DOI: 10.3745/KIPSTA.2009.16-A.1.9

그리드 시스템에서 정적정보를 활용한 작업큐 중복 스케줄링 알고리즘

강 오 한[†] · 강 상 성[††] · 송 희 헌[†††]

요 약

그리드 시스템은 넓은 지역에 분산되어 있는 이질적인 자원들로 구성되어 있어서 가까운 지역에 비교적 동질적이고 통제가 가능한 자원들을 대상으로 하는 전통적 병렬시스템의 스케줄링 알고리즘으로는 효율적인 작업처리가 불가능하다. 본 논문에서는 그리드 시스템의 특성을 반영한 알고리즘을 제안하기 위해 기존의 스케줄링 알고리즘에서 사용하고 있는 정보의 종류에 초점을 두고 선행연구에서 제안된 알고리즘들을 비교 분석하여 개선할 수 있는 요소들을 도출하였다. 알고리즘들을 비교 분석한 결과 프로세서의 수나 성능과 같은 자원의 정적 정보가 스케줄링 알고리즘에 유용하게 사용될 수 있으며, 처리속도가 극단적으로 느리거나 사용이 불가능한 자원을 회피하기 위한 수단이 필요하고, 비교적 장시간 처리를 하는 그리드의 특성상 자원의 실시간 부하정보를 이용하는 경우 효용성이 떨어지는 것을 확인할 수 있었다. 본 논문에서는 이러한 분석 결과를 바탕으로 WQR(Workqueue Replication) 알고리즘의 논리에 정적 자원정보를 고려하도록 개선한 새로운 알고리즘(WQRuSI)을 제안하였으며, 시뮬레이션을 통하여 새로운 알고리즘의 성능이 우수함을 확인하였다.

키워드 : 그리드, 스케줄링, 정적 정보, 작업큐 중복

A Workqueue Replication Scheduling Algorithm Using Static Information on Grid Systems

Oh-Han Kang[†] · Sang-Sung Kang[††] · Hee-heon Song[†††]

ABSTRACT

Because Grid system consists of heterogenous computing resources, which are distributed on a wide scale, it is impossible to efficiently execute applications with scheduling algorithms of a conventional parallel system that, in contrast, aim at homogeneous and controllable resources. To suggest an algorithm that can fully reflect the characteristics of a grid system, our research is focused on examining the type of information used in current scheduling algorithms and consequently, deriving factors that could develop algorithms further. The results from the analysis of these algorithms not only show that static information of resources such as capacity or the number of processors can facilitate the scheduling algorithms but also verified a decrease in efficiency in case of utilizing real time load information of resources due to the intrinsic characteristics of a grid system relatively long computing time, and the need for the means to evade unfeasible resources or ones with slow processing time. In this paper, we propose a new algorithm, which is revised to reflect static information in the logic of WQR(Workqueue Replication) algorithms and show that it provides better performance than the one used in the existing method through simulation.

Keywords : Grid, Scheduling, Static Information, Workqueue Replication

1. 서 론

그리드(Grid) 시스템은 지역적으로 분산되어 있는 자원을 묶어 하나의 고성능 처리기 또는 대용량 저장소처럼 사용할 수 있도록 해주는 병렬·분산 시스템이다. 그리드를 고성능 처리기로 활용하는 계산그리드는 프로세서의 처리능력을 공유함으로써 복잡하고 많은 시간을 소요하는 작업들을 각 자원에 배분하여 보다 짧은 시간에 작업을 완료하도록 한다.

넓은 지역에 분산되어 있는 이질적인 자원들로부터 최적의 처리결과를 얻기 위해서 효율적이면서도 효과적인 스케줄링 알고리즘은 그리드 시스템의 중요한 기반이 된다. 병

※ 이 논문은 2008년도 정부(교육인적자원부)의 재원으로 한국대학교육협의회 대학교수 국내교류 연구비 지원에 의한 것임.
† 종신회원 : 안동대학교 컴퓨터교육과 교수
†† 정 회 원 : 안동대학교 교육공학과 박사과정
††† 정 회 원 : 안동대학교 컴퓨터교육과 교수
논문접수 : 2008년 8월 25일
수 정 일 : 1차 2008년 11월 11일, 2차 2008년 11월 25일
심사완료 : 2008년 11월 25일

