scalasca

Scalasca 2.1 用户手册

可扩展自动性能分析

JÜLICH
FORSCHUNGSZENTRUM

German Research School
for Simulation Sciences

目录

1 引言

超级计算是现代科学与工程中的关键技术，在解决高复杂度关键问题时是必不可少的。然而，现代超级计算机中处理器核的数量一代代地增加，高性能计算应用必须利用更高的并行度来满足它们对计算能力日益增长的需要。因此，作为高效利用当今大规模计算系统的先决条件，HPC 社区需要功能强大且健壮的性能分析工具，使并行应用的优化更有效且效率更高。

Scalasca 工具集由德国 Jülich 超算中心（JSC）和德国仿真科学研究院（GRS）共同开发，是一组基于事件跟踪的性能分析工具集。这些工具专门针对大规模系统（如 IBM Blue Gene，Cray XT 等）设计，但也适用于小型 HPC 平台。目前主要关注使用 MPI[8]、OpenMP[10]以及 MPI+OpenMP 混合编程的应用，对其它并行编程模型的支持可能在将来会添加进去。Scalasca 工具集一个独具特色的特征是它的可扩展自动跟踪分析组件。这个组件提供确定等待状态的能力(比如，由于工作负载分布不均衡[5]而引起的等待状态)，特别是增大通信密集型应用的进程数时，这种等待状态将是获得高性能的严重挑战。另外，跟踪分析器能够确定目标应用关键路径上的行为[2]，突出强调那些决定程序执行长度的例程，而这些例程也是优化的最佳候选者。

与 Scalasca 工具集的早期版本相比，旧版本采用自定义测量系统和跟踪数据格式，而 Scalasca 2.x 基于社区驱动的插桩和测量基础架构 Score-P[7]。Score-P 软件由来自德国和美国的合作伙伴协作开发。通过使用通用的 CUBE4 剖析数据格式及 OTF2(Open Trace Format 2)[4] 事件跟踪数据格式，Score-P 显著地改进了 Scalasca 与 Vampir[6]以及 TAU[13]等其它性能分析工具套件之间的互操作性。 同时，2.x 版本尽可能维持对 Scalasca 1.x 的向后兼容性，例如 Scalasca 2.x 跟踪分析器仍可以处理 Scalasca 1.x 生成的跟踪测量。

本用户手册面向 Scalasca 新用户及已经熟悉 Scalasca 早期版本的老用户设计。对所有用户，我们建议首先阅读本用户手册的第二章以熟悉 Scalasca 的一般工作流程，并注意相比于 Scalasca 1.x 做出的更改。后续章节提供更有深度的单个 Scalasca 命令和工具集的参考信息，供需要时查阅。

2 开始

本章通过对一个示例程序的性能分析介绍 Scalasca 跟踪工具集的使用，涉及到 Scalasca 最主要的特性，必要时还会给出指向后续章节中相关主题更深层介绍的引用。

使用 Scalasca 跟踪工具集包括三个阶段：目标应用的插桩，执行测量采集与分析，查看分析报告。对于插桩和测量，Scalasca 2.x 使用 Score-P 基础架构，且使用 Cube 图形用户界面查看分析报告。Scalasca 使用可扩展自动跟踪分析组件、控制执行测量采集与分析的简易命令及分析报告后处理来完善 Cube 和 Score-P 的功能。

Scalasca 的大部分功能可通过 Scalasca 命令访问，这些命令提供操作选项以调用对应的底层命令：scorep、scan 及 square。这些操作如下：

- scalasca -instrument

  Scalasca 1.x 用户熟悉的 scalasca -instrument （简写形式为 skin）已被摒弃，只是为了提供向后兼容性。尽可能的将 Scalasca 1.x 插桩工具的命令行选项映射到 Score-P 插桩工具的命令 scorep 的对应选项。然而，为了充分利用 Score-P 改进后的功能，强烈建议用户直接使用 scorep 插桩工具命令。详细信息请参考 Score-P 用户手册[12]。为协助将现有的测量配置到 Score-P, Scalasca 插桩工具封装程序将转换后实际执行的 scorep 命令打印到标准输出。

- scalasca -analyze

  scalasca –analyze（简写形式为 scan）用于在目标应用程序执行过程控制 Score-P 的测量环境（支持运行时总结和/或事件跟踪采集, 可选包括硬件性能计数器信息）。当需要使用基于事件跟踪的性能分析时，测量完成后 scalasca –analyze 将自动执行 Scalasca 的跟踪分析过程。

- scalasca -examine

  scalasca -examine(简写形式为 square)用于后处理 Score-P 剖析测量和/或 Scalasca 自动事后跟踪分析工具生成的分析报，并启动分析报告查看浏览器 Cube.

调用不带参数的命令 scalasca 可获得简短的用法总结，或者使用 scalasca –quickref 命令可以打开 Scalasca 快速参考(前提是要有匹配的 PDF 阅读器)。

注意：

在底层，Scalasca 简易命令使用了 Score-P 和 Cube 提供的命令。因此，我们建议将三个软件安装路径的可执行程序所在的目录添加到 shell 环境变量 PATH 中。

在 2.4 节给出完整工作流程之前，以下三节将给出上面每个操作以及在性能分析相应阶段如何使用它们的快速概述。

2.1 插桩

为了生成用作 Scalasca 跟踪工具集输入的测量结果，首先需要对用户的应用程序进行插桩。也就是说，需要在用户程序代码中插入特殊的测量调用，然后在应用运行的特定重要点（比如事件）上执行这些插桩代码。与 Scalasca 早期版本使用自定义测量系统不同，现在这个任务由插桩和测量基础架构 Score-P 完成。

前面章节已经提到，我们不鼓励使用命令 scalasca -instrument 和 skin，因此不再详细阐述。取而代之，对用户程序、OpenMP 结构和 MPI 函数的所有必须的插桩都应由 Score-P 插桩工具处理。这些操作需要通过命令 scorep 访问，因此在性能分析的目标应用程序的编译和链接命令前需要在 Makefile 中添加 scorep 前缀。

例如，为了插桩由两个 Fortran 源文件 foo.f90 和 bar.f90 生成的 MPI 可执行程序 myapp，下面的编译和链接命令

```
% mpif90 -c foo.f90
% mpif90 -c bar.f90
% mpif90 -o myapp foo.o bar.o
```

必须替换成使用 Score-P 插桩工具的对应命令

```
% scorep mpif90 -c foo.f90
% scorep mpif90 -c bar.f90
% scorep mpif90 -o myapp foo.o bar.o
```

这将自动插桩编译器检测到的每个例程的进入和退出，拦截 MPI 函数调用以收集消息传递信息，并链接必要的 Score-P 测量库。

注意：

Scorep 插桩器必须和链接命令一起使用，以确保可执行程序链接了所有需要的 Score-P 测量库。然而，并不是所有的目标文件都需要插桩，因此应避免对此类文件中的例程和 OpenMP 结构进行测量和数据采集。对定义了 OpenMP 并行区的文件进行插桩是必要的，因为 Score-P 必须跟踪新线程的创建。

虽然默认使用的基于编译器的自动函数插桩最方便，但可能导致太多和/或破坏性太强的测量，这可通过选择性插桩及测量过滤来解决。本手册 2.4.4 节简短介绍了选择性插桩及测量过滤的最基本步骤，所有可用的插桩和过滤选项的详细介绍请参阅 Score-P 手册[12]。

