学习报告

报告人：黄一智
指导老师：李仁发教授、刘彦老师
2017-04-07

概要大纲

- 第四届太湖论坛学习总结

- HPCG学习阶段总结

第四届太湖论坛侧重——超算的应用

- 计算密集型应用（经典超算问题）
- 通信、仿存密集型应用
- 特殊应用

计算密集型应用

- 应用举例：稠密方程组（Linpack）、结构网络、谱方法（FFT）、MapReduce、组合逻辑、N-body、动态规划
- 计算特点：并行性好、计算量大、计算仿存比高
- 仿存特点：连续仿存
- 通信特点：通信量少（甚至无通信）、通信模式固定、通信量不会随问题规模增长变化太多
- 总结：容易将之前的10P~100P级应用扩展到E级应用。
- 体系结构需求：需要多态、多尺度、多粒度并行、多层次异构、多层次存储、以及更快的网络（当前传统的超级计算机、集群）

通信、仿存密集型应用

- 应用举例：稀疏线性方程（HPCG）、非结构网络（高超声速飞行数值模拟）、图相关算法
- 计算特点：程序不规则、计算通信比低
- 仿存特点：离散仿存或完全随机仿存
- 通信特点：动态、不规则通信，随问题规模增长，通信量增加大、数据交互复杂
- 总结：应用不容易直接通过并行获取更快加速
- 体系结构需求：存储上支持离散仿存（SSD、3D Xpoint SSD（Intel Optane Memory））、支持细粒度并行机制、创新通信方式（如：Mellanox的以数据为中心的网络内计算）。

特殊应用

- 应用举例：机器学习（之前两大类应用均有在神威太湖之光上做的很好，机器学习例外）

- 计算特点：计算量大，但模型本身的可扩展性受限（SWDnn当前仅仅做到了单机，无法做整机）
- 通信特点：学习数据量大
- 总结：当前超算的体系结构可能很难支持这类特殊应用，需要体系结构变革，甚至需要不同体系结构支持不同应用分类。

神威太湖之光超级计算机

- 系统结构：计算节点、计算节点板、超节点、机柜、计算系统的5层级集成方案
- 系统生态：国产自主CPU、自主编译链、操作系统、并行软件开发、调试、应用软件生态
- 维护、开发团队：清华大学老师和学生（移植了大量库、应用程序）

HPCG学习阶段总结——什么是HPCG

- 一个超算的Benchmark
- HPCG（High Performance Conjugate Gradient）相较Linpack所测的峰值性能更注重测试超算的通信与仿存，因此更为真实。

November 2016 HPCG Results

November 2016 HPCG Results

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak
1	RIKEN Advanced Institute for Computational Science Japan	**K computer** – SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10.510	7	0.6027	5.3%
2	NSCC / Guangzhou China	**Tianhe-2 (MilkyWay-2)** – TH-IVB-FEP Cluster, Intel Xeon 12C 2.2GHz, TH Express 2, Intel Xeon Phi 31S1P 57-core NUDT	3,120,000	33.863	2	0.5801	1.1%
3	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** – PRIMERGY CX600 M1, Intel Xeon Phi Processor 7250 68C 1.4GHz, Intel Omni-Path Architecture Fujitsu	557,056	13.555	6	0.3855	1.5%
4	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** – Sunway MPP, SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93.015	1	0.3712	0.3%
5	DOE/SC/LBNL/NERSC USA	**Cori** – XC40, Intel Xeon Phi 7250 68C 1.4GHz, Cray Aries Cray	632,400	13.832	5	0.3554	1.3%

