맵리듀스 프레임워크를 이용한 대용량 공간 추론 방식

남상하*, 김인철*
*경기대학교 컴퓨터과학과
e-mail:namsh@kgu.ac.kr, kic@kgu.ac.kr

Large-scale Spatial Reasoning using MapReduce Framework

Sang-Ha Nam*, In-Cheol Kim*
*Department of Computer Science, Kyonggi University

요 약

Jeopardy 퀴즈쇼와 같은 DeepQA 환경에서 인간을 대신해 컴퓨터가 효과적으로 답하기 위해서는 인물, 지리, 사건, 역사 등을 포함하는 광범위한 지식베이스와 이를 토대로 한 빠른 시공간 추론 능력이 필요하다. 본 논문에서는 대표적인 병렬 분산 컴퓨팅 환경인 하둡/맵리듀스 프레임워크를 이용하여 방향 및 위상 관계를 추론하는 효율적인 대용량의 공간 추론 알고리즘을 제시한다. 본 알고리즘에서는 하둡/맵리듀스 프레임워크의 특성을 고려하여 병렬 분산처리의 효과를 높이기 위해, 지식 분할 문제를 맵 단계에서 해결하고, 이것을 토대로 리듀스 단계에서 효과적으로 새로운 공간 지식을 유도하도록 설계하였다. 또한, 본 알고리즘은 초기 공간 지식베이스로부터 새로운 지식을 유도할 수 있는 기능뿐만 아니라 초기 공간 지식베이스의 불일치성도 미연에 감지함으로써 불필요한 지식 유도 작업을 계속하지 않도록 설계하였다. 본 연구에서는 하둡/맵리듀스 프레임워크로 구현한 대용량 공간 추론기와 샘플 공간 지식베이스를 이용하여 성능 분석 실험을 수행하였고, 이를 통해 본 논문에서 제시한 공간 추론 알고리즘과 공간 추론기의 높은 성능을 확인 할 수 있었다.

1. 서론

최근에 IBM의 Watson 시스템이 Jeopardy 퀴즈쇼에서 인간 경쟁자들을 이기고 우승한 사건은 자연어 처리(natural language processing), 질의응답(question answering), 지식 표현 및 추론(knowledge representation and reasoning), 증거-기반 학습(evidence-based learning) 등 거의 인공지능 전 분야에 걸쳐 새로운 원동력을 제공하는 기회가 되었다. 퀴즈쇼에서 주어지는 질문들에 효과적으로 답하기 위해서는 인물, 지리, 사건, 역사 등을 포함하는 광범위한 지식베이스와 빠른 시공간 추론(temporal and spatial reasoning) 능력이 필요하다. 퀴즈쇼에서 등장하는 공간 질의(spatial query)들은 주로 주요 장소들 사이의 방향(direction), 포함(containment), 그리고 경계(border) 관계 등을 포함하고 있다. 이러한 공간 관계들로 두 공간 혹은 두 장소(GeoInstance) 사이의 관계를 표현한 것을 공간 지식이라 한다. 그리고 공간 추론이란, 기존의 공간 지식을 바탕으로 새로운 공간 지식을 유도해 내는 것을 의미한다. 대표적인 공간 추론기(spatial reasoner)들로는 PelletSpatial[1], QUSAR[2] 등이 있다. PelletSpatial은 효율성이 높은 경로 일관성(path-consistency) 알고리즘을 채용한 RCC-8 공간 추론기이다. QUSAR[2]은 PelletSpatial을 확장하여 RCC-8 및 CSD-9 추론도 가능하고, 뿐만 아니라 이 둘 간의 상호 교차 일관성 검사 기능도 추가된 공간 추론기이다.

