데이터 마이닝을 위한 이동 에이전트의 효율적인 이주 전략

권 혁 찬† · 유 우 종†† · 김 흥 환††† · 유 관 종††††

요 약

본 논문에서는 데이터 마이닝 (data mining)을 위한 이동 에이전트의 효율적인 이주 전략 알고리즘을 제시한다. 제시한 알고리즘의 목적은 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 계획을 세우는 것이다. 본 논문의 이주 전략 알고리즘을 검증하고 평가하기 위해 데이터 마이닝을 수행하기 위한 세 가지 패러다임 - RPC (Remote Procedure Call), 이동 에이전트, locker 패턴이 적용된 이동 에이전트 - 에 대한 수행 평가 모델을 제시하였으며, 시뮬레이션을 수행하여 알고리즘을 평가하였다.

An Efficient Migration Strategy of Mobile Agents for Data Mining

Hyeok-Chan Kwon† · Woo-Jong Yoo†† · Heung-Hwan Kim††† · Kwan Jong Yoo††††

ABSTRACT

In this paper, we present an efficient migration strategy of mobile agent for data mining applications. The purpose of the proposed algorithm is to set up the best migration plan of mobile agent with regard to minimizing network execution time. In order to verify the effectiveness of the proposed algorithm, we designed a performance evaluation model for three paradigms from data mining, i.e: RPC, mobile agent and mobile agent with locker pattern, and we then evaluated the algorithm by simulation.

1. 서 론

최근 네트워크 환경이 급부상 하면서 이동 에이전트(mobile agent)에 대한 요구가 급증되고 있다. 기존에 클라이언트/서버 구조에 기반을 둔 RPC나 멀티 에이전트(multi agent)의 경우 원격 통신에 의해 작업을 수행했던 반면 이동 에이전트 시스템은 에이전트 코드 자체가 서버로 직접 이동하여 주어진 작업을 수행하는 구조를 갖는다[1, 2].

많은 논문에서는 이러한 이동 에이전트의 이동성(mobility)이라는 특성이 원격 통신비용을 줄여줌으로, 네트워크 트래픽(network traffic)을 감소시킬 수 있다고 주장한다[2, 3]. 이러한 주장은 신중한 분석과 양적인 수치를 통해 지지되기보다는 보통 직관적이고 피상적인 방법으로 지지되고 있다. 실제로 이동 에이전트를 이용하여 개발한 분산 시스템이 기존의 접근 방식에 비해 성능이 좋은 지의 여부는 아직도 의견이 분분하다. 노드간의 통신 횟수, 전송 데이터의 양, 에이전트의 크기, 네트워크 상태 등의 요소에 따라 이동 에이전트는 매우 다양한 성능을 보이기 때문이다. 또한

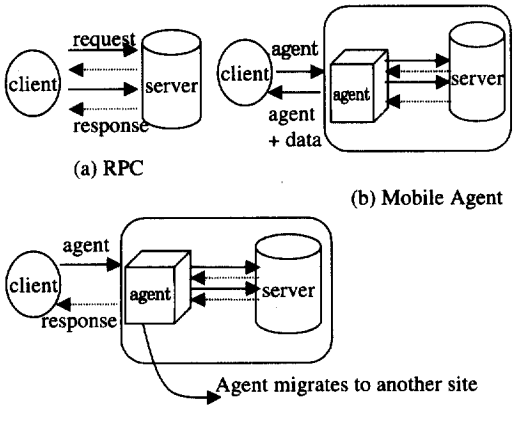
† 준 회 원 : 충남대학교 대학원 컴퓨터과학과
†† 정 회 원 : 대전 보건대학 전산정보처리과 교수
††† 정 회 원 : 서원대학교 컴퓨터정보통신학부 교수
†††† 정 회 원 : 충남대학교 정보통신공학부 교수

논문접수 : 2000년 1월 18일, 심사완료 : 2000년 4월 28일

분산 시스템의 전체 성능에 큰 영향을 주는 요소로서 에이전트의 이주방식이 있다.

본 논문에서는 데이터 마이닝을 위한 이동 에이전트의 효율적인 이주 전략 알고리즘을 제시한다. 제시하는 알고리즘의 목적은 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 계획을 세우는 것이다. 제안한 이주 전략 알고리즘을 검증하고 평가하기 위해 데이터 마이닝을 수행하기 위한 세 가지 패러다임의 수행 평가 모델을 제시하였으며, 시뮬레이션을 수행하여 알고리즘을 평가하였다.

분산 시스템의 성능은 네트워크 소요 시간, 노드의 프로세싱 로드(processing load), 보안등의 문제를 함께 고려하여 평가되어야 하지만, 본 연구에서는 네트워크 소요시간만을 고려하였으며, 다른 요소들은 추후의 연구에서 고려할 예정이다. 멀티캐스트(multicast) 방식도 고려하지 않았다. 본 논문에서 고려하는 패러다임(paradigm)은 RPC, 이동 에이전트, 그리고 locker 패턴을 적용한 이동 에이전트이다. Locker 패턴[4]은 이동 에이전트의 수행패턴 중 하나로, 에이전트가 방문하는 각각의 노드에서 획득한 결과를 임의의 영역에 보관시키고 다음의 노드로 이동하는 구조이다. 이 때 결과는 또 다른 이동 에이전트를 파견하여 수집하거나, RPC 방식으로 클라이언트에게 전달하게 된다. 본 논문에서는 후자의 경우로 가정하였다. (그림 1)에서는 각각의 패러다임의 수행 방식을 보인다.



