[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 13/18 (2006.01)
G06F 11/30 (2006.01)
G06F 9/445 (2006.01)

[12] 发明专利说明书

专利号 ZL 02159492.9

[45] 授权公告日 2006 年 4 月 12 日

[11] 授权公告号 CN 1251103C

[22] 申请日 2002.12.31 [21] 申请号 02159492.9

[71] 专利权人 联想（北京）有限公司
地址 100085 北京市海淀区上地信息产业基地创业路 6 号
[72] 发明人 李电森 冯 锐 黄 平 肖利民
审查员 郑 红

[74] 专利代理机构 北京同立钧成知识产权代理有限公司
代理人 余 丽 刘 芳

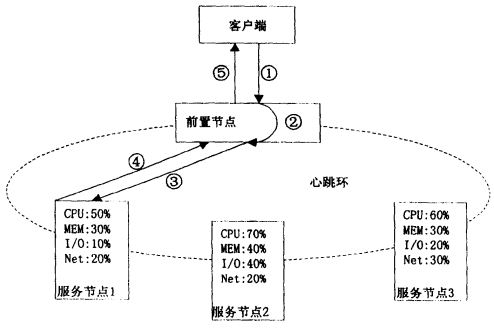
权利要求书 2 页 说明书 8 页 附图 4 页

[54] 发明名称

提高商务机群可服务性的方法

[57] 摘要

一种提高商务机群可服务性的方法，客户端向前置节点发送其服务请求；服务节点通过心跳环将本节点负载信息传递给前置节点，前置节点根据配置信息和负载信息决定出服务节点当前处理能力，并依次对其优先级排序，按照排序分配负载；前置节点定期接收整个心跳环的负载信息，控制客户端服务请求分发和心跳环重构；如果前置节点在规定时间内未能采集到发生故障的服务节点的心跳信息，则重构整个心跳环，并把故障服务节点排除在新的心跳环之外，不向该故障服务节点分发客户端的服务请求；当发生故障的服务节点恢复后，重新申请加入心跳环。 本发明为服务节点提供了双重保障；实现了负载的动态均衡；充分利用服务节点的处理能力，为用户提供高可用的服务。



1、一种提高商务机群可服务性的方法，其特征在于：

客户端与设有节点服务监控器模块的前置节点通信，向所述前置节点发送其服务请求；各设有所述节点服务监控器模块的服务节点通过由各服务节点与前置节点一起构成的心跳环将本节点的负载信息传递给前置节点，前置节点根据各服务节点的配置信息和收到的负载信息决定出这些服务节点的当前处理能力，并依次对这些服务节点的当前处理能力的优先级进行排序，在客户端服务请求到达时，按照该排序分配负载；

所述的前置节点定期接收整个心跳环的负载信息，并根据该负载信息控制客户端服务请求的分发和心跳环的重构；

如果前置节点在规定的时间内未能采集到发生故障的服务节点的心跳信息，则重构整个心跳环，并把故障服务节点排除在新的心跳环之外，不向该故障服务节点分发客户端的服务请求；

当发生故障的服务节点恢复之后，可重新申请加入心跳环。

2、根据权利要求 1 所述的提高商务机群可服务性的方法，其特征在于：所述的各服务节点中设有代理模块，用于搜集本节点的负载信息，并将该负载信息通过心跳环发送给前置节点。

3、根据权利要求 1 或 2 所述的提高商务机群可服务性的方法，其特征在于：所述服务节点中设有的节点服务监控器模块用于监控商务机群系统的进程、服务以及资源的状态，其监控商务机群系统的进程、服务以及资源的状态的流程具体如下：

步骤 A：服务节点的节点服务监控器模块获取服务节点的配置信息，启动服务节点的相关进程和服务；

步骤 B：服务节点的节点服务监控器模块监控服务节点的资源、进程、服务状态；

步骤 C：服务节点的节点服务监控器模块判断服务节点的资源是否达到

资源利用率水位线，如果没有达到则执行步骤 E;

