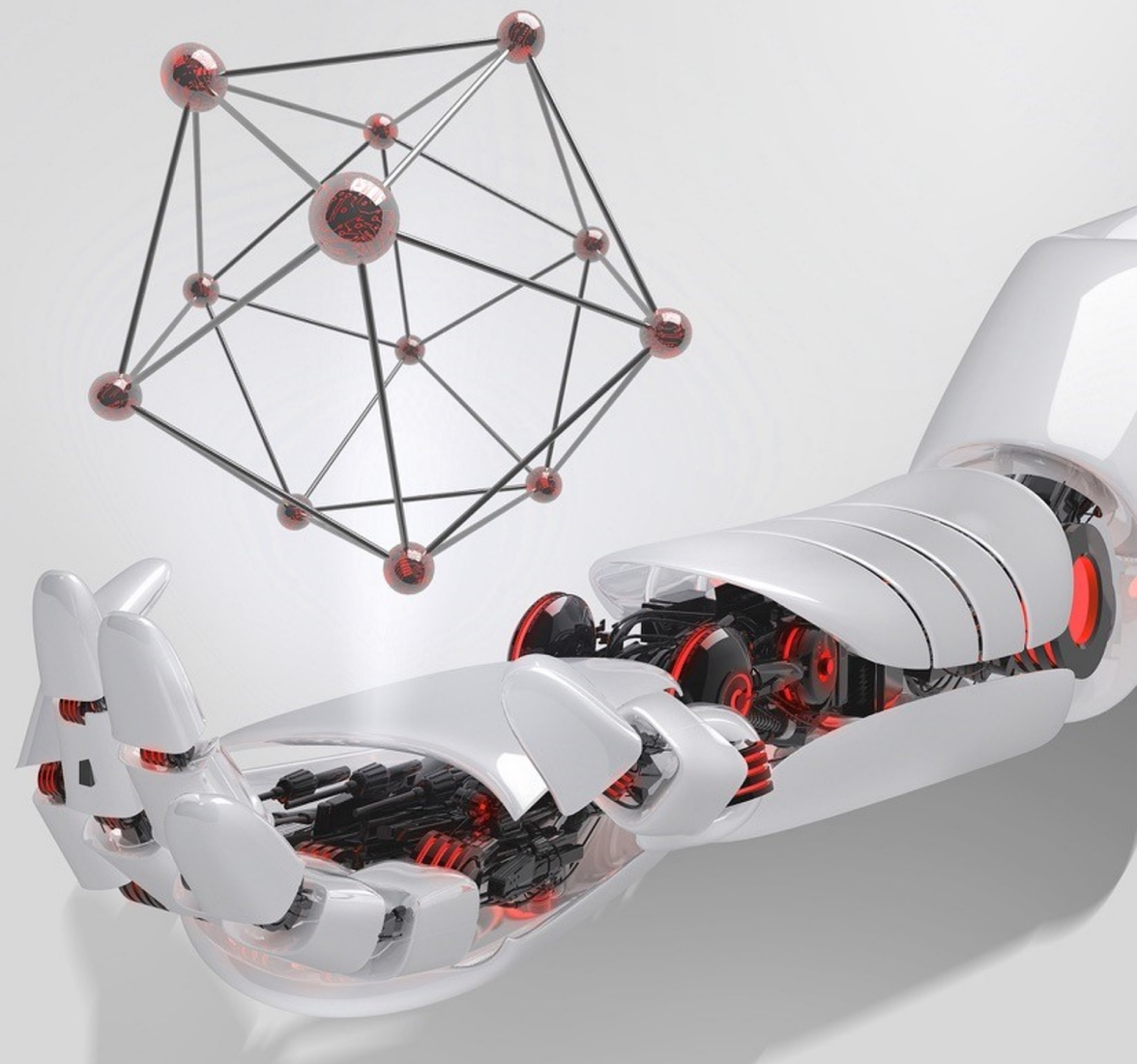


# 从 PyFlink 到 Klein : 当我们决定做流批统一的 推理计算引擎

演讲人：唐云

小红书实时计算引擎负责人 /  
Apache Flink Committer



RAY CONNECT 2024

# 「目录

01

PyFlink基于CPU  
混部资源的大规模推理

02

一个优秀的推理引擎  
应该具备什么

03

如何设计Ray Klein

04

未来与展望

# 01 PyFlink基于CPU混部资源的大规模推理

- 小红书的以图搜图业务与大规模推理诉求
- 项目落地过程中的挑战
- 基于PyFlink的大规模推理暴露出来的问题

# 小红书的以图搜图业务与大规模推理诉求

- 通过以图搜图，用户检索相似的笔记或者相似的商品
- 寻找同好，提高用户粘性



自爆版捉迷藏，一小时翻  
这两天在小红书的艺术假日发了个假，临走前在阿那亚藏了8个我的分身，有缘可拿，先送先拿。  
有如图示这么小的，也有爆炸迷你小小小的，遇到别太吃惊。  
温馨提醒，图上提示归提示，有的位置有那么两米的偏差，升级了一下难度，拿到的发在评论区喂一下别的小红薯们。  
下次放这种好事再记得叫我啊@艺术薯  
#小红书的艺术假日 #阿那亚戏剧节 #捉迷藏

