The Average Diffusion Method for the Load Balancing Problem*

Gregory Karagiorgos and Nikolaos M. Missirlis

Department of Informatics and Telecommunications,
University of Athens,
Panepistimioupolis 157 84, Athens, Greece,
{greg,nmis}@di.uoa.gr

Abstract. This paper proposes the *Average Diffusion* (ADF) method for solving the load balancing problem. It is shown that a sufficient and necessary condition for the ADF method to converge to the uniform distribution of loads is the induced network of processors to be d-regular, connected and not bipartite. Next, we proceed and apply Fourier analysis determining the convergence factor γ in terms of the diffusion parameters c_{ij} (weighted case) when the network of processors is a ring and 2D-torus. It is shown that $c_{ij} = \frac{1}{2}$ and $c_{ij} \in (0, \frac{1}{2})$ when the network is a ring and 2D-torus, respectively, thus solving partially the open problem which concerns the determination of the diffusion parameters c_{ij}.

Keywords: Load balancing, diffusion method, multiple parameters, d-regular graphs, distributed computing.

1 Introduction

1.1 Environmental models: the origin of the problem

Recently a number of numerical models concerning the simulation of environment (weather prediction, air pollution, occean simulation) have been parallelized resulting in a considerable reduction in time (e.g. see [13] and the references herein). These studies show the suitability of the atmospheric computations for parallelization. These models use a three dimensional grid to simulate the physical processes at the atmosphere. The computations involved in such simulation models are of three types: "**dynamics**", "**physics**" and "**chemistry**". Dynamics computations simulate the fluid dynamics at the atmosphere (advection, diffusion) and are carried out on the horizontal domain. Since these computations use explicit numerical schemes to discretize the involved differential equations, they are inherently parallel. Alternatively, the physical and chemistry computations simulate the physical and chemistry processes such as clouds, precipitations, radiative transfer and are carried out on the vertical grid. These

* Research is supported by the Greek General Secretary of Research and Technology ($E\Pi ET$-II No. 87) and the National and Kapodistrian University of Athens (No. 70/4/4917).

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 623–632, 2002.

computations must be carried out for each grid point and do not require any data from its neighbour grid points. As the computations for each grid column are independent, domain decomposition techniques is best to be applied to the horizontal domain. The column computations refer to the physical and chemistry processes which can be subject to significant spatial and temporal variation in the computational load per grid point. As more sophisticated physics and chemistry will be introduced in the above environmental models, these computational load imbalances will tend to govern the parallel performance. Furthermore, on a network of processors, the performance of each processor may differ. To achieve good performance on a parallel computer, it is essential to establish and maintain a balanced work load among the processors. For this reason, it is necessary to calculate the amount of load to be migrated from each processor to its neighbours. Then, it is also necessary to migrate the load based on this calculation. In case of the environmetal models the amount of work on each processor is proportional to the number of grid points on the processor. To our knowledge there is a limited use of distributed load balancing schemes. In [1] we use the diffusion method to solve the load balancing problem in the Regional Atmospheric Modelling System (RAMS) obtaining encouriging results.

1.2 The load balancing problem

The ultimate goal of any load balancing scheme, *static* or *dynamic*, is to improve the perfomance of a parallel application code, using efficiently the parallel or distributed systems. For this purpose the computational work must be well balanced across processors and second the time spent performing interprocessor communication must be small. For many applications it is possible to make a priori estimates of load distribution and such load balancing schemes are called *static*. In other applications the computational requirements vary over time in an unpredictable way and it is no possible to make a priori estimates of load distribution. Such load balancing schemes are called *dynamic*.

Research on dynamic load balancing has focused on suboptimal algorithms that use local information in a distributed memory architecture. These algorithms describe rules for migrating tasks on overutilized processors to underutilized processors in the network. Tradeoffs exist between achieving the goal of completely balancing the load and the communications costs associated with migrating tasks. Every load balancing algorithm decides to give load to the neighbouring processors depending only on local information, i.e. by comparing the load of the underlying processor with the load of the neighbouring processors and thereby trying to reach a *local* equilibrium. After a number of diffusion cycles, which in the worst case is quadratic to the number of processors, a *global* equilibrium of the load will be reached [5].

The parallel and distributed system is modeled as an undirected graph where nodes represent the processors and the edges represent the network connections. Each processor is associated with a real variable which reflect the work load currently running on it. We assume that during the balancing process no load

is generated or consumed and the graph does not change. The system is synchronous, homogeneous and the network connections have unbounded capacity. We concentrate only on solving the data flow problem using the ADF method ignoring the migration problem [11].

