(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115757039 A
(43)申请公布日 2023.03.07

(21)申请号 202211492584.X

(22)申请日 2022.11.25

(71)申请人 惠州市德赛西威智能交通技术研究院有限公司
地址 516006 广东省惠州市惠南高新科技产业园惠泰北路6号

(72)发明人 王碧

(74)专利代理机构 北京品源专利代理有限公司 11332
专利代理师 赵翠香

(51)Int.Cl.
G06F 11/30(2006.01)
G06F 11/14(2006.01)
G06F 9/54(2006.01)

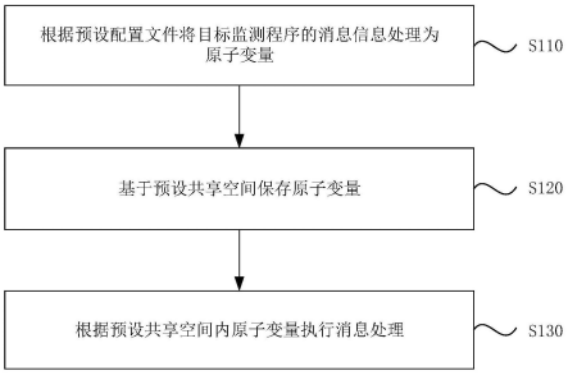
权利要求书2页 说明书12页 附图7页

(54)发明名称

一种程序监控方法、装置、电子设备和存储介质

(57)摘要

本发明公开了一种程序监控方法、装置、电子设备和存储介质。其中，该方法包括：根据预设配置文件将目标监测程序的消息信息处理为原子变量，基于预设共享空间保存所述原子变量，根据所述预设共享空间内所述原子变量执行消息处理。本发明实施例通过原子变量读写实现无锁通讯以及将包含数据和状态的原子变量保存在共享空间中，可以减少系统资源开销，降低通讯时延，可以支持服务进程崩溃重启后，能无损地恢复之前的数据和状态，可靠性更高。



CN 115757039 A

1.一种程序监控方法,其特征在于,所述方法包括:

根据预设配置文件将目标监测程序的消息信息处理为原子变量;

基于预设共享空间保存所述原子变量;

根据所述预设共享空间内所述原子变量执行消息处理。

2.根据权利要求1所述方法,其特征在于,所述根据预设配置文件将目标监测程序的消息信息处理为原子变量,包括:

在所述预设配置文件查找所述消息信息对应的数据标识以及状态标识;

调用预设监测库将所述数据标识和所述状态标识封装为所述原子变量。

3.根据权利要求1或2所述方法,其特征在于,所述原子变量包括以下至少之一:时间标识字段、状态标识字段、序号字段、任务标识字段、校验码字段。

4.根据权利要求1所述方法,其特征在于,所述预设共享空间至少包括高速缓存和共享内存,相应的,所述基于预设共享空间保存所述原子变量,包括:

判断所述预设共享空间内所述高速缓存内是否存在所述原子变量;

若存在,则将所述原子变量存储到所述高速缓存;

若不存在,则将所述原子变量存储到所述共享内存,其中,所述共享内存至少包括心跳监控序列和时间时序监控序列。

5.根据权利要求1所述方法,其特征在于,所述根据所述预设共享空间内所述原子变量执行消息处理,包括:

判断所述预设共享空间内高速缓存是否存在所述原子变量;

若存在,则读取所述高速缓存内所述原子变量直到所述高速缓存为空;

若不存在,则读取所述预设共享空间内共享内存的所述原子变量;

提取所述原子变量的状态标识和数据标识,并执行所述状态标识和所述数据标识对应的所述消息处理。

6.根据权利要求4所述方法,其特征在于,所述原子变量的存储长度与所述高速缓存的缓存行相同。

7.根据权利要求5所述方法,其特征在于,还包括:

按照所述异常处理的处理结果在所述预设共享内存空间更新对应所述原子变量的状态信息。

8.一种程序监控装置,其特征在于,所述装置包括:

变量生成模块,用于根据预设配置文件将目标监测程序的消息信息处理为原子变量;

变量暂存模块,用于基于预设共享空间保存所述原子变量;

消息处理模块,用于根据所述预设共享空间内所述原子变量执行消息处理。

9.一种电子设备,其特征在于,所述电子设备包括:

至少一个处理器;以及

与所述至少一个处理器通信连接的存储器;其中,

所述存储器存储有可被所述至少一个处理器执行的计算机程序,所述计算机程序被所述至少一个处理器执行,以使所述至少一个处理器能够执行权利要求1-7中任一项所述的程序监控方法。

10.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质存储有计算机指

令,所述计算机指令用于使处理器执行时实现权利要求1-7中任一项所述的程序监控方法。

一种程序监控方法、装置、电子设备和存储介质

技术领域

[0001] 本发明涉及计算机应用技术领域，尤其涉及一种程序监控方法、装置、电子设备和存储介质。

背景技术

[0002] 程序监控技术，作为一项汽车软件功能安全开发的重要技术，主要包括以下三部分功能：活跃点检测(监测程序是否正常运行)、截止时间监控(监测程序是否在规定的时间内执行完成)以及逻辑时序监控(监测各程序段是否按照既定的时间顺序运行)。现有的程序监控方法一般由被监测进程周期性地发送心跳数据(如时间、进程号、程序号和状态等)给服务进程，再由服务进程进行预设逻辑判断，判断是否存在活跃点超时错误、截止时间超时错误和逻辑时序错误等问题，并在发生错误时进行相应的错误处理。

[0003] 然而，现有的程序监控方法存在以下问题：

[0004] 1、现有程序监控方法主要使用Unix域套接字(Unix Domain Sockets，UDS)和共享内存(Share Memory，SHMEM)方式进行进程间通讯，上述两种通讯方式均为有锁通讯，会增加系统资源开销，增加通讯时延。

[0005] 2、被监测进程发送的心跳数据本身没有状态，状态数据均在服务进程中保存，这会引起：如果服务进程崩溃退出，会导致被监测进程的状态数据丢失，无法实现热切换和无损恢复，可靠性较低。

发明内容

[0006] 本发明提供了一种程序监控方法、装置、电子设备和存储介质，以实现通过原子变量读写将有锁通讯变更为无锁通讯，可以减少系统资源开销，降低通讯时延，同时通过将包含数据和状态的原子变量保存在共享空间中，可以支持服务进程崩溃重启后，能无损地恢复之前的数据和状态，可靠性更高。

[0007] 根据本发明的一方面，提供了一种程序监控方法，其中，该方法包括：

[0008] 根据预设配置文件将目标监测程序的消息信息处理为原子变量；

[0009] 基于预设共享空间保存原子变量；

[0010] 根据预设共享空间内原子变量执行消息处理。

[0011] 根据本发明的另一方面，提供了一种程序监控生成装置，包括：

[0012] 变量生成模块，用于根据预设配置文件将目标监测程序的消息信息处理为原子变量；

[0013] 变量暂存模块，用于基于预设共享空间保存原子变量；

[0014] 消息处理模块，用于根据预设共享空间内原子变量执行消息处理。

[0015] 根据本发明的另一方面，提供了一种电子设备，所述电子设备包括：

[0016] 至少一个处理器；以及

[0017] 与所述至少一个处理器通信连接的存储器；其中，

[0018] 所述存储器存储有可被所述至少一个处理器执行的计算机程序，所述计算机程序被所述至少一个处理器执行，以使所述至少一个处理器能够执行本发明任一实施例所述的程序监控方法。

[0019] 根据本发明的另一方面，提供了一种计算机可读存储介质，所述计算机可读存储介质存储有计算机指令，所述计算机指令用于使处理器执行时实现本发明任一实施例所述的程序监控方法。

[0020] 本发明实施例的技术方案，通过根据预设配置文件将目标监测程序的消息信息处理为原子变量，基于预设共享空间保存原子变量，根据预设共享空间内原子变量执行消息处理。本发明实施例通过原子变量读写实现无锁通讯以及将包含数据和状态的原子变量保存在共享空间中，可以减少系统资源开销，降低通讯时延，可以支持服务进程崩溃重启后，能无损地恢复之前的数据和状态，可靠性更高。

[0021] 应当理解，本部分所描述的内容并非旨在标识本发明的实施例的关键或重要特征，也不用于限制本发明的范围。本发明的其它特征将通过以下的说明书而变得容易理解。

附图说明

[0022] 为了更清楚地说明本发明实施例中的技术方案，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。

