일관성에 기초한 적응 부하 평형

김 준 형† 오 하 령†† 이 재 문†††

요 약

부하 평형은 과부하 사이트의 부하중 일부를 다른 사이트로 옮김으로써 분산 처리 시스템의 성능 개선을 꾀한다. 본 논문에서는 source-initiate와 server-initiate 방식을 모두 지원하며 극단적인 예로서 m/m/1 대기행렬(부하 비평형)과 m/m/n 대기행렬(완전한 부하 평형)도 모델화 할 수 있는 부하 평형 방식을 제안하였다. 이 방식은 상태 변수를 각 사이트에 복제하고 각 복사본의 일치성을 완화한다. 그리고 오래된 상태 정보에서 현재의 상태 정보를 유추할 수 있는 완화된 복사본의 일치성 제약을 제안하였다. 또한 스케줄러가 통신을 하지 않고도 부하 평형을 보장할 수 있는 충분조건을 유도하였다. 이 모델을 이용하면 부하 평형의 문제가 상태의 일관성을 유지하는 문제로 바뀌게 되며 기존의 비딩 알고리즘보다 통신량이 적어지게 된다. 이벤트 트레이싱 방법으로 모의 실험하였으며 2개의 극단적인 경우 및 비딩알고리즘과 비교하였다. 그 결과 평균 응답 시간과 통신량에 있어서 비딩알고리즘보다 각각 0-35%, 40-100%의 성능향상을 보였다. 마지막으로 각 시스템 파라미터의 영향을 분석하였다.

Adaptive Load Balancing based on Consistency

Jun Hyung Kim†, Ha Ryoung Oh†† and Jae Moon Lee†††

ABSTRACT

Load balancing attempts to improve the performance of a distributed computing system by transferring some of the workload of a congested site to others. A load balancing scheme that supports both the source-initiated and the server-initiated load balancing is proposed in this paper. It can model both the m/m/1 queue(no load balancing) and the m/m/n queue(perfect load balancing) as the extreme cases. State variables are replicated into every site, and copy consistency constraints are relaxed more weakly. We propose weak copy consistency constraints which correlate the outdated state information to that of the current state. We also derive sufficient conditions under which each scheduler can guarantee the load balancing without communication. Using this model, the problem of load balancing is converted to that of maintaining the consistency of states and communication overhead becomes less than that of the bidding algorithm. The scheme is simulated by event tracing, compared to the two extreme cases and the bidding algorithm. The results show that the mean response time and the number of messages are reduced by 0-35% and 40-100% respectively, compared with the bidding algorithm. Finally the effects of some system parameters are described.

1. Introduction

Load balancing attempts to improve the performance of a distributed computing system (DCS) by smoothing out the periods of high congestion at each site. This is done by transferring some of the workload of a congested site to others. To balance workload between a congested site(C-site) and lightly-loaded site(L-site), the two sites have to establish a migration agreement. The procedure to establish an agreement is called task bidding[1]. Task bidding can be done in three possible ways[2] : ① A C-site sends requests to L-sites(*source-initiated*). ② An L-site sends available messages to C-sites(*server-initiated*). ③ A third party, called an arbitrator, observes and notifies sites to establish a bidding

* He was supported by KOSEF and this work was partially done when he worked his postdoc at Univ. of Twente Nether lands.

† 정 회 원 : 덕성여자대학교 전자계산학과 조교수
†† 정 회 원 : 국민대학교 전자공학과 조교수
††† 정 회 원 : 한성대학교 전산통계학과 전임강사

논문접수 : 1995년 2월 14일, 심사완료 : 1995년 4월 22일

agreement. In fact source-initiated or server-initiated task bidding can be implemented by distributed or centralized control strategies[3]. But one with an arbitrator is essentially a centralized control strategy. A drawback of centralized ones is that the arbitrator can be overloaded and the system can be vulnerable.

A load balancing strategy consists of three component policies[4, 5]. The *information policy* specifies the amount of load and job information made available to the job placement decision makers, and the way by which the information is distributed. The *transfer policy* determines the eligibility of a job for load balancing based on the job and the loads of the hosts. Finally, the *placement policy* decides the hosts to which the eligible jobs should be transferred. The load balancing algorithms can be classified into *static* or *adaptive* ones according to the policies used. Static algorithms[6, 7] may be either *deterministic* or *probabilistic*.

Numerous deterministic algorithms have been proposed[8]. Typically the goal was to find a technique to allocate tasks to sites deterministically so that the total time to schedule all tasks can be minimized. Several algorithms for probabilistic load balancing have been proposed[9, 10, 11]. These can reduce the communication traffic since broadcasting is much seldom done compared to the average arrival rate of jobs. However, they fail to adjust to the fluctuations in the system load since they use only the information on the average behavior of the system.