The original diffusion algorithm has been proposed by Cybenko [9] and, independently, by Boillat [5]. Diffusion type algorithms [9, 5, 17] are some of the most popular ones for the flow problem. The quality of a diffusion algorithm can be measured in terms of number of iterations it requires to reach a balanced state. Recently the Diffusion method was combined with semi-iterative techniques [16] reducing the number of iterations by an order of magnitude [10–12, 14]. In addition, diffusion type methods are also used solving the flow problem for asynchronous distributed systems [3].

This paper is organized as follows. In section 2 we introduce the Average Diffusion (ADF) method. In section 3 we study its convergence analysis and in particular, we find necessary and sufficient conditions for convergence. In section 4 we use Fourier analysis to determine the values of the diffusion parameters c_{ij} and present our results for the ring and the 2D-torus processor graphs. It is shown that $c_{ij} = \frac{1}{2}$ and $c_{ij} \in (0, \frac{1}{2})$ when the network is a ring and 2D-torus, respectively, thus solving partially the open problem which concerns the determination of the diffusion parameters c_{ij}. Finally, our conclusions are discussed in section 5.

2 The Average Diffusion Method

2.1 The method

Let $G = (V, E)$ be a connected, undirected graph with $|V|$ nodes and $|E|$ edges. Let $u_i \in \Re$ be the load of node $v_i \in V$ and $u \in \Re^{|V|}$ be the vector of load values. The average load per processor is

$$\bar{u} = \frac{1}{|V|} \sum_{i=1}^{|V|} u_i.$$

The computation of $\bar{u}$ requires global communication between the processors which is a time consuming process. To avoid the aforementioned problem we restrict the computation between neighbour processors. The load of processor i is computed as the weighted average work load of its direct neighbours as follows:

$$u_i = \frac{1}{\sum_{j \in A(i)} \hat{c}_{ij}} \sum_{j \in A(i)} \hat{c}_{ij} u_j$$

or

$$\sum_{j \in A(i)} \hat{c}_{ij}(u_i - u_j) = 0 \tag{1}$$

where u_i and u_j are the workloads of processors i and j, respectively, $A(i)$ is the set of nearest neighbors and $\hat{c}_{ij}$ are the nonnegative diffusion parameters. In matrix form, (1) is written as

$$L\overline{u} = 0 \tag{2}$$

where L is the weighted Laplacian matrix [11] of graph G and has the splitting

$$L = D - A,$$

where $D = (d_{ii})$ is the diagonal matrix with $d_{ii} = deg(i)$, $deg(i)$ denotes the degree of i and A is the adjacent matrix of the graph G. For solving the homogeneous linear system (2), we consider the iterative scheme

$$u^{(n+1)} = Bu^{(n)}, \tag{3}$$

where

$$B = D^{-1}A. \tag{4}$$

Let $B = (b_{ij})$ with

$$b_{ij} = \begin{cases} c_{ij}, \text{ if } i \neq j \text{ and } j \in A(i) \\ 0, \quad \text{otherwise} \end{cases}$$

where

$$c_{ij} = \frac{\hat{c}_{ij}}{\sum\limits_{j \in A(i)} \hat{c}_{ij}}. \tag{5}$$

From (5) it follows that

$$0 < c_{ij} < 1$$

since $\hat{c}_{ij} > 0$. Moreover, for B to be symmetric we must have $c_{ij} = c_{ji}$, which (because of (5)) imposes the condition $A(i) = A(j)$ since $\hat{c}_{ij} = \hat{c}_{ji}$.This means that all the processors must have the same number of neighbors, that is $d_{ii} = d$. In other words, the matrix B is symmetric when the graph is d-regular. In the sequel we develop our analysis under this assumption. The iterative method given by (3), will be referred to as the *Average Diffusion* (ADF) method and the matrix B is its *iteration matrix*. In case of the nonweighted Laplacian L, $\hat{c}_{ij} = 1$ and

$$b_{ij} = \begin{cases} \frac{1}{d} \text{ if } i \neq j \text{ and } j \in A(i) \\ 0 \text{ otherwise.} \end{cases}$$

2.2 The iteration matrix B

For the average work load of ADF to be invariant B must be doubly stochastic [2]. Let us first prove that B is row stochastic, i.e

$$\bar{u} = B\bar{u}, \tag{6}$$

where $\bar{u}$ is the vector whose every entry is exactly $\frac{\sum_i u_i}{|V|}$, which implies that (6) will hold if $\sum_{j \in A(i)} b_{ij} = 1$. But,

$$\sum_{j \in A(i)} b_{ij} = \sum_{j \in A(i)} c_{ij} = 1.$$

The last equality holds, because of (5). Therefore, B is row stochastic.

