组会报告

徐益

2018 年 8 月 30 日

1 工作内容

1. 实现基于 C 的 avx2-OMS 译码测试平台；
2. 提高 avx2-OMS 吞吐量。

2 提高 avx2-OMS 吞吐量的具体方法

2.1 改变数据结构使内存空间连续

原数据结构：

```c
typedef struct check_node
{
        int8_t degree;          // number of connective variable nodes
        int16_t *index;         // index of connective variable nodes
        float* message;         // message from cn to vn
        int8_t* message_fixed;  // fixed message from cn to vn
        __m256i* message_avx2;  // avx2 message from cn to vn
        int8_t pointer;         // pointer to the message that will be used
} check_node;
```

现数据结构：

```c
typedef struct nr15_ldpc_simd_t
{
        ......
        int8_t* degree; // number of connective check nodes(length:M)
        int16_t* index; // index of connective check nodes(length:M_whole)
        __m256i* cn_msg_avx2;   // avx2 message from cn to vn(length:M_whole)
        __m256i* llr_avx2;      // avx2 llr(length:N)
        __m256i** llr_addr;     // llr address from cn to vn(length:M_whole)
        __m256i* vn_msg_avx2;   // temp avx2 message from vn to cn(length:19)
} nr15_ldpc_simd_t;
```

