Journal of Korea Game Society 2019 Jun; 19(3): 5-14
http://dx.doi.org/10.7583/JKGS.2019.19.3.5

게임 프로그래밍

Proximal Policy Optimization을 이용한 게임서버의 부하분산에 관한 연구

박정민*, 김혜영**, 조성현**
홍익대학교 일반대학원 게임학과*, 홍익대학교 게임학부**
bjmbam12@naver.com, disys17@gmail.com, scho@hongik.ac.kr

A Study on Load Distribution of Gaming Server Using Proximal Policy Optimization

Jung-min Park*, Hye-young Kim**, Sung Hyun Cho**
Game Department, The Graduate School*, School of Games, Hongik University**

요 약

게임 서버는 분산 서버를 기본으로 하고 있다. 분산 게임서버는 서버의 작업 부하를 분산하기 위한 일련의 알고리즘에 의해 각 게임 서버의 부하를 일정하게 나누어서 클라이언트들의 요청에 대한 서버의 응답시간 및 서버의 가용성을 효율적으로 관리한다. 본 논문에서는 시뮬레이션 환경에서 기존 연구 방식인 Greedy 알고리즘과, Reinforcement Learning의 한 줄기인 Policy Gradient 중 PPO(Proximal Policy Optimazation)을 이용한 부하 분산 Agent를 제안하고, 시뮬레이션 한 후 기존 연구들과의 비교 분석을 통해 성능을 평가하였다.

ABSTRACT

The gaming server is based on a distributed server. In order to distribute workloads of gaming servers, distributed gaming servers apply some algorithms which divide each of gaming server's workload into balanced workload among the gaming servers and as a result, efficiently manage response time and fusibility of server requested by the clients. In this paper, we propose a load balancing agent using PPO(Proximal Policy Optimization) which is one of the methods from a greedy algorithm and Policy Gradient which is from Reinforcement Learning. The proposed load balancing agent is compared with the previous researches based on the simulation.

Keywords : Load balance(부하분산), Game server(게임서버), Reinforcement Learning (강화학습)

Received: Mar. 13. 2019 Revised: Apr. 16. 2019
Accepted: May. 7. 2019
Corresponding Author: Hye-Young Kim(Hongik University)
E-mail: disys17@gmail.com

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

MMORPG(Massively Multi-player Online Role-Playing Game Server) 서버는 동일한 가상 공간상에서 대규모 사용자가 동시에 접속해서 상호 작용하는 게임의 서버를 말한다. 이러한 시스템에서는 사용자 규모에 따라 서버에 가해지는 부하가 결정된다.

강화학습은 기계학습의 한 영역으로, 심리학에 기반한 기법이다. 주어진 환경에서 agent가 환경의 상태를 인식해 선택하는 행동에 따라 보상을 최대화하는 행동이나 행동 순서를 선택하는 것이다. 주어지는 환경은 마르코프 결정 프로세스로 정의되어야한다. 일반적인 지도학습과는 달리 입출력의 쌍으로 이루어진 집합이 주어지지 않아도 되고, 잘못된 행동에 대해서 명시적인 정정 과정이 없다. 데이터의 생성과 동시에 학습이 가능하다는 장점이 있다. 그 중, Policy Gradient 기법 중 하나인 PPO(Proximal Policy Optimization)는 환경의 데이터를 샘플링하고 surrogate objective function을 최적화하는 것을 번갈아하는 방법으로 훨씬 간단하게 학습하고, 일반적이다[12]. 따라서 본 논문에서는 화면의 픽셀 데이터가 아닌 환경의 실제 데이터를 이용해 학습하기 위해 PPO를 선택했다.

본 논문에서는 데디케이트 서버를 기반으로 구현하는 MMORPG 서버 시스템에서 발생하는 부하를 적절하게 분산하는 방법에 대해서 조사하고 딥러닝을 부하분산에 적용해보는 연구를 한다. 2장에서는 기존 부하분산의 기준 및 기법들의 장단점을 알아보고, 3장에서 본 논문에서 제안하는 알고리즘에 대해 설명하고, 4장에서 비교분석할 실험 환경과 구현 내용, 실험 결과에 대한 분석 및 결과를 정리하고, 5장에서 연구 결론 및 향후 연구 방향을 정리한다.

2. 관련 연구

2.1 부하 분산

부하 분산이란 네트워크 시스템에서 많은 사용자들의 요청을 나누어 처리하는 것으로 수많은 사용자들의 다양한 요청을 하나의 시스템에서 처리하기 위해 필요한 기법이다[1].