[0023] 图1是根据本发明实施例一提供的一种程序监控方法的流程图；

[0024] 图2是根据本发明实施例二提供的一种程序监控方法的流程图；

[0025] 图3是根据本发明实施例三提供的一种程序监控方法的流程图；

[0026] 图4是根据本发明实施例三所适用的程序监控系统的结构示例图；

[0027] 图5是根据本发明实施例三提供的原子变量的数据结构示例图；

[0028] 图6是根据本发明实施例三提供的数据状态处理的流程示例图；

[0029] 图7是根据本发明实施例三提供的CPU Cache伪共享的示例图；

[0030] 图8是根据本发明实施例三提供的缓存优化的流程示例图；

[0031] 图9是根据本发明实施例四提供的一种程序监控装置的结构示意图；

[0032] 图10是实现本发明实施例的程序监控方法的电子设备的结构示意图。

具体实施方式

[0033] 为了使本技术领域的人员更好地理解本发明方案，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，所描述的实施例仅仅是本发明一部分的实施例，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，都应当属于本发明保护的范围。

[0034] 需要说明的是，本发明的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换，以便这里描述的本发明的实施例能够以除了在这里图示或

描述的那些以外的顺序实施。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含，例如，包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0035] 实施例一

[0036] 图1为本发明实施例一提供的一种程序监控方法的流程图，本实施例可适用于对目标监测程序进行监控的情况，该方法可以由程序监控装置来执行，该程序监控装置可以采用硬件和/或软件的形式实现，该程序监控装置可配置于电子设备中，例如，电子设备可以包括车载设备、移动设备等。如图1所示，本实施例一提供的一种程序监控方法，具体包括如下步骤：

[0037] S110、根据预设配置文件将目标监测程序的消息信息处理为原子变量。

[0038] 在本发明实施例中，预设配置文件可以理解为针对目标监测程序配置的文件，预设配置文件可以包括任务号、程序号、程序的前后依赖关系、程序本身的耗时、整个任务的耗时和任务的心跳时间等参数。目标监测程序可以理解为待监测的程序段，目标监测程序可以包括待监测任务的一个或多个程序段。消息信息可以理解为包含目标监测程序的监测数据和监控状态的消息，消息信息可以包括目标监测程序的运行时间、任务号、程序号、超时错误状态和时序错误状态等。原子变量可以理解为在多线程无锁通信中的一种整型数据变量，对原子变量进行的操作是一个原子操作，原子操作指的是不会被线程调度机制打断的操作，这种操作一旦开始，在执行完毕前不会被任何其他任务或事件打断，通过原子变量操作可以使有锁通信变为无锁通信，降低通讯时延。

[0039] 具体地，可以根据预设配置文件中的配置项去确定目标监测程序的消息信息，预设配置文件中可以包括但不限于以下几项参数信息之一：任务号、程序号、程序的前后依赖关系、程序本身的耗时、整个任务的耗时、任务的心跳时间，并将消息信息处理成原子变量，处理过程可以包括但不限于以下几种方式：通过调用一些应用程序界面(Application Program Interface，API)接口函数库将消息信息封装成对应的原子变量、通过调用一些自定义的或者第三方的原子变量操作工具包将消息信息封装成对应的原子变量等。在一个实施例中，可以启动加载包含任务号、程序号和程序的前后依赖关系等参数的预设配置文件，以及运行目标监测程序，可以在目标监测程序运行完毕或者每间隔预设时间将包含该程序的监测数据和监控状态的消息信息，通过一些API接口函数库或原子变量操作工具包将消息信息封装成对应的原子变量。

[0040] S120、基于预设共享空间保存原子变量。

[0041] 在本发明实施例中，预设共享空间可以理解为用于存储原子变量的数据存储空间，预设共享空间可以至少包括高速缓存和共享内存。

[0042] 具体地，在目标监测程序的消息信息处理为原子变量后，可以将原子变量保存到预设共享空间的相应位置，预设共享空间可以至少包括高速缓存和共享内存，在编译期间可以为原子变量分配指针或者内存地址，可以根据原子变量的数据类型对原子变量进行保存，示例性地，可以将原子变量以其原码或补码的形式存储在共享内存中。在一个实施例中，预设共享空间可以包括高速缓存和共享内存，在对原子变量进行保存时，若高速缓存中还存在该原子变量，则可以优先将原子变量保存在高速缓存中；若高速缓存中无该原子变

量,则可以将原子变量保存到共享内存中。

[0043] S130、根据预设共享空间内原子变量执行消息处理。

[0044] 在本发明实施例中,消息处理可以理解为针对目标监测程序的消息信息的处理方法,消息处理可以包括消息读取、超时错误处理、时序错误处理和数据状态更新等。

[0045] 具体地,可以对保存在预设共享空间内的原子变量进行一些消息处理,消息处理过程可以包括但不限于几种:读取预设共享空间内的原子变量,从中解析出目标监测程序的状态信息;判断该程序是否出现超时错误或者时序错误,若出现错误信息,可以将对应的故障码和故障原因等故障信息上报给后台管理人员或者功能安全管理模块进行程序错误状态处理;在程序错误状态处理后,清除当前目标监测程序的错误状态,并对预设共享空间内原子变量进行更新。

[0046] 本发明实施例的技术方案,通过根据预设配置文件将目标监测程序的消息信息处理为原子变量,基于预设共享空间保存原子变量,根据预设共享空间内原子变量执行消息处理。本发明实施例通过原子变量读写实现无锁通讯以及将包含数据和状态的原子变量保存在共享空间中,可以减少系统资源开销,降低通讯时延,可以支持服务进程崩溃重启后,能无损地恢复之前的数据和状态,可靠性更高。

[0047] 进一步地,在上述发明实施例的基础上,所述原子变量包括以下至少之一:时间标识字段、状态标识字段、序号字段、任务标识字段、校验码字段。

[0048] 在本发明实施例中,时间标识字段可以理解为用于表征目标监测程序的运行时间的字段,时间标识字段可以从0开始计算,单位可以为10微秒。状态标识字段可以理解为用于表征目标监测程序状态信息的字段,状态标识字段可以包括有无存在乱序错误、有无存在超时错误以及是否处于激活状态等。序号字段可以理解为用于表征程序号的字段。任务标识字段可以理解为用于表征任务号的字段。校验码字段可以理解为用于表征合法性校验的字段,校验码类型可以包括奇偶校验码、海明校验码和循环冗余校验码等。

[0049] 实施例二

[0050] 图2为本发明实施例二提供的一种程序监控方法的流程图,基于上述实施方式进一步进行优化与扩展,并可以与上述实施方式中各个可选技术方案结合。如图2所示,本实施例二提供的一种程序监控方法,具体包括如下步骤:

[0051] S210、在预设配置文件查找消息信息对应的数据标识以及状态标识。

[0052] 在本发明实施例中,数据标识可以理解为消息信息包含的数据信息,数据标识可以包括程序号、任务号和校验码等信息。状态标识可以理解为消息信息包含的状态信息,状态标识可以包括时序错误状态、超时错误状态、开始状态和激活状态等。

[0053] 具体地,可以根据预设配置文件内的任务号和程序号等配置信息,查找与目标监测程序的消息信息对应的数据标识和状态标识,其中,数据表示可以包括但不限于:程序号、任务号、校验码;状态表示可以包括但不限于:时序错误状态、超时错误状态、开始状态、激活状态。

[0054] S220、调用预设监测库将数据标识和状态标识封装为原子变量。

[0055] 在本发明实施例中,预设监测库可以理解为用于实现原子变量封装的API接口函数库,预设监测库可以包括C++提供的atomic类原子操作API函数以及等。

[0056] 具体地,可以通过调用预设监测库将包含数据标识和状态标识的消息信息封装成

原子变量，预设监测库可以包括但不限于：C++提供的atomic类原子操作API函数、Java提供的java.util.concurrent.atomic类原子操作API函数、自定义的原子变量封装库，进一步地，可以根据实际需要，对数据标识和状态标识分配不同的Bit位和Bit大小，并利用atomic_int64_t等API函数将数据标识和状态标识封装成对应数据类型的原子变量。

