大规模并行计算程序优化

高性能服务器产品部

应用支持处

刘羽 博士

inspur 浪潮

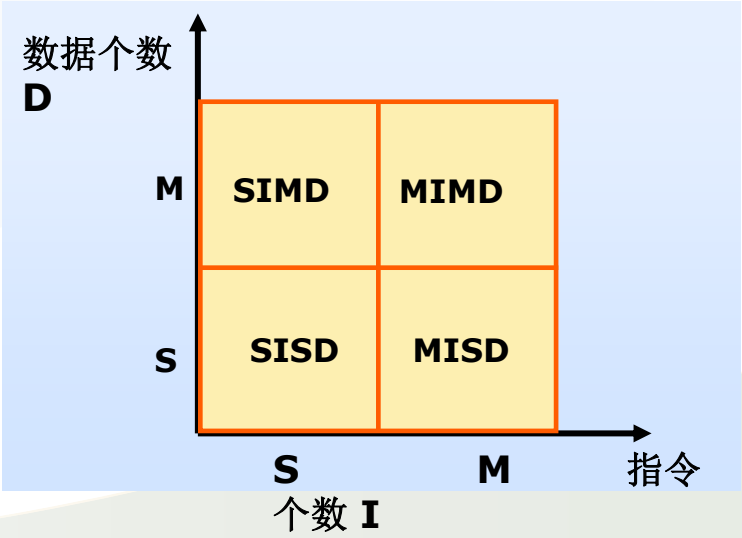
主要内容

- 并行计算机体系结构
- 并行程序设计及编译优化
- TAU分析器简介

并行计算机体系结构

并行计算机结构模型

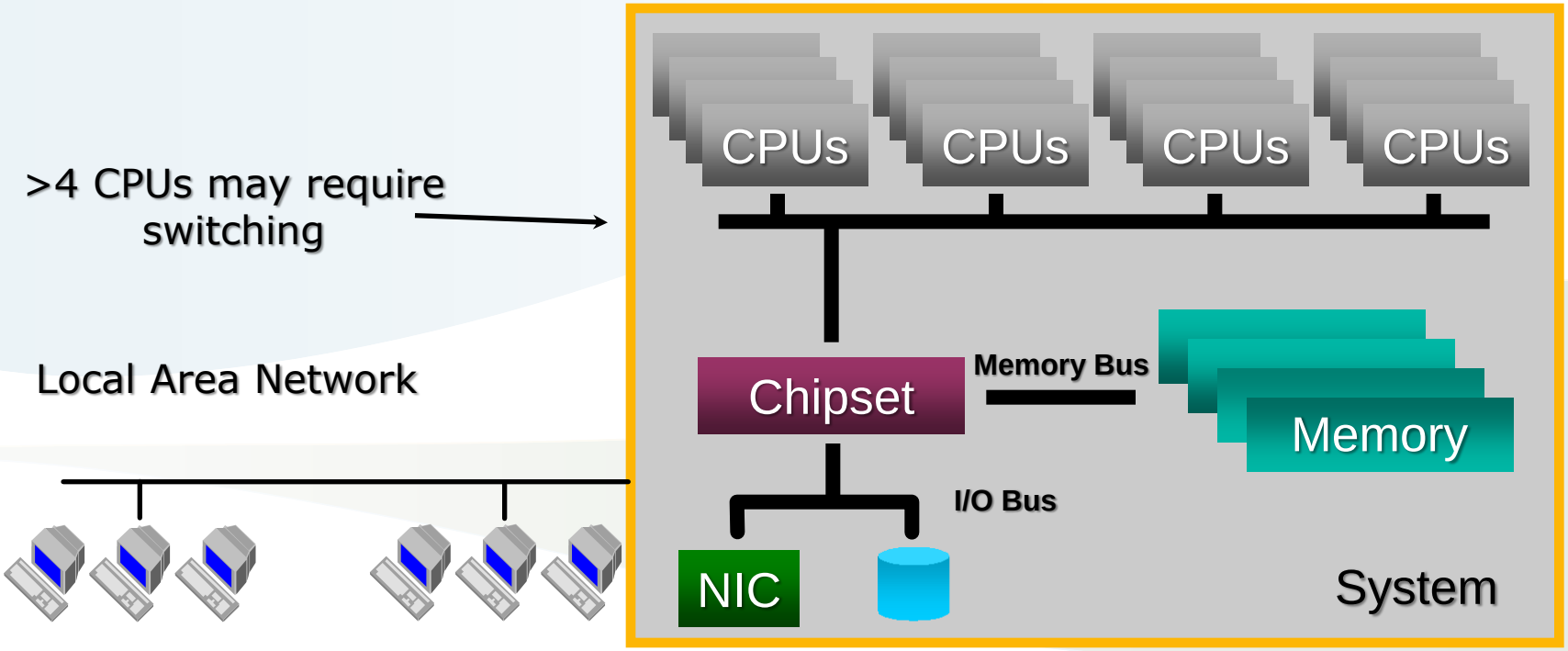
- 根据指令流和数据流的不同，通常把计算机系统分为：
  - 单指令单数据流 **(SISD)**
  - 单指令多数据流 **(SIMD)**
  - 多指令单数据流 **(MISD)**
  - 多指令多数据流 **(MIMD)**



- 并行计算机系统多大部分为**MIMD**系统，包括：
  - 对称多处理机（**SMP**）
  - 分布式共享存储多处理机（**DSM**）
  - 大规模并行处理机（**MPP**）
  - 机群（**Cluster**）