(그림 1) 데이터 마이닝을 수행하기 위한 3가지 패러다임

본 논문의 구성은 다음과 같다. 먼저 2장에서 이동 에이전트의 이주 전략과 관련된 기존의 연구를 설명하였다. 3장에서는 수행 평가 모델과, 이 모델을 기초로 한 이동 에이전트의 이주 전략 알고리즘을 제안한다. 4장에서는 실제 샘플 데이터를 제안된 알고리즘으로 실험한 결과를 제시하고, 마지막 결론을 5장에서 맺는다.

2. 관련연구

[5]에서는 이동 에이전트가 이주할 때 발생할 수 있는 노드 혹은 호스트의 결점에 대처하기 위한 이주 정책을 제안하였다. 호스트나 노드의 결손이 발생한 경우에 이동 에이전트의 이주를 보장하기 위해 경로 재조정과 후위 복구 기법을 통한 이주정책을 제안하였다. [6, 7]에서도 방문하는 노드의 결점에 대처하기 위한 프로토콜을 제시하였다. 이처럼 이동 에이전트의 이주 전략과 관련하여서는 노드의 결점이 발생하였을 때 이주 신뢰성을 보장하기 위한 연구가 주류를 이루고 있으며, 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 계획을 세우는 알고리즘에 대한 연구는 아직 제시되지 않았다.

[8]에서는 분산 시스템 개발을 위한 세 가지 패러다임 - Client/Server, Remote Evaluation, Mobile Agent - 의 수행방식과 특징을 비교하였으며, 분산 시스템을 개발할 때 각각의 패러다임을 사용하는 경우, 애플리케이션 전체를 수행하는 동안 네트워크 상에서 전송되는 총 데이터 양을 계산하는 수식을 제공하였다. 이 수식을 기초로 몇 가지 응용 분야에 대해 세 가지 패러다임 중 적합한 한가지 패러다임만을 선택하기 위한 가이드라인을 제시하였다. [9]에서 가정한 이동 에이전트는 고정된 순서대로 노드를 방문하며, 방문하는 노드에서 검색한 데이터를 계속 누적시키며 이동하는 단순 구조만을 갖기 때문에, 노드의 수나 검색되는 데이터의 양이 조금이라도 많은 시스템에서는 이동 에이전트를 사용하는 것이 매우 비효율적이라는 문제가 있다. 실제 [8]에서는 거의 대부분의 분산 시스템에서 이동 에이전트가 가장 많은 네트워크 트래픽을 발생하는 패러다임이라고 지적한다. 그러나 실세계에서 이동 에이전트는 다양한 방식의 수행이 가능하다. 본 논문에서 고려하는 것처럼 이동 에이전트에 locker 패턴을 적용하거나, RPC를 혼합하여 사용하는 경우도 가능하

다[4, 9, 10]. 또한 [8]에서는 분산 시스템을 개발할 때 시스템의 성능을 평가하기 위해 네트워크 상에서 전송되는 총 데이터 양을 기준으로 비교하였으나, 그보다는 네트워크 소요시간을 기준으로 비교하는 것이 더 실제적일 것이다. 만약 네트워크 상태가 매우 좋은 경우라면 네트워크 상에 전송되는 총 데이터 양은 시스템을 평가하는데 중요한 요소가 될 수 없을 것이며, 오히려 그러한 경우라면 노드간의 원격 통신 횟수가 더 중요한 요소로 작용할 것이다.

본 논문에서 고려하는 수행구조는 이동 에이전트, RPC 그리고 locker 패턴이 적용된 이동 에이전트를 혼합하여 분산 응용 시스템을 개발하는 것이며, 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 전략을 세우는 것이다. 알고리즘을 평가하기 위해 작성한 수행 평가 모델은 각 패러다임의 특성과 데이터 마이닝에 적용 가능하도록 파라미터를 정하고, 각각의 노드에서 발생하는 네트워크 소요시간을 수식화 한 것이다. 또한 데이터 마이닝의 특성에 맞도록 이전에 검색된 데이터와 현재 방문한 노드간의 중복 데이터를 계산하기 위한 수식도 함께 제시하였다.

3. 이주 전략

3.1 수행 평가 모델

수행 평가 모델은 균등 네트워크(uniform network) 환경을 가정하였다. 이는 수식의 복잡성을 줄이기 위해 가정한 것이며, 비균등 네트워크 환경에 적용하기 위한 연구는 향후 수행할 예정이다. 모든 요구 메시지(request message)의 크기는 평균적인 수치로 통일하였다. 실제 데이터 마이닝에서 요구 메시지는 헤더에 대한 요구와 다큐먼트에 대한 요구 두 가지 뿐이며, 그 차이는 실제 전송되는 데이터들에 비해 무시해도 좋을 정도로 매우 작기 때문에 그렇게 가정하였다. 각각의 노드에 존재하는 다큐먼트가 필요로 하는 정보일 확률은 r로 고정하였다($0 \le r \le 1$). 실제 응용에서 필요한 정보가 해당 노드에 있을 확률에 대한 정확한 수치를 얻는 것이 현실적으로 어려운 일이므로, 임의로 고정된 값을 갖도록 가정하였다.

