阿里文娱技术 阿里云 开发者社区

阿里文娱在线票务技术大揭秘

抢票秒杀 | 10W+选座 | 刷脸验票

——阿里文娱技术精选系列——

阿里文娱C端技术

关注我们

（阿里文娱技术公众号）

关注阿里技术

扫码关注「阿里技术」获取更多资讯

加入交流群

1）添加“文娱技术小助手”微信

2）注明您的手机号 / 公司 / 职位

3）小助手会拉您进群

By 阿里文娱技术品牌

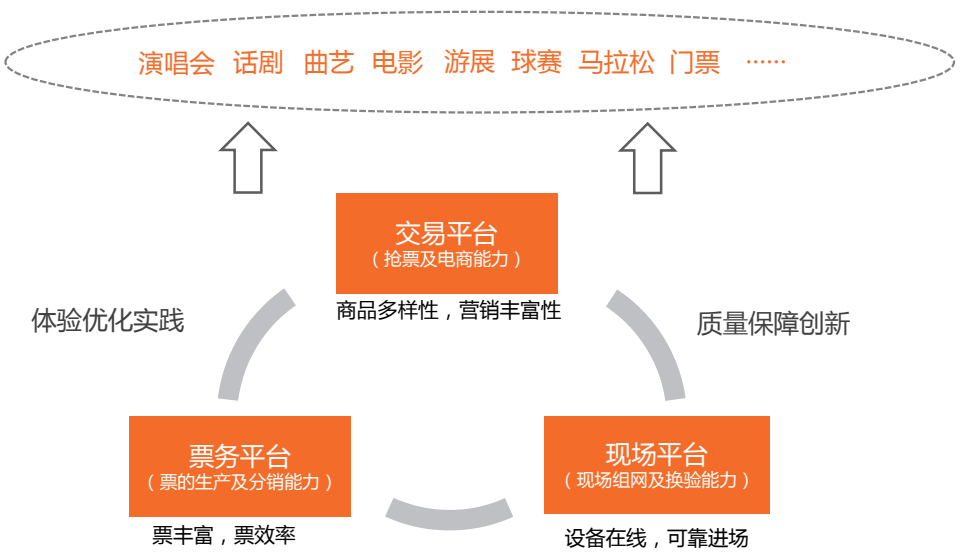
更多电子书

扫码获取更多技术电子书

目 录

序

在线票务是电商领域的一个特殊类目，它既有电商的公共属性，也有很多基于垂直业务领域的特征。同样的，相比机票、火车票等交通类票务，演出、电影在票务领域中又是独具特色，在整个票的生产、销售、履约、现场使用等环节都有不少个性化的需求，技术架构需要能够满足电商属性的同时支持好行业特性。阿里文娱在这个过程中沉淀了许多面向行业的技术能力，在演唱会、电影、话剧、体育赛事、游展等多类目得以复用。



本章按功能领域将这些技术能力分成三大部分，分别是：票务平台、交易平台、现场平台。每部分能力的实现都有一些特色的技术创新和实践呈现出来，曾经面临的诸多问题，如：几万座场馆的高并发选座、库存一致性、常态化抢票支持、现场验票网络可靠性问题等都能从中找到相应解决方案，希望能让读者对现场票务这个行业面临的技术挑战和机遇有较深入的了解。

阿里文娱产品技术平台资深技术专家　解缙

2020.1.20

1

票务平台

10W 座位的大场馆究竟是怎么画出来的？

作者| 阿里文娱技术专家 莫那、阿里文娱高级工程师 土豪

一、背景

1. 网络售票需要画座：购票所见即所得

大麦主要业务是票务，包括演唱会、体育赛事、音乐节等，品类繁多。卖票就要画场馆、画座位，大家都在网上买过电影票，这不难理解。虽然可以拿电影售票做类比，但底层难度差异很大。没有 10W 座的电影院，却有 10W 座的演唱会，而且演出/体育类场馆变化多，座位不固定，场景非常复杂，想随心所欲画出 10W 座的场馆，挑战相当大。

2. 大麦绘座演进：从示意图到实际场景图

大麦以前的绘座系统，是安装版的程序，画座位只能一个看台一个看台的画，看台之间完全无关联，画出来几乎每个看台都长一个模样，座位只有相对位置的示意图，没有角度、距离，更谈不上精确定位。

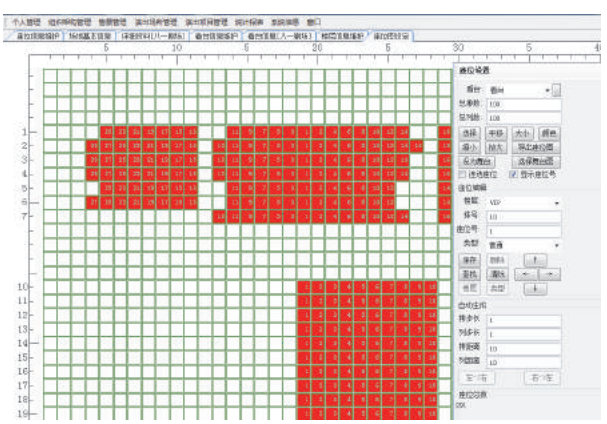


图 1：老版绘座页面（已淘汰）

大麦网从 2017 年，开始设计新版绘座系统。这里没有修补，没有重构，新版绘座完全重来，连技术栈也由.NET 换成了 Java，由 C/S 换成了 B/S。

新绘座以 SVG 矢量图为核心，通过 Canvas 进行绘制，在演进的过程中攻克了大量的性能卡点和技术难点，最终打造成型，堪以重任。

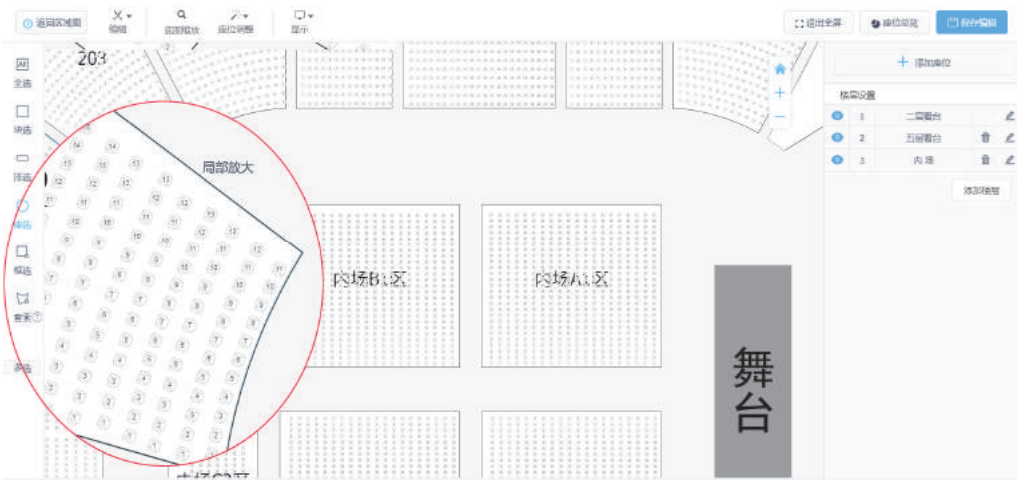


图 2：新绘座页面

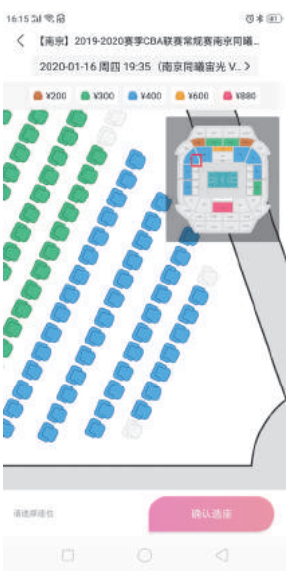


图 3：座位效果图

二、新绘座：Flash 走了，Canvas 来了

1. Flash 已死，来到路口，别无选择

新绘座已诞生 2 年多，现在回首，这条路似乎早已注定。

老版绘座和选座是基于 Flash 的，悲剧的是，Adobe 宣称 Flash 2020 年后，将不再维护，相关技术会在 2020 年底全部退役，大麦的绘座和选座系统，都被迫转型。

Flash 只是原因之一，看过竞争对手的产品，会发现 SVG 是条不错的道路，即使没有 Flash 这一出，大麦网也会朝这个方向迈进。

2. 技术选型

1）任何过度使用 DOM 的应用，都不会快

经过技术调研，发现国外一些场馆座位绘制，选用的是 SVG 方案，每个座位都是一个独立的 SVG 元素。但如果直接把 SVG 搬到浏览器，无法支持几万座位的场馆，因为浏览器无法支持过多的 DOM 数量，并且，一旦 DOM 数量太多，操作一定是低效的，“任何过度使用 DOM 的应用，都不会太快”。

于是，技术同学想到了 Canvas，Canvas 是浏览器上的一个画布，无论上面绘制多少元素，对于浏览器而言，都只是一个 DOM 元素。

对于不了解 Canvas 的同学，我们可以简单做个说明，Canvas 在浏览器上，就是下面一个标签：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

在 Canvas 上绘图，就是使用 JS 获取 Canvas 对象，使用封装好的方法进行绘制。Canvas 画布上的图形变化，完全通过擦除+重绘的方式展现。

那么新绘座的目标就变得很明确了，我们就是要在 Canvas 上绘制出想要的场馆座位图，然后以 SVG 的格式把图形保存起来，用以选座、售票。

2）Canvas 也不是银弹：单个 Canvas 的大小是有限制的，超限之后也会卡顿

选型初期，技术同学使用 Canvas+SVG 做了个 Demo，模拟了 10W 座位的渲染，并实现了拖拽、缩放。但真正作为画座组件开发的时候，发现座位达到 2W 就出现了卡顿，因为 Canvas

的宽高达到一定的数值，就会出现卡顿。于是，沿着化整为零的思路，技术同学将整个画布，分成了多份Canvas，形成了一个Canvas矩阵，通过对每个Canvas的操作，完美解决了单个Canvas过大引起卡顿的问题。

关于 Canvas 绘图组件，大家可以在网上搜到很多资料，这里不再赘述。

3. 新版绘座上线初期：青苹果

刚上线的新版绘座，就像个青涩的苹果，虽然漂亮，却没那么好吃。

最突出的问题有 2 个：第 1 个是变形难用，由于算法比较初级，座位矩阵变形很难满足用户需求；第 2 个是接口速度慢，打开一个 1W 座的场馆，好几分钟，超过 5W，直接崩溃，根本无法支持。

为什么理想很丰满，结果却差强人意呢？根源在于第一版只重功能，忽略了算法效率。与服务端的接口调用，都是整个场馆级别的，几万座位数据，加上关联的看台、票、以及状态等，一个硕大的数据包在前后端丢来丢去，系统不堪重负，用户受尽折磨。

4. 艰苦改进之旅

新绘座上线后，立刻启动了改进优化工程，主要攻克的难关有三点：页面响应时间；座位自由组合变形；打印顺序计算。

1）交互+接口优化，进入秒开时代

首先要解决接口慢的问题，解决效率低的一大法宝：化整为零。

从一次 load 一个场馆的数据，改成一次 load 几个看台的数据。服务端数据随着前端视口（页面可视范围）的变化，逐渐加载，类似地图常用的“拉框查询”。前端交互也从全加载，改为按视口取数据。仅此一项优化，几万座大场馆的系统响应速度，立刻由几分钟，降到了 1~2s，小场馆更是瞬时打开，系统好用了不少。

这里面最重要的一个技术点，就是视口计算，原理如下：

前端首先获取到屏幕视口在 Canvas 画布上的坐标，然后和看台的外接矩形进行碰撞检测，两个矩形一旦相交，就说明该看台已暴露在视口之内，于是就加载该看台的数据。

从接口优化开始，新绘座逐渐走向成熟。

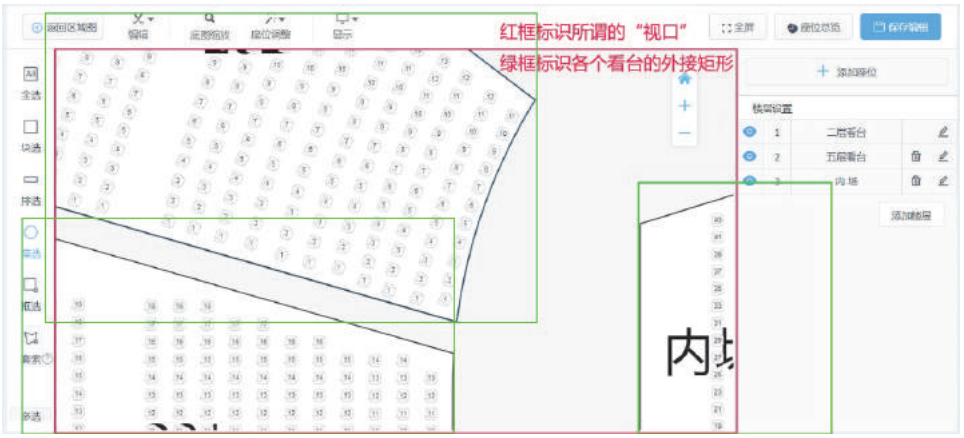


图 4：按视口加载原理图

2）合并座位矩阵，自由变形

座位自由变形包括倾斜、错位、排距、座距、旋转、弧度等多种操作。除了弧度变形，其它基本上是一些数学上的坐标计算，我们不赘述，这里重点说一下弧度变形。

新弧度变形，运用贝塞尔二阶曲线原理，根据用户的数据输入，计算出相应的贝塞尔曲线，再把每排座位，均匀排列到曲线上。下面是贝塞尔二阶公式：

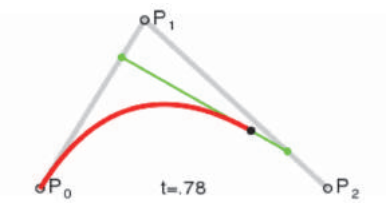


图 5：贝塞尔曲线示意图

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2 \ , \ t \in [0,1]$$

注释：P0、P2 为一排座位的左右端点（一排的第一个座位和最后一个座位）

看似套公式就可以搞定，非常简单的样子。但是这里有一个难点：从图中可以看出，t 为比例值，处在线段 P0P2 不同的比例，所在的弧度位置也是不一样的。

如果所有的座位都在 P0P2 线段上，很好算，但是如果座位之前就是一条弧线呢？中间所有的座位都不在 P0P2 线段上，要怎么算出中间座位的每个比例？

我们通过弧线上的每个座位，做一条 P0P2 线段的垂线，垂线与线段 P0P2 的交点，就是这些座位所在这一排的原始位置，计算出这些原始位置的坐标，根据这些原始位置，就可以算出中间所有座位的比例了。

这样，弧度变形问题就通过贝塞尔二阶曲线完美解决。

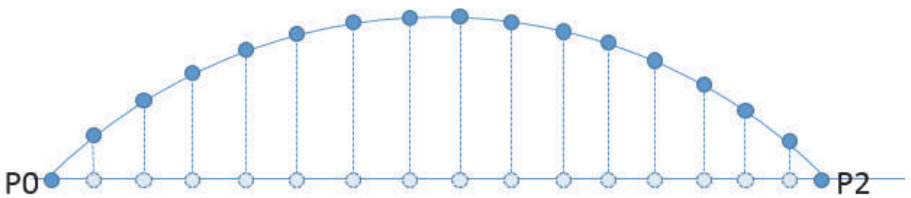


图 6：弧形座位矩阵贝塞尔曲线变形原理图

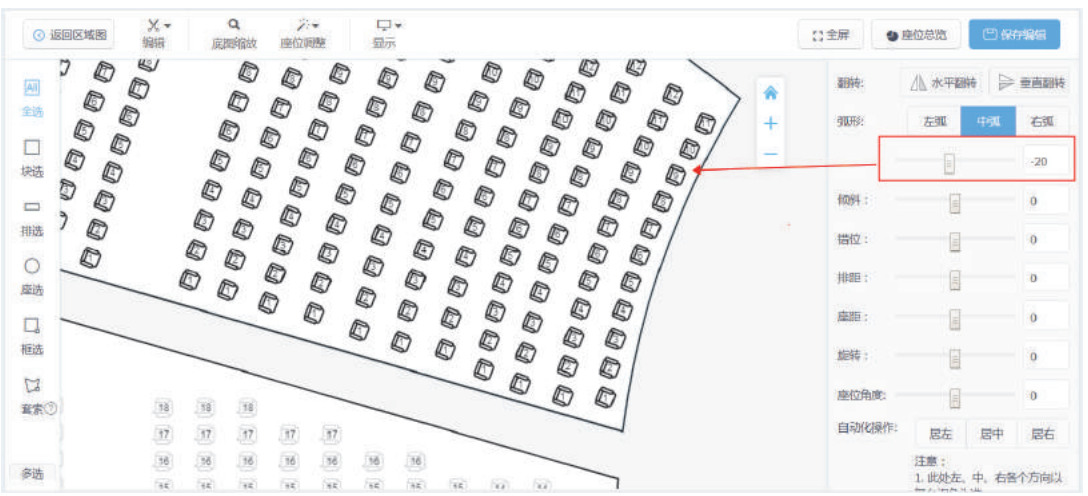


图 7：弧度变形实际操作

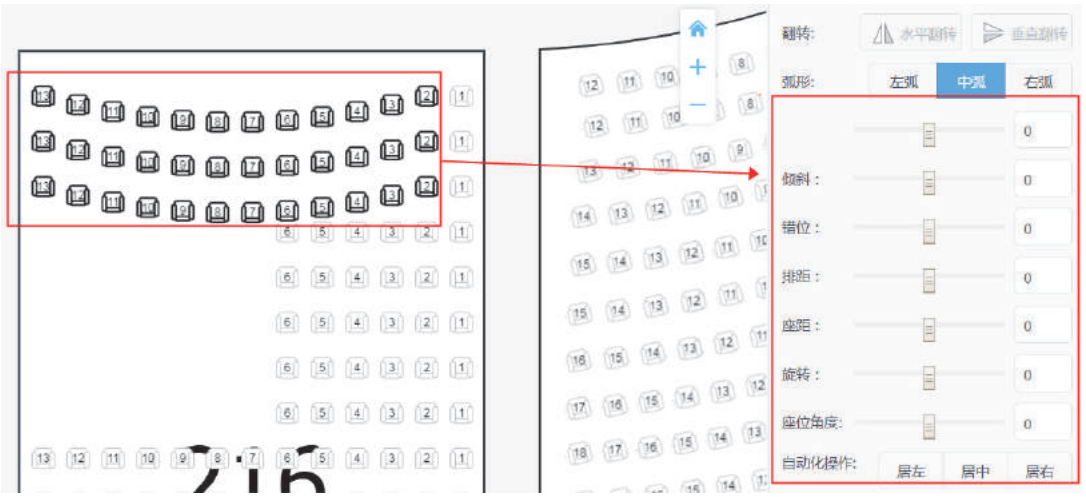


图 8：座位自由组合，随意变形

3）打印顺序计算

“打印顺序”是个什么鬼？

这得从大麦的业务特点说起，主办有时候会批量出票并将票配发给相关人群，有时整个看台一起打印。在配票的时候，需要按照座位的物理位置关系排序，避免座位没挨着、“2 个情侣”被“拆散”的情况发生。举个例子：下图中，主办期望打印票的顺序是“5-3-1-2-4-6”，而不是“1-2-3-4-5-6”，这样他们就可以按打印顺序配票，而不用担心两张票不挨着。那么，在绘座过程中，我们就要计算出座位的顺序，看似简单，实现起来有难度很大，原因只有一个，场馆形状各异，座位排列多样。

图 9：北京奥体中心的某个看台

如果说，上图还能按照座位 Y 坐标对比进行排序，那么下面的几个情形，就不那么好处理了：

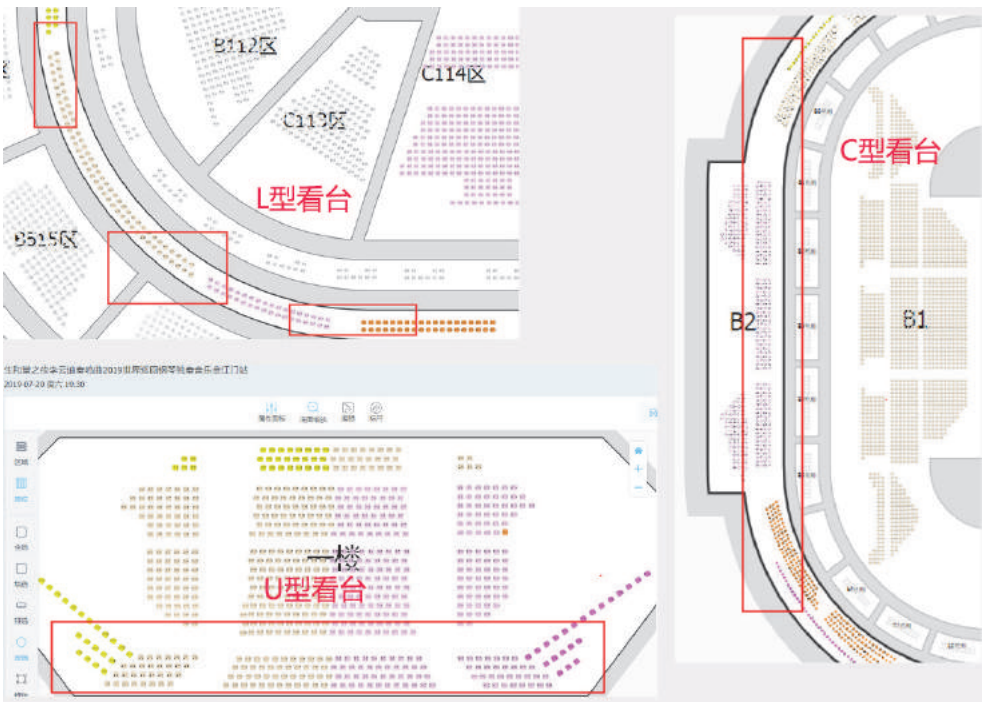


图 10：各种特殊的座位排列场景

打印问题，我们通过场景汇总，对场馆进行分区，最终找到了排序的规律，得以解决。打印问题技术方案原理：

第 1 步：将场馆分成 8 个象限，象限内的座位，已标识出该如何排序（标识出了应该对比 X 坐标还是 Y 坐标来进行排序）；

第 2 步：每一组座位矩阵，取出首排，求首尾座位连线的斜率，然后根据斜率将座位矩阵划分到对应象限；

第 3 步：按照对应象限的排序标识，对比座位的 X 坐标（或 Y 坐标），进行座位排序。

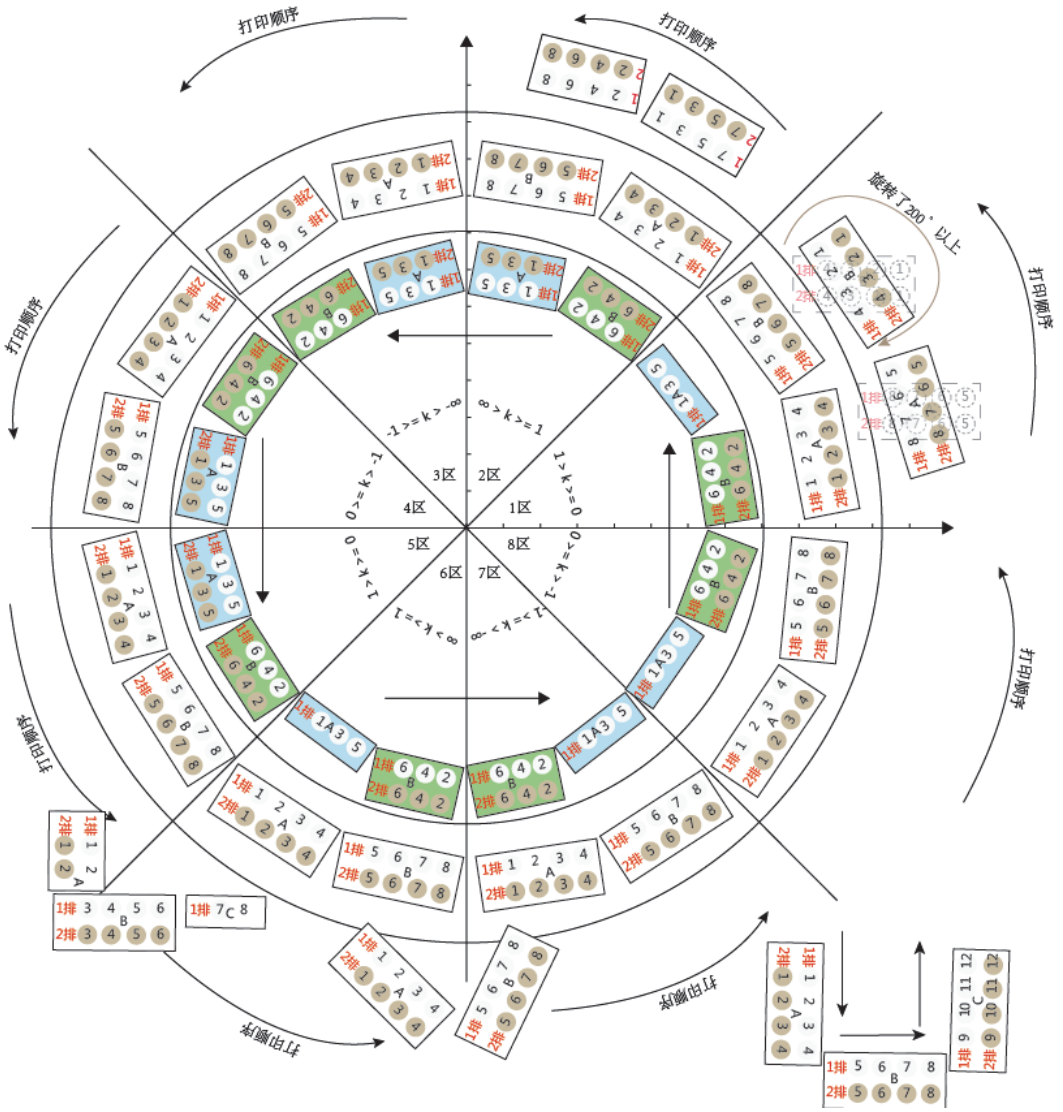


图 11：座位排序原理图

4）小彩蛋之“沙发、角度”

效率、变形和打印 3 个主要问题根解之后，随之出现了大量的产品优化需求，开始着眼于细微之处，小沙发和座位角度就是 2 个典型的功能。这两个功能虽然难度不大，但却在体验上有了一大步的提升。

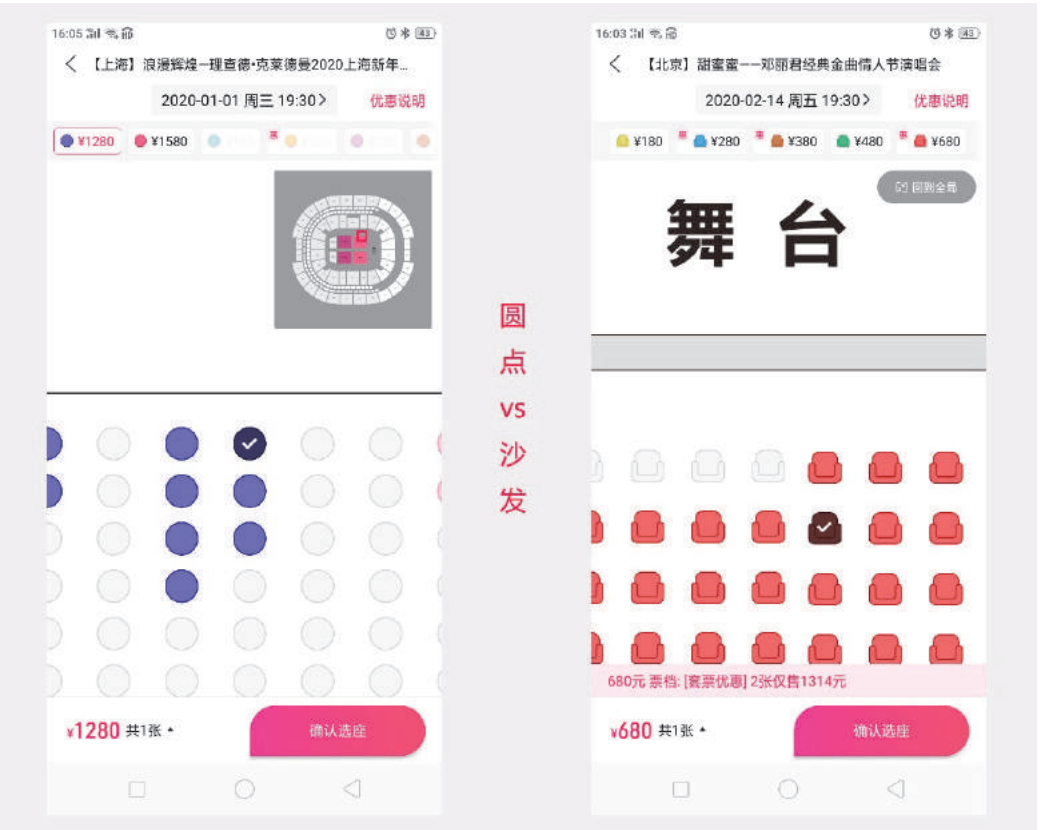


图 12：圆点、沙发效果对比

5）小彩蛋之“撤回”

经过不断优化和添砖加瓦，大麦的绘座系统，越来越像一款专业的绘图工具。好的绘图工具一定需要“前进&撤回”功能。

新绘座系统的撤回功能实现原理：设计一个“历史数据”数组，数组里的每个元素，记录一个操作步骤对应的被编辑座位数据以及座位位置信息，回退时，找到对应操作步骤的数组元素，重绘座位位置，这样就回退了整个操作。因为无论座位相对位置如何变形，本质上，其实都是坐标数据的改变，通过记录和重绘历史坐标信息，就达到了撤回操作的目的。

三、在正确的路上继续前行

到目前为止，新绘座系统已能够承接国内外任何大型场馆的绘座工程，各种细节的优化也日臻完善，效率大幅提升。但产品和技术同学的努力，并没有终止，而是在正确的道路上，继续前行。

以下简单列举几个很实用的功能，供大家参考：

1. 区域编辑

自由绘制矩形、圆形、多边形等各种形状，并自由变形；

2. 一键自动变形

全选看台内的座位，点击“一键变形按钮”，座位瞬间适应看台形状，自动排列；

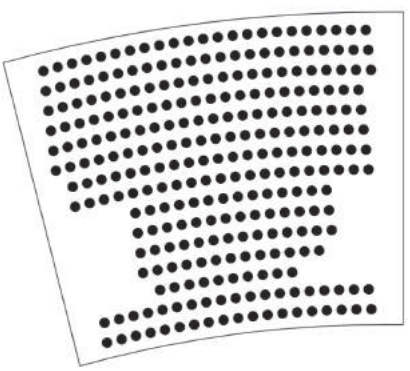


图 13：一键变形效果图

3. 座位复制、镜像

区用户可以自由复制选中座位，并且支持镜像、翻转等多种复制模式，排号、座位号根据设置自动处理；

4. 一键朝向舞台

用户选中一个看台的数据，点击“一键朝向舞台”，系统会自动计算舞台方向和座位角度，瞬间将整个看台座位“摆正”。

10 倍高清不花！大麦端选座 SVG 渲染

作者| 阿里文娱无线开发专家 波涛

一、背景介绍

用户在大麦上购票，需要自行选座。在大型场馆下，如何让 10 万+座位绘制达到闪开？这需要技术在绘制上保证性能流程，在选座渲染上通过技术手段赋予更多可能性。因此，大麦引用 SVG 绘制技术，并根据业务场景下作了很多优化，本文是大麦在用户端的技术方案设计与应用实践。

二、10 万+座位绘制面临以下挑战

1.如何丰富标签样式及属性；

2.SVG 渲染性能优化；

3.SVG 如何与业务场景结合；

4.如何将 CSS 能力应用到 SVGKit，保持（iOS\Android\H5）一致性。

三、大麦 C 端场景下 SVG 应用

1. SVG 介绍

可伸缩矢量图形 (Scalable Vector Graphics)，用来定义用于网络的基于矢量的图形，使用 XML 格式定义图形，图像在放大或改变尺寸的情况下其图形质量不会有所损失，是万维网联盟的标准， DOM 和 XSL 之类的 W3C 标准是一个整体，不失真，兼容现有图片能力前提还支持矢量(浏览器兼容情况)，通过浏览器很早版本支持情况在主流浏览器都支持，SVG 提供的功能集涵盖了嵌套转换、裁剪路径、Alpha 通道、滤镜效果等能力，它还具备了传统图片没有的矢量功能，在任何高清设备都很高清。

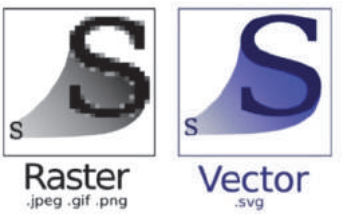


图 1 SVG 与其他格式图片比较

2. SVGKit 使用

浏览器默认就支持 SVG 渲染，属于 XML-Dom 家族系列，但是在移动端上并没有做原生支持，还是按照 XML 进行的读取，支持的开源库也不多，在 IOS 上，目前 OC 版本 SvgKit 还不错，官方 Github 也在继续维护，虽然更新较慢，通过几次 patch 提交 PR 还是很快 merge 的，一些通用属性和控件支持的不够完善，需要进行定制开发，swift 版本的 macaw 也不错，在动画效果上更加酷炫，目前也正在做 swift 效果迁移到 OC 中，渲染流程如下

