韓國 컴퓨터情報學會 論文誌
第11卷 第4號, 2006. 9.

2006-11-4-1-9

인터넷 기반 분산컴퓨팅환경에서 자원할당을 위한 피어 가용길이 예상 기법

김 진 일*

A Peer Availability Period Prediction Strategy for Resource Allocation in Internet-based Distributed Computing Environment

Kim Jin-Il *

요 약

과학 기술이 발전함에 따라 대량의 정보를 처리하기 위해 대두된 인터넷을 기반으로 하는 분산 컴퓨팅 환경은 대규모의 독립된 자원을 공유하여 과학 연구와 같은 문제를 해결하기 위한 구축된 환경이므로, 사용자 작업을 효율적으로 할당하기 위한 스케줄링 알고리즘이 필요하다. 현재까지 여러 스케줄링 알고리즘이 연구되어 왔지만, 대부분 피어의 자율성을 고려하지 않는 문제점을 가지고 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 인터넷기반 분산 컴퓨팅 환경에서의 피어 가용길이 예상 기법을 제안하였다. 또한 인터넷기반 분산 컴퓨팅환경에서 사용되는 SRTFIT 알고리즘에 적용하여, 시뮬레이션을 통하여 제안된 기법이 단순한 예상기법보다 성능이 뛰어남을 보였다.

Abstract

Internet-based distributed computing environment have been developed for advanced science and engineering by sharing large-scale resources. Therefore efficient scheduling algorithms for allocating user job to resources in the Internet-based distributed computing environment are required. Many scheduling algorithms have been proposed, but these algorithms are not suitable for the Internet-based Distributed computing environment. That is the previous scheduling algorithm does not consider peer self-control. In this paper, we propose a Peer Availability Period Prediction Strategy for Internet-based distributed computing environment and show that our Strategy has better performance than other Strategy through extensive simulation.

▸ Keyword : 분산컴퓨팅환경(Distributed Computing Environment), 피어(Peer), 자원할당(Resource Allocation), 가용길이(Availability Period)

• 제1저자 : 김진일
• 접수일 : 2006.08.07, 심사일 : 2006.08.17, 심사완료일 : 2006.09.20
* 배재대학교 교양교육지원센터 교수

Ⅰ. 서 론

컴퓨터 통신 환경과 웹 기술의 급속한 발전으로 인터넷을 이용하는 사용자가 기하급수적으로 증가되고 있다. 하지만, 인터넷 사용자의 대부분이 문서작성 및 정보검색, 게임, 메일 등의 사용으로 컴퓨팅 파워의 90% 이상이 유휴 자원으로 낭비되고 있는 것이 현실이다. 특히 CPU의 성능과 메모리 용량은 증가하는 반편에, 실제 CPU의 이용률은 낮은 것이 현실이다. 그러므로 이처럼 초고속화된 인터넷 환경과 유휴 PC 자원의 잠재적인 능력을 적극적으로 활용할 수만 있다면, 짧은 시간에 대규모의 데이터 처리가 요구되는 바이오, 의료, 기상관측, 지능형 로봇 분야의 고성능 컴퓨팅 요구를 충족시킬 수 있다[8,9].

