第31卷 第16期 计 算 机 工 程 2005年8月
Vol.31 № 16 Computer Engineering August 2005
·工程应用技术与实现· 文章编号：1000—3428(2005)16—0188—03 文献标识码：A 中图分类号：TP316

基于QNX的高可用性工具的研究与实现

常政威，傅 鹏，陈海春
(中国科学院等离子体物理研究所，合肥 230031)

摘 要：在实时操作系统QNX6.20下，设计了毫秒级、快速响应的高可用性工具（HAT）。它采用被动侦听和主动轮询两种模式，实时监控系统服务和用户进程，确保每个线程都能正确、可靠地工作。HAT应用于核聚变装置极向场电源控制系统中，取得了良好的效果。
关键词：极向场；实时控制；QNX；高可用性

Research and Implementation of High Availability Toolkit Based on QNX

CHANG Zhengwei, FU Peng, CHEN Haichun
(Institute of Plasma Physics, Chinese Academy of Sciences, Hefei 230031)

【Abstract】 This paper presents a high availability toolkit(HAT) which can respond quickly within some milliseconds based on QNX6.20 RTOS. Through passive listening and active polling, HAT monitors the system services and user processes to guarantee that any thread would run reliably and correctly. It is applied to a poloidal field power supply control system of nuclear fusion device, and achieves good effects.
【Key words】 Poloidal field; Real-time control; QNX; High availability

热核聚变实验装置“EAST(Experimental Advanced Superconducting Tokamak)”是国家“九五”重大科学工程，将是世界上第一个同时具有全超导磁体和灵活的冷却结构的托卡马克。EAST实验装置上均匀分布着14个极向场超导线圈，根据等离子体运行条件和极向场参数，可以将其等效为12对线圈。极向场电源控制系统为线圈提供约束等离子体所需要的电流，由12套软、硬件子系统组成，其控制周期为1ms。当某套电源发生内部故障、计算机硬件或软件出错时，若不及时进行处理，将会影响装置的安全和实验的成功。这对于计算机控制系统的实时性、可用性提出了很高的要求。

极向场电源控制系统基于QNX6.20(一个符合POSIX 1003标准、支持内核抢占、真正微内核、基于消息传递的实时操作系统)，在Intel的x86处理器和通用的工业器件上得到了良好的实时性能。本文介绍了在QNX平台下，如何实现该系统的高可用性（High Availability，HA）。

1 HA的设计思想

可用性是用来衡量系统可靠性的尺度，指系统在某一时刻t可正确运行的概率。在实际中，可用性表现为系统正确运行、能够提供服务的时间所占的百分比，即MTTF/(MTTF+MTTR)。其中，MTTF指平均无故障时间，MTTR指平均修复时间。由此可见，通过减少MTTR，可以提高系统的可用性，获得HA。因此，一个HA系统允许出现失效，只要失效和系统恢复的时间足够短。

研究表明，在二十世纪五六十年代，绝大部分的系统故障都来自于计算机硬件设备。而今天，相对于软件及操作错误，硬件故障只占了相当小的比例。因此，提高软件的可用性，减少其错误恢复的时间也变得越来越重要。

基于模块化的设计思想，我们设计了一套毫秒级、快速响应的高可用性工具(HA Toolkit，HAT)，确保系统中的每个工作线程都能正确、可靠地运行。HAT主要包括被动侦听和主动轮询两个模块，用于实时监控系统服务和用户进程。当被监控进程中的线程进入非正常状态、出现故障、中止服务时，可及时地调用故障处理程序，对进程和服务进行错误屏蔽、修复，恢复正常运行。

2 被动侦听模块

2.1 QNX进程间的消息传递

QNX是一个真正的微内核、模块化的操作系统，表现为一个软件总线的模式；除内核外的操作系统模块，如文件管理器，可以动态地从软件总线上加载或卸除。QNX也是第一个采用消息传递作为模块间通信基本模式的商业操作系统。虽然QNX提供了如下几种IPC: Message-passing(消息传递)、Signal(信号)、POSIX message queues(消息队列)、Shared memory(共享内存)、Pipe(管道)、FIFO，但所有的IPC都是建立在消息传递的基础上。

消息传递是这样的一个同步Send/Receive/Reply的过程：

(1)接收方进程A创建一个通道，调用MsgReceive()等待接收消息，并进入Receive blocked状态；

(2)发送方进程B调用ConnectAttach()与进程A建立一个连接，获得一个连接ID，然后调用MsgSend()发送消息，进入Send blocked状态；

