深度计算平台 与 响应时效管理的思考
SysML and Time based model serving

车轮互联 数据VP 张翔

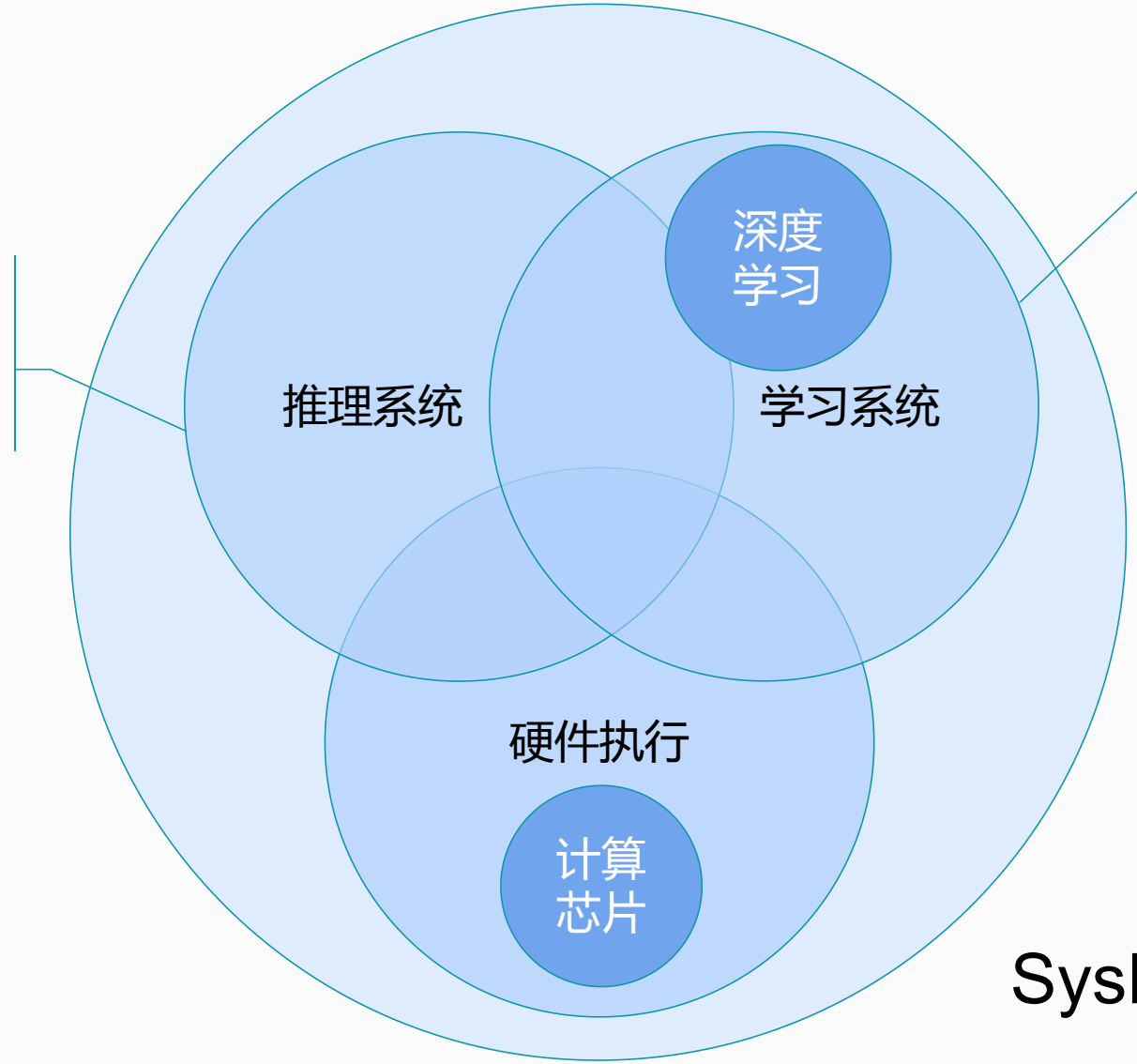
CHELUN

深度计算平台与大数据平台的差异

	Big data	AI
应用密集型	从大量非标数据中寻找一致性规律	按需而变的个性化结果
计算资源密集型		GPU 高速计算
存储资源密集型	从海量数据存放着手解决大数据问题 分布式存储	从集中计算资源着手解决复杂计算问题

