Proceedings of the 16th IASTED International Conference
PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS
November 9-11, 2004, Cambridge, MA, USA

GRID SCHEDULING DIVISIBLE LOADS FROM MULTIPLE SOURCES VIA LINEAR PROGRAMMING

Mequanint A. Moges
Department of Electrical and Computer Engr.
Stony Brook University
Stony Brook, NY 11794
email: mmoges@ece.sunysb.edu

Dantong Yu
Department of Physics
Brookhaven National Laboratory
Upton, NY 11973
email: dtyu@bnl.gov

Thomas G. Robertazzi
Department of Electrical and Computer Engr.
Stony Brook University
Stony Brook, NY 11794
email: tom@ece.sunysb.edu

ABSTRACT
To date solutions for optimal finish time and job allocation in divisible load theory are largely obtained only for network topologies with a single load originating (root) processor. However in large-scale data intensive problems with geographically distributed resources, load is generated from multiple sources. This paper introduces a new divisible load scheduling strategy for tree networks with two load originating processors. Solutions for an optimal allocation of fraction of loads to nodes in single level tree networks are obtained via linear programming. Performance evaluation of a two source homogeneous single level tree network with concurrent communication strategy is presented.

KEY WORDS
Divisible Loads, Scheduling, Tree Networks, Linear Programming, Multiple Source.

1 Introduction

The problem of minimizing the processing time of extensive processing loads originating from various sources presents a challenge that, if successfully met, could foster a range of new creative applications. Inspired by this challenge, we sought to apply divisible load theory to the problem of grid computing scheduling involving multiple sources connected to multiple sinks. So far research in this area includes [1] where tasks arrive according to a basic stochastic process to multiple nodes and [2] presents a first step technique for scheduling divisible loads from multiple sources to multiple sinks, with and without buffer capacity constraints.

Divisible load theory [3],[4],[5] is characterized by the fine granularity and large volume of loads. There are also no precedence relations among the data elements. Such a load may be arbitrarily partitioned and distributed among processors and links in a system. The approach is particularly suited to the processing of very large data files in signal processing, image processing, experimental data processing, grid computing and computer utility applications.

There has been an increasing amount of study in divisible load theory since the original work of Cheng and Robertazzi [6] in 1988. The majority of these studies develop an efficient load distribution strategy and protocol in order to achieve optimal processing time in networks with a single root processor. The optimal solution is obtained by forcing the processors over a network to all stop processing simultaneously. Intuitively, this is because the solution could be improved by transferring load if some processors were idle while other are still busy [7]. Such studies for network topologies including linear daisy chains, tree and bus networks using a set of recursive equations were presented in [6],[8],[9] respectively. There have been further studies in terms of load distribution policies for hypercubes [10] and mesh networks [11]. The concept of equivalent networks [12] was presented for complex networks such as multilevel tree networks. Work has also considered scheduling policy with multi-installment [13], multi-round algorithms [14], independent task scheduling [15], fixed communication charges [16], detailed parameterization and solution reporting time optimization [17] and combinatorial optimization [18]. Recently, though divisible load theory is fundamentally a deterministic theory, a study has been done to show some equivalence to Markov chain models [19].

As mentioned earlier, almost all of the previous research has assumed a network in which the processing load originates at a single node. This paper, unlike the previous research papers, presents the application of divisible load theory to tree networks with two load originating (root) processors. Applications include high energy and nuclear physics experiments that require an effective analysis of extensive data by geographically distributed nodes that must work closely together.

The organization of this paper is as follows. In section 2, the system model used in this paper is discussed. The scheduling of divisible load in single level tree networks

for concurrent communication strategy with two root processors is presented in section 3. Section 4 presents the respective performance analysis results in terms of finish time. Finally the conclusion appears in section 5.

2 Two Root Processors System Model

In this section, the various network parameters used in this paper are presented along with some notation and definitions. The network topology discussed in this study is a tree network consisting of two root processors (P_1 and P_2) and $N-2$ child processors (P_3, ... , P_N) with $2(N-2)$ links as shown in Fig. 1. It will be assumed that the total processing load considered here is of the arbitrarily divisible kind that can be partitioned into fractions of loads to be assigned to each processor over a network. The two root processors keep their own fraction of loads (α_1 and α_2) and communicate/distribute the other fractions of loads (α_3, α_4, ... α_N) assigned to the rest of processors in the network. Each processor begins to process its share of the load once the load share from either root processor has been completely received.

