组会报告

徐益

2018 年 9 月 3 日

1 工作内容

1. 使用 avx2 指令实现限幅部分；
2. 完成代码基于 linux 平台的调试；
3. 在服务器上对译码模块进行性能测试。

2 使用 avx2 指令实现限幅部分

2.1 使用 packs 相关指令

原模块：

```
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];
        }
```

现模块：

```
for (n = 0; n < Nd; n++)
{
        for (i = 0; i < 4; i++)
        {
                vllrf = _mm256_load_ps((float *)p_tabI);
                resf = _mm256_mul_ps(vllrf, fact);
                resf = _mm256_max_ps(resf, vminf);
                resf = _mm256_min_ps(resf, vmaxf);
                resi[i] = _mm256_cvttps_epi32(resf);
                p_tabI += 1;
        }
        vtemp16[0] = _mm256_packs_epi32(resi[0], resi[1]);
        vtemp16[1] = _mm256_packs_epi32(resi[2], resi[3]);
        vtemp8 = _mm256_packs_epi16(vtemp16[0], vtemp16[1]);
```

```
15          _mm256_store_si256(p_tab0, vtemp8);
16          p_tab0++;
17  }
18  uchar_transpose_avx(tab0, h->llr_avx2, Nd);
```

2.2 遇到的问题

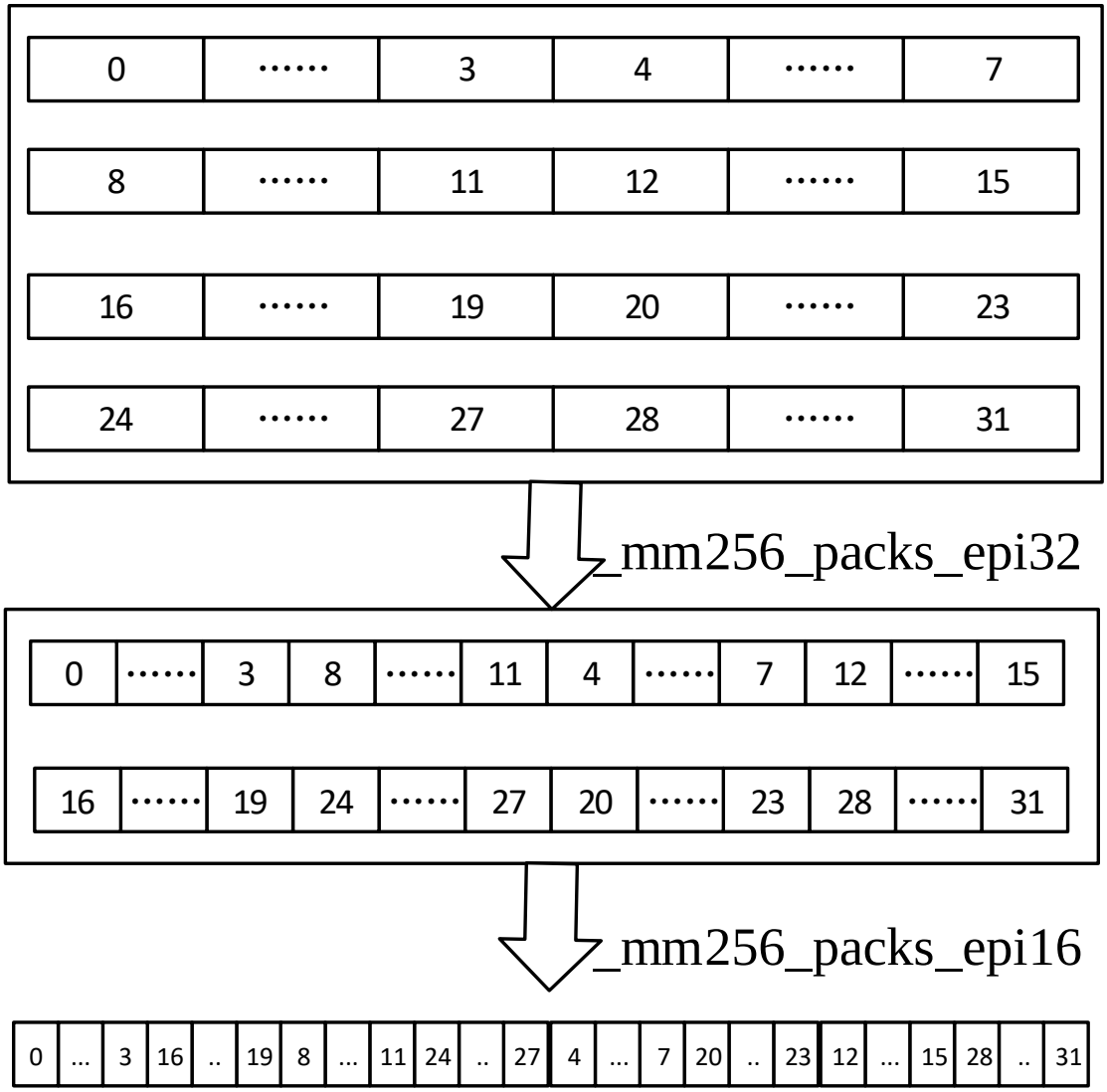

图 1: packs 相关指令的过程

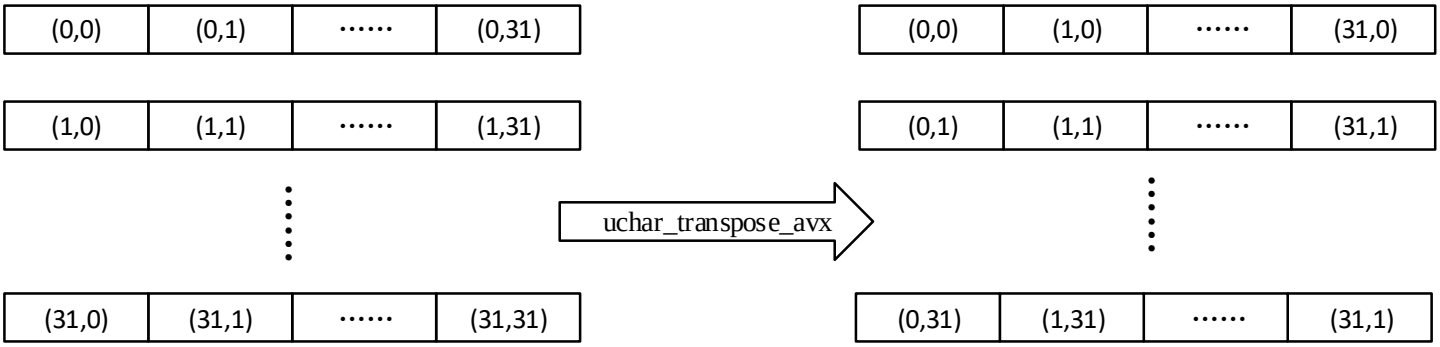

图 2: uchar_transpose_avx 函数的过程

3 代码基于 linux 平台的调试

3.1 遇到的问题

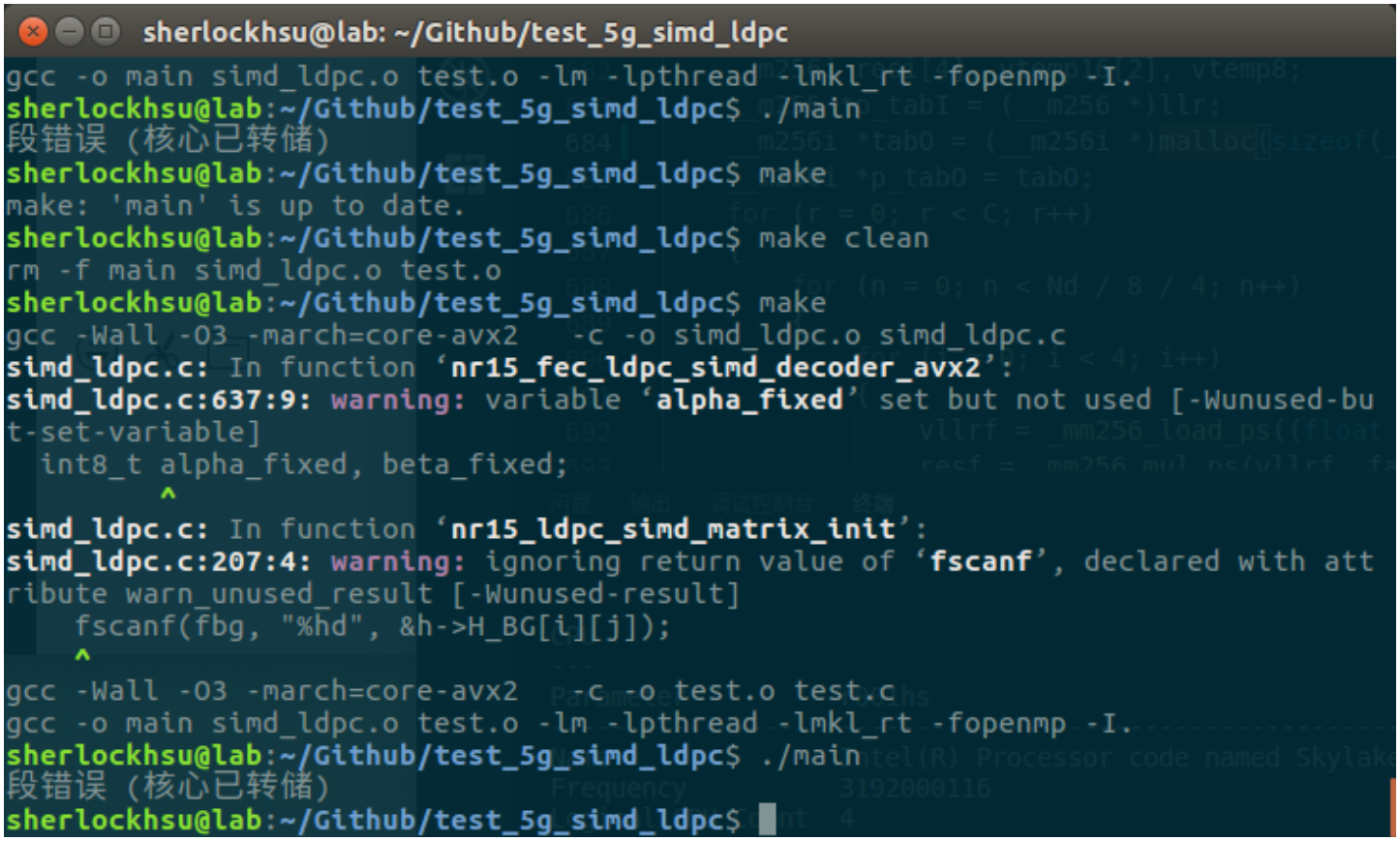

图 3: 段错误

错误原因:
使用 malloc 函数分配空间时，未对齐寄存器变量地址。
解决方法:
使用 _mm_malloc 函数分配寄存器相关地址空间;
使用 _mm_free 释放相关地址空间。