Adaptive algorithms are to probe the state of other sites when necessary. Livney and Melman [12] showed, using simple queuing network models and simulation, that there is a high probability that at least one site is idle while tasks are queued at some others. It indicates the potential benefit of adaptive load balancing. One of the adaptive algorithms is the bidding algorithm. Major advantages of the bidding algorithm are its generality and extensibility. But the cost of making and acquiring bids may become excessive, and the factors to be used in making bids have not been studied extensively. However many algorithms have been proposed which reduce the costs[13, 14, 15].

In an adaptive load balancing, a scheme which enables the schedulers to obtain a consistent and identical view of the global state of the system should be provided without a unique centralized database. But, since the message passing is the only means of communications(i.e., no global memory), message propagation time is unpredictable and each scheduler has only a limited view of the global state, a considerable communication overhead is required to deduce the global state from partial state informations distributed over the system and to guarantee the consistency of the state. Thus it is important to keep the extra traffic as little as possible.

In this paper a load balancing scheme with fully distributed control strategy is proposed. The approach to reduce the communication overhead is to replicate the partial informations referred to frequently, relax the copy consistency constraint resulting from the replication, and make each scheduler deduce the global state from the replicated informations[16, 17, 18, 19]. In order to treat the distributed load balancing theoretically, we review the consistency and specify the uncertainty of the global state[20, 21]. Based on the uncertainty, a *weak copy consistency* is proposed which requires less communication overhead to maintain the copy consistency and correlates the outdated state information to that of the current state. And sufficient conditions are described under which each scheduler guarantees the consistency without communication. Finally, we discuss the simulation results and the effects of some system parameters.

2. Model of Load Balancing

We deal with the coordination of processes which cooperate and compete to achieve a systemwide common goal in DCS's. At first a general model is formalized which can be applied to several distributed controls and then it is applied to the distributed load balancing.

2.1 Model of Distributed Control

Definition-1 : Associated with the *systemwide common goal* G there is a set of consistency constraints in the form of predicates on the states of G. A consistent state of G, S_G, is an n-tuple satisfying the set of consistency constraints. This is denoted by $B_G(S_G)=1$, where B_G denotes a consistency predicate(Boolean function) for G.

Definition-2 : The DCS for G is described as a tuple, $G(P, X)$, where P is a set of processes cooperating to achieve the goal and X is a set of state variables.

Definition-3 : A state variable $X^i_i \in X$ is the information variable of site N_i. Associated with each X^i_i, a set $Dom(X^i_i)$ represents the domain of X^i_i, which consists of all possible values taken by X^i_i.

For example, X^i_i may represent length of a job queue in the case of load balancing model and $Dom(X^i_i)$ is the set of all non-negative integers in that case. According to Definition-3, a global state is an n-tuple $S_G \in \prod^n_{i=1} Dom(X^i_i)$ where n is the number of sites. Since a state variable is only a partial state of global state, each site can not check the consistency of the global state from state information known to it. That is, the consistency of global state is dependent on the state variables of other sites.

Definition-4 : X^i_i is said to be dependent on X^i_j if the consistency of global state can not be determined without the information of X^i_j. Since B_G is given as a boolean function of the dependent state variables, it is called *dependency constraint.*

X^i_i and X^i_j can be no longer updated independently since X^i_i is constrained by X^i_j and vice versa. If it is found to be inconsistent, the sites cooperate to make it consistent. As a result, when the rate of state changes is relatively high, the communication overhead is considerable and degrades system performance critically. Therefore the following problems are to be solved : ① How can we make the sites to execute more autonomously in order to reduce the communication overhead required to maintain the consistency? ② How can we maintain the consistency constraint with increased autonomy?

Definition-5 : X^j_i is a duplicate of X^i_i stored at(i.e. copied to) N_j where X^i_i is a state variable of N_i ($i, j = 1, \cdots\cdots, n$). Then X^i_i is called the *primary copy* of the state of N_i. X^j_i is called the *secondary copy* of X^i_i stored at(copied to) N_j.

The communication overhead can be reduced by replicating the state variables into every site, even though the secondary copies may be outdated. The replication of state information imposes another problem called copy consistency. The constraint is called a *strict copy consistency constraint* if all secondary copies should be always identical with their primary copy. It is hard to achieve such strict copy consistency in a DCS and a great deal of communication traffic is required to keep it. So, in a distributed database, a weak copy consistency has been suggested such that it is only required that multiple copies must converge to the same final value once all access activities cease[22]. Such a weak constraint is still not enough when the rate of state changes is high.

Definition-6 : X^j_j and X^i_j are said to be

weakly copy consistent, or simply *w*-consistent if $X^i_j \in E_{st_{i,i}}(X^i_j)$, where $E_{st_{i,i}}(X^i_j) \subset Dom(X^i_j)$ is an *estimation function* and represents the *uncertainty of state* in DCS's.

