메모리기반 데이터 그리드 환경에서 확장성을 고려한 분산 캐시 구조 및 데이터 조달 기법

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

Distributed Cache Framework and its Data Procurement Algorithm on In-Memory Data Grid

Byungs-Sang Kim*, Chan-Hyun Youn**
*Dept of Information and Communication Engineering, KAIST
**Dept of Electronic & Electrical Engineering, KAIST

요 약

본 논문은 그리드 혹은 클라우드 컴퓨팅환경과 같은 인터넷 기반의 대규모 분산 환경에서 데이터집약적인 작업의 실행에 있어서 확장성을 위해 필수적으로 고려되는 데이터 전송 부하를 분산시키는 기법을 논하고 있다. 우리는 다수의 메모리기반의 데이터 노드를 활용하여 분할기법(Partitioning)을 기반으로 데이터 전송 부하를 줄이고자 하며 다수의 데이터 노드에 실시간으로 최적의 데이터의 양을 공급하는 기법에 대한 이론적인 분석과 시뮬레이션을 통한 성능 검증을 포함하고 있다.

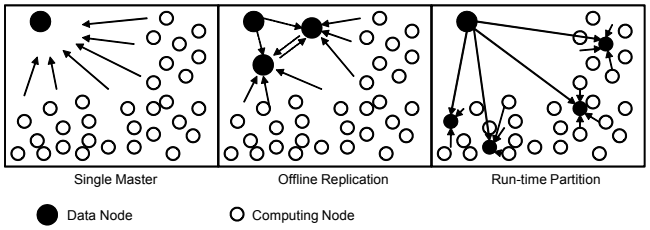
1. 서론

그리드 혹은 클라우드와 같은 인터넷 기반의 글로벌 컴퓨팅 환경이 성숙됨에 따라 인터넷 상에 흩어져 있는 수천 혹은 수만대의 컴퓨팅 자원을 동시에 활용하는 것이 가능해지고 있다. 하지만 컴퓨팅 자원이 규모가 커지더라도 선형적인 확장성을 보장하지 못하는 경우 예상되는 성능을 보장할수 없을뿐만 아니라 계산 자원의 낭비를 초래하게 된다. 특히 데이터 집약적인 작업의 경우 성능 저하는 대부분 데이터를 계산 노드로 이동할 때 걸리는 전송 부하에 의해서 나타난다. 일반적으로 이와같은 확장성의 저하를 막기위해서 데이터의 복제(Replication)를 통하여 성능 향상을 가져왔다[1]. 하지만 복제기법은 노드에 대용량의 데이터를 동일하게 저장하여야 하기 때문에 복제노드간의 동기화가 필수적이며 복제노드의 수가 증가할수록 저장 공간의 낭비와 동기화를 위한 부하가 증가하게 된다.

본 논문에서는 이와같은 대규모 데이터 분석 환경의 확장성을 향상시키기 위하여 소량의 데이터를 분할(Partition)하여 저장하는 기법을 적용하고자 한다. 우리는 데이터의 분할 저장을 위하여 메모리기반의 데이터그리드 환경[2]을 접합하고자 한다. 즉, 계산 자원중에 일부를 데이터 노드로 활용하여 디스크기반의 영구 저장이 아닌 메모리기반 캐시에 데이터를 저장하는 기법을 제안한다. 따라서 계산자원에서는 접근이 가장 효율적인 분산 데이터 노드에서 데이터를 가지고와서 단위 작업을 수행하게 된다. 이러한 메모리기반의 저장 구조는 접근성을 향상시킬 수 있지만 협소한 저장공간을 제공하기 때문에 동시에 저장할수 있는 데이터의 양이 제한될수밖에 없다. 따라서 본 논문에서는 분산 데이터 노드에 데이터의 사용량을 감시하여 실행시간중에 최적의 데이터의 양을 주기적으로 공급해주는 데이터 조달 알고리즘을 제안하여 안정적인 분산 데이터 캐시를 유지하는 기법을 논하고 있다.

