The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 6, pp.99-105, Dec. 31, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.6.99
JIIBC 2022-6-15

주기억장치 할당 문제의 블록 채우기 알고리즘

Algorithm for Block Packing of Main Memory Allocation Problem

이상운*

Sang-Un Lee*

요 약 본 논문은 주기억장치의 사용자 공간이 컴파일 시간에 가변적 크기의 블록들로 분할된 상태에서, 준비상태 큐에 도착한 다중 프로세서들을 적절히 블록에 할당하는 문제를 다루었다. 기존의 할당법인 최초적합, 최적합, 최악적합과 다음 적합 방법들은 준비상태 큐에 도착한 모든 프로세서들을 할당하지 못해 특정 프로세서는 대기상태가 되는 단점을 갖고 있었다. 본 논문에서 제안된 알고리즘은 분할된 블록(홀)의 크기와 준비상태 큐에 있는 프로세서 크기를 내림차순으로 정렬하여 가장 큰 크기의 블록에 가능한 많은 프로세서들을 할당하는 단순한 블록 채우기 알고리즘이다. 제안된 알고리즘을 9개의 벤치마킹 실험 데이터에 적용한 결과 분할 오류로 인해 대기상태 프로세서가 발생하는 1개 데이터를 제외한 8개 데이터 모두에 대해 최소의 내부 단편(IF)을 가지면서도 모든 프로세서들을 할당하는 성능을 보였다.

Abstract This paper deals with the problem of appropriately allocating multiple processors arriving at the ready queue to the block in the user space of the main memory is divided into blocks of variable size at compilation time. The existing allocation methods, first fit(FF), best fit(BF), worst fit(WF), and next fit(NF) methods, had the disadvantage of waiting for a specific processor because they failed to allocate all processors arriving at the ready queue. The proposed algorithm in this paper is a simple block packing algorithm that allocates as many processors as possible to the largest block by sorting the size of the partitioned blocks(holes) and the size of the processor in the ready queue in descending order. The application of the proposed algorithm to nine benchmarking experimental data showed the performance of allocating all processors while having minimal internal fragment(IF) for all eight data except one data in which the weiting processor occurs due to partition errors.

Key Words : Memory partitioning, Memory allocation, Internal fragmentation, Wastage, Block packing

Ⅰ. 서 론

주기억장치(main memory, MM)의 운영체제(operating system, OS)는 준비상태 큐(ready queue, RQ)에 도착한 다중 프로세서(multiple processors)들을 동시에 실행시키기 위해 MM의 사용자 공간(user space)을 분할(partition)하고 할당(allocation)하여 실행시킨다.[1] 본 논문에서는 사용자 공간이 프로세서 실행시간 이전에 가

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 2022년 3월 9일, 수정완료 2022년 11월 10일
게재확정일자 2022년 12월 9일

Received: 9 October, 2022 / Revised: 10 November, 2022 / Accepted: 9 December, 2022
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

변적 크기의 고정된 분할이 된 상태에서 효율적으로 프로세서들을 할당하는 문제(memory allocation problem, MAP)[2-5]에 한정한다.

메모리 할당 방법에는 최초적합(first-fit, FF), 최적합(best-fit, BF), 최악적합(worst-fit, WF)과 다음적합(next-fit, NF)법이 있다.[2-18]

FF는 준비상태 큐의 프로세서 크기를 저장할 수 있는 블록인 홀(hole)들 중에서 첫 번째로 탐색된 홀에 해당 프로세서를 할당하는 방법이며, BF는 준비상태 큐의 프로세서를 저장 가능한 홀들 중에서 낭비(wastage, W)가 최소인 홀에 프로세서를 할당하는 방법이다. WF는 준비상태 큐의 프로세서 크기를 저장 가능한 홀들 중 W가 최대인 홀에 프로세서를 할당하는 방법이며, NF는 현재 할당된 블록(홀)의 포인터부터 시작하여 FF를 수행하는 방법이다. 결국 FF, BF와 WF는 사용자 공간의 처음부터 탐색을 수행한다. 이들 방법은 모두 n개의 메모리 블록에 대해 m개의 준비상태 큐에 도착한 프로세서들을 할당해야 하므로 수행 복잡도는 $O(mn)$이 소요되며, 특정 할당 법은 준비상태 큐에 도착한 m개 프로세서 모두를 할당하여 실행시키지 못하고 특정 프로세서는 대기상태(waiting)로 남겨놓아야 하는 비효율적인 할당 성능을 보인다.