렬·분산 컴퓨팅 환경에서 서로 독립적인 작업들을 최소의 시간에 완료하기 위한 스케줄링 알고리즘은 MCT, MET, SA, KPB 등의 온라인 알고리즘과 Min-Min, Max-Min, Sufferage 등의 배치(batch) 모드 알고리즘이 있다[1]. 이와 같은 알고리즘이 사용된 환경은 모든 자원이 단일관리영역에 위치하고 있으며, 스케줄러가 모든 자원을 제어할 수 있고, 자원의 수와 각 자원의 특성이 고정되어 있으며, 작업의 길이 및 자원의 성능을 쉽게 예측할 수 있는 환경이다[2]. 그러나 그리드 시스템을 구성하는 자원들은 보다 이질적이고 자율적이다. 또한 동적인 성능 특성을 가지고 있으며 자원의 선택과 연산에 필요한 데이터가가 분리되어 있다[3]. 따라서 비교적 동질적이고 전용성이 보장된 자원들로 구성된 시스템에 사용된 전통적 병렬·분산 시스템의 알고리즘들은 그리드 환경에서는 좋은 성능을 기대할 수 없다. 현재까지 그리드 시스템의 특성을 반영한 다양한 스케줄링 알고리즘[4-7]이 연구되어 왔지만 여전히 전통적 병렬·분산 환경의 알고리즘에 의존한 연구에 비하여 그 수가 적고 연구의 수준 역시 초기 단계에 머무르고 있다. 그리드 스케줄링에 관한 다양한 연구들이 진행되고 있으며, 특히 자원관리 시스템과 연계된 연구들이 최근에 국내외에서 발표되었다[12-17].

본 논문에서는 스케줄링에 있어서 전통적인 병렬·분산 환경과 그리드 환경의 주된 차이점이 스케줄링에 활용되는 정보의 종류와 질에 있는 것으로 판단하였다. 기존의 알고리즘들이 작업과 자원의 연결을 결정하는 판단근거로 사용하고 있는 정보 중, 실시간 CPU 부하와 같은 동적 정보는 그리드의 특성상 활용가치가 낮고 CPU 처리능력, CPU 개수 등 장시간 변하지 않는 정보는 활용가치가 높을 것으로 예상하였다. 이것을 확인하기 위해 그리드 환경을 위한 스케줄링 알고리즘에 대한 관련연구를 살펴보고 웹기반 그리드 스케줄링 시뮬레이션 도구인 WGridSP[8]를 사용하여 해당 알고리즘들의 성능을 분석하였다. 본 논문에서는 이러한 분석결과를 바탕으로 그리드 환경에서 스케줄링에 활용가치가 높은 정보를 포함하고 작업완료시간을 줄이는데 공헌하지 못하는 정보는 배제하여 새로운 알고리즘을 제안하였다.

2. 관련 연구

그리드 시스템을 위한 스케줄링 알고리즘은 자원과 작업의 특성, 스케줄링 시점, 지향하는 목표 등에 따라 여러 가지로 분류할 수 있다. 여기서는 계산그리드를 사용하여 상호 독립적인 작업들을 배치 모드에서 최소의 완료시간을 얻기 위한 알고리즘들을 비교하였다. 그리고 본 논문에서 관심을 둔 알고리즘이 활용하는 정보의 종류를 기준으로 분류하여 관련연구를 살펴보았다.

2.1 성능예측 정보를 사용하는 알고리즘

배정할 작업의 길이와 자원의 준비시간 및 처리능력에 대한 정보를 활용하는 알고리즘은 대부분 전통적 병렬·분산 환경에서부터 연구되어 왔다. 대표적인 알고리즘으로 Min-Min과 Max-Min이 있다. Min(Max)-Min 알고리즘은 배정되지 않은 작업들 중에서 최소(최대) 완료시간을 가진 작업을 선택하여 가장 빠른 완료시간이 기대되는 자원에 작업을 배정한다.

Min(Max)-Min 알고리즘은 간단하게 구현할 수 있어서 다른 상황에 쉽게 적용할 수 있다. He 등[9]은 그리드 컴퓨팅에서 QoS가 요구되는 작업들을 최단시간에 완료하기 위해 Min-Min 알고리즘을 수정하여 제안하였으며, Wu 등[10]은 정렬된 작업들을 세그먼트로 나누어서 Min-Min 알고리즘을 적용시켰다.

Min(Max)-Min과는 다르게 처음부터 그리드 시스템을 위해 설계된 알고리즘이 Buyya[4], Muthuvelu 등[5]에 의해 제안되었다. Buyya는 그리드시스템의 자원들이 요구하는 경제적 비용을 고려하여 지정한 예산 범위 내에서 최적의 자원조합을 찾기 위한 알고리즘 모델을 제안하였다. 그가 제안한 모델에는 비용최적화, 시간최적화, 비용-시간 최적화 알고리즘이 포함되어 있어서 그리드 사용자의 예산과 작업의 중요도에 따라 적당한 알고리즘을 선택할 수 있다. Muthuvelu 등은 수많은 작은 조각으로 구성된 작업들을 일정규모의 크기만큼 묶어서 자원에 배정하는 전략을 제시하였다.

