架构师

ARCHITECT

特刊

2022 年度
技术盘点和展望

目录

2022 年度技术盘点

争相上市、抢夺本土市场，未来三五年数据库将迎来大洗牌 | 解读数据库的 2022 1

解读数字化的 2022：不再追求大而全的“军备竞赛”，用聚焦来提高转型“成功率” 16

直面成本“刺客”、拒绝繁杂技术花样，压力之下云厂商改变方向 | 解读云原生的 2022 31

市场增速超 20%，国产操作系统“浴火重生” | 解读操作系统的 2022 54

取代搜索，“干掉”艺术家？顶流「AIGC」的疯狂与争议 | 解读 AIGC 的 2022 69

解读 Go 语言的 2022：泛型时代开启 87

大数据管理工具走向更易用与专精，但中外发展现状存在明显差异 | 解读大数据的 2022 118

奇点已来，推进 All on Serverless 有哪些困难、如何破局？| 解读 Serverless 的 2022 130

微服务虽已老生常谈，但生命力超出不少人想象 | 解读微服务的 2022 139

2023，可观测性需求将迎来“爆发之年”？ | 解读可观测技术的 2022 152

数据烟囱亟需打破，云原生融合数据库雪中送炭 | 解读云原生数据库的 2022 166

从大前端“穿越”到终端，开发者应该必备什么技能？| 解读终端的 2022 176

展望技术人的 2023

降本增效的大环境下，该不该砍掉测试工程师？| 展望测试工程师的 2023 198

一个架构师在 2023 年需要掌握哪些“必杀技”？ 209

展望后端研发工程师的 2023：“后端难学”源于知识体系匮乏，面试时这三点是加分项 225

寒冬未散，手里没点“必杀技”，怎么翻过浪浪山？ | 展望数据库工程师的 2023..... 239

从 JS 到全生态，云原生时代下的前端成长演进之路 | 展望前端工程师的 2023..... 262

技术人穿越周期的生存之道是转型管理？ | 展望技术管理者的 2023 290

卷首语

作者：蔡芳芳

热热闹闹的 InfoQ 2022 年度技术盘点与展望，已经到了尾声。

在这场盘点中，InfoQ 编辑部全体编辑同学，细数了操作系统、数据库、AI、大数据、云原生、架构、大前端、编程语言、数字化、可观测等超过十个方向，同时，围绕“不同岗位的技术人，2023 年可能需要聚焦什么方向、什么技能、做好哪些准备”邀请专家进行直播对话，同时收集了数十条社区提问，在线反馈给专家们。

在这场盘点中，我们收获了关于技术圈的 2022、2023 的许多精彩观点和认知，比如：

- 2022，标准化、国产化、DevSecOps、云原生网关、Proxyless、统一控制面和多语言，正在给微服务带来的新的市场机遇；
- 2022，eBPF 成为当下最火的技术之一，为操作系统内核提供了新的可能性；从生态繁荣程度上来看，RISC-V 生态正处于增长关键期；当前国产移动操作系统还处于准备期，离带动整个产业发展还有一定的距离，但离成熟期也并不遥远；
- 2022，容器的“黑盒”打开，混部带来效率提升，备受企业欢迎；Serverless 基于容器完成标准化，开启应用“元年”；降本增效大主题下，FinOps 理念得到快速发展；越来越多传统行业开始应用云原生技术；
- ……

2022 年度技术盘点

争相上市、抢夺本土市场，未来三五年数据库将迎来大洗牌 | 解读数据库的 2022

作者：李冬梅

本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦数据库领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年数据库领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

特此感谢 Jags Ramnarayan、Pater、Michael Cahill、孙元浩、陶建辉、杨传辉

> （花名日照）（按姓名首字母排序）对本文的贡献，他们的真知灼见，是本文能与大家见面的关键。

对于数据库技术领域而言，2022 年是一个激动人心的重要年份。过去一年，数据库技术仍围绕着云原生、Serverless、分布式等核心方向快速发展，但是相较于前几年，数据库市场的客户逐渐成熟，对产品的理解更加理性，相应地要求也越来越高。不难看出，数据库领域的发展已经步入深水区。

如果说过去数据库厂商们把更多的注意力聚焦在框架和架构搭建上，那么 2022 年，则进入了对数据库性能、成本、易用性等细节打磨的阶段，开始了对性能的极致追求。

2022 年数据库领域大事件回顾

1 月 21 日，北京国家金融科技认证中心正式发布《2021 年分布式数据库标准符合性检测参检机构名单》，腾讯云分布式数据库 TDSQL 名列其中，成为首款通过检测的互联网公司数据库产品。

2 月 11 日，Elastic 8.0 版本正式发布，重点改进 Elasticsearch 的矢量搜索功能，并提供对现代自然语言处理模型的原生支持、进一步简化数据上线过程，以及精简的安全防护体验。在速度、扩展幅度、相关性和简便性方面，Elastic 8.0 版本迎来了一个全新的时代。

2 月 17 日，由信通院主办的国内首款金融数据库性能测试工具开源发布会在线上召开。会上正式开源了金融数据库性能测试工具 DataBench-T。

4 月 1 日，TiDB 宣布正式开源分析引擎 TiFlash，它是为 TiDB 提供 HTAP 能力的重要组成部分。TiFlash 的框架代码基于 ClickHouse，但和 ClickHouse 拥有完全不同的擅长场景，TiFlash 完全偏重于事务性数据的分析。

4 月，星环科技升级多模型大数据基础平台 TDH 9.0，实现 9 种存储引擎支持 10 种数据模型。

4 月 27 日，Redis 正式发布 7.0 版本，添加了近 50 个新命令和选项来支持这种演变并扩展 Redis 的现有功能。

5 月 12 日，在谷歌 I/O 2022 开发者大会上，谷歌云平台 GCP 宣布推出 AlloyDB for PostgreSQL，这是一个全托管的、与 PostgreSQL 兼容的数据库服务。

5 月 16 日，在中国信通院组织的第十四批“可信数据库”产品能力评测中，华为云云数据库 GaussDB（for openGauss）完成信通院首个防篡改数据库产品测评。

6 月 8 日，MongoDB World 2022 大会上，MongoDB 宣布推出可查询加密功能和关系型数据库迁移工具（Relational Migrator），借助这项技术，不具备密码学相关经验的开发者可以在不影响性能的情况下，以简单直观的方式查询经加密的敏感数据。

6 月 29 日，武汉达梦数据库股份有限公司正式递交招股说明书，拟科创板挂牌上市。根据招股说明书，达梦本次公开发行股票数量不超过 1900 万股，计划募资 23.51 亿元。

6 月 29 日，一体化实时 HTAP 数据库 StoneDB 正式开源。StoneDB 是一款支持行列混存+内存计算的 HTAP 数据库，采用自主研发的存储和计算引擎，可将 MySQL

的分析性能提高 100 倍。

7 月 23 日消息，蚂蚁集团时序数据库 CeresDB 正式开源，并发布开源版本 CeresDB 0.2.0 。蚂蚁集团表示：“通过开源，希望帮助用户解决时间序列数据存储的水平扩展与高可用的痛点，乃至针对时序数据的复杂分析计算能力的需求”。

8 月 10 日，OceanBase4.0（小鱼）正式发布，这是业内首个单机分布式一体化数据库，可以实现单机部署并兼顾分布式架构的扩展性与集中式架构的性能优势，RTO 从 30 秒迈入 8 秒，进入真正的秒级容灾时代。发布会还宣布全球开服 OceanBase Cloud。

8 月 13 日，涛思数据发布 TDengine 3.0 版本，该版本带来了几大核心特性，包括云原生架构、流式计算，还增强了数据订阅功能；更重要的是，3.0 系列版本开始提供了全平台支持，服务器/客户端均已支持 Linux、Windows 和 MacOS。

10 月 18 日，“国产大数据基础软件第一股”星环科技正式登陆科创板，发售价为每股 47.34 元，开盘后报每股 72 元，大涨 52%。

10 月 18 日，Oracle 宣布推出 MySQL HeatWave Lakehouse 云数据湖库。据 Oracle 称，MySQL HeatWave 可以在多个云上提供一种集成服务，用于事务处理、跨数据仓库和数据湖的分析、以及无需 ETL 的机器学习。

11 月，MariaDB 团队宣布正筹备在中国成立实体公司，并积极准备拓展中国市场。明年，MariaDB 将有一个专门的本地团队为中国市场服务，并将发布新产品，如 Xpand（MariaDB 的分布式 SQL 数据库），此外，MariaDB 团队还计划将云版本 SkySQL 引入中国市场。

12 月 19 日，MariaDB Plc.公司表示，已经完成了与特殊目的收购公司 Angel Pond Holdings Corporation 的合并，合并后的公司更名为 MariaDB Plc.。MariaDB 的普通股和权证已于 12 月 19 日登录纽约证券交易所，此次合并已获得两家公司董事会和股东的批准。

数据库领域年度关键词

数据库并不是一个新领域，它已经发展了 40 年，作为基础软件之一，它是一个既传统又古老的领域。但近几年数据量爆发、云化趋势加深等因素，正在推动数据库领域进入新的发展阶段。

InfoQ 在采访国内外多名资深数据库专家后得出了 2022 年数据库领域的三大关键词，其中，“融合”一词成为了被提及次数最多的词语。

融合

这里的提及的“融合”不仅仅指的是数据库各种类的融合，更包括软硬件融合、混合负载融合、分布式与单机的融合、多模态的融合等，但实际上融合问题并不好解决。

比如，软硬件融合就是个难题。数据库经历了多年发展，其底层架构和算法针对过去的硬件网络等基础设施经过充分设计，以获得最佳性能，但是随着新型硬件的发展，例如新 CPU 架构、新存储介质，以及新网络架构的发展与成熟，数据库底层机制也必须有相应的调整，才能充分发挥新硬件的特性。

此外，分布式与单机的融合也有一定技术难度。单机数据库相较于分布式数据库，由

于不需要大量的分布式事务处理，所以单核性能会高于分布式数据。但是数据库面临的市场是一个复杂多样的市场，有一些场景可能一个单机数据库就能支持，用分布式显然有点“大材小用”，但是有一些场景又需要分布式数据库以支撑海量数据和海量交易。那么如何在数据库规模比较小的时候，表现为一个单机数据库，不需要复杂的分布式事务机制，而随着数据规模增长，又能自动生长为一个分布式数据库，而不需要大规模的数据库迁移等，就是一个必须考虑的关键问题。

以 TDSQL 为代表的多家厂商开始尝试采用对等架构实现存算分离与存算一体的兼顾，使数据库产品能兼顾不同规模的场景，取得不同的平衡；而 OceanBase 则推出单机分布式一体化架构，打造更加开放、灵活、高效、易用的下一代企业级分布式数据库。实现了在线水平扩展的同时不增加分布式相关 overhead，从而能够像集中式数据库一样部署在小规格的服务器上，做到单节点性能达到甚至超越集中式数据库的水平。

AI 与数据库的融合也在进一步加深。进入 2022 年以后，AI 和数据库融合趋势愈加明显，AI 已经成为企业挖掘数据商业价值的最大推动力。过去，企业数据可用性不高，样本容量也有限，无法在毫秒内分析大量数据；而人工智能和机器学习技术，让企业在数据容量和数据源可用性方面有了质的飞越，才有了真正的大数据访问和实时处理能力。

大统一是诸多自然科学领域的最终梦想，而数据库的大融合也是数据库从业人员的梦想，但是这里面往往涉及到诸多因素的权衡，以及要面临越来越复杂多样的场景，现阶段还很难做到完美的统一。

数据云

数据云也是今年被频频提及的关键词之一，它更多是一个数据基础设施的概念。从技术的角度来讲，数据云是为了提升整个数据分析能力，提升数据分析的实时性；从业务角度来讲，数据云可以帮助数据库厂商在云上拓展更多不同客户。

可以看到，今年很多开源数据库产品都发布了自己的 Cloud 版本，也就是云服务订阅版本，包括 TiDB、OceanBase、SelectDB、TDengine 等等国内知名数据库厂商。这无疑是开源数据库产品探索商业化一个比较好的路线。相较于云厂商凭借数据库的深度软硬件协同提供产品竞争力，开源厂商基于多个云平台提供的多云容灾切换方案，也有一定的市场发展空间。

Serverless

这几年，Serverless 数据库大火，被称为是下一代数据库的演进方向。事实上，Serverless 数据库并非这两年才有的新概念。早在 2004 年，由于亚马逊的电商网站面临数据库扩展性的挑战，Serverless 数据库的探索之旅便已经开始。

Serverless 数据库可以根据应用程序的需求自动扩缩容，并内置高可用和容错能力。Serverless 数据库背后由云服务商动态管理计算资源分布，云服务商将计算资源变成一个池子，用户无需创建实例，就能快速完成部署。

具体到数据库层面，用户只要使用了基于 Serverless 特征的数据库，就可以基于函数运行所消耗的内存和执行时间，按使用量计费，极大地降低了数据库的使用成本。

Serverless 发展至今，Serverless 数据库开始强调极致的资源分配曲线与业务实际需求曲线匹配，目前已经可以做到秒级的按需计费，既能提高资源利用率，还不影响业务的突发流量，因此也被越来越多企业所采用。

2022 年，数据库领域这些趋势值得关注

作为三大基础软件之一，如今数据库领域面临着重大发展机遇——据 IDC 预测，2022 年全球数据管理软件市场规模将达到 849.7 亿美元，到 2026 年将增至 1378.2 亿美元。如此大的市场空间，到底潜藏着哪些机遇和挑战？放眼即将过去的 2022 年，有哪些重要趋势值得关注？

自研分布式数据库迎来落地爆发期

在云计算发展和国产化趋势的双重驱动下，国产数据库需求迎来爆发式增长。在过去数年间，分布式数据库发展迅速，成为面对海量规模及高并发等场景的利器。通过数年实践，产品的基本功能、稳定性、性能等方面已趋于成熟，各数据库厂商开始在产品的易用性、可观察性、诊断能力、生态兼容等方向发力，并陆续有成果落地。

回首过去一年，随着国产数据库合作生态的日渐壮大以及对降本增效的迫切需求，传统行业纷纷开启了向分布式数据库转型升级之路。国产数据库投产的广度和深度持续增加，广度即商机增多，这一方面给诸多厂商提供了很大的机会，另外一方面也给当前尚不成熟的国产数据库产品能力和服务能力带来了更大的挑战；从深度来看，分布式数据库开始真正落地于金融业、证券业、餐饮业、汽车业、房地产业等传统企业的

核心系统中。

例如 TDSQL 今年就服务了 TOP 10 银行中的 7 家，OceanBase 也在富滇银行、招商证券、红塔银行等企业关键系统中大显身手，服务了头部金融机构中的四分之一。这些无疑都证明了国产自研数据库的能力，但也对数据库的稳定性、数据一致性等提出了更高要求。

上云趋势持续火热

近日，Gartner 发布了最新的数据库中国市场指南报告，报告指出中国数据库行业将加速增长并逐步向云端迁移。

其实数据库的上云趋势并不是今年才有的，早在几年前，许多国外公司甚至许多民营医院已经将数据库部署在云端。以往，金融、政府和公共服务等特定行业的组织对数据和信息保密要求较高，因此对公有云的数据安全、数据主权和法规合规有顾虑，更倾向私有云或本地部署。然而，Gartner 观察到一些国内企业、机构正考虑将核心任务事务型数据库迁移到公共云，这反映出最近行业对公共云的顾虑正在下降，混合云模式比重将进一步提升。

云数据库凭借按需付费、按需扩展、高可用性以及存储整合等优势已经在海外市场占据了大半市场份额。据业内权威人士称，美国至少有超过一半的数据库跑在云上。虽然目前国内数据库上云趋势还未如国外一样迅猛，但这一趋势已经十分明显。

对数据安全的重视达到了前所未有的程度

近年来，全球关键信息基础设施网络安全事件层出不穷，涉及电力、石油、制造等国计民生领域。就在 2022 年初，因遭到勒索软件的攻击，位于荷兰阿姆斯特丹和鹿特丹、比利时安特卫普的几处港口的石油装卸和转运受阻，让国际油价进一步走高到近年来最高的水平，数据面临的威胁从未像今天这样严重。

当前，用户主要面临的数据库安全威胁包括 SQL 注入攻击、缓冲区溢出、默认设置或弱口令、配置错误、用户帐户破坏等。数据被泄露或篡改可能导致企业面临无法通过审计导致的资产负债、监管罚款、盈利受损或客户投诉等诸多负面影响，作为承载数据的数据库系统，其安全性也关乎企业生存发展。

尤其是今年，政企各界对数据安全的重视达到了前所未有的程度。今年 8 月，国家互联网信息办公室编制了《数据出境安全评估申报指南（第一版）》，对数据出境安全评估申报方式、申报流程、申报材料等具体要求作出了说明。10 月 12 日，国家市场监督管理总局（国家标准化管理委员会）发布公告，批准 2 项国家标准。其中，《信息安全技术关键信息基础设施安全保护要求》作为 2021 年 9 月 1 日《关键信息基础设施安全保护条例》正式发布后的第一个关基标准，将于 2023 年 5 月 1 日实施。

产研结合，深化数据库人才培养

除了数据及数据库系统的安全外，今年数据库行业内的人才生态培养也迈向了新高度。为了推动高校与企业强化创新合作，促进创新链产业链深度融合，教育部、工业和信息化部、国家知识产权局决定共同组织开展“千校万企”协同创新伙伴行动（简称“千校万企”行动）。该行动主要目标就是把这些与数据库等“卡脖子”技术相关的

企业与高校联动起来，把基础软件这一数字化转型的底座构建得更扎实、更安全。

以星环科技为例，今年星环科技与国内多所知名高校达成合作，共同培养创新方面人才，通过这种创新对接的合作方式，运用信息化等手段建立完善校企对接常态化机制，降低对接成本，提升对接效率，促进协同创新。

数据库未来展望

如前文所述，在刚刚过去的 2022 年，以云原生和分布式为主的新型数据库开始大规模落地于各行业核心系统，此外，各类型数据库趋于融合的趋势也愈加明显。未来，数据库将如何发展？InfoQ 综合多位受访专家的思考，主要概括为以下几点：

国内数据库厂商 3~5 年内将仅存个位数

随着基础软件领域成为资本热土，在 2020-2021 年获得融资的国产数据库公司就超过 40 家。值得一提的是，仅 2021 年一年，新成立数据库公司就超过 30 家，获得新一轮融资的企业多达 20 家。

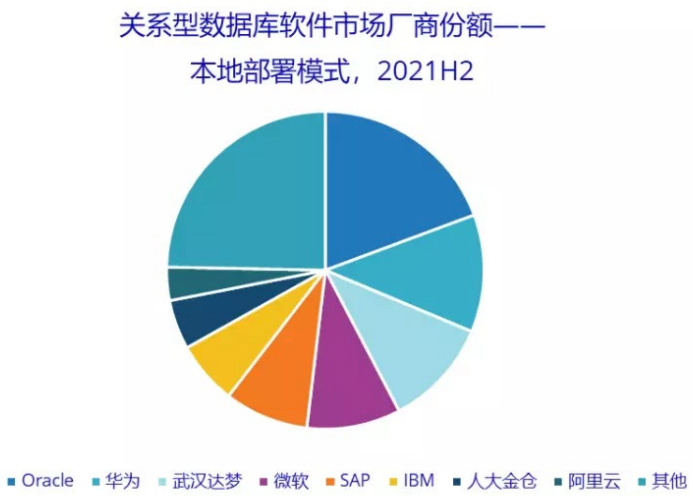
资本的青睐带来了过去几年数据库厂商的井喷增长，但资本过热也带来了相应的弊端：产品能力参差不齐、同质化现象严重、技术储备力量不足.....

虽然近年来国内数据库产业发展迅速，借助开源和云计算，中国数据库有机会在全球市场占据一席之地，但国内数据库市场整体相较于海外已经发展了四五十年来说还稍显不足。

多位受访人表示，2021 年，中国数据库资本处于泡沫的最顶峰，未来三到五年间，数据库厂商数量将会越来越少，因为客户和应用开发商不可能用那么高的成本去适配和选用不同的数据库，这个市场不可能容得下上百种数据库。随着客户逐渐开始使用国产数据库，只有那些个别坚持在数据库领域投入、并踏实打磨数据库产品的长期主义者才能提供优良的服务，才能最终留存下去。

抢夺本土市场之外，出海成为新增量

IDC 发布的《2021 年下半年中国关系型数据库软件市场跟踪报告》显示，前八强中除了亚马逊云科技、Oracle 和微软外，其余 5 家都是中国厂商，合计占据超过 74% 的市场份额。



来源：IDC 中国，2022

在本土做大做强后，数据库出海成为国产厂商的全新增量逻辑，欧美线路和东南亚线路是国产厂商出海的优选项。两年前，腾讯云数据库 TDSQL 就已经在印尼 Bank Neo Commerce 银行新核心系统完成搭建并正式投入使用；蚂蚁集团的 Oceanbase 先后服务过印度、印尼、菲律宾、巴基斯坦等国家的金融科技公司，并开始进军北美市场；涛思数据、星环科技等企业的出海业务也在持续增长；以 PingCap、Zilliz、Singularity 等为代表的数据库创业公司，则在自成立之初就定位国际化。

但中国企业出海面临的信任度、监管、地理位置、地缘政治等障碍也不可回避，上云、开源、新技术敏感度成为除基础技术以外，国产数据库顺利出海的关键要素。

开源与商业化相辅相成

自 21 世纪以来，数据本身的增长速度极快，尤其随着最近几年各行各业数字化，例如智能汽车、物联网、VR/AR 等等，每年产生和存储的数据始终保持高增长率。但是从管理数据规模来看，包括 Oracle/微软/Teradata 等这类传统商业数据库的比重逐渐在降低，而开源数据库所占比重越来越大。

据 DB-Engines 发布的 2022 年 12 月全球数据库排行榜可以看到，以 MySQL、PostgreSQL、MongoDB 为代表的开源数据库占据着数据库市场的半壁江山，对于这些软件而言，开源是其取得成功的关键。

Rank			DBMS	Database Model	Score		
Dec 2022	Nov 2022	Dec 2021			Dec 2022	Nov 2022	Dec 2021
1.	1.	1.	Oracle	Relational, Multi-model	1250.31	+8.62	-31.43
2.	2.	2.	MySQL	Relational, Multi-model	1199.40	-6.14	-6.64
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	924.35	+11.84	-29.67
4.	4.	4.	PostgreSQL	Relational, Multi-model	617.97	-5.18	+9.76
5.	5.	5.	MongoDB	Document, Multi-model	469.33	-8.57	-15.34
6.	6.	6.	Redis	Key-value, Multi-model	182.57	+0.52	+9.03
7.	↑ 8.	7.	IBM Db2	Relational, Multi-model	146.61	-2.95	-20.56
8.	↓ 7.	8.	Elasticsearch	Search engine, Multi-model	144.93	-5.40	-12.80
9.	9.	↑ 10.	Microsoft Access	Relational	133.83	-1.20	+7.84
10.	10.	↓ 9.	SQLite	Relational	132.44	-2.19	+3.76

图片来源：DB-Engines

虽然开源的增长会对传统商业数据库市场带来一定冲击，但开源也助力了云数据库市场的发展。客观来讲，开源本身跟商业不冲突，有很多开源数据库与云这种结合产生了很多新的商业模式，例如很多开源数据库推出了云服务版，可以做多云容灾部署，为客户提供订阅类服务。

此外，云厂商本身也提供很多开源数据库的服务，例如 MySQL、PostgreSQL 等，开源一定程度也助力了云数据库这种订阅类商业模式的发展。

下一步多模数据库?

据 IDC 发布的《数据时代 2025》报告预测，全球每年产生的数据将从 2018 年的 33ZB 增长到 175ZB（1 ZB=1024 EB），以 25 MB/秒的网速进行下载，需要的时间约为 18 亿年，除了庞大的数据体量外，数据类型也变得复杂多样。如何将多类型数据进行统一管理，成为了数据库领域亟待解决的问题之一。

近年来，多模数据库开始兴起，并逐渐发展成一个主要的技术方向。多模数据库，指

的是能够将分散的数据，聚合于同一个系统中的数据库。它因具备多模式（Multi-model）数据管理和存储功能而得名，诸如图、键值、关系、文档等数据库模型，都能够得到多模数据库的有效支持。多模数据库代表了在云化架构下，多类型数据管理的一种新理念，也是简化运维、节省开发成本的一个新选择。

随着 NoSQL 系统在单模能力上的成熟，以及应用数据需求的多样化，业界开始进入对多模数据库的探索和实践，通过 DB-Engine 数据库排行榜可以看到，越来越多的流行数据库已经走向多模数据库。

采访嘉宾（按姓名首字母排序）

郭上阳，MariaDB 中国

Jags Ramnarayan，MariaDB 全球副总裁&SkySQL 总经理

Michael Cahill，MongoDB 研发实验室副总裁

Pater，腾讯云副总裁

孙元浩，星环科技创始人&CEO

陶建辉，涛思数据创始人&CEO

杨传辉（花名日照），OceanBase CTO

解读数字化的 2022：不再追求大而全的“军备竞赛”，用聚焦来提高转型“成功率”

作者：罗燕珊

本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦数字化领域在 2022 年的行业发展现状和趋势，希望能帮助你准确把握 2022 年数字化领域的核心发展脉络，在行业内始终保持足够的敏锐度。

特此感谢郭华、孔云飞、柳忠阳、沈旸、史凯、史喆、朱辉（按姓名拼音首字母排

> 序）对本文的贡献，他们的真知灼见，是本文能与大家见面的关键。

2022 年，数字化已成业内共识，它是必答题，但它没有标准答案，所有企业都在摸索中前行。

本文主要目的在于简要阐述数字化行业的现状和趋势，“寒冬”之际，数字化作为长期投入，企业会如何权衡数字化所带来的价值，他们在关心数字化的哪些方向，是技术？是效益？还是数据？他们是如何思考自身的定位和未来的趋势，将作何布局？

基于此，我们采访了数位专家，他们的真知灼见，是该文能与大家见面的关键。以下内容基于公开资料和受访专家意见整理，本文篇幅有限难以面面俱到，未尽之处还请见谅，希望能为读者带来参考价值，并激发更深入的探讨。

不确定时代，需要更务实的数字化

2022 年是疫情第三年，经济下行，市场低迷，企业面临各方面的压力与不确定性，纷纷勒紧裤腰带准备过冬。“这样的大背景给很多企业的数字化投资带来了挑战，很明显感觉有的企业的数字化项目取消了，有的项目停滞了。”凯捷亚太区副总裁，亚太区首席数据创新官史凯直言道。

据他介绍，今年在跟企业交流时听到的最多的一个词是“效益”。从短期来看，单个项目投入更注重效果和速度，不再执着于长期的大项目，更加希望找到投资小见效快能持续迭代的投资。这意味着，企业不再只关注内部的管理，也不满足于系统和应用的成功上线，更多地以终为始，如何提升企业的盈利能力，降本增效的实际价值角度

去思考，这些数字化技术和数据能够给他们带来什么样的效益，这也就是史凯在原创著作《精益数据方法论》中所梳理的，数字化转型将从业务价值出发，减少浪费，提升效益。

换句话说，与过往热衷于创新事物的现象相比，如今企业的数字化转型路径会显得更务实，不再追求大而全的“军备竞赛”，而是回归业务本身，聚焦业务价值。联易融副总裁沈旸表示，“在经济下行的情况下，企业会更清楚自己需要的是什么，从而把数字化的焦点放在主营业务的优化上。”

德勤中国数字化服务副总监孔云飞同样指出，企业如今会更注重理性分析自身经营最痛之处，从降本、涨收、促毛利出发，聚焦在某个领域形成局部以多打少的局面，塑造自己独有的具有差异化的竞争力。

比如，重制造企业选择投资数字化工厂，通过产线的自动化改造和生产工艺的改良，提升生产线的全过程管控，降低对人工的依赖，逐步完善成本管理。再比如，银行选择利用前沿的数据技术、机器学习，提升业务预测、主动获客能力等等。

普华永道发布的《2022 年数字化工厂转型调研报告》亦指出，危机时期，数字化工厂转型至关重要，是构建运营灵活性和韧性的可选路径，且将进一步推动卓越运营。

总的来说，从短期来看，清晰的目标，坚定的决心，务实的心态是当下大多数企业应对“寒冬”的不二法则。

有限资源下，聚焦更迫切的数字化场景

不过，虽然企业不能像原来在相对宽松的经济环境下那样，去进行更多的试错，但也不意味要盲目地“省钱”。

资源有限，那就集中精力办大事，把钱花在刀刃上。

以雪花啤酒为例，雪花啤酒数字化负责人郭华提及公司在三个方面的调整：策略聚焦、打法聚焦、考核聚焦。

- 策略聚焦，指的是方向聚焦，目标聚焦，要有重点地去推进工作。在每个工作领域里，确定一个旗舰项目和中心任务，如果选不准，那该领域的工作就会被认为是没有成效的；
- 打法聚焦，比如雪花现阶段主要是在队伍建设层面发力，拉通信息，降低沟通和管理成本，提升整体工作组织能力；
- 考核聚焦，指加强指标和目标化的管理，全面推进量化考核。

“无论是从选择的切入点，还是系统项目建设过程的管理能力、管理水平，或是从业务层面解决问题、释放价值的角度来看，当下数字化的建设都要更加聚焦、精准和精益。”郭华强调。

当然，每个企业由于所处行业、市场环境、业务形态、发展阶段等各方面的差异，聚焦的数字化场景不尽相同。对此，史凯总结了四类企业需求更为迫切的数字化场景：

第一，营销获客类，也就是能直接助力提升企业盈利能力的场景；第二，运营优化类，能够优化流程，降低成本，提升效率的场景；第三，用户体验类，通过数据提升用户

体验，深挖存量；第四，风控合规类，能够通过数据来提升企业风险管理和合规能力的场景。

孔云飞表示，尽管不同企业的痛点不一，但大部分企业都是想通过提高技术应用能力推动业财数据标准化、流程效率最大化，如此既降低了经营的不确定性，又可以显性提升管理层关注的毛利。很多企业想通过小投入撬动大收益，首先从重复劳动自动化开始，见效快、短期收益明显。对数字化着迷的企业，也会考虑轻量型创新技术以尝试改变现有管理模式，推动数据服务化的同时，更是朝着服务资产化的长远目标逼近，例如有很多国内外企业开始引入流程挖掘作为流程管理和业务变革的基座平台，让企业管理层也能轻松参与到持续改善经营的最前线。

而沈旸也提到，今年很多企业做的工作越来越像以前在信息化时代所做的，更关注业务流程设计，关注流程优化，包括在制造企业，装备和设备的自动化的需求会更加迫切。

数字化转型“成功率”仅有 16%，挑战来自方方面面

综上，数字化依然被认为是「不确定的时代里，唯一的确定性」：不少企业在面临增长挑战的同时，依然笃定数字化的投入。但知易行难，根据麦肯锡全球调研显示，全球范围仅有 16%的企业认为自己的数字化转型取得了持续成效。

那么，是什么原因造成企业数字化转型“成功率”如此之低？问题究竟出自哪里？

挑战一：组织、流程、文化的转型滞后

IBM 大中华区客户成功管理部总经理朱辉认为，当前企业数字化转型面临着技术与非技术的双重挑战。

他表示，从非技术的角度来看，数字化转型是“在充分释放数据价值和利用人工智能能力的前提下，重新打造一家企业的过程”。如果企业仅把数字化转型理解为，在现有的组织架构、业务流程、企业状况之下，利用技术实现降本增效的过程，那么从希望值上就已经出现了比较高的风险。

史凯举例，当组织架构和业务流程滞后于技术转型时，业务人员之间，以及业务与技术人员之间就会很难就场景的度量效果达成一致。“通常情况下，往往是在谈理念、谈方法都很和谐，但是一旦谈到具体的场景的效果度量，落到实施层面，就会各种扯皮。”所以对齐业务价值，找到杀手场景是转型的抓手和驱动力。

“很多数字化转型的企业到最后会发现，它其实要做的是组织架构的调整，是职位岗位的调整，是员工技术技能的转换或者是改变。”朱辉表示。

因此，文化、组织架构以及流程的改变，是当下数字化转型过程当中碰到的主要障碍，企业数字化转型必须要有全面的战略，但凡其中一项是短板，都很有可能成为失败的导火索。

换言之，一方面，企业需要有全局的、长远的视角，循序渐进推进数字化工作，其中战略、节奏的调整很重要；另一方面，需要找到合适的落地场景，并进行与战略相匹配的组织流程重构工作。

挑战二：数据“地基”还没打好

数据是数字经济时代的基础性资源和战略性资源，也是重要的生产力。如今业内常说的数字思维，也可以简易理解成“数据思维”。在数据分析已渗入到各行各业的商业活动的当下，“数据思维”对企业的重要性毋庸置疑。

从技术角度的维度看，朱辉认为企业数字化转型面临的最大挑战恰恰与“数据”息息相关：企业级数据的质量和治理的成熟度，会直接影响数字化转型的效果。“数据治理是与数据相关的项目当中难度最高、复杂度最高、时间跨度最长、投入最大的一类项目。但如果没有把数据治理作为数字化转型过程中的必经一环，以及必须要做好的那一环，那么很多工作都会难以顺利推动。”

从目前来看，很多企业在使用数据的过程中还存在诸多问题。

比如，管理者有数据意识，但员工却缺少数据思维，他们面对纷繁的业务，不知道如何利用数据来指导决策，多数情况下仍然依赖于人的经验；比如，业务人员缺少技术与数据知识，而技术人员和数据人员缺少业务思维，业务与技术之间无法融合，使得数据的使用缺少价值场景；再比如，由于数据孤岛严重、数据质量不高，就会造成数据资源浪费，最终结果仍然是无法利用数据提升企业的盈利能力。

“为了打好数据这个‘地基’，部分企业正在尝试综合各种管理手段和技术应用一步一步往前走，但是企业情况各异、内功参差，有的走得快，有的走得慢些。”孔云飞说道。

今年 6 月，中央全面深化改革委员会第二十六次会议审议通过了《关于构建数据基础制度 更好发挥数据要素作用的意见》，初步构建了数据基础制度体系的“四梁八柱”，

对数据确权、流通、交易、安全等方面做出部署。

在相关政策的引导和扶持下，我国正加快培育数据要素市场，促进数据要素价值释放。各地积极开展探索，布局数据交易平台建设，国内最早成立的大数据交易所为贵阳大数据交易所，于 2015 年开始运营。去年至今，北京、上海、广州、深圳等地陆续有数据交易机构设立。这也为推进数据合法合规交易，进一步释放数据价值提供了支撑平台。

挑战三：人才缺口大

再往深处看，“人”这个因素，也被不少专家认为是普遍的挑战。

“一个是人的能力，一个是人的思维，其实这是最大的困难，人的问题解决了，很多问题都迎刃而解。”在郭华的感知中，企业人员若守着旧的方法和惯性，容易盲目自信和“交学费”。

富士康首席数字化官史喆也强调，数字化转型是需企业全员重度参与的改革项目，只有全面部署，系统深入才能解锁和释放数字价值。很多传统的人员观念，系统模式等都要有全新全方位的思考和重建，只有这样才能满足转型需要。

但是，从目前来看，数字化人才的缺口是有目共睹的。举例来说，在中国，目前 ICT 产业相关的人才供需比是 1:9，ICT 领域从业人员缺口达 1200 万，其中人工智能、云计算、大数据等新兴领域的缺口超过 900 万。

此外，新兴技术人才及复合型人才尤为紧缺。以制造业为例，在智能制造的大背景下，

很多企业需要的不再只是熟悉生产制造的作业人员，而是既懂设备又懂技术，同时又有行业经验的复合型人才——而现实情况是，这类人才目前在任何一个行业都是稀缺资源。

展望 2023，加速从点到面的数字化

虽然困难重重，但是孔云飞仍然表示，经过近几年的专注投入和快速发展，未来，将有越来越多的企业从单个业务环节、单个产业的数字化走向全集团、全产业链的数字化。“他们更注重将业财沉淀下来的数据资产化以支撑决策智能化，深度挖掘端到端的业务转型洞察，探索业务增长第二曲线，也是为了支持正在经历寒冬的企业能通过降本增效达成收入稳定、毛利上扬的经营目标。”

重视平台化建设

而为了挖掘数据价值，让数据充分流动起来是基本前提之一。这意味着，企业一方面要从内部打破各种部门墙和系统孤岛，另一方面还要对外建立连接，实现全方位的打通。

“比如，对于中小型制造业，他们通常分布在比较小的领域端，他们比较迫切的需求是，如何成为大型企业的产业链条中最核心的一环，通过与大型企业的绑定，去优化自己的生产制造工艺。尤其是在这两年，全球供应链体系被彻底打破，变得越来越碎片化，供应链的重建就成了很大的挑战。”沈旸举例。

落地到技术层面，这表现为越来越多的企业开始重视平台化建设运营。

孔云飞指出，一些成熟玩家开始崇尚云计算、端到端超自动化，实现应用云服务化、全链路自动化，大幅提升业务运转。更有高阶玩家已经着手投资云平台、数据中台、机器学习等技术能力的探索。

金蝶中国平台解决方案部总监柳忠阳也表示，如今很多中大型企业开始关注“平台”，尽管大家并没有特别想明白平台具体是什么样的，但会开始强调一些平台底座需求，希望构建起自身的平台能力。其认为，当下行业云平台可以站在客户的角度去提供资源和产品的组合。

今年，Gartner 发布了企业机构在 2023 年需要探索的十大战略技术趋势。其中，行业云平台被列入十大战略技术趋势之一。在 Gartner 的定义中，行业云平台是通过组合 SaaS、平台即服务（PaaS）和基础设施即服务（IaaS）提供支持行业应用场景的行业模块化能力。

企业可以将行业云平台的打包功能作为基础模块，组合成独特、差异化的数字业务项目，在提高敏捷性、推动创新和缩短产品上市时间的同时避免单一厂商锁定。Gartner 预测，到 2027 年，超过 50%的企业将使用行业云平台来加速他们的业务项目。

在我们看来，本质上，行业云平台也理解为更广义的行业解决方案。原先，行业云在业内争议较大，阿里云总裁胡晓明早些年曾在采访中谈到：“如果从基础设施 IaaS 层面看，行业云是一个伪命题。”

但经过这几年的发展，具有行业属性的云，越来越备受关注。Forrester《2022 年中

国行业云系列报告》指出，金融服务、制造、政府、零售和医疗等行业用户正在加速行业云应用，一些行业云先锋案例引领了同行云实践。

Forrester 将中国行业云厂商生态分为 7 部分：公有行业云平台、私有行业云平台、行业从业者、SaaS 行业云应用、行业专家、领域技术推动者和 IT 服务，能力覆盖行业用户场景的底层基础设施、开发平台和应用服务三个技术栈。报告认为，行业云对于不同垂直行业的企业加速数字化转型和适应未来发展至关重要，但是没有一个通用的行业云解决方案可以解决一个企业的所有业务需求。对此，其建议中国企业的技术高管在实施行业云的过程中采取务实的态度并建立正确的认知。

此外，从架构层面来看，由于能够支持企业工作负载在内部环境、私有云和多云环境之间的迁移，以及在不同供应商的云环境之间按需迁移，让企业实现数据的无缝衔接，混合云平台也成为越来越多企业的优选项。

加速智能化升级

数智战略离不开智能化，“云”解决了如何开始计算的问题，效率的问题，“智能化”则解决如何把计算结果用得好的问题。

以智能制造为例，史喆介绍道，富士康有大量的生产运营场景，也有众多需要决策的点。“如何运营能降低成本，如何高效管理规格种类繁复的大量设备？如何把生产经营中的 Know-How 和决策以数字化的形式共享到全集团内？这些都是我们正在做的工作。”

柳忠阳也从客户方感知到，如何让数据和智能结合起来并融入业务场景，是如今被重

点关注的问题。

如今，不少企业已经实施智能解决方案，比如流程自动化、智能分析等等。对应的技术应用方面，自动化工具 RPA（机器人流程自动化）依然是热门主流选择，它是当今许多企业踏入数字化转型大门的重要技术工具和平台，充当着企业的辅助生产工具以及数字化劳动力等角色。根据德勤的一项调查，53%的组织已经开始实施 RPA。

RPA 的核心逻辑是从数据里能挖掘出有效信息，并迅速解决效率层面的问题。目前，行业内的 RPA 基本朝着智能化全场景服务的方向发展。

但如果要深度应用人工智能技术，则远不止这些。从行业来看，目前主要是高科技、电信、金融服务等行业对 AI 的应用比较深和广，比如金融服务类企业则将其应用于提升客户体验的相关职能。此外，许多汽车制造商也在运用人工智能技术开发无人驾驶汽车、改善企业运营。

据朱辉介绍，汽车行业近年发展的重点，已经从传统的造好车（“酒香不怕巷子深”）转变到更加提倡体验化、服务化、智能化、生态化。“未来的汽车就是一个软件能力的集成体，是一个人工智能能力的集成体，也是一个数据的产生和应用的集成体。在汽车制造以及汽车生态里面都正在发生一些颠覆性的变化。人工智能、混合云平台的技术在汽车行业当中是会发挥越来越大的作用。”

麦肯锡曾在一份报道中将人工智能视为数字化的下一个前沿，并指出人工智能技术对企业的数字化实力有一定要求，而且往往需要以特定数据进行训练，这就意味着企业没有捷径可走，只能加快数字化转型的步伐，包括人工智能技术的发展。先行企业现已逐渐形成竞争优势，将后来者甩得越来越远。

补齐人才短板

如果说技术是企业转型的外驱力，那么人才则是内生力。为全面推动数字化，人才的问题必须解决。

作为数字化转型的亲身实践者，郭华提出，最快最直接的数字化人才培养方式是“做项目”，训战结合。但这里面有一个非常重要的前提条件，即企业必须要建立起选择人的标准，基于标签、画像、数字化人才发展路径对所需的数字化人才进行规划与培养。

从技术服务商的角度来看，越来越多的科技企业也在近些年纷纷推出相应举措以加速人才培养。比如，华为在去年发布了“未来种子 2.0”计划，承诺在未来五年投入 1.5 亿美元用于数字化人才培养，新增受益人数预计将超过 300 万人。官方数据显示，截止目前，华为数字化人才发展服务已覆盖企业客户 1200 多家，累计为企业培养数字化人才近 5 万人。此外，腾讯教育也与 600 多所高校展开人才培养方面合作，累计为社会培养了超过 10 万名数字人才。

当然，无论是吸引人才抑或是培养人才，企业首先要搞清楚的是，自己缺的到底是哪类人才，才能有的放矢，有针对性地投入。过去，业内一般会把数字人才分为三类：数字管理人才、数字专业人才和数字化应用人才。而近年来，一些机构开始也对数字化人才做了更深层次的诠释。

以极客邦双数研究院今年发布的《数字人才发展体系：粮仓模型白皮书》为例，该白皮书将上述三类数字人才进一步细分为更加匹配企业架构的五层数字人才，分别是：

- 数字思维管理者：对应企业 CXO 层级。他们需要具备数字化思维，负责数字化战

略创新，是带领团队推进企业数字化变革的企业管理者；

- 数字思维业务人才：具备数据思维，掌握数字工具的使用技能，基于不同业务场景，善于利用新技术手段提高业务效率和价值的人才；
- 业务架构人才：将业务战略解码，将企业内部业务进行横向打通的复合型架构人才；
- 技术架构人才：能力横跨多个技术领域，通过技术架构设计和实施来支撑业务的复合型架构人才；
- 专项技术人才：掌握软件技术或者新兴技术，并能够创造应用工具的人才。

以上五层数字人才粮仓模型，从人才分类、所需能力以及人才占比等维度给企业提供了建立数字化转型人才战略的思路，帮助企业先识别人才，再补齐短板。

总结

近几年，我们可以明显看到，疫情促使企业加快数字化、智能化升级。最初，数字化转型升级对多数企业来说是权宜之计，主要转型场景围绕数字化办公、消费场景数字化、业务流程进一步线上化等展开。

虽然受到 2022 年的经济形势影响，企业的“口袋”捂得更加严实，更注重业务价值回报，更看重各种技术的投入产出比，以及数字化能够带来的经济价值。但是，展望 2023 年，数字化依旧势不可挡。

数字化的范围逐渐从单个企业的数字化转向产业性、地域性的数字化，智慧供应链、产业互联网、智慧城市、无人驾驶等将成为新的技术集聚热点。放眼全球，借助云计

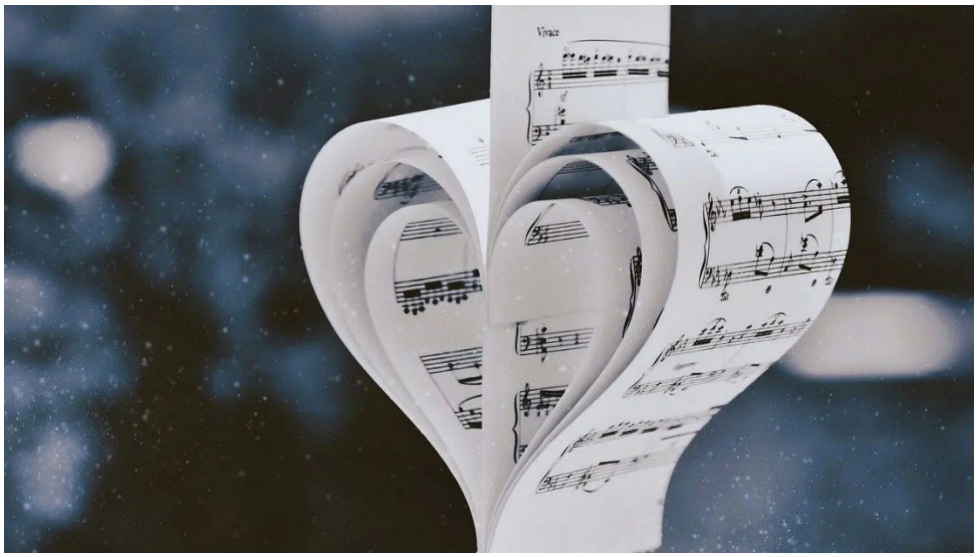
算等数字化技术加快出海，构建全球化品牌，将成为众多领先企业新的增长引擎。而为了满足全球监管合规要求，全面的安全策略和生态系统将成为标配。

除此之外，为落实可持续发展目标，如何利用数字化技术实现低碳、绿色、降低能耗，实现高质量的可持续发展，也必将成为企业数字化的重要议题。

可见，如今数字化转型的意义已远超当初的考量，它可以帮助企业寻找新的发展机遇，甚至决定了企业发展的命脉，在充满不确定性和不断变化的大环境下，觅得新生机。

直面成本“刺客”、拒绝繁杂技术花样，压力之下云厂商改变方向 | 解读云原生的 2022

作者：褚杏娟



本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦云原生领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年云原生领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

特此感谢丁宇（叔同）、董晓聪、裴超、冯常健、柯琪、吕亚霖、王泽锋、于广游

> （按姓名首字母排序）对本文的贡献，他们的真知灼见，是本文能与大家见面的关键。

数据显示，云原生计算基金会（CNCF）拥有超过 850 家参与组织，相较去年增长 15%，其中今年新增的 220 多个成员中，有 19 个来自中国。

去年的盘点中，我们用了“抢占技术 C 位，迎来落地大爆发”的标题来总结云原生整体的发展势头。今年，云原生的发展依然可圈可点，比如开源容器编排平台采用率激增、各种行业加入到了云原生大家庭等。这次，我们将从底层基础技术和偏向场景化的应用技术两方面，对云原生今年的发展情况进行盘点，以期自顶向下地呈现出相关亮点和挑战：

- 容器的“黑盒”打开，混部带来了效率提升，备受企业欢迎；
- Serverless 基于容器完成标准化，应用“元年”开启；
- Service Mesh 进行新尝试，落地方式还在探索；
- 降本增效大主题下，FinOps 理念得到快速发展；
- 越来越多的传统行业开始应用云原生技术等。

基础技术篇

从物理机到虚拟机再到云原生，底层基础技术对上层应用赋能的边界在不断提升，业务研发越来越从底层事物中抽离，聚焦于本身的业务逻辑。

容器的“黑盒”已经打开

- 7 月，Kubernetes 发布了 Gateway API 0.5 版本，主要组件的 Gateway API 资源首次升级为 Beta 版。同时社区还探索如何将 Gateway API 用于网格并引入新的实验概念。这也促成了 Envoy Gateway 项目的推出。
- 12 月，Kubernetes 原生工具集合项目 Argo 正式从 CNCF 毕业。

容器发展至今，已经完成了自身的蜕变。

在第一阶段，容器只是在解决如何做好自己内部技术组合的问题，比如怎么与服务网格、可观测性的产品结合。这期间，企业只是选一部分比较周边的业务使用容器，而容器化业务内部系统就像一个“黑盒子”，各细分组件通过 ELB 调用实现交互，容器并不能直接与每一个部分做精细互通，致使容器化业务系统跟周边业务系统的交互效果很不理想。

到了第二阶段，“黑盒子”打开，容器开始深入到整个云原生体系中，真正作为云最核心的部分，与网络、存储、数据库等做很好的结合，比如容器化业务应用可以直接与其他所有云服务通信等。

作为云原生的典型代表，如今容器技术的成熟度是最高的，同时这也意味着容器技术自身很难再有非常大的突破和变化。Kubernetes 成为行业事实标准，如今定位也类似 Linux 内核，虽然没有大的功能性迭代，但已经成为云原生领域最基本的设施，也成为企业 IT 系统的基本设施。

今年，容器的演化进入了“扫尾阶段”，开始解决原来不好做或不能做的场景应用问题。Kubernetes 这一年稳定更新的三个版本，更多是在 API 怎么更加健壮、流控怎

么更好更安全、改进 Windows 支持等细节上做优化。

另外，内核提升可以为容器带来更好的隔离和性能等，因此随着容器的广泛应用，其对内核的要求也越来越高。大厂们开始往更高内核版本 5.10 上迁，Kubernetes 社区也在落地内核方面的内容，给企业带来的直接影响就是观测性上有了很大的进步，性能和隔离性的提高也很明显。

应用上，虽然容器正在大规模铺开，但如何用好容器也很有挑战，比如 Kubernetes 在调度、场景优化、降低复杂性等方面还有很大的进步空间。

微软全球副总裁柯琪表示，随着容器的广泛应用，企业部署了很多 Kubernetes 集群，如何对其管理是企业面临的问题之一。鉴于这部分与多云有部分重合，我们留到多云部分再表。

混部

今年，容器混部取得了较大的进展，不过这个进展并不是体现在技术上。之前混部主要是各大头部企业内部使用，对于技术细节有些“秘而不宣”的意味，但今年云厂商开始把混部技术开源或者变成云产品提供给用户。

混部就是要把各种不同的业务放在一起后对它们进行复用。听着简单，但混部有着较高的技术门槛，主要体现在调度、隔离和干扰处理上。

具体地，Linux 操作系统内核的隔离性做得不好，但 Linux 内核面向通用场景，如果要解决隔离问题就要对内核做非常深度的改造，但国内大部分企业并没有独立的内核团队，况且这样的人才也是凤毛麟角。当然，不做内核隔离只做调度混部也可以，但

对资源利用率的提高就很有限。

另外，合理的调度需要有全面的应用画像，这需要大量的数据和业务去打磨算法，这在实现上也非常困难。调度实际上是一个系统性工程，并没有标准化方案，企业可以自行摸索初级方案，但更深入的可能就要使用市面上的产品。

其实，企业目的不是混部，而是为了提高资源利用率、降低成本。降低成本有两种方式：一是企业已经买好资源，内部不同业务复用这些资源，这条路线的代表就是混部；二是企业直接使用云厂商提供的成本很低的产品，这条路线的代表是 Serverless，也是大家比较好看的方向。

可以预见，下一年，Kubernetes 的演化依旧以解决场景化的细节问题为主。腾讯云容器技术总监、TKE 产品负责人于广游总结道，整体上，容器未来将呈现出以下趋势：

第一，往更高层的抽象演进。最早用容器的时候，大家关心的是容器中的资源、pod、container，但是业务并不关心这些，他们真正在意的是应用。因此，容器平台向应用平台演进会是一个趋势。

第二，不断拓应用宽度。Kubernetes 的应用已经延伸到了混合云、边缘云，在对中心、边缘、IDC 的资源进行统一调度等方面，还有技术挑战。

第三，多样化应用。企业会逐渐通过采用容器技术重塑自己的业务和全部技术栈，比如监控、日志、中间件、数据库、运维等。容器带来了好处，但它跟企业之前的技术会产生割裂，这时更多的企业会选择将所有工作负载运行到容器上，容器化间接成为企业重塑技术栈的动力。

结果就是，企业业务容器化后，还会把有状态应用、AI 业务、大数据业务、转码业务、离线业务等都容器化。所以现在的容器不仅仅与微服务挂钩，还在朝大数据、AI 等方向发展，这个趋势是非常明显的。

第四，更下沉的方向。对于平台自身而言，如何更稳定、成本更低？还有更多的细节亟待完善。

Serverless：与容器技术体系相通

- 3 月，开源 Serverless 应用框架 Knative 成为 CNCF 孵化项目。谷歌此前曾明确 Knative 不捐给任何基金会，但去年底宣布捐赠到今年 Knative 正式成为 CNCF 孵化项目，极大促进了社区的发展。
- 5 月，开源 FaaS 项目 OpenFunction 成为 CNCF 的沙箱项目。
- 9 月，Serverless Devs 进入 CNCF 沙箱，这也是 CNCF 首个 Serverless 工具项目。

当前用户对云服务的使用主要有两种方式。一种是把云服务作为资源使用，比如云服务器、容器等，它们不会影响用户应用的开发方式，这类就是容器化 Serverless；另一种是把云服务作为应用构建的模块，例如对象存储，消息队列等服务，用户在应用开发过程中使用这些云服务，但它们会影响用户构建应用的方式，这就是应用层 Serverless。

传统意义上的 Serverless，也被称为 FaaS，即函数形态的 Serverless。2014 年，亚马逊推出了 Amazon Lambda，其将 FaaS 理念延伸到数据库、中间件等产品，让各种应用场景下的用户都不用关心资源和集群，而是直接使用 API，这是业内公认最早

的 Serverless 服务。

FaaS 的核心价值在于让整个云产品体系及生态形成一个有机整体，而不是只用来提供弹性资源。阿里巴巴研究员、阿里云智能云原生应用平台总经理丁宇（叔同）指出，用户可以利用 FaaS 组合其他 Serverless 云服务（BaaS）快速构建应用，这种组装式的研发模式将对研发效率带来革命性变化，是云解决大规模复杂软件研发挑战的关键。

另一方面，由于单纯的 FaaS Serverless 是基于函数的，其开发模式、技术体系、产品形态与容器并不相同，不能为用户提供“原汁原味”的 Kubernetes API。该方式下，用户用提交的一个代码片断进行托管，各厂商自成体系，业内没有统一的标准，短期内也很难看到有事实标准诞生。因此在于广游看来，容器的大规模流行反而暂时抑制了 FaaS Serverless 的发展，因此落地也相对缓慢。

目前，业内对于 FaaS Serverless 的发展还存在不同的看法和方向，现在也没有孰优孰劣的定论。

作为应用层面的 Serverless，以 FaaS 为核心的 Serverless 体系更加强调 Serverless BaaS 的丰富度和 BaaS 之间的深度集成，用户上手快，但需要花费时间和精力去打磨整个云产品体系。但 FaaS Serverless 的核心价值依然是被肯定的，像亚马逊云科技和阿里云还在继续这方面的深研，用更被接受的方式发挥其价值。

不过当前的实际生产中，不要求用户改变研发方式的容器化 Serverless 获得了更多的青睐，很多企业多少都了有一定的应用。

实际上，亚马逊云科技首先提出 FaaS Serverless 后，其他云厂商纷纷跟进。但单纯

的 FaaS Serverless 由于对云产品要求高等原因，在国内的接受度不如海外。这种情况下，业内也开始探索其他方式。

之前，大家把 Serverless 作为一种具体的实现，而现在将其视为一种理念和价值，并尝试为这种理念和价值找到更合理的实现方式，容器化 Serverless 就是这种转变下的解决方案之一。

容器化 Serverless 发展的目标是为了兼容原本容器的技术体系、降低企业迁移的成本。容器化 Serverless 将 Serverless 与容器结合，Serverless 技术兼容容器的 API，两个技术体系打通后也意味着容器化 Serverless 是标准的。因此，各云厂商的产品标准在趋于一致，能力也在逐步完善。在这个前提下，企业服务的迁移会更容易。

当然，容器化 Serverless 不同厂商之间的产品策略也有差异。比如，微软的 Container Apps 支持 Dapr 分布式应用环境，运行在基于 AKS 的隐藏的抽象 Kubernetes 集群之上，甚至不向用户公开 Kubernetes API，阿里云则走在提供 Serverless 容器服务 ASK 的道路上。

