Hadoop应用开发实战 第14周

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

课程大纲

- 项目背景：分布式消息中间件
- 需求分析：业务系统升级方案
- 架构设计：搭建Zookeeper的分步式协作平台
- 程序开发：基于Zookeeper的程序设计
- 程序运行

项目背景: 系统集成

- 软件系统集成一直是工业界的一个难题，像10年以上的遗留系统集成，公司收购后的多系统集成，全球性的分步式系统集成等。虽然基于SOA的软件架构，从理论上都可以解决这些集成的问题，但是具体实施过程，有些集成项目过于复杂而失败。

- 随着技术的创新和发展，对于分步式集群应用的集成，有了更好的开源软件的支持，像zookeeper就是一个不错的分步式协作软件平台。本文将通过一个案例介绍Zookeeper的强大。

项目背景：Hadoop集群

- 随着Hadoop的普及，越来越多的公司开始构建自己的Hadoop系统。有时候，公司内部的不同部门或不同的团队，都有自己的Hadoop集群。这种多集群的方式，既能让每个团队拥有个性化的Hadoop，又能避免大集群的高度其中化运维难度。当数据量不是特别巨大的时候，小型集群会有很多适用的场合。

- 当然，多个小型集群也有缺点，就是资源配置可能造成浪费。每个团队的Hadoop集群，都要配有服务器和运维人员。有些能力强的团队，构建的hadoop集群，可以达到真正的个性化要求；而有一些能力比较差的团队，搭建的Hadoop集群性能会比较糟糕。

项目背景：SOA

- 有一些时候，多个团队需要共同完成一个任务，比如，A团队通过Hadoop集群计算的结果，交给B团队继续工作，B完成了自己任务再交给C团队继续做。这就有点像业务系统的工作流一样，一环一环地传下去，直到最后一部分完成。

- 在业务系统中，我们经常会用SOA的架构来解决这种问题，每个团队在ESB服务器上部署自己的服务，然后通过消息中间件完成调度任务。对于分步式的多个Hadoop集群系统的协作，同样可以用这种架构来做，只要把消息中间件引擎换成支持分步式的消息中间件的引擎就行了。

项目背景：Zookeeper

- Zookeeper就可以做为 分步式消息中间件，来完成上面的说的业务需求。
- Zookeeper是Hadoop家族的一款高性能的分布式协作的产品，是一个为分布式应用所设计的分布的、开源的协调服务，它主要是用来解决分布式应用中经常遇到的一些数据管理问题，简化分布式应用协调及其管理的难度，提供高性能的分布式服务。

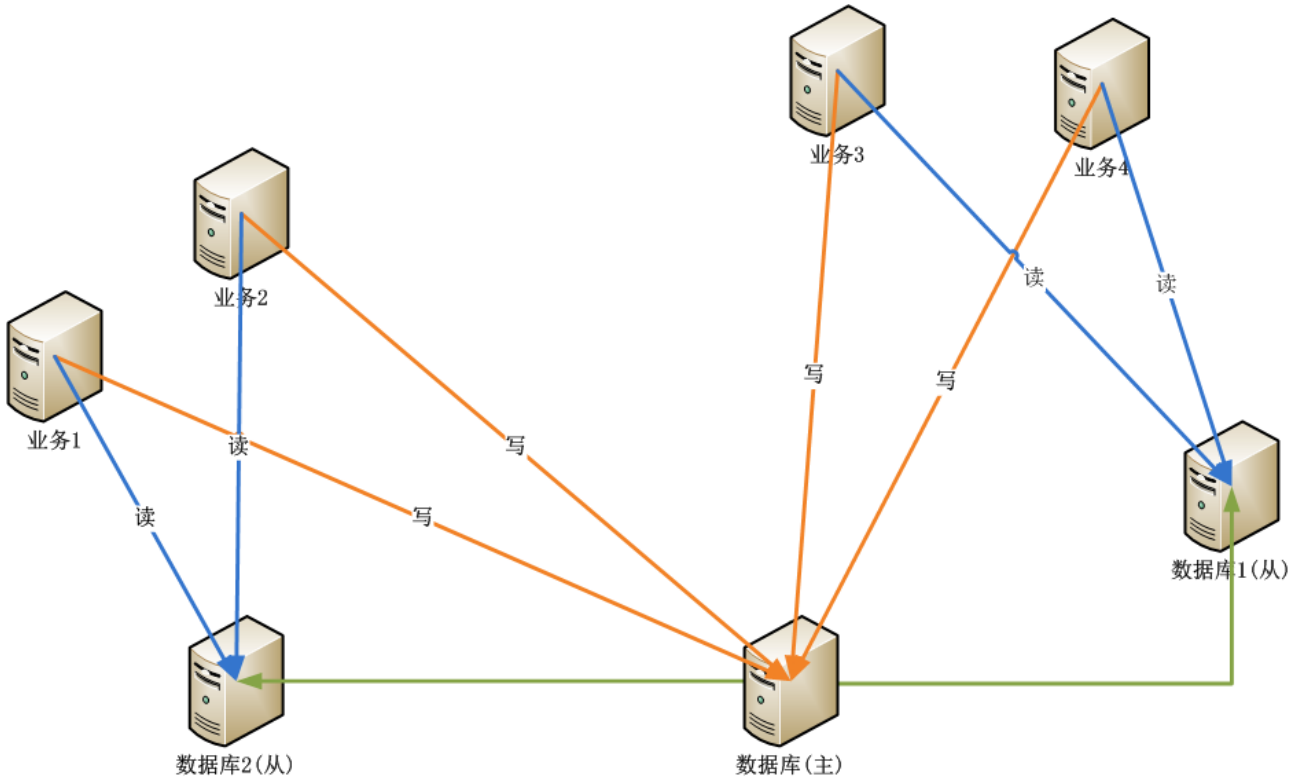
- Zookeeper的安装和使用，请参考文章 ZooKeeper伪分布式集群安装及使用。
- Zookeeper提供分布式协作服务，并不需要依赖于Hadoop的环境。

