http://dx.doi.org/10.7583/JKGS.2012.12.3.49

게임 서버 클러스터에서 서버들간의 부하 균형을 고려한 게임 배치 방법※

차충용, 김상철
한국외국어대학교 컴퓨터공학과
chezhongyong@hotmail.com, kimsa@hufs.ac.kr

A Game Placement Method Considering Load Balancing between Game Servers in a Game Server Cluster

Zhong Yong Che, Sangchul Kim
Dept. of Computer Science and Engineering, Hankuk University of Foreign Studies

요 약

한 업체가 신규 게임 또는 사용자가 많지 않은 게임들을 서비스 하는 경우, 게임별 별도의 서버를 두는 것 보다 서버 여러 대에서 함께 서비스하는 경우가 편리할 것이다. 본 논문에서는 동일 성능의 서버들로 구성된 클러스터 상에서 게임 여러 개를 효과적으로 서버에 배치하는 방법을 제안한다. 우리의 방법에 의하면, 서버들간 부하 불균등을 줄여 사용자 기각률을 최소화 할 수 있는 게임 배치를 구할 수 있다. 본 논문에서 제안한 방법은 게임 서비스 시 서버 수, 서버당 게임 수 등과 같은 서버 운영 사항들을 결정하는데 도움을 줄 것으로 예상한다. 우리의 조사에 따르면, 서버 클러스터에서의 게임 배치에 관한 기존 연구는 거의 발표되지 않았다.

ABSTRACT

When a company provides new games or games with a small number of users, it is convenient to provide the game service in which those games are placed in a server cluster instead of a separate server for an individual game. This paper proposes a method for placing games in a cluster of servers with the same capacities. The method reduces the load imbalance among servers and minimizes the rejection ratio of user requests. We expect that the proposed method is useful for determining the server operation factors such as the number of servers, the number of games for a server, etc. To our survey, little research has been published on game placement on a cluster of game servers.

Keywords : Game Server(게임 서버), Game Placement(게임 배치)

접수일자 : 2012년 04월 17일 심사완료 : 2012년 05월 14일
교신저자(Corresponding Author) : 김상철
※ 본 논문은 2011년 한국외국어대학교 교내연구비 지원에 의한 것임.

1. 서 론

대부분 온라인 컴퓨터 게임의 경우, 많은 사용자들이 동시에 서버에 접속하는 방식으로 동작한다. 상업화에 성공한 MMORPG 게임의 경우에는 그 수가 수천내지 수만에 이르지만, 출시된 지 얼마지 않은 게임, 특정 사용자 집단을 위한 게임, 또는 특정 목적 하에 만들어진 기능성 게임은 잠재 사용자수가 많지 않아 동시 접속자 수백정도에 불과한 경우가 있다. 동시 접속자 수가 게임 서버 한대의 처리 능력을 초과하는 게임의 경우에는 여러 대의 서버를 사용하고 그 부하를 동적으로 균등하게 분배하게 된다. 서버 부하란 해당 서버에 접속된 동시 접속자 수로 측정할 수 있을 것이다. 따라서 사용자의 서버 접속 요구를 여러 서버들에게 최대한 균등하게 분배하는 소위 로드 발란싱[1,2,3,4,5]에 대한 연구가 국내외적으로 많이 진행되고 있다.

한 업체가 동시 접속자 수가 적은 게임 여러 개를 서비스하는 경우, 게임마다 별도의 서버를 운영하지 않고, 서버 한대에 게임 여러 개를 설치하고 필요시 이와 같은 서버를 여러 대 운영하게 된다. 예를 들면, 플래시로 제작한 미니 게임 수백 개를 운영하는 사이트는 서버 한 대에 여러 게임들을 설치하는 방식으로 서비스하고 있다. 게임의 인기도에 따라서 한개 게임을 서버 클러스터(즉, 서버 여러 대)에 배치하거나 또는 여러 게임을 서버 한 대에 동시에 배치하게 될 것이다. 본 논문에서는 게임 인기도는 평균 동시 접속자 수로 측정한다고 본다.

서버에 배치된 게임들의 인기도 합이 해당 서버의 정상 처리 능력보다 크면, 해당 서버의 처리 능력을 초과해서 발생하는 신규 접속 요구는 기각될 수밖에 없다. 서버가 정상 처리 능력을 초과하는 큰 수의 동시 접속자를 처리하게 할 수도 있겠지만, 이 경우에는 사용자 반응시간이 늘어져 정상적인 게임 플레이가 불가능할 것이다. 따라서 한 업체가 동시 접속자 수가 적은 게임 여러 개를 운영하고자 하는 경우, 서버 성능과 서버 수뿐만이 아니라 게임 배치의 결정도 고려하여야 한다. 여기서, 게임 배치란 각 게임을 어떤 서버에 배치하느냐는 것을 나타낸다. 게임 배치가 중요한 이유는 서버 배치 결과가 사용자 기각률에 영향을 미치기 때문이다. 인기도가 서로 다른 여러 게임들을 여러 서버에 배치하게 되면, 각 서버에 설치된 게임들의 인기도 합계는 서버별로 서로 다를 수 있다. 배치된 게임들의 총 인기도가 자신의 처리능력을 넘는 서버의 경우, 운영 중에 과부하가 걸려 신규 접속 요구를 거절해야 하는 경우가 발생하게 될 가능성이 높다. 따라서 특정 서버에 과부하가 걸리는 현상의 발생 가능성을 낮추기 위해 서버에 배치된 게임들의 인기도 합이 서버간에 균등하게 유지하는 것이 중요하다. 게임들의 게임 배치의 결과에 따라 사용자 기각률이 달라지는 것은 나중에서 기술될 우리의 실험에서 확인할 수 있다.