The load distribution strategy from either root processors to the child processors may be sequential or concurrent. In the sequential load distribution strategy, each root processor is able to communicate with only one child at a time. However, in the case of concurrent communication strategy, each root processor can communicate simultaneously/concurrently with all the child processors. The latter communication strategy can be implemented by using a processor which has a CPU that loads an output buffer for each output link. In this case it can be assumed that the CPU distributes the load to all of its output buffers at a rapid enough rate so that the buffer outputs are concurrent. In this paper the concurrent communication strategy is considered.

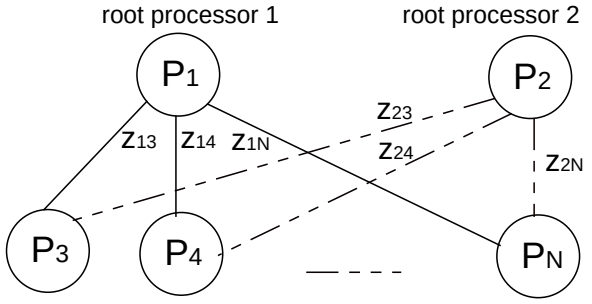

Figure 1. Single level tree network with two root processors.

2.1 Notations and Definitions:

L_i: Total processing load originated from root processor i, (i = 1, 2).

α_i: The total fraction of load that is assigned by the root processors to child i.

α_{1i}: The fraction of load that is assigned to processor i by the first root processor.

α_{2i}: The fraction of load that is assigned to processor i by the second root processor.

$$\alpha_i = \alpha_{1i} + \alpha_{2i},\ i = 3, 4, ..., N.$$

ω_i: A constant that is inversely proportional to the processing speed of processor i in the network.

z_{1i}: A constant that is inversely proportional to the speed of link between the first root processor and the i^{th} child processor in the network.

z_{2i}: A constant that is inversely proportional to the speed of link between the second root processor and the i^{th} child processor in the network.

T_{cp}: Processing intensity constant. This is the time that it takes the i^{th} processor to process the entire load when $\omega_i = 1$. The entire load can be processed on the i^{th} processor in time $\omega_i T_{cp}$.

T_{cm}: Communication intensity constant. This is the time that it takes to transmit all the processing load over a link when $z_i = 1$. The entire load can be transmitted over the i^{th} link in time $z_i T_{cm}$.

T_i: The total time that elapses between the beginning of the scheduling process at $t = 0$ and the time when processor i completes its processing, $i = 1, ..., N$. This includes communication time, processing time and idle time.

T_f: This is the time when the last processor finishes processing.

$$T_f = \max(T_1, T_2, \ldots, T_N).$$

One convention that is followed in this paper is that the total load originating at the two root processors is assumed to be normalized to be a unit load. That is,

$$L_1 + L_2 = 1.$$

3 Optimal Scheduling Strategies

The load scheduling strategies presented here targets finding solutions for optimal finish time (make-span) and job allocation in single level tree networks with two root processors. Most previous load scheduling strategies in divisible load models can be solved algebraically in order to find the optimal finish time and load allocation to processors

and links. In this case optimality is defined in the context of the specific interconnection topology and load distribution schedule used. An optimal solution is usually obtained by forcing all processors to finish computing at the same time. Intuitively, if there exist idle processors in the network, load can be transferred from busy processors to those idle processors [7]. This section covers the load scheduling strategies proposed for tree networks with two root processors.

3.1 Single Level Tree Network with Two Root Processors

The network topology considered here is a tree network with two root processors and $N-2$ child processors. In this case, it is assumed that the total processing load originates from the two root processors (P_1 and P_2). The scheduling strategy involves the partitioning and distribution of the processing loads originated from P_1 and P_2 to all the processors. The load distribution process proceeds as follows: the total processing loads originated from P_1 and P_2 are assumed to be L_1 and L_2 respectively. Each root processor keeps some fraction of the respective processing load for itself to compute and distributes the remaining load simultaneously to the child processors. The timing diagram shown in Fig. 2, shows the load distribution process discussed above. The figure shows that at time $t = 0$, the processors are all idle. The child processors start computation only after completely receiving their assigned fraction of load from either of the two root processors.

