软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(6):1577−1593 [doi: 10.13328/j.cnki.jos.005737]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

高性能联盟区块链技术研究*

朱 立[1], 俞 欢[2], 詹士潇[2], 邱炜伟[2], 李启雷[3]

[1](上海证券交易所 技术规划与服务部,上海 200120)
[2](杭州趣链科技有限公司,浙江 杭州 310000)
[3](浙江大学 软件学院,浙江 杭州 310000)
通讯作者: 李启雷, E-mail: liqilei@zju.edu.cn

摘 要: 以上海证券交易所“去中心化的主板核心交易系统”作为业务场景,旨在研究高性能联盟区块链的优化算法.在联盟链关键技术研究的基础上,结合现有主板证券竞价交易系统的业务,提出了系统架构以及关键技术的实现.对业务逻辑与共识分离、存储优化和数字签名验证优化(包括合并验签和 GPU 加速)等可提高联盟链性能的优化策略进行了详细的介绍和分析.最后,通过一系列对比实验来验证优化策略的有效性.实验结果表明,这些优化手段极大地提高了去中心化的主板核心交易系统的性能.

关键词: 联盟链性能;业务逻辑与共识分离;合并验签;GPU 加速;存储优化
中图法分类号: TP311

中文引用格式: 朱立,俞欢,詹士潇,邱炜伟,李启雷.高性能联盟区块链技术研究.软件学报,2019,30(6):1577−1593. http://www.jos.org.cn/1000-9825/5737.htm
英文引用格式: Zhu L, Yu H, Zhan SX, Qiu WW, Li QL. Research on high-performance consortium blockchain technology. Ruan Jian Xue Bao/Journal of Software, 2019,30(6):1577−1593 (in Chinese). http://www.jos.org.cn/1000-9825/5737.htm

Research on High-performance Consortium Blockchain Technology

ZHU Li[1], YU Huan[2], ZHAN Shi-Xiao[2], QIU Wei-Wei[2], LI Qi-Lei[3]

[1](Technical Planning and Service Department, Shanghai Stock Exchange, Shanghai 200120, China)
[2](Hangzhou Qulian Technology Ltd., Hangzhou 310000, China)
[3](College of Software Technology, Zhejiang University, Hangzhou 310000, China)

Abstract: This study takes the “securities trading system” of the Shanghai Stock Exchange as a business scenario to study the optimization algorithm of the high-performance consortium blockchain. based on the research of the key technologies of the consortium blockchain and the business of the securities transaction system, this study proposes a design of the consortium blockchain architecture and conducts a detailed analysis of the key technologies that can improve the performance of the consortium blockchain, such as separation of business logic and consensus, optimization of storage, and optimization of digital signature verification (including merger verification and GPU acceleration). Finally, the study conducts a series of comparative experiments to verify the effectiveness of these optimization strategies.

Key words: consortium blockchain performance; separation of business logic and consensus; merger verification; GPU acceleration; storage optimization

区块链是由中本聪于 2008 年提出的一种支持比特币运行的底层技术,其去中心化、可追溯、信息不可篡改等特性,给金融服务等一系列行业带来了重大影响.区块链本质上是一个去中心化的数据库,是分布式数据存

* 本文由区块链与数字货币技术专题特约编辑斯雪明教授和陈文光教授推荐.
收稿时间: 2018-04-11; 修改时间: 2018-10-12; 采用时间: 2018-12-18; jos 在线出版时间: 2019-03-27
CNKI 网络优先出版: 2019-03-27 16:40:16, http://kns.cnki.net/kcms/detail/11.2560.TP.20190327.1640.003.html

储、点对点传输、共识机制、加密算法等计算机技术的新型应用模式.区块链将给金融业、社会生活、政府管理等各方面带来变化,尤其是在金融服务业,区块链技术可能会是金融业下一个拐点.2016 年,Oliver Wyman 国际咨询公司在一份报告“区块链在资本市场”[1]中指出:目前,银行每年的 IT 和运营支出接近 1 000 亿~1 500 亿美元.此外,交易后和证券服务费用也在 1 000 亿美元左右.由于目前市场运作低效率问题,导致了严重的资金损失和流动成本.

分布式账本技术通过减少重复流程、结算时间、抵押要求和业务开销,为涉及的实体节省开支.近年来,各国证券交易所都在积极探索区块链技术在市场基础设施领域上的应用,旨在利用分布式账本技术来大幅度降低交易成本和系统复杂性,并提高交易和结算速度,从而彻底改变全球资本市场的基础设施体系,带来更大的透明度和交易效率.

2016 年 12 月,日本银行(BoJ)和欧洲央行(ECB)成立了 Stella 项目,目的是为了研究分布式账本系统是否能够取代当前日本央行和欧洲央行部署的实时全额结算系统(RTGS).2017 年 9 月发布了对分布式账本技术(DLT)的评估报告[2],报告指出:当节点数量增加,DLT 方案将会导致网络中交易结算时间的延长,验证节点之间的距离也会对性能产生影响.两家银行基于 Fabric 进行了相关测试,使用正常智能合约时,其 TPS 是 10~70 之间,最高可达到 250TPS.报告得出结论:目前,DLT 方案能够符合大额价值支付系统的性能需求,但当前 DLT 技术还不够成熟,不应被用于替代两家央行的现有系统.

日本交易所集团(JPX)联合东京股票交易所、大阪交易所以及日本证券清算集团组建了区块链联盟,旨在测试一种区块链基础设施概念验证.2016 年 2 月,JPX 宣布与 IBM(日本分部)合作测试区块链技术,将使用 IBM 的 Fabric 区块链平台作为测试实验的基础.2016 年 8 月底,JPX 发布了相关工作报告[3],JPX 分析并对比了 HyperledgerFabric,R3Corda 等不同的分布式账本技术如何帮助公司提高交易效率,以及该技术在金融服务领域的可行性.

2015 年,证券交易巨头纳斯达克推出了他们的区块链产品——Nasdaq Linq.2017 年 5 月,纳斯达克和花旗银行宣布采用初创企业 Chain 的分布式账本技术来整合支付方案.Linq 能够展示如何在区块链技术上实现资产交易,同时也是一个私人股权管理工具,是纳斯达克私人股权市场的一部分.目前,Linq 还处于概念探索阶段,性能还不能满足 Nasdaq 主流业务的需求.

2016 年初,主流区块链的吞吐量依然比较慢,比如,比特币一秒钟只能完成 7 笔交易.目前,以太坊真实吞吐量大约是 10 笔/s 左右.通过调整区块大小,其吞吐量可以达到 30~40 笔/s.但与此同时,调整区块大小也会带来一系列负面影响,比如叔区块率增加以及矿工收益降低等.随着一年以来的技术发展,尤其是共识算法和分片(sharding)技术的逐步成熟,使得区块链的性能得到了较大的提升.一些项目组宣称,自己可以达到上万级别的吞吐量,但离开业务背景和测试环境配置,单纯提 TPS 大小是不具备说服力的.因为证券交易业务是一个很好的压力测试场景,因此,本文将以集中交易的订单簿的撮合作为业务背景,然后通过控制各种软硬件配置,尽可能去探索联盟区块链的性能上限.

本文旨在研究高性能联盟区块链的优化算法,结合上交所主板证券竞价交易系统的业务和现有联盟链技术,针对实际业务场景,对联盟链的共识架构、数字签名验证和存储进行优化,并进行性能基准测试,验证优化手段的效果.本文的主要贡献如下.

