클라우드 환경에서 대규모 콘텐츠를 위한 효율적인 자원처리 기법†

(ECPS: Efficient Cloud Processing Scheme for Massive Contents)

나문성*, 김승훈**, 이재동***
(Moon-Sung Na, Seung-Hoon Kim, and Jae-Dong Lee)

요 약 주요 IT 벤더들은 클라우드 컴퓨팅 기술을 이용하여 설치과정 생략, 운용비용 절감, 서비스품질 등에 중점을 두어 대규모 콘텐츠 서비스를 제공하고 있다. 반면에, 대규모 콘텐츠 데이터의 가공, 분석을 수행하는 데이터 처리 프로세스는 처리 시간의 단축을 위한 방법론이 요구되고 있다. 이에 본 논문에서는 클라우드 환경에서 대규모 콘텐츠를 위한 효율적인 자원처리 기법(Efficient_Cloud_Processing_Scheme : ECPS)을 제안한다. 제안한 기법은 리소스 확장 방안을 CPU 및 스토리지 등의 인프라스트럭처 단계에서 설계한다. 대규모 콘텐츠에 대한 자원 할당 방안을 Hadoop 플랫폼 기반의 MapReduce 프로그래밍 기법과 데이터마이닝 분야에서 숨겨진 패턴을 탐지하는데 사용되는 연관규칙을 이용하여 제시한다. 기존 설정값으로 자원을 할당하여 운용되는 환경과 비교하여 ECPS기법을 적용한 결과, 제안 기법이 20% 이상의 성능 및 속도가 향상되었음을 확인하였다.

핵심주제어 : 클라우드 컴퓨팅, 자원분배, 대규모콘텐츠

Abstract Major IT vendors expect that cloud computing technology makes it possible to reduce the contents service cycle, speed up application deployment and skip the installation process, reducing operational costs, proactive management etc. However, cloud computing environment for massive content service solutions requires high-performance data processing to reduce the time of data processing and analysis. In this study, Efficient_Cloud_Processing_Scheme(ECPS) is proposed for allocation of resources for massive content services. For high-performance services, optimized resource allocation plan is presented using MapReduce programming techniques and association rules that is used to detect hidden patterns in data mining, based on levels of Hadoop platform(Infrastructure as a service). The proposed ECPS has brought more than 20% improvement in performance and speed compared to the traditional methods.

Key Words : Cloud Computing, Resource Allocation, Massive Contents

1. 서 론

인터넷 환경의 일반화 및 이에 따른 사용자의 대규모 콘텐츠 요구 증가에 따라 컴퓨팅 환경의 비용 절감과 IT 인프라의 효율적인 활용을 위한 방안으로 클라우드 컴퓨팅이 활성화되고 있다. Amazon, Google 등 글로벌 IT 벤더들의 콘텐츠 서비스는 영화, 음악

† 본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2010년도콘텐츠 산업기술지원사업의 연구결과로 수행되었음
* 단국대학교 정보컴퓨터과학과, 제1저자
** 단국대학교 멀티미디어공학과, 교신저자
*** 단국대학교 컴퓨터학부, 제2저자

등의 디지털 콘텐츠를 중심으로 대규모화되어 가는 추세이다. 이러한 대규모 콘텐츠 서비스의 해결과제인 접속 속도, 서비스 품질 등의 차세대 디지털 콘텐츠 서비스의 핵심 대안으로 클라우드 컴퓨팅 기술이 부각되고 있다. 대규모 콘텐츠 서비스는 현재의 낮은 CPU 활용률, 스토리지 사용률과 함께 복잡한 인스톨 과정, 높은 설비투자비용, 지속적인 사후 관리 등의 문제점이 있다. 이에 주요 IT 벤더들은 클라우드 컴퓨팅 기술을 이용하여 설치 과정 생략, 운용비용 절감, 사전 관리 등에 중점을 두어 대규모 콘텐츠 서비스에 대한 서비스 사이클 단기화, 실시간 피드백, 어플리케이션 배포 속도 증가 등의 효과를 기대하고 있다[1].

본 논문에서는 클라우드 컴퓨팅 환경에서 플랫폼 간 상호호환성 있는 콘텐츠 관리 기법과 분산병렬 처리기법을 이용한 대규모 콘텐츠 서비스 기법을 제안한다. 이를 위해, 리소스 확장 방안을 CPU, 네트워킹 및 스토리지 등의 인프라스트럭쳐 단계의 Hadoop 플랫폼 기반으로 대규모 콘텐츠에 대한 고성능 서비스를 위한 자원 할당 방안을 MapReduce 프로그래밍 기법과 데이터마이닝 분야에서 숨겨진 패턴을 탐지하는데 사용되는 연관규칙을 이용하여 고성능 콘텐츠 처리를 위한 자원 할당 방안을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대하여 살펴보고, 3장에서는 본 논문에서 제안한 자원분배기법에 적용하기 위한 클라우드 시스템 설계방안을 제시하고, 4장에서는 효율적 자원 할당을 위한 Efficient_Cloud_ Processing_Scheme(ECPS)을 설계 및 구현한다. 5장에서는 실험을 통하여 제안한 방법의 효율성을 증명한다. 마지막으로 6장에서는 본 연구의 결론과 향후 연구계획에 대하여 기술한다.

2. 관련 연구

2.1 클라우드 컴퓨팅

클라우드 컴퓨팅은 각 기업이 애플리케이션을 개발하거나 서비스할 때 서버나 스토리지 등 컴퓨팅 자원 등을 자체적으로 보유하기보다 각 자원을 보유하고 있는 클라우드 컴퓨팅 플랫폼 제공자를 통해 운용하는 것을 의미한다. 클라우드 컴퓨팅은 인터넷을 통하여 제공되는 어플리케이션들과 이 서비스들을 제공하는 데이터 센터 내의 모든 하드웨어와 소프트웨어를 포함한다[2].

2.2 대규모 콘텐츠 기술

대규모 콘텐츠 서비스는 검색, 업로드와 같이 간단한 트랜잭션 처리와 소량의 데이터 접근이 필요한 온라인 서비스 분야와 키워드 추출, 로그 분석 등과 같이 대량의 데이터 접근이 필요한 배치 서비스 분야가 있다. 기존 데이터베이스 기술은 인터넷 서비스 데이터 규모 및 동시 사용자 수에 대한 효율성 문제로 인해 비용 문제나 성능 문제를 야기한다. 또한 인터넷 서비스 응용에서 필요로 하는 기능보다 너무 많은 기능을 제공하거나, 응용에 맞게 최적화 할 수 있는 유연성이 부족하다.