2.2 성능예측 정보를 사용하지 않는 알고리즘

자원의 현재 부하를 고려하여 계산된 성능과 배정하려는 작업의 길이를 정확하게 예측하는 것은 로컬시스템에서도 쉽지 않은 문제이다. 특히 그리드 시스템의 특성상 자원의 부하(load)정보를 고려한 성능과 상태를 실시간으로 유지하고 작업을 완료하는데 소요될 시간을 예측하는 것은 더욱 어렵다. 비록 이것을 정확하게 예측할 수 있더라도 그리드 시스템을 이용하는 작업들은 비교적 긴 실행시간이 지속되기 때문에 시간이 경과하면서 자원의 부하가 변화하므로 예측치의 효력은 제한적이다. 이러한 이유로 성능예측정보를 사용하지 않는 그리드 스케줄링 알고리즘이 제안되었다[6, 7].

성능예측정보에 독립적인 알고리즘으로 가장 간단한 것이 Workqueue이다. Workqueue는 큐에 있는 작업들을 순서대로 모든 자원에 하나씩 배정하고 결과를 반환하는 자원에 한해 또 다른 작업을 즉시 배정한다. 결과적으로 느린 자원에는 적은 수의 작업이, 빠른 자원에는 많은 수의 작업이 배정되도록 하여 작업완료 시간을 최소화하려는 알고리즘이다. Subramani 등[7]은 특정 작업을 정해진 수만큼의 자원에 중복 배정하고 그 중 하나의 자원에서 작업을 실행하게 된다면 다른 자원에서 대기상태에 있는 작업의 배정을 취소

하는 방법을 사용하였다. 이 경우 작업대기 큐가 각각의 자원에 위치하여 자원의 활용률을 높이면서 전체 완료시간을 단축시킬 수 있다.

위에서 소개한 두 개의 알고리즘은 간단하게 구현할 수 있다. 그러나 지나치게 느린 자원 또는 여러 가지 이유에 의해 계속해서 실행이 불가능한 자원이 존재하는 경우에 그곳에 배정된 하나의 작업 때문에 전체 작업의 완료시간이 과도하게 길어지는 문제점을 안고 있다. 이 문제를 해결하기 위해 WQR(Workqueue Replication)[6] 알고리즘이 제안되었다. WQR 알고리즘은 기본적인 배정 방법이 Workqueue와 비슷하나 작업을 한 번만 배정하는 것이 아니라 모든 작업이 완료될 때까지 도착하지 않는 작업을 일정한 횟수만큼의 한계를 두고 중복적으로 배정한다. WQR 알고리즘은 특정 자원의 과도한 부하나 장애로부터 안정성을 확보할 수 있다. 그러나 같은 스케줄링 전략을 사용하는 사용자가 동시에 그리드를 사용할 경우 동일한 작업을 여러 자원에서 중복하여 실행하는데 따른 오버헤드가 발생할 수 있다.

3. 시뮬레이션 환경

3.1 시뮬레이션 도구

알고리즘의 성능분석 및 비교를 위하여 WGridSP[8]을 사용하였다. WGridSP[8]은 자바 기반의 시뮬레이션 도구인 GridSim[11]을 엔진으로 사용하여 웹상에서 알고리즘 개발 및 성능분석이 가능하도록 한 스케줄링 플랫폼이다.

본 논문에서는 자원의 상태와 정보에 따른 스케줄링 알고리즘의 성능을 분석하기 위해서는 시뮬레이션 도구를 통하여 작업뿐만 아니라 자원의 정적 및 동적 속성을 변화시킬 수 있도록 WGridSP의 성능분석 모듈의 기능을 보완하였다. 즉, 이전 WGridSP는 사전에 설계한 테스트베드를 선택하고 작업의 속성만을 정규분포 또는 단일분포에서 추출하거나 점진적 변화를 줄 수 있었다. 그러나 보강된 성능분석 모듈은 자원집합의 구성을 무작위로 추출할 수 있도록 하였다. 특히, 각 자원은 각 시간대별로 현재의 부하가 동적으로 변화하여 실제 그리드 환경과 유사한 환경에서 시뮬레이션이 가능하다.

그리드를 구성하는 자원의 정보는 프로세서의 수, 처리능력, 통신 속도 등의 정적 정보와 동적 정보인 현재의 부하(load)로 구성되어 있으며, WGridSP는 이러한 자원의 속성들을 지정한 분포에서 무작위 추출하여 그리드를 구성해준다.

자원의 부하는 사용자가 최대부하를 지정할 수 있도록 하였다. 최소부하는 0%로서 자원의 처리능력을 100% 활용할 수 있음을 의미하며 정해진 시간대에 따라서 <표 1>과 같이 시간대별 가중치가 변화하도록 되어 있다.

