JINI 기반의 컴퓨팅 리소스 매칭 최적화 시스템

서현승[O] 양성봉
연세대학교 컴퓨터과학과
{hsseo[O], yang}@cs.yonsei.ac.kr

A JINI based optimal matching system for computing resources

Hyunseung Seo[O] Sungbong Yang
Dept. of Computer Science, Yonsei University

요 약

클라이언트/서버 시스템은 클라이언트가 수행할 수 없거나 곤란한 프로그램을 서버에서 수행하여 클라이언트의 부하를 줄여준다. 하지만 서버의 리소스를 요청하는 클라이언트의 수가 많아지면 이미 많은 리소스를 사용하고 있는 서버에게 리소스를 요청하거나, 리소스를 사용하지 않는 서비스에게는 요청하지 않는 경우가 발생하여 서버에게 너무 많은 부하를 주거나 전혀 부하가 생기지 않을 수 있다. 이에 본 논문에서는 Jini 기반의 Agent Manager를 이용하여 서버와 클라이언트의 정보를 수집한 다음 MAUT와 Maximum-Weight Matching을 이용하여 클라이언트와 서버 간 연결의 전체적인 만족도를 높이는 Jini 기반의 최적의 컴퓨팅 리소스 제안 시스템을 연구하였다.

1. 서 론

초기 컴퓨터 환경은 호스트 중심의 중앙집중식 처리 환경으로 컴퓨터를 이용하려는 사람이 학교나 연구소와 같이 컴퓨터가 있는 장소로 가서 이용하는 방식이었다. 하지만, 근래에 개인용 컴퓨터 성능의 비약적인 발전, 컴퓨터 가격 하락, 컴퓨터의 소형화와 더불어 네트워크의 발전은 대부분의 사람들이 데스크탑을 이용하여 자신의 집에서 혹은 모바일 디바이스나 PDA를 이용하여 거리를 지나다니면서도 컴퓨터를 이용할 수 있는 환경으로 바뀌었다.

데스크탑 혹은 PDA, 모바일폰과 같은 모바일 디바이스가 연산량이 많거나 메모리 점유율이 높은 프로그램을 수행하기 위해선 자신의 많은 리소스를 한 프로그램에 할당해야 하므로 수행 시간이 많은 걸리거나 심지어 프로그램 수행이 되지 않을 수도 있다. 그러므로 개인용 컴퓨터 혹은 모바일 디바이스를 클라이언트로 하고, 클라이언트에서 수행하기 어려운 작업을 워크스테이션과 같은 서버에서 수행하여 하나의 응용을 두 개의 프로세서가 분산시켜서 처리하는 클라이언트/서버, 클라이언트/서버 데이타베이스 시스템이 출현하게 되었다. 클라이언트/서버 시스템은 클라이언트가 특정 프로그램이나 특정 IP를 이용하여 특정 서버의 리소스를 사용하므로 서버를 이용하려는 클라이언트가 적을 경우는 문제가 되지 않지만, 그 반대일 경우 서버에 과부하를 일으킴으로써 결과적으로 클라이언트가 빠른 시간 안에 수행 결과를 얻지 못한다.

본 연구에서는 Jini 환경에서 클라이언트와 서버가 산재해 있을 때 클라이언트와 서버를 연결함에 있어서 연결 중개자인 Agent Manager가 클라이언트에게 최적의 컴퓨팅 리소스를 제안하여 전체적인 만족도를 높이는 시스템을 제안한다.

2. Jini

Jini는 썬마이크로시스템에서 개발한 미들웨어로서 자바를 기반으로 하여 다양한 방식으로 네트워크에 접속된 장치나 소프트웨어를 동적으로 상호 작용하도록 하게 하는 기술[1][2][3]이다. Jini 클라이언트, 서비스가 Jini 네트워크에 참가하면 멀티캐스트 요청, 멀티캐스트 알림, 유니캐스트 발견 프로토콜[4]을 이용하여 조회 서비스를 찾는데 이 과정을 발견 프로세스라 한다. 조회 서비스를 발견한 후에 Jini 서비스는 자신이 제공할 서비스를 서비스 프락시 형태로 조회 서비스에 등록을 하게 된다. Jini 클라이언트는 자신이 이용할 서비스를 제공하는 서비스 프락시를 조회 서비스에서 검색하여 다운로드 후 서비스 프락시를 이용하여 서비스 제공자에게 서비스를 받게 된다. 이 때 서비스 프락시는 임대를 통해 사용기간을 연장할 수 있다. Jini의 장점으로는 서비스 이용자가 사전에 서비스 제공자의 IP나 정보를 알고 있지 않더라도 Jini 네트워크에 참가하면 바로 서비스를 이용할 수 있다는 것이다. 그리고 자바로 구현되어 있으므로 JVM이 설치된 모든 플랫폼에서 이용할 수 있다. 그림 1은 Jini 클라이언트인 디지털 카메라가 저장된 사진 이미지를 Jini 서비스인 프린터로 출력할 때의 과정을 나타낸다.

