Oracle® Solaris Studio 12.4 新增功能

文件号码 E57205
2014 年 12 月

ORACLE

目录

使用本文档

- **概述** – 介绍了此 Oracle Solaris Studio 12.4 发行版的编译器和工具中的新增和更改的功能
- **目标读者** – 应用程序开发者、系统开发者、架构师、支持工程师
- **必备知识** – 编程经验、软件开发测试以及构建和编译软件产品的能力

产品文档库

有关编译器和工具的更多信息，请参见相关手册页或 "Help"（帮助）以及 http://docs.oracle.com/cd/E37069_01 中的文档库。

系统要求和已知问题包括在《Oracle Solaris Studio 12.4：发行说明》中。

获得 Oracle 支持

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

反馈

可以在 http://www.oracle.com/goto/docfeedback 上提供有关本文档的反馈。

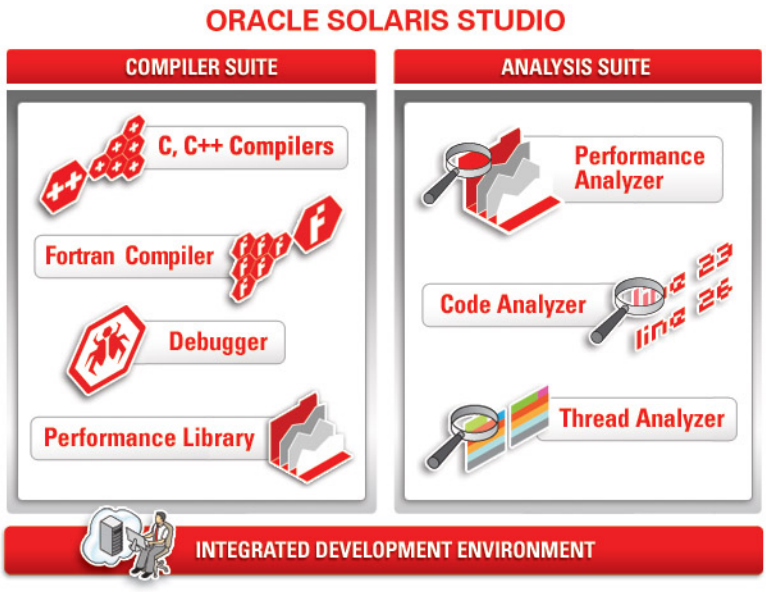
♦ ♦ ♦ 第 1 章

Oracle Solaris Studio 12.4 发行版简介

本章概述了此发行版中的主要更新。

Oracle Solaris Studio 概述

Oracle Solaris Studio 包括编译器套件、分析套件以及图形集成开发环境 (integrated development environment, IDE)，该环境已调整以与那两个套件中的编译器和工具配合使用。它们一起提供一个开发环境，已进行了优化来开发在 Oracle Sun 硬件上具有最佳性能的应用程序。



此发行版中的主要功能

Oracle Solaris Studio 12.4 包括对其所有编译器和工具的增强功能。主要功能体现在以下方面：

C++ 编译器	对 C++11 标准的支持。请参见第 2 章 C++ 编译器。
性能分析器	重新设计的图形界面，可以更容易地导航和了解性能数据并指定要分析的应用程序部分。 可以在 Linux、Windows 或 Mac 客户机上运行性能分析器以及远程访问 Oracle Solaris 或 *Linux* 系统上的性能数据。 请参见第 3 章 性能分析工具。
代码分析器	静态分析过程中在消除误报方面进行了显著改进。 减少了收集运行时数据时的开销时间。 请参见第 4 章 代码分析工具。
调试器	处理大的二进制文件时更快的 `dbx` 启动时间。 减少了调试信息大小。 请参见第 5 章 调试工具。
IDE	显著减少了 IDE 中的内存占用，所以可以更快地打开、搜索和修改大的项目。 使大的项目更易于用于版本控制系统的项目设置。 请参见第 6 章 Oracle Solaris Studio IDE。
针对新服务器的优化	编译器和库优化，以实现 Oracle Sun 硬件服务器的应用程序性能改进：SPARC T5、SPARC M5、 SPARC M6、SPARC M10、SPARC M10+、Intel Haswell 和 Intel Ivy Bridge。请注意，在特定于 Oracle Solaris Studio 12.3 平台的增强功能发行版中已经引入了这些改进中的一些改进。 请参见“新硬件上的应用程序性能” [63]。
OpenMP 4.0	在 OpenMP API 版本 4.0 中实现了新功能，对 OpenMP API 标准语言规范的主要升级。 请参见第 7 章 OpenMP API 和线程分析器。

♦ ♦ ♦ 第 2 章

2 C++ 编译器

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

关于 C++ 编译器

使用 Oracle Solaris Studio C++ 编译器，可以为 Oracle 的基于最新 SPARC 处理器的系统以及基于 Intel x86 处理器的系统构建高性能的并行 C++ 应用程序。

C++ 编译器 (`CC`) 根据指定的命令行选项，生成面向特定操作系统、处理器、体系结构、内存模型（32 位和 64 位）、浮点算法等等的代码。该编译器会自动将序列源代码并行化以生成在多核系统上有更好性能的二进制文件，并且还可以准备二进制文件以便其他 Oracle Solaris Studio 工具更好地进行调试或分析。该编译器还支持 GNU C/C++ 兼容功能。

支持 C++11 标准

新 C++11 标准增强了 C++，提供了其他工具来使您的代码更准确且更安全。编译器在 Oracle 硬件上保持加速的 SPARC 和 x86 性能。auto、smart pointers、nullptr、range for、nonmember begin and end、lambda 函数和算法、rvalue 引用、"move" 构造函数、统一初始化和初始化程序列表等功能应该可以更改 C++ 库的设计方式。

这是支持 C++11 标准的第一个 Oracle Solaris Studio 发行版。除了下面列出的功能，此发行版中支持 C++ 11 的其他所有功能：

不支持下列功能：

- C++ 11 并发性和原子操作

- 用户定义的文字

使用 C++11 功能

在 Oracle Solaris Studio 12.4 中，C++ 编译器支持 C++11，一种新语言和 ABI（Application Binary Interface，应用程序二进制文件接口）。

在 C++ 11 模式下，CC 编译器使用 g++ ABI 和随 Oracle Solaris Studio 提供的 g++ 运行时库版本。对于此发行版，使用 4.8.2 版本的 g++ 运行时库。

ABI 描述生成的对象代码中的低级别详细信息。使用不同 ABI 的模块无法成功链接在一起成为程序。这意味必须在程序中的所有模块上使用 C++11 模式，或者不在任何模块上使用该模式。

如果使用 Oracle Solaris Studio 12.4 C++ 作为对 Oracle Solaris Studio 12.3 (C++ 5.12) 的升级，则不使用 C++11 功能时不需要对脚本或 makefile 进行更改。一个异常是 Rogue Wave Tools.h++ 不可用。有关不再支持的功能的更多信息，请参见《Oracle Solaris Studio 12.4：发行说明 》中的“此发行版中已经删除的功能”

