Hans Journal of Data Mining 数据挖掘, 2016, 6(1), 9-14
Published Online January 2016 in Hans. http://www.hanspub.org/journal/hjdm
http://dx.doi.org/10.12677/hjdm.2016.61002

Hans汉斯

Efficiency Optimization Methods of the Signal Level Real-Time Simulation Software

Haijun Peng, Yugang Wang, Sichen Wang

Naval Aeronautical Engineering Academy Qingdao Branch, Qingdao Shandong
Email: flyingmanw@163.com

Received: Dec. 26th, 2015; accepted: Jan. 10th, 2016; published: Jan. 14th, 2016

Abstract

Signal level simulation software to calculate needs great amount, under present conditions should try to optimize hardware program code, in order to obtain the best efficiency. Compile layer, determinants of program optimization algorithm gradually shift from layer to layer, grammar, and then from the programmer to the compiler. According to the radar signal level simulation engineering practice, presented at the algorithm level optimization mathematical model, the software model and loop optimization algorithm optimization methods presented in the syntax layer optimization pointers and embedded functions, etc., with strong problem targeted signal level simulation of other areas also have reference value.

Keywords

Software Engineering, Real-Time Simulation, Efficiency Optimization

信号级实时仿真软件的效率优化方法

彭海军，王玉刚，王思臣

海军航空工程学院青岛校区，山东 青岛
Email: flyingmanw@163.com

收稿日期：2015年12月26日；录用日期：2016年1月10日；发布日期：2016年1月14日

文章引用: 彭海军, 王玉刚, 王思臣. 信号级实时仿真软件的效率优化方法[J]. 数据挖掘, 2016, 6(1): 9-14.
http://dx.doi.org/10.12677/hjdm.2016.61002

摘　要

信号级仿真软件计算量极大，在现有的硬件条件下应当尽力优化程序代码，以期获得最好的运行效率。从算法层到语法层、再到编译层，程序优化的决定因素从程序员到编译器逐步转移。本文根据雷达信号级仿真工程实践，在算法层提出了数学模型优化、软件模型优化和循环算法优化等方法，在语法层提出了指针使用和嵌入函数等优化方法，具有较强的问题针对性，对其它领域的信号级仿真也具有参考价值。

关键词

软件工程，实时仿真，效率优化

1. 引言

对于大型软件的编制，程序效率非常重要。在没有严格时间限制和存储空间限制的情况下，程序效率仅仅是影响到工作完成的时间。如果程序计算量很大，而且有严格的时间要求，比如对雷达系统进行信号级的实时仿真，程序效率就成为软件是否能成功的一个决定性因素。当然硬件的改善也可以适当缓解这一需求，但是硬件水平总要受到时代限制，并且还要考虑成本因素。因此，优化程序结构，提高程序效率，对于降低硬件成本，完成仿真任务具有举足轻重的作用[1]。程序的运行效率都是一个值得重视，并为之付出努力的问题。但是程序性能的优化也是一门复杂的学问，需要很多的知识，然而并不是每个问题的解决都能很好的利用好，如何提高程序的运行效率，下面本文通过实例进行介绍学习。

2. 程序效率的含义

程序效率包括两个方面的意义：程序的时间效率是指程序的运行速度要快；程序的空间效率是指程序在运行时的占用内存空间要少。提高程序效率也就是对程序进行优化。

程序对运行速度的要求是比较直观的一个要求，但是程序速度微量的提高，最终用户并不一定能感觉到，所以在提高程序时间效率的时候先要考虑值不值得。程序的空间效率是程序效率的另一个方面，特别是在 Windows 这种操作系统下，具有较小内存映象的程序，其引发页面错误的机会更少[2]。因为在程序出现错误时，操作系统必须从磁盘上重新加载内存，这种操作的代价是很昂贵的。时间效率和空间效率都具有重要意义。有时候我们在提高程序时间效率的同时会牺牲程序的空间效率，这时就应该权衡利弊，努力找到兼顾两者的最佳方法。

3. 提高程序效率的方法

提高程序效率可以在多个层次上进行考虑，最高层次是算法级别的优化，只能由程序员来进行。最低层次的优化只能由编译器来进行[3]。还有一些优化既可以由编译器进行，也可以由程序员来进行，这时候程序员应该适当的介入，不然编译器将错过一些优化的机会。关于编译层的优化在很多软件领域的文献里都有提到，此处不再赘述。本文主要研究信号级仿真软件中在算法层和语法层提高程序效率的方法，这些方法对其它领域软件编制同样具有重要的参考意义。

