실시간 스트림 프로세싱 시스템에서의 버퍼 컨트롤 최적화 기법

김병상*, 김대순**, 윤찬현**
*한국과학기술원 정보통신공학과
**한국과학기술원 전기전자공학과
e-mail:{bs.kim, sundae21, chyoun}@kaist.ac.kr

Optimal Buffer Control in Real-Time Stream Processing Systems

Byungsang Kim*, DaeSun Kim**, Chan-Hyun Youn**
*Dept of Info. and Comm., KAIST
**Dept of Electrical and Electronic, KAIST

요 약

스트림 프로세싱 시스템은 실시간 데이터 수집 장치와 대규모 분산 컴퓨팅 환경이 결합되어 데이터 생성과 가공을 통하여 다수의 결과를 병렬적으로 도출하는 분산 프로그래밍 모델이다. 본 논문에서는 프로세스간에 필수적으로 요구되는 유입데이터 버퍼 관리에 초점을 두고 있다. 데이터의 유입률에 따라 동적으로 분석 프로세스를 확장시킴으로서 프로세스간 버퍼의 크기를 제어하는 기법을 제안하며 시뮬레이션을 통하여 성능 분석을 하였다.

Real-Time Stream Processing System, Optimal Buffer Control, Adaptive Scheduling

1. 서론

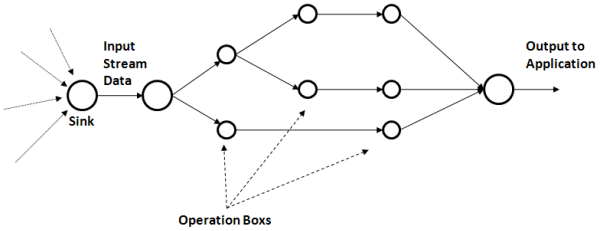
대규모 분산 컴퓨팅 환경에서 대용량의 데이터 분석을 위해 분산 파일 시스템과 MapReduce 모델의 유용성이 높이 평가되고 있다. 이러한 모델은 대규모의 데이터를 분산된 파일 시스템에 독립적으로 분할하여 저장함으로서 의존성을 최소화 시키고 병렬 처리를 최대화함으로서 산출성능(throughput)을 높이고자 한다. 하지만 이러한 모델은 모든 데이터가 파일 시스템에 저장되어 있는 배치(batch) 모형을 기반으로 하기 때문에 대규모 데이터의 실시간 분석에는 적당하지가 않다. 스트림 프로세싱 시스템은 실시간으로 생성되는 데이터를 저장과 동시에 사전에 정의된 실행인자(쿼리)를 기반으로 결과를 산출하는 모형이다. 따라서 데이터 수집장치(센서, 이벤트 시스템)와 연동하여 데이터의 흐름을 기반으로 결과를 획득하는 것을 목적으로 한다[1]. 최근에 이와같은 스트림 프로세싱 엔진으로서 Aurora[2], STREAM[3], TelegraphCQ[4] 와 같은 인프라스트럭처 소프트웨어가 활용 되고 있으며 클라우드와 같은 대규모 인터넷 기반의 분산 컴퓨팅 환경으로 그 영역을 확대하고 있다. 이러한 스트림 프로세싱 엔진은 실시간으로 제공되는 스트림 데이터를 영구저장장치(Hard disk)가 아닌 메모리기반(In-Memory) 버퍼에 저장 후 전송(Store and Forward)되기 때문에 분석 노드로의 유입(Input Stream) 데이터와 분석 후 유출(Output Stream) 데이터의 흐름 관리는 필수적이다.

본 논문에서는 유입률 기반의 최적 버퍼 제어 모형을 제안한다. 유입률에 따라 동적으로 프로세싱 능력을 확장 및 감소 시킴으로서 프로세스간 버퍼의 크기를 일정하게 유지함으로서 결과적으로 실시간 데이터 스트림의 대기지연시간을 조절하는것을 목표로 한다.

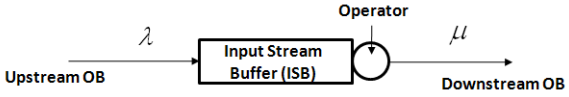
2. 유입률 기반의 최적 버퍼 제어 모형

2.1 스트림 프로세싱 모형

스트림 프로세싱 모델은 수학적으로 그래프 모델(V,E)로 정의될수 있으며 그림 1-a에서 보는것과 같이 점(vertices)은 분석을 담당하는 Operation Box (OB)로, 각의 선(edge)은 데이터의 입/출력 데이터 흐름을 나타낸다.



(a) 스트림 프로세싱 모델



(b) Queuing Model for Operator Box

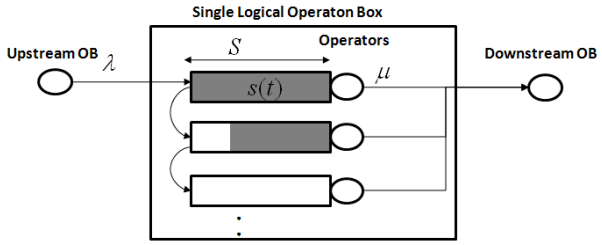
(그림 1) 실시간 스트림 프로세싱 시스템 모형

따라서 단일 OB는 그림 1-b와 같이 유입 데이터 버퍼(Input Stream Buffer:ISB) 와 분석기(Operator)로 모델링 될 수 있으며 각각의 OB는 자신의 버퍼(ISB)의 데이터를 파라미터로 분석작업을 수행한뒤 하위에 위치한 OB의

ISB로 출력 데이터를 넘겨준다.