要在 C++11 模式下进行编译，请将选项 `-std=c++11` 添加到 `CC` 命令行。命令行上的位置并不重要。通过该选项，编译器可以识别 C++11 中的新语言功能，并使用标准库（提供的 g++ 运行时库）的 C++11 版本。除了标记为与 `-std=c++11` 不兼容的选项，所有其他命令行选项都可以与 C++11 一起使用并具有其通常所有的效果。必须在用于构建库或可执行程序的每个 `CC` 命令上一致地使用 `-std=c++11` 选项。

注 - 缺省情况下无法使用 C++11 功能。要使用 C++11 功能，必须将新的 `-std=c++11` 选项与 `CC` 编译器配合使用。此选项使用 g++ ABI，不可以选取不同的 ABI。必须使用该选项来编译程序的所有模块。

此发行版中 C++11 的兼容性信息

此发行版中的 C++ 编译器具有以下更新版本详细信息。

编译器版本： C++ 5.13

编译器版本宏： `__SUNPRO_CC = 0x5130`

编译器版本宏完全大于所有早期发行版，所以 `__SUNPRO_CC>=0x5100` 等版本比较将继续运行。

编译器缺省模式： 具有 `-compat=5` 的 C++03。

这是与 Oracle Solaris Studio 12.3 发行版中的 C++ 5.12 相同的缺省模式。

Oracle Solaris Studio 12.3 具有以下编译器模式选项：

`-compat=5` 此选项选择 C++03 和 Sun ABI。这是缺省值。

`-compat=g` 此选项选择 C++03 和 g++ ABI，使用随编译器提供的 gcc 头和库。在 Oracle Solaris Studio 12.3 中，gcc 运行时库安装在 Oracle Solaris 上的 `/usr/sfw/lib` 中（如果存在的话）。在此发行版中，将使用随编译器提供的 gcc 运行时库。

该发行版添加了选项 `-std=[ c++11 | c++0x | c++03 | sun03]`，其中选项值定义如下：

`-std=c++11` 此选项选择 C++11 和 g++ ABI，并使用在 Oracle Solaris Studio 12.4 中安装的 g++ 4.8.2 运行时库。

`-std=c++0x` 此选项等效于 `-std=c++11` 选项，为 GCC 兼容性而提供。C++11 标准最初昵称为 C++0x。

`-std=c++03` 此选项等效于 `-compat=g` 选项。

`-std=sun03` 此选项等效于 `-compat=5`

注 - 不能混合 `-compat` 和 `-std` 选项，如果同时使用两者将产生错误。如果多个 `-std` 选项或多个 `-compat` 选项显示在命令行上，指定的最后一个选项将覆盖先前指定的选项。例如：

```
-compat=g   -compat=5   // OK, -compat=5 is used
-std=c++11  -std=c++03  // OK, -std=c++03 is used
-std=c++11  -compat=g   // always an error
-compat=g   -std=c++03  // always an error
```

---

16 位 Unicode 与 C++11 的不兼容性

选择 `-xustr=ascii_utf16_ushort` 与 C++11 不兼容，不允许使用。

该选项将 `U"`*ASCII_string*`"` 解释为 16 位 Unicode，但对于该语法 C++11 需要 32 位 Unicode。

此发行版中库与 C++11 的不兼容性

对于 `-std=`*x* 或 `-compat=g`，不允许使用以下 `-library=`*v* 选项：

- `Cstd`
- `stlport4`
- `stdcxx4`

- Crun
- iostream

需要列出要链接的库时，例如创建共享库时，如果使用 -compat=g 或 -std=*x* 选项，则按此顺序使用以下选项：

```
-lstdc++ -lgcc_s -lCrunG3
```

使用 CC 创建可执行程序时，不应列出这些库，因为 CC 驱动程序将为您列出它们。

有关此发行版中删除的库的信息，请参见《Oracle Solaris Studio 12.4：发行说明 》中的"此发行版中已经删除的功能"。

使用 C++11 模式的示例

以下命令可用于从模块 main.cc、f1.cc 和 f2.cc 构建可执行程序 myprogram：

```
% CC -std=c++11 -m32 -O -c main.cc
% CC -std=c++11 -m32 -O -c f1.cc f2.cc
% CC -std=c++11 -m32 -O main.o f1.o f2.o -o myprogram
```

如果具有使用较旧版本的 Oracle Solaris Studio 构建 C++ 程序的 Makefile，可以通过以下方法将其转换为构建 C++11 程序：向每个 CC 命令行添加 -std=c++11（通常在 CCFLAGS 和 LFLAGS 宏中）并删除任何不兼容的选项，例如 -compat=5 或 -library=stlport4。有效 C++ 程序在 C++11 模式下编译时通常会在不更改的情况下编译和运行。不过，许多实际的程序有时会偶尔依赖于非标准的编译器行为或扩展。此类代码可能无法在 C++11 模式下编译。

其他 C++ 编译器更改

下面列出了此 5.13 发行版中特定于 C++ 编译器的新增和更改的功能。有关更多信息，请参见 CC(1) 手册页。

C++ 编译器更改包括"编译器通用的新增和更改的功能" [63]中介绍的更改。

有关详细信息，请参见《Oracle Solaris Studio 12.4：C++ 用户指南 》和 CC 手册页。

- 在所有平台上支持 -compat=g。
- 通过新编译器选项：-std，可以选择 C++ 03 或 C++ 11 语言变种并具有 g++ 二进制文件兼容性。使用 -std=c++11 时，存在以下限制：
  - 当前不支持通用字符名称（转义的 Unicode 字符）。
  - 以 .h 结尾的非标准 iostream 标题（例如 <iostream.h>、<fstream.h> 等）不可用。这些标题用于促进从旧样式的 C++ 转换为 C++98。

- 新编译器选项：`-features=[no%]rtti` 可以禁用运行时类型标识 (runtime type identification, RTTI)。
- 新编译器选项：`-xprevise` 生成可以在代码分析器中查看的源代码的静态分析。
- 具有 `-x`*option* 等效项的以下选项现在已弃用：

  ```
  -help
  -inline
  -libmieee
  -libmil
  -nolib
  -nolibmil
  -pg
  -time
  -unroll
  ```

  相反，您应该使用 `-xhelp`、`-xinline`、`-xlibmieee` 等。
- 在 x86 上支持 `-xregs=float`。
- `-errtags` 的行为现在和 C 编译器一样，但仅针对警告消息显示标记。在前面的 C++ 编译器中，`-errtags` 选项导致在警告和错误消息中输出一个标记。
- 缺省 `-template` 选项从 `-template=extdef` 更改为 `-template=no%extdef`。