모델링을 위해 사용되는 파라미터는 <표 1>과 같다. <표 1>에서 전송을 위한 이동 에이전트(Ma)의 크기는 식 (1)과 같다.

〈표 1〉 모델링에 사용된 파라미터

	설명	단위
N	네트워크 노드(node)의 수	개수
M_j	j 노드에 존재하는 다큐먼트의 수	개수
r	노드내에 필요한 정보가 있을 확률	$0 \le r \le 1$
req	요구 메시지의 크기	bytes
rep_{hj}	j 노드에서 반환되는 하나의 헤더의 크기	bytes
rep_{dj}	j 노드에서 반환되는 하나의 다큐먼트의 의 크기	bytes
δ	네트워크 지연시간 (network delay)	ms
β	평균 네트워크 대역폭(bandwidth)	KB/sec
Ma	이동 에이전트의 크기	KB

$$Ma = DMA + (DMA/payload) * (IPheader + TCPheader) \quad (1)$$

$$DMA = ATP_{header} + SMA$$

$$SMA = M_{state} + M_{code} + M_{data}$$

이동 에이전트의 크기는 식(1) 에서처럼 이동 에이전트의 코드와 상태와 데이터 부분을 합한 SMA이다. 이동 에이전트의 전송을 위해서 SMA에 ATP(Agent Transfer Protocol) 헤더가 붙는다. ATP 헤더가 붙은 DMA는 TCP 메시지로 분할된다. 식 (1)의 payload는 TCP payload 이다. 분할된 각각의 조각들은 TCP헤더와 IP헤더로 싸여 TCP/IP 방식으로 전송된다. request, reply 메시지는 IP 헤더와 TCP 헤더에 싸여 전송된다[9].

<표 2>는 각 노드간의 중복 데이터를 계산하기 위해 추가되는 파라미터이다.

〈표 2〉 중복 데이터 계산을 위해 추가되는 파라미터

	설 명
UR	전체 데이터에 대한 unique한 데이터의 비율 (unique rate)
NUR	1-UR (non-unique rate)
TD	전체 노드의 다큐먼트 총 합(total document)
VD	이전에 발견한 다큐먼트의 합(visited document)

본 논문에서는 노드 내의 중복 데이터는 없는 것으로 가정하였다. 일반적으로 동일한 서버 내에 데이터가 중복되는 경우는 거의 없기 때문이다. 중복된 데이터를 제거하고 남은 데이터의 수는 TD*UR 이 된다. j 노드에서 제거되어야 할 중복데이터의 평균적 수는 식 (2)와 같다.

$$NM_j = 0 \quad \text{if } j = 1 \quad (2)$$

$$M_j \times \{VD/(TD - M_j) \times (2 \times NUR)\} \quad \text{if } j > 1$$

<표 3>에서는 네트워크 노드(N)가 5개, unique rate (UR)가 80%(0.8)일 때 중복데이터의 제거 과정을 보여준다. 제거되는 중복데이터의 수는 소수 둘째 자리에서 반올림 한 것이다.

〈표 3〉 중복 데이터 제거

노드	다큐먼트수	제거되는 중복데이터		
		수	계 산 식	비율
1	10	0	0	0%
2	30	1.7	30*{10/(100-30) * (2*0.2)}	5.7%
3	20	4	20*{40/(100-20) * (2*0.2)}	20%
4	10	2.6	10*{60/(100-10) * (2*0.2)}	26.7%
5	30	12	30*{70/(100-30) * (2*0.2)}	40%
합계	100	20.3		
j	M_j		M_j*{VD/(TD-M_j)*(2 * NUR)}	

중복 데이터의 여부는 헤더의 정보를 통해 판별한다. 만약 다큐먼트 내의 텍스트 내용에 의해 중복 데이터를 판별한다면, 모든 데이터를 갖고 이동하는 이동 에이전트의 경우에 제거시키는 중복 데이터가 더 많겠지만, 중복 데이터를 판별하기 위해 소요되는 시간이 상당히 길어지는 문제가 있기 때문에 일반적으로 헤더의 정보를 이용한다.

본 논문에서 고려하는 3가지 패러다임 각각의 중복 데이터 제거 방법은 다음과 같다. 먼저 RPC 방식의 경우, 클라이언트 노드에서 서버 노드로부터 받은 헤더 정보를 계속 보관, 유지하므로 중복 데이터의 여부를 판별한다. 서버로부터 헤더 정보를 받으면 먼저 필요로 하는 데이터를 가려낸 후, 이전에 보관한 헤더와 비교하여 중복되는 데이터를 확인한 후에 서버로 필요한 다큐먼트를 요청한다. 이동 에이전트의 경우, 수집된 다큐먼트를 갖고 이동하기 때문에 각 노드에서 중복 데이터의 여부를 판별 할 수 있다. locker 패턴의 경우엔, 데이터의 중복 여부를 판별하기 위해, 노드들을 방문할 때 수집된 다큐먼트를 서버로 전송하고, 수집된 다큐먼트의 헤더는 이동 에이전트의 데이터 영역에 누적시키며 이동한다.

