深度信息技术（精品）专辑

第一期

2021.3.9

- 工业互联网
- 区块链
- 云原生+KUBE编排
- 自动驾驶与无人驾驶
- 智能家居生态系统
- 人工智能医疗影像系统
- 自主操作系统和自建生态
  （含自主CPU/SoC芯片）阶段成果

中国开源软件推进联盟
China OSS Promotion Union

目录

■ 工业互联网

■ 区块链

■ 云原生+KUBE 编排

[1] 注：COPU 编者为陈伟、鞠东颖。

■ 自动驾驶与无人驾驶

■ 智能家居生态系统

■ 人工智能医疗影像系统

■ 自主操作系统和自建生态

工业互联网

现代创新引擎：“互联网+基于知识社会的创新 2.0”

“互联网+创新 2.0”体现工业互联网的创新机制

“互联网+创新 2.0”与“工业 4.0”、“工业互联网”任务相同、机制相通

陆首群

2015.10.11

现在的问题是：如何促使在目前现实的工业社会中的“传统业态”实行“业态转型”或“业态提升”？我们可在工业社会这个几乎无限空间中划出一个用以考察的有限的物理空间，在物理空间中考察“业态转型”，这个物理空间（Physical Space）简称“物空”或“实空”或 P 空间。“传统业态”是什么？“传统业态”指传统工业业态，可分为生产、经济、社会的三种业态，传统的生产业态指工业生产方式（或工业产品、工业系统），传统的经济业态指工业经济或市场经济，而工业城市可看成为一种传统的社会业态。

只有创新动能才能促使“业态转型”，而只有采用比“工业社会”高出一个时代差的“高阶社会”中的创新动能，才能促使“传统业态”实行“0→1”颠覆性的转型。什么是“高阶社会”？相对于工业社会而言指信息社会或知识社会。但在目前的现实世界中，总体上尚不存在信息社会或知识社会，我们只能构建一个虚拟化的“数字网络空间”，在其中影射知识社会的场景，这个数字网络空间（Cyber Space）简称“数空”或“虚空”或 C 空间。

以创新促业态转型的机制如下所述：

在虚空（C 空间）中架构以现代互联网（≥Web2.0）为载体，以信息、知识为资源（高于工业社会较低层次的人力、自然资源），以深度信息技术（云、物、社、移、大、智、区、5G、AR/VR、量…）和适配先进管理为作用力,而由其中的互联网载体+知识资源+信息技术+适配管理综合构成的创新动能，驱动“传统业态”实行“0→1”的转型。其操作程序为：将“虚空”与“物空”对接，在经历碰撞、交互、融合过程后，以“虚空”中的创新动能作用于“物空”中的“传统业态”，催生其“0→1”的转型，以重构新业态（如智能生产方式，或新经济/数字经济，或智慧城市，这里的新经济指工业经济向数字经济转型时的过渡经济形态）。上述机制可概括为“互联网+创新 2.0+传统业态以重构新业态。

基于知识社会创新 2.0 是基于工业社会创新 1.0 的升级版。

如上所述，谈的是 P+C 二元空间，但在贯彻“互联网+创新 2.0”时要以人为本（Human，H），H 包括人和人的关系和互动，以及人和物的关系，如此说来应是 P+C+H 融合互动的三元空间。采用“互联网+基于知识社会创新 2.0”在三元空间中互动、融合、促进、创新。

“互联网+创新 2.0”与德国人在 2011 年提出的“工业（制造）4.0(战略)”，及美国人（GE 公司）于 2013 年提出的“工业互联网”，任务相同、机制相通，但“工业 4.0”侧重于智能制造(智能产品、智能工厂、智能制造)，“工业互联网”侧重于重构智能化的工业体系（即数字化、网络化、智能化的工业体系），而“互联网+创新 2.0”则机制更强，涉及面更宽、创新力度更大。

华为工业互联网实践

华为技术有限公司

一、产业趋势

在经历了机械化和电气化革命之后，制造业正在朝着数字化和智能化的方向飞速发展。工业互联网是新一代 ICT 技术与制造业深度融合的产物。工业互联网以数据为核心要素实现全面连接，构建起全要素、全产业链、全价值链融合的新制造体系和新产业生态，是数字化转型的关键支撑和重要途径。

工业转型契机：当前，全球工业互联网正处在新产业格局发展的关键期和规模化扩张的窗口期，我国在体制和市场层面具备优势。

智能优化决策：工业互联网以数据闭环为核心，通过对物理资产的全面深度感知，实现海量工业数据的高效集成与管理，开展各类工业模型与数据模型的构建与分析，形成优化决策并反馈至物理系统，这是工业互联网最基本的模式和方法，也是驱动制造业智能化转型的关键

提质降本增效：推进工业互联网需要理解和遵循工业的规律，真正以业务场景为驱动，始终关注提质、降本、增效等业界核心诉求，给客户带来切实的业务价值。

工业互联网通过人、机、物的全面互联，构建起“全要素、全产业链、全价值链” 全面连接的新型工业生产制造和服务体系，是促进传统产业转型升级、实现经济高质量发展的重要驱动力。其核心包括两部分：

1) 工业园区内网及外网的网络基础设施，支撑工业制造过程中的设计流、生产流、物资流、资金流等不同流程的协同，发挥基础设施带来的效率提升作用以及工作协同开放创新的支撑作用。
2) 新兴业态与应用，包括工业平台的创新及应用的创新。工业平台的创新围绕运营优化，资源协同及模式创新。应用创新主要实现智能化生产、网络化协同、个性化定制、服务化延伸，进一步提升从产品生产过程到服务过程的全面创新力度。

二、华为实践

当前工业领域在实现数字化转型中，面临着现实的业务挑战：

业务系统割裂、数据难以汇聚：企业的业务系统建设多数是项目驱动的，企业在生产和经营的各个环节存在大量业务孤岛，同时数据源种类多样，包

括设备数据、应用数据、视频等媒体数据。因而，企业的数据采集和汇聚，是一个基础且普遍的难题。

业务架构滞后、业务难以演进：工业企业普遍采用 ISA-95 架构是上世纪 90 年代的产物，在全要素感知、业务集成、智能决策、灵活扩展等方面存在明显短板。过去 10 年 ICT 技术取得了突破性的进展。如何进一步实现 IT/CT/OT 的深度融合，构建一套支撑工业互联网转型升级的开放、可扩展架构，是业界持续在思考的方向。

业务建模困难、缺乏有效的支撑工具：工业软件的核心是工业知识长期积累、沉淀并在应用中迭代的模型化、软件化产物。我国的工业企业，虽然也掌握了一定程度的工业知识，但由于缺乏合适的平台、工具去实现从“工业知识”到“工业模型”及“工业软件”的沉淀，从而多采用项目级的定制交付，没有实现产品化。

企业有转型诉求、缺乏所需的人才和技能：工业互联网转型是个复合型的系统工程，需要大量既精通工业技术又懂信息技术，同时又能够实现跨界融合创新的人才。单独依赖工业企业自身招聘，既难以招到也难以留住合适的人才。因而，工业企业与 ICT 企业及其生态链需要进一步加强合作，发挥各自所长，实现共赢。

华为经过在电子行业近 30 年的发展，在产品研发、生产、销售、服务以及供应链等整个制造业生命周期各环节积累了大量的经验，在提供使能数字化技术与方案的同时，已经在自身数字化和智能化的道路上深耕多年，通过华为的生产实践，华为工业互联网平台 FusionPlant，包含联接管理平台、工业智能体、工业应用平台三大部分。定位于做企业增量的智能决策系统，实现业务在云上敏捷开发，边缘可信运行。赋能行业合作伙伴深耕工业核心业务流，持续释放潜在业务价值。

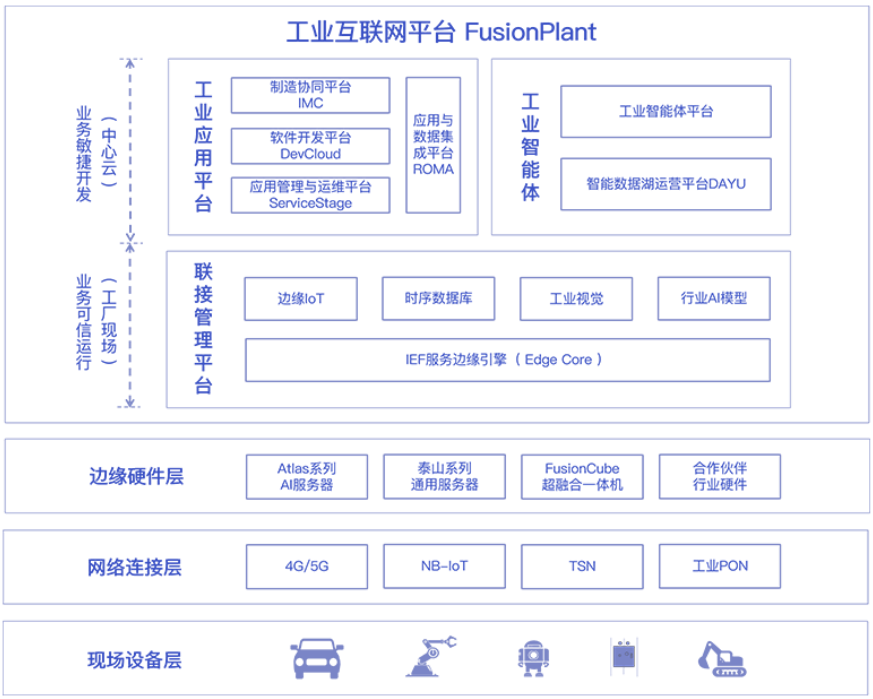


Figure 1 华为工业互联网平台

联接管理是工业领域的基础，为生产现场数据采集层、面向制造企业的设备运行监控提供稳定的网络联接。利用先进的 IOT 和通信技术，实现设备运行数据、设备生产数据、产品质量数据的全面和实时性采集，同时对生产管理系统的控制进行反馈，并通过现场处理单元提供现场的边缘计算能力，实现对现场数据的实时处理。

工业智能体服务作为一站式 AI 开发与运行、数据采集、治理与开发平台，为工业 AI 算法开发者提供低代码或拖拽式 AI 算法开发环境、丰富的行业机理模型和行业知识图谱；为工业项目交付者提供边缘设备管理、AI 模型部署的能力；为数采建模者提供拖拽式工厂物化建模，数据处理和监控告警规则的可视化组态；为数据运营者提供数据资产地图与治理能力；为数据开发者提供可视化作业开发和调度、数据服务化等能力，以支撑工业模型沉淀的能力。

工业应用平台围绕支撑与使能工业企业或者集成商开发工业应用软件，聚焦应用和数据连接，提供快速、简单的消息、数据、API、设备集成能力，简化企业上云，支持公有云到用户本地机房的应用集成，OT 与 ICT 集成。

华为工业互联网平台的架构重点结合工业企业应用场景及安全生产，有以下特点：

云网协同：依托华为的联接方案，如 5G、NB-IoT、TSN、工业 PON，满足企业在工厂内、外网络各类场景的连接需求。

边云协同：提供纳管海量分布式边缘节点，并能够将云上的丰富的服务、生态伙伴和客户的应用部署到边缘节点运行的能力。

工业数据湖：提供数据全生命周期一站式开发运营平台，提供数据集成、开发、治理、服务等功能，帮助企业快速构建全局数据运营能力。

全栈 AI：提供从芯片、硬件、算子库、AI 框架、通用 AI 开发平台、工业智能体的全栈方案，实现协同优化和多点创新。

安全可信：华为工业互联网整体方案在设备认证、隐私数据加密保护、业务运行的可靠性和可用性、安全启动等多个方面提供全方位保障。

三、开源开放

开源已成为当代全球的一种创新和协作模式，华为在发展工业互联网过程中拥抱开源。开源降低了学习者的准入门槛，客观上给入门者提供了快速学习的机会，有利于知识更高效率的重新利用。面向工业互联网，依托开源手段，团结更广泛的开发者，促进工业软件能力的增强以及工业制造过程问题的创新性解决，实现供应链、价值链和商业生态的和谐构建。

1、基础软件开源

华为通过自身的实践，深刻体会到工业互联网本质是 IT 和 OT 的深度结合，在这当中需要全新的软件思维和架构。通过生产资源关键要素数据资源的汇聚，抽象提炼建立可运营的数据平台，同时从企业运营角度建立企业产、销、服的管理数据平台，通过 OT 和 IT 两者的充分结合，建立可互操作、可分布式扩展以及数据分析的工业软件体系架构。基于开源技术搭建工业互联网平台和应用在当前成为主流，开源 PaaS 架构已成为平台使能框架的共性选择，多数平台采用 Hadoop、Spark 等开源数据工具支撑数据服务，以及利用多种开源的开发工具构建开发环境。

华为在组织自身生产的同时，逐步构建起工业互联网端边云平台底座，这个过程中，也经历了从使用开源到回馈开源的过程，并逐步将自身的一些实践通过开源方式开放出来，坚持通过开源推动开放式创新。

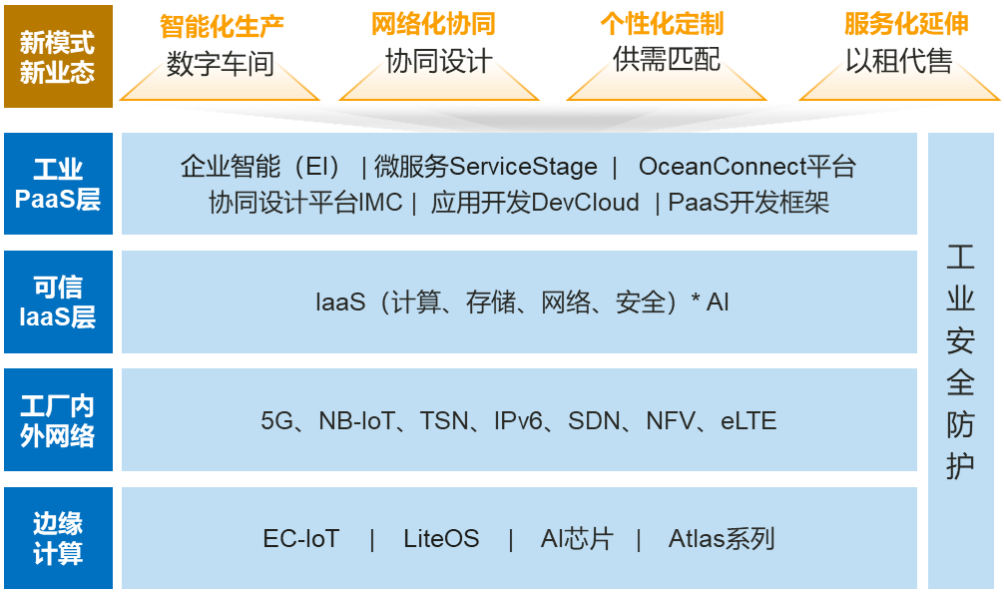


Figure 2 工业互联网基本框架

面向全场景智慧终端的移动系统领域，推出创新的分布式操作系统 OpenHarmony 项目，实现跨终端全场景无缝协同以及一次开发多终端部署的便捷体验；在基础计算领域，先后开源服务器操作系统 openEuler、数据库 openGauss、大数据 openLookeng、AI 框架 MindSpore 等项目，打造异构融合和灵活演进的开放创新 IT 基础设施体系，在云服务领域，贡献云原生 KubeEdge 和 Volcano 项目，构建云边协同和高性能异构计算调度架构，支持场景化行业应用高效构建，同时，面向微服务领域开源的 ServiceComb 也成为 Apache 基金会孵化的微服务框架；在通信领域，开源 5G 边缘计算原生的 EdgeGallery 项目，构建 MEC 边缘的资源、应用、安全、管理高效协同的基础框架，加速网络服务开放及创新应用构建。

面向工业制造领域，这些开源项目以及基于其构建的商业解决方案已广泛应用于石化、制造、煤炭、环保、水利、磨具等企业应用中。

2、项目案例

工业制造领域如何有效管理遍布各处的传感器及 OS 异构的边缘设备，解决海量边缘应用接入及高效运维、升级等始终是很有挑战的问题。华为基于云原生技术构建的边云协同操作系统可运行在多种边缘设备上，将华为云丰富的 AI、IoT 及数据分析等智能应用以轻量化的方式从云端部署到边缘，同时实现了软件和算法的持续升级迭代。与此同时，为有效帮助行业应对共同挑战，华为将该边云系统操作系统内核 KubeEdge 开源，贡献给 CNCF 基金会，这是边缘计算领域热度最高的开源项目之一。目前已有 500 多位核心贡献者，已在多个行业广泛应用。

以煤炭行业为例，煤矿安全长期以来都是全民关注的重点社会问题，而导致矿难产生的主要原因是人的不安全行为、设备的不安全状态及环境的不安全因素等。即便经过多次升级优化，当下的煤炭行业仍存在水害问题突出、瓦斯防治任重道远、上覆岩层失稳存在重大隐患、皮带运输效率有待提升等诸多问题。华为与煤炭行业合作伙伴一起，打造云边协同的智慧解决方案，方案包括了矿山安全态势感知与信息共享体系化协同模型，设计、实施、评测一体化智能监控平台，视觉、语音、OCR 多维度作业场景分析模型，层级职能部门联合执行异常事件联动与处置机制。方案依托合作伙伴在该行业中的深厚积累，融合各方优势，运用人工智能、物联网、大数据、云计算等技术，为煤炭行业安全生产中的标准化、体系化、技管结合、预警与应急响应等重大科学与工程问题提供解决方案，并通过“云边端”协同实现态势感知，辅助有关部门优化政策的定制和监管。以探水作业为例，探水作业的好坏，不仅直接关系到探水人员的安全，而且影响到探放水周围地区甚至整个矿井的安全。但由于探水作业工程资料普遍纸质化，探水钻孔深度谎报无有效监督手段，监管部门无法及时掌握探水作业情况等原因，导致探水作业假探、少探、漏探甚至不探行为难以杜绝，存在严重的水害隐患。为此，基于人工智能视频识别技术，华为云通过提供边缘计算 IEF 服务、云上数据服务、一站式 AI 开发平台 ModelArts，实现对探水作业全过程监控管理，自动识别卸杆数量，统计钻孔深度，防止工程资料造假，杜绝煤矿探水作业假探、漏探、少探。在云侧，人工智能识别模型云端管理，保障煤矿识别模型实时更新；部署高并发、低延时 SAAS 系统，通过移动终端实现井下视频实时调阅，探水作业数据查询、审批指令“云、边”同步；在边侧部署 AI 服务器，云端人工智能模型同步更新，用于煤矿探水视频识别；部署探水管理应用系统，用于探水作业管理。在端侧，则通过矿用摄像机，实现井下探水作业智能识别，及数据回传。

三、未来展望

华为在工业互联网领域采用开放、合作、共赢的策略，汇聚各类合作伙伴，共同打造满足工业企业需求的工业互联网解决方案，提供工业互联网的应用和服务。始终坚持和行业伙伴、应用与服务伙伴，包括石化、汽车、冶金、工程机械、电力、电子制造、装备制造等各行各业，在全国各地开展工业互联网实践、共同服务各类工业企业，共同开发完成 2000 多个工业应用，助力 2 万家企业利用数字化技术优化生产流程并提升效率，为工业互联网落地作出扎实的贡献。

同时，华为将坚持基础软件开源开放的基本策略，从工业装备的设计研发、生产制造、消费流通、管理维护等全链全周期加强基础软件开源贡献，坚持“解决问题，创造价值”的开源理念，推进工业互联网开放式创新。

浪潮云洲工业互联网平台

浪潮集团

一、产业背景

工业互联网概念是在 2012 年由美国通用（GE）公司最先提出，之后在 2015 年德国西门子也推出了 Mindsphere 工业互联网平台，随后 PTC、AWS、微软、东芝等国外企业纷纷开展工业互联网平台建设。2017 年，国务院发布了《关于深化“互联网+先进制造业”发展工业互联网的指导意见》。之后，工业互联网作为我国新型基础设施关键组成部分，在推动工业技术改革创新、促进产业数字化转型、释放经济发展新动能等方面的基础性作用日益凸显。目前工业互联网已经成为国家战略，为双循环的经济格局发展带来了新机遇。

浪潮集团响应国家号召，建设的云洲工业互联网平台目前已经连续三年入选国家级跨行业、跨领域工业互联网平台。依托“云、QID、云 ERP”和智能制造方面的优势，浪潮逐渐形成了“云、数、用”一体化的工业互联网发展模式。浪潮云洲工业互联网全栈智能制造解决方案，为装备制造、离散制造、流程制造等企业提供一体化网络协同制造、智能数字工厂、高效设备运维、大规模个性化定制等服务。建立一个以企业高质量发展为高点、以安全生产为底线的可信赖的工业互联网平台，并以全要素质量保障体系为贯穿的工业互联网运营商，才能真正实现一业带百业。

开源软件吞噬世界，开源是软件技术发展的核心方向。相对于国外先进企业，我国企业在工业互联网领域发展起步相对较晚，而且从工业软件的积累上，始终未突破国外企业对核心技术的垄断。在这样的形势下，利用开源技术，发展开源软件，是我国工业互联网产业快速发展，实现弯道超车的必由之路。浪潮集团也高度重视开源软件技术，从平台建设开始就大量利用开源技术快速构建平台，在平台建设逐渐成熟，也将平台研发成果，作为开源项目赠送给开放原子基金会，并作为开源项目利用产业生态共同推动技术发展。

二、利用开源技术打造新型工业基础设施

工业互联网作为国家新型基础设施的主要代表之一，是中国制造业数字化、网络化、智能化转型的基石。发展以工业互联网为代表的新基建，促进产业数字化转型，首先就应尽快构建面向新型基础设施建设的核心技术平台。

2.1 构建全国最大分布式云推动企业上云服务

浪潮云依托在全国布局的 7+69 个云数据中心构建浪潮分布式云，依托浪潮云洲工业互联网 IOP 云基础平台形成了工业云的基本架构，拥有中心节点、区域节点、边缘节点，并通过统一的运维平台 Ops Center 进行统一的运营、治理、更新和演进，既能实现一体化的运行，满足工业数据的连接，对工业体系形成支撑，又能实现各节点相对独立地运行。

作为浪潮云的领航者，在浪潮云洲工业互联网 IOP 云基础平台的打造过程中，充分注重与开源的结合与应用，并基于开源社区推动平台在开放性和兼容性、创新性方面达到业界优秀水准。目前，浪潮云 IOP 平台已成功通过 CNCF Kubernetes 一致性认证，能为用户带来无缝、稳定的体验，并依托 IOP 平台研发实现了上百个独立灵活的微服务，服务开发维护复杂度降低 50%以上。

以 OpenStack 为核心的云平台，基于开源架构实现了从不同芯片架构（X86、ARM，MIPS、Alpha）到不同虚拟化技术（裸机、虚拟机、容器），以及不同应用场景（边缘、云数智融合、混合云管理）的全面支持；平台四个方向都是开放标准的，包括北向被多个 PaaS 云管平台兼容，南向兼容主流存储及 SDN 厂商，东西向兼容第三方安全、灾备产品等，并提供标准开放 API。

一直以来浪潮也秉持“源于社区，贡献社区”思想，先后加入云计算大数据等国际主流开源社区，成为 OpenStack 、CNCF 等社区金牌会员，并持续激励研发人员投入社区，不断向社区输出贡献。在最新发布的连续两个版本（U 版本、V 版本），浪潮在社区贡献的 5 项关键指标中均位列中国第一。目前，浪潮在最新的 OpenStack V 版本中 COMMIT 数量和 Drafted BP 排名进入中国第一、全球 TOP5。浪潮工程师也在国内外各类会议上受邀参加 BUG FIX、SPEC 规范分享等议题，主导 OpenStack Cyborg、Nova 等核心研发项目的讨论。

工业互联网头部客户和区域公共服务平台建设中，云基础平台建设规模往往非常大，他们对于云平台规模性、健壮性有很高的要求，单一集群有的需达到 1000 台服务器，基于该需求洞察，浪潮持续打磨大规模云建设能力，目前已实现单一集群 1000 节点规模建设内容，在业界领先，也为这类行业头部客户大规模云建设提供了技术保证。

2.2 依托开源 ERP 服务中小企业数字化专型

工业互联网不仅推动了传统工业转型升级，实现各种资源更加优化配置，提升工业经济效益。而且加快新兴产业培育，催生智能化生产、网络化协同、服务化延伸、个性化定制的诸多新产业。对于中小企业而言，如何能够低成

本，快速实现数字化转型，提升制造业数字化、网络化、智能化发展水平，开展新模式新业态创新应用，是企业急需解决的核心问题。

Odoo 是全球最成功的开源云 ERP 厂商，为全球 410 多万用户提供 ERP 解决方案，其开源特性受到全球爱好者的追捧。浪潮作为亚洲 Odoo 代码量贡献第一、世界第二的企业，从 Odoo 8 版本发布时，浪潮就已经参与到 Odoo 的软件开发中。随着产品不断成熟，2018 年浪潮与 Odoo 合资合作推出新一代开源云 ERP 产品浪潮 insuite，重点面向中小企业提供 SaaS 服务。2020 年 10 月 Odoo 最新发布 Odoo14，在系统兼容性、平台功能以及销售、物流等应用功能上都有重大改进。浪潮为该版本软件贡献了 12%的代码，结合双方合作，浪潮 insuite 也成为 Odoo 在中国唯一的 SaaS 营销平台。Odoo 本身提供了免编程开发工具和 130 余种 SaaS 应用。开发者可通过开发工具完成应用研发并提交到 Odoo 市场进行销售，实现开发到应用服务全过程。

浪潮 inSuite 是一款全新的云 ERP 产品,是浪潮新一代互联网+时代的云 ERP，是基于 WEB2.0 与云计算技术的新时代企业信息化管理服务平台。浪潮围绕着“生态、人效、增长”，旨在帮助企业打造面向新时代的生态互联与协同平台，围绕生态中的每个角色，提供随手可得的云服务，最终实现敏捷协同，智慧运营的企业运营新境界。整个产品采用 SOA 架构，业务架构上贯穿流程驱动与角色驱动思想，结合中国管理模式与中国管理实践积累，精细化管理支持企业财务管理、供应链管理等核心应用。技术架构上该产品采用平台化构建，支持跨数据库应用，是国内第一款开源的云 ERP。浪潮积极开展开源 ERP 的应用建设推广，目前已经有超过 8000 家中小企业应用了浪潮 insuite 企业管理软件，赋能浪潮基础计算能力，推动社区和联盟实现微服务化、开源化、社区化，并积极向合作伙伴开放共享，共同提供覆盖 IaaS+PaaS+SaaS 领域的整体云解决方案。

三、面向企业数字化、网络化、智能化建设提供核心技术服务

3.1 面向新一代智能制造提供开源工业 PaaS 技术服务

随着工业 4.0 的发展，企业生产效率提升和业务模式变革的速度日益加快，如何快速、高效的开发工业软件，提升工业企业运行效率是工业互联网解决企业核心问题的关键。而要解决这个问题的关键是提供高效的工业互联网及工业软件开发平台。

而作为国内率先提出数字化转型战略的公司，从 2014 年至今，浪潮先后提出并实践“大数据重构企业智慧”、“互联网+企业”和“数字化转型成就智慧企业”等理念，一直在数字化转型之路上积极探索与实践，帮助企业打造了连接、共享、精细、智能的智慧企业。作为开源领域的积极推动者和重要贡献者，2018 年 6 月，由浪潮牵头发起的“中国开源工业 PaaS 联盟”

正式授权。为进一步推动中国开源工业 PaaS 产业发展，提升工业互联网核心能力，提高中国工业互联网产业的国际竞争力，浪潮还牵头成立了中国开源工业 PaaS 联盟，将围绕工业 PaaS 价值链，运用市场机制集聚创新资源，实现企业、大学和科研机构等在战略层面有效结合，共同推动中国开源工业 PaaS 事业发展。

在产品研发过程中，浪潮对标了国外十余种低代码开发平台。结合研究成果，浪潮基于云原生、前后端分离、领域驱动设计、跨平台等架构与设计理念，形成 UBML 低代码建模体系，实现了软件核心技术突破，并构建了对应的软件开发标准。浪潮 iGIX 核心低代码模型体系 UBML（Unified Business Modeling Language）项目成为首批捐赠给开放原子开源基金会的项目，并已通过基金会 TOC 技术委员会的评审，正式进入项目孵化阶段。

企业对底层的信息化架构提出了全新的要求。由于数字化转型涉及的技术复杂度呈指数级增加，很多大中型企业现有的 IT 架构同样也无法支撑企业有效的纳入新技术。根据 IDC 的统计显示，在很多企业里新技术与业务的真正结合需要三年的时间，而近三年新出现的技术就超过 10 种，导致现有 IT 部门无法支撑技术的快速爆发。UBML 是一种基于领域特定语言（Domain-Specific Language DSL）的、用于快速构建应用软件的低代码建模语言。内容包括模型标准及其默认实现、SDK、运行时框架等组件。UBML 定位于 APaaS （应用程序平台即服务）领域，是低代码开发平台（Low-Code-Development-Platform）的核心基础，致力于在低代码领域建立应用软件建模开发的事实标准。通过低代码开发，能够解决企业数字化转型过程中效率这个核心问题，是工业互联网应用的核心关键技术。

UBML 作为低代码开发平台的开发语言，是低代码开发平台的核心基础，包含开发语言无关性的建模标准（UBML-Standard），内置了基于 UBML 标准的全栈业务模型（UBML-Models），并提供了可与模型进行全生命周期交互的开发服务与套件（UBML-SDK）及支撑模型运行的运行时框架（UBML-Runtime）。目前开放的代码有建模标准（UBML-Standard）及 UBML-Models，包括面向后端开发的核心模型 BE（Business-Entity）、VO（View-Model）和服务模型中的 EAPI（External-API）。

UBML 的标准与内置模型解耦，标准具有开发语言无关性，可与各种领域标准（例如 OpenAPI、BPMN）进行集成与适配，模型的种类可基于 UBML 核心机制，按照行业类型或应用类型进行扩展定制，具有良好的开放性与扩展性。目前，业内的低代码平台主要分为两种模式：一种是基于引擎的解析型模式；一种是基于源代码生成的生成型模式。UBML 提供了同时支持上述两种模式的混合（Hybrid）模式。无论是解析型模式，还是生成型模式，两者均基于经典的模型驱动架构（MDA），以模型为核心，因此模型的丰富度与深度代表

着低代码平台的核心能力。UBML 提供了几十种涵盖了从前端到后端的全栈业务模型体系，将为低代码平台的建模与开发能力提供全面支撑。UBML 基于“模型即源码”的理念，将模型视作源代码进行工程化管理，可以与主流研发过程管理工具进行集成，支持 DevOps。此外，UBML 还提供统一的模型全生命周期管理能力。

浪潮 iGIX 企业数字化能力平台包含技术、数据、业务三大中台，基于云原生和微服务架构，融合弹性计算、智能物联、机器学习、认知服务等数字技术，提供低代码开发、DevOps、混合云集成、生态开放等应用创新加速能力；内置数据资产管理与丰富的数据服务，打破数据壁垒，全面整合企业数据资源，构建基于数据的创新能力；沉淀共享业务服务，构建基于大共享的业务服务能力。浪潮 iGIX 企业数字化能力平台，是浪潮云 ERP GS Cloud 的基础支撑平台，iGIX 结合浪潮实际情况，制定了 UBML（社区版）、Open iGIX（社区版）、iGIX（商业版）分步的开闭源结合的策略，旨在通过开源开放助力浪潮生态发展。浪潮 iGIX 企业数字化能力平台的推出，不仅是浪潮在云原生、微服务架构、容器、人工智能领域技术创新的“集大成者”，同时也是浪潮在企业数字化转型中所积累出来的丰富实战和落地经验的高度总结，更是撬动企业数字化转型的新支点。

3.2 云原生国产化开源分布式数据库

浪潮第二个贡献给开放原子基金会的开源项目是云溪数据库项目。云溪数据库是基于 Google Spaner 的分布式数据库，向云原生和分布式方向发展。目旨在通过发展开源社区和用户生态，打造成为国内主流的分布式数据库产品。推进数据库开源和基金会项目捐赠机制的良性循环。

云溪数据库是一个 NewSQL 数据库，更准确的说是一个云原生 HTAP 数据库。在工业互联网应用场景下，随着 5G 和 IOT 的蓬勃发展，数据量爆炸式增长，传统数据库力不从心，而 NoSQL 不能满足实时和事务需求。只有 NewSQL 能兼顾传统数据库和 NoSQL 之长，免除分库分表之烦恼，满足 OLTP、OLAP 等不同应用交换的需求，云原生结合无限水平扩展，高效建设完美数字世界。

“云溪”NewSQL 数据库定位于在线事务处理/在线分析处理（HTAP）的融合型数据库产品，实现了一键水平伸缩，强一致性的多副本数据安全、分布式事务、实时 OLAP 等重要特性。云溪数据库早期基于 CockroachDB19.1 版本，目前与 CockroachDB 已经完全分道扬镳。CockroachDB 在全对等，乐观锁，混合逻辑时钟，算子等方面满足不了物联时代众多场景要求。因此，云溪在架构上做了彻底的重构。“云溪”数据库采用无锁的两阶段提交事务模型实现了对并发操作的 ACID 事务支持，采用全球原子钟+区域事务表解决

多节点间事务原子性，既做到本地趋于事务的高性能，又能兼顾全球跨区域数据库的全球一致性读。

同时，“云溪”数据库提供传统关系型数据库 OLTP 的所有特性，并且改变了传统数据库集中存储的限制，采用分布式架构、无共享存储，从而实现了从计算层、存储层的可扩展性。在“云溪”数据库中，将用户库表数据按照/DB/Table/PrimaryKey 编码成 KV 对，数据划分成 64M 大小的分片（Range)。每个 Range 默认 3 副本存储，若少于一半的副本丢失，会自动在其他可用节点上补齐。根据不同的容灾等级，Range 数据多副本可配置为跨机器、跨数据中心、跨地域存储。

分布式事务需要全局时钟，“云溪”数据库采用混合逻辑时钟（HLC），这种方案不依赖中心节点，方便扩展，即使是跨地域部署也没有什么问题。HLC 由 WallTime 和 LogicTime 两部分组成，WallTime 为节点 n 当前已知的最大的物理时间，通过先判断 WallTime，再判断 LogicTime 确定两个事件的先后顺序。“云溪”数据库这种分布式、无锁事务的原子性策略，即保证了分布式系统下难以处理的分布式事务一致性问题，同时避免传统两阶段事务的繁琐，消除了两阶段锁，同时大大降低了事务提交和回滚的开销，可以大大提升分布式事务的吞吐能力。

目前，浪潮为“云溪”数据库单独成立了独立公司负责软件开源生态建设和核心技术研发，未来也将持续发力，推动开源技术在工业互联网相关领域应用。

四、总结

在工业互联网平台建设上，浪潮将坚定的走开放路线，推进分层解耦，持续夯实核心技术能力，通过平台发展生态，构建有竞争力的行业解决方案。管子有言，“夫轻重强弱之形，诸侯合则强，孤则弱”，为了达成战略愿景，不仅需要规划出可以一个打十个的有核心竞争力的产品，还要通过开源技术构建有生态优势的朋友圈，形成十个人围着打一个人的战略优势，未来浪潮工业互联网平台将致力于以产业互联为核心，以增强标识解析为抓手，以云计算、大数据、区块链、5G 等新兴技术为支撑，构建工业大数据服务体系；通过标准化引领，推进产业数字化、产业能力汇集和产业服务能力开放；通过工业互联网平台链接政府、企业、互联网等多种数据构造智慧企业大脑，通过打造智能制造、智能供应链、智能仓储，智能决策等智慧应用，实现智能化生产、网络化协同、个性化定制、 服务化延伸等新模式，助力企业实现融合创新、产业链协同、供应链效率提升，实现各地产业的高质量、可持续发展。

工业互联网：若干认识与思考

——在 2.26 工联智库成立大会上的报告

中国科学院 梅宏院士

提纲

- 众说纷纭的工业互联网
- 工业互联网的认识与思考
  - 定义内涵与发展规律
  - 科学问题与技术挑战
  - 发展趋势与未来愿景

提纲

- **众说纷纭的工业互联网**
- **工业互联网的认识与思考**
  - **定义内涵与发展规律**
  - **科学问题与技术挑战**
  - **发展趋势与未来愿景**

从人类社会的发展说起

- **人类经历了农业社会、工业社会，正在进入信息社会**
- **在信息社会中，数字经济成为新经济形态**
  - 数据是重要生产要素
  - 网络是重要载体
  - 信息技术是重要推动力

农业社会

农业社会持续了几千年，直至18-19世纪

工业社会

工业革命开始以后到19世纪末，人类进入工业社会

信息社会

1990年代起互联网商用并获得高速发展。当前正在开启信息化引领社会经济发展的新阶段

4

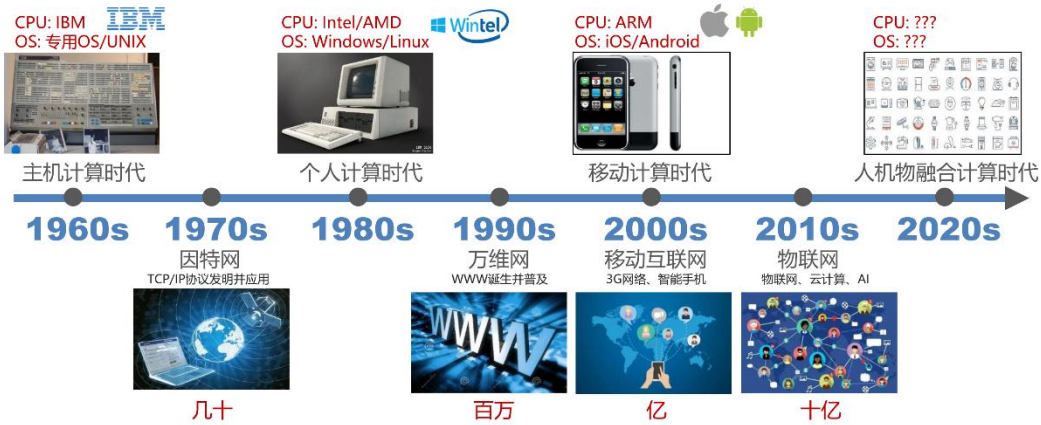
工业发展的回顾

- **工业是国民经济的命脉，工业史上的历次工业革命极大提升了人类的物质文明**
- **新一轮工业革命正在开始已成为共识！不同的是说法：**
  - **趋势学家杰里米•里夫金，《第三次工业革命》（2012.2）：经济和社会变革总是来自新能源与新通信方式的结合**
    - 19世纪，煤炭和蒸汽机与印刷品相结合
    - 20世纪，石油和内燃机与电话及广播电视相结合
    - 21世纪，新能源和互联网的结合（可再生能源的互联网）
  - **德国工业4.0**

5

计算技术及互联网发展深刻改变人类社会

计算机深刻改变了社会生活，CPU+OS生态每隔二十年发生一次重要变革



过去20多年，互联网带来了一场社会经济的革命！

6

互联网革命正在进入下半场，开启新的工业革命

- 万物互联时代，联网终端数量**即将突破百亿**，超过全球人口数量。未来，千亿计算设备联网已然可期！
- 信息技术（计算机、互联网）与生产技术（工业控制系统）两方面都孕育着巨大变革！

工业革命不到300年间对人类社会带来了巨大而深刻的变化。然而，以互联网为代表的新一代信息技术所带来的新一轮工业革命和社会经济“革命”，在广度、深度和速度上都将是空前的，也会是远远超出我们从工业社会获得的常识和认知、远远超出我们的预期！

7

工业互联网：新工业革命的基石

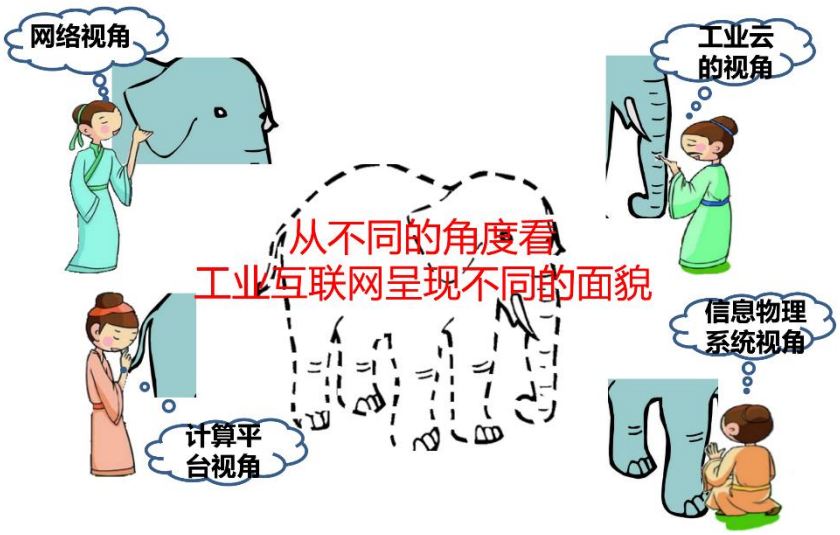
- 世界：新一轮**科技革命**和**产业变革**蓬勃兴起，在全球范围内不断颠覆传统制造模式、生产组织方式和产业形态
- 中国：正处于由**数量和规模扩张**向**质量和效益提升**转变的关键期，促进**新旧动能转换**，形成竞争新优势

数字化转型
网络化重构
智能化提升

8

什么是工业互联网?