  此更改是因为没有其他编译器使用由 `-template=extdef` 假定的 `definitions separate` 模板模型。`-template=extdef` 选项对如何组织源代码有严格要求，大多数代码不遵循该要求。除非仅使用 Oracle Solaris Studio C++ 进行开发，否则可能需要 `-template=no%extdef` 选项。

  有关更多信息，请参见《Oracle Solaris Studio 12.4：C++ 用户指南 》中的第 6 章 “创建和使用模板”和《Understanding the Effects of the Changed Default C++ Template Compilation Model (http://www.oracle.com/technetwork/articles/servers-storage-dev/changed-default-cpp-template-model-2292727.html)》（《了解更改的缺省 C++ 模板编译模型的效果》）。

注 - `--library=stdcxx4` 选项当前不与 `-template=no%extdef` 配合使用。如果使用 `-library=stdcxx4` 选项，则在命令行上编译 C++ 代码时指定 `-template=extdef`，直到提供了库的修补程序。

编译器执行更严格的 C++ 规则

C++ 编译器比过去的编译更严格地执行一些 C++ 规则。有关更严格的执行说明、违例代码以及用于更正违例代码的解决方法，请参见《Oracle Solaris Studio 12.4：C++ 用户指南 》中的“Oracle Solaris Studio 12.4 C++ 5.13 编译器的新特性和新功能”。

♦ ♦ ♦ 第 3 章

3 性能分析工具

性能分析工具配合使用，可供您分析应用程序的行为和找到影响性能的问题症结。

本章介绍此 Oracle Solaris Studio 发行版中性能分析工具的新增和更改的功能。

关于性能分析器

利用性能分析器可以深入了解应用程序的行为，从而能够找出代码中的问题方面。性能分析器可确定哪些函数、代码段和源代码行占用的系统资源最多。性能分析器可以分析单线程、多线程和多进程应用程序，然后提供分析数据以帮助确定可提高应用程序性能的方面。

性能分析器由一组命令和工具构成，包括收集分析数据的 `collect` 实用程序，以文本格式显示解释性分析信息的 `er_print` 实用程序，以及以图形方式显示分析信息的性能分析器 GUI。

线程分析器是一款可让您专注于多线程问题的相关工具。有关更多信息，请参见“线程分析器” [60]。

性能分析器文档

本发行版包括新的《Oracle Solaris Studio 12.4：性能分析器教程》手册，该手册包含有关在性能分析器中分析样例应用程序和检查数据实验的逐步说明。

《Oracle Solaris Studio 12.4：性能分析器》参考手册包含有关性能分析器 `collect`、`er_print` 以及其他实用程序的详细信息。

性能分析器新功能

本节概述了此发行版性能分析器和相关工具的新功能。有关更多信息，请参见性能分析器中的 "Help"（帮助）以及每个命令行工具的手册页。

- 性能分析器 UI 经过重新设计后具有新的导航功能、新 "Overview"（概述）和 "Welcome"（欢迎）屏幕。请参见“用户界面重新设计” [18]。
- 时间线功能改进了数据显示、导航和过滤。请参见“"Timeline"（时间线）改进” [21]。
- "Source"（源）和 "Disassembly"（反汇编） 数据视图突出显示语法并提供用于导航的超链接。有关更多信息，请参见“"Source"（源）和 "Disassembly"（反汇编）的改进” [22]。
- "Call Tree"（调用树）视图有新的图形指示器和导航改进。有关更多信息，请参见“"Call Tree"（调用树）方面的改进” [23]。
- 分析 Java 应用程序变得更易执行。请参见“Java 分析改进” [24]。
- 改进了用于空间分析的数据收集和显示。请参见“内存空间分析方面的改进” [25]。
- 可在硬件计数器实验中计算新度量。有关更多信息，请参见“新的派生度量：CPI 和 IPC” [26]。
- 在任意平台上记录的实验均可使用在任何其他平台上运行的性能分析器或 `er_print` 实用程序读取。请参见“跨平台分析” [27]。
- 现在，性能分析器可从 Windows 或 Mac 计算机远程使用，也可以从没有完整的 Oracle Solaris Studio 安装的 Oracle Solaris 或 Linux 计算机远程使用。请参见“性能分析器的远程使用” [27]。
- 这些工具现在支持目标程序的 I/O 跟踪。请参见“新的 I/O 数据视图” [29]。
- 新的 "Heap"（堆）视图改进了堆跟踪数据的显示，有助于找到内存泄漏。请参见“新建堆视图” [30]。
- “性能分析器的其他更改” [31]

有关详细信息，请参见性能分析器中的 “Help”（帮助）。

用户界面重新设计

性能分析器用户界面功能改进了数据显示和导航。UI 更改的一些亮点包括：

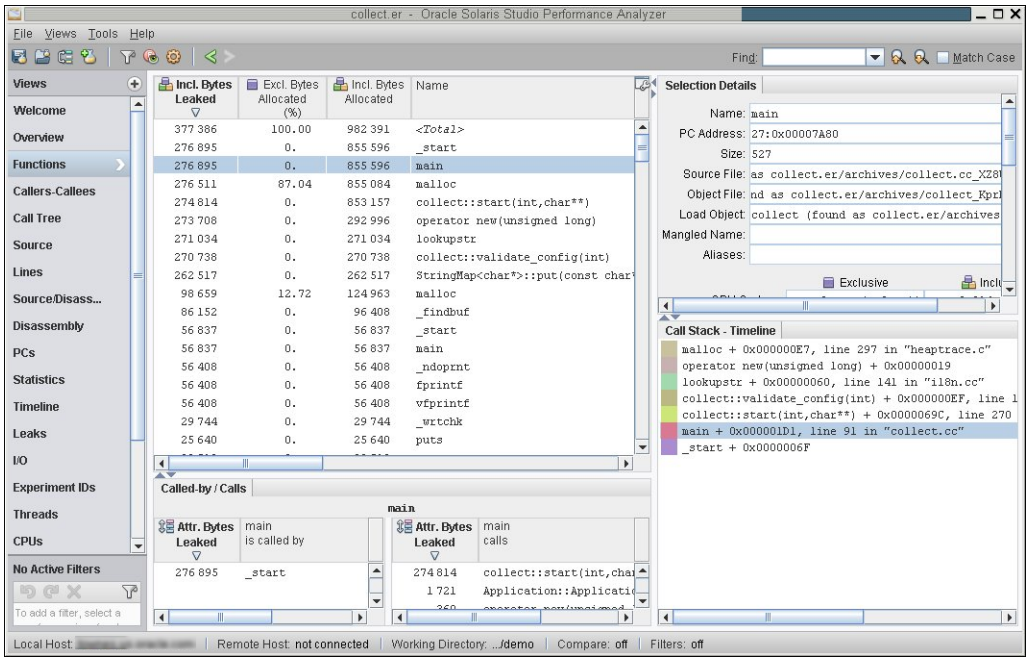
- 性能数据现在显示在通过左侧导航栏访问的数据视图中。此方式取代了以前使用的水平标签导航。有关更多信息，请参见“性能分析器导航” [18]。
- 不指定实验名称的情况下启动性能分析器时将显示新的 "Welcome"（欢迎）屏幕。有关更多信息，请参见“性能分析器的 "Welcome"（欢迎）屏幕” [19]。
- 打开实验时，新的 "Overview"（概述）屏幕显示为数据的第一个视图。有关更多信息，请参见“性能分析器 "Overview"（概述）屏幕” [20]。