최근 시맨틱 웹의 발전에 따라 웹에 존재하는 정보를 반-구조적 형태로 표현하는 것이 가능해지고 이를 이용해 대용량의 지식 베이스를 생성할 수 있게 되었다. 그에 따라 공간 추론에 사용되는 지식 베이스도 수십, 수백억 개의 지식들로 구성됨에 따라 단일 머신으로 공간 추론을 수행하기에는 성능적 한계가 존재한다. 이에 대한 해결책으로 최근 대용량 웹 스케일 지식 베이스 추론에 관한 연구가 활발히 진행 중이고, 대표적인 연구로는 WebPIE[3]가 있다. WebPIE는 분산 환경에서 RDFS 및 OWL 추론을 수행하는 대용량 분산 추론기이다. 그러나 RDFS 및 OWL 추론 규칙에 관해서만 추론을 수행하고 공간 지식에 대한 추론은 하지 못한다.

일반적으로 분산 시스템의 장점을 잘 이용하려면, 대규모 입력 데이터를 클러스터의 각 노드에 효과적으로 나누어서 할당해야 하고, 각 노드는 자신에 할당된 데이터를 토대로 서로 독립적으로 계산을 수행할 수 있어야 한다. 그러나 만약, 데이터들 사이에 강한 연관성과 의존성이 존재한다면, 이와 같이 각 노드가 서로 독립적으로 작업을 수행하도록 데이터를 분할하기 어렵게 되고, 결국 노드간의 빈번한 통신 유발로 인해 분산 시스템의 성능을 저하시키는 원인이 된다. 불행하게도 본 연구에서 다루고자 하는 CSD-9 및 RCC-8 기반의 대용량 공간 지식 베이스는 지식들 간의 연관성도 높고, 새로운 지식을 추론하기 위한 연산들 간의 상호 의존성도 크다. 따라서 분산 시스템을 이용한 대용량의 공간 추론기를 개발하기 위해서는, 먼저 효과적인 지식 분할(knowledge partitioning)을 통해 병렬화의 이점을 살릴 수 있도록 알고리즘을 설계해야 한다.

본 논문에서는 대표적인 병렬 분산 컴퓨팅 환경인 하둡/맵리듀스 프레임워크를 이용하여 방향 및 위상 관계를 추론하는 효율적인 대용량의 공간 추론 알고리즘을 제시한다. 본 알고리즘에서는 하둡/맵리듀스 프레임워크의 특성을 고려하여 병렬 분산처리의 효과를 높이기 위해, 지식 분할 문제를 맵 단계에서 해결하고, 이것을 토대로 리듀스

※ 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 SW컴퓨팅산업원천기술개발사업(SW)의 일환으로 수행하였음. [10044494, WiseKB: 빅데이터 이해 기반 자가학습형 지식베이스 및 추론 기술 개발]

단계에서 효과적으로 새로운 공간 지식을 유도하도록 설계하였다. 또한, 본 알고리즘은 초기 공간 지식베이스로부터 새로운 지식을 유도할 수 있는 기능뿐만 아니라 초기 공간 지식베이스의 불일치성도 미연에 감지함으로써 불필요한 지식 유도 작업을 계속하지 않도록 설계하였다. 본 연구에서는 하둡/맵리듀스 프레임워크로 구현한 대용량 공간 추론기와 샘플 공간 지식베이스를 이용하여 성능 분석 실험을 수행하고, 실험 결과를 소개한다.