工业互联网的理解之盲人摸象



10

网络视角下的工业互联网

“三网四层”架构，通过信息网络使原本割裂的工业数据实现流通，从而变成一个“智能网络”

控—将数据分析所得到的信息形成开放式服务，从而反馈到工业生产。

感—复杂多样的工业生产实体智能地识别、感知和采集生产相关数据。

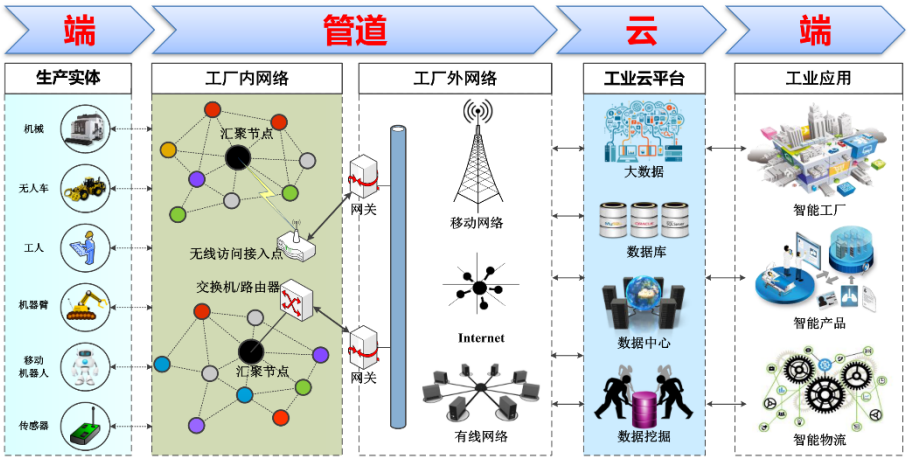
知—对这些网络化的工业大数据形成快速处理和实效分析。

联—工业数据在互联互通的网络上进行传输和汇聚。

11

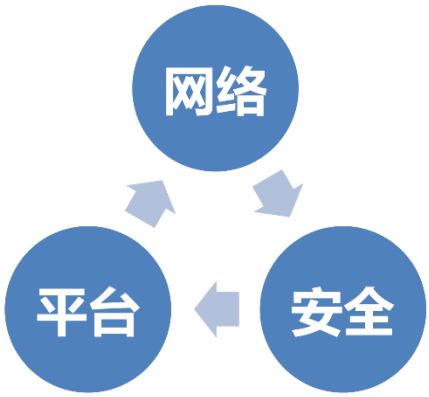
工业云视角下的工业互联网

云管端：工业生产实体（端）与工业云平台（云）通过网络（管道）协作计算，实现网络化、智能化的工业生产



12

计算平台视角下的工业互联网



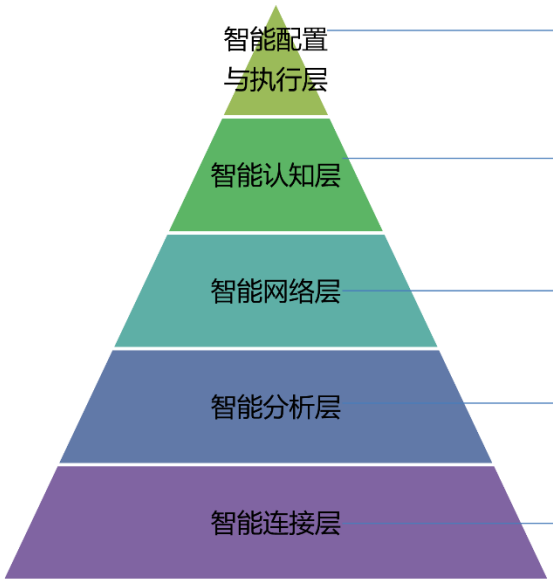
工业互联网

由网络、平台、安全三个部分构成。其中：

- **网络是基础**
- **平台是核心**
- **安全是保障**

13

物理信息系统视角下的工业互联网



- 弹性化的自重构能力
- 可变化的自调节能力
- 多维协同的自优化能力

- 一体化模拟与综合分析
- 人在回路的进程交互
- 评估、预测与决策支持的协同

- 装备“部件级-系统级”实体的网络综合模型
- 基于Time-machine的变化特征识别与提取
- 基于数据相似性挖掘的聚类分析

- 装备健康的智能评估
- 装备状态的综合分析
- 多维度的数据关联
- 衰退与性能的变化趋势分析与预测

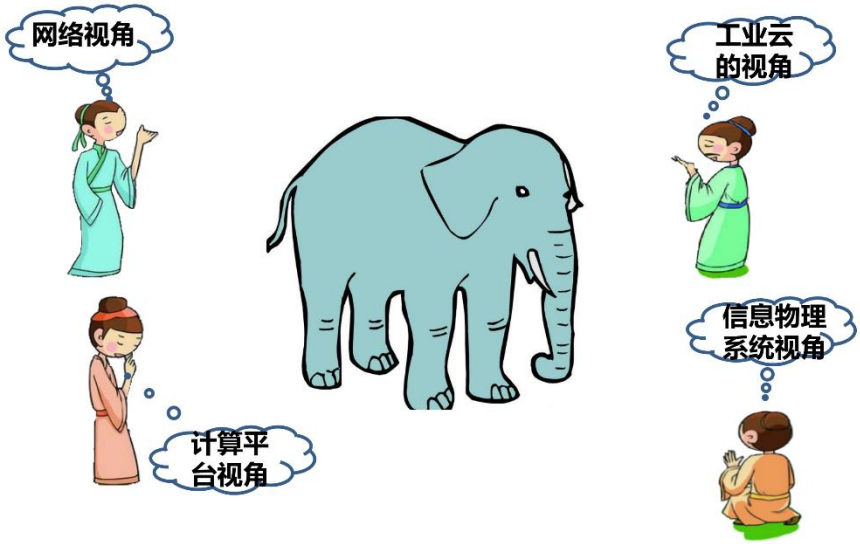
- 智能传感网络
- 非接触式采集与传输交互
- 敏捷、高效、即插即用

李杰, 邱伯华, 刘宗长, 魏慕恒. CPS:新一代工业智能. 上海交通大学出版社. 2017

14

工业互联网发展亟待解决的问题

需要凝练共识、归纳定义、总结规律、提出问题、给出建议



15

提纲

- 众说纷纭的工业互联网
- 工业互联网的认识与思考
  - 定义内涵与发展规律
  - 科学问题与技术挑战
  - 发展趋势与未来愿景

工业互联网的定义

工业互联网是通过新型网络、人工智能、大数据等新一代信息技术在工业中的深度融合和创新应用，建立广泛连接人、机、物等各类生产要素的全球性网络，形成贯穿全产业链的实体联网、数据联网、服务联网的开放平台，是重塑工业生产制造与服务体系，实现产业数字化、网络化、智能化发展的重要基础设施。

新兴业态

- 工业云服务
- 工业大数据
- 工业APP
- 众包众创

新兴应用

- 智能化生产
- 供应链管理
- 网络化协同
- 个性化定制

22

工业互联网的发展特点

工业互联网是互联网发展的新领域，是在互联网基础之上、面向实体经济应用的演进升级。通常所说的互联网一般是指消费互联网，与之相比，工业互联网有四个明显特点：

消费互联网		工业互联网
以人为主	连接对象	以物为主、人机物协同
“尽力而为”	技术要求	低延迟，安全可靠
应用门槛低 发展模式可复制	发展模式	应用专业化 无普适发展模式
我国起步晚 处于跟随状态	时代机遇	全球格局未定 正在战略机遇期

23

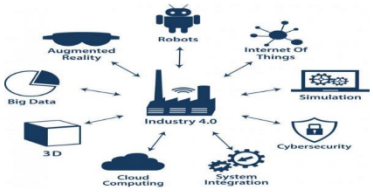
工业互联网的内涵

内涵：多学科交叉、多应用驱动、多技术融合

多学科交叉：计算机、自动控制、通信、电子、安全、机械、材料、制造…

多应用驱动：多样的应用场景，工业生产中产生新需求，颠覆性生产模式

多技术融合：物联网、人工智能、大数据、云计算、机器人、5G、AR/VR



工业互联网的发展规律

核心规律：互联范围的逐渐扩大，最终实现人机物的全面、深度、安全互联。可从三个维度具体体现：

维度一（企业内部）

- 企业内部互联范围逐渐扩大，覆盖“人、机、料、法、环”各方面，打通企业内部各系统间的信息孤岛，改变工业系统“七国八制”的格局，促进信息技术与生产技术的深度融合，打造智能工厂。

维度二（产品生命周期）

- 产品生命周期上互联范围逐渐扩大，实现产品全生命周期的联网。设计阶段联网，实现用户个性化定制以及设计与制造的协同优化；制造阶段产品物料与生产设备的互联，实现定制化生产的规模化；售后阶段产品的联网，实现远程的预测性维护。

维度三（产业链）

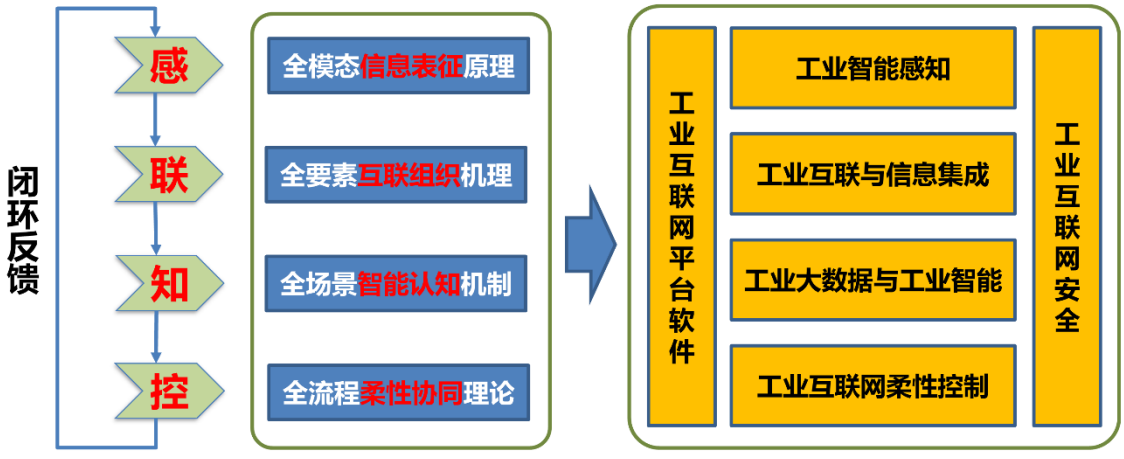
- 产业链上互联范围逐渐扩大，借助工业互联网平台，实现产业链上下游企业（供应、制造、销售、金融）之间的网络化协同。

提纲

- 众说纷纭的工业互联网
- 工业互联网的认识与思考
  - 定义内涵与发展规律
  - 科学问题与技术挑战
  - 发展趋势与未来愿景

工业互联网关键科学问题与技术方向

围绕工业互联网内涵，结合工业互联网发展规律和特点，凝练工业互联网领域的关键科学问题与技术方向



内涵映射　　关键科学问题　　关键技术方向

科学问题一：全模态信息表征原理

- **问题的由来**
  - 如何完成全模态工业数据表示
  - 如何发现深层次信息转换规律
  - 如何刻画泛时空动态过程模型

- **问题的内涵**
  - 统一的工业数据管理与表示
  - 深度的工业信息转化与验证
  - 动态的工业过程建模与迭代

- **问题的作用范畴**
  - 通过建模工业实体、生产要素，工业场景，刻画工业过程依赖关系和时空关联，以实现从物理世界到数字世界的映射

科学问题二：全要素互联组织机理

- **问题的由来**
  - 如何完成海量多元工业要素接入
  - 如何构建实时稳定工业网络互联
  - 如何实现异质离散工业实体组织

- **问题的内涵**
  - 工业要素泛在分型接入
  - 工业网络安全可靠互联
  - 工业实体集成统筹组织

- **问题的作用范畴**
  - 通过海量多元工业要素泛在分型接入，构建安全可靠工业互联网络，实现异质离散工业实体集成统筹管理

科学问题三：全场景智能认知机制

- **问题的由来**
  - 难以刻画全场景的工业认知表达范式
  - 难以形成自演化的智能认知学习能力
  - 难以构建强实时的工业计算决策方法

- **问题的内涵**
  - 全场景工业认知表达范式理论
  - 自演化工业认知智能学习机理
  - 自适应工业认知计算决策机制

- **问题的作用范畴**
  - 旨在赋予工业互联网机器认知表达、自主学习、智能决策的能力，通过建立有效的认知表达、获取和应用的机制，实现工业感知智能到工业认知智能的关键突破

30

科学问题四：全流程柔性协同理论

- **问题的由来**
  - 如何描述柔性程度与作用
  - 如何刻画协同过程与方式
  - 如何实现柔性协同与决策

- **问题的内涵**
  - 工业过程柔性表征与评估
  - 工业环境协同模型与决策
  - 工业全流程柔性协同方法

- **问题的作用范畴**
  - 基于工业泛在操作系统、软件定义工业网络、智能计算等信息化技术，对工业全流程进行全局的优化与协同，提高生产线的柔性反应能力和供应链的敏捷精准的反应能力，实现全流程柔性生产和精准控制

31

关键技术——工业智能感知

主要目标

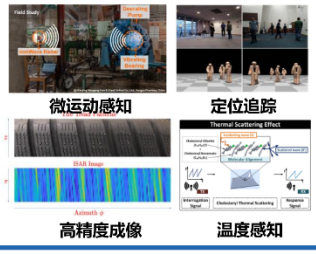
针对强干扰、大范围、多目标的复杂工业环境，通过智能化、微型化、集成化、网络化、多功能、低功耗的智能感知技术，实现面向工业环境的全方位感知。

技术挑战

- 如何针对新型非传感器智能感知技术，探索高感知分辨率保障方法?
- 如何针对无线信号特征的强部署环境依赖，实现高普适、低成本的智能感知?
- 如何开展无线信号空间的工业场景精细时空建模，实现高精度、强鲁棒的智能感知?

核心内容

- **感知通信一体化**
  - 通过分析无线信号传播过程中的变化，获得信号传播空间特性，实现场景感知
- **非接触式高精度感知**
  - 基于感知对象、环境与信号传播的关系，实现对目标位置、运动状态、环境的高精度感知



32

关键技术——工业互联与信息集成

主要目标

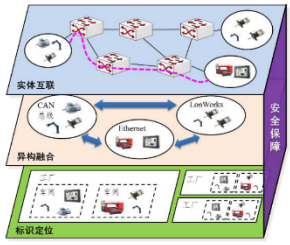
针对工业互联网异质工业实体与异构互联网络，设计面向大规模异质实体的高效自适应互联技术，实现泛在工业互联网络数据实时传输和信息高度共享。

技术挑战

- 如何为大规模联网工业实体提供统一有效的标识符，实现唯一标识及解析?
- 如何屏蔽工业网络协议异构性，实现异构工业互联协议间的互联互通及联网实体间有效数据交互?
- 如何保障工业互联网络的高带宽、低时延、安全可靠?

核心内容

- **分布式异构资源的标识解析**
  - 新型的资源标识和解析方法，统一标识大规模联网实体，支持超大规模和高性能标识解析
- **异构工业网络融合及高效互联**
  - 通用的通信协议标准和接口规范，通过异构网络融合及网络优化技术，实现高效自适应互联



33

关键技术——工业大数据与工业智能

主要目标

针对海量、多源、多维、异构的工业大数据，通过大数据的实时处理分析并融合新型人工智能技术及工业领域知识，支持工业环境下的智能感知、分析、决策。

技术挑战

- 如何实现异构异质工业大数据的采集、预处理、存储、分析的全流程优化?
- 如何从异构异质工业大数据源中高效抽取工业知识，支持知识管理和演化?
- 如何在更高维且动态的真实工业生产场景中，实现基于多智能体强化学习的分布式人工智能，以支持错综复杂任务的完成?

核心内容

- **工业大数据实时分析处理**
  - 实现工业图数据、时序数据的实时存储和处理
- **工业知识图谱及分布式工业智能**
  - 通过知识/认知图谱技术，实现工业知识抽取、推理和协同
  - 基于多智能体强化学习技术，实现多智能工业设备的交互和决策

自组织的分布式工业智能

34

关键技术——工业互联网柔性控制

主要目标

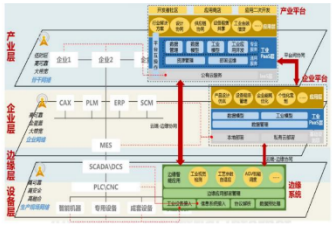
面向智能制造和智能电网等领域，通过分布式协同控制方法和控制技术，实现基于信息物理体系的智能控制及系统，支持工业互联网柔性控制。

技术挑战

- 如何实现新型工业互联融合控制(IT+OT+CT)技术，支持整体的融合贯通、全面连接和决策控制?
- 如何实现工业互联网全要素全过程管控技术?
- 如何实现基于工业网络的产业链和价值链调控技术?

核心内容

- **面向“设备-边缘-企业-产业”的协同控制**
  - 基于多Agent的协同控制理论与方法、“云-管-边-端”分布式控制技术等
- **基于信息物理体系的智能控制及系统**
  - 数字孪生模型/物理模型交互控制、信息流-决策流双向作用的优化闭环控制等



35

关键技术——工业互联网平台软件

主要目标

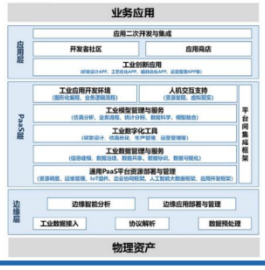
面向产线、车间、工厂、产业链等不同层级对资源管理和应用开发的需求，作为一种系统软件，提供工业资源的泛在连接、弹性供给、高效配置，以及面向工业场景的应用系统开发支撑。

技术挑战

- 如何针对工业生产场景中大规模、高异构、强动态的资源进行有效的管理?
- 如何针对工业生产场景多样、流程复杂、状态多变，实现平台服务的按需灵活定制?
- 如何针对应用需求与形态多样的情况，凝练应用的共性?

核心内容

- **海量、多类、异构、跨域资源管理**
  - 资源统一抽象、资源高效虚拟化、资源灵活调度、资源按需聚合
- **需求多变、形态多样、环境不确定的软件系统开发**
  - 复杂场景分析与建模、人机协作编程、开发运维一体化、适应演化支撑



关键技术——工业互联网安全

主要目标

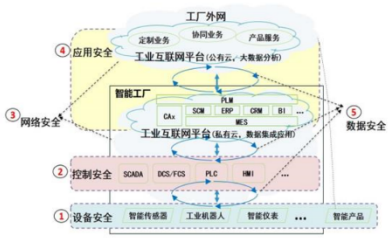
针对工业互联网大量设备导致安全攻击面扩大、多层级网络导致安全边界模糊、分布式数据导致隐私保护困难等安全特性，构建涵盖设备层、网络层、数据层、应用层、控制层的工业互联网安全防护技术体系。

技术挑战

- 如何保障芯片、固件、操作系统等设备相关的内生安全?
- 如何实现轻量级的认证机制和加密算法，保障工业互联网络安全?
- 如何提供统一灵活的认证、授权、审计等应用安全保障机制?
- 如何实现具有完整性保证的访问控制安全协议?

核心内容

- **设备安全：设备内生安全机制**
- **网络安全：动态网络安全防御机制**
- **数据安全：数据分类分级保护机制**
- **应用安全：应用平台安全保障机制**
- **控制安全：访问权限控制安全机制**



提纲

- 众说纷纭的工业互联网
- 工业互联网的认识与思考
  - 定义内涵与发展规律
  - 科学问题与技术挑战
  - 发展趋势与未来愿景

工业互联网的发展趋势

随着人工智能、区块链、物联网、云计算、大数据、5G通信等工业互联网核心技术的进步和在工业场景中的不断应用，工业互联网将向泛在化、协同化、智能化的形态不断发展，最终有效提升工业产能并演化新的业态，支撑新一轮科技革命和产业变革。

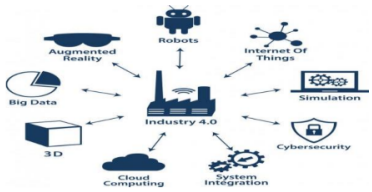
总体发展趋势：泛在化、协同化、智能化

工业互联网的发展趋势

总体发展趋势：泛在化、协同化、智能化

泛在化：工业互联网将接入工业全场景、各环节涉及的人、机、物，包括海量的传感器、可穿戴设备、虚拟设备等，**实现无处不在、无迹可寻工业数字孪生世界。**

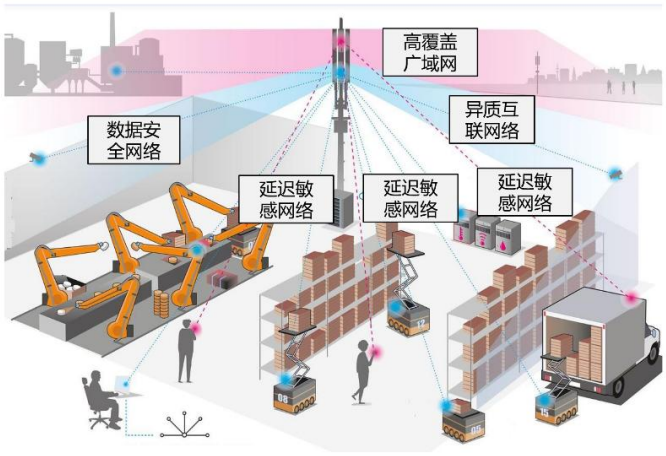
- 接入设备远超百亿级
- 更全面更精确地感知物理世界的状态
- M2M连接为主要通信模式
- …



工业互联网的发展趋势

总体发展趋势：泛在化、协同化、智能化

协同化：工业全场景、全链条人、机、物高度协同。通过**亚微秒级低延时、高可靠、广覆盖**的网络基础设施，实现信息数据在生产各环节和全要素的无缝传递，从而支撑形成实时感知、协同交互、智能反馈的融合生产新模式。



工业互联网的发展趋势

总体发展趋势：泛在化、协同化、智能化

智能化：通用人工智能技术与工业场景、机理、知识的深度结合，工业互联网将贯穿于工业设计、生产、管理、服务等工业领域各环节，实现模仿或超越人类感知、分析、决策等能力的技术、方法、产品及应用系统。

人工智能　　工业互联网　　工业领域　　工业新认知

工业互联网的未来愿景

随着工业互联网技术、标准、形态的不断发展和成熟，工业互联网最终将在制造、能源、交通、医疗等行业深度应用融合，并带来革命性的产业变革。

实现智能制造新模式

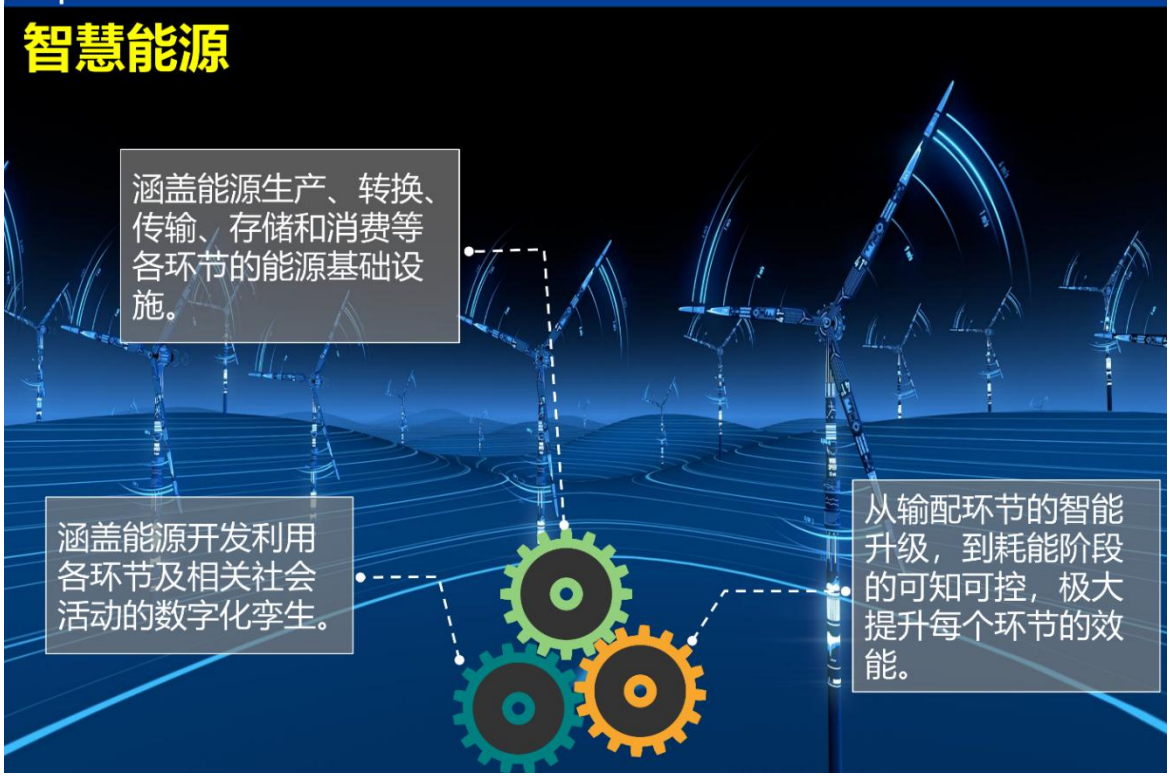
构建智慧能源新产业

工业互联网的未来愿景

智能制造

工业互联网的未来愿景

智慧能源



COPU 编者的话：

组织“卡脖子”的“工业软件”攻关

建设工业互联网涉及开发工业软件问题。

我国工业软件，特别在（1）制造业数字生态及基础前沿技术；（2）产业生命周期核心软件；（3）智能工厂技术与系统；（4）产业协同技术与平台等四个技术方向，存在被国外“卡脖子”的隐患。今年 2 月初，科技部发布《国家重点研究计划“工业软件”重大专项 2021 年度项目申报指南建议》，掀起了国内攻克“卡脖子”“工业软件”的高潮。攻关方向为：平台及工业 APP；研究设计软件：3DCAD、CAE、EDA；生产管控软件：DCS、SCADA、MES；经营管理软件：ERP、CRM、SCM。

借本次出版“深度信息技术（精品）专辑”工业互联网栏目的机会，我们还推荐清华大学王建民教授团队研发的工业互联网时序数据库管理系统（Apache IoTDB）。

工业物联网时序数据库管理系统 Apache IoTDB

清华大学 王建民 黄向东

摘要

近年，工业 4.0 和工业互联网成为发展焦点。在这次工业革命中，工业生产和运维过程中机械设备产生的时序数据成为了新的核心资产。Apache IoTDB 是清华大学最初研制、于 2018 年捐赠给全球最大的开源软件基金会 Apache 的时序数据库管理系统软件，并于 2020 年 9 月正式成为 Apache 全球顶级开源项目。迄今为止已在中国、美国、德国、日本、澳洲等积累大量用户。

时序数据及其管理需求

物联网诞生于 1999 年，在其理念和技术的不断革新下，无处不在的设备和设施正在被越来越多的通过网络连接起来，并不断向云端发送实况数据。近些年，工业 4.0 和工业互联网成为发展焦点。在这次工业革命中，工业生产和运维过程中机械设备产生的数据成为了新的核心资产。随着 Devops 的兴起，运维人员开始追求对 IT 基础设施、各种服务的运行情况了如指掌的境界，从而开始大量采集对这些设施、进程、用户的监控数据。上述这些场景中产生的数据，主要都是用来描述一个对象在时间维度上的变化的数据，这类数据就被称为**时间序列数据**，简称时序数据。

工业中的时间序列数据具有**超高通量与元数据多变**的特点。由于工业设备的数量可能非常庞大，同时每一个设备上都会有着上百种传感器同时采集数据，并且这些设备很多都是 7*24 小时不间断地运行，因此其采集的时间序列数据是源源不断的。以金风科技的风机组管理为例。当前金风科技拥有 2.8 万多台风机，其中 2.2 万台风机可以采集时序数据并向数据中心回传。以每个风机上约 100 种测点(传感器)为例，总时间序列数量可达 220 万条。以其 SCADA 系统中的 7 秒时间间隔采集数据 为例，一天将产生 271.5 亿个数据点。又以中国中车青岛四方所在上海地铁的应用为例，单列车拥有 3200 个测点，当前采集频率为 500 毫秒/次，共有 180 列车，其每秒的吞吐量达 118 万点每秒。未来的采集间隔将缩短至 200 毫秒，列车数量将增至 300 列，则每秒吞吐量将达 492 万点每秒。可见，必须首先解决数据的存储效率问题，才能有效管理工业场景中的时序数据，从而进一步利用其价值。更甚至，由于这些数据是昂贵的设备在真实生产环境中产生的，其数据价值远高于模拟仿真数据，随意删除这些数据将给日后的数据分析带来巨大的损失，因此全时全量的存储是必需的。此外，不同设备类型的传感

器不同，随着设备运行升级，不同设备实例上新增的传感器、可采集的测点也不同，这就造成了工业场景下的时序数据元数据多变的情况。

在此基础上，工业中的时间序列数据还具有**低质乱序传输**的特点。在一些实 际应用场景中，时间序列数据存在乱序的问题。例如:在物联网与工业传感网中， 因为基础设施或者网络的不稳定，数据的采集和回传会出现中断，在这类情况下数据会出现一定范围的乱序;还有一些比较严重的情况，数据在进入存储系统之前需要经过 kafka 等流式处理系统的处理，同一条时间序列数据可能会被 kafka 的多个 partition 消费，从而导致数据在写入存储系统的时候出现乱序。此外，由于 PLC 与 DTU 模块的衔接或网络等问题，一些数据点会在回传前或者回传过程中丢 失，造成数据缺失等质量低下问题。

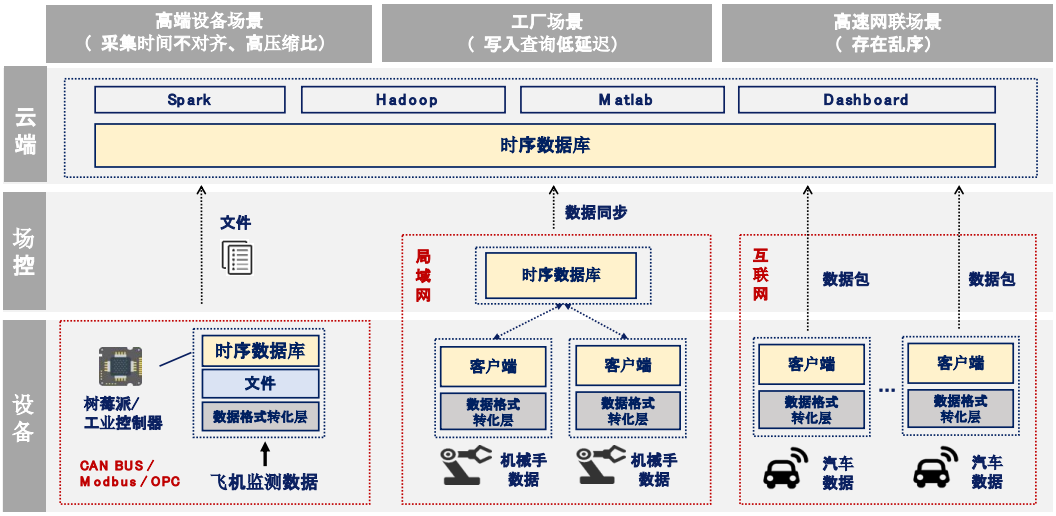
此外，工业中的时间序列数据具有**高质全序查询**的特点。时序数据虽然在写 入过程中存在乱序现象，但用户在查询过程中，大多都是要求按照时间维度顺序读取数据。此外，工业互联网中设备的每个传感器会产生一条甚至多条时间序列数据，这些时序数据最有效的存储方式是采用列式存储。然而，实际应用中需要对一个设备的多个度量指标数据，或多个设备的数据同时进行分析，这就要求必须将多条列式存储的时序数据进行连接查询(即按时间戳对齐查询)。该查询的性能对数据分析的效率起到了决定性作用。

现在有很多种数据库可以用于管理时序数据，包括关系数据库和 NoSQL 数据库和时序数据库，但是他们都存在或多或少的问题。以关系数据库为例，其 schema 是有限制的:比如 MySQL InnoDB 中单表的列数上限为 1017 列，PostGreSQL 的列数上限是 1400 多列。然而，一个设备可能拥有上千个测点甚至上万个测点，一张关系表的列数是不够的。此外，在关系数据库中若单表的行数超过千万量级，性能往往会大幅下滑。因此就需要 DBA 进行复杂和精细的水平垂直分库分表。带来了使用上的不便。此外，关系数据库的写入性能也远远达不到时序数据的要求。对于以键值为代表的 NoSQL 数据库而言，其高速的写入性能在一定程度上解决了关系数据库的性能问题，灵活的 schema 也解决了关系数据库的表行、列数量的限制，但是其能够支持的查询往往没有关系数据库丰富。前文提到的时序数据时间维度的查询、值查询、多序列聚合、多序列时间对齐等查询都存在较大的挑战。TimescaleDB、OpenTSDB、KairosDB 等基于关系数据库或 NoSQL 数据库的系统在一定程度上解决了上述的问题，但难以根本上解决，例如 TimescaleDB 随着导入时间的增加其导入速度速率会不断地下降。KairosDB 在压缩、查询和写入性能上表现都不够突出。一些原生的时序数据库如 InfluxDB 针对时间序列数据进行了专属的文件结构优化和专属查询优化，然而在一些工业场景下仍然存储无法存储全量数据、性能不足的问题。

Apache IoTDB 及其发展历史

清华大学软件学院大数据团队自 2011 年起，在国家 863 项目的支持下，以三一重工的工程机械装备监测数据管理分析为依托，展开对时序数据管理技术的研究。期间，深入分析了现有 NoSQL 技术应对工业时序数据管理应用的不足，突破了多项核心技术，提出了时间序列存储格式、针对时间序列数据改进的 LSM 引擎、海量数据范围的新型聚合索引、设计了双层元数据组的分布式架构。

经过两代技术迭代，团队最终研制了支持物联网时序数据收集、存储、查询与分析一体化的数据管理引擎 IoTDB，支持“端-边-云”一体化部署，适用于高端装备管、工厂设备、高速网联设备等多种数据管理场景，为工业互联网时序数据管理提供核心基础支撑。



IoTDB 具有七大功能特点：

（1）高速吞吐：支持数据乱序写入、更新、删除，每秒千万点吞吐，让存储更高效。

（2）高压缩比：独创针对时间序列优化的列式文件存储格式 TsFile，支持有损、无损等多种高效编码、压缩方法，让存储更经济。

（3）海量序列管理：单节点管理百万设备、千万条时间序列，让覆盖更广泛。

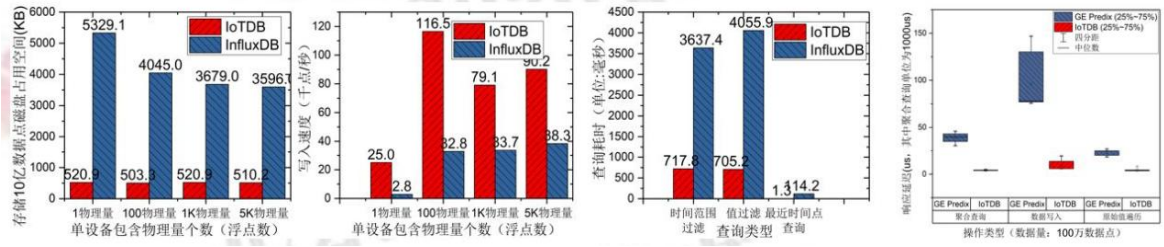
（4）低延迟查询：通过预聚合和时序索引支持快速数据过滤、高效聚合查询、降采样查询等典型时序数据查询种类，让查询更快速。

（5）查询分析一体化：采用存储和计算分离的架构，一份数据同时支持实时查询和大数据分析，避免数据迁移代价，缩短数据分析延迟，降低系统复杂度，让分析更容易。

（6）轻量化部署：云侧、边缘侧开箱即用、一键部署，让运维更低成本。

（7）生态丰富：与 PLC4X、Pulsar、Flink、Spark、Grafana、Zeppelin 等 IoT 与大数据系统无缝集成，覆盖时序数据的采集、处理存储、分析、可视化应用等全生命周期。

第三方测试表明，IoTDB 在磁盘占用空间、写入性能、 查询性能三个主要指标上领先于同类产品中性能最好的 InfluxDB；IoTDB 在写入延迟、遍历查询、聚合查询三个主要指标上优于美国 GE Predix。



2018 年 11 月，经国际顶级开源基金会 Apache 评估和认可，IoTDB 成功进入 Apache 孵化器，成为中国高校目前唯一主导的 Apache 项目。2020 年 9 月，经社区投票和董事会批准，**Apache IoTDB 正式成为国际顶级开源软件基金会 Apache 全球顶级项目**。数据库事务恢复技术奠基者之一、美国工程院院士 C. Mohan 评价“IoTDB 是中国高校首个达到国际顶级标准的数据库项目”。中国工程院院士廖湘科评价“IoTDB 针对边缘和云侧不同的运行环境、操作系统、工作负载进行了优化，创新了数据存储与查询分析技术，是工业物联网的核心基础软件。Apache IoTDB 透过开源模式，把这些创新技术带给全世界”。

目前，社区用户群体数千人、开发群体百余人，分别来自中国高校、德国工业 4.0 相关企业、南北美工业信息化企业、东方国信、阿里、云智慧、用友、联想、华为、四维图新、网易等头部公司。一个以中国高校和企业开发者为主、国际贡献者和用户众多的全球性工业物联网数据管理开源社区逐渐形成。

IoTDB 开源社区还积极探索推动国内院校和社会的开源教育与开源文化建设。多次在北京航空航天大学、湖南科技大学、湖北大学、新疆大学等院校举行开源讲座，形成超过 500 人的学生线上开源交流社区。连续三年举办面向社会的开源知识讲座，Apache 基金会主席 Crag Russell 关注并多次出席。其中“开源之道大讲堂”活动作为典型开源社区建设实践被 Apache 2020 年全球大会主题报告介绍。梅宏院士评价 IoTDB 是“中国开源教育与文化建设的成功实践。”

IoTDB 获 2019 年优秀大数据产品称号、第二届中国优秀开源项目一等奖、2019 年度最受欢迎中国开源项目称号。被国际著名数据库排名网站 DB-Engines、CMU 数据库名录等收录。

Apache IoTDB 典型工程应用案例

目前，IoTDB 已在钢铁冶炼、石油化工、飞机制造、核电、风电、智慧电厂、城市交通运输等多个领域得到应用，用户超过千余用户/企业，覆盖中国、德国、澳大利亚、美国、印度等多个国家。在国内，IoTDB 有效支撑了中航成飞、中车四方、中国船舶、国家电网、中国烟草、金风科技、大唐电力、联想、东方国信、北京汽车等龙头企业工业互联网落地升级。

以中车青岛四方为例，基于 IoTDB 建成城市轨道交通车辆运行状态数据中心，替代 了 KairosDB 时序数据库，存储空间节省 14 倍，服务北上广等城市近 20 条地铁线路、400 列车。湖南大唐先一在湘潭电厂、耒阳电厂等上线 IoTDB，替代了基于 Hadoop 和 HBase 的大数据管理系统,支撑了单电厂 30 万序列实时读写，比 HBase 性能提升 50%。

国际上，全球最大的钢铁生产公司 ArcelorMittal 美国公司使用 IoTDB 替代其 HBase+Spark 数据管理体系、德国联邦经济和能源部资助成立的 Pragmatic Industries 使用 IoTDB 为德国宝马发动机缸体制造实时数据提供有力支持。

区块链

COPU 编者的话：探索数字货币发展之路

本来想发表一篇区块链与数字人民币的专著，目前似乎还不成熟。但在本专辑中，我们发表陈钟教授和 Brain Behlendorf 大师有关区块链的文章，其中也涉及数字货币的问题。

包括中国在内世界多国都在研究发行数字货币。中国已在国内若干个城市开展法定数字人民币试点工作，在现阶段，我国希望通过这些试点，摸索数字人民币在流通过程上的规律和经验，以及如何制定运行规则。了解数字人民币如何解决及完善防盗、防伪、防洗钱及减少偷税和漏税等功能。

