Oracle® Solaris Studio 12.2 发行版的新增功能

文件号码 821–2518
2010 年 9 月

ORACLE

110209@25097

目录

前言

本指南介绍了 Oracle Solaris Studio 12.2 发行版中的新增功能和更改的功能。

Oracle Solaris Studio 现在是 Sun Studio 编译器和工具软件集合的新名称。本发行版以前的发行版中会保留 Sun Studio 名称。

受支持的平台

此 Oracle Solaris Studio 发行版支持使用 SPARC 和 x86 系列处理器体系结构的系统：UltraSPARC、SPARC64、AMD64、Pentium 和 Xeon EM64T。可从以下位置获得硬件兼容性列表，在列表中可以查看您正在使用的 Oracle Solaris 操作系统版本所支持的系统：`http://www.sun.com/bigadmin/hcl`。这些文档中给出了平台类型间所有实现的区别。

在本文档中，与 x86 相关的术语的含义如下：

- "x86" 泛指 64 位和 32 位的 x86 兼容产品系列。
- "x64" 是指有关 AMD64 或 EM64T 系统的特定 64 位信息。
- “32 位 x86”是指有关基于 x86 的系统的特定 32 位信息。

有关受支持的系统，请参阅硬件兼容性列表。

访问 Solaris Studio 文档

可以访问以下位置的文档：

- 可从以下位置的文档索引页面获得文档：`http://www.oracle.com/technetwork/server-storage/solarisstudio/documentation`。
- IDE、性能分析器、dbxtool 和 DLight 的所有组件的联机帮助可在这些工具中通过“帮助”菜单、F1 键以及许多窗口和对话框上的“帮助”按钮获取。

采用易读格式的文档

该文档以易读格式提供，以方便残障用户使用辅助技术进行阅读。可以按照下表所述找到文档的易读版本。

文档类型	易读版本的格式和位置
手册	HTML，docs.sun.com 上的 Oracle Solaris Studio 12 Update 2 Collection - Simplified Chinese 中
《Oracle Solaris Studio 12.2 发行版的新增功能》（以前的组件自述文件）	HTML，docs.sun.com 上的 Oracle Solaris Studio 12 Update 2 Collection - Simplified Chinese 中
手册页	使用 man 命令显示在 Oracle Solaris 终端中
联机帮助	HTML，可在 IDE、dbxtool、DLight 和性能分析器中通过“帮助”菜单、“帮助”按钮和 F1 键获取
发行说明	HTML，docs.sun.com 上的 Oracle Solaris Studio 12 Update 2 Collection - Simplified Chinese 中

相关第三方 Web 站点引用

本文档中引用了第三方 URL，这些 URL 提供了附加的相关信息。

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、产品或服务而实际或声称造成的或连带产生的损坏或损失，Oracle 概不负责，也不承担任何责任。

开发人员资源

请访问 `http://www.oracle.com/technetwork/server-storage/solarisstudio`，查找以下经常更新的资源：

- 有关编程技术和最佳做法的文章
- 软件文档以及随软件一起安装的文档的更正信息
- 指导您使用 Oracle Solaris Studio 工具执行开发任务的分步教程
- 有关支持级别的信息
- 用户论坛 (`http://forums.sun.com/category.jspa?categoryID=113`)

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:`
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

下表列出了 C shell、Bourne shell 和 Korn shell 的缺省 UNIX 系统提示符和超级用户提示符。

表 P–2 shell 提示符

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

文档、支持和培训

请参见以下 Web 站点了解其他资源：

- 文档 (http://docs.sun.com)
- 支持 (http://www.oracle.com/us/support/systems/index.html)
- 培训 (http://education.oracle.com) – 单击左侧导航栏中的 Sun 链接。

Oracle 欢迎您提出意见

Oracle 欢迎您针对其文档质量和实用性提出意见和建议。如果您发现任何错误，或有其他任何改进建议，请转至 http://docs.sun.com，然后单击 Feedback（反馈）。请提供文档的标题和文件号码，以及章节和页码（如果有）。如果您需要回复，请告知。

Oracle 技术网络 (http://www.oracle.com/technetwork/index.html) 提供了一系列与 Oracle 软件有关的资源：

- 在论坛 (http://forums.oracle.com) 上讨论技术问题和解决方案
- 通过 Oracle By Example (http://www.oracle.com/technology/obe/start/index.html) 获得分步实践教程。
- 下载样例代码 (http://www.oracle.com/technology/sample_code/index.html)。

◆ ◆ ◆ 第 1 章

1 Oracle Solaris Studio 12.2 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。本指南替换了在 Sun Developer Network 门户站点上早期发行版中发布的组件自述文件。

当然，最大的更改是名称：Sun Studio 改为 Oracle Solaris Studio。

什么是 Oracle Solaris Studio？

Oracle Solaris Studio 包含一整套工具，可用于 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 3.0 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 调试器和 dbxtool 调试器 GUI。
- 新增的用于搜索内存泄漏和代码覆盖率的工具：`discover` 和 `uncover`。
- 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 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。
- 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 Technical Network 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

本指南分为几个关于编译器、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 12.2 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 技术网络上的 Solaris Studio 门户站点。

◆ ◆ ◆ 第 2 章

2 编译器

本章介绍了此 Oracle Solaris Studio 发行版中新增和更改的编译器功能。

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。

- 编译器支持 SPARC-V9 ISA 的 SPARC VIS3 版本。通过使用 `-xarch=sparcvis3` 选项进行编译，编译器可以使用 SPARC-V9 指令集中的指令，以及 UltraSPARC 扩展（包括 Visual Instruction Set (VIS) 版本 1.0）、UltraSPARC-III 扩展（包括 Visual Instruction Set (VIS) 版本 2.0、积和熔加运算指令和 Visual Instruction Set (VIS) 版本 3.0）。
- `-xvector` 选项的缺省值已在基于 x86 的系统上更改为 `-xvector=simd`。缺省情况下，在基于 x86 的系统上使用流化处理扩展，这样在优化级别 3 以及更高级别上很有好处。可使用子选项 `no%simd` 禁用该选项。在基于 SPARC 的系统上，缺省值为 `-xvector=%none`。
- 现在提供了对 AMD SSE4a 指令集的支持。使用 `-xarch=amdsse4a` 选项进行编译。
- 手册页已使用 `-xtarget` 值的正确扩展 `ultra3`、`ultra3i`、`ultra3cu`、`ultra4` 和 `ultra4plus` 进行更新。
- 使用新增的 `-traceback` 选项，当出现严重错误时，可执行文件可以输出栈跟踪。此选项使可执行文件能够在退出之前捕获一组信号，并输出栈跟踪和核心转储。如果有多个线程生成了信号，仅为第一个线程生成栈跟踪。要使用追踪，请在将程序与 `f95`、`cc` 或 `CC` 链接时添加 `-traceback` 选项。为了方便起见，也可在编译时接受此选项，但是会将其忽略。使用 `-traceback` 选项和 `-G` 选项创建共享库将发生错误。有关 `-traceback` 选项的详细信息，请参见编译器手册页。
- `-mt` 选项已更改为 `-mt=yes` 或 `-mt=no`。`-mt=yes` 选项可确保以适当的顺序链接库。有关详细信息，请参见编译器手册页。
- 为 C 和 C++ 添加了新的 pragma。有关详细信息，请参见编译器用户指南。
- `#warning` 编译器指令（C 和 C++）在指令中发出文本作为警告，然后继续编译。

