提 供 无 限 存 储 能 力

去 中 心 化 结 构 云 存 储

Lambda

lambda.im

Lambda

摘要

Lambda是一个高速、安全、可扩展的区块链基础设施项目，通过对Lambda Chain和Lambda DB的逻辑解耦和分别实现，我们向去中心化应用提供可无限扩展的数据存储能力，并实现了多链数据协同存储、跨链数据管理、数据隐私保护、数据持有性证明、分布式智能计算等服务。另外，作为区块链的基础服务，Lambda Chain通过Sharding技术，提供了每秒钟数百万笔的请求能力(RPS)，并且可以随着系统规模的扩大而增长；通过子链技术，提供了面向未来无限扩充的技术服务能力。

概述

在过去的几年里，区块链技术蓬勃发展，借助于区块链账本技术及POW工作证明，人类第一次实现了去中心化的大规模共识，随后比特币和其他数字加密货币的出现，使得点对点的价值交换问题得到了完美的解决。以太坊出现以后，通过图灵完备的智能合约技术，为去中心化的应用开发提供了方便易用的平台，随之而来的就是基于Erc20的多种token发行。token交换降低了协作和价值交换的成本，降低了项目中网络效应形成的成本，加快了项目发展的速度。

区块链及相关的社区在加速发展和繁荣，但是，到目前为止，很多去中心化应用项目面临着无法落地的风险，目前真正落地的区块链项目只有两个，第一个是比特币，第二个是基于以太坊发行TOKEN以及ICO。其他领域区块链项目，目前很多都无法落地，根本原因在于基础设施的不完善，这体现在多个方面:

第一是区块链系统整体吞吐量的问题，也就是常说TPS问题。比特币网络每秒仅能处理7笔交易，以太坊的处理速度也并不理想，约在每秒15笔左右，而中心化金融类系统的TPS均在数万到数百万TPS规模。 EOS、Tron这样的项目致力于使用DPos共识机制来解决TPS的问题，但目前尚在进展之中，从目前测试数据来看，超级节点机制的实现方案和预期存在比较大的差距，EOS在提出设想时的目标是百万TPS，从当前测试系统来看，真实TPS应该在每秒数千的规模，另外的问题是，通常分布式系统都具备水平扩展能力，而超级节点机制则无法让吞吐量随系统规模扩展。Algorand、Cardano、Dfinity采用VRF共识算法来解决交易速度问题，从目前看，VRF算法都是依赖于强同步网络的假设，对网络延迟导致的分叉并没有很好的解决。

其二是数据存储的问题。互联网的本质是人和信息的连接，无论是中心化应用还是去中心化应用均是如此，离开数据，互联网无法工作，整个互联网本身就是依靠数据进行相互操作的过程。然而，今天的去中心化应用缺乏数据落地的能力，很多缺乏技术认知的去中心化应用开发团队都误以为区块链是一个去中心化的分布式数据库(database)，甚至是一种云端的“魔法数据库”，而事实上现在所有的公链的本质只是一个去中心化的分布式账本(ledger)，是一个简单的交易记录的线性表，能够存储的数据只有交易双方的地址以及交易金额，其他数据均无法存储。在没有数据的情况下，区块链的实际使用场景受到了很大的限制。Sia、Storj、MaidSafe等项目提供的电子化网盘并非是针对程序访问，价值相对较小。IPFS和FileCoin项目致力于提供一种分布式的文件存储，这是有意义的尝试和探索。但正如在互联网领域曾经发生过的一样，文件是一种非格式化数据，只能用于人的整体观看浏览，却无法让应用程序方便的查询和访问，大规模的数据交换和应用需要基于格式化/半格式化的对象存储系统或数据库系统，来提供对象数据、KV数据、表格数据和关系型数据的访问能力。并且，IPFS的设计时间较早，其设计思想更多受到P2P网络项目如Kademlia的影响，这也导致其主链项目FileCoin的进展不如预期。

第三是数据孤岛和资产孤岛的问题。在计算机技术的早期发展阶段，数据和应用都是限制在企业内部，甚至是企业的部门之内，互相之间缺乏数据和应用连接的能力。2008年前后，通过面向服务的架构SOA和企业级的服务总线技术，使得数据之间可以互相连接，提供了数据的互操作性。当前阶段，各个不同的区块链系统之间缺乏互相连接和验证的能力。围绕比特币的侧链项目如RSK承诺提供比特币领域的资产转移能力，Polkadot和Cosmos也正在这个领域提供更多的尝试，到目前为止，这些项目依然在进展之中。

第四是可开发性和易用性的问题。比特币系统使用功能有限的脚本系统来提供了有限的可编程支撑，以太坊以后，EVM成为事实上的标准，提供了一种图灵完备的编程方案来编写智能合约,通过智能合约来操作链上的数据。然而，Sodility语言，以及其他新的区块链语言本身，对于很多人来说存在学习曲线和学习成本。并且，很多已经成熟的开发方法，比如SSDLC、单元测试等技术，均无法直接移植到区块链领域。这就使得区块链的可开发性存在一定问题，并且，由于区块链是直接对资产进行编

程，程序的漏洞会直接导致资产的损失，如BEC、SMT等项目均暴露出了合约漏洞的问题。目前在虚拟机和形式化验证等领域，并没有突破性的进展。

基于以上四个问题的前三个问题，Lambda项目试图提供一种新的解决方案，基于将不同类型的数据在不同的链和块上分开存储的设计，Lambda致力于为去中心化应用提供一个基础设施，在此基础设施平台之上，提供可无限扩展的存储、高吞吐量计算、快速网络传输在内的一系列基础能力服务，使分布式应用可以轻松完成数据的生成、传输、存储、检索和计算。不仅如此，Lambda 项目还具有数据结构灵活、编程接口强大、系统可扩展、原子化操作等特点。另外，在数据索引技术、多链数据协同处理、跨链数据管理、隐私保护、存储持有性证明、分布式智能合约（存储过程计算）、查询优化技术等领域，Lambda项目也将全力进行探索。

开发人员可以自由使用Lambda项目的代码，并在Apache协议下进行修改。用户可以自由下载Lambda的二进制文件并在自己的私有环境下进行部署，并且不与外网进行网络连接，在这种情况下，Lambda项目等同于带有自动审计功能和审计数据不可篡改特性的分布式数据库，在银行间清算、穿透式审计和众多金融场景下发挥作用。人们也可以选择加入Lambda云端网络，成为整个Lambda云的一部分，并获得我们的原生Cryptocurrency：LAMB。随着Lambda项目的发展，会有越来越多的节点加入到网络中来，最后，会形成一个云端的涵盖数十万节点的自组织自管理的多个数据存储集群，所有的去中心化应用均可以通过API来方便的使用这个集群系统来存储和查询数据。特别的是，由于无需负担昂贵的中心化IAAS庞大的组织成本，这个去中心化的数据集群系统一定具备极高的性价比优势，以及天生具备的异地灾备、跨洲际数据共享等能力。这已经初步形成一个去中心化的IAAS，具备存储、计算、网络带宽三种基本的IAAS能力，再加上区块链网络提供的价值交换体系，Lambda将是一个具有无限想象空间的全球基础设施网络。

Lambda项目从初始就针对物联网和人工智能场景进行了优化设计，物联网领域的物联网数据是海量的、严格按时间递增的、结构非常简单的各种指标，而人工智能领域则要对数据的提供和标注提供大量的激励。未来，这些海量的数据一定是只能在去中心化的基础设施上进行计算和处理。Lambda项目通过Agent和CEP技术支持物联网数据上链；在人工智能领域，Lambda项目通过货币激励使得人工智能领域数据的创造被价值化、共识化，形成一个大的数据交易市场，从而推动物联网行业和人工智能行业的进步。

数据开放共享是推动数据产业发展的源动力，数据是未来世界的石油和源动力，然而现在的数据市场，数据提供方将数据存在在单方私有的数据存储中心，其他需求方下载数据以实现分析。这种模型已经被证实存在较大的弊端，数据交易缺乏透明性，数据所有者的数据所有权和控制权依赖于单方存储节点的职业道德，无法保证数据的安全。我们的最终理念是，通过可被公证和公信的数据加密存储基础设施，通过不同安全级别不同带宽的存储节点，分级存储可公开验证和可匿名验证的不同类型不同等级的数据，从而推动人类社会公共数据的共享和共有，最终做到共同获益。未来，医疗、教育、交通、能源、农业、天气、水文、基建、科研、公共安全数据等等，这些所有人类共有的公共数据，都可以在Lambda平台进行分布式可验证的存储、分析、交易和使用。

Lambda

章节内容

一、Lambda项目介绍(链的设计)

1/项目概述

Lambda项目致力于为区块链和去中心化应用提供一个数据存储基础设施，在此基础设施平台之上，提供去中心化的云数据库存储和访问能力，并且，Lambda基于水平可扩展性和分片技术，提供了高速的交易能力；通过中继链技术和跨链交易验证，提供了系统内的跨链交易和数据访问及验证能力，通过对BCP协议的支持，提供和其他链系统的跨链通信能力。通过提供可无限扩展的块存储、文件存储、对象存储、KV存储和表格存储，以及快速网络传输(rsync)在内的一系列基础能力服务，Lambda使分布式应用可以轻松完成数据的生成、计算、传输、存储、检索。通过属性加密和代理加密技术，我们对数据的隐私性提供了保护。不仅如此，Lambda 项目还具有数据结构灵活、编程接口强大、高效备份等特点。

随着Lambda项目的发展，会有越来越多的服务能力，如分布式缓存、基于非易失性内存的分布式共享内存计算、分布式关系型数据库、分布式MapReduce等项目，作为平行子链加入到网络中来，通过Lambda一起对外提供基础设施服务。最后，会形成一个云端的涵盖数千万节点的自组织自管理的数据管理系统，所有的去中心化应用均可以通过API来方便的使用这个云端数据库来存储和查询数据。特别的是，由于无需负担昂贵的中心化IAAS庞大的组织成本，这个去中心化的云数据库一定具备极高的性价比优势，以及天生具备的异地灾备、跨洲际数据共享等能力。这已经初步形成一个去中心化的IAAS，具备存储、计算、网络带宽三种基本的IAAS能力，再加上区块链网络提供的价值交换体系，Lambda将是一个具有无限想象空间的全球基础设施网络。

Lambda作为一个去中心化的区块链数据基础设施，并不会对标中心化数据库的性能指标，并且，考虑到对等节点分布式网络的特性，Lambda云数据库在当前时间肯定存在使用场景的限制。Lambda的优势在于开源、社区治理、经济模型和信任机制可验证，对于今天的技术人员来说，Lambda可以作为一个云端的KV数据库(如redis)、文档型数据库(mongodb)和时间序列数据库(druid)来使用，可以先从备份和归档场景开始使用，来存储不可变数据。从业务场景来说，Lambda更加适合于交易速度频繁，数据流入流出数量较高但改变量较小的场景。Lambda可以用来存储物联网和AI数据，此外，也可以用来存储KV数据、日志数据、Mertrics和Event数据、物联网和AI数据、feed流数据等，因此，类似于去中心化的彩票、低频博彩、视频网站、Feed流应用、博客、论坛等去中心化业务均可以使用Lambda云数据库作为数据的后端存储。

2/Why We Need Lambda

互联网应用开发需要数据库系统

今天，互联网应用已经深入到每一个人的生活，普通用户很少感知到互联网背后的计算机科学。然而，互联网从一开始就依赖于计算机技术的发展，特别是其中几项关键技术，比如HTTP协议、Web服务器和数据库软件。

1970年，IBM的研究院埃德加.弗兰克.科德发表了题为“大型共享数据库的关系模型”论文，开创了数据库软件的历史，被称为关系型数据库之父。1977年，拉里埃里森创建了Oracle公司，并研发出商用数据库软件Oracle。1989年蒂姆·伯纳斯·李在欧洲原子核研究会（CERN）建立的粒子实验室开发出了HTTP协议、Web服务器和浏览器，在这几项技术的基础上，互联网的帷幕从此拉开。随着时间的推移，互联网的使用人群一直在扩大，直到世界每一个角落，技术也一直在进步，然而，无论技术如何变化，互联网应用的开发始终需要数据库系统来存储数据。

区块链上的去中心化应用需要去中心化的数据存储方案

今天，随着区块链技术的发展，去中心化的理念深入人心。然而，去中心化的应用程序在进行软件开发和运行的时候，只能将数据存储于中心化的IDC之上，本质上，这依然是一个中心化的系统。传统的数据库管理系统往往由单一机构进行管理和维护，该机构对整个数据库具有最高权限。这种模式并不适用于在并非完全互信的机构之间互相管理数据，在互联网应用环境中该问题尤为突出。区块链作为一种去中心化、不可篡改、可追溯、多方共同维护的新型分布式数据管理技术，适合于在这种非互信场景下进行有效的数据管理。由于新旧数据管理架构的差异性非常显著，无法照搬现有的数据库管理技术，必须在区块链数据管理技术上有所创新。

Lambda项目本质是一个去中心化的数据库系统，通过区块链技术完成不可信机构之间的授权、加密与通信，向去中心化应用提供数据存储和管理服务。我们认为，只有去中心化的数据库，才能满足去中心化应用的数据存储需求。在Lambda项目之前，开发去中心化的DAPP和应用链是困难的事情，是因为当前的公链无法提供大量的数据存储和检索服务，这就需要DAPP自己去处理数据，这和上个世界九十年代开发程序自己去操作文件没有区别。

当前的区块链存储方案不是DAPP的良好选择

目前去中心化的区块链应用程序在选择针对的存储数据的时候，可用的去中心化的选择是：

将所有内容存储在区块链本身中

今天，在以太坊上面运行了大约有几百个去中心化应用，导致今天的以太坊账本大小已经到了100多GB，这需要每一个节点上面都有如此大的一个存储空间。随着时间的推移，可能成千上万应用会创建，那么，每台机器需要的账本会越来越大。未来，随着磁盘存储空间不足的用户的退出，以太坊会逐渐变得中心化。而且，以太坊的交易速度只有15笔每秒，在交易场景下，等待金钱一分钟是可以接受的，但是等待网页一分钟打开是无法忍受的。

点对点文件系统，如IPFS

点对点文件系统，如InterPlanetary文件系统。 IPFS允许在客户端计算机上共享文件，并将其集成到全局文件系统中。IPFS技术基于BitTorrent协议。分享文件的时候，首先要把文件放在自己的电脑上，有人需要时就可以下载 。文件的哈希中包含下载者或发布者的IP。由于BitTorrent协议，流行的文件下载的人越多，提供的带宽也越多，下载速度就越快。但是，IPFS也有一些缺点，第一，如果你想分享你的文件必须保持在线状态，至少要等到第一个感兴趣的人下载成功。关键是，IPFS只提供静态文件，上传后不能修改或删除。并且不支持通过实际包含的内容来搜索这些文件。通过IPFS实现的一个去中心化的社交应用AKASHA，在发送消息之后，需要等待对方接收才能下载

分布式云文件存储，如Storj，Sia，Ethereum Swarm等

分布式云文件存储：分布式的云文件存储可以解除IPFS的一些限制。从用户的角度来看，这些存储比较像Dropbox这类的云存储，不同之处在于内容存储在出租自己硬盘空间的个人计算机上，而不是在数据中心里。现在有很多这样的项目例如，Sia，Storj，Ethereum Swarm。用户不需要保持在线去分享文件，只需上传文件到云。这些存储是高可靠的，下载速度快，有丰富的存储空间。但仍然只提供静态文件，无法支持内容搜索，也无法通过程序直接访问。

去中心化应用的快速发展需要高速度高可用性的数据基础设施

在过去数十年，信息化和数字化浪潮席卷全球，移动互联网让几乎所有的业务都变成了在线业务，移动互联网也去掉了业务的地域性，很多系统的使用者用户数从数十万用户变成全球的数千万用户甚至数亿用户，淘宝、Amazon类似的电商平台在高峰期为数千万客户提供服务，这种应用要使用位于全球许多数据中心的数以万计的服务器，这就要求后端服务的高可用性和低延迟。在中心化的系统架构中，这种高可用性是用昂贵的硬件堆积出来的，比如Oracle的ExaData系统。今天，区块链蓬勃发展，去中心化的互联网概念深入人心，随着去中心化应用的持续发展，对于后台数据库系统的可用性要求必然越来越高。这就要求存在于一个云端的支持数据持续增长，平台要高度可扩展的数据库系统。对于未来的去中心化应用，高可用性是最重要的要求之一，因为即使最轻微的宕机也会产生重大的经济损失，并影响客户的信任。但是去中心化的网络中只有廉价的PC和移动算力和物联网设备，我们需要在这些不可靠的设备之上构筑一个100%高可用性的数据基础设施系统。

Lambda项目是一个完全没有中心节点的分布式数据存储系统，计算能力依赖于矿工所共享的计算、存储和带宽能力。Lambda数据库是对Amazon DynamoDB论文的一个开源实现，并加入了类Druid和Clickhouse的时间序列数据处理能力，是唯一一个和区块链技术深度融合的开源实现。Lambda云数据库提供简单易用的数据写入和读取服务，通过DynamoDB论文的论证过程，我们可以保证整个系统是一个AP倾向的系统，也就是倾向于保证分区容错性和高可用性。在Amazon内部，DynamoDB的可用性保证了整个Amazon的电商系统正常运行，我们希望通过区块链方式的开源实现，对开发者群体、开源社区和去中心化APP开发者提供一个超越中心化IAAS体验的完全去中心化分布式数据库服务。

万物互联的IOT需要能存储海量数据的低成本数据库

过去的三十年里，随着互联网的发展，我们从物质时代、模拟信号时代进入了进入了数字时代，信息化、数字化成为席卷一切的浪潮，改变着人类社会现存的每一个行业的生产力，伴随着数字化的进程，数据，作为一个必须存在的副产品，成为越来越庞大的存在。在2010年，我们创造了1ZB的数据，2016年我们创造了16ZB的数据，根据预测，在2025年我们，我们生产的数据将达到160ZB。数据将在未来的经济活动上发挥巨大的作用，并将影响到每一个企业、政府机构和消费者个人。大型组织越来越认识到，数据本身就是一项具有战略价值的资产。数据，正在逐渐独立于物理的世界而形成一个数字的世界。

然而，今天数据的价值并没有被完全发挥出来，根据互联网女皇Mary Meeker的报告，目前我们产生的数据中，能够被存储的数据7%，能够被分析的数据只有1%，其他的数据则处于被丢弃的处境。主要原因是，中心化的数据存储和分析算力的增长速度远远落后于数据本身的增长速度，并且中心化的算力价格过于高昂。在今天，机器已经取代人类成为数据的主要生产来源，要想充分利用人类已经产生的数据的能力，我们必须借助于分布式去中心化的雾计算的架构，将更多的分散的存储和算力资源连接起来，以处理这些日益增长的数据，而当前区块链领域去中心化的架构中，只有几种点对点的文件存储系统，没有针对数据的存储和管理系统，后者就是Lambda项目的第一个意义。

Lambda项目可以Cryptocurrency的激励，对提供计算和存储能力的矿工给予奖励，并且连接众多的弱算力设备，构成一个低成本的分布式数据存储和计算网络。通过这种架构，Lambda平台可以将原来单位价值偏低但总量价值巨大的数据进行存储、计算和交易。特别的是，Lambda平台上的数据都具有时间序列属性，而时间序列数据的价值一般随时间流逝而衰减，因此，数据的Owner可以通过智能合约自由选择存放时间，不同的存放时间对应不同的价格。

数据信任，个人消费者和企业消费者都需要有一个可信任数据云存储

在过去的十年里，云计算业务蓬勃发展，AWS全年收入超过百亿美元，SaaS成为美国经济发展的强劲驱动力，但是在很多时候，尤其是在缺乏信任基础的中国，很多公有云服务被转化为私有化部署和内部使用，企业客户的主要顾虑是担心SaaS企业后台会泄露数据。在消费者领域，随着精准广告技术的发展，给个人客户打tag成为精准营销的一个主要手段，而信息则更多来自于地下市场的交易。个人消费者的快递数据、电商数据甚至社交数据都被各家厂商通过各种方式进行变现，但作为数据的主人，个人消费者没有从社会和技术的发展中获益，甚至成为了各种骚扰的受害者。

通过永远在线的Lambda数据平台，企业和个人消费者、企业消费者之间可以构筑对于数据的智能合约约定，数据通过Lambda^2 Agent采集之后存放到区块链上面，并且在传递过程中进行脱敏处理。企业可以通过API使用Lambda 平台上面的经过计算的数据，而个人消费者可以得到TOKEN。

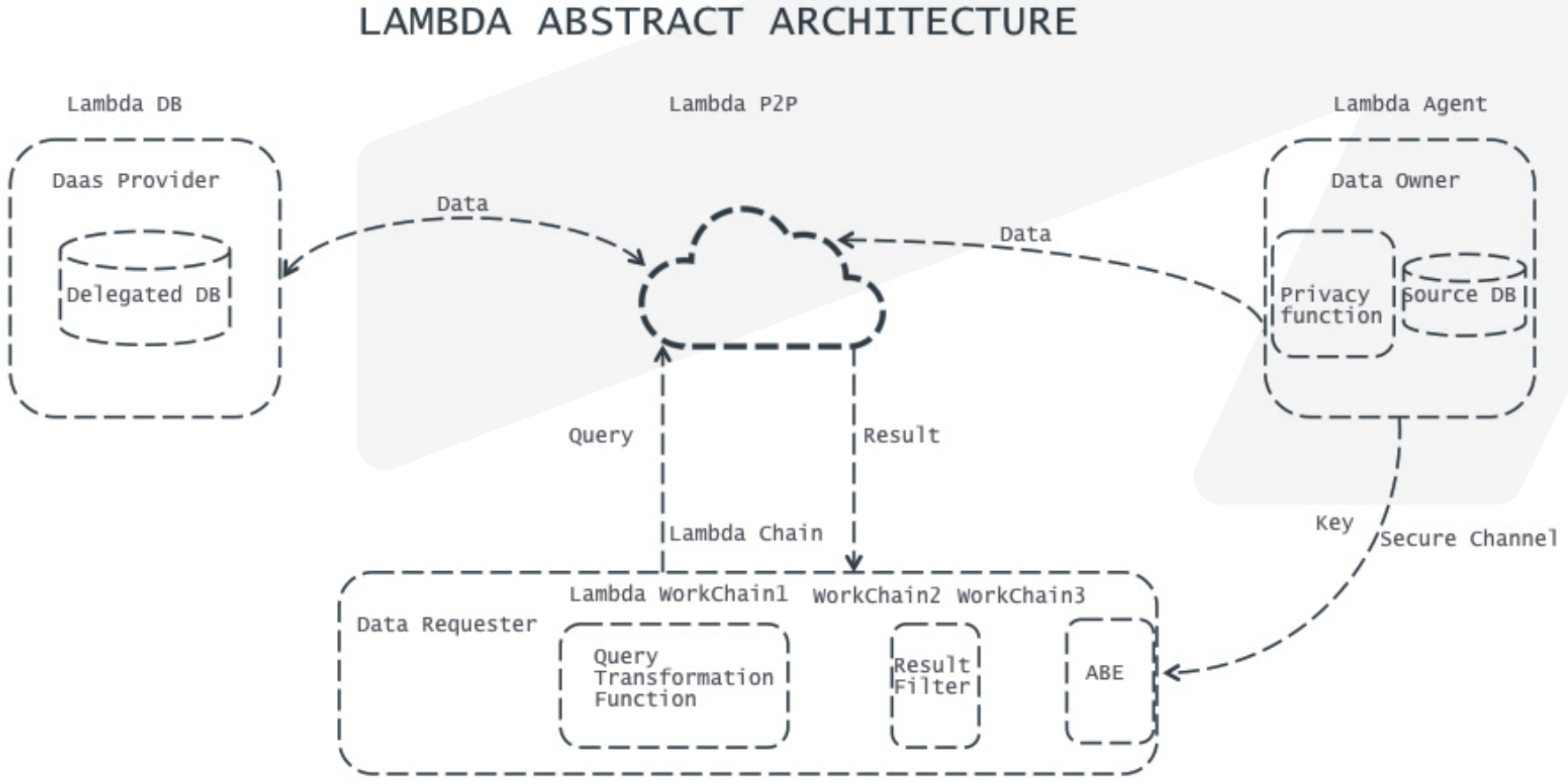
人工智能，数据是一切算法的驱动

人工智能技术现在已经影响到我们经济体系的方方面面，包括广告、金融、医疗、零售、自动驾驶、能源、运输和物流等行业。到2025年，人工智能技术相关的软件和服务将达到600亿美元的规模，覆盖到29个行业的150多种场景。人工智能的进步需要数据的支持，没有数据的支撑，人工智能的建模将失去准确性，而没有了准确性，意味着人工智能模型会失去作用。当前人工智能领域的领先公司，包括Google和Facebook，都是拥有大量数据资源的公司。

在Lambda平台得到发展以后，个人和个人，个人与组织，企业与企业，都可以通过Lambda进行原始数据和学习结果的交易，。从而使得数据变成信息，信息变成知识，知识最终成为全人类的智慧。

3/Lambda Project设计架构

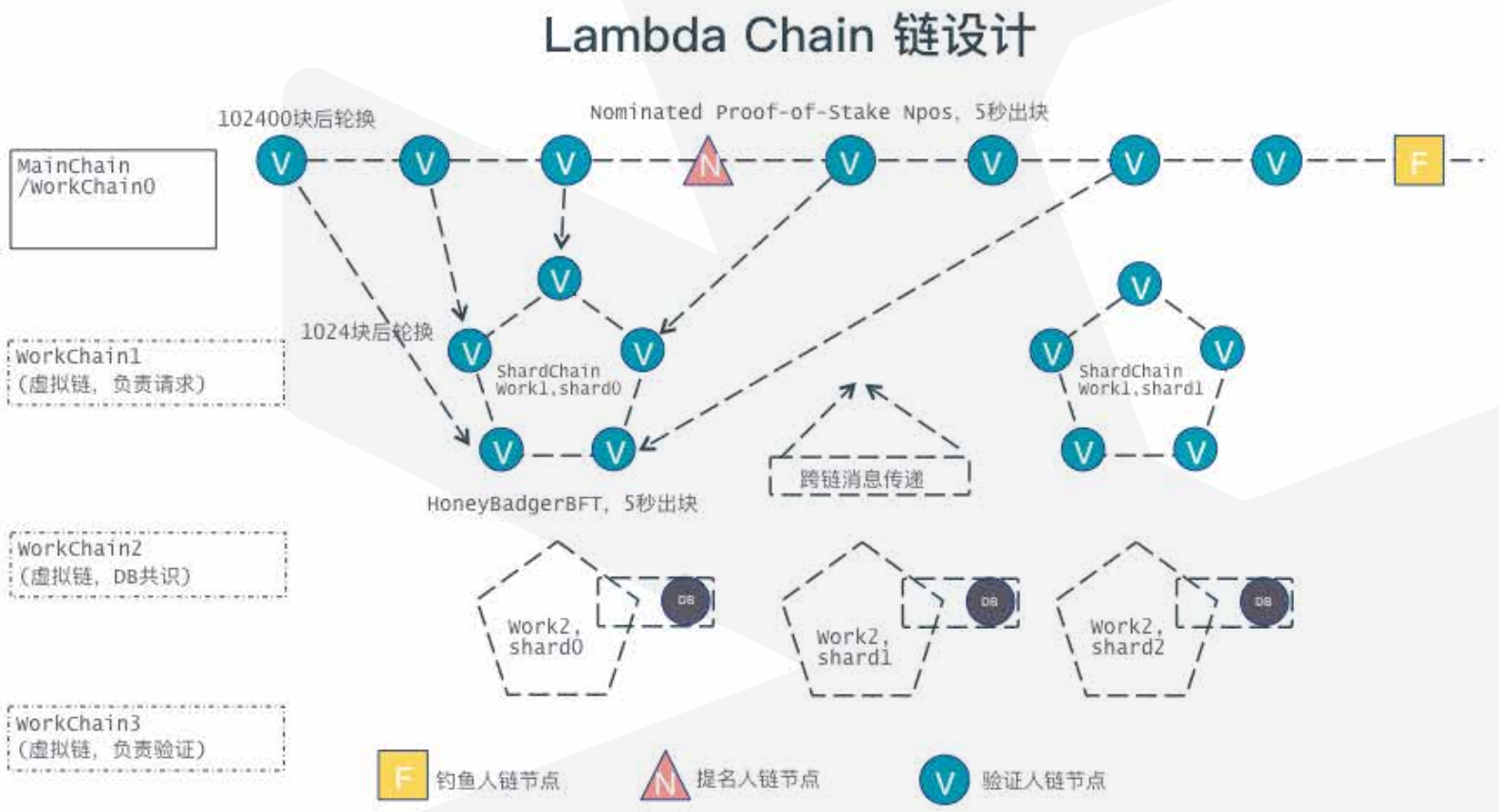
Lambda是一个高速、安全、可扩展的区块链基础设施项目，通过分片技术，可以处理每秒钟数百万笔的交易，通过安全的去中心化云数据库，为Dapp提供无限扩展的存储能力。Lambda是由以下几个部分组成，分别是：



