计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190100238

# 基于跳跃 Hash 和异步共识组的区块链动态分片模型

潘吉飞　黄德才
浙江工业大学计算机科学与技术学院　杭州 310023
(2809956575@qq.com)

**摘　要**　区块链系统的实现方案普遍存在性能和容量上的缺陷,使其无法取得更广泛的普及和应用。分片被视为最有可能解决区块链瓶颈的技术,然而目前主流的实现方案普遍存在牺牲去中心化或者安全性来提升性能的问题。基于现有分片技术的研究,文中提出了基于跳跃 Hash 和动态权重的分片构建算法,该算法满足高效性、公平性、自适应性等特点,网络分片效率对比以太坊提升了 8%,分片数量动态增减时节点迁移的工作量对比以太坊降低了 25%;同时引入了异步共识组机制,提升了分片的交易安全性,能够有效处理跨分片交易。理论分析和实验证明,基于跳跃 Hash 和异步共识组的区块链动态分片模型的最大交易性能可达 5000 笔每秒。

**关键词**:区块链;分片;跳跃 Hash;异步共识组;动态权重
**中图法分类号**　TP315

## Blockchain Dynamic Sharding Model Based on Jump Hash and Asynchronous Consensus Group

PAN Ji-fei and HUANG De-cai
College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China

**Abstract**　The current implementation of blockchain systems generally suffer from performance and capacity deficiencies, making it impossible to achieve deeper popularity and wider application. Sharding is considered as the most likely technology to solve the blockchain bottleneck. However, at present, the mainstream sharding schemes generally suffer from the problem of sacrificing decentralization or security to improve performance. Based on the existing sharding technology, this paper proposed the jump Hash wight asynchronous consensus group scheme, which builds shards based on jump hash and dynamic weights, to improve the efficiency and rationality of shards creation. The algorithm satisfies the characteristics of high efficiency, fairness, and adaptability. The network fragmentation efficiency is improved by 8% compared with Ethereum. The workload of node migration is reduced by 25% compared with Ethereum. The asynchronous consensus group mechanism is introduced to improve the transaction security of sharding and effectively handle cross-shard transactions. Through theoretical analysis and experiments, the maximum transaction performance of blockchain dynamic sharding model based on jump Hash and asynchronous consensus group can reach 5000 transactions per second.

**Keywords**　Blockchain, Sharding, Jump Hash, Asynchronous consensus group, Dynamic weight

## 1　引言

2008 年中本聪发表了比特币白皮书[1],比特币背后的区块链技术以去中心化为宗旨,在过去的 10 年得到了较为蓬勃的发展。区块链技术目前处于深度探索阶段,研究重点在于性能和容量等系统瓶颈的突破[2],以促进区块链得到更广泛的实际应用。

目前,区块链存在较大的性能瓶颈。比特币最大交易吞吐量为 7 笔每秒,等待 6 个区块的可信确认需要约 1 小时的最终确认时间;以太坊的交易速度仅为 20 笔每秒,平均时延为 12 秒[3]。对比 Visa 这类中心化支付方案平均每秒上千笔的交易处理速度,区块链目前存在一定的差距。

吞吐量主要受限于网络带宽和通信延迟,区块链在添加新的区块前,必须保证上一个区块在全网已经得到一定的同步率。对于基于数据中心高速链路的 EOS[4],网络带宽和延迟可以被忽略,账本容量成为制约系统扩展的另一个瓶颈。账本容量本质是单台全节点的内存容量,直接限制了接入的用户量和挂载的 DApp(去中心化应用)数量。在交易验证和执行的过程中,全节点必须将与全网的每一个账户、每一个 DApp 相关的状态存储在内存中,以供交易验证实时访问。当前运行以太坊完整节点需要 150 GB 的磁盘空间[5],并且增长趋势变快,基于多级缓存的技术[6]可以缓

到稿日期:2019-01-29　返修日期:2019-06-18　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:水利部公益性行业科研专项基金(201401044);浙江省基础公益研究计划(GG19E090005)
This work was supported by Ministry of Water Resources Public Welfare Industry Research Special Fund(201401044) and Zhejiang Basic Public Welfare Research Program(GG19E090005).
通信作者:黄德才(hdc@zjut.edu.cn)

解但无法根本解决账本容量问题。

由于单个全节点的带宽和存储资源有限，并且不能对部署全节点机器的配置要求过高(门槛过高容易导致中心化)，因此性能瓶颈和容量瓶颈在目前的单链结构下无法突破。区块链“不可能三角”理论[7]表明要想获得大容量、高吞吐率的设计范式，最有效的方案是横向扩展(Scale-Out)。分片(Sharding)技术就是基于这一思想提出的[8]。

本文的主要贡献如下：

(1)介绍区块链分片技术，分析区块链分片技术的难点和已有的解决方案，对比方案的优缺点；

(2)提出基于跳跃 Hash 和动态权重的分片构建算法，提升分片构建的合理性和分片可维护性；

(3)提出异步共识组概念，有效处理跨分片交易，保障分片的安全性；

(4)实验对比本文提出的分片模型和以太坊、Zilliqa 的性能，在分片数量为 15 时，本文系统的吞吐量为 5 000 tps，是以太坊的 5 倍，是 Zilliqa 的 4 倍；同时交易时延仅为以太坊的 2/5，为 Zilliqa 的 1/2。

## 2 相关工作

### 2.1 分片要求

分片的目标是将网络中的所有节点分成若干个子群体，在子群体之间通过预定义的方法执行原来所有节点都要处理的工作，从而达到提高系统处理能力的效果，其本质上是一个分而治之的策略。分片方案必须切分系统网络带宽和内存容量相关的工作，才能提升系统的性能。分片的具体要求如下。