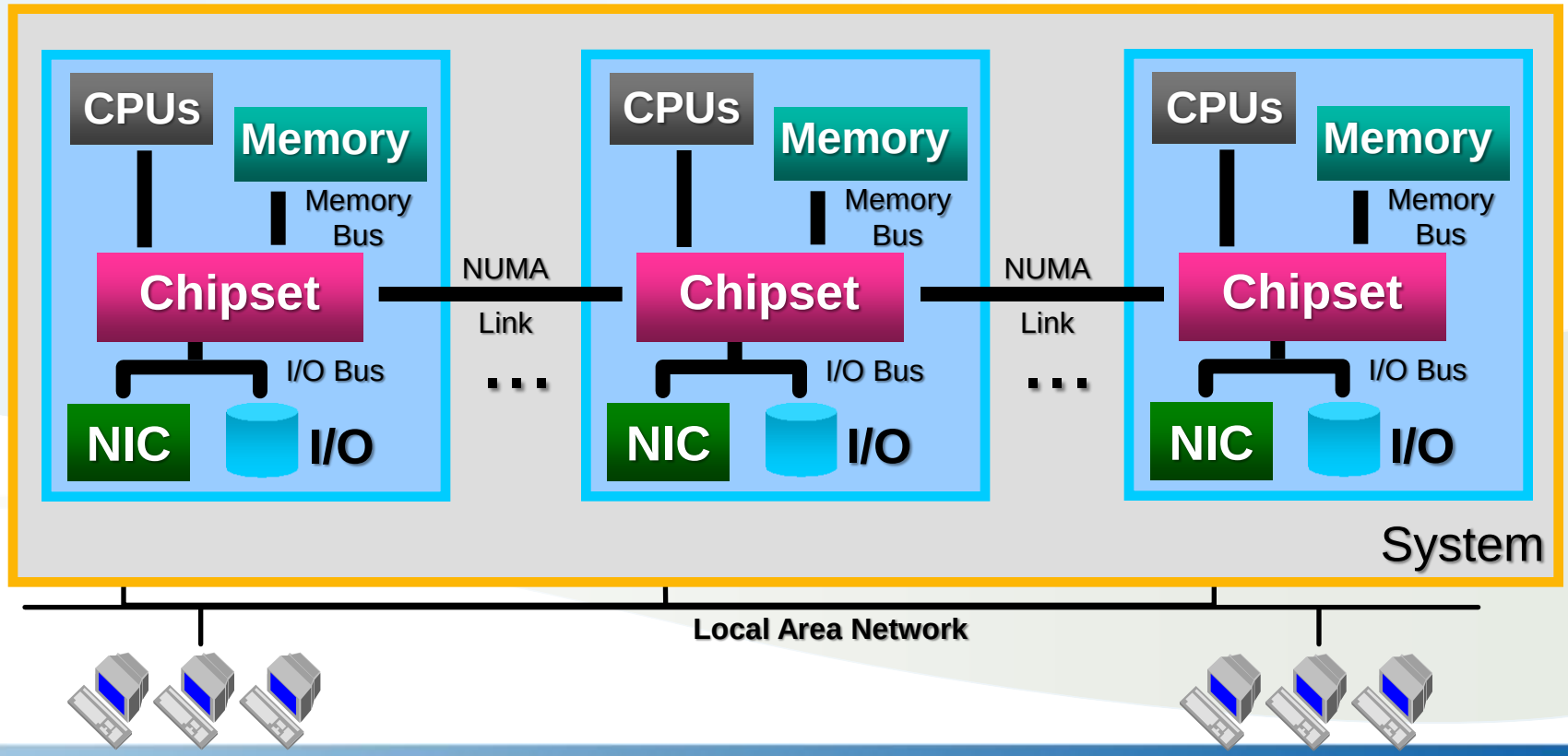
SMP- Symmetric MultiProcessing

- 多个CPU连接于统一的内存总线
- 内存地址统一编址，单一操作系统映像
- 可扩展性较差，一般CPU个数少于32个
- 目前商用服务器多采用这种架构



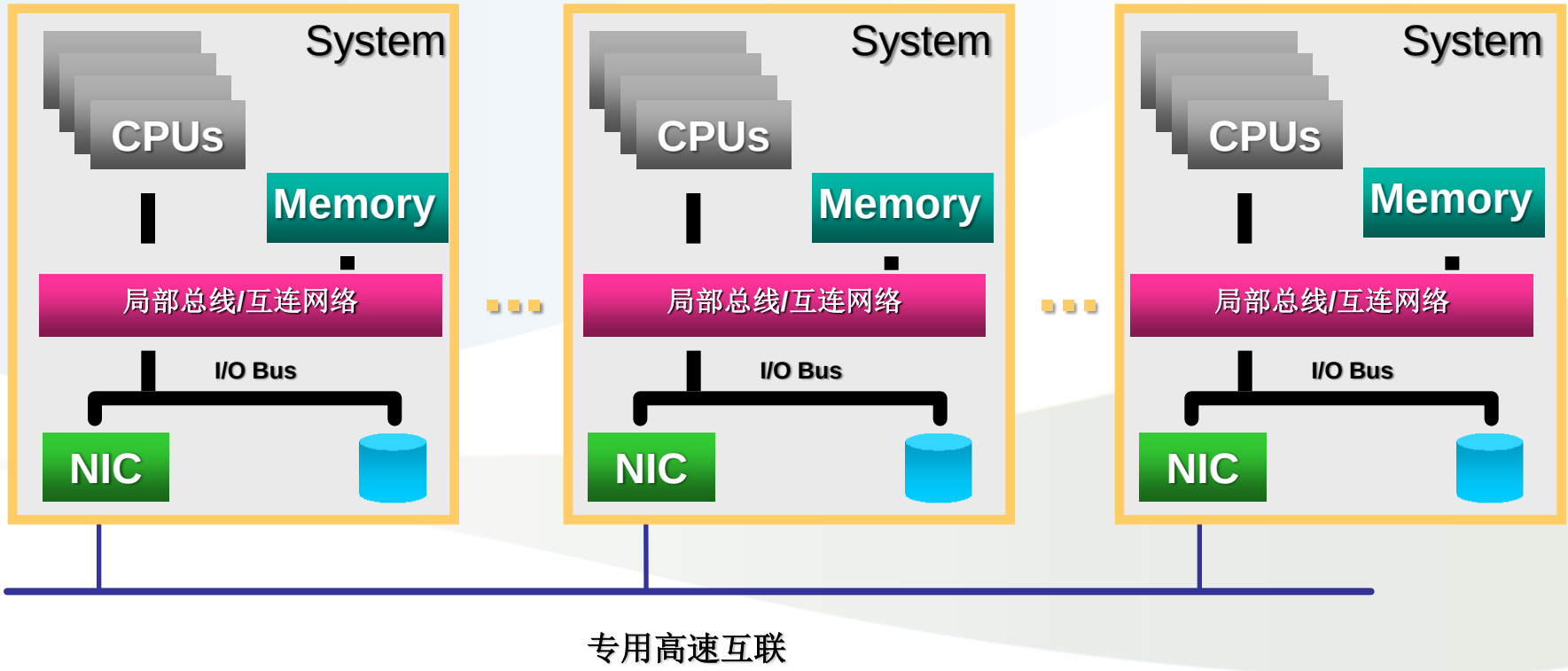
DSM- Distributed Shared Memory

- 物理上分布存储、所有内存模块统一编址
- 非一致内存访问（NUMA）模式
- 基于Cache的数据一致性，又称CC-NUMA
- 节点数可扩展到几百个，小型机多为此类架构



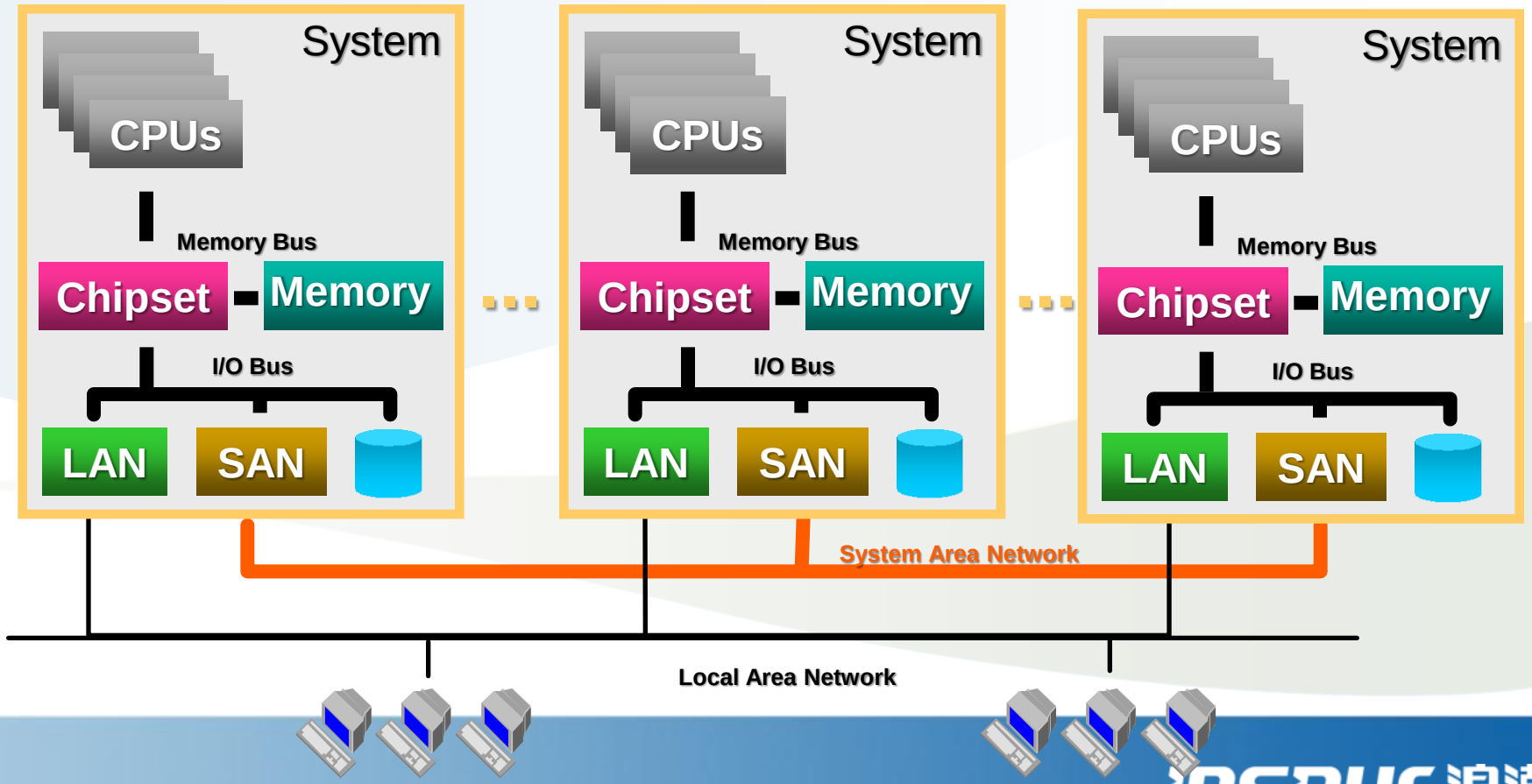
MPP- Massively Parallel Processors

- 节点个数可达成百上千个
- 节点类型可以是DM、SMP、DSM
- 节点之间采用专用高速互连设备
- 排列在Top500前面的多数系统属于这种类型



Cluster

- 独立的节点、商品化网络连接
- 开放的硬件设备、操作系统、应用编程接口
- 节点数可达几百个，性能已接近超级计算机系统
- 近年来发展很快，已广泛应用到高性能科学计算领域



各种架构对比

- SMP瓶颈在访存带宽，限制了可扩展性（性能会严重下降），通过增大Cache来缓解，又有数据一致性的问题。

- DSM具有较好的可扩展性，且具有与SMP相同的编程模式。共享存储的机器可以运行各种编程模式的程序，包括串行程序、共享存储并行程序和消息传递并行程序。

- 共享存储的机器首先可以看作是一个内存扩充了的串行机器，可以运行数据量极大的串行程序，如大数据量的数据库应用。利用多个处理器的并行处理性能，自动并行（靠编译器）作用有限，需要其他编程方式的应用，如OpenMP。

各种架构对比（续）

- 分布式存储体系结构
  - 无共享的机器，MPP和Cluster，具有最好的可扩展性，采用消息传递编程，可编程性不如OpenMP，也可以通过DSM软件的方式来模拟实现共享存储（其实还是通过消息传递，只不过对上层透明），性能受到影响，但可以运行OpenMP应用程序。
  - MPP/PVP在构造大规模系统，应用饱和性能方面具有优势，资金充足的依然会选择；
  - Cluster由于无可比拟的性价比优势占据主流位置。
    - 系统的高可用性
    - 更高的计算能力
    - 良好的可扩展性
    - 更高的性价比

并行计算机访存模型

- UMA（Uniform Memory Access）模型是均匀存储访问模型的简称
- NUMA(Nonuniform Memory Access)模型是非均匀存储访问模型
- COMA(Cache-Only Memory Access)模型是全高速缓存存储访问的简称
- CC-NUMA（Coherent-Cache Nonuniform Memory Access）模型是高速缓存一致性非均匀存储访问模型的简称
- NORMA（No-Remote Memory Access）模型是非远程存储访问模型的简称

优秀高效的并行程序—困难

- 技术先行,缺乏理论指导
- 程序的语法/语义复杂，需要用户自己处理
  - 任务/数据的划分/分配
  - 数据交换
  - 同步和互斥
  - 性能平衡
- 并行语言缺乏代可扩展和异构可扩展，程序移植困难，重写代码难度太大
- 环境和工具缺乏较长的生长期，缺乏代可扩展和异构可扩展

并行化的基本思想—任务划分

- 原则：使系统大部分时间忙于计算，而不是闲置或忙于交互；同时不牺牲并行性(度).
- 划分：切割数据和工作负载
- 分配：将划分好的数据和工作负载映射到计算结点(处理器)上
- 分配方式
  - 显式分配：由用户指定数据和负载如何加载
  - 隐式分配：由编译器和运行时支持系统决定
- 就近分配原则：进程所需的数据靠近使用它的进程代码

并行化的基石—通信

- 通信方式
  - 共享变量
  - 父进程传给子进程(参数传递方式)
  - 消息传递
- 同步：导致进程间相互等待或继续执行的操作
  - 原子同步
  - 控制同步(路障,临界区)
  - 数据同步(锁,条件临界区,监控程序,事件)
- 聚集：用一串超步将各分进程计算所得的部分结果合并为一个完整的结果，每个超步包含一个短的计算和一个简单的通信或/和同步.
  - 聚集方式:
    - 归约
    - 扫描