Estimation of X^i_j from X^i_j can be represented as a mapping from X^i_j to the range of states, in which X^j_j would be included. Thus, the estimation function of X^i_j maps $Dom(X^i_j)$ to the power set of $Dom(X^i_j)$. By using the function, the true value is not known exactly, but it is known that it is in some range. In fact, if X^j_j and X^i_j are w-consistent, then X^i_j can be treated as approximate information about X^j_j known to N_i. In other words, X^j_j can be estimated from X^i_j, and X^j_j is said to be estimable from X^i_j. Thus, the w-consistency is another form of consistency constraints. The global state is w-consistent, if each pair of the primary and secondary copies are w-consistent. Notice that the strict copy consistency in database is a special case of the w-consistency such that $E_{st_{i,i}}(X^i_j) = \{X^i_j\} = \{X^j_j\}$.

The w-consistency may give us a chance to make each site estimate the global state more precisely from outdated information. It also reduces the communication overhead required to maintain the consistency. One of our main ideas is to replicate state variables into every site, and relax copy consistency constraints more weakly. The other is to make each site have a deduction mechanism which can deduce the global state and guarantee the consistency of the global state from the outdated informations.

Definition-7 : The *global deduction mechanism* M consists of a set of *local deduction mechanism* M_i at each site. M_i at N_i is defined as a Cartesian product of estimation functions,

$$M_i(X^i) = \prod_{X^i_j \in X} Est(X^i_j) \qquad \text{(Eq. 1)}$$

where X^i represents the global state known to the site N_i.

Definition-8 : A possible global state s, $s \in M_i(X^i)$, is called an *estimated global state*. And s is said to be consistent if and only if it satisfies B_G. If the true global state is in $M_i(X^i)$, then M_i is said to be correct.

Lemma-1 : If each state information $X^j_j (1 \le j \le n)$, $X^i_j \in X^i$, is w-consistent with its primary copy X^j_j at N_n then the local estimation M_i of global state is correct.

Proof : If X^i_j is w-consistent with X^j_j, then $X^j_j \in Est(X^i_j)$ for all $1 \le i \le n$. So the true global state$(X^1_1, X^2_2, \cdots\cdots, X^n_n) \in \prod_{X^i_j \in X^i} (X^i_j)$.

Lemma-2 : The global state satisfies B_G when M_i is correct and all its estimated global state $s \in M_i(X^i)$ are consistent.

Proof : Suppose all $s \in M_i(X^i)$ are consistent. From Lemma-1, the current global state S_G is included in the local estimation, i.e., $S_G \in M_i(X^i)$, if the estimation is correct.

In practice, when a state variable is changed and a site does not have informations about others dependent on it, it has to communicate all other sites to check the consistency. However, if it has some informations on dependent state variables, it can check the consistency of the new state by communicating only with the sites whose state variables are suspicious to be inconsistent. Furthermore, if all estimated global states are consistent, then no communication may be required. In the next section, we will apply the estimation mechanism for the load balancing. Also, we will find sufficient conditions under which each site can find the consistency of the global state with less communication overhead.

2.2 Model of load balancing

We at first describe the model of load balancing over two sites and then extend it to the case over arbitrary n sites.

Definition-9 : A load balancing system over two sites N_1 and N_2(LB2) is described as a tuple(S, X), where S is the set of schedulers S_1 and S_2, X is the set of state variables composing the global state such as load indices X^1_1 and X^2_2.

The estimation of load index is so difficult that there is no completely satisfactory solution. Some load indices are discussed in [15, 23]. But the time average of the queue length has a correlation of nearly 1 with the job response times of CPU-intensive jobs[24]. So it is used as the load index in this paper. Therefore X^i_i represents the queue length of the site N_i.

Definition-10 : Two sites N_1 and N_2 are *load balanced* or *consistent* if the following inequality satisfies between the load states of the sites :

$$| X^1_1 - X^2_2 | \leq D \qquad \text{(Eq. 2)}$$

where D is *load imbalance factor* which can be tuned according to the system parameters. We call(Eq. 2) the *global consistency constraint*(GCC) of LB2.

Since X^1_1 and X^2_2 are constrained by GCC, they can not be updated independently. Therefore, when a state change occurs due to the arrivals and/or departures of jobs, the commit action for the state change must be deferred until the new state proves to be consistent. If it is found to be inconsistent, the schedulers cooperate to make the state consistent. Consequently, the problem of LB2 is converted to that of maintaining GCC.

Definition-11 : The estimation function for LB2 is defined as follows :

$$Est(X^2_1(t)) = \{x_K \mid x_K \in Dom(X^1_1),$$
$$| X^2_1(t) - x_K | \leq D_1\}, \text{ and} \qquad \text{(Eq. 3.a)}$$
$$Est(X^1_2(t)) = \{x_K \mid x_K \in Dom(X^2_2),$$
$$| X^1_2(t) - x_K | \leq D_2\}, \text{ where } D_1 + D_2 = D. \qquad \text{(Eq. 3.b)}$$

Definition-12 : The *weak copy consistency constraints*(WCC's) of LB2 is defined as follows :

$$| X^1_1 - X^2_1 | \leq D_1, \qquad \text{(Eq. 4.a)}$$
$$| X^1_2 - X^2_2 | \leq D_2, \text{ where } D_1 + D_2 = D. \qquad \text{(Eq. 4.b)}$$

In order to estimate X^j_j from X^i_j, X^j_j and X^i_i must satisfy WCC's. By the replication of the state variables and WCC's, each scheduler has a partially correct view of the global load state. As a result, it can estimate the current global state from the outdated information known to itself, but it is not able to make a decision whether GCC is satisfied or not.