性能分析器导航

性能分析器现在围绕着数据视图进行组织，这些数据视图可以从性能分析器窗口的左侧导航栏中访问。每个视图都针对分析的应用程序从不同的角度显示性能度量。与之前一

样，"Functions"（函数）视图是一个焦点，当选择一个函数时，其他数据视图也会更新以重点显示有关所选函数的信息。

图 3-1 选中了 "Functions"（函数）视图的垂直导航区域



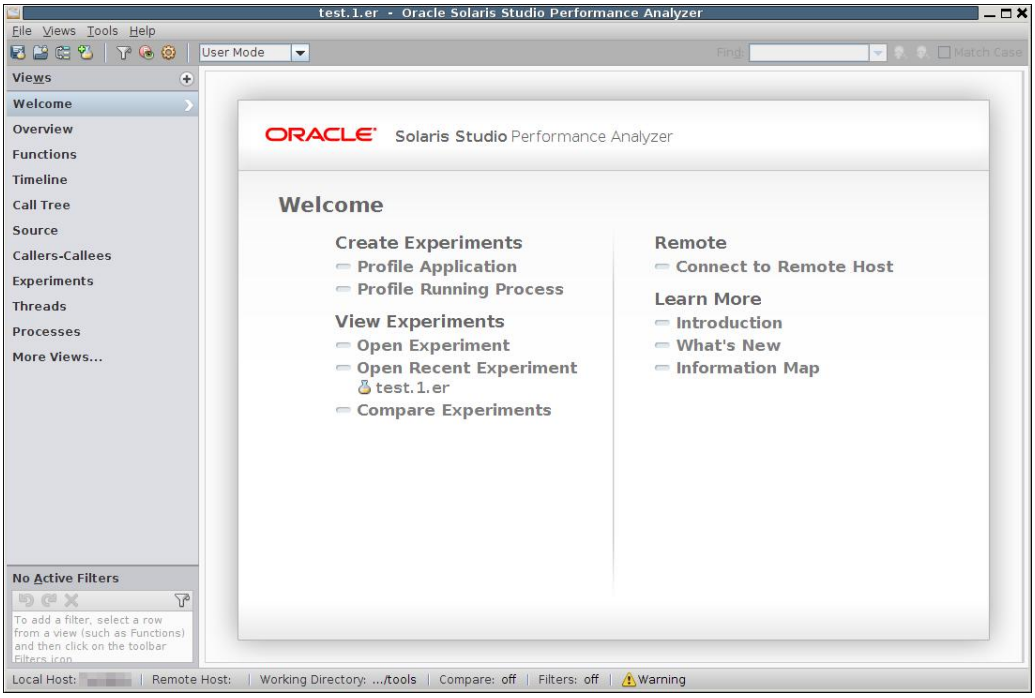
导航栏中可用的数据视图取决于您打开的实验中可用的数据。使用 "Views"（视图）菜单或导航栏上的 + 按钮可以轻松地添加和删除数据视图。您在数据视图中所选的项目的相关详细信息显示在右侧，如上图所示。

在导航栏下方，活动过滤器区域显示当前应用的数据过滤器的名称，您可在此区域删除和重新应用过滤器或删除所有过滤器。通过右键单击数据视图中的某个项并从弹出列表中选择过滤器，或通过单击工具栏上的过滤器图标，可以应用过滤器。

性能分析器的 "Welcome"（欢迎）屏幕

当您启动性能分析器且不指定实验名称时，将显示一个新的 "Welcome"（欢迎）屏幕。"Welcome"（欢迎）屏幕提供了一些链接，让您可以更轻松地分析应用程序，查看最近的实验，浏览实验和查看文档。

图 3-2 性能分析器的 "Welcome"（欢迎）屏幕



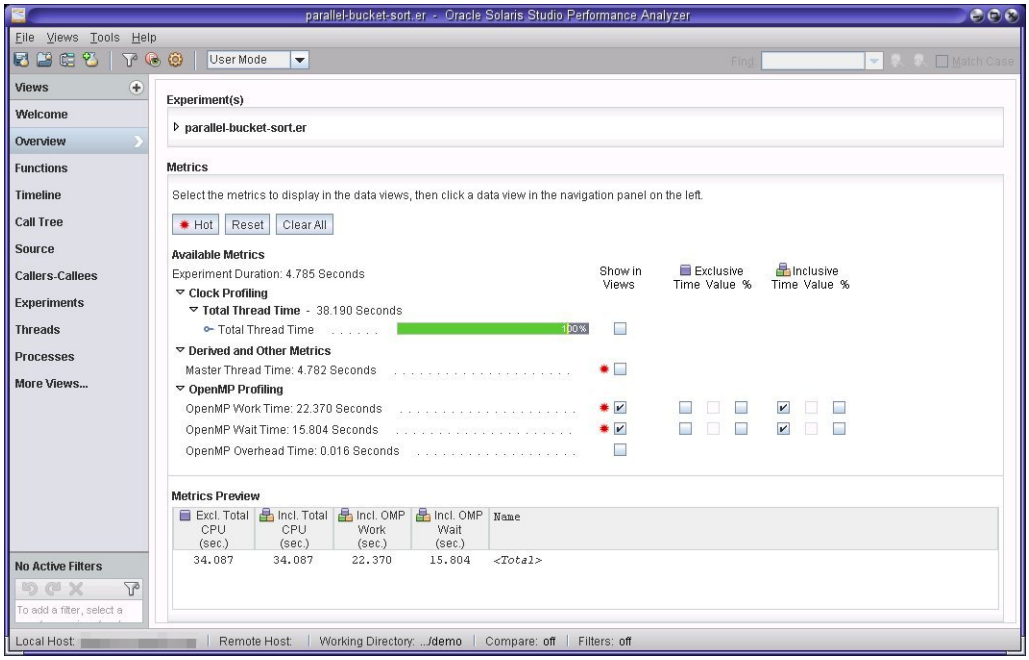
您随时可从导航栏访问 "Welcome"（欢迎）屏幕，因此可以将它作为使用性能分析器的主页。在此屏幕中，可以单击链接来执行活动，还可以打开性能分析器的 "Help"（帮助）查看器来阅读文档。

性能分析器 "Overview"（概述）屏幕

"Overview"（概述）显示已装入实验的性能度量。可以使用 "Overview"（概述）选择要在其他数据视图中了解的度量。

对于每个度量，都将显示一个表示所有已装入实验的总计的值。将显示带颜色的条形以显示相关度量的相对值。高度活跃度量的突出显示标记指示与 CPU 时间比较时表现得高度活跃的度量。这些度量可以帮助识别程序中的问题领域。

