蚂蚁金服 ANT FINANCIAL 阿里云 开发者社区

双11背后支付宝技术升级战

支付宝新技术最佳演练

了解更多蚂蚁金服技术，欢迎关注蚂蚁金服技术新媒体矩阵

公众号：蚂蚁金服科技

抖音号：支付宝技术

微博：蚂蚁金服科技

知乎：支付宝科技局

目录

万字长文丨1分36秒，100亿，支付宝技术双11答卷：没有不可能

作者：蚂蚁金服科技

2019年双11来了。1分36秒100亿，5分25秒超过300亿，12分49秒超500亿……如果没有双11，中国的互联网技术要发展到今天的水平，或许要再多花20年。

从双11诞生至今的11年里，有一个场景始终在支付宝技术团队之中循环往复——每一年确定目标时，大家都将信将疑，或惊呼或腹诽：“不可能！太夸张了吧！”但每一年的夸张目标，到最后都能奇迹般地成为现实。

前一年需要拼命跃起才能够到的果实，后一年就会成为再普通不过的日常。不知不觉之间，双11已经从最初启航时的小船，成为了承载数十亿人快乐和梦想的巨舰。

在这个举世瞩目的“奇迹工程”背后，是技术和技术人一起，十余年如一日，以难以置信的“中国速度”在不知疲倦地向前奔跑。

技术人的初衷往往极致单纯——既然决定要做，那就全力以赴，一往无前，但当他们一步一个脚印风雨兼程地走来，蓦然回首，就发现奇迹已经在那里了。

那时距离宕机只有几十秒

2009 年 11 月 11 日，对于支付宝工程师陈亮而言，本来是与往常没有任何不同的一天。

那年还没有支付宝大楼，更没有 Z 空间，他趟过早高峰的车流，坐到华星时代广场的工位上时，一封来自 CTO 程立的邮件发到了他的电脑里：今天淘宝商城要搞一个促销活动，预估交易量比较大，大家盯着点系统。

陈亮当时所在的团队主要职责是保障整个系统的稳定可靠。在促销活动的场景下，通俗地说来，就是要保障服务器“坚挺”，别被蜂拥而来的用户挤爆了。

淘宝商城在 2009 年的 8 月刚刚重组上线，日均交易量对于当时的支付宝而言，要稳稳接住不在话下。就算搞促销临时出现洪峰，不怕，扩容就好。

事实上也就是这么操作的。全团队的同学聚在办公室里，盯着电脑屏幕，只要发现交易量逼近系统承载上限，就马上进行扩容。

交易量的上涨有点不寻常，一片安静的办公室里本来只有键盘声，忽然有人高喊一声：“我秒到了！”紧接着又有人跟着喊：“我也秒到了！”办公室里嗡地一下，热闹了起来。

原来有人很好奇淘宝商城究竟在做什么促销让交易量涨成这样，点过去一看，发现有折扣高达 50% 以上的“秒杀”，忍不住出手一试。

“已经想不起当时究竟秒到了什么，只记得大家都特别快乐。”陈亮说。

快乐，是陈亮对于这个促销活动最初、也最鲜明的印象。

不过除了一整天都忙于扩容的同学之外，当时支付宝的大多数人都对这场促销并无感知。“事后才知道前一天有促销，同事说流量有点猛。”现在已成为蚂蚁金服研究员的李俊奎说，运维负责人很紧张地在第二天的复盘会议上提出“抗议”：“淘宝商城那边在搞什么？支付量一下子提升了这么多，万一我们提前准备的量不够，就危险了。”

淘宝商城搞了什么？站在今天回头去看，他们只是搞了一件不算很大的事：在“光棍节”当天，联合 27 个品牌做了一场促销活动，单日 GMV 5000 万。

当时没有任何人能够预计这个促销活动日后会成长为什么模样，不过支付宝从数据的增长之中嗅到了山雨欲来的气息：这个活动带来的交易峰值超过平日的 5 倍，虽然这次平稳过关，但已经逼近了当时支付宝的承载极限。

2010 年的年中刚过，支付宝就去跟淘宝商城通气：去年那个促销，今年还搞吗？淘宝商城说，搞。

好汉不打无准备之仗，如何筹备“双 11”被提上了支付宝每周稳定性会议的议程。首当其冲的是要准备充足的容量。但是按多少准备呢？谁都没经验。

“拍脑袋估个数据，然后按预估数据乘以三去买机器，简单粗暴。”李俊奎直言不讳。

为了检验这样拍脑袋的决策行不行，他还和团队一起搞了个测试：通过手动更改配置，把多台机器上的流量导到一台机器上，测试一台机器的能接住多大的流量。“现在想起来，那就是压测最早的雏形。”

他们甚至准备了一个备用的工作联络群。当时还没有钉钉，工作群都搭在旺旺上，“万一旺旺服务器也出问题了，不能及时联络怎么办？”

筹备的时间虽不长，倒也方方面面都有兼顾，“但是不管事先做了怎样万全的准

备，每年总有意外发生。”金融核心技术部工程师赵尊奎说。他当年所在的团队是账务会计组，一举一动都关系到钱，丝毫不容有错。

意外真的来了。

11 日凌晨，促销活动刚开始不久，支付宝的账务数据库就容量告急。

病来如山倒。发现问题时，状况已经十分危急，“只能再撑几分钟！”运维心急如焚，如果不能马上找到解决办法，支付宝就面临宕机风险，交易链路一断，谁也买不成。

怎么办？运维把心一横，说，砍了会计系统吧，给核心的账务系统腾空间。

时间已经容不得多加斟酌，一群高管站在背后，支付宝中间件团队的工程师蒋涛感到前所未有地紧张，“操作的时候手都在抖。”

这个当机立断的决策将支付宝从距离宕机只差几十秒的悬崖边挽救了回来。事后的数据显示，2010 年的双 11，参与用户达到 2100 万，总 GMV 达到 10 亿，是上一年的 20 倍，这是任何人都很难在事先预估到的涨幅。

“能想到会涨，但谁也想不到涨势会这么猛烈。”赵尊奎说，“也就是从那年起，我们开始隐隐觉得，更猛烈的还在后头。”

代码的力量

85 后的肖涵和 90 后的郑洋飞，都是在读大学时就知道了双 11。

肖涵喜欢网购，09 年就成了第一批尝鲜双 11 的剁手族，还在一个技术交流群里认识了参与过双 11 的支付宝工程师；郑洋飞常买《电脑报》，那上面说 2010 年双 11 一天的销售额等于香港一天的零售总额，他一边惊叹，一边心生向往。

“觉得好牛 B，想进去看看。”

当年互不相识的两个年轻人，不约而同地产生了这样的想法。

肖涵在 2011 年加入了支付宝，那一年支付宝已经开启了“上半年搞建设、下半年搞大促”的模式，筹备工作从 5、6 月起就着手进行，他刚一入职，就被调去开发流量接入和调拨系统 spanner。

这个系统相当于支付宝交易链路的第一道门户，“好比餐厅上菜的推车。一般餐厅，一个服务员只能每次上一盘菜，但双 11 的挑战，就是要让一位服务员同时能上十盘菜，因此我们需要一个推车。不过业界没有现成的推车能满足支付宝的需求，我们得自己造。”

差不多一整年的时间中，肖涵和团队为这个项目废寝忘食，spanner 终于在 2012 年的双 11 迎来了第一次大考。

谁曾想，意外又发生了。

那一年支付宝的大促监控系统也已经上线，流量曲线能够秒级实时显示，零点将近时，所有人都紧盯着屏幕，翘首以盼。

——零点一到，流量进来了，曲线开始增长，形成很漂亮的弧度，所有人开始欢呼，但是忽然，它跌了下去，然后开始像心电图那样抖动。

监控系统没有问题，也没有报错，为什么流量会进不来呢?

一石激起千层浪。他所能看到的抖动，同样实时显示在了淘宝的作战指挥室里。支付宝工程师贺岩正作为支付宝的唯一“代表”，在那里和淘宝的技术同学一起备战。这是个极其考验心理承受能力的工作，在支付曲线发生抖动的时刻，“淘宝的技术同学们一下子就把我围在了中间。连问‘支付宝出什么事了’?”贺岩回忆道。

肖涵脑子里一片空白，唯一的念头是，“不能让交易停下。”

0:00 到 0:20，短短的 20 分钟里，10 分钟用来定位问题，10 分钟用来解决问题；同样在这短短的 20 分钟里，外面已经天翻地覆：“‘支付宝不能付款了’登上了微

博热搜，家人、亲戚、朋友都给我打电话问是什么情况，手机都要被打爆了。”

关掉了一个健康监控模块之后，系统终于恢复了稳定。比起紧张，肖涵感到的更是前所未有的震撼：原来自己所做的事已经和千万人息息相关，每一点微小的疏漏，所影响的都是难以估量的庞大群体。

“没有身在其中过，就很难意识到自己敲下的每一行代码有着怎样的分量。”郑洋飞说。他在 2013 年加入支付宝实习，带他的师兄巩杰说了一句令他印象极深的话：你看看那些客服 mm，你敲代码时仔细一点，少出一个错，她们就不知能少接多少个报错电话。

架构革命

跨过了 2012 年的坎儿，DBA 就再三给出警告：扩容已经到头了，顶多再撑几个月，按这个增速，如果不想点别的办法，肯定坚持不到明年双 11。

祸不单行。另外的“紧箍咒”也接连落下：Oracle 数据库的连接数上限成为扩容的瓶颈，更要命的是，由于机房的一再扩容，杭州的电力已不足以支撑。有时候为了保机房供电，“大夏天的，办公室都会停电，不得不运冰块过来降温。”巩杰苦笑着说，杭州的盛夏，谁过谁知道。

治标的方法快要山穷水尽，必须要从治本的角度出发寻找新的解决方案，比如，从架构层面“搞革命”，做单元化。

革命不是请客吃饭，要从架构层面做根本性的调整，举步维艰：一来没有任何成功经验可以借鉴，只能摸索着走；二来牵涉到众多部门，大家需求不同，意见难免相左；三来，既然要革命，那目光必须放得更加长远，不能只是为了解决今年或明年的问题，至少也要做未来三年的规划。

与此同时，在和淘宝商城——现在叫天猫了——沟通之后，支付宝毫不意外地定下了又一个令人惊呼“不可能”的目标：支付峰值每秒 2 万笔。

事关重大，人人都很谨慎。“光是架构调整的方案就讨论了很久。”陈亮说，作为项目的架构师，他费了不知多少口舌去说服所有人都认同这个方案。

重担的一头落在他的肩上，另一头则交给了 2010 年抖着手化解危机的蒋涛，蒋涛更愁稳定性问题：“做技术架构变更的同时还得稳住业务，这件事非常复杂，技术风险也很高。”

留给他们的时间也不多了。LDC 架构的立项已是 2012 年年底，距离 2013 年的双 11 不足一年，对于这样浩大的工程来说，就一个字，紧。

陈亮最初构想了一个宏大的体系，要把所有系统都一口气单元化，但这个方案被程立否了：“主要问题在淘宝的交易上，先把淘宝做了。”按他的意思，哪怕先做第一期，2013 年也必须上线。

一堆不可能的目标聚集在了一起。但目标既然定了，就只剩向前这唯一的方向。

“立项之后，我们几乎每个月都做发布。”蒋涛说，这个频率是一般项目开发的好几倍，但即便如此，直到双 11 之前半个月，整套系统才算部署完成，小错仍然不断，不过，随着越来越多的小问题和被发现和修正，他终于感到，“心里总算慢慢有点底气了”。

2013 年，支付宝 LDC 架构首次在双 11 亮相，支付宝也第一次派“代表”前往双 11 的总指挥室——阿里巴巴西溪园区的“光明顶”。

这位“幸运”的代表就是李俊奎。“我就是去当‘炮灰’的。”他笑称自己一走进光明顶就感受到了热烈的压力。当年的总指挥李津当着全集团几百位工程师的面，指着大屏幕点名喊他：“向秀（李俊奎的花名）！你看看支付宝!”

这项压力山大的任务，李俊奎连做了好几年，乃至于做出了经验。“首先是不要慌，无论接到什么反馈，先说‘知道了，我看看’。因为你一个人在现场其实什么也做不了，你的职责其实是传达，以最快的速度，把问题传达给后方的伙伴，然后，相信他们。”

他说这是支付宝技术团队的重要制胜秘诀之一：你永远都不是一个人在战斗，你也无法一个人战斗，但你的身后永远有最靠谱的伙伴。

至于这一年的战果如何，按蒋涛的话说，“硬扛过去了”。新架构有惊无险，走出了第一步。

关公、灵隐寺和压测

2013 年双 11 的另一个特殊之处是，支付宝的备战室里多出来一幅关老爷的挂画。

挂画是郑洋飞“请”来的，不过“拜关公”作为支付宝技术团队的一项传统，早在他入职之前就由来已久。源头据说要追溯到支付宝建立之初，每到重要的系统更新时，工程师们就会在旺旺群里转发关公表情包，以求更新顺利，“别出 bug”。

隔了一年之后，关公像“升级”了，有同学去西安校招时看到了关公的皮影艺术品，就“请”了一个回来放在备战室。后来，程立买了一尊木质关公像放过来，去

年，副 CTO 胡喜又买了一尊关公铜像。

除了拜关公，去寺庙烧香也是例行项目，视目的地不同，还分为“灵隐寺派”和“法喜寺派”两大派别。至于哪边灵验，说法不一，但据观察，每年双 11 过后，程立、胡喜就会亲自率队上山还愿，从支付宝大楼一路步行到上天竺法喜寺，回来的途中，还会沿途捡垃圾做公益。

技术是纯粹的科学。技术人难道真的相信求神拜佛能避免系统故障和 bug 吗?

“心理上，我觉得还是挺有用的。”陈亮说，“主要是表达对于不可预知之物的一种敬畏。虽然我们已经做了多年技术，但技术的道路上还是充满了很多不可预知的东西。”

不可预知，构成了工程师们每年面对双 11 最大的焦虑感来源。

他们用各自的办法缓解双 11 迫近的压力。有人是运动派，用跑步或打球放空大脑，有人是“强迫症”派，一遍又一遍地 check 代码才能安心，还有人是“吃货”

派，迎战之前必定要先组团去吃海底捞。

全程参加了过去 11 年全部双 11 的赵尊奎，在被问到“哪年最不好搞”时，秒答曰：“哪年都不好搞。”同样“全勤”的陈亮则表示：“14 年之前，我们对于双 11 零点的信心，如果非要说一个数字的话，60% 吧。”

但他很快补充：“不过 2014 年之后，这个数字就变成 95% 了。”

陈亮的信心，来自于当年支付宝压测体系的建立。这一次不再是手动调配置测单机了，而是创建出仿真环境让系统去跑，提前找出系统的问题并及时修复，以免在正式战场被打个措手不及。

“压测让双 11 开始从一个不确定的事逐渐变成确定的事，它极大地改变了我们对于双 11 稳定性的保障方式。”有“压测小王子”之称的郑洋飞说。

虽然 2014 年的压测仅覆盖核心系统，但这个体系已经帮了大忙。在双 11 之前的一个月里，它至少让一百多个致命的问题提前暴露出来。“如果其中有一个没有修复，我们 2014 年的双 11 肯定就挂了。”陈亮说。

1%？或 10%？

压测这一“压”，既压出很多隐患，也压出了一个大问题：支付宝所用的 Oracle 数据库在压测之中“抖”了起来，性能眼见得触到了天花板。

2014 正是移动互联网大爆发的年份。指数增长的移动支付比例势必带来比往年更汹涌的流量峰值，Oracle 肉眼可见地支撑不住了。

再买服务器？成本吃不消，而且为了应对峰值而增添的机器，平日里没有用武之地，完全是资源的浪费。

还有没有别的办法？有的。阿里自研的分布式数据库 OceanBase，从淘宝被划到支付宝后，已经沉寂了两年，正在焦急地寻找一展身手的舞台。

但是一听说是自研的数据库，业务满脸都是狐疑。跟交易和金额直接相关的数据库，只要错一个数据，后果就不堪设想，别说双 11 这么大的流量，即使平日，要不要用这个没经过验证的产品，也颇要斟酌一番。

先切 1% 的流量给 OceanBase 试试吧。这是大家争论了好一阵后得出的方案。

但是 Oracle 在压测中的表现显示，缺口不止 1%，而是 10%。

OceanBase 说，我们来承接这 10%。

10%，听起来不多，但双 11 的 10%，相当于平日里的最高峰值。如果 OceanBase 能平安无事地接住这 10%，就意味着它可以担起支撑支付宝日常运行的重任。

OceanBase 必须证明自己有这样的能力。“我们找了淘宝的同学，协调了很多资源做了一个测试，主要校验淘宝订单的金额和支付宝交易金额是否能吻合。”DBA 团队的工程师师文汇说，当时的方案很谨慎，如果 OceanBase 出现了问题，随时都可以切回来。

测试结果，OceanBase 没有错漏一个数据。程立当即拍了板：10% 都切给你们。

这个决定成就了 OceanBase 在双 11 的首秀，“相当于 Oracle 和鲁肃（程立的花名）都帮了我们一把。”师文汇笑着说。

这时距离 2014 年的双 11，时间已经不足两周。可靠性虽然经受住了考验，但 OceanBase 毕竟是个诞生才四年的年轻数据库，小问题层出不穷，比如响应时间长达 10 毫秒，比 Oracle 差了好几个数量级。最后十来天，师文汇和全团队的同学一起，硬是把它优化到了 1 毫秒以下。

“做了这么些年，对它的容量和性能还是比较有信心的。”师文汇说。

他说得轻描淡写，但在这个曾经面临团队解散项目取消的产品身上，他和整个团

队一起倾注了多少心血，除了他们自己之外，谁也说不清楚。

OceanBase 最初并不是为双 11 而做的，但在双 11 这个舞台上，它第一次获得了聚光灯下的位置，并且表现卓越，从此，支付宝开启了核心交易系统完全搬迁上 OceanBase 的进程。

到今年，OceanBase 对内 100% 承载蚂蚁业务的流量。对外，在被誉为“数据库领域世界杯”的 TPC-C 基准测试中，打破了由美国公司 Oracle(甲骨文) 保持了 9 年之久的世界记录，成为首个登顶该榜单的中国数据库产品。

