SPARK SHUFFLE INTRODUCTION

蘑菇街
mogujie.com

天火@蘑菇街

About Me

Spark / Hadoop / Hbase / Phoenix contributor

For spark mainly contributes in:

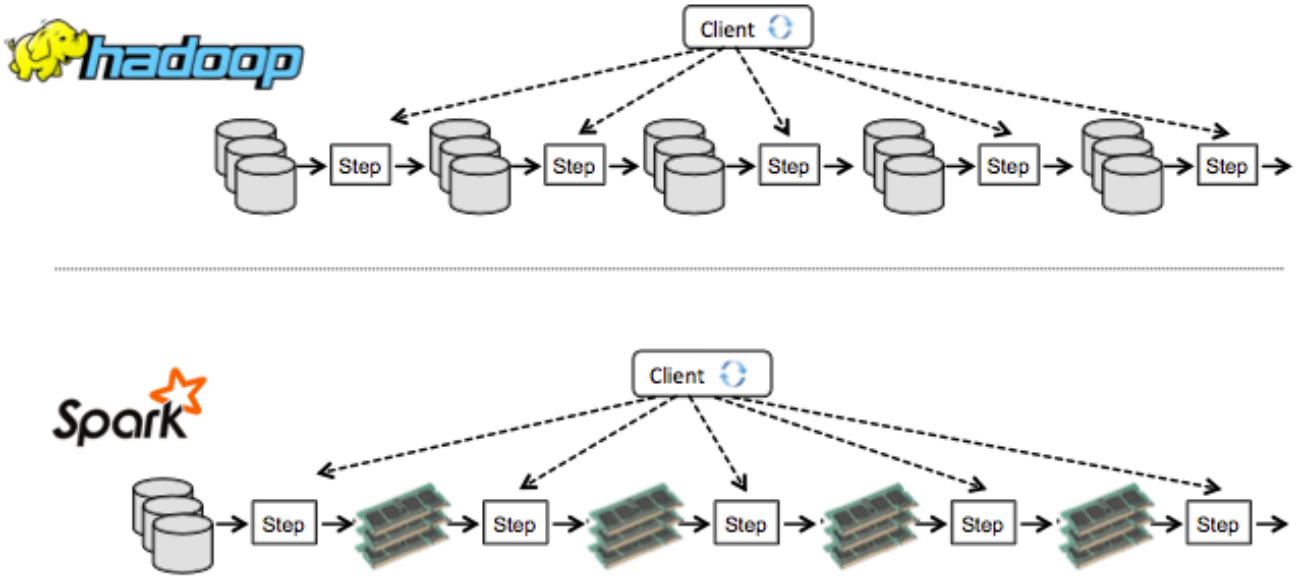
- Yarn
- Shuffle
- BlockManager
- Scala2.10 update
- Standalone HA
- Various other fixes.

tianhuo@mogujie.com
Weibo @冷冻蚂蚁
blog.csdn.net/colorant

Why Spark is fast(er)

- Whom do we compare to?

- What do we mean by fast?
  - fast to write
  - fast to run



Why Spark is fast(er) cont.

- But the figure in previous page is some how misleading.

- The key is the flexible programming mode.
  - Which lead to more reasonable data flow.
  - Which lead to less IO operation.
    - Especially for iterative heavy workloads like ML.
  - Which potentially cut off a lot of shuffle operations needed.

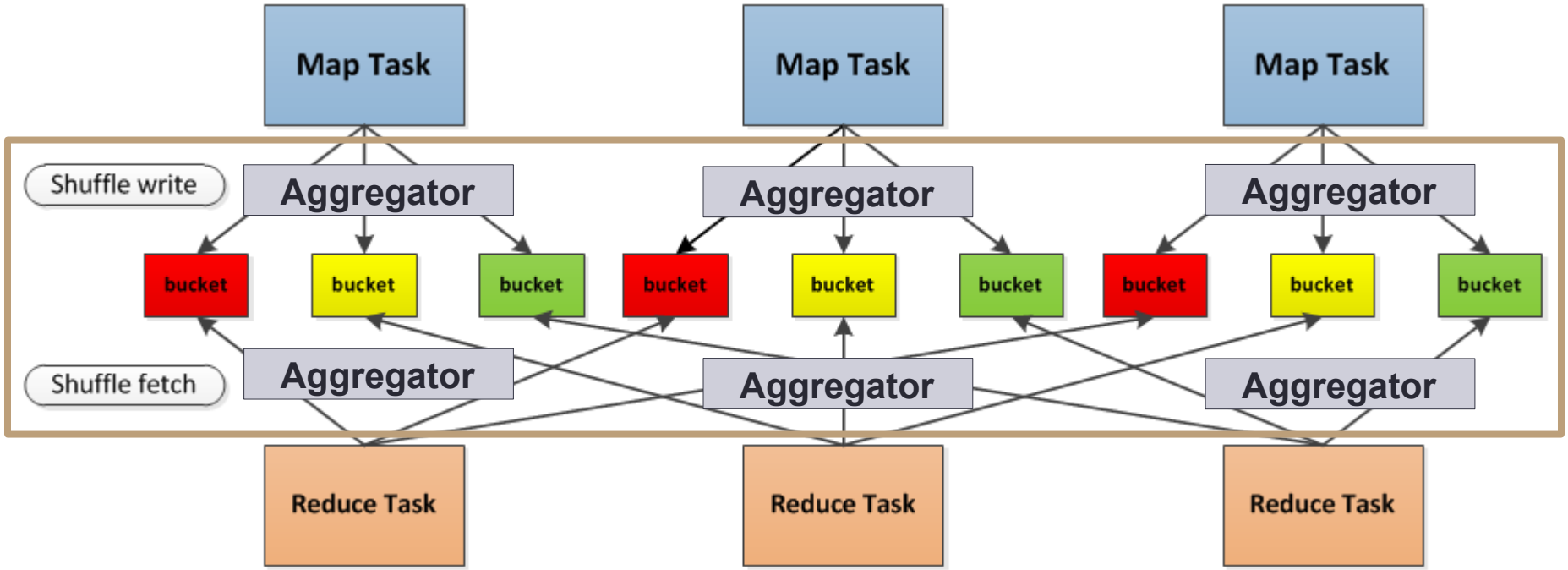
- But, you won't always be lucky.
  - Many app logic did need to exchange a lot of data.
  - In the end, you will still need to deal with shuffle
    - And which usually impact performance a lot.