Now the equations that govern the relations among various variables and parameters in the network can be written as follows:

$$T_1 = \alpha_1\omega_1 T_{cp} \quad (1)$$

$$T_2 = \alpha_2\omega_2 T_{cp} \quad (2)$$

$$T_3 = (\alpha_{13} + \alpha_{23})\omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} \quad (3)$$

$$T_N = (\alpha_{1N} + \alpha_{2N})\omega_N T_{cp} + \alpha_{1N} z_{1N} T_{cm}. \quad (4)$$

As it was mentioned earlier, since total measurement load originating at the two root processors is assumed to be normalized to a unit load, the fractions of the total processing load should sum to one as:

$$L_1 + L_2 = 1 \quad (5)$$

$$\alpha_1 + \alpha_2 + \alpha_3 + ... + \alpha_{N-1} + \alpha_N = 1 \quad (6)$$

Since

$$L_1 = \alpha_1 + \sum_{j=3}^{N} \alpha_{1,j} \quad (7)$$

$$L_2 = \alpha_2 + \sum_{j=3}^{N} \alpha_{2,j} \quad (8)$$

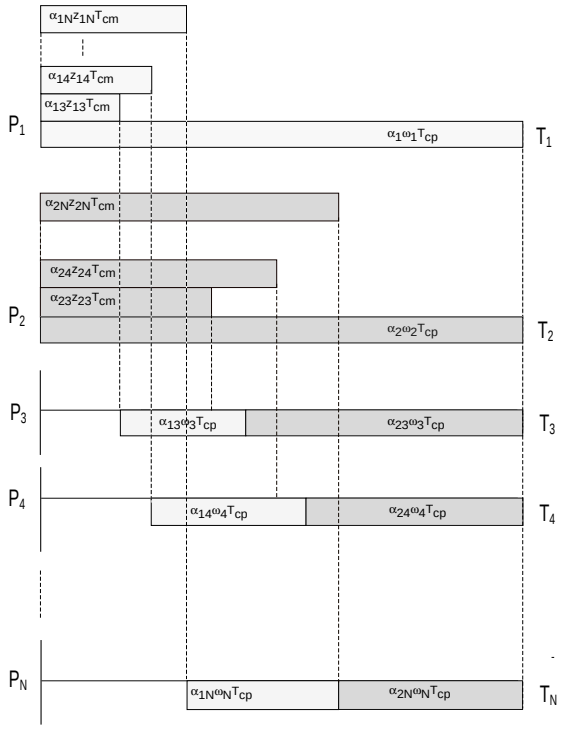

Figure 2. Timing diagram for a single level tree network with two root processors and concurrent communication.

The normalization equation given above can also be written in terms of the fraction of loads as:

$$\alpha_1 + \alpha_2 + \sum_{j=3}^{N} \alpha_{1,j} + \sum_{j=3}^{N} \alpha_{2,j} = 1 \quad (9)$$

As it can be seen from the timing diagram shown in Fig. 2, all processors stop processing at the same time, thus we have:

$$T_1 = T_2 = T_3 = \ldots = T_N$$

Based on the above set of equations, one can write the following set of $N-1$ equations:

$$\alpha_1\omega_1 T_{cp} = \alpha_2\omega_2 T_{cp} \quad (10)$$

$$\alpha_2\omega_2 T_{cp} = \alpha_3\omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} \quad (11)$$

$$\alpha_3\omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} = \alpha_4\omega_4 T_{cp} + \alpha_{14} z_{14} T_{cm} \quad (12)$$

$$\alpha_{i-1}\omega_{i-1} T_{cp} + \alpha_{1i-1} z_{1i-1} T_{cm} = \alpha_i\omega_i T_{cp} + \alpha_{1i} z_{1i} T_{cm} \quad (13)$$

where i = 3, 4, ..., N.