3.2 编写 makefile

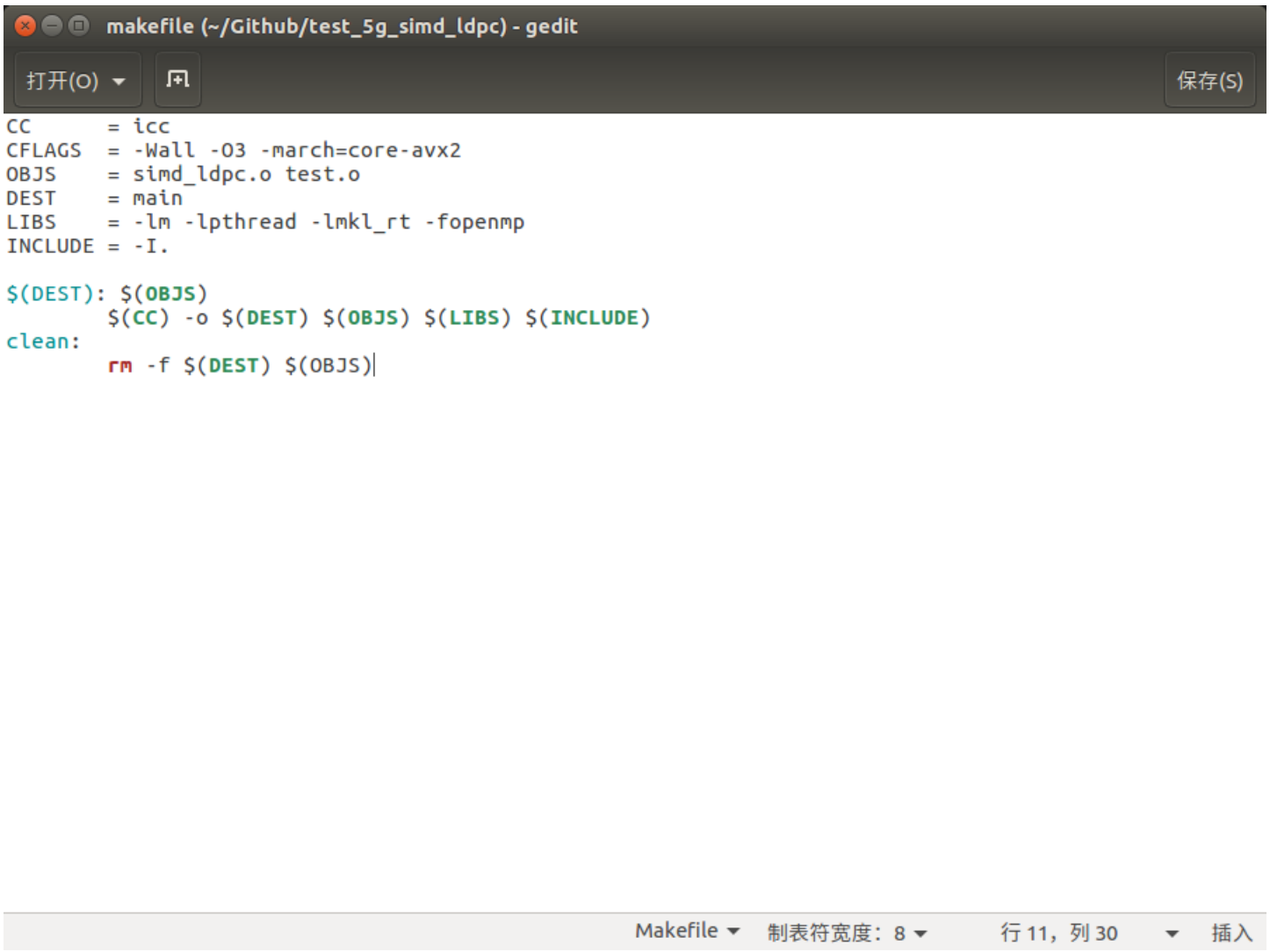

图 4: makefile

4 Linux 平台上的性能测试

4.1 Linux 平台上的 VTune 测试方法

1. source /opt/intel/vtune_amplifier/amplxe-vars.sh
2. amplxe-cl -collect hotspots ./main
3. amplxe-cl -report hotspots r000hs