图 3-3 性能分析器的 "Overview"（概述）和新导航



有关 "Overview"（概述）屏幕的更多信息，请参见性能分析器中的 "Help"（帮助）。

"Timeline"（时间线）改进

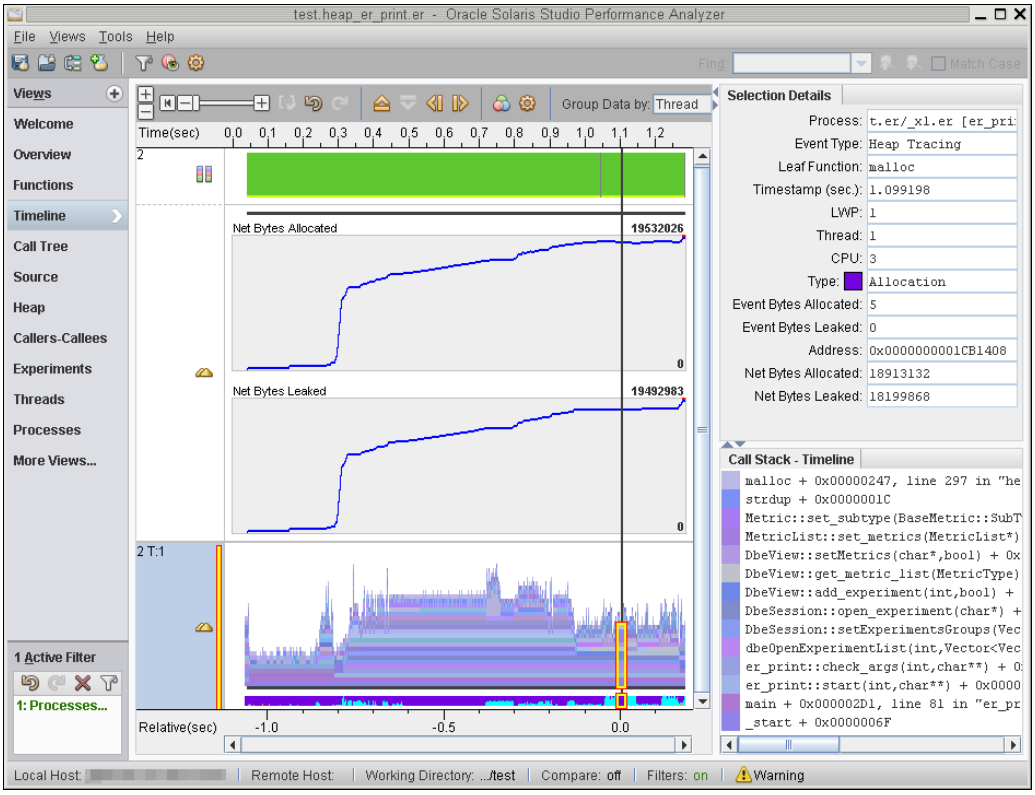
"Timeline"（时间线）视图有以下增强功能，让您可以更轻松地检查数据。

- 单击并拖动鼠标可以设置标记以定义用于缩放和过滤的精确时间段。
- 通过使用新的垂直缩放工具栏控件，可以缩放显示最大进程和线程数。
- 通过按住 Shift 键的同时拖动鼠标进行放大。

有关时间线导航的更多信息，请参见 "Help"（帮助）菜单中 Keyboard Shortcuts and Mnemonics（键盘快捷键和助记符）的 "Timeline"（时间线）部分。

下图显示了堆跟踪数据的 "Timeline"（时间线）视图。时间线显示堆分配和泄漏的图形，以及分配和可用空间的调用堆栈。右侧显示有关当前选择的内存分配和调用堆栈的详细信息。

图 3-4 "Timeline"（时间线）视图



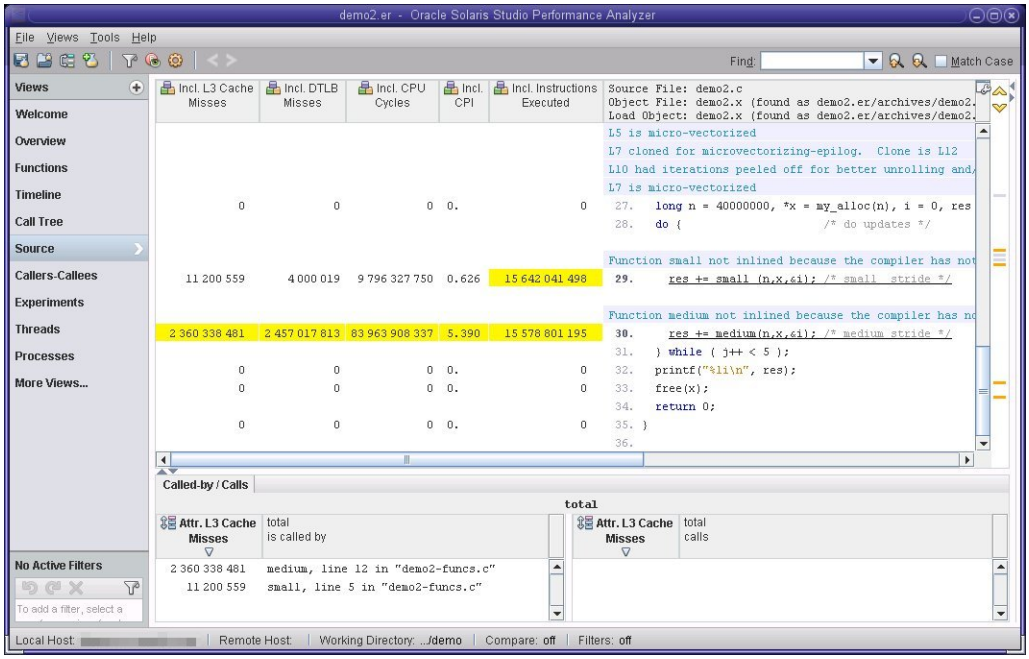
有关在样例实验中使用时间线的逐步说明，请参见《Oracle Solaris Studio 12.4：性能分析器教程》。

"Source"（源）和 "Disassembly"（反汇编）的改进

"Source"（源）视图现在显示时会根据源语言突出显示语法，它有多项导航改进，包括指向调用方和被调用方函数的超链接。

下面的 "Source"（源）视图显示了归属于两行源代码的硬件计数器度量。

图 3-5 "Source"（源）数据视图中的语法突出显示和导航改进



"Source"（源）和 "Disassembly"（反汇编）视图的其他更改包括：

- 右边缘中的导航控件可以帮助跳到带有高度量值的行。
- 通过右键单击菜单和超链接，可以轻松地在函数的调用方和被调用方的 "Source"（源）和 "Disassembly"（反汇编）视图之间导航。
- "Source"（源）和 "Disassembly"（反汇编）视图底部的 "Called-by/Calls"（调用方/调用）标签允许导航调用路径。选择视图中的一个函数，然后可以使用这些标签导航到调用它的函数或它执行的函数调用。当单击 "Called-by/Calls"（调用方/调用）标签中的某个函数时，会在数据视图中选择该函数。
- 如果源文件比实验新，则 "Source"（源）视图会显示警告。
- 前进和后退按钮允许导航在 "Source"（源）或 "Disassembly"（反汇编）视图中执行的操作的历史记录。

