

# Нadoop. MapReduce. Примеры Join


Возможности `hadoop.mapreduce` в реальных  
задачах

# Join

- Задача состоит в том чтобы связать два набора данных по foreign ключу

id	systema	systemb
1	system1	system2
2	system1	system3
3	system3	system1

code	name
system1	МВД
system2	ФМС
system3	РосРеестр

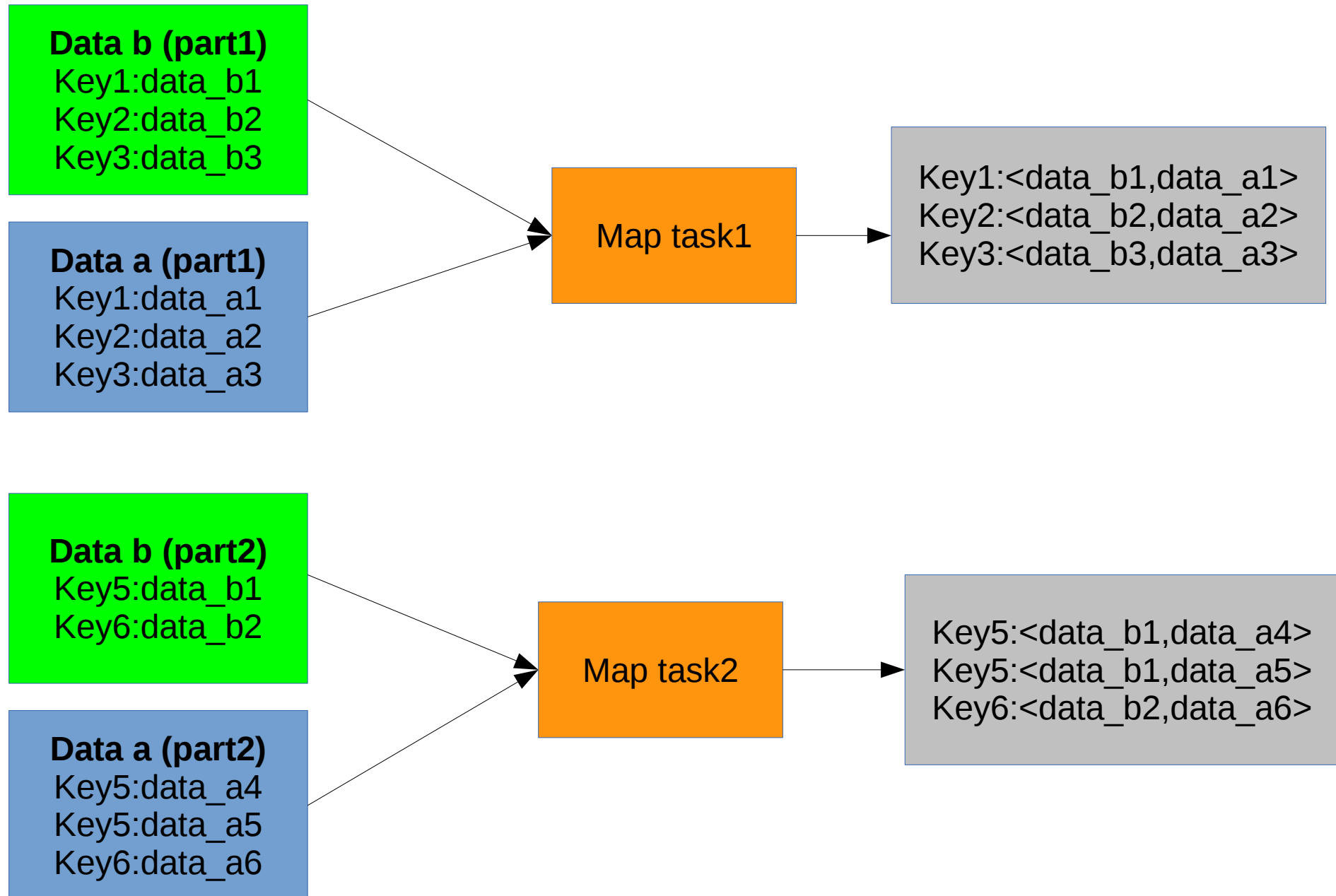


id	systema	System a name	systemb
1	system1	МВД	system2
2	system2	ФМС	system3
3	system3	BC	system1

