디지털산업정보학회 논문지 제13권 제4호-2017년 12월
http://dx.doi.org/10.17662/ksdim.2017.13.4.153

분산 시스템에서 파일 이전과 부하 균등을 위한 수학적 모델

문 원 식*

Mathematical Model for File Migration and Load Balancing in Distributed Systemsc

Moon Wonsik

〈Abstract〉

Advances in communication technologies and the decreasing cost of computers have made distributed computer systems an attractive alternative for satisfying the information needs of large organizations.

This paper presents a distributed algorithm for performance improvement through load balancing and file migration in distributed systems. We employed a sender initiated strategy for task migration and used learning automata with several internal states for file migration. A task can be migrated according to the load information of a computer. A file is migrated to the destination processor when it is in the right boundary state. We also described an analytical model for load balancing with file migration to verify the proposed algorithm. Analytical and simulation results show that our algorithm is very well-suited for distributed system environments.

Key Words : Dynamic File Migration, Learning Automata, Mathematical Model

Ⅰ. 서론

다수의 컴퓨터 시스템이 통신선을 통하여 연결된 분산 시스템의 성능을 향상시키기 위한 많은 연구가 이루어졌다. 그 가운데서 부하 균등과 파일 이전은 분산 시스템의 성능에 중요한 영향을 미친다고 할 수 있다.

부하 균등은 태스크를 높은 부하 상태인 컴퓨터에서 낮은 부하 상태인 컴퓨터로 이전시키고, 컴퓨터들 사이의 부하 균등을 잘 맞추어 줌으로써 시스템 안의 컴퓨터들에게 상대적인 처리 작업량을 균형있게 할당하여 보다 나은 컴퓨터 이용률과 전체 시스템의 성능을 향상시킨다[1-4].

파일 이전은 시간대에 따라 액세스 빈도수가 다를 경우 어떤 시간대에 임의의 파일에 대해 특정한 컴퓨터의 액세스가 빈번히 일어나면 이 컴퓨터에게 파일을 보내 주어 사용하게 하는 방법이다[5].

* 평택대학교 융합소프트웨어학과 교수

분산 처리 시스템에서 자동적인 파일 이전에 관한 연구들 가운데서 특히, Wah[6]는 파일 이전 문제가 갖는 특성에 관심을 두었고, 한 파일의 여러 복사본을 이전할 때 소요되는 비용을 결정하는 문제가 NP-hard임을 증명하였다. Sheng[5]은 그리디(greedy) 원리를 이용하여 파일 이전을 시도하였다. Hac의 방법은 부하 균등이 이루어질 때 관련된 파일을 같이 이동시킨다[4]. 하지만 관련된 파일들을 전부 찾아 이동시키는 것이 실제적으로 쉽지가 않을 뿐만 아니라 다른 태스크가 이전되는 파일을 더 많이 참조한다면 오히려 이전시키는 것이 나쁜 결과를 초래할 수 있다.

지금까지 제안된 파일 이전을 위한 방법들은 파일을 이전하기 위한 결정 함수(decision function)의 계산이 복잡하고 결정 함수의 수행을 위한 제어 메시지(control message)의 전달이 필요하였다. 그러나 이런 결정 함수의 수행은 컴퓨터에게 많은 계산 부담을 준다. 또한 모든 컴퓨터가 한 네트워크를 공유하므로 네트워크에 연결된 컴퓨터의 수가 많아질수록 파일 이전을 위한 제어 메시지의 전달 때문에 네트워크가 분산 처리 시스템의 병목 현상(bottleneck)의 원인이 된다. 따라서 이전되는 제어 메시지의 수를 줄이는 연구가 필요하게 되었다. 그러나 본 논문에서는 파일 이전 시 학습 오토마타를 사용함으로써 파일 참조에 관한 정보만 있으면 파일 상태에 따라 파일 이전 여부, 파일 이전의 시기 및 장소를 결정함으로써 현재 시스템의 상태를 잘 반영하고 파일 이전을 위한 메시지 전송과 추가적인 계산 오버헤드를 제거할 수 있다.

지금까지 분산 시스템에서 부하 균등과 파일 이전에 관한 연구 결과가 많이 발표되었지만, 이러한 연구들은 부하 균등과 파일 이전 중에서 한 가지만 고려했다. 본 논문에서는 부하 균등과 파일 이전을 함께 고려한 모델을 제시하고 분석을 시도했다.

2장에서는 부하 균등과 파일 이전 알고리즘을 기술하고 두 알고리즘의 결합 방법에 대해 기술한다. 3장에서는 제안한 모델을 분석하기 위한 수학적 모델을 기술하고, 4장에서는 컴퓨터 시뮬레이션 결과를 기술하고 분석한다. 5장에서 결론을 맺는다.