2.2 使用更小的校验矩阵

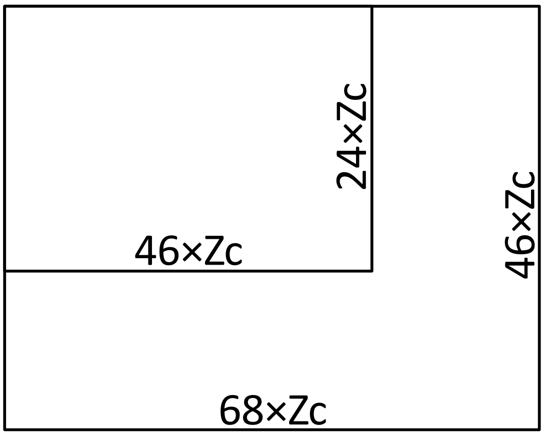


图 1: 优化前后校验矩阵尺寸变化

2.3 编译方式的优化

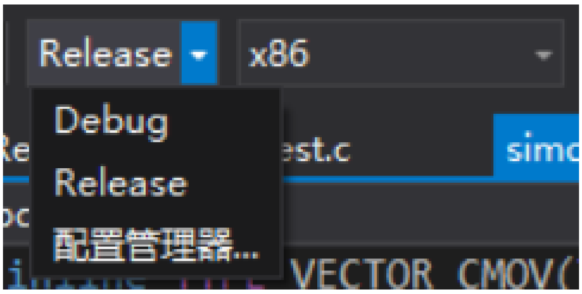


图 2: 项目版本的选择

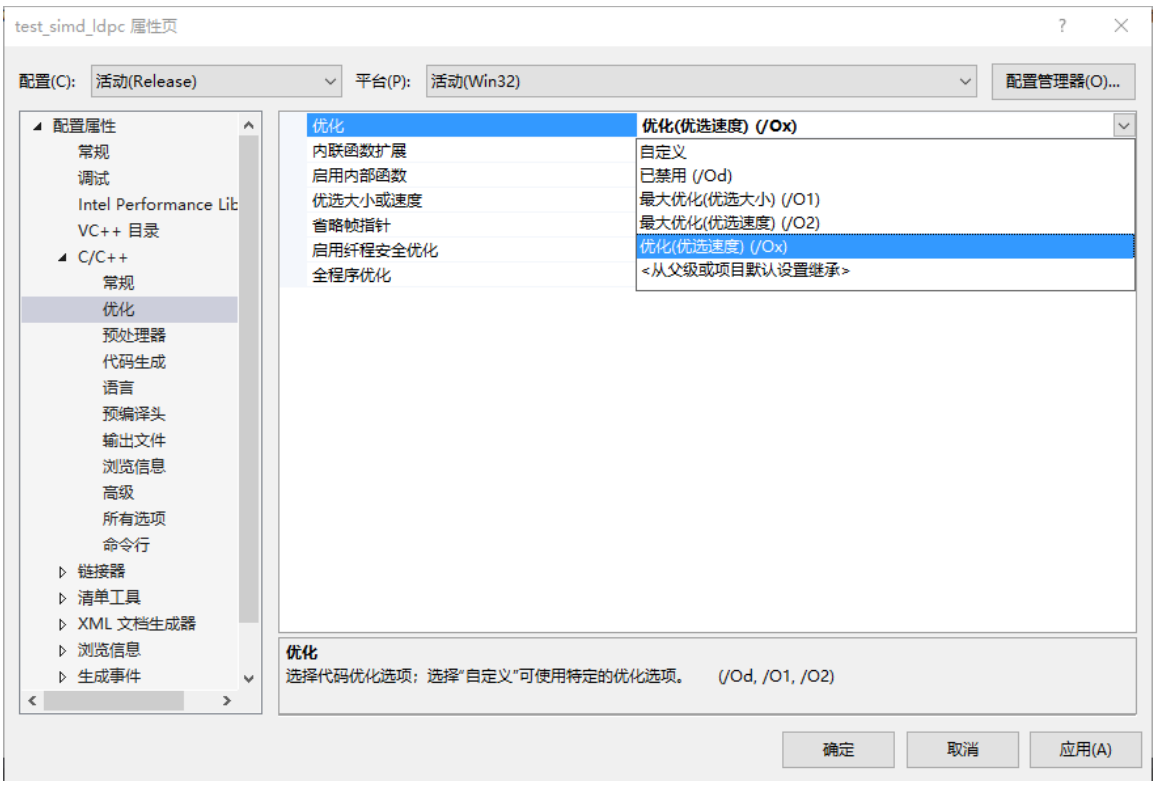


图 3: 优化方式的选择

2.4 限幅部分的优化

```
for (r = 0; r < C; r++)
        for (n = 0; n < Nd / 8; n++)
        {
                resf = _mm256_mul_ps(*p_tabI, fact);
                resf = _mm256_max_ps(resf, vminf);
                resf = _mm256_min_ps(resf, vmaxf);
                resi = _mm256_cvttps_epi32(resf);
                p_tabI += 1;
                for (i = 0; i < 8; i++)
                        ptr_llr[32 * (8 * n + i) + r] = (int8_t)p_resi[i];
        }
```

2.5 判决部分的优化

```
uchar_itranspose_avx(h->llr_avx2, (__m256i*)decoded_bits, K);
```

3 VTune 测试结果分析

3.1 吞吐量计算

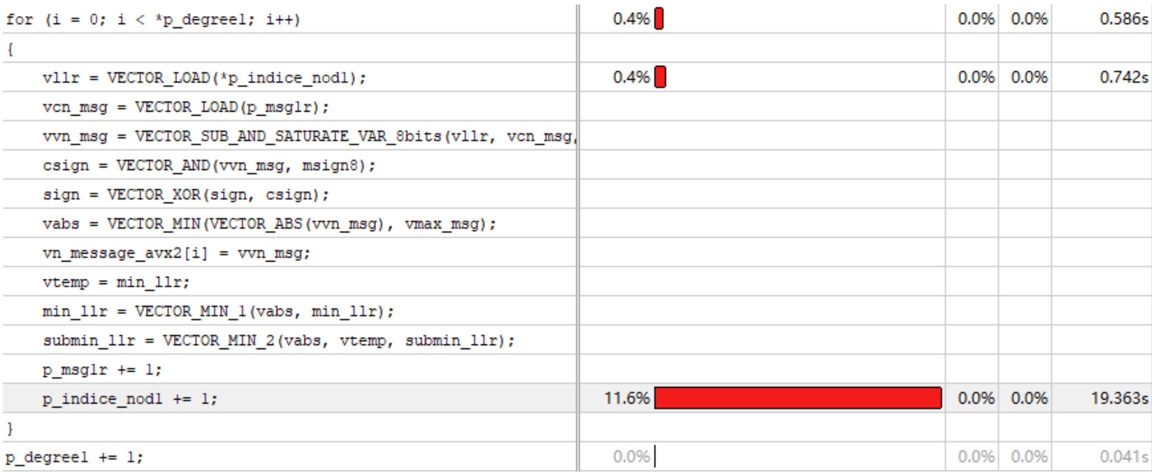
Function	Module	CPU Time
nr15_fec_ldpc_simd_decoder_avx2	test_simd_ldpc.exe	44.101s
main	test_simd_ldpc.exe	25.251s
vslsRngGaussian	mkl_vml_avx2.dll	20.966s
nr15_fec_ldpc_simd_rdm_dec_decbs	test_simd_ldpc.exe	16.834s
nr15_fec_ldpc_simd_cbs_enc_rm	test_simd_ldpc.exe	16.250s
[Others]		43.388s