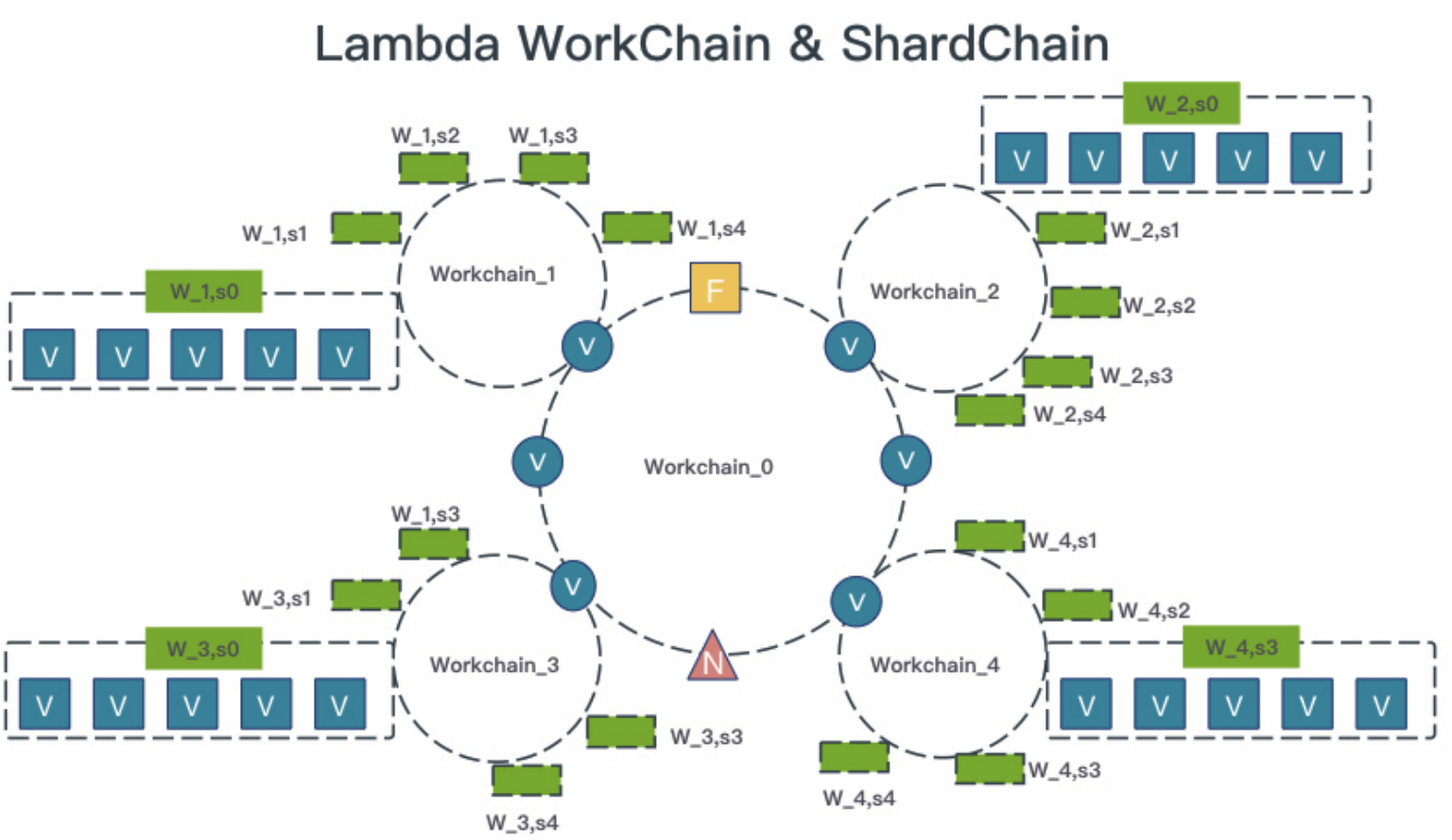
- 一个同构多链的链系统Lambda Chain，提供高TPS的访问能力、图灵完备的智能合约、跨链交易能力等
- 一个p2p的网络系统Lambda p2p，提供网络层的寻址能力
- 一个多数据库集群系统Lambda DB，提供可无限扩展的安全加密数据存储能力
- 一个Lambda DB的底层结构支撑系统，包括一个块存储系统和一个分布式文件系统Lambda FS
- 一个由多节点共识组成的属性基加密认证访问系统Lambda ABE，数据库的访问控制网关
- 一个由多个验证人节点组成的数据完整性验证组织Lambda TPA Chain(WorkChain3)
- 一个自适应的探针系统Lambda Agent，提供内存数据存储、性能监控、安全监控和Metrics数据上传能力

Lambda项目核心的整体设计思想是链库分离机制和功能子链设计。去中心化应用可以按照数据不同的信任和公开验证等级，将数据分别存放在链上和数据库系统中，Lambda项目提供了不同类型不同层级的数据协同管理。并且，由于整个Lambda DB是一个Permissionless的环境，Lambda还完成了基于多授权机构属性基加密的访问控制机制，以及完整的对存储数据的持有性证明。

链库分离设计的主要原因是为了系统未来的升级和更新考虑，由于区块链系统的更新会导致系统的分叉，总而对整个经济系统造成不可逆的影响，因此，我们把主要的数据处理能力放在数据库系统之上，把对于数据库系统的访问控制体系通过功能子链来完成。功能子链设计一是为了未来的扩展性，更多是为了完成去中心化存储系统的两个核心功能：隐私保护和数据持有性证明。我们通过一种高效的多授权机构属性基加密方案实现了对云存储数据访问控制功能和加密功能，并且通过PDP机制(Provable Data Possession, PDP)完成了数据的持有性证明。



Lambda链本身是一种同构多链的设计，从概念上来说，有三个层次和三个角色。三个层次又划分为两个真实链层和一个虚拟链层，分别是MainChain(真实层，WorkChain0)、WorkChain(虚拟层，WorkChain0到 WorkChainN)和ShardChain(真实层)，除了WorkChain0以外，其他所有的WorkChain都是由多个ShardChain组成的概念上存在的链。其中，MainChain的共识机制是Nominated Proof-of-Stake NPos，ShardChain的共识机制是HoneyBadgerBFT。MainChain是所有链的主链和中继链，拥有全部的节点，包括提名人节点、验证人节点和钓鱼人节点。任意一条WorkChain都是由MainChain指派的验证人节点负责交易的打包和出块，Sharding的机制则是通过对验证人节点的分组之后进行BFT共识来完成。因此，Lambda支持跨链的交易和跨链的通信。为了实现对数据库系统的管理，Lambda团队自己实现了前三条子链，分别是对数据请求(Request)进行授权、记录和请求转发的WorkChain1、对数据响应(Response)情况进行统计和对数据库节点进行共识管理的WorkChain2，以及对数据完整性验证的WorkChain3。未来，通过对更多子链的加入，Lambda可以实现更多的功能。



Lambda三个角色的设计分别是提名人、验证人和钓鱼人：

验证人：验证人有最高的权限，在整个Lambda网络中进行交易的打包和出块，验证人需要抵押足够多的TOKEN。另外，我们允许拥有资金的提名人推举一个或者多个代表他们的验证人，所以，验证人的押金不完全是自己所拥有的，有部分是属于提名人的。验证人的硬件环境必须符合相应的要求，硬件要求主要是针对CPU内核数、内存、SSD硬盘，验证人的网络环境必须符合高可用、高网络带宽以及低延迟的要求。验证人在以上符合要求的机器环境中运行我们的中继链的全账本客户端，也就是MainChain的客户端。在每个区块上，节点都准备验证和打包已经在子链上完成了共识的区块的哈希值。另外，验证人节点被随

机分配到不同的WorkChain上面，进行子链交易的验证打包和出块。主链和子链都是每间隔5秒出一个块，每过1024个块，验证人节点会进行轮换。在我们选择的共识算法下，会惩罚一个没有履行职责的验证人，如果是偶发性的错误，只是会扣留出块奖励，但重复的错误会导致扣减验证人的押金（通过燃币），严重的错误如双签或合谋攻击等情况发生的时候，会扣除该名验证人全部的押金，燃烧一部分，并将大部分奖励给诚实验证人和提供信息的钓鱼人。

在某种程度上，验证人和比特币的矿池作用类似，我们预期系统会有数百个验证人节点。

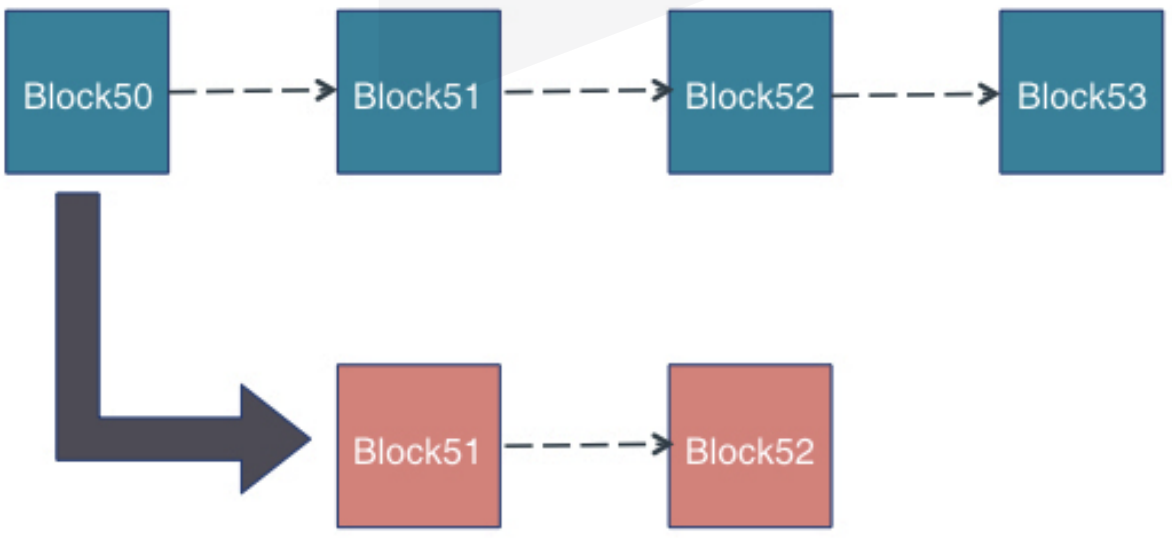
提名人：提名人是一个拥有权益的群体，他们把安全性押金委托给验证人，他们除了投入资金以外，没有更多的职能。

在我们的设计中，每一个存储节点/数据库节点也都要进行安全性押金的委托，因此，每一个数据库节点和存储节点也都是提名人。

提名人和比特币系统的矿工类似，他们起到了监督的作用。我们预期全网有数千名提名人

钓鱼人：钓鱼人和区块打包的过程并不相关，他们的角色类似于现实世界中的“赏金猎人”，激励他们的是一次性的大额奖励。由于钓鱼人的存在，我们才能减少恶意行为的发生。钓鱼人只需要及时举报并证明至少一个有抵押的参与方存在非法行为，他们就能获得奖励。比如说对有相同父块的两个区块进行签名或者在子链上打包出一个无效区块。成为钓鱼人的要求是最低的。

短程攻击、长程攻击和检查点机制：对于一个POS的共识机制来说，一个必须要解决的问题就是恶意的验证人的短程攻击和长程攻击。我们通过验证人抵押资金的方式，来避免验证人对两个高度相同的区块的同时签名。对于长程攻击来说，我们计划采用类似于Polkadot和Casper项目的检查点机制，来引入强制确定性(Finality)的概念。



权益合约：我们通过权益合约来管理整个验证人集合，合约主要解决这些问题：

1 哪些账户是验证人

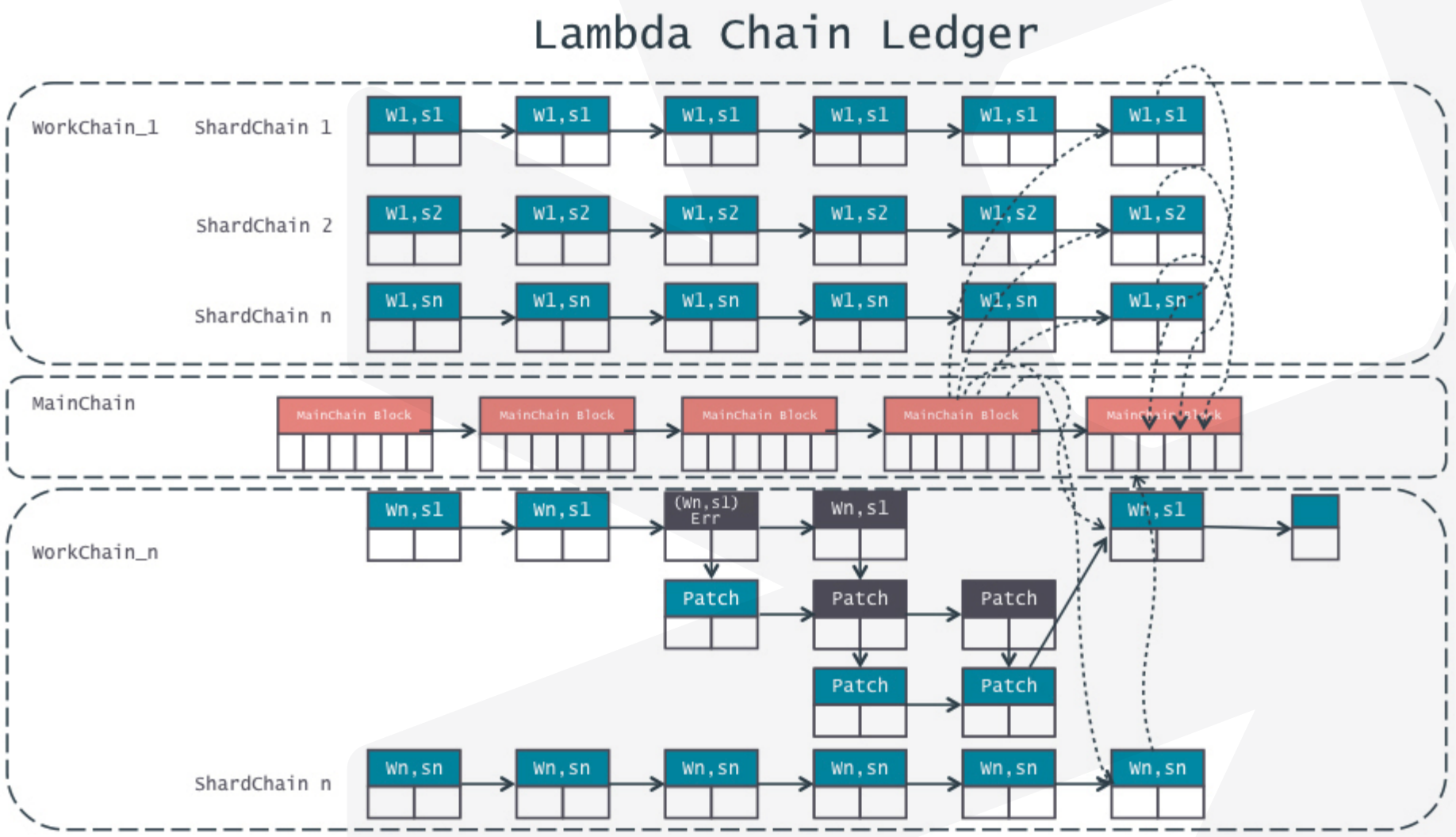
2 哪些账户在短期内可以变成验证人

3 哪些账户为了提名验证人而进行了TOKEN质押

4 每个账户的属性状态，包括账户余额、接收的质押比例、地址列表等

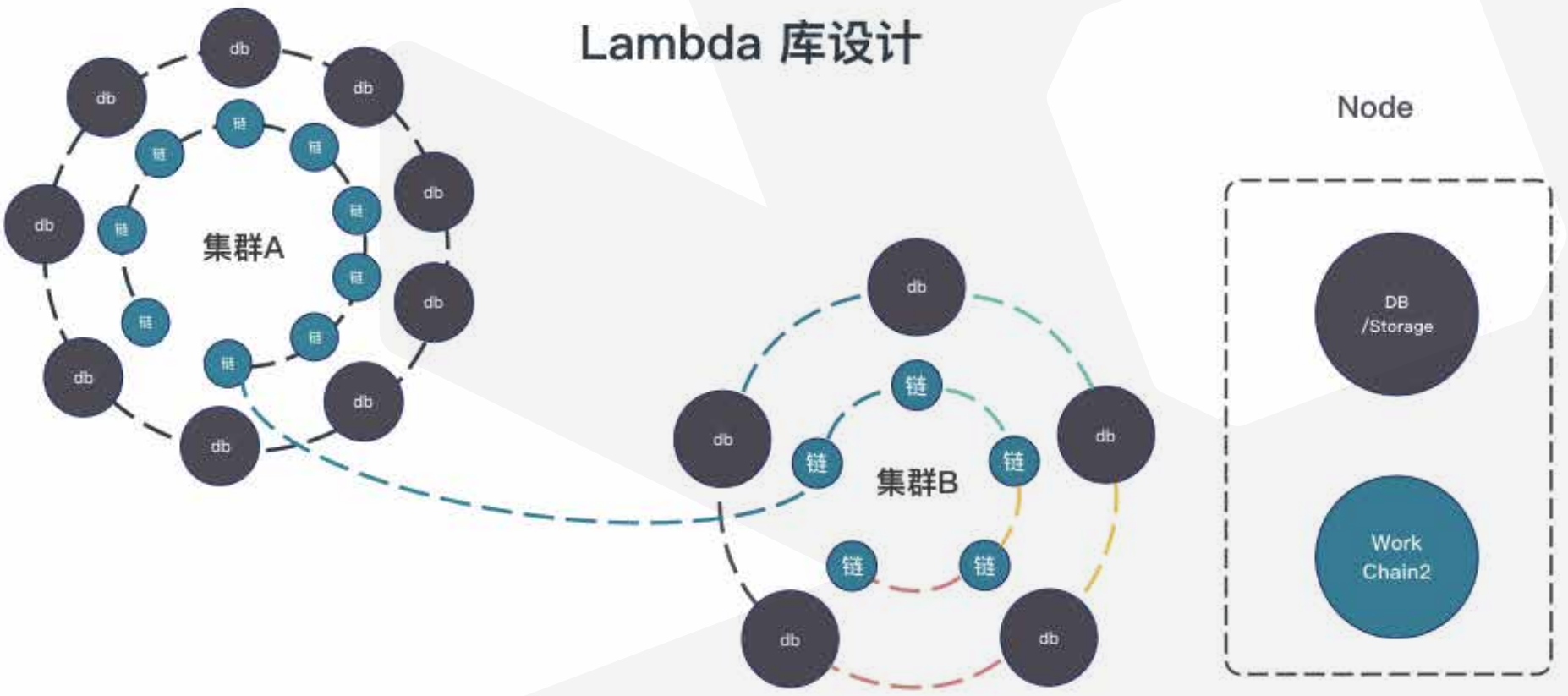
账本结构：MainChain是Lambda中的总账本，每个Workchain都会根据负载情况进行自动拆分与合并，通常情况下一个WorkChain会根据所处理的账户前缀拆分为多个ShardChain，每个ShardChain均为一条完整的区块链。为了使Lambda中主账本与分片账本相互验证形成网状的互验证结构，每个MainChain的新区块中包含了ShardChain所有新区块的哈希（除非部分ShardChain在超时时间内未出区块），ShardChain的新区块包含了上一个MainChain的区块的Hash。

在Lambda中，引入了FishMan角色，即该节点可以对历史区块的有效性发起验证，如果检测到该区块无效或区块中的Transaction无效，则会发起更为严厉的验证并对历史区块进行修复，所以Lambda为每一个区块本身也设计为一个blockchain，正常状态下该blockchain为一个区块，当该区块处于修复的情况下，会在该区块的垂直方向产生新区块对对应历史区块进行整体替换或修复失效区块中的部分Transaction信息。同时引用该失效区块的其他区块同样需要进行修复直至所有状态修改正确为止。所以Lambda区块链账本不会产生分叉，即使部分交易遭到破坏，Lambda可以通过垂直链进行修复，并同时保存了所有历史交易的有迹可寻。



其他未在本文描述的问题：关于WorkChain的注册、跨链交易的消息和队列、p2p网络架构设计、垂直链账本、自我修复能力等基础设计，以及业务逻辑中的数据请求统计、数据响应统计、请求响应对账、数据库节点加入交易、数据库节点离线共识处理、Lambda　FS详细设计等，限于篇幅，我们未将此类问题在本文中列出，在正在编写的技术黄皮书中，我们会详细描述此类问题。

关于链架构的思路
Lambda最关键的特性，也是Lambda优于IPFS的地方，是对于数据的持有性证明验证机制。任何存储类的项目都理论上来说都必须提供持有性证明机制，而任何可实现的验证机制，必然存在着挑战方和证明方。在已有的存储类项目中，Sia和Storj都是选择让用户作为挑战方，这要求用户端自己需要保存文件的副本，或者是在特定时间内拥有文件副本，这种机制只能用来实现网盘，而无法用来实现云存储和云数据库。IPFS是让存储节点既作为挑战方，又作为证明方，现在看来这种机制是无法实现的，这引入了大量的复杂逻辑，比如零知识证明等等。对于我们来说，我们需要链节点作为挑战方，然后，这就带来了两个特性： 1　我们的验证子链节点需要Sharding，也就是说，不能所有节点(几千个)同时去验证一个文件，这是极大的资源浪费。同时，验证节点对于验证结果要有强一致性共识，也就是说，通过BFT形成共识。那么，这就决定了我们需要有一条Sharding的链，而且Sharding内部的共识机制是BFT，也就是强一致性共识。 2　我们需要一条另外的链来实现属性基加密，并且对整个后端的存储和数据库资源池进行访问控制，通常来说，属性基加密是基于一个可信的密码授信中心，而我们用一条链和多个机构来代替可信中心，这就需要validator角色的引入和BFT的共识。 3　我们需要一条链对数据库节点的加入和数据库节点的离线形成共识，共识算法也是BFT。这条链的节点和数据库的节点是同一个runtime，并且需要TEE和SGX的保护。 4 我们需要一条链来统计用户对数据库系统的请求，并且这条链是一条高速链。 所以，现在提出的需求是： 1 Sharding　2 高速子链　3 同构验证机制如BFT　4 链间通信



在进行数据系统设计的时候，为了保证服务的可用性、性能、可扩展性，我们采用数据库集群的技术，向去中心化应用提供服务。集群是Lambda数据库的一个基本概念，在提供服务的时候，数据总是在一个数据库集群内部进行副本的复制，而不会跨越集群。另外，集群技术可以使负载在所有的设备之上保持相对的均衡。

Lambda在设计数据库系统的架构时，实现了在异构环境下均衡分布数据的CCHDP算法。目前,大部分的布局算法面向同构的存储系统，而互联网的设备主要是异构设备，在这些设备中有效地存放PB规模的数据,是一个具有挑战性的问题，不合理的数据布局将会导致大量的存储空间白白浪费,而且用户需要花费大量的时间寻找所请求的数据,因而数据布局算法在大规模网络存储系统中非常重要，并且均衡地使用设备容量可以平衡I/O负载。

为了尽可能地发挥每个设备的存储能力,使得存储系统的整体性能最大化, Lambda会根据设备的容量以及带宽公平地分配数据。而且,由于整个系统会不定时添加新的节点以及删除不合格的设备,或者某些节点的设备发生失效等.为了让存储系统的性能最大化,需要重新公平地分布数据。Lambda数据库系统的某些应用可能是24 小时运转的,迁移大量的数据必然会消耗大量的带宽,降低Lambda数据库系统对应用的服务质量,甚至影响整个系统的可用性。为了不影响系统的存储服务， Lambda的算法保证了再次分布的数据量要尽可能地少。因而， Lambda数据库系统的数据布局算法能够动态自适应存储规模的变化,在系统规模变化时迁移尽可能少的数据.同时,也保证了布局算法能够利用很少的时间和空间信息计算出某个数据的存放位置。

Lambda使用的CCHDP(clustering-based and consistent hashing-aware data placement)算法可以在异构的设备集合中按照设备的权重公平地分布数据,自适应存储规模的变化,在存储规模变化时迁移最少的数据量,而且能够在较短的时间内计算数据的位置,仅需要引入少量的虚拟设备,大大减少了存储空间，CCHDP是一个聚类和一致性哈希技术相结合的算法。

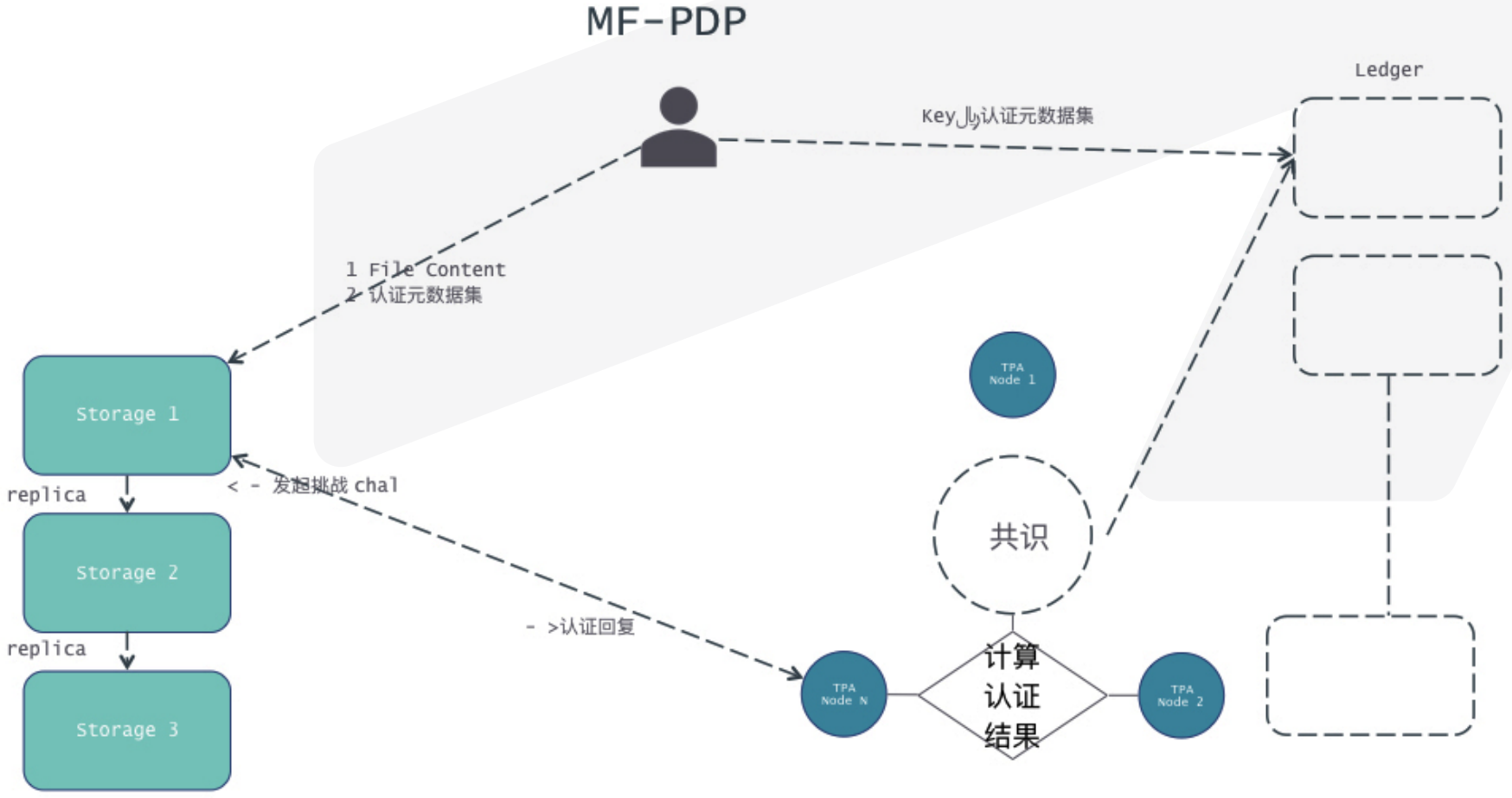
4/Lambda 数据完整性验证的基本原理(选读)

在云计算技术发展初期，学术界就将数据的完整性证明和持有性证明视为研究的前沿领域，并不断的进行相关的探索。在常规云计算环境中，由于大规模数据所导致的巨大通信代价,用户不可能将数据下载后再验证其正确性.因此,云用户需在取回很少数据的情况下,通过某种知识证明协议或概率分析手段,以高置信概率判断远端数据是否完整。典型的工作包括:面向用户单独验证的数据可检索性证明(POR)方法、公开可验证的数据持有证明(PDP)方法。NEC实验室提出的PDI(provable data integrity)方法改进并提高了POR 方法的处理速度以及验证对象规模,且能够支持公开验证。其他典型的验证技术包括:Yun 等人提出的基于新的树形结构MAC Tree 的方案;Schwarz 等人提出的基于代数签名的方法;Wang 等人提出的基于BLS 同态签名和RS 纠错码的方法等。

去中心化的云存储和云数据库，相比较于中心化的云，其最大区别在于Permissionless的存储节点，并且，云用户不可作为验证发起人，也没有完全可信的TPA节点。在Lambda的设计中，我们采用的是支持公开验证的PDP方法的两个升级版本，BLS-PDP和MF-PDP，我们将通过多个验证人节点的共识来共同一个可信TPA的工作，也就是完成数据的持有性和完整性验证，并将验证结果写到链上，以备FisherMan查验。在区块链系统中，由于数据本身具备按照时间的自增加特性，没有更新和删

除的逻辑，数据的修改模式为just append。另外，在云存储环境下，针对动态数据更新问题,我们观察到,云存储有别于传统存储(如文件系统)的数据更新模式,即，典型的数据更新操作不是对文件内容的插入或修改,而是静态文件数量的变化(通常为增加)，因此，我们可以将数据库的表空间文件视为具备静态文件的特点。

Lambda验证方案中的BLS-PDP基于BLS同态签名算法，BLS签名机制是由Boneh等人提出的一种短消息签名机制，相比目前最常用的两种签名机制RSA 和 DSA 而言, 在同等安全条件下(模数的位数为 1024bit), BLS 签名机制具有更短的签名位数,大约为 160bit(RSA 签名为 1024bit, DSA 签名为320bit). 另外, BLS 签名机制具有同态特性, 可以将多个签名聚集成一个签名. 这两点好的特性使得基于 BLS 签名的 PDP 机制可以获得更少的存储代价和更低通信开销来实现. 基于 BLS 签名的 PDP机制是一种公开验证机制, 用户可以将繁琐的数据审计任务交由我们的WorkChain3 来完成, 满足了云存储的轻量级设计要求, 具体实现如下图所示. 另外, 相比其他机制而言, 该机制具有更小的存储开销和通信开销。



Algorithms of BLS-PDP scheme

选取公共参数e: $G \times G \rightarrow G_T$为双线性映射，g为G的生成元，H:$\{0,1\}^* \rightarrow G$为BLS哈希函数。

Setup阶段：

随机选取私钥 $\alpha \leftarrow Z_p$，计算相对应的公私$\upsilon = g^{\alpha}$，私秘的sk = (α) 公钥为pk = (υ)

随机唯一的文件表示$\upsilon \xleftarrow{\mathcal{R}} \{0,1\}^{\mathcal{K}}$，随机选取辅助变量$\upsilon \xleftarrow{\mathcal{R}} G$ 对文件进行分块 :F = $\{\mathfrak{b}_1, \ldots, \mathfrak{b}_n\}$，生成认证元数据集合：$\Phi = \{\sigma_i\}_{1<i<n}$，这里$\sigma_i = (h(\nu \| i) \cdot u^{m_i})^{\alpha}$ 将 F和认证元数据Φ存入远程服务器

Challenge阶段：

（1）验证者从块索引$[1,n]$随机选取$\mathfrak{c}$个块索引，并对每个块索引选取一个相应的随机数$\nu_i \xleftarrow{R} Z_{p/2}$组成挑战请求$chal = \{i, \nu_i\}_{s_1<i<s_c}$，并将其发送给证明者

（2）证明者接收到chal请求后首先计算$\sigma = \prod_{i=s_1}^{s_c} \sigma_j^{v_i} = \prod_{i=s_1}^{s_c} H(\nu \| i)^{v_i} u^{v_i m_i}$;之后计算

$\mu = \sum_{i=s_1}^{s_c} v_i m_i = v_1 m_{s_1} + \ldots + v_s m_{s_c}$ 将$\{\sigma, \mu\}$作为证据返回给证明者。

（3）验证者接收到证据$\{\sigma, \mu\}$后，根据以下等式判断外包数据是否完整：

$$e(\sigma, g) \stackrel{?}{=} e\left(\prod_{i=s_1}^{s_c} H(\nu \| i)^{v_i} \cdot u^{\mu}, \nu\right)$$

Algorithms of MF-PDP scheme

KenGen()→(pk,sk)

1. 生成公钥pk=(N,e,g)和私钥sk=(N,d,g).
2. 输出(pk,sk).