需求分析：案例介绍

- 某大型软件公司，从事领域为供应链管理，主要业务包括了 采购管理、应付账款管理、应收账款管理、供应商反复管理、退货管理、销售管理、库存管理、电子商务、系统集成等。

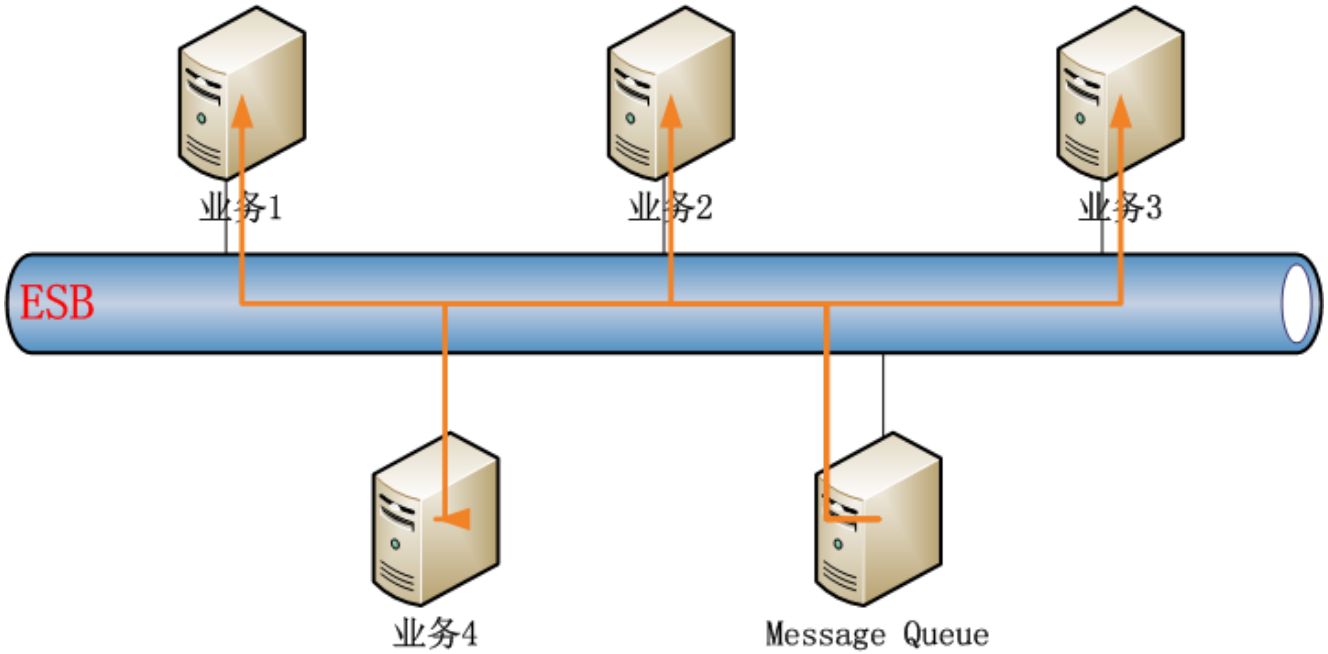
需求分析：数据库共享

- 每块业务的逻辑都很复杂，由单独部门进行软件开发和维护，部门之间的系统没有直接通信需求，每个部门完成自己的功能就行了，最后通过数据库来共享数据，实现各功能之间的数据交换。



需求分析：ESB + Message Queue

- 随着业务的发展，客户对响应速度要求越来越高，通过数据库来共享数据的方式，已经达不到信息交换的要求，系统进行了第一次升级，通过企业服务总线(ESB)统一管理公司内部所有业务。通过WebServices发布服务，通过Message Queue实现业务功能的调度。

