데이터 스트림 기반 빅데이터 시스템에 대한 연구

이승용* · 김종민* · 황대훈**

1. 서 론

2000년을 맞이하던 IT의 상황은 급변하는 인터넷에 의해, 생산해 내는 데이터 역시 다양성과 규모에서 엄청난 성장을 했다. 포털 중심의 기업형 인터넷 정보에서 개인 미디어 정보를 바탕으로 한 소셜 네트워크 인터넷 정보로 IT 환경이 바뀌었고, 이러한 변화는 사용자들의 정보의 취득 방법을 기업에서 제공해 주는 정보만을 확인하는 수동적 정보에서 인터넷을 사용하는 모든 사람들의 정보를 검색하여 확인하는 능동적 정보 유형으로 바뀌었다는 것을 의미하고 있다.

또한, 분산 환경을 가진 기업이나 조직들이 사물인터넷 개념을 이용하여 지리적으로 넓게 퍼져 있는 분산 응용을 개발하는 경향이 증가하고 있으며, 분산 응용의 증가는 지리적으로 분산된 데이터 소스로부터의 예측할 수 없는 많은 양의 데이터의 유입을 초래하였다.

이와 같이 데이터의 증가에 따른 데이터 발생과 동시에 처리를 필요로 하는 즉시성(timeliness)과 흐름처리(flow processing)라는 개념은 정보흐름 처리 시스템이라는 새로운 유형의 시스템의 수요에 대한 근거를 제공한다. 정보흐름에서 유용한 정보를 얻기 위해서는 이를 지속적으로 적시에 처리해야 하는데, 전통적인 데이터베이스 관리 시스템(Database Management System: DBMS)에서는 정보를 얻기 전에 먼저 데이터를 저장하고 인덱스등을 생성하는 과정을 거쳐야 하고, 사용자가 질의어의 작성을 통해 명확히 요청하는 경우에만 비동기적으로 정보를 제공하기 때문에 즉시성을 요하는 정보 흐름 처리에 대한 요구사항을 충실히 수행하기가 어렵다.[1]

수많은 정보흐름 처리 시스템들이 이러한 요구사항들을 반영하여 이미 설치된 처리 규칙에 따라 정보를 처리하도록 개발되었다. 정보흐름 처리 시스템들은 공통된 목적을 가지고 개발되었음에도 불구하고 해당 시스템들이 서로 다른 커뮤니티에서 고안되었기 때문에 시스템 구조, 데이터 모델, 규칙 언어, 처리 메커니즘 등의 관점에서 많은 차이가 있다.

현재까지 제안된 대표적인 정보흐름 처리 모델은 데이터 스트림 처리 모델(Data Stream Processing Model)[2] 과 복잡 이벤트 처리 모델(Complex Event Processing Model)[3]로 구분할 수 있다. 데이터 스트림 처리 모델은 정보 흐름 처리 문제를 서로 다른 데이터 소스로부터 유입되는 데이터 스트림을 처리하여 새로운 출력 스트림을 생성하는 것으로, 정보 흐름 처리를 기존의 데

* ㈜카디날정보기술클라우드연구소
** 가천대학교 IT대학

이터베이스 관리 시스템이 수행하였던 전통적 데이터 처리의 진화된 형태이다.

반대로 복잡 이벤트 처리 모델은 정보항목(information items) 흐름을 높은 수준의 이벤트의 관점에서 무엇이 일어났는지를 이해하기 위해 필터링 또는 결합되어야 하는 외부세계에서 일어난 이벤트 통지로 정의한다. 따라서 이 모델은 특정한 이벤트 패턴의 발생을 탐지하는 데 중점을 두고 있으며, 분산처리 시스템, 비즈니스 자동화, 컨트롤 시스템, 네트워크 모니터링, 센서 네트워크, 미들웨어 등 다양한 분야에서 제시되어 왔다.

본 논문에서는 빅데이터 시대의 데이터 처리를 위한 정보 흐름 처리 모델과 빅데이터의 배치 처리 시스템과 실시간 처리 시스템에 대해 알아보고, 데이터 처리에 대한 다양한 요구사항을 충족할 수 있는 배치 처리와 실시간 처리가 융합된 빅데이터 처리 시스템의 구현 방안을 제시한다.

