분산 서버 클러스터 시스템의 부하 분산 및 성능 분석 시뮬레이션

최은미*, 이원규*

Workload Distribution and Performance Analysis Simulation for a Distributed Server Cluster System

Eunmi Choi, Wonq Lee

Abstract

A distributed server cluster system is a cost-effective system to provide a service application for clients with reliable, scalable, available, and fault-tolerant features. In order to provide high quality services, it is necessary to evaluate service performances, tune the server system, and analyze performances. In this paper, we propose a simulator to generate workloads based on statistic configuration according to estimated application traffics, apply workload scheduling algorithms, and evaluate the simulation results. We introduce the simulator design modelling and architecture. By using flexible parameters, the simulator is able to generate various patterns of workloads with different statistics, and configure system environments such as the number of server nodes, system resources considered, and their capacities. With this simulator, we introduce two scenarios: one is to find appropriate thresholds for the best performance of cluster system, and the other is to find the suitable scheduling algorithm for workload characteristics of service applications.

Key Words: Workload generation, performance simulation, performance analysis, simulator modeling, simulation

* 한동대학교 전산전자공학부

1.. 서론

인터넷 이용의 확산으로 다양한 어플리케이션 서비스를 제공하는 분산 서버 클러스터 시스템은 서버단의 신뢰성과 확장성, 가용성을 제공하는 비용 대 효과(cost-effective)를 높여주는 서버 단 시스템으로 사용되고 있다. 실제 분산 서버 클러스터 시스템에서의 시스템 성능 평가와 어플리케이션에 따른 부하의 분석을 할 필요성이 있으나. 다양한 부하 생성과 부하 분산 알고리즘의 실제 시스템에로의 자유로운 적용에 한계가 있으므로 여러 가지 상황을 분석하는 데에는 어려움이 있다. 또한, 특정 어플리케이션을 서비스하는 서버 클러스터 시스템에서의 최적합한 성능을 취득할 수 있는 시스템 상의 configuration 값의 설정은 실제로 시스템의 성능을 심각하게 좌우할 수 있게 된다. 이러한 값들은 시스템의 주어진 자원들의 한계, 어플리케이션의 특성, 부하 특성에 따른 속성, 서버 간의 부하 분산 알고리즘의 적합성 등으로 정하여 질 수가 있다.

본 논문에서는, 분산 서버 클러스터 시스템에서의 부하 생성 및 서버로의 부하 분산, 그리고 그 결과의 성능을 분석하는 시뮬레이터를 모델링하고 개발을 하였다. 시뮬레이터의 제안 단계로, 클러스터 시스템 시뮬레이터의 설계 모델링을 소개하여준다. 서버 노드 단에 agent 로 역할을 담당하는 NodeService, 서버단 앞단에 위치하여 클러스터 시스템의 들어오는 Traffic을 받고 부하 분산의 역할을 맡는 TrafficManager, 클라이언트 단의 다양한 부하 생성을 할 수 있는 역할을 담당하는 TrafficGenerator, 시뮬레이션을 돕는 Clock과 Log, Parameter 관리자와 GUI 관련 모듈들을 소개한다. 이 시뮬레이터를 통하여 여러 통계적인 부하 상황을 조절할 수 있으며, 분산 알고리즘들도 다양하게 적용을 할 수 있게 하였다. 시스템의 특성에 따라서, 서버의 개수와 서버의 용량도 자원별로 조절을 할 수 있도록 하였으며, 서버들에게 부하를 주는 클라이언트의 개별 부하의 특성도 여러 자원들의 요청량을 합하여 생성할 수 있도록 하였다.

이 시뮬레이터를 사용하여 두 가지 관점을 시뮬레이션 하였다. 첫째는, 부하 분산을 적응력(adaptive) 있게 적용시키기 위한 상한선과 하한선의 한계점 값 (threshold)을 각종 클러스터 어플리케이션의 부하 특성에 따라서 정할 수 있게 된다. 이 한계값은 실제 클러스터 시스템에 적용시킬 수 있는 configuration 값으로 직접 설치가 가능하게 되어 시스템에 적합한 성능을 줄 수 있게 된다. 둘째는, 시뮬레이션을 통하여 부하의 특성에 따른 가장 적절한 부하 분산 알고리즘을 찾을 수 있게 된다. 부하의 서버의 할당 문제는 NP-Complete 문제이므로, 이 문제를 시뮬레이터를 통하여 적절한 approximation 접근 방법을 찾게 되는 결과를 낳게 된다. 시뮬레이터의 사용자의 여러 가지 편리한 인터페이스와 결과 분석을 이 두 가지 시나리오를 통하여 보인다.

