大规模集群上多维FFT算法的实现与优化

李琨

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

背景

- 介绍
  - 快速傅里叶变换 (FFT, Fast Fourier Transform)
    - 用于计算傅里叶变换的快速算法，应用广泛
    - SKA中数据处理的关键算法
    - 带宽密集型计算
    - 由于周期性带来的矩阵蝶形因子的计算特征，可通过将FFT算法中的矩阵逐级分解进行计算

背景

(a): $$y_k = \sum_{j=0}^{N-1} \omega_n^{jk} \cdot x_j \; (0 \leq k < n) \qquad \omega_n = e^{-\frac{2\pi i}{n}} (i = \sqrt{-1})$$

$$DFT_n = \begin{bmatrix} 1 & 1 & 1 & & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & & \omega_n^{2(n-1)} \\ & \vdots & & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix}, y = DFT_n \cdot x$$

(b): $$\begin{aligned} j &= (j_1, j_2) = j_1 \cdot r + j_2 \quad (0 \leq j_1 < q,\ 0 \leq j_2 < r) \\ k &= (k_2, k_1) = k_1 + k_2 \cdot q \;\; (0 \leq k_1 < q, 0 \leq k_2 < r) \\ n &= q \cdot r \end{aligned}$$

(c): $$y_k = \sum_{j_2=0}^{r-1} [(\sum_{j_1=0}^{q-1} \omega_q^{j_1 k_1} \cdot x_{j_1 \cdot r + j_2}) \cdot \omega_n^{j_2 k_1}] \cdot \omega_r^{j_2 k_2}$$

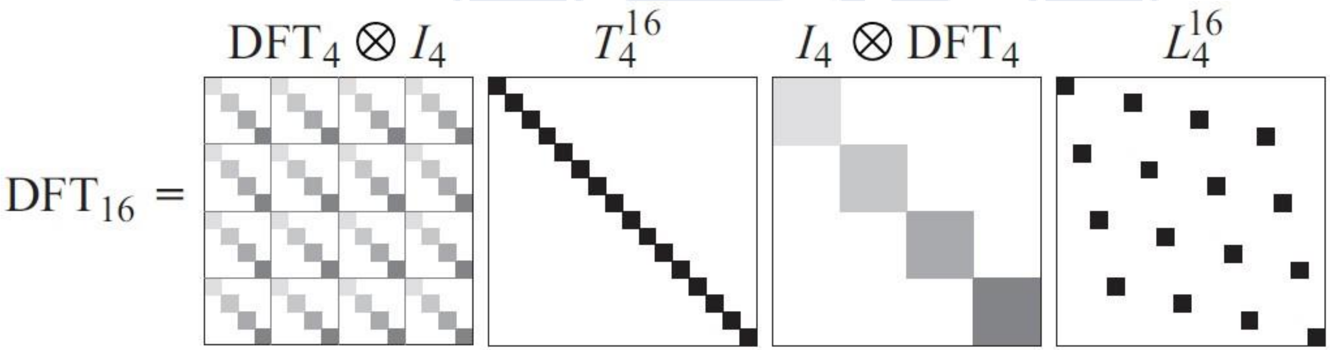
背景

(c): $$y_k = \sum_{j_2=0}^{r-1} [(\sum_{j_1=0}^{q-1} \omega_q^{j_1 k_1} \cdot x_{j_1 \cdot r + j_2}) \cdot \omega_n^{j_2 k_1}] \cdot \omega_r^{j_2 k_2}$$

$$A \otimes B = [a_{k,\ell}B], \text{ for } A = [a_{k,\ell}].$$

$$A \otimes B = \begin{pmatrix} a_{0,0}B & \cdots & a_{0,m-1}B \\ \vdots & \ddots & \vdots \\ a_{m-1,0}B & \cdots & a_{m-1,m-1}B \end{pmatrix}.$$