2. 파일 이전을 고려한 부하 균등

부하 균등과 파일 이전을 함께 고려하기 위해서 부하 균등은 동적 태스크(task) 이전 방법을 사용하고, 파일 이전은 학습오토마타를 활용한다.

2.1 부하 균등

지금까지 발표된 부하 균등 알고리즘은 지역 부하 균등(local load balancing)과 전역 부하 균등과 중앙 집중식 및 분산형 알고리즘으로 구분된다. 또 다른 분류 방식으로 송신자 위주(sender-initiated) 방식과 수신자 위주(receiver-initiated) 방식 및 혼합 방식(hybrid)으로 구분할 수 있다. 여기서 기술한 알고리즘은 송신자 위주 방식으로서 부하 균등을 위해서 컴퓨터의 부하 상태를 크게 과부하(heavy load), 보통 부하(normal load), 경 부하(light load)로 나눈다. 과부하인 컴퓨터는 갖고 있는 태스크의 수가 많으므로 보유하고 있는 태스크를 다른 컴퓨터에게 이전해 줄 수 있는 상태이고, 저부하인 컴퓨터는 갖고 있는 태스크의 수가 적으므로 다른 컴퓨터가 갖고 있는 태스크를 이전 받을 수 있는 상태이고, 보통 부하인 컴퓨터는 갖고 있는 태스크의 수가 적당하므로 태스크를 이전하거나 이전 받지 않는 상태이다. 본 논문에서는 컴퓨터의 상태를 저 부하, 정상 부하, 그리고 과부하 상태로 분류하며 저 부하 상태는 태스크의 개수가 하한 값 T1보다 작거나 같을 경우이고 정상 부하 상태는 하한 값 T1보다 크고

상한 값 T2보다 작은 경우이고 과부하 상태는 상한 값 T2보다 크거나 같은 경우이다.

도착한 태스크에 대한 전달 정책은 해당 컴퓨터의 부하를 측정하고 현재 부하의 상태가 과부하 상태이면 태스크를 이전하기 위해 목적 컴퓨터를 찾는 송신자 위주 방식을 사용한다. 이전될 태스크를 받을 컴퓨터의 위치 선정 정책은 이전될 태스크가 있을 때 이 컴퓨터에 있는 부하 표를 참조하여 최소의 부하를 갖고 저 부하 상태인 컴퓨터를 찾고 이 컴퓨터에게 이전될 태스크를 이동시킨다. 부하 표는 모든 컴퓨터가 가지고 있으며, 시스템 상의 모든 컴퓨터들의 부하 상태를 나타낸다. 부하 표는 <표 1>과 같다. 여기서 부하 량은 태스크의 수를 의미한다.

위치 선정을 위해서 컴퓨터의 부하에 변동이 있으면 이 정보를 모든 컴퓨터에게 방송한다. 예를 들면 컴퓨터에 새로운 태스크가 생성되어 부하의 크기

<표 1> 부하 표

컴퓨터 번호	부하량
C_1	L_1
C_2	L_2
C_3	L_3
.	.
.	.
.	.
C_M	L_M

가 증가한 경우와, 태스크의 수행이 완료되어 부하의 크기가 감소한 경우 및 부하 균등 작업의 수행에 따라 태스크를 받거나 보내준 경우의 부하 변동 시에 다른 모든 컴퓨터에게 자신의 변화된 부하를 알려줌으로써 다른 컴퓨터에 있는 부하 표를 갱신할 수 있게 한다. 또한, 임의의 컴퓨터로 이전된 태스크는 서비스 큐로 들어가고 서비스 큐 안의 태스크는 이전되지 않게 함으로써 끊임없이 태스크가 돌아다니는 것을 방지했고, 각 컴퓨터에서는 컴퓨터에 먼저 들어온 태스크가 먼저 처리되는 선입선출 구조로 하였다.

다음에 기술된 알고리즘 Job_Sender는 태스크 이전과 태스크 수행 시 파일 이전 알고리즘을 호출하는 과정을 기술한 것이다.

Algorithm : Job_Sender
/* Ci : i번 컴퓨터, Si : Ci의 상태, Cd : 목적지 컴퓨터, S : 이전될 태스크들의 집합 */
· 컴퓨터 Ci에 태스크 Jk가 도착.
· Si := transfer_routine()
If Si = sender Then
· 태스크 Jk를 전달 큐에 삽입한다.
· Cd := location_routine()
· 집합 S := selection_routine(Cd)
· 집합 S에 속하는 태스크들은 목적지 컴퓨터 Cd로 이전한다.
· 부하값 갱신.
· information_routine()에 의해 Ci의 부하는 다른 모든 컴퓨터에 전달된다.
Else
· 태스크 Jk를 서비스 큐에 삽입한다.
EndIf
· Ci는 태스크 수행을 계속한다.
· 태스크 수행 중 파일 참조가 발생하면 참조된 파일이 저장된 컴퓨터에서 파일 이전 알고리즘(LearningMigration)이 수행된다.
End{Job_Sender}