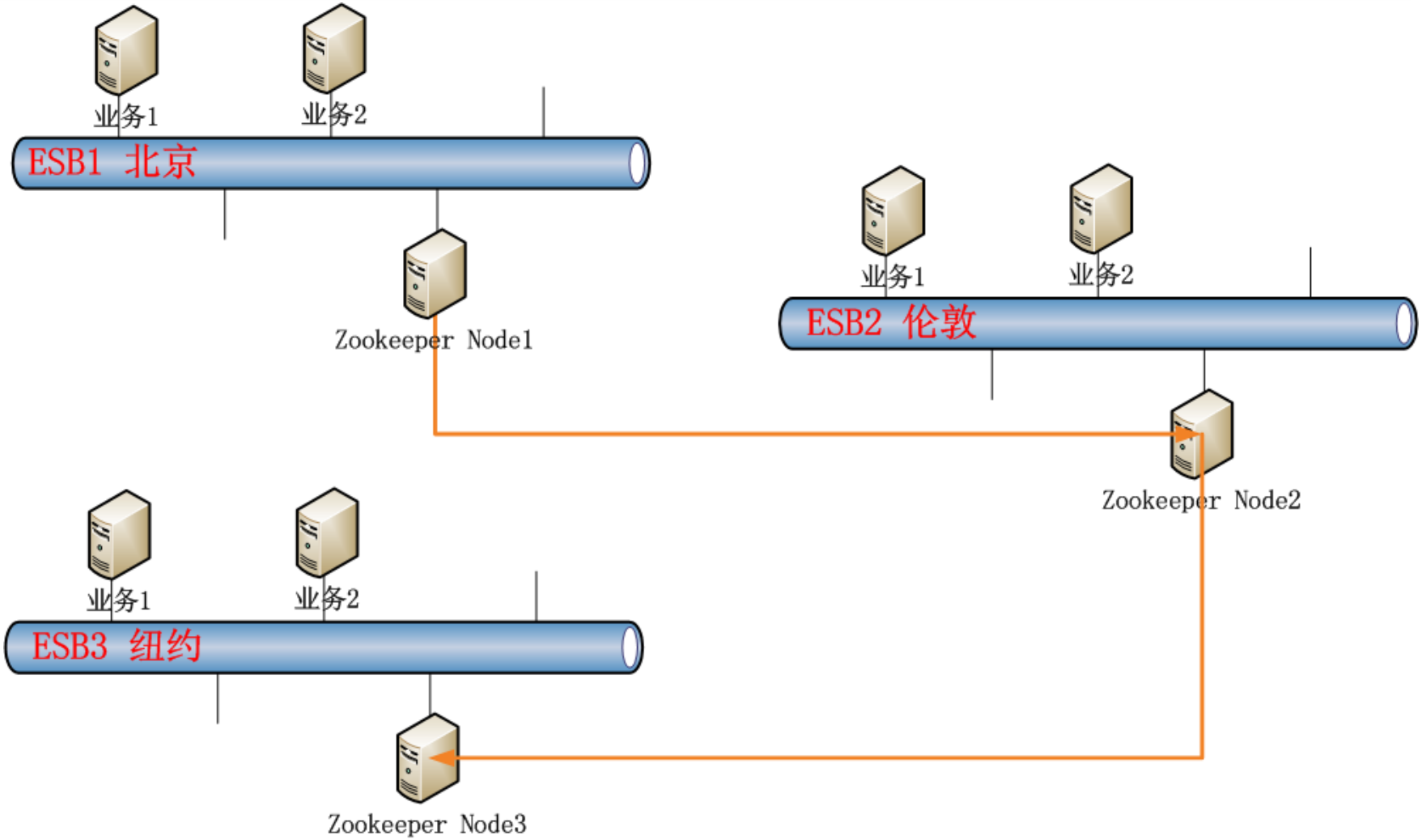


需求分析：ESB + Zookeeper

- 公司业务规模继续扩大，跨国收购了多家公司。业务系统从原来的一个机房的集中式部署，变成了全球性的多机房的分步式部署。这时，Message Queue已经不能满足多机房跨地域的业务系统的功能需求了，需要一种分步式的消息中间件解决方案，来代替原有消息中间件的服务。

- 系统进行了第二次升级，采用Zookeeper作为分步式中间件调度引擎。

需求分析：ESB + Zookeeper



需求分析：功能需求

- 全球性采购业务和全球性销售业务，让公司在市场中处于竞争优势。但由于采购和销售分别是由不同部门进行的软件开发和维护，而且业务往来也在不同的国家和地区。所以在每月底结算时，工作量都特别大。