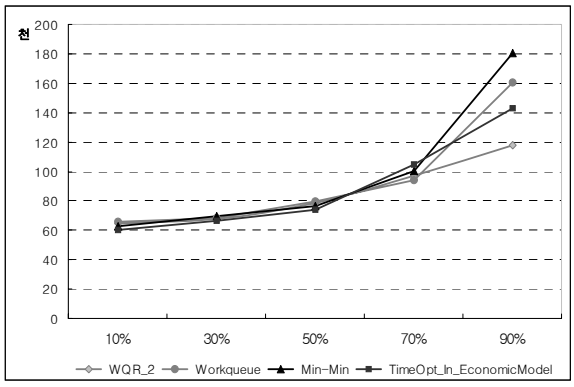
3.2 시뮬레이션 설정

시뮬레이션을 위해 그리드를 구성하는 자원의 개수는 50개로 지정하고 각 자원의 최소부하는 0%로 지정되어 있으며, 최대 부하는 10%, 30%, 50%, 70%, 90%로 변화를 주도록 하였다. 이와 같이 부하에 따라 알고리즘의 성능을 비교한 이유는 그리드 시스템에서 반환시간에 가장 많은 영향을 주는 요인이 자원의 부하인 것으로 판단하였기 때문이다. 자원의 부하가 과도하게 높은 경우는 자원이 로컬에서 사용되고 있거나 많은 작업이 몰려 있을 수 있으며 때로는 일시적으로 사용이 불가능 경우로도 받아들여질 수 있다. 각 자원의 부하가 최소 부하에서 최대부하까지 변화하도록 구성하고, <표 1>과 같이 최대 부하와 시간별 가중치를 곱하여 각 자원의 현재부하가 변하도록 하였다. 또한 모든 자원이 동시에 같은 부하를 가지지 않도록 하기 위해 자원의 GMT 기준 시간대를 [0, 23]의 단일분포에서 무작위 배정하였다.

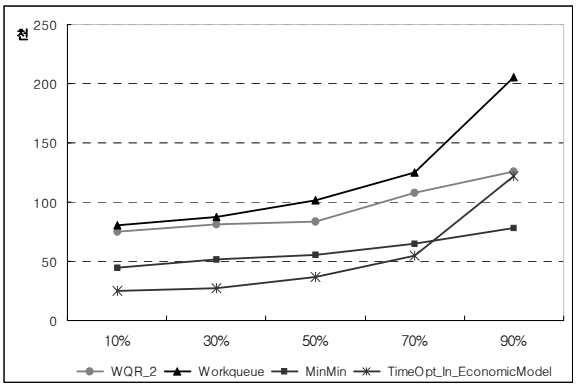
각 알고리즘에서 이용하는 정보에 대한 효용성을 중심으로 분석하기 위하여 두 가지 종류의 시뮬레이션을 수행하였다. 첫 번째는 자원의 정적 정보인 CPU 처리능력을 300으

〈표 1〉 최대 부하가 50%일 때 시간대별 자원 부하

시간	0~7	7~8	8~9	9~10	10~11	11~13	13~15	15~16	16~18	18~19	19~20	20~21	21~24
가중치	0	.1	.2	.4	.8	1	.6	.9	1	.5	.2	.1	0
부하(%)	0	5	10	20	40	50	30	45	50	25	10	5	0



(그림 1) 정적 자원정보를 고정한 시뮬레이션 결과



(그림 2) 정적 자원정보를 변화시킨 시뮬레이션 결과

로, CPU 개수를 1개로 고정하였다. CPU 처리능력을 나타내는 숫자는 1초당 처리할 수 있는 명령어의 개수를 의미한다. 예컨대 길이가 1200인 작업을 처리능력 300인 CPU 1개를 소유한 자원은 4초만에 처리할 수 있다. 두 번째는 CPU 처리능력과 개수를 무작위로 배정하여 각 자원이 서로 다른 CPU 성능과 개수를 가지도록 하였다. CPU 처리능력은 [100, 500]의 단일분포에서, CPU 개수는 [1, 8]의 단일분포에서 무작위 선택하도록 설정하였다.

처리할 응용을 구성하는 작업은 총 200개로 구성하였으며 작업의 길이는 [1000000, 5000000]의 단일분포에서 무작위 선택되도록 하였다. 결과적으로 부하가 0%인 자원에 배정된다고 가정하였을 때 하나의 작업 처리시간은 2000초에서 50000초 사이가 된다. 작업을 자원에 배정하고 결과를 반환받는데 소요되는 통신시간은 무시되도록 각 작업의 입출력 데이터를 모두 0으로 설정하였으며, 모든 시뮬레이션은 동일한 조건하에 10회 실시하여 산술평균값을 총 완료시간(Makespan)으로 사용하였다.

3.3 기존 알고리즘 분석

앞서 소개한 스케줄링 알고리즘 중 Min-Min 알고리즘, Buyya의 경제모델 중 시간최적화 알고리즘, Workqueue, WQR 등 네 가지 알고리즘을 비교분석 하였다. Min-Min 알고리즘은 전통적 병렬·분산 환경을 기반으로 한 알고리즘으로 작업의 길이와 자원의 정적 성능 그리고 실시간 부하정보를 필요로 한다. Buyya의 시간최적화 알고리즘은 작업의 길이와 자원의 정적 성능을 필요로 한다. Workqueue와 WQR은 가용자원 목록만을 필요로 한다. 알고리즘의 개선점을 찾기 위해 알고리즘에 활용되는 정보가 다양하게 포함되고 동일 조건에서 성능을 비교할 수 있는 알고리즘들을 선택하였다.

