Journal of Convergence for Information Technology
Vol. 9. No. 1, pp. 45-53, 2019
e-ISSN 2586-4440
DOI : https://doi.org/10.22156/CS4SMB.2019.9.1.045

실시간 웹 크롤링 분산 모니터링 시스템 설계 및 구현

김영아[1], 김계희[2], 김현주[3], 김창근[3*]
[1]경남과학기술대학교 컴퓨터공학과 학생, [2]경남과학기술대학교 컴퓨터공학과 강사
[3]경남과학기술대학교 컴퓨터공학과 교수

Design and Implemention of Real-time web Crawling distributed monitoring system

Yeong-A Kim[1], Gea-Hee Kim[2], Hyun-Ju Kim[3], Chang-Geun Kim[3*]
[1]Student, Department of Computer Science & Engineering, GNTECH
[2]Lecturer, Department of Computer Science & Engineering, GNTECH
[3]Professor, Department of Computer Science & Engineering, GNTECH

요 약 급변하는 정보화 시대에서 웹사이트에 서비스되는 정보 과잉에 대한 문제들을 접하곤 한다. 정보가 많아도 쓸모 있는 정보는 없고, 필요한 정보를 선택하는데 불필요한 시간이 많이 소비 된다. 검색 엔진과 같은 여러 사이트에서는 데이터의 최신 상태 유지를 위해 웹 크롤링을 한다. 웹 크롤링은 대부분 방문한 사이트의 모든 페이지의 복사본을 생성하는 데 사용되며 검색 엔진은 이렇게 생성된 페이지를 더욱 빠른 검색을 위해 인덱싱 한다. 많은 데이터 중에 정보가 실시간으로 변경되는 도매정보, 주문정보 등의 제한된 웹 데이터 수집은 일반적인 주제 중심의 웹 데이터 수집으로 무리가 있다. 현재 제한적 웹 정보를 실시간으로 수집하고 저장하는 방법에 대한 대안이 제시되고 있지 않다. 본 논문에서는 제한된 웹 사이트의 정보를 수집하고, 데이터의 상세 분석을 통한 수집 시간 예측과 분류 작업을 통해 병렬 시스템에 저장하는 웹 크롤링 분산 모니터링 시스템(R-WCMS)을 제안한다. 실험 결과 웹 사이트 정보 검색을 제안모델에 적용하여 15-17% 시간이 감소됨을 입증했다.

주제어 : 웹 크롤링, 빅 데이터, 하둡, 스파크, 카프카, 병렬시스템, 모니터링

Abstract We face problems from excessive information served with websites in this rapidly changing information era. We find little information useful and much useless and spend a lot of time to select information needed. Many websites including search engines use web crawling in order to make data updated. Web crawling is usually used to generate copies of all the pages of visited sites. Search engines index the pages for faster searching. With regard to data collection for wholesale and order information changing in realtime, the keyword-oriented web data collection is not adequate. The alternative for selective collection of web information in realtime has not been suggested. In this paper, we propose a method of collecting information of restricted web sites by using Web crawling distributed monitoring system (R-WCMS) and estimating collection time through detailed analysis of data and storing them in parallel system. Experimental results show that web site information retrieval is applied to the proposed model, reducing the time of 15-17%.

Key Words : Web Crawling, big data, hadoop, spark, kafka, Parallel systems, Monitoring

*This work was supported by Gyeongnam National University of Science and Technology Grant in 2018~2020
*Corresponding Author : Chang-Geun Kim(cgkim@gntech.ac.kr)
Received November 5, 2018 Revised December 4, 2018
Accepted January 20, 2019 Published January 28, 2019

1. 서론

웹 데이터 수집은 데이터의 생산·수집·처리·분석·유통·활용 등을 통해 가치를 창출하는 상품과 서비스를 생산·제공하는 데이터 산업의 한 분야를 차지한다. 지능형 정보사회의 도래는 사물인터넷, 빅 데이터, 인공지능 등 ICT(Information and Communication Technology) 환경 변화로 데이터 산업 시장 성장을 이끄는 주요 동력이 된다. 웹의 규모가 커지면서 서비스 되는 방대한 정보 수집을 웹 문서로 주기적으로 읽어 들여 자동으로 데이터베이스에 저장하는 웹 크롤러를 이용하고 있다. Apache Nutch과 Scrapy와 같은 오픈 소스 웹 크롤러 및 검색 엔진의 개발이 활발하게 이루어지고 있으며, 웹 자체를 제공하는 서비스도 있다[1]. 기존 상용 검색 엔진 구축에 사용 중인 소프트웨어 인 웹 크롤러는 앞으로의 연구와 데이터 산업의 비즈니스 분야에서 널리 이용 될 전망이다.

