



先进编译实验室
Advanced Compiler

程序性能的度量指标及 优化流程

嘉宾：王梦园



先进编译实验室
Advanced Compiler



2.1 程序性能的 度量指标

- ➡ 程序执行时间
- ➡ 计算与访存效率
- ➡ 吞吐量与延迟
- ➡ 加速比
- ➡ Amdahl定律
- ➡ Gustafson定律





程序的执行时间是判断程序性能优劣较为简单的方式之一，在使用相同计算设备且保证程序正确的前提下，程序运行时间越短意味着其性能越高效。

例如：在linux下time命令可以获取到一个程序的执行时间，包括程序的实际运行时间(real time)，以及程序运行在用户态的时间(user time)和内核态的时间(sys time)，命令行为time ./a.out，执行结果如下：

```
real    0m5.724s
user    0m5.689s
sys     0m0.031s
```





实际优化过程中，某些情况下需要对少量代码精确计时，采用普通的计时方法测试的结果误差可能较大，此时可利用嵌入汇编rpcc指令或多次执行后取平均值的方式尽量减少结果误差。

```
unsigned long rpcc(){  
    unsigned long result;  
    unsigned hi,lo;  
    asm volatile("rdtsc":"=a"(lo),"=d"(hi));  
    result = (((unsigned long long)lo)/(((unsigned long long)hi)<<32));  
    return result;  
}  
  
int main(){  
    unsigned long b[m],start,end,k;  
    for(j=0;j<10;j++){  
        start = rpcc();  
        fun(16);  
        end = rpcc();  
        b[j]=end -start;  
        k+=b[j];  
    }  
}
```





计算效率是指实测浮点性能与理论浮点峰值性能之比。而访存效率是指程序的有效访存带宽与存储器理论带宽之比，其中带宽是计算平台每秒内存交换量的最大值。

当程序的访存效率接近于1时，说明程序已经将整个存取器的带宽都利用了起来，与之对应的当访存效率远小于1，则说明存储带宽利用率较低，程序还有一定的访存优化空间。





吞吐量和延迟是衡量软件系统最常见的两个指标。但高吞吐量并不意味着低延迟，高延迟也不代表吞吐量变小，它们之间的关系并不是简单的一一对应。延迟测量的是用于等待的时间，广义来说，延迟可以表示所有操作完成的耗时，例如一次应用程序请求、一次数据库查询、一次文件系统操作等，可以表示从单击链接到屏幕显示整个页面加载完成的时间。





加速比speedup是指同一个任务在单处理器系统和并行处理器系统中运行消耗的时间的比率，用来衡量并行系统或程序并行化的效果，也可以用于衡量程序优化前后的效果，由于加速比是一个相对比值，因此在保证程序正确性的前提下加速比数值越大，代表着优化的效果越显著。计算加速比的公式为：

$$\text{加速比} = \text{优化前的执行时间} / \text{优化后的执行时间}$$



Amdahl定律

8



先进编译实验室
Advanced Compiler

Amdahl定律将程序划分为可加速与不可加速两大部分，程序总的加速比S是一个关于程序中这两部分所占比例以及可加速部分性能加速程度的函数，用公式表示为：

$$S=1/((1-a)+a/n)$$

其中a为并行计算部分所占比例，n为并行计算部分获得的加速比。例如当a=50%，n=1.15时， $S=1/((1-0.50) + (0.50/1.15)) = 1/(0.50 + 0.43) = 1.08$ 。此外若应用程序有50%的代码是串行部分，那么该程序最终所能够达到的加速比上限为 $1/0.5=2$ 。