2. 공간 추론 규칙

대용량 공간 추론 알고리즘을 설계하기 위해서는 먼저 공간 추론 규칙들을 정의할 필요가 있다. QUSAR[2]의 논문에서 밝힌 바와 같이 본 논문에서 사용하는 공간 추론 규칙은 크게 세 가지 유형이 있다. 첫 번째는 9개의 방향 관계들로 표현되는 공간 지식에 적용되는 CSD-9 공간 추론 규칙들이다. 9개의 방향 관계들이란, 동(E), 서(W), 남(S), 북(N), 북동(NE), 북서(NW), 남동(SE), 남서(SW), 그리고 동향(O)을 의미한다. 예를 들어, 'A'가 'B'의 북쪽에 위치하고, 'B'가 'C'의 북서쪽에 위치한다. 이러한 경우, 'A'가 'C'의 북 혹은 북서쪽에 위치 한다는 새로운 사실을 추론해낼 수 있다. 두 번째는 8개의 위상 관계들로 표현되는 공간 지식에 적용되는 RCC-8 공간 추론 규칙들이다. 8개의 위상 관계들이란, DC(disconnect), EC(externally connected), PO(partially overlapping), EQ(equal), TPP(tangential proper part), TPPi(tangential proper part inverse), NTPP(non-tangential proper part), NTPPi(non-tangential proper part inverse)를 의미한다. 예를 들어, 'A'가 'B'의 경계에 접해 있고, 'B'가 'C'를 접점 없이 완전히 내포한다. 이러한 경우, 'A'가 'C'와 서로 떨어져 있다는 새로운 사실을 추론할 수 있다. 이와 같이 기존의 공간 지식으로부터 공통 인자를 기준으로 새로운 사실들을 유도하는 추론 과정을 조합(composition)이라고 한다.

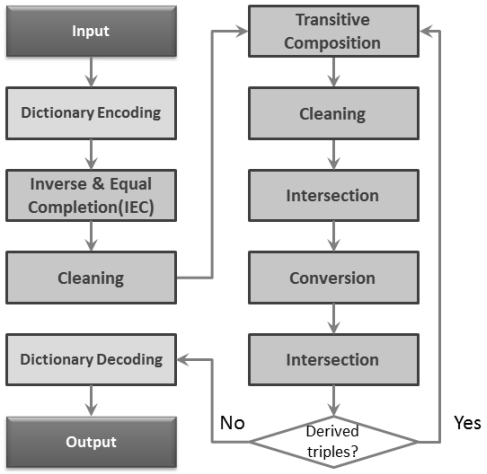
<표 1> CSD-9과 RCC-8 관계들 간의 변환 표

CSD-9	RCC-8
O	EQ, PO, TPPi, NTPPi, TPP, NTPP
N, NE, E, SE, S, SW, W, NW	DC, EC, PO

마지막으로, CSD-9과 RCC-8간의 변환 규칙들이다. 본래 CSD-9과 RCC-8은 각자 서로 다른 관점에서 공간 지식 표현과 추론 방법을 다루는 독립적인 이론들이다. 그러나 실세계의 많은 공간과 장소들은 CSD-9과 같은 방향 관계와 RCC-8과 같은 포함 관계를 함께 표현하고 추론해야 하는 경우가 많다. 따라서 이들을 통합적으로 추론 가능하게 하는 규칙이 필요하다. 따라서 <표 1>과 같은 CSD-9과 RCC-8 관계들 사이의 변환 규칙을 정의한다. 예를 들어, 'A'가 'B'를 완전히 내포하고 있을 때, 'A'는 'B'의 8개 방향 중 어느 방향에도 위치한다고 말할 수 없고, 동향(O) 관계라고 해석할 수 있다. 그리고 'A'가 'B'와 떨어져 있을 경우 'A'의 위치는 'B'로부터 8개 방향 중 하나에 해당한다고 해석할 수 있다. 이와 같이 기존의 공간 지식으로부터 서로 다른 관점의 새로운 사실들을 유도하는 추론 과정을 변환(conversion)이라고 한다.