본 논문에서는 준비상태 큐에 도착한 m개의 프로세서 모두를 n개 블록(홀)에 할당하여 동시에 실행시키면서도 비효율적인 분할로 인해 발생하는 메모리 낭비(W)를 최소화시키는 알고리즘을 제안한다. 제안된 알고리즘은 $O(n \log n)$의 수행 복잡도를 갖는다. 2장에서는 주기억장치 분할과 할당의 개념과 할당 법의 문제점을 예제를 통해 고찰한다. 3장에서는 메모리 낭비를 최소화 시킬 수 있는 블록 채우기 알고리즘을 제안한다. 4장에서는 다양한 벤치마킹 실험 데이터에 대해 제안된 알고리즘을 적용하여 알고리즘의 단순성과 더불어 메모리 낭비 최소화의 효율성을 검증한다.

II. 주기억장치 분할과 할당

주기억장치는 고속, 고가이면서 저용량의 휘발성인 RAM으로, 그림 1과 같이 운영체제(OS)가 일부분을 차지하고, OS는 나머지 부분인 사용자 공간을 m개 다중프로세서가 동시에 실행될 수 있도록 n개 블록(block)으로 분할하고 준비상태 큐(RQ)에 도달한 프로세서들에게 블록을 할당하여 실행시킨다.[7]

사용자 공간 메모리를 효율적으로 활용하기 위한 분할 방법에는 단일 분할(single partition)과 다중 분할(multiple partition)법이 있으면 단일분할은 단 하나의 프로세서만을 할당하는 방법으로 MS-DOS에서 사용되었다. 다중 분할방법에는 사전에 동일한 크기로 분할하는 고정된 동일크기 분할(fixed equal-size partition, FEP), 고정된 가변 크기 분할(fixed variable-size partition, FVP)과 동적 분할(dynamic partition, DP)법이 있으며, FVP와 DP가 일반적으로 적용되고 있다.[13]

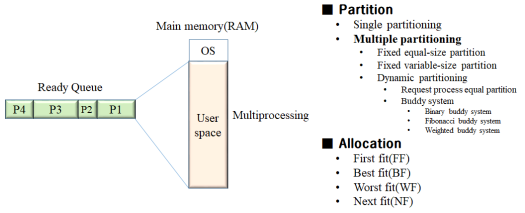

그림 1. 주기억장치 분할과 할당
Fig. 1. Main memory partition & allocation

DP는 프로세서 실행시간에 프로세서들을 사용자 공간에 할당하는 방법으로 W가 전혀 발생하지 않는 장점이 있는 반면에, 추후 메모리를 재배치하는 폐 영역 회수(garbage collection)의 압축(compression)을 해야 하는 OS의 과부하에 시간이 많이 소요되는 단점을 갖고 있다. 반면에 FVP는 컴파일 단계에서 사용자 공간을 분할함으로서 W가 발생하는 단점이 있는 반면에 인접 블록을 하나의 블록으로 병합(merge)하는 합침(coalescing)으로 OS의 부하를 경감시킬 수 있는 장점을 갖고 있다.

따라서 다중 프로세서에 대한 주기억장치를 효율적으로 활용하기 위해서는 메모리 분할 문제(memory partitioning problem, MPP)와 메모리 할당 문제(MAP)를 다루어야만 한다. 본 논문은 사전에 FVP로 사용자 공간이 분할되었다고 가정한 상태에서 준비상태 큐의 프로세서들을 할당하는 문제(MAP)에 한정하여 할당 방법을 다룬다.

FVP 또는 DP로 분할된 사용자 공간에 준비상태 큐의 다중 프로세서들을 효율적으로 할당하는 방법에는 최초적합(FF), 최적합(BF), 최악적합(WF)과 다음적합(NF)법이 알려져 있다.[13]

이러한 할당을 하는 과정에서 하나의 블록(홀)에 특정 프로세서가 할당되고 남은 단편(fragmentation)의 크기로는 준비상태 큐의 어떠한 프로세서들도 할당할 수 없어 메모리 낭비(W)가 발생할 경우 $\sum_{i=1}^{n} W_i$를 내부 단편

(internal fragmentation, IF)이라 한다. 반면에 비록 사용자 공간의 남아있는 총 사용 가능 공간이 준비상태 큐에 존재하는 프로세서 크기를 초과하더라도 단편들이 분산되어 있어 프로세서를 할당하지 못하는 경우 이를 외부 단편(external fragmentation, EF)이라 한다.[8] 따라서 EF는 분할방법의 부 적절성에 기인하며, IF는 할당방법의 부 적절성으로 발생하는 경향이 있다.