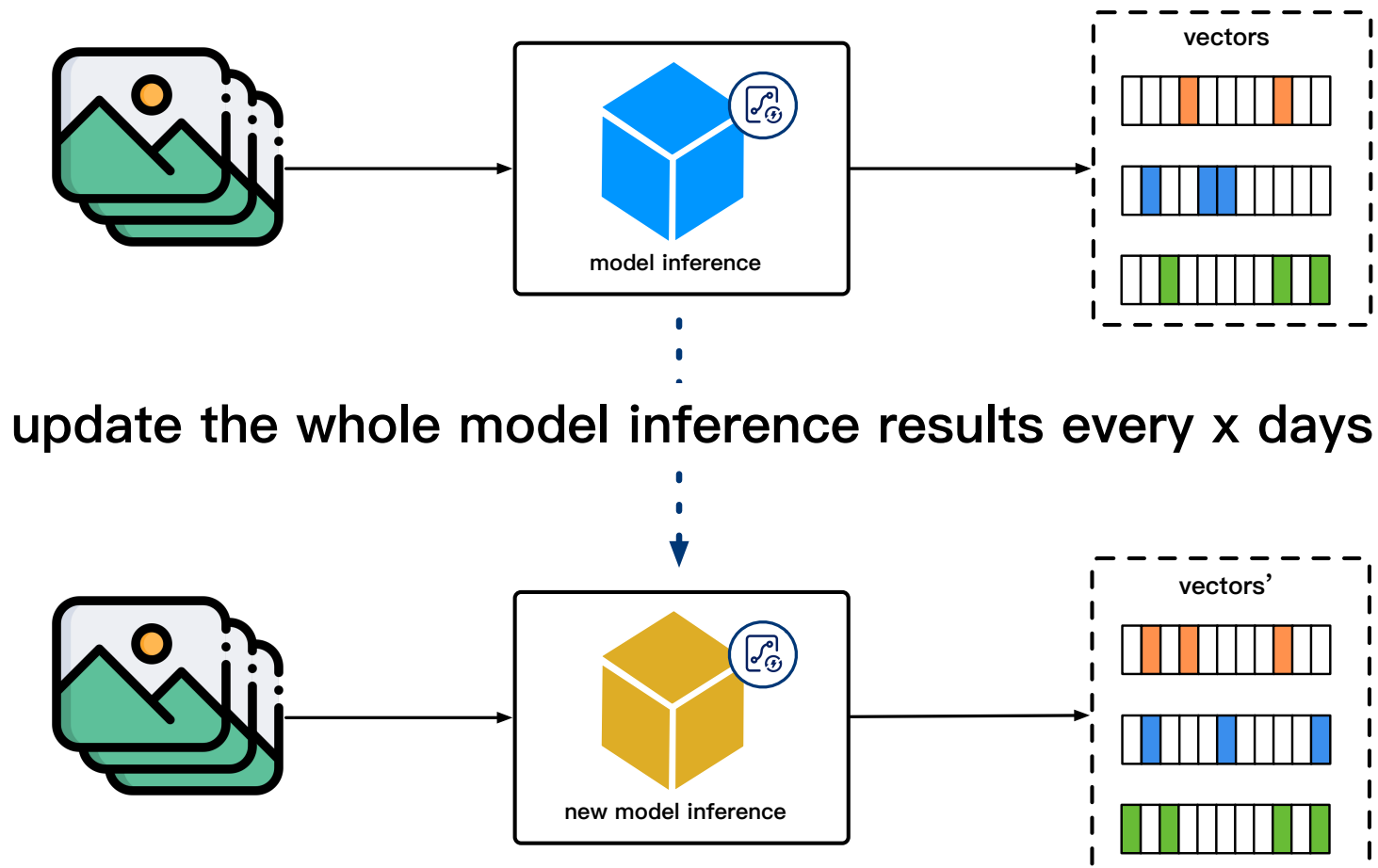


自爆版捉迷藏，一小时翻  
这两天在小红书的艺术假日发了个假，临走前在阿那亚藏了8个我的分身，有缘可拿，先送先拿。  
有如图示这么小的，也有爆炸迷你小小小的，遇到别太吃惊。  
温馨提醒，图上提示归提示，有的位置有那么两米的偏差，升级了一下难度，拿到的发在评论区喂一下别的小红薯们。  