3. 최적의 컴퓨팅 리소스 매칭 시스템

3.1 시스템 동작과정

본 논문에서 제안한 시스템은 서버의 리소스를 이용하려는 클라이언트와 클라이언트에게 리소스를 할당해 줄 서버는 Agent Manager에게 자신들의 정보를 주기 위한 Interface가 미리 정의 되어있고, 클라이언트는 어떤 서버와도 연결되어 그 서버의 리소스를 이용할 수 있다고 가정한다. 그리고 클라이언트는 미

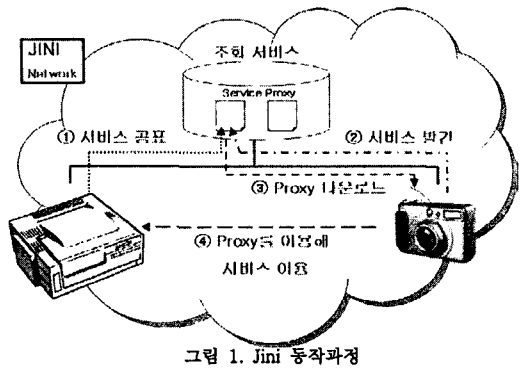

그림 1. Jini 동작과정

리 프로그램 제작사로부터 실행할 프로그램이 서버의 CPU, Memory, Network 리소스들을 어느 정도 사용할 것인지를 나타내는 중요도를 받아서 사용하고, 1Round안에서는 서버는 한 클라이언트와만 연결된다고 가정한다. 하지만 다음 Round 에서는 이미 연결된 서버도 다른 클라이언트와 연결될 수 있다.

우선, 클라이언트나 서버가 Jini 네트워크에 참여하면 Agent Manager가 클라이언트나 서버의 정보를 알 수 있게 자신의 프락시를 조회 서비스에 등록을 한다. 클라이언트는 서버에서 수행하려는 프로그램이 서버의 어떤 리소스를 많이 사용할지를 CPU, Memory, Network 항목으로 각각 1에서 5까지의 중요도를 지니고 있다. 가령, 서버에서 수학적 계산이 많은 프로그램을 이용할 시에는 CPU의 중요도가 높고, 프로그램을 수행하는데 있어서 클라이언트가 지속적으로 데이터를 서버로 보내야 한다면 Network의 중요도가 높다. 서버는 프락시를 통해 Agent Manager가 현재 서버의 리소스 상황을 알 수 있게 한다. 리소스 속성으로는 CPU성능, 이용 가능한 Memory량, Network 접속속도가 있다. Agent Manager는 일정한 간격으로 Jini 네트워크에 참여한 클라이언트와 서버의 정보를 수집한다. 수집한 정보를 이용하여 다중속성을 이용한 대표적인 협상 시스템인 MAUT(Multi-Attribute Utility Theory)의 유틸리티 함수[5]를 계산하고, 유틸리티 값을 이용하여 모든 클라이언트와 서버간의 Weight를 결정한다. 그리고 Weight를 이용하여 Maximum-Weight Matching[6]을 이용하여 클라이언트와 서버의 연결에 있어서 전체적으로 만족도가 높은 매칭을 결정한다. 마지막으로 매칭 정보를 이용하여 클라이언트가 매칭된 서버를 이용할 수 있도록 매칭된 서버의 프락시를 다운받을 수 있게
하여 서버의 리소스를 이용할 수 있도록 한다.

3.2 클라이언트와 서버의 매칭

클라이언트와 서버 간에 매칭을 하기 위한 Weight를 계산하기 위해 클라이언트와 서버가 갖는 속성은 다음과 같다. 클라이언트는 서버에서 수행할 프로그램의 종류에 따라 CPU, Memory, Network 에 대한 1~5까지의 중요도를 속성으로 갖는다. 그리고 실제 컴퓨팅 환경과 비슷하게 하기 위해 서버의 리소스를 점유할 Round를 1~5까지 가진다. 본 논문에서는 1Round를 Agent Manager를 통해 한 번의 매칭이 끝날 때로 정의한다. 서버는 자신이 보유한 CPU 성능, 현재 이용 가능한