3. 대용량 공간 추론

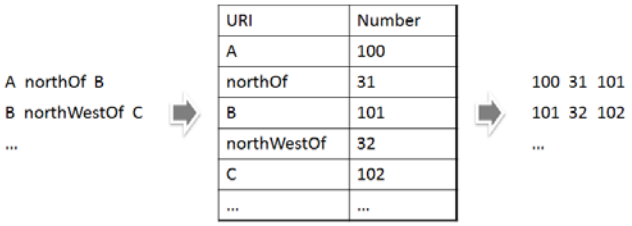
본 논문에서는 앞서 2장에서 언급한 공간 추론 규칙들을 토대로 맵리듀스 기반의 대용량 공간 추론 방식을 설계하였다. 전체 작업의 구성과 흐름은 (그림 1)과 같다. 공간 지식들은 서로간의 연관성이 강하고, 특히 추론을 위해 조인 작업을 해야 할 경우가 많다. 따라서 WebPIE[3]에서 밝힌 바와 같이, 입력 데이터를 각 노드에 알맞게 분배해야 알고리즘의 성능도 높이고 부하 분산(Load balancing) 문제도 해결할 수 있다. 본 알고리즘의 맵 단계들은 입력 데이터를 각 추론 작업에 알맞도록 각 노드에 분할 하는 역할을 한다. 그리고 리듀스 단계에서는 분할되어 들어온 데이터들에 대해 적절한 방법으로 새로운 공간 지식을 추론하는 역할을 한다.



(그림 1) 작업의 구성과 흐름

3.1 인코딩과 디코딩

(그림 1)에서 보는 바와 같이, 먼저 입력 데이터(Input)에 대한 인코딩 작업(Dictionary Encoding)이 진행된다. 이 작업은 문자열로 표현된 공간 지식을 임의의 숫자로 변환하여 공간 지식의 각 요소별로 하나의 숫자에 대응되는 사전을 만드는 작업이다. 이는 [4]에서 밝힌 바와 같이, 인코딩 작업 이후의 추론 단계뿐만 아니라 데이터 전송에 있어서도 효율성을 높일 수 있는 방법이다. 인코딩 작업의 예제는 (그림 2)와 같다.



(그림 2) Dictionary Encoding 예제

(그림 2)의 왼쪽은 공간 지식을 N-Triple 형태로 나타낸 것이다. 이처럼 공간 지식 내에는 장소(A, B, C)와 공간 서술자(northOf, northWestOf)가 존재한다. 이 때 공간 서술자는 총 17가지로 한정되기 때문에 미리 정의된 숫자로 할당되고, 장소는 임의의 숫자를 할당한다. 인코딩된 공간 지식은 그림의 오른쪽과 같이 트리플(Triple)형태로 표현된다. 입력 데이터에 대한 인코딩 작업을 성공적으로 마치게 되면 5가지의 추론 작업을 순차 또는 반복적으로 수행한다. 마지막으로, 유도된 공간 지식을 출력 데이터(Output)로 내보내기 전 임의의 숫자로 표현된 공간 지식을 문자열로 변환하기 위해 디코딩 작업(Dictionary Decoding)을 수행한다.

3.2. 역 관계 및 동일 관계 추론

입력 데이터에 대한 인코딩 작업(Dictionary Encoding)이 완료된 후, 역 관계 및 동일 관계 추론(Inverse & Equal Completion) 작업을 가장 먼저 수행한다. 이 작업은 입력으로 들어온 모든 공간 지식들에 대해 역(inverse) 관계와 동일(equal) 관계들을 생성한다. 예를 들어 (A northOf B)라는 하나의 공간 지식에 대해 역 관계인 (B southOf A)와 동일 관계들인 (A O A), (B O B)를 각각 생성한다. 역 관계 및 동일 관계 추론(IEC) 작업의 의사코드(pseudo code)는 (그림 3)과 같다. 맵 단계에서는 공간 지식의 주어와 목적어가 같지 않을 때, 역 관계를 생성한다. 반대로, 주어와 목적어가 같을 때 둘 사이의 공간

```
map(key, value) :
  // key : irrelevant
  // value : triple
  if (value.subj != value.obj) then
    inversePred = inverse_table.get(value.pred)
    write(triple(value.obj, inversePred, value.subj, CSD/RCCflag), null)
  else
    if (value.pred != EQ || value.pred != O) then
      exit          // not consistency

reduce(key, iterator values) :
  write(null, triple(key.subj, O, key.subj, CSDflag))
  write(null, triple(key.subj, EQ, key.subj , RCCflag))
  write(null, triple(key.obj, O, key.obj , CSDflag))
  write(null, triple(key.obj, EQ, key.obj , RCCflag))
  write(null, key)
```

