그리드 툴킷인 GridTool2를 사용한 스케줄링 알고리즘의 성능 평가

강오한†

요 약

본 논문에서는 그리드 시스템의 스케줄링 알고리즘을 시뮬레이션 할 수 있는 웹 기반의 스케줄링 툴킷(GridTool2)를 소개한다. 그리고 기존 MinMin과 Suffrage 스케줄링 알고리즘에 통신비용을 적용하여 성능이 향상된 수정 알고리즘을 제안한다. GridTool2는 서버와 데이터베이스를 기반으로 웹 환경에서 동작하므로 별도의 컴파일이나 실행 환경을 구축하지 않아도 된다. GridTool2는 통신비용과 함께 성능분석을 위한 변수들을 웹에서 입력하며, 시뮬레이션 결과를 웹페이지에 나타낸다. 통신비용을 적용한 수정된 알고리즘의 향상된 성능을 확인하기 위하여 GridTool2를 사용하여 실험하였다. 실험 결과에 따르면 기존 알고리즘보다 통신비용을 고려한 수정 알고리즘의 성능이 향상되었으며, 특히 작업량이 많아지면 성능향상의 폭이 증가하는 것으로 확인되었다.

주제어 : 그리드, 시뮬레이터, 스케줄링 알고리즘, 성능분석

Performance Evaluation of Scheduling Algorithms Using a Grid Toolkit(GridTool2)

Kang Oh-han†

ABSTRACT

In this paper, we introduce a web-based scheduling toolkit(GridTool2), which can run simulation of scheduling algorithm in grid system. And we suggest new algorithms which apply additional communication costs to the existing MinMin and Suffrage scheduling algorithms. Since GridTool2 runs in web environment using server and database, it does not require a separate compiler or runtime environment. The GridTool2 allows variables such as communication costs on the web for performance evaluation, and shows simulation results on the web page. The new algorithm with communication costs was tested using GridTool2 to check for performance improvements. The results revealed that the new algorithm showed better performance as more workloads were incorporated to the system.

Keywords : Grid, Simulator, Scheduling Algorithm, Communication Cost, Performance Evaluation

† 종신회원: 안동대학교 정보과학교육과 교수(교신저자)
논문접수: 2015년 4월 2일, 심사완료: 2015년 5월 7일, 게재확정: 2015년 5월 22일

1. 서론

그리드(Grid) 시스템을 위한 작업 스케줄링 알고리즘의 세부동작과 성능을 분석하기 위해서는 실제로 구축된 그리드 시스템에 알고리즘을 적용하여 로그를 분석하고 수행시간을 측정해야 한다. 그러나 그리드를 구성하는 자원들은 비교적 원거리에 위치해 있고, 서로 다른 여러 가지 특성을 가지며, 시간과 장소에 따라 자원을 통제하는데 어려움이 있다. 따라서 작업 스케줄링 알고리즘의 동작상태 및 성능을 분석하기 위해 그리드의 자원과 작업을 모델링하는 시뮬레이션 툴킷(toolkit)이 대안으로 사용되고 있다. 시뮬레이션 툴킷에서는 실제 시스템이 동작하는 것처럼 그리드 환경을 소프트웨어로 구현하여 결과를 분석한다. 툴킷에서는 그리드 환경이 가지는 다양한 특성들을 쉽게 반영할 수 있고, 결과의 확인 및 분석이 쉬운 장점이 있다.

그리드 환경의 스케줄링 시뮬레이션에서는 그리드 환경, 작업, 사용자 환경 등이 고려되어야 한다. 그리드 환경에서는 자원의 수, 각 자원의 프로세서 할당 전략, 각 자원의 통신망 전송 속도, 각 자원을 구성하는 노드의 수, 각 노드의 프로세서 수와 처리 능력 등이 고려되어야 한다. 작업과 관련된 내용으로는 처리할 작업의 수와 길이, 작업의 입출력 데이터 크기, 작업의 요구 프로세서 수, 각 작업의 선후 관계 등이 고려되어야 한다. 사용자와 관련된 내용으로는 사용자 수, 사용자의 통신망 속도 등이 고려되어야 한다.

그리드 환경에서 처리되는 응용프로그램은 연산량, 사용자 요구, 통신 유형, 입출력 비율 등의 측면에서 서로 다른 다양한 특성들을 가지고 있다. 이러한 상이한 특성을 갖는 응용프로그램은 서로 다른 스케줄링 알고리즘의 적용이 요구된다[1]. 넓은 지역에 분산되어 있는 이질적인 자원들로부터 최적의 처리결과를 얻기 위해서는 통신망의 특성을 고려한 효과적인 스케줄링 알고리즘이 필요하다. 그리드 환경에서 네트워크 대역폭은 노드간의 데이터 전송에 필요한 시간이므로 통신비용으로 계산될 수 있다.

본 논문에서는 그리드 컴퓨팅 환경에서 시스템을 모델링하고 스케줄링 알고리즘을 시뮬레이션 할 수 있는 웹 기반의 그리드 툴킷(GridTool2)을 소개한다. 그리고 기존 MinMin과 Suffrage 스케줄링 알고리즘에 통신비용을 적용하여 성능이 향상된 수정 알고리즘을 제안한다. 수정된 새로운 알고리즘의 향상된 성능을 확인하기 위하여 본 연구에서 개발한 GridTool2를 사용한다.

본 논문의 구성은 다음과 같다. 2장에서는 그리드 환경에서 사용할 수 있는 시뮬레이션 도구와 스케줄링 알고리즘에 관한 선행연구를 소개한다. 3장에서는 그리드 시뮬레이션 툴킷(GridTool2)의 구현에 관하여 기술한다. 그리고 4장에서는 구현된 GridTool2를 사용하여 스케줄링 알고리즘의 성능을 비교하고, 5장의 결론으로 끝을 맺는다.

