

智能弹性

在虎牙降本增效上的探索与实践

郑健彦 虎牙 AIOps leader

讲师简介



郑健彦

HUYA 基础保障部
AIOps leader

2017年华南理工大学硕士毕业后，加入联想数据中心部门负责故障预测等工作。2019年加入虎牙直播，在质量，效率和成本三个方面落地AIOps。涉及异常检测，多维度根因定位，时序预测，智能弹性和大数据算力调度等相关工作。

”

目录

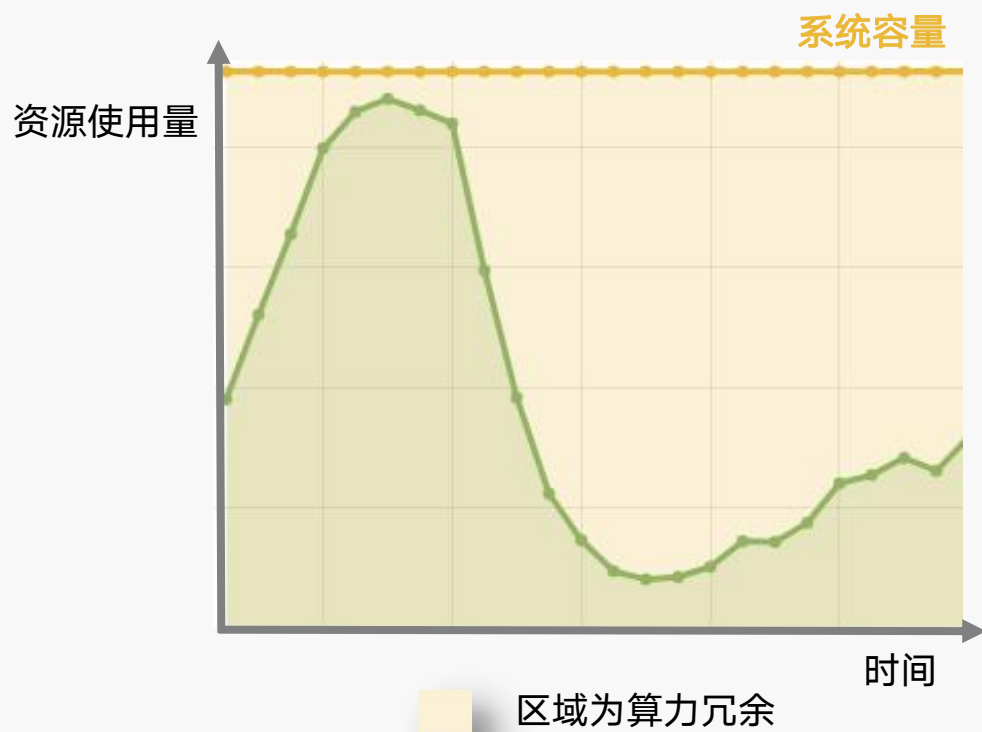
- 亮点介绍 & 案例背景
- 基于AI预测的日常弹性
- 业务容量模型
- 基于容量模型的赛事弹性
- 边缘算力智能弹性

亮点介绍

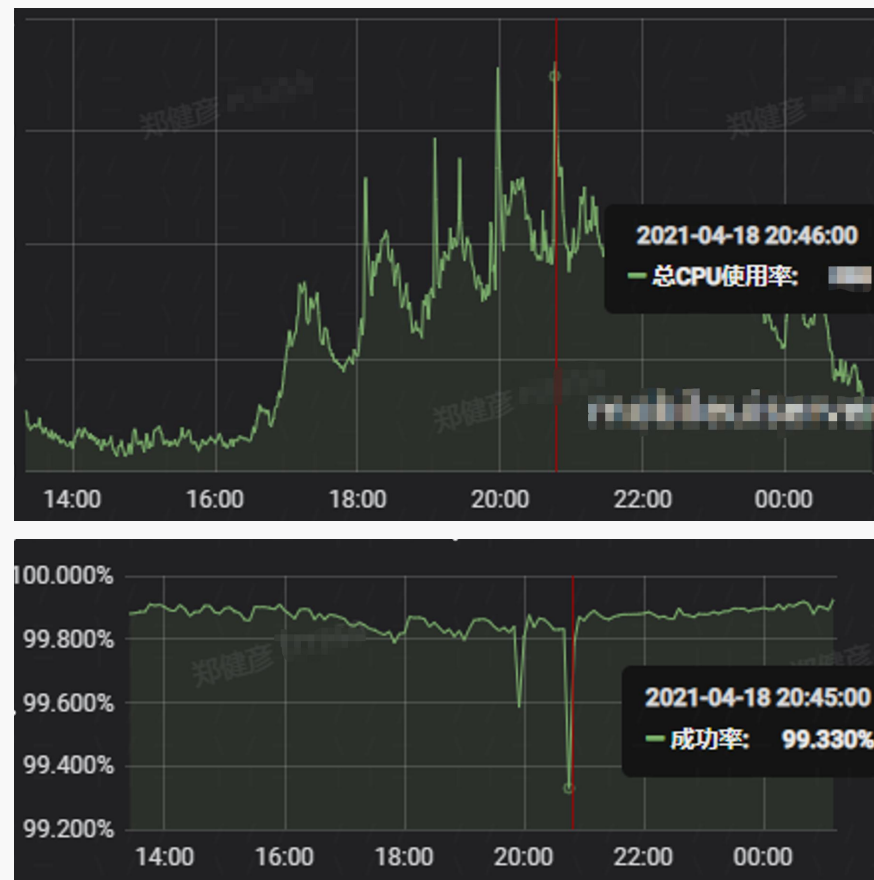
- 什么是智能弹性 AI-HPA ？
- 科学计算一年流量最高的时刻需要准备的算力资源
- 保证在线服务在7*24h下不过载
- 一键对所有服务预扩容到所需的容量

案例背景

- 在直播场景下，流量呈单峰结构

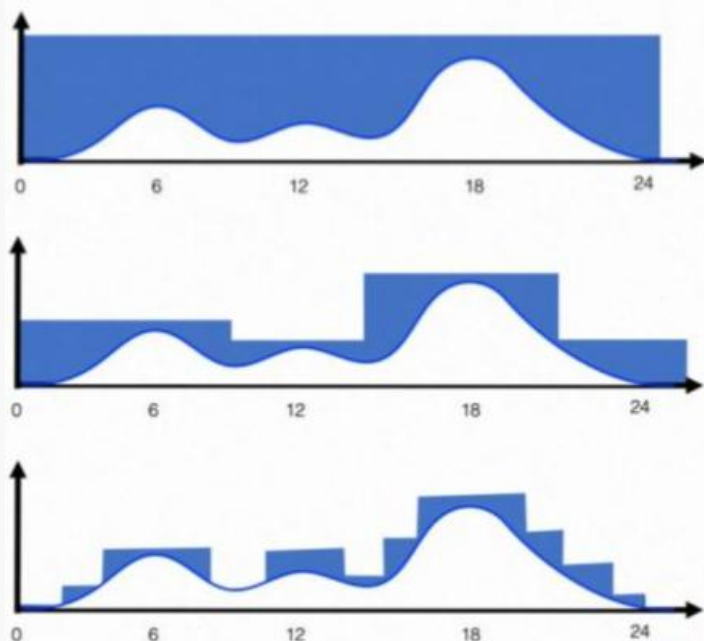


- 赛事（游戏直播界的双十一）期间质量抖动



案例背景

Kubernetes 中应用实例数设置的三种策略



固定实例数

- 为应用负载设定固定的Pod数量
- 缺点：业务存在波峰浪谷，固定实例数会造成较大的资源浪费

定时弹性 (CronHPA)

- 设定定时规则，在固定时间进行实例数伸缩
- 缺点：设定定时规则较为复杂，定时间隔不能太短，会造成资源浪费

自动弹性 (HPA)

- 根据应用实时负载设置实例数量
- 缺点：弹性滞后，导致业务部分流量响应慢或者超时



利用率低



周期偏移适应差
配置繁琐



健壮性差
配置复杂

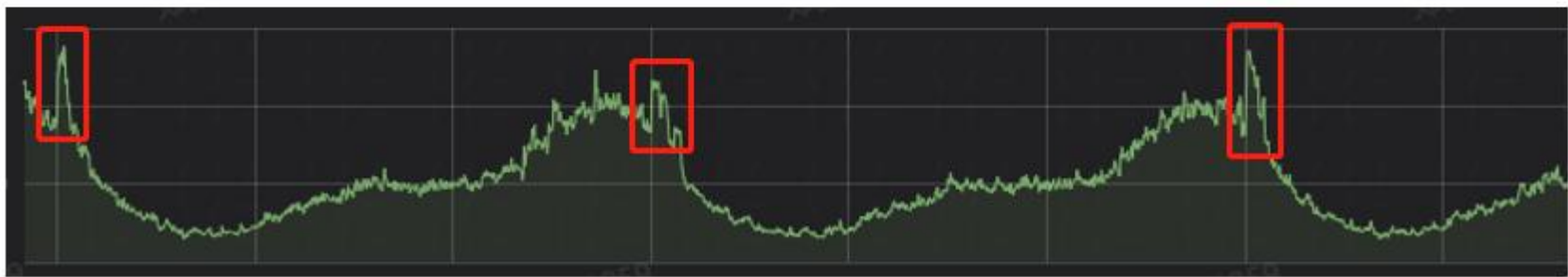
基于AI预测的日常弹性

(AI-HPA的常规模式)