태스크들은 각 컴퓨터에 지수 분포로 독립적으로 도착한다. 태스크가 컴퓨터 Ci에 도착하면 컴퓨터

Ci는 transfer_routine()을 수행하여 Ci의 상태를 결정한다. 태스크 수가 T2보다 크거나 같아서 Ci의 상태가 송신자 상태이면 새로 도착한 태스크는 전달 큐에 저장하고 태스크를 이주시킬 상대 컴퓨터를 찾는 루틴인 location_routine()을 수행한다. Location_routine()은 지역 컴퓨터(local computer)가 갖고 있는 부하 표를 참조하여 부하 량이 가장 적고 저 부하 상태인 목적 컴퓨터(destination computer)를 찾는다. 이전된 태스크를 받아들이는 목적 컴퓨터가 결정되면 목적 컴퓨터의 부하 량도 알 수 있으므로 과부하 상태인 지역 컴퓨터의 전달 큐에 있는 여러 태스크 중 어떤 태스크들을 이전시킬 것인지를 결정하는 selection_routine()을 수행한다. 이때 이전시킬 태스크의 선택은 선택된 태스크들이 목적 컴퓨터에 이전됐을 때 부하의 하한 값 T1을 넘지 않는 범위로 한다.

Selection_routine()에 의해서 결정된 이전 가능한 태스크들을 집합 S로 표현한다. 이전될 태스크들이 결정되면 이 태스크들을 location_routine()에 의해서 결정된 목적 컴퓨터로 이전한다. 지역 컴퓨터와 목적 컴퓨터의 부하가 변했으므로 이들이 갖고 있는 information_routine()에 의해서 이 두 컴퓨터의 부하를 다른 컴퓨터들에게 브로드캐스팅(broadcasting)해서 알린다. 지역 컴퓨터는 진행하던 태스크의 수행을 계속한다. Transfer_routine()의 결과 Ci의 상태가 송신자가 아니면 새로 입력된 태스크를 서비스 큐에 삽입하고 컴퓨터 Ci는 남은 태스크의 수행을 계속한다.

Job-Receiver는 목적 컴퓨터가 이전된 태스크를 받았을 때의 동작을 기술한 알고리즘이다.

Algorithm : Job_Receiver
- Ci에서 보낸 태스크들 도착.
- 태스크들 서비스 큐에 삽입.
- 부하값 갱신.
- information_routine()에 의해 변화된 부하값 전송.

End{Job_Receiver}

2.2 파일 이전

현재까지 많은 파일 이전 방법들이 발표되었지만, 이 방법들은 파일 이전을 하기 위한 결정 함수가 복잡해서 이에 따른 오버헤드가 크다. 본 논문에서는 학습 오토마타를 사용한 파일 이전 방법을 채택한다. 이 알고리즘은 파일 이전을 위해서 컴퓨터 사이에 추가적인 메시지 전송 없이 각 컴퓨터가 유지하는 파일 상태 정보를 이용하여 파일 이전 여부를 결정한다.

본 연구에서 사용한 학습 오토마타는 고정 구조 스토케스틱 오토마타(Fixed Structure Stochastic Automata ; FSSA)로서 다음과 같이 정의된다[7-9].

FSSA = (α, S, β, F, G)

1) $\alpha = (\alpha 1, \alpha 2, \ldots, \alpha r)$: 행동들의 집합.
2) $S = \{S1, S2, \ldots, Sk\}$: 내부 상태 집합, $r \leq k$.
3) $\beta = \{0, 1\}$: 입력 집합 . 입력이 "1" 이면 처벌을 나타내고 "0"이면 보상을 나타낸다.
4) $F : S \times \beta \rightarrow S$: 상태 변환 함수.
5) $G : S \rightarrow \alpha$: 행동 선택 함수.

이 오토마타에서 α는 파일이 행할 수 있는 행동을 나타낸다. 즉 파일 이전 동작과 이전하지 않는 동작을 나타낸다. S는 파일이 가질 수 있는 상태들을 나타낸다. β는 입력으로 임의의 컴퓨터가 다른 컴퓨터에 있는 파일을 액세스하면 처벌 '1'을 주고 자신에게 있는 파일을 액세스하면 보상 '0'을 준다. F는 입력에 따라 파일의 상태를 변화시키는 변환 함수이고 G는 파일의 상태에 따라 행할 수 있는 행동을 선택하는 선택 함수이다.