【视频：使命必达——OceanBase 登顶 TPC-C 测试】

我赢了一只 apple watch

2015 年，李俊奎去拜访了上海证券交易所，那里的交易系统部署在 6 台大型计算机上，交易峰值能达到每秒 10 万笔。

他大为惊叹：10 万笔！何等高不可攀的数字！什么时候支付宝也能达到就好了！

回到杭州，他马上与同学们分享了这次见闻，结果同学们告诉他说，今年我们的目标就要超过每秒 10 万笔了。

李俊奎一想，这种一听就不可能的目标，是支付宝的作风，没毛病。

与此同时，郑洋飞则在为这个目标头痛不已。他刚刚跟他的主管打了一个赌，赌的是他作为 2015 年双 11 全链路压测的负责人，能不能保障双 11 的支付不出任何问题。赌注是一只 apple watch。

这一年是 90 后的郑洋飞第一次挑大梁，从双 11 的参与者转换角色成为一个项目的主导者。但这一年也是他和团队都“忍辱负重”的一年，上半年，因为频繁不断的可用率问题，他们做稳定性的团队也在频繁遭受打击，士气不振，不少同学选择了离开，内外的质疑声也接连不断，那几个月，空气里都仿佛写满了“难熬”二字。

“当时团队没几个人了，但是人人都憋着一口气，想着一定要把双 11 这个事情搞好。”郑洋飞说，“就是不想让人觉得支付宝不行。”

局面有如背水一战，如果失败了，想要“翻盘雪耻”就要再等一年。因为双 11 每年只有一次，不仅是一年一度的大考，更是一年一度的舞台。按照系统部资深技术专家杨海悌的说法，“人人都想把自己一年的努力拿到双 11 去验证和展示，不让上还不高兴。”

跟 2014 年的全链路压测比起来，2015 年主要要做几个方面大刀阔斧的改进：一是要从核心系统扩展到全部系统，二是要平台化，也就是打造一个全链路压测的平台工具，三是要跟整个集团的压测打通联动。

“老实说，非常忐忑。”郑洋飞心里没底。

当双 11 零点的洪峰扑面而来时，他已经忘掉了 apple watch 这回事。有一个压测没验证到的数据库定时任务，让曲线看上去不那么平滑。“稳定压倒一切”的宗旨之下，系统只要一抖，整个团队的心也跟着抖了起来。迅速排查的结果是，系统整体没出问题，但压测遗漏的一些细节，让结果不是那么完美。

“曲线不是特别好看。”他不无遗憾地说。

郑洋飞最终赢得了这只 apple watch，但对于他而言，除了奖品之外，这只 apple watch 更有了别样的意义，时刻提醒他即使做了再充分的准备，也没有万无一失。

对“丝般顺滑”的追求永无止境

其实每一位支付宝工程师的心中，都有一条的“完美曲线”。

理想之中，它应该是这样的：双 11 零点，洪峰来了，曲线漂亮地攀升，没有骤升骤降，不要用频繁的抖动去折磨大家脆弱的神经。

如果要浓缩成一个词，那就是“丝般顺滑”。

但是，每一次为此而做的技术演进和架构变更进行到一定阶段，“你都会发现一开始可能设想得非常美好，但到了一定的规模之后，挑战就接二连三地来了。”杨海悌感叹道，“量变产生质变，这句话不是虚的。”

双 11 的“量”，早已一骑绝尘地进入前所未有的领域，2016 年双 11 仅用了 6 个多小时，交易额就已超过 2014 年全天。这些年以来，都是自己在不断刷新自己的纪录。

在这样的量之下保障稳定，难度不止提高了一个数量级。

还记得 2012 年底制定的架构革命之“三年计划”吗？它真的持续了三年，这场最初是为了解决数据库连接数和机房限制而进行的架构革命，在三年的演进过程中，又衍生出很多其他的架构，比如异地多活、容灾，弹性的容量调度等，直到 2016 年，才算全部落地。这之中每一步的演进，都是为了让系统具备动态扩容能力，能够顺滑地进行弹性的扩展和伸缩。

“大”是考验，“小”也是考验。

师文汇体会最深刻的瞬间，不是 2014 年 OceanBase 一举成名的时刻，而是 2016 年一次小小的测试。在这个测试里，他发现了一个指标有一点异常——非常不起眼，2 毫秒的偏差。

“2 毫秒而已，如果在别的地方，很可能就会被判断为无关紧要，然后漏过了。”但他的团队中的一位小伙伴，非常认真地去检查了这个问题——万幸如此。后来事实证明如果不解决这个问题，那年的双 11 就会有大麻烦。

“即使资源不足、时间紧张、软件有各种不完善的地方，但我们的小伙伴不会放过任何一个问题。”师文汇感慨。

早些年，流量曾是实现完美曲线最主要的挑战，但是越到后来，随着业务的不断拓展，工程师们越来越清楚地认识到，稳定压倒一切不假，但技术更要着眼于未来。

2017 年，是贺岩加入支付宝的第九年。这一年支付宝实现了离线在线混布，离线任务的大量闲置资源可以被用于在线任务，从而大大提升了资源的利用率。

“在一些小的场景中，这种效率提升能带来的节约可能不那么突出，但在我们这样的体量上，它所带来的就是不可估量的一整个未来。”贺岩说。

有了前人积淀这么多年的基础，面向未来的路，就越走越顺利了起来。

2018 年双 11，支付宝其实保障了两个大促，天猫的大促，和支付宝自己的“码上双 11”等玩法。这一年的故障数比前一年下降了 70-80%，首次实现大促全天平稳。

大队长李铮非常淡定：“说白了，就是我们把各种风险通过系统化或工程化的流程，控制得比较好。峰值出现的过程，也都在我们的预期之内。”

2019，则是双 11 的“云原生”元年。

如果说技术是像叠积木那样一层一层累积起来的，那云原生就是最下面的基础，打好了这层基础，上层的应用就像是站在了巨人的肩膀上，生来就具备了一系列强大的能力。业务无需再过多地担忧技术问题，只需要专注于业务代码即可。

点亮全世界

故事讲到这里，不知大家是否还记得，当年因为每秒两万笔的峰值目标而惊呼“不可能”的同学们。

当年，每秒两万笔是他们举全体之力奋斗半年才能冲上的高峰，而去年，每秒两万笔已经成为支付宝再日常不过的状况，随随便便，一秒钟的事。

这样的巨变真实发生了，只是身处当时当地，谁也没有想那么多。几乎每一位工程师都表示：“每年搞完双 11，下一年的目标就出来了，然后我们就为着下一年的目标去进行相应的准备和努力。”

一个目标，又一个目标，在征服一个又一个“不可能”的过程中，曾经以为遥不可及的标高，都一一被甩到身后。当年高不可攀的每秒 10 万笔，今天看来不过小菜一碟。

只有当回头的时候才有所感觉，在某一天忽然发现，原来已经走出了那么远，原来已经攀登到了那么高的地方。

而当年那些惊呼不可能却又拼命将不可能变成现实的年轻人，已经纷纷长大，他们现在有着更多的从容淡定，上限在哪里没人知道，更大的可能是，没有上限。

流量数据的增长也早已不是双 11 技术保障的全部。更多复杂的业务和玩法，在技术的成果之中生长起来，反过来又为技术的发展提供动力。走出双 11，它们还能走入很多的场景：新年红包、五福集卡……

——或者走出支付宝和阿里巴巴。

那些由支付宝的工程师们创下的奇迹，正一一变成产品，服务更多的金融机构。

至今已有数十家银行和金融机构用上了 OceanBase，压测平台、云原生等技术也纷纷走向产品化，支付宝通过历年双 11 沉淀下的技术和经验，正在拉动整个中国的互联网金融科技一起飞奔。

——或者走向全世界。

双 11 早已不仅是中国的双 11，而是成为了一场全球的狂欢。与双 11 一起，技术也在走向全世界。

“不过要是说到我们的理想，那就是将来某一年双 11，整个备战室空空荡荡，除了关公像之外，不需要有任何同学留守，智能化的系统能搞定一切问题，而我们只需要捧着茶杯或喝着酒，看着丝般顺滑的曲线。”

对于巩杰所展望的这种未来，现场就有同学笑了。“不可能吧！?”每年双 11 都如同打仗一般，救兵如救火。

但是谁说不可能呢？毕竟，他们是那样一群已经把太多的不可能变为现实的人。

支付宝双十一支付峰值背后的技术

作者：蚂蚁金服科技

和过去 10 年一样，2019 年天猫双 11 又创造了一个全新的纪录。

这个数字背后，是数代支付宝工程师们殚精竭虑、不断突破技术难关。

今年双 11 之前，小编邀请到 11 位经历双 11 的技术同学口述实录，特别筹备了纪录片《一心一役》，讲述这一路走来的那些隐秘往事。

对于技术人员来说，维持双 11 全天 24 小时稳定流畅固然不易，但最为考验的时刻当属零点刚过，人们操起手机，刷新早已存好的购物车，点击支付的那一刻！

11 年，零点越来越平滑的双 11 购物背后，支付宝有过哪些不为人知的技术探索，今天也特别放送。

从外部瓶颈说起

事情从一开始就显得不是很顺利。

2011 年的双十一，在高峰时期少数用户无法付款，经过调查发现，这是因为少数银行的网银系统在压力下出现故障。早年的支付宝交易，用户点击支付后需要从支付宝和银行的接口去付款，而早年这个接口的性能很差，每秒只能支持几十到上百笔交易，稳定性也比较差，一旦流量上来，容易发生故障。

如果不解决这个问题，今后的每次大促都会出现无法付款的情况，极大影响用户体验。但是，这个问题单靠技术是很难解决的，银行对网银系统的演进有自己的规划，支付宝无法去干涉它们的系统。

不过，聪明的运营人员想出了一个变通的办法。在 2012 年的双十一，支付宝通过活动吸引用户先充值后付款，让用户先将钱充值到支付宝余额上，到双十一直接从余额里面扣款就行，这样，外部的瓶颈就被转换到内部了。这样做效果非常显著，付款失败的问题大为缓解。

然而，外部的瓶颈始终存在，面对每年翻倍提升的流量峰值，支付对外部的依赖始终是一个隐患，不知道什么时候就会爆发。

解决这个问题最好的办法，就是不通过网银，让资金在内部的系统中流转，先充值后付款就是这个原理。那么，有没有一个方法，吸引用户把钱放到支付宝里呢？2013 年 6 月，支付宝推出余额宝，歪打正着的解决了这个问题，到 2014 年底余额宝就吸引了 1.85 亿用户，在 13 年和 14 年的双十一，交易峰值也分别实现了 4 倍和 3 倍的增长。

2018 年 5 月，支付宝接入网联清算平台，同时在这些年里，银行也在大力提升自己的系统能力，中大型银行的网银系统支持的交易笔数已经达到 2 万笔 / 秒以上，外部问题基本得以解决。

解决了外部瓶颈之后，支付峰值的数字能有多高，就看支付宝的系统如何化解一

年比一年更凶猛的流量洪峰。

容量规划：三军未动粮草先行

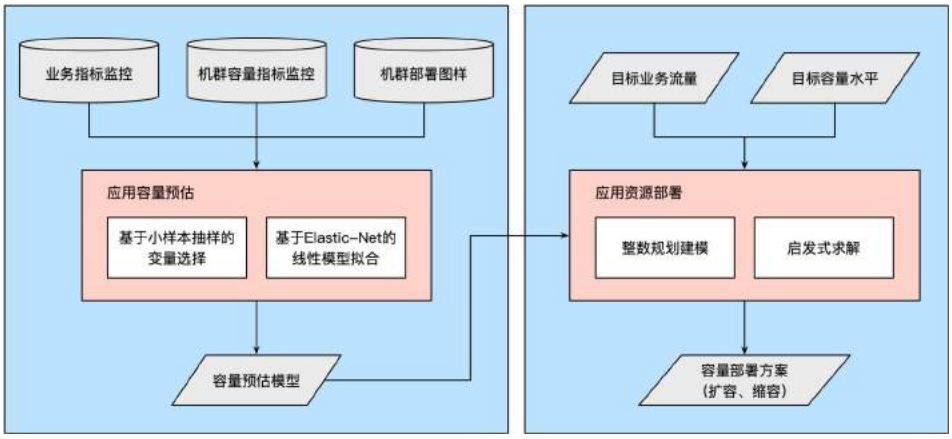
事实上，支持交易笔数峰值面临的首要问题，并不是设计一个完美支持横向扩展的架构，而是对可能的流量峰值进行准确估计，然后安排对应的机器和资源。如果不做估计，可能发生两种情况：预备资源过多，架构过度设计，造成资源浪费；预备资源过少，无法完美支持大促，造成部分支付排队或失败。 每年双十一备战，负责大促的决策团队会根据历史数据、大促目标来拟定一个交易数值，然后将这个数值拆解为各个系统所需要应对的流量，从而进行系统容量规划。

双 11 大促的场景指标一般包括交易创建数、收银台展现数、交易支付数。总的支付目标数已经有了，运维人员根据总 tps/ 单机 tps 的算法计算出应用在每个指标下的单机能力，然后，参考历史活动数据，可以计算应用在不同场景链路下的单机 tps。

但是，这种做法人工干预较多，对于各个应用的容量预估的粒度比较粗，后来，支付宝又建设了容量分析平台，可以进行自动化的细粒度的容量分析。

它的原理是，如果我们把一个链路理解为一个业务，链路根节点可以理解为业务的源头流量请求，每个链路上的节点（这里的节点包括应用、DB、tair 等）都能计算出该节点调用次数相对于根节点流量的系数。因此，当业务源头的 QPS 确定时，就可以基于链路数据，计算出每个节点的 QPS。

2018 年的双十一，支付宝还建设了智能容量模型，不但可以根据业务流量进行容量预估，还可以智能的产出应用资源部署方案，使得在该方案下，部署单元在承载给定业务流量时的容量水平处于目标范围。



智能容量模型是支付宝对 AIOps 探索的一部分，也是对数据技术和人工智能在系统中落地实践的一部分，这方面也是当前支付宝技术探索的方向之一。

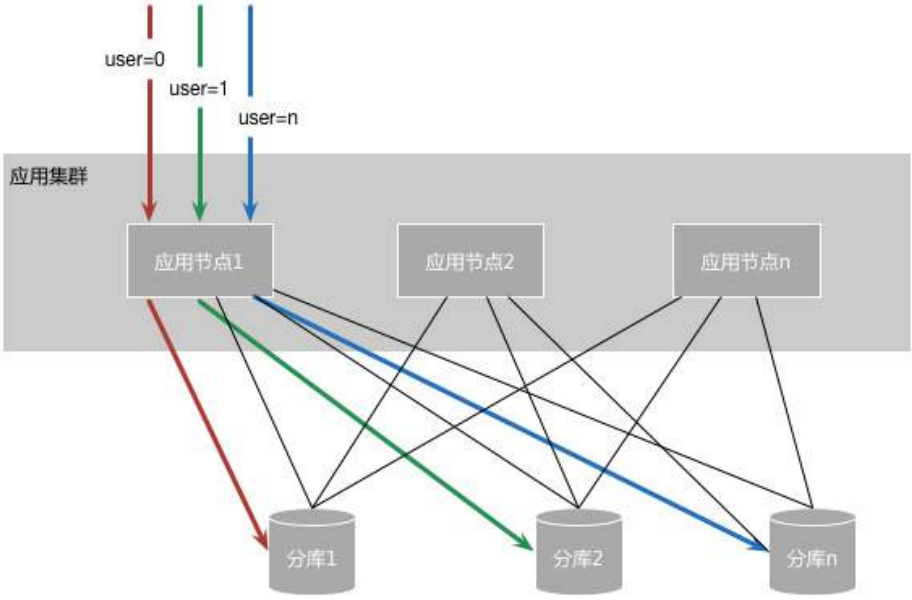
LDC 与弹性架构：大促最强武器

对流量进行预估并进行合理的容量规划之后，接下来就看我们的架构是否能支持流量峰值了。

首先需要说明的是，流量高峰涉及到一个系统的方方面面，支付宝的整个系统极其复杂，而且面向 toC 和 toB 都推出了很多业务，即使只关注核心支付系统，也包括支付清算、账务、核算等子系统。

系统部分组件由通用型的中间件提供支撑，如负载均衡中间件 LVS/Spanner、阿里巴巴的分布式缓存中间件 Tair 等，其它则由支付宝自研的 SOFAStack 金融级分布式中间件负责。

支付峰值的本质是一个高并发问题，互联网公司解决高并发的思路是横向扩展水平拆分，用分布式的方式来应对流量洪峰，支付宝也不例外。支付宝很早完成了服务化架构和核心数据库的水平拆分，成功应对了前几年的双十一。

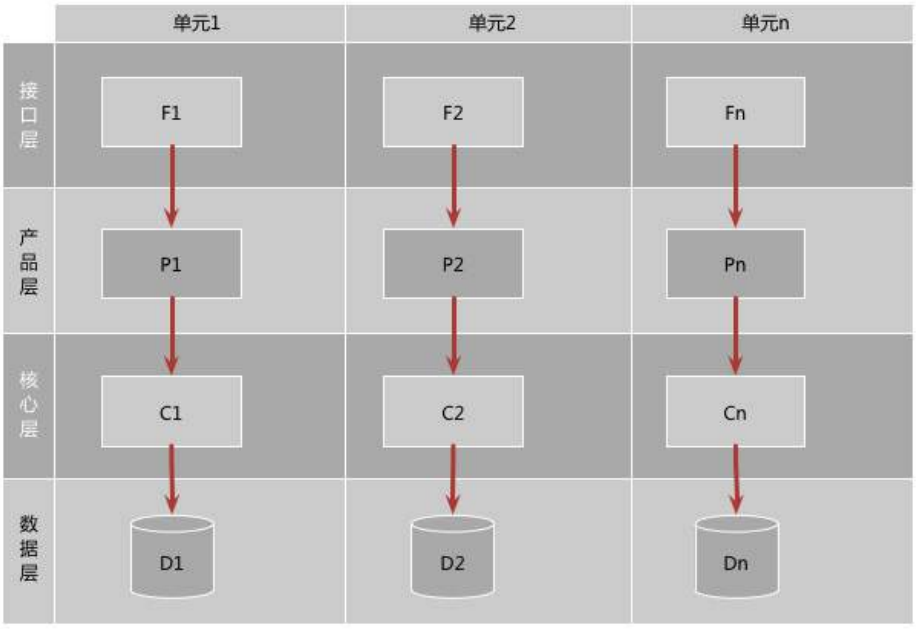


分布式系统示意图

这个架构的问题是，所有子应用都需要访问所有数据库分库，但是数据库连接是有限的。当时主流的商业数据库，连接都不是共享的，就是说一个事务必须独占一个连接。而连接却又是数据库非常宝贵的资源，不能无限增加。当时的支付宝，面临的问题是不能再对应用集群扩容，因为每加一台机器，就需要在每个数据分库上新增若干连接，而此时几个核心数据库的连接数已经到达上限。应用不能扩容，意味着支付宝系统的容量定格了，不能再有任何业务量增长，别说大促，很可能再过一段时间连日常业务也支撑不了了。

