淘宝网
Taobao.com

淘宝分布式框架
fourinone

彭渊 (千峰)

简单自我介绍

从业java技术领域十多年

现在淘宝网任高级专家，从事互联网核心技术研究,

之前在金蝶总体架构部任SOA架构师，负责设计ESB

创业生涯

……

分布式核心技术（背景）

分布式并行计算、分布式缓存、一致性、消息队列、分布式文件系统为大型互联网应用背后的核心技术，是从业互联网技术的工程师最关心和想掌握的，目前广泛应用于搜索、云计算、大数据等领域

Fourinone介绍

淘宝Fourinone2.0提供了一个4合1分布式框架和简单易用的编程api，实现对多台计算机cpu，内存，硬盘的统一利用，从而获取到强大计算能力去解决复杂问题。

1、提供了一系列并行计算模式（农民工/包工头/职介绍/手工仓库）用于利用多机多核cpu的计算能力；

2、提供完整的分布式缓存和小型缓存用于利用多机内存能力；

3、提供像操作本地文件一样操作远程文件（访问，并行读写，拆分，排它，复制，解析，事务等）用于利用多机硬盘存储能力；

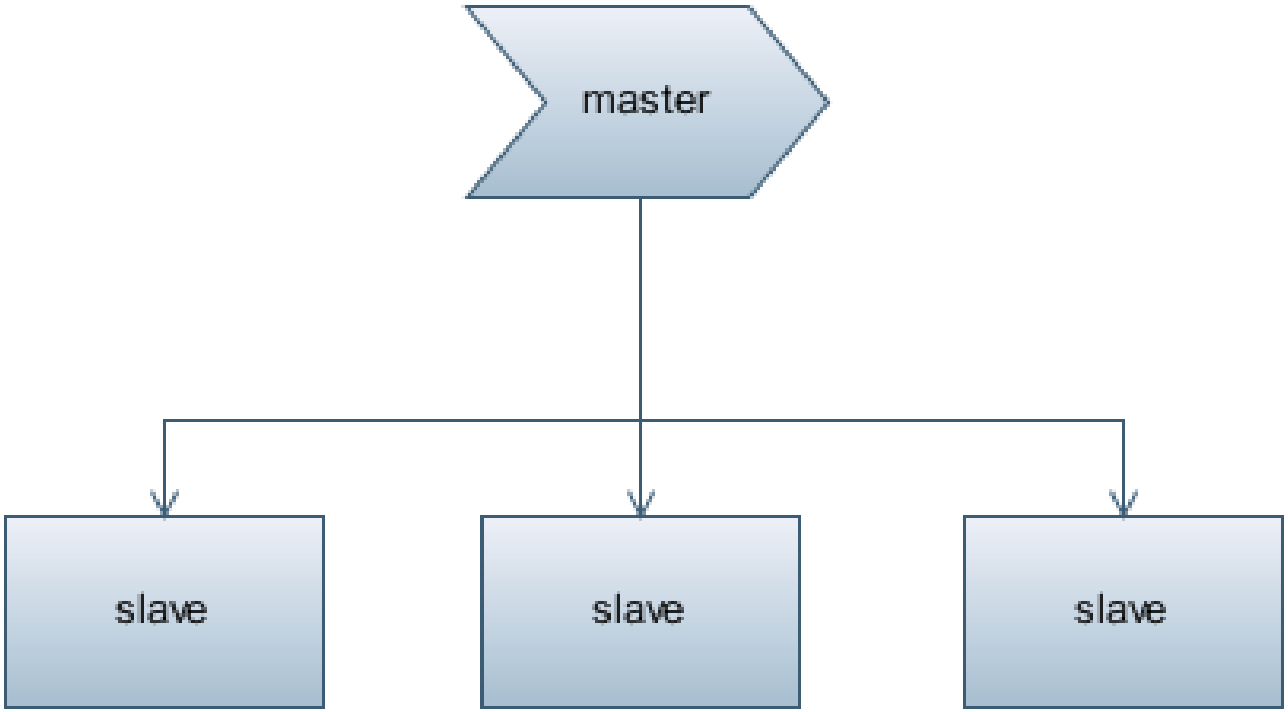
4、由于多计算机物理上独立，Fourinone框架也提供完整的分布式协同和锁以及简化MQ功能，用于实现多机的协作和通讯。

