1.SIMD 优化

所谓 SIMD (单指令多数据流）就是 Single Instruction Multiple Data 的简称，可以理解成能够同时操作多个数据，并把储存在大型寄存器的一组指令集。

SIMD 的数据类型：

__m64：64 位紧缩整数（MMX）。

__m128：128 位紧缩单精度（SSE）。

__m128d：128 位紧缩双精度（SSE2）。

__m128i：128 位紧缩整数（SSE2）。

__m256：256 位紧缩单精度（AVX）。

__m256d：256 位紧缩双精度（AVX）。

__m256i：256 位紧缩整数（AVX）。

注：紧缩整数包括了 8 位、16 位、32 位、64 位的带符号和无符号整数。

其中，选择对 SM3 的迭代压缩函数进行优化，因为压缩函数实际上是一个多次循环的过程，我们可以利用_m256i 的 256 位宽，同时对 8 路循环进行处理。辅助值在声明的时候就是 int32 类型，context->state[0]经过 set1_epi3 的初始化后，内部就会有 8 个相同辅助值。

SIMD 指令集可以使 1 条指令同时对 8 路数据进行加减乘除、与非或操作，发挥 CPU 性能，提升运行效率。

```
static void avx_decode(__m256i *output, unsigned char *input1, unsigned char *input2,
 unsigned char *input3, unsigned char *input4, unsigned char *input5,
 unsigned char *input6, unsigned char *input7, unsigned char *input8, unsigned int len)
{
    unsigned int i, j;
    for (i = 0, j = 0; j < len; i++, j += 4)
    {
        output[i] = _mm256_set_epi32(
            ((unsigned long int)input8[j]) | (((unsigned long int)input8[j+1]) << 8) ),
            ((unsigned long int)input7[j]) | (((unsigned long int)input7[j+1]) << 8)),
            ((unsigned long int)input6[j]) | (((unsigned long int)input6[j+1]) << 8)),
            ((unsigned long int)input5[j]) | (((unsigned long int)input5[j+1]) << 8)),
            ((unsigned long int)input4[j]) | (((unsigned long int)input4[j+1]) << 8)),
            ((unsigned long int)input3[j]) | (((unsigned long int)input3[j+1]) << 8)),
            ((unsigned long int)input2[j]) | (((unsigned long int)input2[j+1]) << 8)),
            ((unsigned long int)input1[j]) | (((unsigned long int)input1[j+1]) << 8)),
            );
    }
}
```

对压缩函数最后的 8 组数据循环赋值再异或进行优化

FOR j=0 TO 63

$$SS1 \leftarrow ((A \lll 12) + E + (T_j \lll j)) \lll 7$$
$$SS2 \leftarrow SS1 \oplus (A \lll 12)$$
$$TT1 \leftarrow FF_j(A,B,C) + D + SS2 + W_j'$$
$$TT2 \leftarrow GG_j(E,F,G) + H + SS1 + W_j$$
$$D \leftarrow C$$
$$C \leftarrow B \lll 9$$
$$B \leftarrow A$$
$$A \leftarrow TT1$$
$$H \leftarrow G$$
$$G \leftarrow F \lll 19$$
$$F \leftarrow E$$
$$E \leftarrow P_0(TT2)$$

ENDFOR

$$V^{(i+1)} \leftarrow ABCDEFGH \oplus V^{(i)}$$

```
_m256i input=_mm256_ster_epi32(input1,input2,input3,input4,inpu5,input6,input7,input8);
_m256i output=_mm256_ster_epi32(A,B,C,D,E,F,G,H);
```

2. 多线程优化

利用多线程并行工作，将待处理数据进行分块，达到同时运行多个线程，提升效率的目的。