RPC의 경우 j 노드에서의 네트워크 부하(network load)는 식 (3), 네트워크 소요시간(network execution time)은 식 (4)와 같다. UM_j는 중복데이터를 제거한 후의 다큐먼트 수이다. 즉 M_j에서 식 (2)의 NM_j를 뺀 수가된다.

$$L_{RPC-S}=(1+r\cdot UM_j)req+UM_j(rep_{hj}+r\cdot rep_{dj}) \quad (3)$$

$$T_{RPC-S}=\frac{L_{RPC-S}}{\beta}+(2+2r\cdot UM_j)\delta \quad (4)$$

RPC의 경우 request는 처음에 헤더를 요구하기 위해 1번, 원하는 다큐먼트 각각에 대해 $r\cdot UM_j$ 번 하게 된다. 네트워크 지연시간 발생 횟수는 헤더에 대해 2번과 각각의 다큐먼트에 대해 $2r\cdot UM_j$번 소요된다. 먼저 헤더만을 가져와서 필요한 다큐먼트인지 평가하기 때문에 실제 서버에 존재하는 모든 다큐먼트를 다 가져오지 않아도 된다.

이동 에이전트의 경우 j 노드에서의 네트워크 부하는 식 (5), 네트워크 소요시간은 식 (6)과 같다.

$$L_{MA}=2Ma+\sum_{k=1}^{t}(r\cdot UM_k rep_{dk})+r\cdot rep_{dj}UM_j \quad (5)$$

(단, t는 이동 에이전트가 현재까지 방문한 노드 수)

$$T_{MA}=2\delta+\frac{L_{MA}}{\beta} \quad (6)$$

이동 에이전트의 경우에는 에이전트가 직접 서버로 이동하여 작업을 수행하기 때문에 에이전트 이동에 관한 2번의 네트워크 지연시간이 소요된다. 식 (5)에서 $\sum_{k=1}^{t}(r\cdot UM_k rep_{dk})$은 이전에 방문했던 노드들에서 누적된 데이터의 크기를 나타낸다. $r\cdot rep_{dj}UM_j$ 은 현재의 노드에서 누적되는 데이터의 양이다.

locker 패턴이 적용된 이동 에이전트의 네트워크 부하는 식(7), 네트워크 소요시간은 식 (8)과 같다.

$$L_{MA(L)}=MH+r\cdot UM_j rep_{dj} \quad (7)$$

$$MH=Ma+\sum_{k=1}^{t}(r\cdot UM_k rep_{hk})$$

(단, t는 locker 패턴으로 방문한 노드 수)

$$T_{MA(L)}=\frac{L_{MA(L)}}{\beta}+(1+r\cdot UM_j)\delta \quad (8)$$

식 (7)에서처럼 네트워크 부하는 이동하는 에이전트의 크기와, 서버에서 작업수행 후 수집된 결과를 클라이언트로 전달하기 위한 데이터만 고려하면 된다. locker 패턴은 중복 데이터 제거를 위해 다큐먼트의 헤더를 누적시키며 이동하지만 전체 이동하는 데이터에 비해 헤더의 크기는 매우 작다. locker 패턴의 경우 현 노드

에서 수집된 결과를 RPC방식으로 서버에 전달하기 때문에 $r \cdot UM_j rep_{dj}$만큼의 네트워크 부하가 추가로 소요된다.

본 모델에서 가정한 네트워크 환경은 균일한 대역폭을 갖는다. 만일 각기 다른 대역폭을 갖는 서브네트워크(sub network) 환경이라면 조금 다른 모델이 적용될 것이다. 만약 A 노드에서 B 노드로 RPC를 이용하거나, 이동 에이전트가 이동하는 경우, 각 서브 네트워크간의 대역폭이 β_0이고 B노드가 속한 서브 네트워크 내의 대역폭이 β_1이라면 실제 적용되는 대역폭은 두 대역폭의 최소 값인 $\min(\beta_0, \beta_1)$이 될 것이다. 이러한 비균일 네트워크에 대한 모델링은 현재 작업중이다.