Definition-13 : The *local copy consistency constraints* (LCC's) of LB2 is defined as follows :

$$| X^1_1 - X^1_2 | \leq D_1 \text{ at } N_1 \text{ and} \qquad \text{(Eq. 5.a)}$$
$$| X^2_1 - X^2_2 | \leq D_2 \text{ at } N_2 \text{ where}$$
$$D_1 + D_2 = D. \qquad \text{(Eq. 5.b)}$$

Each scheduler may estimate the global state more precisely under LCC's. We can derive the conditions which can be used in LB2 by relaxing the strict copy consistency to WCC's and replacing GCC with the above four inequalities.

Lemma-3 : GCC always satisfies, if(Eq. 4.a), (Eq. 4.b), (Eq. 5.a), and (Eq. 5.b) satisfy.

Proof : If we add both sides of(Eq. 4.a) and (Eq. 5.b), it becomes the sufficient condition to meet GCC. And if we add both sides of(Eq. 4.b) and (Eq. 5.a), it also becomes the sufficient condition to meet GCC.

When copies are w-consistent, any state change at a site can be committed immediately without violation of GCC if the new state satisfies LCC at the site, similarly with the safety conditions in lower-upper bound couples[19]. LCC makes each scheduler communicate only when the consistency of the global state is suspi-

cious. Consequently, the problem of load balancing with GCC can be decomposed into those of maintaining WCC's and deducing the current global state from the outdated state information by using LCC's.

The technique used in the case of two sites can be extended easily to the case over n sites($n>2$). In case of arbitrary n sites, when a state change occurs at a site, the scheduler at that site can check the consistency of the global state by communicating only with the schedulers whose secondary copy does not satisfy LCC's.

Definition-14 : In the case of load balancing over arbitrary n sites(LBn), GCC(Eq. 2), uncertainty(Eq. 3), WCC-s(Eq. 4) and LCC's(Eq. 5) defined in the case of LB2 can be extended respectively to the following inequalities :

$$| X^i_i - X^j_j | \le D, \text{ where } 1 \le i, j \le n, j \ne i, \quad \text{(Eq. 6)}$$

$$Est(X^i_i(t)) = \{x_K \mid x_K \in Dom(X^i_i), \ | X^j_i(t) - x_K | \le D_i\}, \text{ where } D_i \le D \ 1 \le i, j \le n, j \ne i, \quad \text{(Eq. 7)}$$

$$| X^i_i - X^j_i | \le D_i, \text{ where } 1 \le i, j \le n, j \ne i, \text{ and} \quad \text{(Eq. 8)}$$

$$| X^i_i - X^j_j | \le D_i, \text{ where } D_i + D_j = D, \ 1 \le i, j \le n, j \ne i. \quad \text{(Eq. 9)}$$

We call the inequality(Eq. 6) *global consistency constraint* of load balancing over the corresponding two sites. Thus, the global state is said to be *load balanced* or *consistent* only when GCC holds for any pair of sites. Lemma-3 can be extended to LBn.

Theorem-4 : When the global state is w-consistent, the global state is always load balanced if LCC holds at each site.

Proof : For a pair of sites N_i and N_j they must be load balanced, by Lemma-1, if the load information of the two sites are w-consistent and LCC-s hold at both sites. Therefore, when the global state is w-consistent, if LCC holds at each site, then any pair of two sites are load balanced. This proves the theorem.

The scheme for LBn is very similar to that of LB2, except the multiple candidates for job migration. Thus in the case of arbitrary n sites, only placement policy becomes more complex. Also, the likelihood of concurrent state changes may cause *thrashing* that a lightly loaded site may become heavily loaded due to the jobs migrated from multiple sites to that site. This difficulty may be cleaned by some reservation scheme.

3. Experiments

3.1 Simulation model

In this section the model of simulation is described. Our measures of performance are mean response time and average number of messages exchanged during a negotiation as a function of various parameters such as job arrival rate λ, the average communication cost T_C, the average cost of job transfer T_m and the imbalance factor D.