- 比如，计算利润表
- 当月利润 = 当月销售金额 - 当月采购金额 - 当月其他支出
  - 注：请不要纠结于公式的准确性

- 采购系统是单独的系统，销售是另外单独的系统，及以其他几十个大大小小的系统，如何能让多个系统，配合起来做这道计算题呢？？

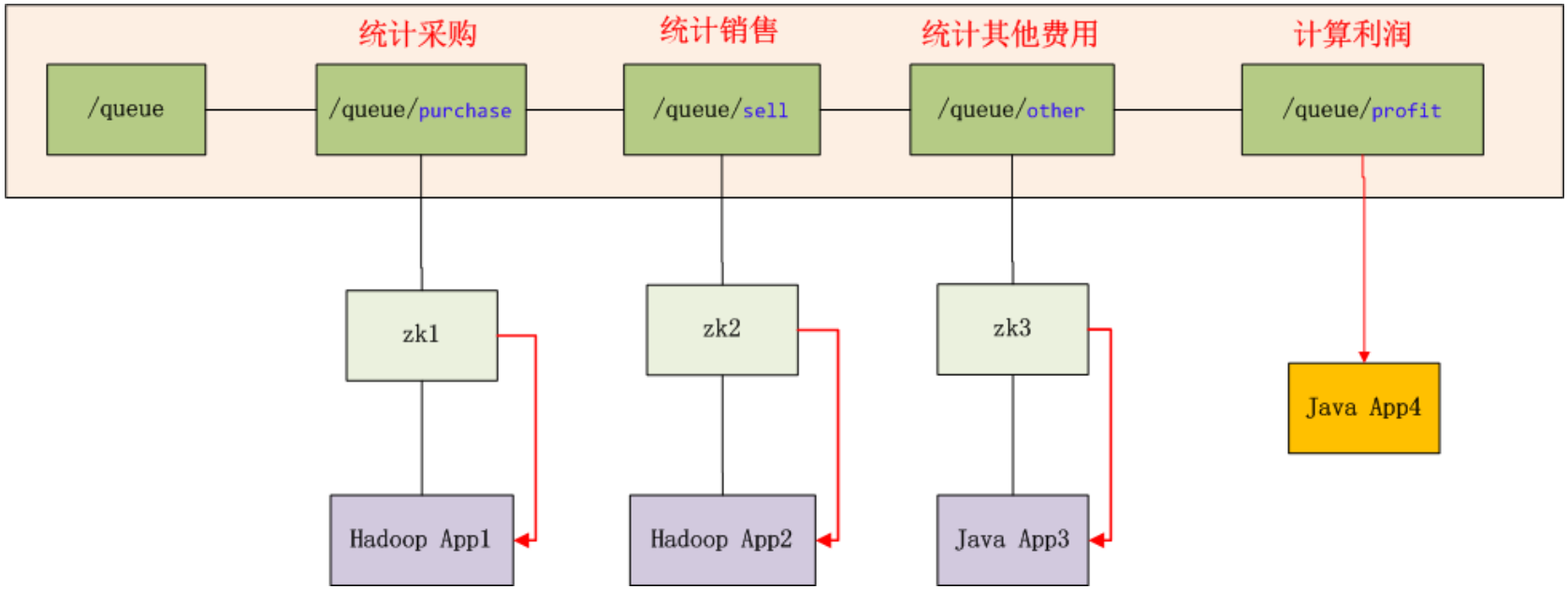
架构设计：系统边界

- 我们基于zookeeper来构建一个分步式队列的应用，来解决上面的功能需求。
  - 注：排除了ESB的部分，只保留zookeeper进行实现。

- 采购数据，为海量数据，基于Hadoop存储和分析。
- 销售数据，为海量数据，基于Hadoop存储和分析。
- 其他费用支出，为少量数据，基于文件或数据库存储和分析。

架构设计： Zookeeper分步式协作平台

- 我们设计一个同步队列，这个队列有3个条件节点，分别对应采购(purchase)，销售(sell)，其他费用(other)3个部分。当3个节点都被创建后，程序会自动触发计算利润，并创建利润(profit)节点。上面3个节点的创建，无顺序要求。每个节点只能被创建一次。



架构设计：系统环境

系统环境：2个独立的Hadoop集群，2个独立的Java应用，3个Zookeeper集群节点

图标解释：

- Hadoop App1,Hadoop App2 是2个独立的Hadoop集群应用
- Java App3,Java App4 是2个独立的Java应用
- zk1,zk2,zk3是ZooKeeper集群的3个连接点
- /queue，是znode的队列目录，假设队列长度为3
- /queue/purchase，是znode队列中，1号排对者，由Hadoop App1提交，用于统计采购金额。
- /queue/sell，是znode队列中，2号排对者，由Hadoop App2提交，用于统计销售金额。
- /queue/other，是znode队列中，3号排对者，由Java App3提交，用于统计其他费用支出金额。
- /queue/profit，当znode队列中满了，触发创建利润节点。
- 当/qeueu/profit被创建后，app4被启动，所有zk的连接通知同步程序(红色线)，队列已完成，所有程序结束。

架构设计：补充说明

- 补充说明：
  - 创建/queue/purchase,/queue/sell,/queue/other目录时，没有前后顺序，程序提交后，/queue目录下会生成对应该子目录
  - App1可以通过zk2提交，App2也可通过zk3提交。原则上，找最近路由最近的znode节点提交。
  - 每个应用不能重复提出，直到3个任务都提交，计算利润的任务才会被执行。