通信模式

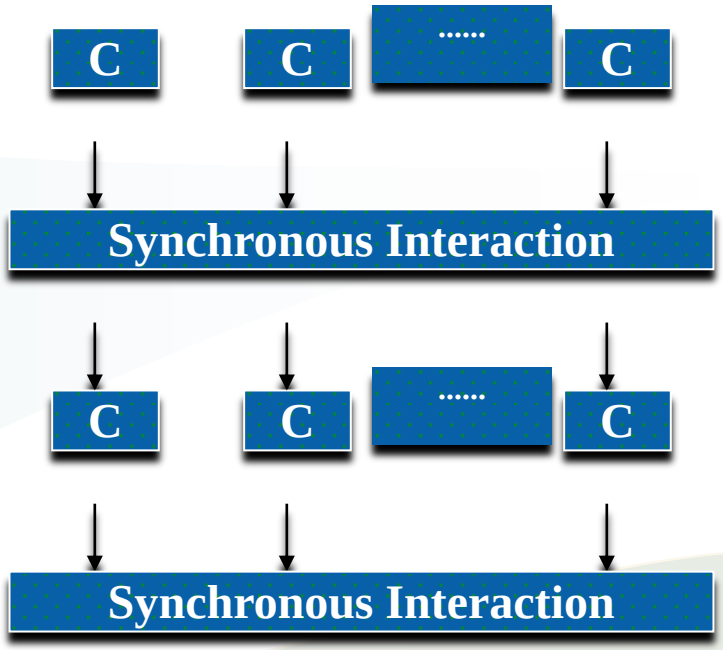
- 一对一:点到点(point to point)
- 一对多:广播(broadcast),播撒(scatter)
- 多对一:收集(gather), 归约(reduce)
- 多对多:全交换(Tatal Exchange), 扫描(scan) , 置换/移位(permutation/shift)

五种并行编程风范

- 相并行（Phase Parallel）
- 分治并行（Divide and Conquer Parallel）
- 流水线并行（Pipeline Parallel）
- 主从并行（Master-Slave Parallel）
- 工作池并行（Work Pool Parallel）

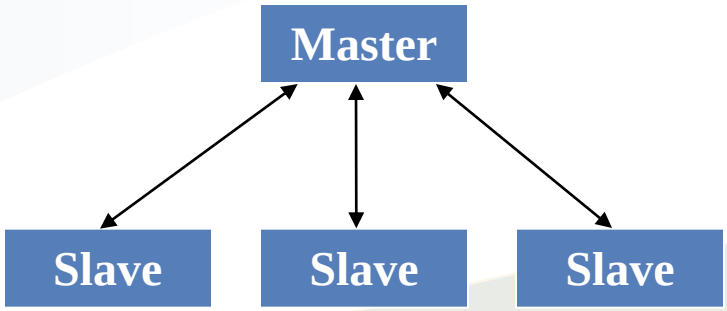
相并行（Phase Parallel）

- 一组超级步（相）
- 步内各自计算
- 步间通信、同步
- BSP
- 方便差错和性能分析
- 计算和通信不能重叠



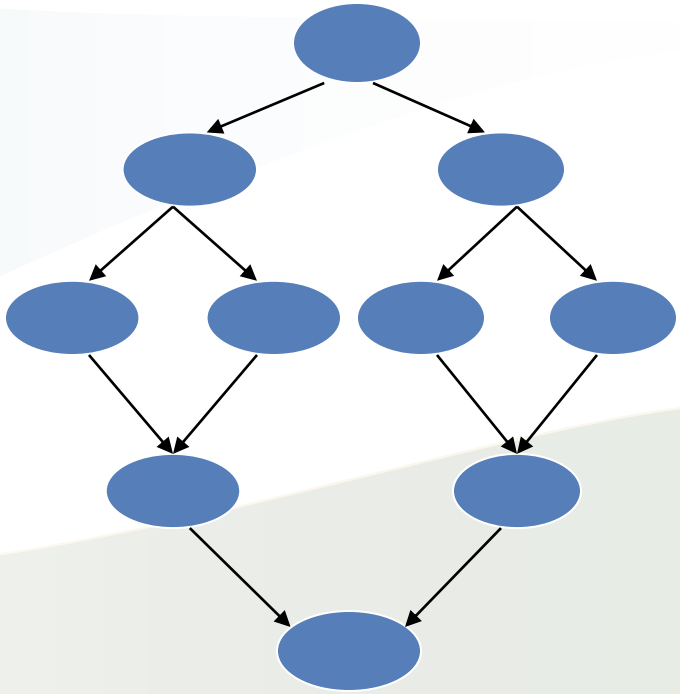
主—从并行（Master-Slave Parallel）

- 主进程：串行、协调任务
- 子进程：计算子任务
- 划分设计技术
- 与相并行结合
- 主进程易成为瓶颈



分治并行（Divide and Conquer Parallel）

- 父进程把负载分割并指派给子进程
- 递归
- 重点在于归并
- 分治设计技术
- 难以负载平衡

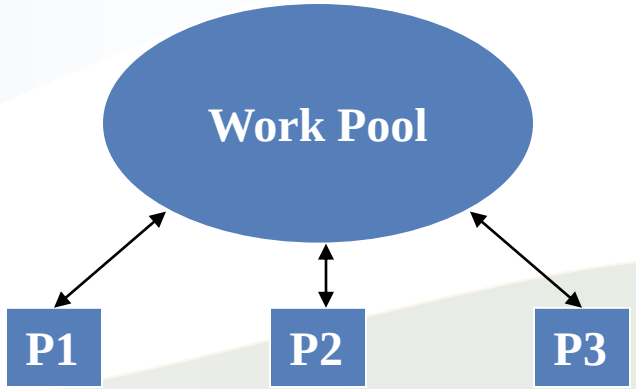


流水线并行（Pipeline Parallel）

- 一组进程
- 流水线作业
- 流水线设计技术

工作池并行（Work Pool Parallel）

- 初始状态：一件工作
- 进程从池中取任务执行
- 可产生新任务放回池中
- 直至任务池为空
- 易与负载平衡
- 临界区问题（尤其消息传递）



程序优化

级别	优化层	优化目标	关注问题	工具
1	硬件-系统层	通过查看系统资源使用情况，了解应用特点	CPU/内存，网络,磁盘IO等	系统命令，TOP,PS,IOSTAT,VMSTAT等
2	应用层	应用算法-代码级（算法优秀，考虑到硬件结构）	进程-线程锁，数据结构等	Intel Thread Profiler TBB Intel Trace collector & Analyzer
3	微架构层	依硬件特点，充分发挥系统的资源	指令集、cache大小、资源参数等	Intel Compiler, Intel MPI, MKL

程序编译

- 使用最新版的Intel Compiler，及相应的优化选项
  - 针对通用的性能优化：-O, -O0, -O1, -O2, -O3, –static, -static-intel, -fast
  - 针对向量化：-xS，-xT(Core)，-xP(NetBurst), -xSSE4.2, -xSSE4.1, -xAVX, -mtune=<processors> ……
  - 针对Cache Miss： -fno-alias
  - 高级优化选项：-ip/-ipo , -prof-gen / -prof-use
  - 串行程序并行化：-openmp, -parallel
  - 其他优化：-no-prec-div, -no-prec-sqrt, ……
- 使用最新版的Intel MKL，及其提供的常用数学库
  - 函数调用接口一致，直接连接库文件即可
  - 最大的避免了Cache Miss等性能问题
  - 最大限度使用处理器内部资源，比如SSE处理单元，64位扩展寄存器等
  - MKL提供的常用数学函数：FFT , BLAS , LAPACK , …

并行作业提交

mpiexec -genv I_MPI_DYNAMIC_CONNECTION 0-genv I_MPI_ADJUST_BCAST '4:0-380473;7:380473-4194304'-genv I_MPI_ADJUST_BARRIER '4'-genv I_MPI_ADJUST_GATHER '2:0-173;3:173-4194304'-genv I_MPI_ADJUST_ALLGATHER '1:0-384;3:384-4194304'-genv I_MPI_ADJUST_ALLGATHERV '1:0-224;3:224-4194304'-genv I_MPI_ADJUST_SCATTER '2:0-226;3:226-4194304'-genv I_MPI_ADJUST_SCATTERV 2-genv I_MPI_ADJUST_REDUCE '4:0-3925;4:3925-4160;3:4160-14004;4:14004-29771;3:29771-162610;1:162610-4194304'-genv I_MPI_ADJUST_REDUCE_SCATTER '5:0-0;4:0-23;1:23-45;3:45-4194304'-genv I_MPI_ADJUST_ALLTOALL '4:0-8;1:8-210;4:210-345;2:345-704;3:704-1588;2:1588-3503187;3:3503187-4194304'-genv I_MPI_ADJUST_ALLTOALLV 1

What is TAU?

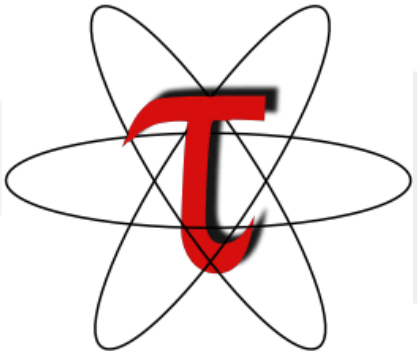
Tuning and Analysis

Department of Computer and Information Science,
University of Oregon Advanced Computing Laboratory

Los Alamos National Laboratory

NM Research Centre Juelich, ZAM, Germany

http://tau.uoregon.edu



- TAU is a performance evaluation tool
- It supports parallel profiling and tracing toolkit
- Profiling shows you how much (total) time was spent in each routine
- Tracing shows you *when* the events take place in each process along a timeline
- Profiling and tracing can measure time as well as hardware performance counters from your CPU
- TAU can automatically instrument your source code (routines,loops,I/O,memory,phases,etc.)
- TAU runs on all HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement and analysis tools
- Using TAU requires setting a couple of environment variables and substituting the name of the compiler with a TAU shell script

How to use TAU?