- （C 和 C++）头文件 `mbarrier.h` 现在可用。该文件用于在 SPARC 和 x86 处理器上为多线程代码定义多个内存屏障内部函数。有关详细信息，请参见编译器用户指南。
- `-xprofile=tcov[:` *prof_dir*] 选项接受可选的配置文件目录文件名参数。如果指定了配置文件目录文件名，编译的程序将生成可供 `tcov`(1) 或使用 `-xprofile=use:`*prof_dir* 的反馈编译使用的数据。有关详细信息，请参见编译器用户指南。
- 在此发行版中，由 `-xMD` 和 `-xMMD` 选项 (C/C++) 写入的相关文件覆盖了先前存在的文件。文件名派生自 `-o` *filename*（如果已指定）、具有 `.d` 后缀的输入源文件名或者由 `-xMF` 选项指定的文件名。使用 `-xMD` 或 `-xMMD` 选项指定了 `-o` *filename* 或 `-xMF` *filename* 时，仅接受一个源文件。使用此方式编译多个源文件将发生错误。

C 编译器

下面列出了 C 编译器版本 5.11 的此发行版中的新增功能和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.2：C 用户指南》和 `cc` 手册页。

- 对 C 编译器的更改更正了包含复杂类型的 `struct` 在 64 位模式下的 SPARC 处理器上传送和返回的方式。以前，这些 `struct` 值有时会在错误的寄存器中传送和返回，导致创建的二进制文件与 `gcc` 编译器创建的二进制文件不兼容。由于此更改在 Solaris Studio C 编译器中实现时会影响现有 ABI 的元素，因此，如果应用程序中有任何源文件使用具有复杂字段的 struct，**必须重新编译应用程序的整个源代码库，以避免出现错误答案**。针对 32 位 SPARC 处理器、32 位或 64 位 x86 处理器的编译不受此更改的影响。

C++ 编译器

下面列出了 C++ 编译器版本 5.11 的此发行版中的新增功能和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.2：C++ 用户指南》和 `CC` 手册页。

- `-g` 选项带有 `-O` 或 `-xO` 选项但不带有 `+` 选项可产生内联。示例：
  - `CC -g foo.cc` 产生没有内联的可调试 `a.out`
  - `CC -g -O foo.cc` 产生有内联的可调试 `a.out`
  - `CC -g0 foo.cc` 产生有内联的可调试 `a.out`
- C++ 选项 `-xalias_level=compatible` 声明程序符合 C++ 标准的要求。
- 添加对安装在 Oracle Solaris 中的 Apache C++ 库的支持。
- `-compat=g` 选项添加了某些 `gcc` 兼容性。
- `-features=[no%] rvalueref` 选项将编译器对非 `const` 引用的处理恢复为临时值或右值。

Fortran 编译器

下表列出了此 Fortran 编译器版本 8.5 的此发行版中的新增功能和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.2： Fortran 用户指南》和 `f95` 手册页。

- 新增的 **`-xkeepframe`**[**`=`**[**`%all`**,**`%none`**, *name*, **`no%`** *name*] 选项禁止对已命名函数进行与栈相关的优化。**`%all`** 禁止对所有代码进行与堆栈相关的优化。**`%none`** 允许对所有代码进行与堆栈相关的优化。缺省值为 **`-xkeepframe=%none`**。
- F2003 Fortran 标准中的其他功能已实现。
- 为进行优化，**`IVDEP`** 指令要求编译器忽略某些或所有在循环中找到的对数组引用的循环迭代依赖关系。这使得编译器能够执行使用其他方式无法实现的各个循环优化。**`-xivdep`** 选项可用于禁用 **`IVDEP`** 指令或确定应如何解释指令。

OpenMP

下面列出了此发行版中由 C、C++ 和 Fortran 编译器实现的 OpenMP 3.0 共享内存 API 的新增功能和更改的功能。有关详细信息，请参见《Oracle Solaris Studio 12.2： OpenMP API 用户指南》。

- 在 dbx 调试程序中支持 OpenMP 调试。对 dbx 进行了以下改进：
  - 新增用于显示有关 OpenMP 区域、任务和线程集的信息的命令。
  - `print –s`、`thread –info`、`whatis` 和 `where` 命令的扩展
  - 新增 OpenMP 同步事件。
- 自动确定作用域扩展到了任务区域。此功能使程序员无需在并行区域或任务区域中明确确定变量的作用域。编译器通过分析代码和应用一些智能规则来确定变量作用域。
- 新增的 `SUNW_MP_WAIT_POLICY` 环境变量改进了 程在程序中的等待行为，使程序员能够很好地控制等待运行（闲置）、在屏障处等待或者在 `taskwait` 处等待的线程的行为。
- 向 `SUNW_MP_WARN` OpenMP 环境变量中添加了新功能：除了控制由 OpenMP 运行时库发布的警告消息外，`SUNW_MP_WARN` 设为 `TRUE` 时，运行时库还将输出所有环境变量的设置以供参考，包括用户明确设置的环境变量和由库缺省设置的环境变量。
- 由 `SUNW_MP_PROCBIND` 环境变量控制的行为已在 Oracle Solaris 平台上更改：将 `SUNW_MP_PROCBIND` 设为 `TRUE` 会将主线程绑定到绑定时正在其上运行的处理器。绑定的时刻是第一次遇到并行区域或第一次调用 OpenMP 运行时例程（如 `omp_set_num_threads()`）的时刻。从线程以循环（共享）方式从主线程所绑定到的处理器开始绑定。
- 使用线程分析器工具可检测 OpenMP 程序中的数据争用和死锁现象。在此发行版中，线程分析器功能已经扩展，无需重新编译即可检测二进制文件中的数据争用现象。有关详细信息，请参阅《Oracle Solaris Studio 12.2： 线程分析器用户指南》。

