4K−4

A Decomposition Algorithm for Optimal Load Balancing in Tree Hierarchy Networks

Jie LI and Hisao KAMEDA

University of Electro-Communications

1. Introduction We study the static load balancing problem in a computer network with the tree hierarchy configuration. It is formulated as a nonlinear optimization problem. After studying the optimal solution to the tree hierarchy network optimization problem, we demonstrate that the special structure of the optimization problem leads to an interesting decomposition technique. A new decomposition algorithm to solve the optimization problem is presented. The proposed algorithm is compared to two other well known algorithms, the FD algorithm [2] and the D-S (Dafermos–Sparrow) algorithm [1]. It is shown that the proposed algorithm has good performance both on the storage requirements and the computational time requirements.

2. Model Description and Problem Formulation We define $G = (N, L)$ a *network* where N is a node set and L is a link set. If there is a path from node x to node y, then y is said to be *reachable* from x. We call node j an *ancestor* of node i (or node i a *descendant* of node j) if node j is reachable from node i. We call node j a *parent* of node i (or node i a *child* of node j) if there is a link from node i to node j. We call a node that has no ancestor nodes a *root* node. Now we define that a *tree hierarchy network* is the acyclic network in which there is a root node, and a node has the unique parent if the node is not the root node. In the tree hierarchy network, a node which has both its parent and children is called a *relay node*, and a node which has its parent and has not any child is called a *terminal node*. Here we divide the nodes in the tree hierarchy network into several groups, which are called *layers*. The root node belongs to the top layer (layer 0). Let us number the layers from top to bottom as layer $0, 1, 2, ..., p$. Nodes may be heterogeneous computer systems, that is, they may have different configurations, number of resources, and speed characteristics. However, they have sufficient processing capabilities, that is, a job may be processed from start to finish at any node in the network. Jobs arrive at node i ($i \in N$) according to a time-invariant Poisson process. A job arriving at node i can be processed at node i or be transferred to the nodes which are the ancestors of node i.

We give the following notation and assumptions.

- B: the set of relay nodes.
- Le: the set of terminal nodes.
- V_k: the set of nodes which belong to layer k, $k = 0, 1, 2, ..., p$.
- V_{kj}: the set of nodes in layer k that are children of node j, i.e., $V_{kj} = \{i \mid$ *node* i *is the child of node* j, $j \in V_{k-1} \cap B$, $i \in V_k\}$.
- β_i : the rate at which jobs are processed, also referred to as *load*, at node i, $i \in N$.
- x_{ij} : the job flow rate from child node i to parent node j on link (i,j), $(i,j) \in L$.
- ϕ_i: the mean external job arrival rate at node i, $i \in N$.
- ϕ'_i: the total job flow into node i, i.e.,

$$\phi'_i = \begin{cases} \phi_i + \sum_{l \in V_{k+1i}} x_{li}, & i \in V_k \cap B \text{ or } i \in V_0, \\ \phi_i, & i \in Le. \end{cases}$$

- $F_i(\beta_i)$: the mean node delay of a job processed at node i, $i \in N$.
- $G_{ij}(x_{ij})$: the mean communication delay on link $(i,j) \in L$.

In general, we assume the node delay function $F_i(\beta_i)$ is the differentiable, increasing, and convex function and the communication delay $G_{ij}(x_{ij})$ is the differentiable, non-decreasing, and convex function.

Let $D(\boldsymbol{\beta}, \mathbf{x})$ denote the mean delay (mean system response time) that a job spends in the tree hierarchy network from the time of its arrival until the time of its departure. We assume that the nodes and the communication links have product-form queuing network models. The optimization problem is as follows:

Minimize

$$D(\boldsymbol{\beta}, \mathbf{x}) = \sum_{i \in N} \frac{\beta_i}{\Phi} F_i(\beta_i) + \sum_{(i,j) \in L} \frac{x_{ij}}{\Phi} G_{ij}(x_{ij}), \qquad (1)$$

with respect to the variables $\{\beta_i\}$, $i \in N$, and $\{x_{ij}\}$, $(i,j) \in L$, (where $\Phi = \sum_{i \in N} \phi_i$)

subject to

$$\phi'_0 = \beta_0,$$

$$\phi'_i = \beta_i + x_{ij},\ i \in V_{kj},\ j \in V_{k-1} \cap B, \qquad (2.a)$$

$$k = 1, 2, ..., p,$$

$$\beta_i \geq 0,\ i \in N,\quad x_{ij} \geq 0,\ (i,j) \in L. \qquad (2.b)$$

3. Optimal Load Balancing The load balancing strategy which minimizes the mean response time is obtained by solving the optimization problem (1) and (2). For the nodes in the set V_{kj}, we divide them into three types of

A Decomposition Algorithm for Optimal Load Balancing in Tree Hierarchy Networks

Jie LI and Hisao KAMEDA

University of Electro-Communications

nodes, *idle source nodes, active source nodes,* and *neutral nodes* which we denote by Rd_{kj}, Ra_{kj}, and Nu_{kj}, respectively.