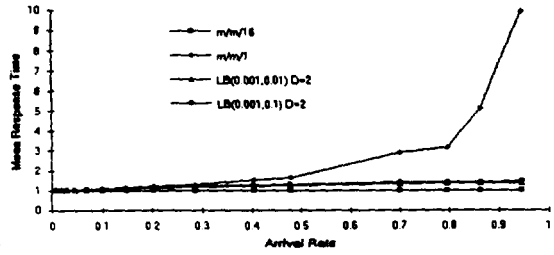
The comparison is potentially difficult because of a large number of parameters involved and the non-deterministic behavior of the scheme. So, the performance comparison is studied by event-tracing simulation. We assume that jobs arrive at each site according to the Poisson distribution with arrival rate λ, and job service time is Gaussian distributed with average service time $S=1$ and its deviation $q=0.5$. 2000 jobs are simulated for each scheme with parameters(T_C, T_m, λ, D) where the number of sites is 16. The arrival rate is varied from 0.01 to 0.9. For simplicity, T_C and T_m between any two schedulers is assumed to be constant values($T_C=0.001S$, $0.01S$ and $T_m=0.01S$, $0.1S$). Job transfer costs higher than 10% of average service time would be infrequent. We can expect that in any practical implementation of load balancing with a relatively high ratio of processing cost to transfer cost would be selected

for migration. One can easily imagine more efficient protocols. The advent of systems based on file servers or networkwide file systems(e.g., NFS and RFS) will further decrease the cost of job transfer. It is assumed that the communication network is reliable and each site processes the jobs as they arrive at the site one by one (FCFS).

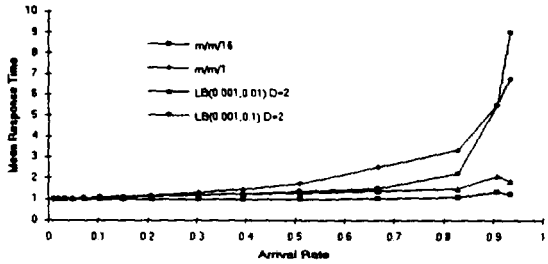
3.2 Simulation results

In this section the performance of our scheme is analyzed and compared to the two extreme cases : no load balancing and perfect load balancing with no cost. Our scheme is represented by LB(T_G, T_m), where T_G and T_m is normalized to the mean service time($S=1$), the no load balancing by 16 independent m/m/1 queues and the perfect load balancing by an m/m/16 queue respectively. (Fig. 1) shows the mean response time versus job arrival rate λ, where the number of sites is 16. It shows that the mean response time is heavily dependent on T_G and T_m. When they are negligible or the sites are lightly loaded, the result is almost close to that of m/m/16. As λ increases, the load state at each site changes more frequently and the informations on other sites would be outdated. As a result, the probability that LCC's are satisfied decreases, and the communication traffic increases in order to gather global load state information. Also, the frequent state changes increase the probability of load imbalance during a short period, and thus the probability of job migration. Consequently, as λ increases, the communication overhead increases abruptly, and the performance degrades. However, one remarkable result is shown in(Fig. 2) which shows the average response time versus job arrival rate in the case that jobs arrive at only 8 sites among 16 sites. In this figure, even though T_m is relatively large, the mean response time can be dramatically decreased compared to m/m/1. We expect that our scheme can be adapted to the DCS's such as a lot of workstations interconnected by a LAN.

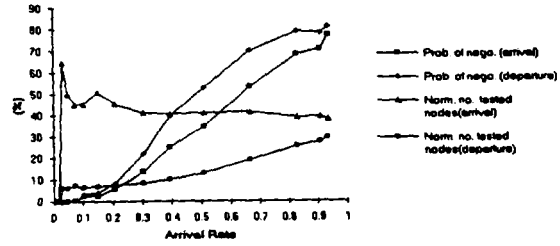
(Fig. 3) shows the mean probability of negotiation between schedulers and the average number of tested sites when a negotiation occurs due to an event such as job arrival and/or departure. It shows that only about 40% of jobs arrived at each site are assigned through negotiation even when the job arrival rate λ is up to 0.6, and that the rest of them are directly assigned to that site without negotiation, that is, without communication. Even though λ is higher, about 20%



(Fig. 2) Comparison of mean response time vs.arrival rate when job arrives at only 8 sites among 16 sites



(Fig. 1) Comparison of mean response time vs. arrival rate($n=16$)



(Fig. 3) Mean probability of negotiation and mean number of tested sites(n=16, D=2)

of jobs are directly assigned without negotiation. As a result, our scheme could reduce the overhead compared to the bidding algorithm.

It also shows that about 7 sites, not all, are tested when $\lambda=0.6$ due to an arrival of a job. It is a little more than 40% of all the sites. It shows that more sites are tested as the job arrival rate increases and the average number of tested sites due to an arrival of a job is smaller than that due to a departure of a job.

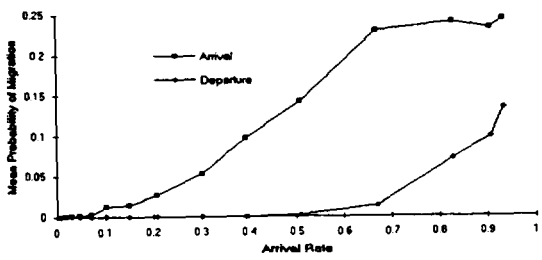
(Fig. 4) shows the probability of job migration when a new job arrives/departs at a site. It implies that the probability of load imbalance during short period increases when λ is high. The probability of migration of an already assigned job increases abruptly, when λ exceeds some threshold. It implies that when λ is relatively high, the server-initiated scheme is more desirable than the source-initiated scheme. From the results, the load balancing scheme which can support both of the source-initiated and the server-initiated schemes is desirable, especially when λ is relatively high.