2. 관련 연구

그리드 시스템의 성능향상을 위해서는 그리드 시스템과 응용프로그램의 특성을 반영한 다양한 스케줄링 알고리즘이 개발되어야 하며, 스케줄링 알고리즘의 성능을 효율적으로 비교하고 분석하기 위해서는 시뮬레이터가 필요하다. 현재까지 그리드 컴퓨팅 환경에서 스케줄링 알고리즘을 시뮬레이션 할 수 있는 다양한 도구들이 개발되었다[2][3][4][5][6]. 본 연구와 관련된 그리드 도구에는 Simgrid[2], GridSim[3][4], GridTool[6]이 있다.

기존의 그리드 스케줄링 도구를 활용하여 스케줄링 알고리즘을 시뮬레이션하기 위해서 적절한 개발환경의 구축, 소스코드의 분석, 자원과 작업 모델링을 위한 프로그래밍, 스케줄링 알고리즘 구현 등의 작업이 필요하다. 이러한 작업들은 연구자들에게 중복된 작업 과정을 유발시켜 스케줄링 알고리즘 연구의 효율성을 저하시킨다. GUI 기반의 모델링 도구인 Visual Modeler[4]는 GridSim을 위한 자원 모델링과 어플리케이션 모델링, 소스코드의 작성 등을 쉽게 할 수 있도록 환경을 지원한다. Visual Modeler는 모델링 과정을 쉽게 처리할 수 있도록 구현하였으나 자원 및 어플리케이션 정보의 저장과 관리, 시뮬레이션, 성능분석 등의 기능을 지원하지 않는다.

GridTool[6]은 그리드 시스템을 모델링하고 스케줄링 알고리즘의 시뮬레이션이 가능한 툴이다. 그러나 이것은 통신비용을 적용하지 않으며, 공개

버전이 없어서 외부에서 접근이 불가능하다. 본 논문에서는 GridTool의 이러한 단점을 개선하여 웹 기반의 스케줄링 툴킷인 GridTool2을 개발하였다. 웹에서 동작하는 GridTool2는 시뮬레이션을 위한 별도의 컴파일이나 실행 환경을 구축할 필요가 없다. 자원이나 응용프로그램의 모델링을 위해 별도의 텍스트 편집기를 사용하지 않고 웹브라우저를 도구로 사용한다. 또한 데이터베이스를 사용하며, 소스 코드를 쉘에서 컴파일하지 않고 웹브라우저에서 컴파일 기능을 지원한다. 스케줄링 알고리즘을 구현하기 위해 웹브라우저를 도구로 사용하며, 이를 통해 서버에서 컴파일이 이루어지도록 한다. 시뮬레이션과 성능분석 단계에서도 별도의 텍스트 편집기 없이 웹브라우저를 도구로 사용하며, 시뮬레이션 결과가 웹페이지로 나타낸다. 성능분석을 위한 변수를 웹에서 입력하여 실행하며, 결과를 웹페이지나 그래프로 나타낸다. 본 논문에서 소개한 GridTool2의 서버 주소는 'http://220.69.220.250/wgridsp'이며, 인터넷 접속을 통해 스케줄링 툴킷을 그리드 시스템 연구에 활용할 수 있다.

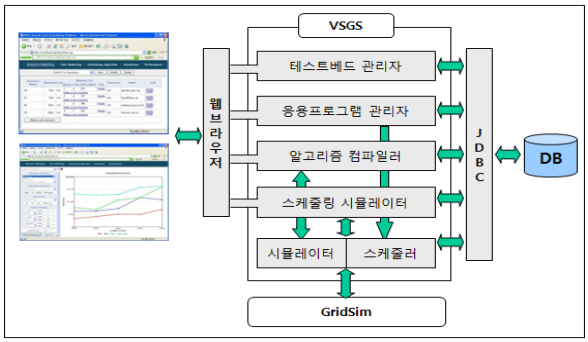
현재까지 그리드 환경에서 사용될 수 있는 다양한 형태의 스케줄링 알고리즘이 국내외에서 연구되었다[7-14]. 이들은 응용프로그램에 대한 비용과 시간의 최적화를 고려한 기법, 휴리스틱에 기반한 스케줄링 기법 등이 적용되었다.

그리드 환경에서 사용되는 스케줄링 알고리즘으로 MinMin[15][16][17]과 Suffrage[18][19][20] 알고리즘이 있으며, 기존에 발표된 다수의 논문에서 성능분석을 위해 이들 알고리즘을 사용하였다. 또한 이들을 수정하여 성능을 개선한 알고리즘들이 발표되었다. 본 논문에서는 MinMin과 Suffrage 스케줄링 알고리즘에 통신비용을 적용하여 성능이 향상된 수정 알고리즘을 제안한다. 이와 함께 기존 알고리즘과 수정된 알고리즘의 성능을 비교하기 위하여 GridTool2를 사용한다.

3. 그리드 툴킷(GridTool2)

본 논문에서 구현한 웹 기반의 그리드 툴킷인 GridTool2는 <그림 1>과 같이 테스트베드 관리자, 응용프로그램 관리자, 알고리즘 컴파일러, 스케줄링 시뮬레이션, 성능분석 모듈로 구성된다. GridTool2는 그리드에 적합한 스케줄링 알고리즘의 개발과 성능분석에 필요한 개발환경을 제공한다. 네트워크 대역폭을 고려한 통신비용이 적용된 스케줄링 알고리즘의 시뮬레이션이 가능하다.