有关 "Source"（源）和 "Disassembly"（反汇编）视图的更多信息，请参见性能分析器中的 "Help"（帮助）。

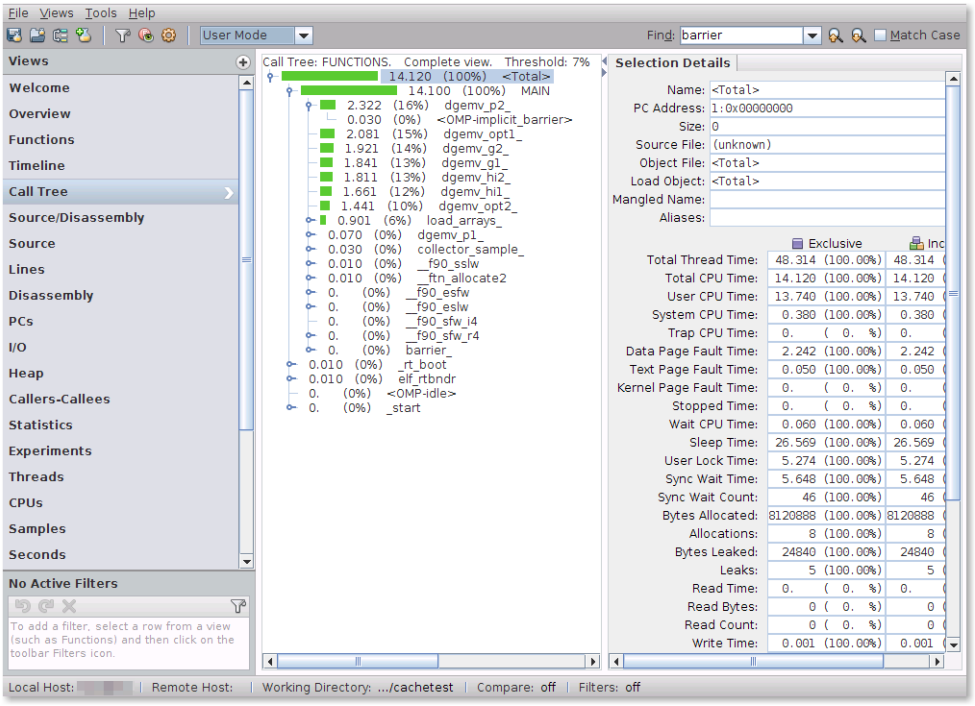
"Call Tree"（调用树）方面的改进

"Call Tree"（调用树）视图具有以下增强功能：

- 度量百分比以颜色条方式显示。
- 新的 "Threshold"（阈值）设置，您可以用来指定何时展开具有高度量值的分支。
- "Show Next Reference"（显示下一个引用）和 "Show All References"（显示所有引用）上下文菜单项，您可以用来查找包括选定函数的分支。
- 右键单击菜单上提供了用于显示和隐藏工具提示和颜色条的新操作。
- 您在其他数据视图中选择的函数将反映在 "Call Tree"（调用树）中（通过展开包含该函数的最常用分支），并且您在 "Call Tree"（调用树）中选择的函数在其他视图中也将被选定。
- 您可以按名称或度量进行排序。

有关 "Call Tree"（调用树）视图的更多信息，请参见性能分析器中的 "Help"（帮助）。

图 3-6 "Call Tree"（调用树）视图



Java 分析改进

通过以下方式改进了 Java 应用程序的分析：

- 现在，只要目标应用程序是 Java 虚拟机，缺省情况下就会启用 Java 分析。以前，必须使用 `collect` 命令指定 `-j on` 选项才能分析 Java 应用程序。
- 现在，使用性能分析器中的 "Profile Running Process"（分析运行的进程）对话框即可分析运行中的 Java 应用程序。收集器会连接到进程，并记录 JVM 和 Java 程序的分析数据和调用堆栈。
- 性能分析器和 `er_print` 可以显示 Java 线程名称和线程组，并允许将它们用于过滤。
- 现在支持分析 JDK 8。

请参见 Java 分析教程《Oracle Solaris Studio 12.4：性能分析器教程 》中的第 3 章 “Java 分析简介”。

简化的硬件计数器分析

对于 Oracle 硬件，现在可以使用新选项 `-h on` 启用缺省分析。此选项将选择度量 CPI/IPC 和高延迟内存访问的计数器。

硬件计数器的规格也已简化。在 Solaris 上，[`on`|`high`|`low`] 现在选择与用于时钟分析的速率相当的分析速率。在选择合适的溢出期间时，[`on`|`high`|`low`] 选项承担了大部分的推测工作，这样可以极大地降低抽样过少或过多的风险。在 Linux 上，支持将 [`on`|`high`|low] 选项用于有别名的硬件计数器，但是需为原始计数器提供数字规格。

有关如何使用硬件计数器分析的详细信息，请参见《Oracle Solaris Studio 12.4：性能分析器教程 》中的第 5 章 “多线程程序上的硬件计数器分析”。

内存空间分析方面的改进

内存空间分析允许您查看哪些内存地址消耗的性能最多。内存空间分析适用于运行 Oracle Solaris 10 和 11 的 SPARC 平台，以及运行 Oracle Solaris 11.2 的 x86 平台。

这种分析使用称为 *precise load-store* 计数器的硬件计数器。

使用命令 `collect -h` 查看您的系统中有哪些 precise load-store 计数器可用。有关如何使用这些计数器执行内存空间分析的更多信息，请参见 `collect`(1) 手册页。

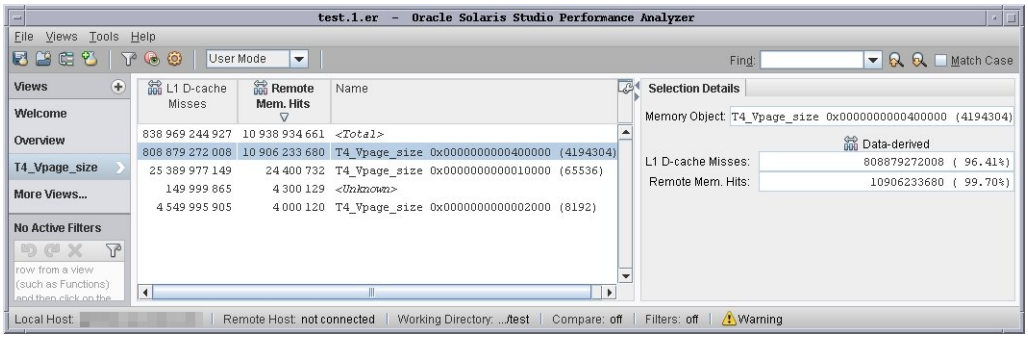
内存空间分析的改进包括：

- 使用缺省 `-h on` 选项的硬件计数器分析通常至少包括一个内存空间计数器
- 不再需要将 + 号和精确的计数器结合使用来触发内存空间分析
- 如果数据存在于实验中，则内存空间分析数据视图可用

