KIPS Trans. Comp. and Comm. Sys.
Vol.10, No.4 pp.93~100
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2021.10.4.93

Adaptive Load Balancing Algorithm of Ethereum Shard Using Bargaining Solution

Baek Dong Hwan† · Kim Sung Wook††

ABSTRACT

The Ethereum shard system for solving the scalability problem of the blockchain has a load balancing issue, which is modeled as a graph partitioning problem. In this paper, we propose an adaptive online weighted graph partitioning algorithm that can negotiate between two utility of the shard system using the game theory's bargaining solution. The bargaining solution is an axiomatic solution that can fairly determine the points of conflict of utility. The proposed algorithm was improved to apply the existing online graph partitioning algorithm to the weighted graph, and load balancing was performed efficiently through the design considering the situation of the sharding system using the extension of Nash bargaining solution, which is extended to apply solution to non-convex feasible set of bargaining problem. As a result of the experiment, it showed up to 37% better performance than typical load balancing algorithm of shard system.

Keywords : Ethereum, Blockchain, Sharding, Bargaining Solution, Game Theory, Graph Partitioning, Load Balancing

협상 해법을 이용한 이더리움 샤드 부하 균형 알고리즘

백 동 환† · 김 승 욱††

요 약

블록체인의 확장성 문제를 해결하기 위한 이더리움 샤드 시스템은 부하 균형 문제가 존재하며 이는 그래프 분할 문제로 모델링된다. 본 논문에서는 게임 이론의 협상 해법을 사용하여 이더리움 샤드 시스템의 상반된 효용에 대한 협상이 가능한 적응적 온라인 가중그래프 분할 알고리즘을 제안한다. 게임 이론의 협상 해법은 상반된 효용의 협상점을 공정하게 결정할 수 있는 공리적 해법이다. 제안 알고리즘은 기존 온라인 그래프 분할 알고리즘을 가중그래프에 적용할 수 있도록 개선하였으며 대표적인 교섭 해법인 내쉬 협상 해법을 확장한 확장 내쉬 협상 해법을 사용하여 이더리움 시스템 상황을 고려한 설계를 통해 효과적으로 부하 균형을 수행하였다. 실험 결과, 대표적인 온라인, 오프라인 그래프 분할 알고리즘에 비해 최대 37% 우수한 성능을 보였다.

키워드 : 이더리움, 블록체인, 샤딩, 협상 해법, 게임 이론, 그래프 분할, 부하 균형

1. 서 론

비트코인은 2008년 사토시 나카모토(Satoshi Nakamoto)라는 가명 개발자가 작성한 논문과 함께 소개되었다[1]. 하지만 비트코인은 단순한 화폐수단 이외에는 사용하기 어렵고 거래당 극히 소량의 데이터만 담을 수 있다는 사용성의 한계를 지니고 있다.

비탈릭 부테린(Vitalik Buterin)에 의해 개발된 이더리움은 블록체인의 장점을 이용해 비트코인이 저장하는 화폐 전송기록을 넘어서 계약서 등의 복잡한 추가 정보를 기록하고 다자간의 복잡한 계약 이행까지도 블록체인의 컴퓨팅 자원을 활용하여 실행, 기록하는 시스템이다[2]. 이러한 스마트 계약 시스템을 통해 이더리움은 블록체인 기술을 더 범용적으로 사용할 수 있도록 구현했다. 하지만 블록체인의 근본적인 구조상 이더리움 또한 초당 수십건의 거래조차 처리하기 어려워 범용성에 비해 확장성의 한계가 존재하였다. 이를 해결하기 위해 허가형 블록체인에서는 합의알고리즘을 경량화하였으나[3] 공개형 블록체인인 이더리움에서는 샤드 시스템을 제안하였다[4]. 샤딩은 주된 메인체인(Main Chain)이 존재하고 샤드라 불리는 여러 개의 소규모 그룹을 두어 처리량을 분산시켜 동시다발적으로 처리하는 방법이다. 이더리움 사용자 계정 또는 스마트 계약 계정은 샤드 중 한 곳에 배치되고 계정 간 거래가 일어나면 각 계정이 포함되어 있는 샤드가 통신하여 거래 데이터가 생성된다. 동일 샤드 내에 있는 계정 사이에서 발생한 거래는 거래 집합을 의미하는 콜레이션을

※ 이 논문은 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT연구센터 지원사업의 연구결과로 수행되었음(IITP-2020-2018-0-01799).
† 비 회 원 : 서강대학교 컴퓨터공학과 석사
†† 종신회원 : 서강대학교 컴퓨터공학과 정교수
Manuscript Received : June 17, 2020
First Revision : July 31, 2020
Accepted : September 6, 2020
* Corresponding Author : Kim Sung Wook(swkim01@sogang.ac.kr)

생성함으로써 완료되며 여러 샤드 내에서 동시에 처리가 가능하다. 하지만 서로 다른 샤드에 포함된 계정 사이에서 일어난 교차 거래의 경우는 샤드 간 동기화 등 추가 절차가 필요해 계산, 통신 비용이 더 많이 소모된다[5]. 교차 거래를 최소화하기 위해 계정을 재배치 하게되면 반대로 샤드 별 계산 부하 격차를 발생시키기 때문에 샤드 계산 부하 균형과 상반된 관계에 있게된다. 해당 문제는 자료구조 중 하나인 그래프로 나타낼 수 있으며 적절한 균형을 찾기 위해서는 그래프 분할 문제를 해결해야 한다. 그래프 분할 문제란 그래프를 여러 종속 그래프로 나눌 때 가능한 한 적게 연결되도록 나누고 동시에 각 종속 그래프의 크기는 동일하도록 하는 문제이다. 해당 문제의 최적 해법을 계산하는 것은 정점을 종속 그래프에 배치하는 모든 경우의 수를 계산해야 하기 때문에 다항시간 내에 계산이 불가능한 NP-Hard 문제이다[6]. 따라서 그래프 분할 문제를 풀기 위한 여러 가지 휴리스틱 한 기법을 통해 접근해야 한다. 또한 블록체인의 실시간으로 변하는 그래프 상황을 대응하기 위해서는 최대한 경량 알고리즘을 필요로 하며 동시에 우수한 분할 성능을 가진 그래프 분할 알고리즘을 사용해야한다. 특성상 그래프 분할 알고리즘 수행 시 샤드의 부하가 균일할수록 교차 거래의 수가 많아지고 교차 거래 수를 줄이면 샤드의 부하 균형이 어긋난다. 즉 상반된 관계에 있는 효용함수 사이에 협상이 필요한데, 이때 가장 최고의 협상점을 찾기 위해서는 게임 이론의 협상 해법을 적용하기 적합하다. 게임 이론의 협상 해법은 상반된 효용에 대한 공정한 분배를 공리적으로 증명한 이론이기 때문이다.