온라인 게임의 품질 결정에는 네트워크 성능이 최우선이고 성능에는 대역폭, 지연성 두 가지 요소가 있다. 대역폭(Bandwidth)은 게임의 확장성(동시 접속자 수)를 나타내며, 지연성(Latency)은 게임의 응답속도를 나타낸다. 이러한 요소들은 한정된 자원이고, 게임에 대한 정보들을 한정된 자원을 적절히 활용해 부하를 분산시키는 기술이 부하 분산이라고 할 수 있다[2]. 자원이란 네트워크 대역폭과 프로세서 성능을 말하며 자원소모의 제일 중요한 요소는 플레이어이다. 플레이어가 게임 공간 안에 한명씩 증가할 때 마다 공유해야할 데이터와 인터렉션이 증가하게 되고, 그에 따라 대역폭 소모량이 증가되고, 처리를 위해 프로세서 사용률이 증가하게 된다. 이를 식으로 나타내면 아래 식(1)과 같다. 자원 소모를 최소화해야 네트워크 부하와 프로세서 성능이 개선되어 서버의 성능이 개선된다. 식의 우변의 각 요소를 조절하는 방법은 대상으로 하는 게임이나 서비스의 특성에 따라 달라지므로 자원관리 기법을 유동적으로 사용할 필요가 있다[2].

$$Resource = M \times H \times B \times T \times P \qquad (eq.1)$$

M : 게임공간에서 처리되는 메시지 수
H : 각 메시지당 목적지 클라이언트 수
B : 목적지별 메시지별 요구 평균 대역폭
T : 패킷의 목적지 도달 시간
P : 메시지당 처리 프로세서 사이클 값

실제 MMORPG 서버에서 부하 분산 연구를 위한 트래픽 수집이 어렵기 때문에 클라이언트 측 스위치에서 트래픽 수집을 하기도 한다. Eq.1 은 트래픽 계산의 한 방법을 보여준다. MMORPG의 특성상 접속자수, 시간대, 플레이어의 행동 패턴에 따른 패킷사이즈와 개수, 대역폭에 다양한 패턴을 보였다[3].

2.2 부하분산 기법

온라인 게임에서 사용자들이 발생 시키는 부하를 어떻게 효율적으로 분산해서 처리하는지, 그에 따른 응답시간은 얼마나 빠른지가 중요하다. 기본적으로 각각 독립적인 서버를 서버군으로 구성해서 분산하는 방법과 하나의 월드를 복제해서 서비스하는 게임 채널 방식이 있다[4]. 첫째의 경우, 각 서버가 전부 독립적이므로 다른 서버에 있는 플레이어들 간의 상호작용이 불가능해 함께 게임을 즐길 수 없다. 두 번째의 경우, 각 채널에서 서비스하는 월드의 구성은 동일하지만, 각 채널에 따라 플레이어의 활동에 따른 데이터는 개별적으로 저장된다[4].

게임 서버는 서비스 하고자하는 게임의 장르와 성격에 따라 부하 분산의 방법이 달라진다. 방(Room)을 구성해서 게임이 진행되는 아케이드 장르나 RTS(Real Time Strategy) 장르, FPS(First-Person Shooter) 장르의 경우 위에서 설명한 서버군이나 게임채널 방식으로도 충분히 가능하다[4]. MMORPG 장르는 존(Zone)분산 방식으로 구현하는 경우도 많은데, 이 방식의 경우, 서버는 다수의 존을 처리하며 서비스를 하고, 각 플레이어가 존을 이동하는 방법을 제한해서 부하를 명확하게 처리할 수 있도록 한다[5].

다른 방식으로는 맵분산 방식이 있는데 전체 게임 월드를 작은 단위로 세분한 뒤에 분산서버에 알고리즘을 이용해 분산해서 처리한다[5]. 주로, 셀(Cell)이라는 단위로 표현하며 각 셀에는 플레이어와 NPC(Non-Player Charater)들, 오브젝트들이 포함된다. 플레이어는 셀을 이동할 때 각 셀이 어느 정도의 범위를 가지고 있는지 알 수 없고, 서버에서 전적으로 처리하는 방식이다. 그러므로 플레이어의 이동에 따라 어느 셀에 부하가 편중될지 예측하기 힘들게 된다. 그러나, MMORPG의 경우 플레이어간의 상호작용이 매우 중요하고, 월드 구성에 필수적인 요소이기 때문에 구성하기 어렵다. 그래서 MMORPG 서버에 대해서는 일반적으로 맵 분산 방식의 연구가 많이 이루어지고 있다[6,9,10,12,13].

2.3 부하분산의 요소

부하 분산의 요소로는 플레이어 가중치, 서버 성능 두 가지 정도로 볼 수 있다. 플레이어 가중치는 클라이언트에서 패킷을 보내는 빈도 수(Pn)와 서버에서 클라이언트로 보내는 패킷의 대상의 수(N)와 크기(S)로 계산될 수 있다.

$$W = Pn * S * (N-1) \qquad (eq.2)$$

이 가중치를 합산하는 방식으로 셀의 가중치(Cw)를 계산할 수 있다.