3.2 이주 전략 알고리즘

각각의 패러다임은 파라미터(parameter) 값에 의해서 성능의 좋고 나쁨이 결정된다. 주된 영향을 미치는 파라미터로 네트워크 대역폭, 누적되는 데이터 양, 서버와의 상호작용(interaction) 횟수, 이동 에이전트의 크기를 들 수 있다. 이동 에이전트의 경우 식(5), (6)에서 볼 수 있듯이, 네트워크 연결을 처리하기 위한 네트워크 지연시간은 적은 반면, 이동 에이전트와 누적되는 데이터 양이 많은 문제가 있다. 반면 RPC의 경우는 식(3), (4)에서 볼 수 있듯이, 글로벌 통신 횟수가 많으므로 네트워크 연결을 처리하기 위한 네트워크 지연시간은 많이 소요되는 반면, 네트워크 상에서 이동하는 데이터 양이 적다는 장점이 있다. 따라서 네트워크 대역폭이 좋을수록 이동 에이전트가 RPC에 비해 유리하며, 글로벌 통신 횟수가 적을수록 RPC가 유리할 것이다. 본 논문에서 고려하는 데이터 마이닝에서의 가정에 의하면 글로벌 통신 횟수는 각 노드에 존재하는 다큐먼트의 수에 비례하여 증가한다. 이동 에이전트의 크기는 이동 에이전트와 locker 패턴 모두의 성능에 영향을 미치는 요소이다.

RPC와 이동 에이전트의 경우 식 (9)의 조건이 만족되는 경우 이동 에이전트를 이동하는 것이 효과적이다. 식 (9)는 네트워크 지연시간 δ의 발생 빈도를 기준으로 정리한 것이며, 식 (9)의 AD(t)는 $\sum_{k=1}^{j}(r \cdot UM_k rep_{dk})$이다.

$$T_{RPC-S} > T_{MA}$$

$$r \cdot UM_j \delta > \frac{2Ma + AD(t) - (r \cdot req + rep_h) UM_j - req}{\beta} \tag{9}$$

식 (9)에서처럼 총 소요되는 네트워크 지연시간과 실제 데이터의 전송시간을 비교하여 RPC와 이동 에이전트 중 하나의 패러다임을 선택할 수 있다.

RPC와 locker 패턴이 적용된 이동 에이전트의 경우 식 (10)이 만족되는 경우 locker 패턴이 적용된 이동 에이전트를 이동하는 것이 효과적이다. 식 (10)도 네트워크 지연시간 δ 발생 빈도를 기준으로 정리한 것이다.

$$T_{RPC-S} > T_{MA(L)}$$

$$(1 + r \cdot UM_j)\delta > \frac{MH - (r \cdot req + rep_h) UM_j - req}{\beta} \tag{10}$$

식 (10)에서도 역시 총 네트워크 지연시간과 실제 데이터의 전송에만 소요되는 시간을 비교하여 결정한 형태가 된다.

이동 에이전트와 locker 패턴이 적용된 이동 에이전트의 경우 식 (11)이 만족되는 경우 locker 패턴이 적용된 이동 에이전트를 이동하는 것이 효과적이다. 식 (11)은 이동 에이전트에 누적되는 데이터 량을 기준으로 정리한 것이다.

$$T_{MA} > T_{MA(L)}$$

$$\frac{AD(t) - Ma}{\beta} > (r \cdot UM_j - 1)\delta \tag{11}$$

식 (11)에서 보면 총 네트워크 지연시간과 누적되는 데이터 양을 비교하여 결정한 형태가 된다.

실제 분산 응용 시스템 개발 시 적합한 수행 패턴을 결정하기 위한 알고리즘은 다음과 같다.

3.2.1 노드 방문 순서가 고정된 경우

노드 방문 순서가 고정된 경우, 즉 노드 방문 순서를 변경하지 않은 경우, 알고리즘은 다음과 같다. 알고리즘에 나오는 MF(Migration Function)은 3.1절에서 제안한 수행 평가 모델과 비교 수식을 기초로 작성된 것으로, 특정 노드에서의 파라미터 값들을 참조하여 각각의 패러다임의 수행 시간을 예측하는 함수이다.

```
input : node list, other parameters ....
output : selected_paradigm_list, agent_position_list

Set current_agent_position as client_node
For node# from 1 to last_node
  Choose paradigm which has minimum execution
                                    time by MF.
  Add selected_paradigm into selected_paradigm_list
  If selected_paradigm = locker pattern
      Update accmulated_header value
      Set current_agent_position as node#
  If selected_paradigm = MA
      Update AD(t) value
      Set current_agent_position as node#
End loop;
```

3.2.2 노드 방문 순서를 변경하는 경우

노드 방문 순서가 고정되지 않은 경우, 방문 순서를 변경함으로서 성능을 향상시킬 수 있을 것이다. RPC와 locker 패턴이 적용된 이동 에이전트의 경우의 성능은 노드의 방문 순서에 거의 영향을 받지 않는다. 한 노드에서 다른 노드로 이동시 누적되는 데이터가 없기 때문이다. locker 패턴의 경우 헤더를 누적시키기는 하지만 전체 이동하는 데이터 양에 비해 헤더의 크기는 무시할 만큼 매우 작기 때문이다. 그러나 이동 에이전트의 경우에는 노드의 방문 순서에 의해 매우 다른 성능을 보일 수 있다. 데이터를 계속 누적시키며 이동하기 때문이다. 방문 순서를 변경하는 것을 포함한 알고리즘은 다음과 같다.

```
input : node list, other parameters ....
output : migration table that composed of pair of
         (selected_node, paradigm)

Set current_agent_position as client_node
While unvisited_node = none
  Visit node by increasing order of total doc. size
  For each node
    Choose paradigm which has minimum execution
                                    time by MF.
    Add selected_node and paradigm into migration
                                    table
    If selected_paradigm = locker pattern
        Update accmulated_header value
        Set current_agent_position as node#
    If selected_paradigm = MA
        Update AD(t) value
        Set current_agent_position as node#
    End loop; /* for */
End loop; /* while */
```

4. 실 험

본 장에서는 실제 알고리즘을 적용하여 시뮬레이션한 결과를 보인다. 실험을 위한 도구로는 Visual C++ 6.0을 이용하였다.