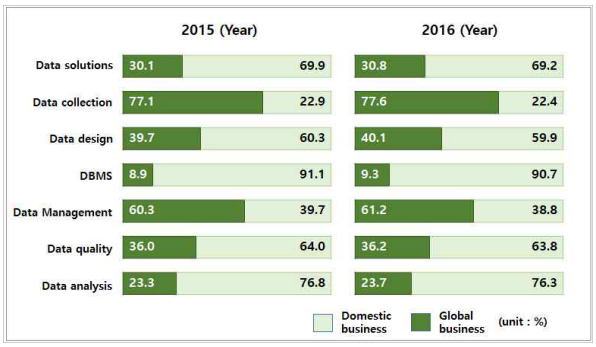

Fig. 1. Domestic data solution market share

Fig. 1의 통계청 자료에 따르면 국내 데이터 솔루션 시장 점유율 중 데이터 수집과 관리가 글로벌 산업체에 비해 점유비율이 0.5%, 0.9%로 증가한 것을 확인 할 수 있다. 이에 향후 데이터 산업은 2020년까지 연평균 증가율 3.5%의 성장세로 16조원 대 시장 진입이 임박할 것으로 전망된다고 보고되어 있다. 웹 크롤러는 시드 URL을 바탕으로 선별 - 수집 - 저장- 전달하는 과정을 통하여 새로운 웹 정보를 수집한다. 색인 URL을 바탕으로 반복적인 수집 과정을 거쳐 많은 양의 정보를 분류하고 저장한다[2]. 효율적이고 빠른 수집을 위해 병렬 웹 크롤러를 이용하며, 이런 과정에서 일반적인 웹 크롤러는 같은 정보의 중복 수집을 피하고자 URL매니저를 활용하여 중복된 페이지를 걸러낸다. 이와는 달리 상품정보 제공 사이트, 주문정보 수집사이트와 같은 특수 정보를 취급하는 도매 정보 사이트, 오픈마켓 SCM 등에서는 일정한 형식과 속성(attribute)을 가진 정보를 추출한다. 실제 판매 거래가 완료될 동안의 발생되는 정보 등으로 데이터의 잦은 변화가 일어난다. 제한된 웹 사이트에서의 웹 데이터 수집은 일반적인 웹 데이터 수집과는 다르게 중복된 URL을 걸러내지 않고, 같은 페이지에서 정보를 재수집하여 변경된 정보를 확인하고, 데이터를 갱신하는 시스템이 요구된다. 또한 웹은 광대하기 때문에 통신 대역을 활용하고 최대한 빠른 시간에 포괄적인 웹 데이터를 수집 할 필요가 있으며 동시에 각 웹 서버에 부하를 최소화 할 필요가 있다. 웹 크롤러가 동일한 웹 서버에 연속 사용되면 서버에 큰 부하가 생겨 최악의 경우 웹 서버가 중단 될 가능성이 있다[3,4]. 따라서 웹 크롤러는 동일한 웹 서버에 연속 사용을 피하면서 새로운 정보를 빠르게 서비스 한다는 목표를 달성해야한다. 본 논문은 수시로 변경되는 정보를 웹 환경에 적용할 수 있도록 주기적인 병렬 수집과 서버에 액세스 간격을 효율적으로 제어 하는 웹 크롤링 모니터 시스템인 R-WCMS를 제안한다.

2. 관련연구

2.1 Structure of Web Crawler

일반 검색 엔진에서 사용하는 매우 방대한 웹 전체의 문서를 원활하게 수집하기 위해 분산 웹 크롤러 방식을 사용한다. Google에서 제공하는 중앙 집중식과 Mercator와 같은 P2P방식 및 Multiple Site Crawlers 등이 있다. 웹 데이터 수집은 다운로더, 저장소, 스케줄러 등으로 이루어진다. 웹 데이터 수집 중 가장 중요한 것은 스케줄러이다. 스케줄러는 '프론티어(frontier)'로도 알려져 있다. 방문할 URL의 큐를 통해 하나 이상의 다운로더를 특정 순서로 관리 되며 큐에서 나오는 URL에 보낸다. 다운로더는 각 URL에 해당하는 내용을 검색해서 구문 분석 후 저장소에 보낸 다음 색인화 작업을 통해 내용을 검색하게 된다. 스케줄링 정책을 구동하는 중요한 정보를 가진 저장소는 해당 페이지에서 검색된 메타데이터를 스케줄러에 제공하게 된다.

Fig. 2는 웹 데이터 수집의 구조를 표현한 것이다. 웹 데이터 수집 기는 seedURL을 바탕으로 웹 문서를 다운로드하고 다운로드 받은 페이지에서 다시 URL을 추출한다. 추출된 URL의 중복을 확인하고 새로운 URL list를 저장하는 것을 반복하는 과정을 통해서 웹을 수집한다[5].