(그림 1)은 자원의 CPU 개수를 1개로, CPU 처리능력을 300으로 고정한 상태에서 네 가지 스케줄링 알고리즘의 성능을 나타낸 것이다. X축은 각 자원의 최대 사용률의 변화이며, Y축은 200개의 작업을 모두 완료하는데 소요된 시간이다. 네 개의 알고리즘은 자원의 최대 부하 70%이하에서는 거의 비슷한 성능을 보이다가 70% 이상에서 차이가 나타나기 시작한다. 자원의 최대부하가 100%에 근접할수록 소수의 극단적으로 느린 자원의 영향을 많이 받는 상황에 직면하게 되며 그러한 자원을 피해서 작업을 배정하는 알고리즘이 좋은 성능을 보이게 된다.

(그림 1)에서 가장 좋은 성능을 보인 알고리즘은 WQR로서 동일한 작업을 두 개의 자원에 중복 배정하여 극단적으로 부하가 집중된 자원에 배정된 작업을 기다리지 않아도 되기 때문인 것으로 판단된다. 특히 주목할 만한 것은 Min-Min 알고리즘으로서 자원의 동적 정보를 활용함에도 불구하고 가장 낮은 성능을 나타내고 있다. 전통적인 병렬·분산 컴퓨팅 환경은 하나의 작업이 몇 시간 이상씩 장시간 실행하는 경우가 거의 없기에 작업배정 시점에서 사용했던 동적 정보가 유용하다. 그러나 그리드 시스템에서는 배정 시점에서 사용한 동적정보는 긴 시간이 흐르면서 그 가치를 상실하게 됨으로써 극단적으로 부하가 증가하거나 장애가 발생한 자원을 피할 수 없게 된 것이다.

(그림 2)는 자원의 CPU 개수를 [1, 8]의 단일분포에서 무작위 선택하도록 하고, CPU 처리능력도 [100, 500]의 단일분포에서 무작위 선택하도록 한 상태에서 네 가지 스케줄링 알고리즘의 성능을 나타낸 것이다. 자원의 최대 부하가 70% 이상인 지점에서 Workqueue와 시간최적화 알고리즘의 완료시간이 길어지는 것은 앞선 시뮬레이션과 비슷하다. 주목할 점은 (그림 1)에서 최고의 성능을 보인 WQR 알고리즘과 가장 나쁜 성능을 보인 Min-Min 알고리즘의 순위가 반대로 나타난 것이다. WQR 알고리즘은 CPU의 수와 관계없이 모든 자원에 하나씩의 작업만을 배정하지만 Min-Min 알고리즘은 자원의 동적 정보를 활용하기 때문에 CPU의 수가 많으면 상대적으로 많은 작업을 배정한다. 그 결과 Min-Min 알고리즘이 WQR보다 전체 자원의 활용도를 높임으로서 더 빠른 시간 내에 전체작업을 완료할 수 있는 것으로 보인다.

시간최적화 알고리즘은 자원의 정적정보인 CPU의 개수와 처리능력, 그리고 작업의 길이를 활용함으로써 동적정보의 시간적 격차를 극복하지 못한 Min-Min 알고리즘 보다 자원 최대 부하 70% 이하에서는 더 좋은 성능을 보여준다. 그러나 극단적인 부하를 가진 자원을 피해가는 장치가 없으므로 70% 이상에서 급격하게 처리시간이 증가하고 있는 것을 볼 수 있다.

4. 새로운 알고리즘 제안

4.1 기존 알고리즘의 개선요소

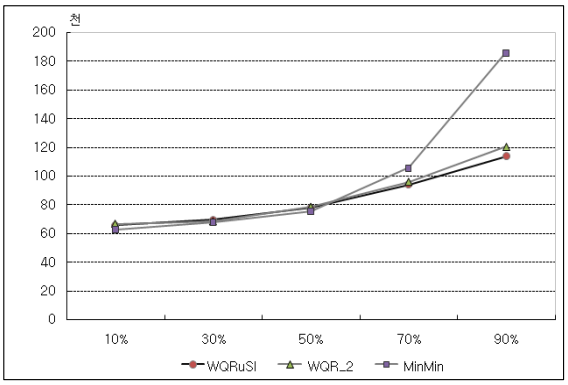
그리드 스케줄링에 활용하는 정보의 종류와 자원의 분포에 따른 시뮬레이션 결과를 정리하여 기존 알고리즘의 개선요소를 도출하였다.