成熟的数字货币国际化（成为国际货币）是数字货币发展的必由之路。国际货币的影响力是由国家跨境汇款和结算的经济实力、资本市场的发达程度以及核心技术如区块链技术的支撑力度和本国货币国际化程度共同决定的。目前，人民币国际化进程，我们是走了一半，即已经实现了人民币经常项目的自由兑换，还没有实现资本项目可自由兑换，人民币已成为国际储备货币之一，因数额较小食用不够充分，探索数字人民币发展之路还要走很长的路。

区块链与数字货币

北京大学 陈钟

1. 引言

区块链（Blockchain），又称为分布式记账技术（Distributed Ledger Technology），是一种共享的、分布式账本，用于在商业网络中促进交易记录和资产跟踪，采用密码学和分布式共识机制保证了记账信息不可篡改和账本一致性，可广泛用于多方参与、协同合作、记录和利用交易历史的应用场景。

区块链的第一个也是最具影响的应用是比特币（Bitcoin）。2008 年 11 月，比特币以一篇《一种点对点的电子现金系统》的论文面市，作者署名中本聪至今仍匿名。比特币的这篇文章中甚至完全没有“区块链”这个术语，而是后来人们总结出来比特币的底层技术是区块链，并将其发展成为能支持包括比特币在内的多种应用的基础平台。短短十年中，基于区块链技术的比特币系统以其开源的特性和社区化的运行机制，引发了后续区块链与加密数字货币创新热潮。

区块链技术最早是从比特币系统中剥离出来用以实现比特币应用的技术平台，也被称为分布式账本技术或分布式记账技术。我国成立的区块链与分布式记账技术全国标准化技术委员会就沿用了“区块链与分布式记账”的名词术语。中国人民银行 2020 年发布的行业标准《金融分布式账本安全规范》中，则用金融分布式账本的术语完全代替金融区块链，通篇没有用到区块链的名词。特别是 2014 年面世的以太坊，创新性地提出支持包含比特币在内的各种应用的区块链平台（称为以太坊）和智能合约，进一步扩展了区块链技术的内涵与外延；2016 年建立的 Linux 基金会 Hyperledger 社区则以企业区块链应用为目标，拓展了许可或称联盟区块链相关的概念和技术支撑，进一步引领区块链的应用从加密数字货币向企业信息化的其他领域扩展。

区块链与数字货币的出现，引起人们对未来互联网从信息互联网向价值互联网发展的憧憬与探索，数字经济在互联网的基础上也被赋予新的内涵，特别是区块链技术所带来的价值与信任的技术支撑，有可能带来人类生产和生活的变革。

2019 年 10 月 4 日，习总书记在组织中央政治局第十八次集体学习时，对区块链做出重要指示，指出要把区块链作为核心技术自主创新的重要

突破口，加快推动区块链技术和产业创新发展。2020 年，我国数字人民币开始在深圳、成都、苏州、雄安新区及未来冬奥会场应用试点。本文就区块链与数字货币安全这一问题展开讨论。

2. 数字货币的含义和影响

随着 20 世纪计算机、通信和网络技术的发明和普及应用，货币与货币系统的电子化、数字化也进一步得到科技的支撑，曾经被称为金融电子化。特别是 20 世纪 90 年代末互联网开始大规模商用以后，金融科技发展成为热点。

电子货币（Electronic Money），是指用一定金额的现金或存款从发行者处兑换并获得代表相同金额的数据或者通过银行及第三方推出的快捷支付服务，通过使用某些电子化途径将银行中的余额转移，从而能够进行交易。严格意义是消费者向电子货币的发行者使用银行的网络银行服务进行储值和快捷支付，通过媒介（二维码或硬件设备），以电子形式使消费者进行交易的货币。此外，还有电子现金（e-cash）、电子支票（e-check）、电子票据（e-bill）、电子支付（Electronic Payment）等，均可以表示为传统的现金、支票、票据和支付在信息技术的支撑环境中的电子化形态，并且得到相关法律的支撑。

互联网大规模应用催生了支付结算的便利化、通常称为电子支付或者移动支付，后统称为网路支付。网络支付，是指依托公共网络或专用网络在收付款人之前转移货币资金的行为，包括货币汇兑、互联网支付、移动电话支付、固定电话支付、数字电视支付等。由于移动互联网的便捷性和广泛性，腾讯和阿里巴巴以其在我国第三方支付服务的垄断地位，其微信支付和支付宝普及率达 90%以上，也为我国无现金支付做出了巨大贡献。

然而，比特币面世引发加密数字货币的科技创新，同样也冲击并带动传统金融机构和央行的金融科技创新。数字货币是金融科技（FinTech）领域中的一个重要组成部分，作为金融科技中的一项具体应用，自其诞生以来得到了快速发展。数字货币发行，不仅仅会对本国金融和货币系统产生直接影响，也可能会对全球金融体系、全球货币生态体系带来重大影响。

总体上说，数字货币是一种基于信息网络、密码学算法设计的虚拟货币，当前按照是否由中央银行发行、是否法定货币、是否被中央银行支持、是否与法定货币挂钩、是否支持点对点传输、是否可编程等六个维度对数字货币进行划分，主要存在四种类型的数字货币，即中央银行数字货币（CBDC）、合成央行数字货币（sCBDC）、稳定币和加密资产。中央银行数字货币（CBDC）是一种主权货币的数字表现形式，由一个地区金融监管机构发行，

并出现央行资产负债端。sCBDC 是由中央银行支持发行的法定货币，但本身不会对中央银行形成求偿权。稳定币是与法定货币挂钩的加密资产，而加密资产一般由私人发行的代币，是价值的数字表示形式，但并不以法定货币计价。需要说明的是，数字货币的分类未来也有可能会发生变化。

数字货币一般是法律上认可的一种货币形式，或者法律不禁止、大众认可的一种货币形式，它既可以像传统货币一样用于购买商品或者服务、同时支持履行相关金融义务，又可能承担数字经济中的价值交换的创新、呈现出传统货币所不具有的特性。例如，加密数字货币基于区块链所具有的支持可编程属性，可以将契约以计算机程序的形式内嵌进去，支持自动执行。目前，基于智能合约的 DeFi 应用就显示出许多金融创新的特性，是以往金融领域所未有的。

数字货币安全一般具有两个方面的含义：一是数字货币本身的安全，主要表现在实现数字货币形态及特性所必须采取的安全技术，包括密码技术、安全芯片技术、系统安全、网络安全、数据安全技术等；另一个是数字货币带来的货币制度或者金融体系的安全，其中包括了法律、监管、合规、风险管控等，例如金融监管通常所涉及到了 KYC/AML/CTF 等。

2016 年，王永红在《数字货币技术实现框架》一文中总结概括了数字货币的主要特性，如表 1 所示。较全面地描述了数字货币及其安全要求,也是目前数字人民币所具备的全部特性。其中，除可离线交易性以外，其它的全部特性同样适用于基于区块链的加密数字货币。数字货币的安全包括了这些特性的保障措施和技术手段，在此不作赘述。

表 1 数字货币主要特性及其主要内容

主要特征	主要内容
可流通性	数字货币可作为流通和支付的手段在经济活动中进行的价值运动
可存储性	数字货币以电子数据的形式安全储存在机构或用户的电子设备中，可供查询、交易和管理
可离线交易性	数字货币通过电子设备进行交易时可以不与主机或系统发生直接联系，不通过有线或无线等通信方式与其他设备或系统交换信息
可控匿名性	数字货币相关流通信息除货币当局外，任何参与方不知拥有者或者以往使用者的身份信息，确保交易过程中的匿名性与不可追踪。可采用“前台自愿、后台实名”的形式，可控匿名性兼顾了反洗钱、反恐怖融资等监管要求
不可伪造性	数字货币在制造和发行过程中通过多种安全技术手段保障货币不能被非法复制和伪造
不可重复交易性	数字货币拥有者不可将数字货币先后或同时支付给一个以上的其他用户或商户
不可抵赖性	数字货币交易双方在交易后不可否认交易行为及行为发生的各类要素

3. 中央银行数字货币（CBDC）国内外发展

加密数字货币带来全球货币发展的强劲势头，包括欧美等西方发达国家以及柬埔寨等新兴经济体，均已开展中央数字货币研究。根据 2019 年国际清算银行对 66 家中央银行数字货币调查结果显示，已经接近 80%已经开展 CBDC 发行可行性分析，10%已经接近发行阶段。表 2 总结了当前世界范围一些国家 CBDC 发行现状。

表 2 国外数字货币发展现状总结

发达经济体	美国	2019 年 Facebook 公司宣布发行 Libra，受限于美国国家政府压力，担心此举可能会动摇美元世界货币霸权地位，Libra 的货币属性已经从跨主权的数字货币属性演变为抵押发行的数字美元。更名为 Diem
	俄罗斯	2020 年，俄罗斯普京签署加密货币法案，虽然承认数字货币属于合法资产，但仍禁止数字货币作为支付手段。
	法国	2020 年初，法兰西银行已经开始初步展开中央数字货币 MDBC 试点服务，其中包括跨境支付功能。
	韩国	2020 年 3 月，韩国央行韩国银行展开为期 22 月数字货币发行和试点计划，预期 2021 年建立相应的试点系统。
	日本	2020 年 7 月，设立数字货币组，开展数字货币结算系统研究工作。
	英国、瑞士等欧盟国家	2020 年 8 月 19 日，相应国家在参加国际货币金融机构官方论坛时表示，将会开展区块链和 CBDC 相关可行性研究。
新兴经济体	柬埔寨	2019 年 7 月，柬埔寨国家银行开始试点数字货币 Bakong，预期在 2020 年第三季度正式投入使用，以期优化跨境业务处理效率。
	泰国	2020 年 6 月，泰国正式启动数字泰铢支付系统试点工程，预期在 9 月份正式与香港开展数字货币交易。
	菲律宾	2020 年 7 月 29 日，菲律宾宣布已经开展数字货币政策可行性相关研究。
	巴哈马	巴哈马宣布将于 2020 年 10 月开始在全国流通数字巴哈马货币沙元（Sand Dollar）。

从表 2 可知，最近几年，各国纷纷开始布局 CBDC 相关研究活动，并不仅限于金融业发达的欧美国家。当前人民币占据了全球外汇储备的 2%左右，欧元占了 20%左右，而美元则占据了 60%以上，说明了美元一家独大的货币体系仍是常态。随着“一带一路”政策的推行，使用人民币的国家和人口也将会继续增加，DCEP 的推出，能够进一步提升人民币使用便捷度，但其中仍需要关注相应的风险点。

在 CBDC 领域，中国是率先展开相关业务的国家。中国早在 2017 年，就开展中国版 CBDC 的研究与实验，后来立项命名为中央数字货币电子支付 DCEP（Digital Currency Electronic Payment）,并在 2020 年 8 月 14 日，开始在京津冀、长三角、粤港澳大湾区等地区试点 DCEP 发行。DCEP 并不是对现有货币的数字化，而是 M0 的代替。它可以有效降低交易环节中对账户的依赖性，有利于人民币的流通和国际化，同时可以实时收集货币流动的各个环节，有利于实现货币监管。

CBDC 是当前国际上的一个研究热点。首先 CBDC 可以增强本国支付系统的市场竞争力，有效抵御传统的支付体系垄断；其次 CBDC 可以降低发行实物货币成本，为一些金融体系不发达的偏远地区提供普惠金融服务；可以增强政策调控经济市场的敏感度，有利于货币政策快速传导和生效；还可以减少或阻止私人货币发行，避免这些货币诱发的金融风险，实现货币市场的有效监管；帮助增强货币支付吸引力，降低美元化；在财政激励不容易涉及到的地区或人群，CBDC 能够发挥一定的作用。

需要说明的是，发行 CBDC 也面临一些风险。首先 CBDC 可能会改变汇率传导渠道，影响货币政策的传导。CBDC 引入后，可能会以一种不可预测的方式改变基础货币需求组成，改变货币构成元素，影响货币市场需求量对利率变化的弹性。CBDC 还可能加快货币管理政策出台的步伐，以应对全球市场的汇率变化，这反而会引发更强更不可预测的货币市场汇率变动。其次 CBDC 可能会在与银行存款竞争过程中，影响金融稳定。在引入 CBDC 时，需要考虑是否为其附属利息属性，如果没有利息，CBDC 无疑最接近现金作用，但可能会影响其推广力度；而如果需要为用户持有 CBDC 支付利息，这必定会与银行存款形成竞争关系。其次 CBDC 还可能会对央行资产负债表产生重要影响。如果央行将 CBDC 发行去中介化，那么央行可以将原来从银行分流的资金重新贷回给这些银行，增多市场货币发行量，有可能诱发通货膨胀，这将会背离央行发行 CBDC 的初衷。如果央行仅针对实物现金发行数字货币，虽然会对市场正常货币秩序产生的干扰可能最小，保持央行资产负债表规模不变，但是也可能会减弱 CBDC 的影响力。其次设计不佳的 CBDC 可能会触发严重的经济危机，甚至加速银行挤兑。在个别银行出现资不抵债的情况下，挤兑苗头很容易在极短时间内蔓延到整个银行系统，CBDC 虽然可以让央行尽快向陷入困境的银行注资，缓解相应的金融风险，但仍存在相应的可能。最后，在通货膨胀和货币汇率不稳定的国家，国民可能会增持其它国家的 CBDC，这可能会加速货币替代，最典型的可能就是会促进国家货币美元化。

从上述分析可知，国家在发行 CBDC 时，需要进行综合性分析，提前作为风险管控。这些考虑因素包括法律法规审查、利益相关群体有效合作、技术基础设施支撑、人才队伍和专业知识储备等。发行 CBDC 是一个综合性工程，已经超过传统中央银行角色功能，需要对其进行综合考虑。

4. 基于区块链的数字货币

尽管中国的 DCEP 作为 CBDC 的一员并未采用基于区块链的技术基础，区块链分布式账本技术依然是国际 CBDC 考虑的主流技术路线，但是大多尚未推出试点，部分早期试点的项目也并未取得实质性的成功（如：英国英格兰

银行的 RSCoin，加拿大央行的 Jasper 等）。这里重点介绍区块链用于 CBDC 以外的数字货币实践，主要包括数字资产和稳定币。

2009 年 1 月比特币系统以开源软件的形态发布并产生了创始区块，随后采取开源和社区治理模式缓慢运行，2010 年 5 月 22 日被命名为比特币披萨日，原因是一位弗罗里达的程序员用 1 万比特币购买了价值 25 美元的披萨优惠券。那时的比特币并未得到像今天的追捧，更没有人想到今天比特币可以超过 5.7 万美元一枚，总市值过万亿美元。比特币作为一种实验，证明了一件事：在无中心权威机构控制的前提下，一种数字货币得以发行和利用，打破了传统的货币概念和体系。

按照传统货币概念和理论来衡量，比特币完全不是货币，准确地说，比特币是一种数字资产。但是，如果按照新的经济理论来看，比特币的创新带来了新的挑战与机遇。比特币面世之后，因其开源与开放的创新发展模式，激励了数以千万计的加密数字货币创新，其中不乏开源创业社区领袖、知名大学教授、图灵奖获得者，同时也招引了全球以“发币”为名的非法募集资金的骗局泛滥。还有一些个人和组织利用加密数字货币监管缺失进行洗钱、非法跨境交易等违法、违规甚至犯罪行为。由此可见，比特币的创新可以被认为是最具争议的信息技术创新。从一开始就把金融与科技紧密地联系在一起，以至于全球近 200 个国家和地区对比特币及后续的加密数字货币采取了包括合法、限制、禁止等不同的法律与监管政策。

锚定法币的加密数字货币被称为稳定币（stablecoins）已经有多年的实践，不仅技术上需要区块链技术的创新支持，同时还需要得到有关国家权威机构的许可。例如，在美国的 USDT 和等。据有关统计，2020 年 12 月稳定币的链上总交易量创下 1783.4 亿美元的新高。

2021 年 1 月 4 日美国货币监理署（OCC）发布一封解释信声明，美国国家银行和联邦储蓄协会可以成为区块链稳定币的运行节点，并将关联的稳定币用于“许可的支付活动”，认为区块链稳定币网络可以作为“更便宜、更快捷、更高效”的支付方式来减轻跨境交易的成本，因此授权银行在汇款期间利用区块链及其稳定币与法定货币之间来回转换，如果银行选择的话，也可以自己发行稳定币（比如，摩根大通银行就提出要推出自己的美元稳定币 JPMcoin 等），这意味着稳定币的监管者与创新者有了新的里程碑，其中区块链技术采用许可区块链（联盟链）技术以满足用户实名、反洗钱、反恐怖主义融资等监管合规要求，并且通过锚定美元单一货币体现美元霸权的扩张。这里的稳定币并非是全部，而是指经过监管部门审查通过并接受其持续监管的稳定币。目前这种稳定币主要包括 USDC、GUSD、PAX 等。

Libra 是 Facebook 推出的虚拟加密货币，最初在设计时，不追求对美元汇率稳定，而追求实际购买力相对稳定，2020 年 12 月更名为 Diem,并准备 2021 年推出与美元挂钩的稳定币。2019 年 6 月 18 日，Facebook 联合全球 28 家企业正式发布 Libra 白皮书。Libra 本身基于联盟区块链技术进行构建，它本身是安全的，具有一定的稳定价值。它的出现，打破了传统意义上货币主权的限制，具有排他性，主权国家对其监管难度大。2020 年发布的白皮书 2.0，声明从锚定一篮子货币变为首先锚定美元货币，区块链技术也变为永久使用联盟链技术以满足监管合规要求。Diem 不仅因其初始“建立一套简单的、无国界的数字货币和为数十亿人服务的金融基础设施”的愿景引起全球瞩目，而且因其采用联盟链技术自主创新、拥抱监管而成为技术标杆。尽管 Diem 尚未推出试用，其创新性不可忽视。此外，Diem 协会 1 月 7 日发文表示，希望在今年某个时候启动此稳定币支付项目，目前正在等待监管部门的批准。预计在 2021 年，将会出现包括 Diem 在内的更多稳定币项目。

5. 以 Diem 为例对比 DCEP 的发行意义

Diem 与现有其它数字货币相比，具有一定的优势。首先 Diem 锚定相关法定货币，实现 1 对 1 兑换。它本身只能通过实体货币交易时才能体现出自身价值，本身的价值并不是凭空产生的。其次 Diem 本质上是基于 BFT 共识机制实现的一种区块链技术，具有去中心化、不可篡改等特征，兼具低延迟等优良特性。其次，Diem 与实体货币挂钩，可获得主权国家背书。Diem 与主权国家货币锚定后，每发行和销毁一个单位 Diem，就要储备和销毁等值货币，防止出现通货膨胀。

Diem 与 DCEP 虽然都属于数字货币，但两者存在较大的区别。首先两者受众对象不同，DCEP 主要服务中国用户，而 Diem 受众群体更广泛，目标是打造成一个全球货币。其次两者之间实现架构不同，DCEP 主要采用中心化的实现架构，而 Diem 在顶层结算模块主要采用了去中心化的实现架构。然后就是两者治理模式不同，对于 DCEP 而言，发行和赎回操作主要由中央银行完成，但分发和支付操作主要委托给第三发完成，而 Diem 治理机构主要由金融领域重要组织和其它非盈利组织构成，具有去中心化特征。然后是用户匿名性，DCEP 需要用户实名注册，基本上不具有匿名性，而 Diem 是一种联盟链，在加入时需要进行授权，通过相应的处理仍能够挖掘出用户身份信息。然后是两者锚定资产不同，DCEP 主要是人民币，而 Diem 是以一篮子货币作为资产储备。最后是两者之间的网络要求不同，DCEP 支持双离线支付，即使双方都不在线，仍能够完成支付，而 Diem 需要网络支撑。

6. 结束语

在当前中美对决场景下，这场无硝烟的战争已经蔓延到数字货币领域，谁能占有先机，谁就能掌控未来金融领域发展的方向。Diem 剑指国际货币体系，可能会对各国自己的货币主权和法币地位构成挑战，引起了各国政府高度关注。虽然 Diem 声称不与各国法定货币展开竞争，只是对法币的一种补充，但客观来讲，Diem 与法币之间竞争不可避免。与 DCEP 相比，Diem 使用范围更广，参与的人群种类更多，产生的影响更为深远。但是 DCEP 也有一定的优势，它可以触及到一些网络基础设施不完善的偏远地区，让贫苦人口能够用上 DCEP，可以有效实现普惠金融，尤其适合于精准扶贫等领域中。

在彰显我国央行数字货币 DCEP 强大威力之余，我们仍需要保持冷静观察和灵活的战略策略：（1）应密切关注 Diem 的进展动态以及其背后的技术基础和创新逻辑。（2）应密切关注加密数字资产在数字经济中的创新意义和作用，研究其背后的技术创新及金融创新的结合；（3）应密切关注区块链技术中智能合约编程语言及其可编程性对未来数字经济的影响，在监管科技、数字政府等方面实现创新引领。

参考文献（略）

超级账本和开源，今天和明天企业级区块链

——在《第十五届开源中国开源世界高峰论坛》（2020 线上会议）上的报告

Brian Behlendorf

区块链超级记账(Hyperledger)基金会执行董事

大家好！我是 Brian Behlendorf，Hyperledger 超级账本的执行董事，超级账本是 Linux 基金会主导的项目。今天我非常高兴能在这里同大家交流，参加“开源中国开源世界”高峰论坛，论坛由中国开源软件推进联盟举办的，我已经受邀在该组织做了超过 15 年演讲，每次我来到中国，看到中国开源社区都会有着特别可喜的进步，真的很高兴能再次在线上和大家交流，我其实希望能和大家面对面交流，但如今形势和以往大不一样，我希望，大家能在我的演讲中知道一些区块链社区相关的新进展，也许我们还有机会可以共同合作。

今天,我非常高兴能给大家多讲讲超级账本，超级账本是 Linux 基金会主导的项目，超级账本关注于开源软件，构建分布式账本网络，大家可以称之为企业级区块链网络，在很多方面同比特币和以太坊很像，但它用的是分布式账本，在中国受到高度重视，并且中国政府也很支持，习近平主席表示[2]要加快推动产业创新发展企业级区块链，支持各行业的应用。同时习主席也表示中国要把区块链作为核心技术自主创新的重要突破口，明确主攻方向，加大投入力度，着力攻克一批关键核心技术，加快推动区块链技术发展，加速产业创新发展。

区块链技术已经成为许多国家布局的关键领域，并且对其敞开大门。大力推进该技术，推动国家产业发展，推动经济增长，这是一项伟大的技术。它将市场规则整合在一起，并推动规则的执行，它使得经济系统能够自查。

我认为这对于中国来说是非常重要的，对于整个世界而言也是如此，我相信大家看到区块链技术应用的落地案例，一定能令大家叹为观止，在超级账本中，我们对企业区块链技术采取了宽泛的处理方式，大家可以把区块链

[2]（引用自新华网“2019 年 10 月 24 日下午，中共中央政治局就区块链技术发展现状和趋势进行第十八次集体学习。”）

想成非常特别的数据库。它可以延伸到整个网络，任何人都能读取和写入，但它也需要遵循一定的规则，因此有许多不同的方式来构建数据库，例如在世界其他地区有外部区块链技术，有 MySQL 或 MongoDB 这样的数据库，它们之间差异很大、运作方式也不同。

因此在超级账本中，我们很早就决定以一个保护伞或者一个温室下开展，我们会有许多不同的技术项目，利用不同的方法，构建企业区块链平台，在分布式账本项目中，我们有六个重点项目，它们是在分布式网络上的节点执行的软件。因此 如果你正在建立分布式供应链，或是分布式付款网络，或分布式存货管理系统。

在这一网络中的所有，都会在其系统中使用该软件，其他人也会得到通知。在此范畴内，我们的核心项目包括 Hyperledger Fabric，它是目前最受欢迎的企业区块链平台，中国和世界其他国家都会使用该平台，之后我会详述这一点。

我们的项目也包括 Hyperledger Sawtooth，它使用完全不同的方法建立这些分布式系统，非常值得大家花时间了解，Hyperledger Indy 则是关注分布式数字身份可验证凭据，还有 Hyperledger Besu 是一个拨人心弦的项目，它是很新的项目，1 年多前开始，它的关注是将以太坊生态系统的技术应用于企业之中。因此 Hyperledger Besu 既能在许可区块链 (permissioned blockchain)中运行，也能在公有链中运行，因此很多活动都是借助它完成的 ，互相交织。

我们有程序库项目，这是与加密管理相关，运行在智能合药系统，诸如此类。我们还有一些非常棒的工具，如 Hyperledger Caliper 是区块链性能基准测试的工具， 我们在中国社区的成员，为社区做了许多贡献。Hyperledger Avalon 专注隐私和保密性，因此若想实现安全私人交易，可以在一个共同的链上建立。

另外，Hyperledger Cactus 是我们刚发布的最新项目，致力于将这些不同的区块链网络进行整合，这样就可以在当中进行交易，或制定智能合药，合药可跨不同区块链网络执行，社区许多活动都在迅速发展壮大，许多新的开发者也纷纷加入，我们诚邀各位也能参与进来、尝试一下，

现在我想重点说一下 Hyperledger Fabric，因为它是目前最成熟的技术项目，Hyperledger Fabric 现在已经部署在世界上所有主流云服务供应商上了，这当然包括中国的云服务供应商，例如蚂蚁金融、百度、华为、京东、联想、腾讯等云服务供应商，在国外的云服务供应商有亚马逊网络服务和 IBM、微软、SAP、谷歌等。当然，这些公司都能共同合作，在同一系列的区块链网络中，所以使用这些技术不存在供应商锁定或云锁定 ，我刚提到的 Hyperledger Fabric 是目前应用最广泛的平台，每年《福布斯》公布全球区块链 50 强榜单（Blockchain 50），他们会和 50 家非常庞大规模的企业交流 ，这些企业每年的收入超过 10 亿美元，你可能会问这些企业使用什么区块链技术，在这 50 家企业中有一半以上的公司，都在使用 Hyperledger Fabric 区块链平台，其中许多公司也在使用其他超级账本技术运行项目，在此很高兴看到这项技术得到广泛应用，从蚂蚁金服、腾讯 、戴姆勒、霍尼韦尔、汇丰和嘉吉等企业，在汽车领域、医疗保健、供应链等众多不同用落地案例中采用超级账本区块链技术。

当然，其中有许多是在金融服务领域，贸易融资是一个非常大的领域，在中国有很多公司是超级账本的会员，这些公司在其基础上构建产品和服务，我在此提到过其中很多公司。比如，华为是我们的会员，做出了非常大的贡献，我想强调一下，你也可以看到大量的初创公司和中型公司，它们是协作环境的核心部分，像我们其中一家中国会员 eVisible 易见股份，在社区中非常活跃，产生了很多非常有趣的代码，我还想提一下我们正在与 CAICT 等组织合作，为中国区块链技术产业发展提供政策支撑。另外，超级账本的联盟会员北京大学，浙江大学和香港大学等，他们有大量的学生和研究工作，投入到区块链技术中，这确实有助于推动整个超级账本社区，在超级账本社区中， 这对我们来说真的很重要。

在过去四年里对我们来说非常重要的是，要确保这是一个全球协作的全球项目，所以我们从一开始就致力于在中国构建开发者社区，并确保他们能够成为项目的合作者，即使他们不熟悉英语，而英语却是开发代码的通用语言，我们意识到要在中国有一个本地工作组，专注于帮助那些开发者从仅仅使用代码到能够贡献代码。

IBM 的郭剑南，MediConcen 的刘宇翔和杭州趣链的宣章炯都是超级账本在中国技术工作组的联席组长，对中国社区提供了极大的帮助，他们凝聚中国社区，Linux 基金会与超级账本也非常专注，构建商业生态系统。

同时我们最近推出了一个经过验证的超级账本认证服务供应商计划，目前有 12 家会员公司成为超级账本认证服务供应商，其中 4 家是来自中国的会员公司，这些会员公司具有成功帮助企业采用超级账本技术的深度经验，他们真正知道如何帮助公司构建区块链网络，如果你是一家公司，想在这个领域投入更多资金，想要了解该如何启动超级账本技术项目，列表中的会员公司可以帮助构建区块链环境。

现在有大量的项目都是建立在超级账本技术之上，尤其是在中国，很抱歉，我不能对所有的超级账本技术项目进行深入介绍，这些是国内和国外一些公司，它们真正构建出了一些有趣的产品和服务，我想特别强调，中国建设银行和中国民生银行的项目，它们建立了一个非常非常庞大的贸易融资系统，在 Hyperledger Fabric 上运行，每周处理数十亿人民币的交易，以及贸易金融信用证之类的东西。另外，腾讯云构建了一个有趣的应用程序，涉及到管理仓库中的库存，并能够将其作为贷款的基础，这真的很酷。另一家名叫真相科技，做司法联盟链的初创公司，用区块链极大地提高了中国线上法院系统的自动化和可验证性。能将这项技术应用于此真是令人惊叹，所有这些都不可能实现，如果不是中国的会员帮助我们构建这项技术，然后将其部署于各种应用案例。

我们非常感谢中国社区对我们的回馈，我真的很想多花点时间来讲。我们的亚太区总部设在香港，我们的团队帮助建立、招募开发者和推进社区，欢迎和我们保持联系。

现在是最后一张幻灯片，我非常渴望与大家合作，帮助大家了解更多有关超级账本的信息，并了解作为开发者如何参与社区，或者作为一家公司可以如何使用此技术，甚至在此技术上构建产品或服务，希望大家能加入超级账本社区。

谢谢大家！

Hyperledger and Open Source: Enterprise Grade Blockchain for Today and Tomorrow

(Speech on 15th OCOW Summit)

Brian Behlendorf

Executive Director, Hyperledger, The Linux Foundation

Hello everyone, my name is Brian Behlendorf and I'm executive director of Hyperledger an initiative hosted at the Linux foundation. I'm very excited to be speaking with you today at the Open Source China Open Source World Summit hosted by COPU who is a great organization. I've been giving talks with this organization for over 15 years and every time I come to China there's always an exciting degree of growth in the open source community here. And so it's really exciting to be here again and to be here virtually with all of you, I wish I could be there in person of course, And the conditions are different today, But hopefully you'll hear something new in my talk and we'll find perhaps some opportunity to collaborate together. I'm excited to tell you today more about Hyperledger. Hyperledger is an initiative under the Linux Foundation and stepping into my next slide, Hyperledger is really all about open source software for building distributed ledger networks you could call them enterprise block chain networks in many ways, they work very similarly to Bitcoin or Ethereum but it very much is the flavor of distributed ledger that is in very well supported here in China, in fact there is support even from the Top. President Xi Jinping has been very supportive of the importance and the industry adoption of enterprise block chain technology for all sorts of purposes. 3In fact he says that China must take blockchain as an important breakthrough for independent innovation of core technologies clarifying the main direction increase investment focused on a number of key core technologies and accelerate the development of blockchain technology and industrial innovation this is really a technology that countries should embrace with both hands and dive in and adopt for the growth of their industries for the growth of their economies, this is a great technology for weaving together these marketplace rules are in a way that automatically enforce those rules. it allows for these economic systems to be self-auditing and I think that's something that's very important here in China but also very important globally as well. And I think you'll see some of these cases that can be applied to as very very compelling.

so, in Hyperledger ,we take a very broad approach to enterprise blockchain technology, one way to think of blockchains is as a very special kind of database

[3] （引用自新华网“2019 年 10 月 24 日下午，中共中央政治局就区块链技术发展现状和趋势进行第十八次集体学习。”）

but it's a database that spread out and network and anybody on that network can read from it and anybody can write to it. But their rights must follow certain rules. And so there are many different ways to build these kinds of databases. We know for example in the rest of the world you have outside blockchain technology, you have databases like MySQL and MongoDB. And they are very different from each other and how they work. So within Hyperledger, we decided very early that we would be an umbrella or what we call the greenhouse where we would have a number of different technology projects aimed at different approaches to building enterprise block chain platforms, the key six projects are what we call the distributed ledger projects. These are focused on software to run a node on a distributed network. So if you are building a distributed supply chain or a distributed payments network or distributed inventory control system, this is software that everybody in that network runs on their own systems but then it talks to everybody else. Some of our key projects within this category are projects like Hyperledger Fabric which is the most widely used enterprise blockchain platform out there today, both here in China as well as in the rest of the world.

And I'll talk more about that in a bit, But it also includes projects like Hyperledger Sawtooth which has a very different approach to building these kinds of distributed systems that are very much worth your time to learn more about Hyperledger Indy which is very much focused on distributed digital identity and verifiable credentials, as well as Hyperledger Besu that is really exciting, because it is one of our more recent projects that came in about a year ago. And it's very much focused on applying the technologies that work in the Ethereum ecosystem into the enterprise, and so Hyperledger Besu can run both on a permissioned blockchain as well as a public blockchain and so lots of activity happening there and cutting across these.

We have projects that are library kinds of projects around cryptography and around running smart contract systems that sort of thing. We also have some tools that we think are very compelling. A bench marking tool called Hyperledger Caliper which has a tremendous amount of contributions from our community members there in China.

Hyperledger Avalon which is focused on privacy and confidentiality so you can build secure and private transactions on top of a common blockchain, as well as our most recent project Hyperledger Cactus which is really focused on being able to weave these different blockchain networks together into, so that you can do transactions across these or write right smart contracts, that can operate across different blockchain networks. Lots of activity going on the community is expanding very quickly, lots of new developers coming in. we really invite everybody to come in and have a chance.

Hyperledger Fabric in particular, I want to highlight, because it is again most widely adopted out there every major cloud computing environment in the world

now offers supported Hyperledger Fabric as a service offering. And that includes of course are the ones that are in China, Tencent and Ant Financial ,Baidu as well as JD,Lenovo has a cloud offering there and this is also very compatible with the clouds outside of China South the clouds at Amazon web services and IBM, Microsoft,SAP,Google Clouds . Of course, all these companies now can work together on a common series of block chain networks. So, there is no vendor lock in or clouds lock in to using these technologies.

Hyperledger Fabric as I mentioned is the most widely used platform out there, there is a report that Forbes does every year called the Blockchain 50. Where they talk to 50 very large companies with more than a billion dollars in revenue each year, and ask them what are the block chain technologies you're using. And out of 50 more than half of them are running projects on top of Hyperledger Fabric. And many of them are also running projects with other Hyperledger technology. but it's exciting to see the diverse range of adoption of this technology from Ant Financial and Tencent to Daimler and Honeywell and HSBC and Cargill in so many different use cases all around the map in healthcare and supply chain in the out of motor space. of course, many many of them in the financial services space. Trade finance being a very big one.

In fact, there in China, we have a very large number of companies as members of Hyperledger, these are companies who are building products and services on top of them, I mentioned many of them, Huawei is a member it has made very large number of contributions. I want to highlight as well， you also see a large number of startups and mid-size companies as well, who are a core part of the collaborative environment. Companies like eVisible, who are very active inside of the community and produce a lot of interesting code. I also want to mention where we are collaborating with organizations like CAICT, which oversees a lot of the blockchain policy for the Government of China as well as Peking University and the University of HK and Zhejiang University. where there's a tremendous amount of student and research work going into the blockchain technology that's really helping feed the entire Hyperledger.

In Hyperledger it was really important to us, it has been important to us for the last four years to make sure that this is a global project with global collaboration. So, we from the very beginning have focused on building the developer community in China and making sure that they are empowered as collaborators on the project even if they struggle with the English language which is the common language for development code. we realized to support have a local working group there in China, focused on helping those developers across the bridge from just using the code to being able to contribute to the code. So, Jay Guo from IBM, David Liu from MediConcen and John Xuan from Hyperchain have all been leaders of that China technical working group and have been extremely helpful and pulling together the local Chinese community.

we also though have a really big focus at the Linux Foundation of course and also at Hyperledger in building commercial ecosystems. so we recently launched a verified Hyperledger certified service provider program and we have 12 participants in that program now, four of those are companies based in China. And these are companies who have qualified developers and administrators working for them, who really know how to help you build your blockchain networks.

So if you're a company and you want to invest more in this space, you want to understand how this works potentially launch a project, this is the chart list of companies in an approach to help you build your block chain environment, there are a large number of projects are building on top of Hyperledger now, especially based there in China.

I don't have time to go into depth on all of them, These are a few of the companies both inside of China and outside, who have really built some interesting products and services. I want to highlight in particular , China Construction Bank as well as China Minsheng Bank, which had built a very very large trade finance system, now running on top of Hyperledger Fabric processing billions and billions of yuan of deals over each week and trade financial letters of credit and those sorts of things as well. Tencent Cloud has built an interesting application involved with managing inventory in their warehouse and being able to use that as the basis for loans really cool stuff. And LegalXchain is using it, as a way to verify the integrity of evidence created for the internet court system being used there in China, which is really amazing and cool application of this technology.

All this wouldn't be possible, if it weren't for the members there in China, who are helping us build the technology and then who are deploying this for all sorts of use cases.

So, we're incredibly appreciative of the response that we've received from China, I'm really eager to spend more time. We have a team that's based in HK to go and help build and recruit developers and companies there, So please be in touch I just on the last slide now, very eager to work with you and help you all of you watching this learn more about Hyperledger and learn how to get involved in the community as a developer ,or as a company thinking about using this technology. Maybe even building a product or service on Top of the technology. We'd love to have you as a part of the community.

Thank you very much.

COPU 编者的话：

中国科学家突破区块链关键技术：异步共识算法

（“小飞象拜占庭容错/Dumbo BFT 算法”）

中科院软件所研究员张晓峰团队联合美国新泽西理工学院唐强团队，在区块链核心技术拜占庭容错（BFT）共识研究中取得重要突破，在国际上提出首个完全实用的异步共识算法“拜占庭容错（ Dumbo BFT）算法”（简称“小飞象算法”）。

区块链领域这一重大突破性成果，近日在网络安全旗舰会议——第 27 届国际计算机与通信安全大会上发表，这也是在异步 BFT 共识算法设计领域，中国科学家首次在国际顶级会议上发表的重要研究成果。张晓峰研究员同意在本期“深度信息技术（精品）专辑（区块链）”上发表。他认为作为区块钱链关键核心技术 BFT 共识算法是确保区块链安全可靠运作、提升区块钱链扩展能力和运行性能的核心算法。BFT 共识算法具有运行性能高、资源消耗低、易于部署等特点，得到工业界的青睐，广泛应用于中外区块链系统中。异步 BFT 算法能够容忍网络通信故障，抵抗拜占庭敌手恶意攻击，是保障区块链在互联网环境下健壮运行的理想共识技术。

Dumbo: Faster Asynchronous BFT Protocols

——在第 27 界国际计算机与通信安全大会上的报告[4]

中科院软件所 张振峰

Dumbo: Faster Asynchronous BFT Protocols

Bingyong Guo*†
Institute of Software
Chinese Academy of Sciences
guobingyong@tca.iscas.ac.cn

Zhenliang Lu†
Department of Computer Science
New Jersey Institute of Technology
zl425@njit.edu

Qiang Tang†
Department of Computer Science
New Jersey Institute of Technology
qiang@njit.edu

Jing Xu†
Institute of Software
Chinese Academy of Sciences
xujing@tca.iscas.ac.cn

Zhenfeng Zhang†
Institute of Software
Chinese Academy of Sciences
zfzhang@tca.iscas.ac.cn

ABSTRACT

HoneyBadgerBFT, proposed by Miller et al. [34] as the first practical asynchronous atomic broadcast protocol, demonstrated impressive performance. The core of HoneyBadgerBFT (HB-BFT) is to achieve batching consensus using asynchronous common subset protocol (ACS) of Ben-Or et al., constituted with n reliable broadcast protocol (RBC) to have each node propose its input, followed by n asynchronous binary agreement protocol (ABA) to make a decision for each proposed value (n is the total number of nodes).

In this paper, we propose two new atomic broadcast protocols (called Dumbo1, Dumbo2) both of which have asymptotically and practically better efficiency. In particular, the ACS of Dumbo1 only runs a small κ (independent of n) instances of ABA, while that of Dumbo2 further reduces it to constant! At the core of our techniques are two major observations: (1) reducing the number of ABA instances significantly improves efficiency; and (2) using multi-valued validated Byzantine agreement (MVBA) which was considered suboptimal for ACS in [34] in a more careful way could actually lead to a much more efficient ACS.

We implement both Dumbo1, Dumbo2 and deploy them as well as HB-BFT on 100 Amazon EC2 t2.medium instances uniformly distributed throughout 10 different regions across the globe, and run extensive experiments in the same environments. The experimental results show that our protocols achieve multi-fold improvements over HoneyBadgerBFT on both latency and throughput, especially when the system scale becomes moderately large.

CCS CONCEPTS

• **Security and privacy** → Distributed systems security; • **Computer systems organization** → Reliability; Availability.