(그림 3) 기초 지식 추론 작업

서술자가 EQ혹은 O가 아닌 것은 공간 지식 베이스의 불일치성을 발견한 경우이다. 이때는 더 이상의 추론을 하지 않고 모든 작업을 종료한다. 리듀스 단계에서는 주어와 주어, 목적어와 목적어 간의 동일 관계를 생성한다. 여러 노드에서 역 관계 및 동일 관계 추론(IEC) 작업이 수행되기 때문에, 이 작업의 결과로 중복된 공간 지식들이 생성될 수 있다.

따라서 이 작업 직후 중복된 지식을 제거(Cleaning)함으로써 다음 작업인 이행적 조합 추론(Transitive Composition) 작업의 부하를 줄일 수 있도록 설계하였다.

3.3. 이행적 조합 추론

앞선 작업이 완료된 후, 이어서 이행적 조합 추론(Transitive Composition) 작업을 수행한다. 이 작업은 입력으로 들어온 공간 지식들의 조합 관계를 생성한다. 예를 들어, (A northOf B)와 (B northWestOf C)라는 두 공간 지식을 이용해서 (A [northOf | northWestOf] C)라는 새로운 지식을 유도한다.

```
map(key, value) :
  // key : irrelevant
  // value : triple
  write(value.CSD/RCCflag + value.subj, '0' + value)
  write(value.CSD/RCCflag + value.obj, '1' + value)

reduce(key, iterator values) :
  for triple in values
    if (value[0] == 0) then
      join_right.add(triple)
    else
      join_left.add(triple)

  for left in join_left
    for right in join_right
      if (left.obj == right.subj) then
        composedPred = composition_table.get(left.pred, right.pred)
        inversePred = inverse_table.get(composedPred)
        write(null, triple(left.subj, composedPred, right.obj)
        write(null, triple(right.obj, inversePred, left.subj)
```

(그림 4) 이행적 조합 추론 작업

이행적 조합 추론(Transitive Composition) 작업의 의사 코드는 (그림 4)와 같다. 맵 단계에서는 공통 인자(match point)로 사용될 URI, 즉 주어와 목적어를 각각 출력의 키 값으로 구성한다. 밸류 값에는 키 값이 주어인지 목적어인지를 구분하는 인자와 입력으로 들어온 공간 지식으로 구성한다. 이러한 기법은 공통 인자를 기준으로 지식 분할이 이루어지기 때문에 일종의 조인 작업인 이행적 조합 추론을 수행하기에 적합한 방법이다. 리듀스 단계에서는 입력으로 들어온 밸류 값을 두 개의 서로 다른 집합으로 나눈다. 그 다음, 두 집합에서 공통 인자를 갖는 공간 지식들 간의 조합 추론을 수행한다. 이 때 조합 추론을 위해 조합표(composition table)가 있어야 하고, 그 표는 QUSAR[2]에서 제시한 것과 동일하다. 그리고 새롭게 생성된 지식의 역 관계도 함께 유도한다. 역 관계 및 동일 관계 추론(IEC) 작업과 마찬가지로, 이 작업의 결과로 중복된 공간 지식들이 생성될 수 있다. 따라서 이들을 제거(Cleaning)하여 다음 작업의 효율성을 높일 수 있도록 설계하였다.

