A Graph-Based Subcube Allocation and Task Migration in Hypercube Systems

Ohan Kang*, Byeong Man Kim**, Hyunsoo Yoon, Seung Ryoul Maeng, and Jung Wan Cho

*Applied System Research Center, Qnix Computer Co., Seoul, Korea
**Kumoh National Institute of Technology, Gumi, Kyung-Book, Korea

Center for Artificial Intelligence Research(CAIR) and Department of Computer Science
Korea Advanced Institute of Science and Technology(KAIST), Taejon, Korea

Abstract. In this paper, we propose a task migration scheme based on the HSA (Heuristic Subcube Allocation) strategy to solve the fragmentation problem in a hypercube. Extensive simulation results show that the migration scheme improves the efficiency and significantly reduces the migration cost compared to the previously proposed schemes.

1. Introduction

The performance of the hypercube system is enhanced by an allocation scheme with a good subcube recognition capability, but its merits are degraded by the hypercube fragmentation. Although various researches [1-6] related to the subcube allocation have gone to great lengths to improve the subcube recognition capability, few solutions to the fragmentation problem have been proposed. There have been two schemes for the task migration: one is the full compaction scheme [7] and the other is the partial compaction scheme [8]. But these migration schemes are based on the bottom-up approach instead of the top-down approach. Although the top-down approach can reduce fragmentation compared to the bottom-up approach, it should adopt a migration scheme in order to enhance the system performance.

The main objective of this paper is to propose an efficient task migration scheme which is based on the HSA strategy [9] to solve the fragmentation problem with reduced migration cost compared to the previously proposed schemes, and to evaluate the performance improvement.

From the simulation results, it is observed that the efficiency of the HSA strategy with a new task migration scheme is much higher than that of the Buddy and GC strategies [1]. Furthermore, the HSA strategy with a new migration scheme noticeably reduced migration cost compared to Buddy and GC strategies.

2. Task migration in the HSA strategy

In the hypercube system, a subcube allocation scheme must solve the fragmentation problem by task migration to improve processor utilization. In the HSA strategy, when an incoming task can not be satisfied due to the fragmentation, a task migration scheme, called CSC (Complement Subcube Coalescence) strategy, generates a free subcube which is large enough for the incoming task, and tasks to be migrated are relocated to the free subcube which has the minimum degree to reduce hypercube fragmentation.

Since a largest idle subcube \( \Gamma(C_{\text{max}}) \) has the greatest possibility to generate the requested subcube with the minimum task migration cost, the CSC strategy selects it to form a higher dimension subcube. Since the free subcubes are already collected as a linked list, the selection of \( \Gamma(C_{\text{max}}) \) is very simple, and the new subcube can be determined in \( O(n) \) time for each task because the HSA allocation algorithm has \( O(n) \) time complexity. If the dimension of the coalesced subcube is less than \( k \), the CSC strategy generates a higher dimension subcube using the same procedure recursively until a \( k \)-subcube is generated, and then allocates the \( k \)-subcube to the incoming task.

To minimize the migration cost, the CSC strategy uses the subcube information of the SC-graph \( G \) [9], the degree of vertex. To be an efficient migration algorithm, the candidate subcubes to be selected are restricted to the complement subcubes of \( \Gamma(C_{\text{max}}) \) to gen-
erate the higher dimension subcube. In the CSC strategy, the tasks which have to be relocated are sorted in a descending order of their sizes to assure that no task is relocated more than once. The extracted tasks in the complement subcube and SC-graph configuration are maintained in the CSC strategy, and the actual task migration and SC-graph modification take place only when all the task assignments are obtained by the algorithm. The task migration algorithm in the HSA strategy is as follows:

The CSC algorithm
Comment: Let a \( k \)-subcube be requested, and an IRQ (Input Request Queue) be used for incoming and extracted tasks.

Step 1: If the number of free nodes is less than \( 2^k \), then report "no free \( k \)-subcube", and stop. Otherwise, the incoming task, which cannot be satisfied due to the fragmentation, is inserted into the IRQ.

