多线程技术在本地网管系统中的实现与应用

中央财经大学　孙宝文
北京邮电大学　范春晓

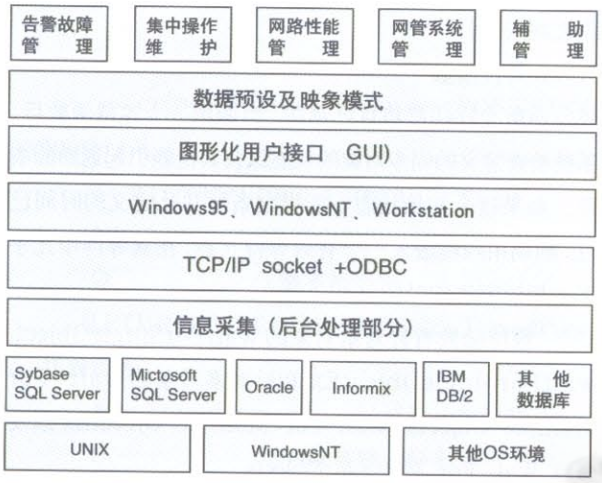
本文介绍了本地网管系统中（TIMS）多线程的应用和作用，用信号量方法实现线程间的同步，以正确访问数据临界区。节省系统时间开销，提高系统效率。

TIMS 系统

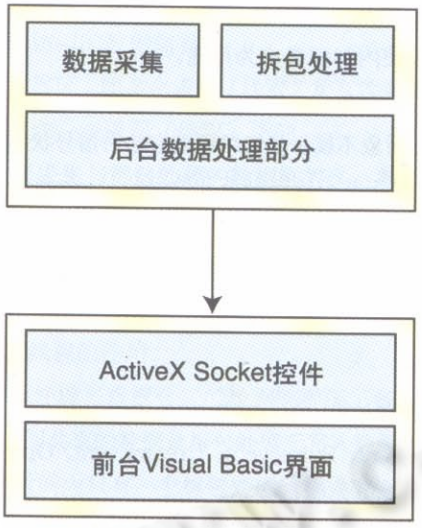
TIMS系统是根据邮电部电信总局《本地网网管和监控系统总体技术要求》，并参考国内外有关网管和集中监控系统，结合中国电信体制和电信改革方向研制，开发的一个通用的工具化的本地网网管和集中监控系统。

TIMS系统对电信网中网元实行集中监控，对有关告警、故障、话务等信息进行分析处理，使机房无人值守，信息资源共享。系统是一个通用的与环境无关的网络管理

系统，系统结构如下图：



系统采用Client_Server模式，有一台专用的通信服务器负责信息采集，并为前台表达部分提供数据。如下图示意：



其中信息采集部分是后台运行部分，要完成：

1. 数据采集工作，从交换机上采集原始信息。

2. 拆包处理，将原始数据拆包入库并向前台发通告。

3. 数据处理，对数据分析及处理。

这三部分工作是TIMS系统的基础服务工作，由于对数据的采集是24小时实时的，前台监视也是24小时实时，为了保证从网元发出的9600 bit/s 的数据完整无失，所以各个部分工作都要并行进行，如何处理好这三部分工作，使整个系统数据有效、正确处理，流动，是TIMS系统运行效率的关键。

后台信息采集的并发与互斥

1. 从事件并发整个控制流程上讲，数据采集、拆包处理和数据分析处理是顺序进行的，但是由于数据采集是24小时实时的，不间断的，并且所采集的不是一个交换机而是多台交换机的数据，所以不可能找到数据采集的间断点，来顺序调用拆包处理和数据分析，因此数据采集，拆包处理及数据分析要按三个并发事件考虑及处理。

TIMS系统后台程序是在Windows NT环境下运行，利用Win32及Visual C开发的。Windows NT支持多进程及多线程。在设计时考虑整个后台信息采集作为一个进程，而数据采集、拆包处理及数据分析三部分作为该进程的三个线程，这样既节省系统对资源的开销，也便于各部分工作间联系，因为线程共用进程申请的资源。

这三个线程可以有两种方式运行，一是从数据控制流程的角度出发，当一个工作完成能进行另一个工作时，建立线程。例如，当数据采集线程采集到数据，则建立拆包处理线程，某一拆包处理线程工作完毕即等待下次建立，这种方式能够保证后台信息采集进程的各线程的工作并发，但是根据Windows NT的原理，每建立一个线程是要有时间开销的，这样后台进程24小时工作，频繁建立线程，大大增加系统时间开销，会影响到TIMS系统的运行效率。

另一种方法就是系统启动时后台进程创建n个线程运行，各线程间均有不同的共享数据区，这个数据是访问临界区，即n个进程间的关系是典型的生产者及消费者问题。

2. 生产及消费者问题。所谓生产者问题是在实现进程（线程）间同步和互斥时描述进程（线程）的关系的。即一个进程（线程）如果它使用资源（尤其是软资源——缓冲区中的数据），可称其为消费者，而另一个进程（线程）如果是制造并释放上述资源，则可把它称为生产者。它们二者之间的同步关系问题即称为生产者和消费者关系问题。

对多个生产者和多个消费者同时共用某一仓库存储资源时，它们之间既需同步，也需要互斥。

(1)同步：当仓库已满，生产者要等待消费者取出产品方能再生产；当仓库为空时，消费者必须等待生产者生产产品。

(2)互斥：同一时刻，只能有一个生产者或一个消费者对仓库进行访问。可以利用P、V操作描述生产者与消费者的同步及互斥关系，如下：

```
Var empty, full, mutex; shared semaphore;
Begin
```

```
    empty=n; //表达仓库空间个数，初值为n，即无产品
    full=Null; //表示仓库装产品个数，初值为空
    mutex=1; //互斥信号量
cobegin
producter i (生产者进程):
    begin
      repeat
        P(empty);
        P(mutex);
        生产产品把产品送入仓库;
        V(mutex);
        V(full);
      until false
    end
consumer i (消费者进程):
    begin
      repeat
        P(full);
        P(mutex);
        取走产品消费;
        V(empty);
        V(mutex);
      until false
    end
```