GridTool2는 GridSim을 사용하여 실제 그리드와 유사한 실험 환경을 제공하며, GridBroker를 사용하여 다중사용자를 가정한 동적 스케줄링을 지원한다. GridSim은 가상적인 분산환경의 구현을 위해 자바 범용 라이브러리인 SimJava 패키지를 기반으로 개발되었다. GridTool2는 사용자와 GridSim 사이에서 웹을 매개로 한 인터페이스 역할을 하며, 자원과 응용프로그램 모델링 자료, 스케줄링 알고리즘을 데이터베이스에 관리하여 재사용이 가능하도록 한다.



<그림 1> GridTool2의 모듈 구조

3.1 GridTool2의 구성

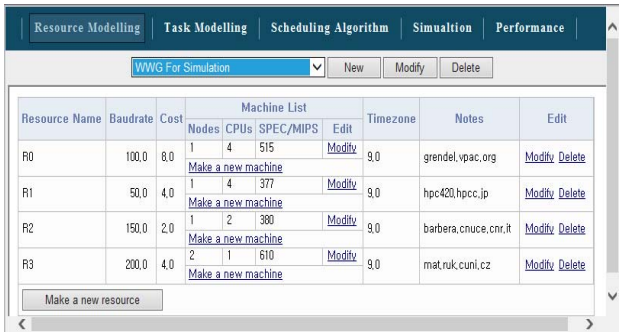
3.1.1 테스트베드 관리자 모듈

테스트베드 관리자는 사용자에게 자원 모델링을 쉽고 빠르게 할 수 있도록 지원한다. 자원 모델링은 실세계의 그리드 시스템을 가정하여 스케줄링 알고리즘을 시뮬레이션하기 위한 환경을 정의하는 것이다. 자원 모델링은 개별 테스트베드 단위로 저장, 수정, 삭제가 가능하다. 사용자가 모델링한 자원 정보는 테스트베드별로 관리되어 데이터베이스에 저장되고 시뮬레이션 및 성능분석에 사용된다. 테스트베드는 하나 이상의 자원으로, 각 자원은 하나 이상의 머신으로, 각 머신은 하나 이상의 프로세서로 구성될 수 있다.

GridTool2에서는 스케줄링 알고리즘을 시뮬레이션하기 전에 새로운 테스트베스와 작업을 정의

할 수 있다. GridTool2에서 자원에 지정하는 속성에는 전송 속도, 프로세서 처리 능력, 노드 수, 프로세서 수, 비용, 프로세서 할당 전략이 있다. 속성에서 비용은 해당 자원의 사용비용을 정의한 것이며, 비용을 고려한 스케줄링 알고리즘에 활용될 수 있다. 프로세서 할당 전략은 자원이 소유하고 있는 프로세서 수보다 처리를 기다리는 작업의 수가 많은 경우 프로세서를 어떻게 할당할 것인가를 결정한다. 프로세서 할당 전략은 특정 프로세서에 두 개 이상의 작업을 할당하여 시분할 방식으로 처리하는 TimeShared 방식, 먼저 할당받은 작업을 종료한 후 대기 중인 작업을 처리하는 SpaceShared 방식 중 하나를 선택할 수 있다.

<그림 2>는 자원 모델링의 결과를 나타낸 테스트베드의 예제이다. 자원 모델링에서는 새로운 테스트베드를 생성하거나 기존 테스트베드의 수정, 삭제가 가능하다. 이 테스트베드는 R0~R3까지 네 개의 자원으로 이루어져 있으며 서로 다른 특성을 가지고 있다. 예를 들면, 자원 R0는 통신 속도가 100, 초당 비용이 8이며, 성능이 515MIPS인 프로세서 4개를 장착한 시스템이다.



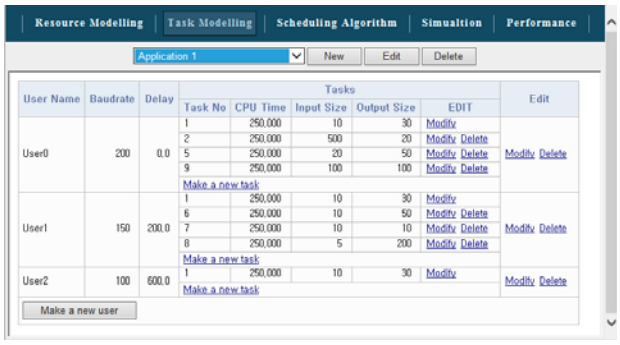
<그림 2> 자원모델링의 예

3.1.2 응용프로그램 관리자 모듈

응용프로그램 관리자는 각 응용프로그램을 구성하고 있는 작업들의 모델링을 지원한다. 모델링한 응용프로그램 정보는 데이터베이스에 저장되어 시뮬레이션에 사용된다. 각 응용프로그램은 하나 이상의 작업으로 구성되어 있으며 각 작업은 속성을 가지게 된다.

사용자에 속한 각 작업에 지정하는 속성에는 길이, 입력 데이터 크기, 출력 데이터 크기가 있다. 길이는 작업의 CPU 요구시간으로 이 값이 클수록 프로세서에서의 처리시간이 길어진다. 실제 처리시간은 자원의 처리속도로 나누어서 계산한다. 입력 데이터의 크기는 작업을 처리하기 위해 필요한 실행코드를 포함한 입력할 데이터의 크기를 말한다. 작업은 스케줄러에 의해 할당된 자원으로 전송해야 하므로 자원까지 전달하는데 소요되는 시간에 영향을 준다. 출력 데이터의 크기는 자원이 작업의 처리를 완료한 후 산출되는 데이터의 크기이며, 자원으로부터 스케줄러까지 결과를 전송하는데 소요되는 시간에 영향을 준다.