Step 2: For all tasks \( T_i \) in the IRQ, execute from step 3 to step 5.

Step 3: Select a largest idle subcube \( \Gamma(C_{a,k}) = C_j \), where \( |C_j| = m \). If there exist more than one \( m \)-subcube, select a subcube which has the minimum degree in \( G_f \). If \( |C_j| \) is larger than or equal to the size of \( T_i \), allocate \( C_j \) to \( T_i \), change the configuration of \( G_f \), and go to step 2.

Step 4: If \( |C_j| \) is smaller than the size of \( T_i \), take the complement subcube \( \bar{C}_j \) of \( C_j \), which has the largest degree connected to \( C_j \) in \( G_f \). If there exists a task which is assigned to the subcube containing the nodes in \( \bar{C}_j \), and the task is larger than \( \bar{C}_j \), then go to step 5. Extract all tasks in \( \bar{C}_j \), and insert the tasks into the IRQ in descending order of task sizes. The two \( m \)-subcubes, \( C_j \) and \( \bar{C}_j \), are coalesced into one \((m+1)\)-subcube \( C_j \). If \( |C_j| \) is equal to the size of \( T_i \), then allocate it to \( T_i \) and change the configuration of \( G_f \). If \( |C_j| \) is less than \( T_i \), go to step 3.

Step 5: Take the complement subcube \( \bar{C}_j \) of \( C_j \), and coalesce them into one subcube. If this step cannot satisfy the incoming request, then suspend the request.

Example. Fig. 1(a) represents an idle set of subcubes \( \Gamma(C_4) = \{0100, 11*0, 00***, 1111\} \) and the corresponding \( G_f \) is shown in Fig. 1(b). Let us assume that there is an incoming task \( T_i \) which requires a 3-subcube and also assume that the subcubes 011* and 0101 are allocated to the tasks \( T_1 \) and \( T_2 \), respectively. The input task \( T_1 \) is inserted to IRQ by step 1. By step 3, the 2-subcube \( C_3 \{00**\} \) which has the highest dimension is selected, but it cannot satisfy the task \( T_1 \). Hence, the complement subcube of \( C_3 \) which is composed by \( T_1, T_2 \), and \( C_1 \) is selected by step 4. The tasks, \( T_1 \) and \( T_2 \), are extracted from \( C_3 \{01**\} \), and inserted into the IRQ in the descending order by step 4. The 3-subcube \( C_4 \{0***\} \), which is formed by \( C_3 \) and \( C_5 \), is generated and \( C_j \) is allocated to \( T_1 \). As a result, \( v_1 \) and \( v_3 \) corresponding to \( C_5 \{0100\} \) and \( C_5 \), respectively, are deleted. Since the IRQ contains \( T_1 \) and \( T_2 \), step 3 and step 4 are processed by step 2. By step 3, \( C_2 \{11*0\} \) and \( C_4 \{1111\} \) are allocated to \( T_1 \) and \( T_2 \) respectively, and \( v_2 \) and \( v_4 \) are deleted from \( G_f \), and the algorithm terminates.

![Fig. 1 An idle set of subcubes and SC-graph before task migration; (a) An idle set of subcubes and the assigned tasks; (b) The SC-graph.](a) (b)

3. Simulation

In the simulation, an incoming request is assumed to arrive in every time unit, and the dimensions of the requested subcubes are assumed to follow the uniform distributions. The generation of the requests are continued until the predetermined time.
limit $T$ is reached. The request is queued until a suitable subcube is obtained when the number of free nodes in the hypercube is less than the incoming request, while the task migration is performed when an input request is not satisfied due to the fragmentations. The residence times of the allocated subcubes are assumed to have a uniform distribution. Under these simulation conditions, the performance of the subcube allocation and task migration strategies is measured in terms of $E$, $N_e$, and $N_p$, which are averaged over 100 independent runs and defined as follows:

$U$: Total utilization of processors by the requests in time $T$

$$U = \sum_{i=1}^{J} 2^{I_i} I_i,$$

where $I_i$ is the size of the requested subcube, $i$ is the residence time until $T$ of the request $I_i$, and $J$ is the number of requests that can be satisfied in time interval $T$.