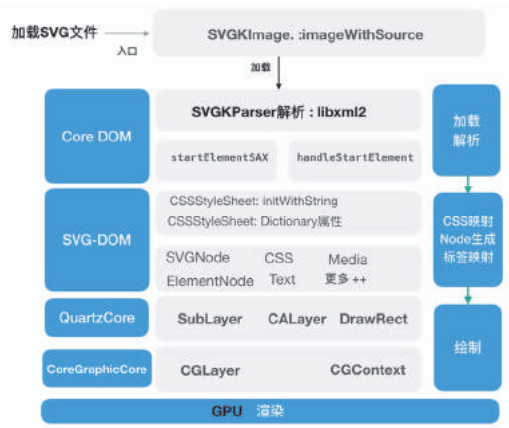


图 2 SVGKit 渲染加载流程图

1）SVGKit 有哪些标签？

```
circle = SVGCircleElement;  【圆形】
clipPath = SVGClipPathElement;【层叠路径】
description = SVGDescriptionElement; 【描述】
ellipse = SVGEllipseElement; 【椭圆】
g = SVGGElement; 【容器标签】
image = SVGImageElement;【图片】
```

```
line = SVGLineElement;【直线】
path = SVGPathElement;【路径】
polygon = SVGPolygonElement;【多角形】
polyline = SVGPolylineElement;【多边形】
rect = SVGRectElement;【矩形】
svg = SVGSVGElement;【SVG 容器标签】
switch = SVGSwitchElement;【选择】
text = SVGTextElement;【文本】
textArea = TinySVGTextAreaElement;【区域文本】
title = SVGTitleElement;【标题】
```

2）扩展基于三端统一 SVG 标签和属性

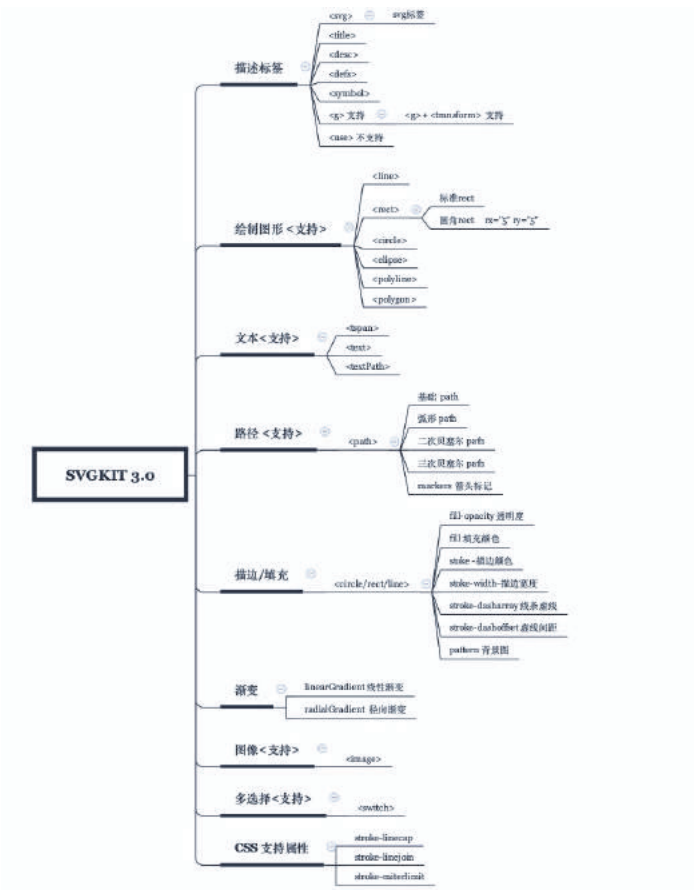


图 3　SVG 标签属性扩展大图

3）SVG 标签在 SVGKit 中渲染流程

a）SVGKit 核心渲染原理分析

视图 SVGKFastImageView.m 加载到窗口显示

核心中心处理类，主要加载 SVG 文件资源文件

```
类：SVGKimage : NSObject
SVGKParseResult* parsedSVG = [parser parseSynchronously]; // 解析
SVGKImage* finalImage = [[SVGKImage alloc] initWithParsedSVG:parsedSVG
fromSource:source];
```

b）分析：

```
解析 SVG-到合成 ViewLayer
初始化 SVGKSource source svg 资源实例
初始化 SVGSVGElement -> DomTree
初始化 SVGDocument -> DomDocument
CALayerTree 最终合成的 Layer 树
SVGKParser* parser = [SVGKParser newParserWithDefaultSVGKParserExtensions:source]; 开
始解析
```

c）解析 XML 类 SVGKParser： NSObject 解析 SVG（XML）文件

```
+(SVGKParser ) newParserWithDefaultSVGKParserExtensions:(SVGKSource )source
(SVGKParseResult*) parseSynchronously. 解析异常处理 XML 解析处理
XML 解析过程 SAX
// 每解析一个 Node 添加到 DOMTree 中. (SVGKParserStyles)
SVGKParserDefsAndUse 【解析 useAndDefs 样式】
SVGKParserDOM 【解析 DOM】
SVGKParserGradient【解析渐变标签】
SVGKParserPatternsAndGradients【解析图案】
SVGKParserStyles【解析样式】
SVGKParserSVG【解析 SVG 标签】
```

d）解析 XML 中 CSS 样式类 SVGKParserStyles :

```
标签解析到生成 Layer 层
1.类：SVGKParserDOM.m: SVGElement.
2.核心思想：
SVG 标签渲染流程一、SVGKImage.m 渲染 核心思想：遍历 DOM 映射到 iOS layer 绘制
```

```
3.生成 UILayer:
-(CALayer *)newCALayerTree
CALayer* newLayerTree = [self newLayerWithElement:self.DOMTree];
CALayer sublayer = [self newLayerWithElement:(SVGElement )child];
[newLayerTree  addlayer. Sublayer]
[element layoutLayer:layer];
[layer setNeedsDisplay];
```

4）SVGKit 分析总结

SVGKit 版本升级 2.X 升级 3.X-Release，升级后主要是一些属性的支持度更完善，包括 Text 富文本渲染，字体多样式支持，还有一些渲染上的优化，可通过 patch 提交 查看，比较一下 W3C 下 SVG 图在 2.X 分支及 3.X 分支的解析及渲染时间，性能能也有提升， 同时增加 image 图片加载 base64 图片，加载在线 URL 及本地资源图片，我们也在 SVGImageElement 中提供扩展 API 增加 Webp 支持，因为 SVG 本身是为矢量图方案加入 PNG 等图存在一定模糊情况，不过运营可能会在底图上做一些 Logo 展示，为了减少 SVG 编辑复杂度，做了一些 png\jpg 图的嵌入，一般图片都不大，以下 API：展示 base64 运营位图片而设计的

```
[NSData dataContentWithBase64Str:str]
```

3. 基于 CSS 着色能力

1）为什么用 CSS 着？

SVG 虽然是绘制图形，原理如同 HTML，是给每一个标签设置一个单独 style 好还是通过 CSS Id /class 映射好呢，这个思路和 HTML 处理 STYLE 样式是一样的，便于更改和维护，在性能上也更好，同时增加了 important 属性，可以更好的配置样式，可做到运营侧根据样式 style 下发方式达到更改 SVG 图效果，可以做到更多活动效果及个性化需求。

2）SVG-CSS 着色渲染过程

SVG 标签基于 CSS 样式快速应用，通过遍历 DomTree，找到对应的 Node 节点，在给 node 节点设置 id 或者 class，然后局部刷新 Tree 父节点，实现换色，细节流程如下

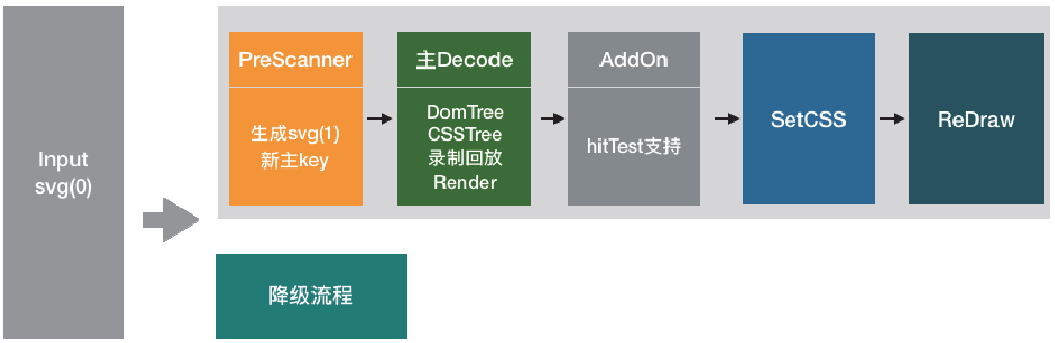


图 4　CSS 着色原理与时序图

3）大麦端选座渲染效果

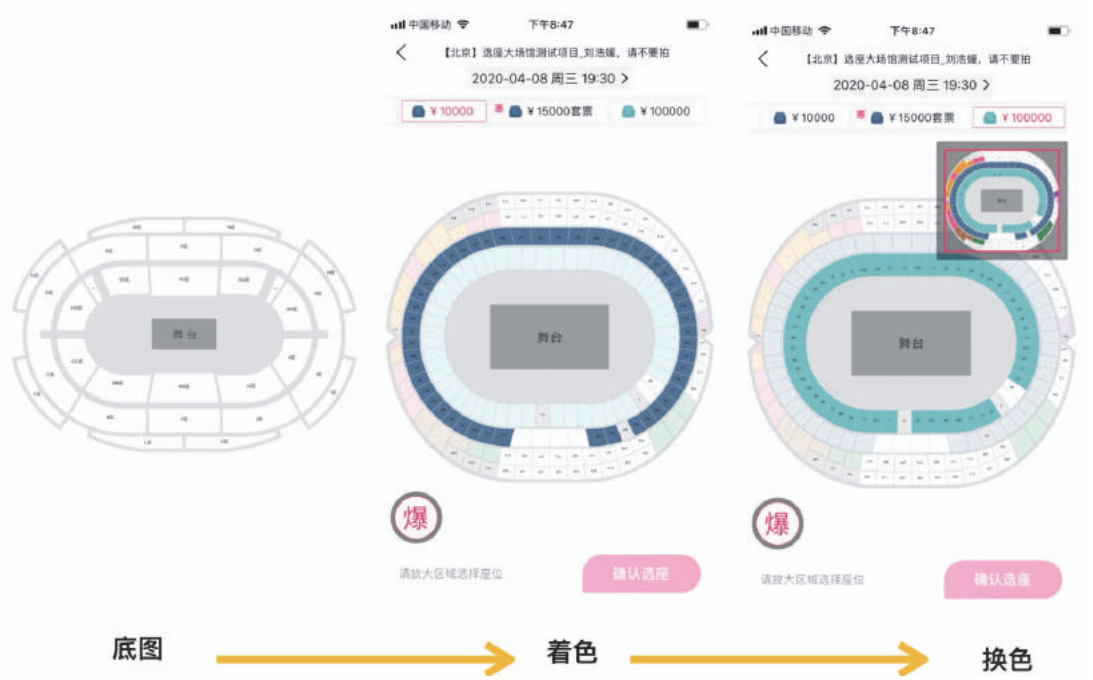


图 5　CSS 着色渲染效果

4）CSS 着色原理总结及性能比较

如何确定属性使用的是 CSS 颜色还是自带 style 属性？

当 SVG 在解析生成 DomTree 后，我们可以根据 CSSStyle 样式存储的 CSS 样式，给 Node 标签设置 id 及 class，当更改 nodeList 后，相当于树结构进行了修改，在绘制时候查找属性会根据优先策略 id > class > 进行查找进行属性赋值，我们根据 CSS 属性 !important 来设置最高优先级，这样就避免了此问题。

端侧渲染流程如下，左侧：是基于 node 遍历后修改，右侧是修改 id/class 方式【推荐】。

性能比对：为了兼容 W3C 标准，端上增加了 CSS 特殊属性 important。

文件名	SVG名字	解析	解析时间	渲染	渲染时间
文件名称	BlankMap-World6-Equirectangular .svg	解析时间	0.85 秒	渲染时间	0.00 秒
文件名称	Blank_Map-Africa .svg	解析时间	0.04 秒	渲染时间	0.00 秒
文件名称	CSS .svg	解析时间	0.02 秒	渲染时间	0.00 秒
文件名称	Coins .svg	解析时间	0.02 秒	渲染时间	0.00 秒
文件名称	CurvedDiamond .svg	解析时间	0.02 秒	渲染时间	0.00 秒
文件名称	Cycling_Lambie .svg	解析时间	0.08 秒	渲染时间	0.00 秒
文件名称	Europe*states*reduced .svg	解析时间	0.18 秒	渲染时间	0.00 秒
文件名称	ImageAspectRatio .svg	解析时间	0.02 秒	渲染时间	0.00 秒

文件名	SVG名字	解析	解析时间	渲染	渲染时间
文件名称	Blank_Map-Africasvg	解析时间	0.04 秒	渲染时间	0.00 秒
文件名称	BlankMap-World6-Equirectangularsvg	解析时间	0.91 秒	渲染时间	0.00 秒
文件名称	CSSsvg	解析时间	0.04 秒	渲染时间	0.00 秒
文件名称	Coinssvg	解析时间	0.05 秒	渲染时间	0.00 秒
文件名称	CurvedDiamondsvg	解析时间	0.01 秒	渲染时间	0.00 秒
文件名称	Cycling_Lambiesvg	解析时间	0.08 秒	渲染时间	0.00 秒
文件名称	Europe*states*reducedsvg	解析时间	0.18 秒	渲染时间	0.00 秒
文件名称	ImageAspectRatiosvg	解析时间	0.01 秒	渲染时间	0.00 秒

图 6　SVG-Codec 总体性能提升对比

4. SVG 约束 DTD

1）背景介绍

当 SVG 生产端在制作 SVG 图，可能会用到 Adobe 等软件，有很多复杂属性及层叠，可能会产生复杂 XML 格式，这样在渲染过程中会造成大量遍历，影响性能，也有一些特殊属性，端上并没有支持，例如滤镜、动画，这样，我们就需要有一种约束来校验生产和渲染 SVG 能够一致。

2）文档类型定义

（DTD）可定义合法的 XML 文档构建模块。它使用一系列合法的元素来定义文档的结构。DTD 可被成行地声明于 XML 文档中，也可作为一个外部引用。

DTD 被定义在 xml 的 DOCTYPE 声明中。

3）定义一个名为 note 的 DTD

如果要使用内部定义，则在 xml 文件的 xml 版本声明头下面添加如下代码块：

<!DOCTYPE svg [// DTD 内容]> 如果要引用外部 DTD，那么它应通过下面的语法被封装在一个 DOCTYPE 定义中：<!DOCTYPE root-element SYSTEM "filename"> 例如，在 xml 文件的 xml 版本声明头下面添加如下代码块：

<!DOCTYPE note SYSTEM "note.dtd"> 一个 DTD 的内容示例：

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

其中：

!ELEMENT note 定义 note 元素有四个元素："to、from、heading、body"

!ELEMENT to 定义 to 元素为 "#PCDATA" 类型。

PCDATA 的意思是被解析的字符数据（parsed character data）。可想象为 XML 元素的开始标签与结束标签之间的文本，PCDATA 是会被解析器解析的文本。这些文本将被解析器检查以及标记，文本中的标签会被当作标记来处理，而实体会被展开，不过，被解析的字符数据不应当包含任何 &、< 或者 > 字符；需要使用 &、< 以及 > 实体来分别替换它们。

4）例如在 DTD 中声明

<!ELEMENT name (#PCDATA)> 它表示在<name>和</name>之间可以插入字符或者子标签，CDATA 的意思是字符数据（character data, CDATA 是不会被解析器解析的文本。在这些文本中的标签不会被当作标记来对待，其中的实体也不会被展开。

5）如何校验

在完成 DTD 文件的编写后，就是使用 DTD 了,1、 一般使用代码解析的方式，进行 DTD 对 xml 的规范性校验，首先在 svg 的头部加入：<!DOCTYPE svg SYSTEM "svg_note.dtd">然后在解析 svg 时，声明合法性校验例如，以 SAX 解析 XML 为例：

```
SAXParserFactory spf = SAXParserFactory.newInstance();
spf.setValidating(true);  // 关键设置
SAXParser sp = spf.newSAXParser();
XMLReader xr = sp.getXMLReader();
XMLParser.SAXHandler handler = new XMLParser.SAXHandler();
xr.setContentHandler(handler);
xr.setErrorHandler(new SAXErrorHandler()); // 输出校验出错的信息
xr.setProperty("http://xml.org/sax/properties/lexical-handler", handler);
xr.parse(new InputSource(is));
```

其中，必须设置 setValidating(true);才能使 DTD 校验 xml 生效，为方面使用，提供了可以动态读取 dtd 的方式，为不需要将 dtd 信息添加到 svg 的文件中：执行 Java -jar 命令，传入两个参数：一个是 svg 的全路径；一个在同目录下的 dtd 的文件名（带扩展名）。

5. 选座性能优化

1）性能调研：APP 端/H5 上渲染如何解决 10 万座位渲染，端侧通过组件复用，手机设备性能天然还是不错的，加上我们通过预加载资源与分区加载结合方式，点击区域后进行绘制策略，避免一次性加载全量 10 万数据，也给用户更好的交互体验，然而 H5 侧，浏览器就不那么流畅了，随着 H5 技术发展，H5 新特性的支持，通过实践使用 SVG 方案，每个座位都用一个 svg 元素显示，由于 svg 的矢量特性，缩放无锯齿，展现效果比较好，但也就支持到 3 万左右的座位，座位再多也会出现渲染慢和缩放卡顿等问题。Svg 的每个元素也算是一个浏览器的 dom，dom 数量一多起来，达到 3 万到 10 万，浏览器渲染显然不行。如何减少 dom 节点，又能显示这么多内容呢？很容易就能想到用 Canvas 来绘制座位，无论在 Canvas 绘制多少元素，在浏览器都是一个元素，这样就很好解决了 dom 数量的问题，网上能搜到多个应用比较多的开源 Canvas 组件，能实现绘制元素的实时更新，还支持元素的鼠标/手势时间响应。但是都有一个问题，基本上的原理都是采用帧刷新重绘 Canvas 画布的方式来实现视图更新，确定采用 Canvas +svg 渲染座位图，模拟了 10 万座位的渲染并实现了缩放拖拽等操作，达到了预期的效果。

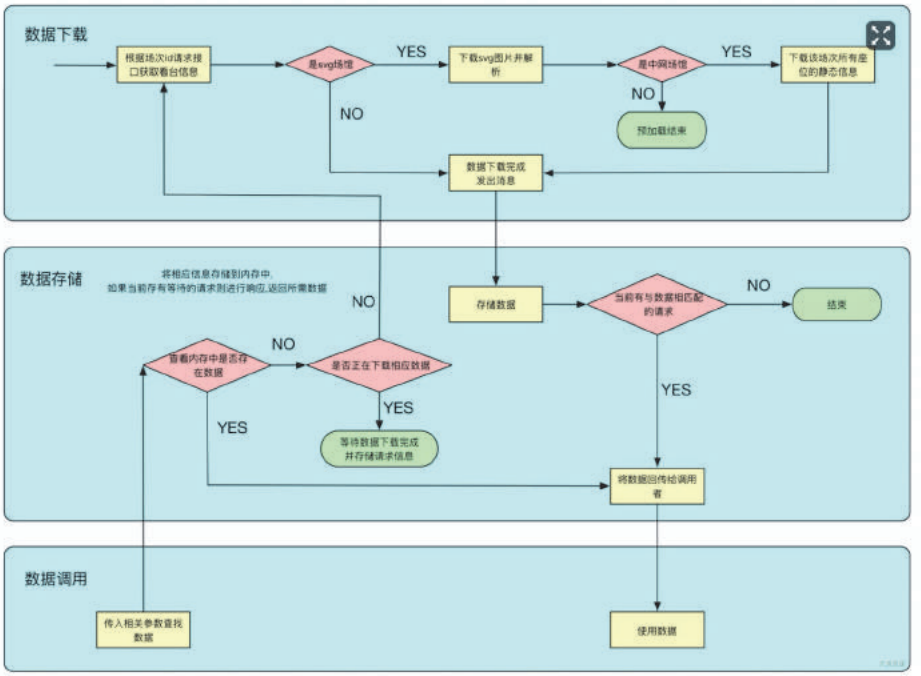


图 7 选座性能优化-预加载

图 8 端选座性能交互图

5. SVG 场馆彩虹图实现

1）SVG 彩虹图介绍：

在售票选座业务中，需要为用户显示场馆图（SVG 格式），给用户一个场馆的整体印象，同时方便用户选择场馆的看台，进而展示看台座位进行选座。但是，在使用彩虹图展示场馆图之前，场馆图的看台区域仅展示看台当前可售座位中的最高票价对应的颜色进行展示，如下图所示：

图 9 非彩虹场馆图

每个看台都是单一颜色的，不能反映出当前看台中可售的座位的价格分布情况，很容易迷惑用户，造成每个看台只有一种票价的印象，同时不方便用户快速定位他的目标价位所在的区域。为了准确的反映出场馆每个看台的价格情况，需要将看台中所售的所有的座位的价格展示出来，因此，采用彩虹图的形式。目标效果如下所示：

图 10 彩虹场馆图

每个区域的所有的座位价格以彩虹的形式显示出来，相比以前的只显示最高票价的颜色，彩虹图可以清晰的展示每个区域中座位的价格情况。

2）总体思路：

在每个看台区域中以彩虹图形式展示多个颜色，就需要将每个看台区域进行划分，放弃之前用一个这类的绘制标签来展示一个看台区域，一个区域内应该包含多个排，对每个排按照座位价格进行着色，进而对每个看台进行同样的处理，总体上形成彩虹样式展示。

因此，一个看台区域，应该是多个 svg 标签组合而成的。如图：

图 11　SVG 文件说明

对 SVG 底图进行改造，将老的一个 SVG 标签代表一个看台区域的形式，改造成每个看台区域由标签进行包裹的多个标签的组合。

为了方便降级，处理不显示彩虹图的业务需求，同时约定标签下的第一个标签，表示整个区域，同时不再解析渲染后面的排信息。

3）算法生成彩虹图方案

a）介绍：

彩虹图生成算法主要是通过座位的分布和座位的票档圈出一个看台中相同票档座位的范围，然后生成一个 path 路径。将这些 path 路径加入到 svg 底图中去并且和相应的票档绑定，就能实现一个区域多种颜色的效果。

b）步骤如下：

①对看台中所有排和座位进行分组排序；

②计算看台方向；

③获取同种颜色座位边界；

④计算色块的方向；

⑤获取色块的路径；

⑥生成看台的彩虹图效果；

⑦遍历所有看台生成完整的彩虹图。

首先将某个看台所有座位按排分组，然后将排按排号从小到大进行排序。数据结构如下：

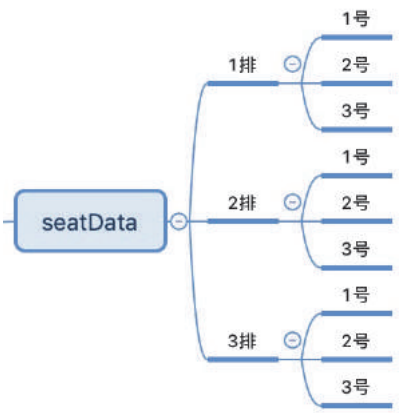


图 12 看台编号与座位号示例

座位数据是必备的基础数据，后续一切的计算都依赖于座位数据。

通过第一排和第二排座位的相对位置算出看台的方向。

比如:拿到 1 排 1 号和 2 排 1 号座位的坐标，从 2 排 1 号向 1 排 1 号画一条射线，这个射线的角度就当做看台的方向。

后期如果在生产 SVG 的时候将舞台位置标记出来的话就可以利用舞台来确定看台方向。

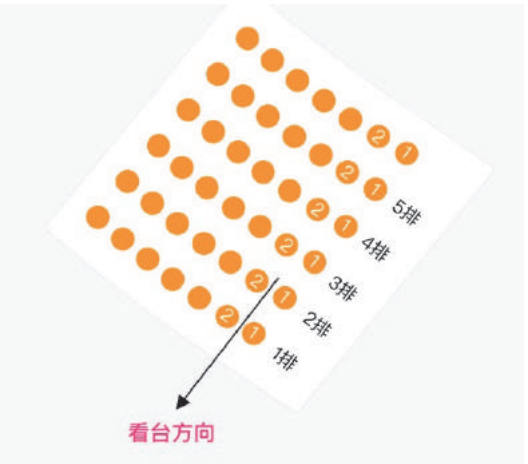


图 13 看台方向-1

每个看台的座位分布可以分成两种，一排一种价格和一排多种价格。

其中一排一种价格的情况就以同色最后一排为边界。如下图绿色的线。

一排多种价格的情况就需要把每一排不同颜色转换处的那两个点记录下来连成边界线。如下图红色和黄色的线。

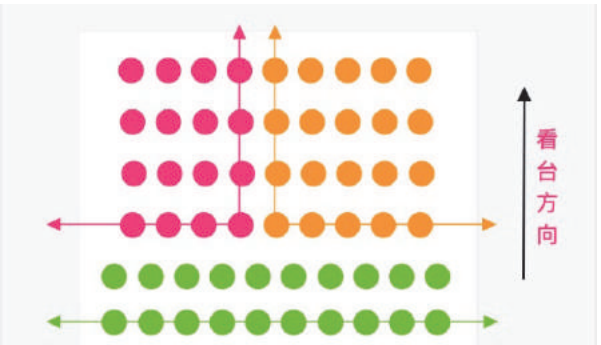


图 14 看台方向-2

红黄绿三条线所在的座位就是我们需要的色块边界。

拿到色块边界座位数组之后还需要知道色块的角度。一排同色的色块方向就直接使用看台的方向。一排多色的色块方向计算方法如下：

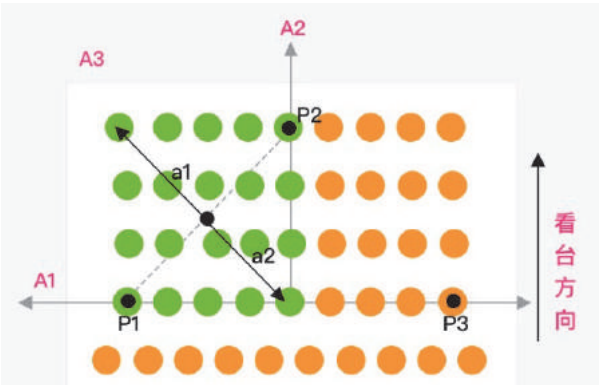


图 15 确定颜色区域-1

将第一个座位 P1 和最后一个座位 P2 连线，取这个线段的垂直线 a1 和 a2。用这两条垂直线分别和看台方向取夹角。夹角小于 90 度的垂直线的角度作为色块的方向。图中 a1 就是色块的方向。得到色块边界和色块方向后就可以计算色块 path 的路径了。

①得到包含这个看台 path 路径的最小矩形 rect (图 1)；

②从上一步获取的座位边界数组中取四个点分别为:第一个点 P1，第二个点 P2，倒数第二个点 P3，最后一个点 P4 (图 2)；

③由 P2 向 P1 方向做一条射线得到和 rect 的交点 A1，由 P3 向 P4 方向做一条射线得到和 rect 的交点 A2 (图 2)；

④根据色块方向获取色块路径的几个关键点:A1 A2 A3 (图 2)；

⑤将几个关键点和座位边界所有点连接起来生成一个闭合的路径(图 3 橙色线框)。

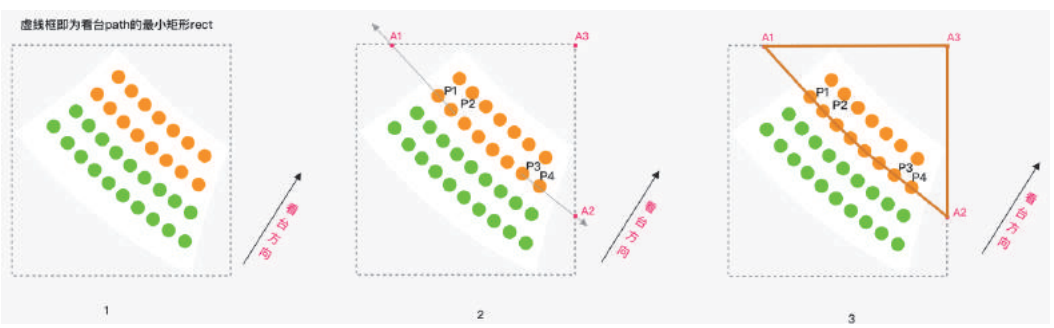


图 16 确定颜色区域-2

拿到所有色块路径后就可以将色块填充对应的颜色并且按顺序叠加到看台上形成彩虹图效果。

色块的叠加方式如下:色块的生成顺序是 path1->path2->path3->path4，然后倒序叠加到看台上 path4->path3->path2->path1。最后就是遍历所有的看台，生成一张完整的彩虹图。

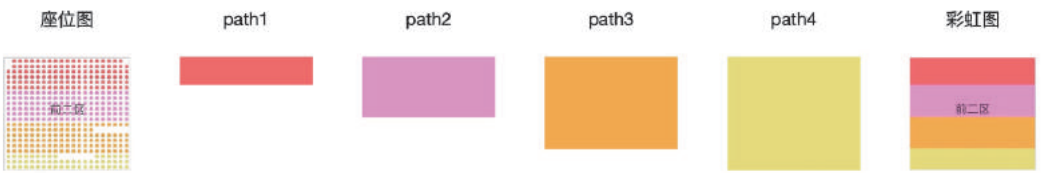


图 17 确定颜色区域-3

四、总结

本文主要讲解了大麦核心链路选座 SVG 应用，并结合实际场景做了一些创新尝试，包括：丰富 SVG 应用的业务场景、SVG 标签属性及扩展、CSS 着色、渲染性能优化等，目的是让端解析接近浏览器解析效果，并提供更好的端选座性能体验。

首次揭秘！看大麦如何掌控超大规模高性能选座抢票

作者| 阿里文娱技术专家 恒磊、阿里文娱高级开发工程师 新钱

一、背景介绍

随着现场娱乐行业的不断发展，各类演出层出不穷，越来越多的演出开启选座购票满足用户的自主选座需求。大麦的选座不仅面向中小场馆类的剧院演出，还面向大型体育赛事、大型演唱会等超大型场馆（如鸟巢近 10 万座）。选座类型抢票的特点是“选”，由于“选”的可视化以及超大场馆在数据量上对大麦是很大的挑战。本文通过服务端和前端上的一些解决方案来探讨如何支撑超大规模场馆的高性能选座，通过本文的一些技术方案希望可以对读者在一些高并发实践中提供帮助。

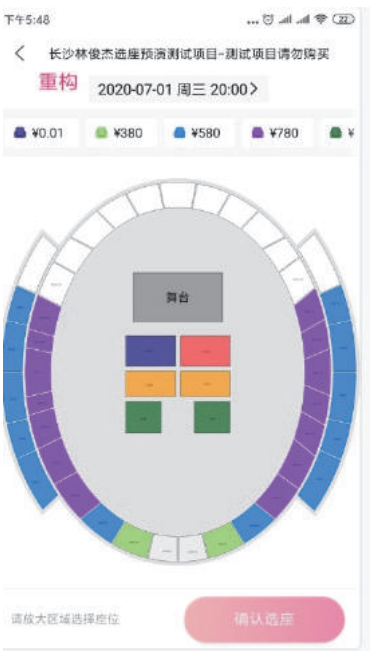
二、核心问题

首先看一下普通商品的秒杀，拿某款手机秒杀来说，一般情况下，一场手机只有几个型号，比如中低档、高性能档、旗舰等，处理好这几个商品的库存即可。对于选座类抢票而言，每一个场次的所有的每一个座位都可以认为是一个不同的商品，场馆大小不一，大的鸟巢有 10w 座位。也即是说一个选座类抢票就涉及 10 万级别的商品，如果多个项目同时开抢，整体数量会很庞大。目前大麦电商侧的商品维度是票档，座位并不是商品粒度，所以无法利用集团的秒杀能力，选座类抢票涉及电商、选座、票务云产销，是对大麦整体能力的考验。