- Dynamic instrumentation through library pre-loading
- TAU scripted compilation
- Selectively Profiling

Method	Requires recompil-ing	Requires PDT	Shows MPI events	Routine-level event	Low level events (loops, phases, etc...)	Throttling to reduce overhead	Ability to exclude file from in-strumenta-tion
Interposi-tion			Yes			Yes	
Compiler	Yes		Yes	Yes		Yes	Yes
Source	Yes	Yes	Yes	Yes	Yes	Yes	Yes

library pre-loading

```
tau_exec –io YourPrograms
mpirun –np 4 tau_exec –io YourPrograms
```

-v	**Verbose mode**
-io	**Track I/O**
-memory	**Track memory**
-cuda	**Track GPU events via CUDA**
-opencl	**Track GPU events via OpenCL**

TAU scripted compilation

tau_f90.sh –tau_makefile=/opt/TAU/x86_64/

lib/Makefile_pdt_mpi –tau_options=-optVerbose –optDetectMemoryLeaks –optRevert –o exe yourprogram.f90

-optVerbose	Turn on verbose debugging message
-optDetectMemoryLeaks	Track mallocs/frees using TAU's memory wrapper
-optPdtGnuFortranParser	Specify the GNU gfortran PDT parser gfparse instead of f95parse
-optPdtCleanscapeParser	Specify the Cleanscape Fortran parser
-optTauSelectFile=""	Specify selective instrumentation file for tau_instrumentor
-optPreProcess	Preprocess the source code before parsing. Uses /usr/bin/cpp -P by default
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optShared	Use shared library version of TAU
-optCompInst	Use compiler-based instrumentation
-optPDTInst	Use PDT-based instrumentation

Selectively Profiling

-tau_options=-optTauSelectFile=<file>

```
#Tell tau to not profile these functions
BEGIN_EXCLUDE_LIST
void quicksort(int *, int, int)
# The next line excludes all functions beginning with "sort_" and having
# arguments "int *"
void sort_#(int *)
void interchange(int *, int *)
END_EXCLUDE_LIST
#Exclude these files from profiling
BEGIN_FILE_EXCLUDE_LIST
*.so
END_FILE_EXCLUDE_LIST
BEGIN_INSTRUMENT_SECTION
# A dynamic phase will break up the profile into phase where
# each events is recorded according to what phase of the application
# in which it occured.
dynamic phase name="foo1_bar" file="foo.c" line=26 to line=27
# instrument all the outer loops in this routine
loops file="loop_test.cpp" routine="multiply"
# tracks memory allocations/deallocations as well as potential leaks
memory file="foo.f90" routine="INIT"
# tracks the size of read, write and print statements in this routine
io file="foo.f90" routine="RINB"
END_INSTRUMENT_SECTION
```

Environment variables

TAU_VERBOSE	**决定是否要列出所有计算中间状态（过程）。1为设置该标志，0为清除该标志。**
PROFILEDIR=/dir	指定TAU输出文件文件夹位置dir
TAU_TRACK_MESSAGE	跟踪和统计MPI函数消息
TAU_COMM_MATRIX	生成MPI通信矩阵数据
TAU_THROTTLE	判断是否忽略对于一些小函数的分析，这些小函数的调用次数阈值和时间开销阈值可以用TAU_THROTTLE_NUMCALLS 和TAU_THROTTLE_PERCALL（微秒）来控制。对于调用次数大小TAU_THROTTLE_NUMCALLS，而每次执行时间却小于TAU_THROTTLE_PERCALL的函数将不予分析。
TAU_CALLPATH	判断是否需要剖析函数的调用路径。1设置该标志，0清除该标志。
TAU_CALLPATH_DEPTH	函数调用路径剖析的深度（默认值为2，可大于2）
TAU_TRACE	判断是否需要剖析事件并跟踪。1设置该标志，0清除该标志。

Profiling

- Command line

  pprof

- Visualization

  paraprof --pack YourProfile.ppk

  paraprof YourProfile.ppk

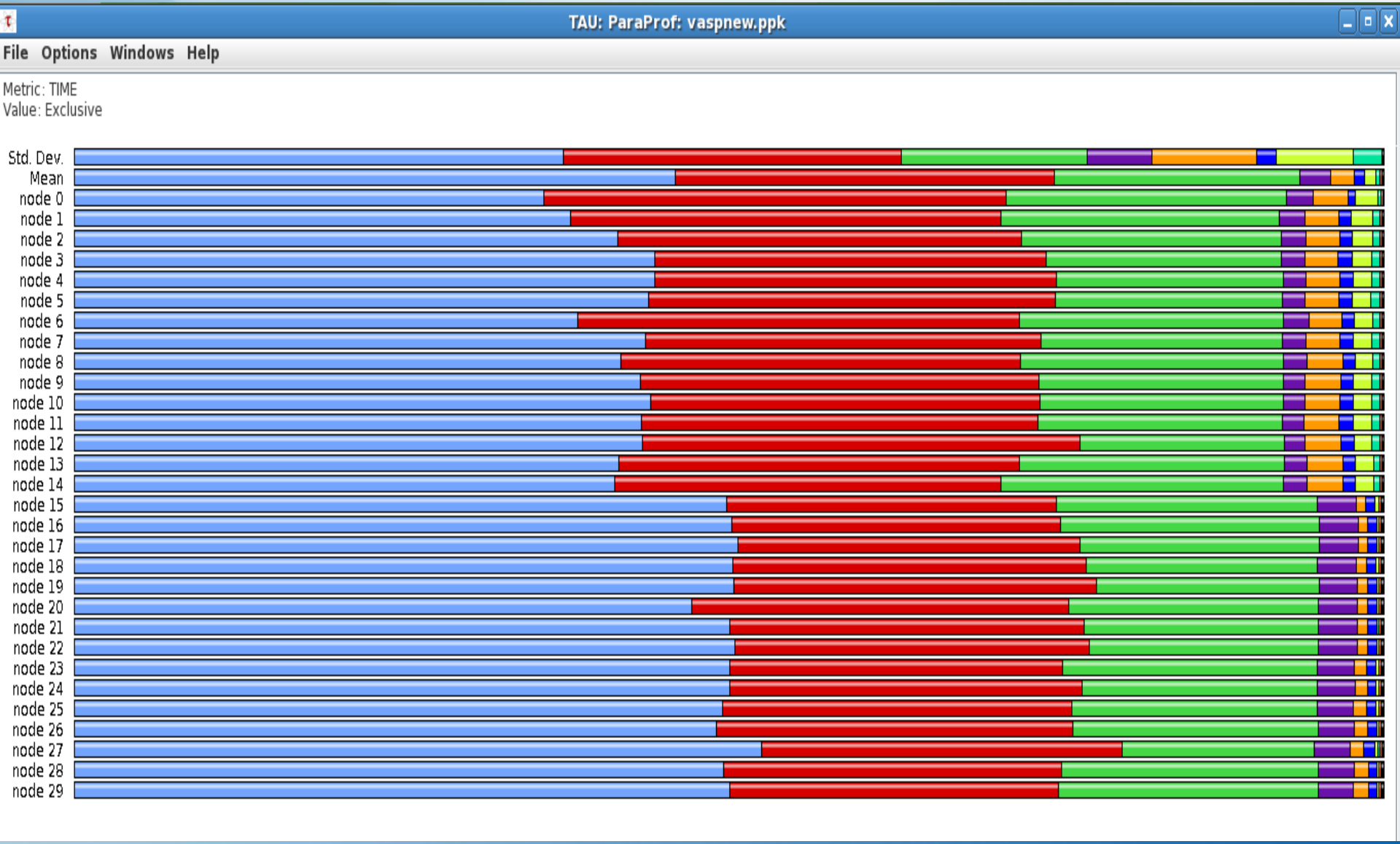
or

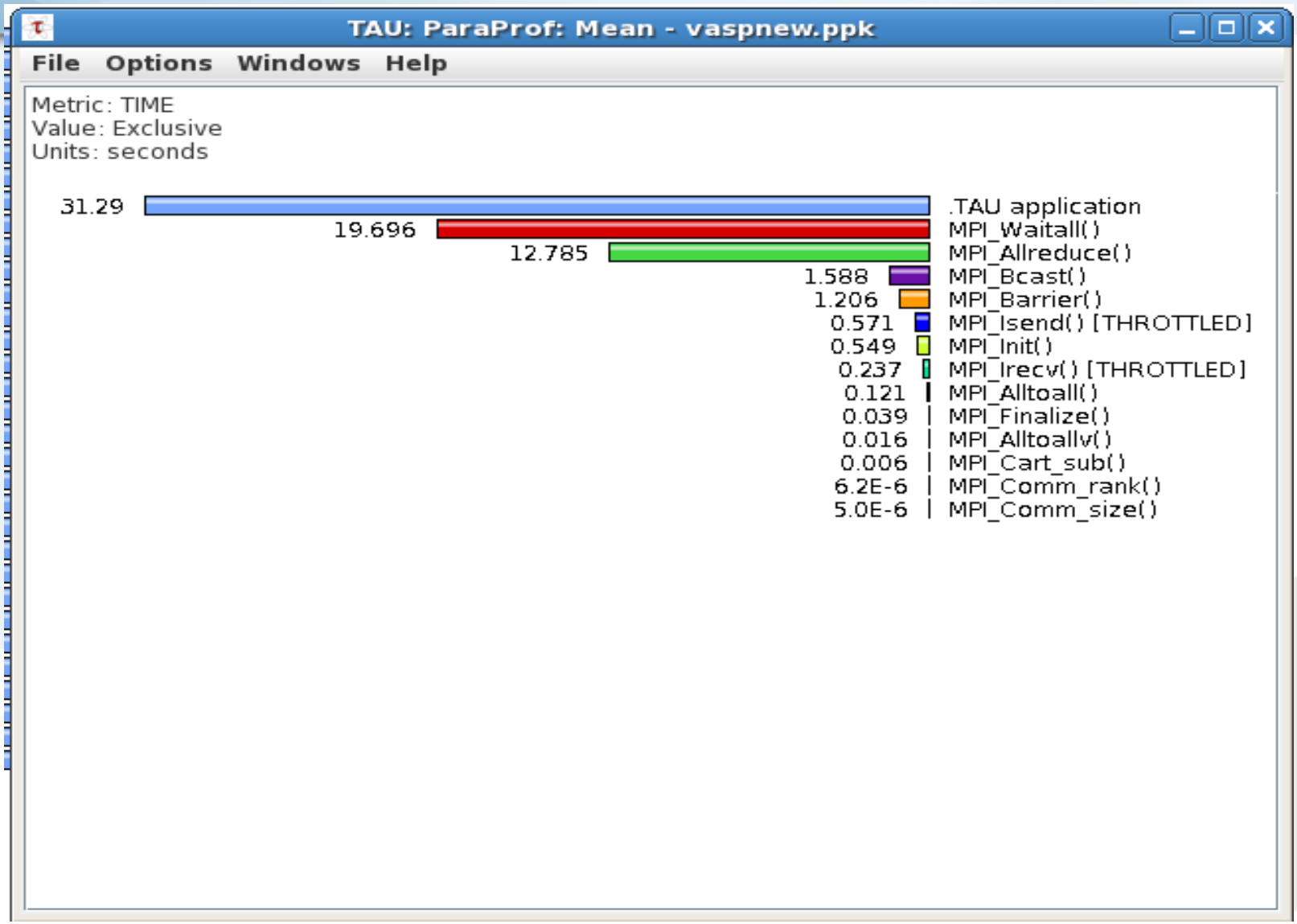
tau_merge -e events.*.edf -m events.edf tautrace.*.trc tautrace.trc