APPROACHING MATTERS!

Photo By Raymond

What is shuffle

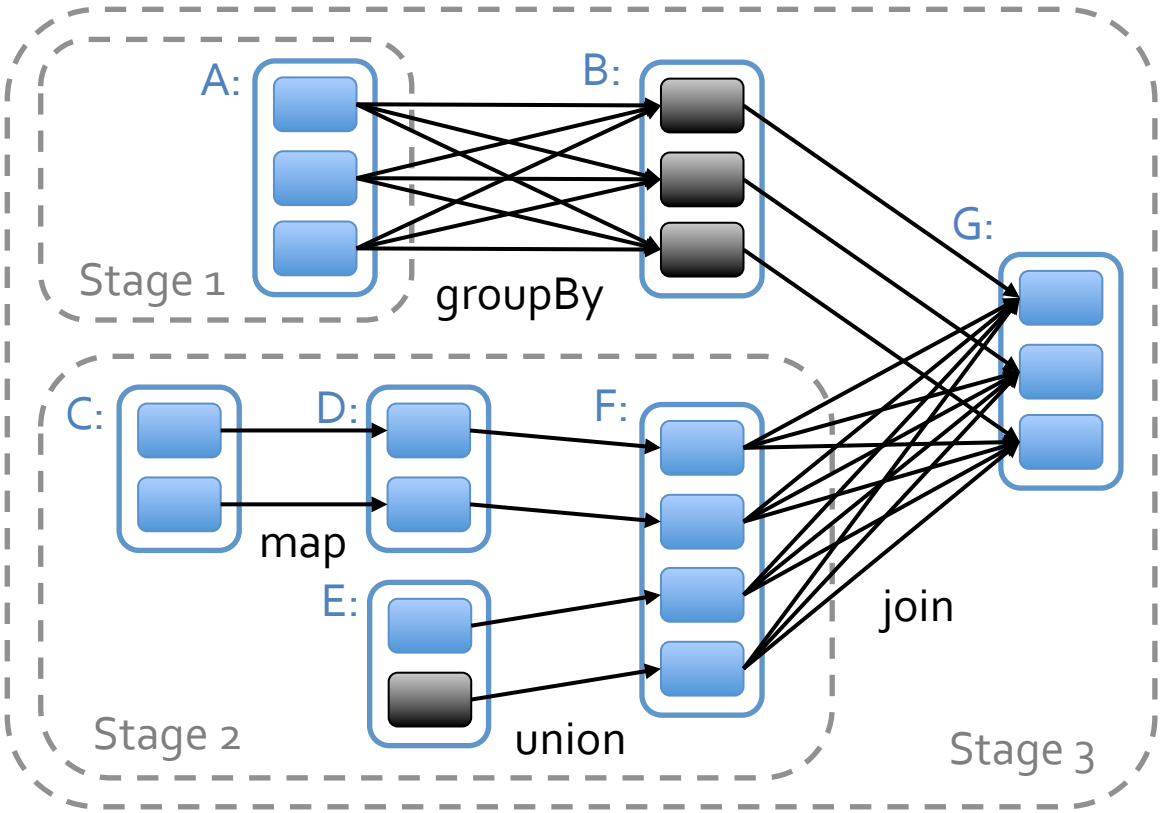
Shuffle overview



How does shuffle come into the picture

- Spark run job stage by stage.
- Stages are build up by DAGScheduler according to RDD's ShuffleDependency
  - e.g. ShuffleRDD / CoGroupedRDD will have a ShuffleDependency
  - Many operator will create ShuffleRDD / CoGroupedRDD under the hook.
    - Repartition/CombineByKey/GroupBy/ReduceByKey/cogroup
    - many other operator will further call into the above operators
      - e.g. various join operator will call cogroup.
- Each ShuffleDependency maps to one stage in Spark Job and then will lead to a shuffle.

So everyone should have seen this before



why shuffle is expensive

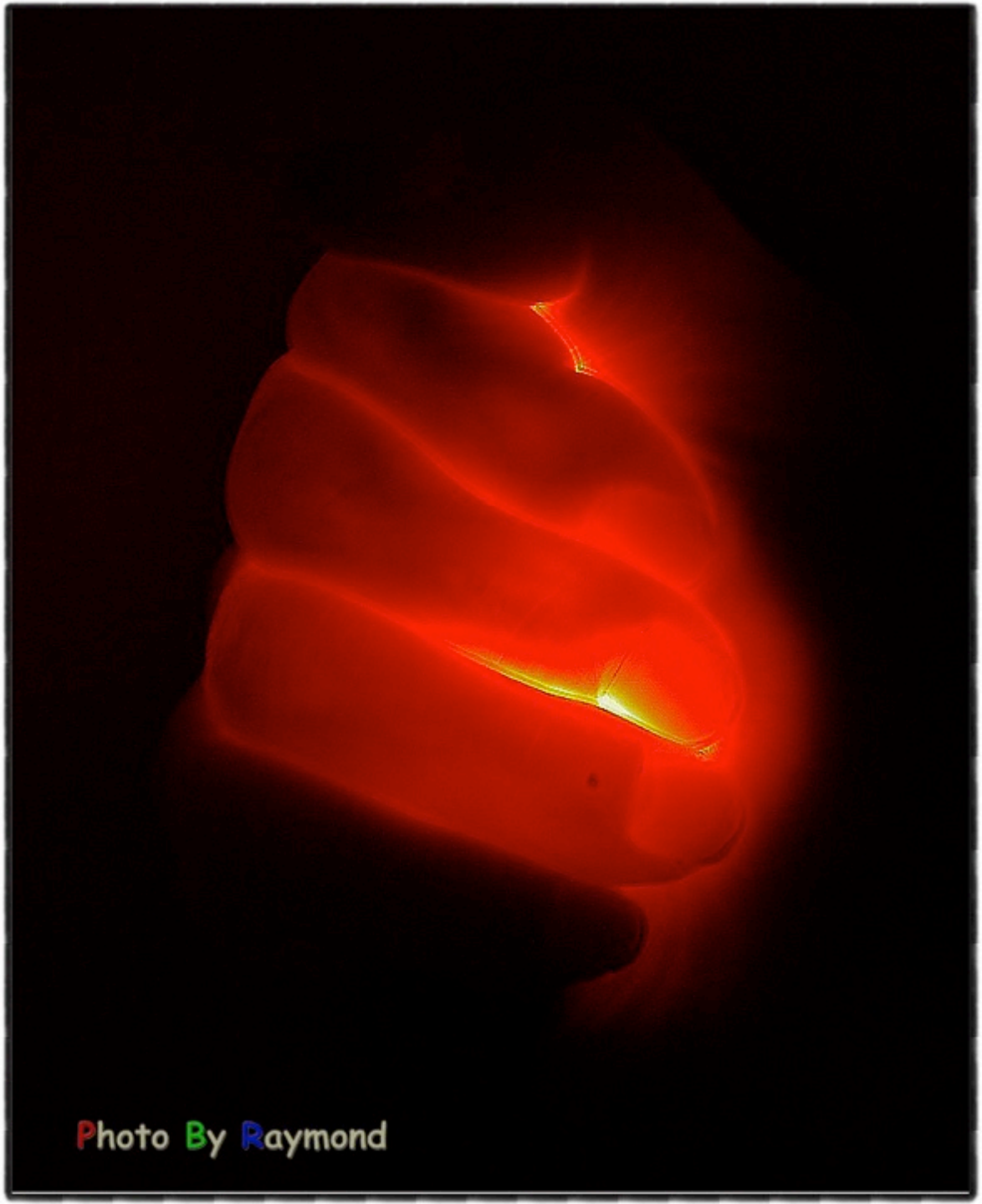
- When doing shuffle, data no longer stay in memory only

