the non constant implementation. Further investigations will be done to characterize which parameters in the model are the "sensitive" parameters, that is, which parameters when varied slightly result in significant changes in the response. Another consideration is the fact that the load balancing operation involves processor time which is not being used to process tasks. Consequently, there is a trade-off between using processor time/network bandwidth and the advantage of distributing the load evenly between the nodes to reduce overall processing time. This is now under investigation.

7 Acknowledgements

The work of J.D. Birdwell, V. Chupryna, Z. Tang, and T.W. Wang was supported by U.S. Department of Justice, Federal Bureau of Investigation under contract J-FBI-98-083. Drs. Birdwell and Chiasson were also partially sup-

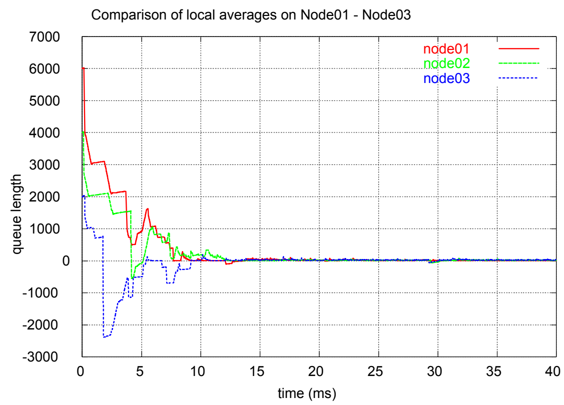

Figure 4: Experimental response of the load balancing algorithm. The average value of the gains are $(K_z = 0.5)$ $K_1 = 6667, K_2 = 4167, K_3 = 5000$ with constant p_{ij}.

ported by a Challenge Grant Award from the Center for Information Technology Research at the University of Tennessee. The work of C.T. Abdallah was supported in part by the National Science Foundation through the grant INT-9818312. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] C. Abdallah, N. Alluri, J. Birdwell, J. Chiasson, V. Churpryna, Z. Tang, and T. Wang. A linear time delay model for studying load balancing instabilities due to time delays in parallel computation. (2002). Submitted for Publication.

[2] C. Abdallah, J. Birdwell, J. Chiasson, V. Churpryna, Z. Tang, and T. Wang. Load balancing instabilities due to time delays in parallel computation. In "Proceedings of the 3rd IFAC Conference on Time Delay Systems" (December 2001). Sante Fe NM.

[3] J. Birdwell, J. Chiasson, C. Abdallah, Z. Tang, N. Alluri, V. Churpryna, and T. Wang. Load balancing instabilities due to time delays in parallel computation. (2002). Submitted for Publication.

[4] J. Birdwell, R. Horn, D. Icove, T. Wang, P. Yadav, and S. Niezgoda. A hierarchical database design and search method for codis. In "Tenth International Symposium on Human Identification" (September 1999). Orlando, FL.

[5] J. Birdwell, T. Wang, R. Horn, P. Yadav, and D. Icove. Method of indexed storage and retrieval of multidimensional information. In "Tenth SIAM Conference on Parallel Processing for Scientific Computation" (September 2000). U. S. Patent Application 09/671,304.

[6] J. Birdwell, T.-W. Wang, and M. Rader. The university of tennessee's new search engine for codis. In "6th CODIS Users Conference" (February 2001). Arlington, VA.

[7] A. Corradi, L. Leonardi, and F. Zambonelli. Diffusive load-balancing polices for dynamic applications. *IEEE Concurrency* **22**(31), 979–993 (Jan-Feb 1999).

[8] P. Dasgupta. "Performance Evaluation of Fast Ethernet, ATM and Myrinet under PVM, MS Thesis". University of Tennesse (2001).

[9] P. Dasgupta, J. D. Birdwell, and T. W. Wang. Timing and congestion studies under pvm. In "Tenth SIAM Conference on Parallel Processing for Scientific Computation" (March 2001). Portsmouth, VA.

[10] L. Kleinrock. "Queuing Systems Vol I : Theory". John Wiley & Sons (1975). New York.

[11] F. Spies. Modeling of optimal load balancing strategy using queuing theory. *Microprocessors and Microprogramming* **41**, 555–570 (1996).

[12] T. Wang, J. Birdwell, P. Yadav, D. Icove, S. Niezgoda, and S. Jones. Natural clustering of DNA/STR profiles. In "Tenth International Symposium on Human Identification" (September 1999). Orlando, FL.

[13] T. Wang, J. D. Birdwell, P. Yadav, D. J. Icove, S. Niezgoda, and S. Jones. Natural clustering of DNA/STR profiles. In "Tenth International Symposium on Human Identification" (September 1999). Orlando, FL.

[14] M. Willebeek-LeMair and A. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Systems* **4**(9), 979–993 (1993).