대규모 다중 서버 분산 가상 환경 시스템을 위한 확장성 있는 로드 밸런싱 기법

이경민[0] 이동만 김성훈 장소희 박은광 현순주
한국정보통신대학원대학교
(kmlee[0], dlee, kimsh, sjang, pek, shyun)@icu.ac.kr

A Scalable Load Balancing Scheme for Large-Scale Multi-Server Distributed Virtual Environment (DVE) Systems

Kyungmin Lee[0] Dongman Lee Sunghoon Kim Sohee Jang Eunkwang Park Soonju Hyun
Information and Communications University

요 약

분산 가상 환경(Distributed Virtual Environment, DVE) 시스템의 규모가 커지면서 확장성(scalability)이 중요한 이슈가 되고 있다. 시스템의 확장성을 높이기 위해 제안된 다중 서버 DVE 시스템에서 사용자들의 집단적인 움직임 특성은 서버간 작업부하의 불균형을 초래한다. 이는 전체 시스템의 효용을 떨어뜨리고 다중 서버의 효과를 감소시킨다. 결과적으로 시스템의 확장성이 저하된다. 이러한 문제를 해결하기 위해 기존에 제안된 로드 밸런싱 기법은 크게 로컬 기법과 글로벌 기법으로 구분할 수 있다. 로컬 기법은 적은 비용을 갖지만 작업부하의 분포가 한쪽으로 치우쳐 있을 경우 성능이 낮다. 글로벌 기법은 좋은 성능을 보여주는 반면, 서버의 수가 증가하면 비용이 크게 증가한다. 본 논문은 서버들의 작업부하 분포에 동적으로 적응(adaptation)하는, 대규모 다중 서버 DVE 시스템을 위한 확장성 있는 로드 밸런싱 기법을 제안한다. 제안하는 기법에서 과부하된 서버는 그 주변 서버들의 작업부하가 적을 경우 로컬 확산 (*Local Diffusion*) 로드 밸런싱을 수행하고, 그렇지 않을 경우 제한적 글로벌 (*Limited Scope Global*) 로드 밸런싱을 수행한다. 시뮬레이션 결과를 통해 제안하는 기법이 효율(efficiency)의 측면에서 기존의 로컬 기법과 글로벌 기법보다 각각 14%, 3%가 우수함을 보인다. 또한, 제안하는 기법은 서버의 수가 증가함에 따라 효율이 감소하는 추세가 기존 기법들보다 완만함을 보인다.

1. 서론

분산 가상 환경 (Distributed Virtual Environment, DVE) 시스템의 규모가 커지면서 확장성(scalability) - 사용자들의 상호작용 성능(interactive performance)을 떨어뜨리지 않으면서 많은 수의 동시 사용자를 처리하는 것이 중요한 이슈가 되고 있다 [1]. 대규모 DVE 시스템의 확장성을 높이기 위해 다중 서버 기법 (multiple server architecture)을 이용한 DVE 시스템 - DIVE[2], RING[3], BrickNet[4]이 제안되었다. 다중 서버 기법은 가상 세계(virtual world)를 여러 영역(region)으로 분할하여 각 서버들이 나누어 관리함으로써 하나의 서버가 처리하는 작업부하를 줄이고 시스템의 확장성을 향상시킨다. 하지만, 가상 세계 내에서의 사용자들의 집단적인(aggregated) 움직임 특성[5]은 서버간 작업부하의 불균형을 초래한다. 많은 사용자들이 몰려있는 영역을 관리하는 서버는 많은 작업부하를 처리해야 하는 반면, 사용자들이 상대적으로 적은 영역을 관리하는 서버는 적은 작업부하를 처리한다. 이는 전체 시스템의 효용(utilization)을 떨어뜨리고 다중 서버의 효과를 감소시킨다. 결과적으로 시스템의 확장성이 저하된다. 이 문제의 해결을 위해 로드 밸런싱 (load balancing) - 작업부하가 많은 서버의 작업을 작업부하가 적은 서버로 이동시켜 서버간 작업부하의 불균형을 감소시키는 기법이 제안되었다.

