Flink State Best Practice

唐云 · 阿里巴巴 / 高级开发工程师

Apache Flink Community China

CONTENT

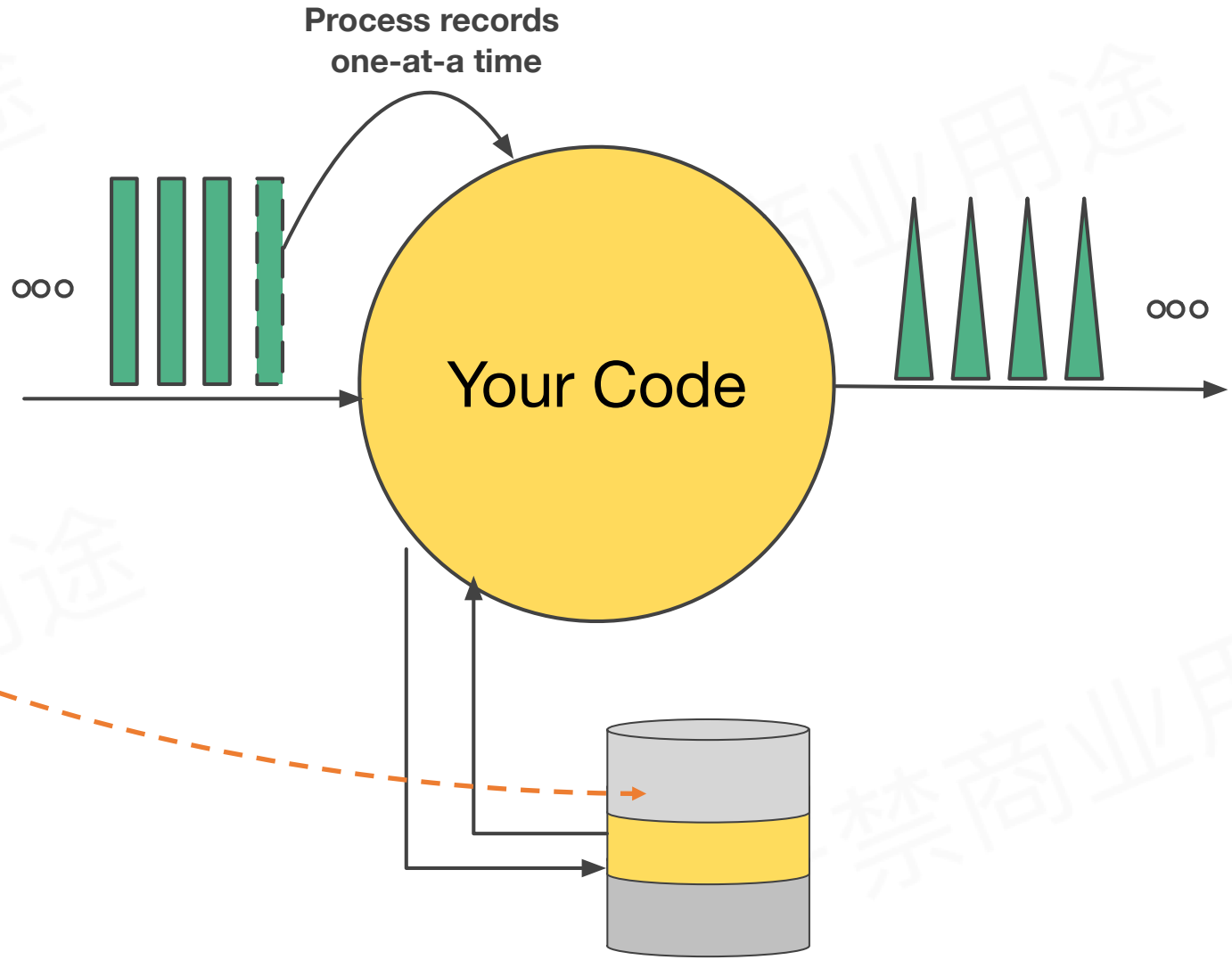
目录 >>

01

State Overview

State overview

State:流式计算中持久化了的状态



OperatorState V.S KeyedState

- OperatorState 没有current key 概念
- KeyedState 的数值总是与一个current key对应的

current key

- OperatorState 只有堆内存一种实现
- KeyedState 有堆内存和RocksDB两种实现

heap

OperatorState V.S KeyedState

snapshot

- OperatorState 需要手动实现snapshot和restore方法
- KeyedState由backend实现，对用户透明

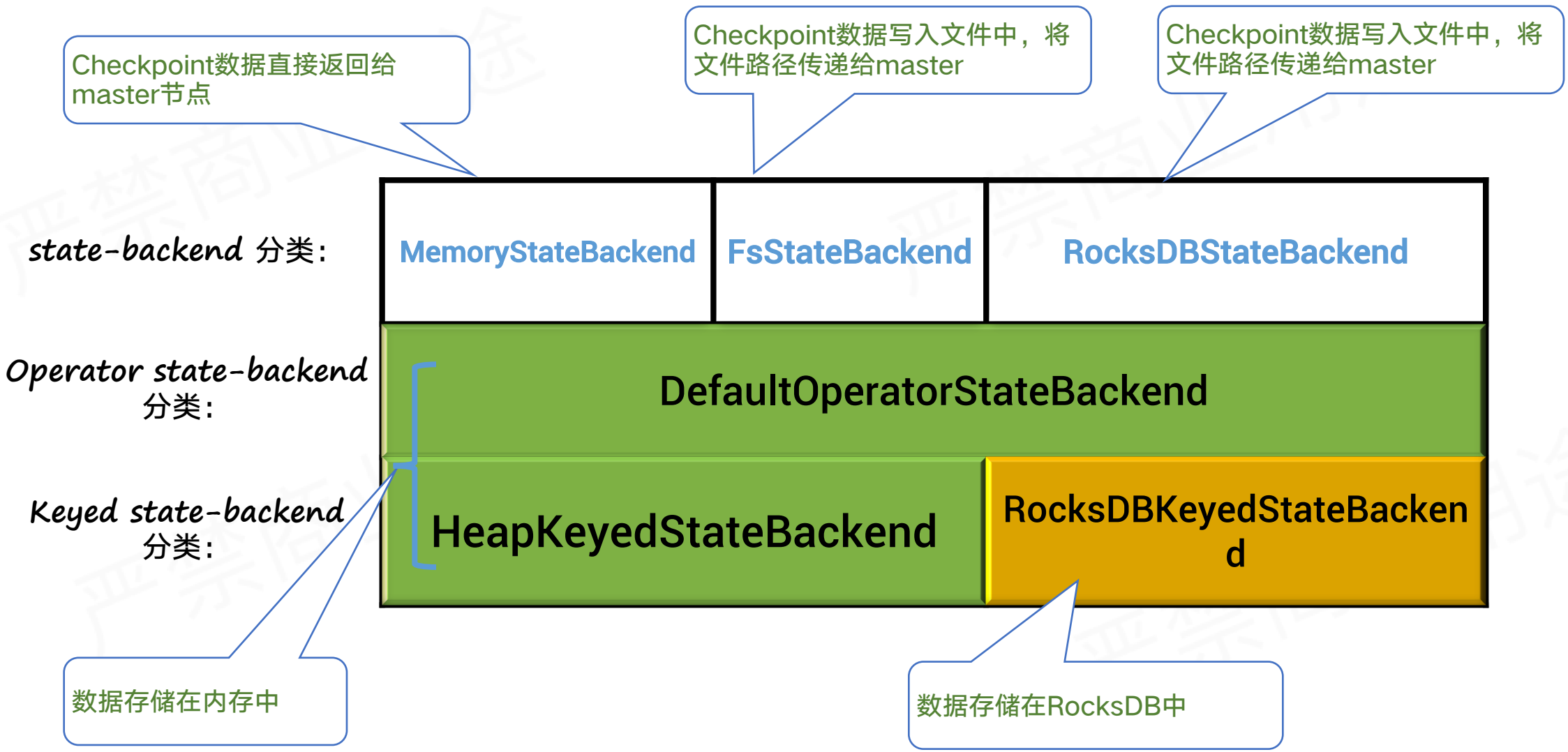
Size

- OperatorState 一般被认为是规模比较小的
- KeyedState 一般是相对规模较大的

02

Which state backend to use

State的存储



StateBackend 的选择

- **FsStateBackend**：性能更好；日常存储是在堆内存中，面临着OOM的风险，不支持增量checkpoint。

- **RocksDBStateBackend**：无需担心OOM风险

RocksDB的state存储

State1		State2	
KeyGroup + Key + Namespace	value	KeyGroup + Key + Namespace	value
(1, K1, Window(10, 20))	v1	(2, K2, Window(10, 20))	v2
(1, K3, Window(10, 20))	v3	(2, K4, Window(10, 25))	v4
(1, K1+MK1, VoidNameSpace)	v5	(2, K2+MK2, VoidNameSpace)	v6
…	…	…	…