◆ ◆ ◆ 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` 现在已过时。库 `libm9x.so.0`, `libmvec.a` 和 `libmvec_mt.a` 也已过时，并已从此发行版中删除。

Sun 性能库

关于 Sun 性能库

此 Sun 性能库发行版可在 Solaris 操作系统版本 10 上使用。它也可在一些 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。性能库包括以下标准库：

- 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，用于解决方程的稀疏线性系统。

Sun 性能库包含了下面更多的例程：

- 快速傅立叶变换 (Fast Fourier transform, FFT) 例程
- 直接稀疏求解器例程
- 区间 BLAS 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 中的内部更改而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
- 每一个区间 BLAS 例程的区间 BLAS 手册页（3pi 部分）
- 《Oracle Solaris Studio 12.2: Sun Performance Library User's Guide》介绍和显示了以下示例：
  - 使用 Sun 性能库例程
  - 使用 Fortran 和 C 接口
  - 使用优化和并行选项
  - 使用 SPSOLVE 和 SuperLU 稀疏求解器软件包
  - 使用 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

- 在此发行版中，为 Fortran 和 C 编译器更改了与 Sun 性能库的链接方式。现在，Fortran、C 和 C++ 均使用 `-library=sunperf` 选项，而不使用 `-xlic_lib=sunperf` 选项。如果想要静态链接，请将 `-staticlib=sunperf` 选项添加到 `-library=sunperf` 选项的后面。
- 在此发行版中，`libsuniperf` (IBLAS) 归类为“已过时”，并已从 Oracle Solaris Studio 中删除。

上一发行版中引入的新增功能和更改的功能

Sun Studio 12 Update 1 发行版在 Sun 性能库中引入了以下功能：

- 现在，Sun 性能库包含 ScaLAPACK 1.8.0 高性能群集库。该库可与基于 OpenMPI 1.3 发行版的 Sun HPC ClusterTools 8.1 一起使用。可以在 `http://www.netlib.org/scalapack/` 上找到参考实现以及文档。
- 新增的定制库工具 (Custom Library Tool) 提供了用于创建 Sun 性能库缩减版本的选项。定制库工具 `gen_custom` 可从归档库中提取例程，然后将这些例程重新合并到定制库中。这可以减少大型库（例如 Sun 性能库）的资源占用大小，以只包含那些用户需要的例程。有关详细信息，请参见 `gen_custom`(3p) 手册页。
- 对 BLAS、LAPACK 和 FFT 例程进行了众多性能改进。
- 目前可以支持 Intel(R) CoreTM i7 (Nehalem) CPU 和 AMD Quad-Core OpteronTM (Shanghai) CPU。要与此库进行链接，请使用以下选项：

  `-m64 -xlic_lib=sunperf`（C 和 Fortran）

  `-m64 -library=sunperf` (C++)
- 目前可以支持 Fujitsu SPARC64-VII(R) CPU。此版本的 Sun 性能库使用浮点乘加指令获取可能的最佳性能。要与此库进行链接，请使用以下选项：

  `-xtarget=sparc64vii -fma=fused -xlic_lib=sunperf`（C 和 Fortran）

  `-xtarget=sparc64vii -fma=fused -library=sunperf` (C++)
- 针对 SPARC64-VI 和 SPARC64-VII 进行了 ZGEMM 改进
- 更新了 LAPACK 例程以符合最新的 LAPACK 3.1.1 规范
- 目前可以支持 Woodcrest CPU。
- 目前可以支持 SPARC64-VI CPU。

对于基于 x86 的系统：

- 在 SuSE Linux Enterprise Server 9 或 Redhat Enterprise Linux 4 操作环境的 32 位和 64 位系统上提供了性能库。

- 目前提供了带有 64 位整型参数的例程。即，Sun 性能库的所有版本中均提供 `DAXPY()` 和 `DAXPY_64()`。
- 提供了稀疏求解器软件包 SuperLU 的系列版本，您可以从 C 驱动程序或通过性能库中基于 Fortran 的现有稀疏求解器调用此版本。
- 目前，四精度例程（`dqdoti` 和 `dqdota`）不可用。
- 在支持 SSE2 和基于 x86 的系统上提供了适用于 Solaris OS 和 Linux OS 的区间 BLAS 例程。

对于 SPARC 处理器：

- 提供了对 UltraSPARC IV+ 和 UltraSPARC IV 处理器的 BLAS 和 FFT 改进。
- 目前可以支持 SPARC64VI CPU。此版本的 Sun 性能库使用浮点乘加指令在 SPARC64VI CPU 上获取最佳的性能。要与此库链接，请使用 `-xtarget=sparcfmaf` 标志进行编译/链接。
- 提供了稀疏求解器软件包 SuperLU 的系列版本，您可以从 C 驱动程序或通过性能库中基于 Fortran 的现有 SPSOLVE 稀疏求解器调用此版本。

◆ ◆ ◆ 第 4 章

4 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 Solaris Studio 性能分析器发行版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 12.2：性能分析器》手册。

对实验格式的更改

扩展了实验格式，但是版本号当前没有更改 (10.1)。

这些工具可以读取使用 Oracle Solaris Studio 12.2 的 FCS 版本创建的实验，也可以读取使用 Studio 12 Update 1 和 Studio 12 的 FCS 和修补版本创建的实验。

无法使用 Oracle Solaris Studio 12.2 工具读取早于 Sun Studio 12 的版本创建的实验。

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。

新增“调用树”选项卡

新增的“调用树”标签以树的形式显示程序的动态调用图，每个函数调用均显示为可以展开和折叠的节点。展开的函数节点显示由该函数生成的所有函数调用，以及这些函数调用的性能度量。当您选中某个节点时，右侧的“摘要”选项卡将显示该函数调用方和被调用方的度量。归属度量的百分比是总程序度量的百分比。

要轻松查找花费时间最多的分支，请右键单击任一节点，然后选择“展开热门分支”。

“调用方-被调用方”选项卡的增强功能

通过将调用方和被调用方添加到调用栈，您可以在中间的“堆栈片段”面板中构建调用栈片段，一次可构建一个调用。调用方是调用该片段的函数；被调用方是从该片段调用的函数。功能包括：

- 当您在栈片段中添加和删除函数时，将为整个片段计算度量，并将结果显示在片段中的最后一个函数旁边。
- 您可以右键单击某个调用方，将函数添加到栈片段的顶部，也可以右键单击某个被调用方，将函数添加到栈片段的底部。也可以使用“栈片段”面板上方的按钮处理调用栈片段。
- 您可以使用位于“栈片段”面板上方的“后退”和“前进”按钮查看您对调用栈片段的更改历史记录。
- 您可以在上下文（右键单击）菜单中的“调用方-被调用方”选项卡中过滤数据。