  - /queue/profit被创建后，zk的应用会监听到这个事件，通知应用，队列已完成！

- 这里的同步队列的架构更详细的设计思路，请参考文章 ZooKeeper实现分布式队列Queue

程序开发：功能需求

- 计算2013年01月的利润。

- 在真正企业开发时，我们的实验环境应该与需求是一致的，但我的硬件条件有限，因些做了一个简化的环境设置。
  - 把zookeeper的完全分步式部署的3台服务器集群节点的，改为一台服务器上3个集群节点。
  - 把2个独立Hadoop集群，改为一个集群的2个独立的MapReduce任务。

程序开发：实验环境

- 开发环境：
  - Win7 64bit
  - JDK 1.6
  - Maven3
  - Juno Service Release 2
  - IP：192.168.1.10
- Zookeeper服务器环境：
  - Linux Ubuntu 12.04 LTS 64bit
  - Java 1.6.0_29
  - Zookeeper: 3.4.5
  - IP: 192.168.1.201
  - 3个集群节点
- Hadoop服务器环境：
  - Linux Ubuntu 12.04 LTS 64bit
  - Java 1.6.0_29
  - Hadoop: 1.0.3
  - IP: 192.168.1.210

程序开发：实验数据 - 采购

- 一共4列
- 产品ID,产品数量,产品单价,采购日期。

```
1,26,1168,2013-01-08
2,49,779,2013-02-12
3,80,850,2013-02-05
4,69,1585,2013-01-26
5,88,1052,2013-01-13
6,84,2363,2013-01-19
7,64,1410,2013-01-12
8,53,910,2013-01-11
9,21,1661,2013-01-19
10,53,2426,2013-02-18
11,64,2022,2013-01-07
12,36,2941,2013-01-28
13,99,3819,2013-01-19
14,64,2563,2013-02-16
15,91,752,2013-02-05
16,65,750,2013-02-04
17,19,2426,2013-02-23
18,19,724,2013-02-05
19,87,137,2013-01-25
20,86,2939,2013-01-14
21,92,159,2013-01-23
22,81,2331,2013-03-01
23,88,998,2013-01-20
24,38,102,2013-02-22
25,32,4813,2013-01-13
26,36,1671,2013-01-19
```

