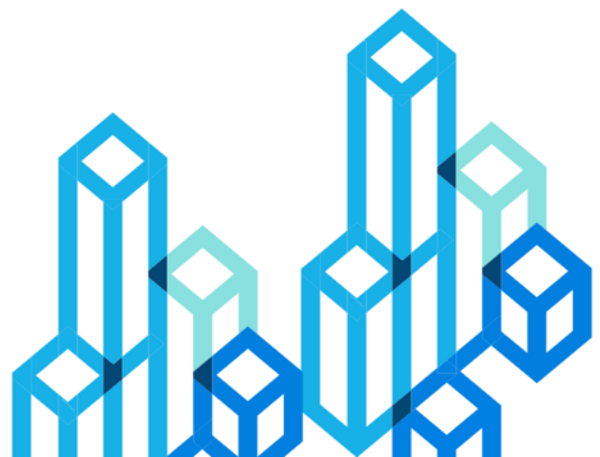
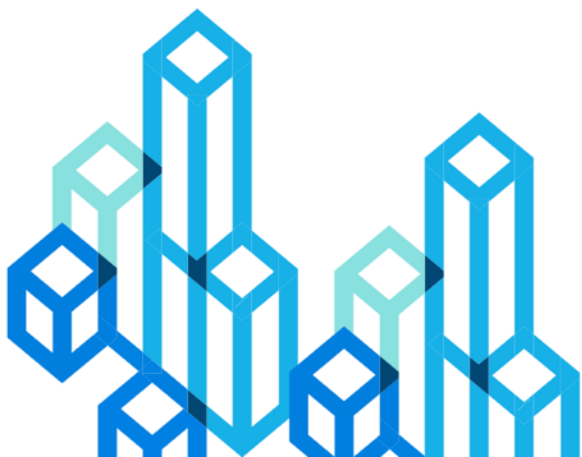


虎牙大数据融合云实践





姓名： 陈仕明

当前工作： 虎牙计算平台负责人

过往经验： 在数据领域精耕十余年，从传统企业的数仓，到互联网大数据。熟悉数仓建模，以及分布式存储/计算的原理及实现

集群搬迁

大数据集群搬迁，不再需要拉上数据开发同学折腾半年，数据平台自行一个月搞定？

资源成本

大数据作为公司重点资源投入，虎牙40%的成本节省如何做的？

云上优势

放下大胆的使用云上的技术，不好用时随时可下可迁！



场景问题



解法思路及落地效果

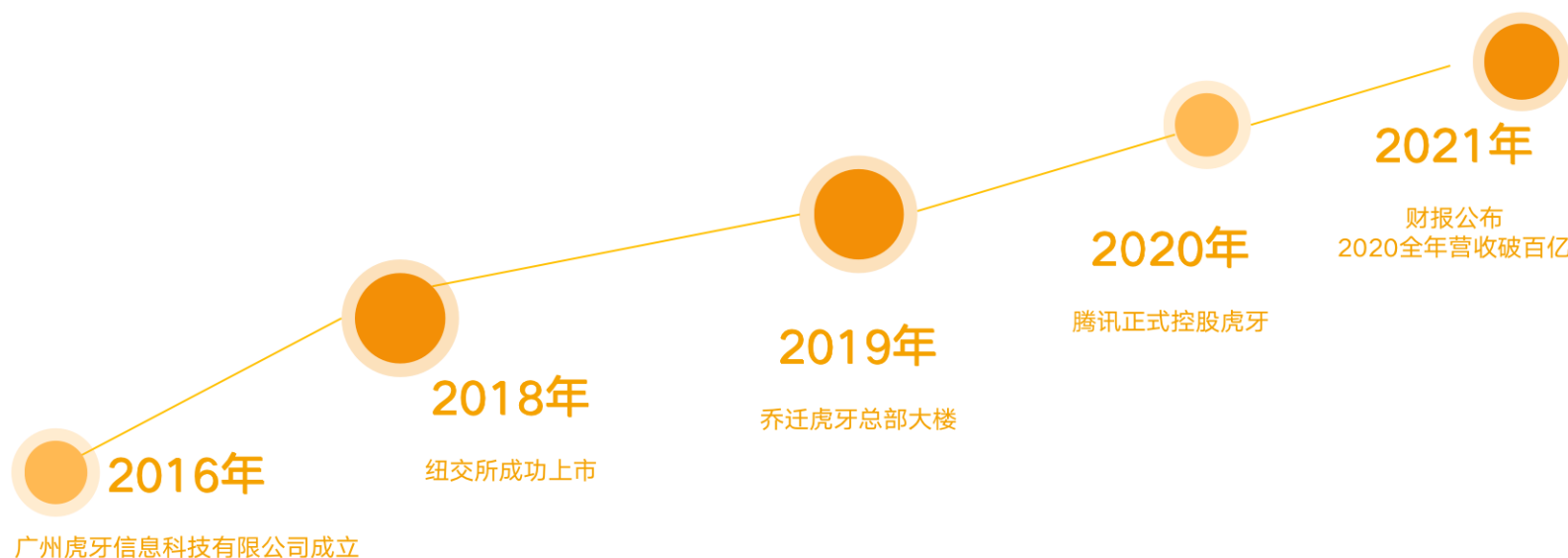


核心架构及关键实现



未来展望

虎牙直播是以游戏直播为核心业务，致力打造全球领先的直播平台，涵盖游戏、娱乐、综艺、户外、美食、体育等多元化内容，覆盖超4000款游戏，国内平均MAU突破1.78亿，移动端MAU达7550万。