M개의 컴퓨터로 구성된 분산 처리 시스템에서 N개의 파일 집단 {f1, f2, ... , fN}이 컴퓨터들 사이에 분산 저장되어 있다고 한다. 이 시스템에서 i번째 컴퓨터 Ci가 파일 fj를 액세스하기 위한 요구를 (Ci, fj)로 표현한다.

파일 액세스 요구 (Ci, fj)에 대해서 파일 fj가 컴퓨터 Ci에 존재하면 지역 액세스(local access)를 하므로 가능하면 이 파일을 컴퓨터 Ci에 계속 보유시키는 것이 유리하기 때문에 이에 따른 보상을 위해 '0'을 입력한다. 파일 fj가 컴퓨터 Ci에 존재하지 않으면 원격 액세스(remote access)를 하므로 파일 fj를 이전시키는 것이 유리하므로 처벌을 위한 '1'을 입력한다. 파일 액세스 요구 (Ci, fj)에 대해서 그 당시의 파일 fj의 상태를 Sj라 할 때 입력이 I인 경우에 대한 상태 변환 함수 값 F(Sj, I)는 다음과 같이 정의한다.

1) I = 0인 경우 (보상)
   F(Sj, 0) = Sj-1 , if j ≠ 1
   F(Sj, 0) = S1 , if j = 1
2) I = 1인 경우 (처벌)
   F(Sj, 1) = Sk
   k = min{파일 액세스 거리*가중치+j, N},
   if j ≠ N
   F(Sj, 1) = SN/2 , if j = N

이와 같이 학습 오토마타를 사용한 파일 이전 알고리즘은 비교적 간단히 기술되고, 수행 시간이 짧기 때문에 분산 운영체제에 쉽게 활용할 수 있고 이를 기술하면 다음과 같다.

```
Algorithm : LearningMigration
  For a sequence of Request(Ci, fj)
    If Ci = inprocessor(fj) Then /* 지역 파일 참조, fj는 Ci에 있다. */
      If Sj ≠ S1 Then     /* 보상 */
        · Sj := Sj-1
      EndIf
      Else   /* 원격 파일 참조, 파일 fj는 Ci에 있지 않다. */
       k := min{i+length*weighting_factor, N}
       Sj := Sk
       If Sj = SN Then      /* 우측 경계 상태 */
         · Sj := SN/2
         · 파일 fj를 컴퓨터 Ci로 이전한다.
         EndIF
      EndIf
  EndFor
End{LearningMigration}

Algorithm : Receive_File
   · 파일 fj가 이전돼 온다.
   · 파일 fj의 상태를 초기화한다.
End{Receive_File}
```

2.3 파일 이전과 부하 균등 알고리즘의 결합

2.1과 2.2에서 기술한 연산들로서 부하 균등 알고리즘과 파일 이전 알고리즘을 결합하는 것은 용이하다(알고리즘 Job-Sender참조). 이 알고리즘들은 분산 알고리즘으로서 각각의 컴퓨터에 저장된다. 부하 균등 알고리즘은 각 컴퓨터의 부하가 변화되는 시점, 즉 태스크가 도착했을 때나 태스크의 수행이 끝났을 때에 부하 균등을 수행하는 것이 좋지만 본 논문은 송신자 위주 정책이므로 부하가 증가하는 시점인 태스크가 도착했을 때 부하 균등 알고리즘을 수행한다. 파일 이전은 태스크의 수행 시 파일에 대한 액세스가 발생하면 파일상태표의 내용에 따라 파일 이전이 수행된다.

태스크는 각 컴퓨터에 독립적으로 도착하고 부하

균등은 태스크가 입력될 때 수행되고, 이전될 태스크는 이전시키고 그렇지 않은 태스크는 서비스 큐에 집어넣는다. 각 컴퓨터는 서비스 큐에서 한 개의 태스크를 가져와서 수행한다. 태스크 수행 중 파일 액세스를 위한 이벤트가 발생하면, 참조되는 파일이 저장되어 있는 컴퓨터에서 참조된 파일의 상태를 변화시킨다. 각 컴퓨터에서 지역 참조가 많이 발생하면 파일의 상태가 최소 내부 상태로 접근할 것이고 외부 참조가 많이 발생하면 파일의 상태가 우측 경계 상태에 도달하게 된다. 파일의 상태가 우측 경계 상태에 도달하면 파일은 가장 많이 참조한 컴퓨터에게 이전되고, 태스크는 계속 수행된다.