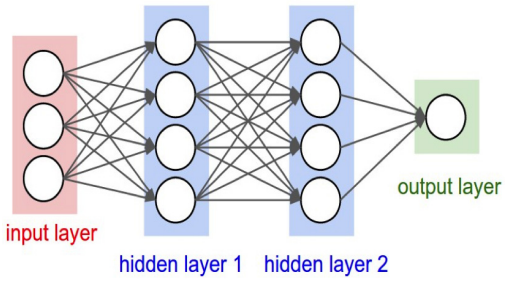
$$Cw = W * N \qquad (eq.3)$$

이 가중치를 기준으로 맵 분산 부하분산을 동작시킨다. 서버의 성능은 계산된 가중치로부터 예측된 최대 부하에 맞추어 임계치를 설정한다.

보통의 게임 서버들이 50~60퍼센트의 CPU 사용률을 기본으로 유지하려한다. 이러한 서버의 최종 성능은 게임 기획, 유저규모, 컴퓨터 사양에 따른 유동적이기 때문에 일률적으로 나타내기는 힘들다. MMORPG 서버의 성능을 단적으로 나타낼 요소는 동시 접속자 수를 기준으로 설정한다[6].

2.4 딥러닝

딥러닝이란 기계학습 분야의 한 갈래로 인간의 신경망을 흉내내서 일련의 동작을 학습하여 작동하는 인공 신경망에서 발전한 분야로 [Fig. 1]과 같이 인공신경망을 여러 층으로 쌓아서 이전 층의 출력을 다음 층의 입력으로 사용해 학습하는 방법이다.



[Fig. 1] DNN(Deep Neural Network)

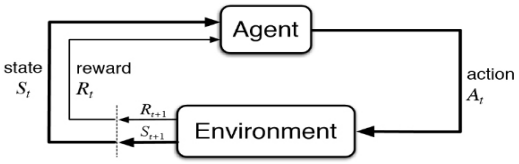
2.5 강화학습

마르코프 의사 결정 프로세스는 동적 프로그래밍과 강화학습 계열에서 문제 해결에 적용하는 모델링 기법으로 해결할 문제를 상태와 행동으로 정의하고 상태사이를 이동하는 확률, 그에 따른 보상을 이용해 학습하거나 문제 해결한다.

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A)) \quad (eq.\ 4)$$

Eq.4 는 강화학습 알고리즘 중, SARSA알고리즘을 나타내는 식이다. 각 상태의 집합 S가 존재하고 결정자가 취할 수 있는 행동의 집합 A가 존재, 상태 S1에서 상태 S2로 어떤 시점 t에서 전이할 확률을 나타내는 변수 α, 상태 전이가 일어났을 때 받을 수 있는 보상 R, 현재의 보상이 미래의 보상보다 중요한 정도를 나타내는 γ의 쌍이다.

[Fig. 2]는 강화학습의 학습 과정을 보인다. 구성한 환경에서 에이전트가 선택한 동작대로 환경이 변화하며 변화한 정도를 계산하여 보상을 주고, 상태를 변화시킨다. 보상이 최대화되는 방향으로 동작을 선택할 수 있도록 학습한다.



[Fig. 2] Reinforcement Learning

2.6 PPO(Proximal Policy Optimization)

강화학습의 한줄기인 Policy Gradient의 발전형인 PPO는 agent가 환경과 상호작용을 통해 데이터를 샘플링 하는 것과 확률적 기울기 상승을 사용해 대리 목표 함수를 최적화 하는 것을 반복하여 학습하는 기법이다.

Algorithm 1 PPO, Actor-Critic Style

```
For iteration=1,2,... do
    For actor=1,2,...,N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1,...,Â_T
    End For
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θold ← θ
End For
```

[Fig. 3] PPO algorithm

[Fig. 3]은 PPO 알고리즘의 한 예를 보여준다. 한 번의 반복에서 모든 actor에 대해 정책 π_{old} 를 따라 T번의 동작을 진행한다. 그 결과로 추정 보상 $\hat{A}_1,\ldots,\hat{A}_T$ 를 계산한다. 모든 actor의 동작이 끝나면 정책과 손실을 반복수와 미니배치의 크기로 최적화하여 정책을 갱신한다[7].

따라서 본 논문에서 화면의 픽셀 데이터가 아닌 환경의 실제 데이터를 이용해 학습하고 간결하고 고성능의 알고리즘 구현을 통해 최소한의 규칙을 발견하고 규칙의 연관성을 발견하기 위해 PPO를 적용하였다.