(1) 采用了业务逻辑与共识分离的架构,将最为耗时的交易执行操作转交给上交所现有的撮合系统,不再采用耗时的智能合约来执行业务.与此同时,共识过程不再包括业务逻辑的执行,只需负责交易的定序和交易内容的校验,从而简化了共识流程,并提升了执行的速度;

(2) 我们对存储模块进行了优化,由于本文采用了业务执行和共识分离策略,并使用上交所外部业务系统来执行交易,外部系统数据库将存储订单流水和账户信息,故区块链底层平台不再存储这部分信息(只需存储区块信息),用户交易流水和账户信息的查询功能也相应地转移给了上交所的外部系统.机构查账或审计时才会从区块中解析出交易信息.考虑到机构查账或审计是一种低频率事件,且延时要求低,所以我们对区块信息的存储方式进行了优化.不再采用 levelDB 存储区块信息,而是使用顺序写

文件的方式存储区块信息,从而大幅度提高了写性能,更加符合业务场景的需求;

(3) 提出了两种数字签名验证的优化策略,分别是多订单打包验签和基于 GPU 加速的椭圆曲线验签算法.证券交易存在 OPS(orders per second)的概念,一个交易(transaction)里面通常打包了若干笔订单(order),对这些订单只需要进行一次签名,并打包成单个交易一起发送.同时,我们还发现,验证交易的签名是整个系统处理流程中较为耗时的一个步骤.故多订单打包验签可以将共识节点签名和验签的开销平摊在交易中的每笔订单上,从而降低总体的开销.同时,合并订单还减少了节点间的通信量,降低网络带宽的负担.此外,经过对椭圆曲线验签算法的分析,我们从有限域运算和椭圆曲线标量乘法运算这两方面进行了优化,提出了一种全新的快速有限域求模运算,并在此基础上实现了椭圆曲线标量乘法的并行运算.

本文第 1 节对区块链(尤其是联盟链)相关工作进行回顾.第 2 节介绍基本算法和实现.第 3 节展示实验结果并进行实验结果分析.第 4 节总结本文工作内容,并规划未来的工作.

1 相关工作

共识算法和安全机制是联盟链的核心要素,下面我们将介绍这两方面的相关研究.

(1) 共识算法

1982 年,Lamport 等人提出了拜占庭将军问题(Byzantine generals problem)[4],它是一个分布式计算领域的问题,设法建立具有容错性的分布式系统,即:在一个存在故障节点和错误信息的分布式系统中,正常节点达到共识,保持信息传递的一致性.区块链共识层的作用就是在不同的应用场景下,通过使用共识算法,在决策权高度分散的去中心/多中心化系统中,使得各个节点高效地达成共识.

最初,比特币区块链选用了一种依赖节点算力的工作量证明共识(proof of work,简称 POW)机制来保证比特币网络分布式记账的一致性.随着区块链技术的不断演进和改进,研究者们陆续提出了一些不过度依赖算力而能达到全网一致的算法,比如权益证明共识(proof of stake,简称 POS)机制、授权股份证明共识(delegated proof of stake,简称 DPOS)机制、实用拜占庭容错(practical Byzantine fault tolerance,简称 PBFT)算法等.

然而,POW,POS 和 DPOS 等共识机制不太适合联盟链的应用场景,联盟链中通常采用 BFT(Byzantine fault tolerance)类共识机制.这是因为在恶意节点数不超限制的前提下,BFT 类算法可以支持较高的吞吐量和极短的终局时间,其正确性和活动性又可被严格证明,非常符合大机构的需求.经典的 PBFT 算法及其变体是最为常见的联盟链共识算法,PBFT 算法最初出现在学术论文中[5],初衷是为一个低延迟存储系统所设计,降低算法的复杂度.它解决了原始拜占庭容错算法效率不高的问题,将算法复杂度由指数级降低到多项式级,使得拜占庭容错算法在实际系统应用中变得可行.PBFT 可以应用于吞吐量不大但需要处理大量事件的数字资产平台,它允许每个节点发布公钥,任何通过节点的消息都由节点签名,以验证其格式.PBFT 是首先得到广泛应用的 BFT 算法,随后,业界还提出了若干改进版的 BFT 共识算法[6−10].

文献[6]提出了一种可伸缩的故障容忍方法,系统可根据需要配置可容忍的故障数量,而不会显着降低性能.Q/U 是一种 quorum-based 协议,可用于构建故障可扩展的拜占庭式容错服务.相较使用 agreement-based 的副本状态机协议,Q/U 协议可以提供更好的吞吐量和故障可伸缩性.使用 Q/U 协议构建的原型服务在实验中优于使用副本状态机实现的相同服务,使用 Q/U 协议时,性能减少了 36%,而拜占庭式容错的数量从 1 增加到 5,使用副本状态机协议时,性能下降了 83%.

文献[7]提出了一种混合拜占庭式容错状态机副本协议,在没有争用的情况下,HQ 采用轻量级的基于仲裁的协议,节省了副本间二次通信的成本.一旦出现争议,HQ 则使用 BFT 解决争用.此外,总共仅使用 $3f+1$ 个副本来容忍 f 个故障,为节点故障提供了更加优秀的恢复能力.

文献[8]提出了一种高吞吐量的拜占庭式容错架构,它使用特定应用程序的信息来识别和同时执行独立的请求.该体系结构提供一种通用的方法来利用应用程序间的并行性,在提高吞吐量的同时,还不损害系统工作的正确性.

文献[9]提出了一种使用推测来降低成本并简化拜占庭容错状态机副本的协议.在 Zyzzyva 中,副本可直接响应客户端的请求,而不需要首先运行 PBFT 的三阶段共识协议来完成请求的定序处理.副本节点可采用主节点提出的请求定序,并立即回应客户端.副本节点可能会出现不一致,一旦客户端检测到不一致,将帮助副本节点收敛在单一请求定序上.同时,Zyzzyva 将副本节点开销降低到了理论最小值附近.

Aardvark 算法[10]提出了一种实用的 BFT 方法,通常被称为 RBFT(robust BFT)算法,RBFT 算法使得系统在面对最好和最坏的情况下,性能都能大致保持不变,极大地提高了系统的可用性.

(2) 安全机制

区块链系统通过多种密码学原理进行数据加密及隐私保护.目前,区块链上传输和存储的数据都是公开可见的,仅通过“伪匿名”的方式对交易双方进行一定的隐私保护.但对于某些涉及大量商业机密和利益的联盟链业务场景来说,数据的暴露不符合业务规则和监管要求.同态加密、环签名和零知识证明等技术被认为是解决区块链隐私问题的重要技术手段.

- 同态加密技术允许区块链节点在不知道原始数据的情况下完成对交易信息的验证,即验证运算是在加密后的数据之上进行的,得到的结果仍然是加密的,而交易发起方对运算得到的加密结果进行解密,从而得到原始运算结果[11−13];
- 环签名技术允许节点在不知道交易双方是谁的情况下完成对交易的验证和存储[14];
- 零知识证明技术允许证明者向验证者证明,并使其相信自己知道或拥有某一消息,但证明过程不能向验证者泄漏任何关于被证明消息的信息.大量事实证明:如果能够将零知识证明用于区块链,将很好地解决区块链的隐私保护问题[15].

此外,联盟链中常见的非对称加密算法有 RSA、Elgamal、背包算法、Rabin、D-H、ECC(椭圆曲线加密算法)[16,17]和 ECDSA(椭圆曲线数字签名算法)[18].对部分投产使用的区块链产品,要求使用国密级别的安全算法.

2 高性能联盟区块链架构设计与实现

本节将介绍提升我们区块链平台(以下简称 Hyperchain)性能的几种优化策略,包括业务逻辑和共识分离、存储优化、数字签名验证优化这 3 个部分.

2.1 系统架构

一个典型联盟链系统的工作流如图 1 所示.

一笔交易最初从客户端发起,在签名后发送给联盟链的节点.服务器端(Http 服务器)在收到交易请求后,会校验交易签名和交易证书等信息.校验通过后,再将交易推送给共识模块,并通过 P2P 模块进行广播,随后开始节点间的共识.

执行模块和虚拟机模块在收到共识模块的交易信息后,会基于区块号调取执行环境,并执行交易.执行内容包括签名验证和交易执行等过程.验签确认了交易的来源和内容的完整和安全性,该过程在执行模块中进行,而具体的交易执行则会以智能合约等形式在虚拟机中进行.交易执行结束后,变化量会被保存在执行模块缓存中,供后来的交易或账本持久化使用.交易结果的 Hash 会返回给共识模块用于和其他节点的执行结果进行比对.