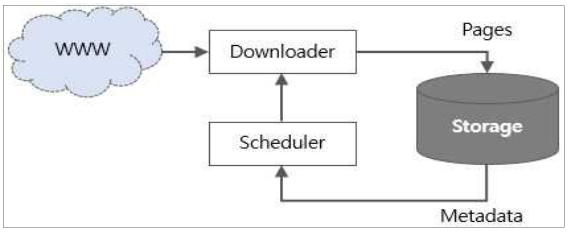

Fig. 2. Architecture of web crawler

```
Initialize:
  UrlsDone = ∅
  UrlsTodo = {"yahoo.com/index.htm" ..}

Repeat:
  url = UrlsTodo.getNext()
  ip = DNSlookup( url.getHostname() )
  html = DownloadPage( ip , url.getPath() )
  UrlsDone.insert( url )
  newUrls = parseForLinks( html )
  For each newUrl
    If not UrlsDone.contains( newUrl )
    then UrlsTodo.insert( newUrl )
```

Fig. 3. Typical Web Crawl algorithm

Fig. 3은 일반적인 웹 데이터 수집의 알고리즘 이다. 대역폭과 저장 용량과는 별도로 웹 데이터 수집 기의 구현 방법은 많은 어려움이 있다[6,7]. 웹 데이터 수집기가 다른 시스템과의 원할한 상호작용 문제와 여러 시스템 간의 다양한 표준 서비스 품질을 가지고 있기 때문이다. 가장 큰 어려움은 다중 자원으로부터 웹 정보 데이터를 받는 방법이다. 도메인과 IP를 이용한 DNS 서버를 이용하고, 데이터를 다운받은 이후에 쉬글링(shingling)을 통해 간단한 문법 규칙의 집합을 URL에 적용함으로써 중복 데이터를 제거한다. 웹 페이지를 통한 DOM(Document Object Model) 트리를 이용한 구문 파싱등의 중복된 데이터 수집 후 저장한다. 현재 방문할 페이지를 크게 줄여나가는 주제 중심의 웹 데이터 수집 방법(Focused Crawler)과 개인화 웹 검색 등이 연구되고 있다. 주제 중심의 웹 데이터 수집방법은 수평적인 검색이 아닌 수직적 검색이다. 특정 목적에 따라 맞춤 정보를 검색하는 방법을 통해 결과물의 품질을 높이는 방식이 주목된다.

2.2 Topical crawler

특정한 주제의 문서만을 수집하는 Topical crawler[8]는 해당 사이트의 구조를 미리 분석하여 정보를 관리 한다. 주제 중심으로 웹 문서를 수직적으로 수집하면 학술 문서, 커뮤니티 페이지, 뉴스 등에서도 일정한 속성(attribute)들로 분류하여 정보를 추출하여 저장할 수 있다. 학술 문서의 경우 저자, 제목, 출처, 작성년도, 참고 문헌 등으로 속성별로 추출이 가능하고, 커뮤니티 페이지의 경우도 게시판명, 글쓴이, 제목, 내용, 작성일, 태그 등으로 나누어 추출하여 데이터베이스에 저장할 수 있다. 또한 분류페이지(category), 목록페이지(list), 상세페이지(contents) 등 페이지의 형태에 따라 수집되는 정보의 개수나 속성이 달라진다. 상업용 웹 사이트는 비즈니스 특성에 따라 일정한 형식의 문서들로 구성된다. 대표적으로 오픈마켓이나 도매사이트, 쇼핑몰과 같은 온라인 유통 사이트의 경우를 보면 기능적으로는 상품판매와 고객 응대로 크게 나눠지고, 상품판매 페이지들도 다시 카테고리, 목록, 상세페이지로 분류할 수 있다. 상품 정보를 담고 있는 페이지들에서는 사용자에게 노출되어야 하는 필수 정보들이 요구되어 공통적인 속성을 가지고 구조화되어 있다. 그러나 상업용 웹 문서들은 다른 웹 문서들에 비교하여 잦은 정보의 변경이 일어난다. 일반적으로 전체 웹 페이지 중 7%가 1주일에 변하는데[9] 반해 상업용 웹 문서들은 전체적으로 많은 변화가 수시로[10] 일어난다. 따라서 상업용 웹 데이터 수집 기는 잦은 변화에 대응할 수 있게 설계되어야 한다.

2.3 Apache Kafka

Kafka[11]는 Pub / Sub 메시징 모델을 기반으로 한 분산 메시지 큐이다. Kafka는 여러 대의 Broker 노드에서 클러스터를 구성하고 클러스터에 Topic 이라는 큐를 만든다. 입력 메시지를 전송 받은 Producer는 입력 라이브러리를 통해 Broker 클러스터의 Topic을 저장한다. Consumer는 출력 라이브러리를 통해 Topic에서 메시지 정보를 전송 받는다.

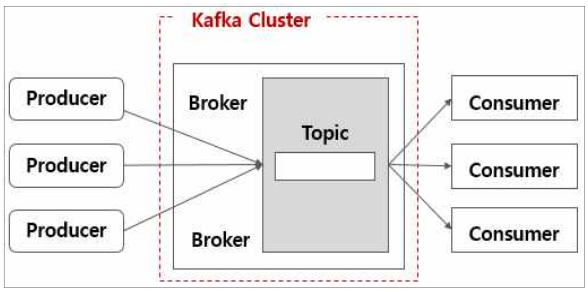

Fig. 4. Architecture of Apache Kafka

Fig. 4는 Kafka의 분산 메시지 큐의 구조를 표현한 것이다. Kafka는 하나의 Topic을 여러 대의 Broke에 분산 배치 한 파티션으로 구성된다. 파티션 단위로 데이터를 읽고 쓰며 하나의 큐에 병렬 단위로 입출력을 한다.