<그림 3>은 이름이 'Application 1'인 응용프로그램을 나타낸 것으로, User0~User2까지 3명의 사용자에 각각 4개, 4개, 1개씩의 작업이 배정되어 있다. User0은 통신속도 200에 연결되어 있으며, 지연시간이 0이므로 시뮬레이션 시작 직후 작업을 수행할 수 있다. User1~User2는 각각 통신속도 150, 100에 연결되어 있으며, 시뮬레이션을 시작한 후 200, 600 이후부터 스케줄링 알고리즘에 따라 각 작업에 자원을 배정한다.

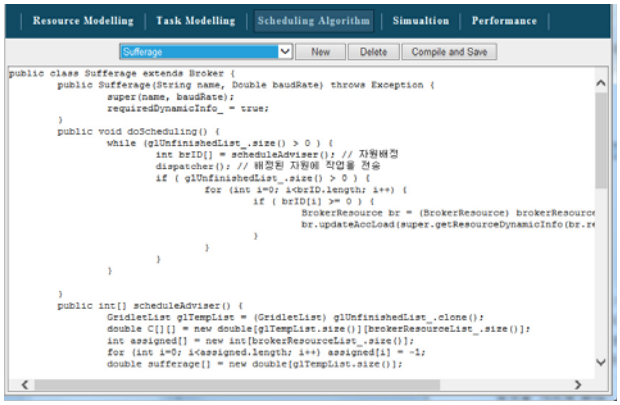


<그림 3> 응용프로그램 모델링의 예

3.1.3 스케줄링 알고리즘 작성 모듈

GridTool2에서 알고리즘의 소스 코드는 입력하면 자동으로 컴파일 되고, 컴파일 결과를 사용자에게 전달하여 오류를 수정할 수 있도록 한다. 알고리즘 개발자가 웹 환경에서 작성한 알고리즘은 컴파일이 이루어지며 알고리즘에 대한 문법오류는 즉시 피드백 되므로 수정이 가능하다. 컴파일이 성공적으로 이루어지면 시뮬레이션에서 사용가능한 스케줄러 클래스가 생성되고, 작성한 소스코드는 데이터베이스에 저장되어 필요시 수정하거나 다시 컴파일 할 수 있다. <그림 4>는 스케줄링 알고리즘 작성 모듈에서 Sufferage 알고리즘

의 소스코드 일부를 나타낸 것이다.



<그림 4> Sufferage 알고리즘의 소스 코드

3.1.4 스케줄링 알고리즘의 시뮬레이션 모듈

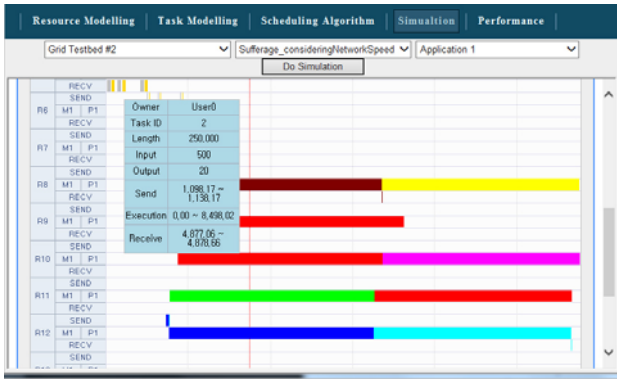
스케줄링 알고리즘의 시뮬레이션에 필요한 테스트베드, 응용프로그램, 알고리즘을 선택한 후 실행하면 시뮬레이션 결과를 웹브라우저에서 확인할 수 있다. 각 작업이 어떤 자원에 어느 시점에 전송, 실행, 반환되는지 사용자별로 확인함으로써 알고리즘의 세부동작을 분석할 수 있다.

<그림 5>은 'Grid Testbed #2' 테스트베드에 'Application 1' 응용프로그램과 Suffrage 알고리즘을 적용하여 시뮬레이션 한 결과이다. 시뮬레이션 결과는 모든 사용자들의 작업을 동시에 볼 수도 있으며, 특정 사용자에 속한 작업만을 볼 수도 있다. 사용자가 그리드의 자원정보를 요청하고 받는 것과 특정 작업을 자원에 전송하여 실행하고 결과를 반환받는 과정을 확인할 수 있다. 이러한 정보는 차트에 색깔로 구분하여 나타내며, 각 차트에 마우스 포인터를 올리면 작업의 자세한 수행 정보를 확인할 수 있다.

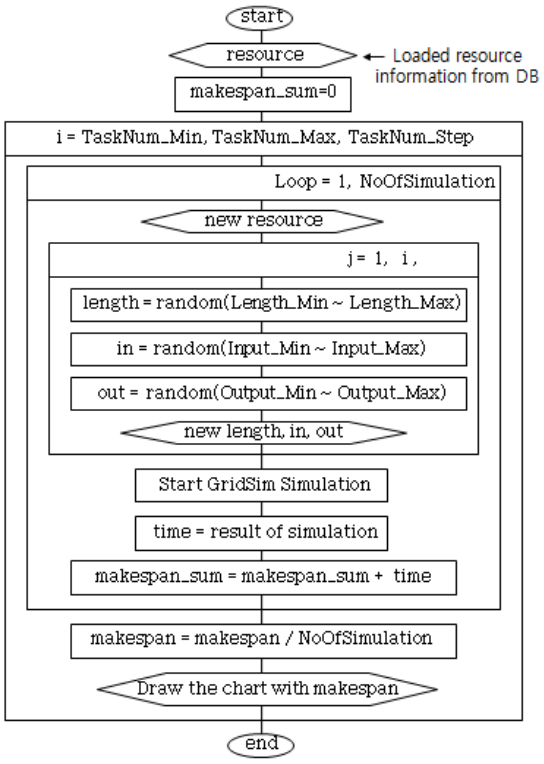
3.1.5 시뮬레이터 모듈

시뮬레이터는 GridTool2의 핵심 구성요소로서 자원정보, 사용자 및 작업 정보, 사용자가 입력한 변수를 이용한다. 시뮬레이터는 자원, 사용자, 작업 개체를 생성하여 GridSim이 시뮬레이션을 수행도록 하고 그 결과를 사용자에게 전달한다.