본 논문에서는 이더리움의 부하 균형을 위해 샤드의 계산 부하와 교차 거래로 인한 통신 부하 모두를 고려한 그래프 분할 기반 부하 균형 알고리즘을 제안한다. 제안 알고리즘은 게임 이론의 협상 해법을 사용하여 샤드의 계산 부하 효용과 교차 거래에 대한 효용의 최적의 협상점을 결정하여 시스템의 전체 효용을 높인다. 또한 최근 거래 기록과 처리 대기중인 거래를 고려하여 현재 상태에 적합한 알고리즘을 결정해 적응적으로 시스템에 적용한다.

2. 관련 연구

본 장에서는 이더리움 샤드 시스템에 적용 가능한 대표적인 부하 균형 알고리즘을 설명한다. 특히 본 논문에서 제안하는 알고리즘이 기반하고 있는 그래프 분할 알고리즘에 대해 여러 종류를 비교하고 특장점을 설명한다.

2.1 해시 기반 부하 균형

해시 기반 부하 균형(Hash based Load Balancing)은 계정이 지닌 유일한 식별자를 이용해 부하 균형을 하는 방법이다. 해시 알고리즘은 동일한 출력 데이터 크기를 가지며 입력 데이터가 동일하다면 출력 데이터도 반드시 일치하는 특성을 갖는다. 과정이 단순하며 해시 알고리즘 결과 정수 값을 샤드 수로 나눈 나머지 값의 분포가 균일하다면 계정 또한 균일하게 샤드에 배치할 수 있는 방법이다. 하지만 계정의 부하와 교차 거래를 고려하지 않으므로 샤드가 늘어날수록 교차 거래에 대한 부하 또한 심해지는 경향이 있다.

2.2 가스 사용량 기반 부하 균형

가스 사용량 기반 부하 균형(Gas Consumption-aware Dynamic Load Balancing)법은 각 이더리움 계정에서 발생하는 거래의 최대 가스 사용량이 높은 계정부터 가장 여유있는 샤드에 우선 배치하는 방법이다[7-8]. 하지만 해당 알고리즘은 샤드 계산 부하만 고려하며 교차 거래를 고려하지 않는다. 따라서 샤드 개수가 많을수록 교차 거래에 따른 통신 부하가 크게 증가하여 시스템 전체 효율이 낮아지는 문제가 있다.

2.3 그래프 분할 기반 부하 균형

그래프 분할 알고리즘(Graph Partitioning Algorithm)은 본 논문에서 정의하는 이더리움 부하 균형 문제에 대한 해법이다. 크게 오프라인 방식과 온라인 방식으로 나누어지는데 오프라인 방식은 그래프의 정점과 간선이 고정된 상태로 그래프 분할을 진행하며 온라인 방식은 그래프의 정점과 간선이 실시간으로 변경되는 상황에서 정점을 하나씩 즉각 배치한다. 대표적인 오프라인 방식의 알고리즘으로는 커니핸-린(Kernighan-Lin)[9], METIS[10]가 있으며 온라인 알고리즘으로는 LDG[11]와 Fennel[12]이 있다. LDG는 새로 배치할 정점을 각 샤드에 배치했을 때 샤드 별 점수를 계산하여 가장 높은 점수를 갖는 샤드에 배치하는 탐욕 알고리즘이다. Fennel의 경우 LDG와 유사하지만 계수를 이용하여 부하 균형과 교차 거래의 협상점을 시스템 상황에 맞게 조절할 수 있다는 장점이 있다. 온라인 알고리즘은 그래프 전체를 고려하지 않기 때문에 오프라인 알고리즘에 비해 분할 성능은 떨어지지만 수행 시간이 매우 빨라 동적으로 변하는 그래프를 지속적으로 균형 있게 유지해야 하는 상황에 적합하다. Fennel는 온라인 알고리즘이기 때문에 METIS에 비해 분할 성능은 떨어지나, LDG에 비해 개선된 성능을 보이며 METIS에 견줄 수 있는 분할 성능을 보였다[12]. 블록체인의 특성상 블록체인 프로토콜 차원에서 계산되는 모든 결과는 탈 중앙화된 시스템에 의해 도출되어야 한다. 만약 중앙화된 시스템에서 분할 알고리즘을 수행하고 그 결과를 전적으로 신뢰한다면 오라클 문제가 발생하며 특정 계정에 대한 재배치 정책을 고의적으로 조작할 가능성이 있는 등 여러 가지 가용성 및 보안적 측면의 문제를 야기한다[13]. 그래프 분할 과정과 결과를 신뢰하기 위해서는 노드 간의 합의가 필요한데 이더리움에서는 빠른 속도로 거래 데이터가 추가되며 이에 기반한 그래프 모델 또한 실시간으로 변하기 때문에 이에 따른 합의 비용을 줄이기 위해 온라인 알고리즘을 사용하는 것이 적합하다. 단 앞서 언급한 LDG와 Fennel 두 알고리즘 모두 가중치가 없는 무가중치 그래프에만 사용 가능한 알고리즘이라는 단점이 있다. 이더리움의 샤드 시스템은 정점과 간선에 가중치가 존재하기 때문에 본 논문에서는 Fennel 알고리즘을 변형하여 가중치 그래프

에 적용 가능하고 오프라인 알고리즘에 비해 월등히 수행 속도가 빠른 실시간 그래프 분할 알고리즘을 제안한다.