核心实践1 - 基于AI预测的日常弹性

为什么需要预测？

- 标准的 HPA 是基于指标阈值进行伸缩的，常见的指标主要是 CPU、内存。
- 但是基于指标的伸缩存在一定的时延，这个时延主要包含：采集时延+ 上报时延+ 伸缩时延。
- 当负载的峰值毛刺非常尖锐时，可能会由于 HPA 的时延造成副本数目无法及时变化，短时间内应用的整体负载飙升，响应时间变慢。



资源调度

拉镜像

容器创建

容器启动

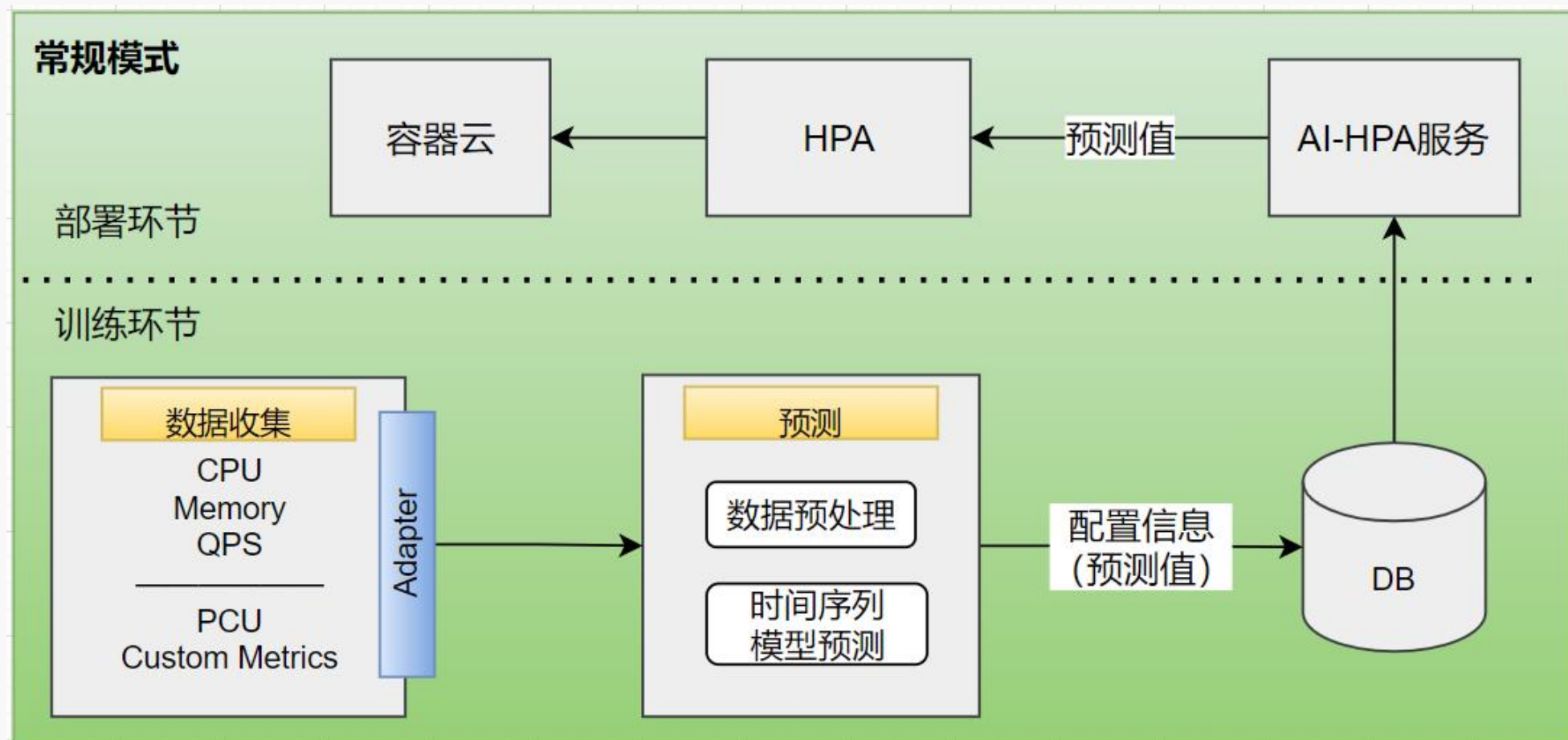
应用启动

启动耗时长

核心实践1 - 基于AI预测的日常弹性

Idea: 系统在服务指标突增之前进行提前扩容

总体架构:



核心实践1 - 基于AI预测的日常弹性

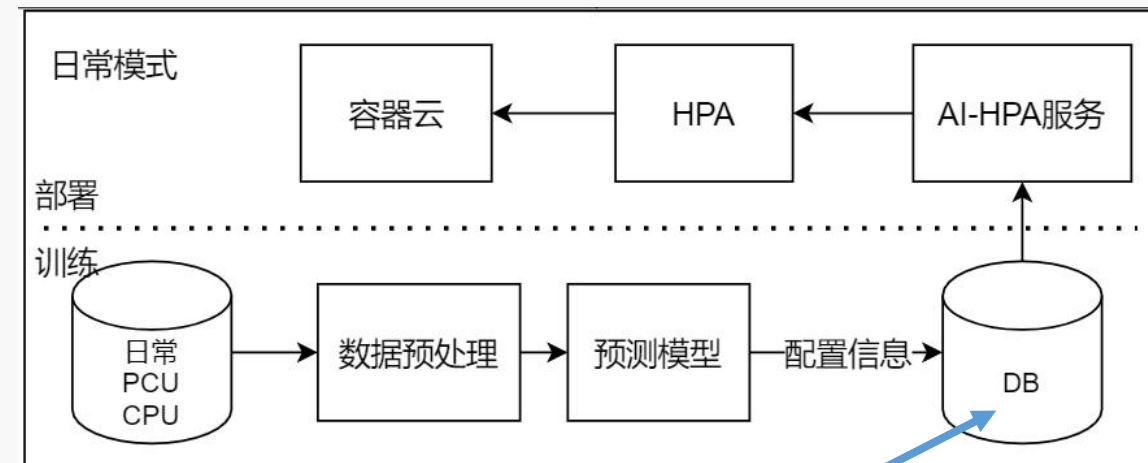
• 服务的副本数计算公式

期望副本数 = $\text{ceil} \left(\text{当前副本数} \times \frac{\text{max(周期预测, 当前指标)}}{\text{期望指标}} \right)$

$\text{max}(\text{预测指标}, \text{当前指标}) > \text{扩容阈值} ? \rightarrow \text{扩容}$

$\text{max}(\text{预测指标}, \text{当前指标}) < \text{缩容阈值} ? \rightarrow \text{缩容}$

• 通过训练和预测生成 “AI HPA定时伸缩配置”



```

{ 'type': 'cron',
  'segments': [
    { 'start': '00:00:00', 'end': '00:15:00', 'cpusum': 124.1979 },
    { 'start': '23:55:00', 'end': '23:59:59', 'cpusum': 124.1979 },
    { 'start': '00:15:00', 'end': '00:35:00', 'cpusum': 124.1979 },
    { 'start': '00:35:00', 'end': '00:55:00', 'cpusum': 77.2948 },
    { 'start': '00:55:00', 'end': '01:15:00', 'cpusum': 51.6321 },
    { 'start': '01:15:00', 'end': '01:35:00', 'cpusum': 37.9913 },
    { 'start': '01:35:00', 'end': '01:55:00', 'cpusum': 33.185 },
    { 'start': '01:55:00', 'end': '02:15:00', 'cpusum': 28.4009 },
    { 'start': '02:15:00', 'end': '02:35:00', 'cpusum': 27.0142 },
  ]
}
  