现状及趋势

Serverless 的应用如今实现了“从点到面”的转变。

现在，Serverless 的稳定性和效率已经被逐渐接受，一些企业之前只是在周边的个别业务上牛刀小试，现在开始将部分核心业务放到 Serverless 上。比如阿里云完成核心云产品全面 Serverless 化，作业帮将某部分核心业务放到 Serverless 后成本节省了 20%-30%，高德业务投放平台全面采用基于函数计算构建的 Serverless 架构来支撑

百万 QPS 等。

今年，很多云厂商在各自大会上都提到了 Serverless 的重要性，比如阿里云栖、亚马逊 re:Invent、腾讯数字生态大会等，但热闹的背后反而说明了 Serverless 还处于发展初期，需要推新和推广。这一年，国内 Serverless 被认为是发展元年，虽然还称不上是大规模落地，但业内在 Serverless 成为云原生未来重点方向这一点上几乎没有太多分歧。

Serverless 适用于流量波动大和需要更敏捷、更灵活开发的场景。现在 Serverless 架构可以解决微服务架构无法带来足够灵活度的问题，帮助企业快速适应业务变化。

Serverless 为能力不同的开发者抹平了技术鸿沟。如果将应用分为应用运行时（即计算部分）和 BaaS 化服务两部分，那么 Serverless 化后的架构会变得更简单。对于 BaaS 服务，可以进行全面托管并免运维 API 化，开发不需要关心资源等问题；对于定制代码部分则可以把 BaaS 的 API 组装起来，让应用开发更快建出新能力，同时不需要关心容量问题。

成本方面，对于企业为波峰预置服务器产生浪费的费用、业务需要扩容时购买机器花费的成本和精力、机器维护成本等，Serverless 都可以将这些交给程序运行，按需使用资源、动态伸缩，为企业带来降本增效的效果。

Serverless 的快速发展也会对运维领域产生一次颠覆性升级，工程师将精力更多放在开发降本等平台上，而不再关注底层的扩容、采购、网络等问题。

预计下一年，会有更多产品具备 Serverless 能力，Serverless 会变得更加普适。另外，Serverless 产品的更多细节也会进一步完善，比如如何提高产品弹性、如何进一

步降低成本（如从之前的包年包月到按 CPU 利用率的计费模式）、功能如何更加丰富等。

应用上，以上两种方式虽是业内对 Serverless 应用的不同探索方向，但都各具优势和不足。业内现在有将 FaaS Serverless 和容器化 Serverless 结合起来提供解决方案的尝试：容器化 Serverles 解决资源层面的问题，应用层面的问题让 FaaS + Serverless Baas 解决。用户将不同 Serverless 化产品通过事件驱动等方式深度集成后，通过 FaaS 组合其他云服务来快速实现弹性、高可用。

当然，Serverless 虽被认为是大势，甚至被视为云未来发展最重要的价值，但从趋势到产品、到行业普遍应用，再到真正大规模落地，还需要业内的长期投入和推动。

Service Mesh：还有挑战需要解决

- 4 月，谷歌声明将 Istio 捐赠给 CNCF，9 月份 Istio 正式成为 CNCF 孵化项目。这一事件使 CNCF 社区的确定性更强，也消除了前些年大家对社区治理、法规等方面的顾虑。
- 5 月，Envoy Gateway 项目宣布开源。该项目旨在大幅降低将 Envoy 作为 API 网关的使用门槛。
- 9 月，Istio 宣布引入了一种新的数据平面模式 Ambient Mesh，该模式取消了以 sidecar 为中心的架构，取而代之的是无 sidecar 的方法，同时保留了 Istio 的零信任安全、遥测和流量管理的核心功能。

很多企业都是多协议、多语言栈的，他们选择使用 Service Mesh 来解决复杂的服务治理问题。Service Mesh 理念本质上是把一些非功能性的基础设施拆解到中间件中，

即 sidecar。

在之前的一些实践取得正反馈后，Service Mesh 使用范围也在扩大。今年的 Service Mesh 不再局限于 RPC，开始向对象存储、加解密、MySQL、Redis 等领域深入。

但总体看，这一年，Service Mesh 落地还是遇到了大的技术挑战，远没有达到企业理想的使用状态。有一定研发能力的企业使用传统治理模式也可以做得不错，这时就不会选择完全换成 Mesh 架构，只会在一些新的、没有历史负担的业务上试用。

归根到底，Service Mesh 只是转移了复杂度，但到了一定规模后复杂度问题就会再次显现。拿当前业内应用较多的 sidecar 模式来说，它很适用于逻辑复杂的场景，如路由、治理，灵活且对业务无入侵。但是，流量劫持和对流量进行逻辑处理需要很强的扩展能力，在规模特别大的场景下，sidecar 模式的复杂度就上来了，性能优势不再明显，资源占用也变得不可忽略。可以说，sidecar 模式天生在大规模场景应用中就有一定的局限性。

为解决这个问题，今年九月，Istio 推出了 Sidecarless 的 Ambient Mesh。Ambient 是将 Istio 分成两个不同的层次：安全覆盖层（四层治理）和七层处理层（七层治理）。但在网易数帆云原生平台负责人冯常健看来，四层治理模式将复杂度降到了 Node 级别，但可能只有对网格安全能力感兴趣的企业会尝试，而七层治理模式本质上还是独立的应用层代理，链路也并未减少。因此，对于该模式的应用，业内更多还是持观望态度。

网关层面，社区基本分为 NGINX 和 Envoy 两派：Kong、APISIX 等基于 NGINX，网易、阿里云等更多应用 Envoy 技术栈。有人认为 NGINX 及其生态已经比较成熟了，

但随着 Kubernetes Gateway API 的成熟，以及社区推出 Envoy Gateway 组件，新一轮网关标准定义的争论再次掀起。

Kubernetes Gateway API 对标的是 Ingress API。Ingress 的 API 解决流量从集群外导入集群内的问题，但表达能力较弱，使用场景有限，因此社区推出了 Kubernetes Gateway API，希望其提供更高级的网络能力。

Kubernetes Gateway API 直接促进了 Envoy Gateway 项目的发展。Envoy Gateway 进而统一了网关的控制面 API。原先网关控制面是通过 xDS 控制数据面，现在更多会基于 Kubernetes Gateway API。

实际上，现在各个企业都在从不同的方向尝试对 Service Mesh 进行完善和补充，如网易向 Envoy 社区贡献 Dubbo 核心支持能力来促进落地。虽然社区有了各种开源产品，但业内还没有形成像 Kubernetes 这样的事实标准。当有这样的一个事实标准出来后，Service Mesh 才会迎来自己的爆发。这与容器的发展轨迹是类似的。

Service Mesh 也在寻找更适合的落地方式。现在，业内有尝试不再将 Service Mesh 作为一个独立的产品，而是将其与 Serverless 结合。Serverless 不让用户去关心服务器，Service Mesh 不让用户关心服务治理，如果将服务治理的 Service Mesh 容器内置到 Serverless 平台里面，企业提交一个业务的容器进项后也会拥有 Serverless 的能力。

硬件与云，互相影响

- 6 月，阿里云发布云数据中心专用处理器 CIPU，形成“倚天+飞天+CIPU”组合。

- 11 月，亚马逊发布 Nitro v5、Graviton 3E 系列芯片。
- 12 月，腾讯云发布了新代次的裸金属云服务器、GPU 云服务器，此外还发布了银杉智能网卡等。
- 同月有消息称，微软以 1.9 亿美元（约 13.2 亿人民币）的价格收购了加州 DPU 初创公司 Fungible。

从整个计算机发展历史看，软硬件迭代呈现出了交替螺旋的方式，即某个新技术出来后，都先用软件测试可行再进行小规模应用，当发展规模大到一定程度时，人们便开始考虑用硬件方式加速实现，云原生也不例外。

如今，云已经取代传统 IDC 服务器成为新的基础设施，企业开始通过硬件进一步提高效能。企业选择云原生硬件的考量主要有两点：成本和不同应用上的性能表现。

成本方面，虽然硬件开发的一次性投入相对较大，灵活性相较软件也差很多，但只要需求固化下来并开发成功后，企业采购的边际成本是非常低的。

性能方面，硬件对云的性能提升包括可靠性、稳定性、安全性等方面：

- 热操作能力。与传统业务不同，云是不能停机的，热操作能力，包括热升级、热迁移、热插拔等都是云提出的特殊要求，而传统硬件并不具备这些能力。现在，云计算领域里的自研硬件会在设计要求有热升级等功能，像 DPU 里任何一个组件都有热升级能力。
- 租户隔离。云的特性之一是弹性共享，一个物理机上可能会有很多用户在运行虚拟机，怎么加强用户之间的隔离、避免互相影响，也是云对硬件提出的新要求。
- 安全性。云上很多用户运行在一台物理机、一个网络里，如果某个用户恶意利用

一些漏洞获取到其他用户的数据或者逃逸到云的平台层，那么会产生非常严重的后果。因此，云对硬件的安全性要求也非常高。传统 IDC 对带内、带外并没有做非常强的隔离，但在云上就不允许互通，这也是调整改造的方向之一。

当前云原生领域的硬件设备可以分为通用型和专用型两类，前者如 CPU，后者有 GPU、DPU。DPU 是专门为云而生的。传统情况下，服务器的 IO 更多是通过网卡、硬盘等硬件直接实现。但在云原生领域，网卡、硬盘等都是虚拟的，用户把东西发到网络或写在硬盘的动作实际上是先交给了云平台侧的软件进行加工处理后再发到网络或写入硬盘。

腾讯云高性能网络产品中心总监裴超表示，云和硬件也在彼此影响着向前发展。一方面，随着云原生应用的需求不断提高，硬件设备为了满足这些需求也在快速迭代。比如虚拟机网卡从 20G、50G，甚至迈向 100G，云原生对硬件的蓬勃发展起到了一定的刺激作用。另一方面，很多云原生软件产品的设计之初，就会有硬件专家参与进来，一起考虑未来用硬件加速的可能性。可以预计，未来的架构师、技术负责人多少都需要对硬件有所了解，以便对总体架构进行把握。

目前，云厂商已经越来越多地参与到硬件和芯片的研发当中。相较之前，一款芯片可以用三四年，现在基本保持一年一更新的频率，迭代可以算是频繁。像英特尔、AMD 等垂直芯片厂商也在努力直接了解终端客户需求，尽快推出新的芯片。

产业落地篇

行业主题：降本增效

“降本增效”无疑是今年行业的大主题。在这个背景下，整个行业不再像之前那样去追逐前沿技术，而是回归到了云的最基本价值上。这一变化也可以从今年云厂商讲故事的角度上看出来。往年，云厂商会争相发布各种技术，努力往技术前沿方向走。但今年，云厂商追求的是“do more with less"，即如何用更少的成本帮助客户获得更多的收益。

FinOps

谈起降本增效，之前企业常常在这年做完后不知道下一年该怎么做，本质上是缺乏方向性的指导。正因如此，这也促进了今年指导云财务管理的 FinOps 理念的快速发展。

FinOps 的历史并不悠久，公有云早期传播者 Adobe 和 Intuit 在 2012 年首次描绘出了 FinOps 的雏形，直到数年后云成本问题日益严重才逐渐崭露头角，更多企业也是今年才刚刚接触。

成本管理的难点在于资源并不是独享的，而是共享。通常单个集群可以托管多个工作负载和应用，但云厂商的账单无法体现每个工作负载或应用消耗的资源。缺乏对多个团队如何利用或共享基础设施的可见性造成了成本“黑洞”。

FinOps 本质上是把财务和整个架构技术结合在一起，弄清楚各业务对云服务使用的具体账目，然后把资源利用率提上来，减少成本消耗。该理念要求企业首先要清楚自

己具体拥有哪些资产、这些资产属于谁；其次是看这些资产的利用率怎么样，然后再结合相关的技术进行优化。

实践中，企业的架构团队搭建相应平台将成本可视化，并在 DevOps 里加入管控审批等流程，用流程去约束成本。另外企业还要引入一个“成本负责人”的角色，这个角色主要由业务部门承担，负责掌握每个部门申请资源的原因、能否创造足够的利润覆盖这部分成本、资源使用率低的话是否减配等，具体执行时则主要由 SRE 把控。

FinOps 相关产品之前更多是多云管理的商业化公司推出，今年云厂商也有陆续推出，如阿里云的 ACK FinOps 套件。云厂商有很多运行时数据，如闲置率、平均负载等，也可以为企业提供优化建议。但企业如果想真正控制成本，更多还是要自己去做。

现在也有一些 FinOps 开源项目，如腾讯云开源的 Crane 等，这些项目本身就是一个成本管理平台，会把涉及到的成本信息汇集起来做出趋势图等，帮助企业做决策。但业内目前没有完全的开源实施标准，也很难做出一个大统一的产品，因为这样的产品需要考虑的因素是方方面面的，现在只是每个细节和分支有产品雏形。

作业帮基础架构负责人董晓聪认为，FinOps 的理论体系会逐渐沉淀和完善，各企业的落地方式相差无几，不过使用的具体产品可能不同。事实上，FinOps 可能目前只要满足企业的特定需求就好，不是非要演化出这样的标准。

FinOps 一定程度上会倒逼企业进行整个技术架构的演进，促进企业用更好的技术、更好的生产力工具，或者是更少的服务去支撑更大的服务。一定程度上，FinOps 会成为撬动企业增长的一个杠杆。

当然，FinOps 不是万能的，因为“降本增效”也有成本。对于成本已成为不可承受之重、

之前也没有进行过优化的企业，里面一定存在大量的技术浪费，这时候就需要花费人力优化，企业肯定能从中享受到优化红利。但如果一家企业处于高速发展阶段，有能力赚取非常高的利润去完全覆盖浪费的成本，那么肯定业务为王，暂时还不必在降本上花费很多精力。

如今，FinOps 还处在大家都很需要、但没有做得特别好的阶段。下一年，FinOps 的趋势会延续，甚至未来几年内都会是很多企业关注的重点，其应用的领域也不会局限在互联网。

多云

很多企业前几年就开始考虑多云架构，直到今年开始真正在生产落实。这背后的原因也是企业对成本的考虑，还有就是对稳定性需求的提升。

实际上，国内外各大云厂商都出现过大大小小的故障，很多情况下企业要依靠多云架构去弥补这部分损失。另外，只要选择了两家云厂商，选择权就到了企业手里，企业与云厂商谈判的议价关键取决于企业在云之间迁移的能力。当然，企业如果实践不好，反而会增加成本。

总之，多云要做的就是云间的闭环和自由迁移，以及能够做得更高可用，抵销对云厂商的依赖。

在多云的初级阶段，不同云上的应用管理是手动的，业务量多的时候管理员会非常痛苦。之后，多云才有了一定的自动调度和差异化配置能力，业务不用感知资源池细节和授权等问题，这样的资源池化能力正是企业需要的。

现在社区比较流行的开源集群管理产品是 Cluster API，该产品使用 Kubernetes 风格的 API 和模式自动化集群生命周期管理，谷歌、微软等云厂商都有基于 Cluster API 的服务。华为云等多家企业联合发布并开源的 Karmada，是 CNCF 首个多云容器编排平台项目。今年，Kubernetes 多集群管理平台 OCM v0.9.0 发布，来进一步改善托管集群安全性问题。实际上，虽然每个云厂商都在自己推出产品，但很大程度上也是基于上游社区的建设。

不过，华为云云原生开源负责人、CNCF 大使王泽锋也指出，现在多云、多集群的编排调度等问题还没有特别成熟的商用级解决方案。

首先是多集群的负载管理。一些负载不能放到一个集群里，而是需要建立一个容错性更强的集群体系，这个系统通常由多个 Kubernetes 集群组成。多个集群间如何部署应用、主副关系还是平等，很多细节问题都要考虑。

其次是多集群的流量管理。单集群内的流量管理，一定程度上是通过控制面来维护各个实例的健康状态、调整流量分布。但是到了多集群架构下，由于单集群自治属性强，控制面与各个单集群之间的联系减弱，并不能简单地通过两者的联系来判定哪个集群无法处理流量、需要重分配。如何实现数据面自治的多集群流量管理，是个很大的挑战。

再者，业务部署在不同的云或集群里，业务互相访问时会有网络连通性、身份授权等问题。多云的场景是多样化的，很多用户不会接受拉昂贵的专线方式，网关方式又比较适合允许有延迟、对网络可靠性要求不高的业务场景。身份授权更为棘手，多云、多集群本就是天然屏障，如何身份授权进行全局管理就是一个很大的难点。

最后，实现多云、多集群的互访后，业务还需要有较强的适应性，能够在不同的模式下工作。

越来越多传统行业入场

除了降本增效，今年很多企业也纷纷投入到了数字化转型当中。实践中，转型企业是用互联网技术和互联网思维方式构建自己新的数字化体系。数字化中，很多企业首选云原生技术。

今年，除了有像腾讯这样业务完成全面上云的互联网企业，还有越来越多的传统企业开始上云。工业制造行业围绕业务需要主动上云，云厂商也会根据其业务需求提供配套解决方案，金融、零售、政务、能源、物流等行业也都入局云原生。

根据云原生技术实践联盟（CNBPA）调查，95%以上的受访央国企已经考虑或正在使用云原生技术，其中近 80%的企业已经使用容器、微服务、DevOps 等云原生技术。支持一云多芯的全栈云原生是更贴近业务的技术应用架构，更能聚焦央国企的数字化转型。

传统企业使用云原生技术更多会从 to C 业务入手，比如个税办理等面向个体消费者的 App。当然还有像光伏行业通过“云原生+AI”技术提升良品率类的应用。另外，传统企业在新建或升级系统时，也会优先选择更流行和最有效的技术，而云原生就在其中。

非互联网企业也积极拥抱开源，并参与云原生社区，比如波音今年加入了 CNCF 成为白金会员，中国电信、中国电子云成为 CNCF 黄金会员。

在今年 3 月，波音宣布与国外三大云计算提供商亚马逊云科技、微软和谷歌，签署了协议，扩大云运营规模并减少对本地系统的依赖。波音大部分应用都是在本地服务器上托管和维护的，很多遗留系统正在老化，需要大量维护工作，阻碍了波音开发和部署新型数字化解决方案。波音选择将大部分工作负载转移到云端的方式消除基础设施带来的限制。

像波音这样的非互联网企业进入云原生领域，是该技术更大规模落地的预兆之一。

云安全

或受去年 Log4j 漏洞事件的影响，加上 Kubernetes 安全事件频发，今年业内在安全方面的实际行动也很多，尤其是在供应链安全和模糊测试上。

供应链安全更靠近企业实践。之前，企业为了安全直接做物理隔离，但由此也带来了很多问题。当前阶段，企业做的比较多的是扫描检测部署的业务应用对应的版本，对于有问题的版本进行拦截，不允许其部署到生产环境中，初步保证业务的生产系统不暴露在漏洞之下。

另外，云原生给安全提供了一个新的抓手。之前的 DevOps 体系里融进安全的东西很难，而云原生的 DevOps 体系下，企业可以结合 CI/CD 技术，从软件的编码构建到测试、部署和运维的整个链路中，加入面向代码级的扫描能力、面向依赖的组件做版本的分析，并与漏洞库进行匹配，如果 CI/CD 部署对接到多云上，中间还会做防篡改等设计。

不过，作业帮基础架构-架构研发团队负责人吕亚霖表示，云原生也打破了安全原有

的物理边界，这对安全领域也是场考验。

模糊测试是一种常用的应用程序安全测试，主要用来发现应用中的功能性缺陷、安全问题等。当前，大家对模糊测试的使用更偏单点技术。CNCF 中的多个项目都在使用模糊测试，如 Argo、Envoy、Fluent-bit、Kubernetes 等。这一年，Ada Logics 的团队持续努力将模糊测试集成到 Kubernetes Cluster API 项目中。

今年，国外云厂商在安全方面也做了很多事情，比如微软发布的 Azure AD workload identity 预览版可以在 Pod 级别进行身份托管，还宣布实现在 Kubernetes 中支持 AMD SEV-SNP，以此保护数据安全；亚马逊云科技扩展了 Amazon GuardDuty 安全监控服务的范围，增加了对部署容器的运行时环境的威胁检测。另外，很多创业公司也在做相关工作，比如五名前谷歌员工共同创建了专注开源供应链安全的 Chainguard。

云原生安全相关项目

但在下一年，行业是否能延续对安全的重视其实还是未知的。实际上，还有很多除漏洞外的安全问题。比如多云和边缘都会加倍放大安全问题，多云里的资源访问权限和身份授权就是很大的问题，边缘很多是离线场景但又不是绝对的物理安全。能否继续

重视安全，取决于人们的意识。

写在最后

今年在互联网历史上并不是高速发展的一年，甚至全球经济下行给行业发展蒙上了一层“寒霜”。但反过来，这样的环境也促进了行业不去关注各种花样繁多的新技术，而是朝着更好、更深度的方向发展，这对云来说是一种利好。

这一年，我们看到有些企业出于各种考虑开始“下云”，有人认为正常，也有人认为这是“开倒车”，而新上云的企业也要面对走出舒适圈带来的阵痛。实际上，上了云和能用好云是两码事，还有各种规模的企业坚持在云上运行，现在大家都还远没有触碰到云效益的天花板。

总体来看，明年，云原生领域的主旋律依旧是价值回归，业内围绕着切实可行的技术去落地、深入实践降本增效，变得更加务实。不够成熟的技术会努力找到降低门槛、找到被开发者接受的方式，相对成熟的技术继续深挖，给开发者创造更大的价值。

采访嘉宾介绍（按姓名首字母排序）

丁宇（叔同），阿里巴巴研究员、阿里云智能云原生应用平台总经理

董晓聪，作业帮基础架构负责人

裴超，腾讯云高性能网络产品中心总监

冯常健，网易数帆云原生平台负责人

柯琪，微软全球副总裁

吕亚霖，作业帮基础架构-架构研发团队负责人

王泽锋，华为云云原生开源负责人、CNCF 大使

于广游，腾讯云容器技术总监、TKE 产品负责人

市场增速超 20%，国产操作系统“浴火重生”| 解读操作系统的 2022

作者：凌敏

本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦操作系统领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年操作系统领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

特此感谢郭振宇、江大勇、刘恺、马涛、任革林、张磊、张家驹（按姓名首字母排序）对本文的贡献，他们的真知灼见，是本文能与大家见面的关键。

在整个软件生态中，操作系统起到重要的承上启下作用。经过数十年的持续迭代和演进，操作系统整体发展稳健。从技术角度来看，虽然近几年并未涌现突破性成果，但不少受访专家对 InfoQ 表示，巨变正在酝酿中。当前，全球数字经济进一步发展，企业数字化转型持续深入，以及摩尔定律的失效，都为软硬件带来新的挑战，而这也是操作系统变革的重要驱动力。

回顾操作系统的 2022 年，可以发现，有些改变正在悄然发生。随着“量变”的积累，操作系统必将迎来“质变”的飞跃。

2022 年，值得关注的大事件

2022 年，全球操作系统市场格局稳定。看国外主流操作系统，桌面操作系统方面并未带来太多惊喜。与去年重磅发布的 Windows 11、Windows 365 相比，Windows 今年仅带来了一些小范围的更新。

移动操作系统方面，Android 13 与 iOS 16 相继登场，但也都属于常规升级。2 月 11 日，谷歌发布了首个 Android 13 开发者预览版，8 月 16 日，谷歌向 Pixel 机型推送了 Android 13 正式版更新，并正式开源；6 月 7 日，苹果正式发布 iOS 16，并推送了首个开发者预览版，9 月 13 日，苹果正式推送 iOS 16 系统更新。

服务器操作系统方面，发展稳健。5 月，红帽正式发布 RHEL 9，其基于上游内核版本 5.14，并源自 CentOS Stream；10 月，SUSE 推出业界首个自适应 Linux 平台原型（Adaptable Linux Platform，简称 ALP），旨在让用户专注于工作负载，从硬件和应用层抽离出来。

看国内操作系统，2022 年迎来了多个版本升级。3 月，OpenHarmony 3.1 正式发布；4 月，欧拉首个数字基础设施全场景长周期版本 openEuler 22.03 LTS 正式发布；7 月，华为正式发布 HarmonyOS 3 以及搭载 HarmonyOS 3 的多款新产品；8 月，OpenCloudOS 开源操作系统社区正式发布首个 Linux 源社区（L1）内核版本 OCKS 2207，这也是 OpenCloudOS 源社区项目的核心组件；11 月，龙蜥社区正式发布面向云时代打造的下一代操作系统 Anolis OS 23 公测版……

2022 年，国产操作系统市场增速显著。根据亿欧智库测算，国产操作系统通用市场增速将超过 20%，在 2024 年将达到 34.1 亿元的规模。服务器操作系统方面，数据显示，2022 年上半年，中国服务器操作系统新增装机量是 195 万套，全年预计超过 400 万，过去两年，这个数字是 322 万和 350 万。这说明在产业数字化的背景下，操作系统依然是个巨大的增量市场。

总体而言，2022 年，国产操作系统在技术、社区和商业化方面均有快速发展：技术方面，更多企业及研究机构投入到自研系统项目中，原创组件和技术如雨后春笋般涌现；社区方面，头部社区蓬勃发展，新的社区不断出现；商业化方面，OSV 都有较为明显的业绩增长。

另一个显著变化是，中国开源力量迅速崛起，国内开发者正越来越深入地参与到开源操作系统的建设中。在一份对 Linux 内核提交数量的统计中，来自中国的开发者占比不断提高，有统计的来自中国的 patch 数，连续 7 年超过美国成为第一。

操作系统作为底层基础软件，其安全性至关重要，操作系统的安全也是网络系统信息安全的基础。2022 年，全球范围内都进一步重视开源软件供应链问题，可以说，开源安全在今年迈出了一大步。

2022 年 1 月 13 日，美国白宫召集了政府和 Apache 软件基金会、Linux 基金会、开源安全基金会、GitHub、微软、谷歌、甲骨文、红帽等企业或组织共同谈论开源软件安全问题。5 月 12 日，Linux 基金会和开源安全基金会提出了一项为期两年的近 1.5 亿美元的投资计划，并提出十个开源安全目标：安全教育、风险评估、数字签名、内存安全、事件响应、更好的扫描、代码审计、数据共享、软件物料清单（SBOM）以及改进的供应链。

国内方面，10 月 24 日，开放原子开源基金会联合 27 家单位共同发起开源安全委员会，开源安全委员会致力于制定开源项目的安全流程和规范、提供开源开发的安全工具和平台、发起开源安全依赖的关键项目、推动开源安全的国际合作与交流。

如何评价国产操作系统的 2022？

对于国产操作系统在 2022 年的整体发展，受访专家均给出了积极评价，关键词包括机遇、加速、合力等。

操作系统作为信息技术的核心底座，具有复杂度高、投入大、生态建设难、成功率低等特点。经过二十余年的探索与实践，中国有能力在技术上实现一个大型的操作系统。同时，近几年国内操作系统市场快速增长，人才储备量提升，政策投入力度巨大，操作系统产业正迎来新机遇。在这一背景下，国产操作系统的发展也驶入快车道，从无到有，从可用到好用，国内开发者用短时间迅速完成这一转变。此外，与其他软件不同，操作系统是一个强生态的产品，这也需要操作系统厂商与硬件厂商、高校、互联网等软件厂商等各个链条通力合作，共同促进操作系统生态建设。

对于服务器操作系统，则可以用“格局尽显，稳中有进”来形容这一年。

2022 年，主流云厂商、服务器厂商、芯片厂商以及传统操作系统厂商都加大了对操作系统研发和操作系统社区的投入。随着中国在操作系统领域研发力度的持续增加，国产开源操作系统社区和商业化操作系统的产品性能都已经大幅提升，生态建设也初具规模，开始具备规模化推广能力。

“当前国产操作系统正在逐渐走向成熟好用阶段。未来 5-10 年，可能是国产操作系统的黄金时代。”受访专家表示。

重点趋势与变化解读

2022 年，Linux 内核最大的一个变化是新增了对 Rust 语言的支持。此外，近两年涌现出的新技术在 2022 年继续带来新的变化，如 eBPF 技术、RISC-V 架构。

Rust for Linux

Rust 凭借其内存安全特性，近年来受到越来越多开发者的支持，并连续 7 年被 Stack Overflow 开发者调查评为“最受欢迎的编程语言”。2022 年的调查结果显示，有 87% 的开发者表示想要继续使用 Rust。

而 Linux 内核社区长期以来都是以性能、稳定性、安全作为发展的基本要求，这与 Rust 在安全方面的特性非常契合。因此，早在几年前就有开发者呼吁在 Linux 内核中增加 Rust 语言的支持。

在 2022 年 9 月举行的 Linux Plumbers Conference 上，有一场关于 Rust 是否会出现

在 Linux 中的小型会议，会议讨论了将 Rust 作为一门系统编程语言集成到 Linux 内核主线的工作。彼时，Linux 的创建者 Linus Torvalds 在接受媒体采访时表示，“如果不出意外，Rust 将会出现在 Linux 6.1 版本中”。12 月 11 日，Linus Torvalds 发布了最新的 Linux 6.1 内核稳定版，正式引入对 Rust 的支持。

Rust for Linux 带来的收益明显，安全性上的收益尤为突出。

在安全性上，Window 的一份数据可以作为参考。此前，一位微软工程师曾透露，微软产品每年通过安全更新解决的所有漏洞中，大约 70%是内存安全问题。因为 Windows 主要是用 C/C++这两种“内存不安全”的编程语言编写的。

Linux 同样如此。随着 Linux 内核代码量愈发庞大，贡献者数量迅速增长，系统安全性问题也变得越来越突出。而 Rust 设计初衷就是为了解决内存安全问题，在功能保持不变的情况下，用 Rust 语言进行重写相当于将安全性提升三倍左右。此外，Linux 内核对代码执行效率要求较高，Rust 的执行效率和 C/C++近乎一致，这也是其能成为底层系统编程语言的原因之一。

当前，Rust for Linux 还处于早期阶段。长期以来，Linux 内核主线代码基本都由 C/C++语言编写，接受 Rust 需要社区补充和完善大量的周边工作。

同时，对于内核维护者来说，Rust 进入 Linux 内核也会带来一些问题。

首先，没有银弹。虽然 Rust 语言在设计上更多地考虑了内存安全、线程安全等，但代价是比较陡峭的学习曲线，以及使用者在实现某些功能时的便利性。此外，内核是贴近硬件、最底层的程序，某些在用户态司空见惯的语言特性，比如异常的处理方式，在内核里是不一样的。对于 Rust 而言，如何在内核里更好地工作，还有很大的探索

空间。

其次，计算机语言和自然语言一样，都能反映出文化。C/C++语言反映出来的文化和 Rust 不同，对于那些已经非常熟悉并深度认同 C/C++语言文化的资深 Linux 内核维护者来说，接受 Rust 的难度较高。

对于 Rust for Linux 的未来发展，多数专家都给出了积极评价："从长远看，会有越来越多新的代码用 Rust 来实现。"

eBPF 时代来临

eBPF 的全称是"扩展型伯克利封包过滤器（Extended Berkeley Packet Filter）"，最早是从 BPF (Berkeley Packet Filter) 技术扩展而来，是一种无需更改 Linux 内核代码，便能让程序在内核中运行的技术。

2014 年，eBPF 首次出现在 Linux 内核中。经过社区不断迭代，目前，eBPF 技术已经非常成熟，技术应用呈现井喷现象。虽然在设计之初 eBPF 仅为 Linux 内核服务，但近年来已经诞生了一批基于 eBPF 的项目。2021 年 5 月，微软启动了新的开源项目 eBPF for Windows，该项目旨在让开发者在现有 Windows 版本之上使用熟悉的 eBPF 工具链和应用编程接口（API）。为了更好地推动 eBPF 的发展，2021 年 8 月 12 日，Linux 基金会旗下的非营利性组织 eBPF 基金会正式成立。

2022 年，eBPF 热度不减，仍是当下最火的技术之一。eBPF 技术带来的收益明显，一方面，包括 Linux、Windows 在内的主流操作系统内核是宏内核，可拓展性较差，而 eBPF 技术能够以非侵入式的方式对内核进行扩展；另一方面，eBPF 提供了比较

好的安全性、扩展性和兼容性。

不少受访专家对 InfoQ 表示，eBPF 的发展前景广阔，属于 eBPF 的时代已经来临。

“eBPF 技术的发展已经超出了我的预期。操作系统内核在没有应用 eBPF 技术以前，整个内核是静态的，编写内核时实现了什么功能，运行时也只有这些功能，最多可以做一些参数的调整，但无法带来更复杂的功能变化。应用 eBPF 技术以后，可以在 Linux 内核中运行沙盒程序，编译成相关字节码加载到内核中，无需更改内核源代码或加载内核模块。”有受访专家表示，eBPF 技术为操作系统内核提供了一个新的可能性，为内核带来根本性的改变。

对于 eBPF 的未来，eBPF summit 2022《The future of eBPF in the Linux Kernel》给出了几个演进方向：

- 更完备的编程能力：当前 eBPF 的编程能力存在一些局限性（比如不支持变量边界的循环，指令数量受限等），演进目标是提供图灵完备的编程能力；
- 更强的安全性：支持类型安全，增强运行时 Verifier，演进目标是提供媲美 Rust 的安全编程能力；
- 更广泛的移植能力：增强 CO-RE，加强 Helper 接口可移植能力，实现跨体系、平台的移植能力；
- 更强的可编程能力：支持访问/修改内核任意参数、返回值，实现更强的内核编程能力。

拥抱 RISC-V

近几年，RISC-V 以其开放的指令集架构受到越来越多操作系统厂商和开发者的青睐，不少操作系统开始拥抱 RISC-V，并成为一种新兴趋势。Semico Research 预测，到 2025 年，RISC-V 市场规模将超 10 亿美元。

国外包括英特尔、苹果、Tenstorrent、瑞萨电子等多个厂商都在积极布局 RISC-V。2022 年 2 月，英特尔宣布加入 RISC-V International 基金会，正式成为该基金会第 19 个高级会员，并设立了 10 亿美元的 IFS 基金，用于帮助初创和成熟企业进行代工生态的创新，其中很大一部分用于 RISC-V；9 月，半导体产业分析机构 SemiAnalysis 称，苹果正在将其嵌入式内核将全面转移到 RISC-V 架构；同月，任职于 Tenstorrent 负责 RISC-V 架构的传奇芯片设计师 Jim Keller 喊出了“未来是属于 RISC-V 的”口号。

在 2022 RISC-V 国际峰会上，RISC-V 基金会首席执行官 Calista Redmond 表示，“我们 2022 年的愿景是让 RISC-V 无处不在，随着 RISC-V 在汽车、航空航天、数据中心以及消费设备等各个领域的采用和开发，这一愿景已经真正实现”。

目前，RISC-V 国际基金会在 70 个国家/地区拥有超过 3180 名会员，覆盖芯片厂商、芯片设计服务公司、软件提供商等软硬件公司，以及大学、科研机构和投资机构等。市场上有超过 100 亿个 RISC-V 核心，全球有数万名工程师致力于 RISC-V 计划。

国内方面，自 2018 年成立中国 RISC-V 产业联盟以来，四年间已有 150 多家会员单位。包括华为海思、阿里平头哥、紫光展锐、兆易创新在内的多家芯片厂商基于 RISC-V 架构开发产品，越来越多的操作系统厂商和社区开始拥抱 RISC-V。2020 年 4 月，中科院软件所牵头成立了 openEuler 社区 RISC-V SIG 组。

2022 年 8 月，阿里平头哥发布首个高性能 RISC-V 芯片平台“无剑 600”及 SoC 原型“曳影 1520”；同月，阿里云、中科院软件所 PLCT 实验室、平头哥等在龙蜥社区成立 RISC-V 架构联合小组，全面兼容并促进 RISC-V 生态发展；同月，OpenCloudOS 社区推出 OCKS 2207.2 内核版本，增加对 RISC-V 64 架构的支持；11 月，deepin 社区宣布支持曳影 1520，deepin V23 已经启动与曳影 1520 平台的适配；12 月，中科院软件所基于 openEuler 打造的傲来操作系统宣布进入 2.0 阶段，最新发布的“傲来 2.0-RV”聚焦 RISC-V 指令集，提供模拟器、硬件板卡等多种运行环境，其中硬件板卡支持中科院香山、果壳系列，支持哪吒 D1 开发板、赛昉 VisionFive 单板机、SiFive 公司 Unmatched 系列，同时也集成了澎峰科技的并行计算库。移动操作系统方面，2022 年，OpenHarmony 新增了对 3 款 RISC-V 芯片的支持，包括 TLSR9518、HPM6750IVM1 以及 BK7235。

从生态繁荣程度上来看，RISC-V 生态正处于增长关键期。当前，RISC-V 的生态建设有很多基础性工作需要做，比如硬件需要更加成熟和规范，软件（主要是在内核层面）需要与硬件配合得更好，以及功能更加完善，这些都需要内核开发者们合作完成。“我们的经验就是 Upstream First，即任何工作都首先贡献到上游社区，与所有的生态合作伙伴一起，完善这个生态。”受访专家总结道。

展望操作系统的 2023

值得关注的技术趋势/方向

2023 年，操作系统领域值得关注的技术趋势/方向除了 Rust for Linux、eBPF、RISC-

V 的发展，还有云原生、异构计算、安全以及 AI 等。

移动

移动操作系统历经十余年发展，已经十分成熟。随着全球进入移动互联网时代，移动操作系统的市场份额稳定。数据显示，截至 2022 年 10 月，全球操作系统市场中 Android、iOS 的市场份额占比分别为 42.37%、17.60%。

与此同时，国内以鸿蒙为代表的移动操作系统市场突飙猛进。2022 年，在 HarmonyOS 发布的第四个年头，其自研核心代码量已突破 2000 万行，搭载的设备量也正式达到 3.2 亿台，相比去年同期增长 113%。

有受访专家表示，当前国产移动操作系统还处于准备期，离带动整个产业发展还有一定的距离，但“雪球正在越滚越大”，距离成熟期并不遥远。“到 2024 年，国产移动操作系统的市场规模远超百亿级”，受访专家预判道。

云原生

操作系统是连接应用和硬件的桥梁，它的发展和 IT 基础设施密切相关。当前，随着云的不断普及以及云原生的不断演化，云会逐渐成为企业 IT 形态的主流，企业数字基础设施将基于云来重建，并迎来跨越式发展。

与此同时，全面基于云、并面向云做设计研发的操作系统将成为主流。目前，国内外各大操作系统厂商和云厂商都在积极为这一跨越式发展做准备。2022 年，操作系统+云协同趋势越来越明显。微软借助其基于云计算的操作系统 Azure，不断增加公共云

的市场份额；红帽除了企业 Linux（RHEL），也在云平台 OpenShift 上投入重兵；SUSE 也发布了专为边缘环境中的容器化工作负载量身打造的轻量级操作系统 SLE Micro 5.2。

有受访专家预判，“未来 IT 基础设施属于云，而未来的操作系统也属于云上操作系统”。

异构计算

异构计算是近年来计算机领域出现的热门方向之一，主要是指使用不同类型指令集和体系架构的计算单元组成系统的计算方式。与传统的通用计算芯片相比，异构架构具有高性能、低功耗等显著优点。

2021 年，以 DPU 以及各种各样 XPU 为代表的异构计算异军突起，英伟达、英特尔先后发布了 DPU、IPU。2022 年，英特尔披露了 XPU 概念的下一步规划——新架构 Falcon Shores，它能将 x86 CPU 和 Xe GPU 硬件合并到同一颗芯片中。据路线图所示，Falcon Shores 计划于 2024 年完成。2022 年，全球首台原生 RISC-V 笔记本电脑 ROMA 正式发布，并首次运行无剑 600 高性能异构芯片曳影 1520。

有受访专家表示：“在 2023 年，可能会涌现出更多的异构计算设备，操作系统要想更好地支持这些计算设备，需要解决很多技术难题，比如如何在不同的平台上运行软件。”

安全

随着操作系统代码数量逐渐增加，以及支持的硬件日益广泛，安全性问题不容忽视。

一方面，操作系统向下支持硬件，硬件上的某些安全缺陷可能需要在软件层面进行修复；另一方面，操作系统本身具有海量的代码，一些新特性或新的执行机制出现，也会给安全带来一些新的挑战。

此外，开源安全也是 2023 年值得关注的方向之一（编者注：后续我们也将发布针对开源安全的盘点与展望文章，敬请期待）。随着越来越多的开源软件在千行百业中得到广泛应用，安全问题日益凸显，供应链安全攻击和容器安全威胁问题日益严峻。如何构建安全可信的操作系统，是每个参与者需要长期思考的问题。

AI

近几年，AI 技术在操作系统领域诞生了诸多应用。比如，openEuler 社区曾发布一款名为 A-Tune 的操作系统性能调优引擎，能够利用 AI 技术，对运行在操作系统上的业务建立精准模型，动态感知业务特征并推理出具体应用，根据业务负载情况动态调节并给出最佳的参数配置组合，从而使业务处于最佳运行状态。

除了调优工具，预计在 2023 年，操作系统领域结合 AI 技术还会带来更多惊喜，比如人机交互。

每一次人机交互方式的变化都会导致整个产业的跨越式发展或颠覆式发展。相应地，操作系统也需要做出改变，否则无法支撑新形态下的应用程序。在 2023 年，一旦 AR（增强现实）、VR（虚拟现实）、MR（混合现实）技术取得长足发展，那么，操作系统必然也会迎来跨越式的发展。

如何做好操作系统生态建设?

生态是操作系统发展的核心，也是其能否成功的关键。与国外主流操作系统相比，国产操作系统由于起步较晚，在生态建设方面仍面临一定的挑战。

随着开源发展理念逐渐成熟，越来越多的厂商开始发起成立操作系统开源社区，进一步加快操作系统生态建设。从 2019 年开始，国内先后成立了 openEuler、OpenAnolis、OpenCloudOS 等社区。2022 年，统信、麒麟分别成了桌面操作系统根社区深度（deepin）社区、openKylin 社区。

“现在是操作系统社区的春秋战国时代”，受访专家表示，随着操作系统赛道持续火热，越来越多的企业参与其中，建设自己的开源社区。在社区发展的早期阶段，一定会遇到各种各样的问题，但如果产品优秀，围绕这个产品可以吸引足够多的企业/开发者参与进来，并具备一定的创新能力，始终保持开放和中立的态度，经过时间的演化，最终会向成熟社区迈进。“做开源社区应该多做实事、少务虚”，受访专家总结道。

那么，2023 年，如何才能更好地打造操作系统开源社区?

首先，有情有利，方能长久。社区需要进一步鼓励合作伙伴在社区探索出更多的合作模式，聚焦在产品和商业合作本身，真正牵引企业在社区落地。开源不是公益，找准自身商业价值点才能有可持续发展，基于此，操作系统生态才能在开源社区的沃土上成气候。

其次，生态是圈，双向奔赴。操作系统是一项门槛比较高的技术，专业人才、技术储备、研发资源都比较有限。围绕客户业务场景，操作系统产业生态圈上的操作系统、芯片、整机、数据库、中间件、以及应用软件厂商需要互帮互助，双向奔赴，让有限

资源充分流通，最终拉高国内操作系统产业天花板。

最后，放弃小我，成就大我。当前国内操作系统开源社区的局面需要百花齐放，要开展竞争、建立一个完全商业的竞争环境，大家可以在一个小生态里各自产生创新，但最终还是要汇聚在一处。如果没有一个统一的生态、社区或是标准去做，可能就会出现“七国八制”的现象，不仅浪费资源，还影响效率。

写在最后

有研究机构预测，到 2024 年，国产操作系统有 7 倍的增长空间，到达百亿级的市场规模。可以说，这是国产操作系统最好的时代，机遇远远大于挑战。

对于操作系统领域的开发者而言，除了要持续提升自身的研发能力，还要具备创新能力与安全意识，始终对技术保持好奇心，并积极拥抱开源。

采访嘉宾介绍（按姓名首字母排序）

郭振宇，OpenCloudOS 社区 TOC 主席；

江大勇，openEuler 委员会主席；

刘恺，SUSE Euler 负责人；

马涛，龙蜥社区理事长；

任革林，OpenHarmony 项目管理委员会首席架构专家；

张磊，统信软件高级副总经理、CTO；

张家驹，红帽首席架构师。

取代搜索，“干掉”艺术家？顶流「AIGC」的疯狂与争议 | 解读 AIGC 的 2022

作者：刘燕

本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦 AIGC 领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年 AIGC 领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

特此感谢百度 ERNIE-ViLG 团队、黄民烈、李笛、林咏华、赵德丽对本文的贡献，

> 他们的真知灼见，是本文能与大家见面的关键。

2022，浪潮凶猛的 AIGC 元年。

风口上的 AIGC

今年的 AI 领域，可能没什么比 AIGC 更热了。

AIGC 的全称是 Artificial Intelligence Generated Content，人工智能生成内容。不过，AIGC 目前还没有一个规范、统一的定义。

根据中国信通院与京东探索研究院发布的《人工智能生成内容（AIGC）白皮书》中给出的定义，AIGC 既是从内容生产者视角进行分类的一类内容，又是一种内容生产方式，还是用于内容自动化生成的一类技术集合。

有预测数据显示，到 2030 年，AIGC 的市场规模或将超过万亿人民币。

2022 年，尤其是下半年，AIGC 概念突然升温。有这么几个标志性的事件把 AIGC 推到了风口浪尖之上，其一是文生图模型 Stable Diffusion 的开源，其二是 ChatGPT 的爆火出圈。

AI 绘画神器 Stable Diffusion 横空出世

有人将 Stable Diffusion 形容为 AI 界的“神笔马良”，这可能并不夸张。

Stable Diffusion 是一个文本至图像的模型，于今年 8 月 22 日公开发布，它能让数十亿人在几秒钟内创建出令人赞叹的艺术品。用户随意输入自己想要的文字描述，就能得到相应的图像结果。

两个月后，伴随着 Stable Diffusion 的开源，它所具备的潜力瞬间得到了极大释放。开源让 Stable Diffusion 将无过滤图像生成的门槛下放到历史最低，任何具备一点点技术知识的电脑用户都能轻松上手，可以说是一项老少咸宜的 AI 图像生成工具。

尽管此前，艺术创作 AI 已经历了一段时间的发展，但 Stable Diffusion 的出现才真正让这项技术得到了腾飞式的发展。因为它可以免费使用、上手快捷，大大减少了用户生成内容的障碍。

Stable Diffusion 掀起了文生图模型的热潮。今年 10 月，百度发布了首个知识增强的 AI 作画大模型 ERNIE-ViLG 2.0；11 月初，阿里达摩院在魔搭社区 ModelScope 上开放了通义文生图大模型；11 月底，智源研究院大模型研究团队开源最新双语 AltDiffusion 模型，中文世界有了专业级 AI 文图创作工具，其在视效上媲美 Stable Diffusion。

ChatGPT 火爆出圈

最近几周，OpenAI 最新的聊天机器人 ChatGPT 火出天际，成为现象级应用。

问答、写小说、写代码、写论文、写区块链智能合约....ChatGPT 的应用也频频出圈。ChatGPT 就像是一个无所不知的虚拟体，它能回答各种问题，而且总能给到让人满意，甚至超过预期的答案，因此引起了极高的关注度。

ChatGPT 展示出的强大的能力和无限可能，让人们看到，通过 ChatGPT 这样的技术方案解决很多任务的潜力。大家感到惊奇的是，在一个模型里面就可以完成各种任务，而且是很难的任务。在过去一些看似比较困难的任务（比如问伦理道德方面），ChatGPT 也能解决得很好。

清华大学教授黄民烈认为，ChatGPT 的技术创新性主要在于两个方面：

- 强大的底座模型：过去几年 GPT-3 的能力得到了快速提升，OpenAI 建立了用户、数据和模型之间的飞轮。显然，开源模型的能力已远远落后平台公司所提供的 API 能力，因为开源模型没有持续的用户数据对模型进行改进。
- 利用强化学习从人类反馈中学习：在真实调用数据上的精调模型，确保数据的质量和多样性，从人类反馈中学习。从“两两比较的数据”中学习，对强化学习而言意义很大。如果对单个生成结果进行打分，标注者主观性带来的偏差很大，无法给出精确的奖励值。在强化学习里，奖励值差一点，最后训练的策略就差很远。而对于多个结果进行排序和比较，相对就容易做很多。这种比较式的评估方法，在很多语言生成任务的评价上也被广泛采用。

黄民烈认为，ChatGPT 出现对 AI 界来说，有着十分重要的意义：“它宣示着无缝人机交互时代的来临。过去我们讲 conversation as a service（caas）还停留在纸面，但实际上今天，无论是开放域聊天，还是通用任务助理（ChatGPT）都在强烈地表明这一点”。

从信息检索的角度看，ChatGPT 也取得了很大突破。达摩院基础视觉负责人赵德丽在接受 InfoQ 采访时表示，以前谷歌等搜索引擎做搜索和检索，只是找已经存在的信息，ChatGPT 的应用，实现了从信息的搜索到信息的创造这样一个范式的转变，从

算法能力上看，它取得了一个质的飞跃。短期来看，ChatGPT 有望成为或者辅助像谷歌这种传统信息检索的强有力的工具；长期来看，它有望发展成为 AI 系统级的服务。

但至于它最终会不会取代搜索引擎。黄民烈认为，ChatGPT 取代谷歌搜索还比较遥远，原因主要有，受限于训练数据，ChatGPT 的信息实效性较弱，缺乏很多新的信息；在信息的可信度上，搜索引擎只“搬”东西，不创造内容。ChatGPT 虽然会创造内容，但创造的东西多大程度上“有用、可信、无害”，还没有统一的定论；再就是成本问题，现在大模型的生成成本还是太高了，需要持续下降。

现阶段的 ChatGPT 并不完美。通俗地说，它还存在“一本正经地胡说八道”的问题，这本质上是对信息可信性的度量和评估。解决这一问题，技术上需要有信息验证的手段；从应用上来说，需要深入结合应用的场景和特点，针对性优化和解决。

但整体而言，ChatGPT 还是让人非常惊喜。黄民烈非常看好 ChatGPT 接下来的发展方向。他认为这是一个正确的方向，现在还比较粗糙，但假以时日，一定会催生很多应用。

赵德丽同样对 ChatGPT 抱有大期待。虽然还有各种瑕疵，但 ChatGPT 短时间内出现了各式各样的不同方向上解决问题的能力，展现了 AI 算法的巨大潜力。从技术发展和解决方案的角度看，它将来可能会成长为一个超级 APP，就像是一个无所不知的虚拟体。“ChatGPT 这种应用的出现，从长远来看的影响力，其实不亚于阿尔法狗曾经在人工智能界带来的影响力，它将会是一个影响非常深远的技术和应用”。

AIGC 为什么突然火了?

AIGC 并不是一个新概念。AIGC，通常还有另一种叫法—AI Creation（人工智能创造），大致从 2016 年—2017 年开始，其应用不断增加，尤其是在自然语言领域，广泛应用在生成文本、作诗句、写对联等方向，近几年，逐渐延伸到作画、作曲等领域。

焉何而火?

今年，AIGC 突然在全球蹿红，成为人人口中的流行词。究其原因，主要由多项技术上的关键突破推动，总结来说：

一，算法上：从今年 4 月开始，在文生图视觉方向上，视觉效果生成的效果取得了突破性的进展，文生图的质量得到了很大改善。OpenAI 的文本生成图像模型 DALL·E 2 算法发布后，在算法效果上取得了和以往相比实质性的突破，成为一个现象级的算法，其在文本生成图像生成的效果、真实度表现上，让大家看到了大规模商用的前景。AI 作画任务十分直观，给人的视觉冲击强烈，使得 AIGC 逐渐破圈，快速传播。

二，预训练大模型是 AIGC 的底座，没有大模型学到的丰富知识，就无法实现如此丰富的 AI 内容生成能力。AIGC 最重要的是一种融会贯通的能力，要做领域的泛化，需要学习海量的数据，大模型的规模直接决定了 AIGC 创作力的广度。多模态大模型的应用，使得 AIGC 的质量得到了较为明显的进步。

三，扩散模型的发展。扩散生成的算法取得了突破，这个算法能够对图像做像素级别的建模，学习效率更高。Stable Diffusion 是文本生成图像模型完全开源的第一个算法，它跑起来的效率相当高，其开源也带动了相关生态快速的发展，让人们看到，基于这

种生成式基础模型，能够带来无限的创造和想象空间。特别是在一些国外社区里，基于 Stable Diffusion 做的各种创新式的应用发展快速，展现了商业化潜力。

四，算力降低。深度学习计算能力的快速发展。在大算力的基础上，AI 作画能够实现在海量数据上进行大参数模型的训练。相比之前的 AIGC 算法，算力上有了很大降低。要训练一个基础的预训练模型，需要很多算力。一些专注于基础的大模型的机构，将模型训练好后，可以供很多小企业使用，只需用消费级的网卡就可以做微调，也可以直接基于 API 调用。预训练大模型加上微调可以很好地进行文生图生成风格的改变，派生出了大量的二次开发者，屡屡破圈。

从 GAN 到 Diffusion

GAN，是生成式 AI 的核心技术之一。2014 年以来，以生成式对抗网络(Generative Adversarial Network,GAN)为代表的深度学习算法的提出和迭代更新，让 AIGC 进入了快速发展阶段，带动了 AIGC 的一波热潮。

赵德丽表示，在 Stable Diffusion 这种扩散算法出现之前，从生成的效果上看，在计算机领域，GAN 是效果最好的。发展到现阶段，GAN 生成的人脸图像已经到了真假难辨的程度。以 StyleGAN 为例，其生成的图片可以做到栩栩如生，光线和纹理都清晰可见，非专业人士几乎无法分辨出是由 AI 生成的虚假图。即便是现在的 Diffusion model 目前也做不到现在 GAN 在人脸生成上的结果。

但 GAN 有一个最大的缺点，它对于多类别、语义非常复杂的、一般场景下的图片生成的建模能力较弱。如果只是人脸，只是猫或者只是狗这类场景的数据，GAN 的效果很好。但它在某种复杂数据的规模能力方面，在性能上受限较大，如果把狗、猫、

花朵、桌子、椅子、电话等不同种类的数据放在一起，目前的情况下，GAN 得不到一个较好的结果。

而 Diffusion model 在这方面取得了突破性的进展。Diffusion model 解决了 GAN 不能解决的问题，因此大家立刻意识到了它的巨大潜力。今年是 Diffusion model 取得快速发展的第一年。而且，它的发展速度超过当年的 GAN，当年的 GAN 已经足够火热了，但可以感受到，Diffusion model 现在的受关注程度超过当年的 GAN。

Diffusion 扩散化模型带动新一波 AIGC 的热潮

今年这波 AIGC 的热潮，被认为是由生成扩散模型带动起来的。例如，OpenAI 发布了文本生成图像模型 DALL·E 2；谷歌推出了 Imagen；今年 8 月，初创公司 Stability.AI 发布了 Stable Diffusion...