3. 수학적 모델

파일 이전을 고려한 부하 균등 모델을 검증하기 위해서 제안된 모델에 대한 수학적 모델을 제시하고, 이를 위한 가정과 매개변수를 설정한다. 각 컴퓨터는 서비스 큐와 전달 큐의 두개의 큐를 가지고 있다. 태스크가 들어올 때 서비스 큐가 비어 있으면 서비스 큐로 들어가 컴퓨터의 서비스를 받기 위해 기다리고 서비스 큐가 찬 경우는 전달 큐로 들어가서 대기하고 이 태스크가 이전되기 위하여 부하 균등 알고리즘을 수행하여 목적 컴퓨터를 찾고 태스크 이동을 수행한다. 이동된 태스크는 목적 컴퓨터의 서비스 큐로 들어가서 더 이상 이전되지 않는다. 전달 큐에 있는 태스크는 이전될 수 있지만 서비스 큐에 있는 컴퓨터는 이전될 수 없다. 한 태스크는 수행 도중에 파일에 대한 참조를 하므로 한 태스크의 수행 완료 시간은 태스크의 수행 시간과 수행 도중 태스크가 파일을 참조하는데 걸린 시간의 합이다. 따라서 임의의 컴퓨터 c에 도착한 태스크의 응답 시간을 나타내면 식(1)과 같다.

식 (1)은 Schaar 등[10]이 제안한 이더넷으로 연결된 시스템에서의 부하 균등에 관한 모델에서 파일 이전에 관한 사항을 고려한 확장된 수학적 모델이다. 여기서 R_c은 컴퓨터 c의 평균 응답 시간을 나타내고, V는 방문 확률을 나타내고, S는 서비스 시간을 나타내고, 그리고 Q는 큐의 평균 길이를 나타낸다. m은 전체 시스템이 갖는 컴퓨터 수를 나타낸다. 첨자 c, k, s 는 각각 지역 컴퓨터 번호와 서비스 해 주는 컴퓨터 번호 그리고 지역 컴퓨터의 상태를 나타낸다. g는 도착한 태스크가 전달될 확률을 나타내고, C는 압축비용을 나타내고, H_t는 태스크가 목적 컴퓨터를 찾기 위하여 기다리는 시간을 나타내며, H_n은 목적 컴퓨터를 찾으려고 했지만 찾지도 못하고 기다린 시간이고, DP는 파일 참조 시 네트워크 지연시간을 D는 파일 이전 시 네트워크 지연을 나타내고, DAP는 파일 참조 시 평균 디스크 액세스 시간을 나타내고 DA는 전체 파일 이전 시 평균 디스크 액세스 시간을 나타낸다. f_i^j는 컴퓨터i에 있는 파일j를 참조할 확률이고, M_i^j는 컴퓨터i에 있는 파일j가 이전될 확률이다. 어떤 컴퓨터에서의 태스크의 응답 시간은 이 태스크의 수행 시간과 파일 참조 시간의 합이다. 첫 번째, 두 번째, 세 번째

$$
\begin{aligned}
\text{식(1)} \cdots\ R_c \;=\; & \sum_{k=1}^{m}\sum_{s=1}^{3} V_{c,k,s}\times S_k\times(1+Q_{k,s}) \;+\; g(C+H_t+D) \;+\; V_{c,c,3}\times H_n \\
& +\; \sum_{j} f_c^j \times DAP + \sum_{i(i\neq c)}\sum_{j} f_i^j \times (1-M_i^j)\times(C+H_t+DP+DAP) \\
& +\; \sum_{i(i\neq c)}\sum_{j} f_i^j \times M_i^j \times (C+H_t+D+DA) \quad - \quad (1)
\end{aligned}
$$

항은 태스크가 지역 컴퓨터에서 수행되는 시간, 원격 컴퓨터에서 수행되는 시간 그리고 원격 컴퓨터에서 수행하려고 했지만 저부하인 목적 컴퓨터를 찾지 못해서 결국 지역 컴퓨터에서 수행된 시간에 각각의 확률을 곱한 값이므로 결국 한 태스크가 순수하게 수행되는 시간을 나타낸다. 네 번째, 다섯 번째, 여섯 번째 항은 태스크가 수행 중에 파일을 참조할 때 지역 컴퓨터에 있는 파일을 참조하거나 원격 컴퓨터에 있는 파일을 참조하거나 원격 컴퓨터에 있는 파일을 이전 받는 시간에 각각이 발생할 확률을 곱한 값들의 합으로서 한 태스크가 파일을 참조하는 시간이 된다.