(1)分片大小限制：为了维护系统的容错性，分片中的节点数不能太少，存在下限值。

(2)分片内部的共识机制：由于节点个数有限，需要同时考虑共识机制的安全性和性能。

(3)分片数据一致性：分片内部交易和跨分片交易时，维护多个分片数据的一致性。

### 2.2 分片构建

区块链分片策略划分为网络分片、交易分片和状态分片 3 个层次[9]。

#### 2.2.1 网络分片

网络分片是交易和状态分片的基础。传统区块链需要抵御 51%的攻击，而在分片机制中仅需要抵御 1%的攻击[10]。分片中节点数目下降，攻击者更容易掌控 51%的算力，当一个分片被完全控制时，将直接影响其他分片和整个网络系统。因此对于分片系统，需要更加注重安全问题。

网络分片常采用 3 种方式，具体介绍如下。

(1)简单 Hash 取模

对节点的某个关键特征值进行 Hash 运算后取模，根据取模结果将节点划分到对应分片。这种方法通过哈希函数计算结果的无规律性，从而达到随机分片的目的。在区块链分片系统中，需要合理高效的机制来确定节点应划分到的分片，同时需要保证分片的安全性和负载均衡。目前，当分片动态增减时，常见的基于简单账户 Hash 的方式或者采用先 PoW 后 Hash 的方式，缺乏扩展性和自适应性。

(2)一致性 Hash[11]

一致性 Hash 算法先求出分片的哈希值，并将其配置到圆环上的第 0～232 号位置；采用同样的方法求出节点间的哈希值，并将其映射到相同的圆环上；从节点映射到的位置开始顺时针查找分片，将节点划分到找到的第一个分片，如果超过 232 号位置仍然找不到分片，则将其保存到第一个分片上，如图 1 所示。带虚拟分片的一致性 Hash 将虚拟分片映射到 Hash 环上，一个分片维护多个虚拟分片状态，当分片数目增加或者减少时，分片负载能继续保持均衡，但代价是分片状态信息的维护需要考虑一致性问题。

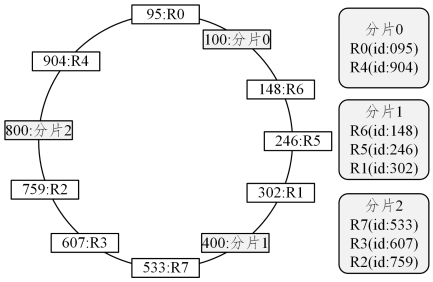


图 1 一致性 Hash 环

Fig. 1 Consistent Hash circle

(3)Range Based[12]

将数据的某个关键特征值划分成不同的区间，每个节点对应一个或多个分片，与一致性 Hash 方式一样，需要维护较多的状态信息。

#### 2.2.2 交易分片

交易分片确定区块链交易的处理模型，包括交易划分到分片的方式和交易的一致性确认方式。主流的交易分片有基于 UTXO 账本系统模型和基于账户系统模型[13]。

(1)基于 UTXO 账本系统模型

在基于 UTXO 账本系统中，一笔交易可以由多个输入和多个输出构成，因此无法按照地址进行交易分片来有效避免双花问题，通常按照交易的 Hash 值的最后几位进行分片。

假设有 4 个分片，如果一笔交易的 Hash 值的最后两位是 10，则将交易划分到第 3 个分片进行处理(Hash 值的末两位 00，01，10，11 分片对应第 1，2，3，4 个分片)。如果交易发起者同时发起另一笔具有相同输入、不同输出的交易(双花交易)，交易 Hash 值的最后两位是 00，则这两笔交易将同时在两个分片中进行验证处理。为了防止造成双花交易，必须进行跨分片通信确认。

(2)基于账户系统模型

在基于账户系统的交易分片中，每一笔交易都包含一个发送者的地址，根据发送者的地址对交易进行划分，促使两笔相同输入的交易在同一个分片中进行验证，从而轻易地识别双花交易，不需要复杂的跨分片的通信。但是，在处理跨分片交易时，基于账户系统的分片模型还是不可避免地要进行跨分片通信。

#### 2.2.3 状态分片

状态分片将系统的存储区分开，每个分片只负责托管本分片的数据，不用存储完整的区块链状态，这使得跨分片通信不可避免。

如果分片遭受重大攻击而处于脱机状态，网络将无法验

证那些依赖于脱机分片的交易。采用存档或分片数据备份的方案，可以帮助系统进行故障修复并恢复那些不可用的数据。然而这样需要一些节点存储整个系统的状态，容易引发中心化的风险。

如何确保跨分片通信不会超过状态分片的性能收益，同时保证数据的高可用性，是状态分片最需要权衡的问题[14]，同时还要考虑节点重新分配分片的场景。

### 2.3 分片技术的研究现状

目前，公有链采用分片技术的代表有以太坊[15]、RChain[16]、Zilliqa[17]和 Monoxide[18]等。

#### 2.3.1 以太坊分片方案

为了解决扩展性问题，以太坊在 2.0 版本中引入了链上状态分区(On-chain State Partition)的概念，将以太网络划分为两层，上层为现有的以太坊主链(Main Chain)，基本保持不变，下层为分片链(Shard Chain)，进行分片交易运算。该系统的特点如下：

(1)分片链上的交易处理在各自独立的空间中进行，由内部的验证器(Shard Validator)验证；

(2)分片链通过信标链机制依附于主链，借助信标链进行更高层次的共识(Higher Level of Consensus)。

该网络结构如图 2 所示。

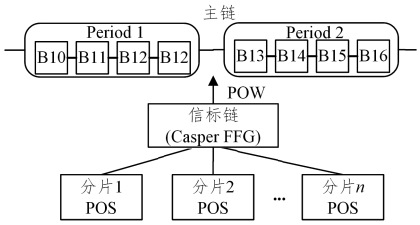


图 2 以太坊 2.0 网络架构

Fig. 2 Architecture of Ethereum2.0 network

以太坊每个分片只保存一部分的历史状态数据，主链只存储分片提交的校验块的头部信息，可以在一定程度上降低节点的存储压力，但在交易量骤增的情况下主网的性能成为系统瓶颈。