3. 게임 이론 및 제안된 기법

제안 알고리즘을 설명하기 앞서 문제 해결에 사용되는 게임 이론의 협상 해법에 관한 사전 지식과 이더리움 시스템의 구조를 설명한다. 그리고 본 논문이 제안하는 알고리즘에 협상 해법을 사용하여 시스템 효용을 최대화하는 새로운 방법에 관해 설명한다.

3.1 확장 내쉬 협상 해법

내쉬 협상 해법(Nash Bargaining Solution)은 상반된 효용함수의 협상 가능한 집합을 의미하는 협상집합 내에서 가장 공정한 협상점을 공리적 증명을 통해 결정하는 해법이다[14]. 확장 내쉬 협상 해법(Extention of Nash Bargaining Solution)은 기존 내쉬 협상 해법에서 협상집합이 반드시 볼록함수여야 하는 제약을 없앤 더 느슨한 해법이다[15]. 따라서 내쉬 협상 해법의 일부 공리를 불만족하는 대신 최적의 협상점을 도출하는 철학을 유지하여 협상 해를 구할 수 있게된다. 확장 내쉬 협상 해법은 다음과 같은 과정으로 구해진다.

$$L(S,d) \equiv con(N(Co(S),d),d) \neq (\mathrm{S,d}) \equiv \{\max(x) | x \in L(S,d) \cap S\} \quad (1)$$

$con(a,b)$는 a와 b점을 잇는 직선을 의미하며 $Co(S)$는 협상집합 S의 볼록껍질(Convex hull)을 의미한다. $N(S,d)$는 협상집합 S와 협상 실패점 d를 가진 협상 문제의 내쉬 협상 해법을 의미한다. 즉 직선 L은 내쉬 협상 해법과 협상 결렬점을 잇는 직선이다. 해법을 구하는 과정을 그림으로 나타내면 Fig. 1과 같다.

NE(S,d)는 확장 내쉬 협상 해를 의미한다. 즉 협상집합에 대한 볼록껍질을 구하여 효용함수가 볼록함수임을 가정하고 내쉬 해법을 구하는 것이 핵심이다. 그 후 볼록껍질에서의 내쉬 해와 협상 실패점을 이은 선분을 기준으로 기존 협상집합의 파레토 최적점과의 교점을 대체 해로 삼는다. 파레토 최적점은 한 사람의 효용 증가가 반드시 나머지 사람의 효용 감소를 일으키는 지점들을 통칭한다.

3.2 이더리움 샤드 시스템[4]

이더리움은 확장성을 위해 블록체인의 거래 처리를 샤드라는 단위로 나누어 병렬처리한다. 이더리움의 계정은 샤드에 배치되며 해당 샤드를 운영하는 검증자에 의해 처리, 저장된다. 샤드 별 검증자는 블록체인에 참여하는 노드가 임의 배정된다. 그리고 거래의 집합을 뜻하는 콜레이션에 신규 거래를 포함해 자신의 샤드 체인에 추가한다. Fig. 2는 이더리움 샤드 시스템의 구조를 나타내며 Fig. 3은 이더리움 샤드 시스템의 거래를 그래프 모델로 표현한 것을 나타낸다.

Fig. 3의 그래프를 $G(V,E,W)$로 정의하며 V는 정점 집합,

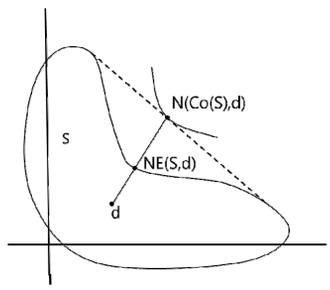


Fig. 1. Extension of Nash Bargaining Solution

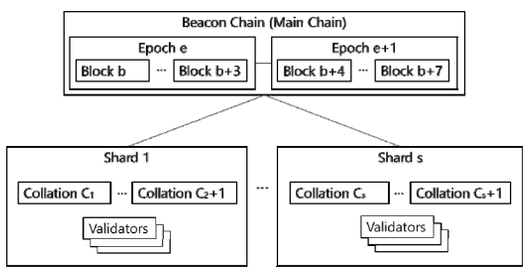


Fig. 2. Architecture of Ethereum Shard System

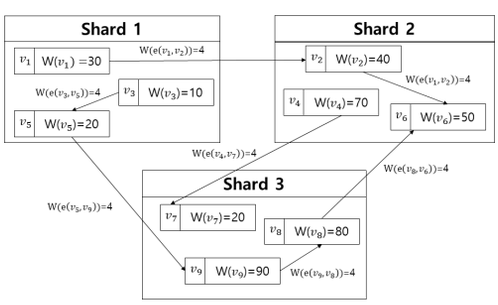


Fig. 3. Graph Model Representing Transactions

E는 간선 집합, W는 정점과 간선의 가중치 함수로 정의한다. 위 그래프에서 각 샤드는 전체 그래프의 종속 그래프로 볼 수 있고 이를 $S \subset G$로 정의한다. 정점 $v \in V$는 이더리움의 일반계정 또는 스마트 계약 계정이며 간선 $e(v_1,v_2) \in E$은 v_1,v_2계정 간 발생하는 거래를 나타낸다. $E(S_i,S_j)$는 두 샤드 i,샤드 j에 포함된 계정들 간 교차 거래를 나타내는 간선 집합이다. 거래의 종류는 일반계정 간의 단순한 송금이거나 또는 스마트 계약의 호출일 수 있다. 스마트 계약 호출인 경우 해당 스마트 계약의 계산 부하에 따라 가스를 지불하므로 정점의 가중치 $W(v)$는 송금 또는 스마트 계약의 실행을 위해 계정 v가 소모하는 가스로 결정된다. 간선 가중치 $W(e(v_1,v_2))$의 경우 계정 간의 호출 회수로 결정된다. 그리고 $W(V)$는 모든 계정에 대한 계산 부하의 합으로 정의하고 $W(E)$는 모든 간선 가중치의 합으로 정의한다.