3.4. 교차 추론

다음으로 교차 추론(Intersection) 작업을 수행한다. 앞선 이행적 조합 추론 작업의 결과로 주어와 목적어 그리고 공간 관점은 같지만, 공간 서술자가 서로 다른 다양한 공간 지식이 생성될 수 있다. 따라서 이 작업은 위와 같은 조건의 공간 지식들의 교집합을 구하는 작업으로써, 확실성 높은 정제된 공간 지식을 유도한다. 예를 들어, (A [northOf | northWestOf] B)와 (A [northOf | northEastOf] B)라는 두 공간 지식이 존재할 때 (A northOf B)라는 확실성 높은 정제된 공간 지식을 유도한다.

```
map(key, value) :
  // key : irrelevant
  // value : triple
 write(value.subj + value.obj + value.CSD/RCCFlag, value)

reduce(key, iterator values) :
  for value in values
    refinedTriple = triple(value.subject, pred.intersect(
                refinedTriple.pred, value.pred), value.object)

  if(refinedTriple.pred == null) then
    exit        // not consistency

  if(isIntersected)  then  write(null, refinedTriple)
  else  write(null, value)
```

(그림 5) 지식 정제 작업

교차 추론(Intersection) 작업의 의사 코드는 (그림 5)와 같다. 맵 단계에서는 공간 지식의 주어와 목적어 그리고 공간 관점을 나타내는 인자를 출력의 키 값으로 구성하고 공간 지식을 밸류 값으로 구성한다. 이러한 기법은 교차 추론 가능한 공간 지식들을 기준으로 지식 분할이 이루어지기 때문에 리듀스 단계에서 정제된 공간 지식을 쉽게 유도해낼 뿐만 아니라, 중복된 공간 지식을 생성하지 않는 장점이 있다. 리듀스 단계에서는 함께 들어온 공간 지식들의 교집합, 즉 정제된 공간 지식을 유도하고 그 결과를 출력한다. 그러나 정제된 공간 지식이 존재하지 않는 경우, 즉 교집합이 공집합인 경우에는 공간 지식 간에 불일치성을 발견한 경우이므로 더 이상 추론을 하지 않고 모든 작업을 종료한다. 예를 들어, (A northOf B)와 (A southOf B)라는 지식이 존재하는 경우에는 공간 지식들 간의 불일치성이 존재하는 경우이다.

3.5. 변환 추론

다음으로 변환 추론(Conversion) 작업을 수행한다. 이 작업은 두 관점의 공간 지식을 통합적으로 추론하기 위함이다. 예를 들어, (A northOf B)라는 공간 지식에 대해 위상 관점의 새로운 공간 지식인 (A [DC | EC | PO] B)를 생성한다.

```
map(key, value) :
  // key : irrelevant
  // value : triple
  conversionPred = conversion_table.get(value.pred)
  write(triple(value.subj, conversionPred, value.obj), null)

reduce(key, iterator values) :
  write(null, key)
```

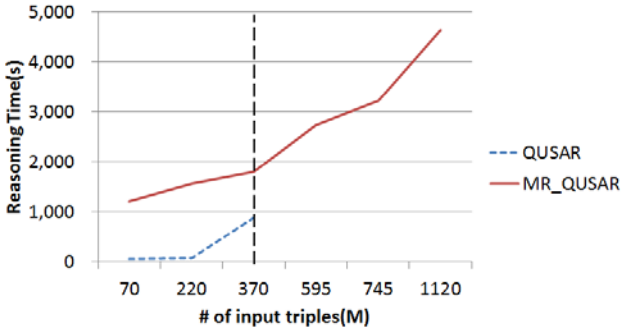
(그림 6) 변환 추론 작업

변환 추론(Conversion) 작업의 의사 코드는 (그림 6)과 같다. 맵 단계에서는 <표 1>과 같은 변환 규칙을 이용하여 서로 다른 관점의 공간 지식을 유도해낸다. 이 작업은 다른 작업들과는 다르게 지식 분할이 불필요한 작업이므로, 맵 단계에서 추론을 수행한다. 그리고 맵의 출력 중, 키 값을 변환된 공간 지식으로 구성함으로써 리듀스 단계에서 중복 제거 효과가 나타나도록 설계하였다. 따라서 이 작업 이후에 중복 제거 작업을 따로 수행하지 않아도 되기 때문에, 알고리즘의 효율성을 높일 수 있다. 이 작업이 완료되면 변환 추론을 통해 얻어진 새로운 공간 지식과 기존의 공간 지식간의 교집합을 통해 확실성 높은 공간