$E$: Efficiency of the strategy

$$E = \frac{U}{2^T}$$

$N_e$: Moving frequencies in time interval $T$

$N_p$: Moving processes in time interval $T$

The number of migrations in the given time interval is called moving frequencies. When a task is moved from one subcube to another, the migration cost is dependent on the size of the task. We assume that a $k$-cubed task contains $2^k$ processes, one process per processor of the $k$-cubed. The relocation of a process from one processor to another is referred to a process move, and the number of process moves in a given time interval is called moving processes. We will compare the migration cost of the CSC scheme with other schemes in terms of moving frequencies and moving processes. If a subcube allocation scheme allows a task migration it is called adaptive mode, otherwise fixed mode.

The HSA strategy is compared with the Buddy and GC strategies in the adaptive mode with respect to the efficiency, and migration cost versus hypercube dimension. The results are shown in Figs. 2-4. In Fig. 2, the mean system efficiency versus hypercube dimension with different residence time for each dimension is shown. Since each dimension has the different residence time, the results only have the meaning for each dimension. As shown in Fig. 2, the HSA strategy performs better than both the Buddy and GC strategies in terms of the efficiency for each dimension. This is caused by the fact that the processor utilization of the Buddy and GC strategies is poor than that of the HSA strategy. The task migration scheme is closely related to the subcube allocation scheme, and task migration is performed within the recognized subcubes, i.e., the Buddy and the GC strategy can only recognize $2^{n-k}$ and $2^{n-2 \cdot k}$ $k$-subcubes in an $n$-cube. On the other hand, the HSA strategy has a complete subcube recognition capability, and the fragmentation can be minimized using the subcube information in the SC-graph.

In Figs. 3-4, the average migration cost as a function of hypercube dimension with the fixed residence time is sketched. For a given residence time, moving frequencies decrease as the hypercube dimension increases as shown in Fig. 3. This is due to the fact that more tasks can be satisfied simultaneously on the hypercube as the dimension increases while the residence time is fixed. The average moving processes versus hypercube dimension for Buddy, GC, and HSA strategies are shown in Fig. 4. It is interesting that the ratio of the moving processes is increased for a larger dimension. This is due to the fact that the size of requested subcube is increased as the hypercube dimension increases. From the results in Figs. 3-4, we know that the HSA strategy achieves a considerable reduction in the migration cost compared to the Buddy and GC strategies.

Fig. 2 Efficiency comparison between different strategies for large cube size; Cube size and resident time: uniform distribution ($P_0 = P_1 = \ldots = P_n$).

In Figs. 3-4, the average migration cost as a function of hypercube dimension with the fixed residence time is sketched. For a given residence time, moving frequencies decrease as the hypercube dimension increases as shown in Fig. 3. This is due to the fact that more tasks can be satisfied simultaneously on the hypercube as the dimension increases while the residence time is fixed. The average moving processes versus hypercube dimension for Buddy, GC, and HSA strategies are shown in Fig. 4. It is interesting that the ratio of the moving processes is increased for a larger dimension. This is due to the fact that the size of requested subcube is increased as the hypercube dimension increases. From the results in Figs. 3-4, we know that the HSA strategy achieves a considerable reduction in the migration cost compared to the Buddy and GC strategies.

Fig. 2 Efficiency comparison between different strategies for large cube size; Cube size and resident time: uniform distribution ($P_0 = P_1 = \ldots = P_n$).
The simulation results show that the HSA strategy gives the better efficiency compared to the Buddy and GC strategies in the adaptive mode. Moreover, the HSA strategy has a significantly reduced migration cost than that of the the Buddy and GC strategies.

References


4. Conclusion

We have proposed a new task migration scheme, called CSC (Complementary Subcube Coalescence), which uses a heuristic and an undirected graph, called SC (SubCube)-graph. If an incoming request is not satisfied due to the system fragmentation, the task migration scheme is performed to generate higher dimension subcubes. The simulation results show that the HSA strategy gives the better efficiency compared to the Buddy and GC strategies in the adaptive mode.