程序开发：实验数据 – 销售

- 一共4列
- 产品ID,销售数量,销售单价,销售日期。

```
1, 14, 1236, 2013-01-14
2, 19, 808, 2013-03-06
3, 26, 886, 2013-02-23
4, 23, 1793, 2013-02-09
5, 27, 1206, 2013-01-21
6, 27, 2648, 2013-01-30
7, 22, 1502, 2013-01-19
8, 20, 1050, 2013-01-18
9, 13, 1778, 2013-01-30
10, 20, 2718, 2013-03-14
11, 22, 2175, 2013-01-12
12, 16, 3284, 2013-02-12
13, 30, 4152, 2013-01-30
14, 22, 2770, 2013-03-11
15, 28, 778, 2013-02-23
16, 22, 874, 2013-02-22
17, 12, 2718, 2013-03-22
18, 12, 747, 2013-02-23
19, 27, 172, 2013-02-07
20, 27, 3282, 2013-01-22
21, 28, 224, 2013-02-05
22, 26, 2613, 2013-03-30
23, 27, 1147, 2013-01-31
24, 16, 141, 2013-03-20
25, 15, 5343, 2013-01-21
26, 16, 1887, 2013-01-30
```

程序开发：实验数据 – 其他费用

- 一共2列
- 发生日期，发生金额

```
2013-01-02,552
2013-01-03,1092
2013-01-04,1794
2013-01-05,435
2013-01-06,960
2013-01-07,1066
2013-01-08,1354
2013-01-09,880
2013-01-10,1992
2013-01-11,931
2013-01-12,1209
2013-01-13,1491
2013-01-14,804
2013-01-15,480
2013-01-16,1891
2013-01-17,156
2013-01-18,1439
2013-01-19,1018
2013-01-20,1506
2013-01-21,1216
2013-01-22,2045
```

程序开发：程序设计

- 我们要编写5个文件：
  - 计算采购金额，Purchase.java
  - 计算销售金额，Sell.java
  - 计算其他费用金额，Other.java
  - 计算利润，Profit.java
  - Zookeeper的调度，ZookeeperJob.java

程序开发：计算采购金额 Purchase.java

```
    public static class PurchaseMapper extends Mapper {

        private String month = "2013-01";
        private Text k = new Text(month);
        private IntWritable v = new IntWritable();
        private int money = 0;

        public void map(LongWritable key, Text values, Context context) throws IOException,
            System.out.println(values.toString());
            String[] tokens = DELIMITER.split(values.toString());
            if (tokens[3].startsWith(month)) {// 1月的数据
                money = Integer.parseInt(tokens[1]) * Integer.parseInt(tokens[2]);//单价*数
                v.set(money);
                context.write(k, v);
            }
        }
    }

    public static class PurchaseReducer extends Reducer {
        private IntWritable v = new IntWritable();
        private int money = 0;

        @Override
        public void reduce(Text key, Iterable values, Context context) throws IOException, I
            for (IntWritable line : values) {
                // System.out.println(key.toString() + "\t" + line);
                money += line.get();
            }
            v.set(money);
            context.write(null, v);
            System.out.println("Output:" + key + "," + money);
        }

    }
```

程序开发：计算销售金额 Sell.java

```
public static class SellMapper extends Mapper {

    private String month = "2013-01";
    private Text k = new Text(month);
    private IntWritable v = new IntWritable();
    private int money = 0;

    public void map(LongWritable key, Text values, Context context) throws IOException,
        System.out.println(values.toString());
        String[] tokens = DELIMITER.split(values.toString());
        if (tokens[3].startsWith(month)) {// 1月的数据
            money = Integer.parseInt(tokens[1]) * Integer.parseInt(tokens[2]);//单价*数
            v.set(money);
            context.write(k, v);
        }
    }
}

public static class SellReducer extends Reducer {
    private IntWritable v = new IntWritable();
    private int money = 0;

    @Override
    public void reduce(Text key, Iterable values, Context context) throws IOException, I
        for (IntWritable line : values) {
            // System.out.println(key.toString() + "\t" + line);
            money += line.get();
        }
        v.set(money);
        context.write(null, v);
        System.out.println("Output:" + key + "," + money);
    }

}
```