As it can be seen from the above set of equations, there is a smaller number of equations than the number of unknowns. Another $N-2$ equations can be written by setting up relationship between the fractions of loads within each child processor as:

$$\alpha_{23} z_{23} T_{cm} \leq \alpha_{13}(z_{13} T_{cm} + \omega_3 T_{cp}) \quad (14)$$

$$\alpha_{24} z_{24} T_{cm} \leq \alpha_{14}(z_{14} T_{cm} + \omega_4 T_{cp}) \quad (15)$$

$$\alpha_{2N} z_{2N} T_{cm} \leq \alpha_{1N}(z_{1N} T_{cm} + \omega_N T_{cp}) \quad (16)$$

In this case, there will be $2N-1$ equations (including the normalization equations) and $2N-2$ unknowns. This will lead us to a linear programming problem with the objective function that minimizes the total processing time of the network. In this case the objective function will be:

Minimize:

$$T_f = \alpha_1 \omega_1 T_{cp} \quad (17)$$

Subject to:

$$\alpha_1 \omega_1 T_{cp} - \alpha_2 \omega_2 T_{cp} = 0$$
$$\alpha_2 \omega_2 T_{cp} - \alpha_3 \omega_3 T_{cp} - \alpha_{13} z_{13} T_{cm} = 0$$
$$\alpha_3 \omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} - \alpha_4 \omega_4 T_{cp} - \alpha_{14} z_{14} T_{cm} = 0$$
$$\vdots$$
$$\alpha_{N-1} \omega_{N-1} T_{cp} + \alpha_{1N-1} z_{1N-1} T_{cm} - \alpha_N \omega_N T_{cp} - \alpha_{1N} z_{1N} T_{cm} = 0$$

$$L_1 - \alpha_1 - \sum_{j=3}^{N} \alpha_{1,j} = 0$$
$$L_2 - \alpha_2 - \sum_{j=3}^{N} \alpha_{2,j} = 0$$

$$\alpha_{23} z_{23} T_{cm} - \alpha_{13}(z_{13} T_{cm} + \omega_3 T_{cp}) \leq 0$$
$$\alpha_{24} z_{24} T_{cm} - \alpha_{14}(z_{14} T_{cm} + \omega_4 T_{cp}) \leq 0$$
$$\alpha_{2N} z_{2N} T_{cm} - \alpha_{1N}(z_{1N} T_{cm} + \omega_N T_{cp}) \leq 0$$

$$\alpha_i \geq 0$$

The first set of equality equations enforce the constraints that all processors should stop processing at the same time for the optimality condition. The inequality set of constraints state that the child processors do their computation without any interruption. The last equation is that the fractions of the assigned load should be positive. Finally, the objective function is to minimize the total processing time needed to process the loads from the two root processors.

4 Processing Finish Time (Make Span) Results

This section presents the plots of finish time vs. number of processors in a single level tree network with two root processors. The results are obtained by using linear programming with the objective function of minimizing the total processing time. In this case a homogeneous network is considered to study the effect of communication and computation speed variations and the number of processors on the total processing time.

In Fig. 3, the finish time is plotted against the number of processors in the network for different inverse bus speeds, z_1 which is the communication link between the first root processor and the child processors. The communication link between the second root processor and the child processors is set to be fixed to z_2 = 1.

The tree network that is used to obtain the plot in Fig. 3 has a homogeneous link and processor speed. In this case ω = 2 and the values of T_{cm} and T_{cp} are also set to be equal to one. The plot shows that a better finish time is obtained as the number of processors in the network is increased and when the link speed is faster. This is expected as more processors would have been involved in computation as the link speed is increased.

