Oracle® Solaris Studio 12.3 发行版的新增功能

文件号码 E26449
2011 年 12 月，E26449-01

ORACLE

120220@25097

目录

前言

本指南介绍了 Oracle Solaris Studio 12.3 中的新增功能和更改的功能、已知问题和限制。

支持的平台

此 Oracle Solaris Studio 发行版支持使用以下体系结构的平台：运行 Oracle Solaris 操作系统的 SPARC 系列处理器体系结构，以及运行 Oracle Solaris 或特定 Linux 系统的 x86 系列处理器体系结构。

本文档使用以下术语说明 x86 平台之间的区别：

- "x86" 泛指 64 位和 32 位的 x86 兼容产品系列。
- "x64" 指特定的 64 位 x86 兼容 CPU。
- “32 位 x86”指出了有关基于 x86 的系统的特定 32 位信息。

在 SPARC 和 x86 系统中，特定于 Linux 系统的信息仅指受支持的 Linux x86 平台，而特定于 Oracle Solaris 系统的信息仅指受支持的 Oracle Solaris 平台。

有关支持的硬件平台和操作系统发行版的完整列表，请参见《Oracle Solaris Studio 12.3 发行说明》。

Oracle Solaris Studio 文档

可以查找 Oracle Solaris Studio 软件的完整文档，如下所述：

- 产品文档位于 Oracle Solaris Studio 文档 Web 站点，包括发行说明、参考手册、用户指南和教程。
- 代码分析器、性能分析器、线程分析器、dbxtool、DLight 和 IDE 的联机帮助可以在这些工具中通过 "Help"（帮助）菜单以及 F1 键和许多窗口和对话框上的 "Help"（帮助）按钮获取。
- 命令行工具的手册页介绍了工具的命令选项。

相关的第三方 Web 站点引用

本文档引用了第三方 URL，以用于提供其他相关信息。

注 – Oracle 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的（或通过它们获得的）任何内容、广告、产品或其他资料，Oracle 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的（或通过它们获得的）任何内容、产品或服务而造成的或连带产生的实际或名义损坏或损失，Oracle 概不负责，也不承担任何责任。

开发者资源

对于使用 Oracle Solaris Studio 的开发者，可访问 Oracle 技术网 Web 站点来查找以下资源：

- 有关编程技术和最佳做法的文章
- 软件最新发行版完整文档的链接
- 有关支持级别的信息
- 用户论坛。

获取 Oracle 支持

Oracle 客户可通过 My Oracle Support 获取电子支持。有关信息，请访问 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info`，或访问 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs`（如果您听力受损）。

印刷约定

下表介绍了本书中的印刷约定。

表 P–1 印刷约定

字体或符号	含义	示例
`AaBbCc123`	命令、文件和目录的名称；计算机屏幕输出	编辑 `.login` 文件。 使用 `ls -a` 列出所有文件。 `machine_name% you have mail.`
`AaBbCc123`	用户键入的内容，与计算机屏幕输出的显示不同	`machine_name%` **`su`** `Password:`

表 P–1 印刷约定 （续）

字体或符号	含义	示例
aabbcc123	要使用实名或值替换的命令行占位符	删除文件的命令为 `rm` *filename*。
AaBbCc123	保留未译的新词或术语以及要强调的词	这些称为 *Class* 选项。 **注意**：有些强调的项目在联机时以粗体显示。
新词术语强调	新词或术语以及要强调的词	**高速缓存**是存储在本地的副本。 请**勿**保存文件。
《书名》	书名	阅读《用户指南》的第 6 章。

命令中的 shell 提示符示例

下表显示了 Oracle Solaris OS 中包含的缺省 UNIX shell 系统提示符和超级用户提示符。请注意，在命令示例中显示的缺省系统提示符可能会有所不同，具体取决于 Oracle Solaris 发行版。

表 P–2 shell 提示符

shell	提示符
Bash shell、Korn shell 和 Bourne shell	`$`
Bash shell、Korn shell 和 Bourne shell 超级用户	`#`
C shell	`machine_name%`
C shell 超级用户	`machine_name#`

◆ ◆ ◆ 第 1 章

1 Oracle Solaris Studio 12.3 发行版简介

此 Oracle Solaris Studio 发行版提供了许多新增功能和更改的功能，在本新增功能指南中进行了介绍。本指南取代了在 Sun Developer Network 门户站点上早期发行版中发布的组件自述文件。

什么是 Oracle Solaris Studio？

Oracle Solaris Studio 包含一整套工具，可用于 Oracle Solaris 和 Linux 操作环境中的应用程序开发：

- 适用于 C、C++ 和 Fortran（`cc`、`CC` 和 `f95`）的高性能优化编译器和运行时库，可本地实现 OpenMP 3.1 API，从而使共享内存并行化。
- 可脚本化的多线程感知交互式 dbx 命令行调试器和 dbxtool 调试器 GUI。
- 高度优化的多线程 Sun 性能库。
- 用于分析单线程和多线程应用程序以检测性能瓶颈和低效现象的性能分析器，以及使用 DTrace 技术在 Oracle Solaris 环境中进行系统分析的 DLight。
- 用于在多线程应用程序发生问题之前识别运行时潜在和难以检测的数据争用和死锁现象的线程分析器。
- 新增的代码分析器工具，用于结合分析静态代码错误、动态内存访问错误和代码覆盖数据，在代码中找到其他错误检测工具无法找到的重要错误。
- 专为与组件编译器、调试器和分析工具一起使用而定制的 IDE，以及用于生成应用程序的代码感知编辑器、工作流和项目功能。

可在 Oracle 技术网门户站点中的以下位置找到 Oracle Solaris Studio 完整文档集的链接：`http://www.oracle.com/technetwork/server-storage/solarisstudio/documentation`。

关于本新增功能指南

本指南分为关于编译器、库、性能分析工具、调试工具、IDE 和其他相关工具的独立章节。有关已知问题、限制和解决方法的章节概括介绍了关于 Oracle Solaris Studio 12.3 工具的其他信息。

要随时了解有关此发行版的信息，请转到 Oracle 技术网 上的 Oracle Solaris Studio 门户站点。

◆ ◆ ◆ 第 2 章

2 编译器

本章介绍了此 Oracle Solaris Studio C、C++ 和 Fortran 编译器发行版中新增和更改的功能。

编译器通用的新增/更改的功能

下面列出了自上一发行版起 C、C++ 和 Fortran 编译器通用的重大更改。可在编译器手册页和用户指南中找到详细信息。

- 支持新 SPARC T4 平台：`–xtarget=T4`, `–xchip=T4`, `–xarch=sparc4`
- 支持新增的 x86 Sandy Bridge AVX 平台：`–xtarget=sandybridge` `–xchip=sandybridge` `–xarch=avx`
- 支持新增的 x86 Westmere AES 平台：`–xtarget=westmere` `–xchip=westmere` `–xarch=aes`
- 新增编译器选项：`–g3` 添加了扩展的调试符号表信息。
- 新编译器选项：`–Xlinker` *arg* 将参数传递给链接程序 `ld`(1)。
- OpenMP 缺省线程数 `OMP_NUM_THREADS` 现在为 2（以前是 1）。
- 支持 3.1 OpenMP 共享内存并行化规范。
- 使用 `–library=sunperf` 可链接到 Sun 性能库。`-xlic_lib=sunperf` 已废弃。
- 支持用户提供的编译器选项缺省值。
- 对传统 SPARC 体系结构 V7，V8 和 V8a 的 `–xarch` 支持已经删除。

C 编译器

下面列出了此 5.12 发行版中特定于 C 编译器的新增和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.3：C 用户指南》和 cc 手册页。

- 新子选项 –xbuiltin=%default 仅内联未设置 errno 的函数。errno 的值在任何优化级别上都始终是正确的，并且可以可靠地检查。
- –xkeepframe 选项禁止对命名函数进行与堆栈相关的优化。
- 此发行版中已废弃对 -features=%none 和 -features=%all 的使用。
- 可识别新属性 vector_size 和 returns_twice 。
- -xcheck=init_local 现在可根据基本类型初始化 VLA（variable length array，可变长度数组）。
- aligned 属性的功能已扩展为包括自动变量以及全局变量和静态变量。
- -xdumpmacros 提供了诸如宏定义、取消定义的宏和用法实例等的信息。
- 新选项 –xanalyze={code| no} 会生成对源代码的静态分析，可使用 Oracle Solaris 代码分析器进行查看。

C++ 编译器

下面列出了此 5.12 发行版中特定于 C++ 编译器的新增和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.3：C++ 用户指南》和 CC 手册页。