본 논문에서는 동일한 성능의 게임 서버를 클러스터 형태로 운영하는 환경에서 서버 부하가 서버들간에 최대한 균등해 사용자 기각률을 최소화 할 수 있는 게임 배치 방법을 제안한다.

게임 서버에 관한 기존 연구는 크게 네트워크상의 서버 배치와 사용자 접속 요구의 균등 배분이라는 주제로 진행되어 왔다. 전자의 경우, 사용자와 서버간의 지연시간을 최소화하도록 사용자에게 가장 가까운 서버 배정하는 방법[6,7], 사용자와 서버간의 최대 지연 시간이라는 조건하에서 서버의 수를 최소화하는 방법[8] 등이 제시되었다. 후자의 경우, 전체 게임 맵을 여러 지역으로 나눈 후 지역별로 서버를 두어 서버 부하를 분산하는 방법[3,4,9], 신규 사용자 접속 요구에 대해 최소 부하를 갖는 서버를 동적으로 선택하는 방법[2,5] 등이 제안되었다.

우리가 조사해 본 바, 본 논문에서 다루고자 하는 다수 게임을 부하 균등을 고려하여 서버 클러스터에 배치하는 방법에 대한 기존 연구는 거의 발표되지 않았다. 대신에 VOD(Video-On Demand)와 웹 콘텐츠 분야에서는 관련 연구가 존

재한다. 사용자의 접속 요구면에서 서버들간의 차이를 최소화하도록, 비디오 데이터를 다수의 비디오 서버에 배치[10]하거나 웹 콘텐츠를 다수의 프락시 서버들에 배치[11]하는 연구가 다수 발표되어 왔다. 이들 연구에서는 휴리스틱 방법을 이용하여 많은 수의 비디오나 웹 콘텐츠를 작은 수의 서버에 분산 배치하는 방법을 제안하고 있다. 게임은 비디오나 웹페이지와 같이 단순히 저장한 후 클라이언트들에게 전송 서비스만 해주는 것이 아니라 복잡한 코드 실행 및 사용자와의 쌍방향 데이터 전송이 필요한 서비스이다. 따라서 서버 성능만을 주로 고려하는 비디오 및 웹 콘텐츠의 데이터 배치에 관한 연구 결과를 게임 배치에 그대로 적용할 수는 없다. 본 연구는 이들 기존 연구를 게임 배치에 확장한 것으로 볼 수 있다.

실험 결과, 서버들간 부하 불균형이 작은 게임 배치일수록 사용자 기각률이 낮아짐을 알 수 있었다. 제안된 게임 배치 방법은 서버들간 부하 불균형을 최대한 줄여 줌을 알 수 있었다. 또한, 서버의 수, 게임의 최대 복사 수, 각 서버당 배치하는 게임의 최대 수 등이 서버들간 게임 배치의 결과에 영향을 미침을 알 수 있었다. 본 연구 결과는 여러 게임들을 동시에 운영하는 환경에서, 사용자 기각률을 최소화시키기 위한 서버의 수나 게임 배치를 결정하고자 할 때, 유용하게 이용될 수 있을 것이다.

2. 문제 정의

2.1 용어 정의

$S=\{s_1, s_2, ..., s_{MAX_S}\}$는 게임 서버 MAX_S개의 집합이고, $G = \{g_1, g_2, ..., g_{MAX_G}\}$는 게임 MAX_G개의 집합이다. 각 게임 서버 s_i에 대해서, $G_s(s_i) = \{g_j | \ g_j \in G, \ \ g_j$는 s_i에 배치됨$\}$는 s_i에 배치된 게임들의 모임이다. 각 게임 g_j에 대해서, $S_g(g_j) = \{s_i | s_i \in S, \ g_j$는 s_i에 배치됨$\}$는 g_j가 배치된 서버들의 모임이다. 이 경우 게임 g_j는 $|S_g(g_j)|$개의 복사를 갖는다고 한다. $Noc(g_j)$는 게임 g_j의 복사 수를 나타낸다.

게임들 간의 인기도가 다를 수 있는데, 게임의 인기도는 앞에서 언급한대로 사용자의 접속 요구를 바탕으로 측정할 수 있을 것이다. 각 게임 g_j에 대해서, $P(g_j)$는 g_j의 상대적 인기도를 나타낸다:

$$0 \ \leq \ P(g_j) \ \leq \ 1, \ \sum_{g_j \in G} P(g_j) = 1$$

서버 한 개에 배치되어 서비스되던 어떤 게임이 서버 α개에 배치된다면, 각 서버에 대한 해당 게임을 위한 사용자의 접속 요구 수는 평균적으로 한 개 서버일 때의 접속 요구 수의 $1/\alpha$으로 줄어들 것이다[10]. 예를 들어, 게임 g_k의 인기도가 0.4이고, g_k를 서버 두 개에 배치한다면, 게임 g_k의 각 복사의 인기도는 0.2=0.4/2가 될 것이다. 각 게임 g_j에 대해서, $P_{AVG}(g_j)$는 게임 g_j의 평균 인기도를 나타내는 것으로서,

$$P_{AVG}(g_j) \ = \ P(g_j) \ / \ \ Noc(g_j) \qquad \text{(식 1)}$$

로 정의된다. $P_{AVG}(g_j)$는 게임 g_j의 각 복사의 인기도라고 할 수도 있다.