# Map side join

- Join осуществляется на шаге MAP
- На исходные данные накладываются требования :
  - Каждый набор данных должен быть разбит на одинаковое число partitions
  - Данные в каждом наборе должны быть отсортированы по одному и тому же ключу – join key
  - Все данные с одним и тем же ключом должны быть в одном и том же partition

# Map side join



# Реализация map side join

- Используем класс `CompositeInputFormat`
- Составляется выражение, описывающее наборы данных и вид join
- `CompositeInputFormat.compose("inner",`
- `KeyValueTextInputFormat.class,`
- `file1,`
- `file2`
- `));`
- На вход mapper поступает `TupleWritable` в который входят данные из обоих наборов данных

# Пример Map Side join

- `JobConf conf = new JobConf(JoinJob.class);`
- `conf.setJobName("map join");`
- `conf.setInputFormat(CompositeInputFormat.class);`
- `FileOutputFormat.setOutputPath(conf, new Path(args[2]));`
- `conf.set("mapred.join.expr", CompositeInputFormat.compose("inner",`
- `KeyValueTextInputFormat.class,`
- `args[0],`
- `args[1]`
- `));`
- 
- `conf.setMapperClass(MapJoinMapper.class);`
- `conf.setOutputKeyClass(Text.class);`
- `conf.setOutputValueClass(Text.class);`
- `JobClient.runJob(conf);`

# Map side join mapper

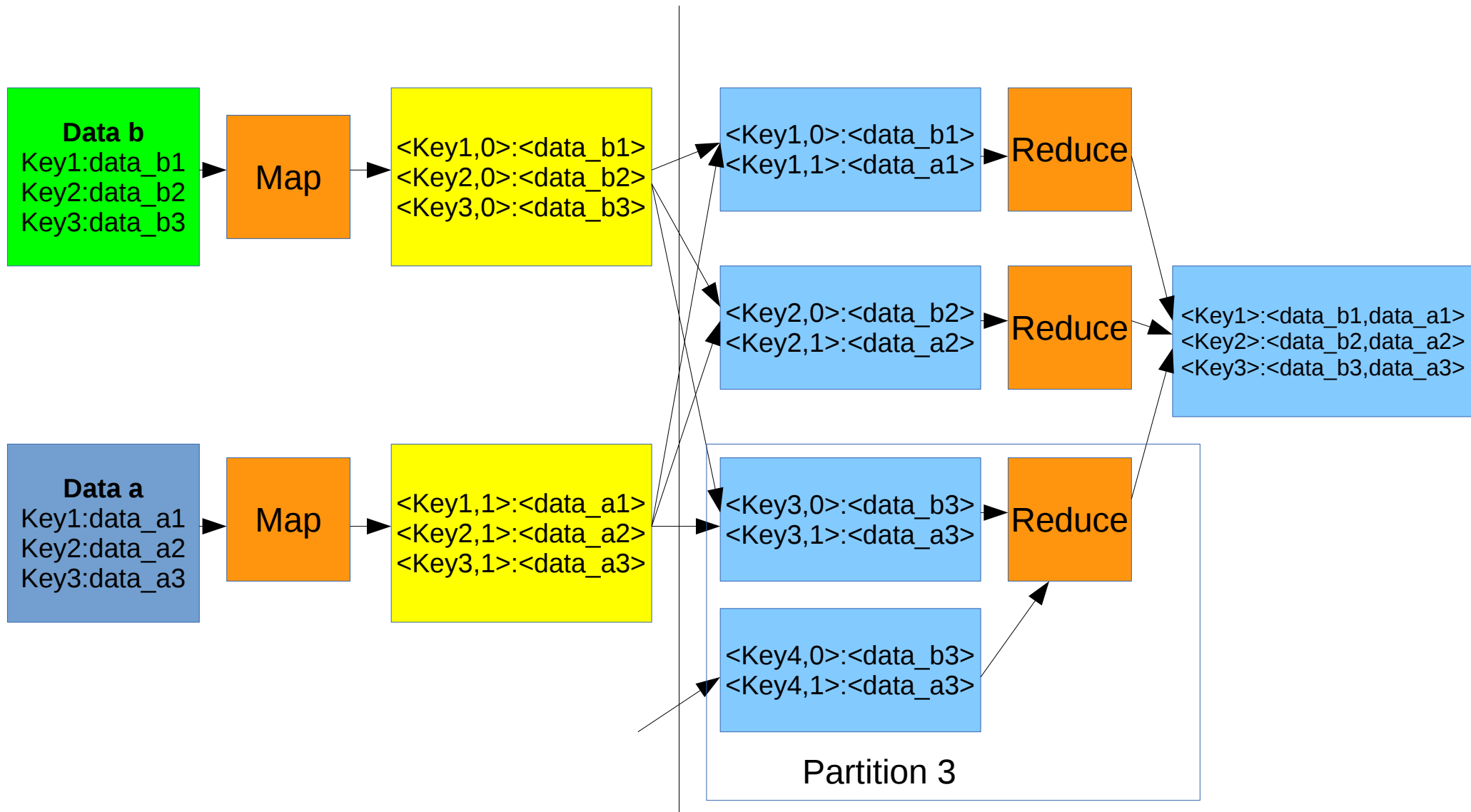
- public class MapJoinMapper extends MapReduceBase implements Mapper<Text, TupleWritable, Text, Text> {
- @Override
- public void map(Text key, TupleWritable value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {
- Text call = (Text) value.get(0);
- Text system = (Text) value.get(1);
- output.collect(call, system);
- }
- 
- }