다음 장에서는 시뮬레이터의 설계 구조도와 설명을 하며, 3장과 4장은 시뮬레이션 결과를 시나리오에 따라서 보이며 분석을 한다. 5장에서 결론을 맺는다.

2.. 시뮬레이터의 설계 구조도

시뮬레이터는 기능에 따라 크게 세개 패키지 부분으로 나뉘어지며, 이 외에도 몇 가지 시뮬레이션에 필요한 공통 클래스들이 존재한다.

1) **trafficManager 패키지** : trafficGenerator에서 생성된 workload를 받아서 부하분산 알고리즘에 따라 노드에 전달하는 역할을 한다. 적응력 있는 알고리즘을 사용하는 경우에는 각 노드의 상태를 변화시키는 시기를 알려 주는 일도 한다.

2)**trafficGenerator 패키지** : 부하 설정에 맞

는 workload를 생성하는 역할을 담당한다.

3) **nodeService 패키지** : TM으로부터 받은 workload내의 load들을 그에 맞는 자원에 할당해서 그 자원을 소비하는 실제 서버 노드의 역할을 한다.
4) **SimulatorManager** : 시뮬레이션에 필요한 인스턴스를 생성하고, 시뮬레이션의 전체적인 동작을 제어한다.
5) **TrafficGeneratorParameter** : workload를 생성의 필요한 파라미터들을 가지고 있다.
6) **NodeParameter** : 각 가상 노드(virtual node)들에 필요한 파라미터들을 담고있다.
7) **Log** : 시뮬레이션의 결과를 분석, 저장한다. 저장되는 결과로는 전체 요청 처리수, 실패한 처리수, 각 노드별 요청 처리수, 각 노드별 과부화 비율 등이 있다. 통합적인 결과는 생성시 시스템 시간을 참조하여 특정 이름을 갖는 엑셀파일로 저장하며, 각 가상 노드들의 자원에 대한 시뮬레이션 동안의 상태를 기록하는 엑셀 파일이 상기 파일명에 노드 번호를 붙인 이름으로 저장한다.
8) **Clock** : 시뮬레이션의 논리적 시간을 의미하는 globalTime이 static변수로 선언되어 있고, 이 변수는 전체 시뮬레이션 시간, 각 노드에서 사용한 자원의 할당, 회복 시기 등을 계산할 때 참조된다.

2.1 trafficManager 패키지의 구조

trafficManager 패키지는 그림 2의 설계 구조를 가지고 있다. 사용하는 부하분산 알고리즘들을 용이하게 추가시킬 수 있도록 factory design pattern을 사용하여 구현되었다.

적응력(adaptive)있는 부하분산 알고리즘을 사용하는 경우, 한계값을 이용하여 각 가상 노드는 과부화 상태를 선언하여 과부화 상태를 벗어날 때까지 서비스를 중지할 수 있는데, 이 때 각 노드의 상태를 변화시키는 시기를 알려주는 역할도 한다.

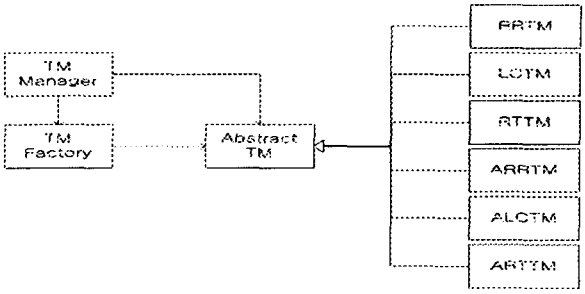

그림 1. trafficManger 패키지 구조도

1) **TMManager** : 시뮬레이션 초기 단계에서 TMFactory에 TM을 생성한다. 시뮬레이션 동안에 traffic generator로부터 workload를 가져와서 생성된 TM에 전달한다.
2) **TMFactory**: 현재 시뮬레이션에 맞는 TM을 생성한다.
3) **AbstractTM** : TM으로 전달된 workload의 해제 시간을 계산해서 그 workload에 기록하고, 모든 가상 노드들의 상태들을 점검, 부하분산 알고리즘에 의해 선택된 가상 노드에 workload를 전달하는 동작들이 구현되어 있다. workload를 처리할 노드를 선택하는 getNextNode() 메서드는 이 추상클래스를 상속받은 자식 클래스에서 각 부하분산 알고리즘에 맞게 구현되도록 하였다.
4) **ConcreteTM**: 각종 부하 분산 알고리즘이 구현되어 있다. Round Robin(RR), Least Connection(LC), Response Time(RT), Adaptive-RR, -LC, -RT이 구현이 되었다.