#### 2.3.2 RChain 分片方案

RChain 采用独立的智能合约开发语言 RhoLang 和 Rho 虚拟机执行环境，支持并发且多线程处理交易，效率很高。通过命名空间(Namespace)分片技术[19]，将一个事务拆分成多个子集，每个节点处理其中一个子集，最后把这些子集连接起来，组成完整的事务。

不同于以太坊的 Casper 协议，RChain 采用 Casper CBC 协议，不要求每个全节点都参与整个区块的验证，只需检查链上的交易、交易顺序等属性即可，能够方便地分片，提高处理速度。同时，RChain 运用 DAG 技术，进一步提升了系统的可扩展性。图 3 是 RChain 的网络状态图。

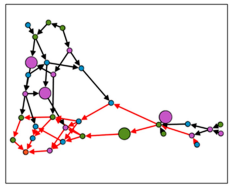

图 3 RChain 网络状态

Fig. 3 Network state of RChain network

#### 2.3.3 Zilliqa 分片方案

Zilliqa 实现了网络分片、交易分片和智能合约分片。区块链网络由 DS 委员会分片和普通分片构成。DS 委员会采用先进先出的策略，每次运行 PoW 共识后，将最快计算出结果的节点加入其中，同时将会有一个节点退出，从而保持 DS 委员会中的节点数量不变。当新的 DS 委员会形成后，其余节点立即执行下一轮 PoW，由 DS 委员会验证各个节点提交的计算结果，根据后几位二进制数将节点分配到不同的分片中，完成 Zilliqa 的网络分片。网络分片实现后，根据交易发送地址的前几位将交易分配到不同的分片中，完成 Zilliqa 的交易分片。Zilliqa 在实现智能合约分片时，将交易进行分类处理。

Zilliqa 采用两轮 EC-Schnorr[20] 多重签名来代替传统 PBFT 共识中的 prepare 和 commit 阶段，将原本的通信复杂度 $O(n^2)$降低为 $O(n)$，共识系统中存在$(n-1)/3$ 个以上恶意节点的概率降低为百万分之一[21]。

Zilliqa 分片方案也存在一定缺陷。由于 Zilliqa 智能合约分片将交易分类后，3 类交易都由 DS 委员会来处理，随着分片数量的增多，交易数量也会急剧增加，因此 DS 委员容易成为分片新的性能瓶颈；为保证分片安全性，Zilliqa 要求分片内的节点数至少为 600，主网正常启动需要 DS 分片和普通分片一共至少 1200 个节点，主网处于高负载状态；Zilliqa 将节点分为两类，即普通节点和种子节点，普通节点只存储用户的状态信息而不存储交易信息，由于目前 Zilliqa 还未实现存储分片，种子节点需要存储完整的账本，随着账本容量的飞速扩张，Zilliqa 将面临严峻的存储问题。

## 3 JHDWS 分片构建算法

### 3.1 跳跃一致性 Hash 算法

跳跃一致性 Hash(Jump Consistent Hash)[10]算法是一种零内存消耗、均匀分配、高效的一致性 Hash 算法，该算法的设计目标为：

(1)平衡性。把节点均匀地分布在所有分片中。

(2)单调性。当分片的数量变化时，只需要把一些节点从旧分片移动到新分片，不需要做其他移动操作。

跳跃一致性 Hash 算法如算法 1 所示。

**算法 1** Jump Consistent Hash 算法

```
Int JumpConsistentHash(int key,int num_buckets):
  random.seed(key);
  int b=-1,j=0;
  while(j<num_buckets):
    b=j;
    double r=random.next();
    j=floor((b+1)/r);
  return b;
```

当节点关键值为 key 和分片数目为 num_buckets 时，JumpConsistentHash(key,num_buckets)返回节点所划分到的分片的编号。

(1)当 num_buckets=1 时，只有一个分片，对任意的 key 都有 JumpConsistentHash(key,1)= 0(从 0 开始编号)，所有节点都划分到 0 号分片；

(2)当 num_buckets=2 时，为了使 Hash 结果保持均匀，

JumpConsistentHash(key,2)有 1/2 的节点保持 JumpConsistentHash(key,1)=0 不变,另外 1/2 的节点跳变到 JumpConsistentHash(key,2)=1,即划分到 1 号分片;

(3)一般地,当 num_buckets 从 $n$ 变到 $n+1$ 后,有 $n/(n+1)$ 的节点所在的分片保持不变,有 $1/(n+1)$ 的节点跳变到 $n+1$ 号分片。假设节点上一次跳变的结果为 $b$,下一次跳变的结果为 $j$,则对于任意的 $i(i\in[b+1,j-1])$,增加分片数量并没有发生跳变的概率为:

$$P(i)=\frac{b+1}{b+2}\times\frac{b+2}{b+3}\times\cdots\times\frac{i-1}{i}=\frac{b+1}{i} \tag{1}$$

在[0,1]区间取一个均匀分布的随机数 $r$,由式(1)得,当 $r<(b+1)/i$ 时,节点就会跳变成 $j$,则 $i$ 的上界为 $(b+1)/r$。由于对于任意的 $i$ 都有 $j\geqslant i$,则 $j=floor((b+1)/r)$。跳跃 Hash 将传统一致性 Hash 算法的时间复杂度 $O(n)$ 降低到 $O(\log n)$。

### 3.2 JHDWS 算法

本文在跳跃一致性 Hash 算法的基础上,提出了基于动态权重的跳跃 Hash 分片构建算法(Jump Hash based Dynamic Weight Sharding Algorithm ,JHDWS)。该算法支持动态权重,并通过构建二维权重矩阵 $\boldsymbol{M}$ 进行节点和分片的映射,提升网络分片的合理性、公平性和自适应性。

假定节点数目为 $n$,分片数目为 $m(m\ll n)$,第 $i$ 个分片表示为 $s_i(i=0,1,\cdots,m-1)$,每个分片的权重 $w_i$ 通过 3 个分量指标进行衡量,即节点负载、分片信用和交易速率,每项指标所占权重如表 1 所列,其中的百分比是实验效果最佳的比例。