기존에 제안된 다중 서버 DVE 시스템을 위한 로드 밸런싱 기법은 크게 로컬(local) 기법[6,7]과 글로벌(global) 기법[8]으로 구분할 수 있다. 로컬 기법은 오직 과부하된 서버만이 그 주변 서버(neighbor server)들과 함께 로드 밸런싱을 수행한다. 반면, 글로벌 기법은 시스템의 모든 서버가 로드 밸런싱에 함께 참여한다. 로컬 기법은 과부하된 서버의 주변 서버들의 정보만을 필요로 하고 또한 적은 양의 작업부하 이동(workload migration)을 초래하므로 적은 비용을 갖는다. 그러나, 작업부하의 분포가 한쪽으로 크게 치우쳐 있을 경우, 로컬 기법은 낮은 성능을 보인다. 즉, 서버간 작업부하의 불균형을 제대로 해소하지 못한다. 이와 달리, 글로벌 기법은 모든 서버가 로드 밸런싱에 참여하므로 좋은 성능을 보이지만 전체 서버의 수가 많아지면 비용이 크게 증가한다.

본 논문은 서버들의 작업부하 분포에 동적으로 적응(adaptation)하는, 대규모 다중 서버 DVE 시스템을 위한 확장성 있는 로드 밸런싱 기법을 제안한다. 과부하된 서버의 과잉 작업부하(excess workload)를 주변 서버들이 충분히 받을 수 있을 경우에는 로컬 확산(*Local Diffusion, LD*) 로드 밸런싱을 수행하고, 그렇지 않을 경우에는 제한적 글로벌(*Limited Scope Global, LSG*) 로드 밸런싱을 수행한다. 제한적 글로벌 로드 밸런싱에 참여하는 서버의 범위 역시 서버들의 작업부하 분포에 따라 동적으로 결정된다. 제안하는 기법은 참여하는 모든 서버의 작업부하를 미리 정의된 한계치(threshold) 이내로 유지한다. 시뮬레이션 결과를 통해 서버의 수가 100개이고 작업부하의 분포가 한쪽으로 크게 치우쳐 있는 경우에, 제안하는 기법이 작업부하 분산 감소율(workload variance reduction ratio)의 측면에서 로컬 기법보다 대략 50%가 우수하며, 작업부하 이동 비율(workload migration ratio)의 측면에서 글로벌 기법보다 17%가 우수함을 알 수 있다. 전체적으로, 제안하는 기법은 효율(efficiency)의 측면에서 로컬 기법과 글로벌 기법보다 각각 14%, 3%가 우수하다. 또한 제안하는 기법은 서버가 증가함에 따라 효율이 감소하는 추세가 기존 기법들보다 완만함을 보여준다. 이를 통해 제안하는 기법이 대규모 다중 서버 DVE 시스템을 위한 확장성 있는 로드 밸런싱 기법임을 알 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 기존의 다중 서버 DVE 시스템을 위한 로드 밸런싱 기법에 대해 기술한다. 3장에서는 제안하는 기법을 자세히 기술하고, 기존 기법들과의 성능 비교 결과를 4장에서 기술한다. 마지막으로, 5장에서 결론을 맺는다.

2. 관련 연구

기존에 제안된 다중 서버 DVE 시스템을 위한 로드 밸런싱 기법은 크게 로컬 기법과 글로벌 기법으로 구분할 수 있다. [6], [7]과 같은 로컬 기법에서 과부하된 서버는 오직 그 주변 서버들