用户 体验

架构透明：资源在哪儿，用什么系统实现的，对我完全透明，特别是机房迁移，不要来骚扰我！

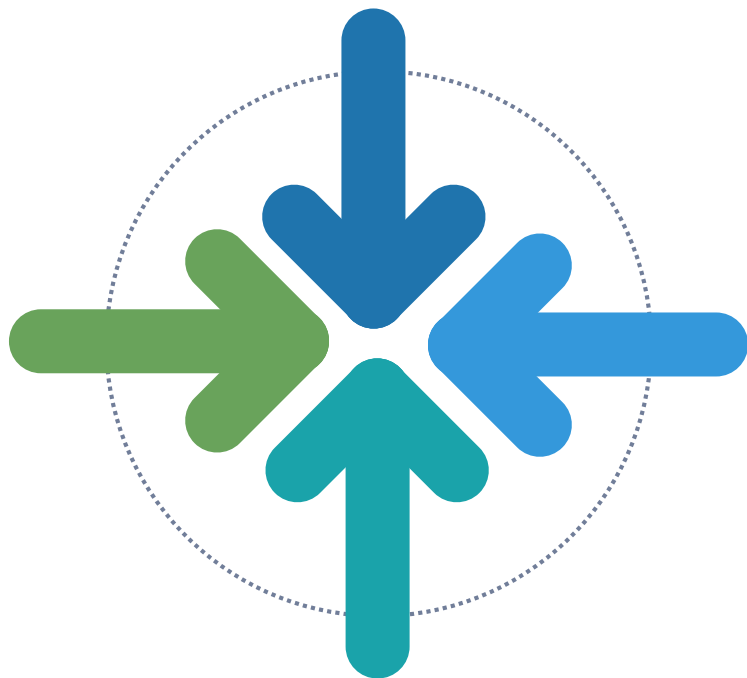
容量保障：需要算力时，很快就能够交付，我愿意花钱，你就给量！



老板 成本

较低的资源利用率：公司所有服务器上全天利用率才20%，大量闲置，能不能用起来？

云产商能力：云产商号称的好东西，能不能去试一下，但千万不要产商绑定！



混部

通过容器离在线混部，充分使用在线业务的闲时算力



多机房

在多个机房交付大数据的算力和存储



上云

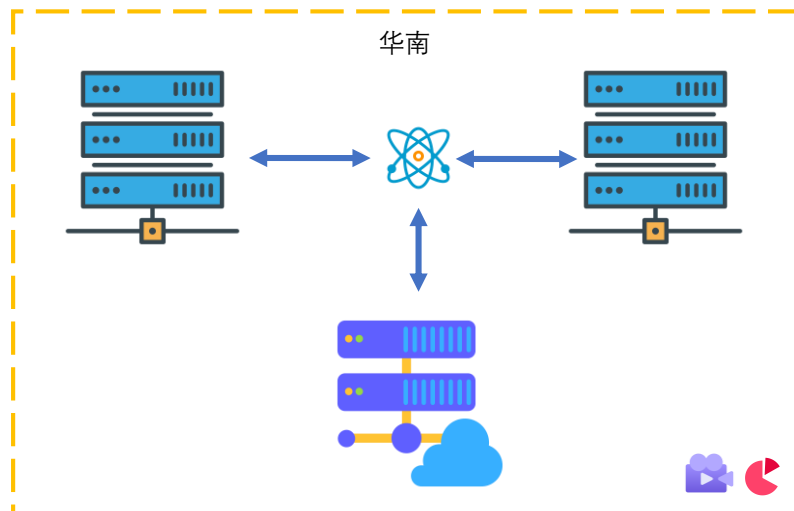
和云上打通，需要紧急算力时，通过云上扩缩



用户透明

P层平台对用户透明，用户对多机房，云上云下无感知

集成IDC和公有云，结合各自优势的融合云大数据架构



机房分布

在线业务两地三中心部署，核心在华南IDC，机房间百G带宽互通

算力分布

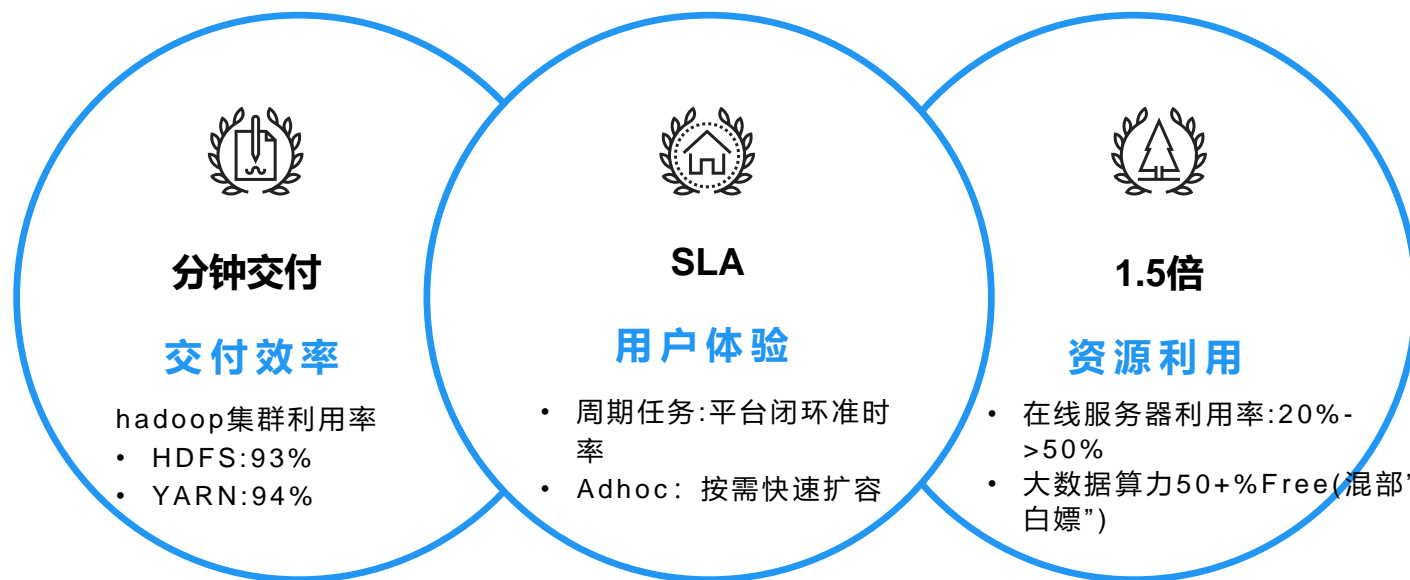
大数据算力和在线业务全部采用容器混部在三个核心机房，分布情况5：4：1(云)，其中弹性部分上云

存储分布

大数据存储分布在三个核心机房，分布4：2：4(云)，其中冷数据在云上

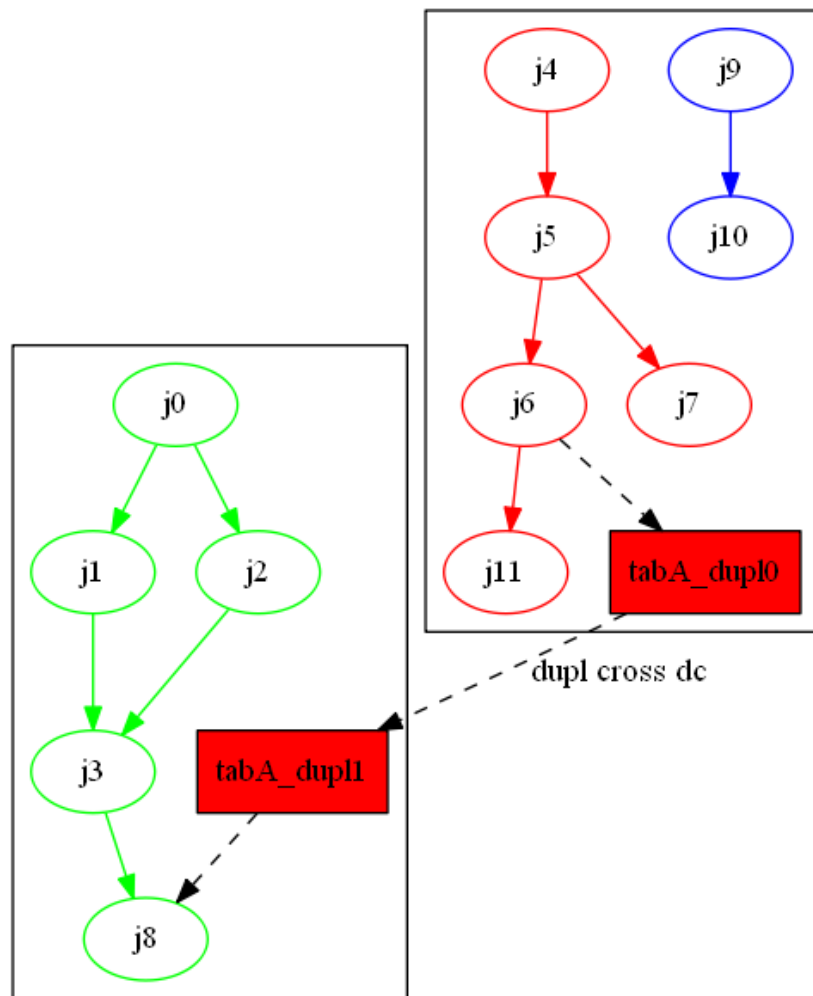
调度效率

大数据的周期任务分钟级跨机房迁移
adhoc任务根据算力和数据分布情况自动路由



大数据综合成本降低:40+%

- 绝大部分的周期任务，只会用到当天的新数据
- AdHoc查询集中在少数表上



- 多机房内独立部署大数据集群
- 将DAG任务使用算法分簇，以减少簇间流量为目标
- 以簇为单位将任务调度到合适机房执行
- 跨簇的数据在机房间副本冗余存储，任务默认就近读写本机房存储，减少跨网传输