2.2 运行时测量采集与分析

虽然 Score-P 插桩后编译生成的应用程序可在由环境变量定义的测量配置直接执行，在目标应用执行的过程中，Scalasca 提供的命令 scalasca –analyze（scan）可在目标应用程序的执行过程中控制 Score-P 测量环境的某些方面。为了生成插桩后执行程序的性能测量，目标应用的执行命令前需添加前缀 scalasca -analyze(或简写形式 scan)：

```
% scalasca -analyze [options] \
        [<launch_cmd> [<launch_flags>] ] <target> [target args]
```

对于纯 MPI 或 MPI+OpenMP 混合应用程序，*launch_cmd* 常常是 MPI 执行命令，比如 mpirun 或 mpiexec，*launch_flags* 是相应的未插桩程序运行命令行变量，如指定计算节点数和 MPI 进程数。对于非 MPI 应用(如串行和纯 OpenMP)，通常可以省略 *launch_cmd* 和 *launch_flags*。

以前面提到的 MPI 应用的可执行程序 myapp 为例，用四个 MPI 进程启动应用的测量命令为：

```
% scalasca -analyze mpiexec -n 4 ./myapp
```

注意：

每个测量实验使用唯一的目录保存测量数据。测量开始时这个目录不能存在，否则测量将立即中止。

实验目录的默认名称由以下几部分组成：前缀”scorep_”、目标应用可执行程序名、运行配置(比如 MPI 进程数和 OMP_NUM_THREADS)及一些其它的测量配置参数。例如，上面提到的应用 myapp 的测量将产生测量实验目录，其目录名为”scorep_myapp_4_sum”。另外，实验目录名可显式通过 scalasca -analyze 的选项 -e <experiment_name>指定或者通过环境变量 SCOREP_EXPERIMENT_DIRECTORY 指定。

注意：

关于测量配置的许多设置可通过不同的方式具体指定。首选提供给 scalasca -analyze 的命令行选项具有最高优先级，其次是 Score-P 环境变量，最后是自动确定的默认值。

测量完成后，测量实验目录包含各种日志文件和一个或多个分析报告。默认情况下，运行时总结用于提供一个总结报告，报告中包含每个进程/线程上每个调用路径的访问次数和时间消耗及硬件性能计数器度量值。对于 MPI 或 MPI+OpenMP 应用，还包括 MPI 消息统计。

也可以将采集事件跟踪数据作为性能测量的一部分。这种测量模式可通过命令 scalasca -analyze 的选项"-t"实现 （或者将环境变量 SCOREP_ENABLE_TRACING 设置为 1 ）。

注意:

使能事件跟踪采集并不会自动关闭剖析模式（即总结剖析和事件跟踪都会被同时采集)。如果不希望这样，必须通过某种方式显示禁止。

采集跟踪测量时，测量完成后实验跟踪分析将自动启动，以量化不能被剖析总结确定的等待状态。除了查看跟踪分析报告之外，生成的跟踪分析也可通过第三方图形化跟踪浏览器(比如 Vampir[6])实现可视化查看

警告:

跟踪很容易变得非常大且难处理，中间过程的跟踪缓冲区刷新可能导致结果扭曲，这会让跟踪的价值变的很小。因此，启动跟踪采集和分析之前，建立充足的测量配置至关重要。关于如何建立过滤文件及调整 Score-P 内部存储管理的详细信息，请参考本手册 2.4.4 节及 Scorep-P 用户手册[12]。

2.3 查看分析报告

运行时总结和/或自动跟踪分析的结果存储在测量实验目录下一个或多个报告中(即 CUBE4 文件)。可使用命令 scalasca -examine 对这些报告进行后处理和查看。命令需要提供一个实验目录名作为参数:

```
% scalasca -examine [options] <experiment_name>
```

在加载 Cube 分析报告浏览器之前，后处理会在实验被第一次查看时完成。如果提供给命令 scalasca -examine 的参数是已经处理过的实验目录或者 CUBE4 文件，那么浏览器将立即启动。

除了交互式查看测量分析结果外，还可以使用选项-s 得到不用加载图形浏览器的文本处理报告:

```
% scalasca -examine -s <experiment_name>
```

这份处理报告由 scorep-score 工具生成，报告提供了测量中不同类型区域的分类及其估算的相关跟踪缓冲区所需大小、总跟踪大小和最大进程跟踪缓冲区大小（max_buf）等信息，这些信息可用于设置随后跟踪测量的过滤文件及环境变量 SCOREP_TOTAL_MEMORY 的大小。详细信息请参考手册 2.4.4 节。

Cube 浏览器可直接用于实验归档文件，（打开对话框窗口然后选择其中包含的一个 CUBE4 文件），Cube 也可直接打开单个 CUBE4 文件，命令如下:

```
% cube <experiment_name>
% cube <file>.cubex
```

然而，请记住，这种情况没有进行后处理，所以只能显示一部分 Scalasca 的分析和度量值。

2.4 完整工作流程举例

前面章节基于一个抽象的例子介绍了 Scalasca 的一般工作流程，现在我们将采用一个适度复杂的 MPI 标准测试代码（NPB-MPI 3.3 BT）[9]进行分析。标准测试程序 BT 是计算流体力学模拟方面的应用，采用块三对角解法器求解非线性偏微分方程组，包含 20 个 Fortran77 代码文件。尽管 BT 没有展示显著的性能瓶颈，毕竟它是一个高度优化的标准测试程序，但它是解释整个工作流程很好的例子，包含典型的配置步骤及如何避免常见错误等。

本示例测量 (可在 Scalasca 文档网站下载[11])使用 Scalasca 联合 Score-P 1.3 和 Cube 4.2.3 在 Jülich 超算中心的 JUROPA 集群上运行。JUROPA 的计算节点配备两个 4 核 Intel Xeon X5570 处理器，CPU 主频为 2.93GHz，计算节点通过 QDR IB 胖树网络连接。代码使用 Intel 编译器编译，与基于 MPICH2 的 ParTec ParaStation MPI 库链接。下面例子中的命令对于在典型 HPC 集群环境下使用 Scalasca 具有代表性。

注意:

Scalasca 命令需要使用 Score-P 和 Cube 提供的命令。我们假定系统中这三个软件包的可执行程序所在目录位于 SHELL 的搜索路径（PATH 环境变量）中。

2.4.1 准备参考执行

执行未插桩的可执行程序作为性能参考是任何性能分析的第一步。一方面，这一步骤可证实代码执行结果正确，另一方面，可用于评估由插桩和测量引入的开销。在这个阶段，注意选择恰当的测试配置，使得程序执行可重复且足够长，从而具有代表性。(注意，过长执行时间使测量分析不方便，有时甚至不可行，所以应避免。)

解压 NPB-MPI 的源码压缩文件后，根据特定环境调整其构建系统。对于 NAS 标准测试程序而言，主要涉及到修改目录 config/subdirectory 中的 make.def 文件。该文件中定义了若干变量，这些变量在通用 Makefile 中被使用。同时，config/subdirectory 目录中还包含一个模板文件,可以拷贝并适当调整该模板文件从而形成自己需要的 Makefile。特别地，需要指定 MPI Fortran 编译器包装命令和编译选项，例如:

```
MPIF77      = mpif77
FFLAGS      = -O2
FLINKFLAGS  = -O2
```

