Flink 在字节跳动的实践和优化

Flink Practice and Optimization at ByteDance

张光辉 Guanghui Zhang
流式计算 技术负责人 Tech Lead of Streaming Compute

FLINK FORWARD # ASIA

实时即未来 # Real-time Is The Future

FLINK FORWARD

Contents
目录

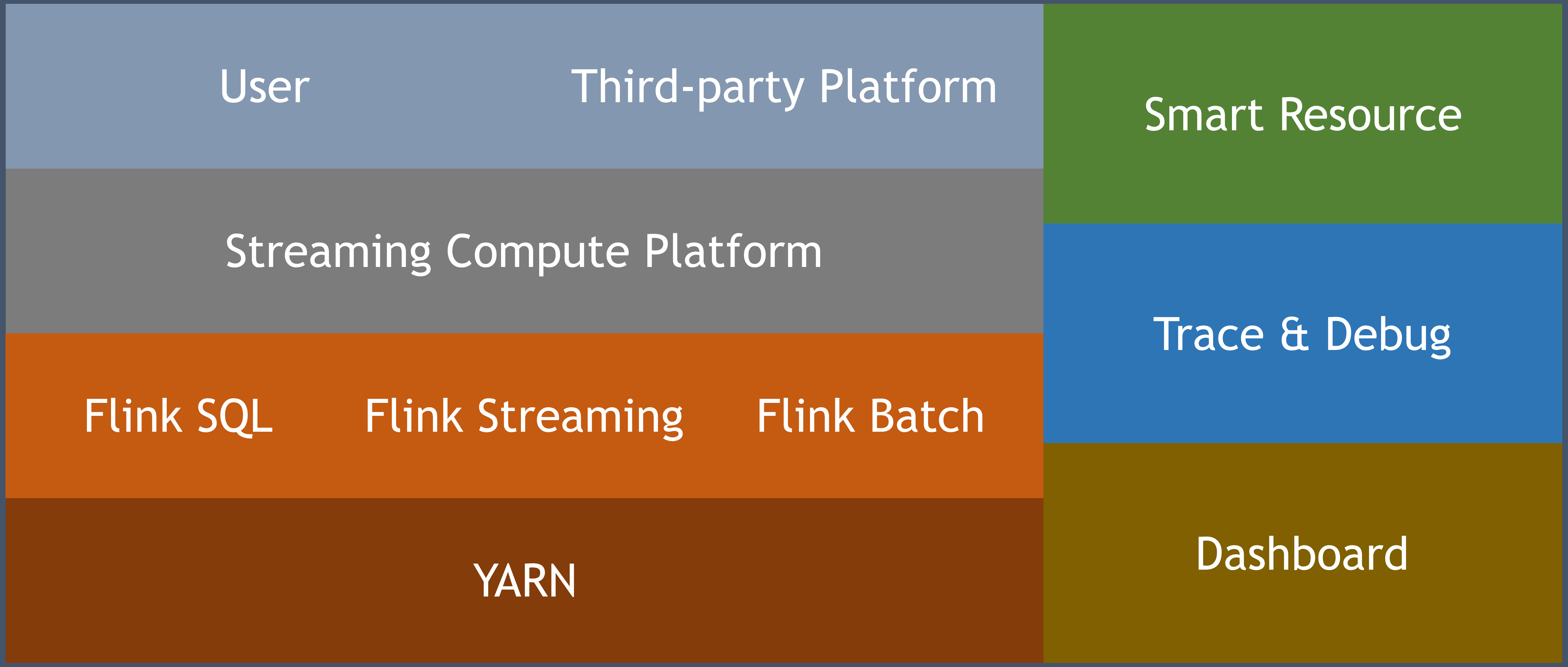
相关背景

Background

01

计算架构

Computing Architecture



计算规模

Scale of Computing

1s+ Yarn Cluster
1w+ Machines

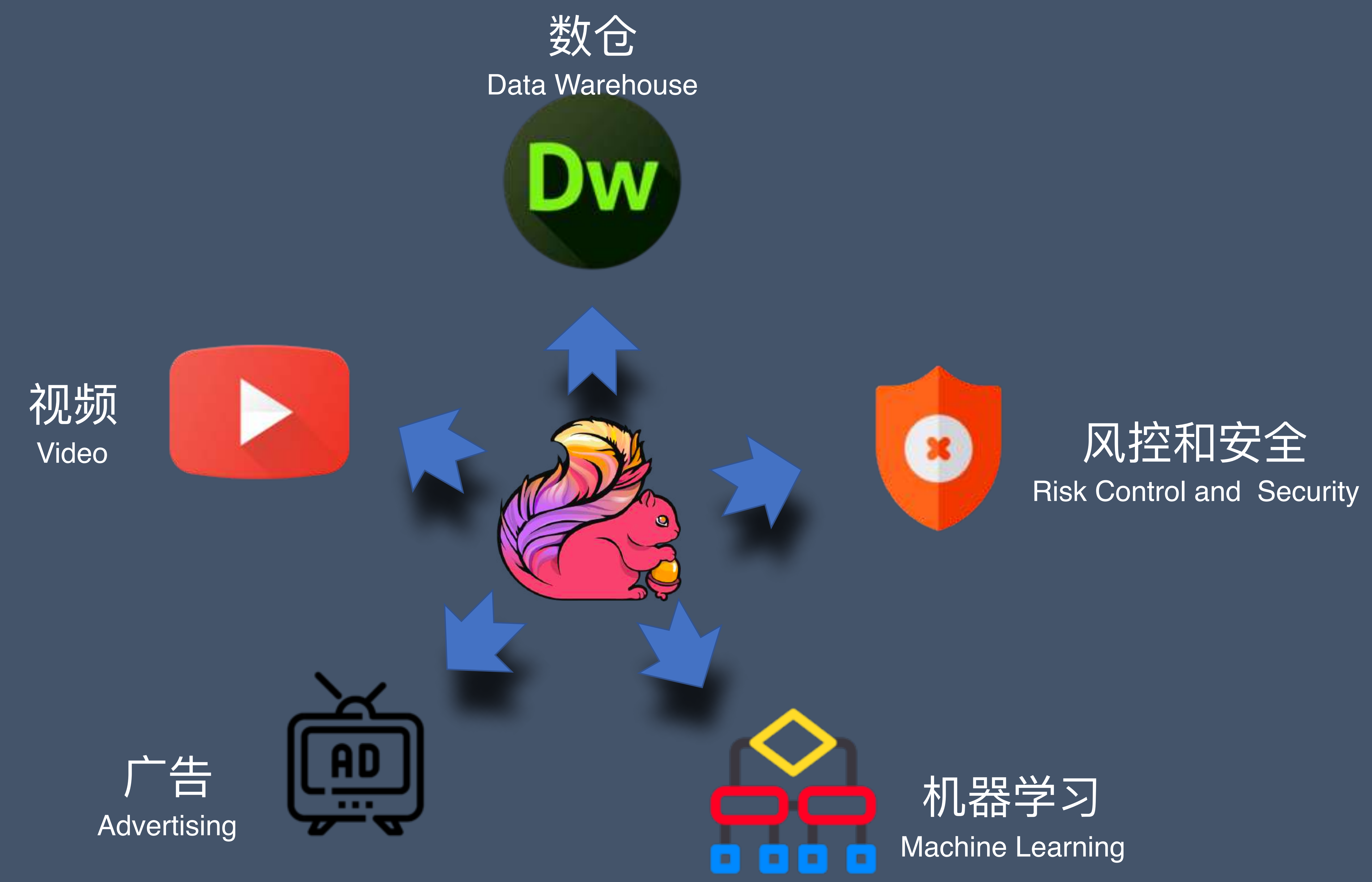
