阿里巴巴在Flink大规模持久化存储的实践之道

Zen of Large-Scale Storage when Alibaba Meets Apache Flink

唐云 Yun Tang

chagan.ty@alibaba-inc.com
阿里巴巴高级工程师

FLINK FORWARD # ASIA
实时即未来 # Real-time Is The Future

FLINK FORWARD

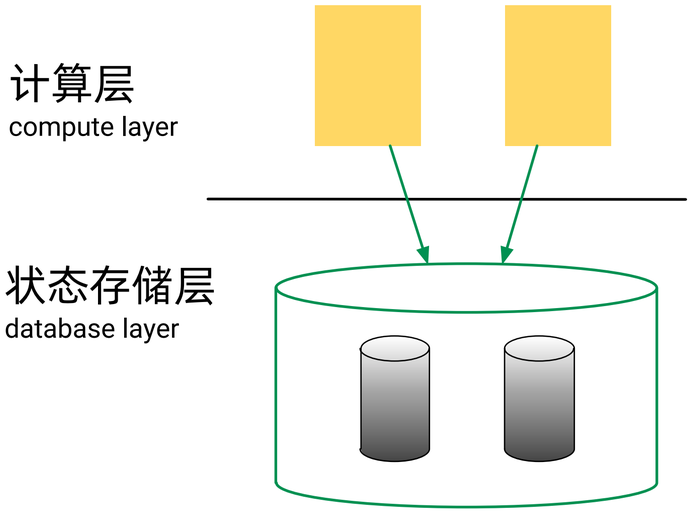
目录
Contents

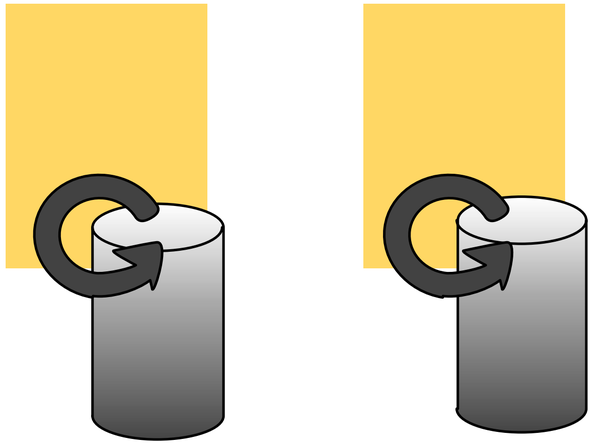
Why we care large-scale storage in Flink?

外部状态存储 vs. 内部状态存储 External vs. Internal

- 外部状态 External State
  - 状态存储在独立的数据库中
    state in separate data store
  - 比内部状态读写慢 slower than local state
  - 数据大规模时一致性代价更大
    Consistency becomes complex and expensive at scale

- 内部状态 Internal state
  - 状态存储在流计算处理算子中
    State in the stream processor
  - 本地状态读写更快 State is local & much faster
  - 容易实现数据一致性
    Exactly-once guarantees aren't expensive
  - 需要能很好的保证状态的可用性，可扩缩容和可维护
    Need to manage making the state durable, scalable, evolvable.



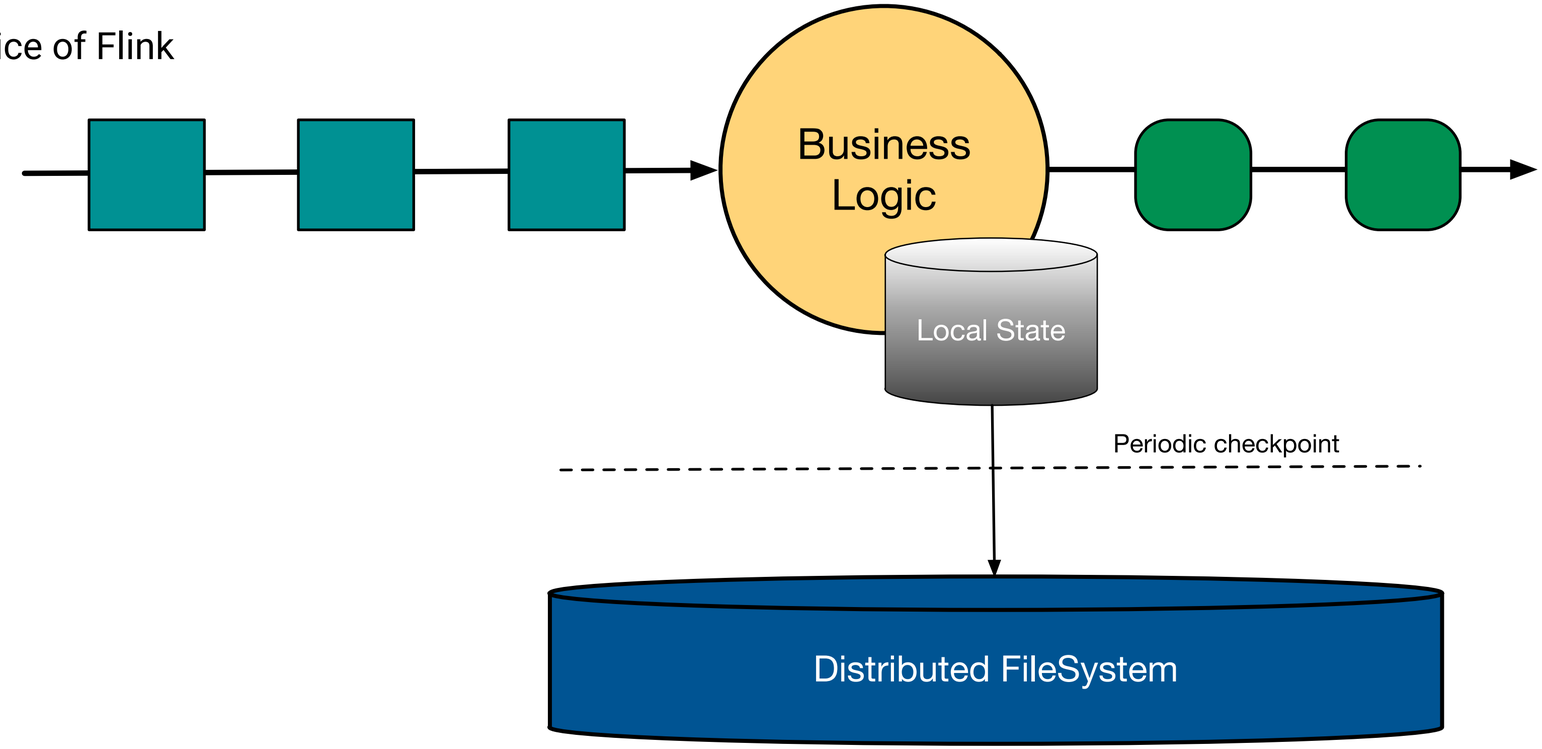


Why we care large-scale storage in Flink?

FLINK FORWARD

Flink的选择

The Choice of Flink

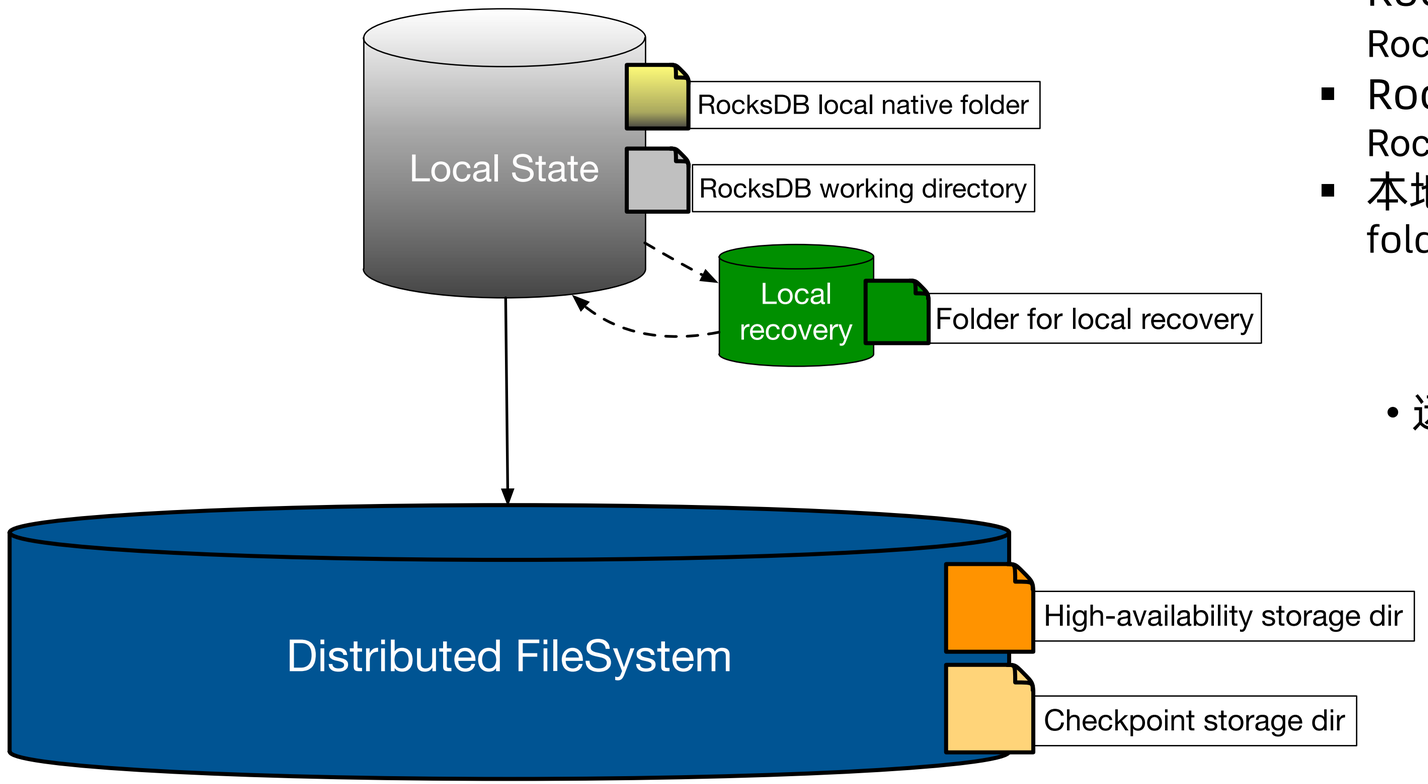


Actually, other streaming computing systems would also adopt to use local state to refactor previous design.