注意，MPI C 编译器包装命令和编译选项不用于构建 BT 程序，但可用于构建对于其它 NPB 标准测试程序。

下一步，在顶层目录运行 make 命令来构建标准测试程序。在命令行中需要指定 MPI 进程数及问题规模，对于 BT 程序，进程数必须是平方数。

```
% make bt NPROCS=64 CLASS=D
   ========================================
   =      NAS Parallel Benchmarks 3.3      =
   =      MPI/F77/C                        =
   ========================================

cd BT; make NPROCS=64 CLASS=D SUBTYPE= VERSION=
make[1]: Entering directory '/tmp/NPB3.3-MPI/BT'
make[2]: Entering directory '/tmp/NPB3.3-MPI/sys'
cc -g  -o setparams setparams.c
make[2]: Leaving directory '/tmp/NPB3.3-MPI/sys'
../sys/setparams bt 64 D
make[2]: Entering directory '/tmp/NPB3.3-MPI/BT'
mpif77 -c  -O2 bt.f
mpif77 -c  -O2 make_set.f
mpif77 -c  -O2 initialize.f
mpif77 -c  -O2 exact_solution.f
mpif77 -c  -O2 exact_rhs.f
mpif77 -c  -O2 set_constants.f
mpif77 -c  -O2 adi.f
mpif77 -c  -O2 define.f
mpif77 -c  -O2 copy_faces.f
mpif77 -c  -O2 rhs.f
mpif77 -c  -O2 solve_subs.f
mpif77 -c  -O2 x_solve.f
mpif77 -c  -O2 y_solve.f
mpif77 -c  -O2 z_solve.f
mpif77 -c  -O2 add.f
mpif77 -c  -O2 error.f
mpif77 -c  -O2 verify.f
mpif77 -c  -O2 setup_mpi.f
cd ../common; mpif77 -c  -O2 print_results.f
cd ../common; mpif77 -c  -O2 timers.f
make[3]: Entering directory '/tmp/NPB3.3-MPI/BT'
mpif77 -c  -O2 btio.f
mpif77 -O2 -o ../bin/bt.D.64 bt.o make_set.o initialize.o exact_solution.o
exact_rhs.o set_constants.o adi.o define.o copy_faces.o rhs.o solve_subs.o
x_solve.o y_solve.o z_solve.o add.o error.o verify.o setup_mpi.o
../common/print_results.o ../common/timers.o btio.o
make[3]: Leaving directory '/tmp/NPB3.3-MPI/BT'
make[2]: Leaving directory '/tmp/NPB3.3-MPI/BT'
make[1]: Leaving directory '/tmp/NPB3.3-MPI/BT'
```

实际问题的有效规模（按递增顺序）包括：W, S, A, B, C, D, E。可根据具体执行环境，选择相应的问题规模，从而调整标准测试程序的运行时间。例如，规模 W 或 S 适合单核笔记本上运行 4 个 MPI 进程，其它的问题规模更适合实际系统配置。

生成的可执行程序位于目录 bin/subdirectory 中，其文件名中包含了标准测试程序的配置。对于上面例子中的命令 make，生成的可执行程序名为 bt.D.64. 现在这个二进制文件可以通过提交批处理作业的方式(超出本手册讨论范围）或者直接通过交互式会话方式进行执行。

```
% cd bin
% mpiexec -n 64 ./bt.D.64


 NAS Parallel Benchmarks 3.3 -- BT Benchmark

 No input file inputbt.data. Using compiled defaults
 Size:  408x 408x 408
 Iterations:  250    dt:   0.0000200
 Number of active processes:    64

 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Time step  220
 Time step  240
 Time step  250
 Verification being performed for class D
 accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of RMS-norms of residual
           1 0.2533188551738E+05 0.2533188551738E+05 0.1479210131727E-12
           2 0.2346393716980E+04 0.2346393716980E+04 0.8488743310506E-13
           3 0.6294554366904E+04 0.6294554366904E+04 0.3034271788588E-14
           4 0.5352565376030E+04 0.5352565376030E+04 0.8597827149538E-13
           5 0.3905864038618E+05 0.3905864038618E+05 0.6650300273080E-13
 Comparison of RMS-norms of solution error
           1 0.3100009377557E+03 0.3100009377557E+03 0.1373406191445E-12
           2 0.2424086324913E+02 0.2424086324913E+02 0.1582835864248E-12
           3 0.7782212022645E+02 0.7782212022645E+02 0.4053872777553E-13
           4 0.6835623860116E+02 0.6835623860116E+02 0.3762882153975E-13
           5 0.6065737200368E+03 0.6065737200368E+03 0.2474004739002E-13
 Verification Successful


 BT Benchmark Completed.
 Class           =                        D
 Size            =            408x 408x 408
 Iterations      =                      250
 Time in seconds =                   485.47
 Total processes =                       64
 Compiled procs  =                       64
 Mop/s total     =                 120162.50
 Mop/s/process   =                   1877.54
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              03 Apr 2014

 Compile options:
    MPIF77       = mpif77
    FLINK        = $(MPIF77)
    FMPI_LIB     = (none)
    FMPI_INC     = (none)
    FFLAGS       = -O2
    FLINKFLAGS   = -O2
    RAND         = (none)
```

```
 Please send the results of this run to:

 NPB Development Team
 Internet: npb@nas.nasa.gov

 If email is not available, send this to:

 MS T27A-1
 NASA Ames Research Center
 Moffett Field, CA  94035-1000

 Fax: 650-604-3957
```

注意，这个应用程序证实了计算正确性并报告了相应核心计算的执行时间。

2.4.2 插桩应用代码

既然参考执行已成功，是时候为初始测量使用 Score-P 准备一个插桩后的可执行程序了。 默认地，Score-P 通过编译器来自动插桩每个函数的入口和出口。当你不知道应用的详情且需要定位代码中的热点时，这往往是最好的第一种方法。对于 BT 程序，采用 Score-P 来进行插桩，只需要对 config/make.def 中的编译和链接命令添加前缀，所加前缀为 Score-P 插桩命令 scorep：

```
MPIF77   = scorep mpif77
```

注意，config/make.def 中链接器的变量 FLINK 默认值为 MPIF77，即这种情况下不需要对 FLINK 进行任何修改。

在顶层目录重新编译 BT 源代码即可创建插桩后的可执行程序，并覆盖先前编译的未插桩二进制文件(如果需要存档，可在重新编译前重命名可执行程序)：

```
% cd ..
% make clean
rm -f core
rm -f *~ */core */*~ */*.o */npbparams.h */*.obj */*.exe
rm -f MPI_dummy/test MPI_dummy/libmpi.a
rm -f sys/setparams sys/makesuite sys/setparams.h
rm -f btio.*.out*
% make bt NPROCS=64 CLASS=D
   ========================================
   =      NAS Parallel Benchmarks 3.3      =
   =      MPI/F77/C                        =
   ========================================

cd BT; make NPROCS=64 CLASS=D SUBTYPE= VERSION=
make[1]: Entering directory '/tmp/NPB3.3-MPI/BT'
make[2]: Entering directory '/tmp/NPB3.3-MPI/sys'
cc -g  -o setparams setparams.c
make[2]: Leaving directory '/tmp/NPB3.3-MPI/sys'
../sys/setparams bt 64 D
make[2]: Entering directory '/tmp/NPB3.3-MPI/BT'
scorep mpif77 -c  -O2 bt.f
```

```
scorep mpif77 -c  -O2 make_set.f
scorep mpif77 -c  -O2 initialize.f
scorep mpif77 -c  -O2 exact_solution.f
scorep mpif77 -c  -O2 exact_rhs.f
scorep mpif77 -c  -O2 set_constants.f
scorep mpif77 -c  -O2 adi.f
scorep mpif77 -c  -O2 define.f
scorep mpif77 -c  -O2 copy_faces.f
scorep mpif77 -c  -O2 rhs.f
scorep mpif77 -c  -O2 solve_subs.f
scorep mpif77 -c  -O2 x_solve.f
scorep mpif77 -c  -O2 y_solve.f
scorep mpif77 -c  -O2 z_solve.f
scorep mpif77 -c  -O2 add.f
scorep mpif77 -c  -O2 error.f
scorep mpif77 -c  -O2 verify.f
scorep mpif77 -c  -O2 setup_mpi.f
cd ../common; scorep mpif77 -c  -O2 print_results.f
cd ../common; scorep mpif77 -c  -O2 timers.f
make[3]: Entering directory '/tmp/NPB3.3-MPI/BT'
scorep mpif77 -c  -O2 btio.f
scorep mpif77 -O2 -o ../bin/bt.D.64 bt.o make_set.o initialize.o exact_solution.o
exact_rhs.o set_constants.o adi.o define.o copy_faces.o rhs.o solve_subs.o
x_solve.o y_solve.o z_solve.o add.o error.o verify.o setup_mpi.o
../common/print_results.o ../common/timers.o btio.o
make[3]: Leaving directory '/tmp/NPB3.3-MPI/BT'
make[2]: Leaving directory '/tmp/NPB3.3-MPI/BT'
make[1]: Leaving directory '/tmp/NPB3.3-MPI/BT'
```

