(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109327509 A
(43)申请公布日 2019.02.12

(21)申请号 201811057446.2

(22)申请日 2018.09.11

(71)申请人 武汉魅瞳科技有限公司
地址 430000 湖北省武汉市东湖新技术开发区关山大道1号软件产业园4.1期E3栋703室

(72)发明人 邹复好 李开 熊饶饶 刘鹏坤 孙斌

(74)专利代理机构 武汉蓝宝石专利代理事务所（特殊普通合伙）42242
代理人 廉海涛

(51)Int.Cl.
H04L 29/08(2006.01)
H04L 29/06(2006.01)
H04L 12/26(2006.01)
H04L 12/24(2006.01)

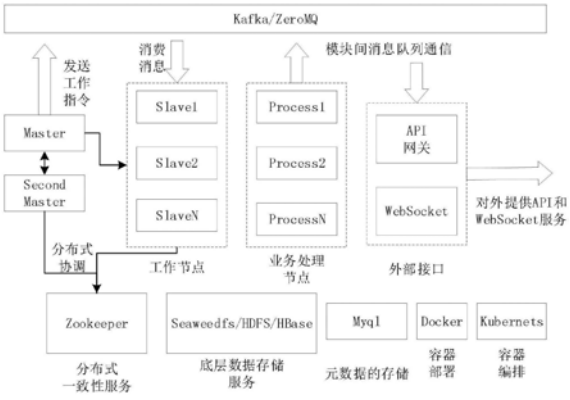
权利要求书1页 说明书6页 附图3页

(54)发明名称

一种主/从架构的低耦合的分布式流式计算框架

(57)摘要

本发明实施例提供了一种主/从架构的低耦合的分布式流式计算框架，包括：分布式服务环境、集群管理主节点、集群的工作从节点、集群的业务处理节点；所述分布式服务环境用于存储各个分布式节点的运行状态、负载状态以及任务执行状态；所述集群管理主节点用于管理集群节点的运行状态，并向各个集群节点分发任务；所述集群的工作从节点用于执行所述集群管理主节点分发的任务，并将任务执行的中间结果在Kafka中缓存；所述集群的业务处理节点用于消费Kafka中产生的消息，并得到业务结果。能够适用于多种情况下的流式数据处理，模块之间充分解耦合，能够实现动态缩扩容，实现数据的冗余备份，主节点的备份机制能实现服务的可靠性。



CN 109327509 A

1.一种主/从架构的低耦合的分布式流式计算框架,其特征在于,包括:

分布式服务环境、集群管理主节点、集群的工作从节点、集群的业务处理节点;

所述分布式服务环境用于存储各个分布式节点的运行状态、负载状态以及任务执行状态;

所述集群管理主节点用于管理集群节点的运行状态,并向各个集群节点分发任务;

所述集群的工作从节点用于执行所述集群管理主节点分发的任务,并将任务执行的中间结果在Kafka中缓存;

所述集群的业务处理节点用于消费Kafka中产生的消息,并得到业务结果。

2.根据权利要求1所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述主/从架构的低耦合的分布式流式计算框架还包括:

API网关节点,所述API网关节点用于为外部提供API服务或者WebSocket服务。

3.根据权利要求2所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述API网关节点采用了SSL加密和标准的RestfulAPI的方式来保证结果的安全获取。

4.根据权利要求1所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述分布式服务环境包括:

Zookeeper集群模块、负载均衡模块、分布式服务通信模块以及分布式服务部署模块;

所述Zookeeper集群模块用于保存分布式节点的运行状态和服务状态;

所述负载均衡模块用于实现任务的负载均衡;

所述分布式服务通信模块用于实现分布式环境下的节点之间的通信;

所述分布式服务部署模块用于采用容器技术实现分布式服务的部署,采并用统一的RestfulAPI接口用于容器服务间的通信。

5.根据权利要求4所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述分布式服务通信模块还用于实现流数据的传输。

6.根据权利要求1所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述集群管理主节点包括:

备份模块和监控模块,所述备份模块采用Zookeeper的Watch机制来进行主从备份,当主节点任务失败时,备份节点立刻进行工作状态;

所述监控模块采用Zookeeper的心跳机制来实时监控从节点的工作状态。

7.根据权利要求1所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述集群的工作从节点还用于将节点状态保存在Zookeeper中。

8.根据权利要求1所述的主/从架构的低耦合的分布式流式计算框架,其特征在于,所述集群的业务处理节点还用于将流式数据的结果实时推送到WebSocket中,以使所述流式数据的结果显示到前端浏览器上。

一种主/从架构的低耦合的分布式流式计算框架

技术领域

[0001] 本发明实施例涉及大数据处理和流式计算技术领域，尤其涉及一种主/从架构的低耦合的分布式流式计算框架。

背景技术

[0002] 近年来，随着信息技术的快速发展，数据量呈现飞速增长的趋势，对于海量数据，单台计算机的处理能力已经远远不够，由此推动了分布式系统的研究和进展。分布式计算系统的核心思想就是“分而治之”，将海量数据源进行任务分割，将分割后的任务分发给多台计算机并行处理，并将并行处理的结果合并为最终的结果。分布式计算机集群通过网络互连，可以实现资源的共享、协同工作、并行化处理，对外提供统一的接口，呈现单个完整的计算系统。在海量的、复杂的数据环境中，不仅包含静态的、结构化的数据，还包括源源不断持续产生的、实时性强、非结构化的数据，例如摄像头采集的视频数据，服务器产生的日志数据，搜索引擎的日志等。如何在海量数据中快速分析获取有用的信息是现在分布式计算领域的研究热点。

[0003] 针对流数据应用场景，与传统的存储在磁盘或内存中的数据不同，流数据的特点在于：实时性：数据流实时产生，需要实时得出分析结果；持久性：数据流无限、持续产生和流入；容错性：对于流数据，经过系统处理后变丢弃，很难恢复数据流，因此需要保证数据源的可靠性处理。例如在分析摄像头的视频流的应用场景中，对于每个摄像头抓取到的视频帧的处理，要考虑到在有限的计算资源的情况下，如何能够实现生产消费平衡，保证流数据处理的实时性，确保每一个数据都被处理好。现有典型的分布式流计算框架有Storm、Sparkstreaming、Flink等，这些框架在分布式环境下的实时性和容错性都很不错，但是针对特定的业务场景，耦合度过高，增加开发维护成本，降低模块间的异构性，且框架的代码利用率低，造成一定的系统资源浪费。

[0004] 在流式计算场景下，通用的流计算框架有相对笨重、耦合度高、异构性低的缺点。因此，现在亟需一种新的分布式流式计算框架来解决上述现有技术中存在的问题。

发明内容

[0005] 为了解决上述问题，本发明实施例提供一种克服上述问题或者至少部分地解决上述问题的一种主/从架构的低耦合的分布式流式计算框架。

[0006] 第一方面本发明实施例提供一种主/从架构的低耦合的分布式流式计算框架，包括：

[0007] 分布式服务环境、集群管理主节点、集群的工作从节点、集群的业务处理节点；

[0008] 所述分布式服务环境用于存储各个分布式节点的运行状态、负载状态以及任务执行状态；

[0009] 所述集群管理主节点用于管理集群节点的运行状态，并向各个集群节点分发任务；

[0010] 所述集群的工作从节点用于执行所述集群管理主节点分发的任务,并将任务执行的中间结果在Kafka中缓存;

[0011] 所述集群的业务处理节点用于消费Kafka中产生的消息,并得到业务结果。

[0012] 其中,所述主/从架构的低耦合的分布式流式计算框架还包括:

[0013] API网关节点,所述API网关节点用于为外部提供API服务或者WebSocket服务。

[0014] 其中,所述API网关节点采用了SSL加密和标准的RestfulAPI的方式来保证结果的安全获取。

[0015] 其中,所述分布式服务环境包括:

[0016] Zookeeper集群模块、负载均衡模块、分布式服务通信模块以及分布式服务部署模块;

[0017] 所述Zookeeper集群模块用于保存分布式节点的运行状态和服务状态;

[0018] 所述负载均衡模块用于实现任务的负载均衡;

[0019] 所述分布式服务通信模块用于实现分布式环境下的节点之间的通信;

[0020] 所述分布式服务部署模块用于采用容器技术实现分布式服务的部署,采并用统一的RestfulAPI接口用于容器服务间的通信。

[0021] 其中,所述分布式服务通信模块还用于实现流数据的传输。

[0022] 其中,所述集群管理主节点包括:

[0023] 备份模块和监控模块,所述备份模块采用Zookeeper的Watch机制来进行主从备份,当主节点任务失败时,备份节点立刻进行工作状态;

[0024] 所述监控模块采用Zookeeper的心跳机制来实时监控从节点的工作状态。

[0025] 其中,所述集群的工作从节点还用于将节点状态保存在Zookeeper中。

[0026] 其中,所述集群的业务处理节点还用于将流式数据的结果实时推送到WebSocket中,以使所述流式数据的结果显示到前端浏览器上。

[0027] 本发明实施例提供的主/从架构的低耦合的分布式流式计算框架,能够适用于多种情况下的流式数据处理,模块之间充分解耦合,能够实现动态缩扩容,实现数据的冗余备份,主节点的备份机制能实现服务的可靠性。

附图说明

[0028] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作一简单地介绍,显而易见地,下面描述中的附图是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0029] 图1是本发明实施例提供的一种主/从架构的低耦合的分布式流式计算框架结构示意图;

[0030] 图2是本发明实施例提供的API网关结构示意图;

[0031] 图3是本发明实施例提供的分布式基础环境构成示意图;

[0032] 图4是本发明实施例提供的集群管理主节点功能示意图;

[0033] 图5是本发明实施例提供的集群的工作从节点功能示意图;

[0034] 图6是本发明实施例提供的业务处理节点功能示意图。

具体实施方式

[0035] 为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0036] 目前，现有典型的分布式流计算框架有Storm、Sparkstreaming、Flink等，这些框架在分布式环境下的实时性和容错性都很不错，但是针对特定的业务场景，耦合度过高，增加开发维护成本，降低模块间的异构性，且框架的代码利用率低，造成一定的系统资源浪费。

[0037] 针对上述现有技术中存在的问题，图1是本发明实施例提供的一种主/从架构的低耦合的分布式流式计算框架结构示意图，如图1所示，所述一种主/从架构的低耦合的分布式流式计算框架包括：

[0038] 分布式服务环境、集群管理主节点、集群的工作从节点、集群的业务处理节点；

[0039] 所述分布式服务环境用于存储各个分布式节点的运行状态、负载状态以及任务执行状态；

[0040] 所述集群管理主节点用于管理集群节点的运行状态，并向各个集群节点分发任务；

[0041] 所述集群的工作从节点用于执行所述集群管理主节点分发的任务，并将任务执行的中间结果在Kafka中缓存；

[0042] 所述集群的业务处理节点用于消费Kafka中产生的消息，并得到业务结果。

[0043] 需要说明的是，本发明实施例实质上为了提供该主/从架构的低耦合的分布式流式计算框架，实质上是在计算机软件层面上进行框架搭建的过程。

[0044] 具体的，首先，本发明实施例需要搭建主/从架构的低耦合的分布式流式计算框架中的分布式服务环境。本发明实施例采用了Zookeeper作为分布式协调服务工具，搭建分布式环境，存储分布式节点的运行状态、负载状态、任务执行状态。采用SeaweadFS/HDFS/HBase来作为分布式数据存储环境。利用Kafka/ZeroMQ作为分布式环境下的消息中间件，负责各个模块直接的通信，以及流处理中间结果的推送，以实现低耦合特性。利用Docker和kubernetes来实现服务的容器化部署和分布式环境的容器编排。