4.2 本地测试

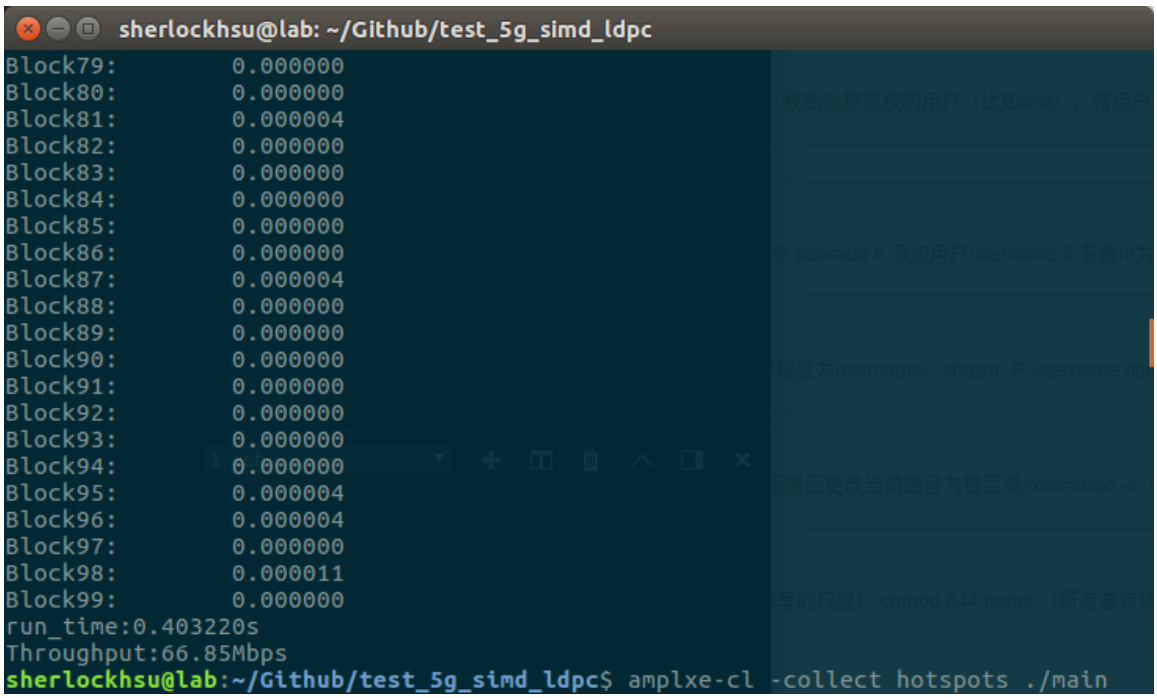

图 5: 本地运行结果

