



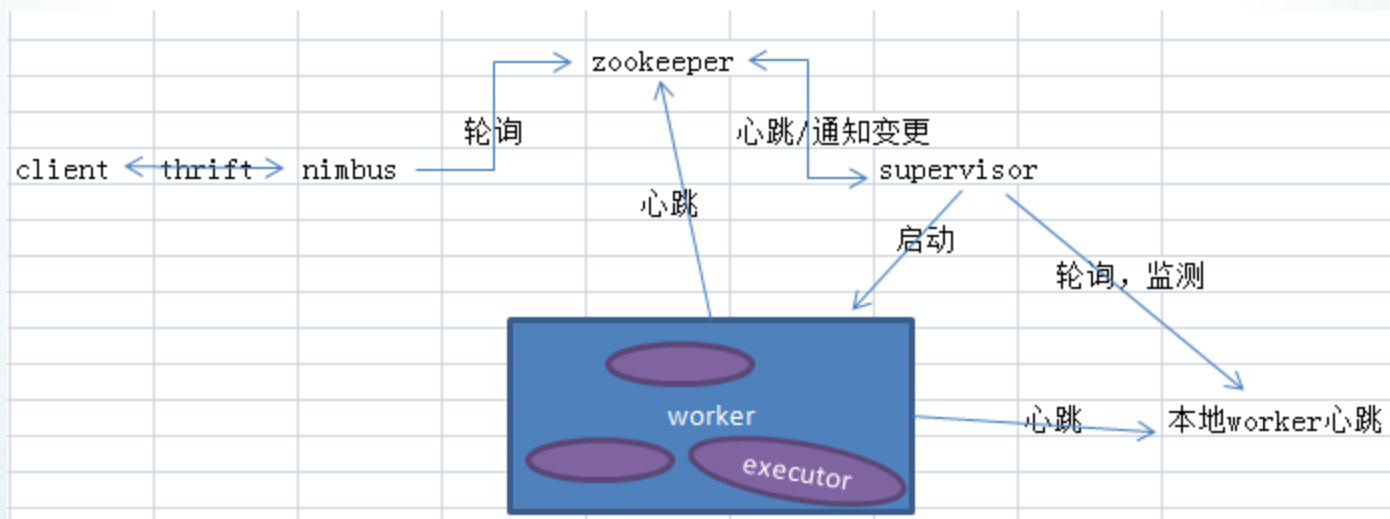
Storm 分享

邓志华
2015年6月

主要内容

- Storm 是如何维护集群的?
- Storm 是如何分配任务的?
- Storm 是如何分发消息的?
- Task 是如何收到消息?
- Ack机制
- Storm调优手段