Why we care large-scale storage in Flink?

Recap: Flink的数据存储目录

Recap: Data storage dir for Flink



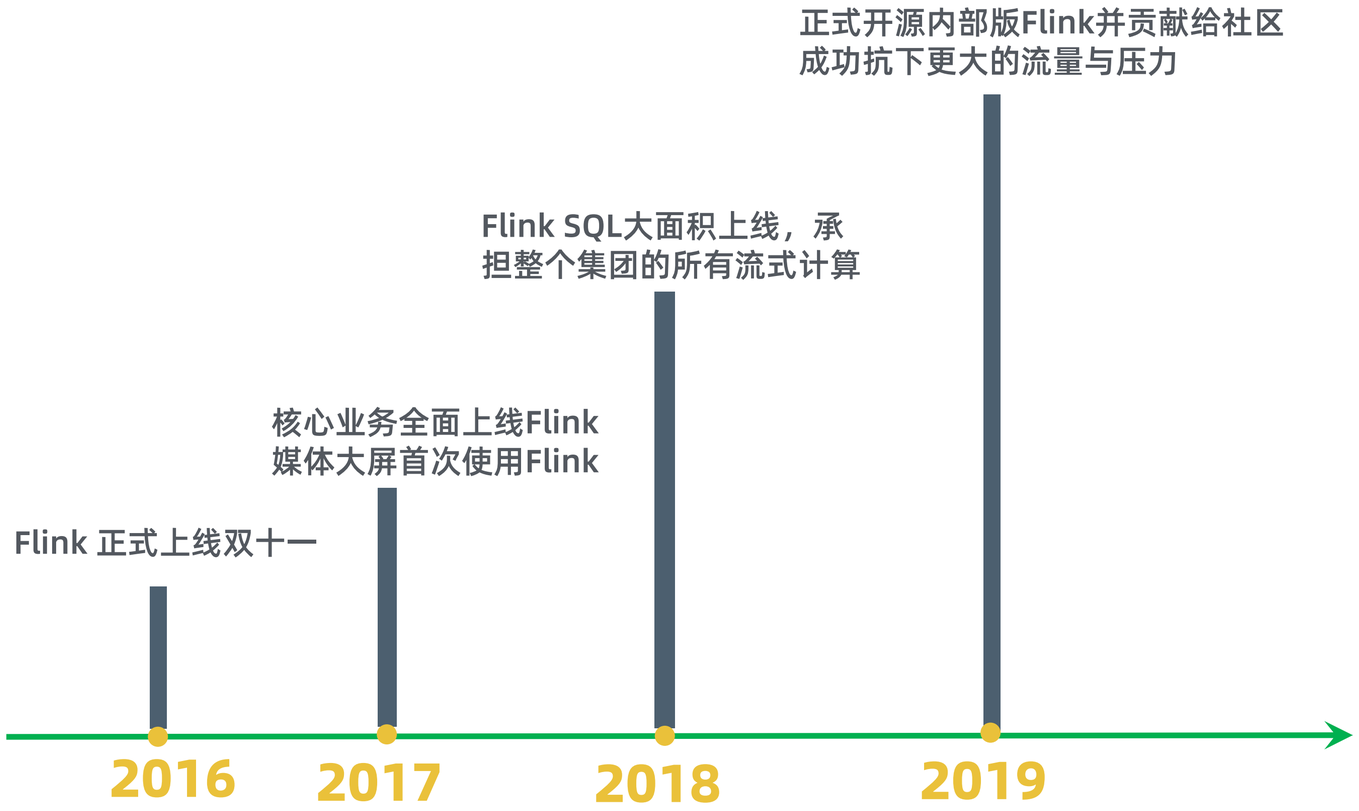
- 本地存储交互目录 Local storage directory
  - RocksDB的本地JNI加载目录
    RocksDB local native folder
  - RocksDB的工作目录
    RocksDB working directory
  - 本地恢复时的备份目录
    folder for local recovery

- 远程存储交互目录 Remote storage directory
  - 高可用系统的存储目录
    High-availability storage directory
  - Checkpoint的存储目录
    Checkpoint storage directory

目录
Contents

阿里巴巴双十一历练下的存储之道

Zen of large-scale storage when Flink meets Alibaba global shopping festival



More throughputs, more responsibility on Flink

What we have learned from recent years?

- **点到为止**

  Avoid unnecessary interaction

- **“清理门户”**

  Resource Release in Time

- **化大为小**

  Make bigger things smaller

阿里巴巴双十一历练下的存储之道

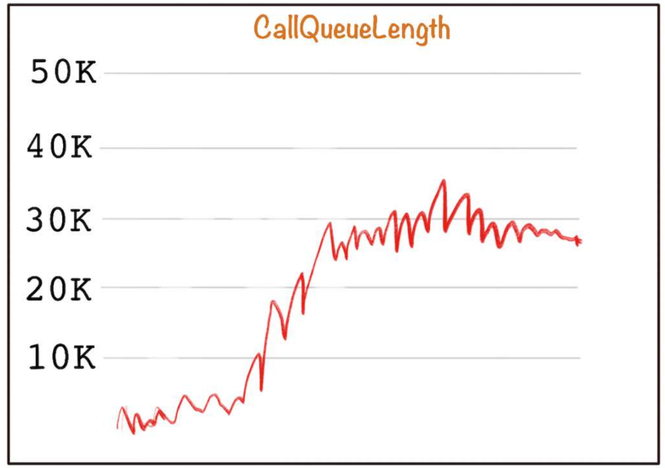
Zen of large-scale storage when Flink meets Alibaba global shopping festival

FLINK FORWARD

Problem-1

HDFS 响应慢，RPC 请求堆积大

HDFS too slow, call queue length too long



FSNReadLockListStatusNanosAvgTime

ListStatus 请求太多，抢占FSNamesystemLock 的读锁，导致堆积

FileUtils#deletePathIfEmpty

```
public static boolean deletePathIfEmpty(FileSystem fileSystem, Path path) throws IOException {
    final FileStatus[] fileStatuses;

    try {
        fileStatuses = fileSystem.listStatus(path);
    }
    catch (FileNotFoundException e) {
        // path already deleted
        return true;
    }
```

阿里巴巴双十一历练下的存储之道

Zen of large-scale storage when Flink meets Alibaba global shopping festival

FLINK FORWARD

Problem-1

HDFS 响应慢，RPC 请求堆积大: list请求数过多

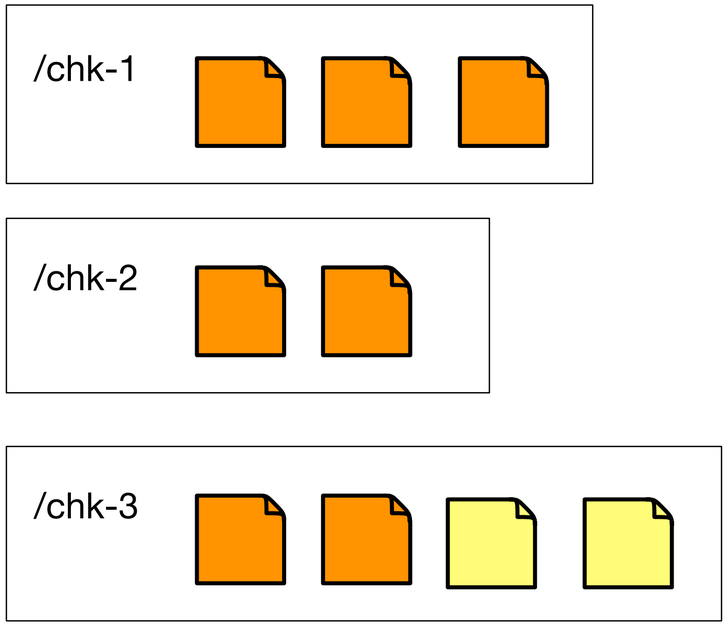
HDFS too slow, call queue length too long: Too much list status request

Before Flink-1.5

Checkpoints layout

:shared state

:exclusive state



以前Flink是将share state和exclusive state混合存储在每个checkpoint目录下。
删除旧的checkpoint时，需要判断目录是否为空才能删除chk父目录

As Flink mixed shared and exclusive state previously, it need to know whether parent folder is empty, so that the parent chk-id folder could be removed safely.

拆分shared 和 exclusive state目录，使得不再需要无谓的list请求

Separate shared and exclusive state folder to avoid unnecessary list status requests.