2.2.1 대규모 콘텐츠 저장 관리 기술

대규모 콘텐츠 저장 및 관리를 위한 분산 클러스터 저장 시스템은 네트워크상에 분산된 대량의 서버들을 클러스터로 구성함으로써 대용량의 저장 공간과 빠른 입출력 성능을 제공할 수 있어야 한다. 또한 시스템 확장이 용이하며, 서버 고장과 같은 시스템 장애가 발생하더라도 계속해서 안전하게 서비스를 제공할 수 있는 신뢰성과 가용성을 보장하여야 한다. 따라서 대규모 클러스터 시스템 플랫폼의 근간이 되는 파일 시스템은 위의 요구 사항을 최대한 만족할 수 있어야 한다[3]. Hadoop은 Apache Lucene 프로젝트의 일부분으로 진행되고 있는 프로젝트로 Hadoop 분산 파일 시스템(HDFS)과 MapReduce 구현 등을 포함한다[4].

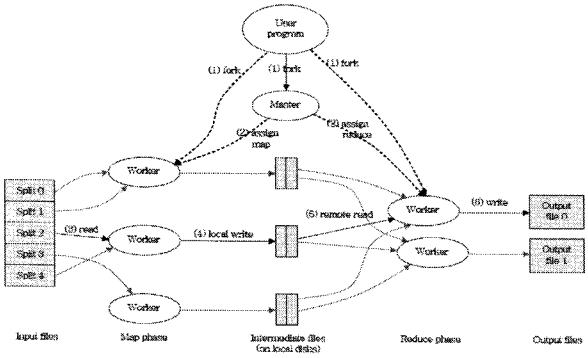
2.2.2 대규모 콘텐츠 처리 기술

대규모 콘텐츠 서비스를 위한 분산 컴퓨팅 플랫폼은 분산 환경에 대한 모니터링 및 스케줄링을 제공하는 클러스터 관리 시스템의 지원 하에 대규모 데이터 저장을 위한 분산 파일 시스템 기술, 대규모 데이터 검색 및 변경을 지원하는 분산 데이터 저장 관리 기술, 대규모 데이터 분석 처리를 지원하기 위한 분산 병렬 처리 및 분석 기술 등이 통합되어 활용되고 있다. 분산 컴퓨팅 플랫폼 기술은 가상화 기술과 함께 클라우드 컴퓨팅 기술의 핵심이 되고 있다[5].

데이터 가공, 분석을 수행하는 배치성의 데이터 처리 업무는 인터넷 데이터의 규모가 지속적으로 증가

함에 따라 처리 시간이 수 일 혹은 수십 일이 소요되므로 이의 단축이 필요하다. 이를 위해 각 응용에서는 독자적으로 병렬 분산 처리를 활용하여 개발해 오다가 이를 공통 플랫폼으로 제공하기 시작했다. 구글에서 2004년 MapReduce 병렬처리 시스템을 발표 후, 이를 기반으로 한 기술들이 야후, Hadoop 등에서 개발되어, 현재는 대규모 데이터 처리 분야에서는 MapReduce 병렬 처리 모델이 표준이 되고 있다[6]. MapReduce는 인터넷 서비스를 위한 배치 업무들을 빠르게 수행하기 위해 제안된 병렬 처리 모델로 구글의 다양한 서비스에 적용되고 있다. 대규모 데이터 병렬 처리를 위해서는 데이터 증가에 따른 효율성을 제공할 수 있어야 하며, 노드 간 데이터 이동에 따른 네트워크 트래픽을 최소화 할 수 있도록 업무 분산이 필요하다[6].

MapReduce는 이와 같은 요구사항을 고려하여 만들어진 시스템이다. MapReduce는 (key, value) 기반의 데이터를 병렬 처리하는 모델로, <그림 1>과 같이 입력데이터 소스를 기반으로 Map 태스크를 수행하여 중간 결과를 생성하고 이를 입력으로 Reduce 태스크를 수행하여 최종 결과를 산출하는 2단계로 구성된다.



<그림 1> MapReduce 실행 모델

2.2.3 대규모 콘텐츠 관리를 위한 연관 규칙 기법

데이터 마이닝 분야에서 대량의 데이터에서 빠르게 유용한 정보를 획득하고 지식을 창출하는 여러 가지 기법이 연구되었다. 그 중 대표적인 연관규칙은 방대한 데이터 안에 존재하는 유용한 패턴을 찾아내는 기법이다. 대표적인 연관규칙 추출 기법으로는 이진 연관규칙에 대한 빈발항목집합을 찾아내는 Apriori 알고리즘이 있다. Apriori 알고리즘은 빈발항목집합의 공집합이 아닌 모든 부분집합은 반드시 빈발하다는 법칙을 이용하여 연관규칙을 추출해 내는 기법이다. 하지만 빈발항목집합을 찾아내기 위하여 트랜잭션 내에 존재하는 항목의 후보 집합을 매번 생성해야 하기 때문에 관련 데이터를 여러 번 스캔해야 한다는 단점이 있다. 이러한 문제점을 해결하기 위하여 후보생성 없이 분할-정복(Divide-and-Conquer) 기법을 사용하여 완전한 빈발항목집합을 발견하는 기법이 고안되었는데 이것이 빈발 패턴 증가(Frequent Pattern Growth) 알고리즘이다[7].

3. 대규모 콘텐츠 서비스를 위한 효율성 있는 클라우드 시스템 설계

본 장에서는 대규모 콘텐츠 서비스를 위한 가변적인 효율성 지원을 위한 클라우드 시스템의 기능 요건을 정의하고, 해당 요건을 충족하는 기능구조(Functional Architecture)를 설계하며, 시스템 아키텍처를 설계한다.

