天河高性能计算开发与环境建设

国家超级计算天津中心

2014年9月23日 于中科大

概要

- NSCC-TJ与天河一号简况
- 程序并行化设计
- 程序并行化实现
- 程序并行优化
- 应用案例
- 未来合作

NSCC平台与基础

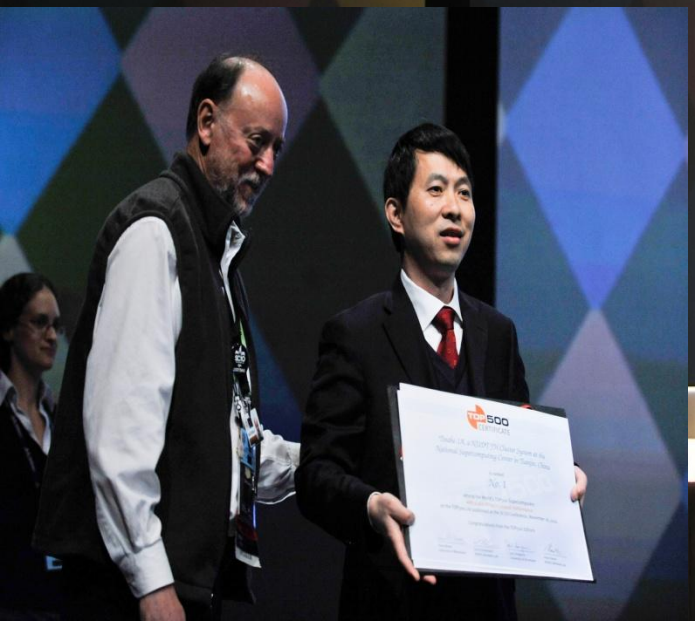
- 我国目前已投入运营的规模最大、计算能力最强的国家级超算中心
  - 峰值4700万亿次的天河一号超级计算机；
  - 400台以上服务器的通用云计算系统；
  - 容量超过10PB的海量存储系统；
  - 多领域行业软件；
  - 完善的网络基础设施；
  - 完善的机房、供电、制冷等基础设施。
- 主要业务为高性能计算、云计算和大数据服务，目前服务的政府、企业及科研院所用户数已达600余家

NSCC平台与基础

- 国家发改委“大数据技术与应用”国家地方联合工程实验室
- 国家工信部工业云试点单位
- 国家科技部示范型国际合作基地
- 博士后科研工作站
- 与多家企业和研究机构建立的联合实验室


天河
天河
天河
胡锦涛总书记为“天河一号”超级计算机研制成功题名
二〇〇九年十一月二十五日

“天河一号”超级计算机于2010年11月16日获得世界超级计算机500强排名第一



36th List: The TOP10

Rank	Site	Manufacturer	Computer	Country	Cores	Rmax [Tflops]	Power [MW]
1	National SuperComputer Center in Tianjin	NUDT	Tianhe-1A NUDT YH MPP, Xeon 6C, NVidia	China	186,368	2,566	4.04
2	Oak Ridge National Laboratory	Cray	Jaguar Cray XT5, HC 2.6 GHz	USA	224,162	1,759	6.95
3	National Supercomputing Centre in Shenzhen	Dawning	Nebulae TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU	China	120,640	1,271	2.58
4	GSIC, Tokyo Institute of Technology	NEC/HP	TSUBAME-2 HP ProLiant, Xeon 6C, NVidia, Linux/Windows	Japan	73,278	1,192	1.40
5	DOE/SC/ LBNL/NERSC	Cray	Hopper Cray XE6, 6C 2.1 GHz	USA	153,408	1.054	2.91
6	Commissariat a l'Energie Atomique (CEA)	Bull	Tera 100 Bull bullx super-node S6010/S6030	France	138.368	1,050	4.59
7	DOE/NNSA/LANL	IBM	Roadrunner BladeCenter QS22/LS21	USA	122,400	1,042	2.34
8	Univ of Tenn ee	Cray	Kraken Cray XT5 HC 2.36GHz	USA	98,928	831.7	3.09
	Fo rum	IB	Jugene e Gene/P Solution	Germany	294,912	825.5	2.26
			Cielo C 2.4 GHz	USA	107,152	816.6	2.95

三大技术创新

- CPU+GPU异构融合体系结构
- 64位多核多线程自主飞腾1000 CPU
- 自主高速互连通信技术

国际学术界的评价

中国的“天河一号”采取的CPU与GPU融合的结构，代表了未来超级计算机的发展趋势。随着计算机规模的不断拓展，**这种结构虽然不是唯一的解决方法，但目前看来是最好的。**

-- 美国斯坦福大学计算机系主任比尔•戴利

“天河一号”的运算速度比橡树岭国家实验室的要快大约40%，这是运算速率的极大提升。中国同时研制了一种互联技术，让这些处理器相互联系，**这不是美国的技术，而是中国自己的技术。这是一个创举。**

-- 美国田纳西大学教授杰克•唐加拉

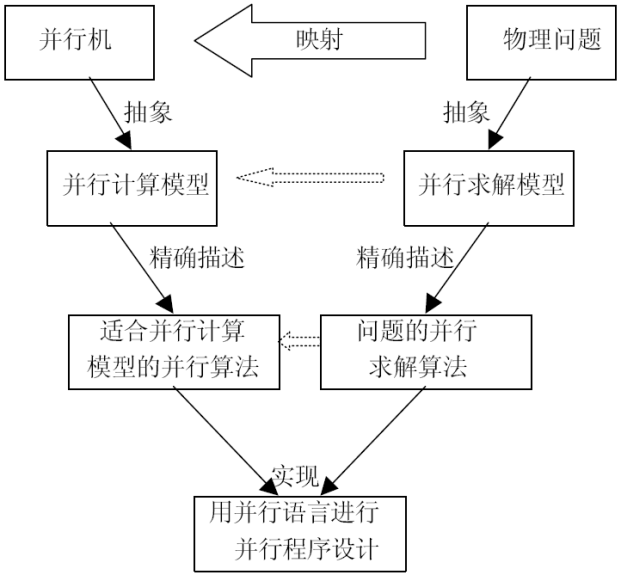
概要

- 程序并行化设计
- 程序并行化实现
- 程序并行优化
- 应用案例
- 未来合作

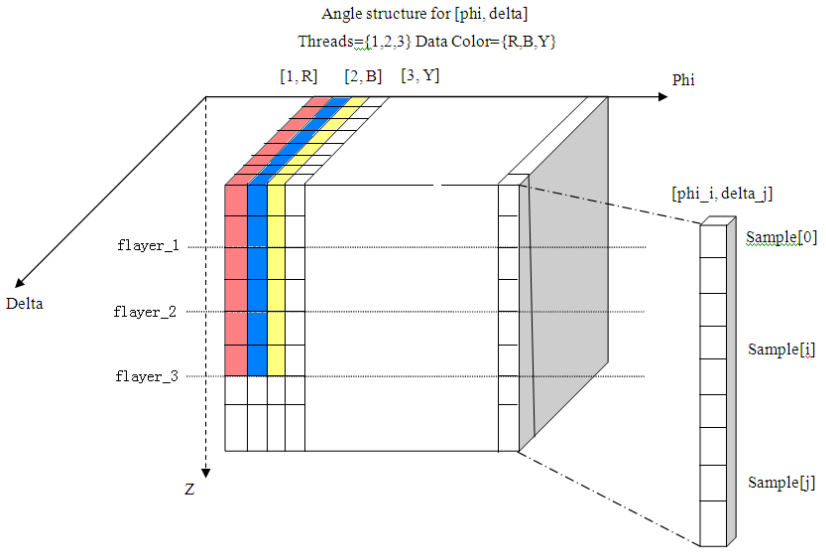
程序并行化设计

并行模型与设计

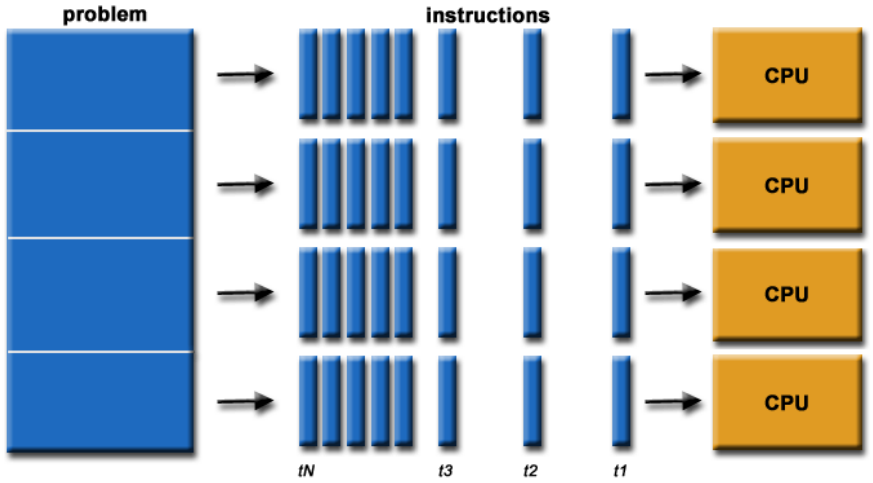
- **<问题求解-程序并行>**映射：物理求解过程与并行处理过程的映射关系分析。
- **任务并行**：根据硬件资源，将多个任务分配给不同硬件进行处理。例如多任务处理系统等；
- **数据并行**：算法执行过程具有对大量数据元素同时做相同或者类似操作的特征。例如偏微分方程求解、蒙特卡洛随机化计算、线性代数、FFT等；