图 2 是改进后的系统架构,相较图 1 的典型联盟链系统架构,该架构最大的不同在于将业务逻辑执行模块与共识验证模块解耦合.由于本文是以“去中心化主板证券竞价交易系统”为应用场景,上交所现有业务系统的 TPS 可以达到 10 万~30 万.而我们在实际测试中发现,在 Hyperchain 上使用智能合约执行交易时,TPS 仅为几千,这个吞吐量远不能满足主板证券交易的需求.

优化后的联盟链系统架构将交易的执行和共识进行了解耦,使得交易执行任务的转移和并行变得可能.如果将交易执行的任务交由上交所现有的外部业务系统完成,而不是采用智能合约的形式,区块链底层平台只负责交易定序及合法性验证工作,那么系统的吞吐量则可以有很大的提升.故优化后的系统架构更加适合证券交易这类较高频的业务场景.

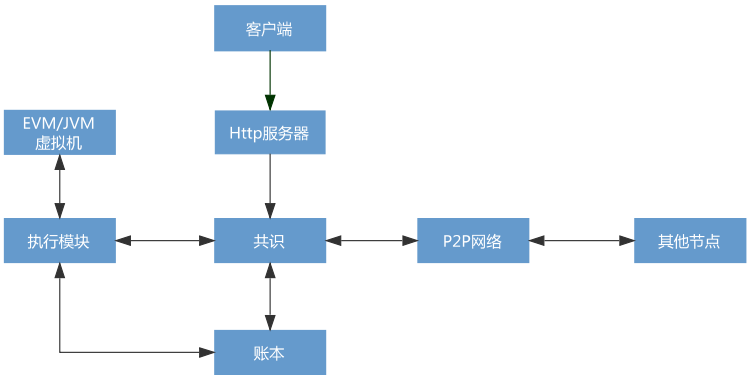

Fig.1 Typical architecture of consortium blockchains

图 1 典型的联盟链系统架构

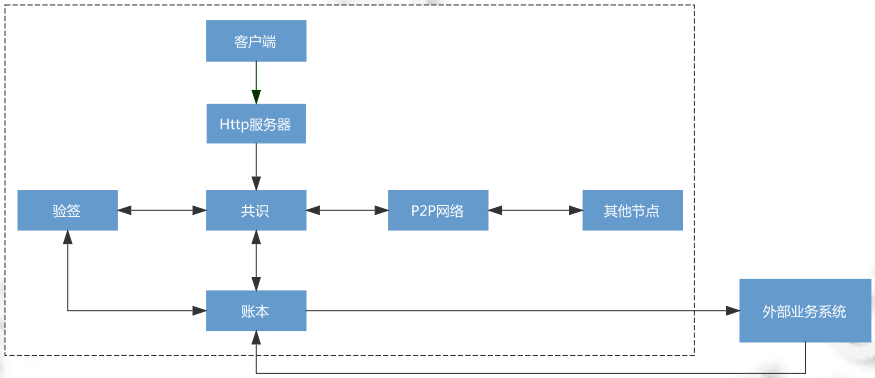

Fig.2 Optimized architecture of consortium blockchains

图 2 优化后的联盟链系统架构

下文中,第 2.2 节针对的是共识模块的优化,第 2.3 节是关于账本模块的存储优化,第 2.4 节提出了两种优化验签模块的方法.

2.2 业务逻辑和共识分离

在联盟链中,使用智能合约执行交易,并将执行结果加入共识流程是一种较为通用的模式(如图 3 所示).将执行结果加入共识可以保证节点间数据的一致性,而使用智能合约可以增强整个联盟链的通用性.针对不同的业务场景,只需要设计不同的智能合约,而不需要更改架构本身.

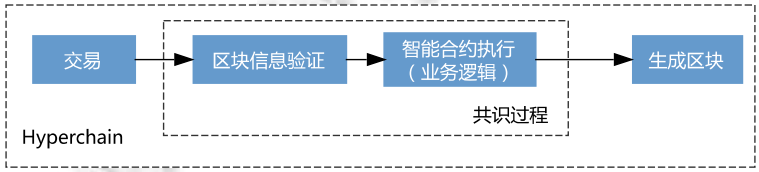

Fig.3 General consensus approach

图 3 常规共识方式

在模拟上交所证券撮合业务场景的过程中,我们发现最为耗时的步骤是交易的共识和执行.表 1 中,交易的共识和执行占一个区块生成时间的 90%.生成一个区块的流程大致可以分为区块打包、区块共识和区块提交这

3 个部分.

Table 1 Duration of each phase

表 1 每阶段占用时长

流程阶段	所占时长(%)
区块打包	4.7
区块共识和执行	90.6
区块提交	3.7

区块打包的耗时与客户端发送交易的内容相关,区块共识和执行、区块提交的耗时则与区块大小和智能合约代码相关.Hyperchain 采用的 PBFT 共识一般可以分为 3 个阶段,分别是 Pre-prepare 阶段、Prepare 阶段和 Commit 阶段.

因为区块执行是按序串行执行的,交易执行大大增加了区块生成所用的时间,是整个系统的性能瓶颈.为了提高交易执行的效率,首先需要解决智能合约运行效率的问题.在实际研究中,我们发现很难用智能合约进行证券交易撮合的业务.一是证券撮合相对一般业务更为复杂,设计一个高效完备的智能合约难度较大;二是智能合约架构设计本身和证券撮合这类业务没有很好地适应.智能合约运行的生命周期是完成一笔交易所用的时间,每处理一笔新的交易,系统都要重新从内存或者数据库中将数据加载进合约,合约处理效率低,无法满足证券交易所需的吞吐量.

为了解决这一问题,我们采用了一种新的架构——业务逻辑和共识分离(如图 4 所示).这是一种在联盟链中越来越流行的架构,如著名开源项目 Fabric 1.0(https://hyperledger-fabric.readthedocs.io/en/release-1.3/arch-deep-dive.html#system-architecture).采用该架构,交易的执行不再受到节点共识的影响,交易可并行执行.

交易到达节点后,共识节点只负责共识交易的排序和交易的内容,不再执行交易和共识执行结果.定序以后的交易通过网络通信发送给外部业务系统,由外部业务系统执行具体的交易,并保存交易结果.不同的外部业务系统可以根据需求,决定是否返回执行结果.由于共识节点不再负责交易的执行和执行结果的共识,Hyperchain 会在区块到达检查点(checkpoint)时再对执行结果进行共识,以保证节点之间数据的一致性.这种架构下,共识节点的功能更倾向于定序和存证.

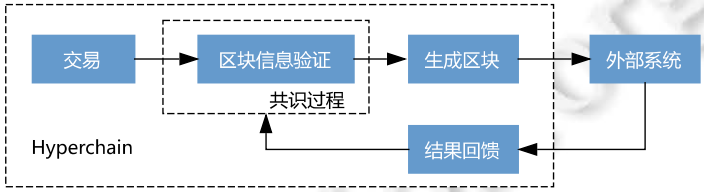

Fig.4 Separation architecture of business logic and consensus

图 4 业务逻辑和共识分离结构

在这种系统架构下,Hyperchain 共识过程不再包括业务逻辑的执行,简化了共识流程,提升了共识速度.此外,将业务逻辑从共识剥离出来后,交易执行模块的可扩展性大大提高.Hyperchain 可以根据业务需求选择使用内嵌的 EVM/JVM 虚拟机,或者使用网络通信对接企业原有的业务系统,不仅降低了开发成本,还能真正实现交易的并行执行.

从图 4 可知:Hyperchain 平台是先进行共识定序和校验,再调用执行模块或外部系统来执行交易.而 Fabric 是先由客户端指定所需的背书节点提前执行交易,再由共识(order)节点进行定序和交易合法性校验.采用先执行后定序的方式必须保证有关联性的交易可以按序执行,否则会导致交易失败.先执行后定序的方式并不适合证券交易场景,因为在该场景下,订单的先后顺序直接影响成交,订单之间的相关性较强.