- 新编译器选项 –xivdep 设置 ivdep pragma 的解释。ivdep pragma 指示编译器忽略在循环中找到的部分或全部对数组引用的循环附带依赖性，以进行优化。这使得编译器可以执行各种循环优化，如微向量化、分发、软件流水操作等，其他情况下，无法执行这些优化。当用户知道这些依赖性无关紧要或者实际上永远不会发生时，可以使用该指令。
- –compat=4 子选项（“兼容模式”）被删除。缺省设置现在为 –compat=5。此外，针对 g++ 源和二进制兼容性的 -compat=g 选项先前仅适用于 Linux 平台，现在也已扩展到 Solaris/x86。
- 新选项 –features=cplusplus_redef 允许在命令行中通过 –D 选项重新定义以常规方式预定义的宏 __cplusplus。现在仍不允许在源代码中通过 #define 指令重新定义 __cplusplus。此外，此发行版中已废弃对 –features=%none 和 –features=%all 的使用。
- 新子选项 –xbuiltin=%default 仅内联未设置 errno 的函数。errno 的值在任何优化级别上都始终是正确的，并且可以可靠地检查。
- C99 头文件 stdbool.h 和 C++ 等效项 cstdbool 现在可用。在 C++ 中，头文件不起任何作用，提供它们只是为了与 C99 兼容。
- 新选项 –xanalyze={code| no} 会生成对源代码的静态分析，可使用 Oracle Solaris 代码分析器进行查看。

Fortran 编译器

下面列出了此 8.6 发行版中特定于 Fortran 编译器的新增和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.3： Fortran 用户指南》和 `f95` 手册页。

- 内部函数例程 `LEADZ`、`POPCNT` 和 `POPPAR` 以前的返回类型与参数类型相同。在此发行版中，为符合 Fortran 2008 标准，这些内部函数将返回一个缺省整数，无论参数为何种类型。这引入了与以前发行版的轻微不兼容性。
- 现在支持与多态性相关的面向对象的 Fortran 功能：
  - 支持的 OOF 功能：类型扩展和多态实体：`CLASS` 语句、无限制多态性、`SELECT TYPE` 构造、`ABSTRACT` 派生类型、`EXTENDS_TYPE_OF` 和 `SAME_TYPE_AS` 内部函数，以及到无限制指针的序列类型分配。
  - 不支持的 OOF 功能：类型绑定过程：类型绑定 `PROCEDURE` 声明、`GENERIC`、`DEFERRED`、`NON_OVERRIDABLE`、`PASS` 和 `NOPASS`。
- 其他 F2003/2008 新增功能：
  - 增强的结构构造函数：使用组件名称构造结构常量。
  - 模块派生类型和组件上的增强 `PUBLIC/PRIVATE` 访问控制。
  - 更多 Fortran 2008 数学内部函数支持。在 x86 平台上，除了 `ERFC_SCALED`、`NORM2` 以及某些 `REAL*16` 变量外，现在多数 Fortran 2008 数学内部函数都受支持。
  - 不带组件的派生类型。
  - `KIND` 参数已添加到 `ICHAR`、`IACHAR`、`ACHAR`、`SHAPE`、`UBOUND`、`LBOUND`、`SIZE`、`MINLOC`、`MAXLOC`、`COUNT`、`LEN`、`LEN_TRIM`、`INDEX`、`SCAN` 和 `VERIFY` 内部函数中。
  - `BACK` 参数已添加到 `MINLOC` 和 `MAXLOC` 内部函数中。
  - 添加了新的内部函数 `FINDLOC` 和 `STORAGE_SIZE`。
  - 新的关键字 `ERRMSG`、`SOURCE` 和 `MOLD` 已添加到 `ALLOCATE` 语句中，`ERRMSG` 已添加到 `DEALLOCATE` 语句中。

OpenMP

下面列出了此发行版中由 C、C++ 和 Fortran 编译器实现的 OpenMP 共享内存 API 的新增和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.3： OpenMP API 用户指南》。

- 此发行版完全支持 3.1 OpenMP API 规范。
- `PARALLEL` 和 `OMP_NUM_THREADS` 环境变量的缺省值已更改为 2。以前版本中为 1。这意味着缺省情况下，自动并行化和显式 OpenMP 并行化在执行期间可使用两个线程。以前，缺省情况是在一个线程中串行运行这些程序。

- `SUNW_MP_PROCBIND` 环境变量新增了两种用于将线程绑定到处理器的模式：`COMPACT` 和 `SCATTER`。有关详细信息，请参见《Oracle Solaris Studio 12.3： OpenMP API 用户指南》。

◆ ◆ ◆ 第 3 章

3 库

本章介绍了与此 Oracle Solaris Studio 发行版中与库相关的新增功能和更改的功能。

数学库

数学库 `libcx` 已从此发行版中删除。

Sun 性能库

此 Sun 性能库发行版可在 Oracle Solaris 操作系统上使用，也可在多种 Linux 操作环境中使用。

Sun 性能库是一组优化的高速数学子例程，用于解决线性代数和其他数字密集型问题。Sun 性能库基于可从 Netlib（网址为 `http://www.netlib.org/`）获得的公共域子例程集合，这些子例程经过增强和优化，并绑定在一起共同构成 Sun 性能库。性能库包括以下库：

- LAPACK 版本 3.1.1，用于解决线性代数问题。
- BLAS1（Basic Linear Algebra Subprograms，基础线性代数子程序），用于执行向量间运算。
- BLAS2，用于执行矩阵向量运算。
- BLAS3，用于执行矩阵间运算。
- Netlib Sparse-BLAS，用于执行稀疏向量运算。
- NIST Fortran Sparse BLAS 版本 0.5，用于执行基本的稀疏矩阵运算。
- SuperLU 版本 3.0，用于对稀疏线性方程组求解。
- 快速傅里叶变换 (Fast Fourier transform, FFT) 例程
- 直接稀疏求解器例程

兼容性

Sun 性能库中的 LAPACK 3.1.1 例程与 LAPACK 早期版本（包括 1.x、2.0 和 3.0）中的用户例程兼容，并与 LAPACK 3.1.1 中的所有例程兼容。但是，由于 LAPACK 3.1.1 中的内部更改，不能保证与内部例程相兼容。

可能不兼容的内部例程在 LAPACK 源代码（可通过 Netlib 获取）中称为辅助例程。有关辅助例程的某些信息，请参见《LAPACK Users' Guide》（《LAPACK 用户指南》），可通过工业和应用数学学会 (Society for Industry and Applied Mathematics, SIAM) Web 站点 `http://www.siam.org/` 获取该指南。

由于 LAPACK 辅助例程的用户界面会随着 LAPACK 发行版的不同而有所更改，所以 Sun 性能库中的 LAPACK 辅助例程的用户界面也会随之更改。用户通常可以调用与 LAPACK 3.1.1 兼容的辅助例程；但是未对辅助例程进行专门地说明、测试或支持。请注意，LAPACK 辅助例程的用户界面在未来的 Sun 性能库发行版中会发生改变，因此用户界面应符合该版本的 Sun 性能库所支持的 LAPACK 版本要求。

文档

目前，Sun 性能库包括以下文档：

- 手册页（3p 部分），用于描述库中的每个函数和子例程
- 《Oracle Solaris Studio Sun Performance Library User's Guide》介绍和显示了以下内容的使用示例：Sun 性能库例程、Fortran 和 C 接口、优化和并行选项、SPSOLVE 和 SuperLU 稀疏求解器软件包及 FFT 例程。

有关其他参考信息，请参见《LAPACK Users' Guide》（《LAPACK 用户指南》）第三版，Anderson, E. 与他人合著，工业和应用数学学会 (Society for Industrial and Applied Mathematics, SIAM) 于 1999 年出版，您可以通过该学会或当地书店找到本书。《LAPACK Users' Guide》（《LAPACK 用户指南》）是 Netlib 上提供的 LAPACK 3.1.1 基础例程的正式参考资料，它提供了 LAPACK 3.1.1 例程的数学说明。

此发行版中的新增功能和更改的功能

- 提高了 BLAS 在新增 Intel 和 SPARC 平台上的性能。

◆ ◆ ◆ 第 4 章

4 代码分析工具

本章介绍了此 Oracle Solaris Studio 发行版的代码分析工具中新增和更改的功能。

Discover

此发行版的 Discover 内存分析工具中增加了以下功能：

- 新增的 `-a` 选项可将错误数据写入 *binary_name.analyze*`/dynamic` 目录以供代码分析器使用。
- 新增的 `-F` 选项可确定运行已使用 Discover 检测过的二进制文件时如果该二进制文件派生会出现的情况。缺省情况下，Discover 继续从父进程收集内存访问错误数据。如果希望 Discover 在派生之后从子进程收集内存访问数据，请在使用 `discover` 命令来检测二进制文件时指定 `-F child`。
- 新增的 `-b` 选项可在运行检测过的二进制文件时自动启动指定的浏览器。
- 新增的 `-c` 选项可以告知 Discover 检查所有库、指定库或指定文件中所列出的库中的错误。
- 新增的 `-n` 选项可告知 Discover 不检查可执行文件中的错误。

有关更多信息，请参见 `discover`(1) 手册页和《Oracle Solaris Studio 12.3：Discover 和 Uncover 用户指南》。