3. 제안 모델

제안모델은 인터넷의 실시간 업데이트 되는 상품 데이터를 수집, 저장하고 상품의 상세 정보를 분석 하는 웹 데이터 수집 모니터 시스템 R-WCMS이다.

R-WCMS은 수집된 데이터를 비교, 저장 분석하는 에이전트와 병렬 시스템에 적재된 데이터의 업데이트 시간을 예측하는 매니저로 설계했다.

3.1 웹 데이터 수집기

R-WCMS 에이전트는 웹의 다양한 문서 규칙에 따라 사이트의 링크를 정기적으로 순회하고 해당 페이지의 데이터를 수집 한다. 원하는 상품 코드로 데이터를 분류한 시스템에서 URL패턴을 이용하여 원하는 상품 코드로 범위를 제한 한다. 상품의 상세 정보를 반복적으로 수집한 데이터는 업데이트 된 데이터만 선별하여 R-WCMS 매니저를 통해 시스템에 적재 한다. 적재된 데이터는 업데이트 된 자료를 기반으로 비교 분석하며 시간당 발생할 수 있는 웹 데이터의 목록을 예측 하는 알고리즘을 적용한 정보를 모니터 한다.

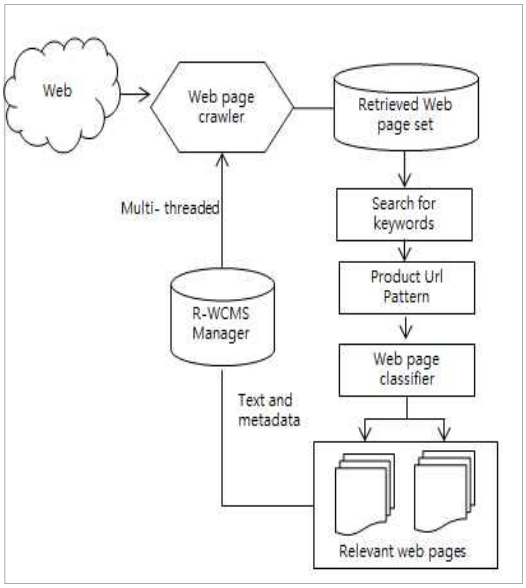

Fig. 5. Architecture of R-WCMS Agent

Fig. 5는 R-WCMS 에이전트의 전체 흐름을 보여준다.

A사:http://www.yyy.com/shop/search.php?page=5&sort=&rows=5&cat_search=&cat_code=&pick=
B사:http://www.xxx.com/shop/goods/goods_list.php?category=001&page=2

Fig. 6. Product Listing Page URL Pattern

Fig. 6은 상품 목록 페이지 URL 패턴의 구조를 표현한 것으로 수집한 상품 코드로 전체 상품 목록 페이지나 혹은 검색 페이지의 결과를 수집한다. 상품목록 페이지에서 전체 상품의 수와 상품코드를 수집한다. 전체 상품의 상품코드나 인덱스의 리스트를 확정하기 위해서 웹에서 정해진 상품 들을 주제어로 검색 한다. 전체목록이 없는 경우에는 카테고리별로 목록 페이지에서 상품 코드를 수집하여 통합을 한다. 목록페이지의 경우 상품의 수량에 따라 페이징처리가 되어있기에 첫 페이지부터 마지막 페이지까지 URL 패턴을 활용하여 반복 수집한다. 전체 상품의 수는 보통 목록페이지 상단에 별도로 표시(예: “총 4,053개의 상품이 준비되어 있습니다”) 되어 있는 경우가 많다. 이를 활용하여 목록페이지의 숫자를 추정하여 첫 페이지부터 마지막 페이지까지의 상품코드 인덱스를 안정적으로 수집 한다.

A사:http://www.xxxx.com/shop/goods/goods_view.php?goodsno=1169820&category=001

B사:http://www.yyy.com/shop/detail.htm?brandcode=1209-0416&page=&back_file=/

Fig. 7. Product detail page URL pattern

Fig.7은 상품 상세 페이지 URL 패턴으로 상품의 상세 정보는 전체 상품목록의 코드 값을 이용한다. 상품 상세페이지의 상품 URL 패턴으로 상품코드나 인덱스를 반복하여 수집을 한다. 수집된 상품코드는 seedURL과 함께 키와 값으로 저장된다. 웹 데이터 수집 시스템은 상세 페이지에 병렬적으로 접근한 후 세부 정보를 수집, 파싱하여 텍스트 파일로 저장 된다.