2.4.3 初始概要测量

前面准备的插桩后可执行程序现在可在命令 scalasca –analyze（缩写为 scan）的监控下执行，完成初始概要测量：

```
% cd bin
% scalasca -analyze mpiexec -n 64 ./bt.D.64
S=C=A=N: Scalasca 2.1 runtime summarization
S=C=A=N: ./scorep_bt_64_sum experiment archive
S=C=A=N: Thu Apr  3 20:21:22 2014: Collect start
mpiexec -n 64  [...]  ./bt.D.64


 NAS Parallel Benchmarks 3.3 -- BT Benchmark

 No input file inputbt.data. Using compiled defaults
 Size:  408x 408x 408
 Iterations:  250    dt:   0.0000200
 Number of active processes:    64

 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Time step  220
 Time step  240
 Time step  250
```

```
 Verification being performed for class D
 accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of RMS-norms of residual
          1 0.2533188551738E+05 0.2533188551738E+05 0.1479210131727E-12
          2 0.2346393716980E+04 0.2346393716980E+04 0.8488743310506E-13
          3 0.6294554366904E+04 0.6294554366904E+04 0.3034271788588E-14
          4 0.5352565376030E+04 0.5352565376030E+04 0.8597827149538E-13
          5 0.3905864038618E+05 0.3905864038618E+05 0.6650300273080E-13
 Comparison of RMS-norms of solution error
          1 0.3100009377557E+03 0.3100009377557E+03 0.1373406191445E-12
          2 0.2424086324913E+02 0.2424086324913E+02 0.1582835864248E-12
          3 0.7782212022645E+02 0.7782212022645E+02 0.4053872777553E-13
          4 0.6835623860116E+02 0.6835623860116E+02 0.3762882153975E-13
          5 0.6065737200368E+03 0.6065737200368E+03 0.2474004739002E-13
 Verification Successful


 BT Benchmark Completed.
 Class           =                        D
 Size            =            408x 408x 408
 Iterations      =                      250
 Time in seconds =                   947.46
 Total processes =                       64
 Compiled procs  =                       64
 Mop/s total     =                 61570.85
 Mop/s/process   =                   962.04
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              03 Apr 2014

 Compile options:
    MPIF77       = scorep mpif77
    FLINK        = $(MPIF77)
    FMPI_LIB     = (none)
    FMPI_INC     = (none)
    FFLAGS       = -O2
    FLINKFLAGS   = -O2
    RAND         = (none)

 Please send the results of this run to:

 NPB Development Team
 Internet: npb@nas.nasa.gov

 If email is not available, send this to:

 MS T27A-1
 NASA Ames Research Center
 Moffett Field, CA  94035-1000

 Fax: 650-604-3957

S=C=A=N: Thu Apr  3 20:37:18 2014: Collect done (status=0) 956s
S=C=A=N: ./scorep_bt_64_sum complete.

% ls scorep_bt_64_sum
profile.cubex  scorep.cfg  scorep.log
```

可以看到，测量运行成功，生成实验目录 scorep_bt_64_sum，该目录下包含如下文件：

- profile.cubex：运行时概要结果文件
- score.cfg：包含测量配置拷贝的文件
- score.log：测量日志文件

然而，插桩后应用执行时间大约是（未插桩）参考应用执行时间的两倍(947.46s VS 485.47s)。也就是说，插桩和相关的测量引入了不可忽略的时间开销。虽然也可以使用 Cube 报告浏览器交互式查看生成的概要结果文件，但需非常谨慎，因为巨大的开销给测量精度带来了负面影响。

2.4.4 优化测量配置

过度的测量开销将导致性能数据出现偏差。为避免基于偏差性能数据得出错误结论，往往需要在进行后续实验之前优化测量配置。可通过多种不同的方式优化测量配置，比如使用运行时过滤，选择性记录或者人工插桩控制测量。请参考 Score-P 手册[12]以获取更多关于相关选项的细节信息。然而，在多数情况下，只需要过滤掉少量“频繁执行且不重要的用户函数”即可将测量开销降到可接受水平。选择这些例程时需要格外小心，因为它将影响性能测量的粒度。过于粗粒度的过滤可能不利于定位重要热点的位置。

为了确定运行时过滤的候选函数，可使用命令 scalasca -examine 的-s 选项来处理初始实验的概要报告：

```
% scalasca -examine -s scorep_bt_64_sum
INFO: Post-processing runtime summarization report...
scorep-score -r ./scorep_bt_64_sum/profile.cubex > ./scorep_bt_64_sum/scorep.score
INFO: Score report written to ./scorep_bt_64_sum/scorep.score

% head -n 20 scorep_bt_64_sum/scorep.score

Estimated aggregate size of event trace:                   3700GB
Estimated requirements for largest trace buffer (max_buf): 58GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       58GB
(hint: When tracing set SCOREP_TOTAL_MEMORY=58GB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt type     max_buf[B]          visits  time[s] time[%] time/     region
                                                         visit[us]
     ALL 62,076,748,138 152,783,214,921 61176.89   100.0      0.40 ALL
     USR 62,073,899,966 152,778,875,273 57811.61    94.5      0.38 USR
     MPI      2,267,202       2,909,568  3087.68     5.0   1061.22 MPI
     COM        580,970       1,430,080   277.60     0.5    194.12 COM

     USR 20,525,692,668  50,517,453,756  7019.03    11.5      0.14 matvec_sub_
     USR 20,525,692,668  50,517,453,756  7800.42    12.8      0.15 matmul_sub_
     USR 20,525,692,668  50,517,453,756 12256.11    20.0      0.24 binvcrhs_
     USR    447,119,556   1,100,528,112   124.29     0.2      0.11 exact_solution_
     USR     50,922,378     124,121,508    18.23     0.0      0.15 binvrhs_
     MPI        855,834         771,456    10.98     0.0     14.23 MPI_Isend
     MPI        855,834         771,456     5.06     0.0      6.55 MPI_Irecv
```

从处理输出的顶部可看到，不使用任何过滤的事件跟踪测量时，其预估的事件跟踪的大小接近 3.7TiB，所有进程中最大事件跟踪大约是 62GB(~58GB)。考虑到 JUROPA 的计算节点可用主内存为 24GiB 并且每节点运行 8 个 MPI 进程，如果同时希望避免刷新中间过程的临时跟踪缓冲区，使用这种配置进行跟踪实验显然是不可行。