지식을 유도해야하기 때문에, 교차 추론(Intersection) 작업을 다시 수행한다. 그 다음 새롭게 유도된 지식이 있는지 검사한다. 새롭게 유도된 지식이 있다면 이행적 조합 추론(Transitive Composition) 작업으로 돌아가 다시 5개의 작업을 순차적으로 수행하는 것을 반복한다. 반대로 새롭게 유도된 지식이 없다면 디코딩(Dictionary Decoding) 작업을 수행하고 추론 작업을 마친다.

4. 구현 및 실험

본 논문에서 제안한 대용량의 공간 추론 알고리즘의 성능 분석 실험을 위해, 맵리듀스 프레임워크를 이용한 대용량의 공간 추론기 MR_QUSAR을 구현하였다. 구현 환경은 자바 1.6 버전과 하둡 1.2 버전을 사용하였고, 실험 환경은 5개의 노드로 구성된 하둡 완전 분산 모드 클러스터를 사용하였다. 마스터(Master) 노드는 네임노드(Namenode)와 잡트래커(JobTracker)를, 나머지 4개의 슬레이브(Slave) 노드는 데이터노드(Datanode)와 태스크트래커(TaskTracker)로 구성하였다. 각 슬레이브 노드는 쿼드코어 CPU와 8GB 메인 메모리, 500GB 하드 디스크로 구성되어있다.



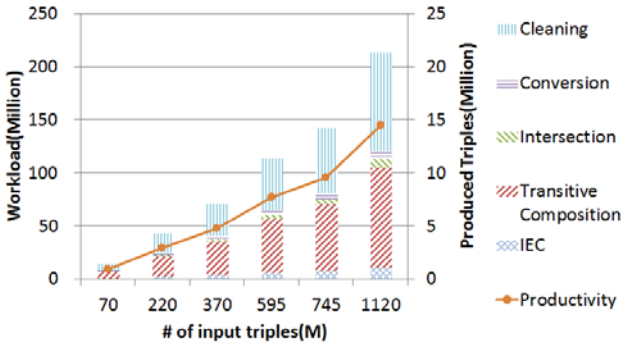
(그림 7) 추론 시간

확장성(Scalability)은 분산 시스템이 만족해야 하는 매우 중요한 요소이다. 따라서 첫 번째 실험에서는 데이터 확장성(Data Scalability) 면에서 QUSAR[2]과 본 논문에서 제안한 MR_QUSAR을 서로 비교해보았으며, 실험 결과는 (그림 7)과 같다. 그림을 통해, 우리는 약 100 만개 트리플로 구성된 대용량의 공간 지식베이스에 대한 MR_QUSAR의 추론시간은 약 4500초 내외임을 알 수 있다. 또한, 입력 공간 지식베이스의 크기가 증가해도 본 논문에서 제안한 MR_QUSAR의 추론 응답 시간은 선형적으로만 증가한 것을 확인할 수 있다. 이러한 결과는 본 논문에서 제안한 MR_QUSAR가 충분한 데이터 확장성을 가지고 있다는 것을 의미한다. 한편, QUSAR[2]의 경우에는 입력 공간 지식의 개수가 약 40만개에 도달하면 더 이상 추론이 불가능하였다. 이러한 결과는 QUSAR와 같은 단일 머신 공간 추론 알고리즘으로는 데이터 확장성에 한계가 있음을 의미한다.