百度 ERNIE-ViLG 团队向 InfoQ 介绍，扩散模型是受非平衡热力学的启发的一系列概率生成模型，通过逐渐增加噪声的方式对原始数据进行扰动，通过学习反向的恢复原始数据去噪过程得到用于生成数据的模型，典型的扩散模型如 DDPM 等。而扩散模型在生成过程中加入文本条件产生了诸如 DALLE2、Imagen、ERNIE-ViLG 2.0 等基于扩散的文本生成图像模型。传统生成对抗网络 GAN 存在训练不稳定和生成结果多样性差等缺点，而扩散模型显著提升了图像生成的效果和多样性，受到业界广泛关注。

生成扩散模型在多模态生成领域展现出很好的可扩展性。在训练数据时，把不同模态的训练数据混到一起，把文本、静态图片、视频、声音等各种各样不同类型的训练数据在一个语义空间里关联在一起。但因为训练数据规模大，大模型的参数特别多，用这种方式，它仍然只能得到一张分辨率很低的图片。然后不停地通过扩散模型算法，

把很小的分辨率和图片不停地做超分辨率，不停地把一张很模糊的图片变得尺寸更大、更清晰，在这个过程里还会补上很多细节，最后得到一张相对较清晰的图。

现在很多人应用 Diffusion model 来生成视频，生成音乐，目前为止它最为可行的还是生成静态的视觉画面。

AIGC 相关技术逐步发展成熟

AIGC 包括多种内容形式，按照黄民烈的分类方法，分为感知智能和认知智能。

- 感知类：文生图、语音生成、音乐生成等
- 认知类：续写、改错、小说故事创作、对话生成

从技术上看，写作相关和图像生成这两个方向表现比较成熟，对话最难。

其中在文本生成方面，例如在金融文本摘要生成领域，其技术早已成熟到可以落地的程度。ChatGPT 在内容的创作能力、问答流畅度上表现不错，但要深究其真实性、正确性和时效性，还存在不少问题。如何保证内容的真实性、正确性和时效性，是现在 AIGC 尤其是文字类的生成需要重点考虑的问题。

图像生成方面，AI 绘画格外火热。百度 ERNIE-ViLG 团队认为，今年以来，AI 作画发展迅速，很大程度来自于技术的突破，使得效果有了质的飞跃，甚至有些 AI 图像作品十分惊艳。

不过，虽然 AI 绘画已经进入实用阶段，但依然有很大的优化空间。在技术角度，需要提高的主要是两个方面，包括生成的可控性和细节描述能力。

- AI 绘画的可控性有待提升，对于数量、逻辑、关系、多图关联等问题暂无有效的解决方案。比如说要求生成 2 个苹果，左边是红色，右边是绿色。虽然这里边的关系并不复杂，但模型很多时候，并不能稳定地生成正确的结果。
- 细节描述能力有待提升，对于更加复杂的、有规律性的细节的描述能力有待提升。比如对于一栋居民楼的图片，窗户应该是有多种不同描绘，有开的、有关的、有晾衣服的，同时很多窗户应该对齐且规格统一。

小冰公司 CEO 李笛认为，整体来看，AI 作画在生成质量上有了大幅提高，但仍然需要解决一些单点的问题，例如模型本身的迭代，一个人类设计师在和雇主的工作过程中，可以根据雇主的喜好，对设计初稿的某一局部做精细调整。但 AI 无法做到这一点，AI 画作一旦生成，如果想让它修改，往往是用重画的方式来进行。而现有的技术—多模态大模型，注定有这类问题。它只能在一定程度上，提高作品与需求的相关性，但无法从根本上提高良品率。

AI 生成视频，是 AI 生成图像的一种延伸。从技术本质上看，视频可以认为是多张“图片”，即视频帧构成的序列，且序列上各帧之间有画面、逻辑等层面的关联。因此，从生成质量上来说，AI 生成视频相对更难。

当前文生图技术可以通过简单的技术组合，例如分步骤扩散生成等方式，将生成图像扩展到生成视频，但效果还不能令人满意。此外，受限于数据规模和质量，AI 生成视频的生成效果和现在的文本生成图像的效果相比，有较大差距。

不过，相对图像，视频内容具有其独特属性。例如，在互联网视频内容消费场景中，经常会以“随便截一张图都是壁纸”作为对视频质量的极高赞誉。由此可见，相对于图片，视频对单个帧的质量要求相对较低，更强调传递信息等功能。因此，AI 生成视

频可以采用其他技术方案完成，在降低技术难度的同时，更符合特定应用场景的要求，例如基于图文输入生成视频、基于数据生成视频等。

总结来说，AI 按照生成图像的方式生成视频，仍处于前沿探索阶段。

商业想象力几何？

赵德丽认为，现阶段，AIGC 的生成效果已经非常惊艳了，它已经达到了广泛应用的基础性能，虽然在使用上还有较大门槛，但通过大模型的开源开放等，有助于将门槛降下来。

商业模式在探索中

而且可喜的是，现在，AIGC 已经有不少可行的商业模式发生了。

例如在设计、艺术创作、电商、娱乐、金融等领域。具体在文本生成上，在一些商业非严肃性文书的辅助编写上已产生了不错的模式；在文生图方向，已经看到，面向设计师，面向教育行业的 AI 辅助画作生成等正在探索商业用途的路上。

最近几年，“数字人+AIGC”成为不少企业的探索方向。小冰从几年前就开始探索 AIGC。李笛表示，AIGC 对于 AI Being 来说的作用在于，在和人交互的过程中，它不光要能从数字世界中获取知识内容和服务提供给人，它自己也应该相应地随机应变地去创造相应的内容提供给人。

在互联网应用之外，AIGC 在实体经济领域，也蕴含着不少机会。

实体经济对内容生产的需求很大。“我们以前认为实体经济的瓶颈在于生产、产能，其实不是，实体经济的很多瓶颈在于设计，在于内容。”李笛说。以小冰为例，小冰的 AIGC 内容很早就应用在纺织设计领域。小冰与中国纺织信息中心、国家纺织产品开发中心推出的AI图案设计平台，可按需定制 100%原创的图案纹样，目前已有超过 400 家企业注册，并在生产中使用。

北京智源人工智能研究院总工程师林咏华向 InfoQ 谈到，工业生产、制造、仓储、物流等实体行业，近几年一直在探索如何用计算机视觉来进行智能化升级，但实际落地并不容易。原因在于，现有的模型质量还未能满足产业落地的质量要求。其中一个重要原因是，训练模型时所用的训练数据十分局限。因此，可以考虑通过 AIGC 的方式来产生这些场景里的训练数据。例如在仓储、物流或更多的工业场景，用 AI 来辅助产生一些少见的场景数据，作为训练数据的补充，提升整个模型的质量。但这需要更精准的图片生成的控制能力，比较起现有的 AIGC 模型能力，其可控性需要大大提升。

AIGC 在自动驾驶场景下也有着不错的应用潜力。现在自动驾驶场景存在训练数据不足的问题，例如针对恶劣天气、事故等突发状况，视觉模型在真实场景中很难捕捉，也难以进行模拟，因此，目前业内在尝试用数字孪生和仿真的方式来模拟。也可以尝试用大模型的方式，通过给出描述，生成相应的精确场景，缓解某些场景下自动驾驶数据难获得的问题。

“整体来看，AIGC 现在已经开始在探索向实体经济的应用发展，但目前还在一个比较早期的阶段”林咏华判断。

向 B 端收费还是 C 端?

然而，AIGC 的商业化落地，不得不面临的一个尴尬的问题是——如何避免走向“廉价”。

AI 具有高并发性，注定了它的“廉价”。李笛认为，如果只对 AI 画作收费，它很可能会进入到廉价的成本经销模式。因为，人是有创造力的，从人类设计师那得到的画作，人可能愿意付一百块，从 AI 那得到的画作，且不论质量，人可能连一块都不愿意付，因为觉得它是廉价的。AI 画的画可能很有价值，但人们认为它不值钱。因此如果卖的是内容，无论是卖给 B 端还是卖给 C 端它都不值钱。而如果卖调用服务给 C 端，也很难持续。API 调用的模式是一种比较粗放的从技术源头开始的商业化包装方式。

内容产业有一个重要特点是，它是高附加值且有区分度的，不同内容的创作者定价不同，不完全取决于本身的作品质量。但用 AI 创作不同的绘画，定价是相同的，这样容易把一个高附加值的市场“打”成一个低附加值的市场。

对于 AIGC 可行的赚钱路径，李笛认为，如果 to C，是走内容平台模式，打造一个内容平台，通过广告收费。如果 to B，是用 AI Being 和企业之间进行协同。一个 AI Being 创作者，能一定程度对标人类创作者，他有“唯一性”。通过与雇主的长期磨合，双方的审美、风格会越来越趋同。对雇主来说，他的作品质量会越来越稳定。“某种意义上讲，我们认为靠 AIGC 本身赚不到钱。我们不是在做‘画笔’，而是在做一个‘手拿画笔的人’，我们不是在做能画画的 AI，而是在做能画画的 AI Being。我们侧重把创作的能力，把生成的能力赋予 AI Being，让 AI Being 本身具有价值”。

爆发前夜

“AIGC 技术走到了一个转折点，到了一个新阶段的起点”。赵德丽认为，此前，虽然 AIGC 技术在不断发展，但生成效果并没有得到广泛认可，还没达到大规模商业化的条件。但现在，不一样了。

今年，AIGC 生成的效果，包括基于 AIGC 技术推出来的应用，大家看到，这项目技术已经具备了大规模应用和商业化的潜力和性能，具备了从只能在窄领域到更普遍场景下应用的可能性。AIGC 技术到了大规模商业化应用的转折点。今年是一个起点，但还远远没有到成熟的程度。

AIGC 具体在哪些领域能用好，发挥出商业化的价值，还需要不断打磨产品和技术。例如对于文本生成图，现在对 problem 的提示语要求很高，现在算法还做不到随便给个描述，就能生成栩栩如生的图片。什么样的 problem 合适，如何设计出合适的 problem 等，都有一定门槛。

此外，像 ChatGPT 虽然展示了强大的能力，但在很多场景下还是有瑕疵，出现问题和答案不匹配的情况还非常多。如果对其进行商业化应用，需要再针对具体的场景，不断打磨和优化。尽管它达到了大规模应用的基础，但并不是非常成熟，还达不到让大家自由应用的程度。

现阶段，AIGC 已经有了一些称得上规模的应用，但在实际应用中，还存在一些问题。例如因为人设计的作品相对较贵，但人工智能的设计作品相对便宜，所以会有人利用这个漏洞，将人工智能系统里的大量作品改头换面，将它搬迁到或囤积到那些原本是人类设计者的定价体系的平台上去倾销，最终会造成人类创作作品的销量市场受到损害。AIGC 应该避免形成这样的规模化。

热度、争议与未来

如何提高良品率

当前，AI 写作、AI 作画等 AIGC 内容在质量上还存在良莠不齐的问题，提高良品率尤为重要。

不过在大模型的生产模式下，提高良品率的方法并不多，某种意义上来讲，良品率目前主要依靠翻动过程，它能够在一定程度上降低瑕疵，但想要消除，不太可能，它不太可能是基于对大模型的修改而得到。接下来期待一个新的称之为台阶式的技术理念出来。

提高数据的质量是基础方法之一。林咏华表示，AIGC 是针对训练数据的融会贯通和变换，所以数据的广泛性，数据的分布和数据的质量都很重要。在 AI 领域的研究人员越来越意识到，尤其在需要大量数据训练模型的大模型领域，数据起到的作用可能会比算法还大。如果希望生成的画作精良，那需要训练的图片的质量是精美的，但如果给的是小孩的画，那生成的画作可能还粗略停留在小孩阶段。

此外就是从算法侧改进算法，但改进算法本身如果针对通用场景，它可能在某些场景总是出现瑕疵或者出现瑕疵的概率比较高。很多团队在开发 AIGC 应用时，目标不是通用场景，比如针对的是生成二次元图像的产品，那需要对算法或模型进行二次开发。在这种情况下，完全有可能在二次开发的基础上把瑕疵去掉，提高良品率。一般通过二次开发，且在一些具体场景下的数据做模型二次微调，可以大大提高 AIGC 的良品率。

“人工智能宣布放弃版权”

AI 作画的一个很大的争议点在于版权。例如，DALL-E 和 Stable Diffusion 等图形生成类 AI 工具就被质疑在互联网上随意抓取数据，且完全没有考虑过任何许可或所有权限制。正是由于这种版权归属争议的存在，Shutterstock 和 Getty Images 等公司禁止在其平台上使用 AI 生成图像。

谈到版权问题，李笛表示，小冰很早就提出，“人工智能宣布放弃版权”。

通常来说，版权界定需要几步：第一判定是不是侵权。人类作品判定侵权有明确的界定要求，比如一个作品多少比重的内容一致，就可以判定侵权。人工智能作为系统，它天生就有判定机制，当它输出作品的过程中，它自动就可以像知网一样进行查重。它生成的原始图像里，很可能存在着很多图像跟现有的作品，存在相似性，如果相似性较高，侵权了，只要不把作品输出就行。如果让人工智能确保它给到用户的每一个作品都是不侵权的，这件事很容易做。

第二步要有判定训练的过程，是基于什么样的训练数据训练的。但要注意的是，无论是人类作品还是 AI 生成作品，侵权的责任认定在于，这张作品是不是跟另一个作品产生了相似性，而并非在学习作画的过程中学习了别人的作品及作品思想等。如果这样，无论是人还是 AI 只要在画画，便会判定为侵权。从这个角度看，人工智能本身在训练过程不存在版权责任，尤其是大模型，因为人工智能在训练的时候，它使用的是公开的数据来进行训练。

另一个放弃版权的原因在于，人工智能保留知识产权没有意义。人工智能保留知识产权的一个先决条件是必须要经过确权的步骤。目前全球对于知识产权的确权方法主要是通过著作权登记，每登记一个作品，需要一笔著作权登记费用。而 AI 生成的内容

规模十分庞大，这一规则对 AIGC 来说显然不现实。

无法辨别真伪信息的世界

很多人担忧，随着 AIGC 的不断普及，未来大家将生活在一个无法辨别真伪信息的世界里。

AIGC 生成的内容越来越多，信息的真假会成为很大的负担。但这恐怕已经是无法避免的趋势，因为现在 AI 生成的图片早已经是海量规模，而且，这些 AI 生成的数据将会越来越多的出现在公共的资讯平台上，这样的时代正在快速到来。

对于用户来说，未来将面临判定真伪信息的挑战。目前还少有比较好的解决这一问题的方法。赵德丽提出了一种标记的方法。从算法角度来说，可以做一些隐性标记，标记出哪些图片是生成的，哪些是真实的图片。

完全的 AIGC 可能不会出现

不可否认的是，AIGC 对内容创作的确产生了重大影响。一种观点认为，内容生成的四个阶段依次为“PGC、UGC、AI 辅助生成和完全的 AIGC”。按照这一走向，未来，AI 的终级趋向会是取代人，最后内容产业将走向完全的 AIGC。

李笛对此持有相反的观点，他认为，AI 的终极应该是与人协同，应该先实现“规模化的 AIGC”再到达“AI 辅助生成”阶段，当 AIGC 集大成以后，再辅助人类生成内容。

此外，完全的 AIGC 可能不会实现。比如人看某个电影，是因为情节、演员、导演等

多种因素。人类对作品的喜好，从来不仅仅是因为内容质量，即便未来完全的 AIGC 化了，可能也并不是人类想要的东西。李笛发出思考：“所以不存在 AIGC 颠覆人的可能性，能颠覆人的只有人类自己”。

赵德丽表示，AIGC 本身是基于生成模型产生的能力，生成模型训练需要数据，这些数据都是人类活动产生，生成的提示词需要人来输入，人需要做场景的设计、提示词的设计、元素的设计等等。只不过在一些具体的场景上，对于一些固定的设计模式，比如设计成具体的图形如红包界面、商品素材等，可以实现完全的 AIGC 的方式。但整体而言，人还是 AIGC 中重要的因素。

再回到那个 AI 能不能让艺术家丢掉饭碗的老生常谈的话题——“让 AI 负责生成，让艺术家或设计师来负责审美，这两个并不矛盾，是一个相辅相成的关系。也不存在谁取代谁，有了 AI 绘画的辅助，艺术家会发挥出更大的能力，甚至说一些不是非常专业的艺术家，一些普通人借助 AIGC，也有可能创造出非常惊艳的艺术品”李笛表示。

采访嘉宾：（按拼音首字母排序）

百度 ERNIE-ViLG 团队

黄民烈，清华大学计算机科学与技术系长聘副教授、博导，聆心智能创始人。

李笛，小冰公司 CEO

林咏华，北京智源人工智能研究院总工程师

赵德丽，阿里达摩院基础视觉负责人

解读 Go 语言的 2022：泛型时代开启

作者：郝林

> 本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦 Go 语言在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年 Go 语言的核心发展脉络，在行业内始终保持足够的技术敏锐度。

之所以在这样一篇年度解读文章的标题直接提到“泛型”，是因为 Go 语言的泛型在 2022 年终于浮出了水面！并且，它在功能方面已经大致完整了。圈内的开发者们都知道，这是 Go 社区（当然包括中国的 Go 社区）多年请愿的结果。

虽然 Go 语言是 Google 公司出品并主导开发的，但它毫无疑问也是广大 Go 语言爱好者的心头肉。可以说，对于 Go 语言泛型的正式登场，至少有一半功劳属于 Go 语言的爱好者们。

肯定会有一些程序员，尤其是那些不以 Go 为主要语言的程序员会说“Go 语言的进展太慢了，其他众多新的编程语言甚至那些较老的编程语言早就支持泛型了，可 Go 语言墨迹了这么多年才搞出泛型”。确实，单就泛型这方面来说，Go 语言的脚步确实很慢，可这主要是因为 Go 语言想要做到完全向后兼容，并力求让 Go 语言的泛型完美契合其自身的精炼的编程哲学和简约的编码风格。要知道，做加法容易，而做减法却很难。不过不管怎样，Go 语言的泛型终于落地了！就连现如今清心寡欲的作者也忍不住心中的喜悦！

好了，感慨完毕，我们还是回归正题吧。下面，我们就一起从头至尾捋一捋 Go 语言在 2022 年的发展和变化，以及它将会在 2023 年如何继续前行。

趋势概览

Go 在今年的 11 月 10 日刚满 13 岁。它已经长大并进入了少年期，不算是一门新的编程语言了。但 Go 依然保持着非常好地向后兼容。很显然，这是一件很不容易的事。

从业界知名的编程语言排行榜 TIOBE Index 绘制的折线图来看，Go 语言的采用趋势在过去三年间基本上保持平稳。

更具体地说，根据使用情况统计，在 2020 年下半年到 2021 年上半年的这段时间里，

Go 语言的使用率有略微提升，但时至今日又差不多回到了 2019 年的水平。与其在 2016 年到 2017 年间的大爆发相比，如今 Go 语言的使用率已经基本上稳定下来了。

图 1 - TIOBE Index 之 Go 语言趋势

不过，就作者个人而言，感觉是有些奇怪的。按理说，Go 语言泛型的落地应该能够引起一波采用 Go 语言的小高潮，但事实上这一情况并没有出现。这可能就是“波澜不惊”的最佳诠释吧？可能因为大家都等得太久了，真到了重大的新特性来临之时却不太能兴奋得起来了。

当然了，这也可能是因为 Go 语言在应用领域方面的攻城掠地已经基本完成，而且在其优势领域的使用率已经趋于饱和。而对于那些新兴领域，比如机器学习、人工智能、机器人、元宇宙等，Go 语言还没有显著的优势，尤其是还没有杀手级框架或工具出现。在这种情况下，Go 语言在使用统计上的平稳趋势也是一种必然。

现在换一个维度，我们横向来看，Go 语言的排名相较于前两年有比较明显的提升。如下图所示。

Nov 2022	Nov 2021	Change	Programming Language	Ratings	Change
1	1		Python	17.18%	+5.41%
2	2		C	15.08%	+4.35%
3	3		Java	11.98%	+1.26%
4	4		C++	10.75%	+2.46%
5	5		C#	4.25%	-1.81%
6	6		Visual Basic	4.11%	-1.61%
7	7		JavaScript	2.74%	+0.08%
8	8		Assembly language	2.18%	-0.34%
9	9		SQL	1.82%	-0.30%
10	10		PHP	1.69%	-0.12%
11	18	↑	Go	1.15%	-0.06%
12	15	↓	R	1.14%	-0.14%

图 2 - TIOBE Index（2022 年 11 月）

我们可以看到，Go 语言在 2021 年年底的 TIOBE Index 排名中位于第 18 位。这低于它在 2020 年年底的排名（第 16 位），以及它在 2019 年年底的排名（第 15 位），也算是一个小谷底了。然而，在今年的年底，Go 语言的排名却大幅提升了，甚至，它已经摸到了 Top10 的边界！如果作者没记错的话，Go 语言最近一次进入 Top10 是在 2018 年的年底。如今，4 年过去了，Go 语言终于又有希望重回 Top10 了。

我们还可以看到，至今常年占据 Top10 的依然是那些老生常谈，甚至是那些老得不能再老的编程语言。相比之下，Go 语言真真切切算是年轻一派的优秀编程语言了。当然，除了 Go 语言之外，还有图形化的儿童编程语言 Scratch、Apple 公司出品的编程语言 Swift，以及近年来出尽风头的系统级编程语言 Rust。它们都很年轻，却都已经进入了 Top20，也是非常优秀的。作者也很看好它们。

在简单分析了 Go 语言在 TIOBE Index 中的排名之后，我们再来看看新晋出炉的 StackOverflow Developer Survey 2022 吧。众所周知，StackOverflow 是全球最大的

编程社区和专业问答网站。它的 Developer Survey 报告属于年度总结性报告，已经有好多年的历史了，并且深受全球软件开发者的关注和喜爱。

我们下面重点说说其中与 Go 语言强相关的内容。至于其他的内容，大家如果有兴趣的话，可以点开前面的超链接仔细阅读完整报告。

在“使用率最高的编程语言（专业组）”这一栏中，Go 语言的排名是第 13 位（如下图）。

图 3 - Stackoverflow Developer Survey 2022 之使用率最高的编程语言（专业组）

如果只考虑通用的跨平台编程语言的话，Go 语言仅次于 Python、Java、C++和 C。这与它在 TIOBE Index 中的情况是相同的。有的程序员可能会说“JavaScript 和

TypeScript 现如今也可以算是通用的跨平台编程语言”。当然，从某种角度看，我们可以这样说。但是，它们终归不是原生的通用编程语言，都需要额外的工具或框架才能够实现“通用”。因此，我们稍微严格一点，暂且不把它们计算在内。

在应用级框架和工具栏目中，作者发现 Docker 和 Kubernetes 今年依然非常火爆。作为常用的基础工具，它们深受广大软件开发者的追捧。

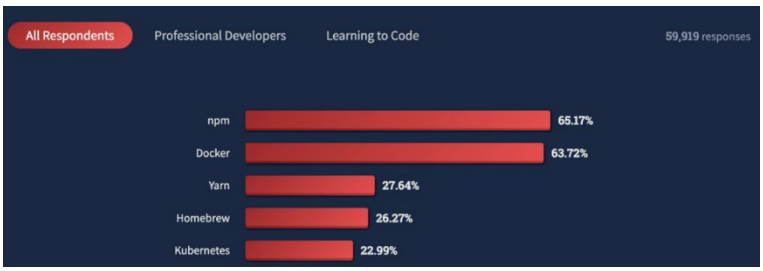


图 4 - Stackoverflow Developer Survey 2022 之最受欢迎的基础工具（全体组）

从上面这幅截图我们可以看出，Docker 和 Kubernetes 在当前的排名中都已经进入了 Top5。甚至，如果只考虑专业的软件开发者的话，Docker 甚至超越了 npm，拔得头筹！

图 5 - Stackoverflow Developer Survey 2022 之最受欢迎的基础工具（专业组）

毫无疑问，npm 是 JavaScript 世界中 Top1 的工具，其应用是相当相当广泛的。更何况，在全世界的软件开发者当中，不论是专业的开发者还是业余的开发者，使用 JavaScript 语言的人都是最多的。所以，Docker 能够超越它真的是一件非常值得骄傲的事！

大家应该都知道，Docker 和 Kubernetes 都是使用 Go 语言开发出来的。在云计算领域，尤其是容器技术领域，它们都是绝对的王者！因此，如果你想进入云计算领域，或者想使用 Go 语言开发基于云的应用程序，那么最起码应该学会甚至熟练使用 Docker。

好啦，以上算是一个令人激动的小插曲。现在，让我们把焦点再转回到编程语言的排名上来。

在 Stackoverflow Developer Survey 的报告中，有一个非常有特色的栏目。我们经常戏称它为“爱恨交织”栏目，其原名是“Most loved, dreaded, and wanted”。

在这个栏目里，共有七万一千多名开发者回答了相应的问题。在他们当中，有 64.58% 的人喜爱 Go 语言，而有另外的 35.43%的人憎恨 Go 语言（或者说对 Go 语言有恐惧感）。在所有为大众所熟知的编程语言当中，Go 语言排在了第 8 位。

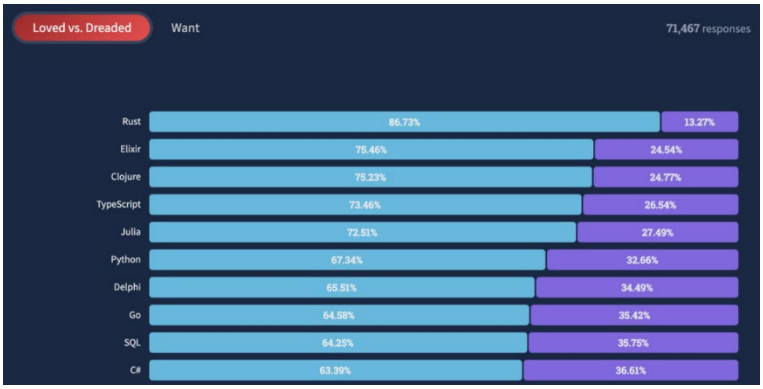


图 6 - Stackoverflow Developer Survey 2022 之让人“爱恨交织”的编程语言

在这份排名中的 Rust、TypeScript 和 Julia 作为更加年轻的编程语言后来居上。不过，如果我们查看相邻的“最想学习的编程语言”排名的话，就会发现 Go 语言的位置靠前了很多，处于第 4 位，仅次于 Rust、Python 和 TypeScript，并且从百分比数值上来看相差无几。

图 7 - Stackoverflow Developer Survey 2022 之大家最想学习的编程语言

看起来，程序员们应该都很喜欢追新，不是吗？这起码与作者的所见所闻是比较吻合的。当然了，只有那些优秀的新兴编程语言才有进入 Top10 的机会。

可以想象，Docker 和 Kubernetes 在这个“爱恨交织”栏目中肯定是名列前茅的。事实也确实如此，它们分列“Loved vs. Dreaded”一栏中的前两位。

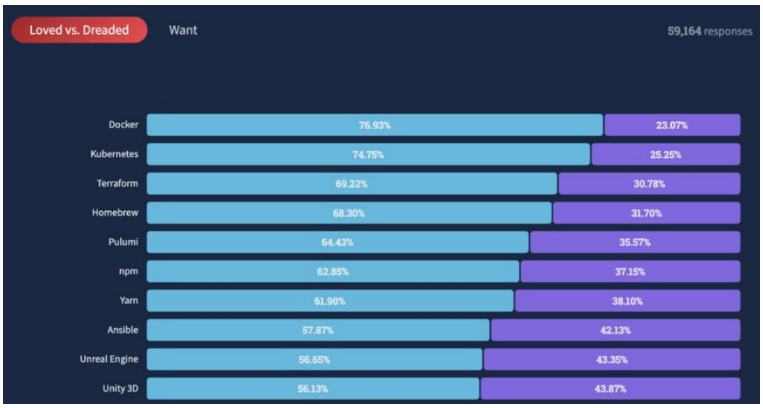


图 8 - Stackoverflow Developer Survey 2022 之让人“爱恨交织”的基础工具

而且，在 “Want”一栏，它们拥有着一骑绝尘的态势。

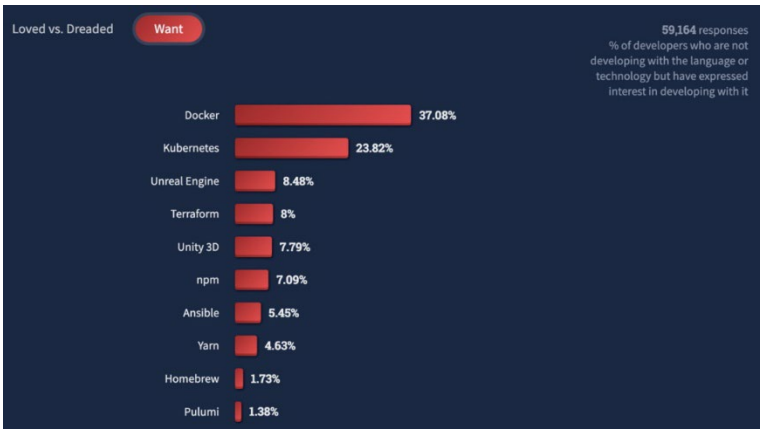


图 9 - Stackoverflow Developer Survey 2022 之大家最想学习的基础工具

顺便说一句，在“爱恨交织”栏目里，同样拥有“一骑绝尘”态势的还有 Visual Studio Code（以下简称 VSCode）。

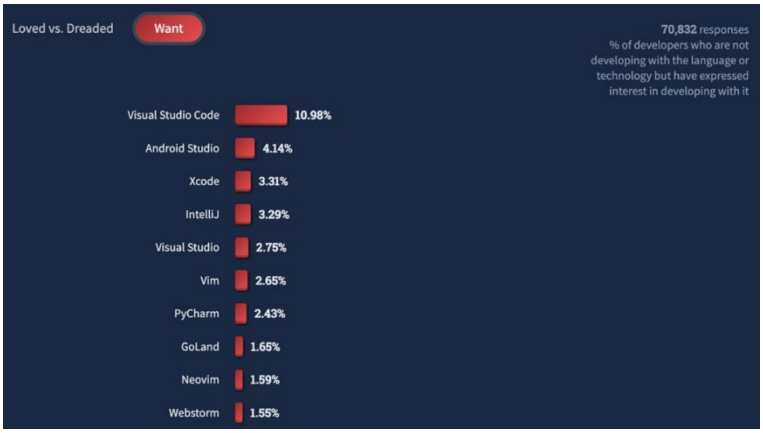


图 10 - Stackoverflow Developer Survey 2022 之大家最想学习的集成开发环境

作者为什么会在这里提到 VSCode 呢？因为，在 2022 年订阅我的技术专栏的读者当中，非 gopher（gopher 的意思是 Go 语言粉丝）的人明显增多。他们问的最多的一个问题就是：“要是使用 Go 语言开发程序的话，有哪些好用的代码编辑器或者 IDE 吗？”

我的回答一般都是 VSCode 和 GoLand。

微软公司出品的 VSCode 是 Go 语言爱好者最常用的免费代码编辑器。我们使用它时，外加 Go 语言官方团队出品的 vscode-go 插件就基本上可以满足平常的需求了。

而 GoLand 是 JetBrains 公司出品的，但它收费而且价格不菲。作者在这里绝不是给 JetBrains 公司做广告，但是 GoLand 真的很贵、很好用。如果你对于配置编程工具非常不耐烦，并且预算很充足的话，那么我会强烈推荐你使用 GoLand。

顺便说一句，如果你是 Vim 那一派的话，Go 社区中也有相应的 vim-go 插件。

好了，关于 Go 语言发展趋势的整体解读，我们就暂告一段落吧。下面，我们从更新迭代的角度来看看 Go 语言在 2022 年都有哪些进展。

年度回顾

支持泛型

Go 语言在 2022 年最大的变化莫过于添加了对泛型（generics）的支持。这对于 Go

语言的广大使用者来说，也是感受最明显的变化。

早在 2022 年 2 月正式发布的 1.18 版本，Go 语言就包含了类型参数提案（即 Type Parameters Proposal）中描述的通用功能的实现。这包括了需要对 Go 语言本身进行的主要的更改。注意，这些更改是完全向后兼容的。这也是 Go 语言官方对于 Go 1.x 版本更新的一贯原则和保证。

Go 语言官方团队为此编写和修改了大量的代码。虽然这些代码并未经过超大规模的验证，但是 Go 语言团队对它们的质量和功效是非常有信心的。不过，也不排除这种可能：一些使用泛型的代码适用于 1.18 版本，但在以后的版本中会变得无效，或者说需要稍加修改才能使其有效。因为，这次为了支持泛型的改动实在是太大了。按照 Go 官方的说法，这种不完全兼容的可能性会被尽可能减小，但是无法对此做出 100% 保证。

虽然我们都喜欢 100%确定的东西，但是万事万物都不可能有 100%的稳定性和可预测性。尤其是，对软件开发有一定理解的朋友们肯定都知道，没有哪一个软件是没有 bug 的，也没有哪一个软件功能可以保证 100%的正确。所以，我们需要用更加宽容的心态来看待 Go 语言的这次超大级别的更新。

实际上我们也不用太过担心。因为从 Go 语言的 issue 列表上看，泛型相关的 bug 如今也已经发现和修复得差不多了。Go 语言的泛型已经趋于稳定，我们已经可以放心地将其用在生产级代码上了。

从语法上说，Go 语言的类型参数（可以理解为“泛型”的另一种称谓）并未使用那个在其他编程语言中常见的尖括号（即“<”和“>”），而用的是方括号（即“[”和“]”）。

这也是程序员们感受的最大一点不同。

请注意，我们在这里所说的“对泛型的支持”实际上是“对自定义泛型的支持”。因为 Go 语言内置的一些数据类型从一开始就是支持泛型的，比如：

```
var map1 = map[int]string{}
```

或者

```
var slice1 = []string{}
```

然而，在 Go 1.18 之前，使用者们自己编写的函数或者类型是无法添加泛型定义的。

下面重点来了。在 Go 1.18 中，对泛型的具体支持都有哪些呢？请看下文。

1. 自定义的函数声明或类型声明都可以包含类型参数。

请看如下代码：

```
type Pair[K ~int, E ~string] struct {
    Key  K
    Elem E
}
```

所谓的类型参数声明，其实就是对一个类型所涉及到的关联类型的泛化定义。例如，对于结构体类型中的每个字段，我们都必须分别定义其类型。这些字段的类型就是它们所属的结构体类型的关联类型。

这里的泛化定义的意思是，我们在声明一个类型的时候，并不指定它的关联类型具体是哪一个或哪几个，而只定义相应的范围。等到这个类型被实例化的时候，再由那个

编写实例化代码的程序员去设定类型参数的具体信息。这样的话，我们就可以定义出更加抽象的类型，而把具体化的工作留给使用它的人。

这与“声明接口类型，并把其作为某些函数参数或变量的类型”的编程手法有着异曲同工之妙，都是为了提升代码的灵活性，并增强其扩展的能力。不过需要注意的是，类型参数的值与函数参数的值或变量的值是不同的。一个类型参数的值必须是代表了某个已存在类型的标识符（或者说类型字面量）。另外，类型参数值代表的既可以是结构体这样的具体类型，也可以是接口那样的抽象类型。而函数参数或变量的值，则必须是某个具体类型的确切值。

另一方面，Go 语言的类型参数声明与它的函数参数声明是类似的。在上述代码的方括号中，K 和 E 分别是两个类型参数的标识符，类似于函数参数的名称。而 ~int 和 ~string 则分别是两个类型参数的类型约束（type constraint），类似于函数参数的类型声明。至于在 int 和 string 的前面为什么会有“~”这个符号，我们稍后再说。

正因为结构体类型 Pair 的声明里包含了类型参数 K 和 E 的声明，所以在它的主体当中，我们自然可以自由地使用 K 和 E。如代码所示，我们把 Pair 的字段 Key 的类型声明为 K，并把字段 Elem 的类型声明为 E。这样一来，Pair 的类型参数就与其主体实现了联动。这种联动将会在我们对 Pair 类型进行实例化的时候得以体现。

2. **对于带有类型参数的函数或类型，可以通过在它们的名称后面追加方括号并指定相应的类型参数值来进行实例化。**

示例如下：

```
var pair1 Pair[int, string] = Pair[int, string]{
```

```
    Key: 1,
    Elem: "a",
}
```

我们在这里声明了一个 Pair[int, string]类型的变量 pair1，并把一个 Pair[int, string]类型的值赋给了它。请注意，我们在对一个带有类型参数的类型进行实例化的时候，也必须对它的类型参数进行实例化。在这里，Pair[int, string]中的 int 和 string 就是分别对 Pair 的类型参数 K 和 E 的实例化。

还记得吗？我们当初在声明 Pair 类型的时候，把它的类型参数列表编写成[K ~int, E ~string]。其中，~int 是类型参数 K 的类型约束，而~string 则是类型参数 E 的类型约束。那么，这里的 Pair[int, string]中的 int 和 string，分别作为 K 和 E 的值就是合法的，可以通过编译。至于为什么，我们马上就会说到。

先接着看其余的代码。因为在 Pair 类型的声明当中，字段 Key 的类型声明是 K，字段 Elem 的类型声明是 E。所以，在实例化 Pair[int, string]的时候，我们自然就可以把某个 int 类型的值（这里是 1）赋给 Key，并把某个 string 类型的值（这里是"a"）赋给 Elem。

3. 新符号“~”已被添加到了运算符和标点符号的集合中。

我们再看 Pair 类型的声明：

```
type Pair[K ~int, E ~string] struct {
  Key  K
  Elem E
}
```

我们大可以把这里的符号“～”理解为“潜在”。代码“K ~int”的意思是，只要一个类型（假定为 A）的潜在类型是 int，那么就可以满足我们在这段代码中对 K 所做的类型约束，这就意味着 A 类型的字面量可以作为类型参数 K 的值。同样的道理，代码“E ~string”的意思是，只要一个类型（假定为 B）的潜在类型是 string，那么就可以满足我们在这段代码中对 E 所做的类型约束，这就意味着 B 类型的字面量可以作为类型参数 E 的值。也正因为如此，类型 Pair[int, string]才是合乎语法规则的，它的类型参数都已通过了有效的实例化。

至于什么是“潜在类型”，Go 语言规范对此有明确的解释。具体内容是：每个类型 T 都有一个潜在类型。如果 T 是 Go 语言内置的布尔类型、数字类型、字符串类型之一，或者是某个类型字面量，那么相应的潜在类型就是 T 本身。否则，T 的潜在类型就是 T 在其声明中引用的类型的潜在类型。

下面举个例子。如果我们又编写了如下代码：

```
type MyInt int
type MyStr string
```

那么，对于当前的 Pair 类型声明来说，下面的代码也是合法的：

```
var pair2 Pair[MyInt, MyStr] = Pair[MyInt, MyStr]{
  Key: 2,
  Elem: "b",
}
```

更确切的说，类型 Pair[MyInt, MyStr]是合乎语法规则的。因为，从前面的说明和代码可知，MyInt 的潜在类型是 int，而 MyStr 的潜在类型是 string。它们分别符合 Pair 类型的声明里对类型参数 K 和 E 的定义。

4. **接口类型的声明中现已允许嵌入任意类型，以及由符号“｜”联结的联合类型和由~T 代表的类型元素，而不只是之前允许的其他接口类型。不过，这样的接口只能用于泛型中的类型约束。**

这段话是什么意思呢？我们来详细解析一下。

为了配合 Go 语言对泛型的支持，官方团队对接口类型的声明语法做了很多的增强。

使用 Go 语言的程序员们都知道，以前的接口声明只能像下面这样：

```
type Reader interface {
 Read(p []byte) (n int, err error)
}
```

或者这样：

```
type ReadCloser interface {
 Reader
 Close() error
}
```

也就是说，在接口类型声明的主体中，我们可以嵌入任意数量的非重复的方法声明，也可以嵌入任何其他非重复的接口类型（用接口名称来代表）。我们称这两者为合法的接口元素。但除此之外，我们就不能添加任何东西了。

然而，从 Go 1.18 开始，合法的接口元素又多了一种。Go 官方把这种接口元素称为类型集合（type set）。

一个类型集合可以仅包含单独的类型（由类型的名称代表，如：T），也可以包含代

表了某个潜在类型的～T，还可以是联合类型（如：T1｜T2｜T3，其中的符号“｜”可以被理解为“并集”），甚至可以是它们的混合（如：T1｜～T2｜～T3）。而且，对此我们可以分多行来写，只要它们所代表的类型是具体的且不存在重复即可。

不过要注意，包含了类型集合的接口类型只能被用在泛型相关的类型约束当中。例如，有如下代码：

```
type FloatUnion interface {
 ~float32 | ~float64
}
type FloatMaker interface {
 FloatUnion
 MakeZero()
}
```

可以看到，含有类型集合的接口 FloatUnion 是可以被嵌入到其他接口类型的声明里面的（或者说，其他的接口类型可以扩展 FloatUnion 接口）。但如此一来，不但 FloatUnion 接口不能被用作任何值、变量、字段、参数、结果的类型，而且 FloatMaker 接口也会是这样。换句话说，对这种接口的用途限制具有传递性。

5. 新的内置标识符 any 是空接口的别名。它可以代替 interface{}。

这一条说得很直白。单词 any 现在属于关键字了。它代表了空接口，也就是 interface{}。但是，空接口本身的含义却因泛型支持的出现而增多了。

从 Go 1.18 开始，空接口自带类型集合，并且它的类型集合包含了所有的非接口类型。注意，这里的“所有”不但代表当前已存在的所有非接口类型，而且还囊括了将来会在程序中出现的所有非接口类型。也就是说，空接口的类型集合拥有无限多的非接口类

型。

这与空接口的设立初衷是一致的，即：空接口是包罗万象的，也是类型之树的唯一树根。在 Go 语言中，任何接口都是空接口的扩展接口，任何类型都是空接口的实现类型。这样来看，任何类型，不论是抽象类型还是具体类型，都是对空接口所代表的类型空间的进一步圈定。

对于类型参数中的类型约束来说也是这样。空接口的类型集合包括了无限多的非接口类型，这使得任何类型约束所代表的类型集合都将是空接口的类型集合的一个子集。这是“进一步圈定”的另一种表现形式。因此，空接口在 Go 语言全面支持泛型之后，依然能够作为其类型系统的根基。

6. 新的内置标识符 comparable 也代表一个接口类型。

顾名思义，comparable 接口的含义是“可比较的”。只要一个类型符合以下两种情况中的一种，我们就可以断定它实现了 comparable 接口：

- 这个类型不是接口类型，并且它的值可以通过使用操作符==或!=进行比较。
- 这个类型是接口类型，而且其类型集合中的每一个类型都实现了 comparable 接口。

比如，像 int、float32、rune、string 这样的基本类型肯定都实现了 comparable 接口，而切片（slice）类型、字典（map）类型以及任何的函数类型肯定就不是 comparable 接口的实现类型。

再比如，我们在前面声明过的 FloatUnion：

```
type FloatUnion interface {
```

```
 ~float32 | ~float64
}
```

可以确定它肯定实现了 comparable 接口。但如果我们把其中的~float64 替换为~[]float64，那么它就不再是 comparable 接口的扩展接口了。

请注意，comparable 接口，以及任何直接或间接地嵌入了（或者说扩展了）comparable 的接口都只能被用于类型约束。它们不能被用作任何值、变量、字段、参数、结果的类型。

显而易见，与 any 接口一样，comparable 接口也是专门为了类型参数（或者说泛型）才引入的。同样的，comparable 接口也自带了类型集合。它的类型集合包含了所有可以被比较的类型。这样的类型既可以是已经存在的，也可以是尚未出现的。

除了上述 6 个很重要的改动之外，Go 团队还为使用者们准备了 3 个实验性质的代码包。这些代码包并不在 Go 标准库当中，而是位于 Go 语言官方专门设立的实验包 golang.org/x/exp 里。这意味着，它们的 API 并不在 Go 1.x 的兼容性保证范围之内。并且，随着 Go 团队对泛型支持的进一步深入，这些代码包也可能会发生非向后兼容的变化。具体如下：

- 代码包 golang.org/x/exp/constraints：其中包含了对泛型编程非常有用的一些类型约束，如 constraints.Ordered 接口等等。
- 代码包 golang.org/x/exp/slices：其包含了不少对于切片操作非常有用的函数。而且，对于这些函数所操作的切片，其元素类型可以是任意的。比如，泛型函数 func BinarySearch(x []E, target E) (int, bool)、func CompactS ~[]E, E comparable S、func SortE constraints.Ordered 等等。从这些函数的签名上我们就可以看出，

它们的通用性都得益于泛型。这样的通用性在 Go 语言支持泛型之前都是不可能存在的。

- 代码包 golang.org/x/exp/maps：与 golang.org/x/exp/slices 包类似，其中包含了一些针对字典的非常通用的泛型函数，如：func ClearM ~map[K]V, K comparable, V any、func CloneM ~map[K]V, K comparable, V any M、func KeysM ~map[K]V, K comparable, V any []K 等。

真正了解 Go 语言的程序员们肯定都知道，Go 团队经常会向 golang.org/x/exp 包中放入一些实验性的代码。他们往往会通过这种方式来实现一些或新鲜或激进的想法。如果某些代码在这里通过了使用者们的检验，并被认为已经足够成熟，那么它们就有希望被添加到 Go 语言的标准库当中。Go 团队正是依靠这种渐进式升级的方式，在保证标准库稳定的同时，使其创新性得以延续。

再说回泛型。尽管 Go 语言团队为了泛型做了如此多的工作，但到目前为止，Go 语言的泛型实现仍然存在一些小限制（主要体现更加细致的编程语法、值成员访问等方面）。不过，这些小限制在大多数情况下并不会妨碍我们在应用程序中使用泛型。而且，Go 语言团队也已经预告将在未来的版本中对此进行改进。所以，作者就不在这里一一列举了。

到这里，相信大家已经有所体会，**“支持泛型”可以说是 Go 语言正式发布以来最大、最复杂且最重要的一项变化了**。很显然，Go 语言本身的泛型支持工作离彻底完成还有一小段距离。而**对于 Go 语言的技术社区来讲，更加重要的是，这项变化将意味着 Go 语言生态系统的大规模翻新**。

到目前为止，Go 语言的生态系统已经非常庞大。因此，Go 语言的这项变化将会给

Go 社区带来很可观的压力。那些 Go 程序员们常用的第三方开发框架和工具必然需要一定的时间才能够跟进这项变化，而完美契合这项变化也许还需要更多的时间。这其实也是 Go 团队当初在考虑“是否添加泛型支持”的时候，涉及到的一个很重要的负面因素。

但无论如何，Go 语言在这件事情上的第一步（也是非常重要的一步）已经迈出并平稳落地了。我们现在只希望，Go 语言以及 Go 语言技术社区能够在这个良好的基础之上继续稳步前行、平滑过渡。

模糊测试

我们都知道，Go 语言原生支持的测试已有三种，即：功能测试、基准测试（或称性能测试），以及示例测试。而从 1.18 版本开始，Go 语言本身支持的测试又多了一种，那就是模糊测试（fuzz test）。

所谓模糊测试是一种自动化测试技术，这种测试可以通过不断地调整应用程序的输入值来试图查找应用程序内部可能存在的错误，尤其是那些我们平常不太能注意到（或称边缘情况）的错误。正因为不太能注意到，所以在编写测试代码的时候，我们往往会有意或无意地忽视掉它们。这个时候，模糊测试就可以被用来查缺补漏了。而且，这种查缺补漏也是非常重要的，特别是在面向安全的测试当中。

这里所说的模糊，其含义是对应用程序的输入值（或者说参数值）的模糊。更确切地讲，模糊测试程序会对应用代码（如函数、结构体、接口方法等）进行若干次调用。而每一次调用都会在预先准备好的语料库（corpus，此为模糊测试的专用术语）当中随机地挑选出一个条目（corpus entry），并把该条目作为参数值输入给应用代码。

请注意，模糊测试的语料库与我们熟知的普通单元测试（如功能测试、基准测试等）的测试用例在生成方面有一个很大的不同。在普通的单元测试中，测试用例都是需要我们专门去准备的。这也算是普通单元测试的一种局限性。因为测试用例的规模与我们的测试工作量是成正比的，而且起码是线性的关系。我们越想全面地测试应用代码，我们的工作量就越大。由于我们的精力和时间是有限的，而且也不可能无限期地测试下去，因此普通的单元测试在测试覆盖度方面就会存在天然的限制，而且很难突破。

而在模糊测试中，语料库的生成是自动化完成的。模糊测试程序会自动生成一定规模的语料库。而我们只需事先向它提供一个规模非常小的种子语料库（seed corpus），甚至只包含一两个条目就可以。

这个种子语料库存在的目的，只是为了指导模糊测试引擎对语料库的自动生成，比如，单个语料库条目里需要包含几个输入值，以及每个输入值都是什么类型的，诸如此类。假设，作为测试目标的函数 A 只有一个参数，那么种子语料库中的每个条目只包含一个值即可。如果作为测试目标的函数 B 有两个参数，那么种子语料库里的单个条目就应该包含两个值。以此类推。

当然了，没有任何一个工具是完美的。虽然模糊测试可以大大地节省我们的测试工作量，但它也有一个小缺点，那就是：我们无法对语料库提供的参数值进行精细的控制。更确切地说，在模糊测试的过程中，我们不能确定应用代码每次被调用时接收到的输入值具体是什么。在某些情况下，这可能会妨碍我们对应用代码返回的输出值（或者说结果值）的正确性判断。倘若确实需要，我们可能就要额外添加一些测试代码来做专门的判断了。如果这样的情况很多，那么我们可能还需要重新权衡是否一定要使用模糊测试，而不是用普通的功能测试或基准测试。

好了，到了这里，想必大家已经大致了解模糊测试是什么，以及我们大概应该如何运用它。至于具体怎样给 Go 程序做模糊测试，其实还是很简单的。它与之前已经存在的功能测试和基准测试的编写方式大同小异，我们稍微迁移一下以前的测试经验就基本上可以搞定。因此，作者在这里就不再赘述了。如果你想对此做深入了解，那么可以去访问 fuzzing landing page 以及 the fuzzing proposal。

工作区模式

Go 语言的工作区模式（go workspace）是建立在 Go 模块模式（go module）之上的一种项目管理模式，主要体现在 go 标准命令之中。

Go 模块模式在 2021 年发布的 Go 1.17 中已经完全稳定下来了。我们可以通过 go mod 命令来创建和管理我们的 Go 模块。如果我们开发的 Go 项目已经达到了一定的规模，那么往往就需要把它拆分成多个模块，以便让它继续保持高内聚、低耦合的状态。这对于代码和项目的维护都是相当有利的。

比如，我们有一个名叫 MyProject 的 Go 项目（同时也是一个 Go 模块）。它的存放路径是/path/go/haolin/Demo/MyProject。现在，我们想把其中的一些基础代码分离出来，以便进行独立的开发和维护。我们仔细地抽出那些基础代码，并它们放入到了一个名叫 MyLib 的模块中，并同样存放在/path/go/haolin/Demo 目录之下。

这两个模块分别有自己的 go.mod 文件。MyProject 模块在它的 go.mod 文件中的名称为 github.com/hyper0x/MyProject，而 MyLib 模块在它的 go.mod 文件中的名称为 github.com/hyper0x/MyLib。不过请注意，这两个模块并不实际存在于网址 https://github.com/hyper0x 之下。这里只是作为示意而已。

在这之后，我们在 MyProject 模块里导入并使用 MyLib 中的代码包。但当我们想编译 MyProject 模块的时候却发现，go 命令无法正确编译 MyProject，提示找不到 MyLib 里的那个代码包。碰到这种情况，我们需要在 MyProject 所在的路径下运行：

```
$ go mod edit -replace github.com/hyper0x/MyLib=../MyLib
```

这条命令的主要功能是，在 MyProject 模块的 go.mod 文件中加入如下内容：

```
replace github.com/hyper0x/MyLib => ../MyLib
```

其含义是，将本模块中的源码文件里的 github.com/hyper0x/MyLib 定位到本地的路径（即/path/go/haolin/Demo/MyLib）之上。

做好以上准备工作后，我们再在同样的路径下运行：

```
$ go get github.com/hyper0x/MyLib
```

其功能是，在对应的 go.mod 文件中添加相应的依赖包信息。

好了，现在我们再次编译 MyProject 模块就不会有问题了。因为我们已经把本地的 MyLib 模块正确地设置为了 MyProject 模块的依赖包。

不过，这里有一个问题，上述的 replace 指令中存在一个本地的路径。如果开发这两个模块的人不止我一个，也就是说需要通过多人协作来开发项目，那么，我这里的本地路径在别人那里就是不正确的了。因为不同的开发者几乎不可能确保把同一个项目存放在自己的计算机中的相同路径之下。我们常常称这类问题为“本地路径问题”。

这个时候，Go 语言的 workspace 模式（也就是工作区模式）就派上用场了。具体的做法是，我们在/path/go/haolin/Demo 这个路径（也就是 MyProject 模块和 MyLib 模

块的上一级目录的路径）之下运行：

```
$ go work init
$ go work use ./MyLib
```

这里的第一条命令的功能是，把路径/path/go/haolin/Demo 所代表的目录设置为 Go 工作区。它会在这个路径下添加一个名为 go.work 的文件。第二条命令的功能是，让子目录./MyLib 所代表的模块成为该工作区的一个共用模块。如此一来，该工作区中的其他 Go 模块就可以直接导入并使用 MyLib 模块中的代码包了。

此外，我们还需要在 MyProject 模块所在的路径下运行命令：

```
$ go mod edit -dropreplace=github.com/hyper0x/MyLib
```

并以此删除掉对应的 go.mod 文件中包含的那个带有本地路径的 replace 指令。由于我们前面针对（/path/go/haolin/Demo 路径所代表的）相应 Go 工作区的设置，这里的这个 replace 指令的消失并不会造成任何问题。MyProject 模块依然可以通过编译。

至此我们可以看到，由于 Go 工作区模式的出现，我们的 Go 项目无论包含有多少个模块，也无论需要多少人进行怎样的协作调试和开发，都不用担心出现“本地路径问题”。这给团队级别的项目开发，尤其是大规模的协作开发，带来了非常大的便利。我们终于不用再为此频繁地修改 go.mod 文件了。这个 Go 项目管理上的痛点终于被消除了！

内存模型更新

在 1.19 版本中，Go 语言的内存模型已经过了修改（详见 the Go memory model）。

这使得它与其他编程语言如 C、C++、Java、JavaScript、Rust 和 Swift 使用的内存模型保持了一致。

Go 语言只提供可以保证访问顺序一致性的原子操作，而不是像其他编程语言那样使用更加宽松的形式。更确切地说，在 Go 语言中，除非直接或间接地使用了相应的原子操作，否则并发地读写共享的数据必会引发数据竞争（data race）。顺便说一句，从根本上讲，Go 语言标准库中提供的其他同步或异步的数据访问工具基于的其实都是原子操作。

随着内存模型的更新，Go 1.19 在 sync/atomic 包中加入了一些新的类型，如 atomic.Bool、atomic.Int32、atomic.Int64、atomic.Uint32、atomic.Uint64，以及 atomic.Uintptr 和 atomic.Pointer[T]。很显然，与 sync/atomic 包中原有的诸如 AddXXX、CompareAndSwapXXX、LoadXXX 之类的原子函数相比，新的原子类型可以帮助我们更加方便和彻底地运用原子操作，同时也可以避免很多容易发生的编程失误。不得不说，我们已经等待这些原子类型很久了。

到这里，作者已经对 Go 语言在 2022 年的重大更新进行了相应的说明。这其中的重中之重肯定是泛型，所以作者对此的着墨也明显更多。至于那些在作者看来不那么重要或相对较小的更新，大家可以去浏览 Go 语言官方出具的版本说明（Go 1.18 Release Notes 和 Go 1.19 Release Notes），这里就不一一列出了。

未来展望

实际上，与 Go 1.18 相比，Go 1.19 算是一个更新量很小的版本迭代。这主要是因为，

在影响巨大的特性加入之后，Go 语言及其团队和社区确实需要一段时间去消化它。这里其实有很多的工作要做，比如：进一步评估和验证、功能调整和修补、社区推广和反馈收集、代码优化和改进、文档补全和细化，等等。

不过，在明年即将发布的 Go 1.20 当中，随着上述工作的稳步推进和趋于完善，还是有一些值得关注的新东西的。下面只列举作者认为最喜闻乐见且可能最常用的几个更新：

- **支持同时囊括多个其他错误值的单个错误值**：届时 errors 包中会提供一个新的函数 Join，用来把多个错误值包装成一个单一的错误值。另外 fmt.Errorf 函数也将允许在其模版字符串（即第一个参数值）中出现多个（用于包装错误值的占位动词）%w。函数 errors.Unwrap 到时候也会以[]error（即 error 的切片）作为结果值的类型，而不是之前的 error，以便一次性地返回所有被包装的错误值。这无疑大大增加了错误包装代码的灵活性，并间接地加强了错误判断代码的功能。
- **sync.Map 的增强**：sync.Map 类型里将会出现几个新方法，如 Swap、CompareAndSwap、CompareAndDelete 等。这些方法将如 sync/atomic 包里的相应函数那样，可以让单一的原子操作中包含多个动作。在 sync.Map 的场景下，这将支持针对键值对的更加复杂的原子操作，如上述方法所对应的比较并更新、比较并删除等。使用过 sync.Map 的程序员们应该都知道，对于这种操作，我们在之前不得不使用额外的同步工具来保证其并发的安全性。
- **支持从切片到数组的转换**：我们都知道，把数组转换成切片很容易，只要在一个数组上执行切片操作就可以得到一个指向那个数组的切片。这样的切片就相当于一个架设在数组之上的窗口。然而，现在的 Go 语言却不支持对应的反向操作。虽然我们通过少许的魔法代码就可以实现这样的操作，但是它肯定没有“array2 :=

[5]int(slice1)”这样的代码方便。后者正是将会在 Go 1.20 中合法化的代码，只一行就能完成从切片到数组的转换。不过要注意，上面这小段代码只会返回 slice1 的底层数组的副本，而不是这个底层数组本身。所以，我们之后对 slice1 中元素的修改将不会影响到 array2（但肯定会继续影响到 slice1 的底层数组）。

顺便提一下。已经在 Go 社区中引发热议的代码包 arena 及其配套代码可能会在 1.20 版本中以实验特性的方式出现。作者很看好 arena，因为它可以在某些情况下大大地减少内存分配和释放的次数，从而显著减轻 GC 的压力。