步骤 D: 向前置节点发送预警信息; 执行步骤 B;

步骤 E: 服务节点的节点服务监控器模块判断服务节点进程、服务是否异常，不异常执行步骤 B;

步骤 F: 重启服务节点进程、服务;

步骤 G: 判断重启是否成功，若成功执行步骤 B;

步骤 H: 向前置节点发送故障信息，重启商务机群系统; 结束。

4、根据权利要求 1 或 2 所述的提高商务机群可服务性的方法，其特征在于: 所述的服务节点还设有命令执行器模块，用于和服务节点中的节点服务监控器模块互相监控，实现服务节点的高可靠运行。

5、根据权利要求 4 所述的提高商务机群可服务性的方法，其特征在于: 所述服务节点中的节点服务监控器模块监控服务节点中的命令执行器模块的流程具体如下:

步骤 10: 商务机群系统初始化，启动包括命令执行器模块在内的相关进程;

步骤 11: 进行服务节点的正常应用服务事务处理;

步骤 12: 监控命令执行器模块的状态，如果正常则继续执行步骤 12;

步骤 13: 重新启动该命令执行器模块;

步骤 14: 进行服务节点的其他应用服务事务处理。

6、根据权利要求 4 所述的提高商务机群可服务性的方法，其特征在于: 所述的服务节点中命令执行器模块监控服务节点中节点服务监控器模块的流程如下:

步骤 20: 服务节点系统初始化，启动包括节点服务监控器模块在内的相关进程;

步骤 21: 进行服务节点的正常应用服务事务处理;

步骤 22: 监控节点服务监控器模块的状态，如果正常则继续执行步骤 22;

步骤 23: 重新启动该节点服务监控器模块;

步骤 24: 进行服务节点的其他事务处理。

提高商务机群可服务性的方法

技术领域

本发明涉及一种提高商务机群可服务性的方法，尤其是一种在计算机集群系统中的服务节点上对系统状态进行监控，实现负载的动态均衡的系统及其方法；属于计算机网络技术领域。

背景技术

在计算机集群环境中，监控节点负责收集各个节点的资源、进程及服务的状态，并根据特定的指令执行启停进程或服务的操作。通常，监控节点只是简单地搜集各个节点在系统中进程的状态，并根据控制节点的指令来启动或停止相应的进程。

实际上，人们主要的目标是使用所有的服务节点提供高可用的服务。所谓服务的可用性是指：利用软/硬件资源为用户提供服务的能力；高可用的服务必须是可靠、稳定的，而且响应时间为用户所能接受。而进程状态不是系统服务可用性的唯一决定因素，运行正常的服务节点可能由于某种资源的匮乏而不能提供某种服务。为了避免这种资源匮乏的现象出现，需要对服务节点进行负载均衡。

由于各种因素的限制，现有的负载均衡策略都比较简单。例如：轮询法只是循环地把客户端的请求依次转发到各个服务节点上；最少连接数法优先把客户端请求发送到连接数最少的服务节点上；等等。实际上，这些方法并不能实现真正的负载均衡；其主要原因是：1）每个服务的连接请求各不相同，有些可能需要消耗大量的资源，连接数目并不能真实地反应服务节点的负载情况；2）服务节点的配置各不相同，由于集群良好的可

扩展性，后来扩充的服务节点的资源配备可能比原来的服务节点要好得多；3）每个服务节点中运行的任务各不相同，有些节点可能会提供多个服务，或者运行了大量的进程，资源消耗情况千差万别，而且会随时变换。因此，如果想要实现真正的负载均衡，就必须实时地监控服务节点的状态，根据服务节点的现有处理能力决定如何分发负载；另外，一个服务往往是由一组相关进程共同提供的，这些进程之间通常都存在一定的依赖关系，一旦某个进程出现问题，整个服务将都会受到影响。

