# Percolator Demo

**Presentation:** 陈渤、黄思、秦培杰
**Demo:** 丁峰、徐华韬

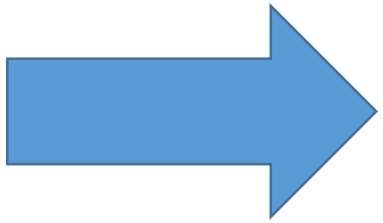
# Contents

# Contents

# Motivation

| Google |
| --- |
| Big Table |
| GFS |
| Chubby |

| Open Source Community |
| --- |
| HBase |
| HDFS |
| ZooKeeper |



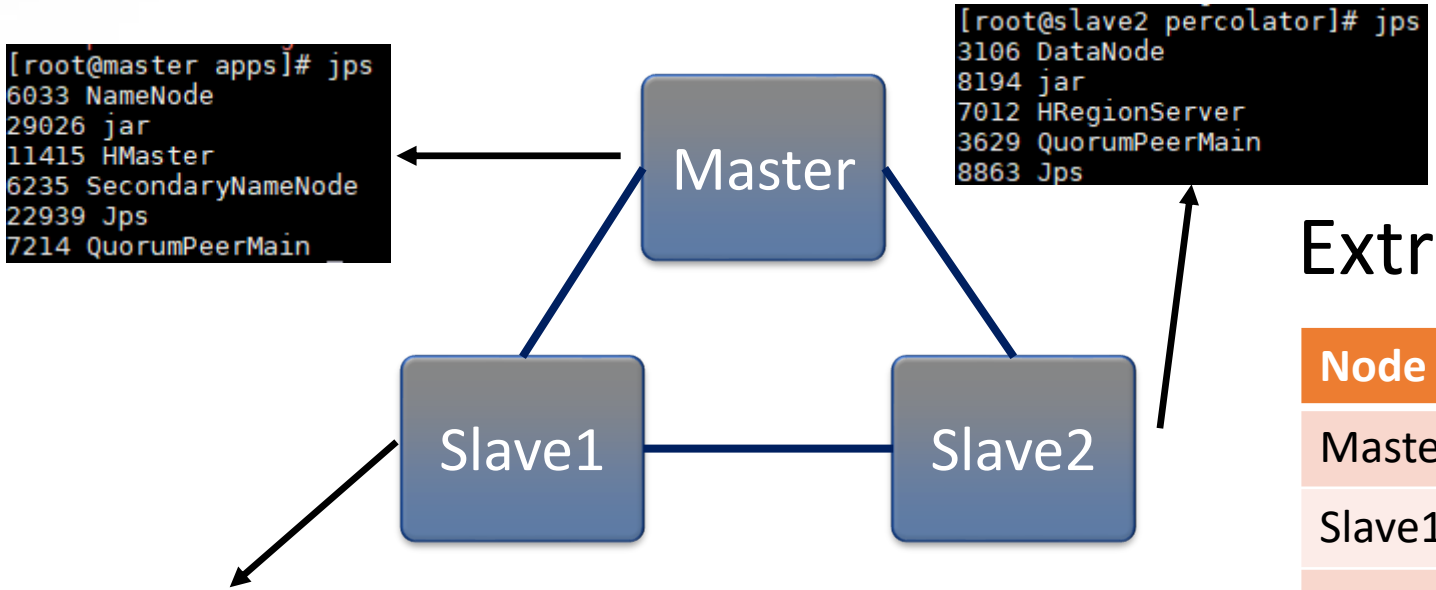
Percolator

Our Demo?

multi-row transaction, notification and observer

# Contents

# Infrastructure




| Software | Version |
|---|---|
| Centos | 7 |
| Hadoop | 2.7.3 |
| Hbase | 1.2.5 |
| ZooKeeper | 3.4.6 |
| JDK | 1.8 |

## Extra:

| Node | Address | Demo Apps |
|---|---|---|
| Master | 192.168.0.201 | SupprtServer |
| Slave1 | 192.168.0.202 | PercolatorDemo |
| Slave2 | 192.168.0.203 | PercolatorDemo |

## VM Info:

| Info | Capacity |
|---|---|
| Memory | 1 ~ 1.5GB |
| CPU | 3192.606MHZ, 1 core |
| Disk | 35GB |

# Contents

# Improvement

1. Transaction ID and its timeout ( 5s in our demo )

   - set start timestamp as Transaction ID, send it to Support Server
   - fine granularity (compared with client id)