<问题-并行化>映射



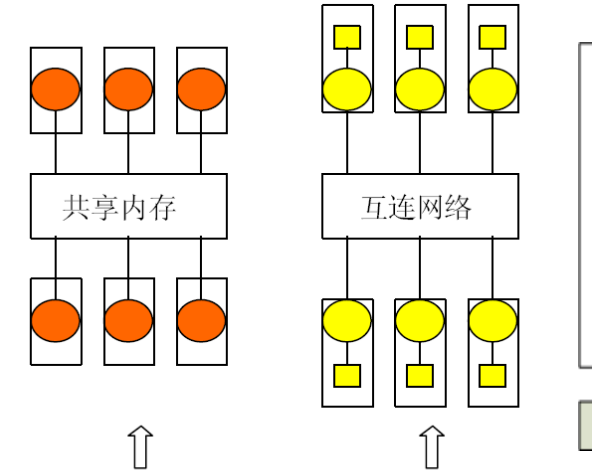
数据并行模型

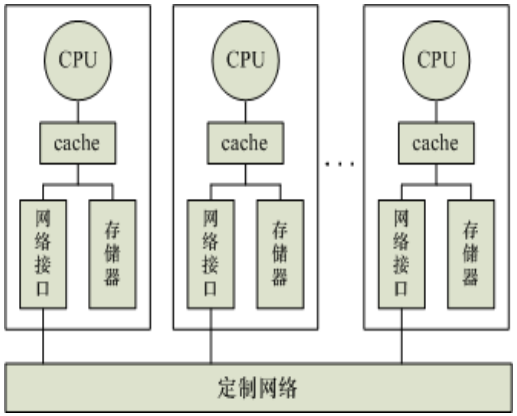


任务并行模型

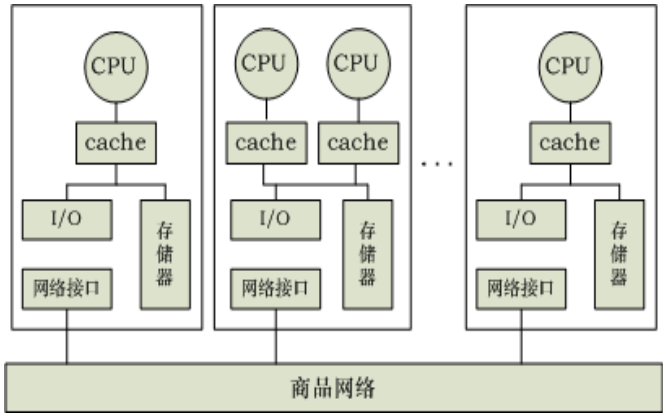
并行计算

- **并行计算机**：即能在同一时间内执行多条指令或处理多个数据的计算机，并行计算机是并行计算的物理载体。包括共享内存并行计算机和分布式内存并行计算机。





MPP



Cluster

- 共享内存并行计算机：各个处理单元通过对共享内存的访问来交换信息，协调各处理器对并行任务的处理
- 共享内存编程实现相对简单，但共享内存往往成为性能特别是扩展性的重要瓶颈。

- 分布式内存并行计算机：各个处理单元都拥有自己独立的局部存储器，各个处理器之间通过消息传递来交换信息协调和控制各个处理器；
- 分布式内存并行程序编程较难，通信对分布式内存并行计算机的性能有重要的影响；
- 但具有较好的扩展性和较高的性能。

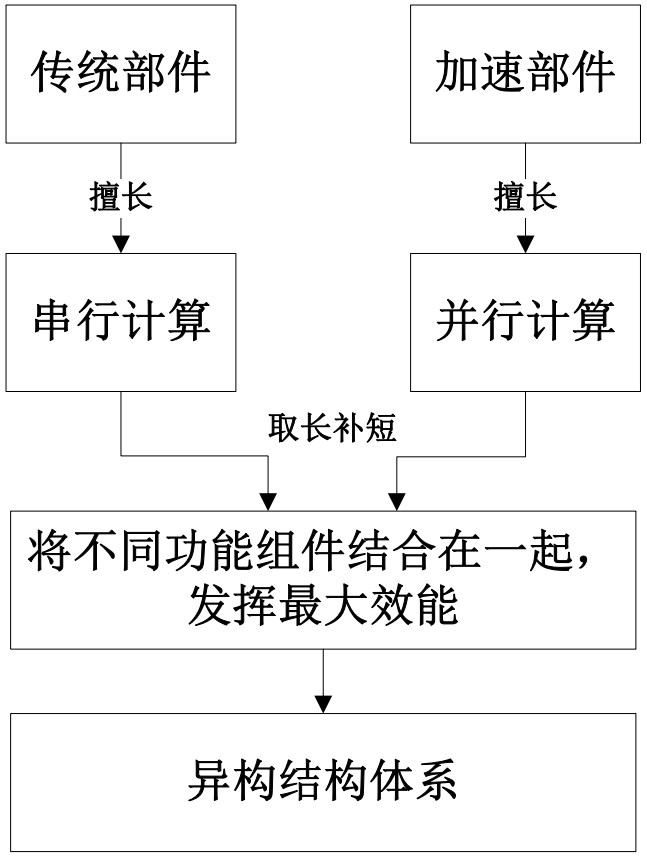
程序并行化设计

- **适用于共享内存的多线程编程模型**
  - 硬件环境：支持超线程的单核CPU、多核CPU，SMP系统，三者组合
  - Win32多线程、Pthread、TBB(intel)、**OpenMP**
- **适用于分布内存的消息传递编程模型**
  - 硬件环境：MPP、Cluster等分布式环境
  - PVM、**MPI**
- **混合编程模型**
  - SMP、MPP、集群等
  - MPI+Pthread、**MPI+OpenMP**、MPI+TBB

- **异构体系结构：传统计算部件+加速计算部件**

  常见的计算单元：CPU、GPU、ARM、DSP、FPGA、ASIC等协处理器

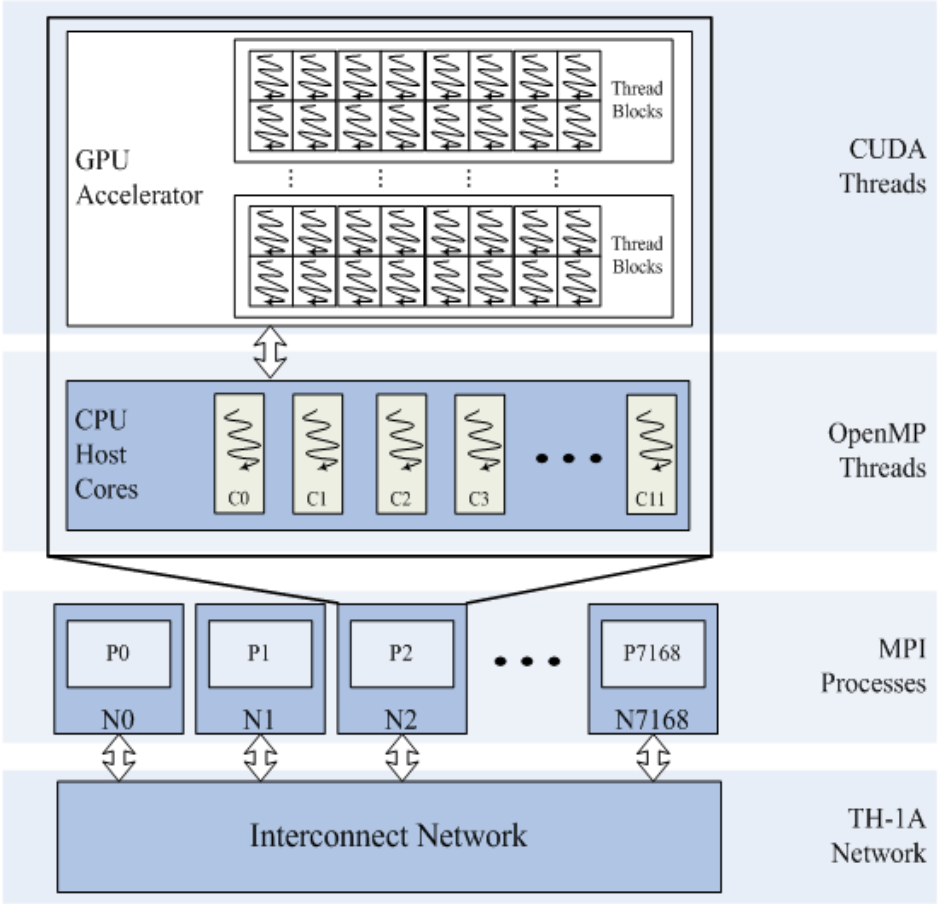
- **异构体系结构三要素：**
  - 两个或以上不同类型的计算核心
  - 连接异构部件的高速网络
  - 将各部件结合，进行协同处理任务的软件支撑环境