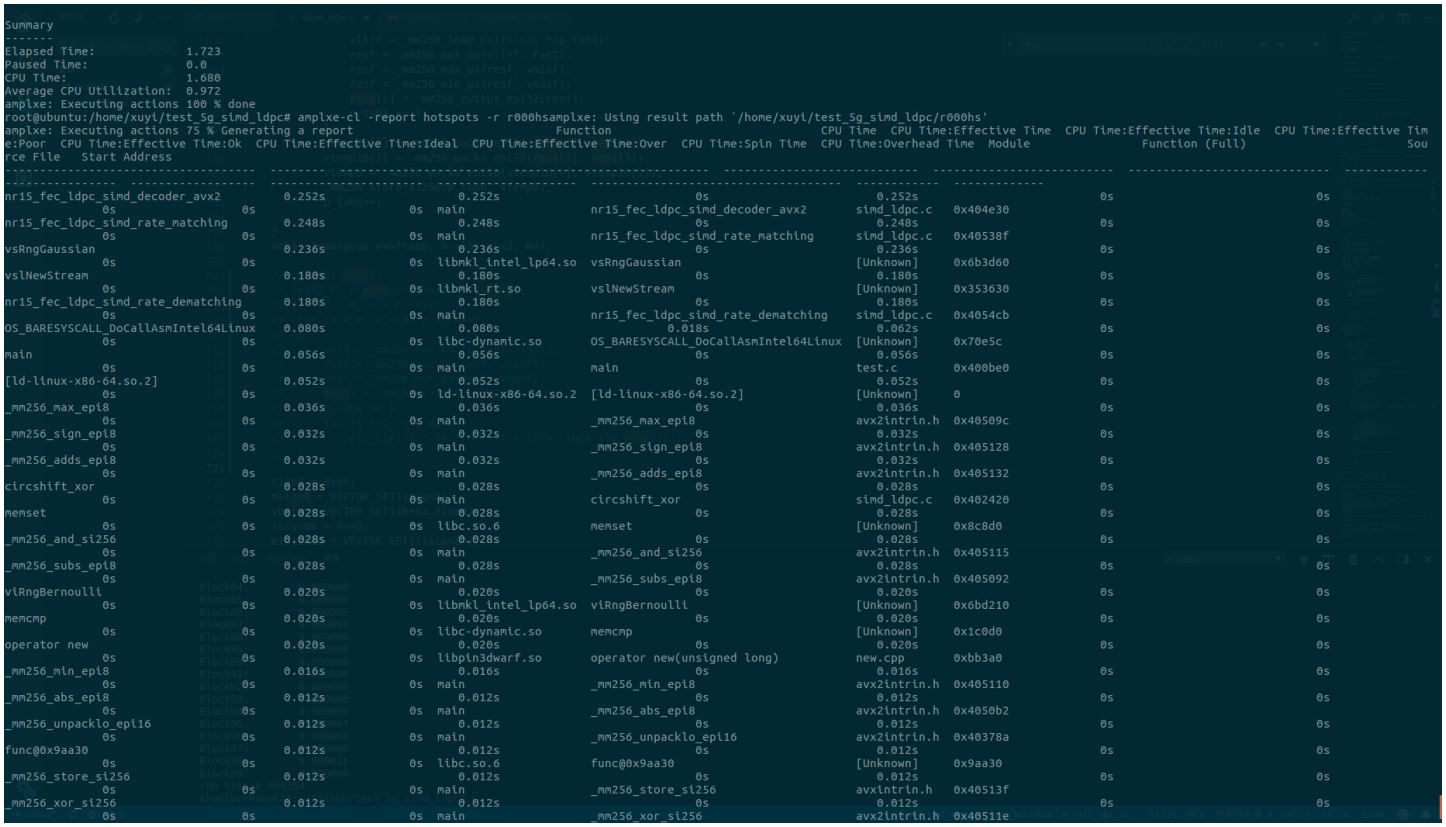

图 6: 服务器 VTune 测试结果（错误）

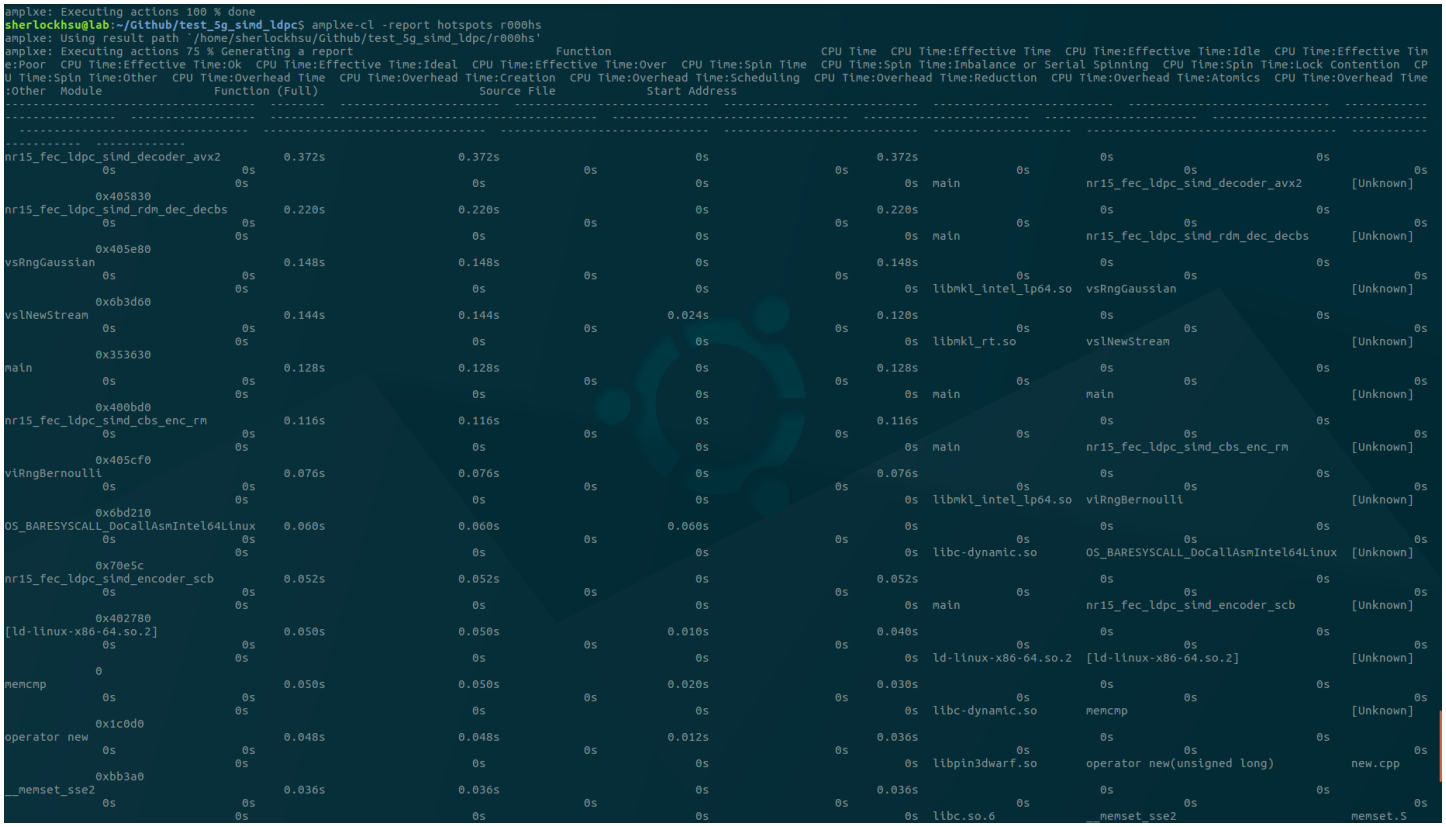