Lemma 1 *If the network graph is d-regular, then B is symmetric.*

Proof. For B to be symmetric we must have $B = B^T$, or because of (4) $A = DAD^{-1}$ which holds in case the elements of D are all equal. This is true since the network graph is d-regular. ∎

Finally, B is doubly stochastic since it is symmetric and row stochastic. In the following, we use the properties of B to establish conditions under which the ADF method converges to the uniform load distribution and determine its rate of convergence. Note, that B is also irreducible and nonnegative matrix. The matrix B satisfies the conditions to the *Perron-Frobenius* theorem [2, 16], hence for its eigenvalues μ_i, $i = 1, 2, \ldots, n$, we have

$$\mu_n \leq \mu_{n-1} \leq \ldots \leq \mu_2 < \mu_1 = 1.$$

The last inequality is strict since 1 is a simple eigenvalue. For convergence of ADF we must find conditions under which $\gamma(B) < 1$, where

$$\gamma(B) = \max_{i \neq 1} |\mu_i|$$

which will be referred to as the *convergence factor*.

3 Convergence Analysis

In this section we find necessary and sufficient conditions for the ADF method to converge.

Theorem 1 *The ADF method always converges to the uniform distribution if and only if the induced network is connected and not bipartite.*

Proof. A graph is bipartite if its vertex-set can be partitioned into two parts V_1 and V_2 such that each edge has one vertex in V_1 and one vertex in V_2. If we order the vertices so that those in V_1 come first, then the adjacency matrix of a bipartite graph takes the form [4]

$$A = \begin{bmatrix} 0 & K \\ K^T & 0 \end{bmatrix}, \tag{7}$$

where 0's are used to denote square zero block matrices on the diagonal of B and K is a rectangular nonnegative matrix. Since $B = D^{-1}A$ it is evident that B will also possess the form (7). It is well known that the eigenvalues of a matrix which takes the above form (7) occur in pairs $\pm\mu_i$ [4]. Hence, the eigenvalues of B satisfy

$$-1 = \mu_n < \mu_{n-1} \ldots \leq \mu_2 < \mu_1 = 1.$$

Therefore, the ADF method converges to the uniform distribution ($\gamma < 1$) if and only if -1 is not an eigenvalue of B. If the network graph is not bipartite, then -1 is not an eigenvalue of B. ∎

4 Local Fourier analysis

The conventional way to analyze the ADF method is to use matrix analysis [17]. This approach depends on the properties of the resulting diffusion matrix which in turn depends on the topology of the graph. In this section we use an alternative technique to analyze diffusion algorithms. This technique is the Fourier analysis. Assuming that the iterative diffusion methods solve numerically a Partial Differential Equation (e.g. the Convection-Diffusion equation) we can apply Fourier analysis to study its error smoothing effect [6–8, 15]. Fourier analysis applies only to linear constant coefficient PDEs on an infinite domain or with periodic boundary conditions. However, at a heuristic level this approach provides a useful tool for the analysis of more general PDE problems. Following the same idea, we will apply the Fourier analysis approach to the ADF method. When the graph is the ring or the 2D-torus we obtain the same results for the nonweighted case as in [17]. However, our derivation produces results in the weighted case also. In particular, we are able to determine the convergence factor $\gamma(B)$ in terms of the diffusion parameters c_{ij} and study their role in the convergence behaviour of ADF. Next, we will apply this technique for the ring and 2D-torus processor graphs.

4.1 The ring

At a local node the ADF scheme (3) can be written as

$$u_j^{(n+1)} = B_j u_j^{(n)} \tag{8}$$

where $B_j \equiv (c_{j+1}E + c_{j-1}E^{-1})$ is the *local ADF operator* and

$$Eu_j = u_{j+1}, \;\; E^{-1}u_j = u_{j-1}$$

are the *forward-shift* and *backward-shift* operators in the x-direction, respectively. Expressing (8) in terms of the error vector $e^{(n)} = u_j^{(n)} - u$ we have

$$e_j^{(n+1)} = B_j e_j^{(n)}, n = 0, 1, 2, ...$$

If the error function $e_j^{(n)}$ is the complex sinusoid e^{ikx}, we have

$$B_j e^{ikx} = \mu_j(k) e^{ikx}$$

where

$$\mu_j(k) = c_{j+1} e^{ikh} + c_{j-1} e^{-ikh},\ h = \frac{1}{N}$$

where N is the number of processors. So, we may view e^{ikx} as an eigenfunction of B_j with eigenvalues $\mu_j(k)$. Furthermore,

$$|\mu_j(k)| = ([(c_{j+1} + c_{j-1}) \cos kh]^2 + [(c_{j+1} - c_{j-1}) \sin kh]^2)^{1/2}. \tag{9}$$