整体结构



Nimbus-维护集群，分配任务

thrift 服务

zookeeper

java.util. PriorityQueue

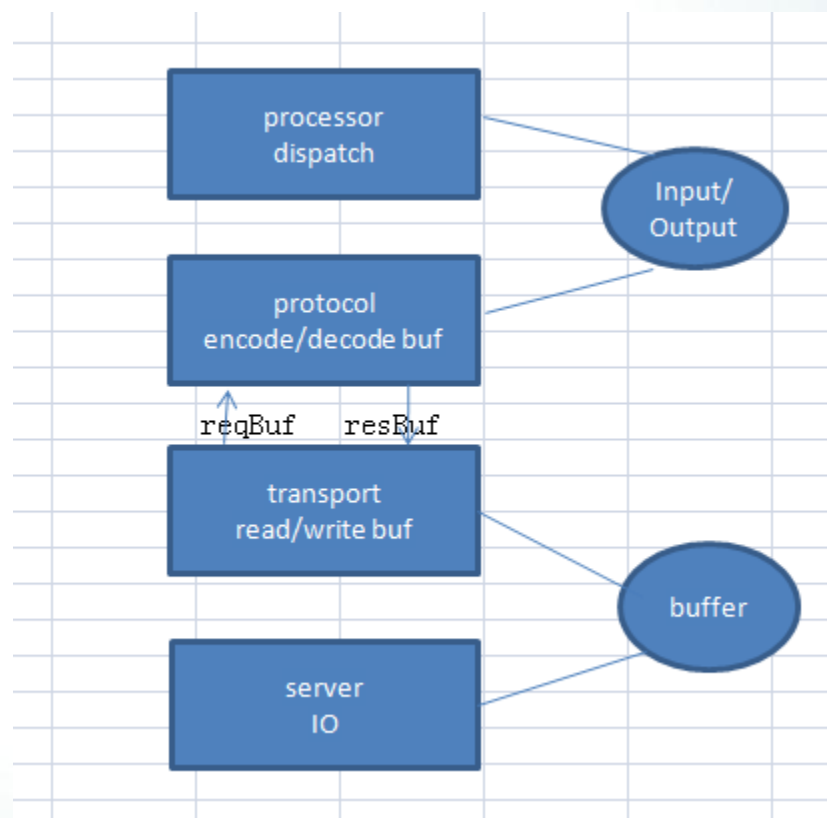
抽屉算法

Nimbus- thrift 服务

C/S架构

NIO

实现MVC中VC



namespace java com.kuxun.generated
service Hello{
 string welcome (1: string ame)
}
->
thrift --gen java

```
public class Hello {  
    public interface Iface {  
        public String welcome(String name) throws org.apache.thrift.TException;  
    }  
    public interface AsyncIface {  
        public void welcome(String name, org.apache.thrift.async.AsyncMethodCallback resultHandler) throws org.apache.thrift.TException;  
    }  
    public static class Client extends org.apache.thrift.TServiceClient implements Iface {  
        public static class Factory implements org.apache.thrift.TServiceClientFactory<Client> {  
            public Factory() {}  
            public Client getClient(org.apache.thrift.protocol.TProtocol prot) {  
                return new Client(prot);  
            }  
            public Client getClient(org.apache.thrift.protocol.TProtocol iprot, org.apache.thrift.protocol.TProtocol oprot) {  
                return new Client(iprot, oprot);  
            }  
        }  
    }  
    ...  
}  
...  
public static class Processor<I extends Iface> extends org.apache.thrift.TBaseProcessor<I> implements org.apache.thrift.TProcessor {  
    private static final Logger LOG = LoggerFactory.getLogger(Processor.class.getName());  
    public Processor(I iface) {  
        super(iface, getProcessMap(new HashMap<String, org.apache.thrift.ProcessFunction<I, ? extends org.apache.thrift.TBase>>()));  
    }  
    protected Processor(I iface, Map<String, org.apache.thrift.ProcessFunction<I, ? extends org.apache.thrift.TBase>> processMap) {  
        super(iface, getProcessMap(processMap));  
    }  
    private static <I extends Iface> Map<String, org.apache.thrift.ProcessFunction<I, ? extends org.apache.thrift.TBase>> getProcessMap() {  
        processMap.put("welcome", new welcome());  
        return processMap;  
    }  
}
```