处理输出的下一部分给出一个表，这个表显示了单进程对跟踪内存的需求（max_buf 列），同时显示了某个函数组总的访问次数和 CPU 时间分配情况。目前，主要包括以下函数组：

- MPI： MPI API 函数
- OMP：OpenMP 结构和 API 函数
- COM：出现在通向 OpenMP 结构或 OpenMP/MPI API 函数的调用路径上的用户函数/区域，用于提供 MPI/OpenMP 使用的上下文。
- USR： 未出现在通向 OpenMP 结构或 OpenMP/MPI API 函数的调用路径上的用户函数/区域。

对于概要报告中的每个区域，根据函数组（type 列）对其进行了进一步的细分。通过对这部分处理报告的详细研究，可以看出绝大部分跟踪数据由对 USR 组内 matvec_sub, matmul_sub 和 binvcrhs 三个例程超过 500 亿次的调用产生。从消耗的时间百分比来看，这些例程似乎很重要。但是平均每次访问所消耗的时间小于 250 纳秒（time/visit 列）。也就是说，这些函数的相对测量时间开销很大，因此大量时间其实是花在 Score-P 测量系统，而不是应用本身。因此，这些例程是最佳的过滤候选者（就像它们是编译器内联的最佳候选者一样）。另外，例程 exact_solution 在很短运行时间内单进程上产生 447MB 事件数据，也是一个很好的过滤对象。因此一个合理的 Score-P 过滤文件如下：

```
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    binvcrhs_
    matvec_sub_
    matmul_sub_
   exact_solution_
SCOREP_REGION_NAMES_END
```

关于过滤文件格式、如何基于文件名过滤、定义黑名单和白名单及为了方便怎样使用通配符的详细描述，请参考 Score-P 手册[12]。

过滤器的有效性可通过再次处理初始概要报告来检验。这次使用命令 scalasca -examine 的选项-f 指定过滤器文件。采用这种方式，可以递增式渐进开发过滤器文件, 避免为了验证单个函数的过滤效果而进行多次性能测量。

```
% scalasca -examine -s -f npb-bt.filt scorep_bt_64_sum
scorep-score -f npb-bt.filt -r ./scorep_bt_64_sum/profile.cubex \
             > ./scorep_bt_64_sum/scorep.score_npb-bt.filt
INFO: Score report written to ./scorep_bt_64_sum/scorep.score_npb-bt.filt

% head -n 25 scorep_bt_64_sum/scorep.score_npb-bt.filt

Estimated aggregate size of event trace:                   3298MB
Estimated requirements for largest trace buffer (max_buf): 53MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       57MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=57MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)
```

```
flt type     max_buf[B]          visits  time[s] time[%] time/     region
                                                         visit[us]
 -   ALL 62,076,748,138 152,783,214,921 61176.89   100.0      0.40 ALL
 -   USR 62,073,899,966 152,778,875,273 57811.61    94.5      0.38 USR
 -   MPI      2,267,202       2,909,568  3087.68     5.0   1061.22 MPI
 -   COM        580,970       1,430,080   277.60     0.5    194.12 COM

 *   ALL     54,527,956     130,325,541 33977.03    55.5    260.71 ALL-FLT
 +   FLT 62,024,197,560 152,652,889,380 27199.86    44.5      0.18 FLT
 *   USR     51,679,784     125,985,893 30611.75    50.0    242.98 USR-FLT
 -   MPI      2,267,202       2,909,568  3087.68     5.0   1061.22 MPI-FLT
 *   COM        580,970       1,430,080   277.60     0.5    194.12 COM-FLT
 +   USR 20,525,692,668  50,517,453,756  7019.03    11.5      0.14 matvec_sub_
 +   USR 20,525,692,668  50,517,453,756  7800.42    12.8      0.15 matmul_sub_
 +   USR 20,525,692,668  50,517,453,756 12256.11    20.0      0.24 binvcrhs_
 +   USR    447,119,556   1,100,528,112   124.29     0.2      0.11 exact_solution_
 -   USR     50,922,378     124,121,508    18.23     0.0      0.15 binvrhs_
 -   MPI        855,834         771,456    10.98     0.0     14.23 MPI_Isend
```

在原始函数组概要下方，处理报告同时给出了另一个使用过滤器后的性能概要。这个新增的概要报告里，额外增加了一个 FLT 组，该 FLT 组包含所有被过滤的区域。而且，报告的 flt 列使用加号（“+”）表示一个区域/函数组被过滤，减号（“-”）表示未过滤，星号（“*”）表示函数组被部分过滤。

正如预期，总的事件跟踪大小的估计降至 3.3GiB，且所有进程中局部进程最大值减小到 53MiB。考虑到 Score-P 测量系统创建了大量的内部数据结构（例如，用于跟踪 MPI 请求和通信子），当使能事件跟踪分析时，用于调整 Score-P 内存管理所使用的最大内存大小的环境变量 SCOREP_TOTAL_MOMERY 建议设置为 57MiB。

2.4.5 概要测量与查看

现在可将 2.4.4 节准备的过滤文件用于生成新概要测量，并通过减小测量开销提高性能测量的准确性。可通过-f 选项向命令 scalasca -analyze 提供过滤文件。

注意：

重新分析应用程序前，需要重命名(或删除)先前的未过滤实验数据，因为 scalasca -analyze 不会覆盖现存实验目录，相反会因为先前同名实验目录的存在而立即中止当前性能实验。

```
% mv scorep_bt_64_sum scorep_bt_64_sum.nofilt
% scalasca -analyze -f npb-bt.filt mpiexec -n 64 ./bt.D.64
S=C=A=N: Scalasca 2.1 runtime summarization
S=C=A=N: ./scorep_bt_64_sum experiment archive
S=C=A=N: Thu Apr  3 20:37:18 2014: Collect start
mpiexec -n 64  [...]  ./bt.D.64


 NAS Parallel Benchmarks 3.3 -- BT Benchmark

 No input file inputbt.data. Using compiled defaults
 Size:  408x 408x 408
 Iterations:  250    dt:   0.0000200
 Number of active processes:    64

 Time step    1
 Time step   20
 Time step   40
 Time step   60
```

```
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Time step  220
 Time step  240
 Time step  250
 Verification being performed for class D
 accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of RMS-norms of residual
           1 0.2533188551738E+05 0.2533188551738E+05 0.1479210131727E-12
           2 0.2346393716980E+04 0.2346393716980E+04 0.8488743310506E-13
           3 0.6294554366904E+04 0.6294554366904E+04 0.3034271788588E-14
           4 0.5352565376030E+04 0.5352565376030E+04 0.8597827149538E-13
           5 0.3905864038618E+05 0.3905864038618E+05 0.6650300273080E-13
 Comparison of RMS-norms of solution error
           1 0.3100009377557E+03 0.3100009377557E+03 0.1373406191445E-12
           2 0.2424086324913E+02 0.2424086324913E+02 0.1582835864248E-12
           3 0.7782212022645E+02 0.7782212022645E+02 0.4053872777553E-13
           4 0.6835623860116E+02 0.6835623860116E+02 0.3762882153975E-13
           5 0.6065737200368E+03 0.6065737200368E+03 0.2474004739002E-13
 Verification Successful


 BT Benchmark Completed.
 Class           =                        D
 Size            =            408x 408x 408
 Iterations      =                      250
 Time in seconds =                   495.32
 Total processes =                       64
 Compiled procs  =                       64
 Mop/s total     =                117773.61
 Mop/s/process   =                  1840.21
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              03 Apr 2014

 Compile options:
    MPIF77       = scorep mpif77
    FLINK        = $(MPIF77)
    FMPI_LIB     = (none)
    FMPI_INC     = (none)
    FFLAGS       = -O2
    FLINKFLAGS   = -O2
    RAND         = (none)


 Please send the results of this run to:

 NPB Development Team
 Internet: npb@nas.nasa.gov

 If email is not available, send this to:

 MS T27A-1
 NASA Ames Research Center
 Moffett Field, CA  94035-1000

 Fax: 650-604-3957


S=C=A=N: Thu Apr  3 20:45:38 2014: Collect done (status=0) 500s
S=C=A=N: ./scorep_bt_64_sum complete.
```