表 1 分片权重的 3 个衡量指标

Table 1 3 elements of wieght for one shard

| 指标名称($e'_{ij}$) | 权重($w_j$)/% | 说明 |
|---|---|---|
| 节点负载 | 30 | 负相关 |
| 分片信用 | 30 | 正相关 |
| 交易速率 | 40 | 正相关 |

由于各个分量权重的单位尺度不统一,需要进行规范化处理,分片整体信用值、分片整体交易速率数值的处理公式为:

$$e_{ij}=\left\lfloor\frac{e'_{ij}+1-min}{max+1-min}\times\frac{n}{m}\right\rfloor \tag{2}$$

节点负载数值处理公式为:

$$e_{ij}=\left\lfloor\frac{1}{\log(e'_{ij}+1)+1}\times\frac{n}{m}\right\rfloor \tag{3}$$

式(2)和式(3)中的 $e'_{ij}$ 表示原始数值,$e_{ij}$ 为处理后的数值,$max$ 和 $min$ 分别为该项分量最大数值和最小数值。第 $i$ 个分片的权重计算式为:

$$w_i=\sum_{j=1}^{3}w_{ij}e_{ij}$$

易得 $w_i\leqslant\left\lfloor\frac{n}{m}\right\rfloor$,当且仅当各项 $e'_{ij}$ 都为 0 时取等号(实际情况下不会发生)。$w_i$ 决定每个分片的虚拟分片数量,$w_i=\left\lfloor\frac{n}{m}\right\rfloor$ 表示理想状态下一个节点对应一个虚拟分片,达到完全均匀状态。JHDWS 算法的步骤如下:

(1)构建二维权重矩阵 $\boldsymbol{M}$

根据分片数构建二维加权矩阵 $\boldsymbol{M}[p][q]$,$p\geqslant\sqrt{m}$,$q\leqslant\sqrt{m}+1$,对于任意一个分片 $s$,$\boldsymbol{M}[s/p][s\%q]=w_s$,$\boldsymbol{M}$ 中 $w_s$ 填不满的部分用 0 填充。

(2)虚拟分片和分片的对应关系

对分片权重进行归一化处理后,按比例将其转化成数值在[1,$Q$]内的比例权重,按照比例权重分配虚拟分片数,如分片比例权重为 8,则该分片对应 8 个虚拟分片。将所有分片的虚拟分片编号为 $0\sim\sum_{i=0}^{n-1}w_i-1$,则第 $k$ 个分片对应的虚拟分片序号为$[\sum_{i=0}^{k}w_i,\sum_{i=0}^{k+1}w_i-1]$,即如果知道虚拟分片的编号,就可以通过查表法 *find_in_table*()很快得到分片编号。跳跃搜索算法如算法 2 所示。

**算法 2** 跳跃搜索算法

```
jumpSearch(key,r,num):
  b=-1,j=0;
  random.seed(key,r);
  while (j<num) :
    b=j;
    j=floor((b+1) / random());
  return find_in_table(b);
```

(3)建立节点和分片的映射

JHDWS 分片算法将分片映射到二维矩阵 $\boldsymbol{M}$,采用跳跃搜索算法计算节点在矩阵内的行号和列号,进而确定分片,JHDWS 算法如算法 3 所示。

**算法 3** JHDWS 算法

```
JHDWS(key,r,M):
  total_row=∑(i=0..p-1)∑(j=0..q-1) M_ij;
  row=jumpSearch(key,r,total_row);
  total_col=∑(i=0..q-1) M_row|i
  col=jumpSearch(key,r,total_col);
  clomun=jumpSearch(key,r,col);
  return (row * q+clomun);
```

算法 3 中,key 为 32 位的节点的唯一标识,由 8 位分片地址码和 24 位节点随机码组成,如图 4 所示,key 值在节点申请加入系统时创建;$r$ 为[0,1]区间均匀分布的随机数,随机数生成器采用 64 位的线性同余随机数生成器[22]。

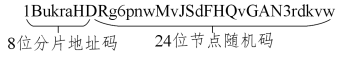


图 4 节点 key 的组成

Fig. 4 Composition of node's key value

### 3.3 JHDWS 算法分析

(1)均衡性分析

对于跳跃搜索算法,任意的虚拟分片划分到一个节点的概率为 $1/\sum_{i=0}^{m-1}w_i$,因此平均每个分片包含的节点数为 $n/\sum_{i=0}^{m-1}w_i$,分片 $s$ 包含的节点数为 $n\times w_s/\sum_{i=0}^{m-1}w_i$,则跳跃搜索算法是均衡的。

JHDWS 算法通过调用跳跃搜索算法来计算节点对应二维权重矩阵中目标分片的行号和列号,搜索目标行时,第 $i$ 行被选中的概率为 $\sum_{j=0}^{q-1}M_{ij}/\sum_{i=0}^{p-1}\sum_{j=0}^{q-1}M_{ij}$,在目标行 $i$ 中搜索到目标列 $j$ 的概率为 $M_{ij}/\sum_{j=0}^{q-1}M_{ij}$,因此节点映射到二维权重矩阵中的对应分片的概率为:

$$P(i,j)=\sum_{j=0}^{q-1}M_{ij}/\sum_{i=0}^{p-1}\sum_{j=0}^{q-1}M_{ij}\times M_{ij}/\sum_{j=0}^{q-1}M_{ij}$$
$$=M_{ij}/\sum_{i=0}^{p-1}\sum_{j=0}^{q-1}M_{ij}$$