In the *nonweighted* case $c_j = \frac{1}{d}$, where $d = 2$ for the ring, hence (9) becomes

$$|\mu_j(k)| = cos(kh) \tag{10}$$

where k is selected such that $|\mu_j(k)|$ attains its maximum value less than 1. Letting $k = 2\pi\ell$, $\ell = 0, \pm 1, \pm 2, \ldots, \pm(N-1)$, (10) yields

$$\gamma(B_j) = cos(2\pi h)).$$

Note, that $\gamma(B_j)$ is equal to $\gamma(B)$, where $\gamma(B)$ is determined using matrix analysis [17].
In the weighted case for the operator B_j to be symmetric we must have

$$c_{j+1} = c_{j-1} \tag{11}$$

and (9) yields

$$|\mu_j(k)| = 2c_{j+1} \cos kh.$$

Moreover, for the operator B_j to be row stochastic we must have

$$c_{j-1} + c_{j+1} = 1. \tag{12}$$

From (11) and (12) it follows that

$$c_j = \frac{1}{2},\ j = 0, 1, 2, \ldots, N-1.$$

Therefore, the diffusion parameters must be equal to $\frac{1}{2}$ in case of the ring network topology and the weighted case coincides with the nonweighted one.

4.2 The 2D-torus

Using a similar approach as in the previous section, we will define the local ADF operator for 2D-torus network graph. Define the x_1-direction, (x_2-direction) *forward-shift* and *backward-shift* operators, E_1 and E_1^{-1} (E_2 and E_2^{-1}), as

$$E_1 u_{ij} = u_{i+1,j},\ E_1^{-1} u_{ij} = u_{i-1,j},$$
$$E_2 u_{ij} = u_{i,j+1},\ E_2^{-1} u_{ij} = u_{i,j-1}.$$

Then, the *local ADF operator* for 2D-torus network graph is $B_{ij} \equiv (c_{i+1,j}E_1 + c_{i-1,j}E_1^{-1} + c_{i,j+1}E_2 + c_{i,j-1}E_2^{-1})$ and thus we have

$$B_{ij}e^{i(k_1x_1+k_2x_2)} = \mu_{ij}(k_1,k_2)e^{i(k_1x_1+k_2x_2)},$$

where

$$\mu_{ij}(k_1,k_2) = (c_{i+1,j}e^{ik_1h} + c_{i-1,j}e^{-ik_1h} + c_{i,j+1}e^{ik_2h} + c_{i,j-1}e^{-ik_2h}),$$

$h = \frac{1}{\sqrt{N}}$. Furthermore,

$$|\mu_{ij}(k_1,k_2)| = \big([(c_{i+1,j} + c_{i-1,j})\cos k_1h + (c_{i,j+1} + c_{i,j-1})\cos k_2h]^2 + [(c_{i+1,j} - c_{i-1,j})\sin k_1h + (c_{i,j+1} - c_{i,j-1})\sin k_2h]^2\big)^{1/2}. \quad (13)$$

In the *nonweighted* case $c_{ij} = \frac{1}{d}$, where $d = 4$ for the 2D-torus, hence (13) becomes

$$|\mu_{ij}(k_1,k_2)| = \frac{1}{2}(cos(k_1h) + cos(k_2h)) \quad (14)$$

where k_1,k_2 are selected such that $|\mu_{ij}(k_1,k_2)|$ attains its maximum value less than 1. Letting $k_1,k_2 = 2\pi\ell$, $\ell = 0, \pm1, \pm2, \ldots, \pm\sqrt{N}-1)$, (14) yields

$$\gamma(B_{ij}) = \frac{1}{2}(1 + \cos(2\pi h)).$$

Note, that $\gamma(B_{ij})$ is equal to $\gamma(B)$, where $\gamma(B)$ is determined using matrix analysis [17].
In the weighted case for the operator B_{ij} to be symmetric we must have