任务调度过去了，数据不迁移？

不涉及历史数据的迁移，因为绝大部分任务都是读当天新增数据，簇内任务读写的新增数据都在任务所属机房



如此错综复杂的周期任务如何做自动拆分？



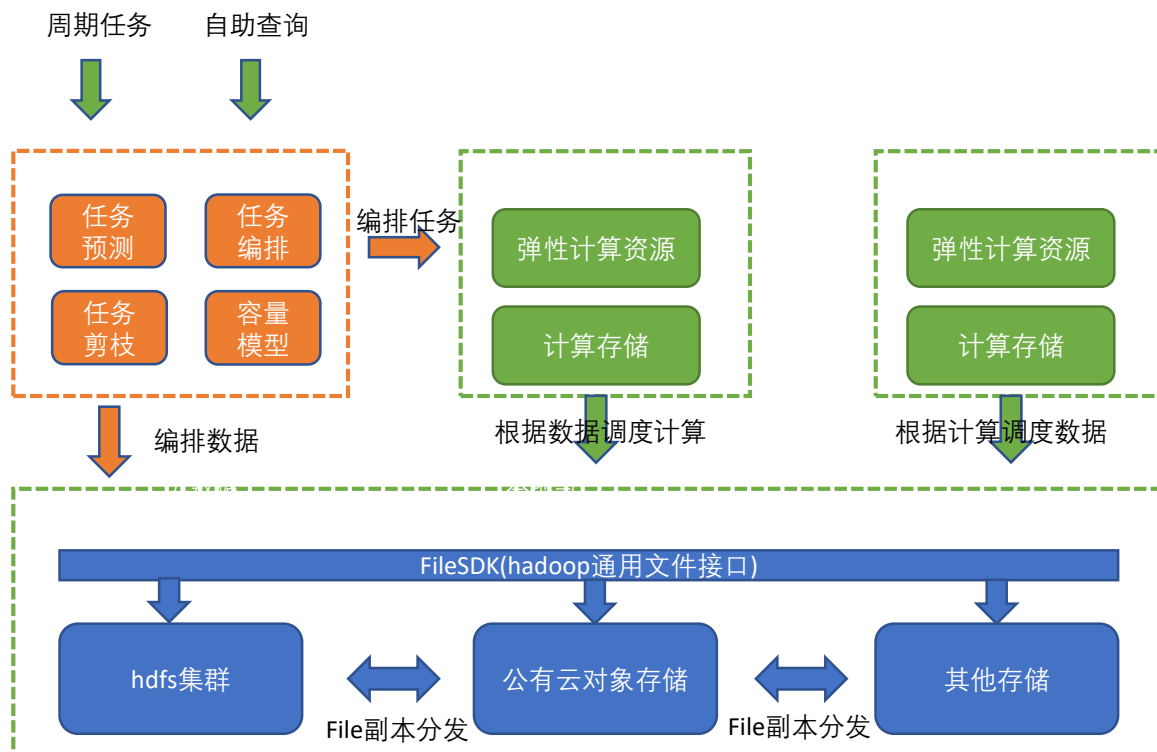
数据怎么做到云上云下的多个机房内可用？



如何降低跨机房之间的专线带宽依赖？



云上算力按量交付，如果做到算力容量精准？



编排调度系统

以算力和跨网流量为目标，针对任务和数据进行基于透明编排，提高YARN集群的资源利用率

计算系统

分机房独立部署计算集群，通过离在线混部的方式提高物理服务器的资源利用

存储系统

对用户屏蔽文件在多个机房内的读写，充分考虑跨网的流量带宽



融合云存储

多机房下的数据读写

任务编排/调度

快速透明化的调度任务

跨机房网络

网络隔离

云上大数据

云上基础设施差异性对大数据的影响

算力精准扩缩

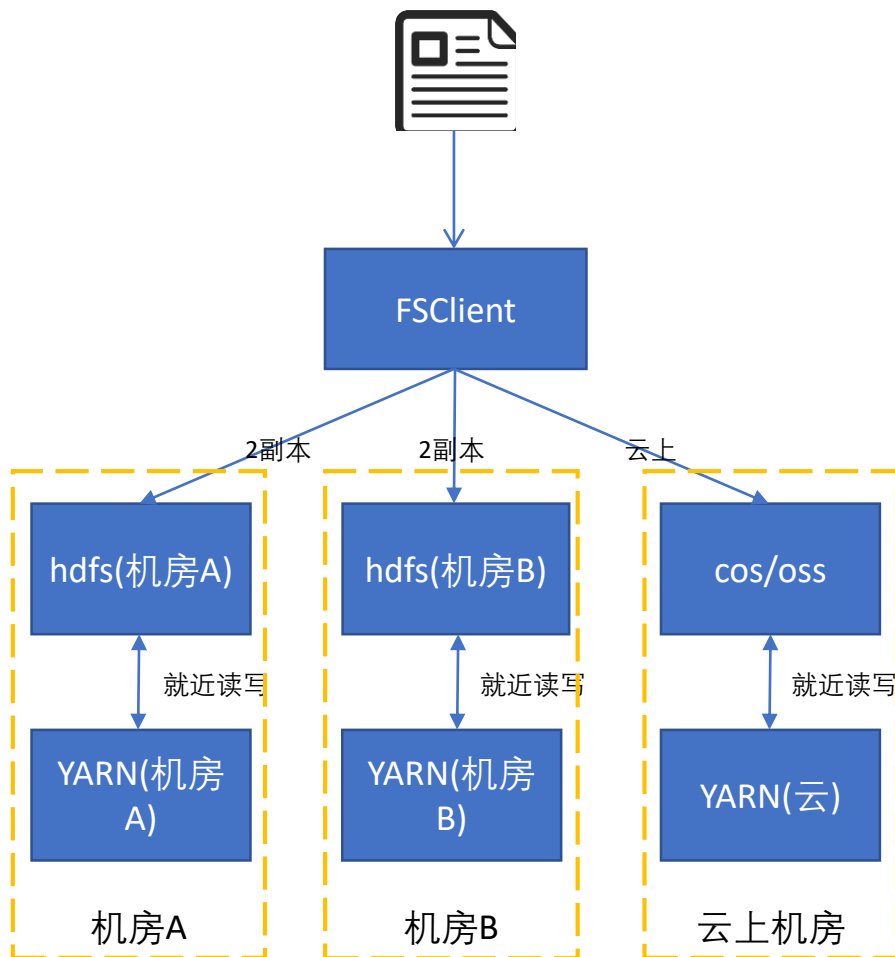
- 避免算力挤兑
- 算力预测

混部下的大数据

- 驱逐、YARN的动态超卖
- 任务+算力的画像

关键目标：尽量降低跨机房带宽的前提下，解决数据在多个机房内的数据可用性

hyfs://xxxnamenode/path/000.orc



兼容

极少实施成本

- **使用方**：client端使用Hadoop接口,改URL即可，对hive完全透明，修改hive meta location url
- **历史数据**：使用hdfs namenode存储元数据，hdfs中的旧数据不需升级；对象存储中的数据支持import重建元数据，无需数据拷贝
- **无额外系统**：backend沿用已有存储引擎，

异构异地多副本

- **副本异构**：以文件粒度存储于多backend fs中
- **一致性**：复用backend引擎的checksum算法，存储在元数据中

跨机房网络

- **就近读写**：自动选择离client网络最近的backend进行读写，降低带宽消耗
- **复制任务统一协调调度**：进行复制任务的限流，降级，优先级控制

使用举例

engine:

instances:

```
-
  type: oss
  code: shanghai
  region: shanghai-ali
```

```
-
  type: cos
  code: guangzhou
  region: guangzhou-tx
```

```
-
  type: cos
  code: guangzhou-path
  region: guangzhou-tx
```

```
-
  type: hdfs
  code: gz-nx
  region: guangzhou-nx
```

```
-
  type: hdfs
  code: gz-ly-ec
  region: '92093'
```

network:

bandwidth:

```
-
  pointLine: 92093 <-> guangzhou-tx
  size:
```

```
-
  pointLine: 92093 <-> guangzhou-nx
  size:
```

```
-
  pointLine: guangzhou-tx <-> guangzhou-nx
  size:
```

```
-
  pointLine: guangzhou-tx <-> shanghai-ali
  size:
```

```
-
  pointLine: 92093 <-> shanghai-ali
  size:
```

```
-
  pointLine: guangzhou-nx <-> shanghai-ali
  size:
```

backend集群实例

跨机房双向带宽

在IDC上传文件，数据被存储在idc的backend集群中

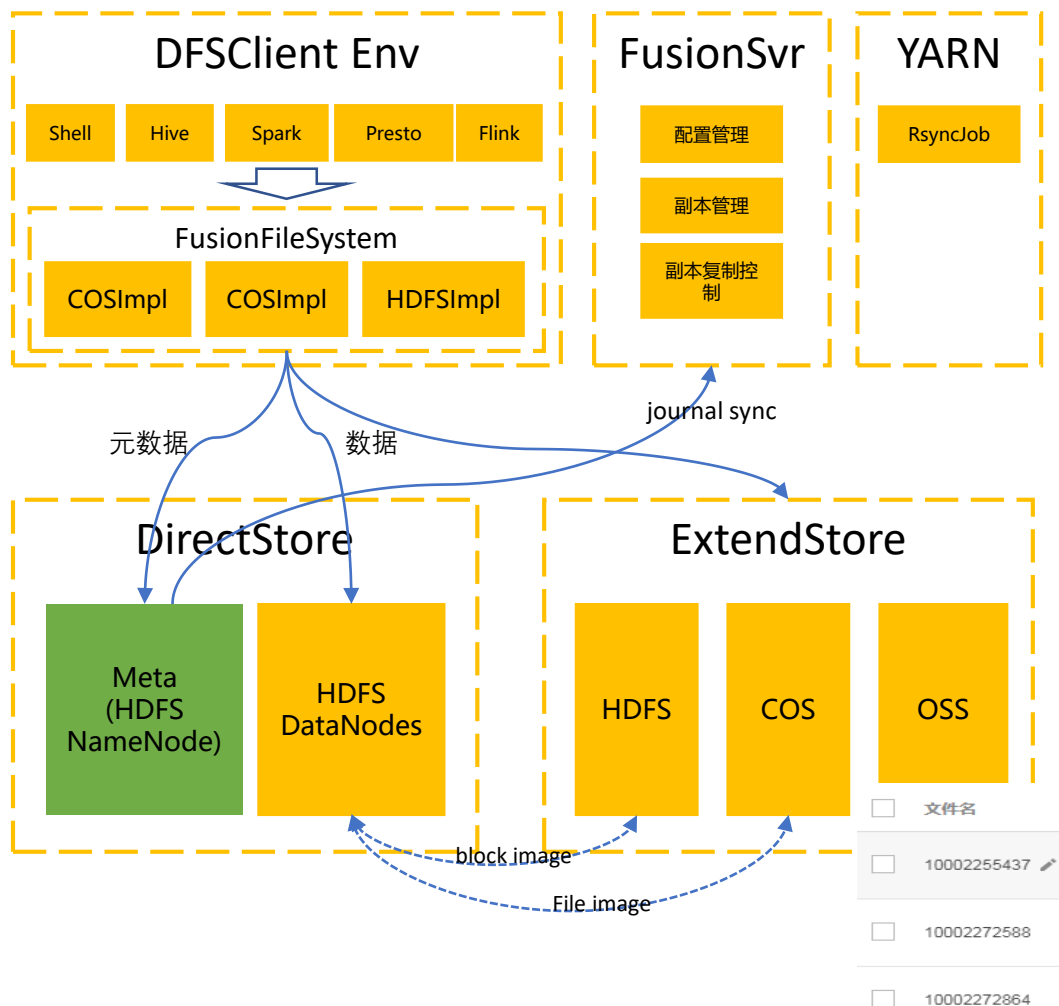
```
/data/apps/hadoop/bin$ ./hadoop efs -put /tmp/a.log hyfs://huyardcluster/tmp/a.log
stan@ip-10-64-136-16:/data/apps/hadoop/bin$ ./hadoop efs -lsDupl hyfs://huyardcluster/tmp/a.log
-rw-r--r-- 3 8075294749 hadoop hive 145 direct huyardcluster 2021-05-24 18:21 hyfs://huyardcluster/tmp/a.log
```