100s+ Users

1k+ Flink Streaming
1w+ Flink Batch
100s+ of Flink SQL

FLINK FORWARD

应用场景

Application Scenarios

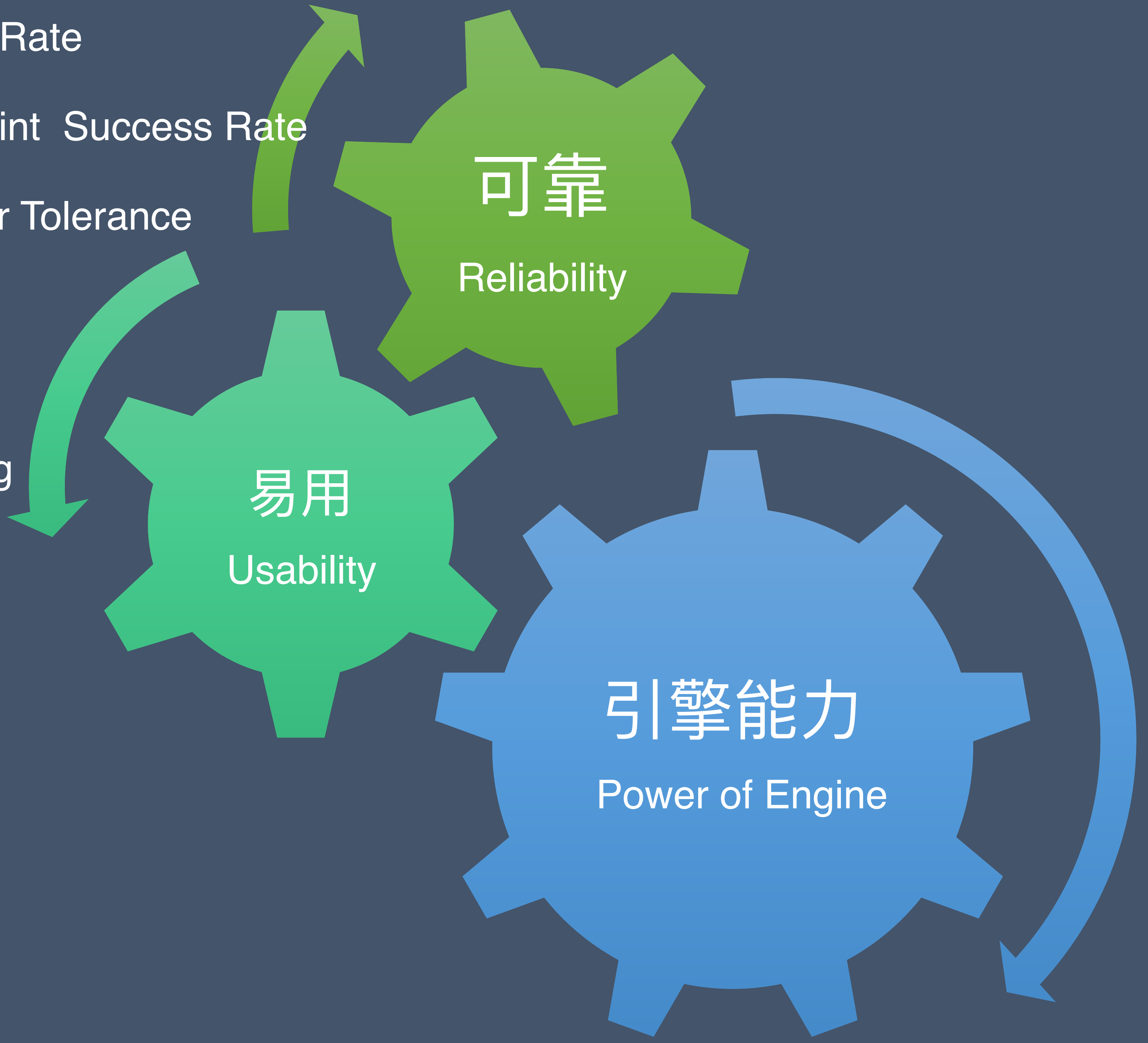


核心关注

Main Focuses

- 负载均衡 Load Balancing
- 环境隔离 Isolation
- 作业故障率 Failure Rate
- CP 成功率 Checkpoint Success Rate
- 双机房容灾 Disaster Tolerance

- 平台化 Platformization
- 便捷 Debug Easy Debug



- 状态计算 Stateful Computations
- 有且仅有 Exactly-Once
- 窗口 Windowing
- SQL SQL
- 批流统一 Unified Batch & Streaming

Flink SQL 的应用和扩展

Flink SQL Applications and Extensions

02

Flink 1.9 SQL 上线

Flink 1.9 SQL Goes Live

升级 Upgrade

- 升级 Flink 版本到 1.9
  Upgrade Flink version from 1.5 to 1.9, and cherry pick 200+ commits

验证 Verify

- 验证 Flink 1.9 SQL 功能
  Verify Flink 1.9 SQL features including DDL, DML, QUERY, UDX, CEP, EMIT, JOIN etc.