此外,由于使用网络通信,交易执行模块的插拔可以跨语言和平台,不再限于 Go 语言,各模块可以部署在一台服务器上以降低成本,也可以选择分别部署在不同服务器上,让业务逻辑和共识逻辑可以分布在不同的物理

机上,降低 IO 和 CPU 的抢占.交易执行模块可以针对不同业务场景编写特定交易处理系统,提升效率.以去中心化主板证券竞价交易系统为例,简易 Java 撮合系统每秒可以撮合几十万笔交易,远远超过了目前所有智能合约执行器能达到的效率,使得交易执行的速率可以满足实际生产中的高频交易处理需求.

2.3 存储优化

在 Hyperchain 运行过程中,我们一般会存储 3 部分内容.

(1) 区块信息:包括区块内的交易信息、交易执行的顺序、交易到达区块的时间戳、区块打包时间戳、区块执行时间戳和区块哈希等信息,区块信息包含了区块上链所需的所有内容;

(2) 单笔交易信息和交易回执:Hyperchain 为了提供快速的交易查询功能,交易信息除了储存于区块中,也会独立存储于数据库中(如 levelDB).交易哈希作为存储的 key,交易内容作为存储的 value.以空间换时间,一次数据库读取操作就可以完成交易信息的查询.交易回执在数据库的存储方式和交易信息类似,key 是交易哈希加特定前缀,value 则记录了交易执行的结果.用户通过查询交易回执可以快速获知交易结果;

(3) 账户数据:区块链是一个分布式的大账本,每个节点共同参与大账本的记账.Hyperchain 系统支持使用智能合约执行交易,因此和以太坊一样,采用账户模型来组织数据.每执行一笔交易,都会读/写相关账户的状态数据.比如,一个模拟银行转账的合约会生成或者改变用户余额、转账信息等数据.执行结束,会将期间所有的改动统一写入.

由第 2.2 节可知:针对上交所的证券交易业务场景,Hyperchain 采用了业务执行和共识分离的策略,不再使用智能合约来执行交易,而是直接使用上交所 Java 撮合系统来执行交易,交易流水和账户信息会储存于上交所的数据库中,所以 Hyperchain 不再存储交易流水和账户信息.与此同时,用户的交易流水和账户信息的查询功能也相应地转移给了上交所外部业务系统.从图 5 虚线条可知:当用户需要查询订单信息或账户信息时,我们会默认从上交所数据库中读取信息并返回给用户.如果普通用户指定要从 Hyperchain 查询订单信息时,我们会从内存(内存中只保存近期的订单信息)中读取该笔订单信息并返回给用户,仅当用户需要查询历史订单信息时,我们才会从区块中解析出该笔订单信息并返回给用户.采用该存储方式,是因为我们发现证券业务具有很强的时效性,这样不仅减少了 I/O 消耗,也节省了磁盘空间.

值得一提的是,智能合约为了满足通用性,Hyperchain 统一使用了 NoSQL 类型数据库去存储数据,故在性能上会逊色于上交所定制化的外部业务系统.一般使用智能合约执行交易时,TPS 仅为几千,但上交所现有业务系统的 TPS 可以达到 15 万,深交所新系统 TPS 可以达到 30 万左右,在这种高频业务场景下,I/O 可能会成为 Hyperchain 的性能瓶颈.所以,交易执行和存储任务的转移,缓解了 Hyperchain 的数据库读写压力.

由上可知,Hyperchain 不再单独存储每笔交易的信息和回执以及用户的账户信息,只存储了区块信息.一段时间后,再向 Hyperchain 请求交易信息的,可能是机构为了查账或审计,这时,可从区块中还原出订单信息(见图 5 实线条).尤其在多中心的业务场景下(如深交所和上交所合作的交易场景),由于区块链具有区块数据不可篡改的特性,审计单位可以从节点回溯到原始数据,避免了多中心数据不一致的问题.

一般情况下,由于 LevelDB 有很多和 Hyperchain 相适应的特性,Hyperchain 会选择 LevelDB 作为默认数据库.LevelDB 是 Google 实现的一个高效 KV 数据库,支持多条操作的原子批量操作,满足 Hyperchain 对原子性写入区块信息的需求.借助 LSM(log-structured merge tree)树和 WAL(write-ahead logging)的设计,LevelDB 写性能较为出色,随机写性能达到每秒 40 万条记录,可以支持 Hyperchain 快速存储大量的区块信息.

考虑到机构查账或审计是一种低频率事件,且延时要求低,所以我们对区块信息的存储方式进行了优化.不再采用 levelDB 存储区块信息,而是以一定形式组织区块,然后存储在特定后缀的文件中.每个文件存储固定数目的区块,在文件被关闭时,文件头部预留处将添加区块索引.采用这种形式存取区块的时间复杂为 $O(1)$,不会随着数据量改变而变化.区块按序存储的特性也使得待写入的数据会在缓存中连续存储,磁盘 IO 多为顺序写,以提高 IO 效率.这种存储方式大幅度提高了写性能,更加适合本文的业务场景.

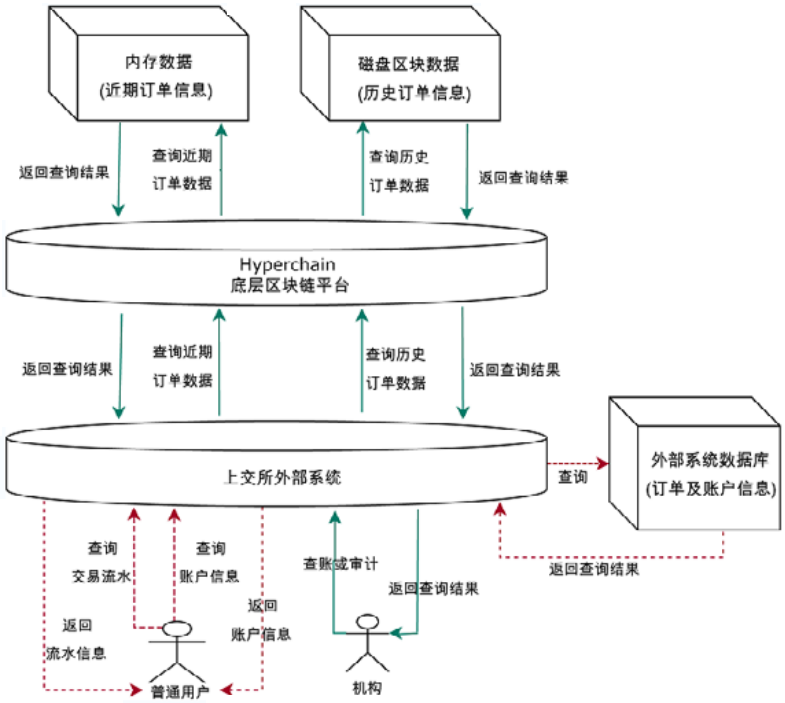

Fig.5 Users or organizations query transaction information

图 5 用户或机构查询交易信息

2.4 数字签名验证优化

区块链所有交易都带有交易签名,区块链节点通过验证签名,确保交易的有效性.在证券交易这种高频业务场景中,交易数目大,要求程序能快速完成验签运算,系统才能具备较低的响应延迟和较高的吞吐量.Hyperchain 对验签策略进行了优化,加快了签名运算.Hyperchain 对数字签名验证优化包括两方面:多订单打包验签和基于 GPU 加速的椭圆曲线验签算法.

2.4.1 多订单打包验签

由于本文是以去中心化主板证券竞价交易系统为应用场景,我们了解到证券交易是一个 B2B 的市场,证券交易存在 OPS(orders per second)的概念,即:订单数(order)跟交易(transaction)并不是一一对应的关系,一个 transaction 里面可以打包若干笔 order.对这些订单只需要在一次签名后打包到单个交易中一起发送.实际业务场景中,券商报单就是十几笔、二十几笔订单一起打包发送.

在研究过程中,我们发现交易的签名验证是整个系统处理流程中最为耗时步骤之一.在分析了上述证券交易的实际情况以后我们发现,合并验签的方式可以提升系统效率.因为单个区块通常包含不止一笔交易,而共识是按块进行的,所以打包验签是最为直接的优化方案.