“天河一号”异构并行开发

多级混合并行编程模型

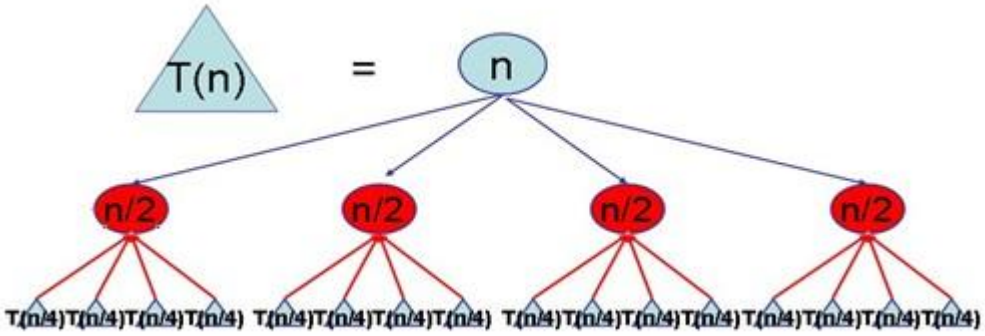
- 体系结构
  - 节点间，对称结构
  - 节点内，异构结构
- 编程模型
  - 节点间，消息传递
  - 节点内，共享存储
    - 纯CPU线程
    - CPU线程（控制GPU线程）
  - MPI+OpenMP+CUDA（OpenACC等）



并行化基本思路：分治法

- 分治法，“分而治之”，把一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题……直到最后子问题可以简单的直接求解，原问题的解即子问题的解的合并。
- 算法模式：

```
1. Divide-and-Conquer(P)
2. if |P|≤n0
3.     then return(ADHOC(P))
4. //将P分解为较小的子问题 P1 ,P2 ,...,Pk
5. for i←1 to k //Parallelism Point
6.     do yi ← Divide-and-Conquer(Pi) // 递归解决Pi
7. T ← MERGE(y1,y2,...,yk) // 合并子问题
8. return(T)
```



矩阵乘法并行化

- 存储方式：对于一个N*N的矩阵，其内存排列方式如下

$$\begin{pmatrix} A[0][0], & A[0][1], & A[0][2], & \cdots, & A[0][n-1] \\ A[1][0], & A[1][1], & A[1][2], & \cdots, & A[1][n-1] \\ . & & & & \\ . & & & & \\ . & & & & \\ A[n-1][0], & A[n-1][1], & A[n-1][2], & \cdots, & A[n-1][n-1] \end{pmatrix}$$

- 矩阵分块：对于一个4Cores的系统，分块矩阵可以进行2等分，分成四个小矩阵快a，每一个Cores对应其中一个小矩阵快

$$\begin{pmatrix} A[0][0], & A[0][1], & A[0][2], & \cdots, & A[0][n-1] \\ A[1][0], & A[1][1], & A[1][2], & \cdots, & A[1][n-1] \\ . & & & & \\ . & & & & \\ . & & & & \\ A[n-1][0], & A[n-1][1], & A[n-1][2], & \cdots, & A[n-1][n-1] \end{pmatrix} = \begin{pmatrix} a[0][0], & a[0][1] \\ a[1][0], & a[1][1] \end{pmatrix}$$

矩阵乘法并行化

- 行列分块并行乘法：

$$A = \left[A_0^{\mathrm{T}} \;\; A_1^{\mathrm{T}} \;\; \cdots \;\; A_{p-1}^{\mathrm{T}} \right]^{\mathrm{T}}, \quad B = \left[B_0 \;\; B_1 \;\; \cdots \;\; B_{p-1} \right] \qquad (1)$$

- 算法描述：伪代码如下

mp1 $\equiv$ myid+1 mod p, mm1 $\equiv$ myid$-$1 mod p
for $i = 0$ to $p-1$ do
  $l \equiv i$+myid mod p
  $C_l = A \times B$
  if $i \neq p-1$, send(B, mm1), recv(B, mp1)
end$\{for\}$

矩阵乘法并行化

- 列行分块并行乘法：

$$A = \begin{bmatrix} A_0 & A_1 & \cdots & A_{p-1} \end{bmatrix}$$

$$B_i = \begin{bmatrix} B_{i0} & B_{i1} & \cdots & B_{i,p-1} \end{bmatrix}$$

（2）

- 算法描述：伪代码如下

$C = A \times B_{\text{myid}}$
for $i = 1$ to $p - 1$ do
  $l \equiv i$+myid mod p, $k \equiv p - i$+myid mod p
  $T = A \times B_l$
  send(T, l), recv(T, k)
  $C = C + T$
end$\{for\}$

蒙特卡罗算法并行化

- 特点：统计模拟方法，采用随机数（伪随机数）解决计算问题，模拟过程相互不会产生影响，没有数据依赖性；
- 应用领域：金融工程学，宏观经济学，生物医学，计算物理学（如粒子输运计算、量子热力学计算、空气动力学计算）等；
- 分子模拟计算步骤：
  1. 使用随机数生成器产生一个随机的分子构型。（MPI Level Parallelism Point）
  2. 对此分子构型的其中粒子坐标做无规则的改变，产生一个新的分子构型。
  3. 计算新的分子构型的能量。（OpenMP Level Parallelism Point）
  4. 比较新的分子构型于改变前的分子构型的能量变化，判断是否接受该构型。
  5. 若新的分子构型能量低于原分子构型的能量，则接受新的构型，使用这个构型重复再做下一次迭代。
  6. 若新的分子构型能量高于原分子构型的能量，则计算玻尔兹曼因子，并产生一个随机数。
     a) 若这个随机数大于所计算出的玻尔兹曼因子，则放弃这个构型，重新计算。
     b) 若这个随机数小于所计算出的玻尔兹曼因子，则接受这个构型，使用这个构型重复再做下一次迭代。