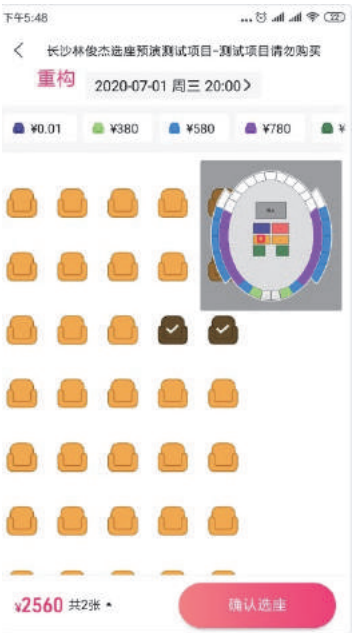
先来看看整个选座购票的流程：以林俊杰长沙测试项目购票为例。

①用户打开需要的场次项目详情页

②点击选座购买，打开选座页面，查看座位图及票档



③选择一个看台区域，选择喜欢的座位，点击确认选座



④进入下单页面，填写手机号收货地址，创建订单

⑤提交订单完成付款、出票。

其中，2、3、4 环节都与选座相关。

从流程上看，选座的核心关键技术在于：

①座位图的快速加载。快速加载其实就是选座页面的读能力。选座页面需要下载座位底图、座位基础信息（排号等等）来做渲染，同时需要票档、该场次每个座位的状态，来决定是可售还是锁定还是已经售出。

对于大型场馆座位 5 万-10 万，渲染一个选座页面需要的数据量都很大。

②高并发。由于热门演出票的稀缺性，同时抢票的人数可能达到几十万。如何支撑如此高的并发和吞吐是一大考验。

③座位状态更新的及时性

当某个座位售出后，需要及时更新座位状态。

④抢票体验：抢票时热门的看台某个座位可能几十个人并发去抢，如何尽量提升用户的体验，尽量让更多用户一次性购买成功，体验更好。

三、高性能选座实践

针对高性能选座的核心要求，我们从如下几个维度去阐述我们在选座类抢票上的实践。

1. 动静结合

选座的瓶颈数据量“首当其冲”。从逻辑上讲，一个座位完整的展现到用户面前，需要包含座位的看台、排号、价格、售卖状态等信息，其中 看台、排号等等是不变的，并可提前预知的；售卖状态等时根据项目的进行会动态变化的。所以把选座的数据拆分为动态、静态数据。对于大型场馆如 10 万场馆，用户打开一个选座页，座位的静态数据（座位 id，票价 id，是否固定套票，坐标 x，y，和舞台角度，哪个看台，几排几号等等），这些数据大概 15M 左右。再加上动态数据如票档状态、颜色、看台状态、座位状态，10w 场馆大概 2M 左右。也就是说如果不做处理用户仅仅打开选座页就需要 17M 以上的数据量。如果选座数据存储在 oss 上，如果每人下载 15M，10w 人同时抢票则需要 1500G 带宽，带宽成本很高。为了解决静态文件访问速度问题，将静态数据从 oss 直接接入到 cdn。同时为了保障数据最新，静态数据采用版本控制。

2. 静态数据的预加载

上面提到带宽峰值很高，为了降低峰值且提升体验，客户端引入了静态数据预加载。静态信息结合预加载的处理，为处理大数据量的座位信息提供了时间上的余量，用户在打开选座页时优先显示静态信息，可有效降低用户等待时间，提升用户的体验。

通过大数据分析结合用户的行为习惯，确定预加载的场次类型，提高预加载的命中率。

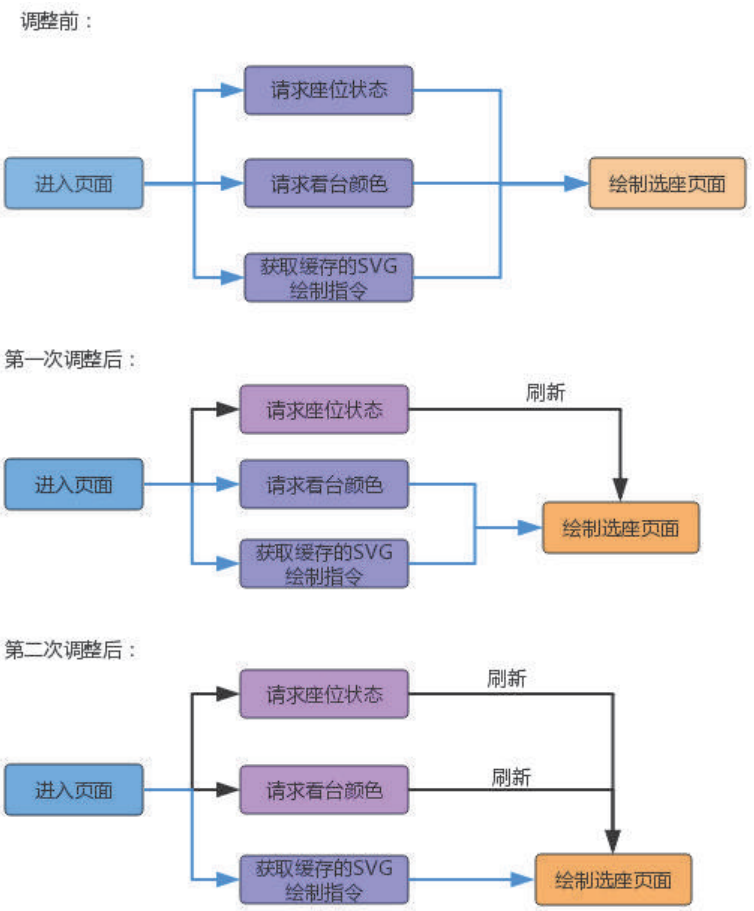


图 2.1：预加载调整流程

预加载+预解析，完成了绘制基本场馆信息的数据准备，再将数据提前与画座 View 绑定，预渲染 View 进而达到选座页秒开效果。

3. 座位数据编码

动态、静态数据量大制约了我们实现高吞吐，同时也浪费了服务带宽和用户流量。所以需要压缩，压缩到一定的可接受的范围。只有数据量足够小，才有办法做到高吞吐。

1）静态数据编码

在处理大数据量的座位（例如十万级）仅有静态数据的预加载往往是不足的，预加载并没有从根本上处理座位数据量大的问题，同时对于类似体育比赛这种多日期多场次的场景，由于预加载的使用存在缓存量的控制，往往影响预加载的命中率。

而数据重编码和数据压缩的使用，是从源头解决问题的有效思路。

座位数据的重编，舍弃传统的 xml 和 json 格式，不仅可以有效压缩数据大小，还对数据安全提供了保障，即使被拿到了接口数据，因为缺乏数据编码的协议，也无法获取有效的原始信息。

2）座位静态数据压缩整体框架

目前大麦针对自己特有的座位数据特点，结合高效二进制编码方案进行座位数据的重新编码，再使用通用的无损压缩进一步缩小数据体积，从而减少了座位数据的网络传输时间，从根本上解决大数据传输导致的时延问题：

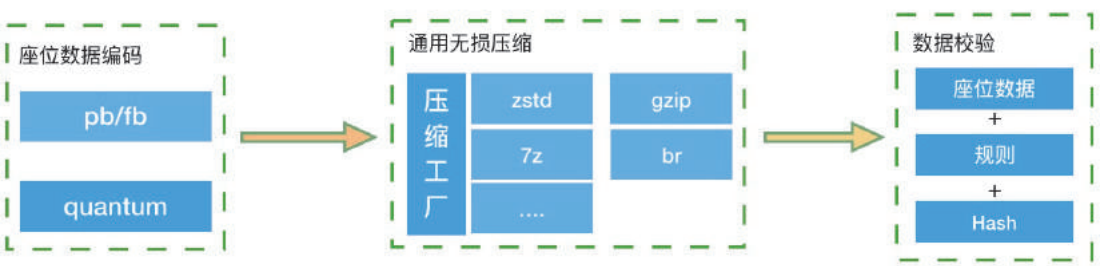


图 3.1：静态数据压缩流程

基于二进制的数据编码，既保障了数据安全性，又保证了在数据解析中的高效性，即使数据压缩的使用也比 json、xml 的解析具有更少的时延。

同时兼具工具化的批量生产方式，又进一步解决了数据构建问题。

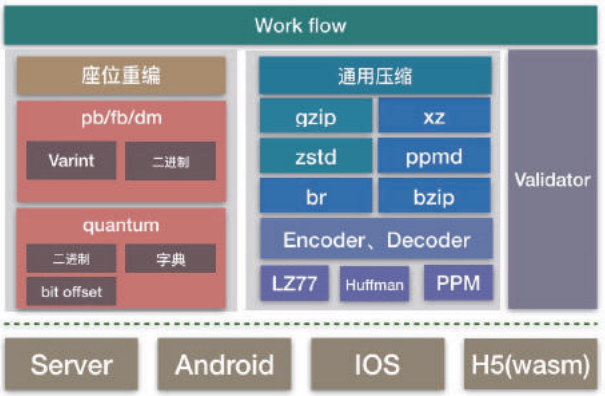


图 3.2：静态数据压缩工具架构

通过端上和 server 端的握手协商过程，实现了数据编码和通用压缩方式灵活组合，sever 端可以针对不同的端下发相应的压缩格式文件：

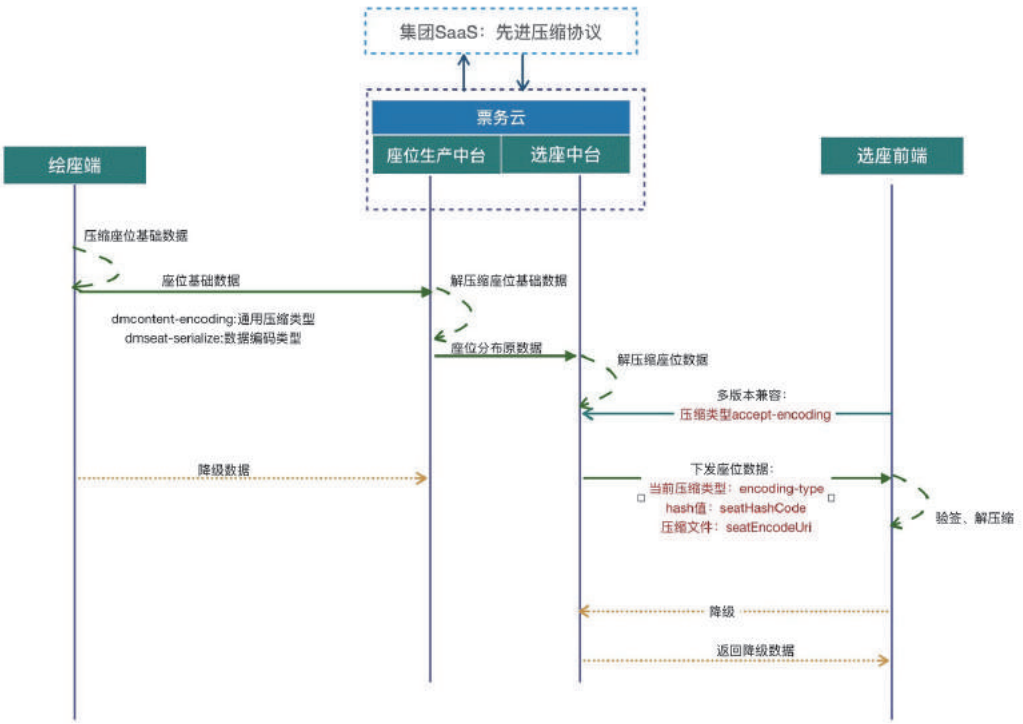


图 3.3：静态数据压缩，握手协商工作流

结合 CRC 的思想，制定了兼容 IOS、Android 和 H5 三端的基于座位属性的纵向 hash 校验。相比 md5 等普通的散列计算方式，在处理 6 万级座位多维度信息时，在端上实现了十几毫秒、H5 侧 50 毫秒左右的全量数据检测，使得在不占用多长时间的情况，可以验证数万级座位解析的准确性。

经过这编码到检测的完整链路，实现了减少数据的体积的同时，又能达到比传统 xml 和 json 数据的高效解析、高安全性。

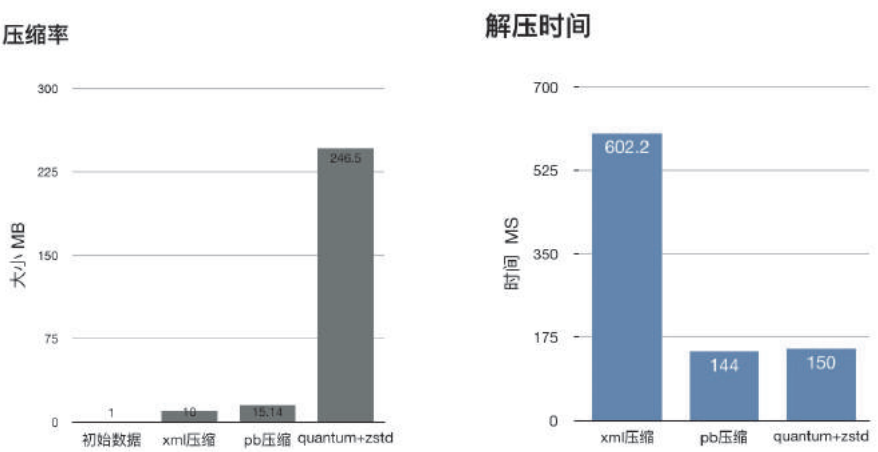


图 3.3: 压缩数据对比

其中 quantum 是大麦端上自研的基于动态比特和字典的压缩算法，结合大麦特有业务场景，实现了高压缩比和快速数据解析:

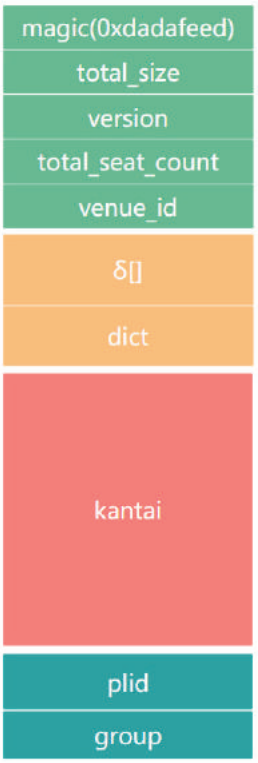


图 3.4: quantum 压缩文件结构

3）座位动态数据的编码处理

a）动态数据的难点

选座动态接口主要涉及票档情况、看台情况、座位状态。数据量最大的是座位状态。以一个 5 万座位的场馆为例，每个座位都要返回一个状态供前端渲染。采用座位 id 和状态键值对的方式，由于座位较多使得整个返回结果过大，5 万座位的场馆返回 1M 以上的数据。如果打开一个选座页需要吞吐 1M 数据量的化，接口基本不可用了。之前的策略是按照分组策略，5 万大概会分 10 个组，这样每个请求大概 100k 数据量，这样才能达到接口基本可用，不过端上需要请求 10 次才能拿到整个场次的状态，可想而知性能会有多大影响。假如 5 万座位的场馆，10 万峰值抢票，那么仅仅这个接口就需要 100 万的 qps，所以肯定会逢抢必挂。

b）方案

目前接口是通过 mtop 协议，我们思考的前提：目前 mtop 不支持 byte[]数组流，只支持 json

等格式的字符串结构。如何把返回的数据减小，采用一个尽量简洁的方案，同时调用一次返回整个场馆座位状态，是我们努力的方向。

数据量大主要是因为有很多冗余的座位 id。有没有办法不依赖这些座位 id？既然我们做的动静分离，静态数据里已经涵盖了座位 id，我们动态接口里只需要对应的返回状态即可，即按照静态里面的顺序返回座位状态。同时我们把返回结果进行简单的相邻状态合并将进一步降低返回结果大小。假设用户选座座位符合正态分布概率，平均长度 5w 座位平均返回不到 20k。

当然如果我们 mtop 协议支持二进制流，那么我们可以用 bit 位进行存储，可以进一步降低返回结果的大小。

4. 高效缓存

1）缓存

面向这么高的 TPS，tair 是不二首选。采用 tair + 本地缓存，来支撑如此高的数据峰值。

提到 tair，提一下我们这边的一些策略。

用户进到选座页是一个个的看台，我们设计了一级 stand cache，即看台级别的 cache；用户会进行选座位，我们又加了一级 seat cache，即为座位粒度的 cache。两级 cache 保障用户查看台和用户下单选座都能命中缓存。standcache 是热点 key，从选座的场景是允许数据非准实时的，所以引入了 tair localcache 和 guava localcache 来增加吞吐，以此降低对 tair 的读压力。

2）保护下游系统

目前采用的策略是 对下游的调用采用加锁，tair 全局锁。拿到锁的才去请求票务云底层数据。拿到锁的进程去更新 tair 缓存。其实从这里看对 tair 的写还是 qps 比较小的，但是每次争抢锁对 tair 的读还是不算太小。通过采用本地的锁 + 随机透传来减少 tair 锁耗费的读 qps。为了对下游依赖做降级，增加了数据快照，每次对下游的调用记录数据快照。当某次调用失败采用之前的快照进行补偿。

3）保障数据更新及时

由于我们采用了 tair 全局锁，可以按照秒级控制下游调用。调用采用异步触发。最短 1s 内会触发我们发起对下游的调用。如果我们想最大化利用票务云库存能力，给用户的延迟在 1s 以内，我们有一些策略。拿到锁的线程 1s 内调用数据更新任务，在数据更新任务里做一些策略，1s 内是发起 1 次还是多次对票务云的调用，调用越多 tair 更新越及时。由于用户有一定的选座

动作，一般情况下 500ms 的延迟用户基本无任何感知的。

4）缓存预热

预热一下缓存，对用户体验和系统性能很有帮助。抢票类项目采用一定的策略做自动化预热。

5. 安全及容灾

1）安全

从上文看大麦座位数据做了编码和加密，同时存储路径做了混淆，保障不到开抢时间座位数据无法被破解，保障了选座数据的安全性。此外选座页布局防控策略，保障是真正需要点击座位才能完成下单，防止机刷、防止绕过选座直接下单。通过类似策略降低了选座的无效流量，提高了稳定性。

2）容灾

选座主要在以下几个方面做了容灾。静态数据存储在 oss 上，目前采用跨地区存储容灾；tair 缓存采用主备缓存，出故障时可以做切换；服务端在 pc 和无线做了集群隔离。

四、总结

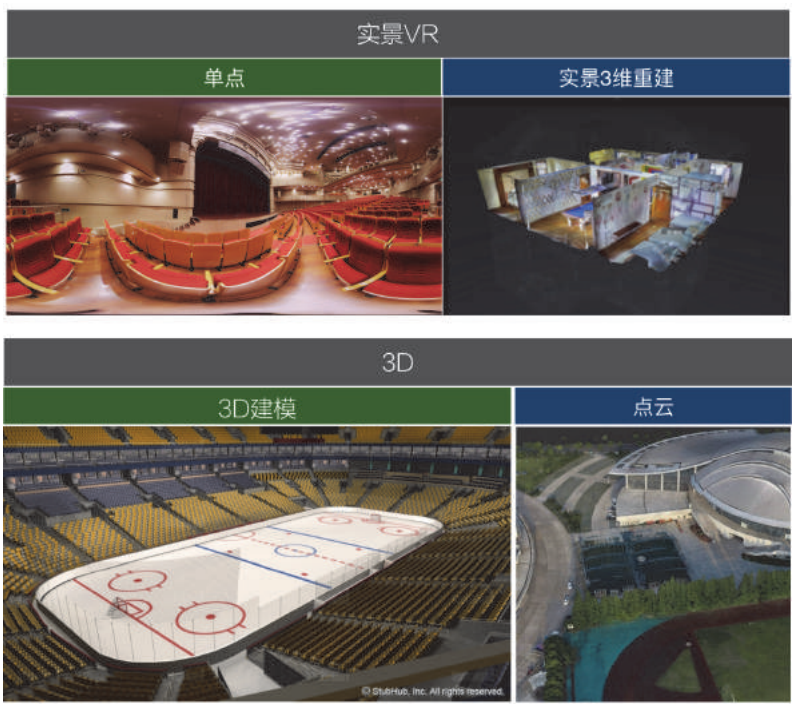
本文通过在数据处理、选座性能、缓存等等策略上来阐述了笔者在大麦高性能选座上的一些实践。通过这些实践目前大麦可以轻松的承载数十万人的顶级流量的抢票项目。

3D/VR 选座技术探索

作者| 阿里文娱无线开发专家 王璟瑶

一、行业现状

实景 VR 目前的行业应用案例逐渐增多，在使用 720° 全景相机拍摄，部分厂商基于多实景照片进行多叉数建模，在链家等房产行业获得广泛应。在票务行业，场馆选座的国内外的同类产品中也有试点落地，国外的有 TicketMaster、Stubhub 等，国内尝试落地的有摩天轮，针对大型场馆，目前的实现思路偏向于使用 3D 建模+后渲染输出基于 ECB 的全景照片，下发后用于大前端多端展示。



二、大麦解法

大麦落地全景 VR 主要是为用户的选座决策提供辅助和沉浸式体验，目前综合考虑选座页面用户交互和落地成本，采用大场馆建模+全景渲染和小场馆实景 VR 拍摄节奏。

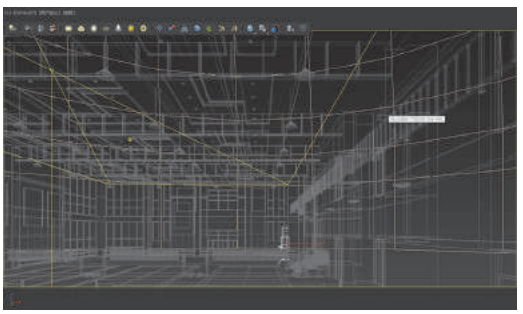
1. 落地策略

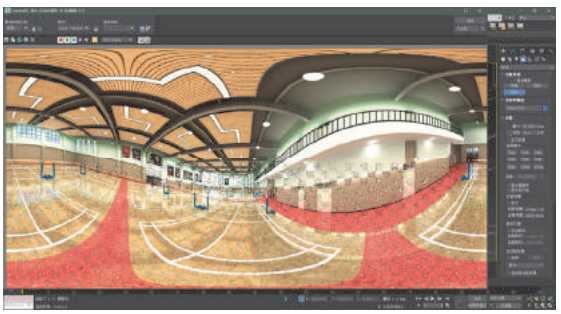
大场馆-3D 建模+渲染

小场馆-实景 VR

2. 建模全链路流程

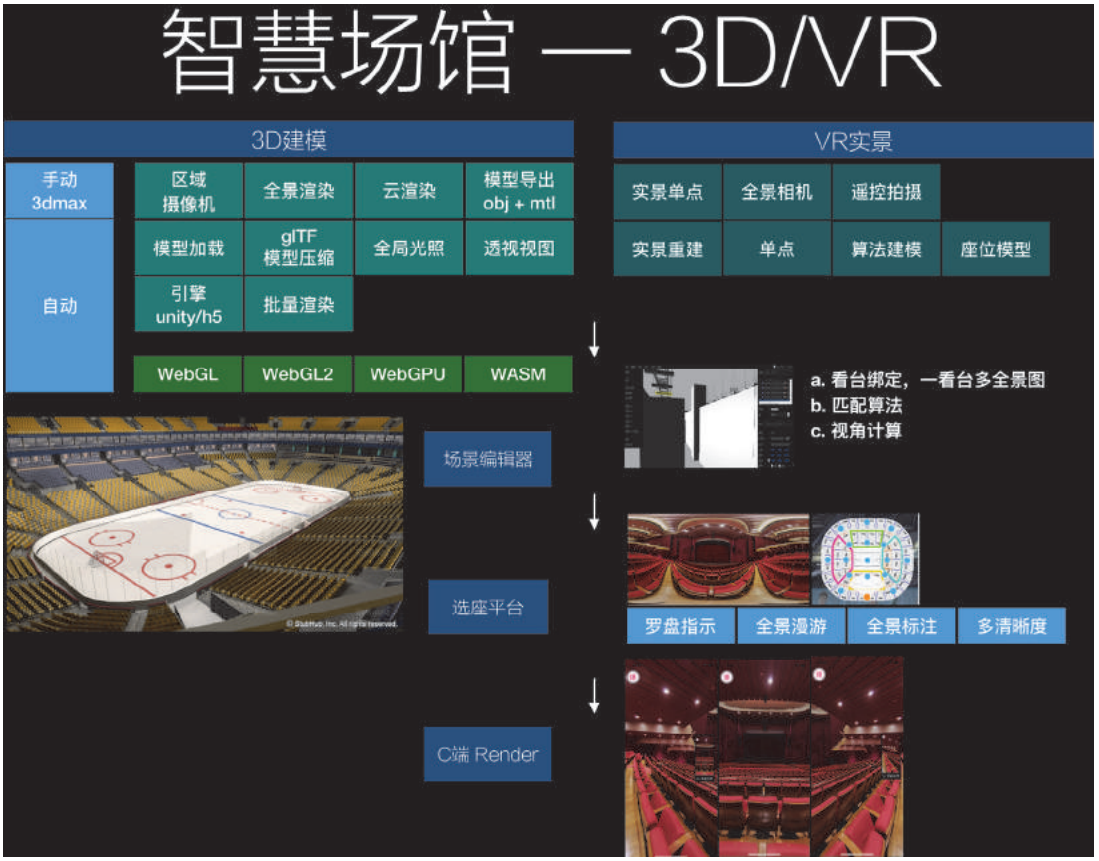
场馆建模经过输出白模、材质纹理贴图等流程进行输出建设，基于经典 3dmax 进行近似建模。





3. 全链路流程

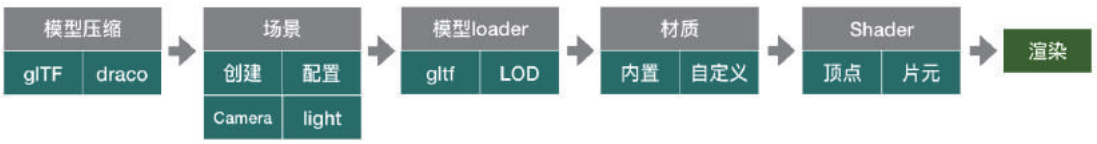
整体流程按大节点分成：建模/拍摄→ 生产→ 选座基础平台→ 大前端主要环节，建模部分目前使用 3dmax 的常规方案，并调研基于 3D 激光雷达 + SLAM 的量产方案。



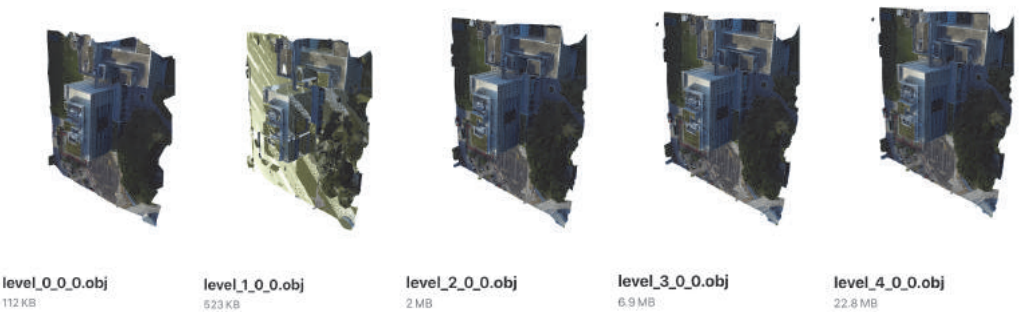
三、3D 场馆

1. 模型纹理加载

针对建模输出的场馆模型，3dmax 导出后容量在 50M~200M 之间，采用 glTF 模型压缩，再配合 LOD 多层次细节纹理，可有效提升基于 three.js 的超大纹理场馆的打开速度。



LOD 效果图展示



四、VR

1. VR Engine

VR 整体方案选择使用基于 ECB 球体坐标的投影方案，渲染合成 2:1 的全景图片，图片本身经过 moz-jpeg 压缩、智能降噪和超分重建，供 VR Engine 渲染，也为为全景图片展示秒开打下基础。大前端 VR Engine 层面，APP 侧 android/ios 较为成熟，选用 google/apple 自带方案，h5 侧 engine 性能和集成度差异较大，调研了目前市面上常见的几款 engine，最终选择使用 Pannellum 作为首选引擎。支持罗盘指示、全景漫游、全景标注、多清晰度等扩展功能。

框架	功能	模型	上手	锯齿	缩放	移动端
Threejs	强	多	中	有	支持	支持
Aframe	强	多	易	有	支持	支持
Pannellum	强	中	易	需优化	支持	支持
WebGL	强	无	难	无（自定义）	支持	支持

2. VR 视图及优化

透视视图、鱼眼视图、立体视图、建筑视图、潘尼尼观、小行星等。GVR 进行了优化封装，包括但不限于：

a. 镜头失真校正（Lens distortion correction）

b. 空间音频（Spatial audio）

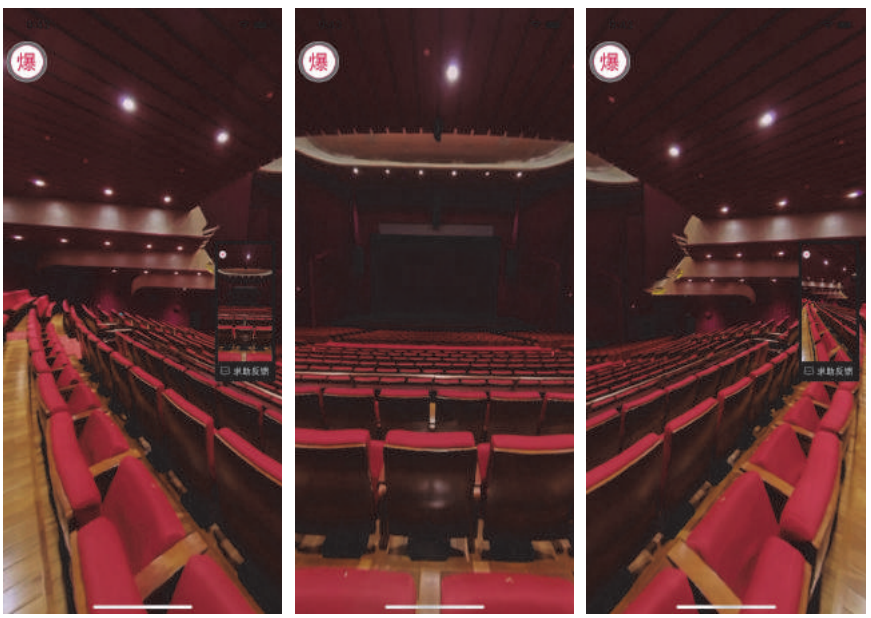
c. 头部跟踪（Head tracking）

d. 3D 校准（3D calibration）

e. 并排渲染（Side-by-side rendering）

f. 立体几何配置（Stereo geometry configuration）

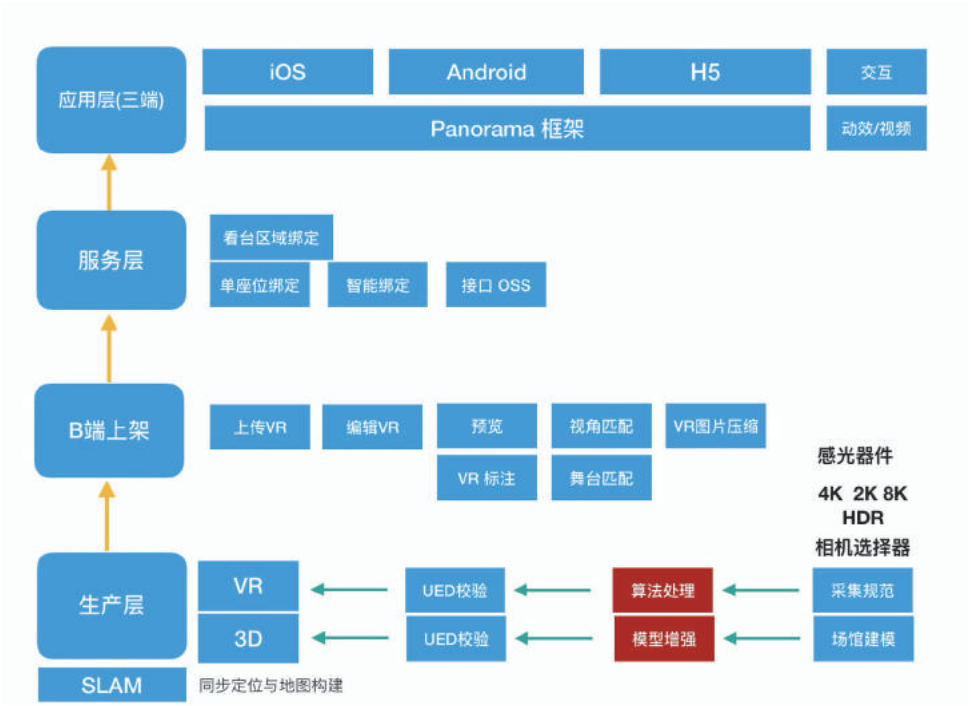
g. 用户输入事件处理（User input event handling）



五、全链路量产

全面落地，需要建模和拍摄的场馆较多，针对 VR 落地应用，需要进行量产操作，按照梳理，核心步骤大致分为以下几步：

- 拍摄：专业 camera + 大麦 B 端 = 遥控拍照 + 携带座位信息
- 生产：绑定、fov 倾角计算、压缩、超分重建
- 选座基础平台: 底图保护-暗水印、流式加密
- 大前端：三端 VR Engine、渐进式加载、流式加密



六、总结

针对上线的图片，进行了合成优化以及基于 Lanczos/hpx 的图像压缩，以及流式密码加密，以保障用户的秒开和数据安全。3D/VR 的持续建设是智慧场馆重要的一环，也为用户购买决策提供了可视化手段，量产阶段后继续探索基于大场景点云渲染建模和商业化营销能力，以期为用户带来更好的沉浸式体验。