2.2 trafficGenerator 패키지의 구조

서비스의 요청을 논리적으로 표현한 workload를 만드는 역할을 한다. 각 요청은 시스템 서버의 자원을 사용하며, 실제 서버는 이 요청을 처리하기위해 여러 자원을 사용한다. 그래서 하나의 workload는 여러개의 load로 구성되어 있다. 각 load는 CPU, Memory, Network와 같은 시스템 자원을 가지며, 각기

다르게 통계적 부하상황을 조절할 수 있다.

1) **TrafficGenerator** : 각 resource generator로부터 생성된 load를 얻어와서 하나의 workload를 만든다.

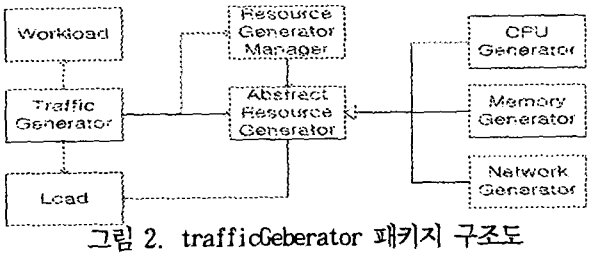

그림 2. trafficGeberator 패키지 구조도

2) **Load** : 한가지의 자원을 사용한다. 사용량, 시스템에 도착 시간, 자원 점유시간 등의 정보를 가지고 있다.
3) **Workload** : 시뮬레이션의 설정에 맞는 개수의 load를 Vector형태로 가지고 있다.
4) **ResourceGeneratorManager** : 각 자원의 generator를 생성해서 traffic generator에 전달한다.
5) **AbstractResourceGenerator** : 사용자의 입력에 맞게 원하는 통계적인 값들을 생성한다. 이 generator가 생성하는 분포는 Normal 분포, Exponential 분포, Uniform 분포가 있다. 각 분포에 따른 값을 생성할 때, 임의의 난수를 생성하거나, 종자값(seed)를 이용해서 같은 파라미터에 대해 같은 값들이 생성되게 할 수도 있다.
6) **ConcreteGenerator**: 실제 자원인 CPU, memory, network load를 생성한다.

2.3 nodeService 패키지의 구조

실제 시스템에는 CPU, memory, network bandwidth등 서버의 성능에 영향을 미치는 여러 가지 자원이 있다. 본 simulator에서 서버 노드 agent역할을 하는 가상 노드는 각 load들에 대응하는 자원을 배열 형태로 가지고 있다.

1) **Resource** : 하나의 자원을 의미한다. 이 클래스의 멤버로는 용량(capacity), 상한선, 하한선 한계점(threshold)이 있으며, 자원을 사용한 load가 자원을 반환하는 시점을 확인하여 자원을 회복하는 행동과 이 자원의 상태를 검사하는 행동을 한다.
2) **VirtualNode** : 여러개의 자원을 배열 형태로 가지고 있으며, 클라이언트의 요청에 대한 서비스를 제공하는 서버 노드 역할을 한다. 이 클래스는 노드의 상태, 과부화 횟수, 노드 ID등이 있으며, 들어온 workload안의 load들을 각 resource에 맞게 할당하는 행동과 노드 내 각 자원의 상태를 검사해서 노드의 상태를 바꾸는 행동 등이 있다.

3.. 시뮬레이터를 통한 부하생성 및 분산

시뮬레이터를 이용해서 부하를 생성하고, 분산 처리하는 시뮬레이션의 과정을 보이겠다.

3.1 부하 생성 단계

각 자원별 생성시킬 load의 자원 소모량(amount)과 생성된 load가 가상 노드의 자원을 소모하는 기간(duration time)을 알맞은 통계적 상황에 따라 생성할 수 있도록 파라미터 값을 할당하고 조정하여 준다.

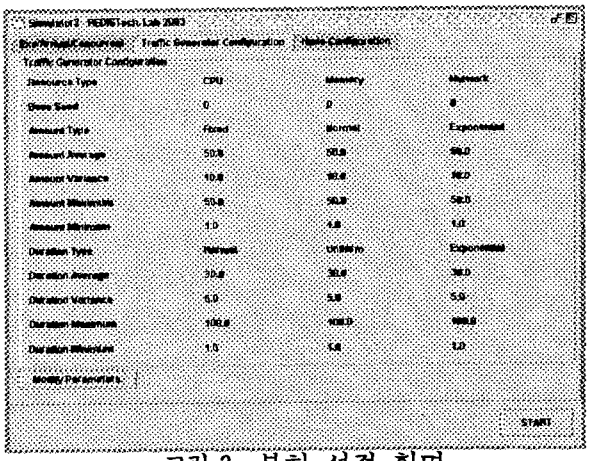

그림 3. 부하 설정 화면

3.2 시뮬레이션 환경 설정

전체 시뮬레이션의 환경을 설정한다. 해당 시뮬레이션에서 고려하는 가상 노드의 개수, 시뮬레이션 시간, 각 노드 내 자원의 수, 부하분산 알고리즘 등을 설정한다.