这个问题迫在眉睫，从 2013 年开始，支付宝开始新一轮的架构改造，实施单元化的 LDC 逻辑数据中心，双十一的流量峰值，终于还是成功的扛下来了。

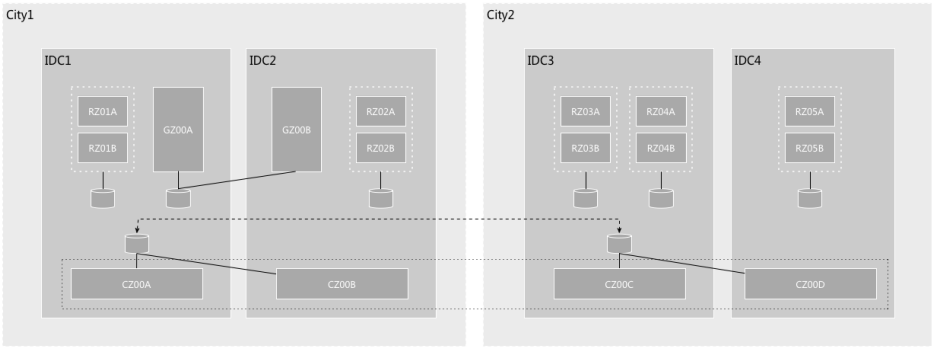
一个单元，是一个五脏俱全的缩小版整站，它是全能的，因为部署了所有应用；但它不是全量的，因为只能操作一部分数据。这样，只要将数据分区增加单元，就可以提升整个系统的处理性能上限。



单元化示意图

但是，并不是所有的数据都能拆分，比如部分底层数据是全局数据，所有单元的应用都需要访问。并且，支付宝经过近十年建设，有些架构也并不能很好的拆分成单元。在这个前提下，支付宝设计了 CRG 的单元化架构，既能利用单元化的优点，也能支持现有的架构。

- RZone（Region Zone）：最符合理论上单元定义的 zone，每个 RZone 都是自包含的，拥有自己的数据，能完成所有业务。
- GZone（Global Zone）：部署了不可拆分的数据和服务，这些数据或服务可能会被 RZone 依赖。GZone 在全局只有一组，数据仅有一份。
- CZone（City Zone）：同样部署了不可拆分的数据和服务，也会被 RZone 依赖。跟 GZone 不同的是，CZone 中的数据或服务会被 RZone 频繁访问，每一笔业务至少会访问一次；而 GZone 被 RZone 访问的频率则低的多。CZone 是为了解决异地延迟问题而特别设计的。



CRG 架构示意图

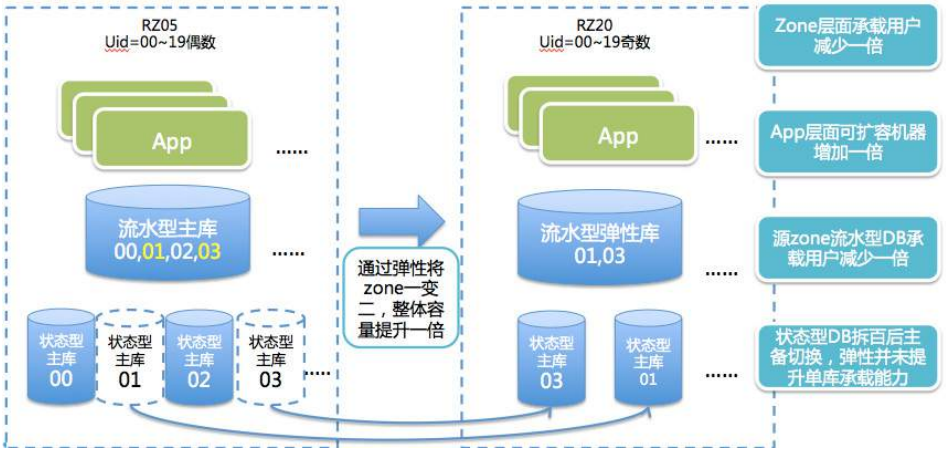
关于支付宝单元化和 LDC 的更多信息可查看这篇文章。

实施了 LDC 之后，系统容量实现水平扩展，顺利支持了 2013 年及之后的双十一流量洪峰，并且系统不再受到单点故障限制，经过完善之后还做到异地多活，最终形成了三地五中心的金融级架构。

理论上，只要无限扩展 LDC 的计算资源，就可以应对无限大的流量，但是，这样做的话，大部分机器只有在大促时才能派上用场，平时就是闲置的，造成资源浪费。最好能做到平时用少量资源支持常规流量，大促时经过容量规划，提前启用部分空闲或第三方资源应对高峰流量，这就是弹性架构的由来。

2016 年，支付宝开始为大促进行弹性架构的改造。弹性架构基于业务链路，因为大促时只有部分链路的流量激增，因此只需要针对大促关键链路进行弹性扩容即可。

弹性架构涉及到多个层面的改造，首先是弹性机房和弹性单元，需要在 LDC 逻辑机房架构上按照业务纬度继续切片，保证单片业务可以独立逻辑单元部署，并保持与非弹性单元的联通性，并且可随时弹出和回收。



其次是弹性存储，包括流水型数据和状态型数据的弹性。流水型数据包括支付订单，为了支持这些数据的弹性，创建了弹性位 + 弹性 UID，然后路由根据弹性 UID 将订单分配至弹性单元中进行处理。状态型存储比如用户的账户余额，进行整体弹出，具体实现方式是通过 DB 层的主备切换，将主库压力分流至备库。

然后是中间件层面的改造，包括路由、RPC、消息队列、流量管理等等。应用层面也需要进行相应的改造，因为每个弹性单元需要做到独立逻辑单元部署，因此需要从服务到数据进行梳理并剥离，同时添加弹性 id 等弹性逻辑处理。

除了这些之外，还需要对运维平台、压测工具进行相应的改造。

2016 年弹性架构上线后，成功支撑了当年双十一，满足大促要求和预定目标，节省了机房物理资源，成为应对大促类流量洪峰最有力的武器。

弹性架构里的弹性单元都是新增的集群，但其实还可以进一步的提高资源利用率。方法就是离在线混部技术，因为有些集群是用作离线的大数据分析，但并不是全天 24 小时都满负荷工作，当没有任务时，集群资源利用率极低。如果将离线的应用和在线的业务应用部署在一起，让大促高峰时段能够利用这些资源，就可以减少大促期间采购的资源，进一步节省成本。混部技术需要运维的分时调度配合，在不同的时段将资源分配给不同的应用。

从 2017 年起，支付宝开始尝试离在线混部和分时调度技术，在大促时利用离线技术所使用的集群资源，大大提升了集群资源利用率。

百万支付：解决数据库扩展瓶颈

2016 年的双十一，交易笔数峰值达到 12 万笔每秒，这场高并发之战仍在继续。 前面提到了很多应对大促的技术手段，但其实漏掉了一个最重要的部分，那就是数据库。在流量洪峰时，受到压力最大的就是数据库。这是因为，在前台我们看到是一个成功交易，但拆解之后，一个交易可能平均要产生数百甚至上千个请求，数据库的压力要远远大于我们所能看到的数字。

从最开始，数据库就一直是支付宝系统的瓶颈之一，在之前，其实已经配合架构改造对数据库做了诸多升级，除了上面提过的弹性化的改造，还包括：

1. **分库分表**，将原有的交易账户库分离为交易库和账户库，并通过分布式事务解决数据一致性问题。
2. **数据库水平拆分**，将所有的用户按照 1% 粒度分为 100 份，配合单元化的逻辑隔离。
3. **数据库读写分离、多点写入、数据复制**，通过这些方式，可以大大提升性能。

早年支付宝采用的商业数据库能进行的改进是有极限的，为了成本考虑，不可能为了一年仅仅几天的大促活动去采购额外的数据库系统和设备。

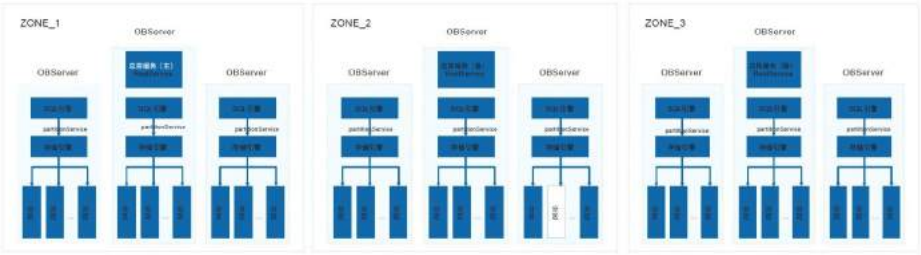
早在 2014 年的双十一，支付宝自研数据库 OceanBase 就开始承担 10% 双十一核心交易流量，随后一步步承担交易、支付、账务等核心系统的 100% 流量，经受住了极端条件下的严苛考验。

OceanBase 从第一天开始，就计划成为一个分布式的关系数据库，也就是天然支持大规模和高并发的场景。但是，支付宝本身的用户体量太大，再加上双十一所面临的的系统压力太大，到 2017 年双十一的时候，即使采用了额外的弹性库，数据库 CPU 压力也接近上限，成为继续扩容的瓶颈所在。

2018 年的双十一，支付宝在内部提出了百万支付架构，意思是这套架构可以支持百万笔 / 秒量级的系统压力。而这套架构的核心，就是 OceanBase 2.0 分布式分区方案。

过去架构下的 DB 扩展，由于 DB 单机存在极限，且一个 UID 最多对应一台机器，所以这里的扩展能力是通过 DB 新增集群，应用加数据源来实现的。这就会带来一系列的问题，比如应用的内存增长、多数据源导致弹出弹回费时费力、多个 DB 集群的日常维护成本高等。为解决这个问题，考虑让 DB 也能像应用一样可以动态扩容，且必须突破一个 UID 最多一台机器的限制，从而能达到应用和 DB 同步扩容，不用增加新 DB 集群就能达到新的容量支撑能力。

由此，基于 DB 的分区功能，将 DB 的扩展性大大增强，避免了必须增加集群来扩容的尴尬。同时对应用进行了相关的升级改造，如全站流水号架构的升级，系列中间件的改造，以及任务捞取场景的改造等。



OceanBase 分区架构

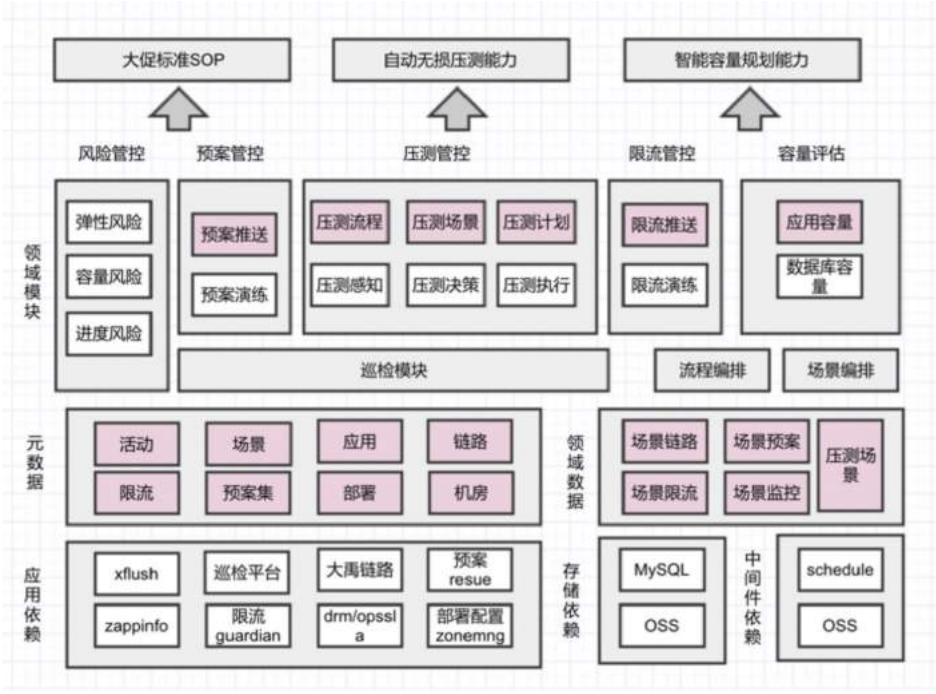
传统数据库弹性架构，将数据进行物理拆分到不同机器，业务在数据访问 / 研发 / 后期维护及数据配套设施上非常繁琐；同时拆分后资源很难快速回收，且数据拆分及聚合无法实现业务无损。相比于传统数据库的弹性架构，OceanBase 2.0 架构完全不侵入业务，内部通过分区实现数据分片的自组织及负载均衡，通过生成列及分区规则实现自动路由，通过分区聚合（partition_group）消除分布式事务性能开销以提升性能，从而实现无损线性伸缩。另数据分片间 share_nothing 的架构，实现分片故障隔离及单点故障消除的高可用架构。

2018 年双十一，OceanBase 2.0 成功上线，并支持了交易和支付的全部流量。并且，基于 OceanBase2.0 分区方案的这套架构可以轻松扩展到支持百万级交易，关于应对流量洪峰的战役暂时告一段落。

技术保障：大促技术标准化

双十一是新技术的演练场，那么怎么确定这些技术能有效支撑流量高峰呢？特别在支付宝，涉及到人们的资金安全，一旦发生问题后果极其严重，更是要慎之又慎。

2014 年，支付宝上线了全链路压测，成为系统化技术验证的神器；从 2017 年起，支付宝开始打造自动化和智能化的技术风险防控体系；2018 年的双十一，大促中控上线，大促相关的技术开始走向标准化。



大促中控示意图

大促中控也就是一站式的大促保障解决方案，它的目的，就是将之前大促的经验沉淀下来，形成套路和规范，最终向无人值守方向发展，搞大促不需要技术人再熬夜了。

有了大促中控，可以进行自动化的无损压测，线上压测能得到想要的结果的同时，不影响正在进行的业务。它的核心技术能力是对环境、机器、线程的隔离，以及在压测异常时的智能熔断。

压测并不是万能的，有些问题可能在压测中难以暴露，从 2018 年起，支付宝还展开了红蓝攻防演练，为了在大促峰值出现异常时，检查应急策略、组织保障、响应速度是否到位，以及验证新技术的稳定性是否达标。

对于大促中的资金安全，支付宝自研了实时的资金核对系统，实现峰值的资金安全实时验证，验证每一笔资金准确无误。

当所有技术准备就绪并不是就可以了，每次大促之前还有很多配置需要切换，一旦出错就会造成严重影响，因此支付宝打造了面向终态的技术风险巡检能力，在大促前一天进行成百上千的配置自动化检查，确认所有系统进入大促状态，确保万无一失。

随着时钟渐渐指向零点，大促一触即发。

未来可期，我们一路同行

总结起来，双十一流量峰值考验的是架构的可伸缩性、数据库的承载能力、运维的强大调度能力，以及完善的技术保障能力。为了确保大促顺利完成，需要做的技术准备也远远不只文中提到的，诸如全链路压测这样的幕后功臣还有很多，由于篇幅所限，这里就不再一一介绍了。

支付宝也在持续更新着自己的技术装备库。今年的双十一，支付宝也有几项新能力得到实战检验：OceanBase 2.2 上线，该版本在 TPC-C 基准测试中取得第一名，

平稳支撑了新大促；自研的 Service Mesh 首次登上大促舞台，目前已经 100% 覆盖支付宝核心支付链路，是业界最大的 Service Mesh 集群。

使命必达——OceanBase 登顶 TPC-C 测试

随着普惠金融的落地，以及万物互联的发展，支付平台面临的流量压力会进一步提升。现在我们看到的峰值，未来也许稀松平常；未来的峰值，也许比今天还要高几个量级。支付峰值这场战役仍会继续下去，其中的技术也将不断的更新进化，未来双十一的技术之战将更加精彩。

11 年天猫双 11 对支付宝技术有什么意义？

作者：蚂蚁金服科技

刚刚过去的双十一全天交易额 2684 亿，再创新高，支付宝自研分布式数据库 OceanBase 每秒处理峰值达到 6100 万次，再次展示世界级数字金融运算能力。

今年是支付宝参加双十一的第十一年，也是一个新的起点，我们将支付宝历年重要技术制作成一张大图，但图所包含的信息毕竟有限，本文将这些技术分为三个阶段，为你详细解读。

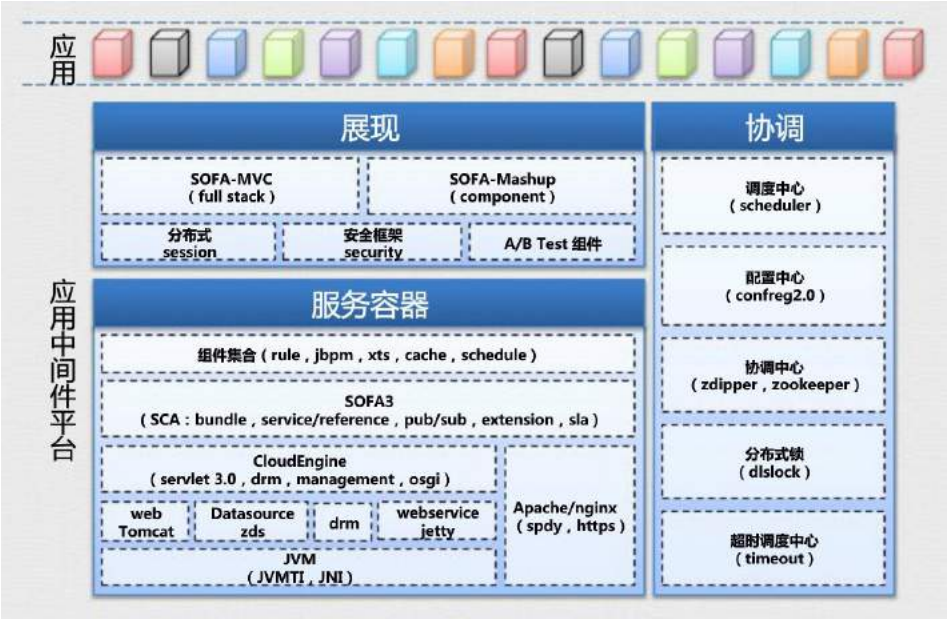
2009-2013：被大促拽着前进

2009 年光棍节，第一次双十一来的很突然，谁也没有想到在网民自创的“节日”搞的促销活动会成为中国最大的购物狂欢节，当天淘宝商城的成交额激增 10 倍。

支付宝第一年参加双十一，第一代对账系统完成大促全量资金核对，同时整体完成面向服务的架构

SOA 改造，虽然手忙脚乱，不过还是扛了下来。

在 2008 年以前，支付宝用的是阿里的 WebX 框架，采用 IOE 商业系统 + 开源软件的方式搭建后台，但这套架构是烟囱式架构，难以维护和扩展。2007 年开始，支付宝自研 SOFA 框架，并于 2008 年 1 月上线。