在腾讯云上传文件，数据被存储在cos中

```
ata/apps/hadoop_tx/bin$ ./hadoop efs -put /tmp/a.log hyfs://huyardcluster/tmp/b.log
lata/apps/hadoop_tx/bin$ ./hadoop efs -lsDupl hyfs://huyardcluster/tmp/b.log
-rw-r--r-- 3 8075339209 hadoop hive 145 cos~ guangzhou, 2021-05-24 18:24 hyfs://huyardcluster/tmp/b.log
```

手动针对特定文件增加一个副本冗余

```
:/data/apps/hadoop/bin# ./hadoop efs -syncDupl cos guangzhou hyfs://huyardcluster/tmp/a.log
21/05/24 10:00:10 INFO CLIENT: EXECUTING HADOOP SYSTEM: THROUGH SUCCESS, UNDOING SUCCESS, UNDOING SUCCESS, UNDOING SUCCESS
-rw-r--r-- 3 8075294749 hadoop hive 145 cos guangzhou,direct huyardcluster, 2021-05-24 18:21 hyfs://huyardcluster/tmp/a.log
stan@ip-10-64-136-16:/data/apps/hadoop/bin#
```



核心实现

- **扩展性:** 存储分为Direct/ExtendStore, 原本数据存储的hdfs集群为DirectStore, ExtendStore通过Adaptor可扩展。历史数据不用升级
- **可靠性:** 元数据存储在与DirectStore hdfs namenode的NodeXAttr中, 复用hdfs的HA
- **性能:** 通过backend存储引擎的hdfs读写, 无IO性能的降低; 元数据存储在hdfs namenode中, 大部分数据存储在DirectStore中, 无额外对namenode的性能消耗
- **数据一致性:** 使用nodeid作为extendstore中文件名称, 解决文件“重新生成”场景下的一致性; 不支持file append, 读取数据时再次做checksum, 防止副本之间内容不一致
- **跨机房的有限网络:** 副本采用异步复制, 根据任务优先级和降级策略, 机房间带宽上限, 由FusionSvr统一调度

虎牙任务现状

- **任务多**: 每天X0W的周期任务实例数
- **流量大**: 计算产生的内网流量每天XPB级, 峰值带宽XTb; 计算产生的hdfs读写操作X0PB

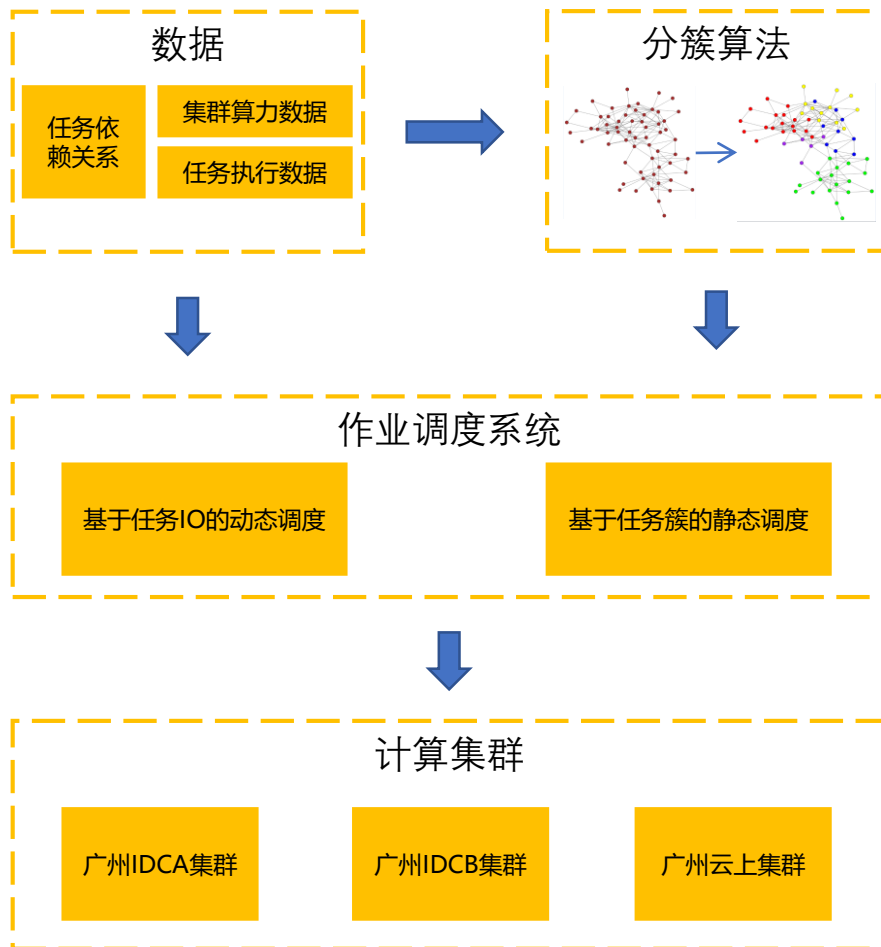


系统设计目标

- 数据和任务迁移, 对用户透明
- 尽量降低对跨机房带宽的要求
- 算力尽量避免空洞, 造成浪费

如何自动选择合适的任务
调度到其他机房?

关键目标：透明，算力利用率



◆ 数据

- **关系:**任务依赖关系
- **权重:**任务输入输出大小、任务算力大小
- **时间:**任务执行时间点

◆ 分簇算法

- 基于图的社区发现算法
- 模型天级别动态更新，尽量保持任务簇稳定

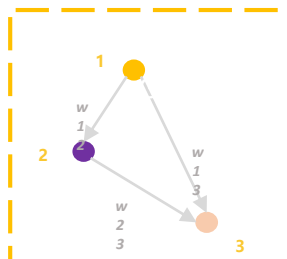
◆ 作业调度

- **静态调度任务:**不定期选择合适的任务簇，指定其执行机房；
- **动态调度:**根据任务的历史IO数据，结合当前集群算力空闲情况，动态调度小IO任务，填补集群的算力空洞

原数据

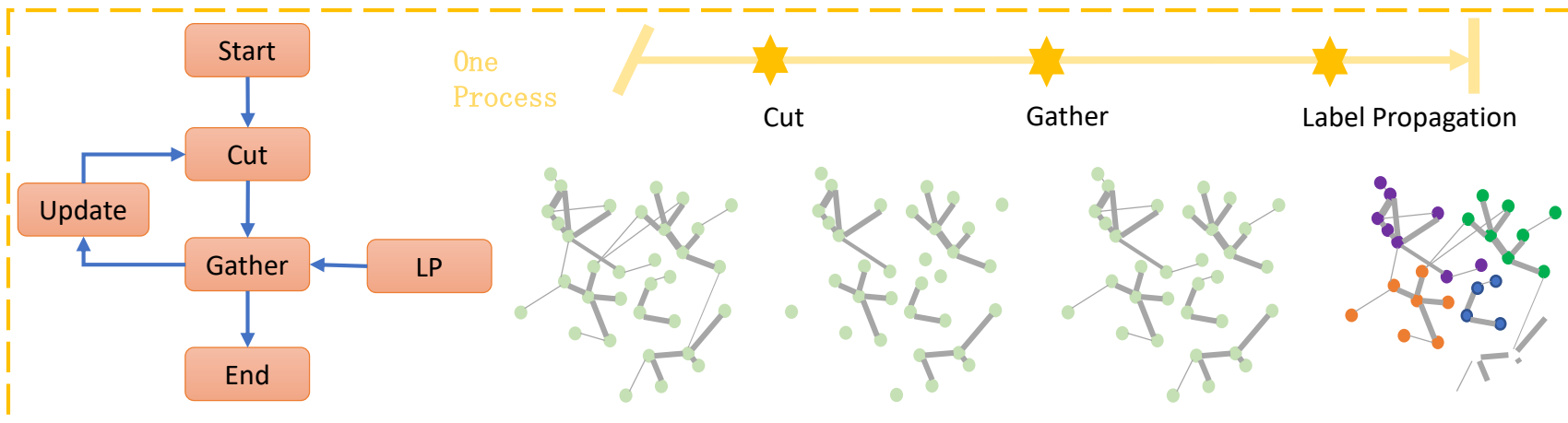
- 任务依赖关系
- 任务读写表及数据大小
- 任务算力大小
- 表与权重的映射关系