Refer to community's solution: FLINK-7266 FLINK-8540

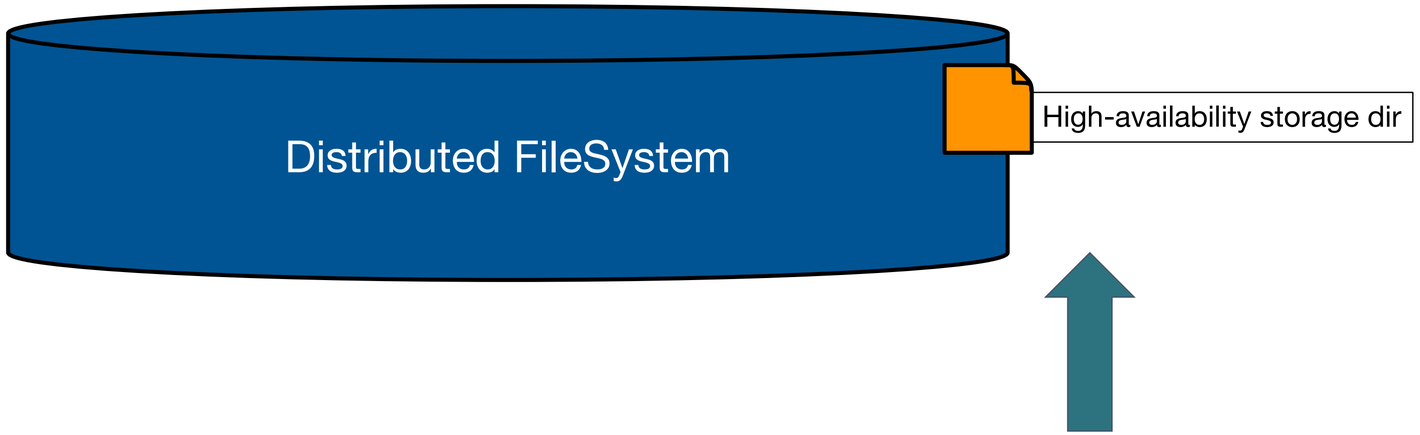
阿里巴巴双十一历练下的存储之道

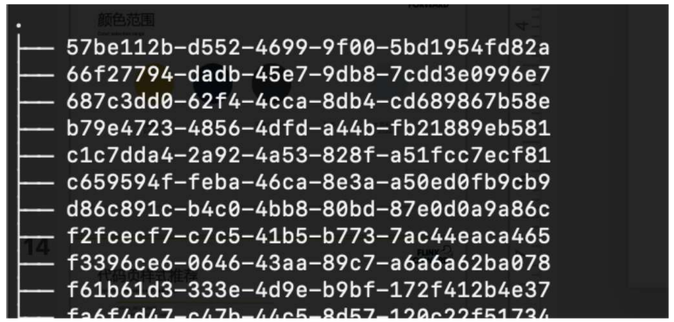
Zen of large-scale storage when Flink meets Alibaba global shopping festival

FLINK FORWARD

Problem-2.1 HA 目录被写满

High-availability storage directory if full





阿里率先使用流处理去实现批处理时，默认的MemoryStateBackend在特定情况下会在HA目录创建大量的UUID子目录。

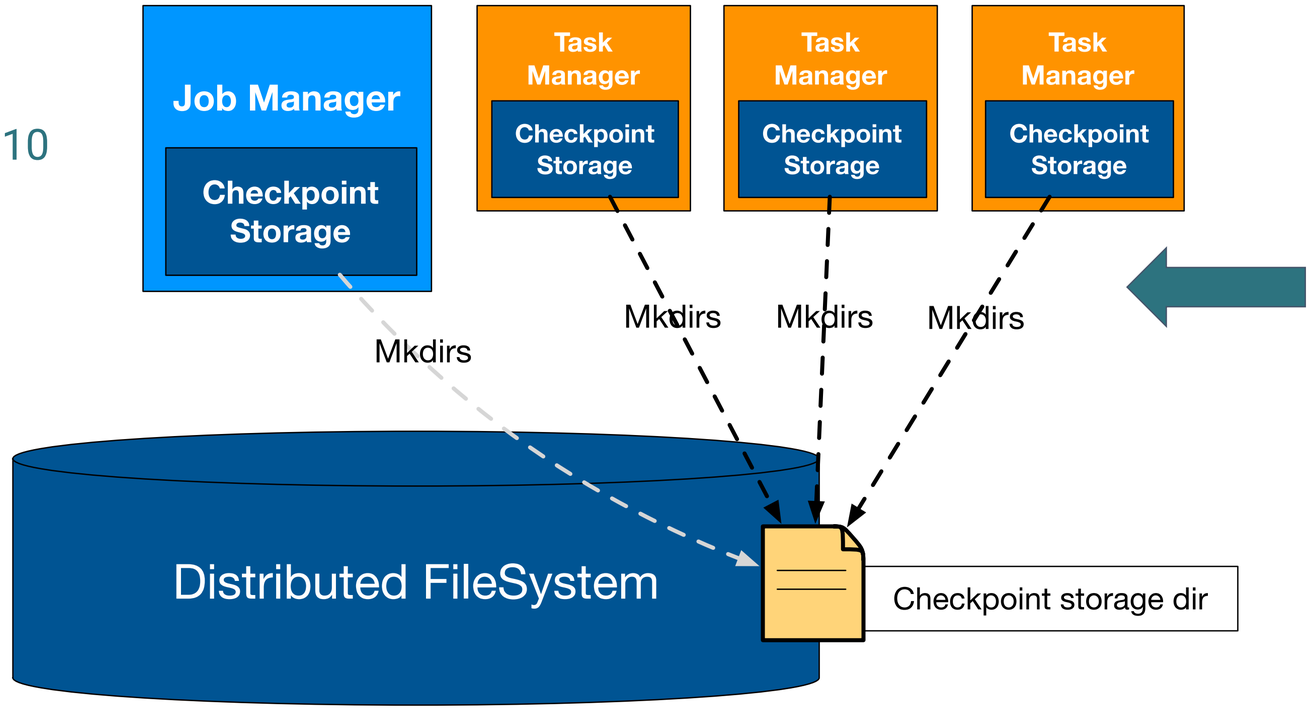
MemoryStateBackend would crate a lot of sub-directory under HA path under specific situation.

Refer to solution FLINK-11107, contributed from Alibaba

Problem-2.2 作业启动/failover时，总是有大量的Mkdirs请求

Massive Mkdirs requests when job submit/failover

Before Flink-1.10



JobManager和TaskManager内的checkpoint storage职责含混不清，重复创建Mkdirs请求。

Mixed responsibility for checkpoint storage within JobManager and TaskManager, created duplicate ‘Mkdirs’ requests

阿里巴巴双十一历练下的存储之道

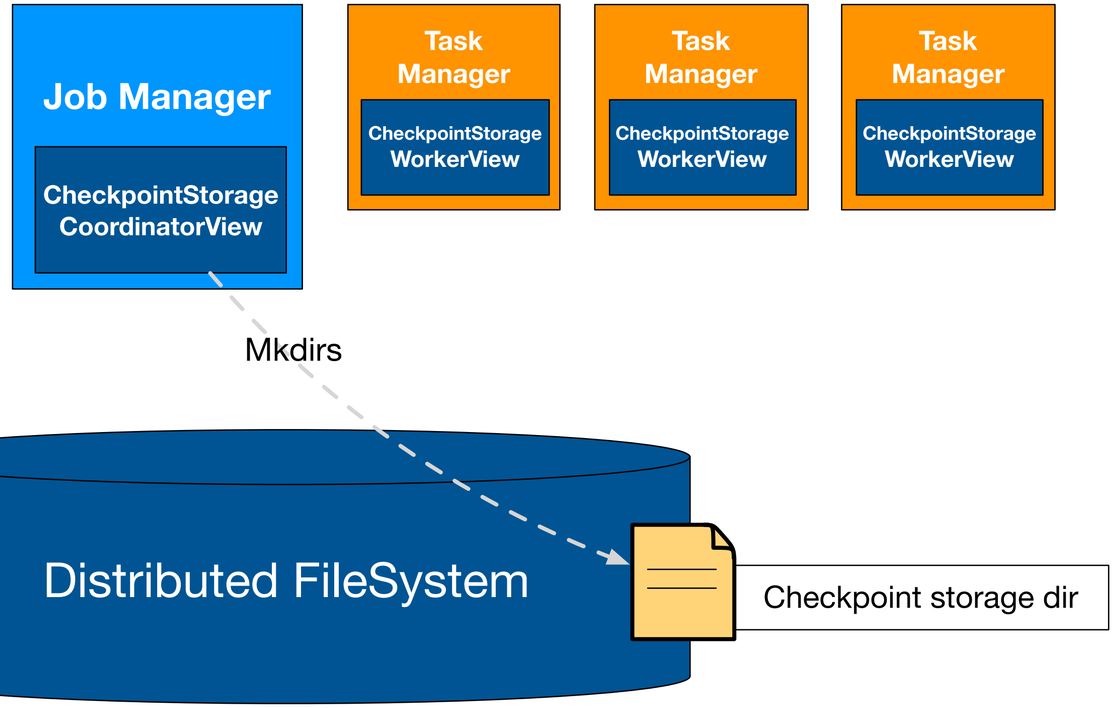
Zen of large-scale storage when Flink meets Alibaba global shopping festival

FLINK FORWARD

Problem-2.2 作业启动/failover时，总是有大量的Mkdirs请求

Massive Mkdirs requests when job submit/failover

After Flink-1.10



Checkpoint storage →

Checkpoint Storage Coordinator view
+
Checkpoint Storage Worker view

Refer to solution FLINK-11874 and FLINK-11696, contributed from Alibaba

点到为止

Avoid unnecessary interaction

- 只有在大规模体量时，这些多余的交互才会导致问题

  Only when we have large-scale jobs, those redundant interactions could cause problems.

- 在与外部系统服务打交道的过程中，永远都是在“过度使用”和“限制使用”中做权衡

  We always need to make the balance between ‘overuse’ and ‘restriction’ when interacts with external systems.

阿里巴巴双十一历练下的存储之道

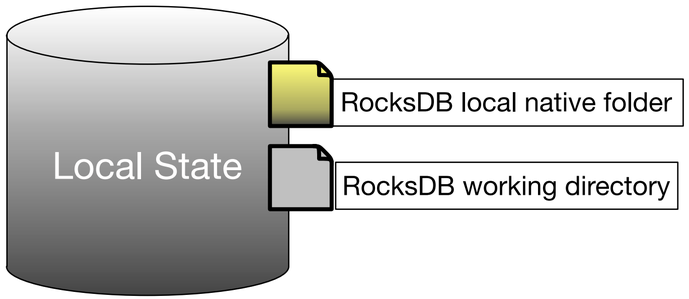
Zen of large-scale storage when Flink meets Alibaba global shopping festival

FLINK FORWARD

Problem-3

个别机器的inode被打满：过多的native库加载目录

Inode is out of usage on some specific machine ： too many local native load library directory



加载的库目录依赖于container的退出清理，
而没有在加载失败之后及时清理
Not clean failed native library directory in time.