시뮬레이터 클래스는 GridTool2 내의 다른 클래스, GridBroker, GridSim 등과 상호작용 함으로써 사용자 인터페이스를 담당하는 GridTool2와 가상 그리드 환경을 제공하는 GridSim을 매개하는 역할을 한다. <그림 6>은 시뮬레이터의 동작을 순서도로 나타낸 것이다.



<그림 5> 시뮬레이션 결과 예시

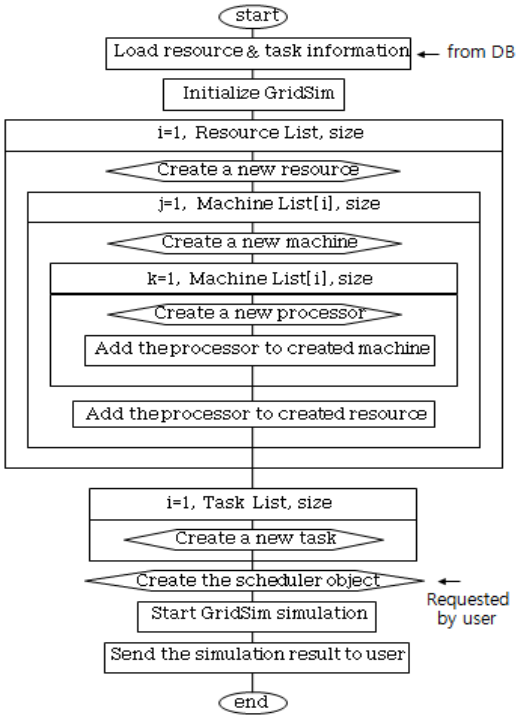


<그림 6> 시뮬레이터의 동작 순서도

3.1.6 성능분석 모듈

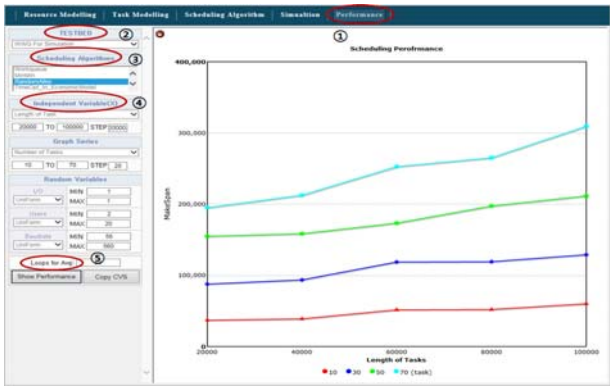
스케줄링 알고리즘의 성능분석은 하나 이상의 스케줄링 알고리즘을 선택하고 성능분석에 사용할 테스트베드, 작업의 속성을 위한 난수 범위, 독립변수 등을 선택하면 사용자의 평균 작업완료 시간의 변화를 차트로 확인할 수 있다. 각 작업의

수행길이, 입출력 크기 등은 지정한 범위 안에서 균일분포 또는 정규분포로 난수를 발생시키고, 정확한 측정값을 위해 지정한 반복횟수에 따른 평균값을 구해서 최종 수행시간을 산출한다. <그림 7>은 성능분석 모듈의 처리과정을 순서도로 나타낸 것이다.



<그림 7> 성능분석 모듈의 순서도

<그림 8>는 작업 길이에 따라 스케줄링 알고리즘의 성능을 비교하여 나타낸 예이다.



<그림 8> 성능 분석 모듈의 구성

스케줄링 알고리즘의 성능분석 모듈은 다양한 랜덤변수와 독립변수를 선택하여 변화를 관찰할 수 있는 기능을 제공한다. 알고리즘의 성능분석을 위한 변수에는 테스트베드, 스케줄링 알고리즘, X축 독립변수, Y축 독립변수, 난수가 있다. <그림 8>에서 ①~⑤까지의 설명은 다음과 같다.

① Performance: 알고리즘의 성능분석 기능을 선택하는 탭이다.
② 테스트베드: 어떤 테스트베드를 기반으로 성능분석을 할 것인지 선택한다. 테스트베드는 테스트베드 관리자에서 추가, 수정, 삭제가 가능하다.
③ 스케줄링 알고리즘: 성능분석을 위한 스케줄링 알고리즘을 선택한다. 두 개 이상의 알고리즘을 선택하면 선택한 스케줄러들의 성능을 비교할 수 있다.
④ X/Y축 독립변수, 난수: 그래프의 X축으로 변화될 변수를 선택하는 것으로 작업의 수, 사용자의 수, 작업의 길이, 입출력 파일의 크기 등이다. 선택한 변수의 범위를 기준으로 정규분포에 의한 난수를 사용하여 시뮬레이션을 실시한다. 스케줄링 알고리즘을 하나만 사용한 경우에 선택할 수 있다. 일반적으로 차트에서 범례로 구분하여 그래프가 그려진다. 독립변수의 선택이 완료되면 나머지 변수들의 범위를 지정한다. 이때 난수 분포를 산발분포와 정규분포 중 하나를 선택한다.
⑤ 재실행 횟수: 난수의 사용으로 데이터가 분산되는 것을 방지하기 위하여 재실행한 평균값이 나타나도록 되어 있다. 이때의 재실행 횟수를 지정하기 위한 입력 항목이다.