其与分片权重成正比，表明 JHDWS 算法满足分片均衡性。

(2)算法复杂度分析

JHDWS 算法的时间开销由构建二维权重矩阵 $\boldsymbol{M}$ 和映射节点组成。由于分片的权重是动态变化的，计算新加入的节点的分片时，需要重新计算二维权重矩阵 $\boldsymbol{M}$ 的元素。单个分片权重计算的时间复杂度是一个常数，因此计算二维权重矩阵 $\boldsymbol{M}$ 的时间复杂度与分片数量 $m$ 线性相关，为 $O(m)$；计算节点分片时，在 $\sqrt{m}\times(\sqrt{m}+1)$ 的矩阵中查找对应分片的行和列，时间复杂度为 $O(\log m)$，因此总的时间复杂度为 $O(m)+O(\log m)$。实际上，分片的权重变化不会太大，因此实验中每隔一个 epoch 时间才会重新计算一次 $\boldsymbol{M}$，计算时间可以被多次节点映射的过程分担，从而忽略不计，则称 JHDWS 算法的时间复杂度为 $O(\log m)$。

(3)节点量迁移分析

分片数量增加或减少时都会引起节点的迁移。每个分片包含的节点数目和分片的权重成正比，因此分片数量变化后分片间仍保持均衡性。

1)增加分片

当系统中的大部分分片都处于高负载状态时，在不破坏原有分片安全性的前提下，通过增加分片数量的方式来提升系统性能，从原有分片中迁移一部分节点到新分片。假设新增分片的权重为 $w_k$，根据分片结束后的均衡性，新增分片最终获得 $n\times w_k/\sum_{i=0}^{m-1}w_i$ 个节点，即所有原来分片总的节点迁移量。

2)减少分片

当分片节点数目低于安全阈值时，取消该分片，将分片内的节点迁移到其他分片。依据跳跃搜索算法的设计，节点划分映射到分片与分片的序号有关，分片失效后，节点会迁移到下一个分片，假设第 $k$ 个分片失效，对应的权重为 $w_k$，则分片内共有 $n\times w_k/\sum_{i=0}^{m-1}w_i$ 个节点需要迁移到第 $k+1$ 个分片，第 $k+1$ 个分片的节点迁移到第 $k+2$ 个分片，直到第 $m-2$ 个分片节点迁移到第 $m-1$ 个分片，则总的节点迁移量为：

$$Work=\sum_{i=s+1}^{m-1}(n\times w_i/\sum_{i=0}^{m-1}w_i)+n\times\Delta w/\sum_{i=0}^{m-1}w_i$$

当 $k=0$ 时，即首个分片失效时 $Work$ 最大。JHDWS 算法通过二维矩阵的方式组织分片，某一行分片失效只会导致行内分片节点迁移，不会导致其他行分片节点的迁移。对于任意一个原本权重为 $w_s$ 的分片，需要迁入 $n\times(w_s/(\sum_{i=0}^{m-1}w_i-w_k)-w_s/\sum_{i=0}^{m-1}w_i)$ 个节点。

## 4 MACG 共识算法

理想的分片共识设计要求各个分片是同质的，功能一致且地位平等，由逻辑上最大程度隔离的共识系统构成，所有分片并行工作，分摊全网的吞吐压力和状态维护。其具体特点如下：1)相对稳定的节点集合，理论上一个节点只隶属于一个分片；2)相对独立且功能完整的区块链，用于记录分片内部的区块信息；3)不需要同步其他分片的所有区块信息；4)低频率的跨分片通信。

每一个分片可以独立地选择自己的共识算法，如 POW，POS，DPOS 或者 PBFT 等改进算法，在分片之间达成一种共识，将分片有效联系起来，组成全局的区块链分片系统。

### 4.1 MACG 算法介绍

本文在基于动态权重的跳跃 Hash 分片架构的基础上，提出了多主节点异步共识组架构(Mutil-Primary-Node Asynchronous Consensus Group Algorithm，MACG)，其分片架构如图 5 所示。首先介绍几个重要概念。

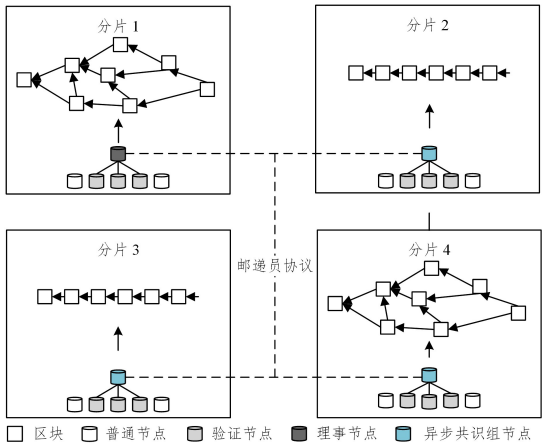


图 5 多主节点异步共识组架构

Fig. 5 Mutil-primary-node asynchronous consensus group

(1)运作周期(Epoch)

MACG 通过验证节点、理事节点和异步共识组共同维护分片区块链，为了确保分片的安全性，每隔一个 Epoch 需要更换验证节点、理事节点和异步共识组节点。

(2)验证节点(Validation Node)

分片初始运作时，选择性能优秀的 $R(R>2m_i/3)$ 个节点作为验证节点。每一轮运作周期结束后，普通节点运行一次 PoW 共识，最先解决问题的节点替换当前性能最差的验证节点，进行重新组合。验证节点需要缴纳一定的押金作为信用保证，如果验证节点被发现作恶，押金将被直接扣除。验证节点通过 PBFT 共识算法来验证分片内的交易是否满足分片一致性，然后将交易打包成一个区块，当超过 $2R/3$ 的验证节点对区块联合签名后，将交易提交给理事节点。分片 $s_i$ 的验证节点表示为 $v_{ij}(j\in[0,R-1])$。

(3)主节点(Primary Node)

每轮 Epoch 开始时，从验证节点中随机选择一个节点作为主节点，同一个节点不得连续担任主节点。主节点用 $d$ 表示，执行 Mailman 协议，具体工作包括：

1)发起异步共识组构建请求；

2)接收验证节点提交的区块，提取区块中所有交易的 Hash 值，构建 $hashSet$ 数据；

3)将 $hashSet$ 发送给异步共识组，以验证交易是否满足全局一致性；

4)交易验证通过全局一致性后，将区块写入分片区块链；

5)当检测到验证节点作恶时，产生一个惩罚区块，废除验证节点之前获得的报酬，扣除押金，并将该验证节点加入分片黑名单。

(4) 异步共识组(Asynchronous Consensus Group，ACG)