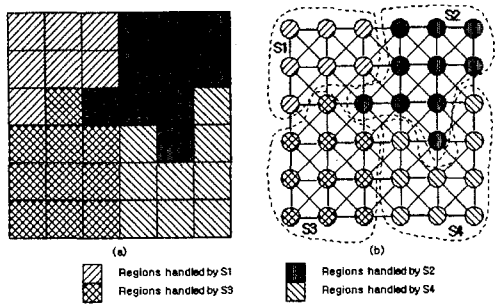


그림 1. (a) 간단한 DVE 시스템 (b) 해당 그래프 모델

에게 과잉 작업부하를 전달한다. 로컬 기법은 분산된 방식으로 로드 밸런싱을 수행하기 때문에 시스템의 서버 수가 증가하더라도 적은 비용을 갖는다. 그러나, 작업 부하의 분포가 한쪽으로 치우쳐 있는 경우 – 즉, 많은 사용자들이 가상 세계의 작은 지역에 몰려있는 경우 – 낮은 성능을 보인다. 그 이유는 과부하된 서버의 주변 서버들 역시 작업부하가 많을 경우에는 과잉 작업부하를 주변 서버에게 나누어 줄 수 없기 때문이다. 일반적으로 로컬 기법은 글로벌 기법보다 적은 오버헤드를 갖지만 낮은 성능을 보인다.

글로벌 기법은 시스템 전체 서버들의 정보를 이용해 각 서버들의 작업부하가 균일하도록 전체 작업부하를 나눈다. J. Lui 와 M. Chen [8]은 이를 위해 [9]에서 제안한 그래프 분할 기법을 이용하였다. 먼저, 모든 서버들과 사용자들의 정보를 이용해 그래프를 구성하고 이를 그래프 분할 알고리즘을 이용해 분할한다. 분할된 결과에 따라 각 사용자들을 과부하된 서버에서 작업부하가 적은 서버로 이동시킨다. 글로벌 기법은 서버간 작업부하의 불균형을 많이 줄일 수 있으므로 로컬 기법보다 좋은 성능을 갖는다. 하지만 그래프 분할 알고리즘의 수행 비용이 서버와 사용자의 수에 비례하여 급격히 증가하고, 작업부하의 이동이 많다. DVE와 같은 실시간 상호 작용 애플리케이션에서 이는 상호 작용 성능을 크게 저하시킨다.

3. 제안하는 기법

3.1 시스템 모델

제안하는 기법은 다음과 같은 DVE 시스템을 가정한다

① N개의 서버 $S_1, S_2, \ldots, S_N$ 으로 구성되어 있으며, M개의 직사각형 영역 $R_1, R_2, \ldots, R_M$으로 분할된 가상 세계를 지원한다. 사용자들은 가상 세계를 자유롭게 이동하며 다른 사용자들과 상호 작용한다. 각 서버는 하나 이상의 영역과 그 영역 안의 사용자들을 담당한다.

① 클라이언트/서버 통신 구조[1]를 갖는다. 각 사용자는 자신의 상태 정보를 자신을 담당하는 서버로 보내고 서버는 그 정보를 다른 사용자들에게 전달한다. 정보 교환의 범위는 인접 영역(neighbor regions)으로 제한된다.

② 두 개의 영역 R_i와 R_j가 적어도 하나의 영역 경계(boundary)를 공유하고 있으면, R_i와 R_j를 인접 영역이라 정의한다. 영역간의 정보 교환을 영역간 상호 작용(inter-region interaction) 혹은 영역간 통신(inter-region communication)이라 한다.

③ 두 개의 서버 S_i와 S_j가 적어도 하나의 인접 영역을 가지고 있으면 서버 S_i와 S_j를 인접 서버(neighbor server)라 정의한다. 각 인접 서버들은 서로 초고속 네트웍으로 연결되어 있다. 영역간 통신으로 인해 서버간에도 통신이 필요하며 이를 서버간 통신(inter-server communication)이라 한다.

④ 영역 R의 작업부하를 $w(R)$로 표시한다. 모든 사용자가 단위 시간당 동일한 수의 메시지를 발생시킨다고 가정하면, 영역의 작업 부하 – 즉, 사용자들의 정보를 전달하는 처리 비용(processing cost) – 는 그 영역의 사용자 수에 비례한다. 따라서, 영역 R의 작업부하 $w(R)$은 영역 R의 사용자 수로 정의한다.

⑤ 서버 S의 작업부하를 $w(S)$로 표시한다. 서버 S의 작업부하 $w(S)$는 서버 S가 관리하는 영역들의 작업부하의 합으로 정의한다.