[0057] 进一步地，在上述发明实施例的基础上，所述预设共享空间至少包括高速缓存和共享内存。

[0058] 在本发明实施例中，高速缓存可以指CPU缓存(Cache)，是位于CPU和主内存之间的一种容量较小但速度很快的存储器，由于CPU的速度远高于主内存，CPU直接从内存中存取数据要等待一定时间周期，Cache中保存着CPU刚用过或循环使用的一部分数据，当CPU再次使用该部分数据时可从Cache中直接调用，减少CPU的等待时间，提高了数据读写的效率。共享内存可以理解为在进程间通信中能够被不同进程访问的大容量内存，即不同进程可以通过访问同一块内存区域(即共享内存)实现数据共享和交互。每个进程可以将自身的虚拟地址映射到物理内存中的特定区域，当不同进程将相同的物理内存区域与各自的虚拟地址空间关联时，这些进程就能实现通过共享内存来完成进程间通信。若某进程更改了共享内存区的内容，其他进程都会觉察到该区域的更改。

[0059] S230、判断预设共享空间内高速缓存内是否存在原子变量。

[0060] 具体地，可以根据原子变量对应的指针或者内存地址，采用直接映射、全相连Cache和组相连Cache等方式，通过缓存行中对应存储数据的有效位是否为0或者1，判断高速缓存是否命中，进而确定高速缓存内是否存在原子变量。

[0061] S240、若存在，则将原子变量存储到高速缓存；若不存在，则将原子变量存储到共享内存，其中，共享内存至少包括心跳监控序列和时间时序监控序列。

[0062] 在本发明实施例中，心跳监控序列理解为用于存储包含目标监测程序心跳数据的原子变量，心跳监控序列可以包含一个或多个表征目标监测程序心跳数据的原子变量。时间时序监控序列可以理解为用于存储包含目标监测程序时间时序数据的原子变量，时间时序监控序列可以包含一个或多个表征目标监测程序时间时序数据的原子变量。

[0063] 具体地，若预设共享空间内高速缓存内存在原子变量，则可以根据原子变量的对应的指针或者内存地址，采用直接映射、全相连Cache和组相连Cache等方式，将原子变量存储到高速缓存中；若不存在，则可以根据原子变量对应的指针或者内存地址，将原子变量存储到共享内存内的相应存储位置，其中，共享内存至少包括心跳监控序列和时间时序监控序列；进一步地，心跳监控序列和时间时序监控序列可以分别位于两条独立的消息队列中，消息队列类型可以包括ActiveMQ消息队列、RabbitMQ消息队列、ZeroMQ消息队列和Kafka消息队列等，本发明实施对此不进行限制。

[0064] S250、判断预设共享空间内高速缓存是否存在原子变量。

[0065] S260、若存在，则读取高速缓存内原子变量直到高速缓存为空；若不存在，则读取预设共享空间内共享内存的原子变量。

[0066] 具体地，若预设共享空间内高速缓存内存在原子变量，则可以根据原子变量对应的指针或者内存地址，优先读取高速缓存内的原子变量；若不存在，则可以读取共享内存内的原子变量。

[0067] S270、提取原子变量的状态标识和数据标识，并执行状态标识和数据标识对应的

消息处理。

[0068] 具体地，可以从读取到的原子变量中提取出任务号、程序号和校验码等数据标识，以及时序错误状态和超时错误状态等状态标识，具体提取方式可以包括但不限于：根据原子变量的封装原理对原子变量中的状态标识和数据标识进行提取、对原子变量包含的字段信息进行解析获取，进而根据状态标识和数据标识判断当前目标监测程序的状态，若出现超时错误或者时序错误，可以将故障码和故障原因等故障信息上报给后台管理人员或者功能安全管理模块进行程序错误状态处理，并在程序错误状态处理后，清除当前目标监测程序的错误状态，例如可以将原子变量中的表征超时错误或者乱序错误的标记位进行重置。

[0069] 本发明实施例的技术方案，通过在预设配置文件查找消息信息对应的数据标识以及状态标识，调用预设监测库将数据标识和状态标识封装为原子变量，判断预设共享空间内高速缓存内是否存在原子变量，若存在，则将原子变量存储到高速缓存；若不存在，则将原子变量存储到共享内存，其中，共享内存至少包括心跳监控序列和时间时序监控序列，判断预设共享空间内高速缓存是否存在原子变量，若存在，则读取高速缓存内原子变量直到高速缓存为空；若不存在，则读取预设共享空间内共享内存的原子变量，提取原子变量的状态标识和数据标识，并执行状态标识和数据标识对应的消息处理。本发明实施例通过原子变量读写将有锁通讯变更为无锁通讯，可以有效减少系统资源开销和降低通讯时延，同时，消息的原子变量中带有数据和数据状态，即数据和数据状态均保存在共享空间中，可以支持在服务进程崩溃重启后，实现服务的热切换和无损恢复，具有较高的可靠性。

[0070] 进一步地，在上述发明实施例的基础上，所述原子变量的存储长度与高速缓存的缓存行相同。

[0071] 在本发明实施例中，原子变量可以为多种基本变量类型，例如可以包括uint64_t、long、char32_t和uint_least8_t等，每种基本变量类型对应不同的字节数(即存储长度)；高速缓存的缓存行一般是2的整数幂个连续字节；为避免伪共享问题的出现，可以将根据不同平台的高速缓存的缓存行的大小，将原子变量的存储长度扩充至高速缓存的缓存行大小。在一个实施例中，若高速缓存的缓存行大小为64字节，而原子变量使用占用8个字节的uint64_t类型进行表示，为避免因伪共享导致缓存命中率较低的问题，可以在8个字节的原子变量基础上，在其前后各放置7个long类型变量(8字节)，即通过对原子变量填充无意义的变量，来保证整个原子变量独占整个缓存行。

[0072] 进一步地，在上述发明实施例的基础上，还可以包括：

[0073] 按照异常处理的处理结果在预设共享内存空间更新对应原子变量的状态信息。

[0074] 具体地，在处理完当前目标监测程序的超时错误或者时序错误后，可以将预设共享内存空间的对应原子变量的状态信息进行更新，可以将原子变量中的表征超时错误或者乱序错误的标记位重置。

[0075] 实施例三

[0076] 图3为本发明实施例三提供的一种程序监控方法的流程图。本实施例在上述实施例的基础上，提供了一种程序监控方法的一个实施方式，能够基于原子变量读写实现对目标监测程序的监控。图4为本发明实施例三所适用的程序监控系统的结构示例图。

[0077] 如图3所示，本发明实施例三提供的一种程序监控方法，具体包括如下步骤：

[0078] S310、启动被监测进程和监测服务进程，并加载配置文件。

[0079] 在本发明实施例中，配置文件中会配置所有的任务号、程序号、程序的前后依赖关系、程序本身的耗时、整个任务的耗时和任务的心跳时间等参数。具体地，启动被监测进程和监测服务进程后，被监测进程、监测服务进程和监测库会加载所需的配置文件。

[0080] S320、被监测进程上报状态消息，写入到原子变量并保存至共享内存。

[0081] 在本发明实施例中，将每条消息压缩成如图5所示的一个8字节的原子变量，该原子变量包含的消息字段信息如下表所示：

[0082]

消息字段	消息长度	消息内容
时间	40bit Bit24 ~Bit63	从开机以来的运行时间，从 0 开始计算，单位为 10 微秒
状态	Bit23	1：乱序错误

[0083]

		0：无乱序错误
	Bit22	1：超时错误 0：无超时错误
	Bit21	1：已激活(正确收到了 start/stop 消息) 0:未激活（未正确收到 stop 消息）
	Bit20	1：已开始(处于收到 start，未收到 stop 状态) 0:未开始（未处于收到 start，未收到 stop 状态）
序号	4bit Bit16 ~Bit19	程序号：0 ~15 最大支持 16 个程序号。
任务号	8bit Bit8 ~Bit15	任务号：0 ~255，最大支持 256 个任务号。
校验码	8bit Bit0 ~Bit7	CRC8 校验码，校验范围为：Dataid+ Bit8 ~Bit63 Dataid 为 16 位外部通讯识别码，用于合法性校验。

[0084] S330、监测服务进程周期性地读取内存队列，处理其中的数据和状态，并回写状态数据到共享内存。

[0085] 在本发明实施例中，共享内存中有两个独立的消息队列，即心跳监控列表和时间时序监控列表，每个队列内均由原子变量消息集成，原子变量中包括数据和状态。

[0086] 图6为本实施例三提供的数据状态处理的流程示例图。具体数据状态的更新逻辑如下：被监测进程在写消息时，同时更新消息状态，如时序错误状态、超时错误状态、开始状态和激活状态等；监测服务进程在读取消息时，进行逻辑判断和消息状态判断，在处理消息后，同时更新数据的消息状态；这样可以保证，如果数据或状态未处理，则一直保存在内存中，在监测服务进程异常重启后，能无损地恢复之前的数据和状态，实现热切换和无损恢复。

