분산 환경에서 노드 부하를 고려한 향상된 맵리듀스 스케쥴링 기법

An Enhanced MapReduce Scheduling Scheme Considering Node Load in Distributed Environments

황재민°, 오현교, 김천중, 임종태, 복경수, 유재수(교신저자)
충북대학교 정보통신공학과

Jaemin Hwang°, Hyunkyo Oh, Cheonjung Kim, Jongtae Lim, Kyoungsoo Bok, Jaesoo Yoo
Department of Information & Communication Engineering, Chungbuk National University

요약

데드라인을 고려한 스케쥴러는 데드라인 내에 잡을 완료시키기 위해 노드의 실시간 I/O 부하, 데이터 지역성 등을 이용한다. 하지만 데드라인을 만족시키기 위한 기법이 오히려 노드부하를 야기하는 현상이 나타난다. 본 논문에서는 노드의 부하에 따라 맵리듀스 처리 성능이 저하되는 문제점을 해결하기 위해 새로운 스케줄링 기법을 제안한다. 제안하는 스케줄링 기법은 우선 순위에 의해 중지되는 작업의 발생을 감소시키기 위해 동일한 작업을 중복적으로 수행시키는 사행 작업(speculative task)를 처리한다. 맵리듀스 작업의 지연이나 취소를 방지시키기 위해 핫 데이터 청크를 성능이 우수한 노드에 복제한다.

I. 서론

하둡의 HDFS는 클러스터 내에 데이터를 분산 저장한다. 이로 인해 지역성을 고려한 스케쥴링 기법이 필요하다. 즉, 지역성을 고려한 스케쥴링은 잡 수행에 필요한 데이터가 존재할 경우 해당 노드에 잡을 할당하는 것이다. 데이터 지역성을 고려한 대표적인 스케쥴링 기법에는 Delay Scheduler[1]와 LATE[2]가 제안되었다.

맵리듀스를 이용한 특정한 응용은 정해진 시간 내에 작업 수행을 완료하기 위해 데드라인 제약(deadline constraint)을 부여한다. 지역성을 고려한 스케쥴링 기법은 수행 성능을 향상시켰지만 데드라인을 부여한 작업을 수행할 경우 주어진 시간 내에 작업이 완료되지 못하는 문제가 발생한다. 잡 완료 시간에 영향을 미치는 요소는 데이터 지역성 외에 노드 I/O 부하 및 성능, 네트워크 부하 등 있다. 참고문헌 [3]에서는 노드의 I/O부하를 고려한 스케쥴링 기법을 제안 하였지만 상황에 따라 노드 부하가 집중되며 전체 잡 완료시간이 연장된다.

본 논문에서는 [3]의 맵리듀스 환경의 I/O 부하와 데드라인을 고려한 스케줄링 기법을 제안한다. 제안하는 기법은 데드라인이 시급한 잡들의 처리를 우선시 하며 발생하는 기존 할당 잡 작업들의 정지현상이 가져오는 부하를 해결한다. 사행 작업과 핫 데이터 복제를 이용한 스케쥴링은 정지현상이 반복됨으로 인해 해당 잡 완료시간이 증가하여 데드라인 실패율을 높이는 단점을 보완한다.

* 본 연구는 교육부와 한국연구재단의 지역혁신인력양성사업(No.2013H1B8A2032298), 미래창조과학부 및 정보통신산업진흥원의 대학IT연구센터육성 지원사업/IT융합고급인력과정지원사업(NIPA-2014-H0301-14-1022)과 2014년도 산업통상자원부 재원으로 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구과제입니다(NO. 20144030200450).

II. 제안하는 기법

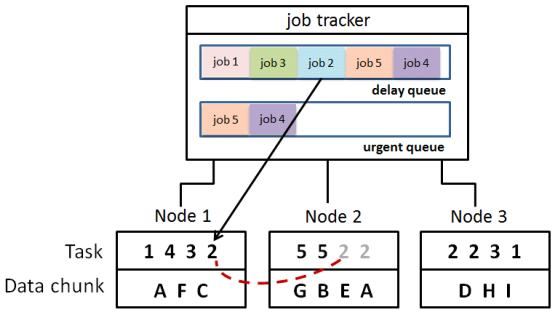
2.1 사행 작업(speculative task) 처리

처리가 가장 시급한 잡의 처리를 위해 기존 할당 된 작업들의 정지가 빈번해질 경우, 정지 되는 작업의 잡들 또한 잡 완료시간이 연장된다. 결국, 처리가 시급한 잡들이 증가하고, 데드라인 실패율이 증가한다. 이를 방지하기 위하여 본 논문에서 제안하는 기법은 사행 작업을 활용한다. 사행 작업은 같은 작업을 서로 다른 노드에 중복 할당시켜, 먼저 완료된 작업의 결과를 취하고 나머지 작업을 취소한다. 데드라인이 시급한 잡의 작업을 실행이 반복되는 노드에 이미 할당되어 있는 잡의 작업의 정지 상태가 N_{halt}번 이상 발생하면, 사행 작업을 실행시킨다. 만약 사용자가 N_{halt}을 2로 설정하였다면, 두번의 정지가 발생한 작업은 다른 복제본이 존재하는 다른 노드 중 I/O 부하가 가장 적은 노드에 사행 작업을 할당하게 된다.

그림 1은 정지가 빈번히 발생하는 작업의 스케쥴링이다. 표 1은 그림 1에서 제출된 잡들을 처리하기 위해 필요한 데이터 청크를 나타낸다. 처리가 가장 시급한 잡인 job 5의 작업을 빠르게 수행하기 위해 노드 2에 할당된 job 2의 작업이 정지되어 있는 상태이다. 사용자가 정한 N_{halt}값은 2로 가정하며 노드 2의 세번째 테스크가 이를 만족한 상태이다. 이 경우, 잡 2의 데드라인 실패를 방지하기 위하여 노드 2의 잡 2의 작업과 같은 사행 작업을 노드 1에 할당한다.