图 7: 本地 VTune 测试结果

4.3 服务器测试

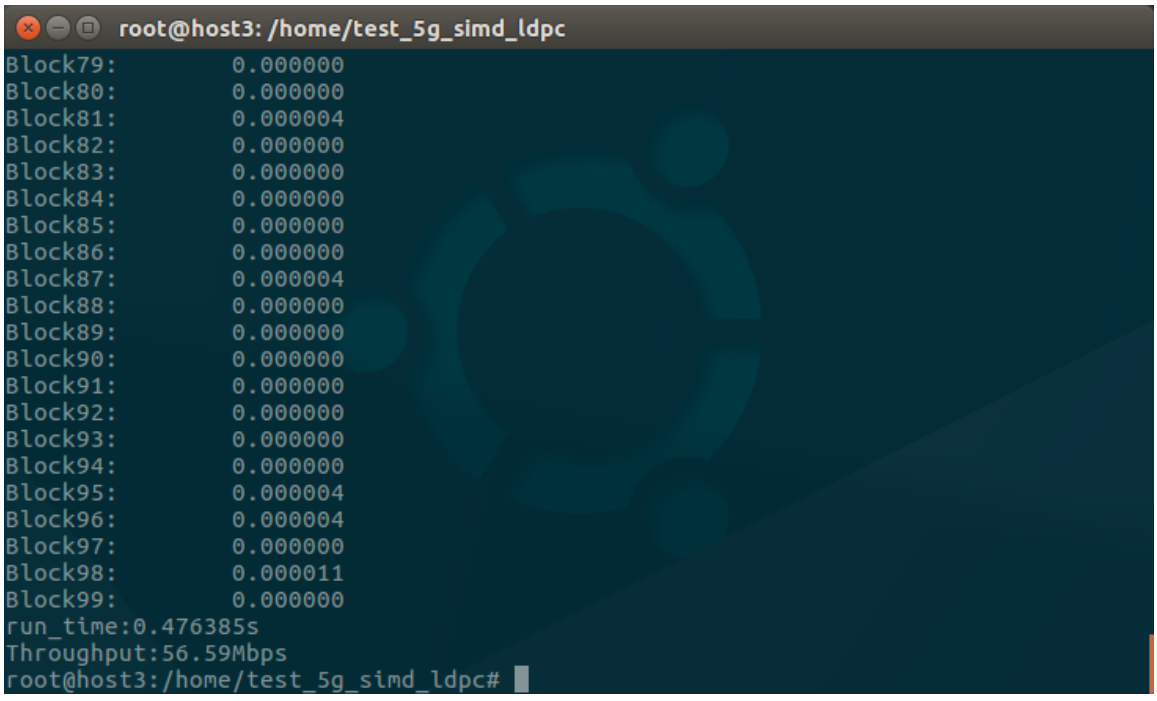