3. 제안 기법

3.1 부하분산(ProGReGA) 알고리즘을 이용한 부하분산

그리드 구조의 맵을 가진 MMOG 서버에서 부

하분산을 위한 Greedy 알고리즘을 사용한 기법이다. 부하의 기준은 사용자 캐릭터의 상태 업데이트 빈도를 삼고, 합계를 이용해 분산 단위인 셀의 부하 가중치로 사용한다.

각 서버는 게임 월드의 구역을 할당받고, 각 구역은 최소한 하나 이상의 셀을 가진다. 구역의 가중치는 셀들의 가중치의 합으로 나타낸다. 각 셀은 유저 아바타들의 이동에 의해 가중치를 가지며, 각 유저 아바타의 가중치는 유저들 간의 상호작용을 기준으로 계산되기 때문에 접속한 유저의 수에 비례하여 증가한다. 각 유저 사이의 상호작용 부하를 1로 상정하면 접속한 유저의 수가 N 명일 때 유저 1명당 가중치는 N으로 계산된다.

셀(Cell) : 게임 월드를 일정한 크기로 나눈 단위이다. 크기와 위치가 고정되어있고, 경계를 공유하는 셀을 서로 '인접', '이웃'이라고 한다. 본 논문에서는 좌표상 10×10의 크기를 가진 기본 단위이다.

구역(Region) : 연속적인 셀의 집합으로, 서버에 할당되는 기준이다.

서버 : 분산 게임서버의 노드이다. 하나의 구역을 할당받는다.

셀 가중치 : 아바타 가중치의 합으로 나타낸다. 가상 환경에 존재하는 모든 아바타를 고려한다.

$$w_c(C) = \sum_{i=1}^{n} w(Ai) = \sum_{i=1}^{n} \sum_{j=1}^{t} R(A_i, A_j) \quad \text{(eq.5)}$$

구역 가중치 : 셀 가중치의 합으로 나타낸다. 서버 가중치와 같다.

$$w_r(R) = \sum_{i=1}^{p} w_c(Ci) \quad \text{(eq.6)}$$

월드 가중치 : 게임상의 모든 가중치의 합. 가상 환경의 분산에 사용되는 가중치로, 모든 셀의 가중치 합이다.

$$W_{\text{total}} = \sum_{i=1}^{w} w_c(Ci) \quad \text{(eq.7)}$$

서버 처리량 : 업로드 대역폭을 기준으로 설정되는 서버의 처리량이다. p(S) 로 표기한다.

시스템 처리량 : 시스템에 존재하는 서버의 처리량 합으로 나타낸다. P 로 표기한다.

자원 사용량 : 가중치/처리량으로 계산되는 변수로, 서버, 월드 단위에서 사용되는 수치다.

Algorithm 2 ProGReGA

```
1  Weight_to_divide <- 0
2  Free_capacity <- 0
3  For each region R in region_list do
4      Weight_to_divide <- Weight_to_divide + w_r(R)
5
6      Free_capacity <- Free_capacity + p(s(R))
7  End For
8  Sort region_list in decreasing p(s(R)) order
9  For each region R in region_list do
10     Weight_share <- weight_to_divide × p(s(R)) / free_∩acity .
11     While w_r(R) < weight_share  do
12         If there is any cell from R neighboring a free cell then
13           R <- R ∪ {neighbor free cell with the highest Int_c(C)}
14         Else if there is any free cell then
15           R <- R ∪ {heaviest free cell}
16         Else
17           Stop. No more free cells.
18     End while
19 End For
20
```

[Fig. 4] ProGReGA Algorithm

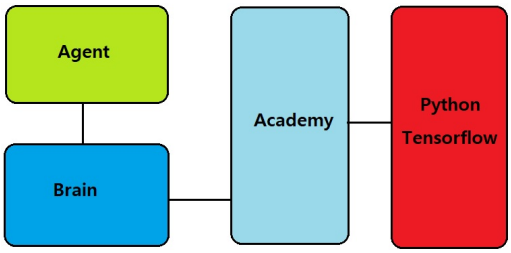
모든 셀을 할당된 지역에서 해제하여 자유 셀로 만든다. 모든 지역의 처리량과 가중치를 합산하여 분산의 입력으로 사용한다. 구역을 처리량의 내림차순으로 정렬하고, 잔여 가중치에 구역 처리량을 남은 처리량으로 나눈 값을 곱해 나온 분산할 가중치를 구한다. 현재 구역의 가중치가 분산할 가중치를 넘지 않는 동안 구역에 선택된 셀을 추가하며 값을 갱신한다. 이때, 셀 선택 기준은 인접 셀 중에 연결 가중치가 가장 높은 자유 셀을 우선 선택한다. 인접 셀 중에 더 이상 선택할 셀이 없거나, 현재 가중치가 분산 가중치 보다 높아지면 다음 구역으로 넘어간다. 이후 모든 구역을 돌며 분산이 마무리될 때 까지 반복된다.