We can expect that a large D reduces the communication overhead, since it increases the probability of satisfying LCC's and WCC's for each load state change. As a result, a larger D will give better performance when all sites are heavily loaded and T_C and T_m are relatively large(i.e DCS with relatively slow communication channel). However, since a larger D allows more load imbalance, it will decrease the performance when the job arrival rate is low or T_C and T_m are relatively small.(Fig. 5) and(Fig. 6) prove our observation.(Fig. 5) shows an effect of tunable parameter D with respect to system parameters such as T_C and T_m. When T_C and T_m is relatively large and arrival rate is high, i. e. communication subnet is relatively slow, a large D is desirable. That means large imbalance with less communication is desirable in such an environment.(Fig. 6) shows that when a job arrives, the probability of negotiation decreases and the negotiation requires smaller number of tested sites as D increases. D can be tuned according to the system parameters such as T_C, T_m and n. Futhermore, it can be accommodated adaptively according to the system workloads.

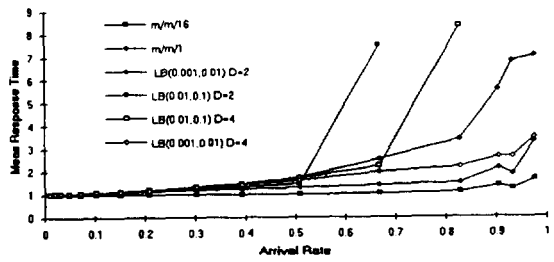
In fact, the no load balancing is one end of the spectrum of our scheme where D is infinite, and the perfect load balancing is the other end where $D=1$, and T_C and T_m are zero.

3.3 Comparison with the bidding algorithm

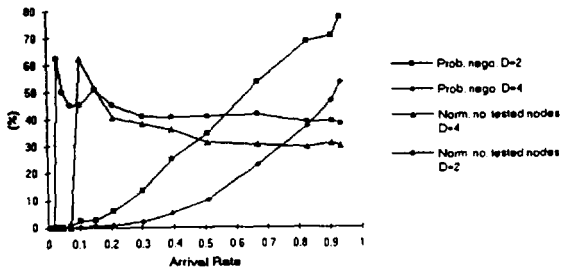
In this section, the performance of the pro-



(Fig. 4) Mean probability of job migration



(Fig. 5) Effect of system parameters such as D, Tc, Tm on response time



(Fig. 6) Mean probability of negotiation and mean number of tested sites(n=16)

posed algorithm is compared with that of the bidding algorithm with respect to the mean response time, the mean number of messages per job and the mean probability of migration. The model of simulation is the same with the previous section but T_C is fixed to 0.001S. The results are summarized in 〈Table 1〉, (Fig. 7) and (Fig. 8), where the definition of gain is given in(Eq. 10).

〈Table 1〉 Results of performance comparison with the bidding algorithm

arrival rate (λ)	Bidding algorithm						Proposed scheme(D=2)					
	T_c=0.001S, T_m=0.01S			T_c=0.001S, T_m=0.1S			T_c=0.001S, T_m=0.01S			T_c=0.001S, T_m=0.1S		
	MRT	MNM	MPM	MRT	MNM	MPM	MRT	MNM	MPM	MRT	MNM	MPM
0.010	1.005	30.0	0.000	1.005	30.0	0.000	1.004	0.0	0.000	1.004	0.0	0.000
0.048	1.032	30.0	0.002	1.032	30.0	0.002	1.029	0.1	0.002	1.029	0.1	0.002
0.103	1.078	30.0	0.012	1.078	30.0	0.013	1.073	0.5	0.012	1.075	0.4	0.012
0.207	1.138	30.0	0.029	1.145	30.0	0.030	1.127	1.0	0.026	1.132	1.0	0.027
0.303	1.204	30.0	0.056	1.217	30.0	0.059	1.188	2.4	0.054	1.200	2.5	0.057
0.394	1.252	30.0	0.106	1.277	30.0	0.115	1.228	4.5	0.097	1.254	4.7	0.104
0.509	1.321	30.0	0.156	1.360	30.0	0.169	1.290	6.9	0.148	1.332	7.2	0.155
0.668	1.422	30.0	0.259	1.572	30.0	0.287	1.378	11.6	0.251	1.484	12.5	0.284
0.829	1.705	30.0	0.315	3.057	30.0	0.350	1.504	15.2	0.323	2.243	16.7	0.365
0.907	2.760	30.0	0.354	8.362	30.0	0.383	2.114	16.6	0.368	5.510	18.1	0.405

MRT : Mean response time
MNM : Mean number of messages per job
MPM : Mean probability of job migration