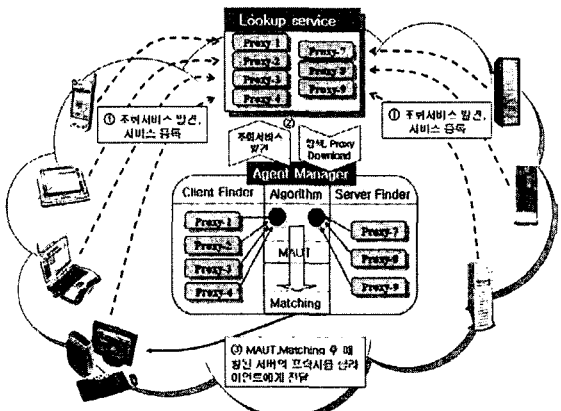

그림 2. Agent Manager를 이용한 최적의 리소스 매칭 시스템

Memory 공간, Network에 연결된 회선속도를 속성으로 갖는다.

▸ 1단계 : Evaluation Function

Agent Manager는 서버에서 수집한 정보 중에 CPU 성능, 회선 속도를 이용해 모든 서버 중에 각각의 서버가 클라이언트에게 어느 정도의 메리트가 있는지를 계산하기 위해 CPU성능, 회선 속도에 대한 Evaluation function (E_{cpu}, E_{net})을 다음 수식과 같이 계산한다.

$$E_{cpu_k}=\frac{Server_{cpu_k}}{Server_{cpu_1}+.....+Server_{cpu_n}}=\frac{Server_{cpu_k}}{\sum_{i=1}^{n}Server_{cpu_i}}\ (1\le k\le n)$$

$$E_{net_k}=\frac{Server_{net_k}}{Server_{net_1}+.....+Server_{net_n}}=\frac{Server_{net_k}}{\sum_{i=1}^{n}Server_{net_i}}\ (1\le k\le n)$$

- $Server_{cpu_k}$: k번째 서버의 CPU성능
- $Server_{net_k}$: k번째 $network$ 회선속도

Memory에 대한 Evaluation function (E_{mem})은 현재 서버에서 이용할 수 있는 메모리 공간으로 부터 클라이언트에서 제시한 최소 필요 메모리 공간을 뺀 값이 크면 클수록 좋은 값을 갖는다.

$$E_{mem_k}=\begin{cases}\dfrac{Server_{mem_k}-Client_{mem[\min]}}{Client_{mem[\max]}-Client_{mem[\min]}} & (Client_{mem[\min]}\le Server_{mem_k}\le Client_{mem[\max]})\\ 0 & (Client_{mem[\min]}\le Server_{mem_k})\\ 1 & (Server_{mem_k}<Client_{mem[\min]})\end{cases}$$

- $Server_{mem_k}$: k번째 서버의 이용 가능한 $Memory$량
- $Client_{mem[\min]}$: i번째 클라이언트의 최소 $Memory$ 요구량
- $Client_{mem[\max]}$: i번째 클라이언트의 권장 $Memory$ 요구량

▸ 2단계 : Weight Evaluation

Agent Manager는 각각의 클라이언트에서 수집한 CPU, Memory, Network에 대한 중요도(Cpu_M, $Memory_M$, $Network_M$)를 이용해 서버의 자원에 대한 가중치 (Cpu_{weight}, $Memory_{weight}$, $Network_{weight}$)를 계산한다. 수식은 다음과 같다.

$$Cpu_{weight_k}=\frac{Cpu_{M_k}}{Cpu_{M_k}+Memory_{M_k}+\neq twork_{M_k}}$$

$$Memory_{weight_k} = \frac{Memory_{M_k}}{Cpu_{M_k} + Memory_{M_k} + \neq twork_{M_k}}$$

$$\neq twork_{weight_k} = \frac{\neq twork_{M_k}}{Cpu_{M_k} + Memory_{M_k} + \neq twork_{M_k}}$$

▸ **3단계 : MAUT를 이용한 Matching Weight 계산**

위에서 계산한 평가함수와 가중치를 이용하여 MAUT를 이용하여 Matching Weight(MW)를 계산한다. 이 값이 모든 클라이언트와 서버 간 연결의 만족도가 된다. 수식은 다음과 같다.

$$MW_{ik} = (Cpu_{weight_i} \times E_{cpu_k}) + (Memory_{weight_i} \times E_{mem_k}) + (\neq twork_{weight_i} \times E_{\neq t_k})$$

▸ **4단계 : Maximum Weight Matching**

1~3단계 과정을 통해 각각의 클라이언트는 모든 각각의 서버에 대한 만족도를 갖는다. Maximum Weight Matching을 이용하여 1Round에서 서버와 연결되는 클라이언트들의 모든 만족도의 합이 가장 큰 클라이언트와 서버간의 M:N 연결을 찾아내고, 클라이언트에게 연결될 서버를 알려주어 서버의 프락시를 다운받아 서버의 리소스를 이용한다.