각 서버 s_i에 대해서, $P_{TOT}(s_i)$는 s_i에 배치된 게임 복사들의 인기도의 합이다. 즉,

$$P_{TOT}(s_i) \ = \ \sum_{g_j \in G_s(s_i)} P_{AVG}(g_j) \qquad \text{(식 2)}$$

본 논문에서 서버 s_i들의 성능은 동일하다고 가정한다. 본 논문의 내용은 각 서버의 성능이 다른 경우에도 쉽게 확장할 수 있을 것이다.

2.2 문제 정의

여러 게임들을 여러 대의 서버를 이용하여 동시

에 서비스하는 업체가 각 게임을 한 대 또는 여러 대의 서버에 별도로 설치하여 운영하는 경우를 고려하자. 이 경우, 만약 이들 게임들 간의 인기도의 차이가 크다면 서버들 간 부하의 균형을 유지하기 어려울 것이다. 즉, 어떤 서버에 있어서는 자신의 성능보다 사용자 접속 요구가 적고, 어떤 서버에 있어서는 자신의 성능보다 많은 사용자 접속 요구를 처리해야 할 수도 있을 것이다. 따라서 여러 대의 서버를 운영하면서 한 서버가 여러 게임을 호스팅할 수 있도록 하는 것이, 서버들 간 부하의 균형을 유지하기 쉬울 것이다.

서비스하고자 하는 게임들의 평균 동시 사용자 접속 수의 총계와 서버 한 대가 처리할 수 있는 동시 사용자 접속 수를 알면, 필요한 서버의 개수는 계산할 수 있을 것이다. 즉, 전체 게임의 동시 사용자 수가 M이고, 서버 한 대가 동시에 처리할 수 있는 동시 사용자 수가 m이라면, 필요한 서버의 개수는 $\frac{M}{m}$가 될 것이다. 물론, 게임별로 한 개의 사용자 접속이 서버에 미치는 부담은 차이가 날 수 있을 것이다. 이런 경우, 게임 별로 가중치를 주어 M이나 m을 계산하면 될 것이다. 예를 들면, 어떤 게임을 기준으로 정해서, 다른 게임들의 동시 사용자 수는 같은 부담을 나타내는 기준 게임의 사용자 수로 표현하면 될 것이다. 본 논문에서는 편의상 게임별로 사용자 한 명이 서버에 미치는 부담은 동일하다고 본다.

본 과제에서 해결하고자하는 문제는 서버들간 부하 불균형을 최소화하는 게임들의 게임 배치를 구하는 것으로서, 다음과 같이 정의한다:

게임 배치: 서버 집합 S={s_1, s_2, ..., s_{MAX_S}}와 게임 집합 G = {g_1, g_2, ..., g_{MAX_G}}에 대해서, 아래 제약 1~2를 만족하면서, 부하 불균형 지표인 $\mathcal{L}$을 최소화시키는 각 서버 s_i에 대한 $G_s(s_i)$를 구하는 것이다.

$$\mathcal{L} = MAX_{s_i \in S}\{P_{TOT}(s_i)\} - MIN_{s_i \in S}\{P_{TOT}(s_i)\} \quad \text{(식 3)}$$

제약 1: $\forall g_j \in G,\ 1 \le Noc(g_j) \le COG$

제약 2: $\forall s_i \in S,\ |G_s(s_i)| \le SC$

제약 1에서 $Noc(g_j)$가 1보다 크다는 것은 각 게임은 적어도 한 개 서버에 배치되어야함을 나타낸다. 또한, $Noc(g_j)$가 파라미터 COG (게임별 복사 수의 최대치)보다 작다는 것은 한 게임의 배치 수에 제약을 두는 것이다. 당연히 COG는 서버 개수인 MAX_S보다 같거나 작은 값이다. 게임의 배치 수에 제약을 두는 이유는, 이 수치가 크면 그 만큼 게임 운영상 유지 관리에 어려움이 따를 것이기 때문이다.

제약 2는 서버 한 개에 배치할 게임의 개수는 파라미터 SC (서버별 게임 수의 최대치)를 넘지 않음을 나타낸다. 한 개 서버에서 여러 종류의 게임 프로세스가 동시에 동작하면, 한 종류의 게임 프로세스인 경우 보다 더 많은 하드웨어 및 소프트웨어 리소스를 많이 사용하게 되어, 게임 서버의 운영 속도가 느려질 가능성이 높을 것이다. 서버 한 개가 수용(hosting)할 수 있는 게임의 개수는 서버의 메모리 용량, 프로세스 속도등과 같은 서버 성능에 따라 결정될 것이다.

3. 게임 배치 알고리즘

게임 배치 알고리즘은 크게 두 작업으로 구성된다. 첫 번째 작업은 각 게임별로 배치 수를 구하는 것이고, 두 번째 작업은 게임들을 배치한 서버들을

결정하는 것이다. 2.2절의 게임 배치 문제는 빈 패킹(bin packing) 문제의 일종으로서 NP-complete 복잡도를 갖기 때문에, 우리는 휴리스틱 알고리즘을 제안한다.