Add(usk,F,α,GID)→(F′,M,α′)

1. 令(N,d,g)=usk,(B)=α.
2. 将文件F 分割成t 个数据块,F[1],…,F[t],每个数据块的长度为L 位.
3. 对于1≤i≤t:

   a) 计算数据块F[i]的同态认证标签:A[i]= $((H(GID)||B+i)\cdot g^{F[i]})^{d}$ mod N.
4. 记录F 在文件组中的下标范围R=(B,B+t).
5. 更新文件组最大下标B′=B+t.
6. 输出(F′=F,M=(A[1],…,A[t],R),α′=(B′)).

Challenge(α)→chal

1. 生成随机挑战密钥 k_1和 k_2: $k_1 \xleftarrow{R} \{0,1\}^k$, $k_2 \xleftarrow{R} \{0,1\}^k$.
2. 根据目标安全等级定义将要挑战的数据块的数量c.
3. 输出chal=(c, k_1, k_2).

Prove(upk,chal,F′,M,α)→P

1. 令(N,e,g)=upk,(c, k_1, k_2)=chal,(B)=α
2. 对于1≤j≤c:

   a) 计算要检测的数据块的虚拟下标值:$i_j = \pi[B]_{k_i}(j)$;

   b) 计算系数: b_j= $e_{k_2}(j)$;

   c) 定位要检测的数据块及它们的同态认证元: $\mathrm{f}\left[i_j\right] = F'\left[i'_j\right] = F'\left[i_j - F'!.R.start\right], a\left[i_j\right] = M.A\left[i'_j\right] = \mathrm{M.A}[i_j - F'!.\mathrm{R.start}]$
3. 计算 $\mathrm{a} = \mathrm{a}[i_1]^{b_1} \bullet \ldots \bullet a\left[i_c\right]^{b_c} \bmod \mathrm{N}$ ，这个值应与 $(H(GID||i_1))^{b_1} \bullet \ldots \bullet H(GID||i_c)^{b_c} \bullet g^{b_1 f[i_1]+\ldots+b_c f[i_c]})^{d} \bmod \mathrm{N}$ 相等.
4. 计算f=b_1f[i_1]+...+b_cf[i_c].
5. 输出P=(a,f).

Verify(upk,chal,P,α)→{0,1}

1. 令(N,e,g)=upk,(c, k_1, k_2)=chal,(a,f)=P,(B)=α.
2. 对于1≤j≤c:

   a) 计算 i_j =$\pi\left[\mathrm{B}\right]_{k_1}(j)$和$b_j = e_{k_2}(j)$
3. 计算τ_1=$a^e mod\ N$,$\tau_2 = H(GID||i_1)^{b^1} \bullet \ldots \bullet H(GID||i_c)^{b_c} \bullet g^f\ mod\ N$.
4. 如果$\tau_1 = \tau_2$则输出1,否则输出0.

5/Lambda 的多授权机构属性基加密数据访问控制方案(选读)

当前，区块链上的数据都是可公开访问的数据，这极大限制了数据的使用场景，为了扩展使用场景，Lambda提供了基于多授权机构属性基加密的访问控制方案，并提供了数据加密的能力，以及通过代理加密对属性撤销的能力。属性基加密(attribute-based encryption,简称ABE)机制以属性为公钥,将密文和用户私钥与属性关联,能够灵活地表示访问控制策略,从而极大地降低了数据共享细粒度访问控制带来的网络带宽和发送结点的处理开销.因此,ABE 在细粒度访问控制领域具有广阔的应用前景。

在属性基加密(Attribute-Based Encryption,简称ABE)方案中,其密钥构成与属性集合相关,密文构成与访问结构相关;如果属性集合能够满足密文中的访问结构,便可以获取明文。ABE 不仅具有一对多的特点,而且可以实现不确定用户数的解密,因此被广泛地应用于云存储数据的细粒度访问控制。但常规ABE加密计算代价大,并且计算开销随着访问结构中属性个数的增加而线性增加,不适合直接应用于电力资源有限的移动终端，因此不适合在区块链领域直接使用。

在线-离线(Online-Offline)和转换密钥技术可以通过预处理及外包解密的方式来降低用户端加密和解密的计算开销,但前者需要在离线加密阶段确定访问结构,实际上不同数据的访问结构并不相同,不便于提前确定;后者通过把解密外包到不完全可信的第三方,不能保证解密的正确性.尽管Shao 等人提出的方案可以不用提前确定访问结构,但是用户的属性集合只受到一个属性授权机构的管理,不利于系统规模的扩充;而且其没有验证外包解密的正确性.。

面对以上的挑战, Lambda采用了一个在线- 离线的多授权机构属性基加密(Online/Offline Multi-Authority Attribute Based

Encryption,简称OO-MA-ABE)方案,其主要思想是把用户端在线计算代价转移到离线阶段或者云服务器上.本方案的主要优势如下:

1) Lambda利用在线-离线和外包解密技术,构造了一个高效的移动云存储数据访问控制方案.在加密阶段,把

大量的配对操作提前预处理;在解密阶段,把配对操作外包到云存储服务器;用户端只需要运行简单的运算操作就可以完成加密和解密,从而大大降低用户端的在线计算开销。

2) Lambda提出了一种验证外包解密正确性的方法.在加密阶段,用户利用加密密钥和明文生成哈希值作为数据的验证令牌;在解密阶段,用户用验证令牌验证解密结果的正确性,进而检验云存储服务器解密是否正确.同时,本文方案可以抵抗单个授权机构获取用户的身份信息,保证了用户身份隐私。

3) Lambda团队对方案进行了安全分析和仿真实验,结果表明本方案的安全性和高效性。

ABE机制的形式化定义

定义**1**(访问结构). 假定$\{P_1, P_2, P_3, P_4 \ldots P_n\}$是参与方的集合,$P = 2^{\{P_1, P_2, P_3, P_4 \ldots P_n\}}$.访问结构A是$\{P_1, P_2, P_3, P_4 \ldots P_n\}$的非空子集,即$A \subseteq P \backslash \{\varnothing\}$.若访问结构A 是单调的,则$\forall B, C$,若$B \in A$ 且$B \subseteq C$,那么$C \in A$.

定义**2**(双线性对). 映射$e: G_1 \times G_1 \to G_2$ 若满足下列特征就是双线性对:(1) 双线性:$\forall a, b \in Z_q, \forall f, h \in G_1$,都有$e(f^a, f^b) = e(f, h)^{ab}$称映射$e: G_1 \times G_1 \to G_2$ 是双线性的;(2) 非退化性:$\exists f \in G_1$使$e(f,f) \neq 1$:(3) 可计算的:$\forall f, h \in G_1$,存在一个有效的算法计算$e(f,h)$.注意:$e(*,*)$是对称操作,即$e(f^a, h^b) = e(f,h)e(f, h)^{ab} = e(f^b, h^a)$.

定义**3**(计算!**Diffie-Hellman**(**CDH**)问题). 随机选择$a, b \in Z_q^*$,给定三元组(g, g^a, g^b),计算g^{ab}.

定义**4**(判定双线性**Diffie-Hellman**(**DBDH**)). 随机选择$a, b, c \in Z_q^*$,$R \in G_2$,给定元组(g, g^a, g^b, g^c, R),判断等式$e(g, g)^{abc} = R$ 是否成立.

定义**5**(判定线性(**D-Linear**). 随机选择阶为q 的群G 的生成元g, f, v,随机选择指数$a, b \in Z_q$,$R \in G$,给定元组(g, f, v, g^a, f^b, R),判断等式$v^{a+b} = R$ 是否成立.

定义**6**(选择密文攻击(**IND-CCA**)游戏). 敌手和受挑战者进行如下交互:(1) 受挑战者对加密方案进行系统建立,输出公私钥对,并将公钥交给敌手;(2) 敌手可以向受挑战者进行一些解密询问,受挑战者解密密文,返回结果给敌手;(3) 敌手选择两个明文M_0, M_1 ,然后发送给受挑战者.受挑战者投掷一个公平硬币$b \in \{0,1\}$,对明文M_b加密,得到密文C^*并发送给敌手;(4) 敌手可以继续向受挑战者进行一些同步骤(2)中的解密询问,但询问密文不能为C^*;(5) 最后,受挑战者必须回答0 或者1(记为b'),作为对密文C^*的猜测.

若$b' = b$,则敌手在该游戏中获胜.敌手在游戏中的优势定义为$\Pr[b' = b] - 1/2$.

6/典型案例场景之物联网

随着传感器成本的逐渐降低和网络速度的飞速提升，物联网迎来快速发展，但是，在技术落地过程中，物联网也存在着很多问题，比如可扩展性、安全性、价值流通性等。人们逐渐发现区块链的技术可以解决物联网领域的诸多问题，比如身份识别、互联互通等，但从实际情况来看，现存项目由于缺乏对于数据层面的关注，从而在落地的时候遇到诸多困难。

首先，物联网的数据上链，缺乏合适的数据载体。物联网数据是海量的、严格按时间递增的、结构非常简单的各种指标（英文为 metric、measurement）数据。其产生过程可以保证数据的真实性，而且数据本身从逻辑上来说不需要修改，也不需要更新。但是目前区块链并没有适合这种数据存放的数据库，当前的时间序列数据库大多是构建在传统数据存储如HBase、Rethinkdb之上，最终数据文件会保存在中心化的数据库之中，这些数据可能会被人为篡改，也有可能会泄露和丢失。

其次，物联网数据存储占用空间数量巨大，数据的规模大多是以TB甚至是PB来计算，但是单个数据价值微小，但合在一起的众多数据价值巨大，从数据中提取价值需要持久的工作，这个过程类似淘金，而这需要大量的存储资源以及适量的计算资源。

物联网产生的时间序列数据带有时间戳、本身逻辑上不需要更改这两个特性，决定了区块链技术在物联网数据存储领域必然存在的价值。区块链的Token可以衡量微小的价值，激励人们共享和贡献计算以及存储资源以获取代币的奖励。这两个领域价值和想象空间都是巨大的，比如说卫星遥感数据、水文数据、大气数据等，通过传感器收集的数据可以按照时间段、地域、事件变更等因素，通过智能合约出售给任何需要和感兴趣的人以及机构。

同时，在区块链领域，现存的基础链如以太坊等协议并没有对数据存储等问题进行很好的解决，很多数据存储类的项目都是早期基于比特币的协议，在今天看来已经过时，比如Sia和Storj。在以太坊协议设计的时候，主要考虑的是金融体系的需求，

因此对于图灵机的实现考虑较多，对于数据的输入考虑较少，并且完全没有考虑到数据库类型的需求。

Lambda项目提供了一个面向物联网领域的数据收集、数据传输、数据存储、数据计算和数据交易体系，这个体系由Lambda Agent、Lambda CEP和Lambda DB和Marketplace四部分组成。其中，Lambda DB在区块链技术上构建一个时间序列数据库系统，不同于BigChainDB这样的底层基于传统数据库RethinkDB的实现，Lambda DB基于Amazon Dynamo论文，并将底层的数据文件分发到各个节点，并将哈希信息存储到区块链上。在Lambda DB完成之后，物联网数据的存放将变得和传统编程领域一样简单，而且自动解决了可扩展性、可用性、安全性等问题。Lambda Agent提供了应用程序快速部署、开发、监控和安全保障能力，所有应用程序都通过调用Lambda Agent来传递消息到Lambda DB。

为软件开发者和投资者获取分布式计算和存储的价值，Lambda项目将自主开发使用基于原生区块链的Cryptocurrency :LAMB。目前世界上所有的在线服务商都需要提升其产品的计算能力，包括IOT、IOE等，大数据公司以及各种流行的应用程序，尤其是在区块链领域，算力主要是由矿工来提供，Lambda项目提供了另外一种实用性更加广泛的解决方案。世界上所有依赖于互联网运行业务个人或厂商都可以选择使用LAMB Cryptocurrency来解决他们迫切需要的计算能力问题。而且，所有的互联网用户都可以通过租赁计算资源给LAMB项目来获得不间断的收入。

特别的，Lambda项目的雾计算架构更加适用于物联网场景。IoT（物联网）/IoE（万物联网），作为现在世界互联网计算能力的重要组成部分，同时也是Lambda项目的突破和应用的主要方向。本项目加入了高性价比的雾计算而不是完全采用耗用较大的的云计算，与云计算相比，雾计算所采用的架构更呈分布式，更接近网络边缘。雾计算将数据、数据处理和应用程序集中在网络边缘的设备中，而不像云计算那样将它们几乎全部保存在云中。数据的存储及处理更依赖本地设备，而非服务器。

物联网Use Case

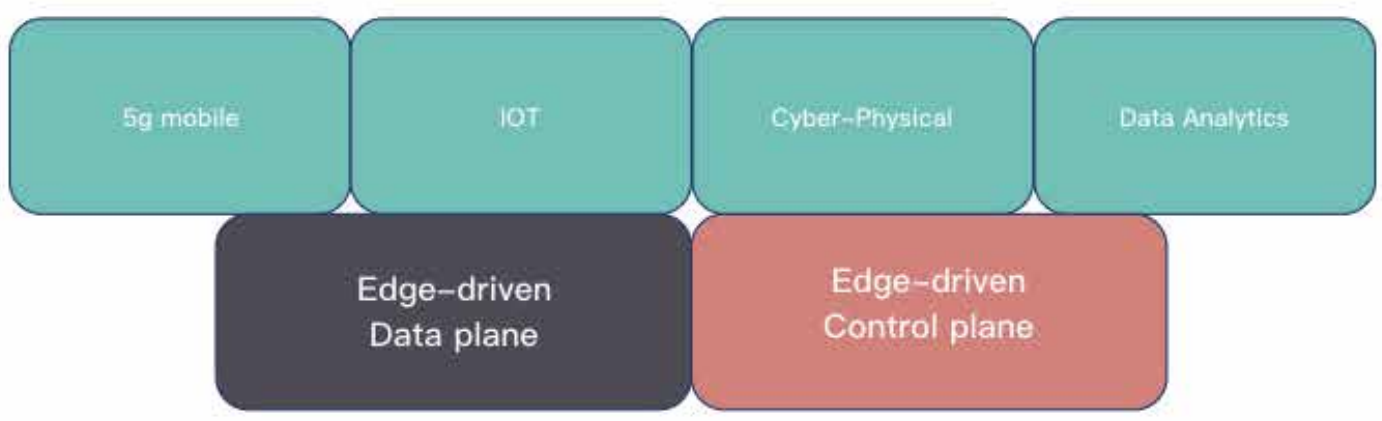


Figure 1: Data plane and control plane of Fog networks enable different applications

7/典型案例场景之人工智能

人工智能技术现在已经影响到我们经济体系的方方面面，包括广告、金融、医疗、零售、自动驾驶、能源、运输和物流等行业。到2025年，人工智能技术相关的软件和服务将达到600亿美元的规模，覆盖到29个行业的150多种场景。人工智能的进步需要数据的支持，没有数据的支撑，人工智能的建模将失去准确性，而没有了准确性，意味着人工智能模型会失去作用。当前人工智能领域的领先公司，包括Google和Facebook，都是拥有大量数据资源的公司。

在人工智能领域，目前的共识是数据的作用比模型大很多倍，然而，人工智能数据的获得是一个很困难也很昂贵的事情，

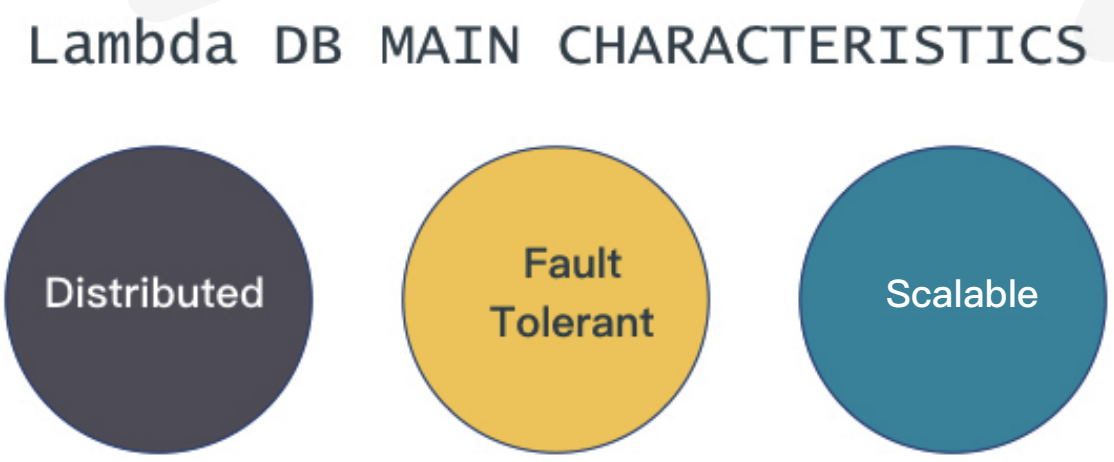
数据的缺乏，限制了人工智能行业的发展。当前，由于缺乏信任，数据滥用等多方面的原因，自由个体不愿意将数据提供给人工智能技术的拥有者，而需要数据的商业机构更多通过灰色的手段去获取数据。

Lambda项目最终会形成一个全球所有时间序列数据的一体化网络，数据的拥有者可以在Lambda平台的Marketplace上通过智能合约来出售自己手中的数据，特别是，时间序列数据的价值在于分析之后的洞察，也就是说，数据的拥有者无需出售原始数据，只需要出售对于数据的分析结果。甚至，只需要出售对于数据分析结果的洞察。

未来，Lambda项目将通过零知识证明等技术，进一步发展关于数据交易的核心场景相关的智能合约技术。

二、Lambda DB技术原理(库的设计)

Lambda DB的特点是完全的去中心化、分布式、分区容错性和极度的扩展性。



去中心化：没有中心节点

对称性：所有节点都是对等的

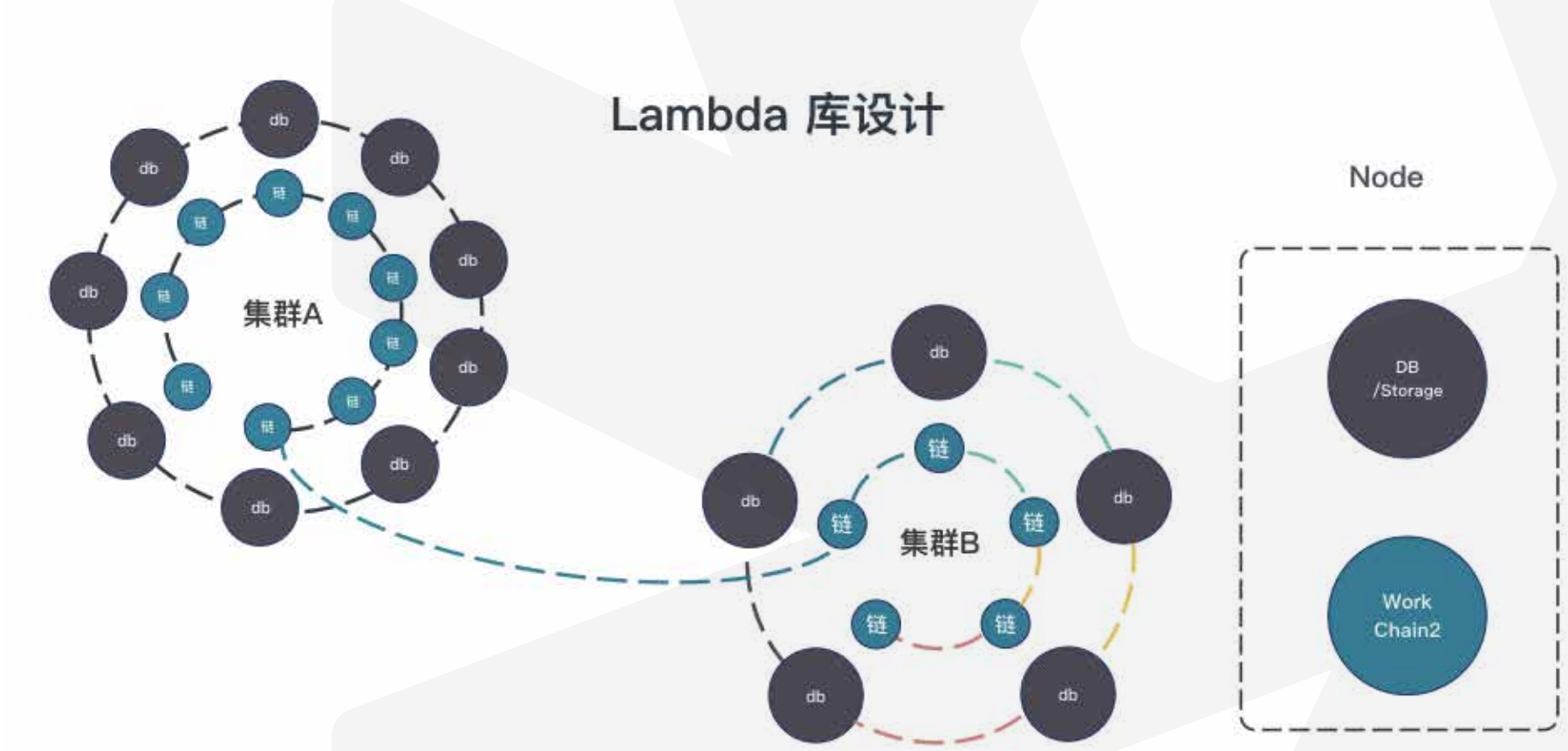
可扩展性：采用增量扩展策略

微粒性：节点的计算能力可以细分

Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

Table 1: Summary of techniques used in Lambda DB and their advantages.

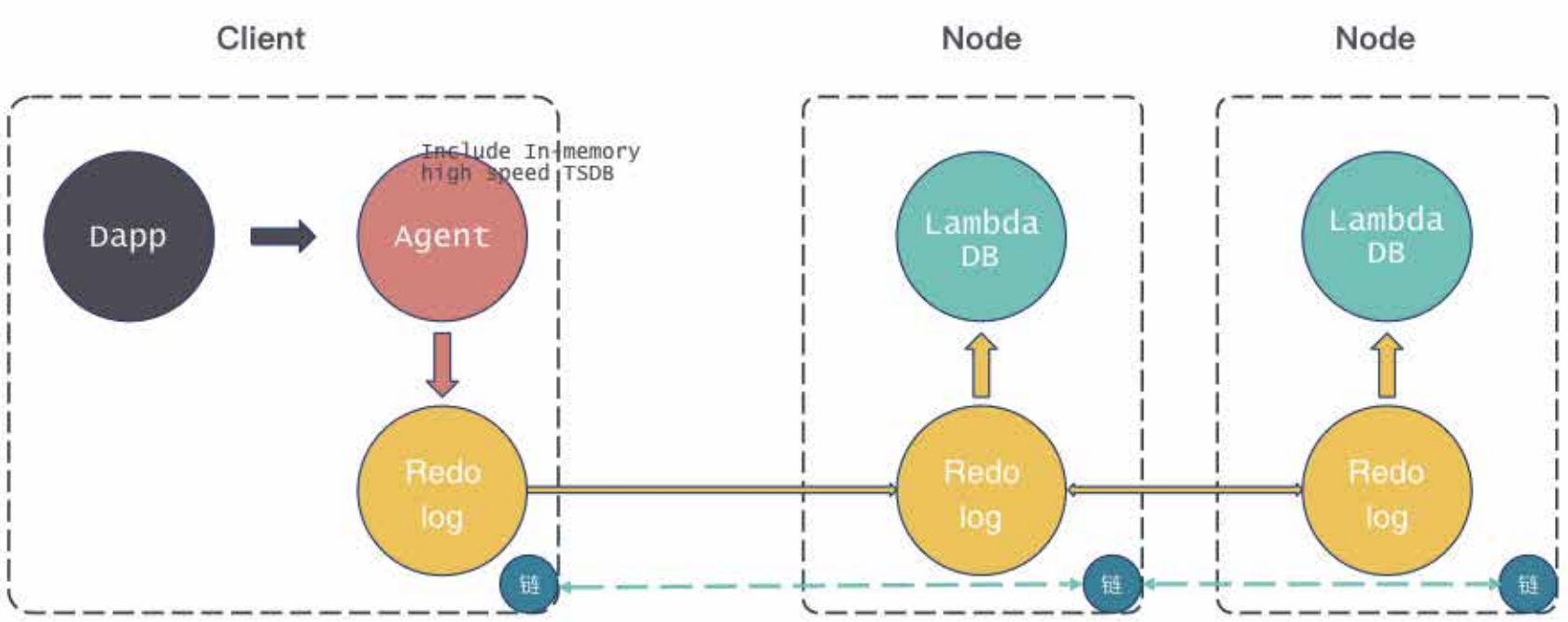
1/Lambda DB总体架构



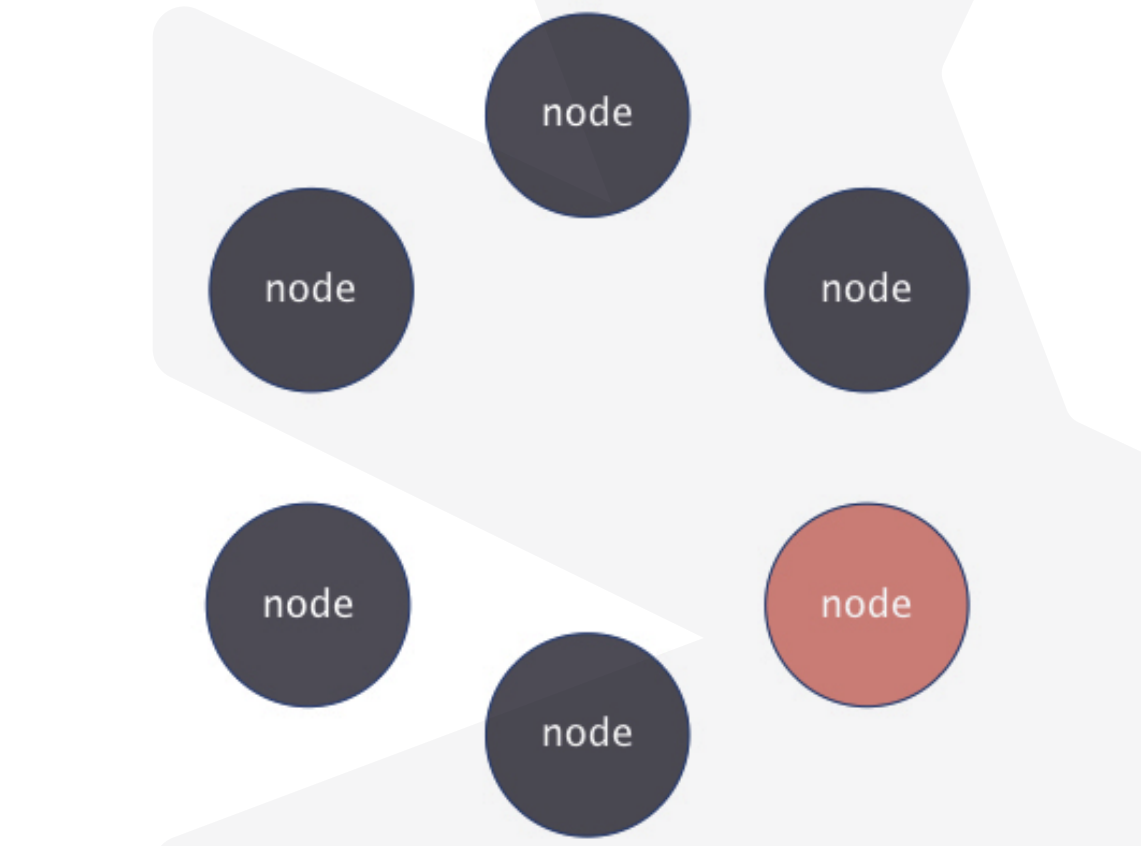
Lambda　DB是一个分布式对等网络节点系统，每一个节点从逻辑上都含有一个ShardChain(WorkChain2)的账本和一个数据库系统的运行时系统。数据库节点之间按照一定规则组成小网络集群，集群的种子路由信息存在链上，路由表本身在存在P2P系统之上。客户端在使用系统的时候，通过区块链系统完成缴费，并获得数据库系统的访问权限。由此，整个MainChain链本身是一个低频交易系统，WorkChain1是一个持续计费的高TPS系统，数据库是一个兼顾高频和低频的数据访问系统，数据库本身通过WorkChain2对数据的响应进行统计，并在检查点的时候对请求和响应信息进行对账。从思路来说，链提供的是类似于公有云的计费系统，并提供了跨越多个物理IDC的信息连接能力。

在使用场景下，Lambda　Client端都有Agent部署，在Agent中内置实现了Facebook的高性能时序数据存储引擎Beringei，10分钟的数据默认存放在本地，提供急速访问能力。Beringei是Facebook之前开源的Gorilla系统更为高效的实现，其设计目的是追求极致的速度，并提高读取效率，其时间戳压缩算法使用了delta-of-delta编码算法，数据值采用XOR进行压缩，存储容量压缩了近10倍。Beringei将数据放置于内存中，与基于HBase的传统数据库存储时间序列数据方式相比，查询延时缩短了73倍，吞吐量提高了14倍。

10分钟后的数据通过RPC传输到Lambda 数据库集群进行存储。



在 Lambda DB中没有中心化的管理节点，只有数据存储节点。所有的存储节点都有着完全相同的职责，会对外界提供同样的服务，所以在整个系统中永远不会出现单点故障的问题。去中心化的架构使得系统的水平扩展非常容易，节点可以在任何时候直接加入到整个 Lambda DB的集群中，并且只会造成集群中少量数据的迁移。



2/虚拟节点、分片和复制

Lambda DB的核心理念中有增量扩展（Incremental Scalability）的原则，这就需要一种能够在一组节点中动态分片的机制，这种机制又称为DHT(distributed hash table)，通过DHT这种策略 ，Lambda TSDB能够将负载合理的分配到不同的存储节点上。Lambda DB的分片策略依赖于一致性哈希(Consistent hashing)，并且针对于节点的异构性做了进一步的调整，我们使用的算法称为CCHDP算法，其主要步骤为：

(1) 首先采用聚类算法对设备集合进行分类,使得每个类中设备的权重差异在预设的范围内;

(2) 聚类完成后,类间的布局机制按照类的权重将[0,1]区间划分为多个子区间,为每个类分配一个子区间,将落入某个子区间的数据分配给相应的类;

(3) 每个类的内部布局机制使用一致 hash 方法进行数据的再次分配,将数据布局到具体的设备上。

CCHDP算法

CCHDP

定义1(公平性). 设存储系统的设备集合$D_0=\{d_1,\ldots,d_n\}$,设备$d_i\in D_0$的相对权重为w_i,数据集合$X_0=\{x_1,\ldots,x_m\}$,函数$f_0:X_0\to D_0$将数据集合X_0映射到设备集合$D_0=\{\}$,$\forall x\in X_0$,布局算法A将数据x分配给$d_i\in D_0$的概率为$p(A(x,f_0)=d_i)$.$\forall\varepsilon>0$,若$|p(A(x,f_0)=d_i)-w_i|<\varepsilon$,则算法$A$是公平的

算法的公平性使得数据$x\in X_0$分配给设备$d_i\in D_0$的概率无限接近d_i的相对权重,保证数据公平地分配给每个设备.