2. 정보 흐름 처리 모델

2.1 데이터 스트림 관리 시스템

데이터 스트림 관리 시스템(Data Stream Management System: DSMS)[4]는 정적인 데이터를 처리하는 전통적인 데이터베이스인 데이터베이스 관리 시스템과 유사한 시스템이지만, 정적인 데이터가 아닌 연속된 데이터 스트림을 관리한다.[그림 1]의 (a)와 같이 DBMS는 DSMS와 같이 외부 이벤트 소스를 고려하고 있지만, 모든 관련 데이터가 저장되고, 비교적 자주 갱신되지 않는 영구 저장소에 설치되기 때문에, 외부 이벤트 데이터의 유입이 한계점을 초과하면 성능에 부정적인 영향을 미친다.

반면, DSMS는 데이터에서 정보를 취득하기 위해, DBMS에서 제공되는 유사한 질의를 사용하지만, DBMS와는 달리 한번 실행된 질의는 시스템에 설치되어 스트림 데이터가 입력되는 특정 주기마다 질의가 실행된다. 그러므로, 질의는 실행된 시점부터 사용자에 의해 실행을 종료할 때까지 연속적으로 수행된다. 대부분의 DSMS는 상기와 같이 데이터 중심(data-driven)이기 때문에, 연속 질의는 한 새로운 데이터가 시스템에 입력될 때마다 새로운 결과를 얻어 낼 수 있다.

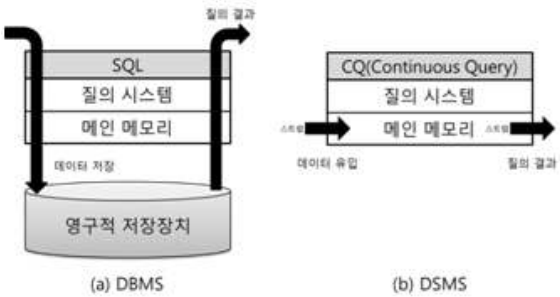

그림 1. DBMS와 DSMS 시스템

DSMS는 전통적인 DBMS와 차이점을 [표 1]에서 살펴 볼 수 있다. DSMS에서 처리되는 스트림은 일반적으로 무한하며 무작위로 유입되기 때문에, 데이터의 도착 순서에 대한 가정을 할 수 없고, 규모와 시간 제약은 데이터 스트림 요소가 도착한 후에 저장하고 처리하는 것을 어렵게 한다. 따라서 DSMS는 유입된 데이터 스트림을 특정 주기로 한 번에 데이터를 처리하는 것이 일반적인 메커니즘이다.

분류	DBMS	DSMS
데이터 지속성	영구적 데이터	휘발성 데이터 스트림
액세스 유형	무작위 액세스	순차 액세스
질의형태	일회성 질의	연속성 질의
저장용량	저장장치 용량	메인 메모리 용량
데이터 대상	현재 저장된 데이터 고려	데이터 입력 순서 고려

갱신속도	상대적으로 낮은 갱신 속도	잠재적으로 높은 갱신 속도

표 1. DBMS와 DSMS의 비교

DSMS을 가장 잘 설명하는 모델은 Babu와 Widom(2001)[3]가 제안하였다. DSMS는 끊임없이 변화하는 입력 데이터 스트림을 지속적으로 갱신되는 쿼리에 대한 응답을 만드는데 초점을 맞추고 있다. 이 시스템은 일반적으로 시퀀스를 고려하고, 관계를 정하는 복잡한 패턴 감지와 통지는 지원하지 않는다.

2.2 복합 이벤트 처리 시스템

복잡 이벤트 처리(Complex Event Processing: CEP)[6] 시스템은 처리되는 정보 항목들에 정확한 의미를 부여하여, 여러 데이터 소스로부터 발생된 정보에 대해 관찰 결과 이벤트를 통지하는 처리과정을 말한다. CEP은 여러 데이터 소스로부터 발생한대량의 데이터를 입력 이벤트로 정의한다. CEP은 입력 이벤트를 실시간으로 전달 받아 의미 있는 데이터를 추출하기 위해이벤트에대응되는 액션을 수행하고, 결과 이벤트를 통지한다. 이때데이터 소스로부터 발생되는 이벤트는 대량으로 지속적으로 입력되는 데이터 스트림으로 시간 순서가 중요한 데이터, 끝이 없는 지속적으로 발생되는 데이터의 성격을 가진다.