(1) 극단적 부하 또는 불능 자원의 회피

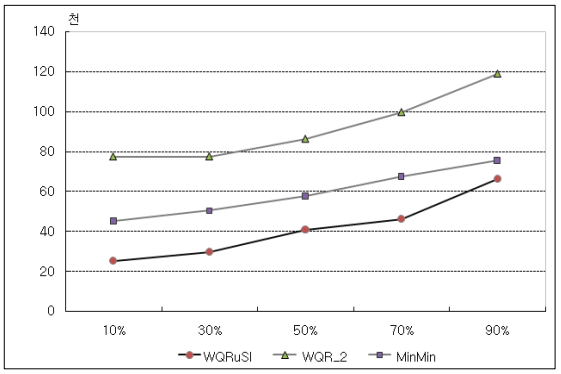
앞선 네 가지 알고리즘의 성능분석 결과를 살펴보면 자원의 최대 부하가 70% 이상인 경우에 작업완료시간의 증가

```
가용자원들을 프로세서의 성능에 따라 내림차순 정렬;
모든 작업(task)을 MaxReplication개씩 복제;
복제된 작업들을 TaskManager에 보관;
for i:=1 to ResourceList.size do
  for j:=1 to ResourceList[i].PEList.size do
  동일한 자원에 같은 작업이 배정되지 않는 조건하에
  TaskManager에서 하나의 작업을 꺼냄;
  꺼낸 작업을 ResourceList[i]에 배정;
  if TaskManager is empty break all for;
  end for
end for
while TaskManager is not empty do
  자원들로부터 완료된 작업의 반환을 기다림;
  if 완료작업과 똑같은 작업이 다른 자원에 존재 then
    해당 작업을 모두 취소시킴;
    동일한 자원에 같은 작업이 배정되지 않는 조건하에
    취소시킨 수만큼 TaskManager로부터 꺼냄;
    취소한 자원에 꺼낸 작업을 배정;
  end if
  동일한 자원에 같은 작업이 배정되지 않는 조건하에
  TaskManager에서 하나의 작업을 꺼냄;
  꺼낸 작업을 직전에 반환해준 자원에 배정;
end while
작업반환을 기다림;
```

(그림 3) 정적 자원정보를 사용한 WQRuSI 알고리즘



(그림 4) 제안된 알고리즘의 성능(정적 자원정보 고정)



(그림 5) 제안된 알고리즘의 성능(정적 자원정보 변화)

정도가 가장 적은 알고리즘이 WQR이다. 이는 WQR의 작업 중복배정 전략이 큰 영향을 준 것으로 볼 수 있다. 그리드 시스템을 위한 스케줄링 알고리즘은 극단적으로 낮은 성능을 보이거나 여러 가지 이유로 사용할 수 없는 자원을 회피할 수 있는 전략이 포함되어야 할 것이다.

(2) 동적 정보 활용의 배제

실시간 동적 정보를 활용하는 Min-Min 알고리즘은 정적정보를 모두 동일하게 적용한 자원집합에서 최악의 성능을 보였다. 이는 시간의 격차로 인해서 자원의 사용률이 변화될 때 적절히 대처할 수 없게 되어 실시간 자원정보인 동적정보가 그리드 시스템의 작업 스케줄링에 도움이 되지 못하는 것으로 해석될 수 있다. 또한 각 작업을 배정할 때마다 실시간 정보를 갱신하는 과정에 발생하는 오버헤드가 추가적으로 알고리즘의 성능을 저하시킨다. 따라서 지속적으로 동적정보를 감시하고 실행 중의 작업을 동적정보에 맞추어 취소하거나 이전할 수 있는 장치가 없다면 동적 정보의 활용은 지양해야 할 것이다.

(3) 정적 정보의 활용

동일한 정적 정보를 가진 자원들로 이루어진 그리드 시스템에서 좋은 성능을 보인 WQR 알고리즘이 다중 처리기가 포함된 시스템에서는 70% 이하의 최대 부하에서 Min-Min 알고리즘 보다 좋은 성능을 보이지 못하였다. 결국 정적 정보에 포함된 자원의 처리능력과 프로세서의 수를 감지하지 못하여 전체 자원을 효율적으로 활용하지 못하는 결과를 낳게 된다. 정적 자원정보는 시간이 흘러도 변할 가능성이 낮으므로 스케줄링 알고리즘에서 적극적으로 활용하는 것이 작업완료에 소요되는 시간을 줄이는데 도움이 될 것으로 판단된다.

4.2 새로운 알고리즘

본 논문에서는 앞에서 도출한 세 가지 개선요소(극단적으로 느리게 동작하거나 장애가 발생한 자원의 회피, 실시간 부하정보와 같은 동적 정보 배제, 장기간 변하지 않는 정적 정보 활용)를 포함하는 WQRuSI(Workqueue-Replication using Static Information) 알고리즘을 제안한다.

세 가지 요소를 포함하기 위해 우선 각 자원들을 기본적인 정적 정보인 CPU 처리능력 순으로 정렬하고, 작업을 배정할 때는 자원이 소유한 CPU의 수만큼 배정하도록 한다. 또한 기본적인 배정 전략은 동일한 작업을 두 개 이상의 작업에 중복 배정하여 어느 하나의 자원이 지나치게 느리거나 장애가 발생할 경우에 대비하는 WQR 기법을 활용하였다. WQRuSI 알고리즘의 상세한 흐름은 (그림 3)과 같다.