⑥ 두 영역의 상호 작용은 두 영역의 사용자 수의 합에 비례한다. 따라서, 두 영역의 상호 작용량 (영역간 통신)은 두 영역의 사용자 수의 합으로 정의한다.

⑦ 두 서버의 서버간 통신은 두 서버에 각각 속한 인접 영역들의 영역간 통신의 합으로 정의한다.

⑧ 각 서버의 작업 부하 상태를 정량화 하기 위해 혼잡도 (degree of congestion, DOC)를 다음과 같이 정의한다.

$$DOC(S) = w(S) / w^* \qquad (1)$$

⑨ 서버의 혼잡도가 미리 정의한 한계치 $THRESHOLD$보다 크면 그 서버는 과부하된 것으로 간주하고 로드 밸런싱을 시작한다. $THRESHOLD$ 값은 시스템 관리자가 시스템의 성능을 고려하여 결정한다.

⑩ 각 서버는 주기적으로 자신의 DOC 값을 계산하여 시스템의 다른 서버들에게 멀티캐스트한다.

그림 1은 4개의 서버로 구성된 간단한 DVE 시스템과 그에 해당하는 그래프 모델을 보여준다.

3.2 제안하는 로드 밸런싱 알고리즘

각 서버는 자신이 과부하가 되었는지 주기적으로 체크하여 과부하가 되었을 경우 로드 밸런싱을 시작한다. 이때 과부하된 서버는 주변 서버들의 부하 상태에 따라 서로 다른 로드 밸런싱 알고리즘을 수행한다. 만약 과부하된 서버와 그 주변 서버들의 평균 DOC값이 $THRESHOLD$보다 작으면 LD 로드 밸런싱 알고리즘을 수행하고, 그렇지 않으면 LSG 로드 밸런싱 알고리즘을 수행한다.

3.3 로컬 확산 로드 밸런싱

로컬 확산 (LD) 로드 밸런싱은 과부하된 서버가 자신의 과잉 작업부하를 주변 서버 중 그 작업부하를 받은 후에도 과부하가 되지 않을 서버들에게 넘긴다. 과부하된 서버 S가 수행하는 LD 로드 밸런싱 과정은 다음과 같다.

① 서버 S의 과잉 작업부하량($w(S) - w^*$)을 계산한다.

② 서버 S의 경계 영역 (border region) 리스트 R_LIST 를 만든다.

③ R_LIST로부터 임의로 영역을 선택하여, 선택한 영역의 작업부하가 과잉 작업부하량 보다 적으면 주변 서버 중 그 영역을 받은 후에도 과부하가 되지 않을 서버로 그 영역을 이동시킨다. 만약, 그러한 주변 서버가 여러 개일 경우 그 중 영역 이동 후 서버간 통신이 가장 적어지는 서버로 영역을 이동시킨다.

④ 과잉 작업부하를 모두 이동시키거나 미리 정한 횟수만큼 수행할 때까지, ②와 ③과정을 반복한다.

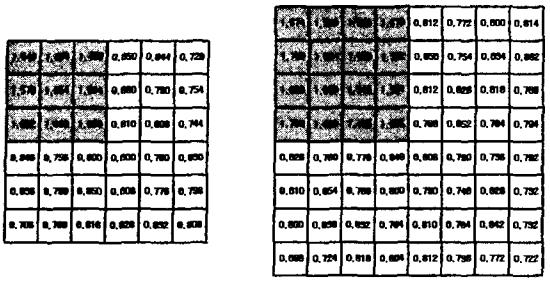
3.4 제한적 글로벌 로드 밸런싱

제한적 글로벌 (LSG) 로드 밸런싱은 로드 밸런싱에 참여하는 서버의 범위가 제한된다는 점에서 모든 서버가 로드 밸런싱에 참여하는 기존의 글로벌 기법과 다르다. 먼저, 과부하된 서버는 로드 밸런싱에 참여할 서버의 범위를 결정한다. 이를 통해 참여가 결정된 서버들과 그 서버들이 담당하는 영역에 대한 그래프를 구성한 후 그래프 분할 알고리즘 [11]을 이용해 분할한다. 과부하된 서버 S가 로드 밸런싱에 참여할 서버들의 범위를 결정하는 과정은 다음과 같다.