CEP은 구독자가 복합 이벤트로 그들의 관심을 표현하는 것을 허용하는 전통적인 게시-구독(publish-subscribe) 시스템의 확장이라고 볼 수 있다. 복합 이벤트는 다른 이벤트의 발생에 의존한다. CEP은 대부분의 데이터 스트림 관리 시스템의 한계, 즉 정보 흐름의 복잡한 패턴을 감지하는 능력을 강조하기 때문에, 일반적으로 무시하였던 대역폭 활용과 종간(end-to-end) 지연 같은 문제를 최적화하는 데 집중한다.

정보 흐름 처리 시스템은 서로 다른 커뮤니티에 의해 개발되었다. 데이터 스트림 관리 시스템은 주로 데이터의 흐름과 데이터의 변환에 중점을 두었다. 반면 복잡 이벤트 처리 시스템은 복잡한 이벤트 패턴을 포착하는 것에 특별한 관심을 가지고, 이벤트 통지를 처리하는 데 중점을 두었다. 복잡 이벤트 처리 시스템은 부하를 분산하고 통신 비용을 줄이기 위해 통신 측면에 중점을 두었다.

CEP에서 처리하는 이벤트 처리 기술은 다음과 같이 분류된다.

- 이벤트 패턴 인식
- 이벤트 추상화
- 이벤트 필터링
- 이벤트 집합과 변환
- 모델링 이벤트 계승
- 이벤트 사이의 관계(인과, 멤버쉽, 타이밍 등) 인식
- 추상화된 이벤트 위주 처리

3. 연속 질의(Continuous Query)

DSMS와 CEP은 모두 연속적으로 발생되는 데이터 스트림을 입력 받아 특정한 조건에 의한 결과를 생산한다. 연속 질의를 제공하는 대부분의 시스템은 표준 아키텍처가 없기 때문에, 전통적인 DBMS의 SQL과 같이 사용자 질의를 연산 계획에 따라 변환할 수 있는 언어 형식을 기반으로 하고 있으며, 처리 과정은 [그림 2]와 같이 4단계로 구성된다.

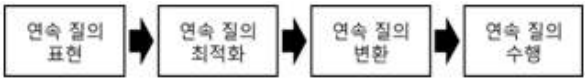

그림 2. 연속 질의 처리 과정

3.1 연속 질의 표현

대부분의 연속 질의는 DBMS의 SQL과 같은 선언적 언어를 사용한다. 연속 질의 표현 방법에 대한 표준이 제정되지 않았기 때문에, 연속 질의를 제공하는 방법이 여러가지 존재하지만, 대표적인 연속 질의 언어로 CQL(Continuous Query Language)[5], StreamQL(Stream Query Language), 그리고 EPL(Event Processing Language)[7]들을 예로 들 수 있다. 상기 연속 질의 표현언어는 각 처리 단계를 한 개의 박스로 표현하고 각 박스들 사이를 화살표를 이용하여 표현함으로써 처리 순서를 그래프로 제공 할 수 있다.

연속 질의 표현 언어는 프로세스 모델(process model)로 특정 기간(시간 또는 입력 받은 데이터 개수)동안의 데이터를 처리하는 윈도우(window)를 정의한다. 연속 질의 언어의 윈도우는 DBMS 시스템의 SQL에 정의되지 않은 개념으로 뚜렷한 표준이 없다. CQL과 EPL은 시간 개념의 윈도우를 제공하는 반면, StreamQL은 입력 요소의 크기에 따른 윈도우를 제공한다. 다음 예는 StreamQL의 예로 마지막 10개의 튜플중에서 100이 넘는 값의 평균을 계산하는 연속 질의를 정의한다.

```
SELECT AVG(price) FROM examplestream [SIZE 10 ADVANCE 1 TUPLES] WHERE VALUE > 100.0
```

선언된 질의는 다음 단계에서 논리적 질의 계획으로 변역된다. 질의 계획은 연산을 표현하는 노드와 처리 흐름을 나타내는 연결선으로 구성된 방향 그래프로 표현되고. 질의 계획의 각 노드는 필터링(filtering)과 집합(aggregation)과 같은 특정 연산을 정의할 수 있다. 필터링은 SQL의 WHERE 절의 조건 검색을 의미하고, 집합은 함수를 이용하여 질의 단위를 시간 또는 이벤트 개수 단위를 의미한다.

3.2 질의 최적화

논리적 질의 계획은 스트림 모델(stream model)을 기반의 최적화를 수행한다. 최적화된 연속 질의의 기본 개념은 데이터베이스 시스템과 유사하다. 연속 질의가 서로 연관된 두 개 이상의 데이터 스트림을 입력 받고, 이를 데이터 스트림을 관계대수로 번역하여 동치관계를 가지는 관계 연산을 최적화를 수행한다.