Uncover

此发行版的 Uncover 代码覆盖工具中增加了以下功能：

- 新增的 `-a` 选项可将错误数据写入 *binary_name.analyze*`/coverage` 目录以供代码分析器使用。
- 新增的 `-c` 选项可打开指令、块和函数的执行计数报告。
- 新增的 `-o` 选项可将检测过的二进制文件写入指定文件。

有关更多信息，请参见 uncover(1) 手册页和《Oracle Solaris Studio 12.3：Discover 和 Uncover 用户指南》。

代码分析器

新增的代码分析器工具可以结合三种分析类型，帮助您生成既安全又强健的高质量 C 和 C++ 应用程序。代码分析器可以显示以下三种类型的数据：

- 使用 `-xanalyze=code` 选项生成二进制文件时收集到的静态代码问题
- 在使用 Discover（Oracle Solaris Studio 内存错误搜索工具）检测并运行您的二进制文件时所检测到的动态内存访问问题
- 在使用 Uncover（Oracle Solaris Studio 代码覆盖工具）检测并运行您的二进制文件时检测到的代码覆盖问题

代码分析器会显示分析结果，包括检测到问题的源文件中的代码片段（相关源代码行突出显示）、静态问题的错误路径以及动态问题的调用堆栈（如果可用，还有 "Allocated At Stack"（堆栈上的已分配空间）和 "Free At Stack"（堆栈上的可用空间））。

您可以从错误路径或堆栈中的函数调用跳转到关联的源代码行，找到程序中此函数的所有使用实例，跳转到此函数的声明，以及显示此函数的调用图。

代码分析器可精确定位代码中的核心问题，修复这些问题后可能会消除其他问题。

有关更多信息，请参见代码分析器 GUI 中的联机帮助、《Oracle Solaris Studio 12.3 代码分析器用户指南》、《Oracle Solaris Studio 12.3 代码分析器教程》和 `code-analyzer`(1) 手册页。

◆ ◆ ◆ 第 5 章

性能分析工具

本章介绍了此 Oracle Solaris Studio 发行版的性能分析工具中新增和更改的功能。

性能分析器

本节介绍了此 Oracle Solaris Studio 发行版的性能分析器以及相关工具中新增和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.3： 性能分析器》手册和性能分析器中的帮助。

对性能分析器工具的更改

性能分析工具包含下列增强功能。

- 显著提高了处理大型实验（特别是 Java 实验）的性能。
- 对数据过滤做了多项改进，如第 22 页中的 “过滤增强功能”中所述。
- 现在，大多数数据标签都包括上下文菜单，右键单击标签即可打开上下文菜单。可使用这些上下文菜单找到特定于标签的高级功能（如过滤）。
- 现在，“实验比较”模式可以比较在不同的可执行文件和装入对象上的实验。要比较的源代码和反汇编代码现在在拆分窗格中显示两个版本。可以从 "Functions"（函数）和 "Source"（源）标签的上下文菜单启用“实验比较”模式。
- 标签中改进的快速导航使您可以执行以下操作：
  - 双击 "Functions"（函数）标签中的某个函数可以打开该函数的 "Source"（源）标签。
  - 在 "Lines"（行）标签中双击某一行可以在该行或其附近打开 "Source"（源）标签。
  - 在 "Source"（源）标签中双击某一行可以在该行的第一条指令或其附近打开 "Disassembly"（反汇编）标签。

  - 在 "PC" 标签中双击某一 PC 可以在该地址或其附近打开 "Disassembly"（反汇编）标签。
- 用于搜索在实验中引用的源文件的方法已更改。首先尝试路径映射，然后尝试将搜索路径与路径映射相结合，最后尝试原始完整路径。
- 对时间线做了多项改进，如第 22 页中的 “时间线增强功能”中所述。
- "Call Tree"（调用树）标签可将 HW 周期转换为 "User CPU Time"（用户 CPU 时间），您可从上下文菜单设置 "Call Tree"（调用树）中显示的度量。
- "Threads"（线程）标签新增了一个显示模式，称为 "Chart"（图表）。当包含时钟分析数据的实验启用了 "Chart"（图表）时，缺省 "Load Imbalance"（负载不平衡）图表会显示归属于每个线程的总 CPU 时间量。

时间线增强功能

性能分析器的 "Timeline"（时间线）标签包含下列增强功能：

- 可右键单击 "Timeline"（时间线）打开上下文菜单，以过滤数据、选择事件、进行缩放、恢复至前一视图或更改时间线属性。
- 事件频率图，以时间的函数形式显示事件频率的折线图。缺省情况下不显示此图表，必须在 "Set Data Preferences"（设置数据首选项）对话框中进行选择。
- 事件状态图，以时间的函数形式显示应用程序在各种状态下所花费时间的分配情况的条形图。对于在 Oracle Solaris 上记录的时钟分析数据，事件状态图显示 Oracle Solaris 微状态。缺省情况下不显示此图表，必须在 "Set Data Preferences"（设置数据首选项）对话框中进行选择。
- 双击 "Timeline"（时间线）现在将放大函数颜色选择器对话框，而不是打开此对话框。
- 新增的 "Timeline Details"（时间线详细信息）标签取代了 "Event"（事件）标签。新增的标签不仅如之前一样提供了事件信息，而且还包括用于导航时间线、缩放和更改函数颜色的按钮。
- 现在，时间线使用您在启动性能分析器时所指定的字体。
- 时间线可以设置为显示从各实验聚集的事件。这不是缺省设置，必须通过在 "Set Data Preferences"（设置数据首选项）对话框的 "Timeline"（时间线）标签中选择 "Group Data by: Experiment"（数据分组方式：实验）来进行设置。

过滤增强功能

性能分析器的数据过滤已经简化和增强：

- 现在，大多数数据标签都包括上下文过滤器，右键单击标签即可选择。新增的上下文过滤器和现有上下文过滤器的名称也更易于理解。
- 现在，上下文过滤器可以立即过滤数据，不再需要您在单独的对话框中进行应用。上下文菜单过滤器的选择会影响所有标签使用的数据。所选的每个过滤器都与现有的过滤器相结合，可以对数据进行渐进式过滤。

- "Filter Data"（过滤数据）对话框已经简化，并重命名为 "Manage Data Filters"（管理数据过滤器）。但在过滤数据时，建议从数据标签的上下文菜单中进行过滤。

  可使用 "Manage Data Filters"（管理数据过滤器）对话框的 "Custom"（定制）标签查看已从上下文菜单应用的过滤器的当前状态。"Custom"（定制）标签允许用户编辑当前的过滤器表达式。它还支持撤消、重做和 "Show Keywords"（显示关键字）（一个描述由上下文菜单过滤器生成的过滤表达式中可能会出现的符号的对话框）。
- 可使用通过新增的 `er_label` 命令添加的标签来过滤实验。
- 联机帮助中有关于过滤的扩展信息。

有关更多信息，请参见《Oracle Solaris Studio 12.3：性能分析器》中的“过滤数据”。

新增 er_label 命令

`er_label` 命令可以定义实验的一部分并为其分配名称或添加标签。标签捕获在实验中以开始时间和停止时间标记定义的一个或多个时间段内发生的分析事件。

您可通过在命令行运行 `er_label` 命令或在脚本中执行该命令来为实验分配标签。向实验添加标签后，便可以使用标签进行过滤。例如，您可以通过过滤实验包括或排除标签所定义时间段内的分析事件。

`er_label` 的一个用途是支持将由客户端驱动的服务器程序作为一个独立的进程或多个进程进行分析。在这种使用模型中，使用 `collect` 命令启动服务器，以便开始在服务器上创建实验。服务器启动并准备好接受客户端请求后，您便可运行客户端脚本，以请求驱动服务器并运行 `er_label` 来标记发生客户端请求的实验部分。

有关更多信息，请参见《Oracle Solaris Studio 12.3：性能分析器》中的“标记实验”。

对实验的更改

实验格式已更改，版本号现在为 12.3，与 Oracle Solaris Studio 版本号匹配。

Oracle Solaris Studio 12.3 中的工具可以打开具有以下版本号的实验：

- 版本 10.1，使用 Oracle Solaris Studio 12.2 或 Sun Studio 12 Update 1 创建的实验。
- 版本 10.2，使用早期 Oracle Solaris Studio 12.3 发行版（如 Beta 发行版）创建的实验。
- 版本 12.3，使用 Oracle Solaris Studio 12.3 发行版创建的实验。