分片 $s_i$ 的异步共识组接收 $s_i$ 发送的 $hashSet$ 数据，通过交易的 Merkle 树[23]结构，根据简化支付验证协议(Simplified Payment Verification，SPV)[24]快速验证交易是否满足全局一致性。

首轮异步共识组的产生方法为：分片 $s_i$ 的理事节点随机地选择 $T(T>2n/3)$个分片，加上本分片的主节点，构建一个分片主节点集合，称为分片 $s_i$ 的异步共识组，表示为 $ACG_i=\{d_j|j\in[0,T]\}$。

次轮异步共识组的产生方法为：接收到分片 $s_i$ 发出的异步共识组构建请求后，所有未参与异步共识组的主节点执行一次 PoW，用最先找到正确答案的主节点随机替换原 $s_i$ 分片异步共识组中的一个节点，所有主节点组成集合 $nextACG_i=\{d_j|j\in[0,T]\}$，表示分片 $s_i$ 的次轮异步共识组。

成功打包一个区块获得的收益奖励由验证节点、主节点和异步共识组 3 种节点共同瓜分，依据工作量，验证节点平分收益奖励的 $R/(R+T+2)$，主节点平分收益奖励的 $2/(R+T+2)$，异步共识组平分最后的 $T/(R+T+2)$。

分片一致性验证算法如算法 4 所示。

**算法 4** MACG 算法

```
ShardConsistentValidation
  FOR s_i IN {s_i|i∈[1,m]}
    ACG_i=ACGGenarator();
    block_i=VN_PachTrans();
    VN_MutilSignature(block_i);
    VN_SubmitBlockToPN(block_i);
    hashSet_i=PN_CreateHashSet(block_i);
    PN_BroadcastSetInACG(ACG_i,hashSet_i);
    ACG_Validate(hashSet_i);
    IF (checked) PN_SaveBlock
ENDFOR
```

### 4.2 安全性分析

(1)异步共识组的可靠性

实用拜占庭容错(Practical Byzantine Fault Tolerance，PBFT)算法表明，在作恶节点少于 1/3 的情况下，可以保证系统的正确性(避免分叉)。在 MACG 算法中，每个分片最初选取性能良好的 $R(R>2m_i/3)$个节点作为验证节点，通过 PoW 的方式更新成员，再从全局中选取 $T(T>2n/3)$个分片的理事节点作为异步验证组节点。由拜占庭定理可知，如果异步共识 ACG 是可信的，则 $T$ 个分片中至多存在 $n/3$ 个不可信分片，异步验证组中至少存在$(T+1)/2$ 个诚实节点，即如果验证组中至少存在一半以上的诚实节点，那么异步验证组就是可信的。由于异步验证组的节点在本质上是整个系统的节点执行 PoW 产生的，因此能够有效防止攻击者的女巫攻击，验证组的可靠性非常高。

(2)分片内的安全性

在分片之后可信大多数被稀释，容易造成单个分片的攻击成本极大下降，易实现“1%攻击”。MACG 算法引入了异步共识组机制，分片产生的区块需要得到异步共识组的确认，才能最终被写入分片区块链。在分片超过 51%的节点被控制的情况下，即使篡改了区块链，异步共识组在验证时仍然可以通过之前保存的分片状态信息和区块头信息发现异常，拒绝新的区块加入区块链。如果理事节点在一个 1/3 Epoch 的时间内没有通过验证节点提交的任何区块，验证节点将发送询问区块，一旦理事节点失去响应，就认为理事节点出现故障，验证节点将重新选举理事节点进行工作。同时，MACG 算法通过加大验证节点的作恶代价来防止验证节点违反规则。

(3)分叉的处理

分片内的区块链遵守比特币系统的“最长链原则/算力最多的区块链为主链”规则，如果验证节点私自挖矿，但是由于得不到 2/3 验证节点的签名，即使将打包好的区块发送给理事节点也会被认为是无效区块，理论上不会产生分叉。

### 4.3 性能分析

MACG 算法的整个过程包括选择验证节点、选择理事节点、选择异步共识组和交易验证处理。验证节点和异步共识组都是通过运算 PoW 产生的，分片的理事节点发送异步共识组构建请求总的通信代价为 $O(nT)$。

(1)吞吐率和交易延迟

PBFT 适于用节点数量有限情况下的共识，分片的验证群组节点相对较少，能体现出 PBFT 的优势。一个区块创建后，需要经过两轮验证，验证节点通过 PBFT 共识验证分片一致性，然后提交给异步共识组通过 SPV 快速验证全局一致性，验证通过的区块直接添加进分片区块链，不同于比特币网络需要 6 个区块的确认，其交易延迟得到极大降低，有效提升了系统的交易吞吐率。

(2)跨分片交易处理

节点之间进行交易时需要获知对方的 key 值，由于 key 值的前 8 位是分片地址码，验证节点可以由此判断交易是否出现跨分片情况，如果定义为跨分片交易，则将交易提交给理事节点，由理事节点运行 Mailman 协议发起与目标分片的通信，如图 6 所示。

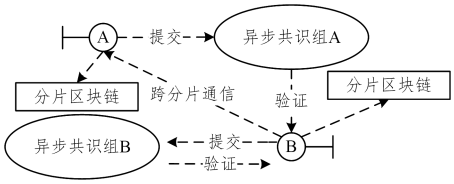


图 6 Mailman 跨分片交易协议

Fig. 6 Mailman protocol for cross shard trading

## 5 实验与结果

本文实验环境为 16 台 1T 存储空间，8G RAM，4 核 CPU 的服务器，服务器之间通过高速网络通信。每台服务器使用 ubuntu 18.04 LTS 系统，采用 Docker 虚拟化技术，每个 Docker 容器代表一个分片，节点通过异步消息机制进行通信，当通信出现较大延迟时表示节点不可信，在任何一个 Docker 容器中有 2/3 以上节点可信就可保证共识机制正常运作。算法采用 Python+Java 语言实现，对比算法为开源项目 Ethereum(分片 2.0 方案)和 Zilliqa。