扩展 Extend

- 自定义 Watermark
  Customize watermark
- 计算列
  Computed column
- 多种 Source/ Sink/维表
  Multiple sources/ sinks/dimension tables

平台化 Platformization

- 集写 SQL，解析，测试，上线为一体的平台
  Unify SQL writing, analysis, testing, push online on one platform

上线 Go Live

- 上线
  Push Online
- 历史任务迁移
  Migrate tasks from 1.5 to 1.9
- 推广新业务
  Promote new business applications

Flink SQL 扩展

Flink SQL Extensions

- Source/Sink: Kafka, Redis, Mysql, ES, ClickHouse, Stdin/Stdout
- Dimension Table: Redis/MySQL
- Format: JSON, Protobuf, Binlog
- Create Function/View/Resource
- Customize Watermark
- Computed Column

Task 调度优化

Task Scheduling Optimizations

03

FLINK FORWARD

Flink 希望要一个什么样的调度器?

What Scheduler Does Flink Want?

分配 Container 时，尽可能选择 CPU/Mem 空闲的机器

When assigning containers, choose the idlest CPU/Mem machines as possible

针对 IO 密集型作业，希望 Container 尽可能打散

For I/O bound jobs, distribute containers as discrete as possible

Deploy Task 时，希望 Task 在 TM 中分配均衡

When deploying tasks, distribute tasks among TMs as balanced as possible

YARN Gang 调度器

YARN Gang Scheduler

- ➢ 一次分配请求返回所有或者没有资源

  All or nothing resources for an allocation request

- ➢ 为应用调度而不是为节点调度

  Schedule for application instead of node

  - 低延迟 (每次请求毫秒延迟)

    Low latency (RT in milliseconds per request)

  - 全局视野

    Global view

    - ◆ 强约束，例如：属性，负载

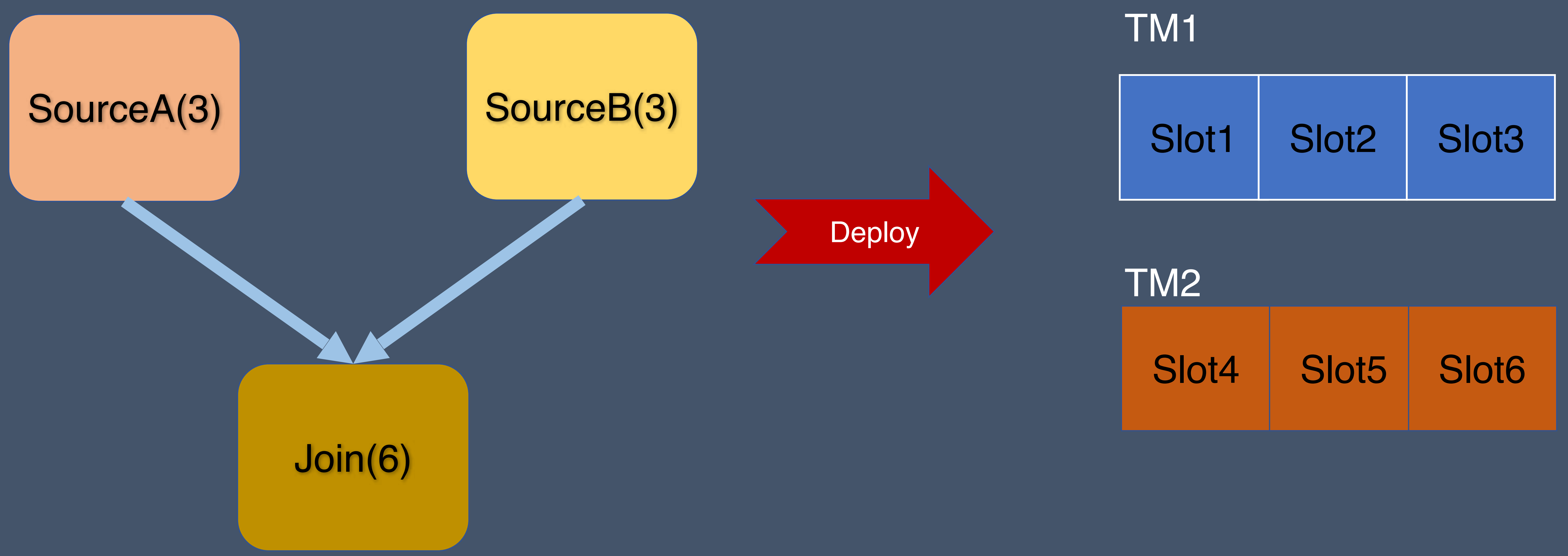
      Hard constraints, e.g., Attributes, Load avg…