```

核心实践1 - 基于AI预测的日常弹性



预测算法：

LightGBM

支持分位数回归

不用处理缺失

方便利用特征工程做一些处理

成本：

每个分组 “取数据+训练+预测” 15s 完成

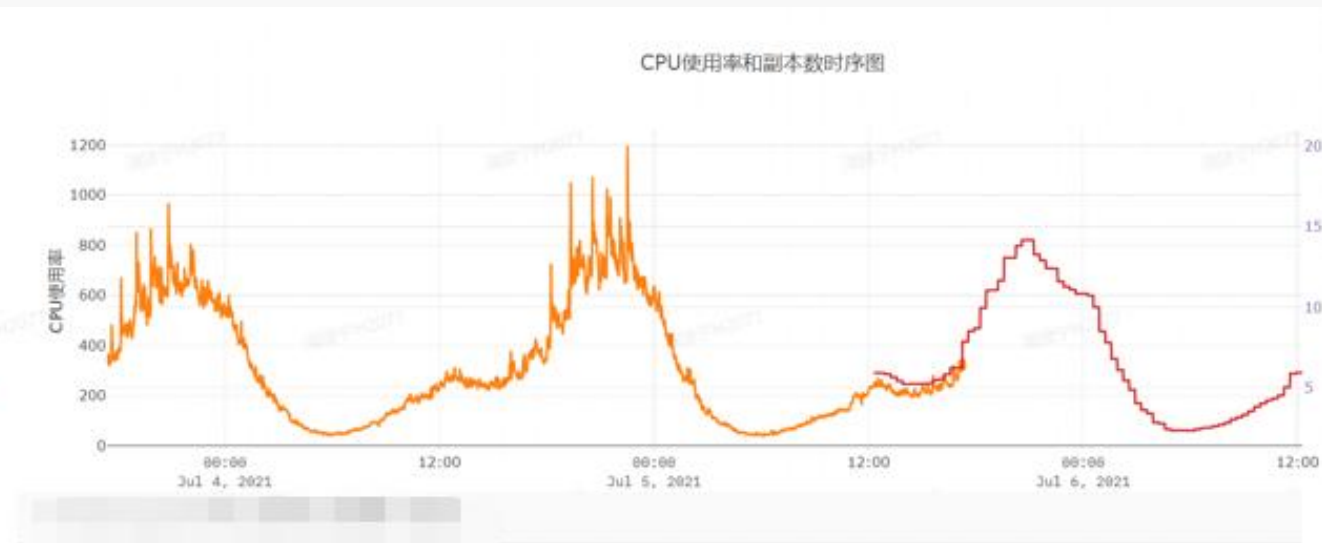
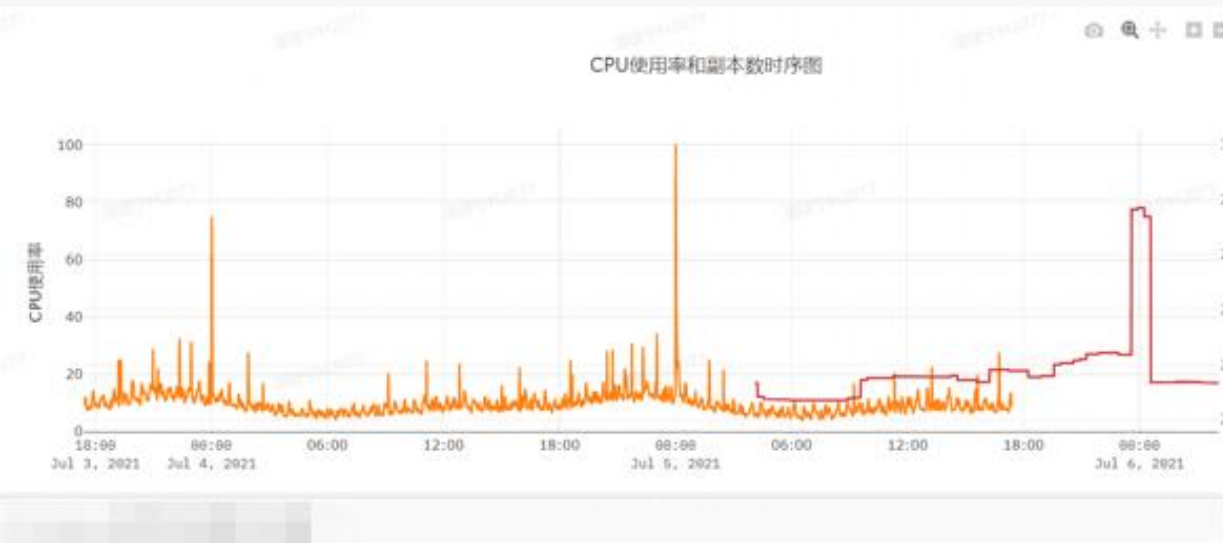
典型服务预测准确率：
90+%

Question

我理解AI-HPA相对于HPA解决了弹性扩容滞后的问题，是一种成本换质量的方法，如何能利用这个节省成本呢？

核心实践1 - 基于AI预测的日常弹性

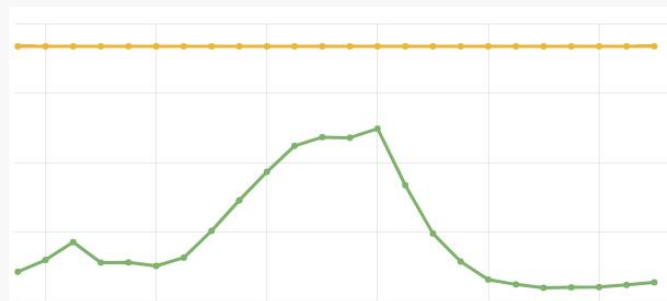
预测效果图



成果展示1 - 基于AI预测的日常弹性

开启弹性伸缩

节省成本28%



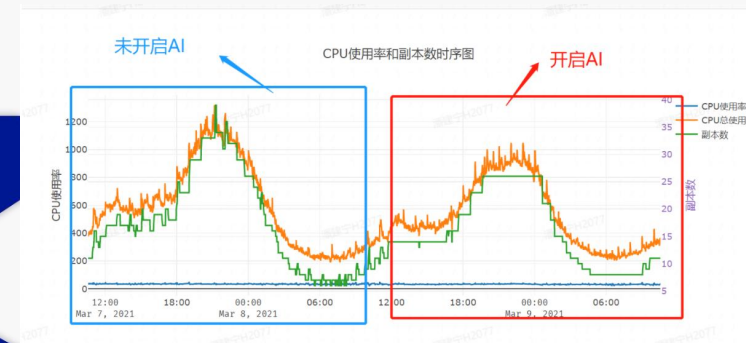
— 核时基线
— 弹性核时

提前扩容

对周期性流量突增，在突增之前提前扩容

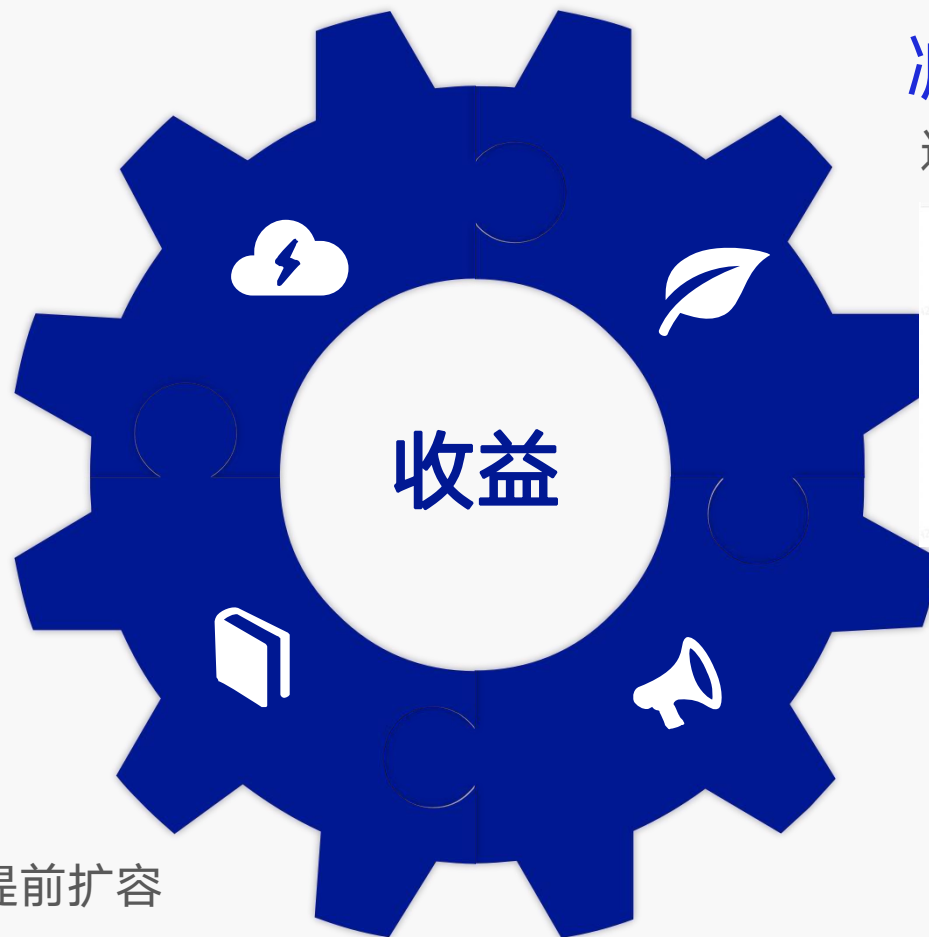
减少频繁扩缩容

避免副本数震荡



避免监控数据出错 造成的误缩容

预测值和真实值相互兜底



业务容量模型

保障服务在赛事期间的质量稳定

核心实践2 - 业务容量模型

赛事-游戏直播界的“双十一”

赛事进度 »

春季赛常规赛

春季赛季后赛

夏季赛常规赛

夏季赛季后赛

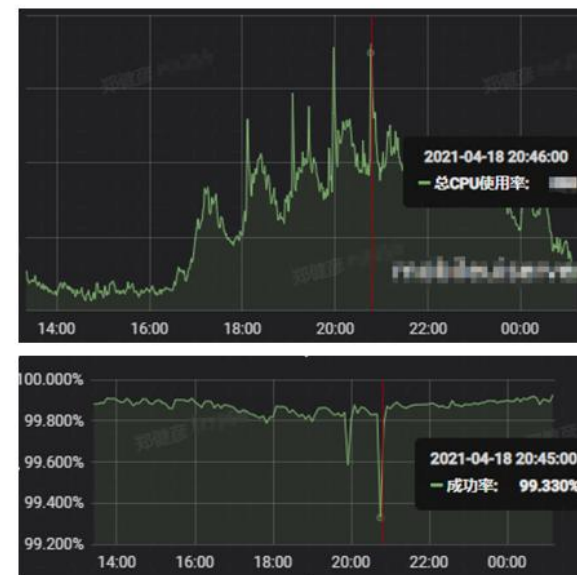
全球总决赛资格赛

全球总决赛

如何用**最低的算力成本**保障赛事期间质量的**稳定**？

- 哪些应用是赛事PCU相关的？
- 应用在特定的PCU下需要多少资源？

(游戏直播界的双十一) 赛事期间质量抖动



核心实践2 - 业务容量模型

哪些应用是赛事PCU相关的？

容量模型：

PCU与使用核数的关系

- 很多核心服务的峰值核数
和峰值PCU的线性
相关性很强
- 应用发版等可能会导致
资源使用发生变化（如
右图三）这意味着我们
不能直接取长时间每天
一个点的数据来回归。