이런 이유로, 최근에는 컴퓨팅 파워를 요구하는 응용분야에 값비싼 고성능의 슈퍼컴퓨터를 대신하여, 인터넷상에 분산되어 있는 컴퓨터를 이용하여 문제를 해결하기위한 다양한 연구가 활발하게 진행되고 있다[1,2]. 이러한 분산 시스템은 자원들과 연산 처리 주체들 간의 효과적인 의사교환을 수행하기 위해서 초창기 메시지 기반 컴퓨팅 방식부터 시작하여, 클라이언트/서버 기반의 컴퓨팅 방식, 그리고 현재의 P2P기반 컴퓨팅 방식으로 발전되어 왔다. 현재, 이에 대한 연구가 국내외적으로 활발하게 진행 중에 있으며 이 연구의 응용분야도 점차 영역을 넓혀가고 있다. 그러나, 기존의 분산 컴퓨팅과는 달리 인터넷을 기반으로 하는 분산 컴퓨팅 환경에서는 각 피어들이 자유롭게 연산에 참여 할 수 있고 언제라도 사용자가 임의로 연산 작업 중간에 탈퇴를 할 수 있는 인터넷 환경이기 때문에 기존의 스케줄링 정책을 그대로 사용하기는 어렵다. 즉, 작업이 임의적으로 시작되고 중지되는 현상이 발생할 수도 있고 각 피어의 자원이 분산응용을 위해서만 쓰는 것이 아니기 때문에 개인적인 용도로의 사용으로 인해 일시적으로 중지되는 경우가 발생할 수 도 있다. 이와 같은 피어의 자율성은 연산 지연 현상을 초래하거나 심지어는 부분적으로 연산을 손실하는 결과를 초래한다[1]. 더욱이, 응용 작업간의 연산 의존성이 발생하는 경우에는 연산 진행 중에 탈퇴에 대한 처리가 더욱 복잡해진다. 따라서, 실시간 작업에 대한 총 실행시간을 정확하게 예측할 수 없기 때문에 인터넷기반 분산컴퓨팅 환경에서 자원 할당의 효율성이 떨어지는 문제점이 있다.

본 논문에서는 이러한 문제점을 해결하기 위하여 컴퓨터 사용자의 이용시간이 일정한 패턴을 가진다는 점에 착안하여 이 정보를 이용하여 피어 가용길이 예상 기법(PAPA, Peer Availabilty Prediction Strategy)을 제안한다. 제안된 기법은 개인적인 사용으로 인해 작업이 중지될 가능성이 있는 피어를 미리 예측함으로써 재처리 기회의 최소화로 수행시간을 감소시킬 수 있다.

Ⅱ. 작업 할당 기법

스케줄링 알고리즘은 크게 2가지로 구분할 수 있는 데, 단순한 우선순위 알고리즘과 적응 알고리즘이다. 만약 피어 가용길이 정보와 태스크 예상 시간이 주어졌을 때, 피어들의 자원을 계속해서 사용할 수 있고, 재처리를 고려하지 않는다고 가정한다면, SRT(Shortest Remaining Time) 우선순위 스케줄링 알고리즘을 사용하는 것이 유리하다. 이 알고리즘은 응용에 적합하도록 스케줄링되면, 최적의 반환 시간을 가질 수 있다. 반면에, 작업 전체 길이에 대한 정확한 태스크 길이 정보와 계산될 태스크 수에 대한 정보를 요구한다. SRT 알고리즘과 비슷하게, 작업 길이에 대한 정확한 정보를 요구하는 SPT(shortest processing time)알고리즘도 있다.

임의의 작업 또는 태스크에 대한 길이 정보가 없는 경우에는 라운드로빈(round robin, ROR)이나 SET(Shortest Elaped Time)과 같은 알고리즘을 사용한다. SRT 알고리즘과 비슷하게, 이 알고리즘들은 짧은 작업이 먼저 완료되는 경향이 있다. 만약 모든 태스크들이 거의 비슷한 길이를 가지고 있다면, LET(longest elapsed time)알고리즘이나 FCFS (first-come first-served) 알고리즘은 SRT 알고리즘과 거의 유사하게 동작한다[3,4,5].

SRT 알고리즘의 변형인 SRTFIT 알고리즘은 기존의 SRT 우선순위 알고리즘에 작업 우선순위를 부여한 형태이다. 만약 다음에 수행할 작업을 스케줄링할 때, 피어의 평균 가용길이가 작업이 예상된 태스크 시간을 초과하면 그 피어에 주어진 작업을 스케줄링한다. 이것은 할당된 작업이 예상된 가용길이 내에 태스크의 수행을 완료할 수 있음을 의미한다. 만약 가장 높은 우선순위를 가지는 작업이 적합하지 않으면, 그 다음으로 높은 우선순위를 가진 작업을 할당한다. 다시말해, 기존의 우선순위 알고리즘들이 작업의 우선순위를 고려하는 경우에는 SETFIT와 비슷하게 만들면 된다[6,7]

본 논문에서는 제안된 피어 가용길이 예상기법을 평가하기 위하여 전역 스케줄러로는 여러 스케줄링 알고리즘들 중에서 SRTFIT 알고리즘을 사용한다.