ACM Reference Format:
Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. 2020. Dumbo: Faster Asynchronous BFT Protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3372297.3417262

1 INTRODUCTION

Byzantine fault tolerant (BFT) protocols enable a set of untrusted peers to reach consensus. As one fundamental research area in distributed computing, the problem has been extensively studied and many variants exist for different application scenarios. One main categorization of the BFT protocols is based on the timing (or network) assumptions. A synchronous BFT protocol assumes all values sent by honest peers will be delivered to the recipients within a certain period of time, which is known to everyone (including the protocol designer). While a partially synchronous BFT protocol relaxes this network requirement by allowing the time bound to be exist but unknown. An asynchronous BFT protocol relies the least on the network assumption that it does not require such a time bound to exist, just that all values will eventually be delivered.

The favored asynchronous BFT. In the earlier years, considering the deployment of BFT protocols mostly in conventional in-house scenarios that the peers are well-connected, research efforts of BFT focused on reducing (cryptographic) computations [5, 40], or increasing the threshold of malicious participants the protocol can tolerate [21], assuming a synchronous network. Nice works on synchronous setting continue to emerge in recent years [1, 11, 32]. Efforts also exist relaxing the network synchrony assumption. One notable example is the classic Practical Byzantine Fault Tolerance (PBFT) protocol [18] that requires partial synchrony. However, removing the synchrony assumption completely in practice becomes more and more desirable for both robustness and efficiency reasons.

Recently, the success of cryptocurrencies and blockchain technology in general brings much broader application scenarios to the BFT protocols, and also demonstrates the possibility of consensuses over wide-area network (WAN). The open Internet environment provides a more adversarial setting that the network latency among

*Bingyong Guo is also with State Key Laboratory of Cryptology and School of Computer Science and Technology, University of Chinese Academy of Sciences.
†Authors are named alphabetically, and the first two authors contributed equally. All authors are also with JDD-NJIT-ISCAS Joint Blockchain Lab.

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3417262

[4] 2021 年 2 月 23 日张振峰研究员致信 COPU 副主席兼秘书长刘澎：刘老师好！这是小飞象共识算法英文论文，请指教！并附“Dumbo: Faster Asynchronous BFT Protocols”英文原文。

the peers could be time-varying. However, the synchronous (or partially synchronous) BFT can only perform in the relatively "private" network with well-connected nodes that guarantees network delivery within certain time bound. Those protocols would fail to make progress and get stagnated if the timing assumption does not hold. Indeed, it was shown formally in recent work [34] that PBFT cannot make any progress in "intermittently synchronous network", where the adversary only chooses to delay messages at certain time points. The "attack" could similarly be applied to a class of leader-based BFT protocols [4, 5, 10, 19, 20, 41].

Another important reason that asynchronous protocols maybe favorable is due to efficiency, particularly a property called *responsiveness*. A synchronous BFT protocol, when designed, is parameterized by the *assumed* network latency, which is normally chosen to be large so that the actual network latency is indeed smaller thus the synchrony assumption can be ensured. As a consequence, the efficiency of most of the synchronous BFT protocols depends on the assumed network latency. While "responsiveness" instead requires the performance is only related to the *actual* network latency, thus it should not rely on any timing assumption and the protocol makes progress as soon as messages are delivered.

Moreover, it is well-known that asynchronous protocols simplify the engineering efforts substantially when actually building the distribute system, as no time-out mechanism will be needed. While building a system implementing a synchronous protocol, one should design all kinds of ad-hoc, error-prone time-out mechanisms.

The first practical asynchronous BFT [34]. Even though it is preferable or even necessary in many cases when deployed in real-world WAN, most of the previous researches on asynchronous BFT are theoretical in nature until the first practical protocol HoneyBadgerBFT was proposed in [34]. Previous asynchronous BFT protocols normally are inefficient, e.g., having a high (per message) communication complexity (up to $O(n^2)$ or even $O(n^3)$ if there are n peers) [2, 9, 15, 17, 26, 40]. The performance of these protocols will drop sharply when the system scales up. The elegant work of HoneyBadgerBFT [34], on the other hand, made several critical observations to push asynchronous BFT towards being practical.

The first observation is that an atomic broadcast protocol which is a continuous execution of BFT protocols maintaining an ever-growing log, (or to put it another way, regular BFT protocols can be considered as a one-shot instance of it), could be built very lightly from a weaker variant called asynchronous common subset (ACS) together with a threshold encryption scheme. An ACS protocol only requires peers to agree on a subset of all their inputs and was originally proposed for different purposes [7].

More importantly, it was observed in [34] that the classic ACS protocol from Ben-Or et al. [9] is much more promising for efficiency both asymptotically and practically (than another related protocol called multi-valued validated Byzantine agreement (MVBA) [15], to be further explained soon), when picking the underlying building blocks carefully. The ACS protocol from [9, 34] was built from two sub-protocols: *reliable broadcast* (RBC) and *asynchronous binary agreement* (ABA). The structure is fairly simple: each node invokes an RBC to broadcast its input value, and participates in n instances of the ABA protocol to agree on which subset of inputs to include, see Figure 1.

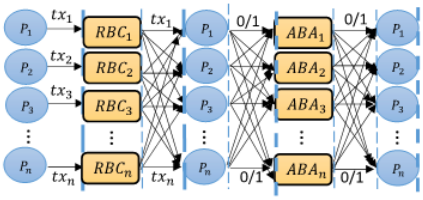


Figure 1: The structure of ACS in HoneyBadgerBFT [34]

Experimental results show impressive performance of HoneyBadgerBFT. In a nice work BEAT [22], the authors gave an extensive study about most suitable instantiations of the building blocks for HB-BFT (while keeping the protocol structure intact) when considering diverse deployment scenarios. [1] We take a different path and ask the following question:

Can we redesign the ACS protocol to improve both its asymptotic and practical efficiency?

1.1 Our contributions

We design two new ACS protocols, both of which improve the running time asymptotically and practically. Our experimental results demonstrate a multi-fold improvements over HoneyBadgerBFT [34] when they are run back to back in the same environment on Amazon AWS. More interestingly, our two main observations (1. the number of ABA instances should be reduced; 2. MVBA would be more efficient if used carefully for ACS) that lead to our two protocols would be of independent interests. Let us elaborate in more detail below.

Dumbo1: a faster asynchronous BFT. We first go over the structure of the ACS protocol used in HB-BFT in slightly more details: first each peer broadcasts its input via an RBC instance; whenever a peer receives value from peer P_i, it sets its input for the i-th ABA instance to be 1 and starts the ABA protocol. Once an honest peer has got 1 from $n - f$ ABA instances, it will input 0 to all the remaining ABA instances which have not input yet and move on.

Identifying the major bottleneck. Due to the famous FLP impossibility [23], an ABA must be a randomized protocol. This brings in the following drawback: though the expected number of "rounds" of each ABA protocol is constant, the expected number of rounds of running n concurrent ABA sessions could be significant, i.e., at least $O(\log n)$ [8] More seriously, those ABA instances do not really execute in a fully concurrent fashion: as (1) not all instances start at the same time, some of the instances may start later as inputs of (the previous RBC) haven't been delivered; (2) normal node also has an efficiency degradation facing large scale concurrent execution (not enough CPU cores etc). When n gets larger, and the network is unstable, there would likely be some ABA instances that terminate very slowly. The slowest ABA instance determines the running time of the ACS of HoneyBadgerBFT.

[1] We remark here that most of the techniques of BEAT [22] can be directly applied to our protocols as well, to choose more suitable instantiations of the underlying building blocks such as RBC, and further optimize the performance. Our focus is to show asymptotic and practical improvements due to **protocol redesign**, so we mainly compare a **basic** instantiation of our protocols with HoneyBadgerBFT in experiments. See more comparison in section 1.2.

To see the practical impact of ABA protocols on the performance, we carry out experiments of HB-BFT and do statistics about the average running time between RBC and ABA. As shown in Figure 2, it is clear that for HB-BFT, the cost of ABA is dominating [2]. The pattern becomes even more significant when the scale of the system grows. This simple observation inspires us to reduce the number of ABA instances needed in the ACS protocol.

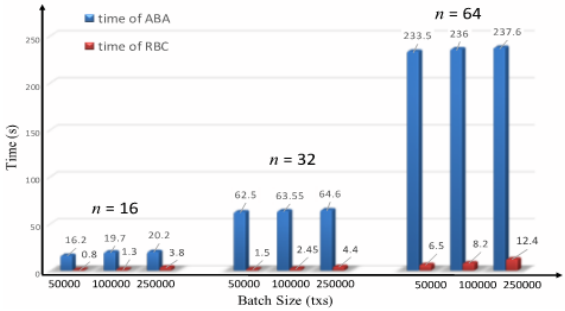


Figure 2: Time costs of RBC and ABA in HoneyBadgerBFT, we get the running time of RBC by starting a timer when protocol starts and ending the timer when nodes get the output of RBC, averaging among multiple nodes and instances. The time of ABA is taking the maximum among all ABA instances a node needs to run.

Reducing # of ABA instances. We redesign the structure of ACS, and propose Dumbo1-ACS. Different with the HoneyBadgerBFT (and also the BEAT protocols), Dumbo1-ACS only needs to run κ instead of all the n ABA instances, and achieves $O(\log \kappa)$ running time, where κ is a security parameter independent of n. Other complexity metrics remain the same.

In a simplified view, the first phase remains unchanged: every node broadcasts its input through an RBC instance. Then, imagine that if we have one honest node to take the role as the leader, then it can first finish $n - f$ RBC instances and then informs all other nodes to output the deliveries of these RBC instances. To get such an honest node, we can select a small number κ of nodes as the "leaders" such that at least *one of them* is honest with an overwhelming probability.

Further care is needed as now two honest nodes may receive different values from different selected "leaders". Next, we should enable honest nodes to decide which of the κ selected nodes to believe. It actually becomes similar to HB-BFT that we can invoke ABA instances to confirm whose nomination of subset to include. Once some ABA instances output 1, the corresponding messages can be identified and output. Importantly, now the peers just need to agree on the κ (which could be much smaller than n) nodes. See pictorial illustration in Fig. 3.

We would like to stress that the changes in the remaining parts are kept at minimal so that the reduction of ABA instances also yields significant practical improvements. It looks like we have

[2]We ignored some "unnoticeable" time cost of local computations such as threshold encryption/decryption, picking an input from the buffer etc as they do not change the ratio much.

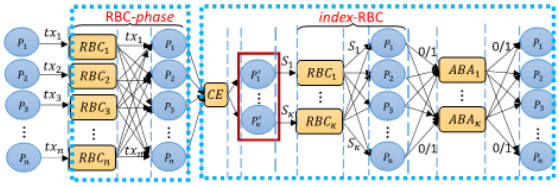


Figure 3: The structure of Dumbo1-ACS

a handful extra RBC instances than HB-BFT; however, we make the input of each peer in those extra (index)-RBC instances to be a tiny index-set (S_i instead of the actual data loads). An honest player inputs 1 for the i-th ABA if he indeed receives all messages corresponding to S_i. Moreover, the added coin-tossing protocol to select κ nodes is just a sub-routine of ABA protocol. So those added overhead would be unnoticeable compared to the cost of eliminated ABA instances.

To see why it works: when one honest node determines to output the values corresponding to S_i, it must be the case that the i-th ABA instance outputs a bit 1. The property of ABA ensures that (1) all other honest nodes will also output 1 and (2) at least one honest node inputs 1 for this ABA instance. The latter means at least one honest node indeed receives all the input values $\{v_j\}$ corresponding to the index set S_i. Thus following the security of RBC, all other honest nodes will also receive those values eventually. While condition (1) ensures all honest nodes will actually output the same subset of values.

Dumbo2: an even faster asynchronous BFT. Dumbo1 now runs only κ concurrent ABA instances, we now ask a more ambitious question: can we push it all the way to constant?

Pushing # of ABA to minimum. HoneybadgerBFT requires n executions of ABA instances, due to the fact that each ABA instance determines only for input from one peer. Dumbo1 can reduce it as now the "committee" members are prepared with a vector of values. But still, Dumbo1 needs to run κ instances: the procedure after the RBC phase is very similar to the structure of HB-BFT, that picks a common subset containing the index-sets $\{S_i\}$ as elements. Since each node will invoke/enter the i-th ABA instance once it receives S_i from the i-th committee and all values corresponding to the S_i. This causes a challenge that different nodes may enter different ABA instances, there is no "global coordinator" for those instances, thus the only viable way is to concurrently run all of them.

In principle, we still "waste" $\kappa - 1$ ABA instances. This inspires us to find a way to correctly identify only one input vector, thus leads us to re-examine the applicability of multi-value validated Byzantine agreement (MVBA), which outputs one of the inputs of n peers as long as the input satisfies some pre-defined predicate. MVBA was considered impractical for building ACS in [34]. The reason was that existing constructions suffer from a high communication complexity, i.e., the MVBA protocol in [15] has communication complexity $O(n^2|m| + \lambda n^2 + n^3)$ in expectation [3], where $|m|$ represents the size of MVBA's input values. In many cases, $|m| > \lambda n \log n$, thus the dominating term in the per message communication of

[3]In a very recent work, we brought down this complexity to $O(n|m| + \lambda n^2)$ [29]

its direct construction of ACS [15] is $O(n^2|m|) = \Omega(n^3)$ (Although [15] did not explicitly mention ACS, their atomic broadcast already contains the construction from MVBA to ACS, and the complexity remains even for the recently improved MVBA [2]) and makes the MVBA protocol impractical for building ACS.

But the above claim holds only when MVBA is directly invoked with large size inputs. If we give a closer look, we notice that if $|m|$ is small, then the overall communication complexity of MVBA (and also the corresponding ACS [15]) is no bigger or even substantially smaller than the ACS in HoneyBadgerBFT ! [4] See Table 1 in Sec. 6. And MVBA has the benefit of a constant number of ABA instances [15]. The key challenge is now reduced to how to invoke MVBA with small inputs to construct an ACS which *may still have large inputs*. This reminds us the widely used conventional wisdom of *"Hybrid Encryption"* in the setting of cryptography.

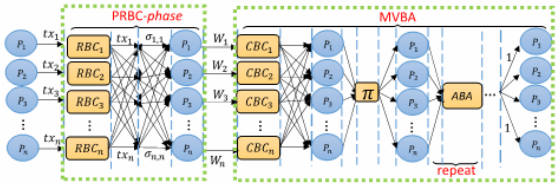


Figure 4: The structure of Dumbo2-ACS

The right way of applying MVBA. We present an even faster asynchronous BFT protocol via an innovative use of MVBA, we call it Dumbo2. It achieves asymptotically optimal (constant) running time, i.e., Dumbo2 only needs to run (expected) three consecutive instances of ABA, and other complexities remain the same [5].

To work out the details of ACS requires further ideas. Since ACS outputs a subset of inputs, we would first prepare each peer node with a vector of inputs via RBC type of protocols. More importantly, instead of feeding those message vectors into the MVBA protocol, we further prepare each peer with a very *short* "indicator" (the W_i in Fig. 4) and use it as input to join the MVBA protocol. The MVBA protocol will output one such "indicator" which would be used to inform each honest peer to pick the corresponding RBC instances. The tricky part is, in an MVBA protocol, honest peers may output the input (here the short "indicator") from a malicious peer.

We resolve this by designing the "indicator" in a way that any of it serves as a warrant all honest peers would receive the corresponding messages. We formulate a new primitive called *provable reliable broadcast* (PRBC) which augments RBC and further outputs a succinct proof (even by a malicious node) that at least one honest peer has received the input. This can be realized by threshold signing on the RBC index. The ABA within MVBA only needs to be repeated (expected) three times. See Figure 4 for pictorial illustration, where π is a random permutation.

To see why it works: the actual inputs W_i to MVBA includes a indices set and the corresponding proofs. When one honest node outputs W_i, the proof in W_i is valid. This means the messages corresponding to the indices in W_i were all received by enough peers which include at least one honest peer. Then all other honest node will eventually receive those as well.

We remark that though Dumbo2 out-performs Dumbo1 in most of the cases, we choose to keep Dumbo1 for clarity of the presentation: the idea of using each ABA to vote whether to output the *vector* of each "committee" member in Dumbo1, instead of each input as in HB-BFT is simple and intuitive. Such a possibility of more effective voting could be viewed as a stepping stone to motivate the idea of voting to output only one guy's vector, which eventually leads to Dumbo2's idea of using MVBA. Also, since MVBA is still fairly complicated, in some benign cases when f is very small, Dumbo2 might not be better than Dumbo1.

Implementation and experimental evaluations. Besides the asymptotic improvements (see Table 1 in Sec. 6), we implement Dumbo1 and Dumbo2, and also test the performance of the our schemes in the practical WAN environment. We deploy Dumbo1, Dumbo2 and HoneyBadgerBFT on 100 Amazon EC2 t2.medium instances uniformly distributed from 10 different regions across the globe. For a fair comparison, we use the same language and cryptography libraries as [34], and carry out a variety of tests in the same environment. Results show that the efficiency of our schemes indeed outperforms HB-BFT by multifolds, especially when the system is sufficiently large. For example, when $n = 100$, Dumbo1 has a basic latency that is only 22% and Dumbo2 has only 5% of that of HB-BFT. Moreover, Dumbo1 has a peak throughput 3.5× and Dumbo2 has more than 9× of that of HB-BFT. See more details about more tests in Sec. 7.

To showcase the effectiveness of our observation to reduce the number of ABA instances, we pick the result (the running time recorded of each sub protocol from a random node) from one experiment where $n = 32$ with 10^5 transactions (250 bytes each) as input, see Fig. 5. In the figure, for each protocol each line denotes the execution of a sub-protocol instance, e.g., the two bars in the first line of HB-BFT correspond to the first instances RBC_1 and ABA_1 respectively, the second line corresponds to the second instances $\text{RBC}_2/\text{ABA}_2$, and so on. The *consistent broadcast* (CBC) protocol in Dumbo2 is a part of MVBA, which can be regarded as a simplified version of RBC (see appendix for detailed definition).

1.2 Related work

The consensus problem was firstly introduced by Shostak, Pease and Lamport [28]. As a fundamental problem in distributed computing, it has received extensive attention such that many different variants of the consensus problem have been studied, e.g., [18, 25, 27, 38].

[4] Similar phenomenon was also noticed in BEAT [22] that they chose a seemingly more expensive RBC in BEAT1,2, but getting a more efficient protocol for small message.

[5] There exist theoretical works [8] that can achieve constant expected running time for n concurrent execution of ABA protocol, but at the cost of larger message and communication complexities; As pointed out in [34], if we directly adopt their technique to construct ACS, the message and communication complexities will be $O(n^4)$, which render the ACS infeasible for practice.

[6] For Dumbo2 experiments, we intentionally run ABA more times to "simulate" potential adversarial network scheduler (otherwise there could be only one ABA), while the experiments of HB-BFT, Dumbo1 are done without scheduler intervention (same as before [34]). We also note that in theory, in some very rare cases, it may be possible that some RBC instance gets slow so that users have to wait after ABA instances are finished. We do not observe the case that an RBC takes longer time than the slowest ABA in the experiments. Further optimizations beyond reducing # of ABA instances for the asynchronous atomic broadcast are interesting open problems.

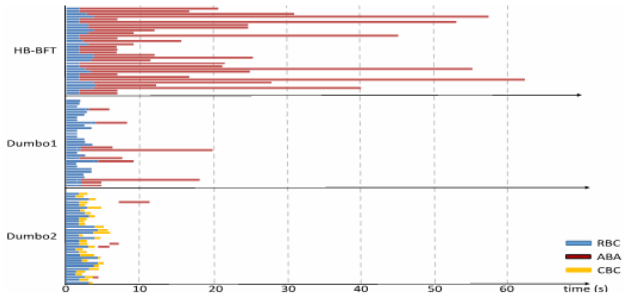


Figure 5: Running time breakdown of Dumbo1/2 and HoneyBadgerBFT on one random node [6].

Classic research on asynchronous BFT focused more on understanding the theoretical limits and feasibilities. The famous FLP-impossibility [23] shows that no deterministic consensus protocol can be possible in asynchronous settings as soon as one node may crash. In contrast, Ben-Or [7] and Rabin [39] showed how to circumvent the impossibility via randomization. Those pioneering works inspire many other classic works along the line of asynchronous binary agreement (ABA) [7, 13] which consider input of each node to be just a bit. ABA protocols are known to be an important component towards building a full-fledged BFT or atomic broadcast protocol [2, 15, 22, 24, 26, 34, 35, 40]. We observe (and verified in experiments) that running a large number of ABA instances becomes the bottleneck for efficiency and we strive to minimize the use of it.

HoneyBadgerBFT [34] is the first practical asynchronous atomic broadcast protocol that comes with two major observations: (1) a weaker problem of asynchronous common subset (ACS), originally proposed by Ben-Or et al. [9] can easily be converted to an atomic broadcast without much overhead; (2) the ACS protocol constructed from *reliable broadcast* (RBC) and *asynchronous binary agreement* (ABA) with careful instantiations over-performs the previous thought of constructing it directly from a multi-valued Byzantine agreement (MVBA) [15].

A few recent practical improvements of HoneyBadgerBFT come from the nice works of BEAT [22] and Aleph [24]. In particular, the BEAT carefully examine different use cases, and make suggestions about the suitable component to choose to deploy in practice. In more detail, besides BEAT3,4 are for BFT storage only, they presented BEAT0-2 to meet different goals. The components in BEAT1 and BEAT2 are chosen in a delicate way that even though the communication complexity seems to be larger for reasonably large messages, but if the message size is small, they are actually faster, see Table 1 in Sec. 6. The Aleph is trying to improve latency obtain $\log n$ factor improvement by proposing a different assumption on the transaction buffers, however, the ABA still influence the latency such that the latency still needs $O(\log n)$. One interesting technique of Aleph is to remove the trusted dealer assumption, which also may be used in our Dumbo protocols.

As we briefly mentioned earlier, our methods and BEAT are orthogonal and compatible: their work kept the structure of the HoneyBadgerBFT intact, thus have the same round complexity, but cherry-pick the best instantitaions of the underlying components; we focus on restructuring the ACS protocol, but majority of components are the same. Their techniques from BEAT0-BEAT2 can all be applied to our protocols as well. So for experiments, we focus on comparisons with HB-BFT. Combining all their techniques with ours would be interesting future work.

2 MODELS AND PROBLEM STATEMENT

2.1 System model

We now describe our system model.

Setup. In particular, it involves a designated set of n nodes $\{P_i\}_{i\in[n]}$, we use $[n]$ to denote the integers $\{1, 2, \ldots, n\}$. We consider the identities of these nodes to be public, e.g., certified by a PKI. We denote by (PK_i, SK_i) the public/private key pair of nodes P_i. In addition to the already-established identities, a trusted third-party also runs before the protocol to set up all involved threshold cryptosystems.

Static corruptions. We assume that there are f faulty nodes ($3f + 1 \leqslant n$), and consider these faulty nodes are fully controlled by the adversary [15, 34]. Such adversary model means that before the start of the protocol, the adversary is allowed to choose f nodes to completely corrupt them, then the adversary can gets all the faulty nodes' initial internal states and also can let these nodes arbitrarily misbehave during the execution of the protocol.

Asynchronous network. We consider the underlying communication network consisting of asynchronous fully-meshed authenticated point-to-point (p2p) channels. In this model, between any two nodes, there is an established authenticated p2p channel. However, the adversary can fully control the value delivered over all channels, i.e., the adversary can arbitrarily delay, but the values send between honest nodes will eventually be delivered, which explicitly implies two facts: (i) the adversary can arbitrarily reorder values and (ii) the network will not drop any values from honest nodes.

2.2 Design goals

Atomic broadcast. Our end goal is to design an atomic broadcast protocol among n nodes under the system model above. Formally, an atomic broadcast protocol satisfies the following properties with an overwhelming probability:

- *Agreement.* If one honest node outputs a value v, then every honest node outputs v;
- *Total order.* If two honest nodes output sequences of value $\langle v_0, v_1, \ldots, v_j\rangle$ and $\langle v'_0, v'_1, \ldots, v'_{j'}\rangle$, respectively, then $v_i = v'_i$ for $i \leqslant \min(j, j')$;
- *Censorship resilience.* If a value v is input to $n - f$ honest nodes, then it is eventually output by every honest node.

We require the three properties hold with an overwhelming probability. In short, we will adopt the same model as HoneybadgerBFT [34], i.e., atomic broadcast among n nodes against f static corruptions in an asynchronous network.

Atomic broadcast protocol proceeds in consecutive epochs, after each epoch, a new batch of transactions is output and appended to the committed log (see Appendix A).

Asynchronous common subset (ACS). One nice observation from HoneyBadgerBFT [34] is an efficient and simple conversion to an atomic broadcast from a weaker variant called asynchronous common subset (ACS) together with threshold encryption. An ACS

essentially let each node output a common subset of all the node inputs. Formally, it satisfies:

- *Agreement.* If an honest node outputs a set V, then every honest node outputs V;
- *Validity.* If an honest node outputs a set V, then $|V| \geqslant n - f$ and V contains the inputs of at least $n - 2f$ honest nodes;
- *Totality.* If $n - f$ honest nodes have an input, then all honest nodes can produce an output.

Remark that there exists a simple conversion from ACS to atomic broadcast by adding threshold encryption, we refer the details in Appendix A and [34].

Complexity measures. The practicality of BFT protocols depends heavily on their computational complexity. In this paper, we consider the following three metrics:

- *Message complexity:* the expected total number of messages that honest nodes generate during the protocol;
- *Communication complexity:* the expected total bit-length of messages that honest nodes generate during the protocol;
- *Time (round) complexity:* the expected number of rounds of communication before the protocol terminates.

Besides, note that we always consider $n = 3f + 1$ throughout this paper, hence, our BFT protocol is also an optimal resilience which just considers how many nodes may be corrupted.

3 PRELIMINARIES

We introduce definitions for some underlying building blocks.

Reliable broadcast (RBC**)** is a protocol running among a set of n nodes in which there is a node called sender whose aim is to broadcast a value to all the other nodes. More formally, an RBC protocol satisfies the following properties:

- *Agreement.* If any two honest nodes output v and v' respectively, then $v = v'$;
- *Totality.* If an honest node outputs v, then all honest nodes output v;
- *Validity.* If the sender is honest and inputs v, then all honest nodes output v.

Consistent broadcast (CBC) is similar to RBC, but it does not provide *Totality*.

Asynchronous binary agreement (ABA**)** is a special asynchronous Byzantine agreement protocol among n nodes. In an ABA protocol, each node has a single-bit (0/1) input, and their goal is to reach an agreement on the decided bit [14, 16, 36]. More formally, an ABA protocol has the following guarantees:

- *Agreement.* If any honest node decides the bit b, then every honest node decide b;
- *Termination.* If all honest nodes receive input, then every honest node decides a bit;
- *Validity.* If any honest node decides b, then at least one honest node received b as input.

Remark: As many previous works [14, 16, 34, 36], the *termination* property here for ABA only requires all honest node to *decide* (in the sense of outputting a value for further applications, while not halting the protocol). It is possible that they each decide a value, but some node still continues waiting messages in the protocol [34]. As in [34], we use "output" and "decide" interchangeably on a bit in ABA. Please see Appendix B for details of concrete constructions of RBC, CBC and ABA [7] protocols.

Multi-valued Byzantine agreement (MVBA**):** The MVBA [2, 15] allows agreement on arbitrary values instead of being restricted to binary values. The protocol has a global, polynomial-time computable predicate Q known to all nodes, which is determined by the particular application. The basic idea of the protocol is that each party proposes a (different) value that contains certain validation information as input and outputs an value which satisfies the Q as the decision value. The protocol ensures that the decision value was proposed by at least one party. Each honest node only inputs a value to MVBA that satisfies Q.

More formally, an MVBA protocol satisfies the following properties except with negligible probability:

- *Termination.* If every honest node P_i inputs with an externally valid value v_i, then every honest node outputs a value;
- *External-Validity.* If an honest node outputs a value v, then $Q(v) = True$;
- *Agreement.* All honest nodes that terminate output the same value;
- *Integrity.* If all nodes are honest and if some nodes output v, then some nodes proposed v.

Threshold signature scheme: Let $0 \leqslant t \leqslant n$, A (t, n)-non interactive threshold signature scheme is a tuple of algorithms which involves n nodes and up to $t - 1$ node can be corrupted, where each node have a private function SigShare, and three public functions ShareVerify, Combine and Verify (see Appendix B for formal definitions). The signature schema satisfies the following properties:

- *Unforgeability:* No polynomial-time adversary can forge a signature that can be verified correctly (by honest parties) of any message m without querying the signature algorithm;
- *Robustness:* When a message m is provided as the input of the signature algorithm, eventually all honest parties can get a signature of m that can be correctly verified.

$(1, \kappa, \epsilon)$-Committee election (CE**):** A CE protocol is executed among n nodes (identified from 1 through n). If at least $f + 1$ honest nodes participate, the protocol terminates with honest nodes output a κ-sized committee set C such that at least one of C is honest nodes. In particular, a protocol is said to be $(1, \kappa, \epsilon)$-committee election, if it satisfies the following properties except with negligible probability in cryptographic security parameter λ:

- *Termination.* If $f + 1$ honest nodes activate the protocol CE, all messages among honest nodes arrive, then all honest nodes output C;
- *Agreement.* Any two honest nodes output the same set C;
- *Validity.* If any honest node outputs C, (i) $|C| = \kappa$, (ii) the probability of every node $P_i \in C$ is same, and (iii) C contains at least one honest node with at least probability $1 - \epsilon$;
- *Unpredictability.* Before invocation by one honest node, the probability of the adversary to predict the returned committee is at most $1/\binom{n}{\kappa}$.

[7] The original HoneyBadgerBFT [34] used an ABA protocol [36] which requires a strong common coin that cannot be realized by the threshold coin scheme [16]. The revised version of HB-BFT added a fix [33] of the ABA protocol (see also Alg. 7 without the "amendament"). In our experiments, we adopted this revised ABA.

Remark that a $(1, \kappa, \epsilon)$-CE can be constructed directly from a threshold coin-tossing (see Appendix B), which can be readily derived from threshold signatures. At least one honest node is elected in C with an overwhelming probability $1-\epsilon-\mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda)$ is a negligible function in cryptographic security parameter λ, and ϵ is $exp(-\Omega(\kappa))$ (c.f. Lemma 4.1 for details).

4 DUMBO1: A FAST ASYNCHRONOUS BFT PROTOCOL

In this section, we present our first ACS, which is called Dumbo1-ACS. Applying the same conversion from ACS to atomic broadcast [34] (adding threshold encryption), we can obtain a new atomic broadcast: Dumbo1. We will focus on the ACS protocol below.

4.1 Dumbo1-ACS

High level overview. The core of our design is to reduce the number of ABA instances needed in an ACS execution. As briefly elaborated in the Introduction, the reason that HoneyBadgerBFT (and also BEAT) needs n instances of ABA is due to the following reason: the peers need to agree what to do for each peer's input via one ABA instance. Observe that after the first RBC phase, each peer is prepared with a subset of inputs.

Instead of investing an ABA for each input, we would let a small number κ of "aggregators" to nominate which *subset* of inputs to output (based on what it has already received). In this way, each ABA instance is now used to let the nodes to determine whether they agree on the i-th nominated subset S_i. We remark again that the nomination procedure is also using RBC, however, the inputs are just indices-set S_i instead of the actual data load. Also the nominator/committee election is just one coin-tossing at best, thus the overhead is minimal compared to the saved ABA instances.

Note that κ instances of ABA protocol are still needed, otherwise one honest node may decide to follow S_i while the other honest node may decide to follow S_j, as each of them indeed receives the corresponding input values but ends up violating the agreement.

In slightly more detail, as illustrated in Figure 3 in Introduction, our Dumbo1-ACS includes two phases of RBC, denoted as *data*-RBC and *index*-RBC respectively. The data-RBC instances are executed by the nodes to broadcast their inputs. κ leaders will then be selected. The index-RBC instances are only initiated by the selected members when they have received $n-f$ values from those data-RBC instances. Each index-RBC is used to broadcast the indexes indicating which $n-f$ values that a selected member has already received. In the last phase, an honest node will input 1 to the i-th ABA instance if it has already received S_i (with size $n-f$) and all the corresponding values in the data-RBC instances.

Committee election. The election of those committees/nominators is standard practice, i.e., randomly choosing κ nodes to ensure the probability that one of them is honest with an overwhelming probability in κ. Usually, the probability that none of κ random peers to be honest is at most $(1/3)^\kappa$ (see Lemma 4.1). In practice, we could let system designer to choose $\kappa = min\{\kappa_0, f+1\}$ in a way that $(1/3)^{\kappa_0} \leq \epsilon_0$ for any small ϵ_0 he likes.

Construction of Dumbo1-ACS. Now we describe the construction of our Dumbo1-ACS. The detailed process of Dumbo1-ACS is shown in Algorithm 1. As illustrated in Figure 3 in Introduction, our Dumbo1-ACS includes two phases of RBC, the first phase is to broadcast value, the second phase is to broadcast indices.

The Dumbo1-ACS protocol composed of five logical phases, the detailed protocol proceeds as follows:

- *Value broadcast*: (line 02). All nodes P_i input their value v_i to RBC_i protocol.
- *Committee election*: (line 03-03). All nodes participate CE protocols to select committee C, and take the committee member' identities into a set CMIS.
- *Indices broadcast*: (line 06-09). When the committee members have received $n-f$ Value message from distinct RBC instances, then they will broadcast these indexes of $n-f$ Value messages through RBC.
- *ABA phase*: (line 10-17). When one honest node has already received Index message (S_j) from committee member P_j and all corresponding Value message from the RBC instances, then input 1 to the ABA_j instance; if one honest node gets an output from any ABA_j and $j \in \mathsf{CMIS}$, then input 0 to other ABA instance which no input has been provided to yet.
- *Output phase*: (line 18-25). When a node terminates in all κ instances of ABA, for any $x \in \mathsf{CMIS}$, if ABA_x outputs 1, then the node waits Index message from RBC_x and gets a set S, and further waits Value message to get v_j for all $j \in S$ from $\mathsf{RBC_j}$ and finally outputs $\{v_j\}_{j\in S}$.

Algorithm 1 The Dumbo1-ACS protocol (for party P_i) in consecutive epoch r

01: **Let** $\{\mathsf{RBC}_j\}_n$ refer to n instances of the reliable broadcast protocol, where P_j is the sender of RBC_j, and ABA_j refer to the ABA instance corresponding committee member P_j. **Initial:** Committee member identities set $\mathsf{CMIS} = \emptyset$.
02: **Input** (Value, v_i) to RBC_i; ▷ data-RBC
03: **Invoke** Committee Election protocol CE(r);
04: **wait** until Committee: $\{P_{j_1}, P_{j_2}, \cdots, P_{j_\kappa}\} \leftarrow \mathsf{CE}(r)$
05: $\mathsf{CMIS} \leftarrow \{j_1, j_2, \cdots, j_\kappa\}$;
06: **if** $P_i \in$ Committee **then**
07: **wait** until receiving $n-f$ Value messages $\{(\mathsf{Value}, v_{i_1}), (\mathsf{Value}, v_{i_2}), \cdots, (\mathsf{Value}, v_{i_{n-f}})\}$ from distinct RBC instance
08: let $S_i = \{i_1, i_2, \cdots, i_{n-f}\}$;
09: input (Index, S_i) to RBC_i; ▷ index-RBC
10: **upon** receiving Index message (Index, S_j) from committee member P_j and $|S_j| = n-f$ **do**
11: **if** no input has been provided to ABA_j **then**
12: **wait** until all $(\mathsf{Value}, v_x)_{x\in S_j}$ have received
13: input 1 to ABA_j;
14: **upon** receiving 1 from any ABA_j and $j \in \mathsf{CMIS}$ **do**
15: **for** x: $x \in \{\mathsf{CMIS} - j\}$ **do**
16: **if** no input has been provided to ABA_x **then**
17: input 0 to ABA_x;
18: **upon** all κ ABA instances have completed **do**
19: **for** x: $x \in \mathsf{CMIS}$ **do**
20: **if** ABA_x output 1 **then**
21: **wait** Index message: $(\mathsf{Index}, S_x) \leftarrow \mathsf{RBC}_x$
22: $S \leftarrow S \cup S_x$;
23: **for** $j \in S$ **do**
24: **wait** Value message: $(\mathsf{Value}, v_j) \leftarrow \mathsf{RBC}_j$
25: **Output** $\cup_{j\in S} v_j$.

4.2 Security analysis

Dumbo1 realizes an atomic broadcast via the combination of ACS and threshold encryption. To prove the security of Dumbo1, we

need to go in two steps: (1) a reduction from atomic broadcast to ACS and threshold encryption; (2) to show our new constructed Dumbo1-ACS protocol indeed satisfies the ACS properties.

The proof for Step 1 has been given in [34], for more details please refer to Appendix A. Here we focus mainly on the Step 2 and prove Dumbo1-ACS satisfies all properties of ACS.

Lemma 4.1. *(Validity of* CE.*) If* $n = 3f + 1$, $\kappa \leqslant f$ *and* CE(id) *returns a set C, then the set C containing at least one honest nodes except with* $exp(-\Omega(\kappa))$ *probability.*

Proof. Due to the pseudo-randomness, hence, the total case is $\binom{n}{\kappa}$ of random choose κ nodes, and the total case is $\binom{f}{\kappa}$ of the set C containing no honest nodes. Let p is the probability of the set C containing no honest nodes. So, we have

$$p = \frac{\binom{f}{\kappa}}{\binom{n}{\kappa}} = \frac{f!(n-\kappa)!}{(f-\kappa)!n!} \leqslant (\frac{1}{3})^{\kappa} = exp(-\Omega(\kappa)). \qquad \square$$

Remark: If f is small, we can simply set $\kappa = f + 1$. Algorithm 4 satisfies the *Termination, Agreement and Unpredictability* properties of CE follows from the properties of threshold coin-tossing.

Theorem 4.2. *With except* $exp(-\Omega(\kappa))$ *probability, the Dumbo1-ACS protocol satisfies Agreement, Validity, and Totality of ACS, assuming the underlying* RBC, CE *and* ABA *protocols are secure.*

Proof. *Agreement:* To prove that Dumbo1-ACS satisfies the agreement property, we show that when an honest node outputs V, then every honest node outputs V. Assume that an honest node P has a output $V = \{v_j\}_{j \in S}$.

The indices S must be contained in some index sets. W.l.o.g, we assume S is included only one index set S_k, (Index, S_k) was received in some RBC (denoted as RBC_k), then the node must have received 1 in the corresponding ABA instance (denoted as ABA_k). Due to the *agreement* property of ABA, all honest nodes will also receive 1 in ABA_k. Hence, all honest nodes will wait an Index message output from RBC_k, due to *totality* and *agreement* of RBC, so all other honest nodes will receive same Index message (Index, S_k).

On the other hand, due to the *validity* of ABA, at least one honest node P' inputs 1 to ABA_k. It implies that this honest node must have received Index message (Index, S_k) and these Value messages $\{\text{Value}, v_j\}_{j \in S_k}$ corresponding to the index set S_k. The *totality* and *agreement* of RBC now can ensure that all other honest nodes including P will receive $\{\text{Value}, v_j\}$ for any $j \in S_k$.

Hence, every honest node outputs $\{v_j\}_{j \in S_k} = \{v_j\}_{j \in S} = V$.

Validity: To prove that Dumbo1-ACS satisfies the validity property, we show that $|V| \geqslant n - f$ and V contains the input of at least $n - 2f$ honest nodes when an honest node outputs a set V.

If an honest node P_i outputs a set $V = \{v_j\}_{j \in S}$, W.l.o.g, we assume S is included only one index set S_k, (Index, S_k) was received in index-RBC_k. According to the Algorithm 1, we can know ABA_k return 1, due to the *validity* of ABA, at least one honest node (say P') inputs 1 to ABA_k. It implies that P' must have received Index message (Index, S_k) and all Value message $\{\text{Value}, v_j\}_{j \in S_k}$ corresponding the index set S_k, where $|S_k| = n - f$.

The *totality* and *agreement* of RBC now can ensure that all honest nodes including P_i will receive $\{\text{Value}, v_j\}_{j \in S_k}$, where $|S_k| = n - f$. So, $V = \{v_j\}_{j \in S_k}$. Hence, we have $|V| \geqslant n - f$. Notice that there are at most f byzantine nodes, there must be at least $n - 2f$ inputs from honest nodes in set V.

Totality: To prove that Dumbo1-ACS satisfies the totality property, we show that all honest nodes produce an output if $n - f$ honest nodes have an input.

Since $n - f$ honest nodes have an input, according to the *validity* of RBC, hence, every committee member can receive $n - f$ Value messages from distinct RBC instance. So, every committee member can activate the second phase RBC instance. Besides, according to the CE protocols, there exists at least an honest node (say P_i) belongs to the committee.

Next, we will prove that at least one ABA instance returns 1.

Firstly, suppose all ABA instances output 0, in this case, line 14-17 will never execute, that is to say, 0 will never input to any ABA instance by honest nodes. However, according to the *validity* of ABA, at least one honest node inputs 0 to ABA, which induces contradiction.