SOFA 分布式中间件早期架构

SOFA 最开始的目标是对后端架构进行分布式服务化改造，而双十一大促将这个进程大大加快，刚完成这项改造，团队又马不停蹄的投入下一项任务中。

2010 年第二次双十一，大家虽然有了准备，但实际的流量和成交额仍然远超估计，据当年的参与者表示，虽然准备了比平时多 3 倍的容量，但最后的流量是平时的 20 倍，在最后关头，运维人员将会计系统降级，避免核心系统崩溃。

在这一年里，支付宝数据库团队完成数据库的垂直拆分和水平扩展，将交易账户库分为交易库和账户库，奠定了弹性扩容的基础。但是，拆分带来了数据一致性问

题，中间件团队研发并上线了基于两阶段提交的分布式事务中间件 XTS，解决了这个问题。XTS 经过不断完善，现在已经进化为阿里巴巴和支付宝共建的 Seata 项目，并已经在 GitHub 上开源。

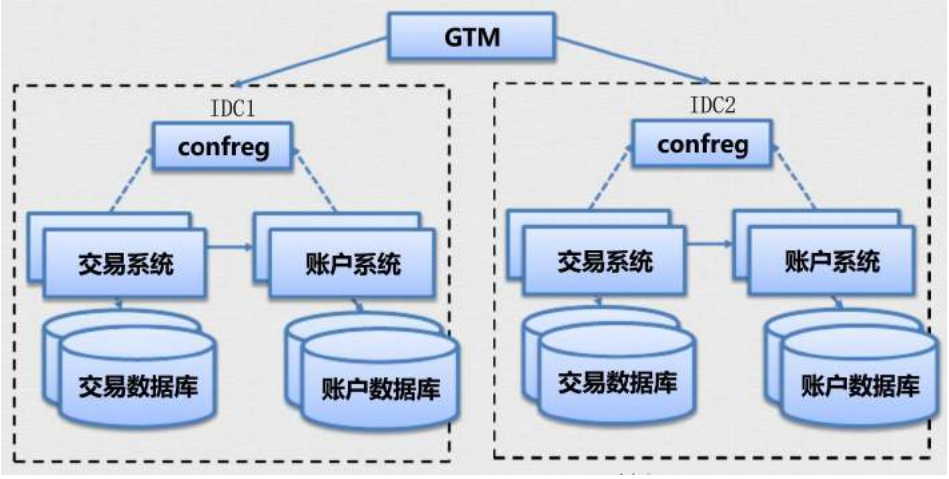
也正是在这一年，分布式关系数据库 OceanBase 开始立项，开始了技术长跑。

2011 年，支付宝运维监控系统 xflush 上线，实时秒级监控大促的峰值，业务健康度不再是黑盒，后来，升级为全功能监控平台 AntMonitor。

2012 年双十一单日交易一亿笔，在这背后，是支付宝紧急通过数据、网络、机房的改造，将单日的支付能力从百万级提升到了亿级。

当时，支付宝系统伸缩的瓶颈在基础设施和应用层面，首先是负责流量接入和调拨的负载均衡，支付宝自研了负载均衡中间件 Spanner，2012 年上线并成功支持大促，在不断升级改造后，到今天也仍在支撑日渐高涨的巨大流量。

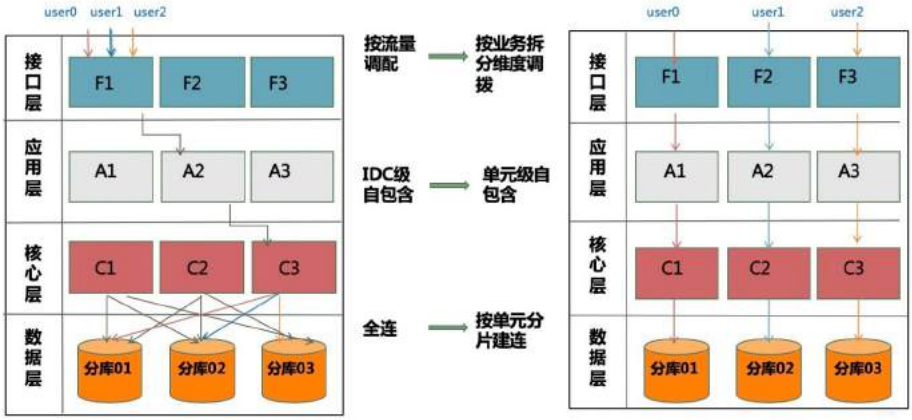
其次的瓶颈在交易系统和账务系统，支付宝通过多 IDC 的改造，将交易系统和账务系统进一步拆分，扩展了系统承载能力，并通过 FailOver 机制，避免单点故障。



多 IDC 架构

2013 年，支付宝 All-In 无线，移动端首次在双十一大促中承担重要职责。

但实际上这一年最重要的变化是，支付宝开始实施单元化的 LDC 逻辑数据中心，并在此基础上实现了交易支付系统的异地双活，有效提升了核心系统的持续服务能力。



单元化改造示意图

这里的单元是指一个能完成所有业务操作的自包含集合，在这个集合中包含了所有业务所需的所有服务，以及分配给这个单元的数据。单元化架构就是把单元作为系统部署的基本单位，在全站所有机房中部署数个单元，每个机房里的单元数目不定，任意一个单元都部署了系统所需的所有的应用，数据则是全量数据按照某种维度划分后的一部分。

支付宝单元化改造并不是一蹴而就，过程十分艰辛，不过带来的好处也非常多，包括理论上无限可伸缩微服务架构、异地多活部署、全站蓝绿发布和线上灰度仿真、异构机房上的弹性混合云架构等。

从 2009 年到 2013 年的双十一，支付峰值从数百笔增长到 1.3 万笔每秒，每年的增长速度都是数倍甚至十几倍的提升。为了支撑疯涨的流量，这些年里，支付宝技术的最重要目标是扩展系统的承载能力，可以说是被大促拽着前进。到 2013 年，

LDC 架构实现了理论上无限伸缩的能力，技术团队终于可以稍作喘息，开始追求精细化，大促相关技术也开始走到新的阶段。

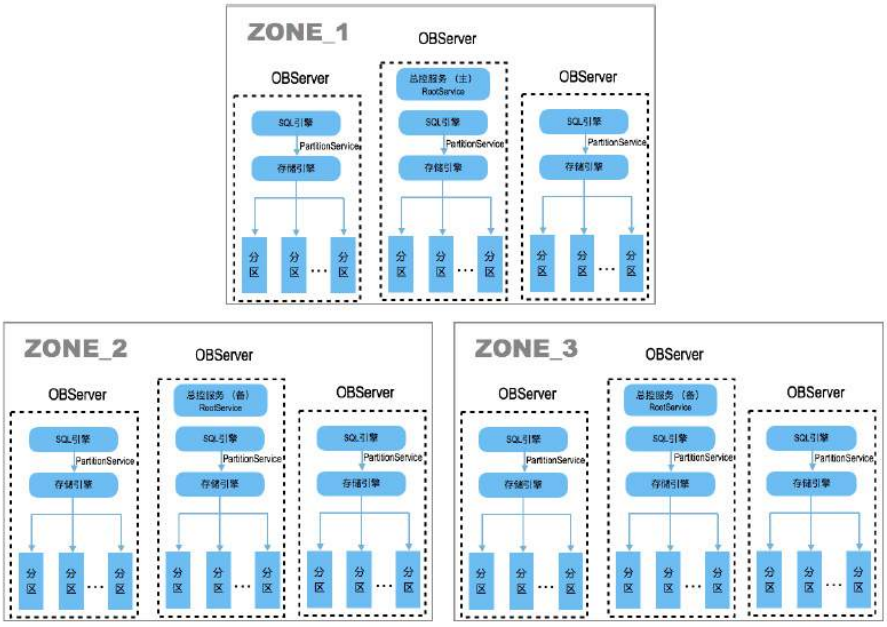
2014-2018：从去 IOE 到大促技术标准化

2014 年对于支付宝来说，是一个全新阶段的开始：去 IOE 开始落地，大促神器全链路压测正式启用，最后大促中控平台上线，标志着支付宝双十一大促相关技术走向标准化。

首先是去 IOE，OceanBase 在 TPC-C 数据库基准测试取得第一名，这个蚂蚁自研的分布式关系数据库一下子为世人所知，这里我们看一下这些年来它在幕后一路走过的里程碑：

- 2014.11 OceanBase 承担支付宝交易 10% 流量；
- 2015.11 OceanBase 承担支付宝交易 100%、支付 50% 流量；
- 2016.11 OceanBase 承担支付宝交易 100%、支付 100%、花呗账务 30% 流量；
- 2017.11 OceanBase 承担支付宝交易 100%、支付 100%、账务 100% 流量，“去 O”大功告成！并第一次走出阿里，应用到南京银行互联网核心系统；
- 2018.11 基于 OceanBase2.0 分区的百万支付架构上线，正式兼容 Oracle；
- 2019.11 OceanBase 取得 TPC-C 测试第一名，双十一每秒处理峰值 6100 万次；
- ……

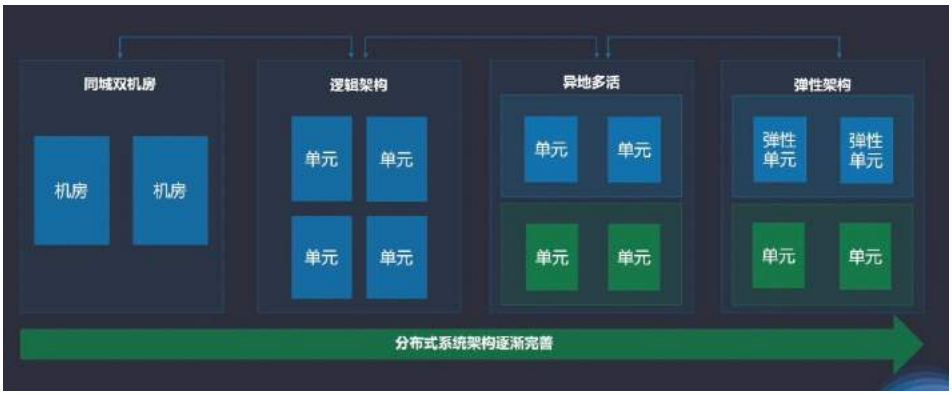
在 2014 年时，支付宝决定将核心交易系统的 10% 流量放到 OceanBase 上，在今天看来是非常冒险的举动，不过当时之所以这么做，是因为之前所采用的 Oracle 数据库已经无法支撑下一次大促了。OceanBase 也经历了非常多的优化，在 2018 年的双十一中，OceanBase 2.0 上线，并确立了以 OceanBase 2.0 分区为主的架构，彻底解决了数据库层面的瓶颈，实现百万支付架构。



OceanBase 分区架构

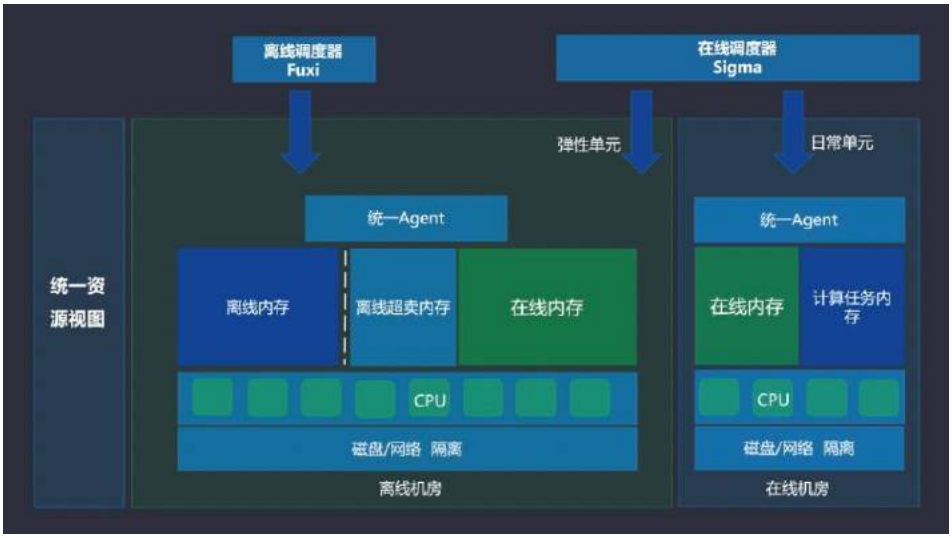
2014 年双十一首次采用全链路压测，这可以说是大促保障的核武器。全链路压测解决了之前的大促规划靠拍脑袋、成功看运气的问题，通过模拟用户在大促时的行为和流量，将涉及到的每个系统、每项技术都测试到位，大大减少了大促出事故的概率。

2016 年，支付宝还在 LDC 逻辑数据中心的基础上，在支付核心链路成功实现弹性架构，100% 按照运营需求弹性扩充资源，极大节省了成本。弹性架构可以让系统随流量自动化的伸缩，可以降低成本，提升运维效率。



从多 IDC 架构到弹性架构的演变

2017 年，支付宝首次实现离在线混合部署，80% 在线资源，20% 离线资源。随着大数据和机器学习的落地，支付宝平时有一些离线计算集群，通过调整工作时间，可以实现和大促高峰流量错峰，再和弹性架构结合，就实现了分时调度，同一批资源在不同的时间运行不同的应用，最大程度提升资源利用率。



离在线混部示意图

2018 年双十一，支付宝推出系列活动，“码上双十一”联动手淘拉新促活用户，相当于开启了双大促，如果按以前的做法，需要采购双倍的服务器，有了混部和分时调度之后，才得以在新增少量服务器的情况下支撑，极大的节省了成本。

2017 年其它的新技术还包括：

- 首次启用智能客服系统，单日服务总量超过 500 万人次。
- 智能风控 AlphaRisk 多模式切换落地，确保账户资损率低于千万分之五，遥遥领先于全球平均水平。
- 首个在国内自主研发的图数据库 GeaBase 及自研图查询语言第一次参加双十一，实现了对百亿级关系网络毫秒级的查询和变更。

2018 年双十一的时候，关于大促保障的相关技术基本已经成熟，但都比较零散，如果想更好的利用这些技术，需要把它们整合到一起，大促中控平台也正是在这个背景下出现的，它标志着支付宝将之前的大促经验沉淀，大促保障相关的技术实现标准化。

2018 年双十一的新技术还有：

- 首次在大促中进行红蓝攻防演练，在全链路压测的基础上进一步提升整个系统的可靠性。
- 大促巡检平台上线，实现大促相关的配置的全面检查，防止配置失误。
- 区块链技术第一次全面参战，百余个国家和地区、1.5 亿跨境上牌的原产地溯源不再是难题。
- 生物支付首次成为天猫双十一主流支付方式，每 10 笔支付就有 6 笔采用生物支付，标志着中国的生物支付时代来临。

2017 年双十一支付峰值 25.6 万笔 / 秒，2018 年支付宝内部实现可扩展的百万支付架构，再加上大促中控平台上线，我们可以看到，支付宝大促相关的技术已经成熟，并在不断打磨中追求可靠性、资源利用率、自动化、灵活性，把大促相关技术做到极致。

2019+：面向未来的金融科技

从 2018 年起，支付宝开始尝试和探索云原生架构的落地，同时数据智能相关技术也在公司内广泛使用。2019 年，相关的技术开始亮相大促舞台，以支付宝自研产品 SOFAMesh 搭建的金融级云原生基础设施首次登上双十一，打造了业界最大的 Service Mesh 集群。

OceanBase 2.2 上线大促核心支付链路，每秒处理峰值 6100 万次，这是 OceanBase 数据库同时兼容 MySQL 以及 Oracle 两种模式的一个里程碑的版本，性能和稳定性上也相对 2.0 版本有大幅提升。

最重要的是，支付宝技术开始走向无人区，未来的道路只能靠自己开拓，没有别人可以参考。支付宝开始自己描述面向未来的金融技术架构，推出了金融级云原生、开放计算架构等技术和理念，打造面向未来的金融科技。

金融级云原生是指在 CNCF 所定义的云原生基础上，发展出适合金融机构使用的一套技术栈，其中包括全面对外开放的 SOFAStack 金融级云原生分布式平台，以及通过安全容器、TEE、Service Mesh 等构建的全栈云原生安全架构。

SOFAStack 不仅采用 Kubernetes、容器等社区化的云原生基础技术，同时考虑金融行业的特殊性，支持之前的传统架构，形成双模技术，包括：

- 双模 PaaS：包括传统 PaaS 和基于 Kubernetes 的 PaaS，在 PaaS 中用容器来模拟 VM 的运行模式，同时支持容器和虚拟机；
- 双模微服务：通过 Mesh 来统一基于 Kubernetes 的微服务和老的服务注册中心式微服务；
- 双模 Serverless：同时支持基于云原生的 Nodejs Serverless 模式、基于 JVM 的 Serverless 模式，以及基于云原生的 Java Serverless 模式。

在开放计算架构方面，支付宝通过设计一套符合当下计算体系，同时又能应对未来计算趋势的技术框架，来解决计算引擎更新、统一研发体系、数据共享互通、数据

风险防控等几方面问题。

首先在计算引擎方面，针对不同的计算作业状态进行统一管理，达到兼容任何一种计算引擎，并且实现插件化能力；在研发层面，推出了 SmartSQL，在标准 SQL 规范之上扩展了部分功能及语法，希望用最简单通用的语言，描述绝大部分的计算及机器学习作业；在存储层面，支付宝自主研发了统一存储系统，支持多种类型的数据存储格式，同时支持一份数据不同格式之间的自动转换及迁移，极大地简化了引擎层对存储的使用，同时节约了大量成本。

另外，开放计算架构之所以叫开放，就是它能包容不同的计算模式和引擎。在开放计算架构之中，包含了离 / 在线一体的图计算引擎及存储，从使用场景，可以覆盖在线、实时、离线各个场景，用于支持不同时效性的业务。从功能上，具备金融级图数据库，超大规模图计算，流图混合的动态图计算以及超快内存图等计算能力，覆盖了不同量级的数据计算能力。在机器学习方面，开放计算架构还包含了之前已经开源的 SQLFlow 以及前段推出的 ElasticDL 弹性深度学习框架。

这一整套体系将 BigData 和 DataBase 的理念揉合在一起，又称为 Big Data

Base，是支付宝摸索出来的金融数据智能的最佳实践，下一代大数据的基石。

2019 年，开放计算架构在支付宝不同场景中落地。随着图计算在花呗，蚂蚁森林等场景中大规模上线，图数据库 Geabase 突破万亿边，在线图分析百亿关系特征计算秒级响应。同时，通过融合计算引擎，协同流、图、并行计算、机器学习等不同计算模式，在支付过程中提供秒级智能决策能力。

结语

互联网技术的更新迭代永无止境。

对支付宝人来说，虽然已经经历了这么多的大促，但每一年的双十一仍然是全新的挑战。这个挑战不仅来自于不断推陈出新的业务和运营，也来自于技术人对自己的严格要求，就像那句话所说的，今天的最好表现，是明天的最低要求。

正是这样的驱动力，驱使着支付宝的技术人不断的超越自我，突破舒适区，寻找下一场技术革命。也许，这也正是双十一的意义所在。

2019 双 11，支付宝有哪些“秘密武器”？

作者：蚂蚁金服科技

2019 双 11，支付宝参战的第十一年。

与十一年前相比，双 11 的许多东西都改变了。比如金额——2684 亿，差不多是十一年前的 5000 倍；比如流量——订单峰值 54.4 万笔 / 秒，曾经是想都不敢想的数字；再比如层出不穷的新技术，就是这些惊人数字背后的“秘密武器”，给迎战双 11 的战士们作最完备的武装。

也有始终不变的东西。大战来临前的紧张、不安、如履薄冰，对每一个细节反复 check 的“强迫症”，以及胜利之后的欣喜、释然、满心充实，和下一步砥砺前行。

支付宝的技术工作，就是“半年搞建设，半年搞大促”。虽然是一句戏言，但足

够从侧面证明大促作为实践战场的重要性。而每当双 11 圆满落下帷幕，技术人也就到了收获的季节。那些历经双 11 大考的新技术，就像经历过了“成人式”一样，一一走到台前开始独当一面。

SOFAMesh：金融级云原生第一步

众所周知，金融机构因为肩负的责任重大，面对新技术时，普遍都是比较保守的。支付宝也不例外，尤其是在双 11 这种场景下，流量大，峰值高，平时不管多小的问题，在这时候都可能被放大成不得了的大问题。

于是，今年的大促迫在眉睫时，SOFAMesh 团队还在纠结。来自周围的各种声音，让他们感到压力很大。被问到的最多的问题，就是“这个靠不靠谱?”