注意到，通过使用运行时过滤器，显著降低了性能测量开销，新的测量开销降低到 2%(过滤运行的 495.32 秒 VS 参考运行为 485.47 秒)。因此，这个使用优化后配置的新测量精确地反应了 BT 程序的实际运行时行为，可对其进行后处理并使用 Cube 浏览器交互式分析查看。这两个任务可方便地通过命令 scalasca -examine 启动：

```
% scalasca -examine scorep_bt_64_sum
INFO: Post-processing runtime summarization report...
INFO: Displaying ./scorep_bt_64_sum/summary.cubex...
```

CUBE 概要分析(见图 2.1 中屏幕截图及 2.4.5.1 节中如何使用 Cube 浏览器的简要概述)显示，总 CPU 时间的 97%花在执行计算任务的用户函数，只有 2.7%的时间花在 MPI 点到点通信函数及剩余分散的其它活动上面。点到点通信时间几乎全部花在三个解法器函数(x_solve, y_solve, z_solve)内部调用的 MPI_Wait 以及边界交换例程 copy_faces 调用的 MPI_Waitall 上。虽然执行时间也大部分花在解法器例程和边界交换函数上，然而是在三个不同的函数 solve_cell，backsubstitute 和 compute_rhs 内。虽然不同 MPI 进程花在计算例程上的总时间相对均衡(请参见右侧面板中 box plot 视图)，但是对 MPI_Wait/MPI_Waitall 调用却存在较大变化。

2.4.5.1 使用 Cube 浏览器

后续段落提供使用 Cube 分析报告浏览器的简要介绍。为了有效使用 Cube 图形用户界面，请参考 Cube 用户手册[3]。

Cube 是用于展示和浏览并行应用程序性能和调试信息的用户界面。底层数据模型独立于所展示的特定性能属性。Cube 主窗口(图 2.1)由三个面板组成，每个面板提供分析报告的树形显示或可选的图形视图。左侧面板显示程序执行的性能属性，比如时间或访问次数等。中间面板显示应用的调用树或平面剖析。右侧面板可以显示系统层级组成，例如机器、计算节点、进程、线程、应用程序的进程和线程的拓扑视图等，也可提供 box plot 视图显示某个数值在系统范围内的统计分布。所有树节点标有度量值以及有助于识别热点的着色编码方框。度量值的颜色取决于所占总值或其它给定参考值的比例。窗口底部的彩色条给出了色彩与比例之间的大致映射关系。

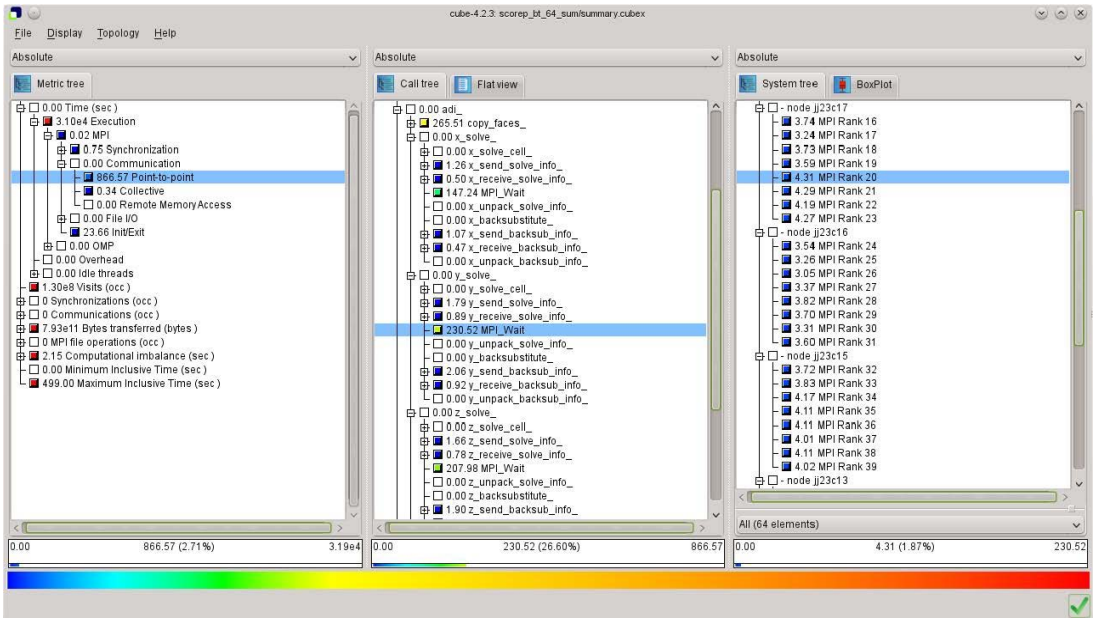

图 2.1 使用 Cube 报告浏览器查看概要分析数据

鼠标点击某个性能属性或调用路径将选中相应节点，并在该节点右侧面板中显示该度量值在其内部各组成部分之间的细分。例如，选择某个性能属性后，中间面板将显示该性能属性基于调用树的分布。选择某个调用路径 (即调用树上的一个节点) 后，系统树显示该系统范围内性能属性在该调用路径上的分布情况。通过展开或折叠三种树（度量树，调用树以及系统树）上的节点，可从不同粒度层次上查看分析结果。

所有树形显示支持上下文菜单，鼠标右键可访问该上下文菜单并提供更多选项。例如，为获得某个性能属性的精确定义，在与每个性能属性相关联的上下文菜单中选择“Online Description”选项，也可通过菜单选项“Info”获得简短描述。

2.4.6 跟踪收集与分析

概要剖析仅仅提供进程局部或线程局部时间总和数据，而事件跟踪包含详细的带时间戳标记的事件数据，通过这些数据能够重建应用的动态行为，也使得比如 Scalasca 跟踪分析器这样的工具能够提供更多关于应用程序性能行为的细节分析，例如进程花费在 MPI 通信上的时间是否真的是消息处理时间，或是引起显著的等待状态（即一个进程由于等待来自其它进程的数据而处于空闲状态的时间间隔）。

可使用命令 scalasca -analyze 的选项-t 使能 Scalasca 跟踪分析器的跟踪分析及随后的自动分析功能。由于该选项同时使能了跟踪采集与概要测量，所以通常与选项-q 连用。（-q 选项可以同时关闭跟踪测量和概要测量。）（注意， -q 与-t 两个选项指定的先后次序是有影响的。）

注意：

不要忘记指定恰当的测量配置（即相应的过滤文件和设置环境变量SCOREP_TOTAL_MEMORY）。否则，大量的跟踪数据很容易填满你的磁盘，同时性能测量容易遭受由于刷新临时跟踪缓冲区而导致的干扰，这往往导致所获得的跟踪数据价值很小（甚至没有任何价值）。