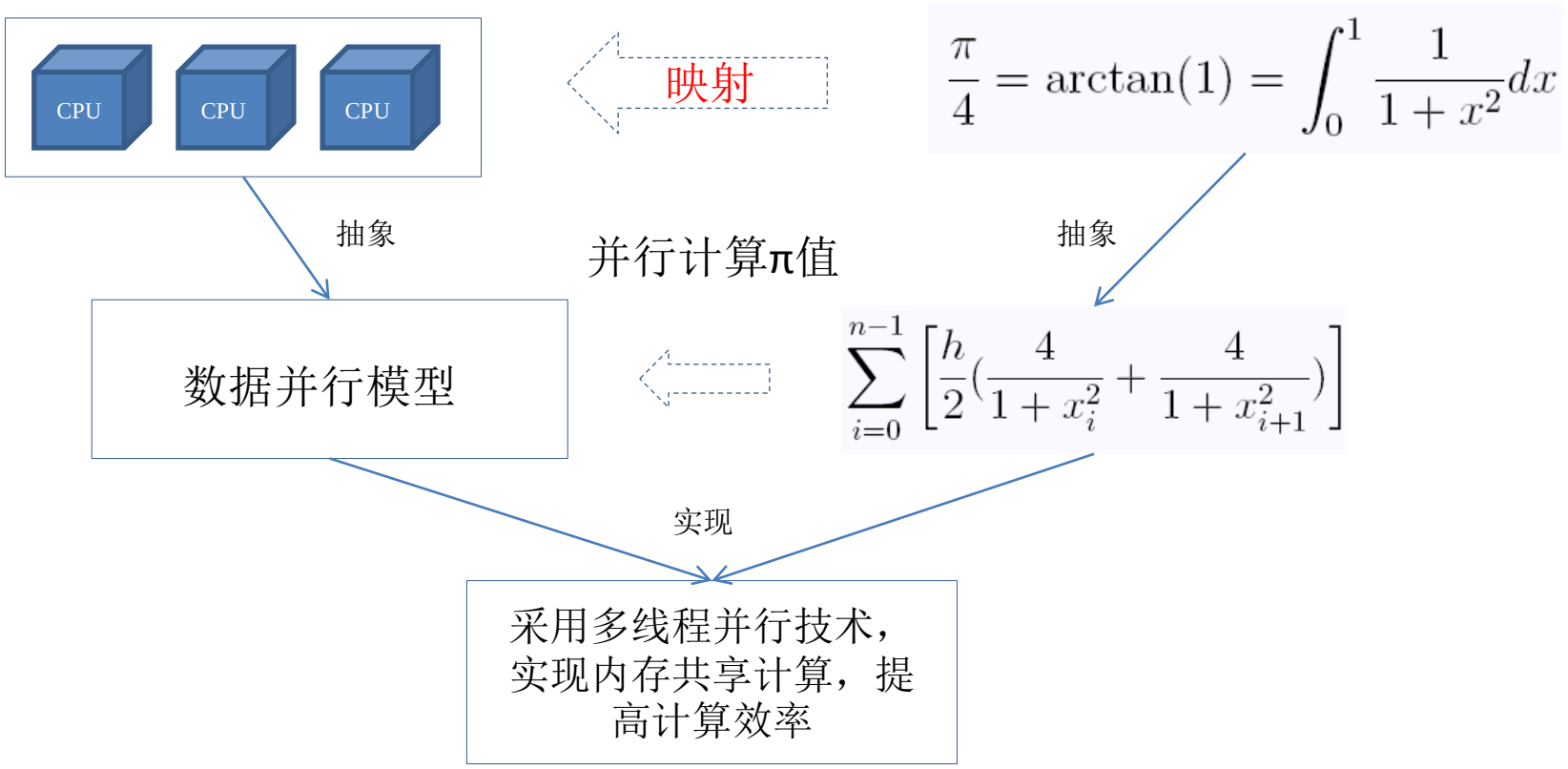
概要

- 程序并行化设计
- 程序并行化实现
- 程序并行优化
- 应用案例
- 未来合作

程序并行化实现

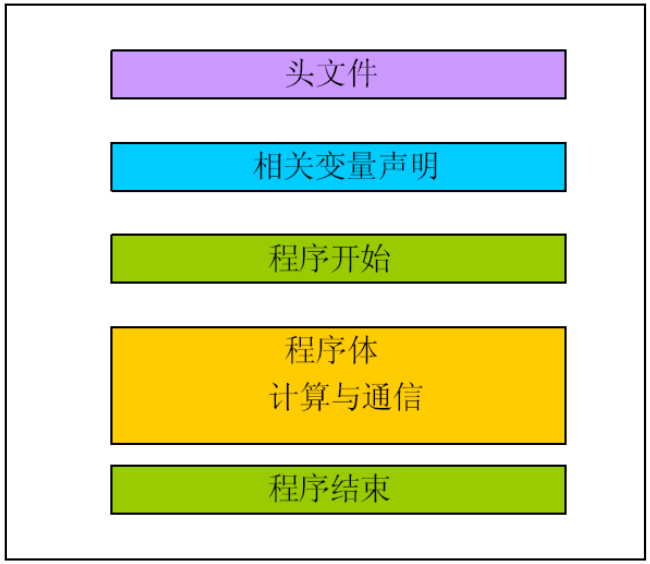
并行方案实现

- 任务划分：分析物理求解过程中数据并行和任务并行的区域和比例；
- 处理机分配与映射：根据物理求解过程，设计任务并行方案，根据硬件资源确定处理机与任务的映射关系；
- 通信与同步：考虑求解过程中的数据通信与同步要求，确保任务并行的基础上，实现数据并行；
- 算法框架：根据上述分析结果，确定并行算法框架，采用MPI、OpenMP、MPI+OpenMP、MPI+OpenMP+CUDA等并行模型实现。



MPI（Message Passing Interface）

- MPI并行库：一种新的库描述，不是一种语言，共有上百个函数调用接口，能够被Fortran、C、C++等语言调用。
- MPI标准规范：成为并行编程的标准，一个正确的MPI程序可以不加修改地在所有的并行机上运行
- MPI消息传递编程模型：MPI并行编程的目的就是服务于进程间通信，实现多进程间的数据交互，多应用在分布式内存并行计算机。
  - **较高的通信性能**，每个进程均有自己独立的地址空间，相互之间访问不能直接进行；通过显式地发送和接收消息来实现进程间的数据交换（初始化问题）
  - 较好的程序移植性
  - 较好的编程性



MPI程序框架图

程序并行化实现

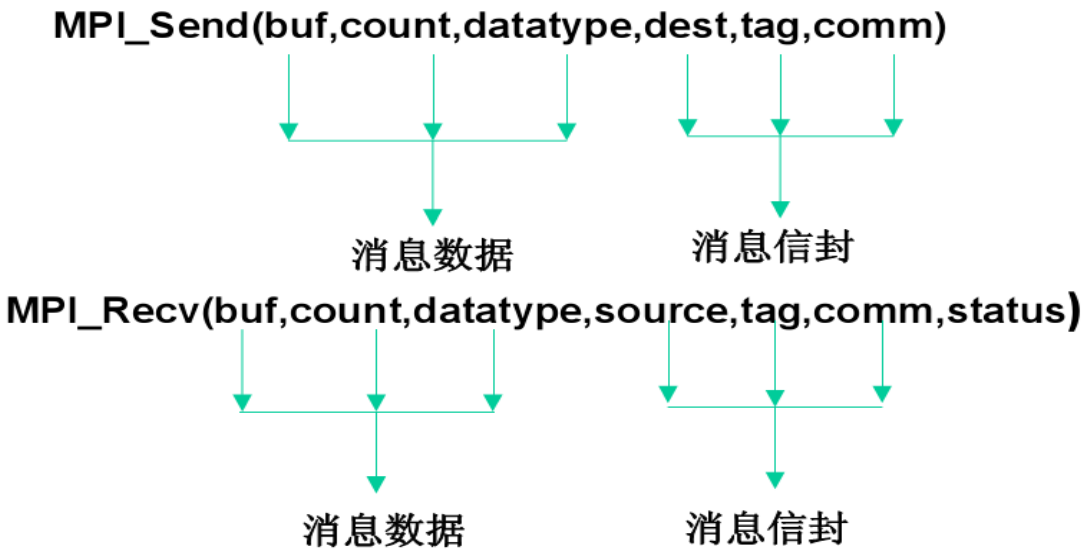
1. MPI初始化：通过MPI_Init函数进入MPI环境并完成所有的初始化工作。
   - ✓ int MPI_Init(int *argc, char * * * argv)
2. MPI结束：通过MPI_Finalize函数从MPI环境中退出。
   - ✓ int MPI_Finalize(void)
3. 获取进程的编号：调用MPI_Comm_rank函数获得当前进程在指定通信域中的编号（0 – n-1)，将自身与其他程序区分。
   - ✓ int MPI_Comm_rank(MPI_Comm comm, int *rank)
4. 获取指定通信域的进程数：调用MPI_Comm_size函数获取指定通信域的进程个数，确定自身完成任务比例。
   - ✓ int MPI_Comm_size(MPI_Comm comm, int *size)
5. 通信域上下文的复制：调用MPI_Comm_dup函数将指定通信域进行复制，获得完全一样的通信域上下文环境
   - ✓ int MPI_Comm_dup (MPI_Comm comm, MPI_Comm *newcomm)
6. 通信域上下文的分割：调用MPI_Comm_split函数将指定通信域进行分割，实现子通信域的各自通信
   - ✓ int MPI_Comm_split（MPI_Comm comm, int color, int key, MPI_Comm *newcomm)

程序并行化实现

7. 消息发送：MPI_Send函数用于发送一个消息到目标进程。
   - ✓ int MPI_Send(void *buf, int count, MPI_Datatype dataytpe, int dest, int tag, MPI_Comm comm)

8. 消息接受:MPI_Recv函数用于从指定进程接收一个消息
   - ✓ int MPI_Recv(void *buf, int count, MPI_Datatype datatyepe,int source, int tag, MPI_Comm comm, MPI_Status *status)

➢ 一个消息好比一封信



天河一号编译环境（登录节点）

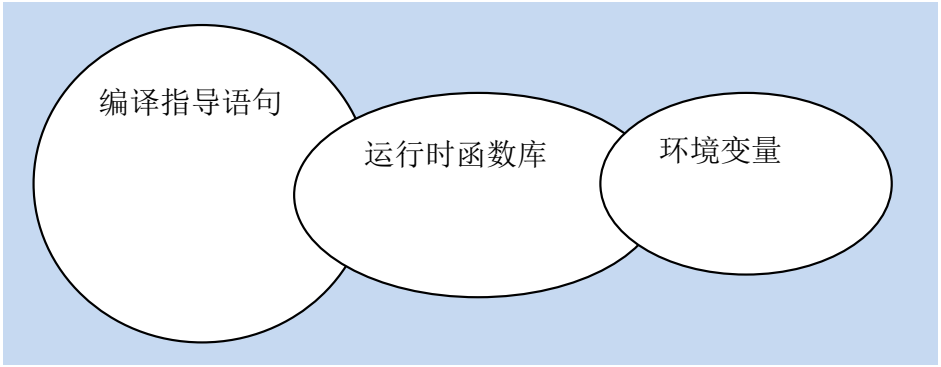
- Intel编译器（推荐）
- Gcc编译器
- MPI编译环境
  - 基于Intel编译器的mpi版本：/usr/local/mpi（推荐）
  - 基于gcc编译器的mpi版本：/usr/local/mpi-gcc