程序开发：计算其他费用金额 Other.java

```
public class Other {

    public static String file = "logfile/biz/other.csv";
    public static final Pattern DELIMITER = Pattern.compile("[\t,]");
    private static String month = "2013-01";

    public static void main(String[] args) throws IOException {
        calcOther(file);
    }

    public static int calcOther(String file) throws IOException {
        int money = 0;
        BufferedReader br = new BufferedReader(new FileReader(new File(file)));

        String s = null;
        while ((s = br.readLine()) != null) {
            // System.out.println(s);
            String[] tokens = DELIMITER.split(s);
            if (tokens[0].startsWith(month)) {// 1月的数据
                money += Integer.parseInt(tokens[1]);
            }
        }
        br.close();

        System.out.println("Output:" + month + "," + money);
        return money;
    }
}
```

程序开发：计算利润 Profit.java

```
public class Profit {

    public static void main(String[] args) throws Exception {
        profit();
    }

    public static void profit() throws Exception {
        int sell = getSell();
        int purchase = getPurchase();
        int other = getOther();
        int profit = sell - purchase - other;
        System.out.printf("profit = sell - purchase - other = %d - %d - %d = %d\n", sell, pu
    }

    public static int getPurchase() throws Exception {
        HdfsDAO hdfs = new HdfsDAO(Purchase.HDFS, Purchase.config());
        return Integer.parseInt(hdfs.cat(Purchase.path().get("output") + "/part-r-00000").tr
    }

    public static int getSell() throws Exception {
        HdfsDAO hdfs = new HdfsDAO(Sell.HDFS, Sell.config());
        return Integer.parseInt(hdfs.cat(Sell.path().get("output") + "/part-r-00000").trim()
    }

    public static int getOther() throws IOException {
        return Other.calcOther(Other.file);
    }

}
```

程序开发：调度程序 ZookeeperJob.java

```
public class ZooKeeperJob {

    final public static String QUEUE = "/queue";
    final public static String PROFIT = "/queue/profit";
    final public static String PURCHASE = "/queue/purchase";
    final public static String SELL = "/queue/sell";
    final public static String OTHER = "/queue/other";

    public static void main(String[] args) throws Exception {
        if (args.length == 0) {
            System.out.println("Please start a task:");
        } else {
            doAction(Integer.parseInt(args[0]));
        }
    }

    public static void doAction(int client) throws Exception {
        String host1 = "192.168.1.201:2181";
        String host2 = "192.168.1.201:2182";
        String host3 = "192.168.1.201:2183";

        ZooKeeper zk = null;
        switch (client) {
        case 1:
            zk = connection(host1);
            initQueue(zk);
            doPurchase(zk);
            break;
        case 2:
            zk = connection(host2);
            initQueue(zk);
            doSell(zk);
            break;
```

```
    // 创建一个与服务器的连接
    public static ZooKeeper connection(String host) throws IOException {
        ZooKeeper zk = new ZooKeeper(host, 60000, new Watcher() {
            // 监控所有被触发的事件
            public void process(WatchedEvent event) {
                if (event.getType() == Event.EventType.NodeCreated && event.getPath().equals
                    System.out.println("Queue has Completed!!!");
                }
            }
        });
        return zk;
    }

    public static void initQueue(ZooKeeper zk) throws KeeperException, InterruptedException
        System.out.println("WATCH => " + PROFIT);
        zk.exists(PROFIT, true);

        if (zk.exists(QUEUE, false) == null) {
            System.out.println("create " + QUEUE);
            zk.create(QUEUE, QUEUE.getBytes(), Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT);
        } else {
            System.out.println(QUEUE + " is exist!");
        }
    }

    public static void doPurchase(ZooKeeper zk) throws Exception {
        if (zk.exists(PURCHASE, false) == null) {

            Purchase.run(Purchase.path());

            System.out.println("create " + PURCHASE);
            zk.create(PURCHASE, PURCHASE.getBytes(), Ids.OPEN_ACL_UNSAFE, CreateMode.PERSIST
        } else {
            System.out.println(PURCHASE + " is exist!");
        }
        isCompleted(zk);
    }
```