图 8: 服务器运行结果

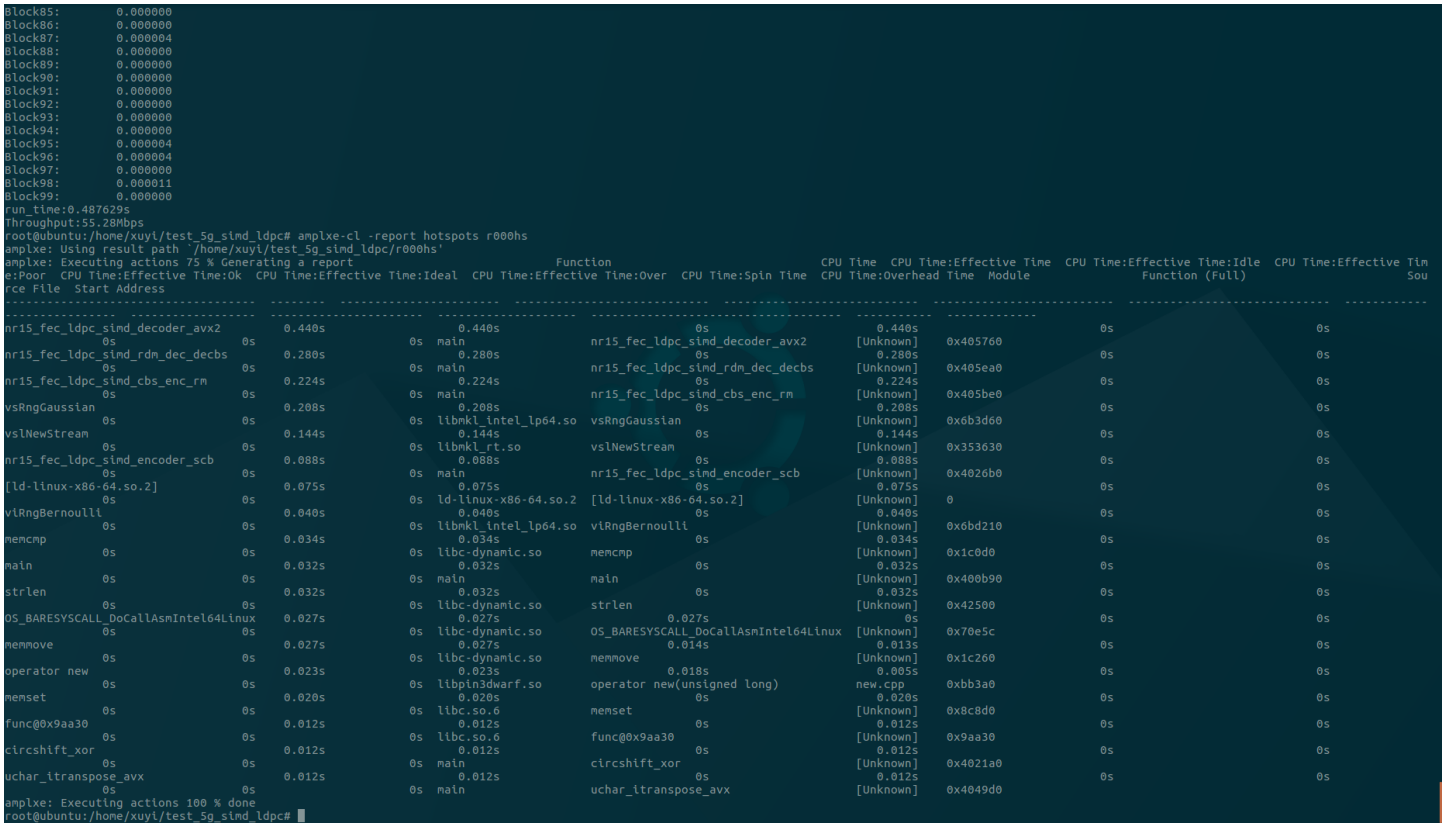

图 9: 服务器 VTune 测试结果