- 支持自己安装的MPI，如OpenMPI、MPICH2等，但通信效率相对天河自主MPI会低很多

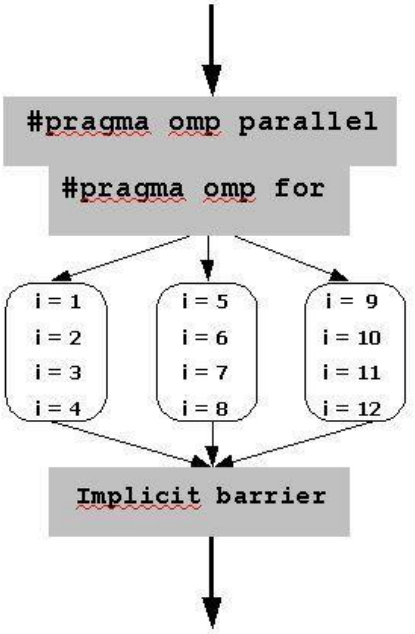
程序并行化实现

OpenMP（Open Multi-Processing）

- OpenMP并行库：支持跨平台共享内存方式的多线程编程并行库；
- 并行框架：具备高可移植性、高可扩展性，其并行函数库能够被C、C++、Fortran等语言调用；
- 特点：降低了并行编程的难度和复杂度，简化具体实现细节，通过相对简单的指导语句，实现并行化，多应用在共享内存式并行计算机。



OpenMP 组成



OpenMP 并行框架

程序并行化实现

```
#include <stdio.h>
#include “omp.h” //openMP头文件
int main(int argc,char *argv[] ){
   int nthreads, tid;
   omp_set_num_threads(4); //设置并行区域线程数目
   #pragma omp parallel    //编译指导语句，创建一个并行区
   {
      //得到并行区域中总的线程数目
      nthreads = omp_get_num_threads( );
      //得到当前线程的标识号
      tid = omp_get_thread_num( );
      printf (“Hello  World! Thread %d of %d”, tid, nthreads);
   }
   return 0;
}
```

运行时函数

编译指导语句

运行4个线程结果：

```
Hello World! Thread 0 of 4
Hello World! Thread 2 of 4
Hello World! Thread 3 of 4
Hello World! Thread 1 of 4
```

概要

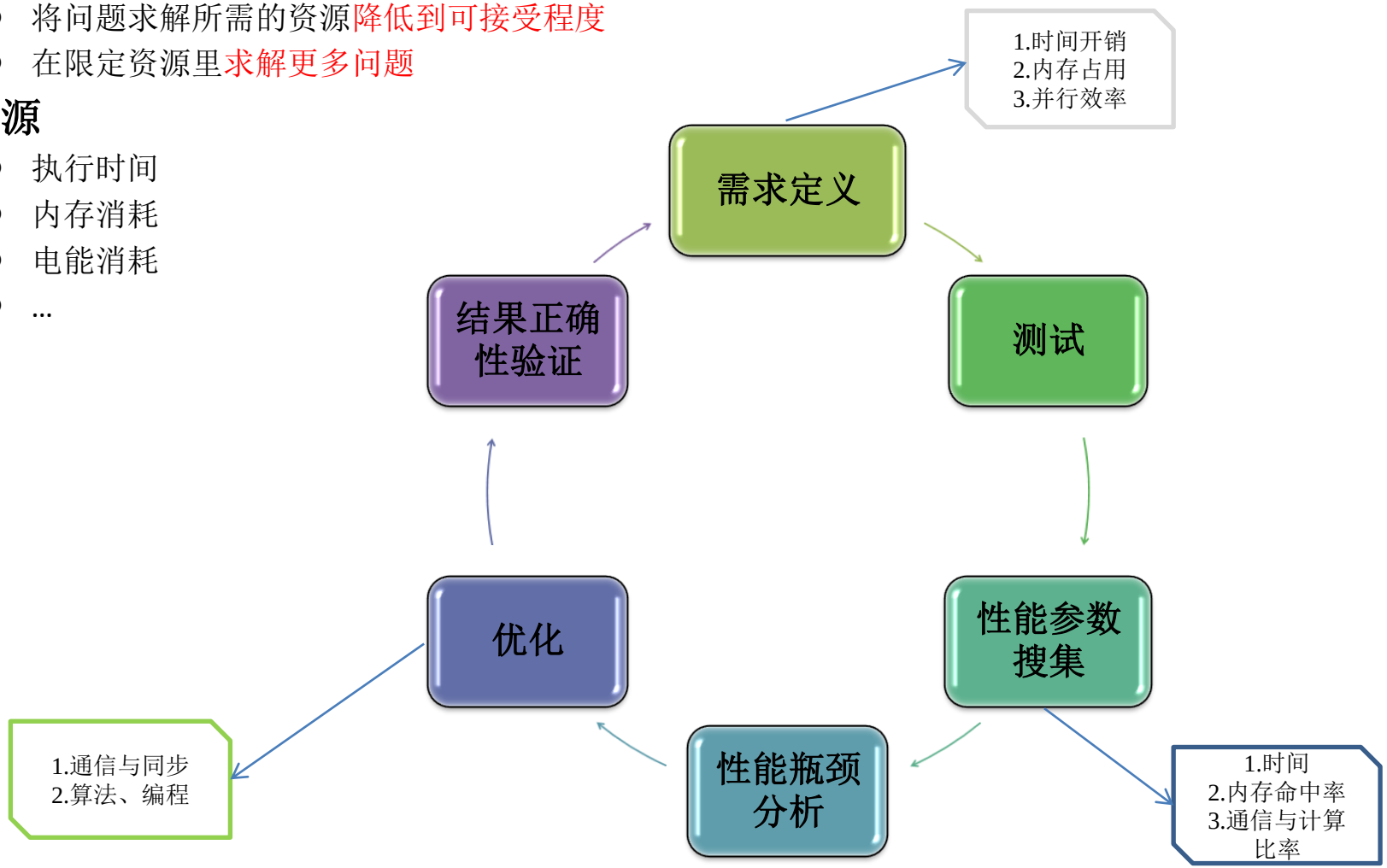
- 程序并行化设计
- 程序并行化实现
- 程序并行优化
- 应用案例
- 未来合作

并行与分布式系统的复杂性

- 应用
  - 算法，数据结构
- 并行编程接口
  - 编译器，并行库，通信与同步…
- 操作系统和文件系统
  - 进程与存储管理， I/O
- 硬件
  - CPU, Cache, memory, network

程序并行优化

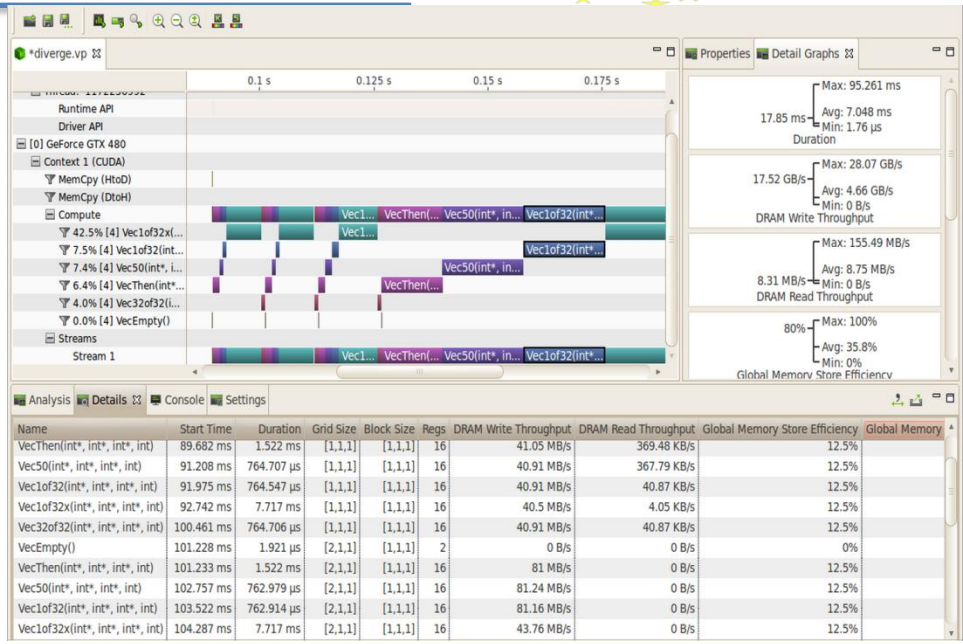
- **性能优化的目标**
  - 将问题求解所需的资源降低到可接受程度
  - 在限定资源里求解更多问题
- **资源**
  - 执行时间
  - 内存消耗
  - 电能消耗
  - …