III. 피어 가용길이 예상 기법

기존의 스케줄링 기법에서는 피어 자율성 결함과 피어 간섭 결함과 같은 결함을 발생하게 되면 이에 대한 처리를 수행하는 스케줄링 기법을 제안하고 있다. 하지만 이러한 문제를 미리 예방하기 위한 방법을 고려하지 않았기 때문에 재처리 문제가 발생하게 된다. 그래서, 전체 작업에 대한 연산이 더 이상 진행되지 않아서 전체 연산이 지연되고 봉쇄되는 현상이 발생하게 되어 전체적인 작업 시간이 지연되는 문제점이 있었다. 그러므로, 피어 결함을 예방하고 재처리 과정을 되풀이 하지 않도록 피어 가용길이 예상기법을 제안한다. 이를 위해 먼저 인터넷 기반 분산컴퓨팅 환경에서 피어 자원의 가용길이, 신뢰성, 작업의 의존도 등이 반영된 피어 ID와 작업 ID 기법을 제안한다[8].

피어 ID와 작업 ID는 각 피어들을 구분하기 위해 사용되며 피어의 특성에 따라 작업 분산을 위한 피어 작업 그룹을 구성한다. 작업을 분배하는 과정에서 피어의 특성을 고려하여 피어 ID를 부여함으로써 분산 응용 프로그램의 요구에 맞는 작업분배 기능을 수행할 수 있다. 피어 ID는 피어가 피어 등록 이름으로 시스템에 자신을 등록한 후, 피어가 작업에 참여하는 경우에 부여되는 식별자를 나타낸다. 피어 ID는 피어의 특성, 즉 성능, 위치, 가용길이, 신뢰도, 사용패턴 등을 바탕으로 피어 ID 부여 함수에 의해 새로운 피어 ID를 부여 받는다.

$$m\_nPeerId = SelectPeerId(PeerList, Method) \quad \cdots (1)$$

$$Method = (Performance, Location, Available\_Period, Trust, UserPattern)$$

식(1)에서, *PeerList*는 피어의 목록을 나타내고, *Method*는 피어의 성능, 위치, 가용길이, 신뢰도, 사용자패턴에 대한 선택 정책을 나타낸다.

분산 작업을 수행하는 경우, 피어 ID 함수는 먼저 피어의 위치에 따라 지리적으로 같은 지역이나 서로 인접한 지역에 속하는 피어들을 하나의 그룹으로 묶은 다음에 피어의 성능, 가용길이, 신뢰도, 사용패턴에 따라 피어 ID를 부여하도록 한다. 피어의 성능은 피어가 해당 작업을 수행하는 데 소요되는 연산 시간과 관련이 있고, 피어의 위치는 참여하는 피어의 위치에 따라 작업을 할당함으로써 통신비용의 절감 효과와 지역별로 작업을 할당할 수 도 있다. 피어의 가용길이는 피어가 결함이나 연산 중단을 고려하여 얼마나 많은 시간 동안에 주어진 태스크 수행에 참여하는 가를 나타내는 기여도를 말한다. 피어의 신뢰도는 피어가 수행한 태스크 결과에 대한 정확성과 관련된다. 사용 패턴은 피어가 일정한 시간대에 반복적으로 사용 패턴을 의미한다.

수행되어야 하는 응용작업은 태스크 단위로 나뉘어 지는 데, 이때 작업 ID는 작은 크기의 단위 작업에 부여되는 식별자를 의미한다. 작업 ID를 부여하는 방법은 정적 ID 부여 방법과 동적 ID 부여 방법이 있다. 여기에서는 작업을 태스크 단위작업들로 나눌 때 피어의 특성에 따라 서로 다른 크기로 나누는 동적 ID 방법을 사용한다[8].