Fig. 8은 에이전트를 통해 저장된 파일을 스트림 형식의 데이터로 변환하여 매니저의 프레임 워크에 전달하는 알고리즘이다. 웹 데이터 수집 서버의 데이터 전송 프로그램은 파일에 기술된 센서 데이터를 일정한 간격으로 의사 스트리밍 데이터를 매니저의 프레임워크에 보내진

다. 매니저는 처리 데이터량의 증가에 대응하기 위해 수집 서버로부터 수신한 데이터를 큐에 저장한다. 웹 데이터 수집은 주기적으로 수행하며 일련의 수집 작업이 완료되면 저장된 상품 이벤트를 모니터 시스템에 전달한다.

```
procedure KafkaProducer
 object.file<-paths
 object.message<-*.json
 SEND_Callback =new ContCallback()
 Sand_Kafka_Broker(object)
end procedure

Sand_Kafka_Broker(object)
     mq <- object
      while 1:
       if mq.file :
         KafkaProducer message= new KafkaProducer()
          SEND_Callback.send(message,object);
           end if
 end:
```

Fig. 8. Messages to the R-WCMS Manager

3.2 실시간 웹 데이터 모니터 시스템

R-WCMS 매니저는 실시간 웹 데이터 모니터 시스템의 병렬 데이터 저장 프레임 워크이다.

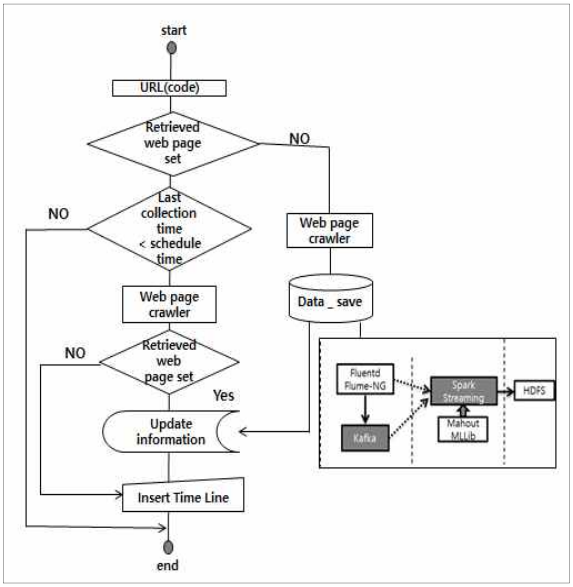

Fig. 9. Architecture of R-WCMS Manager

Fig. 9는 실시간 웹 데이터 수집 모니터 시스템의 흐름도를 표현 한 것이다. 에이전트의 메시지로 저장된 상품 데이터와 전송받은 상품 이벤트를 연속 범위 질의 처리를 통해서 검색한다. 연속 범위 질의는 실시간으로 변경된 데이터를 저장된 영역에 대한 데이터와 비교한 연산의 결과를 모니터 시스템을 통해 업데이트 된다. 기술된 모든 절차는 Spark Streaming을 기반으로 설계되고 실행되었으며 연속 범위질의 목록은 Spark의 RDD로 생성되어 인-메모리에서 관리 한다. 상품 이벤트로 전송된 메시지들을 1초 미만 수집하여 처리 시간 마다 RDD로 생성한다. 생성된 RDD는 그리드 색인을 이용하여 미리 저장된 색인을 기반으로 분류된 영역에 연속 범위 질의를 수행한다. RDD의 고유 셀 번호는 상품 이벤트의 상품코드를 색인화 하고 그룹화 한다. 각 그룹의 색인 코드를 이용하여 그리드 색인을 사용한 하위 데이터 셀을 만든다. 하위 데이터 셀은 연속범위 질의를 통해 업데이트 된 상품의 세부정보를 빠르게 탐색 한다.

```
procedure ECD_Message
  block_list   w_info(web page)
  for each block in block_list
     if  ( block ==null ) then
        Create_MQ(object)
    else
        Update_MQ(MQ,object)
    end if
 end for
 end procedure
 Create_MQ(object)
         old_MQ=objec.MQ
         if(old_MQ   ==null )
                old_MQ =object
         else
             new_MQ =new MQ(object)
          new_MQ.detali_Obj += object
          end if
 Udpate_MQ(MQ,object)
         for each   MQ.code in MQ_Gcode
          for each   MQ.detali_Obj in MQ.code
         Check_MQUpdate(MQ.detali_Obj,object)
                end for
         end for
```

Fig. 10. Find messages in the R-WCMS

Fig.10 은 실시간 발생되는 데이터의 세부 정보를 관

리하는 알고리즘이다. 수집한 웹 데이트의 존재 여부를 확인하고 업데이트된 데이터를 이벤트 메시지로 모니터 시스템에 전송한다.