# Reduce side join

- Менее эффективен чем map side join
- Не накладывает жестких требований на входные данные
- Использует MultipleInputs
- MultipleInputs позволяет добавить несколько наборов входных данных и каждому набору указать свой mapper
- Используем технику аналогичную сортировке по ключу и значению. Но в качестве дополнительного значения используем 0 для набора данных справочника и 1 для набора данных основной таблицы
- На вход reducer подаются наборы данных в которых в первой строчке будет строка справочника, а в последующих строки основной таблицы



# Reduce side join



# Пример reduce side join

- `public class CallsJoinMapper extends Mapper<LongWritable, Text, TextPair, Text> {`
- `@Override`
- `protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {`
- `ServiceCall call = new ServiceCall(value);`
- `context.write(new TextPair(call.getSystemA().toString(),"1"),`
- `new Text(call.toString()));`
- `}`
- `}`
- `public class SystemsJoinMapper extends Mapper<LongWritable, Text, TextPair, Text> {`
- `@Override`
- `protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {`
- `SystemInfo system = new SystemInfo(value);`
- `context.write(new TextPair(system.getSystemCode().toString(),"0"), new Text(system.toString()));`
- `}`
- `}`

# Reducer

- public class JoinReducer extends Reducer<TextPair, Text, Text, Text> {
- @Override
- protected void reduce(TextPair key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
- Iterator<Text> iter = values.iterator();
- Text systemInfo = new Text(iter.next());
- while (iter.hasNext()) {
- Text call = iter.next();
- Text outValue = new Text(call.toString() + "\t" + systemInfo.toString());
- context.write(key.getFirst(), outValue);
- }
- }
- }