피어그룹은 비슷한 성능, 위치, 가용길이, 신뢰도, 사용패턴을 가진 피어들의 집합에 대한 식별자이다. 즉, 피어의 자원의 제공시간, 위치, 가용길이, 신뢰도, 사용자 패턴을 기반으로 그룹을 만든 다음, 의존성이 있는 작업들을 동일 피어 그룹에 할당함으로써 태스크의 효율적인 분산을 지원할 수 있다. 아래와 같이 피어 그룹 ID 부여 함수 SelectPGroupId는 피어 ID 부여 함수에 의해 할당된 피어 ID를 기반으로 그룹의 크기에 맞게 그룹으로 구성한 후 m_nPeerGroupId를 부여한다.

$$m\_nPeerGroupId = SelectPGroupId(PeerList.PeerId, GroupSize) \quad \cdots (2)$$

식(2)에서, *PeerList.PeerId*는 피어 목록에 소속되는 피어의 피어 ID를 나타내고, *GroupSize*는 그룹의 크기를 나타낸다.

시스템 자원을 최대한 활용하기위한, 효율적인 스케줄러는 시스템 내에 작업이 있을 때, 어떤 작업이 사용 가능한 피어를 이용할 준비가 되어 있는 지를 보증해야 하고 또한 재처리도 고려해야 한다. 이상적인 스케줄러는 예상한 피어 가용길이보다 크지 않은 길이를 가지는 태스크 작업을 피어에 할당하는 것이다. 이것은 한 태스크가 재처리 발생하지 않고 각 피어가용길이 동안에, 계산적인 처리가 완료됨을 의미한다.

작업 태스크 시간을 피어 가용길이에 대응하기 위하여, 스케줄러는 각각의 작업을 위한 태스크의 길이를 예상하기 위한 방법뿐만 아니라 시스템 내의 피어에서 유효한 기간의 길이를 예상하기 위한 방법을 가지고 있어

야 한다. 피어의 가용길이를 예상하기 위하여, 먼저, 관찰하는 방법이다. 피어는 하루 중 예상할 수 있는 시간에 유휴상태가 되는 경향이 있다. 일반적으로 사용자는 피어 즉, 자원제공 PC을 밤새도록 거의 사용하지 않을 뿐만 아니라 점심시간 동안에도 사용하는 경우가 드물다. 그리고, 어떤 사용자는 매일 같은 시간대에 컴퓨터를 사용하는 경향이 있고, 특히, 주말과 휴일에 유휴상태가 되는 경우가 많다. 만약, 어떤 사용자가 유휴상태가 예정된 시간이 되었음에도 불구하고 작업을 계속하고 있다면, 그 사용자는 갑자기 작업을 멈추는 것이 아니라 계속해서 작업을 하도록 한다. 이러한 관찰을 이용하여, 분 단위, 시간 단위, 하루 단위, 일주일 단위로부터 피어 활동성의 상호 연관성을 찾을 수 있다. 이러한 과정 속에서 예전의 사용 패턴(historical usage pattern)에 대한 중요한 상호 연관성을 발견할 수 있다.

피어가 정상적으로 동작하면서 주어진 태스크를 수행할 수 있는 확률을 피어 가용길이 P_a라고 정의한다.

$$P_a = \mu \frac{MAT}{MAT + MFT} \quad \cdots\cdots (3)$$

식(3)에서, *MAT*는 평균 자원제공 시간(Mean Avaible Time)을 의미하고, *MFT*(Mean Failure Time)는 평균 자원제공 실패시간을 의미한다. 그리고, μ는 이전 단계에서의 피어가용 길이 사용패턴을 의미한다.