### 5.1 计算时间

通过服务器构建 10 个分片，每个分片初始化 600 个节点，初始分片负载为 100，块高度初始化为 0，分片信用为 100，交易速率为 20。通过 1 个服务器产生节点，分配节点到各个分片中，分片节点数目从 100 增加到 640。对比 JHD-

WS、Zilliqa 和 Ethereum 的节点分配时间，结果如图 7 所示。JHDWS 构建分片的平均时间比 Ethereum 快 1 s，比 Zilliqa 快 2.5 s。JHDWS 算法在节点加入时依据分片当前的动态权重快速定位。Zilliqa 需要运行 PoW1 算法来选举出 $n$ 个节点，构成 DS 委员会，剩余的节点运行 PoW2 算法来选举出 $m*n$ 个节点，并将其划分到 $m$ 个分片当中，每个分片有 $n$ 个节点，划分时间最长。Ethereum 分片网络分为主链和分片链，其先通过校验器管理建立合约和主链，然后构建分片。

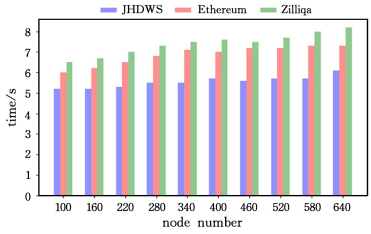


图 7 构建分片的计算时间

Fig. 7 Computation time of building shard

## 5.2 权重标准差

在 5.1 节实验的基础上，动态调整分片的信用值和交易速率，考查 JHDWS，Ethereum 和 Zilliqa 3 种方式的分片权重的标准方差，实验结果如图 8 所示。

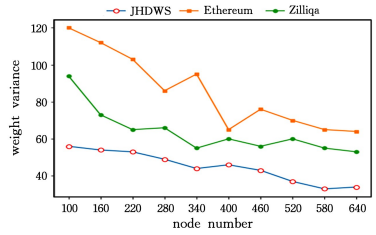


图 8 分片权重的标准差

Fig. 8 Weight variance of shards

Pandora 分片由于在节点划分时综合考虑了分片的多个特性，因此与 Ethereum 和 Zilliqa 相比有最小的权重标准方差。

## 5.3 交易吞吐量

Zilliqa 官网介绍，在测试网络时采用 3 600 个节点、6 个分片，处理能力达 2 488 TPS，它的速度是以太坊最高速度 20TPS 的 100 多倍，是比特币的 355 倍[25]，当节点数量达到数万个时，交易处理速度很有希望达到 VISA 这种中心化机构的速度，即每秒破万笔的交易速度。

本文通过实验对比 MACG 方案、以太坊 2.0 版本和 Zilliqa 三者的交易吞吐量。每个分片有 600 个节点，区块大小限制为 1 MB，分片数量从 2 增加到 15，实验测试 TPS 的增长情况，结果如图 9 所示。

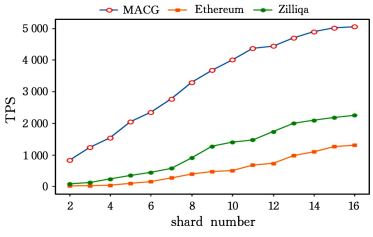


图 9 不同分片数量下的 TPS

Fig. 9 TPS for increasing shard number

当分片数量少于 10 时，MACG 的 TPS 随片数的增加基本呈线性增长；当分片数量大于 10 后，增长变缓，稳定在 5 000 左右，因为随着分片数量的增加，构建异步共识组的通信量也随之增加。

## 5.4 交易延迟

分片数量从 2 增加到 15，每个分片有 100 个节点，区块大小限制为 1 MB，对比 3 种算法的交易延迟，结果如图 10 所示。MACG 具有更低的交易延迟，平均延迟为 8 s 左右。Zilliqa 采用的是 Pow 和 PBFT 混合共识算法，DS 委员会控制整个分片交易的确认，同时在网络分片的基础上采用了交易分片，延长了交易的确认时间。Ethereum 采用 Casper 共识机制，主链上的全节点成为交易的性能瓶颈。

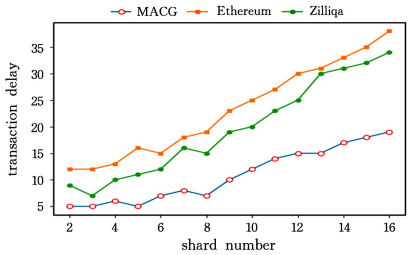


图 10 不同分片数量下的交易延迟

Fig. 10 Transaction delay for increasing shard number

## 5.5 交易吞吐量

采用 10 个分片，每个分片有 200 个节点，区块大小从 1 MB 增加到 8 MB，对比 3 种方案的交易吞吐量，结果如图 11 所示。提升区块大小意味着每个区块打包的交易数量增大，能在一定程度上提升系统的吞吐量。增加区块容量的同时会导致区块构建时间延长、交易验证和通信代价提升，如图 11 所示，随着区块大小的增大，3 种方案的 TPS 均出现先增加后减小的趋势。Ethereum 和 Zilliqa 分别依赖主链和全局的 DS 委员会进行交易处理，MACG 的每个分片都有各自的异步共识组验证交易，因此在前期 MACG 的性能提升最快。当区块大小达到 6 MB 后，通信趋于饱和，进一步加大区块大小并不能继续提升性能。