一些应用每天的PCU峰值和使用核数峰值的散点图

核心实践2 - 业务容量模型

应用在特定的PCU下需要多少资源？



对总PCU做线性回归，但只考虑赛事那一段

$$\text{核数} = k * \text{PCU} + b$$

- 可解释性：

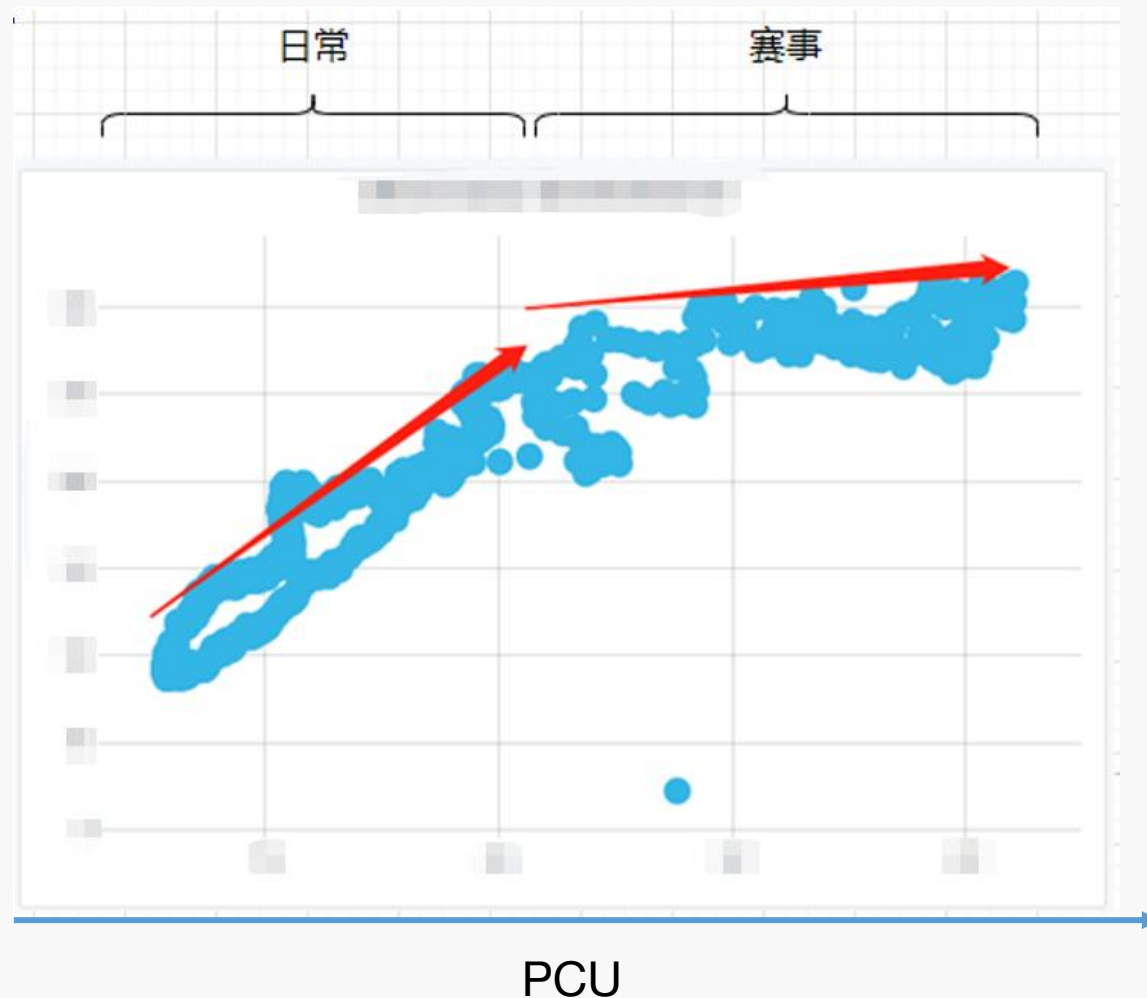
k 意味着每观众PCU需要多使用多少核

- 是否赛事PCU相关

赛事PCU相关应用：**相关系数** > 0.9

对与赛事PCU无关的服务，k为0，b为高峰期使用核数

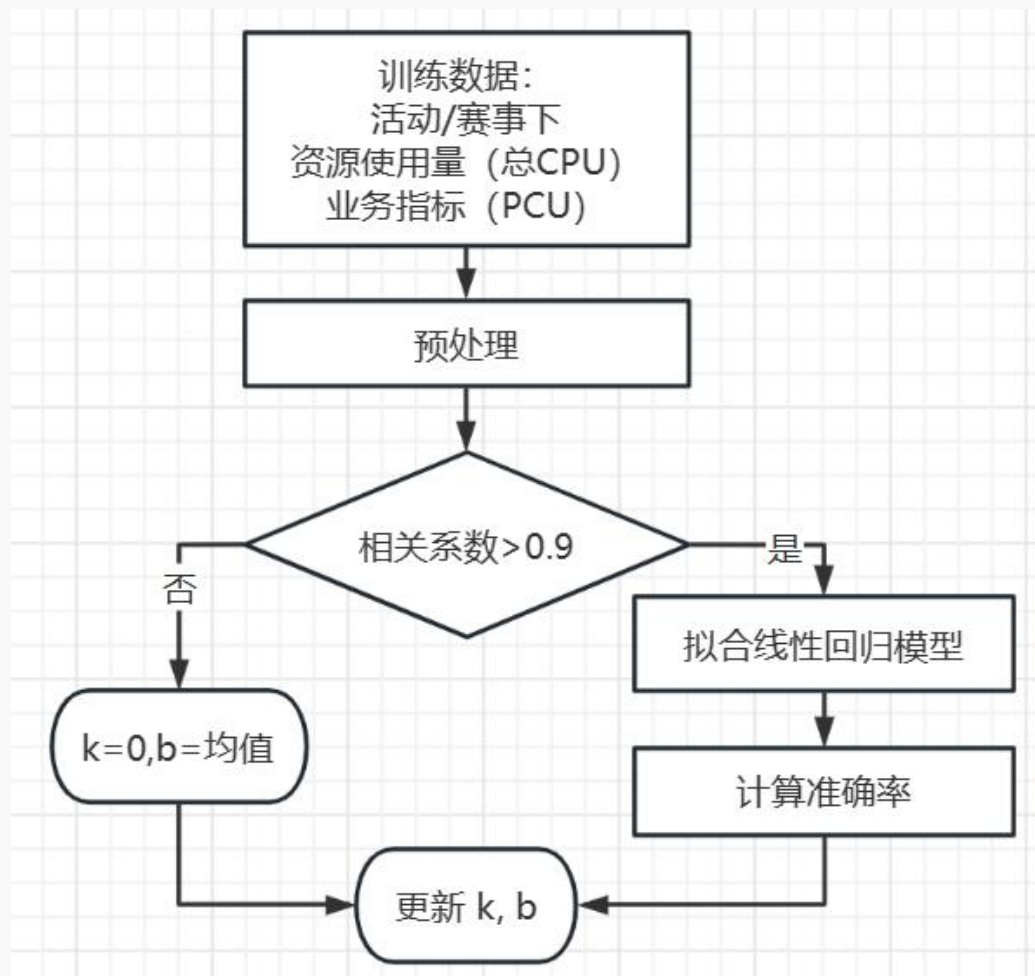
使用核数



每分钟粒度的PCU和使用核数散点图

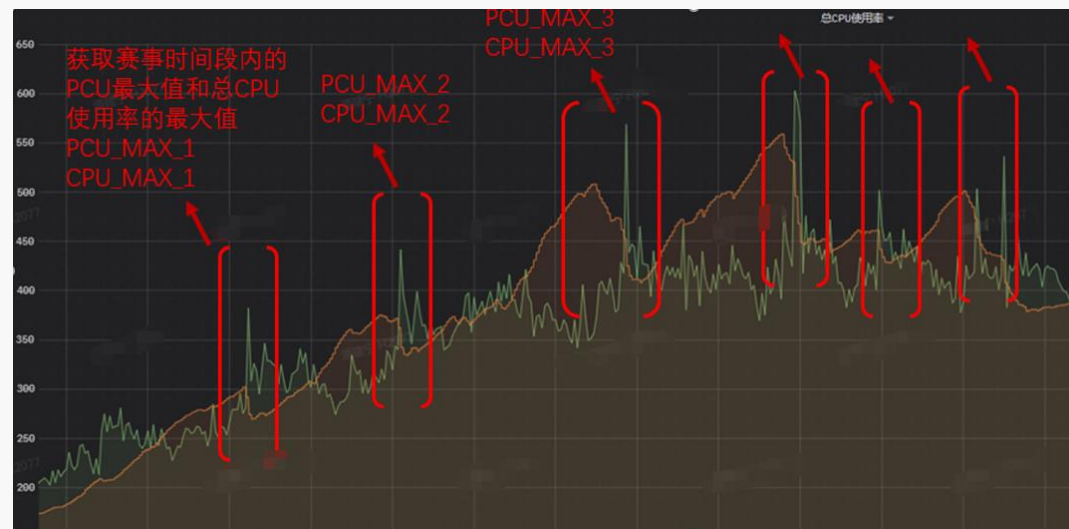
核心实践2 - 业务容量模型

训练和更新



— CPU

— PCU



- 更新周期：一周一次

- 准确率

整体准确率为89.35%

资源使用top100的应用准确率： 90.43%

基于容量模型的赛事弹性

(AI-HPA的赛事模式)