大麦库存的高性能及一致性是如何设计的？

作者| 阿里文娱技术专家 古行

一、背景

大麦网作为现场娱乐票务平台，其业务覆盖了各大顶级演唱会和大型赛事等高流量项目。票务行业库存系统不同于普通电商库存系统，瞬压过高的秒杀抢票，多场景、多阶段的售卖，对一致性和稳定性提出更高要求。本文将为读者介绍现场娱乐行业票务库存高性能和一致性难点解决和沉淀下来的库存稳定性建设经验。

二、库存场景分类

在售卖中支持多渠道：主办方售卖、大麦网售卖、其他分销渠道售卖等。来满足不同场景下的票源管理。大麦网库存可以分为无座库存、选座库存、预售数字库存和现票库存。

1）无座库存：比如音乐节、游展类项目，观众持票入场，无需安排座位；

2）选座购买：用户可以自己通过座位图选择座位，下单购买；

（图：大麦网选座购买）

3）预售数字库存：预售阶段用户先按照票档价位购买，然后系统按购买顺序进行自动配座；

（图：大麦网预售数字库存购买）

4）现票库存：如非大麦网票务系统生产的，由大麦网从主办方提取现票在大麦网售卖。

1. 库存核心技术点

目标：高性能、强一致性

库存的高性能和一致性主要从以下几个方面保障：

1）进行事务拆分，执行粒度最小化，优化提高性能；

2）基于正向流水防止重复扣减；

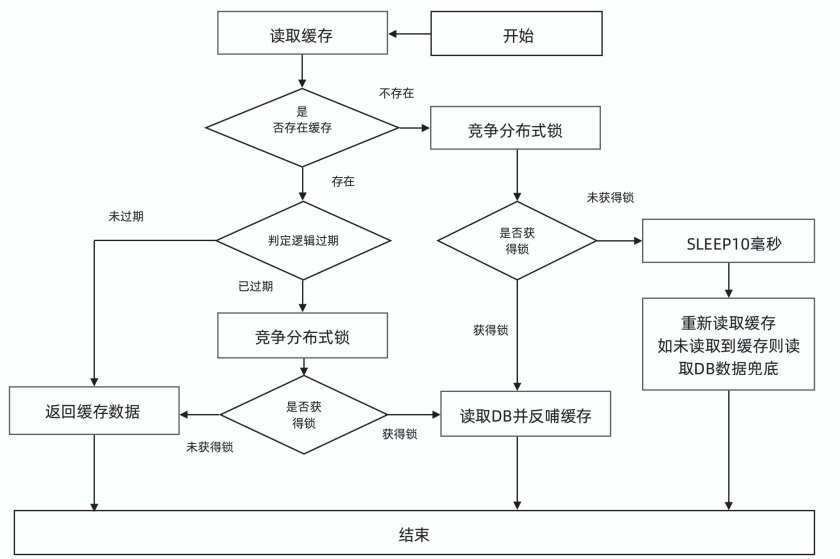
3）基于逆向流水防止重复回滚。

2. 扣减预校验：防击穿缓存设计

如果所有的库存扣减拦截都放到最后一步扣减库存来做，无疑会对数据库造成巨大的冲击。毕竟库存有限，总有一部分请求是无法购买成功的，这一部分请求流量应该在库存实际扣减之前拦截下来。

预校验阶段即“高并发读”链路时，进行一些不影响性能的检查操作，比如请求的合法性校验、票品是否可售、可售数量是否足够、渠道是否授权等。在这个阶段，系统采用分布式缓存来抵抗高并发读。

传统意义的缓存比如 Memcached，一个比较头疼的问题是缓存击穿。为了保护 DB，我们对传统缓存做了一层封装，即在传统缓存写入时增加当前毫秒级时间戳属性，读取缓存后与当前时间戳比较其差值判定其是否逻辑过期，如果逻辑过期则在竞争分布式锁后读取 DB 数据并反哺缓存。



（图：防击穿缓存流程图）

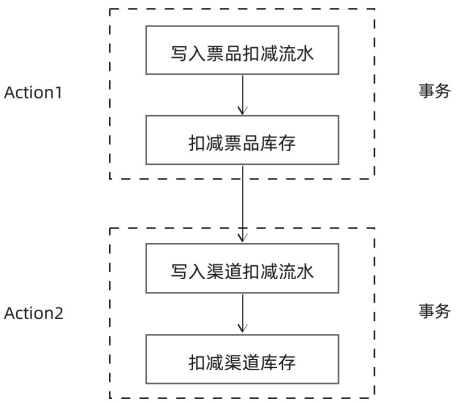
3. 库存扣减的一致性保障

库存扣减要同时扣减票品库存和渠道库存，所以扣减阶段势必要进行拆解执行，拆解成最小粒度执行是为了命中数据库热点补丁，达到快速执行的目的。但拆解后面临的问题是如果一个扣减动作执行失败，如何回滚已经执行的动作。基于分库分表的限制，以上两个扣减动作显然无法使用数据层的事务做保证。本库存系统选用的是“职责链模式”进行库存扣减步骤的定制化。

我们把“票品库存扣减”、“渠道库存扣减”作为职责链中的两个独立 Action，任何一个 Action 发生异常则向上抛出，在上层处理异常并回滚已经执行过的 Action。确保“票品库存”“渠道库

存”的扣减共进退，理论上达到数据一致性的事务保障。

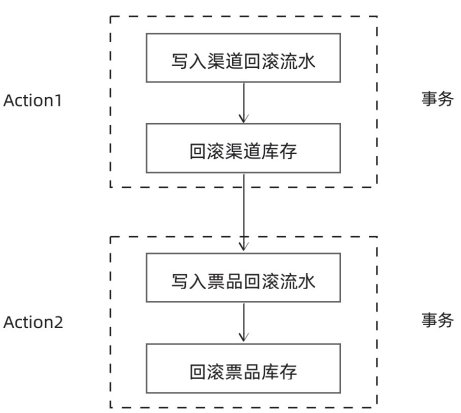
在动态组合职责时，两个 Action 的执行顺序相当关键。在扣减顺序上，首先扣减票品库存，因为票品库存的共享性，在多渠道同时扣减的情况下，票品库存一般先于渠道库存售罄。所以首先扣减票品库存相较于首先扣减渠道库存可以减少很多回滚情况的发生。



（图：库存扣减顺序）

4. 库存回滚的一致性保障

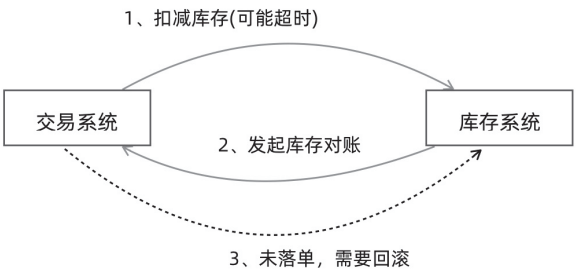
在回滚职责链中，动态组合职责时，同样讲究执行的先后顺序，与正向扣减刚好相反，首先回滚渠道库存，然后回滚票品库存。这样在渠道库存已经回滚的情况下如果回滚票品库存发生异常，结合上述的扣减顺序，票品库存的超卖扣减几率将大大降低。



（图：库存回滚顺序）

5. 异常场景的实时对账

交易系统与库存系统之间的对账可以让双方系统互通有无。如果交易系统在扣减库存时发生读取超时，库存系统实际已经完成库存扣减，但交易系统并未成功落单。则库存系统在实际扣减成功后会发起对账消息，此时交易系统对账结果显示交易系统并未落单，立即主动发起库存回滚请求，撤销刚才的事实扣减。



（图：系统对账交互图）

6. 数据库层面的极限优化

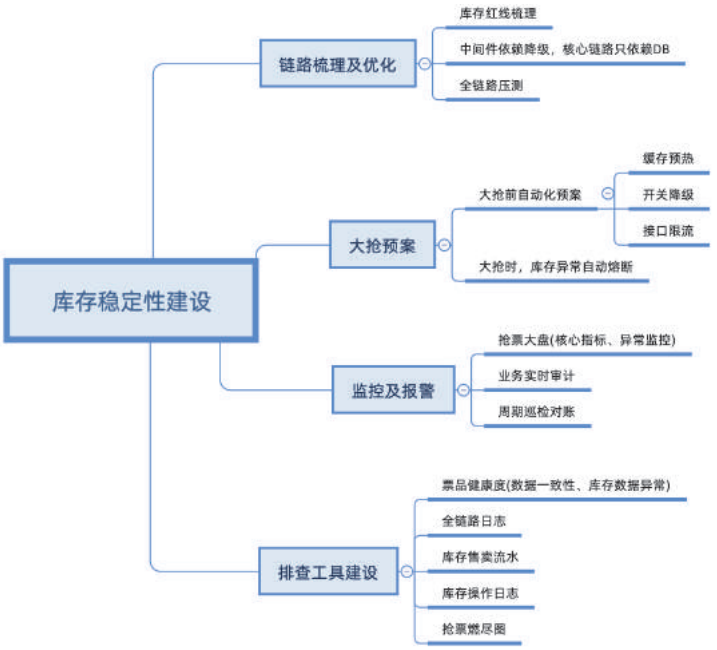
库存扣减会在同一条票品库存或者渠道库存记录上“高并发写”。此阶段会直接面对数据库的一条记录进行高频次的写操作，业界普遍采用的方案是应用层排队或者行级乐观锁保障对同一条记录进行操作的并发。考虑到应用层排队有损性能，库存系统采用了较为理想的数据层排队，主要基于阿里的数据库团队开发的针对 InnoDB 层上的补丁程序（patch），可以基于 DB 层对单行记录做并发排队，从而实现秒杀场景下的定制优化。

7. 高并发场景下日志优化

业务日志对于发现业务问题极为有效，也有助于测试阶段的问题定位。上线后有详细的全链路日志协助问题排查。为了避免引入的外部 jar 输出不必要的日志，所以上线后的系统业务日志级别调整为 ERROR 级别。另外，为了防止大抢期间日志落地造成 IO 竞争导致性能下降，库存系统裁剪了大量的无效日志输出，并且采用 Logback 的 LoggingEventAsyncDisruptorAppender 异步写日志，防止写日志阻塞(极端情况下有抛弃日志的可能)引起的系统性能下降。

三、库存稳定性

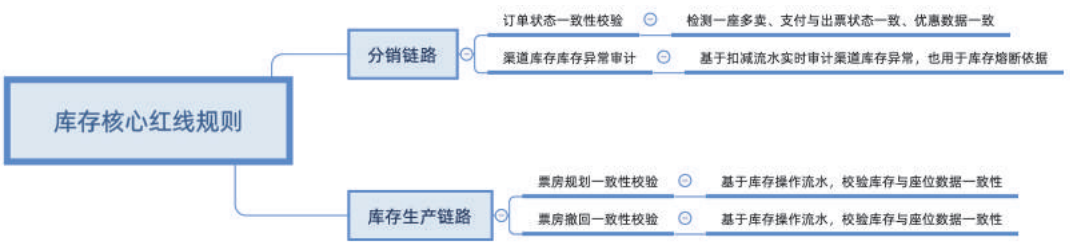
稳定性目标：2 分钟内发现(业务监控 100%覆盖)，5 分钟内定位，15 分钟内止血。



（图：库存稳定性建设包含内容）

1. 链路梳理及优化

库存红线：是库存系统的生命线，其规则如下图：

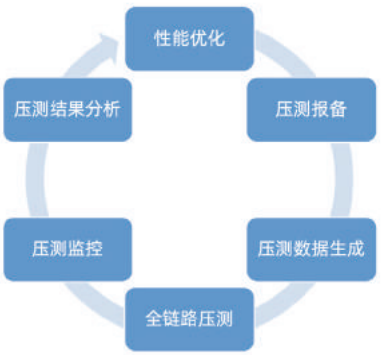


（图：库存核心红线规则）

强弱依赖：对库存的核心链路进行强弱依赖梳理，保证核心链路只依赖 DB，其他依赖均可降级。

2. 全链路压测

完备的常态化全链路压测体系，保证库存的一致性和性能符合大抢预期。



（图：全链路压测流程）

3. 大抢预案自动化

预案自动化的核心在于两点：哪些项目是热门项目？哪些预案需要执行？

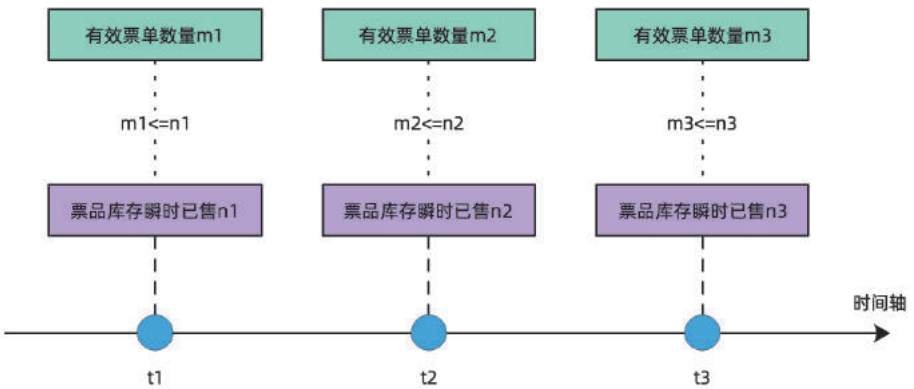
1）发现热门项目，预案根据 PV 和参与人数，自动发现热门项目，加入预案执行队列。

2）自动执行预案项，根据预先编排好的执行时间自动对抢票场次执行缓存预热、开关降级和接口限流等操作。

（图：大抢预案自动化配置示例图）

4. 库存异常熔断

在抢票场景下，库存异常的诊断是非常困难的，基于库存扣减流水可以准确计算出库存在某一时间段内变化的情况，和当前生成的票单对比后即可以判断库存是否不一致，进而触发渠道熔断，防止情况恶化。



（图：基于时间轴的异常熔断检测）

5. 监控及报警

在监控方面主要有：核心指标及异常监控、业务实时审计和周期巡检。

1）核心指标及异常监控

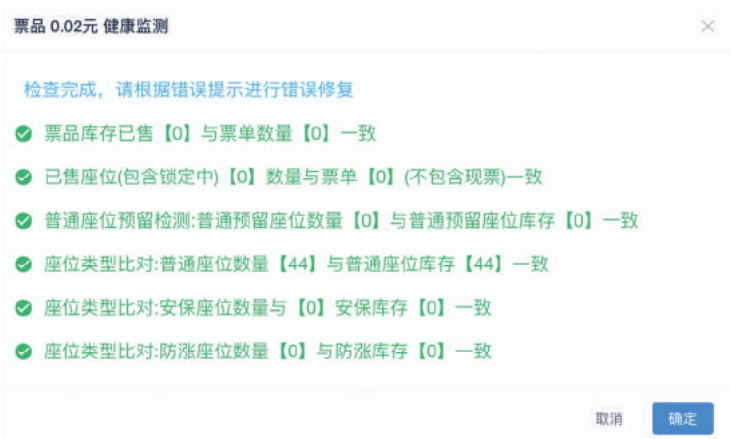
核心指标及异常监控可以有效监控下单成功率和 RT 抖动，如果有异常错误码或者抖动可以及时发现，快速介入；

2）业务实时审计

业务实时审计是数据一致性检测的重要手段，由于链路超时或异常下状态不一致的时进行报警；

3）周期巡检

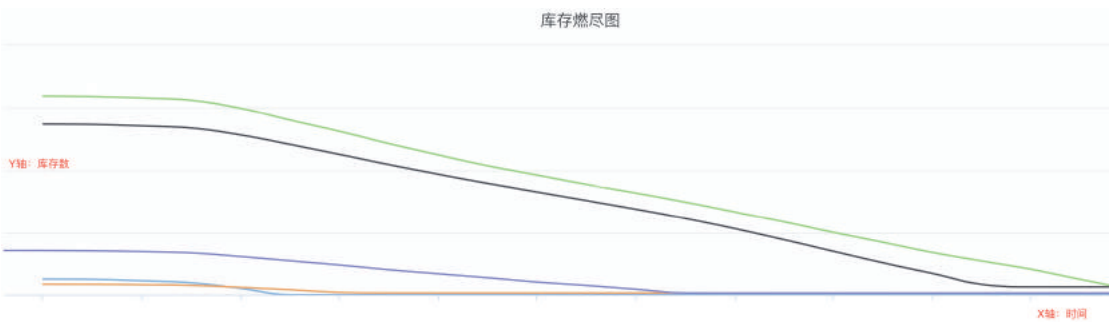
周期巡检是对上述监控的重要补充，业务实时审计其实是数据状态变化触发的，当没有触发事件或者没有覆盖到其他规则时，周期巡检可以避免监控遗漏，虽然是非实时的，但是也在实践中起到不错的效果。



（图：票品健康监测示例图）

6. 排查工具建设：快速定位，快速止血恢复

（图：排查工具包含内容）



（图：库存燃尽图示例）

四、总结

以上是票务行业库存系统建设及项目中的一些探索经验，基于正逆向流水的防重，预校验拦截，防缓存击穿，职责链模式的步骤拆分，系统间对账，数据库热点补丁的使用，日志裁剪优化等一些列技术手段，达成库存高性能、强一致性目标。

通过梳理库存红线，常态化全链路压测，异常熔断机制，监控报警以及排查工具，大抢预案自动化建设，持续提高库存系统稳定性。

大麦票夹：从工具到服务的技术演进之路

作者| 阿里文娱技术专家 义豪

本文基于大麦票夹过去工作以及未来思考总结而成。按基础电子票功能、行业无纸化、服务化的三个阶段阐述了大麦票夹的发展及背后技术架构的变化。希望以此引发大家一起思考：技术架构如何提前布局以满足业务在不同阶段的要求。

一、背景 – 什么是票夹

票夹，顾名思义，装票的小夹子。在大麦 APP 这个曾经工具性很强的应用中，承担起了电子票收录和使用的功能。

用户在大麦购买一张演出票，如果是电子票，出票后用户就可以在票夹找到这张票，看到票的场次、看台、座位号等基本信息。在入场的时候，可以展示票的二维码信息给验票设备，扫码验证通过后即可入场。

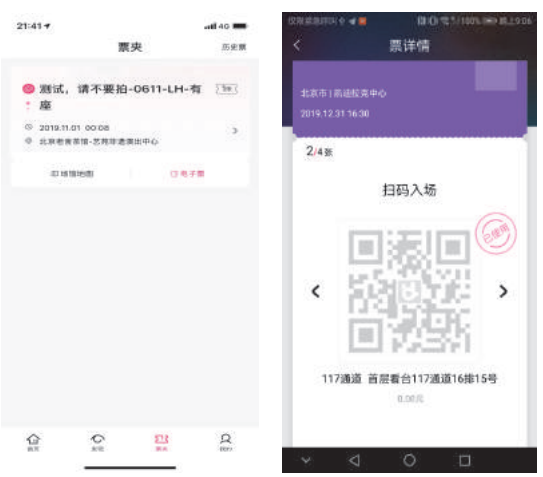


图 1

从上图可以看出来，为了方便使用票，所有票是按场次组织在一起的；如果该演出场次可以换纸质票，会有换票点的位置引导；在验票后会更新票的状态……

可以说，最初票夹的所有功能和交互设计都是为了方便用户行使最基本的使用权利。这也是第一阶段票夹所做的主要工作。下面从第一阶段开始介绍票夹的发展历程。

二、电子票基本服务

票夹第一阶段的重点是实现电子票服务的基本功能和可靠性。

基本功能很好理解，电子票如何加密，在 APP 侧的二维码如何生成；在验票端如何校验二维码是有效的。

为了利用好阿里生态，在大麦 APP 落电子票的同时还向用户的支付宝卡包中落入电子票，让用户不但可以使用大麦 APP 入场，还能使用支付宝卡包入场。

下图系统的描绘了大麦票夹的电子票基本服务。其核心还是二维码电子票部分，稍详细介绍一下。

图 2

1. 动态二维码

二维码电子票分静态二维码和动态二维码。

- 静态二维码就是一个固定的码，实现简单，但是安全性较低，如果二维码被拍照或者截图，理论上相当于这张票被复制了。

- 动态二维码在整个产票落票过程进行了多重签名和加密，在端上还通过算法结合 APP 环境进行了动态加密生产可变化的二维码，有效防止票被复制。

目前大麦主流使用动态二维码模型的电子票，在大麦 APP 和支付宝卡包同样支持。

2. 客户端本地化缓存

在现场验票的场景下，需要面对一些极端条件，其中之一便是用户手机的无线网络。目前 4G 网络已经普及，但是在大型演唱会的时候，场馆附近会聚集大量的人，每人一个手机很轻松就超过了 4G 基站的承载能力，导致现场用户手机端的网络很差或者几乎处于无网状态。

如果该演唱会采用电子票入场的方式，那就是场灾难，大批用户因为打不开大麦 APP 中的电子票无法验票入场。针对这种情况，大麦 APP 采用了客户端本地化缓存的方案。

APP 将请求过的票数据保存在本地数据库中，只要用户曾经在 APP 中打开过这个电子票，以后即时没有网络，也能从本地数据库中把票读取并展示出来。

但是电子票可能并不是下单完成后立刻出票的，也就意味着用户不一定有机会亲手点开电子票，我们采用的方法是在真正落票的时候，给 APP 端一个推送消息（通过长连接通道），APP 接收消息后会从后台拉取电子票保存到本地。

3. 票的状态实时更新

这其实是用户体验的提升点，即在验票扫码后，用户 APP 上的二维码立刻被打一个“已使用”的戳。

看似很简单的一个功能，其实需要全链路的配合。

首先，验票端在现场网络不是那么稳定的情况下，需要及时的将验票状态发送到验票服务系统；

其次，验票服务系统需要把状态更新告诉票夹；

再次，票夹需要通过长链接通道将票状态的变换推送到大麦 APP。

全链路整体延迟需要控制在 2 秒左右，再长用户就没机会看到实时状态变换了。

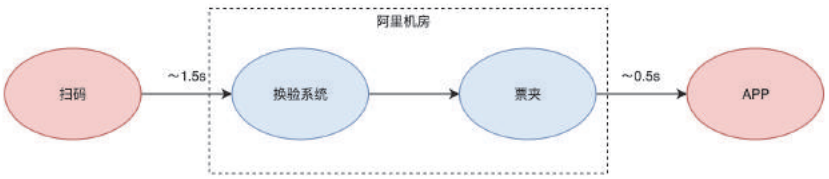


图 3

4. 转赠

转赠服务是推广电子票特别是动态二维码形式的电子票必不可少的服务，是纸质票体感在电子票领域的自然延伸。

设想 4 个人相约去看一场球赛，其中一个人买了 4 张纸质票，因为座位不是连续的，入场时间不同，入场口可能也不一样，所以需要把票分给每个人，大家各自持票入场。

如果是电子票也存在这个场景，总不能大家先后用同一个人的帐号登录大麦 APP 入场，最自然的做法就是购票人将电子票以转赠的方式分发给每个人，大家各自持 APP 入场。

用户下单购票的时候，票和订单是绑定的，但是到转赠的时候，票和订单就无关了，只是持有人的信息发生了变换。

这里票夹维护了当前票和用户之间的关系，即一张票当前的所有人是哪个用户。

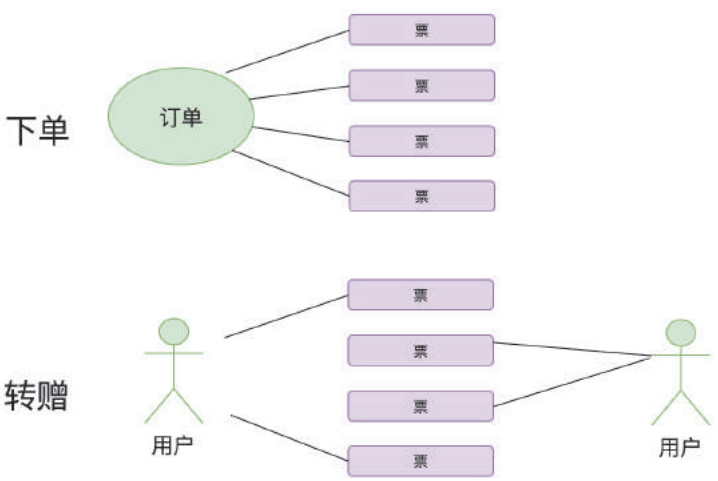


图 4

三、行业化

整个演出市场的总票量中大麦占了很大比重，可以将整体演出市场的票分为三大部分：

1）大麦自产的票；

2）大麦的部署给场馆方的麦座系统生产的票；

3）其它公司产票系统生产的票。

大麦票夹最初实现的电子票服务只是针对大麦自产票这一部分。但是要成为电子票的先行者，需要对整个行业的无纸化体验产生足够影响力。为了实现这个目标，大麦票夹需要具备开放性，能落其他票仓生产的票。

1. B 端多商户的支持

对于票夹来说，一个非大麦的商家就是一个入驻商户，独立产票，并使用票夹提供的电子票能力，同时又彼此隔离（数据和配置都隔离）。要做到这一点，需要解决很多问题。

1）统一的数据模型和开发接口（OpenAPI）

在面向大麦自产票的时候，因为只有一个用户，票夹专门设立了一个接入层应用来做票的接入。这个应用监听交易发出的消息，并查询产票系统获取票的详细信息，同时还查询场次、场馆等信息最终落票。但是在支持多商户后，票夹不可能针对每个商户都去了解对方的下单、产票逻辑来做票的接入。可行的办法是定义一套统一的接口和模型，由商户自己决定何时落票，自己按票夹的要求组织好相关信息，票夹只做信息完整性的校验。

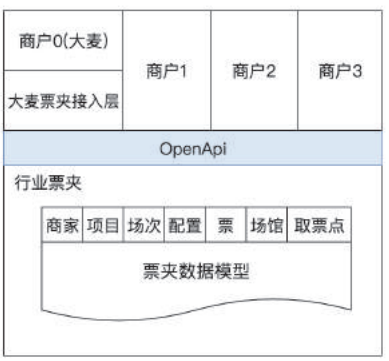


图 5

2）商户间的分销关系

这里之所以叫商户而不是租户，是因为商户之间是可以有关联的，并不是简单的增加一个商户 ID 做隔离就一切 ok 了。

例如，商户 A 是产票方，除自己卖票外，还把票卖给商户 B、商户 C。大家落票的时候都是各自落入票夹，但是这些票是同一个场次的，商户 B 和商户 C 只需要使用产票商户 A 的场次信息即可，场次相关的配置也可以复用一套。

在行业票夹中就做了这样的支持，可以让商户直接复用产票方的场次数据，从验票系统的角度，只有一套场次，那就是产票方的场次数据，与验票相关的信息要与产票方保持一致。

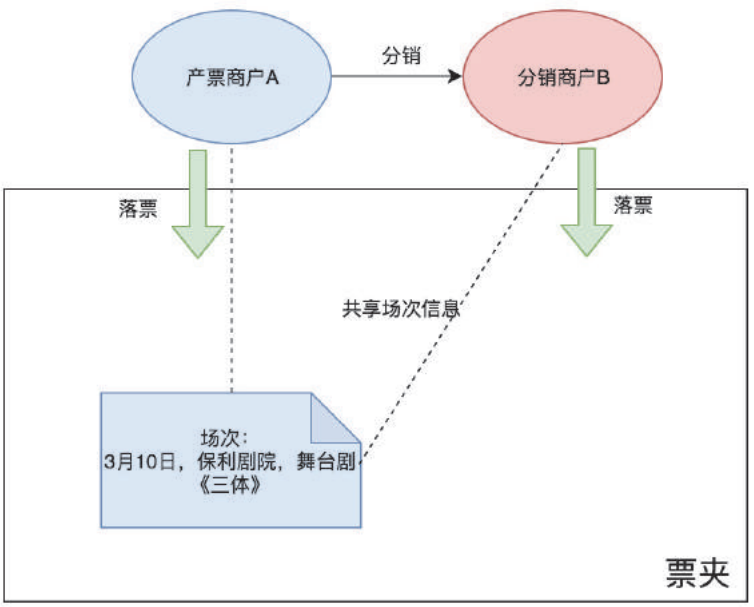


图 6

2. C 端页面的复用与隔离

前面提到了，当电子票落入票夹，实际上就和订单没太大关系了，票夹主要维护的是用户和票的关系。而每个商家都是有自己的用户体系的，商户的隔离同时也要求 C 端用户的隔离。

大麦票夹是通过一套定制化的 OAuth 协议完成各商户的用户对票夹访问时的鉴权和数据隔离的。

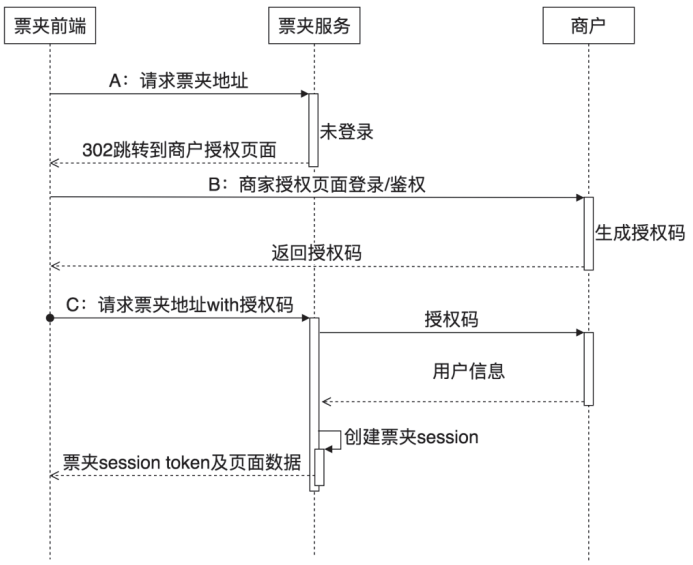


图 7

四、聚焦服务

1. 为什么要聚焦服务

前面介绍的两个阶段，一个是提供基本的电子票服务，另一个是将基本的电子票服务推广到更多的用户。虽然对 B 端实现了平台化，但对 C 端始终还没有脱离一个工具化应用的范畴。

票夹这块阵地除了是用户使用票的入口，更是用户和演出方/场馆的连接点，我们的用户如果仅仅在入场或者换票的时候才进入票夹，那其实是巨大的浪费。

对用户来说，我们可以通过票夹让其在观演前后享受到更贴心的服务；对场馆/主办来说，可以通过票夹触达用户、了解用户、挖掘更多周边的价值。这两点如果做好，不但让我们的用户更有粘性，而且还会吸引更多非大麦的用户使用大麦 APP，进而转化成大麦的用户。

因此票夹需要从一个工具型的功能模块进化为一个服务的载体，串联整个用户购票、观演的生命周期，打通阿里生态甚至外部服务。

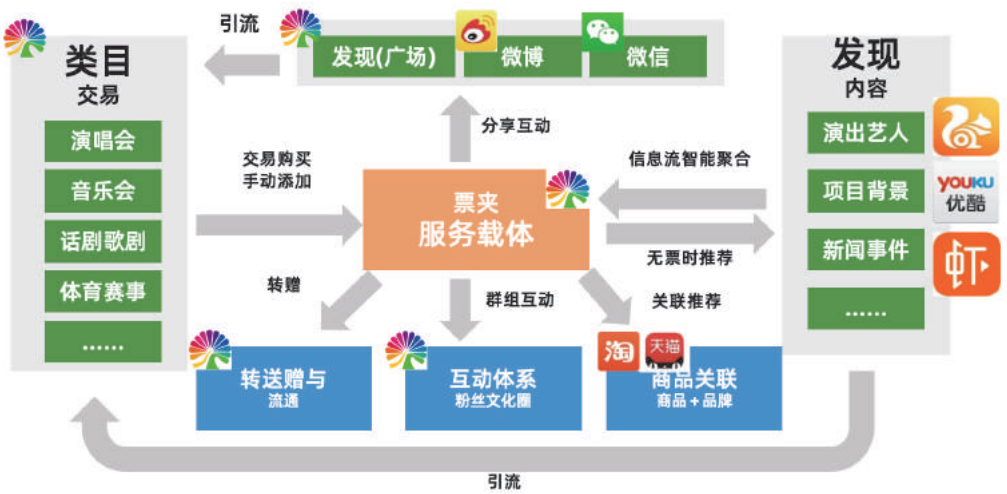


图 8

2. 票夹服务周期和对象的拓展

用户观演的整个周期并非始于演出开始的那一刻，从用户有观演的打算开始，这个周期就开始了，一个用户可能经历了想看、决定购票、下单、等待出票、筹划出行观演、出行、入场、观演、观演后这样一个漫长的周期，这个周期可能长达一个月甚至几个月，这是和看电影有巨大区别的地方。