3.1 효율성 있는 시스템 설계를 위한 기능 요건

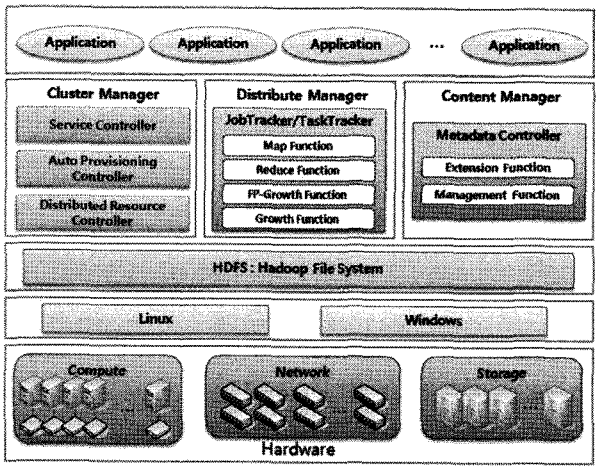
대규모 콘텐츠를 위한 클라우드 컴퓨팅 기반 서비스는 분산 파일 시스템의 데이터 입출력과 처리에 대해 효율적인 성능을 제공해야 한다. 이를 위해 대용량 콘텐츠의 입출력 성능, 파일 시스템 상에서 데이터의 적절한 배치, 효율적인 캐시 사용 등이 고려되어야 하고, 시스템에 저장된 데이터의 데이터 오류 감지 및 복구, 데이터 중복 제거 기능과 최적의 저장 공간을 확보할 수 있어야 한다. 또한, 클라우드 컴퓨팅 시스템은 대규모 컴퓨팅 환경에서 발생하는 비용적인 측면에서의 효율성, 지속적으로 증가하는 데이터의 수용, 빈번하게 발생하는 고장에 대한 대처, 관리의 편리성과 요건들을 충족하여야 한다[3][8][9].

3.2 클라우드 시스템 아키텍처

앞 절에서 제시한 기본요건 및 기능구조를 만족하는 효율성 있는 클라우드 컴퓨팅 시스템의 모듈 구성

도는 <그림 2>와 같다. 가상화 기능의 기본적인 요소로 Hadoop 기반의 스토리지 가상화 기술을 적용한다.

Cluster Manager는 Hadoop 기반의 가상화 구조에서 대규모 콘텐츠를 효율적으로 서비스하기 위한 세 개의 Controller로 구성된다. Service Controller는 클라우드 시스템의 효율성을 확보하기 위하여 서비스를 동적으로 관리하고, Auto Provisioning Controller는 서비스를 자동 배치하며, Distributed Resource Controller는 분산된 스토리지 자원을 자동 관리한다.

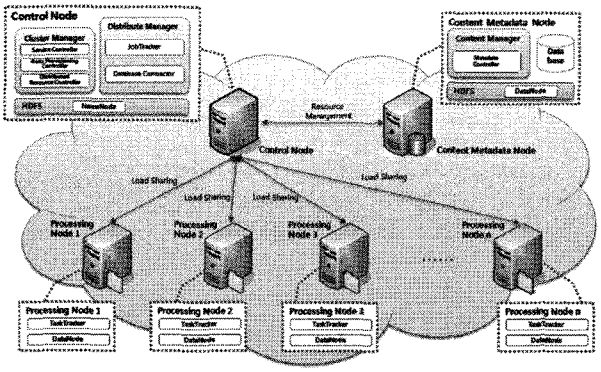


<그림 2> 클라우드 시스템 모듈 구성도

Distributer Manager는 Hadoop의 분산병렬 프로그래밍 모델인 MapReduce 프레임워크를 활용하여 효율적인 자원할당 연산을 수행한다. MapReduce 프레임워크에서는 JobTracker와 TaskTracker를 이용하여 작업을 분배하고 수행한다. 이를 위해 기본적으로 제공되는 Map Function과 Reduce Function 외에 자원 이용률과 시스템 성능을 기반으로 적절한 자원을 할당하는 Growth Function, FP-Growth Function을 포함한다.

Content Metadata Manager는 Metadata Controller를 통하여 UCI 식별체계를 이용한 메타데이터 확장(Extension Function) 및 관리(Management Function)를 수행한다. Extension Function에서는 UCI 식별 메타데이터를 총괄기관에 전송하며, 콘텐츠 전송을 요청하는 기능을 추가로 제공한다. Management Function에서는 UCI를 이용하여 콘텐츠를 저장하고 관리하는 기능을 제공한다.

이제 본 논문에서 제안한 목표 시스템의 기능들을 처리하는 대규모 콘텐츠 서비스를 위한 클라우드 시스템 구성도는 <그림 3>과 같다. 대규모 콘텐츠 서비스를 위한 클라우드 시스템은 크게 Control Node, Content Metadata Node 및 Processing Node들로 구성된다. Control Node는 전체 클라우드 컴퓨팅 시스템을 관리하고 Processing Node로 자원을 할당하기 위한 Node로 목표 시스템상의 Cluster Manager와 Distribute Manager로 구성되며, Distribute Manager를 통해 Processing Node에게 자원할당 연산을 수행한다. 또한, Content Metadata Node와의 연결을 통해 UCI 식별체계를 활용한 콘텐츠 관리 기능을 제공받게 된다. Content Metadata Node는 UCI 식별체계를 이용하여 대규모 콘텐츠를 관리하기 위한 Content Manager를 포함한다. 또한, Processing Node는 콘텐츠 저장을 위한 DataNode 및 고성능 연산을 위한 TaskTracker 역할을 수행한다.



<그림 3> 대규모 콘텐츠 서비스를 위한 클라우드 시스템 구성도

4. 효율적인 자원 할당을 위한 Efficient_Cloud_Processing_Scheme(ECPS)

대규모 콘텐츠 서비스에서 콘텐츠의 종류와 사용 목적에 따라 자원을 재구성하고 효율적으로 제공하기 위한 자원 할당을 위해 본 논문에서는 각 노드의 현재 자원 이용률과 시스템 성능을 기반으로 적절한 자원을 할당하는 Efficient_Cloud_Processing_Scheme (ECPS) 기법을 제안한다. ECPS 기법에서는 클라우드 시스템을 구성하는 노드들에 대하여 연관규칙을 추출하기 위하여 FP-Growth 알고리즘을 활용하고 계산의

효율성을 위하여 클라우드 컴퓨팅 환경에서의 분산병렬처리 계산 방법인 MapReduce프로그래밍 기법을 이용한다.