Refer to solution FLINK-14378 contributed from Alibaba

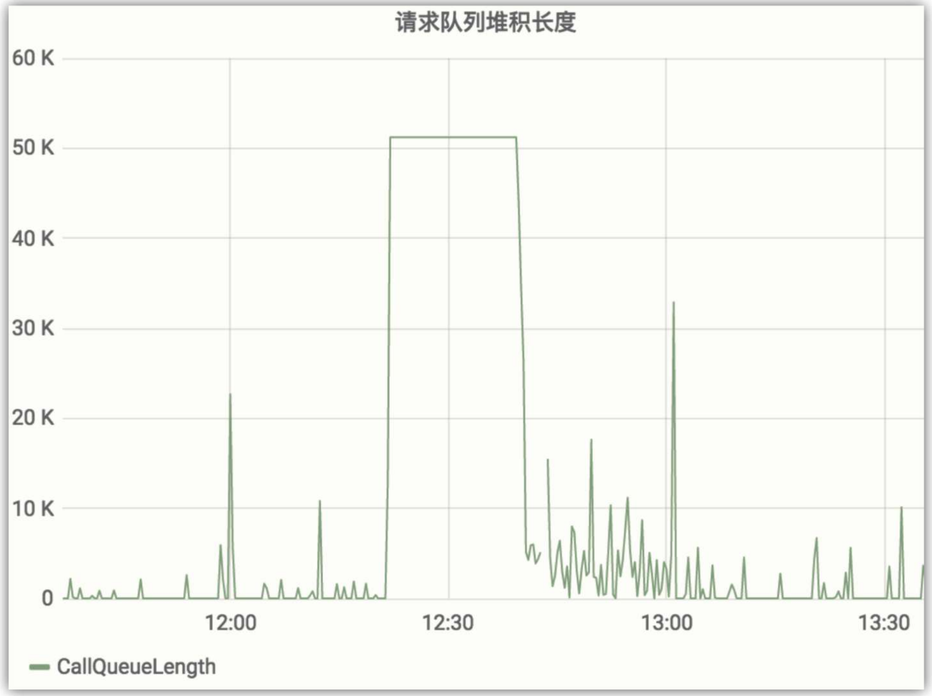
阿里巴巴双十一历练下的存储之道

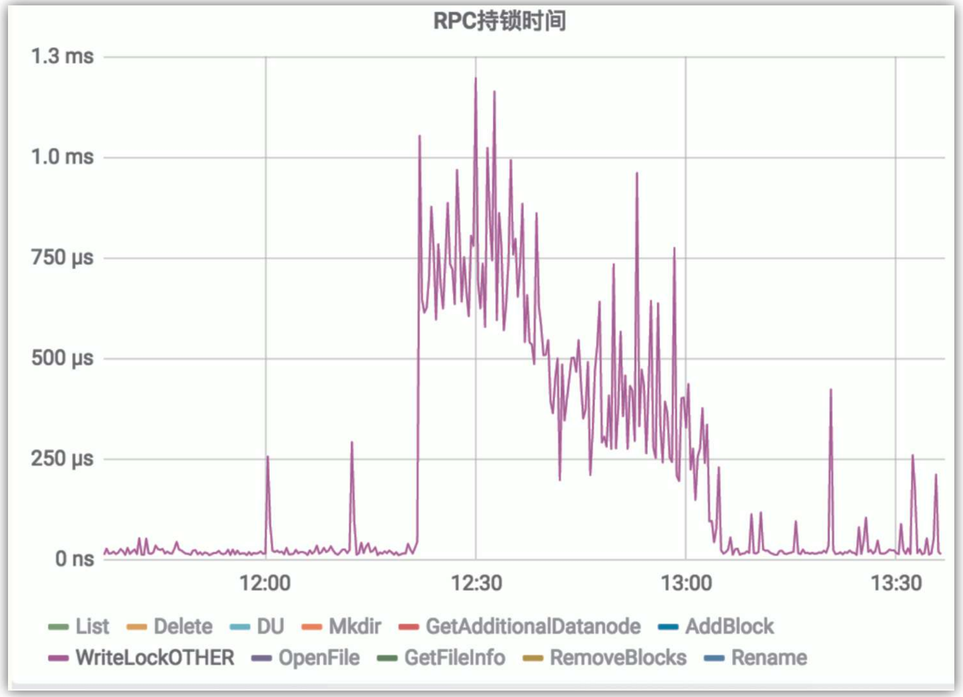
Zen of large-scale storage when Flink meets Alibaba global shopping festival

FLINK FORWARD

Problem-4.1 HDFS 因为激增的创建block请求而无法响应

HDFS has no response due to booming write block requests



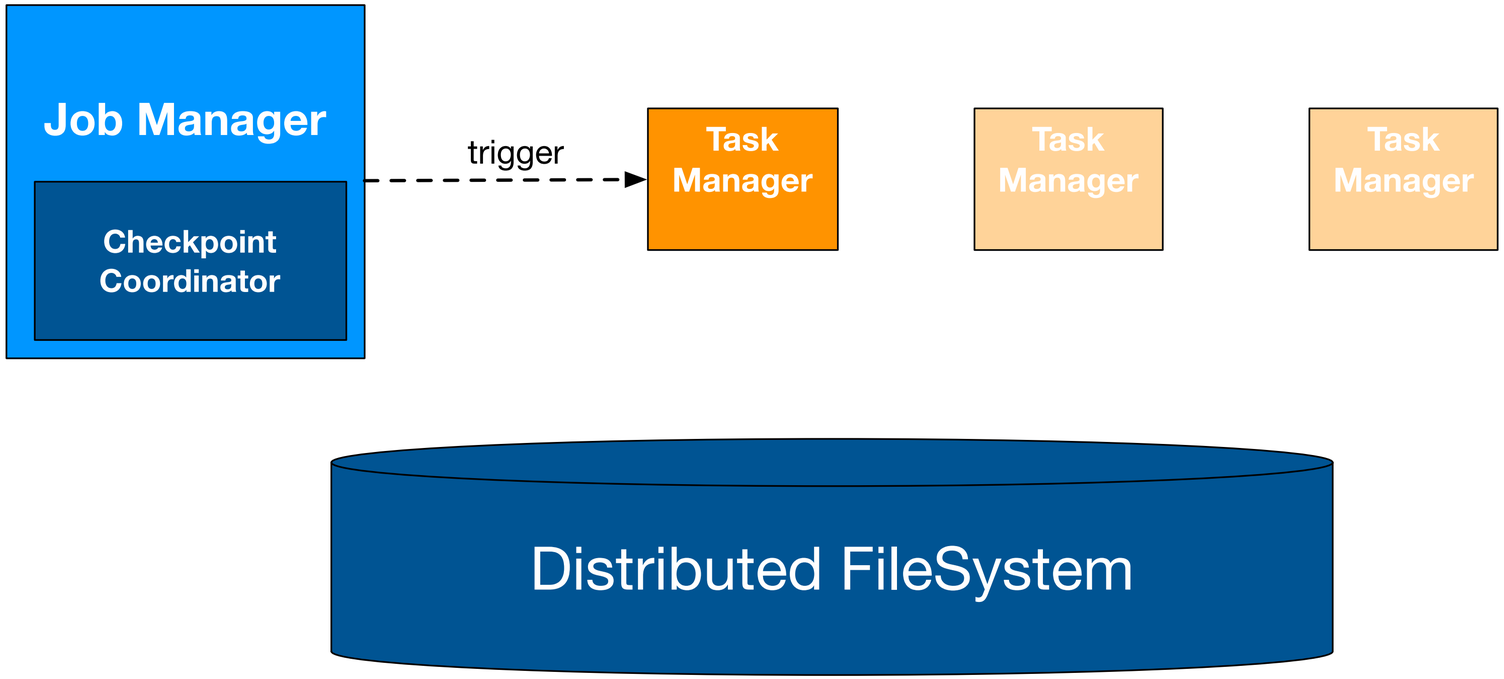


Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (within timeout duration)