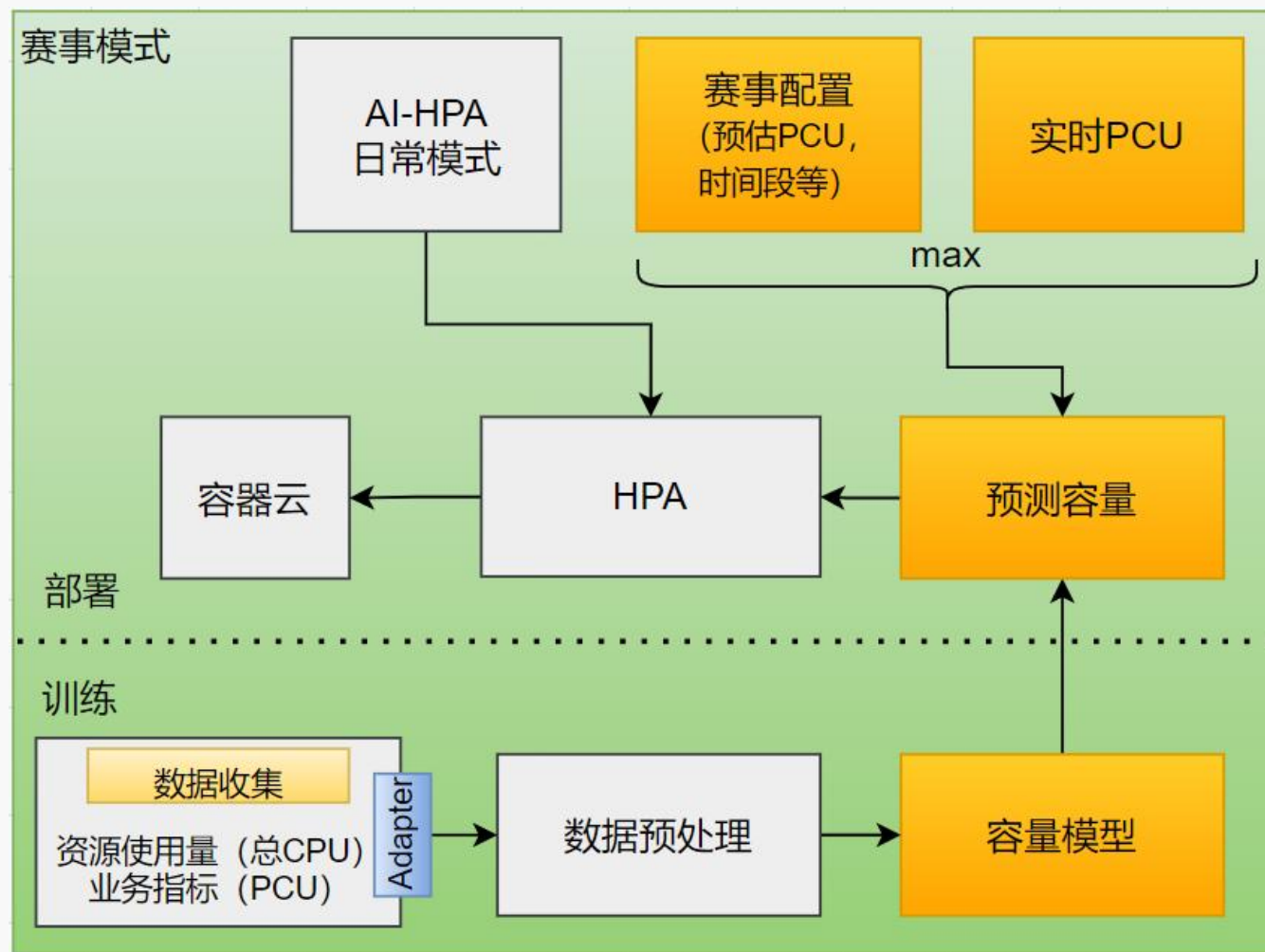
核心实践3 - 基于容量模型的赛事弹性

总体架构

将容量模型植入AI-HPA, 实现根据PCU扩容

将预估PCU和实时PCU输入容量模型，
得到预测使用核数，进而得到预测CPU使用率，
k8s比较预测CPU使用率和扩缩容阈值，决定是否要扩缩容

- 日常小赛事由实时PCU进行扩容
(PCU > ? 赛事AI-HPA模式自动生效)
- 大赛事配置预估PCU，提前扩容



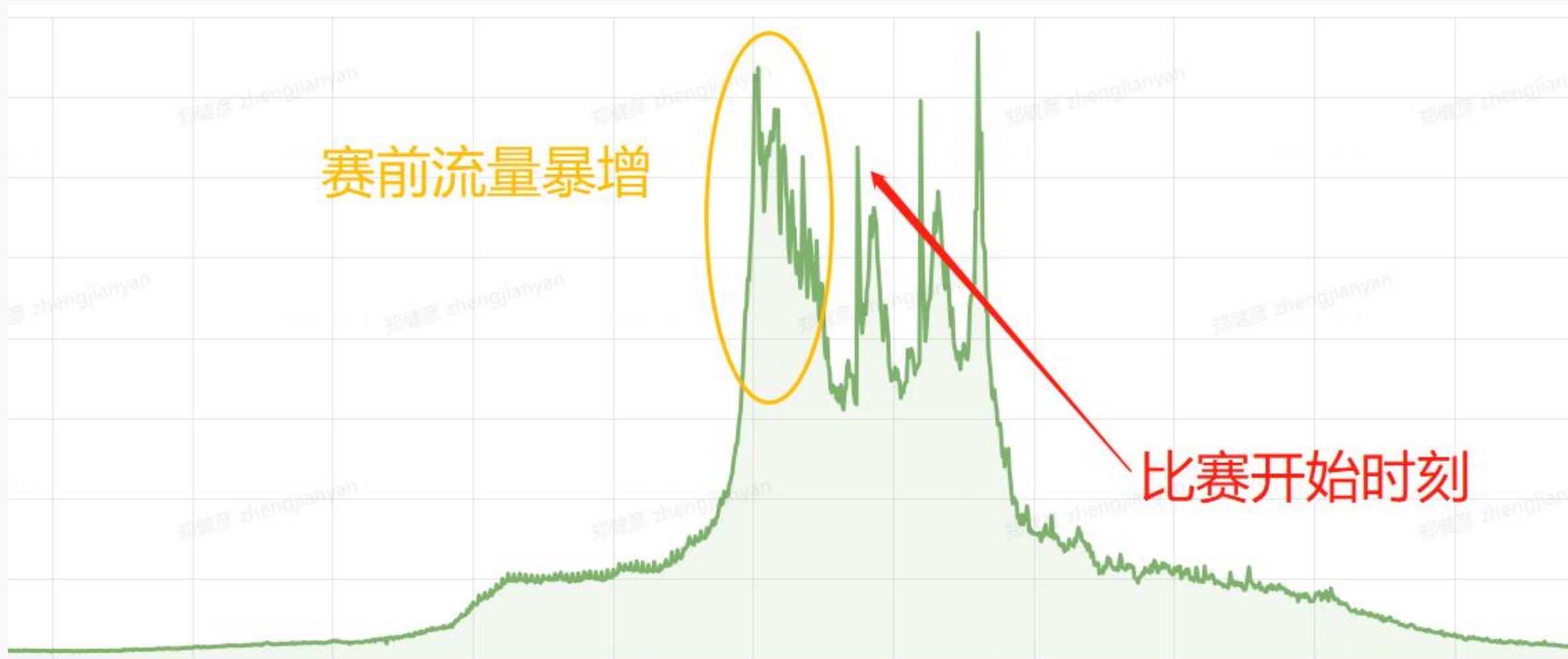
$$\text{期望副本数} = \text{ceil} \left(\text{当前副本数} \times \frac{\text{max}(\text{周期预测}, \text{赛事预测}, \text{当前指标})}{\text{期望指标}} \right)$$

