Memory Conscious Scheduling for Cluster-based NUMA Multiprocessors

Takahiro Koita†, Tetsuro Katayama††, Keizo Saisho †† and Akira Fukuda††
†：Osaka Sangyo University
††：Nara Institute of Science and Technology

This paper proposes several policies for cluster-based NUMA multiprocessors that are combinations of a processor scheduling scheme and a page placement scheme and investigates the interaction between them. The simulation results show that policies that cooperate to employ the home-cluster concept achieve the best performance. The paper also compares the best of the proposed policies with other existing dynamic processor scheduling policies. Based on our study reported here, the best policy is found to perform better than other existing policies.

クラスタ型 NUMA マルチプロセッサにおける メモリ協調スケジューリング方式

小板 隆浩†, 片山 徹郎††, 最所 圭三††, 福田 晃††
†：大阪産業大学 工学部 情報システム工学科
††：奈良先端科学技術大学院大学 情報科学研究科

本研究では，クラスタ型の NUMA マルチプロセッサを対象として，ホームクラスタという概念をもとにプロセッサスケジューリング方式とページプレースメント方式が協調的に動作する手法について提案する．プロセッサスケジューリング方式とページプレースメント方式をそれぞれ独立なものとしてではなく，両方式を組み合わせたものを 1 つのポリシとして扱う．両方式のいくつかの組み合わせを考え，提案方式の有効性をシミュレーションにより評価する．シミュレーション結果から，プロセッサスケジューリング方式とページプレースメント方式がホームクラスタを中心に協調動作する方式は，他の方式に比べリモートアクセスのオーバヘッドが少なく，平均応答時間が短くなる．さらに，提案方式を従来の動的なプロセッサスケジューリング方式と比較し，本提案方式がすべての負荷条件において，最も効率良くプロセスを実行し，平均応答時間を改善することを示した．

1 Introduction

Non-Uniform Memory Access (NUMA) multiprocessors have great potential for achieving high performance. These machines are frequently used as compute servers with multiple parallel processes executing at the same time.

For such multiprogrammed environments, the processor scheduling scheme, which is responsible for processor allocation, significantly affects the efficiency of process execution. In addition, the page placement scheme, which is responsible for page allocation, is also important because the memory access cost is non-uniform due to distributed memory modules. This non-uniformity of memory access cost is an almost inevitable feature of NUMA multiprocessors. Ideally, it is preferable that virtual pages accessed by a processor are local the processor (in the same cluster) to avoid the higher cost of remote memory access. Therefore, the processor scheduling scheme and the page placement scheme should work together to assure this locality of memory access.

In this paper, we propose several policies for cluster-based NUMA multiprocessors(Figure1) that consider the combinations of a processor scheduling scheme and a page placement scheme. We introduce the concept of a home-cluster, which is the basis of some of our proposed policies.

2 Proposed Schemes

In this section, we propose several processor scheduling schemes and page placement schemes and consider their combinations.

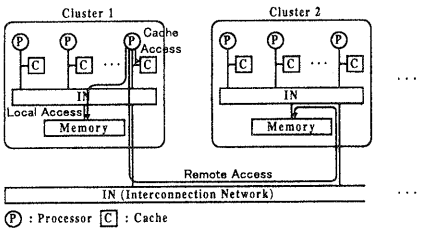

Figure 1: Cluster-based NUMA multiprocessor model

2.1 Home-Cluster

Before describing each scheme, we need to introduce the concept of a home-cluster. The basic idea of introducing the home-cluster is to reserve one cluster for one process to allocate processors and physical pages in the cluster. Reserving a cluster allows only one process to use the processors and the memory module in the cluster. We call the cluster reserved for a process the home-cluster of the process.

2.2 Processor Scheduling

We employ two-level scheduling as the basic scheduling framework [2, 4], which consists of a kernel scheduler and user schedulers. We focus on the schemes employed by the kernel scheduler. The user schedulers employ the First-Come First-Served (FCFS) scheme. The processor scheduling scheme determines processor allocation/deallocation to/from processes and threads in the processes. Here we explain the common scheduler actions for all processor scheduling schemes.

When a process is created or threads in a process are forked, the process issues a processor-allocation request to the kernel scheduler if not enough processors are allocated to the process. After receiving the request, the kernel scheduler tries to allocate processors according to the following base priority. In this priority, a "free processor" means a processor that is not allocated to any process, and a "before-processor" of a process means a free processor that was allocated to the process before the latest deallocation.

