과학 워크플로우를 위한 자원 가용 변화에 따른 데이터 재배치

김희원 O 김윤희 O*
O 숙명여자대학교 컴퓨터과학과
O{gmldnjs0610, yulan}@sookmyung.ac.kr

An Data Replacement Considering Resource Availability Change for Scientific Workflows

Heewon Kim O Yoonhee Kim O
O Dept. of Computer Science, Sookmyung Women's University

요 약

과학 워크플로우 응용의 데이터는 데이터 센터에 대량의 데이터가 분산되어 있고, 이를 이용하며 응용을 실행하기 때문에 저장되는 데이터 위치에 따라 실행 결과가 달라질 수 있다. 또한 실행 도중 생산된 중간 데이터의 위치도 전송에 영향을 미쳐 이를 최소화하기 위해 데이터의 위치가 중요하다. 이를 위해 데이터 집약적인 워크플로우 응용을 위해 동적으로 변하는 자원의 상태를 고려한 데이터 재배치를 제안한다. 이를 적용한 실험을 통해 데이터 이동을 줄일 수 있음을 보여준다.

1. 서 론

데이터 집약적인 과학 워크플로우 응용은 천문학[1], 생물 정보학[2]과 같은 분야에서 사용되며, 대용량의 입력 데이터를 분석하고, 수행 도중 중간데이터 생성 및 처리를 통해 대용량의 최종 데이터를 산출한다. 과학 워크플로우 응용의 데이터는 여러 데이터 센터에 대량의 데이터가 분산되어 있고, 생산되는 중간 데이터는 서로 다른 데이터 센터간에 접근을 통해 전송된다.
응용을 실행하면서 이를 이용하기 때문에 데이터 위치에 따라 실행 결과가 달라질 수 있다. 따라서 데이터들을 저장하는 위치에 대한 고려가 필요하며, 이를 위해 데이터배치가 중요하다.

최근 데이터 배치 기법 연구는 실험 초기 단계에서 데이터 간의 의존도[3], 데이터 크기[4], Bandwidth[5]에 집중되어 있으나 실험 중 변하는 자원의 상태에 대한 고려는 하지 않고 있다. 따라서 본 논문에서는 데이터 집약적인 워크플로우 응용을 통해 동적으로 변하는 자원의 상태를 고려한 데이터 재배치를 제안한다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에는 관련 연구를 살펴보고, 3장에서는 본 논문에서 제안하는 데이터 재배치에 대해 설명한 후, 4장에서는 제안한 데이터 재배치를 적용한 실험을 진행한다. 마지막 5장에서는 결론을 설명한다.

2. 관련 연구

데이터 집약적인 과학 응용에서는 데이터배치가 중요한 문제이다. Dong Yuan[3]는 의존도(dependency)에 따라 k-means 클러스터링을 통해 데이터를 배치한 전략을 제안하였다. Qing Zhao[4]는 데이터 크기를 고려하여 데이터 레이아웃을 위한 온라인 전략과 오프라인 전략을 제안하였다. Qiang Xu[6]는 유전 알고리즘(Genetic algorithm)을 통해 데이터를 배치하고 데이터센터 간의 데이터 스케줄링을 최소화하는 전략을 제안하였다.

위의 논문들의 연구에서는 데이터 집약적인 과학 응용에서 데이터 이동을 줄일 수 있지만, 실행 중간에 변하는 자원의 상태에 대한 고려는 하고 있지 않는다. 이에 본 논문에서는 실행 도중 동적으로 변하는 자원의 상태를 고려한 데이터 재배치를 제안한다.

3. 본 론

본 장에서는 데이터들 간에 의존도를 고려한 논문[3]을 기반으로 데이터들을 데이터 센터에 배치한 후, 실험 중 동적으로 변하는 자원의 상태를 고려한 데이터 재배치를 제안한다.