[0087] 本实施例三还提供了一种关于CPU Cache的优化方法。CPU的Cache是以Cache Line(缓存行)为单位进行数据存取的，大多数CPU的Cache Line为64字节，如果一个Cache Line中包含多个数据变量，则在多并发读写场景下，每个变量的新写入都会引起Cache Line中其他变量的失效，造成伪共享问题。图7为CPU Cache伪共享的示例图。为避免伪共享

问题，本实施例提出的CPU Cache的优化方法如下：将每个消息原子变量扩展为64字节，刚好独占一个Cache Line，消除CPU Cache伪共享问题，进而提高通讯性能。图8为本实施例三提供的缓存优化的流程示例图。

[0088] 进一步地，在上述发明实施例的基础上，原子变量只能是基本变量类型，实施例三配置为uint64_t类型(8字节)，也可以替代为4字节或其他平台的大于8字节。主要思想是通过原子变量读写实现无锁通讯。

[0089] 进一步地，在上述发明实施例的基础上，状态数据也可以包括更多的状态定义或不同的Bit位及Bit大小。主要思想是消息体包括数据和状态，以实现热切换和无损恢复

[0090] 进一步地，在上述发明实施例的基础上，CPU缓存的优化可以根据CPU Cache Line的大小来具体扩展变量大小，主要思想是一个CPU的Cache Line中只包含一个变量，消除伪共享问题。

[0091] 下表为使用原子变量实现无锁通讯以及使用有锁通讯的耗时结果示例。

[0092]

序号	锁类型	操作：5 个进程，并发互斥++数据 10 万次	耗 时（秒）	应用场景
1	原子变量	atomic_add_value(&a_count,1)	0.011	简单数据结构

2	原子操作	while(atomicflag.test_and_set());
		g_count++;
		atomicflag.clear();
			0.037	确保能快速获取锁不会长时间等待

3	自旋锁	pthread_spin_lock(&spinlock);
		g_count++;
		pthread_spin_unlock(&spinlock);
			1.915	确保能快速获取锁，不会长时间等待

4	信号量	sem_wait(&sem); g_count++;	1.97	支持加锁、等待休眠

[0093]

		sem_post(&sem);		

5	互斥锁	pthread_mutexlock(&mutexlock);
		g_count++;
		pthread_mutex_unlock(&mutexlock);
			2.105	支持加锁、等待休眠

6	文件读写锁	pthread_rwlock_wrlock(&rwlock);
		g_count++;
		pthread_rwlock_unlock(&rwlock);
			2.236	支持加锁、等待休眠，一写多读场景

[0094] 由上表可以看出，使用原子变量实现无锁通讯的耗时，要比有锁通讯的耗时少得多，进而说明被监测进程与服务进程通过原子操作，对消息进行读写，从有锁通讯变更为无锁通讯，可以降低通讯时延。

[0095] 本发明实施例的技术方案，通过启动被监测进程和监测服务进程，并加载配置文件，被监测进程上报状态消息，写入到原子变量并保存至共享内存，监测服务进程周期性地读取内存队列，处理其中的数据和状态，并回写状态数据到共享内存。本发明实施例通过被监测进程与服务进程采用原子操作对消息进行读写，实现了从有锁通讯变更为无锁通讯，可以有效减少系统资源开销和降低通讯时延；同时，消息的原子变量中带有数据和数据状态并都保存在共享内存中，可以支持在服务进程崩溃重启后，实现热切换和无损恢复，提高通讯的可靠性；此外，通过CPU缓存的优化将原子变量扩充为独占一个缓存行，消除了伪共享问题，提高了CPU缓存的命中率，进一步提高了通讯性能。

[0096] 实施例四

[0097] 图9为本发明实施例四提供的一种程序监控装置的结构示意图。如图9所示，该装置包括：

[0098] 变量生成模块41，用于根据预设配置文件将目标监测程序的消息信息处理为原子变量。

[0099] 变量暂存模块42，用于基于预设共享空间保存原子变量。

[0100] 消息处理模块43，用于根据预设共享空间内原子变量执行消息处理。

[0101] 本发明实施例的技术方案，通过变量生成模块根据预设配置文件将目标监测程序的消息信息处理为原子变量，变量暂存模块基于预设共享空间保存原子变量，消息处理模块根据预设共享空间内原子变量执行消息处理。本发明实施例通过原子变量读写实现无锁通讯以及将包含数据和状态的原子变量保存在共享空间中，可以减少系统资源开销，降低通讯时延，可以支持服务进程崩溃重启后，能无损地恢复之前的数据和状态，可靠性更高。

[0102] 进一步地，在上述发明实施例的基础上，变量生成模块41包括：

[0103] 标识查找单元，用于在预设配置文件查找消息信息对应的数据标识以及状态标识。

[0104] 原子变量封装单元，用于调用预设监测库将数据标识和状态标识封装为原子变量。

[0105] 进一步地，在上述发明实施例的基础上，预设共享空间至少包括高速缓存和共享内存，相应的，变量暂存模块42包括：

[0106] 第一判断单元，用于判断预设共享空间内高速缓存内是否存在原子变量。

[0107] 原子变量存储单元，用于若存在，则将原子变量存储到高速缓存；若不存在，则将原子变量存储到共享内存，其中，共享内存至少包括心跳监控序列和时间时序监控序列。

[0108] 进一步地，在上述发明实施例的基础上，消息处理模块43包括：

[0109] 第二判断单元，用于判断预设共享空间内高速缓存是否存在原子变量。

[0110] 原子变量读取单元，用于若存在，则读取高速缓存内原子变量直到高速缓存为空；若不存在，则读取预设共享空间内共享内存的原子变量。

[0111] 消息处理单元，用于提取原子变量的状态标识和数据标识，并执行状态标识和数据标识对应的消息处理。

[0112] 进一步地，在上述发明实施例的基础上，还包括：

[0113] 状态更新模块，用于按照异常处理的处理结果在预设共享内存空间更新对应原子变量的状态信息。

[0114] 进一步地，在上述发明实施例的基础上，原子变量包括以下至少之一：时间标识字段、状态标识字段、序号字段、任务标识字段、校验码字段。

[0115] 进一步地，在上述发明实施例的基础上，原子变量的存储长度与所述高速缓存的缓存行相同。

[0116] 本发明实施例所提供的程序监控装置可执行本发明任意实施例所提供的程序监控方法，具备执行方法相应的功能模块和有益效果。

[0117] 实施例五

[0118] 图10示出了可以用来实施本发明的实施例的电子设备50的结构示意图。电子设备旨在表示各种形式的数字计算机，诸如，膝上型计算机、台式计算机、工作台、个人数字助理、服务器、刀片式服务器、大型计算机、和其它适合的计算机。电子设备还可以表示各种形式的移动装置，诸如，个人数字处理、蜂窝电话、智能电话、可穿戴设备(如头盔、眼镜、手表等)和其它类似的计算装置。本文所示的部件、它们的连接和关系、以及它们的功能仅仅作为示例，并且不意在限制本文中描述的和/或者要求的本发明的实现。

[0119] 如图10所示，电子设备50包括至少一个处理器51，以及与至少一个处理器51通信连接的存储器，如只读存储器(ROM)52、随机访问存储器(RAM)53等，其中，存储器存储有可被至少一个处理器执行的计算机程序，处理器51可以根据存储在只读存储器(ROM)52中的计算机程序或者从存储单元58加载到随机访问存储器(RAM)53中的计算机程序，来执行各种适当的动作和处理。在RAM 53中，还可存储电子设备50操作所需的各种程序和数据。处理器51、ROM 52以及RAM 53通过总线54彼此相连。输入/输出(I/O)接口55也连接至总线54。

[0120] 电子设备50中的多个部件连接至I/O接口55，包括：输入单元56，例如键盘、鼠标等；输出单元57，例如各种类型的显示器、扬声器等；存储单元58，例如磁盘、光盘等；以及通信单元59，例如网卡、调制解调器、无线通信收发机等。通信单元59允许电子设备50通过诸如因特网的计算机网络和/或各种电信网络与其他设备交换信息/数据。