2. Row transaction HBase

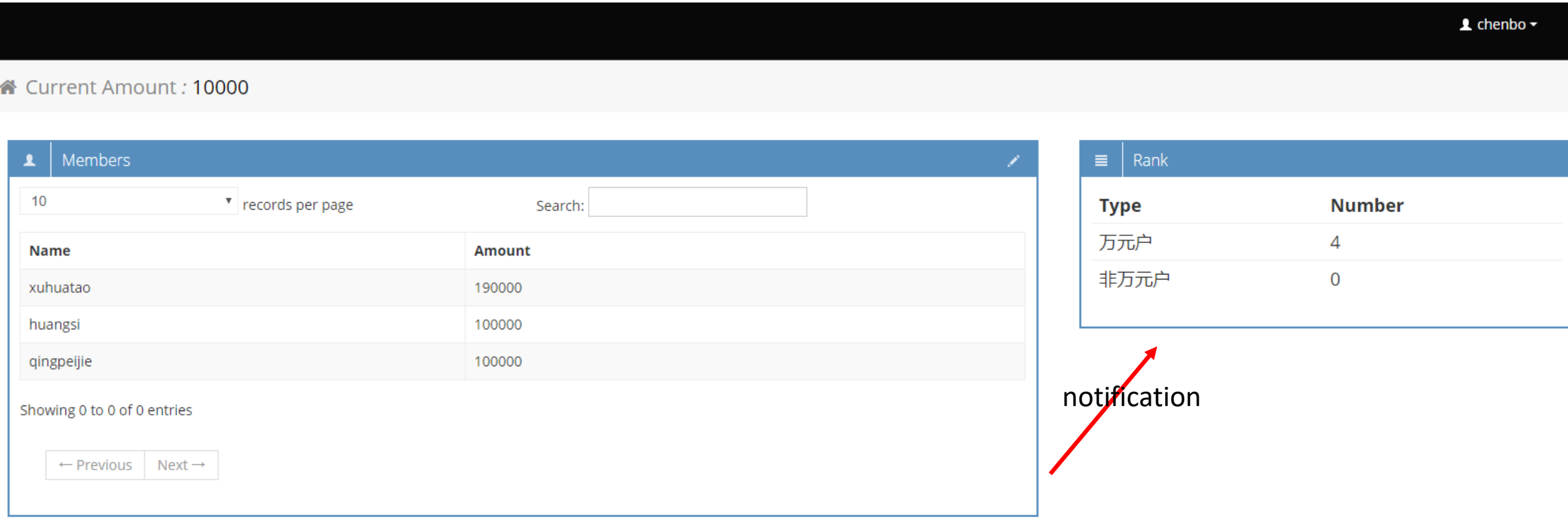
**1. Get lock**
**2. Read row**
**3. AtomicWrite Row**
**4. Release lock**

requirement:  distributed non-reentrant  lock (provided by ZooKeeper)