如果您尝试打开从以前发行版创建的实验，您将会得到一个错误，指示该实验必须用工具的早期版本读取。

对数据收集的更改

数据收集更改会影响 `collect` 命令、`dbx collector` 命令和 `er_kernel` 命令。

collect 实用程序

collect 实用程序在此发行版中进行了如下更改：

- collect 不会验证目标程序是否为 ELF 可执行文件，从而使您能够分析脚本而无需设置环境变量。如果目标是一个 ELF 可执行文件，collect 会检查该文件是否与其运行所在的计算机兼容。
- Oracle Solaris 上增加了对 SPARC T4、Intel Westmere 和 Sandy Bridge 芯片的硬件计数器支持。增加了 Linux 对 Westmere 的支持，但尚未实现 Linux 对 Sandy Bridge 的支持。
- 硬件计数器分析现在可以通过将 "+" 前置于任何精密硬件计数器前面，对二进制文件执行内存空间分析。目前只有 SPARC T4 和 T3 处理器支持精密硬件计数器。

  要确定哪些硬件计数器是精密的，请在 collect -h 命令的输出中查找关键字 precise。要在包含内存空间分析数据的实验中分析内存访问模式，请在 "Set Data Presentation"（设置数据显示）对话框的 "Tabs"（标签）标签中选择 "Memory Objects"（内存对象）标签，例如，Vaddress 或 Vline_64b。然后，您可使用所选 "Memory Objects"（内存对象）标签中的上下文过滤器按数据地址进行过滤。
- 通过 PerfEvents 框架在运行 2.6.32 以上版本的 Linux 内核的 Linux 版本上实现了硬件计数器支持，不需要内核修补程序；但对于早期系统，仍需要 perfctr 修补程序。
- 不带任何参数运行 collect 现在仅显示用法消息。要显示有关可用硬件计数器的信息，必须不带任何附加参数运行 collect–h。
- 现在，可以在 Linux 系统（包括 Oracle Linux 6）上使用 collect -p high 执行高分辨率时钟分析。

dbx 收集器

dbx 收集器在此发行版中进行了如下更改：

- Oracle Solaris 上增加了对 SPARC T4、Westmere 和 Sandy Bridge 芯片的硬件计数器支持。增加了 Linux 对 Westmere 的支持，但尚未实现 Linux 对 Sandy Bridge 的支持。
- 通过 PerfEvents 框架在运行 2.6.32 以上版本的 Linux 内核的 Linux 版本上实现了硬件计数器支持，不需要内核修补程序；但对于早期系统，仍需要 perfctr 修补程序。

er_kernel 实用程序

对用于分析 Oracle Solaris 内核的 er_kernel 实用程序进行了如下更改：

- er_kernel 现在可以对内核和应用程序执行分析。可使用 -F 选项控制是否跟随应用程序进程以及是否将其数据记录为内核实验的子实验。
- er_kernel 不再支持用于分析指定进程的 -T 选项。但是，可以在正则表达式中使用 -F 选项。

- er_kernel 实用程序可以使用 DTrace cpc 提供者（仅在运行 Oracle Solaris 11 的系统上可用）为内核收集硬件计数器溢出分析。就如使用 collect 命令一样，可以使用 er_kernel 命令的 -h 选项来执行内核的硬件计数器溢出分析。但不支持数据空间分析，所以 er_kernel 将忽略数据空间请求。
- 不带任何参数运行 er_kernel 现在仅显示用法消息。要显示有关可用硬件计数器的信息，必须不带任何附加参数运行 er_kernel–h。
- 如果芯片的硬件计数器溢出机制允许内核告知哪个计数器溢出，您可以分析芯片提供的任意数量的计数器；否则只能指定一个计数器。er_kernel–h 输出通过显示消息来指定您是否可以使用多个计数器，例如，"specify HW counter profiling for up to 4 HW counters"（为多达 4 个 HW 计数器指定 HW 计数器分析）。
- er_kernel 不会验证目标装入是否为 ELF 可执行文件，从而使您能够分析任何命令或脚本。

er_print 命令

er_print 命令在此发行版中进行了如下更改：

- 显著提高了处理大型实验（特别是 Java 实验）的性能。
- 用于搜索在实验中引用的源文件的方法已更改。首先尝试路径映射，然后尝试将搜索路径与路径映射相结合，最后尝试原始完整路径。
- 可使用 tlmode 子命令设置时间线的缺省值，以显示从各实验聚集的事件。

线程分析器

Oracle Solaris Studio 12.3 线程分析器中新增或更改了以下功能。

- 现在，如果应用程序没有针对争用检测分析进行检测，线程分析器会通知您
- 线程分析器现在会自动在 "Races"（争用）标签中打开第一个调用堆栈。

DLight

DLight 是一种采用 Oracle Solaris 动态跟踪 (DTrace) 技术的交互式图形监测工具。DLight 可以同步方式运行多个 DTrace 脚本，并以图形方式向您显示输出，从而帮助您追踪到应用程序中运行时问题的根本原因。

Oracle Solaris Studio 12.3 DLight 中新增或更改了以下功能。

- 新增的 "Process Tree Target"（进程树目标）可以分析它所创建的一个或全部进程，并以图形方式显示以下内容：
  - DLight 使用 "Process Tree Target"（进程树目标）分析的所有进程的所有线程的微状态集合。

  - 目标进程及其子进程的各线程的时间线形式的微状态转换。
  - 进程及其子进程的锁定统计信息。
  - 在运行目标进程的所有 CPU 分析的所有目标进程的所有线程的累积 CPU 使用情况。
  - 程序的进程树中大量使用 CPU 的区域，显示程序中的函数以及该函数及其调用的所有函数使用的 CPU 时间。
- 用于分析单一运行进程的 "Attach"（连接）目标已被重命名为 "Process"（进程）目标。
- AMP 目标已经删除。

有关更多信息，请参见 DLight 中的帮助和《Oracle Solaris Studio 12.3： DLight 教程》。

◆ ◆ ◆ 第 6 章

6 调试工具

此 Oracle Solaris Studio 发行版的调试工具中的新增功能。

dbx

新增功能和更改的功能

Oracle Solaris Studio 12.3 dbx 中新增或更改了以下功能。

- `dbx` 现在包括宏扩展。有关详细信息，请参见《Oracle Solaris Studio 12.3：使用 dbx 调试程序》中的附录 C "宏"，或在运行 `dbx` 时在 (dbx) 提示符处键入 `help macros`。
- 面向对象的 Fortran 支持：
  - `dbx` 现在支持类型扩展和多态指针。与 C++ 支持一致。
  - `output_dynamic_type` 和 `output_inherited_members` dbx 环境变量现在还可用于 Fortran。
  - 可以在 `print` 和 `whatis` 命令中使用 `-r`、`+r`、`-d` 和 `+d` 选项，以获得关于继承（父）类型和动态类型的信息。
- 除可分配的数组类型外，`dbx` 现在还支持 Fortran 可分配的标量类型。

◆ ◆ ◆ 第 7 章

7 Oracle Solaris Studio IDE

Oracle Solaris Studio 12.3 IDE（Integrated Development Environment，集成开发环境）提供了创建、编辑、生成、调试 C、C++ 或 Fortran 应用程序并分析其性能的模块。本章重点介绍此 Oracle Solaris Studio 发行版中有关 IDE 的重要信息。

用于启动 IDE 的命令是 `solstudio`。有关此命令的详细信息，请参见 `solstudio`(1) 手册页。

有关 IDE 的完整文档，请参见 IDE 中的联机帮助和《Oracle Solaris Studio 12.3 IDE 快速入门教程》。

新增功能和更改的功能

Oracle Solaris Studio 12.3 IDE 中新增或更改了以下功能：

- 基于 NetBeans IDE 7.0.1
- 新增的 "C/C++ Project From Binary File"（基于二进制文件的 C/C++ 项目）项目类型可以通过指定二进制文件、从中生成该文件的源文件位置、希望包含在项目中的文件以及是否希望项目中包括相关项，从现有二进制文件创建项目。
- 您现在可在本地主机上处理位于所定义远程主机上的项目，包括浏览和编辑远程主机文件系统中的文件。
- 可打开远程主机的终端窗口。
- 新增的模板特化功能简化了通用模板与模板特化之间的导航。要使用此导航，可在源代码编辑器的边界中右键单击标注图标或按 Ctlr+Alt 的同时右键单击模板类或模板方法。
- 您可以为 Oracle 数据库应用程序创建项目。要执行此操作，您使用的 Oracle Solaris Studio 安装必须包括可选的 Oracle Instant Client 组件。IDE 现在包括 ProC 支持。
- 源代码编辑器会在您键入时对项目执行静态代码错误检查，如果它检测到错误，则会在左边界处显示一个错误图标。
- 您可对项目运行内存访问错误检查。

- 新增的 "Run Command"（运行命令）项目属性可以指定在运行项目时应向项目的命令行提供的命令和参数。运行命令可以是一个 shell 脚本，对于库项目也可以是一个二进制文件。
- 您现在可以选择使用 `gdb` 调试器来调试使用 gcc 工具集合编译的代码。缺省值为 `dbx` 调试器。
- 新增的 "Desktop Distribution"（桌面分发）功能可以生成一个 zip 文件，其中包含将几乎在所有操作系统上运行的 IDE 和代码分析器分发，并可在远程服务器上使用 Oracle Solaris Studio 编译器和工具。在桌面系统上运行 IDE 时，它将生成分发的服务器视为远程主机，并访问 Oracle Solaris Studio 安装中的工具集合。