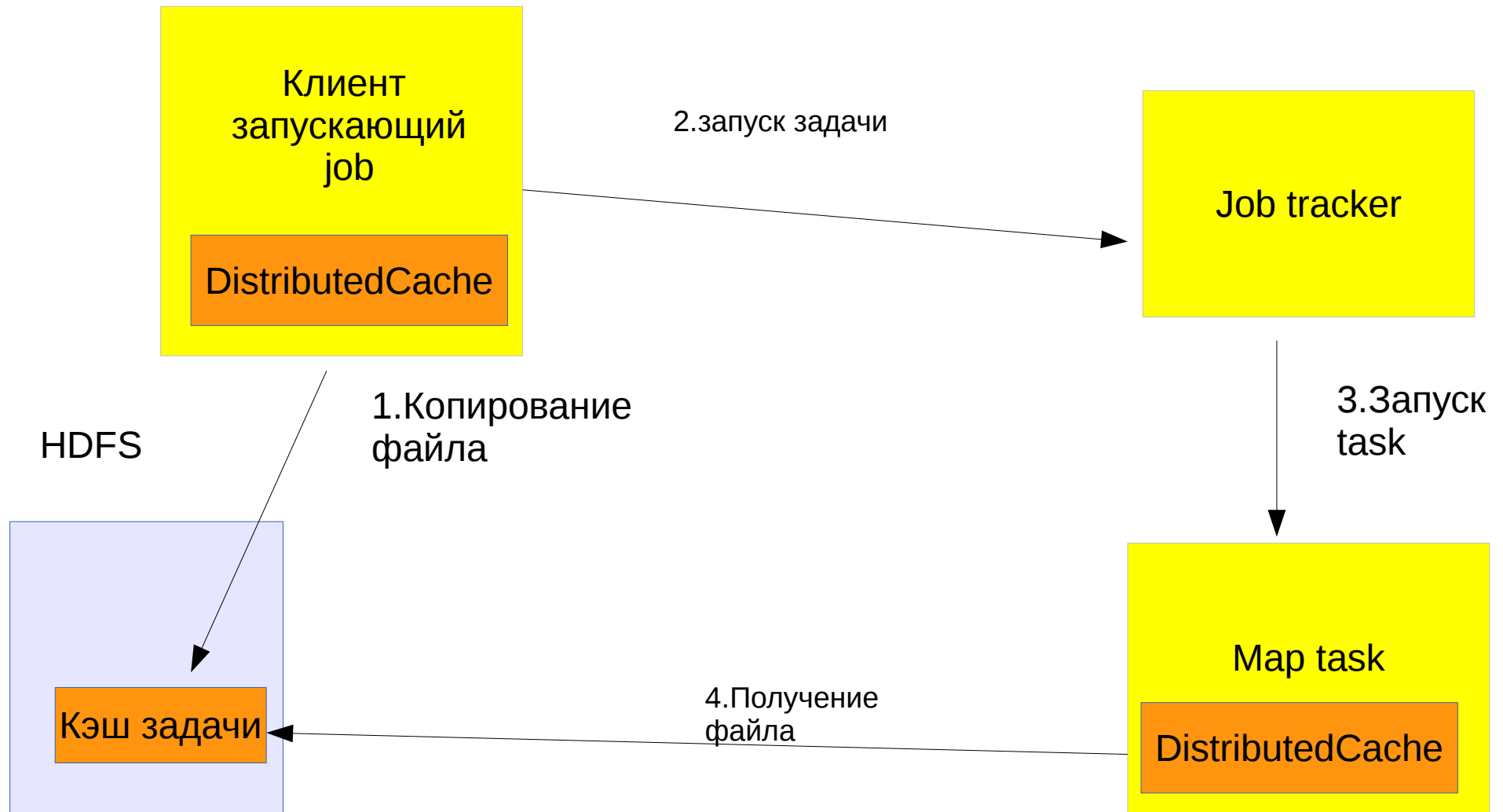
# job

- `Job job = Job.getInstance();`
- `job.setJarByClass(JoinJob.class);`
- `job.setJobName("JoinJob sort");`
- `MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, CallsJoinMapper.class);`
- `MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, SystemsJoinMapper.class);`
- 
- `FileOutputFormat.setOutputPath(job, new Path(args[2]));`
- `job.setPartitionerClass(TextPair.FirstPartitioner.class);`
- `job.setGroupingComparatorClass(TextPair.FirstComparator.class);`
- `job.setReducerClass(JoinReducer.class);`
- `job.setMapOutputKeyClass(TextPair.class);`
- 
- `job.setOutputKeyClass(Text.class);`
- `job.setOutputValueClass(Text.class);`
- `job.setNumReduceTasks(2);`
- `System.exit(job.waitForCompletion(true) ? 0 : 1);`

# Distributed cache

- Позволяет добавлять в ресурсы job файлы с данными, архивами и классами.
- Перед запуском task, Hadoop скачивает из хранилища все файлы добавленные в кэш.
- Во время работы mapper и reducer файлы кеша доступны через api
- Эффективный способ join, когда один из справочников гарантировано поместится в памяти Task.

