加速比量化分析

肖峰
武汉大学计算机学院
2014301500048

December 25, 2016

这篇文章，我们详细地量化分析加速比，首先我们分析为何使用SIMD会有加速，接着我们分析了加速比随迭代次数增加可能的原因。

1 Preliminary

我们用Map[M][N]来表示整个图

$$Map = \begin{array}{c} \\ row_1 \\ row_2 \\ \vdots \\ row_m \end{array} \begin{array}{c} \begin{array}{cccc} col_1 & col_2 & \cdots & col_n \end{array} \\ \begin{pmatrix} Ce_{11} & Ce_{12} & \cdots & Ce_{1n} \\ Ce_{21} & Ce_{22} & \cdots & Ce_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ Ce_{m1} & Ce_{m2} & \cdots & Ce_{mn} \end{pmatrix} \end{array} \tag{1}$$

其中Ce_{ij}表示第i行j列Cell的存活情况,1为存活，0为死亡。

2 加速比成因

这里我们暂时先忽略Cache命中率提高带来的影响

2.1 串行算法

在一轮迭代过程I中:

原有算法遍历每一个Map[i][j] (Ce_{ij}) 的周围3×3方格，这种遍历可以拆解为行遍历I_r与列遍历I_c。

$$I = I_r + I_c \tag{2}$$

设在I_r中遍历所需CPU周期为Cy_r，($Cy_r \propto$ M)。在I_c中遍历所需CPU周期为Cy_c，($Cy_c \propto$ N)。故串行算法一轮迭代所需周期总数C_1为

$$C_1 = Cy_c + Cy_r \tag{3}$$

2.2 SIMD算法

在一轮迭代过程I中:

对每一个Map[i][j] (Ce_{ij})，我们先后进行了行遍历I_r与列遍历I_c。

$$I = I_r + I_c \tag{4}$$

设向量长度为L:

则行遍历在I_r中遍历所需CPU周期为Cy_r/L，($Cy_r \propto$ M)。在I_c中遍历所需CPU周期为Cy_c不改变。

故SIMD算法一轮迭代所需周期总数C_2为

$$C_2 = Cy_r/L + Cy_c \tag{5}$$

故加速比SP为

$$SP = \frac{Cy_r + Cy_c}{Cy_r/L + Cy_c}$$

在实际代码中，L=4

当$Cy_r = Cy_c, SP = 1.6$,这与实验中迭代次数为50时结果相近，符合实验结果，如Figure 1

3 加速比趋势

这里我们总结了两条原因:

迭代所占比例变高当迭代次数越多，I所花周期C相比于其他过程（计时，初始化），所占比例越高，所以I的改进对更多次数的迭代有更大的提高。

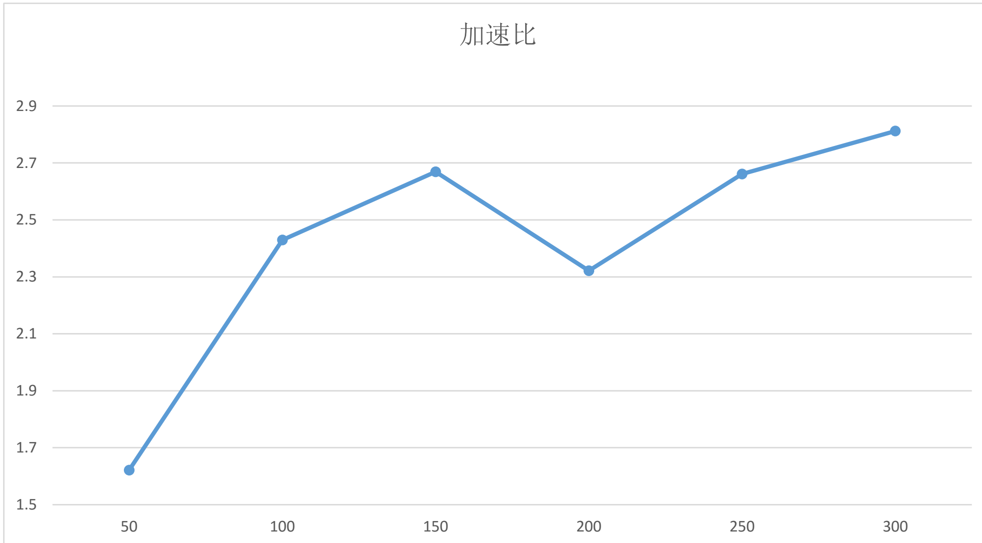
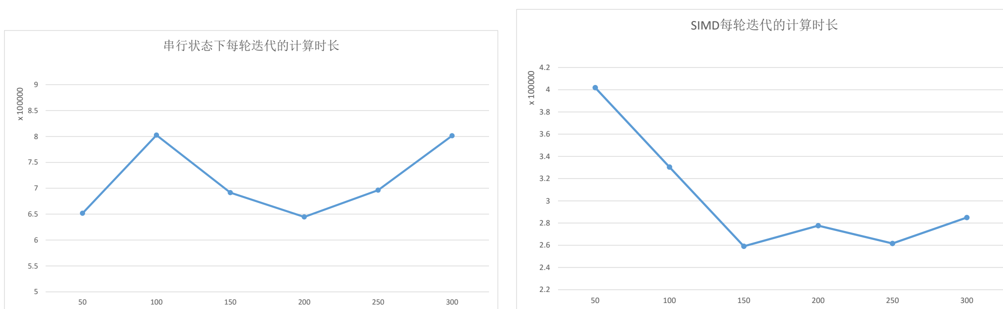


Figure 1: SP changes when iterate time increases.



(a) 串行程序 (b) SIMD 程序

Figure 2: The impact of the number of loop on the performance of each iteration of Conway game.

Cache命中率提高 Figure 2统计了在不同迭代次数下，平均每一轮迭代的时间变化。可以看到，随着迭代次数增加，串行程序的每轮平均执行时间执行时间在一定范围内抖动，而SIMD版本却均匀下降。这里说明迭代次数越多，SIMD版本的执行效率反而越高了，因为L1 L2层的cache被填充了更多与SIMD有关的数组。

References