一个“行”字，在双 11 的面前，可能有千钧之重。能不能扛过零点的流量峰值？能不能保障稳定？能不能保证不出差错?

Mesh 是一项很新的技术，社区开源项目本就不成熟，而 SOFAMesh 是支付宝从第一行代码就开源加自主开发的项目，在金融级的严苛要求面前，在双 11 的极端场景之下，究竟行不行？谁心里都没有底。

然而此时不上，整整两年的心血就白费了。反过来说，如果能打赢这一仗，就证明云原生之路在双 11 这种体量的考验之下都是可行的，这对于整个行业而言，会是一个很好的标杆。

“蚂蚁金服要做金融行业技术的拓荒者和实践者。”资深技术专家杨海悌说。

这已不是蚂蚁金服第一次做“吃螃蟹的人”，在金融机构普遍依赖 IOE 时，他们率先开始探索分布式，现在分布式渐渐成为主流，他们又率先琢磨起云原生。

“以前都是业务推动技术，现在到了技术为业务提供红利的时候了。”对于自己看着长大的 SOFAMesh，杨海悌一面很有信心，一面也十分忐忑。

SOFAMesh 是支付宝针对金融行业的特殊需求而开发的金融级中间件，属于金融级云原生分布式框架 SOFAStack 的一部分，这个框架的开发始于 2009 年，几乎和双 11 同龄。

是骡子是马，总得遛过了才知道。SOFAMesh 的第一份答卷很快交了出来——以往分时复用的资源切换需要 4 小时，用上了 SOFAMesh 之后，不到 4 分钟。性能提升将近百倍。

分时复用，顾名思义，就是在不同的时间段里让同一个资源能够“复用”于多个应用。这一技术能够减少资源闲置，提高资源的利用效率。这一技术在 2018 年双 11 就曾立过功——当时，支付宝面对这天猫双 11 和自己的会员大促的“双大促”挑战，为了节约成本少采购一些资源，上线了分时调度 1.0，使用同一批资源同时支持两个大促，在支撑天猫双 11 和经济体用户增长两个大促的同时，IT 成本一分钱也没有涨。

但去年在弹性架构模式下做分时调度，切换资源需要重新配置和部署相关系统，4 个小时的切换时间，虽然成功支持了“双大促”，还是满足不了对短时间内快速调用资源有需求的业务。

到了今年，由于 SOFAMesh 的上线，切换资源不再需要重新部署，切换时间缩短到了 3 分 40 秒。这意味着，像蚂蚁森林那样每天都会面临流量小高峰的业务，无需事先留足资源余量，提前 10 分钟开始切换资源，都绰绰有余。

“将来，切换时间还有望缩短到秒级。”杨海悌说。

2019 年双 11，SOFAMesh 扮演了非常重要角色——100% 覆盖蚂蚁金服核心支付链路，几十万容器，峰值千万 QPS，平均 RT（响应时间）0.2ms，是业界最大的 Service Mesh 集群。它在洪峰面前的稳定性和平滑性，以及对效率的显著提升，都是有目共睹的。

这张漂亮的成绩单背后，其实就是一个字——行。

“云原生”已经成为业界公认的技术趋势，它的目标是提升运维效率、降低资源使用成本、提升服务安全可靠性等。云原生带来的基础设施升级，为技术演进提供基础能力支撑，并且提升未来架构空间的想象力。2019 也是支付宝的金融级云原生落地元年，包括 SOFAMesh 在内的一系列云原生技术，经历双 11 的考验之后，向整个业界证明——我们行，云原生这条路，也行。

双 11 之后，蚂蚁金服举办的发布会上，副 CTO 胡喜宣布，会将 SOFAMesh 也对外公开。

正如“元年”一词所说，这只是蚂蚁金服在新的开拓之路上迈出的第一步。

OceanBase 2.2：世界纪录就是用来打破的

OceanBase 被人质疑“行不行”的次数，更是多到数不过来。

数据库是命脉，尤其是金融机构的数据库，出一点问题都是真金白银的问题，哪个业务都不敢冒风险，老老实实抱着老牌进口货 Oracle，图个太平。

但 Oracle 也没见过双 11 这种阵仗，随着双 11 的流量连年翻番，它的性能眼见着碰到了天花板。2014 年双 11 前的压测，Oracle 出现了 10% 的流量缺口。

OceanBase 感到机会来了。在那之前，他们已经“蛰伏”了四五年，没有固定的业务，最落魄的时候，甚至面临团队解散和项目取消的境况。

当时的 OceanBase 将满 5 岁，版本号却还是 0.x，外表看来甚至还是个 demo，一上来就要承接双 11 的 10% 的流量，相当于支付宝平日流量的最高峰，而且要做的还是最核心的交易系统——一分钱都不能出错的那种。

一时之间，“你们行不行”的质疑声此起彼伏。

“别人说我们不行的时候，我们都非常坚定地认为，行。”蚂蚁金服研究员杨传辉说。他是 OceanBase 开发团队的初期成员之一，亲眼见过 OceanBase 写下第一行代码。

从拿下 10% 的任务，到双 11 的正式大考，时间不足两周。最后十来天，资深运维专家师文汇带着全团队几乎不眠不休地做优化，硬是把长达 10 毫秒的响应时间降低到了 1 毫秒以下。

那一年的双 11，OceanBase 没出一个差错，一战成名。

今年的双 11，OceanBase 的版本号是 2.2。在为版本命名方面，他们的谨慎作风一如既往。

但是 OceanBase 的每一次版本迭代，发生的都是“脱胎换骨”的变化，自己创下的纪录，也由自己不断刷新——

2018 年双 11，基于 OceanBase 2.0 分区方案的架构正式上线，这一架构解决了数据库可扩展的瓶颈，将每秒交易的承载能力提升到百万级，并让性能提升了 50%。

50% 的提升不是个小数目，但更令人惊讶的是，仅仅一年之隔，在 2019 年的双 11 中，全新上线的 OceanBase2.2 版本，在 2.0 的基础上，又让性能提高了 50%。

就在今年的 10 月 3 日，权威机构国际事务处理性能委员会 TPC 披露：蚂蚁金服的分布式关系数据库 OceanBase，打破美国甲骨文公司保持了 9 年的世界纪录，登顶 TPC-C 榜单，同时也成为首个登上该榜单的中国数据库系统。

短短的一个月之后，在 2019 年双 11 的考场之上，OceanBase2.2 又再次刷新了数据库处理峰值，达 6100 万次 / 秒，创造了新的世界纪录。

在金融级核心数据库的严格要求之下，OceanBase 为何还能有这样跨越式的性能升级?

关键的秘密在于，OceanBase 背后是原生的分布式数据库设计以及 PAXOS 协议，通过水平扩展 x86 服务器就可以达到无限伸缩，支持大规模高并发的效果。

另一方面，今年为了进一步提升性能和降低延迟，OceanBase 还通过中间件的优化，自动将多条 SQL 聚合成轻量级的存储过程，这个过程让原本需要数十次 SQL 网络交互的任务降低为单次网络交互，整体 RT 降低了 20%。

现在，支付宝的业务已经 100% 跑在 OceanBase 上，作为我国第一个自研的金融级分布式数据库，经过六年的双 11 锤炼，它也已经具备了走出蚂蚁金服、走向更广阔天地的底气。

今年双 11 中，支付宝支付业务 100% 切换到 OceanBase 内置的 Oracle 兼容模式上，支持 Oracle 语法以及存储过程优化的同时，又兼具 OceanBase 的分布式能力，如分布式分区表、全局事务等，响应时间也更加平稳。双 11 之后，OceanBase2.2 也将正式公开发布。

“不过，在别人觉得我们什么都行的时候，我们反而会冷静下来，想想自己还有哪些不行的地方。”杨传辉说，对技术上一切未知的敬畏，才能让大家走得更远。

图智能：复杂金融关系的最优解

“过去很长一段时间图数据库和图计算一直停留在学术研究阶段，行业应用场景不多，是因为没有强的场景驱动，所以市场没有太多发展”，蚂蚁金服计算存储首席架构师何昌华指出。但是反过来看，图相关的产品近年来热度不断攀升，其核心原因是因为强场景的驱动，特别是金融场景，它非常善于处理大量的、复杂的、关联的、多变的网状数据，通过节点和关联的数据模型去快速解决复杂的关系问题。

蚂蚁的一站式图平台的诞生，也有着鲜明的蚂蚁金服特色，同样是“被业务倒逼出来的”。

蚂蚁金服在 2014 年左右就开始跟进社区的图计算的研究，当时的团队在一些开源产品基础上进行了小规模的尝试，做了之后发现效果很好，图数据库能够很好地和金融、社交业务结合起来。但是，蚂蚁金服有着巨大的数据量，需要以分布式架构来支撑高并发的大数据量和大吞吐量，但当时无论是开源还是商业数据库产品都只是单机版，都难以适应蚂蚁金服如此大的数据量和复杂的环境。于是，艰难而又步步扎实的自研之路开始了。

最开始，要解决的是图数据的存储和在线查询的问题。

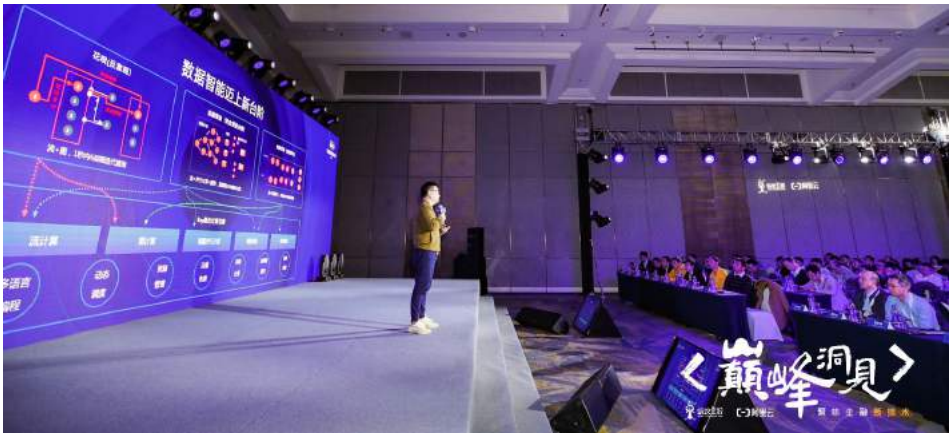
从数据量来看，分布式架构是唯一的选择。从满足金融场景高并发低延时的需求来看，选择原生图结构而非基于关系型数据库基础上封装图数据，成为必然。但也因为以上两点，导致整个开发难度大大增加。

从 2015 年初团队开始组建，经过“冬练三九、夏练三伏”的苦修，以及在代码、运维、稳定性等每一环节的极致追求，第一个图数据库版本 GeaBase 在 2016 年初发布。

而这时候，刚好遇到支付宝史上最大一次改版，模块化功能被替换成信息流，大大强化了社交关系属性，GeaBase 开始接入支付宝链路。

百炼成钢，经过几个月的压测，2016 年 6 月，新版支付宝上线，GeaBase 迎来了第一笔流量。接着几年，从支付宝大改版到新春红包再到双 11，GeaBase 迎来了业务的绽放期，到 2019 年双 11，GeaBase 双 11 主链路上单集群规模突破万亿边，点边查询突破 800 万 QPS，平均时延小于 10ms；成为支付宝核心链路上非常重要的一环；

数据存储和查询的问题解决了，紧接着要解决的是分析计算的问题。



在一开始，我们思考的是如何在海量的图数据里做数据挖掘的问题。在面对千亿乃至万亿级规模，几 TB 到几百 TB 的数据，用超大内存物理机和高速网络来实现离线全图计算，对企业来说不太现实，资源也存在极大的浪费。因此，我们重点放在如何在满足业务功能 / 性能需求的同时，利用碎片化的现有资源，实现“按需计算”的能力。

因此，2017 年，我们在海量数据基础上，设计了一套离线计算的框架，提供自适应的分区策略，资源消耗能比同类产品降低一个数量级，同时性能还能远远优于 GraphX 等开源产品。

同时，为了方便业务算法人员根据其业务进行二次开发，还开放了 C++ 和 JAVA 的接口，除了业界常见的图编程框架的 Pregel、GAS，我们还做了一定的“微创新”和能力扩展，提供了更高性能，更加丰富功能的接口。

全量分析计算的事情解决了，但随着“310”策略的推进，风控业务的发展，对分析的时效性的要求越来越高，分析需要更快，更实时，2018 年，我们开始考虑在线图计算的能力。

有时候，并不是所有业务都需要进行复杂的图分析，而是在满足一定的条件后才开始进行子图的迭代计算。最后，基于图的迭代计算的结果，在进行数据链路的处理后再提供给在线使用。

因此，一个场景在完整的计算链路中，需要流计算和图计算两种模态的融合计算。我们打破了传统计算模态的边界，提供流图融合的计算系统。通过将数据流和控制流相结合，并提供动态 DAG 的能力，从而实现按需计算，弹性扩缩容。

用户可以通过一套统一的 DSL（SQL+Gremlin/GQL）、一套计算系统来实现完成流图融合的链路，实现基于数据驱动的在线图计算能力，同时，减少了用户的学习、运维成本。

在 2019 年双 11 上，在线图计算技术大放异彩，通过秒级决策，在花呗等场景帮助业务效果提升 12 倍。

从“海量”图存储，到离线全图“按需计算”，再到“实时”在线图计算，蚂蚁的图智能技术跟随业务一步步发展，壮大。

融合计算引擎：新计算威力初现

今年的双 11 还落地应用了一套新的“神器”——融合计算引擎，它耗费了近百位工程师一整年的心血。

融合计算引擎的基础，是蚂蚁金服联合 UC Berkeley 大学推进的新一代计算引擎 Ray，它很年轻，2018 年融合计算引擎项目启动时，它只有几万行代码，距离金融级线上环境的应用还差得很远。

“我们用了一整年，把它增加到了几十万行代码，并且涵盖了 C++、java、Python 等所有语言。”蚂蚁金服资深技术专家周家英说。

至少 4 个团队在共同“养育”这个引擎，四个奶爸带娃，磕磕绊绊，在所难免，难度远远大于一个团队负责一个引擎。

但开发时的“难”，是为了应用时的“简”。

在计算引擎执行层面，不同计算模式的数据是可以在引擎内共享的，很少借助第三方存储，因此对外部存储和网络传输的开销也都有极大的节省。

在应用方面，融合计算引擎不仅能够解决金融场景中需要衔接多个不同计算模式的难题，还能支持各种不同时效性的业务，并在支付过程中提供秒级智能决策能力。

并且随着融合引擎的落地，也改变着技术同学的研发习惯。我们希望通过融合计算引擎，达成研发态，运行态，运维态三位一体的统一：例如在动态图计算场景，计算开发同学只需要编写一个流 + 图的计算作业，就可以实现秒级 6 度邻居的图迭代计算；同样在机器学习领域，通过编写一个包含流 + 模型训练 + 服务的计算作业，就可以实现端到端秒级模型导出的在线学习能力。这样从研发到运行态，计算整体效率都得到了极大提升。

2018 年，融合计算就在花呗反套现的智能甄别之中表现卓越。到了 2019 年，融合计算引擎已经在支付宝不同场景中落地——图计算在花呗，蚂蚁森林等场景中大

规模上线，图数据库 Geabase 突破万亿边。

2019 年支付宝新春红包活动中，融合计算引擎用在线学习能力支持了新春红包的智能文案，让它的算法跑在了新的在线学习的体系上。这个体系融合了流计算和机器学习，让机器学习的模型迭代速度从以前的小时级别，提升到了现在的秒级别。本次双 11 时，它在“支日历”的推荐算法方面发挥了重要作用。

通过融合流计算、服务和并发查询，融合计算引擎减少了 60% 的机器资源使用，把端到端的延迟压低到了毫秒级，同时还能支持金融网络的业务查询和监控。

今年双 11 中，融合计算引擎在至少三个场景中成功落地并被验证可行，“还跑在了蚂蚁金融级关键决策链路上。”周家英不无兴奋，“这证明了我们的计算引擎具备了金融级的能力。”