Fourinone采用java开发，2.0版本整体大小150k，就一个jar和一个配置文件，没有任何依赖。

提纲

- 分布式并行计算
- 分布式协调
- 分布式缓存
- 消息队列
- FTTP分布式文件操作
- 分布式作业调度平台
- 应用场景:上亿数据排序

Fourinone分布式计算

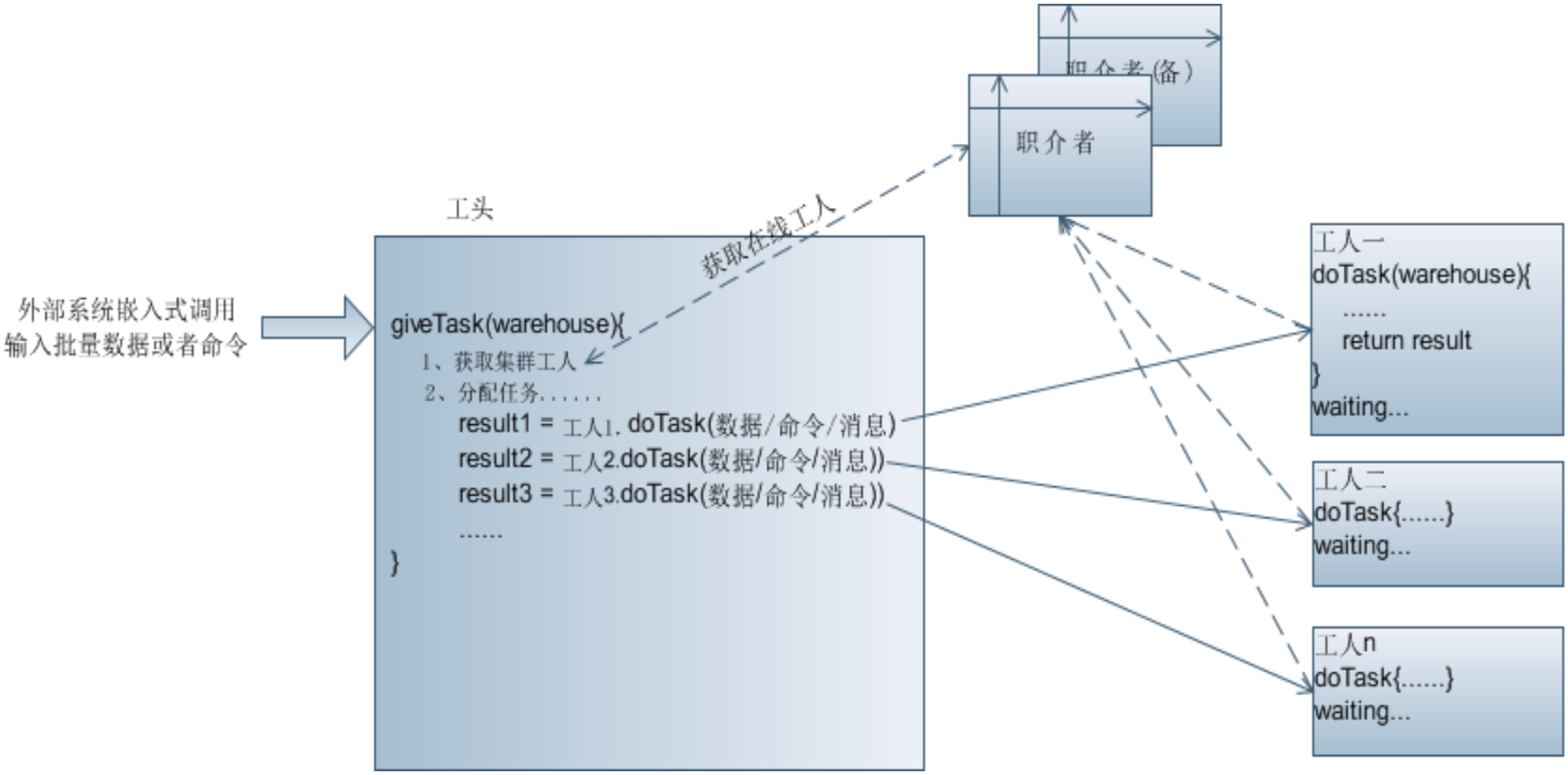


最简单的master-slave计算结构

master是一个服务程序, slave跟master耦合太紧

master除分配任务外需要负责协同一致性等处理

Fourinone分布式计算



fourinone的简化分布式并行计算结构

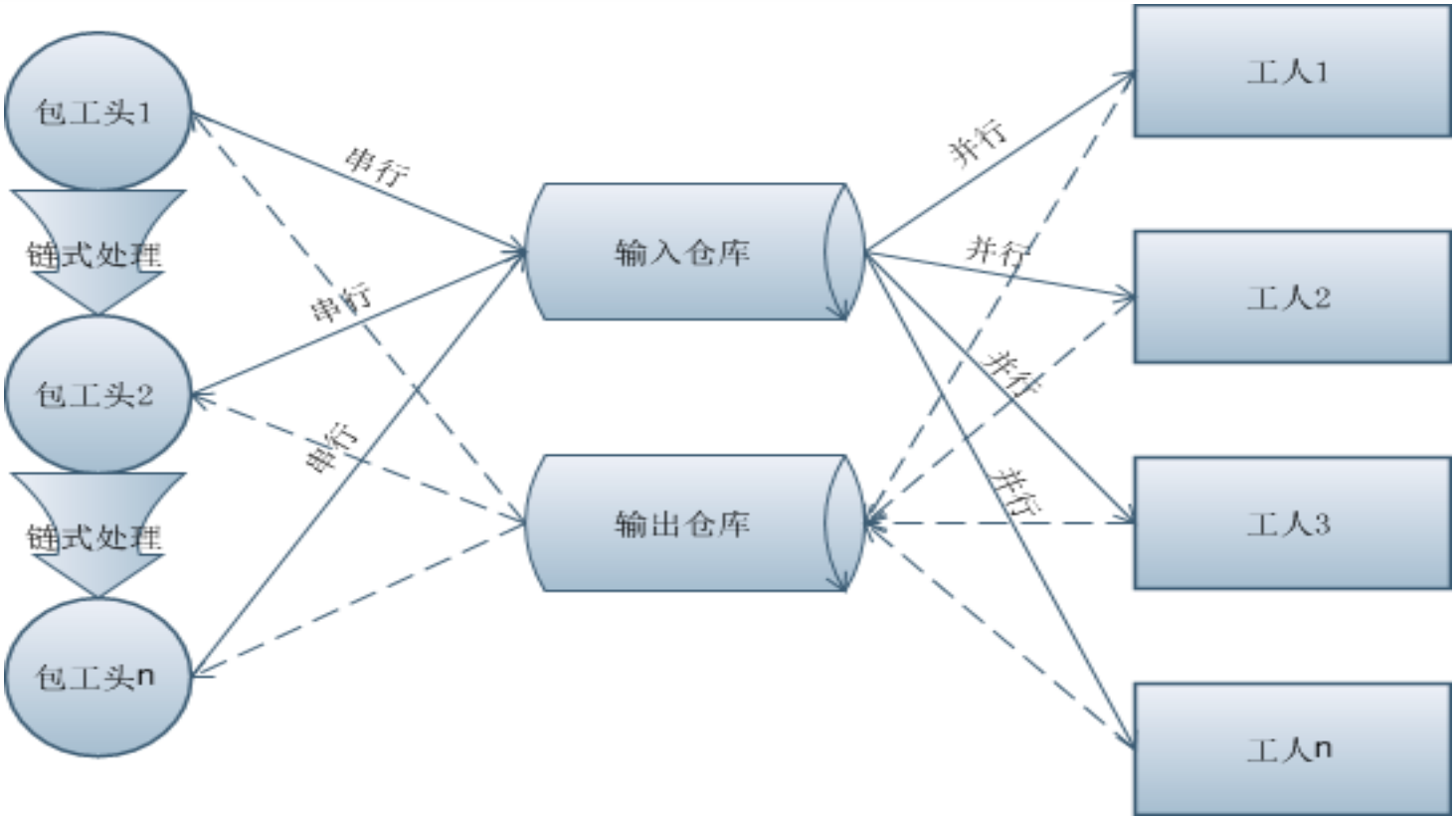
包工头去服务化，嵌入式，负责分配任务，开发者实现分配任务接口

农民工负责执行任务，开发者实现任务执行接口

职介者负责协同一致性等处理（登记，介绍，保持联系）

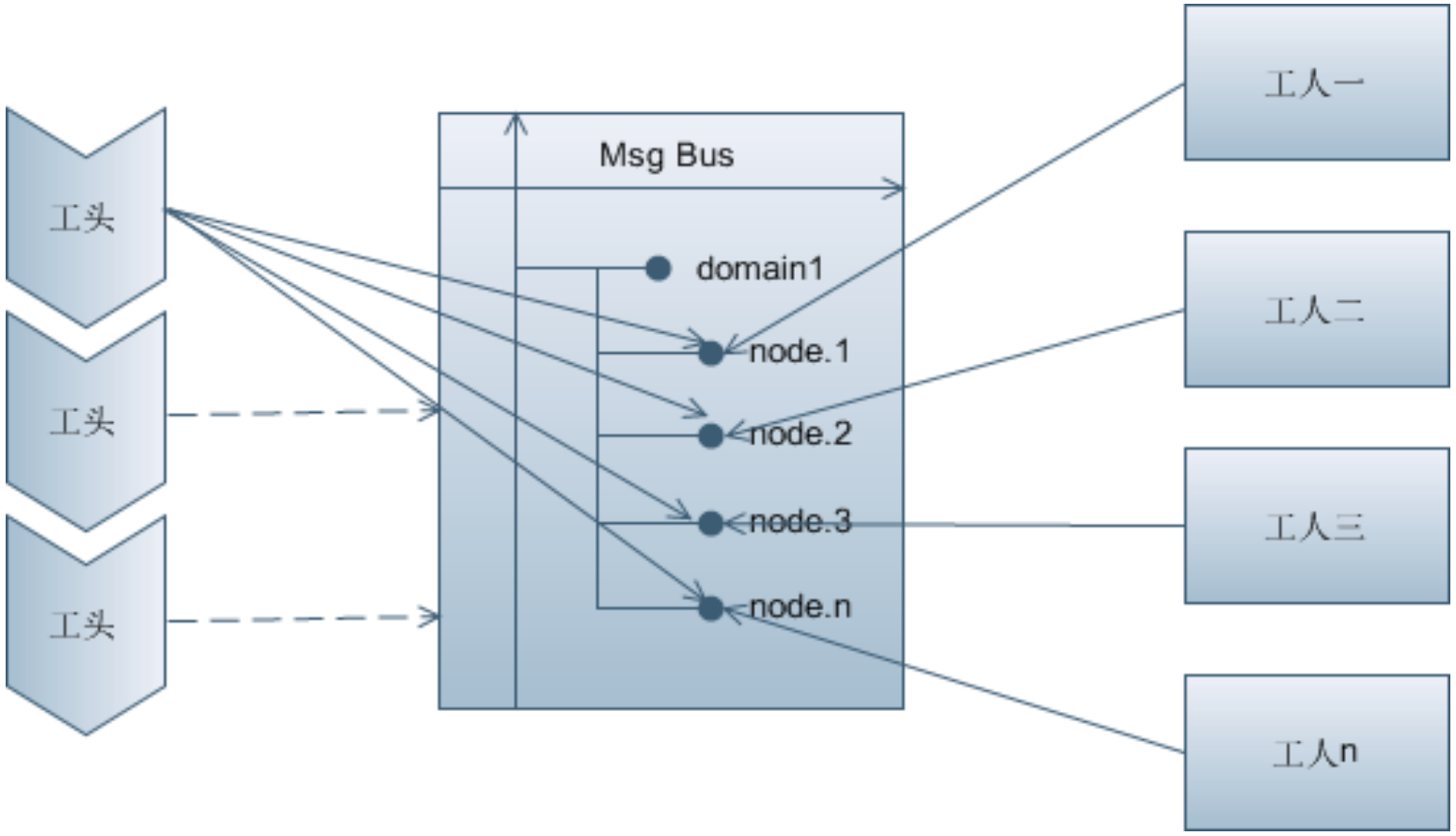
思考：是否能满足storm这样的实时流计算模型？

Fourinone分布式计算



总的来说，是将大数据的复杂分布式计算，设计为一个链式的多“包工头”环节去处理，每个环节包括利用多台“农民工”机器进行并行计算，无论是拆分计算任务还是合并结果，都可以设计为一个单独的“包工头”环节。这样做的好处是，开发者有更大能力去深入控制并行计算的过程，去保持使用并行计算实现业务逻辑的完整性，而且对各种不同类型的并行计算场景也能灵活处理，不会因为某些特殊场景被map/reduce的框架限制住思维，并且链式的每个环节也方便进行监控过程。