Secondly, since the committee member P_i can activate the second phase RBC_i instance, if all honest nodes have not received 1 from any ABA all the times, in this case, all honest nodes can receive valid Index message from P_i (the *validity* of RBC) and corresponding Value message from RBC instances (the *totality* of RBC), next, all honest nodes input 1 to ABA_i. Again according to the *validity* of ABA, the ABA_i will return 1 to all.

Hence, there exists at least one ABA (say ABA_k) instance returning 1. Due to *validity* of ABA, at least one honest node (say P') inputs 1 to ABA_k. It implies that such an honest node must have received Index message (Index, S_k) and all Value message $\{\text{Value}, v_j\}_{j \in S_k}$ corresponding the index set S_k. The *totality* and *agreement* of RBC now can ensure that all honest nodes will receive $\{\text{Value}, v_j\}_{j \in S_k}$. Hence, all honest nodes can produce an output $\{v_j\}_{j \in S_k}$. $\square$

5 DUMBO2: A FASTER ASYNCHRONOUS BFT PROTOCOL

In this section, we present a further improved ACS protocol Dumbo2-ACS, which reduces the number of ABA instances to constant, thus guarantees termination within a constant running time. We show a new construction of ACS using RBC and MVBA. More interestingly, our new method demonstrates an innovative use of MVBA can actually lead to more efficient ACS, which was considered less promising than using RBC and ABA in [34].

5.1 Dumbo2-ACS

High level overview. As discussed above, Dumbo1 improves HB-BFT in the sense that reduces the number of ABA instances to κ, but still they all need to be run. In order to minimize the usage of ABA, we need to (1) prepare each peer with a vector of inputs from enough peer nodes; (2) find a way to identify and output one of them. The former is easy that an RBC phase already achieves it. The latter inspire us to re-examine the possibility of MVBA which outputs only one input (not necessarily from an honest node, but satisfies some condition). As explained in Introduction (and also Table 1 in Sec. 6), current MVBA constructions were considered impractical for ACS due to its high communication complexity. However, if we examine all the terms in the communication complexity, the dominating term changes when message size becomes small, MVBA

could even over-perform! More importantly, MVBA [15] only needs to run three consecutive ABA instances in expectation.

Considering the conventional wisdom of *hybrid encryption*, the heavier public key encryption is only used to hide a short session key, while the actual (potentially large) message will be encrypted using symmetric key encryption using the session key. In analog to that, we similarly use indices as input to invoke MVBA. Now we have one more challenge that MVBA may output one such index-set from a dishonest node. To resolve this, we propose *provable* RBC that further outputs a succinct proof s.t., whoever produces such a proof, it guarantees that all honest nodes will receive the input value.

Provable reliable broadcast (PRBC). A natural way to obtain such a (succinct) proof is to get acknowledgement from enough nodes, which can be realized via threshold signing on the RBC identifier. The PRBC protocol with an identifier id, and a verify algorithm Verify is denoted PRBC_{id}. Formally, an PRBC_{id} protocol satisfies the following properties except a negligible probability:

- *Agreement.* If one honest node outputs v, another honest node outputs v', then $v = v'$;
- *Totality.* If any node outputs a pair (id, σ) and $\text{Verify}(id, \sigma) = 1$, then all honest nodes output a value v and (id, σ);
- *Validity.* If the sender is honest and inputs v, then all honest nodes output v and valid string (id, σ);
- *Succinctness.* The length (size) of valid string σ is independent with the length of value v.

A PRBC can be constructed from RBC and threshold signature, it is shown in Algorithm 2. The PRBC protocol can be decomposed in three logical phases, the details are as follows:

- Value broadcast phase: (line 02-04). If the node is sender, then the node P_s inputs the value v_s to RBC protocol.
- Output value phase: (line 05-07). If honest nodes receive a value from sender, then the nodes send a threshold share signature of id to all.
- Output signature phase: (line 08-13). If nodes received $f + 1$ valid threshold share signature of id, they can combine these share signature into a threshold signature σ of id, then output the σ.

Algorithm 2 The PRBC_{id} protocol with epoch r (for party P_i, where the sender is P_s and $id = \langle r, s \rangle$)

1: **Let** RBC_{id} refer to the instance of the reliable broadcast protocol, where P_s is the sender of RBC_{id}; $\{DS_s\} = \emptyset$.
2: **if** $P_i = P_s$ **then**
3: **upon** receiving input value v_s **do**
4: input $\{\text{Value}, v_s\}$ to RBC_{id};
5: **upon** receiving Value message $\{\text{Value}, v\}$ from RBC_{id} **do**
6: $\sigma_{is} \leftarrow \text{SigShare}_{f+1}(sk_i, id)$;
7: multicast $(\text{Done}, id, \sigma_{is})$;
8: **upon** receiving a Done message $(\text{Done}, id, \sigma_{js})$ from node P_j for the first time **do**
9: **if** $\text{ShareVerify}_{f+1}(id, (j, \sigma_{js})) = 1$ **then**
10: $DS_s \leftarrow DS_s \cup \{j, \sigma_{js}\}$;
11: **upon** $|DS_s| = f + 1$ **do**
12: $\sigma_s \leftarrow \text{Combine}_{f+1}(id, DS_s)$;
13: **return** $(\text{Finish}, id, \sigma_s)$.

Construction of Dumbo2-ACS. Now we give the construction of our Dumbo2-ACS protocol, the details of which are shown in Algorithm 3. We denoted MVBA_r as MVBA protocol with identification r. As illustrated in Figure 4 in Introduction, the Dumbo2-ACS includes two part: PRBC and MVBA.

We will use $W = \{(s_1, \sigma_1), (s_2, \sigma_2), \cdots, (s_n, \sigma_n)\}$ as the input to MVBA_r for each node. In particular, s_i is put in W if the corresponding σ_i on s_i is received. The predicate Q of the MVBA_r will output 1 if at least $n - f$ distinct i satisfy $s_i \neq \perp$ in W, and for each (s_i, σ_i) of W, it is a valid output of PRBC, i.e., $\text{Verify}_{f+1}(\langle r, s_i \rangle, \sigma_i) = 1$ if $s_i \neq \perp$. The Dumbo2-ACS protocol can be decomposed in three logical phases, the detailed protocol proceeds as follows:

- Value broadcast phase: (line 03-04). All nodes P_i input their value v_i to PRBC protocol, and wait for $n - f$ distinct Finish messages.
- MVBA phase: (line 08-10). Upon receiving $n - f$ distinct Finish messages, then invoke the MVBA protocol and wait to get an output $\overline{W}$ from MVBA.
- Output phase: (line 11-13). All honest nodes wait Value message from PRBC according to the $\overline{W}$.

Algorithm 3 The Dumbo2-ACS protocol (for party P_i) in consecutive epoch r

1: **Let** $\{\text{PRBC}_{\langle r,j \rangle}\}_n$ refer to n instance of provable reliable broadcast protocol, where P_j is the sender of $\text{PRBC}_{\langle r,j \rangle}$, and the Q be the following predicate: $Q_r[\{(s_1, \sigma_1), (s_2, \sigma_2), \cdots, (s_n, \sigma_n)\}] \equiv$ (at least $n - f$ distinct i satisfy $s_i \neq \perp$ and $\text{Verify}_{f+1}(\langle r, s_i \rangle, \sigma_i) = 1$).
2: **Initial:** $W = \{(s_1, \sigma_1), (s_2, \sigma_2), \cdots, (s_n, \sigma_n)\}$, where $(s_j, \sigma_j) \leftarrow (\perp, \perp)$ for all $1 \leqslant j \leqslant n$; $FS = 0$.
3: **upon** receiving input value v_i **do**
4: input $\{\text{Value}, v_i\}$ to $\text{PRBC}_{\langle r,i \rangle}$;
5: **upon** receiving a Finish message $(\text{Finish}, \langle r, j \rangle, \sigma_j)$ **do**
6: $(s_j, \sigma_j) \leftarrow (j, \sigma_j)$;
7: $FS = FS + 1$;
8: **upon** $FS = n - f$ **do**
9: propose W for the MVBA_r;
10: **wait** the MVBA_r to return $\overline{W} = \{(\bar{s}_1, \bar{\sigma}_1), (\bar{s}_2, \bar{\sigma}_2)), \cdots, (\bar{s}_n, \bar{\sigma}_n)\}$
11: Let $S \subset [n]$ be the set of $\bar{s}_j \neq \perp$ for $1 \leqslant j \leqslant n$.
12: Wait until receive v_j from $\text{PRBC}_{\langle r,j \rangle}$ for all $j \in S$.
13: Finally output $\cup_{j \in S} v_j$.

5.2 Security Analysis

Intuition: Similarly with the Dumbo1-ACS, when an honest node outputs a subset of values, it implies that the node has received the corresponding index subsets. Due to the termination and agreement of the MVBA, all honest nodes will receive the same index-set W.

Besides, the external-validity of MVBA ensures that the index-set W satisfies the predicate Q, which means that the input message corresponding to each index will be received by all honest nodes in the PRBC, and the size of subsets is at least $n - f$.

Like in Dumbo1, we also focus on the proof of Dumbo2-ACS. Note that the property of *succinctness* follows from proper choice of threshold signature scheme whose signature size is λ.

Lemma 5.1. *The Algorithm 2 satisfies the Agreement, Validity and Totality properties of* PRBC*, assuming the underlying* RBC *and threshold signature scheme are secure.*

Proof. *Agreement:* If one honest node outputs v and another honest node outputs v', according to the *agreement* property of RBC, we have $v = v'$.

Totality: If any node outputs string (id, σ) and $\text{Verify}_{f+1}(id, \sigma) = 1$, it implies that at least one honest has received a value v from the sender P_s. If not, at most f share signatures on id will be generated; hence, it's impossible for any malicious node to outputs a valid threshold signature. Otherwise, it will violate the *unforgeability* property of threshold signature scheme. Due to *totality* property of RBC, all honest nodes eventually output v.

Validity: If the sender is honest and inputs v, from the *validity* property of RBC, all honest nodes output v. Besides, according to the Algorithm 2, after receiving a value from the sender, each honest node will multicast a share signature of id to all, so each honest node can receive at least $f+1$ valid share signatures. Hence, all honest nodes can output a valid string (id, σ). □

Theorem 5.2. *With except negligible probability, the Dumbo2-ACS protocol satisfies the Agreement, Validity, and Totality properties of ACS, assuming the underlying* PRBC *and* MVBA *are secure.*

Proof. *Agreement:* To prove that Dumbo2-ACS satisfies the agreement property, we need to prove that when an honest node outputs V, then every honest node outputs V.

Assume an honest node P_i outputs $V = \{v_j\}_{j \in S}$. It implies that for any $j \in S$, $\bar{s}_j \neq \perp$ in $\overline{W}$, following *external-validity* of MVBA, we know there is a valid proof (threshold signature) $\bar{\sigma}_j$ for $\bar{s}_j$. Hence, according to the *totality* of PRBC, all honest nodes including P_i output v_j. Besides, the *agreement* of MVBA ensures all honest nodes have the same S. So every honest node also outputs $V = \{v_j\}_{j \in S}$.

Validity: To prove that Dumbo2-ACS satisfies the validity property, we show that $|V| \geqslant n - f$ and V contains at least $n - 2f$ inputs from honest nodes when an honest node outputs a set V.

If an honest node P_i outputs a set $V = \{v_j\}_{j \in S}$. Due to the predicate Q of MVBA, (1) for any $\bar{s}_i \in \overline{W}$, there is a corresponding valid threshold signature $\bar{\sigma}_i$ if $\bar{s}_i \neq \perp$, (2) at least have $n - f$ distinct $\bar{s}_i \neq \perp$. Hence, according to the *totality* of PRBC, P_i outputs the data set V including $n - f$ values.

Note that there are at most f byzantine nodes, hence, there must be at least $n - 2f$ values from honest nodes' input in set V.

Totality: To prove that Dumbo2-ACS satisfies the totality property, we show that all honest nodes produce an output if $n - f$ honest nodes have an input.

According to the *validity* of PRBC: if a sender i is honest, then all honest nodes receive v_i and (i, σ_i). Now that $n - f$ honest nodes have an input, thus every honest node can receive at least $n - f$ distinct valid pairs (id, σ_{id}). Hence, every honest node can receive at least $n - f$ distinct Finish messages and define an externally valid value W_i as input to MVBA. Following *agreement* and *termination* of MVBA, all honest nodes can get the same output $\overline{W}$ from MVBA.

Besides, the value $\overline{W}$ satisfies the predicate Q due to the *external-validity* property of MVBA. Hence, for any $\bar{s}_i \neq \perp$, there is a corresponding valid threshold signature $\bar{\sigma}_i$. Let S be the set of $\bar{s}_i \neq \perp$ for all $1 \leqslant i \leqslant n$. The *totality* of PRBC now can ensure that all honest nodes will receive $\{v_i\}_{i \in S}$. □

6 EFFICIENCY ANALYSIS

Throughout the paper, we consider $|m|$ denotes the message size, λ is the security parameter for the cryptographic primitives and also denotes the size of (threshold) signature.

Table 1: Detailed performance metrics of ACS.

	Complexity‡		
Protocol	Time	Communication	Message
HB-BFT/BEAT0	$O(\log n)$	$O(n^2\|m\| + \lambda n^3 \log n)$	$O(n^3)$
BEAT1/BEAT2	$O(\log n)$	$O(n^3\|m\| + \lambda n^3)$	$O(n^3)$
Dumbo1	$O(\log \kappa)$	$O(n^2\|m\| + \lambda n^3 \log n)$	$O(n^3)$
Dumbo2	$O(1)$	$O(n^2\|m\| + \lambda n^3 \log n)$	$O(n^3)$

‡ Time means expected running time (or communication rounds).

One may notice that the communication complexities here look different with that in [34]: here the communication and message complexity both refer to the *total* complexity for the whole ACS with all terms, while in [34] they calculated complexity per transaction, and "ignored" the terms they considered small for *large-size* input.

Efficiency of Dumbo1. Firstly, let's go through the process of the Dumbo1-ACS. Following Alg. 1, the messages exchange in four places. First, all parties participate n concurrent RBC instances to broadcast input values to all; second, all parties to participate the committee election; third, all parties to participate κ concurrent RBC instances to broadcast indexes message to all; last, all parties to participate κ concurrent ABA instances to agree on whose indices to adopt. First we present the following observation from [8]. Suppose $X_1, X_2, \ldots, X_n$ be independent random variables such that for every $1 \leqslant i \leqslant n$, $Pr[X_i > j] = q^j$ $(0 < q < 1)$. If $Y = max\{X_i\}$, then $EXP[Y] = O(\log n)$.

Hence, the expected *time complexity* of Dumbo1 is $O(\log \kappa)$ due to the κ ABA instances (and all other concurrent RBC instances have constant rounds in total). *Message complexity* is still $O(n^3)$ as the added components κ (κ is normally substantially smaller than n) more RBC instances cost no more than $O(\kappa \cdot n^2)$, while there are also some reductions due to smaller # of ABA instances. Regarding *communication complexity*, the data-RBC instances generate $O(n^2|m| + \lambda n^3 \log n)$ bits communication, while the added index-RBC instances generate only $O(n^2\lambda)$ bits communication, (ignoring the reduced part due to ABA reduction). Hence, the communication complexity of Dumbo1-ACS is still $O(n^2|m| + \lambda n^3 \log n)$ as before.

Efficiency of Dumbo2. Similarly, here we also go through the process of Dumbo2-ACS. From the Algorithm 3, the message exchange appears in two places. First, all parties participate n concurrent PRBC instances, the second phase an MVBA instance.

The expected *time complexity* of Dumbo2 is $O(1)$ due to all concurrent PRBC instances have constant rounds in total and the running time of MVBA is also constant. *Message complexity* still keeps same with the HoenyBadgerBFT and is $O(n^3)$, due to it needs n PRBC instances (incur $O(n^3)$ message) and the MVBA message complexity is $O(n^2)$. Since it needs n PRBC instances, hence, the *communication complexity* of the Dumbo2 was dominated by concurrent n PRBC instances and up to $O(n^2|m| + \lambda n^3 \log n)$. In fact, the MVBA phase only generates $O(\lambda n^3)$ bit communication.

	Basic Latency (s)					Throughput (tx/s)				
System Scale	HB-BFT [34]	Dumbo1		Dumbo2		HB-BFT [34]	Dumbo1		Dumbo2	
n=32	70	19	↓ 73%	7.5	↓ 89%	8430	11313	↑ 34%	15121	↑ 79%
n=64	240	49	↓ 80%	14	↓ 94%	4453	12111	↑ 172%	18692	↑ 320%
n=100	491	90	↓ 78%	24	↓ 95%	1934	8814	↑ 356%	17767	↑ 819%

Table 2: Improvements of latency and throughput

7 EXPERIMENTAL EVALUATIONS

We implement Dumbo1, Dumbo2 (the full fledged atomic broadcast) and HoneyBadgerBFT, and deploy them on Amazon AWS. Note that we used same parameters and environment as HB-BFT. We carry out a serial of experiments in various settings with different system scales and input sizes. The results demonstrate that we have significant improvements on both latency and throughput over HBBFT, especially when n gets moderately large. Some example comparisons are given in Table 2 collected from random nodes.

Implementation details. Our prototypes of Dumbo1 and Dumbo2 are implemented in Python, part of which were developed from the implementation of HoneyBadgerBFT provided by [33, 34]. Each node runs on a separate EC2 instance. At the beginning of the program, nodes establish communication channels with each other through unauthenticated TCP sockets. All nodes behave honestly by default. In the implementation of Dumbo1 and Dumbo2, the ABA instantiation was the "corrected" version, see [33] and Alg. 7 below in the appendix for details.

We implement Boldyreva's pairing-based threshold signature scheme [12] on MNT224 curve for threshold signature (also for random-number generation, coin-tossing and committee-election). For threshold encryption, we adopt the threshold encryption scheme from Baek and Zheng [6] using SS512 symmetric bilinear group. These threshold cryptography schemes were implemented with Charm [3] Python wrappers for PBC library [30]. To implement Reed-Solomon codes, we use the zfec library [42].

Evaluation. We deploy the protocols on Amazon EC2 services, run them on 100 Amazon EC2 t2.medium instances uniformly distributed from 10 different regions (Tokyo, Singapore, Mumbai, Stockholm, Paris, Frankfurt, St. Paulo, California, Virginia and Central Canada) across the globe, each with two virtual CPUs and 4GB memory. We carry out several groups of tests in different system scale, varying the batch size B from 4 to 2×10^6 transactions. We assume the size of each transaction are 250 bytes [8] and use the parameter $n = 4f$. Besides, we set the error parameter $\epsilon = 10^{-8}$ to determine the size of committee, which is sufficiently small in practice to ensure error-free operation within ten years even if suppose the time is 1 second for each epoch. Note that we always try to make instances distributed geographically heterogeneous to simulate the practical WAN environment. E.g., When 8 nodes are tested, they will be located in 8 different areas. Significant delay and fluctuation therefore commonly exist in the communication channels. It is natural that if more instances are located in the same area, the efficiency of the protocols would be higher.

Latency. Similar as HB-BFT, latency is defined as the average time interval between the time the first node starts the protocol and when the $(n - f)$-th node gets the result. Latency is related to protocol, as well as the batch size of the input. We also consider basic latency by letting each nodes propose only one transaction.

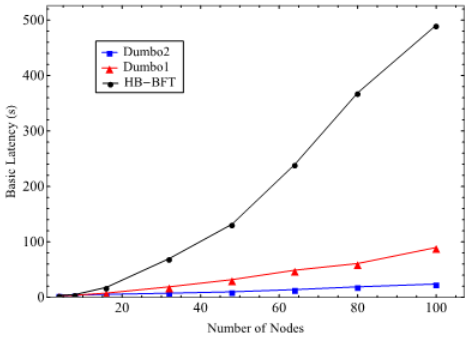


Figure 6: Basic latency of Dumbo1/2 and HoneyBadgerBFT

In Figure 6, we show the basic latency of the protocols with different system sizes n. It increases when n increases. When n is very small, the basic latency of the three protocols is almost the same. However, the latency of HB-BFT increases much faster than Dumbo1 and Dumbo2 when n gets larger. For example, when $n = 100$, the basic latency of HB-BFT has been up to 500 seconds while that of Dumbo1 is only about 80 seconds and Dumbo2 is only about 20 seconds. The reason lies in that when n gets larger, the latency improvement from reducing number of ABA instances becomes more and more significant.

Throughput. Throughput is defined as the number of transactions committed per second. We use different batch sizes to test, and show the relationship between throughput and batch size in Fig. 7.

As shown in Figure 7, the throughput of protocols get larger with the increase of batch size if the bandwidth and computing resources are sufficient. Note that the throughput reaches its peak at certain point and then may decrease if the batch size continues to grow, due to the limitation of the bandwidth or computing resources, see the result when $n = 8$. More importantly, when n gets larger, the advantage of Dumbo1/2 become more and more significant. For example, when batch size is $2{\times}10^6$ and $n = 100$, Dumbo2 running for one minute and achieve throughput more than 17000 transactions/s, which is 9 times as much as that in HoneyBadgerBFT.

Trade-off of latency and throughput. Figure 8 shows the relationship between latency and throughput in different settings ($n = 8/32/64$). For all protocols, latency grows with the increase of throughput, but the growth rate is obviously accelerating. Combined with our analysis of Fig. 7, this reflects that bandwidth (and other resources) are gradually consumed with the increase of system load. Another observation is when n is properly large, Dumbo1/2

[8] sufficient to contain an ECDSA signature, two PKs, and a typical Bitcoin transaction

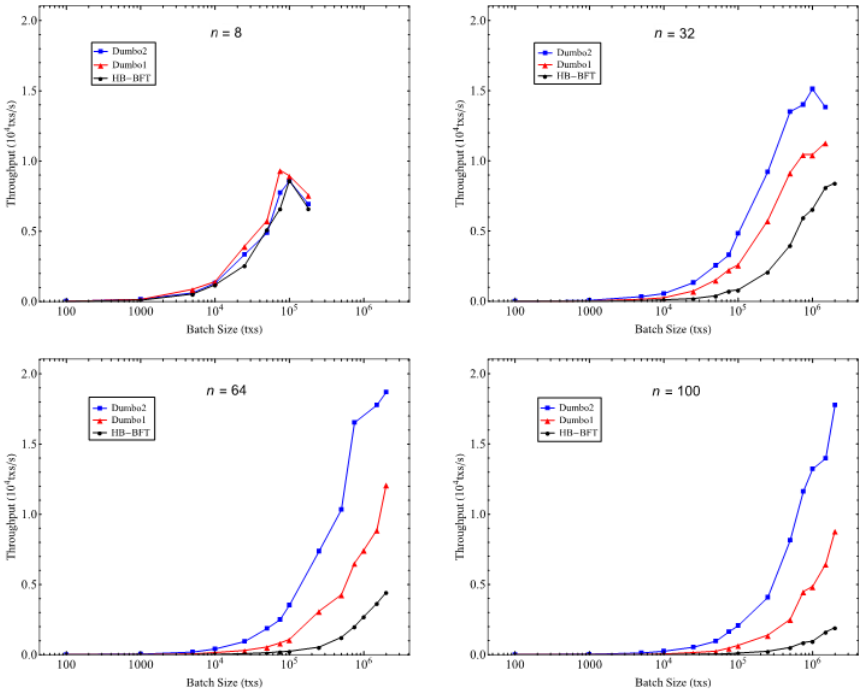


Figure 7: Throughput of Dumbo1/2 and HoneyBadgerBFT

can provide much higher throughput at the same cost of latency, which means Dumbo1/2 have better scalability so that are more applicable to larger systems.

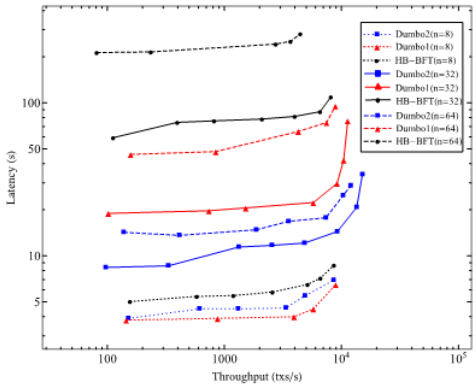


Figure 8: Throughput vs. Latency

8 CONCLUSIONS

We propose two efficient asynchronous BFT protocols named Dumbo1 and Dumbo2, each of which has an asymptotically and practically better construction of ACS. The experiments on 100 AWS EC2 instances dispersed in 4 continents of the world show that our schemes outperform HoneyBadgerBFT, known as the first practical asynchronous BFT, by several times. For example Dumbo2 can even achieve throughput of tens of thousands in a system with hundreds of nodes, meanwhile the delay is within one minute.

Our core technical contributions include two methods reducing the number of randomized ABA instances. We remark that Dumbo2 deviates the design methodology of ACS as in [34] and turn back to MVBA which was not considered optimal in building ACS. Our construction of Dumbo2 draws an analogy to the widely used conventional wisdom of hybrid encryption.

We remark that we haven't done any optimization, all improvements are demonstrated in a basic instantiation. There are multiple ways to further improve our protocols, e.g., applying the techniques from BEAT to choose the best instantitions. More importantly, we may further reduce the (message or communication) complexity and push the asynchronous atomic broadcast towards optimal. We leave them as interesting open problems.

ACKNOWLEDGMENTS

We thank anonymous reviewers for pointing out the problem in the ABA of [34] and in our previous description, and various other valuable comments. We also thank Yuan Lu for fruitful discussions. The authors are all supported in part by JD Digits via the JDD-NJIT-ISCAS Joint Blockchain Lab. Qiang Tang is also supported in part by a Google Faculty Award; Zhenfeng Zhang is also supported in part by the National Key R&D Program of China (No. 2017YFB0802500); and Jing Xu is also supported by the National Natural Science Foundation of China under Grant 61572485.

REFERENCES

[1] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. 2019. Synchronous Byzantine Agreement with Expected $O(1)$ Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience. In *International Conference on Financial Cryptography and Data Security*. Springer, 320–334.

[2] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. 2019. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 337–346.

[3] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. 2013. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering* 3, 2 (2013), 111–128.

[4] Yair Amir, Brian Coan, Jonathan Kirsch, and John Lane. 2010. Prime: Byzantine replication under attack. *IEEE Transactions on Dependable and Secure Computing* 8, 4 (2010), 564–577.

[5] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. 2013. Rbft: Redundant byzantine fault tolerance. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 297–306.

[6] Joonsang Baek and Yuliang Zheng. 2003. Simple and efficient threshold cryptosystem from the gap diffie-hellman group. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, Vol. 3. IEEE, 1491–1495.

[7] Michael Ben-Or. 1983. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*. ACM, 27–30.

[8] Michael Ben-Or and Ran El-Yaniv. 2003. Resilient-optimal interactive consistency in constant time. *Distributed Computing* 16, 4 (2003), 249–262.

[9] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. 1994. Asynchronous secure computations with optimal resilience. In *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*. ACM, 183–192.

[10] Alysson Bessani, João Sousa, and Eduardo EP Alchieri. 2014. State machine replication for the masses with BFT-SMaRt. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 355–362.

[11] Erica Blum, Jonathan Katz, and Julian Loss. 2019. Synchronous Consensus with Optimal Asynchronous Fallback Guarantees. In *Theory of Cryptography Conference*. Springer, 131–150.

[12] Alexandra Boldyreva. 2003. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *International Workshop on Public Key Cryptography*. Springer, 31–46.

[13] Gabriel Bracha. 1984. An asynchronous [(n-1)/3]-resilient consensus protocol. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*. ACM, 154–162.

[14] Gabriel Bracha. 1987. Asynchronous Byzantine agreement protocols. *Information and Computation* 75, 2 (1987), 130–143.

[15] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference*. Springer, 524–541.

[16] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. *Journal of Cryptology* 18, 3 (2005), 219–246.

[17] Christian Cachin and Jonathan A Poritz. 2002. Secure intrusion-tolerant replication on the Internet. In *Proceedings International Conference on Dependable Systems and Networks*. IEEE, 167–176.

[18] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

[19] Allen Clement, Edmund L Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti. 2009. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults.. In *NSDI*, Vol. 9. 153–168.

[20] Flaviu Cristian, Houtan Aghili, Raymond Strong, and Danny Dolev. 1986. *Atomic broadcast: From simple message diffusion to Byzantine agreement*. International Business Machines Incorporated, Thomas J. Watson Research Center.

[21] Danny Dolev, Michael J Fischer, Rob Fowler, Nancy A Lynch, and H Raymond Strong. 1982. An efficient algorithm for Byzantine agreement without authentication. *Information and Control* 52, 3 (1982), 257–274.

[22] Sisi Duan, Michael K Reiter, and Haibin Zhang. 2018. BEAT: Asynchronous BFT made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2028–2041.

[23] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1982. *Impossibility of distributed consensus with one faulty process*. Technical Report. Massachusetts Inst of Tech Cambridge lab for Computer Science.

[24] Adam Gągol, Damian Leśniak, Damian Straszak, and Michał Świętek. 2019. Aleph: Efficient Atomic Broadcast in Asynchronous Networks with Byzantine Nodes. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 214–228.

[25] Juan A Garay and Aggelos Kiayias. 2018. SoK: A Consensus Taxonomy in the Blockchain Era. *IACR Cryptology ePrint Archive* 2018 (2018), 754.

[26] Klaus Kursawe and Victor Shoup. 2005. Optimistic asynchronous atomic broadcast. In *International Colloquium on Automata, Languages, and Programming*. Springer, 204–215.

[27] Leslie Lamport. 1998. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)* 16, 2 (1998), 133–169.

[28] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.

[29] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. 2020. Dumbo-MVBA: Optimal Multi-Valued Validated Asynchronous Byzantine Agreement, Revisited. In *PODC '20: Proceedings of the 39th Symposium on Principles of Distributed Computing*. ACM, 129–138.

[30] Ben Lynn. 2007. *On the implementation of pairing-based cryptosystems*. Ph.D. Dissertation. Stanford University Stanford, California.

[31] Ethan MacBrough. 2018. Cobalt: BFT governance in open networks. *arXiv preprint arXiv:1802.07240* (2018).

[32] Dahlia Malkhi, Kartik Nayak, and Ling Ren. 2019. Flexible Byzantine Fault Tolerance. *arXiv preprint arXiv:1904.10067* (2019).

[33] Andrew Miller. 2018. Bug in ABA protocol's use of Common Coin 59. https://github.com/amiller/HoneyBadgerBFT/issues/59. Online Forum.

[34] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 31–42.

[35] Henrique Moniz, Nuno Ferreria Neves, Miguel Correia, and Paulo Verissimo. 2008. RITAS: Services for randomized intrusion tolerance. *IEEE transactions on dependable and secure computing* 8, 1 (2008), 122–136.

[36] Achour Mostefaoui, Hamouma Moumen, and Michel Raynal. 2014. Signature-free asynchronous byzantine consensus with $t < n/3$ and $O(n^2)$ messages. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*. ACM, 2–9.

[37] Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. 2015. Signature-free asynchronous binary Byzantine consensus with t< n/3, O (n2) messages, and O (1) expected time. *Journal of the ACM (JACM)* 62, 4 (2015), 31.

[38] Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX} {ATC} 14)*. 305–319.

[39] Michael O Rabin. 1983. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*. IEEE, 403–409.

[40] HariGovind V Ramasamy and Christian Cachin. 2005. Parsimonious asynchronous byzantine-fault-tolerant atomic broadcast. In *International Conference On Principles Of Distributed Systems*. Springer, 88–102.

[41] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. 2009. Spin one's wheels? Byzantine fault tolerance with a spinning primary. In *2009 28th IEEE International Symposium on Reliable Distributed Systems*. IEEE, 135–144.

[42] Zooko Wilcox-O'Hearn. 2008. Zfec 1.4. 0. *Open source code distribution: http://pypi. python. org/pypi/zfec* (2008).

A FROM ACS TO ATOMIC BROADCAST

In HoneyBadgerBFT, nodes receive "transactions" as input value and store them in their buffers. The protocol proceeds in *epochs*. At the start of each epochs, nodes choose a certain number of txs randomly from their buffer as their inputs to the atomic broadcast, and at the end of each epochs, a final set of txs for this epoch will be chosen. The final agreements are the union of all received txs decided by n ABA instance. So to improve efficiency, in HB-BFT, nodes randomly choose txs from their buffers instead of sequentially to avoid duplication. A naive attempt of atomic broadcast is sequentially invoking multiple ACS instances. the Censorship Resilience property is not satisfied. That is because the adversary knowing the inputs of all RBC instances can prevent some *tx* from being output controlling the network.

To avoid this, HoneyBadgerBFT adopted threshold encryption: all inputs are first encrypted before provided to the ACS; the agreement will be decrypted only after it has been output by the ACS. By this approach, the adversary has nothing to target. For the detailed description building an atomic broadcast from ACS is as follows:

Given an ACS and the already-setup $(f + 1, n)$ threshold encryption, atomic broadcast can be trivially instantiated [34]. In

specific, let each node P_i keep a buffer of values called buf_i. Also let TPE.Enc, TPKE.DecShare and TPKE.Dec represent the relevant algorithms of a threshold encryption scheme, and let PK and SK_i to denote the public key and the node P_i's private key of the above threshold encryption scheme (see [34] for the concrete definitions of the threshold encryption scheme). The main part of the algorithm at each node P_i will proceed as follows (with a consecutively increasing parameter r to denote the epoch number):

(1) Random selection and encryption: let *proposed* be a random selection of $\lfloor B/n \rfloor$ values from the first B elements of buf, then encrypt $x :=$ TPKE.Enc $(PK, proposed)$.
(2) Agreement on ciphertexts: pass x as input to ACS; then receive $\{v_j\}_{j\in S}$ from ACS, where S is a subset of all nodes.
(3) Decryption: for each $j \in S$: let $e_j :=$ TPKE.DecShare(SK_i, v_j), then multicast DEC(r, j, i, e_j), then wait to receive at least $f+1$ value of the form DEC$(r, j, k, e_{j,k})$, decode $y_j :=$ TPKE.Dec $(PK, \{(k, e_{j,k})\}_{f+1})$.
(4) Output the block: let $\text{block}_r := sort(\bigcup_{j\in S}\{y_i\})$, where *sort* represents a pre-specified rules to order values, then buf := buf $-$ block_r.

Theorem A.1. *(Adapted from [34]) The above protocol is atomic broadcast (i.e., it satisfies totality, agreement and censorship resilience except for negligible probability), conditioned on the underlying ACS and threshold encryption satisfy their definitions respectively.*

B INSTANTIATIONS OF BUILDING BLOCKS

Threshold signature scheme: Let $0 \leqslant t \leqslant n$, a (t, n)-non interactive threshold signature scheme is a tuple of algorithms which involves n nodes and up to $t-1$ node can be corrupted. The threshold signature scheme has the following algorithms:

- *Key generation algorithm:* $\text{SigSetup}(1^\lambda, n, t) \rightarrow \{mpk, PK, SK\}$. Give a security parameter λ and generates a special public key mpk, a vector of public keys $PK := (pk_1, \cdots, pk_n)$, and a vector of secret keys $SK := (sk_1, \cdots, sk_n)$;
- *Signing algorithm:* $\text{SigShare}_t(sk_i, m) \rightarrow \sigma_i$. On input a message m and a secret key share sk_i, this deterministic algorithm outputs a signature share σ_i;
- *Share verification algorithm:* $\text{ShareVerify}_t(m, (i, \sigma_i)) \rightarrow 0/1$. Given a message m, a signature share σ_i and an index i, this deterministic algorithm outputs 1 or 0 depending on whether σ_i is a valid signature share generated by P_i or not. The *correctness* requirement needs that: for $\forall$ m and $i \in [n]$, $Pr[\text{ShareVerify}_t(m, (i, \text{SigShare}_t(sk_i, m))) = 1] = 1$;
- *Combining algorithm:* $\text{Combine}_t(m, \{(i, \sigma_i)\}_{i\in S}) \rightarrow \sigma/\perp$. Given a message m, and a list of pairs $\{(i, \sigma_i)\}_{i\in S}$, where $S \subset [n]$ and $|S| = t$, this algorithm outputs either a signature σ for message m, or $\perp$ when $\{(i, \sigma_i)\}_{i\in S}$ contains ill-formed signature share (i, σ_i);
- *Signature verification algorithm:* $\text{Verify}_t(m, \sigma) \rightarrow 0/1$. Given a message m and a signature σ, this algorithms outputs 1 or 0 depending on whether σ is a valid signature for m or not. The *correctness* requires that: for $\forall$ m, $S \subset [n]$ and $|S| = t$, $\Pr[\text{Verify}_t(m, \text{Combine}_t(m, \{(i, \sigma_i)\}_{i\in S})) = 1 \mid \forall i \in S, \text{ShareVerify}_t(m, (i, \sigma_i)) = 1] = 1$.

Threshold coin-tossing: We assume the trust third party (dealer) have an unpredictable pseudo-random generator (PRG) $G : R \rightarrow \{1, \cdots, n\}^s$, that is know only to the dealer, which gets a string $r \in R$ and returns a set $\{S_1, S_2, \cdots, S_s\}$ size of s, where $1 \leqslant S_i \leqslant n$.

At the beginning of the protocol, the dealer gives a private function CShare_i to every node P_i, and two public functions: CShareVerify and CToss. Informally, given $f+1$ validated coin shares, the function CToss returns a unique and pseudorandom set [2]. Formally, the following properties are satisfied except with negligible probability:

- For all $i : 1 \leqslant i \leqslant n$ and for every string r, CShareVerify $(r, i, \sigma) = true$ if and only if $\sigma = \text{CShare}_i(r)$;
- If P_i is honest, then it is infeasible for the adversary to compute $\text{CShare}_i(r)$;
- For every string r, $\text{CToss}(r, \Sigma) = G(r)$ iff $|\Sigma| \geqslant f + 1$ and $\forall \sigma \in \Sigma$, $\exists$ a party P_i s.t. CShareVerify $(r, i, \sigma) = true$.

$(1, \kappa, \epsilon)$-Committee election (CE): Here we describe a committee election construction, the details of this election procedure CE is illustrated in the following Algorithm 4, the threshold coin-tossing scheme is the underlying Algorithm of CE.

Algorithm 4 Committee election (CE): for party P_i

Local variables initialization: $\Sigma \leftarrow \{\}$
upon CE(id) **do**
  $\sigma_i \leftarrow \text{CShare}_i(id)$;
  send (SHARE, id, σ_i) to all parties;
  wait until $|\Sigma| = f + 1$
    return CToss(id, Σ);
upon receiving (SHARE, id, S_j) from P_j for the first time **do**
  if CShareVerify(id, j, σ_j)=true **then**
    $\Sigma \leftarrow \Sigma \cup \{\sigma_j\}$;

Reliable broadcast algorithm (RBC) [34]: The detail of process of RBC shown in Algorithm 5. We remark that in RBC, the erasure code and Merkle-tree are adopted to reduce the communication. In this article, we adopt a $(n-2f, n)$-erasure code scheme in all scenarios, which tolerates the maximal adversary boundary and helps honest nodes recover the message efficient. Note that the RBC message complexity is $O(n^2)$ and the communication complexity is $O(n|m| + \lambda n^2 \log n)$ in expectation.

Algorithm 5 Reliable broadcast (RBC) for party P_i with sender P_s

if $P_i = P_{\text{sender}}$ and received input v **then**
  let $\{s_j\}_{j\in[n]}$ be the blocks of $(n-2f, n)$-erasure coding applied to v;
  let h be a Merkle tree root computed over $\{s_j\}$;
  send $\text{VAL}(h, b_j, s_j) := \{\text{VAL}, h, b_j, s_j\}$ to each party P_j, where b_j is the j^{th} Merkle tree branch;
upon receiving $\text{VAL}(h, b_j, s_j)$ from P_{sender} **do**
  multicast $\text{ECHO}(h, b_j, s_j) := \{\text{ECHO}, h, b_j, s_j\}$;
upon receiving $\text{ECHO}(h, b_j, s_j)$ from P_j **do**
  check that b_j is a valid Merkle branch for root h and leaf s_j, otherwise discard;
upon receiving valid $\text{ECHO}(h, \cdot, \cdot)$ message from $n-f$ distinct parties **do**

10: **interpolate** s'_j from any $n-2f$ leaves received;
11: **recompute** Merkle root h' and if $h' \neq h$ then abort;
12: **if** READY$(h) :=$ {READY, h} has not yet been sent, multicast READY(h);
13: **upon** receiving $f+1$ matching READY(h) messages **do**
14: **if** READY has not yet been sent, multicast READY(h)
15: **upon** receiving $2f+1$ matching READY(h) messages **do**
16: **wait** for $n-2f$ ECHO messages, then decode v.

Consistent broadcast (CBC) **[15]:** The CBC is a weaker version of RBC that has no totality. The detail of process of CBC is illustrated in Algorithm 6. Note that the CBC message complexity is $O(n)$ and the communication complexity is $O(n(|m|+\lambda))$ in expectation.