[그림 1]의 GET_NO_OF_COPIES 프로시저는 첫 번째 작업을 위한 코드로서, 출력은 각 게임 g_j에 대하여 복사 수이다. 이 프로시저의 기본 동작 원리는 2.2절에서 기술한 제약 1과 제약 2를 만족하는 범위에서, 각 게임의 복사수를 늘이는 것이다. 이 프로시저와 아담스 단순 제수 기반의 복제 방법[11,13]과 유사하지만, 여러 차이점중 하나는 전자에서는 게임당 최대 복사 수를 고려하지 않는 것이다.

Input: S, G, SC, COG,
$\forall g_j \in G$에 대한 $P(g_j)$
Output: $\forall g_j \in G$에 대한 $Noc(g_j)$

Step 1: $\forall g_j \in G$ 에 대해서, $Noc(g_j) \leftarrow 1$

while(true) {
  Step 2: $P_{AVG}(g_j)$을 (식 1)에 따라 계산한다.
  Step 3: $G_{temp} \leftarrow \{g_j \mid g_j \in G, Noc(g_j) < COG\}$
    if $G_{temp} \neq \emptyset$) goto Step 4;
    else **break;**

  Step 4: 각 $g_k \in G_{temp}$에 대해서,
   Step 4.1: $Noc(g_k)$를 1만큼 증가시킨다는
    가정 상태에서 $P_{AVG}(g_k)$을 새로 계산한다.
   Step 4.2: $D_k \leftarrow MAX_{g_l \in G_{temp}}\{P_{AVG}(g_k)$ -
    $MIN_{g_l \in G_{temp}}\{P_{AVG}(g_k)\}$

  Step 5: $g_{\min} \leftarrow MIN\_ARG_{g_k \in G_{temp}}\{D_k\}$
    $Noc(g_{\min}) \leftarrow Noc(g_{\min}) + 1$

  Step 6: $T \leftarrow \sum_{i=1}^{MAX\_G} Noc(g_j)$
    if($T >= SC * MAX\_S$) **break**;
    else **continue**;
}

[그림 1] GET_NO_OF_COPIES 프로시저

[그림 2]의 GET_GAME_PLACEMENT 프로시저는 두 번째 작업을 위한 코드로서, 출력은 각 서버 s_i별로 게임 배치 상태 (즉, $G_s(s_i)$)이다.

Step 3은 g_j가 아직 배치되지 않는 서버들 중에서 배치된 게임의 수가 서버 용량에 도달하지 않은 서버들 구하는 단계로서, 그런 서버들의 집합을 T라고 한다.

Input: S, G, SC,
$\forall g_j \in G$에 대한 $Noc(g_j)$
Output: $\forall s_i \in S$에 대한 $G_s(s_i)$

Step 1: $\forall s_i \in S$, $G_s(s_i) \leftarrow \emptyset$
$\forall s_i \in S$, $P_{TOT}(s_i) \leftarrow 0$
$\forall g_j \in G$, $TMP_j \leftarrow Noc(g_j)$

Step 2: 모든 게임들을 $P_{AVG}(g_j)$의 내림차순으로 정렬하고, 그 순서를 $g_{p(0)}$, $g_{p(1)}$, ..., $g_{p(MAX_G-1)}$라고 하자.

for(j = 0; j < MAX_G; j++)
  for(k = 0; k < $TMP_{p(j)}$; k++) {
    Step 3: $T \leftarrow \{s_i \mid$
      $s_i \in S, g_{p(j)} \notin G_s(s_i), |G_s(s_i)| < SC\}$
    Step 4: $s_{min} \leftarrow MIN\_ARG_{s_i \in T}\{P_{TOT}(s_i)\}$
    Step 5: $G_s(s_{min}) \leftarrow G_s(s_{min}) \cup \{g^k_{p(j)}\}$
      $P_{TOT}(s_{min}) \leftarrow P_{TOT}(s_{min}) + P_{AVG}(g_{p(j)})$
  }
}

Step 6: 게임 재배치를 수행하여 $\mathcal{L}$을 개선한다.

[그림 2] GET_GAME_PLACEMENT 프로시저

Step 4에서 s_{min}은 T에 있는 서버들 중에서 지금까지 배치된 게임들의 인기도 합이 최소인 서버이다. $g^k_{p(j)}$는 게임 $g_{p(j)}$의 k 번째 복사를 말하고, Step 5 에서는 $g^k_{p(j)}$를 Step 4에서 구한 s_{min}에 배치하는 것이다. 만약 Step 3에서 T가 공집합

이면, 첫 번째 제약을 무시하고 다시 T를 구한다. Step 6에서는 게임 재배치를 수행하는 단계로서, 일정한 회수만큼 다음 작업을 반복한다: 배치된 게임들의 인기도 합이 최대인 서버 s_x와 최소인 서버 s_y를 찾는다. 이들 서버에 배치된 게임 복사들 중에서 서로 교환하면 $\mathcal{L}$을 감소시킬 수 있는 두 개 $g_x \in G_s(s_x)$ 와 $g_y \in G_s(s_y)$를 발견한다. 게임 g_x와 g_y를 서버 s_x와 s_y간에 서로 교환한다. 만약 그런 게임들이 없다면 반복을 중단한다.