DBMS는 서로 다른 동일한 질의 계획 중에서 가장 적은 비용을 지불하는 질의 계획을 선택하는 비용 기반의 최적화 기술을 사용한다. 예를 들어, 두 개의 연속적인 조인 연산에서 가장 최적의 질의 계획을 수행하는 경로를 찾아 질의를 수행한다. DBMS는 통계적 방법으로 최적화된 경로를 가지는 질의 계획을 선택하지만, 연속 질의는 정적인 데이터이기 보다는 스트림 데이터이기 때문에 DBMS의 통계적 방법에 의한 최적화된 질의 계획을 수립할 수 없다. 하지만, 특정 시간 동안 임의의 통계를 얻기 위해 관찰하는 것이 가능하기 때문에, 질의를 수행하는 실시간 동안 질의를 최적화할 수 있다. 그러므로, 연속 질의 시스템은 질의를 수행하는 과정에서 새로운 질의 계획으로 대체할 수 있는 전략으로 변경하는 기능을 수반하기도 한다.

3.3 질의 변환

연속 질의 시스템의 논리적 연산은 처리 알고리즘이 포함되지 않은 의미적 연산으로만 존재한다. 따라서 논리적 질의 연산은 수행이 가능한 연산 알고리즘으로 변환되어야 한다. 이러한 과정을 물리적 질의 계획이라고 한다. 논리적과 물리적 연산 사이의 차이점은 하나의 논리적 연산이 한 개 이상의 물리적 연산으로 대응된다는 것이다. 예를 들어 조인은 인접 루프 조인(nested loop join) 또는 분류 합병 조인(sort-merge join)과 같은 서로 다른 알고리즘으로 대응될 수 있다.

3.4 질의 수행

연속 질의 시스템은 수행 가능한 알고리즘을 내장한 물리적 질의 계획을 포함하고 있다. 연속 질의 시스템은 데이터 위주(data-driven) 기반 시스템이기 때문에, 질의는 소스로부터 수신되는 데이터 요소를 연속 질의 언어로 표현되어 번역된 질의 계획에전달함으로써 실행된다.데이터 요소는 질의 계획에 대응된 연산자를 통과 할 때마다, 데이터 요소에 대해 특정한 연산을 수행하여 결과를 생산하고, 해당 결과를 다시 입력으로 받는 모든 연산자에게 결과를 전달한다.

4. 빅데이터 시스템

빅데이터는 '기존 데이터베이스 도구의 데이터 수집, 저장, 관리, 분석하는 역량을 넘어서는 데이터셋 규모로, 시간이 지남에 따라 기술 진보로 빅데이터로 간주될 수 있는 데이터셋의 크기도 같이 증가한다고 가정하며, 데이터량 기준에 대해서는 일반적으로 사용되는 소프트웨어의 종류와 산업분야에 따라 상대적이며 현재 기준에서는 몇십 테라 바이트에서 수 페타바이트까지가 그 범위이다.'라고 맥킨지(McKinsey)는 정의하고 있다.

빅데이터는 기존의 도구로 처리의 역량이 넘어가는 데이터 셋을 처리하는 개념이기 때문에, 분산 시스템을 기반으로 하고 있다. 만약, 분산 시스템을 사용할 수 없다면, 1대의 컴퓨터가 처리할 수 있는 데이터의 용량이 한계로1대가 아무리 큰 데이터 셋을 처리하고 있다고 해도 빅데이터 시스템이라고 볼 수 없다. 따라서, 빅데이터 시스템은 데이터 증가에 따라 확장 가능한 분산 시스템을 기반으로 하며, 분산 시스템을 이용한 저장이 가능한 데이터 웨어하우스(data warehouse)와 데이터 분석을 위한 데이터 마이닝(data mining)을 혼합한 시스템이다.