定义2(自适应性). 设存储系统的设备集合$D_0=\{d_1,\ldots,d_n\}$,设备$d_i\in D_0$的相对权重为w_i,数据集合$X_0=\{x_1,\ldots,x_m\}$,函数$f_0:X_0\to D_0$将数据集合X_0映射到设备集合D_0.

若布局算法A满足以下两个条件,则布局算法A是自适应的:

(1) 当前设备集合D_0变为$D_0^+=\{d_1,\ldots,d_{n+1}\}$,设函数$f_0^+:X_0\to D_0^+$将数据集合$X_0$映射到当前设备集合$D_0^+$.$\forall x\in X_0$,若$A(x,f_0^+)=f_0^+(x)\in D_0$,则$A(x,f_0^+)=A(x,f_0)$;

(2) 当前设备集合D_0变为$D_0^-=\{d_1,\ldots,d_{i-1},d_{i+1},\ldots,d_n\}$,设函数$f_0^-:X_0\to D_0^-$将数据集合$X_0$映射到当前设备集合$D_0^-$.$\forall x\in X_0$,若$A(x,f_0^-)=f_0^-(x)\in D_0^-$,则$A(x,f_0^-)=A(x,f_0)$.

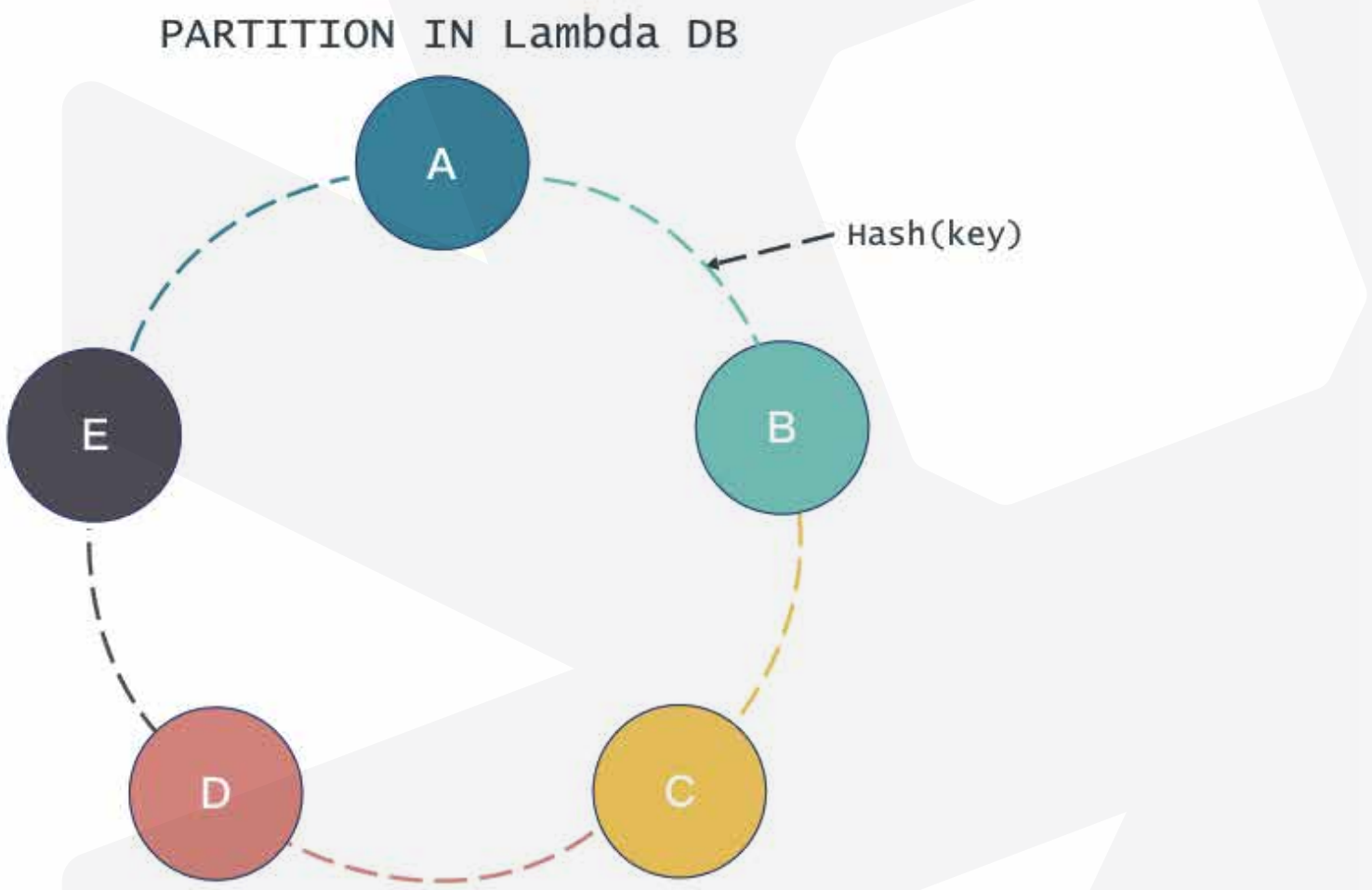
定义3(设备之间的距离). 设备集合D_0中的两个元素d_i和d_j的权重分别为w_i和w_j,则d_i和d_j的距离为$w_{ij}=|w_i-w_j|$.

定义4(设备到类的距离). 设设备类为S,其聚类中心为d_i,其中$d_i\in S$,则设备d_j到类S的距离等于设备d_j到d_i的距离

聚类算法的目标是使得每个类中的设备到其聚类中心的距离小于一个预设的值.按照定义4 中的距离公式计算设备到聚类中心的距离.设类内距离阈值为T,将设备到聚类中心的距离与T进行比较,判断设备所属的类或者将设备作为一个新类的中心.首先取任意的一个设备d_i作为第1 个类S_1的聚类中心,如取d_1作为类S_1的中心, 根据定义4 计算下一个设备d_2到D_1的距离.如果该值小于等于T,则将d_2归到类S_1中;否则,将d_2作为新类S_2的中心.设已有k个聚类中心,计算未分类的设备d_i到k个聚类中心的最小距离,如果该值大于T,则将设备d_i作为新类S_{k+1}的中心;否则,将设备分配给k类中距离其最近的类.重复上述过程,直到将所有的设备划分到各个类中.

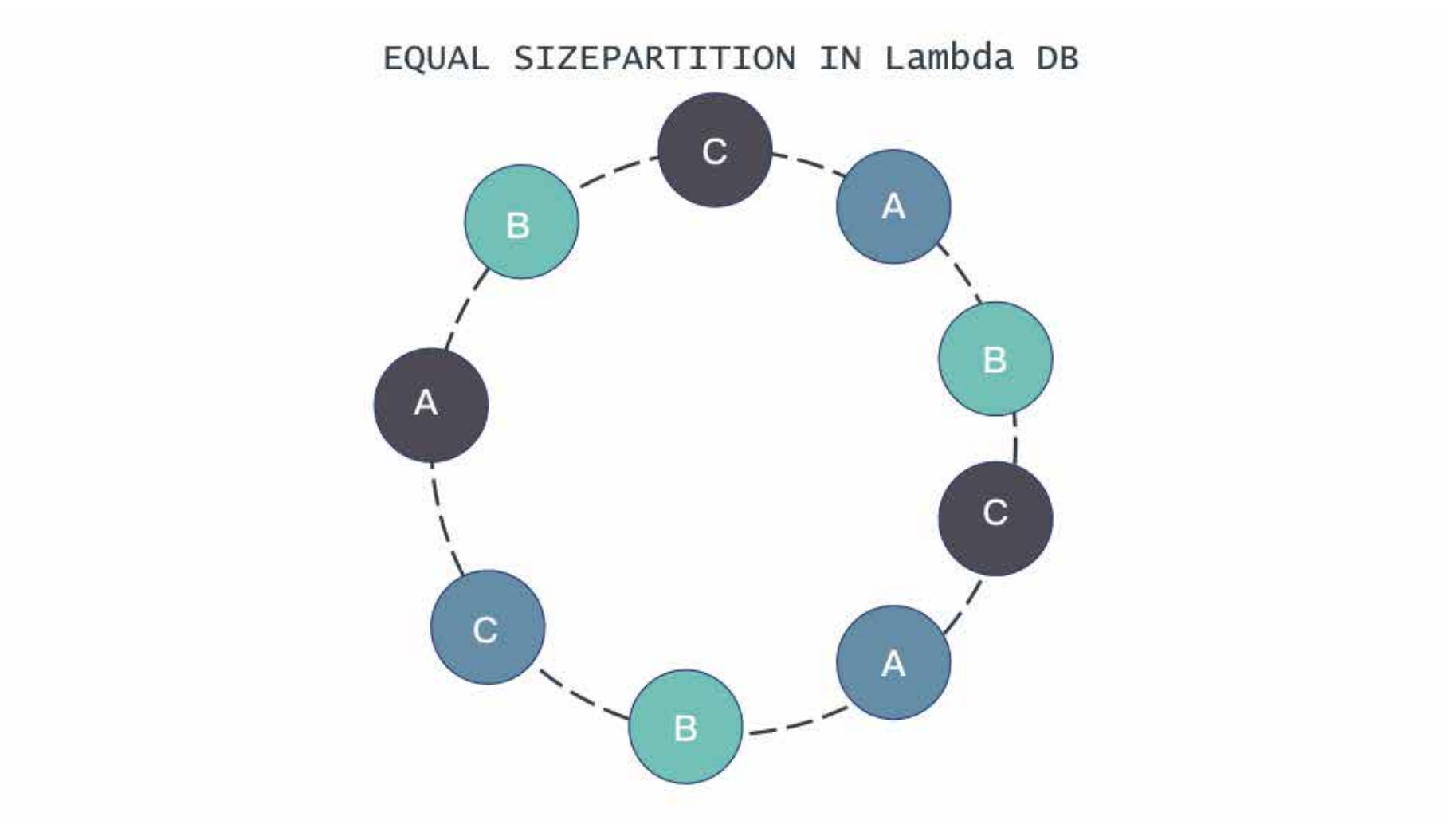
聚类算法将设备集合$D_0=\{d_1,\ldots,d_n\}$划分为g个类的集合$C=\{D_1,\ldots,D_g\}$.设类D_j为,其中$1\le j\le g$,则.设类D { ; 1,2,..., } $_{jijdi}$ n= $_{1gjjnn=}$ $\sum_j$ 的权重为w_j,则 $_{1jnjii}$ $w_{=}=\sum$,g个类总的权重$_{1}$ $1_{gjiw=}=\sum$.问题转化为在异构的类集合C中 w 分布数据.

可以将上述算法简单理解为，所有的键(Key)在存储之前都会通过哈希函数得到一个唯一的值， Lambda DB构造了一个特殊的逻辑结构，所有组成单元连接在一起形成一个固定长度的环，环的的最大值单元和最小值单元连到一起。



每一个节点都会被 Lambda DB在这个环中分配一个随机的位置，而这个节点会处理从哈希的输出在当前节点前的所有键。假设有一个键值对 (key, value)，Hash(key) 的结果位于上图中的绿色区域，从环中的位置开始按照顺时针的顺序寻找，找到的第一个节点,也就是节点 B 就会成为协调者（coordinator）。协调者负责处理当前的键值对，上图中的每一个节点都会负责与其颜色相同的部分。

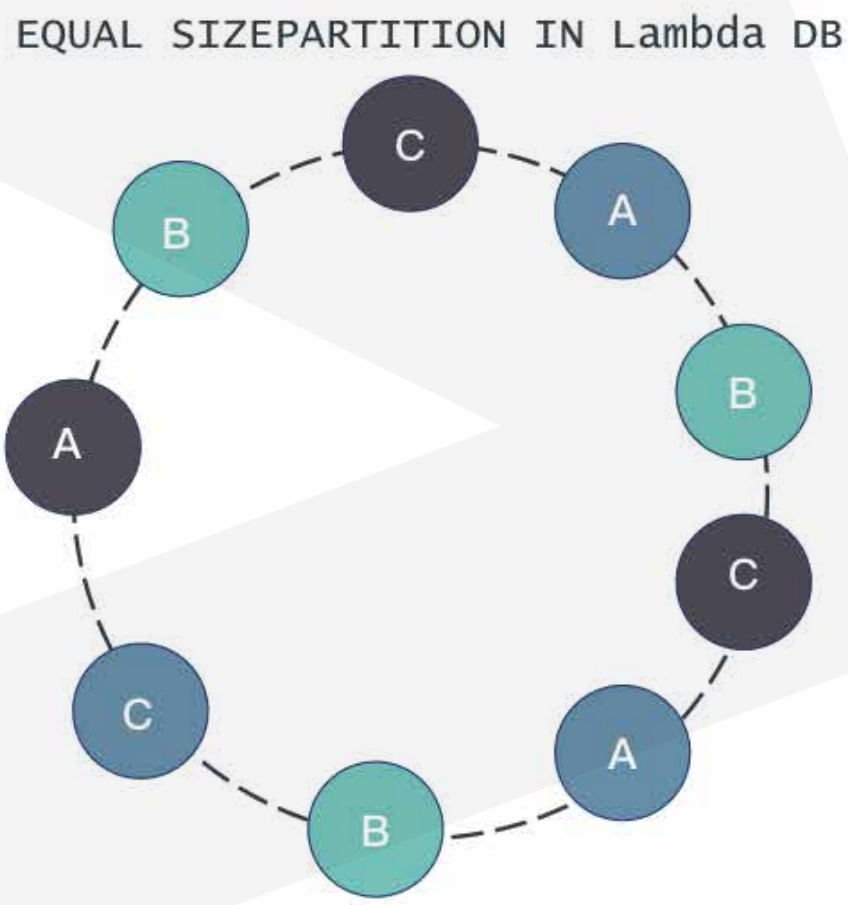
Lambda DB系统中的每一个节点在刚刚加入集群时，会被分配一个随机的位置，由于算法的随机性可能会导致不同节点处理的范围有所不同，最终每一个节点的负载也并不相同。为了解决这个问题，Lambda DB使用了一致性哈希算法的变种，将同一个物理节点分配到环中的多个位置，成为多个虚拟节点。



每一个节点都会被 Lambda DB在这个环中分配一个随机的位置，而这个节点会处理从哈希的输出在当前节点前的所有键。假设有一个键值对 (key, value)，Hash(key) 的结果位于上图中的绿色区域，从环中的位置开始按照顺时针的顺序寻找，找到的第一个节点,也就是节点 B 就会成为协调者（coordinator）。协调者负责处理当前的键值对，上图中的每一个节点都会负责与其颜色相同的部分。

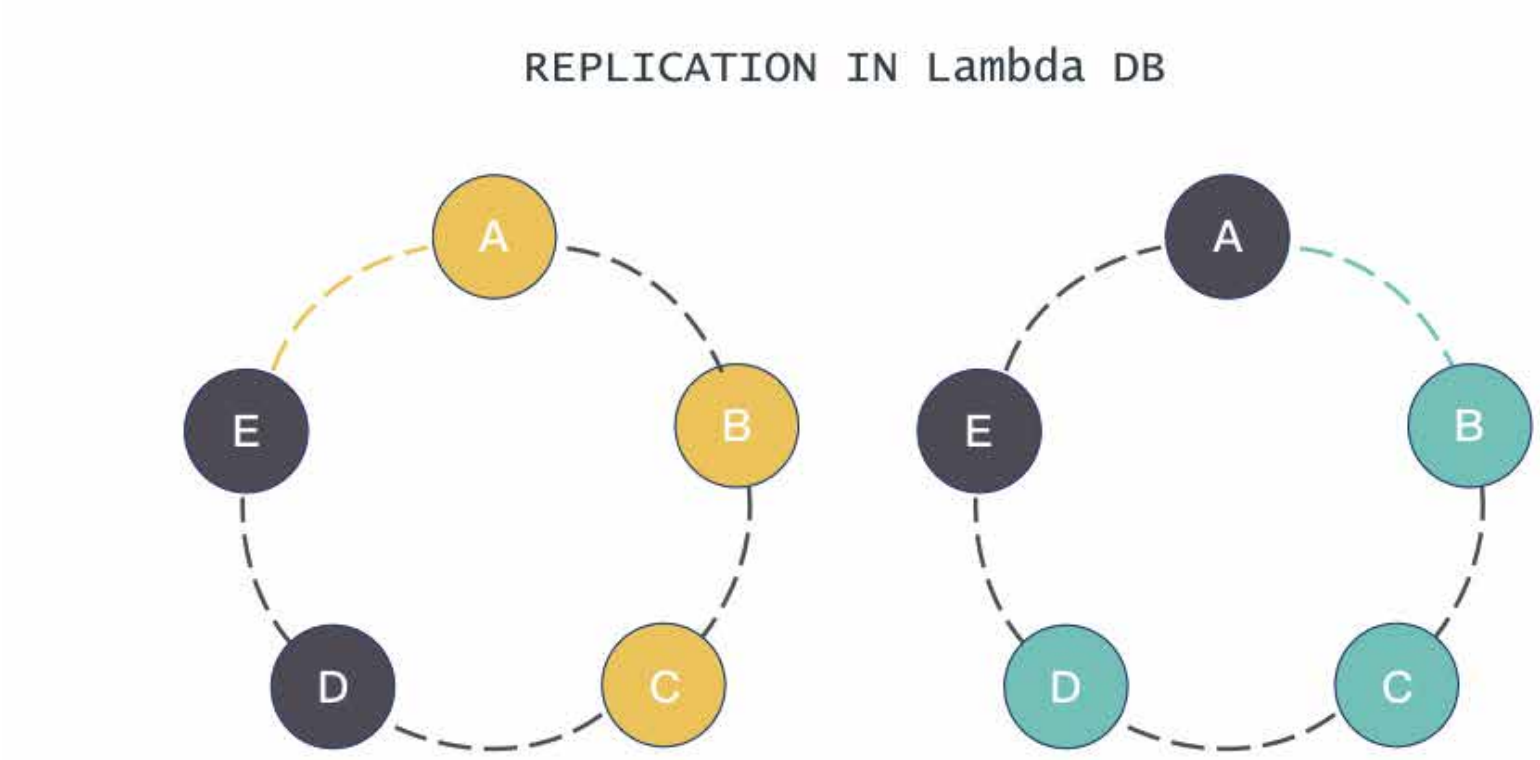
Lambda

Lambda DB系统中的每一个节点在刚刚加入集群时，会被分配一个随机的位置，由于算法的随机性可能会导致不同节点处理的范围有所不同，最终每一个节点的负载也并不相同。为了解决这个问题，Lambda DB使用了一致性哈希算法的变种，将同一个物理节点分配到环中的多个位置，成为多个虚拟节点。



为了解决负载的不均衡的问题，除了上面使用虚拟节点的策略之外，Lambda DB还提供了另外一种策略，可以将数据的哈希分成 m 个大小相等的区域，n个节点每一个处理 m/n个分区，当某一个节点因为故障或者其他原因需要退出集群时，会将它处理的数据分片随机分配给其它的节点，当有节点加入系统时，会从其它的节点中『接管』对应的数据分片。

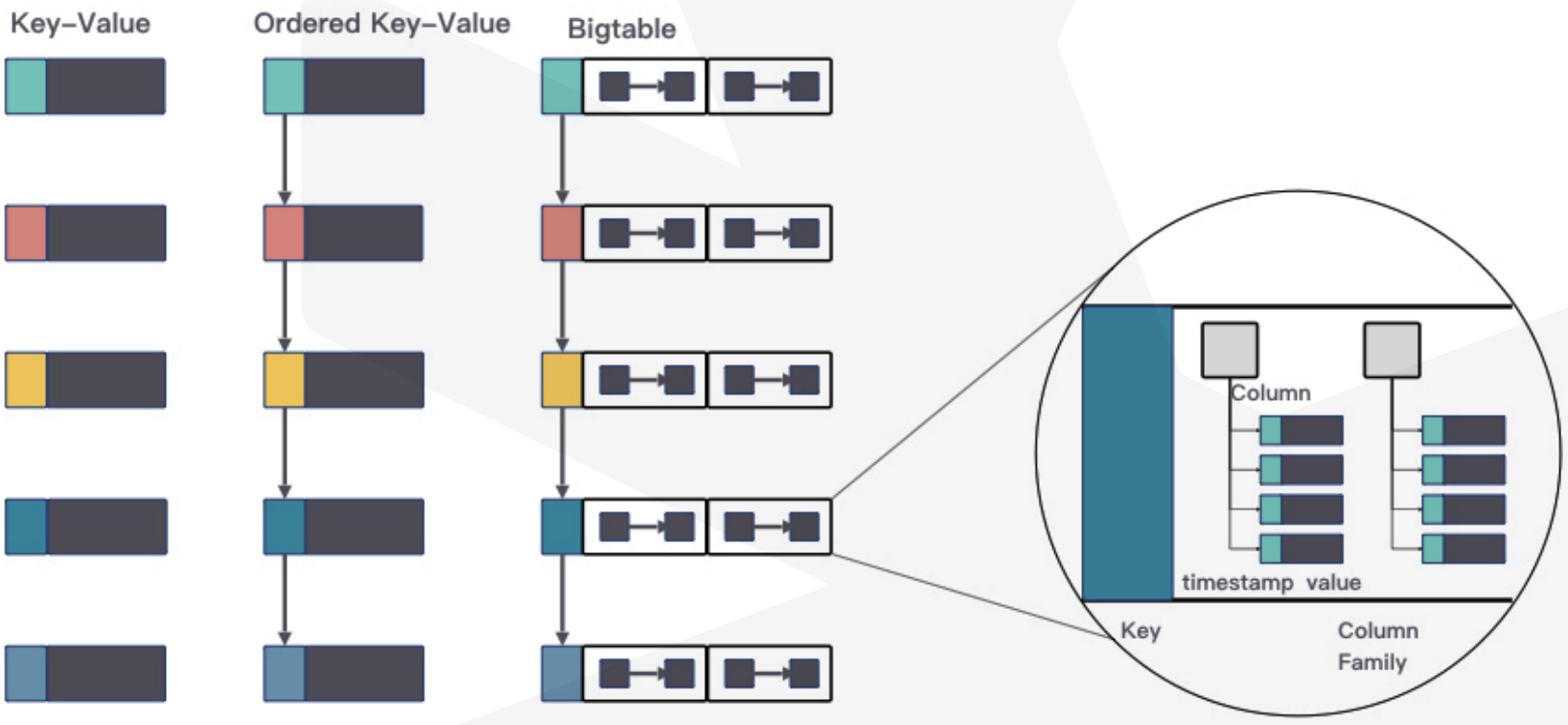
Lambda DB为了达到高可用性和持久性，防止由于节点宕机故障导致数据丢失，将同一份数据在协调者和随后的 N–1 个节点上进行了备份，N 是一个可以配置的值，在一般情况下都为 3。



也就是说，上图中黄色区域的值会存储在三个节点 A、B 和 C 中，绿色的区域会被 B、C、D 三个节点处理，从另一个角度来看，A 节点会处理范围在 (C, A] 之间的值，而 B 节点会处理从 (D, B] 区域内的值。

Lambda DB系统中，负责存储某一个特定键值对的节点列表叫做偏好列表（preference list），因为虚拟节点在环中会随机存在，为了保证出现节点故障时不会影响可用性和持久性，Lambda DB的偏好列表中的全部节点都是不同的物理节点。在Lambda项目中，偏好列表存放在链上,由此可以简单理解为，路由信息存放于链上。

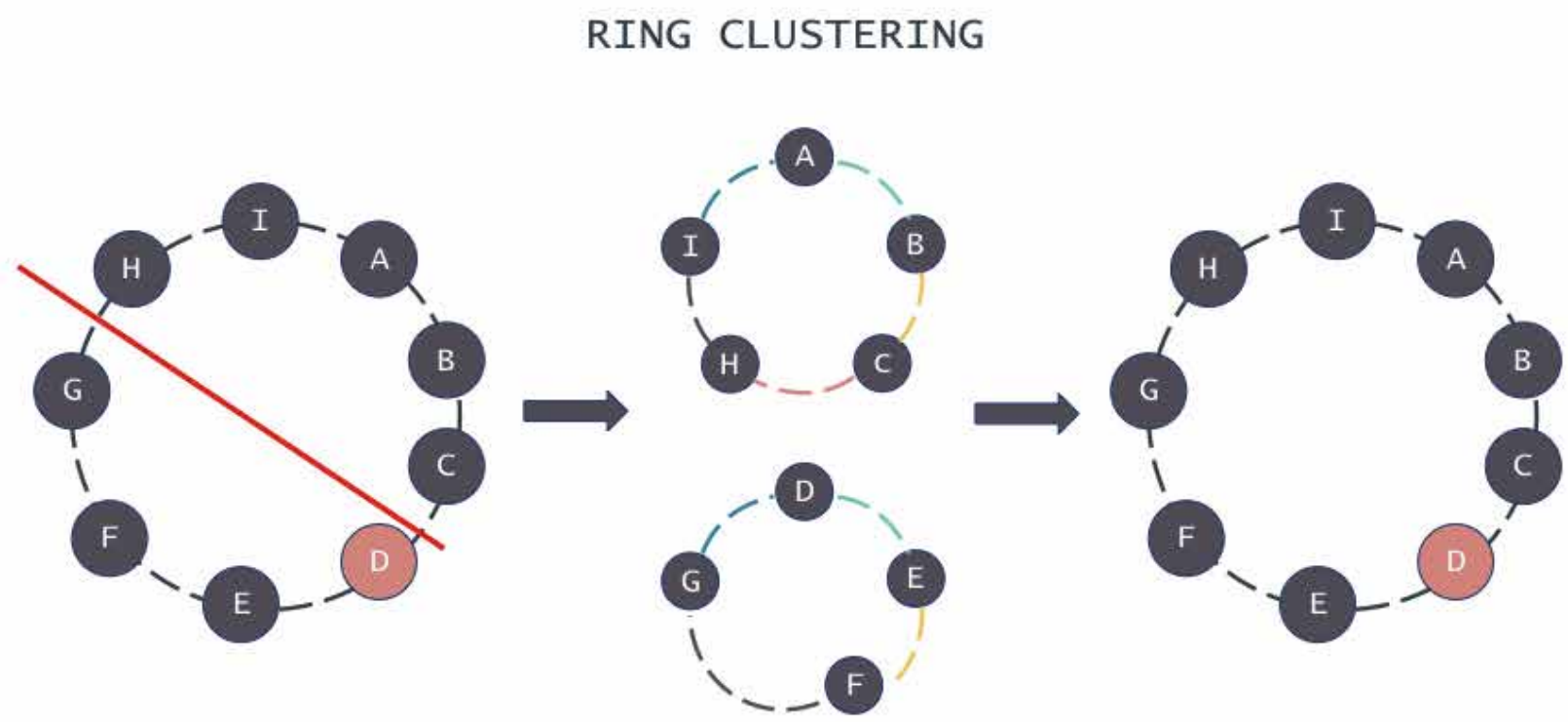
在Key-value模型下，不同的处理可以存储不同类型的数据，我们支持如下方式的存储方式：



3/集群和分区

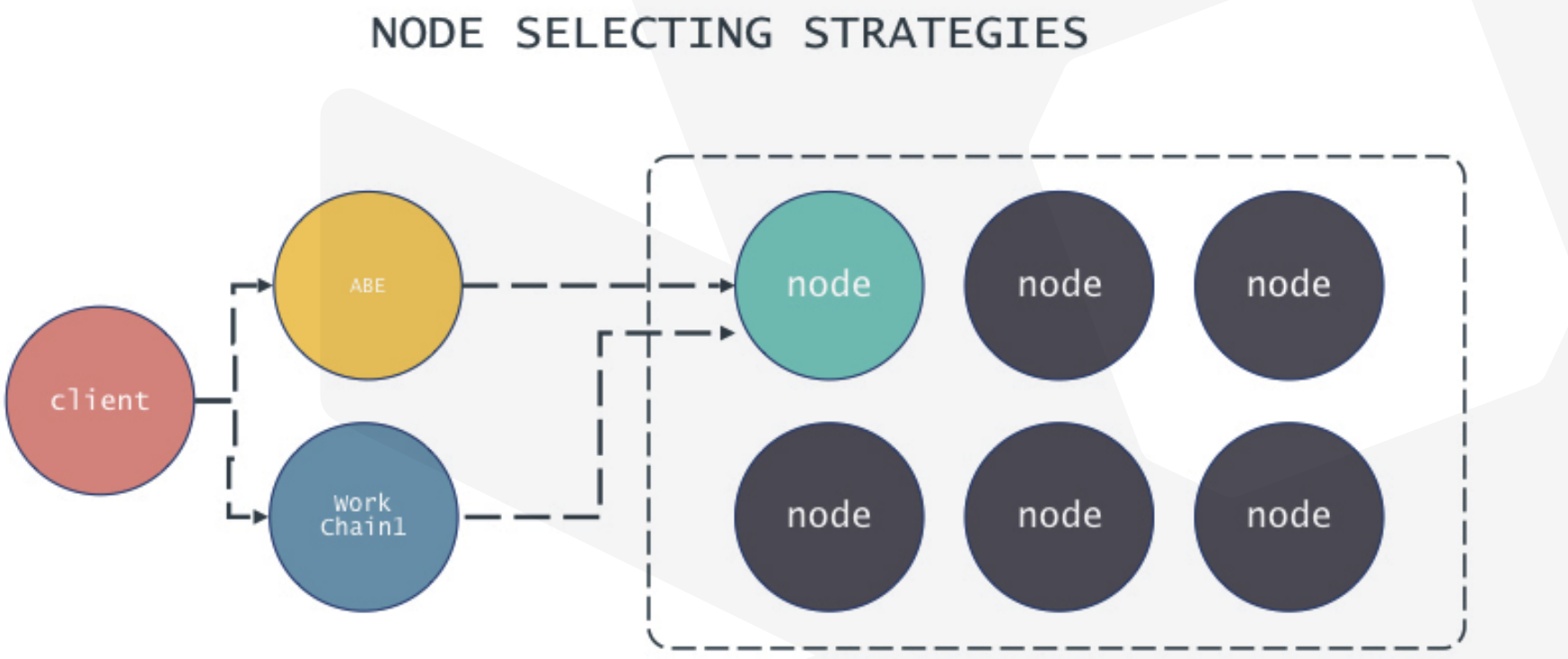
在Lambda DB中，还提供了集群和分区的功能。Lambda DB是一个云端唯一的数据库系统，由于在这个分布式数据库系统中，数据文件要向全网广播，过大的单个集群在极端情况下有可能导致网络消息的传播和拥堵，因此Lambda DB提供了基于地理位置的集群和分片能力。不论节点是位于同一机房内，还是不同机房间，网络故障都可能导致虚拟环的分区。分区后，Lambda DB的节点加入退出机制导致每个分区的节点形成新的虚拟环。新虚拟环的形成过程完全是自动的，无需运维人员参与。