(d): $$DFT_n = (DFT_r \otimes I_q) T_q^n (I_r \otimes DFT_q) L_r^n$$

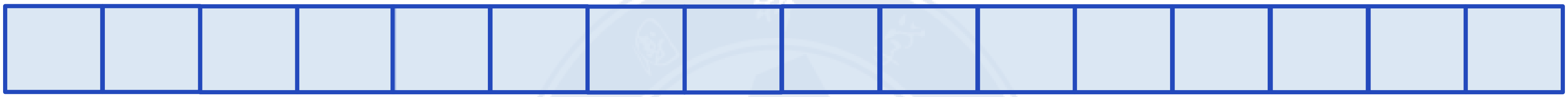


(a) Matrix factorization

背景

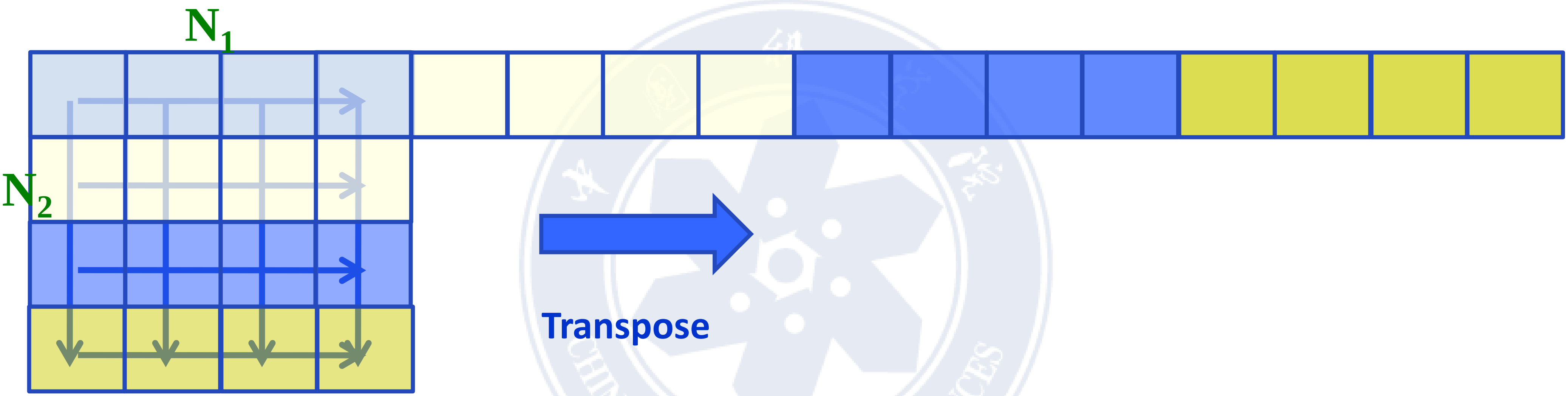
- 分解过程
  - FFT(Fast Fourier Transform)

样例：$N=N_1 \times N_2 = 4 \times 4$



背景

- 分解过程
  - FFT(Fast Fourier Transform)



FFT along rows / FFT along columns

异构集群FFT算法

- **异构计算系统已逐渐成为高性能计算领域的热点研究方向.**
  - 天河-IA
  - 神威太湖之光
  - 曙光星云
- **SDP多样的实验平台**
- Case study: 异构集群上的3D FFT
  - 应用:天体物理N-body模拟, 医学血流量模拟等.
  - 并行算法: 基于1D 、2D 分解