下次放这种好事再记得叫我啊@艺术薯  
#小红书的艺术假日 #阿那亚戏剧节 #捉迷藏



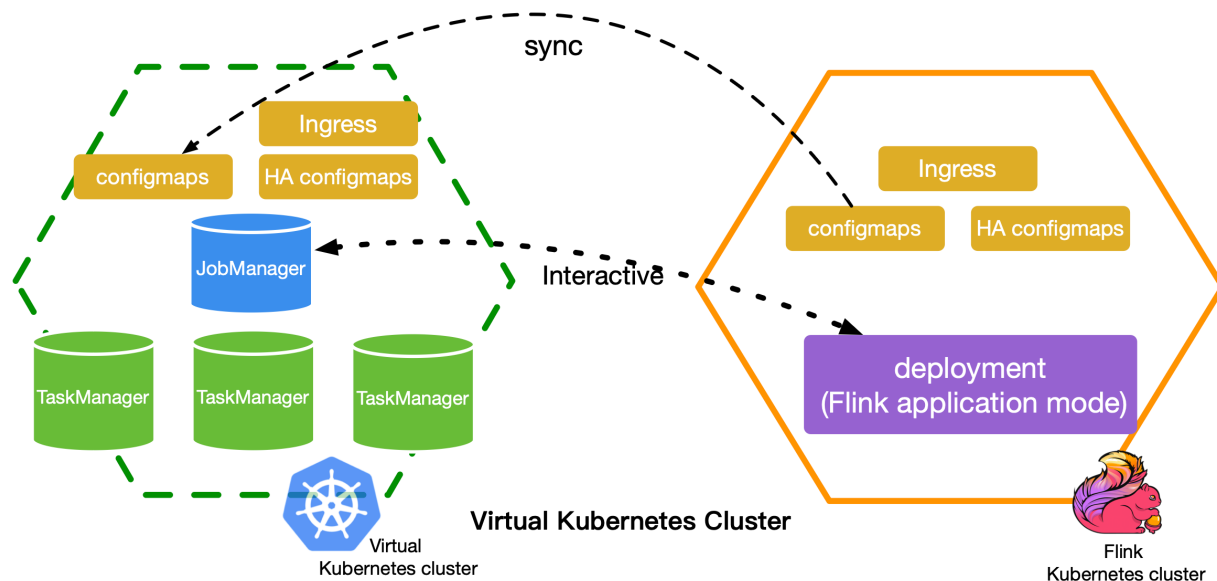
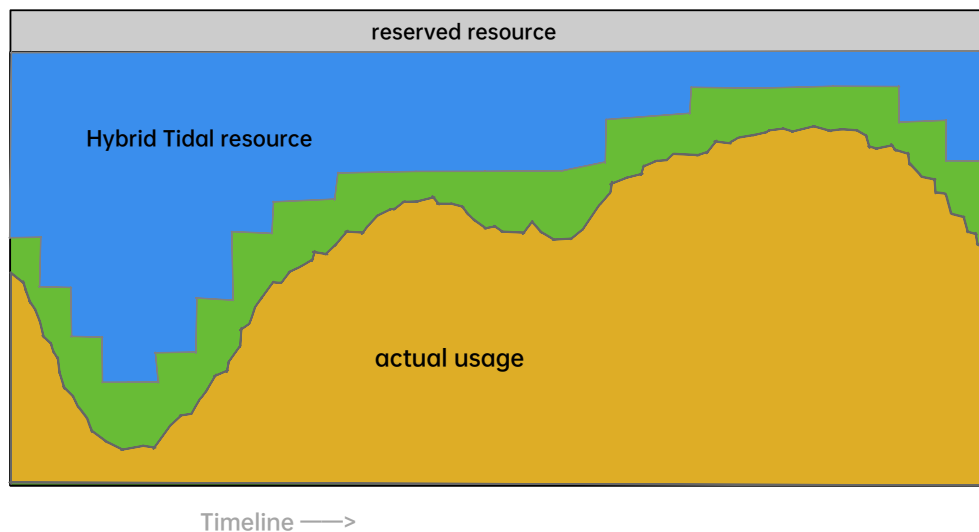
# 小红书的以图搜图业务与大规模推理诉求