<표 2> 대용량 공간 추론기의 성능 분석 표

입력 트리플 수	출력 트리플 수	평균 추론시간(s)	상대적 표준편차(%)
70588	1038928	1213	0.84
220525	3123294	1557	0.93
370360	5198708	1808	2.62
595487	8335376	2727	0.33
745416	10407516	3223	0.68
1120798	15622324	4622	0.51

안정성(Stability) 역시 분산 시스템이 만족해야 하는 또 다른 중요한 요소이다. 시스템의 안정성은 동일한 실험을 반복했을 때 어느 정도 동일한 수행 시간이 소요되는지를 측정해봄으로써 판별할 수 있다고 가정한다. 따라서 두 번째 실험은 본 연구의 MR_QUSAR의 안정성을 분석하기 위한 실험으로서, 각 입력 데이터 집합 별로 5번씩 동일한 실험을 수행하여 수행 시간을 측정해보았다. 이 실험의 결과는 <표 2>와 같다. 이 표를 통해, 우리는 동일한 실험에 대해 MR_QUSAR의 추론 응답 시간의 상대 표준 편차(relative standard deviation) 값이 모두 3%이하 인 것을 알 수 있으며, 이를 통해 본 논문에서 제안한 MR_QUSAR가 높은 안정성을 가지고 있음을 확인할 수 있다.



(그림 9) 생산량 및 작업량

세 번째 실험에서는 본 논문에서 제안한 MR_QUSAR의 생산량(productivity)과 작업량(workload)을 세부 작업별 비교 분석해보았다. 이 실험의 결과는 (그림 9)와 같다. 여기서 생산량은 모든 작업이 끝난 후 새롭게 유도된 지식의 양이라고 가정한다. 작업량은 각 작업의 출력 데이터 양과 입력 데이터 양의 차로 측정한다고 가정한다. 따라서 작업량은 역 관계 및 동일 관계 추론, 이행적 조합 추론, 변환 추론 작업들의 경우에는 각 작업이 얼마나 많은 새로운 지식을 추론 하였는지를, 교차 추론과 중복 지식 제거 작업의 경우에는 얼마나 많은 불필요한 공간 지식들이 정제되었는지를 각각 나타낸다. 그림을 통해, 우리는 약 100 만개 트리플로 구성된 입력 공간 지식 베이스에 대해, 본 논문의 MR_QUSAR은 약 1400 만개가 넘는 새로운 지식들을 유도하였다는 것을 알 수 있으며, 이를 통해 MR_QUSAR의 높은 생산성을 확인할 수 있다. 세부 작업별로 작업량을 살펴보면, 이행적 조합 추론 작업과 중복 제거 작업이 전체 작업 생산성의 약 88% 정도를 차지한 것을 알 수 있다.

5. 결론

본 논문에서는 하둡/맵리듀스 프레임워크를 이용하여 방향 및 위상 관계를 추론하는 효율적인 대용량의 공간 추론 알고리즘과 구현된 공간 추론기를 소개하였다. 샘플 공간 지식베이스를 이용한 성능 분석 실험을 통해, 본 논문에서 제시한 공간 추론 알고리즘과 공간 추론기의 높은 성능을 확인 할 수 있었다.

참고문헌

[1] M. Stocker and E. Sirin, "PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine", OWLED. 2009.

[2] Sangha Nam, et al. “Design and Implementation of a Qualitative Spatial Reasoner Based on CSD-9 and RCC-8 Theories”, Proc. of KIISE Fall Conference, pp 652-654, 2013.

[3] Urbani, Jacopo, et al. "WebPIE: A Web-scale parallel inference engine using MapReduce." Web Semantics: Science, Services and Agents on the World Wide Web, vol.10, pp 59-75 2012.

[4] Urbani, Jacopo, et al. "Massive semantic web data compression with mapreduce." Proc. of the 19th ACM Int. Symp. on HPDC. ACM, pp 795-802, 2010.