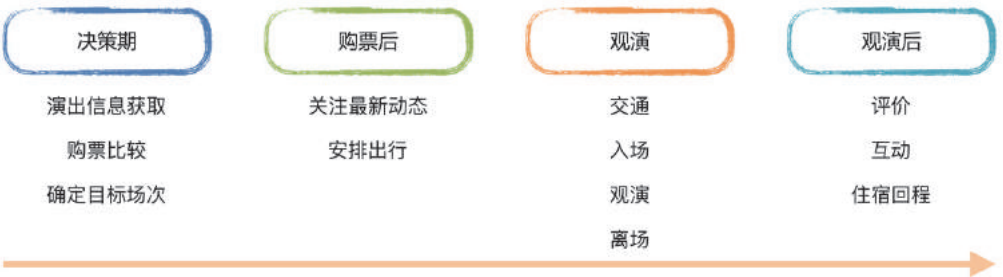


图 9

一个工具化的票夹是服务于“购票后”和“观演”这两个阶段。而一个服务化的票夹是可以向前和向后拓展的，设想用户在未下单的时候就可以通过关注等方式为用户在票夹创建一个场次卡片，虽然还没有买票，用户已经可以开始享受服务了。即使用户没有选择在大麦买票，依然

可以享受后续的一系列服务。

当票夹服务化之后，就不仅仅满足提供电子票功能了，购买纸质票的用户一样可以在票夹展示一个服务卡片，重点聚焦在服务上。这就对票夹的业务模型和架构提出了新的要求：

- 从一开始的处处以票为中心进化到以演出卡片为中心，服务开始的起点从落票变为了用户创建一个场次卡片；
- 整个服务周期的不同阶段，为用户提供不同的观演服务。

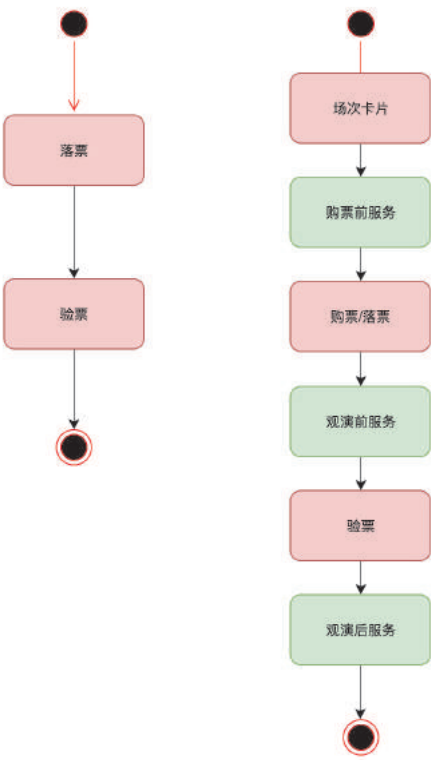


图 10

3. 观演服务组件化

当票夹的服务周期和对象都做了扩展，必然需要引入更多样化、个性化的服务。票夹在架构上需要支持大量服务的接入能力，同时可以在每个观演周期动态化、个性化的提供给用户使用。

我们希望对票夹的观演服务进行组件化设计，组件化这个概念很大，总体来说是关于软件模块的封装、定制和复用，在不同领域有不同的解释，这里不想对组件化展开讨论，但是针对票夹有必要思考清楚组件化的内涵和外延。票夹的观演服务组件化关乎如下几点：

1）观演服务的抽象与封装。

- 票夹本身并不实现这些服务，而是服务的搬运工，充当整合与桥梁的作用为主。对于整合与桥梁这个目的，我们是可以将五花八门的观演服务进行抽象和归类的，这样票夹可以通过对少数几种模型的支持来快速接入各种个样的观演服务。比如，服务可以抽象为类 banner 跳转类、信息透出类、三方客户端 SDK 类等几种类型。
- 前后端对每种类型的组件数据定义、展示和逻辑的封装。

2）每个观演服务组件是可以跨商家、跨场次、跨场景复用的。

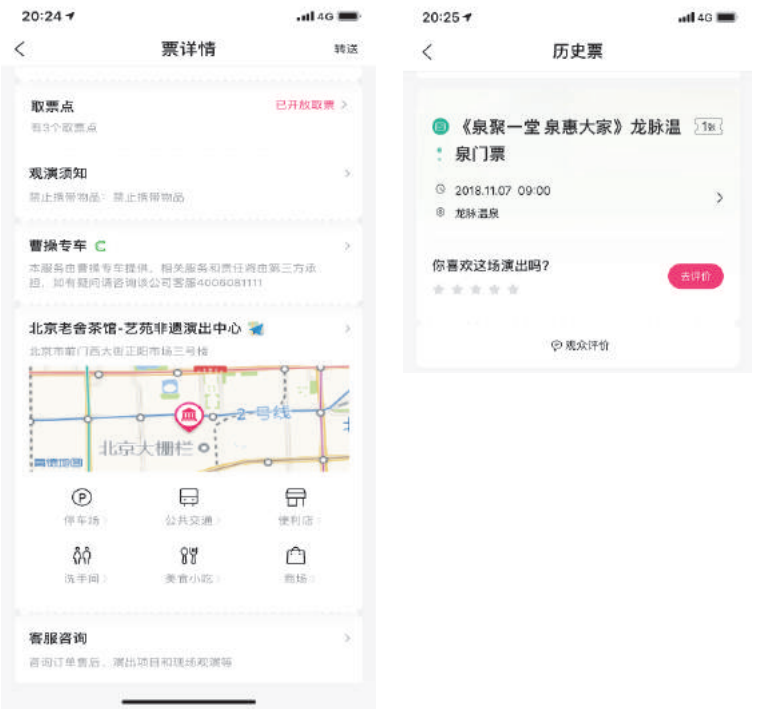


图 11

- 组件是观演服务的最小单位，整个票夹平台维护所有支持的服务组件，每个商家可以定制自己支持的服务组件列表。每个观演服务组件之于一个商家是可以订制的，可订制的

范围被明确定义并将操作接口暴露给商家。

- 每个观演服务组件之于一个场次是有实例化数据/配置的，这部分数据/配置每个场次都可以不同。
- 组件在 APP 出现的场景是可以订制的，可以出现在场次列表，也可以出现在场次详情；可以出现在观演前阶段，也可以出现在观演结束之后。

3）服务组件是能够支持个性化的。

- 针对不同的用户类型、推荐不同的观演服务。端上给用户展示的观演服务都有哪些，顺序是什么是可以进行个性化控制的。

4. 服务的接入

这部分主要是 B 端的范畴，我们的服务提供方可以是大麦（换验、转赠这种基本服务），也可以是第三方。第三方又可以进一步分为阿里系的服务提供方和阿里外的服务提供方。

- 对于第三方的服务，是需要进行入驻管理的，入驻可以限定服务露出的目标组件、人群、项目类型、场馆等等条件，甚至可以千人千面；
- 对于阿里外部的服务提供商，进行服务整合是需要外接的，及彼此接口级的互通。这部分也是可以与三方体系打通，通过统一的外接平台实现服务整合。

五、总结

本文从电子票基本功能展开，拓展到行业无纸化以及未来的服务化，思考和总结了票夹在其中的作用及应具备的技术能力。这些都是基于过去的环境和技术趋势，而对于未来，随着大的行业环境的变化以及技术趋势的改变，需要持续的创新，这里抛出几个问题请读者一起思考一下：

- 基础的电子票能力从安全性、可靠性和便利性方面是否还有改进的空间？
- 行业无纸化的推进过程，如何做到对不同票库进行电子票能力的输出？
- 在行业电子票验真和票流转方面，如何形成行业公信力？
- 围绕电子票，如何将服务更有效的串联在一起，为用户、场馆、内容方提供更大的价值？

分区定价：座位上玩转“精细化运营”

作者| 阿里文娱开发工程师 暮平

分区定价就是将同一个场次的座位分割成不同的区域，以不同的价格进行售卖。不同的座位，不同的价格，在上座率和价格之间求得一个平衡，帮助影院实现收入的最大化。同时，通过异业合作或者绑定会员权益等手段，将权益与座位“对号入座”，实现更加精细化的运营策略。

一、技术挑战-不仅仅是改个价格

1. “同场同价”-在线选座系统的地基

分区定价要想落地，对于技术来说，并非是改个价格那么简单。

自 2010 年，电影行业出现在线选座业务以来，场次都是定价的最小粒度，影院会根据影厅及放映影片、放映时间等对整个场次的座位进行统一定价，这就是所谓“同场同价”。这个思维定势一直贯穿在在线选座系统的设计实现和升级过程中。在线选座系统的基础数据、交易、网关、营销等核心服务模块，都是基于上述思路进行搭建。

发展到今天，在线选座的业务绝非仅仅是锁座-下单-出票那么简单，如果说 2010 年的系统规模是一栋平房，那如今的规模可以称得上是 500 层的高楼了，那么要改动从系统搭建初期就已定下的规则，就如同重新打造这幢高楼的地基一般，这个挑战可想而知。

2. 怎么做分区定价，不同系统商有不同的说法

淘票票作为领先的观影购票平台，中国电影市场的很大一部分票房是由淘票票的用户贡献的。然而，淘票票并非电影票的最终出票者，影院才是。淘票票通过对接影院安装的具有售票资质的售票系统，完成出票。

目前，具有售票资质的系统商有七家，如凤凰云智、佳影、鼎新等，在系统商的基础上，还有搭建自己售票能力的电商，如大地、万达等等。上述提到的系统商加电商（下文统称合作

商），代表全国近万家影院，定义了影院本地售票系统的接口协议和标准。对于分区定价，他们接口层面有不同的诠释。这就为我们带来了第二个挑战，如何适配众多结构不一的分区定价接口。

3. 如何快速响应？

时间是个永恒的话题。如何在较短的时间内，面对较大的技术复杂性，去支撑业务的急速变化，这也是我们在实现分区定价过程中遇到的问题。

二、方案–控边界，以不变应万变

1. 思路

我们从业务建模出发，构建问题空间模型，然后落实到系统上，逐个击破，用系统建模的方式产出解决方案模型，最后用技术手段实现。

根据第二章节的阐述，我们抽象出分区定价的两个核心问题：

1）售票的定价维度从场次精确到座位；

2）适配各个合作商不同的分区定价接口。

按照领域建模的沉淀内聚的原则，尽量将变更收敛控制在相关的业务域内。因此接下来我们针对上述两个问题进行抽解、推理：

a）将座位的定价维度精确到座位，分区定价，核心就是价格，价格属性是贯穿整个服务集群的重要属性。基础数据、交易、营销甚至网关系统都依赖价格。然而价格本身并不是一个独立的模型，而是商品模型的一个属性，这里的商品模型也可以认为是场次或者排期模型。基础数据、交易、营销等各个业务域所依赖的是商品模型，并非价格。一个普通场次对应一个商品，而对分区定价场次来说，可能一个场次可能对应多个商品。因此针对这个问题我们的解决思路是：抽象分区定价数据到商品模型，交易、营销等模块依赖商品中的价格属性去实现收单、交易履行等模块，尽量不感知分区细节逻辑；

b）适配各个合作商的分区定价接口，我们的原则是尽量减少其对上层业务的侵入，将其差异屏蔽到底层模块，并对上层提供统一的接口。虽然各个合作商提供的分区定价服务有所差异，但是基本上可以抽象为排期拉取、锁座、解锁座位、出票等模块，而这些底层模块与上层的交互是统一且确定的。因此针对这个问题我们的解决思路是：网关域适配个合作商的分区定价接

口，组装统一结构的基础数据，暴露统一标准的锁座、出票等服务，将分区定价的基础服务能力沉淀内聚到网关层。

2. 架构与实现

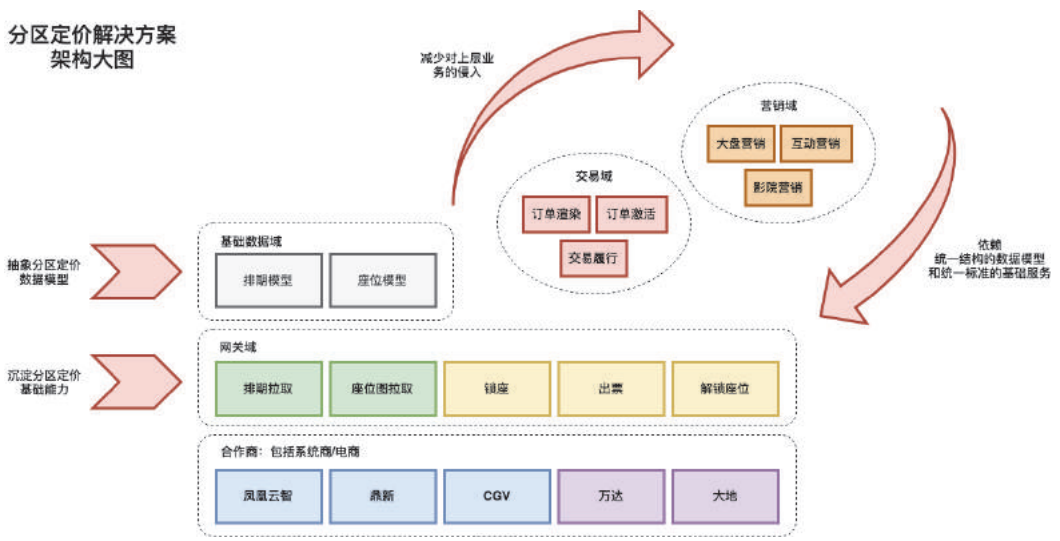
综上所述，淘票票对于分区定价的实现方式可总结为以下几点：

1）网关层沉淀封装排期拉取、座位图拉取、锁座、出票、解锁座位等能力，并对外暴露统一标准的基础服务；

2）抽象分区定价的数据到商品及排期模型中，对外提供统一结构的数据模型；

3）上层模块基于底层的基础数据和服务，以通用逻辑进行适配，尽量减少对上层业务的侵入。

具体如下图所示：



3. 最终答卷

通过上述的解决方案，淘票票在较短的时间内完成了对分区定价业务的支撑，覆盖了行业内多数合作商和影院，同时系统具备一定的可扩展性和复用性，能够以较低成本实现合作商的对接。

四、思考

1. 快速的业务变化，如何应对，并沉淀技术能力？

快速变化是互联网行业一个无法回避的问题，由于技术人员距离业务前线相对较远，在应对过程中难免显得被动和疲倦。“面向复制-粘贴的编程模式”“先 run 起来”“赶一下进度，后面再重构”、“必须马上上线”等等这些传统应对方式就会大显身手。然而，这些手段纵然能够“快速”支撑业务，也造成了一些恶果。其一，代码的可复用性和可扩展性极差；其二，技术债高垒，影响开发效率，增加变更风险，降低技术竞争力，甚至反过来影响业务和团队。在快速交付的前提下，在分区定价项目的实践过程中，我们总结出以下几点经验：

1）规范技术债的管理机制，对于需要优化的模块，在项目上线之初，就要确定迭代计划。这是最简单却很容易见效的手段；

2）定位核心变更，并将核心模块收敛到尽量少的业务域中，沉淀能力，便于统一管理、复用和扩展。比如在分区定价架构设计中，我们把核心模块落地在基础数据域和网关域，在不影响业务交付的前提下，也使得系统具备一定的扩展和复用能力；

3）拉进业务距离，尽可能多的理解业务诉求、未来规划等等，在设计阶段，把这些信息带入思考，能够使我们在面对变化时更加从容。

2. 业务技术，做业务还是做技术，技术性体现在哪里？

在业务开发的角色上，面对纷繁的业务需求，很多人会产生困惑，到底是在做技术，还是在做业务，其技术含量体现在哪里？

通过上文所述，一些业务需求的复杂性未必就逊色于底层技术。我们产生困惑的原因，更多的还是面对的问题不一样。业务技术面对的问题变化更快、领域横向上更广；底层技术面对的问题更加稳定，领域纵向上更加深入，但是大家解决问题的能力是共通的，比如编程能力、抽象建模能力、结构化思维能力、落地能力等等。

因此业务开发并不简单，只是大家想“简单”了，或者说在用“简单”的方式做业务开发。我们认为，尊重业务需求，夯实编程、抽象建模、落地等能力，并在项目实践中充分思考和应用，业务一样可以做的很“技术”。

2

交易平台

阿里怎样守护产品线上质量？大麦用虚拟机器人搞定

作者| 阿里文娱测试开发专家 烈冰

对于大麦这种客户众多，抢票舆情极易爆发的业务，如何保障好线上质量是极大的考验。大麦针对产品的线上问题分两个阶段进行了专项攻坚，拿到理想效果的同时沉淀出一整套解决方案及技术工具，其中核心技术产品钉钉机器人“麦粒儿”被阿里多个 BU 接入使用以应对线上问题。本文分两个阶段来阐述治理的结果、过程及沉淀。

一、第一阶段：提升线上问题的解决效率

1. 业务属性决定大麦要快速处理线上问题

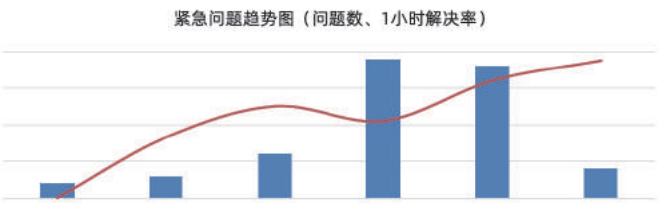
背景：大麦的业务，售卖的是稀缺资源，而面对的客户除了普通的消费者，还有主办方、场馆方和政府公安文化等，这样的业务属性，决定了我们对线上质量的高要求，线上问题的解决效率是我们的第一指标。随着大麦内部一批大型新系统的陆续上线，线上质量的压力越来越大，随即由技术质量牵头对线上问题展开专项攻坚，力保线上质量。

目标：通过专项治理，大幅提升线上问题的解决效率。我们对标了阿里内部各个业务的标准，制定了大麦的核心指标为线上问题的 1 小时解决率。

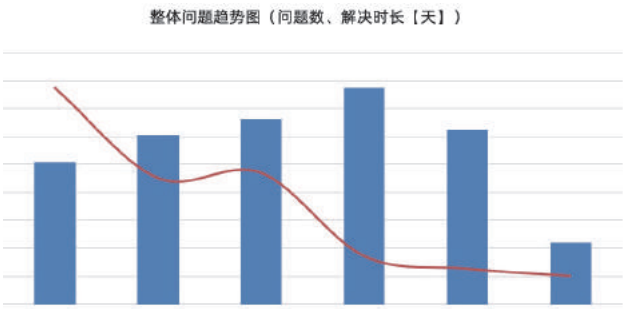
价值：所有影响用户和业务的紧急问题在第一时间内得到解决；打通公司内各个部门，过程结果透明，所有角色信息对等，对质量放心；所有同学对线上生产有敬畏之心，视线上质量为生命线。

2. 经过各部门的通力合作和技术攻坚使核心目标达成

- 紧急问题的 1 小时解决率大幅增长。



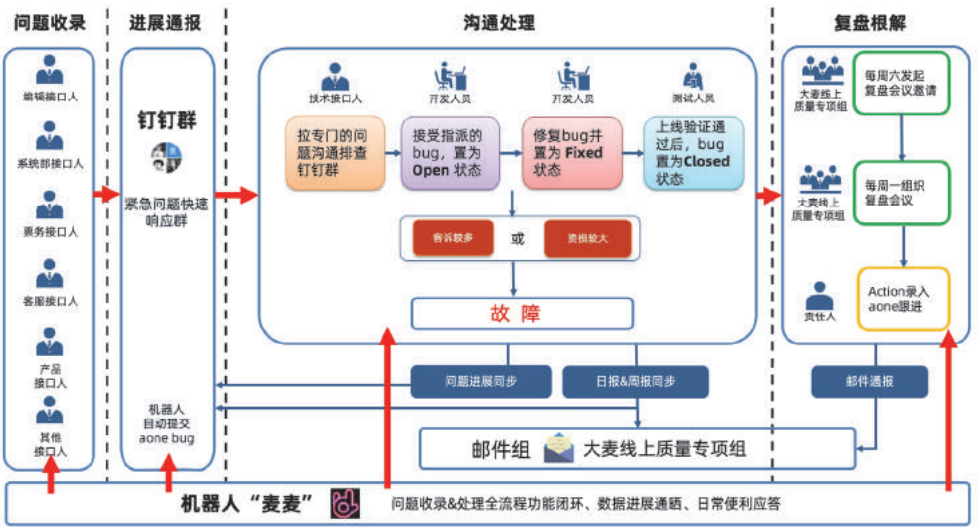
- 问题逐步收敛，解决时长大幅缩短。



- 机器人全流程承接，全角色参与，进度结果全透明。

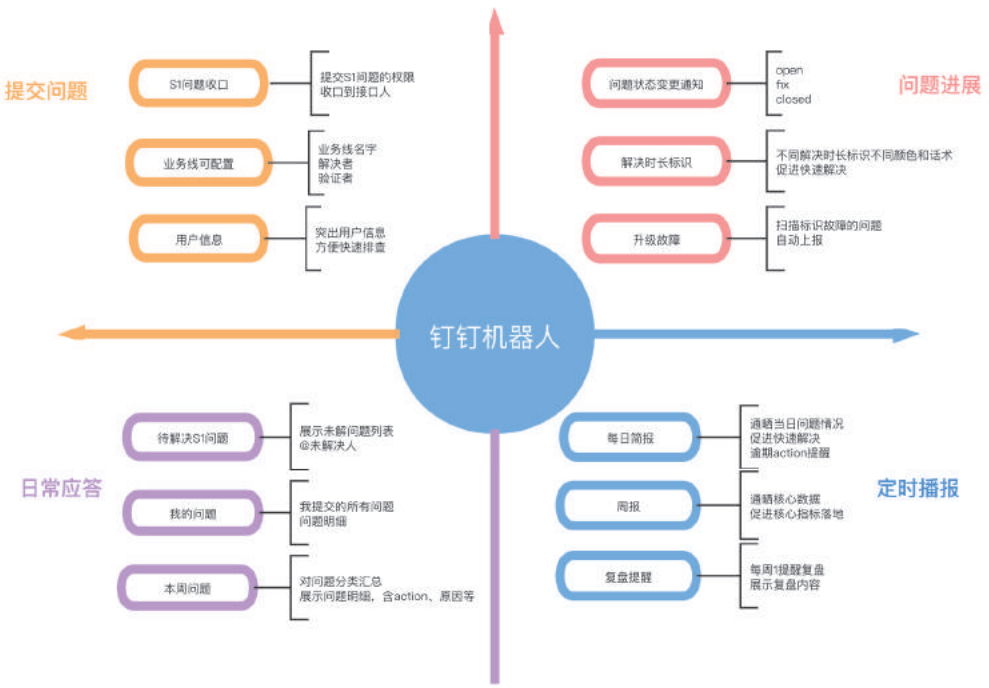
3. 通过建立工具化支撑的处理机制来实现目标

1）面对复杂的线上问题和众多的干系人，我们首先想到的是把问题分级，使那些真正紧急的问题暴露出来，并建立问题的全流程处理机制来标准化运行，如下图：



2）有了处理机制，还需要有工具来承接使之高效运行，我们开发了虚拟机器人依托钉钉群进行问题的收录和沟通，并打通了研发工作平台、邮件系统、故障系统等使问题的快速处理形成闭环，具体功能有：

- 支持上报问题→问题进展同步→日报、周报→复盘提醒→升级故障，形成闭环。
- 通晒紧急问题的核心指标，促进问题快速解决，实现核心目标达成。
- 日常应答，方便主动跟进问题进展，对未关闭的紧急问题直接@接口人处理。



3）影响问题处理效率的核心是问题的定位。在治理过程中，技术团队建立并完善了各自业务线的核心系统监控预警系统，使问题第一时间被发现。同时我们还开发了众多的排查工具并打通全链路排查系统，结合舆情系统的精准信息反馈，使问题可以被快速定位。

二、第二阶段：提升大麦的线上质量水平

1. 重点从问题的解决效率转向问题的收敛

背景：经过第一阶段的治理，线上问题的解决效率已经得到了大幅的提升。但线上质量永

远是我们的生命线，真正影响用户体验的是我们的线上质量，我们随即将目标对准提升线上质量本身，力图实现收敛线上问题。

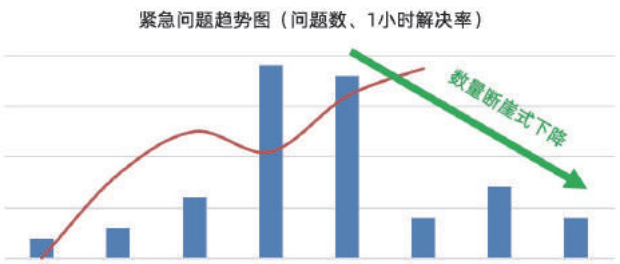
目标：将大麦的线上质量水平大幅提升。核心指标分解为：

- 收敛整体线上问题，腰斩 TOP 问题，控制紧急问题的二次发生率。
- 全面提效，完善机器人建设，全流程自动化承接，减少人力成本。

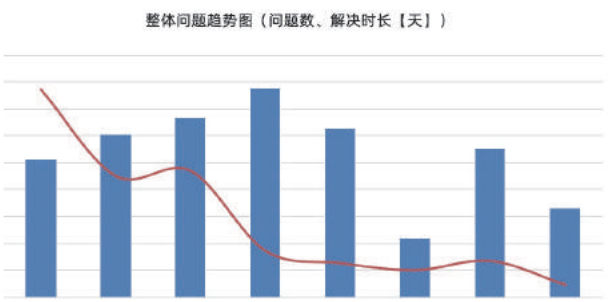
价值：形成一整套专业的线上问题解决方案并有配套工具支撑，阿里集团横向打通；将线上稳定视为生命线，在公司树立稳定压倒一切的共识。

2. 经过各部门同学们的通力合作，核心目标全部达成。

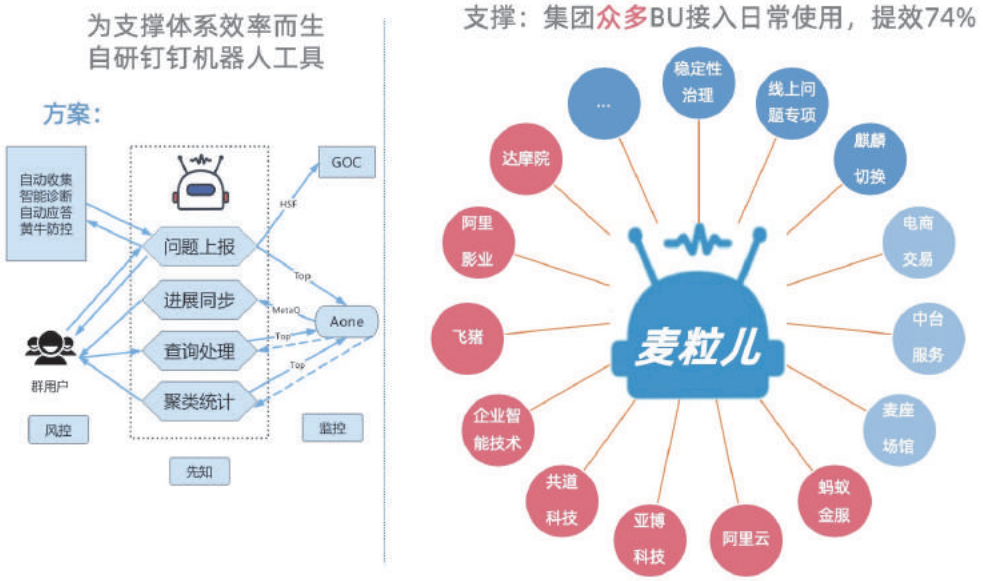
- 线上质量大幅提升，紧急问题断崖式下降。



- 整体问题平均解决时长大幅缩短。



- TOP3 问题被腰斩，紧急问题二次发生率不到 1%！
- 机器人产品化基本完成，支撑业务线自运营，节省人力成本超一倍。“麦粒儿”支撑集团多个 BU 进行线上问题处理。



3. 为了达成核心指标，专项小组主要从专项根解 TOP 问题、完善机器人产品化、止血手册和排查宝典建设三方面出发展开工作，具体事项如下：

1）对 TOP 问题进行专项治理，分级 action 根解严重问题。

- 针对链路问题，成立专项小组进行治理。

我们对历史问题进行了梳理和总结，在多个核心环节进行了多项产品及技术优化，并补齐 SOP 减少人为操作失误，最终使产技问题清零，生产问题腰斩。

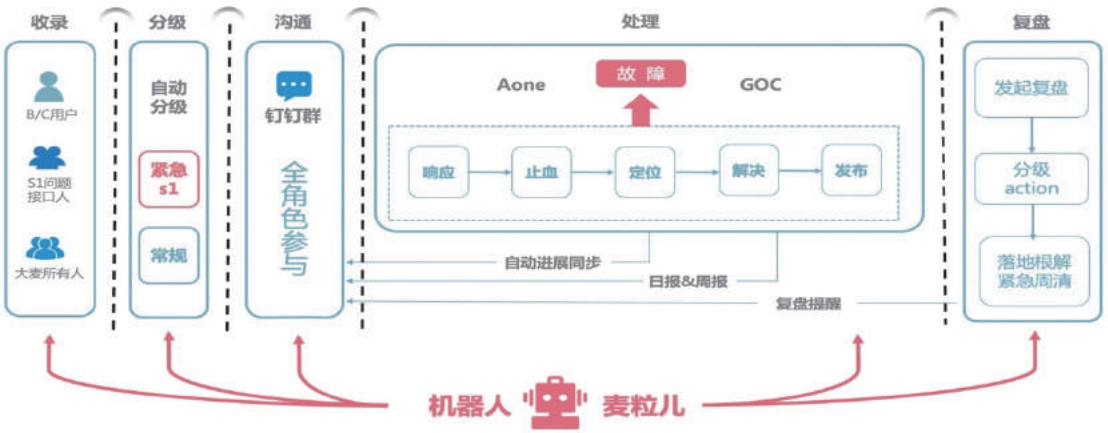
- 制定完善的 action 分级机制并应用于紧急问题中实现问题根解。

为准确找到根解方案、保障 action 及时完结，我们制定了复盘会议规范、action 分类制度和 action 分级处理规则并推动落地，紧急 action 的周清率达到 100%。最终使紧急问题二次发生率不到 1%，超过之前预定目标。

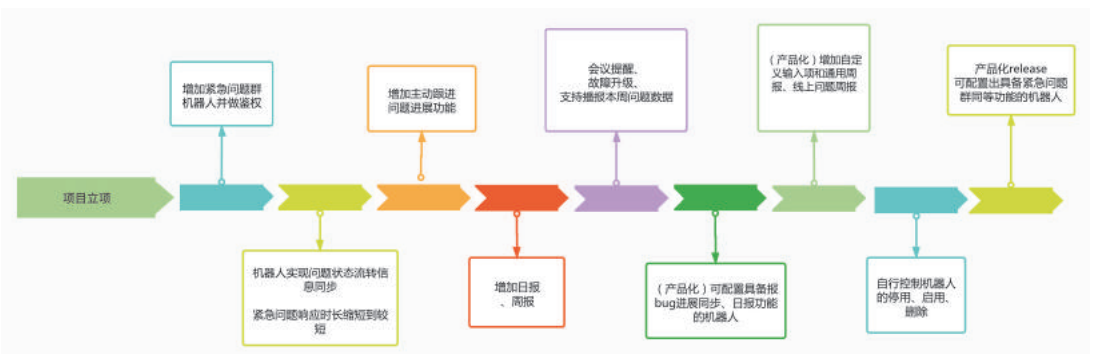
2）完善机器人功能并进行产品化，开始推广至外部 BU。

在一期功能的基础上，机器人二期重点建设了排查止血指引、故障平台打通、各类功能优化等，补齐了全流程闭环并全面提效，完成了创新的自动化承接的线上问题管理方案，主要包含：

- 处理闭环：问题上报（含止血、排查方案）-->处理通知-->问题解决、关闭通知（含解决时长同步及核心指标对焦）-->获取问题原因、解决方案-->复盘提醒。
- 升级流程：问题上报-->问题升级（故障标准）-->对接 GOC。
- 信息同步：支持自动发布线上问题的日报、周报等。



机器人建设里程碑



随着功能闭环的完成，开始进行产品化开发，同时完成多个专利申请；可快速配置个性化机器人，实现分钟级接入全部功能；BU 内部众多问题群接入机器人，整体满意度超 95%；集团多个外部 BU 也进行了机器人的接入工作，小小机器人开始横向打通支撑集团的线上问题处理。

3）完成排查宝典和止血手册建设，指导止血操作并快速定位问题。

- 止血手册建设：

背景：当发生线上问题或故障时，根据阿里安全生产规范，我们首先要做的不是排查问题，也不是解决问题，而是应该立即进行止血操作。争取在最短的时间里，最大程度的降低问题的影响范围。之前在面对线上问题时，大家的止血操作效率低且容易被忽视，专项小组决定产出一份止血手册，指导大家进行止血操作，提高问题解决效率。

工作：我们从实际线上问题出发，借鉴真实线上问题止血的实操经验，对问题进行归类整理，创建了经典场景的止血手册，涵盖了全部的紧急问题，并与机器人结合自动指导大家进行止血操作。

- 排查宝典建设：

背景：专项小组在 TOPIC 一期以解决时长为衡量标准，推动各个业务线建设核心排查工具，从而提高了问题的排查效率。但是随着排查工具的种类和功能的不断增加，工具的操作使用说明没有及时跟进，导致非该业务线的同学在查询使用上存在困难；其次，当遇到需要排查上下游业务的线上问题时，因为不了解其他业务的排查思路和方法，只能等待其他对应业务的同学给出排查结果，导致问题排查效率低。