[0121] 处理器51可以是各种具有处理和计算能力的通用和/或专用处理组件。处理器51的一些示例包括但不限于中央处理单元(CPU)、图形处理单元(GPU)、各种专用的人工智能(AI)计算芯片、各种运行机器学习模型算法的处理器、数字信号处理器(DSP)、以及任何适当的处理器、控制器、微控制器等。处理器51执行上文所描述的各个方法和处理，例如程序监控方法。

[0122] 在一些实施例中，程序监控方法可被实现为计算机程序，其被有形地包含于计算机可读存储介质，例如存储单元58。在一些实施例中，计算机程序的部分或者全部可以经由ROM 52和/或通信单元59而被载入和/或安装到电子设备50上。当计算机程序加载到RAM 53并由处理器51执行时，可以执行上文描述的程序监控方法的一个或多个步骤。备选地，在其他实施例中，处理器51可以通过其他任何适当的方式(例如，借助于固件)而被配置为执行程序监控方法。

[0123] 本文中以上描述的系统和技术的各种实施方式可以在数字电子电路系统、集成电路系统、场可编程门阵列(FPGA)、专用集成电路(ASIC)、专用标准产品(ASSP)、芯片上系统的系统(SOC)、负载可编程逻辑设备(CPLD)、计算机硬件、固件、软件、和/或它们的组合中实现。这些各种实施方式可以包括：实施在一个或者多个计算机程序中，该一个或者多个计算机程序可在包括至少一个可编程处理器的可编程系统上执行和/或解释，该可编程处理器

可以是专用或者通用可编程处理器，可以从存储系统、至少一个输入装置、和至少一个输出装置接收数据和指令，并且将数据和指令传输至该存储系统、该至少一个输入装置、和该至少一个输出装置。

[0124] 用于实施本发明的方法的计算机程序可以采用一个或多个编程语言的任何组合来编写。这些计算机程序可以提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器，使得计算机程序当由处理器执行时使流程图和/或框图中所规定的功能/操作被实施。计算机程序可以完全在机器上执行、部分地在机器上执行，作为独立软件包部分地在机器上执行且部分地在远程机器上执行或完全在远程机器或服务器上执行。

[0125] 在本发明的上下文中，计算机可读存储介质可以是有形的介质，其可以包含或存储以供指令执行系统、装置或设备使用或与指令执行系统、装置或设备结合地使用的计算机程序。计算机可读存储介质可以包括但不限于电子的、磁性的、光学的、电磁的、红外的、或半导体系统、装置或设备，或者上述内容的任何合适组合。备选地，计算机可读存储介质可以是机器可读信号介质。机器可读存储介质的更具体示例会包括基于一个或多个线的电气连接、便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或快闪存储器)、光纤、便捷式紧凑盘只读存储器(CD-ROM)、光学储存设备、磁储存设备、或上述内容的任何合适组合。

[0126] 为了提供与用户的交互，可以在电子设备上实施此处描述的系统和技术，该电子设备具有：用于向用户显示信息的显示装置(例如，CRT(阴极射线管)或者LCD(液晶显示器)监视器)；以及键盘和指向装置(例如，鼠标或者轨迹球)，用户可以通过该键盘和该指向装置来将输入提供给电子设备。其它种类的装置还可以用于提供与用户的交互；例如，提供给用户的反馈可以是任何形式的传感反馈(例如，视觉反馈、听觉反馈、或者触觉反馈)；并且可以用任何形式(包括声输入、语音输入或者、触觉输入)来接收来自用户的输入。

[0127] 可以将此处描述的系统和技术实施在包括后台部件的计算系统(例如，作为数据服务器)、或者包括中间件部件的计算系统(例如，应用服务器)、或者包括前端部件的计算系统(例如，具有图形用户界面或者网络浏览器的用户计算机，用户可以通过该图形用户界面或者该网络浏览器来与此处描述的系统和技术的实施方式交互)、或者包括这种后台部件、中间件部件、或者前端部件的任何组合的计算系统中。可以通过任何形式或者介质的数字数据通信(例如，通信网络)来将系统的部件相互连接。通信网络的示例包括：局域网(LAN)、广域网(WAN)、区块链网络和互联网。

[0128] 计算系统可以包括客户端和服务器。客户端和服务器一般远离彼此并且通常通过通信网络进行交互。通过在相应的计算机上运行并且彼此具有客户端-服务器关系的计算机程序来产生客户端和服务器的关系。服务器可以是云服务器，又称为云计算服务器或云主机，是云计算服务体系中的一项主机产品，以解决了传统物理主机与VPS服务中，存在的管理难度大，业务扩展性弱的缺陷。

[0129] 应该理解，可以使用上面所示的各种形式的流程，重新排序、增加或删除步骤。例如，本发明中记载的各步骤可以并行地执行也可以顺序地执行也可以不同的次序执行，只要能够实现本发明的技术方案所期望的结果，本文在此不进行限制。

[0130] 上述具体实施方式，并不构成对本发明保护范围的限制。本领域技术人员应该明白的是，根据设计要求和其他因素，可以进行各种修改、组合、子组合和替代。任何在本发明