지금까지 알려진 4가지 할당 방법인 FF, BF, WF, NF는 준비상태 큐의 m개 프로세서들에 대해 분할된 n개 블록(홀)을 탐색해야 하므로 $O(mn)$ 수행 복잡도가 요구된다. 또한, 특정 방법은 EF로 인해 준비상태 큐에 있는 m개 프로세서 모두를 할당하지 못하고 특정 프로세서는 대기상태로 해당 프로세서를 할당할 수 있는 충분한 크기를 가진 홀의 프로세서가 종료(release)되면 해당 홀에 할당된다. 즉, $W_i = B_i - P_j \geq 0$인 P_j 프로세서가 B_i 블록(홀)에 할당된다. 또한 B_i- W_i로 갱신된다.

표 1의 MAP-1 예제[10]를 대상으로 FF, BF, WF, NF로 사용자 공간에 대기 큐의 프로세서들이 할당되는 방법을 고찰해 본다. 표 1의 MAP-1 데이터는 사용자 공간을 5개의 가변적 크기로 분할한 상태에서 준비상태 큐에 도착한 5개 프로세서를 할당한 결과 FF와 WF는 프로세서 P5를 할당하지 못한 결과를 나타내었다. 반면에 BF와 NF는 5개 프로세서 모두를 할당하였으나 IF를 살펴보면 BF=15+177+55+15+7=267KB, NF=27+100+60+80+0=267KB의 단편들이 발생하였다. BF의 경우 만약 177KB 보다 큰 프로세서가 준비상태 큐에 존재한다면 이를 할당할 수 없고 대기상태에 머물러야만 한다. 마찬가지로 NF의 경우는 100KB 보다 큰 프로세서가 준비상태 큐에 도달하면 할당을 할 수 없다.

이와 같이 주어진 데이터에 적용되는 할당규칙으로 인해 발생되는 EF로 인해 특정 방법이 프로세서 할당에 실패하여 대기상태로 되는 결과를 유발하기 때문에 FF, BF, WF, NF 중에서 어느 방법이 최적인지에 대해서는 명백히 밝혀진 사실이 없다. 본 논문에서는 최적의 할당 알고리즘을 선정함에 있어 EF가 발생하지 않으면서 IF 최소화를 달성하는 방법으로 판단한다. 만약 모든 할당 알고리즘이 EF를 발생시키면 그들 중 대기(waiting) 프로세서의 크기가 최소인 방법을 최적의 알고리즘으로 판단한다.

3장에서는 준비상태 큐의 모든 프로서서들을 할당하면서도 IF 최소화를 달성할 수 있는 $O(m \log m)$ 수행 복잡도의 블록 채우기 알고리즘(block packing algorithm, BPA)을 제안한다.

표 1. MAP-1 예제 데이터의 메모리 할당
Table 1. Memory allocation for MAP-1 example data

MM : B1(50KB), B2(200KB), B3(70KB), B4(115KB), B5(15KB)
RQ : P1(100KB), P2(10KB), P3(35KB), P4(15KB), P5(23KB)

User space	FF	BF	WF	NF
B1(50KB)	P2(10KB) W=40KB	P3(35KB) W=15KB	P4(15KB) W=35KB	P5(23KB) W=27KB
B2(200KB)	P1(100KB) W=100KB	P5(23KB) W=177KB	P1(100KB) W=100KB	P1(100KB) W=100KB
B3(70KB)	P3(35KB) W=35KB	P4(15KB) W=55KB	P3(35KB) W=35KB	P2(10KB) W=60KB
B4(115KB)	P4(15KB) W=100KB	P1(100KB) W=15KB	P2(10KB) W=105KB	P3(35KB) W=80KB
B5(15KB)	Free	P2(10KB) W=5KB	Free	P4(15KB) W=0KB
Waiting	P5(23KB)	-	P5(23KB)	-
IF	290KB	267KB	290KB	267KB

Ⅲ. 블록 채우기 알고리즘

본 장에서 제안하는 방법은 분할된 블록과 준비상태 큐의 프로세서 크기에 대해 내림차순으로 정렬하여 할당 가능한 블록(홀)에 가능한 많은 프로세서를 할당하는 단순한 방법을 제안한다. 만약 낭비(W)를 보다 줄일 수 있는 홀로 프로세서를 이동시키는 할당 최적화(allocation optimization)을 수행한다. 이 방법을 블록 채우기 알고리즘(BPA)이라 하며, 그림 2와 같이 수행된다.

Step 1. 분할된 블록(B_i)과 준비상태 큐 프로세서(P_j) 크기 내림차순 정렬.
Step 2. $P_j \leq B_i$ or $P_j \leq W_i$인 B_i에 P_j 할당(블록 채우기), W 계산.
Step 3. W_i를 최소화할 수 있는 블록으로 프로세서 이동(할당 최적화) 수행.