运行程序：

- 我们运行整个的程序，包括3个部分。
  - zookeeper服务器
  - hadoop服务器
  - 分步式队列应用

运行程序：zookeeper服务器

启动zookeeper服务器集群：

```
~ cd toolkit/zookeeper345

# 启动zk集群3个节点
~ bin/zkServer.sh start conf/zk1.cfg
~ bin/zkServer.sh start conf/zk2.cfg
~ bin/zkServer.sh start conf/zk3.cfg

~ jps
4234 QuorumPeerMain
5002 Jps
4275 QuorumPeerMain
4207 QuorumPeerMain
```

启动zookeeper客户端：

```
~ bin/zkCli.sh -server 192.168.1.201:2181

# 查看zk
[zk: 192.168.1.201:2181(CONNECTED) 0] ls /
[queue, queue-fifo, zookeeper]

# /queue路径无子目录
[zk: 192.168.1.201:2181(CONNECTED) 1] ls /queue
[]
```

运行程序：hadoop服务器

```
~ hadoop/hadoop-1.0.3
~ bin/start-all.sh

~ jps
25979 JobTracker
26257 TaskTracker
25576 DataNode
25300 NameNode
12116 Jps
25875 SecondaryNameNode
```

运行程序：计算采购金额

DATAGURU炼数成金

5.3.1 启动统计采购数据程序，设置启动参数1

只显示用户日志，忽略系统日志。

```
WATCH => /queue/profit
/queue is exist!
Delete: hdfs://192.168.1.210:9000/user/hdfs/biz/purchase
Create: hdfs://192.168.1.210:9000/user/hdfs/biz/purchase
copy from: logfile/biz/purchase.csv to hdfs://192.168.1.210:9000/user/hdfs/biz/purchase
Output:2013-01,9609887
create /queue/purchase
Queue Complete:1/3
```

在zk中查看queue目录

```
[zk: 192.168.1.201:2181(CONNECTED) 3] ls /queue
[purchase]
```

运行程序：计算销售金额

5.3.2 启动统计销售数据程序，设置启动参数2

只显示用户日志，忽略系统日志。

```
WATCH => /queue/profit
/queue is exist!
Delete: hdfs://192.168.1.210:9000/user/hdfs/biz/sell
Create: hdfs://192.168.1.210:9000/user/hdfs/biz/sell
copy from: logfile/biz/sell.csv to hdfs://192.168.1.210:9000/user/hdfs/biz/sell
Output:2013-01,2950315
create /queue/sell
Queue Complete:2/3
```

在zk中查看queue目录

```
[zk: 192.168.1.201:2181(CONNECTED) 5] ls /queue
[purchase, sell]
```

运行程序：计算其他费用金额

5.3.3 启动统计其他费用数据程序，设置启动参数3

只显示用户日志，忽略系统日志。

```
WATCH => /queue/profit
/queue is exist!
Output:2013-01,34193
create /queue/other
Queue Complete:3/3
create /queue/profit
cat: hdfs://192.168.1.210:9000/user/hdfs/biz/sell/output/part-r-00000
2950315

cat: hdfs://192.168.1.210:9000/user/hdfs/biz/purchase/output/part-r-00000
9609887

Output:2013-01,34193
profit = sell - purchase - other = 2950315 - 9609887 - 34193 = -6693765
Queue has Completed!!!
```

在zk中查看queue目录

```
[zk: 192.168.1.201:2181(CONNECTED) 6] ls /queue
[profit]
```

运行程序：利润结果

- 在最后一步，统计其他费用数据程序运行后，从日志中看到3个条件节点都已满足要求。然后，通过同步的分步式队列自动启动了计算利润的程序，并在日志中打印了2013年1月的利润为-6693765。

- profit = sell - purchase - other = 2950315 - 9609887 - 34193 = -6693765

程序开发: 补充资料

- 程序源代码下载：
- https://github.com/bsspirit/maven_hadoop_template/tree/master/src/main/java/org/conan/myzk/hadoop

- 补充资料：
- http://blog.fens.me/hadoop-zookeeper-case/
- http://blog.fens.me/hadoop-zookeeper-intro/
- http://blog.fens.me/zookeeper-queue/
- http://blog.fens.me/zookeeper-queue-fifo/

关于作者

- 张丹 (Conan)
- DataguruID: bsspirit
- Weibo: @Conan_Z
- Blog：http://blog.fens.me
- Email: bsspirit@gmail.com

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

ATAGURU 炼数成金

Thanks

FAQ时间