核心实践3 - 基于容量模型的赛事弹性

配置预估的PCU

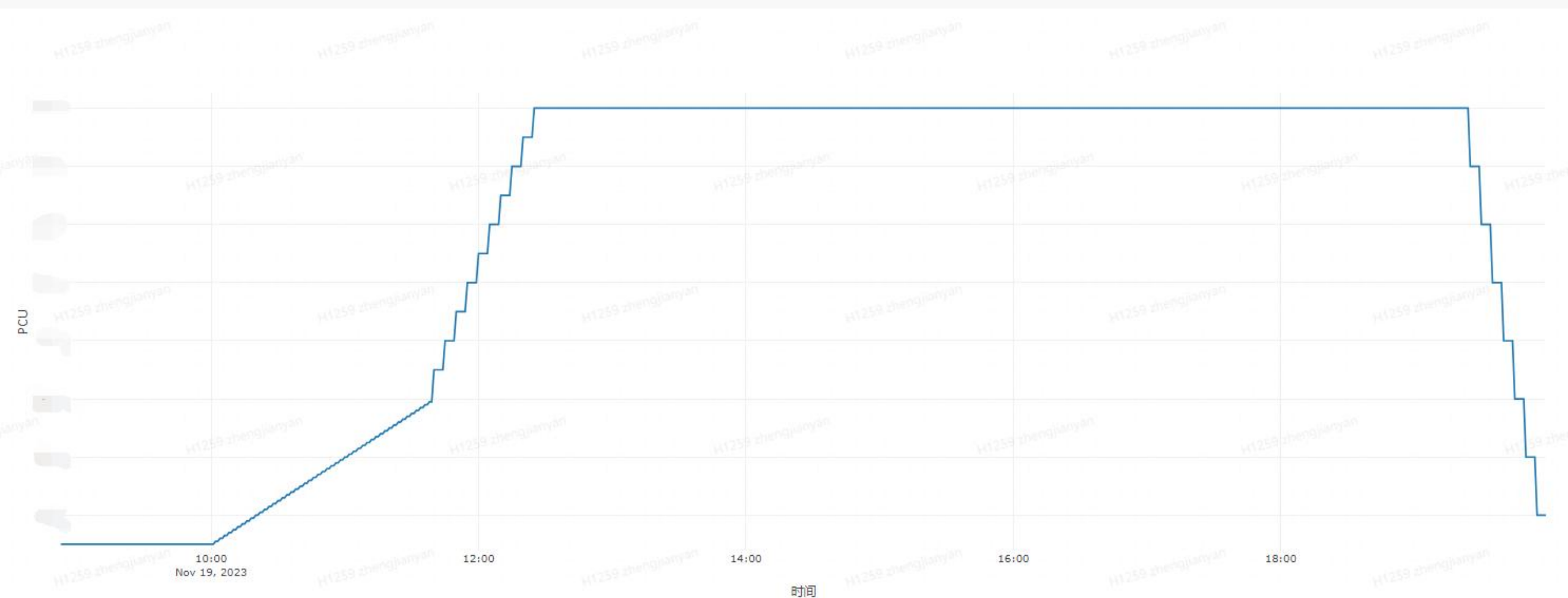
赛前流量暴增

比赛开始时刻



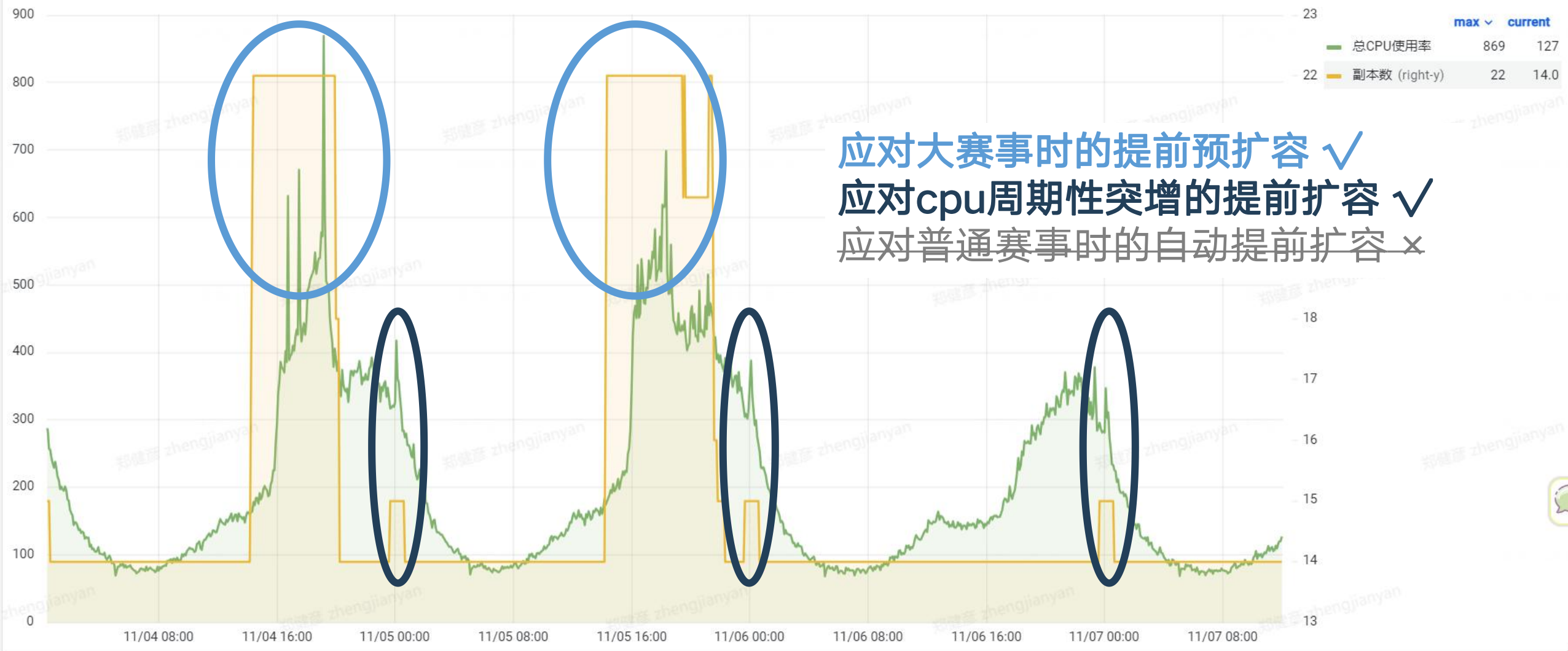
核心实践3 - 基于容量模型的赛事弹性

配置预估的PCU



核心实践3 - 基于容量模型的赛事弹性

应对cpu的赛事突增 & 周期性突增的提前扩容效果



核心实践3 - 基于容量模型的赛事弹性

应对cpu的赛事突增 & 周期性突增的提前扩容效果

~~应对大赛事时的提前预扩容 ×~~

~~应对cpu周期性突增的提前扩容 ×~~

应对普通赛事时的自动提前扩容 ✓



核心实践3 - 基于容量模型的赛事弹性

赛事容量预估




通过精准预估赛事高PCU时所需资源，可以看资源是否充足，且提前和云厂商报备

核心实践3 - 基于容量模型的赛事弹性

容量&准确率巡检

核数将会增长的未预扩容的应用（未开启弹性，未开启AI-HPA，已开启k=0等）

应用名	低峰期核数	预估核数	增长核数
leafgz-serverlessnode			4384.0
leaf-serverlessnode			2404.0
huyamlsz-videorecomengineserver			1080.0
ovms			888.0

最大副本数不足应用

应用名	当前最大副本数	需要副本数	差
ocr			60.0
huyamlsz-featuregeneratorserver			54.0
hymlmodel-video-main-test5			47.0
huyamlsz-recommendeddagliveserver			47.0

核心实践3 - 基于容量模型的赛事弹性

容量&准确率巡检

根据PCU扩容应用

此准确率不适用于非赛事日期

实际使用CPU



预估偏大CPU



预估偏小CPU



准确率

82.22%

预估偏大

应用名(id)	是否开启AI-HPA	最大PCU	cpu实际	cpu预估	预估-实际	预估/实际
dcache-huya5webproxyserver (22695)	True	-1			5.66	326.37%
huyasz-aillivecoverserver (21889)	True	-1			7.02	283.64%
huyasz-itemedgeserver (1843)	True	-1			1.83	263.09%

预估偏小

应用名(id)	是否开启AI-HPA	最大PCU	cpu实际	cpu预估	预估-实际	预估/实际
huyasz-tvstationserver (13883)	True					23.27%
huyasz-livehousetabmanageserver (20143)	True	-1				23.68%
huyasz-broadcastserver (3553)	True					26.12%
thrift-financecheck (8531)	True	-1				26.52%