그림 2. 블록 채우기 알고리즘
Fig. 2. Block packing algorithm(BPA)

표 1의 MAP-1 예제 데이터에 대해 BPA를 적용하여 얻은 결과는 그림 3과 같다. BF와 NF는 Free 블록 없이 IF=267KB를 보인 반면에, 제안된 BPA는 B1, B3, B4와 B5를 Free로 유지하면서도 단지 B2(200KB)에 P1, P3, P5, P4, P2 순서로 모두 할당하여 IF=17KB의 메모리 할당 능력을 보였다.

MM	RQ	Block packing		
		MM	RQ	W
B1(50KB)	P1(100KB)	B2(200KB)	P1(100KB) P3(35KB) P5(23KB) P4(15KB) P2(10KB)	100KB 65KB 42KB 27KB 17KB
B2(200KB)	P2(10KB)	B4(115KB)	Free	
B3(70KB)	P3(35KB)	B3(70KB)	Free	
B4(115KB)	P4(15KB)	B1(50KB)	Free	
B5(15KB)	P5(23KB)	B5(15KB)	Free	

(a) BPA

MM : B1(50KB), B2(200KB), B3(70KB), B4(115KB), B5(15KB)
RQ : P1(100KB), P2(10KB), P3(35KB), P4(15KB), P5(23KB)

User space	FF	BF	WF	NF	BPA
B1 (50KB)	P2(10KB) W=40KB	P3(35KB) W=15KB	P4(15KB) W=35KB	P5(23KB) W=27KB	Free
B2 (200KB)	P1(100KB) W=100KB	P5(23KB) W=177KB	P1(100KB) W=100KB	P1(100KB) W=100KB	P1(100KB) P3(35KB) P5(23KB) P4(15KB) P2(10KB) W=17KB
B3 (70KB)	P3(35KB) W=35KB	P4(15KB) W=55KB	P3(35KB) W=35KB	P2(10KB) W=60KB	Free
B4 (115KB)	P4(15KB) W=100KB	P1(100KB) W=15KB	P2(10KB) W=105KB	P3(35KB) W=80KB	Free
B5 (15KB)	Free	P2(10KB) W=5KB	Free	P4(15KB) W=0KB	Free
Waiting	P5(23KB)	-	P5(23KB)	-	-
IF	290KB	267KB	290KB	267KB	17KB

(b) Allocation & IF

그림 3. MAP-1 예제 데이터의 BPA
Fig. 3. BPA for MAP-1

Ⅳ. 적용 및 결과 분석

본 장에서 실험에 사용된 데이터는 그림 4에 제시하였으며, 기존 연구 결과와 더불어 BPA는 그림 5에 제시하였다.

MM : B1(300KB), B2(250KB), B3(150KB), B4(100KB)
RQ : P1(150KB), P2(200KB), P3(100KB)
(a) MAP-2[16]

MM : B1(100KB), B2(500KB), B3(200KB), B4(300KB), B5(600KB)
RQ : P1(212KB), P2(417KB), P3(112KB), P4(426KB)
(b) MAP-3[18]

MM : B1(200KB), B2(100KB), B3(400KB), B4(300KB), B5(500KB)
RQ : P1(212KB), P2(313KB), P3(100KB), P4(200KB), P5(412KB)
(c) MAP-4[11]

MM : B1(200KB), B2(100KB), B3(400KB), B4(300KB), B5(500KB)
RQ : P1(311KB), P2(101KB), P3(221KB), P4(234KB), P5(401KB)
(d) MAP-5[11]

MM : B1(200KB), B2(100KB), B3(400KB), B4(300KB), B5(500KB)
RQ : P1(111KB), P2(213KB), P3(314KB), P4(100KB), P5(222KB)
(e) MAP-6[11]

MM : B1(500KB), B2(200KB), B3(800KB), B4(400KB), B5(100KB), B6(700KB), B7(300KB), B8(600KB), B9(1000KB), B10(900KB)
RQ : P1(212KB), P2(150KB), P3(375KB), P4(950KB), P5(350KB), P6(632KB), P7(400KB), P8(717KB), P9(811KB)
(f) MAP-7[12]

MM : B1(200KB), B2(400KB), B3(600KB), B4(500KB), B5(300KB), B6(250KB)
RQ : P1(357KB), P2(210KB), P3(468KB), P4(491KB)
(g) MAP-8[13]