假设有一个虚拟环，其部分节点在欧洲数据中心，部分在亚洲数据中心。如果两个数据中心之间的网络连接中断，Lambda DB会在每个数据中心形成新的虚拟环。过了一段时间后，欧洲数据中心和亚洲数据中心的通信恢复，这个时候两个虚拟环需要合并。我们可以指定一些特殊的地标节点。无地标节点的环中的所有节点首先离开它们所在的环，再重新加入到地标节点所在环。一段时间后，整个系统又只剩下唯一的一个虚拟环。

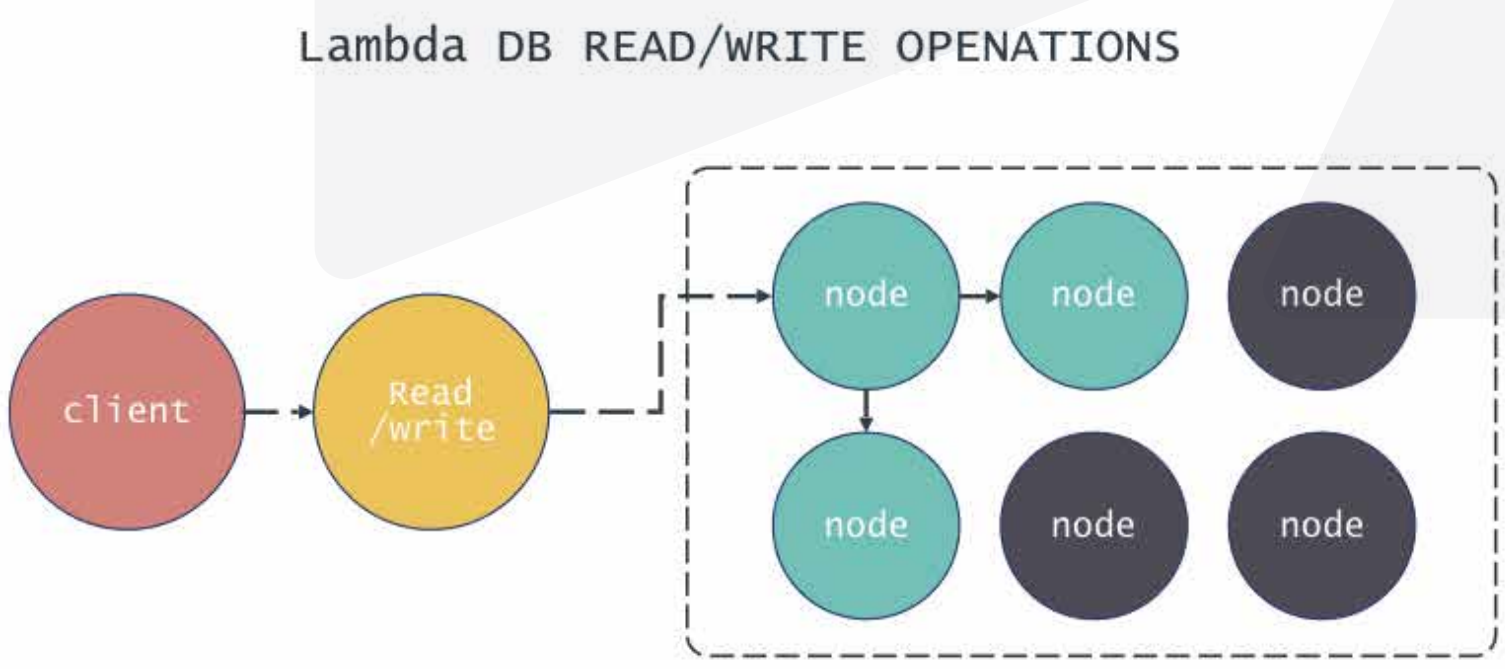


4/数据的读写

Lambda DB的一个指定集群中的任意节点都能够接受来自客户端的对于任意Key的读写请求，所有的请求都通过 RPC 调用执行，客户端在选择节点时，通过WorkChain3获得访问所需要的授权，并且通过WorkChain1进行通信的路由转发。



处理读写请求的节点叫做协调者（coordinator），前 N 个『健康』的节点会参与读写请求的处理。Lambda DB使用Quorum 一致性协议来保证系统中的一致性，协议中有两个可以配置的值：R 和 W，其中 R 是成功参与一个读请求的最小节点数，而 W 是成功参与写请求的最小节点数。

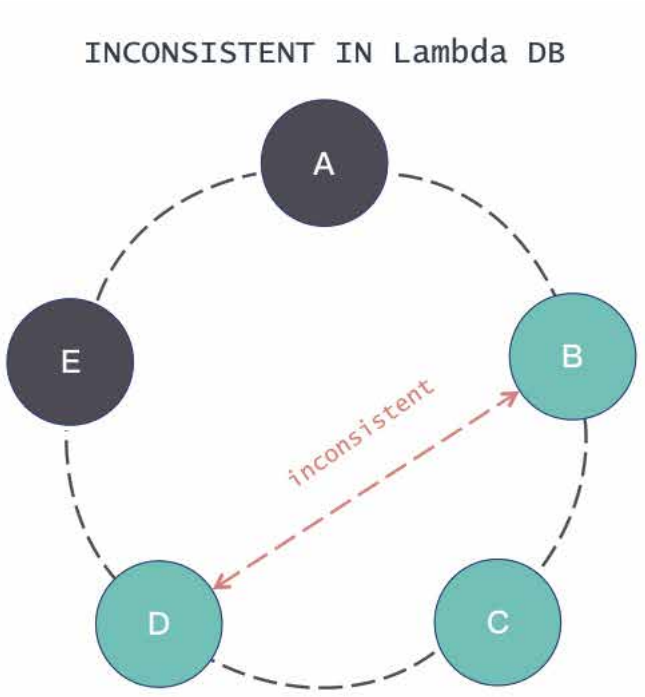


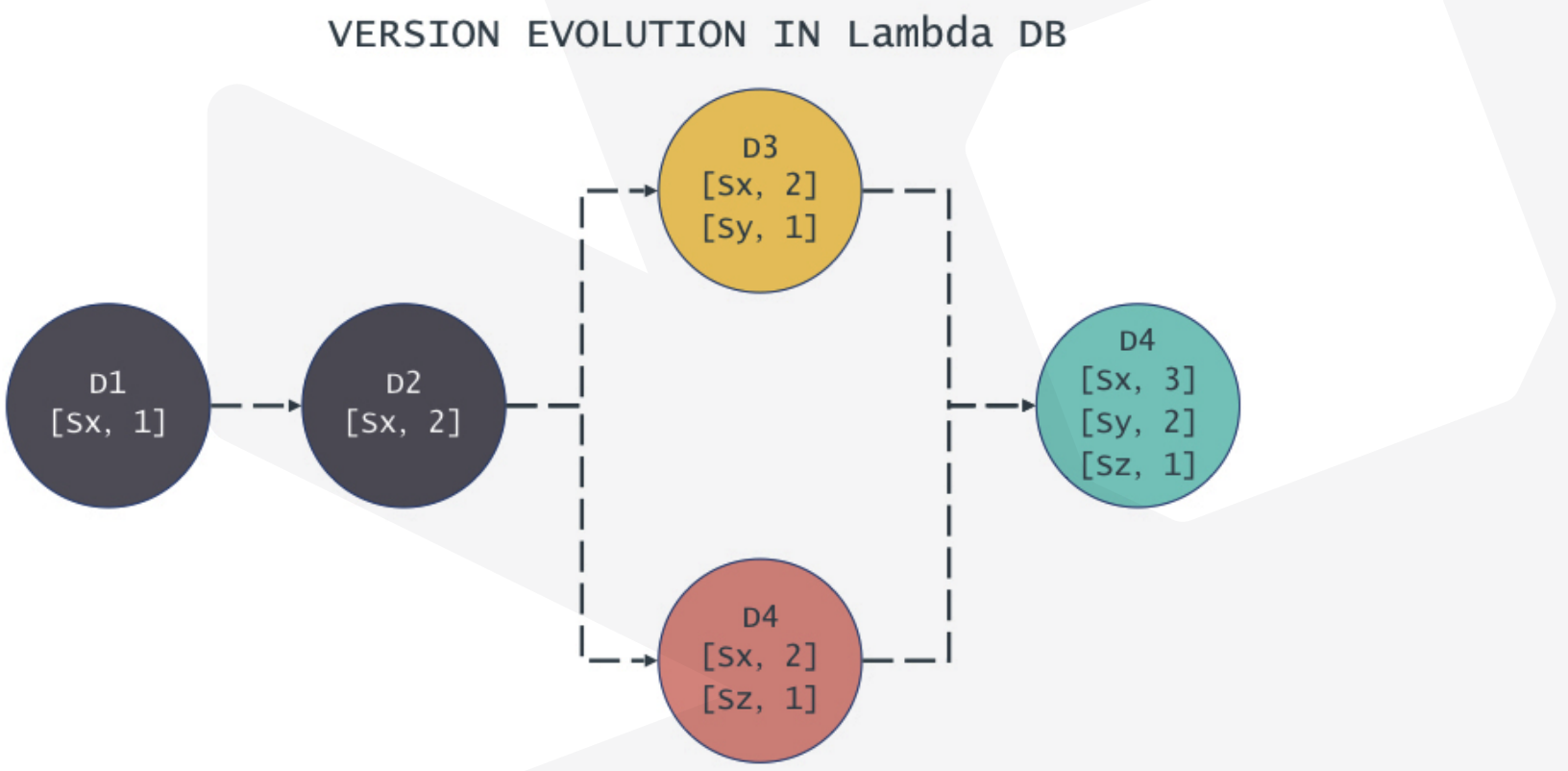
当 R = 2 时，所有的读请求必须等待两个节点成功返回对应键的结果，读请求的时间取决于返回最慢的节点。对于写请求来说也是完全相同的；当协调者接收到了来自客户端的写请求 put() 时，它会创建一个新的向量时钟（vector clock），然后将新版本的信息存储在本地，之后向偏好列表（preference list）中的前 N–1 个节点发送消息，直到其中的 W–1 个返回这次请求才成功结束，读请求 get() 与上述请求的唯一区别就是，如果协调者发现节点中的数据出现了冲突，就会对冲突尝试进行解决并将结果重新写回对应的节点。

R + W > N的情况下，Lambda DB保证读操作总能读到最新的数据，在没有节点故障的情况下，可以近似理解强一致性。

5/数据冲突和向量时钟

Lambda DB系统提供最终一致性，最终一致性能够允许我们异步的更新集群中的节点。



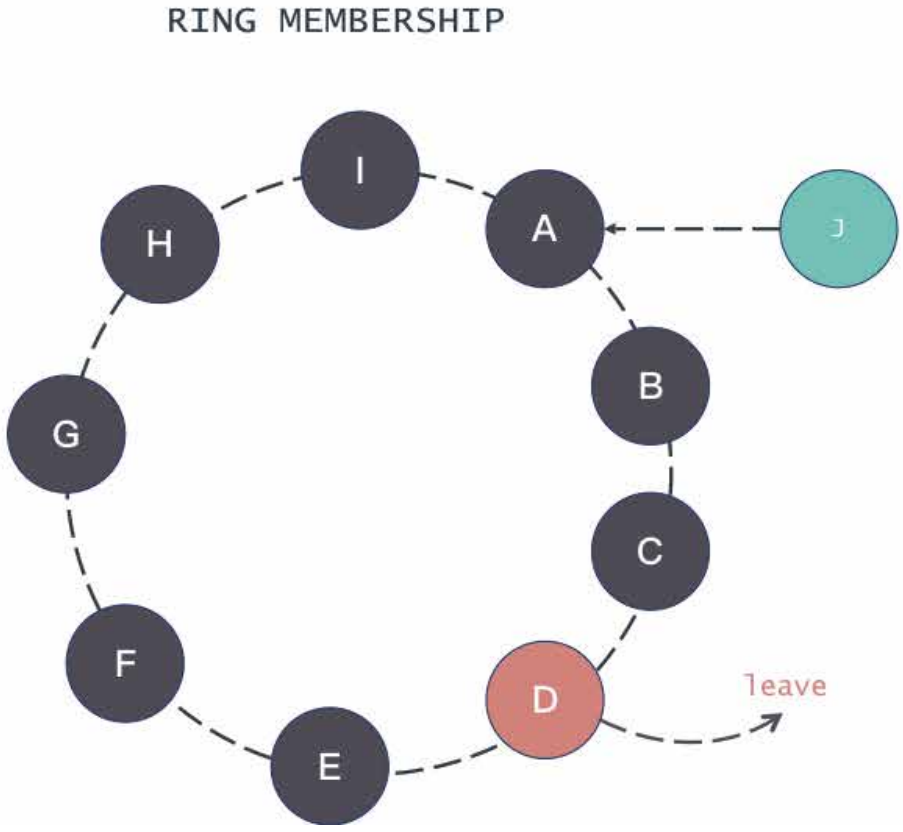


上图中的每一个对象的版本 Dx 中存储着一个或多个向量时钟 [Sn, N]，每次 Lambda DB对数据进行写入时都会更新向量时钟的版本号，节点 Sx 第一次写入时向量时钟为 [Sx, 1]，第二次为 [Sx, 2]，在这时假设节点 Sy 和 Sz 都不知道 Sx 已经对节点进行了写入，它们接收到了来自其他客户端的请求，在本地也对同样的Key做出了写入并分别生成了不同的时钟 [Sy, 1] 和 [Sz, 1]，当客户端再次使用 get() 请求时就会发现数据出现了冲突，这个时候，可以选择last write wins，这依赖于节点时钟的同步。

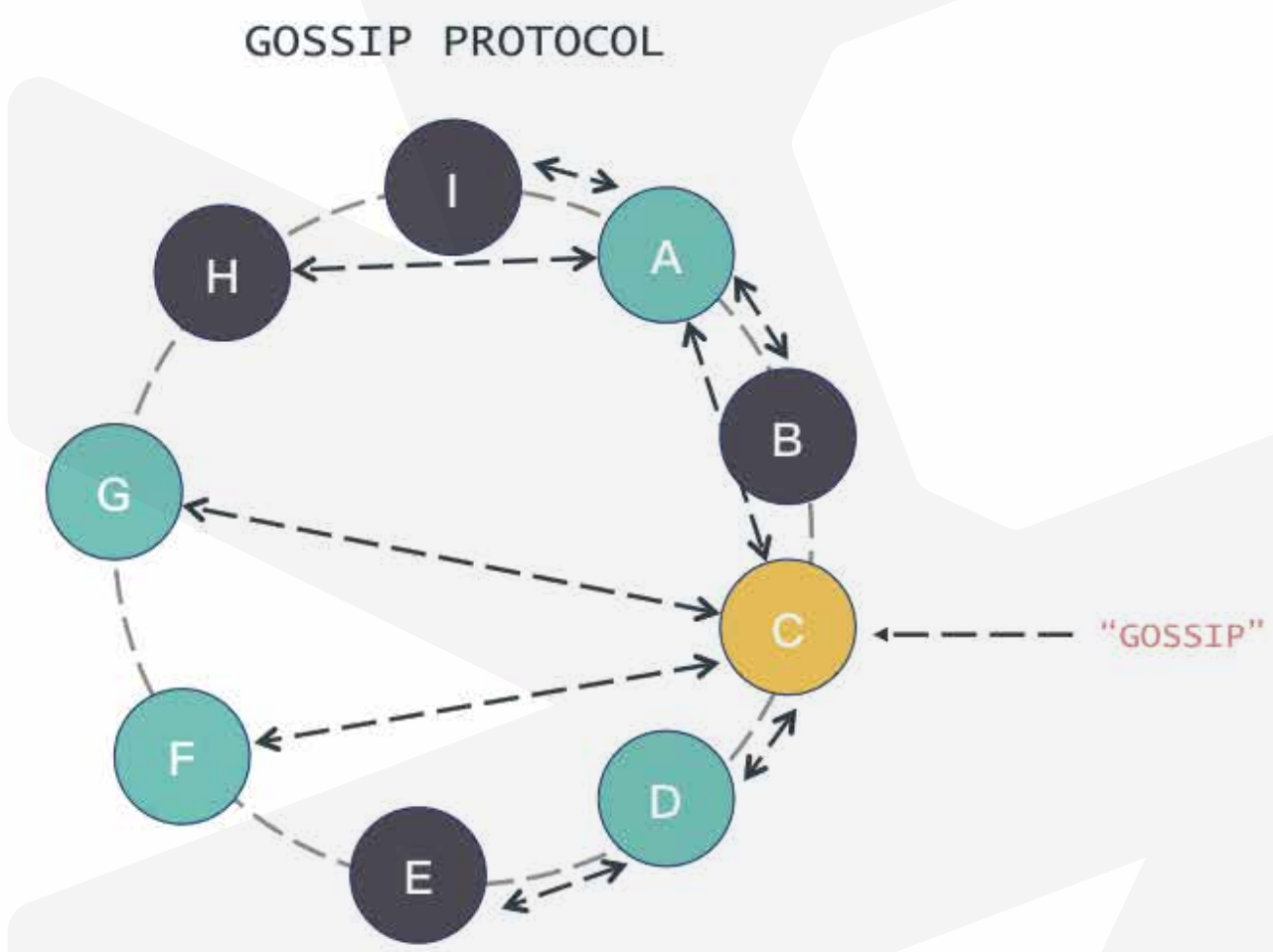
6/节点的增加和删除

因为在分布式系统中节点的失效是非常常见的事情，而节点也很少会因为某些原因永久失效，往往大部分节点会临时宕机然后快速重新加入系统；由于这些原因，Lambda DB选择使用了质押和共识的机制向系统中添加和移除节点，并将变更之后的信息更新到链上。机制是：

- 节点加入时，存储节点所对应的链地址向链上发起一笔质押交易，以交易时间作为节点的上线时间。
- 节点离线时，由节点所在集群的验证人达成共识，以共识时间为节点离线时间。
- 节点的单位存储奖励是节点所在存储池的平均奖励，但要达到最低上线时长
- 存储节点通过定期猜测BFT的出块节点领取在线奖励

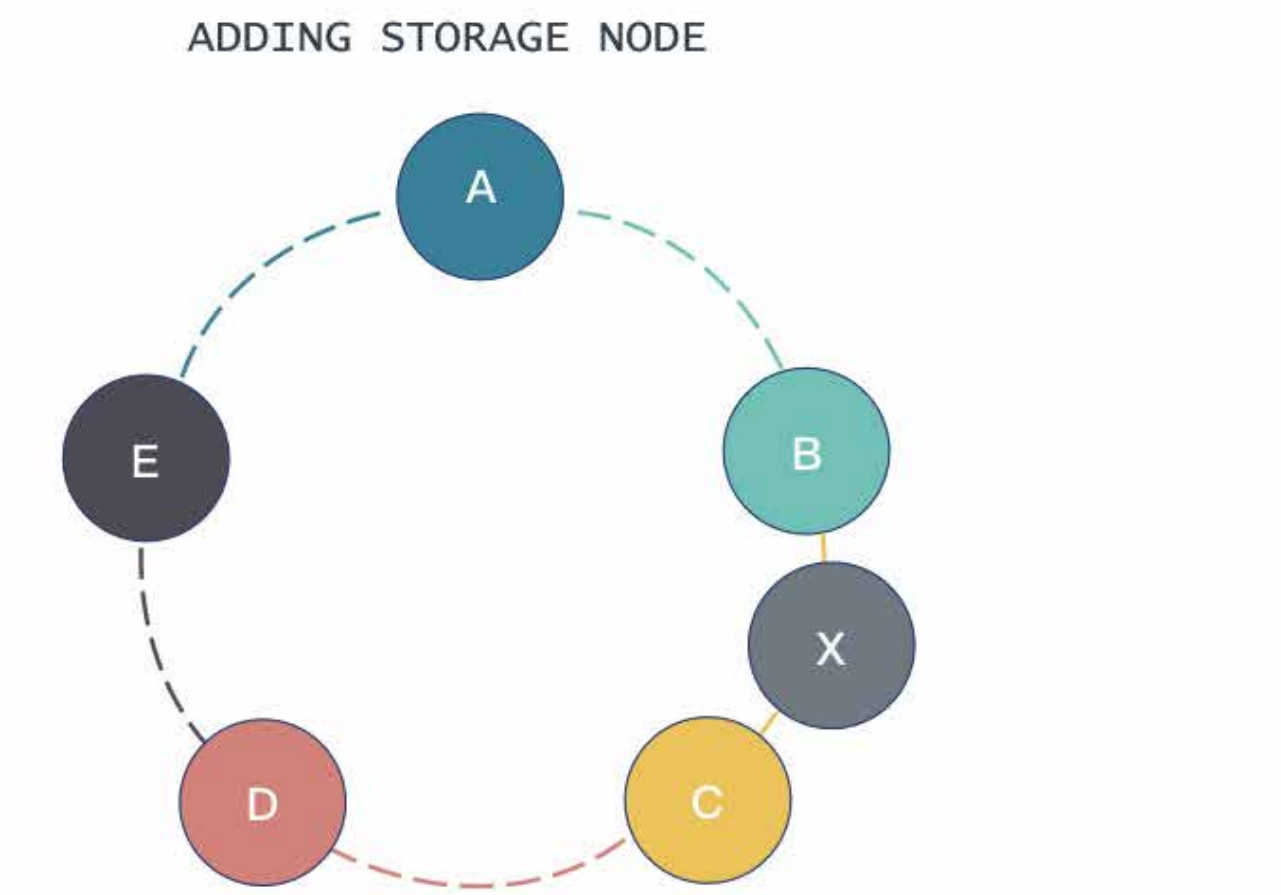


添加节点时可以使用命令行工具或者浏览器连接 Lambda DB中的任意节点后触发一个成员变动的事件，这个事件会从当前的环中移除或者向环中添加一个新的节点，当节点的信息发生改变时，该节点会通过 Gossip 协议通知它所能通知的最多的节点。



在 Gossip 协议中，每次通讯的两个节点会对当前系统中的节点信息达成一致。通过节点之间互相传递成员信息，最终整个 Lambda DB的集群中所有的节点都会就成员信息达成一致。如上图所示，”gossip”首先会被 C 节点接收，然后它会传递给它能接触到的最多的节点 A、D、F、G 四个节点，然后“gossip”会进行二次传播传递给系统中的灰色节点，到此为止系统中的所有节点都得到了最新的“gossip”消息。

当我们向 Lambda DB中加入了新的节点时，会发生节点之间的分片转移，加入现在X节点连接上了 Lambda DB数据库，该节点被分配到了如下图所示的 A 和 B 节点之后。

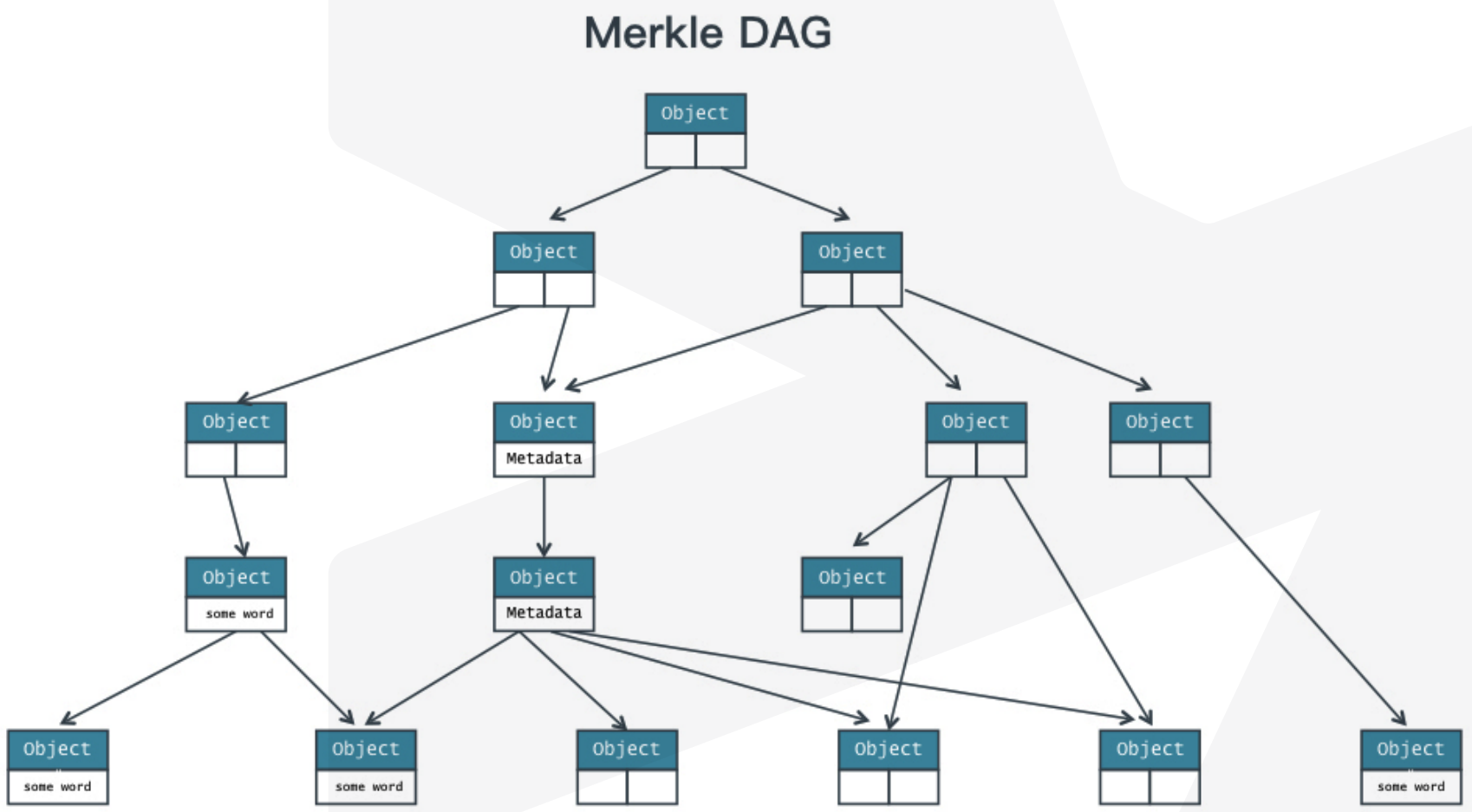


新引入的节点 X 会从三个节点 C、D、E 中接受它们管理的分片的一部分，也就是上图中彩色的 (E, A]、(A, B] 和 (B, X] 三个部分，在 X 节点加入集群之前分别属于与其颜色相同的节点管理

7/副本同步

在 Lambda DB运行的过程中，由于一些情况会造成不同节点中的数据不一致的问题，Lambda DB使用反熵（anti-entropy）的策略保证所有的副本存储的信息都是同步的。为了快速确认多个副本之间的数据的一致性并避免大量的数据传输，

Lambda DB使用了 Merkle DAG对不同节点中的数据进行快速验证与传输，同时DAG可轻易去除重复数据节省存储空间，DAG灵活的数据存储方法可广泛用于Lambda系统中寻址、FS block交换、KV数据交换、文档等富媒体交换。



8/数据的Top-k查询实现及原理

在时间序列数据的处理场景下，其查询主要是对一系列时间范围内的数据的最大值、最小值等进行查询，这属于范围查询，也叫作Top K查询。在DHT中，通常来说，支持复杂的范围查询是非常困难的，这已经是分布式系统科研前沿的主题，在我们的时间序列数据库中，我们实现了对范围查询、连结查询和前K名查询的支持，这也是Lambda^2 TSDB具有价值的特点。限于篇幅，我们只描述了对于TOP k查询的实现原理。在集中式和分布式系统中，关于TOP k查询的一个有效实现是阈值算法TA【nepal and Ramakrishna,1999】，我们实现了TA算法在在DHT情况下的一个变化算法，叫做DHTop算法，该算法来自论文APPA【Akbarinia et al.,2007c】。

在Lambda DB中，节点使用两种互相补充的方式将他们的元组存储在DHT中：元组存储和属性值存储。使用元组存储，每个元组可以使用它的标识符(比如主键)来作为存储键存储在DHT中。这使得它可以通过类似主索引的标识符来查找一个元组。属性值存储单独将属性存在DHT中，该属性可能出现在一个查询的相等谓词或在一个查询打分函数中。因此，类似在次级索引中那样，它允许使用属性值来查找元组。属性值存储有两个重要的特性：

[1] 在从DHT接收到一个属性之后，节点可以轻松获得属性值对应的元组

[2] 相对接近的属性值被存储在相同的节点上

为了满足第一个特性，那些用来存储整个元组的键，将同属性值存储在一起。第二个特性通过如下所述的域划分概念来实现。考虑一个属性a，令Da是它值的域。假定在Da上有一个全序<，Da被划分为n个非空的子域d1,d2…，dn，使得它们的并集等于Da，任意两个不同子域的交集为空，且对于每一个v1 di和v2 dj,如果i<j便有v1<v2.哈希函数被应用在属性值的子域上。因此，对落在相同子域的属性值，存储的键是相同的，并且它们都存储在相同的节点中。为了避免属性存储的偏斜，域划分是通过把属性值均匀分布在子域中的方式来完成的。这种技术使用了描述属性值分布的、基于直方图的信息。

使用这种存储模型，前k名查询处理算法，被称为DHTop，按照如下方式工作：令Q是一个给定的前k名查询，f是它的打分函数，且p0是发出查询Q的节点。令打分属性是那些作为参数准备传递给打分函数的属性的集合。DHTop在p0开始，并以两个阶段处理：首先它准备候选子域的顺序列表，接着它持续地获取候选集属性值和它们的元组，直到它找到前k个元组。两个步骤的细节如下：

算法：DHTop
Input:*Q*:top-k query; *f*:scoring function; A:set of m attrbutes used in f Output:*Y*:list of top-k tuples
Phase 1. For each scoring attribute α do Create a list L_α and add all sub-domains of α to it; Remove from L_α the sub-domains which do not satisfy *Q*'s condition; Sort L_α in descending order of its sub-domains;

Phase 2. end-condition := false;
For each scoring attribute α do in parallel
i := 1;
n := number of sub-domains in L_α;
While (end-condition = false) and ($i \le n$) do
Send *Q* to the peer *p* that maintains the α values whose sub-domain is $L_\alpha[i]$. *p* returns to p_{int} its values of α which satisfy *Q*'s condition, one by one in descending order, along with their corresponding tuple storage key;
v := the first α value returned by *p*;
While ($v \neq$ null) and (end-condition = false) do
Retrieve the corresponding tuple of *v* and compute its score. If it is one of the *k* highest scores, then record the tuple in a list *Y*;
If there are *k* tuples in *Y* whose scores are higher than the Threshold then set end-condition to true and return to the user these *k* tuples;
If end-condition is false then set *v* to the next α value returned by *p*;
If *v* is null (*i.e.* all values returned by *p* have been received) then set *i*:= *i* + *1*;