- HPCG求解的问题模型：狄利克雷边界条件的单自由度三维热扩散模型。形如：

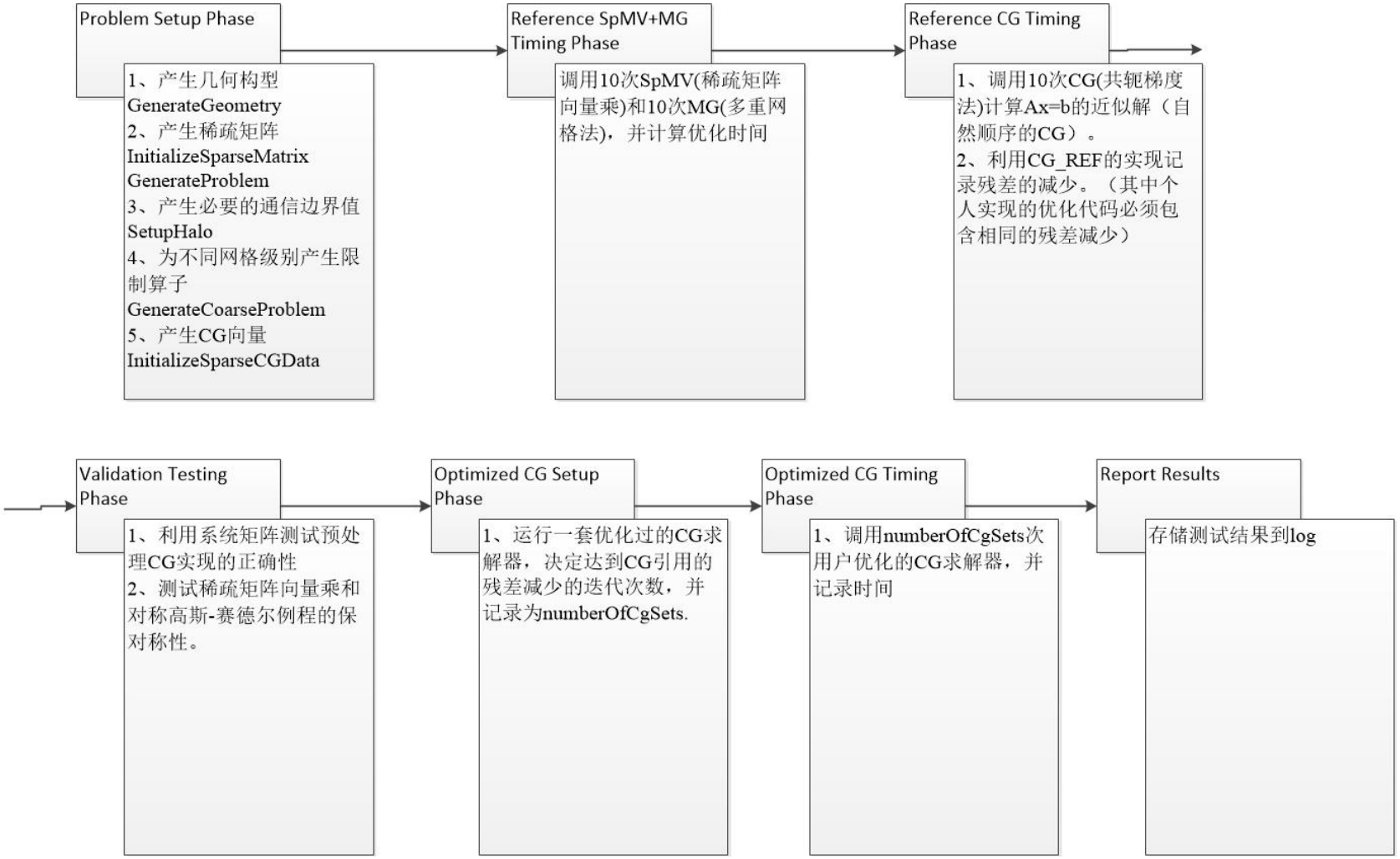
$$-\Delta\mu = f(x,y,z)\ x \in D$$

$$\mu(x,y,z) = g(x,y,z)\ x \in \partial D$$

$$\text{其中}\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

- HPCG将问题模型利用类似有限元、有限差分或有限体积的数学方法离散化到一个半规则的网格上产生一个合成稀疏线性系统，形如：$Ax = b$，
然后对该线性系统求解(benchmark会固定迭代次数，不一定需要求得收敛解)。