在作者看来，arena 包的功能类似于 Zig 编程语言中的内存分配器（allocator）。不过相比之下，Go 语言的实现肯定是大大简化的，因为它在 Go 语言中只需起到一定的辅助作用即可。整体来看，其做法是，在需要使用大块内存时先通过 arena 包中的 API 进行显式的内存空间申请，然后再根据实时的需要对其中的子空间进行取用，最后当无需再使用这块内存的时候再利用 arana 中的 API 进行整体释放。简单来说，这属于一种“整体化分配、碎片化使用、整体化释放”的内存空间使用范式。关于它更详细的说明可参看 Proposal: arena: new package providing memory arenas。

除了上述这些，Go 1.20 还会对编译器、标准工具、标准库等方面进行诸多的改进，详情可查看 Go 1.20 Release Notes(DRAFT)。如果在你阅读这篇文章的时候，Go 1.20 已经正式发布了，那么就可以直接去看 Go 1.20 Release Notes。

总结

好了，我们现在来快速地总结一下。

Go 语言在 2022 年的最大更新莫过于对自定义泛型类型和泛型函数的支持。新增的模糊测试 API 和工具将显著减少程序员们的测试工作量。工作区模式让我们在同时开发多个 Go 模块，尤其是多人协作开发的时候方便了许多（解决了一个很明显的痛点）。而在内存模型方面的修改则进一步增强了 Go 语言在数据访问方面的一致性保证。

在 2023 年，Go 语言将继续全方位地优化和改进其在各个方面的功能和性能。我们在前面已经列举了 Go 官方团队在错误处理、并发编程、便捷语法、内存管理方面的计划。作者相信，这些计划将会有条不紊地进行，并在将来让 Go 语言成为更加优秀、更加流行的编程语言。

希望大家继续关注 Go 语言和 Go 技术社区，尤其是国内的 Go 技术社区。同时，作者也希望能有更多的小伙伴参与到 Go 社区的建设甚至 Go 语言本身的改进当中去。如果正在阅读本文的你还没有尝试过 Go 语言，那么别再犹豫了，赶快到这里下载 Go 语言，并开始尝试编写 Go 程序吧！

1. Go 1.18 Release Notes: https://go.dev/doc/go1.18
2. Go 1.19 Release Notes: https://go.dev/doc/go1.19
3. Go 1.20 Release Notes(DRAFT): https://tip.golang.org/doc/go1.20
4. 其他相关的 Go 语言官方文档
5. Go 语言源码及其修改记录

作者简介

郝林，国内知名编程布道者、技术社群 GoHackers 的发起人和组织者，微信公众号“螺旋码”（视频号同名）主理人。发布过多个 Go 语言技术教程，包括开源的《Go 命

令教程》、极客时间的付费专栏《Go 语言核心 36 讲》，以及图灵原创图书《Go 并发编程实战》，等等。其中专栏和图书拥有数万订阅者或购买者，开源教程 star 数也有数千。另外，他还在 2020 年出版了一本名为《Julia 编程基础》的技术图书。

大数据管理工具走向更易用与专精，但中外发展现状存在明显差异 | 解读大数据的 2022

作者：王宇飞

> 本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦大数据管理工具在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年大数据领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

作者：王宇飞，字节跳动数据平台开发套件技术负责人

以云数仓为中心的数据架构开始形成

数仓的发展历程

纵观整个数仓的发展历程，笔者认为大体上可以分为 4 个阶段：

- 第 1 阶段：传统的一体机数仓，代表产品有 Oracle Exadata、IBM Netezza 等；
- 第 2 阶段：基于 MPP 架构的数仓，它可以实现软硬件分离，数据不用再存储在特定的服务器上，代表产品有 GreenPlum、Vertica 等；
- 第 3 阶段：随着互联网发展，企业数据量迅速增长，擅长处理海量数据的 Hadoop 框架和生态得到了普及与发展，诞生了依托 Hadoop 生态的批处理等模式，代表产品有 Hive；
- 第 4 阶段：伴随云的发展，云上数据仓库解决方案日趋成熟，进一步降低了用户使用成本，这个阶段的代表产品有 Amazon Redshift、Snowflake 等；

尽管在当下，第 2、3、4 阶段的数仓产品仍处于共存状态——它们仍旧分别被各类企业所使用，但在企业数字化转型和企业上云的整体趋势背景下，我们有理由相信，未来处于第 4 阶段的云数仓，将会被越来越多的企业采用。

云数仓架构的核心优势

在笔者看来，云数仓架构最显著的优势在于易用性进一步提升、成本进一步降低。尤其是新一代云数仓采用 shared everything 架构，所有节点共享底层云存储资源池，计算与存储分离，让存储及计算节点都具备灵活弹性扩展的能力。此外，可以以集中

化的方式解决安全信任和数据共享等方面问题。

相比基于 Hadoop 生态的数仓架构，用户不用再维护管理自己的集群，而是可以根据业务发展需要动态扩缩容。综合考虑硬件及自身维护数仓架构的研发等成本来看，云数仓也更有优势。

云数仓开始成为企业的 single source of truth

随着过往数据生态系统的发展，企业其实已经积累了非常多的数据源，包括各种数据库、SaaS 服务等。尤其在海外，SaaS 发展更成熟，多源数据问题更加凸显。

多源数据会给企业带来两方面的困扰：

1. 数据源之间点对点集成方式形成依赖，交织成复杂的网，维持数据间的同步越发困难；
2. 数据源之间都很孤立，形成一个个数据烟囱，数据很难整合起来发挥最大化价值。

因为云数仓的存储计算成本优势，以及随着现代数据堆栈的发展，让数据汇聚到云数仓及加工都更加便捷，这些都是企业选择云数仓作为中心存储的优势。

当云数仓成为企业的 single source of truth 后，可以帮助企业基于整合后的数据更好地做出业务决策。

数据工具更加易用与专精

数据管理成为企业基础设施增长最快的领域之一

A16Z 今年发布了世界排名前 50 的数据创业公司 https://future.com/data50/，文章中指出：数据管理已经成为企业基础设施中增长最快的领域之一，价值预计超过 700 亿美元，2021 年数据管理支出占企业所有基础设施支出的五分之一以上。这 50 家创业公司的估值总计超过千亿美元，总融资额约 145 亿美元，其中 20 家在 2021 年晋级独角兽行列。

笔者认为，背后最主要的原因是数据使用的民主化对帮助企业快速成长起到了愈发关键的作用。这里面包括增长营销、业务诊断、结合机器学习赋能业务过程自动化等等，早已不再只是单纯的报表统计需求。

而海外云数仓的发展，尤其像 Snowflake 的发展与崛起，进一步带动了整个大数据管理工具的生态发展，形成了目前的 Modern Data Stack。Snowflake 作为云中立的数仓，可以做到足够的开放兼容，利用生态的发展增强整体数据管理链路的竞争力。从其对生态公司的投资布局也能看出，涵盖了从数据转换、元数据管理到数据安全等多领域公司。

更多细分创新领域涌现，下一代行业标准正在悄然建立

新一代数据集成系统：ETL 向 EL（T）转变

过去，整个数据开发的基础核心链路是 ETL，即数据抽取加工后导入数仓。随着硬件成本的降低与云的发展，处理模型开始向 EL（T）的方式转变。

EL（T）的优势在于把数据抽取和转换解耦，也就是 EL 和 T，各自由独立组件完成，使得抽取过程可以更自动化，大大降低了数据导入这步的使用门槛。同时，由于 EL（T）可以让数据源更低成本地集成进来，进一步加速了云数仓成为 single source of truth 的进程。

而做好一个数据集成系统主要的挑战有三点：

1. 架构可扩展性，不光是数据源 connector 可插拔，还有不同场景的集成能力，包括批、流和 CDC；
2. 丰富的基础功能，作为一个垂直领域，数据集成存在很多独特的问题，比如脏数据、数据源承载能力、数据源类型不一致、数据源读写延迟、任务读写监控等；
3. 系统稳定性，这对于数据量大的企业而言是非常明显的痛点，比如如何保证几十万任务的海量数据传输；
4. 数据质量，如何做好数据的不重不丢，及异常数据的及时发现等问题。

从行业发展来看，数据集成领域涌现了一些优秀的开源项目，每款产品都有着自身的特点，并聚焦解决其中一部分问题，以下是几种常见的解法：

- 将易用性以及用户体验做到极致，但架构比较简单，不支持分布式部署，容错性一般，处理能力以及场景比较受限；
- 提供了丰富的数据集成基础功能，并将资源使用率做到极致，架构上基于单机多线程架构，不支持分布式和实时功能，使用场景比较受限；
- 主打用户使用场景，支持离线、实时数据同步，并提供分布式的数据处理能力。但引擎资源开销较大，基础功能支持比较一般。

字节基于内部的复杂场景和海量数据传输等需求，自研了一套提供离线、实时、增量场景下全域数据集成解决方案引擎，在架构及功能层面的优化包括：

- 架构灵活性：connector 层、框架层、引擎层，每一层都采取可插拔的设计，同时 connector 层、框架层做到完全与引擎解耦，整体结构比较清晰和灵活，可扩展性较强。可以应对不同规模的数据同步场景，同时具备单机以及分布式能力，在引擎的选择上采取智能选择策略，兼顾小数据量场景的资源利用率以及大数据量场景的稳定性；
- 集成能力完善性：在框架层提供了丰富的基础能力，比如异构数据源类型转换、脏数据处理、流控、自动并发度推断、任务运行监控、事件时间归档等；
- 企业级稳定性：在特定导入场景，提供了更多的企业级特性，在性能、稳定性、成熟度上更有保障。比如 Kafka2Hive 通道，针对大数据量场景做了相关优化和重构，包括如何提升大并发场景下一致性快照成功率、提升大并发场景的容错性、兼容乱序写入场景等等。

今年 11 月，我们已经把这套数据集成引擎 BitSail 正式开源出来（https://github.com/bytedance/bitsail），希望可以把内部沉淀的经验提供给更多外部企业，同时援引社区力量，大家共同解决好数据入仓第一步的问题。

数据可观测性：概念骤热背后，数据质量和 SLA 重要性日益凸显

数据可观测性的核心是帮助企业解决更好地监控和改善企业内可交付的数据质量和效率等问题。通过持续监控、跟踪、警报、分析和故障排除事件来减少和防止数据错误或停机时间，同时能帮助用户了解组织的数据环境、数据管道和数据基础设施的健康状况。

目前，Monte Carlo 等厂商正在大力推广“数据可观测性”这一概念。

在笔者看来，数据可观测性概念背后要解决的问题——即数据质量和 SLA 治理问题，并不是全新场景问题，但其重要性正在不断提升。

核心驱动力来自两个方面：

- 数据链路本身的复杂程度升高：随着现代数据堆栈的发展，整个数据链路的建设需要多个不同的 SaaS 产品和服务，极大增加了数据端到端交付质量的运维和管理成本。
- 数据对业务的重要性上升：数据驱动决策越来越普遍，故数据不及时或数据发生错误造成的成本也变得更高。决策对数据质量和实时性的依赖越高，对数据可观测性的需求也会越高。

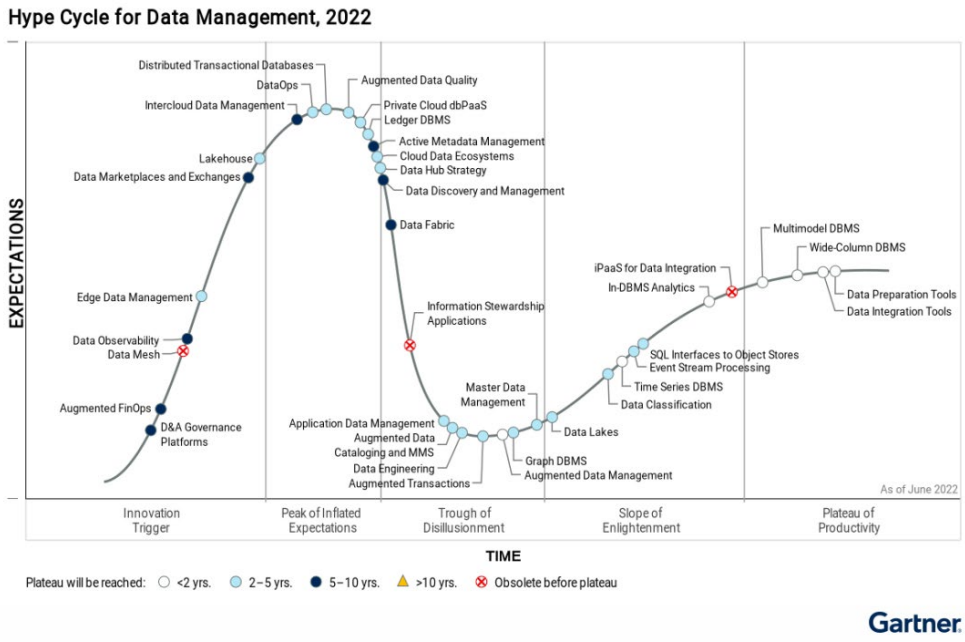
反向 ETL：数据进一步深入业务场景实现提效

反向 ETL 指的是将数据仓库加工好的数据同步到业务系统中，它处于数据集成（EL）的另一端，解决的是数据如何在消费端更好地发挥价值的问题。这种细分场景的出现，

也是以云数仓为中心的场景扩展，意味着数据使用深度增强。

数据从之前的更多用于报表决策、分析等应用场景，开始过渡到更多地输出到企业的 CRM 等系统，帮助优化业务流程和自动化决策等场景能力。这也是数据对业务重要性提升的一个表现。反向 ETL 让用户能够以无代码、自动化等低成本的方式提升业务场景的决策效率。典型的代表公司有 Hightouch、Census 等。

DataOps 兴起，但行业标准尚未成型



图注：Gartner 2022 数据管理成熟度模型

Gartner 对 DataOps 的定义如下：

> DataOps is a collaborative data management practice focused on improving the communication, integration and automation of data flows between data managers and data consumers across an organization. The goal of DataOps is to deliver value faster by creating predictable delivery and change management of data, data models and related artifacts. DataOps uses technology to automate the design, deployment and management of data delivery with appropriate levels of governance, and it uses metadata to improve the usability and value of data in a dynamic environment.

简单来说，DataOps 结合了软件工程 DevOps 理念，来解决数据端到端交付过程中的质量与效率问题，它是一套方法论，而不是单指某项技术或工具。2018 年，Gartner 开始推 DataOps 这个概念，2022 年 DataOps 进入到期望膨胀期，即越来越多的企业开始宣传和投资这个概念。

在笔者看来，DataOps 出现的核心驱动力还是数据对业务的影响和价值越来越大，所以需要更科学、更精细化的方式融入到整个数据开发、生产到交付的全链路中。而 DevOps 已经发展多年，里面很多成熟的理念可以迁移到数据的开发交付链路中，比如持续集成、持续测试、持续监控等等。

但行业整体还处于发展初期，大家对这个概念本身的理解也有差别，还没有真正形成基于普遍认可的业务实践达成的行业标准。从 Gartner 的 Hype Cycle 曲线也可以看出，DataOps 仍需要 2-5 年才能发展得相对成熟。如何在 Data 场景下将 Ops 最优化落地，仍需各个厂商积极探索和完善。从字节的场景需求来看，眼下有以下两方面的

问题是大数据管理套件 DataLeap 在持续投入改进的。

- **研发质量**

这方面的痛点包括如何把口头研发规范应用到开发平台、如何降低由于代码问题或误操作等带来的数据事故，以及如何保障数据 SLA 链路的可观测性等问题。随着一个个场景的问题被解决，沉淀到产品上的能力一定程度上就是软件工程实践的场景迁移。比如，基于数据开发场景结合好 CICD、数据测试、版本管理等。

- **研发效率**

一方面是要制定合理的 DataOps 研发流程，另一方面是如何用更加自动化的方式把各领域工具很好地串联在一起，比如数据建模、数据质量监控、资产管理等目前都是不同的独立产品组件，并没能很好地串联起来。笔者认为海外的现代数据堆栈也有这类问题，虽然每个细分场景有独立产品支撑，提升了单场景易用性，但整体链路串联在一起反而提升了复杂度。目前已经有一些创业公司在尝试解决这个场景的问题，比如 Dataops.live 等。

AI 能力进一步融入，释放更多人力

数据管理工具结合 AI 能力，在笔者看来是必然趋势。究其原因，一方面是人力成本日益攀升，而 AI 可以释放人力，让人力聚焦到更复杂的业务问题上、做更能激发创造力的事情；另一方面是随着数据的民主化，企业收集和处理的数据量远超之前，尤其大型互联网公司的数据规模通常都能达到 EB 级，这种规模的数据管理运维复杂度

也急剧攀升，完全靠人来处理好是比较难的，而 AI 可以完全托管或协助管理。

对于数据管理领域，AI 可以在很多方面发挥作用。比如在数据治理方向，自动检测相似任务、利用 NLP 技术生成数据质量规则、冷热数据分级等；在数据开发方向，自动推断 Join 字段及代码片段等；在安全合规方向，智能审批及异常访问检测等。

从发展阶段来说，AI 在数据管理领域的应用还处于偏早期，大多数场景起到的是锦上添花的作用，仍有很大空间等待探索和挖掘。

结语

当前国内外大数据管理工具的发展现状还存在明显差异，具体体现在：

- 国内的数据管理产品偏 all in one 形式，而国外现代数据堆栈在各领域都有细分的工具支持。以数据集成为例，国外数据转换和任务调度编排等每个领域都有独立的针对这个场景的厂商支持，国内则通常是一个厂商把几个领域场景的问题都解决了。整体上国内更重交付，产品力的打磨相比国外的细分 SaaS 产品存在一定劣势。
- 国内私有化场景交付更多，而国外 SaaS 发展更成熟。国外 SaaS 产品的标准化程度更高，因此用户选择不同细分 SaaS 产品拼接组合的成本也更低。

核心原因是国内外云的发展阶段不同，云的渗透率相差比较大，这也导致国内数据工具生态还没有发展起来。而现阶段国内企业做数字化转型，需要有 all in one 的产品解决方案来支持。未来随着国内云的进一步发展，尤其是云数仓的崛起，笔者相信会

有更多优秀的细分领域 Data SaaS 工具涌现。

这给我们带来的挑战和机遇是如何加速企业上云和数字化转型的过程，让企业有机会跳过工具能用的阶段，直接到好用。

奇点已来，推进 All on Serverless 有哪些困难、如何破局？| 解读 Serverless 的 2022

作者：阿里云开发者社区

从 2012 年提出 Serverless 到今年 2022 年刚好十年。

过去十年，上云是确定性趋势，在这个阶段企业一开始的关注点在于如何实现平滑上云。随着越来越多的企业上云，甚至很多企业系统第一天就是在云上构建，企业的核心关注点转变为如何更好地利用云的能力。

在新的阶段中，云服务的角色也从单纯的提供资源，变成了企业构建应用的新平台，帮助企业尽可能减小机器运维等低价值重复工作，聚焦于业务的创新。云厂商的核心能力转变为企业提供 Serverless 云服务，消除用云复杂度，用更简单的方式提高资源利用率，帮助企业降低成本。

这几年，不只是云厂商，大量的开源商业产品也采用了 Serverless 模式，包括 Confluent Cloud、MongoDB Atlas、Snowflake、Databricks 等。随着云厂商在存储、计算、中间件、大数据等领域推出越来越多的 Serverless 服务，云产品体系逐渐 Serverless 化，也推动着 Serverless 架构从概念进入到大规模生产落地阶段。2021 年 DataDog 发布 Serverless 研究报告，数据表明，从云原生初创公司到大型企业都在关注 Serverless，Serverless 生态已经超越了 FaaS，包含数十种服务，可以帮助开发人员构建更快、更动态的应用程序。

Serverless 奇点已来，所谓奇点，是由平稳发展转向高速发展的转折点，这也意味着 Serverless 在全行业的落地将全面爆发。

但与此同时，我们也看到，一方面是企业关注度不断上升、行业开始落地，另一方面，企业和开发者也对 Serverless 存在一些困惑和顾虑。

阿里云联合 InfoQ 收集了一些企业和开发者对现阶段 Serverless 的反馈，并于 12 月 16 日举办了云原生 Serverless 开发者评测局线下沙龙，邀请了阿里巴巴研究员、阿里云智能云原生应用平台负责人丁宇、阿里云 Serverless 研发团队、上海慧游文化 CEO 宋杰、德基网络科技有限公司大前端技术负责人刘晓伟以及南瓜电影、宝马等企业技术负责人，一起探讨 All on Serverless 过程中的挑战和经验，以期给更多落地 Serverless 的企业提供参考和借鉴。本文截取沙龙上的精彩发言。

> 本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一。

Serverless 到底是什么

在谈论 Serverless 之前，我们先了解下，Serverless 是什么？

2012 年，Serverless 概念正式登上舞台。2016 年，首届 Serverless ConF 将这项技术的讨论推向高峰。2017 年，阿里云推出国内首个重磅 Serverless 产品服务—函数计算 FC（Function Compute），让开发者只需要专注于业务逻辑开发，而不是把大量精力花在服务器等基础设施的管理上，从而快速兑现业务价值。

2018 年至 19 年，伯克利连续发表两篇论文对 Serverless 做出定义，预测 Serverless 将会成为云时代的最重要的计算范式，Serverless 获得了越来越多的关注。

Serverless 是一个非常广义的概念，并不局限于计算。一般同时满足以下条件可称为 Serverless 服务。

第一、全托管服务。意味着企业使用抽象的服务化接口，而不是直接面对底层资源，也就没有安装、配置、维护或者更新软硬件的负担。全托管服务通常也提供了内置的容错、安全、可观测能力，用户通常不需要再重新构建这些能力。

第二、自适应弹性。意味着服务能够根据负载大小自动弹性伸缩，不但让系统能够支撑快速发展的业务，也大大提升了资源使用效率。

第三、按实际用量付费。意味着只需根据实际的执行时间、流量或调用次数付费，降

低了成本。因此 Serverless 服务核心价值在于尽可能消除客户非功能性代码开发，简化应用基础设施管理的工作，从而实现研发效率的飞跃。

上海慧游文化 CEO 宋杰亲历首届 Serverless Conf 现场，在他看来：Serverless 是一项非常颠覆的技术和解决方案，它和之前出现的所有技术创新都有本质区别，因为它牵涉到的核心问题是商业模式的改变。“**类似于机器语言到高级语言的飞跃，一下子就简化了做企业级应用的程序员要关心的问题，复杂度大大降低**”。

Serverless 引领下一代应用架构

Serverless 服务契合了研发模式演进的趋势，过去十年，各大云厂商都在不断将产品体系 Serverless 化。

作为最早推进 Serverless 在中国落地的阿里云，Serverless 演进的过程，也是云计算的发展历程。阿里巴巴研究员、阿里云智能云原生应用平台负责人丁宇在分享中提到，阿里云是国内最早做云计算的企业，从 2009 年成立开始，第一阶段从资源和 IaaS 开始做起，在这个阶段，很多技术被开源并开始大规模流行，包括大数据、AI、中间件等微服务和应用层的技术。

从云的角度来看，这时候云厂商和开源具有非常好的协同关系，于是在云上出现了很多全托管的云产品。从企业客户角度来看，要确保应用在云上可靠、高效的运行，上云后的复杂度是非常高的，这时企业进入到想要更好用云的阶段，对于云的进化来讲，也开始向服务化的方式演进，也就是说，云开始从提供资源逐渐转变成提供服务。在这样的背景下，阿里云逐渐走向 Serverless 服务形态，包括诞生诸如计算类、存储

类、数据类、数据库类、大数据类甚至音视频类等 Serverless 的产品。

阿里云 Serverless 演进和实践

2017 年，阿里云推出 FaaS 产品函数计算 FC，这是一种以事件驱动为核心的全托管计算服务，用户只需编写代码并上传，函数计算就会自动准备好计算资源，以弹性、可靠的方式运行代码，并提供完整的可观测能力，大幅简化开发运维过程。第二年阿里云又推出了 Serverless 应用引擎 SAE，SAE 是业内首款面向应用的 Serverless PaaS 平台，屏蔽底层 IaaS 和 Kubernetes 的复杂度，提供了零代码改造、成本更优、效率更高的应用托管方案，帮用户实现单体 Web 应用、微服务应用以及定时任务的 Serverless 化。

同年领先业界推出 Serverless 容器服务 ASK，基于弹性容器实例 ECI（Elastic Container Instance），实现 1min 扩容 2000 个 pod，降低了 Kubernetes 使用门槛，让用户更专注应用程序，而非管理底层基础设施。

2020 年阿里云开源 Serverless Devs，成为业内首个支持主流 Serverless 服务/框架的云原生全生命周期管理的平台。2022 年 9 月该项目正式进入 CNCF Sandbox，也成为业内首个入选的 Serverless 工具项目。

Serverless 将云计算的粒度变得很细，企业选择是否采用 Serverless 产品不是单纯的软件问题，Serverless 背后需要有足够大的计算机集群和产品体系能力，才能保证其弹性满足企业复杂的业务需求。阿里云提供了从容器到应用等各个层次的 Serverless 产品组合，并且坚定的推进产品体系的全面 Serverless 化。

目前阿里云已有 20 余款核心产品实现了 Serverless 化，阿里云函数计算 FC 日调用次数超过 200 亿次，有效支撑历年双 11 百万 QPS 洪峰，业务增速超过 300%，整体规模位居国内首位，经历超复杂场景的锤炼。Serverless 计算产品凝聚了阿里云在云原生领域的技术沉淀和最佳实践经验。

除了产品形态的改变之外，Serverless 同样带来了软件研发范式的改变。Serverless 云产品是模块化、可组合、高可用的，他们是构建应用的要素。通过 Serverless 计算组合多个云产品就可以快速构建应用。

Serverless 为基础，研发方式升级到组装式研发。组装式研发彻底颠覆了原有的软件研发方式，大幅提升研发效率，灵活应对业务挑战。根据权威机构调研统计，组装式研发相比传统模式，可为研发提效 50%以上。

德基网络科技有限公司大前端技术负责人刘晓伟对 Serverless 的降本提效效果表示肯定。他说：“过去一年我们在做前端的 Serverless 升级改造。从支出成本角度来讲降低很多；从服务角度来看的话，需要分为几个方面，单一应用切成 Serverless 改造成本有适当增加，但一旦改造完成，后续开发效率就会高很多。“

这就是 Serverless 带来的价值：真正让开发者回归业务本身，让企业做得更少而收获更多。

Serverless 推进中的困境和破局

对于目前的 Serverless 来说，价值已经不言自明，当然，在规模化落地的道路上，

Serverless 也会遇到一些阻力。企业开发者在是否采用这一问题上仍有犹豫和顾虑，认同技术趋势与实际应用落地之间存在一道无形的鸿沟。

丁宇认为：这就是一个技术发展过程中常见的现象，是符合技术发展规律的。Serverless 规模化落地需要一个契机。而这个契机已经到来。当前企业及开发者拥抱云计算更加积极，云已经成了一个创新平台。同时，更多产品具备了云特性，如自适应弹性、免运维等。从 Serverless 本身来讲，因为它足够先进，不是简单地替代互联网分布式架构，而是对互联网分布式架构的升级，从而形成了自己的一套体系，能够为企业带来很多显而易见的变化。

由于处在不同的时期，企业会有自己不同的选择，随着 Serverless 技术发展得更成熟，相信会有更多的企业愿意拥抱 Serverless，也许像互联网分布式架构一样成为主流架构还需要 5-10 年的时间，但是先行者往往是会拿到第一波红利的，这会让企业更具有竞争力。

目前，Java 语言仍占据国内后端开发主导地位，而 Java 语言的冷启动特性在函数计算上的并不友好。冷启动时长使得用户很难享受到 FC 毫秒计费，秒级扩容等特性带来的技术红利。

不久前 AWS 发布了消除冷启动的产品 SnapStart，使得冷启动再次成为热议话题。

冷启动包含了系统冷启动和应用冷启动两个部分。对于系统冷启动，阿里云在硬件、MicroVM、操作系统、语言运行时等进行了全栈优化。阿里云函数计算 FC 在系统冷启动阶段的性能指标业界领先。而应用冷启动和用户选择的语言以及程序逻辑本身有关系。比如 Java 类应用，启动时间通常远远高于 Node.js、Python、Go 等语言，阿

里云内部一直在打磨这类技术，但实际应用会比较谨慎。因为 SnapStart 这类技术需要客户的应用程序对 SnapStart 进行适配，否则会出现程序正确性问题。今天 AWS 支持了这个技术，做了大量基础库、三方库的适配，虽然简化了客户的复杂度，但最终仍然需要客户自己保证正确性。因此阿里云提供了一些保证兼容性的方案，希望通过持续优化尽可能减少用户的适配负担。

目前，阿里云主要通过预留和闲置处理冷启动问题，另外，今年云栖大会函数计算 FC 发布了 vCPU 和内存解绑的策略，客户可以更精细化管理资源配比，资源利用率再一次得到优化，成本会进一步降低。

除此之外，厂商锁定（vendor lockin）也是阻碍 Serverless 规模化的一大困扰。对此，阿里云也有解法。

首先，阿里云已开源了 Serverless Devs，Serverless Devs 是一个开源开放的 Serverless 开发者平台，致力于为开发者提供强大的工具链体系。得益于功能的可插拔特性，Serverless Devs 可以非常简单的支持不同云厂商的项目部署，或者一键部署到不同云平台。

其次，Serverless 架构的理念本身是重塑应用的研发模式，云平台提供完整的 Serverless 产品体系，云平台提供一整套 Serverless 化的 BaaS 以及提供更为标准化、流程化的桥接。而 FaaS 更细粒度的业务处理拆分可以让业务代码开发更加简单，进而让企业多云方式的构建更加简单。

最后，企业采用多云方式，也不代表所有的业务都要进行多云，平均地分布在各朵云上，很多时候是在差异化的能力上选择最好的云，只有选择最合适的云、深度用云才

能真正发挥云厂商的优势，获得更好的云资源。综上所述，如何做无锁定，厂商开放规范，不做限制，海纳百川，在一定程度上就是无厂商锁定。

Serverless 未来的畅想和规划

除了技术需要通关，思维的转变也是横亘在开发者面前的一道鸿沟。

Serverless 从技术创新走到应用再走到普及，对开发者而言最大挑战是系统设计模式和编码习惯的改变。

不少开发者对 Serverless 架构的学习都有疑问，到底是用原有开发范式还是需要学习新的范式？对于这个担忧，慧游文化 CEO 宋杰认为：“Serverless 会对传统开发模式和逻辑实现颠覆，对成熟的开发人员而言挑战可能更大，学习成本方面，对于经验丰富的程序员来说，进入云原生世界，最大的障碍不是‘学习’，而是‘遗忘’。“

云原生技术打破了原有的限制，CPU、硬盘、内存没有大小限制。对于越是经验丰富的程序员来说，思维的转换就越是困难。“程序员要需要不断学习 Serverless，才能够克服固有的思维模式。希望阿里云能够提供更多的培训帮助大家实现思维定式的转换。”宋杰表示，这也是他对阿里云的期待。

除了提供更加全面丰富的 Serverless 产品与方案，作为 Serverless 中国的先行者，阿里云将持续拥抱开源，为 Serverless 用户提供多样化、可选择、更灵活的方式，提升开发者的幸福感。

“未来阿里云整个产品体系都将 Serverless 化，产品和产品之间的集成更加细腻，由点及面，将整个链路串联起来，普惠更多的企业和开发者。”丁宇表示。

微服务虽已老生常谈，但生命力超出不少人想象 | 解读微服务的2022

作者：李艳林（彦林）

> 本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦架构领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年架构领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

作者：李艳林（彦林），Nacos 开源创始人，阿里云微服务引擎 MSE 创始人。

笔者参与过阿里巴巴集团电商业务微服务的升级和改造过程，并一直在支持历年双 11 的大促活动，活跃在支持一线，也参与了 Nacos 开源和社区发展，目前正在阿里云负责微服务的整体开源规划和微服务引擎 MSE 的研发管理工作。因此在微服务技术的应用、开源和商业化服务上，有着比较立体的感受和思考。微服务虽已老生常谈，但其生命力之强超出不少人的想象。本文将从微服务发展的行业趋势和技术趋势两方面进行解读，欢迎各位参与评论，一起交流。

技术趋势推进行业变革，行业趋势反哺技术价值，两者相互促进，才能持续保持技术的生命力。我们先来看看这一年国内的一些重要事件，这些事件也恰恰揭示并加速着行业和技术趋势：

- 4 月，Linux 下一代架构基金会下成立了微服务技术组 SIG（Special Interest Group），共同探讨微服务治理标准化的解决方案
- 4 月，跨不同开发语言和技术框架，微服务治理规范 OpenSergo 开源
- 5 月，Envoy 社区推出 Envoy Gateway 产品，加入下一代网关角逐
- 6 月，腾讯正式开源 Spring Cloud Tencent，打造一站式微服务解决方案
- 7 月，Ingress Nginx 项目暂停接收新功能，将专注于稳定性提升
- 8 月，CloudWeGo 正式发布 Rust RPC 开源框架 Volo
- 8 月，直流科技发布服务网格项目（GA 版本）
- 11 月，Apache Dubbo 首个 Rust 语言版本正式发布
- 11 月，阿里巴巴开源下一代云原生网关 Higress，基于 Envoy，支持 Nginx Ingress 零成本快速迁移
- 12 月，字节跳动开源 kube-apiserver 高可用方案 KubeGateway
- 12 月，中国开源云联盟发布第二批开源成熟度评估结果，10 个项目中微服务领域

的开源项目有 4 个

- 此外，腾讯云&华为云先后提供了面向 Nacos 的商业版本

微服务发展的行业趋势

微服务仍在持续高速增长，关注度从开发效率转向交付效率

开源项目 Star 在一定程度上反应了一个项目的热度和生命力，从 Star 的增长上看，目前主流微服务开源项目每年以平均 10%以上的速度在增长，数字上呈现的是不仅是行业规模，也折射了每年新晋微服务开发者的水位。

- Apache Dubbo：4.3w，YoY 12%
- Istio: 3.3w, YoY 11%
- Nacos：2.6w，YoY 23%
- Spring Cloud Alibaba：2.5w，YoY 15%
- Seata：2.4w，YoY 10%
- Envoy: 2.2w, YoY 16%
- SkyWalking：2.2w，YoY 14%
- Sentinel：2.0w，YoY 13%

数据统计自 2022 年 12 月 25 日，计算主子仓库的累加数据。

微服务的持续高速发展，使得他已经和计算、存储、网络、数据库、安全一样成为云

计算的基础设施。只不过在每个不同的发展阶段，微服务面临的挑战并不相同。云原生普及之前，微服务开发者专注的是微服务的架构、迭代、交付和运维。随着云原生技术的成熟，微服务也在被云原生化，这时候，开发者和架构师更关心的是如何借助云的优势，简化微服务的运维问题，并更专注在业务的交付效率上。

微服务价值愈加凸显，侧重点转向降本增效

若干年前，阿里巴巴在做微服务转型的时候，看重的是微服务的可扩展性，因为要去快速应对庞大且日益增长的用户体量。但如今，随着人力成本的不断上升，以及云的算力成本的不断下降，微服务在帮助企业提高研发效率、落地敏捷开发的过程中，其发挥的降本增效价值已经越来越明显，这时候，微服务的价值侧重点是发生了变化的。过去的一年，我们看到大量的中小型企业开始采用微服务，快速上量、高效迭代，并将微服务应用建立在云上，加速迈入云原生时代。

此外，云拓宽了微服务的价值边界，其价值不仅体现在开发态的效率提升，也体现在了运维态上的价值。以 Nacos 为例，注册配置中心是一个旁路应用，但却承载着高频服务任务，一旦遇到可用性问题，会大面积影响线上业务，通过云厂商提供的 Nacos 商业版，可以极大的提高微服务在运维态上的价值，提升性能、提高可用性，阿里云的微服务引擎 MSE，以及腾讯云和华为云都提供了面向 Nacos/ZooKeeper/Eureka 的商业化服务。

从互联网走向各行各业，技术成熟度获得业内公认

这一趋势并不是只在 2022 年才出现，提出中台概念的那时候，很多传统企业就已经

通过微服务的拆分来构建中台架构了。近一年，随着各行各业数字化进程的加速，微服务不仅出现在直面线上用户的客户服务、市场营销等领域，也在帮助重构设计研发、生产制造、企业内销售财务协作、公司治理等领域，以便企业在市场竞争中保持先发优势。例如禾连健康帮助全国 2000 家医院构建在线体检服务，致景科技构建微服务体系打通全国纺织行业信息流、物流和资金流，实现数智化的综合平台，龙湖千丁借助微服务支撑全国 1000+的智慧停车业务，波司登全面推动各项业务容器化、微服务化，建立立体的在线销售体系，提升羽绒服销量。

不仅其他行业在快速落地微服务架构，对技术引入相当严苛的金融行业也正积极拥抱微服务。

随着金融科技的快速发展，市场对金融服务的需求日趋多样化和个性化，对场景金融服务创新、新业务需求敏态响应等提出诸多挑战，这些正不断加速金融企业的微服务架构转型。但是在传统金融领域，微服务架构的落地并没有像互联网企业那么迅速，这主要是金融行业对系统可用性和稳定性的要求非常高。

然而在过去的几年，随着微服务产品的成熟和技术的标准化，这一现状得到了极大地改变，我们看到越来越多的金融企业使用微服务产品改造其核心系统。工商银行通过 Dubbo 将基于 JEE 的单体架构改造成微服务架构，中信、光大、广发银行基于 Seata 的分布式事务能力，来解决交易数据间要求极高的数据一致性难题，中原银行的微服务平台以开源项目 SkyWalking（APM 应用性能管理）为基础，建立了分布式链路追踪系统，为微服务提供分布式追踪、度量聚合和可视化等一体化解决方案。微服务在金融这类严苛的行业下的实践，也充分证明其技术的成熟度和标准化。

微服务发展的技术趋势

标准化，呈现更多市场机遇

事实标准和行业标准并行发展，标准化有利于降低技术的选型和使用门槛，也是技术成熟度的一种体现，微服务领域开源领域呈现出比较明显的头部效应，大部分开发者倾向选择主流方案，以下数据来自我们自行举办的开发者沙龙或线上直播的调研数据，仅供参考。

- 微服务或 RPC 框架：以 Spring 为主，Apache Dubbo 有 20%+的市场份额，在强调国产自主可控的行业，Dubbo 占比会更多些，grpc 以及一些多语言的框架也一些忠实的拥趸。
- 注册和配置：Nacos 占比最高，Eureka/Consul/Apollo 各有一些市场份额，目前这个领域未看到有的开源项目。
- 限流降级：Sentinel 占比更高，虽然也有 Hystrix 等一些海外的开源解决方案，但由于本地化的文档和支持比较受限，开发者会更倾向于选择 Sentinel。
- 分布式事务：分布式事务主要应用于对数据一致性有较强需求的行业，且具有较高的技术门槛，开源选项并不多，目前是 Seata 更受开发者的欢迎。
- 云原生网关：主要分为 Nginx/Envoy 两大技术流派，APISIX 起步比较早，在国内的开源工作推进更快些，国外的开源项目有很多，例如 Kong/Emissary/Gloo 等，我们今年也开源了基于 Ingress 标准的云原生网关 Higress。
- 微服务治理规范和实现：相比以上的开源项目，这是一个相对较新的领域，开源选项主要是 OpenSergo 和 Polaris。

微服务开源领域的上下游关系

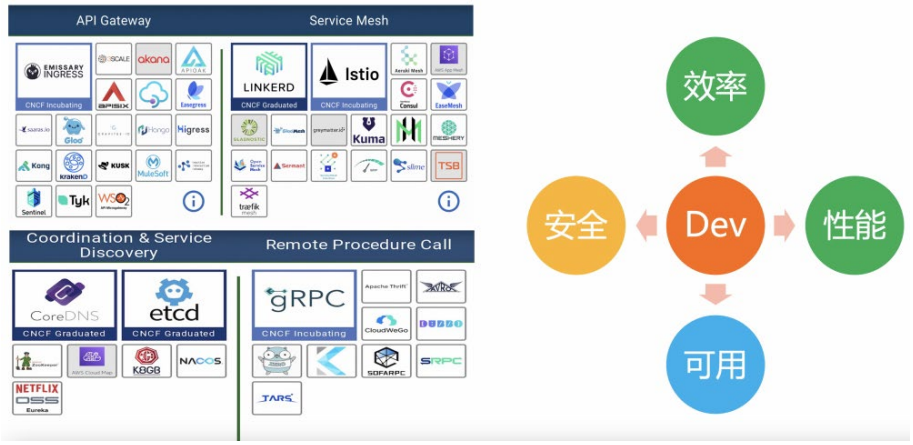
微服务开源虽然趋于标准化，但从不绑定，各个开源项目之间都是解耦和开放的。这时候开发者选型会侧重在高可用上。例如开源项目本身是否经历过大量的生产实践，并且是自主可控，避免一些不可控的风险；二是开源方是否会提供一些云上的商业版。因为当云成为基础设施的时候，开发者会关注开源项目在云上的稳定性、性能、易用性，这些往往是开源版所不具备的。例如面对流量有损的情况，或是版本发布时，没有做全链路灰度，一个线上故障引发全量业务受影响，没有控制爆炸半径，其实，这些场景都是可以通过商业版来避免的。越是主流的开源项目，越是有可能联合云厂商来提供公共云上的商业版。

因此，当微服务的各个领域的开源选型趋于标准化，开发者们将不再过多纠结在选型上，而是在交付效率和稳定性上时，将给围绕微服务提供培训、产品服务的企业呈现出更多的市场机遇，这也会反哺整个行业的良性发展。

DevSecOps，入口安全关注度提升

在过去的一年里，RPC 和注册&配置中心帮助开发者解决研发效率和性能问题，领域趋于稳定；服务网格和服务治理通过流量控制提升高可用能力，领域快速演进；网关帮助企业在数字化入口建立安全和高可用防线，领域处于兴起阶段，开发者关注各种方案的最佳实践和客户案例。

从这些微服务中间件的发展过程中我们发现，以围绕开发者为核心，提升微服务开发效率，保障生产运维高可用的 DevOps 实践越来越多，并且正在向 DevSecOps 快速演进。以云原生网关为例，在流量转发等基础运维（Ops）能力之上，企业和开发者愈加重视作为微服务网关的定制开发（Dev）能力，以及作为统一流量入口的安全（Sec）防线能力。



云原生网关，玩家众多、活跃度高

云原生网关领域，新晋的竞争者数量是去年同期的一倍，正在加速云原生网关的标准化。

技术趋势离不开内外因的合力作用。微服务的技术趋势之所以要单列一项云原生网关，外因是 K8s 逐步成为云计算的新界面，而 Ingress 作为 K8s 的网络通信的标准规范，定义了外部流量进入集群内部的规则描述。而内因则是网关领域长期面临着入口网管零散、诉求各不相同的情况，例如 Nginx 发挥流量访问的网关作用，微服务网关发挥微服务体系下服务和服务之间网络访问的作用，API 网关则实现了不同应用或系统之间的互访。当 Ingress 统一网络互访协议后，就催生了云原生网关的出现。

从技术优劣势上看，传统网关架构 API 定义不一致、多层网关运维代价大、扩展性弱、规则变更连接抖动，越来越不能满足视频/IOT/海外弱网的需求，不能适应云原生时代快速迭代、弹性带来的稳定性挑战。而云原生网关将多层网关进行统一架构和建设，不仅降低了网关架构的复杂度，也提升了缩短了网关的转发链路、提升网关的性能。国内最早开源的 APISIX 已经服务了 1w+ star，并且被 vivo/雪球/众安保险/有赞云应用于生产业务。

我们今年也顺势将在集团已经服务生产业务两年之久的云原生网关 Higress 对外进行开源，他具备标准化、高集成、易扩展、热更新的优势，目的是加速云原生网关在国内的落地。

数据面和控制面解耦，控制面重要性提升、但短期内较难标准化

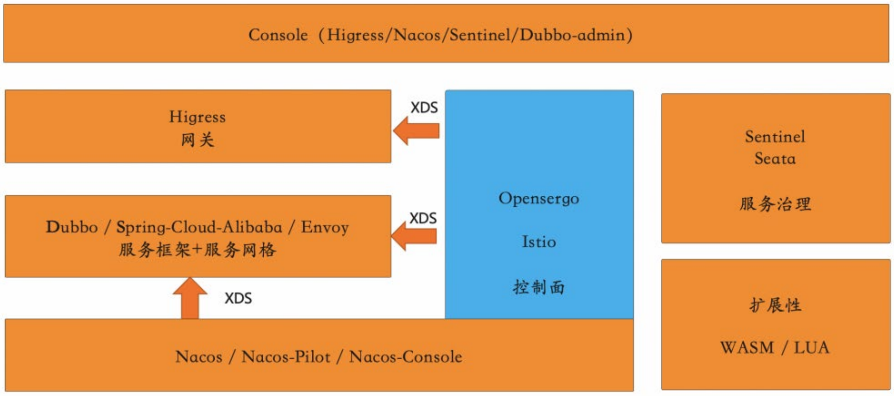
随着 xDS 协议成事实标准，数据面和控制面解耦，控制面作为数据面和治理平面的连接越来越重要。微服务生态可以有很多灵活集成模式，简单的场景注册&配置中心可以直接通过 xDS 协议将服务推到数据面，复杂的场景控制面可以聚合服务和规则推送到控制面，做更灵活、更复杂的治理能力扩展。

控制面的统一将带来诸多好处。

微服务已经从架构演进到治理，从解决如何用好微服务演进到如何管好微服务。管好微服务非常依赖微服务治理，而微服务治理是通过无损上线下、全链路灰度、流量防护等技术手段来减少、甚至避免发布和管理大规模应用过程中遇到的稳定性问题，但微服务治理在落地过程中会遇到各种难题。例如：

- 在企业内部，往往存在着不同语言、不同通信协议的微服务，这会导致治理微服务的过程中，给业务开发者、架构师平添很多的认知负担，而这类异构会衍生出更多的痛点。
- 业内对服务治理的能力和边界没有明确的认识，每个企业所定义的服务治理概念不一致，造成很高的理解和沟通成本。
- 开源微服务框架众多，对于服务治理缺乏一些标准化的约定。例如，Spring Cloud 中定义的微服务接口和 Dubbo 中定义的接口就没有办法互通，通过 Dubbo 和 Istio 管理的微服务也没有办法进行统一治理。
- 缺少真正面向业务、能够减轻认知负担的抽象和标准。开发者真正想要的可能是简单的、指定服务间的调用关系和配置规则。但现在对于业务开发者来说，不仅需要了解不同微服务框架的部署架构，也要了解不同服务治理方式的概念和能力区别，认知成本很大。

此时，控制面的统一能很好的解决以上困扰。因此我们今年和 bilibili、SphereEx、中国移动等企业，以及 Kratos、CloudWeGo、ShardingSphere、Nacos、Spring Cloud Alibaba、Dubbo 等社区联合发起并开源 OpenSergo 项目，目的就是构建一个和语言无关、和技术形态无关，但贴近业务的统一服务治理规范和实现。另外，Linux 下一代架构基金会也正在致力于服务治理标准的制定。但控制面的统一并非一朝一夕，需要参与方在实践中不断对标准进行磨合、逐步达成共识。



其他的一些技术趋势

多语言：Java 作为国内微服务开发最主流的语言，正在加速云原生化演进，通过 Spring 6.0 / Spring Boot 3.0 / GraalVM 让 Java 能在容器和 Serverless 时代更轻量、更快速运行。随着传统客户数字化的升级，之前 C++/C/Python/PHP 研发正在加速向 Go / Rust 演进，目前 Go 微服务生态逐步完善，前端 Node.js 跨语言调用后端服务场景也越来越多，Rust 微服务生态开始起步，Dubbo、CloudWeGo、Nacos 都在今年发布了 Rust 的版本，相信后续会有更多开源项目跟进。

国产化：核心技术自主可控越加重要，整个软件行业都在加速国产化进程，微服务开源项目需要投入资源，支持 ARM/IPv6/Dragonwell/PolorDB/OceanBase 等国产化的上下游生态，才能获得更多开发中的青睐。

Proxyless：xDS 协议成为事实标准，从 Lib 包依赖模式到 Sidecar 模式到 Proxyless 模式，Sidecar 模式无侵入带来一些运维复杂度，Lib 包模式有侵入但易运维、性能好，社区尝试 Node 级 Proxy 方式，介于两者之间，但目前处于前期状态；从当前的开发者体量看，Lib 包依赖是主流，Mesh 是补充。

总结

标准化、国产化、DevSecOps、云原生网关、Proxyless、统一控制面和多语言，给微服务带来的新的市场机遇。近一两年，国内出现了很多围绕这些领域的初创公司，也给很多成熟企业带来的新的商业方向，两股力量的相互交融延续着微服务的生命力。

2023，可观测性需求将迎来“爆发之年”？ | 解读可观测技术的 2022

作者：鲁冬雪

> 本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦可观测技术领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年可观测技术领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

随着 5G、云计算、微服务等技术的广泛应用，企业所面临的 IT 运维环境越来越复杂，需要运维的系统不仅数量多，而且网络架构复杂、基础设施多样。在信息化建设日益

普及的当下，快速提升企业 IT 资产管理能力开始成为企业迫切解决的问题。

过去一年，可观测技术得到了极大关注，并被 Gartner 列为 2023 年十大战略技术趋势之一。然而，企业在实际落地可观测的过程中依旧面临很多问题，比如如何选择合适的方案、平衡新旧系统之间的关系等等。

其实，一个企业 IT 系统的可观测性建的好不好，在新员工入职时就能看出来。如果新员工去看文档时，看的是公司某个架构师画的系统架构图，显然这个公司技术系统是缺乏可观测性的。因为企业系统一直在迭代，而系统的架构图却很难及时更新。如果新员工看到的是完整的可观测性平台数据看板，那这个公司的“可观测性”构建的应该算是比较完善的。当企业的可观测性构建趋于成熟后，开发者可以快速定位系统问题，精准的“观测性数据”可以帮助开发者完成包括自动化巡检在内的各种各样的 Control 或者是金丝雀发布的控制等等。

所以，可观测技术目前发展到哪个阶段了？开发者们对可观测性平台的需求到底是怎样的？带着这些问题我们对观测云 CEO 蒋烁淼进行了专访，我们一起对可观测技术过去一年的发展做了一个盘点。

一、国内外“可观测性”的认知尚存在较大差异

当我们想要盘点一项技术过去一年的发展状况时，首先要看它过去一年的应用率是否有所提升。去年，领域里有相关数据表明，大部分公司都在进行可观测实践，但国外也仅有 27%的公司实现了全栈的可观测，国内可能更少，可能 10%的比例都不到。

而今年，这个比例似乎并没有提高。

正如蒋总所说的那样，“在中国，真正地构建了完全可观测性的企业非常的少，做的好的企业少之又少。”

放眼过去，目前国内确实很缺乏好的可观测性的服务商和技术提供商，这让中国的可观测性技术发展与国外产生了不小的距离。包括字节、阿里云等大厂在内的国内大部分厂商在可观测性方面都还是处于“零散拼装”的状态，他们大多把开源的 Prometheus、Loki、OpenTelemetry Collector 组装在一起形成“可观测性套件”，然后提供一个开源的托管版本给开发者使用。很少有人敢称自己为“可观测性平台”，因为没有一个完整的可观测性的概念。

然而，过去一年里，无论是在国外还是国内，企业 CTO 们对于可观测性的关注度有了明显大幅提升。从全球视角来看，不管是 Grafana 的 ObservabilityCON，还是今年谷歌云举办的 Google Cloud Next'22 Recap、北美的 KubeCon 2022 大会，亦或者是前不久亚马逊云科技举办的 re:Invent 全球大会、NGINX 举办的 NGINX Sprint China2022，可观测性话题的探讨比例都有了明显提升，像 Google Cloud Next'22 Recap 大会谈可观测性几乎占到了大会整体内容的三分之一。

切回国内视角，当我们和国内的企业 CTO 们去聊天的时候会发现，他们今年明显会更关注可观测性这件事，无论是因为业务层面譬如 B 站 713 事件带来的启发，还是受行业大会演讲内容的影响，国内在可观测性方面没有明显的技术发展，大多是因为疫情下的市场经济不景气、企业预算不足下的“按兵不动”之举。

聚焦到具体的业务需求上，目前企业技术决策者正处于一个认知迭代的过程中——

“将监控升级为可观测性”。原来的“监控”大多都是分裂的，可能是每一个项目都有一套监控工具，可能是是一个项目中的日志平台、APM 平台单独运行。所以现在不少开发者会认为可观测性就是新瓶装旧酒，他们认为把传统的监控工具整合在一起，就是可观测性了，甚至不少创业公司的 CTO 会将一些 K8S 监控工具或者 APM 混同于可观测性。

事实上，可观测性相关的工具如果被称为“下一代监控”的话，它的投资占 IT 的比重并不小，可观测性的应用场景远远要超过目前大部分开发者的认知。“可观测性”这件事在大部分 CTO 的意识中，无论是横向扩展还是垂直扩展，大家的认知都还需要有一个“求同”的过程。

就像现在许多人会把“可观测性”定义为是一个降本增效的好工具，很多相关产品厂商会将“降本增效”作为产品的推广利益点，这让许多开发者认为安装了可观测性平台，就可以降低成本、为业务创造价值。事实上，可观测性工具从来不是一个“便宜”的东西，可观测性平台除了基本的工具投入，还需要有一套完整的数据存储方案，企业自研一套优秀的可观测性解决方案投入成本并不低。以美国企业的可观测性相关投入为例，其会占企业整体 IT 支出的 5%-10%。而且，它的“降本增效”不是短时间内就可以显现出来的，而是通过长时间的应用，为业务提供大量的数据支撑进行技术性优化，从而产生价值。

总体来说，真正的可观测性平台应该是，能够将各种各样对于系统的形态、实时的状态进行有结构性的收集并提供一系列的观察、测量手段的平台。简单来说，就是用各种各样的技术，像传感器一样，能够让开发者们的开发、测试、运维过程变的更容易，能够即时了解系统的运行状态，而并不是简简单单的“监控”。

可观测性平台一定是数据组织统一化的，否则分开的日志系统、链路系统、指标系统很难实现数据与数据的关联。比如日志里的字段，主机系统中叫 host，在指标系统里叫 host name，然后有的系统用 IP 地址，有的系统用主机名，那在做关联分析和连接查询时会发现根本无从下手。有的公司也会将不同系统的数据分散开进行监控，利用“数据中台”的一个构建方案将所有数据汇聚到同一个平台中去做分析，但在实践过程中会发现，数据缺少实时性的结构化，无法做实时预警预测，只能做事后故障定位。

二、全球可观测性技术演进主要聚焦在 6 个方面

1、OpenTelemetry 逐渐成为行业标准

OpenTelemetry 作为一套由 CNCF 主导的云原生可观测性的标准协议，目前已经是海外企业在该领域的实践标准。从数据收集的角度来说，基于 OpenTelemetry 或者兼容 OpenTelemetry 的方案，从今年到明年都应该会逐步成为主流。OpenTelemetry 已迭代到了 1.0 版本，作为一个客户端，日志和网络等部分也都正在得到逐步补强，严格意义上讲，OpenTelemetry 1.0 已经算是完成了整体的标准化，OpenTelemetry 标准协议已称得上是一个行业标准。

反观国内，虽未形成类似的标准，却也涌现了诸如 CAT 和 SkyWalking 等一系列国产开源的 APM 系统，通过实时监控企业的应用系统，系统化地提供应用性能管理和故障定位的解决方案，在运维中被广泛使用。得益于对业务代码无侵入，性能表现优秀，

社区活跃，中文文档齐全等众多优秀特性，SkyWalking 在国内异常火爆。

2、Grafana 8.0 修改了 AGPL 协议

开源项目背后的商业公司只有在商业上取得成功，才能更好地为开源项目提供新特性的开发、支持与维护，形成良性循环，这样开源项目的维护才会更加长久。Grafana 在云原生技术领域的监控工具中几乎是标准一般的存在，而在可观测性的投资方面，Grafana 也写下了浓墨重彩的一笔，推出了 Loki 日志存储、时序引擎 Mimir 和调用堆栈的存储引擎等一系列产品，力图通过提供一个复刻 DataDog 所有功能的开源方案，打包售卖 Grafana Cloud 服务。

因此在 Grafana 8.0 后修改了 AGPL，允许甲方有技术实力的公司基于它的开源方案做二次开发，而不允许乙方在没有商业授权的时候拿着 Grafana 的产品包装出售，此举对国内不少的信创公司都造成了巨大的冲击。