3.1 기본 전략

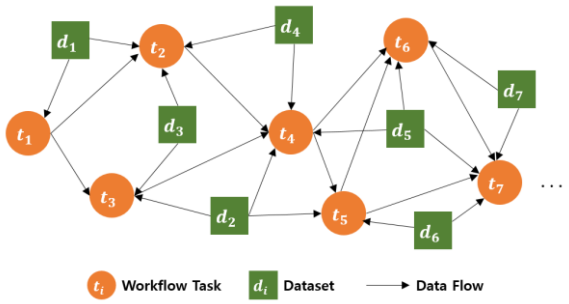
데이터 집약적인 과학 클라우드 워크플로우 응응의 경우 한 가지 태스크를 실행하기 위해 여러 데이터셋이 필요할 수도 있으며, 많은 태스크에서 하나의 데이터셋이 필요할 수도 있다. 이런 응용은 데이터 센터 간 데이터 이동에 많은 시간이 소요되고, 이때 소요되는

* 교신저자

시간을 줄이기 위해 데이터 센터에 데이터들을 랜덤하게 배치하는 것보다는 데이터들 간의 의존도를 고려한 알고리즘을 사용하여 배치한다[3].

본 논문에서는 동적으로 변하는 자원의 상태를 고려하여 작업 실행 중 다시 한번 데이터 재배치를 수행한다.

3.2 의존도에 따른 클러스터링

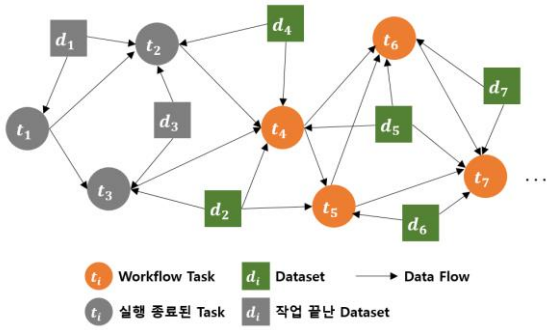


[그림 1] 워크플로우의 예

$$DM = \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ d_1 & 2 & 0 & 1 & 1 & 0 & 0 & 0 \\ d_2 & 0 & 3 & 1 & 1 & 1 & 1 & 0 \\ d_3 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ d_4 & 1 & 1 & 1 & 2 & 1 & 0 & 0 \\ d_5 & 0 & 1 & 0 & 1 & 3 & 1 & 1 \\ d_6 & 0 & 1 & 0 & 0 & 1 & 2 & 1 \\ d_7 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \end{array} \xrightarrow{BEA} CM = \begin{array}{c|ccccccc} & d_1 & d_3 & d_4 & d_2 & d_5 & d_6 & d_7 \\ d_1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ d_3 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ d_4 & 1 & 1 & 2 & 1 & 1 & 0 & 0 \\ d_2 & 0 & 1 & 1 & 2 & 1 & 1 & 0 \\ d_5 & 0 & 0 & 1 & 1 & 3 & 1 & 1 \\ d_6 & 0 & 0 & 0 & 1 & 1 & 2 & 1 \\ d_7 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \end{array}$$

[그림 2] 그림 1의 의존도 행렬과 BEA 변환 후의 행렬

그림 1은 데이터셋과 워크플로우 태스크 간의 워크플로우의 단순한 예를 보여준다[7]. 그림 1에 표시된 데이터 흐름을 통해 태스크 실행 시 다수의 데이터셋이 필요하며 한 개의 데이터셋은 다수의 태스크를 실행하기 위해 필요하다. 이런 워크플로우의 경우 의존도가 존재하며 이를 고려하여 데이터셋을 배치한다. 그림 2는 논문[3]의 알고리즘을 기반으로 그림 1의 데이터셋과 태스크 간의 의존도를 계산하여 DM(Dependency Matrix)를 생성한 후, BEA(Bond Energy Algorithm)[8]을 통해 DM을 CM(Clustered Dependency Matrix)으로 변환한다[7]. 다음으로 클러스터링된 CM 행렬을 통해 모든 데이터셋을 데이터 센터에 배치시킨다.