有关详细信息，请参见 IDE 中的联机帮助和《Oracle Solaris Studio 12.3 IDE 快速入门教程》。

软件要求

Oracle Solaris Studio IDE 需要 Java SE Development Kit (JDK) 6 Update 24 或更高版本。如果 IDE 找不到所需的 JDK，将不会启动，并发出错误消息。

更新 IDE

Oracle Solaris Studio 12.3 IDE、dbxtool、DLight 监测工具和代码分析器的更新将在 Oracle Solaris Studio 产品修补程序中提供，而不通过 NetBeans 自动更新功能获取，缺省情况下 IDE 会禁用此功能。

在以下情况下，当您安装此类产品修补程序时，这些工具中可能会出现冲突：

- 如果您已经启用了工具中的自动更新功能并且进行了自动更新。
- 如果已经从 NetBeans 更新中心安装了插件。

要解决冲突：

- 如果您已使用 Solaris 10 上的软件包安装程序或 Solaris 11 上的 IPS 系统信息库安装了 Oracle Solaris Studio 工具，请从位于 `~/.solstudio` 的 Oracle Solaris Studio 用户目录中删除 `ide-12.3-`*`OS-architecture`*（对于 IDE 或 DLight）、`dbxtool-12.3–`*`OS-architecture`* 或 `code-analyzer–12.3–`*`OS-architecture`*。
- 如果您使用下载 tarfile 安装了 Oracle Solaris Studio 工具，则重新安装该 tarfile。

配置

会自动根据系统中的可用内存量确定 NetBeans IDE 7.0.1 的缺省堆大小。当您开发小型项目（最多包含 500 个源文件和头文件）时，Oracle Solaris Studio 12.3 IDE 通常会在此缺省设置下正常运行。

但是，当您要开发较大的项目时，则需要增加堆大小。如果在开发大型项目时收到 "OutOfMemory"（内存不足）的异常消息，则很可能是由于堆大小造成的。

您可以在 `netbeans.conf` 文件中设置运行 NetBeans IDE 的 Java 虚拟机 (Java Virtual Machine, JVM)* 的堆大小。

更改堆大小：

- 在 */Oracle_Solaris_Studio_installation_directory* `/lib/netbeans/etc` 中，将 `–J-Xmx` 命令行 Java 启动开关添加到 `netbeans.conf` 文件中，然后重新启动 IDE。

  例如：

```
netbeans_default_options="-J-Xms32m -J-Xmx128m –J-XX:PermSize=32m
     –J-XX:MaxPermSize=96m –J-Xverify:none –J-Dapple.laf.useScreenMenuBar=true"
```

对于大中型应用程序，建议 NetBeans C/C++ Plugin 的堆大小为：

- 如果要在具有 1 GB 或更大 RAM 的系统上开发中型应用程序（500–2000 个源文件和头文件），堆大小应为：512 MB
- 如果要在具有 2 GB 或更大 RAM 的系统上开发大型应用程序（2000 个以上源文件和头文件），堆大小应为：1 GB

如果您要运行 Oracle JVM，还可以在 `netbeans.conf` 文件中添加垃圾回收器开关 `-J-XX:+UseConcMarkSweepGC`（并发回收器）和 `-J-XX:+UseParNewGC`（并行回收器）。这些选项允许垃圾回收器以并行方式与主执行引擎一起运行。但是，非 Oracle 实现的 JVM 可能不支持这些选项。

有关 NetBeans 性能调整的更多信息，请参见调整 JVM 开关的性能。

请注意：术语“Java 虚拟机”和 "JVM" 表示适用于 Java(TM) 平台的虚拟机。

◆ ◆ ◆ 第 8 章

8 其他工具

dmake 中的新增功能和此 Oracle Solaris Studio 发行版中的软件安装程序。

dmake

dmake 是一个命令行工具，与 make(1) 兼容。dmake 能够以网格、分布、并行或串行模式生成目标。如果使用的是标准 make(1) 实用程序，在对 makefile 进行任何更改时可以毫不费力地过渡到使用 dmake。dmake 是 make 实用程序的超集。对于嵌套的 make，如果顶级 makefile 调用 make，则需要使用 $(MAKE)。dmake 会对 makefile 进行解析，并确定可以并发生成哪些目标，然后将这些目标的生成版本分布在您设置的许多主机上。

dmake 目前集成在 Oracle Solaris Studio IDE 中。缺省情况下，所有项目都是使用并行模式下运行的 dmake 生成的。用户通过项目属性可以指定完成生成任务的最大数量。缺省情况下，dmake 可以并行运行 2 个任务，这意味着许多项目的生成速度是多 CPU 系统上速度的两倍。

有关如何使用 dmake 的信息，请参见《Oracle Solaris Studio 12.3：分布式创建 (dmake)》手册。

此发行版的 dmake 实用程序中增加了以下功能。

除 rsh 之外，dmake 现在还可以使用 ssh 在生成服务器上远程执行命令。如果要使用 ssh，则必须在 .dmakerc 文件中指定 ssh 命令的远程路径。

可以在 .dmakerc 文件中指定远程 shell 的路径。

例如：

```
host earth { jobs = 3 }
host mars  { jobs = 5 , rsh = "/bin/ssh" }
```

如果未指定 rsh =，则缺省情况下 dmake 将使用 /bin/rsh。

与使用 rsh 时一样，您必须确保 ssh 无需口令即可登录到远程主机，并且不发出任何警告或错误。

此发行版中的软件更正

- 错误更正：dmake 将未转义的 :'s 写入到 .make.state，使自身中断。
- 错误更正：手册页更新：新增选项 -m grid（SGE 支持）。
- 错误更正：dmake 手册页的“用法概要”一节中缺少“分布式”模式。
- 错误更正：dmake 忽略命令行选项 -x SUN_MAKE_COMPAT_MODE=GNU。

Oracle Solaris Studio 安装程序

安装程序中的新增功能和更改的功能包括：

- 非 GUI 安装程序现在允许您指定要安装的组件。
- 使用新 -generate-desktop-dir 安装程序选项，可以生成一个包含 IDE 分发的 zip 文件，此 IDE 分发是为安装有几乎任意操作系统的桌面系统配置的。在安装 Oracle Solaris Studio 软件之后，您可以在桌面系统中解压缩此文件。在桌面系统上运行 IDE 时，它将生成分发的服务器视为远程主机，并访问 Oracle Solaris Studio 安装中的工具集合。
- 新增的 -nfs-server 安装程序选项以 NFS 服务器模式运行安装程序，不会检索服务器是否安装了必需的 OS 修补程序，不会安装指向 Oracle Solaris Studio 软件和 /usr/bin 和 /usr/share/man 目录中手册页的符号链接。
- 使用新增的 -ignore-architecture 安装程序选项，可以在基于 x86 平台上安装基于 SPARC 平台的 Oracle Solaris Studio 组件，或在基于 SPARC 平台上安装基于 x86 平台的组件。
- 使用新增的 -force-uninstall 卸载程序选项，可以在 NBI 注册表被破坏时强制删除 Oracle Solaris Studio 软件包和安装目录。

◆ ◆ ◆ 第 9 章

9 此发行版中的已知问题、限制和解决方法

以下是发行此版本时的一些已知问题以及有关如何解决这些问题的信息。最新问题列在《Oracle Solaris Studio 12.3 发行说明》中。

编译器

本节介绍了此发行版中编译器的已知问题、问题和解决方法。

编译器共有的问题

文档勘误表

下面列出了已发布编译器文档中的错误。

- cc(1)、CC(1) 和 f95(1) 手册页没有列出 –xarch=sse3a 标志，此标志会将 AMD 指令集（包括 3dnow）添加到 SSE3 指令集中。
- C 和 C++ 文档没有指明 –xMF 选项只可用于 –xMD 或 –xMMD，而不可用于 –xM 或 –xM1。如果指定，它会覆盖用于这些选项的缺省 .d 文件名称。

C++

Solaris 上的 Apache 标准库问题

安装在 Solaris 10u10 和更早版本以及 Solaris 11 最初发行版中的 Apache stdcxx 库在头文件 stdcxx4/loc/_moneypunct.h 中有语法错误。早期版本的编译器没有发现此错误，但 Oracle Solaris Studio 12.3 C++ 编译器发现了此错误。无法禁用错误检测。

对此错误的修复在 Solaris 10 的修补程序和首个 Solaris 11 SRU 中提供。Solaris 10u11 和 Solaris 11u1 推出时将包含此修复。

多义性：构造函数调用或指向函数的指针

某些 C++ 语句有可能解释成声明或者表达式语句。C++ 消除歧义规则为：如果一个语句可以处理成声明，那么它就是声明。

早期版本的编译器会错误解释类似于下面的代码：

```
struct S {
  S();
};
struct T {
  T( const S& );
};
T v( S() );    // ???
```