(3)进程A收到消息(此时进程B进入Reply blocked状态)，进行处理后，调用MsgReply()回复，使进程B变为Ready。

QNX同时提供了一种定长、非阻塞的消息，称为Pulse，由5个字节组成，包括1个字节的编码和4个字节的数值。Pulse的调用方式同普通的消息传递，所不同的是

基金项目：国家发展计划委员会“投资(1998)1303号项目”(子项目)
作者简介：常政威(1981—)，男，硕士生，研究方向为实时计算机控制、数据库应用；傅 鹏，研究员、博导；陈海春，硕士生
定稿日期：2004-06-30 **E-mail：**changzw@ipp.ac.cn

MsgReceive()的返回值若为 0，则表明收到的是一个 Pulse，否则是普通的消息。

2.2 被动侦听模块的 API 说明与定义

本模块采用 Client/Server 结构，监控进程(即被动侦听模块)作为服务器，被监控进程作为客户端。服务器随着系统启动作为守护进程，在后台开始运行，等待客户端连接。客户端可通过下面 5 个接口函数，与服务器进行交互：

(1)int ha_connect(int period, char *err_proc_file)

与服务器建立连接。period 为心跳周期，以毫秒为单位；err_proc_file 为客户进程故障处理程序的路径和文件名；返回一个连接 ID coid，作为其他调用的参数。

(2)ha_start_heartbeat(int coid)

发送“开始心跳”消息，在第一次向服务器发送“心跳”消息之前调用。

(3)ha_heartbeat(int coid)

向服务器发送“心跳”消息，此时客户端应该处于正常的轮转运行状态。

(4)ha_stop_heaertbeat(int coid)

向服务器发送“停止心跳”消息，客户端主动停止监控，不再发送“心跳”消息。

(5)ha_disconnect(int coid)

与服务器断开连接。

2.3 被动侦听模块的实现

服务器进程的优先级设为系统的最高级，它拥有一个进程队列，包含所有被监控进程的主要信息，有进程 PID (Process ID)、故障处理程序文件名和丢失的心跳数等。客户端在生存期按照设定“心跳周期”，定时向服务器发送“心跳”消息。服务器则定时对队列中的每个进程进行检查，若连续 N 次收到某进程的心跳消息，则认为该进程出现故障，调用其故障处理程序进行恢复。

图 1 为被动侦听模块服务器的程序流程。

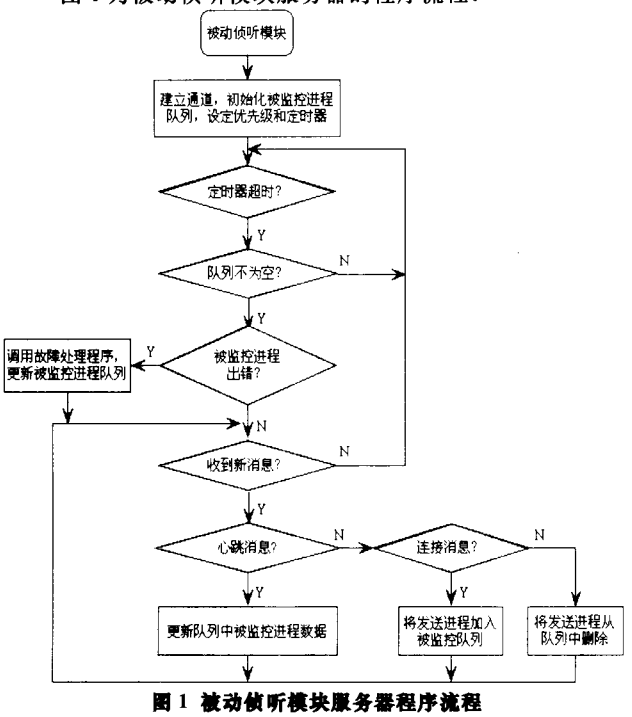

图 1 被动侦听模块服务器程序流程

客户端与服务器之间通过消息传递和 Pulse 通信。ha_connect()由消息传递来实现：服务器首先建立一个通道，等待客户端连接。客户端通过 MsgSend()在服务器登记，服务器则通过 MsgReply()返回 coid。客户端的心跳消息由于没有要传递的额外数据，采用异步的 Pulse，可以减少被监控进程的负担。其中，正常心跳、开始心跳、终止心跳 3 种心跳消息用 Pulse 数据结构中的不同编码值来表示。