전체 노드의 수 N은 8개, request 메시지의 크기는 50bytes, 헤더를 반환(reply)하는 메시지의 크기는 60bytes, 네트워크 지연시간 δ는 20ms, 네트워크 대역폭은 300KB/s, 필요로 하는 정보가 방문하는 노드에 존재할 확률 r은 40%, UR은 80%, 이동 에이전트의 크기는 15KB 인 경우를 예로 하여 실험을 하였다. 각 노드에서의 평균 다큐먼트 크기와 다큐먼트의 수는 <표 4> 와 같으며, 이 수치는 random하게 정한 것이다.

〈표 4〉 파라미터 값

노드 / 파라미터	1	2	3	4	5	6	7	8
평균 다큐먼트 크기(KB)	2	5	10	3	5	15	1.5	0.3
다큐먼트 수	6	7	2	30	50	5	20	40

<표 4>에서 주어진 고정된 노드 순서대로 방문한 경우 알고리즘을 적용하여 생성된 수행패턴은 <표 5>와 같다. <표 6>은 알고리즘을 역으로 적용했을 때의 수행 패턴이다.

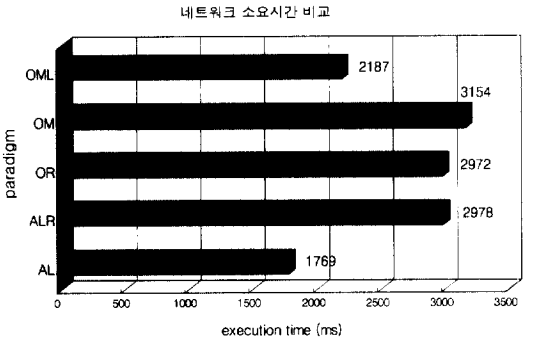
〈표 5〉 알고리즘 적용 후의 수행 패턴
(R:RPC, M:이동에이전트, L:locker 패턴)

방문노드	1	2	3	4	5	6	7	8	Client
패러다임	L	L	R	M	M	L	L	L	M
에이전트의 위치	1	2	2	4	5	6	7	8	Client

〈표 6〉 알고리즘을 역으로 적용한 경우

방문노드	1	2	3	4	5	6	7	8	Client
패러다임	M	M	M	R	R	M	M	R	M
에이전트의 위치	1	2	3	3	3	6	7	7	Client

(그림 2)에서는 RPC만을 이용한 경우와 이동 에이전트만을 이용한 경우, locker 패턴만을 이용한 경우, 알고리즘을 적용한 경우와 역으로 적용한 경우의 총 네트워크 소요시간을 비교하였다. 알고리즘을 적용하여 결정된 수행패턴이 최소의 네트워크 소요시간을 갖는다는 것을 확인 할 수 있다.



(그림 2) 네트워크 소요시간 (OML: locker 패턴만 사용, OM: 이동 에이전트만 사용, OR: RPC만 사용, ALR: 알고리즘을 역으로 적용, AL: 알고리즘 적용)

<표 7>에서는 이동 에이전트의 데이터 영역에 누적되어 전송되는 데이터가 네트워크 상에서 이동한 총 네트워크 로드(network load)를 비교하였다. <표 7>에서 제공한 값은 소수 첫째자리에서 반올림한 값이다. <표 7>의 결과에서 볼 수 있듯이 현재 가정한 시나리오 상에서, 누적되는 데이터에 대한 총 네트워크 로드는 알고리즘을 적용한 경우와 이동 에이전트만 이용한 경우의 수치가 크게 나타났다. 그러나 실제 네트워크 소요시간과 관련하여서는, (그림 2)의 결과에서 확인하였듯이 알고리즘을 적용하여 결정된 수행패턴이 최소의 네트워크 소요시간을 갖는다. Locker 패턴만을 사용한 경우에도 RPC나 이동 에이전트만을 사용한 경우보다 더 적은 네트워크 소요시간을 갖는다는 점을 확인할 수 있다.

〈표 7〉 이동 에이전트에 누적된 데이터에 대한 네트워크 로드 (단위: bytes)

수행패턴 / 누적	알고리즘 적용	알고리즘 역적용	RPC	이동 에이전트	Locker 패턴
다큐먼트	117975	57533	0	178389	0
헤더	1318	0	0	0	3055

4.1 노드 방문 순서를 변경한 경우

노드의 방문 순서를 변경하는 경우의 알고리즘을 적용한 결과 수행패턴은 <표 8>과 같다. <표 9>는 알고리즘을 역으로 적용했을 때의 수행 패턴이다.