工作：若想拥有方便、高效的定位问题的方法，不仅在于有好用的排查工具，还在于有清晰的排查思路。我们从实际线上问题排查定位的实际经验出发，收集建设各个核心系统的排查思路和配套工具，形成了核心业务的主要链路场景的排查宝典，包含问题描述、排查思路、详细步骤、参考案例。并与机器人结合自动指导大家进行问题定位。

- 止血手册和排查宝典的整体结果

止血手册结合排查宝典，使大麦整体的线上问题解决时长持续降低。

4）制定线上问题处理流程规范，助力高效自运营。

- 为了解放人力，使线上问题的管理实现自动化的高效运营，专项小组在不断完善机器人建设的同时，还出台了《大麦线上问题处理流程规范》，并依据规范进行周、月维度度量，逐步使线上问题的管理下放到业务线自运营。
- 整体处理流程规范包含处理流程、aone 操作规范、定级规范、复盘规范、action 规范、故障标准、罚则标准等。
- 结合机器人在群问题管理方面的人力成本降低，整体人力成本节省超一倍。

三、总结

随着两个阶段的建设，大麦线上问题的处理效率和线上质量水平均得到大幅提升。但线上质量永远是我们的生命线，稳定压倒一切！未来我们还将继续建设第三阶段，致力于提升自动化和智能化水平，并把整体的线上问题管理方案体系化推广出去。分解为：

- 继续推进机器人建设，打通全链路日志系统实现问题的智能诊断，并结合止血手册和排查宝典建设，实现问题自动应答。
- 继续为新系统稳定保驾护航，专项推进线上 TOP 问题根解，杜绝二次发生。
- 完善整体的线上问题管理方案，支持更多阿里 BU 使用，为线上问题的专项解决提供更好的支撑。

大麦交易融入阿里电商平台之路

作者| 阿里文娱高级技术专家 江新

2018 年初，大麦决定将电商体系整体融入阿里电商平台，到目前大麦基于阿里共享电商平台体系的商品占比 89%，交易占比超过 90%，全部热门项目抢票已经在跑在这个链路上，在这个时间点来从技术角度回顾一下整个变迁的过程。“星环”是阿里的共享平台接入不同业务方的产品技术体系，大麦的融入也正是基于星环的能力。

一、背景

首先来看下在项目启动之前的一些背景情况，主要是当时大麦交易所处的一个状态，以及我们为什么选择去做这件事情。

2017 年底，大麦主交易流程已经迁移到集团内，在交易核心链路上替换了大麦原有的技术体系，大大的提升了交易体系稳定性。上线后，交易方面以及大型抢票再也没有出现过 P2 及以上的故障了。但是从产品技术看来依然存在很多问题：

- 若干依赖服务（尤其是商品、库存）依然在大麦机房，是重大的稳定性隐患
- 交易容量问题，由于商品库存仍然在大麦机房，库存扣减能力比较低，遇到周杰伦演唱会这样的大型抢票，存在大量限流，用户体验很差
- 与共享电商体系互相独立，无法让大麦借力集团的商业元素

纵观上述几个问题，如果我们让大麦电商整体融入共享电商技术体系，都会迎刃而解。尤其是第三个方面，在大麦面临外部竞争对手逐渐强烈的赶超趋势，如何借力阿里巴巴的商业生态元素来发展，成为大麦战略上的重点。

我们一直相信，大麦电商体系融合到共享电商平台，是一个正确的发展方向。但是在最开始的阶段，大家还是有一些犹豫的。大麦有一些来自飞猪的产品技术同学。飞猪所承载的旅行

业务同样是一个和淘宝、天猫差异很大的垂直电商。当时飞猪也没有融入到共享平台，可以看到原因肯定是多个方面，但是之前共享体系还没有足够好的开放能力来容纳差异很大的业务是其中一个重要的原因，其中一个典型的例子是，共享体系是不允许业务方随意外调服务的。

到 2018 年，共享交易已经把平台和业务的边界逐步划分开来，建立了一套平台提供基础能力，业务自行扩展实现差异化逻辑的标准。经过实际调研看到这些进展之后，更加坚定了我们迈出融合这一步的信心。因而在 2018.02 开始，大麦和共享同学正式启动了“候鸟”项目，大麦交易进入一个全新的阶段，整个大麦电商体系也完全进入了一个新的纪元。

二、交易模块化框架 TMF 与星环

1. 基于 TMF 的交易定制

在 2018.02 项目启动的时候，星环尚未完全就绪。大麦业务的差异化定制，主要还是基于交易模块化框架（TMF）在共享交易的各个应用中做定制来实现的。

TMF 的逻辑是，一次请求来到交易，首先识别业务身份，搞明白是谁家的业务，然后在业务流程运行过程中，相应的业务能力扩展点被执行，达到差异化定制的目的。业务身份的识别标准往往会是商品、类目或者其上的一些属性。例如大麦的业务身份，对应的识别标准是，商品上有一个标。

基于 TMF 来定制，以交易下单服务 buy2 为例，我们需要实现一个扩展包，其中的主要内容是：

- 业务身份识别：大麦基于商品标来识别
- 扩展点定制：例如大麦定制了资金流走即时到账

buy2 的大概结构是：

入口服务->业务流程（BPM）->业务活动->业务能力->能力扩展点。

大麦针对 buy2 的扩展包，就是针对上述能力扩展点的实现，请求经过业务身份识别确定是大麦业务后，大麦会走到这些相关的扩展点实现。

这里其实会埋下一个伏笔，就是在商品查询出来之前，其实是不知道业务身份的。这个会影响后面监控、问题排查等方面的一些做法。

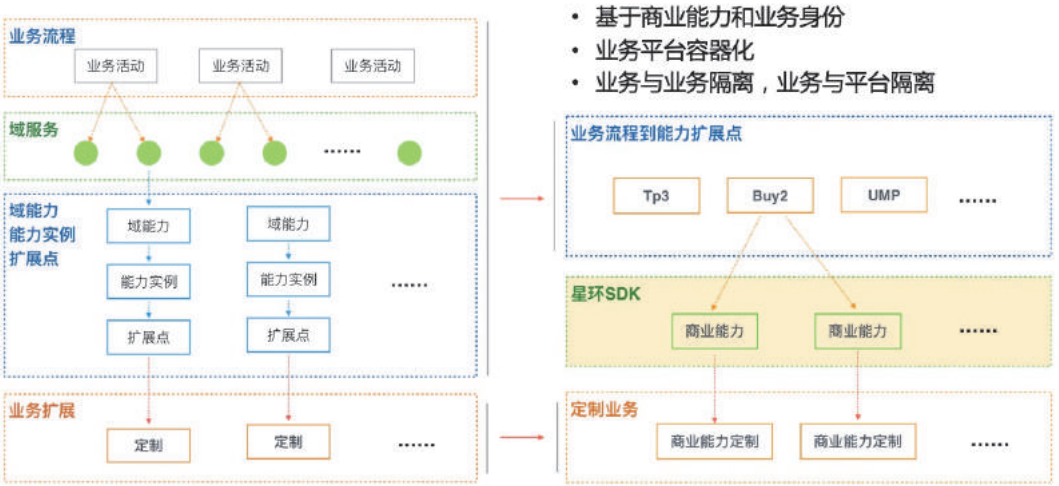
2. 星环与业务定制

星环是个很大的东西，涵盖了需求沟通机制、业务定制、容器化、质量保障多个方面。全面的知识还是要从星环的文档和与星环团队的沟通中获得。这里只是我们作为大麦一个业务方在使用过程中看到的一些侧面。

TMF 框架本身的开放性，已经能够支撑业务方在共享的体系下实现自己的定制，对想要借力共享体系，但是又有一定差异的业务来说，是非常好的。而后面的星环继承和发展了这种开放性，在我看来主要针对以下几个问题：

- TMF 阶段，业务方需要给各个共享应用分别实现定制包
- 因为上述原因，业务需要清楚的知道共享体系的内部结构
- 整体运行在一套基础资源上面，不同业务互相之间影响产生的稳定性隐患

星环架构师毗卢讲市面上复杂的 SAAS 系统，背后的物理结构是非常复杂的，而使用者要定制一些逻辑的时候，是不需要理解其背后的物理结构的。星环在扩展方面也肩负着向这个方面演化的使命。在 TMF 时代电商体系每个应用都有一套扩展点，每个业务方需要的话都要为多个共享应用实现扩展包。星环抽象了一个叫做“商业能力”的层次出来，统一全部业务能力的扩展点，最终实现定制方基于一套扩展点来定制一个包，部署到共享各个系统中这样的效果。



具体来说，例如“基础交易”是一个商业能力，他的定制点，包含了商品、交易多个方面，很多扩展点。涉及多个共享应用。商业能力扩展点这一层，包含了上述多个方面的扩展点，业务方通过这个统一的层次提供的语义来扩展，而在其背后，星环的实现层会把原始的业务能力的扩展点，桥接到星环商业能力的扩展点上来。

三、容器化与发布流程

星环里面一个重要的事情是容器化。按照星环网站上的说法，目标是四个：

- 业务自治：业务方独自完成业务的设计开发、测试和部署运行
- 业务隔离：业务与&业务/业务&平台隔离再不担心业务变更互相影响
- 业务动态部署：业务方可以在线进行发布和部署，快速响应业务需求
- 业务快速恢复：业务应用发布上线平台保留历史版本，可在线版本回退

这个事情涉及从系统逻辑结构，到发布方式的一系列变化。从这些变化里面可以了解到“容器”究竟是什么。

1. ClassLoader 隔离

阿里共享业务平台的应用以 buy2 为例，作为一个大容器，业务方例如大麦的定制包作为一个子应用，就是一个子容器。

每一个子容器都有自己独立的 ClassLoader。这个 classLoader 以平台的 ClassLoader 为

parentClassLoader，同时，自己内部基于配置知晓自己这个业务 APP 的 jar 包在哪里。

我们知道 Tomcat 作为一个容器是违背了双亲委派模型的，优先加载应用的类，从而实现 WEB-INF/lib 以及 WEB-INF/classes 类的优先级更高，而星环的容器没有这样做，依然是原本的双亲委派模型。这样做的结果是：

- 对于平台已经存在的类，会使用平台公共的实例
- 对于仅存在于业务 APP 包中的类，平台层找不到，所以会加载业务 APP 中的类

这个方式有优有劣。优点在于尽可能的共用了平台层面所能提供的类定义，可以控制 metaspace 的大小。在另外一个方向上，阿里中间件的 Pandora 隔离容器使用类似 tomcat 的机制以插件的 class 优先，从而做到各插件类路径隔离，从 2017 年开始要求 metaSpace 要配置 512M 了。而且，星环容器所承载的业务方子容器量会远超过 Pandora 所加载的插件量，如果不做这个控制，诸如 commons.lang 下面的类也是一个容器一份，MetaSpace 要的内存简直无法接受。

上述优势从另外一个角度看恰好又成了一个潜在的威胁。各方 APP 所以依赖的一些公共的三方包，甚至集团内提供的二方包，具体在系统启动起来之后使用的版本，是有互相影响的。两个 APP 使用了相同的三方包的场景下，实际上最终平台是按照 maven 的就近原则来打包其中一个到 war 中，于是实际过程中，一个 APP 升级该三方包，也就影响到了另外一个 APP。实际上导致容器隔离不彻底。当然，这只是一个权衡。实际上如果真完全以 Pandora 模式，又会出现平台 export 哪些类给到业务 APP 的细节问题。

2. 独立的 Spring 子容器

Spring 的容器也做了类似上述结构的父子关系，不用担心各个 APP 内的 bean 之间的冲突。一个结果是，子容器里面仅包含业务 APP 中所定义的 bean。那么，类似限流、降级这样的平台层自动代理的一些事情，在子容器里面是没有的。如果你的业务 APP 希望使用到限流、降级，那么你需要在业务 APP 的 spring 文件中引入相关的 bean 定义。

3. 自主热发布

共享应用由于结构、业务及其复杂，发布一次的时间，非常长，这是一个极其痛苦的过程。以 buy2 为例，一次构建几分钟，一次部署要 10 几分钟到半个小时的时间。相比于业务方自己的小应用一次构建加部署 5 分钟以内，这个时间简直长的要命。曾经看沈洵很久之前的 PPT，讲阿里做服务化这个事情，说服务化之前淘宝整体是一个大应用，构建部署一次够出去抽根烟，回来还没完事儿。服务化分拆应用之后，这个事情有了大大的改善。但是整个体系演化到今天，

昨日重现了。

基于 classLoader 隔离和 Spring 子容器，整个业务 APP 的热发布成为可能。所谓热发布就是在不停 tomcat 的情况下，把一个业务的变更发布到线上。一次热发布过程大约如下：

- 应用 offline 当然要首先让流量不再进来
- 下载更新应用的 jar 包，这个过程使用了目标应用的主干配置项来替换 jar 包中的引用
- 容器销毁-重建，过程中抛弃了原有的 classLoader 和 spring 子容器，创建了新的实例
- 应用 online

热发布大大缩短了业务构建部署的时间，基本上一次构建部署 5 分钟内搞定。相比之前每次 war 部署，幸福的像是回到了伊甸园。

实际上平台一次 war 发布经常需要几个小时的时间，这其中固然有分批的因素，但是本身应用构建、启动慢也是一个重要的原因。从稳定行方面来讲，线上发现一个问题，如果改代码、war 发布，这样的时间加起来几乎足以似的问题变成故障，小故障升级成大故障。有了热发布，腰不酸了，腿不疼了，轻轻松松搞定发布。线上问题发现到改动、恢复时间也大大缩短。
但是热发布的逐步完善过程还是很快的，最初有一些限制，后来都逐步优化掉了：

- 静态配置项变化，不能热发布。目前已经使用了对应变更的配置项得以解决
- 依赖包变化不能热发布。这个事情曾经想过用 fatjar 来解决，后来已经支持附属 jar 发布，大麦已经在使用
- 前端交互协议组件奥创的修改不能热发布。这个事情使用新奥创得以解决，大麦 19 年完成了切换

热发布上线后，大麦不能热发的场景，大多数是奥创改动和依赖包改动。现在正在解决依赖包的问题。

目前，业务定制改动后，发布的时候还是要勾选一下要发布的目标系统，例如 buy2。未来的方向应该是业务方不用关心改动发布到共享的什么系统里面去。

4. 独立分组与环境隔离

大麦在 buy2 上划分了独立分组。后面飞猪等业务方也都是用了独立分组。星环目前采用中间件提供的环境隔离方案把业务方的流量隔离到目标独立分组。

首先 buy2 给大麦划分了一个独立机器分组叫做 buy2_damai_host。

然后，大麦的流量通过三种方式路由到大麦独立分组：

- nginx 路由，部分流量在公共分组的 nginx 层可以识别到 url 参数或者 Header，然后走到 buy2_damai_host 的 Java
- Java 兜底路由，部分流量只能在 buy2host 这个机器分组里面把商品查出来之后，才能识别到是大麦的业务，此时再通过一次 RPC 调用，走到大麦分组
- RPC 调用路由，大麦自己的 Java 应用在调用 buy2 的 RPC 服务之前，可以通过设置路由相关的鹰眼标来指定调用到大麦分组

大麦最初采用独立分组是基于物理隔离和降低运维成本的考虑。发展到现在，独立分组这个事情主要看到以下几个点（包括优势和问题）：

优势一：独立分组让大麦只关心自己分组的机器，在遇到某些问题的时候，处理起来比较方便。共享应用动辄几千台机器，查个问题需要好久。但是，独立分组大麦也就 100 多个机器，查问题成本完全不是一个水位。

优势二：独立分组让大麦的监控可以只关心大麦分组，这个事情详细在下一小节来描述。

优势三：独立分组带来的物理隔离，让大麦在自己的分组上开启一些特殊的能力不用担心影响别的业务方，具体参考下面的问题排查小节。

优势四：独立分组在平台整体 war 发布的时候，可以自主控制发布节奏。一般来说平台整体的发布因为应用众多，所以分批较多，而且间隔也经常拉的很长，经常跨天。一些需要快速生效的问题修复场景，这个过程慢的让人捉急。独立分组对应在应用发布平台上是一个独立的环境，发布单也是独立的，因而可以在保障稳定的前提下继续发布，提高发布效率。

同时独立分组也带来一些困扰，这也是目前大麦和共享同学还在想办法优化的一些事情：

一方面，共享的应用都是单元化的，这个独立分组也要保障多个单元的容量，在单元化流量调整的一些场景下，需要密切的做好沟通，避免特定单元容量不足引发问题。

另外一方面，独立分组还是带来了一定的资源浪费。大麦业务的流量有个特点是大型抢票来的时候（每周几次）流量瞬间很高，而无大型抢票的时候流量低的不好意思说。为了维持大抢容量，独立分组必须有足够的资源配置。

目前大麦仅在 buy2 上做了独立分组，其他几个应用都还没有做。飞猪和新零售做的多一些。

四、监控

监控部分，说三个方面：

1. 系统监控

系统监控说的是 cpu、load、网络这些方面。对于做了独立分组的应用，可以在集团监控平台上针对这个分组配置一个虚拟应用，看起来和其他应用没什么区别，看起来很方便。如果没有独立分组，那么系统监控这个事情，基本上就只能是平台层面关注了。

2. 业务监控

共享的主要应用都会有统一的业务监控以及对应的大盘。但是这些大盘的视角都是平台视角，大麦这样一个小业务放在里面，典型的效果就是下单全部失败，成功率曲线也看不出来明显的下跌。

所以必须做业务线视角的监控和大盘。因而大麦参考共享的监控大盘配置了自己的业务监控。包括渲染量、下单量以及相关的成功率、错误码等方面。

这里面有一个日志数据筛选的事情也经历了一些变化过程。早期，大麦是在日志里面过滤有大麦业务身份的数据，然后基于过滤出的日志内容做监控。后来发现，对于走到业务身份识别逻辑之前就报错了的请求，在这些日志里面是不可能有业务身份数据的。所以就修改成了拿 buy2_damai_host 上的所有数据来做监控。正好环境隔离方案上线后，大麦分组不再会有别的业务的流量，所以这样做是完备的。

可以看出来，上述逻辑严重依赖独立分组。没有独立分组的业务或者应用，会存在一个问题，代码走到业务身份识别之前出现的问题，是很难做到业务方独立的监控里面的。

另外一个实践是，对于大麦这样一个小业务，平时的请求量非常低。以渲染为例，平时 1 分钟也就几十个请求。渲染失败的原因里面，类似库存不足、超过限购数量、相同身份证已经购买过之类的这样的原因，如果都作为请求失败来看的话，成功率的曲线会很难看。基本上跌的很多，真正的线上问题会被淹没。所以大麦在 GOC 监控里面，会把类似库存不足这样的错误码排除出去，从而做到一个基本上维持 100%成功率的曲线。

这样做能够达到曲线跌了一般来说就是真出事儿了。这不是一个完备的方案，但是实践下来证明是有效的。

3. 异常监控

一个应用的异常堆栈日志是应用是否正常的重要表征。有新的异常出现或者某类异常大量飙升的时候，基本上可以判断有问题了。一个好的实践是，系统异常打堆栈，业务异常（例如库存不足）不打堆栈，这样能够让应用的异常量维持一个比较低的水平。大麦技术团队经常会使用鹰眼团队提供的监控产品来看共享应用的异常信息。根据某一类型的异常突增这样的信息来分析判断问题。

五、问题排查工具

技术团队的工作里面，线上杂七杂八的问题的排查工作，一直以来都是重要的组成部分，而通过工具建设来提升问题排查效率也是技术团队一直追求的事情。这个方面大麦以往有一些实践，融入共享之后也在共享的体系下共建了一些能力，这里分享出来。

这里描述的“问题”，一般是指两个方面，一个是用户或者客服、业务同学反馈过来的一些问题场景。例如某某用户具有购买资格但是下单提示不能购买，或者某某订单没有用到一个优惠。另外一个方面是业务监控发现的，例如某一类错误码突增。

问题排查过程要基于已知的这些少量的信息，去追踪还原当时的场景。典型的想法是去机器上找日志。但是这种方式效率比较低，且会把这个工作局限在技术人员身上，无法给到上层。

阿里的鹰眼的业务日志是一个很好的工具，现在大家也经常可以拿着订单号、traceId 来查询一些链路上的日志。但是会有一些局限：一来，没有订单号的场景，例如订单确认页打开失败，这种无从查起；二来，很多时候是没有 traceId 的……

大麦和共享协同建立的天秤业务日志查询工具，能够做到基于自定义的索引来打日志、查询。目前大麦和飞猪用的挺好。

1. 效果

我们做到可以根据一个错误码，查询这个错误码产生的链路日志（利器），根据某一次请求失败的日志详情来判断问题的具体原因。可供查询的日志数据里面实际上包含了 3 个部分的日志：

1）交易下单应用的入口服务和依赖服务的拦截日志，

2）交易下单应用的入口异常日志

3）业务扩展包中自主打的一些日志

业务 APP 可以自主打印一些指定格式的日志。日志内容、索引都是自定义的。配置采集之后即可在一个统一的页面上查询。

2. 怎么实现的

上述日志最终都是通过一个通用的二方包打到磁盘上，然后通过日志采集服务采集投递到了阿里云的日志服务 SLS 里面（需要各业务方自己提供 LogStore，并承担成本）。然后基于 SLS 的能力提供查询。

3. 更进一步

大麦基于上述日志，做了一层格式化和语义转换，把技术人员才能看懂的信息，转换成为非技术同学也能看懂的说法，从而把一部分问题的排查交给了测试乃至业务同学。

六、尾声

大麦通过把整个电商融入阿里共享电商平台，稳定性、性能上都得到了大幅的提升。在产品能力上也自然而然的可以低成本的借助共享的各种基础设施，在这个基础上，大麦的业务也在逐步尝试与淘系共舞。过程中有很多抉择、取舍，这篇文章也是希望从技术过程的角度尝试对一个小业务如何融入大平台做一个分享，希望有类似场景读者，不论是做平台还是做业务，能够有所收获。

高并发大流量，大麦抢票的技术涅槃之路

作者| 阿里文娱测试开发专家 舒琴、阿里文娱测试开发 锦烨

一、大麦抢票背景

大麦网主要的业务范围为演唱会、音乐会、体育赛事、话剧、展销会、亲子活动等现场类的票务业务，其业务链条涵盖从 B 端生产、C 端销售、现场换验的全套流程。大麦网一类典型项目是稀缺的火爆 IP 项目，如演唱会、游戏体育赛事，这类票务隐含了时间、空间的特殊限制属性，是需要抢的。大麦抢票是演出行业的双 11，涉及场景复杂、系统较多、链路较长，抢票保障尤为重要。

大麦抢票保障大致经历了几个阶段：

第一阶段：“原始”阶段，保障不健全，设施不完善；

第二阶段：“弹内化”阶段，部分大型抢票顺利完成；

第三阶段：“体系化”阶段，能够承接所有大型抢票；

第四阶段：“常态化”阶段，大抢体验优化升级。

二、“原始”阶段

大抢容易出现限流，体验不顺畅等现象，囧！

1. 为何会这样

此阶段的大麦还处在原技术团队和阿里系业务、产品、技术各方都在讨论及对焦阶段，大部分大型抢票核心系统还在大麦的 IDC 机房，技术体系走.net 和 java 的混合体系，大麦原体系主要承载此阶段的大型抢票任务。

为什么此体系下，每逢大型抢票就会面临如此大的压力和风险呢？分析主要有以下几点原因：

1）保障设施不健全：大麦 IDC 机房硬件设备、带宽等均有限制；DB 采用 SQL SERVER 企业版，很多数据库都是单库。在应对大型抢票时（特别是选座购买，耗带宽、耗资源），会面临严峻挑战；

2）预案/限流待建设：系统在高流量、高压力下的保护措施待建设，比如：限流和降级，造成系统一旦超过压力，就直接报警；

3）监控运维零散：定位问题、解决问题耗时较长。

2. 方案和结果

这阶段也采取了一些临时方案，比如：极简限流方案、一些点的性能优化、修改应用配置参数、整理抢票预案等等，也缓解了一些问题，但整体上仍未解决大型抢票问题。

三、“弹内化”阶段

基于第一阶段的诸多问题，产品、技术已经启动大面积系统改造，直接重构新系统到阿里域内，用新系统建设逐步替代老系统，即空中换引擎方案。

1. 空中换引擎

要快速将关键系统重构到阿里域内，确立了直接迁移、临时方案或长期重构等步骤，具体方案是：

1）APP 链路部分弹内化：技术改造重点放在无线端，所有 APP 用户调用接口的入口先走到阿里域，再路由到大麦 IDC，让阿里机房来抵档大量流量；

2）借助阿里基础运维能力：由于入口接口入到阿里域，一些限流及降级的事情利用平台就可以做，运维监控也完全可以利用 eagleeye、maieye 等排查工具来做了；此过程中

用户中心、消息中心等服务开始向阿里域内重构并上线；

3）抢票预案初步建立：在主站的预案平台建立了大麦的大型抢票预案，比如：商品详情页增加了 tair 缓存，靠 tair 在阿里域内扛住流量，减少打到大麦 IDC 的请求调用；

2. 坚定路线不动摇

优化后的热量抢票项目系统正常，但限流会影响用户体验。分析抢票过程中出现的问题，集中在了弹内化范围、机房的瓶颈、运维经验不足等。所以，催生了第三阶段，大型抢票全链路收口进阿里域内。

四、“体系化”阶段

针对上阶段遗留的问题，业务和技术针对抢票流程和系统做了全体系化升级，确立了完善的抢票流程和抢票保障机制。升级后的大麦能承接住所有大型抢票，且用户体验有所提升。

1. 弹内化全面开花

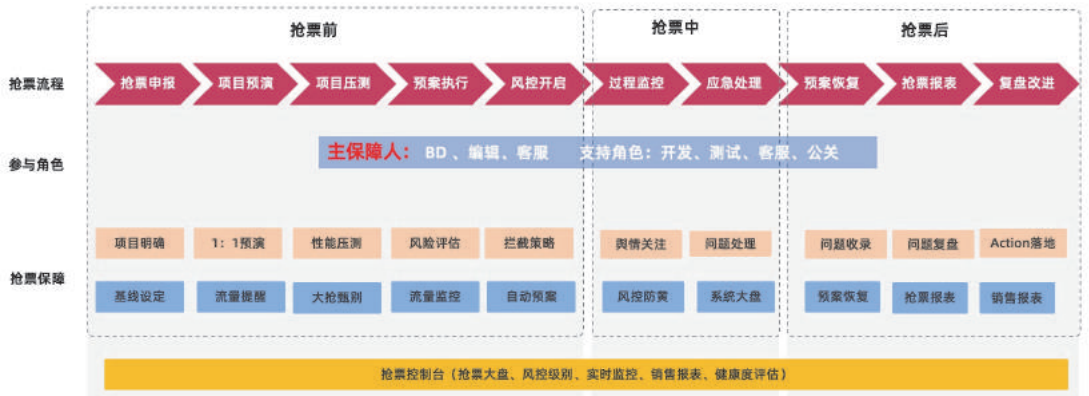
1）新搜索在阿里域内上线，搜索 response 过大的问题也都解决，不存在对带宽的影响了；

2）新选座在阿里域内上线，大型抢票的选座流量直接打到阿里域内，利用异步化和类似 ConcurrentHashMap 的机制平衡了对大麦 IDC 选座的调用量及缓存的一致性；

3）新交易在阿里域内上线，将核心的创建事务号接口、下单接口全部都放在了阿里域内，单下单后订单要同步到大麦机房的进行后续履约服务；

2. 保障流程建起来

基于抢票活动业务方和技术方信息不一致、或临时抢票活动准备不充分等情况，由测试方牵头和各业务方、各技术方针对抢票流程、参与人员、抢票设置各项达成一致，并沉淀出抢票保障流程和方案，相关人为操作建立 SOP 保障，优化后的流程为：



1）【流程建设】抢票阶段分为抢票前、抢票中、抢票后：

抢票前重点是由业务方抢票申报，再由技术方确认是否安排预演或压测，根据业务方和历史抢票信息判断抢票级别来决定抢票预案执行范围和风控级别；

抢票中重点是过程监控和应急处理；

抢票后重点是预案恢复、抢票报表输出，以及抢票过程中问题复盘；

2）【流程建设】抢票参与角色中：产品、开发、测试、公关

抢票涉及多个业务方，主要业务保障人是 BD、编辑、客服，主要支持角色有开发、测试、客服、公关等相关人员；

抢票过程中相关角色各司其职，共同保障抢票。印象比较深的就是这个阶段每逢大抢，一屋子相关抢票人员聚集，在抢票顺利完成后，会议室传出的阵阵欢呼声！

3. 预案/预演/容量/Action 专项

抢票保障的每一项操作都需要在业务方明确项目信息后，由测试牵头拉各方参与人员整体评估和协调执行，不管是抢票前的准备还是抢票后的复盘，每个小的专项都执行到位。

1）【质量保障】项目预演：一般已有成熟流程的项目不会再安排预演，对大型抢票或未抢过的新玩法，测试方会安排模拟抢票。主要由各业务方、技术方和各端测试同学共同参与，提前暴露业务、设置问题或体验问题；

2）【质量保障】性能容量：技术拉取全链路最近类似项目的最新压测数据，和线上实际容量做评估，分析抢票预估量是否能顺利支撑，是否有性能瓶颈或限流情况，提前通知业务方；

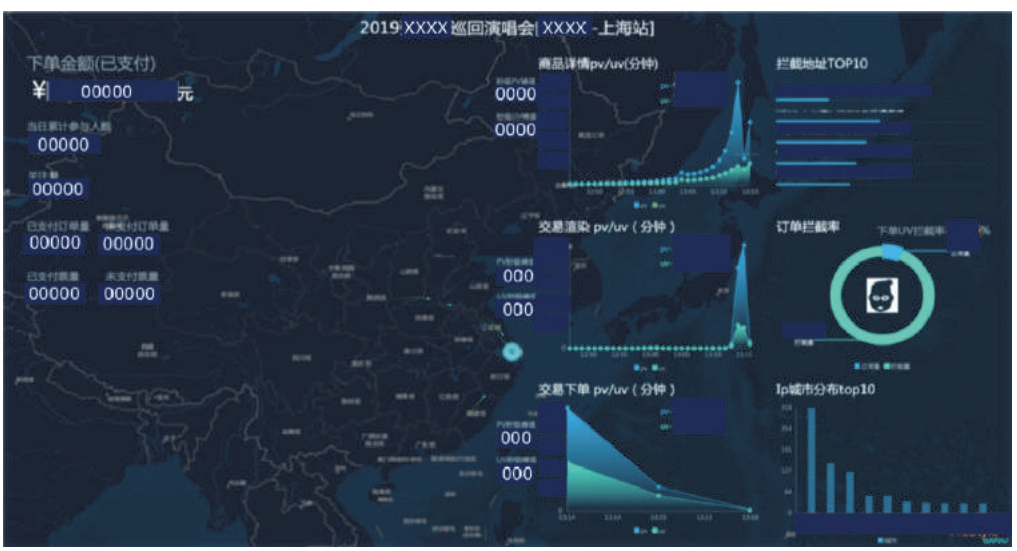
如项目玩法没有最近的压测数据支撑，由测试方安排大抢项目压测；

3）【技术优化】预案执行：全链路涉及的首页搜索、商品详情、票务云选座、交易下单、票务云库存、订单服务、无线端等梳理大抢模式的前置预案和紧急预案。如商品详情前置预案：对艺人、关注、营销等降级处理，调整用户、场馆、详情缓存时长，抢票前 30 分钟预热限购服务的 BD 和 tair 等；

4）【质量保障】问题复盘：每次抢票完成后对抢票过程中各方暴露的问题、客服反馈的高咨询、线上收集的 bug、内网帖子吐糟、外部用户微博或微信转载等问题，测试方统一收集，组织各方复盘后优化方案落实到 action，并跟进 action 执行进度；

4. 抢票监控大盘

除各业务定制的抢票监控项外，抢票期大盘的汇总数据监控，可以为每次抢票更好地提供监控数据支持，方便业务方一目了然 get 到抢票数据，具体信息如下：



五、“常态化”阶段

上阶段系统化升级完善后，大家肯定会问，大麦抢票不会出问题了吧？

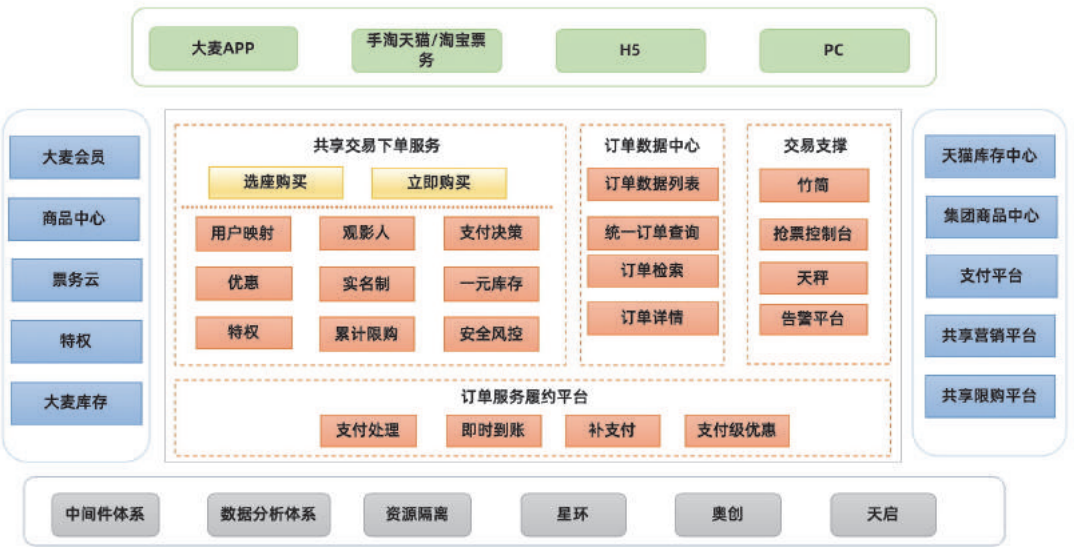
我们可以很负责地说肯定再不会出现宕机情况，但是在近期超稀缺 IP 抢票因为项目太火爆，抢到票的毕竟是少数人，大抢之后出现的二手高价售卖现象、抢票不流畅导致抢不到票等