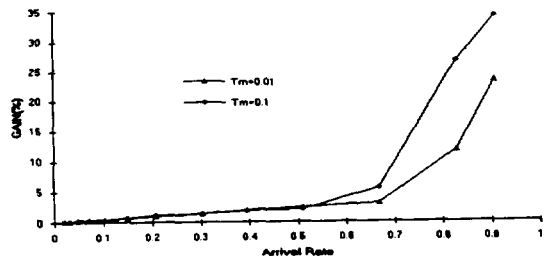
$$GAIN(\%) = \frac{Performance_{bidding} - Performance_{proposed}}{Performance_{bidding}} \times 100 \qquad \text{(Eq. 10)}$$

We can observe that the mean response time of the proposed algorithm is comparable to that of the bidding algorithm when λ is low but the proposed algorithm outperforms the bidding algorithm as λ increases. The comparable performances at low arrival rate is due to the fact that it is very hard for a site to be overloaded and so load balancing schemes have almost no effects at low arrival rate as shown in (Fig. 1). But when λ becomes high, the overhead of messages becomes critical, so the gain of the proposed algorithm becomes large because the number of messages is much smaller than that of the bidding algorithm. The gain will be much greater when T_C becomes larger by the same reason.

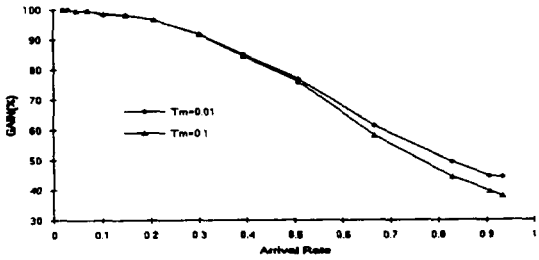
By using the replication of state information and deduction mechanism, the proposed algorithm reduces the number of messages considerably compared to the bidding algorithm as shown in (Fig. 8). Finally 〈Table 1〉 shows that the mean probability of job migration is lower than the bidding algorithm when λ is low but it becomes higher as λ increases. It is because our scheme permits more load imbalance than the bidding algorithm and source-initiated scheme is dominant when λ is low. But the probability of migration becomes higher as λ increases since the proposed scheme can support both the source-initiated and the server-initiated load balancing.

4. Conclusion

Load balancing attempts to improve the performance of a DCS by smoothing out the periods of high congestion at each site. This is done by transferring some of the workload of a congested site to others. A load balancing scheme that supports both the source-initiated and the server-initi-



(Fig.7) Gain of mean response time over the bidding algorithm



(Fig. 8) Gain of mean number of messages per job over the bidding algorithm

ated load balancing is proposed in this paper. It can model both the no load balancing and the perfect load balancing as the extreme cases. By defining the global copy consistency, the problem of load balancing is converted to that of maintaining the global state consistent between schedulers. State variables are replicated into every site, and copy consistency constraints are relaxed more weakly. We propose *weak copy consistency constraints* which correlate the outdated state information to that of the current state. We derive the sufficient conditions under which each scheduler can guarantee the load balancing without communication.

The scheme is simulated by event tracing, compared to two the extreme cases and the bidding algorithm. And the effects of some system parameters described. We have shown that our scheme requires less communication overhead and shorter mean response time than the bidding algorithm, by estimating the global state from the outdated state information. The simulation results show that the mean response time and the number of messages of the proposed algorithm are reduced by 0-35% and 40-100% respectively, compared with the bidding algorithm. We expect that our scheme can be adapted to the distributed systems such as a lot of workstations interconnected by a local area network. The problem of tuning and/or adaptation of D with the parameters T_G, T_m, n and λ should be studied further.

References

[1] B. W. Wah and J. Y. Juang, "Resource Scheduling for Local Computer Systems with Multiaccess Networks," *IEEE Tr. Computers*, Vol. C-34, No. 12, pp. 1144-1157, Dec. 1985.

[2] 류재철, "분산시스템에서의 부하균형을 위한 알고리즘," 한국정보과학회 논문지, Vol. 20, No. 3, pp. 430-441, March 1993.

[3] H. C. Lin and C. S. Raghavendra, "A Dynamic Load Balancing Policy with a Central Job Dispatcher(LBC)," *IEEE Trans. Software Eng.*, Vol. 18, No. 2, pp. 148-158, Feb. 1992.

[4] S. Zhou, "A trace-driven simulation study of dynamic load balancing," *IEEE Trans. Software Eng.*, Vol. 14, No. 9, pp. 1327-1341, Sept. 1988.

[5] P. Kureger and N. G. Shivaratri, "Adaptive Location Policies for Global Scheduling," *IEEE Trans. Software Eng.*, Vol. 20, No. 6, pp. 432-444, June 1994.

[6] 임경수, 김수정, 김종근, "스타형 컴퓨터 네트워크의 부하균형 방향정책," 한국정보처리학회 논문지, 제1권, 제4호, pp. 427-437, Nov. 1994.