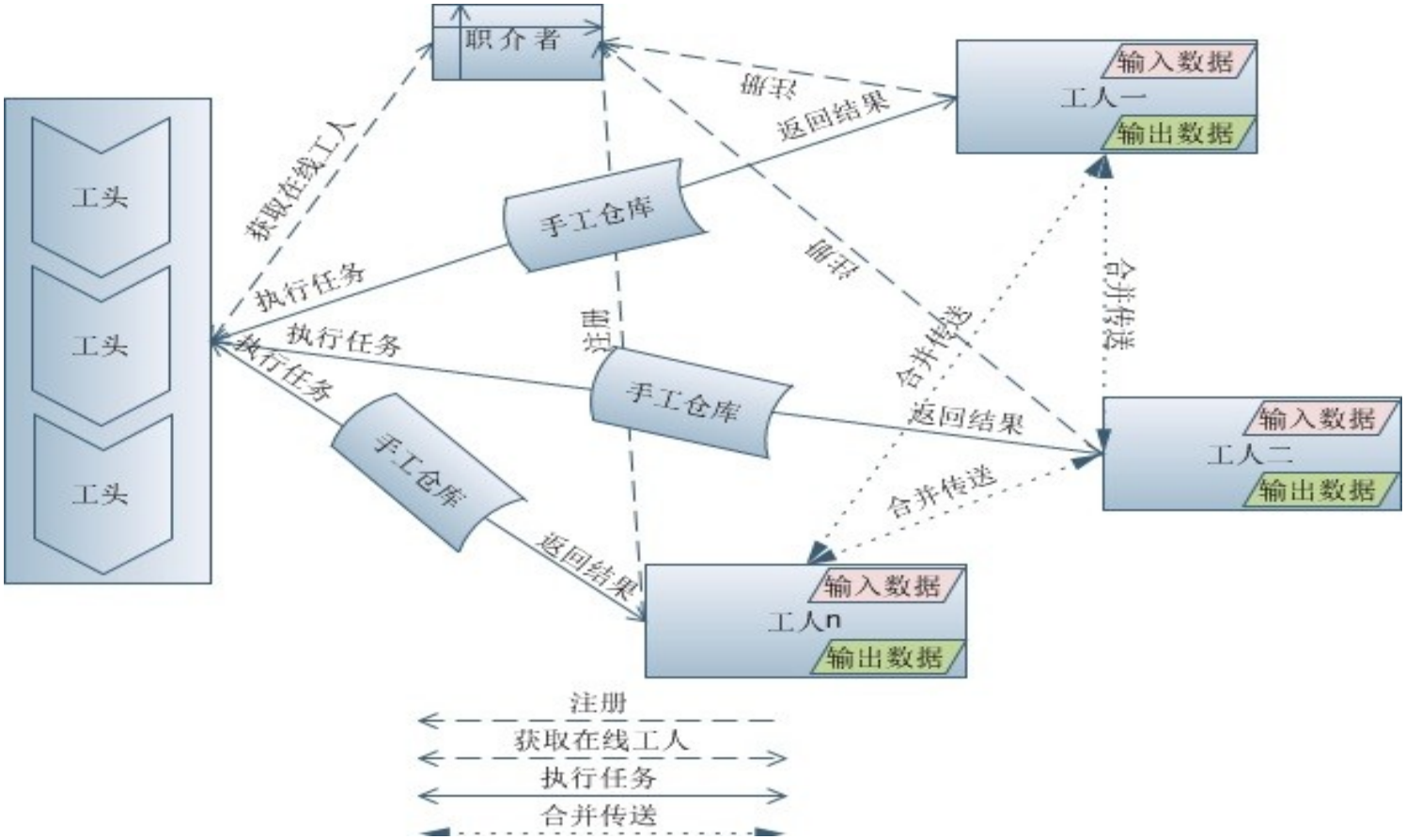
Fourinone分布式计算



- 模式一：基于消息中枢的计算模式

  优势？缺点？能否满足mpi的send/recv模式和迭代计算

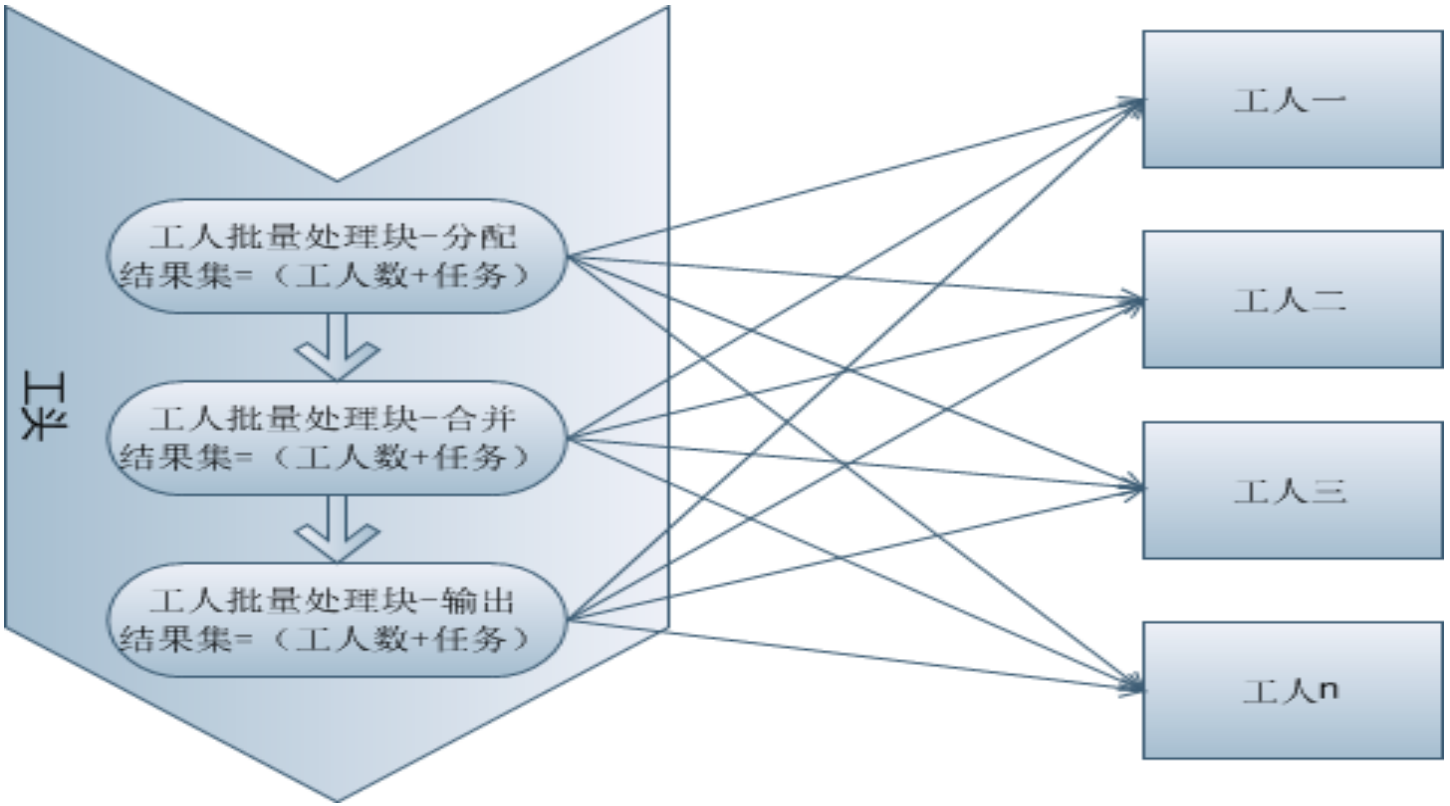
Fourinone分布式计算



- 模式二：基于工人服务的网状交互计算模式

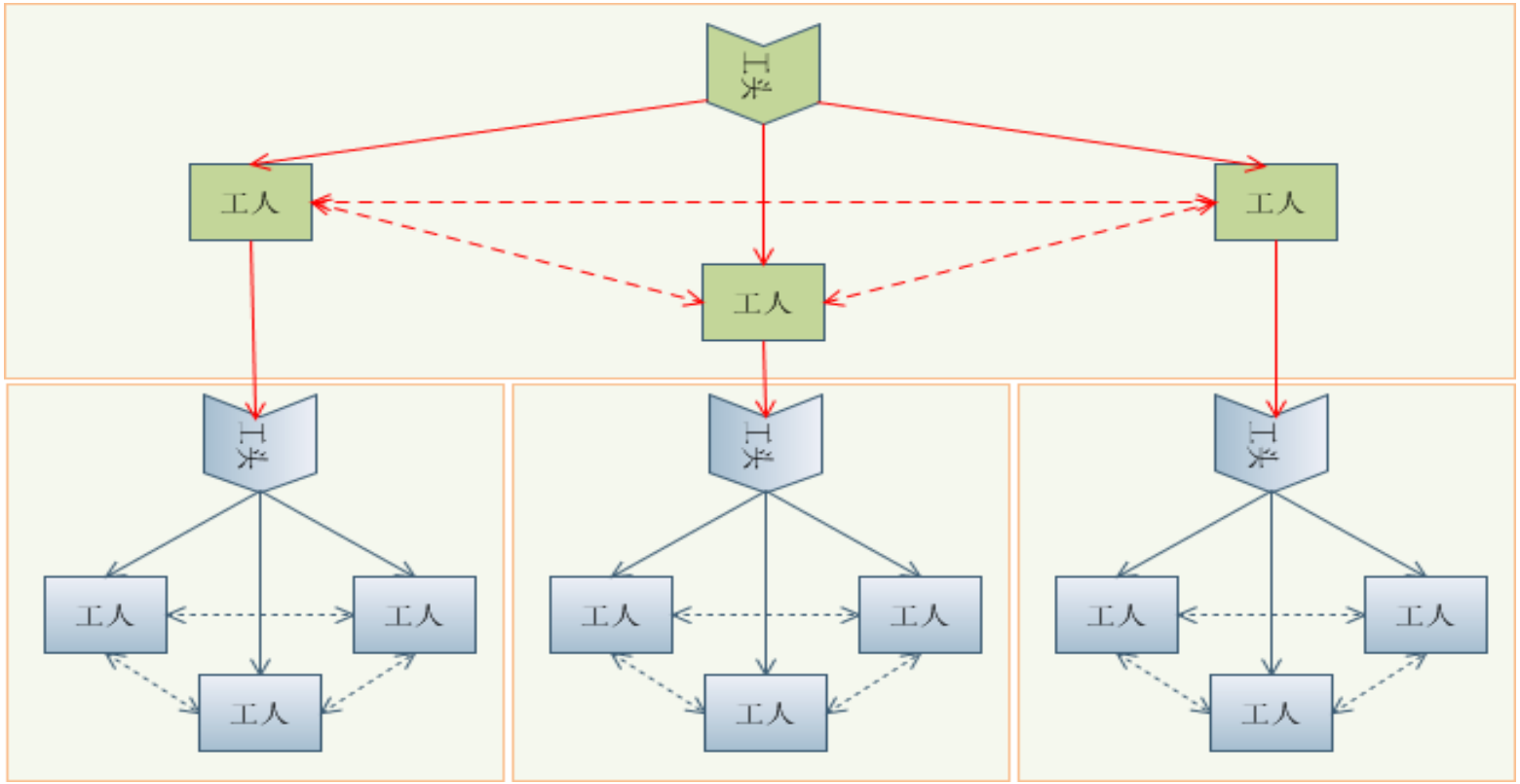
  优势？缺点？

Fourinone分布式计算



- 单个工头支持多阶段工人批量任务处理
- 思维发散：多工头的任务并行拆分

Fourinone分布式计算



- 多工头并行的计算集群搭建（兼顾遗留计算系统）
- 模仿现实中加工生产原材料承包分配

思考:能否满足DAG（有向无环图）并行作业流

Fourinone和hadoop的对比

	fourinone-1.11.09	hadoop-0.21.0
体积	82K	71M
依赖关系	就一个jar,没有依赖	约12项jar包依赖
配置	就一个配置文件	较多配置文件和复杂属性
集群搭建	简单，每台机器放一个jar和配置文件	复杂，需要linux操作基础和ssh等复杂配置，还需要较多配置文件配置
计算模式	提供两种计算模式：包工头和工人直接交互方式，包工头和工人通过消息中枢方式交互，后者不需要工人节点可直接访问	计算更多倾向于文件数据的并行读取，而非计算过程的设计。JobTracke 跟 TaskTracker 直接交互， 查询 NameNode 后，TaskTracker直接从Datanode获取数据。
并行模式	N*N，支持单机并行，也支持多机并行，多机多实例并行	1*N，不支持单机并行，只支持多机单实例并行
内存方式	支持内存方式设计和开发应用，并内置完整的分布式缓存功能	以hdfs文件方式进行数据处理，内存方式计算支持很弱
文件方式	自带文件适配器处理io	Hdfs处理文件io
计算数据要求	任意数据格式和任意数据来源，包括来自数据库，分布式文件，分布式缓存等	Hdfs内的文件数据，多倾向于带换行符的数据
调度角色	包工头，可以有多个，支持链式处理，也支持大包工头对小包工头的调度	JobTracke，通常与NameNode一起
任务执行角色	农民工，框架支持设计多种类型的工人用于拆分或者合并任务	TaskTracker，通常与Datanode一起
中间结果数据保存	手工仓库，或者其他任意数据库存储设备	Hdfs中间结果文件
拆分策略	自由设计，框架提供链式处理对于大的业务场景进行环节拆分数据的存储和计算拆分根据业务场景自定义	以64m为拆分进行存储，以行为拆分进行计算 实现map接口，按行处理数据进行计算
合并策略	自由设计，框架提供农民工节点之间的合并接口，可以互相交互设计合并策略，也可以通过包工头进行合并	TaskTracker不透明，较少提供程序控制，合并策略设计复杂 实现reduce接口进行中间数据合并逻辑实现
内存耗用	无需要制定JVM内存，按默认即可，根据计算要求考虑是否增加JVM内存	需要制定JVM内存，每个进程默认1G，常常namenode，jobtracker等启动3个进程，耗用3G内存
监控	框架提供多环节链式处理设计支持监控过程，通过可编程的监控方式，给于业务开发方最大灵活的监控需求实现，为追求高性能不输出大量系统监控log	输出较多的系统监控log，如map和reduce百分比等，但是会牺牲性能，业务监控需要自己实现
打包部署	脚本工具	上传jar包到jobtracker机器
平台支撑	支持跨平台，windows支持良好	多倾向于支持linux，Windows支持不佳，需要模拟linux环境，并且建议只用于开发学习
其他	协同一致性、分布式缓存、通讯队列等跟分布式计算关系密切的功能支持	不支持
总结：	Hadoop并不是为了追求一个并行计算的框架而设计，提供快捷和灵活的计算方式去服务各种计算场景， 它更多的是一个分布式文件系统，提供文件数据的存储和查询，它的map/reduce更倾向于提供并行计算方式进行文件数据查询。而fourinone相反。	

Fourinone和hadoop的对比

Fourinone 和 hadoop 运行 wordcount 的对比测试（平均 4 核 4g 配置，输入数据为文件）：

	fourinone-1.11.09(n*4)	fourinone-1.11.09(n*1)	hadoop-0.21.0(n*1)
3 台机器*256M	4s	12s	72s
3 台机器*512M	7s	30s	140s
3 台机器*1G	14s	50s	279s
19 台机器*1G	21s	60s	289s
10 台机器*2G	29s		
5 台机器*4G	60s		

N*4 说明:Fourinone 可以充分利用单机并行能力,4 核计算机可以 4 个并行实例计算,hadoop 目前只能 N*1；另外，可以由上图看出，如果要完成 20g 的数据，实际上 fourinone 只需要使用 5 台机器用 60 秒完成，比使用 19 台机器完成 19g 的 hadoop 节省了 14 台机器，并提前了 200 多秒

提纲

- 分布式并行计算
- 分布式协调
- 分布式缓存
- 消息队列
- FTTP分布式文件操作
- 分布式作业调度平台
- 应用场景:上亿数据排序

Fourinone分布式协同

分布式协同方面，fourinone实现了Zookeeper所有的功能，并且做了很多改进：

1、简化Zookeeper的树型结构，用domain/node两层结构取代

2、简化Watch回调多线程等待编程模型，用更直观的容易保证业务逻辑完整性的内容变化事件以及状态轮循取代

3、Zookeeper只能存储信息不大于1M的内存内容，fourinone提供了内存管理控制，针对jvm的默认内存和调优内存等情况都能进行内存占用报警异常，避免内存溢出。