- 为了提高用户体验，算法团队每隔一段时间就需要更新模型，并且需要对全量图片进行回刷推理
- 所需资源量极大，需要使用混部资源降低成本



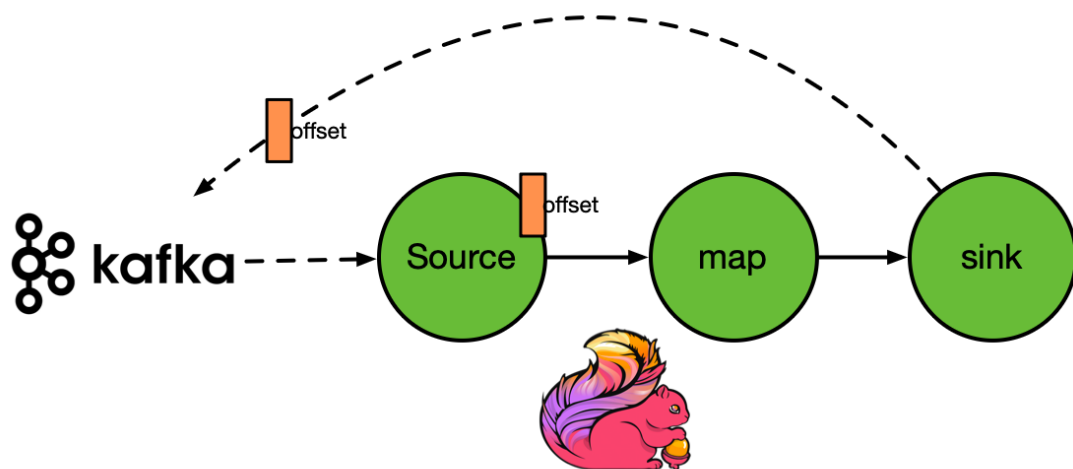
# 项目落地过程中的挑战

- 基于混部潮汐资源的PyFlink部署



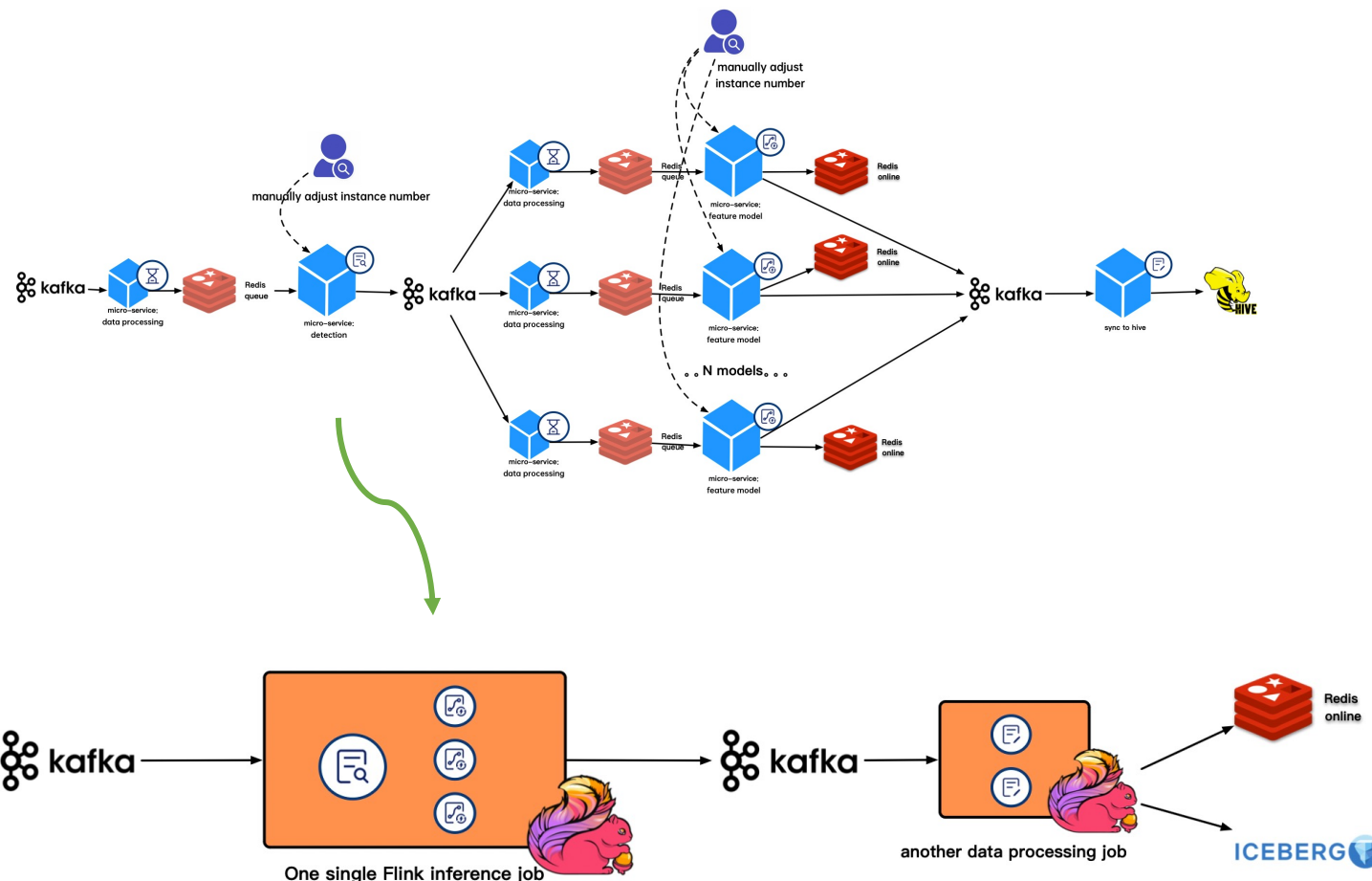
# 项目落地过程中的挑战

- 在Adaptive scheduler中支持region failover  
<https://issues.apache.org/jira/browse/FLINK-32818>
- 由于CPU推理太慢，改良了更轻量级的容错，而不是直接使用Flink checkpoint



# 项目落地过程中的挑战

- 利用潮汐资源，节省了~40万 core\*day的CPU用量
- 大大简化了推理链路
- 与之前的方案相比，几乎没有数据丢失
- 无需手动调整并发度与资源用量，降低了运维成本












# 基于PyFlink的大规模推理暴露出来的问题

- 传统大数据引擎主要基于JVM，即使是PySpark或者PyFlink，也只是python interface，而不是python native
- Java生态对GPU支持不友好，无法在更多场景推广

## 02

## 一个优秀的推理引擎应该具备什么

	Spark	Flink (streaming & batch)	Ray
Python-native	 python-interface	 python-interface	✓
是否很方便得调用GPU	 具备有限能力	 具备相关理论能力，但是使用不方便	✓
是否支持CPU/GPU异构计算	✗	✗	✓
是否支持流式执行 (吞吐更高，更适合GPU)	✗	✓	✓
是否支持低延迟近线计算能力	 spark streaming支持，但是延迟表现不好	✓	 开源版不支持
流批一体的开发接口	✗	✓	 开源版不支持
dataflow的编程模式	✓	✓	✓
client/server 模式的在线推理服务	✗	✗	✓ Ray Serve支持
容错能力	✓	✓	✓
占用资源可以随用随停，周期性调度	✓	✓	✓