核心实践3 - 基于容量模型的赛事弹性

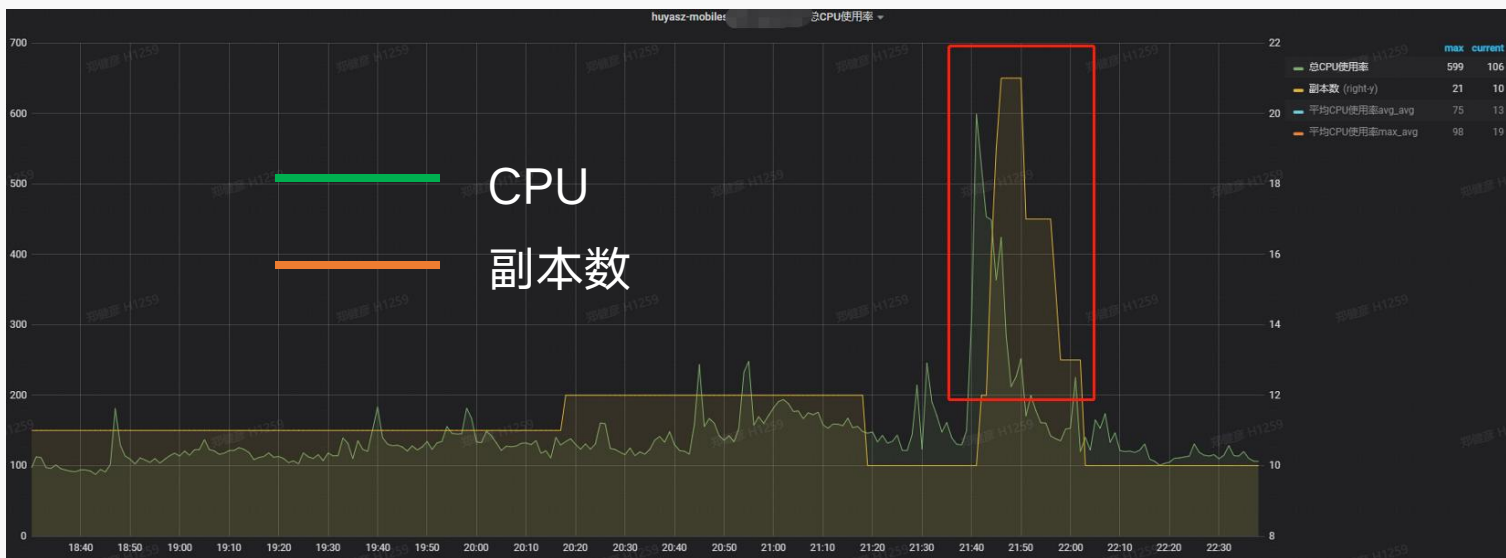
容量&准确率巡检

- 可视化调整模型
- 确保整体服务容量模型准确率 > 85%



反例：

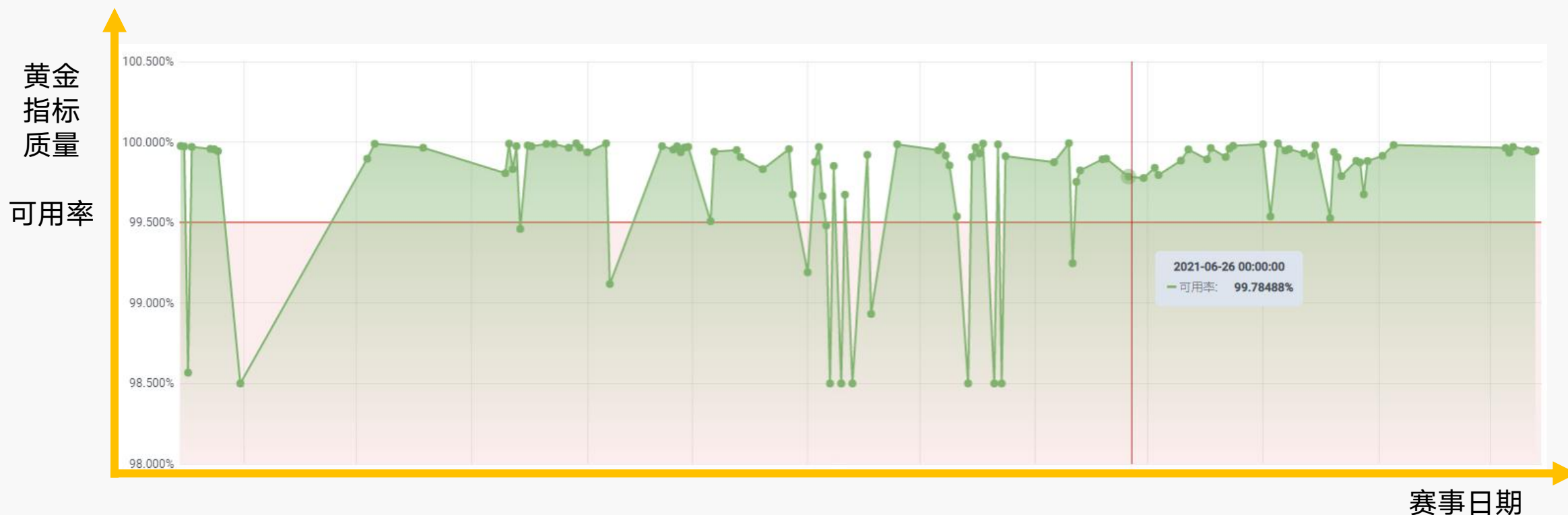
(无容量模型)
or (容量模型不准)
导致扩容滞后
→ 服务过载
→ 影响用户



成果展示3 - 基于容量模型的赛事弹性

红色竖线处上线了赛事模式，整体的黄金指标质量开始变好

可用率：晚高峰时所有黄金指标成功率的最小值的平均（分钟粒度）



成果展示3 - 基于容量模型的赛事弹性

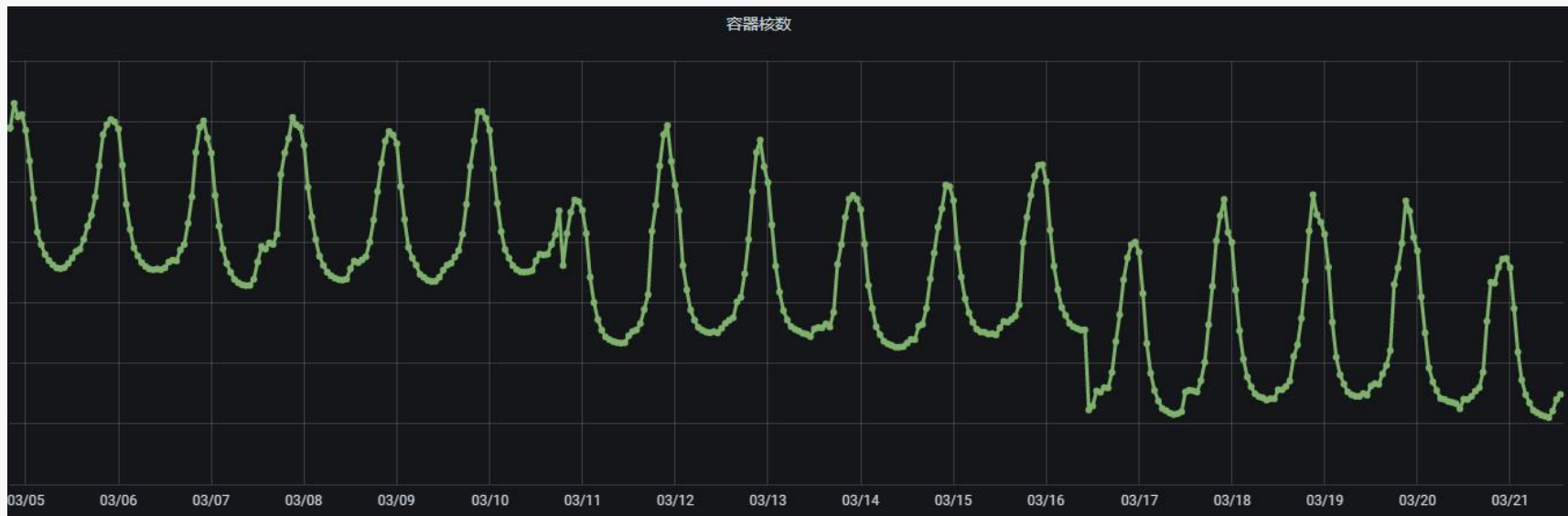
AI-HPA & 容量模型如何影响应用部署架构 & 节省成本:

1. 改变了赛事时中心机房部署架构

增加SET模式 -> 固定SET+AI-HPA赛事模式弹性
成本收益: 节省SET内非弹性应用的算力

2. 改变了日常时的应用架构

日常SET数量缩小一倍



边缘算力智能弹性

核心实践4 - 边缘算力智能弹性

边缘算力背景：

- 边缘算力特点：机器按天/月收费 or 按小时计费但手续费很高
- 虎牙部署在边缘机房的业务：信令，音视频相关服务（转码，P2P）等
- 信令，音视频P2P是高观众PCU相关的业务，转码服务是高主播PCU业务

痛点&问题：

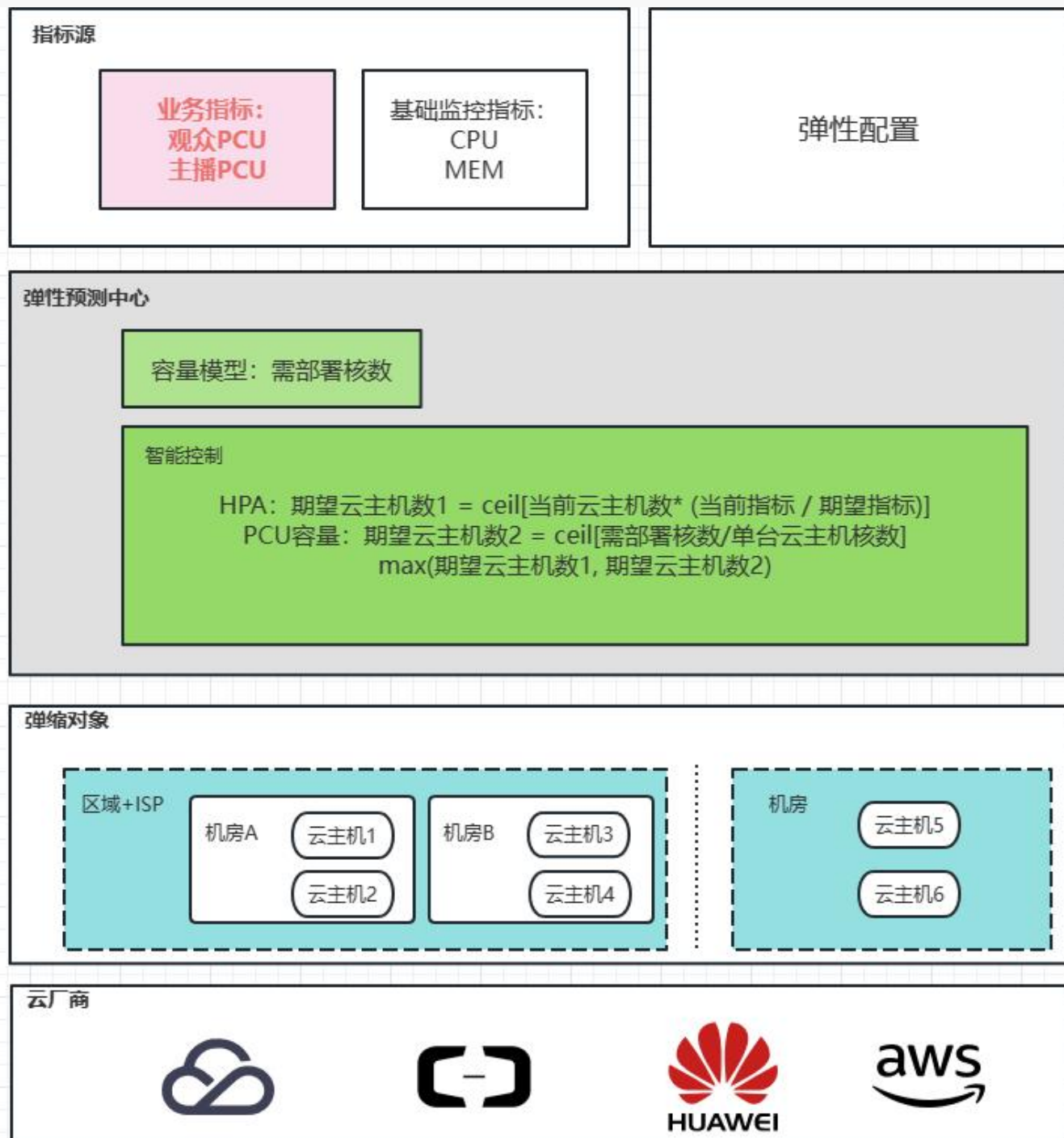
- 边缘算力保持在一个比较高的水平，利用率低，浪费资源
- 面临高PCU时资源预估不准