4、简化了Zookeeper的ACL权限功能，用更为程序员熟悉rw风格取代

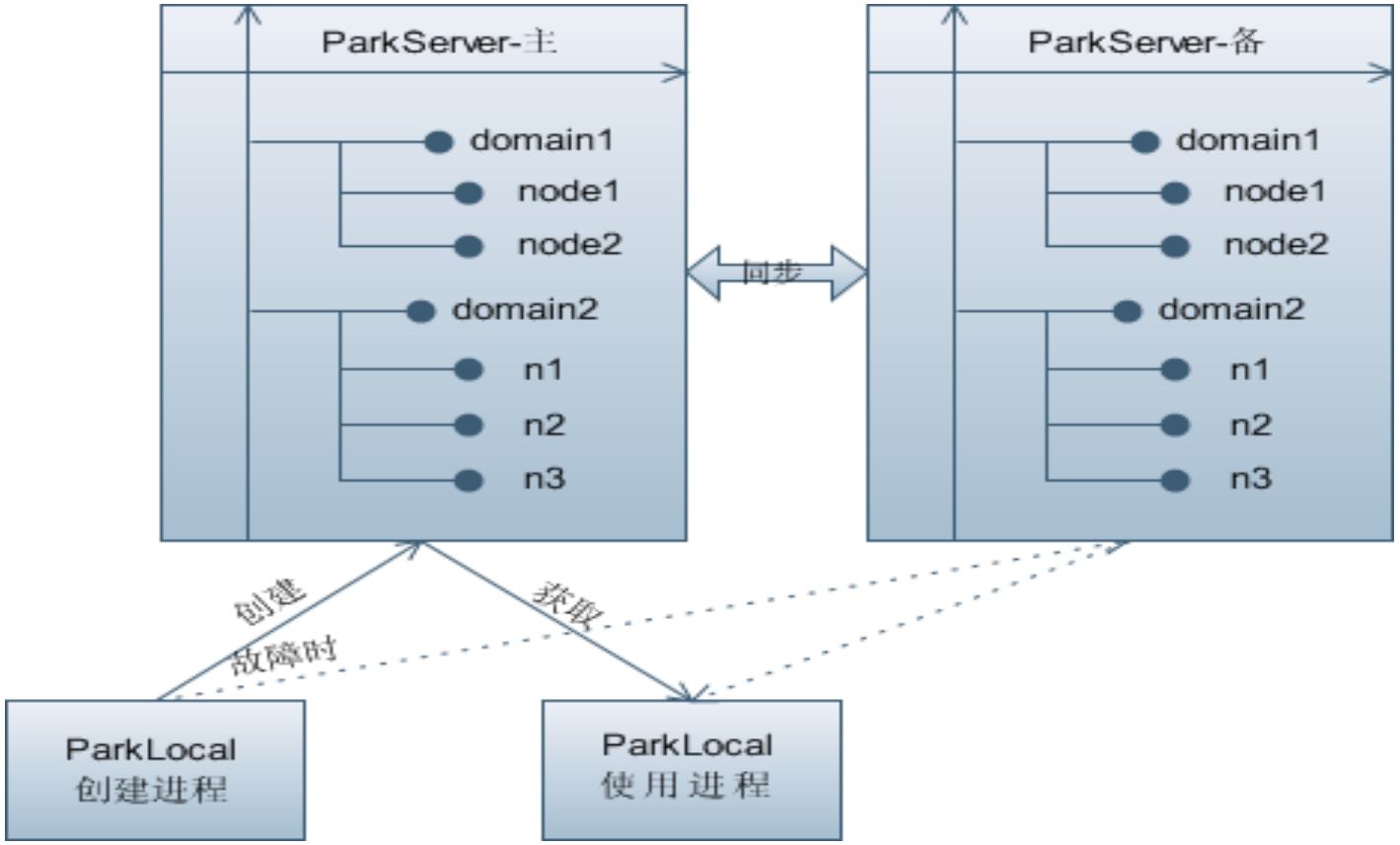
5、简化了Zookeeper的临时节点和序列节点等类型，取代为在创建节点时是否指定保持心跳，心跳断掉时节点会自动删除。

6、FourInOne是高可用的，没有单点问题，可以有任意多个复本，它的复制不是定时而是基于内容变更复制，有更高的性能

7、FourInOne实现了领导者选举算法（但不是Paxos），在领导者服务器宕机情况下，会自动不延时的将请求切换到备份服务器上，选举出新的领导者进行服务，这个过程中，心跳节点仍然能保持健壮的稳定性，迅速跟新的领导者保持心跳连接。

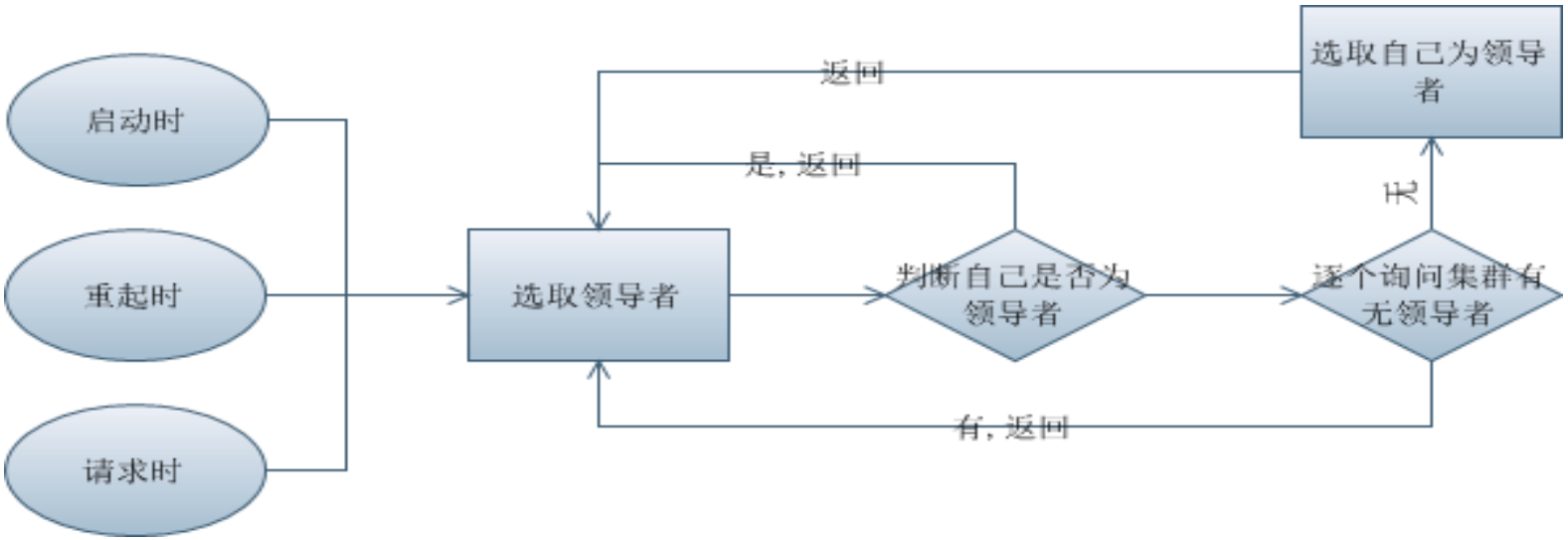
基于FourInOne可以轻松实现分布式配置信息，集群管理，故障节点检测，分布式锁，以及淘宝configserver等等协同功能。

Fourinone分布式协同



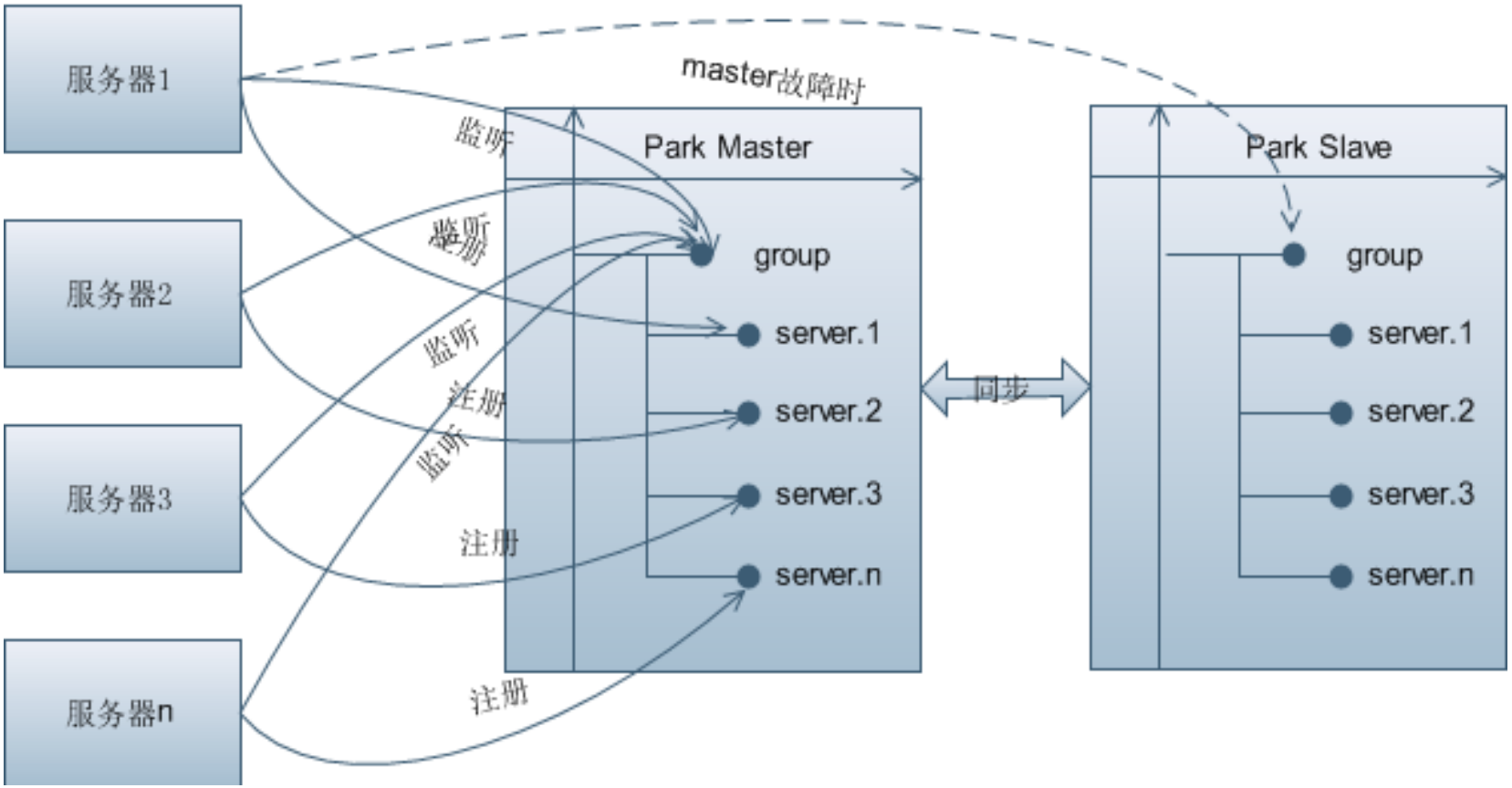
从上图可以看到，fourinone对分布式协同的实现， 是通过建立一个domain,node两层结构的节点信息去完成，分布式进程可以通过parkserver的用户接口ParkLocal，对节点进行增加、修改、删除、指定心跳、指定权限等操作，并且结合parkserver提供同步备份、领导者选举、过期时间设置等功能，共同来实现众多分布式协同功能。

Fourinone分布式协同



领导者选举：ZooKeeper的领导者选举实现虽然比原始的Paxos要简化，但是它仍然存在领导者（Leader)、跟随者（Follower）、观察者（observer）、学习者(Learner）等众多角色和跟随状态（Following)、寻找状态（Looking）、观察状态（Observing）、领导状态 (Leading）等复杂状态。fourinone的集群领导者算法，只存在领导者和候选者两种角色，同一时刻只有一个领导者处于领导状态，其余处于候选状态，对领导者选举算法进一步简化，能够更快捷的实现。

Fourinone分布式协同



我们需要一个集群管理者管理集群里的服务器，同一个集群中任何一台服务器宕机,其他服务器都能感知. 如果是集群管理者宕机，集群中所有的服务器不能受任何影响，能实时切换到备份管理者上被提供服务。