# Distributed cache



# Distributed cache

## Использование командной строки

- Используем класс  
`org.apache.hadoop.conf.Configured`
- В функции `main` запускаем приложение с помощью `ToolRunner`
- Добавляем в строку запуска параметры
  - files <добавляемые файлы>
  - archives <архивы>
  - libjars <jar файлы которые будут добавлены в classpath>
- Пример  
`hadoop example.join.cache.JoinJob -files systems.txt calls.txt result`

# Исходный код job

```
public class JoinJob extends Configured implements Tool {  
    public static void main(String[] args) throws Exception {  
        int res = ToolRunner.run(new Configuration(), new JoinJob(), args);  
        System.exit(res);  
    }  
    @Override  
    public int run(String[] args) throws Exception {  
        Configuration conf = getConf();  
        Job job = Job.getInstance(conf);  
        job.setJarByClass(JoinJob.class);  
        job.setJobName("CacheJob sort");  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        job.setMapperClass(CacheJoinMapper.class);  
        job.setOutputKeyClass(ServiceCall.class);  
        job.setOutputValueClass(Text.class);  
        job.setNumReduceTasks(2);  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
        return job.waitForCompletion(true) ? 0 : 1;  
    }  
}
```



# Исходный код mapper

```
public class CacheJoinMapper extends Mapper<LongWritable, Text, ServiceCall, Text> {  
    Map<String, SystemInfo> metadata;  
    @Override  
    protected void setup(Context context) throws IOException, InterruptedException {  
        metadata = SystemsFileParser.parseSystems(  
            new FileInputStream(new File("systems.txt"))  
        );  
    }  
    @Override  
    protected void map(LongWritable key, Text value, Context context) throws ... {  
        ServiceCall call = new ServiceCall(value);  
        SystemInfo systemA = metadata.get(call.getSystemA().toString());  
        context.write(call, systemA.getSystemName());  
    }  
}
```

# Использование API DistributedCache

- Используем API для добавления файла в main классе
  - `DistributedCache.addCacheFile(...)`
- Внутри mapper и reducer обращаемся к файлу с помощью API
  - `DistributedCache.getLocalCacheFiles(...)`

# Исходный код job

- `public class JoinJobAddCache {`
- `public static void main(String[] args) throws Exception {`
- `Job job = Job.getInstance();`
- `DistributedCache.addCacheFile(new File(args[0]).toURI(), job.getConfiguration());`
- `job.setJarByClass(JoinJobAddCache.class);`
- `job.setJobName("CacheJob sort");`
- `FileInputFormat.addInputPath(job, new Path(args[1]));`
- `FileOutputFormat.setOutputPath(job, new Path(args[2]));`
- `job.setMapperClass(DistributedCacheJoinMapper.class);`
- `job.setOutputKeyClass(ServiceCall.class);`
- `job.setOutputValueClass(Text.class);`
- `job.setNumReduceTasks(2);`
- `System.exit(job.waitForCompletion(true) ? 0 : 1);`
- `}`
- `}`

# Исходный код mapper

- `public class DistributedCacheJoinMapper extends Mapper<LongWritable, Text, ServiceCall, Text> {`
- `Map<String, SystemInfo> metadata;`
- `@Override`
- `protected void setup(Context context) throws IOException, InterruptedException {`
- `try {`
- `FileSystem fs = FileSystem.getLocal(new Configuration());`
- `Path[] files = DistributedCache.getLocalCacheFiles(context.getConfiguration());`
- `metadata = SystemsFileParser.parseSystems(fs.open(files[0]));`
- `} catch (Exception err) {`
- `throw new RuntimeException(err);`
- `}`
- `}`
- `@Override`
- `protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException`
- `{`
- `ServiceCall call = new ServiceCall(value);`
- `context.write(call, metadata.get(call.getSystemA().toString()).getSystemName());`
- `}`
- `}`