我们已经证明了，DHTop能够正确的处理单调函数和一大类非单调函数。

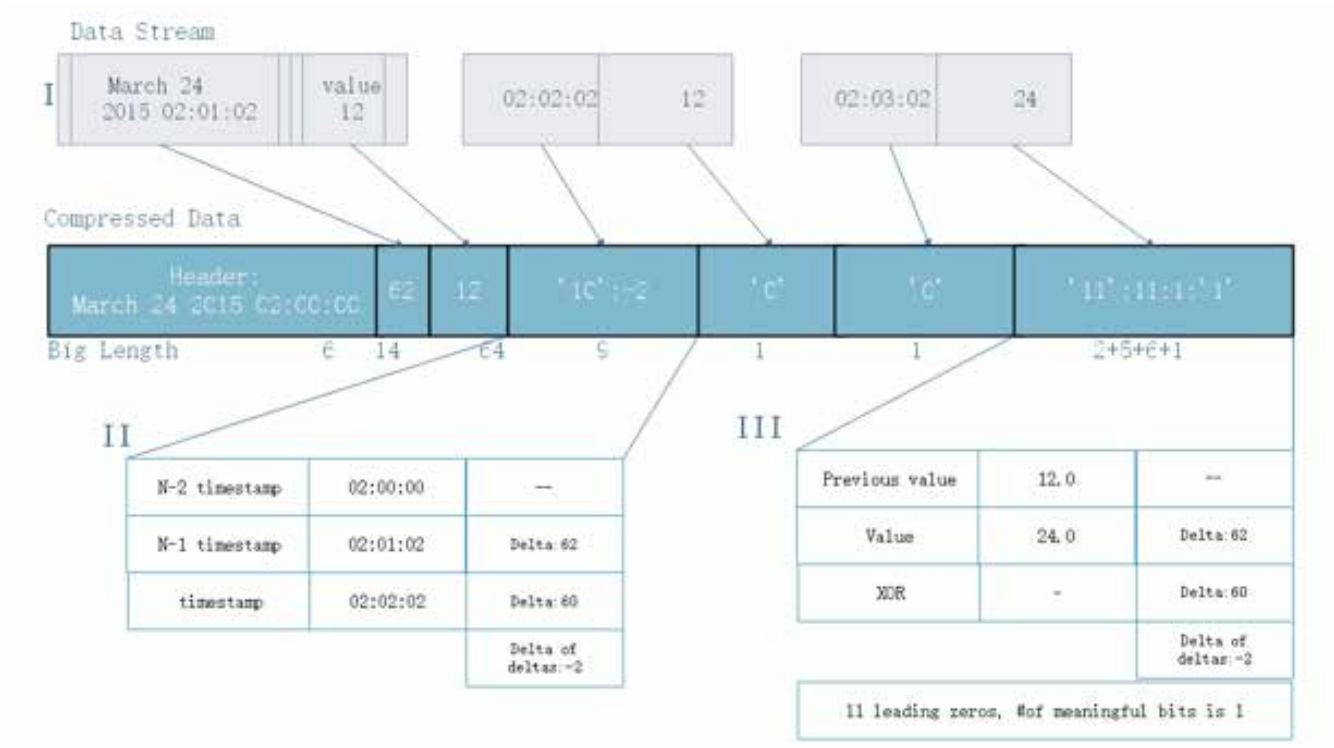
9/数据采集端——Lambda Agent

所有的分布式应用能够都通过调用Lambda Agent来进行数据的接入。Lambda Agent是一个运行部署在操作系统层面的程序，其本质是一个Lambda的钱包，存有Lambda的区块链账本和路由表。多个Lambda Agent的同时运行将组成雾网络，也可以提供一定的数据验证能力。

我们的Lambda Agent SDK除了可以接入数据以外，还提供多种免费的数据采集和发送能力，包括性能监控、安全分析和数据分析等。通常这一类商业产品的价格非常昂贵，性能监控和安全监控系统的售价从几万到几百万美元不等。现在我们免费提供给用户，这就是基于Lambda开发分布式应用以外的一个用户群体，我们吸引用户安装我们的Agent系统，并提供用户愿意提供的计算和存储能力以获取Lambda Cryptocurrency。这是增加Lambda总体算力的一个非常好的手段，也使得我们的钱包可以到达更多的用户。

内存型TSDB

Agent内置实现了一个内存型TSDB——Beringei，支持速度非常快的内存存储，并由硬盘保证数据持久性。存储引擎的查询总是在内存张处理，提供了极高的查询性能，除非需要到磁盘查询，否则一般不进行磁盘操作，所以可以在停机时间极短、数据没有丢失的情况下重启或迁移。Beringei使用极其高效的数据流压缩算法，采用的数据流压缩算法能够将实际的时间序列数据压缩90%以上。



性能监控

Agent包含了性能管理模块，它满足云环境（公共云，私有云和混合云），数据中心以及大型传统IT基础架构环境的部署要求，并为要求苛刻的分布式、动态、敏捷环境提供应用性能监控解决方案。

Metrics数据上传

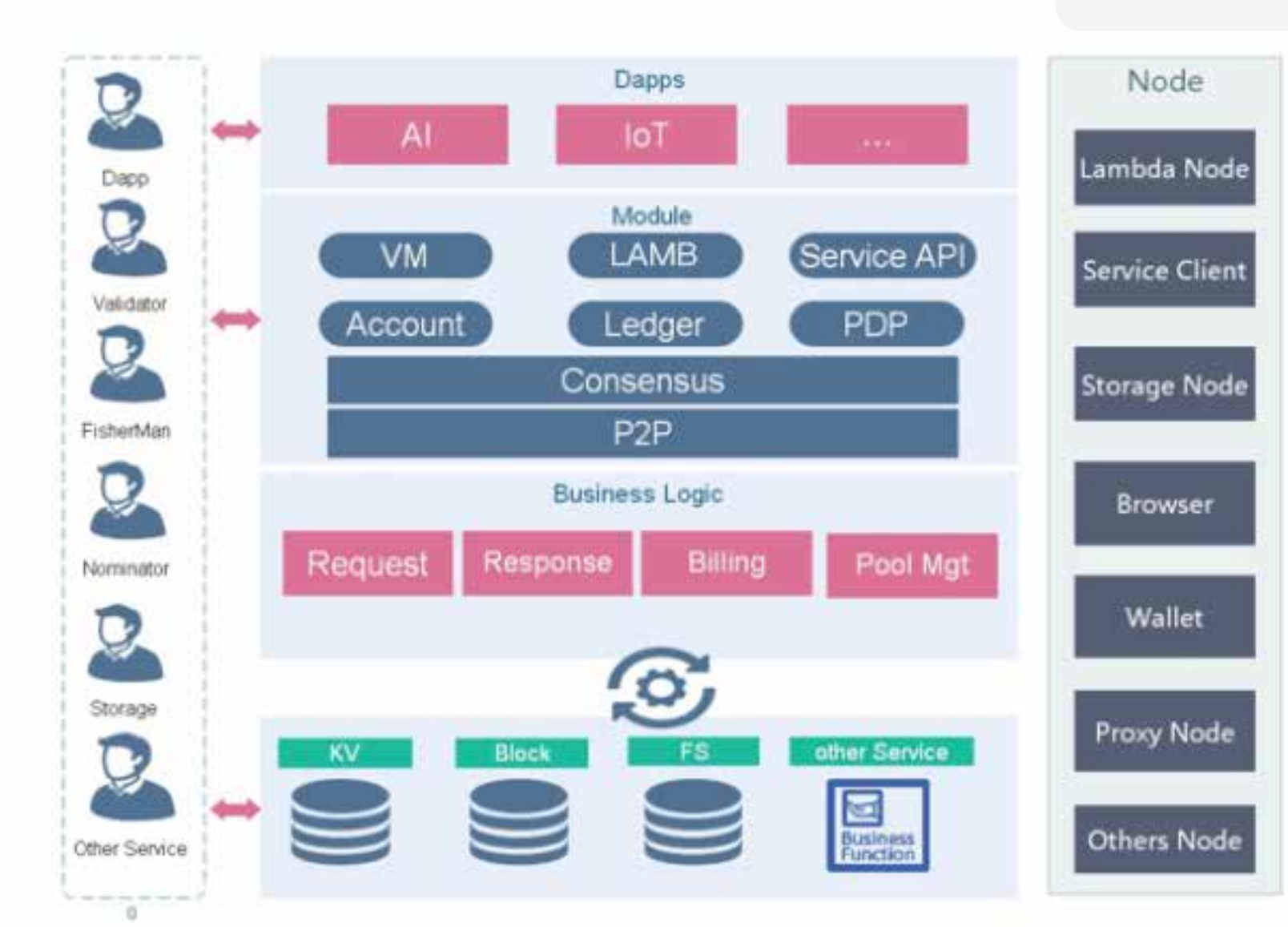
任意Dapp都可以通过调用Lambda Agent实现Metrics数据的上传。

三、Lambda 生态系统

1/Lambda 经济生态

不同于大多数的区块链应用，Lambda是一个区块链的数据存储基础设施，并带有自己的若干条用于计费、交易、加密和访问控制的链。Lambda项目的原生代币LAMB产生要消耗节点的内存和存储资源。在Lambda平台之上，可以交易的资源主要是格式数据的访问能力，快速访问能力是硬盘存储能力和内存大小的结合体，由于存储之后的数据是以加密的形势存放，应用访问的时候需要解密，因此，也需要消耗CPU资源。我们在衡量存储提供方贡献的时候，以对数据块的千次解密并返回作为单位计价单元。

Lambda生态中，参与各方的角色分别是Dapp开发者和项目方、链节点参与方、存储节点参与方和其他参与方如投资人。其中链节点的角色在第一张已经进行过阐述，主要是提名人、验证人和钓鱼人。Lambda不同于其他主链的点在于，我们有自身的业务逻辑(Business Logic)，我们要对用户的请求、返回进行处理和记账，并且管理存储资源池内的共识和结算。



Lambda的数据存储池逻辑由四种角色组成：存储资源提供者，存储资源购买者，社区开发人员和数据购买者这四个群体组成了Lambda相互依存的生态系统。并且，从商业模式来说，这四类角色都有可能是个人用户，也有可能是企业。

Group	Lambda features	Incentive to participate
Tenant	Lambda offers tools to execute compute-intensive tasks.	Tenant get access to affordable and scalable solutions, which combine hardware and software.
Users	Lambda combines and utilizes (almost) any kind of existing computing hardware.	Users get paid for renting out their hardware
Software Developers	Lambda is a flexible platform to deploy and monetize software	Software developers use Lambda as a distribution channel, associated with access to hardware
Data analysts	Lambda DB is a time series database	Data analysts can search and subscribe data source from data producer,and then they can buy and analyze this data.

存储设施提供者

存储能力主要由用户群体来提供，这些人可以是任何安装了Lambda客户端的人和组织，从个人用户将PC的闲置资源出租，到大型数据中心将算力和存储完整的提供出来，我们可以驱动从小到大的无数计算和存储资源。目前，已经确定加入Lambda供应商的，除了个人用户，还有多家IDC云计算中心，有类似于斐讯这样的NAS平台运营者的数万台NAS设备，也有多家网吧和网吧平台的数十万台PC设备。这些供应商有动力加入Lambda，因为他们可以从完成的任务中获得数字货币LAMB的奖励。对于供应商来说，如今的低配硬件和PC资源无法通过工作量证明获得比特币这类需要哈希计算的数字货币，并且，常规CPU和GPU的算力和ASIC相差越来越远，甚至是Sia这样的存储类代币，也都是使用POW工作量证明。Lambda使用的POS工作量证明，以及存储就可以获得数字货币的方式，对PC和IDC来说是最佳选择。同时，Lambda的探针程序可以实时监控到基础设置提供者的相关基础设施运行情况、资源耗费情况、甚至是终端用户的体验情况，以保证设备的出租不会对原有业务造成任何影响。

对数据存储和交易的需求

今天的分布式应用开发者将越来越体会到Lambda项目的价值，在Lambda平台之上，他们不但可以很容易的使用Lambda作为一个Baas服务来开发分布式应用，以远远低于常规云主机和云存储的价格享受各种高性价比的性能分析和安全保证服务，而且可以将数据的价值进行出售，出售数据的回报有可能远远高于产生数据的成本。

Lambda项目正在探索一种安全高效的去中心化数据共享模型，在这个模型中，Lambda首先从共享数据集中提取多层元数据信息，通过各共识节点建立域索引，以解决可连接数据集的高效发现问题。其次，从交易记录格式及共识机制入手，建立基于区块链的数据交易，以实现交易的透明性及防共谋等舞弊行为；最后，根据数据需求方的计算需求编写计算合约，借助安全多方计算及差分隐私技术保障数据所有者的计算及输出隐私。这里主要用到的技术包括分布式哈希表和局部敏感哈希。

软件和微服务

Lambda团队将在未来的时间把更多的特性引入到Lambda平台，但与其他软件开发人员共同开发自己的Lambda应用创意至关重要，这种应用的数量和质量是Lambda未来成功的关键因素之一。我们未来提供的数据存储和数据分析服务，可能会和现有的主流云平台旗鼓相当。对于其他的开源研发团队来说，通过成为Lambda系统的WorkChain，可以轻松的将自己的开源软件能力向区块链领域开放。我们将通过Lambda基金会，推动世界的区块链基础设施向前不断发展。

2/Lambda Cryptocurrency LAMB

主链共识算法: POS

我们选择Nominated Proof–of–Stake NPos作为共识算法

签名算法:可变签名类型

每个公钥均具有一个说明符(specifier)（一个16字节的数组）和包含公钥编码(encoding)的字节切片(byte slice)。 说明符用来指定在校验签名时使用哪个签名算法。，每个签名将是一个字节切片，其编码可以通过查看相应公钥的说明符来确定。

备选签名算法有：ed25519、ECDSA secp256k1、Schnorr secp256k1

Lambda Cryptocurrency（LAMB）

Lambda Coin ： Lambda Coin（简称“LAMB”）是Lambda项目的核心，也是Lambda项目颠覆传统的商业闭源软件开发模式的主要武器。传统软件公司的商业模式下，存在着大量的交易成本和组织成本，软件公司为了价值的转移和交付，不得不雇佣大量的人工来获取用户、提供服务、解决问题，在Lambda项目中，这些可以更加优化的方式完成。

- **POS共识机制：**Lambda链的共识机制采用POS工作量证明，验证人完成哈希计算可以得到Lambda奖励。另外，数据库存储节点在指定时间完成对对指定大小的数据文件存放，根据合约可以得到Lambda Coin奖励；
- **UTXO模型:**作ambda链的账本结构采用UTXO模型，同时支持Account模型，设计实现方式类似于Qtum。
- **跨链交易和BCP协议:**作为UCE经济体系联盟的成员，我们的设计中遵循BCP协议，并且支持在UCE体系内的原子跨链交易和原子合约交易。
- **EVM和智能合约：**我们将在链上实现对智能合约的支持，并提供完备的合约形式化验证能力。
- **总量：**Lambda Coin的总量是100亿枚
- **质押：**验证人和提名人需要质押LAMB，质押总量约等于LAMB总数的8~12%。
- **流通：**第一年，40%的Token还在等待出块奖励，20%的Token锁定在基金会账户，10%的团队Token和20%的私募也在冻结期尚未解锁，市面可流通的Token最大约为10%。
- **租用：**租用云端Lambda数据库的用户需要在交易所购买LAMB。
- **交易：**在Marketplace交易数据的人，买方需要在交易所购买LAMB。
- **规则：**Lambda Coin的的分配规则是，创始团队10%，募资30%，基金会20%，其余40%为出块奖励，20年挖完。20年后，代币总数保持每年约0.5%的缓慢增长。

3/区块产生规则

MainChain区块和ShardChain区块

最大的区块大小为32*10^6(32e6)字节，也就是32M。交易(transaction)大小没有限制，但它必须能够被容纳于区块内部，验证人制每个交易的大小。

每个区块都有一个最小准许时间戳。它是通过取前面10个区块的中间时间戳来确定的。如果之前的区块不足10个，则需重复使用初始时间戳(genesis timestamp)。

区块ID的产生规则是：H(父区块ID + 64位随机数+ 区块Merkle树根节点)。

对于有效的区块，区块ID必须低于某个特定目标。

MainChain和ShardChain的出块预期都是5秒钟。

MainChain中包含了所有ShardChain的区块的哈希，ShardChain包含了上一个MainChain区块的区块头。

交易

一个交易(Transaction)主要由以下几个对象组成：

- LAMB输入(LAMB Inputs)
- LAMB输出(LAMB Outputs)
- 文件合约(File Contracts)
- 文件合约修订(File Contract Revisions)
- 存储证明(Storage Proofs)
- 矿工酬劳(Miner Fees)
- 交易签名(Transaction Signatures)

所有LAMB输入(LAMB input)的总和必须等于所有矿工酬劳(miner fee)、LAMB输出(LAMB output)和文件合约支出(file contract payout)的总和，没有剩余。在上述对象中存有解锁hash值(unlock hash)。 解锁散列值是“解锁条件(unlock condition)”对象的Merkle树根，解锁条件包含一个时间锁数值(time lock)、若干必需的签名以及可在签名期间使用的一组公钥。解锁条件(unlock condition)对象的Merkle树根节点，是由时间锁(timelock)、公钥（每个公钥对应1个叶子节点）以及签名的数量所构成的Merkle树的根。 需要提供足够多的签名，并且区块链的高度至少等于时间锁(timelock)的值，才能满足解锁条件(unlock condition)。

解锁条件(unlock condition)包含一组公钥，每个公钥只能在提供签名时使用一次。相同的公钥可以被包含两次，这意味着它可以使用两次。所需签名的数量指示必须使用多少公钥来验证输入(input)。如果所需的签名数量为“0”，则输入(input)是“任何人都可以消费”。如果所需的签名数量大于公钥的数量，则输入(input)是不可消费的。

合约

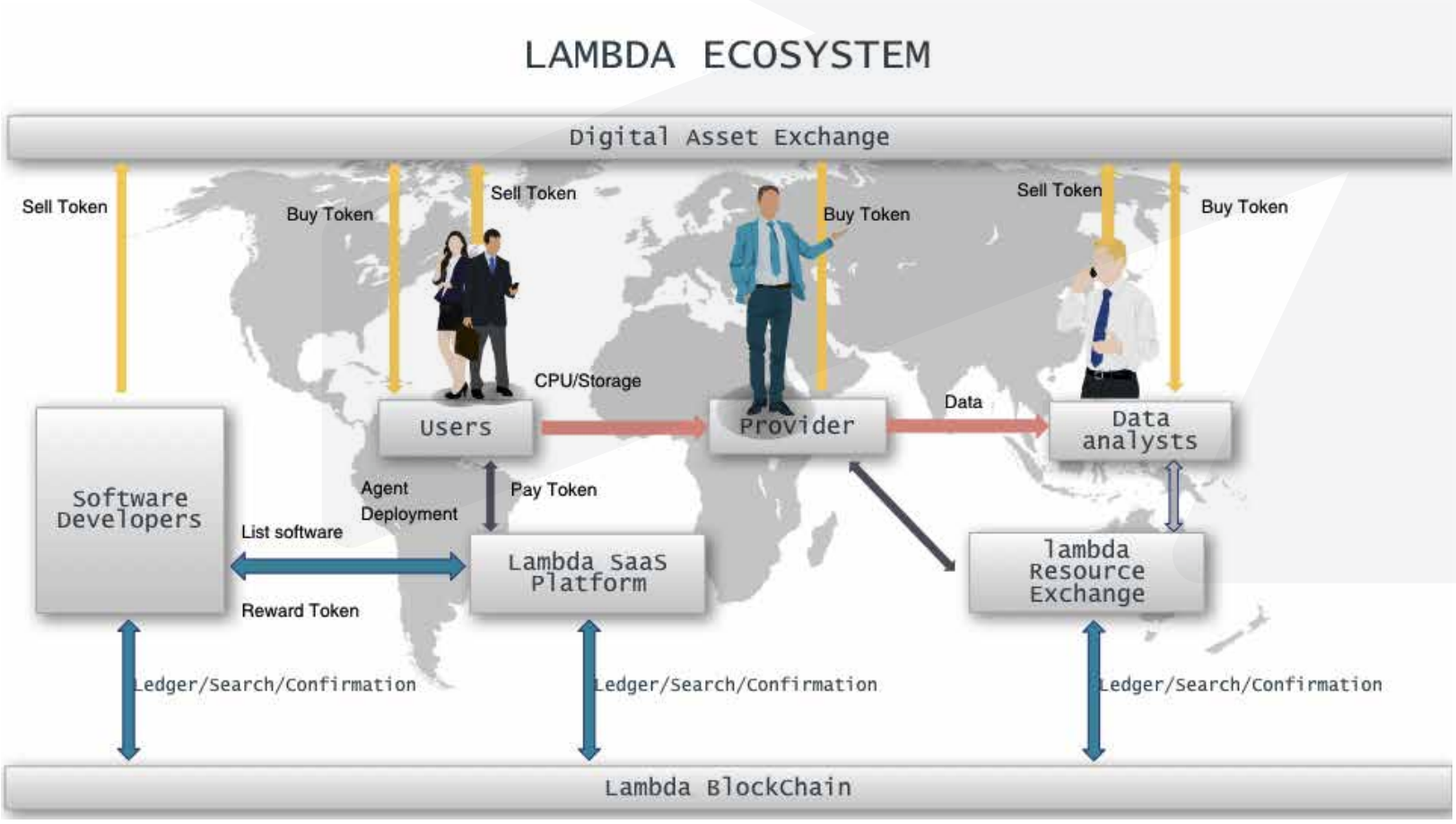
数据存储合约是数据存储提供商与其客户之间的协议。客户选择Lambda提供的虚拟数据节点来存储数据，并进行相应的数据库检索等操作。客户和存储提供商之间签订的是一个持续性协议，并根据一定的条件进行分期付款。不同于文件类存储的是，数据库存储的数据是慢慢增大的，因此，客户向每个节点的数据库服务商的每次付款金额是：

总约定金额* 变更周期 * 当前使用的存储空间大小/ 总持续时间 * 总约定空间

付款周期取决于验证人节点对数据存储节点进行数据持有性证明的周期。

一个数据合约的核心是数据存储文件的Merkle DAG。。

4/Lambda 经济系统



Lambda的经济系统的角色由以下几类组成：

链节点

- 链节点上又分为验证人、提名人和钓鱼人。链节点主要提供了对服务访问的路由、记账、访问控制和加密的能力。
- 链节点的主要受益来自于POS的Staking交易的挖矿收入，另外，钓鱼人可以对验证人的记账信息和存储节点的存储信息进行验证，并获得受益

存储节点

- 存储节点的主要收入来自用户支付的费用
- 存储节点需要预先质押一部分资金，并获得POS的分成受益
- 存储节点需要定期向验证人发送心跳信息，其中的幸运者可以获得一些奖励，这个设计主要是为了激励存储节点的在线

用户

- 用户是采购并使用存储节点的人，通常来说，会是其他应用链和Dapp

投资人

- 投资人投资于Lambda项目，并分享LAMB上涨带来的收益

交易所

- 数字货币交易所

5/平台能力

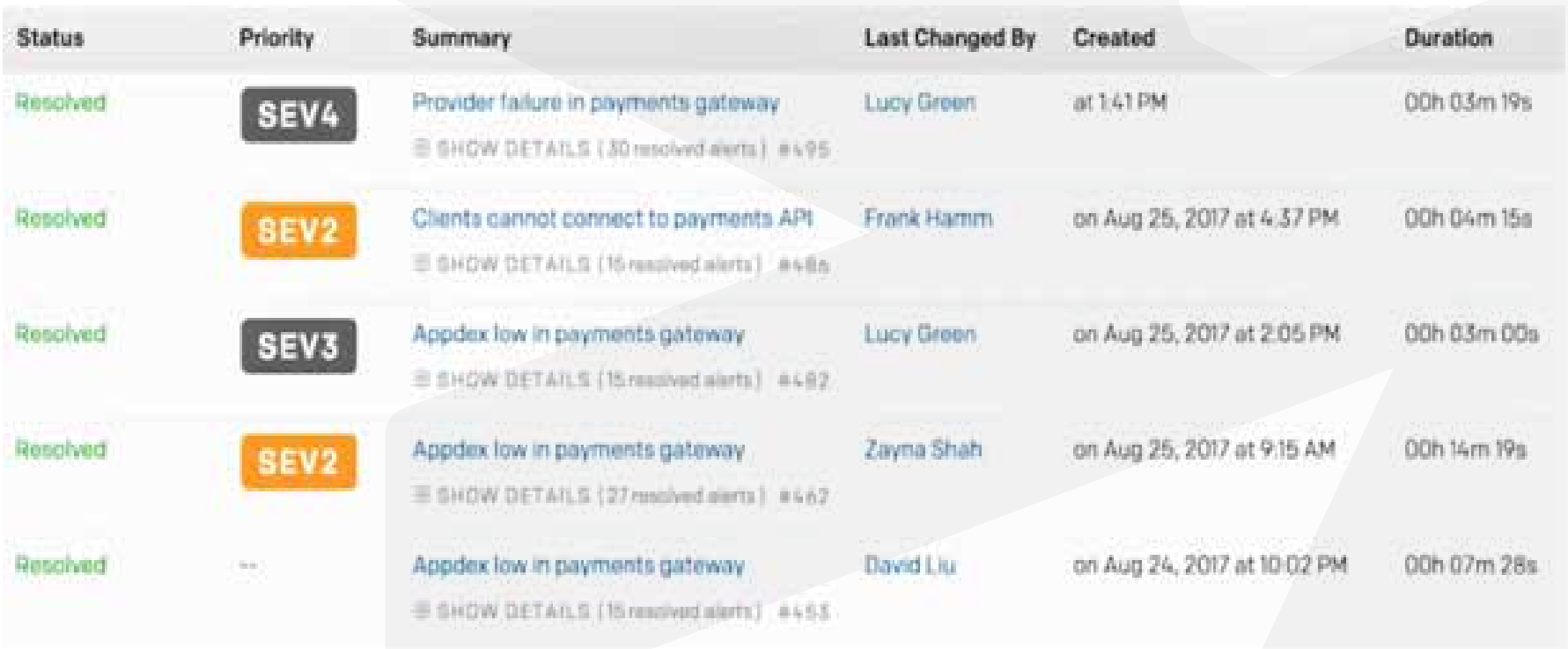
3.5.1 对修复和提交bug的奖励机制

软件的用户在使用过程中遇到问题，会选择向社区提交Bug，社区的代码开发人员会按照代码bug的影响范围和重要程度，依次修复Bug。修复之后的Bug会用Patch的形式做成安装包，放到社区供用户下载。

在这个模型下，由于过程不连贯，并且代码bug的修复判断依赖于人，导致很多基础软件都没有能够升级到最近版本，给系统整体可用性、性能和安全性带来了极大影响。Lambda项目通过智能合约的形式，用户可以发布悬赏，消耗一定的LAMB要求开发人员修复，修复之后的Bug patch会自动通过Lambda Agent Release Automation模块来安装到用户系统之内，无须手动操作。

最初提交Bug的用户，会根据后续的其他用户使用Patch的情况，得到LAMB奖励。

3.5.2 对性能提升的奖励机制



3.5.3 安全漏洞自动防护

Lambda Agent还收集各种安全数据。然后向安全社区报告。我们使用AI检查数据，并将安全数据进行汇总。如果是新的或罕见的安全问题，将会向公司生态安全专家报告。如果确认是安全问题，例如一个成功的CVE被公开承认宣布，第一个报告者将被授予LAMB币。

四、技术路线图

1/周期

我们是一群奋斗在基础软件研发和开源社区一线的程序员群体，我们向Apahce基金会旗下的多个开源项目贡献代码，如Camel、Akka、Drill等项目，都有我们团队成员的深度参与。我们也在中国积极推动开源和社区的发展，成立了Druid中国用户组和Clickhouse中国用户组。

我们的开发团队长期以来一直致力于开发使用Agent模式收集海量数据并存储、分析和展示的软件，过去的几年里，我们基于探针技术成功发布了几款产品。如Cloud Insight，Application Insight。在后面的时间里，我们将逐渐把多年的技术积累运用到Lambda项目中。

阶段一 Lambda技术论证与研究（截止17年 Q4）

- 深度研究区块链各个公链和协议层的技术框架，核心团队阅读了大量的代码和论文(参见附录)
- 论证Permissionless环境下的海量数据的计算和存储模型
- 论证OAS商业模式和Token经济的结合点

阶段二 Lambda技术论证与研究（18年 Q1）

- 选择合适的区块链共识机制，确定了采用Sharding方案的高速链设计，以及子链的HoneyBadgerBFT共识算法
- 确定采用Request和Reponse分别记账的多子链设计
- 完善链库结构分离的结构设计
- 确定采用PDP的数据持有性证明和ABE的访问控制和加密机制
- 确定技术路线和技术方案
- 完成技术白皮书

阶段三 Lambda核心组件开发（18年 Q1、Q2、Q3）

- 对libp2p和devp2p的性能进行评估测试，代码修改
- 实现底层的数据块储系统
- 闭源开发Lambda Chain
- 对Lambda FS进行技术验证
- 对Lambda FS进行社区化开发并实时开源
- 基于Lambda FS实现Lambda数据库
- 基于热点语言开发Lambda Agent方便与现用应用集成。

阶段四 Lambda Chain测试网络开发（18年 Q4）

- P2P网络开发用于同步区块链账本以及存储数据同步。
- 虚拟机开发以及内置合约的开发与验证
- API接口、RPC接口、命令行、外围工具开发
- 开源Lambda Chain

