CPU的时钟周期是非常短的，指令流水线（Instruction Pipeline）中的一个堵塞（或称之为气泡）就会导致时钟周期被浪费。CPU采取了乱序执行(Out-of-order Execution)及预测执行等等手段以减少时钟周期的浪费。Pipeline disruption reduces the effective instruction throughput by introducing extra delays in the pipeline, since branches constitute a large portion of all the executed instructions, the efficiency of handling branches is important.[1] 处理器按照输入数据和执行单元的可用性所决定的顺序执行指令,而不一定按原始的顺序执行。预测执行的本质在于提前执行指令而不管这些指令是否确实需要被执行，尤其是在分支指令中，后续执行分支的跳转与否取决于该分支指令的执行结果时。如果待该分支指令经过fetch, decode, execution之后再执行下一条指令的fetch, decode……那么在CPU的流水线上就会产生气泡，几个CPU时钟周期将会被浪费。对于预测执行来说，如果提前执行的指令恰好就是分支指令确定的那条指令，那么处理器就能提高工作效率。但是不可避免的是，没有根据的预测必然会造成预测的准确率低下，如果预测失败，则从fetch到execution的时间都会被浪费掉，在采用超长流水线设计的现代处理器中可能就要损失数十个时钟周期。预测执行在现代高性能处理器中被广泛地使用，为了规避预测失败的风险，处理器中需要引入一部分的电路作为分支预测器。

分支预测器(Branch Predictor)能在分支指令执行结束之前猜测哪一路分支将会被运行，用于提高分支预测的准确性，常见的分支预测技术有静态预测、动态预测、下一行预测、饱和计数、两级自适应预测器等。目前使用最多的是动态预测，即利用分支指令发生转移的历史来进行预测。

下列测试程序很好地体现了分支预测器的作用:

#include<bits/stdc++.h>

**using** **namespace** std;

**int** main()

{

**double** consumption0=0,consumption1=0;

**const** **int** length=20202;

**int** test[length];

**for**(**int** t=0;t<20;t++)

{

**for** (**int** j=0;j<length;j++)

{

*//用随机数填充数组*

test[j]=rand()%256;

}

*//记录第一次开始时间*

**clock\_t** start=clock();

**long** **long** sum=0;

**for**(**int** i=0;i<10000;i++)

{

**for**(**int** j=0;j<length;j++)

{

*//分支跳转的条件test[j]>128*

**if** (test[j]>=128)

sum+=test[j];

}

}

*//计算未排序前进行10000次loop消耗的时间*

consumption0+=(**double**)(clock()-start)/CLOCKS\_PER\_SEC;

sum=0;

**clock\_t** start1=clock();

sort(test,test+length);

**for**(**int** i=0;i<10000;i++)

{

**for**(**int** j=0;j<length;j++)

{

*//分支跳转的条件test[j]>128*

**if** (test[j]>=128)

sum+=test[j];

*//cout<<sum;*

}

}

*//计算排序后进行10000次loop消耗的时间*

consumption1+=(**double**)(clock()-start1)/CLOCKS\_PER\_SEC;

}

*//输出两次运算时间并计算提升效率*

cout << "排序前: "<<consumption0 << endl;

cout << "排序后: "<<consumption1 << endl;

cout<<"效率提升"<<100\*(consumption0-consumption1)/consumption0<<"%**\n**";

system("pause");

**return** 0;

}

（代码思路由同学提供）

当数组test中存放的数据是无序时，采用动态预测的分支预测器很难通过历史数据对下一条指令的跳转与否做出正确的判断，故运算时间长，而通过sort函数对数组进行排序后，动态预测就可以很好地根据test[i]的情况来预测test[i+1]的情况，这样运算的时间就被缩短了。

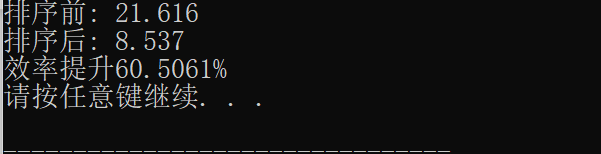


图1 运行结果

参考文献

[1] Shien–Tai Pan, Kimrning So, Joseph T. Rahmeh. Improving the Accuracy

of Dynamic Branch Prediction [J] ASPLOS V: Proceedings of the fifth international conference on Architectural support for programming languages and operating systems, 1992 :76–84. <https://staff.fnwi.uva.nl/s.polstra/psa2017/p76-pan.pdf>