Algorithm 6 Consistent broadcast algorithm (CBC) for party P_i with sender P_s

1: **if** $P_i = P_s$ and received input value v **then**
2: multicast message (SEND, v);
3: **upon** receiving (ECHO, σ_j) from P_j for the first time **do**
4: **if** ShareVerify$_{2f+1}(s, v, (j, \sigma_j)) = 1$ **then**
5: $DS = DS \cup \{j, \sigma_j\}$;
6: **upon** $|DS| = 2f+1$ **do**
7: $\sigma \leftarrow$ Combine$_{2f+1}(s, v, DS)$;
8: multicast (Finish, v, σ);
9: **upon** receiving a SEND message (SEND, v) from P_s for the first time **do**
10: $\sigma_i \leftarrow$ SigShare$_{2f+1}(sk_i, s, v)$;
11: send message (ECHO, σ_i) to P_s;
12: **upon** receiving a Finish message (Finish, v, σ) from P_s for the first time **do**
13: **if** Verify$_{2f+1}(s, v, \sigma) = 1$ **then**
14: output v.

Asynchronous binary agreement (ABA) **[11, 31, 34, 36]:** The detail of process of ABA is illustrated in Algorithm 7. Note that the running time of one ABA is $O(1)$ in expectation, but that of n concurrent ABA is $O(\log n)$ in expectation. Besides, the ABA message complexity is $O(n^2)$ and the communication complexity is $O(n^2\lambda)$ in expectation. Two important remarks are in place:

Termination: deciding (outputting) v.s halting. As we mentioned briefly in section 3, the termination of ABA only requires *all* honest parties to decide/output a bit. The instantiation in Alg. 7 (without the amendment) indeed satisfies this (see more detailed analysis below). Admittedly, it is possible that some honest party decides a bit in round r, but still waits for messages in round $r+1$ (while other honest nodes may exit already) in the ABA protocol. This may waste some thread when the peer concurrently runs multiple threads after he decides. However, note that, once an honest party decides a bit, actions (invoke or not) on other modules of the ACS protocol is determined already, thus the safety of the ACS still holds.

As pointed out in the elegant work of [11, 31, 37], there could be a stronger termination condition that all honest parties not only decide, but also exit/halt the ABA instance. Following [11, 31, 37], we also give an amendment of Alg. 7, which satisfies the stronger termination condition. Since we mainly examine the effectiveness of our ABA reduction techniques to see advantages over HB-BFT, we use the same ABA instantiation (Alg. 7) in all those experiments (as revised HB-BFT [33]). Exploring further optimizations within ABA and its practical impact would be interesting future questions.

Algorithm 7 Asynchronous binary agreement (ABA) for party P_i :

1: **Initialization:** Upon receiving input b_{input}, set $est_0 = b_{input}$, $r = 0$, $e = 0$, $decided$=false, $values_r = \emptyset$ for $r = \{0, 1, 2, \dots\}$ and proceed in consecutive rounds number r for each consecutive epoch:
2: multicast Val$_r(est_r) :=$ {Val, r, est_r} to all;
3: **upon** receiving Val$_r(v)$ messages from $f+1$ nodes **do**
4: **if** Val$_r(v)$ has not been sent **then**
5: multicast Val$_r(v)$;
6: **upon** receiving Val$_r(v)$ messages from $2f+1$ nodes **do**
7: $values_r = values_r \cup \{v\}$;
8: **wait** until $values_r \neq \emptyset$
9: multicast AUX$_r[\omega] :=$ {AUX, r, ω}, where $\omega \in values_r$;
10: **wait** until at least $n-f$ AUX$_r[x]$ messages have been received, such that $val_r \subseteq values_r$ where val_r is the set of values x carried by these $n-f$ messages;
11: multicast CONF$_r[values_r] :=$ {CONF, $r, values_r$};
12: **wait** until at least $n-f$ CONF$_r[S]$ messages have been received, such that $S_r \subseteq values_r$ where $S_r = \bigcup S$ of set S carried by these $n-f$ messages;
13: $s \leftarrow coin_r()$; // see e.g., [16, 34]
14: **if** $S_r = \{b\}$ **then**
15: **if** $b = s\%2$ **then**
16: **if** $decided$=false **then**
once decide(b), later modules of the ACS could be invoked
17: decide(b); ▷ decide but not exit
18: $decided$ = true;
19: **else**
if (i). decide(b) in round r; (ii). $coin_{r'}\%2 = b$ in round $r' > r$
20: halt; ▷ exit
21: $est_{r+1} \leftarrow b$;
22: **else**
23: $est_{r+1} \leftarrow s\%2$;
continue looping from line 2
24: $r = r+1$; goto line 2;

Amendment

replace line 16-20 with line 25-26
25: **if** FINISH$(b) :=$ {FINISH, b} was not yet sent **then**
26: multicast FINISH(b);
include the following instructions before line 24
27: **upon** receiving $f+1$ FINISH(v) from distinct nodes **do**
28: **if** FINISH(v) was not yet sent **then**
29: multicast FINISH(v);
30: **upon** receiving $2f+1$ FINISH(v) from distinct nodes **do**
31: decide(v) and **halt**. ▷ decide and exit

All honest nodes deciding in Alg.7. Algorithm 7 is the fixed ABA protocol as in [33]. In the original HB-BFT [34] (without line 11 and line 12 in Alg. 7), it is possible for the adversary to trick some nodes to never decide. Concretely, the adversary can reconstruct the coin value before delivering certain messages, so that he can arrange the delivered messages to a party P_i and makes his condition always fail in line 15, thus P_i will repeat forever without deciding anything.

Now an extra round of CONF message was added, it guarantees that among honest nodes whose value S_r in round r contains only one bit after line 12, then they must be the same bit (honest parties could also have $\{0, 1\}$). The bad case that some honest node has 1 and some honest node has 0 after line 12 is taken place only when both $\mathsf{CONF}_r[0]$ and $\mathsf{CONF}_r[1]$ was multicasted for $2f + 1$ times in line 11. This means at least one honest user multicasted both $\mathsf{CONF}_r[0]$ and $\mathsf{CONF}_r[1]$, which leads to a contradiction.

In this way, we can derive the following two key lemmas: (1) at least one honest node would decide a bit; (2) if one honest node decides v in a round r, then all honest nodes would decide the same value v in a later round $r' > r$. To see lemma (1): in every two rounds, there must be one honest user who has one bit as value in line 14 (if all of them have $\{0, 1\}$, then they all assign the coin value as input for next round), thus he will have 1/2 probability to decide. To see lemma (2): since one honest node decides a bit in round r, all other honest nodes either decide the same bit, or have $\{0, 1\}$ as S_r, which will become a same bit (the coin value) in next round; The above analysis is similar to that in [11, 31, 36, 37].

小飞象拜占庭容错

——国际首个完全实用的异步共识算法

中科院软件所 张振峰研究团队

在第 27 届国际计算机与通信安全大会上，中国科学院软件研究所张振峰团队与美国新泽西理工学院唐强团队合作，在区块链核心技术的拜占庭容错（BFT）共识研究中取得重要突破，在国际上提出首个完全实用的异步共识算法“小飞象拜占庭容错（DumboBFT）算法”（简称“小飞象算法”）。这是在异步 BFT 共识算法设计领域，我国首次有重要研究成果在国际顶级会议上发表。

1. 小飞象算法动机和背景

共识算法是区块链的关键核心技术，保证在分布式的场景下，数量众多的区块链节点仅通过网络通信，便可以对用户提交的交易记录达成共识，从而共同维护一套“分布式帐本”、提供区块链这一重要的信息基础设施。区块链作为大规模的分布式信息基础设施，常常部署在复杂的广域网环境下，节点之间的通信链路不再稳定、甚至会被恶意控制。为保证区块链在广域网环境下稳定高效地工作，可以设计异步共识，解决一类被称为异步拜占庭共识问题的著名难题。

上世纪 80 年代初，美国工程院和科学院院士 Nancy Lynch 等揭示了异步共识的内在困难，给出了著名的“FLP 不可能性”，表明了确定性算法无法解决异步共识难题。“如何降低异步共识的通信、消息、运行时间等复杂度指标”，提高其可用性、支持安全且高效的“分布式账本”服务，被国际密码学会前主席 Christian Cachin 等人在 2001 年列为一个公开问题。

高性能异步共识算法的匮乏，迫使大部分科技巨头的区块链项目转而寻求安全性更弱的共识机制，比如 Facebook Libra 区块链采用的 Hotstuff 算法和 IBM HyperLedger Fabric 计划采用的 PBFT 算法。此类共识机制无法抵御恶劣广域互联网环境下可能的传输中断和网络封包延迟，存在拒绝服务攻击风险，使得用户的合法消息得不到及时处理，成为了区块链产业发展健康发展的安全隐患。

2. 异步共识算法研究现状和问题

在这一背景下，高性能异步共识算法的研究成为了国际上前沿性的研究热点和难点，多位“图灵奖”得主在内的众多顶尖专家设计了多种随机化异步共识算法、突破“FLP 不可能性”，图灵奖得主 Michael Rabin 等人设计了首个随机化的异步共识算法，可容忍 1/8 恶意节点；图灵奖得主 Silvio Micali 等人就单比特输出给出了容忍 1/4 恶意节点的异步共识算法，国际密码学会会士 Ran Canetti、Tal Rabin、Victor Shoup 和国际计算机学会会士 Dahlia Malkhi、Dawn Song 等人此后设计了各类容忍 1/3 恶意节点的随机化异步共识算法，但这些构造大多复杂度极高，仅仅停留在理论层面、难以实用。

2016 年，国际计算机学会会士 Dawn Song、伊利诺伊大学香槟分校教授 Andrew Miller、康奈尔大学教授 Elaine Shi 等提出了一个接近实用的异步共识算法——蜜獾 BFT（Honeybadger BFT）。该设计引入了门限加密技术，能够更加高效地抵抗敌手针对特定交易的拒绝服务式攻击，它还采用了抹除码和哈希树承诺树等关键技术，大大减少了异步广播组件的通信开销。但该设计必须使用 N 个并行的异步二元共识组件（其中 N 为共识节点数量）。当 N 个随机化的异步二元共识算法一起运行时，往往有少数该模块停机很慢，导致 log(N)级别的运行时间，这一问题大大增加了 Honeybadger BFT 整体上的确认延迟，严重拖累了整体性能和效率，在全球百节点的测试环境下，产生一个区块耗时近 500 秒、吞吐量不足每秒 2000 比交易。

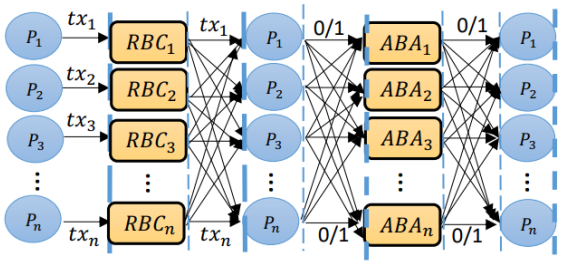


图 1：Honeybadger BFT 执行过程。

Honeybadger 异步共识算法具有 log(N)级别的运行时间、且消息复杂度为三次方，未能全面回答 Cachin 等关于异步共识算法可用性的公开问题，在大规模网络环境中性能表现不佳，无法全面满足区块链信息基础设施系统对共识算法的性能要求。尽管如此，Honeybadger 已经被少数工业界的区块链平台使用。

3. 小飞象 BFT：让异步 BFT 真正走向实用

小飞象深刻剖析了 Honeybadger 性能瓶颈的本质原因——N 个随机化异步二元共识造成的运行速度增大，提出了一个全新的异步共识设计思路，能够

将 N 个随机模块减少到平均 2 个，从而将运行时间从 log(N)级别减少到常数，实现了延迟和吞吐量上质的飞跃。

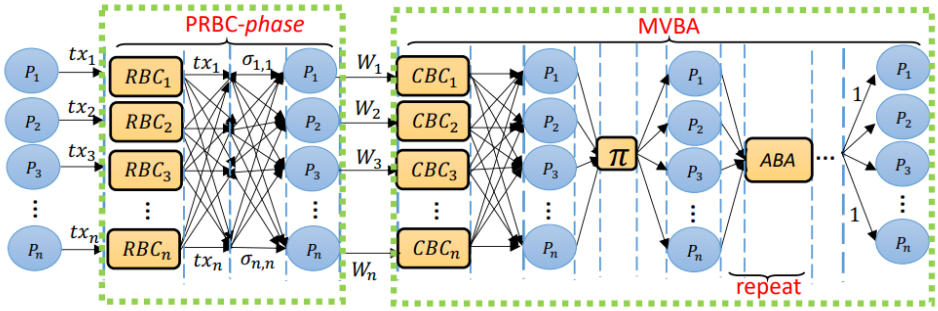


图 2：Dumbo BFT 执行流程。

Dumbo BFT 在设计上有两大要点：一是提出和构造了一种全新的带证明的可靠广播原语（provable reliable broadcast 简称 PRBC），保证交易正确一致地广播到全网所有节点的同时，产生一个短小的证明，任何人都可以验证该证明是否有效，从而判断这笔交易的广播是否真的已经完成；二是巧妙地利用了带验证的多值异步拜占庭共识算法(multi-valued validated asynchronous byzantine agreement 简称 MVBA)，利用其输出的共识结果一定能够满足预设的外部断言函数这一特点，将对交易的共识转化为对证明这类短小元数据的共识，实现了常数时间内的高效、快速共识。

更具体地说，在 Dumbo BFT 的执行过程中，每个节点都通过一个带证明的可靠广播向全网的其他节点广播一笔交易（或一批交易）tx_i，和普通的可靠广播不同，带证明的可靠广播保证每个节点输出交易的同时得到一个证明表明这个广播的确已经正确的完成；随后，每个节点等待 N-f 个可靠广播完成，此时也获得了这 N-f 广播已经完成的证明，并把它们作为一个证明集合 W_i 输入带验证的多值异步拜占庭共识算法，通过设置该共识模块的断言函数、保证该算法输出的共识结果一定为 N-f 个有效的广播证明,这 N-f 个证明一定能够保证 N-f 批交易被正确的广播到了全网，这些交易即为这一轮算法交互执行后最终达成的共识结果。最后，为了预防敌手针对特定交易发起拒绝服务式攻击，上述过程中所有的交易都通过门限加密算法提前进行了加密，以密文形式达成共识后再通过门限解密算法对交易密文进行解密，得到加密前的原始交易。

上述一轮执行过程仅有 N 个确定性的 PRBC 广播和 1 个随机化的 MVBA 多值异步共识（可由期望两个异步二元共识实现），相比 Honeybadger 采用 N 个随机化异步二元共识的设计思路，大大减少了执行时间。

除了大幅度提升了异步共识效率，小飞象 BFT 另一大理论贡献是确认了：HoneyBadger BFT 之前认为无法从 MVBA 多值异步共识算法出发构造实用的异步区块链共识算法的观点是过于悲观的，给出了从 MVBA 构造实用异步共识的一种全性的高效方法，在异步 BFT 共识算法设计理论上做出了突破。

在一百个节点的大规模全球测试网络中，Dumbo BFT 延迟仅为 Honeybadger 的 1/20，而吞吐量则是 Honeybadger 的 9 倍。

表1：蜜獾协议和小飞象协议的实际性能对比（在全球四大洲测试网络中）

	64个全球节点时		100个全球节点时	
	确认延迟	交易吞吐量	确认延迟	交易吞吐量
蜜獾协议	240 秒	4453 笔/秒	491 秒	1934 笔/秒
小飞象协议	14 秒 (延迟降低 94%)	18692 笔/秒 (吞吐量增长 320%)	24 秒 (延迟降低 95%)	17767 笔/秒 (吞吐量增长 819%)

4. 小飞象 MVBA——渐近最优的多值异步共识

小飞象 BFT 在设计理论上的突破实现了异步共识性能质的飞跃。那是否存还能够提出性能更优的异步共识设计理论，设计渐近性能最优的异步共识算法？要回答这一问题，需要设计理论最优的 MVBA，从而直接使用 MVBA 来共识交易（而非广播证明等元数据）。团队成员路远、卢振亮等人针对这问题，提出了 Dumbo-MVBA 这一全新的 MVBA 设计理论，用其构造的异步区块链共识算法能够在运行时间、消息数量、通信代价等多方面达到渐近最优，圆满回答了 Cachin 等关于异步共识可用性的 20 年公开问题。

表2：蜜獾协议、小飞象多值共识协议的复杂度对比（红色表示非理论最优，蓝色为理论最优）

	系统规模为N个共识节点			
	随机化子模块数量	通信代价（比特）	运行时间	消息数量
蜜獾协议	O(N)	O(N)	O(logN)	$O(N^3)$
小飞象多值共识协议	O(1) 即理论最优	O(N) 即理论最优	O(1) 即理论最优	$O(N^2)$ 即理论最优

Dumbo-MVBA 和 Dumbo-BFT 两个结果彼此印证，在方法论层面确认了 MVBA 才是设计异步共识的正确技术路线，也为异步共识算法的应用和部署提供了关键核心技术。

云原生+KUBE 编排

云原生+Kube 编排

(Cloud Native+Kubernetes/Docker+Kube)

陆首群

2017 年 6 月 10 日

云原生代表云计算的发展方向，甚至今天已成为云计算的主流。云原生包括云计算实行容器化，建立分布式运行环境和弹性业务平台， 配置虚拟化编排（竞争中唯一胜出者 Kubernetes,简写 Kube），开发微服务（并可派生无服务器、边缘计算），推行开发运维一体化管理（Dev Ops）等。云原生将平衡敏捷灵活与隔离安全双优功能，高效配置、调度计算机、网络、存储方面的资源，高效迁移公有网、私有网、混合网方面的资源；建立具有敏捷部署、弹性伸缩、灵活调度、故障自动恢复等优势的新型 Paas 平台；实现开发运维一体化，改进持续交付管理。

早在 2015 年云原生解决方案面世时，就受到全球四大云计算公有网的支持（亚马逊的 AWS、微软的 Azure、阿里云、谷歌的 Google Cloud Plate）,也受到 IBM、华为、腾讯、Oracle、VMware、百度、以及 Open Stack 等公有网、私有网、混合网的支持。

微服务（Micro Services）

所谓微服务是一项在云原生容器中部署应用和服务的新技术，围绕业务创建可独立开发、管理和加速的应用，它将使部署、管理和服务功能交付变得更加简单，更易升级和扩展。随着容器技术的快速发展，越来越多的企业从一体化应用转型迁移到了微服务架构。微服务 除带来良好的设计和架构的理念，以及如上新技术外，也引入了微服 务治理的难度，即在微服务架构中，治理安全、流量控制、监控、日志等方而有难度，在处理 A/B 测试、图像发布、访问控制、策略管理 等方面有复杂性，为解决这些治理问题，采用具有负载均衡、服务认证、监控等功能的服务网格技术 Istio,可取得园满解决。

无服务器、边缘计算、人工智能、物联同、区块链、5G 等可称为云原生新兴应用的场景。

无服务器(Serverless)

云原生+ Kube 支持作为云中继的无服务器。

由第三方向用户提供、无需由用户和云原生开发者安装、管理和维护的云服务，实行按需使用、计费、收费(用户用完交费后即可离 开)。

由云原生开发者实现的服务端逻辑运行在无状态的计算容器中： 它由事件触发后，完全由第三方管理；其业务层面的状态则被开发者 使用的数据库和存储资源所记录；客户端逻辑和服务托管依赖第三方，远程代码托管环境由第三方(如亚马逊的 AWS、阿里云等)提供的。

边缘计算(Edge Computing)

现有中心化的云计算模型日渐暴露其不足:数据存储与计算的重心逐渐远高中央数据中心走向边缘。边缘计算是在靠近物或数据源头 的网络边缘侧，融合网络、计算、存储、应用核心能力的开放平台，就近提供边缘智能服务，满足行业或用户数字化在敏捷联接、实时业务、应用智能、安全与隐私保护等方面的需求。

云原生(+Kube)支持边缘计算，可用虚拟化编排(Kube)管理“边缘网络节点(Kube Edge)"，并以“Kube Edge Bus"处理边缘云网络。

虚拟化编排管理 Kubernetes (Kube)

Kube 是一款谷歌原创的虚拟化编排管理软件，用以高效配置、 调度资源(计算机、网络、存储中的资源)或高效部署容器化应用，也可用以高效调度、迁移公有网、私有网、混合网中的资源。

在两年内，Kube 与同质的编排软件 Docker Swarm、Cloud Foundry Diego、Kontena、Apache Mesos、Amazon ECS…等竞争中脱颖而出，拿下了皇冠，因此“云原生+Kube”便成为事实上的标准。

后来谷歌将 Kube 捐献给云原生基金会(CNCF) ，CNCF 将基金会办成“Docker+Kube”的孵化器/开源社区，吸收中外专家联系其应用场景进行孵化工作。

“云原生容器化+Kube 编排”可以建立在各种云架构上，“云原生容器化+Kube 编排”可能建立在裸机上，这时“云原生容器化+Kube 编排”就成为建造云平台的平台。

Istio 与微服务治理有关。Istio 服务网格技术是一个用来连接、保护、管理和监测微服务的开源平台。提供一种简单的方式来为已部署的微服务建立网络，该网络具有负载均衡、服务间认证、监控等功能，而且不需要对微服务的代码做任何改动。

使用 Istio 进行微服务流量管理，进行安全管理，用来监控和可视化微服务，进行跨云管理。Istio 使用案例：Serverless 平台 Knative

Istio 有助于将不同的微服务转化为一个集成的服务网络。

附件一：陆首群 2018.10.12

“云原生+Kube 编排(Cloud Native+kubernetes/Docker+Kube)”

主题演讲大纲

- 云原生(Cloud Native+kube)代表云计算的发展方向
- 云原生(Cloud Native+kube)已成为云计算的主流。
- 云原生解决方案面世时，就受到全球四大云计算公 有网的支持（亚马逊的AWS、微软的Azure、阿里云、谷歌的Google Cloud Plate），也受到IBM、华为、腾迅、Oracle、VMware、百度、以及 Open Stack等公有网、私有网、混合网的支持。

云原生+(Cloud Native+kube)

- 实行（云计算）容器化（Docket）
- 配置虚拟化编排（Kubernetes,Kube/K8s）(一种开放的治理模式)
- 建立分布式运行环境和弹性业务平台
- 开发微服务（MicroService）+服务网格技术Istio; 派生无服务器（Serverless）、边缘计算（Edge Computing）;
- 推行开发运维（Dev Ops）一体化管理
- 建立敏捷部署、弹性伸缩、灵活调度、故障自动恢复等优势的新型Paas平台
- 改进持续交付管理

Kubernetes(Kube) 编排

- Kubernetes是一款谷歌原创的开源的虚拟化管理软件，现在谷歌已将它捐献给云原生基金会(CNCF)
- Kube可以高效配置、调度计算机、网络、存储中的资源或部署容器化应用，也可高效调度、迁移公有网、私有网、混合网的资源。
- 两年来，Kube与同质容器编排软件Doker Swarm、Cloud Foundry Diego、Kontena、Apache Mesos、Amazon ECS…等竞争中脱颖胜出， 使“云原生+Kube”成为事实上的标准。

云原生+ Kube 编排

- 可建立在各种云架构上
- 也可建立在裸机上
- “云原生+ Kube编排”就成为建造云平台的平台

附件二:云原生+ Kube 编排问题

1. 容器和虚拟化编排如何在云上配置?

容器编排平台如 Kubernetes 等可以不需要在云计算平台上配置而直接部署使用，如 IBM Cloud private 可以通过 ansible 脚本实现自动部署。而对于一些其他平台，如 Cloud Foundry 等，在最新版本中已经包含了 Kubernetes 平台，所以仅需配置 Cloud Foundry 平 台即可。对于其他 laaS 平台，目前主要有集中解决方案，一种是利用已有的分布式应 用部署工具，如 BOSH,Ceph 或者 Puppet 等，实现在不同 laaS 平台上读部署，另一种是某 些公司已经针对特定的 laaS 平台，开发的一些开源或者闭源的部署方案，例如 EasyStack 公司开发了有关 Kubernetes on OpenStack 的部署工具等。

2. 各自发挥什么作用？合起来起什么作用？

容器编排平台主要针对批量容器的生命周期以及复杂分布式应用的运行维护作出容器级别的编排、维护、升级以及监控等功能。其在 laaS 平台之上可以利用 laaS 平台所提供的基础设施即服务等功能实现计算等资源的弹性伸缩、SLA 以及多租户管理等功能。

3. CNCF 下一步在研究什么?

CNCF 在云计算的各个领域都有所涉及，目前的主要研究方向在于:

1）服务网格(Service Mesh)

Service Mesh 是用于处理微服务间通信的基础设施层，他负责通过构建现代云原生应用程序的复杂拓扑结构来可靠地传递服务请求。Service Mesh 通常以一种应用程序容器透明的方式构建微服务容器之间的网络，控制服务之间的请求传递。目前，较为流行的微服务平台包括 Istio 以及 CNCF 旗下的 envoy 和 linkerd 等。

2）Tracing & Monitoring

Tracing 是用于在复杂分布式云环境下服务和应用程序以及云平台自身程序依赖关系和性能的工具，通过在应用程序和平台上建立分布式的跟踪上下文来确定应用程序的结构和调用关系。Monitoring 是监控系统中各部分组件健康状况、运行状态的工具。目前在 CNCF 旗下已经收纳了包括 OpenTracing、Jaeger、Prometheus 等在内的 Tracing 和监控 平台。

3）Serverless Computing

“无服务器”是指程序代码运行在服务器端，无需管理，无需考虑服务器端需要关心的内容，例如资源管理、程序安装、运维、横向扩展等，就像面对的是海量服务器一样。“无服务器”技术，通常也被称为“函数即服务”(Function as a Service),目前 CNCF 已经成立了 Serverless 工作组来组织相应无服务器工作。

4) Container Runtime

容器运行时指用于管理和运行容器的各种容器引擎。在 Docker, CoreOS 等倡议和领导下?年发布的 Open Container Initiative 逐渐成为了容器引擎的实际标准。CNCF 旗下目前包括了 Rkt 和 Containerd 两个容器引擎项目，这两个项目也在许多容器编排平台中得到了广泛的应用。

4. “无服务器”(容器技术)是什么概念?

“无服务器”是指程序代码运行在服务器端，无需管理，无需考虑服务器端需要关心的内容，例如资源管理、程序安装、运维、横向扩展等，就像面对的是海量服务器一 样。

“无服务器”技术，通常也被称为“函数即服务”（ Function as a Service），IBM 函数、微软函数、谷歌函数、阿里函数等都采用这个概念，有以下几个特点：

程序代码以函数为单位运行在后端，无需管理。

应用开发者不需要关心基础设施、操作系统、运行环境等任何服务器端的要素。

服务器端的逻辑仍然由程序代码实现，运行在第三方平台上。

程序代码是无状态的。

程序代码的运行是暂时性的，接收请求后，则分配计算资源，启动该代码；请求满足后，则结束代码，收回计算资源。通常有一定限制，如限制运行内存、运行时间等等（当然，可以定制）。

程序代码可以由 API 网关直接调用，或者事件触发。

更宽泛的，也有人将“后端即服务”(Backend as a Service）称为无服务器。亚马逊提到的无服务器架构指由第三方云计算供应商负责后端基础结构的维护，以服务的方式为开发者提供所需功能，例如数据库、消息，以及身份验证等。这里就采用的是这种 宽泛的感念。“容器技术”是指基于 Linux Container 的一种内核虚拟化技术，可以提供轻量级的虚拟化，以便隔离进程和

资源。现有的“无服务器”技术中的代码隔离和代码运行都是基于容器虚拟化技术实现的。

Kubernetes 解析大纲

IBM 中国开放技术工程院 田忠

Kubernetes 于 2014 年诞生于谷歌，并逐渐成为管理、编排容器的主流方。2016 年谷歌将它捐献给云原生计算基金会（Cloud Native Computing Foundation，CNCF），构成发展云原生技术的孵化器，并由 CNCF 负责维护。自 2017 年起大量中外企业加入 CNCF 基金会，发展云原生技术并为 Kubernetes 扩展其应用。谷歌、红帽、IBM 和许多中国公司的开发者都为 Kubernetes 贡献了大量代码。2019 年，Kubernetes 被评为世界十大软件之首。

Kubernetes 是一组用 go 编写的应用程序。这些应用程序一起形成了管理容器的交互式平台，旨在提供一个跨机群的“应用程序容器的自动部署、扩展和运维系统”，它适用于包括 Docker 在内的一系列容器工具。利用容器和容器编排，模块化应用程序设计渐成主流，其中数据库是独立的，并且无需对机器物理扩展即可对应用程序进行扩展。容器化正在革新开发者编写和部署代码的方式。容器加快并简化了代码的编写和部署过程。

Kubernetes 为开发者带来许多激动人心的特征：

- Kubernetes Pod支持将一个或多个容器作为单个原子单元进行部署。
- 服务可以使用固定的地址访问一组Pod，并使用集成IP和基于DNS的服务发现功能将这些服务链接在一起。
- 复制控制器（replication controllers）确保指定数量的Pod的始终运行，并使a用标签（label）来识别Pod和其他Kubernetes对象。
- 借助功能强大的网络模型，可以跨多个主机管理容器。
- 编排存储的能力允许您可以在容器内运行无状态和有状态服务。
- 简化的编排模型可以让应用程序快速运行，而不需要复杂的两层调度器。
- Kubernetes的架构设计同时兼顾开发者和运维人员的不同需求，避免为折衷而牺牲任何一方的情况。

2017 年底，来自 IBM 开放技术工程院的志愿者组织了一个十个环节解析 Kubernetes 的大纲：

1. Kubernetes 初探。从容器技术入手，介绍K8s平台的技术特点和架构，并通过实际应用案例介绍如何将企业应用部署到以K8s为核心的云平台。
2. 上手 Kubernetes：基本概念、安装和命令行工具kubectl。从多种安装方式入手，讲解Kubernetes是如何安装到不同应用环境，具体介绍

kubectl的使用，从而带领大家从命令行角度了解Kubernetes的各种能力。

3. Kubernetes 的资源调度。共享资源（CPU、内存等）的有效管理对于提高集群的利用率和隔离性至关重要。这一环节介绍如何同时实现资源的利用、分离和可用性目标，重点介绍 Kubernetes 1.6 / 1.7 中提供的细粒度控制的特点，并描述了如何使用它们来实现各种具体的策略目标。
4. Kubernetes 的运行时：Kubelet。Kubelet是Kubernetes的核心模块之一，负责在每一个K8s节点上运行和管理Pod所依赖的各种容器，实时监控这些容器的状态。通过像CRI的标准接口，Kubelet可以接入各种不同的容器引擎，如Docker和Rkt。
5. Kubernetes 的网络管理。网络是实现应用访问和互联非常重要的一环。这里介绍Kubernetes如何通过C N I的标准接口，iptables等技术管理集群的容器网络和服务网络，实现内部的网络互通和负载均衡。
6. Kubernetes 的存储管理。应用程序在部署和运行的过程中时刻需要消费或者产生数据，其中许多数据是需要持久化存储的。Kubernetes作为一个通用的应用程序容器部署和运行平台，提供了灵活的持久化存储层供用户存储应用程序数据。
7. Kubernetes 的日志与监控。日志和监控数据是保证云计算平台及云应用程序正常运行的关键性数据。各种云计算平台，自其创建之日起，日志和监控就是其重要组成部分。 Kubernetes作为一个通用的应用程序容器部署和运行平台，也提供了强大的监控和各种日志数据供用户使用。
8. Kubernetes 的用用部署。Helm是一个开源工具，用于安装和管理Kubernetes上的应用程序。它将运行一个应用所需要的镜像、依赖和资源定义等打成一个包来管理。本讲会详细介绍如何定义包以及如何利用Helm简化Kubernetes 应用部署和管理。
9. 扩展Kubernetes 生态：service catalog的概念与应用。能否接入外部服务和企业遗留系统是考察PaaS平台能力的重要指标之一。Kubernetes Service Catalog遵循开放的Open Service Broker API设计，能够方便地让企业接入各种不同的外部服务和系统到Kubernetes平台为其它应用所用。
10. Kubernetes 的企业实践。企业级PaaS平台在实际运行、部署和维护中存在各种各样的实际问题。本环节以某银行PaaS架构和实际应用场景为例，介绍其云平台的主要功能。内容涉及架构介绍、安装部署、弹性伸缩、应用商店、外部服务集成、微服务治理、多云集成等。

自动驾驶与无人驾驶

百度 Apollo 自动驾驶平台

百度公司

总体情况

Apollo 开放平台是一个开放的、完整的、安全的平台,可以帮助汽车行业及自动驾驶领域的合作伙伴结合车辆和硬件系统,快速搭建一套属于自己的自动驾驶系统。自 2017 年开源以来,百度 Apollo 开放平台发布了 9 个版本,从简单循迹出发,逐步增加支持高速场景自动驾驶、城市场景自动驾驶。

2020 年,Apollo 开发者社区已拥有来自全球 97 个国家,55000 多名开发者,涵盖 221 多所高校,308 多家科技公司,57 多家车企。目前,社区累计发布内容超过 1200 篇,其中开发者贡献内容 60 余篇,内容涵盖开发环境搭建、技术解析、前沿论文解读、技术问题解答等,如下图所示。

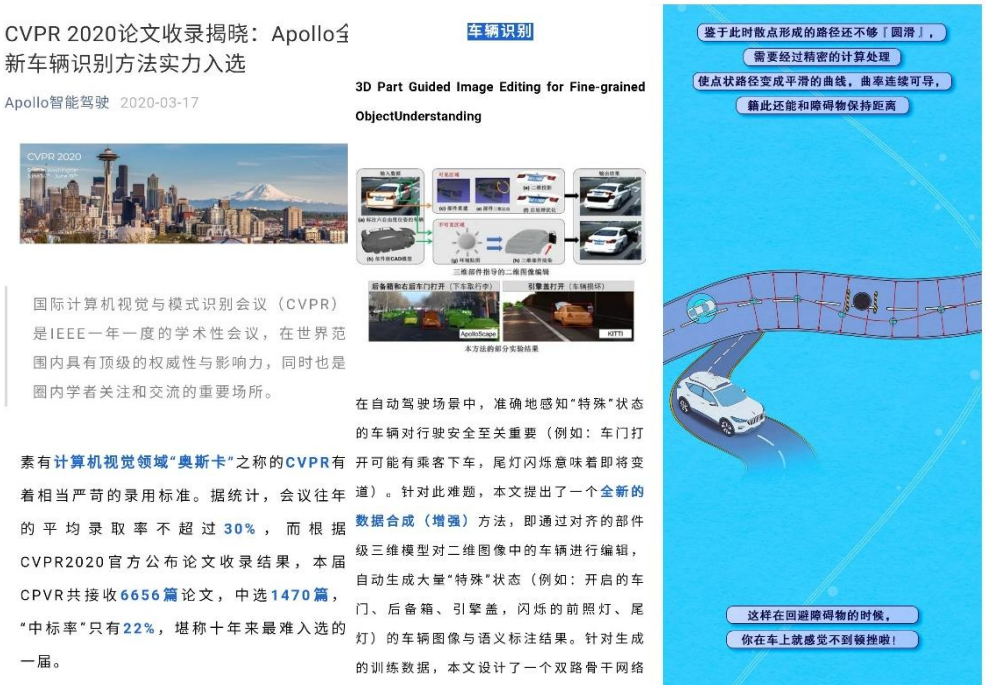


图 1.1 Apollo 社区发布的技术文档

百度外的社区开发者贡献代码日趋增多，社区的互动活跃，百度自身也对社区问题建立了有效的反馈机制。此外,面向自动驾驶行业开发者的研发产品—Apollo 自动驾驶开发套件(Apollo D-KIT),已为 100 多家高校、科研机构及行业伙伴提供研发支持。作为全球最大自动驾驶开放平台,Apollo 拥有生态合作伙伴

达 210 家,几乎囊括全球主流汽车制造商、一级零部件供应商、芯片公司、传感器公司、交通集成商、出行企业等,覆盖从硬件到软件的完整产业链。

截止到 2020 年底，Apollo 开放源码已发布到 6.0 版本，如下图所示。Apollo 6.0 集成了新的深度学习模型，以增强自动驾驶的能力，该版本与新增的 data pipeline 服务无缝配合，更好地服务于 Apollo 开发者。阿波罗 6.0 也是第一个集成了特定功能的版本，以展示我们对无人驾驶技术的不断探索和努力。

图 1.2 Apollo 开放源码历年版本

目前，百度 Apollo 中国专利公开数量达到 2900 件，并在全球主要国家都进行了专利布局。其中，专利“无人车、无人车定位方法、装置和系统”（公告号 CN106023210B）和专利“基于点云数据的车辆轮廓检测方法和装置”（公告号 CN107025642B）分别获得第二十届、第二十一届中国专利奖银奖和优秀奖。“基于点云数据的车辆轮廓检测方法和装置”的方案在全球首次提出结合深度卷积神经网络和三维点云数据，极大提高了车辆检测准确率，不仅已应用到百度研发的各类自动驾驶车辆中，还应用到 Apollo 开源框架中，将百度卓越的技术与世界共享，推动了全球自动驾驶技术的发展。

2020 年，百度 Apollo 自动驾驶平台发布了“智驾、智舱、智图、智云”四大智能化解决方案，为行业带来“乐高式”的领先一代整车智能化的解决方案。其中，Apollo 智舱与超过 70 家车企 600 款车型展开合作，实现超过 100 万台的小度车载 OS 前装量产搭载。Apollo 智图包含给人看的车机导航地图和给车看的高精地图，以及用于城市治理的动态孪生地图，Apollo 高精地图已多年蝉联市占率第一名。Apollo 智驾已开启大规模量产，与广汽、威马、长城等品牌开展量产合作。从智能驾驶出发，Apollo 正走向更大的智能交通及城市数字交通运营。数字交通运营商模式改变了过去传统智能交通建设的业态，由一次性集成商模式改为持续性运营商模式，通过贴合实际场景持续升级的车路智行算法，提升整个城市通行的效率，解决过去靠人力、传统基建解决不了的问题。

百度 Apollo GO 是全球唯一在多城开展 Robotaxi（自动驾驶出租车）与 Robobus（自动驾驶巴士）运营的出行服务，目前 Apollo GO 测试城市已覆盖至 27 个，在全国拿下近 200 张自动驾驶测试牌照。在长沙、沧州、北京三地开放

服务，设置安全停靠站点 620 个，路网覆盖区域达到 391 平方公里。在多城市运营过程中，Apollo GO 不断面对多变的气候、差异化的交通场景和快速部署的挑战。多城多车型目前已完成自动驾驶载人出行服务 21 万人次，其中单城日订单峰值达到 2703 单，整体用户满意度超过 95%。

2021 年 2 月 5 日，《北京市自动驾驶车辆道路测试报告（2020）》正式发布。该报告是中国目前官方唯一的自动驾驶路测报告，已经连续三年发布。在北京市众多的测试企业中，百度 Apollo 作为其中的代表，也连续第三年成为投入测试车数量最多、测试里程最长的企业。百度 Apollo 在北京的测试，也开创了北京市公开道路无人化驾驶测试的先例。

图 1.3 百度无人化测试

在产品形态上，Apollo GO 也不断拓展，并从示范运行向大规模商业化部署加速。以 Robotaxi 为例，早期提供服务的车辆主要为林肯 MKZ 改装车型，目前 Apollo GO 已拓展至一汽红旗量产车型，该车型采用了百度自研的自动驾驶计算平台 HW、自动驾驶操作系统 CarOS 等软硬件，并在进行广汽新能源车型的升级探索；服务车型种类上，Apollo GO 囊括从乘用车、大型巴士、迷你巴士，到道路巡检车、特种作业车在内的丰富车辆。Apollo Go 的服务规模持续壮大，百度计划在未来 3 年将 Apollo GO 自动驾驶出租车服务拓展至 30 个城市，部署至少 3000 辆自动驾驶出租车，为 3000 万用户提供服务。

百度在测试手段和测试里程方面都处于行业领先水平。关于测试手段，自动驾驶无人化测试阶段大体分为以下三个阶段：第一阶段主驾安全员+副驾安全员+远程安全员道路测试。第二阶段为副驾安全员+远程安全员道路测试，处于自动驾驶技术无人化大规模验证中期，自动驾驶能力已经相对完善。第三阶段为车内无人+远程安全员道路测试，处于自动驾驶技术无人化大规模验证末期，

自动驾驶能力已经基本完善。目前百度处于第二阶段，并向第三阶段突破。关于测试里程，尽管 Waymo 在美国亚利桑那州凤凰城共行驶 610 万英里（约 982 万公里），其中包括 2019 年至 2020 年前 9 个月中的 6.5 万英里（约 10 万公里）无人驾驶里程，但是中国的城市道路情况要远比美国复杂的多。截止到 2020 年 12 月，百度累计开展测试里程超过 700 万公里，其中在北京、长沙两地无人驾驶测试里程已超过 5.2 万公里。同时百度会通过仿真等手段，持续在这一领域领跑。