RocksDB中，每个state使用一个Column Family

每个column family使用独占write buffer，整个DB共享一个block cache

RocksDB的相关参数

- Flink-1.8开始支持 ConfigurableOptionsFactory

state.backend.rocksdb.block.blocksize	数据块大小，默认‘4KB‘. 增大会影响减少内存使用，但是会影响读性能
state.backend.rocksdb.block.cache-size	整个DB的block cache大小，默认‘8MB‘. 建议调大
state.backend.rocksdb.compaction.level.use-dynamic-size	如果使用LEVEL compaction，在SATA磁盘上，建议配置成true，默认’false'.

RocksDB的相关参数

- Flink-1.8开始支持 ConfigurableOptionsFactory

state.backend.rocksdb.files.open	最大打开文件数目, ‘-1’ 意味着没有限制. 默认值 '5000'.
state.backend.rocksdb.thread.num	后台flush和compaction的线程数. 默认值 ‘1‘. 建议调大
state.backend.rocksdb.writebuffer.count	每个column family的writebuffer数目，默认值 ‘2‘. 建议调大
state.backend.rocksdb.writebuffer.number-to-merge	写之前的write buffer merge数目，默认值 ‘1‘. 建议调大
state.backend.rocksdb.writebuffer.size	每个write buffer的size，默认值‘4MB‘. 建议调大

如何配置RocksDB的options

```
public class MyOptionsFactory implements ConfigurableOptionsFactory {

    private static final long DEFAULT_SIZE = 256 * 1024 * 1024;  // 256 MB
    private long blockCacheSize = DEFAULT_SIZE;

    @Override
    public DBOptions createDBOptions(DBOptions currentOptions) {
        return currentOptions.setIncreaseParallelism(4)
                .setUseFsync(false);
    }

    @Override
    public ColumnFamilyOptions createColumnOptions(ColumnFamilyOptions currentOptions) {
        return currentOptions.setTableFormatConfig(
            new BlockBasedTableConfig()
                .setBlockCacheSize(blockCacheSize)
                .setBlockSize(128 * 1024));           // 128 KB
    }

    @Override
    public OptionsFactory configure(Configuration configuration) {
        this.blockCacheSize =
            configuration.getLong("my.custom.rocksdb.block.cache.size", DEFAULT_SIZE);
        return this;
    }
}
```

03

State best practice

一些使用state的心得

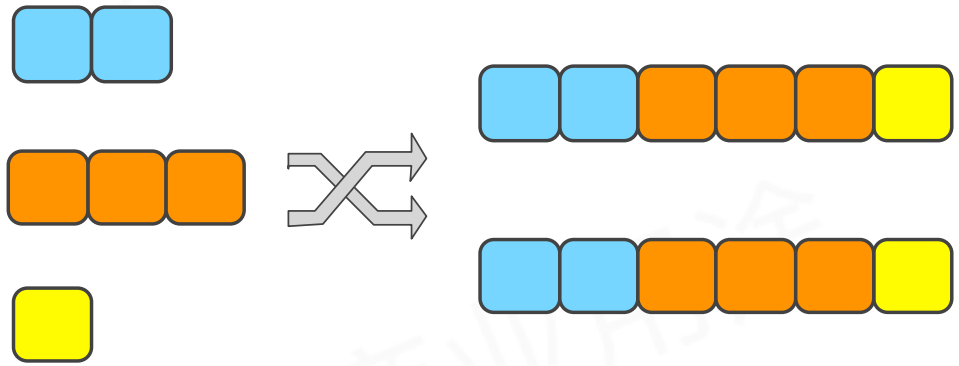
OperatorState 使用建议

慎重使用长list

```
/**
 * Meta information about the operator state handle.
 */
class StateMetaInfo implements Serializable {

    private static final long serialVersionUID = 35938176158589411 66L;

    private final long[] offsets;
    private final Mode distributionMode;

    public StateMetaInfo(long[] offsets, Mode distributionMode) {
        this.offsets = Preconditions.checkNotNull(offsets);
        this.distributionMode = Preconditions.checkNotNull(distributionMode);
    }
```