작업을 스케줄링하기 위해서, 다음 n 분에 대한 피어 가용길이를 예상하는 것이 중요하다. 여기서, n은 스케줄러의 기간을 나타낸다. 아래에서 소개되는 4 가지 예상 기법들은 다음에 수행 가능한 피어의 가용길이 n 분은 이전에 사용한 n분과 정확하게 일치할 것이라고 예상한다. RECENT_HIST 기법은 다음에 수행 가능한 피어의 가용 길이 n 분이 마지막에 수행한 피어의 가용 n 분과 정확하게 같다고 예상한다. 예를 들면 만약 이전의 20분(n=20)에서, 평균 피어가용길이가 4분이라면, RECENT_HIST은 다음 20분 동안 평균 피어가용길이 길이가 4분일 것이라고 예상하는 것이다. DAY_HIST기법과 WEEK_HIST 기법은 예상을 위하여 다른 예전 윈도우를 사용한다는 것을 제외하고는 RECENT_HIST기법과 비슷하다. DAY_HIST는 다음 n 분이 전날 같은 시간의 n 분의 예전 윈도우와 정확하게 같다고 예상한다. WEEK_HIST는 다음 n 분이 지난 주 같은 시간의 n 분의 예전의 윈도우와 정확하게 같다고 예상한다. LENGTH_LAST는 다음 n 분에서 평균 피어가용길이의 길이는 마지막 피어가용길이 길이와 같을 것이라고 예상하는 기법이다.

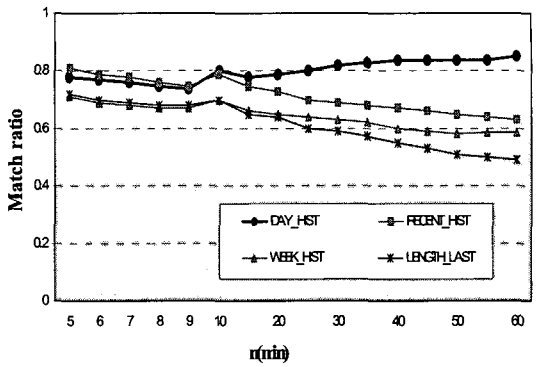

그림 1. 단순한 피어 가용길이 예상 기법
Fig. 1 Peer Availability Period Prediction Strategy

그림 1과 같이, 다음 n 분의 피어 가용길이를 예상하는 가장 좋은 방법은 10분이하일 경우(n≤10)에는 최근 이전의 n 분을 예상하는 RECENT_HIST 경우가 가장 좋은 결과를 나타내었고, 10분 이상일 경우(n>10)에는 피어 가용길이가 전날 같은 시간대를 이용하는 DAY_HIST 경우가 최상의 결과를 나타냈었다. 가장 최악의 예상 방법은 WEEK_HIST이다. 피어 가용길이를 예상할 필요가 있는 스케줄링 알고리즘은 n이 작을 때는 최근 정보를 활용하고 n이 클 경우에는 전날 정보를 이용하면 우수한 알고리즘이 될 수 있음을 알 수 있다.

본 논문에서는 이러한 최근 정보와 전날 정보를 활용한 예상 방법을 기반으로 좀 더 우수한 알고리즘을 만들기 위해서 작업을 할당할 때, 가장 적합한 피어를 찾도록 하는 방법을 추가하고자 한다.