图 4: Top Hotspots in VTune

$B = (8448 - 24) * 32 * 10^5 = 2.6957Gbit$
$t = 44.101s$
$Throughput = B/t = 2.6957Gbit/44.101s = 61.125Mbps$

3.2 主要耗时部分

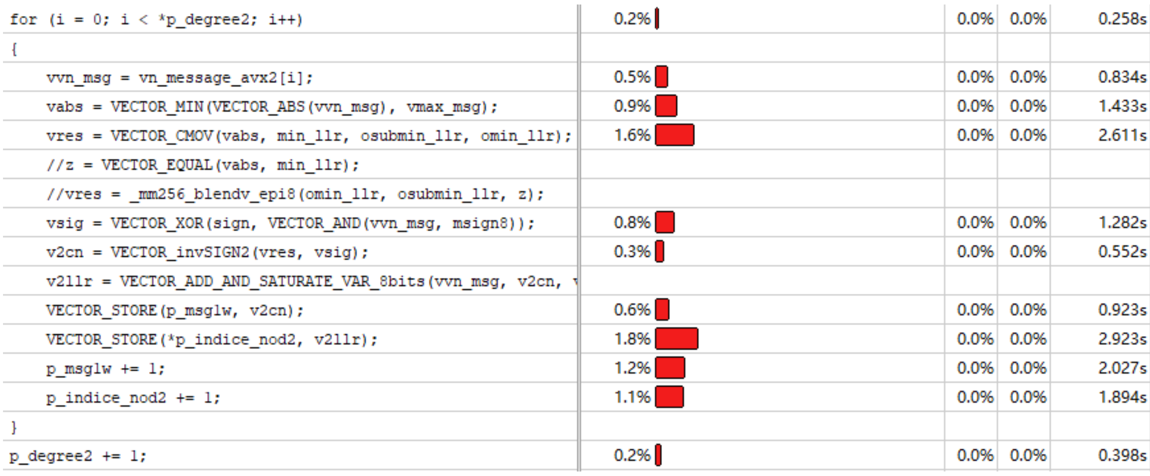
3.2.1 循环 1



for (i = 0; i < *p_degree1; i++)	0.4%	0.0%	0.0%	0.586s
{				
vllr = VECTOR_LOAD(*p_indice_nod1);	0.4%	0.0%	0.0%	0.742s
vcn_msg = VECTOR_LOAD(p_msglr);				
vvn_msg = VECTOR_SUB_AND_SATURATE_VAR_8bits(vllr, vcn_msg,				
csign = VECTOR_AND(vvn_msg, msign8);				
sign = VECTOR_XOR(sign, csign);				
vabs = VECTOR_MIN(VECTOR_ABS(vvn_msg), vmax_msg);				
vn_message_avx2[i] = vvn_msg;				
vtemp = min_llr;				
min_llr = VECTOR_MIN_1(vabs, min_llr);				
submin_llr = VECTOR_MIN_2(vabs, vtemp, submin_llr);				
p_msglr += 1;				
p_indice_nod1 += 1;	11.6%	0.0%	0.0%	19.363s
}				
p_degree1 += 1;	0.0%	0.0%	0.0%	0.041s

