(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 107046510 A
(43)申请公布日 2017.08.15

(21)申请号 201710023537.3

(22)申请日 2017.01.13

(71)申请人 广西电网有限责任公司电力科学研究院
地址 530023 广西壮族自治区南宁市兴宁区民主路6-2号

(72)发明人 吴秋莉 郭丽娟 邬蓉蓉 吕泽承 张炜

(74)专利代理机构 南宁东智知识产权代理事务所(特殊普通合伙) 45117
代理人 戴燕桃 巢雄辉

(51)Int.Cl.
H04L 12/931(2013.01)
H04L 29/08(2006.01)
H04L 12/927(2013.01)

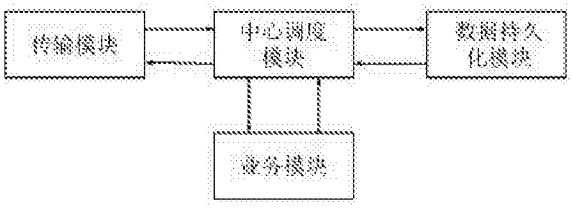
权利要求书1页 说明书6页 附图1页

(54)发明名称

一种适用于分布式计算系统的节点及其组成的系统

(57)摘要

本发明公开了一种适用于分布式计算系统的节点及其组成的系统，涉及数据处理技术领域，能够对“无限”运动着的数据进行高效率的实时处理。该适用于分布式计算系统的节点包括：传输模块，用于将接收到的来自上游节点的输入数据流根据输入数据流名切分，并通过与输入数据流名一一对应的有名管道传送给中心调度模块；中心调度模块，用于接收来自传输模块切分好的输入数据流，并将接收到的输入数据流放入以待处理任务名称命名的链表中，基于各待处理任务的优先级开启待处理任务，并将待处理任务发送给业务模块；还用于接收来自业务模块的输出数据流，并转发给传输模块；业务模块，对来自中心调度模块的输入数据流进行计算处理，并向中心调度模块输出处理之后的数据流。



CN 107046510 A

1.一种适用于分布式计算系统的节点，其特征在于，包括：

传输模块，用于将接收到的来自上游节点的输入数据流根据输入数据流名切分，并通过与所述输入数据流名一一对应的有名管道传送给中心调度模块；还用于将来自所述中心调度模块的输出数据流传输给对应的下游节点；

中心调度模块，用于接收来自所述传输模块切分好的输入数据流，并将接收到的输入数据流放入以待处理任务名称命名的链表中，基于各待处理任务的优先级开启待处理任务，并将待处理任务发送给业务模块；还用于接收来自所述业务模块的输出数据流，并转发给所述传输模块；

业务模块，用于基于开启的待处理任务，对来自所述中心调度模块的输入数据流进行计算处理，并向所述中心调度模块输出处理之后的数据流。

2.根据权利要求1所述的节点，其特征在于，还包括：

数据持久化模块，用于接收来自所述中心调度模块的输出数据流和对应的输出数据流名，并进行数据持久化存储。

3.根据权利要求2所述的节点，其特征在于，所述数据持久化模块用于接收到来自所述中心调度模块的输出数据流名时，从对应的有名管道持续读取来自所述中心调度模块的输出数据流，并进行存储。

4.根据权利要求3所述的节点，其特征在于，所述数据持久化模块还用于将已经进行存储的输出数据流名和对应的输出数据流发送到以下游节点命名的共享内存队列中，并将所述下游节点名封装为任务，放入任务队列中。

5.根据权利要求4所述的节点，其特征在于，所述传输模块用于取出所述任务队列中的下游节点名，从所述共享内存队列中获取对应的输出数据流和输出数据流名，并发送给对应的下游节点。

6.根据权利要求1所述的节点，其特征在于，所述中心调度模块还用于基于所述节点的负载决定启动的待处理任务的数量，并根据各所述待处理任务的优先级开启待处理任务。

7.根据权利要求6所述的节点，其特征在于，所述待处理任务的数量基于所述节点的处理器的个数和密集计算所占的时间比重确定。

8.根据权利要求2所述的节点，其特征在于，所述传输模块、所述中心调度模块、所述业务模块和所述数据持久化模块之间通过进程间通信方式通信。

9.一种分布式计算系统，其特征在于，包括多个如权利要求1至8任一项所述的节点。

一种适用于分布式计算系统的节点及其组成的系统

技术领域

[0001] 本发明涉及数据处理领域，尤其涉及一种适用于分布式计算系统的节点及其组成的系统。

背景技术

[0002] 随着信息技术的飞速发展，基于数据处理和数据分析的应用受到了广泛的欢迎和关注。大量信息源带来数据规模的爆发式增长，对海量数据进行复杂计算已经远远超过单台计算机的处理能力，由此推动了对分布式系统及其关键技术的研究。分布式计算中把需要进行复杂计算的海量数据切分成小块后，分交由多台计算机并行处理，并将局部计算结果整合得出最终结果。