关系图



- 节点:任务
- 边:任务依赖关系
- 权重:任务读写数据大小

图分簇切割



簇元数据

- 簇ID
- 簇内涉及的表和任务的数量及明细
- 簇上按小时分布的算力大小
- 簇内部流量大小
- 簇外交换的流量大小

1: 拆分出300+任务簇，最大的簇算力占比30%，任务数占比19%;

island_id	size	include_tables_num	in_Bytes_str	out_Bytes_str	source_Bytes_str
11662	173	227	781.75 GB	3.69 GB	29.69 GB

- 每个任务簇与外部交换的流量，意味着可能带来的跨机房传输
- 簇内源头任务，意味着对应源头数据的实时任务需一并迁移

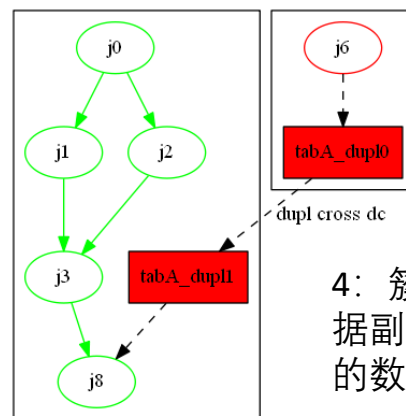
2: 每个任务每个小时消耗的算力情况，用于评估算力需求

task_id	h00	h01	h02	h03	h04	h05	h06	h07	h08	h09	h10	h11
52611	0	0	0	0	0	0	0	0	0	1.5114	0.6993	0.268
177953	0	0	0	0	0	0.2368	0.2431	0.3424	0.8865	0.0062	0.0028	0.04
176323	0	0	0	0	0	0.203	0.3281	0.2587	0.2516	0.1674	0.1192	0.094
168031	0	0	0	0	0	0	0	0	1.1047	0.094	0.0743	0.042
177915	0	0	0	0	0	0.0651	0.1399	0.0806	0.0087	0.0529	0.0031	

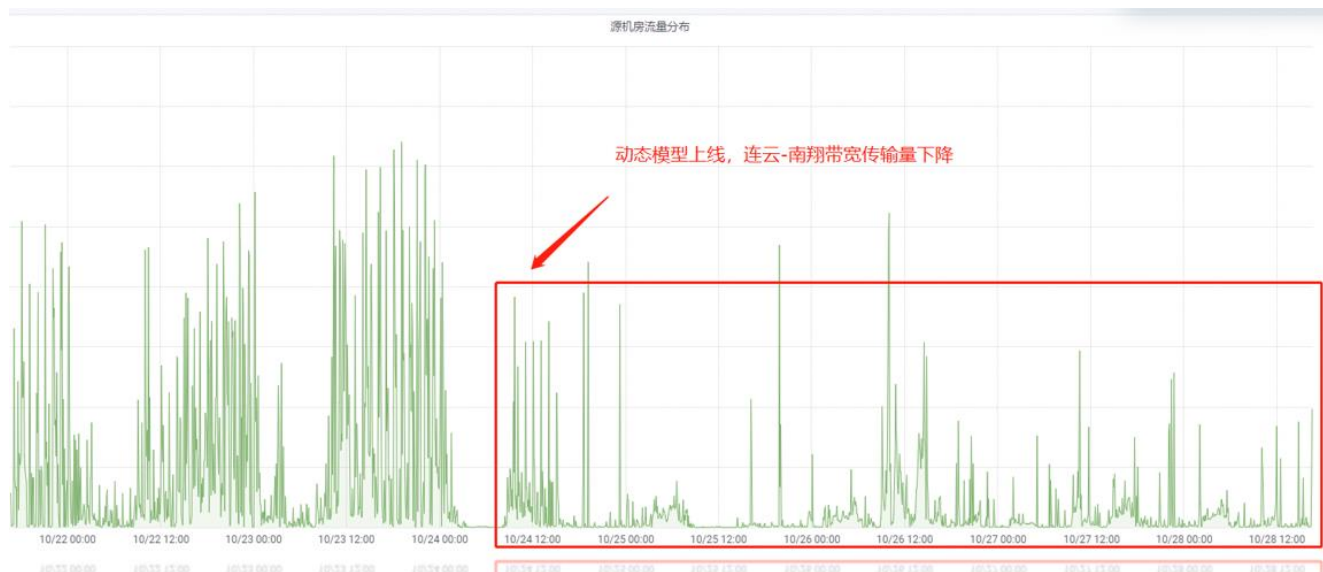
3: 指定需要所选择迁移的任务簇中的任务运行的机房

```
mysql> select * from mixc_task_region limit 10;
```

task_id	region_code	task_date	task_update_time
3015	guangzhou-tx	2021-05-18 10:35:42	2021-05-18 10:35:42
3055	guangzhou-tx	2021-05-18 10:35:42	2021-05-18 10:35:42
3056	guangzhou-tx	2021-05-18 10:35:42	2021-05-18 10:35:42
4055	guangzhou-tx	2021-05-18 10:35:42	2021-05-18 10:35:42
4056	guangzhou-nx	2020-12-21 11:19:36	2020-12-21 11:19:36
4059	guangzhou-nx	2020-12-21 11:19:37	2020-12-21 11:19:37
4061	guangzhou-nx	2020-12-21 11:19:37	2020-12-21 11:19:37
5007	guangzhou-tx	2021-05-18 10:35:42	2021-05-18 10:35:42
5008	guangzhou-tx	2021-05-18 10:35:42	2021-05-18 10:35:42

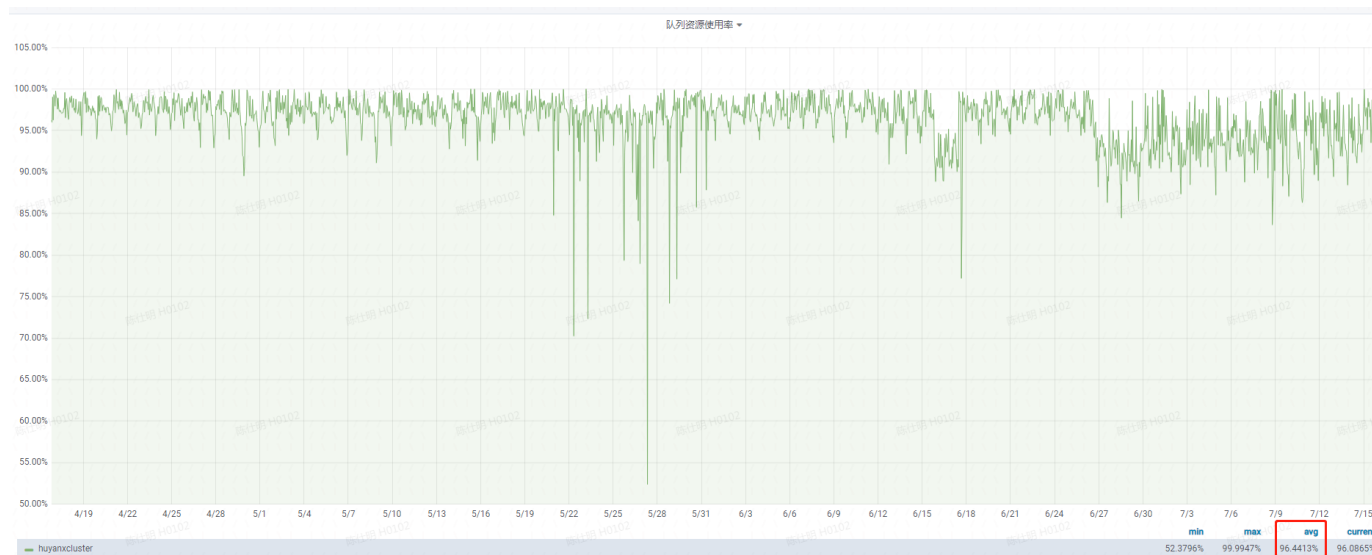


4: 簇外的异地数据副本复制，簇内的数据本地读写



节省区间带宽:

通过任务簇调度优化, 大幅降低跨网流量

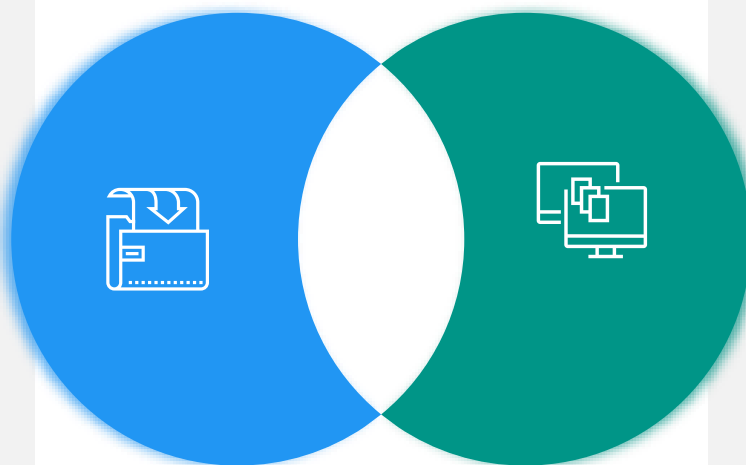


提高集群利用率:

小IO大计算的任务, 很好的填补的算力空洞, 某个YARN集群分配率96%

网络特点

- **容灾冗余：**专线T*2保障在线业务，只用一小半，大半空闲
- **时段错峰：**在线业务具有稳定的高峰时段，外加某个大主播个人突发行为的流量突增
- **离线业务高鲁棒：**大数据业务对网络可用性容忍性高

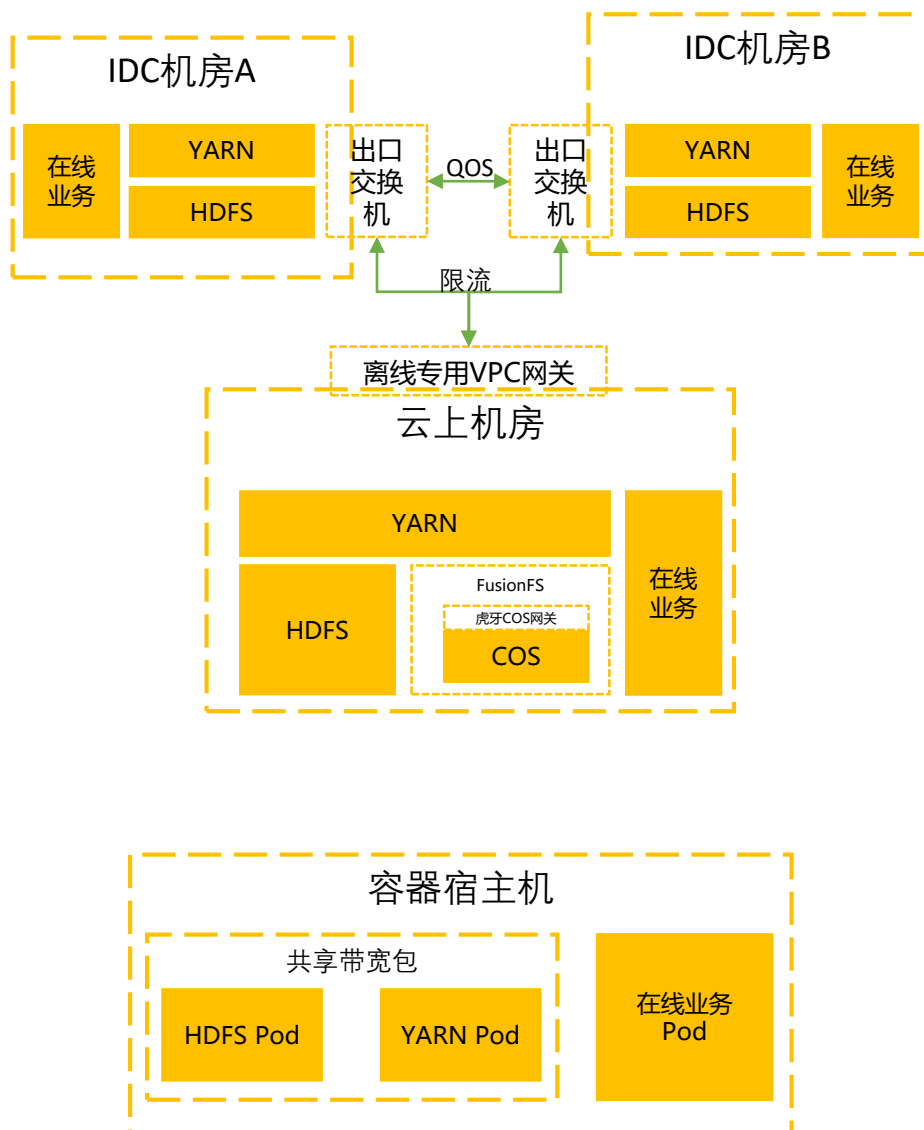


设计目标

带宽可打满：带宽换算力

离线业务让渡在线：在线业务
优于离线业务对网络的使用，**立即实时响应**

离线内部优先级：离线任务之间对带宽使用分优先级，
基于对准时基线和数据价值的权重判定



离线/在线业务隔离

- **IDC机房出口：** 交换机上做QOS，离线业务低优先级
- **云上出口：** 针对大数据所属的COS bucket，cos域名指向到离线场景的专属VPC网关，限流，避免上云专线使用过大
- **主机层：** 离线组件共享带宽包，内部争抢。大数据热点不可控，争抢策略能更好提高网络利用率

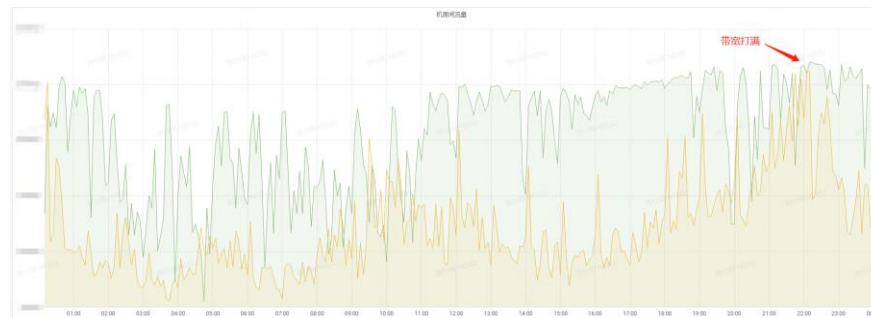
COS网关吞吐

- COS网关按需扩容，产商一般提存储规模要求
- 任务数据通过融合云就近读写云上cos
- 只有计算中间stage数据放hdfs，避免IDC和此hdfs交换数据(否则流量会走在线业务VPC，YARN节点因为混部，所以和在线同一个VPC)

专线网络



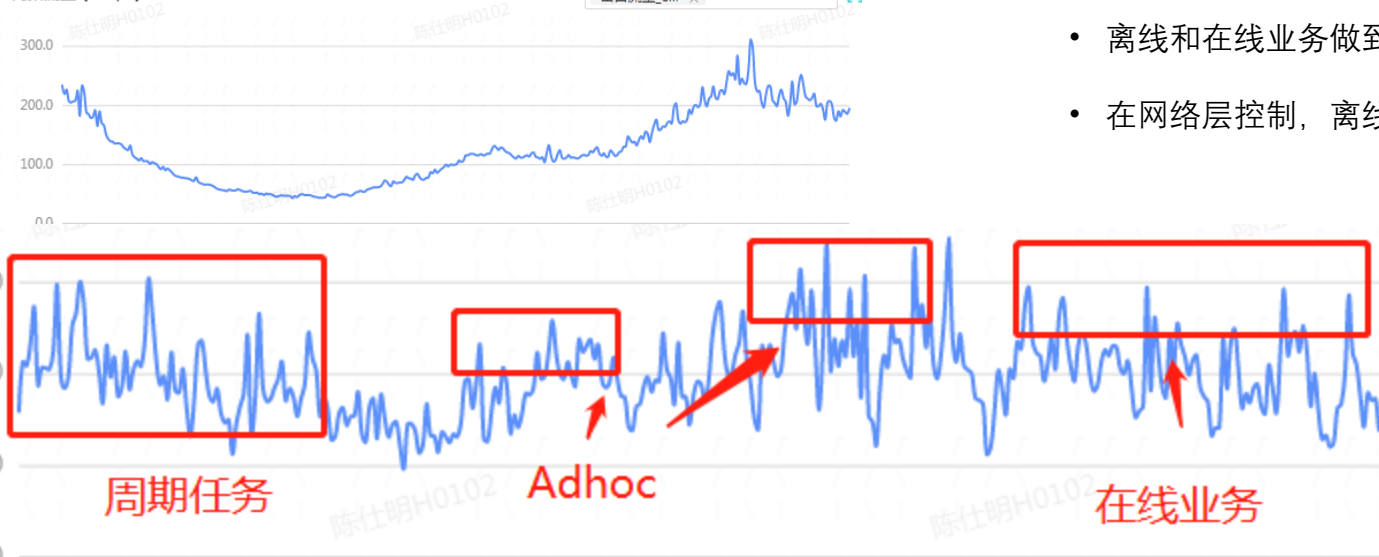
没有大数据的专线网络



叠加大数据之后的专线网络

主机网络

网络流量 (MB/s)




- 较好的拉高了机房和主机的带宽利用率
- 离线和在线业务做到了很好的影响隔离
- 在网络层控制，离线业务放心大胆使用带宽

云上优势

- **廉价资源**：归档存储，竞价实例，ARM算力等
- **云原生**：存储计算分离，按需弹性
- **灵活的资源策略**：如资源置换，GPU换成CPU，T4换V100