Table 1. The Performance data

Prameter	value
Message	Code : Product name
	C_Time : Unix Time spent crawling data web
	Teminal ID
	senser data
Message size	Message size (KB)
Message Type	JSON
Message example	{ code : product name, ctime : 1457056210797, url : seed Url value, data : 1536550395.685611 }

Table 1은 상품 이벤트의 데이터의 메시지 파라미터이다. 수집된 데이터는 상품정보를 카테고리 별로 분류하여 스트림 데이터로 변환하면 스트림을 30초 단위로 일괄 처리 메시지 이벤트를 구현한다.

```
procedure  BashStream
    spark <-SparkSession.builder.getOrCreate()
    sc  <- spark . sparkContext
    sqlContext <- SQLContext ( sc )
    ssc  =  StreamingContext( sc ,  30 )
    def  my_utf8_decoder(s) :
        try  :
            if  s  is  None :
                return  "{}"
            return  s.decode ( 'utf-8' )
        except  UnicodeDecodeError  :
            return  "{}"
   kafkaStream  <- KafkaUtils . createDirectStream
   lines <- kafkaStream.map(lambda  x :  x [ 1 ])
    lines . foreachRDD ( process )
   ssc.start()
end procedure
```

Fig.11. Real-time behavioral model

Fig. 11은 업데이트된 상품을 수집하는 시간 예측 알고리즘이다. Kafka에서 데이터를 읽기 학습한 행동 유형 모델을 이용하여 상품 이벤트 메시지 주기로 판별했다. 발생된 상품 이벤트 메시지 주기는 제안된 모델의 모니터 시스템으로 업데이트된 웹 데이터의 수집 시간을 예측 할 수 있다. 결과를 볼 수 있도록 상품 관리, 상품 정보, 주문 정보 등을 제공한다. 업데이트 예측 시간을 측정하여 알림을 통한 데이터 수집의 시간적인 변화에 대처하는 시스템으로 구현했다.

$$S(\%) = C_t \left| \frac{1}{\sum_{n \leftarrow 1}^{m} *(C_d * D_i)} \right| \qquad (1)$$

Table. 2 Equation parameter

Parameter	Description
$S(\%)$	Success Rate
$\sum_{n \leftarrow 1}^{m}$	Number of attempts
C_d	Number of Collect day
D_i	Number of Document
C_t	Number of gather

계산하는 방법은 식(1)의 수식과 같다. 모니터 시스템을 사용하여 저장된 상품 목록의 코드를 100개의 사이트를 10일 동안 20회씩 웹 수집 결과를 업데이트된 데이터의 URL 데이터 수집 횟수의 시간당 평균을 구한 후 통계를 산출했다. 상품 정보로는 신규 상품, 단종 상품, 품절 상품, 상품 가격, 상품의 상세 정보와 변경 수량 등을 제공한다. 주문 정보로는 신규 주문, 교환요청, 반품요청, 배송정보 변경 등을 제공 한다. 모니터 시스템은 상품 판매 사이트의 갱신된 상품 정보로 비즈니스에 활용할 수 있도록 수집된 각종 정보를 제공한다.

4. 실험 및 평가

실험은 두 가지 방법으로 진행했다. 첫 번째는 웹 데이터 수집 과정 실험이다. 실험 결과, 한정된 시간동안 검색에 대한 수집내용의 정확성을 확인했다. 두 번째는 본 논문에서 제안하는 알고리즘에 대한 평가를 위한 실험이다. 업데이트 된 웹 데이터의 시간을 예측했다.

4.1 실험 환경

제안한 기법의 구현 환경은 CPU 2(4vCore), 메모리 8GB의 Red Hat Enterprise Linux 7.2의 플랫폼을 사용했다.

Table 3. The OSS Ver. and OSS List

Oss Version (node count)	Discribsion
Spark 1.6. 0 (1)	parallel processing framework
Spark Streaming 1.6.0 (5)	stream data processing spark
MLlib 1.6.0 (1)	Machine Learning Spark Library
YARN 2.6.3	Resource management in Spark cluster configuration
HDFS 2.6.3 (master 1/slave 3)	Distributed File System (YARN prerequisite software)
Kafka 0.9.0.0(2)	Data queue
Elasticsearch 1.7.5 (5)	Store and retrieve data
Kibana 4.1.5(1)	Data Visualization

스트림 데이터 실시간 예측 기법을 평가하기 위한 실험환경은 Table 3과 같다. 색인연속 범위질의, 발생되는 이벤트 메시지, 시간 예측 모델링은 모두 Spark Streaming과 Kafka로 설계 했다.

4.2 실험 방법

먼저 제안 모델을 사용하여 상품 코드에 대한 데이터를 가전제품과 생활용품 판매 사이트 100개를 기준으로 5년 동안 분기별로 나누어 데이터를 병렬 수집했다.