新增比较试验功能

现在，通过性能分析器，您可以比较在同一可执行文件上收集的实验。此功能仅部分实现，可能会在后续发行版中有所更改。在当前发行版中，比较试验功能按如下所示运行：

- 缺省情况下，如果打开两个或多个实验或实验组，将聚集数据。
- 如果将 `compare on` 添加到您的 `.er.rc` 文件中，并在性能分析器中打开两个或多个实验或实验组，数据将以比较模式显示。
- 在比较模式中，实验或实验组中的数据将显示在相邻列中，另有标题行显示实验或实验组名称。列中填充了颜色，以便区分实验或实验组。
- 支持比较试验功能的选项卡有“函数”、“调用方-被调用方”、“源”、“反汇编”、“行”和"PC"。您可以在其中任意一个选项卡中通过上下文菜单禁用或启用比较模式。
- 您还可以使用“格式”选项卡中的“比较试验”选项，在分析器的“设置数据显示”对话框中启用和禁用比较模式。

其他增强功能

- “源”选项卡中的高亮显示以橙色显示热门（CPU 使用率最高）行，以黄色显示非零度量行。
- 通过“源”选项卡中的上下文菜单，您可以导航到下一个或上一个热门行或非零度量行。
- 您可以通过“输出”菜单创建时间线、MPI 时间线和 MPI 图表的 JPG 文件。
- 热点编译代码的“源”和“反汇编”会利用更好的映射（如果已记录）。

er_print 命令

er_print 命令在此发行版中进行了如下更改：

- 新增的用于控制“调用方-被调用方”列表的命令现在支持调用栈构建。新增的 er_print 子命令 cprepend、cappend、crmfirst 和 crmlast 可从您正在构建的调用栈片段中添加或删除函数。在每个命令后面，会写入当前片段的调用方-被调用方数据。
- 新增的 calltree 命令用于输出目标的动态调用图，该图显示了所有函数的分层度量。
- 新增的 describe 命令介绍了实验中记录的数据，并输出可用于过滤的标记。
- 热点编译代码的“源”和“反汇编”会利用更好的映射（如果已记录）。
- 通过 er_print 命令，现在您可以比较在同一可执行文件上收集的实验。此功能仅部分实现，可能会在后续发行版中有所更改。在当前发行版中，比较试验功能按如下所示运行：
  - 在两个或多个实验或实验组上调用 er_print 时，将聚集数据。
  - 如果将 compare on 放置在您的 .er.rc 文件，并在两个或多个实验或实验组上运行 er_print，数据将以比较模式显示。
  - 在比较模式中，实验或实验组中的数据显示在“函数”列表、“调用方-被调用方”列表和“反汇编”列表中的相邻列中。这些列按照实验或实验组的加载顺序显示，另有标题行显示实验或实验组名称。可使用 compare 命令启用和禁用比较模式。

新增数据收集功能

collect 命令在此发行版中进行了如下更改：

- 遵循后代进程的缺省设置已更改为 -F on。
- 使用 Sun HPC ClusterTools（现在称为 Oracle Message Passing Toolkit）任一发行版的 MPI 实验均可由 -M OMP 或 -M C 指定。
- 缺省情况下，MPI 实验现在也遵循后代进程。
- 改进了 MPI 跟踪实验的后续处理。
- 添加了 Oracle Enterprise Linux 上对硬件计数器分析的支持。
- 改进了硬件计数器别名，并添加了对下列处理器上的硬件计数器分析的支持：
  - SPARC64 VI 和 VII
  - Intel Core i7：系列 6，型号 30、31、37、44 和 46（包括 Nehalem EP 和 EX）
  - AMD 系列 10h 和 11h
- 实现了对分析脚本的实验支持，在后续版本中可能会有所更改。要分析脚本，请设置环境变量 SP_COLLECTOR_SKIP_CHECKEXEC 并将脚本名称传送到 collect。

- 增强了 Java 分析功能，现在可提供有关热点编译代码的源行映射的更多详细信息。JDK 1.6u20 或更高版本的 JDK 1.6 更新以及 JDK 1.7.0-ea-b85 或更高版本的 JDK 1.7 更新现在支持 Java 分析增强功能。
- 实验的缺省大小限制已删除。您可以使用 `-L` 选项设置大小限制。

新增 dbx 收集器功能

对 dbx 调试器的 `collector` 子命令进行了如下更改：

- 改进了硬件计数器别名，并添加了对下列处理器上的硬件计数器分析的支持：
  - SPARC64 VI 和 VII
  - Intel Core i7：系列 6，型号 30、31、37、44 和 46（包括 Nehalem EP 和 EX）
  - AMD 系列 10h 和 11h
- 实验的缺省大小限制已删除。可以使用 `collector limit` 命令设置大小限制。

对 `er_kernel` 的更改

更改了用于分析 Solaris 内核的命令，使 `er_kernel` 在将信号 `SIGINT`、`SIGTERM` 或 `SIGQUIT` 发送到进程时可执行以下操作：

- 捕捉 `SIGINT`、`SIGTERM` 或 `SIGQUIT`
- 终止实验
- 如果没有指定 `-A off`，请运行 `er_archive`

新增命令 `er_generic`

`er_generic` 命令用于从包含配置文件信息的文本文件生成实验。然后，可使用性能分析器或 `er_print` 命令检验模拟的实验。有关更多信息，请参见 `er_generic`(1) 手册页。

对 `en_desc` 的更改

缺省情况下，`en_desc` 命令现在可读取所有后代进程。

线程分析器

现在，线程分析器支持对在源级别或二进制级别校验的代码进行数据争用检测。源级别校验在此发行版中没有更改。

要校验程序的二进制代码，您需要使用 `discover` 工具，该工具包括在 Oracle Solaris Studio 中，在 `discover`(1) 手册页中进行了介绍。另请参见《Oracle Solaris Studio 12.2 Discover 和 Uncover 用户指南》。

要校验程序的二进制代码以检测数据争用现象，`discover` 工具需要在下列条件下编译输入的二进制文件：

- 操作系统版本必须不低于 Oracle Solaris 10 5/08 或 OpenSolaris 版本 snv_70
- 编译器必须来自不低于 Sun Solaris Studio 12 Update 1 的发行版
- 必须使用其中一个编译器优化标志（`-xO1`、`-xO2`、`-xO3`、`-xO4` 和 `-xO5`）