的精神和原则之内所作的修改、等同替换和改进等,均应包含在本发明保护范围之内。

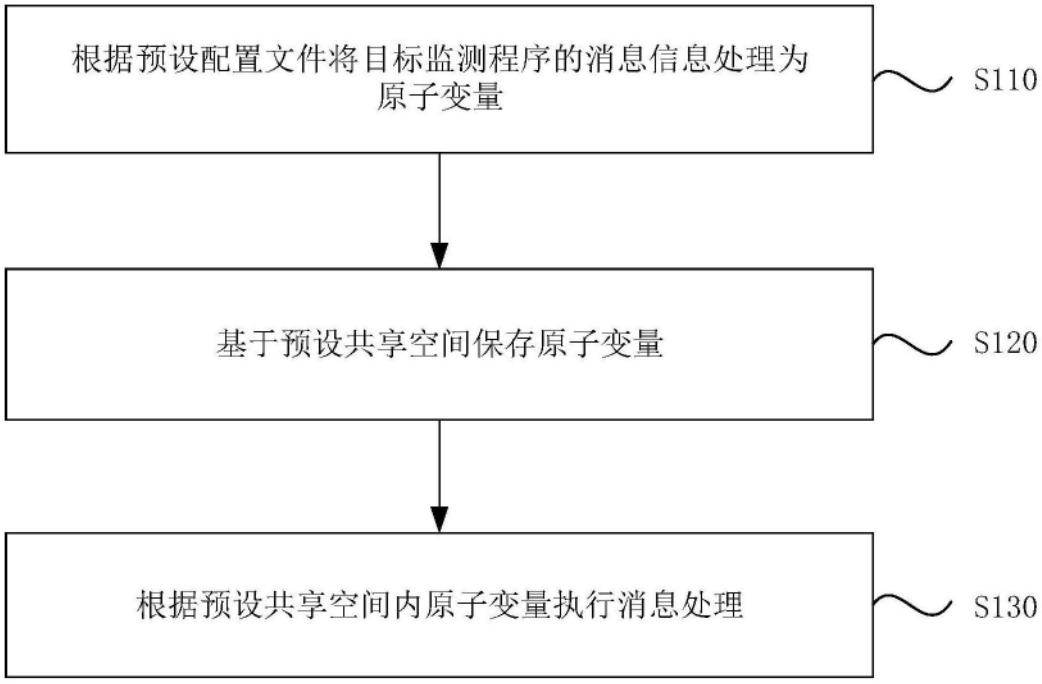

根据预设配置文件将目标监测程序的消息信息处理为原子变量
S110
基于预设共享空间保存原子变量
S120
根据预设共享空间内原子变量执行消息处理
S130

图1

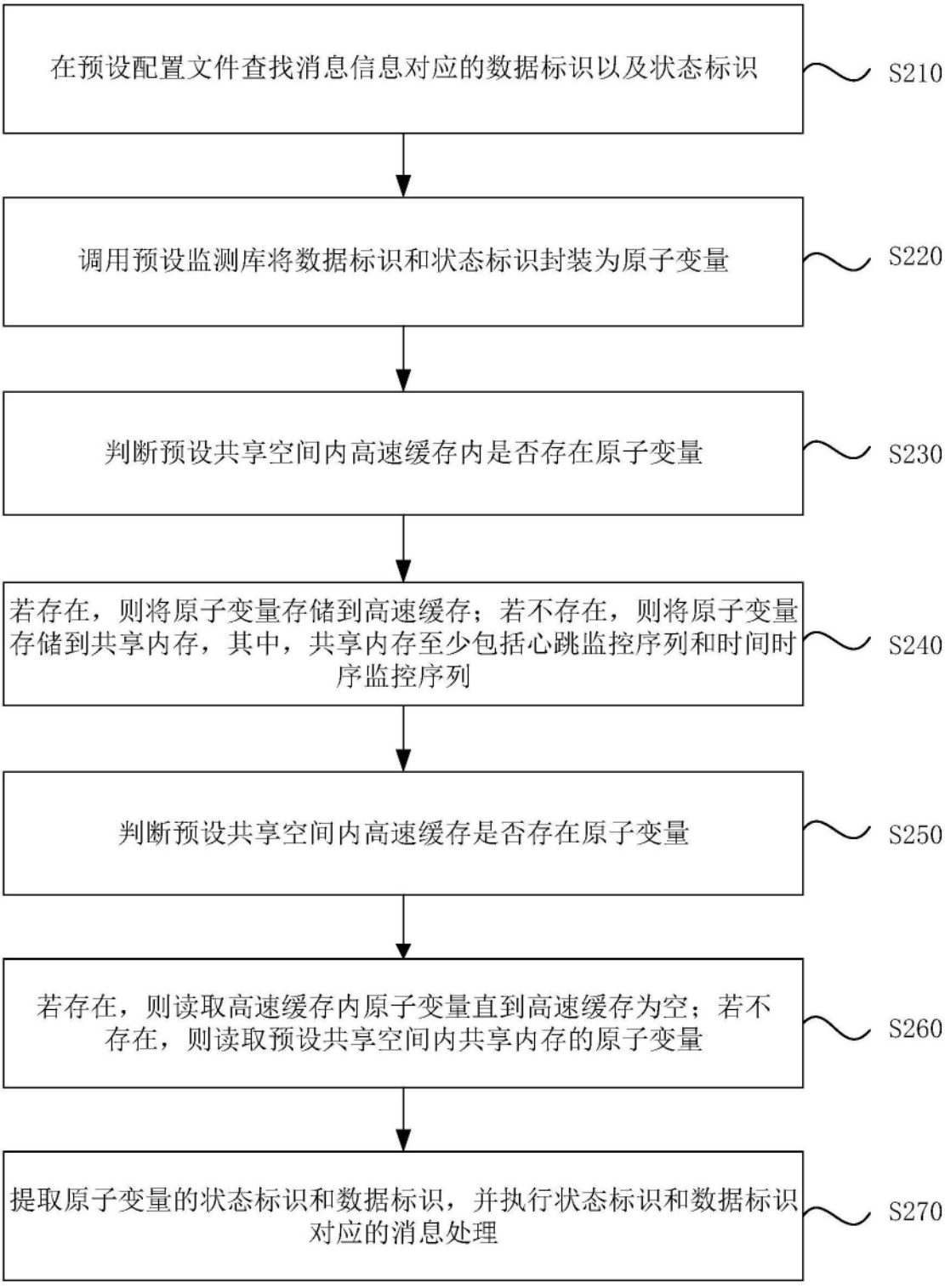

在预设配置文件查找消息信息对应的数据标识以及状态标识
S210
调用预设监测库将数据标识和状态标识封装为原子变量
S220
判断预设共享空间内高速缓存内是否存在原子变量
S230
若存在，则将原子变量存储到高速缓存；若不存在，则将原子变量存储到共享内存，其中，共享内存至少包括心跳监控序列和时间时序监控序列
S240
判断预设共享空间内高速缓存是否存在原子变量
S250
若存在，则读取高速缓存内原子变量直到高速缓存为空；若不存在，则读取预设共享空间内共享内存的原子变量
S260
提取原子变量的状态标识和数据标识，并执行状态标识和数据标识对应的消息处理
S270

图2

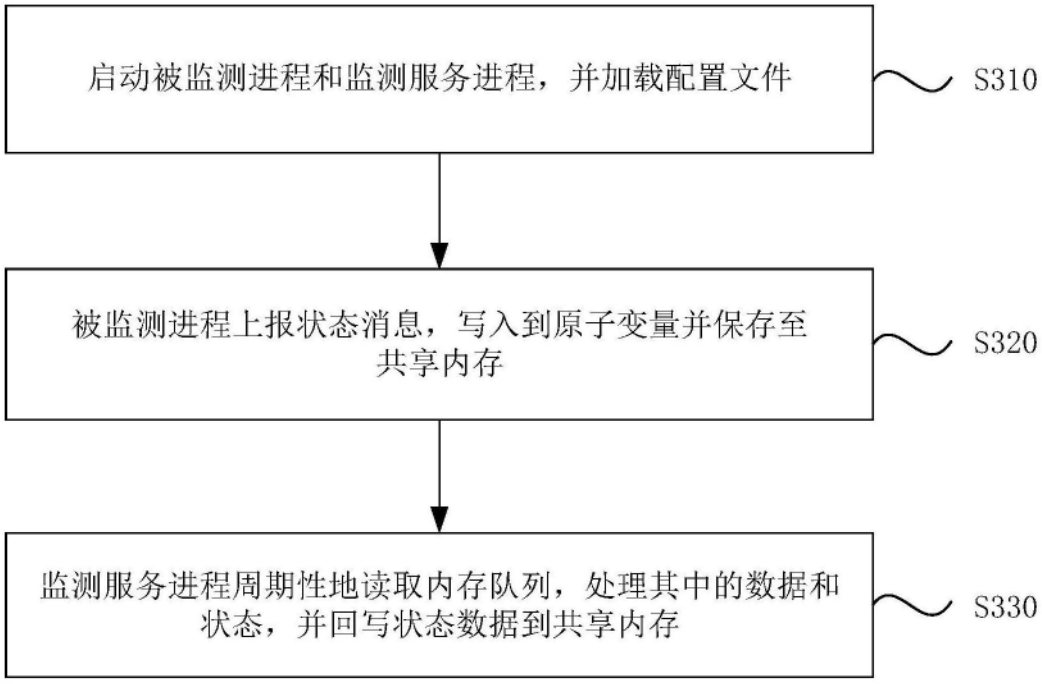

启动被监测进程和监测服务进程，并加载配置文件
S310
被监测进程上报状态消息，写入到原子变量并保存至共享内存
S320
监测服务进程周期性地读取内存队列，处理其中的数据和状态，并回写状态数据到共享内存
S330