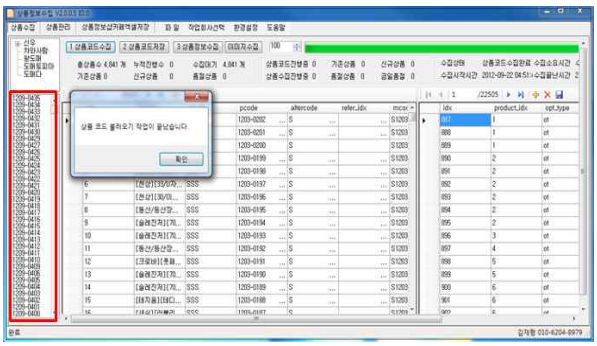

Fig. 12. Code List Web Crawling

Fig.12은 수집된 데이터의 색인된 상품코드로 카테고리별 목록과 세부정보 및 수집된 시간을 확인했다. 수집된 데이터는 웹 수집 모니터 시스템에 스트림으로 전달된다.

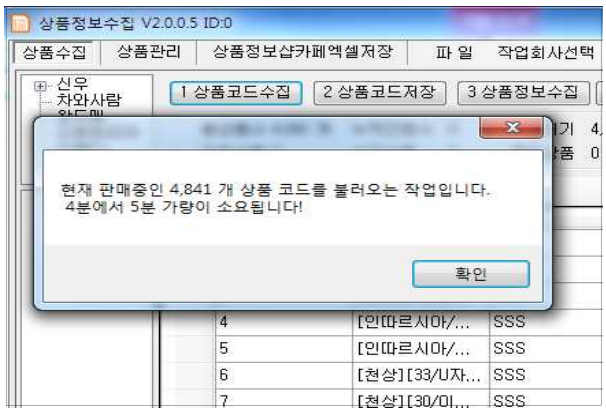

Fig. 13. Estimate update time of R-WCMS

Fig. 13 에서와 같이 적용된 모델은 병렬로 웹 데이터를 수집한 시간과 실시간 업데이트된 자료를 비교한다. 실험은 실시간 웹 데이터 수집 모니터 시스템 미적용 모델과 적용 모델에 대한 편차로 자료 수집의 정확도 및 시간 성능을 평가했다.

4.3 결과 및 분석

URL 수집에 대한 상품 코드 경우 Fig. 14에서 보는 바와 같이 제안모델의 적용과 미적용에 대한 데이터 수집 시간은 크게 차이를 보이지 않았다. 이는 데이터 수집 방법이 URL수집으로 방법이 동일하게 상품 코드로 색인되어 수집하기 때문이다.

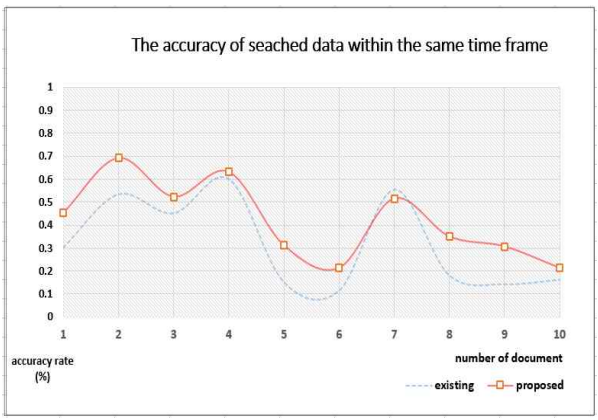

Fig. 14. The accuracy of seached data within the same time frame

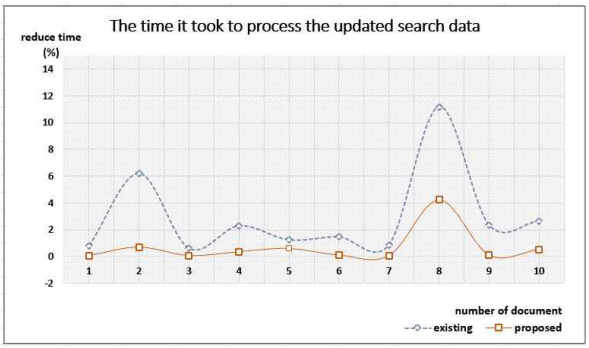

Fig. 15. The time it took to process the updated search data

하지만 모니터 시스템을 통하여 저장된 데이터를 분석 모델링 한 경우에는 Fig.15 에서 보는 바와 같이 데이터 저장 속도와 실시간 분석 시간이 제안 모델 적용인 경우 미적용 모델 보다 15~17% 감소되었다. 이는 비정형 데이터를 이용한 데이터 저장 시간과 업데이트 될 예측 시간 측정이 수집될 시간을 모니터 해주기 때문이다 특히, 실시간 업데이트 될 데이터를 분석 예측하는 알고리즘을 통해 웹 데이터 수집의 횟수를 줄여 주기 때문이다. 제안 모델의 경우 웹 데이터 수집의 시간이 감소한 것은 모니터 시스템으로부터 웹 사이트의 업데이트 할 예측 정보를 받아 무한적인 주기적 탐색시간을 줄일 수 있기 때문이다.