MM : B1(50KB): 사용 중, B2(150KB), B3(300KB) : 사용 중, B4(350KB), B5(600KB) : 사용 중
RQ : P1(300KB), P2(25KB), P3(125KB), P4(50KB)
(h) MAP-9[13,15]

그림 4. 실험 데이터
Fig. 4. Experimental Data

BPA
MM : B1(300KB), B2(250KB), B3(150KB), B4(100KB)
RQ : P2(200KB), P1(150KB), P3(100KB)

MM	RQ	W	AO	W
B1(300KB)	P2(200KB) P3(100KB)	100 0		0
B2(250KB)	P1(150KB)	100	Free	
B3(150KB)	Free		P1(150KB)	0
B4(100KB)	Free		Free	

User space	FF	BF	WF	NF	BPA
B1(300KB)	P1(150KB) W=150KB	Free	P1(150KB) W=150KB	P1(150KB) W=150KB	P2(200KB) P3(100KB) W=0KB
B2(250KB)	P2(200KB) W=50KB	P2(200KB) W=50KB	P2(200KB) W=50KB	P2(200KB) W=50KB	Free
B3(150KB)	P3(100KB) W=50KB	P1(150KB) W=0KB	P3(100KB) W=50KB	P3(100KB) W=50KB	P1(150KB) W=0KB
B4(100KB)	Free	P3(100KB) W=0KB	Free	Free	Free
Waiting	-	-	-	-	-
IF	250KB	50KB	250KB	250KB	0KB

(a) MAP-2

BPA
MM : B5(600KB), B2(500KB), B4(300KB), B3(200KB), B1(100KB)
RQ : P4(426KB), P2(417KB), P1(212KB), P3(112KB)

MM	RQ	W	AO	W
B5(600KB)	P4(426KB) P3(112KB)	174 62	-	-
B2(500KB)	P2(417KB)	83	-	-
B4(300KB)	P1(212KB)	88	-	-
B3(200KB)	Free			
B1(100KB)	Free			

User space	FF	BF	WF	NF	BPA
B1(100KB)	Free	Free	Free	Free	Free
B2(500KB)	P1(212KB) W=288KB	P2(417KB) W=83KB	P2(417KB) W=83KB	P1(212KB) W=288KB	P2(417KB) W=83KB
B3(200KB)	P3(112KB) W=88KB	P3(112KB) W=88KB	Free	P3(112KB) W=88KB,	Free
B4(300KB)	Free	P1(212KB) W=88KB,	P3(112KB) W=188KB	Free	P1(212KB) W=88KB,
B5(600KB)	P2(417KB) W=183KB	P4(426KB) W=174KB	P1(212KB) W=388KB,	P2(417KB) W=183KB	P4(426KB) P3(112KB) W=62KB
Waiting	P4(426KB)	-	P4(426KB)	P4(426KB)	-
IF	559KB	433KB	659KB	559KB	233KB

(b) MAP-3

BPA
MM : B5(500KB), B3(400KB), B4(300KB), B1(200KB), B2(100KB)
RQ : P5(412KB), P2(313KB), P1(212KB), P4(200KB), P3(100KB)

MM	RQ	W	AO	W
B5(500KB)	P5(412KB)	88	-	-
B3(400KB)	P2(313KB)	87	-	-
B4(300KB)	P1(212KB)	88	-	-
B1(200KB)	P4(200KB)	0	-	-
B2(100KB)	P3(100KB)	0	-	-

User space	FF	BF	WF	BPA
B1(200KB)	P3(100KB) W=100KB	P4(200KB) W=0KB	Free	P4(200KB) W=0KB
B2(100KB)	Free	P3(100KB) W=0KB	Free	P3(100KB) W=0KB
B3(400KB)	P1(212KB) W=188KB	P2(313KB) W=87KB	P2(313KB) W=87KB	P2(313KB) W=87KB
B4(300KB)	P4(200KB) W=100KB	P1(212KB), W=88KB	P4(200KB) W=100KB	P1(212KB), W=88KB
B5(500KB)	P2(313KB) W=187KB	P5(412KB) W=88KB	P1(212KB) P3(100KB) W=188KB	P5(412KB) W=88KB
Waiting	P5(412KB)	-	P5(412KB)	-
IF	575KB	263KB	375KB	263KB

(c) MAP-4

BPA

MM : B5(500KB), B3(400KB), B4(300KB), B1(200KB), B2(100KB)
RQ : P5(401KB), P1(311KB), P4(234KB), P3(221KB), P2(101KB)

MM	RQ	W	AO	W
B5(500KB)	P5(401KB)	99	-	-
B3(400KB)	P1(311KB)	89	-	-
B4(300KB)	P4(234KB)	66	-	-
B1(200KB)	Free		-	-
B2(100KB)	P2(101KB)	99	-	-
Waiting	P3(221KB)			