4. 제안된 방법의 분석

GET_NO_OF_COPIES 프로시저의 Step 2 ~ Step 6을 반복되는 회수를 N라고 하면, $N=min(SC * MAX\_S, COG * MAX\_G)$ 이다. N은 서버 클러스터에 설치되는 게임 복사의 총 개수가 된다. N을 증가시키면, 게임 평균 인기도 (즉, $P_{AVG}(g_j)$)들의 평균값과 최대값은 줄어드는 경향이 있다. 두 번째 작업인 게임 배치는 일종의 빈 패킹 문제로 볼 수 있는데, 빈은 서버에 해당되며 일정한 용량을 가지고, 게임은 빈에 배치하는 물체에 해당되며, 게임의 평균 인기도는 물체의 크기로서, 빈 패킹의 목적은 빈들간의 부하 불균형을 최소화시키는 것이라 할 수 있다. 빈 패킹 문제와 유사하게, 게임들의 평균 인기도가 작고 서로간의 차이가 적어지면, 그 만큼 서버들간의 부하 균형을 맞추기 쉽게 된다.

GET_NO_OF_COPIES 프로시저의 시간 복잡도는 $\bigcirc(N * log(MAX_G))$가 된다. 앞에서 언급한 대로 N은 Step 2~Step 6의 반복 회수이고, $log(MAX_G)$는 Step 4나 5에서의 게임 평균인기도들의 최대값 또는 최소값을 구하는데 필요한 작업의 시간 복잡도이다.

GET_GAME_PLACEMENT 프로시저의 시간 복잡도는 Step 3~5의 반복에 좌우되는데, 반복 회수는 N이다. 이들 중 Step 4의 실행시간이 지배적이기 때문에, 본 프로시저의 시간 복잡도는 $\bigcirc(N * MAX_S * log(MAX_S))$이다.

GET_GAME_PLACEMENT 프로시저의 실행로서, 각 서버마다 최대 SC개의 게임이 배치된다. 서버 s_i에 x번째로 배치된 게임을 $g^{x,i}$라고 하는데, 만약 서버 s_i에 x보다 작은 개수의 게임만 배치되었다면, $g^{x,i}$는 인기도가 0인 게임 g_{dummy}로 보자. 모든 서버들에 x번째로 배치된 게임의 집합을 $Layer_x$라고 하자.

$$Layer_x = \{g^{x,i} \mid 1 \le i \le MAX\_S,\ 1 \le x \le SC\}$$

$Layer_x$ 까지만 서버에 배치한 직후 서버들간의 부하 불균형 지표를 $\mathcal{L}_x$라고 하자. 그러면 $\mathcal{L} = \mathcal{L}_{SC}$이다.

예를 들면, $S=\{s_1, s_2\}$, $G=\{g_1, g_2, g_3\}$, $SC=2$, $G_s(s_1) = \{g_1, g_2\}$, $G_s(s_2) = \{g_2, g_3\}$, $P_{AVG}(g_1)=0.4$, $P_{AVG}(g_2)=0.15$, $P_{AVG}(g_3)=0.15$라고 하자. $g^{1,1}=g_1$, $g^{2,1}=g_2$, $g^{1,2}=g_2$, $g^{2,2}=g_3$이다. 그리고 $\mathcal{L}_1 = (0.4 - 0.15) = 0.25$, $\mathcal{L} = \mathcal{L}_2 = (0.4+0.15) - (0.15+0.15) = 0.25$이다.

본 논문에서 제안된 게임 배치 방법에서, 부하 불균형 지표인 $\mathcal{L}$은 보조정리 1과 정리 1과 같은 특성을 갖는다.

정리 1: $Layer_x$에 속한 게임들 중 평균 인기도가 최대인 게임을 g^x_{max}, 최고인 게임을 g^x_{min}라고 할 때,

$$\mathcal{L}_1 = P_{AVG}(g^1_{max}) - P_{AVG}(g^1_{min})$$

$$\mathcal{L}_x \le \max(\mathcal{L}_{x-1},\ P_{AVG}(g^x_{max}) - P_{AVG}(g^x_{min})),\quad 2 \le x \le SC$$

증명: $Layer_1$만 서버에 배치된 경우, 각 서버당 한 개씩의 게임만이 있기에 $\mathcal{L}_1$은 이런 게임들 중 최대 평균 인기도와 최소 평균 인기도간의 차이이다. x가 2 이상인 경우, $Layer_x$까지를 배치한 후,

배치된 게임들의 평균인기도 합이 가장 작은 서버를 s_a, 가장 큰 서버를 s_b라고 하자. 서버 s_a와 서버 s_b에 대해서, $Layer_{x-1}$까지 배치 한 후 이들 서버간의 부하 불균형을 $\mathcal{L}_{a,b}$라고 하자. 그러면, $g^{x,a}$와 $g^{x,b}$의 평균 인기도 간의 대소 관계는 다음 2가지 경우로 나누어진다.

$P_{AVG}(g^{x,a}) \geq P_{AVG}(g^{x,b})$이면,

$\mathcal{L}_x = \mathcal{L}_{a,b} + P_{AVG}(g^{x,b}) - P_{AVG}(g^{x,a})$

$\leq \mathcal{L}_{a,b} \leq \mathcal{L}_{x-1}$ 임

$P_{AVG}(g^{x,a}) < P_{AVG}(g^{x,b})$이면,

$\mathcal{L}_x = P_{AVG}(g^{x,b}) - P_{AVG}(g^{x,a}) - \mathcal{L}_{a,b}$

$\leq P_{AVG}(g^{x,b}) - P_{AVG}(g^{x,a})$ 임.

따라서

$\mathcal{L}_x \leq max(\mathcal{L}_{x-1}, P_{AVG}(g^x_{max}) - P_{AVG}(g^x_{min}))$임.