3.2 통신비용을 적용한 스케줄링 알고리즘

본 연구에서는 네트워크 대역폭 정보를 통신비용으로 점수(point)화하여 스케줄링 알고리즘에 반영하였으며, 점수를 구하는 방법을 수식으로 표현하면 아래와 같다. 수식의 결과는 네트워크를 통한 데이터의 전송시간과 비례한다.

$$Point = Weight \times \frac{Input\ Size + Output\ Size}{Baudrate}$$

수식에서 Weight는 점수의 가중치를 결정하는

상수를 나타낸다. 'Input Size'는 사용자가 그리드 시스템으로 전송해야 할 데이터의 크기를 의미하며, 'Output Size'는 그리드 시스템에서 작업이 처리된 후 사용자에게 전송할 데이터의 크기를 나타낸다. Baudrate는 사용자와 그리드 시스템 사이의 네트워크 대역폭을 의미한다.

수식에서 가중치가 작을 경우 기존 알고리즘과 성능 차이가 거의 없으며, 가중치를 지나치게 크게 설정하면 통신비용이 알고리즘 성능에 미치는 영향이 매우 커지게 된다. 이러한 경우 작업들이 네트워크 대역폭이 큰 자원에 집중적으로 할당되어 오히려 성능이 떨어질 수 있다.

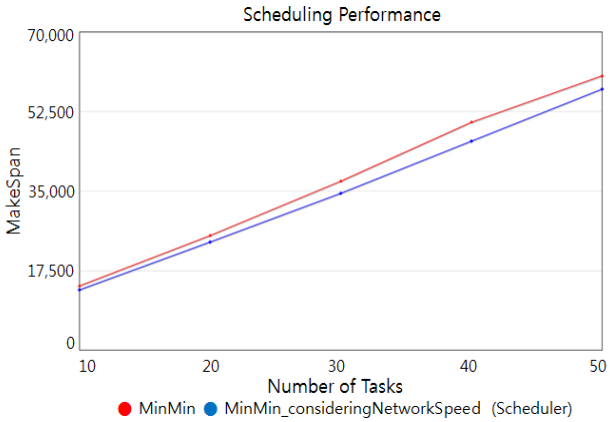
본 연구에서는 통신비용이 적용되지 않은 기존 알고리즘에 위의 수식을 적용하여 네트워크 대역폭이 고려된 개선된 알고리즘을 개발하였다. 실험을 통하여 대역폭 정보의 적용이 스케줄링 알고리즘의 성능 변화에 어떤 영향을 주었는지 분석한다.

4. 실험 및 분석

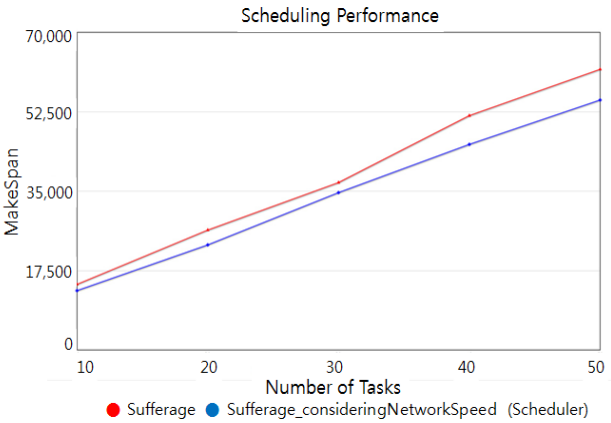
기존의 MinMin, Sufferage 알고리즘과 통신비용을 적용한 수정된 알고리즘을 GridTool2를 사용하여 성능을 비교한다. 실험을 위해 R1~R16의 16개 자원으로 구성된 'Grid Testbed #2' 테스트베드를 정의하였다. 각 자원의 전송 속도(Baudrate)를 6~100으로 설정하였으며, 각 자원의 처리속도는 1800MIPS로 모두 동일하게 설정하였다. 알고리즘의 성능을 평가하는 기준은 모델링한 그리드 환경에서 주어진 작업을 모두 종료하는데 소요되는 시간인 MakeSpan을 사용한다.

4.1 실험-1: 데이터의 크기가 작은 경우

테스트베드를 'Grid Testbed #2'로 선택한 후 모든 조건을 동일하게 설정하고 데이터 크기(Input Size, Output Size)를 32로 지정하여 시뮬레이션을 수행하였다. 동일한 조건에서 작업량의 변화에 따른 실험을 10번 수행한 후 평균값으로 두 알고리즘 성능 변화를 조사하였다. <그림 9>와 <그림 10>은 각각 MinMin 알고리즘과 Sufferage 알고리즘의 성능 변화는 나타낸 것이다. 두 알고리즘 모두에서 기존 알고리즘보다 통신비용을 적용한 수정 알고리즘의 성능이 향상된 것을 확인할 수 있다.