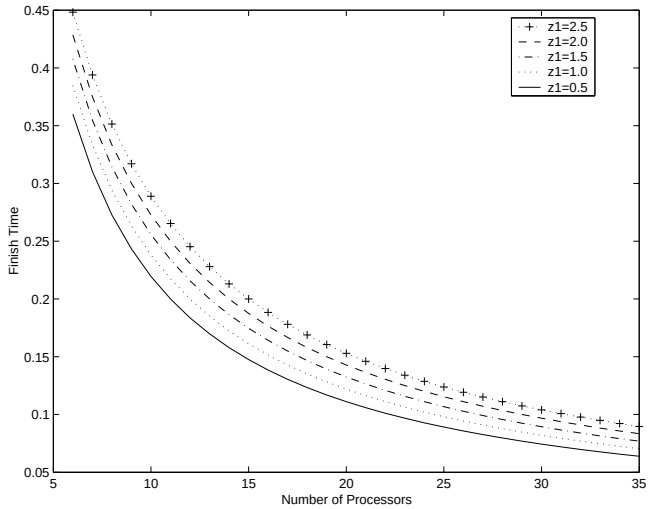

Figure 3. Finish time versus number of processors, for two root sources single level homogeneous tree network and variable inverse bus speed, z1, (first root processor links).

The plot shown in Fig. 4 shows the performance of the network when the communication link between the first root processor and the child processors z_1 is fixed and the communication link between the second root processor and the child processors z_2 varies from 0.5 to 2.5. For these parameters, as shown in the plot the finish time is the same regardless of the variation of z_2. The computation of the fraction of load that originates from the second processor starts only after the completion of the processing load that originated from the first processor. Thus the variation of the link speed z_2 has no effect on the total processing time. As mentioned in the earlier sections, this whole process assumes that the nodes are always busy computing the loads originated from the two root processors. That is, there is no idle time between computation time.

In Fig. 5, the finish time is plotted against the number

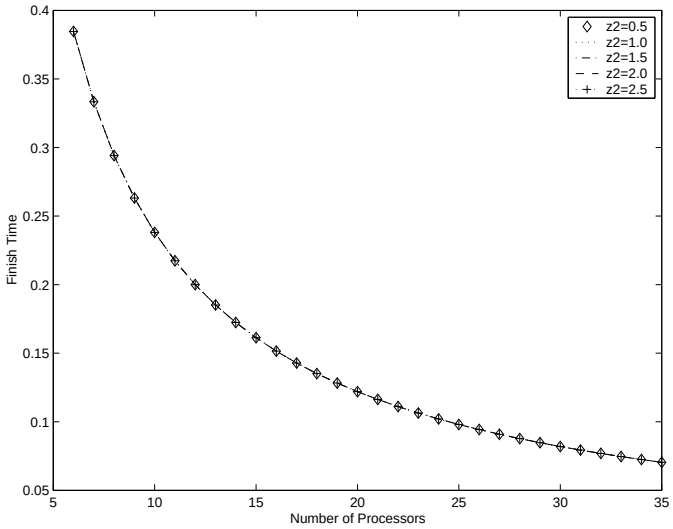

Figure 4. Finish time versus number of processors, for two root sources single level homogeneous tree network and variable inverse bus speed, z2, (second root processor links).

of processors in the network for different inverse processor speed, ω. In this case z_1 and z_2 are set to be equal to 0.5 and the values of T_{cm} and T_{cp} are set to be equal to one. The plot shows that a better finish time is obtained as the number of processors in the network is increased and when the processor speed is faster.

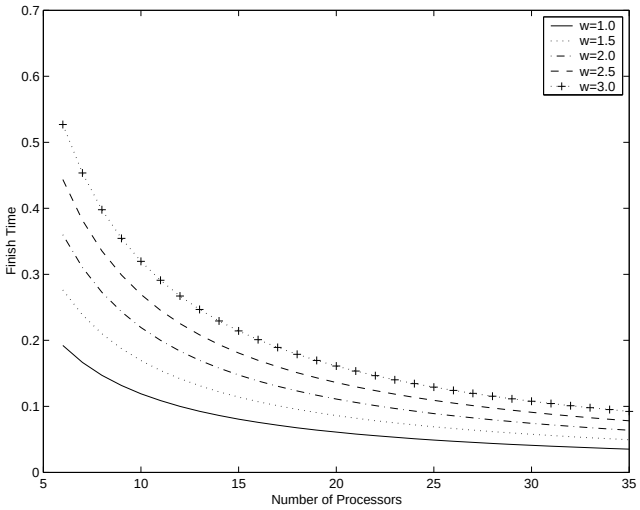

Figure 5. Finish time versus number of processors, for two root sources single level homogeneous tree network and variable inverse processor speed, ω.

