FFT算法的实现与优化

贾海鹏，张云泉，李琨，李志豪

中国科学院计算技术研究所计算机体系结构国家重点实验室

快速傅里叶变换 (FFT, Fast Fourier Transform)是用于计算离散傅里叶变换的快速算法，是SKA数据处理的关键性算法之一。然而，FFT存在实现优化复杂、radix种类繁多等难题。为此，本文提出了一种基于计算模式的FFT代码自动生成方法，并通过蝶形网络优化、蝶形计算优化、蝶形自动生成、SIMD汇编优化、内存对齐、Cache-aware的分块算法和高效转置等优化方法的应用，显著提升了FFT算法的性能。然后，本文介绍了一种在GPU平台上的自适应FFT框架原型，并基于此，描述了一种在CPU+GPU集群上利用2D网格分解来进行3D FFT的算法。最后给出了详细的性能分析和评估。

1. FFT算法的优化与实现

FFT的蝶形网络由stage-section-butterfly三层组成，不同的FFT序列将通过不同的蝶形进行组合，实现FFT变换。不同的基（如radix-2，3，4，5，7，11，13，…）具有不同的蝶形，通过FFT plan的生成以及蝶形kernel的底层优化，能有效地提升FFT的执行效率。

(1)蝶形网络优化。为了更好的把不同的基揉合进同一个高性能蝶形网络框架，本文设计了输入和输入都为自然序列的FFT蝶形网络，如图1所示。该网络具有两个优势：一是完美支持混合基，完美支持对FFT输入数据分解的自适应调优；二是更加有利于SIMD优化。

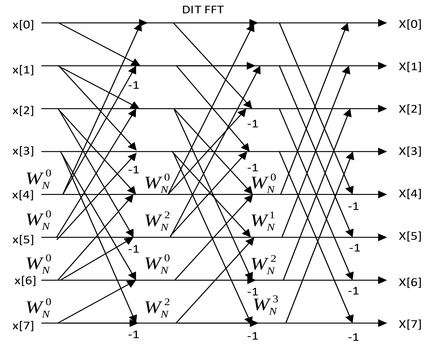


图1 对混合基和SIMD友好的蝶形网络

(2) 蝶形计算优化。FFT的每一种基，均对应着一种蝶形计算序列，需要对每种蝶形进行特定实现和优化。由于质数的个数无穷大，若为每一种质数基都编写蝶形代码，工作量极大，不可能完全手工实现。为了解决这个问题，我们实现了一种基于模板的FFT蝶形网络计算序列的生成系统。通过专家优化经验和FFT蝶形计算元操作的抽取，构建由原子计算模板-混合计算模板-蝶形三个级别组装而成的蝶形计算方法。

```
For(int i=0;i<stage_num;++i) // stages in FFT network (FFT 蝶形网络)
  For(int i=0;i<section_num;++i) // sections in each stage
    For(int i=0;i<butterfly_num;++i) // butterflies in each section
      FFT_R7_KERNEL(...) // 蝶形(此处以radix-7蝶形为例)
        R7_MIX_ATOM_TEMPLATE(out[1], out[6], TMP, TW, in_0, sum) //基7的混合计算模板
        {
          // 共由m=[7/2]=4个原子计算模板构成混合计算模板
          PERF_FORMER_ODD_CPX_WITH_F(out[1], TMP[0], TMP[1], TW7_1R_F, TW7_1I, TMP[10], in_0, sum);
          PERF_FORMER_ODD_CPX_TW(out[1], TMP[2], TMP[3], TW7_2R_F, TW7_2I, TMP[10], TMP[11], sum);
          PERF_FORMER_ODD_CPX_TW(out[1], TMP[4], TMP[5], TW7_3R_F, TW7_3I, TMP[10], TMP[11], sum);
          PERF_ATOM_ODD_ADDSUB(out[6], sum, out[1]);
        }
        R7_MIX_ATOM_TEMPLATE(out[2], out[5], TMP, TW, in_0, sum); //基7的混合计算模板，同上，不再展开
        R7_MIX_ATOM_TEMPLATE(out[3], out[4], TMP, TW, in_0, sum); //基7的混合计算模板，同上，不再展开
```

共由k=[7/2]=3个混合计算模板组成基7的蝶形

图2 原子计算模板-混合计算模板-蝶形(kernels)-section-stage的完整构成。

2.基于MPI的FFT并行库

3D FFT集群算法主要基于1D 和 2D分解策略。假定输入数据的规模为$n_0 \times n_1 \times n_2$ ($n_0 \geq n_1 \geq n_2$)，计算次序依次为n_2，n_1, n_0。

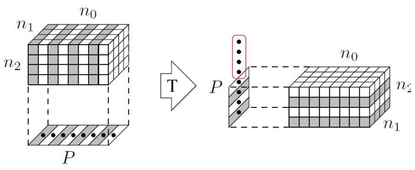
(1) 1D分解

a.计算策略

- 沿 n_0 维，将数据划分到$P(P \leq n)$ 个进程上。
- 其次，P 个进程并行进行n_0/p 批2D FFT(size=$n_1 \times n_2$) 的计算。
- 由于计算n_0 维所需的数据分布在不同的进程上，故需进行一次所有进程上的数据转置来完成n_0维的计算。

b.瓶颈

- 并行计算的规模受限于n_1或n_2 。



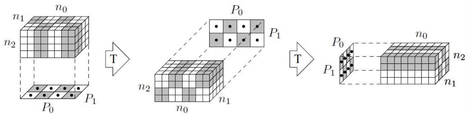
(a) 1D decomposition

(2) 2D 分解

a. 计算策略

- 如图(b)所示，首先沿n_1，n_0进行分解，并行计算n_2维度的数据。其次沿n_2，n_0进行分解，并行计算n_1维度的数据。最后进行计算n_0的数据。

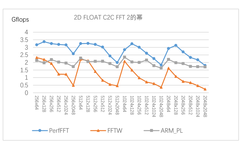
b.解决了1D 分解并行计算规模受限的瓶颈



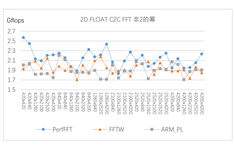
(b) 2D decomposition

3. 性能评估

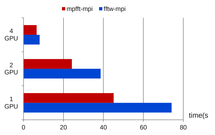
在包括ARM V8架构、天河2超算在内的不同计算平台上对FFT算法进行测试评估，对比结果如下：



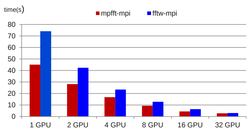
(a) FFT on ARM V8



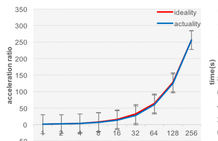
(b) FFT on ARM V8



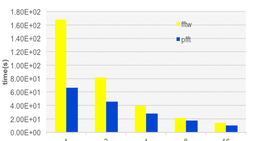
(c) QDR Infiniband



(d) Intel 82573 Gigabit Ethernet



(e) Tianhe2 (PFFT)



(f) QDR Infiniband

Shanghai, China , Dec. 23-24, 2017

2017 2nd SKA Annual Symposium on SDP&HPC