编程人员也许本来打算在最后一行定义变量 v，并且用类型为 S 的临时变量对它进行初始化。早期版本的编译器会这样解释这条语句。

但是在声明环境里，构造符号 "S()" 也可以是抽象声明符（不带标识符），表示“没有返回值类型为 S 的参数的函数”。在这种情况下，该语句会自动转换为函数指针 "S(*)()"。这样该语句仍可作为函数 v 的声明，该函数有一个函数指针类型的参数，返回类型为 T 的值。

当前版本的编译器可以正确地解释该语句，但这未必是编程人员所需要的结果。

可以使用两种方法来修改上述代码以便不产生歧义：

```
T v1( (S()) );  // v1 is an initialized object
T v2( S(*)() ); // v2 is a function
```

第一行中另加的圆括号表明它不是 v1 作为函数声明的有效语法，所以它的唯一可能解释是“利用类型为 S 的临时值进行初始化的类型为 T 的目标”。

同样，构造符号 "S(*)()" 不可能是一个值，所以它的唯一可能解释是函数声明。

第一行也可以改写为：

```
T v1 = S();
```

虽然这时语句含义非常清楚，但这种形式的初始化有时会创建一个额外的临时变量，而一般情况下不会发生这种情况。

建议不要编写与下面语句类似的代码，因为它的含义不清楚，不同的编译器可能会提供不同的结果。

```
T v( S() ); // not recommended
```

如果将 -xipo 或 -xcrossfile 与 -instances=static 组合，链接会失败

模板选项 -instances=static（或 -pto）在与 -xcrossfile 或 -xipo 选项组合时无效。使用该组合的程序会经常发生链接失败。

如果使用 -xcrossfile 或 -xipo 选项，请使用缺省的模板编译模型 -instances=global 进行替代。

通常，不要使用 -instances=static（或 -pto）。它不再有任何优点，此外，C++ 用户指南中还对其缺点进行了说明。

名称改编链接问题

以下情况会导致链接问题。

- 函数在一个地方声明带有一个 const 参数，而在另一个地方又声明带有一个非 const 参数。

  示例：

  ```
  void foo1(const int);
  void foo1(int);
  ```

  这两个声明是等效的，但编译器会将其改编为两个不同的名称。要避免这个问题，则不应将值参数声明为 const。例如，在任何位置都使用 void foo1(int)，包括该函数定义体。

- 函数有两个具有相同复合类型的参数，但只有一个参数是用 typedef 声明的。

  示例：

  ```
  class T;
  typedef T x;
  // foo2 has composite (that is, pointer or array)
  // parameter types
  void foo2(T*, T*);
  void foo2(T*, x*);
  void foo2(x*, T*);
  void foo2(x*, x*);
  ```

  所有的 foo2 声明都是等效的，并且应该改编相同的名称。但是，编译器只会改编部分声明的名称。为了避免这个问题，应该统一使用 typedef。

  如果您无法统一使用 typedef，解决方法是：在定义该函数的文件中使用弱符号，使得声明与函数的定义一致。例如：

  ```
  #pragma weak "__1_undefined_name" = "__1_defined_name"
  ```

  请注意，某些改编名称依赖于目标体系结构。（例如，在 SPARC V9 体系结构 (-m64) 中，size_t 是 unsigned long，而在其他体系结构中是 signed int。）在这种情况下，会出现两个版本的改编名称，分别对应两个模式。这时必须使用两个 pragma，并用适当的 #if 指令对其进行控制。

不支持引用模板中的非全局名称空间目标

如果您使用 -instances=extern 编译，则使用模板和静态对象的程序会出现未定义符号的链接时错误。使用缺省设置 -instances=global 则不会出现问题。编译器不支持对模板中的非全局名称空间作用域目标的引用。请看以下示例：

```
static int k;
template<class T> class C {
        T foo(T t) { ... k ... }
};
```

在本示例中，一个模板类的成员引用了静态名称空间作用域的变量。请记住，名称空间作用域包含文件作用域。编译器不支持模板类的成员引用静态名称空间作用域的变量。另外，如果模板在其他的编译单元实例化，那么每个实例都会指向不同的 k，这破坏了 C++ 一次定义规则，代码的行为将会不可预测。

下面的方法也是可行的，但这取决于您如何使用 k，以及它应有的功能。第二个选项仅可供属于类成员的函数模板使用。

1. 可以为变量提供外部链接属性：

   ```
   int k; // not static
   ```

   所有的实例都使用同一个 k。

2. 也可以使这个变量成为类的静态成员：

   ```
   template<class T> class C {
           static int k;
           T foo(T t) { ... k ... }
   };
   ```

   静态类成员具有外部链接属性。每个 C<T>::foo 的实例都使用不同的 k。而 C<T>::k 的一个实例可以被其他函数共享。此选项可能是您需要的选项。

名称空间内的 #pragma align 需要改编名称

在名称空间内使用 #pragma align 时，必须使用改编名称。例如，在下面的代码中，#pragma align 语句是无效的。要更正此问题，应将 #pragma align 语句中的 a, b 和 c 替换为其改编名称。

```
namespace foo {
  #pragma align 8 (a, b, c) // has no effect
  //use mangled names: #pragma align 8 (__1cDfooBa_, __1cDfooBb_, __1cDfooBc_)
  static char a;
  static char b;
  static char c;
}
```

Fortran

在此 f95 编译器发行版中应注意以下问题：

- 不换行打印行末尾之前的空格不会影响输出位置 (7087522)。

  输出语句格式末尾的 X 编辑描述符不会影响输出记录中后续字符的位置。如果输出语句中有 **ADVANCE='NO'** 并且有更多字符要通过后续输出语句传送到同一记录，则会导致差异。

在很多情况下，可以通过添加空白字符串编辑描述符，而不是 ***n*X** 编辑描述符来解决此问题。它们并不完全相同，因为空白字符串编辑描述符实际上会将空白字符包含在记录中，而 ***n*X** 仅跳过后 *n* 个字符，缺省情况下通常导致跳过的位置中产生空白。

- 一行中有两个连续 & 号的有效代码会被拒绝 (7035243)。

  Fortran 标准禁止有一个 & 号的空续行。但是，仍可以创建同一行中有两个 & 号的空续行，这不在标准限制范围之内。编译器不会处理这种情况，而会显示一条错误消息。解决方法是删除该行，这只会影响程序可读性而并不添加任何语义。

- BOZ 常量有时会被截断 (6944225)。

  在某些相对更复杂的情况下，如数组构造，BOZ 常量可能会被截成 4 个字节的缺省整数大小，即使它所应指定给的相应项是一个 8 个字节的整数实体。解决方法是在数组构造中使用正确类型和种类的常量，而不要使用 BOZ 常量。

Fortran 编译器的早期发行版引入了某些不兼容性，并被此编译器发行版所继承，如果您是从 Fortran 编译器早期发行版进行更新，应注意这一点。请注意下面的不兼容性：

Fortran 77 库已删除

需注意 Oracle Solaris Studio 12.2 发行版已删除了废弃的 FORTRAN 77 库。这意味着使用依赖于共享库 `libF77`、`libM77` 和 `libFposix` 的传统 Sun WorkShop `f77` 编译器编译的旧的可执行文件将不会运行。

数组内部函数使用全局寄存器：

数组内部函数 `ANY, ALL, COUNT, MAXVAL, MINVAL, SUM, PRODUCT`、`DOT_PRODUCT` 和 `MATMUL` 针对相应 SPARC 平台体系结构进行了高度优化。因此，它们使用全局寄存器 `%g2, %g3` 和 `%g4` 作为临时寄存器。

如果调用了上述所列的数组内在函数，则用户代码不应该认为这些寄存器可用于暂时存储。当调用数组内在函数时，这些寄存器中的数据将被覆盖。

归档库中的 F95 模块不包括在可执行文件中：

调试器 dbx 要求编译中使用的所有对象文件都包含在可执行文件中。通常，无需用户执行额外操作，程序即可满足此要求。但使用含有模块的归档文件时例外。如果程序使用了一个模块，但没有引用模块中的任何过程或变量，则产生的对象文件不会包含对模块中定义的符号的引用。只有对目标文件中定义的符号具有引用时，链接器才会链接归档文件中的对象文件。如果不存在此类引用，对象文件将不包括在可执行文件中。当 dbx 尝试查找与使用的模块相关联的调试信息时，将发出警告。对于缺少调试信息的符号，则无法提供有关这些符号的信息。

使用 `-u` 链接程序选项可以解决这个问题。此选项使用一个符号作为其选项参数。它会将该符号添加到未定义的链接程序符号集中，这就需要解析此符号。与模块关联的链接程序符号通常是模块名称，其所有字母均为小写，后面跟有一条下划线。