tau2slog2 tautrace.trc events.edf -o app.slog2

jumpshot app.slog2

TAU的样子

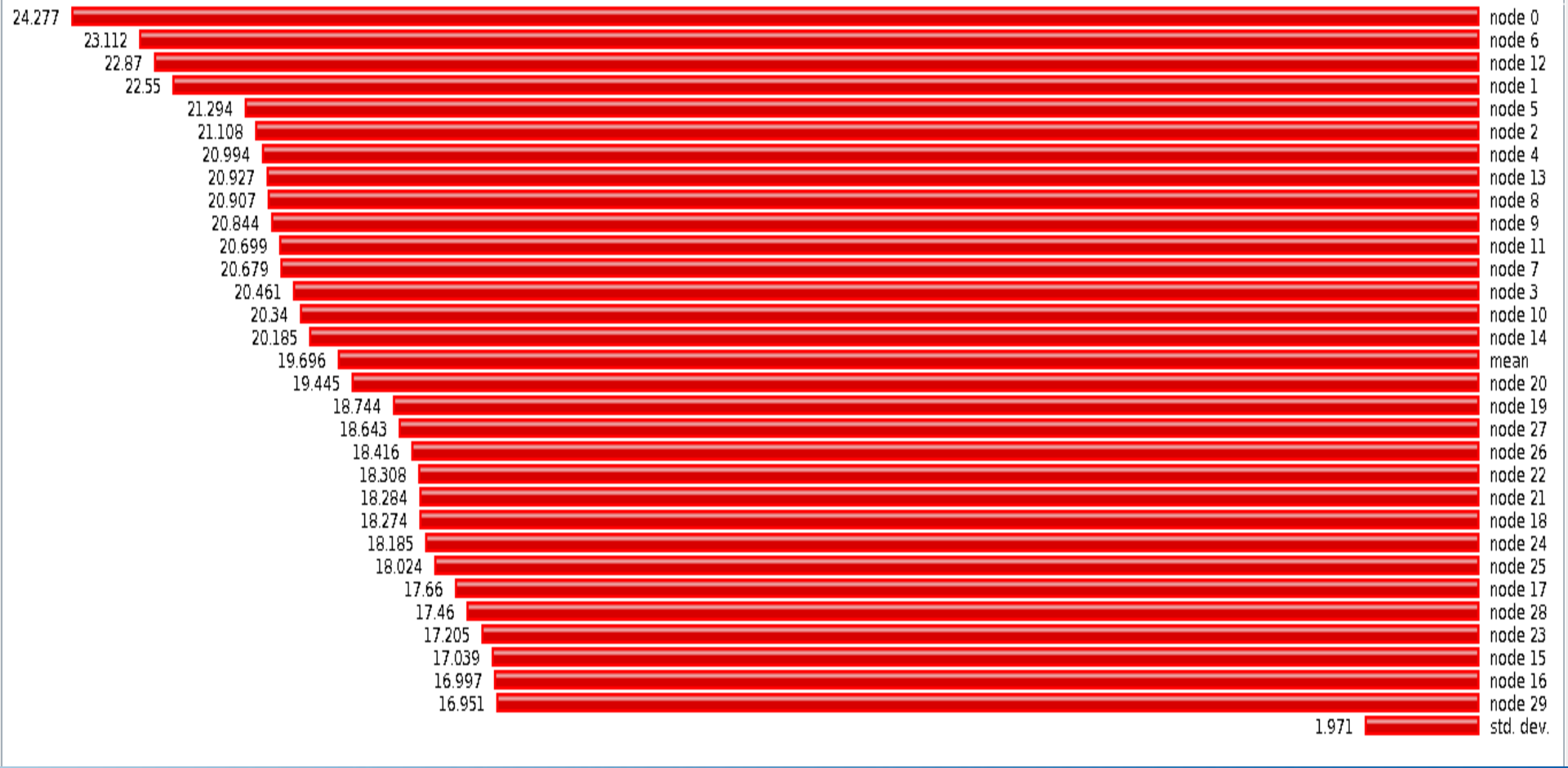

TAU: ParaProf: vaspnew.ppk
File Options Windows Help
Metric: TIME
Value: Exclusive
Std. Dev.
Mean
node 0
node 1
node 2
node 3
node 4
node 5
node 6
node 7
node 8
node 9
node 10
node 11
node 12
node 13
node 14
node 15
node 16
node 17
node 18
node 19
node 20
node 21
node 22
node 23
node 24
node 25
node 26
node 27
node 28
node 29


TAU: ParaProf: Mean - vaspnew.ppk
File Options Windows Help
Metric: TIME
Value: Exclusive
Units: seconds
31.29 .TAU application
19.696 MPI_Waitall()
12.785 MPI_Allreduce()
1.588 MPI_Bcast()
1.206 MPI_Barrier()
0.571 MPI_Isend() [THROTTLED]
0.549 MPI_Init()
0.237 MPI_Irecv() [THROTTLED]
0.121 MPI_Alltoall()
0.039 MPI_Finalize()
0.016 MPI_Alltoallv()
0.006 MPI_Cart_sub()
6.2E-6 MPI_Comm_rank()
5.0E-6 MPI_Comm_size()