## 03 如何设计Ray Klein

- Ray与实时计算引擎
- 如何实现流批一体的Ray Klein引擎
- 落地情况

# Ray与实时计算引擎

- 虽然Ray社区的相关实时计算库不是很成功，但实际上Ray具备很好的底子

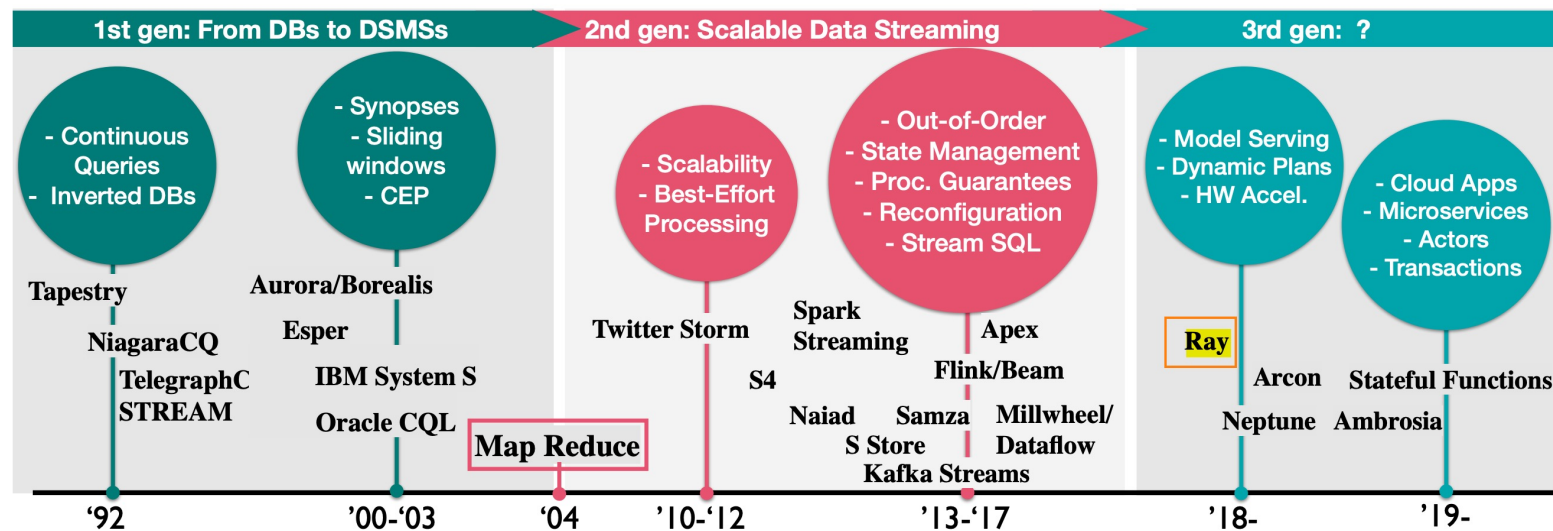
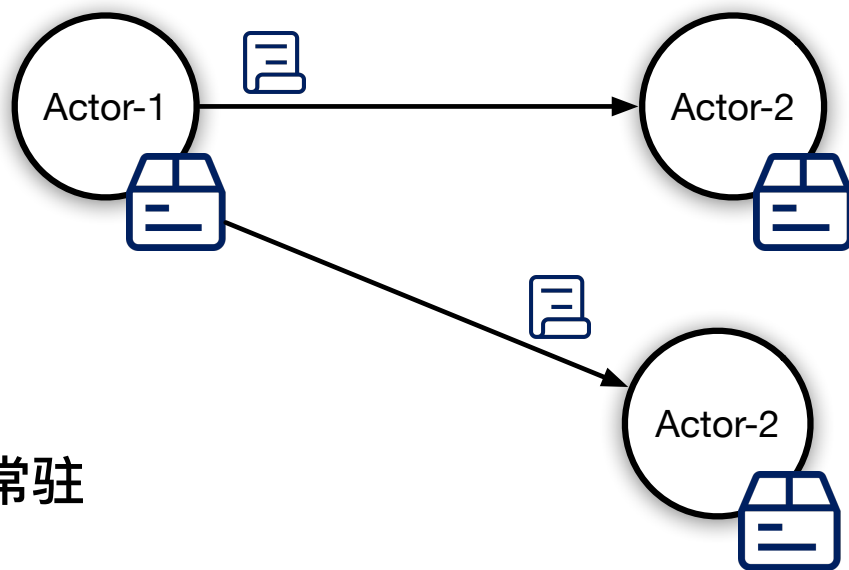


Figure 1: An overview of the evolution of stream processing and respective domains of focus.