图3

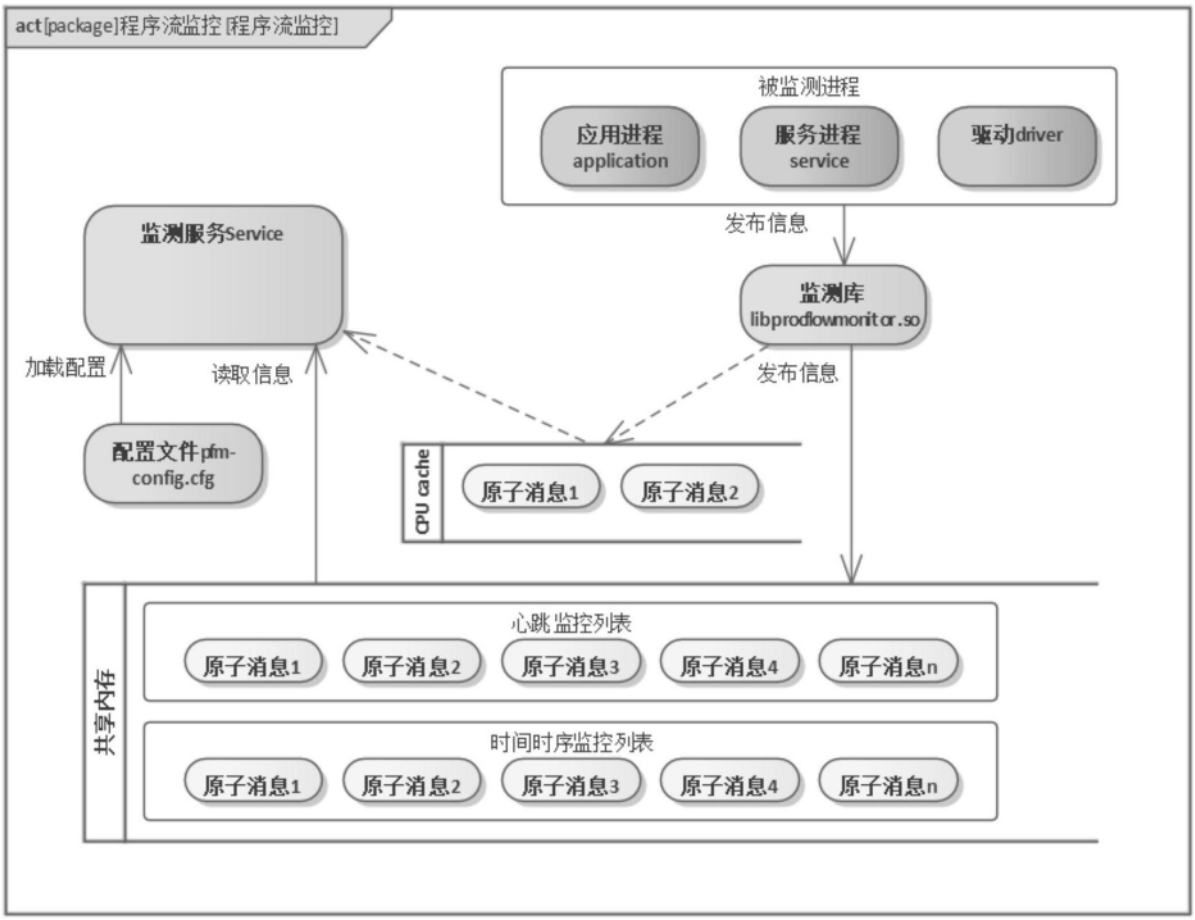

act[package]程序流监控[程序流监控]
被监测进程
应用进程
application
服务进程
service
驱动driver
发布信息
监测服务Service
监测库
libprodlowmonitor.so
加载配置
读取信息
发布信息
配置文件pfm-
config.cfg
CPU cache
原子消息1
原子消息2
共享内存
心跳监控列表
原子消息1
原子消息2
原子消息3
原子消息4
原子消息n
时间时序监控列表
原子消息1
原子消息2
原子消息3
原子消息4
原子消息n

图4

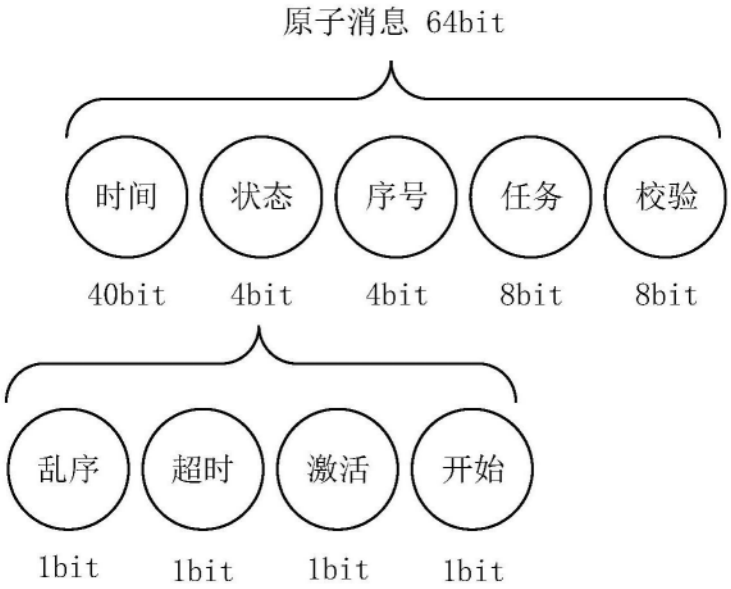

原子消息 64bit
时间
状态
序号
任务
校验
40bit
4bit
4bit
8bit
8bit
乱序
超时
激活
开始
1bit
1bit
1bit
1bit

图5

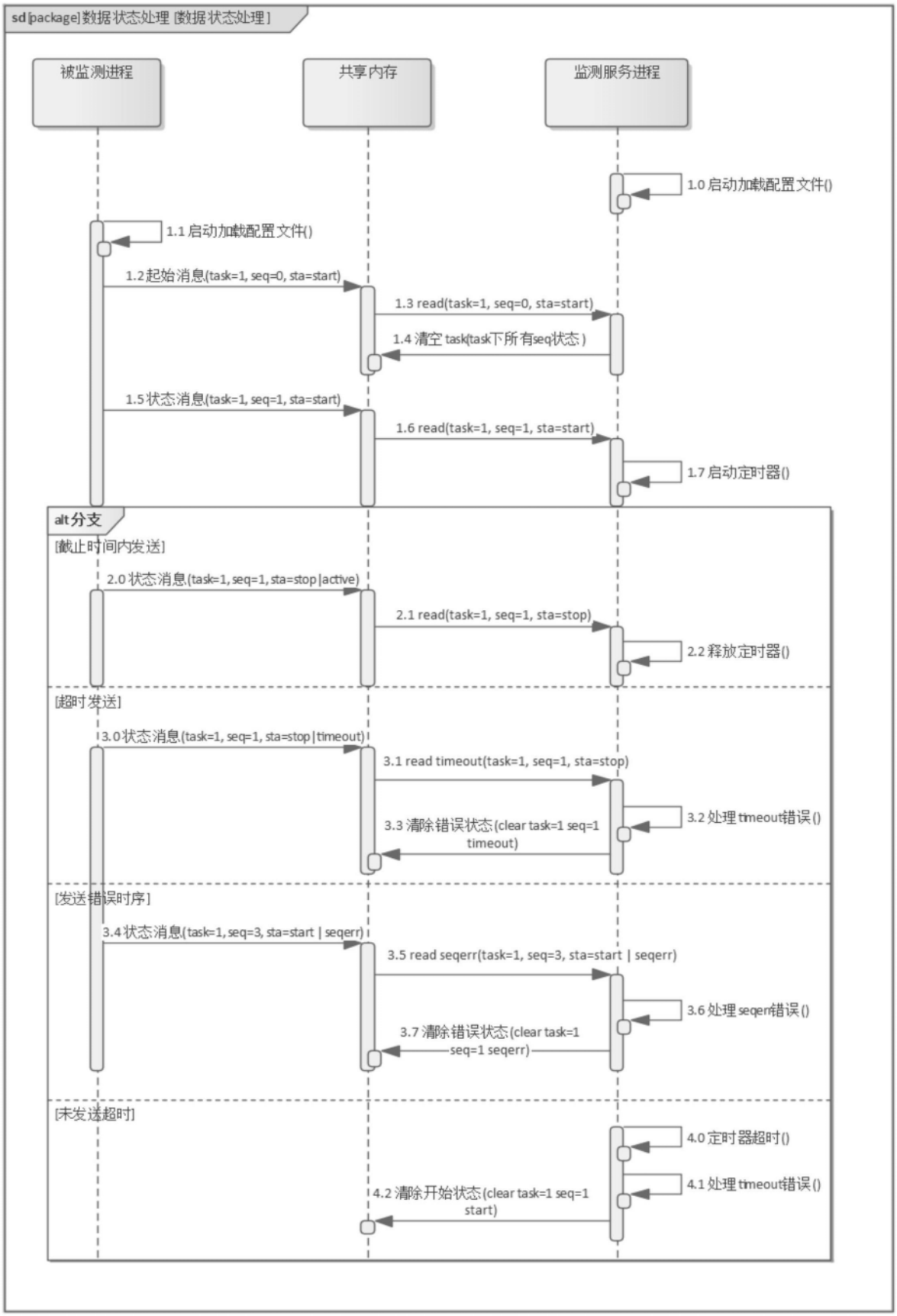

sd [package] 数据状态处理 [数据状态处理]
被监测进程
共享内存
监测服务进程
1.0 启动加载配置文件()
1.1 启动加载配置文件()
1.2 起始消息(task=1, seq=0, sta=start)
1.3 read(task=1, seq=0, sta=start)
1.4 清空 task(task下所有seq状态)
1.5 状态消息(task=1, seq=1, sta=start)
1.6 read(task=1, seq=1, sta=start)
1.7 启动定时器()
alt 分支
[截止时间内发送]
2.0 状态消息(task=1, seq=1, sta=stop|active)
2.1 read(task=1, seq=1, sta=stop)
2.2 释放定时器()
[超时发送]
3.0 状态消息(task=1, seq=1, sta=stop|timeout)
3.1 read timeout(task=1, seq=1, sta=stop)
3.2 处理 timeout错误()
3.3 清除错误状态(clear task=1 seq=1 timeout)
[发送错误时序]
3.4 状态消息(task=1, seq=3, sta=start | seqerr)
3.5 read seqerr(task=1, seq=3, sta=start | seqerr)
3.6 处理 seqerr错误()
3.7 清除错误状态(clear task=1 seq=1 seqerr)
[未发送超时]
4.0 定时器超时()
4.1 处理 timeout错误()
4.2 清除开始状态(clear task=1 seq=1 start)