2. 환경 및 모델 설명

대규모의 데이터를 다수의 독립적인 하위 작업으로 분할하여 병렬처리하는 기법은 일반적으로 단일 작업 분배기(Master) 와 계산 노드(Worker)에 의해서 수행되어왔다. 그림. 1(a) 에서 보는것과 같이 단일 작업 분배기가 모든 계산노드를 지원하는 구조는 작업 분배기의 병목현상에 의해 심한 성능저하를 가져온다. 그림. 1(b) 는 데이터 복제 기법을 사용한 예이다. 계산노드는 자신과 가까운 복제 노드에서 작업을 가지고 와서 수행한다. 하지만 계산노드와의 통신이 없기 때문에 작업분배기는 각 복제노드에서 수행된 작업의 정보를 다른 복제노드에 지속적으로 갱신하여야만 작업의 중복을 피할수 있다. 그림. 1(c)는 분할기법에 대해서 묘사하고 있다. 계산 노드중 선정된 데이터 노드는 자신의 메모리상에 데이터의 일부를 유지한다. 주변의 계산 노드에서는 이러한 데이터 노드에서 작업을 가지고 가서 수행하게 된다. 모든 데이터 노드간의 데이터는 중복되어 있지 않기 때문에 인위적인 동기화가 필요없다. 작업 분배기는 데이터 노드내의 데이터를 실행시간중에 주기적으로 보충하여 데이터 노드의 안정적인 운영을 유지시킨다. 이러한 분할 기법을 통한 데이터 캐싱기법은 각



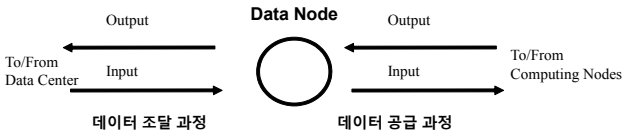
(그림 1) 확장성 지원을 위한 데이터 복제 및 분할 기법

데이터 노드내에 데이터를 안정적으로 유지시키는 것이 핵심이다. 계산노드에서 작업을 요청했을 경우 만약 데이터 노드에 데이터가 없을 경우 계산 노드는 작업을 수행하지 못하고 기다려야 하기 때문이다. 데이터 노드에 도착하는 계산 노드의 작업 요청의 수는 데이터 노드에 포함되어 있는 계산 노드의 수와 단위 작업당 수행시간의 길이에 따라 도착률이 결정된다. 다음 장에서는 데이터 노드의 데이터 요청 과정과 데이터 조달 기법에 대해 자세히 다루고자 한다.

3. 데이터 노드(Data Node) 구조 분석

3.1 데이터 요청 및 공급 과정

각각의 계산 노드는 데이터 노드에서 데이터를 가지고 와서 단위 작업을 수행한다. 작업의 실행이 완료되면 결과를 다시 데이터 노드에 저장하고 또 다른 데이터를 가지고 와서 작업을 수행한다. 그림 2 에서 보는것과 같이 대량의 계산 노드가 단일의 데이터 노드에 접근하여 동일한 데이터 전송 요청을 하게 된다. 따라서 데이터 공급과정에서 데이터 노드는 계산 노드들의 요청을 도착과정으로 하고 데이터 전송시간을 서비스 과정으로하는 대기행렬 시스템으로 간주 할 수 있다. 하나의 데이터 노드에서 데이터를 요청하는 요청률은 계산노드의 수와 단일 작업의 작업 시간에 의해 결정된다. 하나의 데이터 노드에 도착하는 하나의 계산 노드의 요청 간격은 단위 작업 수행 시간이 지수분포를 따르지 않는한 포아송과정을 따른다고 할수 없다. 하지만 데이터 노드에 접근하는 계산 노드의 수가 충분히 크다고 가정할 경우 데이터 노드에서 데이터의 요청과정은 Palm-Khintchine 이론[3]에 의해 이산확률 변수인 포아송 분포를 따른다고 할수 있다. 따라서 우리는 데이터노드의 데이터 요청 분포를 계산 노드의 개수와 작업 수행 시간에 의해 결정되는 포아송과정으로 정의하는것이 가능하다. 따라서 계산노드의 수와 단위 작업의 평균 수행 시간을 알지 못하더라도 데이터 노드에 도착하는 계산 노드의 요청 간격 및 요청률만으로 데이터 노드의 성능 분석이 가능하다.