在某些环境中，各个服务节点上进程、服务和资源的监控由一个中心控制节点实现，这样处理的优点是控制集中；但是，由于发生故障的服务节点往往不仅仅是一个进程出现问题，而相应的资源往往也会出现匮乏的情况，很可能不能正确处理中心控制节点的命令，最终导致该节点的故障一直得不到恢复。

发明内容

本发明的主要目的在于提供一种提高商务机群可服务性的方法，保障算机集群系统运行稳定可靠、服务高可用，并且具有故障自我恢复能力。

本发明的又一目的在于提供一种提高商务机群可服务性的方法，在所有节点之间实现负载均衡。

本发明的再一目的在于提供一种提高商务机群可服务性的方法，对系统资源进行监控，在资源负荷达到指定的水位线时预警并及时处理，从而最小化系统崩溃的风险。

本发明的目的是这样实现的：

客户端与设有节点服务监控器模块的前置节点通信，向所述前置节点发送其服务请求；各设有所述节点服务监控器模块的服务节点通过由各服务节点与前置节点一起构成的心跳环将本节点的负载信息传递给前置节点，前置节点根据各服务节点的配置信息和收到的负载信息决定出这些服

务节点的当前处理能力，并依次对这些服务节点的当前处理能力的优先级进行排序，在客户端服务请求到达时，按照该排序分配负载；

所述的前置节点定期接收整个心跳环的负载信息，并根据该负载信息控制客户端服务请求的分发和心跳环的重构；

如果前置节点在规定的时间内未能采集到发生故障的服务节点的心跳信息，则重构整个心跳环，并把故障服务节点排除在新的心跳环之外，不向该故障服务节点分发客户端的服务请求；

当发生故障的服务节点恢复之后，可重新申请加入心跳环。

所述的服务节点和前置节点中均设有节点服务监控器（LifeGuard）模块，用于监控系统的进程、服务以及资源的状态，具体包括：

步骤 A：获取配置信息，启动相关进程和服务；

步骤 B：监控资源、进程、服务状态；

步骤 C：判断系统资源是否达到水位线，如果没有达到则执行步骤 E；

步骤 D：向前置节点发送预警信息；执行步骤 B；

步骤 E：判断进程、服务是否异常，不异常执行步骤 B；

步骤 F：重启进程、服务；

步骤 G：判断重启是否成功，若成功执行步骤 B；

步骤 H：向前置节点发送故障信息，重启系统；结束。

上述的服务节点还设有命令执行器（Executor）模块，用于和 LifeGuard 模块互相监控，实现节点的高可靠运行。

所述的 LifeGuard 模块监控服务节点中 Executor 模块的流程如下：

步骤 10：系统初始化，启动包括 Executor 在内的相关进程；

步骤 11：进行节点的正常事务处理；

步骤 12：监控 Executor 的状态，如果正常则继续执行步骤 12；

步骤 13：重新启动该 Executor；

步骤 14：进行节点的其他事务处理。

所述的 Executor 模块监控服务节点中 LifeGuard 模块的流程如下:

步骤 20: 系统初始化，启动包括 LifeGuard 在内的相关进程;

步骤 21: 进行节点的正常事务处理;

步骤 22: 监控 LifeGuard 的状态，如果正常则继续执行步骤 12;

步骤 23: 重新启动该 LifeGuard;

步骤 24: 进行节点的其他事务处理。

所述的前置节点定期接收整个心跳环的负载信息，并根据该负载信息控制客户端请求的分发和心跳环的重构。

如果前置节点在规定的时间内未能采集到发生故障节点的心跳信息，则重构整个心跳环，并把故障节点排除在新的心跳环之外，不向该节点分发客户端的服务请求。当出现故障的服务节点恢复之后，可重新申请加入心跳环。