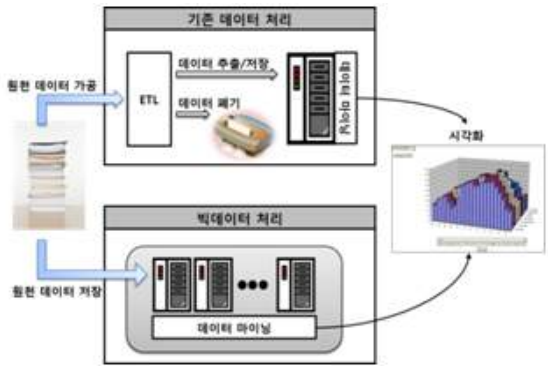

그림 3. 기존 데이터 처리 과정과 빅데이터 초리 과정

[그림 3]과 같이 데이터를 통해 정보를 추출하는 데이터 마이닝 처리 과정을 기존 방식과 빅데이터 방식으로 구분하여 살펴보면, 기존 방식은 분석을 위한 대용량의 데이터 중에서 표본추출을 통해 일정량의 데이터를 추출하고, 추출된 데이터를 기반으로 통계/분석을 수행하는 과정이다. 이때 데이터 마이닝을 처리에 사용되지 않은 데이터는 폐기된다. 하지만, 빅데이터 시스템은 표본추출이 아닌 사용 가능한 모든 데이터를 기반

으로 데이터 마이닝을 수행하여, 기존 방식에서 데이터 폐기에 따른 분석하지 않은 데이터까지 모두 마이닝의 대상이 된다.

4.1 배치 처리 시스템과 실시간 처리 시스템

빅데이터 시스템은 크게 배치 처리 방식과 실시간 처리 방식으로 구분된다. 배치 처리 방식은 기존 DBMS 방식과 같이 영구적인 저장장치에 데이터를 보관하고, 보관된 데이터를 기반으로 질의를 수행하여 모든 데이터들간의 상관관계를 분석하는 방법이고, 실시간 처리 방식은 DSMS 또는 CEP 방식과 같이 스트림으로 유입되는 데이터를 일정 주기(시간 또는 데이터 개수) 동안의 지속적으로 변화는 상관관계를 처리하는 방법이다. 대표적인 빅데이터 배치 처리 시스템으로 Hadoop[8]을, 실시간 배치 처리 시스템으로 Twitter의 Storm과 Yahoo의 S4가 있다.

4.2 Storm

빅데이터 실시간 처리 시스템으로 대표할 수 있는 Strom[9]은 Backtype에서처음 개발되었으며, 트위터가 실시간으로 스펨을 추출하기 위해 Backtype사를 인수하여 실시간 분석용으로 최적화시켜 오픈 소스화 시킨 확장 가능한 실시간 분산처리 시스템이다.

Storm은 노드들이 서로 연결된 네트워크 상에 연속적으로 유입되는 데이터를 처리되는 데이터 흐름 모델(data flow model)을 구현한 것으로, 시스템에 유입되는 데이터는 표준 데이터 유형(정수형, 실수형, 그리고 바이트 배열) 또는 몇 가지 추가된 직렬화 코드를 포함한 사용자 정의 유형으로 정의된 튜플(tuple)로 정의하고, 연속된 튜플을 스트림(stream)으로 정의한다.

각각의 스트림은 데이터 소스와 연결된 Spout이존재하고 Spout는 유입된 데이터를 튜플 단위로 생성한다. Spout에서 생성된 데이터 튜플은 연산을 위해 Spout에 연결된 다수개의 Bolt로 전달되고 Bolt는 연산 처리 결과를 새로운 튜플로 생성한다. Bolt와 Bolt는 서로 연결되어 연속된 연산을 수행할 수 있으며, Spout과도 연결된다. [그림 4]와 같이 Spout과 Bolt가 연산 처리에 따른 논리적 연결 형태를 토폴로지(topology)라고 정의한다.

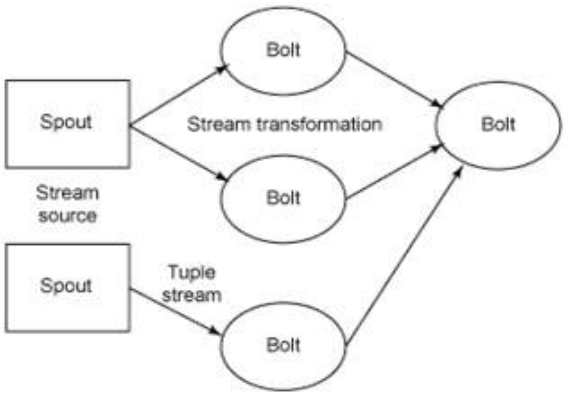

그림 4. Storm의 토폴로지 구성

4.3 융합 빅데이터 처리 시스템

빅데이터 시스템은 처리 방식에 따라 배치 처리와 실시간 처리로 구분되며, 데이터 분석 방법에 따라 적용되는 기술도 다양하다. 일반적으로 실시간 처리 기술은 일정 주기 동안 변화되는 데이터를 분석하여 특정 규칙에 의해 이벤트를 발생시키는 개념이다. 여기서 규칙이란 대수적 연산에 의한 규칙으로 결정된다.