[0045] 然后，本发明实施例需要搭建集群管理的主节点(Master节点)，该节点的功能是负责分布式系统任务调度、负载均衡，管理集群节点的运行状态，任务的分发，机器的任务负载状态和任务的进度管理等。

[0046] 紧接着，本发明实施例需要搭建集群的工作从节点(Slave节点)。该节点通常有多个，可根据机器的配置来动态扩展节点的数量。该节点的功能是负责执行主节点分发的任务，将任务执行的中间结果发送到Kafka上面缓存，等待其他模块消费。

[0047] 下一步，本发明实施例还需要搭建集群的业务处理(Process)节点。该节点通常也有多个，可根据业务需求量动态配置。该节点负责消费Kafka中从节点在分布式消息系统中生产的消息，最终产生业务相关的结果。

[0048] 从而通过上述节点和配置环境实现了框架的构建。与现有技术相比，本发明实施例提供的主/从架构的低耦合的分布式流式计算框架有下述优点：

[0049] 1、能够降低处理模块之间的耦合性，便于开发维护。对于不同的业务，可以在模块之间并行开发，单独测试；

[0050] 2、能够提高框架的异构性，各个模块之间只需要遵守预先定义好的数据通信协议，本身可以根据应用场景采用不同的编程语言实现。

[0051] 3、能够提高计算资源的利用率，和通用的流式处理框架相比，本发明的框架为轻量级的，可减少通用框架部分不需要的功能，降低计算机资源消耗。

[0052] 4、扩展性好，隔离性高。在分布式集群中，本框架可根据业务需求自动缩扩容，满足业务的吞吐量需求。容器化部署的方式以及分布式环境的容器编排技术可以提高服务的可靠性。

[0053] 在上述实施例的基础上，所述主/从架构的低耦合的分布式流式计算框架还包括：

[0054] API网关节点，所述API网关节点用于为外部提供API服务或者WebSocket服务。

[0055] 需要说明的是，如图1所示，本发明实施例提供的主/从架构的低耦合的分布式流式计算框架实质上还包括第五部分的内容，也就是API网关节点，该节点负责对外部提供API服务或者WebSocket服务(针对流式数据)。该节点作为集群的门户，承担集群的管理控制、响应外部请求、主动推送流式数据的处理结果。

[0056] 所述API网关节点采用了SSL加密和标准的RestfulAPI的方式来保证结果的安全获取。

[0057] 图2是本发明实施例提供的API网关结构示意图，如图2所示，可以理解的是，API网关是整个框架的出入口，负责集群的管理、响应外部的请求。在多数情况下，API节点可通过RestfulAPI对集群进行管理或者响应业务相关的请求，但是对于流式数据，例如处理之后的视频流，可以采用WebSocket长连接的通信方式持续获取结果。

[0058] 在上述实施例的基础上，所述分布式服务环境包括：

[0059] Zookeeper集群模块、负载均衡模块、分布式服务通信模块以及分布式服务部署模块；

[0060] 所述Zookeeper集群模块用于保存分布式节点的运行状态和服务状态；

[0061] 所述负载均衡模块用于实现任务的负载均衡；

[0062] 所述分布式服务通信模块用于实现分布式环境下的节点之间的通信；

[0063] 所述分布式服务部署模块用于采用容器技术实现分布式服务的部署，采并用统一的RestfulAPI接口用于容器服务间的通信。

[0064] 图3是本发明实施例提供的分布式基础环境构成示意图，如图3所示，本发明实施例在搭建分布式基础环境时，包含了分布式文件存储、消息队列和分布式协同工具。分布式文件系统可以采用多种方案，这里用HDFS做说明，HDFS是一个可靠的分布式文件系统，适合存储超大的文件，SeaweadFs适合存储大量小文件。消息队列或者RPC是用于框架各个模块之间的通信。Zookeeper作为集群管理工具，用于从节点和主节点之间的通信。