User space	FF	BF	WF	BPA
B1(200KB)	P2(101KB) W=99KB	P2(101KB) W=99KB	Free	P2(101KB) W=99KB
B2(100KB)	Free	Free	Free	Free
B3(400KB)	P1(311KB) W=89KB	P1(311KB) W=89KB	P2(101KB) P4(234KB) W=65KB	P1(311KB) W=89KB
B4(300KB)	P3(221KB) W=79KB	P3(221KB) W=79KB	P3(221KB) W=79KB	P4(234KB) W=66KB
B5(500KB)	P4(234KB) W=266KB	P4(234KB) W=266KB	P1(311KB) W=189KB	P5(401KB) W=99KB
Waiting	P5(401KB)	P5(401KB)	P5(401KB)	P3(221KB)
IF	533KB	533KB	333KB	353KB

(d) MAP-5

BPA

MM : B5(500KB), B3(400KB), B4(300KB), B1(200KB), B2(100KB)
RQ : P3(314KB), P5(222KB), P2(213KB), P1(111KB), P4(100KB)

MM	RQ	W	AO	W
B5(500KB)	P3(314KB) P1(111KB)	186 75	-	-
B3(400KB)	P5(222KB) P4(100KB)	178 78	-	-
B4(300KB)	P2(213KB)	87	-	-
B1(200KB)	Free			
B2(100KB)	Free			

User space	FF	BF	WF	BPA
B1(200KB)	P1(111KB) W=89KB	P1(111KB) W=89KB	Free	Free
B2(100KB)	P4(100KB) W=0KB	P4(100KB) W=0KB	Free	Free
B3(400KB)	P2(213KB) W=187KB	P3(314KB) W=86KB	P2(213KB) W=187KB	P5(222KB) P4(100KB) W=78KB
B4(300KB)	P5(222KB) W=78KB	P2(213KB) W=87KB	P4(100KB) W=200KB	P2(213KB) W=87KB
B5(500KB)	P3(314KB) W=186KB	P5(222KB) W=278KB	P1(111KB) P3(314KB) W=75KB	P3(314KB) P1(111KB) W=75KB
Waiting	-	-	P5(222KB)	-
IF	540KB	540KB	462KB	240KB

(e) MAP-6

BPA

MM : B9(1000KB), B10(900KB), B3(800KB), B6(700KB), B8(600KB),
B1(500KB), B4(400KB), B7(300KB), B2(200KB), B5(100KB),
RQ : P4(950KB), P9(811KB), P8(717KB), P6(632KB), P7(400KB), P3(375KB), P5(350KB), P1(212KB), P2(150KB)

MM	RQ	W	AO	W
B9(1000KB)	P4(950KB)	50	-	-
B10(900KB)	P9(811KB)	89	-	-
B3(800KB)	P8(717KB)	83	-	-
B6(700KB)	P6(632KB)	68	-	-
B8(600KB)	P7(400KB) P2(150KB)	200 50	P3(375KB) P1(212KB)	225 13
B1(500KB)	P3(375KB)	125	P5(350KB) P2(150KB)	150 0
B4(400KB)	P5(350KB)	50	P7(400KB)	0
B7(300KB)	P1(212KB)	88	Free	
B2(200KB)	Free		Free	
B5(100KB)	Free		Free	

User space	FF	BF	WF	BPA
B1(500KB)	P1(212KB) P2(150KB) W=138KB	P5(350KB) W=150KB	Free	P5(350KB) P2(150KB) W=0KB
B2(200KB)	Free	P2(150KB) W=50KB	Free	Free
B3(800KB)	P3(375KB) P5(350KB) W=75KB	P8(717KB) W=83KB	P3(375KB) W=425KB	P8(717KB) W=83KB
B4(400KB)	P7(400KB) W=0KB	P3(375KB) W=25KB	Free	P7(400KB) W=0KB
B5(100KB),	Free	Free	Free	Free
B6(700KB)	P6(632KB) W=68KB	P6(632KB) W=68KB	P7(400KB) W=300KB	P6(632KB) W=68KB
B7(300KB)	Free	P1(212KB) W=88KB	Free	Free
B8(600KB)	Free	P7(400KB) W=200KB	Free	P3(375KB) P1(212KB) W=13KB
B9(1000KB)	P4(950KB) W=50KB	P4(950KB) W=50KB	P1(212KB) P5(350KB) W=438KB	P4(950KB) W=50KB
B10(900KB)	P8(717KB) W=183KB	P9(811KB) W=89KB	P2(150KB) P6(632KB) W=118KB	P9(811KB) W=89KB
Waiting	P9(811KB)	-	P4(950KB) P8(717KB) P9(811KB)	-
IF	514KB	803KB	1,281KB	303KB