3D FFT集群算法主要基于1D 和 2D分解策略

- $n_0 \times n_1 \times n_2$ ($n_0 \geq n_1 \geq n_2$), 计算次序： n_2 , n_1, n_0

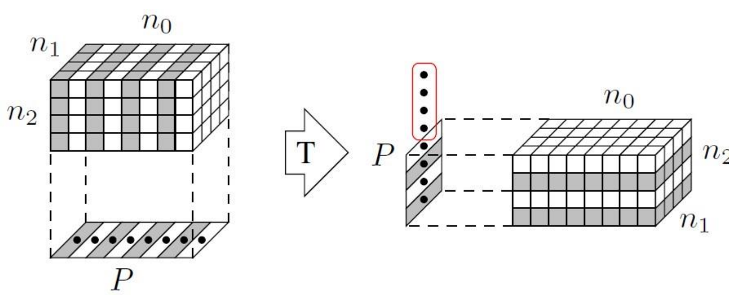
1D 分解

- 计算策略

 1. 沿 n_0 维, 将数据划分到P($P \leq n$) 个进程上.
 2. 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.
 3. 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。

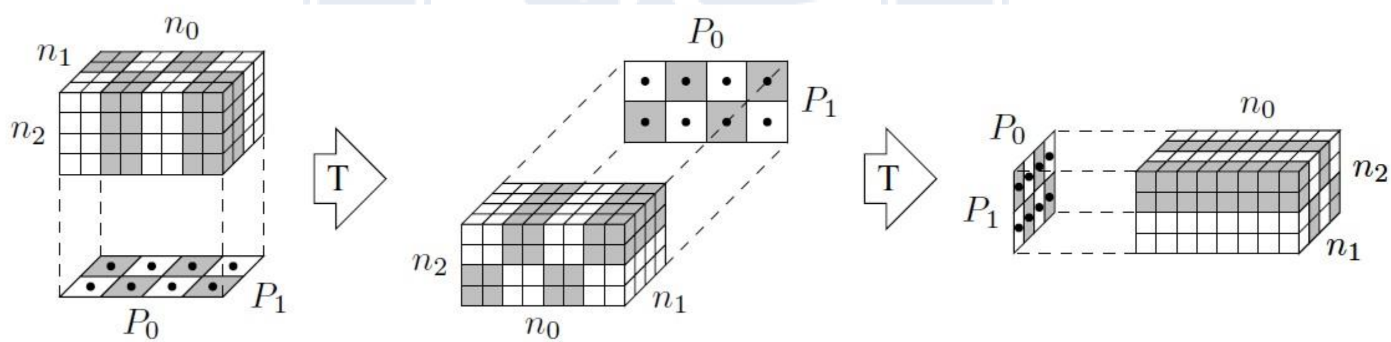
- 瓶颈

 并行计算的规模受限于n_1或 n_2



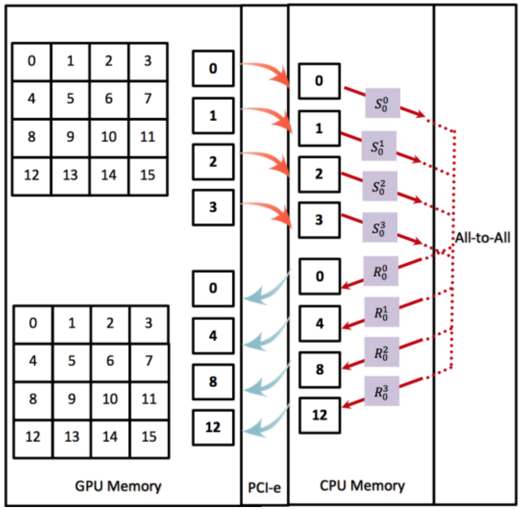
算法策略

- **3D FFT集群算法主要基于1D 和 2D分解策略**
  - $n_0 \times n_1 \times n_2$ $(n_0 \geq n_1 \geq n_2)$
  - **2D 分解**
  - 支持的最大进程数：$n_0 \times n_1$



通信优化

- 瓶颈: 通信时间成为分布式FFT的主要开销.
- 解决：将memcpy划分成每个进程可独立发送的数据块.
- 进程数量：4
- S_i^j：从进程i至进程j的数据发送指令.
- R_i^j：进程j接收来自进程i的数据接收指令.