5 仍存在的问题

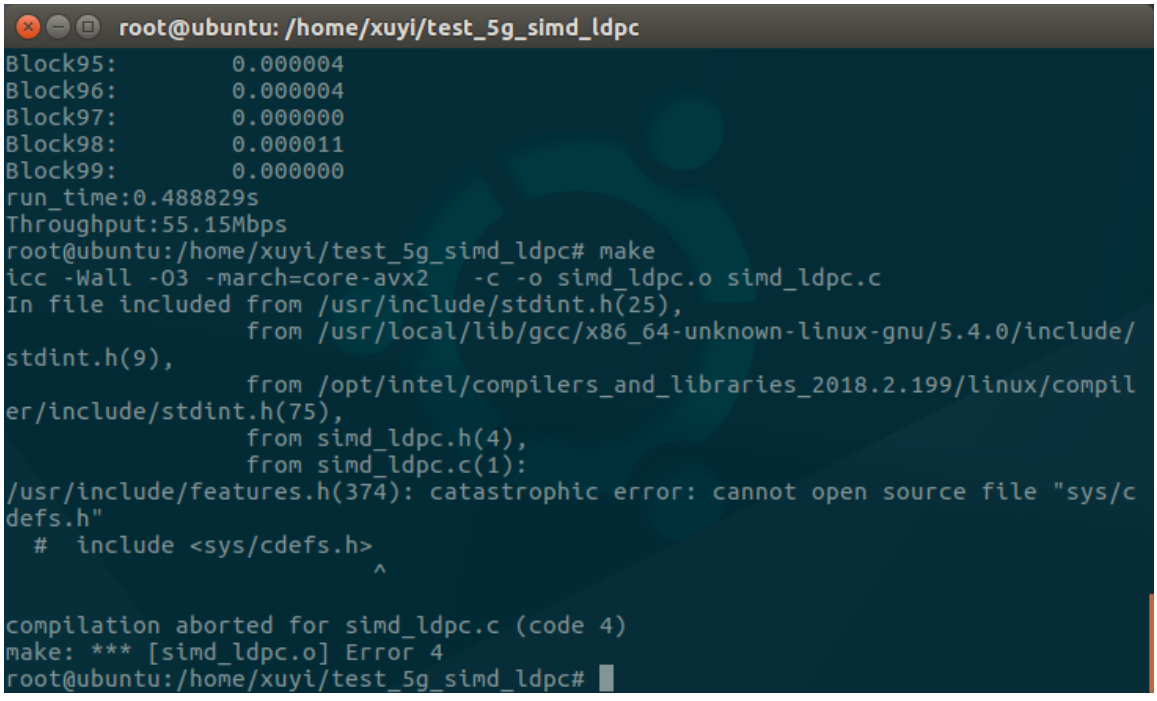

图 10: 更新 gcc 的服务器上遇到的 icc 编译错误问题