3、eBPF 技术日趋成熟

内核通过文件系统或系统调用暴露出的信息有限，单纯从用户态切入无法抓取这些信息，只有从内核本身调用的过程中获取调用信息才可以补足相关能力。eBPF 是一项革命性的技术，它可以在 Linux 内核中运行沙盒程序，而无需更改内核源代码或加载内核模块。通过使 Linux 内核可编程，基础架构软件可以利用现有的层，使它们更加智能和功能丰富，无需继续为系统增加额外的复杂性层。

eBPF 促进了网络，安全性，应用程序配置/跟踪和性能故障排除等领域的新一代工具

的开发，这些工具不再依赖现有的内核功能，而是在不影响执行效率或安全性的情况下主动重新运行。在对内核无侵入的前提下，它通过动态地向内核中插入一段自己的代码，实现定义监控及跟踪能力。

随着内核的发展，eBPF 逐步从最初的数据包过滤拓展到了网络、内核、安全、跟踪等领域，并且它的功能特性还在快速发展中，通过内核虚拟机，所有的 IO 包括程序运行、进程的网络通信等各种细部数据尽收眼底，大大加强了系统的可观测能力。

然而，目前大多数的只通过 eBPF 获取数据并把自己标榜为可观测平台的公司，也仅仅是做了一个 eBPF 的支持而已。从第三方视角看，eBPF 是对传统观测能力一个非常好的补充，可以通过它拿到更多的数据，用户获取内核丰富观测指标的门槛被极大降低。

4、可观测性开始更“注意安全”

云原生数据库等新兴概念的出现，大大消除了大家对于存储成本方面的顾虑，使得企业可以更聚焦于提升观测能力本身，而不必在成本和性能方面分神，这方面的发展也有迹可循。比如今年 DataDog 推出了一个名为哈士奇的存储架构，它是一个完全云原生面向可观测性的数据库，得益于云原生的存储能力，可以在同等价格下多存储 5-10 倍数据，以往即使收集到了海量的数据，但面对高昂的存储成本，这些数据也只能被丢弃或者针对这些数据进行采样，本质上看，这对客户需要的结果也是有浪费的。但这也并不意味着使用这种新技术之后并不需要进行采样，只是新的更低成本更高性能的数据库技术，可以大大提升可观测性能够覆盖的范围，降低存储成本，这也是一个很重要的发展。

安全和可观测性的合并，已在全球范围内形成一种趋势。摩根士丹利在《安全分析和可观测性》文章中也提到，在国外，以 DataDog 为代表的公司在上市之后发布的新增功能中有 70%都是安全相关的，这其中的道理非常简单，可观测性是通过检查其输出来衡量系统内部状态的能力，它收集了系统的方方面面，通过这些数据可以分析出系统的故障，自然也就能够分析出系统有没有被入侵。比如 DataDog 就提供了通过分析当前访问请求，区分哪些可能是黑客在嗅探，或者准备未来做 DDoS 攻击的接口的功能。

也就是说，采集的数据在安全方面也能够发挥作用，而不像传统安全工具那样，需要针对安全场景再进行一次数据采集。所以，安全和可观测性的合并在全球范围内已经成为一种趋势，当然抗 DDoS、挖防火墙这些并不会合并，针对攻击现场的追踪，比如国内的态势感知、国外的 SIEM 这些安全产品都选择了和可观测性进行融合。

5、建立“业务的可观测性”越来越重要

可观测性的概念起源于工业领域，在该领域中，可观测性被定义为从系统外部输出推断系统内部健康状态的能力；在软件产品和服务领域，可观测性就是从应用系统中收集尽可能多的遥测数据，以便可以调查和解决新出现的复杂问题，确保企业能够主动观察系统，在影响客户体验之前解决故障及问题，安全地进行测试并实施优化，它可以更好地管理和控制业务风险，有助于我们了解“正在发生什么”以及“为什么会这样”。可观测性使团队能够更有效地监控现代系统，帮助他们找到并连接复杂链中的影响，并将其追溯到原因。此外，它还使系统管理员、IT 运营分析师和开发人员能够了解他们的整个架构。

如今的 IT 系统，迭代发布更迅速，业务系统更庞大，网络链路更复杂，运行环境更动态。在“业务至上”的互联网时代，技术工程师们保障的核心其实并不是这套 IT 系统或软件，他们保障的核心其实是业务，一笔业务可能会涉及到多个微服务系统，技术工程师们不再追踪一个 API 的全链路调用关系，而是要追踪到整个 API 关联的订单、用户甚至具体到哪一笔交易，这也是可观测性和业务结合的一个重要发展趋势。Gartner 也提到，“未来一切业务皆需可观测性”，简单地讲就是把运营人员、运维人员、IT 人员看到的数据做统一，而不是互相甩锅。

6、多个“新玩家”入局

随着微服务架构的流行，一些微服务架构下的问题也日渐突出，比如一个请求会涉及到多个服务，而服务本身可能也会依赖其他业务，整个请求路径就构成了一个网状的调用链，而在这个调用链中，一旦某个节点发生异常，整个调用链的稳定性就会受到影响。

据此，我们需要一些可以帮助理解系统行为、用于分析性能问题的工具，以便发生故障的时候，能够快速定位和解决问题。近年来，各种调用链监控产品层出不穷，呈现百花齐放的态势，各家你方唱罢我登场，比如 New Relic 收购的那家叫 PIXIE 的公司可以称得上是一个新玩家，又如最近出现了一个面向 K8S 的叫做 Kubeshark 的公司，它专门研发用于网络分析的开源组件。

此外，随着 eBPF 的出现，未来会涌现越来越多的解决方案，尤其是面向整个 K8s 生态环境，这也是全球可观测性技术的一个演进方向。

三、具有“4 个统一性”的可观测性平台是企业刚需

当我们盘点完可观测性的行业趋势，所以到底什么样的平台才是目前企业所需要的呢？国内首批获得中国信通院颁发的「可观测性平台技术能力」先进级认证，新一代云原生全链路数据可观测平台——观测云给出了答案。

观测云能全环境高基数采集数据，支持多维度信息智能检索分析，及提供强大的自定义可编程能力，使系统运行状态尽在掌控，故障根因无所遁形，实现了统一采集、统一标签、统一存储和统一界面，带来全功能的一体化可观测体验。观测云平台从技术层面主要实现了 4 个统一——统一采集、统一治理、统一分析和统一编程：

第一个统一，是统一的数据收集。数据采集是数据关联的基础能力，现今，国内外都有大量的数据采集器，但大多数采集能力单一，比如 Telegraf 仅支持指标，Filebeat 只服务日志，OpenTelemetry 的 Collector 对非云原生的组件并不友好，需要大量安装 Exporter 插件。

为了实现系统的可观测性，一台主机上可能需要安装无数个 Agent，这既引入了管理上的问题，又产生了成本的问题，一定程度上造成了资源的浪费。观测云的 DataKit 是目前唯一的真正一体化实现各种环境（如传统环境、云原生等）统一数据采集平台，它通过一个进程或 Daemonset Pod 就可以实现全方位的数据采集，配置体验良好，开源且拓展性强，涵盖了更为全面的数据采集类型，有海量的技术栈的指标收集能力，采集器的配置更简单，数据质量更好。

第二个统一，是数据采样格式实现统一。可观测性数据的一个重要特性就是实时性，

Logstash 等传统的开源方案均在中心或边缘中心完成数据治理工作，而受限于中心处理的延迟和 delay，在日志收集方面通常有很大的时延，实时性也就无从得到保障。而在观测云的解决方案中，客户只需多拿出 5%的 CPU 用于数据的处理，一方面可以发挥可观测数据的实时性。另一方面，还可以利用数据本身的现状还原。比如当把所有数据收集到中心的时候，实际上很多环境信息或多或少的都会存在遗漏或缺失的情况，如数据来源于哪个容器、哪个 pod、哪个部署单元或者哪个项目等，在边缘侧可以随时获取这些数据作为补充。

行内人都知道，观测云是一个 SaaS 方案，在边缘侧还可以实现采样功能。比如数据中存在的一些电话号码、金额、用户 ID 等信息，完全可以通过边缘脱敏或者删除掉一些不需要上传或者不易上传的数字。这样一来，数据在上传前就已经完成了脱敏、加密或者丢弃，在丢弃情况下，还能大幅降低传输的带宽；数据无需上传到中心再做治理，这也进一步提升了客户的信任。此外，由于减少了一个中间处理模块，系统的整体实时性也得到了很大的保障。

目前观测云也在考虑把端上的统一治理能力做成一个插件，贡献给 OpenTelemetry，使得 OpenTelemetry 的 Agent 也具备边缘计算的能力。据悉，目前观测云平台完全兼容 OpenTelemetry，单一采集器即可实现容器指标采集、应用链路追踪和日志采集。同时能和各种常用采集器实现串联，包括 Promethues、Skywalking、Fluentd 等十几种常见工具，可以替换更可以共存，轻松实现数据高密采集。

第三个统一，是统一的数据治理。用户的数据在被传输到中心后，将会被统一存储、统一压缩、统一部署，保存在观测云自己定义的逻辑数据存储架构中，相较于开源方案，观测云主要有两个优势：

- 在存储的整合性角度，观测云的成本低于开源方案；
- 不论是指标数据、日志数据、链路数据、用户行为甚至未来开展到安全数据，区别于其他厂商平台，观测云通过自研的 DQL (Debug Query Language)语言在数据分析平台进行整体分析，甚至针对不用用户使用不同时代的存储技术。

第四个统一，是统一的拓展开发平台。当 DQL 或者软件本身的能力无法满足用户的需要时，用户可以通过 DataFlux Function 平台进行无限的拓展。比如出于中大型公司的安全合规要求，直接通过第三方平台发送 SaaS 平台的告警邮件到内部邮箱，会被直接屏蔽掉，而如果用户把自己的发件服务器注册到 DataFlux Function，就可以实现向内部邮箱发送告警邮件，类似的能力还有短信、电话等等。

除此之外，DataFlux Function 平台还可以实现对各种云平台甚至业务数据的载入，最近，观测云使用 DataFlux Function 帮客户实现了业务对账系统和 IT 代码调用间的关联。据悉，未来观测云会基于 DataFlux Function 平台推出一个抢占型实例量化交易的 Demo。

除了这四个“统一”，观测云还提供 Site Reliability Engineering (SRE) 和 Observability Engineering 的最佳实践，支持智能推测算法，可自动识别基础设施和应用程序的潜在问题。坚持每两周迭代升级，使工程师们可随时体验最前沿的技术和最尖端的能力。是目前市面上少见的，真正意义上做到了“观测性”实现的平台，真正地满足了企业系统及开发者们在观测性方面的需求。

百闻不如一试。当我们回溯平台选型的根本，发现当前市面上的可观测性平台质量良莠不齐，大多还处于“监控”状态，系统整体性较差，无论是对于企业来说还是对于开发者来说，试错成本高。而目前观测云 SaaS 版完全按量计费，没有初始成本，拒

绝为闲置功能付费，可提供专属服务经理，让初创团队也能体验一流的全链路系统可观测能力，切实地解决了平台选型难的问题。

正如蒋烁淼所说的，“一个好的系统首先要做到采集、治理、分析、变成这四个方面的统一；其次，需要尽量兼容开源，将整个技术栈和开源进行一个双向的连接。观测云在未来将进一步下探存储成本，最终让利用户。”

四、国产可观测性的未来需要更多厂商共同努力

目前，可观测性还在发展初期，很多产品仍在探索阶段，也有很多问题亟待解决，未来人们对可观测性的需求只会越来越高。

谈及“可观测性”的未来，蒋烁淼认为，可观测性未来主要有四大趋势。观测性平台的开发厂商未来需要在以下几个方面做出改进和优化：

- 提供更大的技术环境支持能力，支持标准的可观测协议；
- 引入更多的算法，以智能的方式做数据的巡检；
- 连接安全产品；
- 进一步降低存储成本。

这与媒体第三方视角观察到的趋势保持相似。然而，纸上得来终觉浅，绝知此事要躬行。虽然国内像观测云等厂商紧跟行业需求及技术发展趋势，提供了自定义数据大盘、基础设施监控、日志分析、用户访问监测、应用性能检测、Profiling、云拨测、智能

巡检等多种解决方案，力图打造可观测性最佳实践。但是，仅仅是一家或者几家的努力，是远远不够的。

行业的发展需要更多的人参与进来。观测性平台开发厂商需要挖掘更多用户需求，而普通开发者也应该将更多的注意力放到“可观测性”上，让可观测性工具成为业务优化的得力助手。

数据烟囱亟需打破，云原生融合数据库雪中送炭 | 解读云原生数据库的 2022

作者：郑思宇

采访嘉宾：矩阵起源创始人及 CEO 王龙

> 据 Gartner 预测，到 2025 年，基于云原生平台的数字化业务比例将达到 95%，将带来云原生数据库市场的快速增长。毫无疑问，云原生数据库即将成为云上数据库使用的标准范式。

> 云原生数据库使得传统数据库得以充分结合云服务的免运维、高弹性、高可扩展、高可用、高性价比优势，又顺应了云端应用大爆发的历史趋势，在过去几年成为 IT 领域的大热门方向，但总体仍处于高速发展的早期阶段。过去一年中，企业对于云原生数据库的使用有哪些需求？云原生数据库厂商又该如何满足这些需求？本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一。我们将带你回顾 2022 年云原生数据库领域的重要动态以及技术进展。在本文写作过程中，特此感谢矩阵起源创始人及 CEO 王龙提供的宝贵观点与洞察。

云原生数据库领域的挑战和进展

知易行难，数据库从云服务到云原生还有很长的过程

在过去相当长的一段时间里，云服务中被使用最多的是虚拟机，也就是物理服务器在云端的虚拟化服务。IT 用户享受到的好处是最朴素的自服务、免运维、高可用、高弹性、高可扩展性和按使用付费，当然这些都只是在虚拟机和 OS 层面（Infrastructrue as a service）。

为了更好地满足数据库用户的需求，云厂商开始提供数据库服务（Database as a service，DBaas）。为了降低开发成本和用户的学习难度及应用迁移门槛，最早的 DBaaS 大多是基于 IaaS 的数据库运维自动化，本质是传统数据库系统平移到云端的一种全托管服务，用户看到的和使用的数据库，除了在访问 OS 层面有所限制，其他并无二致。

在云计算（IaaS）成为趋势并逐渐成为 IT 行业的标准基础设施之后，上述这种做法导致的问题便浮出水面：

问题一：数据本身的爆炸式增长，数据库无论在物理环境还是云端虚拟环境，都需要具备更高的扩展性和更强悍的性能。传统数据库的架构由于受限于历史原因，本身具有的缺陷和瓶颈在云端则更加明显。例如，进行性能优化和执行调度时，传统数据库在资源有上限的物理环境中更在意资源的使用率，并且需要投入大量精力来考虑 RTO 和 RPO，那么整个系统就会因此增加复杂度。然而，这种复杂度在云端是没必要的，并且会造成巨大的资源浪费和开发投入。

问题二：随着数字化程度的进一步加深，业务敏捷度也成为 IT 系统最关注的核心指标之一，数据库的弹性能力是支撑应用和业务敏捷度最重要的基础能力。基于物理环境设计的传统数据库，即使平移到云端，也无法充分利用云端计算和存储资源的高弹性能力。例如物理环境和云端环境的硬件特性，包括但不限于吞吐量、读写特性、资源部署时间、资源回收时间以及消耗成本，都是不同的。如果没有考虑到这些不同点，就会严重影响数据库实际的弹性能力。

问题三：由于大多数用户在数据库上的预算增长超过其他组件的增长，使其更加看重数据库服务的性价比。然而，多云部署日益流行的今天，每个云厂商的定价策略，计算、存储和网络资源的计价方式和能力表现都有或多或少的差异。如果仅仅将一个传统数据库平移上云，很难基于这些复杂的因素做出最优的判断决策，也无法达到最优性价比。

因此，用户和市场都需要一款能充分利用云端的计算、存储和网络资源特性，具备真正的高弹性、高可用和高性价比优势的数据库，这就是云原生数据库发展的最核

心驱动力。要实现这样的目的，一个基于云环境精心设计、打磨的技术架构是不可或缺的。这就需要长期持续的投入，才能对云环境足够的了解并对数据库生态充分适应。

需求多样，云原生数据库需要跟上现代软件应用的步伐

曾经有一种说法是“软件吞噬世界”，暂且不管这种说法是否正确，但物理世界愈发数字化是一个正在发生的事实。过去数十年，数字化的加速催生了数以亿级的软件，包括手机 App、SaaS 等等，这些软件对数据库提出了各种各样的需求，也驱动全球出现了多达数百种不同的数据库。

十几年前，我们就知道数字化进程的一大阻碍就是数据烟囱（Data Silo），其严重制约了企业的智能化和创新能力，并以几何级数提高了企业利用数据的门槛和成本，一直以来都是企业信息化部门最头痛的顽疾之一。然而，随着云计算、软件应用和数据库的蓬勃发展，非但没有使这个顽疾得到缓解，反而变得愈发严重。为了解决这个问题，在过去数年中，IT 从业者做了多种尝试。

第一种，简单粗暴地投入最强硬件，搭载一款数据库支持所有软件应用。目前，这种方式已经越来越少见了，因为纯硬件的纵向扩展能力是有限的，而软件应用的组合和变化是无穷的，大部分客户无法负担成本变成了最主要的阻碍。

第二种，整合多种集中式和分布式数据库系统，使用统一界面给数据和应用开发者提供各种能力和服务，并隐藏其底层管理运维的复杂度，这种方式常常被称作“数据中台”和“数据底座”。

基于过往多年的实践，虽然证明这种形式在某些领域和客户群体中是有其价值的，但在某些场景下便会出现局限性：一方面，当后台需要整合的数据库系统过多，又或者前台的应用变化过快、过于复杂时，它的运维、管理和开发成本增长会陷入失控，且相比第一种方案会带来数据实时处理性能的瓶颈；另一方面，当用户需求相对较小和简单时，使用这种方案又会过于臃肿和复杂，投入产出不成正比。

第三种，一些成熟的数据库厂商退而求其次，如果不能一蹴而就地整合这么多数据库系统，那么可以尝试在现有成熟数据库系统中添加新的能力，来减轻一小部分数据烟囱带来的痛苦。

湖仓一体、批流融合、HTAP 等都是这种思路下的产物，但融合的挑战也是巨大的。拿 HTAP 举例，由于 TP 和 AP 数据库在过去默认就是服务两个不同的应用团队，由不同的数据库管理团队来维护，因此对安全性、资源共享和性能隔离等都有各自的需求。当融合在一起的时候，想要性能完全隔离，就应该使用完全独立的计算和存储资源；想要资源利用最大化，就应该使用共享计算和存储资源；同时想要获得更低的处理时延，就应该只存一份数据；想要各自都有读写极致性能，就应该存多份数据……这里有太多矛盾的技术点，要想找到平衡点来解决上述提到的技术矛盾是一件不容易的事情。

当然第三种尝试还有很多其他类型的方案，这里就不一一列举了。在 2022 年，我们也注意到有众多云原生数据库厂商在朝着这个方向演进：

- Snowflake 在其年度用户大会 Snowflake Summit 2022 上，宣布推出 Unistore 存储引擎，使得用户在 Snowflake 平台上运行 OLAP 的同时也可以确保数据的完整性和一致性，而这是 OLTP 的核心特性之一。

- 在 2022 re:Invent 大会中，亚马逊云科技发布了一个新服务——“Zero ETL”，其在后台打通了 Aurora 数据库和 Redshift 数据仓库。用户无需自己开发 ETL，就可以轻松地进行数据分析和机器学习，这更像是数据中台和 HTAP 的结合体。
- 最近刚刚完成 F 轮融资的 SingleStore，也号称其数据库系统能在云上通过结合事务和分析工作负载，消除了性能瓶颈和数据移动，以支持数据密集要求苛刻的工作负载。
- 国内初创公司矩阵起源提出的“HSTAP”更为彻底，将 HTAP 进行了重新定义，融入了串联 AP 和 TP 的 Streaming 能力，并完全重新开发了一款云原生的融合性数据库。目标是让企业只用一款数据库，就能覆盖大中小应用系统的的 TP 和 AP 需求，并能用最高性价比的方式建设好数据中台。

放眼未来，云原生数据库还需要关注基础设施和软件应用的新动向

在基础设施层面，其能力也在不断演进。《2021 InfoQ 年度技术盘点与展望》的“架构篇”里提到。在 2021 年，各个云服务商都在不断拓展 Serverless 产品线的能力。在 Web 应用、微服务、事件处理和批处理任务等场景，Serverless 正在成为最流行的架构。在 2022 年，这个趋势依旧在延续，并且还在不断深化，代表企业有亚马逊云科技、阿里云等，这也意味着 Serverless 已经从概念和愿景逐步演进至落地实践。

Serverless 能更好的提供计算资源的弹性和可扩展性，从而提高应用系统的性价比。对于计算成本占比较多的数据库系统来说，是一个不可忽略的重要技术趋势。同时，能否利用好 Serverless 服务也是架构是否云原生的重要指标之一。相比之下，国外企业的上云速度与规模快于国内，其已经累计了大量的云上数据以及应用场景，可以帮

助数据库厂商打磨云原生数据库能力。以 Snowflake 为代表的云原生数据库，目前已经到达了收获的阶段，在 Snowflake 公布的 2023 财年第三季度的财务数据显示，Snowflake 第三财季的总营收为 5.57 亿美元，同比增长 67%。而在国内，Serverless 在数据库中还没有得到大规模应用。

离商用化尚有距离的 CXL 也是云厂商关注的热点技术趋势，此外，GPU、FPGA 等异构芯片在云端的使用也在加速增长，这些对数据库的技术演进也会有较大影响。

在软件应用方面，智能制造、机器人、自动驾驶和元宇宙等新应用还在不断成长之中，数据库厂商也需要关注这些领域的需求增长和变化。这些应用场景相关的图数据库、区块链数据库、时序数据库等也在持续迭代。

云原生数据库值得关注的其他方向

虽然云原生数据库所带来的价值不言而喻，但坦白讲，其也很难形成一统天下的局面。

对于金融、政府这类对于合规安全有较高要求或者定制化需求较多的客户来说，私有化数据库部署的需求还是普遍存在的，但这也并不意味着云原生数据库的增长空间会被挤压。随着数字经济时代的到来，金融、政府等机构还有很多个性化的业务依旧需要部署在云上。

那么在这种情况下，混合云架构是很多企业的首选，据 Cisco 发布的报告显示，在全球混合云趋势调查中，82%的受访者目前使用混合多云架构来支撑其应用程序，并表示混合多云架构能够使组织实现更敏捷和高扩展性的开发环境，同时加速业务敏捷性

和创新。

云原生数据库厂商若想帮助这些企业管理好数据，让他们更好地进行降本增效、实现创新，就要求数据库架构既能能兼容不同的公有云，也能够进行私有化部署，并且还要保障云上和云下的用户体验一致，数据可以自由的、低成本流动。这就对云原生数据库提出了更高的要求，既能充分利用云上的资源及其特性，也能高效地利用私有数据中心的资源及其特性。**从这个角度来看，数据库云原生的下一步或许会是异构云原生，即数据库无论部署在公有云、私有云、物理服务器上都要有统一的技术架构并且共享调度数据和资源**。

除此以外，DB 和 AI 的结合也是一些团队正在研究的方向，主要分为 AI for DB 以及 DB for AI 两类。

第一种模式是 AI for DB，即利用 AI 算法帮助 Database 提升产品能力。比如通过分析数据库日志或者视图，得出数据库的性能表现，然后去动态地调整数据库参数，来达到优化数据库性能的目的。此外，还可以在数据库中利用一些 AI 算法来进行内核优化、运维优化等。

另一种模式是 DB for AI，即在非结构化数据日益增多的情况下，为分布式数据库提供 AI 分析的能力。目前来看，由于传统的应用开发工程师、数据分析师与 AI 工程师在企业是分割开的，后者还处于探索阶段。但相信随着数据库人才与 AI 人才的交流，共性的需求会变得越来越多，比如如何对数据做预处理、做初步的建模等等，待这些知识融合到一定程度时，必然会出现一个系统来满足他们的需求。

云原生数据库的选型建议

虽然国内数据库的发展相比国外起步得要晚一些，但凭借着后发优势、业务环境等因素，国内数据库市场也逐渐呈现出百花齐放的态势，粗略判断，目前云原生数据库数量已达到几十甚至上百种。

因此，企业在进行选型时，首先要考虑清楚自身选择云原生数据库的驱动力是什么，既要让云原生数据库的特性与自己的业务类型相结合，又要与自己的团队能力进行匹配，是更看重低运维自治，还是更看重资源的弹性扩展，或者更看重性价比，需求不同往往会做出不同的选择。为了避免踩坑，也需要考虑厂商的技术实力和周边生态支持的能力。

其次，企业在选型时还要做好中长期的规划，绝对不要低估数据烟囱对未来业务的影响。随着企业的发展和数据规模的不断攀升，数据库带来的成本增长会远超于业务应用的成本。如果提前做好规划，在数据库的选型与后续建设中就可以做到游刃有余，将来的数字化道路就会事半功倍。**比如，由矩阵起源打造的新一代超融合异构云原生数据库 MatrixOne，便可以帮助用户降低数据的使用难度，提供极简的使用体验，让企业可以将精力从繁杂的技术细节中释放出来，最终达到降本增效的目标。**

写在最后

回顾这一年，云原生数据库领域依然在稳健高速的发展中。云原生数据库依然还需要时间来达到成熟，但无论是使用创新架构来更好地使用云上资源，还是融合多种能力更好地服务数据应用，各云原生数据库厂商都在沿着帮助客户降本增效的初心，持续

进化中。

采访嘉宾

王龙，矩阵起源（Matrix Origin）创始人及 CEO。曾任职腾讯云副总裁，管理数百人的大数据和人工智能团队，从零构建腾讯云 to B 大数据人工智能产品矩阵和商业化体系，三年内完成产品线从百万到十亿级收入的飞跃。王龙之前在美国硅谷和德国慕尼黑工作过，也在北京上海有过创业经历。他具有多年跨国企业级产品建设和商业化经验，对于云计算、大数据、人工智能和多个行业应用场景有着丰富经验和深刻认识。

从大前端“穿越”到终端，开发者应该必备什么技能？| 解读终端的 2022

作者：闫园园　审校：Tina

采访嘉宾：郭虹宇（弘禹）、舒文亮（舒文）

> 本文是“2022 InfoQ 年度技术盘点与展望”系列文章之一，由 InfoQ 编辑部制作呈现，重点聚焦终端领域在 2022 年的重要进展、动态，希望能帮助你准确把握 2022 年终端领域的核心发展脉络，在行业内始终保持足够的技术敏锐度。

特此感谢郭虹宇（弘禹）、舒文亮（舒文）（按姓名首字母排序）对本文的贡献，他们的真知灼见，是本文能与大家见面的关键。

诚然，刚刚过去的 2022，对整个互联网行业来说，是极具挑战且富有变化的一年，然而聚焦到大前端技术领域，整体上仍保持平稳向前的状态，如果一定要用关键词形容，综合受访专家看法，本文认为“深度优化”一词较为贴切。

所谓“深度优化”，是指前端领域经过蓬勃发展，已逐渐进入成熟期，短时间内并没有出现令人眼前一亮的技术拐点，但同时据多位身处技术洪流中的专家预判，新一代的技术也正在悄然孕育与布局。当然，这也意味着又一道难题摆在了当代开发者面前：大前端领域划时代分水岭将发生在何处？

或许，我们能从近几年发生的某些重要事件中看出一些端倪：

- 在 WWDC 2019 上，苹果更新了 ARKit 增强现实平台，并宣布推出全新的 ARKit 3；
- 在 WWDC 2021 上，RealityKit 2 新增加了 Object Capture API，通过该 API，能够帮助开发者将在 iPhone 上拍摄的视频，在几分钟内创建真实世界对象的高质量、照片般逼真的 3D 模型，并将它们转换为针对 AR 优化的 3D 模型；
- 在 2022 Google I/O 上，谷歌 CEO 发布了地图新功能，虚拟与现实相结合，仅通过图像就能生成实景画面，该功能简称为“沉浸式视图”；
- 同样在 2022 Google I/O 上，谷歌发布了一款 AR 眼镜作为“one more thing”压轴登场；2022 年 7 月，华为正式宣布发布 HarmonyOS 3，HarmonyOS 是华为推出的面向万物互联的全场景分布式操作系统；
- 疫情三年，中国互联网 to B 市场大规模增长。2022 年 12 月，钉钉宣布注册用户

破 6 亿，第三方数据显示企业微信 MAU 过亿。

……

那么对于开发者来说，究竟从这些事件中能够得到什么信号，信号又指引了哪些方向？接下来，本文将跟随来自阿里终端委员会两位技术专家弘禹与舒文的脚步，探索终端的过去与未来，试图为身处迷雾中的开发者带来问题的部分答案。

实际上，相较大前端这一技术名词，阿里巴巴曾不止一次提到终端这个概念，这不禁引人好奇，当大前端的含义仍被业内津津乐道之时，终端的说法又从何而来？

从大前端到终端，未来已来

弘禹谈到，终端其实并不是一个新词，甚至对于资深开发者来说已经亲眼见证了几代终端的变化：从最早 80 年代使用的显示器加键盘过渡到 PC 普及，再过渡到如今移动端设备普及，可以看到，大概每十年终端的边界都会迎来一次扩展。基于此技术趋势，阿里巴巴沉淀出对“终端”的定义，即：“为用户提供人机交互的设备及应用。当前以 UI 交互终端为主，如手机、PC、平板等。”

“我们在包括不限于 D2 大会等在内的各个场合都提‘终端’这个概念，是因为阿里正在以自己对行业的理解和认知在对未来做预判。”舒文谈道，同时他借用今年阿里终端委员会成立时对内公布的两段话进一步阐释了终端的内涵：

> 随着技术的演进，当下的前端和客户端有了越来越深的相互渗透：Web 的开放性和高效迭代，客户端技术的即时体验和原生能力，持续升级的技术体系像有生命意

> 识一般朝着“既要又要”的方向攀沿：运行在客户端内的 Web、Hybrid API、拥抱 W3C 标准的动态化跨端引擎（Weex 2.0）、基于开源 Flutter 生态的大前端研发体系，都在让前端和客户端不断地交织融合、模糊着边界；
>
> 更为重要的是，互联网基础设施越来越完备、终端设备规模&类型爆炸式增长，用户的体验将不再单纯的是网页或 App，不同的信息载体都将拥有自己的体验：头戴式 VR 设备的沉浸式体验、扫码枪的爽快支付、车机的语音和硬件呼叫应答、商场的各类娱乐屏幕、家庭的智能设备，甚至健身房的魔镜都不断在印证着科幻小说作家那句，“未来早已到来，只是尚未平均分布”。用户的体验将被空前的满足，而我们则更应该提前做好准备应对变化，面向用户有统一全局的终端视角。

螺旋形理论讲到，事物的发展或进步不是直线式上升，而是类似于“螺旋”般上升，这个理论放到前端技术领域，同样也适用。结合前文提到的重要事件释放的信号来看，大前端领域虽然未曾行至清晰拐点，但端技术的重要性确是日渐凸显。终端概念的提出，背后的本质正是看到了前端和客户端的互相渗透，同时期望二者能够尽快打破过往壁垒，及早为新的终端设备和体验做好技术和人才上的布局。

了解终端的概念后，下面将按照领域中的重点技术方向，感知当下领域的具体发展现状，以及展望未来，领域中面临的机遇与挑战。

重点技术趋势解读

Web 开发仍是主流

2021 年 5 月，InfoQ 发布的编程语言 4 月排行榜中显示：Web 开发依然是终端领域的主流。据投票数据显示，Web 开发得票率为 68%，iOS 开发得票率为 16%，而 Android 开发则为 5%。同时，小程序开发以 6%的得票率超过 Android 开发。那么在 2022 年，这一趋势是否发生了变化？对此，舒文谈道，就其个人观察，这一趋势不仅没有变，从某种角度来讲，反而会变得更为主流。以桌面领域以及移动互联网领域两个纬度分析：

桌面领域

过去两年，to B 市场不断被挖掘，桌面终端技术逐渐升温。基于浏览器的 Web 技术自不必多说，原来的桌面客户端领域也开始大量渗入 Web 技术。究其原因，不外乎三点：

- 桌面硬件性能高度过剩；
- 类 Eeletron/Tauri 的开发框架持续完善带来诸多成功案例；
- 操作系统原生支持 Web 技术进行研发（如 Windows 支持 WebView2、PWA）。

移动互联网领域

其实，无论什么技术形态或者研发模式，本质上都在回答一个务实的问题：如何让技

术方案做到便宜（成本低）、好用（易上手）、质量高（交互体验好）。在过去很多年，移动 App 领域中，Web (WebView) 天生具备了前两者优势，但却也因受限于太多因素，如独立的进程策略、高耗时的启动渲染逻辑、缓存能力不完善等在高质量交付上总是捉襟见肘。因此在过去，开发者们时常会听到一个说法："先用 Web 搞，然后慢慢 Native 化。"

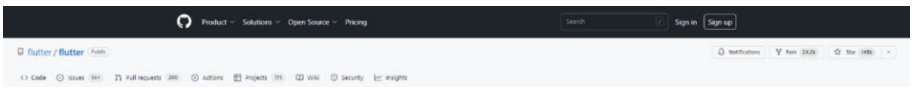
而当下，这种情况已发生了巨大的变化，其中，促使这种变化的原因主要有两个：

- 手机硬件性能的高度过剩，愈发成熟的跨平台基础设施（包括不限于脚本引擎、排版引擎）都使得跨平台研发成为行业的主流诉求；
- 具象到研发模式上，类 Web 的方式在效率和体验上几乎是规模化研发下的最平衡解。

主流跨端框架

根据终端的定义，目前市面上"端"的形态多种多样，为了解决多端独立开发的问题，跨平台技术便应运而生。目前，在各大互联网公司一起努力下，市面上出现了各种跨平台框架。那么跨平台框架 2022 年应用情况如何呢？首先看几个业界主流解决方案：Flutter、React Native、Weex。

Flutter



站在 2023 年初看来，Flutter 已经是不再是以前小众的跨平台框架了。如图所示，可以看到如今的 Flutter 已经有高达 148k 的 Star 和 24.2k 的 Fork。从官方公布的数据上，Flutter 已经基本超过其他跨平台框架，成为最受欢迎的移动端跨平台开发工具，截至 2022 年 2 月，有近 50 万个应用程序使用了 Flutter。

2022 年 5 月，Flutter 3.0 正式发布，不同于之前的版本，Flutter 3 新增了对 macOS 和 Linux 应用程序的稳定支持。“Flutter 3.0 出来之后，我个人感觉它这套体系会更成熟”，弘禹谈道，同时他认为 Flutter 未来可能会向布局多端多设备上更迈进一步。“这也给业界一个更强的信息，就是真正实现过往我们一直都在谈的一套代码多端运行”。不过，他也坦言这在技术上确实是有一些瓶颈和阻力的。

2022 年 8 月，谷歌又发布了 Flutter 的 2022 年的第 3 个版本，即版本 3.3，随之一起发布的还有 Dart 2.18。这两个版本都是改进版，没有添加重大的新特性。

React Native

对于 React Native 来说，它一如既往地在跨平台应用程序开发中占据一席之地。大多数科技巨头都使用它来开发应用程序。2022 年我们看到的最新和关键变化之一是其架构的更新。

React Native 0.69 发布了强大的新功能，例如 React 18 支持和捆绑的 Hermes。它带有默认启用的 React 18、useId 等新钩子以及 useTransition 或完全 Suspense 支持等新功能。

Weex

2013 年，阿里开始 All in 无线，整个互联网都在做 PC 到移动端的转型，那几年移动端流量暴涨，设备性能也较差，这时，肩负着历史使命的 Weex 出现了。

目前，手机淘宝仍在大规模使用 Weex，主要应用于需要快速迭代的产品、业务以及大量卡片场景。2020 年八月份启动 Weex 2.0 项目，不再依赖系统 UI，开发自绘渲染引擎，并且保持向前兼容。

受限于篇幅原因，本文无法一一展现全部众多跨平台框架，那么对于开发者来说，又该如何众多跨端框架作抉择呢?

弘禹认为对于初创公司或者规模较小的团队来说，在人员配置以及能力模型契合的前提下，Flutter 或许是一个比较好的选择。不过，无论选择哪个跨平台框架，最重要的还是依据项目场景，比如可以用 Flutter 做 App 底座或者 iOS、安卓的跨端技术选型，但如果做跨多端甚至跨应用，则建议选择偏 Web 的框架。“不管选 Flutter、RN 还是 Weex，我觉得最重要的还是结合场景，同时结合团队不同发展阶段和人员配置情况，再去做综合的判断取舍和选择。”弘禹总结。

JavaScript & TypeScript

2007 年，Jeff Atwood 提出著名的阿特伍德定律，“any application that can be written in JavaScript, will eventually be written in JavaScript”，意指“任何可以用 JavaScript 来写的应用，最终都将用 JavaScript 来写”。在上述报告中，这一定律再次用数据得到了印证：终端领域编程语言使用情况上，JavaScript 以 78.5%的得票率，排名第一；

而 TypeScript 得票率为 42.9%，排名第二。

对此，舒文判断，目前这一趋势依然不变，并且从长远来看，也会持续较长时间。毕竟终端领域的交付，早已并非过往简单地实现还原界面，而是变得越来越繁复，这意味着逻辑复杂且规模庞大的 JavaScript 应用程序成为普遍事实，当然，也需要越来越多的工程师参与维护迭代。

TypeScript 中的“Type”，正是通过程序中的规则来建立秩序的类型系统，从语言层面提供基础，同时缓解这两类问题。借用《人月神话》一书中的观点：

> 需要协作沟通的人员的数量影响着开发成本，因为成本的主要组成部分是相互的沟通和交流，以及更正沟通不当所引起的不良结果。

不过，需要开发者注意的是，以上说法并不意味着 TypeScript 有朝一日完全成为 JavaScript 的替代品。尽管 TypeScript 的主要贡献者 Hejlsberg 确实提到“竞争”这个词，但此“竞争”单纯从浏览器语言角度出发的。“事实上，它们二者更像是“亲缘”关系，当然，也是超集和子集的关系”。从功能层面，某个功能或特性进入到 ECMAScript 的标准化流程相对稳定阶段，即会在 TypeScript 中被纳入且进入社区化状态。

回顾 2022 年，TypeScript 版本迭代很快，2 月 TypeScript 4.6 发布，5 月 TypeScript 4.7 发布，8 月 TypeScript 4.8 发布，11 月 TypeScript 4.9 发布。几个版本迭代中，除了常规的语法改进和性能增强、工具优化，也有较为亮眼的三点升级：

- 性能分析层面 (v4.6)：针对高耗费性能的类型，TypeScript 有 generateTrace 进行输出，但是易读性很低，现在提供可视化方法进行阅读。在此基础上，还发布了一个 Trace 分析工具；

- Node 支持（v4.7）：对 Node.js 16 的 ECMAScript 模块（ESM）支持以及新的编程语法、新的关键字、新的 Compiler Options、类型推导增强等；
- 引入文件系统事件检测文件（v4.9）：在之前的版本 TypeScript 使用轮询检测文件，它意味着需要周期性的文件检查。虽然轮询不会引入跨平台问题，但在非常大的 TypeScript 工程项目中，意味着额外的性能开销。在 4.9 之后默认了文件系统事件，只做失败回退（至轮询）。

在发布两个 Beta 和两个 RC 版本之后，TypeScript 团队发布了将于 2023 年 5 月释放 v5.0 的计划，在该计划中，团队也明确了 TypeScript 的下一步路线：语言和编译器紧跟标准（社区&标准化）、优化生产工具效率（生态）、优化性能（基座）。“当然，大多优秀的开源项目都有着清晰的演进路线，但是，最近几年 TypeScript 团队在演进确定性、核心逻辑坚持上给予了开发者非常强的信心，这也值得我们每个工程师学习。”

WebAssembly

有人说 WebAssembly 的发明，相当于再一次发明了 Flash、Silverlight 甚至是 Java。作为一种新的运行时，它与 JavaScript 运行时协同工作，具有轻量级、泛语言支持、兼具高性能和移植性等优势，可以说一诞生就自带光环。2022 年 4 月，W3C 公布了 WebAssembly 2.0 的第一批公共工作草案。

目前来看，这项技术在国内外已有了一些非常典型的应用场景，国外包括 Google Earth、Figma、AutoCAD Web 版、Photoshop Web 版、微软的 Blazor WebAssembly 等，国内包括但不限于蚂蚁链智能合约、腾讯邮箱（部分功能）、开源

项目 flv.js 等。

单纯从浏览器厂商来讲，WebAssembly 的重要性已经受到了一致认可，甚至连原本支持计划落后的 Safari 都在 2021、2022 年加快了对它的支持。但也可以看到，WebAssembly 2.0 仍然只是一个草案，意味着仍然有待讨论的问题未达成一致，这里面既涉及到一些技术细节，也受到一定程度的厂商考量。“从这个角度来看，WebAssembly 志远道艰且阻。”

然而，如果换个角度去看，WebAssembly 的前景似乎也谈得上光明：

首先，从 Web 开发者来看，虽然以 JavaScript/CSS/HTML 为基座的应用开发模式占据了浏览器应用的唯一选项，但依然有两个难题自诞生之初就悬在整个开发生态之上：

- 受限于 JavaScript 语言缺陷和 ECMAscript 的完备支持程度，大型应用的可开发、可维护性；复杂应用下的用户端体验性能问题。
- 前者已经出现 TypeScript 及各类生态工具链的完善等解决办法来用以规避，而后者，截至目前，WebAssembly 仍是最有影响力的根因解法之一。

其次，从行业视角来看，大型软件、复杂应用面临最大的问题之一即是跨平台研发的成本问题，而浏览器及其依赖的标准具备天然的、法理性的跨平台特性。由此也带来一个非常严肃的思考：如果有一项技术方案能够将大型软件、复杂应用低成本移植到浏览器环境下，且保持足够的性能，那它除了获得成本收益外，还将具备更低成本的用户体验提升（免安装、中心化迭代），其背后的价值潜力将充满想像。舒文认为，这正是 Figma 具有叹为观止的商业估值的原因之一，再进一步讲，这也正是当前较完备的 WebAssembly 的产品范例都是工业化程度、复杂度较高的软件应用的原因。

站在当下，如果要真正解决前面提到的两个难题，仍需要更完备的技术支撑，并且要解决的问题也要远超过当前此技术的进展：除了 2.0 草案提到的 WebAssembly Core Specification、WebAssembly JavaScript Interface、WebAssembly Web API 亟待解决以外，还有未形成共识的 Relaxed SIMD（宽松 SIMD）、Exception handling（异常处理）、Threads（真正的多线程）等问题需要注意。

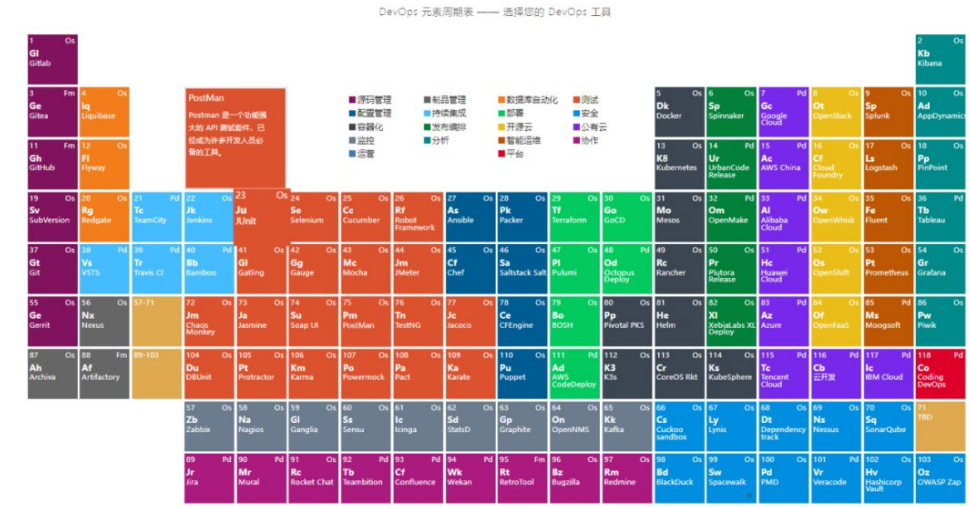
总之，短期来看，WebAssembly 的发展是低于行业观察者预期的，即使是像异常处理这种特性已经在 Phase 3 阶段，但如果搜索常见的 WebAssembly 库，流行的库仍局限在特定数学计算、利用大型成熟 C++代码库将其功能带到 Web（无需移植到 JavaScript），按照这个态势发展，它会转变成是典型的利基（Niche）技术，这其实是失控的。

而长期来看，WebAssembly 是一项非常值得关注的技术领域，其诞生不过十年间快速受到社区认可并纳入标准，并被厂商广泛支持，对接广泛的语言支持（Rust、Go、C、C++等），甚至衍生到公开技术联盟（Bytecode Alliance），还是有相当大的机会解决前面提到的复杂 Web 应用下的端体验问题。“当然，前提是在适合的场景下。”

DevOps

DevOps 是一整套成熟、完整的软件开发&测试运维的方法论和实施体系。在崇尚敏捷迭代、高效部署的现代软件研发思潮中，它在众多高级技术决策者心目中几乎是事实正义。

而它的实践在过去几年，也被执行得坚决且持续：这不仅反映在大量科技公司的运维部门和开发部门的高度合并，同时也催生出海量的衍生技术产品。



Ledge 公司 DevOps 元素周期表

聚焦到终端领域，相对中心化的 Web 产品而言，移动 App 的 DevOps 的贯彻对于技术效率的提升显得尤为关键。这里弘禹、舒文以手淘为例进行了说明：

首先，对于手淘 App 来说，从平台视角看因为淘宝业务体量、架构自身复杂度、代码库规模庞大等原因，单版本集成模块数超过百级，双端构建时长超过 15 分钟，出包成功率极端情况下也不足 80%。因此，阿里终端团队今年将重点解决两大工程技术命题，一方面如何通过构建技术突破持续优化超级 App 的构建时长和成功率，通过包括不限于构建技术升级、构建过程优化、构建软硬件优化等技术手段，最终成功地将“一杯咖啡，构建成功”变成现实。与此同时，为了改善日常工程师的研发环境痛点，通过 Web/CLI 多端工程产品升级使“做 XXX 一行命令就够了、分钟级配好迭代配置”的研发体验交付到工程师手中，极大地提升了日常研发效率，逐渐成为终端的一种工作方式。

同时，对于手淘业务来讲，从业务视角看需求全周期上线时长决定了业务创新速度，App 版效及需求交付效率是最大瓶颈，全量发布流程在叠加各类卡口情况下约需要150+环节，App 版本发布及运维成本居高不下，需求迭代生命周期也相当复杂：需求经历从设计、开发、变更、集成、一灰&BugFix、二灰&BugFix、多外部渠道上架、自有渠道全量整个过程直接制约了业务发展。为了解决这一挑战，阿里终端团队一方面通过工程平台的版效能力升级，实现了全环节 70%自动化率，使手淘实现低成本的周版发布，另一方面基于 Android AppBundle 研发了无 Hack、低适配成本、零性能损耗的原生动态化框架，云端结合实现手淘App的发布动态化，基本实现小时级近千万设备的更新覆盖效率，同时也有机会革新集成窗口机制，让业务创新更快发生，阿里巴巴在终端工程效能水平上明显提升。

另外，关于 DevOps 的讨论其实一直也没有停止。2021 年 7 月，Sid Palas 在推特上一语惊起千层浪："DevOps is dead , long live Platform Engineering!"，2022 年 8 月，InfoQ 也曾转译过 Scott Carey 的文章："要什么 DevOps，我们开发者根本不想做运维！"这两个讨论中，他们都有相类似的观点，即 DevOps 是技术管理者的美好愿望，但是对于编程主业的工作者来说，他们并不希望与 Infra 和 Ops 产生深入联系，因此，大家提出了平台工程（Platform Engineering）的概念。

对于这一观点，这里，舒文也向 InfoQ 提出了自己的看法：

1. 无论是直瀑式开发、敏捷交付、DevOps 以至 Platform Engineering，主要依据的是企业规模以及生产关系，甚至与技术一号位的技术偏好也有关。不同阶段、规模、技术风格的科技企业有权利选择适配于他们的方式并坚持下去，而不是忽略环境条件在不同的技术理念之间反复横跳；
2. Platform Engineering 是对 DevOps 理念的更显性化交付，它有具体、显性的交付

物，横跨在 DevOps 理念与开发者之间，帮助开发者自助、自治的方式进行软件交付。Platform Engineering 不仅没有和 DevOps 产生鲜明的对抗，还直接产生了开发者效率价值，而唯一的成本，则是决策者们投入资源构造相对应 IDP（Internal Developer Platform）的真正决心。

“手机淘宝的移动体验平台团队打造的各类开发者产品即是 Platform Engineering 的典型实践，只是目前处在快速上升期”，舒文介绍。同时，据了解目前对于 Platform Engineering 乃至 IDP 的建设，阿里终端也正在探索自己的发展路线与最佳实践，即基于在过去移动高速发展的 10 年在终端研发、交付、运维、体验、高可用、移动 PaaS 能力、SDK 软件供应链与合规等相关领域的沉淀的完备的中台能力基建与工具链，着手融合升级全新体验的终端一站式工作平台。

低代码

2020 年被称之为低代码元年，在这一年低代码平台在国内起步，到了 2021 年，低代码平台开始在国内迅速蔓延，到了 2022 年，各大资本看好低代码发展，纷纷投资潜力厂商：低代码平台 Swimlane 获得 7000 万美元融资；Baserow 筹集 500 万欧元用于创建无代码工具链；无代码软件测试平台 Sofy 获得 775 万美元种子轮融资；无代码初创公司 Sitemate 获得 520 万美元以加速无代码软件平台.....

虽然低代码热度空前，但关于其的争论却未曾消失，一直到今天仍有人坚持认为它是“新瓶装旧酒”，并以诞生于上世纪的“Dreamweaver 也是低代码工具”作为佐证。对此舒文也提出了看法，他认为当一个技术理念开始爆发式地变得平民化且创造价值，那意味着它大概率是正确的。“具体到低代码这个方向，我是非常认可的”，他进一步

解释低代码的本质实际上就是使非专业人员在具备组织逻辑、设定输入、控制输出的信息化能力，并且能将这种能力用于解决日常事务。例如在曾经，“打字员”甚至是一种职业，这对 Z 世代的青少年们来说是很难理解的。

聚焦到终端领域，无论是 to C 还是 to B/G 领域，终端工程师们制造出了不计其数的 "View"。在此过程中，为了代码层复用，工程师们设计了 UI 组件库、功能模块库；为了能够更高效生产，工程师们借助了各种理念做到 Image to Code；为了能够规模化生成，工程师们设计语义化的流程编排能力，进行低成本组装，而这正是工程师们一步步优化生产效率，持续追求精细化、规模化的过程。“它就是低代码/无代码，我认为这是非常了不起的”，舒文谈道，但同时他也认为，“我们也不能够迷信它，让它解决所有的问题，必须不断利用更先进的方式来提升零代码覆率”。

移动端性能优化

“在多元化体验时代的今天，移动互联网流量规模见顶后，各赛道对时长争夺的白热化阶段，消费者面对多平台选择下对体验容忍度、忠诚度也逐年降低。对手机淘宝来讲如何让前台用户操作体验流畅、全程购物体验满意，特别在平台用户时长不断提升及浏览深度在不断加深的同时，保障好基本盘稳定性、长时在线的网络可用性、全链路性能的确定性可能是下一阶段赢得用户信任的关键所在。而这些的背后都这意味着终端团队一直在跟业务和技术复杂度、设备性能挖掘与合理利用、复杂网络环境在做对抗。”弘禹谈道。

他提出性能优化相关技术其实已经是老生常谈，业界今天的关注点也已经从过去的常规手段，比如包大小治理、启动性能优化、外链唤端跳失率等，转移到今天的看可视

可交互时长影响访问体验、帧率与流畅度影响浏览体验、图片/视频上屏影响转化、内存可用性及崩溃率影响浏览深度等问题，从过去的平台技术视角转换到用户体验/业务价值重新审视今天体验技术命题。

然而，一直到现在，有一个问题一直悬而未决，那就是在面对前台业务快速变化、表达型态更加多样化，如何将性能体验常态化、不劣化，“如何将能力建设跑在业务前面，用较低的技术投入成本下，让业务天然跑在一个性能高水位上，这可能是终端技术人需要思考并去解决的问题”。弘禹进一步谈道，针对此问题，可能还并没有确切答案，需要业界持续探索，但就当下来看，可以从眼下三点入手：

- 终端高可用：如今的移动互联网业务型态已经发生巨大变化，从过去单一图文型态开始走向直播/短视频为主流、无游戏/无互动不增长、3D/AR 不断体验创新，在此背景下终端设备始终是资源受限系统，如何做到长时间使用及深度浏览不崩溃，做好全局资源使用合理性与基础可用性的治理能力，以手机淘宝为例从过去看平台 Crash 率到分场景的精细化可用性治理、关键场景如加购下单阻断，在“留量时代”都是在对大盘增长做贡献。过去几年在阿里终端团队不断推进架构治理与技术攻坚下，终端高可用技术体系也有了长足的演进，体现在 Crash/Abort 率、技术舆情率等方向逐渐在收敛至良好水平。
- APM/NPM：据不完全统计，中国目前有超过 500 万个不同类型的 App 并且数量每天都在持续增加，从各渠道不完全统计阿里系活跃 App 就超数百款，覆盖多平台/多设备/多系统/多渠道，且因业务型态与用户场景的不同，所选择的技术选型各异。在此背景下终端如何面向异构复杂技术架构、多平台 x 系统 x 设备差异性下持续做好体验管理与性能表现的一致性，基础性能管理 APM 以及网络性能管理 NPM 的建设，对每个互联网企业的终端团队都至关重要。可以看到的是，过去几

年阿里在不断完善性能可信指标体系、线上性能数据化能力、设备体验分级能力、弱网对抗与策略化能力等方面建设已取得一定阶段性的结果，使用更科学有效的方法使核心场景的流畅度指标达标率提升到了 90%以上，卡顿类 NPS 调研也升 10 个 pt 以上。

全链路可观测性：“当前每个移动 App 都希望留住更多用户，增加访问时长和浏览深度”，弘禹坦言，因此可以看到用户长时间使用产品时，在各种复杂无线环境下用户难免面临一些弱网掉线、无响应无返回、异常白屏等情况，叠加东西向多容器技术选型、南北向从业务到基础的链路复杂性，就要求终端团队对全链路有能力进行贯穿治理，“明确使用过程中压倒用户的最后一根稻草是什么”。

新兴热点融合

移动互联网的爆发和发展，带动了相关技术的一并兴起。对于移动端来说，目前也正在向智能汽车、IoT、元宇宙等方向探索。那么这些新兴技术热点又会对终端产生怎样的影响?