- For spark, shuffle process might involve
  - data partition: which might involve very expensive data sorting works etc.
  - data ser/deser: to enable data been transfer through network or across processes.
  - data compression: to reduce IO bandwidth etc.
  - DISK IO: probably multiple times on one single data block
    - E.g. Shuffle Spill, Merge combine

History

- Spark 0.6-0.7, same code path with RDD's persistent method, can choose MEMORY_ONLY and DISK_ONLY (default).
- Spark 0.8-0.9:
  - separate shuffle code path from BM and create ShuffleBlockManager and BlockObjectWriter only for shuffle, now shuffle data can only be written to disk.
  - Shuffle optimization: Consolidate shuffle write.
- Spark 1.0, pluggable shuffle framework.
- Spark 1.1, sort-based shuffle implementation.
- Spark 1.2 netty transfer service reimplementation. sort-based shuffle by default
- Spark 1.2+ on the go: external shuffle service etc.

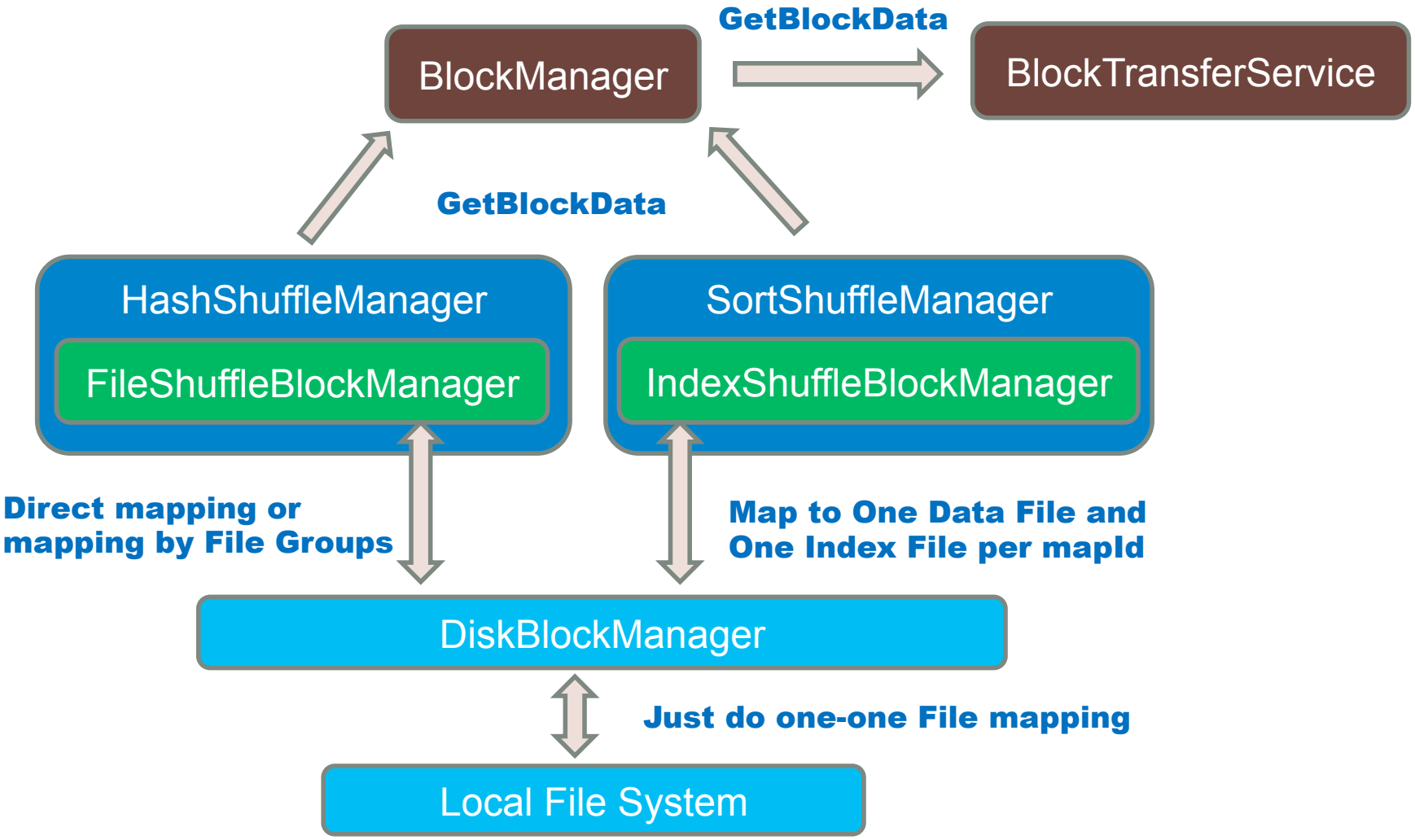
LOOK
INSIDE



Pluggable Shuffle Framework

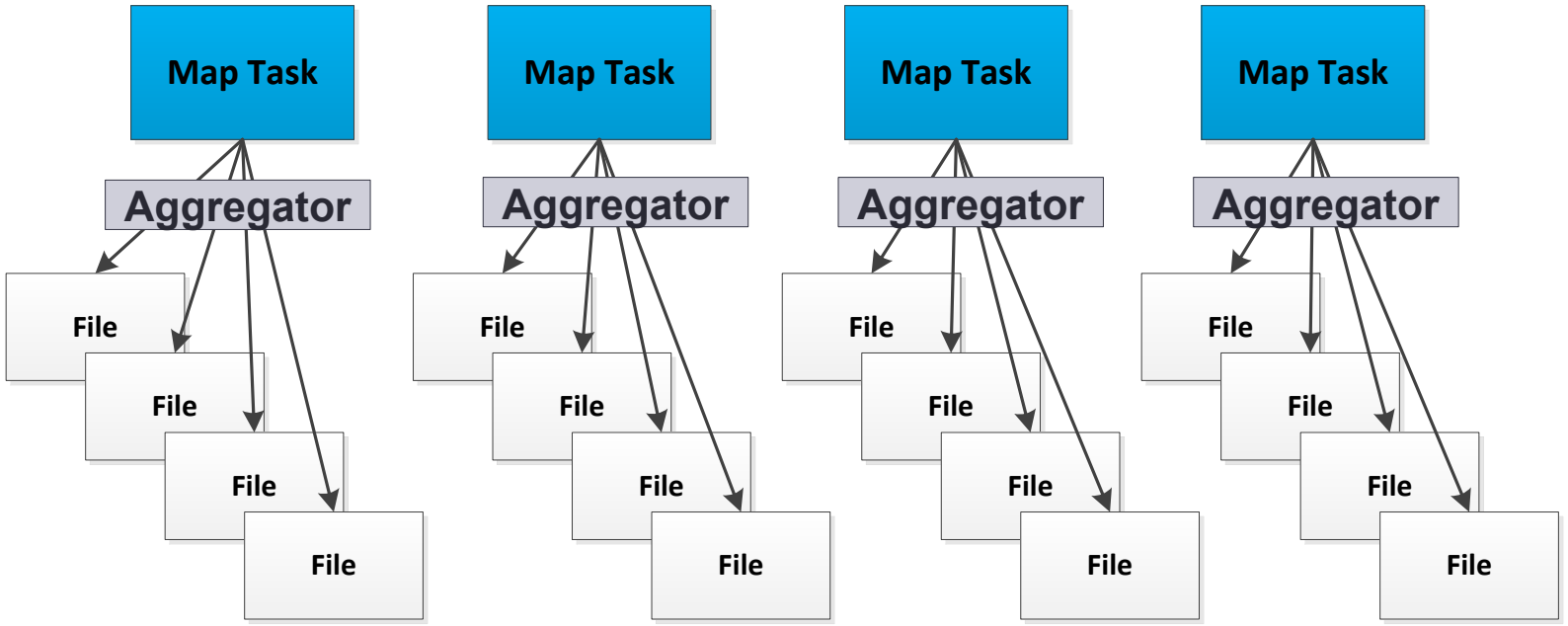
- ShuffleManager
  - Manage shuffle related components, registered in SparkEnv, configured through SparkConf, default is sort (pre 1.2 is hash),
- ShuffleWriter
  - Handle shuffle data output logics. Will return MapStatus to be tracked by MapOutputTracker.
- ShuffleReader
  - Fetch shuffle data to be used by e.g. ShuffleRDD
- ShuffleBlockManager
  - Manage the mapping relation between abstract bucket and materialized data block.