深度计算平台与深度学习系统的关系

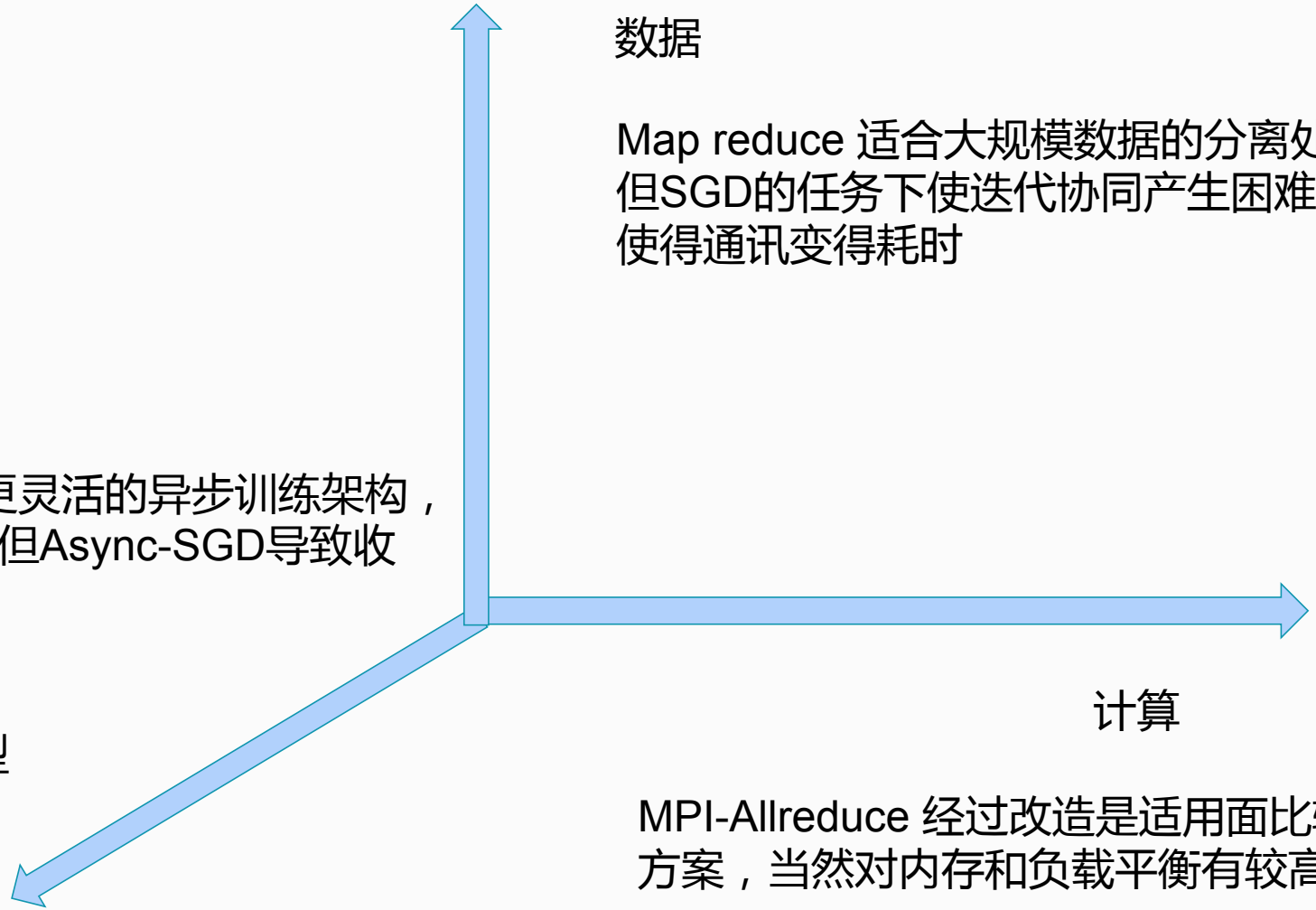
Serving:
大量并发请求
复杂逻辑推理
高实时性约束



Training:
大量训练数据
超高迭代次数
实时性不敏感

SysML 深度计算平台

Model Training 的分布式选型



数据

Map reduce 适合大规模数据的分离处理，但SGD的任务下使迭代协同产生困难，也使得通讯变得耗时

Parameter server是更灵活的异步训练架构，支持online training，但Async-SGD导致收敛会比较慢

计算

模型

MPI-Allreduce 经过改造是适用面比较广的方案，当然对内存和负载平衡有较高要求

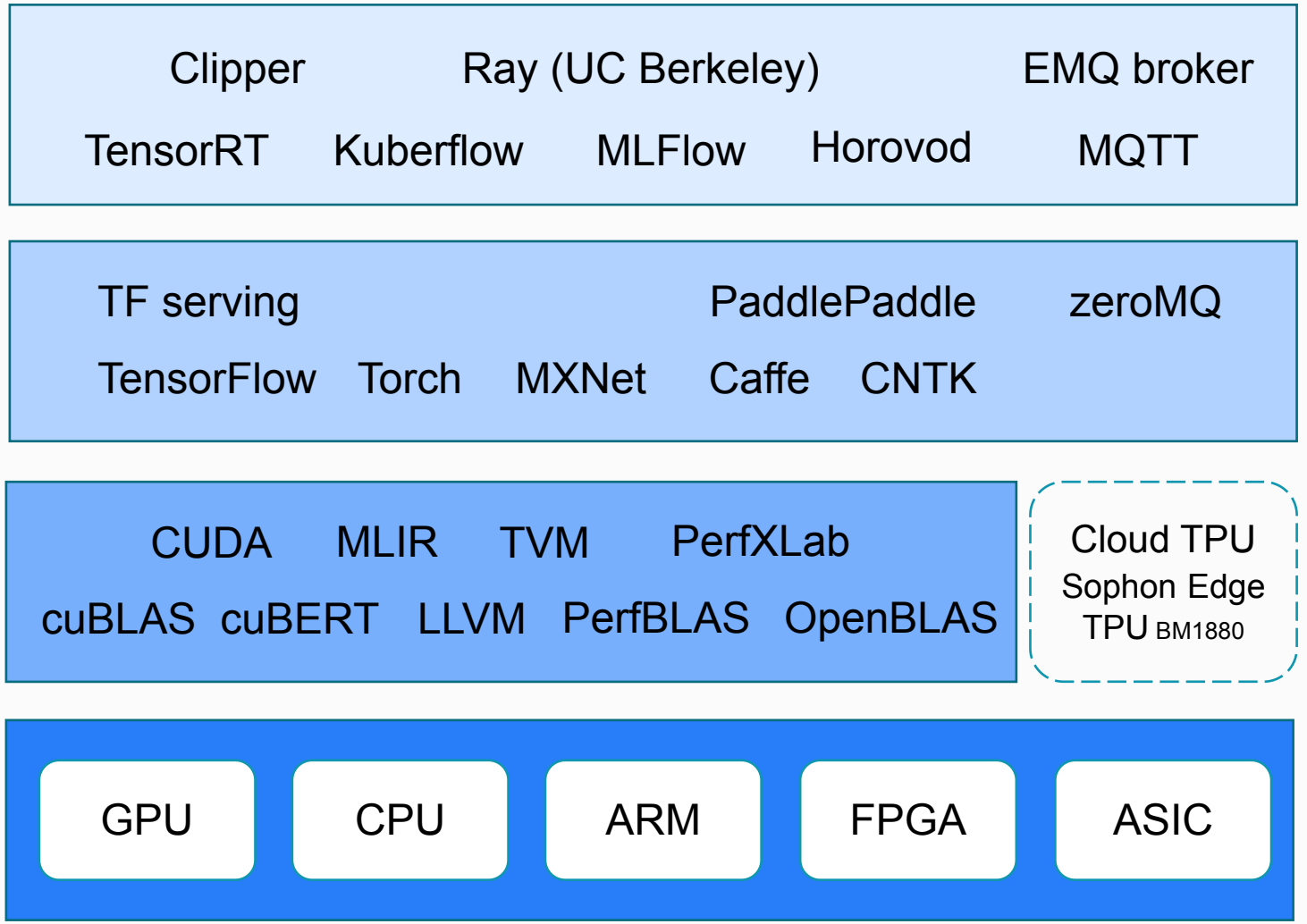
Model serving 生态环境

行业应用者：
非技术公司使用云服务
中小互联网公司用开源

技术领先者：
腾讯bert-as-service
美团的TFS+yarn集群
知乎cuBert
星环科技

行业领导者：