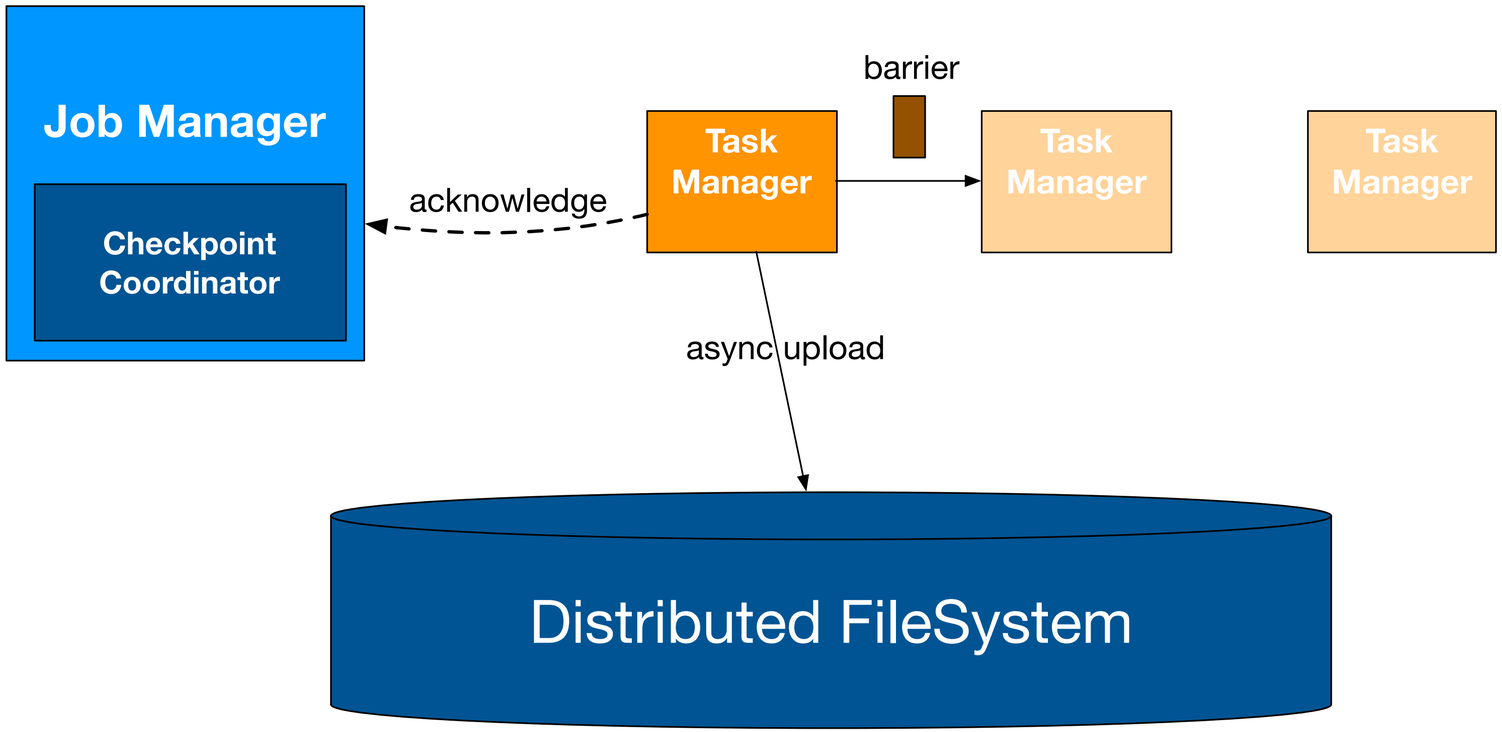


Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (within timeout duration)

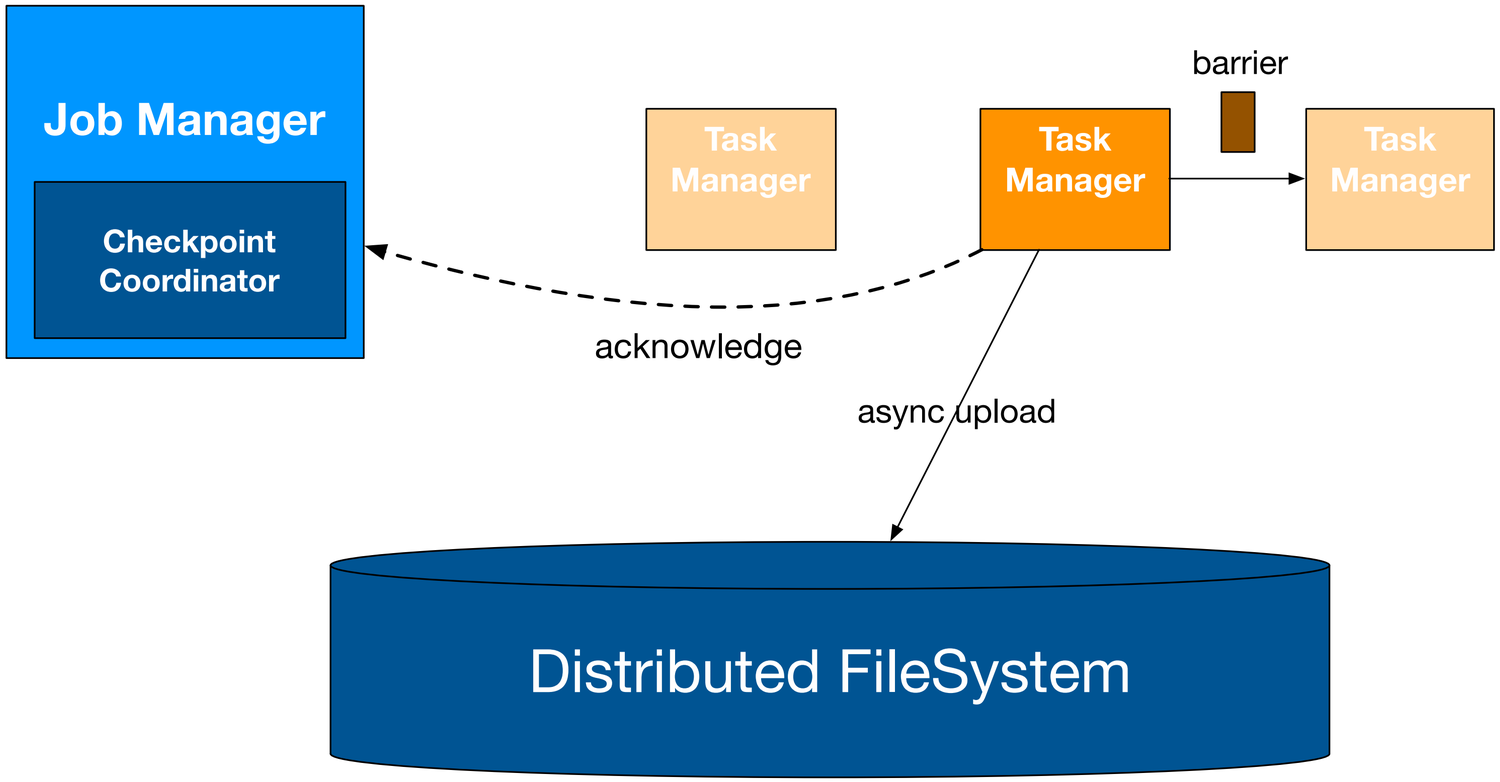


Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (within timeout duration)

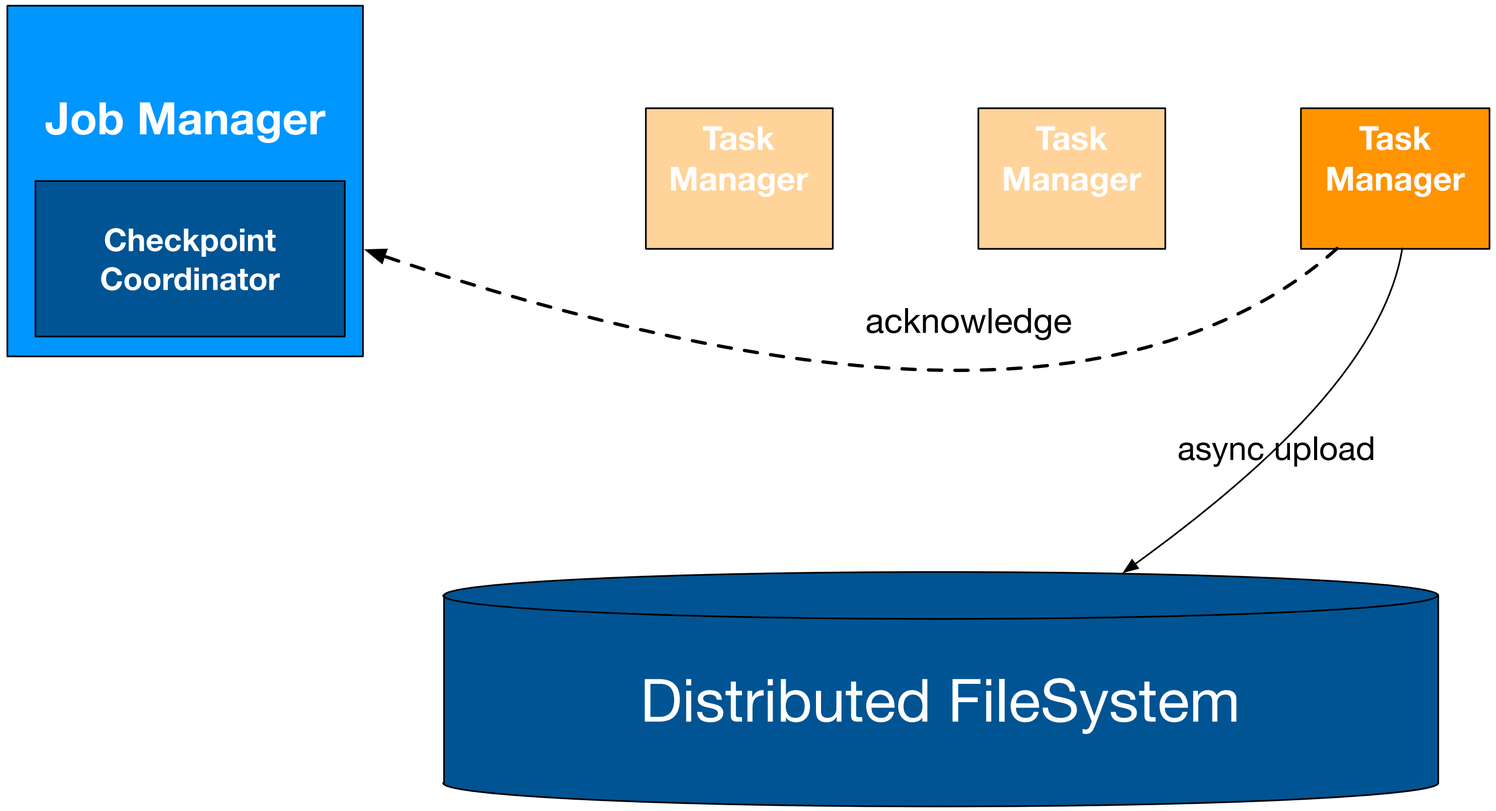


Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (within timeout duration)