Furthermore, here we introduce two delay functions as follows,

$$f_i(\beta_i) = \frac{d}{d\beta_i}\beta_i F_i(\beta_i), \quad g_{ij}(x_{ij}) = \frac{d}{dx_{ij}}x_{ij}G_{ij}(x_{ij}).$$

We note that $f_i(\beta_i)$ is an increasing function. We can define the inverse of the incremental node delay f_i^{-1} by

$$f_i^{-1}(x) = \begin{cases} a, & f_i(a) = x, \\ 0, & f_i(0) \geq x. \end{cases}$$

Theorem 1 *The optimal solution to problem (1) and (2) satisfies the relations,*

$$f_i(\beta_i) \geq \theta_{k-1j} + g_{ij}(x_{ij}), \quad \beta_i = 0, \qquad (i \in Rd_{kj}),$$

$$f_i(\beta_i) = \theta_{k-1j} + g_{ij}(x_{ij}), \quad 0 < \beta_i < \phi_i', \quad (i \in Ra_{kj}), \quad (3)$$

$$f_i(\beta_i) \leq \theta_{k-1j} + g_{ij}(x_{ij}), \quad \beta_i = \phi_i', \qquad (i \in Nu_{kj}),$$

$$\theta_{ki} = \theta_{k-1j} + g_{ij}(x_{ij}), \; if \; i \in Rd_{kj}, \qquad (3')$$

$$\beta_j = f_j^{-1}(\theta_{k-1j}), \qquad (4)$$

(which means that $f_j(\beta_j) = \theta_{k-1j}$, *if* $\beta_j > 0$ *and* $f_j(\beta_j) \geq \theta_{k-1j}$, *if* $\beta_j = 0$, *)*

subject to

$$\sum_{i \in Ra_{kj}} f_i^{-1}(\theta_{k-1j} + g_{ij}(x_{ij})) + \sum_{i \in Nu_{kj}} \phi_i' + f_j^{-1}(\theta_{k-1j}) = \sum_{i \in V_{kj}} \phi_i' + \phi_j - x_{ju}, \qquad (5)$$

$j \in V_{k-1u} \cap B$, $u \in V_{k-2} \cap B$ *or* $u = 0$, $k = 2, 3, ..., p$, *where* θ_{k-1j} *is the Lagrange multiplier.*

4. Proposed Algorithm According to Theorem 1, a decomposition algorithm is presented to solve the problem (1) and (2).

- Decomposition Algorithm

1. Initialization. $r = 0$ (r : iteration number).

   Find $(\boldsymbol{\beta}^0, \boldsymbol{x}^0)$ ($(\boldsymbol{\beta}^0, \boldsymbol{x}^0) \in FS$) as an initial feasible solution to the problem (1) and (2).

2. Solve the subproblem iteratively. $r = r + 1$.

   For $k = 1$ to p do (k: layer number in the tree network)

   begin

   for each j, ($j \in V_{k-1} \cap B$ or j is the root node), apply the K&K[3] algorithm [1] to solve the subproblem as follows,

Min

$$\frac{1}{\Phi}\{\beta_j^{(r)} F_j(\beta_j^{(r)}) + \sum_{i \in V_{kj}} x_{ij}^{(r)} G_{ij}(x_{ij}^{(r)}) + \sum_{i \in V_{kj}} \beta_i^{(r)} F_i(\beta_i^{(r)})\},$$

with respect to the variables $\beta_j^{(r)}$, $\{x_{ij}^{(r)}\}$ and $\{\beta_i^{(r)}\}$, $\forall i \in V_{kj}$,

subject to (2).

end

3. Stopping rule. Compare $D(\boldsymbol{\beta}^{(r)}, \boldsymbol{x}^{(r)})$ and $D(\boldsymbol{\beta}^{(r-1)}, \boldsymbol{x}^{(r-1)})$.

   If $|D(\boldsymbol{\beta}^{(r)}, \boldsymbol{x}^{(r)} - D(\boldsymbol{\beta}^{(r-1)}, \boldsymbol{x}^{(r-1)})| < \epsilon$ then STOP, where ϵ is a proper acceptance tolerance, otherwise, goto step 2.

5. Comparison of Algorithm Performance The decomposition algorithm proposed above (we will call it the DC algorithm) will be compared with the two other algorithms, the FD algorithm and the D–S algorithm. Two performance measures are examined: the storage requirements and the computational time requirements. The storage requirements of the three algorithms are shown in Table 1.

In the numerical experiments, one case with 13 nodes and 3 layers in the tree hierarchy network is considered. We suppose that the node model is M/M/1 and the communication link model also is M/M/1. The numerical results in table 2 show that the DC algorithm has fast convergence in terms of the iteration number and the CPU processing time.

Table 1: Storage requirements of algorithms

Algorithm	Storage Requirement
FD	$O(n)$
D-S	$O(n \log(n))$
DC	$O(n)$

Table 2: Computational Time Requirements

Tolerance	Algorithm	P.T.(I.N.)	M.R.T.
10^{-4}	FD	6.05 (194)	0.07913
	D-S	2.01 (3)	0.07911
	DC	1.20 (5)	0.07908
10^{-6}	FD	3784(57575)	0.07908
	D-S	3.34 (4)	0.07911
	DC	1.79 (7)	0.07908

P.T.: Processing Time(sec.). I.N.: Iteration Number.
M.R.T: Mean Response Time(sec.).

References.

[1] Dafermos, S.C. and Sparrow, F.T. The Traffic Assignment Problem for a General Network, *Journal of Research of National Bureau of Standards-B. Vol.73B, No.2* (1969), pp.91-118.

[2] Fratta, L. and Gerla, M. and Kleinrock, L. The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design, *Networks, Vol.3* (1973), pp.97-133.

[3] Kim, C. and Kameda, H. An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems, *IEEE Trans. on Computers* (to appear).

[1] The K&K algorithm is an effective algorithm solving the static load balancing problem in star network configurations.