[0065] 具体的，本发明实施例提供的Zookeeper集群模块、负载均衡模块、分布式服务通信模块以及分布式服务部署模块实质上也是计算机软件搭建的过程。

[0066] 首先，本发明实施例需要搭建Zookeeper集群，即本发明实施例所述的Zookeeper集群模块。Zookeeper集群负责保存分布式节点的运行状态，服务状态。利用Zookeeper实现心跳检测机制，保证主和从之间的关联，一旦发现从无法访问，则将已分配出去的任务重新

分发到新的节点。

[0067] 紧接着，本发明实施例需要实现负载均衡设置，即本发明实施例所述的负载均衡模块，负载均衡设置是采用加权最少任务算法来实现任务的负载均衡。对于不同的从节点，根据性能来设置一个最大的处理并发数，根据集群所有机器的负载情况来选择一个当前正在处理的任务的数量最小工作节点。

[0068] 然后，本发明实施例需要实现分布式服务通信，具体的实现方式是采用消息队列的方式来实现分布式环境下的节点之间的通信以及流数据的传输。

[0069] 最后，本发明实施例需要完成分布式服务部署，即本发明实施例所述的分布式服务部署模块，具体的是采用容器技术实现分布式服务的部署，采用统一的RestfulAPI接口用于容器服务间的通信。在分布式环境下，可以采用分布式容器编排技术来实现容器的跨机器的通信。

[0070] 在上述实施例的基础上，所述分布式服务通信模块还用于实现流数据的传输。

[0071] 由上述内容可知，本发明实施例提供了节点之间的通信以及流数据的传输两种方式，对于流式数据，数据源不断产生数据，从节点对源数据不断进行处理，将中间结果传入到KafkaTopic中，同时将处理的部分信息存储到MySql中。对于Kafka中传输的数据，采用base64来对二进制数据编码传输。采用JSON定义通信协议，方便后续Process的业务处理。

[0072] 在上述实施例的基础上，所述集群管理主节点包括：

[0073] 备份模块和监控模块，所述备份模块采用Zookeeper的Watch机制来进行主从备份，当主节点任务失败时，备份节点立刻进行工作状态；

[0074] 所述监控模块采用Zookeeper的心跳机制来实时监控从节点的工作状态。

[0075] 图4是本发明实施例提供的集群管理主节点功能示意图，如图4所示，在搭建好分布式服务环境的基础上，主节点和Zookeeper保持长连接，通过Zookeeper来获取从节点的运行状态，通过心跳机制来保证从节点的在线状态。

[0076] 可以理解的是，本发明实施例利用Zookeeper的Watch机制作为备份模块来实现主节点的主从备份，一旦主节点失败挂掉，备份节点立即进入工作状态；与此同时，本发明实施例利用Zookeeper的心跳机制作为监控模块来实时监控从节点的工作状态。

[0077] 在上述实施例的基础上，所述集群的工作从节点还用于将节点状态保存在Zookeeper中。

[0078] 图5是本发明实施例提供的集群的工作从节点功能示意图，如图5所示，本发明实施例提供的集群的工作从节点是逻辑上的一个节点，负责执行主节点分发的任务，任务处理的结果送入消息队列缓存起来。从节点的状态保存在Zookeeper中。

[0079] 需要说明的是，本发明实施例中的搭建从节点的阶段，从节点会执行主节点分发的任务，对流式数据源进行预处理，将预处理的结果放到消息队列中，供业务处理节点调用。

[0080] 在上述实施例的基础上，所述集群的业务处理节点还用于将流式数据的结果实时推送到WebSocket中，以使所述流式数据的结果显示到前端浏览器上。

[0081] 图6是本发明实施例提供的业务处理节点功能示意图，如图6所示，本发明实施例提供的业务处理(process)节点也是逻辑上的一个节点，负责最终的业务处理，处理的结果会持久化到数据库中，对于流式数据，直接将结果实时推送到WebSocket中，显示到前端浏