4.1 자료 수집 단계 및 전처리 단계

클라우드 컴퓨팅 환경의 각 노드의 이용형태 분석을 위해 시스템 사양, 리소스 사용량, 작업처리능력 등의 정보를 일정 시간 간격으로 수집하여 로그 형태로 저장한다. 수집된 데이터를 기반으로 적합한 노드를 선택하여 사용자 요청 시 자원을 할당함으로써 클라우드 시스템은 적절한 자원을 사용자에게 제공할 수 있다. 본 연구에서는 각 자원의 시간대별 이용 형태 분석 따른 자원의 안정성과 성능을 예측 하는 것에 초점을 둔다. 이러한 예측에 의해 동적으로 서비스를 구성할 수 있으며 효율적인 작업의 분산 처리가 가능하다.

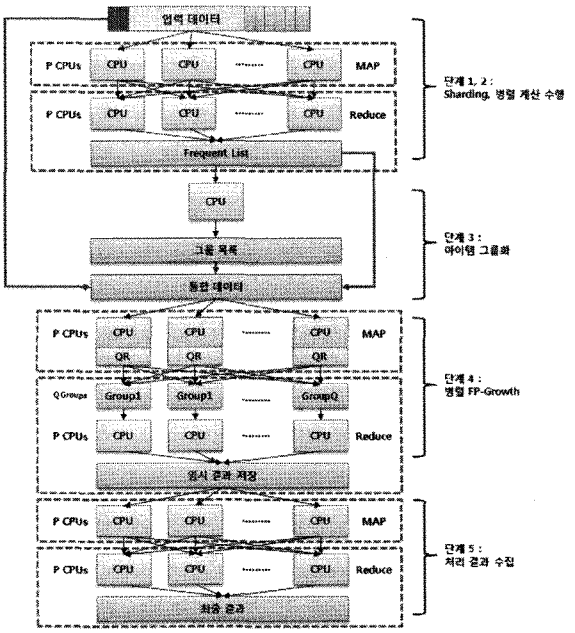
본 논문에서는 노드의 기본 정보, 노드의 평균 리소스 사용률, 노드의 평균 작업 수행시간, 노드의 평균 작업 성공률 등을 사용한다. 이를 위하여 Hadoop File System에서 제공하는 노드별 클러스터 정보와 각 노드의 자원 이용률과 시스템 자원 모니터링 도구를 이용하여 자원 이용률을 수집한다. 각 노드의 자원에 대한 모델링을 표현하면 다음과 같다. NodeRes = <R, T, C, W, U>, T는 시간, R은 자원의 집합, C는 각 자원의 수치, W는 자원이 행한 작업을 나타내며, U는 작업 가능한 자원 이용률 나타낸다.

수집된 정보는 다음 절에서 제안할 알고리즘에 적용될 수 있도록 전처리 과정을 통해 재편집된다. 또한, 처리 중인 콘텐츠의 UCI 메타데이터를 참조하여 콘텐츠의 형태 및 포맷을 고려하여 Hadoop 파일 시스템의 자원 할당 과정에 참조되도록 전처리과정에서 콘텐츠의 UCI 정보를 수집하여 노드별 데이터셋에 포함하도록 하였다.

4.2 ECPS 기법 설계

4.2.1 ECPS 개요

FP-Growth는 Apriori 알고리즘의 단점인 후보 집합을 생성 하지 않는 연관규칙추출 기법이다. 빈발항목을 가지는 데이터를 FP-Tree로 압축하고 분할정복(Divide and conquer)기법을 사용하여 각 항목에 대하여 연관된 트리를 추출한 Conditional Pattern Tree를 생성하고 연관규칙을 추출한다. 본 절에서는 클라우드 컴퓨팅 환경에서 분산 처리를 위해 FP-Growth 알고리즘을 MapReduce 계산과 연동하기 위한 ECPS 기법을 다음 <그림 4>와 같이 설계한다.



<그림 4> ECPS 기법의 단계별 프레임워크

① Step 1 : Sharding : 로그 파일이나 데이터베이스를 분할하여 클라우드 컴퓨팅 환경의 각 시스템 P에 저장한다. 클라우드 컴퓨팅에서 각 시스템에 분산된 데이터를 Sharding이라고 하고, 각 부분을 Shard라고 부른다.

② Step 2 : 병렬 계산 수행 : 각 데이터의 모든 아이템의 지지도를 구하기 위해 MapReduce 계산을 수행한다. 입력된 각 Map은 하나의 Shard이다. 이 단계에서는 아이템 항목인 I를 탐색하고, 이를 F-List(그룹 목록)에 저장한다. 아이템 I는 $\alpha_\xi \in Ti$로 표현하고, Mapper는 <key′=a_ξ, value′=1>로 생성한다. Mapper 처리가 끝나면 key′가 생성된다. key′ 집합은 S(key′)로 표현하고, key와 값을 <key′, S(key′)> 쌍으로 Reducer에 전달된다. Reducer는 처리 후 <key′′=null, value′′=key′+sum(S(key′))>을 출력한다.

③ Step 3 : 아이템 그룹화 : 그룹 Q의 F-List(그룹 목록)에 저장된 아이템 |I|를 분할한다. 이 단계는 각 슬레이브 시스템이 수행한다.

④ Step 4 : 병렬 FP-Growth : 1단계에서 생성된 Shard를 Mapper 계산에 입력하여 그룹에 독립적인 프로세서를 생성한다. 각 아이템 aj∈Ti를 구하기 위해 Group ID에 대응하는 aj를 치환한 후, <key'=gid, value'={Ti[1]…Ti[L]}>의 데이터를 생성한다. Mapper 계산 처리 후, <Key', S(key')> 값을 Reducer에 전달한다. Reducer는 <Key''=null, value''=key'+sum(S(key'))>값을 출력한다.

⑤ Step 5 : 처리 결과 수집 : 4단계에서의 결과를 수집하여 최종 결과로 처리한다.

4.2.2 ECPS 알고리즘

각 단계별 과정에 대한 알고리즘은 다음과 같다.