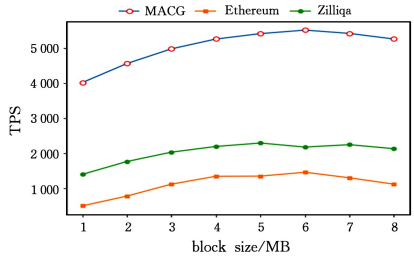


图 11 不同区块大小下的交易吞吐量

Fig. 11 Transaction throughput at different block sizes

**结束语** 分片技术被视为最有希望突破区块链性能和容量瓶颈的解决方案[26]，越来越多的学者开始投入到区块链分片方案的研究中。分片技术应用在公链中面临的主要挑战包括：在分片内需要解决 51%的攻击问题、PBFT 共识的节点数量限制以及女巫攻击；在分片间需要解决双花问题和跨分片交易过载问题；在系统层面需要解决单点过热和分片状态的动态调整问题。

本文在传统分片技术的基础上提出了动态权重的跳跃

Hash一致性算法和多主节点异步共识组算法，能够在提升分片效率和交易速率的同时确保安全性。通过分析和实验验证，本文算法能将区块链的性能提升至5000量级。

由于区块链分片系统的复杂性，本文提出的方案并不完善，需要进行进一步的研究，具体工作包括：智能合约分片和存储分片、更高效的交易验证方式、分片被攻击后数据的恢复、结合实际的应用场景实现技术落地等。

## 参 考 文 献


[1] Satoshi NAKAMOTO. Bitcoin: A Peer-to-Peer Electronic Cash System [EB/OL]. https://bitcoin.org/bitcoin.pdf.

[2] BONNEAU J, MILLER A, CLARK J, et al. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies [C]// 2015 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2015.

[3] HE P, YU G, ZANG Y F, et al. Survey on Blockchain Technology and Its Application Prospect [J]. Computer Science, 2017, 44(4): 1-7.

[4] EOS. EOSIO Technical White Paper v2 [EB/OL]. (2018-3-16). https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

[5] PAN C, LIU ZQ, LIU Z, et al. Research on Scalability of Blockchain Technology: Problems and Methods [J]. Journal of Computer Research and Development, 2018, 5(10): 2099-2010.

[6] CHIKHALE K, SHRAWANKAR U. Hybrid Multi-level Cache Management Policy [C]// 2014 Fourth International Conference on Communication Systems and Network Technologies. 2014: 1119-1123.

[7] Julianbrowne. Brewer's CAP Theorem [EB/OL]. http://www.julianbrowne.com/article/brewers-cap-theorem.

[8] LUU L, NARAYANAN V, ZHENG C D, et al. A Secure Sharding Protocol For Open Blockchains [C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16). 2016: 17-30

[9] KARGER D, LEHMAN E, LEIGHTON T, et al. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web [C]// Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. New York, USA, 1997: 654-663.

[10] LEE B, JEONG Y, SONG H, et al. A scalable and highly available network management architecture on consistent hashing [C]// IEEE Globecom. 2010.

[11] CASTRO M, LISKOV B. Practical Byzantine Fault Tolerance and Proactive Recovery [J]. ACM Transactions on Computer Systems Association for Computing Machinery 1999, 20(4): 398-461.

[12] PAUL J, TOOSKA D, REZA M, et al. Parizi, Kim-Kwang Raymond Choo, A systematic literature review of blockchain cyber security [J]. Digital Communications and Networks, 2018, 34(6): 154-168.

[13] WANG M M, WU Q H, QIN B, et al. Lightweight and Manageable Digital Evidence Preservation System on Bitcoin [J]. Journal of Computer Science & Technology, 2018, 33(3): 568-586.

[14] Vitalik. EthereumSharding [EB/OL]. https://github.com/ethereum/sharding/blob/develop/docs/doc.md.

[15] Rchain. The Rchain Technical Whitepaper [EB/OL]. (2017-8-29). https://www.chainwhy.com/upload/default/20180620/c982b5a83e4fde627ca2dda33f3263bc.pdf.

[16] ZILLIQA. The ZILLIQA Technical Whitepaper [EB/OL]. https://docs.zilliqa.com/whitepaper.pdf, 2017-08-10.

[17] WANG J P, WANG H. Monoxide: Scale Out Blockchain with Asynchronized Consensus Zones [C]// Networked Systems Design and Implementation. Boston MA, USA, 2019.

[18] LAMPING J, VEACH E. A Fast, Minimal Memory, Consistent Hash Algorithm [J]. arXiv: 1406.2294, 2014.

[19] Mitclub. MIT whitepaper. pdf [EB/OL]. https://github.com/Mitclub/Documents/ blob/master/whitepaper.pdf.

[20] AL-BASSAM M, SONNINO A, BANO S, et al. Chainspace: A Sharded Smart Contracts Platform [C]// Network and Distributed System Security Symposium (NDSS). San Diego: IEEE Press, 2018.

[21] EYAL I, GENCER A E, SIRER G, et al. Bitcoinng: A scalable blockchain protocol [C]// 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2016). Santa Clara, CA, USA, 2016.

[22] SHAO Q F, JIN C Q, et al. Blockchain: Architecture and Research Progress [J]. Chinese Journal of Computers, 2017, 41(5): 971-988.

[23] YUAN Y, NI X C, ZENG S, et al. Blockchain Consensus Algorithms: The State of the Art and Future Trends [J]. Acta Automatica Sinica, 2018, 44(11): 2011-2022.

[24] YIHUA D, JAMES W, PRADIP K, et al. Scalable practical byzantine fault tolerance with short-lived signature schemes [C]// Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering (CASCON'18). 2018: 245-256.

[25] JAKOBSSON M, LEIGHTON T, MICALI S. Fractal Merkle Tree Representation and Traversal [C]// Topics in Cryptology—CT-RSA. 2003: 314-326.

[26] MIN X P, LI Q Z, KONG L J, et al. Permissioned Blockchain Dynamic Consensus Mechanism Based Multi-Centers [J]. Chinese Journal of Computers, 2018, 41(5): 1005-1020.


**PAN Ji-fei**, born in 1993, postgraduate. His main research interest include data mining and artificial intelligence.

**HUANG De-cai**, born in 1958, Ph. D, professor, doctoral spervisor. His main research include database, data mining, artificial intelligence and so on.