보조정리 1: 서버들에게 배치되는 게임을 순서대로 나열하였을 때, n번째로 서버에 배치된 게임을 $g(n)$, $1 \leq n \leq N$라고 하자. 그러면,

$N = MAX\_S$ 이면,

$\mathcal{L} = P_{AVG}(g(MAX\_S)) - P_{AVG}(g(1))$

$N > MAX\_S$ 이면,

$\mathcal{L} \leq max(P_{AVG}(g(MAX\_S)) - P_{AVG}(g(1)), P_{AVG}(g(MAX\_S+1)))$

증명: 정리 1에 따라서, $N = MAX_S$의 경우는 $\mathcal{L} = \mathcal{L}_1$이기에 당연하다. $N > MAX_S$인 경우에는

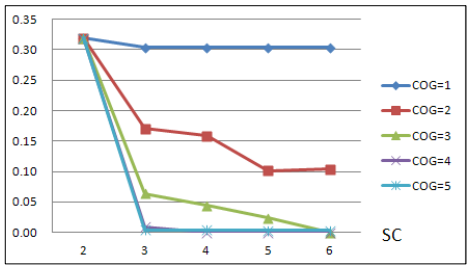
$\mathcal{L} \leq max(\mathcal{L}_1, \max_{2 \leq x \leq SC}(P_{AVG}(g^x_{max}) - P_{AVG}(g^x_{min})))$

이다. $P_{AVG}(g^x_{max})$와 $P_{AVG}(g^x_{min})$간의 차이를 최대로 만들 수 있는 g^x_{max}는 $g(MAX_S+1)$이고, g^x_{min}은 각 서버에 배치된 게임의 수가 서로 같은 지에 따라 달라진다. 만약 같다면 g^x_{min}은 $g(1)$이고 아니면 g_{dummy}이다. $P_{AVG}(g(1)) > P_{AVG}(g_{dummy})$이기에, $P_{AVG}(g^x_{max})$와 $P_{AVG}(g^x_{min})$간의 차이를 최대로 만드는 $P_{AVG}(g^x_{min})$은 0임.

5. 실 험

본 논문에서 제안한 알고리즘을 VC++로 구현하고, 3.2GHz의 Pentium CPU의 컴퓨터에서 성능을 테스트하였다. 다양한 서버 개수 및 성능, 게임별 최대 설치 수에 대해서, 우리의 방법으로 게임 배치를 구한 후 사용자 요구에 대한 기각률을 시뮬레이션하였다. 게임의 인기도는 온라인 콘텐츠 사용 분석에 많이 사용되는 Zipf[12] 분포를 사용하였다. Zipf 분포의 특징에 따라, 게임들을 사용자들이 갖는 선호도의 내림차순으로 정렬 하면 게임별 인기도는 순번의 지수 승에 반비례하게 된다. 게임들의 동시 접속자 수의 합계를 10,000으로 설정하고, 각 게임별 동시 접속자 수는 앞에서 언급한 인기도 비에 따라 결정하였다. 본 실험의 목적은 게임별 복사 수의 최대치, 서버별 설치할 수 있는 게임 수의 최대치, 서버 성능 및 서버 개수 등의 파라미터들이 게임 배치 및 사용자 기각률에 미치는 영향을 분석하는 것이다.

시뮬레이션 시각 t에서, 게임 서버 클러스터에 대한 사용자들의 총 접속 요구 수는 평균 λ을 갖는 포아송 분포를 따르고, 게임별 접속 요구의 비율은 게임의 인기도 비율과 같다고 보았다. 우리의 시뮬레이션에서는 λ = 10,000으로 설정하고, 사용자의 접속 유지 시간은 같다고 가정한다. 어떤 게임에 대한 사용자 접속 요구는 해당 게임을 가지고 있는 서버들 중에 가장 부하 (즉, 동시 접속자 수)가 가장 작은 서버에게 우선 배정하도록 하였다. 게임의 개수인 MAX_G는 10로 설정하였다. SC은 한 개 서버의 성능, 즉 처리할 수 있는 동시 접속자 수를 나타내는데, 우리의 시뮬레이션에서는 기본적으로 SC=2,000으로 설정하였다.

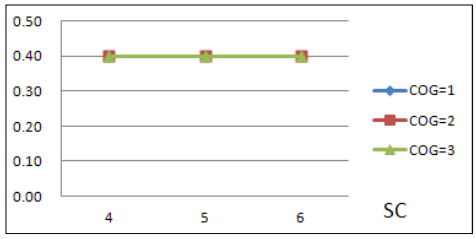


[그림 3] 사용자 기각률 (*MAX_S*=5)

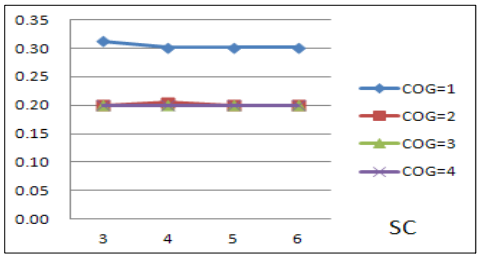
실험 결과, 사용자 기각률은 게임별 복사 수의 최대치인 *COG*, 서버별 설치할 수 있는 게임 수의 최대치인 *SC* 및 서버 개수인 *MAX_S*에 영향을 받았다. [그림 3]은 *MAX_S* = 5인 조건하에서 *SC* 및 *COG*를 다양하게 설정하면서 구한 게임 배치들에 대한, 사용자 기각률 차이를 보여준다. 그림에 나타나듯이 사용자 기각률은 대체적으로 *COG*나 *SC*가 증가하면 낮아지는 경향이 있다. 그 이유는 *COG*나 *SC*가 증가하면 서버에 설치되는 게임의 총 복사 수가 증가하게 되고, GET_NO_OF_COPIES 프로시저 속성상 게임 복사들간 인기도의 차이가 줄게 되는 경향이 있기 때문이다. 또한 게임 복사들간 인기도의 차이가 작은 게임 복사들을 서버에 배치하면, 서버들간 부하 불균형도 작아질 수 있기 때문이다.