如果二进制文件是使用编译器选项 `-xbinopt=prepare` 编译的，您可能还能够在基于 SPARC 的系统上运行的早期 Solaris 版本上使用 `discover` 工具。有关此编译器选项的信息，请参见 `cc`(1)、`CC`(1) 或 `f95`(1) 手册页。

如果二进制文件名为 `a.out`，您可以使用以下命令创建名为 `a.out_i` 的校验二进制文件：

```
% discover -i datarace -o a.out_i a.out
```

有关详细信息，请参见《Oracle Solaris Studio 12.2：线程分析器用户指南》或 `tha`(1) 手册页。

◆ ◆ ◆ 第 5 章

5 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 12.2 dbx 中新增或更改了以下功能。

- 改进了对优化代码调试的支持：
  - x86 平台上提供了有关查找参数和局部变量的信息。
  - 提供了针对 SPARC 平台的有关内联函数的信息。
- 新增了用于显示有关 OpenMP 区域、任务和线程集的信息的命令：
  - `omp_pr` [*parallel_region_id*] [`-ancestors`|`-tree`] [`-v`]

    输出有关当前并行区域或由 *parallel_region_id* 指定的区域的说明，包括并行区域 ID、类型（隐式或显式）、状态（活动或非活动）、组大小（线程数）和程序位置（程序计数器地址）。
  - `omp_tr` [*task_region_id*] [`-ancestors`|`-tree`]

    输出当前任务区域或由 *task_region_id* 指定的区域的说明，包括任务区域 ID、类型（隐式或显式、绑定或未绑定）、状态（已生成、正在执行或正在等待）、遇到的线程、正在执行的线程、程序位置、未完成的子线程和父线程。
  - `omp_team` [*parallel_region_id*]

    输出当前组中的所有线程。如果已指定 *parallel_region_id*，将输出该区域的组中的线程。
  - `omp_loop`

    输出当前循环的说明，包括调度类型（静态或动态）、等待或非等待、是否有序、界限和迭代数。此命令只能从当前正在执行循环的线程发出。

- `omp_serialize`
  将当前线程遇到的下一个并行区域序列化。

- OpenMP 程序的现有命令的扩展 :
  - `print -s` *expression*
  - `thread -info`
  - `what is` *name*
  - `where`
- 新增的 OpenMP 事件 :
  - `omp_barrier` [*type*] [*state*]
    跟踪线程进入屏障事件。
  - `omp_taskwait` [*state*]
    跟踪线程进入任务等待事件。
  - `omp_ordered` [*state*]
    跟踪线程进入有序区域事件。
  - `omp_critical`
    跟踪线程进入重要区域事件。
  - `omp_atomic` [*state*]
    跟踪线程进入 atomic 区域事件。
  - `omp_flush` [*type*]
    跟踪线程执行刷新事件。
  - `omp_task` [*state*]
    跟踪任务的创建和终止。
  - `omp_master`
    跟踪主线程进入主区域事件。
  - `omp_single`
    跟踪线程进入单个区域事件。

软件更正

本节介绍了此 Oracle Solaris Studio 12.2 dbx 发行版中解决的问题。

1. 执行 tracei 步骤时无法停止 dbx 执行
   在 Solaris 平台上执行 `tracei` 步骤时，用户无法通过键入 control-C (^C) 停止 dbx。这实际上是一个 Solaris OS 错误，但是已修改 dbx 以解决此问题。
2. 无法步入校验过的 debuglog uttsc 二进制文件

dbx 没有在词法块中正确处理 C++ 名称空间别名。这导致了 dbx 无法步入专门校验的二进制文件的问题。

3. 对于使用 Purify 校验的多线程程序，dbx 中未提供与线程相关的命令。

   Purify 会向所校验的所有共享库的名称添加后缀。例如，`libc.so.1` 将变为 `libc.so.1_pure_p3_c0_1005282029_510_32`。dbx 基于是否存在 `libc.so.1`（不再加载）做出了决定。现在，dbx 可理解 `_pure*` 后缀。

4. dbx 无法取消重整特定 GCC 4.x。sybx 无法从 SLES 10.2 上的核心转储文件提取程序名称。

   在 SuSE Linux Enterprise Server 10.2 系统中，dbx 无法从核心转储文件获取程序，由于更新的 Linux 系统的每个核心转储文件包括两个注释部分，第二个注释部分为空。dbx 将从第一个部分中获取名称。

5. dbx 在 gcc 代码中查找构造函数的副本

   gcc 有时会为某个成员生成多个条目，其中包括 DWARF debug.pubnames 部分中的一个条目，一个作为原型的条目以及一个摘要实例。看到原型条目的实例并对其进行处理后，dbx 需要删除该原型条目。

6. dbx 无法从 SLES 10.2 上的核心转储文件中提取程序名称。

   在 SuSE Linux Enterprise Server 10.2 系统中，dbx 无法从核心转储文件获取程序，由于更新的 Linux 系统的每个核心转储文件包括两个注释章节，第二个注释章节为空。dbx 将从第一个部分中获取名称。

7. dbx 在输出变量时发生 SIGSEGV

   从一些不是使用 -g 选项编译的目标文件 (.o) 创建可执行文件，并且 dbx 需要导入这些目标文件之一来计算变量时，由于 dbx 没有检查此条件，因此导入将失败。

8. dbx－如果无效，核心转储文件将发生 segv

   dbx 没有检查是否可能存在长度为零的核心转储文件，也没有正常处理该文件。

9. “内存”和“反汇编程序”窗口可导致 dbx 在 IDE 下崩溃

   如果 IDE 中的调试会话在显示“内存”或“反汇编程序”窗口时结束，然后重新启动，则显示这两个窗口中的任何一个都将导致基础 dbx 崩溃。

dbxtool

dbxtool 中的新增功能和更改的功能如下所示。（有关详细信息，请参见 `dbxtool(1)` 手册页和《Oracle Solaris Studio 12.2 dbxtool 教程》。）

- “选项”窗口中的全局调试选项已重新排列到会话启动属性和窗口属性中。删除了四个属性设置：“程序停止时显示‘Dbx 命令’选项卡”、“保存并恢复断点”、“允许单步启动进程”和“气球表达式计算”。
- “局部变量”窗口现在为“变量”窗口。
- “新建监视”按钮和“表达式计算”按钮已从工具栏中删除。