例如，为了强制包含模块 MODULE_1 的对象文件被归档文件采用，请指定链接程序选项 -u module_1_。如果使用 f95 命令进行链接，请在命令行上使用 -Qoption ld -umodule_1_。

Linux 平台上的 gethrtime(3F)

启用系统节能功能时，没有可靠方式可以精确地获得 AMD 处理器上的时钟速率。因此，使用基于 gethrtime(3F) 的计时函数（Fortran 编译器的 Linux 版 Solaris gethrtime(3C) 函数）在 Linux 平台上获得高精度实际时间的方法只有在禁用了节能功能的 AMD 系统才会精确。要禁用节能功能，可能需要重新引导系统。

工具

dbx

已知的 dbx 问题和解决方法

1. 当 dbx 连接到进程时发生数据收集问题

   如果将 dbx 连接到一个正在运行的进程，但是没有预先装入收集器库 libcollector.so，将发生一系列错误。

   - 无法收集任何跟踪数据：同步等待跟踪、堆跟踪或 MPI 跟踪。跟踪数据是通过对各个库执行插入操作而收集的。如果没有预先装入 libcollector.so，将无法执行插入操作。
   - 如果在 dbx 连接到进程后程序安装了一个信号处理程序，并且该信号处理程序不传递 SIGPROF 和 SIGEMT 信号，则分析数据和抽样数据将会丢失。
   - 如果程序使用异步 I/O 库 libaio.so，则基于时钟的分析数据和抽样数据将会丢失，因为 libaio.so 需要使用 SIGPROF 来执行异步取消操作。
   - 如果程序使用硬件计数器库 libcpc.so，则硬件计数器溢出分析实验将会遭到破坏，因为收集器和程序都在使用该库。如果在将 dbx 连接到进程后装入了硬件计数器库，只要通过广义搜索而不是在 libcpc.so 中搜索来解析对 libcpc 库函数的引用，硬件计数器实验即可顺利进行。
   - 如果程序调用 setitimer(2)，则由于收集器和程序同时使用定时器，可能会使基于时钟的分析实验中断。

2. dbx 在调试 Java 代码时可能会崩溃

   如果从 dbx shell 内部发出一个 cd 命令，或者设置 CLASSPATH 环境变量或 CLASSPATHX 环境变量，dbx 随后可能会因分段错误而崩溃。

   解决方法：

   - 请勿执行上述任何操作。

- 在执行上述任何操作前，请删除所有监视（显示）。

3. dbx 在重新调试 Java 代码时崩溃

   在 Java 代码的一行内发出两条 debug 命令可能会导致 dbx 崩溃。

4. 调试应用程序的 J2SE 不同于最初构建应用程序的 J2SE 时，dbx 会抛出异常。

   如果您在一个版本的 J2SE 技术下构建了应用程序，又在另一个不同发行版的 J2SE 下调试此应用程序，dbx 会抛出异常。

5. 由于 RTC 预监视分配而报告伪 RUA 错误

   在具有多线程程序的不寻常情况下，如果运行时检查 (runtime checking, RTC) 检测到在其开始监视内存分配之前分配的与线程有关的内部数据访问，会报告伪 RUA 错误。由于这些情况是正常线程切换行为的一部分，因此可以使用 dbx suppress 命令放心地忽略这些伪 RUA 报告。

dbx 限制和不兼容情况

Oracle Solaris Studio 12.3 dbx 有以下限制：

- 不能从 .dbxrc 文件连接至正在运行的进程。.dbxrc 文件不应含有执行代码的命令。但是，您可以将此类命令放在一个文件中，然后使用 dbx source 命令来执行该文件中的命令。
- dbx 无法正确地取消改编指向使用 -compat=4 选项编译的成员函数的指针。对于 -compat=5 选项不会发生此问题。

  注 – 此发行版中已删除了 -compat=4 选项。有关详细信息，请参见第 14 页中的 “C++ 编译器”。

- 在 SPARC V9 (-m64) 系统中，使用 call 命令或输出函数调用对作为参数或返回值的嵌套小结构不起作用。
- 使用 libC.so.5 或者 libC.so.4 的旧副本可能会在 C++ 异常区域中引起 dbx 问题。可能会出现关于错误的 stab 和未处理的异常等警告消息。

  解决方法：在所有系统上安装最新的 libC.so.5。
- Fortran 用户应该用 -stackvar 选项进行编译，以便充分利用运行时检查。
- 某些程序可能无法正常使用 -stackvar 选项。在这种情况下，请尝试使用 -C 编译器选项，它将在不使用运行时检查的情况下启用数组下标检查。
- 对于多线程的应用程序，跟踪派生可能不可靠。
- 使用 call 或输出函数调用可能会导致多线程应用程序发生死锁。
- 如果文件是预编译的头文件 (Pre-Compiled Header, PCH) 集合的一部分，请不要使用 dbx 的修复并继续功能来更改头文件。
- dbx 命令行解释器是旧版本的 Korn shell (ksh)，不支持代码集独立 (Code Set Independence, CSI)。当在 dbx 命令行上键入多字节字符时，会发生解释错误。

- Linux OS 上不能使用以下 dbx 功能：
  - 修复并继续功能
  - 收集性能数据
  - 在下列事件中设置断点：
    - fault
    - lastrites
    - lwp_exit
    - sysin
    - sysout
    - sync
    - throw
  - Java 调试
  - 调试 32 位程序（除非使用 -x exec32 选项启动 dbx）。
- 在 Linux 平台上调试程序时可能会发生下面的问题：
  - 在调用 exec() 时，dbx 无法跟踪 Linux 平台上的派生进程，或更改为新程序
  - Korn shell 中的管道操作符仅限于 Linux 平台。任何需要访问目标进程的 dbx 命令不能作为管道的一部分使用。例如，下面的命令可能会导致 dbx 挂起：

    ```
    where | head -1
    ```

    解决方法：
    - 按 Ctrl-C 组合键以显示新的 dbx 提示符。
    - dbx 可缓存大量信息，对于上述示例，下列命令序列会起作用：

      ```
      where
      where | head –1
      ```
  - 如果程序使用 clone() 实现自己样式的线程，则 dbx 中提供的线程支持不能正确识别这些线程。

    解决方法：

    使用 libthread.so，而不是 clone()。
  - Linux 操作系统中的线程库使用 SIGSTOP 信号作为其内部机制的一部分。通常，dbx 会对您隐藏这些信号，并允许您通过其他源监视真正的 SIGSTOP 信号。偶尔，Linux 操作系统会以意想不到的方式使用 SIGSTOP，dbx 将系统生成的 SIGSTOP 解释为用户生成的 SIGSTOP。

    解决方法：

    使用 ignore 命令告知 dbx 不要捕获 SIGSTOP 信号。
  - 有时线程退出，但 Linux 操作系统并不将退出行为报告给 dbx。

    当线程退出，但是未报告此退出操作时，dbx 会等待永远不发生的事件并且不显示新的提示符。这种情况最可能发生在您在 dbx 中提供了 cont 命令之后，但它也可能发生在 step up 命令、step 命令、next 命令和其他命令之后。

解决方法：

- 有时，键入 Ctrl+C 组合键会导致 dbx 停止等待并显示新的提示符。
- 如果 Ctrl+C 组合键不起作用，请退出 dbx 并重新启动。

- dbx 不支持 GNU C 和 C++ 编译器的下列功能：
  - VL 数组
  - 异常处理
  - OpenMP
  - RTTI
  - 模板定义
  - 缺省参数
  - using 指令
  - friend
- Oracle Linux 6 上的 dbx 存在以下问题：
  - 系统库中使用的间接引用符号有时会导致 dbx 在引用处不是实际函数处设置断点。
  - 在调试使用 gcc 4.4.4 编译器编译的代码时：
    - 在输出可变长度数组时 dbx 可能会挂起。
    - 在函数结尾处（} 行）停止时 dbx 无法查看局部变量。
    - dbx 不查看使用 -D 编译器选项定义的宏。

性能分析器

本节介绍性能分析器工具的已知问题。

- 有时 "Callers-Callees"（调用方-被调用方）标签会显示一个截断的度量值。
- 在 SPARC T4 处理器上可能会高估 icache 延迟时间。
- OMP 任务和 OMP 并行区域的 OpenMP 等待度量不会累加到 <Total> 的 OpenMP 等待度量中，因为在第一个并行区域项之前的任何分析数据包都不会计算成任何任务或区域的 OMP 等待时间，而是计入总计值。
- 有时不能正确处理添加和删除实验；解决方法是先装入所有实验，然后进行过滤以排除要去掉的实验。
- 有时不能正常移动标签。
- 有时不能正常选择标签中的多个项目。
- 分析器有时在 SummaryDisp.updateSummary(SummaryDisp.java:249) 处挂起
- 在 "Source-Disassembly"（源-反汇编）标签上做出选择时 "Summary"（摘要）标签不会更新。
- 即使存在非零度量时，也可能不显示 "Source"（源）标签的边界中的突出显示。
- 共享对象的 "Show/Hide/API-only"（显示/隐藏/仅 API）功能有时无法与过滤正常配合使用。使用 er_print 查看实验数据时也会出现此问题。