[0003] 在复杂、海量、异构的数据环境中，不仅包含静态的、离线的、结构化的数据，还有实时传输的、持续产生的、非结构化的数据。例如传感器网络实时产生的监控数据，在线服务器产生的统计信息，路由器数据报的统计，这些数据时时刻刻都在“运动”的。面对这些“无限”运动着的数据，如果不能对其进行高效率的实时处理，将错过数据流中携带的关键信息。整合来自多个异构数据源的“运动”数据，在其“运动”的过程执行复杂的逻辑处理，包括数值计算、数据挖掘和模型预测，实时地给出用户关心的结果，这是传统分布式计算模式所无法做到的。

发明内容

[0004] 本发明所要解决的技术问题在于提供一种适用于分布式计算系统的节点及其组成的系统，能够对“无限”运动着的数据进行高效率的实时处理。

[0005] 为解决上述技术问题，本发明采用如下技术方案：

本发明提供了一种适用于分布式计算系统的节点，该节点包括：

传输模块，用于将接收到的来自上游节点的输入数据流根据输入数据流名切分，并通过与所述输入数据流名一一对应的有名管道传送给中心调度模块；还用于将来自所述中心调度模块的输出数据流传输给对应的下游节点；

中心调度模块，用于接收来自所述传输模块切分好的输入数据流，并将接收到的输入数据流放入以待处理任务名称命名的链表中，基于各待处理任务的优先级开启待处理任务，并将待处理任务发送给业务模块；还用于接收来自所述业务模块的输出数据流，并转发给所述传输模块；

业务模块，用于基于开启的待处理任务，对来自所述中心调度模块的输入数据流进行计算处理，并向所述中心调度模块输出处理之后的数据流。

[0006] 优选的，该节点还包括：

数据持久化模块，用于接收来自所述中心调度模块的输出数据流和对应的输出数据流名，并进行数据持久化存储。

[0007] 优选的，所述数据持久化模块用于接收到来自所述中心调度模块的输出数据流名

时，从对应的有名管道持续读取来自所述中心调度模块的输出数据流，并进行存储。

[0008] 优选的，所述数据持久化模块还用于将已经进行存储的输出数据流名和对应的输出数据流发送到以下游节点命名的共享内存队列中，并将所述下游节点名封装为任务，放入任务队列中。

[0009] 优选的，所述传输模块用于取出所述任务队列中的下游节点名，从所述共享内存队列中获取对应的输出数据流和输出数据流名，并发送给对应的下游节点。

[0010] 优选的，所述中心调度模块还用于基于所述节点的负载决定启动的待处理任务的数量，并根据各所述待处理任务的优先级开启待处理任务。

[0011] 优选的，所述待处理任务的数量基于所述节点的处理器的个数和密集计算所占的时间比重确定。

[0012] 优选的，所述传输模块、所述中心调度模块、所述业务模块和所述数据持久化模块之间通过进程间通信方式通信。

[0013] 本发明实施例提供了一种适用于分布式计算系统的节点，该节点能够有效地对“无限”运动着的数据进行高效率的实时处理。

[0014] 本发明另一方面提供了一种分布式计算系统，该分布式计算系统包括多个节点。

附图说明

[0015] 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，下面将对实施例描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅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。

[0016] 图1为本发明实施例提供的适用于分布式计算系统的节点的结构示意图。

具体实施方式

[0017] 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、完整地描述，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。

[0018] 本发明实施例提供了一种分布式计算系统，该分布式计算系统包括多个节点。如图1所示，任一节点包括：

传输模块，用于将接收到的来自上游节点的输入数据流根据输入数据流名切分，并通过与输入数据流名一一对应的有名管道传送给中心调度模块；还用于将来自中心调度模块的输出数据流传输给对应的下游节点；

中心调度模块，用于接收来自传输模块切分好的输入数据流，并将接收到的输入数据流放入以待处理任务名称命名的链表中，基于各待处理任务的优先级开启待处理任务，并将待处理任务发送给业务模块；还用于接收来自业务模块的输出数据流，并转发给传输模块；

