维普资讯 http://www.cqvip.com

计算机科学1997 Vol. 24 №. 3

RISC处理机, Transputer, 容错系统 TP302.8

一个基于 Transputer 的容错系统

苏 森 唐雪飞 刘锦德
(电子科技大学微机所 成都610054)

摘 要 Fault tolerant is a major way to improve computer's reliability. This paper represents a fault tolerant system based on transputers. The major feature of this system is that the task on the failed computer run only from the point where the node failed.

关键词 Fault tolerant, Topology, Dynamic task migration, Process, Workspace

Transputer 是英国 INMOS 公司推出的 RISC 处理机。由于其优良性能以及在容错和并行处理方面得天独厚的优势，国内外专家已对其进行了广范深入的研究。Standven 在 Transputer 上实现了一个基于表决器思想的容错系统，每个结点由三个重复的处理器组成，每一对结点之间有三对表决器，按照多数表决来获得容错性能[3]；Castro 和 Gough 设计了一个用于航天的4-Transputer 容错系统，利用 Transputer 的四条链路来动态改变 Transputer 阵列的拓扑结构以获得容错性能[4]；文[6][7]中也概述了作者设计和实现的各种容错系统。以上提到的系统各有优点，但同时又有一个共同的致命弱点：当正在运行任务的处理机出现故障后，故障结点上任务必须在备份机上从起始位置重新执行。如果故障结点上原来运行的是大型任务，这将浪费大量的计算资源。更为严重的是，在这种系统中存在由于并行进程的推进顺序不当而造成死锁的可能性。R.Krishna 宣布它的基于 Transputer 的容错系统克服了此缺点[5]。但经过仔细分析，我们发现，该系统仅仅是保存了任务中的一些重要变量的中间结果，并在其他结点上备份。当有结点出现故障时，该结点上的任务在备份结点上以保存的中间值作为初值从任务的起始处重新执行。可见这种系统只适用于有关循环迭代之类的任务，而不是真正的从任务代码断点处执行。为了克服现存容错系统的缺点，实现"故障结点上的任务在备份结点上从断点处执行"，下面将详述在系统设计过程中涉及到的有关技术。

1 系统拓扑结构

在容错系统中，当检测到硬件故障时，需要结点重组，这必然涉及拓扑重构。传统的拓扑重构途径是，首先选择一个静态拓扑为基础，即网络中的结点固定连成一个拓扑结构，但可以重构。在使用过程中，应用程序将问题映射到其中的一部分拓扑上。例如，取超立方体为基本拓扑结构，其中可嵌入立方体、环、树和链等结构。这种方法的优点是实现简单，系统开销低，缺点是受基本拓扑结构的限制，重构灵活性较差。

为克服传统途径的缺点，并针对 Transputer 的特点，我们设计了以下方案，如图1。

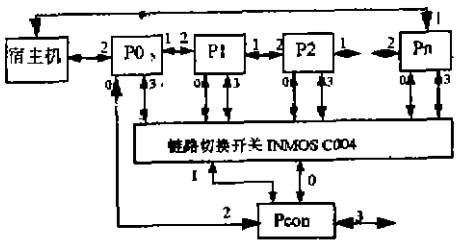

图1 系统拓扑结构

P_0～P_n 是 n+1个 Transputer，它们是整个系统的处理结点。考虑到链表结构在 Transputer 网络中的广泛应用和硬件(INMOS B008)强有力的支持，本方案通过 P_0的链路1、P_n 的链路2和其它 Transputer 的链路1与2连成一个链表结构；将 P_0的链路0和其它 Transputer 的链路0与3与链路切换开关 INMOS C004[1]的相应链路相连。INMOS C004的配置端和一个专门用于控制的 Transputer(P_{con})相连，运行在 P_{con}上的重构软件通过向 C004发送各种互连配置命令来控制其链路的连通和断开状态，以完成整个系统拓扑重构。宿主机是一台 AST486，和它直接相连