2.2. 유입률 기반의 동적 OB 확장 기법

본 논문에서는 논리적으로 단일의 OB를 구성하는 다수의 물리적 Operator 모델을 제안한다. 그림 2에서 보는것과 같이 유입률 기반의 OB 할당 기법은 Operator 수의 변화를 통하여 ISB의 크기를 동적으로 제어하는 것을 의미한다.

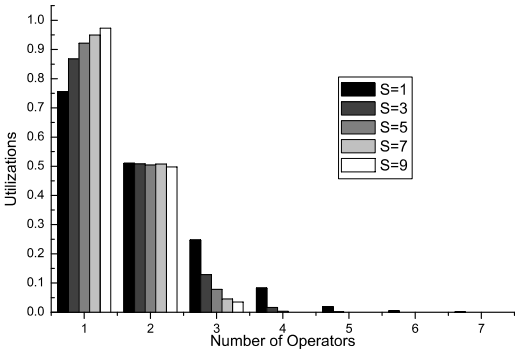


(그림 2) Dynamic Creation and Deletion of the Operations

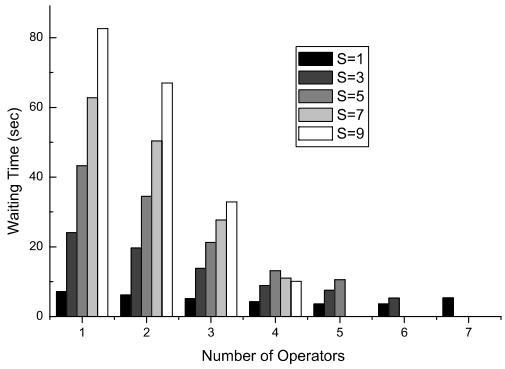
데이터 유입률을 λ, 단일 Operator의 분석 시간을 μ로 가정하자. 또한 ISB의 최대 버퍼 크기(Maximum Buffer Size:MBS) 로서 S 를 정의하자. 즉, 시간 t 에서의 대기하는 데이터 스트림의 양 $s(t)$가 S보다 클 경우 Operator를 증가시키고 반대로 현재의 $s(t)$가 S보다 작을 경우 Operator를 감소시킨다. 이와 같은 모델은 유입률(λ)가 포아송분포를 따르고 서비스 시간 (μ)가 지수분포를 따를때 $M/M/c$ 모델의 서버수 제어 문제로 모델링 할 수 있다. 여기서 c 는 제안된 모델에서 Operator의 수를 의미한다. λ 와 μ 를 마코비안 특성을 가진다고 가정할 경우 이론적인 출생사멸과정을 통하여 본 모델의 안정상태를 구할 수 있지만 복잡도가 증가하며 일반 분포에 적용할 수 없는 제약이 있으므로 본 논문에서는 시뮬레이션을 통하여 성능 분석을 도출하고자 한다.

3. 성능 분석

시뮬레이션은 SimJava[5] 툴을 이용하였다. 유입률(λ)과 단위 Operator의 서비스 시간(μ) 은 마코비안 성질을 가정하였다. 그림 3은 $\mu = 10s$ 및 $\lambda = 0.25/s$로 가정하고 MBS의 크기 S={1,3,5,7,9}로 변화 시켰을 때 각 Operator의 사용 개수 및 평균 이용률, 평균 대기시간을 나타내고 있다. 그림 3-a 과 3-b에서 가로축은 각각의 설정에서 사용된 Operator의 개수이다. S=1 일 경우 최대 7개의 Operator가 사용되었으며 MBS가 증가함에 따라 사용되는 Operator의 개수는 감소한다. 또한 그림 3-a는 사용된 Operator 에서의 이용률을 나타내고 있다. MBS가 작을수록 전체적인 이용률은 감소하지만 그림 3-b에서 보는것과 같이 평균 대기시간은 크게 감소하는 것을 볼수 있다. 따라서 우리는 최적의 MBS 결정하기 위해 이용률과 대기시간과의 상충관계를 통하여 결정하는 것이 가능하다.



(a) ISB Utilization 분석



(b) OB 대기시간 분석

(그림 3) MBS 변화에 따른 성능 요소 분석 $\mu = 10s$ and $\lambda = 0.25/s$

4. 결론

본 논문에서는 실시간 스트림 프로세싱 시스템에서의 유입률 기반의 최적의 버퍼 제어 기법에 대해서 분석하였다. 분석된 결과로서 버퍼의 크기는 전체 Operator 개수 및 이용률 그리고 대기시간을 결정할수 있으며 따라서 최적의 버퍼 크기는 두 개의 성능 요소를 고려하여 결정되수 있음을 성능분석을 통하여 고찰하였다.

Acknowledgement

"이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단-미래기반기술개발사업(첨단융복합분야)의 지원을 받아 수행된 연구임[N01100428]"

참고문헌

[1] S. Michael, et.al. The 8 Requirements of Real-time Stream Processing, ACM SIGMOD, 2005

[2] D. Carney, et.al. Monitoring Streams: A New Class of Data Management Applications. VLDB'02, Hong Kong, China, 2002.

[3] A. Arasu, et.al. STREAM: The Stanford Stream Data Manager. In ACM SIGMOD, June 2003.

[4] S. Chandrasekaran, et.al, TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. 1st CIDR Conference, Asilomar, CA, 2003.

[5] http://www.dcs.ed.ac.uk/home/hase/simjava/