业务模块，用于基于开启的待处理任务，对来自中心调度模块的输入数据流进行计算处理，并向中心调度模块输出处理之后的数据流。

[0019] 其中，传输模块负责节点与外界节点之间的数据转发，包括上游节点推送的数据和推向下游节点的数据。传输模块使节点的数据传输和内部的逻辑计算剥离开来，包括该节点的分布式计算系统的用户可以仅仅关注于业务逻辑的开发，而不用了解数据的底层通信机制，提高了用户的使用体验。

[0020] 具体的，在传输模块中会将上游节点推送的输入数据按输入数据流名切分，并通过有名管道将输入数据流传送给中心调度模块。中心调度模块是节点的枢纽，它负责接收传输模块传送的多条输入数据流。中心调度模块根据输入数据流和本节点的待处理任务的订阅关系，维护一个针对所有处于就绪状态的待处理任务的优先级队列。中心调度模块会根据节点的负载确定启动多少个任务，并从优先级队列中选择相应个数的、优先级较高的待处理任务启动。此外，中心调度模块还负责将输入数据流传送给执行待处理任务的业务模块，并接收经过业务模块处理后的输出数据流。

[0021] 和传统关系型数据处理将数据保存在磁盘上并在内存中维护磁盘数据的索引的方式不同，在包括该节点的分布式计算系统中，对输入或输出数据流的切割、融合和处理都在内存中完成，保证了极快的处理速度和响应速度。但是这也带来了弊端，即相对于磁盘上“落地”的长期数据，内存数据是一种非稳定的数据，当节点因故障重新启动或直接宕机时，内存中的数据将全部消失，在该分布式计算系统中引起事故扩散，影响系统计算结果的准确性。为此，本发明实施例提供的节点还包括数据持久化模块。当中心调度节点收到业务模块处理、计算完毕的输出数据流时，通过对应的有名管道发送给数据持久化模块，数据持久化模块按输出数据流名将输出数据流保存在磁盘上做持久化备份后，按照输出数据流和下游节点的订阅关系，将输出数据流发送到以下游节点命名的共享内存队列上，由传输模块统一发送。

[0022] 进一步的，中心调度模块还负责在指定端口上监听上游海量节点的请求，建立连接，并将连接派发给合适的业务模块执行。由于在本分布式计算系统中，每个节点管理上游节点的多个连接请求，可以利用多路复用技术epoll。节点中包括的传输模块、中心调度模块和数据持久化模块都是通过epoll管理多个事件源，并通过管道或是生产者-消费者模型的中间队列方式耦合。

[0023] 在本分布式计算系统中，所有的节点既是客户端，又是服务器，将高性能网络服务器处理高并发的网络模式移植到节点中，是其支持高连接数、高吞吐量和高时效性的关键。本分布式计算系统的节点的核心模块都是采取这种模式，传输模块通过epoll管理输入数据流的接收和输出数据流的发送，中心调度模块和数据持久化模块通过epoll管理用于模块之间数据传输的管道端口。所有模块并行处理并异步执行逻辑，通过各种进程间通信方式藕合。这样的结构使得本分布式计算系统可以最大化利用中央处理器(Central Processing Unit，简称CPU)资源和输入输出(I/O)资源，拥有极大的吞吐量和数据处理能力。

[0024] 中心调度模块还负责在节点启动时的初始化和监听工作。中心调度模块负责监听指定端口并接收来自外部节点的连接请求并初始化业务模块的线程。中心调度模块根据每个业务模块的线程的负载情况决定将包装成“任务”的连接分配给哪一个线程执行。这里体现了单线程非阻塞模式的优化，在单线程非阻塞模式中，只有一个线程，它负责处理所有的连接，不仅对连接的处理无法分派到多核上进行，限制了对称多处理(Symmetrical Multi-

Processing,简称SMP)的优势,也会使任务队列堆积过长导致排在尾端的任务长时间得不到响应。本分布式处理系统采用的多线程服务器模型(one loop per thread)能够很好地解决了这一问题。

[0025] 本分布式处理系统采用了自适应负载均衡的策略决定中心调度模块启动多少个线程,以及接收到的新“任务”放入哪一个线程执行。具体的,中心调度模块会实时监控节点的负载,当CPU占用率和内存占用率过高时,说明此时并发的线程太多,中心调度模块将随机选择线程,在其任务处理结束之后关闭它们以减少节点服务器的任务并发量;反之,当CPU占用率低于过低时,中心调度模块会初始化新的线程处理连接。中心调度模块通过最小连接调度算法将连接分派到具有最少连接的线程上去,分派的方式是将连接包装成任务投递到的任务队列中。常用的负载均衡算法有轮询调度法、带权值的轮询调度法、最小连接调度法、散列调度法等,在这里任务连接数可以近似表示线程的“工作量”。