분산 대상이 되는 셀을 선택하는 기준은 셀 사

이의 연결가중치로 결정되며 서버의 처리량을 넘지 않도록 분산한다.

3.2 ml-agents를 이용한 부하분산

우리는 Unity3D의 ml-agents 라이브러리를 이용하여 강화학습을 수행하였다. Unity 3D에 구현된 학습 환경에서 수집되는 변수들을 Tensorflow로 전송해주며 PPO 알고리즘으로 학습된 결과를 Unity 3D로 전송해준다. 이를 반복하며 학습이 진행된다. [Fig. 5]는 Unity3D의 ml-agents 구조도를 나타낸다.

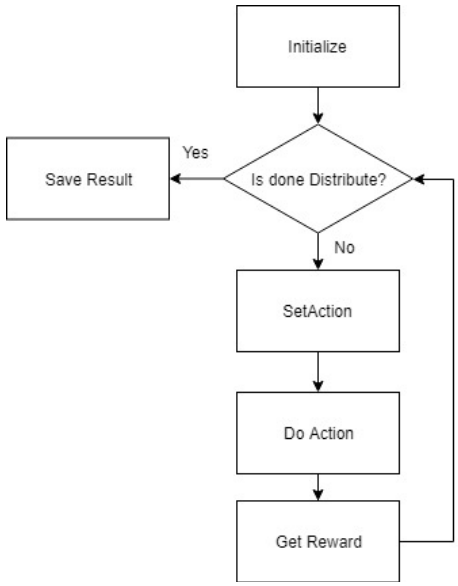


[Fig. 5] ml-agents

각 agent의 행동은 해당 agent가 연결되어 있는 brain에 의해 결정된다. 각 brain은 특정 상태와 행동 공간을 정의하고, 연결된 agent가 어떤 행동을 취할지 결정한다. 현재 릴리스에서는 brain을 다음 네 가지 모드로 설정할 수 있다. 특정 씬에 있는 아카데미 오브젝트는 해당 환경에 포함된 모든 brain을 자식으로 포함하고 있으며, 강화학습을 진행하여 분산 정책을 학습한 모델을 만들어 낸다[8].

[Fig. 6]에서와 같이 Ml-agents에서 학습한 모델의 정책을 기반으로 행동을 선택하고 그에 따른 보상을 얻어 분산 결과에 대한 최종 보상과 합한 값으로 결과를 도출한다.

[Fig.7]은 agent의 행동을 선택하는 기준을 나타낸다. agent가 위치한 셀을 기준으로 사방의 셀을 확인한 후 이웃 셀이 존재하고, 방문하지 않은 셀인지를 탐지하여 선택한다. 모든 이웃 셀을 탐지한 뒤 사방으로 갈 수 없다면 랜덤한 셀을 재선택한다.



[Fig. 6] Proposed Algorithm process

각 행동은 일정한 보상을 소비하며 진행되고 모든 셀이 선택되거나, 상한으로 설정한 학습 단계가 지나면 분산을 중지한다.

Algorithm 3 Decision Making

```
1  Get currnet Cell
2  If Current cell != null
3    Get Current cell`s Edge
4    For all Edge
5        If edge.target != null and edge.target.visited
   == false
6          Neighbor[i] = true
7    End For
8  Set Mask to neighbor
```

[Fig. 7] Decision Making Algorithm

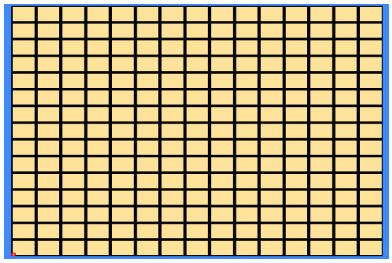
기본적으로 매번 셀을 선택할 때마다 0.01씩 보상을 감소시킨다. 이미 방문한 셀이거나 다른 서버에 포함된 셀을 선택하려 하면 0.05의 보상을 감소시켰다.

또한 잘못된 정책을 거르기 위해 파편화 셀 수에 비례하여 보상을 감소시키고, 서버가 미처 할당이 되지 않은 채 알고리즘이 종료될 때를 대비해 할당되지 않은 그래프 수에 비례해 보상을 감소시켰다.

4. 구현 및 성능 평가

4.1 실험 환경

본 논문의 실험환경은 비교 대상이 되는 논문의 시스템을 모델링한 시뮬레이터를 Unity3D에서 구현하였다.