Input : Data(노드별 데이터셋), r(FP-Tree), a(패턴 정보), ξ(최소지지도 임계치)
Output: 빈발항목 패턴 집합

Procedure: Mapper(key, value=*Ti*)
  Load G-List;
  Generate Hash Table *H* from G-List;
  a[] ← Split(*Ti*);
  for *j* = |*Ti*| - 1 *to* 0 do
    HashNum ← getHashNum(*H*, *a*[*j*]);
    if *HashNum* ≠ *Null* then
      Delete all pairs which hash value is *HashNum* in *H*
      Call *Output*(<*HashNum*; *a*[0] + *a*[1] + ⋯ + *a*[*j*]>);
    end
end

Procedure: Reducer(key=*gid*, value=*DBgid*)
  Load G-List;
  nowGroup ← G-List*gid*
  LocalFPtree ← clear;
  foreach *Ti in DB*(*gid*) do
    Call *insert-build-fp-tree*(*LocalFPtree*, *Ti*);
  end
  foreach *ai in nowGroup* do
    Define and clear a size K max heap : *HP*
    Call *TopK FPGrowth*(*LocalFPtree*, *ai*, *HP*);
    foreach *vi in HP* do
      Call *Output*(<*null*; *vi* + *supp*(*vi*)>);
    end
end

Procedure: FP-Growth(*Data*, *ξ*) ← 1
  Define and clear F-List : *F*[];
    foreach *Transaction* T_i *in DB* do
      foreach *Item* a_j *in* T_i do
        $F[a_i]$++; ← 2
      end
  end
  Sort F[]; ← 3
  Define and clear the root of FP-tree : *r*;
  foreach Transaction T_i in DB do
    Make T_i ordered according to F
    Call *ConstructTree*(T_i, *r*); ← 4
  end
  foreach *item* a_i *in I* do
    Call *Growth*(*r*, a_i, *ξ*);
  end
end

Procedure: Growth(*r*, *a*, ξ)
  if *r contains a single path Z* then ← 5
    foreach *combination(denoted as τ) of the nodes in Z* do
      Generate pattern β=τ ∪ *a* with support = ← 6
      minimum support of nodes in τ;
      if β.*support* >*ξ* then
        Call *Output*(β);
      end
    end
  else
    foreach b_i *in r* do
      Generate pattern β = *bi* ∪ *a* with support = *bi*.support;
      if β.*support* > *ξ* then
        Call *Output*(β);
      end
      Construct β's conditional database ;
      Construct β's conditional FP-tree$Tree_\beta$ ← 7
      if $Tree_\beta \neq \varnothing$ then
        Call *Growth*($Tree_\beta$, *β*, *ξ*); ← 8
      end
    end
end

위 Step 1~Step 2는 FP-Growth 알고리즘이 MapReduce 계산 방법에 적용되기 위한 단계이고, Step 3~Step 4는 MapReduce 계산방법에 의해 FP-Growth 알고리즘이 동작하는 단계이다. Step 1~Step 2는 클라우드 컴퓨팅 환경에서 제안된 알고리즘의 Mapper 함수와 Reducer 함수에 의해 FP-Growth 함수를 호출하여 병렬처리 한다. FP-Growth 알고리즘은 FP-Tree와 패턴 정보 a를 인자로 입력 받는 FP-Growth 함수를 통하여 이루어지고 그 결과로 빈발패턴을 출력한다. Mapper 함수와 Reducer 함수에 의해 호출되는 FP-Growth 함수는 다음과 같이 동작한다. Control Node가 'null'인 FP-Tree는 Item의 이름과 Item의 Node-link 중 첫 번째 Item을 가리키는 링크로 구성된 Header Table을 가지고 있다. 각 노드는 Item의 이름과 출현빈도, 같은 Item을 갖는 다음 노드를 가리키는 링크로 구성된다. FP-Tree 생성을 위해서는 전처리된 로그파일 DB<표 3>와 최소 임계치를 입력한다(1). 전처리된 로그파일 DB 스캔을 통해 빈발항목(Frequent Item)들의 집합 F를 생성한다(2). F를 출현빈도 순의 내림차순으로 정렬하여 빈발항목들의 리스트인 L을 생성한다(3). Control Node가 'null'인 트리 T를 생성한 뒤 전처리된 로그파일 DB의 각 트랜잭션에 ConstructTree 함수를 호출한다(4). Reducer 함수에 의해 호출되는 FP-Growth 함수는 재귀적으로 호출이 되는데, 입력된 트리가 단 하나의 path를 갖고 있는 경우에는 빈발패턴을 생성한 후 종료하게 된다(5). 만약 입력된 트리가 Single Path가 아닌 경우에는 Header Table의 각 item 에 대하여 다음 과정을 수행한다. 패턴 β의 최소 임계치를 적용하여 bi와 입력된 패턴 a의 합집합을 구하여 패턴 β로 정의하고(6), 패턴 β의 Item을 이용하여 Conditional pattern base와 Conditional FP-tree Treeβ를 생성한다(7). 생성된 트리 Treeβ가 공집합이 아니면 Treeβ와 패턴 β를 인자로 다시 FP-Growth 함수를 재귀 호출한다(8).

5. 실험 및 평가

본 장에서는 4장에서 제안한 ECPS 기법을 적용한 자원 확장 성능을 실험을 통해 평가한다.

5.1 대규모 콘텐츠 서비스 지원 효율성 있는 클라우드 시스템 구현

5.1.1 클라우드 시스템 구현 환경

본 논문에서는 클라우드 컴퓨팅 기반의 시스템을 구현하기 위하여 하둡(Hadoop) 플랫폼을 이용한다. 총 10대의 PC를 이용하여 서로 다른 성능의 이질적인 노드로 구성된 클라우드 환경을 구축하기 위하여 각 노드의 성능을 다르게 구성하였다. 각 노드의 운영체제는 Ubuntu Linux를 포팅하고, Hadoop-0.20.2 버전을 설치하였다. 분산 환경을 위하여 Control Node를 Processing Node들과 다른 네트워크에 배치하였다. 각 노드의 운영체제는 Hadoop 플랫폼을 지원하는 Linux와 Windows Server로 구성하였다. Control Node는 작업의 효율성을 높이기 위하여 가장 성능이 좋은 PC를 선정하였으며, 사용자 인터페이스가 보편화된 Windows Server 운영체제를 기반으로 하였다. <표 1>은 실험에 사용된 컴퓨터의 사양이다.