[Base priority]

1. The before-processors of the process
2. Free processors in the cluster having the most processors allocated to the same process
3. Free processors in the cluster having the most before-processors of the process
4. Free processors in the cluster having the most free processors

From the viewpoint of processor allocation rather than processor deallocation, we consider two classes of processor scheduling schemes: the cluster-free scheme and the home-cluster scheme. We also consider three types of cluster-free schemes and two types of home-cluster schemes.

2.2.1 Cluster-Free Schemes

The cluster-free scheme is the simplest implementation of the dynamic processor scheduling scheme. Under the cluster-free scheme, the scheduler does not consider the locations of processors. Therefore, all free processors can be allocated to any process according to the base priority described above. We consider three types of cluster-free schemes with different actions for processor deallocation.

(a) Release Scheme

The release scheme is the simplest form of the cluster-free scheme. All free processors can be reallocated to any process. In addition to free processors, idle processors can also be reallocated.

(b) Hold Scheme

The hold scheme is the same as the release scheme except for how it handles idle processors. All free processors can be allocated to any process in the same way as the release scheme. However, under the hold scheme idle processors are not reallocated to other processes but are kept for the process to which the processors were first allocated.

(c) Partial-Hold Scheme

The partial-hold scheme has intermediate characteristics between the release and hold schemes. Under the partial-hold scheme, when a processor allocated to a process becomes idle because there are no runnable threads in the process, the processor is not released from the process if the processor belongs to the home-cluster of the process. This works in the same way as the hold scheme. However the processor does not belong to the home cluster of the process, the processor is released and free for use, as in the release scheme.

2.2.2 Home-Cluster Schemes

The following two schemes are based on the concept of the home-cluster. Focusing on the home-cluster implies that the scheduler action of the home-cluster scheme is different from that of the cluster-free scheme for allocation; processor allocation priority under the home-cluster schemes is different from that under the cluster-free schemes using the base priority only.

The home-cluster scheme employs the concept of the home-cluster for processor allocation. On the other hand, the cluster-free schemes do not take into account the notion of the home-cluster in allocation.

(d) Basic-Home Scheme
The basic-home scheme is the simplest form of the home-cluster scheme. Under the basic-home scheme, free processors are allocated to a process according to the priority described above; first free processors in the home-cluster of the process and second free processors in the base priority except ones belonging to other home-clusters. Thus, free processors in the home-cluster are given the highest priority as candidates for processor allocation.

(e) Slide Scheme
The slide scheme is the same as the basic-home scheme except for thread handling when a processor in the home-cluster completes the execution of a thread in the owner process and becomes idle. In this case, if there is a thread in the owner process running on another cluster rather than on the home-cluster, the thread is migrated to the idle processor. The slide scheme prefers using the home-cluster more than the basic-home scheme does. Thus, the processors in the home-cluster are more likely to be used.

2.3 Page Placement

Here, we describe the alternatives for the page placement scheme, which determines which physical memory module will be used to load a fetched page. We consider the following two schemes for the page placement scheme.

(a) Distributed Scheme
Under the distributed scheme, when a page fault occurs at a processor, the virtual page is loaded into the memory module of the cluster to which the processor belongs.

(b) Concentrated Scheme
In contrast to the distributed scheme, the concentrated scheme is designed with the notion of the home-cluster. All virtual pages of a process are loaded into the memory module in the home-cluster of the process. Under this scheme, if a thread in a process is executed on a processor that does not belong to the home-cluster of the process, the thread causes remote memory access. However, by migrating this thread to the home-cluster, this kind of remote access can be avoided.

2.4 Summary of Policies

By combining the processor scheduling schemes and the page placement schemes described above, the following ten policies are obtained (Table 1).

Table 1: Summary of policies

Processor Scheduling / Page Placement	Cluster-Free Scheme			Home-Cluster Scheme	
	Release (FR)	Hold (FH)	Partial-Hold (FP)	Basic-Home (HB)	Slide (HS)
Distributed (Dist)	FR-Dist	FH-Dist	FP-Dist	HB-Dist	HS-Dist
Concentrated (Con)	FR-Con	FH-Con	FP-Con	HB-Con	HS-Con

3 Experiments

Here we evaluate the proposed policies by simulation experiments. Through all simulation experiments, the hardware configuration is fixed: 64 processors and 16 clusters. The ration of memory access cost is a low type $((O_{cache} : O_{local} : O_{remote}) = (1 : 1.3 : 4.8))$. O_{cache}, O_{local} and O_{remote} denote the memory access cost for cache, local memory and remote memory respectively. More details of the parameters can be seen in [4].

3.1 Comparison of policies

We first examine the ten combination policies under a low access cost ratio: $(O_{cahce} : O_{local} : O_{remote}) = (1 : 1.3 : 4.8)$. Figure 2 shows the mean response time versus load level, respectively. The load level, which is calculated from a process creation interval, is an index value used to estimate system load. If a process creation interval is used as the x-axis, identifying whether the system load is low or high is difficult. A load level beyond 100% does not always mean that the system is congested.