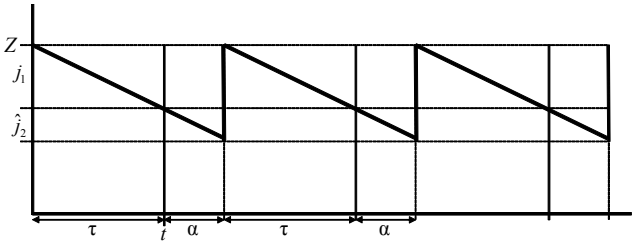


(그림 2) 데이터 조달 및 공급 과정 분석

3.2 안정 상태 데이터 조달 과정 분석

λ_d	계산 노드들에서의 데이터 요청율
π	데이터 센터로부터의 단위 데이터 전송 시간
τ	데이터 공급을 위한 검사 주기

안정상태에서의 데이터 조달 과정이란 데이터 노드내에 저장된 데이터의 양을 일정하게 유지하는 기법을 의미한다. 그림 2에서 보는것과 같이 데이터노드의 데이터는 데이터센터와 같은 단일 소스에서 제공되는 과정이다. 데이터센터에서는 주기적으로 데이터노드를 감시하여 데이터의 상태를 확인하고 필요한만큼의 데이터를 조달한다. 데이터 조달 과정은 위에서 명시한 3개의 파라미터에 의해서 결정된다.



(그림 3) 시간축에서의 데이터 보유량의 변화

그림 3 은 시간축상에서 조달 과정을 나타내고 있다. 시간 t 에서 보충해야할 데이터의 양은 주기 τ 동안의 소비량 $j_1(t)$과 $j_1(t)$을 보충하는데 걸리는 시간동안의 소비예측치인 $\hat{j_2}(t)$ 의 합으로 표현된다. 즉, 시간 t 에서의 데이터 보충량 $J(t)$는 다음과 같다.

$$J(t) = j_1(t) + \hat{j_2}(t). \quad (1)$$

$j_1(t)$의 값은 시간 t 에서의 실측값이며 이것을 이용하여 조달시간을 예측할수 있다. $j_1(t)$만큼의 데이터를 전송하기 위해 필요한 조달 시간 α_0 라고 하면 이것은 다음과 같이 정의된다.

$$\alpha_0 = \frac{j_1(t)}{\pi}. \quad (2)$$

추가하여 우리는 조달시간동안에 소모되는 데이터의 양을 고려해야한다. 이것은 위의 조달 시간의 재귀적 방법으로 표현된다. 즉, α_i를 조달기간 α_{i-1}동안에 소모될 데이터의 조달 시간의 예측치라 정의하면

$$\alpha_1 = \frac{\lambda_d \alpha_0}{\pi},\ \alpha_2 = \frac{\lambda_d \alpha_1}{\pi} \cdots. \quad 3)$$

과 같이 재귀적 표현이 되며, Eq(2)과 Eq(3)의 좌항과 우항을 더하고 $\alpha_0 + \alpha_1 + \cdots$ 을 α로 치환하면 α는 다음과 같이 주어진다.

$$\alpha = \frac{j_1(t)}{\pi - \lambda_d}. \quad (4)$$

따라서 데이터 노드에서 조달 시간동안의 소비예측치 $\hat{j_2}(t)$ 은 시간 α동안의 소비율이므로

$$\hat{j_2}(t) = \lambda_d \alpha = \frac{\lambda_d j_1(t)}{\pi - \lambda_d} \quad (5)$$

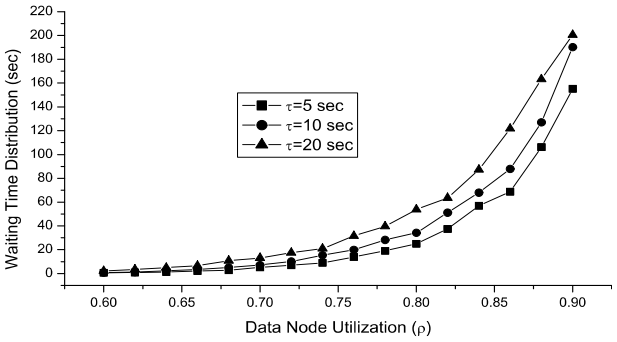
이며 결국 $J(t)$는 다음과 같이 구할수 있다.

$$\begin{aligned} J(t) &= j_1(t) + \hat{j_2}(t) \\ &= j_1(t) + \frac{\lambda_d j_1(t)}{\pi - \lambda_d} \\ &= \frac{\pi j_1}{\pi - \lambda_d} \end{aligned} \quad (6)$$