打包签名和验签可以将共识节点签名和验签的开销平摊在块中的每笔交易上,从而降低总体的开销.尤其是在证券竞价高频交易的情况下,客户端往往会一起发送大量的订单,如果客户端可以将若干笔订单合成在一笔交易中,并进行签名,就会减少大量的验签次数.同时,由于网络带宽是限制联盟链吞吐量的另一个重要因素,合并订单还可以减少节点间的通信量,降低网络带宽的负担.

多订单打包验签即在用户发送交易时候,将多个交易打包成为一个整体,然后通过对应的加密算法(ECDSA 或者国密)来进行统一的签名,随后发送给服务器进行统一验签(如图 6 所示).

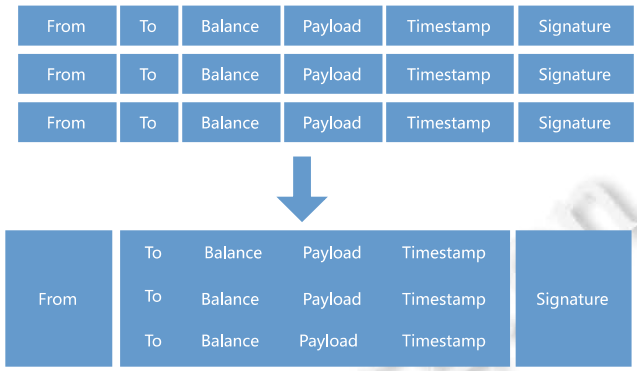

Fig.6 Multi-order packagesignature

图 6 多订单打包签名

多订单打包的内容会被当成一条交易,但在交易验证的过程中,节点会把交易进行反序列化,并按订单打包的顺序,按序执行订单内容.

2.4.2 基于 GPU 加速的椭圆曲线验签算法

Hyperchain 可支持 SM2 椭圆曲线公钥密码算法[19],椭圆曲线密码体制的安全性基于椭圆曲线离散对数问题(ECDLP)的难解性.因此,椭圆曲线密码系统的单位比特强度要远高于传统的离散对数系统.因为区块中所有的交易都带有签名,所以签名速度直接影响了系统响应延时和吞吐量.经过对椭圆曲线验签算法的分析,我们提出了用 GPU 进行硬件加速,实现快速的验签运算的方案.

本文使用 Tesla M40 加速卡来加速 SM2 算法的签名验证.Tesla M40 为 NVIDIA 在 2015 年推出的针对高性能计算领域的通用图形处理(GPGPU),其拥有 3 072 个处理器核心,单精度浮点峰值性能 7Tflops.

任何并行系统为了达到最佳性能,程序的并行度都必须大于等于并行系统的物理并行线程数.对于 NVIDIA Tesla K40,其拥有 3 072 个核心,可以同时并行执行 3 072 个线程,故程序至少应达到 3 072 的并行度.但对于数字签名的验证与签名等密码学操作,其本身运算量庞大并且难以并行.经过数学变换后可以得到签名操作为 6 的并行度、签名验证操作为 12 的并行度.为了完全利用并行系统的性能,需要将多个运算请求打包统一发送同时执行,以平摊并降低单个运算的时间,这个最低的包大小即为

$$BlockSize=\text{物理线程总数}/\text{程序并行度} \tag{1}$$

故对于物理并发度为 3 072 的 Tesla K40,最优区块大小为

$$BlockSize\ \text{签名}=512 \tag{2}$$

$$BlockSize\ \text{签名验证}=256 \tag{3}$$

关于椭圆曲线运算具体理论见文献[20],SM2 算法的具体理论见文献[19].Hyperchain 以相关理论为基础,对现有的有限域运算和椭圆曲线标量乘法运算进行了优化,实现了基于硬件加速的椭圆曲线加密算法:

- 优化 1:提升有限域运算性能

为了提升有限域运算的性能,在工程实现上我们采用了 CUDA 汇编语言 PTX 来实现.在算法层次上,我们则提出了一种全新的快速有限域求模运算.

有限域元素 F_p 在计算机中的表达方式,用整数表达单个域元素需要用 256 位,在目前计算机体系结构下,必须使用多个整数来表达一个域元素.根据目前多数 CUDA 设备的计算能力,32 位整数的每位均摊运算性能远高于 64 位整数[21],故本文采用 8 个 32 位整数来表达一个有限域 F_p 元素:

$$\left.\begin{aligned} &F_p \mapsto \{unsigned\ \ \text{int}32\}^8 : \forall \chi \in F_p, \exists \alpha_0, \alpha_0, ..., \alpha_7 \in \{unsingned\ \ \text{int}32\}: \\ &\chi \equiv \sum_i^8 \alpha_i \cdot 2^{32i} \mod p \end{aligned}\right\} \tag{4}$$

SM2 算法所有的椭圆曲线运算都在 F_p 中进行,每一步基本运算都需要对 p 求模.推荐参数中给出的 p 为:p=

0x*FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF*00000000*FFFFFFFFFFFFFFFF*,p 为 256 位.

直接使用普通的求模运算,开销十分巨大.为了解决这个问题,Hyperchain 提出了一种全新的快速求模算法.快速有限域求模运算的推导过程如下:

对 p 进行变换,得到 $P=2^{256}-2^{224}-2^{96}+2^{64}-1$,并参考公式(4)得到:

$$\chi\cdot 2^{256}=\chi\cdot(2^{224}+(2^{32}-1)\cdot 2^{64}+1) \bmod P \tag{4.1}$$

$$\chi\cdot(2^{32}-1)\cdot 2^{224}=\chi\cdot((2^{32}-1)\cdot 2^{64}+1) \bmod P \tag{4.2}$$

任意 $\omega=\chi\cdot(2^{32}-1)2^{224}+\gamma,\gamma<(2^{32}-1)2^{224}$,可以使用公式(4.2)得到:

$$\omega_1=\omega=\gamma+\chi\cdot((2^{32}-1)\cdot 2^{64}+1) \bmod P \tag{4.3}$$

或者是另一种变体:

$$\omega_1=\omega=\gamma+\chi\cdot(2^{96}-2^{64}+1) \bmod P \tag{4.4}$$

若 ω_1 仍然超出了 $(2^{32}-1)2^{224}$,再次使用公式(4.3)或公式(4.4)即可.

算法 1. 快速有限域求模运算.

输入:标量 ω;

输出:标量 $\omega'<(2^{32}-1)\cdot 2^{224}$,且 $\omega'\equiv\omega \bmod p$.

1. 当 $\omega>(2^{32}-1)\cdot 2^{224}$,执行循环:
2. 求得 γ,x 满足 $\omega=x\cdot(2^{32}-1)\cdot 2^{224}+\gamma$
3. 令 $\omega=\gamma+x\cdot(2^{32}-1)\cdot 2^{64}+x$
4. 输出结果 $\omega'=\omega$

- 优化 2:以并行方式实现椭圆曲线标量乘法运算

椭圆曲线标量乘法很难直接并行,且包含除法等在 256 位有限域内开销极大的运算.Hyperchain 拟采用 Ficher 等人在文献[22]中提出的方法,使用 Montgomery 阶梯算法将一次椭圆曲线数乘运算分解成 256 次加法与加倍运算的迭代,通过将椭圆曲线点坐标投影到仿射坐标系,以变换每次加法与加倍运算,进而消除除法运算,只留下加法与乘法运算,从而可以并行化实现椭圆曲线标量乘法运算.

算法 2. Montgomery 阶梯算法.

输入:标量 K,曲线 $\mathcal{C}$ 上的点 P;

输出:椭圆曲线标量乘法的结果 $K\times P$.

1. 令 $P_1=P,P_2=2P$
2. 令 i 从 $n-2$ 到 0 循环,n 为 K 的位长
3. 如果 K 的第 i 个二进制位为 1
4. 那么令 $P_1=P_1+P_2,P_2=2P_2$
5. 否则,$P_2=P_1+P_2,P_1=2P_1$
6. 输出结果 P_1

Montgomery 阶梯算法拥有一个重要的特殊性质,它保证了任何点加与倍操作时,操作数 P_1,P_2 间的差值始终为 P.而这一重要的性质将允许椭圆曲线点加与倍操作能被并行执行.至此,对椭圆曲线标量乘法的运算已被归约为并行快速地完成椭圆曲线点加与加倍运算.