식(1)에서 첫 번째 항은 서비스 시간과 서비스를 위해 기다리는 시간을 나타낸다. 이는 큐의 길이에 컴퓨터k의 상태가 s일 방문 확률의 곱이다. 두 번째 항은 전달 지연을 나타낸다. 태스크가 전달되기 위하여 먼저 태스크가 압축되어야 하고 목적 컴퓨터를 찾아야 하며 네트워크 지연 시간이 필요하다. 네트워크 지연 시간은 네트워크를 사용할 수 있을 때까지 기다리는 시간과 실제로 데이터가 전달되는 시간의 합이다. 그러므로 전달 시간은 이 세 가지 매개변수에 태스크가 전달될 확률을 곱하면 된다. 세 번째 항은 과부하 상태여서 태스크를 이전하여 수행하고자 하였으나 저부하인 컴퓨터가 없는 경우 할 수 없이 지역 컴퓨터에서 처리되는 것을 나타낸다. 네 번째 항은 지역 참조를 하는 경우로, 컴퓨터c에 존재하는 파일에 대한 액세스가 일어날 확률과 디스크 액세스 시간의 곱으로 나타낸다. 파일이 다른 컴퓨터로부터 참조될 때 파일의 상태가 우측 경계 상태가 아니면 파일이 참조되고 우측 경계 상태이면 파일이 이전된다. 다섯 번째 항은 원격 파일을 액세스하는데 소요된 시간을 나타낸다. 여섯 번째 항은 파일 이전이 발생할 때의 시간을 나타내며 파일 참조 확률과 파일이 이전될 확률과 이동에 필요한 시간의 곱으로 나타낸다. 태스크의 도착율과 서비스율은 각각 λ와 μ이다. 단지 λ와 μ만 주어지고 나머지 매개변수들은 계산된다. 태스크의 도착 율은 매개변수λ를 갖는 푸아송 분포를 갖는다. 또한 컴퓨터에서의 서비스 시간과 태스크 이전 시간은 평균이 1/μ과 1/ɣ인 지수 분포를 갖는다. 태스크들은 각 컴퓨터에서 선입선출 형태로 수행되고 이용 인자(utilization factor) ρ는 λ/μ이다. 임의 컴퓨터의 태스크 수가 k일 확률 P(k)는 (1-ρ)ρk이다.

$STATE(1)$, $STATE(2)$, $STATE(3)$은 컴퓨터의 상태가 각각 저 부하 상태, 정상 부하 상태, 그리고 과부하 상태일 때의 확률을 나타낸다. 컴퓨터가 각각의 상태에 있을 확률은 다음과 같다.

$STATE(1) = \sum_{k=0}^{T_1} P(k)$, 여기서 T_1는 하한 값.

$STATE(2) = \sum_{k=T_1+1}^{T_2-1} P(k)$, 여기서 T_2는 상한 값.

$STATE(3) = \sum_{k=T_2}^{\infty} P(k)$, 즉

$1-(STATE(1)+STATE(2))$.

만일 임의의 상태일 때의 큐의 길이가 Q1부터 Q2까지 이면, 큐 길이의 기대값 E(Q)는 다음과 같다.

$E(Q) = \dfrac{\sum_{k=Q_1}^{Q_2} k \times P(k)}{\sum_{k=Q_1}^{Q_2} P(k)}$, 여기서 P(k)는 큐의 길이가 k일 절대 확률이다.

따라서 임의의 컴퓨터 i에서 이 컴퓨터 i가 저 부하 상태일 때의 평균 태스크의 수$Q_{i,1}$은 $\dfrac{\sum_{k=0}^{T_1} k \cdot P(k)}{STATE(1)}$이고, 정상부하 상태일 때의 평균 태

스크의 수 $Q_{i,2}$는 $\frac{\sum_{k=T_1+1}^{T_2-1} k \cdot P(k)}{STATE(2)}$ 이고, 과부하 상태일 때의 평균 태스크의 수 $Q_{i,3}$은 $\frac{\sum_{k=T_2}^{\infty} k \cdot P(k)}{STATE(3)}$ 이다.

4. 컴퓨터 시뮬레이션 및 결과 분석

파일 이동을 고려한 부하 균등 모델을 검증하고, 이것의 성능을 분석하기 위해서 컴퓨터 시뮬레이션을 시행했다.

본 실험에서 가정한 시스템은 모든 컴퓨터가 동일한 하드웨어와 소프트웨어 구조를 가지고 있는 동형 시스템(Homogeneous system)이고, 각 컴퓨터는 독립적으로 동작하며 자신의 지역 메모리를 갖는다. 모든 컴퓨터는 각각 독립적으로 태스크를 생성하고, 각 컴퓨터마다 몇 개씩의 파일을 갖고 있으며 이 파일들은 다른 컴퓨터로 이동될 수 있다. 사용된 모든 큐는 선입선출로 하였다.