HPCG学习阶段总结——HPCG代码工作流程



HPCG学习阶段总结——关键算法

- 算法1：求解$Ax = b$的预处理CG

Algorithm 1 Preconditioned CG for $Ax = b$

Input: $A, b, x_0, it_{max}, \varepsilon$

1: $r_0 \leftarrow b - Ax_0$
2: **for** $i = 0, 1, ..., it_{max}$ **do**
3: $z_i \leftarrow M^{-1} r_i$
4: $s_i \leftarrow (r_i, z_i)$
5: **if** $(i = 0)$ $p_i \leftarrow z_i$
6: **else** $p_i \leftarrow z_i + (s_i / s_{i-1}) p_{i-1}$
7: $\alpha_i \leftarrow s_i / (p_i, Ap_i)$
8: $x_{i+1} \leftarrow x_i + \alpha_i p_i$
9: $r_{i+1} \leftarrow r_i - \alpha_i Ap_i$
10: **if** $(||r_{i+1}||_2 / ||r_0||_2 \leq \varepsilon)$ break;
11: **end for**

Output: x_{i+1}

HPCG学习阶段总结——关键算法

- 主要代价

1、第3行的多重网格预处理器MG

2、第7行的稀疏矩阵向量乘SpMV

- 算法2：V循环多重网格

Algorithm 2 V-cycle for $x^h = \mathrm{MG}(r^h)$

Input: r^h

1: **if** on the coarse level **then**
2: $x^h = \mathrm{SymGS}(0, r^h)$ ▷ Coarse-level solver
3: **else**
4: $x^h = \mathrm{SymGS}(0, r^h)$ ▷ Pre-smoothing
5: $r^{2h} = I_h^{2h}(r^h - A^h x^h)$ ▷ Restriction
6: $x^{2h} = \mathrm{MG}(r^{2h})$ ▷ Recursion
7: $x^h = x^h + I_{2h}^h x^{2h}$ ▷ Prolongation
8: $x^h = \mathrm{SymGS}(x^h, r^h)$ ▷ Post-smoothing
9: **end if**

Output: x^h

- 算法3：对称高斯-赛德尔迭代SymGS

Algorithm 4. Simplified SELLPACK SymGS native implementation

Input: A (nnz, $cols$, $nrows$, nb, $nnzs\_in\_row$), x, r

```
1: for i = 0, ..., nrows do
2:     sum:=r[i]
3:     block_id := i/nb
4:     id_in_block := i%nb
5:     for k = 0, ..., nnz_in_row do
6:         curCol := cols[block_id × (nb × nnzs_in_row) + id_in_block + k × nb]
7:         sum[j] -= nnz[block_id×(nb×nnzs_in_row)+id_in_block+k×nb]×x[curCol]
8:     end for
9:     sum += x[i] × diagonal[i]
10:    x[i] := sum/diagonal[i]
11: end for
12: for i = nrows − 1, ..., 0 do
13:    sum := r[i]
14:    block_id := i/nb
15:    id_in_block := i%nb
16:    for k = 0, ..., nnz_in_row do
17:        curCol := cols[block_id × (nb × nnzs_in_row) + id_in_block + k × nb]
18:        sum[j] -= nnz[block_id×(nb×nnzs_in_row)+id_in_block+k×nb]×x[curCol]
19:    end for
20:    sum += x[i] × diagonal[i]
21:    x[i] := sum/diagonal[i]
22: end for
```

Output: $SymGS(A, x, r)$

SymGS由一个forward-sweep和一个backward-sweep组成，其中：

forward-sweep： $x \leftarrow (L+D)^{-1}(r-Ux)$

backward-sweep： $x \leftarrow (U+D)^{-1}(r-Lx)$

U为A的上三角部分、D为A的对角部分、L为A的下三角部分。

- 算法存在问题：

1、很强的数据依赖性，不利于并行。

2、访问大规模稀疏矩阵和向量占用大量内存，同时， forward-sweep和backward-sweep的执行过程中，会重复访问大量内存单元，但是，由于顺序执行，破坏了时空的局部性原理，使得缓存命中下降。

后续工作

- 根据HPCG代码和多重网格法推导热扩散模型到稀疏线性方程组的过程
- 根据HPCG源码中SymGS算法存在的问题，以及杨超论文”623 Tflops HPCG Run on Tianhe-2 Leveraging Millions of Hybrid Cores”思想，初步改进HPCG中SymGS算法的实现。