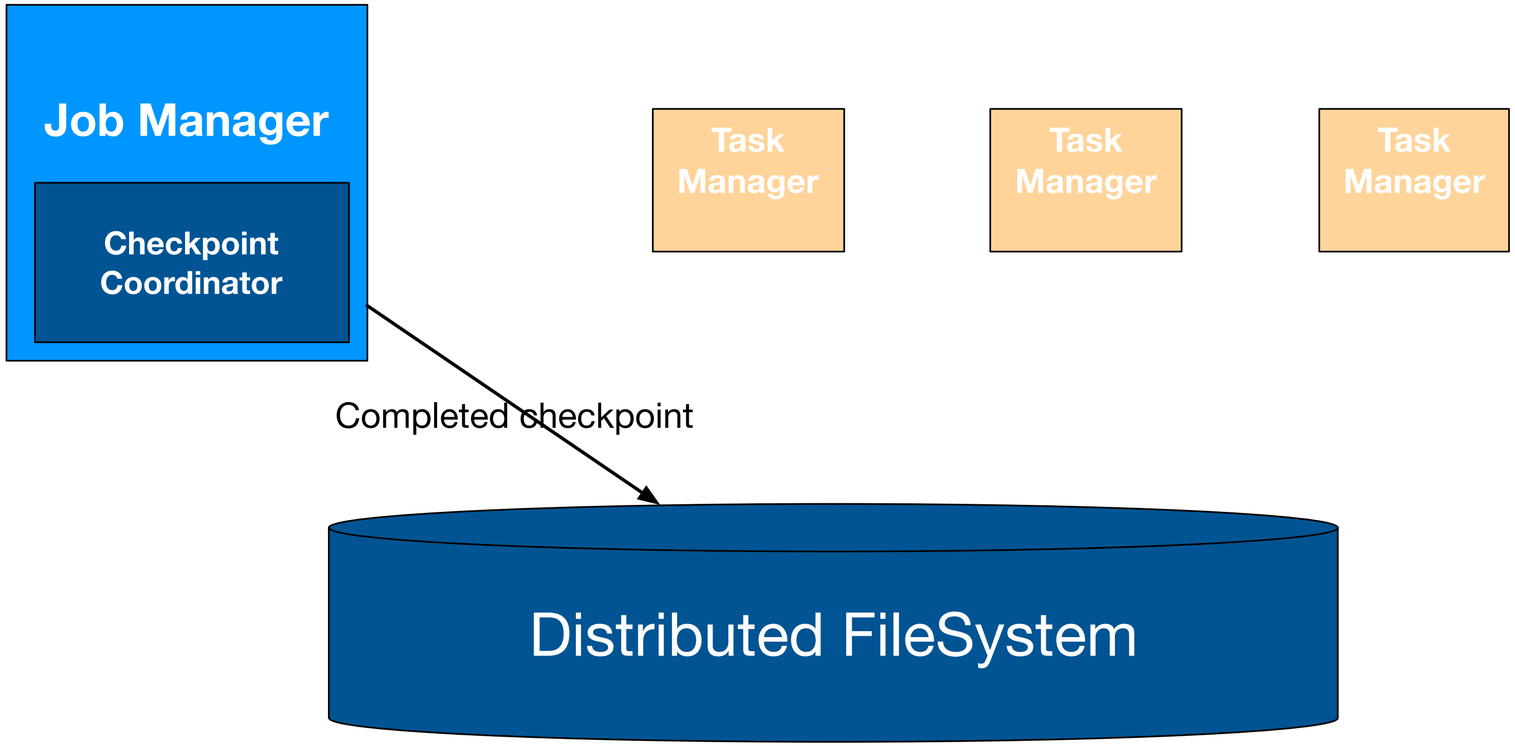


Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (within timeout duration)



阿里巴巴双十一历练下的存储之道

Zen of large-scale storage when Flink meets Alibaba global shopping festival

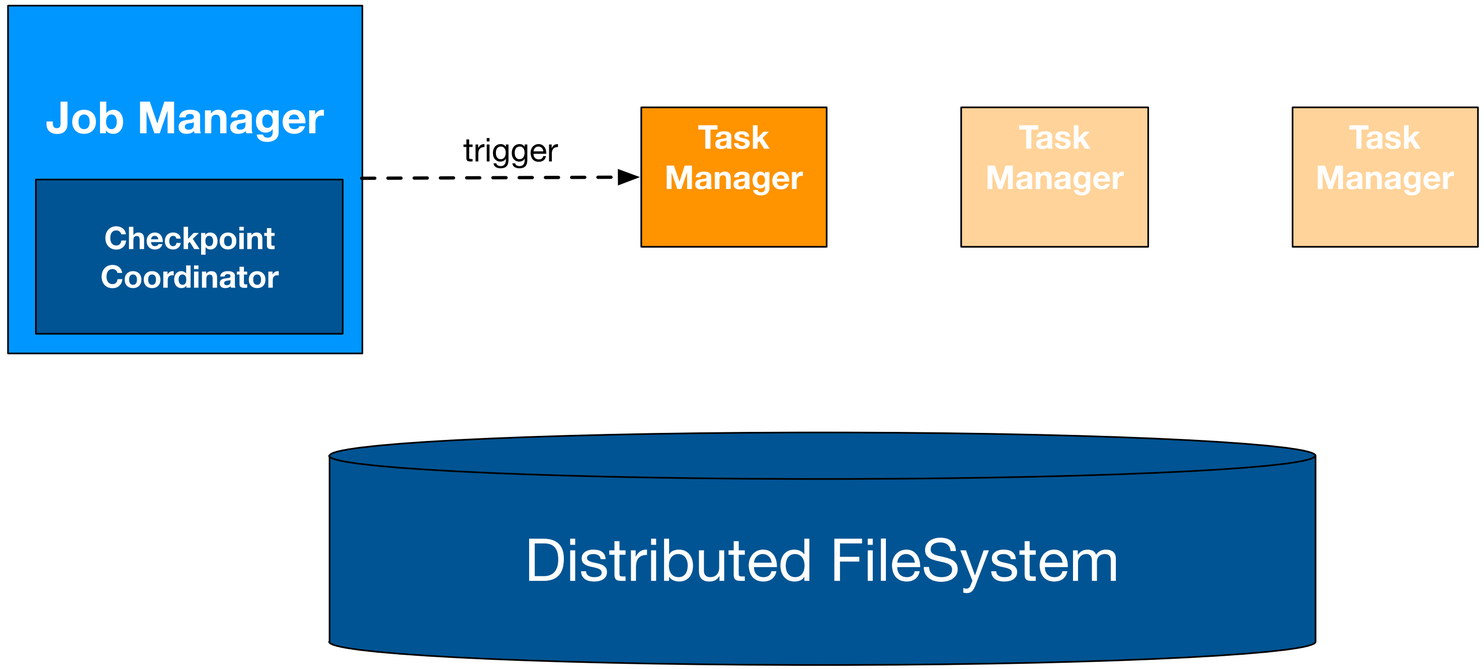
FLINK FORWARD

Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (time out)



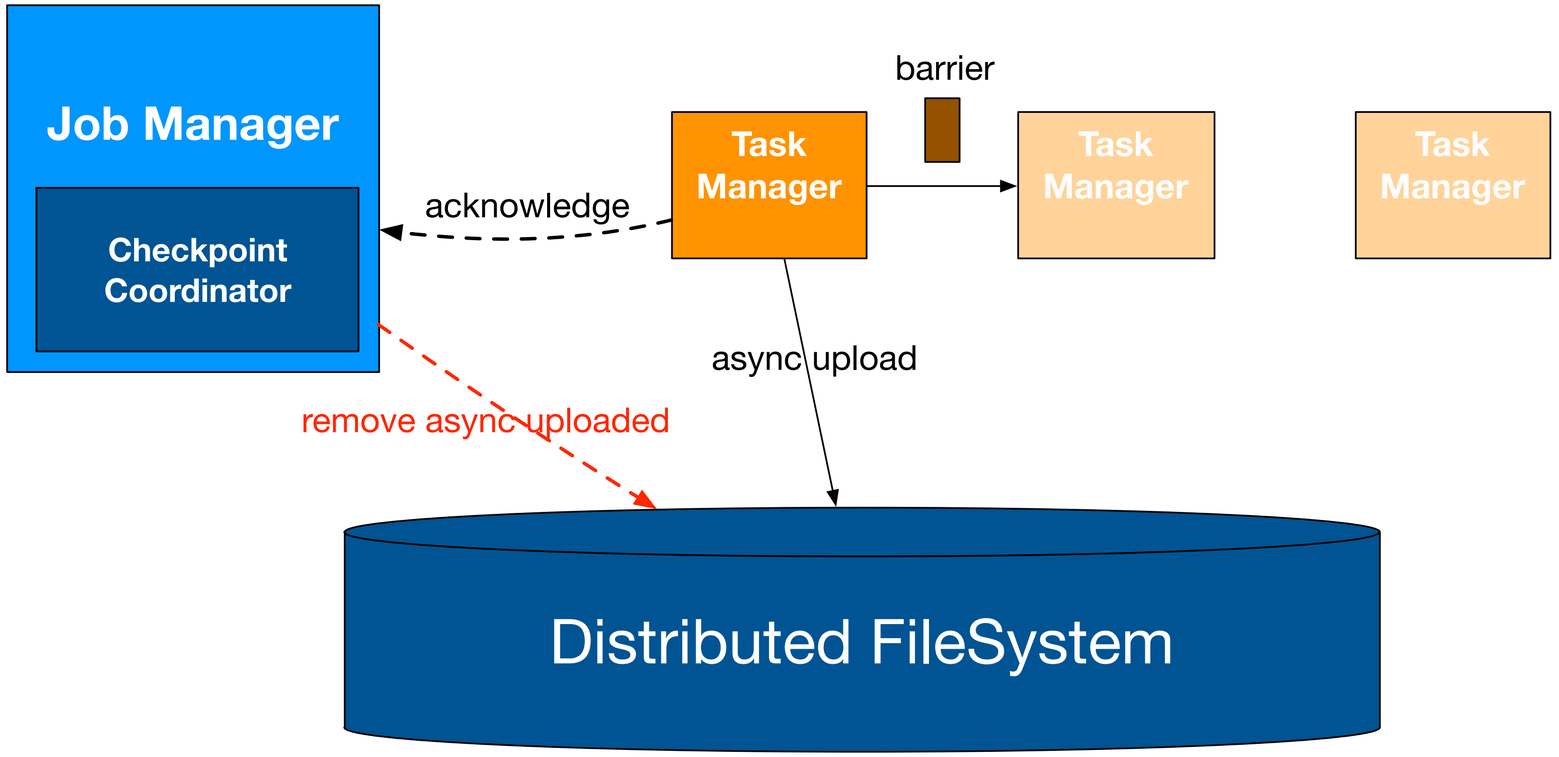
TIMEOUT, mark this checkpoint discarded

Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (time out)