阶段五 测试网络搭建与基于Lambda范例开发（19年 Q1）

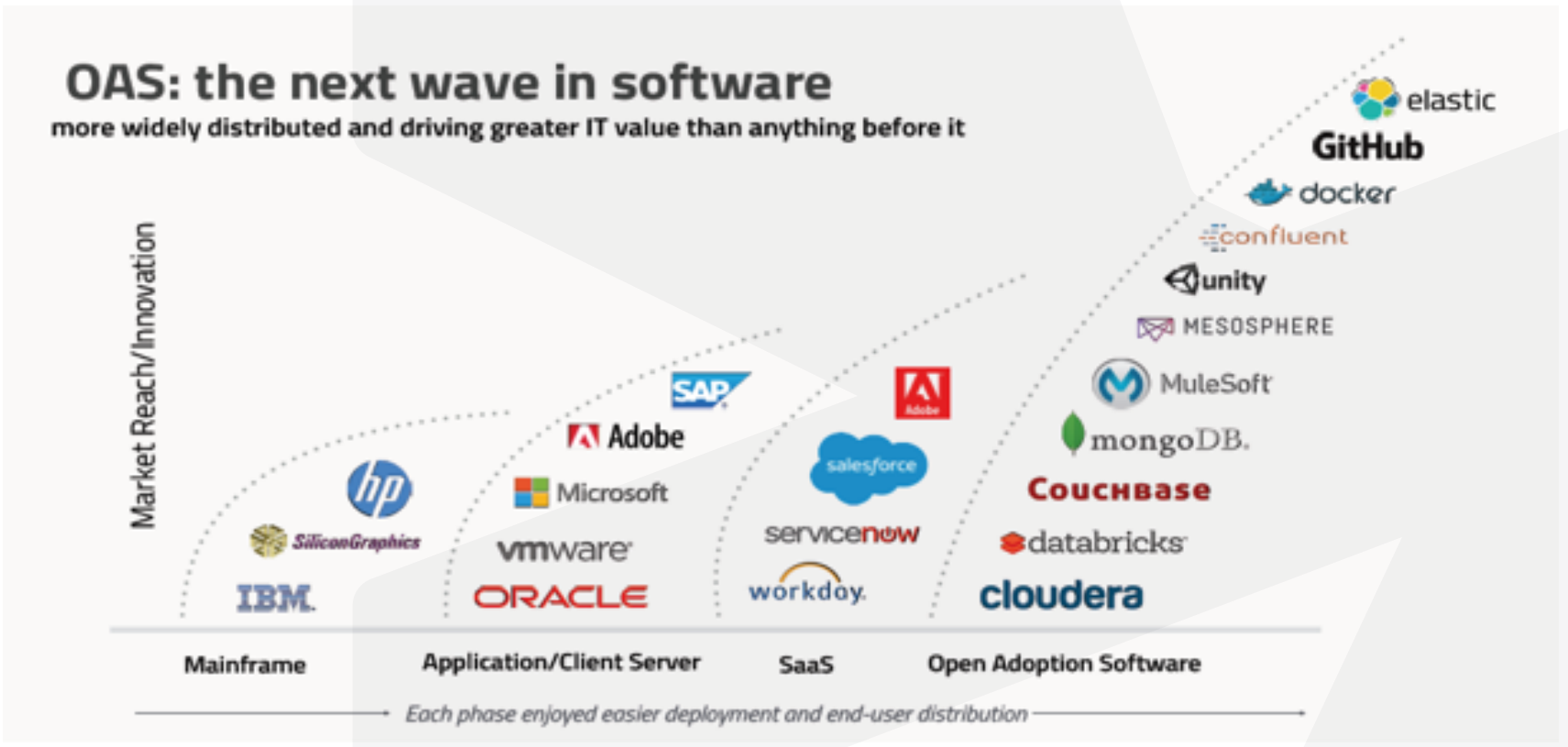
- 基于社区搭建大规模的测试网络，进行全方位的测试。
- 发放LAMB鼓励早期社区爱好者开发基于Lambda的应用示例
- 基于测试网络对应用示例进行安全性与业务漏洞悬赏测试并修正。

阶段六 其他数据服务能力的扩展与数据交易（19年 Q2）

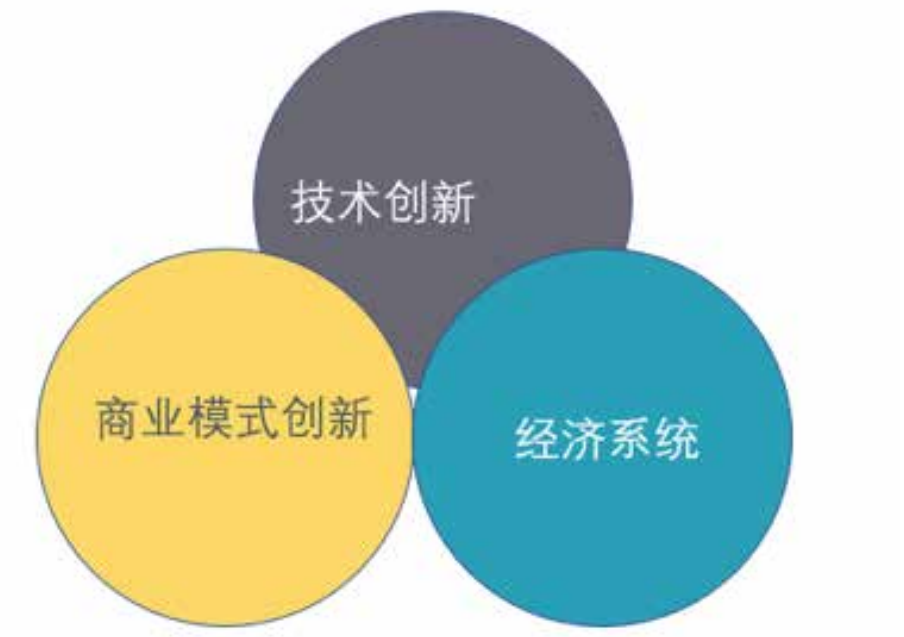
- 由基金会发起鼓励更多的数据库类型与服务能力接入Lambda生态
- 探索基于Lambda生态的数据交易能力

2/我们的理念

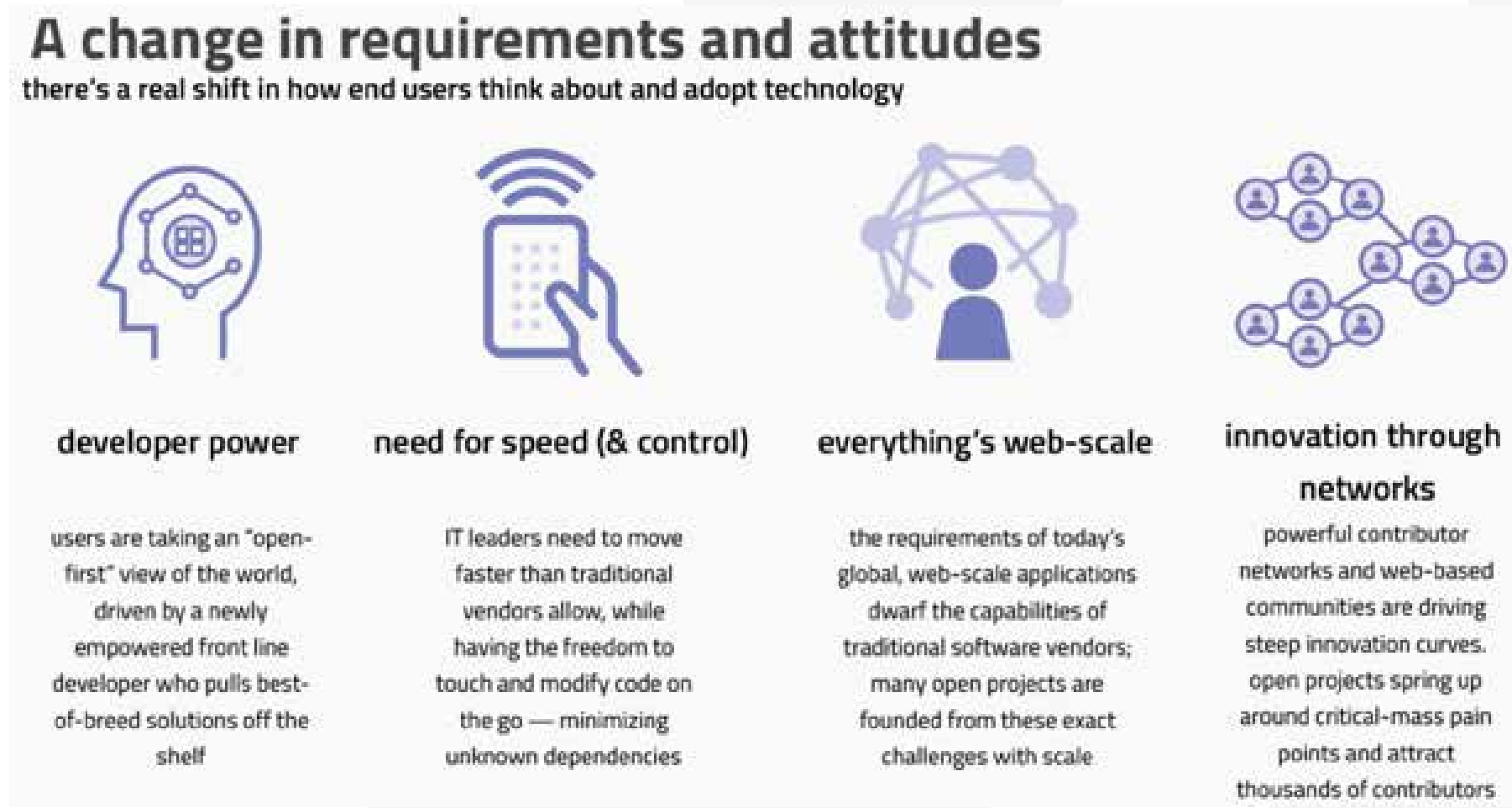
我们坚信OAS模式将是所有的基础软件未来的主流开发方式，在社区协同的模式下和Cryptocurrency激励下，软件领域必将迎来大的变革。但仅有OAS模式是不够的，OAS加上区块链将是完美融合的大规模开发人员合作方式。



当前绝大多数的公链项目都是只有一个公链，比如以太坊或者EOS,包括后续的波卡链等等，另外，提供基础设施的也都是在链的周边提供设施，比如提供去中心化的域名解析、网盘、文件等等。这个时候，链本身就是目的，但是我们这个项目并不是如此，或者说不仅是如此。这个项目中包含的概念，是超越了当前区块链项目的，我们这个项目不仅仅是在区块链领域进行技术创新，更多是技术创新、商业模式创新和经济系统的一个结合。如果比特币的系统中不引入经济模型，那么分布式系统的一致性问题永远无法超越raft、paxos和pbft模型。所以，如果仅仅从技术角度去看待分布式数据库的实现，那么，这个问题会一直存在争论。我们并不是要去做一个数据库软件公司，我们是要走出一条新的路。



我们思考的起点并不是区块链的技术设施，更多是关于如何在中国的基础软件领域建立一个10亿美元以上估值的独角兽公司，全球已经有新的模型出来，就是基于社区的OAS模型，OAS的全程是Open Adoption Software。正如比特币把经济系统引入了分布式领域，OAS模型是全球第一次给软件领域带来了一个新的名词，就是网络效应。



我们希望通过这种全新的模式，在中国的基础软件领域，走出一条全新的路径。

五、Lambda 管理层

1/Lambda创建

Lambda Foundation was established in Singapore. The Foundation will dedicate in Lambda project. Oversee the project development. The General Meeting of Members shall be at the highest authority of this Association.

2/Lambda 董事会

The Council consist of 5 members.

The Council:

1）To carry out the Rules of this Foundation and the resolution of the General Meeting.

2）To appoint or to remove the sub–committees.

3）To take a lease of any equipment or premises or any place in the Republic of Singapore as may be required on 、behalf of this Association.

4）To determine from time to time the amount of entrance fee and monthly subscriptions.

5）To manage all urgent and important affairs or matters.

6）To manage the Foundation's properties and shall file a list of members with the registrar, whenever changes occurred.

3/开发社区治理和激励

Lambda将根据社区开发人员对项目的贡献度对开发人员进行LAMB激励，我们主要的激励对象是对核心文件贡献度较高的开发人员。Lambda项目的源文件是一个高度耦合、高度复杂的体系,要判断其中哪些文件包含了该项目最核心的技术并可以认为是项目的核心文件存在,从不同的角度可以有不同的解释.如果不能妥善解决该问题，将会给我们的激励造成了实际操作性的困难。为了避免绝大多数社区治理中出现的种种弊端，我们采用数学模型的方式事先确定激励规则，并严格按照规则执行。

我们的激励规则基于如下事实：在目前的协同软件开发中,高层次的开发者通常在立项初期将核心技术部分以接口、封装类的形式写入命名空间进行包装,提供给一般开发者让其来频繁调用并将之具体实现,以应对具体的业务需求。这样的形式在面向对象的开发语言中尤为常见。我们选取被引用次数占前 90%的文件作为项目的核心文件组。考虑到可能有部分在后期并入项目,但因为存在时间较短,被调用次数并不多的核心技术文件存在,我们将开发者第一次编译文件的时间等效为文件加入项目的时

间,观察项目中所有文件在项目生命周期中的分布情况以及其自身特性。通过对众多开源项目的分析我们，我们发现有部分文件虽然处在生命周期的中后段,但在加入项目之后的短时间内被编译提交的次数极为频繁,可以认定其为项目的核心技术文件。Lambda社区将对选出的每一个核心文件分别构建起文件关联网络。假设项目中全部的开发者是单个核心文件的作者,通过文件贡献分配算法计算后得到长度与总开发人数相同的贡献分配向量,代表所有开发者在该核心文件中的贡献份额比例。对每一个核心文件反复进行构建和贡献分配向量计算得出每个核心文件对应的贡献分配向量,对其求平均值,将平均向量中值的分布等效为所有开发者对核心文件部分的贡献大小比。并且,为了控制激励过程,我们依照 20/80 法则,按降序取占贡献大小总和前80%的开发者为核心开发者,其余则定义为外围开发者。

在分析获得项目的核心文件组后， Lambda社区将依次对各个文件中开发者贡献比重进行分析。开发者文件贡献分配算法依据文献科学中的共被引机制,利用单个文件在关联网络之间的引用关系,来动态感知文件开发者的贡献度大小变化,同时通过引入控制变量来防止贡献数值的恶性迭代。其从关联网络架构上分为三层， 用以计算单个文件中各开发者的贡献份额比重.选取单个核心技术文件 P0 ,使其成为关联网络的第一层.选取调用该P0文件的所有文件 dk (1≤k ≤ s ,s为文件的个数)为关联网络的第二层,其中包括了对第一层核心文件进行了调用的其他的核心文件和非核心文件。选取被dk文件调用的但没有被选为核心文件的文件 Pj (1≤ j ≤ t , t为文件的个数)作为关联网络的第三层。

建立文件关联网络之后,进行开发者贡献比重计算,步骤如下：假设该核心文件P0由m个开发者共同完成 {ai }(1≤ i≤ m),而调用文件P0的所有文件集合为D={d1 , d2 ,... , dn,} ,所有D 内的文件调用的文件集合定义为P ={ P1 , P2, ..,Pn }。利用以上文件关联引用集合,提出分配矩阵T ,矩阵元素 Tij的值代表项目开发者ai在被调用文件 Pj中的贡献度。这里的贡献度代表开发者在该文件作者中所占的人头比。如开发者是该文件两名作者之一，贡献度为 1/2。若没有参与过该文件的开发,贡献度为 0。矩阵的行代表项目开发者,列代表D集合调用的文件；引入向量集合s = (s0, s1,...,sn)T ,其元素si代表集合D内的文件调用Pj的数目,开发者ai在文件P0中最终贡献度ci的计算如下:

其矩阵形式为:

$$c_i = \sum T_{ij} S_j$$

$$C = TS$$

六、核心团队

1/起源

我们是一群奋斗在基础软件研发和开源社区一线的程序员群体，我们向Apahce基金会旗下的多个开源项目贡献代码，如Camel、Akka、Drill等项目，都有我们团队成员的深度参与。我们也在中国积极推动开源和社区的发展，成立了Druid中国用户组和Clickhouse中国用户组。

我们曾相聚于OneAPM公司的基础架构部。OneAPM是一家 APM SaaS公司，曾获得知名VC多轮投资。OneAPM的业务特点是应用性能监控，因此，会有海量的数据源源不断7X24小时从各个移动终端、浏览器和服务器汇聚到OneAPM的服务端，服务端需要支撑海量数据的实时写入、计算和查询。在高峰时期，OneAPM SaaS系统需要每天处理一千亿条数据。

为了进行这些数据的处理，我们搭建了全球规模居于前列的分析性数据库druid集群，深度改写了Clickhouse数据库，创建了中国druid用户群，编写了多本技术书籍，在整个中国推广了海量数据实时分析的实践经验，团队中也因此而产生了多个著名开源项目的commiter和montor。

从2017年开始，我们根据自己的经验和业务特点，开始用开源社区的方式来创建一个完全去中心化的高可用数据库软件。在此过程中，得到了Apache基金会、Akka社区、Druid社区和Clickhouse团队的大力帮助，并且，OneAPM、亿方云、巨杉数据库三家著名的软件公司也以合作伙伴的方式加入进来，形成了今天的Lambda项目。

Lambda项目在组织方式上，会完全按照社区化的方式来运行，类似于Linux、Docker、Kafka等著名的开源项目。不同的是，经由我们的努力和区块链的结合，会自发将原来开源社区的本地化单点部署连接成一个自动形成的云，这就是区块链的奇妙之处。

2/合伙人

何晓阳	中国著名基础软件专家，JVM社区成员，曾在BEA Systems和Oracle担任研发工程师。 2008年创立蓝海讯通OneAPM，并于2014年推出基于SaaS模式的应用性能管理产品。OneAPM在2014年一年内完成经纬中国、成为资本和启明创投投资的四轮融资，累计三亿元人民币。OneAPM曾入选Forrester和Gartner报告，被评为亚太地区有领导力的ITOM厂商。OneAPM的SaaS产品服务于中国的几十万开发者用户和数千家企业，在金融、电信、政府、大型互联网领域有广泛的影响力。蓝海讯通于2016年登陆新三板创新层，股票代码838699，何晓阳被称为“中国APM行业第一人”。 何晓阳2015年入选创业家 35 under 35榜单，微信公众号“何晓阳读书笔记”在程序员社区和企业服务领域有影响力和相当范围的读者群。
李沫南	中国顶级程序员，Coreseek和Log Insight的创始人 作为搜索引擎和文件系统、存储系统、日志系统专家，李沫南创作的CoreSeek中文分词系统曾广泛用于中文互联网BBS社区。曾作为技术顾问为多家知名互联网公司提供技术咨询与顾问服务，熟悉从操作系统内核到浏览器到数据库的全套技术栈，对全文检索、OLAP数据库、编译理论、虚拟机设计、文件系统和数据压缩有丰富的理论与实践经验。目前主要研究分布式智能存储。
Alex Lototskiy	senior principal security engineer, has been worked for Symantec for 17 years. More than sixteen years’ professional experience in designing and developing software. Strong skills in Object–Oriented design, knowledge of database schema design, and development of Client/Server applications. Major specialization in computer and network security, integration, and creation of systems for detection and prevention of malicious and spyware attacks. Through years of work, acquired excellent practical experience in all phases of product development, starting with taking a business task description, doing analysis, creating requirements and functional specifications, research, and prototyping. Then designing optimal technical solutions, creating a framework, and technically leading the engaged development teams through sprint development cycles, providing good quality with automation coverage.
何冰清	OneAPM 联合创始人 & CTO，汇编语言专家，2013年联合创立OneAPM，负责OneAPM核心产品Application Insight的开发。
郭宏强	数据科学家，在机器学习和人工智能方面有超过10年的经验，曾在美国波士顿的GNS医疗公司担任数据科学经理，监督供应商网络优化和严重疾病干预优化的数据科学项目，他研发产品为客户每年节省4000万美元。在GNS之前，他在美国康奈尔大学任教职5年，他在数据科学方面的研究得到了美国政府的600万美元赞助。
高海强	计算机硕士研究生，先后毕业于天津大学、内蒙古大学计算机系，主要研究方向为P2P网络设计与优化，曾任网络优化公司埃森诺的技术副总裁，2015年联合创立OneAPM的安全子公司OneASP，担任总裁。OneASP在2015年获得三行资本领投，经纬、成为、启明跟投的1500万人民币融资。
Oleg Lyamtsev	Talented engineer, has more than 26 years software development experience.He is very passionate about software security and is always looking for ways to improve the product that we are working on. He is a fast learner and he is a very good team mate to have. He is a very dedicated, and talented C++ programmer. He has excellent problem solving and debugging skill.
赵海俊	OneAPM 技术副总裁，先后就职于去哪儿网和空中网，于2013年加入OneAPM，帮助组建产品研发团队，负责产品的管理和开发工作。

3/研发和团队成员

研发团队	黄东　王子铭　张新勇　张超　Owen Yang　刘派
市场人员	Lucy Wang　Patti Lin

4/专家顾问

田甲	M.S./B.S., Tsinghua University, Distributed SystemInvestor of Bitfinex & Limited Partner of BitfundConsultant of FBG CapitalConsultant of Zcash Jia Tian has served as senior dev at Baidu, Inc. and Alibaba, Inc. During the period he has developed large scale computer systems including the technology behind so.com, supporting >100 million page views per day, and large scale recommendation systems. Jia Tian is also a serial entrepreneur. He has joined the founding team of several companies focusing in AI and related areas. The first company Wolongyun has been acquired by Alibaba, Inc. Then he has joined the beijing machine learning information technology company as CTO, designed and built AI systems including recsys, chatbot, medical image recognition system. After that he has joined pony.ai to build an autonomous driving system, invested by Sequoia and IDG in angel round. Serves as the Chief Scientist in bitfundpe.com, a bitcoin fund dedicated in supporting the bitcoin community since 2013, founded by Xiaolai Li. Jia Tian is also advisor to multiple blockchain tech startups.
孙仲英	原阿里巴巴技术专家，曾任OneAPM和OneASP的技术顾问，原OKEX CTO
刘春华	国内最顶级的黑客及安全专家，杰思安全的创始人、CEO，在硅谷工作多年， Symantec年销售20亿美金的SEP产品创始人 ，Symantec全球级别最高的华人技术专家之一。中科大少年班毕业
王涛	巨杉数据库联合创始人、CTO 前IBM DB2全球顾问专家，参与设计IBM新一代计算平台，前IBM北美数据库研究中心R & D成员
程远	亿方云创始人、CEO 曾在美国BOX研发中心工作
BMAN	BMAN starts his career in global investment firm Blacksmith Global Ltd. Having a wealth of experience in both secondary market and venture investment, BMAN entered the area of blockchain as early as 2013 as an active investor in early–stage technology companies. BMAN also has a wide range of entrepreneurial experience. He cofounded Lijiaoshou (Shoujiao Technology, Inc.) In 2015, which was the top marketing content company in China and was acquired by Baidu, Inc. in 2017. Com–bined with artificial intelligence and big data technology, BMAN has developed Baidu AOD platform and served more than 500,000 business customers. BMAN served as consultant of dozen Internet listed companies, such as Alibaba, Inc, Lenovo, Inc and Qihoo, Inc, etc. and has accumulated strong relationships with some of the most promising entrepreneurs and top investors in both blockchain and internet industry.

5/合作伙伴

OneAPM是中国领先的基础软件公司

亿方云是中国领先的云文档和云协作SaaS公司

区块链参考文献

[1] Amazon. Amazon Web service. http://aws.amazon.com/products/

[2] Houstn D, Ferdows A. Dropbox. https://www.dropbox.com/

[3] Microsoft. Windows azure. http://www.windowsazure.com/en-us/

[4] Schroeder B, Gibson GA. A large-scale study of failures in high-performance computing systems. IEEE Trans. on Dependable and Secure Computing, 2010. 337—350. [doi: 10.1109/TDSC.2009.4]

[5] http://games.qq.com/a/20110503/000013.htm

[6] http://www.chinanews.com/it/2011/09-05/3305704.shtml

[7] http://tech.163.com/07/ 0813/11/3LP86PEM000915BD.html

[8] Lamport L, Shostak R, Pease M. The Byzantine generals problem. ACM Trans. on Programming Languages and Systems, 1982, 4(3):382—401. [doi: 10.1145/357172.357176]

[9] Clement A, Kapritsos M, Lee S, Wang Y, Alvisi L, Dahlin M, Riche T. Upright cluster services. In: Proc. of the ACM SIGOPS 22nd Symp. on Operating Systems Principles. New York: ACM Press, 2009. 277—290. [doi: 10.1145/1629575.1629602]

[10] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Trans. on Computer Systems, 2002,20(4): 398—461. [doi: 10.1145/571637.571640]

[11] 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. Berkeley: USENIX Association, 2006. 177—190.

[12] Kotla R, Alvisi L, Dahlin M, Clement A, Wong E. Zyzzyva: Speculative Byzantine fault tolerance. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles. New York: ACM Press, 2007, 45—58. [doi: 10.1145/1294261.1294267]

[13] Serafini M, Nokor P, Dobre D, Majuntke M, Suri M. Scrooge: Reducing the costs of fast Byzantine replication in presence of unresponsive replicas. In: Proc. of the 2010 IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. 2010. 353—362. [doi: 10.1109/DSN.2010.5544295]

[14] Yin J, Martin JP, Venkataramani A, Alvisi L, Dahlin M. Separating agreement from execution for Byzantine fault tolerant services. In: Proc. of the 19th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2003. 253—267. [doi: 10.1145/945445.945470]

[15] Wood T, Singh R, Venkataramani A, Shenoy P, Ceeehet E. ZZ and the art of practical BFT execution. In: Proc. of the 6th Conf. on Computer Systems. New York: ACM Press, 2011. 123—138. [doi: 10.1145/1966445.1966457]

[16] Wester B, Cowling J, Nightingale EB, Chen PM, Flinn J, Liskov B. Tolerating latency in replicated state machines through client speculation. In: Proc. of the 6th USENIX Symp. on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2009. 245—260.

[17] Hendricks J, Sinnamohideen S, Ganger GR, Reiter MK. Zzyzx: Scalable fault tolerance through Byzantine locking. In: Proc. of the 2010 IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. 2010. 363—372. [doi: 10.1109/DSN.2010.5544297]

[18] Pease M, Shostak R, Lamport L. Reaching agreement in the presence of faults. Journal of the ACM, 1980,27(2):228—234. [doi: 10.1145/322186.322188]

[19] Matin JP, Alvisi L. Fast Byzantine consensus. IEEE Trans. on Dependable and Secure Computing, 2006,3(3):202—215. [doi: 10.1109/TDSC.2006.35]

[20] Baker MG, Hartman JH, Kupfer MD, Shirriff KW, Ousterhout JK. Measurements of a distributed file system. In: Proc. of the 13th ACM Symp. on Operating Systems Principles. New York: ACM Press, 1991,25(5):198—212. [doi: 10.1145/121132.121164]

[21] Leung AW, Pasupathy S, Goodson G, Miller EL. Measurement and analysis of large-scale network file system workloads. In: Proc. of the USENIX 2008 Annual Technical Conf. on Annual Technical Conf. Berkeley: USENIX Association, 2008. 213—226.

[22] Amir Y, Coan B, Kirsch J, Lane J. Byzantine replication under attack. In: Proc. of the DSN. 2008.

[23] Malkhi D, Reiter M. Byzantine quorum systems. Jorunal of Distributed Computing, 1988,11(4):203—213.

[24] Martin JP, Alvisi L, Dahlin M. Minimal Byzantine storage. In: Proc. of the 16th Int'l Conf. on Distributed Computing. London: Springer-Verlag, 2002. 311—325. [doi: 10.1007/3-540-36108-1_21]

[25] Abd-EL-Malek M, Ganger GR, Goodson GR, Reiter MK, Wylie JJ. Lazy verification in fault-tolerant distributed storage systems. In: Proc. of the 24th IEEE Symp. on Reliable Distributed Systems. 2005. 179—190. [doi: 10.1109/RELDIS.2005.20]

[26] Goodson GR, Wylie JJ, Ganger GR, Reiter MK. Efficient Byzantine-tolerant erasure-coded storage. In: Proc. of the 2004 Int'l Conf. on Dependable Systems and Networks. 2004. 135—144.

[27] Hendricks J, Ganger GR, Reiter MK. Low-Overhead Byzantine fault-tolerant storage. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles. New York: ACM Press, 2007. 73—86. [doi: 10.1145/1294261.1294269]

[28] Hendricks J, Granger GR, Reiter MK. Verifying distributed erasure-coded data. In: Proc. of the 26th annual ACM Symp. on Principles of Distributed Computing. New York: ACM Press, 2007. 139—146. [doi: 10.1145/1281100.1281122]

[29] Abd-El-Malek M, Ganger G, Goodson G, Reiter M. Fault-Scalable Byzantine fault-tolerant services. In: Proc. of the 20th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2005. 59—74. [doi: 10.1145/1095810.1095817]

[30] Luo XH, Shu JW. Summary of research for erasure code in storage system. Journal of Computer Research and Development, 2012, 49(1):1—11 (in Chinese with English abstract).

[31] Li MQ, Shu JW, Zheng WM. GRID codes: Strip-based erasure codes with high fault tolerance for storage systems. ACM Trans. on Storage, 2009,4(4):1—22. [doi: 10.1145/1480439.1480444]

[32] Patterson D, Chen P, Gibson G, Katz R. Introduction to redundant arrays of inexpensive disks (RAID). In: Proc. of the Spring COMPCON'89. 1989. 112—117. [doi: 10.1109/CMPCON.1989.301912]

[33] Cachin C, Tessaro S. Asynchronous verifiable information dispersal. In: Proc. of the 24th IEEE Symp. on Reliable Distributed Systems. 2005. 191—201. [doi: 10.1109/RELDIS.2005.9]

[34] Alvisi L, Malkhi D, Pierce E, Reiter MK. Fault detection for Byzantine quorum systems. IEEE Trans. on Parallel and Distributed Systems, 2001,12(9):996—1007. [doi: 10.1109/71.954640]

[35] Chandra TD, Toueg S. Unreliable failure detectors for reliable distributed systems. Journal of the ACM, 1996,43(2):225—267. [doi: 10.1145/226643.226647]

[36] Lima M, Greve F, Arantes L, Sens P. Byzantine failure detection for dynamic distributed systems. SANTOSDELIMA-2010-584597, RR-7222. Paris, 2010.

[37] Delporte-Gallet C, Fauconnier H, Guerraoui R, Hadzilacos V, Houznetsov P, Toueq S. The weakest failure detectors to solve certain fundamental problems in distributed computing. In: Proc. of the 23rd Annual ACM Symp. on Principles of Distributed Computing. New York: ACM Press, 2004. 338—346. [doi: 10.1145/1011767.1011818]

[38] Liu G, Zhou J, Sun Y, Qin L. A fault detection mechanism in erasure-code Byzantine fault-tolerance quorum. Wuhan University Journal of Natural Sciences, 2006,11(6):1453—1456. [doi: 10.1007/BF02831796]

[39] Reiser HP, Kapitza R. Hypervisor-Based efficient proactive recovery. In: Proc. of the 26th IEEE Int'l Symp. on Reliable Distributed Systems. 2007. 83—92. [doi: 10.1109/SRDS.2007.25]

[40] Sousa P, Bessani AN, Correia M, Neves NF, Verissimo P. Highly available intrusion-tolerant services with proactive-reactive recovery. IEEE Trans. on Parallel and Distributed Systems, 2010,21(4):452—465. [doi: 10.1109/TPDS.2009.83]

[41] Distler T, Kapitza R, Reiser HP. Efficient state transfer for hypervisor-based proactive recovery. In: Proc. of the 2nd Workshop on Recent Advances on Intrusiton-Tolerant Systems. New York: ACM Press, 2008. [doi: 10.1145/1413901.1413905]

[42] Paulo E. Travelling through wormholes: A new look at distributed systems models. Journal of SIGACT News, 2006,37(1):66—81. [doi: 10.1145/1122480.1122497]

[43] Chun BG, Maniatis P, Shenker S, Kubiatowicz J. Tiered fault tolerance for long-term integrity. In: Proc. of the 7th Conf. on File and Storage Technologies. Berkeley: USENIX Association, 2009. 267—282.