(f) MAP-7

BPA
MM : B3(600KB), B4(500KB), B2(400KB), B5(300KB), B6(250KB), B1(200KB)
RQ : P4(491KB). P3(468KB), P1(357KB), P2(210KB)

MM	RQ	W	AO	W
B3(600KB)	P4(491KB)	109	-	-
B4(500KB)	P3(468KB)	32	-	-
B2(400KB)	P1(357KB)	43	-	-
B5(300KB)	Free			
B6(250KB)	P2(210KB)	40	-	-
B1(200KB)	Free			

User space	FF	BF	WF	BPA
B1(200KB)	Free	Free	Free	Free
B2(400KB)	P1(357KB) W=43KB	P1(357KB) W=43KB	Free	P1(357KB) W=43KB
B3(600KB)	P2(210KB) W=390KB	P4(491KB) W=109KB	P1(357KB) W=243KB	P4(491KB) W=109KB
B4(500KB)	P3(468KB) W=32KB	P3(468KB) W=32KB	P2(210KB), W=290KB	P3(468KB) W=32KB
B5(300KB)	Free	Free	Free	Free
B6(250KB)	Free	P2(210KB) W=40KB	Free	P2(210KB) W=40KB
Waiting	P4(491KB)	-	P3(468KB) P4(491KB)	-
IF	465KB	224KB	533KB	224KB

(g) MAP-8

BPA
MM : B4(350KB), B2(150KB)
RQ : P1(300KB), P3(125KB), P4(50KB), P2(25KB)

MM	RQ	W	AO	W
B4(350KB)	P1(300KB) P4(50KB)	50 0	-	-
B2(150KB)	P3(125KB) P2(25KB)	25 0	-	-

User space	FF	BF	WF	BPA
B1(50KB)	X	X	X	X
B2(150KB)	P2(25KB) P3(125KB) W=0KB	P3(125KB) W=25KB	P2(25KB) P3(125KB) W=0KB	P3(125KB) P2(25KB) W=0KB
B3(300KB)	X	X	X	X
B4(350KB)	P1(300KB) P4(50KB) W=0KB	P1(300KB) P2(25KB) W=25KB	P1(300KB) P4(50KB) W=0KB	P1(300KB) P4(50KB) W=0KB
B5(600KB)	X	X	X	X
Waiting	-	P4(50KB)	-	-
IF	0KB	50KB	0KB	0KB

(h) MAP-9

그림 5. 실험 데이터에 대한 할당 결과
Fig. 5. Result of allocation for experimental Data

본 논문에서 제시된 벤치마킹 실험 데이터에 대해 제안된 BPA를 수행한 결과 비교는 표 2에 제시하였다. 기존에 알려진 방법은 9개 중 8개 데이터에서 최적의 결과를 나타낸 BF가 가장 우수한 것으로 판단할 수 있다. 그러나 본 논문에서 제안된 BPA는 9개 데이터 모두에서 최적의 결과를 보여 가장 성능이 좋은 할당 알고리즘이라 할 수 있다.

표 2. 알고리즘 성능 비교
Table 2. Comparison of algorithm performance

문제	최적 해(IF)				
	FF	BF	WF	NF	BPA
MAP-1		267		267	17
MAP-2		50			0
MAP-3		433			233
MAP-4		263			263
MAP-5	P5(401KB)	P5(401KB)	P5(401KB)	P5(401KB)	P3(221KB)
MAP-6	540	540			240
MAP-7		803			303
MAP-8		224			224
MAP-9	0		0		0

V. 결 론

본 논문은 OS가 준비상태 큐에 도착한 다중 프로세서들을 사용자 공간에 적절히 할당하여 동시에 실행시키는 문제를 다루었다. 기존에 알려진 메모리 할당 방법인 FF, BF, WF와 NF는 준비상태 큐에 도착한 모든 프로세서들을 할당하지 못하는 경우도 발생하는 문제점을 갖고 있었다.

본 논문에서 제안한 알고리즘은 분할된 블록과 준비상태 큐에 도착한 프로세서들의 크기 내림차순으로 정렬하고, 가장 큰 크기의 블록(홀)에 W가 최소가 되도록 가능한 많은 프로세서들을 할당하는 단순한 전략을 적용하였다. 일단 할당이 완료되면 추가적으로 IF를 최소화 시킬 수 있도록 특정 블록에 할당된 프로세서를 다른 블록으로 이동시키는 할당 최적화를 수행하였다.