[그림 3] 실행 도중의 워크플로우의 예

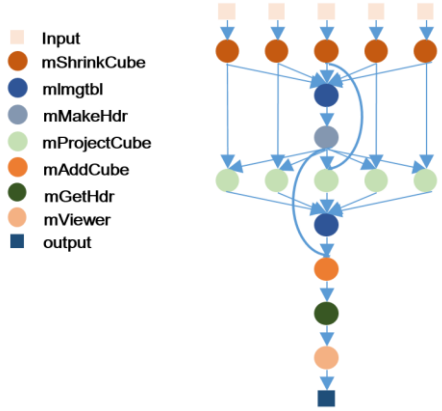
$$DM = \begin{array}{c|ccccc} & d_2 & d_4 & d_5 & d_6 & d_7 \\ d_2 & 2 & 1 & 1 & 1 & 0 \\ d_4 & 1 & 1 & 1 & 0 & 0 \\ d_5 & 1 & 1 & 3 & 1 & 2 \\ d_6 & 1 & 0 & 1 & 2 & 1 \\ d_7 & 0 & 0 & 2 & 1 & 2 \end{array} \xrightarrow{BEA} CM = \begin{array}{c|ccccc} & d_2 & d_6 & d_7 & d_5 & d_4 \\ d_2 & 2 & 1 & 0 & 1 & 1 \\ d_6 & 1 & 2 & 1 & 1 & 0 \\ d_7 & 0 & 1 & 2 & 2 & 0 \\ d_5 & 1 & 1 & 2 & 3 & 1 \\ d_4 & 1 & 0 & 0 & 1 & 1 \end{array}$$

[그림 4] 그림 3의 의존도 행렬과 BEA 변환 후의 행렬

본 논문에서는 어느 정도 작업이 실행 되었을 때, 자원의 상태가 변한다고 가정하였다. 그림 3은 작업 실행 도중의 변화된 데이터셋과 태스크들 간의 상태 변화를 보여준다[7]. 그림 4는 그림 3의 변화된 데이터셋과 태스크들 간의 상태에 따라 의존도를 다시 계산하여 DM 행렬을 만든 후 BEA를 사용하여 의존도에 따라 클러스터링된 CM 행렬을 보여준다. 그림 4에서는 변화된 자원의 상태를 고려하여 클러스터링된 결과에 따라 데이터셋 d_2와 d_6이 같은 데이터 센터에 배치되지만 그림 1에서는 d_2와 d_6이 같은 데이터 센터에 배치되지 않는다[7].

4. 실험 및 결과

본 장에서는 앞 장에서 언급한 동적으로 변하는 자원의 상태를 고려하여 데이터를 재배치 하였을 때, 데이터 재배치 전과 후의 데이터 센터간 이동하는 데이터 횟수와 크기를 통해 성능을 비교하였다.



[그림 5] Montage GALFA 워크플로우

본 논문에서는 워크플로우 특성을 가진 천문학 분야의 Mosaic 이미지 생성 엔진(Montage GALFA)[9]을 대상으로 실험하였다. Montage는 데이터 집약적인 특성을 가진 응용이며, 입력데이터를 사용하여 많은 수의 중간데이터와 하나의 출력데이터를 생성한다. 그림 5는 Montage GALFA 워크플로우를 보여준다[10]. 본 실험에서는 mShrinkCube 단계에서 15 planes를 사용하여 실험하였다.

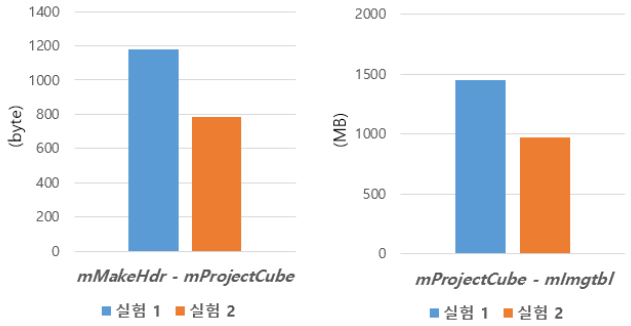
본 실험에서는 클라우드 관리 플랫폼인 OpenStack[11]을 사용하여, 하나의 컨트롤러 노드와

3개의 컴퓨팅 노드로 구성하였다.