각 블록과 콜레이션은 특정 시간대에 처리된 거래의 집합이라고 생각할 수 있다. 즉 그래프를 통해 특정 시간대에 일어난 거래를 나타낼 수 있으며 시간이 지날수록 그래프의 정점과 간선 개수 및 가중치는 늘어나게 된다. 이를 나타내기 위해 특정 시간대에 발생한 거래를 나타내는 그래프를 다음과 같이 정의한다.

$$G_t = G(V_t, E_t, W_t) \quad t = (1,2,..,T) \tag{2}$$

t는 블록 번호를 나타내며 블록 번호는 해당 블록에 포함되어있는 거래의 시작과 종료시간을 특정할 수 있다. V_t, E_t, W_t는 각각 블록 번호가 t인 블록에 담긴 거래를 그래프로 나타내었을 때 정점, 간선 집합과 가중치 함수이다. 이를 이용해 최근 k개의 블록에 포함된 거래를 그래프로 나타내면 다음과 같다.

$$\bigcup_{i=0}^{k-1} G_{T-i} \tag{3}$$

이더리움 시스템에 발생하는 거래는 교차 거래가 통신 및 합의비용이 더 높기 때문에 그 거래 발생 수를 제한할 필요가 있다. 검증자들은 현재 처리 대기 중인 거래들 중 가스 비용이 더 높은 거래를 블록에 포함시켜 이득을 최대화하고자 할 것이다. 이는 경매로 모델링 되며 교차 거래인 경우 시장에 의해 가스 비용이 더 높게 형성될 가능성이 매우 높다. 따라서 가스 비용의 불균형에 의해 일반 거래에 비해 교차 거래를 더 처리하게 되면 거래 처리 시간에 대한 불균형을 야기하고 사용자 입장에서 또한 불공평한 시스템이 될 가능성이 높다. 따라서 콜레이션에 포함될 거래들 중 교차 거래의 비중을 제한하여 높은 수수료를 형성하는 교차 거래가 콜레이션의 대다수를 차지하는 것을 막고 일반 거래와 균형 있게 처리되도록 조절한다.

3.3 제안 알고리즘

샤드의 부하 균형을 맞추기 위해서는 앞서 언급했던 방법으로 모델링 된 그래프 분할 문제를 해결하여 계정을 적절한 종속 그래프에 지속적으로 재할당해야 한다. 본 논문에서 사용하는 그래프 분할 알고리즘은 Fennel[12] 알고리즘이 무가중치 그래프에서만 사용할 수 있는 단점을 보완하여 가중 그래프에 적용할 수 있도록 개선하였다. 다음 Equation (4)~(7)는 그 과정을 나타낸다.

온라인 그래프 분할 알고리즘은 샤드의 균형 및 교차 거래의 수를 모두 고려하므로, 분할 알고리즘 P를 수행한 결과에 대한 시스템 비용 함수를 다음과 같이 정의할 수 있다.

$$\begin{aligned} &f(P) = \sum_{k=1}^{|S|} C_{\in} + \sum_{k=1}^{|S|} C_{OUT} \\ &C_{\in} = c(W(S_k)) \\ &C_{OUT} = W(E(S_k, V \backslash S_k)) E(S_i, S_j) = \{e(v_1, v_2) | v_1 \in S_i, v_2 \in S_j\} \\ &k = \{1,2,..,|S|\} \end{aligned} \tag{4}$$

$C_{\in}$는 슈퍼모듈러 특성을 지닌 샤드 내부비용함수, C_{OUT}는 샤드의 외부비용함수이다. $|S|$는 샤드의 개수이며 S_i는 i번째 샤드를 의미한다. $W(E(S_k, V \backslash S_k))$는 샤드 k에서 다른 모든 샤드와의 교차 거래 간선 가중치 합을 나타낸다. 비용함수 f를 최소화하는 문제를 최대화 문제로 치환하기 위해 다음과 같은 함수 h를 정의한다.

$$h(S) = W(E(S,S)) - W(S) \tag{5}$$

함수 h는 샤드 S의 내부 거래에 대한 통신 비용을 보상으로, 샤드 S의 계산 부하 합을 비용으로 해석했을 때 샤드 S의 효용으로 이해할 수 있다. 분할 알고리즘 P를 수행했을 때 모든 샤드의 효용 합 $g(P)$를 다음과 같이 정의한다.

$$\begin{aligned} g(P) &= \sum_{i=1}^{|S|} h(S_i) = \sum_{i=1}^{|S|} (W(E(S_i, S_i)) - W(S_i)) \\ &= W(E) - \sum_{i=1}^{|S|} W(S_i) - \sum_{i=1}^{|S|} W(E(S_i, V \backslash S_i)) = W(E) - f(P) \end{aligned} \tag{6}$$

Equation (4)와 (5)를 이용해 h(S)를 변형하면 g(P)를 최소화하는 문제는 h(S)를 최대화하는 문제에 대응된다. 이를 바탕으로 함수 h를 변형하여 본 논문에서 제안하는 온라인 그래프 분할 알고리즘을 다음과 같이 정의한다.

$$\begin{aligned} &P_{\gamma}^{\alpha}(v) = \arg\max_i \left\{ \begin{matrix} \gamma\left(1 - (W(N_i(v)) + 1)^{-\alpha}\right) \\ -(1-\gamma)\left(\dfrac{W(S_i)}{C}\right)^{\theta} \end{matrix} \right\} \\ &N_i(v) = \{u \mid \exists e(u,v) \ \text{ and } \ u \in S_i\} \end{aligned} \tag{7}$$