High level data flow



Hash Based Shuffle - Shuffle Writer

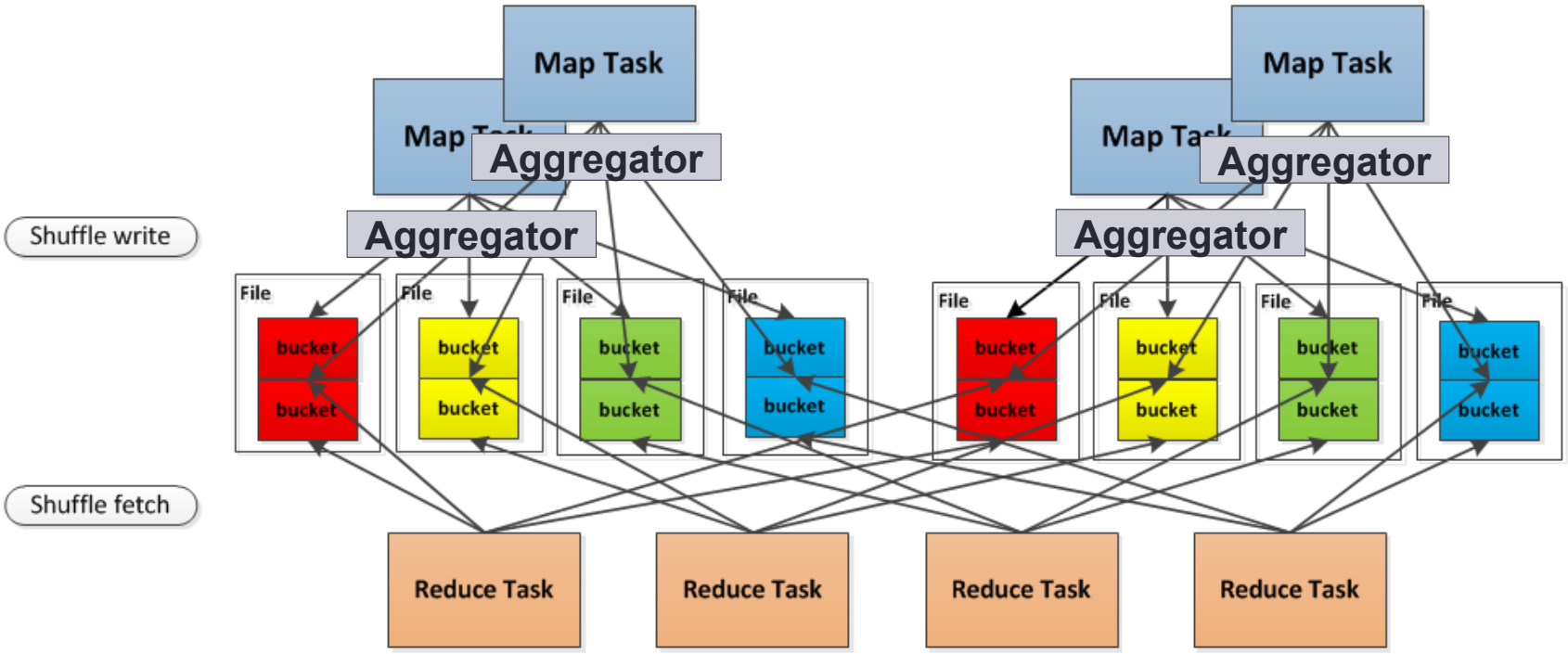
- Basic shuffle writer



Each bucket is mapping to a single file

Hash Based Shuffle - Shuffle Writer

- Consolidate Shuffle Writer



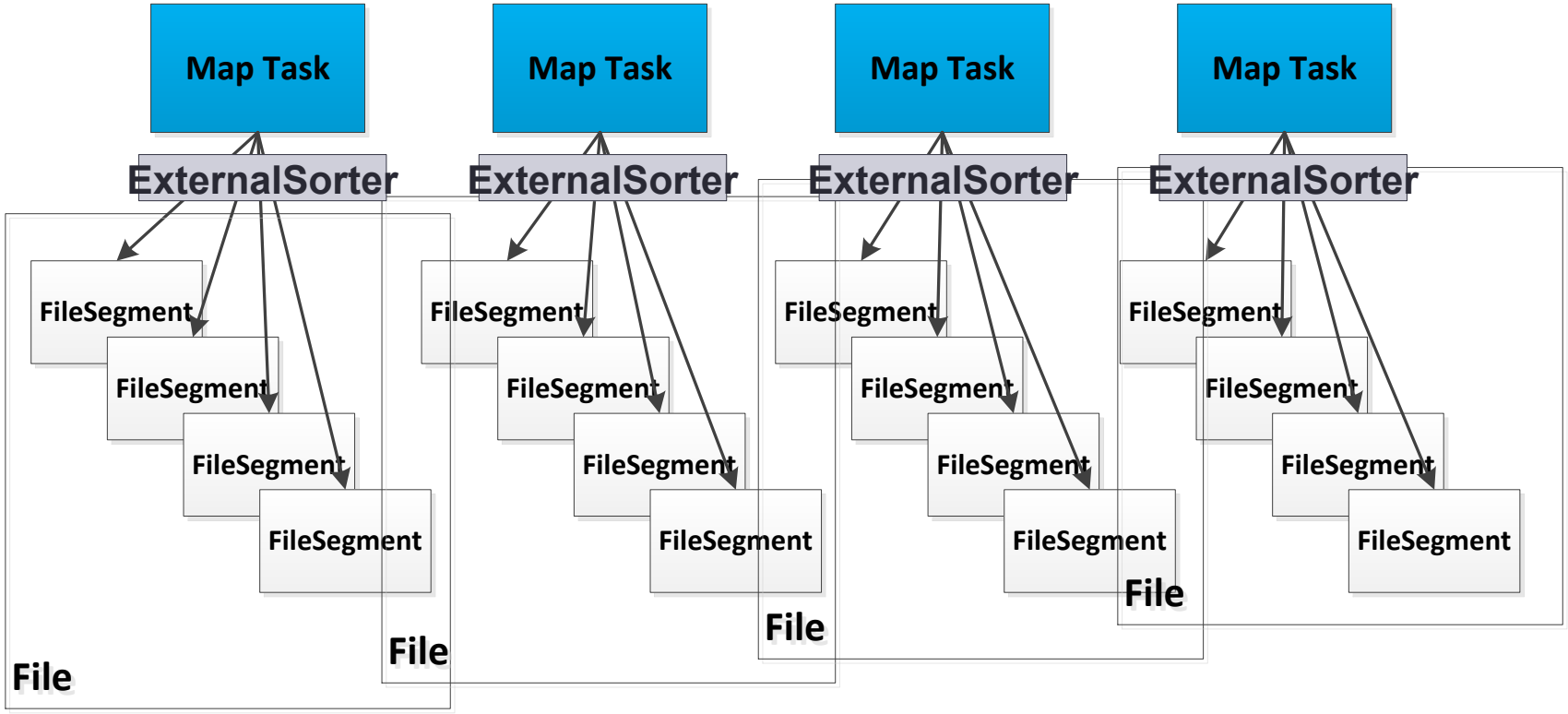
Each bucket is mapping to a segment of file

Hash Based Shuffle - Shuffle Writer

- Basic Shuffle Writer
  - *M* * *R* shuffle spill files
  - Concurrent *C* * *R* opened shuffle files.
  - If shuffle spill enabled, could generate more tmp spill files say N.
- Consolidate Shuffle Writer
  - Reduce the total spilled files into C * R if (M >> C)
  - Concurrent opened is the same as the basic shuffle writer.
- Memory consumption
  - Thus Concurrent *C* * *R* + *N* file handlers.
  - Each file handler could take up to 32~100KB+ Memory for various buffers across the writer stream chain.