“我觉得多端多设备一定是个趋势”，弘禹谈道，所谓的多端多设备正是华为提出的“1+8+N”战略，其中，“1”代表手机，是核心；“8”代表 PC、平板、智慧屏、音箱、眼镜、手表、车机、耳机；“N”代表摄像头、扫地机、智能秤等外围智能硬件。

可以看到，今天的 IoT 或者车载系统，都是朝着智能化方向去发展，因此，未来可能会有一种新的体验出现：一个手机代表一个人的主身份跟汽车或 IoT 连接，即多设备之间发生无缝连接，形成一个智能体验闭环。落到具体技术的话，未来可能会对应出现端端协同、端云协同以及端和周边智能协同的新的技术体系或者方向。

弘禹判断，边缘化大概率是会来到的，但最终要看除了手机之外，其他设备多久能够实现智能化、数字化，如果这一天早点到来，也进一步会加速端交互上的协同。“我觉得这会形成一股新的增量，然后让各家 App 考虑如何将 App 作为用户一个身份，去跟更多的场景去融合和无缝连接。”

展望未来

综合多位技术专家的预判，终端领域已经行至新时代的前夜，那么在这个阶段，终端领域如果想有较大发展，需要解决哪些挑战，同时此阶段又蕴含哪些机遇？

综合两位受访专家的看法，终端领域若要有重大变革，需要考虑以下因素：

- 硬件、软件、应用的关键因素何时能够变化。这里，硬件即指终端设备，比如 VR、AR 或者能够替代手机的其他设备何时能够出现；软件即指与硬件相辅相成的操作系统，比如面向多端多设备协同的新的操作系统何时能够出现；
- 云和无线何时能够成熟。受端上摩尔定律和 CPU 的限制，如何在足够的带宽延时情况下，与云计算结合提升用户体验。

同时，也需要解决以下问题：

- 又快又好地面向业务交付。整个终端领域，无论是 Web 还是移动 App 开发，都面临着纷繁复杂的基础设施、层出不穷的看似新兴的技术方案的困扰，同时隔离的操作系统平台、高度分化的 App 版本和发布渠道等问题严重制约了 App 迭代的效率，而 Web 在移动场景下的体验和原生调用能力，也限制了它的应用场景。

- 在交付的基础上理解业务、通过技术帮助业务成功。聚焦到技术本身，互动游戏领域的工程师，需要更专注在图形技术、渲染技术，为业务抽象出超前、有创意的实现能力；中后台（运营、商家、广告主、政企事业）等业务域的工程师，需要更专注低代码、业务能力编排、流程自动化（RPA）等技术，大大提升客户解决事务的效率；短视频、直播领域的工程师，需要专注在音、视频等媒体技术，为用户提供一流的富媒体体验。（部份）桌面客户端的工程师，也需要关注跨平台研发技术（Node.js、Eletron 或者 Qt 等）。

当然，处在巨变前夕，除了挑战，两位受访专家也在本次采访中预判了最有可能成为下一个风口的技术细分领域：

- 3D/XR。目前，各家终端应用都在往这一方向做相关探索，或许在 2023 年，3D 直播、3D 绘本等内容场景会涌现；
- 原生或跨端体验上或有创新；
- 工程体系上将进一步向平台工程上演进。

不过，弘禹也谈到，他判断短期之内终端生态并不会有爆发式变革：终端大概率以手机为主，设备主场景依然会是以 iOS、安卓为主流的平台。但是正如前文提到，这套体系会深度将前端及客户端进行整合，并且打磨用户体验仍是当下的重中之重。同时，对于业界来说，还要找到降低这部分的复杂度和门槛的有效方法，才能让终端工程师能够以较低复杂度的技术方式，持续提升业务交付效率及全链路体验。

对工程师说

随着终端概念的持续深化，此技术领域也将变得更加完整且全面，这也意味着不同的场景会逐渐产生差异化的技术特征和领域性。这对于一个技术领域来说，无疑意味着向前跨了一步，但对于身处其中的工程师来说，身上的担子却更加沉重。

2022 年是一个非常特殊的年份，大量的互联网公司光环褪色，越来越多的公司逐渐更务实。在降本增效的大环境下，工程师们既要保住基本盘，又要面对即将到来的新技术快速进行转型，“学习力、适应力和韧性是非常重要的，而且要有一个变革自己的心态”，弘禹提醒道。

另一方面，舒文也谈到即使面对诸多压力和挑战，倒也不必刻意追求时髦热门的技术。这里，他也提出几点适用于绝大多数终端工程师的基本逻辑：

- 踏实专注于当前所处的技术领域；
- 适当跳出当前的技术 Scope：如 Web 的工程师一定要跳出浏览器宿主的视野关注关联技术（如 Rust）、移动 App 的工程师避免被操作系统绑架关注跨平台交付技术；
- 面向未来的方向，要有好奇和探索之心：如从事 C 端领域关注 3D/XR 技术；to B&G 领域关注可逻辑化编排的低代码方向；做应用交付要关注基于云化的全栈技术。

“持续面向业务交付当下和未来的技术价值，不痴迷于眼花缭乱的新奇巧技，才是追求专业的中正之道。”

采访嘉宾

郭虹宇（弘禹）：阿里巴巴终端委员会核心委员，大淘宝终端体验平台负责人。

舒文亮（舒文）：阿里巴巴终端委员会核心委员，大淘宝跨端技术、Web 技术负责人。

展望技术人的 2023

降本增效的大环境下，该不该砍掉测试工程师？| 展望测试工程师的 2023

作者：李冬梅

提起软件测试工程师，更多人联想到的是工作轻松，不需要敲代码、工资高、涨薪快，甚至可以发展成为管理层，那么现实真的是这样吗？

刚刚过去的 2022 年，测试工程技术领域又有哪些重要进展？在未来的发展中又面临着哪些机遇与挑战？InfoQ 年度技术盘点与展望栏目特别邀请到了腾讯技术工程事业

群，米大师计费测试负责人朱志杰，请他来和大家分享并展望测试工程师的 2023 年。

朱志杰，15 年腾讯米大师计费服务端测试及团队管理经验。专注于米大师分布式计费、营销、风控、账户以及 TDSQL 数据库等产品的高一致性测试能力构建，支持公司游戏、腾讯云、广告、视频号等业务的计费测试工作；牵头构建了计费完整的质量支持体系，建设了基于服务端的自动化功能/性能测试、现网引流对比应用、环境治理等测试能力。目前聚焦于基于 DevOps 的研效改进工作。

视频回放地址：https://www.infoq.cn/video/z0xL4IhkjkFQiWp5eq50

以下内容节选自当天的分享，InfoQ 做了不改变原意的编辑：

> **InfoQ：志杰老师您好，您当时为什么会选择测试工程师这样一个职位，可以分享以下您的成长经历吗？**

朱志杰：其实我选择从事测试岗位是具有偶然性的。我在大学期间主要学习.NET、Java 以及 PHP 等开发，也做过 OA 类的项目，一直想从事应用开发工程师。毕业进入华为后，被内部分配到了华为智能网平台的平台测试部，也是在工作之后才发现原来测试没有想象中的基础、简单。我当时做的是在 Linux 服务器平台上进行大数据量实时处理的系统，而这之后也做过一些跨城容灾性等非常具有挑战性的项目，慢慢地就扎根在测试的行业之中了。

我在华为做了两年测试后，后面来到腾讯。和在华为智能网中做电信的计费测试相类似，我在腾讯也恰好是在一个做聚合支付的内部计费部门。直白点来说就是早期的 Q 币计费，是与我个人蛮契合的产品和项目。

我在腾讯计费团队，从最初没有 DTO 分离，与开发相捆绑，给开发做辅助的测试角色，慢慢地将自己的测试能力、测试流程化都发展了起来。随着公司的业务开展，我也参与了公司开放平台、国际化、腾讯云等产品，渐渐地在这里扎下了根。部门的项目在发展，团队也在发展，我个人收获了很多锻炼和成长。

> InfoQ：您从事多年测试行业，是如何从普通工程师成长为资深工程师的，中间又付出了那些努力呢？

朱志杰：随着业务的发展，我个人得到了很多锻炼和提升的机会，逐渐获得成长。

同每一位刚进入职场的人一样，刚开始也面对团队缺乏测试方法论、测试能力、测试平台，以及流程规范，这时我就主动站出来推进测试的流程化、工具平台化，完善测试能力，这也是我收获的个人第一阶段成长。

在第二阶段中，随着腾讯的业务做大，我们迎来了开放平台、腾讯云的产品接入，以及王者荣耀等国民级游戏的出现，数据量的增长也给我们带来了非常多的挑战。开放平台的服务质量问题，以及随着王者荣耀所带来的线上最大容量支撑问题，这些涉及到线上拨测、压测等技术，促使我们在这方面能力体系上进一步打磨夯实。随着这些能力的构建和打造，我个人的技术能力在成长，测试团队也随着发展和建设起来。

至于第三阶段，目前我的工作重心主要在于团队的测试效率建设，随着业务服务化发展后，在研发体系上会更倡导测试能力左移、组织效率提升、DevOps 的转变等新模式去契合。

从我个人角度来说，随着业务的发展，我们需要去解决伴随业务出现的问题，去构建测试能力，并在这过程中收获技术上的成长与经验的提升。此外，随着团队规模壮大，

也拥有了自己的团队、参与公司测试通道与人才能力的培养，以及公司层面上技术的协同，这一路上持续发现有很多锻炼与发挥的机会。

> InfoQ：在刚才提到的许多测试工程的细分方向，您平时的工作内容，或者说最具有挑战的是哪些部分呢？

朱志杰：我们是服务端的测试，因此会更偏向于基础架构上模块化的测试。

从技术层面来说，我认为要想做好这方面的测试，必须要能深刻理解架构。以高并发存储模块为例，我们需要对其架构进行评估：其数据更新应用、缓存能力的设计能否满足高并发的查询？热点是否过于集中？高并发时数据如何防重？如何进行服务降级？是否会发生内存数据错位？在分布式容灾能力下要如何保证数据的一致性？这些都需要对架构有较好的理解。

我们还要考虑应用场景。新老业务的版本迭代中要如何保证向前的持续兼容？团队应保证有自动化的能力覆盖。

此外，在业务方面，将业务从旧模块割接到新架构时，要如何做到百分百的兼容？人工测试是否会导致的场景覆盖不充分？是否会有场景遗漏？其中的数据多样性与真实性要如何考虑？一个很好的方法是借助线上场景的对比模拟来解决这些问题。

以上这些都是我们在模块测试中会更多考虑的方面。

> InfoQ：我们平时提起测试岗位可能都会觉得很厉害，是需要掌握很多技术的。那么作为多年从业的老开发者，您觉得实际是如此吗？

朱志杰：其实在互联网行业中，测试都是蛮苦逼的。我们的定位是项目中承前启后、

非常重要的助手，不仅要支撑研发交付高质量的版本及功能，同时也要协助线上运营，提前排查相关风险及问题。因此，我们的工作就是解决问题，是非常务实的。

> InfoQ：会感觉压力很大吗？

朱志杰：每个岗位都有各自的压力，侧重点不一样而已。我们作为互联网服务端的测试，和一般业务测试不同。简单点点鼠标、理解测试理论、设计测试用例，这些对于我们来说都是非常基础的。

我们希望同事们能成为更全栈的工程师，去了解更多、更广的方面。

- 业务相关能力。由于我们是基于服务端的业务，如 Linux、Docker 容器、Kubernetes、MQ 或卡夫卡消息队列等组件，这些都是我们需要掌握的服务器组件。
- 测试开发的能力。针对 Go、C++、Python，以及 Java 等业务服务做自动化时，要根据业务的测试需要去应用，熟悉相关开发语言及其 IDE。
- 熟悉 JMeter、CPP check 等市面上常见的测试工具，以及针对特定业务场景所需的工具，如数据库 ACID 场景的测试方法。
- 流程管控。掌握代码库 Git 的管理、流水线，甚至是 Excel 的度量分析等技能。

> InfoQ：在您看来，作为一名优秀的技术工程师，除了前面提及的技能掌握，还需要逻辑性、思考能力等特征吗？

朱志杰：我认为优秀的测试人员需要具备三个通用素质。

- 沟通与思辨能力。作为开发与后端运维间承前启后的角色，测试要积极对开发运

维与产品进行沟通协调。此外，也要能快速把握业务需求，明确业务需求从开发端传递的是否准确，要如何从前往后进行信息传导，抓住问题本质及需求改造点，评估其可能造成的影响。

- 快速学习能力。无论是校招还是社招，学习能力对测试都是非常重要的。面对业务测试时要能快速熟悉架构、掌握需求，快速将复杂场景环境构建起来。如果新业务需要用到卡夫卡或 RabbitMQ 消息队列，测试人员要能快速掌握对应工具。
- 主动性。成绩突出的人往往拥有更强的主动性。主动思考所负责项目产品的痛点，主动发现问题，找到解决问题的机会点。此外，在交付版本或业务后，要有将其中的能力或问题提炼出来，作为公共能力或公共模块的意识。

> **InfoQ：如果您面试的求职者具备了以上三点通用素质，会是加分项吗？**

朱志杰：可以是加分项，但对技术岗位而言，测试开发能力也是很重要的加分项。

> **InfoQ：测试开发工程师要如何做好职业规划？职业的发展路径又是怎样的呢？**

朱志杰：腾讯有设计技术与管理两个能力提升通道，我个人同绝大多数人一样是走技术通道。但如果公司没有较强的职业指引或规范化工程师培养，可先以全栈的能力做自我要求。

测试理论只是个非常基础的概念，我们要在此基础上深刻掌握所负责的业务。要能从开发或产品的角度掌握业务，也要从运维的角度使用业务，我们才能在评审会议上提出有价值的问题，才能发现项目潜在的问题和风险点。

我们也要能结合业务需求，在具体问题解决上深入掌握相关技术。在遇到问题时，不能死读书，而是要从公司内部或外界同行中寻找好的解决方案。以数据库产品为例，

我们要参考并尝试完善别人在面对 ACID 或高一致性问题上的方案，研究 ACID 场景上的差别，最终用其解决了实际问题，才能算是真正掌握了一项技术。

从长期来看，我们个人也需要打造自己的专长。可以深入研究自动化，做服务端接口自动化、基于接口协议自动化生成、通过模型化场景生成用例，甚至可以做流量自动化，将线上流量引入测试环境中进行回放。我们也可以将自己打造为效能专家，通过全流程进行效能分析，从而推动提升效能。

> **InfoQ：工程师常常要与产品经理及同僚交流，但很多技术人实际并不擅长沟通。您是如何看待沟通能力对技术人的重要性呢？我们又要如何提升自己的沟通能力？**

朱志杰：我在回答前面问题时，将沟通思辨放在了第一位。我认为沟通会影响很多方面：

- 有效沟通可以快速把握需求，把握改动。
- 快速准确地将所发现的问题同步给相关人员。
- 推动问题或改进流程化需要通过沟通调用开发、PM，及相关业务人员的配合完成。
- 有利于合作关系的维护，互利互补开发与测试的关系。
- 个人能力的呈现。

好的沟通不一定需要出口成章的能力，我个人的沟通能力也不算很好，关键是要能表达清楚问题和做事的思路，这是非常值得从日常工作中锻炼和提炼的。

> **InfoQ：对于善于埋头做事、不爱表达的技术人而言，您有什么可以让人走出舒适距离的方法论吗？**

朱志杰：这受个人性格方面的影响，自驱力很强的人会强迫自己进行表达，或许很有帮助，但对我而言却不一定要这样。在工作的小团体中，大家都非常熟悉、没有太多戒备，我们能将事情讲清楚，在向上级表达问题时有很好的呈现，就是一个很好的沟通形式了。

> InfoQ：您认为测试开发工程师在 2023 年所面对的最大技术挑战是什么呢？

朱志杰：技术上可能没有什么太大的挑战。从大环境而言，近两年行业上都在关注降本增效，这对测试而言也是较大的挑战。大厂相对会看得更全面，但对中小公司或团队而言，如果测试左移、开发自测，或者产品运维自测上线，那么我们是否还需要测试呢？这个问题我觉得关键在于测试的价值与产出的呈现，在于项目对测试的依赖。

举例来说，项目的左移意味着开发要在较好的测试环境中进行测试，但这对开发而言是具有困难的，如果测试可以提供平台化工具、保持流程通顺，并提供验证、引流的能力，协助项目左移，那这不失为一个很好的解法。

测试的价值也体现在 Ops 运营类的项目中。测试可以为运营的灰度减少问题出现，提供上线后除监控外的其他测试手段快速发现问题，通过拨测、上线后压测、容量评估等手段，对 Ops 环境右移的能力给予有力支持。

在 2023，甚至是 2024 年，测试人员可能都要面临这一挑战，但我们也可以借此重新定位自己的价值，调整提供价值或服务能力的方向，主动适应环境。

> InfoQ：在降本增效的大环境下，在职的测试人员也会面临就业压力。那么对于想要成为测试工程师的年轻人，您会给出什么建议或忠告呢？

朱志杰：持有“测试比开发简单、轻松”想法的人，所能做的测试内容是有限的，在测试中产生的价值也是有限的。我们也要将自己培养为全栈工程师，找出项目的问题所在，并随时准备为其提供解决问题的能力。

> **InfoQ：您反复强调了全栈的能力，那么除了测试工程师之外，您会建议不适合做测试开发的人，求职 PM 等岗位吗?**

朱志杰：每个人大概都要找到合适自己的工作。我前面的建议对刚毕业，或刚刚走上测试工程师的人而言会更合适。如果你具有沟通推动，以及很好的组织协调能力，那么转职 PM 未尝不是一个好的方向。如果你对产品比较敏感，那么产品方向也是可以的。但如果你只想做业务测试，也可以走交付测试的方向。因为我个人主要关注服务端模块化测试，因此会更强调全栈的能力，要求有更高的平台化能力。

> **InfoQ：过去一年中，在事业上您比较有成就感，印象较深的有什么呢?**

朱志杰：我做过的很多事情都是微不足道的。比较有成就感：一是在过去大环境动荡的 2022 年中，我们团队上下二十余人整体都比较稳定，大家携手共进攻克挑战、提升效能，没有因为过多的冲击而动摇想法或信心。

二是我们在去年成功承担了腾讯广告计费能力下沉项目。我们通过完备的测试方案、测试引流对比、反向对比验证，完成了系统架构和计费能力的合并融合，整体构建过程没有出现故障，我认为是非常了不起的。项目历时一年半，从 2021 年下半年直到 2022 年上半年完成的主要项目进展，并在 2022 年下半年完成了收尾工作。在这一年半的时间中，大家心里的弦一直都没有放下来。

> **InfoQ：在这样的大环境中承担如此重的任务量，您是如何缓解团队紧绷的情绪呢?**

朱志杰：我认为测试首先需要有一个很好的抗压能力，其次也要能正确地分析工作，让团队成员在面对问题时有章法，有思路。在确定方案与策略后，按照节奏推进，大家都会心里有底。

此外，我们其实负责了腾讯收入的百分之七十的虚拟计费，每分每秒都有很高的交易量，因此基于我们测试保障体系、自动化能力、引流能力、代码安全性能力，包括线上拨测与压测等比较完备的保障能力，让大家还是比较放心的，没有感到特别大的压力。

最后，我也会给团队一些宽松度，让大家能较好地劳逸结合，不会产生过多的疲惫感。不会紧盯进度，但如果成员告诉我卡进度了，去协调人力，涉及较大的方案时协助拉相关的专家一起评审，帮助给出建议。

> **InfoQ：您可以给我们推荐一些您最近在看的，比较有启发性的测试相关书籍吗？**

朱志杰：前面提到过技术基础，那么《Linux 从入门到精通》这本我十年前看过的书还是很值得看的。此外，《谷歌软件测试之道》这本书中介绍的很多谷歌测试的思想也很值得借鉴。达里欧的《原则一》我看过，《原则二》还没看，但书中介绍的关于工作透明化、管理方式也很具有启发性。

> **InfoQ：网友提问，2023 年测试工程师要如何做职业规划？您可以说得更具体些吗？**

朱志杰：在未来两年中，我个人会更侧重于测试支持研效的建设，通过提高测试效率帮助研发提效。不止腾讯，对业界内很多公司来说，我觉得在这个方向上都可能在做一些事情，我觉得往这个方向可以去努力一下。比如你怎么样把自动化变得更简单一些，将自动化的成本降得更低；怎么样把一个线上的流量用起来。

通过用流量建设减少自动化，或者说怎么样去除了满足自己的测试需要，去帮助开发去做测试左移，开发自测，支持他在测试环境，自动化的能力，支持他一些工具化的能力，帮他把整个理顺拉通。这个也是我个人 23 年要去做的一个方向。

> **InfoQ：在 2023 年的开端，您能为测试工程师们送上一些祝福吗？**

朱志杰：2023 年是新的一年，祝大家能在测试的工作中找到自己的成就感，找到让自己变强的方法和钥匙，也希望 2023 年测试人员们能更自信。最糟糕的时候已经过去了，未来是美好的，我们要充满憧憬。

一个架构师在 2023 年需要掌握哪些“必杀技”？

作者：褚杏娟

> 2022 年，架构领域发生了哪些值得关注的事情？一位架构师必备哪些技能？哪些架构趋势需要掌握？1 月 6 日晚 8 点，Nacos 和 MSE 创始人、阿里云高级技术专家彦林彦林为我们带来 2023 年的架构师发展指南。以下根据直播内容整理，有不改变原意的删减，完整内容可点击查看完整回放视频。

主持人：今天的主题是 2022 年架构发展的亮点，以及架构师成长的路径。我们请到

了老朋友彦林老师，先请彦老师来给大家打招呼。

彦林：我是来自阿里云的彦林，很荣幸有机会跟大家一起聊聊今年架构的一些趋势。我从事软件开发 10 多年，然后赶上了在阿里做百万实例架构的演进，包括最近 3 年帮助阿里云从自建 IDC 全部搬到了公有云。我也有幸参与到阿里巴巴内部的中间件产品的开源，然后通过开源做技术影响力，也因此和我们 InfoQ 的小伙伴结缘，和更多的开发者结缘。随着开源的影响力越做越大，我们也把开源的一些技术解决方案，使用产品化的模式，通过阿里云去做输出。

目前我主要在阿里云负责微服务产品相关的开发工作，希望后面有更多机会交流。今天晚上主要跟大家聊一聊技术架构的一些趋势。

详解架构技术发展

主持人：彦林老师先带我们回顾下，今年架构领域都有哪些比较重要的进展和亮点？

彦林：我在阿里云每天见非常多的客户，经常跟客户聊架构选型相关的事情，今天以我的视角带大家做一个简单的回顾。一般找我们聊架构，大家都会讲到不同类型的架构，比如单体架构、微服务架构、Serverless 架构以及低代码等等，在不同的场景怎么去选型？哪些场景适合哪些领域？现在每个领域的业务成熟度是什么情况？

面对这几个问题，我们会简单进行分解。首先，单体架构现在基本已经成熟了，各个语言包括 Java、Golang 做单体开发都比较成熟。所以在过去一年 Java 做了一些变化，为了更好地适应云原生，它会通过 GraalVM 技术、Spring 3.0 等关键大版本的发

布，让 Java 变得更轻量、启动速度更快。其他的语言比如 Go，天然就是云原生的语言，所以它在做单体分布式开发的时候，整个启动速度都比较快，相对会比较成熟一些。

第二，微服务。微服务整个领域在过去一年还是发生一些变化。在最早的时候，大多数都是一些互联网大厂在使用微服务，但是去年看到各行各业开始更加深入地去使用微服务。因为疫情虽然给大家带来了一些痛苦，但也给整个数字化的进程带来了一些助力，包括数字化系统的迭代在变快、复杂度在变高，因此微服务在加速渗透到各行各业。在微服务架构的选型过程中我们也可以看到，由于 Spring Cloud 现在成熟度已经比较高、Spring Boot 3.0 最新的架构再往前演进，Java 在解决云原生应用的一些启动速度和加载速度问题。

我们还看到另外一个非常有意思的现象：大家都在往 Go 语言发力。微服务在 Go 语言的整个发展趋势在变快，也标志着今天 Go 语言在做微服务分布式开发方面，进入了一个爆发阶段。

互联网最早以阿里为代表的一些做交易系统的厂商采用 Java 较多，但是随着传统厂商慢慢进入数字化演进过程，他们就从 C++、PHP 开始向 Go 转型，而且转得速度在变快，包括阿里微服务在这方面也做了一些布局，这是今天后端开发方面的一些变化。

在前端，Node.JS 现在也越来越成熟。一般在架构简单的时候，可能采用前端网关直接挂后端应用的方式。随着客户规模变大，就会用 Node.JS 做一代理层，这样有利于前后端的分离，让前端的整个迭代速度更快。在阿里内部分中台、后台、前台分三层，后台应用需要的是稳定性，中台需要的是支撑效率可复用性，前台需要的是业务要跑得足够快，这方面微服务也比较成熟了。

最后讲两个技术趋势。

第一，Serverless。2022 年可以被认为是整个 Serverless 的元年，阿里云在 2022 年 11 月份的云栖大会上，阿里云总裁发布 Serverless 的战略，提供了一系列 Serverless 的产品，从整个计算、存储、网络，到上层的 FC 产品，一直到数据库的 Serverless，演进速度都在持续加快，相信在 2023 年速度会更快。

在 Serverless 架构方向引导之下，我们可以看到事件驱动也在慢慢变成主流。一个个函数之间通过事件驱动模式进行解耦，架构的容错率会非常好。在云厂商推动下，我们可以看到一些新兴的中小型业务、偏前台的业务，包括计算性的任务，都大规模采用了 Serverless 的架构。

第二，低代码。低代码现在每年的增长速度有 40%，速度很快。阿里最早推出的一些产品叫宜搭，基于行业性的解决方案，它有行业属性、业务属性，现在也比较成熟。

从技术上来说，低代码是一个趋势。举例来说，它会通过一些图形化的拖拖拽拽，将模块进行组装，对于业务比较简单的场景，开发效率还是比较高的。低代码和无代码这种拼装的模式，可以让更多的人参与设计数字化的建设过程。阿里在微服务上也做了一些事情，比如开源了一个云原生应用脚手架（https://start.aliyun.com/），能够帮助开发者快速地勾勾拽拽，把一个依赖包快速组建起来，这也是低代码的一种模式。

相信随着整个技术基础架构中间件稳定成熟之后，在 IaaS、PaaS 之上，基于更高层的低代码会成为未来的趋势。从业务方面说，各个行业的 SaaS 会成为一个趋势。

社区：多长时间重构一次？

彦林：重构不是按照时间决定的。重构的第一个原因是迭代效率变低了，比如代码的研发效率越来越低，严重阻碍了它的迭代速度。第二个原因是稳定性，比如开发者修改一 Bug 引出了多个新 Bug，代码变得很复杂了。第三个原因是扩展性不足，性能无法满足要求，比如做大型运营活动时，单体无法满足需求，企业就需要做一些分布式的改造，把主链路和分支链路分开，做更好的扩展性和弹性。

微服务：真的被滥用了？

主持人：您在去年的分享中谈到，微服务每年有差不多 20%以上的增长。今年，微服务的增长趋势如何？

彦林：从三方报告、阿里、官网以及 Star 上的数字来看，今年的增长速度是 15%左右，比去年同比增长速度稍微慢了一点点，和疫情有很大的关系，但其实绝对数字差不多。在中国，每年采用微服务架构的有几十万家企业，虽然增速变慢了，但是每年采用微服务架构的绝对数字依然非常大。另外，整个技术的热度也在持续加强。

主持人：今年有一个争议点，马斯克接手推特之后，吐槽过推特的微服务架构，说运行要调用 1200 多个 RPC，但其实真正需要的微服务不到 20%。其实像 GitHub 前 CTO 也提到过微服务滥用的问题。这就会引发大家一个思考，今年的微服务真的在被滥用吗，您怎么看？

彦林：从我们的角度看，无论是微服务还是单体应用，都是一个技术选型。随着业务公司的发展、需求的不断增多，业务变得更复杂，我认为更多是管理和产品的问题，

不是技术的问题，我们要把技术和业务分开。

“什么时候采用微服务”是技术人要回答的，比如企业只有一个业务系统，其实是不需要上微服务的。那一个公司几十号人、几十个子系统，还要单体应用吗？效率能解决吗？架构其实是一个拆分，服务规模大了后架构师在做架构的时候，就会向两个方向去解决业务的复杂性问题。针对有几十个研发同学、几十个子系统的情况，就需要做一些分支，这个时候用微服务的投入产出比会比较高。单体到微服务有一个交叉的技术曲线，可能是 10 个应用、10 个子系统，达到拐点之后采用微服务确实效率比较高。但在没有达到拐点时，提前采用微服务可能会发现一些问题，随着业务规模增大，微服务后面才可以慢慢释放红利，这是一个技术选择问题。

总的来说，没有最好的，只有最合适的，当系统的复杂度达到 10 个研发、10 个子系统以上，可能就适合用微服务了；当达到 100 个研发、100 个子系统以上，可能就要把公共服务抽成一些中台服务去做，这是架构师基本的服务化、分层设计理念。用这样的理念设计后，我们的组织架构也随着技术架构去调整，让业务尽可能跑得快。

社区：传统项目准备使用微服务，在转型时主要从几方面去转？

彦林：互联网的第一波，包括云的第一波、数字化的第一波已经过去了，第二波就是传统客户包括政企。大部分客户会问，CTO 应该怎么决策？最佳的解决方案是：第一，新老架构前面都加一层网关，用网关将老的服务路由到老的单体架构上面，新的架构采用微服务，让新的业务先跑到新的架构上，老的架构慢慢往前演进，这是一个长期的过程。

第二，新老架构之间也可以通过网关进行代理。有许多厂商向传统公司提供服务，多

个服务之间怎么进行快速组装、相互调用健全，怎么组装更高级、更丰富的上层服务？都是通过网关快速做 API 的聚合与整理等，这都是这些传统企业的诉求。通过网关可以把新老系统、多个 ISV 系统全部互联互通，这是做现代化架构演进或者云原生架构演进最快的模式，让新的架构、新的模式讲清价值、慢慢迭代，把老的架构进行收敛。这就是我们现在看到最好的一种模式。

主持人：马斯克说推特在一些国外地区会比在美国的运营速度要慢很多，产生速度差异的原因是什么？

彦林：从理论上来说大家应该都是一样的，但是当组织离核心业务、权力机构远的时候，很多的组织效能会做一些弱化的事情。我认为更多的是组织的问题，技术的问题并不大。马斯克只是说了一个现象，被一些人拿出来做新闻热点炒作而已。

其实我们也能感觉到，技术和产品有时候有一些博弈，产品总想让技术多做需求，但是需求越多系统就越复杂，随之依赖越多、链路越长，风险也就越大。

阿里巴巴在 2012 年左右的时候，链路也很长、很复杂，所以我们当时抓体验，比如做一秒钟战役，即所有的请求要在一秒钟之内做完。为此，架构做了很多改变，比如分布式架构分为核心链路和非核心链路，核心链路我们用 RPC 微服务同步解决，非核心业务通过异步发消息去解决、通过事件去解决。这样做稳定性、扩展性会有大幅提升。

社区：微服务的注册中心跟 K8S 云原生是不是有些冲突？

彦林：这个问题非常专业。其实没有冲突的，有一些交集。因为容器和微服务是正交关系。注册中心是微服务的存储，etcd 是 K8S 的存储，是分层的。

云原生架构：容器和微服务搭配提效

主持人：云原生架构的技术体系是怎样构成的，怎样才算是云原生架构？

彦林：最早提出云原生概念的是一家小公司，他们以 Spring Cloud 为理念去宣传了云原生技术。在之后 3-5 年的发展过程中，CNCF 重新定义了云原生，大家有了主流的认知，即采用了容器、微服务和不可变基础设施这些关键技术栈的，我们就认为是一个云原生架构。

在做架构演进和变化的过程中，为什么采用云原生架构？云的本质是弹性，每个公司都有低峰和高峰，怎么更好地利用云的弹性、让自己的资源成本和研发效率得到进一步提升才是本质问题。云原生架构比如微服务，解决的核心问题是研发效率问题，因为业务规模上去，复杂度提升，所以需要微服务这样解耦的方式去解决研发效率问题。容器解决的是运维效率，它可以让运维效率得到空前释放，使运维从体力活变成了技术活。

容器和微服务搭配使用的原因在于：第一，一个巨大的单体应用没有调度空间，只有把资源打碎、拆分之后，才可以充分使用资源池，即业务拆分合理之后，能够让开发者更好地调度、提高资金利用率，这是微服务对容器的一个好处；第二，使用微服务后，运维可能会从早期的 1 人运维变成 10 人运维，成本会上升。而容器能帮助微服务解决运维的问题，让每个开发有工具自己写代码自己运维，把整个生命周期管理起来。因此，容器帮助微服务解决了最核心的运维痛点。

总的来说，容器和微服务做搭配，一个解决运维效率，一个解决研发效率。容器帮助微服务解决了最大的运维痛点；微服务把资源打碎，让容器充分地进行资源调度，提高整个资源利用率，所以它们是相辅相成的技术。

社区：云原生网关很热，能展开聊下选型么？

彦林：传统的网关在云原生时代确实遇到了一些挑战，这些挑战在于传统的网关包括最早的单体架构，在不用容器的时候，它的调度和发布都不频繁，因此发布过程中有一些流量的损失问题，但也不大；老的业务，不是端到端的业务、视频的业务，对稳定性要求也不是特别高。

但随着数字化深入，云原生技术的采用就会引发一些问题。第一，对整个连接的稳定性、规则热更新有实时的要求，对网络稳定性的要求会更高。第二，云原生背后隐藏了一个标准化的事情。在云原生时代，未来的网关的 API 标准到底是什么？其实 K8S 通过 Ingress 的 API 定义了网关的标准，这就解决了技术选型的一个痛点。第三，多语言的扩展性。传统的网关在语言扩展性以及热更新上有很大区别，所以面对未来的云原生网关，必须解决掉这些问题。

我们认为云原生网关可能是代表下一代网关的技术，不仅阿里巴巴开源 Higress 进入这个领域，CNCF、APISIX 等等都在加速这个过程的发生，如雨后春笋般地进入云原生网关领域。大家也意识到今天在微服务里，网关是非常重要的角色。云原生网关已经形成了一个技术热点，相信在 2023 年会得到高速发展。

社区：IaaS、PaaS 和 SaaS 的商业模式哪个更好？

彦林：当前来看，IaaS 是技术门槛最高、投入产出比最高模式，需要规模化；PaaS 是技术门槛居中、投入产出比中等模式，需要可复制、解决服务效率问题；SaaS 目前核心是看业务领域创新，如果能搞出钉钉类似的 SaaS 平台，价值是巨大的。未来机会在 SaaS。

“降本增效”对架构有影响吗？

主持人：今年各大厂奉行的宗旨是降本增效，这种内部政策导向的变化会对企业在架构选择上产生什么影响吗？

彦林：其实有影响，包括正向和负向的。如果一个企业真的运营不下去了就不会再为未来投资，如果要投资未来，技术架构就要演进，公司基本上都是要降本增效。采用云原生架构之后，复杂性略有上升，但是多家数据也表明成本至少下降 30%。因为容器解决了成本效率、运维效率的问题，微服务解决了研发效率问题。

但是最近我在想一个事情，云不断地把算力成本降下来了，但是人力成本却一直在提升。很多人可能在关心资源成本，但其实大部分公司现在应该更关心研发效率、研发成本，因为人力成本在不断提升，算力成本在不断下降，杠杆的拐点已经到来了，这就是为什么越来越多的人采用微服务的原因。首先，云原生架构能降本，这是一个事实；第二，提升人的效率其实也就是降低人力成本。

主持人：云原生虽然降低了成本，但是却提高了复杂度，我们应该怎么去平衡？

彦林：这其实很简单。我们要相信后浪能推倒前浪，现在的新人很多都对云原生有了解，一些培训班也在基于云未来做软件开发，新人的学习能力和知识储备都是够的。而对于我们这些老程序员来说，不跟上时代确实会有压力。技术虽然有门槛，但是通过 2-3 个月的学习，突破技术门槛和拐点之后，就能柳暗花明，充分利用工具的优势。

当然，比如在组织做一些变革和技术架构升级时，一定会有很多人阻碍你，因为大家都喜欢不变。但是不变不能带来任何变化，为了做降本增效、技术架构先进性就必须要变，而在变的过程中自然有一些人会感到不适，但这都是短暂的，当你迈过拐点、

享受到红利之后，就自然而然地会加入到推动技术浪潮的过程中。

社区：性能上升了，对于复杂业务，Go 相比 Java 开发维护效率会下降吗?

彦林：会有一些代价，Go 偏系统一些,Java 偏业务一些。根据不同分层选择合适的。

架构下一年发展趋势

主持人：下一年架构领域可能会有什么趋势?

彦林：在微服务领域也会发生很大的变化。比如大家可能觉得微服务很先进，云原生现在虽然讲得很火，一些掌握先进技术的大公司、掌握话语权的人在使用，但今年全世界的采用率可能也就 5%左右，普及度远远不够，这是一个现状。

标准化上，我们可以看到云从 IaaS 到 PaaS，再到容器（当时定义为 CaaS Container as a Service）、微服务都在做这件事。海外也有一些公司比如 Google，也想通过服务网格去定义微服务的标准，但目前大家得到的一致结论是，它不是未来的标准，未来微服务的标准还在定义的过程中。

整个行业在一起定义下一代的微服务体系。相信在 2023 年，下一代微服务的雏形和标准化会慢慢开始呈现。我认为 2023 年不仅是 Serverless 的元年，也可能是下一代微服务到来的前夕。

另外说到云原生网关，其实云原生网关是阿里巴巴在 2022 年提出的，也快速得到了行业上下游的响应，因为相对于传统的流量网关、微服务网关、API 网关，云原生网关定义了一个新的领域，是一个新的、云时代定义的下代网关雏形。

其实阿里在 2018 年就有了云原生网关的雏形，我们经过 3 年的打磨才有的一个雏形。阿里做网关做了十几年，云原生网关领域也做了 3、4 年，基于之前的积累，我们才去尝试定义云原生网关领域，帮助传统企业在云原生时代做架构的演进，也帮助现在上微服务及云原生架构的客户更好地享受云原生的红利。我相信在微服务领域，统一控制面、标准化和云原生网关会得到空前的发展。

另外，Serverless 目前是阿里云的战略级项目，今天它的标杆和可复制的场景已经非常大了。所以，事件驱动加上 Serverless 的架构，会在未来一年得到非常好的发展。

编程语言上，目前来看 Go 语言真的挺有希望。我们拜访很多大客户，Golang CPU 的利用率能比企业 Java 高一倍，这个数字还是比较可观的。Go 语言作为微服务或者云原生的一个主导语言，未来会得到很好的发展。

社区：微服务和 Serverless 会有冲突吗？

彦林：单体应用、微服务和 Serverless 其实不是互斥关系，它们都有各自的场景。对于微服务，单体应用是相对通用的一个技术架构，目前得到了行业的广泛认识，只是说有一个拐点决定了哪个时候采用它最好。Serverless 是一个架构选型，它是一个新技术，有它的适用场景。比如偏前端、简单一点的比如小程序，偏计算性的任务适合跑 Serverless；对一些大型的比如阿里巴巴大型交易系统 Serverless 是不适用的。所以这些技术有各自的适用场景，存在互补关系，而不是互斥关系。

职业成长

主持人：架构师这个职业具体是做什么的?

彦林：在互联网中，虽然我们也叫技术专家，但实际上是架构师。举例来说，有人盖房、有人设计图纸，架构师是设计图纸的，对于一家网站来说是设计网站骨骼架构的。做架构师要有设计架构的能力、画图的能力，比其他所有技术实践的人要先行一步。

主持人：有观点是架构师关注技术的广度比深度更重要，您如何看待这种说法?

彦林：从整个学习习惯或规律上来看，一般都是先有深度再有广度。只要深入地把一个技术、一个语言、一个架构吃透，做其他的很多事情都会触类旁通。如果做任何事情都蜻蜓点水，就很难把一个事情做深。我建议工作的前三年一定要把技术的基础先打牢，这对架构师的成长会比较好。在早期可能 80%是深度、20%是广度，随着技术成长足够了，这个比例再不断去调整。另外，对技术的热情也一定要有。

主持人：架构师是一个长期主义的事情，不可能在 1-2 年就完成一个蜕变。

彦林：是的。架构师是站在开发、产品、运营、技术等等最前面的人，对人的综合素质要求比较高，一般在阿里内部架构师的是点线面体的发展规律。

第一，最早进入这个领域之时，最基本的要求先是“见问题解问题”，掌握一门好的语言，能修复 Bug，但是到后面大家的发展方向、成长速度会不一样。

第二，从点到线。这对开发的抽象能力是一种考验。当每天面对很多问题的时候，能不能把问题抽象成需求，横向拉通所有问题做一层抽象?

第三，从线到面。比如把所有的需求聚集成了 10 个需求，要再从整个产品角度，把需求砍下 4-5 个，要知道做什么不做什么，有产品视角。

第四，从面到体。当技术问题解决完之后，要考虑哪个投入产出比是比较高的，怎么拉通研发团队的前端、后端、产品、技术、运营，做一个很好的技术排期，最快拿到技术结果。这里就涉及到人的协同与沟通，但并不是说沟通能力好就能当架构师，还要懂得画图纸。其实还是要以产品为核心去和上下游沟通，因为用技术语言无法和运营、产品对话，也无法和客户去对话，至少要能抽象到产品，用产品的语言才能和客户对话。

走到“体”这一步后，解决技术、产品、协同以及人等问题的能力基本上会得到一个全面的提升，至少需要 5 年的积累。所以大家不要急，一步一个脚印往前走，只要你有这样的技术情怀和追求，一定可以搞定。

我认为有技术情怀的人才能当架构师，因为我发现很多人随着工作时间增多，慢慢地淡化技术变成了管理者，变成了 PM，这就会有问题。因为如果你画的图纸不合理，最主营的技术业务不行，就会把所有的人拖下水，这就很危险。

社区：架构方面，质量工程师能做什么?如何搭建好 IT 系统的业务架构?

彦林：一般只有大厂有质量工程师角色，核心协同开发、测试、安全做好产品或者版本质量，宣传质量文化。

社区：为什么绝大多数架构师都是后端?

彦林：因为后端技术架构更复杂，能更全局的技术架构。

社区：一定要很强的编码能力之后才能当架构师吗？

彦林：不一定很强，但是需要有一定的技术积累，不然比较难与程序员对话。

社区：架构师平时怎么积累相关领域最前瞻性的一些技术？

彦林：积累深度可以看一些 paper、专利行业的动向，这在子领域里会有很多上下游的竞对、海外竞对的分析；积累广度可以看 InfoQ 以及海外架构师等最顶级的技术媒体信息，也可以看大厂分享的公众号，因为大厂有很多场景，并会根据它的场景做很多技术分享。

之后大家也可以去逛一逛线下的技术 Meetup 以及行业的论坛、分享会等等，参与开源共建，相当于跟全世界一流的工程师一起工作。在和别人沟通的过程中，也能了解到好的技术以及别人在思考哪些事情，这也是学习技术先进性、前瞻性的一个很好的模式。在参加各种论坛、与开发者相互碰撞的过程中，可以很好地帮助你进行架构规划。比如和大厂的人沟通，你可以了解他们怎么一步步走过来的，还可以根据自己公司的发展情况一步步向前演进。如果你已经在大厂，就可以深入想一些比如如何代表中国的技术，在今天的开源、技术影响力、先进性工具上，给世界做一些贡献，打造一些世界性的产品等问题。

主持人：如何理解 Serverless、低代码等对架构的影响，它们会降低对架构师的要求吗？

彦林：我认为是不会的。现在有 10 种技术选型，最早大家写代码的时候有 20 种设计模式，你只有充分掌握哪个工具适合哪种场景，才能画出最好的图纸、做出最适合的架构，只不过早期没有这些技术和工具时，做架构需要更多的思考。而今天在市面上

有很多主流选择，在选择过程中，如何根据自己的业务选择最好的架构和技术，这对架构师是一个考验。

另外，很多人做技术架构师、业务架构师，要对业务很熟悉。微服务的主要挑战不是技术有多难，而是你的业务拆分是否合理。很多人做了微服务之后拆分不合理，就会导致代价变大。所以，对领域模型要有更好地了解，有架构师底蕴，理解业务模型、技术模型，这样的人才能做好一个架构师。

展望后端研发工程师的 2023：“后端难学”源于知识体系匮乏，面试时这三点是加分项

作者：凌敏

后端开发很“卷”吗？学习路线是什么样的？就业前景如何？展望 2023 年，后端研发工程师需要提升哪些技能，做好哪些准备？1 月 4 日，微财数科资深工程师周正杭做客 InfoQ 视频号，畅聊后端研发工程师的技术成长路线与职业发展路径等话题。以下根据直播内容整理，有不改变原意的删减，完整内容可点击查看回放视频。

如何从普通工程师成长为资深工程师？

InfoQ：您当初为什么会选择后端研发这个职业方向？可以分享一下您的技术成长经历吗？

周正杭：我的职业选择其实与学校期间所学的方向、擅长的技术有关。我毕业后的第一份工作是在一家做第三方支付的公司中负责内部管理平台的后端开发，在工作一年后也接触到了公司的通用化支付产品。在这期间，我对复杂的业务流程、高并发处理，以及流量激增的应对方式产生了浓厚的兴趣，并在接下来的近十年时间中一直专注于后端研发的工作。

InfoQ：您是如何一步步地从普通工程师成长为资深工程师的呢？在这个过程中付出了哪些努力？

周正杭：其实毕业生们在刚进入职场时，常常会感到迷茫和挫败。由于理论学习环境与实际工作间的差异，我们很难将学到的点状知识应用到公司长久积累下的技术框架和业务系统中。但这很正常，学校学习的内容很难追赶上互联网的发展速度。不过，既然能加入公司，说明公司是看重我们的能力和未来发展潜力的。因此，我们要抛出这种可能的挫败感，用最短的时间做最多的事，尽量让自己熟悉工作实战的经历。

我在当时花费了两年的时间才逐渐熟悉工作中的单体应用及后期的 Kafka 使用，夯实自己技术能力的同时，也在理论设计模式和基础数据结构的实际应用中，提高了自己工作的信心。在工作两年后，我作为公司的研发主力，进入了个人工作的上升期，也遭遇了工作之中的瓶颈。在面对诸如大流量下如何对服务降级、高并发中如何进行压力处理等情况时，我们会学习开源代码中的优秀思路、设计模式、代码风格以及深层

技术，并将其应用到工作中，解决类似问题。比如，我们会将学习到的 sentinel 窗口算法应用到金融领域中的支付系统中，处理资金系统中业务异常的兼容、成功率调控等场景。这种学习也能快速帮助个人实现成长。

在工作五年后，我遇到了互联网金融的快速发展时期，并加入了一家创业公司。和其他公司的工作模式不同，创业公司没有较为健全的部门体系和职能划分，常常会有一人身担研发、测试、产品、需求数职的情况。我们需要从基础技术实现层面中跳出来，关注产品的规划，明白产品功能的存在意义。虽然很累，对个人也颇具挑战，但却能得到很全面的发展。当然我并不是建议大家都进入创业公司，这里只是提供一个思路，希望大家能多关注其他职能部门同事的工作内容，这会帮你更为出色地完成自己的工作。

在这一阶段，我作为公司技术的负责人之一，组建并带领团队在两周时间完成了一个初版金融产品的上线，并在后续的多年间无数次的快速迭代。期间也遇到了许多初创公司在技术开发工作中常见的配合问题，如项目质量及进度管理等，我们通过研究学习敏捷开发方式，引入多维度管理方式丰富日常工作经历，让项目进展更为顺畅。这段时间对我来说也是快速横向发展的时间。

离开初创公司后，我加入了现在的微财数科公司，担任资深工程师，负责公司核心的账务系统内容。从初创公司的乱战节奏转回到正规的开发模式，整体工作风格再次发生转变，与众多职能部门重新建立合作，这种差异化的工作内容也丰富了我自身的经历。

InfoQ：您在从小公司到大公司的过程中，是否会有不适应呢？

周正杭：这是肯定的，大小公司的主要差异在于其规范性。举例来说，小公司中的需求改版，只需要大家想清楚怎么做，开发自测没什么问题就可以上线，但大公司中却还是需要遵守一定的流程避免出错。除此之外，二者的技术深度与体量也必然是不同的。小公司的金融产品每月放贷量峰值可能只有 6000 万，这在大公司看来可能连一天的量都不够。因此，无论是流量激增还是服务架构的调整升级，大小公司的差异化都是非常大的。

InfoQ：您之前提到，从学校到职场的转变过程中可能会面临很多挑战。可以分享一个您之前面临过的挑战，以及您的解决方法吗？

周正杭：后端技术架构方向的主要工作内容，是向上承接后端业务研发，向下协调基础运维部门。我们既要服务后端业务系统稳定运行（监控组件日常运行、处理极端问题等），又要协调运维等部门多关注机器运行的状况。我认为我们最大的挑战是基础架构，我们不仅要关注纯技术方面的工作内容，也要特别考虑如何服务好业务。

我们先前做的内部云服务发布系统在初期阶段，利用容器化技术进行更稳定、更快速的轻量级发布。底层的 Kubernetes 最初虽然只支持 Rolling Update 的发布模式，但因为其通用化，我们并没有觉得有什么不好。在后期，因为用户及内部维度的多元化，我们对稳定性及发布策略的多样化有了更多的需求。先前单节点的 Rolling Update 在循环滚动发布时会终止老节点，导致在途流量的不稳定以及断连接异常，对于频繁为 C 端提供服务的后台服务而言影响较大。因此，我们基于 Kubernetes 特性，考虑业务对稳定发布、平台上线的需求，研发了类似蓝绿发布、金丝雀等小流量验证发布的功能，从原先纯后台的工作走向了前台业务场景，充分考虑业务需求而进行的优化。

InfoQ：现在新兴技术非常多，我们要如何保持自己的技术敏感度呢？

周正杭：在建立知识体系的基础上，我们可以上网浏览资料、现场参加活动等等。其次，我们也可以在 InfoQ 等渠道上收获他人对新技术及其应用的见解分享。我们今年也与 InfoQ 合作，陆续发布了许多文章，交流见解与思路。在遇到性能瓶颈等问题上，也会从 InfoQ 等网站高效地搜寻适合的解决思路。

后端研发工程师的硬实力和软实力

InfoQ：后端研发给人的第一感觉就是很“高大上”，需要掌握很多种技术，基础架构方向的后端研发工程师最需要掌握哪几项技术？

周正杭：很多像是分布式操作系统之类的经典理论和架构均是建立在存储、计算、通讯等后端底层上，这些协助涉及了底层文件系统进程的 RTC，并以此为基础进行同步调度。只要计算机依旧作为资源存在，这些问题就会以不同的尺度反复出现、反复被解决。

在初期阶段，我们需要了解的有：

- 网络基础通信：TCP/IP 协议、HTTP 协议等；
- Linux 基础操作指令：grep 模糊匹配、kill 杀进程等；
- 操作系统运行原理；
- 进程与线程的关系及其在高并发场景中的应用；
- 列表、数等数据结构；

- 排查找等常用算法。

在掌握一定技术基础后，我们要更深入地解决更多技术问题：

- 框架类问题：Spring Boot、网络框架、.NET 框架等；
- 持久化层：MyBaits、Hibernate 的实现原理，要如何防止 SQL 注入等攻击；
- 应对大流量，系统的削峰策略；
- Kafka、RabbitMQ 的应用场景及各自特性。

而当大家最终成长为技术专家、CTO 级别时，我们就需要关注更高层面的技术，比如 Service Mesh、配置中心等原理。

InfoQ：除了您所提到的这些技术硬实力，优秀的后端研发工程师还需要具备哪些特质呢?

周正杭：后端研发最重要的是细心。如果我们的代码进入了死循环，或引发内存泄漏，都有可能导致线上服务崩溃。

在我毕业后加入的第一家公司中，身为初级工程师的我接触到了公司的核心支付业务。因为一个错误的金额传值，线上出现了发售金额异常的错误，直接导致公司受到了损失。在后续的工作中，我会自我要求不断精进，培养自测试的习惯，严格要求自己做事做到极致。在接手公司的全国通用款产品时，我提前找测试组沟通可能涉及的测试点、异常以及风险点，自己编写了一些测试用例，并提前进行了一周左右的自测时间。在实际进入测试阶段时，我在近 200 个测试用例中没有遇到任何 bug。希望能通过我的这次血淋淋的教训，能让大家牢记后端研发中最基础的要求——细心。

其次，我认为后端研发不仅需要关注技术，也要周全考虑如何能在满足公司业务需求的情况下设计优秀的架构。程序员们或许大多只关注如何将自己的代码写好，而不愿与人交流，但其实一段代码是否符合业务目标才是我们最大的痛点。有时我们常常会发现代码并不能做到自己想做的事，因此我们一定要多思考业务，思考我们是否能对服务起支撑作用。

InfoQ：对于技术人员而言，是所有岗位都需要具备业务思维吗？还是只针对某些岗位而言？

周正杭：我认为所有的技术架构精进都应该以服务为目标。以基础架构为例，我们从单体应用到分布式应用，再到微服务的发展，不仅仅是因为代码会更高级、系统吞吐量处理更强，而是因为这种发展解决了我们对业务稳定性支撑的需求。基础架构中，借由 Kubernetes 的单节点滚动发布确实可以解决现有的发布工作，但由于我们的业务、内部服务对系统稳定性有要求，因此才需要做更多的工作以满足需求。

我认为对后端研发来说，业务思维是尤为重要的，我们要明白代码、流程背后的原因其实都是业务，只有满足了业务才能体现出技术的价值。

InfoQ：有用户提问，您认为最好的编程语言是什么？

周正杭：仁者见仁智者见智，编程语言没有好坏之分，区别只在于适用场景。

InfoQ：您认为后端是否很“卷”呢？

周正杭：我认为“卷”其实并不恰当。技术人最关注的常常是自己的技术是否高超，所用工具是否新潮，这可能也是导致外界会认为后端研发非常“卷”的原因。技术人对自

己的要求，对自己于公司的发展、对未来技术规划等等都与之密切相关。人人都希望在最短的时间做最多的事，接触更多的技术。

如何做好职业规划？

InfoQ：后端研发工程师如何做好职业规划？职业发展路径是什么样的？

周正杭：有的时候我们带着兴趣工作，会更容易达成自己的目标。后端工程师的职业发展大致分为两个方向：

- 希望深挖技术。可以通过大量实战，对基础组件研究深挖，从初级工程师、高级工程师、资深工程师，一路发展到架构师、技术专家、CTO 等职位。对公司的技术方向进行把控，对较低级别工程师进行技术上培训，从而让员工迅速提升，保证整体技术产出。
- 对复杂业务流程感兴趣。可以沿着技术管理、项目管理方向，从工程师晋升至项目经理、技术经理、技术总监等岗位，日常工作中仅一半的时间在编码，另外一半则需要游走在各个部门，负责资源协调、规范制定、资源编排、工期监督等工作。
- 其他方向。售前架构师、技术培训讲师等等。

我们需要尽快制定自己未来发展的目标，并为之努力。任何一个职位都能有自己独当一面的机会。

InfoQ：后端适合转岗到哪个岗位呢？

周正杭：这大概也是很多程序员在工作压力下辗转反侧、夜不能寐时常常思考的问题，问题的答案取决于我们所处的阶段及我们擅长的事。的确，产品经理是个不错的转型方向，但前提是我们不仅需要懂技术，也要对现有业务知识、产品流程有深入的理解，能够与其他团队进行沟通。另外像售前工程师，虽然日常编码工作不多，但却需要对公司已有技术产品有足够的了解，配合公司销售、推销产品，根据用户需求痛点进行定制化推荐。

InfoQ：您在面试后端研发工程师时，会注重求职者的哪方面能力？有哪些是加分项？

周正杭：在面试时我一般会关注三个点：

- 基础知识。初级工程师需要掌握牢固的基础知识能力，中级、高级工程师则需要对中间件使用、高性能场景下对性能瓶颈排查、微服务及分布式系统架构的使用场景、通过深层次源码解释不同注册中心的区别、一致性及高可用性的保障方式等等。
- 产品思维。这点在互联网非纯技术类公司中非常重要，优秀的产品思维不仅可以减少沟通成本、快速传达项目目标，也能在做技术的同时主动思考产品化缺陷。
- 沟通能力。这是程序员普遍的薄弱点，也是我认为的加分项。没有良好的沟通能力，外界无法得知你的想法，真正需要表达的时候又因为没有与他人足够高的默契度而难以传达自身想法。

InfoQ：很多技术人在日常生活中不擅长或不喜沟通，这种情况下要如何提升自己的沟通能力呢？

周正杭：称不上经验，只能说是实战小技巧，但有了这项技能我们才能更全面地提升自己。我先前提到看重沟通能力其实也有我本身经历问题的影响。我刚到公司时，在项目评审、需求评审等场合都不怎么说话，导致沟通成本过高。但沟通能力也不是一夜之间就能练好的，我们要先学着做一个“捧哏”，才能成为一个合格的“逗哏”。我们只有强迫自己走出舒适区开始说话，才能逐渐建立与他人沟通的默契，分享交流技术、业务知识。除此之外，沟通能力在跨部门协作时也非常重要。我在协调产品与研发之间合作时，曾遇到过因为需求沟通不明确而导致的双方不断发生矛盾，作为研发部门负责人，我需要主动打破沟通壁垒、跨部门进行问题剖析。在多轮沟通后，我们才输出了合作的规范化流程，让大家的配合更加顺畅。

InfoQ：有用户提问，“我已经 35 岁了还是一线职员，该怎么办”？

周正杭：这是一个很现实的问题。能够思考这个问题，说明你已经确定了目标，但可能还做不到。但如果你连目标都不确定，那么可能真的需要仔细想想了。我们应当确定自己的远大目标，通过持续努力，培养自己的沟通能力、产品思维，深挖组件原理、源代码等等，通过这些积累才能确定自己未来的方向。

InfoQ：技术架构与技术管理是否均需要全栈方向的发展？

周正杭：只有对自己所用的技术非常了解时，才能去进行技术框架、管理类工作。技术管理人员需要对团队与产品的稳定性负责，要有足够的知识储备，才能应付如 MQ 突然到达水位线，导致堵塞夯死情况的触发及快速恢复方式等日常的异常情况。

技术架构也是如此。公司自有架构的设计、内部效率的提升、分布式事务解决手段、分库分表时 MyCat 中间件还是 JDBC 的选择，等等，在拥有足够的基础知识储备后，我们才能在全面分析后针对应用场景给出最合适的匹配。

像是大数据方向知识目前对架构设计而言可能用不上，因此我们也可以针对自己未来的发展和技术方向，提前做出预言。

2023 年，值得关注的后端技术趋势

InfoQ：2023 年，后端领域有哪些值得关注的技术方向?