3.1. 改进程序算法

改进程序算法对提高程序效率的作用是最大的。无论是在开始编程之前还是编程的过程中，都可以考虑算法的效率问题。

3.1.1. 改进数学模型

在某型雷达的信号级仿真中，有一个部件需要对两个信号积分后比较大小，数学模型如下所示

$$y=\int_a^b f(x)\mathrm{d}x-\int_a^b g(x)\mathrm{d}x \tag{1}$$

因为信号采样率高达 1 兆赫兹，而积分运算采用数值方法，所以模型中的积分模块将消耗大量的计算时间，观察上式可知，数学模型可作如下优化

$$y=\int_a^b\left[f(x)-g(x)\right]\mathrm{d}x \tag{2}$$

这样数学模型中只包含一个积分模块，增加的减法运算速度比积分运算快得多，所以大大提高了时间效率[4]。

3.1.2. 改进软件模型

某些数学模型如果直接转化为软件模型，将会消耗很大的系统资源。比如雷达接收机中视频积累模块，在该模块中，每次新产生数据后都要对最近产生的几十或一百多个数据进行累积相加[4]。设备中的做法是对历史上产生的数据进行延时，然后在新的数据产生时进行累加。最直观的软件实现方法是对应每个延时单元设立一个存储单元，用存储单元的转存表示延时。图 1 表示了 8 次累加的情况。

图中左边 7 个存储单元依次对应 7 个延时单元，D7 是 D0 单元数据经过了 7 次延时后的结果，D0 单元存储最新产生的数值。每产生一个新的视频数据，D6 到 D0 的值都要向左转存一次，然后将新数据写入 D0 单元，然后对 8 个单元求和。因为视频积累往往需要几十个数据延时相加，而且工作频率很高，所以消耗了大量时间。在仔细观察模型之后，我们发现，我们只需要在每次求和前更新这些延时单元中的一个数据就可以了。如果每次用最新产生的数据覆盖最早产生的数据，那么其余数据都不用转存。所有参与求和的数据按顺序循环存放，只需要添加一个指针指示最新数据存放位置，这样同样可以保证 8 个单元中都是最近产生的数据。如图 2 所示。

优化后的程序以维护一个指针的代价节省了几十次转存操作，大大提高了程序时间效率，并且不影响程序的空间效率。

Figure 1. Original simulation model of accumulation module
图 1. 延时相加模块的直接软件模型

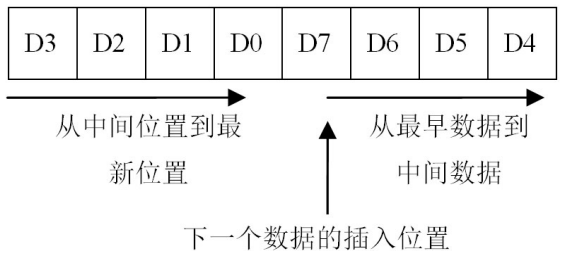

Figure 2. Improved simulation model of accumulation module
图 2. 延时相加模块的改进软件模型

3.1.3. 改进循环算法

在循环体内改进程序算法，是提高程序效率最有效的方法之一，因为每一处改进的效果都根据循环次数而倍增。改进循环的效率大体可以从两方面考虑：

(1) 减少循环次数

首先要消除一些不必要的循环。比如在雷达的信号检测模块中，每一个回波数据都需要与先后相邻时间产生的共 32 个回波数据的平均值进行比较，这个过程暂时称为检测。一种直接的解决办法是先存储一定数量的回波数据(比如 100,000)，然后用一个二重循环完成这一任务，如代码段(a)所示。E 是存储回波数据的浮点型数组，各元素按照产生时间的先后顺序排列，sum 是相邻时间回波数据之和。由于计算需要，在边界上不得不忽略若干数据。

```
float sum;              //sum of 32 sample data
float E[100000];        //sample data
for( int i=16;i<99984;i++)
{
  sum=0;
  for(int j=-16;j<0;j++)
        sum+=E[i+j];
  for(j=1;j<=16;j++)
        sum+=E[i+j];
  if ( E[i]>sum/32)
  {      //Do some thing when
         sample datum is above average}
}
```

(a) 带内循环的检测过程

经仔细观察，代码段(a)中的两个内层循环可以用一个连加计算式代替，这样对每次检测都节省了 32 次循环的开销，连加计算式还可以充分发挥处理器的硬件优化能力，因此可以显著提高程序效率，改进后的代码如(b)所示。

```
for( int i=16;i<99984;i++)
{
  sum=E[i-16]
        +E[i-15]
        ...
        +E[i-1]
        +E[i+1]
        ...
        +E[i+15]
        +E[i+16];
  if ( E[i]>sum/32)
  {// Do some thing when
      sample datum is above average }
}
```

(b) 去掉内循环的检测过程

进一步考察，外层循环也可以去掉，就是说不需要先存储 100,000 个回波数据，在产生回波数据的同时进行检测。在产生了 33 个数据之后就可以开始对第 16 个回波数据进行检测，代码如(c)所示。其中 newData 是最新产生的回波数据，而紧跟的检测是针对之前第 16 个回波数据进行检测。这样优化不仅提高了程序效率，而且实现了雷达仿真模型中对回波信号的准实时检测[4]。

```
E[i]=newData;
if(i>=32)
{
  sum=E[i-32]
        +E[i-31]
        ...
        +E[i-17]
        +E[i-15]
        ...
        +E[i-1]
        +E[i];
  if ( E[i-16]>sum/32)
  {// Do some thing when
     sample datum is above average }
}
```

