程序优化

施朱鸣

11 月 4 日

Outline

优化编译器

优化区块

处理器级优化

优化编译器

减少重复计算同一个数

```
void set_row(double *a, double *b,
   long i, long n)
{
    long j;
    for (j = 0; j < n; j++)
        a[n*i+j] = b[j];
}
```

```
    long j;
    int ni = n*i;
    for (j = 0; j < n; j++)
        a[ni+j] = b[j];
```

用简单的计算代替

乘除法开销大，视情况用累加或者位移运算实现 $16 * x \rightarrow x << 4$

```
for (i = 0; i < n; i++) {
  int ni = n*i;
  for (j = 0; j < n; j++)
    a[ni + j] = b[j];
}
```



```
int ni = 0;
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++)
    a[ni + j] = b[j];
  ni += n;
}
```

用简单的计算代替

分解复杂算式，把重复计算的数提出来

```
/* Sum neighbors of i,j */
up =    val[(i-1)*n + j  ];
down =  val[(i+1)*n + j  ];
left =  val[i*n     + j-1];
right = val[i*n     + j+1];
sum = up + down + left + right;
```

```
long inj = i*n + j;
up =    val[inj - n];
down =  val[inj + n];
left =  val[inj - 1];
right = val[inj + 1];
sum = up + down + left + right;
```

优化区块

编译器优化的局限性

- 为了不编译错，编译器只进行安全的优化
- 优化只在局部（procedures）进行
- 基于静态信息的优化
- 编译器不知道内存的实际情况

消除低效率循环

编译器看不到跨越函数的情况，不敢优化

“边带效应”：做了编译器没意识到的事情

```
void lower(char *s)
{
  size_t i;
  for (i = 0; i < strlen(s); i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

```
void lower(char *s)
{
  size_t i;
  size_t len = strlen(s);
  for (i = 0; i < len; i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

消除不必要的访存

利用局部变量，但如果 B 和 A 相关可能会造成错误，编译器不敢优化

```
/* Sum rows is of n X n matrix a
   and store in vector b  */
void sum_rows1(double *a, double *b, long n) {
    long i, j;
    for (i = 0; i < n; i++) {
        b[i] = 0;
        for (j = 0; j < n; j++)
            b[i] += a[i*n + j];
    }
}
```

```
double A[9] =
  { 0,   1,   2,
    4,   8,  16},
   32,  64, 128};

double B[3] = A+3;

sum_rows1(A, B, 3);
```

减少不必要的访存

- 一段内存有多个别名时，如果编译器进行优化可能发生错误，所以要人工进行优化。
- C 中指针指来指去，编译器感到害怕

处理器级优化

致谢

致谢

祝大家期中顺利，谢谢聆听！