[0026] 传输模块负责对等节点的数据传输,包括接收来自上游节点的输入数据流和向下游节点推送的输出数据流,传输模块相当于分布式处理系统的路由器,传输模块使节点的数据传输和上层应用逻辑完全分离,对上层屏蔽了数据传输的细节,而路由则完全根据Zookeeper服务器的mode结构目录。

[0027] 同样的,为了管理多个I/O数据源,传输模块使用了单线程非阻塞的多路复用模型。一个节点在整个框架中时而扮演客户端的角色,时而扮演服务器端的角色,但其读写都是通过epoll事件循环统一调度的,这是传输控制的核心所在。很多流式计算系统不支持断点续传的功能,因为假设了数据流一次性通过系统,当节点故障或者计算逻辑出错时,只能让原始的数据流重新通过系统再执行一次计算,不仅浪费了资源,特别是在数据流有唯一性且不可重现时,最后的输出结果将不包含这部分数据流,系统的准确性大大降低。本分布式计算系统针对这一弊端在框架节点中加入持久化模块用以支持断点续传、差错重传的功能。这就要求参与数据传送的双方在正式传送数据流前进行一次协议,下游节点通知上游节点前一次数据流传送终止的位置。数据传输模块根据epoll异步读写的特点,实现一个状态机,一个简单的类似TCP/IP的数据传输协议,为数据的断点续传提供支持,保证数据的准确性和真实性。

[0028] 在一般意义上的C/S模式中,由客户端向服务器端发起资源请求,服务器端响应建立TCP连接而后开始向客户端传输数据;但在实时流计算框架中,上游数据流的传送并不是由下游节点的请求来驱动,因为数据流是不间断、流速波动的数据集,下游节点无法预知上游数据流到来的时间点,不仅如此,还会因为对下游节点请求的确认和解析增加系统延时,所以本分布式处理系统数据传输的方式是上游向下游“推送”而不是下游向上游“拉动”,即客户端(上游节点)向服务器(下游节点)请求发送数据,双方通过自定义的协议确定数据的内容后,由客户端主动推送数据,最大程度地减小了延时。

[0029] 具体的,每个线程在初始化时启动数据传输模块,当线程的任务队列中有中心调度模块投递的任务时,传输模块取出任务中的连接端口,加入到自己的epoll事件循环中。传输模块从连接中读取输入数据流并按输入数据流的名称切分,当属于某个输入数据流的数据项第一次被传输模块接收时,传输模块会建立以输入数据流名命名的有名(FIFO)管道,以“写”标志打开该管道(为了向管道中写数据)并传送输入数据流;同时将输入数据流名通过套接字(socket)发送给中心调度模块,中心调度模块收到输入数据流名后,以“读”

标志打开以该输入数据流名命名的FIFO管道，接收传输模块发送的切分完毕的输入数据流。

[0030] 中心调度模块是框架节点的枢纽，负责将输入流送入业务模块，计算后再以输出数据流的形式回到中心调度模块。中心调度模块会根据节点负载决定启动的任务算子的数量，启动次序是依据任务优先级计算的，优先级的影响因子包括该业务在整个任务中的“重要程度”、节点的运行状况、任务算子的类型等等。

[0031] 中心调度模块会通过Zookeeper服务器实时获取数据流和外部处理任务的订阅关系(一个任务实例可能会依赖于多个输入数据流)，将接收到的数据流放入以任务名命名的链表中。当用很小的流量使CPU满负荷或者用很小的CPU占用率使流量占满网卡时，多线程并不能带来系统性能的提升，反而会带来线程间切换、CPU争用的性能损耗。但当I/O操作和CPU计算在时间上“重叠”时，多线程可以有效地降低系统延时。如果仅靠单线程既要负责从传输模块接收输入数据流，又要负责逻辑计算，必然会因为耗时的I/O操作使处理任务等待，CPU闲置，增加响应延迟。因此本分布式处理系统用内部线程池中的多工作线程执行任务算子。中心调度模块针对所有处于就绪状态的任务算子维护一个优先级队列，每当优先级重排后，根据线程池的容量取出相应数量的任务算子执行。