(c) 去除所有循环的检测过程

(2) 减轻循环内部工作量

减轻循环体内部的工作量是提高程序效率最有效的方法，因为每一点改进的效果都按照循环次数而倍增。在这方面最常见的一个优化技术是不变式的提升[5]。就是将循环体中值不变化的计算子式提升到循环外部。下面是有一个简单的例子：

```
for(int i=0;i<10;i++)
    nArray[i]=a+b;
```

若果通过将循环体中的不变子式提出到循环外面，则可以节省 9 次求和运算，以较小空间代价获得了巨大的时间效益：

```
temp=a+b;
for(int i=0;i<10;i++)
    nArray[i]=temp;
```

3.2. 改进语句效率

这一部分的优化有时编译器也可以自动处理，在某些方面编译器处理的效果比自己动手还更好，因为编译器从全局考虑了程序的运行效率。不过程序员应该对明显可以优化的地方加以优化，不能把所有工作交给编译器[6]。

3.2.1. 指针的运用

(1) 作为函数参数

函数的参数有值传递和地址传递两种，如果参数不是编程语言的内部数据类型，那么指针传递，也就是地址传递的效率较高。当然如果参数是一个对象的指针，在 c++里还允许用对象的引用来代替。这两种方法都可以省去临时对象的构造和析构函数，从而提高效率。

(2) 作为数组名的指针

在对数组中的元素进行连续的读写操作时，直接用指针运算比用数组下标效率高，以下面这段代码作为例子：

```
for(int i=0;i<10;i++)
    nArray[i]=4*i;
```

修改后能获得更高的效率：

```
for(int i=0, p=nArray; i<10;i++)
    *p++=4*i;
```

需注意的是如以随机方式访问数组，那么用指针未必效率高[7]。

3.2.2. 嵌入函数的使用

某些高级语言支持嵌入函数。如果一个函数是嵌入函数，那么在编译的时候，对该函数调用代码将被函数体直接替换。通常调用函数之前，程序先要压栈以保存当前的工作环境，程序返回之后还须恢复调用前的环境，这会消耗相当多的时间。在调用嵌入函数时，将不再有函数调用指令，所以省去了压栈和出栈的操作。处理器直接按照函数体内的语句顺序执行，这也使处理器能够更顺利地预取指令。嵌入函数以较小的空间代价换取了较大的时间效率。当函数体较小时，嵌入函数甚至可以减小程序的大小。

但是并非嵌入函数总是能提高程序效率，如果执行函数体内代码的时间相对于函数调用的时间要大得多，那么程序效率的提高会很少，比如说函数体内有循环。另外如果函数体内代码过长，那么在编译的时候会导致程序总代码量过大，这也是不可取的。

3.2.3. 其它提高程序效率的途径

还有很多其它提高程序效率的办法，比如用移位操作降低指令强度，插入汇编代码，使用 register 变量等等，限于篇幅不在此作进一步阐述。值得注意的是，像插入汇编代码和使用低级操作符等方法会影响程序的兼容性，在某些高级编译器中 register 变量已经完全由编译器管理。在这些方面进行优化，程序员更应该谨慎处理，以求得到最佳效果。

4. 结束语

软件行业有一条 80-20 准则：大约 20%的代码使用了 80%的程序资源；大约 20%的代码耗用了大约 80%的运行时间，我们在优化程序效率时一定要找出程序效率的“瓶颈”，不要在无关紧要处浪费功夫。另外，也不能一味地追求程序的效率，而应当在满足程序正确性、可靠性、健壮性、可读性等的前提下，设法提高程序的效率。总之，本文通过改进程序算法、改进语句效率等途径，通过实例印证了如何提高程序运行效率的方法，并着眼于程序的整体性能，在最关键处提高程序效率。本文所述方法已经应用于某型雷达实时仿真项目中，并且收到了良好效果，可以推广应用到类似的教学和软件编制工作中。

参考文献 (References)

[1] 党婧. 软件工程的开发及应用分析[J]. 电子技术与软件工程, 2015(2): 66-67.

[2] 沈非一, 张延园, 林奕. 嵌入式实时系统中动态内存管理算法的设计与实现[J]. 计算机与现代化, 2015(7): 103-107.

[3] 姜军, 王超, 尉红梅. 一种局部寄存器分配的优化策略[J]. 计算机应用与软件, 2013(12): 215-217.

[4] 何炎祥, 陈勇等. 绿色编译优化策略: 研究综述[J]. 计算机科学与探索, 2013, 7(8): 673-690.

[5] 翟京, 孙海波, 王鹏, 冷军强. 基于 vissim 仿真的交叉口左转车流交通组织[J]. 科学技术与工程, 2012, 12(7): 1687-1690.

[6] 祁宏生, 王殿海, 陈松. 基于综合饱和度的单点信号控制方法[J]. 哈尔滨工业大学学报, 2012, 44(2): 134-137.

[7] 包敬海, 韦美兰. 汇编和 C 混合编程的研究[J]. 电脑与电信, 2011(1): 30-31.