图 5: 循环 1 耗时情况

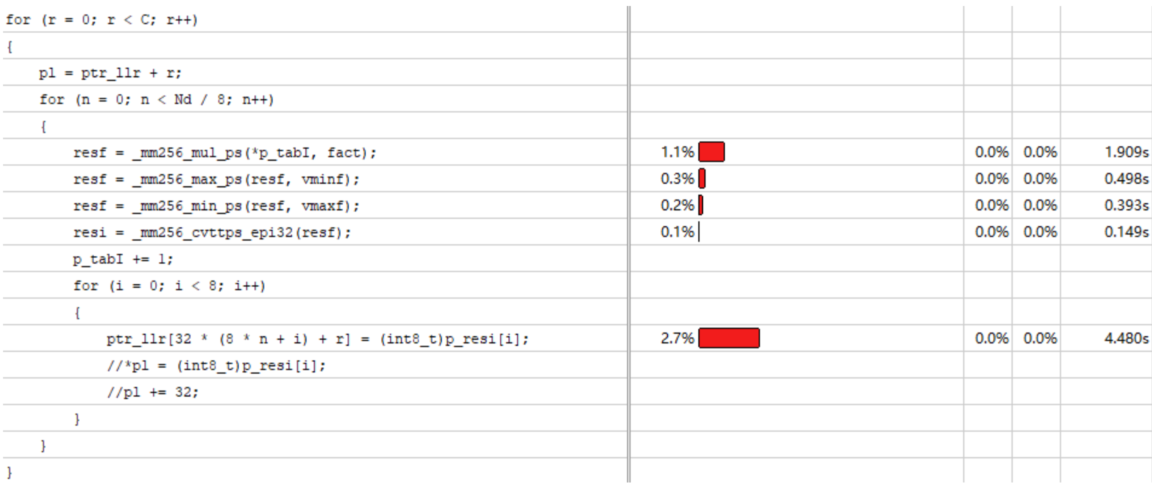
3.2.2 循环 2



for (i = 0; i < *p_degree2; i++)	0.2%	0.0%	0.0%	0.258s
{				
vvn_msg = vn_message_avx2[i];	0.5%	0.0%	0.0%	0.834s
vabs = VECTOR_MIN(VECTOR_ABS(vvn_msg), vmax_msg);	0.9%	0.0%	0.0%	1.433s
vres = VECTOR_CMOV(vabs, min_llr, osubmin_llr, omin_llr);	1.6%	0.0%	0.0%	2.611s
//z = VECTOR_EQUAL(vabs, min_llr);				
//vres = _mm256_blendv_epi8(omin_llr, osubmin_llr, z);				
vsig = VECTOR_XOR(sign, VECTOR_AND(vvn_msg, msign8));	0.8%	0.0%	0.0%	1.282s
v2cn = VECTOR_invSIGN2(vres, vsig);	0.3%	0.0%	0.0%	0.552s
v2llr = VECTOR_ADD_AND_SATURATE_VAR_8bits(vvn_msg, v2cn, v				
VECTOR_STORE(p_msglw, v2cn);	0.6%	0.0%	0.0%	0.923s
VECTOR_STORE(*p_indice_nod2, v2llr);	1.8%	0.0%	0.0%	2.923s
p_msglw += 1;	1.2%	0.0%	0.0%	2.027s
p_indice_nod2 += 1;	1.1%	0.0%	0.0%	1.894s
}				
p_degree2 += 1;	0.2%	0.0%	0.0%	0.398s

图 6: 循环 2 耗时情况

3.2.3 限幅部分



for (r = 0; r < C; r++)				
{				
pl = ptr_llr + r;				
for (n = 0; n < Nd / 8; n++)				
{				
resf = _mm256_mul_ps(*p_tabI, fact);	1.1%	0.0%	0.0%	1.909s
resf = _mm256_max_ps(resf, vminf);	0.3%	0.0%	0.0%	0.498s
resf = _mm256_min_ps(resf, vmaxf);	0.2%	0.0%	0.0%	0.393s
resi = _mm256_cvttps_epi32(resf);	0.1%	0.0%	0.0%	0.149s
p_tabI += 1;				
for (i = 0; i < 8; i++)				
{				
ptr_llr[32 * (8 * n + i) + r] = (int8_t)p_resi[i];	2.7%	0.0%	0.0%	4.480s
//*pl = (int8_t)p_resi[i];				
//pl += 32;				
}				
}				
}				

图 7: 限幅部分耗时情况