[Fig. 8] Simulation environment

Fig.8은 시뮬레이션 환경을 나타낸다. 각 사각형 하나는 셀이다. 셀의 집합을 구역이라고 하며 서버 하나에 할당되는 단위이다. 구역의 집합을 월드라고 하며 시스템 전체를 이룬다.

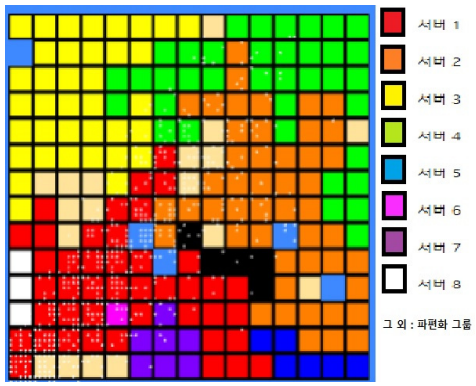
게임 월드는 15*15의 그리드 월드로 총 225개의 셀을 가지며 총 8개의 서버로 분산된다. 사용자의 부하는 FSM(Finite State Machine)으로 동작하는 사용자 AI를 750개 동작시키며 발생시켰다.

또한 서버의 용량은 $i \times 20000$로 정해져있고 이때 i는 1~8사이의 값이다. 사용자 AI 하나의 부하는 모든 사용자와 상호작용하는 것을 기준으로 하고 분산의 결과만을 확인하기 위해 이전 분산 결과에 영향을 받지 않도록 분산한다.

분산 과정을 확인할 수 있도록 각 서버별로 대표 색을 설정하고 분산된 셀의 색을 서버의 색으로 변경한다. 각 사용자 AI는 무작위로 방위를 선택하여 움직이는 시나리오를 선택하였다.

4.2 부하분산(ProGReGA) 알고리즘

분산된 결과를 보면 하나의 서버에 할당된 셀조차 파편화가 일어나는 것을 확인할 수 있다.

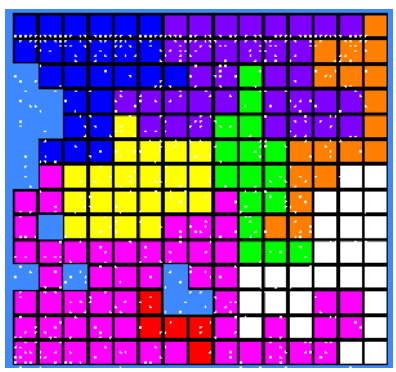


[Fig. 9] ProGReGA Algorithm Distributed result

Fig.9는 시뮬레이션 환경에서 각 알고리즘의 분산 결과를 나타낸다. 서버 1에서 서버 8까지의 색이 아닌 다른 색의 사각형은 분산과정에서 누락된 파편화 셀을 의미한다.

4.3 제안기법 알고리즘

우리가 제안한 PPO 기반의 부하분산 기법은 다음과 같다. 우선, agent를 초기화한다. 이후 agent가 선택하지 말아야할 동작을 현재 셀에 연결된 인접 셀 중에 방문하지 않은 셀을 기준으로 제외한다. 만약 인접 셀을 모두 방문했다면 랜덤으로 위치를 변경한다. PPO로 학습된 모델의 정책대로 제외된 동작을 배제하여, 다음 동작을 선택하고 그 결과를 통해 셀을 할당한다. 이후 초기로 돌아가 반복하며 보상을 최대화하는 정책을 찾는다.



[Fig. 10] Proposed Algorithm distributed result

Fig.10을 보면 Fig.9와는 달리 하나의 서버에서 파편화는 일어나지 않으며 보다 적은 파편화 셀과 파편화 그룹이 나타난다. 즉, 자원 낭비를 줄일 수 있음을 뜻한다.

4.4 실험 결과

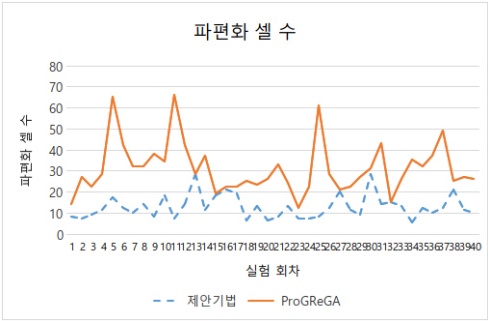
성능 비교 지표는 파편화 그룹 수, 파편화 셀의 수로 나타낸다. 총 40회의 분산결과를 알고리즘 별로 비교하여 나타내었다. 지표들이 적게 나타날수록 자원의 낭비 없이 분산이 잘 이루어졌다고 볼 수 있다. 또한 지표의 변화량이 적을수록 성능이 일정하다고 볼 수 있다.