표 1. 잡 처리를 위한 데이터 청크

잡	필요한 데이터 청크
job 1	E, H
job 2	A, G
job 3	C, F
job 4	B, I
job 5	G, D



▶▶ 그림 3. 정지가 빈번히 발생하는 작업의 스케줄링

2.2 핫 데이터 청크 복제

사행 작업이 빈번해지면 노드에 I/O부하가 증가하게 되며 전체적인 잡 완료시간을 증가시키는 원인이 된다. 이를 방지하기 위하여 사행 작업을 할당할 경우, 노드의 I/O를 고려해야 한다. 스케줄러는 데이터 지역성을 가장 먼저 고려하기 때문에 제출된 모든 잡들이 공통적으로 필요한 데이터 청크를 가진 노드에 잡 할당을 시도할 확률이 높다. 사행 작업의 할당 또한 마찬가지이다.

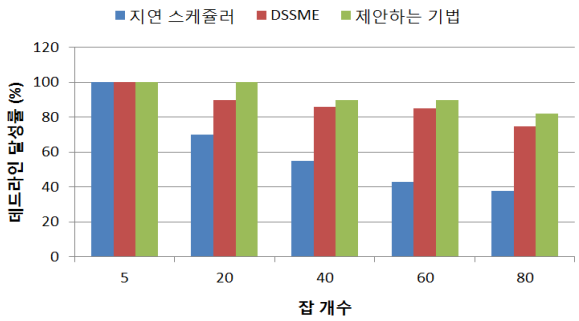
본 논문에서는 많은 핫 데이터 청크를 보유한 노드의 부하 집중을 성능이 좋은 노드에 핫 데이터 청크를 복제하여 대비한다. 제출된 잡들의 처리에 공통적으로 사용되는 데이터 청크를 '핫 데이터 청크'라고 명명한다. 사행 작업 할당 시 필요한 데이터 청크가 복제되어 있는 노드의 수가 많을수록 I/O부하가 가장 적은 노드를 선택할 수 있게 된다. 또한, 처리가 가장 시급한 잡을 처리 할 경우, 핫 데이터 청크가 많은 노드에 분산되어 있을수록 I/O부하가 적은 노드를 선택하여 작업을 수행할 수 있다.

성능이 뛰어난 노드의 판단을 위해 각 잡들의 평균 작업 완료시간 T_{job_i}을 측정한다. 각 노드에서 완료되는 작업들이 T_{job_i}의 1.1배를 초과하는 수를 Count한다. Count한 수를 기준으로 노드를 오름차순하여 N개의 노드를 선택하여 성능이 뛰어난 노드를 선택한다.

Ⅲ. 성능평가

성능평가는 Intel(R) Core(RM) i5 프로세서, 8G의 메모리를 가진 Window 7 운영 체제를 사용하는 PC에서 수행하였다. 실험은 5개부터 20개의 잡, 2개의 랙으로 구성된 9개의 노드를 가진 클러스터로 수행하였다. 각 잡은 30개의 작업으로 구성되어 있으며 각 데이터 청크는 임의의 노드, 같은 랙 내의 다른 노드, 랙 외부의 노드에 복제되어 데이터 청크별로 총 3개의 복제본이 분산 저장된다. 성능 평가는 제안하는 기법과 데이터 지역성만을 고려한 지연 스케줄러[1], DSSME[3]를 비교하였다.

그림 4는 잡 개수에 따른 데드라인 달성률을 나타낸 것이다. 잡 수에 따른 데드라이 부여된 잡수는 5개의 잡을 제외하고 50%를 할당하였다. 5개의 잡의 경우 데드라인이 부여된 잡수는 1개이다. 지연 스케줄링은 잡 개수가 늘어날수록 데드라인 달성률이 급격히 저하되는 반면 제안하는 기법에서는 82%이상의 달성률을 나타내었다. 80개의 잡을 기준으로 DSSME에 비해 7%가 향상되었다.



▶▶ 그림 4. 잡 개수에 따른 데드라인 달성률 (%)

Ⅳ. 결론

본 논문에서는 맵리듀스 환경의 I/O 부하와 데드라인을 고려한 새로운 스케줄링 기법을 제안하였다. 제안하는 기법은 I/O부하를 고려한 기존 스케줄링 연구의 문제점인 빈번한 작업 중단으로 인한 전체 잡 완료시간의 성능 저하를 제시하고, speculative task와 핫 데이터 청크의 복제 정책으로 해결하였다. 성능평가 결과 제안하는 기법이 기존 기법보다 우수한 성능을 나타내는 것을 확인하였다. 향후 연구로는 제안하는 데드라인 스케줄링를 실제 하둡 환경에서 적용할 예정이다.

▌참 고 문 헌▐

[1] Zaharia, M., Borthakur, D., Sarma, J. S., Elmeleegy, K., Shenker, S. and Stoica, I., "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,"Proc. EuroSys, pp.265-278, 2010.

[2] Zaharia, M., Kowinski, A., Joseph, A., Katz, R., and Stoica, I., "Improving MapReduce Performance in Heterogeneous Environments,"Proc. OSDI, pp.29-42, 2008.

[3] 황재민, 김천중, 오현교, 임종태, 리하, 복경수, 유재수, "맵리듀스 환경에서 데드라인을 고려한 잡 스케줄링 기법", 한국정보과학회 학술발표논문집, pp.65-67, 2014.