① 서버 S를 참여 서버 리스트 S_LIST에 포함시킨다.

② 서버 S의 모든 주변 서버들을 S_LIST에 포함시키면서 $QUEUE$에 기록한다. S_LIST에 서버를 포함될 때마다 포함된 서버들의 평균 DOC값을 갱신한다.

③ 이후, $QUEUE$에 기록된 서버의 주변 서버들을 하나씩 S_LIST에 포함시킨다. 서버가 S_LIST에 추가될 때마다 새롭게 갱신되는 평균 DOC값이 $THRESHOLD$보다 작아



(a) (b)

그림 2. 시뮬레이션의 작업부하 분포 예: 서버수 = (a) 36 (b) 64

지면 본 과정을 중단한다. 그렇지 않을 경우에는 계속해서 모든 서버가 포함될 때까지 본 과정을 반복한다.

4. 성능 분석

4.1 성능 척도

다중 서버 DVE 시스템을 위한 로드 밸런싱 기법의 성능은 다음 척도들을 이용해 평가한다.

① **작업 부하 분산 감소율**(Workload Variance Reduction Ratio): 시스템의 작업 부하 분산, 즉 서버간 작업 부하 분산이 로드 밸런싱을 통해 몇 % 감소되었는가를 나타낸다.

② **작업 부하 이동 비율**(Workload Migration Ratio): 로드 밸런싱의 결과로 전체 사용자 중 몇 %의 사용자들이 서버를 옮겨야 하는지를 나타낸다.

③ **효율**(Efficiency): 작업 부하 분산 감소율에서 작업 부하 이동 비율을 뺀 값으로 로드 밸런싱 알고리즘이 얼마나 효율적인가를 나타낸다.

4.1 시뮬레이션 모델

시뮬레이션은 다음과 같은 조건에서 실행되었다.

① 초기에 모든 서버는 10×10 크기의 영역 100개를 관리하고 있으며, 한 영역당 평균 사용자 수는 5명이다. 즉, 한 서버당 평균 사용자 수는 500명이다.

② *THRESHOLD*는 1.05로 정한다.

③ 전체 서버의 25%가 과부하가 되도록 사용자들의 초기 위치를 설정한다. 즉, 전체 사용자의 80%를 전체 가상 세계에 임의로 위치시키고 나머지 20%를 위 25%의 서버가 관리하는 영역에 위치시킨다.

④ 최악의 상황(worst case)을 테스트하기 위해 그림 2와 같이 모든 과부하 서버들을 한쪽에 집중시킨다.

4.3 시뮬레이션 결과

그림 3과 4는 시뮬레이션 결과를 보여준다. 서버의 수가 많아질수록 제안하는 기법이 로컬 기법과 글로벌 기법보다 효율적임을 알 수 있다. 또한, 제안하는 기법은 서버의 수가 증가하면서 효율이 감소하는 추세가 기존 기법들보다 완만함을 알 수 있다. 이러한 결과를 통해 제안하는 기법이 기존 기법들보다 대규모 다중 서버 DVE 시스템을 위한 보다 확장성 있는 로드 밸런싱 기법임을 알 수 있다.