5. 결론

본 논문에서는 실시간으로 변화하는 웹 사이트의 데이터를 병렬 수집하고 시스템에 적재한 후 업데이트 되는 데이터의 수집 시간을 예측하는 방법에 대해서 제안했다. 제안하는 모델은 실시간으로 업데이트 되는 검색 자료를 병렬적으로 수집하여 시스템에 적재하고 수집될 데이터의 시간도 예측하는 방법을 제안했다. 또한 검색된 데이터의 목록과 세부 정보까지 검색하고 수집할 데이터의 개수까지 검출했다.

실험 결과를 통해 30만개의 이벤트가 발생되었고 4,841개의 이벤트가 4분에서 5분 사이에 실시간 발생되는 예측결과를 확인했다. 본 논문에서 제안하는 색인연속 범위질의, 발생되는 이벤트 메시지, 시간 예측 모델링은 모두 Spark Streaming과 Kafka로 설계했다. 실험 결과 웹 수집은 15~ 17%의 시간이 감소했다. 향후 연구에서는 실시간 발생 되는 예측 결과를 초 단위로 분할하고 데이터가 기하급수적으로 늘어나게 되면 적재될 시스템 확장에 대한 연구가 요구된다.

REFERENCES

[1] K. Y. Kim, W. Lee, M. H. Lee, H.M.Yoon & S. H. Shin(2011). *Development of Web Crawler for Archiving Web Resources, International Journal of contents, 11(9),* 9-16.
DOI : 10.5392/JKDA

[2] J. h. Cho & H. Garcia-Molina. (2009), *Parallel crawlers , Proceedings of the 11th international conference on World Wide Web. Honolulu, Hawaii, USA:ACM.* pp.(124 - 135).
DOI :10.1145/511446.511464.ISBN

[3] H. J. Kim, J. Y Lee & S. S Shin. (2017), Multi-threaded Web Crawling Design using Queues. *Journal of Convergence for Information Technology, 7(2),* 43-51.

[4] H. J. Mun. (2015). Polling Method based on Weight Table for Efficient Monitoring. *Journal of Convergence for Information Technology, 5(4),* 5-10.

[5] Olston, Christopher. et al. (2010). *Foundations and Trends® in Information Retrieval, 4(3),* 17.
DOI : 10.1561/1500000017

[6] Y. S. Jeong. (2015). Business Process Model for Efficient SMB using Big Data. *Journal of Convergence for Information Technology, 5(4),* 11-16.

[7] J. H. Ku. (2018). A Study on Adaptive Learning Model for Performance Improvement of Stream Analytics. *Journal of Convergence for Information Technology, 8(1),* 201-206.

[8] G. Pant & F. Menczer. (2002). MySpiders: Evolve your own intelligent Web crawlers. *Autonomous Agents and Multi-Agent Systems 5(2),* 221 - 229.

[9] E. J. Shin, Y. R. Kim, H. S. Heo & K. Y. Whang. (2008). Implementation of a Parallel Web Crawler for the Odysseus Large-Scale Search Engine. *Journal of Computing Science and Engineering, 14(6),* 567-581.

[10] M. Zaharia, M. Chowdhury, M. J. Franklin. (2010). *Scott Shenker, and Ion Stoica, Spark: Cluster Computing with Working Set. Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 10(10-10),* 95.

[11] Kafka. https://kafka.apache.org/intro

김 영 아 (Kim, Yeong A) [학생회원]

· 2018년 12월 : 경남과학기술 대학교 컴퓨터공학과(석사과정)
· 2018년 현재 : ㈜ En-Core HRD 연구원
· 관심분야 : 빅 데이터, 정보 보안, 네트워크, ICT, IoT, 데이터 마이닝
· E-Mail : finish07@gntech.ac.kr

김 계 희(Kim, Gea Hee) [정회원]

· 2013년 8월 : 한국방송통신대학교 정보과학과(이학석사)
· 2017년 2월 : 경남과학기술대학교 컴퓨터공학과(공학박사)
· 2018년 현재 : ㈜ En-Core HRD 연구원
· 관심분야 : Vanet, IoT, 무선네트워크, 차량 간통신
· E-Mail : jenni7@gntech.ac.kr

김 현 주(Hyun Ju Kim) [정회원]

· 1988년 2월 : 경상대학교 컴퓨터과학(이학사)
· 1990년 8월 : 숭실대학교 컴퓨터공학(공학석사)
· 2000년 8월 : 경상대학교 컴퓨터과학(이학박사)
· 2002년 ~ 현재 : 경남과학기술대학교 컴퓨터공학과 교수
· 관심분야 : XML, 정보검색, 데이터마이닝
· E-Mail : khj@gntech.ac.kr

김 창 근(Kim, Chang Geun) [정회원]

·1990년 2월 : 경남대학교 컴퓨터공학과(공학석사)
·1999년 2월 : 경남대학교 컴퓨터공학과(공학박사)
·1995년 3월~현재 : 경남과학기술대학교 교수
· 관심분야 : 컴퓨터네트워크, 데이터통신, e-비즈니스
· E-Mail : cgkim@gntech.ac.kr