<그림 9> 데이터의 크기가 작은 경우 MinMin 알고리즘의 성능 변화

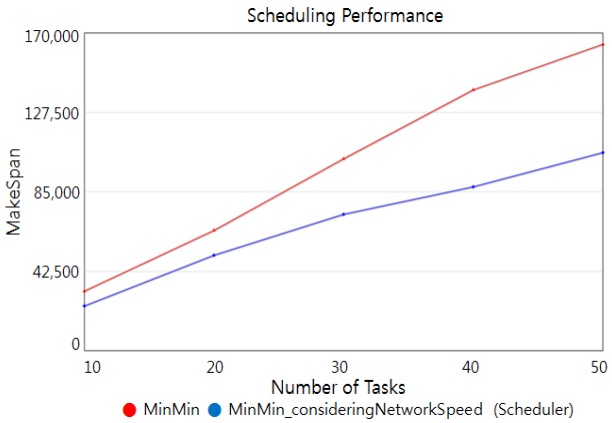


<그림 10> 데이터의 크기가 작은 경우 Sufferage 알고리즘의 성능 변화

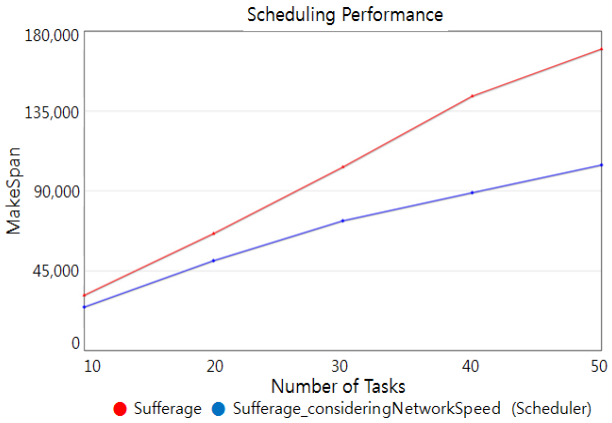
4.2 실험-2: 데이터의 크기가 큰 경우

실험-1과 동일한 조건에서 데이터 크기(Input Size, Output Size)를 768로 지정한 후 시뮬레이션을 수행하였다. <그림 11>과 <그림 12>는 각각 MinMin 알고리즘과 Sufferage 알고리즘의 성능 변화를 나타낸 것이다. 두 알고리즘 모두에서 통신비용을 고려한 알고리즘이 기존 알고리즘보다 성능이 향상된 것을 확인할 수 있다. 특히 실험-1과 실험-2의 결과에 따르면, 데이터의 크기가 32에서 768로 증가한 경우에 작업량이 많아지면 기존 알고리즘과 통신비용을 고려한 알고리즘 사

이의 성능 격차가 더욱 커지는 것을 확인할 수 있다. 즉, 통신비용을 적용한 새로운 알고리즘은 처리할 데이터의 크기가 클수록 성능이 우수한 것을 알 수 있다.



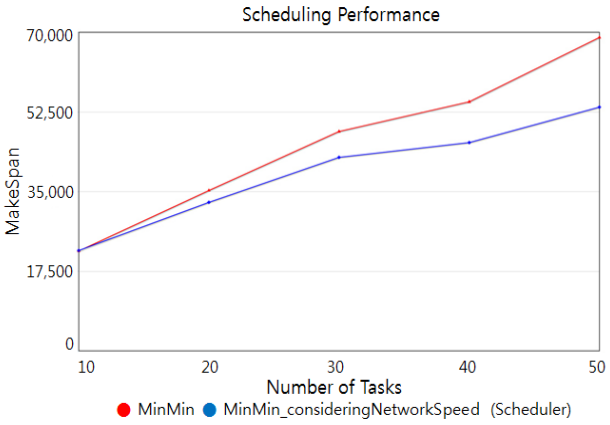
<그림 11> 데이터의 크기가 큰 경우 MinMin 알고리즘의 성능 변화



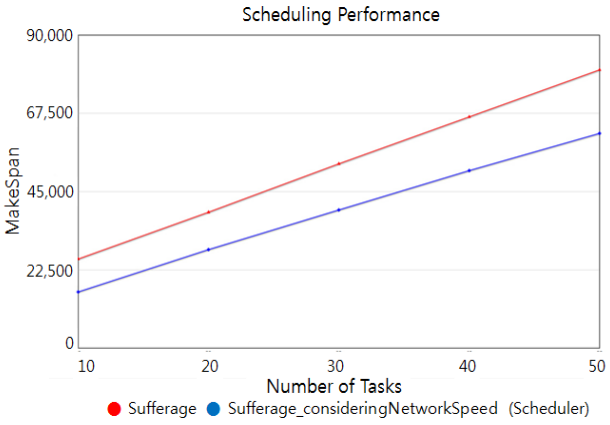
<그림 12> 데이터의 크기가 큰 경우 Sufferage 알고리즘의 성능 변화

4.3 실험-3: 임의의 데이터 크기인 경우

실제 그리드 시스템은 다양한 네트워크 대역폭을 가지고 있고, 서로 다른 크기의 작업들이 주어진다. 이러한 환경을 반영하여 이번 실험에서는 작업 길이(6000~30000)와 데이터의 크기를 임의의 크기(32~768)로 설정하고 실험을 수행하였다. 실험 결과는 <그림 13>과 <그림 14>와 같으며, 작업량이 많아지면 기존 알고리즘보다 수정된 알고리즘의 성능이 증가하는 것으로 나타났다. 특히 MinMin 알고리즘에서는 작업량이 증가하면 통신비용에 따른 성능의 차이가 더욱 커지는 것을 확인할 수 있다.



<그림 13> 임의의 데이터 크기인 경우 MinMin 알고리즘의 성능 변화



<그림 14> 임의의 데이터 크기인 경우 Sufferage 알고리즘의 성능 변화

본 연구에서는 기존 스케줄링 알고리즘에 네트워크 대역폭을 고려한 통신비용을 적용하여 알고리즘을 수정하였다. 그리고 GridTool2에서 시뮬레이션을 수행하여 기존 스케줄링 알고리즘보다 수정된 알고리즘의 성능이 더 우수함을 확인하였다. 하지만 각 자원마다 네트워크 대역폭의 차이가 크지 않거나 전송되는 데이터의 크기가 작을 경우 성능 향상이 미미한 것으로 나타났다. 향후에 이러한 한계점을 극복할 수 있는 방법과 효율성을 더 높이는 방법을 연구할 필요가 있다.

본 연구에서는 네트워크 대역폭 정보를 점수화하고 스케줄링 알고리즘에 반영하였다. 성능평가를 위한 실험에서는 자원 모델링에서 설정한 대역폭 정보를 수식에 반영하였으며, 가중치

(weight) 값을 1로 설정하였다. 향후 연구에서는 자원 모델, 알고리즘 특성, 작업 특성 등에 따른 가중치 값과 성능과의 관계를 분석할 필요가 있다.