3 主动轮询模块

3.1 进程文件系统

QNX6.20 支持很多的文件系统类型，进程文件系统是其中的一种。proc 是一种虚拟的伪文件系统，安装在根目录下的 proc 目录，并不占用实际空间。proc 目录中包含许多以进程 PID 命名的子目录，这些子目录都只有一个 as(address space，进程地址空间)文件，对应系统中的每一个进程。系统创建一个进程时，同时在 proc 中也创建一个相应的目录和文件。对 proc 中文件的操作与普通文件一样，使用通常的 open、read/write、lseek、ioctl、devctl、close 等操作。这样，就可以使用标准的文件系统接口和系统调用，来读取和修改系统中进程的地址空间，并对其进行若干控制操作。

devctl()函数用来对文件和其他设备进行操作，取得需要的信息，它的调用形式是：

```
int devctl( int      filedes,
            int      dcmd,
            void     *dev_data_ptr,
            size_t   n_bytes,
            int      *dev_info_ptr   );
```

filedes 为打开设备时返回的文件描述符，dcmd 表示操作命令，dev_data_ptr 指向与设备通信的数据缓冲区，n_bytes 是它的长度，dev_info_ptr 指向设备返回的信息。这里，主要用到的 dcmd 是：

(1)DCMD_PROC_MAPDEBUG_BASE

发送一个信号给被调试进程，可以获得该进程的有关信息，如它的可执行程序文件名。

(2)DCMD_PROC_INFO

得到由 filedes 指定进程的映射内存空间的相关信息，如进程 PID。

(3)DCMD_PROC_TIDSTATUS

指定进程的当前线程，可以得到当前线程的状态等。

3.2 主动轮询模块的调用方式

在被动侦听模块中，主要关心客户进程是否按时发送了心跳消息，取决于与服务器通信的主线程有没有运行，即处于 Running 状态。实际上，一个进程中的各个线程的状态可能是图 2 中的任何一种。被动侦听模块只能被用户自己编写的程序调用，无法用于一些源代码未知的系统服务和进程。为此，引入了主动轮询模块。

主动轮询模块根据设定周期，主动查询所有被监控进程的各个线程的状态，当某一线程的状态变为指定值时，采取预先设定的动作。其调用格式为：

```
monitor <-p(pid, ...)|-f(file name, ...)> [-t interval ]
[-e execute file] [-g]
```

这样，通过调用 monitor 命令，可对指定 PID 或文件名的系统和用户进程按照设定的时间周期进行监控。

需要进行说明的是其中的-g 选项。在一个 HA 系统里，必须尽量避免出现单点故障(Single Point Of Failure ，SPOF)。monitor 进程拥有所有被监护进程的状态信息和错误恢复信息，尤其不能成为一个 SPOF。解决办法是，创建一个称为 guardian 的镜像进程作为冗余，共享所有监控信息，暂时处于阻塞状态。当 monitor 进程非正常终止时，guardian 进程会马上代替 monitor 进程工作，并创建一个新的 guardian 进程。这样，monitor/guardian 进程对共同存在，任何一个出现故障，另一个都可以马上使之恢复，较好地解决了这个问题。

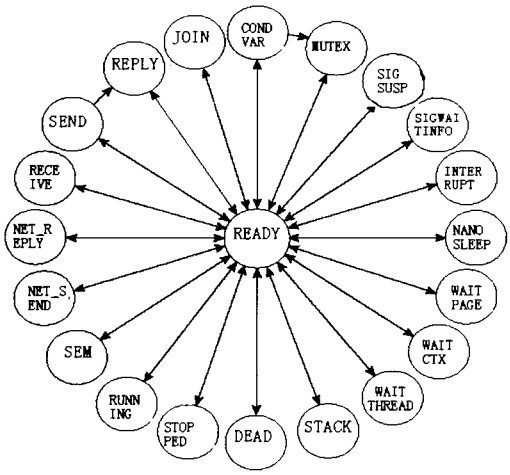

图 2　QNX 线程的状态变化图

4 主动轮询模块的实现

利用进程文件系统的接口，可以随时得到被监控进程中的每个线程的详细信息，从而实现实时监控的目的。

根据实际的调用格式，分别进行处理：

(1)monitor –f filenames

依次访问/proc 的各个目录项，调用 devctl(fd, DCMD_PROC_MAPDEBUG_BASE, &name, sizeof(name), NULL)得到系统各个进程的可执行程序文件名，找到与“filenames”相匹配的，确定用户想要被监控的进程队列。然后，调用 devctl(fd, DCMD_PROC_INFO, &pinfo, sizeof(pinfo))得到对应的 PID。转入步骤(2)。name，pinfo 的定义：

```
struct{   procfs_debuginfo    info;
char      buf[MAX_PATH_LEN];
}name;        //存有进程对应的文件名
procfs_info        pinfo; //pinfo.pid 为进程 PID
```