Figure 2 shows that when the load level is low, the difference between the policies is small. As the load level increases, the response time with the home-cluster policies (HB, HS) becomes shorter than the mean response time with the cluster-free policies (FH, FR, FP). Among the cluster-free policies, the difference between the concentrated (Con) policies and the distributed (Dist) policies is small. In contrast to the cluster-free policies, among the home-cluster policies, the concentrated policies (HB-Con, HS-Con) give better performances than the distributed policies (HB-Dist, HS-Dist). The HS-Con gives the smallest mean response time among all combination policies.

Figure 3 shows the mean response time versus the average number of threads forked at a time, where the load level is 75%. When the number of forked threads is small, the difference among the policies is small except for HS-Con and HS-Dist. The HS-Con policy gives the smallest mean response time over the entire range of the number of forked threads. With HS-Con, the processor scheduling and page placement schemes are designed to implement the

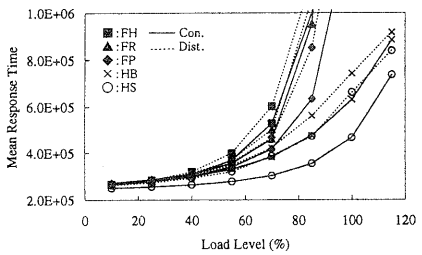

Figure 2: Mean response time vs. load level with low access cost ratio

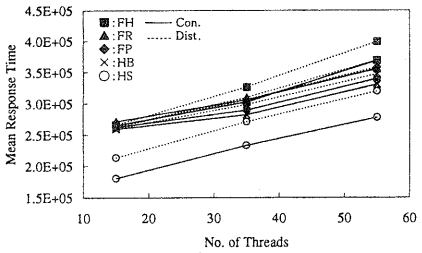

Figure 3: Mean response time vs. the number of forked threads with low access cost ratio

home-cluster concept cooperatively. The collaboration of the schemes reduces the remote access overhead and reduces the mean response time even when the number of forked threads is changed.

3.2 Comparison of best policy with other existing policies

We compare the best of the proposed policies (HS-Con) with other existing policies: the Equipartition policy (EQUI) [1], the Dynamic policy (DYN) [3], and the Cluster-limited policy (LIM) [4]. These are combination policies, that consist of the existing processor scheduling scheme and a page placement scheme. For EQUI, DYN, and LIM, we employ the distributed scheme as a page placement scheme because it is used as the conventional page placement scheme. In these policies, processor scheduling and page placement schemes are designed separately and do not consider the home-cluster.

Figure 4 indicates that the mean response time with HS-Con is better than that with the other policies. HS-Con can still execute processes more efficiently than the other policies because it can allocate an adequate number of processors and keep high access locality.

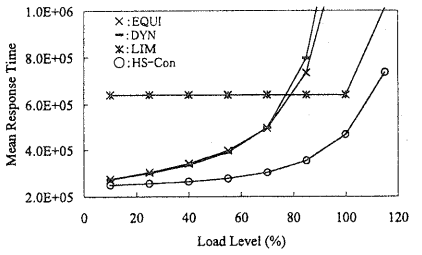

Figure 4: Mean response time for each policy

4 Conclusion

We focused on the interaction between processor scheduling and page placement schemes and proposed several policies that are combinations of the two types of schemes. Furthermore, we introduced the concept of a home-cluster, which is a cluster reserved for a particular process.

These policies were evaluated through simulation experiments. Our results show that the best performance is achieved by the policy in which both the processor scheduling and page placement schemes cooperatively implement the home-cluster concept. The proposed policy is superior to other existing policies in which the processor scheduling scheme and the page placement scheme work independently.

References

[1] A. Gupta, A. Tucker, and S. Urushibara, "The impact of operating system scheduling policies and synchronization methods on the performance of parallel applications," *Proc. of the 1991 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pp.120–132, 1991.

[2] T. Koita, T. Katayama, K. Saisho, and A. Fukuda, "Processor scheduling with page placement for cluster-based NUMA multiprocessors," *Proc. of the Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pp.539–545, 1998.

[3] C. McCann, R. Vaswanim, and J. Zahorjan, "A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors," *ACM Trans. on Computer Systems*, Vol.11, No.2, pp.146–178, 1993.

[4] Y. Ohishi, K. Saisho, and A. Fukuda, "Performance evaluation of two-level scheduling algorithms for NUMA multiprocessors," *IEICE Trans. Information and System*, Vol. J80-D1, No.1, pp.31–41, 1997.