[Table 1] Fragmented cells and groups

Fragment cell		Fragment group	
proposed	ProGReGA	proposed	ProGReGA
9	15	5	9
8	28	4	18
10	23	7	17
12	29	5	13
18	66	4	10
13	43	7	18
11	33	6	17
15	33	6	17
9	39	5	21
19	35	7	17
8	67	4	16
15	43	6	17
29	29	11	14
12	38	10	17
19	20	4	14
22	23	7	18
20	23	9	8
7	26	5	15
14	24	5	13
7	27	5	15

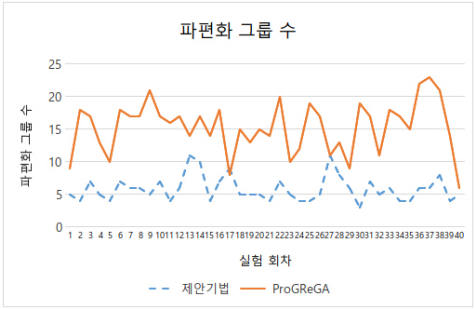
[Table 1]은 각 기법의 실험 결과의 일부를 표로 나타낸 것이다. 각각 강화학습과 ProGReGA 알고리즘의 결과를 파편화 셀과 그룹으로 보여준다.

[Fig.11]에서 X축은 실험 회차, Y축은 파편화 셀의 수이다. 기존의 알고리즘인 ProGReGA 알고리즘의 파편화 셀은 전체 셀의 최소 약 5%에서 최대 약 20% 정도로 나타났다. 그에 비해 제안기법의 경우 파편화 셀이 전체의 약 5%~10%로 나타났다. 큰 편차 없이 성능이 비슷하게 유지될 수 있음을 보인다. 이는 곧, 유저들이 게임을 이용하는 동안 요청 처리에 대한 불만족을 가질 경우가 줄어들 수 있다는 것이다.



[Fig. 11] ProGReGA Algorithm and Proposed Algorithm Fragmented cells

[Fig.11]에서 X축은 실험 회차, Y축은 파편화 셀의 수이다. 기존의 알고리즘인 ProGReGA 알고리즘의 파편화 셀은 전체 셀의 최소 약 5%에서 최대 약 20% 정도로 나타났다. 회차 별 파편화 셀 발생량의 편차가 큰 것은 성능이 일정하지 않다고 볼 수 있다. 그에 비해 제안기법의 경우 파편화 셀이 전체의 약 5%~10%로 나타났다. 큰 편차 없이 성능이 비슷하게 유지될 수 있음을 보인다. 이는 곧, 유저들이 게임을 이용하는 동안 요청 처리에 대한 불만족을 가질 경우가 줄어들 수 있다는 것이다.



[Fig. 12] ProGReGA Algorithm and Proposed Algorithm Fragmented Groups

Fig.12는 각 기법에서 나타난 파편화 그룹 수에 대한 그래프이다. 파편화 그룹의 숫자는 곧바로 서버에 필요한 여유 자원을 뜻한다고 볼 수 있다. ProGReGA 알고리즘의 파편화 그룹 수는 파편화 셀에 비례하여 나타난다. 각 서버에 할당된 그룹에서 발생되는 파편화 그룹도 있지만, 미처 분산되지 못한 셀에 의한 그룹들도 발생하기 때문에 필요한 서버가 늘어난다. 제안기법의 경우, 서버의 성능을 최대한 활용할 수 있도록 분산되는 경향을 보이기 때문에 비교적 파편화 그룹 수가 낮게 발생한다. 그룹 수가 적을수록 추가해야할 서버의 수가 적은 것을 의미하므로 서비스 비용이 절감되는 효과를 볼 수 있다.

제안기법의 분산 과정을 볼 때 ProGReGA 알고리즘보다 분산 시 파편화 경향이 더 낮게 나타났다. 또한 하나의 서버에 분산된 셀들의 밀집도가 높아 클라이언트 간 통신 오버헤드가 줄어든다. [Fig. 11] 과 [Fig. 12]에서와 같이 학습 과정 중에 발견된 몇몇 경우에 파편화 셀이 과도하게 발생하거나 8개의 서버 중에 할당되지 않는 서버가 생기기도 하였고, 모든 분산에서 정상적인 결과가 나오지 않는 경우가 발생하였는데, 이 부분은 실험결과 분석 시에 무시하였다.

5. 결론 및 향후 연구방향

실험의 결과로 볼 때 ProGReGA 알고리즘의 경우, 파편화 셀이 전체 셀 수 대비 5%~20% 가량 발생하고 그만큼의 처리를 위한 자원 낭비가 발생하게 된다. 제안기법의 경우, 최대 분산한 경우는 기존 알고리즘보다 개선된 결과를 나타내었다.