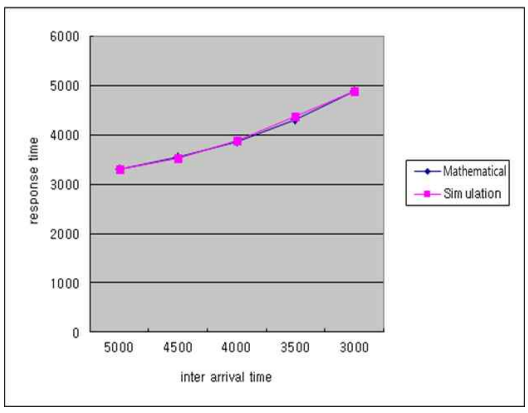
또한, 파일의 크기는 균일 분포(uniform distribution)로 하였고, 컴퓨터에서 태스크의 생성은 푸아송 분포(Poisson distribution)로 독립적으로 발생시킨다.

디스크의 평균 탐색 시간과 평균 회전 지연 시간은 각각 10.4msec와 6.95msec로 균일 분포(uniform distribution)를 갖는다[11]. 태스크는 생성된 후 중앙 처리 장치가 쉬고 있으면 중앙 처리 장치의 서비스를 받다가 파일에 대해 읽거나 쓰는 동작을 한다. 본 실험에서 사용한 매개변수들의 값은 <표 2>와 같다.

<표 2> 매개변수들의 값

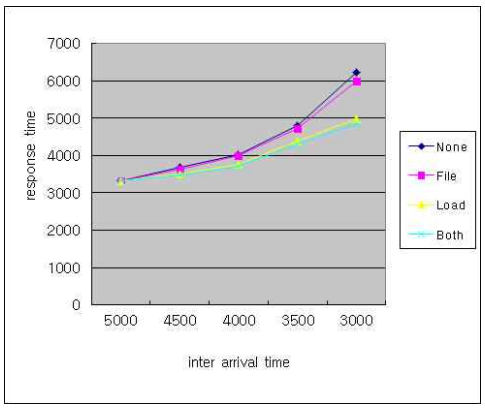
컴퓨터 수	40
컴퓨터 사이의 거리	20m
태스크의 평균 수행 시간	2000ms
태스크의 평균 크기	10Kbyte
파일 상태 수	10
디스크의 평균 탐색 시간	10.4ms
디스크의 평균 회전 지연 시간	6.95ms
데이터 전송 속도	100Mbit/sec
전송 매체에 의한 전달 지연 시간	0.005ms/Km

본 실험에서 제안된 모델을 검증하기 위해서 수학적 모델과 제안된 모델의 시뮬레이션 결과를 비교했다. <그림 1>은 수학적 모델(Mathematical)과 시뮬레이션 모델(Simulation)에서 태스크의 응답 시간을 측정한 것이다. 여기서 두 경우의 결과가 대체로 비슷한 것을 알 수 있고, 이것으로 미루어 보아 제안된 모델이 대체로 타당하다고 할 수 있다. <그림 2>는 분산 시스템에서 부하 균등과 파일 이전을 다 수행한 경우(Both)와 부하 균등만 수행한 경우(Load balancing)와 파일 이전만 수행한 경우(File migration) 및 두 가지 다 수행하지 않은 경우

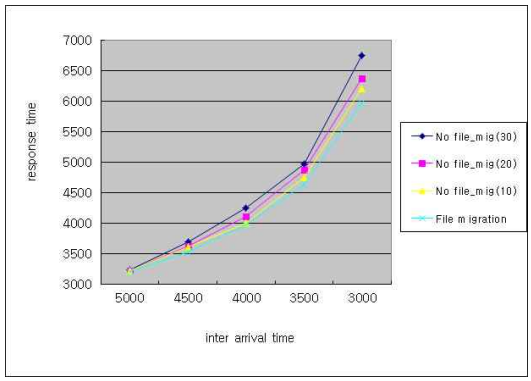


<그림 1> 수학적 모델과 시뮬레이션 모델의 결과

(None)에 대한 태스크의 평균 응답 시간을 측정한 결과를 나타낸 것이다. <그림 2>에서 파일 이전을 고려한 부하 균등과 부하 균등만 시행한 경우의 결과가 비슷하고, 파일 이전만 하든가 혹은 파일 이전이나 부하 균등을 모두 시행하지 않는 경우의 결과가 비슷한 것으로 나타났다. 여기에서 파일 이전의 효과는 태스크를 처리하는 환경에 의존하는 것으로서 원격 파일에 대한 접근이 많고, 특정 컴퓨터에 존재하는 파일에 대한 접근이 많을수록 파일 이전에 따른 성능 향상의 효과는 커진다고 할 수 있다. <그림 3>은 파일 이전에 대한 효과를 평가하는 실험 결과이다. 이 실험에서 파일 이전을 시행하는 경우(File migration)와 파일 이전을 시행하지 않는 경우(No file_mig)를 구분했고, 파일 이전을 하지 않는 경우에 각 태스크가 접근하는 파일 자료량을 원래 파일 크기에 대한 비율로 정해서 실험했다. <그림 3>은 파일 이전을 하지 않는 경우에 접근하는 파일 자료량이 많아질수록 태스크의 평균 응답 시간은 증가하게 되고, 평균 파일 자료 접근 량이 10%만 되어도 파일 이전하는 것이 유리함을 보여준다.