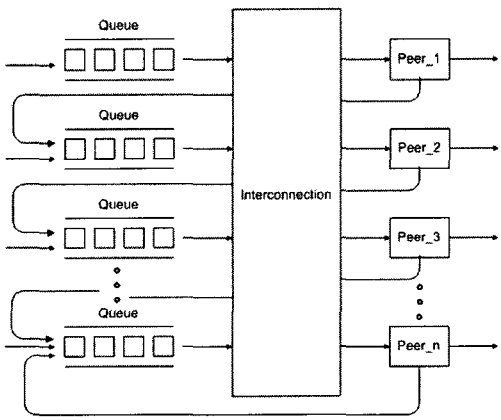

그림 2. 다중 큐를 이용한 작업할당
Fig. 2 Job Allocation using Multi-Queue

일반적으로, 스케줄링 기법은 짧은 작업에 우선권을 주어야 한다. 그리고 가능한 빨리 작업의 성격을 파악하여 그 성격에 맞게 스케줄링을 해야 한다. 그림 2와 같은 다중 큐를 이용한다면 이러한 목적들을 모두 달성할 수 있다. 각각의 피어는 처리할 작업에 대한 개별적인 큐를 가지고 있다. 각각 피어에 대한 이전의 가용길이에 대한 평균 길이를 이용하여 다음에 그 피어에 대한 가용길이를 예상한다. 그런 다음, N개의 피어에 대한 예상 가용길이를 소트하여 가장 작은 길이를 가지는 것부터 가장 긴 것으로 순서를 부여한다. 그리고 태스크들의 예상 처리시간도 마찬가지 방법으로, 그 피어에서 처리된 태스크 시간의 평균 시간을 다음 태스크 시간으로 예상한다. 또한 N개의 태스크 예상 시간도 가장 작은 길이로부터 오름차순으로 순서를 부여한다. 그런 다음에, 같은 순서 번호를 가지는 피어에 태스크의 예상 시간을 할당한다. 또한 외부 스케줄러는 프로그램을 분산하는 경우에 피어에 대한 태스크의 예상 시간 내에 처리할 수 있는 피어에 작업을 할당 한다. 만약, 각 태스크가 각 피어에 할당된 가용길이 이내에 처리가 완료되지 않았다면 그 태스크는 번호를 증가하여 다음 단계의 큐로 배치된다. 그리고 마지막 단계에 있는 태스크들은 완성될 때까지 기다린다. 태스크가 낮은 단계로 내려갈수록 태스크에 대한 피어의 가용길이가 커진다. 하지만 큐내에서는 FIFO형태로 이동한다.

IV. 실험결과 및 고찰

4.1 실험환경

인터넷기반 분산 컴퓨팅 시스템은 1000대 정도의 에이전트가 동시 접속하여 사용되며 분산응용 작업을 수행하는데 있어 특별한 요구 조건이 없고 시스템 성능을 유지할 수 있는 수준으로 설정하는 데, 자원관리 및 스케줄링 서버의 경우, CPU는 셀러론 1.2 GHz * 2, Memory는 1G Byte이상, HDD는 300 GByte 이상, NIC은 100 Mbps급 이상으로 1 Gbps를 권장한다. 그리고, OS는 Windows 2000 서비스 팩 3이상으로 구성한다.

또한, 피어들은 일반적으로 PC정도의 시스템에 HDD 용량과 RAM을 보강한 정도의 시스템으로 구성하는 데, CPU는 셀러론 1.2 GHz, Memory는 512 MByte 정도, HDD는 100 GByte, NIC은 100 Mbps급, SNIC 그리고 OS는 Windows 2000 또는 Windows XP Professional 이상으로 구성한다. 작업 스케줄링 시뮬레이션을 위해 사용된 알고리즘은 그림 3와 같다.

앞 절에서 설명한 바와 같이, 기존의 피어 가용길이 예상 기법은 RECENT_HIST, DAY_HIST인 경우에 가장 좋은 결과를 보였다. 그러므로, 이 두 가지의 경우에 대해서만 제안한 피어 가용길이 예상 기법과 비교한다. 그리고, 시스템의 전반적인 스케줄러 중에서 피어 가용 길이 예상 기법을 중심에 두어 다른 스케줄러 부분은 기존의 시스템 처럼 동작을 한다고 가정을 한다. 또한, 전역 스케줄러로는 여러 스케줄링 알고리즘들 중에서 SRTFIT 알고리즘을 사용한다는 가정한다.

```
각 피어는 ACTIVE[peer] = 0에서 시작
각 시간 단위에 대해
  • 한 job은 시스템 job 큐에 들어갈 수도 들어가지 않을 수도 있다.
  • 각 피어는 스케줄러에 의해 job 큐로부터 그것과 관계된 한 job JOB[peer]를 가진다.
 한 job은 같은 시간에 여러 피어에 할당 가능.
  • 각 피어에 대해
  - 만약 JOB[peer]가 전에 할당된 피어에 같은 job이 아니라면,
   ACTIVE[peer] = 0
  - If AVAILABLE[peer] then
      ACTIVE[peer] = ACTIVE[peer] + 1
   else ACTIVE[peer] = 0
  - If ACTIVE[peer] ==JOB[peer].taskTime then
   * JOB[peer].timeRemaining
     = JOB[peer].timeRemaining -
     JOB[peer].taskTime
   * ACTIVE[peer] = 0
  - If JOB[peer].timeRemaining == 0 then
   JOB[peer]는 시스템 job 큐 로 부터 제거
```

그림 3. PAPA 스케줄링 시뮬레이션 알고리즘
Fig. 3 PAPA scheduling simulation algorithm

4.2 실험결과 및 분석

사용된 데이터는 2주에 걸쳐 모았으며, 피어의 키보드와 마우스는 적어도 5분 동안 유휴상태에 놓이게 된다고 가정한다. Match ratio는 실제 평균 피어가용길이의 허용범위 내에 속하는 성공률이다.