본 논문에서 제안한 알고리즘을 9개의 벤치마킹 실험 데이터에 적용한 결과, 가변적 크기의 메모리 분할 오류로 인한 MAP-5를 제외한 모든 실험 데이터에 대해 준비상태 큐의 모든 프로세서들을 할당하면서도 IF를 최소화 시키는 결과를 보였다.

References

[1] D. Kamar, M. Singh, and H. Kaur, "Memory Management in Operating System," Journal of Emerging Technologies and Innovative Research, Vol. 6, No. 4, pp. 465-471, Apr. 2019.

[2] D. P. Bovet and G. Estrin, "A Dynamic Memory Allocation Algorithm," IEEE Transactions on Computers, Vol. C-19, No. 5, pp. 403-411, May 1970, https://doi.org/10.1109/T-C.1970.222938

[3] D. S. Hirschberg, "A Class of Dynamic Memory Allocation Algorithms," Communications of the ACM, Vol. 16, No. 10, pp. 615-618, Oct. 1973, https://doi.org/10.1145/362375.362392

[4] F. Karabiber, A. Sertbaş, and A. H. Zaim, "Dynamic Memory Allocator Algorithms Simulation and Performance Analysis," Journal of the Electrical & Electronics Engineering, Vol. 5, No. 2, pp. 1435-1441, Mar 2005.

[5] M. A. Awais, "Memory Management: Challenges and Techniques for Traditional Memory Allocation Algorithms in Relation with Today's Real Time Needs," International Journal of Multidisciplinary Sciences and Engineering, Vol. 7, No. 3, pp. 13-19, Mar. 2016.

[6] Webeduclick, "Partition Allocation Methods in Memory Management," https://webeduclick.com/ partition-allocation-methods-in-memory-management/, Retrieved Mar. 2022.

[7] A. Kumar, "EEL 602 Operating Systems : Memory Management," https://docplayer.net/ 25560803-Memory-management-reading-silberschatz-chapter-9-reading-stallings-chapter-7-eel-602.html, Retrieved Mar. 2022.

[8] Varshachoudhary, "Memory Management in Operating System," https://www.geeksforgeeks. org/memory-management-in-operating-system/, Nov. 2021.

[9] Geeksforgeeks, "Partition Allocation Methods in Memory Management," https://www.geeksforgeeks. org/partition-allocation-methods-in-memory-management/, Nov. 2020.

[10] Prepinsta, "Partition Allocation Method in Operating System," https://prepinsta.com/operating- systems/ partition-allocation-method/, Retrieved Mar. 2022.

[11] L. G. Kabari and T. S. Gogo, "Efficiency of Memory Allocation Algorithms Using Mathematical Model," International Journal of Emerging Engineering Research and Technology, Vol. 3, No. 9, pp. 55-67, Sep. 2015.

[12] L. W. Htun, M. M. M. Kay, and A. A. Cho, "Analysis of Allocation Algorithms in Memory Management," International Journal of Trend in Scientific Research and Development, Vol. 3, No. 5, pp. 1985-1987, Aug. 2019, https://doi. org/10.31142/ijsrd26731

[13] G. Vidyalay, "Contiguous Memory Allocation : Static Partitioning," https://www.gatevidyalay. com/contiguous-memory-allocation-static-partitioning/, Retrieved Mar. 2022.

[14] S. Chhabra, "Program for First Fit Algorithm in Memory Management," https://www.geeksforgeeks. org/program-first-fit-algorithm-memory-management/, Dec. 2021.

[15] S. Pandey, "Variable Sized Partitioning: First Fit, Best Fit and Worst Fit(in Hindi)," https://unacademy.com/course/hindi-introduction-to-operating-system-for-gate-aspirants/L4XNB8M5, Retrieved Mar. 2022.

[16] R. Ranjan, "Memory Management," Patliputra University, http://ppup.ac.in/download/econtent/pdf/BCA_MEMORY++MANAGEMENT, Retrieved Mar. 2022.

[17] Idc, "Memory Partitioning," https://www.idc- online.com/pdfs/information_technology/Memory_Partitioning, Retrieved Mar. 2022.

[18] S. Chhabra , "Program for First Fit Algorithm in Memory Management," https://www.geeksforgeeks.org/program-first-fit-algorithm-memory-management/, Dec. 2021.

저 자 소 개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
- 2007.3 ~ 2015.3 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015.4 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수
- 관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 인공지능과 빅데이터분석, 최적화 알고리즘
- e-mail : sulee@gwnu.ac.kr