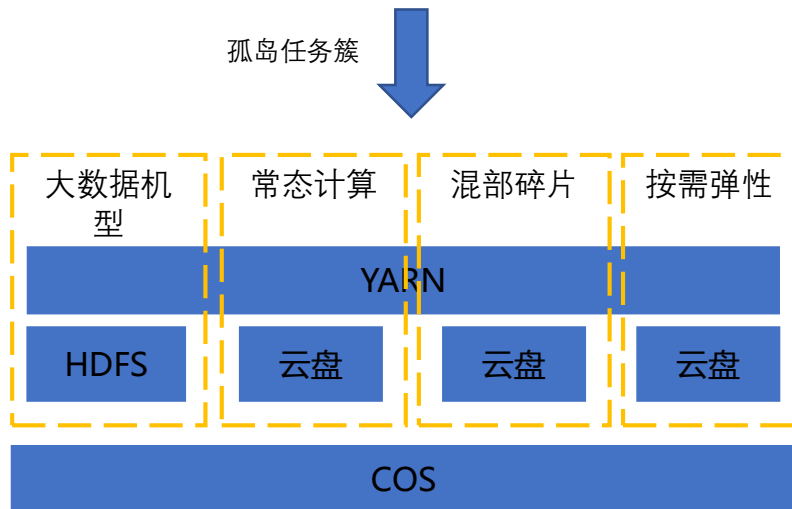


 随时上云/下云/迁云，部分上云

问题担忧

- **实施成本/风险高**：大数据机房迁移动辄半年，不能回退，决策风险很大
- **下云迁云成本高**：上了之后下不来，厂商/架构绑定

公有云定位：常量IDC，弹性上云



算力

- **常态算力**：云上hdfs物理节点上的YARN算力，按月购买的计算型云上实例
- **混部碎片算力**：在线业务真实消耗剩余的物理core/内存，通过容器交付，可实时驱逐
- **按需弹性算力**：临时按需采购的按量和竞价算力，可实时驱逐

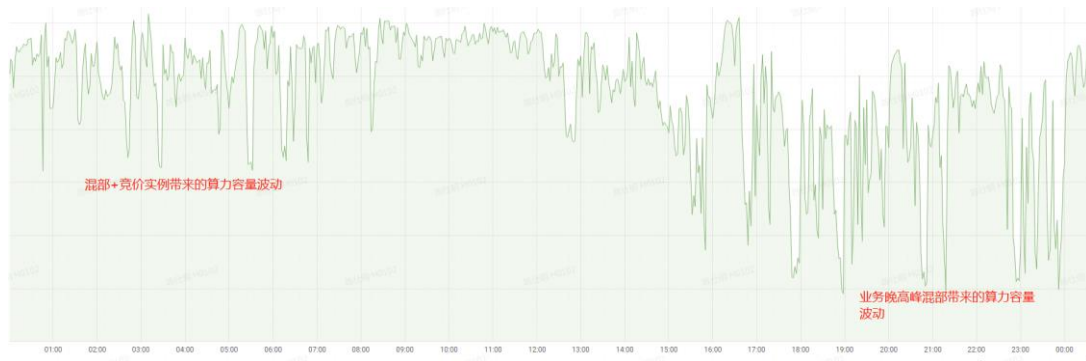
存储

- **表存储**：通过fusionFS存储在COS中，解决存储容量和专线带宽限速
- **hive stage**：存储在hdfs SATA中，缓解COS带宽问题
- **MR shuffle**：计算节点挂载云盘，1Core10G

调度

- 孤岛任务，小IO任务调度上云
- Job Master、关键任务(基于准时基线)，调度到常量算力上
- Task Attempt失败，重新调度到常态算力上

云上集群算力容量



集群容量随着底层资源变化而变化

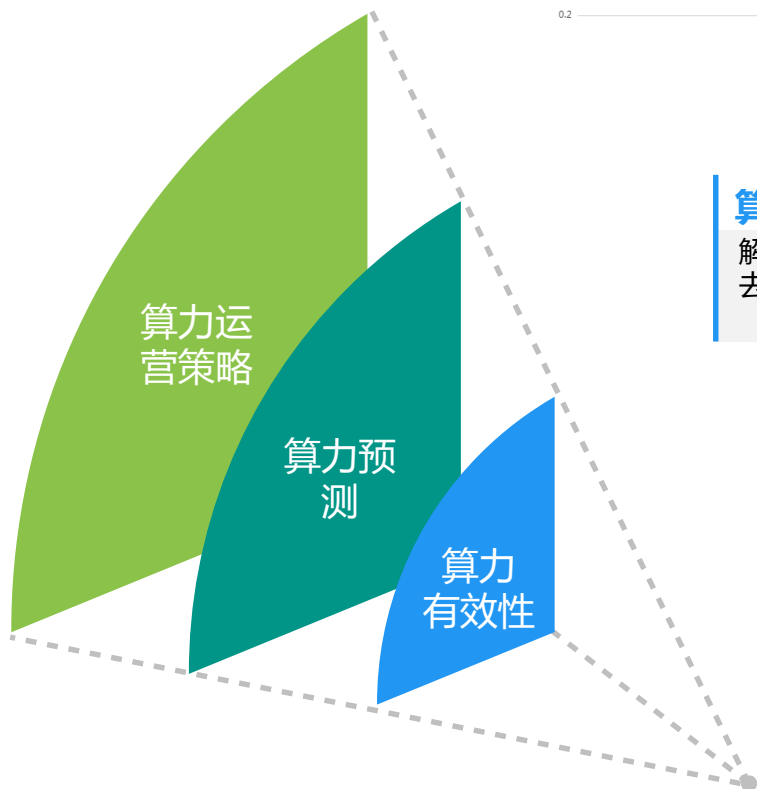
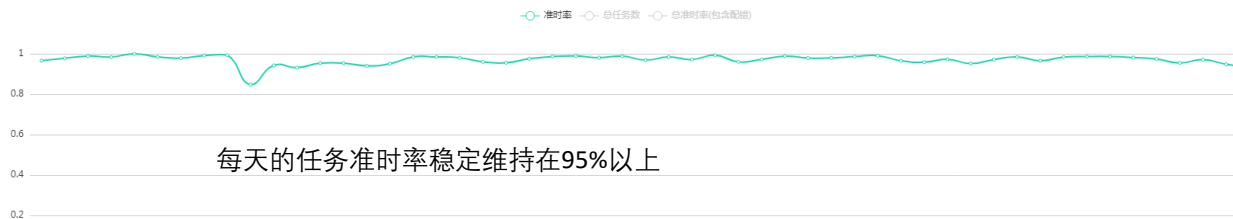
云上集群算力实际使用情况



作业调度，很好的响应了集群的容量变化，云上集群利用率93%



如何做到集群资源利用率如此高，而不影响周期任务准时？



算力有效性

解决算力挤兑问题，没必要现在跑的任务，就不要去挤，能晚点就晚点！不要没意义的拉高算力上限。

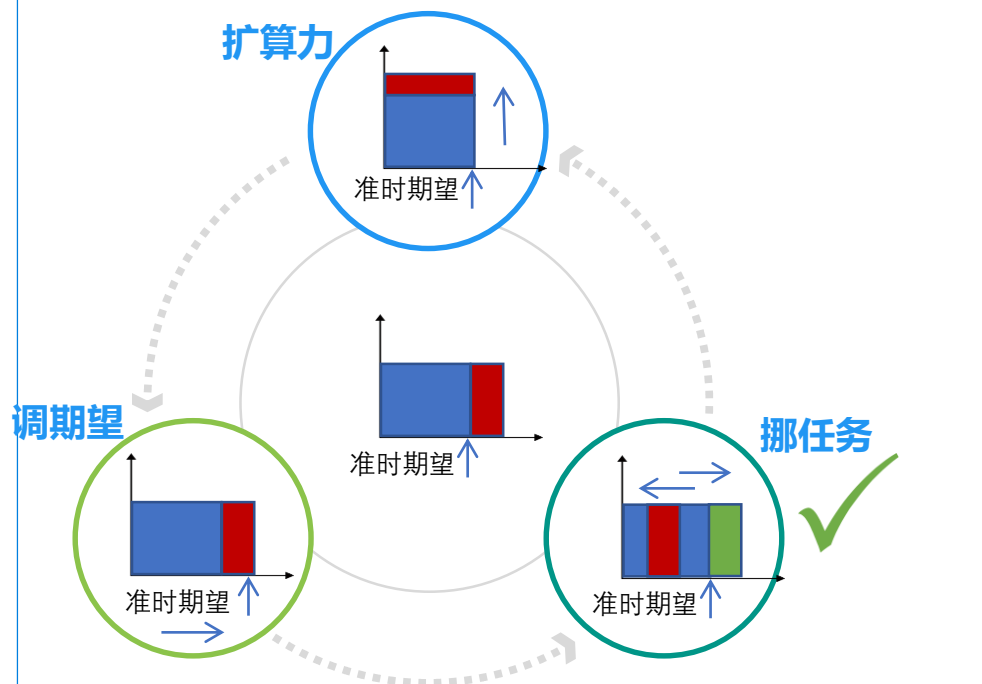
算力预测

解决要准备多少算力合适，比如要重算一个数据，今天晚上给它临时多准备多少算力？新上线了一个任务，对现有任务的准时影响有多大？

算力运营策略

解决任务的准时性设置乱配，预算乱做的问题

算力不足时能够采用的三种手段

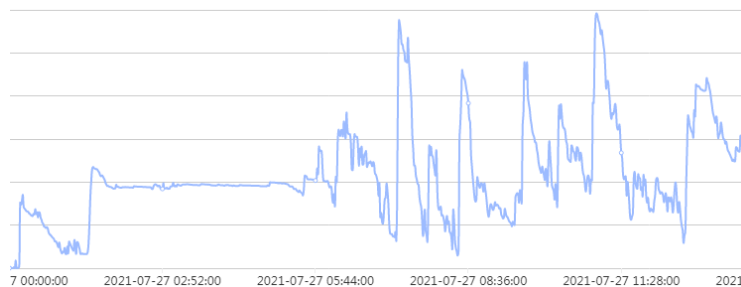
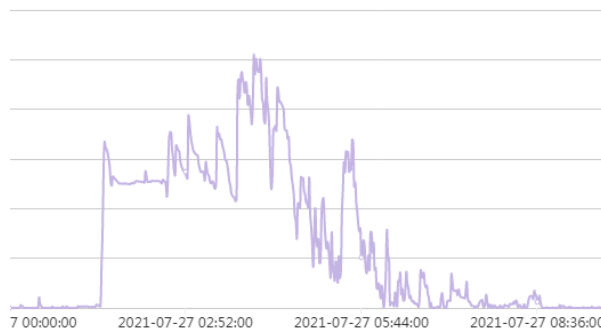