OperatorState 使用建议

正确使用UnionListState



- restore后，每个subTask均恢复了与之前所有并发的state。
- 目前Flink内部的使用都是为了获取之前的全局信息，在下一次snapshot时，仅使用其中一部分做snapshot。
- 切勿在下一次snapshot时进行全局snapshot

KeyedState 使用建议

如何清空当前state?

- state.clear() 只能清理当前key对应的value值
- 需要借助KeyedStateBackend的 applyToAllKeys 方法

```
// clear state via applyToAllKeys().
backend.applyToAllKeys(VoidNamespace.INSTANCE, VoidNamespaceSerializer.INSTANCE, listStateDescriptor,
    new KeyedStateFunction<Integer, ListState<String>>() {
        @Override
        public void process(Integer key, ListState<String> state) throws Exception {
            state.clear();
        }
    });
```

KeyedState 使用建议

考虑value值很大的极限场景（RocksDB）

- 受限于JNI bridge API的限制，单个value只支持2^31 bytes
- 考虑使用MapState来替代ListState或者ValueState

KeyedState 使用建议

如何知道当前RocksDB的使用情况

- RocksDB的日志可以观察到一些compaction信息，默认存储位置在 flink-io 目录下，需要登录到taskmanager里面才能找到。
- 考虑打开RocksDB的native metrics

https://ci.apache.org/projects/flink/flink-docs-master/ops/config.html#rocksdb-native-metrics

KeyedState 使用建议

配置了state TTL，可能你的存储空间并没有减少

- 默认情况下，只有在下次读访问时才会触发清理那条过期数据
  如果那条数据之后不再访问，则也不会被清理。

```
StateTtlConfig ttlConfig = StateTtlConfig
    .newBuilder(Time.days(7))
    .cleanupFullSnapshot()
    .build();
```

Full snapshot 时候会清理snapshot的内容

```
StateTtlConfig ttlConfig = StateTtlConfig
    .newBuilder(Time.days(7))
    // check 10 keys for every state access
    .cleanupIncrementally(10, false)
    .build();
```

Heap state backend的持续清理

```
StateTtlConfig ttlConfig = StateTtlConfig
    .newBuilder(Time.days(7))
    .cleanupInRocksdbCompactFilter()
    .build();
```

RocksDB state backend的持续清理

RawState (timer) 使用建议

Timer state 太大怎么办

- 考虑存储到RocksDB中

```
state.backend.rocksdb.timer-service.factory: ROCKSDB
```

- Trade off,
  - 存储到Heap中，面临OOM风险，Checkpoint的同步阶段耗时大
  - 存储到RocksDB中，影响timer的读写性能

RocksDBState 使用建议

不要创建过多的state

- 每个state一个column family，独占write buffer，过多的state会导致占据过多的write buffer
- 根本上还是RocksDB StateBackend的 native 内存无法直接管理

```
block cache + writebuffer-num * writebuffer-size + index&filter
```

04

Some tips of checkpoint

一些 checkpoint 的使用建议

Checkpoint的 使用建议

Checkpoint间隔不要太短

- 一般5min级别足够
- Checkpoint与record处理共抢一把锁，Checkpoint的同步阶段会影响record的处理

Checkpoint的 使用建议

合理设置超时时间

- 默认的超时时间是10min，如果state规模大，则需要合理配置。
  最坏情况是创建速度大于删除速度，导致磁盘空间不可用。

Checkpoint的 使用建议

FsStateBackend可以考虑文件压缩

- 对于刷出去的文件可以考虑使用压缩来减少Checkpoint体积

```
ExecutionConfig executionConfig = new ExecutionConfig();
executionConfig.setUseSnapshotCompression(true);
```

THANKS

Apache Flink 中文学习网站：ververica.cn