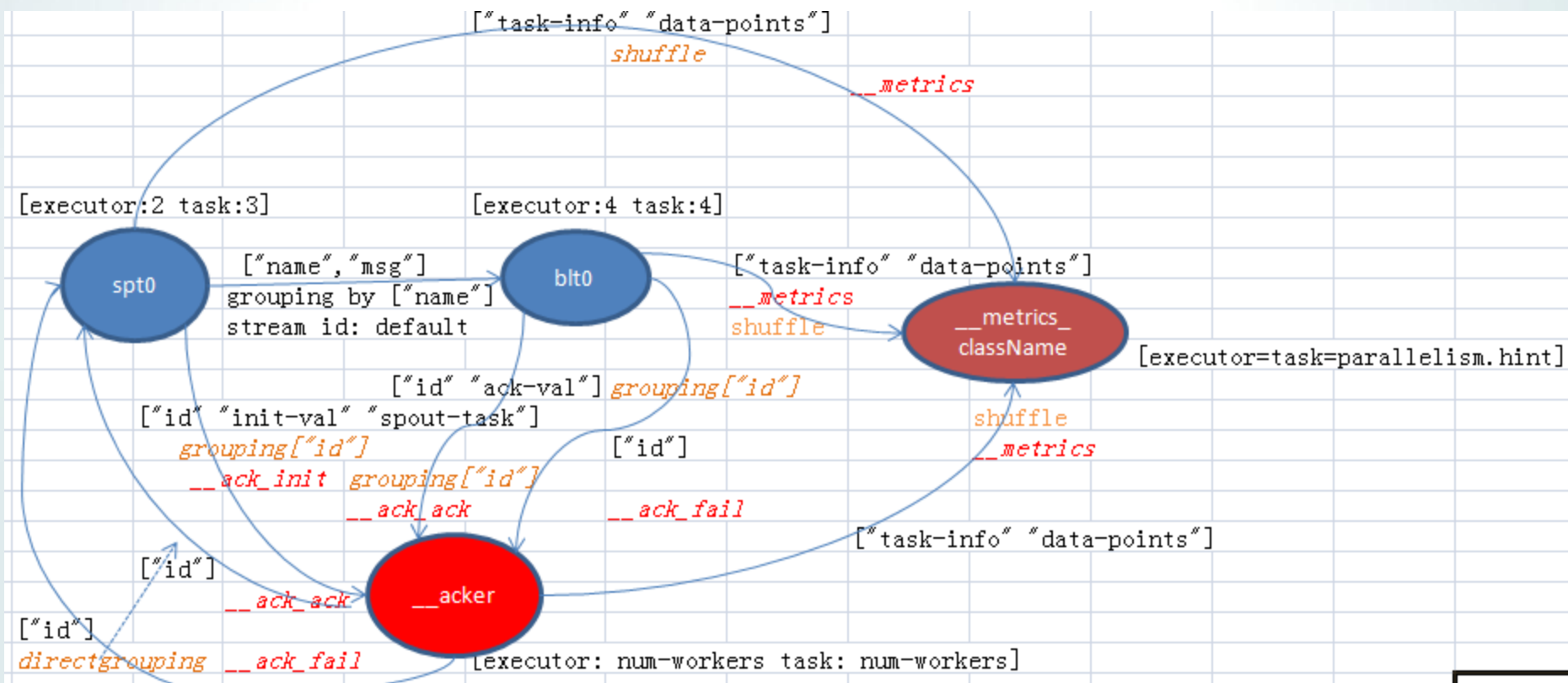
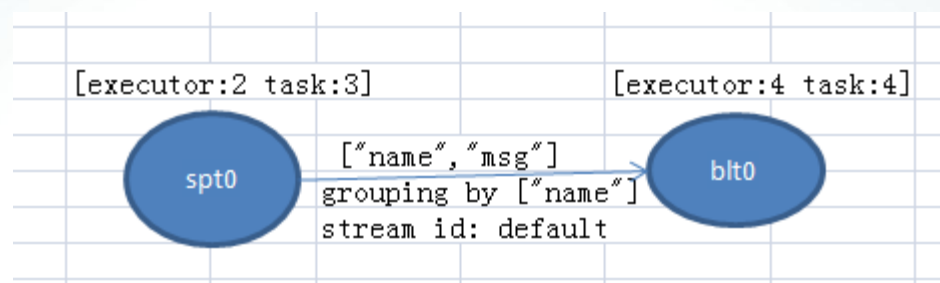
用户需要实现的业务逻辑接口

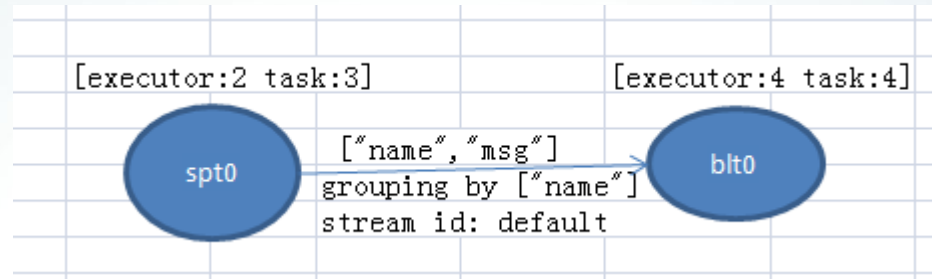
客户端

转发器

Nimbus-Zookeeper

```
-/  
--/supervisors/$supervisor-id  
    :used-ports->(6703 6704 6705 6706 ...)  
--/assignments/$storm-id  
    [193 193]->["73f3da46-85b3-4d7f-86a9-  
f94cf03872a7" 6713]  
--/storms/$storm-id  
--/workerbeats/$storm-id/$worker-id
```





{1 blt0 2 blt0 3 blt0 4 blt0 5 spt0 6 spt0 7 spt0}

blt0: {1 4} -> (1 1 1 1) -> (1) (2) (3) (4)
-> [1 1] [2 2] [3 3] [4 4]

spt0 : {1 1 2 1} -> (1 2) -> (2 1) -> (5 6) (7)
-> [5 6] [7 7]