本发明使用 LifeGuard 和 Executor 为服务节点提供双重保障，使服务节点具有进程级、应用级以及系统级三种自我恢复能力；使用心跳环传递负载信息，真正实现了负载的动态均衡，并可以避免单个（或部分）服务节点崩溃而导致的服务不可用情况；从而充分利用服务节点的处理能力，为用户提供高可用的服务。

附图说明

图 1 为本发明的计算机集群系统构成示意图;

图 2 为本发明中 LifeGuard 模块监控系统并自动恢复系统的处理流程图;

图 3 为本发明 LifeGuard 模块对 Executor 模块的监控流程图;

图 4 为本发明 Executor 模块对 LifeGuard 模块的监控流程图;

图 5 为本发明一具体实施例中 LifeGuard 模块对系统进程、服务及资源监控流程图。

具体实施方式

以下通过具体实施例和附图对本发明进行详细说明。

采用双重监控的计算机集群系统结构如图 1，该系统由客户端 A、前置节点 B、多个服务节点组成。客户端 A 与前置节点 B 通信，客户端 A 的请求都被发送到前置节点 B 上；采用心跳线将服务节点 1、服务节点 2……和前置节点 B 构成一个心跳环，并在每个服务节点上设置一些代理程序，负责搜集本节点的负载信息，并将其通过心跳环传递给前置节点 B。前置节点 B 根据配置信息和搜集到的负载信息决定出这些服务节点的当前处理能力，依次对服务节点的优先级进行排序，在下次客户端请求到达时就按照这个优先级的高低顺序来分配负载，这样，就达到资源的动态平衡，而且可以充分发挥服务节点硬件的处理能力。

采用心跳环机制，除了能传递服务节点的负载信息之外，还有一个功能就是可以实时地发现服务节点的故障。如果一个（或多个）服务节点发生故障，心跳环就会在故障节点处中断。故障节点的下游节点在一定的周期内如果没有接收到上游节点的心跳环负载信息，就向前置节点汇报自己的上游节点出现故障；前置节点接收到汇报之后，会通知故障节点的上、下游节点，分别修改自己的下、上游节点，从而重构整个心跳环，把故障节点排除在新的心跳环之外，而且不向该节点分发服务请求。如果前置节点超时未接收到整个心跳环的负载信息，则按照所有节点新加入心跳环的方式重新构造并初始化整个心跳环。当故障节点恢复之后，可以重新申请加入心跳环。采用这种机制，前置节点就能及时地掌握所有服务节点的负载，动态地实现负载均衡，同时可以避免由于服务节点故障而导致服务不可用的情况出现。

在每个服务节点上设置两个模块：LifeGuard 模块和 Executor 模块。LifeGuard 模块也部署在前置节点上，负责监控系统的进程、服务以及资源的状态。

不同类型的服务对资源的要求也不同，例如：有些应用需要的计算量相当大，这种应用最关心的资源是 CPU，而有些应用则对 I/O、内存或网络的要求非常苛刻。但是每个服务节点所实际拥有的物理硬件总是有限的，硬件的具体配置情况也不可能适用于所有的应用。极端的情况下，服务节点的大部分资源可能都空闲，但由于另外一种关键资源已经耗光，而导致不能提供服务。因此需要对系统的资源进行监控，并对关键资源设置一个水位线，当系统中该资源的利用率超过水位线之后，就向前置节点报警，前置节点对转发策略进行调整，不再向该服务节点转发这种请求，从而避免耗光该系统的所有资源。