- 在“新建断点”对话框中：LWP、语言模式和临时复选框已删除。“条件”、“计数”、"WhileIn" 和“线程”字段已重新排序。“更多/更少”按钮已删除。
- 如果有多个调试会话，“会话”窗口将自动打开。
- “调用栈”窗口最多可显示 40 个帧；用户可单击“更多”查看 40 个以上的帧。
- “变量”窗口和“监视”窗口可显示静态成员。
- “变量”窗口中有一个用于仅显示自动变量的按钮。
- “变量”窗口中有一个“新建监视”按钮。
- “反汇编程序”窗口已重命名为“反汇编”。

Discover 和 Uncover

Sun 内存错误搜索工具 (Discover) 是一个用于检测内存访问错误的高级开发工具，是此发行版中新增的工具。

Uncover 是一个易于使用的命令行工具，用于衡量应用程序的代码覆盖率，也是此发行版中新增的工具。

有关详细信息，请参见 `discover` 和 `uncover`(1) 手册页和《Oracle Solaris Studio 12.2 Discover 和 Uncover 用户指南》 。

DLight

DLight 是供 C/C++ 开发人员使用的基于 Oracle Solaris 动态跟踪 (DTrace) 技术的独立交互式图形观察工具，是此发行版中新增的工具。此工具与 Sun Studio 12 Update 1 中包括的 DLight 工具不是同一个工具。请参见《Oracle Solaris Studio 12.2 DLight 教程》 。

◆ ◆ ◆ 第 6 章

6 Solaris Studio IDE

Oracle Solaris Studio 12.2 IDE（Indegrated 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 12.2 IDE 快速入门教程。

新增和更改的功能

Oracle Solaris Studio 12.2 IDE 中新增或更改了以下功能 :

- 基于 NetBeans IDE 6.9
- 通过 Qt 应用程序开发框架，您可以创建 Qt 文件（如 GUI 表单、资源和转换）。
- 运行监视器显示有关应用程序运行时的信息，例如 CPU、内存和线程的使用情况。在 Solaris 平台上，您可以通过线程详细信息和 I/O 使用情况跟踪线程微观状态。
- 现在，调用图包括从选定函数调用的所有函数或调用该函数的所有函数的图形视图及树视图。
- 通过超级链接导航，现在您可以从被覆盖的方法跳到覆盖该方法的方法，反之亦然。
- 您可以向源代码添加注释以生成函数、类和方法的文档。IDE 可识别使用 Doxygen 语法的注释，并自动生成文档。
- “选项”窗口中的全局调试选项已重新排列到会话启动属性和窗口属性中。删除了四个属性设置 : “程序停止时显示‘Dbx 命令’选项卡”、“保存并恢复断点”、“允许单步启动进程”和“气球表达式计算”。
- “局部变量”窗口现在为“变量”窗口。

- “新建监视”按钮和“表达式计算”按钮已从调试工具栏中删除。
- “使调用方成为当前调用方”和“使被调用方成为当前被调用方”按钮已从调试工具栏中删除。
- “重新启动”按钮添加到了工具栏中。
- 在“新建断点”对话框中：LWP、语言模式和临时复选框已删除。“条件”、“计数”、"WhileIn" 和“线程”字段已重新排序。“更多/更少”按钮已删除。
- 如果有多个调试会话，“会话”窗口将自动打开。
- “调用栈”窗口最多可显示 40 个帧；用户可单击“更多”查看 40 个以上的帧。
- “变量”窗口和“监视”窗口可显示静态成员。
- “变量”窗口中有一个用于仅显示当前源代码行和前一源代码行的变量的按钮。
- “变量”窗口中有“新建监视”按钮。
- “反汇编程序”窗口已重命名为“反汇编”。

软件要求

Oracle Solaris Studio IDE 需要 Java SE Development Kit (JDK) 6 Update 13 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

使用 IDE 的插件管理器，您可以动态更新 IDE 的已安装插件。您也可以使用插件管理器将新的插件和功能添加到 IDE 中。

使用插件管理器更新 IDE 时，IDE 会检查注册的更新中心，以查看是否有新增插件或已安装插件的新版本。如果有新增插件或更新插件，则可以使用插件管理器选择、下载和安装这些插件。

或者，您可以选择“帮助”>“检查更新”来打开插件安装程序。插件安装程序将检查已安装插件的更新。如果更新可用，您可以通过安装程序逐步安装更新。

除了缺省的 IDE 更新中心以外，您还可以从多个更新中心中选择，这些更新中心可提供不同类型的插件，如实验性新插件或不再定期发布的旧插件。

通过更新中心更新已安装的插件：

1. 选择“工具”>“插件”打开插件管理器。
2. 单击“更新”标签显示已安装插件的可用更新。
3. 在左窗格中，选择要更新的插件，然后单击“更新”。
4. 完成安装程序中的页面以下载并安装更新。

“更新”标签的左窗格中显示在更新中心中具有可用更新的已安装插件。缺省情况下，IDE 会定期检查注册的更新中心中是否有已安装插件的可用更新。如果左窗格中不显示任何插件，则意味着 IDE 上次检查更新中心时没有可用的更新。

通过更新中心添加新的插件：

1. 选择“工具”>“插件”打开插件管理器。
2. 单击“可用插件”标签显示可用但尚未安装的插件。
3. 在左窗格中，选择要添加的插件，然后单击“安装”。
4. 完成安装程序中的页面以下载并安装插件。