另外，百度牵头了部分自动驾驶相关国际标准。百度作为 7 个创始成员之一，参与了全球第一个专门针对自动驾驶应用安全标准 ISO/TR 4804:2020 Road vehicles — Safety and cybersecurity for automated driving systems — Design, verification and validation，截止目前百度是唯一中国公司代表，主导了 L3-L5 级系统设计参考，高精地图、定位等部分内容的撰写。同时百度作为参与单位或创始成员参与的其他国际标准还有 ISO 21448 SOTIF、IEEE P2846 行为安全部分、IEEE P2851、ISO/SAE 21434 信息安全等。

开放方式

数据开放方法：

- 摄像头感知的图像、LiDAR 扫描的点云等数据，通过 http://apolloscape.auto/平台开放。平台详细定义相应数据集格式，如 Scene parsing 数据格式定义为：{root}/{type}/{road id}_{level}/{record id}/{camera id}/{timestamp}_{camera id}{ext}，其中如 type 为：ColorImage，Label 和 Pose 等。
- 仿真场景数据，通过通过 http://bce.apollo.auto/平台开放。仿真场景分为 Worldsim 和 Logsim 两类。Worldsim 是由人为预设的障碍物行为和交通灯状态构成的场景，可以简单高效的测试自动驾驶车辆，但缺乏真实交通环境中复杂的情况；Logsim 是由路测数据提取的场景，提供复杂多变的障碍物行为和交通状况。

在线服务开放方法：

- 在线仿真服务，通过 http://bce.apollo.auto/平台开放。提供的仿真服务包括：碰撞检测、闯红灯检测、限速检测、在路检测、到达终点检测、急刹检测、加速度检测、导航检测等。支持多场景的高速并发运行、单个或多个模块的算法验证、3D 动画展示。
- 其他如自动驾驶传感器标定、安全等服务，通过 https://apollo.auto 平台开放。自动驾驶传感器标定开放的内容包括：车辆动力学标定、激

光雷达/摄像头感知设备标定等。安全方面提供渗透测试等服务。

软件开放方法包括：

- 完整、安全的自动驾驶平台软件源码，通过 Github、OpenI 等开放源代码平台开放。开放的内容包括感知、仿真、高精定位、决策与规划、智能控制等功能，同时对底层的操作系统（Ubuntu 18.04）、编译器（GCC 7.5 with full support of C++ 14）、编程语言、依赖库也提供了详细说明。用户可以基于这些源码自行修改和改进，满足开发的需要。开放源码采用对商业化友好的 Apache License, Version 2.0 协议。

硬件开放方法包括：

- 硬件、车辆接口规范，通过 Apollo 官方网站开放。厂商按照接口规范对硬件和车辆进行适配、测试，最后经过开放平台测试验证后，平台给予官方认证（认证与 Apollo 平台对线控车辆需求的兼容性），并开放给更多用户使用。开放的规范和工具包括：a）乘用车线控需求规范、微型车线控需求规范等，b）车辆适配代码开发说明及示例、工具。厂商通过这些工具快速生成 Apollo 参考车辆适配层模版。厂商提供的认证材料包括：a）待认证车辆接口指标自测数据（含车辆动力学参数等），b）车辆改装技术方案，c）接口规范文档反馈表，d）DBC 文件等。
- 开放已适配好的软硬件一体化的自动驾驶车辆（已对线控底盘及整套传感器硬件完成适配），帮助开发者快速进入实车验证，加速自动驾驶研发进程。该软硬件一体化车辆开放的自动驾驶能力包括：a）循迹自动驾驶，b）激光雷达避障自动驾驶，c）摄像头避障自动驾驶等。开放的硬件包括：a）桁架式支架、传感器，b）一体式设备舱+整体桥+拖拽臂底盘结构，c）16 线激光雷达，d）毫米波雷达，e）双天线 GPS+IMU 组合导航等。该自动驾驶车辆提供多冗余安全机制，配备车辆遥控器及碰撞传感器急停系统，可手操控制车辆行驶，一键快速接管车辆控制权，保障人员与车辆安全。

Apollo 自动驾驶系统功能介绍

Apollo自动驾驶系统的软件模块如下图所示。

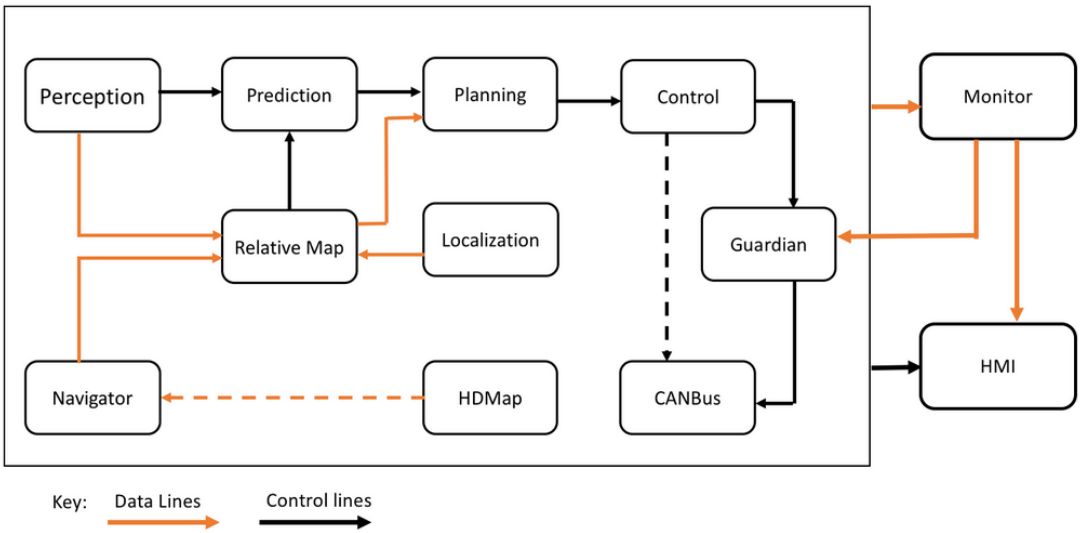


图1.4 Apollo的软件模块

具体包括：

- 感知：感知模块识别自动驾驶车辆周围的世界。感知中有两个重要的子模块：障碍物检测和交通灯检测。
- 预测：预测模块预测感知障碍物的未来运动轨迹。
- 路由：路由模块告诉自动驾驶车辆如何通过一系列车道或道路到达其目的地。
- 规划：规划模块规划自动驾驶车辆的时间和空间轨迹。
- 控制：控制模块通过产生诸如油门，制动和转向的控制命令来执行规划模块产生的轨迹。
- CanBus：CanBus 是将控制命令传递给车辆硬件的接口。它还将底盘信息传递给软件系统。
- 高精地图：该模块类似于库。它不是发布和订阅消息，而是经常用作查询引擎支持，以提供关于道路的特定结构化信息。
- 定位：定位模块利用 GPS，LiDAR 和 IMU 的各种信息源来定位自动驾驶车辆的位置。
- HMI：Apollo 中的 HMI 和 DreamView 是一个用于查看车辆状态，测试其他模块以及实时控制车辆功能的模块。
- 监控：车辆中所有模块的监控系统包括硬件。
- Guardian：安全模块，用于干预监控检测到的失败和 action center 相

应的功能。执行操作中心功能并进行干预的新安全模块应监控检测故障。

其中DreamView是一个web应用程序（见下图），提供如下的功能：

- 可视化显示当前自动驾驶车辆模块的输出信息，例如规划路径、车辆定位、车架信息等。
- 为使用者提供人机交互接口以监测车辆硬件状态，对模块进行开关操作，启动自动驾驶车辆等。
- 提供调试工具，例如 PnC 监视器可以高效的跟踪模块输出的问题。

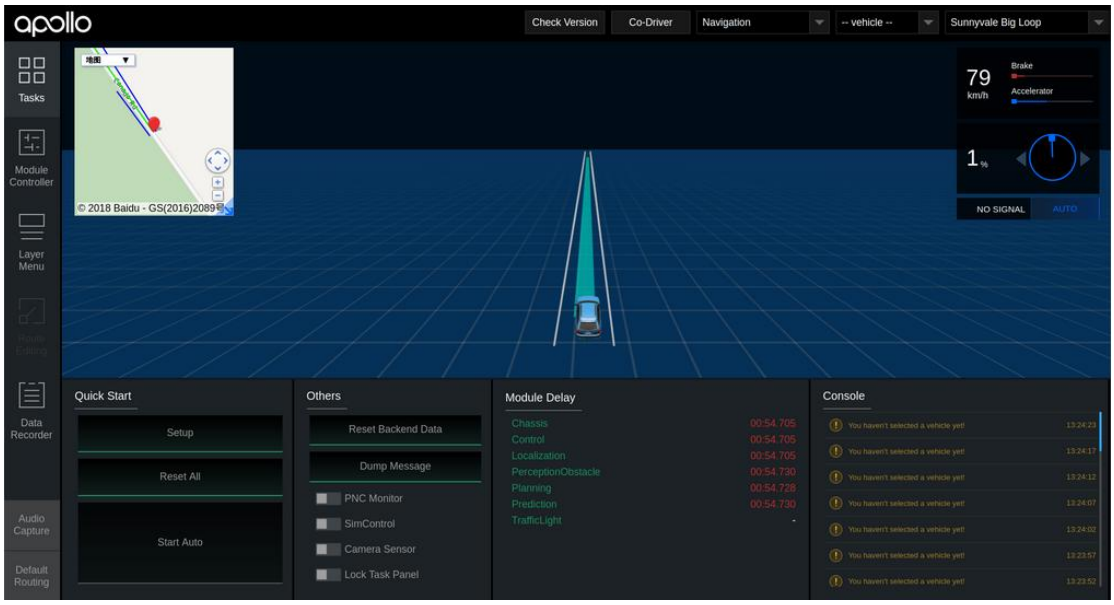


图1.5 Apollo的Dreamview工具显示当前车辆信息

应用情况

中小企业可以利用百度Apollo自动驾驶平台大幅提升自动驾驶技术创新的能力和速度。例如自动驾驶创业公司新石器，2018年7月启动建设全球首个L4级无人车工厂，历时11个月竣工，2019年5月正式启动量产，年产能达10000台。这样的发展速度得益于平台效应，新石器与Apollo平台的创新研发共建的合作模式，快速实现产品化和规模化运营，打通了新型自动驾驶商业模式。在2020年抗击新冠肺炎疫情的工作当中，百度与新石器、智行者等生态合作伙伴一起，在北京、武汉、青岛、深圳等17个城市投放了104台无人驾驶车辆，用于科技抗疫，发挥了重要作用。

为帮助生态伙伴更容易的使用Apollo，快速提升自动驾驶方案落地能力。Apollo开放平台还推出了开发套件产品（D-KIT）。其一站式的解决了用户使用

Apollo所面对的车辆、硬件、软件的诸多问题。其更进一步的降低了生态伙伴使用Apollo的门槛。具体有如下几点：

- 降低了车辆门槛：Apollo 开发套件提供了多款适用于封闭园区低速场景的纯电底盘，车身尺寸较小，面向低速场景，在封闭园区就可以测试，且其底盘通信协议已经适配 Apollo。
- 降低了硬件门槛：提供了和低速场景相关的激光雷达、摄像头、组合惯导设备，其数据通信协议已经适配 Apollo；
- 降低了集成门槛：提供了集成结构件、线束和显示屏幕，极大方便了设备集成与开发调试。
- 降低了采购门槛：提供整体的供应链服务，缩短了采购周期和成本。
- 降低了软件门槛：提供了循迹自动驾驶、低速园区自主避障两个场景的软件适配工具及文档，帮助开发者上手 Apollo。
- 降低了研发门槛：配备了研发流程相关的数据流水线云服务，包括动力学模型标定、控制规划仿真等云服务能力。

图1.6 Apollo开发套件

Apollo开发套件的推出极大加速了自动驾驶开发者创新能力和速度，其客户遍布高校、科研机构及自动驾驶服务提供商等。针对不同类别客户的需求差异，与开发套件配套的自动驾驶二次开发方案、演示搭建方案、人才培养方案已经落地几百家客户，为其科研、教学、产品落地提供了有力的支持。

智能家居生态系统

Kaldi 语音识别与 AIoT 智能家居生态系统

张铎 小米开源委员会主席

仇浩文 新一代 Kaldi 主要开发者

新一代 Kaldi

说起 Kaldi 过去的这一年，那就不得不提两个重要的日子：2019 年的 12 月跟 2020 年的 11 月。前者是 Kaldi 的作者 Daniel Povey 博士受小米副总裁、技术委员会主席崔宝秋博士的邀请，加入小米的日子。后者是新一代 Kaldi 0.1 版本在 2020 年小米开发者大会上正式发布的日子。

从 2010 年以来，Kaldi 以其高效的 C++代码实现，紧跟学术界最新研究趋势的模型更新，以及基于大量公开数据集的示例脚本，逐渐取代 HTK、CMU-Sphinx 等老牌开源语音识别库，成为语音识别领域最有影响力的开源工具包。截止目前，Kaldi 项目累计获得了 4000 多次的论文引用和 10,000 多个 GitHub Stars。最为重要的是，在工业界，几乎大部分成熟的语音产品背后，都离不开 Kaldi 的支持。

相比于 HTK、CMU-Sphinx 等传统的开源语音识别库，Kaldi 始终保持着与时俱进的传统，提出了 TDNN、Chain 等多款至今仍然保持头部竞争力的模型结构，在诸多语音识别公开数据集上都获得了 state-of-the-art 的结果（如在 LibriSpeech 数据集上，基于 Kaldi 训练的语音识别模型，其词错误率仅为 5%，在 2019 年以前，这一直是该数据集上最好的结果）。不仅如此，相比于这些传统语音识别库，Kaldi 还提供了众多数据集的示例脚本，用户可以很容易的基于这些脚本，来搭建自己的语音识别模型，这大大降低了语音识别领域的入门门槛。此外，Kaldi 的核心代码，全部使用 C++实现，而且在诸多方面进行了优化，如支持流式解码等，这使得 Kaldi 在训练和解码速度上，显著的优于其他的语音识别库（如 Kaldi 中基于 GPU 的解码速度是实时的 300 倍），也是 Kaldi 获得工业界青睐的主要原因之一。

最近一两年，随着端到端语音识别的流行，Kaldi 的地位也受到了一些诸如 ESPnet、Espresso 等端到端语音识别库的挑战。但由于 Kaldi 高效的 C++代码实现，海量的基于不同数据集的示例脚本，至今它依然是大家最为信赖和依赖的开源语音识别库。在开源语音识别领域，仍然有着超过 70%的占有率。很多新一代的语音识别库，如 ESPnet、Espresso、Torchaudio 等，都支持 Kaldi 的数

据处理和特征存储格式，Kaldi 的影响力可见一斑。而随着 Daniel Povey 博士来到中国，于去年 11 月下旬在小米科技园举行的 Kaldi 技术交流盛会，更是 Kaldi 巨大影响力的有力见证。

来到小米这一年，Daniel Povey 博士的主要工作就是带领团队开发新一代 Kaldi。现有端到端语音识别模型的流行和准确率的逐步提升，以及 PyTorch、TensorFlow 等易用的深度学习工具包的普及，是开发新一代 Kaldi 的主要动力。但 Kaldi 的目标不仅仅是赶上或者稍微领先这些新一代的语音识别库，而是要彻底地改变人们实现语音识别的方式。

新一代 Kaldi 分为三个部分：lhotse、icefall 和 k2。其中，lhotse 是训练数据准备部分，目前已发布 0.4 版本。不同于上一代 Kaldi 大量使用了 shell 脚本，lhotse 全部使用 Python 写就，方便易用。lhotse 设计了通用又不失去灵活性的接口，以适应包括语音识别，文本转语音等多种语音任务。用户更可以基于 lhotse，来方便地实现针对自己特定任务的接口，来操纵各种不同的音频元数据和文本。lhotse 引入了 Audio Cuts 的概念，将训练数据自动地组织为一组组 Cuts，并基于这种表示，提供了 on-the-fly 的数据混合，裁剪，增强和特征提取等操作，从而在不影响数据处理效率的前提下，降低了数据存储所需空间。此外，lhotse 还提供了很多公开数据集的数据处理脚本，用户可以直接使用这些脚本，来进行相关语音任务的数据处理工作，这大大降低了用户在某个数据集上进行实验的前期成本。

icefall 是训练脚本集合，目前已发布草案项目 snowfall，并完成了基于 LibriSpeech 数据集的训练和解码示例。用过 Kaldi 的人都知道，Kaldi 里有大量的基于不同数据集的示例脚本，这大大降低了用户的学习成本。但这同时也带来一个缺点，就是示例脚本集合太过庞大，代码耦合过于紧密，维护成本较高。考虑到这一点，icefall 将不再是一个大的脚本集合，而是会将不同数据集的示例脚本独立组织，用户从而可以单独下载特定数据集的脚本，来进行学习和使用。此外，由于将数据准备部分单独放在 lhotse 项目中，核心计算部分单独放在 k2 中，icefall 项目只需要关注语音识别模型的结构定义部分，这大大降低了整个语音识别过程的耦合性，也方便了网络结构的复用。

k2 是新一代 Kaldi 的核心，目前已发布 0.3 版本。k2 的核心贡献在于，将加权有限状态转换器(Weighted Finite State Transducers, WFST)和相关算法无缝地集成到基于 Autograd 的机器学习工具包，如 PyTorch（已完成支持）和 TensorFlow 中。WFST 是语音识别领域最为核心的数据结构，可以用来构建诸如“音标->词->句子”的状态转换概率图。支持 WFST 可导意味着我们可以做很多以前很难做到，甚至做不到的事情，如消除以往语音识别任务中训练跟解码过程不匹配的问题、多轮（可求导）的语音识别过程、在声学网络中嵌入任意辅助信息等。k2 也可以用来很方便地实现很多现有的语音识别模型，如 CTC、LF-

MMI、RNN-T 等，相关示例都已经发布在 icefall 的草案项目 snowfall 里。值得一提的是，Facebook 在去年发布了类似的项目 gtn，它跟 k2 几乎是同时相互独立地开始开发的。但相比于 gtn，k2 不仅实现了更多的语音识别相关算法，而且 k2 还高效地支持 GPU（gtn 目前只支持 CPU)。高效的 GPU 实现，使得快速的语音识别训练迭代成为可能。

新一代 Kaldi 之所以将整个项目分为三个部分，一方面是为了降低耦合性，方便用户使用。更重要的是，lhotse 作为数据准备部分，不仅可以用在 icefall 项目里，也可以用在任意其他语音识别库里来处理音频和文本数据。相信在不久的将来，随着新一代 Kaldi 的推广和普及，lhotse 甚至有可能成为语音领域使用最为广泛的数据准备工具。而 k2 作为核心部件，不仅可以用来做语音识别，也可以用来做手写文字识别等其他任务。

Kaldi 在小米智能家居生态中的应用

小爱同学，作为小米“手机×AIoT”战略布局中的一环，承担着小米 AIoT 生态中极为重要的角色。作为一款智能生活助手，通过它，用户可以连接到各种各样的 AIoT 设备并与它们产生互动:智能音箱、手机、电视、智能手表、儿童故事机、车载后视镜等。而通过与空气净化器、扫地机器人、电饭煲、台灯、空调等上亿智能家电的连接，小爱同学更可以帮助每个用户打造属于他们自己的整体智能家居体验。借助小爱同学，用户可以通过语音命令小爱音箱播放音乐，可以控制扫地机器人扫地，可以给小孩讲故事……。小爱同学甚至可以主动地学习你的生活模式，在你回家前帮你提前打开空调，在你进门的时候自动打开客厅的灯，在你要睡觉的时候拉上窗帘……成为你最贴心的的生活助手。

自 2017 年上线至今，小爱同学累计唤醒次数 726 亿，累计激活设备 2.51 亿台，月活用户数 达到 7840 万。 而这一切的背后，都离不开小米语音团队依托于 Kaldi 之上打造的，适用于各种场景的不同语音模型，如远近场语音唤醒、离在线语音识别、说话人识别等通用模型，以及口语评测、语种识别、语音情绪识别等适用于具体场景的特定模型。

Kaldi 在 AIoT 语音相关领域的应用是多方面的：声学模型方面，TDNN、Chain 模型被广泛使用在语音识别相关模型里， X-vector 更是说话人识别领域的标杆；而在工程方面，由于 Kaldi 高质量的 C++代码实现，工程师们可以很容易地基于 Kaldi 的代码来搭建生产环境。 其高效的 GPU 实时解码，更使得基于它的线上流式解码也变得相当容易；不仅如此，由于 Kaldi 中海量的示例脚本，工程师们可以很容易地基于自己公司产品的数据集进行修改，进而快速地搭建线上数据反馈和模型自动迭代更新的流程，大大缩短了模型更新的周期。

正是由于 Kaldi 的上述优点，小米语音团队才可以用一个很小的团队来支持公司的海量语音业务，伴随着小米 AIoT 产品线的扩展，开发出一个又一个的酷炫产品，如 MIUI 声控拍照、千人千面的内容点播、跨设备的声纹追剧、基于童音识别的内容限制等功能，大大方便了普通用户和家庭的生活。随着下一代 Kaldi 的发布和其在小米产品线的逐步落地，相信在不久的将来，小米将和其他公司一起，为普通用户带来更加完善的 AIoT 产品体验，这将是围绕 Kaldi 的“四赢”局面：Kaldi 项目赢，小米语音赢，全球的 Kaldi 社区赢，所有跟 Kaldi 相关的中小型公司赢！

人工智能医疗影像

腾讯在医疗影像人工智能领域的应用与创新

腾讯医疗健康（深圳）有限公司

2017 年，腾讯觅影启动建设医疗影像人工智能开放开源创新平台，平台拥有 50 多位博士学位的 AI 算法专家研究员，400 多位 AI 应用工程师以及多位医学领域专家及资深产品经理；并与施普林格自然集团（Springer Nature）进行战略合作，共同推进“AI+医疗”领域跨学科研究，邀请多位顶级 AI 与医学专家担任学术顾问，与众多三甲医院建立 AI 医学实验室，承接多项高级医疗系统人工智能研究课题，与国际医学出版社 AME 联合出版 AI 医学研究学术期刊 TMAI。

在新冠肺炎疫情爆发和流行期间，腾讯觅影人工智能专家与医务专家、医务人员赴武汉驰援抗疫一线，开展新冠肺炎 CT 快速检查，辅助诊断、远程诊断，并研究快准检查、诊断方案。他们在武汉大学中南医院实现患者 CT 检查，最快 2 秒完成人工智能模式识别，1 分钟内为医生提供辅助诊断参考，助力中南医院医学影像科在两个月内累计为 24000 多名患者进行肺部诊断工作。在武汉日海方舱医院、武汉协和西院和洪湖市人民医院等医院完成部署，利用 AI 与远程诊断技术，为方舱医院新冠肺炎患者及时地进行 CT 检查，

腾讯自起动建设医疗影像人工智能开放开源创新平台以来，开展对肺炎、肺结节、新冠肺炎、肺癌以及乳腺癌、结直肠癌、、宫颈癌、眼底视网膜病变等病种建立辅助筛查系统，帮助临床医生提高医学影像识读效率。

腾讯加强对医疗影像 AI 的理论研究，提高医疗影像检查诊断的精准度和效率。腾讯加大对平台的算法、计算机硬件、医疗数据标注方面的投入，保证了平台运行所需的算法、算力和数据的三大核心需求。腾讯觅影开放实验室与腾讯优图实验室、天衍实验室、AILab 协同，汇集大量常用算法，包括 CNN、RNN、DBN 等深度学习算法，以及 GBDT、FFM 等传统机器学习算法，如面向医学影像的模型算法 X2CT-GAN，MedicalNet 等，用户可以通过平台集成的算法自动训练自己的模型。腾讯觅影在医疗 AI 领域迄今累计申请专利已经超过 300 件（分布于医疗影像医疗辅诊、病案管理、药品管理、风险监控等方面）。

腾讯觅影平台进行大量的基础设施建设，建设有数亿元的 GPU 计算机集群，与上百家医院开展科研合作，建立了成熟的数据库、数据标注平台和管控体系。约 50 万医学术语库、超 100 万的术语关系规则库、超 1000 万的健康知识库和超 8000 万的高质量医疗知识库，可实现 700 多种疾病预测，覆盖超过 90%的医院门诊高频诊断，与众多三甲医院建立 AI 医学实验室。

2017 年 8 月推出首款医疗 AI 产品——“腾讯觅影”，打造的医学影像 AI 辅助诊断产品，已与 120 多家医院开展多方向、多模态、多尺度的科研合作和试点应用。腾讯将 AI 辅助诊疗产品的辅诊能力作为智能工具箱开放给医院和信息化厂商，助力医院 HIS 系统、互联网医疗服务实现智能化，升级构建覆盖诊前、诊中和诊后的智慧医疗形态。

腾讯觅影 AI 影像产品，如眼底疾病 AI 筛查系统、结直肠 AI 辅助诊断系统，与对应的影像采集设备相结合，能更好地服务用户。例如，眼底照相 AI 筛查系统，已经从糖网分期扩展为全眼底 30 多种疾病的筛查。该系统通过与眼底照相机深度融合，可以从传统的眼科应用场景扩展到体检科、产科、内分泌科和高血压门诊等筛查场景，同时还能提高眼底检查的效率。目前，腾讯觅影眼底智能筛查系统已与全国 30 多家医院联合进行验证，并在广东、山东、四川和广西等多个省市基层医疗卫生机构落地试点，已为超过 6.5 万人提供了筛查服务。

当前在智能医疗领域，数据的标注使用、模型效果评估还缺乏伦理和标准体系指导。为此，腾讯积极发挥自身实力，努力补齐行业短板。依托平台，腾讯与国际标准化组织（ITU-4H 焦点组）合作，积极参与国际医疗 AI 标准建设，已经提交了两项提案，分别是《消化内镜数据标注于质控流程标准》和《基于医学影像大数据量安全标注的标准》；与中国医学装备协会合作，建立了《基于眼底照相的人工智能青光眼辅助筛查系统规范化设计及应用指南》。

自主操作系统和自建生态

66 款自主操作系统和自建生态

（含 6 款自主 CPU/SOC 芯片）阶段成果

宋可为、陈渝 整理

（一）服务器操作系统

	厂商	名称	内核特征	上游系统	BIOS	Architecture特征	Framework特征	开源与否	通用/专用/定制	硬件生态（DPI）	应用生态（API）	设计/研发/发布/鉴定/量产
服务器操作系统	华为	OpenEular	宏内核	CentOS	UEFI/BIOS	ARM/x86-64/RISC-V	CLI/GUI	开源	通用	X86/鲲鹏	受Linux自身应用生态限制	鉴定
	阿里	Anolis OS	宏内核	CentOS	UEFI/BIOS	ARM/x86-64	CLI/GUI	开源	通用	X86/龙芯/飞腾	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制	发布
	浪潮	SSR	宏内核	CentOS	UEFI/BIOS	x86-64	CLI/GUI	不开源	浪潮服务器专用	X86/龙芯/飞腾/兆芯/海光	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	麒麟	Kylin	宏内核	CentOS	UEFI/BIOS	86-64/ARM/MIPS/龙芯/兆芯/海光/鲲鹏/申威/飞腾	CLI/GUI	不开源	通用	龙芯/飞腾/兆芯/海光/鲲鹏	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	统信/深度	UOS/Deepin	宏内核	CentOS	UEFI/BIOS	x86-64/ARM/MIPS/龙芯/兆芯/海光/鲲鹏/申威/飞腾	CLI/GUI	不开源	通用	龙芯/飞腾/申威/兆芯/海光/鲲鹏	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	普华	isoft	宏内核	CentOS	UEFI/BIOS	x86-64/PowerPC/MIPS/Alpha/龙芯/兆芯/鲲鹏/申威/	CLI/GUI	不开源	通用	龙芯/兆芯/申威/海光/鲲鹏	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制	量产
	中科方德		宏内核	CentOS	UEFI/BIOS	x86-64/ARM/MIPS/龙芯/兆芯//飞腾	CLI/GUI	不开源	通用	兆芯/海光	信创国产替代生态较一般	量产
	拓林斯	Turbolinux	宏内核	CentOS	UEFI/BIOS	ARM/x86-64	CLI/GUI	不开源	通用	兆芯/海光/鲲鹏	信创国产替代生态弱；受制于Linux应用生态限制	量产
	凝思	磐石Linux	宏内核	CentOS	UEFI/BIOS	x86-64	CLI/GUI	不开源	通用	支持X86/PowerPC/Mips/ARM64	信创国产替代生态弱；受制于Linux自身应用生态限制	量产
	万里红	supperred	宏内核	CentOS	UEFI/BIOS	ARM/x86-64	CLI/GUI	不开源	通用	龙芯/飞腾/兆芯/海光/鲲鹏	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制	量产
	一铭	emindlinux	宏内核	CentOS	UEFI/BIOS	x86-64	CLI/GUI	不开源	通用	龙芯/飞腾/兆芯/海光	信创国产替代生态较弱；受制于国产指令集和Linux自身应用生态限制	量产
	红旗	Redflaglinux	宏内核	CentOS	UEFI/BIOS	x86-64/ARM/MIPS/龙芯/兆芯/海光/鲲鹏/申威/飞腾	CLI/GUI	不开源	通用	龙芯/兆芯/海光/	信创国产替代生态较弱；受Linux自身应用生态限制	量产
	新支点		宏内核	CGL	UEFI/BIOS	x86-64/ARM/PowerPC/MIPS/龙芯/兆芯/鲲鹏/申威/飞腾	CLI/GUI	不开源	通用	龙芯/兆芯/鲲鹏	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制	量产
	思普	SPLinux	宏内核	CentOS	UEFI/BIOS	x86-64/ARM/PowerPC/兆芯/鲲鹏/飞腾	CLI/GUI	不开源	通用	X86	信创国产替代生态较弱；受Linux自身应用生态限制	量产
	龙芯	loongnix	宏内核	CentOS	UEFI/BIOS	MIPS/龙芯	CLI/GUI	开源	龙芯专用基线系统	龙芯	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	航天国盛	绿地	宏内核	CGL	UEFI/BIOS	x86-64/ARM/PowerPC/兆芯/鲲鹏/飞腾	CLI/GUI	不开源	通用、军队专用	龙芯/飞腾/兆芯/海光/鲲鹏;ARM	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制;终端双板系列兼容安卓应用生态和QT生态	量产
	谢宝友	dim-sum	宏内核	N/A	UEFI/BIOS	x86-64	CLI	开源	专用	X86	暂无	研发
	本源司南	origin pilot	N/A	N/A	N/A	N/A	N/A	N/A	量子计算机专用	N/A	N/A	研发

（二）桌面操作系统和云计算操作系统

	厂商	名称	内核特征	上游系统	BIOS	Architecture特征	Framework特征	开源与否	通用/专用/定制	硬件生态（DPI）	应用生态（API）	设计/研发/发布/鉴定/量产
桌面操作系统	麒麟	Kylin	宏内核	Ubuntu	UEFI/BIOS	86-64/ARM/MIPS/龙芯/兆芯/海光/鲲鹏/申威/飞腾	GUI	商业版本不开源/有开源版本	通用	龙芯/飞腾/兆芯/海光/鲲鹏	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	统信/深度	uos/deepin	宏内核	Ubuntu	UEFI/BIOS	x86-64/ARM/MIPS/龙芯/兆芯/海光/鲲鹏/申威/飞腾	GUI	商业版本不开源/有开源版本	通用	龙芯/飞腾/申威/兆芯/海光/鲲鹏	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	中科方德		宏内核	Fedora	UEFI/BIOS	x86-64/ARM/MIPS/龙芯/兆芯//飞腾	GUI	部分开源	通用	兆芯/海光	信创国产替代生态较一般	量产
	万里红	supperred	宏内核	Fedora	UEFI/BIOS	ARM/x86-64	GUI	部分开源	通用	龙芯/飞腾/兆芯/海光/鲲鹏	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制	量产
	一铭	emindlinux	宏内核	Ubuntu	UEFI/BIOS	x86-64	GUI	部分开源	通用	龙芯/飞腾/兆芯/海光	信创国产替代生态较弱；受制于国产指令集和Linux自身应用生态限制	量产
	新支点	GD-linux	宏内核	CGL	UEFI/BIOS	x86-64/ARM/PowerPC/MIPS /龙芯/兆芯/鲲鹏/申威/飞腾	GUI	部分开源	通用	龙芯/兆芯/鲲鹏	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制	量产
	思普	SPLinux	宏内核	Ubuntu	UEFI/BIOS	x86-64/ARM/PowerPC/兆芯/鲲鹏/飞腾	GUI	部分开源	通用	X86	信创国产替代生态较弱；受Linux自身应用生态限制	量产
	广东爱瓦力	StartOS	宏内核	Ubuntu	UEFI/BIOS	x86-64	GUI	部分开源	通用	X86	受Linux自身应用生态限制	量产
	龙芯	loongnix	宏内核	Fedora	UEFI/BIOS	MIPS/龙芯	GUI	部分开源	龙芯专用基线系统	龙芯	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制	量产
	航天国盛	绿地	宏内核	CGL	UEFI/BIOS	x86-64/ARM/PowerPC/兆芯/鲲鹏/飞腾	GUI	不开源	通用、军队专用	龙芯/飞腾/兆芯/海光/鲲鹏;ARM	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制;终端双板系列兼容定点应用生态和OT生态	量产
云计算操作系统	阿里	阿里云os	宏内核	CentOS	UEFI/BIOS	x86-64	CLI	开源	阿里云专用	X86	受Linux自身应用生态限制	量产
	腾讯	tencentOS server	宏内核	RHEL	UEFI/BIOS	x86-64	CLI	开源	腾讯专用	X86/ARM64/海光	受Linux自身应用生态限制	量产
	华云	archerOS	宏内核	CentOS	UEFI/BIOS	x86-64	CLI	不开源	华云专用	龙芯/飞腾/兆芯/海光/鲲鹏	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制。丰富的云服务	量产
	H3C	H3CloudOS	宏内核	CentOS	UEFI/BIOS	x86-64	CLI	不开源	H3C云专用	飞腾/兆芯/海光/鲲鹏	信创国产替代生态较强；受制于国产指令集和Linux自身应用生态限制。丰富的云服务	量产
	品高	BingoCloudOS	宏内核	CentOS	UEFI/BIOS	x86-64	CLI	不开源	品高云专用	龙芯/飞腾	信创国产替代生态一般；受制于国产指令集和Linux自身应用生态限制。丰富的云服务	量产
	浪潮云海云	InCloud OpenStack	宏内核	CentOS	UEFI/BIOS	x86-64	CLI	不开源	浪潮云专用	X86	受Linux自身应用生态限制	量产

（三）智能终端操作系统和物联网操作系统

	厂商	名称	内核特征	上游系统	BIOS	Architecture特征	Framework特征	开源与否	通用/专用/定制	硬件生态（DPI）	应用生态（API）	设计/研发/发布/鉴定/量产
智能终端操作系统	华为	鸿蒙	宏内核	liteOS/Linux	定制	arm	CLI/GUI	开源	华为1+8设备专用	ARM/海思	兼容安卓应用生态（自建应用生态）	研发
	元心	syberOS	宏内核	Meego/Linux+QT	定制	arm	CLI/GUI	不开源	手机专用	ARM/Atom	QT应用生态	鉴定
	上海联彤	COS	宏内核	AOSP	定制	arm	CLI/GUI	不开源	手机专用	ARM	自建应用生态	鉴定
	全智达	全智达TIOS	宏内核	Linux+QT	定制	arm	CLI/GUI	不开源	联通手机专用	ARM	自建应用生态	鉴定
	360	360	宏内核	AOSP	定制	arm	CLI/GUI	安全组件不开源	360手机专用	ARM	安卓应用生态	鉴定
	初心使命	CHOSEN	宏内核	OpenTHOS	UEFI/BIOS	x86-64	Android	商业版本不开源/有开源版本	飞腾终端定制	龙芯/飞腾/兆芯/ARM	兼容安卓应用生态	发布
	技德	JideOS	宏内核	Android X86	UEFI/BIOS	x86-64	Android	不开源	X86桌面专用	ARM/Atom	兼容安卓应用生态	发布
	凤凰	Phoenix	宏内核	Android X86	定制	arm	Android	不开源	X87桌面专用	ARM/Atom	兼容安卓应用生态	发布
	广电总局	TVOS	宏内核	Android	定制	arm	Android	不开源	TV专用	ARM	安卓应用生态	量产
	鲸鲮科技	JingOS	宏内核	Ubuntu	UEFI/BIOS	x86-64	CLI/GUI	不开源	平板专用	X86	集成Linux触控友好的小应用	研发
物联网操作系统	华为	鸿蒙	宏内核	liteOS	定制	arm/RISC-V	CLI/GUI	开源	华为1+8设备专用	ARM/海思	自建应用生态	研发
	华为	liteOS	宏内核	liteOS	定制	arm/RISC-V	CLI/GUI	开源	专用	ARM	自建应用生态	鉴定
	腾讯	tencetOS-tiny	宏内核	N/A	定制	arm	CLI/GUI	开源	专用	ARM	自建应用生态	量产
	阿里	aliOSThing	微内核	N/A	定制	arm	CLI/GUI	开源	专用	ARM	自建应用生态	量产
	小米	vela	宏内核	NuttX	定制	arm	CLI/GUI	开源	专用	ARM	自建应用生态	量产
	上海睿赛德	Rt-Thread	宏内核/微内核	N/A	定制	arm/RISC-V/mips/x86-64	CLI/GUI	开源	专用	ARM	自建应用生态	量产
	翼辉	edgeOS	宏内核	N/A	定制	arm/RISC-V/mips/x86-64/PowerPC	CLI/GUI	不开源	拟通用	ARM/飞腾	自建应用生态	研发
	上海合宙	luatOS	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	飞漫软件	HybridOS	宏内核/微内核	N/A	定制	arm	CLI/GUI	开源	专用	ARM	自建应用生态	量产
	小i机器人	iBot OS	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	光年无限	Turing OS	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	儒博科技	RooBo OS	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	中国移动	OneOS	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	庆科	MiCO IoT OS	宏内核	N/A	定制	arm	CLI	不开源	专用	ARM	自建应用生态	量产
	南潮信息	Ruff	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	H3C	绿洲Oasis	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产
	海尔	Uhome OS	宏内核	N/A	定制	arm	CLI/GUI	不开源	专用	ARM	自建应用生态	量产

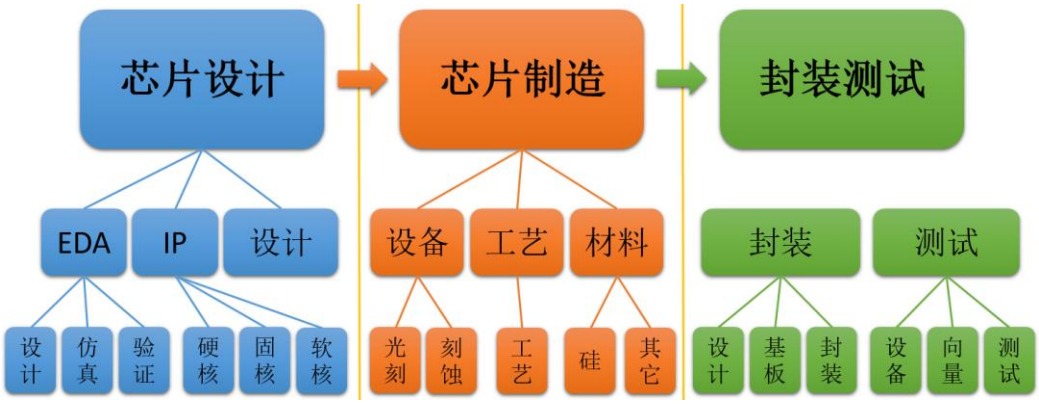
（四）嵌入式操作系统和物联网操作系统

	厂商	名称	内核特征	上游系统	BIOS	Architecture特征	Framework特征	开源与否	通用/专用/定制	硬件生态（DPI）	应用生态（API）	设计/研发/发布/鉴定/量产
嵌入式操作系统	中航631所	天脉CoreOS	微内核	N/A	定制	powerpc/arm/x86/mips	CLI	不开源	专用	powerpc	自建应用生态	量产
	中电科32所	锐华reworks	微内核	N/A	定制	powerpc/arm/x86/mips	CLI	不开源	专用	arm/powerpc	自建应用生态	量产
	——	东风	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	航天5院502所	SpaceOS	微内核	N/A	定制	powerpc/arm/x86/mips	CLI	不开源	专用	arm/powerpc	自建应用生态	鉴定
	翼辉	Sylix	宏内核	N/A	定制	powerpc/arm/x86/mips	CLI		专用		自建应用生态	量产
	科银京诚	DaoOS	微内核	N/A	定制	powerpc/arm/x86/mips	CLI	不开源	专用		自建应用生态	量产
	光轮电子	tree OS	宏内核	N/A	定制	MCS51/STC/AVR/MSP430/STM8/STM32	CLI	不开源	专用		自建应用生态	鉴定
	国讯芯微	NECRO QIUNIU	宏内核	N/A	定制	arm	CLI	不开源	专用		自建应用生态	鉴定
	——	战星	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	神州战旗	战旗	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	航天9院771所	骊山九天	宏内核	N/A	定制	powerpc/arm/x86/mips/dsp	CLI	不开源	专用		自建应用生态	鉴定
	航天2院706所	天熠	宏内核	N/A	定制	MIPS/龙芯	CLI/GUI	不开源	专用		自建应用生态	量产
	中船716	杰锐	微内核	N/A	定制	powerpc/arm/x86/mips	CLI	不开源	专用		自建应用生态	鉴定
	军事科学院	军科机器人	宏内核	N/A	定制	arm/x86-64	CLI	不开源	专用		自建应用生态	研发
	浙江大学	浙大无人OS	宏内核	N/A	定制	arm/x86-64	CLI	不开源	专用		自建应用生态	发布
	南京大学	南大无人OS	宏内核	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	中电科普华	orientais OS	微内核	N/A	定制	英飞凌、意法、恩智浦、瑞萨等国外微控制器芯片（MCU），紫光等国产微控制器芯片	CLI	不开源	专用		自建应用生态	鉴定
	都江堰	djyos	宏内核	N/A	定制	STM32/arm	CLI	开源	专用		自建应用生态	发布
		TeeOS	微内核	N/A	定制	ARM	CLI	不开源	可信安全专用		自建应用生态	量产