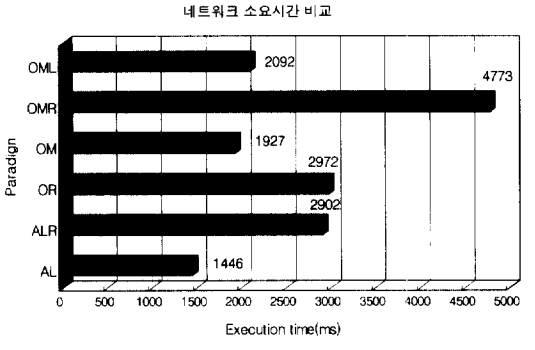
(그림 3)에서는 RPC만을 이용한 경우와 이동 에이전트만을 이용한 경우, 이동 에이전트를 총 다큐먼트 크기를 기준으로 역순으로 이동시킨 경우, locker 패턴만을 이용한 경우 그리고 알고리즘을 적용한 후와 역으로 적용한 후의 총 네트워크 소요시간을 비교하였다. 노드의 순서를 변경한 경우에도 알고리즘을 적용한 경우 최소의 네트워크 소요시간을 가진다는 것을 볼 수 있다.

〈표 8〉 알고리즘 적용 후의 수행 패턴 (방문순서 변경)

방문노드	1	8	3	7	2	6	4	5	Client
패러다임	L	M	R	M	L	L	M	M	M
에이전트의 위치	1	8	8	7	2	6	4	5	Client

〈표 9〉 알고리즘 역으로 적용 후의 수행패턴 (방문순서 변경)

방문노드	5	4	6	2	7	3	8	1	Client
패러다임	M	R	M	R	M	M	R	R	M
에이전트의 위치	5	5	6	6	7	3	3	3	Client



(그림 3) 방문 순서를 변경한 경우의 네트워크 소요시간 (OMR: 이동 에이전트를 역순으로 이동)

노드 순서를 변경하여 방문한 경우, 이동 에이전트의 데이터 영역에 누적되어 전송되는 데이터가 네트워크 상에서 이동한 총 네트워크 로드(network load)를 <표 10>에서 비교하였다. <표 10>에서 제공한 값은 소수 첫째자리에서 반올림한 값이다.

<표 10>의 결과에서 볼 수 있듯이, 현재 가정한 시나리오 상에서, 총 네트워크 로드는 알고리즘을 적용한 경우와 이동 에이전트만 이용한 경우와 이동 에이전트가 알고리즘의 역순으로 방문한 경우의 수치가 크

〈표 10〉 방문 순서를 변경한 경우 이동 에이전트에 누적된 데이터에 대한 네트워크 로드 (단위: bytes)

수행패턴 / 누적	알고리즘 적용	알고리즘 역적용	RPC
다큐먼트	102197	47573	0
헤더	379	0	0
수행패턴 / 누적	이동 에이전트	이동 에이전트 (역순 방문)	Locker 패턴
다큐먼트	149770	185993	0
헤더	0	0	3039

게 나타났다. 그러나 네트워크 소요시간과 관련하여서는, (그림 3)의 결과에서 확인하였듯이 알고리즘을 적용하여 결정된 수행패턴이 최소의 네트워크 소요시간을 갖게된다. 그리고 이동 에이전트만 이동 한 경우에도 더 적은 네트워크 소요시간을 갖는다는 점을 확인할 수 있다.

노드의 방문 순서를 고정한 경우와 변경한 경우의 총 네트워크 시간을 <표 11>에서 비교하였다. <표 11>에서 볼 수 있듯이, 동일한 시나리오에서 방문하는 노드의 순서와 패턴에 따라서 네트워크 소요시간이 큰 차이가 나는 것을 볼 수 있다. 노드의 순서를 변경하는 경우 RPC나 locker 패턴의 경우, 노드 방문 순서가 전체 성능에 거의 거의 영향을 끼치지 못하지만 특히 이동 에이전트의 경우에는 네트워크 소요시간이 많이 적어지는 것을 볼 수 있다.

〈표 11〉 노드 순서에 의한 비교 (단위 : ms)

	알고리즘 적용	알고리즘 역적용	이동 에이전트만 이용		Locker 패턴
방문순서 고정	1769	2978	3154		2187
방문순서 변경	1446	2902	1927	4773(역순)	2092

5. 결론 및 추후과제

본 논문에서는 데이터 마이닝을 위한 이동 에이전트의 효율적인 이주 전략 알고리즘을 제시하였다. 또한 제시한 알고리즘을 검증하고 평가하기 위해 세 가지 패러다임에 대한 수행 평가 모델을 작성하였으며, 시뮬레이션을 수행하였다. 본 논문에서 제시한 알고리즘은 데이터 마이닝 애플리케이션 개발 시 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 전략을 세우는데 도움이 될 수 있을 것이다. 또한 본 모델과 알고리즘을 일부 변경한다면 다른 애플리케이션에도 적용 가능할 수 있을 것이다.