图6

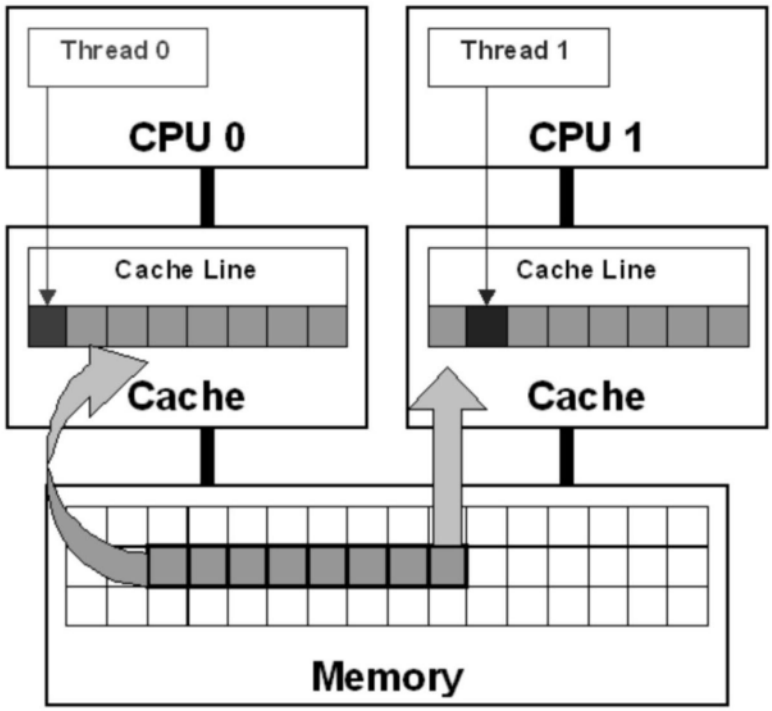

Thread 0
Thread 1
CPU 0
CPU 1
Cache Line
Cache Line
Cache
Cache
Memory

图7

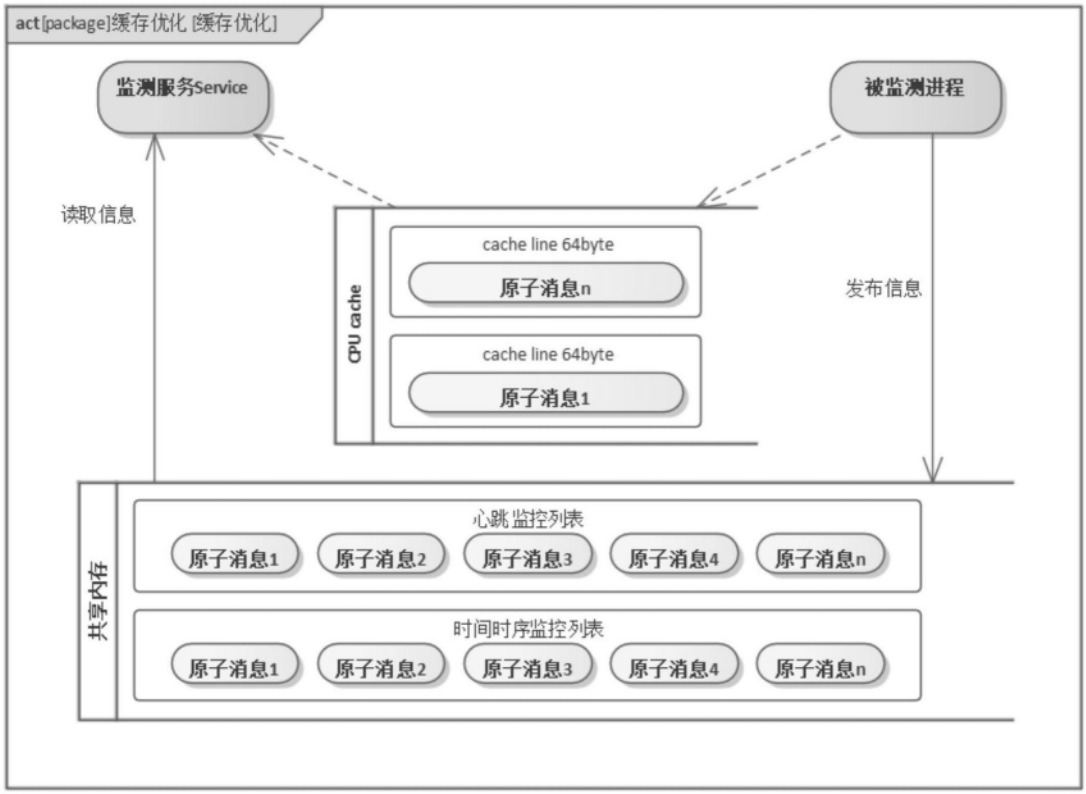

act[package]缓存优化 [缓存优化]
监测服务Service
被监测进程
读取信息
发布信息
CPU cache
cache line 64byte
原子消息n
cache line 64byte
原子消息1
共享内存
心跳监控列表
原子消息1
原子消息2
原子消息3
原子消息4
原子消息n
时间时序监控列表
原子消息1
原子消息2
原子消息3
原子消息4
原子消息n

图8

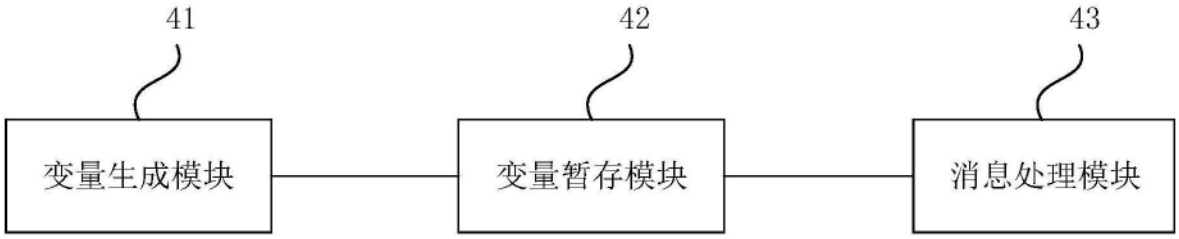

41
42
43
变量生成模块
变量暂存模块
消息处理模块

图9

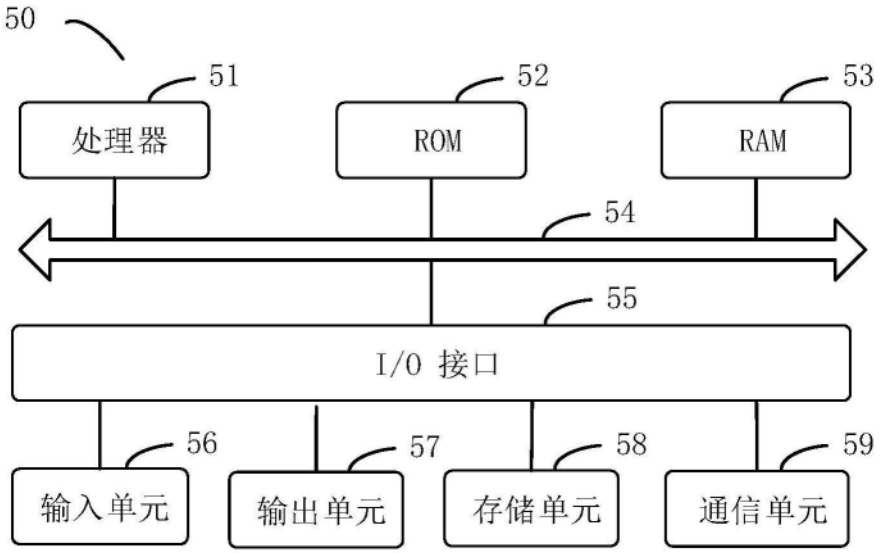

50
51
52
53
处理器
ROM
RAM
54
55
I/O 接口
56
57
58
59
输入单元
输出单元
存储单元
通信单元

图10