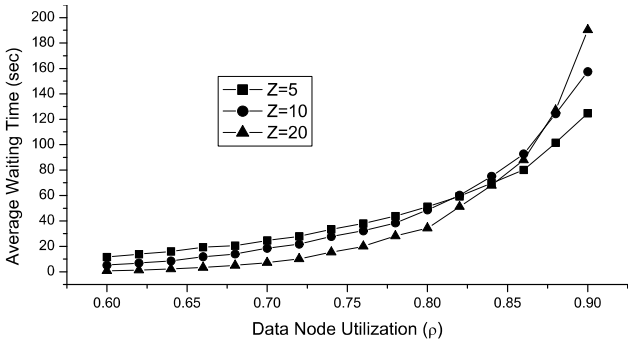
여기서 우리가 추가적으로 고려할것은 그림 3에서의 유지하고자 하는 데이터의 보유량(Z)과 검사주기 (τ)에 대한 고찰이다. 비록 추정값에 의해 데이터를 안정적으로 조달한다고 하더라도 데이터의 요청은 불확실성을 내재한 확률적 분포로 나타나기 때문에 주어진 데이터 요청률에서 보유량과 검사주기에 따른 대기시간의 분포를 분석하고자 한다.

4. 성능 평가

계산노드가 데이터 노드에서 데이터 획득하는 과정은 앞에서 설명한 대기행렬과정으로 모델링할수 있다. 따라서 우리는 대기시간의 성능 평가를 위하여 시뮬레이션 기법을 이용한다. 시뮬레이션 툴은 SimJava[4]를 활용하였다. 우리는 데이터 요청 간격과 데이터 전송 과정은 지수분포를 따르는 랜덤과정으로 가정한다. 이것은 대규모 그리드 환경의 이질성(heterogeneity)를 반영한다. 또한 데이터노드의 이용률(Utilization) $\rho = \lambda_d / \pi$ 로 정의한다. 그림 4 는 최대 데이터 보유량 (Z)를 20으로 고정하고 이용률(ρ)과 검사주기(τ)의 변화에 따른 계산노드의 데이터획득 대기시간의 분포를 나타내고 있다. 그림에서 보는것과 같이 동일 검사주기에서 이용률이 증가할수록 대기시간은 증가한다. 또한 검사주기가 길수록 대기시간의 분포가 높이 위치하는것을 볼수 있다. 따라서 대기시간은 검사주기가 작으면 작을수록 비례하여 작아진다는것을 알수 있다. 그림 5 는 검사시간(τ)을 10 sec로 고정하고 이용률(ρ)과 최대보유량(Z)의 변화에 따른 계산노드의 데이터획득 대기시간의 분포를 나타내고 있다. 동일한 최대보유량일 경우 이용률이 올라갈수록 대기시간은 함께 증가하는것을 볼수 있다. 하지만 최대보유량이 클수록 반드시 대기시간이 작게 나타나지 않는다. 특정 이용률 이상에서는 오히려 최대보유량이 높을수록 대기시간이 높아지는것을 볼수 있다. 이것은 이용률이 일정 수위 이상 올라가면 최대보유량을 채우기위한 소비예측치가 높아지고 이것은 결국 감시주기의 길이를 지수적으로 높이기 때문에 나타나는 현상이다.



(그림 4) 검사주기 및 이용률변화에 따른 대기시간 분포



(그림 5) 최대보유량 및 이용률변화에 따른 대기시간 분포

5. 결론

본 논문에서는 메모리기반의 데이터그리드환경을 활용하여 분할기법을 통한 데이터집약적 작업의 전송 부하를 감소시키는 방법을 논하고 있다. 특히 분산된 메모리기반의 캐시를 활용한 분할 기법과 실시간 데이터 공급 기법에 대한 성능 분석을 포함하고 있다. 향후 우리는 대기시간을 최소화할수 있는 검사주기와 데이터 보유량에 대한 최적점을 구하는 연구를 추가하여 진행하고자 한다.

Acknowledgement

"이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단-미래기반기술개발사업(첨단융복합분야)의 지원을 받아 수행된 연구임[N01100428]"

참고문헌

[1] A. Chakrabarti, etal "Scalable and Distributed Mechanisms for Integrated Scheduling and Replication in Data Grids" LNCS, 2008, Vol 4904/2008, 227-238

[2] K. Yang, et.al, "In-Memory Grid Files on Graphics Processors" Proceedings of the Third Int'l Workshop on Data Management on New Hardware Jun 15, 2007

[3] David R. Cox, et.al, "The theory of stochastic processes" Chapman & Hall/CRC, 2001

[4] The SimJava Tutorial, http://www.dcs.ed.ac.uk/home/hase/simjava/