Sort Based Shuffle - Shuffle Writer

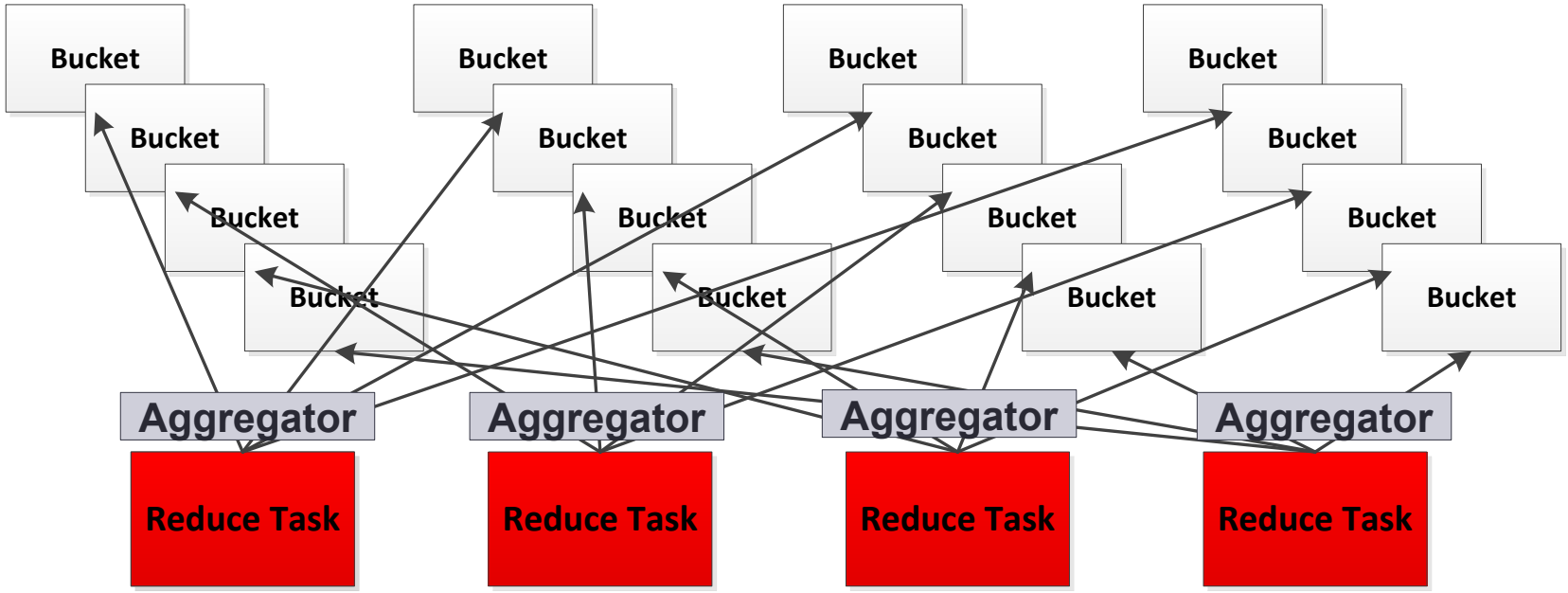
- Sort Shuffle Writer



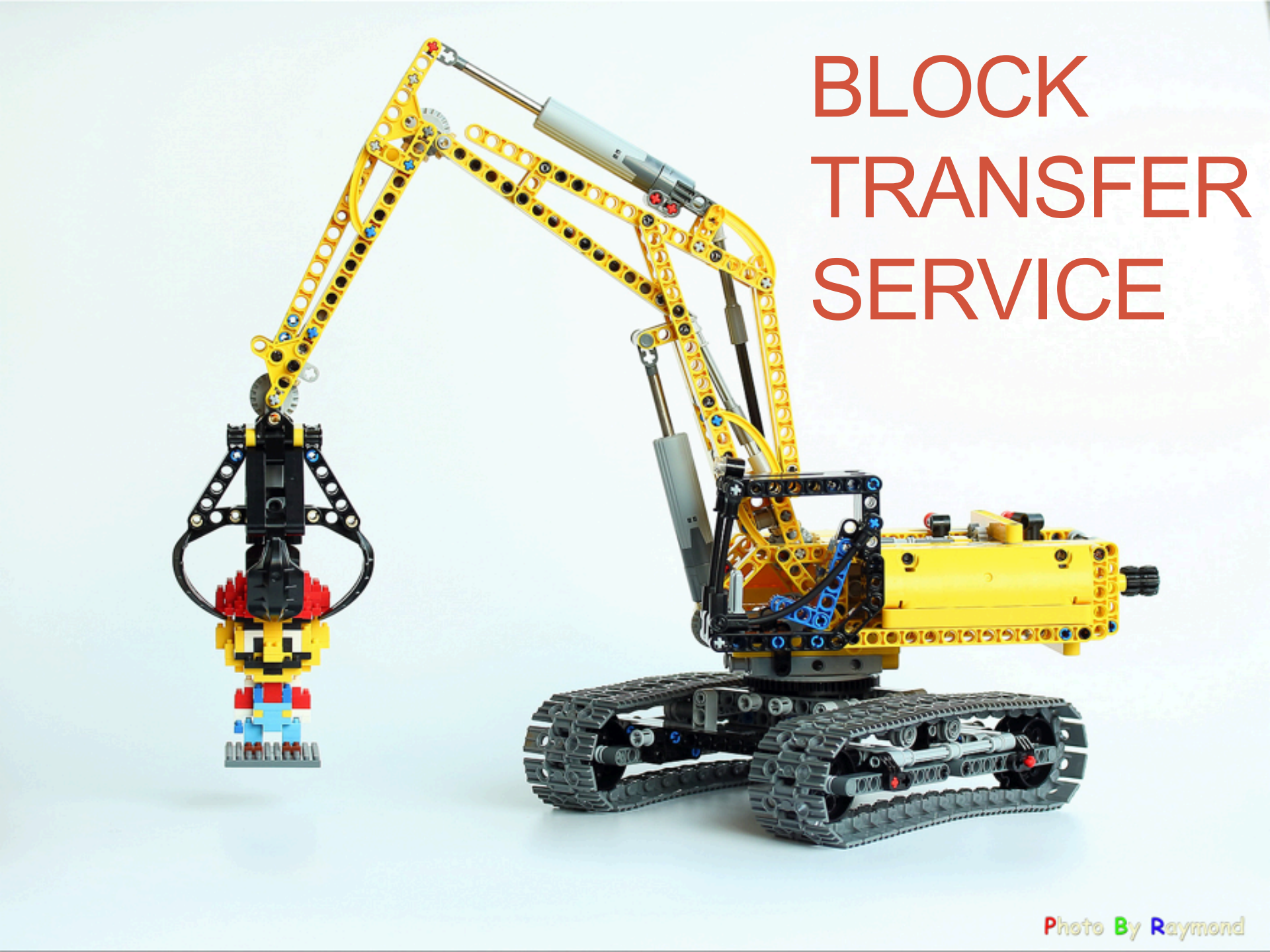
Sort Based Shuffle - Shuffle Writer

- Each map task generates 1 shuffle data file + 1 index file
  - Utilize ExternalSorter to do the sort works.
- If map-side combine is required, data will be sorted by key and partition for aggregation. Otherwise data will only be sorted by partition.
- If reducer number <= 200 and no need to do aggregation or ordering, data will not be sorted at all.
  - Will go with hash way and spill to separate files for each reduce partition, then merge them into one per map for final output.