(2)monitor –p pids

根据给定的进程 PID，构建被监控进程队列。并按照设定的监控周期，依次调用 devctl(fd, DCMD_PROC_TIDSTATUS, &status, sizeof(status), 0)，轮询各进程所有线程的状态。若某个线程进入非正常状态(如 DEAD，STOPPED，INTR)，则记入日志，调用出错处理程序。status 定义为：

```
procfs_status    status; //status.state 保存当前线程的状态
```

5 HAT 的应用及其时间性能

极向场电源控制系统的核心部分包括反馈控制节点，数据库服务器节点和 12 个电源控制节点。整个控制系统采用了 QNX 软件总线和模块化的设计思想，每个节点对应一个功能模块，由一个进程或进程组来实现。HAT 模块可以动态地加载到任何一个节点，实现各个功能模块的 HA，从而提高整个实时控制系统的可用性。其中，被动侦听模块用于单线程轮转的用户进程，基本上不会加重系统负担；而主动轮询模块可实现对系统中任何进程的每一个线程的监控，但对系统的性能有所影响。多数情况，二者同时使用，更为可靠。

在对 HAT 的时间性能进行分析之前，先作如下定义：

(1)故障检测延迟 (Tc)：指系统内某进程发生异常到这个异常被 HAT 捕获到的时间。

(2)进程调度延迟 (Td)：指进程之间进行切换所花费的时间，QNX 下约为 10μs。

(3)故障处理延迟 (Te)：指某进程发生异常到其故障处理程序执行时刻的时间。

(4)进程加载时间 (Tm)：指装载故障处理程序时间，即操作系统从外存（硬盘）中将可执行文件调入内存中，并使之处于 Running 状态。

被动心跳模块 Tc 的大小与心跳周期 period 和丢失心跳数上限 N 的大小有关。由于进程间的消息传递的时间是微秒级的，可以忽略不计，因此 Tc=period×N。Period 取控制周期（1ms）的倍数，N 则一般≥3，故 Tc 最小为 3ms。

对于主动论询模块，Tc 主要取决于监控周期 interval 和被监控的线程个数。在电源控制节点，interval 取 2ms，被监控线程数为 20 个左右时，经过实际测量，在一个控制周期（1ms）内可以获得所有线程的状态，即 Tc 可达到 3ms。

如图 3（由示波器测得），HAT 的故障处理延迟 Te 具体表现为

$$Te = Tc + Td + Tm$$

$$Te \approx 3ms + 0.01 + 42\,ms \approx 45\,ms$$

可见，故障处理延迟 Te 大部分都花在装载故障处理程序的时间（约 40ms）上，若需进一步降低故障处理延迟 Te，可使用以下两种方法：

(1)采用 RAM disk，划分出一定大小的 RAM 空间预先（或动态）装入指定外存文件，减少读写外存的时间。

(2)将故障处理程序预先装入内存中，使之处于 Receive blocked 状态，当进行故障处理时由 HAT 发送一个解阻塞的 Pulse 即可。Tm 即为消息传递的时间(微秒级)，Te 基本上就是 Tc。

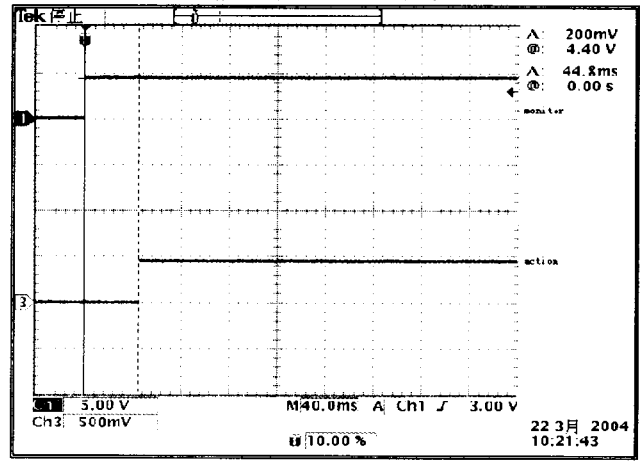

图 3　故障处理延迟 Tc

6 结束语

在现有 4 套电源上的试验结果表明，当某一节点出错时，HAT 可以使之在几十毫秒乃至数毫秒的时间内得到处理，系统的可用性得到了保障。随着 EAST 极向场电源控制系统的实现和完善，我们将研究如何从整个系统的角度、跨网络实现全局的 HA。

参考文献

1 Gray J, Siewiorek D P. High- availability Computer Systems. Proceedings of the 23rd International Conference on Fault Tolerance Computing, 1991:39-47

2 刘　建，沈美明. Unix 进程文件系统及其在调试器设计中的应用. 计算机工程，2004，30(4)：176-178

3 李　艺，李新明，刘绍南等. 高可用性系统软件 HHA 的模块化设计与实现. 计算机工程，2001，27(9)：56-58