<그림 2> 평균 도착 시간에 따른 응답 시간



<그림 3> 파일 참조량이 다른 경우의 결과

5. 결 론

본 논문에서는 분산 시스템에서의 부하 균등과 파일 이전을 함께 고려한 모델을 제시하고, 수학적 모델을 사용해서 이것을 검증했다. 제안된 모델의 성능을 평가하기 위해서 다양한 조건에서 분산 시스템에 도착한 태스크에 대한 평균 응답 시간을 측정했다. 평균 응답 시간의 측정 결과에 따르면, 파일 이전을 고려한 부하 균등이 부하 균등만 시행한 경우와 비슷한 효과를 나타냈지만, 태스크의 파일 접근 자료량이 많을수록 파일 이전의 효과는 커짐을 알 수 있다. 그러나 원격 파일에 대한 접근이 거의 없는 경우에는 파일 이전에 따른 평균 응답 시간의 단축 효과는 없기 때문에 제안된 모델은 원격 파일 접근이 많은 경우에 더욱 효과적이라 할 수 있다.

또한, 제안된 모델에서는 각 컴퓨터가 처리 환경에 따라서 동적으로 부하 균등을 수행하거나, 파일 이전을 수행할 수 있기 때문에 부하 균등과 파일 이전을 함께 고려함으로써 다양한 분산 처리 환경에서 효과적으로 사용될 수 있을 것으로 기대된다.

참고문헌

[1] F. C. H. Lin and R. M. Keller, "The Gradient Model Load Balancing Method," IEEE Trans. on Software Eng. Vol. SE-33, No. 1, Jan. 2007, pp. 32-38.

[2] L. M. Ni, C. W. Xu, and T. B. Gendreau, "A Distributed Drafting Algorithm for Load Balancing," IEEE Trans. on Software Eng., Vol. SE-31, No. 10, Oct. 2005, pp. 1153-1161.

[3] M. Schaar, K. Efe, L. Delcambre and S. Koppolu, "Heuristic Algorithms for Adaptive Load Sharing in Local Networks," Proc. of The First Int. Conf. on Systems Integration, IEEE Computer Society Press, 2000, pp. 367-379.

[4] A. Hac, X. Jin, "Dynamic Load Balancing in a Distributed System Using a Decentralized Algorithm," Proceedings The 17th International Conference on Distributed Computing Systems, 1997, pp. 170-177.

[5] O. R. Liu Sheng and B. Gavish, "Dynamic File Migration in Distributed Computer System," Comm. ACM, Vol. 50, No. 2, Feb. 2007, pp. 122-139.

[6] B. W. Wah, "File Placement on Distributed Computer Systems," IEEE Comput, 37, 1, Jan. 2004, pp. 23-30.

[7] K. S. Narendra and M. A. L. Thathachar, "Learning automata - A survey," IEEE Trans. Syst., Man, Cybern., Vol. SMC-4, 2004, pp. 323-334.

[8] B. J. Oommen, D. C. Y. Ma, "Deterministic Learning Automata Solutions to the Equipartitioning Problem," IEEE Trans. Comp. Vol. 37, No. 1, Jan. 2008, pp. 2-13.

[9] B. J. Oommen and E. R. Hansen, "List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations," Siam J. Comput., Vol. 26, 1997, pp. 705-716.

[10] M. Schaar, K. Efe, L. Delcambre, and L. N. Bhuyan, "Load Balancing with Network Cooperation," Proc. of the 2001 IEEE Int. Conf. on Distributed Computing Systems, 2001, pp. 328-335.

[11] B. Nelson, Y. P. Cheng, "How and Why SCSI is Better than IPI for NFS," Proc. USENIX Winter 1998 Technical Conference, 1998, pp. 253-270.

[12] 이대식, "모바일 인스턴스 메신저를 이용한 양방향 검색 알고리즘의 설계 및 구현," 디지털산업정보학회지, 제11권, 2호, 2015, pp. 55-66.

■ 저자소개 ■

문원식
(Moon Wonsik)

1995년 3월~현재 평택대학교 융합소프트웨어학과 교수
1995년 2월 동국대학교 컴퓨터공학과(공학박사)
1989년 2월 동국대학교 컴퓨터공학과(공학석사)
1987년 2월 동국대학교 수학과(이학사)

관심분야 : 병렬처리, 분산알고리즘
E-mail : moonws@ptu.ac.kr

논문접수일 : 2017년 11월 20일
수 정 일 : 2017년 12월 14일
게재확정일 : 2017년 12월 18일