芯片自主可控深度解析

SiP 与先进封装技术 Suny Li

首先，什么叫自主可控，最直观的理解就是当别人“卡脖子”的时候不会被卡住。集成电路产业通常被分为**芯片设计、芯片制造、封装测试**三大领域，参看下图：



我们逐一进行分析，**芯片设计**主要从 EDA、IP、设计三个方面来分析；**芯片制造**主要从设备、工艺和材料三个方面来分析；**封装测试**则从封装设计、产品封装和芯片测试几方面来分析。

一、芯片设计

如何开始一款芯片设计呢？

首先要有工具（EDA），然后借助现有的资源（IP），加上自己的构思和规划就可以开始芯片设计了。这里，我们就从芯片设计工具 EDA，知识产权 IP，以及集成电路的设计流程来分析芯片设计。

1、EDA

EDA（Electronic Design Automation）电子设计自动化，常指代用于电子设计的软件。

曾经有人跟我说：“EDA 有啥呀，不就是个工具嘛？”是啊，确实就是个工具，可是没这个工具，你啥也设计不了啊！

现在的大规模集成电路在芝麻粒大小的 1 平方毫米内可以集成 1 亿只以上的晶体管，这些晶体管之间的连接网络更是多达数亿个。当今主流的 SoC

芯片，其晶体管数量已经超过百亿量级。如果没有精准的，功能强大的 EDA 工具，怎么设计呢？

EDA 是芯片设计的必备工具，目前，Synopsys、Cadence 和 Mentor（Siemens EDA）占据着超过 90%以上的市场份额。在 10 纳米以下的高端芯片设计上，其占有率甚至高达 100%。也就是说，现在研发一款 10nm 以下的芯片，没有以上三家的 EDA 工具几乎是不可能实现的。

下表所示是目前芯片设计中主流的 EDA 工具：

集成电路设计类型	设计阶段	对应的 EDA 工具
数字前端设计	RTL 仿真	Synopsys 的 VSC, Mentor 的 Modelsim, Questa
	综合	Synopsys 的 Design Complier Cadence 的 Genus
数字后端设计	IC 版图设计	Synopsys 的 ICC Cadence 的 EDI/Innovus
DFT 可测试性设计	BSCAN	Mentor 的 BSDArchit Synopsys 的 BSD Complier
	MBIST	Mentor 的 MBITarchitect, Tessentmbist
	ATPG	Mentor 的 TestKompress, Synopsys 的 Tetra
	MAX Scan chain	Synopsys 的 DFT Complier
Signoff 设计审签	Timing 时序仿真	Synopsys 的 PT 占主导地位, Cadence tempus 也有部分份额
	Physical 物理验证	Mentor 的 Calibre 占主导地位, Synopsys 的 ICV, Cadence 的 PVS 也占小部分份额
模拟电路设计	模拟电路图及版图	Cadence Virtuoso 目前使用最普遍

芯片设计分为设计、仿真、验证等环节，对应的 EDA 工具分为设计工具、仿真工具、验证工具等。设计工具解决的是模型的构建，也就是从 0 到 1（从无到有）的问题，仿真和验证工具解决模型的确认，也就是 1 是 1 还是 0.9 或者 1.1 的问题。因此，从 EDA 开发的角度，设计工具的开发难度更大。此外，设计规模越大，工艺节点要求越高，EDA 工具的开发难度也越大。国产 EDA 工具目前在一些仿真验证点工具上取得一些成绩，在模拟电路设计方面也初步具备了全流程工具，但在大规模集成电路设计上和三大厂商还有很大的差距，尤其在高端数字芯片设计流程上基本还是空白。

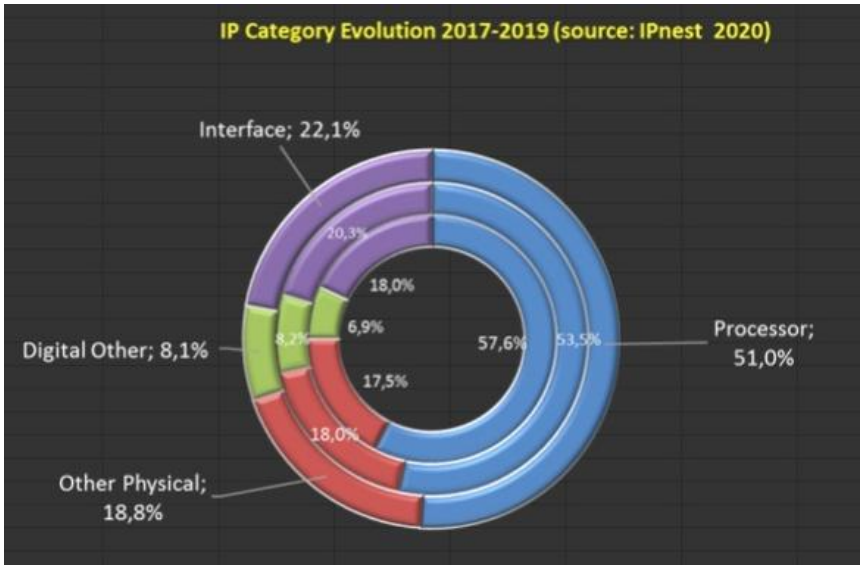
2、IP

IP（Intelligent Property）代表着知识产权的意思，在业界是指一种事先定义、经过验证的、可以重复使用，能完成特定功能的模块，IP 是构成大规模集成电路的基础单元，SoC 甚至可以说是基于 IP 核的复用技术。IP 一般分为硬核、软核和固核。IP 硬核一般已经映射到特定工艺，经过芯片制造验证，具有面积和性能可预测的特点，但灵活性较小；IP 软核以 HDL 形式提交，灵活性强，但性能方面具有不可预测性；IP 固核通过布局布线或利用通用工艺库，对性能和面积进行了优化，比硬核灵活，比软核在性能和面积上更可预测，是硬核和软核的折中。

下表为目前全球前 10 大 IP 提供商，可以看到中国有两家入围前十，但是两家市场份额加起来也仅有 3%，而 ARM 一家就占据了 40%以上的市场份额，美国的企业则占据了 30%的市场份额，如果 ARM 被英伟达收购，基本上 IP 市场就是美国的天下了。此外我们也发现，全球最大的两家 EDA 公司 Synopsys 和 Cadence，在 IP 领域也同样占据的第二、第三的位置。

排名	厂商	营业额(百万美元)	市场份额	国家地区
1	ARM	1608.0	40.8%	英国
2	Synopsys	716.9	18.2%	美国
3	Cadence	232.0	5.9%	美国
4	SST	115.0	2.9%	美国
5	Imagination Technologies	101.1	2.6%	英国
6	Ceva	87.2	2.2%	以色列
7	Verisilicon	69.8	1.8%	中国
8	Achronix	50.0	1.3%	美国
9	Rambus	49.9	1.2%	美国
10	eMemory Technology	46.8	1.2%	中国台湾
	Top 10 Vendors	3075.6	78.1	中国 3%

下图所示为 IP 的种类，其中处理器占 51%，接口 IP 占 22.1%，数字类占 8.1%，其他占 18.8%，处理器类 ARM 一家独大，在接口类 IP 中，Synopsys 是业界领导者。



我们需要考虑的是，在设计的芯片中那些 IP 是自主设计的，那些是外购的，这些外购的 IP 是否存在不可控因素？如果你设计的 SoC 仅仅是把别人的 IP 打包整合，那自主可控性就要大打折扣了。

下面，我们以华为麒麟 980 为例，了解一下芯片研发中的 IP 使用情况。

麒麟 980 芯片集成的主要部件有 CPU、GPU（俗称显卡）、ISP（处理拍照数据）、NPU（人工智能引擎）和基带（负责通信）。

根据华为官方资料，ISP 是华为自研，NPU 是华为和寒武纪合作的成果，至于 CPU（Cortex-A76）和 GPU（Mali-G76）则是华为向 ARM 公司购买的授权，包括指令集授权和内核授权。

如果没有 IP 授权，还有没有可能自研麒麟 980 芯片，目前看来，没有 。

3、设计流程

芯片设计流程通常可分为：数字 IC 设计流程和模拟 IC 设计流程。

数字 IC 设计流程：芯片定义 → 逻辑设计 → 逻辑综合 → 物理设计 → 物理验证 → 版图交付。

芯片定义（Specification)是指根据需求制定芯片的功能和性能指标，完成设计规格文档。

逻辑设计(Logic Design)是指基于硬件描述语言在 RTL(Register-Transfer Level)级实现逻辑设计，并通过逻辑验证或者形式验证等验证功能正确。

逻辑综合(Logic Synthesis)是指将 RTL 转换成特定目标的门级网表，并优化网表延时、面积和功耗。

物理设计(Physical Design)是指将门级网表根据约束布局、布线并最终生成版图的过程，其中又包含：数据导入 → 布局规划 → 单元布局 → 时钟树综合 → 布线。

- **数据导入**是指导入综合后的网表和时序约束的脚本文件，以及代工厂提供的库文件。
- **布局规划**是指在芯片上规划输入/输出单元，宏单元及其他主要模块位置的过程。
- **单元布局**是根据网表和时序约束自动放置标准单元的过程。
- **时钟树综合**是指插入时钟缓冲器，生成时钟网络，最小化时钟延迟和偏差的过程。
- **布线**是指在满足布线层数限制，线宽、线间距等约束条件下，根据电路关系自动连接各个单元的过程。

物理验证(Physical Verificaiton)通常包括版图设计规则检查（DRC），版图原理图一致性检查（LVS）和电气规则检查（ERC）等。

版图交付(Tape Out)是在所有检查和验证都正确无误的前提下，传递版图文件给代工厂生成掩膜图形，并生产芯片。

模拟 IC 设计流程：芯片定义 → 电路设计 → 版图设计 → 版图验证 → 版图交付。

其中芯片定义和版图交付和数字电路相同，模拟 IC 在电路设计、版图设计、版图验证和数字电路有所不同。

模拟电路设计是指根据系统需求，设计晶体管级的模拟电路结构，并采用 SPICE 等仿真工具验证电路的功能和性能。

模拟版图设计是按照设计规则，绘制电路图对应的版图几何图形，并仿真版图的功能和性能。

模拟版图验证是验证版图的工艺规则、电气规则以及版图电路图一致性检查等。

这里，我们做一个简单的总结：

芯片设计：就是在 EDA 工具的支持下，通过购买 IP 授权+自主研发（合作开发）的 IP，并遵循严格的集成电路设计仿真验证流程，完成芯片设计的整个过程。在这个过程中，EDA、IP、严格的设计流程三者缺一不可。

目前看来，在这三要素中最先可能实现自主可控的就是设计流程了。

下表列出了当前世界前 10 的芯片设计公司，供大家参考。

排名	厂商	营收(2019)	营收(2018)	国家地区
1	博通（Broadcom）	17,246	18,547	美国
2	高通（Qualcomm）	14,518	16,370	美国
3	英伟达（NVIDIA）	10,125	11,163	美国
4	联发科（Media Tek）	7,962	7,882	中国台湾
5	超威（AMD）	6,731	6,475	美国
6	赛灵思（Xilinx）	3,236	2,868	美国
7	美满（Marvell）	2,708	2,823	美国
8	联咏科技（Novatek）	2,085	1,813	中国台湾
9	瑞昱半导体（Realtek）	1,965	1,518	中国台湾
10	戴乐格（Dialog）	1,421	1,442	英国
	Top 10 Vendors	67,997	70,901	

二、芯片制造

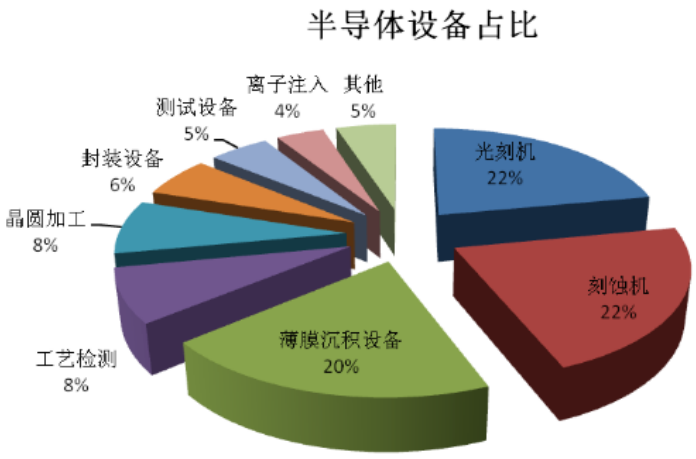
芯片制造目前是集成电路产业门槛最高的行业，怎么看待门槛的高低呢，投资越高、玩家越少就表明门槛越高，目前在高端芯片的制造上也仅剩下台积电（TSMC）、三星（SAMSUNG）和英特尔（Intel）三家了。下面，我们分别从设备、工艺和材料三个方面来分析芯片制造，寻找我们和先进制造技术的差距。

1、设备

芯片制造需要经过两千多道工艺制程才能完成，每个步骤都要依赖特定设备才能实现。

芯片制造中，有三大关键工序：光刻、刻蚀、沉积。三大工序在生产过程中不断重复循环，最终制造出合格的芯片。

三大关键工序要用到三种关键设备，分别是光刻机、刻蚀机、薄膜沉积设备。三大设备占所有设备投入的 22%、22%、20%左右，是三种占比最高的半导体设备。



下面就以最为典型的光刻机和刻蚀机为例进行介绍并分析自主可控。

- **光刻机**

光刻机的原理其实像幻灯机一样，就是把光通过带电路图的掩膜(也叫光罩)Mask 投影到涂有光刻胶的晶圆上。60 年代末，日本尼康和佳能开始进入这个领域，当时的光刻机并不比照相机复杂多少。

为了实现摩尔定律，光刻技术需要每两年把曝光关键尺寸(CD)降低 30%-50%。需要不断降低光刻机的波长 λ。然而，波长被卡在 193nm 无法进步长达 20 年。后来通过工程上最简单的方法解决，在晶圆光刻胶上方加 1mm 厚的水，把 193nm 的波长折射成 134nm，称为浸入式光刻。浸入式光刻成功翻越了 157nm 大关，加上后来不断改进的镜头、多光罩、Pitch-split、波段灵敏光刻胶等技术，浸入式 193nm 光刻机一直可以做到今天的 7nm 芯片（苹果 A12 和华为麒麟 980）。EVU 光刻机 EUV 极紫外光刻（Extreme Ultra-Violet）是一种使用极紫外（EUV）波长的新一代光刻技术，其波长为 13.5 纳米。由于光刻精度是几纳米，EUV 对光的集中度要求极高，相当于拿个手电照到月球光斑不超过一枚硬币。反射的镜子要求长 30cm 起伏不到 0.3nm，相当于北京到上海的铁轨起伏不超过 1 毫米。一台 EUV 光刻机重达 180 吨，超过 10 万个零件，需要 40 个集装箱运输，安装调试要超过一年时间。2000 年时，日本尼康还是光刻机领域的老大，到了 2009 年 ASML 已经遥遥领先，市场占有率近 7 成。目前，最先进的光刻机也只有 ASML 一家可以提供了。国内的情况，上海微电子（SMEE）已经有分辨率为 90nm 的光刻机，新的光刻机也在研制中。

型号	SSA600/20	SSC600/10	SSB600/10
分辨率	90nm	110nm	280nm
曝光光源	ArF excimer laser	KrF excimer laser	i-line mercury lamp
镜头倍率	1:4	1:4	1:4
硅片尺寸	200mm或300mm	200mm或300mm	200mm或300mm

在集成电路制造中，光刻只是其中的一个环节，另外还有无数先进科技用于前后道工艺中。

- **刻蚀机**

刻蚀是将晶圆表面不必要的材质去除的过程。刻蚀工艺位于光刻之后。

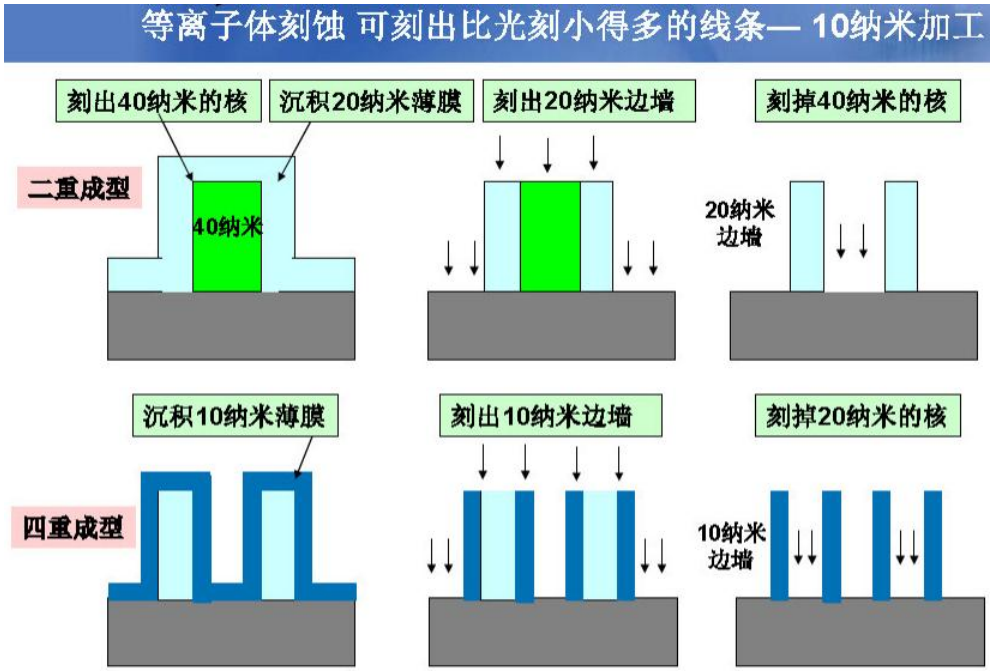
光刻机用光将掩膜上的电路结构复制到硅片上，刻蚀机把复制到硅片上的电路结构进行微雕，雕刻出沟槽和接触点，让线路能够放进去。

按照刻蚀工艺分为干法刻蚀以及湿法刻蚀，干法刻蚀主要利用反应气体与等离子体进行刻蚀，湿法刻蚀工艺主要是将刻蚀材料浸泡在腐蚀液内进行刻蚀。

干法刻蚀在半导体刻蚀中占据主流，市场占比达到 95%，其最大优势在于能够实现各向异性刻蚀，即刻蚀时可控制仅垂直方向的材料被刻蚀，而不影响横向材料，从而保证细小图形保真性。湿法刻蚀由于刻蚀方向的不可控性，在先进制程很容易降低线宽，甚至破坏线路本身，导致芯片品质变差。

目前普遍采用多重模板工艺原理，即通过多次沉积、刻蚀工艺实现需要的特征尺寸，例如 14nm 制程所需使用的刻蚀步骤达到 64 次，较 28nm 提升 60%；7nm 制程所需刻蚀步骤更是高达 140 次，较 14nm 提升 118%。

下图所示为多次刻蚀原理。



和光刻机一样，刻蚀机的厂商也相对较少，代表企业主要是美国的 Lam Research（泛林半导体）、AMAT（应用材料）、日本的 TEL（东京电子）等企业。这三家企业占据全球半导体刻蚀机的 94%的市场份额，而其他参与者合计仅占 6%。其中，Lam Research 占比高达 55%，为行业龙头，东京电子与应用材料分别占比 20%和 19%。

国内的情况，目前刻蚀设备代表公司为中微公司、北方华创等。中微公司较为领先，工艺节点已经达到 5nm。在全球前十大晶圆企业中，中微公司已经进入其中六家，作为台积电的合作伙伴协同验证 14nm/7nm/5nm 等先进工艺。

基于此，如果目前在光刻机领域我们还无力做出改变，那么已经有一定优势的刻蚀机势必会成为国产替代的先锋。

2、工艺制程

芯片制造过程需要两千多道工艺制程，下面，我们按照 8 大步骤对芯片制造工艺进行简单介绍。

（1） 光刻（光学显影）

光刻是经过曝光和显影程序，把光罩上的图形转换到光刻胶下面的晶圆上。光刻主要包含感光胶涂布、烘烤、光罩对准、 曝光和显影等程序。曝光方式包括：紫外线、极紫外光、X 射线、电子束等。

（2）刻蚀（蚀刻）

刻蚀是将材料使用化学反应或物理撞击作用而移除的技术。干刻蚀（dry etching）利用等离子体撞击晶片表面所产生的物理作用，或等离子体与晶片表面原子间的化学反应，或者两者的复合作用。湿刻蚀（wet etching）使用的是化学溶液，经过化学反应达到刻蚀的目的。

（3）化学气相沉积（CVD）

CVD 利用热能、放电或紫外光照射等化学反应的方式，将反应物在晶圆表面沉积形成稳定固态薄膜（film）的一种沉积技术。CVD 技术在芯片制程中运用极为广泛，如介电材料（dielectrics）、导体或半导体等材料都能用 CVD 技术完成。

（4）物理气相沉积（PVD）

PVD 是物理制程而非化学制程，一般使用氩等气体，在真空中将氩离子加速以撞击溅镀靶材后，可将靶材原子一个个溅击出来，并使被溅击出来的材质如雪片般沉积在晶圆表面。

（5）离子植入（Ion Implant）

离子植入可将掺杂物以离子型态植入半导体组件的特定区域上，以获得精确的电特性。离子先被加速至足够能量与速度，以穿透（植入）薄膜，到达预定的植入深度。离子植入可对植入区内的掺质浓度加以精密控制。

（6）化学机械研磨（CMP）

化学机械研磨技术具有研磨性物质的机械式研磨与酸碱溶液的化学式研磨两种作用，可以使晶圆表面达到全面性的平坦化，以利后续薄膜沉积。

（7）清洗

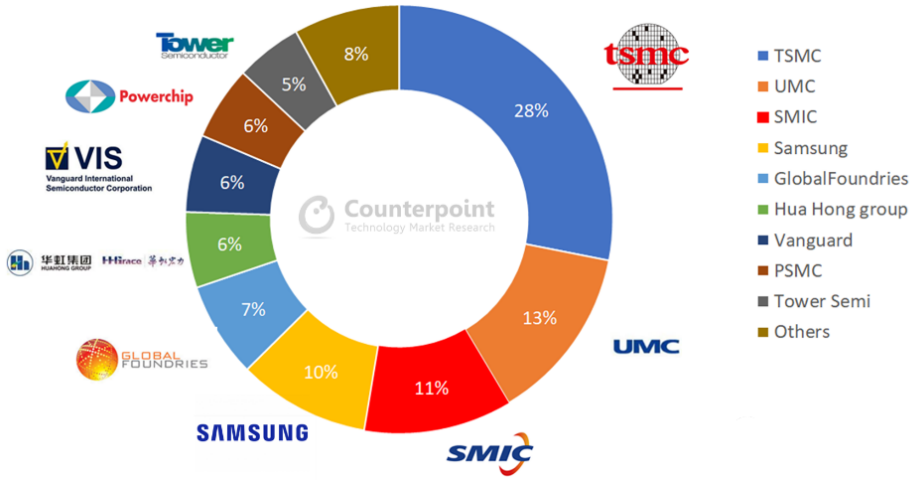
清洗的目的是去除金属杂质、有机物污染、微尘与自然氧化物；降低表面粗糙度；几乎所有制程前后都需要清洗。

（8）晶片切割（Die Saw）

晶片切割是将加工完成的晶圆上一颗颗晶粒裸芯片（die）切割分离，便于后续封装测试。

虽然不同的 Foundry 厂的流程大致相同，但不同的工艺控制能力造就了各厂家在先进制程上的区别，随着制程进入 5nm，能够量产的芯片制造商就屈指可数了，目前能够量产 5nm 芯片的只有 TSMC 和 SAMSUNG。两千多道工艺制程中隐藏着 Foundry 的无穷的智慧和雄厚的财力，并不是说有了先进的设备，就能造出合格的芯片。虽然先进制程是技术发展的方向，我们也不能忽

视成熟制程。成熟制程依然有很大市场份额。下图是按成熟制程（节点≥40nm）产能排序的全球晶圆代工厂商 Top 榜单。



可以看出，成熟制程产能排名前四的厂商分别为：台积电（市占率28%），联电（13%），中芯国际（11%），三星（10%）。成熟制程在2020 年非常火爆，产能严重短缺，这给各大晶圆代工厂带来了巨大的商机。而从 2021 年的产业发展形势来看，这种短缺状况在近期内还难以缓解。

2.3 材料

生产集成电路的材料有成千上万种，我们就以最为典型的硅晶圆和光刻胶进行分析。

■ 硅晶圆

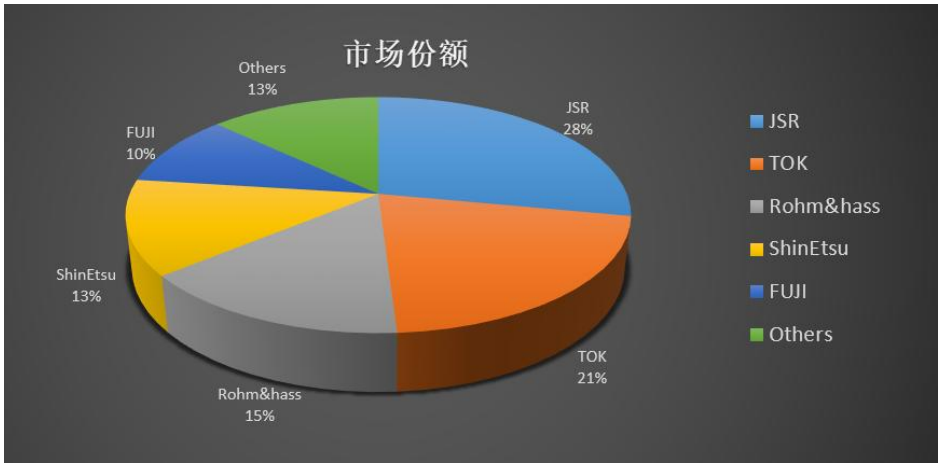
硅晶圆是集成电路行业的粮食，是最主要最基础的集成电路材料，90%以上的芯片在硅晶圆上制造，目前 300mm 硅晶圆是芯片制造的主流材料，使用比例超过 70%。曾经，我国 300mm 半导体硅片 100%依赖进口，是我国集成电路产业链建设与发展的主要瓶颈。全球主要的半导体硅晶圆供应商包括日本信越化学（Shin-Estu）、日本盛高（SUMCO）、德国 Siltronic、韩国 SK Siltron 以及中国台湾的环球晶圆、合晶科技等公司。五大晶圆供货商的全球市占率达到了 92%，其中日本信越化学占 27%，日本盛高占 26%，台湾环球晶圆占 17%，德国 Silitronic 占 13%，韩国 SK Siltron 占 9%。下表列出了全球 10 大硅晶圆提供商，供参考。

排名	厂商名称	LOGO	国家地区	公司网站
1	信越 (Shin-Etsu)		日本	https://www.shinetsu.co.jp/
2	盛高 (Sumco)		日本	http://www.sumcosi.com/
3	环球晶圆 (Global Wafer)		中国台湾	http://www.sas-globalwafers.com/
4	世创 (Siltronic)		德国	https://www.siltronic.com/
5	SK siltron		韩国	https://www.sksiltron.com/
6	Soitec		法国	https://www.soitec.com/
7	合晶 (Wafer Works)		中国台湾	http://www.waferworks.com/
8	Okmetic		芬兰	https://www.okmetic.com/
9	嘉晶(Episil)		中国台湾	http://www.epi.episil.com/
10	上海新昇		中国	http://www.zingsemi.com/

国内的情况，中国大陆半导体硅晶圆销售额年均复合增长率达到 41.17%，远高于同期全球半导体硅片市场的 25.75%。但这块市场并没有掌握在本土厂商手中，在打造国产化产业链的今天，还有很大的空间供国内晶圆制造商去发展。

■ 光刻胶

光刻胶是光刻过程最重要的耗材，光刻胶的质量对光刻工艺有着重要影响。光刻胶可分为半导体光刻胶、面板光刻胶和 PCB 光刻胶。其中，半导体光刻胶的技术壁垒最高。目前全球光刻胶主要企业有日本合成橡胶（JSR）、东京应化（TOK）、信越化学（ShinEtsu)、富士电子（FUJI）、美国罗门哈斯（Rohm&Hass）等，市场集中度非常高，所占市场份额超过 85%。下图显示的是光刻胶企业的市场占有率。



高分辨率的半导体光刻胶是半导体化学品中技术壁垒最高的材料，日美企业技术领先国内企业二十年至三十年。从光刻胶技术水平来看，国内企业在缺乏经验、缺乏专业技术人才、缺失关键上游原材料和设备的条件下，探索出一条自主研发之路，光刻胶高端技术短期内尚难突破，还要很长的路要走。在 PCB 领域，国产光刻胶具备了一定的量产能力，已经实现对主流厂商供货。

三、封装测试

封装测试是集成电路三大产业中的最后一个环节。一般认为封装测试的技术含量和实现难度比前两者低，但是随着 SiP 及先进封装技术的出现和迅速发展，需要重新定义芯片的封装和测试。SiP 及先进封装在封装原来的三个特点：芯片保护、尺度放大、电气连接的基础上，增加了三个新特点：提升功能密度、缩短互联长度、进行系统重构，因此其复杂程度和实现难度与传统的封装相比有很大程度的提升。同时，SiP 及先进封装也给封装测试提出了新的机遇和挑战。

1、芯片封装

我们从封装设计和产品封装两方面来分析芯片封装。

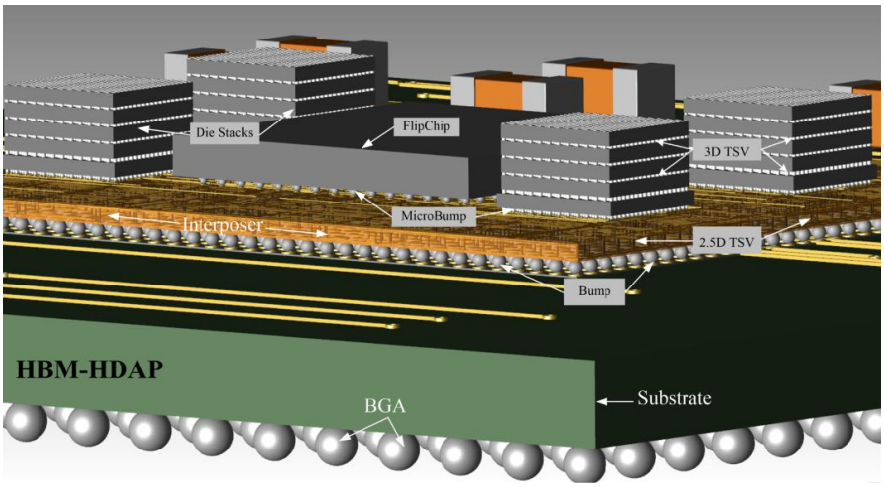
1）封装设计

早先的封装中没有集成（Integration）的概念，封装设计是比较简单的，对工具要求也很低，Auto CAD 就是常用的封装设计工具，随着 MCM、SiP 技术的出现，封装设计变得越来越复杂，加上目前 SiP、先进封装、Chiplet、异构集成概念的市场接受度越来越高，封装内集成的复杂度和灵活度急剧上升，对封装设计的要求也越来越高，SiP 和先进封装设计工具目前只有 Cadence 和 Siemens EDA（Mentor）两家，Cadence 是老牌的封装设计 EDA 提供商，市场占有率高，用户的忠诚度也比较高。

Siemens EDA（Mentor）是封装设计领域的后起之秀，但其技术先进性上则体现了“后浪”的特点。业界大佬 TSMC，Intel，SAMSUNG 纷纷选择 Siemens EDA 作为其先进封装（HDAP）的首选工具，主要在于两点：先进的设计工具和强悍的验证工具。

首先我们说说设计工具，在一次技术论坛中，我说：“不同于传统封装设计，先进封装和 SiP 设计对 3D 环境要求很高，3D 设计环境不在于是否看上去很直观、绚丽，而在于对客观元素的精准描述，包括键合线、腔体、芯片堆叠、硅转接板、2.5D 集成、3D 集成，Bump...”

在这一点上，Siemens EDA 的 SiP 及先进封装设计工具已经远远将其竞争对手抛在身后。下图为先进封装版图设计工具 XPD 中的封装设计 3D 截图，4 组芯片堆叠中，每组 5 颗芯片（4HBM+1Logic）以 3D TSV 连接在一起，和 GPU 一起集成在硅转接板（2.5D TSV）上，硅转接板和电阻、电容等一起集成在封装基板上。

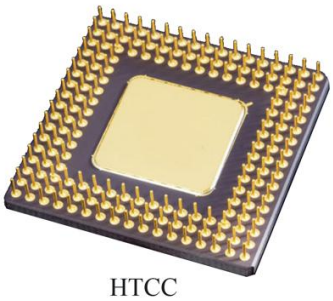


XPD 中的先进封装设计截图(3D)

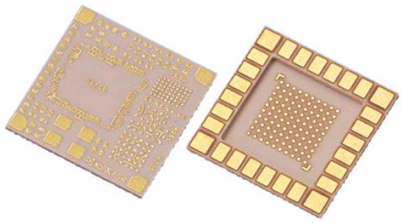
该设计中包含了 3D 集成、2.5D 集成、倒装焊、Bump、多基板集成等多种方式，在 XPD 设计环境中得到了精准的实现。先进封装验证工具包括电气验证和物理验证，电气验证包含 80 多条规则，对整个系统进行信号完整性、电源完整性、EMI\EMC 等电气相关的检查和验证，物理验证则是基于 IC 验证工具 Calibre，整合出 Calibre 3D STACK，专门用于 3D 先进封装的物理验证。随着封装内的集成度、设计复杂度越来越高，对工具的要求也越来越高，另外，在先进封装领域，封装设计和芯片设计的协同度日益提高，在某种程度上有逐渐融合的趋势，因此对协同设计的要求也日益提升。关于 SiP、微系统、先进封装的详细设计方法和实际案例，可参考电子工业出版社近期即将出版的新书：《基于 SiP 技术的微系统》

2）产品封装

根据材料和工艺不同，封装可以分为塑料封装、陶瓷封装和金属封装三种类型。塑封主要基于有机基板，多应用于商业级产品，体积小、重量轻、价格便宜，具有大批量、低成本优势，但在芯片散热、稳定性、气密性方面相对较差。陶瓷封装和金属封装则主要基于陶瓷基板，陶瓷封装一般采用 HTCC 基板，金属封装则多采用 LTCC 基板，对于大功耗产品，散热要求高，可选用氮化铝基板。陶瓷封装特点包括：密封性好，散热性能良好，对极限温度的抵抗性好，容易拆解，便于问题分析；和金属封装相比体积相对小，适合大规模复杂芯片，适合航空航天等对气密性有要求的严苛环境应用；但价格昂贵，生产周期长，重量和体积都比同类塑封产品大。金属封装特点包括：密封性好，散热性能良好，容易拆解，灵活性高；但体积相对较大，引脚数量较少，不适合复杂芯片，价格贵，生产周期长，需要组装金属外壳和基板，工序复杂，多应用于 MCM 设计，航空航天领域应用较为普遍。陶瓷封装和金属封装内部均为空腔结构，具有可拆解的优势，便于故障查找和问题“归零”，因此受到航空航天等领域用户的欢迎。



HTCC



LTCC

2、芯片测试

芯片测试的项目非常多，这里我们重点了解一下机台测试的系统测试。

■ 机台测试

一般是指采用 ATE（Automatic Test Equipment）自动测试设备来进行芯片测试，测试芯片的基本功能和相应的电参数。机台可以提供待测器件 DUT（Device Under Test）所需的电源、不同周期和时序的波形、驱动电平等。测试向量（Test Vector）是每个时钟周期应用于器件管脚的用于测试的逻辑 1 和逻辑 0 数据，是由带定时特性和电平特性的波形代表，与波形形状、脉冲宽度、脉冲边缘或斜率以及上升沿和下降沿的位置都有关系。

测试向量可基于 EDA 工具的仿真向量（包含输入信号和期望的输出），经过优化和转换，形成 ATE 格式的测试向量。利用 EDA 工具建立器件模型，通过建立一个 Testbench 仿真验证平台，对其提供测试激励，进行仿真，验证结果，将输入激励和输出响应存储，按照 ATE 向量格式，生成 ATE 向量文件。

■ 系统测试

系统测试也称为板级系统测试，是指模拟芯片真实的工作环境，对芯片进行各种操作，确认其功能和性能是否正常。除了机台测试和系统测试之外，还需要对芯片进行了一系列的试验和考核，内容包括：热冲击、温度循环、机械冲击、扫频震动、恒定加速度、键合强度、芯片剪切强度、稳态寿命、密封、内部水汽含量、耐湿气等试验。只有所有的测试都顺利通过了，一颗芯片才能算成功，作为合格的产品应用到下一个环节。

四、自主可控总结

最后，结合下面表格，我们对自主可控作一个简单总结。

集成电路产业	设计生产相关环节	自主可控占比
芯片设计	EDA	0~5%
	IP	0~3%
	IC 设计流程	80~100%
芯片制造	光刻机	5%
	刻蚀机	20%
	工艺制程	30%
	硅晶元	5%
	光刻胶	5%
封装测试	封装设计 EDA	0~3%
	封装（SiP）设计	80~100%
	产品封装	80%
	芯片测试	80%
总 结	一颗芯片从规划到产品	？？？

从表格可以看出，我们在 IC 设计流程、封装（SiP）设计，以及在产品封装、芯片测试环节的自主可控程度比较高；在刻蚀机、芯片工艺制程上有一定的自主可控性，而在 EDA，IP，光刻机，硅晶元，光刻胶等环节自主可控的程度非常低，所以高端芯片很容易被“卡脖子”，因为高端芯片所用到的 EDA，IP，光刻机，硅晶元，光刻胶几乎全部依赖进口。自主可控相对较高的 IC 设计流程、封装（SiP）设计也几乎全部依赖进口的 EDA 工具，在产品封装和芯片测试环节，封装设备和测试设备大约 80%以上是进口设备；工艺制程上高端芯片同样也无法自主生产。考虑到这些，不由得让我们无法盲目乐观，因为越往源头挖掘，自主可控的比例就越低。当别人不卡脖子的时候，不要趾高气扬，似乎一切尽在掌控；当别人卡脖子的时候，不要突然发现，竟然全身上下都是脖子！

看完此文，如果以后有人告诉你，他做的芯片实现了 100%的自主可控，我们就可以从上面的环节逐个去分析，一颗芯片从最初的构思到最终的产品，所经历的过程中，那些环节真正是自主可控的？那些环节依然是要被卡脖子的？

只有真正认识到自身的不足，实事求是，踏踏实实，一步一个脚印，并持之以恒，才能在激烈的竞争中不致落后，从而减少卡脖子事件的发生！另外，即使世界出现了诸多不和谐，甚至在某些方面矛盾有激化的可能，但从长远来看，合作依然是人类文明的主流，我们依然要向着这个方向去看，去努力奋斗！

敬请关注联盟微信公众号
COPU开源联盟

中国开源软件推进联盟秘书处　　电话：+86 010-88558999
联盟公共邮箱：office@copu.org.cn　　联盟官网：http://www.copu.org.cn
地址：北京市海淀区紫竹院路66号赛迪大厦18层