<표 1> 실험을 위한 컴퓨터 현황

CPU	RAM	HDD	OS	수량
Intel Pentium 900MHz	4GB	145GB x 2EA	Windows Server 2008	1
Intel Pentium 900MHz	2GB	160GB	Windows Server 2008	5
Intel Pentium 800MHz	4GB	120GB	Linux Ubuntu 10	1
Intel Pentium 800MHz	2GB	40GB	Linux Ubuntu 10	3

5.1.2 제안된 아키텍처 적용 실험

제안된 자원 확장 방법이 적용된 시스템은 성능에 있어서 Control Node와 Processing Node 간의 비효율적인 메모리, 프로세스, 네트워크 상태에 따라 성능 저하가 발생할 수 있다. 본 실험은 제안된 자원 확장 방법이 적용된 시스템과 단일 처리 시스템과의 성능을 처리시간을 비교하여 성능 결과를 산출한다. 구성된 실험 환경의 10대의 Processing Node와 1대의 Control Node로 설정하고, 처리 작업은 RandomWriter 모듈로 40GB의 입력 데이터를 생성하는 과정을 수행하고, 1,000개에서 10,000개의 처리 작업을 요청하도록 설정하였다. RandomWriter 모듈은 Hadoop 분산 파일 시스템에서 MapReduce를 이용하여 랜덤 데이터를 생성하는 프로그램으로 본 실험에서 제안된 시스템의

처리 성능의 효율성 측정을 위한 기본적인 작업 프로그램으로 선택하였다.

단일 노드에서 처리하는 시간(SNPT: Single Node Processing Time)과 제안된 알고리즘이 적용된 클라우드 컴퓨팅 환경에서의 처리 시간(CPT: Cloud Processing Time)은 본 실험에서 활용된 클라우드 환경 노드에서 처리 작업에만 소요된 시간(PT: Processing Time)의 처리 작업(P)에 대한 합을 계산한 결과로 각각 다음 식 (1), 식 (2)로 표현한다. 또한, 처리 시간과의 관계를 확인하기 위한 비율(PTR : Processing Time Ratio)은 식 (3)과 같이 표현한다.

$$SNPT = \sum_{n=1}^{k} P_{SingleNode} T_n \qquad 식(1)$$

$$CPT = \sum_{n=1}^{k} P_{Cloud} T_n \qquad 식(2)$$

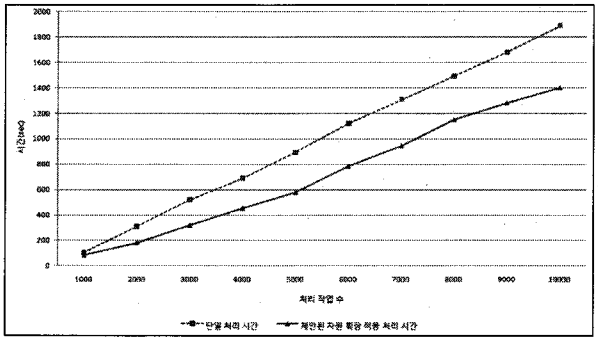
$$PTR(\%) = \frac{SNPT - CPT}{SNPT} \times 100 \qquad 식(3)$$

<표 2>는 1,000개씩 작업이 증가할 때 각 횟수마다 처리 시간의 단축율을 실험한 결과이다. 1,000개의 처리작업에서는 단일노드에서 105.33초의 시간이 소요되었으며, 제안된 알고리즘이 적용된 클라우드 컴퓨팅 환경에서는 83.13초의 시간이 소요되었다. 이 두 결과의 처리시간에 대한 단축율은 21.08%이며, 처리작업을 1,000개에서 10,000개로 증가하면서 나타난 결과의 평균 단축율은 약 30%로 나타났다.

<표 2> 제안된 시스템의 처리 시간 결과

처리 작업 수 (Ea)	1,000	2,000	3,000	4,000	5,000	6,000	7,000	8,000	9,000	10,000
단일 처리 시간 (SNPT, Sec)	105.33	310.25	521.1	689.78	894.39	1122.71	1309.59	1491.81	1681.32	1891.14
제안된 자원 활용 적용 처리 시간 (CPT, Sec)	83.13	181.67	320.34	455.1	581.69	784.91	948.73	1151.45	1284.54	1403.45
처리 시간 단축율 (PTR, %)	21.08	41.44	38.53	34.02	34.97	30.09	27.56	22.82	23.60	25.79

<그림 5>는 본 실험에 대한 측정 결과를 표현한 그래프이다. 클라우드 컴퓨팅 환경에서 제안된 방법을 적용하여 수행한 작업의 수가 증가할수록 적용 효과가 크게 나타나고 있다. 이는 단일 노드 처리 작업의 증가와 네트워크 전송 시간 등의 문제가 발생할 수 있는 상황에서 각 Processing Node의 성능을 최대한 반영할 수 있음을 확인 할 수 있다. 클라우드 컴퓨팅 환경에서 제안된 알고리즘이 적용된 자원 할당 방법에 의한 처리 시간은 클라우드 컴퓨팅의 스토리지 자원을 활용한다는 개념에서 단일노드보다 효율적으로 작업을 처리한다고 볼 수 있다.



<그림 5> 제안된 아키텍처 적용 실험

5.1.3 ECPS 기법 적용 실험

본 실험을 위해 Hadoop에서 기본적으로 제공하는 ExampleRandomWriter 모듈로 400GB(각각 40GB의 파일, 노드 당 10개)의 입력 데이터를 생성하였고, Example Sorter로 정렬을 시도하였다. ECPS 기법 적용을 위해 각 노드의 자원과 Hadoop파일시스템의 버퍼 및 블록 사이즈, 메모리 할당 수치를 변경하여 총 10개의 테스트 노드에 대한 실험 데이터를 구하였다. <표 3>과 같이 각 노드에 대한 CPU, Memory, 네트워크 전송 속도와 Hadoop 파일시스템의 버퍼 및 블록 사이즈, 메모리 할당 관련 옵션을 토대로 ECPS 기법에 적용하였다.