阿里PAI使用TVM
优化 transformer
科大讯飞的FPGA

标准制定者：
Nvidia，google，
寒武纪，海思

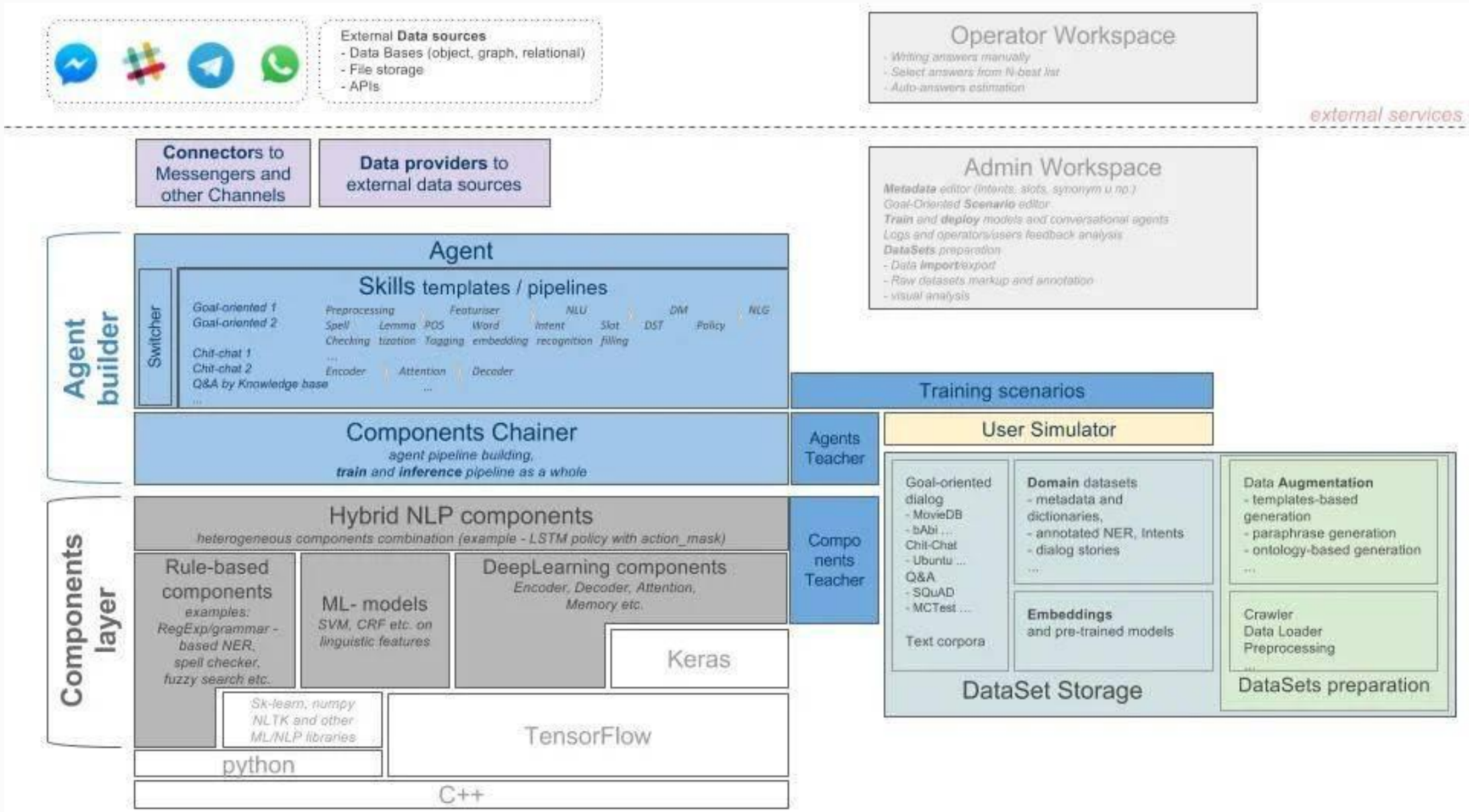


热插拔

云到端

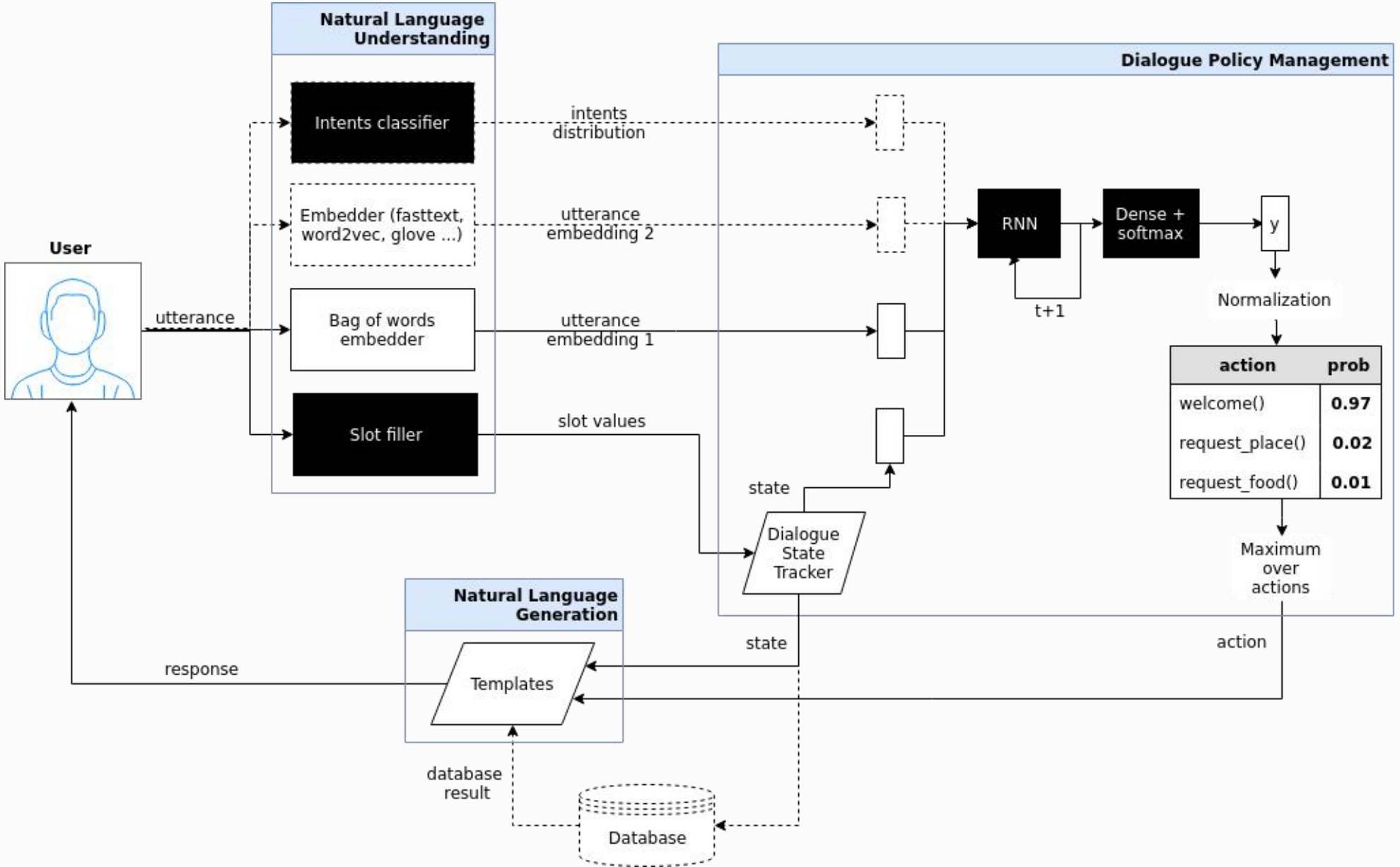
Chat bot的任务案例

来自俄罗斯NLP项目DeepPavlov

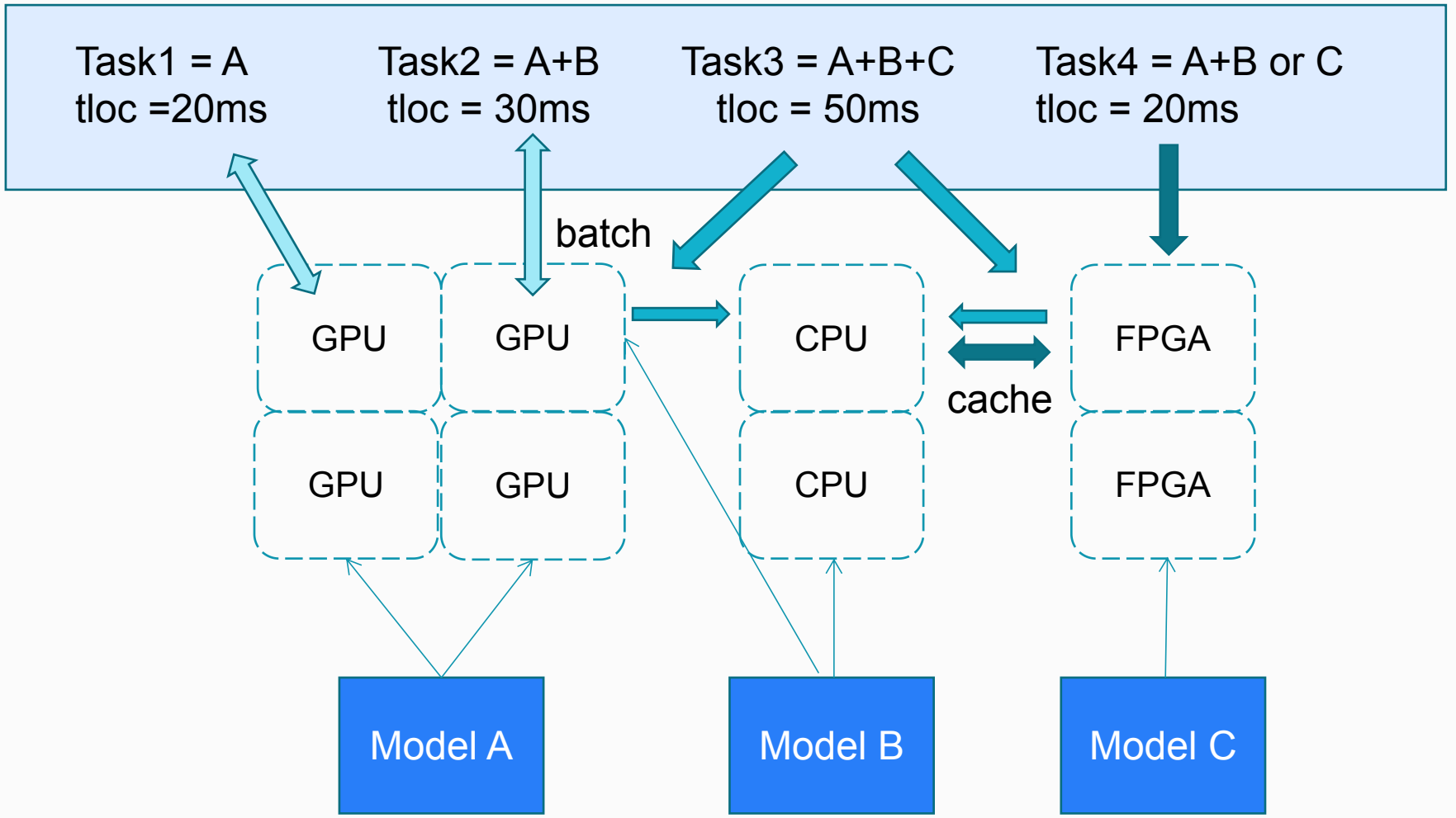


Chat bot的问题抽象

来自俄罗斯NLP项目DeepPavlov

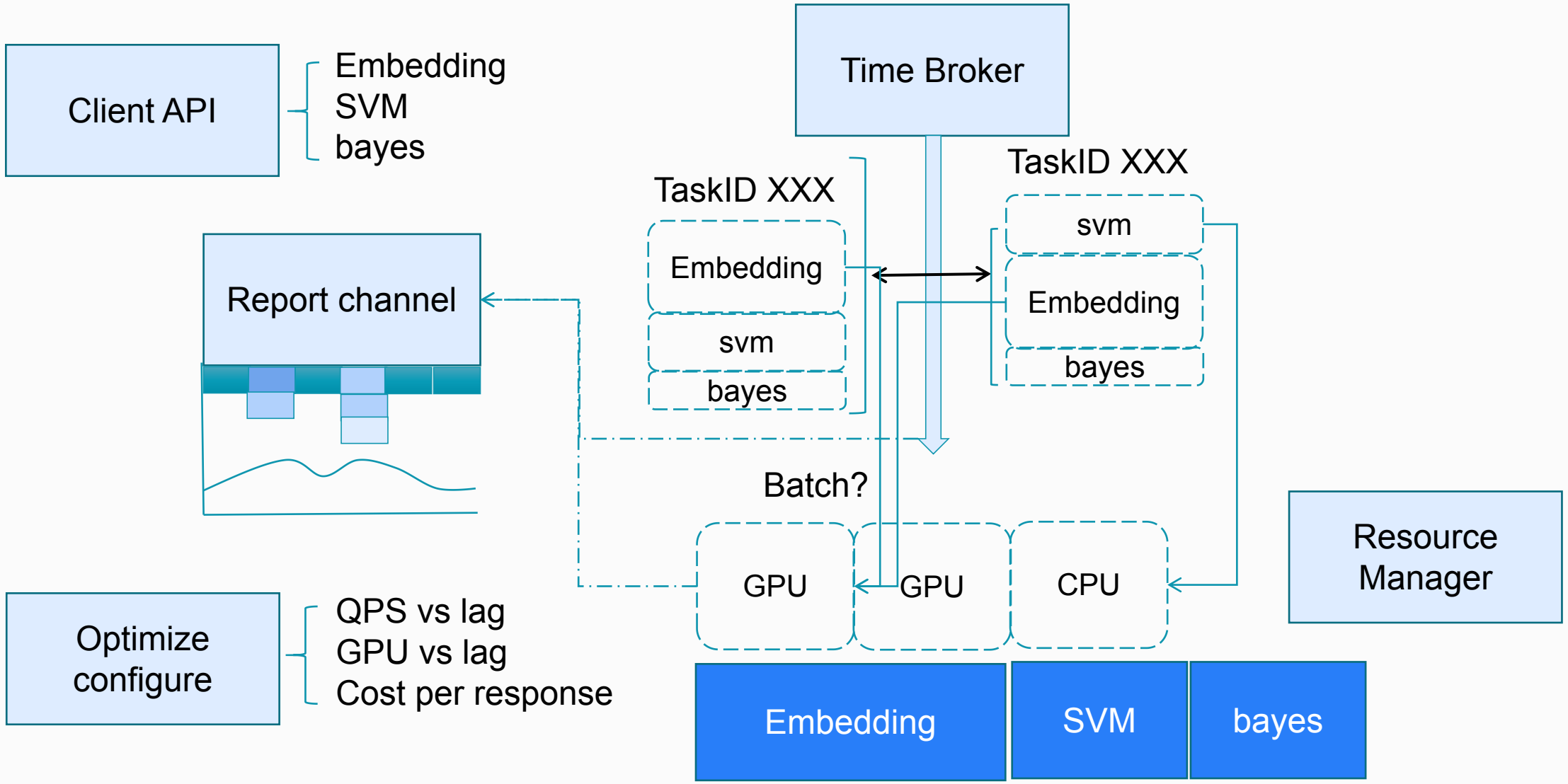


AI产品经理的工作：从模型到算力到用户任务的全流程




12
28
13
0/1210
Combo

TBMS:Time based model serving



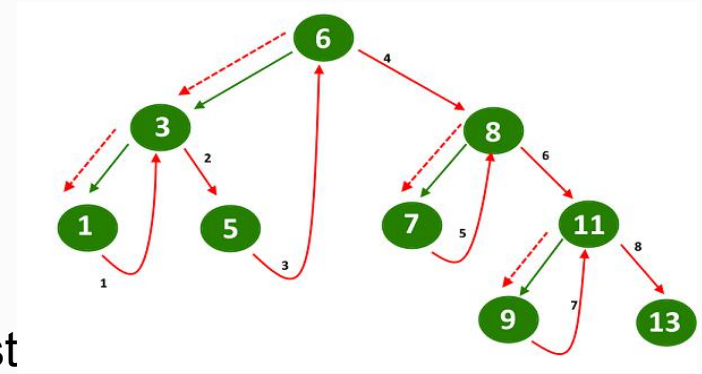
产品服务端请求 Python API mixTry

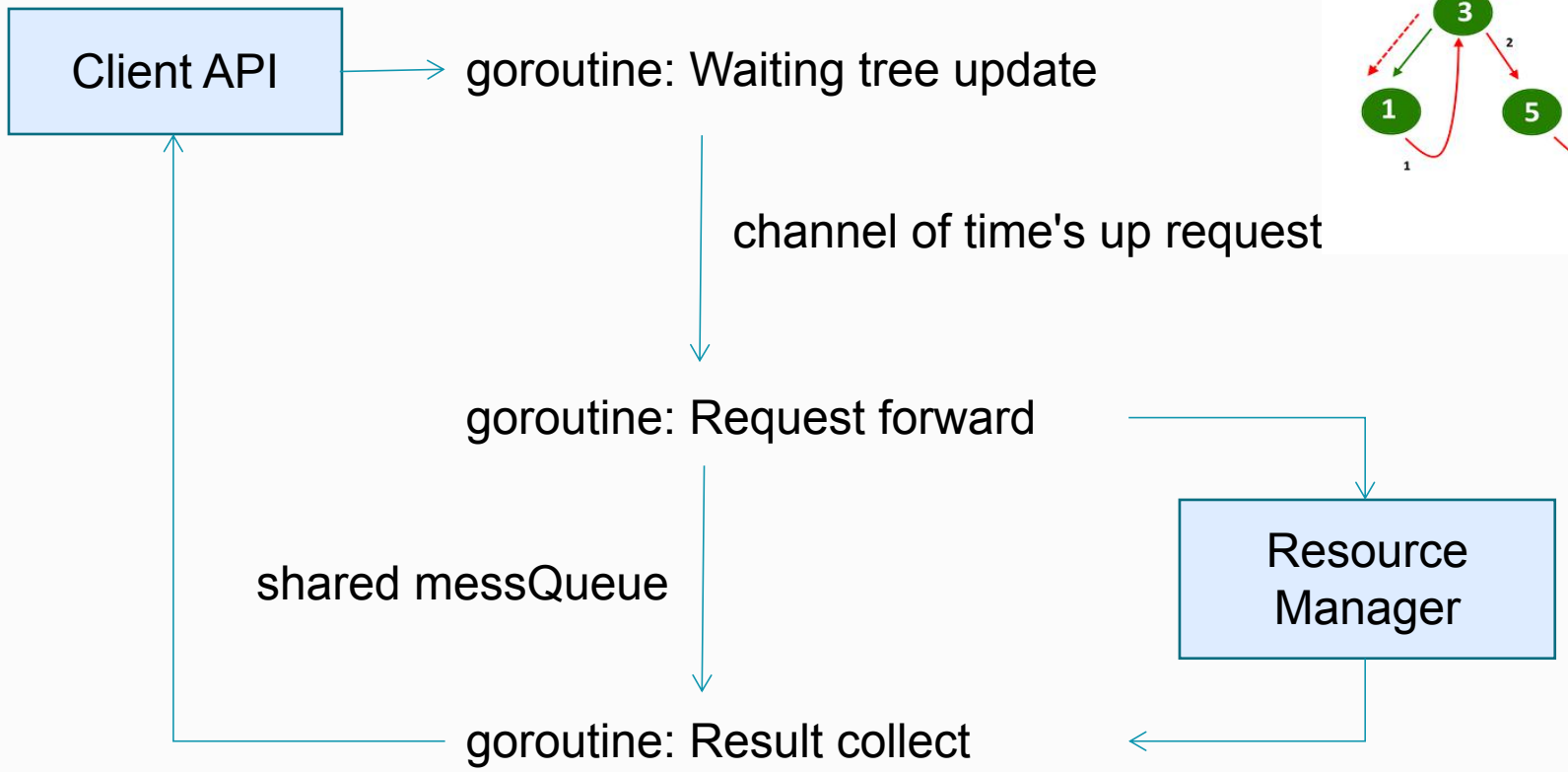