问题，被抢不到票的内网小二、外网用户疯狂吐槽。

总结槽点主要集中在抢不到票、商品详情被限流、下单交互耗时导致抢不到票、抢票异常情况对用户提示不友好等等，针对这些问题除了产品方案的优化，技术同学也成立了很多抢票专项解决用户痛点，这些小而美的体验极致优化，你注意到了吗？

1. 为真实用户护航

此阶段磨砺了近一年的大麦新交易系统上线了，新交易和共享星环平台全面融合，核心的渲染接口、下单接口基于星环能力实现了大麦扩展特性；新交易架构图如下：



融入共享对大麦来说好处很多，针对抢票流程的贡献主要有 3 点：

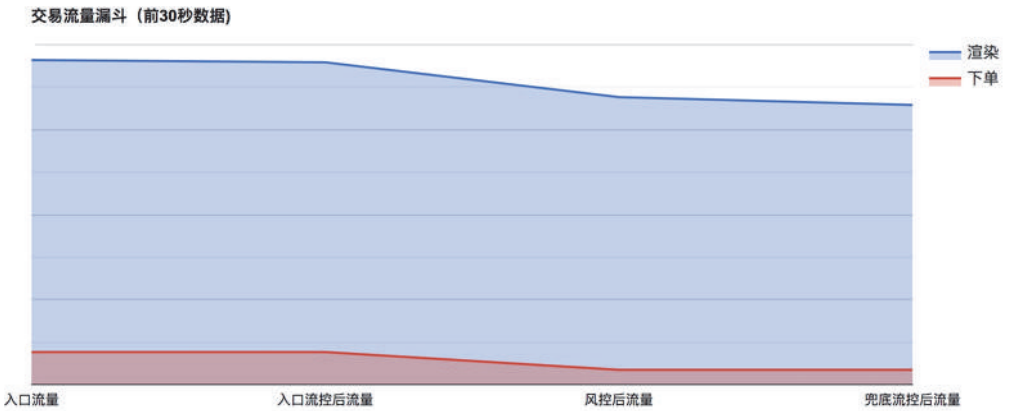
1）【技术优化】依托共享基础能力

除了可以复用共享能力，还可以参考主站交易的大抢方案，比如限流、系统日志监控、问题排查平台等等。技术方实现基于星环体系的未支付关单定制，由业务方指定关单时长，有效降低了恶意占用库存现象发生。

2）【体验升级】接入集团风控体系：

服务于线上火爆抢票的三层防控技术体系，实际由大麦用户风险评分、集团安全 MTEE 人机识别、定制化的策略包组成。在交易流程的渲染、下单以不同维度对非法用户进行二次拦截，

让真实用户的抢票体验更顺滑，大大提升了真实用户的购买率。



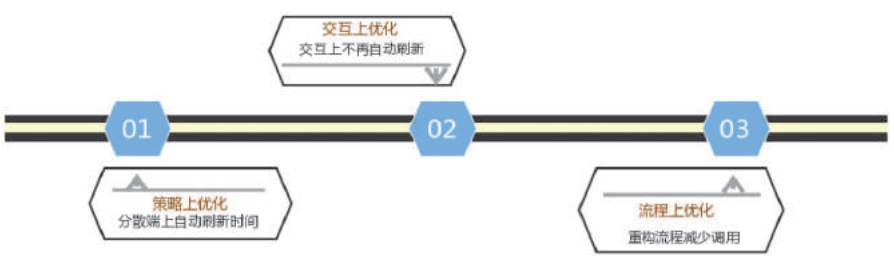
2. 核心链路模型优化

1）【体验升级】商品详情无限流：从以往抢票看，商品详情页每逢大抢不仅要借机器还要限流，成为了用户槽点聚集地。商品详情主要实现了流量分散策略：

a）策略上减少开抢前并发请求，由于散列控制在较短时间，能够快速上线快速验证，但效果不明显；

b）交互上倒计时结束后用户点击替代自动刷新来分散流量，效果明显

c）流程上减少物理调用，倒计时用户点击购买时只调用二级页，倒计时校验交给二级页，效果非常明显



改造后详情页与弹层页流量从巨大差距到基本持平再到流量较大反转，商品详情抢票再也不用借机器，且足以支撑目标最高热门项目抢票，再不会触发限流！

2）【体验升级】交易下单扩入口： 从以往抢票看，交易下单大流量时容易触发限流，用户

反馈抢不到票。针对交易下单的主要策略是：

a）放大入口+风控拦截+兜底限流，首先让真实用户能够进到交易，再通过风控过滤掉异常用户，最后用兜底限流保护下游系统，从而最大限度的保障用户抢票顺滑；

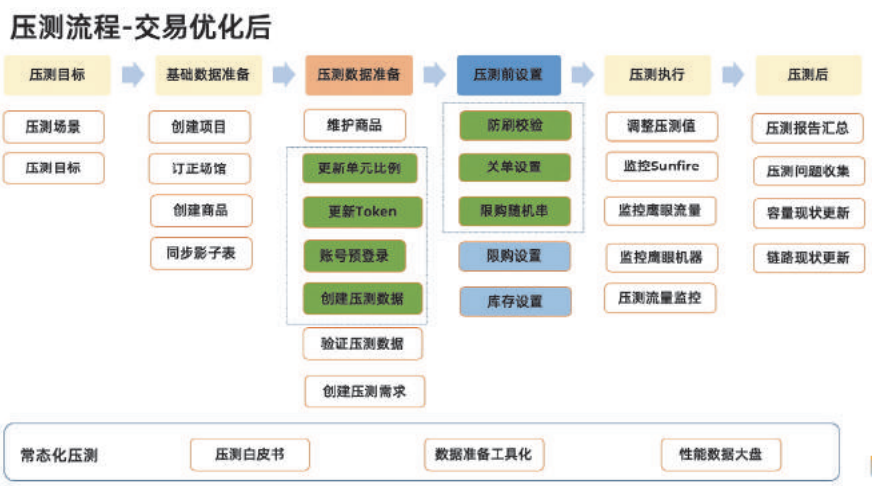
b）对渲染阶段的支付方式和支付特权做优化，实现大抢时渲染对支付方式调用降级，降低用户渲染或异步渲染被流控的风险；

3. 性能常态化

为摸清交易链路最新性能水位，测试团队启动了性能常态化项目。每月定时执行压测，并结合当月各系统功能发布、性能优化或上下游支持，评估具体的压测场景和压测目标，压测完毕后更新链路依赖现状，为抢票提供最有效数据。

1）【质量保障】常态化执行：如近期的稀缺 IP 项目抢票再次刷新各榜单列表，开发和测试一起根据现有流量重新对系统进行压测摸高评估。

2）【质量保障】压测自动化：测试梳理了压测流程，沉淀了各业务线压测白皮书。提炼出过程中耗时较多的步骤，实现自动化执行，如原临时项目生成压测数据和压测前设置需要 11-12 步耗时半天，现在常态化固定项目生成压测数据和仅需要 3-4 步 3 分钟即可。

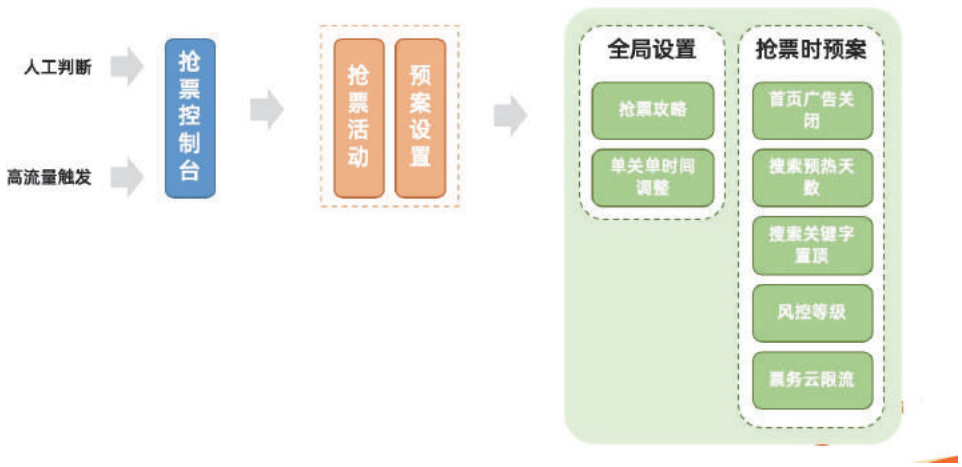


4. 预案自动化

之前抢票存在抢票活动频繁，抢票无统一规范流程（监控、预案、入口），支持人数多，组织抢票会议，耗费人力等问题。

1）【技术优化】抢票控制台：

使 BD 或者运营在【抢票开始前】可以设置一些预案，【抢票过程中】提供统一视图对抢票进行【实时监控】，并且有能力进行【人为的干预和控制】，在【抢票结束后】能够提供历次抢票数据以供分析，从而帮助 BD 自助完成抢票，实现“无人值守”具体实现流程如下：

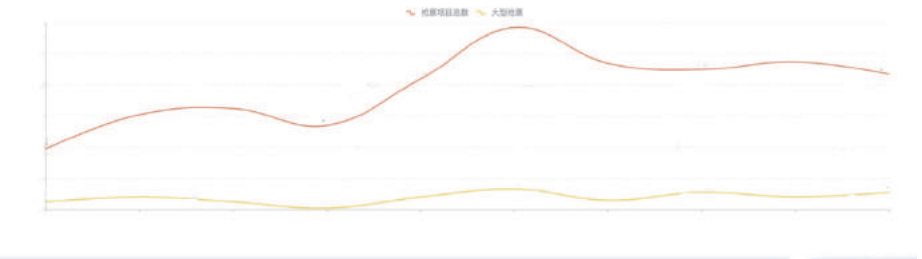


2）【技术优化】商品详情预案：

详情页每次大抢都需要人工降级 6~10 项预案，各项预案设置的值、执行时间等各有差异，在预案设置时还需根据个人经验做评估调整，人工操作太繁琐。详情预案基础策略是对原降级项实现本地缓存或 tair 缓存，第三方依赖接口限流异常降级补充，优化后仅保留 1 项需手动配置的预案，其他皆已实现自动化。

5. 常态化成果

通过常态化各专项保障取得明显成效，近一年支持的抢票项目近千场，其中大型抢票近百场。其中近期热门「超大抢」项目，商详/下单渲染/下单均创历史峰值，系统顺利承接；热门「大抢」项目，特权码选座和普通选座，特权及选座峰值均创新高，系统顺利承接；



六、结语

大麦抢票经历了“原始”阶段、流程建设、技术优化、专项保障等四个阶段建设后系统已稳如磐石，对提升用户体验上也在不断努力。当然可能还是存在或多或少的问题，目前的技术方案也可能不是最优的，比如项目热度智能分析、风控自动调节等等，也已经在技术优化计划中，在抢票的路上大麦会秉持初心一直奔跑！

3

现场平台

基于云原生的边缘计算在大麦现场的探索应用

作者| 阿里文娱技术专家　草薰

一、背景

近年来，我国文化产业蓬勃发展，文化产业价值年均增速远高于同期 GDP 增速，尽管中国演出市场在开放竞争中逐步规范有序，但目前仍处于起步和培育阶段，尚不够完善和成熟。尤其在演出场馆基础设施、管理运营等方面参差不齐。一方面，国内大规模兴建剧场，但是却不重视开展剧场相关管理和服务标准的建立，造成目前中国的剧场硬件设施世界一流，管理与服务却远落后于发达国家。另一方面，随着现场演出形式的变化，兴起了许多独特的演出场地。当前，对场馆的精细化、分层运营已经成为促进现场服务升级的核心抓手。

二、痛点

演出场馆的基础设施、热门 IP 等诸多因素都可能导致演出现场出现无网或弱网问题，在这种情况下如何保障核验、摄像头监控等现场服务可用是技术首先要解决的问题。

面向 AI 与 IoT 结合为代表的未来发展方向，通过收集海量数据形成的消费者/用户画像可以使终端设备更加智能、更为“主动”地为消费者提供个性化服务。需要对现场类型众多、形态不一的智能设备形成通信互联、数据共享、能力互通的“联合体”，为用户提供智能场馆服务。如何保证海量数据处理的时效性和有效性，是面向业务发展要解决的问题。

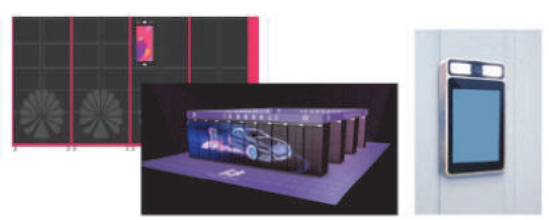


图 1 大麦现场智能终端

目前，在无网和弱网的现场环境下提供现场服务是通过单机应用部署的方式，面临的问题如下：

1. 为保障单机应用服务可用、可靠需要在本地搭建集群；

2. 同样的业务逻辑，本地和云端研发体系不统一，导致架构冗余、研发效率低、发布周期长，降低了业务的迭代效率。

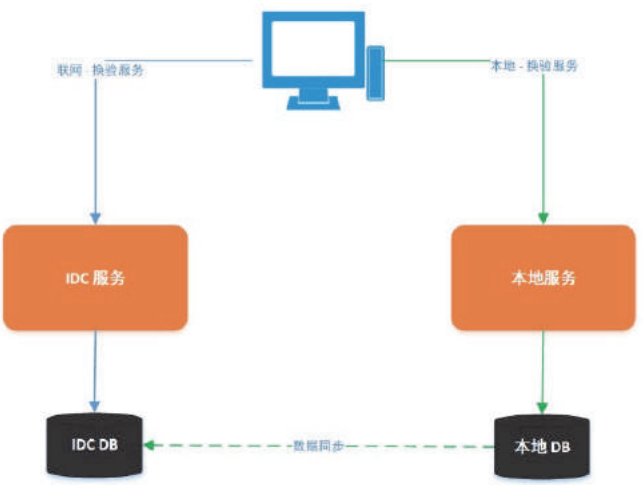


图 2 大麦现场本地/云端验票模式

①IDC 服务和本地服务代码级复用低；

②本地服务单机部署，无高可用性保障；

③本地服务依赖本地数据库，跟 IDC 数据库结构可能不一致。

针对新型的场馆的智能服务需求，譬如 LIVEHOUSE，基于现场观演人群定向提供个性化服务，现有的计算架构在数据时效性和有效性方面不足以支撑。

三、解决方案

基于上述的痛点，现场的整体解决方案，从有效支持 12 个月内的业务和运营的发展，具有 6 个月内创造和技术领先的可能性，及加速研发同学专业化能力的聚焦沉淀和创新孵化，降低体力劳动强度。

1. 边缘计算

边缘计算是网络中最靠近物或数据源头融合网络、计算、存储、应用核心能力的分布式开放平台，就近提供边缘智能服务。在更靠近终端的网络边缘上提供服务是边缘计算最大的特点。在数据处理的时效性与有效性方面成为云计算的有力补充。对于这样的设计，能满足大麦现场业务在智能核验、设备管控、业务监控等方面的关键需求。

图 3 边缘计算

阿里云边缘云原生的产品化落地——ACK@Edge（云边一体的云原生标准 k8s 托管服务），支持海量边缘网关节点接入，深度融合 IoT 云端市场、云端 FaaS、消息、运维等服务，提供针对边缘实例的 DevOps 能力，极大的提升边缘的应用分发及部署运维的效率。

为了**让现场更简单、业务更高效**，需要将原来单机应用+ IDC 的集中式部署架构，演进到云+边缘计算的统一架构，这就需要我们能够有一种方式来统一管理 IDC 应用服务器 和现场众多的边缘节点。通过边缘 k8s 架构，统一管理云与边缘的节点，实现了应用发布和弹性扩缩容的统一。通过弹性能力节省机器成本；现场终端就近访问边缘节点，解决现场弱网/无网、现场视频监控端到端的网络延迟等诸多问题。

2. 云边端协同

云计算与边缘计算是现场数字化转型的两大重要支持。云计算与边缘计算在技术定义上具有差别，使他们所使用的场景有所差异。云计算适用非实时、长周期数据、业务决策场景，边缘计算在实时性、短周期数据、本地决策等场景方面表现更为优异。因而，娱乐现场数字化转型需要云计算与边缘计算的支持。

阿里云在两大边缘计算领域 CDN 和 IoT 深耕已久，边缘规模和业务复杂度的日益攀升带来了不小的效能问题。阿里云容器服务和 IoT 业务团队联合推出 Kubernetes Edge 定制版本，作为云上 IoT 边缘托管服务底座，支持海量边缘网关节点接入，深度融合 IoT 云端市场、云端 FaaS、消息、运维等服务。在这个过程中在这个过程中，云原生让 IoT 托管如虎添翼。在大麦现场业务中尝试使用，如图 2 所示 云边端分层结构也日益显现。

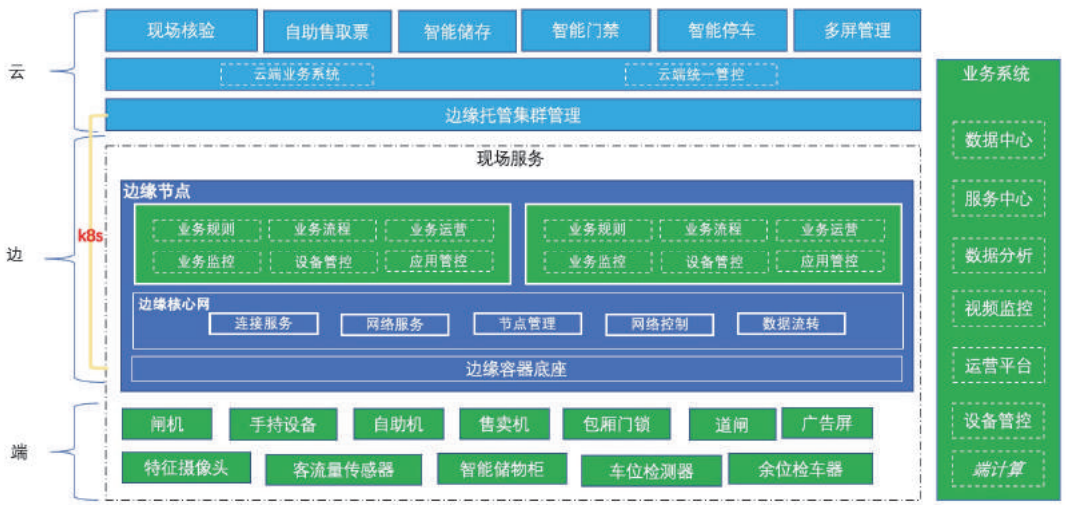


图 4 娱乐现场服务 云-边-端架构逻辑图

①本地和云端应用统一容器化部署运行环境；

②适配不同的网络环境，终端灵活选择本地或云端服务；

③边缘节点优先处理过程数据计算，核心业务数据同步云端。

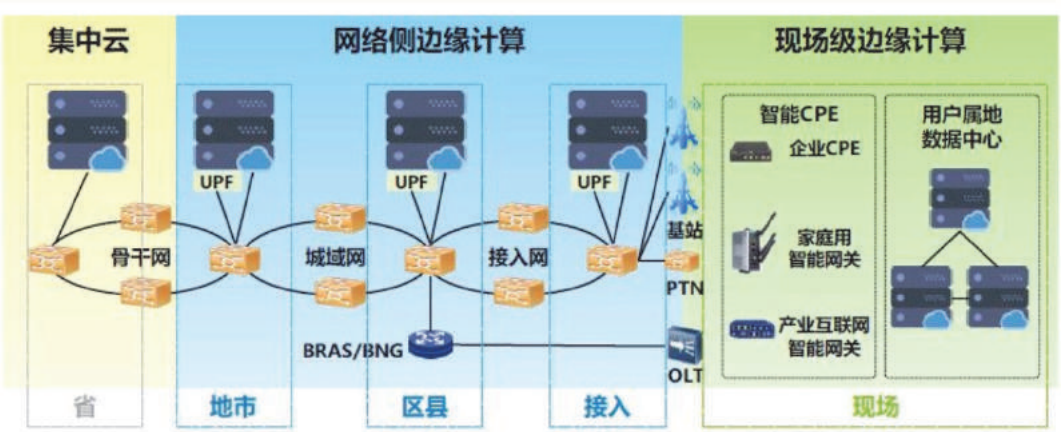


图 5 现场娱乐云-边-端部署图

四、未来

随着 5G 技术的广泛应用、云计算与边缘计算在数字化转型的支持，加快了数字技术在文娱行业现场服务领域的应用。譬如无纸化、人脸识别等智能核验、智能泊车、客群识别等技术逐渐成为智慧娱乐现场新标配。在用户需求不断提升及产业服务不断完善的背景下，未来以安全可靠、优质体验和全链路服务建立的服务平台构筑大麦的行业壁垒。

大麦人脸识别系统，如何支撑马拉松赛事？

作者| 阿里文娱技术专家 墨贤

大麦的人脸闸机在 2019 年杭州马拉松上成功的完成了刷脸入场功能的首秀，相比传统的马拉松入场核验方案在入场体验和入场效率上都有了很大的提升，下面介绍一下大麦的人脸识别是如何支持马拉松赛事的。

一、马拉松赛事流程介绍

马拉松赛事的流程主要分三步：

第一步，参赛者到马拉松官方网站报名，报名成功后会通知选手；

第二步，报名成功的选手需携带身份证到官方指定的地点领取装备；

第三步，比赛当天携带号码簿验票入场进行比赛。

我们面临的挑战是：

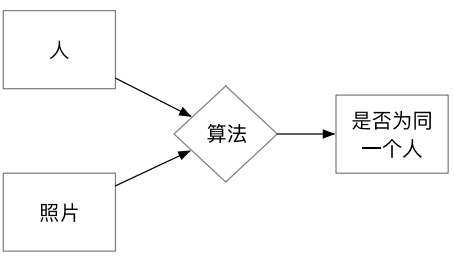
1）如何在指定时间内完成参赛者的装备领取工作：既要保证快速领取不造成人员积压，又要保证装备不会领错；

2）如何保证比赛当天在短时间内完成几万人的入场核验工作。

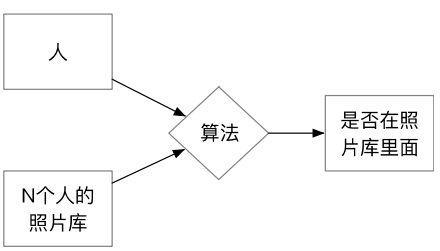
二、大麦人脸识别解决方案

在介绍大麦的人脸识别方案之前，首先介绍人脸比对的几个常用术语：

1 比 1：1 比 1 是指用照片跟人进行比对，通过算法判定照片和人是否是同一个人，简单理解就是证明“你就是你”。



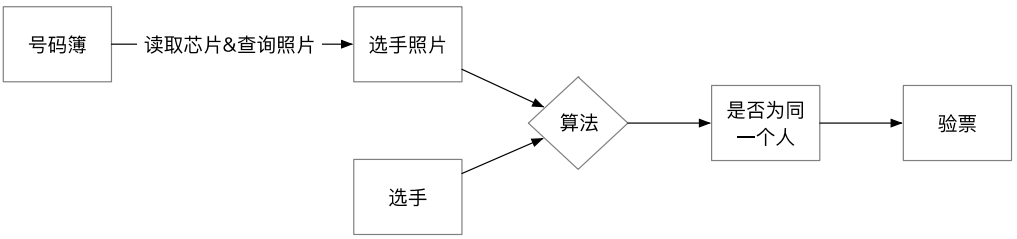
1 比 N：1 比 N 是指在 N 个人的照片库里（底库）进行查找，通过算法判定这个人是否在这些照片里面，通俗来讲就是“我是谁”。



清楚了马拉松赛事的流程之后，下面介绍大麦是如何支持现场领取装备和比赛当天入场核验的，在我们的方案里面比赛当天的入场核验是压力最大，风险最高的，而且这个也是依赖前两步的。

选手入场比赛时不会携带手机和证件，那如何进行验票呢？

大麦传统做法：提前跟主办沟通，在号码簿中放置一个射频芯片，芯片号与号码簿上的号码一一对应，这样通过验票设备扫描到芯片后会查询到这个选手的参赛信息，包括照片（来源于第 1 步和第 2 步），为什么需要照片？只有拿到照片然后和本人进行比对才能确认是不是本人还是替跑，这就用到了人脸识别，也就是上面所说的 1 比 1 比对。



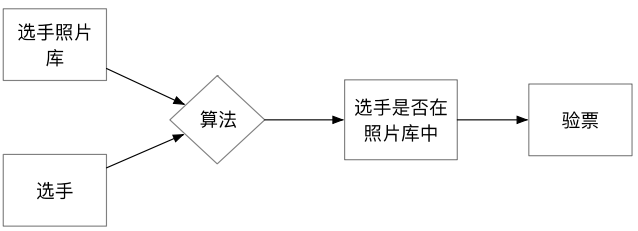
这个方案是可行的，但面临两个问题：

1）号码簿是第三方公司负责的，经常会遇到号码簿里没有芯片或者芯片号跟号码簿上的号码不一致（实际发生率还比较高），这就会造成选手无法直接核验入场，需要人工处理；

2）这种入场方案需要先核验芯片，再进行人脸比对，在马拉松赛事中有一些力不从心，因为几万名选手需要在短短的半个多小时完成入场，压力可想而知。

如何优化选手入场，既避免号码簿芯片的问题，又能快速准确的核验？

这就用到人脸识别的 1 比 N，我们提前拿到所有选手的照片，选手直接刷脸进行比对，快速核验入场。



这个优化的方案需要做到以下两点：

1）安全的人脸比对算法，该进去的人放行通过，不该进去的人拒绝开闸；

2）提前获取所有选手的照片，而且照片质量需要足够好，用作人脸比对的底库。

人脸比对算法这块要求是比较严格的，马拉松赛事每年都有很多替跑的人，我们的人脸算法必须要能找出替跑者，而且大麦的验票场景像演唱会的门票动辄上千，是决不允许让无票的

人入场，当然也要让买了票的人能够正常通过，这样的场景与刷脸支付场景比较像，因此我们使用了成熟的金融级别的人脸算法。

获取所有选手的照片的问题，我们是通过前面介绍的报名和领装备的环节解决的：

我们会要求选手在报名的时候提交一张本人的照片，但是这不能保证所有人都提交了质量足够好的照片，而且提交的照片是本人。这就需要在现场领取装备的时候解决，因为这影响到比赛时能否正常顺利入场。

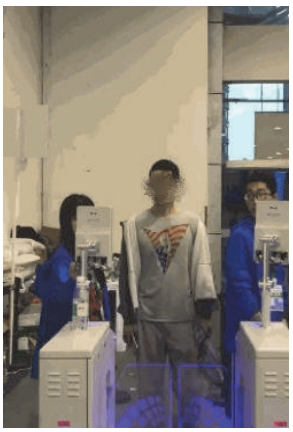
下面介绍我们是如何在完成领取装备的同时获取到选手的照片：

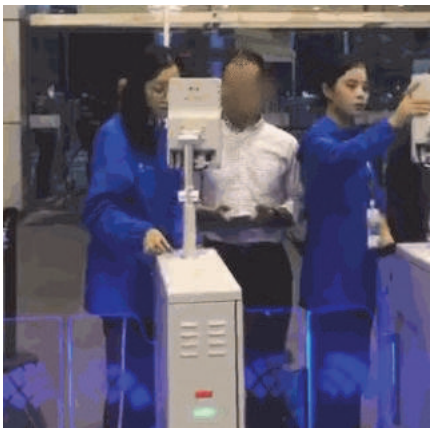
我们都知道，入场验票时是需要票的，选手在使用身份证领装备的时候其实他的身份证就是一张票，选手使用大麦的闸机系统刷身份证后，系统会根据身份证号读取到选手的信息，然后会通过闸机的打印系统打印出一张小票，选手拿着小票到相应的窗口领取参赛装备。但是这样无法满足主办的要求，必须要求选手持本人身份证到现场领取装备，当然也无法满足我们自己的需求——获取到所有选手的照片。因此我们在选手刷身份证的时候对他进行了 1 比 1 比对，确认是本人后再去采集一张用户现场的照片，这就完成了身份确认和照片的获取。

在这个方案里，领取装备的这个流程其实是可以优化的，用户在第 1 步报名时提交的照片有很多是质量很好可以直接用的，没有必要到现场再进行一次采集，对于这部分用户我们有了他们的照片之后会引导他们直接通过 1 比 N 人脸刷脸比对入场打印小票进行领取装备，这个方案的优点是：

1）直接刷脸，无需刷身份证+1 比 1 比对+采集，让领取效率更高，避免现场排队

2）解决了很多已上传人脸的用户忘带身份证或者身份证丢失导致的无法领取装备的问题





小结：我们通过报名的时候引导选手提交照片，通过现场引导让提交照片的选手直接刷脸领取装备，对于刷脸识别失败和未成功提交人脸的选手通过 1 比 1 进行本人确认并进行人脸采集，这样就确保了所有选手的照片都到了我们的人脸底库，也就使得我们在比赛当天可以让选手直接使用 1 比 N 比对直接刷脸入场。

三、保护用户隐私与数据安全

我们的人脸识别入场的方案是基于用户的照片，这就涉及到用户的隐私，我们会在马拉松报名的时候提前告知用户，获得用户的授权，我们需要做的更多的是如何保证用户照片的安全。下面介绍我们的方案：

闸机识别到用户后需要显示用户的照片，为了保护用户的照片不被泄露，我们对读取用户照片的接口做了授权判断，非授权设备无法获取照片数据，只有已授权的设备才能获取到照片数据，且是已经进行过加密处理的，保证了数据不会从端上泄露。

现场采集用户照片时，我们会将采集到的照片进行编码之后再进行一次加密操作，加密之后上传到服务端，上传成功之后会立即删掉本地的照片。在本场比赛结束之后服务器会自动将所有照片删掉。

通过以上操作，保证了用户的照片数据安全，也就保护了用户的隐私。

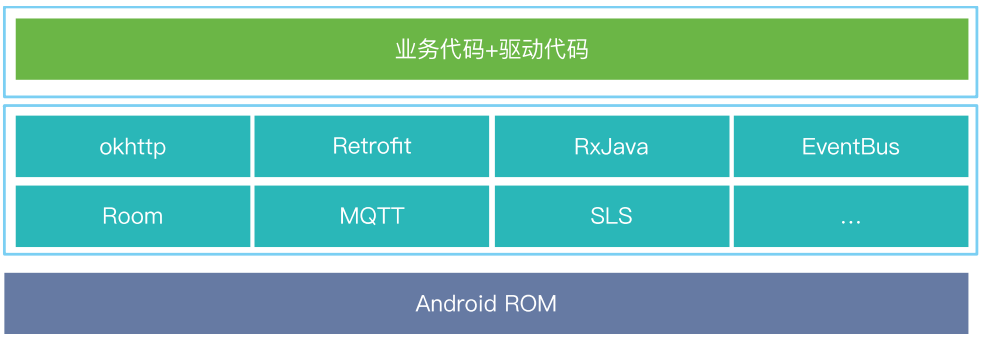
四、人脸闸机软件架构演进

讲完了我们的人脸识别方案，下面再介绍一下我们的软件架构是如何支撑我们的人脸识别

入场：

我们的闸机采用的是安卓系统，开发闸机的人脸核验系统跟开发普通的安卓应用稍有不同，我们需要针对硬件进行适配：机芯（摆闸、辊闸），打印机，二维码模块，RFID 模块，身份证模块等。

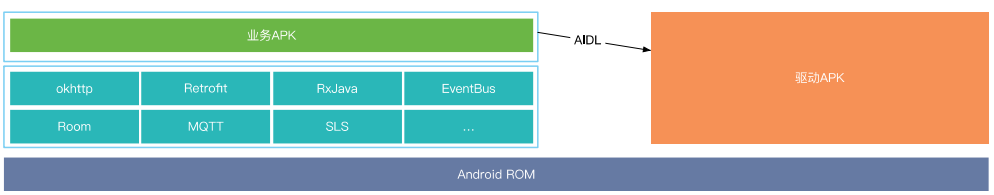
那整个闸机的软件架构很自然的就成下面的样子：



这样的方案用起来是没有问题的，但是时间长了会发现几个问题：

- 无论是改动业务代码还是硬件驱动代码都需要针对整个应用升级
- 随着闸机型号的增多，程序中会有各种型号的驱动，导致硬件驱动的代码臃肿，不易维护
- 关于硬件的适配，由于跟业务代码合在一起，只能我们自己来做，无法交给闸机的厂商