$N_i(v)$는 계정 v와 거래가 발생하는 계정 중 샤드 i에 포함된 계정의 집합을 나타낸다. 즉 $(1-(W(N_i(v))+1)^{-\alpha})$는 계정 v와 많은 거래가 일어나는 계정을 포함한 샤드일수록 1에 가까워지며 계수 α를 통해 보상이 증가하는 속도를 제어할 수 있다. C는 샤드의 최대 부하 값으로, $\frac{W(S_i)}{C}$는 샤드 i의 부하 정도를 정규화한 값이다. 제어 계수 θ를 통해 샤드의 부하 비용의 증가 속도를 조절할 수 있다. 즉 좌항은 계정 v를 샤드에 배치하고자 할 때 교차 거래가 가장 적게 발생하는 샤드에 배치하기 위함이며 우항은 부하가 높은 샤드를 피하기 위한 비용이다. 따라서 위에서 언급한 Equation (5)과 동일한 개념으로 이해할 수 있다. γ는 효용과 비용의 비중을 조절하는 계수이다. 큰 값을 가질수록 교차 거래 비용 최소화를 고려하며 작아질수록 샤드의 부하 균형을 더 고려하게 된다. 따라서 시스템 상황에 맞추어 계수를 조절하여 시스템 효용을 높이는 적응적 분할 알고리즘을 수행할 수 있다. 그래프 분할 알고리즘 수행 결과 성능을 객관적으로 평가하기 위해 다음과 같은 두 가지 지표를 사용한다.

$$edgecut(G) = \frac{\bigcup_{i=1}^{|S|} E(S_i, V \backslash S_i)}{m} \tag{8}$$

$$balance(G) = \frac{|S| \times \max_{1 \le i \le |S|} W(S_i)}{\sum_{i=1}^{|S|} W(S_i)} \tag{9}$$

$edgecut(G)$은 전체 거래 중 교차 거래의 비중을 나타낸다. 즉 0에 가까울수록 교차 거래에 따른 통신 비용이 낮다.

$balance(G)$는 균형도로서 전체 샤드의 부하가 균일하게 분포되어 있을수록 1에 가까워진다. 따라서 두 수치를 통해 상대적으로 그래프 분할 알고리즘의 성능을 판단할 수 있으며 샤드 시스템의 계산 부하 불균형에 따른 비용 및 통신 비용을 계산해볼 수 있다. 하지만 실질적으로 이더리움 샤드 시스템의 비용을 계산하기 위해서는 위 두 지표보다 더 많은 변수들을 고려해야 한다. 첫 번째로, 거래에 대한 평균 가스 비용을 예로 들 수 있다. 특정 샤드의 평균 가스 비용은 해당 샤드에서 일어나는 거래에 포함된 가스 비용의 평균 값으로, 일반적으로 해당 샤드의 계산 부하가 심하고 처리 대기 중인 거래가 많을수록 높은 값을 형성한다. 하지만 정확한 값은 시장에 의해 결정되므로 완벽하게 예측하는 것이 불가능하다. 따라서 그래프 분할 알고리즘 수행 시 현재 샤드별 평균 가스 비용의 격차를 고려해 시장 불균형을 최소화해야 한다. 두 번째로는 처리 대기 중인 거래들 중 교차 거래의 비중을 고려해야 한다. 최근 발생한 거래뿐만 아니라 대기 중인 거래를 참고해 반영해야 이후의 효용을 높일 수 있기 때문이다.

이를 기반으로 이더리움 시스템에 대한 계산 부하 균형 효용함수 U_B와 교차 거래 효용함수 U_E를 다음과 같이 정의한다.

$$U_B\left(G,P_\gamma^\alpha\right)=\eta\times B\left(P_\gamma^\alpha,G\right)^\lambda+(1-\eta)\times T_E \tag{10}$$

$$\begin{aligned}U_E\left(G,P_\gamma^\alpha\right)&=\delta\times\left(1-edgecut\left(P_\gamma^\alpha,G\right)\right)^\mu\\&\quad+(1-\delta)\times T_B\end{aligned} \tag{11}$$

$$s.t.,\begin{cases}B\left(P_\gamma^\alpha,G\right)=\beta\times\dfrac{|S|\times\max\limits_{1\le i\le|S|}Gas\left(S_i\right)}{\sum\limits_{i=1}^{|S|}Gas\left(S_i\right)}\\\qquad+\ (1-\beta)\times balance\left(P_\gamma^\alpha,G\right)\\T_B=\dfrac{|Tx_B|}{|pending\,Txs|}=1-T_E\\T_E=\dfrac{|Tx_E|}{|pending\,Txs|}=1-T_B\end{cases}$$

여기서 $Gas(S_i)$는 샤드 i의 평균 가스 비용을 나타내며 $B\left(P_\gamma^\alpha,G\right)$는 현재 시스템의 샤드 계산 부하 균형도와 평균가스 비용에 대한 균형도를 가중 합산한 수치를 나타낸다. 계수 β에 의해 현재 샤드의 부하 균형도와 평균 가스 비용 균형도의 비중을 제어한다. T_B와 T_E는 각각 처리 대기 중인 전체 거래 중 일반 거래 및 교차 거래에 대한 비율을 나타낸다. η는 계산 부하 균형 효용을 계산할 때 샤드의 부하 균형도와 처리 대기 중인 교차 거래 비율과의 효용 비중을 제어한다. δ는 현재 시스템에서 일반 거래가 차지하는 비율과 처리 대기 중인 일반 거래의 비율과의 효용 비중을 제어한다. λ와 μ는 각각 균형도와 일반 거래 비중에 대한 효용의 제어 계수다. 따라서 U_B는 현재 샤드의 계산 부하 균형과 샤드별 평균 가스 비용의 균형이 균일하고, 처리 대기 중인 거래 중 일반 거래의 비중이 낮을수록 효용이 높아진다. 또한 U_E는 현재 교차 거래가 전체 거래 중 차지하는 비율이 낮고 처리 대기 중인 거래 중 교차 거래의 비중이 낮을수록 효용이 높아진다. 두 효용함수를 통해 현재 시스템의 상태를 객관적으로 파악하고 이후 상태를 예상할 수 있으며 이를 기반으로 그래프 분할 알고리즘의 계수를 조절하여 현재 시스템 상태와 이후 일어날 거래를 모두 고려하여 계정 재배치를 수행할 수 있다. 일반적으로 최근에 발생한 거래일 수록 향후 발생할 거래를 예측하는데 영향이 클 것이므로 최근 k 개의 콜레이션을 참조하여 다음과 같이 그래프를 모델링 한다.

$$\begin{gathered}\bigcup_{i=1}^{k}\frac{2\times i}{k(k+1)}\times G_{T-i}\\c\times G(V,E,W)=G(V,E,c\times W)\\c\in R\end{gathered} \tag{12}$$