在计算机系统中的许多问题都可以将它归结为生产者和消费者问题。

(3)TIMS系统后台进程的生产者及消费者。TIMS系统后台进程中的数据采集、拆包处理和数据分析是互为生产者和消费者的，它们间共用某数据区。

①数据采集与拆包处理。数据采集与拆包处理作为后台进程的两个线程，数据采集是生产者，拆包处理是消费者。数据采集线程专门负责从交换机实时接收数据(即生产产品)，拆包处理将原始数据拆包为可解释的结构数据。每当接收到数据，则进行拆包，否则拆包处理处于等待状态。

②拆包处理与数据分析。拆包处理与数据分析也是后台进程的两个线程，拆包处理是生产者，数据分析是消费者。拆包处理拆出数据后，数据分析线程对数据进行分析、转发等工作。

Win32提供的线程同步方法

Microsoft Win32 API提供了多种方式来协调多线程之间的运行。

1. 等待函数

等待函数允许线程将自己锁住。当调用一个等待函数后，系统检查定义的同步对象的状态以及其他能引起返回的条件，如果这个初始状态不能指定条件并且定义的时间已过，则调用线程进入一个有效等待状态，在其等待中几乎不占用处理机的时间。

等待函数有针对单对象的waitForSingleObject，waitForSingleObjectEx和针对多对象的wait For Multiple Objects，wait For Multiple OjbectEx以及Msg Wait For Multiple Objects.

2. 同步对象

同步对象的句柄可以在等待函数中定义，用以协调多线程的运行，同步对象有两种状态，一是信号的(signaled)，允许等待函数返回；二是非信号的(nosignaled)，阻止函数返回。多个进程可有一个相同的同步对象使内部同步成为可能。

3. 互斥对象

如果互斥对象不属于任何线程则它是带信号状态的同步对象，若它属于某个线程则是非信号的。一个线程某一时刻只能有一个互斥对象。互斥对象包含多个方式实现。

· 信号量对象：运行函数有：

CreateSemaphore，OpenSemaphore，Releasesemaphore.

· 事物对象：运行函数有：

SetEvent，Pulseevent，CreateEvent，Openevent.

· 内部进程同步：

TIMS多线程的实现

1. 建立多线程

TIMS系统后台采集工作做为一个进程，在它启动运行时，建立多个线程。因为系统接收处理的数据以交换机局为单位的，拆包处理、数据分析也是以局数据为处理单位。因此后台进程初始运行时为每个局建立上述三个线程。代码示意如下：

```
DWORD   main(VOID)
    DWORD   dwThread ID, dwThrdParam =1;
    HANDLE   hThread;
    WORD   postnum;
```

```
        .
        .
        .
          for(postnum=0; postnum<POSTNUM;
postnum++){
          hThread=Create Thread(
        NULL,      //无安全属性
        0,          //缺省栈地址
       (LPTHREAD_START_ROUTINE) ThreadFunc,
//线程函数
        &dwThreadParam,     //线程函数参数
        0,                  //缺省建立标志
        &dwThread Id ) ;        //返回线程标识符
        .
        .
        .
    }
```

2. 线程访问的数据临界区

所谓临界区，即多个进程（线程）共享的资源，在某一时刻只允许一个进程（线程）访问。

(1)数据采集与拆包处理的数据临界区。数据采集与拆包处理之间共享的是实时采集到的告警或话务报告，将其放在共享数据区内，同一时间内只有一个数据采集线程或拆包处理线程对其进程写访问，同时这个临界区还起到间接同步二线程的作用。

(2)拆包处理与数据分析的数据临界区。拆包处理与数据分析之间共享的数据是拆包处理后写入数据库表中的数据，同一时间内只有一个拆包处理线程或数据分析线程对其进行写访问。

(3)线程同步的实现。以上二对线程间同步均采用信号量对象的方法，同时对共享数据作区等待，当数据采集线程采集到数据，则将结果写入共享数据区，激活拆包处理线程；若无报告处理，则拆包线程睡眠。共享数据区的大小能较灵活地根据实际需要配置，实际上，需要同步的二线程并行速度相当，一般情况下对共享数据区的读、写速度较均衡。

线程同步用信号量对象方法，首先用 Create Semaphore 函数建立信号量，并定义初始值和最大计数值。

在线程进入临界区之前，用 WaitForSingleObject 函数决定是否允许进入，如果等待函数的超时参数被设为零，并且信号量是无信号状态，则函数立即返回。

```
DWORD   dwWaitResult;
// 申请进入临界区
dwWaitResult = WaitForSingleObject(
        hSemaphore;// 信号量句柄
        0L );//0 秒间隔超时
swith (dwWaitResult)  {
        // 信号对象是信号状态
     case   WAIT_OBJECT_O:
        // 可以进入临界区
          break;
        // 信号量设为无信号状态
     case   WAIT_TIMEOUT:
        // 不能进入临界区
     break;
     }
```

当线程退出临界区时，用 Release Semaphore 函数增加信号量的值。

```
    // 增加信号量值
    if(!ReleaseSemaphore(
      hSemaphore, // 信号量句柄
      1, // 信号量计数加 1
      NULL)){
      // 错误处理
      }
```

实现效果

这种多线程方法在TIMS系统中，解决了多事物的并发，提高了系统的效率和质量。■