事实上，无论是在双 11 这样的极端大考场景中，还是在支付宝、阿里巴巴，以及各个互联网科技公司的日常应用场景中，数据驱动的业务也越来越多。相应地，海量数据的实时处理、分析和应用，以及人工智能、深度学习等新技术的开发，都在要求着更强大的计算能力，以及能够应对复杂场景的多种计算模式。

面对未来，更多的是未知——我们尚且不知未来会出现什么样的场景，这些场景会要求什么样的计算模式和计算能力。“融合计算是真正意义上的新计算的第一步。”蚂蚁金服计算存储首席架构师何昌华说。

双 11 OceanBase 图计算 Cloud Native 存储

金融级云原生如何助力双十一？蚂蚁金服的实践经验是这样

作者：杨冰

蚂蚁金服金融科技产品技术部总经理杨冰，近日分享了蚂蚁金服金融级云原生在双十一的大规模应用实践，以下为演讲整理全文：

2018 年双 11，蚂蚁金服和网商银行正式应用云原生架构，2019 年双 11，蚂蚁金融级云原生架构进一步深入落地，Service Mesh 架构 100% 覆盖蚂蚁金服核心支付链路，是业界最大的 Service Mesh 集群。本文将分享蚂蚁金融级云原生在 2019 年双 11 中的大规模应用实践。

历年双十一的交易峰值不断增长，给整个技术带来了巨大挑战和牵引力。在这样的驱动力下，技术架构又发生什么样的变革？蚂蚁金服从第一代集中式架构，逐步

走向分布式架构，再到云平台，蚂蚁的架构已经形成了比较体系化的金融级分布式架构，能够很好的去支撑业务的发展。

金融级分布式架构需要 Service Mesh

几代架构演进中，虽然解决了很多业务痛点和问题，但也确实让业务做了很多的升级、改造、上云等事情。我们开玩笑说云包治百病，业务研发团队就问，能不能让我们未来只关注业务开发本身，剩下的事情交给基础设施？我相信这个问题也是我们下一代架构往金融级云原生架构演进过程中需要去解决的一个问题，也是云原生这个技术本身要解决的问题。总结来说：交易规模的演进确实需要架构升级，但架构升级不希望成为业务的负担。

如何解决这个问题呢？我们先来回顾下架构演进的过程。

金融交易技术演进的第一阶段是实现金融级分布式架构，SOFAStack 和 OceanBase 是构成这个架构的基础。需要强调的是，整个金融级能力，包括高可用、一致性、可扩展、高性能，都需要应用到数据存储端到端整体能力。

比如分布式事务，OceanBase 从数据库层面提供了分布式事务的支持，但当我们把两个极其重要的业务数据放到两个 OceanBase 集群的时候，就需要由应用层来解决跨数据库的一致性问题，这个就是中间件需要提供的能力了。所以分布式架构下必须具备的端到端的一致性，才能构成整体架构的一致性。高可用、可扩展、高性能也是一样的道理，缺任何一个能力都不能算是真正的金融级分布式架构。

这张架构图的虚线以下表示的是目前蚂蚁的金融级分布式架构中的核心组件，这一层软件经历了十几年的发展已经抽象的比较完备，而且很好的支撑了业务的发展。

但真正系统实现中，业务层跟下面几个组件有着千丝万缕的联系，这个关系可能存在于下层为了减少上层的改动提供的兼容性 SDK 里，也许存在于上层迁就适配下层特殊用法里，这样的例子层出不穷，一直伴随着整个架构演进的过程。

而在下层演进发展过程中，这种耦合和千奇百怪的兼容适配问题就会形成巨大的阻力和风险。金融 IT 同行在做数字化转型的时候也会遇到同样的痛点，甚至于可能会比像蚂蚁金服这样的互联网公司遇到的困难更多一些。相比于互联网公司相对还比较统一的架构来说，很多企业的 IT 系统中就像博物馆，三代同堂，遗留系统一大堆，还有很多外采的系统。所以这个架构分层虽然清晰，但却无法独立演进。

蚂蚁同样有这样的问题，内部有上千个系统，几十万台服务器，在业务迭代发展这么快的情况下，根本没有太多精力来配合进行基础设施的升级。而业务不动基础设施就动不了，很多能力就无法全局生效，这样就发挥不了威力。

举个最简单的例子，全链路压测很强大，但如果没有自上而下的推动全业务线配合升级 SDK，使得整体通信架构能支持全链路标识透传和识别的能力，就无法实现全链路压测，也许到今天我们还没法享受这个技术带来的红利。这样的例子很多，很多的创新也是因为卡在「相互等待」的死锁中，迟迟未能落地，甚至胎死腹中。

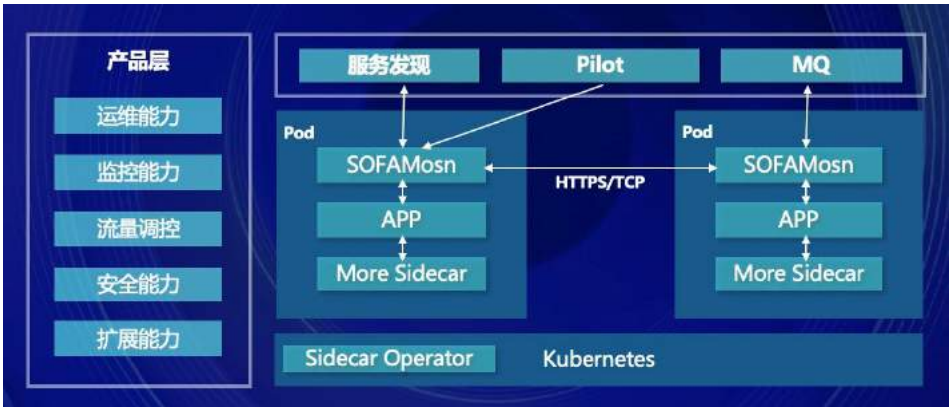
软件世界里面有一句经典的话：没有什么问题是不能通过一层抽象解决的，如果说有，那就再加一层。我们希望能够有这一层将业务和基础设施解耦，正好云原生发展过程中出现了 Service Mesh，而且这一层软件抽象从逻辑分层走向了物理分离，既起到了解耦的作用，又具备独立演进的能力。

康威定律认为，一个软件系统的架构背后一定有对应的组织架构，之前的架构中，业务开发和基础设施开发虽然在组织上已经基本分开，但软件上却没有切开，Service Mesh 的出现正好解决了软件架构和组织架构不匹配的问题。

把这层抽象剥离变成独立的载体是一个趋势，不仅仅 Service 应该被 Mesh 化，更多的中间层可以被 Mesh 化，蚂蚁也是这么实践的。这也是业务应用在云原生时代，如何把自己的一些诉求更好的交给云，让应用跟云之间的交互变得更简单的一个过程。

蚂蚁金服自研的 SOFAMesh 在双十一中大规模应用，100% 覆盖核心支付链路，容器规模几十万，千万 QPS，性能消耗极低，而将迭代速度从 1-2 次每年提升到 10+ 次每月。在 CPU、内存和相应时间等数据上来看，我们通过极小资源的代价，换来了系统快速迭代的速度，在备战双十一的短短两个月，Service Mesh 进行了几十次迭代和发布，快速响应了全新的大促需求，并完整多次的全站升级和全链路的线上验证。

这在以前几乎是不可想象的，今年任务的复杂度和工作量，如果没有 Service Mesh，将无法完成。要么只能砍需求，要么需要消耗更多业务研发的资源来配合我们做大量升级验证工作。遗留系统也能通过装配这样一套软件，就能够具备完整的分布式架构能力，且性能损失只有一点点，同时还获得了分布式架构的快速演进能力，因此这绝对是一个值得投资的技术方向。



SOFAMesh 架构图

这个是 SOFAMesh 的架构图，里面是分成控制面和数据面两个部分，控制面是在产品层大家想要的一些能力，走向微服务架构要解决什么样的问题，希望在这个架构上有更多的运维能力、监控能力、流量调控能力、安全能力、扩展能力。

数据面 SOFAMosn 是独立出来的进程，它可以跟 APP 进行交互，和 APP 在一个 POD 里。图中除了 SOFAMosn 承载了 APP 之间的通信任务之外，还将异步

消息的通信逻辑也沉淀进去了，成为应用对外同步异步通信的统一门面。同时我们也把应用访问数据库的中间层路由逻辑和配置管理逻辑独立成一个 sidecar，称之为 DBMesh，这个图里用 More Sidecar 来表示，意味着未来可能有更多的逻辑可以下沉到 Mesh 层。

未来应用将使用非常轻薄的 SDK 或者简单的 API 跟外界交互，而复杂的路由、治理、高可用等相关的逻辑就下沉到 Mesh 层了，此后全局架构改造就会变得非常的轻松，对业务没有打扰。

接入 Service Mesh 的挑战与解决之道

在应用 Service Mesh 的过程中，蚂蚁金服也遇到了不少困难和挑战，很多的社区方案其实并不能很好的满足金融行业的需求，尤其对高可用、稳定性的苛刻要求，蚂蚁在接入过程中也做了不少创新。

首先第一个问题是，Service Mesh 是如何接入到整个体系里呢，它的资源消耗又怎样呢?

要按云原生社区的方式安装一个 sidecar，就是扩容出来一批新的 POD，里面已经为 APP 装配好了 sidecar，然后把老的 POD 逐步下线销毁。这就像要给一辆车装个新功能，我们需要搞一批有这个新功能的新车给到客户，然后把老车召回，这个非常浪费资源。而且在给大规模集群上 sidecar 时需要更多的资源 buffer 去做滚动升级，成本很高且效率很低，而且连应用和 POD 一并销毁重建，这个变更的动静

太大，也的确有不小的风险。

我们采用了注入升级的方式，这里也对原生的 K8s 做了很多扩展的改动。

在资源消耗的优化方面，我们也做了很多探索。理论上既然将这个进程独立出来跑在容器里，就应该分配独立的 CPU 和内存资源，我们开始也是这么做的，但跑下来反而出现很多性能瓶颈。后来我们发现，根本性问题在于虽然这个进程是独立的，本质上它还是一段从应用里剥离出来的逻辑，无论它是帮助应用做服务通讯，还是访问数据库，都是在应用主链路上，如果为这些逻辑设置了跟应用不一样的 CPU 和内存限制，反而会使得应用在做主链路业务的时候，受到这个独立进程的资源消耗瓶颈影响。最后我们通过实践，通过注入融合的方式，跟应用一起去共享 CPU 和内存，不但能达到最好效果，也最大程度减少资源开销。

这整个升级的过程，类似于我们把一辆普通的车，注入一个模块变成了一个可持续升级的特斯拉，随时随地做空中升级。而且这个模块与应用共享电源和公共资源，达到最小的功耗。

既然空中升级的能力很好，怎么样能稳妥保证这个模块自身的升级呢?

社区的方案是销毁整个 POD，连同 APP 和 sidecar，再创造一个新的，里面包含了原来的 APP 和新的 sidecar。这样一来，虽然这个东西从应用层剥离出来，但是还是跟应用放在一起整体作为一个软件创建、销毁、升级，未免显得有些笨重。

其实大部分是要做这个进程的升级时，应用是没有变化的。这种方案似乎也违背了把它剥离出来进行独立演进的初衷，这也是我们探索过程中发现社区的方式不太理想的地方，而且整个方案也没有流量摘除、优雅上下线这种设计，整个方案比较简陋粗暴。

我们探索了独立升级 sidecar 的模式，并且实现了较为复杂的流量热迁移。整个过程应用完全无感知，就像是我们给别人打电话，如果对方是个双胞胎，当话筒被另一个接过去时，我们还是面对的同样的电话，同样的通话通道，对方换了个人，我们无感知。这样整个升级过程非常的平滑，也对业务无打扰，流量也不会出现波动，符合金融级对稳定性的要求。

整个过程还是用车打比方，之前看过一个劳斯莱斯的广告，车子开过隔离带的时候，里面的乘客没有任何感觉，放在车上那杯水也没有任何的震动，虽然没坐过这么好的车，不知道是否真的能做到像广告里的效果，但整个过程很震撼。我想平滑升级的效果就应该像这个过程一样。

Service Mesh 在双 11 大促中起到了什么作用呢? 我们先回顾下过去做过的一些事情。

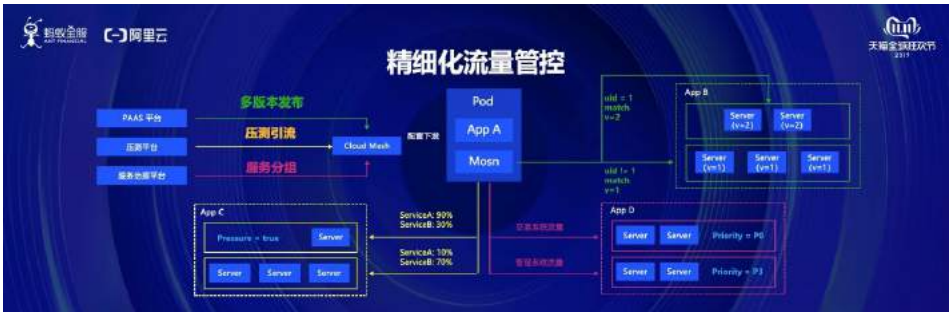
蚂蚁在上云后整个系统具备了弹性伸缩能力，于是每次在搞大促活动，无论双 11、双 12 还是新春红包，我们都将对应业务链路的系统，动态弹到云上，用完了再缩回来。这个动作说起来简单，其实非常复杂，而且操作的变更非常大，但是由于几个活动间隔时间足够长，所以我们可以比较坦然的做这个事情，而且还是会额外消耗资源。

今年提了新要求，双 11 零点要搞天猫大促，第二天早上 7 点蚂蚁森林又要支持偷能量搞活动，考虑到双十一后还有两三个小时处理收尾流量和降级的调度任务，留给系统做调度的时间也就短短的 4、5 个小时，怎么样在这么短的时间内将一个业务链路上的系统全部熄火，再把另外一条拉起来预热到最佳状态，而且不允许额外增加资源，这个挑战非常大。

我们通过类似超卖的方式把两条链路的应用部署在同一组 POD 里，一组处于运行态，一组处于保活态，当需要切换时，我们将运行态的资源收缩，流量比例调低，变为保活态，把另一组应用做反向操作变成运行态。

整个过程我们做了非常精细化的流量调拨、转发、保活。保活非常重要，因为应用在这么短时间热起来，需要有一定的保活流量来维系应用的基本状态，包括 DB 连接、通信连接、缓存等等。Service Mesh 在切换过程中将一部分流量转发到别的机器，一部分放进来用于维系应用的「热状态」。

听到这里大家可能会问，这样的场景是不是只有像蚂蚁金服或者是阿里巴巴大的一些公司才会有的？这个功能对于我来说有什么用？对于大部分的企业的意义是什么呢？这些流量调拨能力只是 Service Mesh 在大促应用场景当中一个小小的尝试。Mesh 架构除了解决基础设施下沉的一些问题之外，我认为最大的价值在于服务化流量的精细化管控，这也是精细化运维的趋势。



以上图为例，这里介绍了精细化流量管控的三个方面，包括多版本发布、压测引流、服务分组。

很多朋友听过灰度发布，新上一个服务，线上灰度验证一下，控制一定的流量比例给新服务，如果 OK 再全量放开，这件事情听上去也不难，例如这个服务就是外部流量入口的话，很容易控制入口流量的比例，因为流量上游就是外部客户请求，一定是有接入层可以控制的，但当是一个上游有好几十个，甚至好几个百各应用来调用就不那么好控制了。

如果我们想同时验证一组新服务可能更难，如何能保证这个灰度流量只在这组新服务之间流转而不逃逸出去？如果没有统一的流量调拨能力就没有办法控制这么细，

那只能用大招，就是用额外的资源，单独开辟一个环境，或者是单独开辟一个机房，专门作为灰度环境，流量在这个环境内闭环，这个方案可行，也是我们目前在用的方案之一，但是成本很高。如果全部装上这套系统，就可以在线划分出来任意的逻辑单元，做流量灰度，做多版本验证发布。

同样的逻辑也适用于细粒度的容灾切换、引流压测、服务分组等，未来可以基于这种全局的流量管控能力做很多创新，来做稳定性和高可用的事情，也能帮助业务做灵活的业务相关的流量调拨。

Service Mesh 带来的全局流量管控能力就好比手机导航。比如要从国贸去北京南站，系统会给你一个推荐路线。当大家都用同类软件的时候，整个系统就能根据整个交通通行状况给不同的人不同的推荐路线，以便最大程度的提升整体交通的吞吐量，在全局形成最优方案。这就是导航系统通过手机这个 sidecar 来进行全局车流量调拨和优化的场景，这也是 Service Mesh 可以给我们系统带来的价值。在这个方向上，我们内部已经有很多新的探索在做，我相信未来会有更多的创新玩法出来。

蚂蚁金服 Service Mesh 正式对外开放

以上提到的技术会我们会在 SOFAStack 的微服务平台上逐步开放，大家可能也都遇到了异构系统无法统一治理、分布式改造成本比较高、技术栈绑定等问题，也在希望能有一种方式简单直接的帮助整个系统改造为一套分布式系统，并且能适配兼容各种框架。我们今天开放的 SOFAMesh 可以以无侵入的方式帮助应用进行分布式

改造，支持多种协议，兼容异构系统，不但可以跑在云原生，还可以跑在虚机上。且经过了大规模金融级的线上验证。

这个图是大概的架构，不展开讲，核心意思是说这样一套体系可以兼容蚂蚁金服的 SOFA 系统，也可以兼容社区的 Dubbo 应用、SpringCloud 应用，整个可以跑在云原生架构上，也可以跑在虚拟机架构上，背后是有经过我们多年考验的服务注册中心作为支撑。

现在我们只迈出一小步，在未来还会继续做更多探索。除了服务化，我们正在把全机房流量都统一管理起来，无论是从网络接入层进来的南北向流量，还是一个机房内部的东西向，或者是机房之间的东西向流量，全部纳入到一套体系上来管理。同时，在对外的产品上，为了帮助更多的客户适配已有的体系，我们也在做针对不同的注册中心适配工作，以便能够纳入到同一个控制面统一管理，这个是未来演进的方向。

这些技术本身有很多核心的代码也是开源的，对内在跟阿里集团在共建，对外也在努力向 upstream 做贡献，跟谷歌这样的公司合作。我们希望能够得到大家更多的关注并参与贡献，也希望大家能积极参与到云原生架构在金融场景的落地探索中来。

OceanBase 创始人阳振坤：什么是面向未来的数据库?