T는 현재 블록 번호를 나타낸다. 즉 각 콜레이션이 나타내는 그래프의 가중 합산을 통해 최근 일어난 거래일수록 정점과 간선의 가중치를 더 크게 반영한다. 이렇게 그래프의 가중치를 가중 합산하여 모델링한 그래프를 기반으로 위에 언급한 두 효용함수를 계산할 수 있다. 두 효용함수는 상반된 관계에 있지만 협상 집합이 볼록함수를 형성하지 않는다. 따라서 시스템 효용을 최대화하기 위해 앞서 설명했던 확장 내쉬 협상 해법을 사용한다. 또한 계수에 따라 협상 가능점을 계산하기 위해 분할 알고리즘을 수행하는 계산 복잡도가 높기 때문에 계수의 중간 값으로부터 일정 수치만큼 증감하여 샘플링을 통해 근사 해를 구한다. 이를 통해 효용함수가 볼록함수가 아니어도 내쉬 협상 해법의 내쉬 곱을 최대화하는 알고리즘 계수 γ^*을 Equation (13)과 같이 계산할 수 있게된다.

$$\begin{gathered}U=\ U_B\left(G,P_\gamma^\alpha\right)\ \times\ U_E\left(G,P_\gamma^\alpha\right)\\x=\neq(\mathrm{U},0)\\\gamma^*=\arg\min_\gamma(|L(x,U_\gamma)|)\end{gathered} \tag{13}$$

계산된 계수 γ^*를 제안한 그래프 분할 알고리즘 P에 적용하여 현재 시스템 상황 및 처리 대기 중인 거래를 고려하여 계정 재배치를 수행하게 된다. 단 새로운 시스템 상황을 지속적으로 반영하기 위해 해당 계수는 일정 시간 이후 재계산한다.

4. 성능 평가

본 논문이 제안하는 알고리즘의 성능을 현실적으로 평가하기 위해 실제 이더리움 시스템 환경에서 발생하는 거래 부하 및 가스 비용을 참고하여 진행한다. 4.1에서는 성능 평가를 위한 환경 구성에 관해 설명하고 4.2에서는 알고리즘 성능 분석 및 타 알고리즘과의 성능을 비교한다.

4.1 시스템 변수 구성

먼저 이더리움 환경을 구성하는 시스템 변수 중 본 실험과 관련 있는 변수는 Table 1와 같다.

콜레이션 생성 시간마다 각 샤드는 콜레이션에 가스 제한

Table 1. System Variables of Ethereum Shard

Variable	Value
Collation Time	12s~15s
Shard Count	64
Validator Relocation Period	1 epoch (384s)
Gas Limit Per Collation	8000000

Table 2. Variables of Transaction Data

Variable	Value
Consumption Gas per Normal Transaction	21000
Consumption Gas per Smart Contract Transaction	$\mathcal{N}(242962,\ 98108^2)$
Ratio of Cross-shard Transaction	40%
Ratio of Normal Transaction	33%
Number of Shards in Cross-shard Transaction	2 or 3
Number of Unique Accounts per Collation	24
Average Number of Transactions per Collation	110

수치 이내의 거래를 담아 체인에 추가한다. 샤드 개수는 이더리움 샤드 시스템에서 초기값으로 정의하고 있는 64개로 한다. 검증자 재배치 주기는 각 샤드마다 배정되어 있는 검증자 집단을 보안을 위해 섞는 주기를 말한다. 해당 주기에 맞춰 부하 균형 계수를 조정하고 새로운 알고리즘으로 계정을 재배치한다. 이제 콜레이션에 포함되는 거래 데이터를 생성하기 위해 실제 이더리움의 5개월간 거래 정보를 기반으로 송금 거래와 스마트 계약 거래의 가스 비용 분포를 분석하였다[16]. 단순 송금 거래는 고정적으로 21000의 가스 제한 값을 사용하므로 전체 거래 중 송금 거래의 비중을 확인하였다. 또한 해당 기간 동안 사용된 계정은 중복제거 시 블록당 24개였고 거래 개수는 평균 110개였다. 정밀한 모의 거래 데이터를 생성하기 위해 위 통계를 기반으로 Table 2와 같은 지표를 결정하였다.

교차 거래 가스 제한 비율은 콜레이션에 포함되는 거래 중 교차 거래의 가스 합의 비율에 제한이 있음을 뜻한다. 이는 높은 수수료를 갖는 교차 거래의 비중이 너무 높아져 일반 거래의 처리가 지연되어 시스템 효용이 감소하는 것을 막기 위한 수치이다. 거래가 발생하는 계정은 무작위로 선택되며 계정 또한 초기에 무작위로 샤드에 배치된다. 위 기준을 통해 생성된 거래 데이터로 Equation (12)에 의해 그래프를 모델링 한다. 그 후 처리 대기 중인 거래를 다양한 크기만큼 모의 생성하여 시스템 부하 변화를 모델링하고 그래프 분할 알고리즘을 통해 계정을 재배치한다. 이를 통해 모의 생성한 처리 대기 거래를 모두 처리하는데 소요한 시간과 콜레이션 수를 측정하여 알고리즘 성능을 분석한다.

4.2 알고리즘 성능 분석

앞에서 구성한 환경을 기준으로 제안 알고리즘의 성능을 평가한다.