실험은 mProjectCube 모듈에서 가용한 자원의 상태가 변하는 경우에 이때의 자원의 상태를 고려하여 작업을 재배치하였다. OpenStack의 5개의 자원(m1.medium)을 사용하여 작업을 재배치하기 전(실험 1)과 후(실험 2)의 데이터 센터간 이동하는 데이터의 횟수와 크기를 비교하는 실험을 진행하였다.

실험 1은 3개의 데이터 센터에 각각 2개의 자원을 생성하여 작업을 배치하였다. 실험 2는 mProjectCube 모듈에서 새로운 자원이 생겨 2개의 데이터센터에 작업을 재배치하였다. 그럼 5는 mProjectCube모듈에서 작업을 재배치한 상태를 보여준다. 데이터 센터간 데이터의 이동 횟수는 재배치하기 전과 후를 비교하였을 때, 6회에서 4회로 줄었음을 알 수 있다.

[그림 5] mProjectCube 모듈에서 재배치 전과 후



[그림 6] 재배치 전과 후의 이동 데이터 크기

그림 6은 작업을 재배치 하기 전과 후의 데이터 센터간의 데이터 이동 크기를 나타내었다. mMakeHdr 모듈에서 mProjectCube 모듈에서의 데이터 센터간 이동하는 데이터 크기는 실험 2가 실험 1에 비해 33.3%줄었음을 알 수 있다. mProjectCube 모듈에서 mImgtbl 모듈에서의 데이터 센터간 이동하는 데이터 크기는 실험 2가 실험 1에 비해 33.4%줄었음을 알 수 있다.

본 논문에서 동적으로 변하는 자원의 상태를 고려하여 작업을 재배치 하였을 때, 데이터 센터간 이동하는 데이터의 횟수와 크기가 줄었음을 알 수 있다.

5. 결론

본 논문에서는 데이터 집약적인 과학 응용을 위한 자원을 고려한 데이터 재배치를 제안했다. 데이터센터간 데이터 이동에 많은 시간이 소요되는 것을 방지하기 위해 작업 실행 도중 동적으로 변하는 자원의 상태를 고려하여 다시 데이터셋과 태스크들 간의 의존도를 계산하여 데이터센터에 재배치하였다. 실험 결과 작업 재배치를 통해 데이터센터간 이동하는 데이터들의 횟수과 크기가 줄었음을 확인할 수 있다.

Acknowledgement

이 논문은 2017년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2017R1A2B4005681)

참고문헌

[1] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny, Pegasus: Mapping scientific workflows onto the grid, in: European Across Grids Conference, pp. 11-20.(2004)

[2] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat, P. Li, Taverna: A tool for the composition and enactment of bioinformatics workflows, Bioinformatics 20 (2004) 3045 3054.

[3] D.Yaun, Y.Yang, X.Liu and J.Chen, A data placement strategy in scientific cloud workflows, J.Future Generation Computer Systems, 26, 8(2010)

[4] Q.Zhao, Xiong, C., Zhang, K., Yue, Y., & Yang, J. A Data Placement Algorithm for Data Intensive Applications in Cloud. International Journal of Grid and Distributed Computing, 9(2), 145-156 (2016)

[5] S. Agarwal et al., Volley: Automated Data Placement for Geo-Distributed Cloud Services, Proc. 7th USENIX Conf. Networked Sys. Design and Implementation (2010)

[6] Q. Xu, Z. Xu, and T. Wang, A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing, International Journal of Intelligence Science 5 (2015)

[7] 김희원, et al. 데이터 집약적인 과학 응용을 위한 적응형 데이터배치 기법. 한국정보과학회 학술발표논문집, 2017, 88-90.

[8] W.T. McCormick, P.J. Sehweitzer, T.W. White, Problem decomposition and data reorganization by a clustering technique, Operations Research, 20, pp. 993-1009 (1972)

[9] 천문학 분야의 Mosaic 이미지 생성 엔진(Montage GALFA), http://montage.ipac.caltech.edu/docs/cubemosaicstutorial.html

[10] Choi, Jieun, Theodora Adufu, and Yoonhee Kim. "Data-locality aware scientific workflow scheduling methods in HPC cloud environments." International Journal of Parallel Programming (2017): 1-14.

[11] OpenStack, https://www.openstack.org/