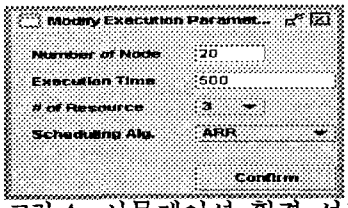

그림 4. 시뮬레이션 환경 설정

3.3 가상 노드 정보 설정

가상 노드 내 각 자원의 용량(capacity)과 상하위 한계점(threshold)을 설정한다(그림5). 사용자의 설정에 따라 동종 서버 혹은 이종 서버의 시스템을 설정 할 수 있다(그림6).

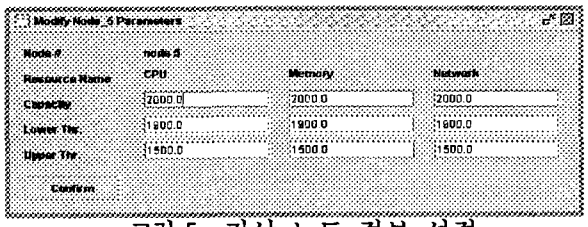

그림 5. 가상 노드 정보 설정

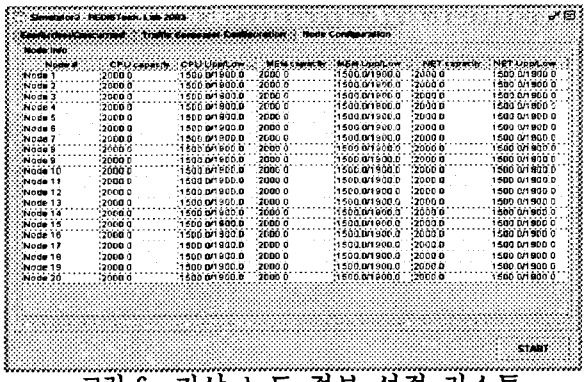

그림 6. 가상 노드 정보 설정 리스트

4.. 시뮬레이션 Scenario

서버 클러스터 시스템의 부하분산과 그 성능 분석을 두 가지의 시나리오를 따라 시뮬레이션 결과를 보이겠다.

4.1 최적 Threshold 결정

표1과 같은 가상의 클라이언트 요청과 이를 처리하는 분산 클러스터 시스템 상에서 적응력 있는 부하분산 알고리즘을 최적화 하기위해서 한계값(threshold)를 결정하는 실험을 하였다.

# of node		5 (Homogeneous)	
node capacity		2000	
execution time		2000ms	
Scheduling Alg.		ARR	
Arrival Time		Concurrent Load	
Type	Normal	Type	Uniform
avg	20	max	100
var	5	min	1
max	20		
Amount		Concurrent Load	
Type	Uniform	Type	Uniform
max	200	max	500
min	1	min	1

표 1 한계값 최적화 실험 환경

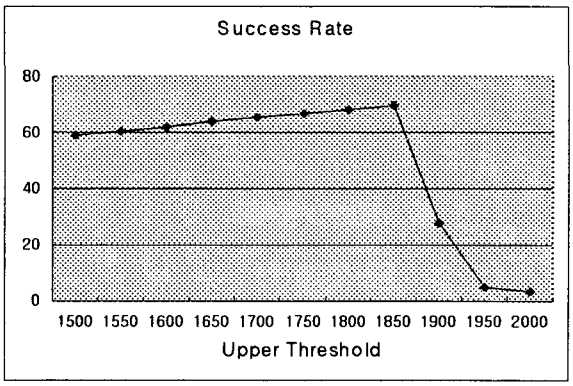

그림 7. 한계값 설정에 따른 성공률 결과

그림 7의 그래프는 하한 한계값이 1500일때, 상한 한계값을 1500부터 2000까지 늘였을때 총 요청수에 대한 처리의 성공률(Success Rate: 단위 %)이 1850부근에서 가장 높은 것을 보여준다. 그러므로 현재 환경에서 상한 한계값을 1850으로 하는 것이 시스템의 성능을 최적으로 사용할 수 있다고 판단할 수 있다.