[0032] 为了保证节点的负载稳定，线程池中工作线程的数量根据需要根据节点负载自适应调整，即：如果线程池中的线程在执行任务时，密集计算所占的时间比重为P(0<P<=1)，而系统一共有C个CPU，为了让这C个CPU均参与执行而又不过载，线程池大小的经验公式T=[C/P]。对这一策略进行边界条件验证：假设C=8，P=1.0，线程池的任务完全是密集计算，则T=8。因为8个活动线程就能让8个CPU饱和，CPU资源已经用尽，更多的线程数量并不能提高效率。假设C=8，P=0.5，线程池的任务一半是CPU密集型，一半是I/O密集型，则T=16。考虑操作系统能灵活合理地调度睡眠(sleeping)/写磁盘(writing)/执行(running)线程，那么大概16个“50%繁忙的线程”能让8个CPU满负荷运转，启动更多的线程并不能提高吞吐量，反而因为增加上下文切换的开销而降低性能。T可以取一个固定值，比如5*C。

[0033] 为了增加框架的健壮性，在节点中添加了数据持久化模块，该模块充当磁盘队列的角色，它遵循队列先进先出(FIFO)的语义。该模块把处理完毕的输出数据流保存在磁盘上。为了避免磁盘容量膨胀，持久化模块赋给每条数据项一定的过期时间并定期将它们从磁盘上删除。提供数据化服务使本分布式处理系统有别于其他流式计算系统：

首先，解决了上下游节点处理数据流速度不匹配的问题。特别是当上游节点的发送速度远大于下游节点的处理速度时，会有大量的数据包在上游节点内核的socket缓冲区中堆积而无法发送，导致内存占用膨胀而使系统运行缓慢。数据持久化模块在上下游节点中间做了一层缓冲，由于上游节点发送的数据总是从持久化模块中取得，所以在该模块中引入流量控制机制可以解决这一问题。

[0034] 其次，提供了断点续传和差错重传的功能。在已有的数据流计算系统中，数据流一次性经过系统内存，数据无法恢复和重现，所以系统对故障非常敏感，容错性较差。本分布式处理系统节点在传输模块中实现了定位传输协议用以定位数据流曾经的传输位置，并从持久化模块中取出相应位置的数据实现数据流的恢复。

[0035] 类似于传输模块与中心调度模块协作完成输入数据流传输的方式，数据持久化模块长期监听一个指定的端口并加入到epoll句柄中，当接收到中心调度模块发送的输出数

据流名时，以“读”标志打开FIFO管道并将管道文件描述符加入到epoll循环中。数据持久化模块从管道中持续读取处理完毕的输出数据流，存入以输出数据流名命名的磁盘队列中。磁盘队列中的数据项以key-value形式储存，key值是数据项的时间戳。这样方便差错重传时从硬盘队列中快速定位数据项。

[0036] 数据持久化模块通过开源软件Tokyo Cabinet实现了高效的磁盘队列。Tokyo Cabinet(简称TC)是一个用C语言编写的数据存储引擎，以key-value的方式存储数据，支持Hash, B+ tree, Hash Table等多种数据结构且读写速度极快。

[0037] 最后，数据持久化模块通过查询Zookeeper服务器上输出数据流与下游节点的订阅关系，将已经在磁盘队列中备份好的数据发送到以下游节点(node)命名的共享内存队列中，为了让传输模块在接收和发送数据时呈现一致的行为，数据持久化模块将下游节点名封装成任务放到任务队列中，传输模块通过解析任务类型为“内部”(从上游节点接收数据为“外部”任务，从共享队列取数据是“内部”任务)，取出任务中的下游节点名，从相应的共享内存队列中取出数据，并根据Zookeeper中该下游节点的配置(IP)发送数据。共享消息队列也是类unix系统中进程间通信(IPC)的一种方式，多个进程可以读取或添加队列中的消息。共享消息队列是随内核持续的消息链表，其中的消息具有规定的格式、特定的类型和相应的优先级。

[0038] 本发明实施例中，上述这四个模块之间通过显式的进程间通信方法(Inter-Process Communication，简称IPC)进行信息的共享和复用，通过相互协作共同构成了分布式计算系统的节点。

[0039] 以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应以所述权利要求的保护范围为准。

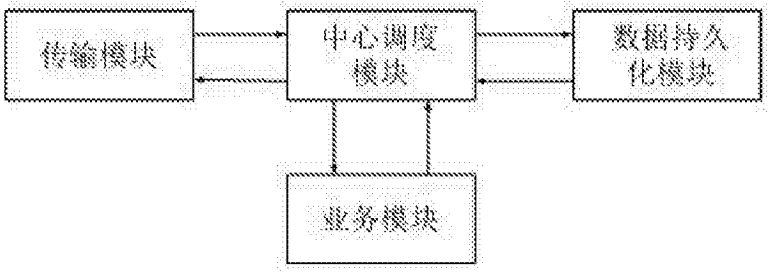


图1