# Contents

# Scene

## Transferring Account



# Test Case 1

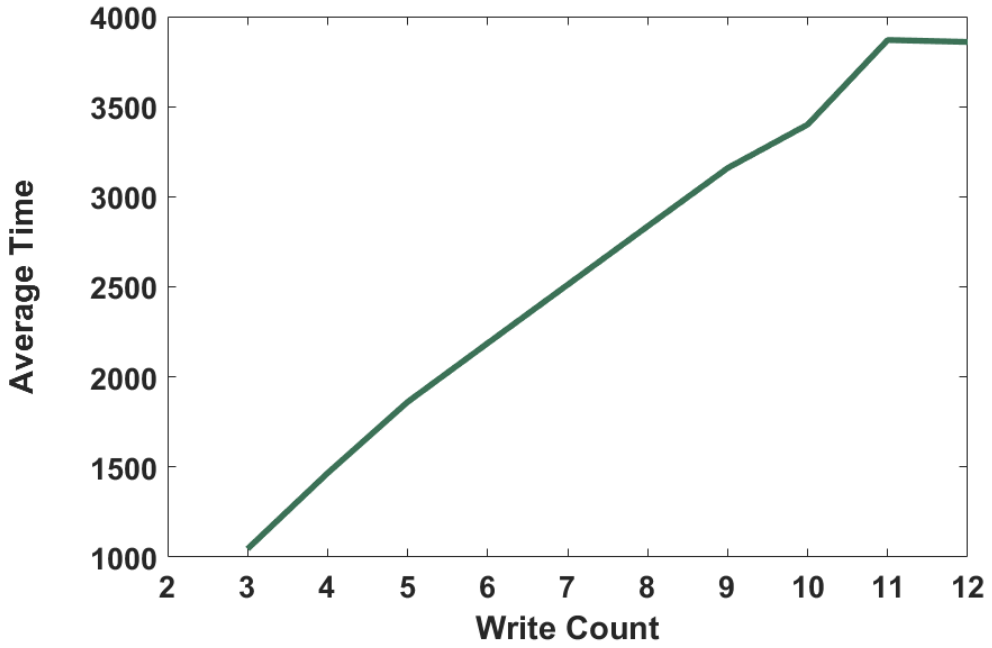
Configuration:

```
public static final int WRITE_SIZE = 10;   //每个事务的写次数在2~12次
public static AtomicLong[] timeRecords = new AtomicLong[WRITE_SIZE + 2];  //时间记录
public static AtomicInteger[] successCounts = new AtomicInteger[WRITE_SIZE + 2];  //成功用例的数量
public static AtomicInteger[] failureCounts = new AtomicInteger[WRITE_SIZE + 2];  //失败用例的数量
public static final int ACCOUNT_LENGTH = 200;  //200个数据规模
public static String[] accounts = new String[ACCOUNT_LENGTH];  //随机产生的账户
public static AtomicLong[] accountDatas = new AtomicLong[ACCOUNT_LENGTH];   //真实账户数据
public static final Long ACCOUNT_AMOUNT = 100000L;
public static final int EXECUTE_COUNT_PER_THREAD = 200;          //每个线程的事务执行次数
public static final long TRANSFER_AMOUNT = 1;
public static Map<String, Integer> accountMap = new ConcurrentHashMap<>();
public static final int WORKTASK_SIZE = 3;                       //线程数
public static WorkTask[] workTasks = new WorkTask[WORKTASK_SIZE];
```

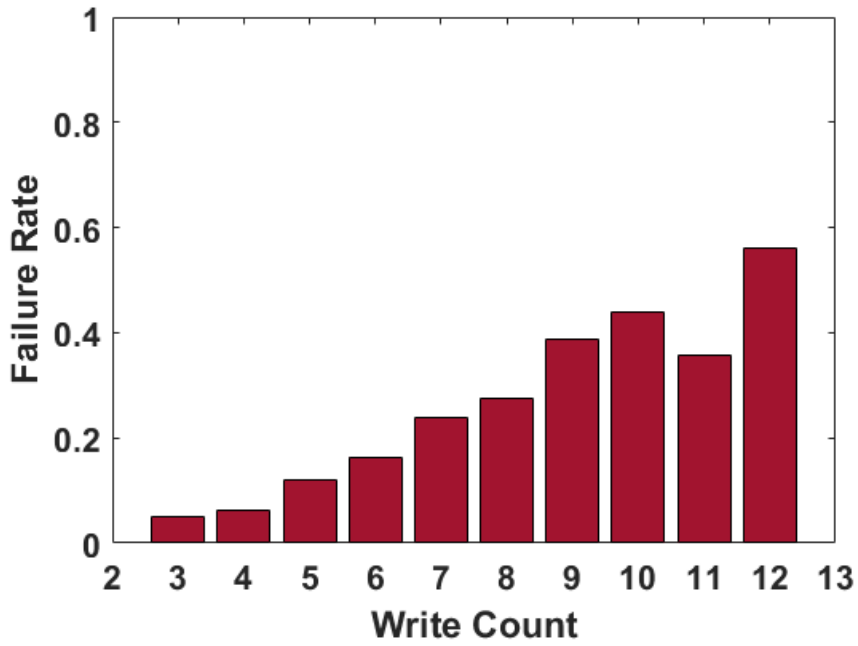
Steps:

```
initTest();
setTestData();
test();
boolean validateResult = validate();
if (validateResult) {
    System.out.println("validate success");
} else {
    System.out.println("validate failure");
}
removeTestData();
```

# Result 1




Average Time – Write Count




Failure Rate – Write Count

# Test Case 2

- Normal Transaction A
  - operate account normally
  - view data from in 3D (row-col-version)

- Abnormal Transaction A
  - set break point for A to cause its timeout
  - Transaction B cleans up A’s lock and commits successfully

# Contents

# Conclusion

- Motivation
  - A open source version of Percolator

- Scene and Test Case
  - High Concurrent Case
  - Normal Case
  - Abnormal Case

github: git@github.com:dingfeng/Percolator.git