服务的高可用性归根结底来自于服务节点的可用性，而每个服务都是由一系列相关进程提供的，因此保证服务节点上应用服务所依赖的相关进程的可用性是提供高可用性服务的基础。

共同提供服务的进程之间往往存在一定的依赖关系，在启动服务时需要按照特定的次序，先启动依赖进程，再启动本进程，过程如图 2 所示：

步骤 A：获取配置信息，启动相关进程和服务；

步骤 B：监控资源、进程、服务状态；

步骤 C：判断系统资源是否达到水位线，如果没有达到则执行步骤 E；

步骤 D：向前置节点发送预警信息；执行步骤 B；

步骤 E：判断进程、服务是否异常，不异常执行步骤 B；

步骤 F：重启进程、服务；

步骤 G：判断重启是否成功，若成功执行步骤 B；

步骤 H：向前置节点发送故障信息，重启系统；结束。

LifeGuard 已经成为服务节点可用性的保障，如果 LifeGuard 一旦发生故障，整个服务节点的自我恢复能力将不复存在。因此本发明采用另外一个模块 Executor 和 Lifeguard 互相进行监控，一旦对方出现故障，即可将其主动恢复，其监控的具体实现如图 3、图 4 所示。

LifeGuard 模块监控服务节点中 Executor 模块的流程如下：

步骤 10：系统初始化，启动包括 Executor 在内的相关进程；

步骤 11：进行节点的正常事务处理；

步骤 12：监控 Executor 的状态，如果正常则继续执行步骤 12；

步骤 13：重新启动该 Executor；

步骤 14：进行节点的其他事务处理。

Executor 模块监控服务节点中 LifeGuard 模块的流程如下：

步骤 20：系统初始化，启动包括 LifeGuard 在内的相关进程；

步骤 21：进行节点的正常事务处理；

步骤 22：监控 LifeGuard 的状态，如果正常则继续执行步骤 12；

步骤 23：重新启动该 LifeGuard；

步骤 24：进行节点的其他事务处理。

本发明 LifeGuard 模块监控系统并自动恢复系统的流程如图 5 所示：

步骤 101：系统如果收到信号，则执行步骤 120，否则按照顺序从步骤 102 开始执行；

步骤 102：读取节点上运行的服务；

步骤 103：读取需该服务监控的资源；

步骤 104：判断是否有需要监控的资源，若没有执行步骤 130；

步骤 105：检查该资源；

步骤 106：资源评价是否正常，若正常，执行步骤 111；

步骤 107：判断该资源是否达到警戒线，若没有达到，执行步骤 110；

步骤 108：警戒线处理，调节该资源在所有资源中所占的权重；

步骤 109：通知主节点资源到达警戒线；

步骤 110：读取下一个要监控的资源，执行步骤 104；

步骤 111：记录错误，执行步骤 110；

步骤 120：判断是否为 SIGKILL，如果是则结束；

步骤 121：判断是否为 SIGHUP 信号，若不是，执行步骤 102；

步骤 122：重新读取资源水位设置；执行步骤 102；

步骤 130：读取需监控的过程队列；

步骤 131：判断是否有需要监控的进程，若没有，执行步骤 136；

步骤 132：检查该进程；

步骤 133：该进程是否异常，若异常，执行步骤 135；

步骤 134：恢复该进程；

步骤 135：取下一个需监控的进程，执行步骤 131；

步骤 136：判断服务进程是否异常，若无异常，执得步骤 138；

步骤 137：向前置节点报告服务故障；

步骤 138：取本节点的下一个服务；

步骤 139：判断是否还有监控的服务；若有，执行步骤 103；

步骤 140：休眠等待后执行步骤 101。

从以上的流程可以看出：LifeGuard 模块可以实现三个层次上的自我恢复能力：

进程级，LifeGuard 可以监控所有进程的状态，而与服务有关的进程都是既定的，一旦这些进程出现故障，就可以按照事先预定的策略重新启动进程。

应用级，服务程序通常以服务程序或守护进程的形式出现，有时即使这些服务程序依赖的所有进程都正常，也会有服务不可用的情况发生，例如套接字（socket）端口释放后必须经过一定的延时之后才可用。LifeGuard 对应用级程序的处理策略是：启用本地客户端或测试程序对应用程序进行测试，如果确定已经不能提供服务，则通知前置节点暂停分发任务，同时重启服务器程序。