在性能分析器中打开内存空间分析实验时，您必须在 "Overview"（概述）页面或 "Settings"（设置）对话框中启用相关的硬件计数器，以在数据视图中显示计数器。从 "Views"（视图）菜单中选择一个内存空间视图。

下图中显示了高速缓存未命中次数（按内存页）的数据视图样例。

图 3-7　显示内存空间分析度量的内存页视图



性能消耗可以归因于高速缓存行和内存页。当与过滤功能配合使用时，该数据可以准确识别出哪些源代码行正在进行特定的高延迟内存引用。

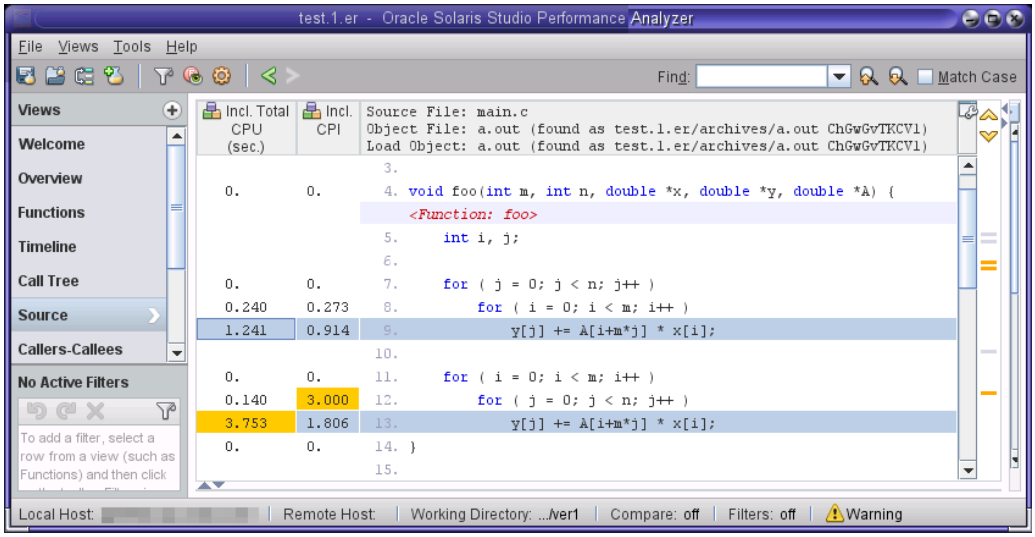
新的派生度量：CPI 和 IPC

性能分析器会显示名为 CPI（Cycles Per Instruction，每指令周期数）和 IPC（Instructions Per Cycle，每周期指令数）的新派生度量，这些度量有助于识别运行效率高或低的应用程序。当您对应用程序执行硬件计数器分析和指定用于周期和指令的计数器时，CPI 和 IPC 度量可用。例如，您可以使用命令 `collect -h cycles,on,insts,on` 生成度量并进行分析。

低效率区域显示高 CPI 或低 IPC。程序的高效率区域显示低 CPI 或高 IPC。

下面 "Source"（源）视图的第 12 行显示的 CPI 度量较高，这表示它的运行效率不高。

图 3-8 "Source"（源）视图中的 CPI 度量



跨平台分析

性能分析器可以读取任何受支持操作系统和体系结构上记录的实验，与运行它的操作系统和系统的体系结构无关。如果性能分析器在远程客户机系统上运行，它可以读取在任何受支持操作系统和体系结构上记录的实验，与远程主机无关。

例如，如果您在 Windows 手提电脑上远程运行性能分析器，您可以连接到基于 x86 的运行 Oracle Solaris 的远程系统，并打开在基于 SPARC 的运行 Oracle Solaris 的系统上记录的实验。

性能分析器的远程使用

可以从性能分析器连接到远程主机，以检查远程主机上的实验。使用远程连接时，具有在记录实验的同一环境中检查实验的优势。

客户机系统要求：

- 客户机操作系统必须为 Mac OS X、Windows、Linux 或 Oracle Solaris。

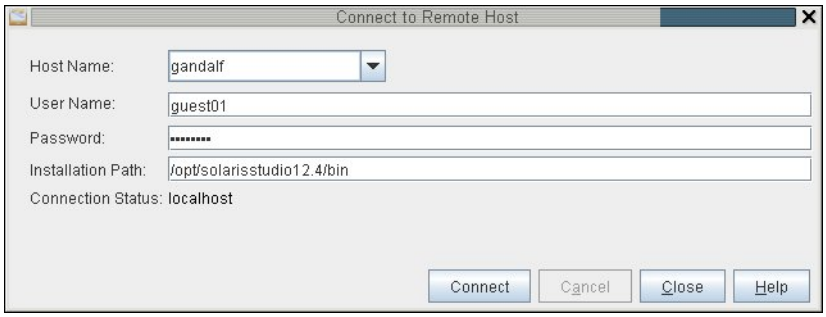
- Java 1.7 必须在您指定的客户机系统路径下。
- 安装在客户机上的性能分析器的版本必须与远程系统上安装的 Oracle Solaris Studio 工具的版本相同。

远程主机要求：

- 远程主机操作系统必须是 Oracle Solaris 或 *Linux*。
- 远程主机必须运行安全 Shell (Secure Shell, SSH) 守护进程 sshd。
- 必须可以在远程主机上访问 Oracle Solaris Studio 12.4 软件，并且需要知道工具的路径。
- 您必须拥有主机上的用户帐户。

在安装了 Oracle Solaris Studio 的 Linux 和 Oracle Solaris 系统上，可以通过以下方法连接到远程系统：选择 "File"（文件）> "Connect To Remote Host"（连接到远程主机）以打开如下图所示的 "Connect To Remote Host"（连接到远程主机）对话框。

图 3-9 "Connect to Remote Host"（连接到远程主机）对话框



为了在未安装此发行版 Oracle Solaris Studio 的系统上使用，在完全 Oracle Solaris Studio 安装的 `lib/analyzer` 目录下以 `RemoteAnalyzer.tar` 文件形式提供了性能分析器的客户机分发。要安装性能分析器的客户机分发，请将该 tar 文件复制到您的系统上并解压缩。目录 `RemoteAnalyzer` 包含了适用于 Windows、Mac OS、Linux 和 Solaris 的脚本、描述功能和用法的 `README.txt` 文件以及包含远程操作所需组件的 `lib` 子目录。