Fourinone分布式协同

fourinone对比zookeeper的优势：

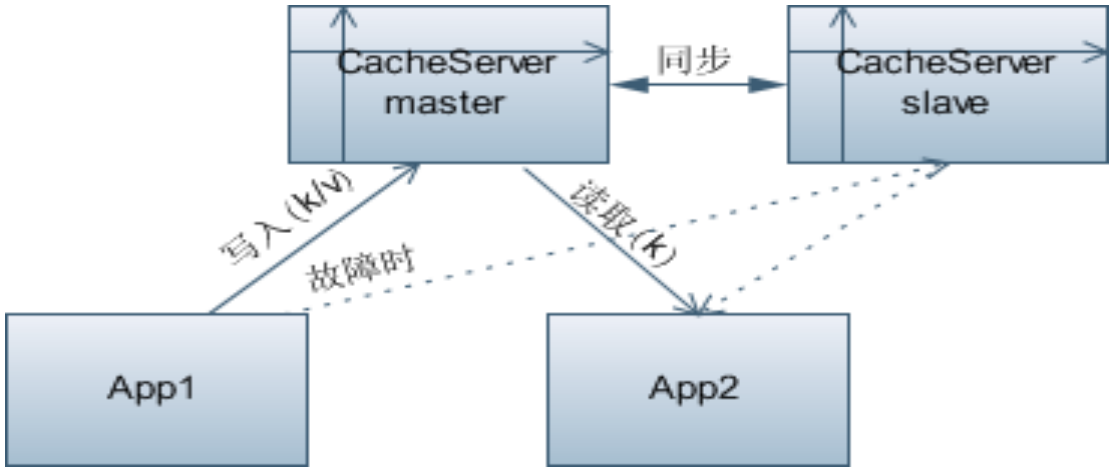
1、zookeeper没有获取最新版本信息的方法支持，它只能粗暴的在每次写入更新等方法时注册一个watch，当这些方法被调用后就回调，它不考虑信息内容是否变化，对于没有使信息内容发生改变的更新，zookeeper仍然会回调，并且zookeeper的回调比较呆板，它只能用一次，如果信息持续变化，必须又重新注册watch, 而fourinone的事件处理则可以自由控制是否持续响应信息变化。

2、领导者选举机制实现的太过局限，集群只有两个节点时，zookeeper无法进行领导者选举，zookeeper的领导者选举必须要奇数节点的奇怪限制。另外，ZooKeeper的领导者选举实现虽然比原始的Paxos要简化，但是仍然不够直观简洁，难以用较少配置和代码演示。

提纲

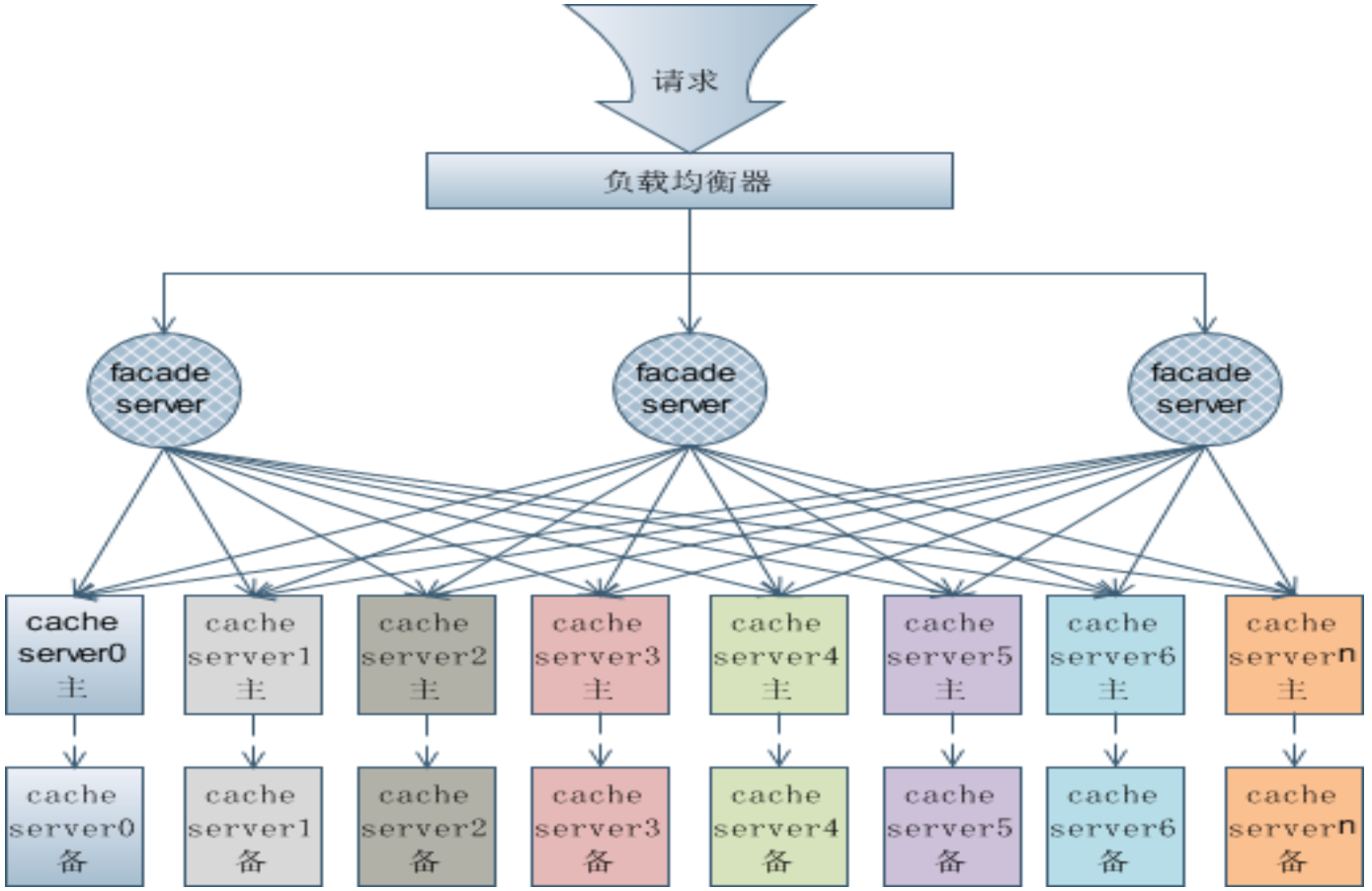
- 分布式并行计算
- 分布式协调
- 分布式缓存
- 消息队列
- FTTP分布式文件操作
- 分布式作业调度平台
- 应用场景:上亿数据排序

Fourinone分布式缓存



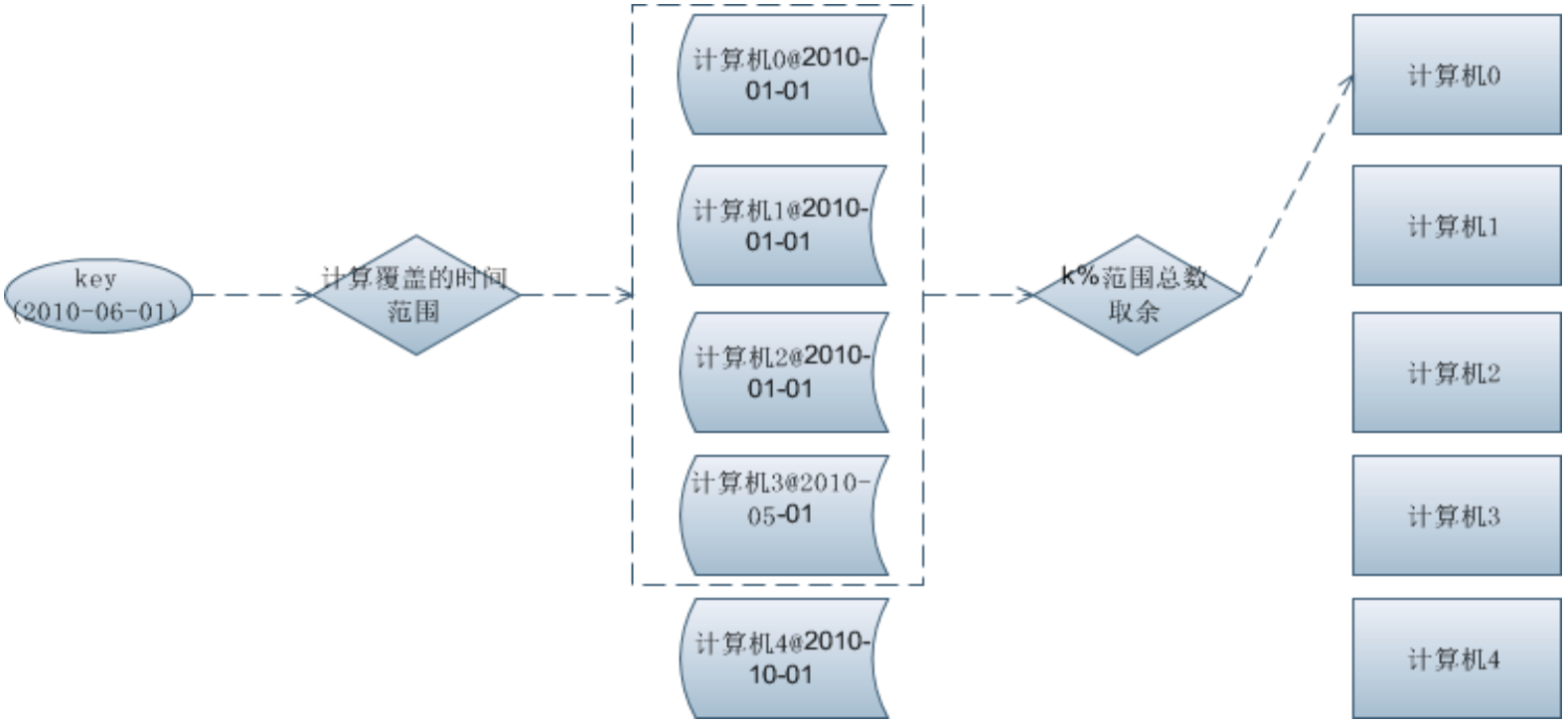
如果对一个中小型的互联网或者企业应用，仅仅利用domain/node进行k/v的存储即可，因为domain/node都是内存操作而且读写锁分离，同时拥有复制备份，完全满足缓存的高性能与可靠性。对于大型互联网应用，高峰访问量上百万的并发读写吞吐量，会超出单台服务器的承受力

Fourinone分布式缓存



Fourinone提供了facade的解决方案去解决大集群的分布式缓存，利用硬件负载均衡路由到一组facade服务器上，facade可以自动为缓存内容生成key，并根据key准确找到散落在背后的缓存集群的具体哪台服务器，当缓存服务器的容量到达限制时，可以自由扩容，不需要成倍扩容，因为facade的算法会登记服务器扩容时间版本，并将key智能的跟这个时间匹配，这样在扩容后还能准确找到之前分配到的服务器。基于Fourinone可以轻松实现web应用的session功能，只需要将生成的key写入客户端cookie即可。