<표 3> 노드별 데이터셋

Node ID	CPU	Memory (GB)	Network (MB)	block size (MB)	memory size	file buffer	Remaining (%)	Blocks	Format
1	800	4	100	128	100	131072	70.21	2	AVI
2	800	2	100	64	100	131072	80.34	26	AVI
3	900	2	100	64	100	131072	69.33	4	AVI
4	800	2	100	128	100	65536	81.21	13	MPEG
5	900	2	100	128	100	131072	71.98	10	MPEG
6	900	2	50	128	100	65536	54.67	13	MPEG
7	800	2	50	128	100	131072	89.12	8	AVI
8	900	2	50	64	100	65536	74.48	12	AVI
9	900	2	100	64	100	131072	61.32	8	AVI
10	900	4	100	64	150	65536	88.19	9	AVI

먼저 디폴트 설정값으로 실행된 Hadoop 파일시스템이 설치된 노드에 테스트 데이터를 생성하여 각 노드의 자원 이용률 중 로딩 프로세스, 메모리 이용률, CPU 이용률, 네트워크 이용률 등을 측정하였다. ExampleRandomWriter 모듈을 10회 실행한 후 노드 Node 1, 2, 3의 자원 이용률 중 전송 시간을 측정하였다. 각 실험에서 측정한 전송 완료 시간은 <표 4>와 같다.

<표 4> 디폴트 설정값으로 측정된 데이터 전송 시간 측정값(단위 : 분)

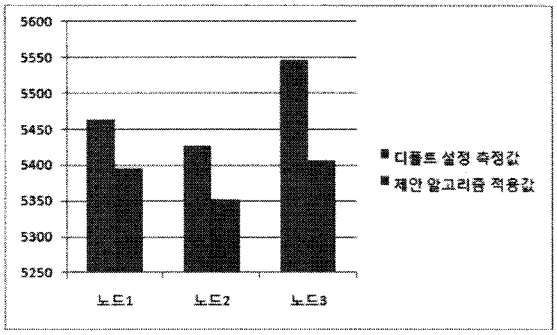
전송 회수	1	2	3	4	5	6	7	8	9	10	평균
Node1	5.417	5.517	5.483	5.450	5.617	5.383	5.467	5.417	5.408	5.475	5.463
Node2	5.400	5.433	5.467	5.450	5.483	5.383	5.350	5.400	5.442	5.458	5.427
Node3	5.750	5.567	5.800	5.783	5.483	5.517	5.267	5.400	5.442	5.458	5.547

ECPS기법 적용을 위해 사용된 데이터는 Hadoop 파일 시스템 정보로 각 노드마다 소유한다. 하나의 가상화 서버 환경에서 디폴트 설정값으로 스토리지 클러스터링을 수행한 결과와 ECPS기법을 적용한 할당 방법을 적용하여 분산 환경에서 스토리지 클러스터링을 수행한 결과를 비교하였다. 각 시스템의 관련 설정값의 최소 지지도는 2%, 맵 함수로 1회에 입력되는 데이터의 수는 10,000개로 하였다. 또한, 데이터의 크기가 늘어날수록 수행시간도 그에 비해서 비례해서 늘어나서 큰 데이터에 대해서도 문제없이 수행이 가능하다. 알고리즘 적용 후 측정한 전송 완료 시간은 <표 5>와 같다.

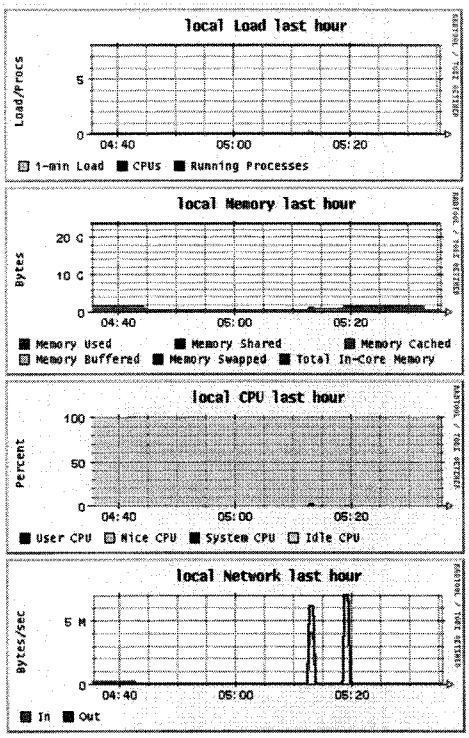
<표 5> 디폴트 설정값으로 측정된 데이터 전송 시간 측정값(단위 : 분)

전송 회수	1	2	3	4	5	6	7	8	9	10	평균
Node1	5,417	5.517	5.317	5.450	5.283	5.383	5.333	5.417	5.358	5.475	5.395
Node2	5,233	5.333	5.467	5.350	5.483	5.217	5.333	5.400	5.408	5.292	5.352
Node3	5,583	5.500	5.633	5.617	5.350	5.350	5.233	5.233	5.275	5.292	5.407

<그림 6>은 디폴트 설정값으로 측정된 결과와 ECPS기법을 통한 설정값으로 측정된 결과를 비교한 그래프이다. 최종적으로 최적화 선택 노드의 리소스 사용량은 평균 20% 정도로 안정되게 유지되고 있었



<그림 6> 디폴트 설정과 제안된 방법이 적용된 설정값의 전송 시간 결과 비교



<그림 7> 제안된 방법을 통한 노드의 성능 측정

고, 리소스 사용률의 변화도 평균 표준편차 이내로 변화도 크지 않음을 알 수 있다. 결과적으로 최근 데이터를 가지고 분석했을 경우 순위가 다소 낮았으나 노드 부하량이 평균적으로 낮고 변화가 적기 때문에 가장 높은 성능을 나타낼 수 있었다. 또한, 제안된 방법을 통한 실험 시 Control Node의 자원 이용에 대한 정보를 Ganglia 모니터링 도구를 통해 살펴보면 <그림 7>과 같으며, 메모리나 CPU 사용이 비교적 안정적으로 동작하였음을 확인할 수 있다.