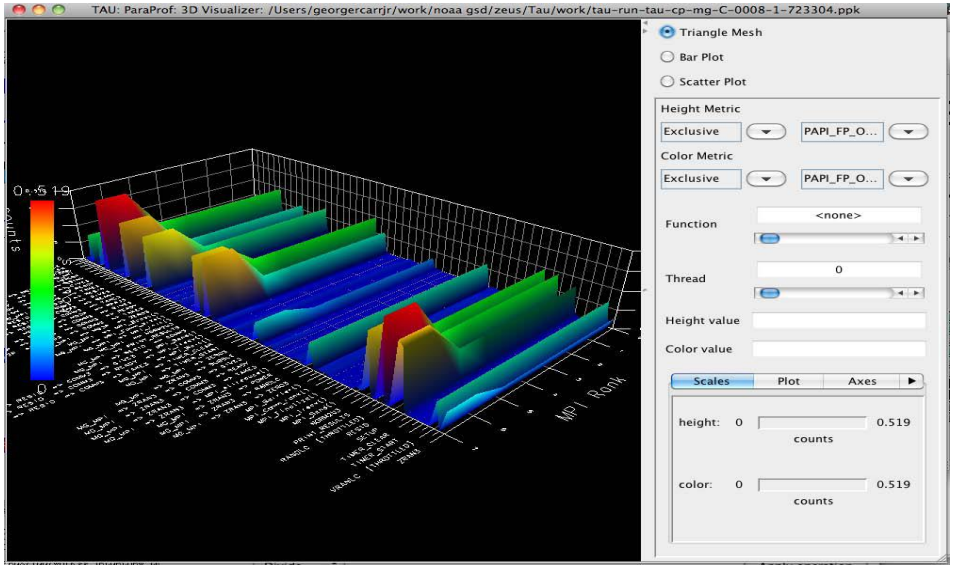


并行优化周期

并行开发工具

- 性能分析工具
  - TAU
  - SCALASCA
  - VTune
  - NVIDIA Visual Profiler
  - ……
- 调试工具
  - CUDA-GDB
  - ……
- 辅助工具
  - ADIOS等





程序并行优化

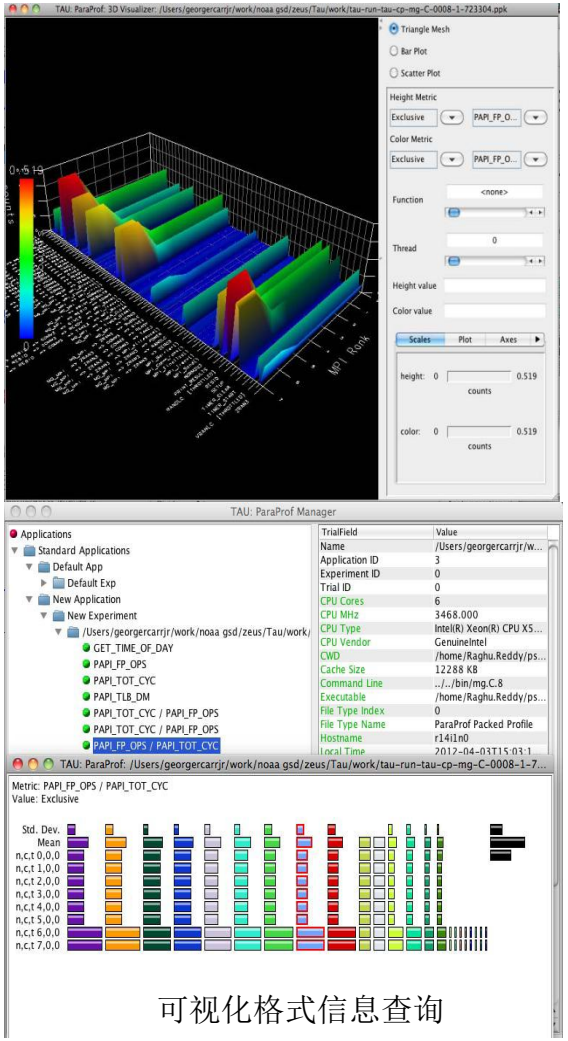
- 性能参数搜集阶段，可以采用TAU、SCALASCA和VTune等性能分析工具，对程序运行过程中的热点、瓶颈等进行侦测，获得程序的性能分布图，能够有效的开展算法设计、数据结构设计和并行框架设计等方面的优化工作。

- TAU (Tuning and Analysis Utilities)：性能分析和跟踪工具包，开源软件，支持数据采集、分析、跟踪由多个结点组成的复杂系统或集群。

```
FUNCTION SUMMARY (total):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0          303        3,649          20          14     182478 .TAU application
 85.2            6        3,109           2           8    1554786 .TAU application => main
 85.2            6        3,109           2           8    1554786 main
 84.1        3,070        3,070           2           0    1535462 .TAU application => main => MPI_Init()
 84.1        3,070        3,070           2           0    1535462 MPI_Init()
  6.5            1          236           6           6      39380 .TAU application => .TAU application
  6.4          234          234           6           0      39083 .TAU application => .TAU application => .TAU application
  0.9           31           31           2           0      15939 .TAU application => main => MPI_Finalize()
  0.9           31           31           2           0      15939 MPI_Finalize()
  0.0        0.114        0.114           2           0         57 .TAU application => main => MPI_Comm_size()
  0.0        0.114        0.114           2           0         57 MPI_Comm_size()
  0.0        0.083        0.083           2           0         42 .TAU application => main => MPI_Comm_rank()
  0.0        0.083        0.083           2           0         42 MPI_Comm_rank()

FUNCTION SUMMARY (mean):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0           37          456         2.5        1.75     182478 .TAU application
 85.2        0.821          388        0.25           1    1554786 .TAU application => main
 85.2        0.821          388        0.25           1    1554786 main
 84.1          383          383        0.25           0    1535462 .TAU application => main => MPI_Init()
 84.1          383          383        0.25           0    1535462 MPI_Init()
  6.5        0.222           29        0.75        0.75      39380 .TAU application => .TAU application
  6.4           29           29        0.75           0      39083 .TAU application => .TAU application => .TAU application
  0.9            3            3        0.25           0      15939 .TAU application => main => MPI_Finalize()
  0.9            3            3        0.25           0      15939 MPI_Finalize()
  0.0       0.0143       0.0143        0.25           0         57 .TAU application => main => MPI_Comm_size()
  0.0       0.0143       0.0143        0.25           0         57 MPI_Comm_size()
  0.0       0.0104       0.0104        0.25           0         42 .TAU application => main => MPI_Comm_rank()
  0.0       0.0104       0.0104        0.25           0         42 MPI_Comm_rank()
[xiazj@ln1%tianhe output_profile1]$
```