系统级在遇到严重错误时，很多程序变得不可用，此时必须重新启动系统才能正常恢复。LifeGuard 的策略是使用软件狗（softdog）机制，重启并恢复系统。

最后应说明的是：以上实施例仅用以说明本发明而非限制，尽管参照较佳实施例对本发明进行了详细说明，本领域的普通技术人员应当理解，可以对本发明进行修改或者等同替换，而不脱离本发明的精神和范围，其均应涵盖在本发明的权利要求范围当中。

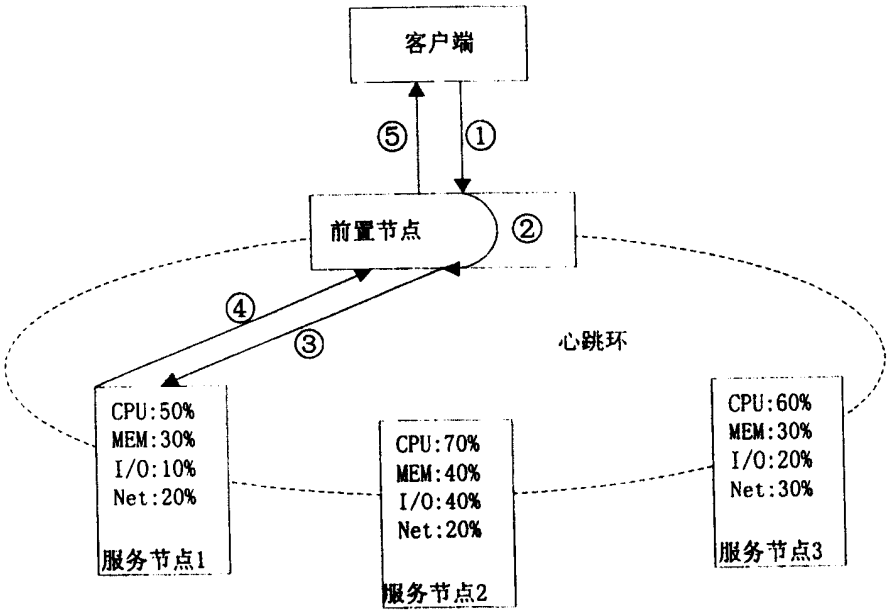

客户端
⑤
①
前置节点
②
④
③
心跳环
CPU:50%
MEM:30%
I/O:10%
Net:20%
服务节点1
CPU:70%
MEM:40%
I/O:40%
Net:20%
服务节点2
CPU:60%
MEM:30%
I/O:20%
Net:30%
服务节点3