4.3 알고리즘 성능

제안된 알고리즘의 성능을 검증하기 위해 기존 알고리즘 분석에서 두 가지 상황에 각각 우수한 성능을 보인 WQR과 Min-Min 알고리즘과 함께 시뮬레이션을 수행하였다. 시뮬레이션 환경은 기존 분석 환경과 동일하게 설정하였으며, 그 결과는 (그림 4)와 (그림 5)에 나타나 있다.

정적 자원정보를 모두 일정하게 구성한 환경에서는 기존의 WQR과 비슷한 성능을 보였고, 정적 자원정보를 무작위로 선택한 환경에서는 WQRuSI 알고리즘이 가장 우수한 결과를 나타내고 있다. 정적 자원정보를 모두 동일하게 한 경우 정적 자원정보를 고려한 알고리즘의 효과가 드러나지 않은데 반해 자원의 프로세서 수와 처리능력을 다양하게 설정한 환경에서는 성능의 개선이 큰 것을 알 수 있다. 이것은 스케줄링 알고리즘에서 프로세서의 수나 성능과 같은 자원의 정적 정보를 유용하게 사용하였으며, 처리속도가 극단적으로 느리거나 사용이 불가능한 자원을 회피함으로써 나타난 결과이다.

5. 결 론

그리드 시스템은 기존의 병렬·분산 시스템에 비하여 보다 이질적이고 스케줄러의 정보접근성과 통제력이 낮으므로 스케줄링 알고리즘은 그리드 시스템의 특성을 반영할 수 있어야 한다. 알고리즘이 활용한 정보의 종류를 기준으로 선정한 Min-Min, Workqueue, WQR, 시간최적화 등의 알고리즘을 시뮬레이션한 결과 정적 자원정보를 동일하게 처리한 환경에서는 WQR이 가장 우수한 반면 Min-Min은 동적 정보를 활용함에도 가장 낮은 성능을 나타내었다. 반대로 프로세서의 수와 처리능력을 단일분포에서 무작위 선택한 환경에서는 WQR과 Min-Min의 성능 순위가 뒤집힌 결과가 나타났다.

시뮬레이션의 결과를 분석하여 극단적 부하가 걸린 자원 또는 장애가 있는 자원을 회피하는 전략이 필요하고 동적 자원정보는 알고리즘의 성능에 도움을 주지 못하며 정적 자원정보는 알고리즘에 반영해야한다는 세 가지 개선요소를 도출하였다.

본 논문에서는 도출한 개선요소를 반영하여 WQRuSI 알고리즘을 개발하였으며, 기존 알고리즘들과 성능을 비교하였다. 기존 알고리즘 분석에서 수행한 시뮬레이션과 동일한 환경에서 성능분석을 한 결과 정적 자원정보를 고정한 환경에서는 WQR과 비슷한 성능을 보였고, 정적 자원정보에 변화를 가한 환경에서는 상당한 수준의 성능개선이 이루어졌다. 실제 그리드 환경에서는 자원의 성능과 프로세서의 수가 다양하므로 WQRuSI 알고리즘은 실제 그리드 환경에서 작업처리시간의 단축에 기여를 할 수 있을 것이다.

본 논문에서는 장애 또는 극단적 저성능 자원을 회피하기 위해 작업을 2개 또는 그 이상의 자원에 중복하여 배정하는 방법을 이용했으나 동시에 여러 사용자가 같은 방법을 사용할 경우 오버헤드가 크게 발생할 위험성이 존재한다. 이를 위해 이상 자원을 회피하기 위한 방법이 더 연구되어져야 할 것이다. 또한 근본적으로 멀리 떨어져 있는 자원으로 구성되어 있는 그리드 시스템에서 데이터의 전송에 소요되는 시간을 무시할 수 없으므로 전송시간에 대한 요소도 알고리즘에 포함하고 실시간 부하정보에 따라 기 배정된 작업을 타 자원으로 이전하는 방법을 사용함으로써 정적 정보뿐만 아니라 실시간 부하정보를 활용하여 성능을 향상시키는 연구가 필요하다. 본 논문의 질을 향상하기 위하여 향후 중복의 가치를 효율성 대비 상대적으로 비교하고, 다양한 작업들의 저성능 자원접근에 대한 효율성 변화를 연구할 필요가 있다.

참 고 문 헌

[1] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, “Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems”, Journal of Parallel and Distributed Computing, Vol.59, pp.107~131, 1999.

[2] F. Berman, High-Performance Schedulers, chapter in the Grid: Blueprint for a New Computing Infrastructure, edited by I. Foster and C. Kesselman, Morgan Kaufmann Publishers, ISBN: 1-55860-475-8, pp.279~309, 1998.