维普资讯 http://www.cqvip.com

的结点叫根结点。当进行系统开发时，它和根 Transputer 一起组成 TDS(Transputer Development System)的硬件平台；当系统正常运行时，它作为整个系统的 I/O 设备。当系统中某结点 $P_i(0<i<n)$出现故障时，运行在 P_{con}上的软件把 P_{i-1}的链路3和 P_{i+1}的链路0连在一起，并在系统中作记录，则故障结点 P_i 被排除在系统之外。当结点 P_i 被修复后，用相似的方法将它重新加入系统。当 P_0和 P_n 中的一个出现故障时，由另一个完成它的全部工作。

由于引入了基本拓扑结构，又存在可以切换的链路，所以在适度的软件开销前提下，大大提高了重构的灵活性。

2 动态任务迁移

动态任务迁移就是在任务执行过程中，将其迁移到备份结点上。当运行任务的结点出现故障时，该任务在备份结点上继续执行。

此外需要说明的是，在有关 Transputer 的经典文献中，进程指的是每条可执行的 OCCAM 语句[1,2]，进程的唯一标志是它的工作空间。工作空间中存放的是进程执行过程中所涉及到的寄存器的值。

本文中提到的任务是指程序在其数据结构上的一次执行，它包括三部分：任务控制块(TCB)、任务代码和任务工作空间。TCB 用于标识任务、描述任务的性质和控制任务的执行；任务工作空间就是该任务中的所有进程的工作空间总和。

2.1 任务控制块

本系统为每个任务配置一个任务控制块(TCB)，其内容如下：

```
VAL  TaskId              IS  0:
--任务标识符
VAL  TaskStatus          IS  1:
--任务状态
VAL  TaskPriority        IS  2:
--任务优先级
VAL  CodeAreaptr         IS  3:
--任务代码区指针
VAL  CodeAreaSize        IS  4:
--任务代码区大小
VAL  WorkSpacePtr        IS  5:
--任务工作空间指针
VAL  WorkSpaceSize       IS  6:
--任务工作空间大小
VAL  EntryOffset         IS  7:
--任务入口偏移地址
VAL  CheckPointWs        IS  8:
--检测点处进程工作空间
VAL  CheckPointIp        IS  9:
--检测点处指令指针
VAL  NextTcbPtr          IS 10:
--链表指针
```

TCB 各域的作用为：①任务标识符：唯一地标识该任务。②任务状态：记录该任务当前所处的状态(运行、就绪或阻塞)。③任务优先级：任务调度时的依据。④任务代码区指针：指向该任务代码区的首地址。⑤任务代码区大小：系统为任务代码分配存储资源的依据。⑥任务工作空间指针：指向该任务工作空间的首地址。⑦任务工作空间大小：系统为任务工作空间分配存储资源的依据。⑧任务入口偏移地址：指向任务的第一条可执行语句。⑨检测点处进程工作空间：保存任务运行到错误检测点时正在执行的进程的工作空间指针。⑩检测点处指令指针：下一条可执行的指令地址。⑪链表指针：用于把该结点上的所有 TCB 组成一个链表。

2.2 任务迁移

在整个系统中，每两个处理机构成一组。在任务开始运行时，同一组中的每个处理机在运行本机上的任务的同时互相备份对方上面的任务，包括任务控制块、任务代码和任务工作空间。此后，每到一个错误检测点，互相备份对方处理机上的任务工作空间。

备份任务和原任务的主要区别是它们的 TCB 不同，所以创建备份任务的主要工作就是根据原任务的 TCB (OldTCB) 创建备份任务的 TCB (NewTCB)。其具体过程如图2。

```
Creat(NewTCB,OldTCB)
SEQ
NewTCB.TaskId:=MaxId+1
--新任务的标识符等于该结点上最大的任务标识符加1
NewTCB.TaskStatus:=Block.t
--原任务正常运行时，备份任务处于阻塞状态
NewTCB.TaskPriority:=take(NewTCB.TaskId)
--给出备份任务的优先级
NewTCB.CodeAreaPtr:=MemAlloc(OldTCB.CodeAreaSize)
--根据代码区的大小为它分配内存
NewTCB.CodeAreaSize:=OldTCB.CodeAreaSize
NewTCB.WorkSpacePtr:=MemAlloc(OldTCB.WorkSpaceSize)
--根据工作空间的大小为它分配内存
NewTCB.WorkSpaceSize:=OldTCB.WorkSpaceSize
CodeAreaOffset:=NewTCB.CodeAreaPtr-OldTCB.CodeAreaPtr
--此任务的代码区在原结点和备份结点上的地址差异
WorkSpaceOffset:=NewTCB.WorkSpacePtr-OldTCB.WorkSpacePtr
--此任务的工作空间在原结点和备份结点上的地址差异
NewTCB.EntryOffset:=OldTCB.EntryOffset+CodeAreaOffset
NewTCB.CheckPointWs:=OldTCB.CheckPointWs+WorkSpaceOffset
NewTCB.CheckPointIp:=OldTCB.CheckPointIp+CodeAreaOffset
NewTCB.NexTcbPtr:=NULL
```