图 1

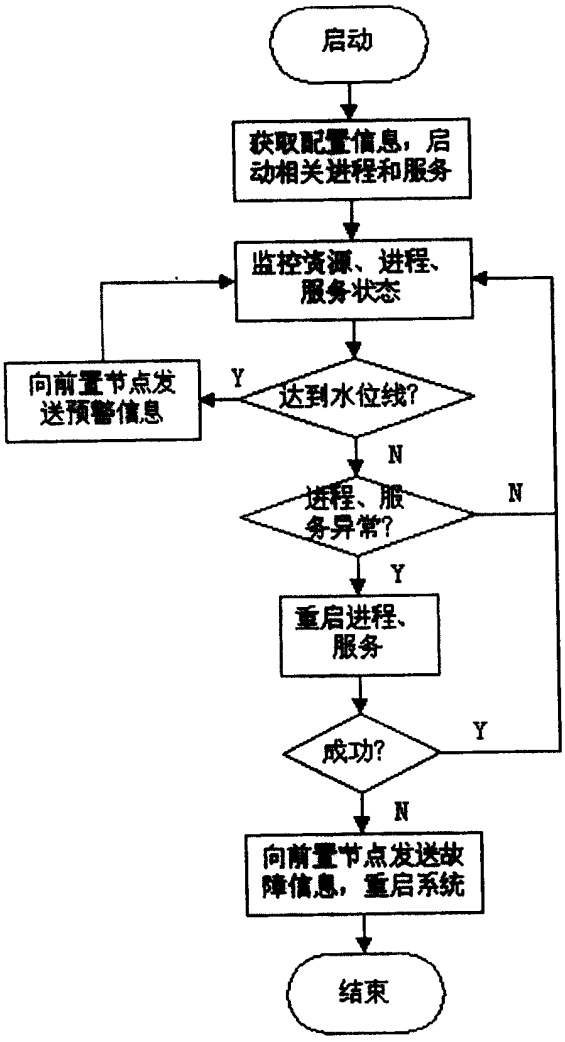

启动
获取配置信息，启动相关进程和服务
监控资源、进程、服务状态
达到水位线?
Y
向前置节点发送预警信息
N
进程、服务异常?
N
Y
重启进程、服务
成功?
Y
N
向前置节点发送故障信息，重启系统
结束