某些插件可能要求重新启动 IDE 才能完成更新过程。

您可以在插件管理器的“设置”标签中设置 IDE 检查更新的频率。您可以单击“重新装入目录”来立即检查更新中心。

配置

NetBeans IDE 6.9 的缺省堆大小为 128 MB。当您要开发小型项目（最多包含 500 个源文件和头文件）时，Oracle Solaris Studio 12.2 IDE 可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“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/netbeans/etc/netbeans.conf 文件中，编辑 netbeans.conf 文件中的 `–J-Xmx` 命令行 Java 启动开关（下面以粗体显示的内容），然后重新启动 IDE。

```
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 Sun 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 Sun 提供的 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 "Tuning JVM Switches for Performance"。

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 7 章

7 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 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《Distributed Make (dmake)》手册。

此发行版中的软件更正

- 已修复的错误：处理内容较长的条件宏时，dmake 将转储核心。
- 已修复的错误：实现 (TXT1/TXT2) 和文档 (TEXT1/TEXT2) 的 DMAKE_OUTPUT_MODE 值不同。现在，dmake 也接受值 "TEXT1" 和 "TEXT2"。
- 已修复的错误："dmake -v" 在 Linux 上输出错误的版本，现在 dmake 输出正确的版本。
- 已修复的错误：Modula 被视为有害，Modula 编译器的旧规则已从 make.rules 文件中删除。
- 已修复的错误：在 KEEP_STATE 模式下的 dmake 内存泄漏

早期发行版中添加的功能：

- dmake 目前集成在 Solaris Studio IDE 中。这意味着，缺省情况下将使用并行模式下运行的 dmake 来生成所有项目。要更改生成模式或更改并行作业的数量，可从 IDE 中来完成：

  1. 从主菜单中选择“工具”->“选项”以打开“选项”对话框。
  2. 在“选项”中，选择 "C/C++" 图标（位于左面板）以便在右面板中显示 C/C++ 选项。
  3. 单击“项目选项”选项卡（右面板）以显示项目选项，然后选择“Make 选项”。
  4. 输入 **`-m parallel -j 24`**
  5. 按“确定”按钮。

  现在，将在并行模式下生成所有项目，并且最多可生成 24 个项目任务。

- `-x SUN_MAKE_COMPAT_MODE=`*compatibility-mode* 命令行选项：

  `-x SUN_MAKE_COMPAT_MODE=SUN`（缺省），与 Solaris `make` 兼容

  `-x SUN_MAKE_COMPAT_MODE=POSIX`，与 POSIX `make` 兼容

  `-x SUN_MAKE_COMPAT_MODE=GNU`，与 GNU `make` 兼容

- 同样，`SUN_MAKE_COMPAT_MODE` 环境变量为用户提供了三个在兼容模式下指定 dmake 行为的相同选项：

  `SUN_MAKE_COMPAT_MODE=SUN`（缺省），与 Solaris make 兼容

  `SUN_MAKE_COMPAT_MODE=POSIX`，与 POSIX `make` 兼容

  `SUN_MAKE_COMPAT_MODE=GNU`，与 GNU `make` 兼容

- 符合 UNIX 2003 标准。Solaris 10 操作系统中的 `dmake` 和 `make` 实用程序均通过了 UNIX 2003 一致性测试 (XPG5)。
- `dmake` 现在包括了对 AMD64 体系结构上 Sun Grid Engine 的支持。
- 可以在 AMD64 体系结构上实现系统重载控制功能。
- `DMAKE_OUTPUT_MODE` 环境变量提供了两个日志文件的格式选项，其中一个选项可将并行任务的输出序列化，以便日志文件更易于阅读。

Solaris Studio 安装程序

安装程序中的新增功能和更改的功能包括：

- 您可以选择安装的组件列表已经更改。现在，可选组件包括：
  - 编译器支持文件（C 和 C++ 编译器所需要的）
  - C 和 C++ 编译器
  - Fortran 编译器
  - dbx 调试器
  - dbxtool

  - dmake
  - DLight 观察工具（仅限 Solaris 平台）
  - IDE
  - 性能和线程分析器工具（仅限 Solaris 平台）
  - 性能库（仅限 Solaris 平台）
  - ScaLAPACK（仅限 Solaris 平台）
- 现在，您可以通过使用 –libraries-only 选项启动 GUI 安装程序或非 GUI 安装程序，仅安装运行时库。此选项将安装 C++ 库、Fortran 90 库和数学库（仅限 Solaris 平台）。
- 如果要在具有多个区域的系统上进行安装，您现在可以使用 GUI 安装程序中的复选框指定仅在当前区域安装，也可以在启动非 GUI 安装程序时通过命令行指定仅在当前区域安装。
- 现在仅在非 GUI 安装程序中支持备用根目录安装，在 GUI 安装程序中不支持。
- 安装程序不会要求您接受许可证。
- 如果系统上已安装早期 Sun Studio 发行版，将不再限制可运行安装程序的区域。

◆ ◆ ◆ 第 8 章

8 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。

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

与 —xprofile 相关的已知问题

- 如果同一进程中加载了同一目标文件的不同版本，`libxprof` 将失败。

  如果在同一目录中使用相同名称生成了同一文件的两个不同版本，将其链接到两个不同的共享库，并加载到同一进程（可能不是同时），将发生此情况。

  解决方法：使用 `-xprofile` 生成共享库时，确保目标文件名称具有不同的 UNIX 路径名。请注意，即使基本名称相同，路径名也可能不同。例如，

  ```
  /work/mylib/unshared/x.o
  /work/mylib/shared/x.o
  ```

  视为不同。

- OMP：`libxprof`：断言失败

  如果 `malloc()` 在调用分析运行时例程期间失败，在低内存情况下可能会发生断言失败。

  解决方法：添加内存或交换空间。

- `-xprofile=tcov:`*prof_dir* 错误地解析了相关 *prof_dir*

  在 `–xprofile=tcov:`*dir* 下，会相对于将生成目标文件的目录解析非绝对 UNIX 路径名。

解决方法：在 –xprofile={collect,use,tcov}:*dir* 下使用绝对路径名。

C++

更正大型十进制整型常量的解释

C++ 标准规定，无后缀的十进制整型常量当作 int 处理（如果值符合 int），否则当作 long int 处理。如果值不符合 long int，结果将不确定。

在 32 位模式中，类型 int 和 long 具有相同的大小和数据范围。C++ 编译器遵循 1990 C 标准规则，将 INT_MAX+1 和 LONG_MAX 之间的值当作无符号的 long 处理。这种处理在某些程序中会产生意外的结果。

1999 C 标准更改了无后缀的十进制整数的规则，这样它们就永远不会被当作无符号的类型来处理。类型为 int、long 或 long long 中第一个可以表示值的类型。

C++ 编译器在标准模式中遵循 C99 规则，但在 -compat=4 模式中继续遵循 C90 规则。（在 -compat=4 模式中，编译器的行为类似于 C++ 4.2 编译器。）

如果您希望大型十进制整数被当作无符号的整数，则简便的解决方法是使用 u 或 U 后缀。同样，可以对其他类型使用其他后缀。例如：

```
// note: 2147483648 == (INT_MAX+1)
2147483648     // (signed) long long
2147483648LL   // (signed) long long
2147483648U    // same as 2147483648u
```

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
```

不再忽略模板语法错误

下面的模板语法是无效的，但 Sun C++ 编译器 4 和 5.0 版并不报告这个错误。在标准模式（缺省模式）下编译时，C++ 编译器 5.1 版本及以后的所有版本都会报告这个语法错误。

```
template<class T> class MyClass<T> { ... }; // definition error
template<class T> class MyClass<T>; // declaration error
```

在这两种情况下，MyClass<T> 中的 <T> 无效，必须删除，如下例所示：

```
template<class T> class MyClass { ... }; // definition
template<class T> class MyClass; // declaration
```