- 并行椭圆曲线点加与加倍算法

首先对椭圆曲线上的点进行仿射变换,将点 $P=(x,y)$ 变换成

$$\left.\begin{aligned}&(x,y)\rightarrow(X:Z:Y)\\&x=X/Z\ ,y=Y/Z\end{aligned}\right\} \tag{5}$$

参见文献[22],点加公式可以被变形为 $P=(P_x,P_z),Q=(Q_x,Q_z)$ 为两个点,那么:

$$(\tilde{P},\tilde{Q})=((\tilde{P}_x,\tilde{P}_z),(\tilde{Q}_x,\tilde{Q}_z))=(P+Q,2Q)=\begin{cases}\tilde{P}_x=2(P_xQ_z+Q_xP_z)(P_xQ_x+\alpha P_zQ_z)+4bP_z^2Q_z^2-G_x(P_xQ_z-Q_xP_z)\\ \tilde{P}_z=(P_xQ_z-Q_xP_z)^2\\ \tilde{Q}_x=(Q_x^2-\alpha Q_z^2)^2-8bQ_xQ_z^3\\ \tilde{Q}_z=4(Q_xQ_z(Q_x^2+aQ_z^2)+bQ_z^4)\end{cases} \tag{6}$$

由此可见:原椭圆曲线运算的加法运算[20]中不可被并行化并包含除法运算的操作,被变形为了可以被并行化并且只有有限域加法与乘法运算的形式.综上,我们得出了并行的椭圆曲线标量乘法算法.

3 实验结果与分析

本节通过多个对比实验来验证优化策略对系统性能提升的有效性.下文中,第 3.1 节详细介绍了本次实验的具体配置和指标,第 3.2 节~第 3.4 节分别是关于业务逻辑与共识分离、存储优化和数字签名等优化策略对性能影响的对比实验,第 3.5 节~第 3.9 节展示和分析了节点数、打包时间、区块大小、网络延迟和带宽等因素对联盟链性能的影响.

3.1 实验设置

3.1.1 集群环境

(1) 客户端节点配置(见表 2);

(2) Hyperchain 节点配置:本次实验将采用腾讯的云服务器,根据不同场景,实验会采用多种类型的服务器,具体配置如下.

- 配置 1:4 台服务器,Hyperchain 默认记账节点服务器配置.型号:16 核 32G 云服务器(标准型 S2,见表 3);
- 配置 2:大于 4 台服务器,用于多节点测试.型号:16 核 16G 云服务器(标准型 S1,见表 4);
- 配置 3:用于国密 GPU 验签测试,型号:CPU 28 核 56G 计算型(GN2).

Table 2 Test environment of the client side

表 2 客户端测试环境

机器名(IP)	CPU/内存	主频	软件环境
节点 1	4 核/8G	2.4ghz	CentOS 7.1
节点 2	4 核/8G	2.4ghz	CentOS 7.1
节点 3	4 核/8G	2.4ghz	CentOS 7.1
节点 4	4 核/8G	2.4ghz	CentOS 7.1

Table 3 Standard S2 server configuration

表 3 标准型 S2 服务器配置

机器名(IP)	CPU/内存	主频	软件环境
节点 1	16 核/32G	2.4ghz	CentOS 7.1
节点 2	16 核/32G	2.4ghz	CentOS 7.1
节点 3	16 核/32G	2.4ghz	CentOS 7.1
节点 4	16 核/32G	2.4ghz	CentOS 7.1

Table 4 Standard S1 server configuration

表 4 标准型 S1 服务器配置

机器名(IP)	CPU/内存	主频	软件环境
节点 1	16 核/32G	2.4ghz	CentOS 7.1
节点 2	16 核/32G	2.4ghz	CentOS 7.1
节点 3	16 核/32G	2.4ghz	CentOS 7.1
节点 n(大于 4)	16 核/32G	2.4ghz	CentOS 7.1

3.1.2 网络拓扑图

本次实验的服务器网络拓扑图如图 7 所示,左边部署了 Hyperchain 平台节点,右边的客户端节点用于发送交易请求,终端节点用于控制 Hyperchain 节点和客户端节点.

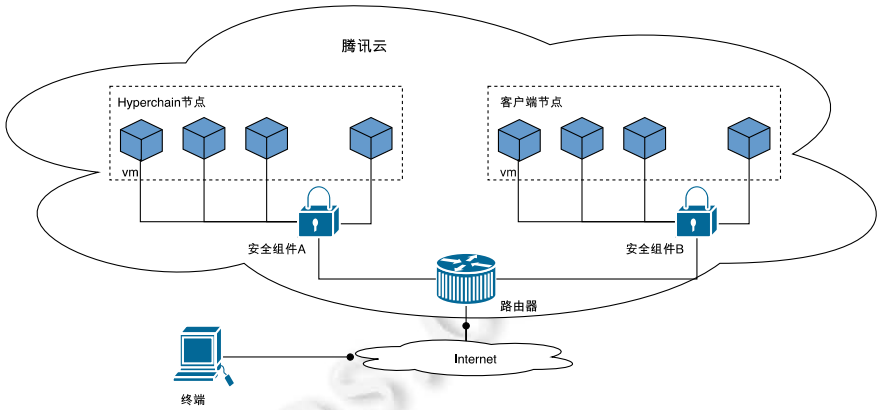

Fig.7 Multi-order packaging verification

图 7 多订单打包验签

注:实验过程中,将使用 Hyperchain agile 工具来模拟客户端发出的交易请求.

3.1.3 参数设置

- 记账节点服务器数量:4 台(标准型 S2);
- 客户端并发度:100×4(4 节点,每个节点并发度为 100);
- 区块大小(batch size):500;
- 打包时间(batch time):500ms;
- 合并验签数:10.

注:区块打包有两种情况,一种是按时间打包(batch time),一种是按交易数目打包(batch size).

3.1.4 性能指标

本实验拟采用的性能指标包括吞吐量(TPS)和延迟(Latency,默认单位为 ms).

(1) 吞吐量计算方式:交易发送完毕后,遍历测试中间三分之一时间区块:

$$TPS=\text{总交易数}/\text{时间} \tag{7}$$

(2) *Latency* 计算方式:遍历测试中间三分之一时间的区块,计算平均时间:

$$Latency=Commit\ \text{时间}-\text{客户端发起请求的时间} \tag{8}$$

注 1:遍历测试中间三分之一时间的区块是为了统计系统处理能力处于相对稳定状态时的实验数据,避免负载不足对实验结果的影响;

注 2:*Commit* 时间为 *Commit* 阶段(即共识流程的 *Commit* 阶段)写数据的时间.

3.2 业务逻辑和共识分离架构对比实验

本节将分别使用传统共识架构与业务逻辑和共识分离架构来执行系统交易.采用传统共识架构时,我们将使用 Hyperchain 实现的 EVM 虚拟机来执行交易.

从表 5 和表 6 可知,采用业务逻辑和共识分离架构的 Hyperchain 平台其系统性能明显提升.系统运行 1 分钟时,TPS 和 Latency 性能效果分别提升了 18.5 倍和 17 倍;系统运行 5 分钟时,TPS 和 Latency 分别提升了 18.7 倍和 17.5 倍.由此可见,采用业务逻辑和共识分离架构,节点只共识区块链上交易的排序和交易的内容,具体交易交由外部业务系统执行,提升了系统性能.

Table 5 Impacts of separation architecture of business logic and consensus on system throughput

表 5 业务逻辑和共识分离架构对系统吞吐量的影响

系统运行时间(min)	分离(TPS)	不分离(TPS)
1	15 621	800
5	15 702	798

Table 6 Impacts of separation architecture of business logic and consensus on system latency

表 6 业务逻辑和共识分离架构对系统延迟的影响

系统运行时间(min)	分离(ms)	不分离(ms)
1	35	627
5	34	629

3.3 存储优化对比实验

Hyperchain 采用区块业务执行和共识分离策略,并使用外部业务系统执行交易,节点不再执行交易,所以数据库只存储区块信息,不再存储交易执行结果的相关数据.本节通过对比存储优化前后系统的 TPS 和 Latency,来验证存储优化对系统性能提升的效果.