Key取模设计

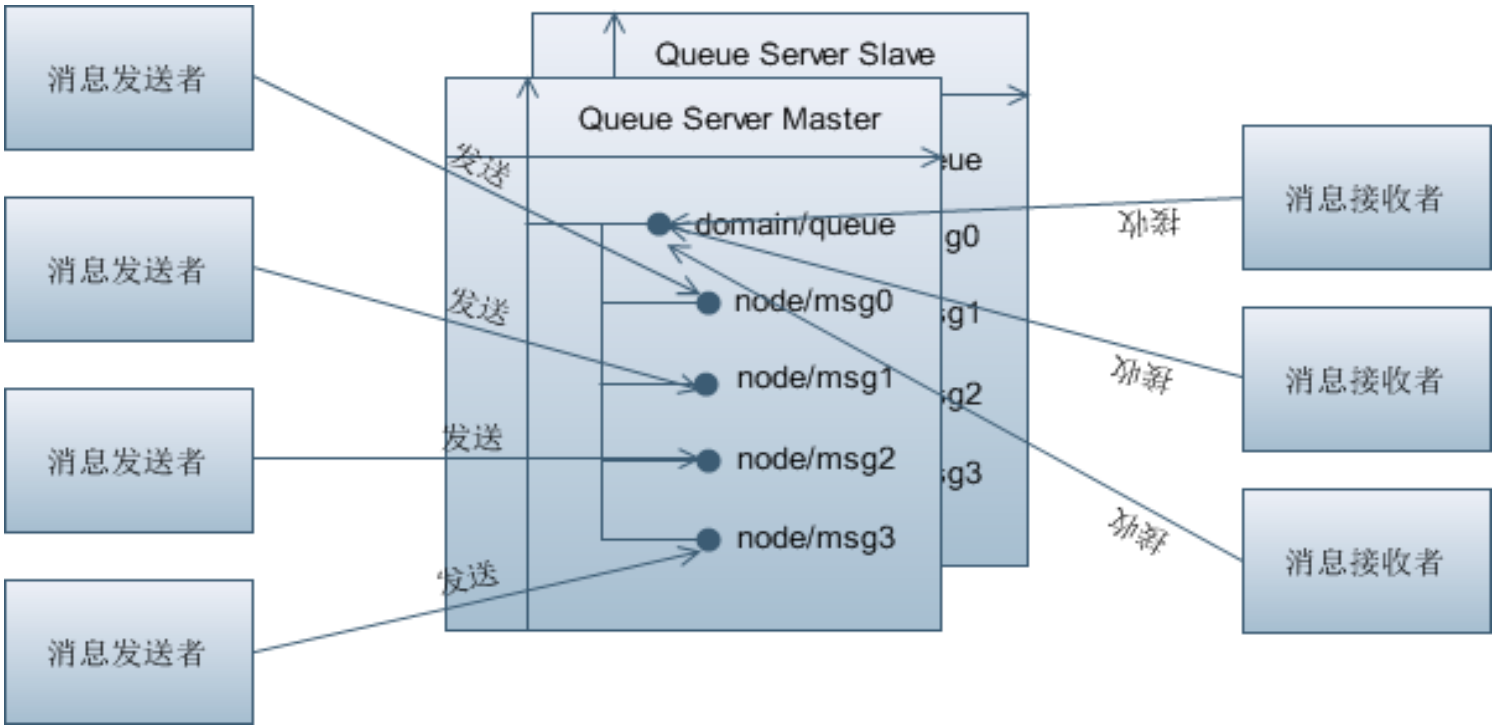


传统key取模这种方式有很大的缺陷，当集群数量扩充时，取模变的不准确，如果要维持准确，通常成倍方式去扩容，会造成成本增加和浪费。本发明通过生成含有日期信息的key，并对集群扩容增加日期配置，通过key和集群配置的日期匹配计算出覆盖范围的机器数，再取模的方式准确得到负载的计算机，对于集群的任意数量的扩容都不会受到影响。

提纲

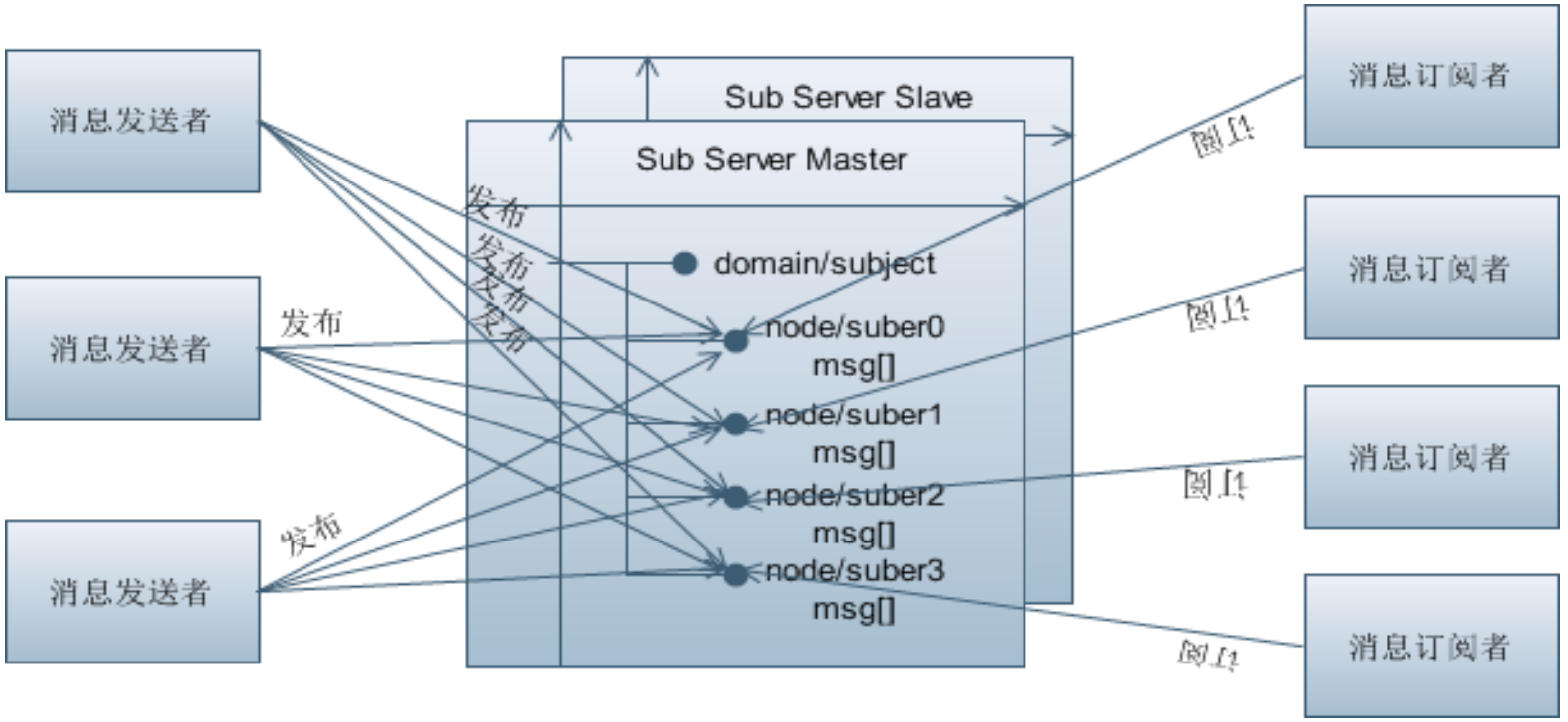
- 分布式并行计算
- 分布式协调
- 分布式缓存
- 消息队列
- FTTP分布式文件操作
- 分布式作业调度平台
- 应用场景:上亿数据排序

MQ发送接收模式



Fourinone也可以当成简单的mq来使用，发送接收模式实现: 将domain视为mq队列，每个node为一个队列消息，监控domain的变化事件来获取队列消息。

MQ主题订阅模式



将domain视为订阅主题，将每个订阅者注册到domain的node上，发布者将消息逐一更新每个node，订阅者监控每个属于自己的node的变化事件获取订阅消息，收到后清空内容等待下一个消息，多个消息用一个arraylist存放

FourInOne不实现JMS的规范，不提供JMS的消息确认和消息过滤等特殊功能，不过开发者可以基于FourInOne自己去扩充这些功能，包括mq集群。如果需要事务处理可以将多个消息封装在一个集合内进行发送，上面的队列接收者收到消息后删除实际上是一种消息确认方式，也可以将业务逻辑处理完后再进行删除。如果需要持久保存消息可以再封装一层消息发送者，发送前后根据需要进行数据库或者文件持久保存。利用一个独立的domain/node建立队列或者主题的key隐射，再仿照上面分布式缓存的智能根据key定位服务器的做法实现集群管理。

提纲

- 分布式并行计算
- 分布式协调
- 分布式缓存
- 消息队列
- FTTP分布式文件操作
- 分布式作业调度平台
- 应用场景:上亿数据排序

FTTP分布式文件操作

把集群当成一个操作系统，像操作本地文件一样操作远程文件

将集群中所有机器的硬盘资源利用起来，通过统一的fttp文件路径直接访问远程文件，如:

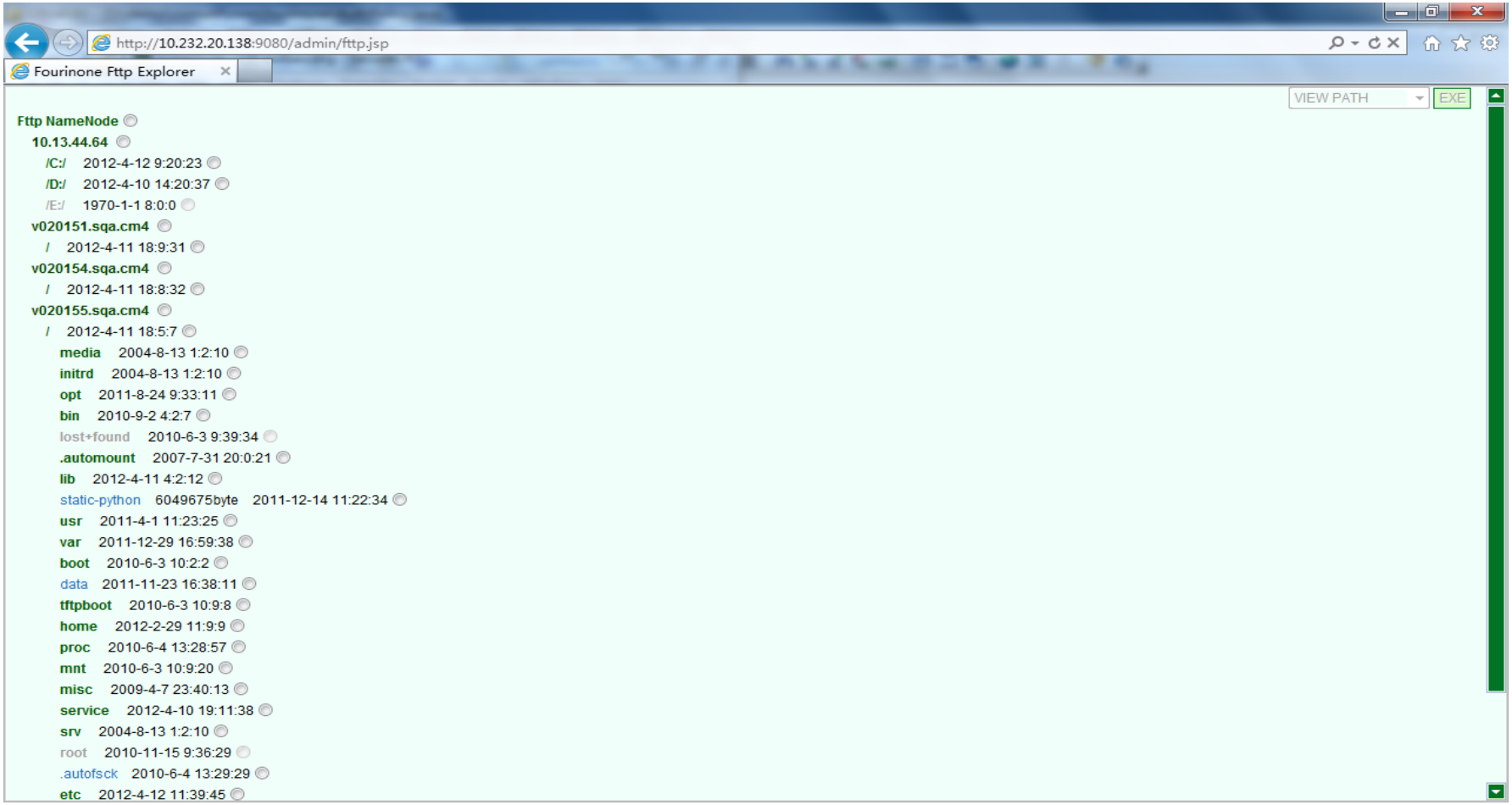
windows：fttp://192.168.0.1/d:/data/a.log

linux：fttp://192.168.0.1/home/user/a.log

以这样的方式读取远程文件：

```
FttpAdapter fa = new
FttpAdapter( “fttp://192.168.0.1/home/log/a.log");
fa.getFttpReader().readAll();
```