文本格式信息查询

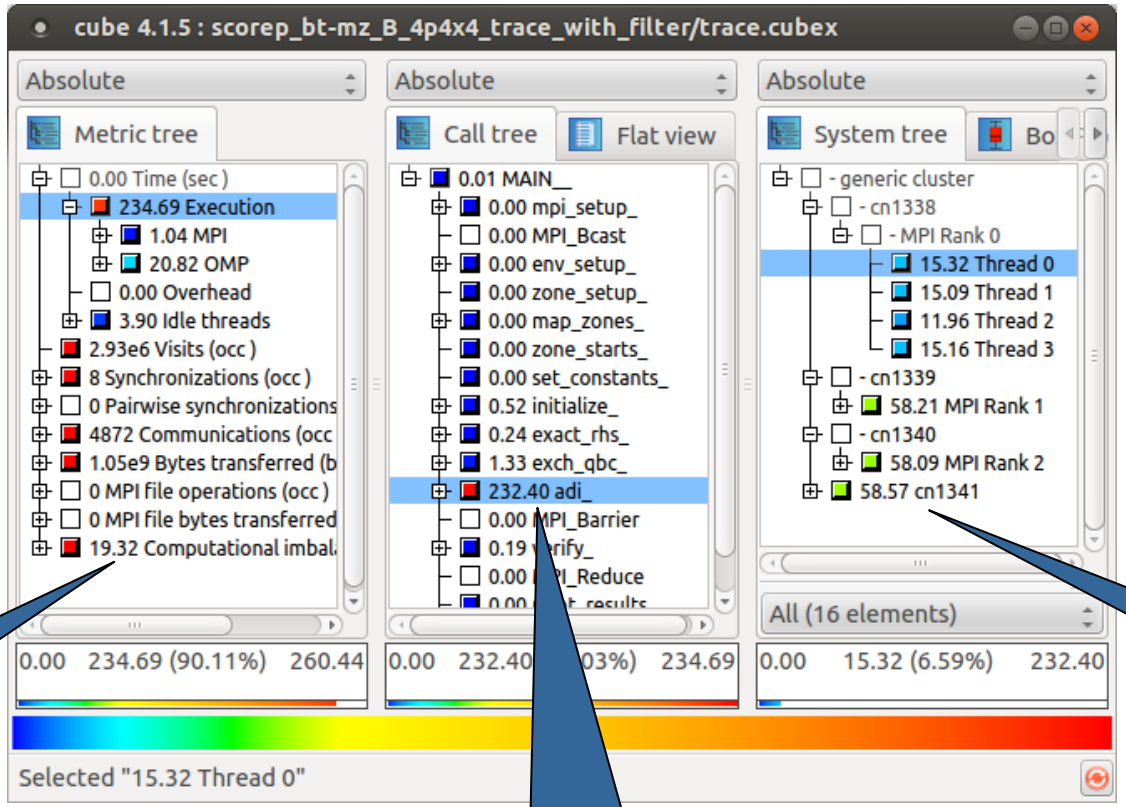


可视化格式信息查询

程序并行优化

SCALASCA：Scalable performance analysis of large-scale parallel applications

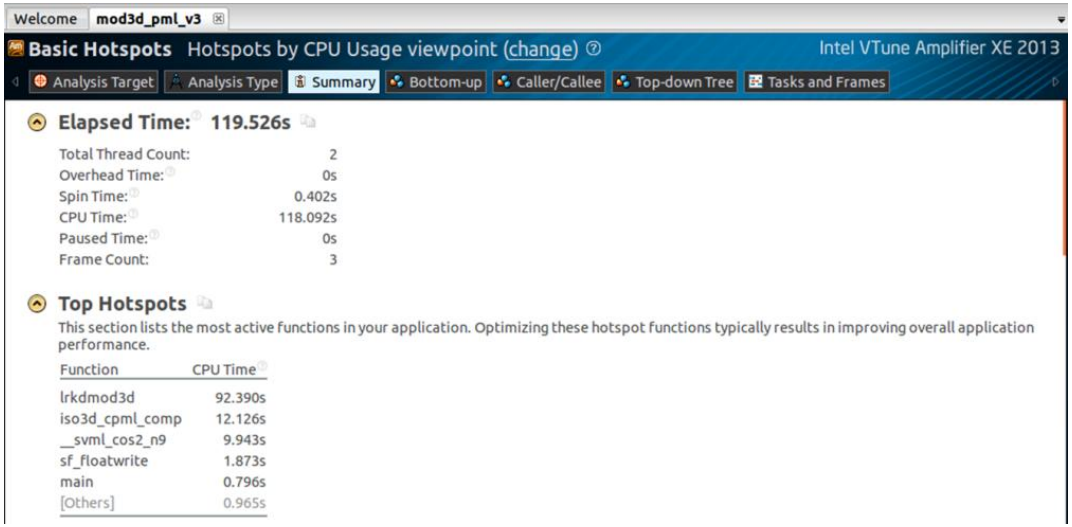
- 起始于2006年，开源软件，支持MPI、OpenMP以及混合并行程序的可扩展性能测量与分析；
- 集成代码插桩、性能测量以及数据分析的完整工具集：
  - 提供可定制的自动/手动代码插桩
  - 支持基于剖析（Profiling）、事件跟踪（Event Trace）的程序性能监控



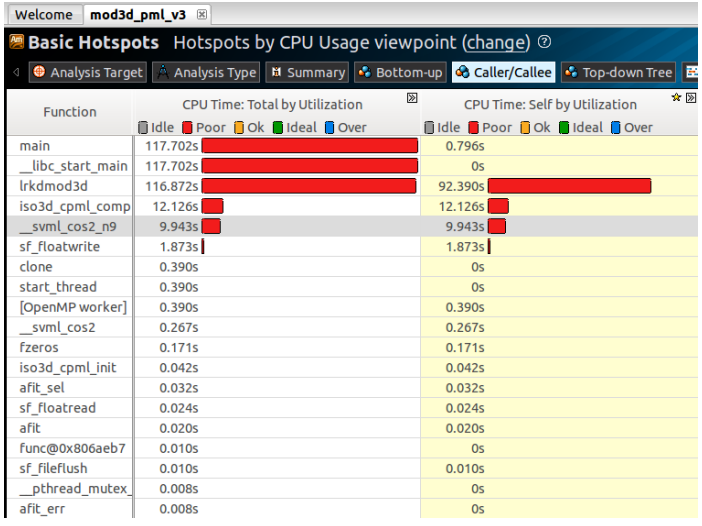
程序并行优化

Intel Vtune Amplifier XE

- Intel 商业性能分析器，图形化用户使用界面，功能和特点包括：
  - 无需重新编译插桩；
  - 支持C、C++、Fortran、Java、.NET等编程语言；
  - 串/并行程序的动/静态插桩；
  - 程序运行态事件追踪和记录；
  - 性能统计结果图形化分析；
  - 内存使用情况监测和调试。



程序性能概览



程序函数调用开销分析

概要

- 程序并行化设计
- 程序并行化实现
- 程序并行优化
- 应用案例
- 未来合作

应用案例 I

地球物理抽道集

- 逆时偏移抽道集程序是将叠前逆时偏移成像后的成像结果按成像点和成像偏移距进行重新排列，以便于后期进行成像噪音压制和成像效果提高的一个处理步骤。

$$I_P\ (s, x, y, z) \Longrightarrow I_{P'}\ (x, y, o, z)$$

- 数据I/O压力大：文件总大小达到近100TB；

技术挑战

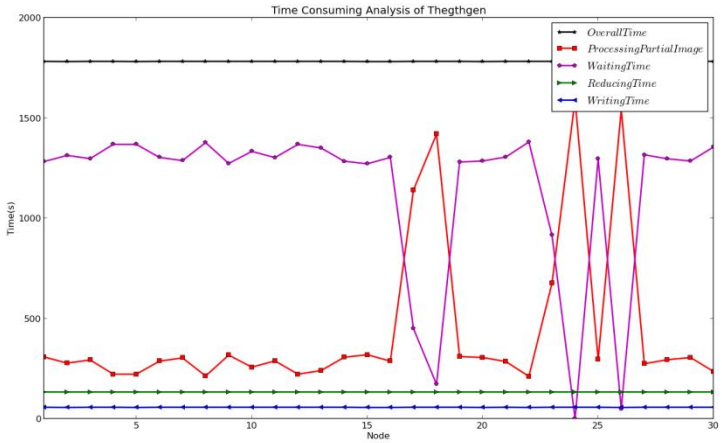
- 节点负载不均衡：单个节点计算涉及2000~3000个文件，数据容量近4TB；

处理时间与逆时偏移本身相当！

应用案例 I

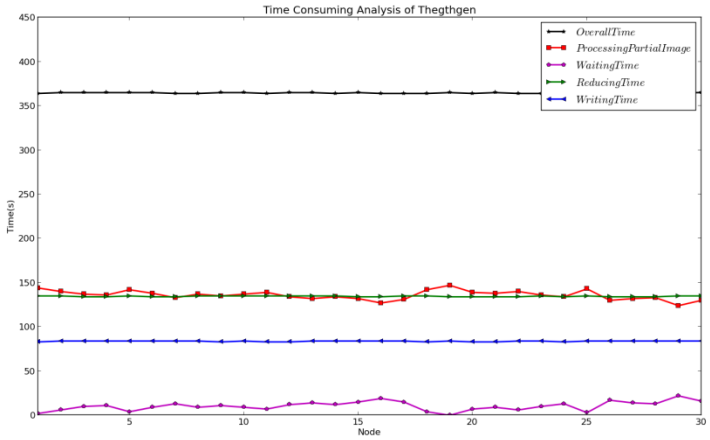
存在问题	应对策略
数据无共享，文件利用率低，重复读取严重，I/O负责高	工区扫描，避免二次读取，降低系统I/O负载
内存占用大，利用率低	压缩数据分布于各个计算节点内存，使用完后立即清除
重复解压，计算负载大	优化压缩解压缩算法，提升解压缩效率
任务并行：节点间负载不平衡	优化作业各节点负载，实现系统整体负载平衡
数据常驻内存后，内存占用量较大	数据列表均匀拆分，用时间换空间，减小内存占用