```
tbms_client = tbmsClient(ENDPOINT, ACCESSID, ACCESSKEY, INSTANCENAME)
tbms_models = tbmsList({"embedding":{"est":35},
                        "svm":{"est":25},
                        "bayes":{"est":15},
                        "keysearch"{"est":5}
                       })

def main():
    questionString = sys.argv
    if len(sys.argv)>=1:
        questionString = sys.argv[0]
        answer =
tbmsTry(tbmsClient,tbms_models,questionString,tloc=50,crossRequest=1,crossLag=10,priority=0)
       if len(answer) >0:
            print answer
        else:
            print u'human assist'
```

任务分发服务 timeBroker

一种基于并行二叉树的Go语言实例





https://github.com/linkinbird/TBMS/blob/master/timeBroker.go

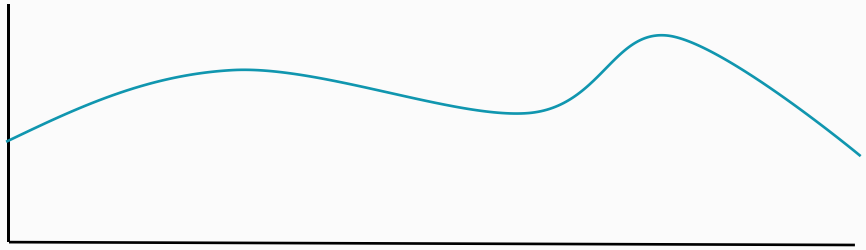
监控管理 reportChannel

同时监听timeBroker和ResourceManager的情况

Down Grade %

TaskA	TaskB	TaskC	TaskE	TaskF
embedding	embedding	embedding	embedding	bayes
bayes	svm	keywords	svm	keywords
	bayes		bayes	
			keywords	

Lag time (10min, avg)