周正杭：这主要与公司所处环境相关，而公司的发展则是影响个人技术发展方向的重要因素之一。在新的一年里，我们会更多地关注效率提升，无论是云原生、Serverless、低代码化技术等等都应该提上日程。我们要通过更少量的资源投入、应用的精细化管理、快速迭代，维护或突破现有业务，从而达到更大产出。

InfoQ：您认为后端学习是否很困难呢?

周正杭：“后端难学”的主要原因是知识发展体系的匮乏。只有建立了技术框架的认知，建立了应用场景，才能明白技术究竟为什么实用。理解了操作系统对线上应用的影响，才明白操作系统为什么有用；理解了分布式缓存时如何抵挡大流量，从而保护数据库、提高相应速度，才明白缓存有什么用。我们在看书时也可以从目录的标题看起，目录的编排路径一定是由浅入深的技术介绍。

InfoQ：有用户提问，重构代码要如何打开思路？

周正杭：我们首先要确定驱动代码重构的原因，以及代码重构后能达成的结果。对于十年来一直保持 Spring MVC 单体应用基础架构的公司而言，代码复杂度不言而喻，业务逻辑全部都在一个服务节点内，那么这种情况下重构是必然的。重构可以为我们解决服务稳定性、维护性，降低内部服务通讯成本。但我们不能盲目进行重构，新框架的应用会带来技术成本投入，未知技术会带来未知异常，从而导致业务事故、生产事故问题。我们也不应直接进行根本性重构，可以从业务逻辑代码重构开始。在重构过程中我们要实时关注业务导向，确保重构不会偏离对业务的支撑。

以我们的低代码改造为例，我们的目的是提高渠道对接效率。在重构过程中，我们前期投入成本非常高，梳理了上千种不同业务流程中的共同点并界定抽象方法，但后期成功后使用非常顺畅。

InfoQ：您认为对于后端研发工程师来说，2023 年可能面临的最大挑战是什么？

周正杭：整体发展方向取决于业务，但个人而言，我们可以自我驱动，向 Serverless、云原生应用、低代码化改造等方向发展；也可以自我确立中短期发展的里程碑，比如学习一门新编程语言，完全了解并实际应用卡夫卡等等。通过连续可触达的发展规划，我们能够持续地自我激励，以高昂的情绪继续寻求发展。

InfoQ：对于希望成为后端开发工程师的年轻人，您会有什么建议或忠告呢？

周正杭：我的建议可以分为三点：

- 编程语言基础。无论是 C、C#、C++、Java、Go、PHP 等语言，掌握了语言才能

掌握工作的工具，才能与同行进行交流。

- 知识体系的建立。
- 工具、技术方法学习。利用业余实践多学习，只有这样才能提前适应工作。

InfoQ：一位合格的研发工程师需要懂几种语言呢?

周正杭：这个问题也是见仁见智。应用场景不同，语言的需求也各异。大部分公司为减少多语言技术投入，会组件化统一技术栈，因此如果只是为了工作，那么一门语言足矣。但如果目标是全栈或创业工作，那么安卓、iOS 客户端的编程语言必然不同，服务后端选择 PHP 或 Java 也是不同，再加上数据库与大数据层面，都需要不同基础。我们可以在学习新的一门编程语言后，思考对比其与已知语言的差异并分析其优劣，更好地交付应用。如果在学习新语言后发现旧语言的缺陷确实很大，那么我们旧需要投入成本去学习另一种语言从而完成应用投产。

InfoQ：请您给大家推荐几本书籍吧。

周正杭：我会从两个方向进行推荐。底层操作系统方向我推荐《深入理解计算机系统》。这本书出版时间较早，从程序员的角度深入讲解操作系统对应用的影响，在每个章节均有实战经验帮我们更好地理解操作系统。后端开发有时会忽视操作系统、JVM 调优方向，只会在 OM 问题、CPU 占用量剧增导致生产问题出现时，才会现场研究操作系统方面知识。

第二本则是经典的《人月神话》。这本书针对技术研发、项目管理等方面提供了很多参考案例。在项目进度不如意时，我们常倾向于投入大量人力追赶进度。但大面积地引入新人会带来大量工作交接及培训成本，浪费了时间也降低了效率，我们应当维持

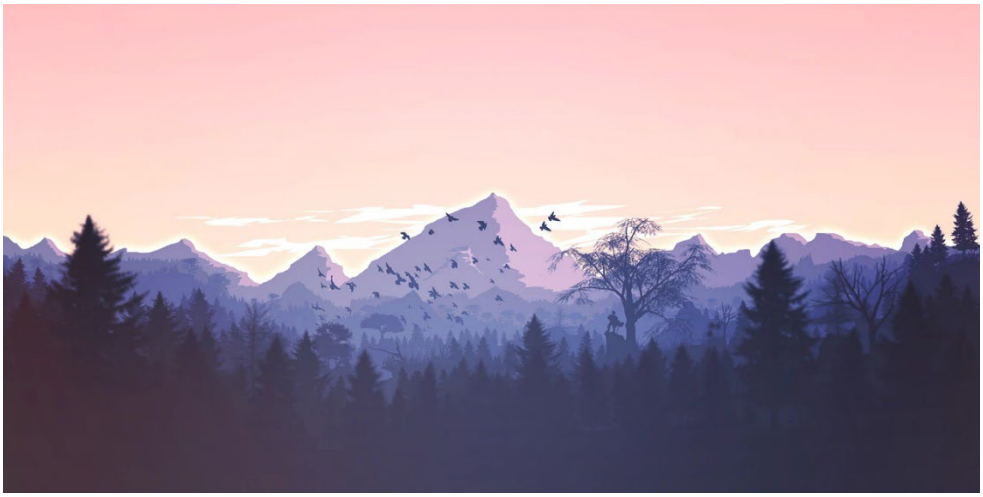
现有团队，通过合理化结构调整，正常进度内追赶上工期。

InfoQ：站在 2023 年伊始，您能为后端研发工程师们送上一句祝福吗？

周正杭：希望大家能保持对自己未来发展方向的渴望，就和乔布斯所说“stay hungry, stay foolish”，时刻清楚自己的成长目标，每天都要学习，让自己进步，积极的情绪能帮助大家快速成长。

寒冬未散，手里没点“必杀技”，怎么翻过浪浪山？ | 展望数据库工程师的 2023

作者：刘燕



采访嘉宾：关胜亮，涛思数据联合创始人，开源云原生时序数据库 TDengine 主要作者

最近几年，数据库技术十分火热，数据库开发也成为大家关注的热门方向。如何成为一名优秀的数据库工程师？2023 年，数据库工程师需要提升哪些技能，做好哪些准备？

InfoQ《极客有约》特别策划的《2022 年年度技术展望》系列直播，邀请到了涛思数据联合创始人，开源云原生时序数据库 TDengine 主要作者关胜亮，一起展望数据库工程师的 2023。

视频回放地址：https://www.infoq.cn/video/KM4KfAtfgcQbKoqZV58M

以下内容节选自当天的分享，InfoQ 做了不改变原意的编辑：

与编程结缘

InfoQ：请您分享一下自己的开发经历，您最初是怎么走上数据库开发这个方向的?

关胜亮：我本科就读于中国科学技术大学，研究生就读于中科院计算，学的都是计算机相关的专业。我参加工作之后就一直在做时序数据库方面的工作，只不过那个时候还没有时序数据库，叫实时数据库，只能存储当前时间的数据，历史数据都是存在其他系统里面。这样一直做了很多年。

后来，包括像 Memcache、Redis 这样的开源软件出现之后，我开始觉得做实时数据库没有什么前景了。后来，得知我的科大校友，涛思数据的创始人陶建辉要开始做一个实时、历史统一的时序数据库。我仔细了解之后发现，自己在时序数据，包括分布式方面的一些知识储备，在大数据领域的一些业务储备，正好跟这样一个产品是匹配的。而且，陶总过往的创业经历和工作激情也让我相信，他能够带领大家开发出一个行业领先的数据库产品。就这样，我就开始了 TDengine 的研发，我十几年的知识积累，也就能够发挥出更大的作用。

我还在写代码，写代码使我快乐

InfoQ：您已经做到联合创始人这个位置，您现在还写代码吗？

关胜亮：对，还在写。2017 年，我加入了涛思数据，是 TDengine 的主要开发者之一。我主导了 1.0、2.0 到 3.0 这三个版本的所有迭代。今年我们做了新的版本，我写了很多很多代码，写代码使我快乐。

相比于管理者，我更加喜欢开发者这样一个角色。每年，我都会在 GitHub 上面深度参与代码开发。今年因为我们发布 3.0，所以我写的代码尤其多。从今年 1024 程序员节公布的数据中可以看到，我的开源贡献全球排名前 50，国内排名第七。

InfoQ：请您从自身经历出发聊一聊，作为一个数据库工程师，日常每天的工作大概是怎么样的？有没有一些工作是比较程式化的，每天都要做的，有没有一些工作是非常有挑战的，比如说需要专门拿出一天之中精神最饱满的一个时间来做？

关胜亮：每天上班第一件事肯定就是看看自己有没有新的问题，尤其是一些 Bug 或者是线上的问题。任何时候，Bug 处理肯定都是第一位的。我们可以因为 Bug，把任何其他低优先级的工作都推迟。

第二就是要定期跟用户进行交流，去修正产品的发展方向。最好就在日历里面记录下来，大概什么时间约了一些什么样的人，跟大家一起交流产品该往哪个方向走，也包括在一些合适的场合，宣传产品的设计理念和应用场景。

第三个也是工作量比较大的，就是新功能开发。其实开发一个全新的功能难度是挺大的，也是非常有挑战性的。很多人都能够维护代码，包括一些大厂的员工，他们可能

维护代码的能力都很强，但真正的从 0 到 1 去做一个产品的时候，就不一定行了。很有可能刚开始要的是 a，最后做成 b。我们的策略一直是以终为始，也就是现在提到的倒推工作法。我们都会先起草新功能的宣传稿，这还不是用户手册，主要内容是怎么跟其他的部门、可能的用户去宣传它。之后就可以认识到，这个产品做起来是有价值的，再开始去起草用户手册，然后再去进行原型开发。最后才是常规的设计开发和测试流程。这种流程可能看起来会浪费时间，但实际上，这种做法能最大限度的做到不返工。并且，能把问题责任局限在一个很小的范围之内。

因此，总结来看，我每天大部分的工作可能都是在新功能的开发上面，少部分的工作在 Bug 处理和用户交流上面。但如果你要是一个领导者的话，你可能还有很多的会议、评审工作。

如何从初级数据库开发者进阶到高级数据库开发者

硬实力和软实力同样重要

InfoQ：经过这么多年，您已经进阶到一个比较高级的开发者角色。相信很多开发者都有这样一个疑问，就是在自己工作了几年之后，能不能够顺利地从一个初级开发者进阶成为一个高级开发者？在这个过程中，您是怎么做到的？您中间做了哪些努力？

关胜亮：我是 2009 年 7 月份参加的工作，当时到当年的春节我就成为一个小组长了，

也算是从这个问题里面的普通工程师变成了资深工程师了。很显然，技术水平确实是非常关键的一环。但是，竞争者的技术水平也都差不太多，尤其技术这个东西，没有说谁就一定强过谁。这个时候，觉得比较重要的就是一些软实力了。

在我看来，当时比较强的软实力有几个部分。第一是责任心，也就是交代的工作都能够按照一定的工期、一定的质量来完成。尤其值得一提的一点是，能承担一些边界比较模糊的工作或者说脏活累活。第二是业务能力，也就是我们要能够理解客户的需求，或者是说，如果做的不是项目的话，要能够理解产品的主要应用场景。我记得，我开始也只是负责开发一个模块，但是，会尽可能利用任何可能的机会去掌握整个产品的脉络。有机会的时候，立刻就能够补充空缺。最后是文档能力，很多开发人员对此不是特别重视。但是，不管是晋升评级的文档或者日常总结的文档，或者产品相关的文档，都要用心对待，因为文档体现了一个人的逻辑、思路和他对待工作的态度。我觉得大家要做的就是，要么不写，要写就一定好好写。所以我觉得，大家从这三个非技术能力的方面去严格要求自己，自然而然就会变成一名资深工程师。

InfoQ：您刚才提到脏活累活要愿意去干，在开发工作中，哪些算是脏活累活？

关胜亮：其实比较简单，例如开发的时候，可能客户会遇到一些 Bug。刚开始的时候，大家都不知道这个 Bug 到底应该由谁来处理，甚至可能都不知道它属于哪个模块。这个时候，很显然，问题抛出来了，却没有任何归属。如果你能够主动去做这样的一些事情，你就会成为这个组里面或者这个产品里面比较重要的角色。当所有人都依赖于你的时候，你的地位，你的经验，就能够更好的发挥出来，大家就更信任你。

优秀数据库工程师必备技能

InfoQ：现在，很多人都希望自己变得更优秀，也有很多程序员在工作之外想学习各种技能、工具、课程等提升自己。但现在知识特别多，可能让人有点眼花缭乱，您能不能给大家分享一下自己的经验？要成为一个优秀的数据库开发者需要掌握哪些技能，哪些是最核心的，重点学习哪些技术和工具就可以了？

关胜亮：这要看开发人员的水平，如果是半路出家，也不是计算机专业的，要学很多东西。如果只是培训机构出来的，那么可能要学的东西就更多。这个时候，想成为一个数据库开发工程师基本是不可能的，你要积累很多的经验和经历才行。

一个正常水平的开发工程师，需要学哪些东西？我觉得开发语言并不是最重要的，因为用各种语言都能够开发数据库，像 C/C++、Go、Rust 等。我觉得主要是要掌握各种数据结构。第一要掌握各种数据结构的使用，最好能够自己去编写一些简单的，基础的库。像哈希、树、图、链表等，也建议你自己去写，而不光是去用。当你写完这些东西之后，自己的水平会得到很大的提高。另外，比较困难的一些数据结构，像红黑树、B+树、LSM 树等等，一定要很清楚概念，并且至少写一个简单的算法，这时，你的基本能力就基本上够用了，编码能力基本上也够用了。

然后要掌握一些网络通信和多线程编程的使用方式。现在，各种语言都比较强调异步调用和异步 I/O。这方面的理念必须要掌握清楚，尤其是要知道各种不同的异步调用之间到底有哪些共享的数据，如何去处理共享数据之间的锁的问题，这个特别考验水平，也只能在工作中才能够摸索的到。

第三，我觉得需要掌握一定的分布式知识，因为单机软件现在越来越少，大多数软件都需要多台机器协同，像 Raft、Paxos 这样常规的分布式算法，需要了解。也可以看

一看 ZooKeeper、etcd 的实现原理，掌握一些基本的概念。

第四，如果有时间，最好是看一些开源软件的代码，简单的像 Nginx、Redis，功能复杂一点的，像 SQLite。掌握这些，我觉得基本能力就够了。

之后，可以试着给一些数据库做 Demo 插件，比如说像 MySQL、Postgre 都支持插件。这些插件不一定要得到多么大的应用，能够做出来就可以了。

最后，可以回过头去看一些数据库书籍，掌握一些理论知识。再然后，就应该想办法在实际的工作中去应用它，你不用它最后你掌握的东西都忘了。可以写一些独立的数据存储的组件，因为不管做什么样的工作，都可能会涉及到一些简单的临时文件，临时数据存储等，可以写一下这些东西。能写这些东西就差不多了，就可以找一些国内的数据库公司去试一试，看看面试能不能通过。

我觉得大体应该是这样的一个过程。

如何让自己不可替代

InfoQ：开发者之间的竞争压力挺大的。关于怎么构建自己的技术壁垒，在技术能力上让你在这个公司不可替代。在这方面，您觉得应该怎么样去努力？

关胜亮：对于一个公司，它可能会有一些跟其他公司不同的地方，对于个人当然也会有。公司间的壁垒，或者产品间的壁垒，很容易从定位或者是产品架构上来实现。

对于个人来讲，其实很难说我有哪些东西，而别人没有。因为这么多的竞争者，没有哪些技术是只有你会别人不会的。所以，如果想在工作上更有区分度，或者在公司中

有更加重要的一个位置，还是从技术和业务两个方面去着手。

这里说的技术不是通用技术，也不是说写几篇技术博客就可以了。这里的技术主要是算法跟具体业务的结合。然后在业务上，要掌握整个产品的应用场景。这些东西，就像打球一样。比如说，我喜欢看 NBA，有很多好的球星看起来天赋很高，但其实背后都是一遍一遍地苦练。在公司也是一样，你做的东西多了，接触的客户多了之后，自然而然的就能够达到一个相对重要的地位。事实上，要想通过走捷径去获得这样的一个位置，几乎是不可能的。没有人可以说我什么都不干，然后还可以位置无忧。

数据库工程师如何做职业规划？

职业发展路径

InfoQ：很多开发者都对职业规划有不少疑问。数据工程师应该怎么去做好自己的职业规划？一个比较好的职业路径应该是什么样子的？

关胜亮：我觉得，首先，数据库工程师的入门难度比较大。所以，当你成为一名资深的工程师之后，上面提到的竞争力基本就形成了。

之后，我觉得可以从四个方面去考虑职业发展的规划。第一个就是继续做技术，成为技术专家。但是，成为技术专家不光要有深度，还要有广度，要扩大自己的知识面，尤其要抽时间去看竞品，看市场上主流技术的发展。其实这方面难度比较大。但如果你是一个比较宅的人，这比较适合你。

第二就是想办法首先提升为组长，然后争取成为部门主管。这样的话，可以走一个技术管理的岗位。

除了技术相关的，也可以向解决方案架构师转变。凭借你早期积累的对于产品的理解，还有技术能力，就可以更好地为客户提供技术咨询。因为现在，技术咨询这个领域对人员水平的要求越来越高了。以前，都是一些技术比较差的人去做，现在，做这种解决方案的咨询，都需要很资深的员工。

也可以向销售去转变。最近五年，也包括疫情这段时间，技术销售越来越吃香了，之前吃饭喝酒这种方式已经 out 了。大家觉得，你技术比较牛，那我就能相信你。

关于职业发展，我觉得可以从这四个方向去做。当然，如果你能力更出众，做成高管，那就是另外的路径了。

要不要转管理？

InfoQ：您提到了好几种转型路径。下一个问题就重点谈谈管理吧。如果你一开始做技术，后来做管理，这个跨度还是挺大的，很多人就想知道，我到底啥时候转，我自己适不适合转，我转过去之后万一不适应，发现我不适合做管理怎么办？您有没有一些时间点建议，以及怎么判断自己适不适合转？

关胜亮：管理能力和技术能力其实是完全不同的能力，也是不同的职业规划。我刚参加工作的时候，还比较年轻，大家说 35 岁就不能写程序了。后来等我到 35 岁的时候，大家又说 45 岁不能写程序了。我觉得，这些说法都不是很准确，因为实际上，走管理和走技术这两种路径，从收入来看的话，差别也不是很大。

大家都说到了一定年龄之后就无法做技术，似乎就必须得转型管理，转型可以，但并不一定要转管理，也可以转型到很多其他的岗位。而且现在的技术岗，如果你做的是一些底层技术的话，年龄越大可能会越受到重视。而如果说你做的是 UI 类的软件，那你挣这个钱主要是靠时间来积累。当你年龄大了之后，你没有那么多时间的时候，你就必须要转型。如果你技术挖得比较深，你的竞争力是你的技术能力，而不是工作时间，不是靠加班来拼的时候，那么你只会做的越来越好。

其实到了一定年龄之后，如果你在自己的岗位上不是特别资深，特别有竞争力的话，那其实管理岗也好，研发岗也好，都会被淘汰，而且管理岗的缺口可能更少。像我们公司，其实就有很多大龄的程序员，很多之前做过管理，都至少带过十几二十人的团队，还有更多的。后来，他们觉得还是回归研发比较好。所以说，不是说你自己适不适合去做研发，不是说做不好研发，就去做管理，肯定不是这样的。其实，大多数不想在技术方面深入的人，大多不是转到管理岗，而可能转到像售前、交付、运维、市场这样的一些非研发岗位。

所以我觉得，判断自己适不适合做技术管理，就是问问自己有没有这样的意愿、能力，或者某一天能够去一个创业公司，能够独立带队完成一个新产品从 0 到 1 的开发。如果是你判断自己在未来某一时间段内能够有这个能力，那么我觉得你可能还适合做技术管理，如果不能的话，就要考虑自己真正的发展方向在哪里，做好自己的职业规划。技术管理的难度很大，大家不要觉得它是一个很容易的事。

InfoQ：管理能力的提升有哪些快速通道?

关胜亮：首先是你到底适不适合管理岗。如果你目前不在这样的岗位上，那么首先，你先制造一个这样的机会，比如说我想要做一个什么样的新特性。而这样一个新特性

又不是自己一个人能够完成的。显然，开发人员分前端、后端以及存储，很多不同的分工。做这样一个小特性时，需要很多人一起配合。这时，你就可以尝试在很小的一个范围之内，让大家围绕你把这件事做好。这就是你自己练手的一个好机会。做不好也没关系，接着回去做独立的模块就是了。做的好的话，就说明你具备了组织大家一起去完成一件事的能力。

然后，如果有机会，就参与一些比较重要的功能的开发。如果你还能够组织一些人把这个事情完成，那么恭喜你，基本上你就能够得到领导或公司的信任了。再有机会的时候，你可能就能够提升为管理人员了。

如果说你已经是管理岗，想提升自己的管理能力，我觉得主要就是从工期和质量两个方面去思考。如何能够更好的评估一件事情，什么时间能够完成。如何能够按照一定的标准去完成。这两个方面是管理里面非常重要的核心指标。当然，这个指标只是一种非客观的评价。

提升管理能力，一方面就是实战，一方面就是多看看书，多听听一些演讲，或者是参与一些培训班。我记得，我刚开始参加一个培训班的时候，我听到那个培训老师讲的，就跟被洗脑一样，我突然觉得，哇，管理还可以这么去搞。过了两天，我回来之后发现，哎，这个老师说的怎么感觉完全实现不了呀。我就对这个老师讲的东西产生了一些疑问。后来，随着经验越来越丰富，我发现，老师讲的是一些理论，他找了一个很好的例子，来证明他的理论能够执行。而如果你把这个理论，原封不动的放到自己的公司里去，可能就会执行不下去。这个其实就是理论和实际相结合，但你要是一点理论都没有，确实也没有办法做好管理。

如何平衡管理和一线技术工作

InfoQ：作为一个技术 leader 的话，您自己在管理工作和一线技术工作这两个方面，是怎么平衡的？

关胜亮：技术管理和一般管理不同。在我看来，技术管理的难度比较大。不像我们砌一堵墙，今天你砌 1 米，明天我砌 2 米，然后拼在一起。技术管理的本质是要把一项任务拆分为多个子任务，然后把这些子任务安排到合适的人，并在规定的时间内完成交付，进行有效的检查。这里面比较难的是任务拆分，估算工期也很难，质量检查也很难。这些都需要 leader 具有很强的技术能力。

大家会看到，没有人会跨技术领域去做管理。本来做后端，现在去管手机 APP，这个确实难度比较大。所以，要想具备较强的技术能力，就需要不断参与一线的技术工作。

作为 leader，我通常会参与一些需要新技术的功能模块的开发，在完成原型开发或者第一版交付之后，再转给其他同事。通过这种方式，我们就可以在新员工中得到一定的威信。另外，自己也熟悉了新技术。还能避免新技术引入对整个研发工作造成太大的风险。所以作为一个技术 leader，一定要找到一些可能性，让自己去学习新技术。这样才能够不断提升自己的能力。

数据工程师有“钱途”吗？

InfoQ：做数据工程师有钱途吗？前几天，我看到有家机构发了一份程序员收入报告。您能不能分享下，数据库工程师的收入到底有多大竞争力呢？

关胜亮：数据库工程师的收入其实还是挺高的，因为数据库也是属于后端研发的一种。任何做后端研发的人，都有可能转化为数据库工程师。数据库开发不一定非得去做 SQL 相关的，它里面也需要很多像分布式、网络、存储这样一些底层的知识。

我们招人，也不是只招数据库这个领域的，也会招一些其他的后端工程师。我们看到，现在去一些招聘网站会发现，后端工程师收入普遍比前端工程师高。而数据库工程师又往往会比普通的后端工程师技术能力更强，所以他的收入更高。不同的公司可能给的钱不一样，但基本上都属于这个公司里面排在前列的岗位。

InfoQ：您在招人的时候，会比较看重哪些方面的能力？比如说有一位求职者，他哪些方面比较强，对于您来说或者对于涛思数据来说是加分项。

关胜亮：我们招人的时候，技术能力当然很重要，因为技术能力是技术型公司所必须的。但是，就我个人来讲，技术方面的能力，由各个用人小组就负责了。比如说哪个小组需要用什么人，他们自然而然就会评估这个人的水平，所以我的考察一般都是基于个人素质方面。一般都是考察一个人在某一件事情上面，曾经到达过那种高度。

我曾经跟 HR 说，给我面试的应届生，我只要 985 的毕业生，其他的不要推给我，就是认为他们既然在学习这种事情上面能够达到前 1/100 的高度和水平，那还是比较厉害的。但如果他不是 985 毕业的，那么我们要招他的时候，他多半已经参加工作好几年了，这时我就会考察他具体参与过哪些项目，或者在哪项技术上面做的比较好，到底好在什么地方。

我为什么要考察这两点？我希望他是有一些成功经验的。对于有成功经验的人，我需要做的事情就很明确了，就是要让这些员工把自己成功的经验复制出来。通过各种方

式激励他，让他投入更多的精力，让他把成功的经验发挥出来，因为你成功过。只要你成功过，无论如何，你总会有一些不同的地方，比如你非常聪明，或者工作非常努力，总会有一些成绩，一些东西是和别人不一样的。所以我一般都会关注求职者的成功经历。

InfoQ：在找工作的时候，是去大厂还是去小厂，您是怎么看的?

关胜亮：去大厂还是小厂，也要看自己做哪些事情。去大厂有个好处，你可能会觉得虽然你没有做到核心开发，这很正常，因为刚开始一两年不太容易做到，但你有很多学习的机会，很多代码仓库，你都有权限去看。这时，你就能够学习很多知识，但这其实是针对大概五年前、十年前而言，因为那个时候开源软件还比较少。你要想学习这些东西，只能去大厂。

现在，这种开源软件越来越多，我们可以通过各种不同的渠道去学习知识，这时，如果你进大厂，还没有比较核心的岗位，可能就不一定好了。如果你真是技术能力比较强，那我建议去一些相对比较小的公司，因为比较小的公司里，你的竞争对手也会相对弱一些。这时，你就很容易出类拔萃。如果你喜欢做管理岗，或者喜欢做核心技术攻关，可能都能够轮得到你，你的成长就会非常快，因为成长还是一定要通过实战来完成。去了大厂你能够学到东西，但不一定有机会去做。

数据库行业发展现状与展望

开源数据库不会冲击商业数据库市场

InfoQ：现在开源数据库越来越多了，会不会冲击商业数据库的市场呢？

关胜亮：这倒不会。开源和商业并不矛盾，完全可以把它们合二为一，我们公司主推的就是开源商业模式。但开源不是免费，开源也是有偿的服务，你可以用我们的软件，但是当需要我们提供服务的时候，我们还是能够获取一定的报酬的。

简单来讲，当你的公司具备一定的技术能力，并愿意在这方面投入自己的维护成本，那你用开源软件就可以了，出了问题你自己可以解决它。那么还有一些公司，尤其是像一些并不以技术特长为主的，比如一些传统企业，还有一些小一点的技术公司，他们虽然技术能力比较强，但他们希望把技术放在真正的业务上面，不希望在基础软件领域投入太多，这时候，他们就会购买开源软件的服务。我们公司的盈利模式也是这样。我们给一些真正有需要的公司提供服务，给一些技术能力比较强的公司提供更多的选择，让他们自己免费使用。同时，这也可让越来越多的用户使用我们的软件。

我们知道，数据库不是设计出来的，而是用出来的。你没有那么多的场景，就没有办法发现那么多的问题。当时有很多人质疑我们说，TDengine 为什么迭代速度这么快，我们两周甚至一周就发一个版本。当时我就说，那是因为有很多用户，他们提出了很多问题，所以我们才能有这么快的迭代速度，才能有这么多的新功能。如果真的都没有人用的话，那么你的软件永远都是稳定的，一个不运行的软件它怎么可能有 Bug。

TDengine 既有开源的部分，也有商业的部分。如果是一个纯粹商业数据库，它就没

有免费版本，它们的生存空间已经越来越小了。尤其是一些新型的数据库，它们没有办法竞争过 Oracle、Postgre、MySQL 这样一些成名的数据库。它们也没有办法去跟一些开源数据库竞争，因此，它们的生存空间会越来越小。

这也是最近几年，开源数据软件蓬勃发展的一个主要原因。像时序数据库、图数据库、内存数据库这样的数据库，现在蓬勃发展起来了。像工业制造业的发展过程一样，大数据架构也会细分为越来越多的零部件，这个领域里也产生了盈利能力很强的公司。但是进一步去看就会发现，这样一些盈利能力很强的公司，他们最核心的技术都是开源的。所以我们说，纯商业收费库是没有发展前景的，我们一边开源，一边做商业，这样的公司是符合技术发展潮流的。

InfoQ：数据库上云对数据库行业的冲击大不大?

关胜亮：其实不是很大。MySQL 这个公司很知名，大家买云上的 MySQL 也花了不少钱，但是实际上，这个钱都被云厂商挣走了，MySQL 公司本身并没有挣到太多的钱。

随着这种方式的发展，大家发现，如果我们做开源，最后自己都没有拿到更多的收益，这肯定不合适，所以现在大家提供了一些新的开源协议，包括我们公司，这并不是为了阻止大家免费使用，而是我们不希望自己的产品被公有云软件公司拿去部署，而我们只是给人做嫁衣。有了这样一些协议的保护之后，上不上云已经对我们本身的发展，影响就不是太大了。

上云的理念在这些年，尤其是近几年也确实是一个蓬勃发展的状态。一般来讲，很多应用方都会主动把大数据平台上云，有的是公有云，有的是私有云或者混合云。而且，

越来越多的数据库公司，像国外的比较厉害的 Snowflake、Databricks，我们发现，上云使他们的产品得到了更好的推广，因为很早以前，软件部署非常困难，需要耗费很多人力。通过上云，就可以扩大这些数据产品的应用范围，甚至创造出新的价值。

所以我的结论应该是影响不大，可能还有很多好处，因为我们公司的产品现在也在上云。我们也跟一些云厂商进行合作，比如华为，在他们的云市场上就可以购买到我们的产品。上云，如果我们把握好这个机遇，对于数据库有非常好的促进作用。

2023 年，数据库最值得关注的技术趋势

InfoQ：新的一年开始了，这几年数据库也挺火的。您能否展望一下今年或者说未来三到五年，数据库的哪些技术点值得大家关注，或者说有比较大的发展潜力?

关胜亮：我觉得潜力还是在一些比较专业的数据库领域。现在来看，通用技术方案越来越难以解决一些专门的技术问题。我们要从技术的角度去寻找自己的赛道。

我们看到，顶级的开源产品，像国外的 Kafka、Mongo，国内的像 TiDB，包括我们的 TDengine，有很多拥有上万 Star、上百贡献者的软件，都把精力放在了自己比较擅长的领域，然后让其他厂商去做数据集成。因此，我觉得，一定要说哪一个技术方向一定会火，这不好判断。但是，这种技术型的公司，尤其是一些有硬核技术的公司，它永远都会活下去。可能你这一家公司活不下去，但是另外一家公司也会活下去。

我们始终相信，硬核技术永远都不会过时，而且只有技术能带来新的元素。因此我也希望，大家如果要找工作，还是要找一些有硬核技术的公司。基本上，这些公司，因为各种原因，可能不会 100%全部都活下来。但是，对于个人发展来讲，它是一种很

好的技术沉淀。

InfoQ：您谈到，很多技术方向都有很大的发展潜力。对于这个行业的从业者，对于数据库工程师来说，他们应该怎样去抓住这些机会？有没有可能，他们可以根据这些技术趋势去提升自己的技能，然后做好准备？

关胜亮：我觉得，很多年轻人都有一种顾虑，就是觉得自己怀才不遇。他们会一边工作，还一边去做一些跟工作本身没有关系的其他准备工作。比如我现在正在开发其他软件，然后去想，什么时候能成为一名数据库工程师。我觉得，大可不必这样做。

我一直都认为，真正的能力都是实战中锻炼出来的。回顾自己成长最快的时候，都是加班最多的时候。我们会发现，积累的并不是技术能力，而是做事情的方法。我觉得，最重要的是，在实际工作中看一些技术书籍、管理书籍，然后去思考怎么样把一件事情做好，掌握把事情做好的方法，并在实践中不断验证和修正。把事情做好的方法才是真正的工作经验。我不建议大家去学习某一些新的方向，除非你现在没有工作，处于待业阶段。

国产数据库如何加速？

InfoQ：这两年，因为一些原因，我们国家在科技方面，尤其是在基础软件方面，国产化的呼声特别高。就数据库来说，在走向全球、走向一流方面，国内似乎没有国外做得那么好？关于这个问题，您觉得应该怎么样解决？

关胜亮：其实我们现在做的也不见得比国外差很多，像现在国内比较流行的一些软件做的都不错。我也会参与一些开源软件活动，他们之所以没有在国外打出名气，主要

是因为国外很多时候也和我们一样。我们都喜欢用本土化的，像之前去 IOE，我们不喜欢用国外的软件。这给类似我们这样的数据库软件提供了很多土壤。国外也是这样，他们一看这个软件不是他们国内本土开发的，也有这样的效果。

我记得，前几年互联网比较火的时候，大家都做互联网，做的差不多了之后，大家就开始希望用互联网的思维去改造传统的工业领域。但是这种改造，说白了，并没有一些硬核技术，都还是用同一种互联网思维去改造工业，并没有产生新的价值和新的业务场景，重写一遍业务系统的意义并不是很大。这时基本上就宣告了，用这种互联网的思路解决不了问题。

所以，经过这两三年的沉淀，我们会发现，越来越多的企业、创业团队，包括一些资本，把资金投入到了小而精的专业数据库上。这些小而精的专业数据库发展的非常好。因为我们国内是一个非常大的市场，什么样的场景都有。

我觉得，再经过三年左右，国内还会涌现出更多的类似这样的数据库软件。像我们 TDengine，可能也会在国外拥有一席之地。

InfoQ：您刚才提到在国外有一席之地，那咱们现在是不是有出海计划了？

关胜亮：对，我们已经在美国组建了销售团队，也招了不少人。预计明年初就会在国外推广。但到底能不能成功，这个不好说。

我觉得，做基础软件一定要耐得住寂寞，不要盲目地扩大，也不能盲目地悲观。如果这次在国外没成功，那最重要的是复盘，看看我们到底还差在哪儿，去做一定的改造。出海不成不一定失败，重要的是得到很多经验教训。我们看到，去美国难度会比较大，但是去东南亚，包括台湾和日本，国内的很多软件竞争力都非常强。所以从这个角度

来讲，我们如果能够真正知道美国到底要什么，我们的基础软件肯定会越来越好。

InfoQ：涛思对数据库未来发展的战略，既有本土化战略，也有出海规划。从这个角度看，是不是说国内数据库的发展，未来必须要走出海这个路径？

关胜亮：目前来看，绝大多数做的不错的技术团队，都不是由一些大公司孵化的。因为我们知道，孵化这样一个团队，收益相对会比较低，尤其是刚开始的创始团队，收入会比较低，获得的股份会比较少。每个人，做自己的事情和做公司的事情，投入肯定是不一样的，战斗力也不一样。所以我们看到，这样的公司恰好都是独立但有资本支持的一些公司。有资本支持的公司又希望投入收益能最大化，那么怎么才能够收益最大化？如果仅仅在国内，虽然国内市场也挺大，但在国内市场，资源很重要。谁拥有了资源，谁就能够获得市场。

相反，对于技术型市场的开拓还不是特别多。虽然近些年来，在国内，我们也凭借技术获得了很多客户。但是，也有很多客户，尤其是一些比较大的客户，他们并不关注你的技术怎么样，他们关心的是你的资源网怎么样。我们要想提升知名度，获得更多的社会认可，包括更高的估值，我们必然就希望到国外去做一些工作。

在国外成功占据一席之地之后，在国内也就更好推动。比如说，我们去做一个数据库软件的投标。在业务选型里面写上 Oracle 和 MySQL，大家都不会有任何问题，因为这都是大家公认的业界标准。但是如果你写上了一个不知名的小数据库，那么大家就会怀疑，这里面是不是有什么套路啊？也就是说，如果能够成为事实标准，就能够给公司的发展带来一个非常好的直接的推动作用。我们希望先从国内走出去，再从国外走回来。通过这种迂回的方式，能够把一个公司真正做大做强，并且能够有更多的业务场景，使产品的稳定性、易用性都变得更好。

InfoQ：国产数据库要想取得突破，一个关键的问题是不是就在于人才？

关胜亮：我觉得，一个软件的成功失败不在于个人能力，最主要的是组织能力。刚开始三五个人的时候，就可以做一个东西出来。但是，如果你投入的只是这样一个团队，做出来的东西，势必就不会有那么大的价值。换句话说，它的功能不会特别全。我们就必不可少的要扩展研发规模，包括其他团队的规模。这时，组织能力就凸显出来了。

我觉得，国内软件行业跟国外软件行业最大的区别还是组织能力。我们看到，我们的员工跟国外的一些顶级研发人员比起来，个人能力差别并不大。那么为什么国外的知名公司，他做出来的东西总是很好，很稳定呢，他有什么秘诀呢？其实就是他的管理能力比较好。

我现在比较喜欢把这个管理能力称为组织能力，就是组织一些人完成一件事的能力，我们比较欠缺这样的能力，而并不在于说我们有什么样的人才。如果一定要说人才的话，就是具有组织能力的人才。

当然，事实上，对于组织能力，并不是单单一个人到了某一个公司之后，就一定能够改变整个公司的气氛。甚至说，当你所推动的制度跟其他人的理念不一样时，你可能就待不了太久了。所以，我觉得，从组织上面、从管理制度上面去学习国外，可能还真是一个比较好的前进方法。我们公司现在也是慢慢向这个方面去转型，去了解国外的一些成功的经验是什么样的。

InfoQ：您刚才谈到了很多关于数据库发展的问题，比如资源，可能大厂在这方面比较有优势。那您觉得，在国产数据库发展方面，创业公司要想获得好的发展，他自己需要在哪些方面做出努力？在整个形势也不是特别热的情况下，这些创业公司还

是要专注于做技术吗?

关胜亮：对，越是到寒冬，大家更是要步子迈的小一点，因为实际上现在有一些对赌等等，把一些本来有很多好思路的公司都搞得最后走向了一个不好的状态。如果你没有一个创业者在前面做领头，虽说资本把公司拿去了，但它最后也发展不了。为了保住公司，让它更好的度过寒冬，如果公司剩了不到半年的钱了，你该怎么办呢?

我觉得还是要在技术上去做，而不是去做太多市场方面的东西。关于市场，我们从两方面去理解。一方面是做一些市场活动，比如到很多地方去做广告、做宣传。这势必会有一些投入，这种投入要降低。

还有一种是为了能够打开市场，获得越来越多的客户，我们需要有很多的销售团队。但是，扩大销售团队如果不能获得高收益的话，投入可能就会损失掉。而且，即便你真正拿到很多合同，但是，那很有可能跟你这个公司本来的定位是不一样的。为了完成这样的合同，你可能会把公司里面比较核心的开发能力，投向了容易拿到市场合同的技术方向上。最后，从一个技术型的公司变成一个项目型的公司，处于一种只是勉强活着的状态。当寒冬过去之后，你就会发现，自己所赖以生存的技术已经被其他公司远远抛在后面了。这时，等待你的最终也还是同样的结局，只不过就是死的晚了一点而已。

所以我觉得，不管怎么样，还是要把握住自己做事的初心。希望做成什么样的产品，希望做成什么样的盈利模式，要秉持住这样的想法，不要被外界的诱惑所吸引。

同侪寄语：穿越寒冬

InfoQ：2023 年刚开始，新的一年，您能为数据库工程师这个群体送上一句祝福的话吗?

关胜亮：我希望大家都能够熬过这个寒冬，在疫情过去的时候，在 2023 年，都能够得到一份满意的工作，获得一份满意的收入。

InfoQ：很多人都在做自我提升，尤其是在寒冬的时候，大家觉得，修炼好自己的内功，才能够让自己具备穿越周期的能力。自我提升很重要的一个渠道就是看书，您能不能给大家推荐三本数据库开发相关的书籍?

关胜亮：我也就看了十来本。我觉得，比较好的一本是《数据库系统概念》，这是一个比较经典的数据库系统教科书。刚入门的话，可以看一看这本。然后可以看一看《Redis 的设计和实现》。它比较全面的描述了 Redis 内部的运行机制。如果你的经验还不算丰富，则可以通过这本书了解目前业界是怎么解决这方面的问题的。第三本是稍微进阶一点的，就是《数据库系统内幕》。它主要介绍了存储引擎并发控制的一些核心技术。这本书讲的稍微乱一点。它最后面的两三章里面讲到了 B+树和 LSM，这一块可能会对你产生一些启发作用。现在所有的关系库，包括 NoSQL 的库，包括 TDengine，可能都用到了这些技术。这应该是必备的一种技术要求，大家可以看一看。

从 JS 到全生态，云原生时代下的前端成长演进之路 | 展望前端工程师的 2023

作者：褚杏娟、Tina

前几年，我们开玩笑说“前端一天一个新框架”、“前端难度 18 个月翻一番”，虽然现在这个领域经过蓬勃发展已逐渐进入成熟期，但新一代的技术也正在悄然孕育与布局。那么 2022 年，前端领域发生了哪些值得关注的事情？未来前端开发需要具备哪些技能？哪些新兴趋势需要我们掌握？

1 月 11 日，我们邀请到了字节跳动云原生 PaaS 资深前端工程师黄健，一起展望前端研发工程师的 2023。本文经过编辑处理，完整原版视频可参看：https://www.infoq.cn/video/HvDdiQutbkVeQQ3B8ysT

嘉宾简介

黄健，字节跳动云原生 PaaS 资深前端工程师。目前就职于北京字节跳动有限公司，是公司较早期的研发工程师，见证了字节跳动云原生发展的从 0 到 1，主攻方向为：大规模高性能 Web 应用，云原生服务平台架构和 Serverless 函数计算，当下所负责的云原生团队分布于北京、上海、杭州、成都等多城市，致力于打造业界领先的 PaaS 平台，提供以应用为中心的服务平台，围绕代码的产物，变更，发布再到运行，编排等提供全面的系统能力和解决方案。除专注于技术研发外，在工程师培养、中大型团队管理、远程跨城市协作、敏捷项目管理等方面具备丰富经验。

第一部分：前端工程师与云原生

InfoQ：可以给 InfoQ 的观众们介绍下你自己吗?

黄健：InfoQ 的朋友们，大家好，我叫黄健，目前就职于字节跳动的架构前端团队，2023 年是我毕业后作为专业前端工作者的第十四个年头。首先要感慨下时光如梭，这些年我非常有幸地亲历 PC 时代、移动互联时代再到今天的 AI/大数据三个大时代，也完整见证了前端技术的高速发展，当然自己作为 InfoQ 的老用户，今天也很荣幸能从一个读者走到台前，和大家一起分享这个前端展望的直播主题。

InfoQ：可以介绍下你在字节跳动负责的具体工作吗?

黄健：我现在在字节跳动主要负责云原生应用平台方向的技术建设，我所在的团队全称叫：**泛 PaaS 前端研发团队**，其实从团队的描述中也可以看出来，我们的团队名称由两个关键词，一个是泛（广泛、通用的意思），另一个是 PaaS，这个属于云计算领域的一个专业术语，即平台即服务的缩写，具体涉及到的产品包含 Kubernetes 的基座、微服务框架、制品产物的构建/发布、函数计算等平台产品，这些基本属于服务于企业内部的研发设施，此外，我们也会尝试做一些对外的应用产品，比如大数据、消息队列等云原生下的商业应用产品。所以我对自己的定位是：一个在互联网时代下征战多年的前端老兵，同时也是云原生时代下的技术布道者和实践者，说点关于我自己有意思的事，虽然我的 base 地在北京，但因为我们的团队业务遍布国内四五个城市，所以我作为出差狂人，经常会在上海、杭州、成都、南京等城市闪现，欢迎大家可以随时约我进行线下交流。

InfoQ：从您的描述里出现了多次云原生这个比较时髦的词汇，那您结合您的专业背景，具体讲讲前端和云原生之间的关系吗？包括当下前端如何应用 Serverless？

黄健：大家知道，整个云计算的市场伴随着虚拟化和容器化等技术的成熟之后，迎来了它的第一个高峰期，在 2010 年年初时，我们谈论最多的一个词叫：**上云，即把传统的物理机房、服务器、数据库等都搬到云平台上**。这样做的好处是普通用户可以通过互联网按需获取各种 IT 资源，同时无需关注机房搭建、服务器硬件采购、软硬件的更新等，并且能做到随用随走，按需付费，对早期企业来说，成本优势还是非常明显的。

这是第一个阶段，即传统企业陆续上云。随之而来的问题是我们不在企业基础设施搭建之初就默认的跑在云上呢？即像原生一样，这可以更高效。理念虽然大家都认可，但是天然跑在云上和以传统形式从线下移到云上，无论是在架构选型、运维设计、服

务类型的部署等方面都会有非常大的差异。这时就提到了云原生的概念，它是云计算发展到一定阶段大家都比较认可的一种形式。

InfoQ：这听上去很有意思，可以展开再详细介绍下云原生吗?

黄健：云原生是一个组合词，由 Cloud 和 Native 组成，“云”表示应用程序运行于云环境中，“原生”表示应用程序在设计之初就需要充分考虑到了云平台的弹性和分布式特性，天生就是为云设计，并沉淀出最佳实践，所以我认为，云原生既是一种构建和运行应用程序的方法，也是一整体套技术体系和方法论，即：生在云上，长在云上，天生为云。

这个可能比较抽象。如果找个前端同学可以比较好理解的概念，那就是我们经常会接触到一个词，W3C，它会为我们前端的网页技术开发提供一整套标准的集合。那在云原生领域，也有一个相应的组织，CNCF（云原生计算基金会），它会对云原生的标准给出最权威的定义，同时也会围绕云原生下的技术发展给出最佳实践的指导，并且维护一个厂商中立的开源生态系统，来推广云原生技术，像大家经常听到的 Docker、K8s、gRPC、Prometheus 等都是这个生态下的作品。它的最终目的是通过将最前沿的模式开源化，让这些创新为大众所用，当然大家的最佳实践也可以开源化，反哺社区。

InfoQ：刚才聊了很多关于云原生概念的内容，那同时作为专业的前端工作者，您能具体讲讲前端和云原生之间的关系吗?

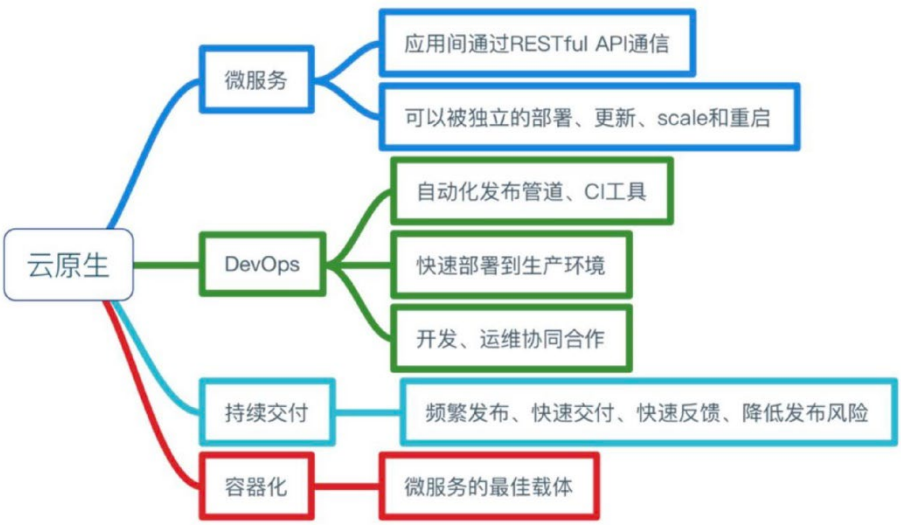
黄健：我觉得这一个非常好的问题，前几天掘金做了一份 2022 中国开发者生态报告，里面提到了多数开发者对云原生技术都有接触但了解较浅，每周投入时间在 1 小时以内，但是结合我自己的实际工作情况来看，最近几年前端有向大前端发展的趋势，这个“大”和“泛”相似，不仅会涉及本地化的页面级开发，也是涉及到很多服务器端或云

环境的基建。

前面也提到云原生的很多概念已经渗透到了我们的很多基建平台中，其实越是大企业封装往往也越好，真正的做到了润物细无声，这样就是我们经常说的：一个好的云计算平台应当是让你感受不到它的存在，但是又能真实地解决你研发链路上的各种问题。

InfoQ：可以给大家来举个例子吗?

黄健：前面我们有提到，云原生不单单是一个产品，而是一套技术体系和方法论的合集，就像我们经常把网页开发的内容概括为：结构（Structure）、表现（Presentation）和行为（Behavior）一样，云原生里也有四大核心要素，分别对应微服务、DevOps、持续交付、容器化，它们都会深刻地影响着我们的开发方式和平台基建。



众所周知，现在前端工程师所接触的技术领域已经和十年前大不相同了，我记得 10 多年前，我刚工作时，前端群里大家经常自嘲为“切图仔”，似乎大家的工作只是将 HTML 跑起来，在本地完成静态页面样式和交互脚本开发即可，剩下的工作就是把开发后的产物打个 zip 包发给我们的服务端同学后就宣告结束了，完全没有部署的概念。后来我们有了前后端分离的概念，前端同学慢慢地需要自己去完成前端构建产物的部署，这就需要大家了解构建工具，构建完成后不论是推到 CDN 上，还是作为内网平台直接挂到静态服务器下，都需要前端自己完成。

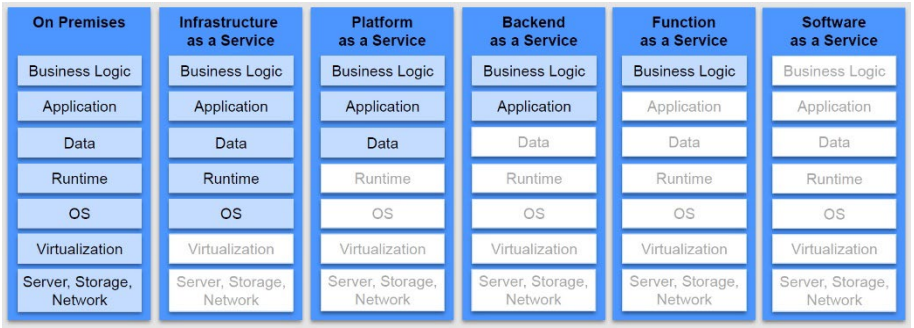
这时大家就会陆续地接触到很多基础设施如：Docker 容器、微服务平台、CI/CD 等，这些本质上都已经属于企业级基础设施了，这刚好就对应了和云原生的第一次联系，即我们作为前端工程师，需要和所有的研发工程师一样，熟练地去使用云原生下的基础设施，而且这些基础设施就是按着我们前面提到的四要素来搭建的。这里我推荐一篇我们团队的博客文章，里面详细地介绍了前端部署时可能会遇到的各种平台和细节思考。这是第一个阶段。

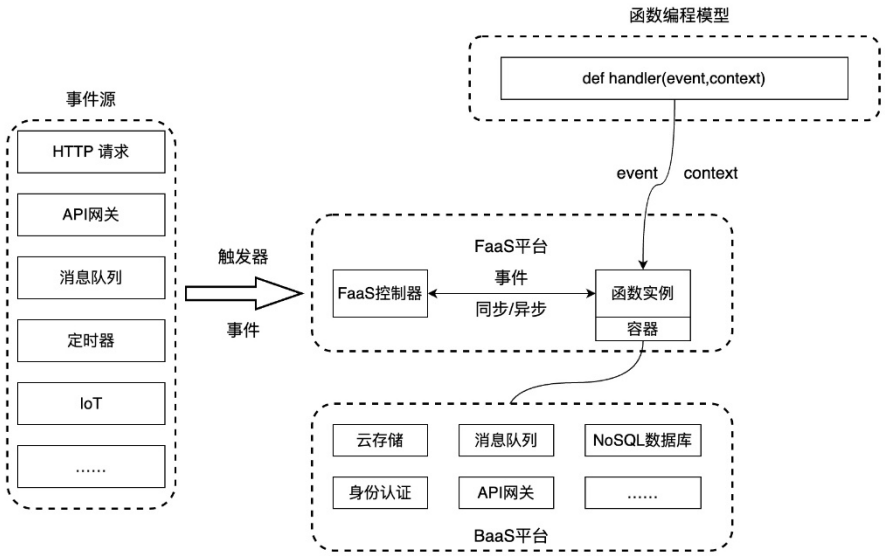
InfoQ：那还有第二阶段吗?

黄健：第二个阶段是伴随着 Node.js 技术的蓬勃发展，现在的前端已经慢慢地朝着大前端的概念演进，这里提到的“大”在我看来代表一些方向和领域，比如有些同学是朝着跨端方向发展，在写移动 H5 应用的同时也会接触到安卓和 iOS 等 Native 平台；还有些同学会朝着全栈化发展，就是尝试用 Node、Golang 等后端语言去包办前/后端产品的开发，这些同学从能力模型上看已经很靠近服务端了，所以服务端同学需要用到的工具平台，前端也都需要了解，比如：Node 的服务如何部署，应用的健康度观测如何做，服务流量上来后如何做扩缩容，线上的白屏化运维如何保障等，这时前端和云原生已经紧密融合了，这是第二个阶段。

InfoQ：相信很多前端的同学都有听过 Serverless 这个词，您在字节跳动负责也是对应平台的研发工作，可以向大家介绍下前端应该如何更好地使用 Serverless 吗?