FFT 算法评估

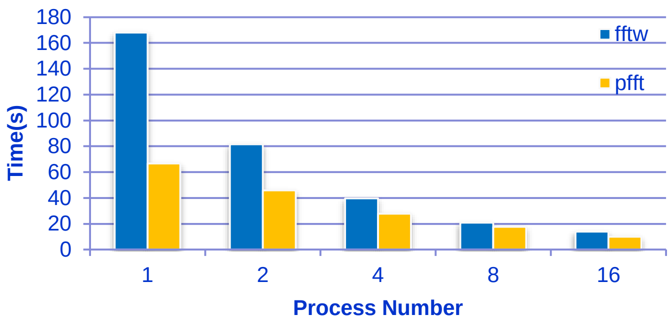
- **FFTW: 用于计算一维或多维、任意输入规模、实数/复数离散傅里叶变换的基本算法库.**
  - 分解策略: 1D 分解
- **PFFT: 基于MPI并行分布式存储架构上的FFT并行算法库.**
  - 分解策略: 2D 分解
- **MPFFT: 异构（CPU+GPU）平台上FFT专用算法库.**
  - 分解策略: 2D 分解

实验结果

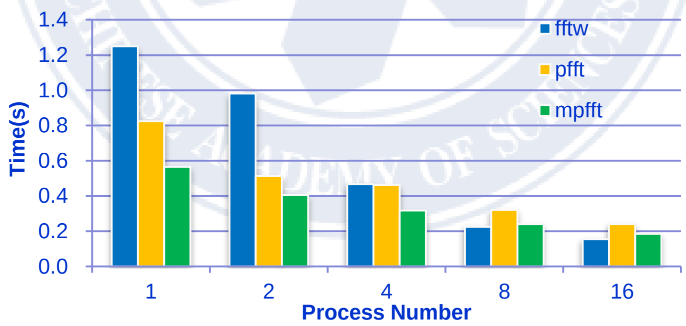
实验平台: 单节点

CPU	Intel Xeon E5-2670 v3	Cores per package	12	GCC version	4.8.4
CPU GHz	2.8	Threads per core	2	Internetwork	Infiniband
Packages	2	RAM	64G	MPI version	mvapich2-2.2