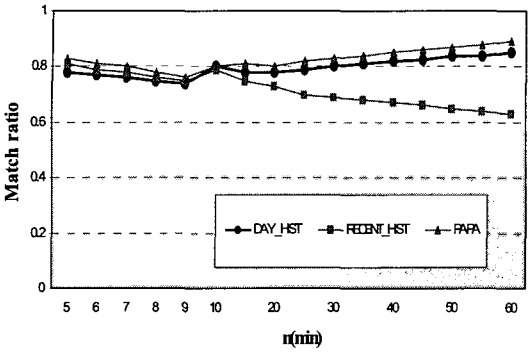

그림 4. 단순 예상기법과 제안된 기법의 비교
Fig. 4 Compare of Sample Prediction Strategy and Proposed Strategy

그림 4은 n이 변할 때, 다음의 n 분에 대한 평균 피어가용길이에 대한 RECENT_HIST, DAY_HIST 예상기법과 제안된 피어가용길이 예상 기법에 대한 성공률을 비교하여 보여준다. 제안된 피어가용길이 예상기법은 n이 작을 때는 최근 정보를 활용하고 n이 클 경우에는 전날 정보를 이용하였기 때문에 n의 크기에 상관없이 대체로 우수한 기법임을 알 수 있다. 다음 n분에 대해서 RECENT_HIST나 DAY_HIST와 같이 마지막 n분에 대한 평균 피어 가용길이의 길이가 다음 평균 피어 가용길이가 되는 고정적인 기법보다는 동적으로 각 피어에 대한 예상 피어 가용길이에 적합한 태스크의 시간을 예상함으로써 재처리의 기회가 줄어들게 되어 우수한 예상기법임을 알 수 있다. 물론, 여기서 제안된 기법이 비록 제한된 환경에서 이루어졌지만 실제 *Korea@Home*과 같은 대규모의 인터넷 기반 분산 컴퓨팅환경의 시스템에 적용했을 때에도 유사한 결과를 얻을 수 있으리라 사료된다. 또한, 기존의 단순한 예상 기법보다 제안된 자원할당의 기법을 작업할당 알고리즘에 적용하면 기존의 단순한 알고리즘에 비해 더 우수한 알고리즘이 될 수 있다. 이러한 기법은 피어의 유동적인 상태 변화에 적합한 적응 기법이다. 하지만 기존의 단순한 기법에 비해서 오버헤드가 크고 부가적인 비용이 든다. 하지만 시스템 내의 상황 변화에 무척 민감하게 반응하므로 오버 헤드의 증가는 그만한 가치가 있다고 본다.

V. 결 론

기존의 분산 컴퓨팅과는 달리 인터넷을 기반으로 하는 분산 컴퓨팅 환경에서는 작업 수행에 참여하는 피어들이 자유롭게 연산 참여할 수 있고 언제라고 사용자가 임의의 연산 작업 중간에 탈퇴를 할 수 있는 자유로운 인터넷 환경이기 때문에 기존의 스케줄링 정책을 그대로 사용하기는 어렵다. 즉, 피어의 자율성은 연산 지연 현상을 초래하거나 심지어는 부분적으로 연산을 손실하는 결과를 초래한다. 따라서, 실시간 작업의 총 실행시간을 정확하게 예측할 수 없기 때문에 인터넷기반 분산컴퓨팅 환경에서 자원할당의 효율성이 떨어지는 문제점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 컴퓨터 사용자의 이용시간이 일정한 패턴을 가진다는 점에 착안하여 이 정보를 이용하여 피어 가용길이 예상 기법(PAPA, Peer Availabilty Prediction Strategy)을 제안하였다. 제안된 기법은 개인적인 사용으로 인해 작업이 중지될 가능성이 있는 피어를 미리 예측함으로써 재처리 기회의 최소화로 수행시간을 감소시킬 수 있다. 결과적으로, 기존의 단순한 알고리즘에 비해 적응 알고리즘은 자원 할당기법에 따라 더 강력한 알고리즘이 될 수 있는 가능성을 보였다. 하지만 기존의 단순한 기법에 비해서 오버 헤드가 크다는 단점도 있다. 하지만 시스템 내의 상황 변화에 무척 민감하게 반응하므로 그다지 문제가 되지 않으리라 본다.