5. 결론

본 논문에서는 통신비용을 포함한 그리드 시스템을 모델링하고 스케줄링 알고리즘의 성능을 분석할 수 있는 웹 기반의 그리드 툴킷(GridTool2)을 소개하였다. GridTool2는 자원이나 응용프로그램 모델링이 가능하고 알고리즘 구현을 위해 별도의 텍스트 편집기를 사용할 필요가 없다. 또한 스케줄링 알고리즘의 컴파일과 시뮬레이션이 서버에서 이루어지므로 쉘을 사용하지 않는다. GridTool2는 자바 환경의 스케줄링 도구인 GridSim을 시뮬레이션에 활용하였으며, 자원 모델링, 작업 모델링, 알고리즘 컴파일, 시뮬레이션, 성능분석을 효율적으로 수행할 수 있다.

본 논문에서는 기존의 MinMin과 Suffrage 알고리즘에 통신비용을 추가하여 성능이 향상된 새로운 알고리즘을 제안하였다. 수정된 알고리즘의 성능 변화를 검증하기 위해 그리드 스케줄링 툴킷인 GridTool2를 사용하여 실험하였다. 실험 결과에 따르면, 기존 알고리즘보다 통신비용을 적용한 새로운 알고리즘의 성능이 우수한 것으로 나타났다. 특히 전송되는 데이터의 크기가 커지고 작업량이 증가할수록 알고리즘의 성능 차이가 더욱 커지는 것으로 확인되었다.

참 고 문 헌

[1] I. Foster & C. Kesselman (2003). *The grid 2: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers.

[2] H. Casanova (2001). Simgrid: A toolkit for the simulation of application scheduling. *Proc. of the 1st IEEE/ACM Int. Symposium on Cluster Computing and the Grid*.

[3] R. Buyya & M. Murshed (2002). GridSim: a toolkit for the modelling and simulation of distributed resource management and scheduling for frid computing. *The Journal of Concurrency and Computation*, 14, 1175-1220.

[4] A. Sulistio & C. S. Yeo & R. Buyya (2003). Visual modeler for grid modeling and simulation (GridSim) toolkit. *ICCS LNCS 2659*, 1123-1132.

[5] H. B. Prajapati & V. A. Shah (2015). Analysis perspective views of grid simulation tools. *J. of Grid Computing,* 13, DOI 10.1007/s10723-015-9328-9.

[6] 강오한 · 강상성 · 송희헌 (2007). 그리드 시스템을 위한 웹 기반 스케줄링 툴킷의 구현. 한국컴퓨터교육학회 논문지, 10(3), 49-56.

[7] 오영은 · 김진석 (2005). 그리드 시스템을 위한 포인트 기반 스케줄링 알고리즘. 한국정보과학회 논문지 A, 32(12), 639-645.

[8] 강창훈 · 최창열 · 박기진 · 김성수 (2007). 그리드 컴퓨팅의 다중 큐 하이브리드 작업스케줄링 기법. 한국정보과학회 논문지 A, 34(7), 304-318.

[9] 조지훈 · 이원주 · 전창호 (2008). 계산 그리드를 위한 효율적인 작업 스케줄링 정책. 한국정보과학회 논문지 C, 14(8), 753-757.

[10] 강오한 · 강상성 · 김진석 (2007). Co-allocation 환경의 그리드 시스템에서 통신비용에 따른 스케줄링 알고리즘의 성능 분석. 한국정보처리학회 논문지 A, 14-A(2), 99-106.

[11] R. Sahu & A. K. Chaturvedi (2011). Many-objective comparison of twelve grid scheduling heuristics. *International Journal of Computer Applications,* 13(6), 9-17.

[12] J. Singh & G. Sharma (2014). A survey on QoS based task scheduling approach in grid computing. *IJETT*, 8(7), 359-366.

[13] K. Gupta & M. Singh (2012). Heuristic based task scheduling in grid. *International Journal of Engineering and*

Technology, 4(4), 254-260.

[14] G. Sharma & P. Banga (2013). Task aware switcher scheduling for batch mode mapping in computational grid environment. IJARCSSE, 3(6), 1292-1298.

[15] O. H. Ibarra & C. E. Kim (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM*, 24(2), 280-289.

[16] M. Wu & W. Shu & H. Zhang (2000). Segmented Min-Min: a static mapping algorithm for meta-tasks on Heterogeneous Computing System. *Proc. 9th Heterogeneous Workshop,* 375-385.

[17] X. He & X. Sun & G. Laszewski (2003). A QoS guided Min-Min heuristic for grid task scheduling. *Journal of Computer Science and Technology.* 18(4), 442-451.

[18] M. Maheswaran & S. Ali & H. J. Siegel & D. Hensgen & R. Freund (1999). Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. *8th IEEE Heterogeneous Computing Workshop.* 30-44,

[19] H. Casanova & A. Legrand & D. Zagorodnov & F. Berman (2000). Heuristics for scheduling parameter sweep applications in grid environments. *9th Heterogeneous Computing Workshop*, 349-363.

[20] E. U. Munir & J. Li & S. Shi (2007). QoS sufferage heuristic for independent task scheduling in grid. *Information Technology Journal*, 6(8), 1166-1170.

강 오 한

1982 경북대학교 전자계열
전산모듈(공학사)
1984 한국과학기술원
전산학과(공학석사)
1992 한국과학기술원 전산학과(공학박사)
1984～1994 (주)큐닉스컴퓨터 선임/책임연구원
1994～현재 안동대학교 정보과학교육과 교수
관심분야: 그리드컴퓨팅, 태스크스케줄링, 컴퓨터교육
E-Mail: ohkang@anu.ac.kr