图2 创建备份任务的 TCB

为更清楚地表达此系统的设计思想，下面简单介绍一下 Transputer 的通信机制。

维普资讯 http://www.cqvip.com

3 Transputer 的通信机制

Transputer 的进程间通信是由通道实现的。在多 Transputer 组成的网络中，在不同 Transputer 上，进程之间的通道跨过连接两个 Transputer 的链；同一个 Transputer 上的进程之间的通道是内存中的特定存储区。Transputer 中的通信是同步和无缓冲的，只有同一通道上的输入进程和输出进程都准备好时，通信才发生，通道两端的任务继续向下执行。

当一个进程 P(其工作空间为 Pws)执行输入语句 channel?x(从通道 channel 中取值并赋予 x)，而相应的发送任务还没有执行到输出语句 channel!y(将 y 的值放入通道 channel 中)时，进程 P 只能等待。此时，Transputer 的内部状态如图3。

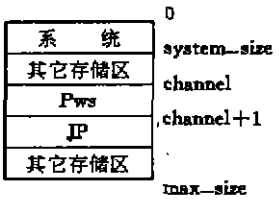

图3　和通信机制有关的内存内容

在 channel 指向的内存单元中，存放的是进程 P 的工作空间指针 Pws，和此单元相邻的下一个单元存放的是指令指针 IP，它指向等待进程的下一条可执行语句。

当相应的发送进程 channel!y 执行结束后，y 的值赋予 x，同时激活 Pws 对应的进程 P。

4 故障恢复

当备份机收到同组处理机出现故障的消息后，备份机上的系统将采取措施使备份任务在断点处重新执行。其具体思想如下：

首先将备份任务在检测点处的进程工作空间指针放入一个特定通道，在和其相邻的下一个单元中放入检测点处的指令指针：

[channel]=NewTCB. CheckPointWs；

[channel+1]=NewTCB. CheckPointIp。

当需要备份任务继续执行时，系统便执行进程 channel!y，即可唤醒被阻塞的备份任务。迁移到备份结点上的备份任务便在原任务的断点处重新执行，并且其执行环境和故障结点上原来的执行环境完全相同。

结束语　本系统已在九个 Transputer 组成的网络上正常运行。由于本系统的软件较大，除本文所述的以外，还包括内存管理、通信管理、任务调度和错误检测等，在此不可能一一详述，本文描述的仅是实现高级容错性能的核心思想。在设计此系统的过程中，虽然利用了 Transputer 的不少特有性能，但其基本思想对于其它处理机上实现容错性能也有借鉴之处。

参考文献

[1]The transputer databook, U K; INMOS Group of Companies, 1989

[2]Transputer Development System, UK; INMOS Group of Companies, 1988

[3]J. Standvan et al., Hardware voter for fault tolerant systems, Microprocessors and Microsystems, 13(9)1989

[4]H. D. S. Castro and M. P. Gough, A fault tolerant multitransputer system for space application, Microprocessors and Microsystems, 15(7)1991

[5]R. Krishna Kumar et al., A fault-tolerant multitransputer architecture, Microprocessors and Microsystems, 17(2)1993

[6]S. Omri, Fault tolerant systems in commerical application, IEEE Comput, 17, 1984

[7]D. A. Rennels, On implementing fault tolerance in binary hypercube, 16th Intl. Syposium on Fault-tolerant Computing, IEEE Computer Society Press, Washington, 1986