그리드상에서 에이전트 그룹 기반의 대규모 데이터 분석 환경 개발

김병상*, 윤찬현*
*한국과학기술원 정보통신공학과
e-mail:{bs.kim,chyoun}@kaist.ac.kr

Large-Scale Distributed Data Analysis based on Agent Group on the Grid Environment

Byungsang Kim*, Chan-Hyun Youn*
*Dept of Information and Communications Engineering, KAIST

요 약

본 논문에서는 그리드 환경에서 대규모의 분산 데이터 분석 환경을 제공하고자 한다. 기존의 접근방식으로서 자원 성능 기반의 사전 예약 기법인 Push 기법과 가용자원이 능동적으로 작업을 가져오는 Pull 방식을 조합하여 분산 효율과 실행 시간을 최소화 하는 기법을 제안하고자 한다.

1. 서론

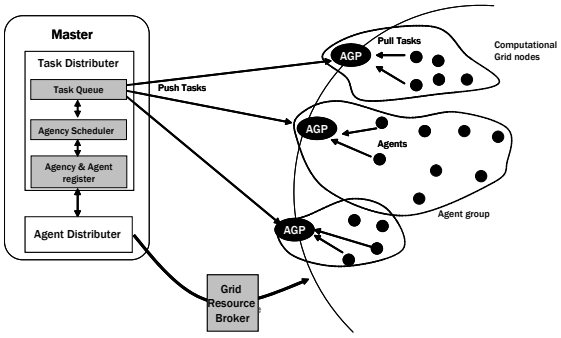
그리드와 같이 원격지간의 분산된 환경의 응용으로서는 분산된 작업들간의 통신 및 동기화를 필요로하는 병렬 작업보다는 대량의 데이터를 분할하여 독립적으로 분석하는 어플리케이션은 통신 비용을 최소화 할수 있기 때문에 많이 활용되는 어플리케이션이라 할 수 있다. 특히 생명정보분야의 단백질 접합구조 최적화 (Optimized Protein Folding) 혹은 단백질/후보물질의 결합구조 최적화 (Protein/Ligand Docking)과 같은 데이터 분석 모델이 그것이라 할 수 있다.

일반적으로 분산 환경에서 데이터 분석을 위해 가장 널리 활용되는것이 Master/Worker (M/W) 방식이라고 할 수 있다. 효과적인 M/W 스케줄링을 위하여 분할 부하 이론(Divisible Load Theory)[1] 은 분석할 데이터의 분할 비율을 가용한 자원의 수행성능과 자원까지의 데이터 전송 성능과 비례하도록 최적의 데이터 분할 비율을 정하여 데이터를 사전에 전송하는 분하 균등 기법(Load balancing)이다. 이와 같은 방법은 데이터 스케쥴링을 먼저 수행하기 때문에 분산효율을 최대화 할 수 있지만 자원의 수행 능력이나 데이터 전송 능력이 변하지 않는다는 가정을 하고 있기 때문에 그리드환경과 같은 원격지간의 통신이 필요한 경우 적합하지 않다. 또한 Worker의 수가 많아질수록 모든 비율을 최적화하는 것은 복잡도 면에서 부정확성을 증가시킬 수밖에 없다. 또 다른 방식으로 데이터의 분할 비율을 사전에 정하지 않고 Worker가 수행할 수 있는 시점에 Master에게 직접 데이터를 가지고와서 수행을 하는 기법이다[2][3]. 이와 같은 경우 부하균등(load balancing)의 최적화는 쉽게 성취할 수 있지만 Worker가 작업을 수행하기 전에 데이터의 전송을 기다리는 시간 비용을 줄일 수가 없어 분산효율이 떨어진다.

따라서 본 논문에서는 위의 두가지 방법을 병행하는 에이전트 그룹 기반의 루프 스케쥴링(AGLS) 기법을 통하여 분산효율을 최대화 하고 총 실행 시간을 최소화 하는 휴리스틱 기법을 제안하고자 한다.

2. Agent Group Peer based Loop Scheduling

AGLS에서는 Worker를 Agent로 표현하며 Agent 는 Master 와 직접 통신을 하기에 앞서 그 중간에 Agent Group Peer(AGP) 를 통하여 사전에 스케쥴링된 데이터를 확인한다. (그림 1)에서 보는것과 같이 Agent Group Peer 는 Master와 통신하여 수행할 데이터를 위치시킨다.



(그림 1) AGP 기반의 작업 스케줄링

따라서 Master와 모든 AGP간에는 분할 부하 이론을 기반으로 데이터 전송을 수행하는 기법을 통하여 사전 작업 계획을 하게 된다. 또한 AGP와 각각의 Agent간에는 Agent가 작업 수행 능력이 될 때 AGP에서 작업을 가지

고와서 수행하는 기법을 통하여 작업을 계획하게 된다.

(그림 2)는 AGP 기반의 Loop scheduling의 알고리즘을 나타내고 있다. 그림에서와 같이 Agent는 먼저 AGP와 통신을 통하여 Task를 획득하게 되며 만일 AGP에 실행할 Task가 존재하지 않으면 Agent 는 AGP의 크기, 즉 AGP에 포함된 Agent의 수만큼 일시에 많은 Task를 AGP에 저장하게 된다.