任务准时率

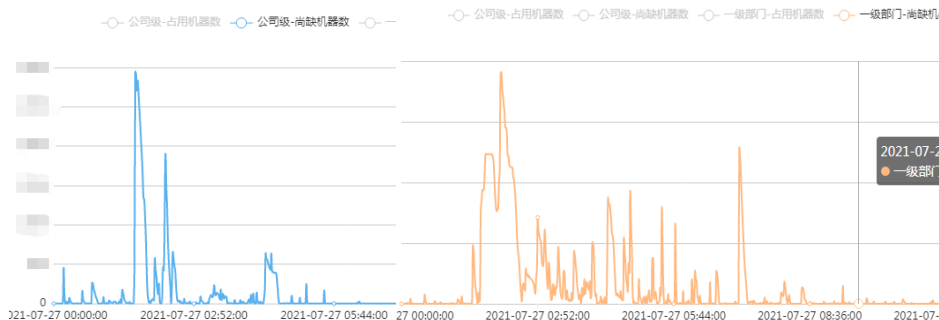
- **任务分级别：**公司级，部门一级，部门二级，普通
- **数据价值权重：**和报表、数据服务打通，该任务衍生出的数据有多少人看，谁看，是否应用到关键产品功能上，计算得到相应的价值权重
- **任务期望完成时间：**末端数据设置期望完成时间



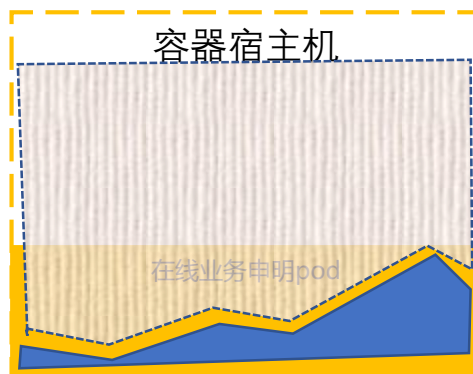
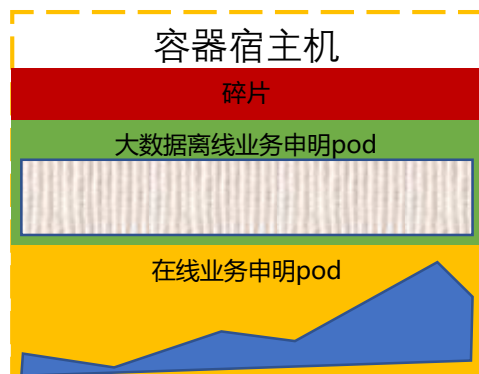
基于综合权重做算力调度，优先给到权重高的任务



从算力需求缺口来看，明显是算力优先让渡给了权重更高的任务



最大的特点：完全白嫖！！！非申明方式，按**实际剩余资源动态供应**给大数据容器

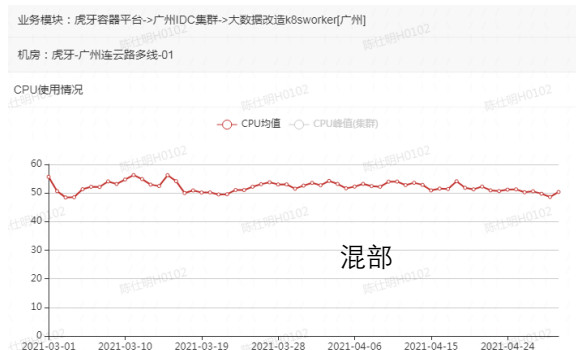


资源利用

- 大数据可用资源大幅提升
- 宿主机上的碎片被利用
- 在线业务可申明的资源增加
- 不符合大数据申明规格的资源被利用，如“没本地磁盘”

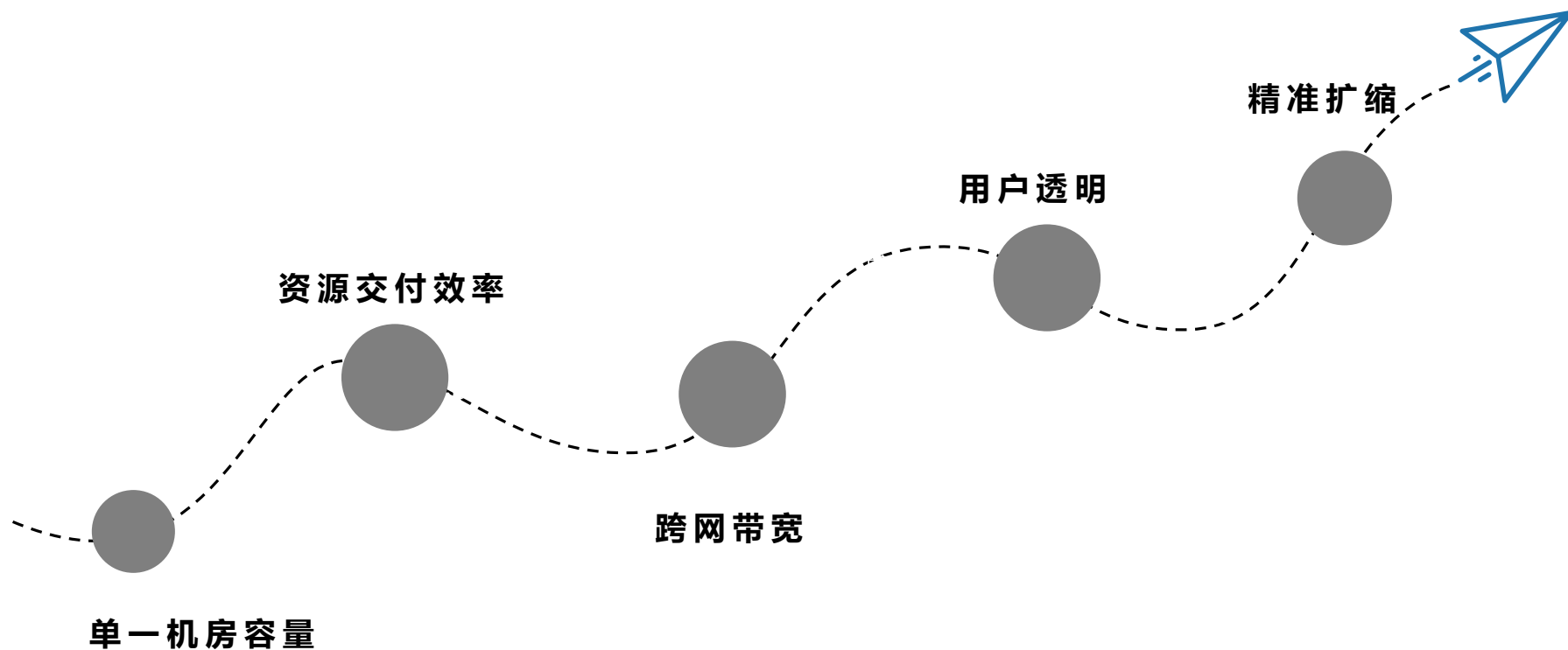
异构的、弹性容量的算力如何使用？

- YARN NM感知容量，秒级更新容量
- YARN NM感知负载，实时分级驱逐
- YARN RM资源/任务画像，差异化调度



全天CPU利用率

- 在线独占：18%
- 大数据独占：63%
- 离在线混部：51%，相比在线独占大幅提高，和大数据独占趋近



1

准实时的任务分簇算法，降低目前人工参与工作，更好的保障SLA

2

云上竞价算力更大力度的使用

3

和云产商更紧密的合作，共同推动大数据融合云的行业发展





麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手**2000**余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过**3000**余家企业续约学习，是科技领域占有率第**1**的客座导师品牌，**msup**以整合全球领先经验实践为己任，为中国产业快速发展提供智库。



高可用架构公众号主要关注互联网架构及高可用、可扩展及高性能领域的知识传播。订阅用户覆盖主流互联网及软件领域系统架构技术从业人员。高可用架构系列社群是一个社区组织，其精神是“分享+交流”，提倡社区的人人参与，同时从社区获得高质量的内容。