반면, 배치 처리 시스템은 모든 데이터의 상관관계를 이용한 규칙 생성이 가능하다. 빅데이터 분석에서 상관관계, 회귀분석, 시계열분석, 및 다차원분석을 통해 데이터의 연관관계를 찾고, 그들의 연관관계를 하나의 규칙으로 정의할 수 있

다.

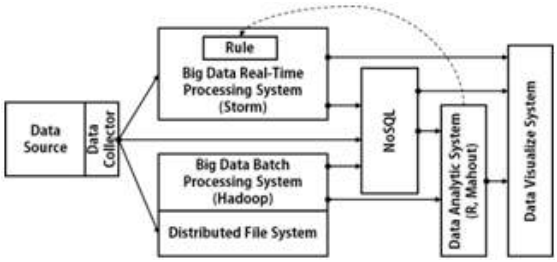

그림 5. 융합 빅데이터 처리 시스템 구성도

[그림 5]는 빅데이터 배치 처리와 실시간 처리가 융합된 빅데이터 처리 시스템의 구성도를 예시한다. 빅데이터 배치 처리 시스템으로 Hadoop을 사용하고, 실시간 처리 시스템으로 Storm을 사용한다. Storm은 특정한 규칙에 의해 데이터 스트림의 연산을 수행하고 수행된 연산 결과를 NoSQL 또는 실시간 시각화 시스템에 전달한다. Hadoop은 데이터 소스로부터 전달 받은 원본 데이터를 보관하여 모든 데이터의 연관관계를 분석하며, 생성된 연관관계는 시각화 자료로 또는 실시간 처리 시스템의 새로운 규칙으로 적용된다.

5. 결론

IT 기술과 무선 통신 기술의 발전은 거대한 데이터 흐름을 생성하고, 이러한 정보의 흐름은 기존의 데이터베이스 시스템보다 새로운 처리 패러다임을 요구하게 되었다. 이러한 요구사항은 빅데이터 분야에 적용되어, Hadoop ECO 시스템과 같은 솔루션 생태계를 구성하게 되었다. Hadoop ECO 시스템은 Hadoop의 배치 처리 방법에 의해 정보 흐름을 실시간으로 처리하지 못하는 문제를 가지고 있기 때문에, Twitter의 Storm과 같은 실시간 분산 처리 시스템이 새로운 대안으로 제시되고 있다.

본 논문에서는 빅데이터를 다루는 배치 처리와 실시간 처리 시스템을 고찰함으로써 각 시스템의 특징을 살펴보았으며, 두 시스템을 혼합한 융합 빅데이터 처리 시스템의 구성 방안을 제안하였다. 빅데이터는 사용자의 변화는 요구를 실시간으로 대응하기도 하면서, 누적된 대용량 데이터에서 새로운 규칙을 찾아 지속적인 변화를 감지하는 시스템이다. 융합 빅데이터 처리 시스템은 이러한 요구사항을 잘 반영하고 있는 시스템으로 여러 분야에서 다양하게 적용할 수 있는 시스템이다.

참 고 문 헌

[1] 황은정(2012-12), 정보 흐름 처리: 데이터 스트림에서 복잡 이벤트 처리까지, Embedded News

[2] Babcock, B., S. Babu, et al. (2002). Models and issues in data stream systems. Proceedings of the twenty-first ACM SIGMOD - SIGACT - SIGART symposium on Principles of database systems. Madison, Wisconsin, ACM: 1-16

[3] Luckham, D. C. (2001). The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Boston, MA., Addison-Wesley Longman Publishing Co., Inc..

[4] ArvindArasu, Brian Babcock, ShivnathBabu(2004), STREAM: The Stanford Data Stream Management System, Technical Report, StanfordInfoLab.

[5] A Arasu, S Babu, J Widom(2004), CQL: A Language for Continuous Queries over Streams and Relations, Database Programming Languages, Springer

[6] DC Luckham(2002), The power of events, Addison-Wesley

[7] Ted Won(2011), Complex Event Processing

with Esper: Real-time Event Stream & Complex Event Processing, JBoss Community
[8] http://hadoop.apache.org/
[9] http://storm.incubator.apache.org/