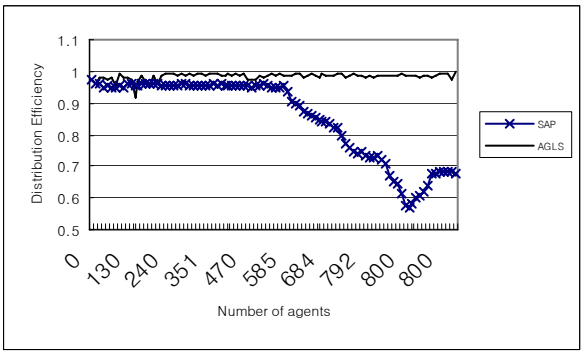
```
Algorithm. AGP based Loop Scheduling
BEGIN Agent
WHILE (AGENT_EXIT)
    Get task from Agent Group Peer (AGP)
    IF (Task(i) is empty)
        Calculate the size of AGP
        Get tasks which number of the size of AGP
      IF (tasks(j) is empty)
        AGENT_EXIT
        ELSE
            Stage-in the input of the task from Master to AGP
        END IF
    ELSE
        Stage-in the input of the task from AGP
        Local execution of application code for the task
      END IF
END WHILE
END
```

(그림 2) AGP 기반의 작업 스케줄링 알고리즘

따라서 AGP 는 Master 와 Agent 간의 임시 저장소 역할을 하게 되며 Group내의 Agent들은 직접 Master로 통신하는 부하를 줄일 수 있게 된다.

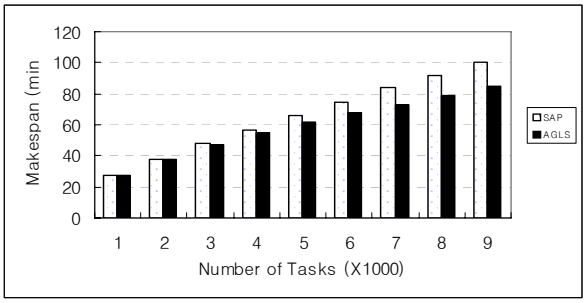
3. 성능 평가

제안한 기법의 성능 평가를 위하여 우리는 AGLS 를 포함하지 않는 SAP(Simple Agent Pull)기법과 AGLS 기반의 M/W 기법을 비교 평가하고자 한다. 우리는 총 100대의 그리드 노드 및 각 노드당 8개의 Agent 를 실행시켜 총 800개의 Agent 를 생성하였으며 사용자의 Task로서 총 10000개의 작업을 수행하면서 분산효율과 총 실행 시간을 비교하였다. 이때, 각각의 작업은 500Kbyte의 입력 데이터를 Master로서 전송받아서 수행되도록 하였다.



(그림 3) Comparison of Distribution Efficiency

(그림 3)은 제안한 AGLS 기법과 그렇지 않은 SAP기법과의 분산효율을 보여주고 있다. Agent 의 수가 500 이 넘어갈 때 SAP 는 분산 효율이 급격히 떨어지는것을 볼 수 있으나 AGLS의 분산효율은 Agent가 최대 800까지 증가하여도 효율이 떨어지지 않고 있다.



(그림 4) Comparison of Makespan

(그림 4)는 두 기법의 Makespan을 보여주고 있다. 5000개의 작업 이후 분산 효율이 <그림 3> 과 같이 차이가 나면서 AGLS 와 SAP의 총 실행 시간 또한 차이가 나고 있음을 알 수 있다. 실험에서 사용된 각각의 작업의 입력 데이터 크기는 고정되었지만 이것의 크기가 크면 클수록 두 기법의 성능은 더 크게 차이가 나게 될것임을 유추할 수 있다.

3. 결론

본 논문에서 우리는 AGP기반의 작업 스케쥴링을 통하여 기존의 단순한 M/W 방식의 분산 데이터 처리방식보다 20% 정도의 총 수행시간의 단축을 가져왔으며 데이터 크기에 관계없이 유연하게 Agent 수를 증가시키는것이 가능하므로 확장성 또한 매우 높다는것을 확인할 수 있다.

3. ACKNOWLEDGMENTS

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업(NIPA-2009-C1090-0902-002)의 연구 결과로 수행되었음

참고문헌

[1] V Bharadwaj et.al,Scheduling Divisible Loads In Parallel And Distributed Systems, IEEE Compuiter Society Press, 1996

[2] B.Kreaseck, L. Carter, H. Casanova, and J. Ferrante, "Autonomous protocols for bandwidth-centric scheduling of independent-task applications", in Proc. Int. parallel Distrib. process. Symp., Apr. 2003, pp.23-25

[3] V Garonne, A Tsaregorodtsev, E Caron, "A study of meta-scheduling architectures for high throughput computing: Pull versus Push" Parallel and Distributed Computing, 2005. ISPDC 2005. The 4th International Symposium on (2005)