览器上。

[0082] 需要说明的是，本发明实施例中的搭建Process节点的阶段，Process节点是处理具体业务的节点，从消息队列中消费数据，然后对数据做具体的业务分析。例如：对摄像头采集到的图片做人脸识别，性别识别，应用日志分析等，这些具体的业务由用户自行定义。

[0083] 综上所述，本发明实施例提供的主/从架构的低耦合的分布式流式计算框架能够适用于多种情况下的流式数据处理，模块之间充分解耦合，能够实现动态缩扩容，实现数据的冗余备份，主节点的备份机制能实现服务的可靠性。

[0084] 通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到各实施方式可借助软件加必需的通用硬件平台的方式来实现，当然也可以通过硬件。基于这样的理解，上述技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在计算机可读存储介质中，如ROM/RAM、磁碟、光盘等，包括若干指令用以使得一台计算机设备(可以是个人计算机，服务器，或者网络设备等)执行各个实施例或者实施例的某些部分所述的方法。

[0085] 最后应说明的是：以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本发明各实施例技术方案的精神和范围。

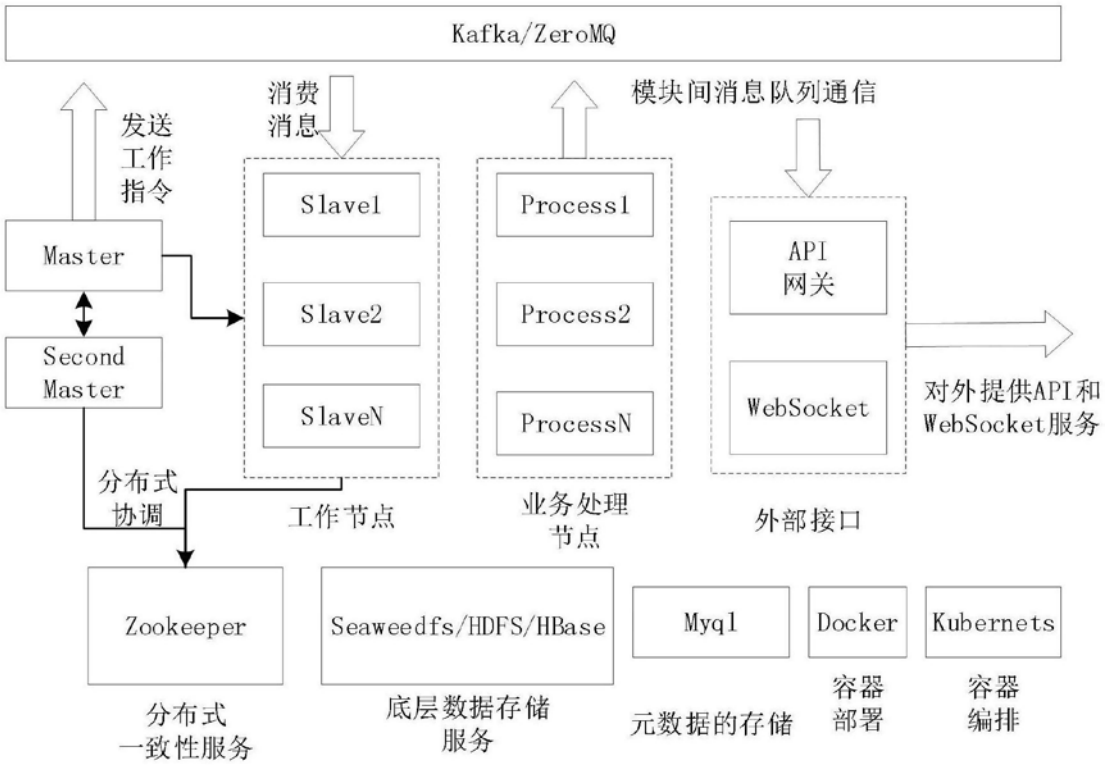

Kafka/ZeroMQ
消费
消息
模块间消息队列通信
发送
工作
指令
Master
Second
Master
Slave1
Slave2
SlaveN
Process1
Process2
ProcessN
API
网关
WebSocket
对外提供API和
WebSocket服务
分布式
协调
工作节点
业务处理
节点
外部接口
Zookeeper
Seaweedfs/HDFS/HBase
Myql
Docker
Kubernets
分布式
一致性服务
底层数据存储
服务
元数据的存储
容器
部署
容器
编排