阿里巴巴双十一历练下的存储之道

Zen of large-scale storage when Flink meets Alibaba global shopping festival

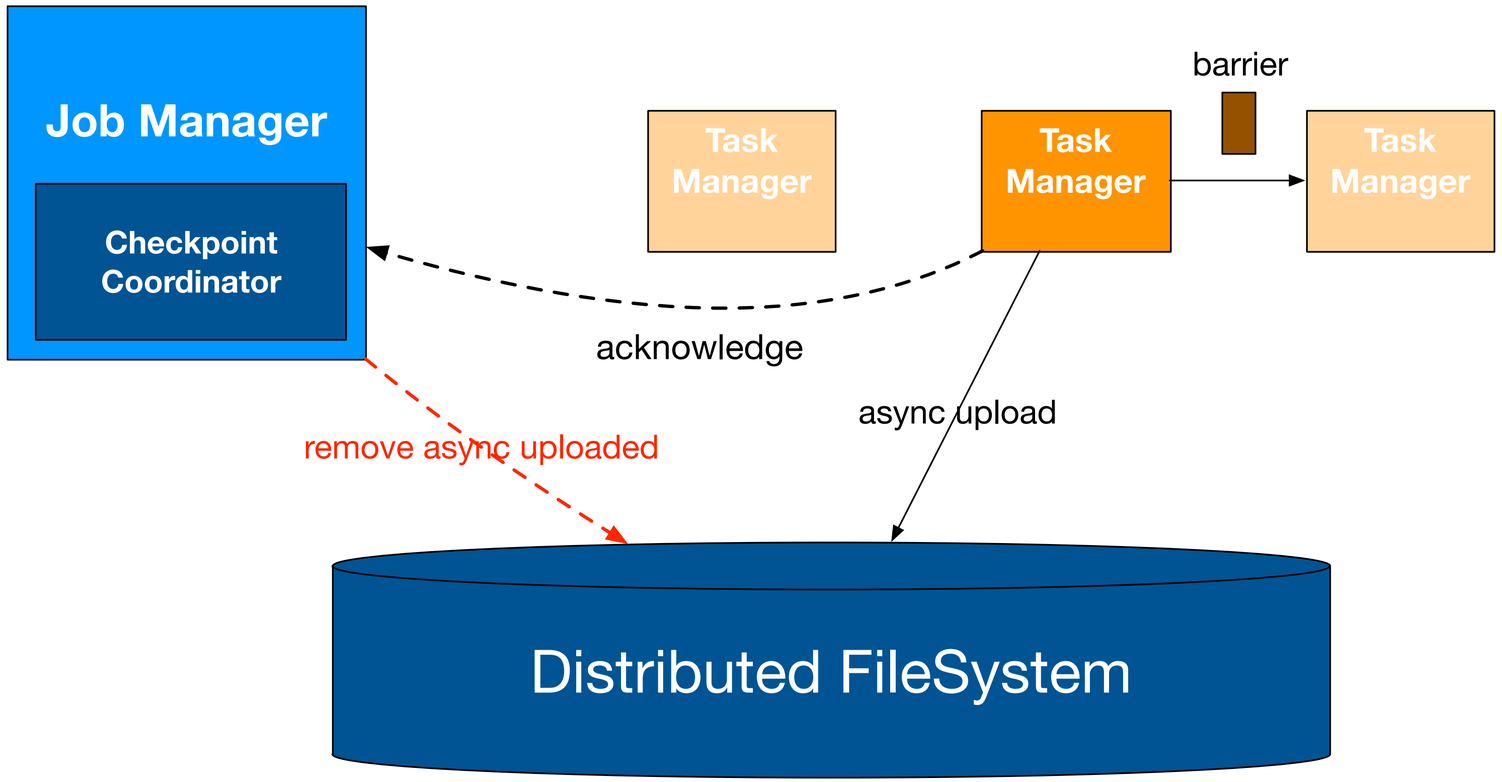
FLINK FORWARD

Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (time out)

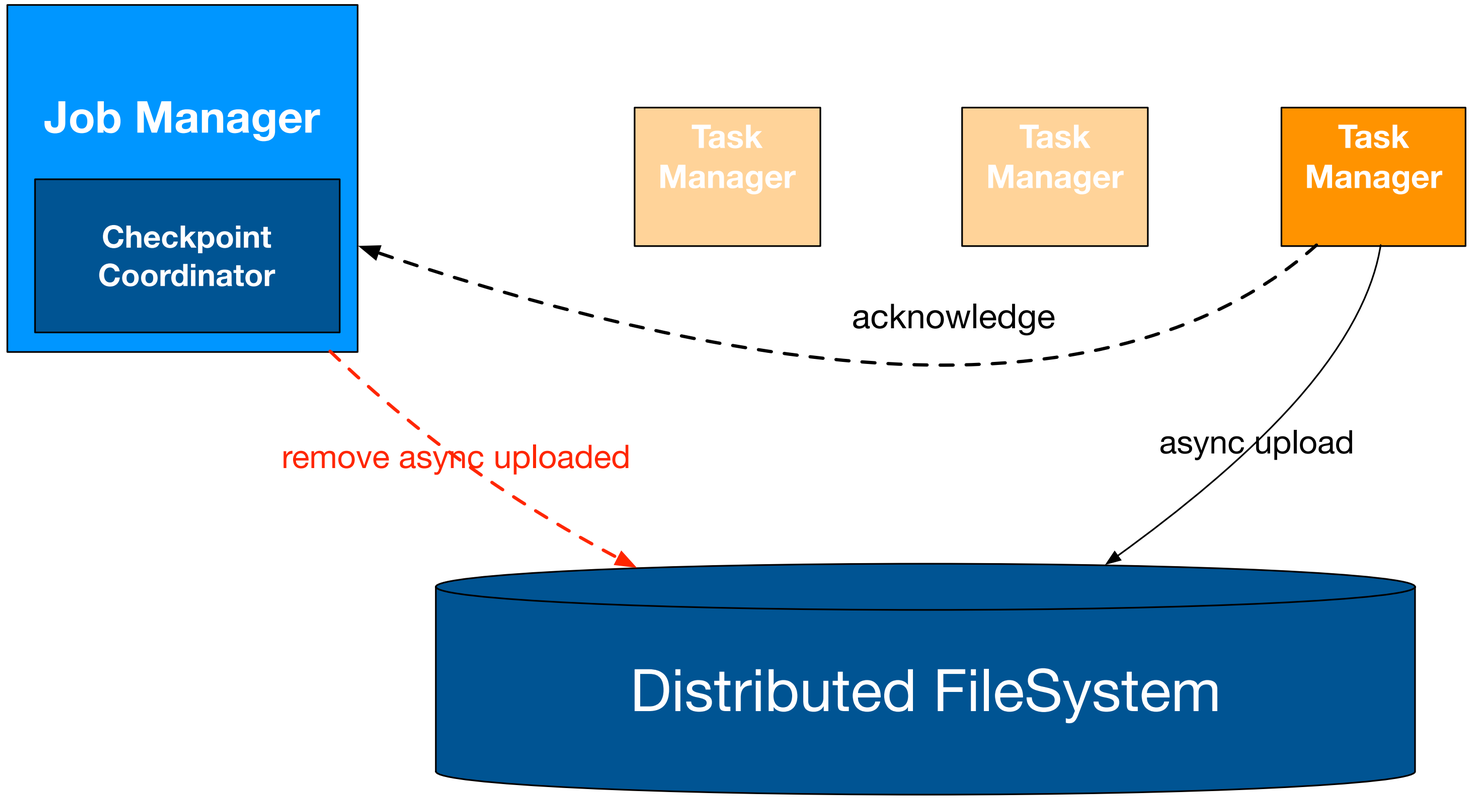


Problem-4.1

HDFS 因为激增的创建block请求而无法响应: 若干大作业因故堆积了近20小时的checkpoint 请求

HDFS has no response due to booming write block requests : several huge jobs accumulate checkpoint requests for nearly 20 hours.

Recap: Checkpoint end-to-end flow (time out)



Why not cancel checkpoint in time?