Hash Based Shuffle - Shuffle Reader



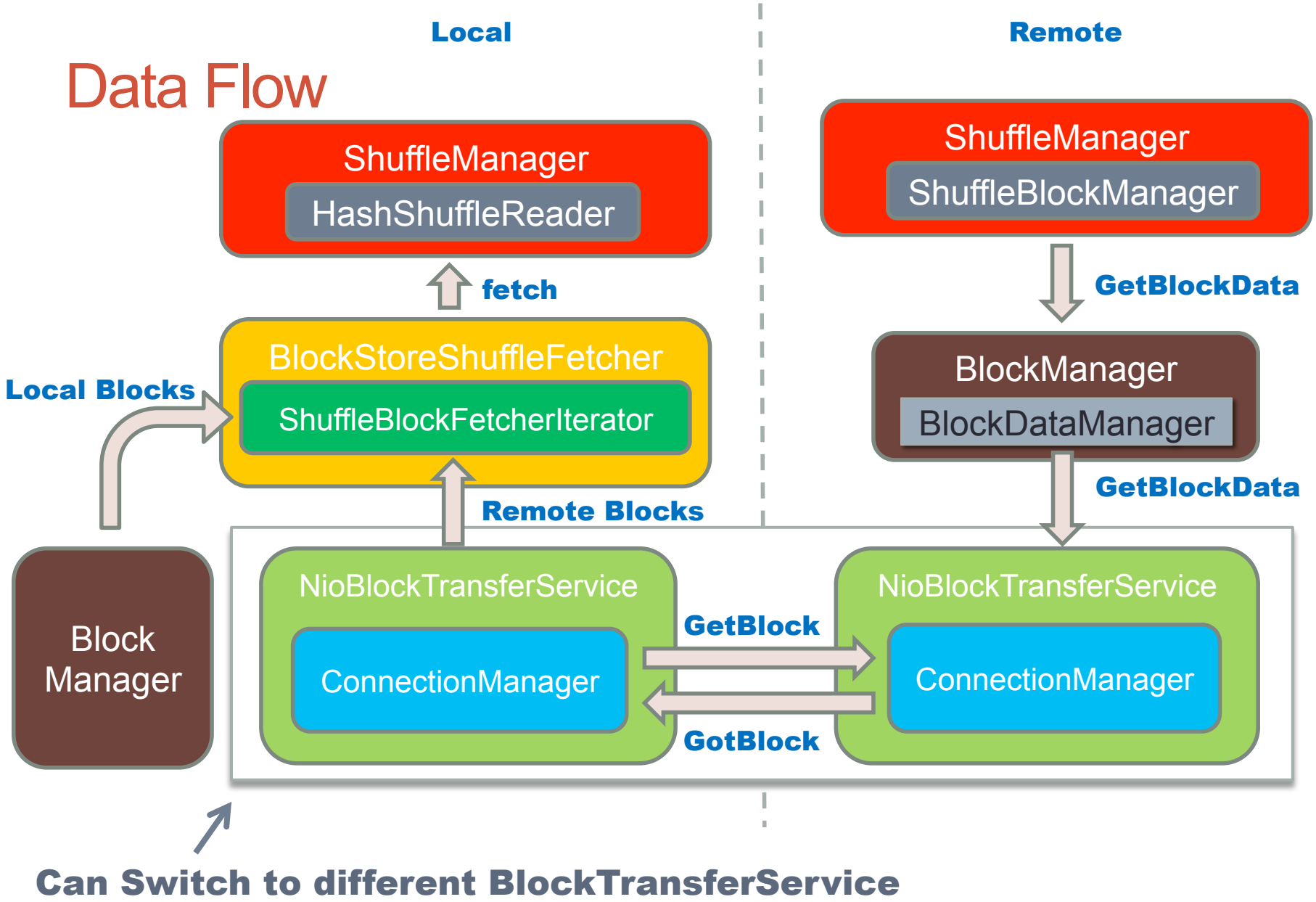
- Actually, at present, Sort Based Shuffle also go with HashShuffleReader