先进编译实验室
Advanced Compiler



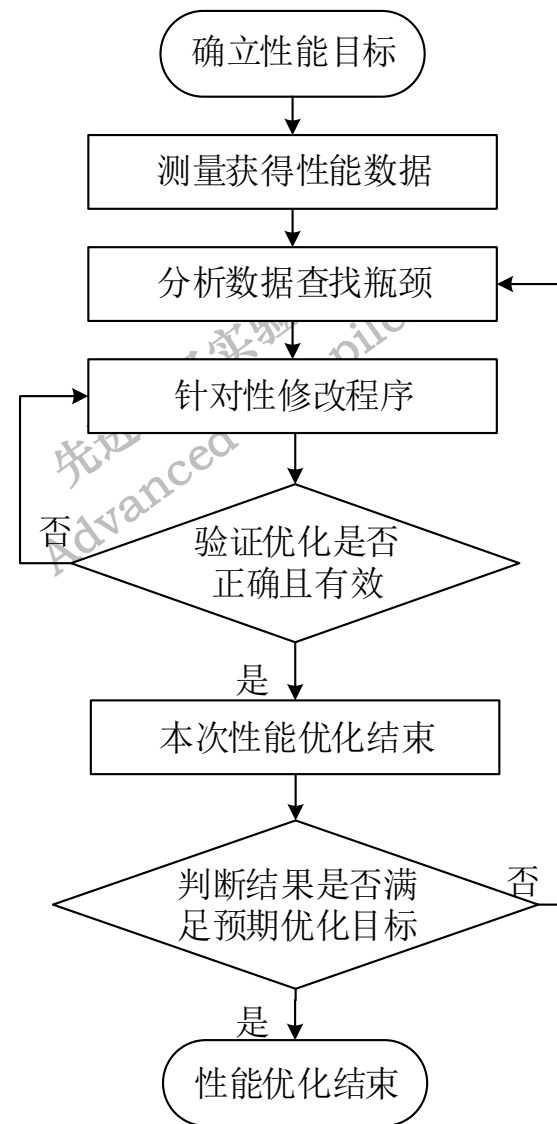
Gustafson定律

9



先进编译实验室
Advanced Compiler

在进行程序优化时之前，除需要了解程序性能常用的度量指标外，优化人员还需要大致了解程序优化的流程及具体实施步骤，从而可以更加高效顺利的开展程序优化。程序性能的常用优化流程如图所示。



先进编译实验室
Advanced Compiler



AI框架发展白皮书（2022年）

<https://syncedreview.com/2020/12/14/a-brief-history-of-deep-learning-frameworks/>