黄健：这里我可以把它称作是前端和云原生结缘的第三个阶段，Serverless，全称 Serverless Computing（无服务器运算），一般的实现形式为 FaaS（Function-as-a-Service，函数即服务），当然也会包含 BaaS 能力（Backend-as-as-a-Service，后端即服务）、存储、消息队列、OSS 等，这是云计算的一种模型。在 Serverless 的概念下，基于事件驱动的全托管计算服务，计算资源会通过函数服务化，屏蔽掉运行环境与机器的概念。对于开发者而言，基于 Serverless 的应用部署可以在函数级别快速实现，而无需过多地关心服务器的问题，省略了部署、运维相关的细节和步骤，从而极大地提高了软件开发的迭代效率。





比较经典的产品像：开源框架 Midway Serverless，它就是一套面向 Serverless 云平台的开发方案，可以使用 Koa、Express 或 Egg.js 作为基础 Web 框架，同时也提供了独立使用的基本解决方案，如 Socket.io、gRPC、Redis 和 RabbitMQ 等。这样它就非常适合前端/全栈开发人员用自己熟悉的语言框架去实现业务功能，同时也可以充分享受云端一体化的应用开发体验，代码完成后即可一键部署到 Serverless/FaaS 环境中，而不用再关心资源配置等细节。所以 Node+Serverless 的组合，在未来很长的时间里应该都是主流趋势。

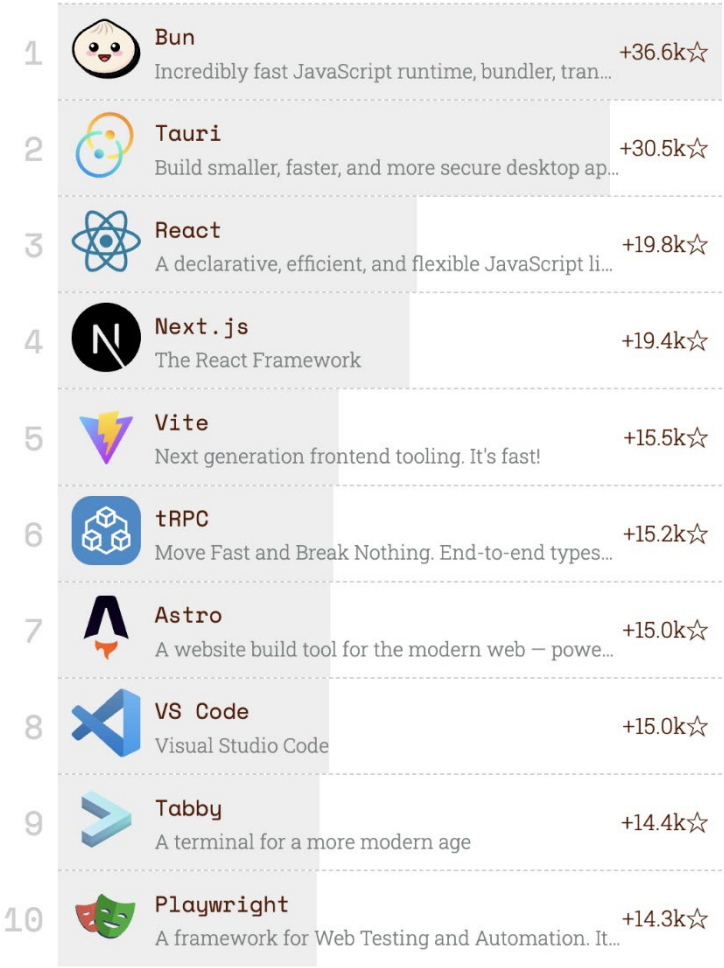
第二部分：回顾过去，畅谈未来

InfoQ：2022 年前端领域发生的最令您印象深刻的事情是什么？

黄健：我最近专门关注了 2022 年 JavaScript 明星项目的榜单，也细读了 2022 年前端大事记，这些对于前端技术的发展趋势都做了很好的总结。最大的变化应该是大家对于前端明星项目的关注不在是只围绕着前端流行库的新特性和各种 Web 应用框架了，而是将视角延伸到了诸如高性能+开箱即用的 JS 运行时之类的更丰富多彩的工具链生态和全栈型项目，其中用更多原生语言来实现前端工具链似乎已经成了行业都认可且在广泛实践的方向，像年度榜单冠军 Bun.js 使用了 Zig 的低级语言，它从头开始实现了数百个 Node.js 和 Web API，内置了构建打包器、转译器和包管理工具，战斗力简直爆表；Turbopack 增量打包工具发布伴随着 Next.js 13 的发布更是一鸣惊人。

大家从未在前端工具链上投入如此高的关注，这些生态里越来越多的工具都在经历从被 JS 实现到被 Rust 重写，效率都是数倍乃至数十倍的提升，这就要求未来前端的技术演进不单单是只需关注工具应用层，而对计算机系统、编译原理、新式编程语言、架构设计能力提出了全新的挑战，总而言之：前端工程师在拥有丰富武器库选择的同时，需要更主动激进地去探索更多工程上的极限！

当然，降本提效也是大家都在聊的一个主题，前端工程师应当持续性地提升团队的经营意识以及如何更好地为企业创造价值（省钱 vs 赚钱）。在这个背景下，我会比较看好全栈型项目的持续推广使用，比如：Next.js 开发商 Verce 在 2021 年就获得了 1.5 亿美元 D 轮融资，估值超过了 25 亿美金；Figma 被 Adobe 200 亿美金收购，这些团队中的核心成员有很多都是明星前端，这也给前端团队的未来经营起到了很好的示范作用：不仅可以挑战新式技术，还能持续创收。


1
Bun
Incredibly fast JavaScript runtime, bundler, tran...
+36.6k☆
2
Tauri
Build smaller, faster, and more secure desktop ap...
+30.5k☆
3
React
A declarative, efficient, and flexible JavaScript li...
+19.8k☆
4
Next.js
The React Framework
+19.4k☆
5
Vite
Next generation frontend tooling. It's fast!
+15.5k☆
6
tRPC
Move Fast and Break Nothing. End-to-end types...
+15.2k☆
7
Astro
A website build tool for the modern web — powe...
+15.0k☆
8
VS Code
Visual Studio Code
+15.0k☆
9
Tabby
A terminal for a more modern age
+14.4k☆
10
Playwright
A framework for Web Testing and Automation. It...
+14.3k☆

InfoQ：以前我们开玩笑说“前端一天一个新框架”、“前端难度 18 个月翻一番”，您是怎么看待这个问题的?

黄健：这句话应该是经典的摩尔定律在前端领域的描述，我自己上网的时候也经常会看到这句话，我觉得可以一分为二地看：

一是前端技术的迭代确实非常快，各种技术推陈出新，我觉得这也间接地证明了这个方向本身还是非常活跃的，还在处于上升期，如果大家都不在讨论新东西了，那反而需要想想是不是到了行业发展瓶颈期了。

二是大家普遍关注的生存问题，即在如此剧烈变化的环境下，如何更好地实现个人发展。我可以介绍下我自己的适应经验，那就是虽然技术变化快，但是难度更多地是体现在设计原理上。应用层的使用一定是倾向于越来越简单的，不然针对新生代的开发者会很难大规模地去推广使用。

因此我觉得可以把知识领域做些细分，做好内外兼修，像基础知识这类，属于内功修炼，任何时候都不会过时，底子打的越牢固，适应能力也会越强，所以要戒骄戒躁，持续学习；再就是领域知识，这里面又分为通用和专用知识，比如：前端工程化、图形可视化、Web 性能优化等就属于通用知识，一般也和大家负责的业务方向紧密相关，这些可以做些深入研究，多积累行业前沿的经验，争取成为该领域的专家；最后就是专用知识，也是大家平常讨论最多的，如：UI 库、Web 框架、新出的构建工具等各种轮子，这个变化确实非常快，和日常开发写代码也最为紧密，我将它们理解为是“术”，一般要按需学习，主要考验的是动手能力，优先满足实际业务需要，熟练后大家的竞争差异会相对较小，如果你愿意折腾也可以多看看里面的一些比较好的设计。

InfoQ：但从近几年发展来看，前端新趋势是否正在逐渐变少并开始趋于稳定呢？

黄健：我觉得每一个开发者是都需要去了解当下的技术新趋势，就像我们每年都会去关注 Tiobe 的编程语言排行榜一样，它代表了开发者的技能能否跟上趋势，或是否有必要作出战略改变。近几年前端新趋势变少了，我觉得是行业普遍认可前端开发已经抵到达了深水区，当跑马圈地到达了一定阶段之后，大家就会慢慢地走向务实，从扩张模式重新回归到精耕模式，关注点会逐步放回脚下，把之前 70 分的工作往 90 分推进，更加注重极致性能、用户体验和效率提升这些老生常谈的问题，也会更加注重投入产出比。

InfoQ：目前跨平台比较成熟且值得推荐的框架有哪些？您认为还有哪些框架是非常有潜力并可能会在未来被大家所熟知或使用的呢？

黄健：跨平台一直以来都是一个比较大的概念，专业术语叫 Cross-Platform，直译为跨平台，也叫跨端，其本质目标都是一样的，就是希望能做到“一次编写，处处运行”

（Write Once, run everywhere），原理上基本就是提供一个标准化的容器，这个容器是跨平台的，并且屏蔽了所有系统底层差异，封装好了系统调用，并对外提供了统一的API。最典型的例子就是我们的浏览器内核Blink。其好处就是只要我们写的页面是按标准规范的实现的，就可以同时跑在多个平台下。

目前跨端概念发展最快的应该是在移动端领域，从早起的 Hybrid、React Native 再到现在的 Flutter 等，技术生态也一直在持续演进、所以如果说要推荐，我觉得还是要结合自己的业务发展阶段和使用目标去看，我更倾向于介绍下如何去提升框架的选用标准，开发者自己做加权，而不是介绍哪一个具体的框架，有如下的选用标准大家可以参考：

- 一次学习，处处编写
- 极致性能
- 技术架构的先进性
- 规模化后的代码可维护性
- 高度化的 UI 组件定制能力
- 二次开发和企业定制能力
- 开发资源丰富，社区活跃度高；

跨端框架的发展趋势和潜力有几点可以关注：一是这几年桌面应用开发的重新回暖。像桌面级的跨端开发框架我觉得热度还会持续走高，这里面就包含了成熟的 Electron 和新起之秀 Tauri；二是结合国内的现状，小程序作为 H5 页面形态的补充，大有持续发展的趋势。围绕小程序开发的开放式跨端跨框架解决方案我觉得还是比较有潜力的；三是字节跳动内部现在做的一些探索，用纯 Web 技术栈去绘制 Native UI 的高性能跨端解决方案，重点围绕全屏和半屏页面的实现，比如：大家在抖音中看到的购物商城、

世界杯专题、新年活动等都是用这类方案做的实现，主打灵活和性能；最后是原生语言自身的演进。声明式 API、响应式设计、组件化、函数式编程这些前端领域已经非常主流的方案同样也在影响着移动端原生 UI 方案的进化，像 iOS 的 SwiftUI、Android 里的 Jetpack Compose 都是值得去长期投入的方向。

InfoQ：老师您是否在过去一年中使用过微前端相关的技术？经常采用的微前端方案有哪些？并请老师介绍下微前端的趋势？

黄健：关于微前端技术我们是在深度使用中，像在字节跳动内部的云计算平台中，从 2021 年开始改造至今，已经有 130 个子应用接入了，我们自身的体会是微前端不只是一门具体的技术，而是整合了技术、策略和方法，可能会以脚手架、配套工具和规范约束等成体系的形式综合呈现，算是一种宏观上的架构，业内也有 Single-App、Qinakun，Microapp 等，它们各有千秋，只要是适用于当前业务场景的就是好方案，所以大家要把重点回归到微前端需要解决的问题上。

因为微前端的使用不只是一门具体的技术，而往往是一套体系，所以大家可以更侧重关注一下如何将微前端的全套体系和自己公司的基建做好结合，这样才能最大程度地释放前端的潜力，具体可能会包括如下的内容：

- **治理体系**：简单看可以视为是一个上线管理平台+上线发布流程，包含应用声明周期管理、依赖管理等；
- **运行时容器**：容器承载着主应用，通过在主应用中注册子应用的方式来实现微前端，也可看做是一种 Runtime 的方案，像我们感知最明显的 JS 运行沙箱、CSS 的隔离、路由的加载/卸载、应用通信等，都是在这里面完成的。
- **开发配套**：给开发者提供完整的文档指引、脚手架和发布平台，方便应用的快速部署调试。
- **微物料**：微物料将微前端的粒度级别从应用级延伸到了组件、函数级，同时也可以让各个接入应用可以共享平台下的生态能力；

综上所述，可以看出来好的微前端方案都应该上在这四个维度展开建设的。

说到经常采用的微前端方案，目前以经典路由分发+资源处理的主子工程方案是大家

最成熟的，开山派的 Single-spa、阿里的 Qiankun、字节跳动的 Garfish 都是在生成环境中验证过的；另外就是以 Webpack5 的新特性模块联邦（Module Federation）为代表的去中心化方案，多个独立构建的应用之间可以动态地调用彼此的模块，轻松地拆分应用，真正做到跨应用的模块共享在业内也有大量实践。另外还可以关注下基于 Web Components 的微前端方案，核心是组件化的思维，以组件加载的方式将微应用整合在一起作为微前端的一种手段，如果对浏览器运行环境无要求，业务也是全新开发的话，可以去尝试一下的。

关于微前端的趋势，大家可以关注的一个点就是微前端的概念是出自传奇软件大师马丁•福勒（Martin Fowler），他也是微服务的概念的提出者，所以微前端其实也可以理解为：用微服务的思想来开发前端工程，那微服务中的很多设计思想都是可以用来借鉴的，比如：

- 更完美的应用隔离（容器运行时）
- 更完善的应用健康度观测（观测治理）
- 更高效的应用加载/卸载（编排调度）
- 更完善的应用间通信（消息队列）
- 更精细的微前端访问治理（流量网关）
- 更快速的前端展现（边缘访问加速，容灾建设）
- 多微算微，服务边界（领域驱动）

InfoQ：现在公司或组织都在向云迁移，这是否会给用户界面带来新的需求？前端开发人员应该如何做好相关准备呢？

黄健：结合云原生的概念描述，有个观点很重要，那就是云原生是一整套利用云计算交付模型的优势来构建和运行应用程序的方法论，但是我们感知最明显的公有云只是

其中的一种形式，但类似云的交付并不仅限于公共环境，还包含私有、混合云、分布式云等。作为云原生下的前端开发，我们面临的第一个挑战是我们的产物交付也需要能在多种云形态之间进行平滑的切换，甚至能做到一次编写，处处运行。这个对工程基建、物料统一、本地开发工具的完善都有非常大的挑战。比如，作为 C 端用户，可能你的一次更新，所有的用户都可以快速体验最新的功能。

但是作为云计算平台的交付产品，我们的产品原子特性、服务版本都是需要各种灵活定制组合的，甚至不同的客户用到的版本和功能都会有差异，这些就对我们的产品研发提出了很高的要求。另一个就是增质提效，云计算平台其实非常类似于 B 端的站点开发，对于组件的丰富度、代码逻辑的抽象都有很高的要求，而且一旦有了异常会直接影响用户的功能使用甚至经济收益，这些对于页面的容错性要求也是非常高的，用户可能会因为你的一次故障而彻底抛弃你。

前端人员的最大准备应该是要适当跳出前端视角，从云计算开发工程师的视角去思考我们做的事情，按云产品的交付标准去打磨产品，不要抵触领域知识的学习，同时也要真正做到从开发到上线都能云原生化，积极融入业务。另外就是云计算的市场空间非常大，行业内可互相学习的产品也非常多，还是要保持积极开放的心态，多体验多交流。

InfoQ：这两年我们看到 Rust 工具融入了前端生态，比如 SWC、Turbopack，那您认为前端工程师需要学习这门新语言吗?

黄健：在前面聊到的 2022 年回顾里，我提到了一个概念就是：前端工程师在拥有丰富武器库选择的同时，需要更主动激进地去探索更多工程上的极限！目前的行业趋势大家也不难发现，更高性能的 TypeScript/JavaScript 转译器已经尝试在用第三方语言实现了，例如：基于 Rust 实现的 SWC、基于 Go 实现的 ESbuild 等，给前端生态带

来了全新的活力。在字节跳动里，我所在的团队也在加大对 Golang、Rust 语言的基建。通过 Rust 自研 JavaScript 底层工具链，实现倍速级别的性能和开发体验提升已经作为团队的一个技术方向在推进落地了，从行业趋势来看，Vercel 和 Rome Tools 两家明显公司也在使用 Rust 整合整套工具链。

所以，未来前端工程师在稳住 JavaScript 基本盘的同时，还需要再深入学习 1-2 门高性能语言已是行业共识，学起来！

InfoQ：在您看来，未来 TypeScript 会超越 Javascript 成为新的前端标准吗?

黄健：TypeScript 是种很棒的语言，它显著地改善了开发者体验。它还新增了安全层，促进了工具质量提升，其技术生态也欣欣向荣。网上之前有做过统计，有八成以上的用户在使用 TypeScript 开发项目之后都表示会继续使用，这说明对这一语言的满意度是非常高的，而且它能真正地解决开发者的很多痛点。但是抛开语言层，TypeScript 本身也不能说是完美的，比如性能一直以来都是 TypeScript 最受诟病的问题。

由于 TypeScript 是自实现的，而且这种实现非常复杂，它的类型系统本身可以算是种迷你编程语言，这导致类型检查的速度极其缓慢，另外它自身也存在很多缺陷，比如 allowJs 配置选项、any 类型和 intersection 类型，其类型系统根本无法保证代码的类型安全。所以我更看好 TypeScript 和 ECMAScript 的标准互相对齐、持续兼容，而不是互相替代。这里额外推荐两篇文章大家可以去看看：《I'm porting tsc to Go》、《下一代前端语言之争，JavaScript 要被新语言反超？》

InfoQ：前端从业者需要掌握哪些工程化技术?

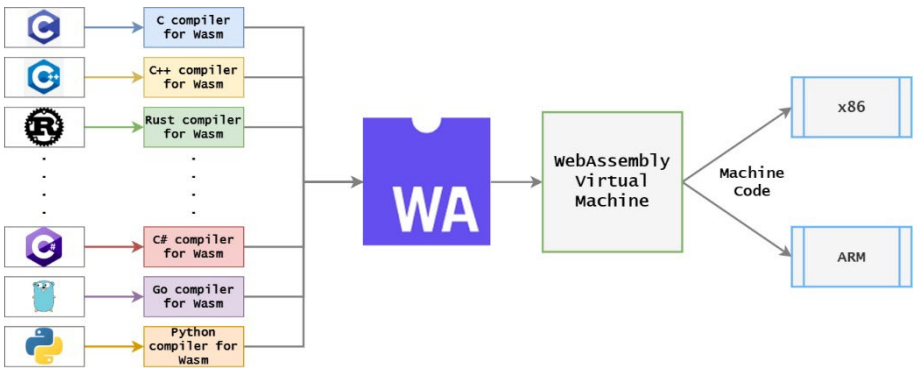
黄健：广义的前端工程化：前端工程是软件工程的一个子类，目的是将软件工程的方法和原理运用在前端开发中，最终实现高效开发、有效协同、质量可控。

狭义的前端工程化：前端工程是指将开发阶段的代码转变成生产环境代码的一系列步骤，主要包括：开发构建、CI/CD、Devops、部署发布、错误与性能监控、自动化测试等。

综上：需要覆盖从开发到线上的完整流程，尽量做到开发快，测试准，部署稳，同时做好对应的配套平台建设。前端工程化的最佳实践网上可以找到很多公开资料，对于这块各个公司都非常愿意拿出来交流，我自己也经常关注 InfoQ 的专家主题分享，收获很大。

InfoQ：您如何看待 WASM 在未来的发展趋势？我们前端从业者需要掌握这门技术吗？

黄健：WebAssembly（缩写 WASM）是一种可运行在现代网络浏览器中的新型代码，它的设计目的不是为了发明一种新的编程语言，而是为诸如如 C、C++、Rust 之类的语言提供一个高效的编译目标。它在安全、可移植、轻量化、高效率的虚拟机沙箱中执行，并且可以在不同平台上实现接近本地的运行速度，具有天然轻量、安全、快速等优势。



在浏览器端，它主要适用如下场景：

1. 将 C、C++、Rust 等高级语言编写的程序移植到浏览器，例如 Photopshop 的在线版、Figma 等；
2. 图形图像处理领域，如：文件解析、加密处理、数据可视化等对 Web 应用有较高性能提升要求的场景；
3. 音视频编解码等，比如 FFmpeg 与 WebAssembly 实现纯前端视频截帧这种流媒体下的解决方案；

不止是浏览器，还包括云端环。值得一提的是，WASM 的轻量级、安全模型、隔离性使得它在广泛的环境中都可以使用，只需要提供一个符合 WebAssembly 标准规范的虚拟机环境即可，从边缘云到 IoT、再到函数计算领域，都可以作为类似 Docker 的轻量级替代方案来使用。云原生社区最近发布了《2023 年 WebAssembly 技术五大趋势预测》非常值得读一下的。

作为开发者，我一直有个观念就是：项目里我可以不用，但是技术上我不能不懂！

InfoQ：您认为在 2022 年及以后，前端开发人员需要具备一些虚拟现实（VR）和增强现实（AR）的技术知识吗？

黄健：这个概念目前还是比较前沿的，我可能没法直接从技术角度给出分析，但是可以看看目前资本的走向和政策引导趋势。

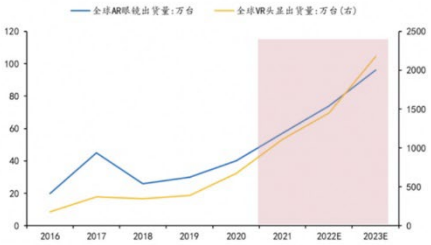
自 2022 年以来，华为、三星、索尼等大厂纷纷在 AR/VR 终端和各类设备中持续布局，期待新产品的推出提升使用体验，推动硬件渗透率提升。据统计 2022 年全球元宇宙产业共发生了 704 笔融资，总金额达 868.67 亿元。伴随重量级厂商的入局，加

之技术迭代发展、内容生态扩容、产品出货量增长，VR/AR 产业生态日益成熟，

图表：2022年上半年国内AR/VR融资并购额和事件均大幅上升

资料来源：陀螺研究院、国海证券研究所

图表：消费端AR产品规模化上市，出货增量提升



资料来源：IDC、智研咨询、我爱音频网、国海证券研究所

国家和地方层面持续推出激励政策。据统计，我国有 15 个省市政府出台了 29 项元宇宙专项扶持政策，市场对 Web3 在传统互联网升级中的重要作用和技术方向上的创新性认识已经逐渐清晰，未来 3 年元宇宙/Web3 将可能会出现 DAU 超过 1 千万的区域性大规模应用，需要提前做好布局准备。

城市/国家部委	日期	政策/会议名称	政策内容
上海	2021年12月21日	市委经济工作会议	引导企业加紧研究未来虚拟世界与现实社会相交互的重要平台，适时布局切入。
工业和信息化部	2021年12月23日	中小企业发展情况发布会	培育一批进军元宇宙、区块链、人工智能等新兴领域的创新型中小企业。
浙江省	2022年1月5日	《关于浙江省未来产业先导区建设的指导意见》	聚力构建以人工智能、区块链、第三代半导体、量子信息、柔性电子、未来网络、空天一体化、生物工程、前沿新材料、先进装备制造、先进能源、元宇宙等领域为重点的未来产业发展体系。
杭州	2022年1月5日	《关于浙江省未来产业先导区建设的指导意见》	元宇宙是浙江到2023年重点未来产业先导区的布局领域之一
合肥	2022年1月10日	《政府工作报告》	未来5年，合肥将前瞻布局未来产业，瞄准元宇宙、超导技术、精准医疗等前沿领域，打造一批领航企业、尖端技术、高端产品。
武汉	2022年1月11日	《政府工作报告》	加快壮大数字产业，推动元宇宙、大数据、云计算、区块链、地理空间信息、量子科技等与实体经济。
中国信息通信研究院安全研究所	2022年1月24日	元宇宙安全研讨会	元宇宙是一种全新的互联网交互模式，将带来开放协作的经济形态和商业模式。
海口	2022年1月28日	《政府工作报告》	建设网易元宇宙产业基地项目，推动海南数字化文创产业高质量发展
成都	2022年2月14日	《政府工作报告》	成都高新区成立元宇宙产业联盟，促进搭建横跨多领域的元宇宙产业发展平台
厦门	2022年3月23日	《厦门市元宇宙产业发展三年行动计划(2022-2024年)》	开展五大行动，抢抓数字经济和元宇宙发展新机遇，打造“元宇宙生态样板城市”和数字化发展新体系。
山东	2022年3月24日	《山东省推动虚拟现实产业高质量发展三年行动计划(2022-2024年)》	山东将打造以青岛为中心，济南、潍坊、烟台、威海四市联动，其他市协同的“1+4+N”虚拟现实产业区域布局。
安徽	2022年3月29日	《安徽省“十四五”软件和信息服务业发展规划》	支持企业开展虚拟现实、增强现实、3D引擎、物联网等技术创新，引导企业积极布局元宇宙新兴业态，开展元宇宙平台建设。
上海	2022年6月16日	2022年上海全球投资促进大会	发布了元宇宙、绿色低碳、智能终端、数字经济等四大“新赛道”投资促进方案。到2025年元宇宙产业规模力争突破3500亿元。
上海	2022年7月8日	《上海市培育“元宇宙”新赛道行动方案（2022-2025年）》	到2025年，“元宇宙”相关产业规模达到3500亿元，带动全市软件和信息服务业规模超过15000亿元、电子信息制造业规模突破5500亿元；培育产业优秀企业；打造元宇宙适用场景；完善产业生态，建设“元宇宙”园区。
河南	2022年9月21日	《河南省元宇宙产业发展行动计划》	到2025年，全省元宇宙产业发展初具规模，核心产业规模超过300亿元，带动相关产业规模超过1000亿元，初步形成具有重要影响力的元宇宙创新引领区。
武汉	2022年11月3日	《武汉市促进元宇宙产业创新发展实施方案（2022-2024年）》	力争到2025年，高水平建成2个以上元宇宙产业基地、3个以上元宇宙重点平台，聚焦重点行业领域打造50个以上元宇宙典型应用场景和项目，培育引进200个以上元宇宙创新企业，建成创新链、产业链、价值链协同发展的我国元宇宙创新发展先导区与核心区。

举个例子，过去疫情期间，大家都会越来越多地采用云上的方式解决实际生活中的刚

需问题，比如：线上租房看房、AR 试穿鞋子衣服、家庭装修效果预览等，这些都是作为实体行业的补充增强，可以考虑朝 VR/AR 的赛道做些转型尝试，但是需要保持足够的热情和耐心。

InfoQ：2023 年前端技术领域您判断会出现哪些比较重大的变化？对于前端从业者在 2023 年可能面临的最大挑战会是什么？？

黄健：我自己比较看关注的几个方向：

1. **人工智能（AI）应用加速，会慢慢地从幕后走向台前。由于 AI 变得更容易获取、成本更低，人们对 AI 的应用正在加速。**无论是在分析最新数据方面，还是在如何最好地处理问题给出建议方面，AI 都能帮助人类理解这一切。前阵子，AI 聊天机器人 ChatGPT 爆火，能够写代码，修复 bug 甚至写小说，推出五天直接突破了 100 万用户，也让其母公司 openAI 的估值翻了一倍，还有 Github Copilot 和开发工具的深度结合已经让开发者在持续深度使用，未来利用好这些 AI 基建设施，前端可以更容易地做出更多有创意的平台或产品。
2. **前端职能泛化，前端开发者朝着全栈开发者的成长趋势会越来越显著。**外部因素包括：全球疫情后的经济复苏，企业更加注重降本提效，用更少的人办更多的事。内部因素包括：伴随云基础设施的完善，多语言的探索尝试，开箱即用的一体化框架持续强大，开发者自身也愿意更前一步，逐步去适应产品开发的全流程，从界面到接口、从架构设计到代码落地、从自动化测试到运维等等，拥有全链路技术有益于技术的闭环，提升个人能力和性价比。
3. **越来越多的前端明星产品会由系统级原生语言来实现。**2022 年可以看作是高级语言在前端工具链大放异彩的一年，前端对于高性能系统级语言的学习热情也在持续走高，大家也不限于隔着各种运行时去和系统做交互，相信到了 2023 年随着基

础层能力的不断补齐，在平台层、业务层前端的创造力都会被更大的释放，会有更多精彩的作品被打造出来。

4. **Serverless 的概念会越来越被接受，更多的 Node BFF 应用会优先跑在 Serverless 平台上。**各种 Web 框架在特性更新上也会持续加大针对于 Serverless 的优化，资源成本低、高可用、免运维、开发成本低的优势也会被持续放大。对应的开发者生态，比如本地环境、调试工具、应用市场等也会得到持续发展。
5. **低代码发展呈现齐头并进的局面。**面向用户侧，在应用层越来越多基于低代码搭建的场景将被持续推广落地，产品会持续变的好用易用；面向专业开发者，低代码引擎类的产品强化会越来越被重视，让上层开发者搭建平台变得更容易更快速，引擎开发者自身也会持续关注能力的不断抽象输出。同时，配套的低代码和无代码云服务也会持续发展。
6. **前端开发者与世界的联系会越来越紧密。**目前在 Vue/React、T39 提案、W3C 等社区中活跃着越来越多影响世界的中国前端开发者，且大有年轻化的趋势，开发者对于新技术的快速跟进和落地在过年的几年里都有了显著的提升，一线优秀的理念从诞生到验证落地会做得越来越快，未来我们不仅仅是技术的追随者，也会是参与共建者。

查看更多的趋势报告：2022 年 JavaScript 生态圈趋势报告、2022 年 CSS 生态圈技术趋势！

对于开发者的挑战可以参考阿里巴巴的年度词：进，以奋进之心，迎接全新一年。同时也分享一下我们团队的年度关键词：突围：走出舒适区，冲出浏览器，不再定位成业务的追随者，需要更主动地去做些产品形态和技术驱动上的革新，让前端开发者在更大的舞台上去体现专业价值。

第三部分：职业发展，未来前景

InfoQ：在大公司和小公司工作，前端技能上有什么不同吗？

黄健：我觉得技能上应该没有本质区别，只是接触到的资源和可以探索的方向不大一样，大公司里的基建一般比较完善，有着很清晰的工程师能力模型以及职业发展规划，所以大家确实在技术视野上要好一些，可以用到的资源也丰富很多。但是存在一个问题是人员越多分工则越细，如果自己不主动破局，时间长了就成了流水上的螺丝钉，需要持续想想自身的独特价值和不可替代性。小公司我觉得大家要多把自身的成长与企业的发展结合好，关注当下痛点问题，在精不在多，用更少的资源办更多的事，同时也可以多主动同社区同行交流，拓展技术视野。

InfoQ：相对后端来讲，学习前端这门技术会更有前途吗？

黄健：任何行业能做到头部的 5%，都会非常有前途，从前面聊到的趋势来讲，我们也认为未来前后端的边界会越来模糊，知识体系也会不断的交融。前几年因为行业处于扩张期，无论是新品还是用户增长都需要前端去创造产品的形态，在 APP 端、小程序、H5、企业中后台都存在大量的岗位需求，但是随着行业开始步入平稳期和深耕期，大家需要把工作重新放回到工作效率、研发成本和极致性能上，举个例子：在企业用人资源紧张的情况下，让一个后端开发者去使用下前端的基建框架，凑合凑合他们也是具备上手能力的，但是反过来让前端去临时支持下后端的业务，那上手的门槛依旧不低，最后你会选择用哪个应对方案，结果是显而易见的。

InfoQ：有同学问，前端工作需求很多，总是改来改去，在实际工作中也是产品决定业务逻辑，那么相对来说从事底层基础服务会不会更有挑战和职业未来？

黄健：这是一个围城现象，外面的人想进来，里面的想出去，很多做基础服务的同学

也会经常找我吐槽，也会面临如下的困境：辛辛苦苦折腾出来的工具或者技术方案推不动，业务侧觉得够用就好，在收益有限的情况下引入新工具会带来额外的成本，服务写好了还得做培训当客服，做底层方案很难直接共享到爆款业务快速增长的红利，做了好一阵子的技术方案结果社区出来了个更好的直接就被降为打击了等等。其实很多从事底层的基础服务的同学也时常想着回业务中去。

解决思路：工作不分高低贵贱，重点还是要关注当下的痛点问题是什么，适合用哪种方式去解决，在解决过程中技术可以带来哪些提升亮点，无论是技术还是业务只要能持续提升都是非常有挑战的。

InfoQ：零基础小白应该怎么学习前端才能少走弯路？

黄健：

1. 多看多练。一万小时定律是绕不开的，看书、看视频、看文章可以交替着来，但是一定要有个完整的知识图谱。
2. 多聊多问。与社区一同成长，我自己也是非科班出身，但是非常感谢一路过来指导过我的很多前辈；
3. 找到自己的擅长方向，尝试做出有自己特色的代表作品，在未来的发展中会是加分项；

InfoQ：以老师见过的一些例子或自身经历来说，我们如何成为一名优秀的、出色的前端工程师？有什么特定的技能或特质可以让我们成为一名强大的前端开发人员吗？

黄健：

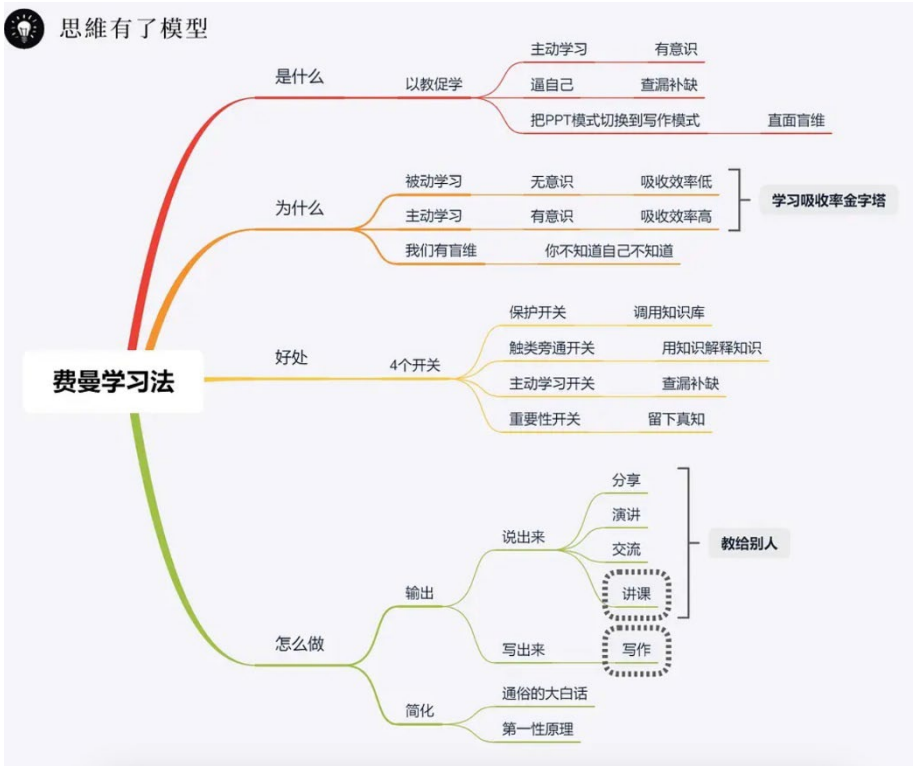
1. 不设边界，保持学习的热情，多折腾，前端工程师中工程师的特征要能被持续放大；
2. 持续学习，尤其是关注基础知识修炼，当初因为快速入门而淡忘的知识，未来也

会随着时代的发展而被要求偿还；

3. 学好英语，能发现更多的机会，在更大的舞台上去展现自我；
4. 劳逸结合，未来的道路很长，身体才是革命的本钱—《小妖怪的夏天，妈妈唠叨》。

InfoQ：如何在一门技术上做到精通？比如深入理解一门框架工具人 React.js？

黄健：推荐一下费曼学习法和番茄工作法，然后把一个知识点的来龙去脉都彻底弄清楚；



InfoQ：前端工程师还应具备哪些出色的“软技能”，例如沟通和团队合作？

黄健：

1. **良好的沟通表达**：前端经常处于多个业务的交叉点上，天然具有连接器属性，需要增加自己的表达沟通能力；
2. **保持积极乐观**：对事情的不确定性保持乐观，只有你更愿意去尝试，才能更多收获成功的机会，努力走出第一步；
3. **做事不设边界**：很多问题从前端界面暴露出来，但是最后的解决往往是在后端、数据或者产品侧，但是一定不要抱着不是我的问题我不管的思想，始终坚持自己

也是产品的一部分，要为最终结果负责；

4. **坚持写作分享**：文字沉淀和口头表达能力未来都会是职业发展过程不可或缺的能力，甚至是加分项；
5. **开卷有益**：短视频时代都是碎片化的信息找人，可以静下心来，多选些好书精读一下，推荐：《软技能 2 软件开发者职业生涯指南》

InfoQ：新年新开始，老师对正准备跨进前端的技术人或者正在前端领域工作的技术人有哪些寄语或者建议呢?

黄健：过去的 2022 年是个非同寻常的一年，相信大家都感受到了近几年来比较严酷的外部环境和业务挑战，不过套用一句时髦的话来说就是“**感觉辛苦，是因为你在走上坡路**”，新的一年我们一起出发，未来加油！

技术人穿越周期的生存之道是转型管理？ | 展望技术管理者的 2023

作者：姜昕蔚、Tina

2022 年，似乎裁员事件很少涉及到管理层，而且职场中年，做管理比做技术更安全？那么我们应该不应该转型管理？另外，在 2023 年里，对于已经在管理岗位上的人员来说，应该如何审视自己的职业发展和定位才能走的更顺？

在 InfoQ 一年一度的《年度技术盘点与展望》系列节目中，我们邀请到了戴尔科技集团软件工程高级总监滕昱，畅聊技术管理者的技术成长路线与职业发展路径等话题。

完整内容可点击查看回放视频：https://www.infoq.cn/video/vPs34zSoBK2Yku08lyme

主持人：请老师给我们简单的自我介绍?

嘉宾：首先欢迎大家今天来参加我的直播，回顾我自己的职业生涯，算到 2022 年末我已经从业了整整 20 年了。而整个经历可大概分为三段：2002 年加入 Konami 公司，大家如果玩电子游戏会知道这家公司开发了魂斗罗/合金装备等一系列大名鼎鼎的游戏。不过入职大概一年后，我发现自己并不适合游戏开发这个领域，于是果断跳槽；2003 年加入 LSI logic，这是一家半导体芯片公司而我当时选择的是 Firmware 开发方向。大概 3 年以后我在 2007 年加入 EMC(现戴尔科技集团)开始第三段长达 16 年的工作经历。我在戴尔科技集团主要经历是设计开发分布式对象存储并支持产品的全球商业化运作。

现在我们回头去看，大家都会把 2006 年当作云计算的元年，而对于存储界来说最重要就是 AWS 发布了 S3 存储服务，所以从那时候起一大批优秀的对象存储产品就逐步出现在业界之中，其中就包括我所参与的戴尔科技集团前后三代产品，并一直保持 Gartner“分布式文件和对象存储”的魔力象限领导者地位。

而我的“额外”任务是在 2016 开始拥抱流处理领域开源浪潮，加入了戴尔科技集团的流存储(Streaming Storage)开源项目 Pravega。并和同事一起最终将该项目贡献给了 CNCF 基金会。在这个过程中，我学习到了开源的理念并也结识了很多开源界的朋友，比如 Flink/Iceberg 社区一些早期开发人员。同时也开展了一些 Pravega 中文社区工作。

技术管理者需要的那些技能

主持人：我们知道老师刚休完一个长假回来，看起来生活和工作很平衡，这也是大家都在讲的一个话题，但其实有很多公司都在鼓励“投入更多时间来工作”，这也国内的一大特色，你认为这是什么因素促成的?

嘉宾：首先纠正一下，作为全球化开发团队的一员并且直接负责支持大客户的我来说，其实也是无法避免在 8 小时之外上线救火，毕竟作为一家面向 500 强企业解决方案的公司来说，保证 SLA 的执行是最重要的事情之一，而这就包括在半夜紧急上线去响应 S1 的 Case(30 分钟内必须响应)。所以为了保障对客户“n 个 9”的承诺，我们是不可能 100%避免在八个小时之外工作的。

所以我理解这个问题其实想问的是如何看待“制度化”加班这个现象的。先给出我的答案，这取决于你的公司或项目处于哪个发展阶段而定。其实软件开发活动和所有其他人类经济活动一样，也是要遵循基本客观规律的。当公司或者项目处于初创阶段，所谓 0 到 1 的阶段，比如才拿到第一个大客户，那么万事草创，个人和团队投入更多时间是有非常大的边际效用的，即使是多投入一个小时，所带来的回报也是能看得到的。

但是天下所有的产品或者项目最终都会只有两种结果。一种是转变成一种成熟产品/商业模式(如按照华尔街的说法，就是是否有 10 亿美元的营收)。另一种结局就是死掉。所以成熟模式不仅做到是 0 到 1，而更是要完成 1 到 10 的蜕变。然而在 1 到 10 阶段，如果只投入更长的劳动时间的话，带来却是的是边际效益的急剧下降。因为这阶段项目/产品的客户基数变的非常大，而增长实际上每年变的只能能有 20%-30%甚至更低。所以此时遇到的问题大多是无法用简单堆人力就可以解决的。

鼓励“投入更多时间来工作”是解决特定阶段问题的“银弹”。但盲目使用地在本不该使用的阶段，最终可能导致整个团队分崩离析(毕竟人不是机器)和商业模式的失败。也就是我们反反复复说到的“没有银弹”这句老话。如果大家去稍微了解下业界那些大型的软件公司，大家其实都心照不宣的区分了这两个阶段，也算是业界的一点共识。

具体到我自己的经验来说，有两点可以和大家分享：

1. 我会给我团队的所有成员设置一个共同的 KPI，那就是每年除公共假期外，每个人至少计划 2 周不带电脑的休假。
2. 减少会议，每周最长会议控制在一小时之内，80%控制在半小时。每周五下午不安排会议(这其实是全球整个工程师团队的最佳实践)。为什么要这样呢？因为我们作为一个普通人是被潜意识中的期望所管理的。我们的焦虑很多时候来自于对未来的无法计划和掌控。一旦设置了正确的休假期望值，把时间控制权交还给员工后，假以时日，你会发现大部分员工潜移默化地提高了自己的效率。例如控制好会议时间后，员工会自觉的在会议前做好充足的 homework(家庭作业)，甚至已经私下就大部分内容进行沟通并达成共识，这样会上就不会过度发散，从而最终提高了会议效率。而这其实就是明确好 8 小时以外时间归员工所有带来的“看不见”的效率提升。

而更神奇的是，我不止一次的发现，很多工程师在休完假后，会提出很多“out of box”的想法和设计。而这些想法往往是天天埋在办公室里，并奔波于会议中的工程师无法得到的意外之喜。也许完全放松后，人们更容易灵感一现，说个笑话，牛顿是在苹果树下(很有可能是下午茶时间)而不是办公室里(这样也没办法被苹果砸到)发现的万有引力规律。

要知道，休假在口语中我们都习惯都用 Recharge,重新充电来表达。也就是意味着放松心态跟自己/家人在一起，而我们都明白家人是我们最大的心理安全感和未来确定性的的来源，如果不将足够的时间给到家庭和自己，长此以往，他们/她们真的能很持续不停的高质量输出吗？我个人觉得这个答案是否定的。

主持人：说到管理层面，大家可能觉得管理可能是比单纯的研发更难的，在您的视角中，您觉得技术管理它到底难在哪些方面，为什么会难呢？

嘉宾：我首先回答难在哪里吧，因为大家处于细分行业，以及产品项目具体的阶段不经相同，所有到底需要做那些事情和那些能力在细节差距会比较多，但是难点却是放之四海而皆准。

真正的难点其实就是大家一提到技术管理，自然把注意力集中在“管理”这个词之上，进而认为一旦自己被赋予了“管理者”这个角色，会自动获得一种神奇的魔力，即所有事情只要自己计划好，执行中亲力亲为，最终这个项目或是产品就一定会大概率获得成功。但非常不幸这是错误的理解，而明白这点这就是技术管理最难的一点。

在现实中，由于我们身处一个高速发展的产业，技术产品理念迭代可能远超其他领域，在我看来技术管理更像在惊涛骇浪上航行的船长/舵手，只拥有一张航海图和罗盘是远远不够的，更重要是技术领导力，即如何带领你的团队在各种变化中不停修正方向，最终到达原设定目标。即通常意义上项目或产品营收增长。

所以在我个人经验中，解决这个所谓难点的一个可能办法是要将“管理”这个带有从上往下俯视概念的词从我们的脑海中划出去，而把自己转变成平行的团队技术领导者。既不是“身处高位”只发号施令，也不是陷入细节泥潭的控制者，而是转变成技术方

向的领导者，带领大家去完成目标。这样的技术领导者的作用就和船长和舵手一样，能为团队指引一条路，并发动每个人朝着这个目标一起努力。并且放手让准备好的成员成为新的领导者，达到技术团队的成长壮大。

软件行业本质是千变万化，许多一开始非常好的项目往往都只是昙花一现，一旦做错了一次技术选择，衰落起来是非常快的。所以作为一个技术领导者，如果不能敏锐的看到行业未来几年的发展，做前瞻性的领导力预测，是不能为团队做好指引的，也就谈不上给团队成员带来成长机会。

下面举个我职业生涯中一个具体的例子去分享一下我如何做技术前瞻性预测并帮助提升技术领导力的：

我做了 16 年的分布式对象存储，其实在 2016 年~2017 年时候有一个很大的难题摆在我的面前，也就是对象存储的未来在哪？要知道业界那个时候对象存储还主要被定位于“二级存储”，“冷存储“，虽然可扩充性很好，但基本上用于存储海量图片或是历史数据归档这样的用例。但是这样的定位，是很难帮助客户产生大量的商业价值，从而导致对象存储本身也是不温不火。所以当时我面临的最大的需要回答两个问题是:

一是你是否相信对象存储会变成大数据处理领域的主存储，服务于最“热”那部分数据。

二是你要在在技术产品上要做哪些准备来迎接这个趋势的到来

第一个问题的答案是肯定的，因为我相信任何一种技术的成功与否，最终取决使用它的人数。开发人员总是倾向于自己最开始接触的技术是这个世界心照不宣的秘密。对象存储在 2016-2017 年时，离开 AWS 发布 S3 已经过去了整整 10 年，第一代使用

S3 开发应用程序的程序员正在成为我们的客户，并且成为我们的企业客户中具有技术选型话语权的的 Tech Leader，VP 甚至 CTO。

所以对象存储这个技过去了十年并且没有消亡，并也有了一定的发展。当时我按照第一性原理判断，既然有那么多的技术人员熟悉它，那么它大概率会成为未来大数据或非结构化数据的主存储。这里的第一性原理也是大家为什么投入开源社区的原动力。精炼成一句话：你只要抓住了技术人员，实际上就等于你就抓住了未来的商机和客户。因为这些技术人员在他们公司产品选型的时，他/她总是会偏向于自己熟悉并投入时间学习过的技术。

1. 那么对象存储是否具有成为主存储的潜力呢？我当时两个分析是：
2. S3 是基于 HTTP 协议的，具有最完全的网络访问潜力。

相比较 POSIX，S3 移除了很多传统存储接口语义上对于“并行”的限制，具有天生“scale-out“的潜力。

所以我认定对象存储在技术和语义上是满足成为大数据处理领域主存储要求的。而唯一的缺点就是基于当时的商业用例，业界并没有把适当的硬件加速功能用在对象存储上。所以我开玩笑说，2014 年-2015 年 Kafka 兴起就完全是打了一个对于对象存储的时间差，因为那时候的对象存储没有办法支持对于海量尾数据低延迟的读写，所以我们不得不用 2 套存储系统来把实时数据和历史数据区分对待，而留下的历史问题(lambda 架构)直到今天也没 100%解决完。

那么基于以上两个原因，在肯定对象存储会成为大数据领域的主存储这个前提下，我要做是让前瞻性的预测顺利在产品中技术落地。那么我同时注意到最近 10 年一个非

常重要转变就是外设速度和 cpu 速度差距的极速靠近，以至于最终诞生了 arm 架构的 DPU 这个趋势，而此趋势在 2016 年-2017 年瓜熟蒂落的结果是 2017 的 AWS re:Invent 上宣布的明星产品 Nitro。所以我也一拍即合判断出这个技术完全可复用在分布式对象存储上加速 IOPS 和吞吐量。而后来用户态编程，RDMA/NVMe 技术发展也完全验证了我的判断，在网络速度从 25Gb 到 100Gb 甚至现在展望 400Gb 过程中，对象存储也伴随着一飞冲天了。

如果我们事后诸葛亮一下的话，Snowflake 差不多也是在 2016 年开始使用对象存储做为数仓的“Primary storage”，大家可谓是所见略同。而且这几年开始大家甚至可以开始看到 HTAP 系统也会开始转向使用多云架构下的对象存储了。

而在这一过程中，“管理”其实就变成了技术领导力的一个附带 buff 了，难题迎刃而解。

主持人：在管理上还有许多需要细化的地方，有的技术管理者比较“微观”，直接上手做，有的则崇尚扁平化管理，您觉得技术管理者需要管理到什么程度?

嘉宾：这个问题其实在技术管理难点那里已经回答了，我的答案就是不要成为这两种类型中的任何一种，而是要成为一个技术领导者。不过还是让我们稍微分析下这个两种类型的管理者各有什么问题

1. 任务分包者/进度整合者：这是一种常见的团队管理方式，以至于我们一提到管理工作就会想到这个类型。这类管理者会将任务分配给每一个成员，列出具体计划，设定任务的提交期限，最终将团队成员的成果合并起来并及时汇报。毋庸置疑，这样的管理者还承担对外沟通工作并根据结果再次用任务分配来“管理”团队。

但是这样的管理者可以被称为合格的“技术”管理者么？实际上任务分包者/进度整合者的价值只是所管理的组内成员价值的总和，而对企业而言的他/她自己的附加价值则很难体现，最终导致工作虽然很繁重但可替代性很强，离开平台后价值可能急剧下降。

2. 细节掌控管理者：细节掌控管理者会将自己与每个团队成员捆绑在一起，每件事情都要百分百的参与和掌控，将自己很大部分的精力都花费在审阅甚至直接接管下属的工作上。诚然这样管理者可能不容易被取代，但是当需要管理的人数多起来后，细节掌控类型管理者很容易被“管理工作”压垮，因为这本质上不是个可扩展(scale out)的领导方式。但职场中这种类型的管理者也是相当多数，特别是工程师出身并转向技术管理的管理者都需要渡过这个阶段的考验。甚至有相当一部分管理者最后被管理工作压垮只能被迫放弃管理道路。同时因为这样的团队中每个团队成员发展都会受到这样领导者“带宽“的限制，很多情况下会影响团队士气，最终造成离职率居高不下。毕竟，没有人在职场中喜欢一直被当作“新手”，也不会喜欢一个“保姆”型上级。

主持人：作为管理者，您是如何管理 10X 程序员？而且这样的程序员通常是非常有性格的，如果团队存在这样的成员，您觉得该如何发挥他的长处？

嘉宾：首先，我可以非常清楚明确的告诉大家，10X 程序员是普遍存在的。但是可能和大部分人理解不一样的是，我所认识 10X 程序员全是在有扎实技术能力之外，还有着超强的沟通能力和超强亲和力的人，而是不是那种大家脑海中那种只埋头写代码不交流的代码狂人。

让我们回到软件行业的本质:一种人类经济活动，其根本在于如何成为人类世界和计算机世界的桥梁(bridge)去解决问题。涉及到如何理解你所面对市场和客户，如何说

服你的合作伙伴、投资方、VC 或者你的上级。一个合格的 10X 程序员往往会将 80%时间花在和人的沟通上，20%时间留给计算机。因为相对于人类沟通的复杂性，计算机世界容易掌控的多了。《人月神话》中有一个经典的论断：往一个落后的项目中投入更多的人会造成这个项目更加落后。为什么？其实道理是加入更多开发人员之后，人与人之间沟通变得更复杂了，指数级消耗后最终造成整个项目进度更加的落后。这也侧面佐证了软件行业中沟通所占比重之大。

甚至在我们刻板印象中觉得一些在生活中有可能并不善于沟通(当然只是我们觉得)开源社区的程序员，但是你会发现在他/她在所有参加的开源社区里面活动中，比如的 PR review/Issue review 里面其实都是非常健谈的一个沟通者。在这里我举一个例子大家都熟悉的 10x 甚至是 100x 程序员的例子：Linus Torvalds:Linux 和 Git 的这两个最伟大的开源产品的创造者。但我认为实际上他是这个星球上最成功的项目管理者之一了，我自己曾经比较深度的参与过 2.4 到 2.6 内核那段时间的 LKML 的讨论。我非常佩服他的沟通协调能力，用他自己方式将横跨这个星球不同组织/公司的开发人员管理起来，带领 Linux 这个奇迹从一个成功走向下一个成功。

我们应该如何做职业规划？

主持人：国内有 35 岁焦虑，到了 30 岁就要转型做管理，无论是出于年龄的焦虑，还是自我成长的规划，其实很多人都要这个的职业路线规划，对于从技术人员转型为技术管理人员，您有什么建议吗？

嘉宾：其实回答这个问题，我认为更重要的是要搞清楚技术管理是否真正适合自己，

实际上如果你对整个业界在 2023 年的展望不是特别有把握的话，那么在这个时候贸然选择管理路线其实是有降低自己的职业安全感(job security)的风险，因为公司对于管理人员的期望和技术人员是不一样的。我这边有测试自己是否适合做技术管理者的 3 个小问题：

1. 是不是能接受“不可控性”才是唯一的的常态。这点对技术出身的我们来说其实挺有挑战，因为我们的潜意识是觉得程序一定会按照自己设计好的样子完美的运行。所以我作为管理者，团队也必然会按照我设计好的方案去运行。可惜的是这个想法完全错误。一个最直接的例子是每个技术管理者都会面临核心人员流失的问题。员工离职原因千奇百怪，这是你无法控制的甚至无法知道背后真正的原因的，而由此带来项目的不确定性，你真的准备好了么？
2. 是不是有很好的心态去持续接受负面情绪。随着你在管理岗位上越走越远，你每天接受 60%-70%信息都会偏向负面，因为有问题都会找船长/舵手。那是否有强大的内心驱动力一直专注于目标也是很高难度的挑战。
3. 是不是在人群中有很强的主观意识成为讨论的焦点和中心。是不是有比较强烈的公开演讲的冲动，因为作为一个技术领导者，如何包装和推销你自己和你团队的故事是需要一定的人格魅力的。

以上 3 个问题以我的经验来说至少得有 2 项答案是“yes”，我才会建议你可以开始考虑转型做管理者。当然我们不否认有些技能是可以后天习得的，但是你需要先想清楚自己真的准备好了么？要知道，一旦转型失败重新做回技术人员的挫折对于大部分人来说是相当巨大的。

主持人：作为一个新手技术管理者，需要从哪些方面着手，如何规划自己的发展道路呢?

嘉宾：这是个千人千面的问题，我不能给大家一个在所有的领域都适用的答案，但也许下面 2 点可能有一定的参考意义:

1. 最最最重要的是，是放弃看待技术管理者的魔力 buff 加持。With great power comes great responsibility。在业界流行多年的双线职业路线就说明，理论上管理岗位需求量更少，要求也更高。最重要一点是你会直面整个组织的不确定性，从本质上挑战人类潜意识中的安全感。此外，技术岗越到后面比管理岗可能更容易升职。管理者的升职，必须要有一个与之匹配的团队/scope 去支撑，而不仅仅是技术能力的提高。所以在我观察到的不少海外同事中，是有相当部分的技术人员有意识不选择管理岗这条职业路线的。
2. 对于我们技术人员来说，也请有意识提高一下自己对于人与人之间沟通能力。尤其是面对面的沟通能力。首先，作为技术管理者需要和关键客户的面对面交流，理解最重要问题是什么，在软件开发领域我的理解是“店大欺客”是不存在，本质上唯一的护城河是性价比和沟通服务好坏。同时，技术管理者也需要与销售部门，产品部门面对面交流，这样才能了解业界需求的全部；并且，技术管理者更需要与上级、上级的上级面对面交流，打造属于你和你团队的故事和位置并得到他们/她们的支持；最后，技术管理者更需要与架构师团队、技术骨干面对面交流，获得他们的支持/她们以及项目执行的第一手状态。所以我一直反复强调的是，软件开发本质是人类的经济活动一种，做为技术管理者，最重要的是和人的交流能力，而并不“只是”超强的编程能力。

主持人：现在经济下行，各公司都在降本增效，而我们的管理者都还很年轻，没有经历大的经济周期波动，那他们在 2023 年应该如何审视自己的职业发展和定位呢，如何才能走的更顺?

嘉宾：我非常幸运的旁观过(但没有被直接波及到)2001 年-2003 年间第一次互联网泡沫破灭全过程，感慨很多,建议可能就是：

1. 不管任何情况首先得“活下去”。可以自己做个简单的 review，比如看看自己团队平均每个人创造的营收到 1M(100w)美元的安全线没有。这数字来自华尔街收购初创公司的不成文的行规，同时也被大家认为是一个项目大概率有资格活下去的安全数字线。只要能活下去，业务会好起来，股价也会好起来的。而现在很多伟大的大公司都是挺过了 2001 年的互联网泡沫破灭最后生存下来的，如亚马逊和 Google。
2. 经济总是有 up 和 down，人生的机会也不是只有一次，所以在 2023 年的巨大不确定面前，永远不要放弃希望，不停地提高自己技术能力和管理能力，只要技术过硬，总是会找到下一步的机会。

主持人：对于 2023 年，作为一个技术的管理者，您对您自己而言，你有什么规划和打算吗?

嘉宾：

1. 最重要是活下去，产品和商业模式都要活下去。
2. 保持一个好心态。把时间放短，人们看到的都很悲观，而把时间放长，其实看到更多就是乐观了。所以不要太焦虑，顺势而为。

3. 把时间更多的投入到生活和家庭中。因为你无法控制经济走向和工作好坏，但是你可以很大程度上掌控和你家人共处时间的质量。而我一直坚信家庭才是我们前进的力量和内心确定感的最重要来源。

主持人：团队中的程序员水平基本是正态分布的，对于初级的程序员如何去管理，老师有什么建议吗?

嘉宾：我想说的一个观点是：其实每个人都是人才(talent)，但是都需要一个合适的舞台去展示。而一个好的技术管理者会尽可能给每个人找到/定制一个适合他/她的舞台，而在我过往的经验中，有很多次真的需要要给到他/她足够时间后，才会看到他/她真正的潜力。(当然这个时间一般也不会超过 2 年)。所以这里的关键是基于工程师这个群体大部分人自驱力比较高的前提下，你和你的组织有没有足够的耐心给员工提供足够的空间/时间去展现他们/她们的实力。