对于我们前述的示例性能测量，使用过滤器处理 2.4.4 节中的初始概要报告后，估计出单进程的内存总需求为 56MiB（这个值可通过重新处理过滤后的概要测量进行证实）。由于这个值超过了环境变量 SCOREP_TOTAL_MEMORY 的默认设置 16MiB，单独使用先前准备好的该过滤文件不足以避免中间跟踪缓冲区的刷新。此外，在开始跟踪采集与分析之前必须对环境变量 SCOREP_TOTAL_MEMORY 做相应的调整。（另外，也可以扩展该过滤文件，从而排除测量中的例程 binvrhs。）注意，不必去重命名或删除概要实验目录，因为创建跟踪实验目录时会添加后缀”trace”。

```
% export SCOREP_TOTAL_MEMORY=56M
% scalasca -analyze -q -t -f npb-bt.filt mpiexec -n 64 ./bt.D.64
S=C=A=N: Scalasca 2.1 trace collection and analysis
S=C=A=N: ./scorep_bt_64_trace experiment archive
S=C=A=N: Thu Apr  3 20:45:38 2014: Collect start
mpiexec -n 64  [...]  ./bt.D.64


 NAS Parallel Benchmarks 3.3 -- BT Benchmark

 No input file inputbt.data. Using compiled defaults
 Size:  408x 408x 408
 Iterations:  250    dt:   0.0000200
 Number of active processes:    64

 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
```

```
Time step  100
Time step  120
Time step  140
Time step  160
Time step  180
Time step  200
Time step  220
Time step  240
Time step  250
Verification being performed for class D
accuracy setting for epsilon =  0.1000000000000E-07
Comparison of RMS-norms of residual
          1 0.2533188551738E+05 0.2533188551738E+05 0.1479210131727E-12
          2 0.2346393716980E+04 0.2346393716980E+04 0.8488743310506E-13
          3 0.6294554366904E+04 0.6294554366904E+04 0.3034271788588E-14
          4 0.5352565376030E+04 0.5352565376030E+04 0.8597827149538E-13
          5 0.3905864038618E+05 0.3905864038618E+05 0.6650300273080E-13
Comparison of RMS-norms of solution error
          1 0.3100009377557E+03 0.3100009377557E+03 0.1373406191445E-12
          2 0.2424086324913E+02 0.2424086324913E+02 0.1582835864248E-12
          3 0.7782212022645E+02 0.7782212022645E+02 0.4053872777553E-13
          4 0.6835623860116E+02 0.6835623860116E+02 0.3762882153975E-13
          5 0.6065737200368E+03 0.6065737200368E+03 0.2474004739002E-13
Verification Successful

BT Benchmark Completed.
Class           =                        D
Size            =            408x 408x 408
Iterations      =                      250
Time in seconds =                   495.56
Total processes =                       64
Compiled procs  =                       64
Mop/s total     =                117715.93
Mop/s/process   =                  1839.31
Operation type  =           floating point
Verification    =               SUCCESSFUL
Version         =                      3.3
Compile date    =              03 Apr 2014

Compile options:
   MPIF77       = scorep mpif77
   FLINK        = $(MPIF77)
   FMPI_LIB     = (none)
   FMPI_INC     = (none)
   FFLAGS       = -O2
   FLINKFLAGS   = -O2
   RAND         = (none)


Please send the results of this run to:

NPB Development Team
```

```
 Internet: npb@nas.nasa.gov

 If email is not available, send this to:

 MS T27A-1
 NASA Ames Research Center
 Moffett Field, CA  94035-1000

 Fax: 650-604-3957


S=C=A=N: Thu Apr  3 20:53:59 2014: Collect done (status=0) 501s
S=C=A=N: Thu Apr  3 20:54:00 2014: Analyze start
mpiexec -n 64 scout.mpi ./scorep_bt_64_trace/traces.otf2
SCOUT   Copyright (c) 1998-2014 Forschungszentrum Juelich GmbH
        Copyright (c) 2009-2014 German Research School for Simulation
                                Sciences GmbH

Analyzing experiment archive ./scorep_bt_64_trace/traces.otf2

Opening experiment archive ... done (0.046s).
Reading definition data    ... done (0.063s).
Reading event trace data   ... done (1.138s).
Preprocessing              ... done (2.129s).
Analyzing trace data       ...
  Wait-state detection (fwd)      (1/4) ... done (0.835s).
  Wait-state detection (bwd)      (2/4) ... done (0.391s).
  Synchpoint exchange             (3/4) ... done (0.645s).
  Critical-path analysis          (4/4) ... done (0.332s).
done (2.217s).
Writing analysis report    ... done (0.191s).

Max. memory usage         : 173.016MB
*** WARNING ***
9982 clock condition violations detected:
   Point-to-point:     9982
   Collective    :        0
This usually leads to inconsistent analysis results.

Try running the analyzer using the '--time-correct' command-line
option to apply a correction algorithm.


Total processing time     : 5.907s
S=C=A=N: Thu Apr  3 20:54:06 2014: Analyze done (status=0) 6s
Warning: 2.968GB of analyzed trace data retained in ./scorep_bt_64_trace/traces!
S=C=A=N: ./scorep_bt_64_trace complete.

% ls scorep_bt_64_trace
scorep.cfg   scorep.log   scout.log  traces.def   trace.stat
scorep.filt  scout.cubex  traces     traces.otf2
```

跟踪采集与分析成功以后，生成的实验目录 scorep_bt_64_trace 包含：测量配置文件 scorep.cfg，测量日志文件 scorep.log 以及过滤文件的拷贝 scorep.filt。另外，还生成 OTF2 跟踪归档文件，其中包括文件 traces.otf2，全局定义文件 traces.def 以及目录 traces，该目录下记录了每个进程的数据文件。最后，实验还生成跟踪分析报告 scout.cubex 和 trace.stat 以及存储跟踪分析器输出的日志文件 scout.log。

Scalasca 跟踪分析工具同时会检测点到点通信过程中的时钟违背条件，并输出警告信息。违背时钟条件指因为各计算节点时钟之间同步不够充分而违反逻辑事件顺序。例如，接收操作可能在相应的发送操作开始前就已结束。这种事显然

是不可能发生的。Scalasca 跟踪分析器包含一个校正算法[1]，可用于恢复逻辑事件顺序，同时尽力保持时钟违背发生处附近的本地事件之间的时间间隔长度。

为了使用该校正算法，必须将命令行选项”--time-correct”传递给 Scalasca 跟踪分析器。然而，由于分析器隐式地通过命令 scalasca -analyze 启动，所以这个选项必须通过环境变量 SCAN_ANALYZE_OPTS 设置。命令 scalasca -analyze 会将此环境变量设置的命令行选项发送给跟踪分析器。可通过命令 scalasca -analyze 的选项-a 来重新分析已有的跟踪测量，以避免重新采集实验：

```
% export SCAN_ANALYZE_OPTS="--time-correct"
% scalasca -analyze -a mpiexec -n 64 ./bt.D.64
S=C=A=N: Scalasca 2.1 trace analysis
S=C=A=N: ./scorep_bt_64_trace experiment archive
S=C=A=N: Thu Apr  3 20:54:07 2014: Analyze start
mpiexec -n 64 scout.mpi --time-correct ./scorep_bt_64_trace/traces.otf2
SCOUT   Copyright (c) 1998-2014 Forschungszentrum Juelich GmbH
        Copyright (c) 2009-2014 German Research School for Simulation
                                Sciences GmbH

Analyzing experiment archive ./scorep_bt_64_trace/traces.otf2

Opening experiment archive ... done (0.006s).
Reading definition data    ... done (0.008s).
Reading event trace data   ... done (1.126s).
Preprocessing              ... done (2.107s).
Timestamp correction       ... done (0.683s).

Analyzing trace data       ...
  Wait-state detection (fwd)      (1/4) ... done (0.829s).
  Wait-state detection (bwd)      (2/4) ... done (0.383s).
  Synchpoint exchange             (3/4) ... done (0.644s).
  Critical-path analysis          (4/4) ... done (0.335s).
done (2.198s).
Writing analysis report    ... done (0.130s).

Max. memory usage         : 204.395MB

 # passes        : 1
 # violated      : 2679
 # corrected     : 18791313
 # reversed-p2p  : 2679
 # reversed-coll : 0
 # reversed-omp  : 0
 # events        : 263738186
 max. error      : 0.000072 [s]
 error at final. : 0.000000 [%]
 Max slope       : 0.010000000

Total processing time     : 6.383s
S=C=A=N: Thu Apr  3 20:54:13 2014: Analyze done (status=0) 6s
Warning: 2.968GB of analyzed trace data retained in ./scorep_bt_64_trace/traces!
S=C=A=N: ./scorep_bt_64_trace complete.
```