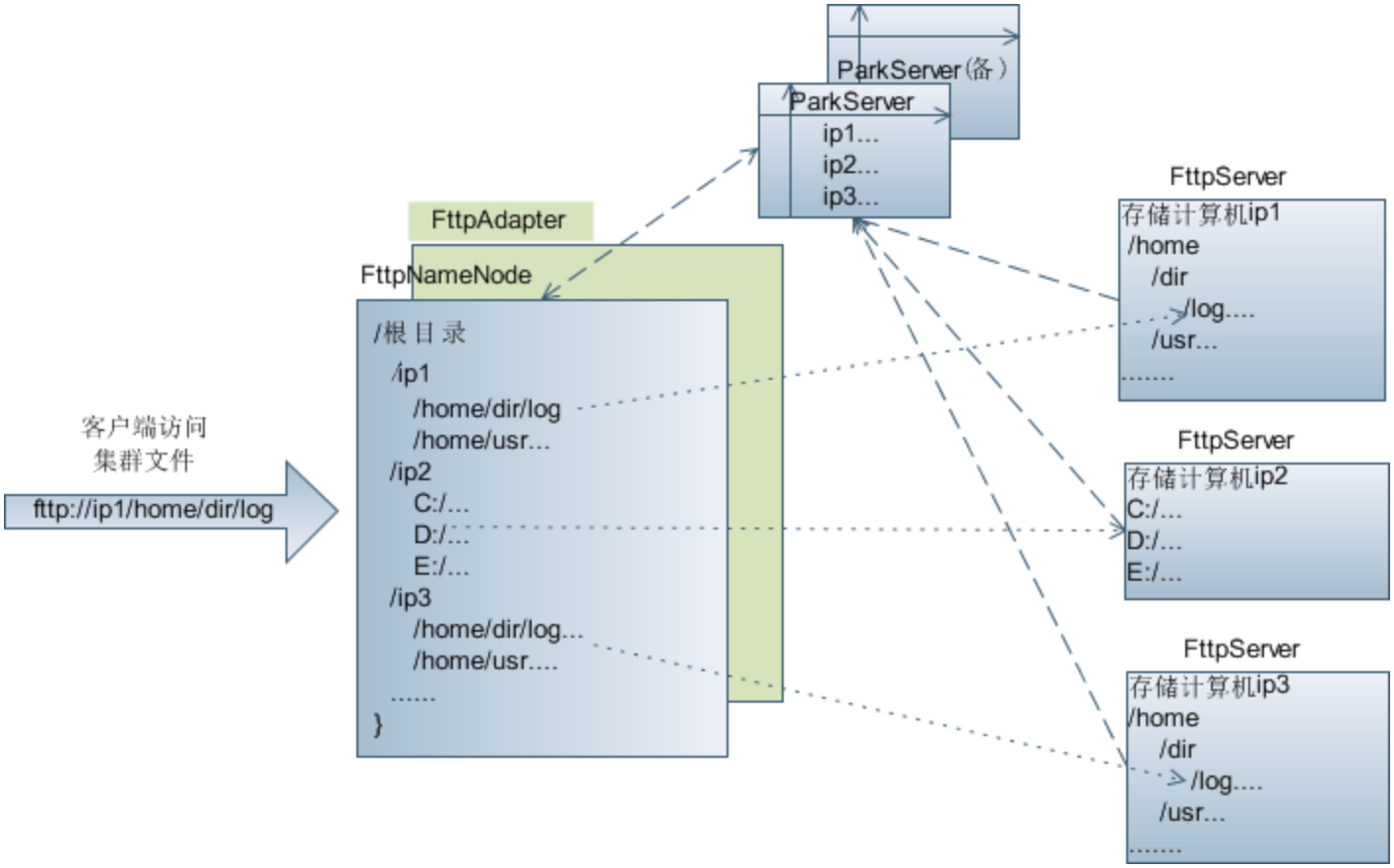
FTTP分布式文件操作



集群分布式文件系统浏览器(FttpNameNode)

搭建集群：启动ParkServer协调服务，每台存储机器启动FttpServer

FTTP分布式文件操作



FttpNameNode：查看集群所有文件和目录
FttpAdapter：提供对远程文件的所有操作和协议转换
FttpServer：提供对存储机器的文件服务
ParkServer：提供协调服务，管理集群存储机器信息

FTTP分布式文件操作

提供对集群文件的操作支持，包括：

1、元数据访问，添加删除，按块拆分，高性能并行读写，排他读写（按文件部分内容锁定），随机读写，集群复制等

2、对集群文件的解析支持（包括按行，按分割符，按最后标识读取）

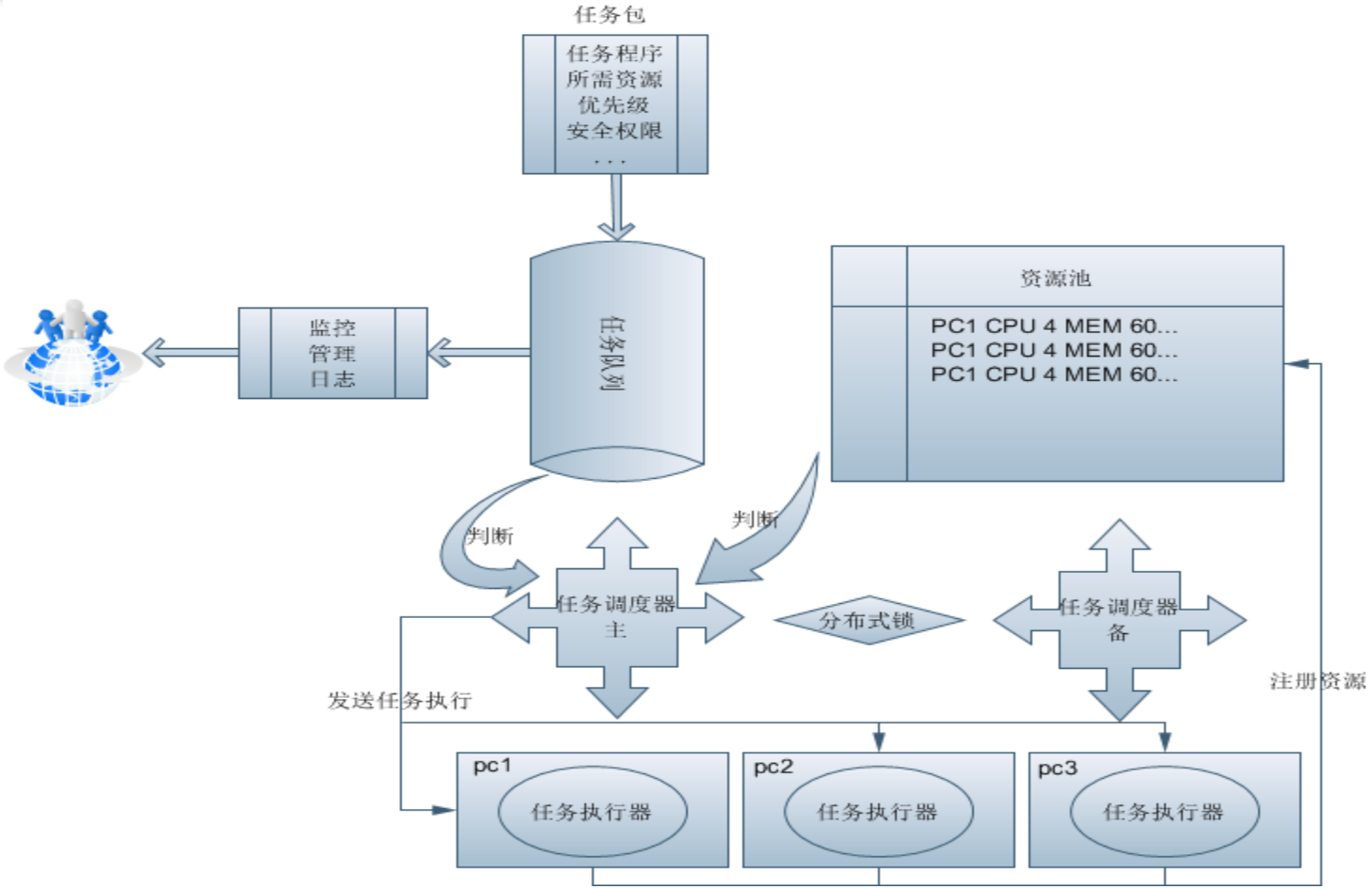
3、对整形数据的高性能读写支持（ArrayInt比ArrayList存的更多更快）

4、两阶段提交和事务补偿处理

5、自带一个集群文件浏览器，可以查看集群所有硬盘上的文件（不同于hadoop的namenode,没有单点问题和容量限制）

但是fourinone并不提供一个分布式存储系统，比如文件数据的导入导出、拆分存储、负载均衡，备份容灾等存储功能，不过开发人员可以利用这些api去设计和实现这些功能，用来满足自己的特定需求。

分布式作业调度平台



如何运用Fourinone实现调度平台：

任务队列使用消息队列实现，资源池使用缓存实现

调度器根据任务队列和资源池条件，根据调度算法进行调度

分布式锁采用分布式协调功能实现，任务执行采用自动部署实现

提纲

- 分布式并行计算
- 分布式协调
- 分布式缓存
- 消息队列
- FTTP分布式文件操作
- 分布式作业调度平台
- 应用场景:上亿数据排序

API

工头:

WareHouse giveTask(WareHouse inhouse)

实现分配工人要做的任务

WorkerLocal[] getWaitingWorkers(String workerType)

获取集群中等待的工人

WareHouse[] doTaskBatch(WorkerLocal[] wks,
WareHouse wh)

所有工人批量完成给定任务处理

doProject(WareHouse inhouse)

工头开始项目启动

toNext

多个包工头链式处理

工人:

WareHouse doTask(WareHouse inhouse);

实现工头分配的任务

waitWorking(String workerType)

等待工作状态,指定工人类型

Workman[] getWorkerAll();

获取所有的工人

Workman[] getWorkerElse();