5 Conclusion

In this paper, solutions for optimum allocation of loads to processors over a single level tree networks with two root processors are obtained via an optimization problem that uses linear programming. The objective is to find a minimum processing time by distributing processing loads (jobs) that originate from different sources. A performance evaluation of these networks with this new scheduling strategy is provided and the effects of the number of processors and speeds of links and processors on the finish time are studied. For the homogeneous single level tree network with a concurrent communication load distribution strategy considered in this study the finish time is inversely proportional to the number of processors in the network.

Currently, we are in the process of extending the two processor model to consider sequential load distribution strategy and multi-level tree networks. It will be also interesting to study the complexity involved in dealing with networks with more than two root processors.

Acknowledgments

The authors would like to acknowledge the support of NSF grant CCR-99-12331.

References

[1] K. Ko and T.G. Robertazzi, Scheduling in an environment of multiple job submissions, *Proceedings of the 2002 Conference on Information Sciences and Systems*, Princeton NJ, USA, 2002.

[2] H.M. Wong, B. Veeravalli, D. Yu and T.G. Robertazzi, Data intensive grid scheduling: Multiple sources with capacity constraint, *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, Marina del Rey, CA, USA, 2003.

[3] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, *Scheduling divisible loads in parallel and distributed systems* (Los Alamitos, CA: IEEE Computer Society Press, 1996).

[4] V. Bharadwaj, D. Ghose, T.G. Robertazzi, Divisible load theory: A new paradigm for load scheduling in distributed systems, *Cluster Computing*, 6, 2003, 7-18.

[5] T.G. Robertazzi, Ten reasons to use divisible load theory, *Computer*, 36, 2003, 63-68.

[6] Y.C. Cheng and T.G. Robertazzi, Distributed computation with communication delays, *IEEE Transactions on Aerospace and Electronic Systems*, 22, 1988, 60-79.

[7] J. Sohn and T.G. Robertazzi, Optimal divisible load sharing for bus networks, *IEEE Transactions on Aerospace and Electronic Systems*, 32, 1996, 34-40.

[8] Y.C. Cheng and T.G. Robertazzi, Distributed computation for a tree network with communication delays,

IEEE Transactions on Aerospace and Electronic Systems, 26, 1990, 511-516.

[9] S. Bataineh and T.G. Robertazzi, Bus oriented load sharing for a network of sensor driven processors, *IEEE Transactions on Systems, Man and Cybernetics*, 21, 1991, 1202-1205.

[10] J. Blazewicz and M. Drozdowski, Scheduling divisible jobs on hypercubes, *Parallel computing*, 21, 1996, 1945 - 1956.

[11] M. Drozdowski and W. Glazek, Scheduling divisible loads in a three-dimensional mesh of processors, *Parallel Computing*, 25, 1999, 381-404.

[12] T.G. Robertazzi, Processor equivalence for a linear daisy chain of load sharing processors, *IEEE Transactions on Aerospace and Electronic Systems*, 29, 1993, 1216-1221.

[13] V. Bharadwaj, D. Ghose, V. Mani, Multi-installment load distribution in tree networks with delays, *IEEE Transactions on Aerospace and Electronic Systems*, 31, 1995, 555-567.

[14] Y. Yang, H. Casanova, UMR: A multi-round algorithm for scheduling divisible workloads, *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[15] O. Beaumont, A. Legrand, and Y. Robert, Optimal algorithms for scheduling divisible workloads on heterogeneous systems, *12th Heterogeneous Computing Workshops*, HCW'2003, 2003.

[16] J. Blazewicz and M. Drozdowski, Distributed processing of distributed jobs with communication startup costs, *Discrete Applied Mathematics*, 76, 1997, 21-41.

[17] A.L. Rosenberg, Sharing partitionable workloads in heterogeneous NOWs: greedier is not better. In D.S. Katz, T. Sterling, M. Baker, L. Bergman, M. Paprzycki, and R. Buyya, editors, *Cluster Computing*, 2001, 124-131.

[18] P.F. Dutot, Divisible load on heterogeneous linear array, *Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[19] M. Moges and T.G. Robertazzi, Optimal divisible load scheduling and Markov chain models, *Proceedings of the 2003 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, USA, 2003.