SIGMOD'20: [Beyond Analytics: The Evolution of Stream Processing Systems](#)

# 如何实现流批一体的Ray Klein引擎

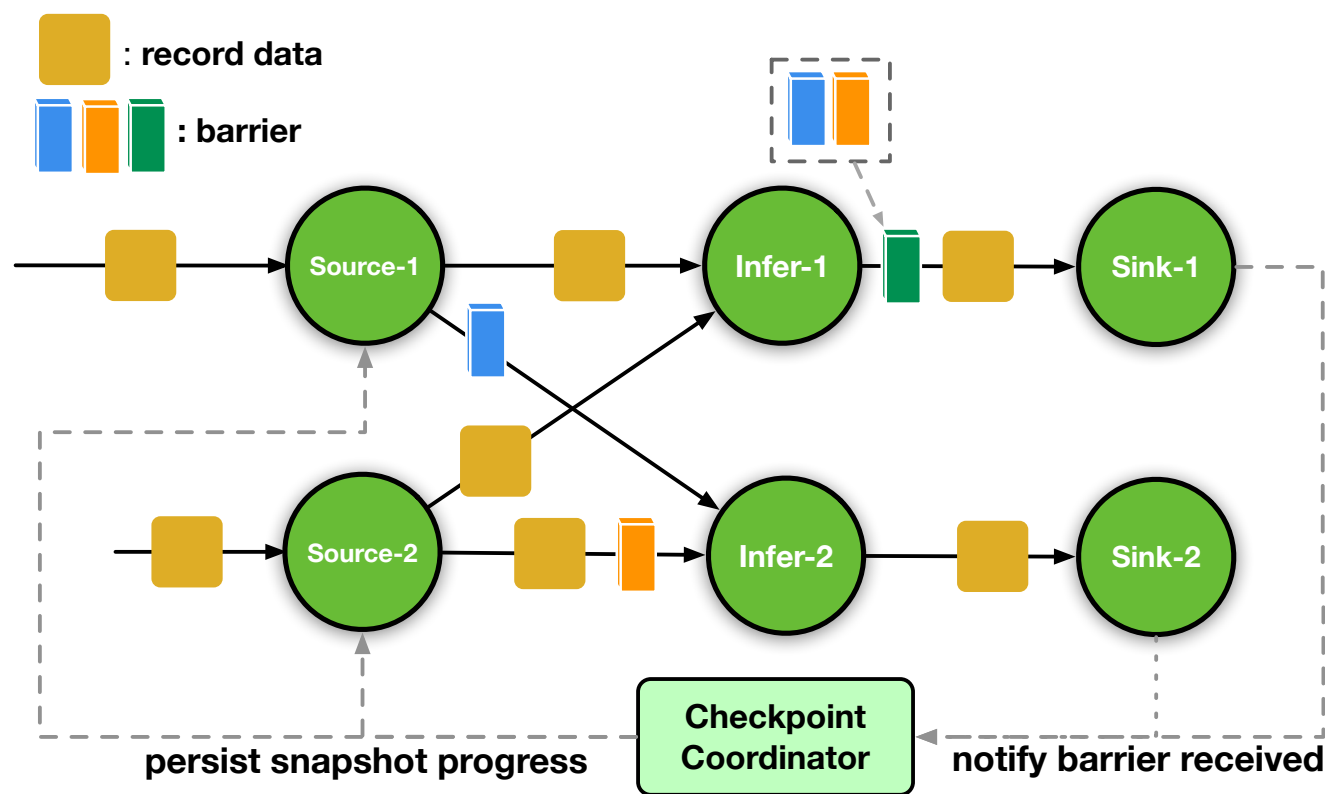
- Actor-based常驻实时算子



实时作业，需要算子常驻

# 如何实现流批一体的Ray Klein引擎

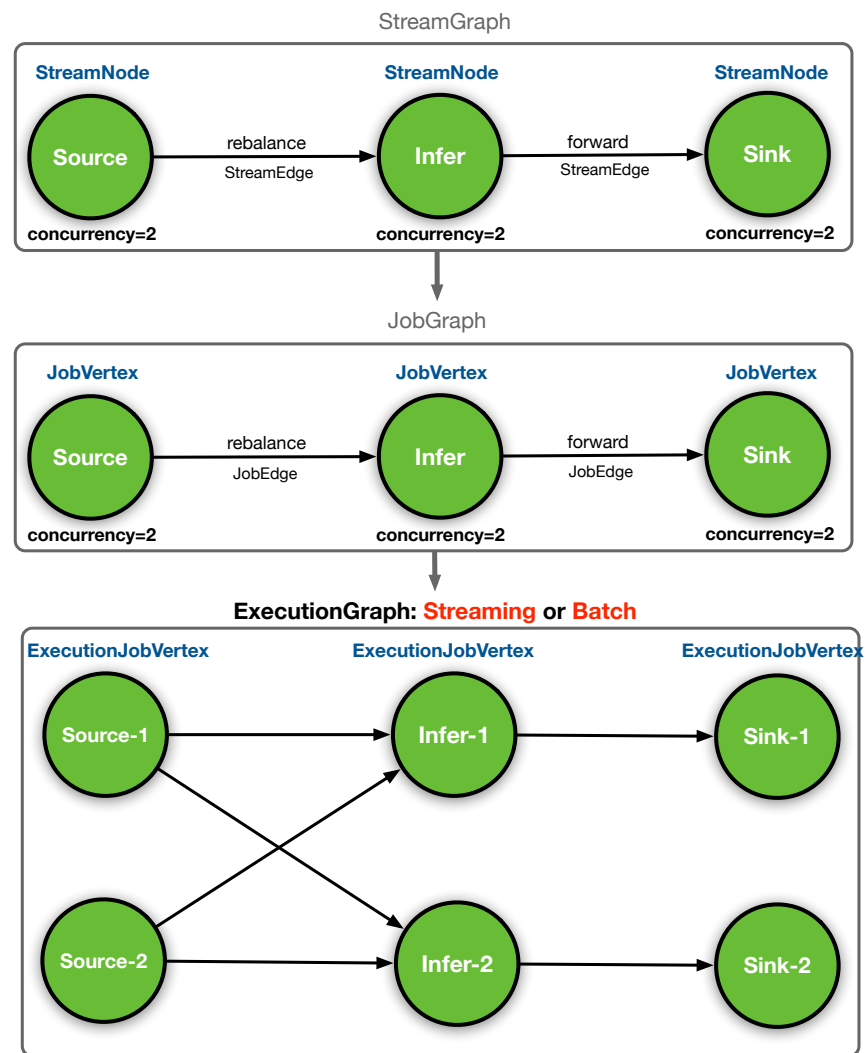
- 轻量版Chandy-Lamport算法



# 如何实现流批一体的Ray Klein引擎

- 流批一体的多层作业执行视图的转换

API上保障了使用的体验一致，根据是否是bounded source进行转换

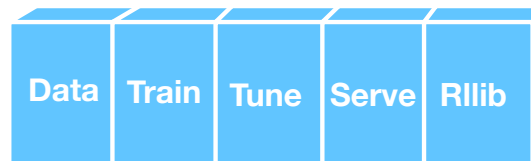


# 如何实现流批一体的Ray Klein引擎

Ray Klein可以替换Ray data并填补了社区版实时处理能力的空白

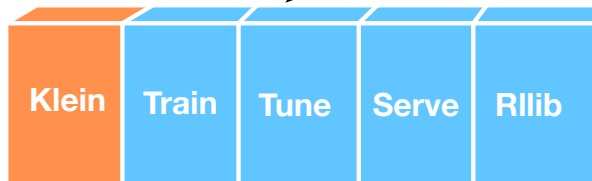
**Ray AI Libraries**  
High-level libraries that enable simple scaling of AI workloads

Low-level distributed computing framework with Python-first API



自研创新

Streaming &  
Batch unified





## 落地情况

- 落地了40+业务场景，场景覆盖：涉及搜索召回、搜索相关性、搜索质量、推荐召回、内容生态算法等子方向
- 在离线场景上，受益于异构计算，相比于之前的方案有性能提升
- 稳定性提升，相比原先单机solo等多种使用方式，具备更全面的监控和告警保障
- 开发成本降低：流批一体大大提高了开发效率

# 04

## 未来与展望

- Ray Klein在实时侧的不断打磨
- 探索更多Ray的使用场景
- Ray Klein项目开源计划