BLOCK
TRANSFER
SERVICE
Photo By Raymond

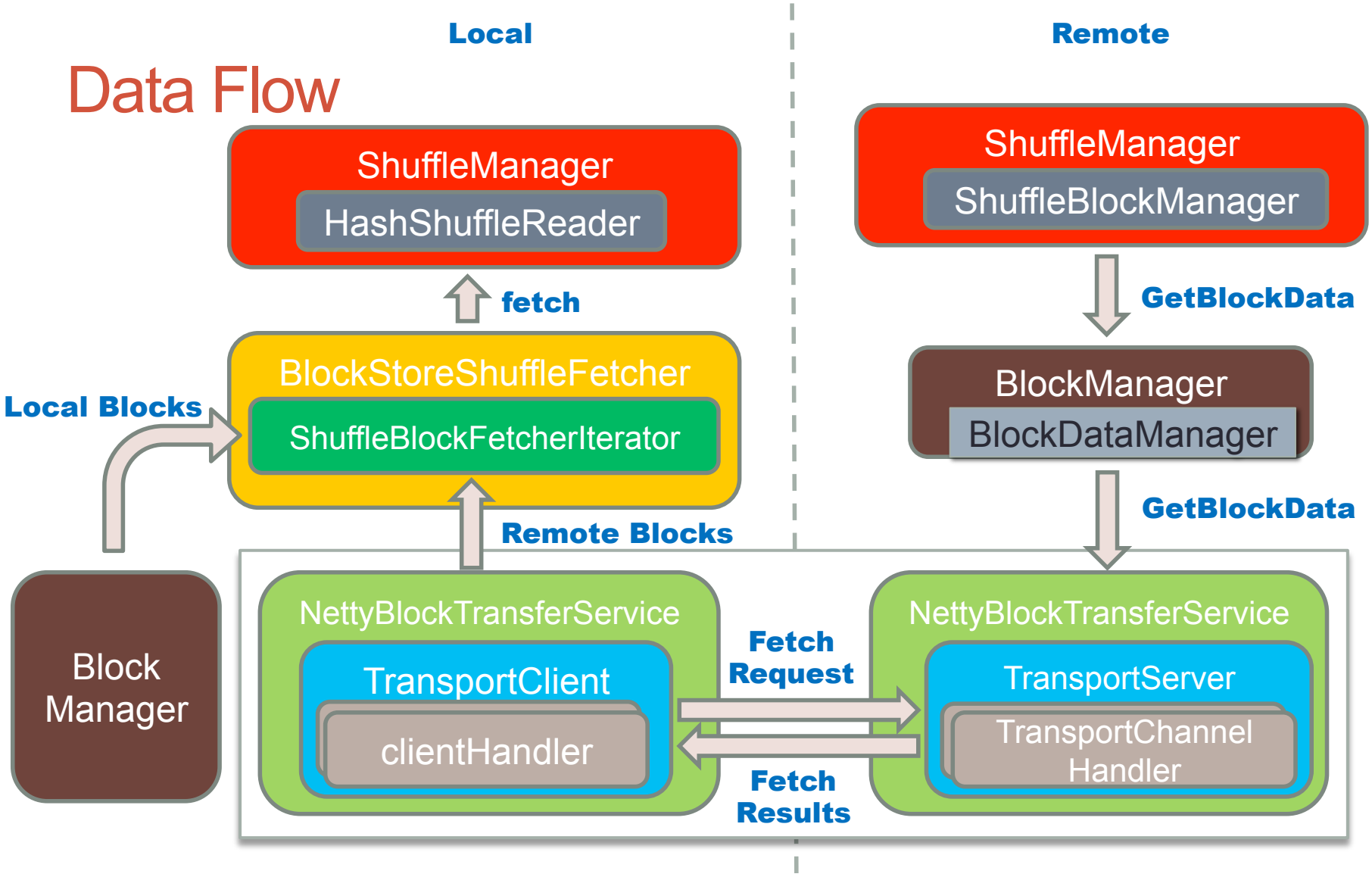
Related conceptions

- BlockTransferService
  - Provide a general interface for ShuffleFetcher and working with BlockDataManager to get local data.
- ShuffleClient
  - Wrap up the fetching data process for the client side, say setup TransportContext, new TransportClient etc.
- TransportContext
  - Context to setup the transport layer
- TransportServer
  - low-level streaming service server
- TransportClient
  - Client for fetching consecutive chunks TransportServer

Data Flow


Local
Remote
ShuffleManager
HashShuffleReader
ShuffleManager
ShuffleBlockManager
fetch
GetBlockData
BlockStoreShuffleFetcher
ShuffleBlockFetcherIterator
Local Blocks
BlockManager
BlockDataManager
GetBlockData
Remote Blocks
Block Manager
NioBlockTransferService
ConnectionManager
GetBlock
GotBlock
NioBlockTransferService
ConnectionManager
Can Switch to different BlockTransferService

Data Flow


Local
Remote
ShuffleManager
HashShuffleReader
fetch
BlockStoreShuffleFetcher
ShuffleBlockFetcherIterator
Local Blocks
Remote Blocks
Block Manager
NettyBlockTransferService
TransportClient
clientHandler
Fetch Request
Fetch Results
ShuffleManager
ShuffleBlockManager
GetBlockData
BlockManager
BlockDataManager
GetBlockData
NettyBlockTransferService
TransportServer
TransportChannel Handler

Future, Brilliant or Transient ?

External Shuffle Service

- Design goal
  - allow for the service to be long-running
    - possibly much longer-running than Spark
    - support multiple version of Spark simultaneously etc.
  - can be integrated into YARN NodeManager, Standalone Worker, or on its own

- The entire service been ported to Java
  - do not include Spark's dependencies
  - full control over the binary compatibility of the components
  - not depend on the Scala runtime or version.

External Shuffle Service

- Current Status
  - Basic framework seems ready.
    - A Network module extracted from the core module
  - BlockManager could be configured with executor built-in shuffle service or external standalone shuffle service
  - A standaloneWorkerShuffleService could be launched by worker
  - Disabled by default.

- How it works
  - Shuffle data is still written by the shuffleWriter to local disks.
  - The external shuffle service knows how to read these files on disks (executor will registered related info to it, e.g. shuffle manager type, file dir layout etc.), it follow the same rules applied for written these file, so it could serve the data correctly.

Sort Merge Shuffle Reader

- Background:
  - Current HashShuffleReader does not utilize the sort result within partition in map-side.
  - The actual by key sort work is always done at reduce side.
  - While the map side will do by-partition sort anyway （ sort shuffle ）
    - Change it to a by-key-and-partition sort does not bring many extra overhead.

- Current Status
  - [WIP] https://github.com/apache/spark/pull/3438

Some shuffle related configs

- spark.shuffle.spill (true)
- spark.shuffle.memoryFraction (0.2)
- spark.shuffle.manager [sort]/hash
- spark.shuffle.sort.bypassMergeThreshold (200)
- spark.shuffle.blockTransferService [netty]/nio

- spark.shuffle.consolidateFiles (false)
- spark.shuffle.service.enabled (false)

What's next?

- Other custom shuffle logic?
  - Alternative way to save shuffle data blocks
    - E.g. in memory (again)
- Other transport mechanism?
- Break stage barrier?
  - To fetch shuffle data when part of the map tasks are done.
  - Push mode instead of pull mode?

Thanks to Jerry Shao

- Some of this ppt's material came from Jerry Shao@Intel weibo: @saisai_shao

- Jerry also contributes a lot of essential patches for spark core / spark streaming etc.

Join Us !

- 加盟 / 合作 / 讨论 统统欢迎
- 数据平台开发，大数据相关技术，只要够 Cool，我们都玩
- tianhuo@mogujie.com

蘑菇街
mogujie.com


QUESTIONS ?