核心实践4 - 边缘算力智能弹性

整体架构

弹性思想：

- 实时监控指标作为兜底（CPU使用率）
- 业务指标弹性为主
 - 信令&音视频P2P -> 观众PCU
 - 音视频转码 -> 主播PCU
- 按天扩缩



核心实践4 - 边缘算力智能弹性

信令业务案例

$$\text{核数} = k * \text{PCU} + b$$

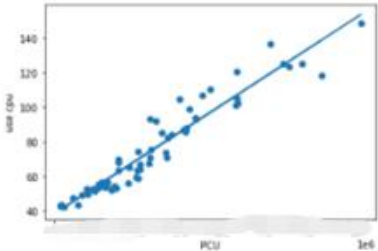
数据准备

每天聚合出每个机房的有效使用核数和当天的PCU

日期	使用核数	PCU	category	ldcName
2022-08-31	52.940		Tel-HD-NT	虎牙-网宿江苏南通电信-01
2022-09-01	125.146		Tel-HD-NT	虎牙-网宿江苏南通电信-01
2022-09-02	120.491		Tel-HD-NT	虎牙-网宿江苏南通电信-01
2022-09-03	123.384		Tel-HD-NT	虎牙-网宿江苏南通电信-01
2022-09-04	124.945		Tel-HD-NT	虎牙-网宿江苏南通电信-01

模型拟合

以PCU为自变量，机房的使用核数为因变量，做一元线性回归



信令服务是用于用户连接的，其资源使用与PCU强相关

保存模型参数

将机房的线性回归模型的参数斜率 $k \times 1e6$ 和截距 b 保存到数据库

category	k	b
0 Tel-HD-NT	31.9903	-10.1858

k表示每百万PCU需要多使用的核数

未来PCU预估

根据LPL和KPL赛程，预估未来一段时间的PCU

start_time	end_time	pcu
2022-09-01 09:00:00	2022-09-05 12:00:00	



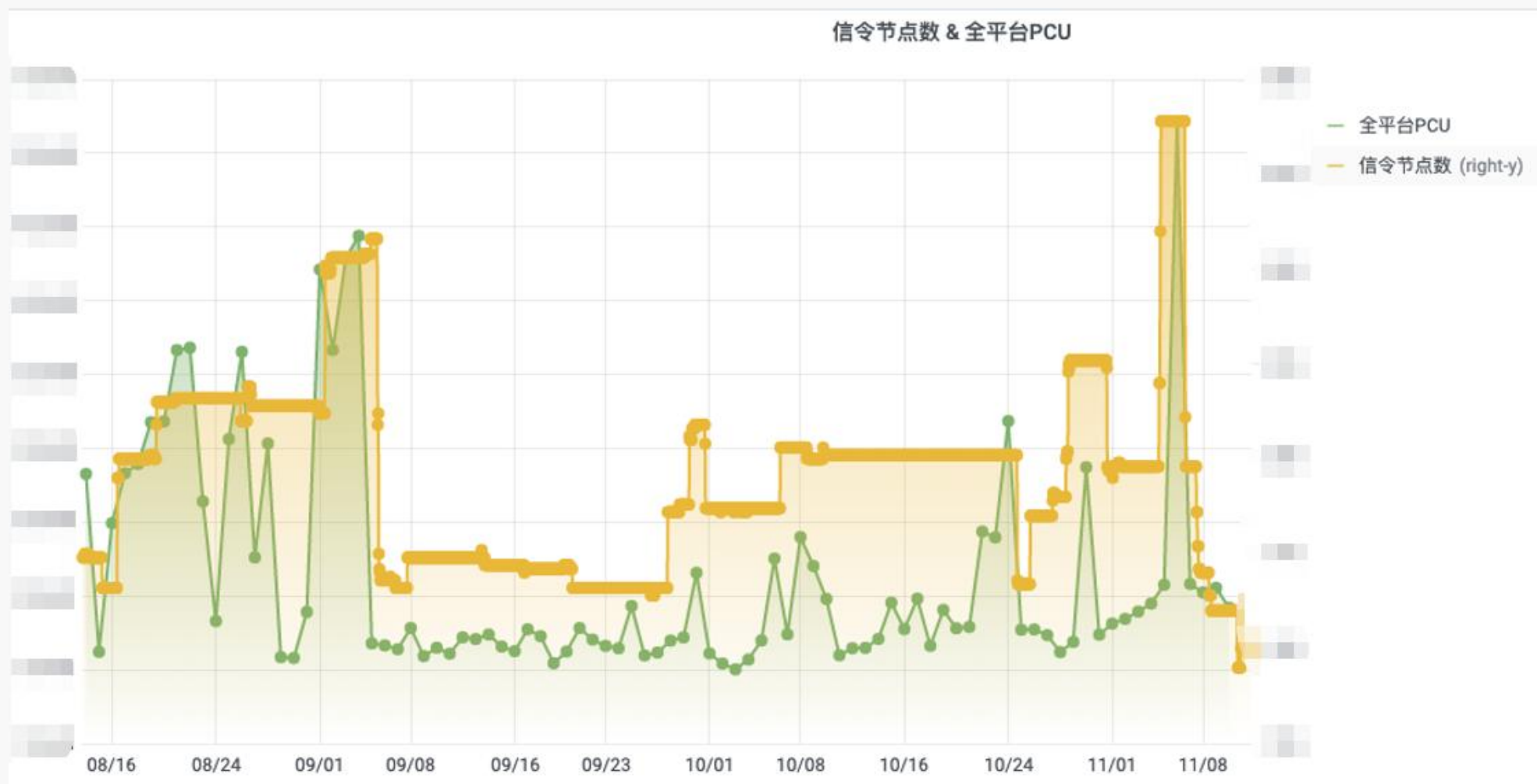
机房需要节点数计算

需部署核数 = $(k * \text{预估PCU} + b) / \text{目标CPU使用率}$

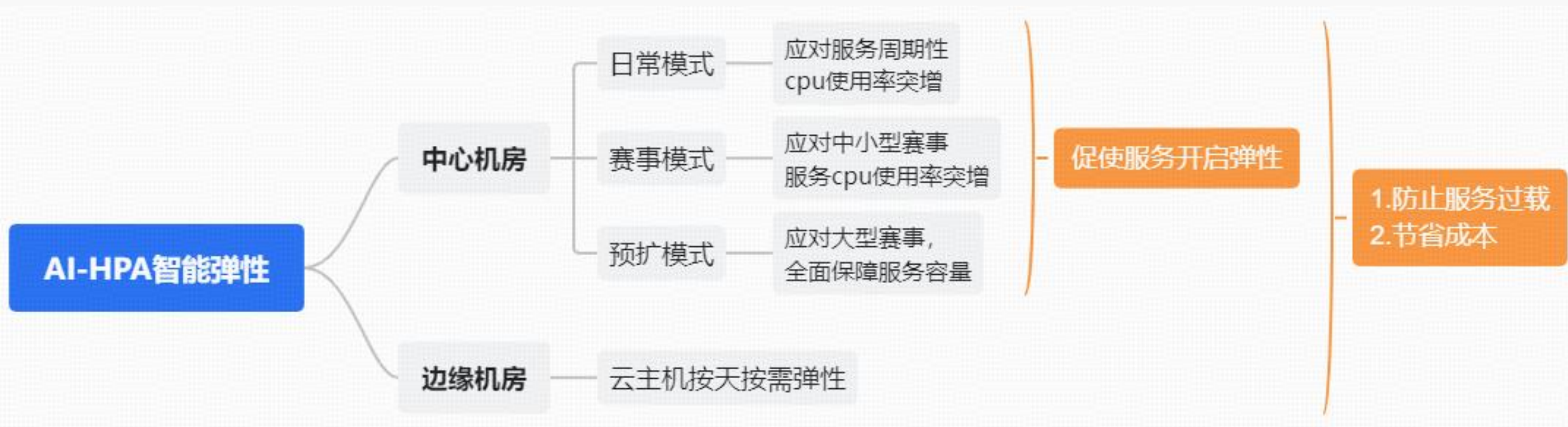
需部署节点数 = $\text{ceil}(\text{需部署核数} / \text{单节点核数})$

成果展示4 -边缘算力智能弹性

信令业务边缘算力智能弹性上线后效果图



小结





微信官方公众号：壹佰案例
关注查看更多年度实践案例