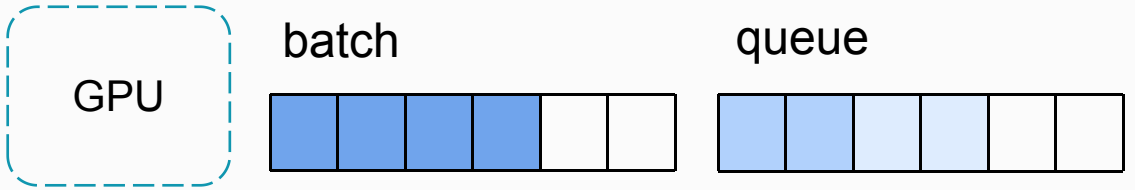


Model resource occupancy %

	embedding	SVM	bayes
GPU1	90%		
GPU2	90%		
GPU3	10% (free)		
CPU1		50%	50%

配置调优 opiConfigure

提早发车保证延迟
但牺牲 QPS



<u>最大QPS 有高lag 甚至降级</u>

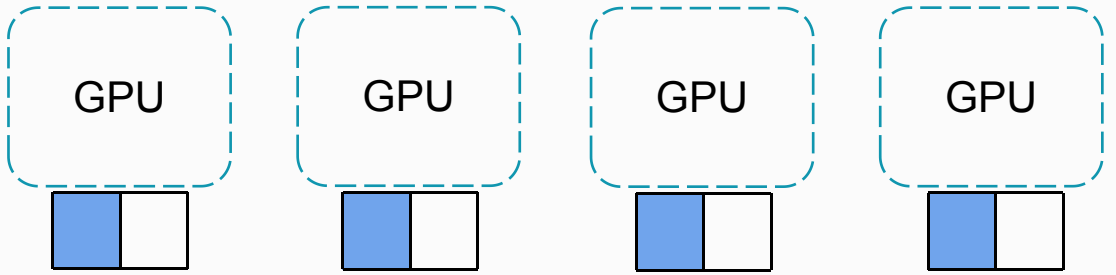
Batch size = 10
Batch time = 20 ms
QPS max = (1000/20) x 10 = 500

$$\text{Lag} = \text{Batch time} + (1000/\text{QPS}) \times \frac{\text{Batch size} - 1}{2}$$

Lag 200 = 20 + 5 x 1.5 = 27.5 ms

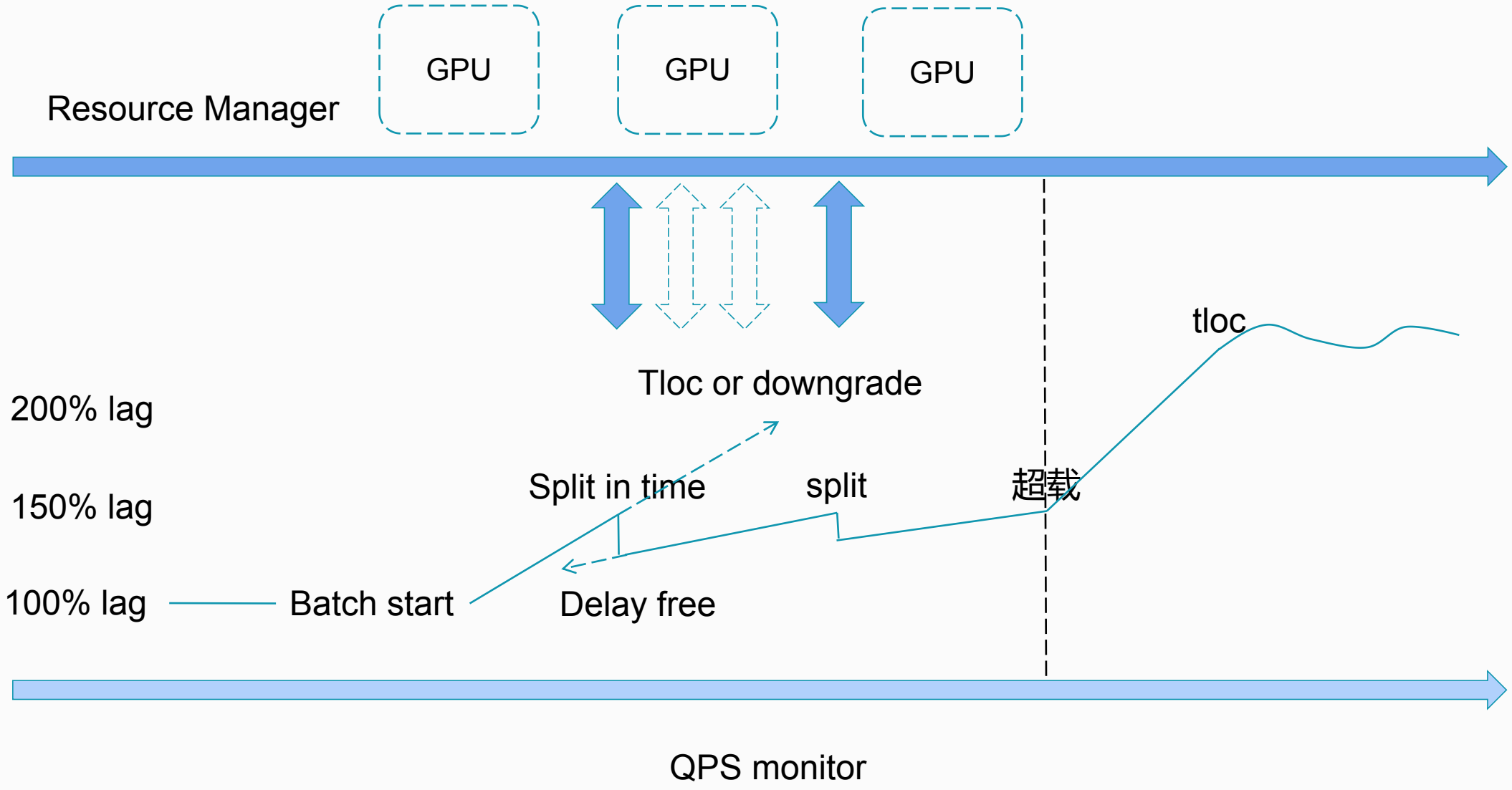
$$\text{Lag} \frac{\text{max}}{\text{min}} = 1 + \frac{\text{Batch size} - 1}{2 \times \text{batch size}}$$

更多GPU资源满足两者
但利用率不一定高



Lag 200 = 20ms < 27.5ms
but occupancy is low

动态配置

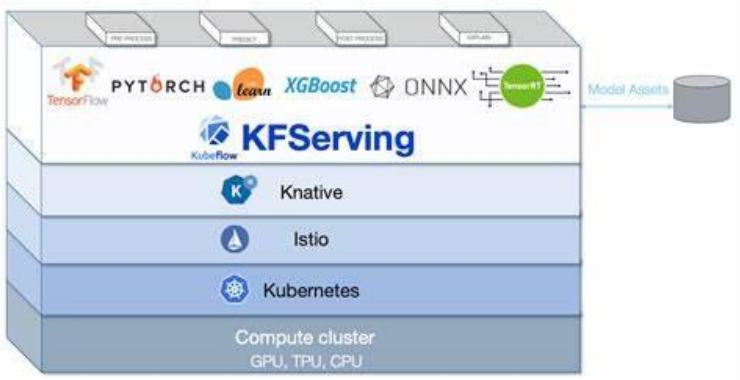

GPU
GPU
GPU
Resource Manager
Tloc or downgrade
tloc
200% lag
150% lag
100% lag
Split in time
split
超载
Batch start
Delay free
QPS monitor

资源管理 Resource Manager

混动 VS 纯电

混动方案：服务网络（Service Mesh）可以和timeBroker层结合在一起：

- 市面上已经有了与kubernetes解耦的方案，比如Kong的Kuma
- 现有的load balance不符合GPU的batch原则，但可以用TensorRT的dynamic_batching和Streaming来配合执行
- 后续兼顾GPU和FPGA物联网混合的场景