[44] Chun BG, Maniatis P, Shenker S, Kubiatowicz J. Attested append-only memory: Making adversaries stick to their word. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles. New York: ACM Press, 2007. 189—204. [doi: 10.1145/1294261.1294280]

[45] Felten EW. Understanding trusted computing: Will its benefits outweigh its drawbacks? Journal of IEEE Security Privacy, 2003, 1(3):60—62. [doi: 10.1109/MSECP.2003.1203224]

[46] Levin D, Douceur JR, Lorch JR, Moscibroda T. TrInc: Small trusted hardware for large distributed systems. In: Proc. of the 6th USENIX Symp. on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2009. 1—14.

[47] Trusted Computing Group. Trusted platform module. http://www.trustedcomputinggroup.org/developers/trusted_platform_module

[48] Singh A, Fonseca P, Kuznetsov P, Rodrigues R, Maniatis P. Zeno: Eventually consistent Byzantine-fault tolerance. In: Proc. of the 6th USENIX Symp. on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2009. 169—184.

[49] Aiyer AS, Alvisi L, Clement A, Dahlin M, Martin JP, Porth C. BAR fault tolerance for cooperative services. In: Proc. of the 20th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2005. 45—58. [doi: 10.1145/1095810.1095816]

密码学参考文献

[1] Fiat A, Naor M. Broadcast encryption. In: Stinson DR, ed. Advances in Cryptology–CRYPTO'93. Berlin, Heidelberg: Springer–Verlag, 1994. 480—491.

[2] Naor D, Naor M, Lotspiech J. Revocation and tracing schemes for stateless receivers. In: Kilian J, ed. Advances in Cryptology–CRYPTO 2001. Berlin, Heidelberg: Springer–Verlag, 2001. 41—62.

[3] Boneh D, Gentry C, Waters B. Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup V, ed.Advances in Crytology–CRYPTO 2005. Berlin, Heidelberg: Springer–Verlag, 2005. 258—275. [doi: 10.1007/11535218_16]

[4] Shamir A. Identity–Based cryptosystems and signature schemes. In: Blakley GR, Chaum D, eds. Advances in Cryptology–CRYPTO'84. Berlin, Heidelberg: Spring–er–Verlag, 1984. 47—53.

[5] Boneh D, Franklin M. Identity–Based encryption from the weil pairing. In: Kilian J, ed. Advances in Cryptology–CRYPTO 2001.LNCS 2139, Berlin, Heidelberg: Springer–Verlag, 2001. 213—229. [doi: 10.1007/3–540–44647–8_13]

[6] Sahai A, Waters B. Fuzzy identity–based encryption. In: Cramer R, ed. Advances in Cryptology–EUROCRYPT 2005. Berlin,Heidelberg: Springer–Verlag, 2005. 457—473.

[7] Goyal V, Pandey O, Sahai A, Waters B. Attribute–Based encryption for fine–grained access control of encrypted data. In: Proc. ofthe 13th ACM Conf. on Com–puter and Communications Security. New York: ACM Press, 2006. 89—98. [doi: 10.1145/1180405.1180418]

[8] Yu SC, Ren K, Lou WJ. Attribute–Based content distribution with hidden policy. In: Proc. of the 4th Workshop on Secure NetworkProtocols (NPSec). Orlando: IEEE Computer Society, 2008. 39—44. [doi: 10.1109/NPSEC.2008.4664879]

[9] Traynor P, Butler K, Enck W, Mcdaniel P. Realizing massive–scale conditional access systems through attribute–basedcryptosystems. In: Proc. of the 15th Annual Network and Distributed System Security Symp. (NDSS 2008). San Diego: USENIXAssociation, 2008. 1—13.

[10] Cheung L, Newport C. Provably secure ciphertext policy ABE. In: Proc. of the ACM Conf. on Computer and CommunicationsSecurity. New York: ACM Press, 2007. 456—465. [doi:10.1145/1315245.1315302]

[11] Cheung L, Cooley JA, Khazan R, Newport C. Collusion–Resistant group key management using attribute–based encryption.http://eprint.ia–cr.org/2007/161.pdf

[12] Yu SC, Ren K, Lou WJ. Attribute–Based on–demand multicast group setup with membership anonymity. Computer Networks, 2010,54(3):377—386. [doi: 10.1016/j.comnet.2009.09.009]

[13] Baden R, Bender A, Spring N, Bhattacharjee B, Starin D. Persona: An online social network with user–defined privacy. In: Proc. ofthe ACM SIGCOMM 2009 Conf. on Data Communication. New York: ACM Press, 2009. 135—146. [doi: 10.1145/1592568.1592585]

[14] Bethencourt J, Sahai A, Waters B. Ciphertext–Policy attribute–based encryption. In: Proc. of the 2007 IEEE Symp. on Security andPrivacy. Washington: IEEE Computer Society, 2007. 321—334. [doi: 10.1109/SP.2007.11]

[15] Beimel A. Secure schemes for secret sharing and key distribution [Ph.D. Thesis]. Technion: Israel Institute of Technology, 1996.

[16] Liang XH. Research on attribute–based cryptosystem [MS. Thesis]. Shanghai: Shanghai Jiaotong University Press, 2009 (inChinese).

[17] Lewko A, Sahai A, Waters B. Revocation systems with very small private keys. In: Proc. of the IEEE Symp. on Security andPrivacy. Washington: IEEE Computer Society, 2010. 273—285. [doi: 10.1109/SP.2010.23]

[18] Shamir A. How to share a secret. Communications of the ACM, 1979,22(11):612—613. [doi: 10.1145/359168.359176]

[19] Pirretti M, Traynor P, Mcdaniel P, Waters B. Secure attribute–based systems. In: Proc. of the ACM Conf. on Computer andCommunications Security. New York: ACM Press, 2006. 99—112. [doi: 10.1145/1180405.1180419]

[20] Baek J, Susilo W, Zhou J. New constructions of fuzzy identity–based encryption. In: Proc. of the ASIAN ACM Conf. on Computerand Communications Security (ASIACCS 2007). New York: ACM Press, 2007. 368—370. [doi: 10.1145/1229285.1229330]

[21] Ostrovsky R, Sahai A, Waters B. Attribute–Based encryption with non–monotonic access structures. In: Proc. of the ACM Conf. onComputer and Communica–tions Security. New York: ACM Press, 2007. 195—203. [doi: 10.1145/1315245.1315270]

[22] Naor M, Pinkas B. Efficient trace and revoke schemes. In: Frankel Y, ed. Proc. of the Financial Cryptography. Berlin, Heidelberg:Springer–Verlag, 2001. 1—20. [doi: 10.1007/978–3–540–68914–0_7]

[23] Nishide T, Yoneyama K, Ohta K. Attribute–Based encryption with partially hidden encryptor–specified access structures. In:Bellovin SM, Gennaro R, Keromytis A, Yung M, eds. Proc. of the Applied Cryptography and Network Security. Berlin, Heidelberg:Springer–Verlag, 2008. 111—129. [doi: 10.1007/978–3–540–68914–0_7]

[24] Emura K, Miyaji A, Nomura A, Omote K, Soshi M. A ciphertext–policy attribute–based encryption scheme with constant ciphertextlength. In: Bao F, Li H, Wang G, eds. Proc. of the InformationSecurity Practice and Experience (ISPEC 2009). Berlin, Heidelberg:Springer–Verlag, 2009. 13—23. [doi: 10.1007/978–3–642–00843–6_2]

[25] Canetti R, Halevi S, Katz J. Chosen–Ciphertext security from identity–based encryption. In: Advances in Cryptology–EUROCRYPT2004. Berlin, Heidelberg: Springer–Verlag, 2004. 207—222.

[26] Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data. In: Proc. of the Theory of Cryptography Conf.(TCC). Berlin, Heidelberg: Spring–er–Verlag, 2007. 535—554. [doi: 10.1007/978–3–540–70936–7_29]

[27] Goyal V, Jain A, Pandey O, Sahai A. Bounded ciphertext policy attribute based encryption. In: Aceto L, Damgård I, Goldberg LA,Halldórsson M M, Ingólfsdóttir A, Walukiewicz I, eds. Proc. of the ICALP 2008. Berlin, Heidelberg: Springer–Verlag, 2008.579—591. [doi: 10.1007/978–3–540–70583–3_47]

[28] Liang XH, Cao ZF, Lin H, Xing DS. Provably secure and efficient bounded ciphertext policy attribute based encryption. In: Proc.of the ASIAN ACM Symp. on

Information, Computer and Communications Security (ASIACCS 2009). New York: ACM Press,2009. 343—352. [doi: 10.1145/1533057.1533102]

[29] Ibraimi L, Tang Q, Hartel P, Jonker W. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In: Proc.of the Information Security Practice and Experience. Berlin, Heidelberg: Springer-Verlag, 2009. 1—12. [doi: 10.1007/978-3-642-00843-6_1]

[30] Waters B. Ciphertext-Policy attribute-based encryption: An expressive, efficient, and provably secure realization. http://eprint.iacr.org/2008/290.pdf [doi: 10.1007/978-3-642-19379-8_4]

[31] Attrapadung N, Imai H. Conjunctive broadcast and attribute-based encryption. In: Shacham H, Waters B, eds. Proc. of thePairing-Based Cryptography-Pairing 2009. Berlin, Heidelberg: Springer-Verlag, 2009. 248—265. [doi: 10.1007/978-3-642-03298-1_16]

[32] Lewko A, Okamoto T, Sahai A, Takashima K, Waters B. Fully secure functional encryption: Attribute-Based encryption and(hierarchical) inner product encryption. In: Advances in Cryptology-EUROCRYPT 2010. LNCS 6110, Berlin, Heidelberg:Springer-Verlag, 2010. 62—91. [doi: 10.1007/978-3-642-13190-5_4]

[33] Waters B. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In: Halevi S, ed. Advances inCryptology-CRYTO 2009. Berlin, Heidelberg: Springer-Verlag, 2009. 619—636. [doi: 10.1007/978-3-642-03356-8_36]

[34] Lewko A, Waters B. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In: Proc. of the 7thTheory of Cryptography Conf. (TCC 2010). Berlin, Heidelberg: Springer-Verlag, 2010. 455—479. [doi: 10.1007/978-3-642-11799-2_27]

[35] Attrapadung N, Imai H. Attribute-Based encryption supporting direct/indirect revocation modes. In: Parker MG, ed. Proc. of theCryptography and Coding 2009. Berlin, Heidelberg: Springer-Verlag, 2009. 278—300. [doi: 10.1007/978-3-642-10868-6_17]

[36] Boldyreva A, Goyal V, Kumar V. Identity-Based encryption with efficient revocation. In: Proc. of the ACM Conf. on Computerand Communications Security. New York: ACM Press, 2008. 417—426. [doi: 10.1145/1455770.1455823]

[37] Ibraimi L, Petkovic M, Nikova S, Hartel P, Jonker W. Mediated ciphertext-policy attribute-based encryption and its application. In:Proc. of the 10th Int'l Workshop on Information Security Applications-WISA 2009. LNCS 5932, Berlin, Heidelberg:Springer-Verlag, 2009. 309—323. [doi: 10.1007/978-3-642-10838-9_23]

[38] Yu SC, Wang C, Ren K, Lou WJ. Attribute based data sharing with attribute revocation. In: Proc. of the ASIAN ACM Conf. onComputer and Communications Security (ASIACCS 2010). New York: ACM Press, 2010. 261—270. [doi: 10.1145/1755688.1755720]

[39] Li J, Ren K, Kim K. A2BE: Accountable attribute-based encryption for abuse free access control. http://eprint.iacr.org/2009/118.pdf

[40] Li J, Ren K, Zhu B, Wan ZG. Privacy-Aware attribute-based encryption with user accountability. In: Proc. of the InformationSecurity Conf. 2009. LNCS 5735, Berlin, Heidelberg: Springer-Verlag, 2009. 347—362. [doi: 10.1007/978-3-642-04474-8_28]

[41] Yu SC, Ren K, Lou WJ, Li J. Defending against key abuse attacks in KP-ABE enabled broadcast systems. In: Proc. of the Securityand Privacy in Communication Networks. Berlin, Heidelberg: Springer-Verlag, 2009. 311—329. [doi: 10.1007/978-3-642-05284-2_18]

[42] Boneh D, Sahai A, Waters B. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay S,ed.Advances in Cryptology-EUROCRYPT 2006. LNCS 4004, Berlin, Heidelberg: Springer-Verlag, 2006. 573—592. [doi:10.1007/11761679_34]

[43] Chase M. Multi-Authority attribute based encryption. In: Proc. of the Theory of Cryptography Conf. (TCC). Berlin, Heidelberg:Springer-Verlag, 2007. 515—534.

[44] Božović V, Socek D, Steinwandt R, Villányi VI. Multi-Authority attribute based encryption with honest-but-curious centralauthority 2009. http://eprint.iacr.org/2009/083.pdf

[45] Lin H, Cao ZF, Liang X, Shao J. Secure threshold multi authority attribute based encryption without a central authority. In:Chowdhury DR, Rijmen V, Das A, eds. Proc. of the Cryptology in India-INDOCRYPT 2008. Berlin, Heidelberg: Springer-Verlag,2008. 426—436. [doi: 10.1007/978-3-540-89754-5_33]

[46] Chase M, Chow SSM. Improving privacy and security in multi-authority attribute-based encryption. In: Proc. of the ACM Conf. onComputer and Communications Security. New York: ACM Press, 2009. 121—130. [doi: 10.1145/1653662.1653678]

[47] Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure distributed key generation for discrete-log based cryptosystems. Journal ofCryptology, 2007,20(1):51—83. [doi: 10.1007/s00145-006-0347-3]

[48] Goyal V, Lu S, Sahai A, Waters B. Black-Box accountable authority identity-based encryption. In: Proc. of the ACM Conf. onComputer and Communications Security. New York: ACM Press, 2008. 427—436. [doi: 10.1145/1455770.1455824]

[49] Kapadia A, Tsang PP, Smith SW. Attribute-Based publishing with hidden credentials and hidden policies. In: Proc. of the 14thAnnual Network and Distributed System Security Symp. (NDSS 2007). USENIX Association, 2007. 179—192.

[50] Li J, Wang Q, Wang C, Ren K. Enhancing attribute-based encryption with attribute hierarchy. In: Proc. of the Mobile Networks andApplications. Berlin, Heidelberg: Springer-Verlag, 2010. 1—9. [doi: 10.1007/s11036-010-0233-y]

[51] Agrawal S, Boneh D, Boyen X. Efficient lattice (H) IBE in the standard model. In: Gilbert H, ed. Advances in Cryptology-EUROCRYPT 2010. Berlin, Heidelberg: Springer-Verlag, 2010. 553—572.

[52] Boneh D, Boyen X, Goh EJ. Hierarchical identity based encryption with constant size ciphertext. In: Cramer R, ed. Advances inCryptology-EUROCRYPT 2005. Berlin, Heidelberg: Springer-Verlag, 2005. 440—456. [doi: 10.1007/11426639_26]

[53] Boyen X, Waters B. Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork C, ed. Advances inCryptology-CRYPTO 2006. Berlin, Heidelberg: Springer-Verlag, 2006. 290—307. [doi: 10.1007/11818175_17]

[54] Horwitz J, Lynn B. Toward hierarchical identity-based encryption. In: Proc. of the Theory and Applications of CryptographicTechniques. Berlin, Heidelberg: Springer-Verlag, 2002. 466—481.

[55] Yao DF, Fazio N, Dodis Y, Lysyanskaya A. Id-Based encryption for complex hierarchies with applications to forward security andbroadcast encryption. In: Proc. of the ACM Conf. on Computer and Communications Security. New York: ACM Press, 2004.354—363. [doi: 10.1145/1030083.1030130]

[56] Attrapadung N, Imai H. Dual-Policy attribute based encryption. In: Abdalla M, Pointcheval D, Fouque P A, Vergnaud D, eds. Proc.of the Applied Cryptography

and Network Security. Berlin, Heidelberg: Springer-Verlag, 2009. 168—185. [doi: 10.1007/978-3-642-01957-9_11]

数据库与存储参考文献

[1] Gray J. What next? A few remaining problems in information technology. 1998. http://research.microsoft.com/~gray/talks/ Gray_Turing_FCRC.pdf

[2] Welch B, Unangst M, Abbasi Z, Gibson G. Scalable performance of the Panasas parallel file system. In: Baker M, ed. Proc. of the 2008 Conf. on File and Storage Technologies (FAST 2008). San Jose: USENIX, 2008. 17—33.

[3] Karger D, Lehman E, Leighton T, Levine M, Lewin D. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proc. of the 29th Annual ACM Symp. on Theory of Computing (STOC'97). El Paso: ACM Press, 1997. 654—663.

[4] Tang H, Gulbeden A, Zhou JY, Strathearn W, Yang T, Chu LK. A self-organizing storage cluster for parallel data-intensive applications. In: Huskamp JC, ed. Proc. of the 2004 ACM/IEEE Conf. on Supercomputing (SC 2004). Pittsburgh: IEEE Computer Society, 2004. 52—63.

[5] Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. In: Cruz R, ed. Proc. of the Annual Conf. of the Special Interest Group on Data Communication (SIGCOMM 2001). San Diego: ACM Press, 2001. 149—160.

[6] Brinkmann A, Salzwedel K, Scheideler C. Efficient, distributed data placement strategies for storage area networks. In: Gary M, ed. Proc. of the 12th ACM Symp. on Parallel Algorithms and Architectures. Bar Harbor: ACM Press, 2000. 119—128.

[7] Brinkmann A, Salzwedel K, Scheideler C. Compact, adaptive placement schemes for non-uniform distribution requirements. In: Maggs B, ed. Proc. of the 14th ACM Symp. on Parallel Algorithms and Architectures. Winnipeg: ACM Press, 2002. 53—62.

[8] Schindelhauer C, Schomaker G. Weighted distributed hash tables. In: Paul S, ed. Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA 2005). Las Vegas: ACM Press, 2005. 218—227.

[9] Honicky RJ, Miller EL. A fast algorithm for online placement and reorganization of replicated data. In: Dongarra J, ed. Proc. of the 17th Int'l Parallel & Distributed Processing Symp. (IPDPS 2003). Nice: IEEE Computer Society, 2003.

[10] Honicky RJ, Miller EL. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In: Bader DA, ed. Proc. of the 18th Int'l Parallel & Distributed Processing Symp. (IPDPS 2004). Santa Fe: IEEE Computer Society, 2004.

[11] Weil SA, Brandt SA, Miller EL, Maltzahn C. CRUSH: Controlled, scalable and decentralized placement of replicated data. In: Miller BH, ed. Proc. of the 2006 ACM/IEEE Conf. on Supercomputing. Tampa: IEEE Computer Society, 2006.

[12] Weil SA, Brandt SA, Miller EL, Long DDE, Maltzahn C. Ceph: A scalable, high-performance distributed file system. In: Bershad B, ed. Proc. of the 7th Symp. on Operating Systems Design and Implementation. Seattle: USENIX, 2006. 307—320.

[13] Gobioff H, Gibson G, Tygar D. Security for network attached storage devices. Technical Report, TRCMU-CS-97-185, Pittsburgh: Carniege Mellon University, 1997.

[14] Sun Microsystems, Inc. Lustre file system: High-Performance storage architecture and scalable cluster file system. 2007. https://www.sun.com/offers/docs/LustreFileSystem.pdf

[15] Nagle D, Serenyi D, Matthews A. The Panasas ActiveScale storage cluster—Delivering scalable high bandwidth storage. In: Huskamp JC, ed. Proc. of the 2004 ACM/IEEE Conf. on Supercomputing. Pittsburgh: IEEE Computer Society, 2004.

[16] Schmuck F, Haskin R. GPFS: A shareddisk file system for large computing clusters. In: Long D, ed. Proc. of the 2002 Conf. on File and Storage Technologies (FAST 2002). Monterey: USENIX, 2002. 231—244.

[17] Liu Z, Zhou XM. A data object placement algorithm based on dynamic interval mapping. Journal of Software, 2005,16(11): 1886—1893 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1886.htm [doi: 10.1360/jos161886]

[18] Wang D, Shu JW, Xue W, Shen MM. Self-Adaptive hierarchical storage management in SAN based on block-level. Advanced Technology Communication, 2007,17(2):111—115 (in Chinese with English abstract).

[19] Wang F, Zhang SD, Feng D, Zeng LF. Hybrid object allocation policy for object storage systems. Journal of Huazhong University of Science & Technology (Nature Science Edition), 2007,35(3):46—48 (in Chinese with English abstract).

[20] Tan HF, Sun LL, Hou ZF. An effective algorithm of data placement in a mass storage system. Computer Engineering, 2006,32(10): 47—49 (in Chinese with English abstract).

[21] Brinkmann A, Effert S, Heide FMAD, Scheideler C. Dynamic and redundant data placement. In: Abdelrahman TS, ed. Proc. of the 27th Int'l Conf. on Distributed Computing Systems. Toronto: IEEE Computer Society, 2007.

社区治理参考文献

[1] Xuan Q, Gharehyazie M, Devanbu P T, et al. Measuring the Effect of Social Communications on Individual Working Rhythms: A Case Study of Open Source Software. In Proceedings of the 2012 IEEE International Conference on Social Informatics (Washington, DC, USA, 2012), pp. 78-85. [doi: 10.1109/SocialInformatics.2012.17]

[2] Ye Y, Kishida K. Toward an understanding of the motivation Open Source Software developers. In Proceedings of the 25th International Conference on Software Engineering (Portland, OR, USA, 2003), pp. 419-429. [doi:10.1109/ICSE.2003.1201220]

[3] Sowe S K, Stamelos I, Angelis L. Understanding knowledge sharing activities in free/open source software projects: An empirical study. Journal of Systems and Software, 2008, 81(3):431–446. [doi:10.1016/j.jss.2007.03.086]
[4] Bachmann A, Bernstein A. When process data quality affects the number of bugs: Correlations in software engineering datasets. In Proceedings of the IEEE Working Conference on Mining Software Repositories (Cape Town, South Africa, 2010), pp. 62–71. [doi: 10.1109/MSR.2010.5463286]
[5] Bird C, Gourley A, Devanbu P. Detecting Patch Submission and Acceptance in OSS Projects. In Proceedings of the 4th International Workshop on Mining Software Repositories (Minneapolis, USA, 2007), pp. 26–26. [doi:10.1109/MSR.2007.6]
[6] Jensen C, Scacchi W. Role Migration and Advancement Processes in OSSD Projects:A Comparative Case Study. In Proceedings of the 29th International Conference on Software Engineering (Minneapolis, USA, 2007). pp. 364–374.
[7] Nakakoji K, Yamamoto Y, Nishinaka Y, et al. Evolution patterns of open–source software systems and communities. In P roceedings of the 14th International Workshop on Principles of Software Evolution, 2002. 76–85. [doi:10.1145/512035.512055]
[8] Crowston K, Wei K, Li Q, et al. Core and Periphery in Free/Libre and Open Source Software Team Communications. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006:118a–124a. [doi:10.1109/HICSS.2006.101]
[9] Mockus A, Fielding R T, Herbsleb J D. Two case studies of open source software development: Apache and Mozilla. Acm Transactions on Software Engineering and Methodology, 2002, 11(3):309–346. [doi:10.1145/567793.567795]
[10] Terceiro A, Rios L R, Chavez C. An Empirical Study on the Structural Complexity Introduced by Core and Peripheral Developers in Free Software Projects. In Proceedings of Software Engineering Brazilian Symposium on Software Engineering, IEEE Computer Society, 2010:21–29. [doi:10.1145/567793.567795]
[11] Cataldo M, Herbsleb J D. Communication networks in geographically distributed software development. ACM Conference on Computer Supported Cooperative Work. 2008:579–588. [doi:10.1145/1460563.1460654]

附录一：Event Souring

We can query an application's state to find out the current state of the world, and this answers many questions. However there are times when we don't just want to see where we are, we also want to know how we got there.

Event Sourcing ensures that all changes to application state are stored as a sequence of events. Not just can we query these events, we can also use the event log to reconstruct past states, and as a foundation to automatically adjust the state to cope with retroactive changes.

The fundamental idea of Event Sourcing is that of ensuring every change to the state of an application is captured in an event object, and that these event objects are themselves stored in the sequence they were applied for the same lifetime as the application state itself.

Let's consider a simple example to do with shipping notifications. In this example we have many ships on the high seas, and we need to know where they are. A simple way to do this is to have a tracking application with methods to allow us to tell when a ship arrives or leaves at a port.

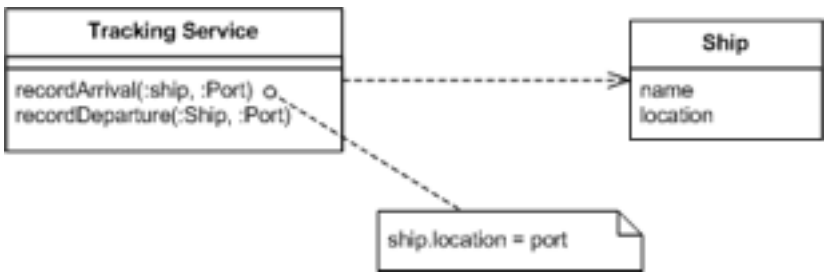


Figure 1: A simple interface for tracking shipping movements.

In this case when the service is called, it finds the relevant ship and updates its location. The ship objects record the current known state of the ships. Introducing Event Sourcing adds a step to this process. Now the service creates an event object to record the change and processes it to update the ship.

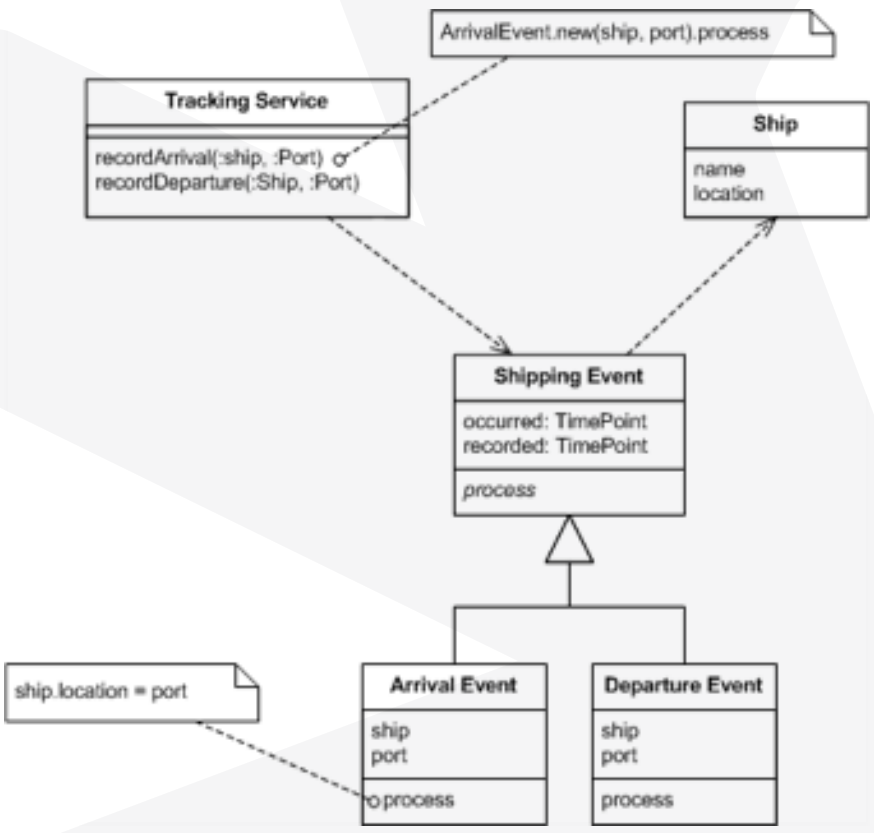


Figure 2: Using an event to capture the change.

Looking at just the processing, this is just an unnecessary level of indirection. The interesting difference is when we look at what persists in the application after a few changes. Let's imagine some simple changes:

- The Ship 'King Roy' departs San Francisco
- The Ship 'Prince Trevor' arrives at Los Angeles
- The Ship 'King Roy' arrives in Hong Kong

With the basic service, we see just the final state captured by the ship objects. I'll refer to this as the application state.

Figure 3: State after a few movements tracked by simple tracker.

With Event Sourcing we also capture each event. If we are using a persistent store the events will be persisted just the same as the ship objects are. I find it useful to say that we are persisting two different things an application state and an event log.

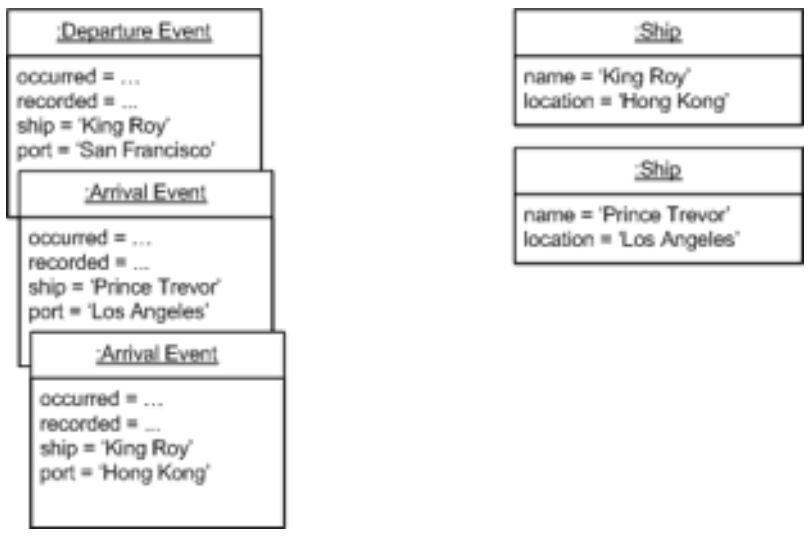


Figure 4: State after a few movements tracked by event sourced tracker.

The most obvious thing we've gained by using Event Sourcing is that we now have a log of all the changes. Not just can we see where each ship is, we can see where it's been. However this is a small gain. We could also do this by keeping a history of past ports in the ship object, or by writing to a log file whenever a ship moves. Both of these can give us an adequate history.

- The key to Event Sourcing is that we guarantee that all changes to the domain objects are initiated by the event objects. This leads to a number of facilities that can be built on top of the event log:
  Complete Rebuild: We can discard the application state completely and rebuild it by re–running the events from the event log on an empty application.

- Temporal Query: We can determine the application state at any point in time. Notionally we do this by starting with a blank state and rerunning the events up to a particular time or event. We can take this further by considering multiple time–lines (analogous to branching in a version control system).

- Event Replay: If we find a past event was incorrect, we can compute the consequences by reversing it and later events and then replaying the new event and later events. (Or indeed by throwing away the application state and replaying all events with the correct event in sequence.) The same technique can handle events received in the wrong sequence – a common problem with systems that communicate with asynchronous messaging.

A common example of an application that uses Event Sourcing is a version control system. Such a system uses temporal queries quite often. Subversion uses com–plete rebuilds whenever you use dump and restore to move stuff between repository files.

开 启 未 来 存 储

Lambda

lambda.im