TAU: ParaProf: Function Data Window: vaspnew.ppk

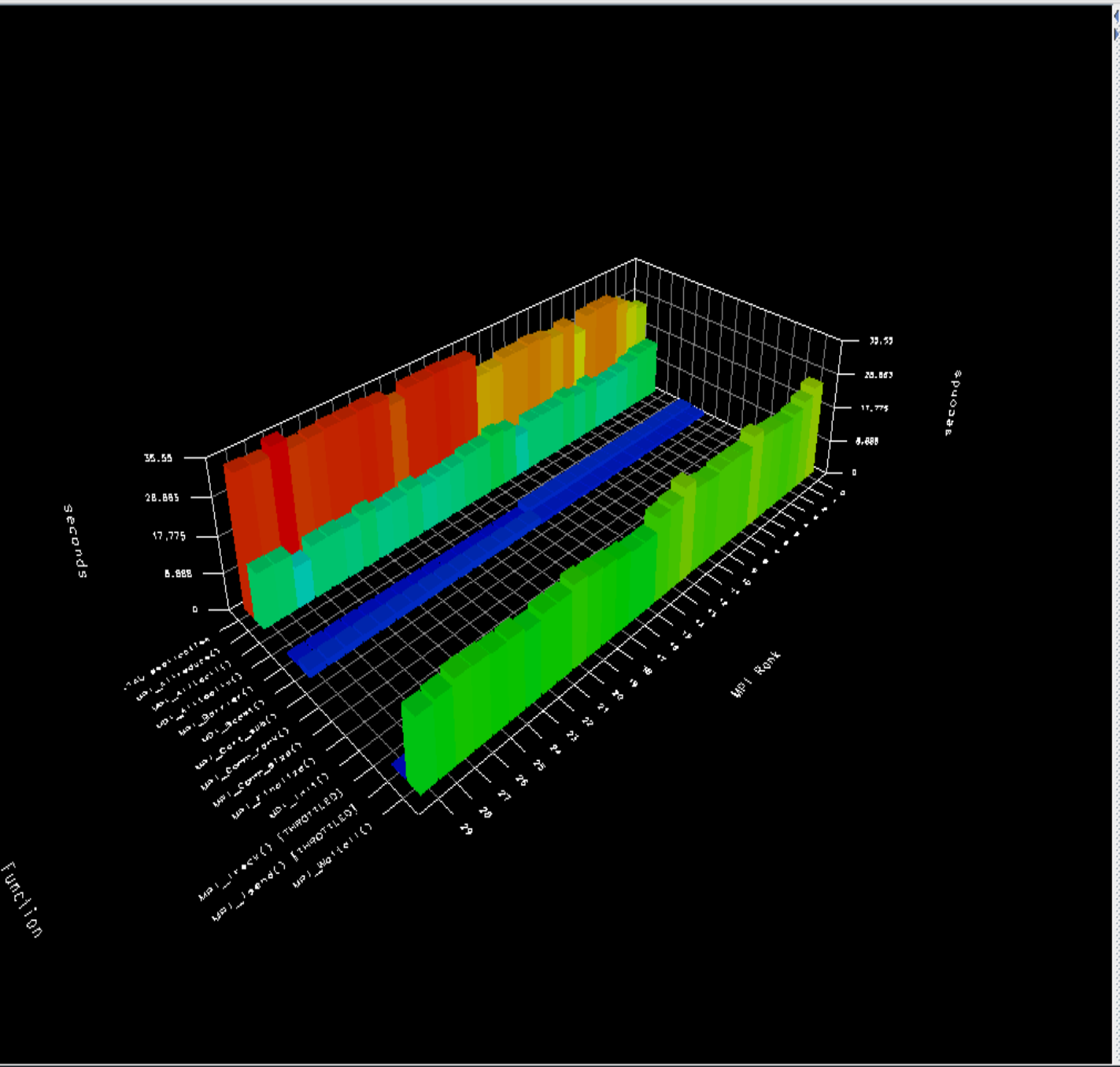
File Options Windows Help

Name: MPI_Waitall()
Metric Name: TIME
Value: Exclusive
Units: seconds



TAU: ParaProf: 3D Visualizer: vaspnew.ppk

File Options Windows Help



Triangle Mesh

Bar Plot

Scatter Plot

Topology Plot

Height Metric

Exclusive | TIME

Color Metric

Exclusive | TIME

Function <none>

Thread 0

Height value

Color value

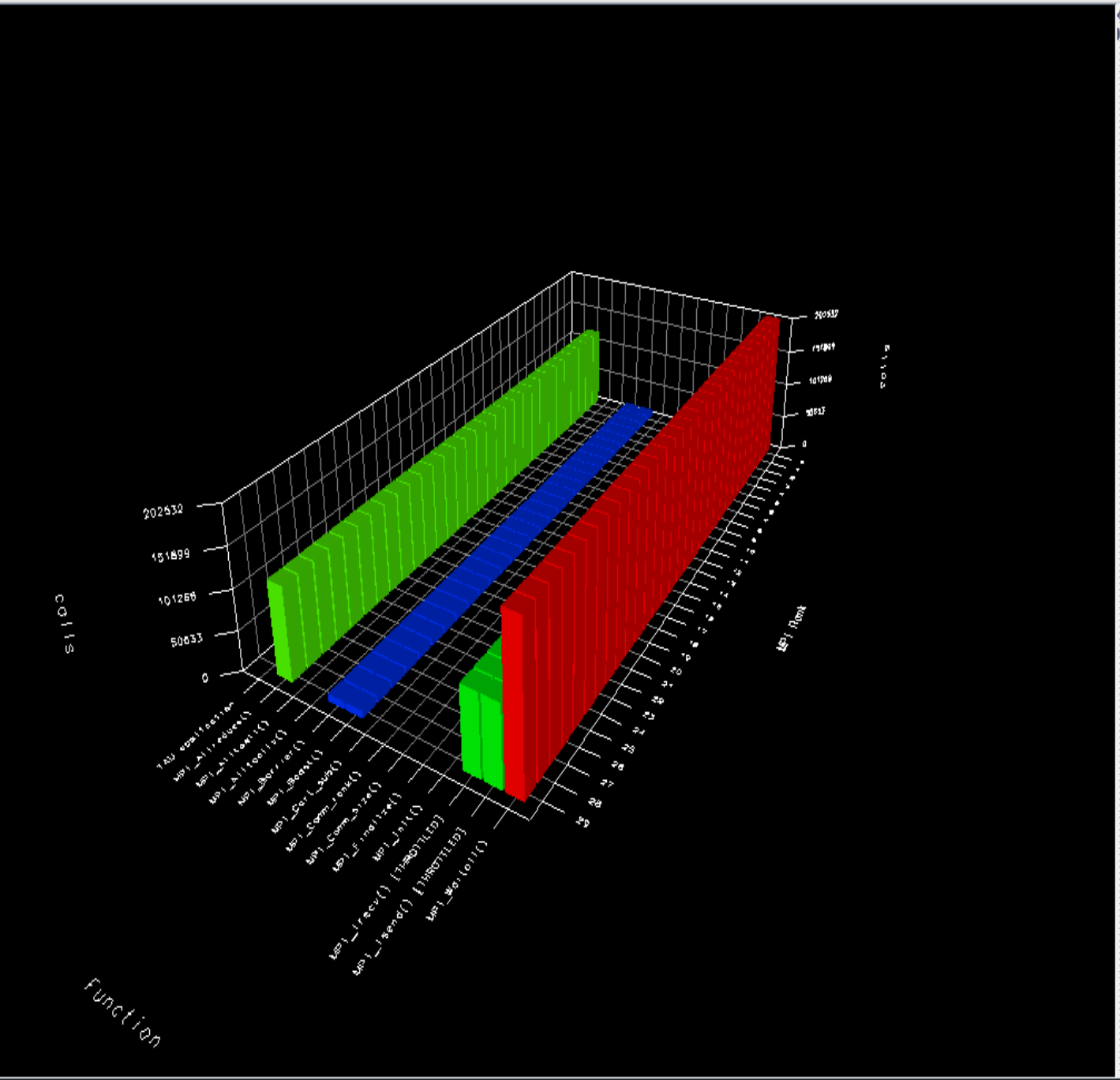
Scales | Plot | Axes | Color | Render

height: 0 35.55 seconds

color: 0 35.55 seconds

TAU: ParaProf: 3D Visualizer: vaspnew.ppk

File Options Windows Help



Triangle Mesh

Bar Plot

Scatter Plot

Topology Plot

Height Metric