노드들을 살펴보면 최종 순위 1위인 노드 2는 CPU

4GHz, 메모리 4,096GB, 하드디스크 300GB, 네트워크 카드 100Mbps의 높은 기본정보를 가지고 있었다. 리소스 사용량은 평균 20% 정도로 안정되게 유지되고 있었고, 리소스 사용률의의 변화도 평균 표준편차 이내로 변화도 크지 않음을 알 수 있었다. 결과적으로 노드1은 최근 데이터를 가지고 분석했을 경우 순위가 다소 낮았으나 노드 부하량이 평균적으로 낮고 변화가 적기 때문에 가장 높은 순위를 가질 수 있었다.

6. 결론 및 향후 연구 방향

본 논문에서는 효율성 있는 클라우드 컴퓨팅 환경 구축을 위해 Hadoop 플랫폼 기반으로 대규모 콘텐츠에 대한 고성능 서비스를 위한 자원 할당 방안을 MapReduce 프로그래밍 기법과 데이터마이닝 분야에서 숨겨진 패턴을 탐지하는데 사용되는 연관규칙을 이용하여 고성능 콘텐츠 처리를 위한 자원 할당 방안을 제시하였다.

제안된 방법에 대하여 2가지 성능실험을 수행하였다. 단일 노드에서 데이터를 처리하는 시간과 제안된 클라우드 컴퓨팅에서 처리하는 평균 시간을 비교한 결과 약 30%의 처리시간 단축 효율을 확인 할 수 있었고, 기존 Hadoop 파일 시스템을 이용하여 자원을 할당하고 운영하는 환경과 본 연구에서 제안한 ECPS 기법을 적용한 자원 할당 방법을 비교한 결과에서는 약 20% 이상의 성능 및 속도가 향상되었음을 확인 하였다. 두 가지 실험결과를 통하여 단일 노드에서 데이터를 처리하는 성능에 비해 제안된 ECPS기법을 적용한 자원 할당 방법이 약 44% 이상의 성능이 향상되었음을 확인 할 수 있었다.

따라서 본 연구에서는 대규모 콘텐츠 서비스를 위한 클라우드 컴퓨팅 환경에서 지속적으로 증가하는 데이터의 가공 및 분석 처리 프로세스의 시간이 단축될 수 있음을 알 수 있었다. 제안된 시스템은 대규모 콘텐츠 서비스를 지원하는 산업 및 기업체에 효과적으로 활용 가능하다. 예를 들면, 최근 IPTV 및 인터넷서비스 업체에서 활발하게 진행 중인 VoD(Video on Demand) 서비스에 적용 가능하며, 온라인 게임 서비스 및 애니메이션 렌더링 작업 등 대규모 자원 할당이 필요한 분야에서 효과를 극대화할 수 있다.

향후에는 클라우드 컴퓨팅 환경에서 효율적인 자원 할당에 대해 다양한 자원에 대한 적용과 패턴 인식에 의한 최적의 자원 할당 방안에 대해 연구할 계획이다. 또한, 대규모 콘텐츠를 서비스하는 영화, 게임, 방송 등 산업체에 적용함으로써 실질적인 활용이 가능하도록 개선할 계획이며, 미래 콘텐츠 산업분야에 화두가 되고 있는 3D입체 콘텐츠 분야에도 적용할 예정이다.

참 고 문 헌

[1] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2.

[2] J. Boulon, A. Konwinski, R. Qi, A. Rabkin, E. Yang, and M. Yang. Chukwa: "A Large-scale Monitoring System," In Cloud Computing and Its Applications, Chicago, IL, Oct 2008.

[3] 민영수, 김홍연, 김영균, "클라우드 컴퓨팅을 위한 분산 파일 시스템 기술," 한국정보과학회 학회지, 제5호, 2009.

[4] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. "Scope: Easy and efficient parallel processing of massive data sets," VLDB'08, 2008.

[5] Dhruba Borthakur, "The Hadoop Distributed File System: Architecture and Design," The Apache Software Foundation, 2007.

[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Sixth Symp. on Operating System Design and Implementation, San Francisco, USA, Dec. 2004.

[7] Lamine M. Aouad, Nhien-An Le-Khac, and Tahar M. Kechadi. "Distributed frequent itemsets mining in heterogeneous platforms," Engineering, Computing and Archtecture, 1, 2007.

[8] 이미영, "클라우드 기반 대규모 데이터 처리 및 관리 기술," 전자통신동향분석 제24권 제4호, 한국전자통신연구원, 2009.

[9] Park Yong Kwang, "A Study on Developmental Direction of the Cloud Computing," Master's thesis, HanYang University, 2009.

나 문 성 (Moon-Sung Na)

- 정회원
- 1989년 2월 : 조선대학교 전자공학과 (공학 학사)
- 2003년 8월 : 한양대학교 경영대학원 석사
- 2010년 8월 : 단국대학교 컴퓨터과학 박사 (공학박사)
- 2000.12 ~ 현재 한국콘텐츠진흥원 제작지원본부장
- 2005.03 ~ 2007.02 단국대학교 정보통신대학원 겸임교수

김 승 훈 (Seung-Hoon Kim)

- 정회원
- 1985년 2월 : 인하대학교 전자계산학과(이학 학사)
- 1989년 9월 : 인하대학교 대학원 전자계산학과(이학석사)
- 1998년 2월 : 포항공과대학교 대학원 컴퓨터공학과 (공학박사)
- 1989.08 ~ 1990.12 한국전자통신연구소 연구원
- 1990.12 ~ 1993.01 포스데이타(주)
- 1998.03 ~ 2001.08 상지대학교 조교수
- 2001.09 ~ 현재 단국대학교 교수

이 재 동 (Jae-Dong Lee)

- 정회원
- 1985년 2월 : 인하대학교 전자계산학과(이학 학사)
- 1991년 3월 : Cleveland State Univ. (USA) (MS)
- 1996년 5월 : Kent State Univ. (USA) (Ph.D)
- 2009.07 ~ 현재 한국문화콘텐츠기술학회 학회장
- 2006.04 ~ 현재 국가지정 CT 연구소 소장
- 1987.07 ~ 1988.11 대우중공업 정보관리센타
- 1996.09 ~ 1997.02 (주) 두루넷 기술기획팀 과장
- 1997.03 ~ 현재 단국대학교 컴퓨터학부 교수

논 문 접 수 일 : 2010년 10월 29일
1차수정완료일 : 2010년 11월 17일
게 재 확 정 일 : 2010년 11월 29일