图 2

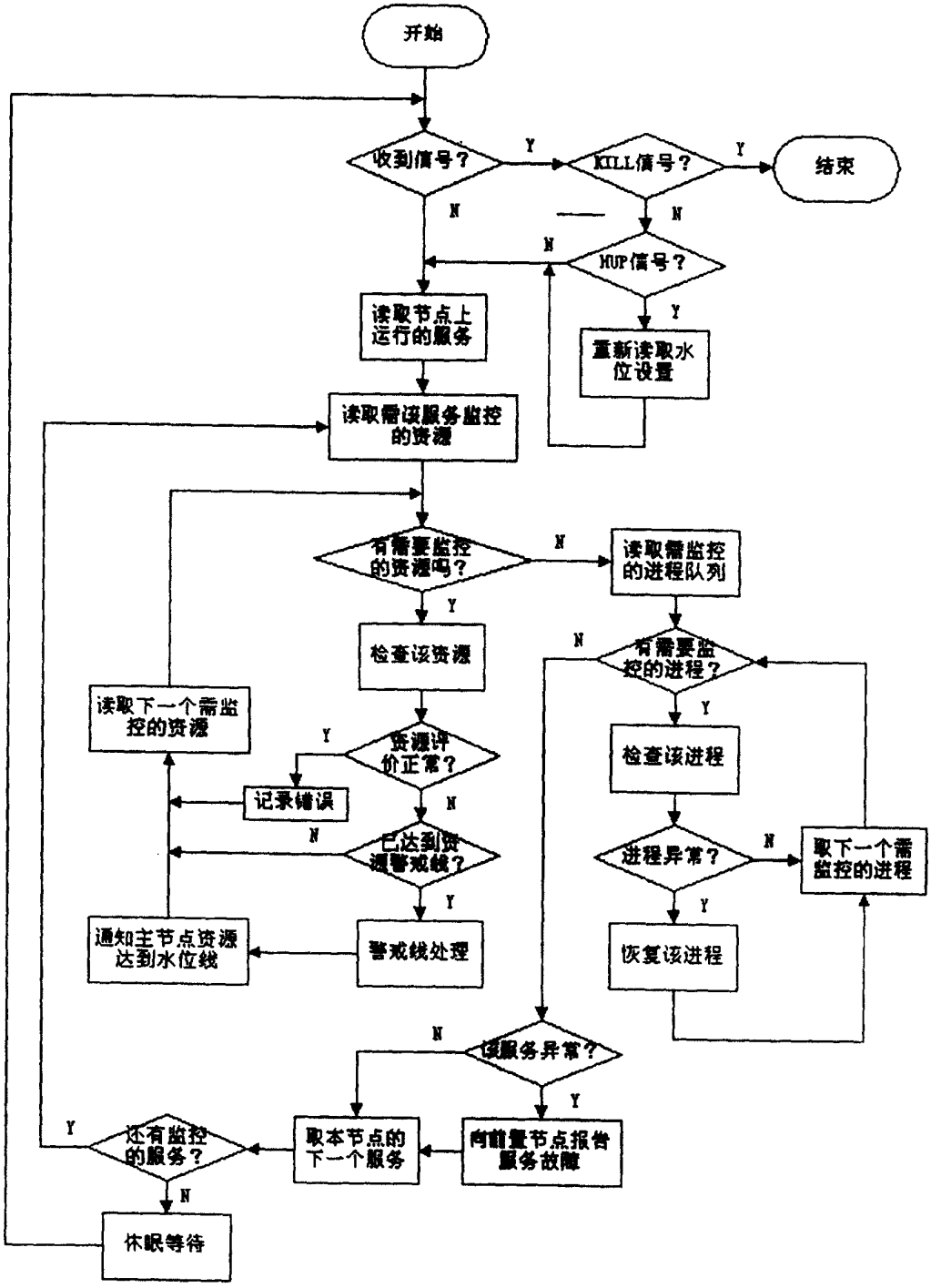

开始
收到信号？
Y
KILL信号？
Y
结束
N
N
N
HUP信号？
Y
读取节点上运行的服务
重新读取水位设置
读取需该服务监控的资源
有需要监控的资源吗？
N
读取需监控的进程队列
Y
检查该资源
有需要监控的进程？
N
读取下一个需监控的资源
Y
资源评价正常？
检查该进程
Y
记录错误
N
进程异常？
N
取下一个需监控的进程
N
已达到资源警戒线？
Y
Y
通知主节点资源达到水位线
警戒线处理
恢复该进程
N
该服务异常？
Y
Y
还有监控的服务？
取本节点的下一个服务
向前置节点报告服务故障
N
休眠等待

图 3

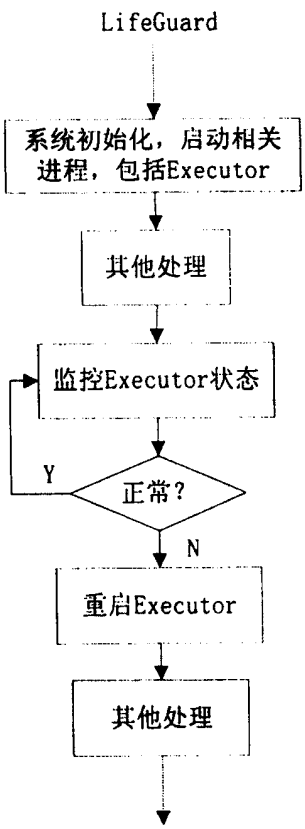

LifeGuard
系统初始化，启动相关进程，包括Executor
其他处理
监控Executor状态
Y
正常？
N
重启Executor
其他处理

图 4

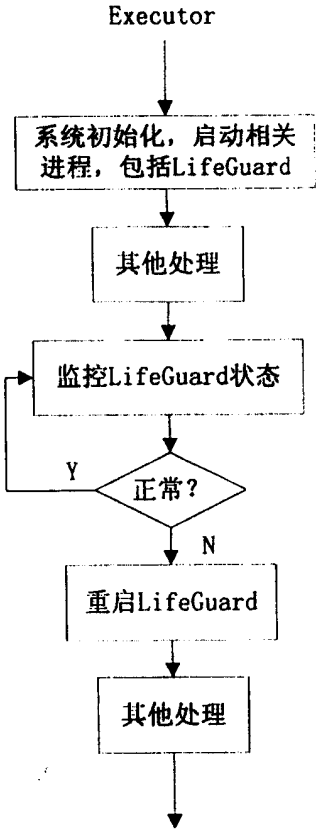

Executor
系统初始化，启动相关进程，包括LifeGuard
其他处理
监控LifeGuard状态
Y
正常？
N
重启LifeGuard
其他处理

图 5