从表 7 和表 8 可知,系统运行 1、5、10、15 和 30 分钟时,系统的 TPS 和 Latency 较为稳定,并无明显波动.采用存储优化策略后,TPS 和 Latency 的性能效果平均提升了 27.3%和 28.7%,系统性能得到了较为明显的提升.

Table 7 Impacts of storage optimization on system throughput

表 7 存储优化对系统吞吐量的影响

系统运行时间(min)	优化前(TPS)	优化后(TPS)
1	15 621	19 921
5	15 702	19 884
10	15 699	19 896
15	15 724	19 992
30	15 713	20 171

Table 8 Impacts of storage optimization on system latency

表 8 存储优化对系统延迟的影响

系统运行时间(min)	优化前(ms)	优化后(ms)
1	35	24
5	34	25
10	34	25
15	34	24
30	34	24

3.4 数字签名验证优化对比实验

3.4.1 多订单打包验签对系统性能的影响

本节我们将改变每个交易包含的订单(order)数量,来验证统一验签功能对系统性能提升的有效性.

每笔交易只包含一个订单(1 order)时,系统的瓶颈在于验签.由表 9 可知:每笔交易分别包含 10 order、20 order、30 order 进行统一验签时,系统交易的吞吐量逐渐下降.一方面是因为网络带宽的压力增加,转发时间变长;另一方面是因为节点处理 order 数增加,反序列化存在内存中更加缓慢.但系统订单的吞吐量在增加,因为验签是系统的瓶颈,打包验签将低了验签的次数.

Table 9 Impacts of multi-order packaging verification on system throughput and latency

表 9 不同订单数量的打包验签对性能的影响

交易包含 order 的数量	交易(TPS)	订单(OPS)	延迟(ms)
1 order	22 630	22 630	22.14
10 order	21 600	216 000	23.08
20 order	20 271	405 420	24.66
30 order	18 466	553 980	27.1

3.4.2 基于 GPU 的国密签名对性能的影响

本节对 Hyperchain 平台实现的 CUDA 并行版本的 SM2 算法与 GmSSL 库进行对比.GmSSL 是一套基于 OpenSSL 使用纯 CPU 实现的 SM 系列算法库,是目前工业界常用的 SM2 算法实现.

测试环境:服务器类型为配置 3(GN2),在本次实验中,平台的记账节点是 4.

由表 10 可知,GPU 验签的吞吐量是 CPU 验签的 4.5 倍左右.由表 11 可知,GPU 验签的延迟时间只有 CPU 验签的一半.

Table 10 Impacts of GPU Optimization on system throughput

表 10 GPU 优化对系统吞吐量的影响

打包验签数	GPU 验签 TPS	CPU 验签 TPS
1	46 421	9 749
10	36 855	8 654

Table 11 Impacts of GPU optimization on system latency

表 11 GPU 优化对系统延迟的影响

打包验签数	GPU 验签延迟(ms)	CPU 验签延迟(ms)
1	21.3	57.3
10	22.4	58.9

3.5 节点数对系统性能的影响

当节点数量增加,联盟链方案将会导致网络中交易结算时间的延长.本节将测试不同节点数下的系统性能.

在服务器配置 2 的情况下,由图 8 和图 9 可知:随着节点数的增加,系统性能略有提升.这是因为节点数在一定范围内增加时(系统带宽还未成为瓶颈的范围内),发起共识之前的验签压力可以分摊给各个节点,每个节点可以拥有更多的资源用于共识.在 8 节点后,系统性能略有下降.这是由于共识节点之间需要相互通信(发送 request,pre-prepare 等消息),节点数过多后,系统带宽会逐渐成为性能瓶颈.

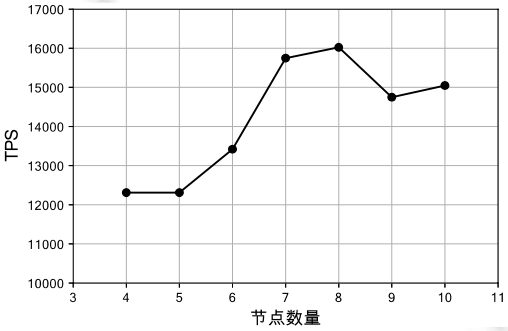

Fig.8 Number of nodes impact on system throughput

图 8 节点数量对系统吞吐量的影响

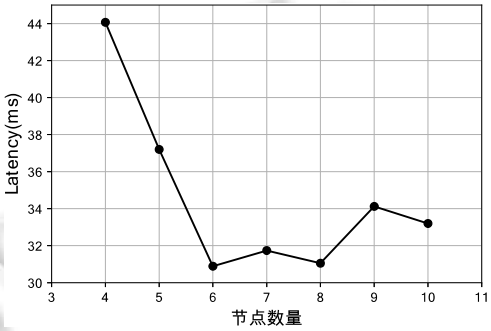

Fig.9 Number of nodes impact on system latency

图 9 节点数量对系统延迟的影响

由于课题经费有限,我们没有测试更多节点的场景.总的来说,从 4 个节点~10 个节点,系统吞吐量变化不大.这是由于此时系统带宽还没碰到瓶颈.

3.6 Batch time对系统性能的影响

在服务器配置 1 的条件下,由表 12 可知,区块打包时间对系统性能没有显著影响.

Table 12 Impacts of batch time on system throughput and latency

表 12 区块打包时间对系统性能的影响

Batch time (ms)	吞吐量(TPS)	系统延迟(ms)
50	21 108	23.58
100	20 898	23.98
200	20 573	24.20
500	21 600	23.08
1 000	20 282	24.6

3.7 Batch size对系统性能的影响

在服务器配置 1(4 节点)的条件下,由图 10 可知,Batch size 小于 200 后 TPS 较显著下降.由图 11 可知,Batch size 越大延迟越高.这是因为区块需要在共识节点间进行流转,受网络带宽的限制,造成 Batch size 越大延迟也越高.选用 200~500 笔交易作为区块大小为佳.

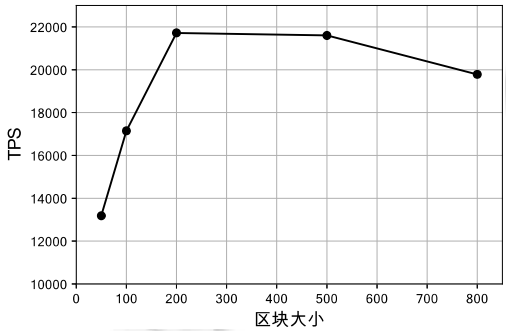

Fig.10 Size of each blockimpacts on system throughput

图 10 区块大小对系统吞吐量的影响

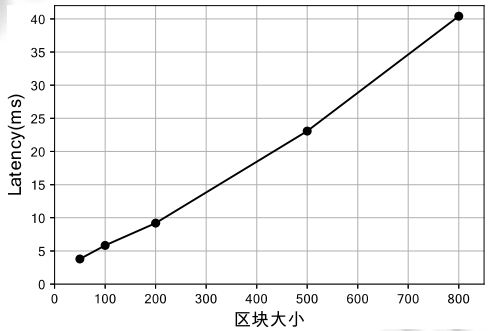

Fig.11 Size of each block impacts on system latency

图 11 区块大小对系统延迟的影响

3.8 网络延迟对系统性能的影响

验证节点之间的距离也会对性能产生影响,物理距离太近使得吞吐量会变得很高,所以我们人为地用 LINUX 的工具软件增加了网卡延迟.

由表 13 可知,随着网络延时的提升,系统 TPS 略有下降,Latency 急剧提升.在网络延时大于等于 500ms 以后,节点间的共识会超时,使交易无法正常执行.