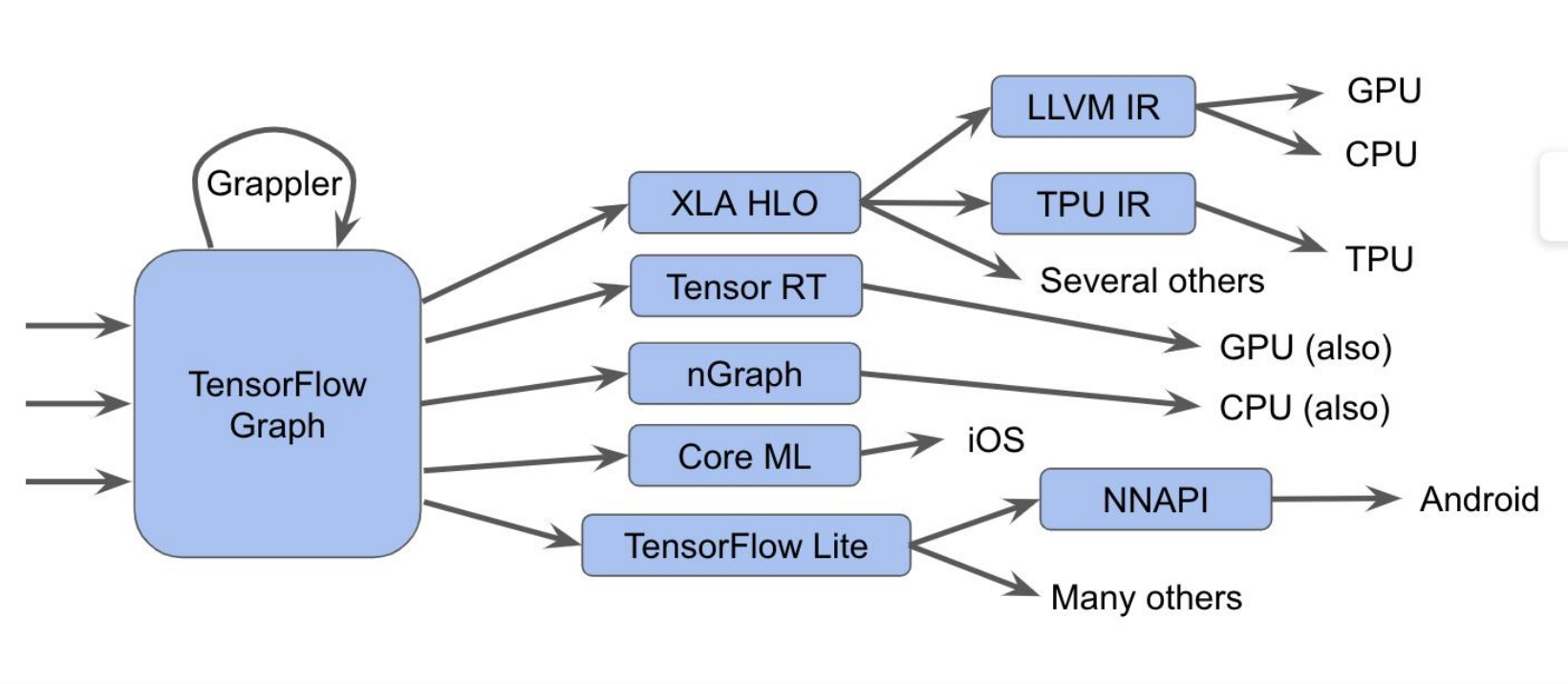


资源执行的复杂度

把资源管理的复杂度放在资源抽象层，而不是kubernetes那种调度层，是因为深度学习的执行层优化还没有统一：开源社区的TVM vs 谷歌主导的MLIR

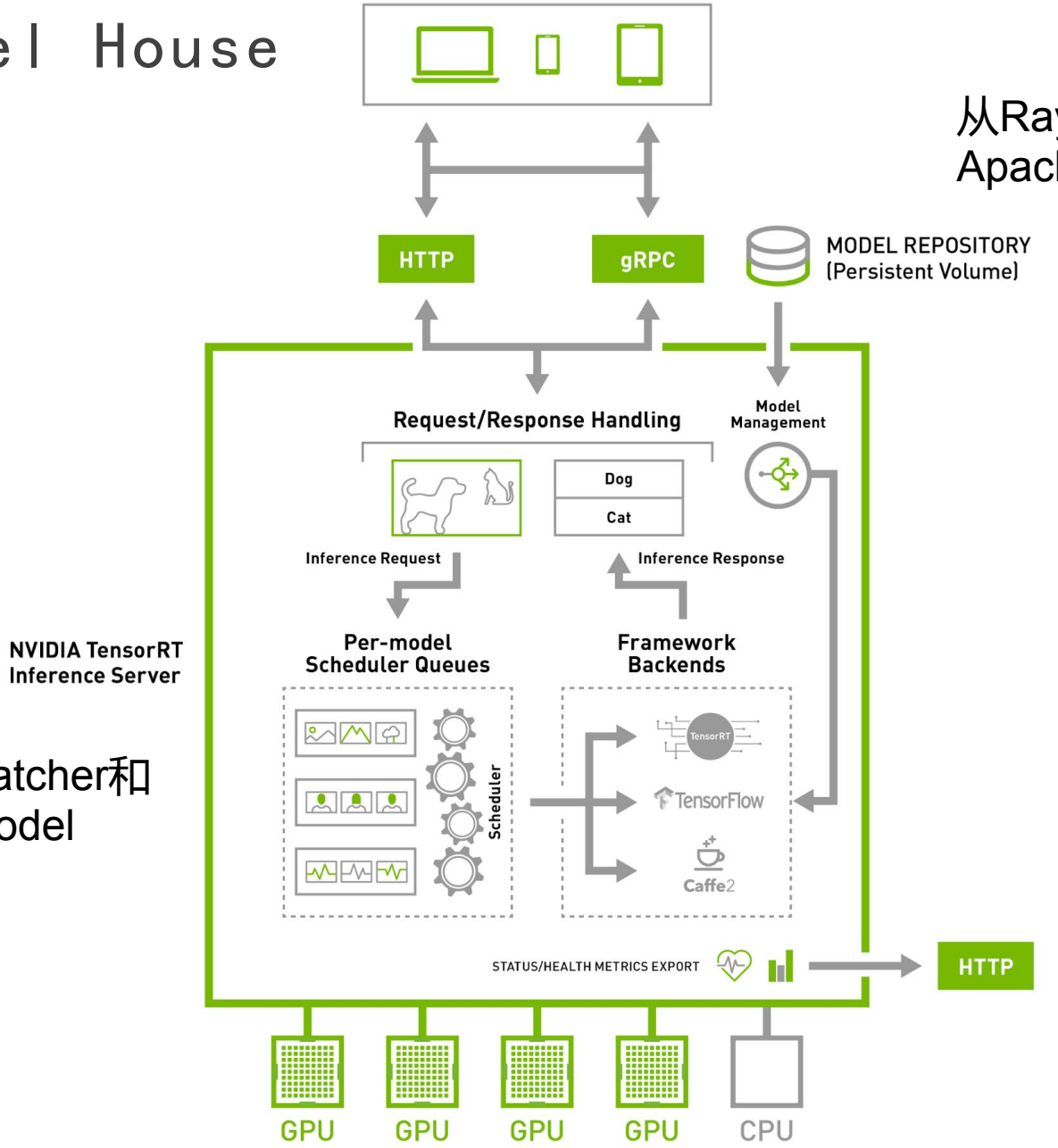
- 用一种统一的build工具去产生pod部署 目前不可行
- Kuberflow/kfserving/Knative/Istio 就不支持adaptive batch和虚拟GPU