- "Manage Filters"（管理过滤器）对话框的 "General"（常规）标签中的 "Apply"（应用）按钮有时不能应用更改。多按几次 "Apply"（应用）也许会起作用。一个更可靠的解决方法是从 "Timeline"（时间线）、"Threads"（线程）和 "CPUs"（CPU）数据标签的上下文菜单中访问过滤器。
- 在 "Timeline"（时间线）标签中，"Threads"（线程）和 "CPUs"（CPU）可能并未按数字顺序显示。

collect 实用程序

本节介绍了 `collect` 实用程序和数据收集的已知问题。

- 对于某些经过优化的代码，有时不能在 Sandy Bridge 计算机上正确地展开堆栈。
- 由于 JDK 1.7 中锁定算法的实现方式有所更改，Java 程序上的同步跟踪会双倍计算所有 Java 同步事件。解决方法是将报告的值除以二。
- 在 Linux 系统中，多线程应用程序的时钟分析所报告的线程数据不精确。内核并不总是按指定的间隔将分析信号传送到每个线程；有时，信号传送到了错误的线程。如果可用，请利用使用 `cycle` 计数器的硬件计数器分析，以获得更精确的每个线程的结果。
- 在具有多个以不同时钟频率运行的 CPU 的系统上，时钟分析将基于一个 CPU 的时钟速率生成事件，这可能会导致多算或少算其他 CPU 上的事件。

线程分析器

本节介绍线程分析器的已知问题。

如果下列所有条件均为真，线程分析器可能会发生运行时失败：

- 在使用 `discover` 进行过数据争用检测的二进制文件上运行 `collect`
- 运行所在的计算机上安装了支持硬件能力过滤器的 SPARC 处理器（如 UltraSPARC T2）
- 计算机正在运行 Solaris 10 Update 9 或更早的更新

要避免此问题，应在运行 `collect` 之前，设置环境变量 `LD_NOAUXFLTR=yes` 以跳过装入过滤器库。由于没有使用过滤器，性能可能会受一些影响。

er_kernel 实用程序

本节介绍 `er_kernel` 实用程序的已知问题：

- `er_kernel` 有时会导致运行它的计算机崩溃。此问题仅在 UltraSPARC T1、T2 和 T2+ 芯片上运行的 Oracle Solaris 10 上出现过，是因虚拟机管理程序错误而引起的。
- `er_kernel` 分析有时会错误地认定 CPU 状态，不能正确地报告用户数据。

- 有时一个或多个 CPU 会停止生成 `er_kernel` 事件，持续时间达 30 秒或更长。
- 用户进程的 DTrace 堆栈展开有时会遗漏一个帧，通常在叶函数执行到其函数序言或函数尾声时发生。
- 在具有多个以不同时钟频率运行的 CPU 的系统上，`er_kernel` 分析将基于一个 CPU 的时钟速率生成事件，这可能会导致多算或少算其他 CPU 上的事件。

IDE

本节介绍 IDE 中的已知问题。

- 版本控制框架无法以完全远程模式运行。(195121)

  版本控制框架通常按照 `java.io.File` 运行，因此无法创建能够处理远程文件对象的插件。

  解决方法：通过 ssh 直接在远程主机上使用版本控制工具。

- 在某些使用 GDB 7.2 的平台上，"Step Over"（步过）的行为有时像 "Continue"（继续）一样。(200196)

  解决方法：尝试较早的 GDB 版本，或在 "Project Properties"（项目属性）的 "Run"（运行）部分中将 "Console Type"（控制台类型）从 "Internal Terminal"（内部终端）更改为其他选项。

- 内存访问错误工具对远程项目不起作用。(7109562)

  如果您在远程主机上创建了一个项目，然后检测项目并对其运行内存分析，则您可能会收到错误消息 "`can't execute: discover: No such file or directory`"（无法执行: discover: 没有这样的文件或目录）。

- 单击 "Debugger Console"（调试器控制台）标签不会将焦点设置到调试器命令提示符处。(7102076)

  解决方法：再次单击此标签设置焦点，以便在提示符处输入命令。

- 使用二进制文件新创建的完全远程项目不显示在 "Projects"（项目）标签中。(7110094)

  如果正在运行 IDE 桌面分发，并使用远程主机上的现有二进制文件创建项目，新创建的项目不会显示在 "Projects"（项目）标签中。

  解决方法：选择 "File"（文件）> "Open Remote C/C++ Project"（打开远程 C/C++ 项目），然后选择要打开的项目。

- 无法使用 SPARC 平台上的二进制文件创建完全远程项目，因为系统不能识别二进制文件。(7109551)

  如果正在运行 IDE 桌面分发，并使用 SPARC 平台的远程主机上的现有二进制文件创建项目，然后选择 "File"（文件）> "Open Remote C/C++ Project"（打开远程 C/C++ 项目），则系统不能识别二进制文件。如果将文件选择器中的过滤器设置为 "All

Binaries"（所有二进制文件），则不会显示二进制文件；如果将过滤器设置为 "All Files"（所有文件），则可以选择二进制文件，但会收到消息 "Binary file not found"（找不到二进制文件）。

dmake

本节介绍了已知的 dmake 软件问题及可能的解决方法。

如果在分布式模式下使用 dmake 出现任何问题，请验证以下内容：

1. $HOME 环境变量应设置为可访问的目录。

   ```
   % ls -la $HOME
   ```

2. 文件 $HOME/.dmakerc 存在且可读，并包含正确的信息。

   ```
   % cat $HOME/.dmakerc
   ```

3. 通过使用 /usr/sbin/ping 命令检查每台主机，确保 $HOME/.dmakerc 文件中提及的所有主机均处于活动状态。

   ```
   % /usr/sbin/ping $HOST
   ```

   其中，$HOST 是系统的名称，它作为主机列于 $HOME/.dmakerc 文件中。

4. 通过使用 dmake、rxm 和 rxs 命令，验证 dmake 二进制文件的路径是否正确。

   ```
   % which dmake
   % which rxm
   % which rxs
   ```

5. 远程登录（rsh 或 ssh）每一台主机时不需要输入口令，并且每次远程登录所花费的时间处于可接受的范围（小于 2 秒钟）。

   ```
   % time rsh $HOST uname -a
   ```

6. 文件 /etc/opt/SPROdmake/dmake.conf 在每台主机中存在并包含正确的信息。如果此文件不存在，dmake 将仅在此系统上分发一个作业：

   ```
   % rsh $HOST cat /etc/opt/SPROdmake/dmake.conf
   ```

7. 对于每台主机，dmake 二进制文件的路径是正确的：

   ```
   % rsh $HOST ‘which dmake‘
   % rsh $HOST ‘which rxm‘
   % rsh $HOST ‘which rxs‘
   ```

8. 可从每台主机获取生成区域 (rwx)：

   ```
   % cd $BUILD
   % rm $HOST.check.tmp
   % echo "Build area is available from host $HOST" > $HOST.check.tmp
   % rsh $HOST cat $BUILD/$HOST.check.tmp
   ```

   其中，$BUILD 是生成区域的完整路径。

9. 可从每台主机获取 $HOME：

```
% cd $HOME
% rm $HOST.check.tmp
% echo "HOME is available from host $HOST" > $HOST.check.tmp
% rsh $HOST cat $HOME/$HOST.check.tmp
```

dmake 限制

您可以将任何计算机作为生成服务器，只要其符合以下要求：

- 对于 dmake 主机（您即将在其中启动生成过程的计算机），您必须在系统不提示输入口令的情况下，能够使用 rsh 或 ssh 在生成服务器上远程执行命令。
- 必须能够从生成服务器访问安装了 dmake 软件的 bin 目录。缺省情况下，dmake 会假设生成服务器上 dmake 可执行文件的逻辑路径与 dmake 主机上的路径相同。您可以通过在运行时配置文件中将路径名称指定为主机条目的属性来覆盖此假设。
- 文件 /etc/opt/SPROdmake/dmake.conf 位于主机上且可读，其中包含正确的信息。如果此文件不存在，dmake 将仅在此系统上分发一个作业。

安装

- 使用 –extract-installation-data 选项运行非 GUI 安装程序可能会失败，但未出现用户可读的错误消息。
- 在某些情况下，如果您在安装目录中运行 register_solstudio 实用程序，它不会生成注册页，也不会在浏览器中打开该页。

  解决方法：

  1. 将 register_solstudio 实用程序从 *installation_directory*/bin 复制到 *installation_directory*/bin/condev/bin。
  2. 使用指向 *installation_directory*/bin/condev/bin/register_solstudio 的符号链接替换 *installation_directory*/bin/register_solstudio。
  3. 运行 register_solstudio 实用程序，它将生成注册页并在浏览器中打开该页。

索引