Utils.integerDivided 方法实现

blt0: -> [1 1] [2 2] [3 3] [4 4]

spt0 : -> [5 6] [7 7]

设有三台supervisor机器(i, j, k), [i 6703] [i 6704]
[j 6705] [j 6711] [k 6700] 资源可用。

排序：

[i 6703] [j 6705] [k 6700] [i 6704] [j 6711]

take 2(假设topology使用2个worker):

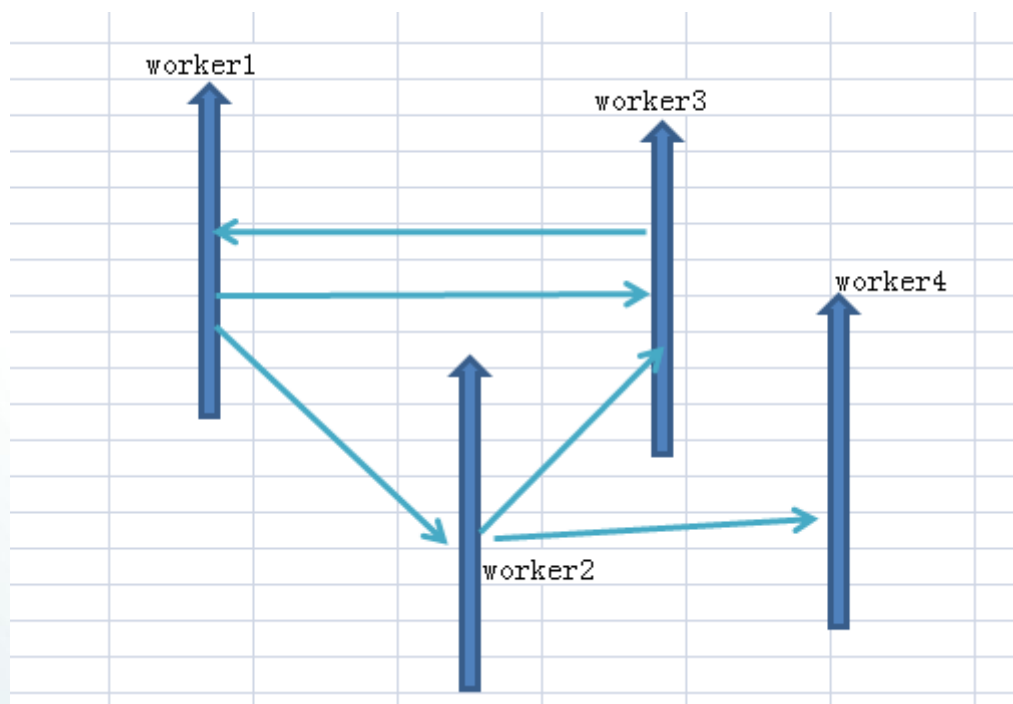
[i 6703] [j 6705]