4. 실험 결과 및 향후 계획

본 논문에서는 Jini 환경에서 클라이언트와 서버를 실제 호스트가 아닌 중요도나 서버의 리소스 상황을 속성으로 부여한 쓰레드로 생성하여 구현하였다. 클라이언트는 프로그램에 따라 다르게 부여될 중요도와 서버의 리소스를 일정 기간 동안 점유할 라운드를 일정 범위에서 랜덤으로 생성하였고, 클라이언트의 수는 제한된 수내에서 동적으로 그 수가 변하도록 하였다. 서버의 리소스 또한 일정 범위에서 랜덤으로 생성하였고, 클라이언트가 서버의 리소스를 점유한 만큼 리소스가 감소되도록 구현하였다. Jini 환경에서 제안한 Agent Manager를 사용했을 때와 사용하지 않고 클라이언트가 아무 서버의 프락시를 이용하여 연결될 때의 전체적인 만족도를 비교하였다. 구현환경으로는 썬마이크로시스템의 Jini 2.0 버전을 이용하였다. 실험에서 사용할 각 클라이언트의 속성은 표 1의 범위에서 랜덤으로 생성하였고, 각 서버의 속성은 표 2와 같다. 향후, Jini를 이용하여 유비쿼터스 환경에 적합한 Agent Manager로 확장할 계획이다.

표 1. 클라이언트가 서버에서 이용할 프로그램의 속성

클라이언트가 서버에서 이용할 프로그램의 속성	
CPU 중요도	1~5
Memory 중요도	1~5
Network 중요도	1~5
최소 필요 메모리	10~35 Mbytes
권장 필요 메모리	40~100 Mbytes
라운드	1~5

표 2. 서버의 리소스 속성

서버의 리소스 속성			
	CPU	Memory	Network
	(MHz)	(Mbytes)	(Mbps)
서버1	600	512	100
서버2	800	1000	10
서버3	600	512	100
서버4	1200	1000	100
서버5	800	512	10
서버6	1200	256	10
서버7	900	256	100

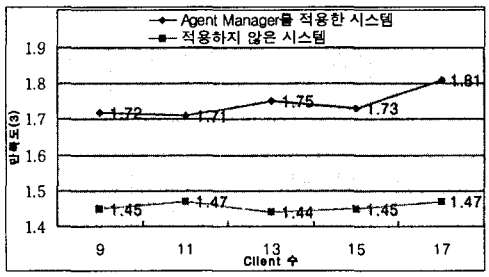

그림 3. 서버의 개수가 3개일 때

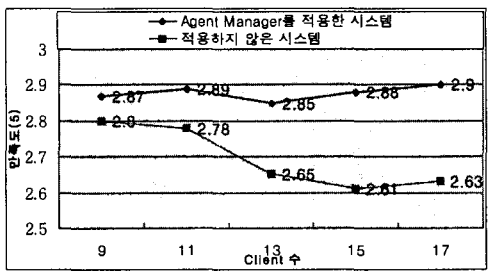

그림 4. 서버의 개수가 5개일 때

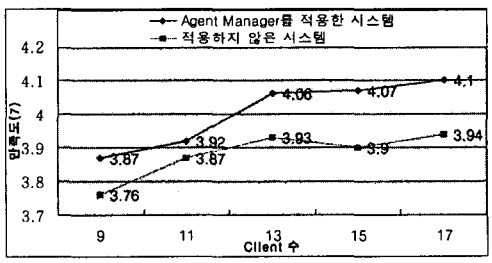

그림 5. 서버의 개수가 7개일 때

참고문헌

[1] Jini, http://www.jini.org.

[2] Jini, http://www.sun.com.

[3] 한상숙, 은성배, 김철민. "UPnP-to-Jini 서비스의 설계 및 구현," 정보처리학회논문지A, 제11-A권 제1호, 2004.

[4] W. K. Edwards, "Core Jini", *Prentice-Hall*, Inc. 1999.

[5] Mihai Barbuceanu, Wai-Kau Lo, "A Multi-Attribute Utility Negotiation for Electronic Commerce," *AMEC2000*, pp.15-30, 2001.

[6] James R. Evans, Edward Minieka, "OPTIMAZATION ALGORITHMS FOR NETWORKS AND GRAPHS", Second edition, *DEKKER*, Inc. 1992.