4.2 부하분산 알고리즘의 성능 비교

부하의 유형을 고정(fixed) 부하와 임의(random)부하로 나누어서 부하분산 알고리즘의 성능을 실험 비교하였다. 각 부하의 환경은 표 2와 3에서 각각 나타나 있다. 고정 부하는 요청되는 작업의 크기와 자원 사용시간 모두 부하가 동일하며, 임의 부하 상황은 Random 한 자원의 크기와 사용시간을 생성 한다.

Arrival Time		Concurrent Load	
Type	Fixed	Type	Fixed
value	5	value	15
Amount		Duration Time	
Type	Fixed	Type	Fixed
value	4.4	value	750

표 2 고정 부하 실험 환경

Arrival Time		Concurrent Load	
Type	Normal	Type	Normal
avg	6.4	avg	15
var	0.5	var	0.5
max	50	max	50
Amount		Duration Time	
Type	Uniform	Type	Uniform
max	400	max	200
min	1	min	1

표 3 임의 부하 실험 환경

알고리즘	부하종류	
	Fixed	Random
RR	37.08%	17.23%
LC	37.08%	44.87%
RT	37.08%	44.99%
ARR	93.63%	97.01%
ALC	93.63%	96.99%
ART	93.63%	96.97%

표 4 부하 분산 알고리즘 실험 결과
(단위: 총 요청에 대한 성공률 %)

고정 부하와 임의 부하 상황에서 서버가 과부하로 다운될 정도의 부하를 보내는 경우에 일반 부하 분산 알고리즘은 표 4와 같이 저조한 결과를 보인다. 특별히 임의 부하 상황에서 RR 경우는 17%의 성공률을 보이게 된다. 그러나, 서버의 상태를 고려하여 부하를 할당하는 알고리즘들이 보다 좋은 성능을 나타내는 결과를 보였다. 다른 흥미로운 점은 ARR과 ALC, ART의 결과 중 미소한 차이로 ARR 이 좋은 결과를 낳았다. 이는 부하 분산 시 적절한 Adaptive mechanism 만을 적용을 하게 되면, First-fit 의 특성을 가진 ARR 이 Best-fit 의 특성을 가진 ALC, ART 보다 성능이 우수할 수 있다는 결론을 보였다.

5.. 결론

분산 서버 클러스터 시스템에서의 부하 생성 및 서버로의 부하 분산, 결과의 성능을 분석하는 시뮬레이터를 모델링하고 개발을 하였다. 여러 통계적인 부하 상황을 조절하며, 분산 알고리즘들도 다양하게 적용을 할 수 있으며, 서버에 대한 각종 파라미터 값들을 자원별로 조절을 할 수 있다. 이 시뮬레이터를 이용하여 한계값을 결정하는 시나리오와 각 부하 분산 알고리즘들의 실험 결과 비교를 하였다.

참고문헌

[1] Andrew S. Tanenbarum, Maarten van Steen, Distributed Systems principles and Paradigms , Prentice Hall, 2002

[2] Mark Grand, "Patterns in Java, Volume1, A Catalog of Reusable Design Patterns Illustrated with UML", John Wiley & Sons, INC., 1998.

[3] Raj Jain, "The Art of Computer Systems Performance Analysis", Wiley, 1991

[4] Wensong Zhang, "Linux Virtual Server for Scalable Network Services", Ottawa Linux Symposium 2000, http://www.linuxvirtualserver.org/ols/clvs.ps.gz

[5] Li Xiao, "Dynamic Cluster Resource Allocations for Jobs with Known and

Unknown Memory Demands", IEEE Transaction on Parallel and Distributed Systems, Vol.13, No.3, 2002

[6] 임유진, 이원규, 최은미, "부하 특성에 따른 분산 스케줄링 알고리즘의 성능 평가 및 비교," 춘계정보처리학회, pp.137-140, 2003

● 저자소개 ●

최은미

1988 고려대학교 이과대학 전산과학과 학사

1991 미국 미시간 주립대학 Computer Science 공학 석사

1997 미국 미시간 주립대학 Computer Science 공학 박사

1998~현재 한동대학교 전산전자공학부 교수

관심분야: 분산 시스템, 클러스터 시스템, 분산 병렬 처리, 분산객체 모델링, 보안 프로토콜

e-mail : emchoi@handong.edu

이원규

1997~현재 한동대학교 전산전자공학부 학사 과정

2001~현재 한동대학교 RedisTec 연구원

2001~2002 (주)Two-way communication 인턴쉽

관심분야: 분산 시스템

e-mail : wonq_lee@hotmail.com