先进编译实验室
Advanced Compiler

先进编译实验室
Advanced Compiler





AdvancedCompiler

Tel: 13839830713



先进编译实验室
Advanced Compiler

程序性能优化指标与常用流程 II

嘉宾：王梦园



先进编译实验室
Advanced Compiler



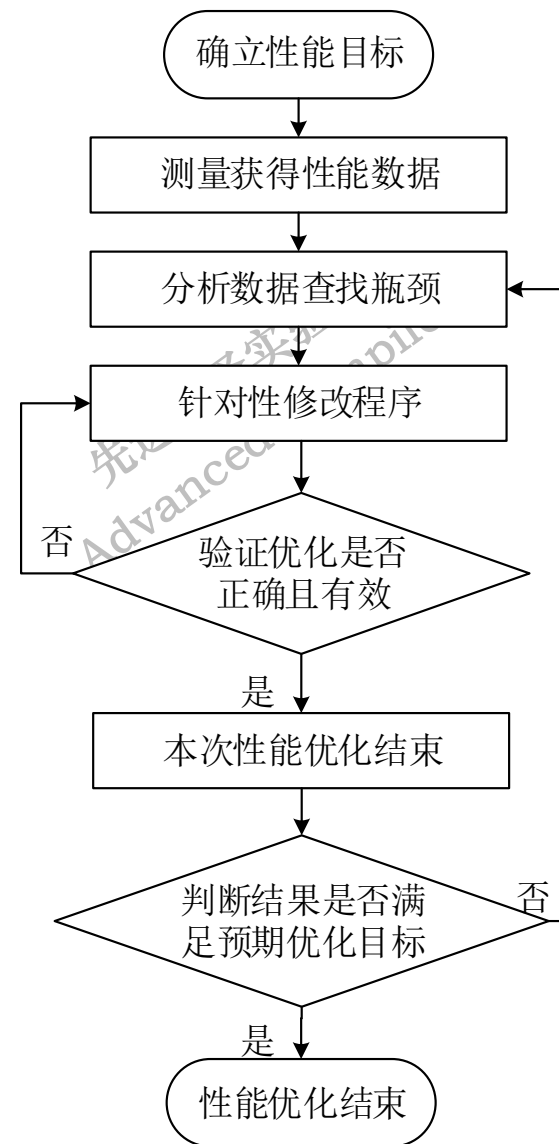
程序性能优化常用流程



先进编译实验室
Advanced Compiler

13

在进行程序优化时之前，除需要了解程序性能常用的度量指标外，优化人员还需要大致了解程序优化的流程及具体实施步骤，从而可以更加高效顺利的开展程序优化。程序性能的常用优化流程如图所示。



先进编译实验室
Advanced Compiler



以矩阵乘优化为例，假设优化后相比优化前需要达到1.5倍的性能目标。在确立性能目标后，需要对程序进行性能度量，获得程序运行时的各方面的数据。

```
void matrixmulti(int N, int x[n][n], int y[n][n], int
z[n][n]){
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        r=0;
        for (k = 0; k < N; k++) {
            r = r + y[i][k] * z[k][j];
        }
        x[i][j] = r;
    }
}
```

函数名	函数执行时间占总运行 时间百分比	函数本身执行时间(s)	程序总运行时间(s)	函数被调用次数
matrixmulti	98.08%	3.07	3.13	1
main	1.92%	0.06	—	—



完成一次性能优化后，程序员需要判断当前的程序性能是否满足需求，根据性能结果决定是否需要开启下一轮性能优化，直至满足最初的程序性能目标后结束优化流程。

```
void matrixmulti(int N, int x[n][n], int y[n][n], int z[n][n]){  
    for (k = 0; k < N; k++) {  
        //循环交换  
        for (i = 0; i < N; i++) {  
            r = y[i][k];  
            for (j = 0; j < N; j++) {  
                x[i][j] += r * z[k][j];  
            }  
        }  
    }  
}
```


函数名	函数使用时间占所有运行时间百分比	函数本身执行时间(s)	程序总运行时间(s)	函数被调用次数
matrixmulti	97.84%	2.26	2.31	1
main	2.16%	0.05	—	—





矩阵乘代码优化后得到的运行数据如下表。此时其加速比相比源程序达到了1.56倍，满足设立的优化目标，可以结束此次程序优化流程。

```
void matrixmulti(int N, int x[n][n], int y[n][n], int z[n][n]){
for (k = 0; k < N; k++) {//循环交换并展开四次
    for (i = 0; i < N; i++) {
        r = y[i][k];
        for (j = 0; j < N-4; j+=4){
            x[i][j] += r* z[k][j];
            x[i][j+1] += r* z[k][j+1];
            x[i][j+2] += r* z[k][j+2];
            x[i][j+3] += r* z[k][j+3];}
        for(;j<N;j++)
            x[i][j] += r* z[k][j];
    }
}
```

函数名	函数使用时间占有所有运行时间百分比	函数本身执行时间(s)	程序总运行时间(s)	函数被调用次数
 matrixmulti	98.53%	2.01	2.04	1
main	1.47%	0.03	—	—



程序性能优化常用流程



先进编译实验室
Advanced Compiler

17

1

测试用例的构造。需要构造出具有待调优程序特征的测试用例

2

性能指标的选择。根据不同的性能指标，分析策略也不尽相同

3

正确性的判定。为避免出现测试不完全的情况，可以增加测试的广度

4

保持程序的可读性。保证程序的可移植性、可读性、可维护性和可靠性

5

程序性能优化结束的时机。优化人员需知道程序与最优状态的差距，选择合适的时机结束程序性能优化。



先进编译实验室
Advanced Compiler



AI框架发展白皮书（2022年）

<https://syncedreview.com/2020/12/14/a-brief-history-of-deep-learning-frameworks/>

先进编译实验室
Advanced Compiler

先进编译实验室
Advanced Compiler





AdvancedCompiler

Tel: 13839830713