作者：阳振坤

蚂蚁金服高级研究员兼 OceanBase 数据库创始人阳振坤，近日分享了《OceanBase：面向未来的企业级关系数据库》，以下为演讲实录：

10 月 2 日，OceanBase 数据库 TPC-C 基准测试结果发布，今天我想跟大家汇报一下它背后的一些东西，然后给大家简单讲一下我们的技术方案，以及给大家介绍 TPC-C benchmark 以及 OceanBase 进行测试认证的一些事情，最后是个简短的小结。

我们做的是在线的交易处理系统（OLTP），但更大的意义不在于 benchmark 本身，而在于我们改变了关系数据库做交易处理的方法，把它由一个集中式系统，变成一个分布式系统。另外，更大的意义不是 OLTP 本身，而是在 OLAP 上。大家可能觉得奇怪，你做的是 OLTP benchmark，它的价值怎么会在 OLAP 上呢?

首先介绍下数据库和业务应用的背景。可以看到当前的集中式数据库所面临的挑战，要做数据库产品的研发首先要有业务需求，数据库经过一些年的发展，到了 80 年代中期自动提款机出现之后，数据库非常重要的特性开始得到普遍的应用，就是在线的交易处理。

后续很多年数据库就围绕这个继续发展，之后企业发现还需要商业智能：需要报表，需要对销售结果进行探讨、进行对比、进行分析等，所以数据库除了在线交易处理的能力之外，还需要在线分析处理的能力，传统的商业数据库在这两个能力上其实做得非常好。

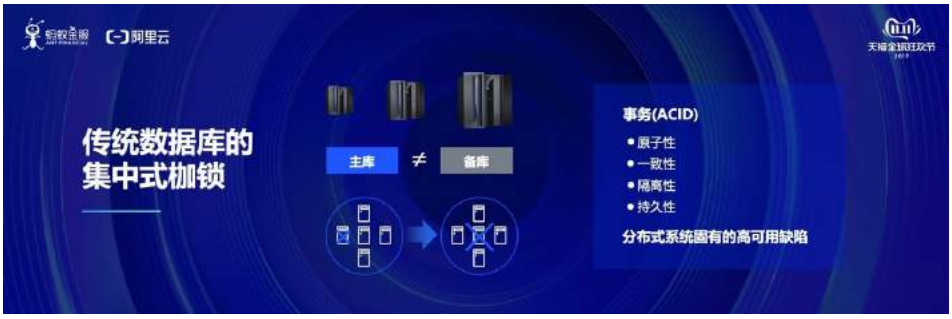
但最近这些年情况发生了变化，原来由同一个关系数据库做的 OLTP 和 OLAP 这两件事情变成了由两个系统来做：关系数据库分库分表继续做在线交易处理，数据仓库则做商业智能分析即在线分析处理。

为什么会出现这样的情况？因为互联网。互联网在短短几年时间里，把整个交易量、数据量翻了几百倍几千倍，发展最快的单个硬件仍然赶不上这个速度，单个硬件就算能够有这个处理能力和容量也肯定不经济。

这样一个系统变成两个系统带来很大的不便。首先，数据仓库本身没有数据，数据还得从交易处理数据库来。所以还得架一个桥梁，把数据从交易数据库通过 ETL 即抽取、转换然后加载到数据仓库，而且这个本质是非实时的，否则的话数据仓库就成为了交易数据库。

其次，交易数据库分库分表后也带来了很多挑战，比如订单号需要全局唯一，如果只有一个数据库这件事情很好办，多个数据库是需要在订单号中加一位代表哪个分库库，而每个分库能做到唯一。当你分库变成两位数变成三位数怎么办？这是业务扩容，如果是业务缩容呢？所以业务需要做很多的改动。

第三是数据仓库本身，数据仓库天然是面向某个主题的，从来没有听说过关系数据库面向某个主题，关系数据库就是关系数据库，你可以根据需要建索引，可以根据需要建物化视图等，但数据仓库只能是面向某个主题的。如果有多个不同的主题，就要建好多个数据仓库，尽管可以把相近的主题合并在一起，但这并不能改变数据仓库面向主题的本质，它会造成大量的数据冗余。另外一个问题是时效性，因为数据仓库本质上不是实时更新的。



交易处理数据库不能扩张，是因为它是集中式。集中式的根本原因还是由于交易处理最重要的一个特性，即事务的 ACID，原子性、一致性、隔离性、持久性。数据库发展了半个多世纪，做交易处理的一直都是集中式系统。

另外一个原因是在线交易处理系统要求非常高的可用性、可靠性，分布式系统有一个固有的缺陷就是可靠性：多台机器在一起的时候，整体可靠性是指数级下降，除非你有特殊的技术。比如当你把一百台 5 个 9 机器放在一起的时候，整个系统只有 3 个 9 的可靠性，任何一个关键业务都不敢使用 3 个 9 可靠性的系统。

这两个原因使得多年来交易处理的系统一直是集中式。我们做 OceanBase 分布式的交易处理系统的时候，就出现很多的质疑，这个质疑声一直到去年还有。最

后公司下决心说好吧，我们就做一次在线交易处理的 benchmark 的认证，这个 benchmark 不仅是跑分，首先你要证明你的系统能够做交易处理，能够满足事务的 ACID，原子性、一致性、隔离性、持久性，只有通过 ACID 测试才能进行下一步测试，所以这个 TPC-C benchmark 不是像有人说的那样，堆砌机器就可以跑个高分的。

假如图中这个球代表一百块钱，金融场景里最常见的是 A 转一百块钱给 B。这个转账过程最大的困难是它的过程必须是原子的，即没有中间过程：这个钱只要 A 这边转出去，B 一定就收到了；如果 B 没有收到钱，同一时刻 A 一定没有转出这个钱。

如果这两个账户在一台机器上，这个事情可以有比较成熟的做法；如果是在两台机器的话，这件事情变得非常困难，怎么样协调两台机器同一时间做这个事情？数据库设计者把这个事情做了两个阶段，第一阶段要检查 A 帐户，看它能不能转钱，也要检查 B 帐户，看它是不是存在，是否能够接收转入。这些检查如果有一个不合格转账就要取消掉，比方说没钱拿什么转账；这些检查如果都合格了就进入第二步：通知 A 扣一百块钱，通知 B 加一百块钱。

这个做法其实有一个大的缺陷：如果第一阶段检查都是好的，在第二阶段 A 这个机器突然出了一点问题，怎么办？ B 那边一百块钱还加不加？按协议第一阶段都正常，那么 B 的一百块应该加，但假如 A 这台机器彻底坏了，拿一台备机来，备机没有这个转帐信息，它根本就不知道曾经有这个转账，这样整个系统里多了一百块钱出来，不知道哪来的；如果 B 的一百块钱不加，假如 A 这台机器只是 CPU 负载高、网络特别忙阻塞了一会儿，过一会儿又正常了，把这 100 块钱扣掉了，B 不加这一百块

钱也不对，所以就出现 B 加也不对，不加也不对，这导致很多年来没有分布式数据库能够用来做交易处理。

是否有个交易处理的系统能够做到随时可以扩展也随时可以收缩呢？这正是 2010 年公司立项做 OceanBase 的目标。

OceanBase 技术方案

OceanBase 项目最早做这件事情首先是有市场驱动的，我们的访问量、数据量都比以前涨了几十倍，甚至几百倍，用传统的数据库很困难了，或者说买不起数据库了。

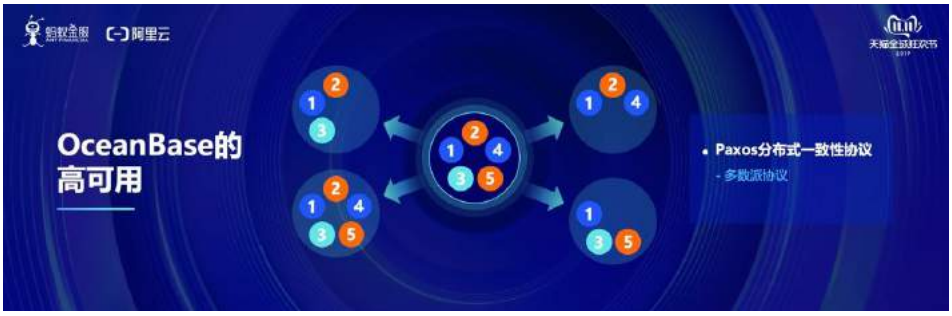
第二，因为在线交易处理是一个实时系统，不能停，否则大家拿支付宝吃个饭、坐个车都不行。

第三，数据库的数据还不能出错，可是软件哪能没有 BUG 呢？所以一个做实时交易处理的数据库系统不只是研发出来的，更是用出来的。当时的淘宝和支付宝就有成千上万的数据库，有这么多的数据库对于这个项目来讲有两个好处：一是经济价值足够大，不用给商业数据库系统付那么多钱；二是这么多的业务提供了一个土壤，让新的数据库成长起来。我们总是讲农村包围城市，总能找到相对边缘一些的业务。

OceanBase 项目一开始的时候，就设定了两个重要的目标：

1. 这个系统要能够水平扩展；
2. 这个系统必须高可用的，尽管使用普通的硬件。

现在让我们看看 OceanBase 是如何解决高可用的问题的。



这个图是传统数据库主备镜像的示意图：主库做事务，并同步给备库。如果要求备库跟主库完全一致，那么每一笔事务都要实时同步给备库，如果备库出现异常或者主备库之间的网络异常，那么主库上的事务就会积压，并且会在很短时间内撑爆主库，然后导致业务不可用，这可能比数据差错更糟糕。大家会问数据仓库也是分布式，它怎么不担心机器坏？根本原因是数据仓库的数据不是实时更新的，如果出现上面的异常，它可以暂停更新。

我们的做法是增加一个备库，主库同步事务到两个备库，只要一个备库收到，加上主库自己至少两个库收到就可以。这个里面关键是多数派，每一笔事务至少在 3 个库中的 2 个库落地，任何一个库坏了，哪怕主库坏了，每笔交易至少在一台机器上存在。我们通过这个方法，把系统做到高可用。

同时坏两台机器的概率，如果是自然损坏确实很低，但如果有人为因素，就不一定了。比如说人为把机器关掉，换一个组件做升级，加上自然损坏，可能出现同时两台机器故障。所以比较重要的业务会写五份事务日志，有三份数据或者是四份数据。即使人工关掉一台机器，自然再故障一台机器，整个业务系统还是可用的。

回到刚才的分布式事务，OceanBase 的方法是：我们把原来的每一个物理节点换成一个 Paxos 组，相当于换成一个虚拟节点，这个虚拟节点背后有三 / 五个物理节点。根据多数派成功协议，三 / 五个节点里有两 / 三个节点写成功这个事务就被判

定为成功。实际上使用了这样一种方式解决了我们提到的万一有一台机器故障，两阶段提交就没办法推进下去的问题。我们就是通过这样一些看上去相对简单的方式解决了分布式事务的问题。

OceanBase 比较早在建设银行就有了一些业务。蚂蚁金服绝大多数的数据库都在 OceanBase，还有一部分在持续迁过来。阿里巴巴是最早立项目用的，后续包括网商银行、南京银行等金融机构也在把一大批业务迁到 OceanBase 上。西安银行是今年发展起来的业务，已经把 Oracle 业务迁移到 OceanBase 上。

TPC-C 基准测试

TPC-C benchmark 诞生于上个世纪 80 年代，ATM 自动提款机问世以后，数据库厂商都希望推销自家的在线交易处理系统。各个数据库厂商在在线交易处理的 benchmark 上各自为政，一直没有一个统一的规范，既缺乏足够的说服力，用户也无法在各个系统之间进行参照和对比。

就在这时，Jim Gray 联合多位学术界、工业界的权威人士提出了 DebitCredit 标准。标准虽然出台了，但是数据库厂商却并没有严格按照标准测试，而是肆意篡改标准让自己跑出更高的结果。这就好比有了法律却没有执法的队伍，每个人都按照自己的理解来解释法律和执行法律。

Omri Serlin 非常了不起，他说服了 8 家公司成立 TPC 组织，并且制定了 TPC 系列标准，相当于立法；同时 TPC 组织还负责监督审核测试过程和测试结果，相当于执法。从这之后，数据库领域各自为政的 benchmark 才有了一个统一的标准。

TPC 的 Benchmark 后来不断的修订，这些年修订了很多版本。一方面适应业务的变化，另一方面是软件硬件的变化。即使放在今天来看，它还是一个普遍适用的场景，不管是金融、交通、通讯等，还是一个非常普遍适用的场景。

TPC-C 测试一共五种事务。里面最多的是订单创建，又叫交易创建。TPC-C 模型是一个销售模型，最多是 15 件商品，平均是 10 件商品。模型是以一个仓库为单位，每个仓库有 10 个销售点 / 仓库，每个销售点服务 3000 个顾客，这个测试是模拟其中某一个顾客到销售点买东西，买的东西可能是 5 件，可能是 15 件，因为买的东西不一定都在本地仓库里有，所以每件商品假设有 1% 概率在别的仓库，每个订单创建的事务如果在分布式系统里有 10% 的概率是分布式的事务。还要模拟整个订单创建里面有 1% 订单是要回滚掉。整个性能指标 tpmC 是每分钟订单创建的数量，假设 100 笔事务其中有 45 笔是订单创建，还有 55 笔是另外的，其中订单的支付是 43 笔。订单支付里面有 15% 概率不在本地仓库支付，要到异地仓库支付，也变成分布式事务。

这是 TPC-C 测试的模型，模拟一个人到一个销售终端买东西。你的请求会发到一个应用系统去，可以想象应用系统是一个简化版的淘宝或者支付宝。然后你的请求会发到数据库里去，整个应用系统和数据库都要公开，你用什么机器，机器配置是什

么，包括价格都要公开。终端模拟器不要求公开。这里面有很多硬要求，仓库里面有 9 张表，每张表有多少数据，每个数据有什么样的分布都有规定，如果你不符合就不能进行测试。

每个仓库最高只允许达到 12.86 个 tpmC。我们做的 6000 万 tpmC，大概按照这个比例大概需要 480 万的仓库，整个算下来数据 336T 左右，我们的数据是乘以 2 的。规范定义的系统单个部件允许失效，如果用了共享存储，就是说共享存储里允许单个部件失效，大家知道共享存储里单个部件失效，共享存储肯定不会失效，我们没有用，我们就是用的虚拟机。如果想通过这个测试，只能把每份数据写两份，这样任何一台机器坏掉，我们的系统还能正常工作。

另外，你在做性能之前必须先测功能，功能有很多，其中比较关键的是要证明你满足数据库事务的功能，就是原子性、一致性、隔离性和持久性。隔离要求串行化，这也是比较难的事情，尤其是分布式数据库。今天分布式数据库中除了 OceanBase 可以做到，还有就是 Google Spanner。

另外，跑性能则有两个要求：第一，要求 8 小时稳定运行，没有任何人工干预的运行；第二，性能采集至少进行 2 小时且期间的性能波动不超过 2%。这些都是实际生产系统的要求。这两个小时用来做性能采集，看这两个小时里必须保持着前面的比例，订单创建多大的比例，支付多大的比例，在这个前提之下把两个小时所有订单创建数给记录下来，然后再 ÷2 小时得到真正的 tpmC 值。OceanBase 做了 8 个小时，审计员看到我们的结果觉得太高，所以，OceanBase 整个性能采集跑了 8 个小

时，而且整个波动小于 0.5%，因为我们不想留下任何给别人质疑的空间。

现在结果在公示期，有 60 天，60 天别人说这个结果有不符合规范的地方，或者弄虚作假的地方，你必须要站出来证明自己，我确实符合这个规范和符合标准。60 天之后结果就是所谓的 Active，这个时候只要在你所在的地区，任何人都可以以公布的价钱买到这个系统。

很多硬件设备三年之后都不生产了，所以它三年之后就认为这是一个历史的记录，即为 histroy，记录还在那儿，还是有效合法的，但是你买不到系统。

我们比较一下历史上最近的三个结果：

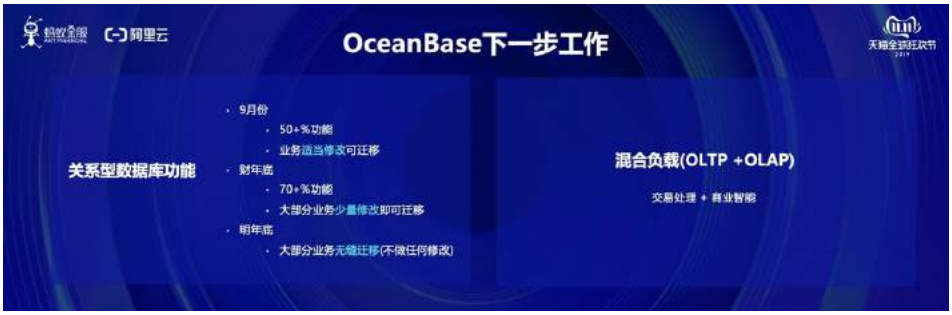
第一，2010 年 8 月份，那个时候 OceanBase 刚刚立项，那个时候还想着怎么样活起来。在 2010 年的时候，DB2 用 3 台 Power 和 30 台存储做到了超过 1000 万的结果。结果 4 个月不到，Oracle 用 27 台 Sparc 和 97 台存储做到了 3000 多万的结果。存储是一个更大的瓶颈。

很多人也在问：为什么 Oracle 后来这么多年没有做？ Oracle 不是没有做过，Oracle 在 2012 年做过一个结果，当时拿 X86 机器做的，做了 500 多万。2013 年又做过一次，也是单机，拿一个更好的工作站做了 800 多万。Oracle 已经做了 3000 多万，为什么做 500 万、800 万这个结果呢？其实我自己的看法是对其他厂商起到威慑作用。这个里面 27 台工作站，平均一台 100 万多一点。2012 年、2013 年做了 500 万和 800 万如果你再做，我可以用单机 800 万做，哪怕不是线性的也是

个很可怕的结果。除非分布式有突破，否则没有单机数据库能够达到 Oracle 这样的性能结果。

我们还有一个变化，没有去买大量的硬件，最后用了 204 台的数据库服务器，还有 3 台是管理节点和 3 台监控节点，一共 210 台。我们用了虚拟机，虚拟机跟物理机相比，内存、CPU 都提高 50%。我们这个结果如果在物理机上跑，会有 50% 的提升，因为虚拟机还是有一定的消耗。