향후에는 제안된 기법을 적용한 여러 가지 적응 알고리즘에 대한 평가가 이루어질 필요가 있다. 분산컴퓨팅 작업을 스케줄링하는 문제는 분산된 시스템에서 피어의 형태과 유휴 자원의 효율적인 사용 등을 포함하여 많은 문제와 연관되어 있다. 그러므로, 인터넷 기반 분산 컴퓨팅 환경에서 작업을 스케줄링하기 위한 최적의 알고리즘을 개발하기 위해서는 여러 가지 가능성을 가진 태스크와 여러 가지 파라메터들을 고려해야 한다. 또한, 이러한 기법을 기초로, PC의 CPU 자원 뿐만 아니라 유무선 인터넷 상에 존재하는 다양한 기기의 공유 및 협업 기술 또한 분산컴퓨팅 기술을 활용하여 연구해야할 부분이다.

참고문헌

[1] L.F.G. Sarmenta, "Bayanihan: Web-Based Volunteer Computing Using Java", http://www.cag.lcs.mit.edu/~bayanihan, 1998.

[2] M.Straber, J.Baumann, M.Schwehm, "An Agent-Based Framework for the Transparent Distribution of Computation", PDPTA, Vol.1, pp. 376-382, 1999.

[3] Carl A. Waldspurger et al., Spawn : A distributed computational economy, IEEE Transactions on Software Engineering, 18(2), February, 1992.

[4] Marc Jourdenais, Extending the process trellis software architecture to distributed environments, Yale CS690/691 Research Project Report, May, 1993.

[5] Nichoals J. Carriero, Eric Freeman, and David H. Gelernter, Adpative Parallelism and Piranha, Yale University Department of Computer Science, New Haven, Connecticut 06520, February 25, 1994.

[6] Nichoals J. Carriero, Eric Freeman, and David H. Gelernter, Adpative parallelism on multiprocessors : Preliminary experience with Piranha on the CM-5. In Utpal Banerjee, David Gelernter, Alex Nicolau, and David Padua(ed.), Languages and Compilers for Parallel Computing, in Lecrute Notes in Computer Science, Springer Verlag, Berlin, 768 : 139-151, 1994

[7] Nichoals J. Carriero, Eric Freeman, and David H. Gelernter, Adpative Parallelism and Piranha, Yale University Department of Computer Science, New Haven, Connecticut 06520, February 25, 1994.

[8] Koera@Home 프로젝트, "인터넷기반 분산컴퓨팅환경 구축사업 수행보고서", p. 408, 한국과학기술연구원, 2004.12.

[9] 한연희, 박찬열, 정영식, 황종선, "성능 기반 태스크 할당을 이용한 웹 기반 병렬처리 시스템의 설계", 정보과학회 논문지:시스템 및 이론, 제27권, 제3호, pp. 264~276. 2000.

저 자 소 개

김 진 일

2000년 한남대학교 컴퓨터공학과 졸업(박사)

1998년~2001년 배재대학교 IT센터 책임강사

2002년~2005년 (주)블루베리소프트 개발팀장

2004년~2005년 : 한국교육콘텐츠 연구개발센터팀장

2006년 현재, 배재대학교 교양교육지원센터 교수

〈관심분야〉 교육콘텐츠, 분산병렬처리, 퍼지이론