Number of Calls | TIME

Color Metric

Number of Calls | TIME

Function <none>

Thread 0

Height value

Color value

Scales | Plot | Axes | Color | Render

height: 0 202532

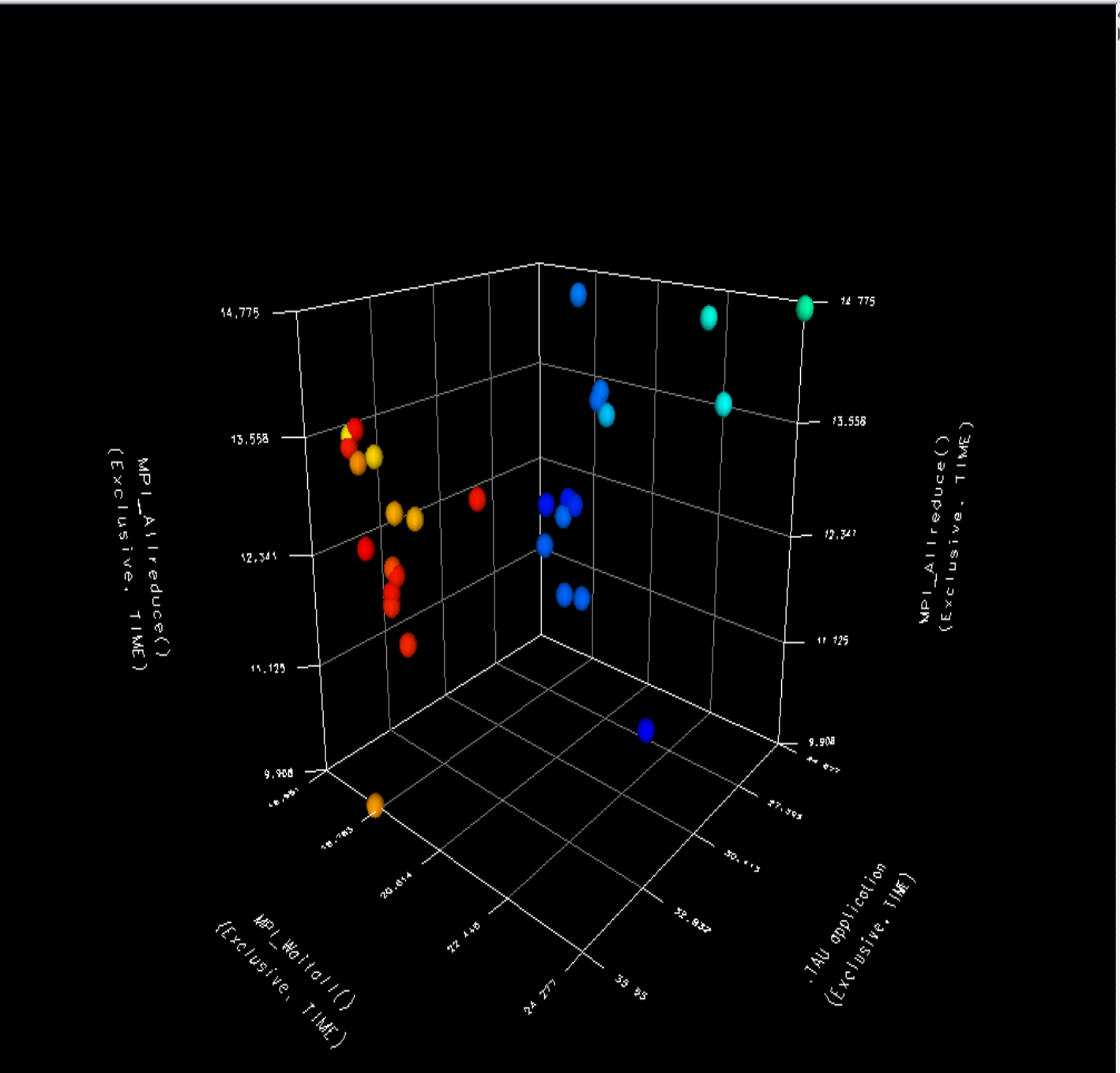
calls

color: 0 202532

calls

TAU: ParaProf: 3D Visualizer: vaspnew.ppk

File Options Windows Help



Triangle Mesh

Bar Plot

Scatter Plot

Topology Plot

Exclusive TIME

Depth: MPI Waitall() ... | Exclusive | TIME

Height: MPI Allreduce() ... | Exclusive | TIME

Color: MPI Bcast() ... | Exclusive | TIME

ColorScale | Render | ScatterPlot | Axes

2.04

Show ColorScale

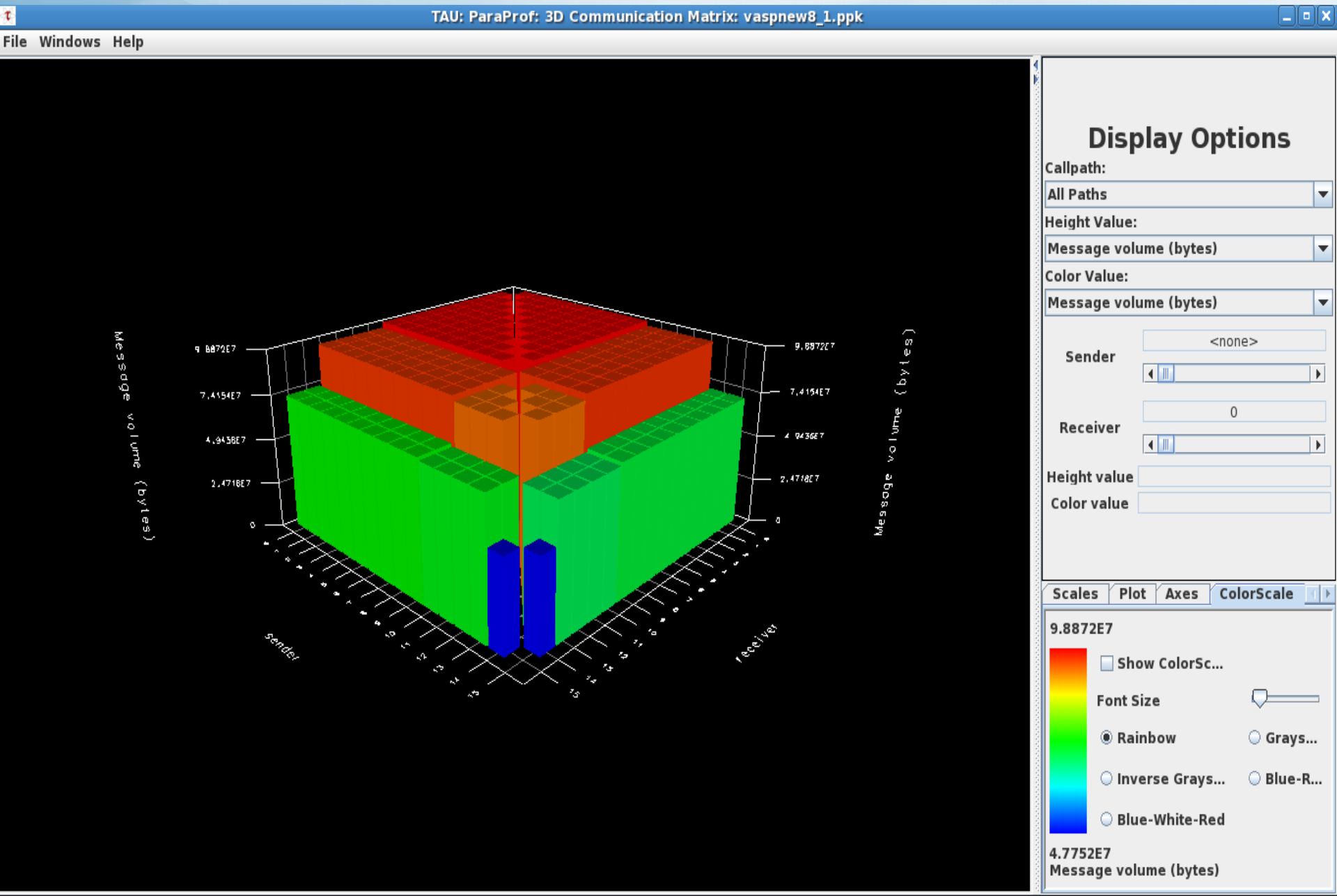
Font Size

Rainbow Graysc...

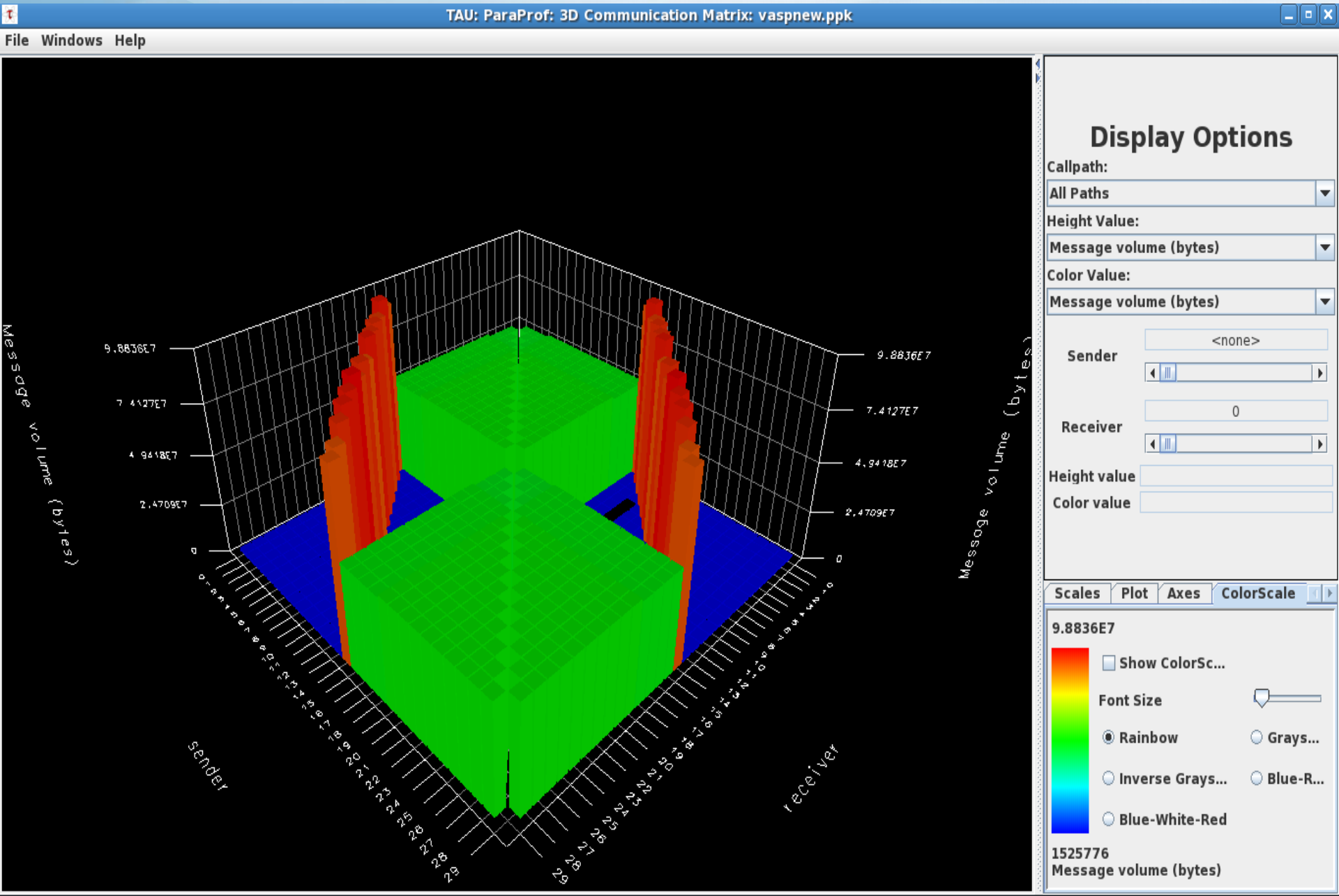
Inverse Graysc... Blue-R...