因此在这个基础上我们对软件架构进行了优化： 业务程序与驱动程序分开，采用了面向接口编程的思想，业务程序通过 AIDL 的方式将命令给到驱动程序。



这样的好处显而易见：

- 业务和驱动代码不再耦合在一起，各自独立，业务应用只需关心业务逻辑，驱动应用做好硬件驱动的事情；
- 业务程序无需关心驱动程序如何实现，驱动程序的实现可以交由厂家实现，我们制定标

准就好了；

- 业务程序和驱动程序独立升级，按需升级；
- 每种型号的设备使用统一的业务程序，只安装自身的驱动程序，不再需要将所有的不同型号的设备驱动代码都放在一起。

五、人脸识别能力优化

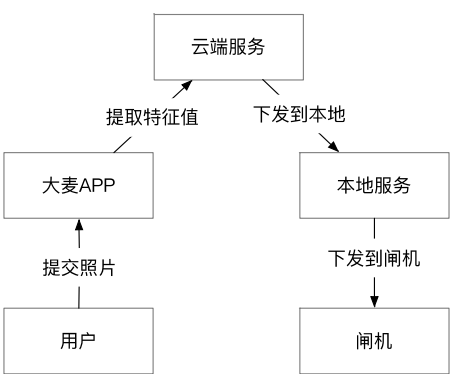
我们采用了安全的人脸识别算法，但这不表示算法就能解决我们现场的所有问题，大麦的现场环境较刷脸支付场景更为复杂，下面介绍我们是如何优化和提升我们的人脸识别能力。

1. 解决底库增大&&降低误识率

研究过人脸算法的同学应该清楚，再完美的人脸算法也会有误识的情况，也就是误识率，而且随着底库的变大，误识率也会跟着上升。马拉松的场景几万人很正常，既要降低误识率又要满足底库的变大，看似一个矛盾的问题，而且算法层面目前是无法解决，但必须要解决业务问题，我们的思路是：既然算法无法解决，那只能通过业务去解决，我们知道马拉松赛事还细分为全马，半马，情侣跑，迷你跑，家庭跑等不同的种类，那我们就可以根据这些不同的种类将人群分开，这样就做到了减少了底库，那误识率自然会降低。

2. 人脸数据全流程打通

上面讲过通过报名网站进行人脸采集或者现场刷身份证进行现场采集，完成人脸采集，但是还有一种情况是用户既没有在报名时上传照片，又没有携带身份证，这些用户只能通过大麦 APP 进行采集，那如何保证 APP 上采集了之后能马上入场领取装备？那就需要把整个流程打通，具体如下：



3. 动态远程调优

马拉松的人脸识别场景其实要比人脸支付的场景复杂，人脸支付大多是在室内，光线的影响会相对小一些，而马拉松既要考虑室内场景（领取装备）又要考虑户外场景（开跑入场），面临着曝光过度、逆光侧脸和远距离等的影响，因此需要根据具体环境来进行调优，我们通过在管理端动态修改影响人脸算法的各种参数，以及是否采用降级方案等，远程下发到现场所有设备，确保选手顺利高效入场，不会引起排长队的问题。

六、总结

我们根据马拉松赛事实际的业务场景，将人脸识别功能 1 比 1 和 1 比 N 应用到马拉松赛事，开启了马拉松行业的新的入场模式，但是我们的路还很长，人脸算法需要继续提升：提高识别率与降低误识率，不同环境下人脸算法的优化配置，室内与户外的不同验票方案以及已经开始商业化的 5G 能否给我们的业务带来新的突破，我们会将人脸识别功能应用到更多的核验场景，持续提升现场用户的入场体验和通行效率。

5G 关键技术及业务结合点

作者| 阿里文娱高级无线开发专家 梓烁

一、5G 的关键技术

5G 的核心技术点挺多，包含了很多技术集。5G 定义了三大应用场景：

eMBB：增强移动宽带，顾名思义是针对的是大流量移动宽带业务；

URLLC：超高可靠超低时延通信(3G 响应为 500ms，4G 为 50ms，5G 要求 1ms)，这些在自动驾驶、远程医疗等方面会有所使用；

mMTC：大连接物联网，针对大规模物联网业务。

1. eMBB

4G 已经那么快了，那么 5G 里面是怎么样继续提升容量的呢？

容量=带宽*频谱效率*小区数量

根据这个公式，要提升容量无非三种办法：增加带宽、提高频谱效率和增加小区数量。增加小区数量意味着建设更多基站，成本太高。增加带宽就意味着更多的资源投入，但频谱资源本身就是有限且稀缺的。因此提升频谱效率成为一种有效的提升容量的方向，采用校验纠错、编码方式等办法接近香农极限速率。相对于 4G 的 Tubor 码，5G 的信道编码更加高效；另外 4G 时代的 OFDM 技术要求频谱正交，而在 5G 中使用了 NOMA(non-orthogonal multiple-access)技术，频谱不再要求正交，利用率进一步提升。除调制技术外，在天线技术上也有了新的进展，通过 Massive MIMO 可实现容量的大幅提升。

2. uRLLC

uRLLC 的全称是超可靠、低时延通信，所以不仅仅只是低时延还需要高可靠。具备时延低且可靠后，一些工业自动化控制、远程医疗、自动驾驶等技术就可以逐渐构建起来了，这方面

带来的变革可能是天翻地覆的。

在 5G NR（New Radio，5G 空口标准）中子载波间隔不再像 LTE 的子载波间隔固定为 15Khz，而是可变的，可以支持 5 种配置，15kHz、30kHz、60kHz、120kHz、240kH。子载波间隔越大则时隙越短（最小的子载波间隔 15KHz 对应的时隙长 1ms、最大的子载波间隔 240KHz 对应时隙长 0.0625ms），对于 uRLLC 场景，要求传输时延低，此时网络可以通过配置比较大的子载波间隔来满足时延要求。

5G 支持更细粒度的网络切片，将网络切出多张虚拟网络，从而支持更多不同需求的业务。网络切片从无线接入网到承载网再到核心网，都是逻辑上隔离。网络切片做到了端到端的按需定制并保证隔离性，其中 NFV 和 SDN 等是这里面的关键技术

3. mMTC

mMTC 的 KPI 的连接密度是 1,000,000/km2，电池寿命是在 MCL（最大耦合损耗）为 164dB 时工作 10～15 年，这是一个非常具有挑战性的目标。目前有两个分支技术都在向这个目标演进，一个是国内主要推行的 NB-IoT 技术，一个是 eMTC 技术。这两个技术各有优缺点，NB-IoT 技术适用具有静止、数据量很小、时延要求不高等特点，但对工作时长、设备成本、网络覆盖等有较严格要求的场景，而 eMTC 则更适合对数据量、移动性、时延有一定的要求。

二、理性看待 5G 速率提升

对于 5G 速率的报道，一会 20Gbps，一会 4.6Gbps，一会 6.5Gbps，为什么会差别那么大？那些 4.6Gbps 的一定就比 6.5Gbps 的差吗？

1）5G 峰值下载速率 6GHz 以下 200MHz 4.6Gbps。这个 6GHz 是指的载波频率在 6GHz 一下，200MHz 指的是带宽，载波频率和带宽概念搞不清楚的同学可以自行百度。4.6Gbps 是峰值速率，按照 5G 的 KPI，频谱利用效率需要为 4G 的 3～5 倍，4G 是多少？4G 频谱效率是 5（即 20MHz 带宽实现 100Mbps 的峰值速率），那么按照 5G 的 KPI 我们来计算一下 200MHz 带宽应该要达到多少才达标，简单的公式计算，达标的速率应该是 3Gbps～5Gbps 。

2）毫米波 800MHz 6.5Gbps(4G LTE 可体验速率的 10 倍)。毫米波指的是频段，国际主流的是 28GHz，这个指的是载波频率。800MHz 指的是带宽，高频段就是好啊，资源相当丰富，动辄都 800MHz 带宽了。这个实际算下来这个的频谱效率只是 4G 的 1.625 倍，这个可能主要是由于带宽较宽，所以使用的 OFDM 子载波带宽也较宽，子载波间隔增大后频谱效率就降下来了。

但这个速率仍然很给力了。

值得注意的是这些速率都是峰值速率，是在一个基站下的你我他共享的资源，所以你的实际体验速率并不会那么快，基站侧会有调度算法来保证公平，但 5G 里面可能不会有绝对的公平了，付费的企业用户可能会获得更多的资源调度，不再是一锅端了。另外要注意的通信里的速率都是 bit，而非 byte，是有 8 倍的差距的，包括你家里装宽带时也是 bps，而非 Bps。

总结来说，就是大家记住 5G 的频谱效率 KPI，然后加上带宽就能知道峰值速率是多少了

而这个峰值速率只能说明你的总容量大小，和个人感知速率是不一样的，但是会明确瓶颈在哪里。不说多大带宽下实现多少速率的都是耍流氓。PS：载波频率和带宽是两码事，峰值速率和带宽和频谱效率有关，和载波频率无关。就和一趟火车一样，决定装载量大小的是车厢的多少，而不是速度。

三、业务结合点

1. VR / AR 技术的发展

伴随着 AR 和 VR 市场规模的不断扩大，视频流也势必会呈现显著的增长，而类似于 6DoF 的下一代内容格式会对网络提出更高的要求，个人数据速率的需求上限也会从 200Mbps 跳到 1Gbps，这些都会需要更多的带宽来支持。做 AR 和 VR 的很多公司已经开始摩拳擦掌了，准备好好把握住这一先机，大家对于 5G 显得热情高涨，都想尽早的拿到那张门票、打造爆款、占领市场。5G 是一种通信技术，本身解决的是传输的问题，本身 VR 和 AR 需要解决的很多体验问题、内容源问题、资源问题等都仍需要产业继续解决，当然谁解决的最好，与 5G 配合得最好，消费者肯买单就会占领市场先机。

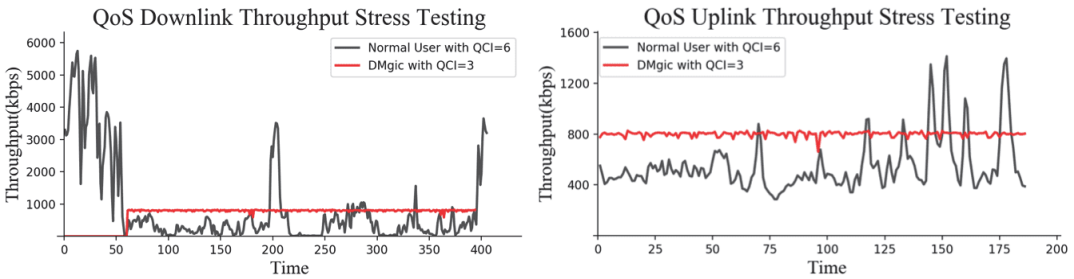
链路传输能力的提升，也会带来很多的好处，可以支持大带宽低延迟的数据传输，这样很多复杂耗时的计算、渲染等都可以在云端完成，云端的机器可以方便的进行弹性扩容来满足业务增长。端侧设备由于计算资源有限，设备耗电续航等因素的限制，很难单独完成这些复杂的功能，因此云端渲染或云端与本地结合的渲染方式成为一个很好的突破点，大带宽低延迟将带来和端侧本地计算一样好的体验。

2. 网络切片技术的应用

网络切片可分为核心网中的网络切片和接入网中的网络切片，核心网中的网络切片与虚拟化技术息相关。NFV（network function virtualization，网络功能虚拟化）与 SDN（software defined

networking，软件定义网络）作为实现核心网中的网络切片的主要技术支撑，受到了广泛的关注和研究。接入网中的网络切片实现更具有挑战性。除了用于不同的商业模型之外，针对业务的指标需求不同，网络切片和接入网络还需同时提供低时延、大连接、高可靠等性能指标，并保证网络切片之间的隔离。

阿里大麦正在探索网络切片相关的应用，在 4G 时代其实没有完整的网络切片方案，3GPP 协议定义了 QCI（QoS 等级标识），不同 QCI 承诺了不同的数据包延迟、误包率等，运营商通过 QCI 提供面向用户与业务等差异化服务。3GPP 定义的 QCI=3 的用户的典型场景是 Real Time Gaming(实时游戏)，一般普通用户的数据业务是在 QCI=6 上进行承载。如果下图所示，为某次压力测试中，QCI=3 的保障用户在数据包平均时延和抖动方面明显优于 QCI=6 普通用户，在带宽资源紧张时，QCI=6 的用户无法抢占足够多的带宽资源完成业务，而 QCI=3 的用户可以保持稳定的 800kbps 的稳定速率。4G 时代的"切片"仅仅是接入网的一个较粗的 QoS 等级划分，5G 的切片将是更完整的端到端的解决方案。



5G 时代将会有各种针对不同场景的网络切片，比如针对车联网的切片、针对 VR 的切片、针对物联网设备的切片等，而网络切片的粒度也会更细，针对不同的服务质量收费也会有差异。

3. 移动边缘计算（MEC）

在不考虑重传的情况下，LTE 网络内部时延是小于 20ms，而要 ping 外部服务器，这个时延通常在 40-50ms 以上，光纤的传播速度是 200 公里，5G 在应对时延敏感用例时，要求接入网时延不超过 0.5ms，这意味着 5G 中心机房(或数据中心)与 5G 小区(基站)之间的物理距离不能超过 50 公里。面对物理时延的挑战，我们不得不考虑在接入网引入移动边缘计算(MEC)、边缘数据中心，也就是将以前核心网和应用网的一些功能下沉到接入网。

边缘计算由于部署在靠近物或数据源头的网络边缘侧，具有融合的网络、计算、存储和应

用核心能力。利用边缘计算提供的计算能力和服务，能够满足低时延、海量连接业务需求和数据的聚合优化需求等，缓解核心网和回程链路的负载压力。因此，边缘计算和网络切片的结合变得尤为有意义。

在网络传输延迟或数据安全等角度考虑，很多的领域无法直接将数据传送至云端处理，因此边缘计算是一个大趋势，大家经常举例的自动驾驶就是一个例子，为了保证实时性和可靠性，图像处理需要在边缘端完成。除了这种意义上的移动边缘计算之外，其实运营商期望的移动边缘计算应该是在更靠近接入网的部分部署算力，支持边缘计算，后续可能应用可以直接部署于基站内的云设备内，这样对一些延迟极度敏感的应用将是一个好消息。

4. 物联网应用

目前物联网逐渐火热起来，mMTC 也是物联网三大场景之一，承担了未来智能世界里的重要想象空间。但是目前的 mMTC 仍然有一些亟待解决的问题，我们可以看到 5G KPI 里是要求能支持每平方公里 100 万个连接，这其实是一个非常让人兴奋的数字，但是这个会有点容易让人误解，100 万个连接并不是同时收发数据，只是连接，连接有可能是时断时续的，有可能是一天只发送一个数据包的监控节点。可以看到目前应用比较广泛的还是电力超表类应用，因为这类数据基本都只是在上报，而且频次要求不高，实时性要求也不高。但是对于很多的应用场景来说，实时 / 准实时的双向通信是很大的需求。

大麦目前已经将 NB-IoT 应用到实际产品中了，由于 NB-IoT 使用的是 CoAP 协议，而 CoAP 协议底层使用的是 UDP，不可靠的，在我们的部分场景中要求数据可靠到达，因此做了应用层的 ACK 应答机制及重传来保证数据可靠到达。

由于一些应用场景对功耗不敏感，但期望数据能尽快到达，所以与运营商沟通后关闭了 PSM 和 eDRX，以便让数据尽快到达，但是对于一些监测类的场景对于功耗是敏感的，因此就会通过睡眠等方式来尽可能的降低能耗，这对于纯上行监测类应用来说还好，但是如果想要准实时下发可就难了，因此也会限制一些场景的想象空间。

未来 5G 的 mMTC 场景还会基于 NB-IoT、eMTC 技术继续演进，期待未来能更好的解决目前存在的一些问题。

5. D2D 的应用

D2D 其实是一项挺有意思的技术，让设备和设备之间能直接通信。当然不是完全的自主通信，是在基站控制下完成数据通信，基站主要负责控制信令，设备间直接进行通信。这可能会

催生一些基于邻近特性的社交应用场景。其中车联网中的 V2V（Vehicle-to-Vehicle）通信就是典型的物联网增强的 D2D 通信应用场景。基于终端直通的 D2D 由于在通信时延、邻近发现等方面的特性，使得其应用于车联网车辆安全领域具有先天优势。在 D2D 通信模式下，两个邻近的移动终端之间仍然能够建立无线通信，为灾难救援提供保障。

家庭应用中的投屏场景是一个很好的 D2D 场景，但是目前基本都是 WiFi-Direct 的天下，如果 D2D 想要应用进来，还需要与这个强大的对手进行竞争。

6. CDN

4G 时代，CDN 基本部署在 CR（核心路由器）、SR（业务路由器）附近，部署位置偏上。同时，节点部署稀疏，平均每个节点覆盖方圆 10 公里。5G 时代，在架构上，CDN 应从 CR、SR 端向用户端迁移。同时在节点部署上，向小型化、高密化发展，原来每节点覆盖方圆 10 公里，现在需缩小到 1 公里甚至更小。网络切片中的 NFV 和 SDN 技术也将应用到 CDN 中，NFV 实现网络资源共享，扩展灵活，CDN NFV 实现硬件和软件解耦。SDN 让调度和路由控制更灵活，网络感知能力和集中控制与能力的开放，提供灵活调度和最优化的路由能力。

7. 产业互联网

借用阿里巴巴陈威如专家演讲中的一些观点，未来十年，是从消费互联化到产业互联化的全面协同升级。未来，产业互联网有两个发展方向，

第一，在你所处的产业环节进行线上线下融合。如果你是做零售的，你就要把线上、线下销售场景，用数字化、可视化的方式重构、融合起来；如果你是做供应链的，也要先做数字化，进行线上、线下融合，达到线上线下一盘货。

第二，做全链路环节的数字化相连。当你把全链路串起来以后，就会对生态圈、消费者、企业商业模式产生一个极大的变革。因此 5G 可能会依托于物联网技术带来全链路的数字化，进而助力产业互联网。

四、总结

在本文中，我们可以了解到 5G 的关键技术。

1）其中单基站的峰值速率要达到 20Gbps，频谱效率要达到 4G 的 3～5 倍，这是关于 eMBB 超宽带的指标，使用的主要技术包括 LDPC / Polar 码等新的编码技术提升容量，使用毫米波拓

展更多频谱，使用波束赋形带来空分多址增益，使用 NOMA 技术实现 PDMA 功率域的增益，使用 Massive MIMO 技术来获得更大的容量，毫米波让波长更短，天线更短，在手机上可以安置的天线数更多，基站侧可支持 64T64R 共 128 根的天线阵列。

2）时延达到 1 毫秒，这是关于 uRLLC 的场景，主要是新的空口标准 5GNR 中定义了更灵活的帧结构，更灵活的子载波间隔配置，最大的子载波间隔 240KHz 对应时隙长 0.0625ms，这样超低时延应用称为可能。通过新的多载波技术解决目前 CP-OFDM 中存在的保护间隔等资源浪费，降低时延增大利用率。除此之外，还有网络切片技术，让网络变得更加弹性，可以更好的支持超低时延的应用，建立一条端到端的高速功率，网络切片技术主要是核心网的 SDN 和 NFV 的应用。

3）连接密度每平方公里达到 100 万个，这是关于 mMTC 的场景，目前标准主要还是基于 eMTC 和 NB-IoT 进行演进，两项标准各有优缺点，对数据量、移动性、时延有一定的要求的场景 eMTC 更合适，具有静止、数据量很小、时延要求不高等特点，但对工作时长、设备成本、网络覆盖等有较严格要求的场景 NB-IoT 更合适，目前国内主要覆盖的是 NB-IoT。这里的连接量其实是一个相对弹性或理想的值，因为连接量的提升主要是以终端通过 PSM 或 eDRX 技术实现休眠所带来的，未来更多的并发能力，更小的网络信令消耗、更多的突发数据包等场景都需要被考虑到，这部分的演进仍然有着较长的路要走。

打破行业困境，大麦如何引领 NB-IoT 技术的创新应用

作者| 阿里文娱高级开发工程师 智毅

一、物联网 NB-IoT 技术简介

1. 业务背景

受限于移动蜂窝网络（2G/3G/4G）容量问题，人流密集的场所会造成网络通信瘫痪，这种况下使用移动网络的业务就会受到灾难性的影响。因此解决移动网络问题，成为现场娱乐行业的首要任务和挑战。5G 通信技术虽说在设备容量上增加到每平方公里 100 万个，但其大规模部署和低廉的模组仍需时日。因此用于物与物通信的窄带物联网技术 NB-IoT 是否可以解决现场网络问题呢？

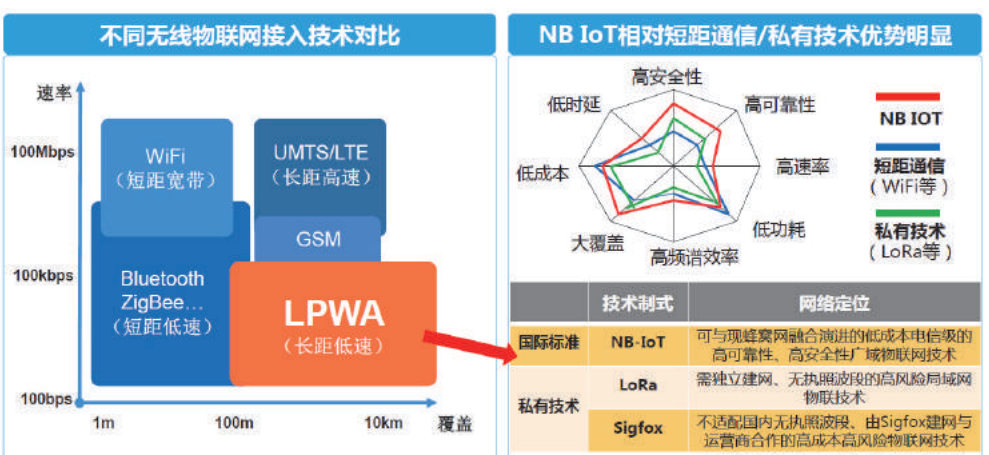


图 1. NB-IoT 技术的优点

2. NB-IoT 的技术特点

NB-IoT 是有电信级保障的物与物之间的长距离通信技术，相对于短距离通信和私有技术优

势明显。因此在大型娱乐现场或者体育赛事中，NB-IoT 就像是一条救援通道，可以正常运行。

图中 1 描述，NB-IoT 带宽 200KHz，上下行最大传输速率 250kbps，单个基站网络容量在 4-5 万个 NB 设备，信号覆盖范围比传统通信网络有 20dB 的增强，因此可以有效覆盖室内，地下室等遮挡严重的环境。NB-IoT 另一个特性就是低功耗，通过使能 PSM（Power Saving Mode）和 eDRX（enhenced Discontinuous Reception）功能可以起到降低功耗的作用。该方法主要是通过减少 NB-IoT 设备和基站的信令交互以及数据上报的频率来达到降低功耗的目的，以上都是 NB-IoT 技术的一些优点。

二、NB-IoT 平台的选择

工程化的目的是针对项目的使用场景，合理和创新地使用已有技术的优缺点，将其业务性能发挥到最优。目前 NB-IoT 由三大运营商来运营，因此基础设施完善，通信质量有保障。我们选择 NB-IoT 原因如下：

1. 运营商 NB-IoT 基站部署早，最先运营，覆盖地区范围最广

如下图所示[1]（数据来源于中国运营商）运营商 NB-IoT 覆盖全国 30 个省和直辖市，且区域覆盖率都在 93%以上，可以满足全国多数大型场馆以及郊区户外场景，其中大麦的业务场景主要包括城市中的大型场馆，以及郊区的音乐节场地；

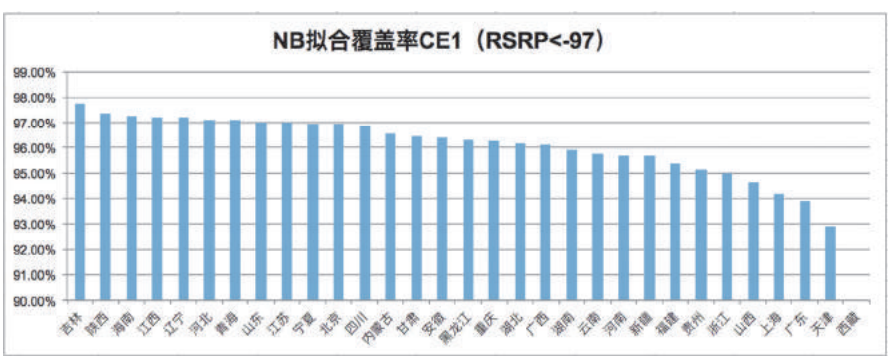


图 2. NB-IoT 全国分布和覆盖率情况

2. 平台有开放的 API 接口，支持第三方应用平台的接入

运营商物联网开放平台提供了海量 API 接口给第三方应用开发者。通过调用平台的接口，开发者可以开发出基于多种行业设备的应用，如公共事业、智慧家庭、智慧场馆等，从而实现

对设备的管理（包括设备的增、删、查、改）、数据采集、命令下发和消息推送等功能。

三、NB-IoT 通信协议

NB-IoT 设备和运营商物联网开放平台之间采用 CoAP 协议通讯（注:在设备侧，CoAP 协议栈一般由 NB-IOT 芯片模组实现），CoAP 消息的 payload 为应用层数据，应用层数据的格式由设备自行定义。由于 NB-IoT 设备一般对省电要求较高，所以应用层数据一般不采用流行的 json 格式，而是采用二进制格式。

NB-IoT 设备只能发送 ASCII 码，因此发送的数据需要先转换成对应的 ASCII。该数据需要通过物联网云平台的编解码插件转换成对应的 json 数据，通过运营商平台预留的回调接口发送给服务端；下行服务端发送的数据，需要对应的编解码插件转化成 NB-IoT 可以识别的 ASCII 码，设备通过串口接收解析数据。详细流程如下图 3 所示。

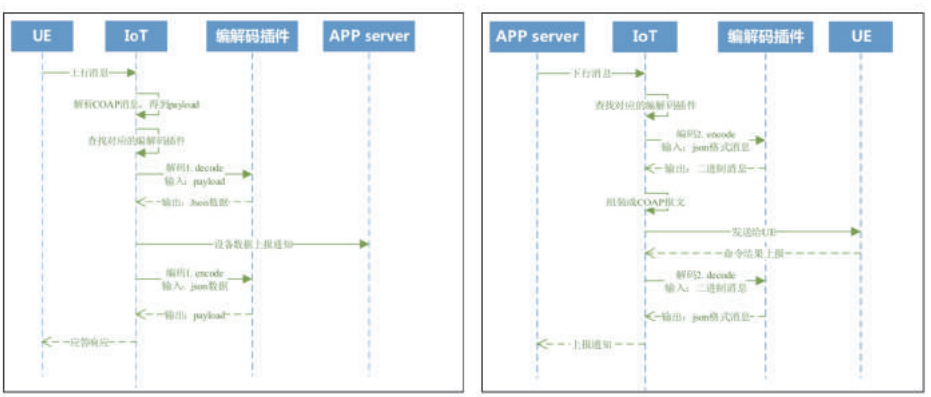


图 3. NB-IoT 终端和物联网平台的交互流程

四、如何提高 NB-IoT 的实时性：

NB-IoT 的一个优点是低功耗，一块电池理论上可以使用 10 年，但低功耗是以牺牲实时性为代价。之所以能够实现低功耗，是由于 PSM 和 eDRX 功能的引入，目的是减小数据发送频次，减少和基站的信令交互。这种低功耗适用于检测类设备，对实时性要求不高，数据重复性大的场景。但我们要应用的场景是演唱会或赛事的新增票，对数据实时性要求很高，也就是不希望 NB-IoT 模块进入 PSM 或 eDRX 模式，设备始终保持和基站的连接。因此选择合适的 NB-IoT 业务模式，也是至关重要的。NB-IoT 的业务模式，决定了 NB-IoT 的使用场景，基站通过获取

设备 SIM 卡 APN（Access Point Name）的业务参数，来对 NB-IoT 设备进行控制。

根据我们的业务场景，我们需要选择设备和基站始终处于交互状态，也就是一旦 NB 设备连接上基站后，不会进入空闲和睡眠模式，会始终和基站保持连接，随时进行数据交互，从而可以有效降低数据接收和发射的延时。其中 eDRX 周期很短，只有 2.56s，也就是在长时间没有数据交互的情况下，设备从 eDRX 跳出进入连接态，只需 2.56s。

五、赋能和创新带来的业务模式

利用 NB-IoT 这条稳定的通信链路，开发了新增票业务模式和现场大盘监控业务。新增票业务是指：当项目开始验票后，售票还在进行，为了确保客户新购买的票能够核验，需要将这些数据实时同步到核验设备上。新增票链路自上而下涉及：大麦服务端（新增票来源）、运营商云平台、麦小智（NB-IoT 设备）、PDA。

一张新增票通过服务端推送给运营商云平台，然后经过核心网，基站发送给麦小智中的 NB-IoT 设备，麦小智接收到 NB-IoT 的数据后经过解码，把新增票通过 MQTT 推送给 PDA，从而完成了一张新增票的流程。

大盘监控数据流程 PDA 将验票数据通过 MQTT 推送给麦小智，麦小智汇总完票务数据后，通过 NB-IoT 发送给物联网云平台，大麦服务端通过回调地址将上报的数据推送到数据大盘监控。其中还涉及到验票项目和 NB-IoT 设备的绑定和编号规则，用于给制定的项目设备组发送新增票，以及利用 NB-IoT 推送设备密钥用于连接阿里云 IoT 平台，这里就不再做详细描述。新增票和监控大盘流程图如下所示：

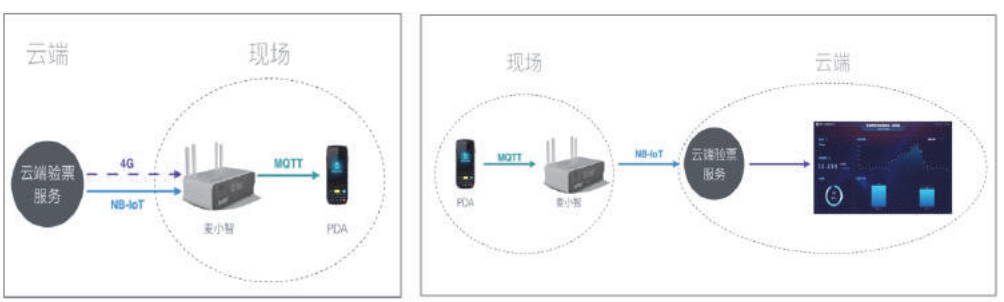


图 4. 大麦 NB-IoT 的上下行业务

工程应用中对 NB-IoT 新增票链路的思考：

1）NB-IoT 只适合固定场景应用，如燃气表，路灯吗？

这些场景只是利用了 NB-IoT 高覆盖和低功耗的特点，而且目前市场上对 NB-IoT 的应用主要是一些重复数据的上报，并没有将 NB-IoT 其他技术特点去进行探索和创新，造成其应用没有任何突破和创新；因此结合自身业务以及合理利用技术本身的优势更为重要。

2）数据通过服务端下发给物联网平台，物联网云平台推送给 NB-IoT 设备，如何保证数据不丢失

除了要考虑物联网平台，基站的通信质量问题，我们还需要关注设备是否可以 100%接收到数据，这就需要我们通过一些应用层的保障机制来确保数据的到达。

1）接收到的新增票通过什么方式推送给 PDA，保证 PDA 能够拿到所有新增票数据

针对不同场景，（如验票开始前后，新增数据开始推送时间点，以及新增加的验票设备 PDA）需要设计一套全面的同步机制，保障云端新增数据能够在验票项目周期中实时快速的将数据推送至 PDA，不影响现场的客户入场体验；

2）NB-IoT 的数据传输能力如何，是否满足业务场景，如何处理高并发业务场景

设备过多且集中注册在同一个基站下，数据收发频率较高时，就需要考虑并发机制，以免造成基站负载饱和问题。因此在多台设备连接同一个基站下，需要通过优化设备接入机制和数据发送/传输频率来解决高并发的冲突。通过实践证明，NB-IoT 的实际能力并没有理论描述的那么弱，并且可以满足我们目前新增票和监控数据的业务场景。

3）优化新增票效率和数据安全

针对 NB-IoT 数据的传输能力，如果用较小的带宽传输较多的有用数据，以及提高传输效率需要着重去设计，比如数据压缩方法；还有从安全的角度，通过加密的算法去保证数据的安全。因此为了保障设备数据的安全和重要信息的存储，需要一套安全的保障机制。

关注我们

（阿里文娱技术公众号）

关注阿里技术

扫码关注「阿里技术」获取更多资讯

加入交流群

1）添加“文娱技术小助手”微信

2）注明您的手机号 / 公司 / 职位

3）小助手会拉您进群

By 阿里文娱技术品牌

更多电子书

扫码获取更多技术电子书