获取除自己外的其他工人

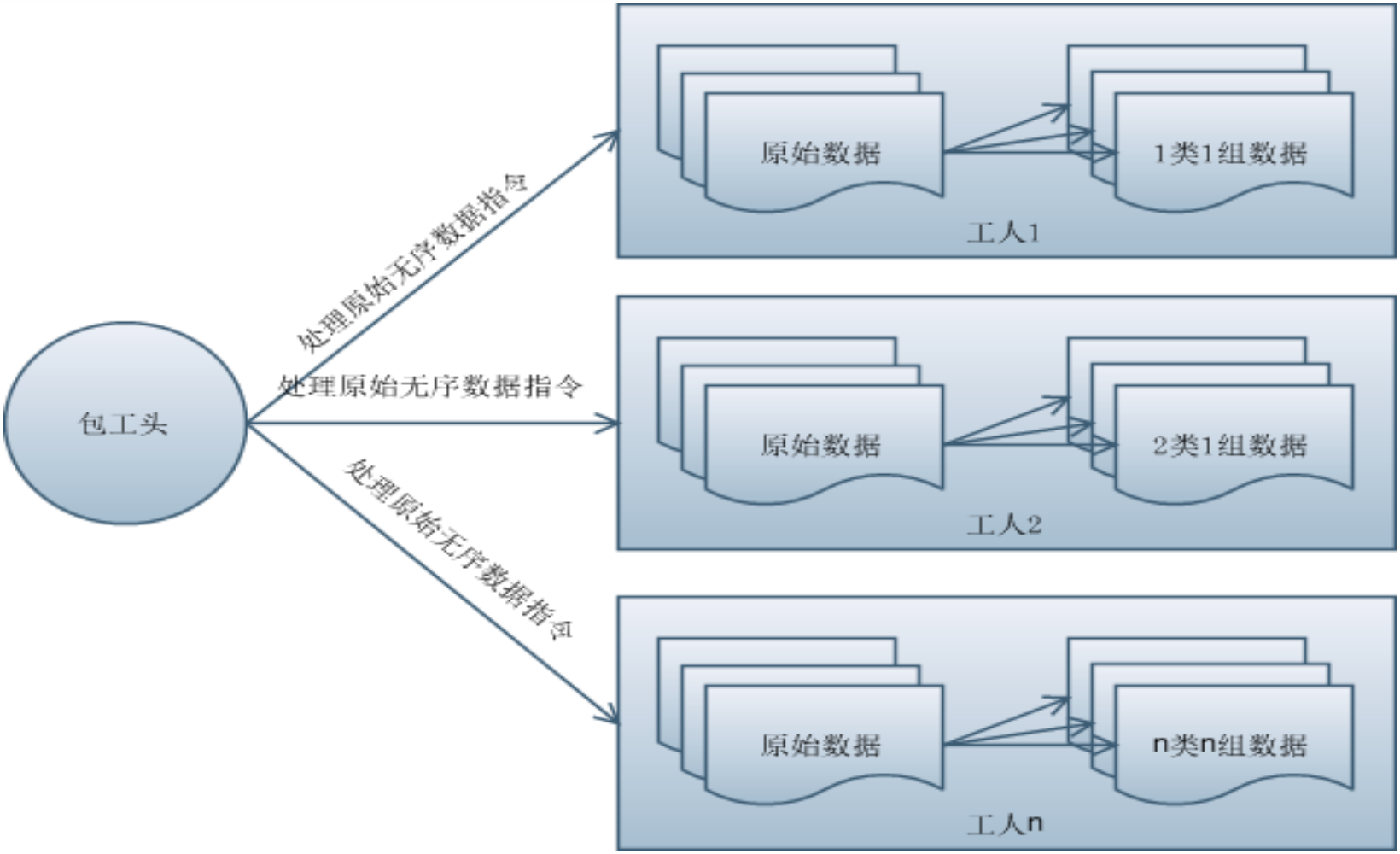
int getSelfIndex();

获取自己在工作中的位置

boolean receive(WareHouse inhouse)

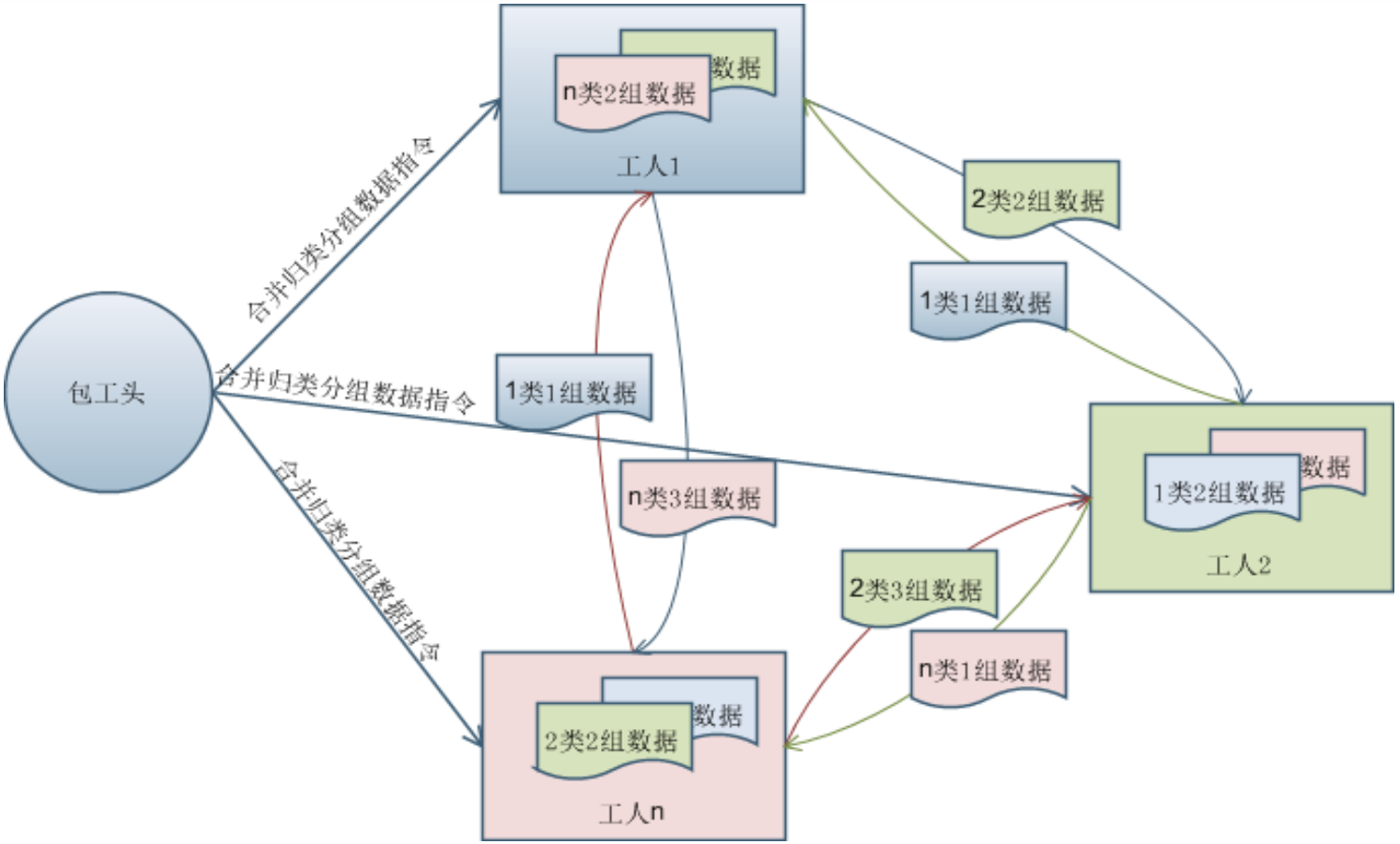
接收来自其他工人的传递

第一个环节:分类



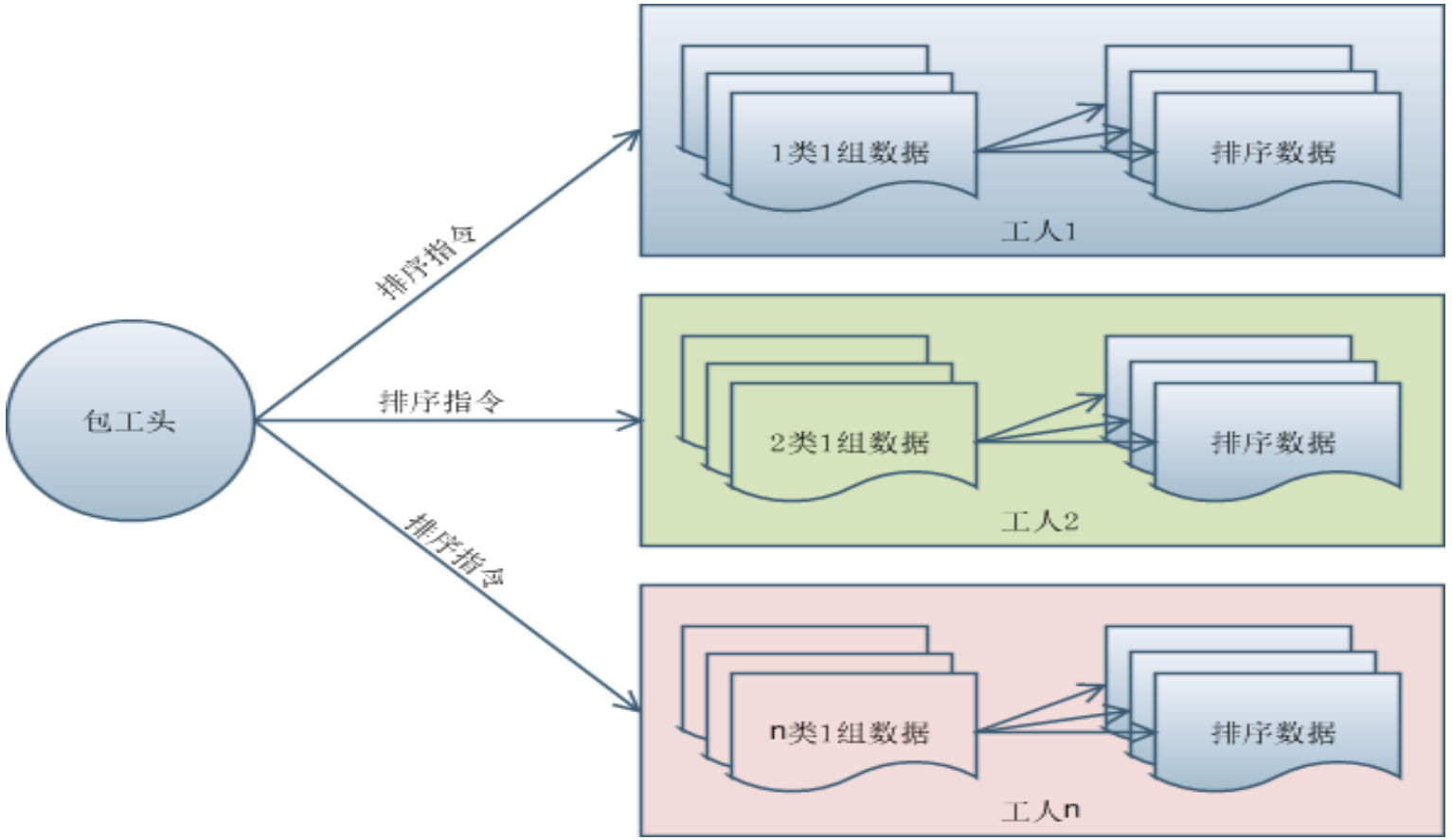
将分散到多台计算机的海量无序数据，按照工人数量和预计处理的每份数据文件大小两个唯度分类，计算出每个工人所属的数据范围

第二个环节:合并



工人彼此之间进行数据合并，合并规则：将属于其他工人的范围数据发给对方，接受对方发给属于自己范围的数据。结果：每个工人机器形成粗的范围的有序数据，但是范围内的数据仍然无序

第三个环节:排序



工人对属于自己范围的数据进行排序，最后得到一个整体原始数据的排序结果，但是它是根据范围分散到不同任务计算机上存放的。完成后返回通知包工头完成排序

淘宝网
Taobao.com

提问/交流

fourinone分布式计算博客

http://3503265.blog.51cto.com/

fourinone群：

qq群: 241116021

旺旺群：849833763

fourinone@yeah.net

qianfeng.py@taobao.com

淘宝网
Taobao.com

Thank You