[3] Y. Zhu, A Survey on Grid Scheduling Systems, Technical Report, Department of Computer Science, Hong Kong University of Science and Technology, 2003.

[4] R. Buyya, “Economic-based distributed Resource Management and Scheduling for Grid Computing”, Ph. D, Thesis, Monash University, Melbourne, Australia, 2002.

[5] N. Muthuvelu, J. Liu, N. L. Soe, S. Venugopal, A. Sulistio and R. Buyya, “A Dynamic Job Grouping-Based Scheduling for Deploying applications with Fine-Grained Tasks on Global Grids”, Australian Workshop on Grid Computing and e-Research(ausGrid2005), Newcastle, australia, Conferences and Practice in Information Technology, Vol.44, pp.41~48, 2005.

[6] D. P. Silva, W. Cirne and F. V. Brasileiro, “Trading Cycles for Informations: Using Replication to Scheduling Bag-of-Tasks Applications on Computational Grids”, in Proc of Euro-Par 2003, pp.169~180, Klagenfurt, Austria, August, 2003.

[7] V. subramani, r. Ketimuthu, S. Srinivasan and P. Sadayappan, “Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests”, in Proc. of 11th IEEE Symposium on High Performance Distributed Computing(HPDC 2002), pp.359~366, Edinburgh, Scotland, July, 2002.

[8] O. H. Kang, S. S. Kang, “Web-based Dynamic Scheduling Platform for Grid Computing”, IJCSNS International Journal of Computer Science and Network Security, Vol.6 No.5B, pp.67~75, May, 2006

[9] X. He, X. sun and G. Laszewski, “A Qos Guided Min-Min Heuristic for Grid Task Scheduling”, in J. of Computer Science and Technology, Special Issue on Grid Computing, Vol.18, No.4, pp.442~451, July, 2003.

[10] M. Wu, W. Shu and H. Zhang, “Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems”, in Proc. of the 9th heterogeneous Computing workshop(HCW'00), pp.375~385, Cancun, Mexico, May, 2000.

[11] R. Buyya, and M. Murshed, “GridSim: A toolkit for the modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing.”, The Journal of Concurrency and Computation: Practice and Experience(CCPE), Vol.14, pp.1175~1220, 2002.

[12] Zhifeng Yu and Weisong Shi, An Adaptive Rescheduling Strategy for Grid Workflow Applications, in Proc. of the 21st IPDPS, pp.1~8, 2007.

[13] Dong F, G. Akl S, “Scheduling algorithms for grid computing: state of the art and open problems.”, Technical Report, School of Computing, Queens's University, Kingston, Ontario, Jan., 2006.

[14] 현주호, 이승구, 김상철, 이민구, 데스크탑 그리드에서 자원 사용 경향성을 고려한 효율적인 스케줄링 기법, 정보과학회논문지: 시스템 및 이론, 33권 7호, pp.429~439, 2006.

[15] 김인기, 이종석, 자원 요구량과 가격 예측 기반의 그리드 자원 거래 모델, 정보과학회논문지: 컴퓨팅의 실제, 12권 5호, pp.275~285, 2006.

[16] 송은하, 정영식, 그리드 서비스 환경에서 효율적인 자원 관리 프레임워크, 정보과학회논문지: 시스템 및 이론, 35권 5호, pp.187~198, 2008.

[17] 이화민, 진성호, 이종혁, 이대원, 박성빈, 유헌창, 그리드에서 서비스 기반 가상 탐색 시스템 설계 및 구현, 정보과학회논문지: 시스템 및 이론, 35권 6호, pp.237~247, 2008.

강 오 한

e-mail : ohkang@andong.ac.kr

1982년 경북대학교 전자계열 전산모듈 (학사)

1984년 한국과학기술원 전산학과 (공학석사)

1992년 한국과학기술원 전산학과 (공학박사)

1984년~1994년 (주)큐닉스컴퓨터 선임/책임연구원

1994년~현 재 안동대학교 컴퓨터교육과 교수

관심분야 : 그리드컴퓨팅, 태스크스케줄링, OVPN 등

강 상 성

e-mail : edukang@andong.ac.kr

1998년 안동대학교 컴퓨터교육과(학사)

2001년 안동대학교 교육대학원 교육공학전공(이학석사)

2001년~현 재 안동대학교 교육공학과 박사과정

관심분야 : 그리드컴퓨팅, 스케줄링

송 희 헌

e-mail : hhsong@andong.ac.kr

1986년 동국대학교 컴퓨터공학과(학사)

1992년 충남대학교 컴퓨터과학과 (이학석사)

1995년 충북대학교 컴퓨터과학과 (이학박사)

1988년~1998년 한국전자통신연구원 선임연구원

1998년~현 재 안동대학교 컴퓨터교육과 부교수

관심분야 : 컴퓨터교육, 신경망, 차세대통신망 등