Table 3. Control Parameters of Algorithm

Parameters	Value
α	1.5
β	0.5
δ	0.5
η	0.5
μ	1
θ	2
$Gas(S_i)$	$W(S_i)$
Init_γ	0.5
$Epoch_\gamma$	1
$Step_\gamma$	0.05
$Count_\gamma$	18

알고리즘의 가중치 계수는 환경에 따라 관리자가 실질적인 비용을 반영하기 위해 조절할 수 있다. 실험에 의해 본 성능 분석에서는 계수 및 함수를 Table 3과 같이 정의하고 실험하였다. $Step_\gamma$는 최적의 계수를 구하기 위한 샘플링 정밀도이며 해당 값만큼 계수 γ를 Init_γ를 기준으로 증가, 감소하며 총 $Count_\gamma$번 반복해 U_γ를 계산한다. $Epoch_\gamma$는 계수 γ를 다시 정하기 위한 주기이며 시스템의 변동성에 따라 조절될 수 있다. 해당 기간 동안 계산된 계수를 사용하여 그래프를 분할하여 부하 균형을 수행하고 이후 본 알고리즘은 다시 수행된다. 위 정의를 기반으로 부하 균형과 교차 거래의 통신 비용 협상 실험을 통해 최적의 계수 γ을 찾는다. 의 경우 실제 환경에서 정해지는 시장의 가스 비용을 반영해야 하지만, 시장에 따른 수수료를 정확하게 예측하여 실험할 수는 없으므로 근사한 값을 갖는 샤드의 계산 부하 값을 사용한다. 그래프 분할 알고리즘의 수행 결과는 그래프 모델마다 다른 결과가 나타나고 계수 γ의 변화가 두 효용의 상반된 변화를 반드시 발생시키지는 않으므로 효용 그래프가 완벽한 볼록함수 형태는 아니라는 것을 알 수 있다. 따라서 앞서 논의한 확장 내쉬 협상 해법을 통해 근사 협상점을 구하면 Fig. 4와 같다.

Fig. 4에서 도출된 내쉬 협상점으로부터 가장 가까운 점은 이 0.4 일 때이며 이때 가장 최적의 효용을 얻어낼 수 있었다. 다음은 샤드 시스템의 전반적인 부하 정도에 변화를 주어 상황 변화에 따른 알고리즘의 성능 변화를 분석하고 타 알고리즘과 그 결과를 비교해보았다. 시스템 부하의 변화를 주기 위해 처리 대기 중인 거래의 수를 조절하였고 그 결과 모든 거래가 체인에 추가될 때까지 소모한 콜레이션 수를 확인하였다. Fig. 5-6은 초당 처리 거래 수(TPS), 그리고 누적 콜레이션 수를 나타낸다. 샤드 시스템의 초당 처리 거래 수의 경우 다음 식으로 추정할 수 있다.

$$TPS = \frac{\sum_{i=1}^{|S|} N_{tx}(S_i)}{\max(N} \tag{14}$$

N는 샤드 i의 콜레이션 수, T는 한 콜레이션이 생성되는

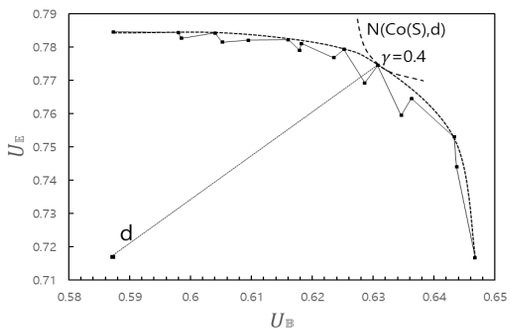


Fig. 4. Feasible Set of Extened Nash Bargaining Solution

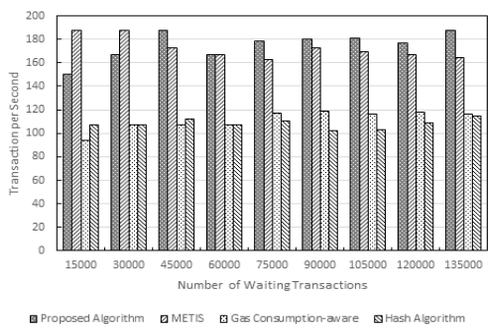


Fig. 5. Average Number of Transactions per Second

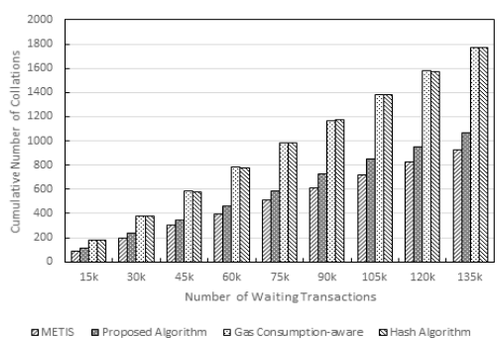


Fig. 6. Cumulative Number of Collations

평균 시간을 뜻한다. $N_{tx}(S_i)$는 해당 시간동안 샤드 i가 처리하는 총 거래 수를 나타낸다. 즉 샤드가 처리하는 총 거래에서 모든 샤드 중 가장 늦게 처리를 마치는 샤드의 소요시간을 나눈 값을 뜻한다. 이를 통해 부하 균형이 효율적으로 이루어졌는지에 대해 측정할 수 있다. 또한 누적 소모 콜레이션 수는 샤드 별로 처리 대기 중인 거래가 모두 완료될 때까지 소모한 콜레이션 수를 합산한 값으로 실질적인 계산 부하 및 합의비용을 나타내는 지표가 된다. 두 지표 모두 낮을수록 효율적인 시스템임을 나타낸다.

결과를 보면 제안된 알고리즘이 해시 기반, 가스 소모량기반 알고리즘에 비해 초당 평균 처리 거래 수는 평균 57%, 누적 소모 콜레이션 수는 평균 37% 더 우수한 성능을 보였다. 오프라인 알고리즘과의 성능 비교를 위해 대표적인 알고리즘

Table 4. Analysis System Environment

Environment	Value
CPU	AMD Ryzen 5 1600 Six-Core 3.20 GHz
RAM	16GB
OS	Ubuntu 18.04
Language	C

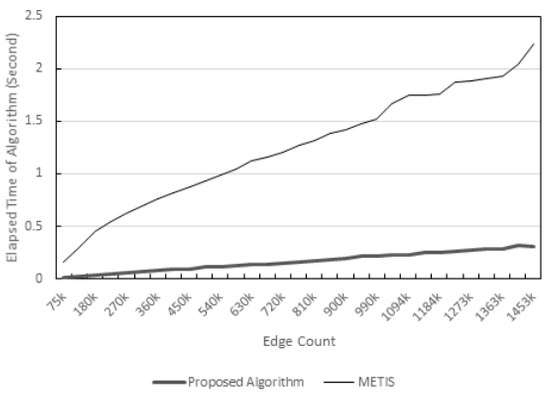


Fig. 7. Elapsed Time of Algorithm

METIS로 그래프 분할을 수행한 결과 METIS가 누적 소모 콜레이션 수에서 근소하게 더 높은 성능을 보여주었다. 하지만 실시간으로 빠르게 변하는 그래프 모델에 적용하기에는 부적합하며 제안 알고리즘과 큰 성능차이를 보이지 않았다.