    - ◆ 弱约束，例如：属性，负载，Container 打散，Quota 平均

      Soft constraints, e.g., Attributes, Load avg, Container Discretization, Quota avg, …

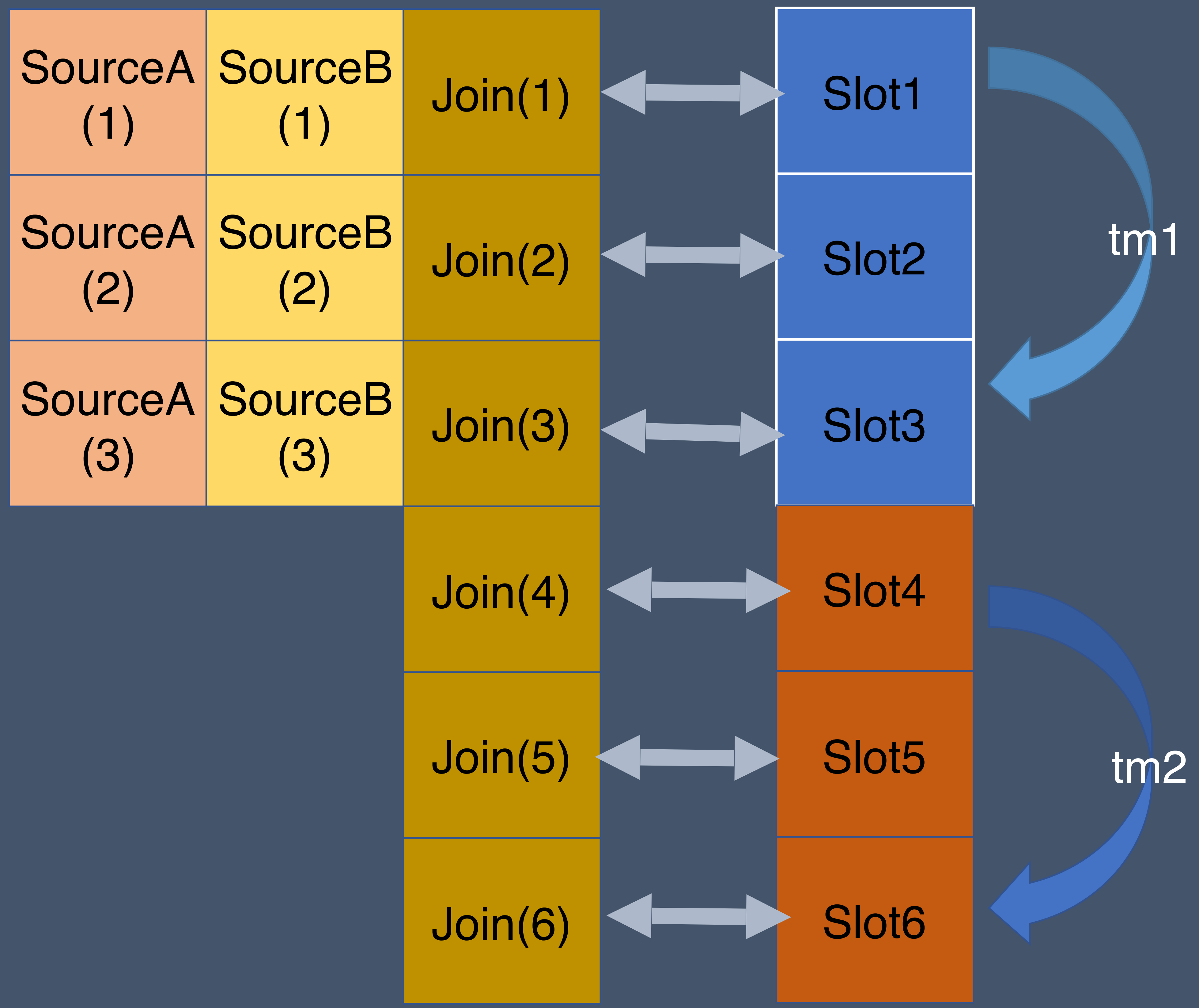
Task 均衡调度

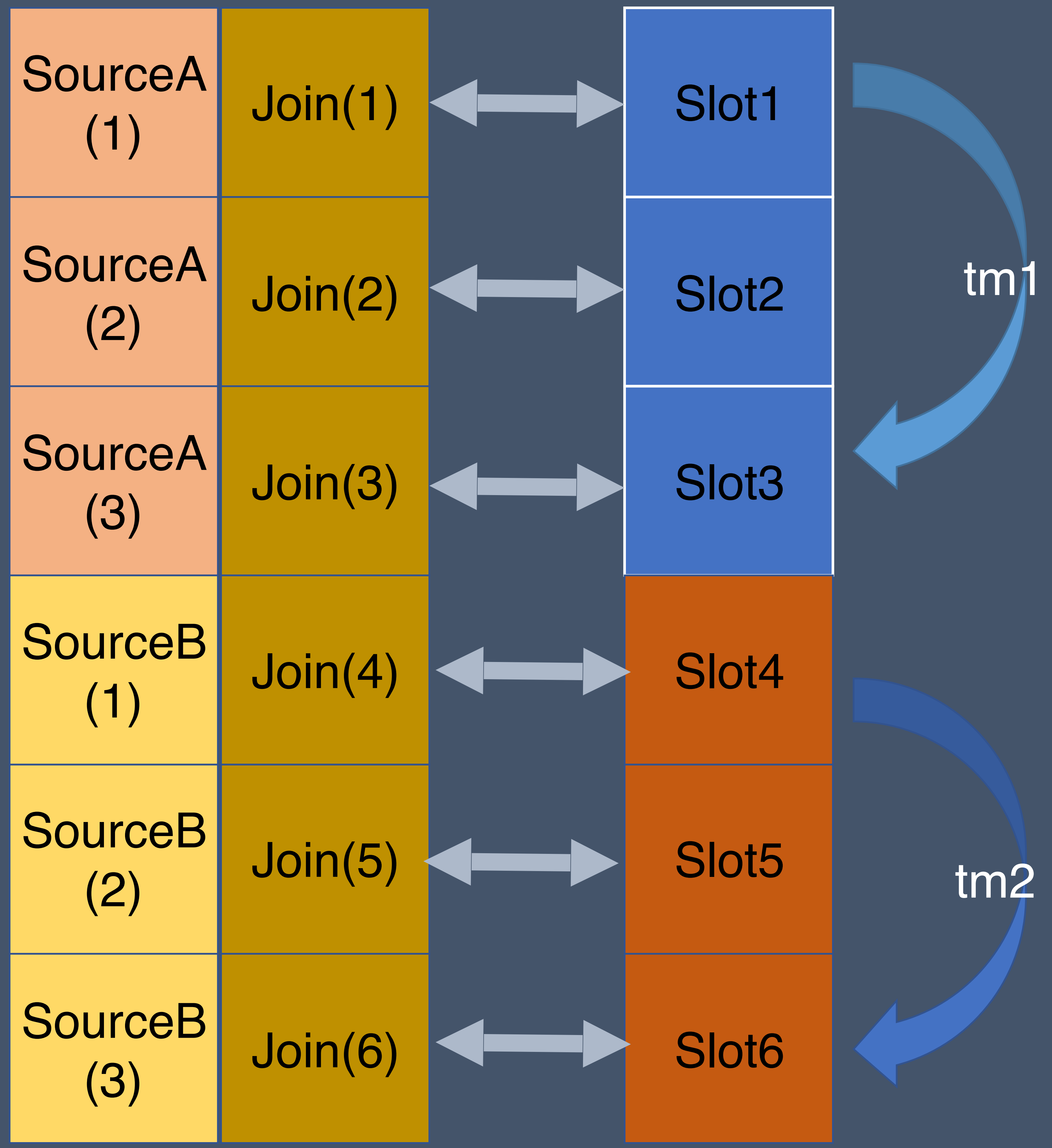
Task Load-Balanced Scheduling



Task 均衡调度

Task Load-Balanced Scheduling





Flink Docker 化

Flink Dockerization

04

Flink Docker 化

Flink Dockeralization

使用 Python udf，包依赖管理需求强烈

To use python udf, users need dependency management

磁盘空间隔离，/tmp，日志目录

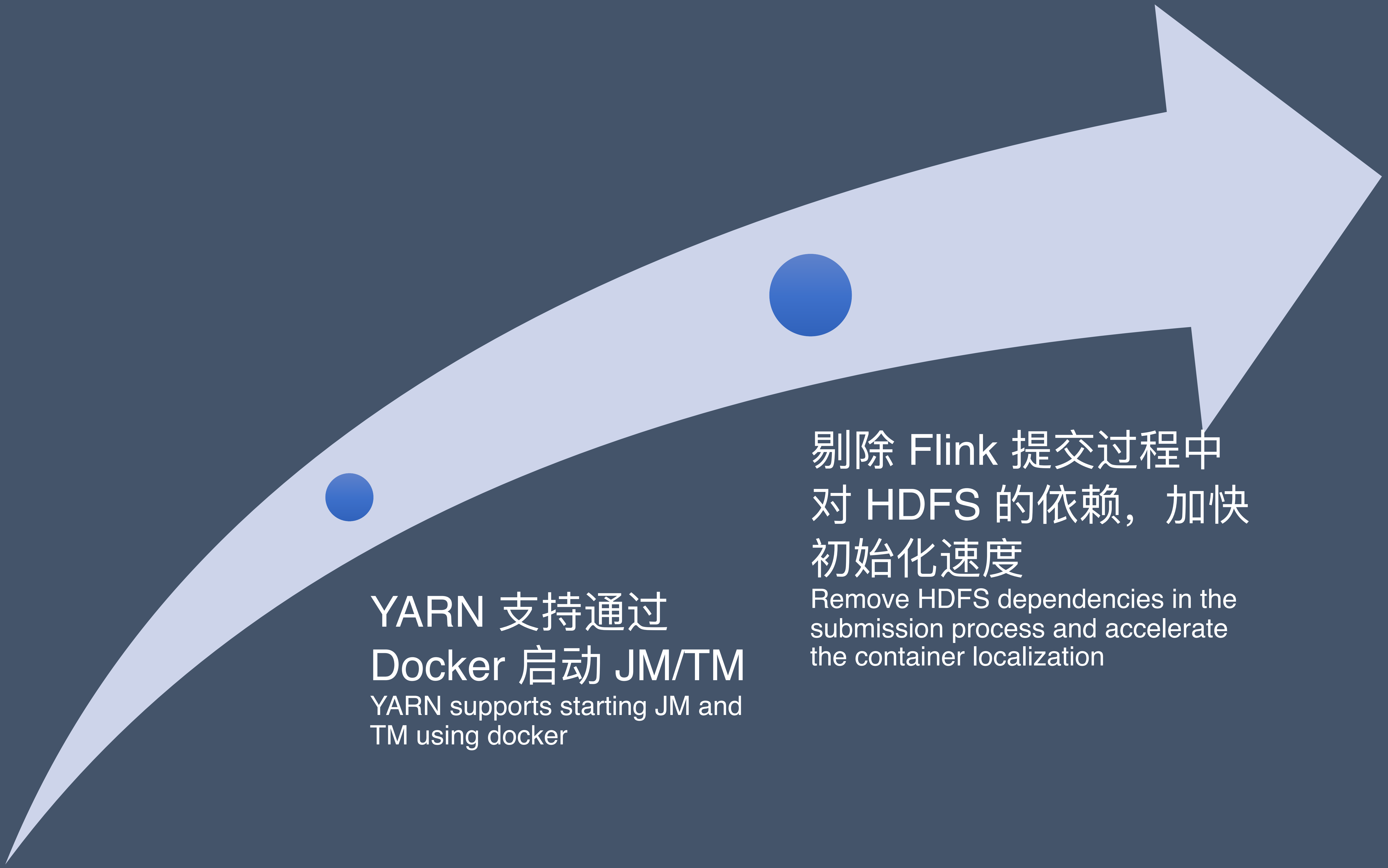
Disk space management, /tmp, log dir

镜像服务是 P2P 分发模式，加快 container 初始化速度

Image distribution service uses P2P, which accelerates the container localization

Flink Docker 化

Flink Dockeralization



YARN 支持通过 Docker 启动 JM/TM

YARN supports starting JM and TM using docker

剔除 Flink 提交过程中对 HDFS 的依赖，加快初始化速度