Blue-White-Red

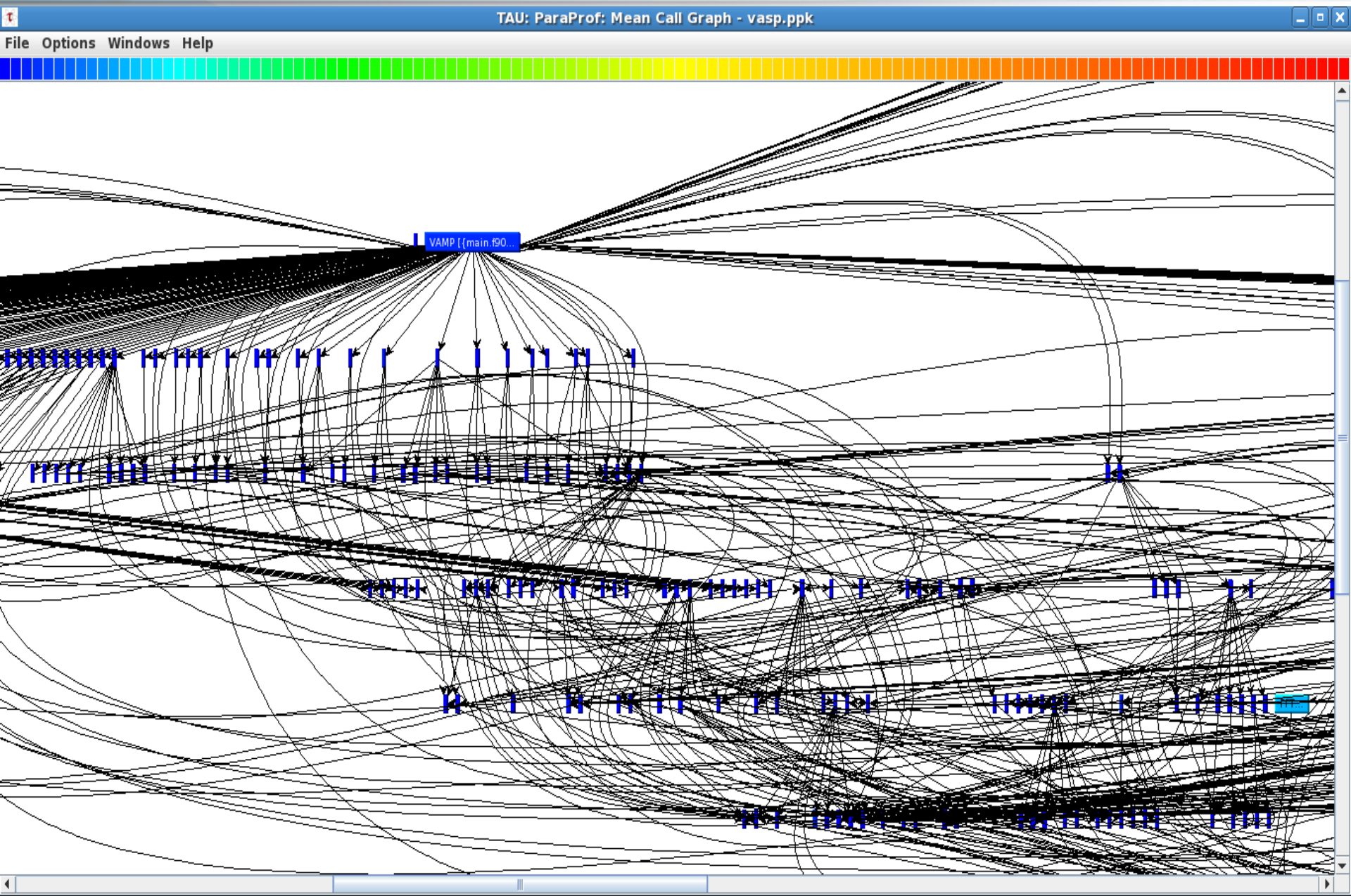
优化是与应用相关的（VASP NPAR=1）



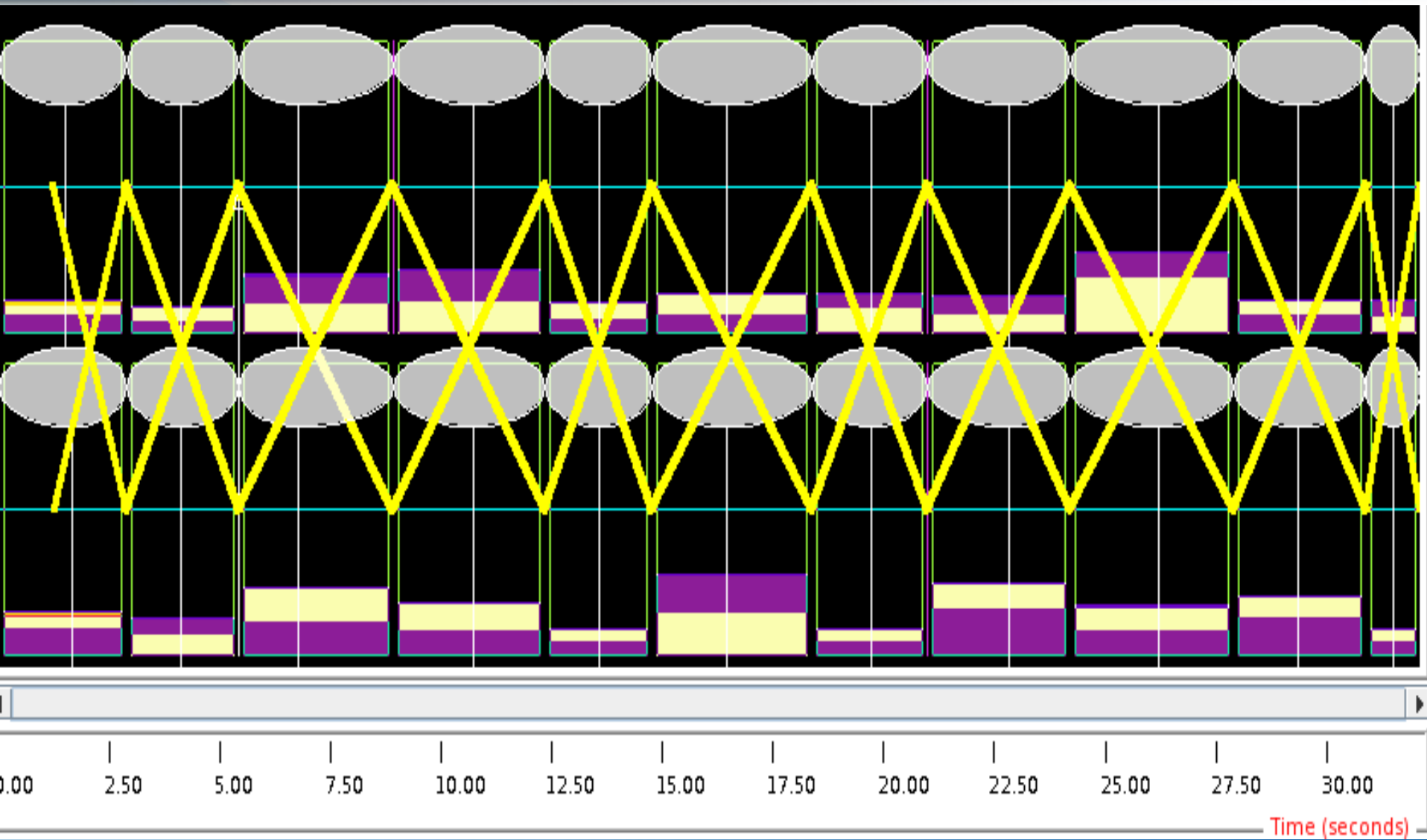
VASP NPAR=2

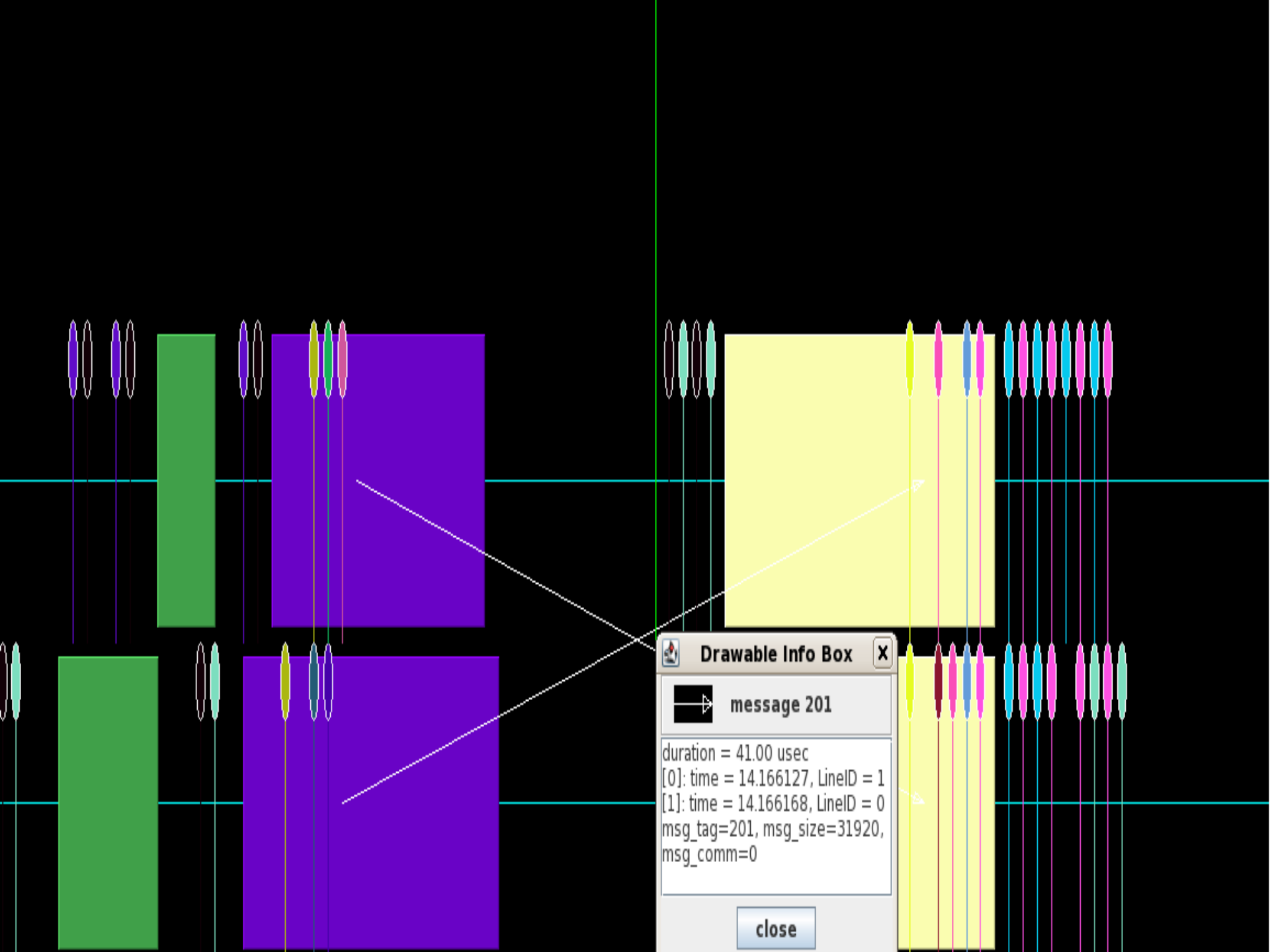

TAU: ParaProf: 3D Communication Matrix: vaspnew.ppk
File Windows Help
Message volume (bytes)
9.8836E7
7.4127E7
4.9418E7
2.4709E7
0
sender
receiver
Message volume (bytes)
Display Options
Callpath:
All Paths
Height Value:
Message volume (bytes)
Color Value:
Message volume (bytes)
Sender
<none>
Receiver
0
Height value
Color value
Scales
Plot
Axes
ColorScale
9.8836E7
Show ColorSc...
Font Size
Rainbow
Grays...
Inverse Grays...
Blue-R...
Blue-White-Red
1525776
Message volume (bytes)

函数调用


TAU: ParaProf: Mean Call Graph - vasp.ppk
File Options Windows Help
VAMP [{main.f90...

事件跟踪


0.00
2.50
5.00
7.50
10.00
12.50
15.00
17.50
20.00
22.50
25.00
27.50
30.00
Time (seconds)


Drawable Info Box
message 201
duration = 41.00 usec
[0]: time = 14.166127, LineID = 1
[1]: time = 14.166168, LineID = 0
msg_tag=201, msg_size=31920,
msg_comm=0
close

谢　　谢！

inspur 浪潮