应用案例 I



Overall time:	Processing:	Waiting:	Reducing:	Writing:
1782.80	442.37	1149.67	133.00	57.77

负载均衡前各节点处理时间统计

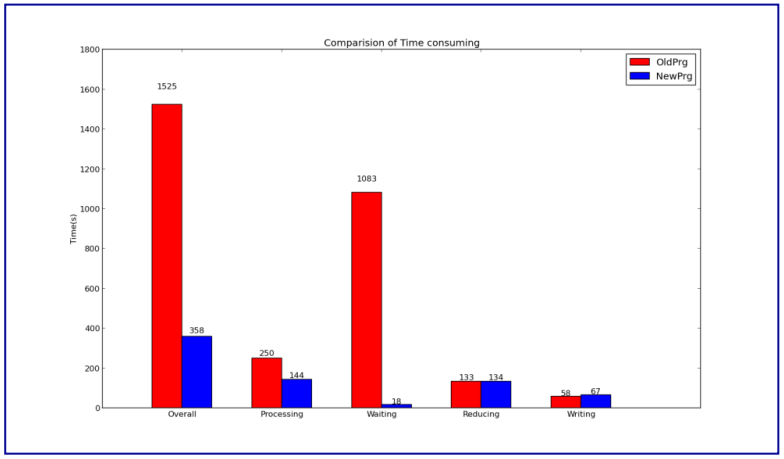


Overall time:	Processing:	Waiting:	Reducing:	Writing:
364.53	135.87	10.40	134.53	83.73

负载均衡后各节点处理时间统计

逆时偏移抽道集优化加速

- 利用TAU、SCALASCA等专业性能分析软件，寻找软件性能瓶颈；
- 针对各个性能瓶颈，开展优化加速工作；
- 优化后，数据重排处理效率提升3x，程序整体执行效率提升5x。



应用案例 II

气候变化：NEMO的I/O优化

- 高性能计算的处理效率取决于：计算能力、带宽、存储
- 存在问题：NEMO 模式在进行大量进程并发I/O时会导致整体应用性能下降

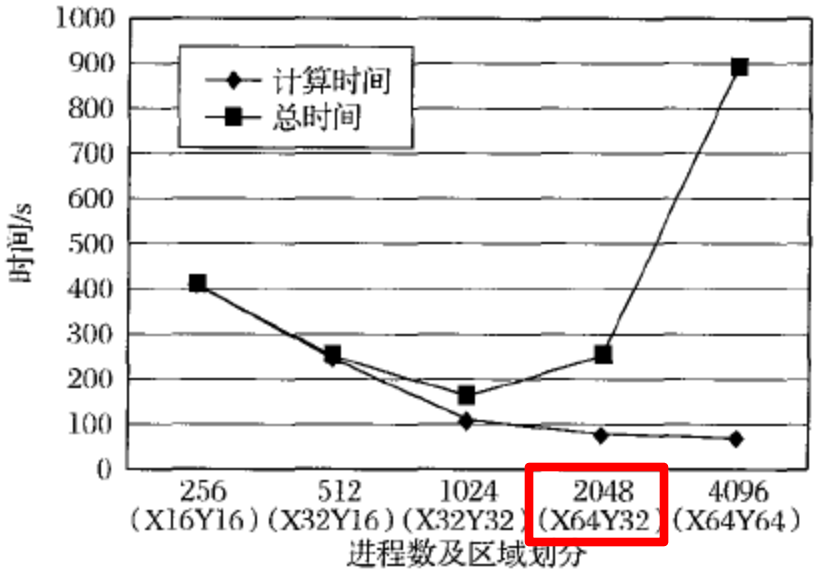


图 1　GYRE012 全球算例计算和总时间变化

表 1　不同进程数时 GYRE012 全球算例测试结果

进程数及区域划分	输出文件大小/GB	存储时间/s	总运行时间/s	存储占总时间百分比/%	存储速率/GBps
256(X16Y16)	13	4	409	0.98	3.25
512(X32Y16)	26	9	252	3.57	2.89
1024(X32Y32)	57	57	163	34.97	1.82
2048(X64Y32)	177	177	252	70.24	2.34
4096(X64Y64)	822	822	890	92.36	2.01

应用案例 II

解决方法

- 并发进程分组输出，通过将大量并发存储的进程进行合理分组并排队输出，以解决大量进程同时读写文件时对存储资源的竞争所导致的存储效率下降问题。

表 2　GYRE012 全球算例分组输出优化测试结果

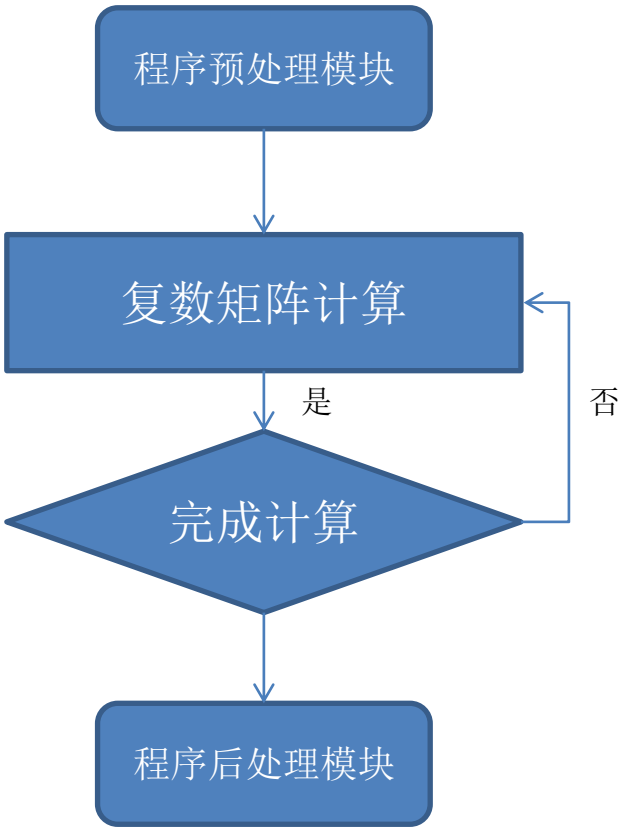
进程及区域划分	优化方案	计算时间/s	存储时间/s	总运行时间/s	存储速率/GBps
1 024(X32 Y32)	无分组	106	57	163	1.82
	256/组	107	41	148	2.54
	512/组	103	37	140	2.81
2 048(X64 Y32)	无分组	75	177	252	2.34
	256/组	72	154	226	2.69
	512/组	77	143	220	2.90
4 096(X64 Y64)	无分组	68	822	890	2.01
	256/组	78	666	744	2.48
	512/组	92	543	635	3.04

NEMO 模式GYRE012 算例的存储性能最高可提升33 %，总体时间性能最高可提升28%。

应用案例 III

电子跃迁模拟程序
并行优化

- 计算处理过程中存在复数矩阵计算的循环，循环计算模块超过程序整体时间开销的95%；
- 每次循环内部存在多次复数矩阵乘法操作，由于矩阵维度大，导致内存压力大、计算时间开销大。



矩阵大小：5GB　循环次数：14

模块	时间开销(S)	比例
rtgw	2315.2	73.5%
Gemm_prdtv	765.2	24.3%

应用案例 III

优化思路及方案

- 每次循环计算的结果相互之间不存在依赖关系；
- 复数矩阵乘法进行线程并行计算，或GPU加速计算。

- 循环展开，采用MPI并行库实现数据并行计算；
- 利用共享内存计算，对复数矩阵乘法进行数据并行计算；

表 1 rtgw_TB_wraper 模块并行优化测试结果

目标模块 时间开销(S)	进程核数							加速比	并行效率(%)
	1	14							
		1	2	3	4	5	平均		
BLAS_Gemm_prdt	73.48	11.02	12.41	12.66	16.00	15.82	13.5	5.4	38.64
rtgw_iter	2315.6	229.1	228.9	230.8	230.8	230.70	230.1	10	71.88
BLAS_Gemm_mprdtv	765.2	50.5	49.2	52.4	57.21	52.09	52.3	14.6	104.5
program	3149.3	372.5	368.2	360.0	363.2	355.6	363.9	8.65	61.8

概要

- 程序并行化设计
- 程序并行化实现
- 程序并行优化
- 应用案例
- 未来合作

未来合作

- **自主软件社区**
  - 基于自主超级计算硬件环境，面向重大重点应用领域，通过协同创新，合作开发并行、GPU等自主高性能软件，提升我国高性能计算应用水平
- **大数据**
  - 通过双方合作，对相关研究领域的观测、研究数据构建完善的数据管理模型（特征提取、整合、检索）
  - 构建行业大数据平台，为生产、科研服务，成为领域支撑
- **系统更新**
  - 双方合作，积极推动计算、数据环境的更新，形成可持续发展的硬件基础

谢 谢

微信号：897963835