图1

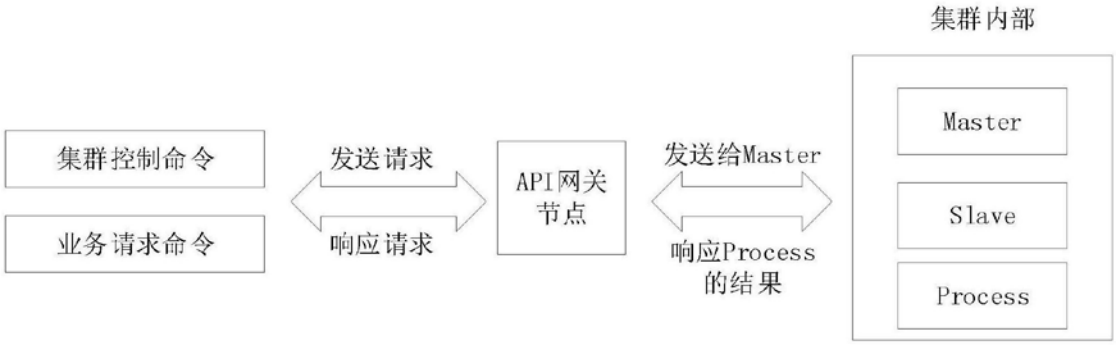

集群内部
集群控制命令
业务请求命令
发送请求
响应请求
API网关
节点
发送给Master
响应Process
的结果
Master
Slave
Process