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。

跨语言链接错误

`-xlang=f77` 命令行选项导致编译进程遇到链接程序错误。要避免该错误，并仍包含相应的运行时库，应改为使用 `-xlang=f77,f90` 进行编译。

名称重整链接问题

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，但编译器会将其重整为两个不同的名称。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，并且应该重整相同的名称。但是，编译器只会重整部分声明的名称。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，某些重整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，会出现两个版本的重整名称，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。

调试工具错误地报告成员函数有多余的前导参数

在兼容模式 (`-compat`) 下，C++ 编译器会错误地重整成员函数指针的链接名称。此错误会导致 demangler 以及其他一些调试工具（例如 dbx 和 c++filt）报告该成员函数有多余的前导参数，该参数包括对该成员函数所在类的引用。要更正此问题，请添加

-Qoption ccfe -abiopt=pmfun1 标志。请注意，使用这个标志进行编译的源代码可能会与不使用此标志编译的源代码在二进制上不兼容。在标准模式（缺省模式）下，不会出现该问题。

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 需要重整名称

在名称空间内使用 #pragma align 时，必须使用重整名称。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 替换为其重整名称。

```
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;
}
```

函数重载解决方案

早期 C++ 编译器发行版并不支持 C++ 标准要求的函数重载。当前发行版修正了调用重载函数时出现的许多错误。具体来讲，当函数调用确实出现多义性时，编译器有时会选择某个函数；或者在函数调用实际上未产生多义性时，编译器会发出警告，指出此调用具有多义性。

采用某些解决方法来避免多义性消息已经没有必要了。也许您会看到以前未报告的新多义性错误。

导致函数调用多义性的一个主要原因在于仅重载内置类型的子集。

```
int f1(short);
int f1(float);
...
f1(1); // ambiguous, "1" is type int
f1(1.0); // ambiguous, "1.0" is type double
```

要解决此问题，要么根本不重载 f1，要么重载所有未提交的类型：int、unsigned int、long、unsigned long 和 double。（也许还有 long long、unsigned long long 以及 long double 类型）。

导致多义性的另一个主要原因在于：类中的类型转换函数，尤其是当您也重载了运算符或构造函数时。

```
class T {
public:
        operator int();
        T(int);
        T operator+(const T&);
};
T t;
1 + t // ambiguous
```

该操作有多义性是因为它可能被处理成：

```
  T(1) + t     // overloaded operator
 1 + t.operator int()    // built-in int addition
```

可以提供重载运算符，也可以提供类型转换函数，但同时提供它们则会产生多义性。

实际上，类型转换函数自身也经常会导致多义性，而且往往在不应该进行转换时进行转换。如果您确实需要类型转换，最好使用命名的函数，而不是类型转换函数。例如，使用 `int to_i t();`，而不是 `operator int();`。

更改后，`1 + t` 操作就不再具有多义性了。它只能解释为 `T(1) + t`。如果您希望有其他的解释，必须写入 `1 + t.to_int()`。

Fortran

在此 `f95` 编译器发行版中应注意以下问题：

- 在此发行版中删除过时的 FORTRAN 77 库的操作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。
- 使用增强的数组构造函数设置参数常量（类型为指定长度的字符）将导致字符元素的值串联在一起。解决方法是使用数组构造函数中所使用的字符长度（而不是指定长度）来定义参数常量。
- 使用空白名称指定 C 绑定过程是错误的处理方式，将导致为该过程使用空白名称。解决方法是指定 C 绑定名称，如果不需要，也可以不使用 C 绑定名称。

f95 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 f95 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：

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目标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，则产生的目标文件不会包含对模块中定义的符号的引用。仅当引用目标文件中定义的符号时，链接程序才会链接归档文件中的目标文件。如果不存在此类引用，目标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` 的目标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_`。

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 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 代码时可能会崩溃

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

4. 如果调试应用程序与生成应用程序时所用的 J2SE 不同，dbx 会抛出异常。

   如果调试应用程序与生成应用程序所用的 J2SE 技术版本不同，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，当运行时检查 (runtime checking, RTC) 检测到对 RTC 开始监视内存分配之前分配的与线程有关的内部数据的访问时，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 12.2 dbx 有以下限制：

- 在基于 x86 系统的 Linux 操作系统上，无法使用 dbx 的以下功能：
  - 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`
- 在基于 x64 系统的 Linux 操作系统上，无法使用 dbx 的以下功能：
  - Java 调试
  - 调试 32 位程序（除非使用 `-x exec32` 选项启动 dbx）。
- 当调用 `exec()` 时，dbx 无法在 Linux 平台上跟踪派生进程，或对新程序执行更改。
- 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：
  - 键入 Ctrl-C 组合键以显示新的 dbx 提示符。
  - dbx 将缓存大量信息，因此，对于上面的示例，您可以使用下面的命令序列：

    ```
    where
    where | head –1
    ```
- 在 Linux 平台上调试程序时可能会发生下面的问题：
  - 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。当使用新的线程库 (NPTL) 时，这种情况发生的次数会减少。

  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。

- C++ 表达式的运行时类型信息不适用于 g++ 编译器编译的程序。
- 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 无法正确地还原指向 compat=4 的成员函数的指针。如果使用 compat=5，则不会出现此问题。

  解决方法：按如下方法重新编译程序：

  ```
  CC –compat=4 –Qoption ccfe –abiopt=pmfun1
  ```

  该标志引入了 ABI 更改并且不应该在产品生成中使用。
- 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。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。
- 使用调用命令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。

性能分析器

如果要在 Linux 上使用 Oracle Message Passing Toolkit 8.2 或 8.2.1，可能需要解决方法。版本 8.1 或 8.2.1c 不需要解决方法，或者如果要使用 Oracle Solaris Studio 编译器，则任何版本都不需要解决方法。

Oracle Message Passing Toolkit 版本号由安装路径指定，例如 `/opt/SUNWhpc/HPC8.2.1`，或者，您可以按照如下所示键入 `mpirun –V` 查看输出，其中版本以斜体表示：

```
mpirun (Open MPI) 1.3.4r22104-ct8.2.1-b09d-r70
```

如果您的应用程序是使用 GNU 或 Intel 编译器编译的，并且要对 MPI 使用 Oracle Message Passing Toolkit 8.2 或 8.2.1，则要获取 MPI 状态数据，必须使用 `-WI` 和 `--enable-new-dtags` 选项和 Oracle Message Passing Toolkit `link` 命令。这些选项将使可执行文件定义 `RPATH` 及 `RUNPATH`，从而可使用 `LD_LIBRARY_PATH` 环境变量启用 MPI 状态库。

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`) 每一台主机时不需要输入密码，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密码的情况下，能够使用 rsh 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，并且不显示用户可读的错误消息。

索引