실제 이동 에이전트의 전체 성능 평가요소로 네트워크 소요시간만을 고려한 것으로는 불충분하며, 추가적으로 노드의 프로세싱 로드, 보안등의 문제를 함께 고려해야 한다. 이에 대한 연구는 추후 수행할 예정이다. 본 논문에서는 균등한 네트워크(uniform network) 환경만을 고려하였다. 현재 균등하지 않은 네트워크(Non-uniform network) 환경을 고려한 모델링에 대한 연구가 진행중이다. 추후 과제로서 이동 에이전트의 좀 더 다양한 방식의 수행 패턴에 대한 분석과 이를 모델과 알고리즘에 적용하는 작업과, 에이전트의 지능성(intelligence)이라는 특성을 적용하여 성능을 높일 수 있는 기법 등에 관한 연구가 필요하다. 또한 실제 구현을 통하여 알고리즘을 검증해 보는 작업도 필요하다.

참 고 문 헌

[1] Bic, L. F., M. Fukuda, and M. B. Dillencourt, "Distributed computing using autonomous objects," IEEE Computer, Aug. 1996.

[2] Harrison, C. G., D. M. Chess and A. Kershenbaum, Mobile Agents : Are they a good idea?, IBM Watson Research Center, Mar. 1995.

[3] Wooldridge, M. and N. R. Jennings, Agent Theories, Architectures and Languages : A Survey, In Michael Jl. Wooldridge and Nicolas R.Jennings, editor, Intelligent Agent, pp.1-39, Springer-Verlag, Germany, 1995.

[4] Yariv, A., D. B. Lange, "Agent Design Patterns : Elements of Agent Application Design," Second International Conference on Autonomous Agents (Agents 98), 1998.

[5] 전병국, 최형근, "이동 에이전트를 위한 효율적인 이주정책의 설계 및 구현", 정보처리논문지, 제6권, 제7호, Jul. 1999.

[6] Baumann, J., A Protocol for Orphan Detection and Termination in Mobile Agent Systems, TR-1997-09, Stuttgart Univ. Jul. 1997.

[7] OMG, Mobile Agent Facility Interoperability Facilities Specification(MAF), OMG.

[8] Carzaniga, A., G. P. Picco, G. Vigna, "Designing Distributed Applications with Mobile Code Paradigms," Proceedings of the 19th International Conference on Software Engineering, Boston, 1997.

[9] Lange, D.B., M. Oshima, Programming and deploying Java Mobile Agents with Aglets, Ad-

dison Wesley Press, 1998.
[10] Nog, S., Chawla S., and D. Kotz, An RPC mechanism for transportable agents, Technical Report TR96-280, Department of Computer Science, Dartmouth College, Hanover, N.H., 1996.
[11] Tennenhouse, D. L., Jonathan M. S., W. David Sincoskie, David J. Wetherall, and Gary J. Minden, "A survey of active network research," IEEE Communications, 35(1) : 80-86, January, 1997

권 혁 찬

e-mail : hckwon@cs.cnu.ac.kr
1994년 서원대학교 전자계산학과 졸업(공학사)
1996년 충남대학교 전산학과 (이학석사)
1996년~현재 충남대학교 컴퓨터과학과 박사과정(수료)
관심분야 : 에이전트 시스템, 병렬처리, 멀티미디어 응용

유 우 종

e-mail : wjyoo@tjhealth.ac.kr
1987년 충남대학교 계산통계학과 졸업(학사)
1990년 충남대학교 계산통계학과 (이학석사)
1998년 충남대학교 전산학과 박사과정 수료
1987년~1993년 한국전자통신연구원 시스템 S/W연구실 선임연구원
1993년~현재 대전보건대학 컴퓨터정보처리과 조교수
관심분야 : 분산 멀티미디어, 실시간 미디어처리, 병렬처리, VOD

김 홍 환

e-mail : khh@seowon.ac.kr
1985년 서울대학교 계산통계학과 졸업(이학사)
1987년 서울대학교 계산통계학과 (이학석사)
1990년 서울대학교 계산통계학과 (이학박사)
1990년~현재 서원대학교 컴퓨터정보통신학부 부교수
1996년~1997년 콜로라도주립대학 전산과학과 객원교수
1997년 4월~12월 한국전자통신연구원 초빙연구원
1998년~1999년 건국대학교 전산과학과 부교수
관심분야 : 프로그래밍언어, 컴파일러, 병렬 및 분산처리 시스템, 웹컴퓨팅, 원격교육

유 관 종

e-mail : kjyoo@cs.cnu.ac.kr
1976년 서울대학교 계산통계학과 졸업(이학사)
1978년 서울대학교 대학원 전산학 전공(이학석사)
1985년 서울대학교 대학원 전산학 전공(이학박사)
1987년 3월~8월 충남대학교 전산학과 학과장
1987년~1989년 충남대학교 전자계산소 소장
1990년 1월~12월 캘리포니아 대학(Irvine) 방문교수
1979년~현재 충남대학교 공과대학 정보통신공학부 교수
1995년~현재 한국정보과학회 이사
관심분야 : Agent, Scalable Coding, 멀티미디어 응용, VOD, 병렬처리, 컴파일러 설계