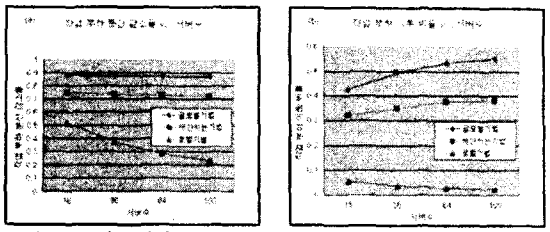


그림 3. (a) 작업 부하 분산 감소율 (b) 작업 부하 이동 비율 대 서버수

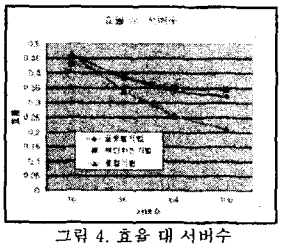


그림 4. 효율 대 서버수

5. 결론

본 논문은 대규모 다중 서버 DVE 시스템을 위한 확장성 있는 로드 밸런싱 기법을 제안하였다. 제안하는 기법은 과부하된 서버의 주변 서버들의 작업 부하가 적을 경우에는 기존의 로컬 기법과 비슷한 로컬 확산 (*LD*) 로드 밸런싱을 수행하고, 그렇지 않을 경우에는 제한적 글로벌 (*LSG*) 로드 밸런싱을 수행한다. 제안하는 기법은 서버들의 작업 부하 분포에 동적으로 적응(adaptation)함으로써, 로컬 기법보다 좋은 성능을 보여주며 글로벌 기법보다 적은 비용을 갖는다. 시뮬레이션 결과를 통해 서버의 수가 많을 경우에 제안하는 기법은 기존 로컬 기법과 글로벌 기법보다 효율의 측면에서 각각 14%, 3%가 우수함을 알 수 있다. 또한, 제안하는 기법은 서버의 수가 증가함에 따라 효율이 감소하는 추세가 기존 기법들보다 완만함을 알 수 있다. 결론적으로, 제안하는 기법은 대규모 다중 서버 DVE 시스템을 위한 확장성 있는 로드 밸런싱 기법임을 알 수 있다.

향후 과제로서 다중 서버 DVE 시스템을 위한 효율적인 그래프 분할 기법에 대한 연구와 서버간 작업부하의 이동에 대한 연구를 진행할 것이다. 제안하는 기법은 현재 대규모 DVE 시스템을 위한 네트웍 프레임웍인 ATLAS [12]에 구현 중이다.

Acknowledgement

본 논문은 한국과학재단 가상현실연구센터의 지원으로 수행하였습니다.

참고문헌

[1] M. Macedonia and M. Zyda, *Networked Virtual Environments Design and Implementation*, Addison Wesley, 1999.

[2] E. Frécon, C. Greenhalgh, and M. Stenius, "The DIVEBONE — an Application-Level Network Architecture for Internet-Based CVEs," *ACM VRST'99*, pp.58–65, 1999.

[3] T. Funkhouser, "RING: A Client-Server System for Multi-User Virtual Environments," *ACM SIGGRAPH'95*, pp.85-92, 1995.

[4] G. Singh, L. Serra, W. Prg, et al. "BrickNet: A Software Toolkit for Network-Based Virtual Environments," *PRESENCE, 3(1)*, pp.19-34, 1994.

[5] C. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics, 21(4)*, pp.25-34, 1987.

[6] T. de Senna Carneiro and J. Cotrim Arabe, "Load Balancing for Distributed Virtual Reality Systems," *IEEE SIBGRAPI'98*, pp.158-165, 1998.

[7] S. Pekkola, M. Robinson, M. Saarinen, J. Korhonen, S. Hujala, and T. Toivonen, "Collaborative Virtual Environments in the Year of the Dragon," *ACM CVE'00*, pp.11-18, 2000.

[8] J. Lui and M. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," *IEEE Transactions on Parallel and Distributed Systems, 13(3)*, pp.193-221, Mar. 2002.

[9] Chao-Wei and R. Sanjay, "Parallel Incremental Graph Partitioning," *IEEE Transactions on Parallel and Distributed Systems, 8(8)*, pp.884-896, Aug. 1997.

[10] K. Schloegel, G. Karypis, and V. Kumar, "Wavefront Diffusion and LMSR: Algorithms for Dynamic Repartitioning of Adaptive Meshes," *IEEE Transactions on Parallel and Distributed Systems, 12(5)*, pp.451-466, May 2001.

[11] "METIS", *http://www-users.cs.umn.edu/~karypis/metis/*.

[12] D. Lee, M. Lim, and S. Han, "ATLAS – A Scalable Network Framework for Distributed Virtual Environments," *ACM CVE'02*, 2002