또 다른 현상은 *SC*나 *COG*가 증가하게 되면 사용자 기각률의 감소 추세가 약화되는 것이다. 그 이유는 앞에서 언급했듯이 *SC*나 *COG*가 총 복사 수에 큰 영향을 미치는데, *SC*나 *COG*가 커질수록 *SC*나 *COG*의 변화에 따라 발생하는 게임 복사들간 인기도의 차이의 변화가 작아지기 때문이다. *COG*=1은 특별한 경우로서, *SC*의 값에 큰 영향이 없게 된다. *SC*=2 경우, 전체 서버에 설치되는 게임의 총 복사 수가 *MAX_G*이기 때문에 $\pounds$이 *COG*에 무관하게 된다. 상세히 설명하면, 시뮬레이션에서 파라미터 *COG*값을 다르게 설정하더라도, GET_NO_OF_COPIES 프로시저는 각 게임당 한 개의 복사만 구하게 된다.

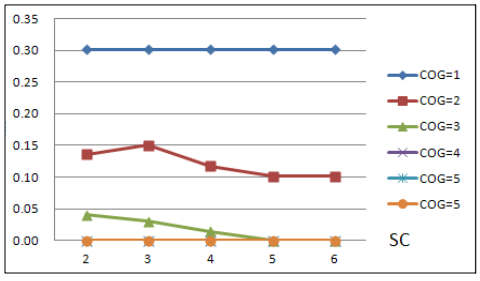
서버 수를 증가시키면 사용자 기각률이 감소하였다. 그 이유는 전체 부하가 더 많은 서버에 분산되므로 각 서버당 사용자 기각률이 낮아지기 때문이다. 또 다른 이유는 *COG*와 *SC*가 같은 조건이라면, *MAX_S*가 커지면 게임의 총 복사 수가 증가하여 $\pounds$이 감소하는 경향을 보이고, 그 결과 사용자 기각률도 낮아지는 것이다. [그림 4,5,6,7]은 *MAX_S*가 각각 3, 4, 6, 7인 경우 사용자 기각률이다. 특히 *MAX_S*가 3 또는 4인 경우, 서버 클러스터가 처리할 수 있는 총 동시 접속자수인 *MAX_S* * *SC*가 동시 접속자 수의 평균인 λ보다 크다. 따라서 사용자 기각률이 0 보다 항상 크다. 서버의 개수를 늘이면 낮은 사용자 기각률을 유지할 수 있겠지만, 그 만큼 서버 구입 및 유지 비용이 늘어 날 것이다.



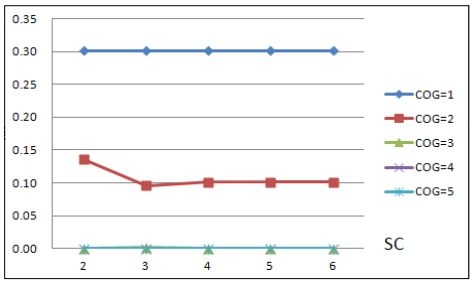
[그림 4] 사용자 기각률 (*MAX_S*=3)



[그림 5] 사용자 기각률 (*MAX_S*=4)

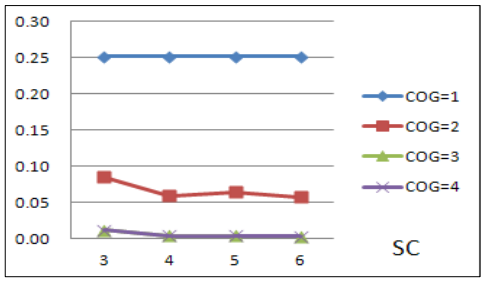


[그림 6] 사용자 기각률 (*MAX_S*=6)

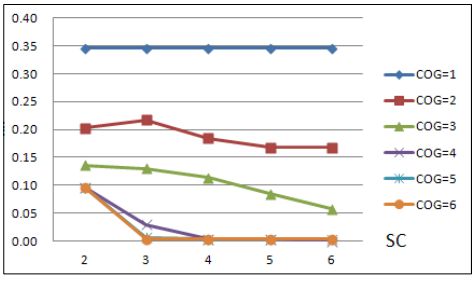


[그림 7] 사용자 기각률 (*MAX_S*=7)

서버 수는 다르지만 서버 성능의 합이 같은 경우, 사용자 기각률에 차이가 남을 알 수 있었다. [그림 8]과 [그림 9]는 서버 성능의 합이 10,000이지만 서버의 수가 다른 경우이다. 이들과 [그림 3]을 함께 살펴보면, 같은 *COG*인 경우에는 서버 수가 적고 서버 성능이 뛰어난 경우가 사용자 기각률면에서 우수함을 알 수 있다. 그 이유는 성능이 줄면서 서버의 개수가 많아지면, 서버들간의 부하 균등을 이루기를 맞추기가 그 만큼 어려워지기 때문이다.