$$c_{i+1,j} = c_{i-1,j} \text{ and } c_{i,j+1} = c_{i,j-1} \quad (15)$$

and (13) yields

$$|\mu_{ij}(k_1,k_2)| = 2(c_{i+1,j}\cos k_1h + c_{i,j+1}\cos k_2h).$$

Moreover, for the operator B_{ij} to be row stochastic we must have

$$c_{i+1,j} + c_{i-1,j} + c_{i,j-1} + c_{i,j+1} = 1. \quad (16)$$

Combining (15) and (16) we obtain

$$c_{i+1,j} + c_{i,j+1} = \frac{1}{2} \quad (17)$$

or any combination of c_{ij}s from (15) must have sum equal to $\frac{1}{2}$. From (17) it follows that for $c_{ij} \in (0, \frac{1}{2})$ the ADF method converges and the "optimum" values for c_{ij}, which minimize $\gamma(B_{ij})$, must lie in the above interval. Choosing $c_{ij} = \frac{1}{4}$, the weighted case coincides with the nonweighted one. This was also the case in the ring topology.

5 Conclusions

In this paper we introduced an iterative distributed load balancing scheme, the ADF method. We showed that for the workload to be invariant the graph must be d-regular. Moreover, we found that a necessary and sufficient condition for the convergence of the ADF method is the processor network to be connected and not bipartite. The use of Fourier analysis gave us the ability to find a closed from formula for the eigenvalues of the iteration matrix B in terms of the diffusion parameters c_{ij}. For the ring and 2D-torus the formula coincides with the one given by [17] in the nonweighted case, thus verifying the validity of our approach. In this work we found that the diffusion parameters c_{ij} must be equal to $\frac{1}{2}$ in case of a ring topology. Moreover, c_{ij} must lie in the interval $(0, \frac{1}{2})$ for the 2D-torus, thus solving partially the open problem which concerns the determination of the diffusion parameters c_{ij}. Currently, our research is focused on the determination of the "optimum" values of c_{ij}s for more general graphs ($2d$-regular).

References

1. Balou A., Karagiorgos G., Kontarinis A., Missirlis N. M., *Optimization and parallelization of the RAMS model*, Thechnical Report 3.3, *ΕΠΕΤ-ΙΙ* No. 87, 2001 (in greek).
2. Berman A., Plemmons R.J., *Nonnegative matrices in the mathematical sciences*, Academic Press, 1979.
3. Bertsekas D.P. and Tsitsiklis J.N., *Parallel and distributed computation: Numerical Methods*, Prentice-Hall, 1989.
4. Biggs N. *Algebraic graph theory*, Cambridge University Press, Cambridge, 1974.
5. Boillat J.E., Load balancing and poisson equation in a graph, *Concurrency: Practice and Experience* **2**, 1990, 289-313.
6. Boukas L.A., Missirlis N. M., The parallel local modified SOR for nonsymmetric linear systems, *Intern. J. Computer Math* **68**, 1998, 153-174.
7. Brandt A., Multi-level adaptive solutions to boundary-value problems, *Math. Comput.* **31**, 1977, 333-390.
8. Chan T. F., Kuo C.-C.J., Tong C., Parallel elliptic preconditioners: Fourier analysis and performance on the connection machine, *Computer Physics Communications* **53**, 1989, 237-252.
9. Cybenko G., Dynamic load balancing for distributed memory multi-processors, *J. Parallel Distrib. Comp.* **7**, 1989, 279-301.
10. Diekmann R., Frommer A., Monien B., Efficient schemes for nearest neighbour load balancing, *Parallel Computing* **25**, 1999, 789-812.
11. Hu Y.F, Blake R.J., An improved diffusion algorithm for dynamic load balancing, *Parallel Computing* **25**, 1999, 417-444.
12. Karagiorgos G., Missirlis N.M., Iterative Algorithms for Distributed Load Balancing, In proc. of 4th Intern. Conf. On Principles Of DIstributed Systems, *Special issue of Intern. Journal on Informatics*, 2000, 37-54.
13. Karagiorgos G. and Missirlis N.M., Iterative Load Balancing Schemes for Air Pollution Models", S. M. Margenov, J. Wasniewski, P. Yalamov (Eds.): *Large-Scale Scientific Computing*, Third Intern. Conference, LSSC 2001, **LNCS 2179**, 291-298.

14. Karagiorgos G., Missirlis N.M., Accelerated Diffusion Algorithms for Dynamic Load Balancing, (*submitted*).
15. Kuo C.-C. J., Levy B. C. and Musicus B. R., A local relaxation method for solving elliptic PDE's on mesh-connected arrays, *SIAM j. Sci. Statist. Comput.*, **8**, 1987, 530-573.
16. Varga R., *Matrix iterative analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
17. Xu C.Z. and Lau F.C.M., *Load balancing in parallel computers: Theory and Practice*, Kluwer Academic Publishers, Dordrecht, 1997.