图2

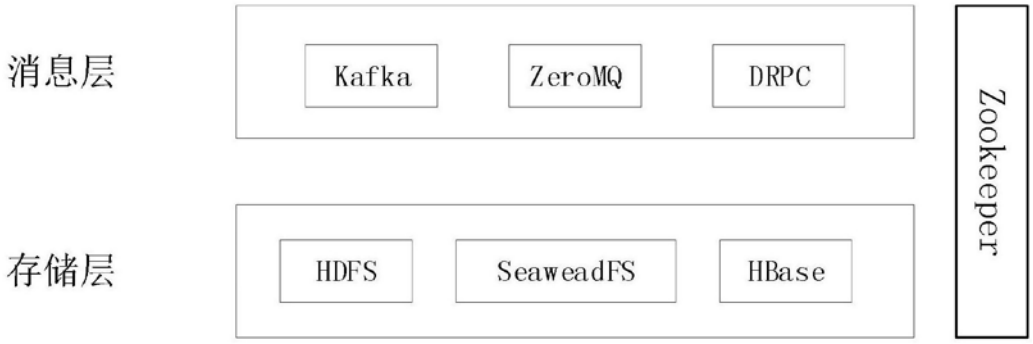

消息层
Kafka
ZeroMQ
DRPC
存储层
HDFS
SeaweadFS
HBase
Zookeeper

图3

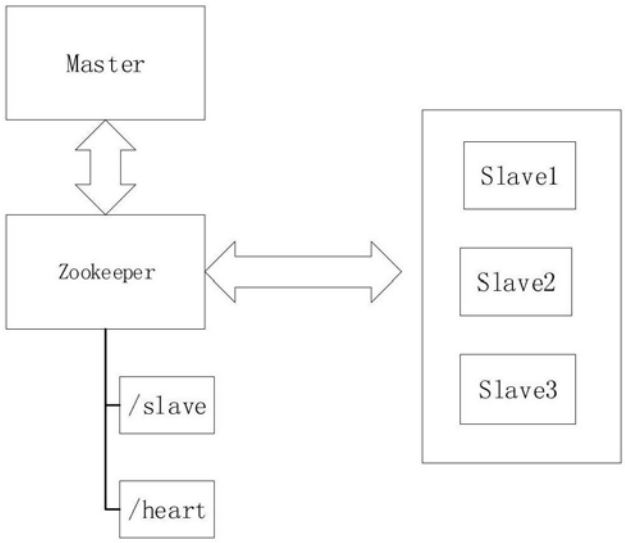

Master
Zookeeper
/slave
/heart
Slave1
Slave2
Slave3

图4

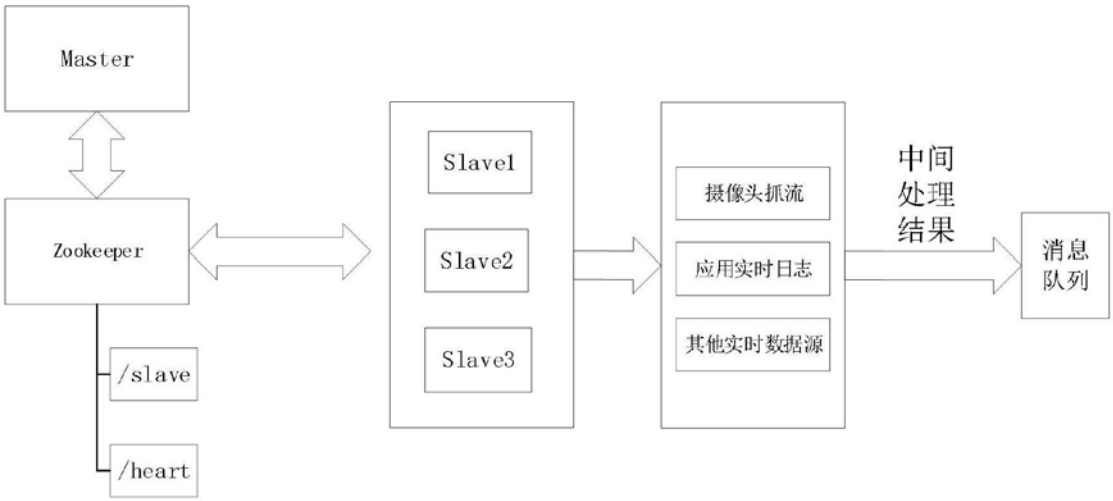

Master
Zookeeper
/slave
/heart
Slave1
Slave2
Slave3
摄像头抓流
应用实时日志
其他实时数据源
中间
处理
结果
消息
队列

图5

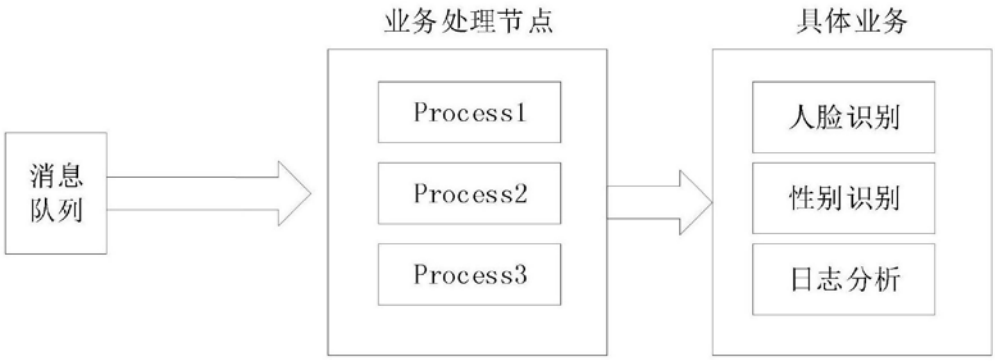

业务处理节点
具体业务
消息
队列
Process1
Process2
Process3
人脸识别
性别识别
日志分析

图6