Remove HDFS dependencies in the submission process and accelerate the container localization

Flink 其他优化

Other Flink Optimizations

05

Checkpoint 使用优化

Checkpoint Usage Optimization

Checkpoint 不能跨应用，当作业重启，无法默认从上一个应用的 Checkpoint 中恢复

Checkpoint does not cross applications. When a job restarts, it can not recover from the checkpoint of the previous invocation.

作业重启默认从 Checkpoint 恢复

When a job restarts, it now by default recovers from the checkpoint of the previous invocation.

JM 异步并发启动 TM

JobManager Starts TaskManager Asynchronously and Concurrently

串行启动 TM，速度慢
Sequentially starting TMs takes long time

慢节点
Slow nodes

异步接口启动
Start TMs asynchronously

多线程并发
Multi-threaded concurrency

超时重试，黑名单机制
Timeout retry and blacklist mechanism

Client/Cluster 类加载机制不同

Different Class Loading Mechanisms between Client and Cluster

Client 端类加载机制与 Cluster 端不同

The class loader used by the client is different from the default one used by the cluster

修改 client 端支持 child-first/ parent-first 类加载机制可配置

The class loader used by the client supports configurable child-first/parent-first class loading mechanisms.

Netty Client/Server 缺失心跳机制

Netty Client/Server Lacks of Heartbeat Mechanism

Netty 是异步的，需要应用层心跳机制来进行探活

Netty is asynchronous and requires heartbeat mechanism at application layer to detect liveness

利用 Netty IdleStateHandler 建立心跳机制

Leverage the Netty IdleStateHandler to establish the heartbeat mechanism

JM 资源可配置

Configurable JobManager Resources

jm vcore 默认为 1， jm mem 与 tm mem 保持一致

The vcore limit of the job manager is 1 by default and the memory limit of the job manager is same as the task manager’s

jm vcore, jm mem 支持单独配置

The vcore and memory limits of the job manager can be configured independently

支持千分之一核

Support Milli-vcores

Vcore 个数必须指定整数，资源会有所浪费

The number of Vcores must be an integer, which is a waste sometimes

Vcore 支持千分之一核

Support milli-vcores to reduce waste of CPUs

Flink 在字节跳动未来展望

Outlook of Flink at ByteDance

06

未来展望

Outlook

Flink Streaming

- 推动所有 Flink Job Docker 化
  Migrate all Flink jobs to run in Docker
- 支持 Flink Job 横向扩展
  Support Flink jobs auto scaling

Flink SQL

- 推动未来一年 SQL 任务达到 50%
  Flink SQL reaches 50% over all Flink applications
- 探索 Streaming/Batch 应用场景
  Explore the applications of unified batch and streaming

Flink Batch

- 推测执行，提升稳定性
  Support speculative execution, improve stability
- Flink Gelly 支持全图离线计算应用场景
  Large-scale graph compute with Flink Gelly

Q&A

FLINK FORWARD

致谢

Thanks

感谢数据开发套件、计算架构团队的小伙伴，以上分享内容是我们共同努力的成果

Thanks my colleagues of the data development kit team and the compute infrastructure team @ ByteDance, the above content is the result of our joint efforts