[7] C. G. Kim and H. Kameda, "An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems," *IEEE Trans. Comp.*, Vol. 41, No. 3, pp. 381-384, March 1992.

[8] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *Journ. of the ACM*, Vol. 32, April 1985.

[9] J. A. Stankovic, "An application of bayesian decision theory to decentralized control of job scheduling," *IEEE Trans. on Computers*, Vol. C-34, No. 2, pp. 117-130, Feb. 1985.

[10] T. C. K. Chou and J. A. Abraham, "Distributed control of computer systems," *IEEE Trans. on Comput.*, Vol. C-35, No. 6, June 1986.

[11] C. Gao, J. W. S. Liu and M. Railey, "Load balancing algorithms in homogeneous distributed systems," Tech. Report No. *UIUCDCS-R-84-1163*, Dept. of CS., Univ. of Illinois at Urbana Champaign, 1984.

[12] M. Livny and M. Melman, "Load balancing in homogeneous broadcast distributed systems," *ACM Comput. Network Performance Symp.*,

pp. 47-55, 1982.

[13] D. L. Eager, E. D. Lazowsaka, and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems," *IEEE Trans. on Software Eng.*, Vol. SE-12, No. 5, pp. 662-675, May 1986.

[14] Y. T. Wang and R. J. T. Morris, "Load sharing in distributed systems," *IEEE Trans. on Comput.*, Vol. C-34, No. 3, pp. 204-217, March 1985.

[15] L. M. Ni, C. Xu and T. B. Gendreau, "A Distributed Drafting Algorithm for Load Balancing," *IEEE Trans. Software Eng.*, Vol. SE-11, No. 10, pp. 1153-1161, Nov. 1985.

[16] J. P. Verjus, "Synchronization in distributed systems : An informal introduction," Distributed Computing Systems, Academic Press, pp. 3-22, 1983.

[17] L. Sha, E. D. Jensen, et. al., "Distributed cooperating processes and transactions," Distributed Computing Systems, Academic Press, pp. 23-50, 1983.

[18] L. Sha, *Modular concurrency and failure recovery-Consistency, correctness and optimality*, Ph.D Dissertation, Dept. of CS, CMU, 1985.

[19] O. S. F. Carvalho and G. Roucairol, "Assertion, decomposition and partial correction of distributed control algorithms", *Distributed Computing Systems*, Academic Press, pp. 67-92, 1983.

[20] J. H. Kim, K. H. Park and M. H. Kim, "A model of distributed control : dependency and uncertainty," *Information Processing Letters*, Vol. 30, No. 2, pp. 73-77, Jan. 1989.

[21] S. S. Lee, H. R. Oh, J. H. Kim, W. H. Chung and M. Kim, "A Distributed Mutual Exclusion Algorithm Based on Weak Copy Consistency," *IEICE Trans. Inf. Syst.*, Vol. E75-D, No. 3, pp. 298-306, May 1992.

[22] H. Garcia-Molina and G. Wielderhold, "Read-only transactions in a distributed database," *ACM Trans. on Database Syst.*, Vol. 7, pp. 186-213, June 1982.

[23] T. Kunz, "Influences of Different Workload Descriptives on a Heuristci Load Balancing Scheme," *IEEE Trans. on Software Eng.*, Vol. 17, No. 7, pp. 725-730, July 1991.

[24] S. Zhou, "An experimental assesment of resource queue length as load indices," in *Proc. Winter USENIX Conf.*, Washington, DC, pp. 73-82, Jan. 1987.

김 준 형

1979년 서울대학교 전자공학과 졸업(공학사)
1984년 한국과학기술원 산업전자공학과(공학석사)
1989년 한국과학기술원 전기 및 전자공학과(공학박사)
1984~91년 한국데이타통신(주) 행정망용 주전산기 안정화팀장
1991~92년 한국전산원 주전산기 표준 연구위원
1992년~현재 JTC/SC21 Internationalization 전문위원
1991년~현재 덕성여자대학교 전산학과 조교수
관심분야 : 병렬 및 분산처리, 운영체제, 화일시스템.

오 하 령

1983년 서울대학교 전기공학과 졸업(공학사)
1988년 한국과학기술원 전기 및 전자공학과(공학석사)
1992년 한국과학기술원 전기 및 전자공학과(공학박사)
1983~86년 삼성전자 종합연구소 연구원
1992년~현재 국민대학교 전자공학과 조교수
관심분야 : 병렬 및 분산처리, 운영체제, 멀티미디어.

이 재 문

1986년 한양대학교 전자공학과 졸업(공학사)
1988년 한국과학기술원 전기 및 전자공학과(공학석사)
1992년 한국과학기술원 전기 및 전자공학과(공학박사)
1992~94년 한국통신 연구개발단 선임연구원
1994년~현재 한성대학교 전산통계학과 전임강사
관심분야 : 병렬 및 분산처리, 데이타베이스, 정보 검색.