그러나 두 알고리즘 모두 항상 파편화 현상이 발견되는 공통적인 문제점이 있었다. 따라서 더욱 많은 학습 과정과 개선된 행동 선택 알고리즘이 필요할 것으로 보이며. 이후 새로운 행동 선택 알고리즘의 개선과 하이퍼 파라메타 조절, 학습 횟수 증가를 통해 학습 결과를 개선시키고 안정적인 부하분산 기법에 대한 연구를 이어나가고자 한다.

ACKNOWLEDGMENTS

This work was supported by 2016 Hongik University Research Fund and this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1008533 and No. 2016RIA2B4012386).

REFERENCES

[1] Dongil Shin, Dongkyoo Shin, Minsoo Kim, Jaehong Jang, Hyunsook Yoon, Junghoon Lee,Changwan Han, “A Research on Implementation of Load Balancing Middleware for the Onlie Game Server”, The Korean Institute of Information Scientists and Engineers, Vol. 27, No 2-1, p178-180, 2000.10

[2] K.H. Yang, K.H. Shim, D.O. Ko, I.K. Park, J.S. Kim, “Technical Trend of Online Game Server”, Electronics and Telecommunications Trends, Volume 16, No 4, p14-22 2001.8

[3] Lim Soo Jung, Hong Dong Chul, Kim Soo Sung, Kim Sung Joo, Yu Seung Han, Joon Taek Han, Jang Tae Moo, Network analysis in 3D MMORPG online games”, [KOCCA]Research Report, Korea Create Content Agency, 2010

[4] Kang Jung Joong, "Online Game Server", Game Academy at Korea Game Industry Development Institute, p8-16, 2005.6

[5] Jong-Gwan Choi, Hye-Young Kim, Won-Sik Woo, “A Study of a Game User Oriented Load Balancing Scheme on MMORPG”, Journal of Korea Game Society, Volume 12, No 3, p69-76 2012.6

[6] Carlos Eduardo Benevides Bezerra, Cláudio Fernando Resin Geyer, “A Load Balancing Scheme for massively multiplayer online games”, Multimedia Tools and Applications, Volume 45, Issue 1-3, p 263-289 2009.10

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, “Proximal Policy Optimization Algorithms”, ArXiv 2017, arXiv:1707.06347v2, 2017.8
[8] “Introducing: Unity Machine Learning Agents Toolkit”, Unity Blog, 2017.9.19., 2019.3.7, https://blogs.unity3d.com/2017/09/19/introducing-unity-machine-learning-agents/
[9] Beob Kyun Kim, Hang Jin Jang, Kang Soo You, “Load Balancing in Seamless Game with MigAgent”, Journal of Internet Computing and Services, Volume 7, No 6, p51-62, 2006.12
[10] Jeongjin Lee, Gilsoo Doo, Dongun Ann, Seungjong Chung,“Design of Dynamic Map-Divide System for Load Distribution of MMORPG (Massively Multi-player Online Role Playing Game)”, The Korean Institute of Information Scientists and Engineers, Vol 32 , No. 1, p802-804, 2005.7
[11] J.Y. Lim, I.K. Park, J.Y. Chung, K.H. Shim, “Technical Trend of Distributed Game Server”, Electronics and Telecommunications Trends, Volume 20, No 4, p93-102, 2005.8
[12] Ashish Revar, Malay Andhariya, Dharmendra Sutariya, “Load Balancing in Grid Environment using Machine Learning - Innovative Approach”, International Journal of Computer Applications, Volume 8, No.10, p31-34, 2010.10
[13] Carlos Eduardo Benevides Bezerra, João Luiz Dihl Comba, Cláudio Fernando Resin Geyer, “A Fine Granularity Load Balancing Technique for MMOG Servers Using a KD-Tree to Partition the Space”,2009 VIII Brazilian Symposium on Games and Digital Entertainment, 2009.10

박 정 민 (Park, Jung Min)

약 력 : 2017 홍익대학교 게임소프트웨어전공 학사
2019 홍익대학교 일반대학원 게임학부 수료

관심분야 : 온라인 게임, 게임 서버, 게임 클라이언트

김 혜 영 (Kim, Hye Young)

약 력 : 2005 고려대학교 컴퓨터학과 이학박사
2005-2006 Wright State Uni. Post-Doc.
2007-현재 홍익대학교 게임학부 교수

관심분야: 게임서버, 딥러닝 기반 부하분산, 게임엔진, 위치관리기법

조 성 현 (Cho, Sung Hyun)

약 력 : 1978 서울대학교 계산통계학과 이학사
1980 서울대학교 계산통계학과 이학석사
1995 UCLA 컴퓨터과학과 이학박사
1996-현재 홍익대학교 게임학부 교수

관심분야 : 게임프로그래밍, 게임인공지능