Table 13 Network delays impacts on system throughput and latency

表 13 网络延迟对系统性能的影响

网络延迟(ms)	吞吐量(TPS)	系统延迟(ms)
0	8 949	57.3
50	8 524	259.8
100	8 399	463.7
200	7 487	933.7
500	–	–

3.9 带宽对系统性能的影响

在服务器配置 1 的条件下,由图 12 可知,带宽对 TPS 有显著影响.在 4 节点情况下,10MB 带宽仅有 642TPS;500MB 带宽时,带宽瓶颈不再显著.由图 13 可知:随着带宽的增加,系统延迟显著减小.目前,网络带宽是限制联盟链吞吐量的最主要因素之一.

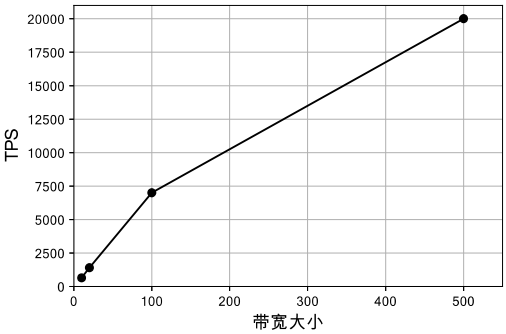

Fig.12 Bandwidth impacts onsystem throughput

图 12 带宽大小对系统吞吐量的影响

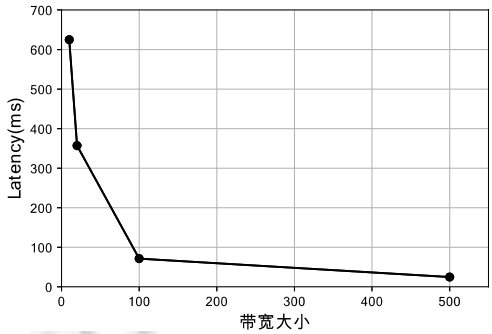

Fig.13 Bandwidthimpacts on system latency

图 13 带宽大小对系统延迟的影响

4 总结与展望

我们已经研究了联盟区块链技术应用于证券撮合系统的技术难点,并提出了高性能撮合系统解决方案.课题组对接了 Java 撮合系统并进行了性能测试、可靠性测试和部分调优等工作.针对证券交易的订单,整个系统的处理速度可以达到 20 万/s.在不同节点数量、区块大小和网络延迟的条件下,也能保证正确性和较高的性能.

未来我们将继续探索 FPGA、微服务架构等技术在区块链领域的相关应用.微服务架构是一个分布式的技术架构,可对应用进行模块拆分,具备敏捷开发、快速演化和弹性伸缩等特性.使用微服务架构可进一步提升联盟链系统性能.目前,我们已经将共识模块和业务逻辑执行模块进行分离,执行过程比较耗时;而共识模块比较稳定,拆分之后的共识模块可不受执行效率的影响.在执行模块中,虚拟机和存储模块也可做进一步拆分.未来还可以使用 FPGA 对更多加密算法与其他功能函数进行更高性能的优化.由于密码学模型与电子电路模型的天然相似性以及 FPGA 模型的流水线并行模式,密码学算法在 FPGA 上的实现能够轻松地发挥 FPGA 的全部性能,同时保持足够的并行度.而且 FPGA 本身的性能上限并不输于 CPU 与 GPU,甚至在整数计算上拥有更强的能力.因此,使用 FPGA 实现密码学算法能够提升更快的性能.

References:

[1] Wyman O. Blockchain in Capital Markets: The Prize and the Journey. Euro Clear, 2016.

[2] STELLA—Joint research project of the European Central Bank and the Bank of Japan. In: Proc. of the Payment Systems: Liquidity Saving Mechanisms in a Distributed Ledger Environment. 2017.

[3] Santo A, Minowa I, Hosaka G, Hayakawa S, Kondo M, Ichiki S, Kaneko Y. Applicability of Distributed Ledger Technology to Capital Market Infrastructure. JPX Working Paper, Japan Exchange Group, 2016.

[4] Lamport L, Shostak RE, Pease MC. The Byzantine generals problem. ACM Trans. on Programming Languages and Systems (TOPLAS), 1982,4(3):382−401.

[5] Castro M, Liskov B. Practical Byzantine fault tolerance. In: Proc. of the OSDI, Vol.99. 1999. 173−186.

[6] Abd-El-MalekM, Ganger GR, Goodson GR, Reiter MK, Wylie JJ. Fault-scalable Byzantine fault-tolerant services. ACM SIGOPS Operating Systems Review, 2005,39(5):59−74.

[7] Cowling J, Myers D, Liskov B, Rodrigues R, Shrira L. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In: Proc. of the 7th Symp. on Operating Systems Design and Implementation. USENIX Association, 2006. 177−190.

[8] Kotla R, Dahlin M. High throughput Byzantine fault tolerance. In: Proc. of the 2004 Int'l Conf. on Dependable Systems and Networks. IEEE Computer Society, 2004. 575.

[9] Kotla R, Alvisi L, Dahlin M, Clement A, Wong E. Zyzzyva: Speculative byzantine fault tolerance. ACM SIGOPS Operating Systems Review, 2007,41(6):45−58.

[10] Clement A, Wong EL, Alvisi L, Dahlin M, Marchetti M. Making Byzantine fault tolerant systems tolerate Byzantine faults. In: Proc. of the NSDI, Vol.9. 2009. 153−168.

[11] Gentry C, Boneh D. A Fully Homomorphic Encryption Scheme. Stanford: Stanford University, 2009.
[12] Gentry C. Computing arbitrary functions of encrypted data. Communications of the ACM, 2010,53(3):97–105.
[13] Van Dijk M, Gentry C, Halevi S, Vaikuntanathan V. Fully homomorphic encryption over the integers. In: Proc. of the Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer-Verlag, 2010. 24–43.
[14] Noether S, Mackenzie A. A note on chain reactions in traceability in cryptonote 2.0. Research Bulletin MRL-0001. Monero Research Lab., 2014. 1–8.
[15] Bitansky N, Canetti R, Chiesa A, Tromer E. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Proc. of the 3rd Innovations in Theoretical Computer Science Conf. ACM Press, 2012. 326–349.
[16] Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation, 1987,48(177):203–209.
[17] Miller VS. Use of elliptic curves in cryptography. In: Proc. of the Conf. on the Theory and Application of Cryptographic Techniques. Berlin, Heidelberg: Springer-Verlag, 1985. 417–426.
[18] Zhou P, Du Y, Li B. White Paper on China Blockchain Technology and Application Development (2016). Beijing: Ministry of Industry and Information Technology, 2016 (in Chinese).
[19] State Cryptography Administration. Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves. Beijing: State Cryptography Administration, 2010 (in Chinese).
[20] Silverman JH. The Arithmetic of Elliptic Curves. Springer Science & Business Media, 2009.
[21] Nvidia. CUDA C Programming Guide. Nvidia Corporation, 2015: Section 5.4.1.
[22] Fischer W, Giraud C, Knudsen EW, Seifert JP. Parallel scalar multiplication on general elliptic curves over Fp hedged against non-differential side-channel attacks. In: Proc. of the IACR Cryptology ePrint Archive 2002. 2002.

附中文参考文献:

[18] 周平,杜宇,李斌.中国区块链技术和应用发展白皮书(2016).北京:工业和信息化部,2016.
[19] 国家密码管理局.SM2 椭圆曲线公钥密码算法.北京:国家密码管理局,2010.

朱立(1972－),男,浙江海盐人,工程师,CCF 专业会员,主要研究领域为区块链,分布式计算,高可用架构.

邱炜伟(1986－),女,博士,助理研究员,CCF 专业会员,主要研究领域为区块链,分布式计算.

俞欢(1990－),女,硕士,主要研究领域为区块链,分布式计算.

李启雷(1982－),男,博士,讲师,CCF 专业会员,主要研究领域为区块链,分布式计算,人机交互技术.

詹士潇(1993－),男,硕士,主要研究领域为区块链,分布式计算.