将针对不同场景和硬件的最佳执行方案交给模型提供方去解决，在上层通过解耦的service mesh管理服务



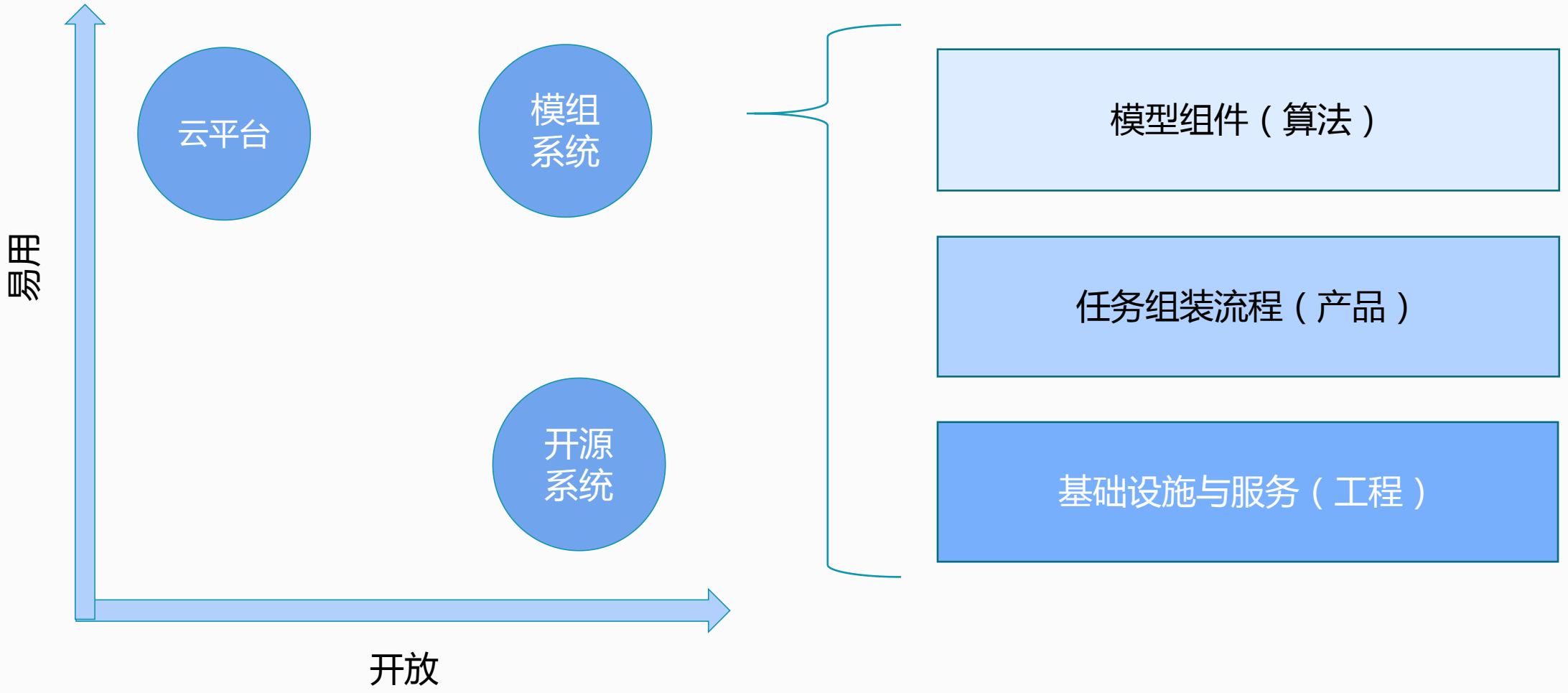
模型仓库 Model House

从Ray独立出来的黑科技：
Apache Arrow

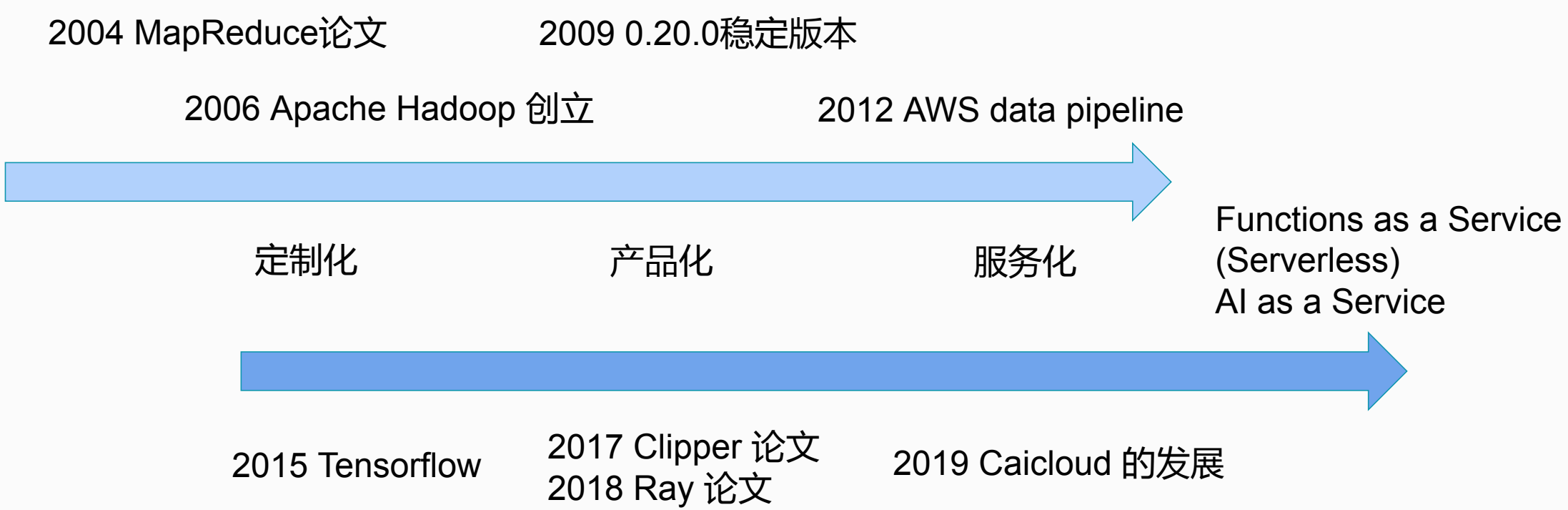


Sequence Batcher和
Ensemble model

Model serving 商业路径选择



深度计算平台发展趋势


2004 MapReduce论文
2009 0.20.0稳定版本
2006 Apache Hadoop 创立
2012 AWS data pipeline
定制化
产品化
服务化
Functions as a Service
(Serverless)
AI as a Service
2015 Tensorflow
2017 Clipper 论文
2018 Ray 论文
2019 Caicloud 的发展

谢谢

birdzhangxiang@gmail.com

让人类平等自由的享受智能带来的福利
让企业可以最低的成本，最便捷的管理，
应用最前沿的智能技术