다음은 대표적인 오프라인 알고리즘 METIS와 수행 시간을 비교해보았다. 성능 분석을 위해 사용한 시스템의 환경은 Table 4와 같다.

샤드는 32개, 정점 개수는 50000개로 고정하고 간선의 개수를 75000개에서 1453000개까지 증가시켜가며 그래프 분할을 수행하였다. 결과는 Fig. 7과 같다.

두 알고리즘 모두 문제 크기가 늘어날수록 소요시간이 증가하였으나 제안 알고리즘이 METIS에 비해 평균 87%의 더 빠른 속도를 보였다. 결과적으로 앞서 진행한 실험과 종합하면 제안 알고리즘이 수행시간 대비 성능이 매우 우수함을 확인하였다.

5. 결 론

블록체인의 확장성을 위한 샤드 시스템은 병렬화를 통해 거래 처리량이 증가한다. 하지만 샤드 간 의존성이 큰 병목현상으로 작용하여 적절한 부하 균형 알고리즘을 필요로 한다. 본 논문에서는 이를 그래프 분할 문제로 모델링 하여 샤드의 부하 균형과 교차 거래 통신 비용 모두를 고려한 적응적 가중 그래프 온라인 분할 알고리즘을 제안하였다. 또한 제안 알고리즘은 시스템의 수수료 시장 상황과 처리 대기 중인 거래를 고려한 효용함수의 협상 해법을 통해 부하 균형을 수행하여 다양한 시스템 상황을 반영하였고 확장 내쉬 협상 해법을 통

해 시스템 전체 효용을 최대화하였다. 그 결과 기존에 제안된 부하 균형 알고리즘인 해시와 가스 소모량 기반 부하 균형 알고리즘에 비해 평균 37%의 성능 향상이 있었다. 또한 대표적인 오프라인 그래프 분할 알고리즘 METIS에 비해 87%의 빠른 수행 속도를 보였다.

References

[1] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System [Internet], https://bitcoin.org/bitcoin.pdf

[2] V. Buterin, "A next-generation smart contract and decentralized application platform," Etherum, No. Jan., pp. 1-36, 2014.

[3] S. Zhang and J. Lee, "Analysis of the main consensus protocols of blockchain," *ICT Express*, Vol.6, No.2, pp.93-97, 2020.

[4] Ethereum Wiki, Sharding introduction R&D compendium [Internet], https://github.com/ethereum/wiki/wiki/Sharding-introduction-R&D-compendium.

[5] Ethereum Wiki, Sharding FAQ [Internet], https://github.com/ethereum/wiki/wiki/Sharding-FAQ#what-is-the-train-and-hotel-problem.

[6] Wikipedia, Graph partition [Internet], https://en.wikipedia.org/wiki/Graph_partition.

[7] S. Kim, J. Song, S. Woo, Y. Kim, and S. Park, "Gas consumption-aware dynamic load balancing in ethereum sharding environments," *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Umea, Sweden, pp.188-193, 2019.

[8] S. Woo, J. Song, S. Kim, Y. Kim, and S. Park, "GARET: improving throughput using gas consumption-aware relocation in Ethereum sharding environments," *Cluster Computing*, 2020.

[7] J. Nash, "Two-Person Cooperative Games," *Econometrica*, Vol.21, No.1, pp.128, 1953.

[8] J. Conley and S. Wilkie, "An Extension of the Nash Bargaining Solution to Nonconvex Problems," *Games and Economic Behavior*, Vol.13, No.1, pp.26-38, 1996. Available: 10.1006/game.1996.0023

[9] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, Vol.49, No.2, pp.291-307, 1970.

[10] G. Karypis and V. Kumar, "MeTiS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 3.0," Univ. of Minnesota, Dept. of Computer Science and Engineering, Army HPC Research Center, Minneapolis, Minn., 1998.

[11] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* - KDD '12, 2012.

[12] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. Microsoft Technical Report MSR-TR-2012-113, 2012.

[13] Hash.Kr, Oracle Problem [Internet], http://wiki.hash.kr/index.php/%EC%98%A4%EB%9D%BC%ED%81%B4_%EB%AC%B8%EC%A0%9C.

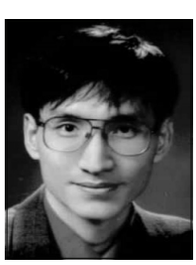
[14] J. Nash, "Two-Person Cooperative Games," *Econometrica*, Vol.21, No.1, pp.128, 1953.

[15] J. Conley and S. Wilkie, "An Extension of the Nash Bargaining Solution to Nonconvex Problems," *Games and Economic Behavior*, Vol.13, No.1, pp.26-38, 1996. Available: 10.1006/game.1996.0023

[16] Etherscan, The Ethereum block explorer [Internet]. http://etherscan.io.

백 동 환

https://orcid.org/0000-0002-8044-1126
e-mail : bdh92123.1@gmail.com
2018년 인하대학교 컴퓨터공학과(학사)
2020년 서강대학교 컴퓨터공학과(석사)
관심분야 : 블록체인, 게임이론



김 승 욱

https://orcid.org/0000-0003-1967-151X
e-mail : swkim01@sogang.ac.kr
1993년 서강대학교 전자계산학과(학사)
1995년 서강대학교 전자계산학과(석사)
2003년 Syracuse University, Computer Science(박사)
2005년 중앙대학교 컴퓨터공학부 조교수
2006년 ~ 현 재 서강대학교 컴퓨터공학과 조교수/부교수/정교수
관심분야 : 게임이론을 이용한 네트워크 자원관리