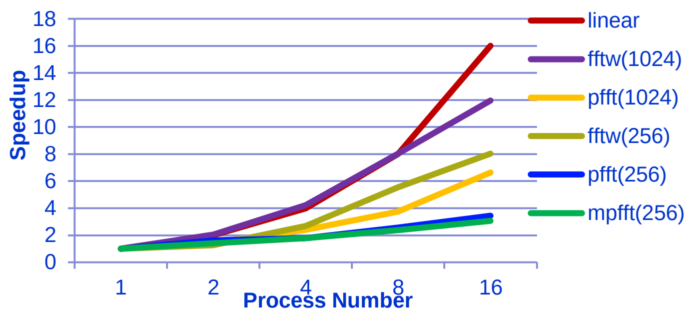
实验结果



A: Input Size: 1024×1024×1024



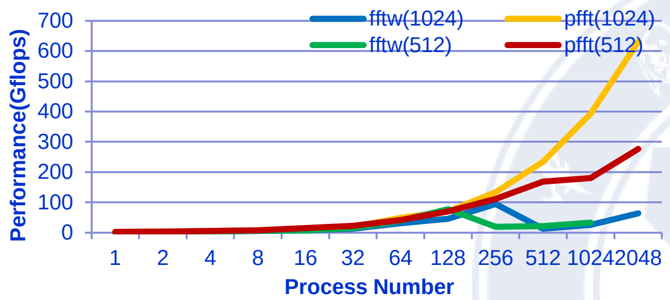
B: Input Size: 256×256×256



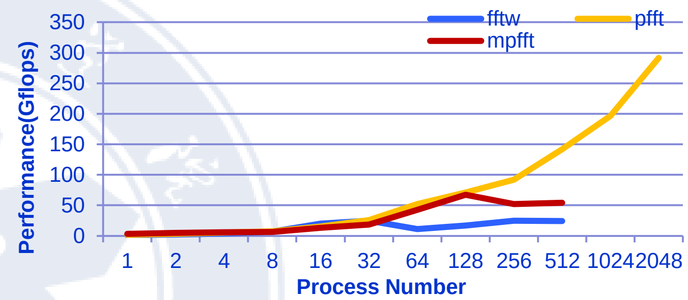
C: Speedup Comparison

实验结果

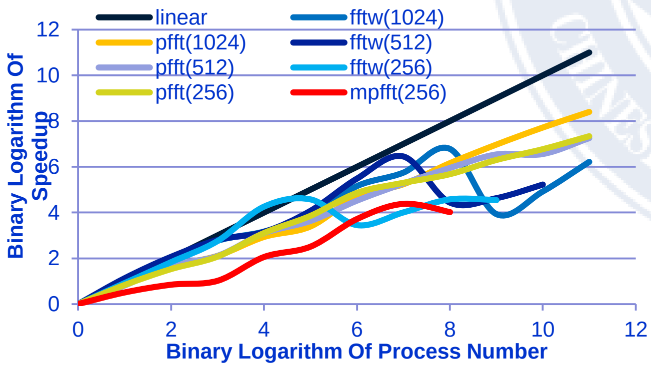
- 实验平台: 天河-1A
  - 在异构 (CPU+GPU) 大规模集群上对FFT算法进行评估.
- 实验结果



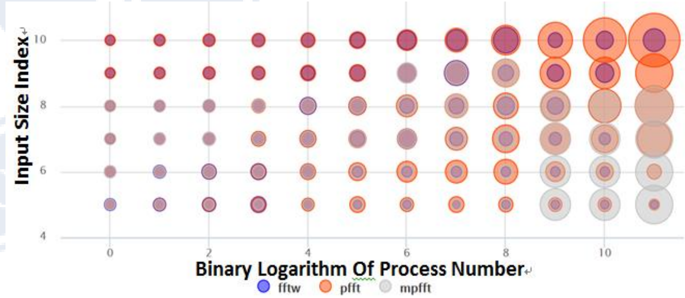
A: Input Size: 1024×1024×1024; 512×512×512



B: Input Size: 256×256×256



C: Different Input Size Comparison

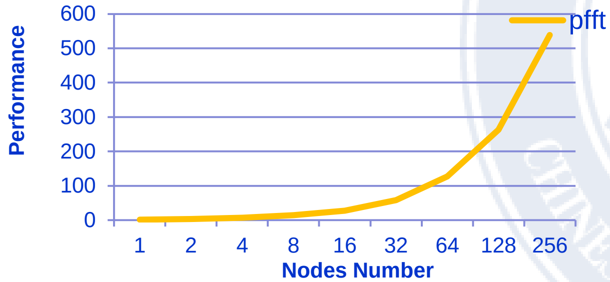


D: Algorithm Performance Evaluation

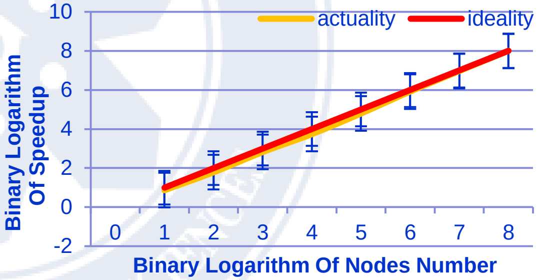
实验结果

- 实验平台: 天河-2
  - 2017年11月超级计算机排行榜第二.
  - 在多节点上对FFT算法进行评估.

- 实验结果



A: InputSize: 1024×1024×1024



B: Speedup Comparison

总结

谢谢!