阿里巴巴双十一历练下的存储之道

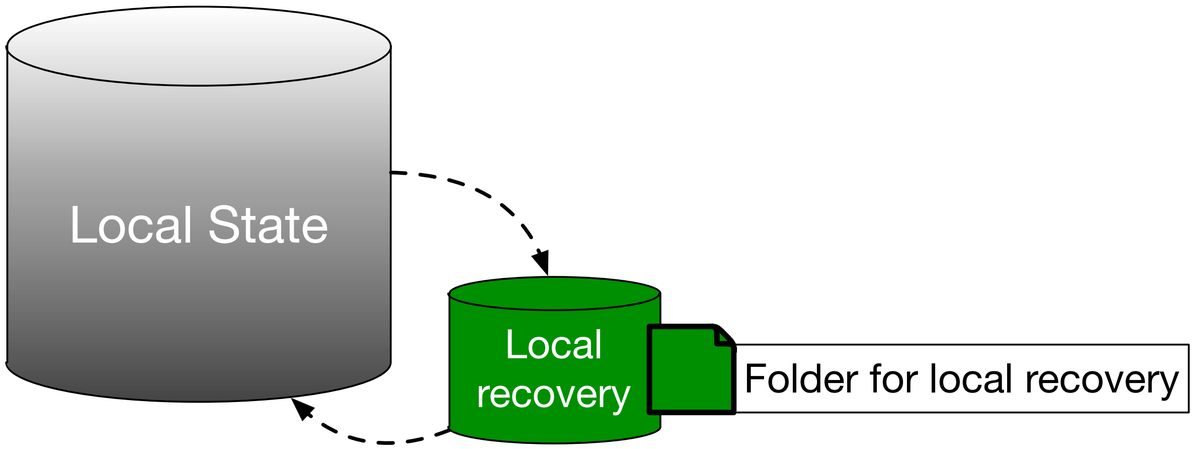
Zen of large-scale storage when Flink meets Alibaba global shopping festival

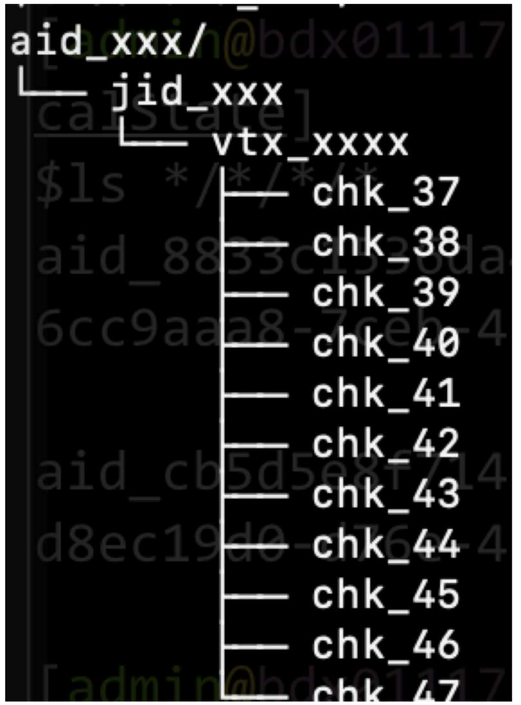
FLINK FORWARD

Problem-4.2

机器磁盘可用空间告急 ：local recovery目录积攒了一定数量的checkpoint目录

Machine run out of disk ： many huge local checkpoints existed in local recovery directory





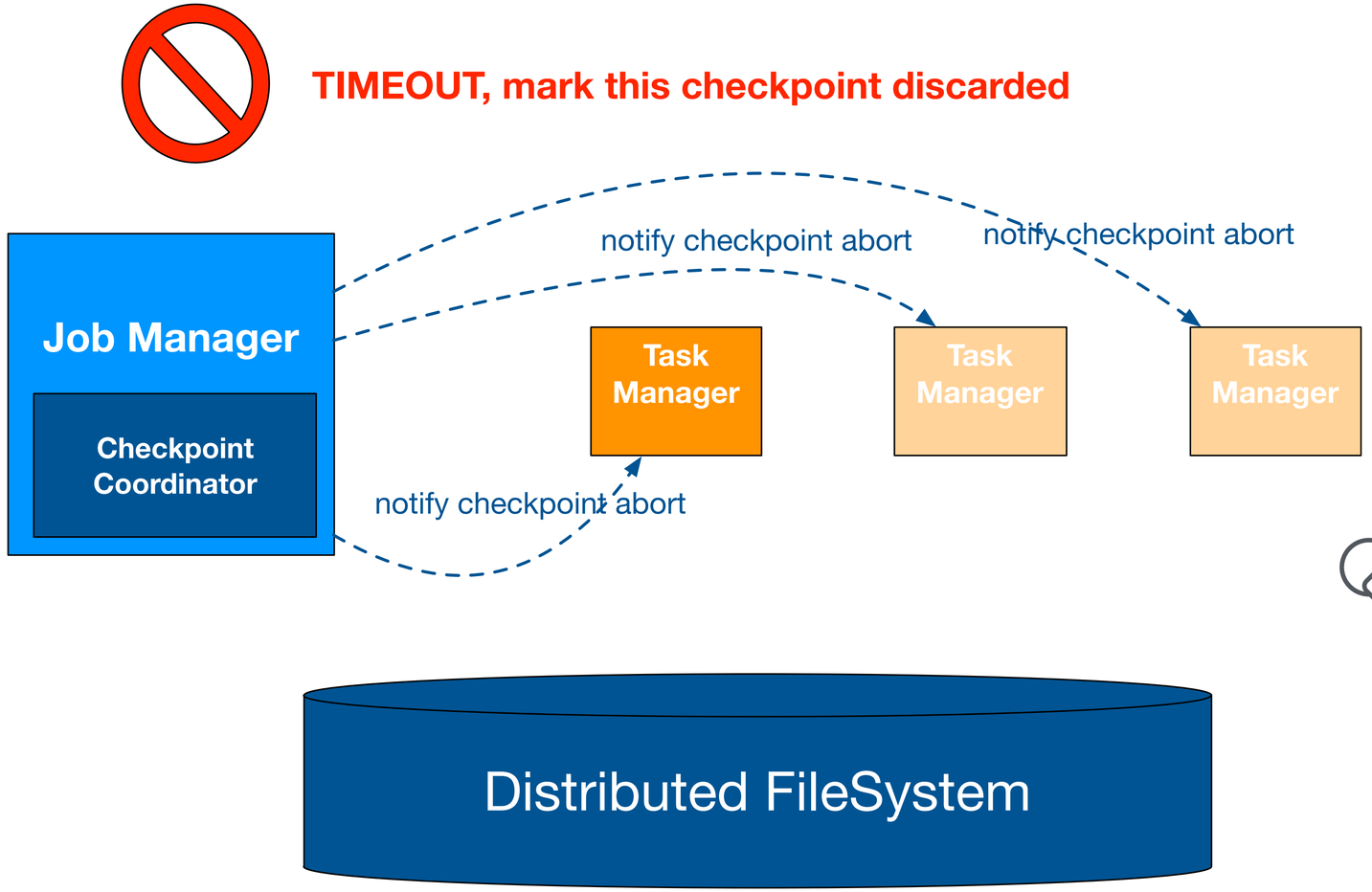
目前，只有当checkpoint完成时，才会触发Local recovery目录的清理

Currently, local recovery directory would only be pruned once a checkpoint completed

Problem-4.1 & 4.2

利用JM和TM之间的RPC通信及时告知不再需要该checkpoint

Use notification of checkpoint abort to tell task managers checkpoint is useless



Refer to in-review solution FLINK-8871, contributed from Alibaba

“清理门户”

Resource Release in Time

- Checkpoint 往往被认为是一种容错备份机制，但是其长时间的持续无法完成也有可能会是“致命”的。
  We might treat checkpoint as a backup or fault tolerance mechanism, however, it might also threaten the running job itself if cannot completed for a long time.

- 资源泄漏在大数据场景下会是更可怕的问题
  Resource leak would become more dangerous in big-data scenario.

阿里巴巴双十一历练下的存储之道

Zen of large-scale storage when Flink meets Alibaba global shopping festival

Problem-5

HDFS 目录写满 : 真的就是创建的文件数目太多写不下了

HDFS exceed max directory items : just too many sub-directories

```
org.apache.hadoop.hdfs.protocol.FSLimitException$MaxDirectoryItemsExceededException:
The directory item limit of xxx/shared is exceeded: limit=1048576 items=1048576
```

既然有这个限制，将目录分层，确保能够装下足够多的目录/文件

Introduce layer with sub-directories

Refer to in-review solution FLINK-11695, contributed from Alibaba

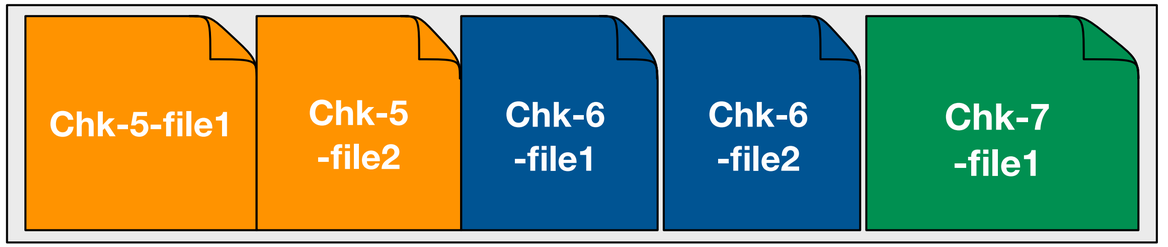
Problem-6

HDFS 整体压力大，创建文件数目多

Too many created files make HDFS under heavy pressure

HDFS是一个对小文件不那么友好的分布式文件系统，可是用户的数据量就是那么大，就是需要创建很多的文件。

HDFS is not so friendly to many small files creation. However, as job scale increases and checkpoint size grows, they would just make really heavy pressure on HDFS.



Make snapshotted files share across checkpoints.

Refer to in-review solution FLINK-11937, contributed from Alibaba

化大为小

Make big things smaller

- 有些问题不是bug或者使用不当，就是达到了大数据的系统瓶颈

  Some problems just would hit the bottleneck of big-data system

- 分布式处理的核心就是将大数据量的问题化为一个个小问题。

  Make big things smaller is the key to distributed environment.

目录
Contents

未来发展与思考

The left problems

FLINK FORWARD

Have we solved all problems regarding to large-scale storage in Flink?

- Who would clean the orphan files?
- Who should clean files job no longer existed?

FLINK FORWARD

THANKS