应用系统用了 64 台的服务器，这都是要求披露的。有人在质疑你们这么多钱测试一次 3.8 亿，谁玩得起？ 3.8 亿是说假设一个用户买下系统并且用三年，包括硬件、软件、技术支持全部在内是 3.8 亿。整个三年的硬件成本是租虚拟机的成本，整个成本大概在系统里面只占五分之一，测试成本大概用租了 3 个月的机器，找阿里云租的，本身你的硬件成本只占 3.8 亿的五分之一不到，这还是 36 个月的成本，实际上我们只用了 3 个月的成本。大家其实可以把我们整个硬件投入能估算出来的。



我们这个测试更大的价值不在于 OLTP，是想跟别人证明，我们在分布式数据库能做交易处理，更重要是想证明这个数据库既能够做交易，也能做智能场景分析。绝大多数场景下，用户、客户不再需要搭建一个数据仓库，去复制数据、去导数据。否则这样一个系统只是为了做交易，其实它没有足够的价值。

最后是一个简短的小结。80 年代后，交易处理和商业智能就成了关系数据库一个核心需求，但是集中式的架构这么多年发展下来，扩展能力有局限，尤其有了互联网、移动互联网出现以后。TPC-C Benchmark 本身无所谓我们做了什么，别人做了什么，它定义的是业务需求。它定义的是订单创建、订单支付、订单查询、订单配送，而这些都是业务需求，只要满足这个业务需求，哪怕拿文件系统去测并且能够测出一个好结果也是本事。

通过这个测试，更多是想证明我们是有史以来第一个分布式数据库具备交易处理的能力，这是以前没有的。也是曾经 80 年代、90 年代末宣判做不到的，OceanBase 接下来做的最重要事情不仅是关系数据库的功能，要做的是把商业智能的能力做进来，能够向客户提供所需要的交易处理和商业智能分析。谢谢大家。

蚂蚁金服资深总监韩鸿源：企业级数据库平台的持续与创新

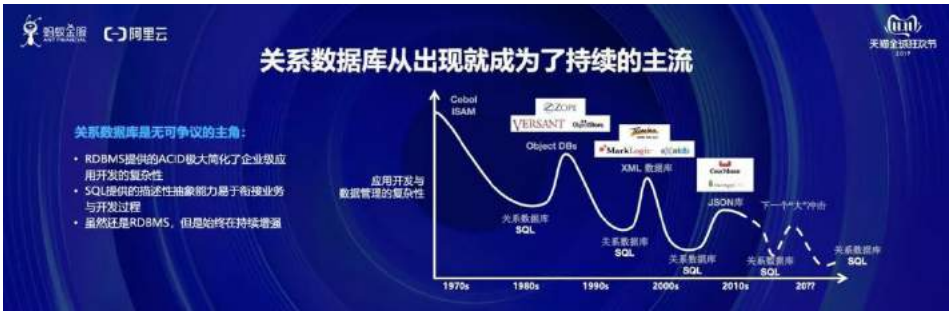
作者：韩鸿源

蚂蚁金服研究员韩鸿源近日分享了《企业级数据库平台的持续与创新》，以下为演讲实录：

在今天企业级的市场里，数据库是根本的基础支持能力之一，传统数据库技术正在面临各种各样全新的挑战，面对这些挑战的时候大家都在寻找一些新的解决方案。这些新的解决方案从什么地方来呢？当我想要替换非常成熟的被大家广泛使用的平台的时候，怎样让客户放心的去替换？在企业级数据库基础平台上，后续可能会是什么样的发展方向？这也是我今天想和大家探讨的内容。

关系数据库已经发展很多年，今天所有企业里面的业务系统，支撑核心的业务系统大部分都跑在关系数据库上面。原来传统企业里的业务系统接入终端数量很有限，不管是自己内部业务系统，还是对客户前端柜台做业务系统，终端数量都非常有限。基本上，系统的扩展能力可以通过数据库服务器的垂直扩展都可以满足要求。

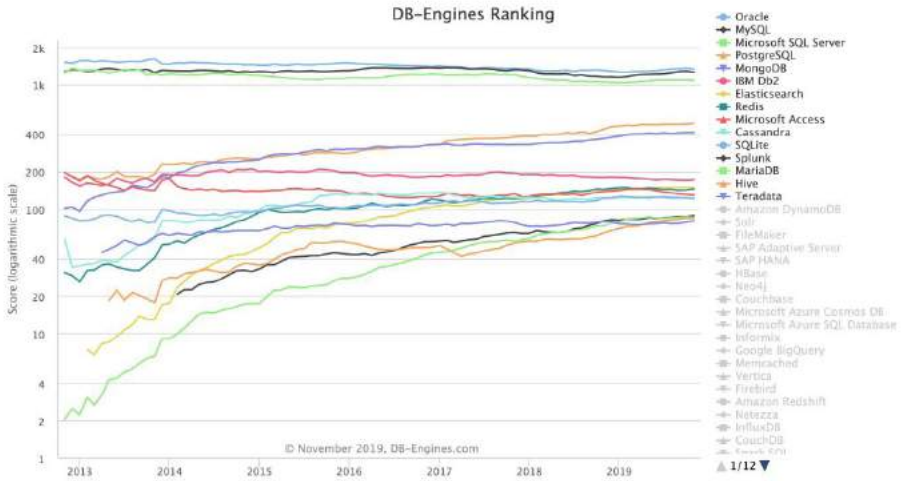
随着移动互联网发展，今天的业务系统不管是银行、政府等，已经广泛暴露在大部分移动接入的客户面前。当这种情况出现时发现很大的变化，传统的系统垂直扩展是不可能解决这个问题。那么如何解决呢？这就衍生出很多不同的解决办法。除了传统的关系数据库持久化处理数据方式之外，演化出非常多的新的数据方式，所谓多态数据持久化等方式不断冒出来。



当所有这些东西来了之后，大家发现回到真正的业务逻辑上去看，关系数据库有它很突出的长处，它可以以一个数学上完善的方式帮助客户有效的构建业务系统，这些能力当你去放松对关系的严格要求之后，很多东西很难满足。包括客户已经有的业务系统，怎么样持续运营下去？怎么样保证客户已经有的这些严谨业务逻辑持续运营

下去？所以带来很多新的挑战。

传统的企业客户不只是商业企业，泛指企事业单位和党政军范围，一般指具有一定规模且有重要性的实体。那么，对企业级系统和平台的要求也包括很多：1）高安全；2）高性能；3）高可靠性；4）高可用性；5）高开发效率，低维护成本；6）高可扩展性。这些对企业来讲都是很核心的东西。



上图底下是一个横轴线，从 2013 年 -2019 年排在最前面几个数据库的活跃程度非常稳定。今天去看前三个最活跃的数据库依然还是关系数据库，关系数据库还是今天整个企业数据管理平台的主流，不但是主流，而且支持所有主要的业务系统。

主流关系数据库相对稳定，这些数据库受到很多的开源的冲击。对于绝大部分企业来讲不是做开发和软件的，需要是平台具备的这些能力，你要高可靠、高可用，能够帮助用户持续发展。只要这个产品足够可靠好用的情况下，客户可以自己把运维接起来的情况下，是没有必要看源代码的。今天我们谈企业级的时候，可能看的更多的是怎么样支持客户有效运营自己的业务。

上图是数据库的发展历史。最早有层次数据库、网状数据库等，当关系数据库出现之后，由于它们突出的特点，基本上主要的业务系统都迁移到关系数据库开发模式中去。从我个人来看，我经历大概 20 多年的历史，关系数据库是所有数据库里被判死刑次数最多的一个技术，到今天为止不但没有死，而且还在不停的焕发新活力。

RDBMS 真正的价值如何体现？我觉得，首先数据库里面强调的 ACID 帮助应用开发，简化了应用开发复杂性；第二点，SQL 这个写法很关键，接近于自然语义的写法，最大优点做业务开发的人写出来的代码可以让做业务人看得懂，带来的好处是大家沟通很方便，写出来的代码可读性可维护性非常强，所以摆脱这些技术是很困难的。但是关系数据库不是一成不变的东西，从集中式到分布式是一个大的发展方向，很多时候这些东西为了突破原来的限制和技术瓶颈。

互联网与互联网公司带来的变化是，用户访问量变化，以及很多新技术的探索和创新。最早互联网公司里有相当多互联网公司做的业务跟传统企业业务有非常大的差异。当你做这些业务的时候，有些业务是有标准答案，有些业务没有标准答案。当你在网上做搜索的时候，你搜之前肯定不知道搜索结果是什么？当你转一笔帐的时候，你转之前一定知道转帐结果是什么。这两件事对于数据库平台的支撑能力有完全不一样的要求。互联网公司面临压力大了之后，需要系统有非常强的可维护性的要求，很多东西是自动的维护和自动高可用的管理，这一块是一个很大技术变化的出发点。比

如说高可用，传统企业的高可用基本靠半人工、半机器方式去做，基本不会完全相信机器，自己把高可用做的很完善。

举例来讲，非常多的金融机构都做了容灾系统，容灾系统切换这种决策没有人敢让机器做自主决策。当你的规模非常大的时候，这种管理很难去做到，触发怎么样实现自动高可用，真正让系统具备完整的判断能力。

当互联网发展到金融这个领域时，刚才我说的问题就很麻烦，很多用户有很多的并发请求，做查询不知道准确结果，做金融一定要知道准确结果，做错任何一点用户直接都会找你算帐。这个里面需要很完备的基础设施的支持，才能帮你支撑这么好的金融业务。

下面，我们希望跟大家探讨一下，**后续企业里的数据库技术会有什么样的发展趋势?**

首先，分布式已经是不可避免的潮流，这个里面有几方面的原因。单一的大服务器加存储的方式扩展能力有限，无法支持企业的持续向前发展。在今天的云环境里面，大家可以看一下市场上不管是哪一家主流的云供应商，现在已经没有任何一家云供应商会让你把服务器连到高端、高性能的存储上去支持数据库来运行。如果你想在云环境里运行数据库，必然要选择其他的实现方式。所以你看在测试的时候，包括系统架构设计方面，实际上是跟云的大趋势是一致的。今天，硬件发生了很多大的变化。像大家质疑我们的 TPC-C 测试结果一样，9 年以前跟现在的硬件有很大差异。

如果我今天给你 200 台服务器，装一个数据库上去，你根本装不上去，更不要说运行结果出来，这不是那么简单的一件事，能有效地把这些新的硬件用起来对软件是一个非常大的考验。如果大家看还在使用中的绝大部分主流关系数据库，它们有一个很大的特点，它们都设计在 30 年以前。30 年以前设计数据库的时候有两个假设，所有硬件内存都是很小的，所有的存储访问速度都是非常慢的，所有的数据库基本今天用的主要数据库都是从那个年代发展起来的，这两个限制条件给它加了很多枷锁。

如果大家尝试过可能会发现，我有一个 oracle 数据库今天装 256 个内存，明天扩 512G，性能能提升多少？能提升一倍吗？事实上能提升 1% 就不错。什么原因？它的软件架构设计决定它不能有效使用新的硬件能力，当今天不停的有新硬件技术出现的时候，需要新的方式把硬件能力用起来，带给用户更好的系统，带给用户更好的回报和更简单的管理。

从云发展趋势来讲，其实数据库是最适合于云化的服务。怎么样能够以云化的方式有效的支撑客户去使用数据库也是一个很大的挑战。

最后，管理的规模也很大，系统也很复杂，怎么样把更多的人工智能带来的优点体现在系统里面去，帮助系统自动去运行，更好的自动去调优。这件事情今天说起来容易，最大挑战是要有足够大、足够广泛的使用环境和使用场景，才能帮你积累到数据，才能算出你想要的模型来。

那么，**新的企业级数据库需要具备什么能力呢？**

首先，数据库最好对硬件不要有特定的依赖，这样会阻止往云方向去发展和做优化。其次，今天所有的企业面临一个发展方向，都是怎么样从传统企业架构转到云原生架构，数据库怎么样支持用户转换前和转换后的平滑过渡。然后，在今天的企业环境里面，很多负载的变化是突发的。你需要能够**在保障数据的情况下，在不同的运行环境实现灵活迁移。**最后一点，数据库是非常适合云计算提供的服务，怎么样能够去**真正把底层硬件能力发挥出来**，比如在设计之初就要考虑多租户的环境，怎么样在多租户环境有效使用资源，支撑所有混合负载的能力。这些加在一起实际上是构建下一代数据库平台一些必须考虑的因素。

OceanBase 对新技术的探索

回到 OceanBase，我们在这些方面做了很多探索。

首先是高可用性。在今天来讲六级的高可用性已经是非常高的可用性，绝大部分机构实现不了这个级别，但是我们今天能实现远远超过这个级别，能够在 30 秒内实现自动恢复。这些靠的是在技术上怎么样把新技术有效用起来，比如说 Paxos。在传统数据库里不会有人去用它，在新的分布式场景下来讲，利用这些新的技术其实它还能帮你实现全自动的高可用。Paxos 这种自动投票的机制带来的优点是系统在不需要在外部干预情况下，把失效的东西替换掉之后持续去运行，在今天大规模运行环境里面是不可或缺的一个东西。

刚才讲到，今天的互联网带来的压力对于业务系统压力变大非常多，靠一个数据库一个系统支撑所有业务，一定不可能。在我们系统运行环境里，经过这么多的实际环境输入之后，今天可以给用户一个很灵活的选择，可以让用户去选择数据库的部署粒度，你可以选择分库分表，也可以选择单库。**把选择权给到用户，用户可以根据自己的需要从不同的方式之间做过渡的融合，都可以帮助用户持续往前发展。**

为什么要走向分布式数据库？传统都是最左边集中部署方式，它的优点是 ACID 不用担心，缺点是想扩的时候扩不上去。为了解决扩展的问题，大家才来做分库分表。分库分表绝大多数情况用中间件的方式来实现。这打破了数据库的边界，又同时引入了很多新问题。最大的问题是对应用不透明，如果你原来是一个复杂的业务应用，想适应分库分表的时候，对应用的改造工作量非常大。

为了解决这个问题，我们做了原生的分布式数据库。最简单的描述，你可以把一个分布式部署和运行的数据库完全当成一个集中的数据库来用，对你来讲不会有任何的差异。就像阳老师讲的，TPC-C 测的是一个系统的业务处理能力，对外表现来讲

TPC-C 所有的检查标准时，它跟单一的数据库对外表现是一样的。这在今天的分布式数据库里是最难实现的一点，怎么样能够把一个分布式的东西，表现给用户用的时候是当作集中式来用，能力有提升，但是使用上不增加复杂性。

我们去看传统数据库的时候，往往强调是 ACID 属性。如果只为了满足 ACID，可以很简单的做到。因为数据库可以停。为了保证 ACID，我们可以在有异常情况出现的时候，把整个系统停掉之后保证 ACID 不会出问题，但是用户用不了系统。在今天，保证系统高可用的同时，高可用反过来可以帮助 ACID。

原来传统系统里使用两阶段提交时，最大问题不是两阶段带来的系统消耗，是两阶段跨系统做交易的时候，一旦有一个参与者出现不可用，整个系统没有办法持续运行，状态不可知的时候，没有办法保证系统一致性。刚才讲的自动 30s 之内的 RTO 的恢复会发生很大的作用。当整个系统所有的交易参与者可以在很短的时间内恢复出来的时候，它是不会把业务挂起，可以确保业务持续运行下去，消灭了传统分布式系统里非常大的一个弱项。

今天整个市场上讲，关系数据库是一个全球范围内早已经划分完势力范围的市场。为什么今天还有新产品出现，是因为很多因素加进来之后，促成了很多新的变化。蚂蚁内部有非常多的使用 OceanBase 的业务，这些业务经历过去七、八年的发展，在这个过程中 OceanBase 增强自身的能力，消除很多问题，在一个大规模的复杂环境里面经历这么多年磨炼之后，从 2017 年我们走出来服务于外部用户。

希望以后有更多的用户给我们机会去尝试 OceanBase，我们也希望这个产品帮助大家解决很多现实中面临的问题，也欢迎更多的企业和组织加入进来。

关于 TPC-C，我想说，首先，**TPC-C 是目前国际上唯一具有公信力的数据库功能与性能结合的公开检测标准。**因为所有市场上主流的玩家，原来都在这个标准上发布过测试结果，即便这个测试模型源于 20 年前，但是所有结果都有意义。而且 TPC-C 的模型定义如果大家深入研究的话，里面有很多科学的东西。

第二点，TPC-C 测试大家往往看到很多误导性的信息，我在家里跑一个结果，单机 TPC-C 可以跑 150 万，200 万，300 万，这件事情没有意义，TPC-C 测试里面除了跑的性能指标之外，它有前提条件。TPC-C 测试过程中的 ACID 的检查，对于分布式数据库是一个非常大的挑战，今天绝大部分的分布式数据库面临这个问题采取的是回避的做法，不是直接解决问题的做法。测出来的结果很多不是有效的结果。我们之所以参与 TPC-C 审计，是为了证明我们的分布式系统是可以像单机数据库一样一分钟处理 6000 万笔新的订单。

第三点也就是在 TPC 认证过的 TPC-C 结果里面 OceanBase 取得的 6 千万 tpmC 排名第一。

第四点，测试是基于公有云通用机型实现的，使用的是和生产系统一致的基础环境。今天最大的变化是传统企业数据库往云环境搬的时候，最大的变化是没有能够匹配原来环境的那么强大的服务器可用。没有那么大的服务器和存储，怎么样解决这个

问题？我们给大家去做了这个证明，你可以用软件的能力实现同样的性能指标。

最后，我不认为 TPC-C 是证明数据库完备性的充分条件，但是它是一个必要条件。当你要接的是这些大型机构核心业务系统时，如果你的数据库没有这种能力，肯定不可能帮助客户简单的把应用迁移过来。

今天，有些时候大家往往关注数据库的一个点，但是整个对于企业来讲走分布式转型这条路的时候，不可能只在数据库一个点上面走。我想说的是，企业整个的分布式转型是需要从上到下，结合中间件和开发过程管理和系统保障管理所有体系加在一起，才能够确保有效的走向分布式转型，真正能够支撑你的业务持续往前发展。

OceanBase 是完全自主研发的分布式关系数据库，我们掌握所有的源代码和系统的设计。在设计系统的时候没有预设限制性条件，没有对特定软件的依赖，没有对特定硬件的依赖，没有对特定系统架构的依赖，这个时候我可以非常广泛的去适配所有新出来的硬件系统和新的运行环境，可以帮助我们去探索更多的系统组合使用的空间。我觉得我们最大的优点是不存在对外部特定软硬件系统及系统架构的锁定性依赖。谢谢大家！

了解更多蚂蚁金服技术，欢迎关注蚂蚁金服技术新媒体矩阵

公众号：蚂蚁金服科技

抖音号：支付宝技术

微博：蚂蚁金服科技

知乎：支付宝科技局