按照executor的数量重复输出上一步结果形成集合I。

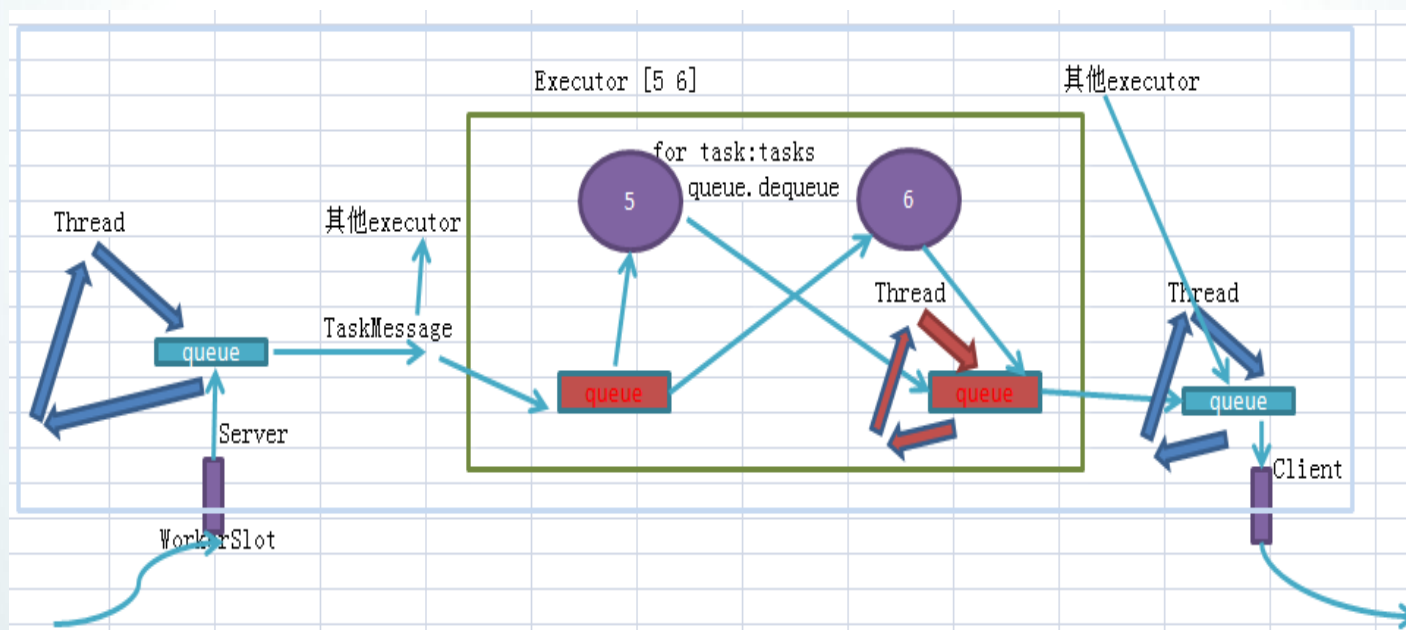
映射 $f(x \text{ in executors}) = g(x \text{ in } I)$

Worker互联

建立TCP长连接



Worker内部



```
private int _task;  
private byte[] _message;
```

北京酷讯科技有限公司

1. fieldGrouping

(-> (.select out-fields group-fields values)

tuple/list-hash-code

(mod num-tasks)

task-getter))))

(defn list-hash-code

[^List alist]

(.hashCode alist))

2. shuffleGrouping

Collections/shuffle(List targetTasks)

洗牌 取出List第一个

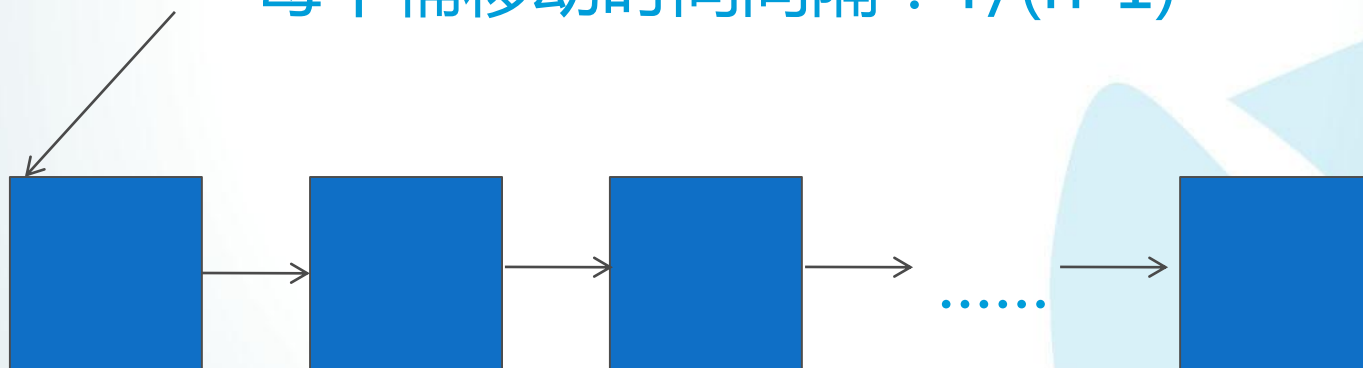
shuffle

```
int size = list.size();
if (size < SHUFFLE_THRESHOLD || list instanceof
RandomAccess) {
    for (int i=size; i>1; i--)
        swap(list, i-1, rnd.nextInt(i));
} else {
    Object arr[] = list.toArray();
    // Shuffle array
    for (int i=size; i>1; i--)
        swap(arr, i-1, rnd.nextInt(i));
```

TimeCacheMap/ RotatingMap

1. 删除尾部节点

每个桶移动时间间隔： $T/(n-1)$



$T \leq \text{对象存活时间} < T + T/(n-1)$

HashMap