注意：

与跟踪数据的 I/O 时间及开始另一个分析作业在批处理队列里的等待时间相比，执行时间戳校正算法所需要的额外时间开销相对较小。在很可能发生违背时钟条件的平台（比如 cluster）上，默认使能时间戳校正算法会很方便。

类似于概要报告，可使用命令 scalasca –examine 调用 Cube 报告浏览器来对跟踪分析报告进行后处理并进行交互式浏览：

```
% scalasca -examine scorep_bt_64_trace
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt_64_trace/trace.cubex...
```

Scalasca 跟踪分析器生成的报告仍然是采用 CUBE4 格式的剖析，不同的是，里面富含额外的性能属性。仔细查看分析报告可发现大约一半的 MPI 点到点通信中是等待时间，这些等待时间均等地分成 *Late Sender* 和 *Late Receiver* 等待状态（参见图 2.2）。 在总结剖析中，solve_cell 的执行时间看上去似乎相对均衡，但对 *Critical path imbalance* 度量值的分析显示，事实上这些例程表现出少量的不均衡性，这将有可能在下一个同步点处导致等待状态。

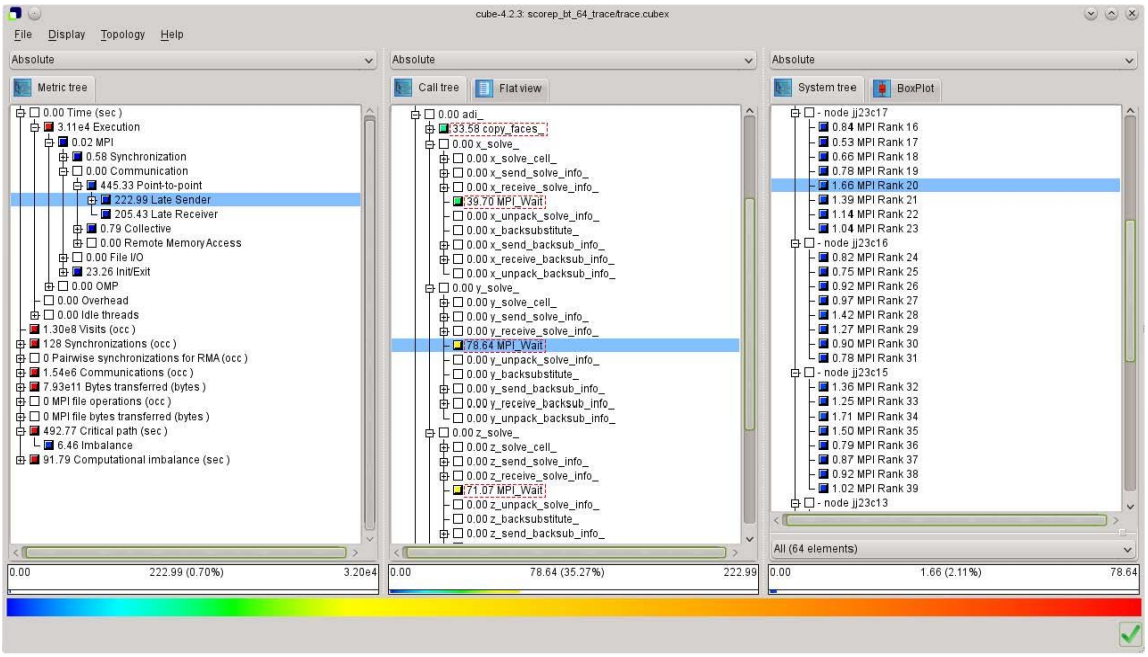

图 2.2　使用 Cube 报告浏览器查看跟踪分析结果

3 参考文献

[1] D. Becker, R. Rabenseifner, F. Wolf, and J. Linford. Scalable timestamp synchronization for event traces of message-passing applications. Parallel Computing, 35(12):595–607, Dec. 2009. 24

[2] D. Böhme, B. R. de Supinski, M. Geimer, M. Schulz, and F. Wolf. Scalable criticalpath based performance analysis. In Proc. of the 26th IEEE International Parallel & Distributed Processing Symposium (IPDPS), Shanghai, China, pages 1330–1340. IEEE Computer Society, May 2012. 1

[3] Cube User Guide. Available as part of the Cube installation or online at http://www.scalasca.org/software/cube-4.x/documentation.html. 19

[4] D. Eschweiler, M. Wagner, M. Geimer, A. Knüpfer, W. E. Nagel, and F. Wolf. Open Trace Format 2 - The next generation of scalable trace formats and support libraries. In Applications, Tools and Techniques on the Road to Exascale Computing (Proc. of Intl. Conference on Parallel Computing, ParCo, Aug./Sept. 2011, Ghent, Belgium), volume 22 of Advances in Parallel Computing, pages 481–490. IOS Press, May 2012. 1

[5] M. Geimer, F. Wolf, B. J. N. Wylie, and B. Mohr. A scalable tool architecture for diagnosing wait states in massively parallel applications. Parallel Computing, 35(7):375–388, July 2009. 1

[6] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel. The Vampir performance analysis toolset. In Tools for High Performance Computing (Proc. of the 2nd Parallel Tools Workshop, July 2008, Stuttgart, Germany), pages 139–155. Springer, July 2008. 1, 6

[7] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, and F. Wolf. Score-P – A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir. In Tools for High Performance Computing 2011 (Proc. of 5th Parallel Tools Workshop, Sept. 2011, Dresden, Germany), pages 79–91. Springer, Sept. 2012. 1

[8] Message Passing Interface Forum. MPI: A message-passing interface standard, Version 3.0. http://www.mpi-forum.org, Sept. 2012. 1

[9] NASA Advanced Supercomputing Division. NAS Parallel Benchmarks website. https: //www.nas.nasa.gov/publications/npb.html. 7

[10] OpenMP Architecture Review Board. OpenMP application program interface, Version 4.0. http://www.openmp.org, July 2013. 1

[11] Scalasca 2.x series documentation web page. http://www.scalasca.org/software/ scalasca-2.x/documentation.html. 7

[12] Score-P User Manual. Available as part of the Score-P installation or online at http://www.score-p.org. 3, 5, 6, 14, 16

[13] S. S. Shende and A. D. Malony. The TAU parallel performance system. International Journal of High Performance Computing Applications, 20(2):287–311, May 2006. 1

www.scalasca.org