[그림 8] 사용자 기각률
(*MAX_S*=4, *SCAP*=2500)



[그림 9] 사용자 기각률
(*MAX_S*=6, *SCAP*=1666)

성능이 같지만 서버 개수가 다른 경우, 사용자 기각률이 0에 가까워지는 지점이 조금씩 달라지는 알 수 있다. 서버의 수가 늘어나면 그런 지점의 *SC*나 *COG*가 줄어드는 경향이 있다. [그림 3,6,7]을 보면, *MAX_S*=5인 경우 *SC*=3과 *COG*=4면 사용자 기각률이 0이 된다. *MAX_S*=6이면 *SC*=2와 *COG*=4, *MAX_S*=7이면 *SC*=2와 *COG*=3면 사용자 기각률이 0이 된다. *SC*나 *COG*가 늘어나면 그만큼 게임 서비스 비용이 늘어날 것이므로, 사용자 기각률을 낮추기 위해, 서버 개수, *COG* 및 *SC*에 대해 어떤 값을 선택할 것인가는 게임 운영자의 선택에 달려 있다.

6. 결 론

사용자가 많지 않은 게임들인 경우, 각 게임을 별도의 서버에서 운영하기 보다는 서버 여러 대에서 함께 서비스하는 경우가 편리할 것이다. 본 논문에서 우리는 서버 클러스터 상에서 게임 여러 개를 효과적으로 배치하는 방법을 제안하였다. 우리의 배치 방법은 다양한 서버 수, 서버에 배치한 최대 게임 수, 게임의 최대 복사 수 등에 대해, 서버들간 부하 불균등을 줄여 사용자 기각률을 최소화 할 수 있는 게임 배치를 구한다.

실험 결과, 우리의 방법에서 서버의 수, 서버에 배치할 최대 게임 수, 게임의 최대 복사 수는 서버 부하 및 사용자 기각률 면에서 게임 배치에 많은 영향을 미침을 알 수 있었다. 또한, 적절한 서버수나 서버 성능 또는 게임의 최대 복사 수를 선택하면, 사용자 기각률을 현저히 낮게(가능하면 0까지) 하는 게임 배치를 구할 수 있음을 알 수 있었다. 본 논문에서 제안한 방법은 게임 서비스 시 서버 수, 서버당 게임 수 등과 같은 서버 운영 정책을 결정하는데 큰 도움을 줄 것으로 기대한다.

참고문헌

[1] D. Saha, S. Sahu, and A. Shaikh, "A service platform for online games," Proc. of the 2nd Workshop on Network and System Support for Games, pp.180-184, 2003.

[2] Herbert Jordan, et. al, "Dynamic Load Management for MMOGs in Distributed Environments," Proc. of the 7th ACM International Conference on Computing Frontiers, pp.337-346, 2010.

[3] F. Lu, S. Parkin, and G. Morgan. "Load balancing for massively multiplayer online games," Proc. of 5th ACM SIGCOM Workshop on Network and System Support for Games, pp.1-5, 2006.

[4] R. Chertov and S. Fahmy. "Optimistic load balancing in a distributed virtual environment," Proc. of the 16th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp.1-6, 2006.

[5] Rynson W.H. Lau, "Hybrid Load Balancing for Online Games," Proc. of the International Conference on Multimedia, pp.100-103, 2010.

[6] J. Lui, M. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," IEEE Trans. on Parallel and Distributed System, Vol 13, No 2, pp.193-211, 2002.

[7] P. Morllo et al., "Improving the Performance of Distributed Virtual Environment Systems," IEEE Trans on Parallel and Distributed Systems, Vol 16, No 7, pp.637-649, 2005.

[8] Kang-Won Lee, Bong-Jun Ko, Seraphin Calo, "Adapter Server Selection for Large Scale Interactive Online Games," Proc. of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp.152-157, 2004.

[9] N. Ng, A. Si, R. Lau, and F. Li, "A Multi-server Architecture for Distributed Virtual Walkthrough," Proc. of ACM Symposium on Virtual Reality Software and Technology, pp.163-170, 2002.

[10] Xiaobo Zhou, Cheng-Zhong Xu, "Optimal Video Replication and Placement on a Cluster of VOD Servers," Proc. of the International Conference on Parallel Processing, pp.547-555, 2002.

[11] A. Dan and D. Sitaram. "On line video placement policy based on bandwidth to space ratio (BSR)," In Proc. of ACM International Conference on Management of Data, pp.376-385, 1995.

[12] T. Ibarkai, N. Katoh, Resource allocation problem - Algorithm approaches, The MIT Press, 1988.

[13] A. L. Chervenak, D. A. Patterson, and R. H. Katz, "Choosing the best storage system for video service," In Proc. of the Third ACM International Conference on Multimedia, pp.109-119, 1995.

차 충 용 (Che, Zhong Yong)

2010.8 연변대학 전산학과 학사
2010.9-현재 한국외국어대학교 컴퓨터공학과 석사과정
2010.9-현재 (주)셀로코 인턴사원

관심분야 : 게임프로그래밍, 기능성게임, WSN

김 상 철 (Kim, Sangchul)

1994.5 미시간주립대학교 컴퓨터공학과 박사
1983.3-1994.8 ETRI 연구원
1994.9-현재 한국외국대학교 컴퓨터공학과 교수

관심분야 : 게임프로그래밍, 기능성게임, WSN