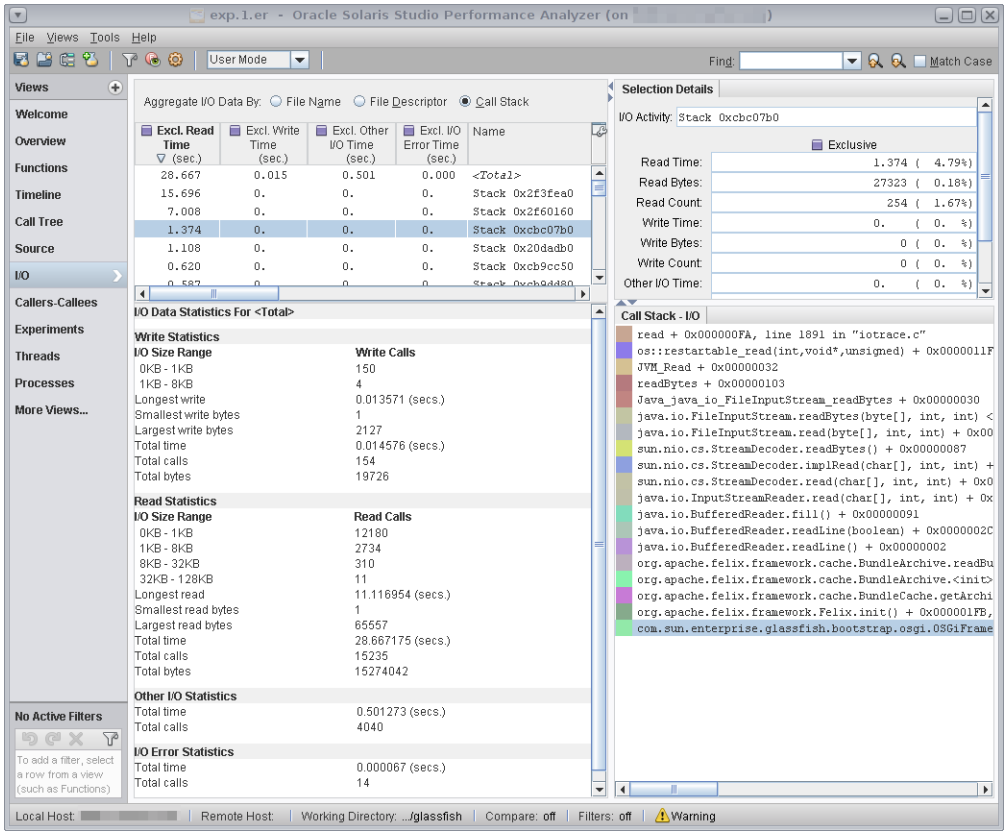
要在客户机系统上启动性能分析器，请执行适合您的系统的脚本。性能分析器启动后，它将显示 "Welcome"（欢迎）屏幕，且该屏幕仅包含可以远程运行的功能。有关更多信息，请参见 `RemoteAnalyzer/README.txt` 文件和性能分析器中的 "Help"（帮助）菜单。

新的 I/O 数据视图

新的 I/O 视图可以帮您识别应用程序的 I/O 模式并确定影响其性能的 I/O 瓶颈。如果从性能分析器或使用 `collect -i on` 命令针对 I/O 跟踪数据分析了应用程序，则 I/O 视图可用。

I/O 视图显示按文件名、文件描述符或调用堆栈聚集的读取和写入数据。您还可以用它来过滤数据中的 I/O 事件。提供了一个新的 "Duration"（持续时间）视图，可分析 I/O 操作的持续时间。还有一个新的 "Data Size"（数据大小）视图按字节大小显示 I/O 操作的分布情况。有关 I/O 视图的更多信息，请参见性能分析器中的 "Help"（帮助）。

图 3-10 I/O 数据视图

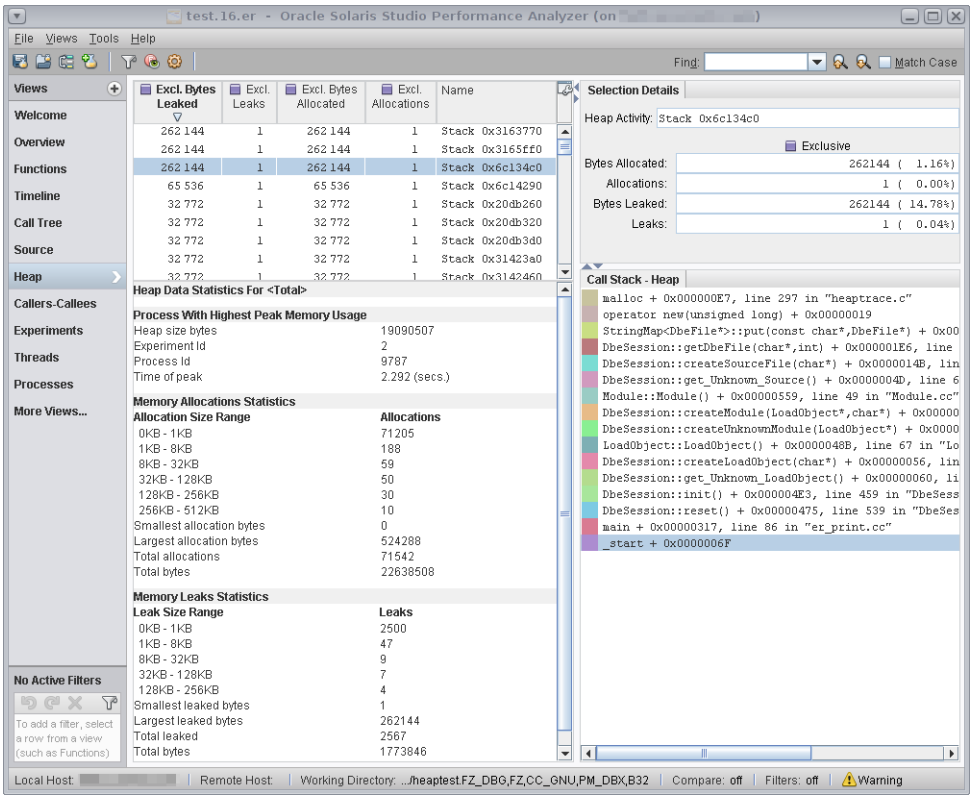


新建堆视图

新的 "Heap"（堆）视图显示程序中可能的内存泄漏。如果从性能分析器或使用 `collect -H on` 命令针对堆跟踪数据分析了应用程序，则此视图可用。

"Heap"（堆）视图显示具有指示可能存在内存泄漏的内存分配度量的调用堆栈列表。现在，时间线将已分配堆的大小显示为时间函数。一个新的 "Data Size"（数据大小）视图按字节数显示分配和泄漏的分布情况。还有一个新的 "Duration"（持续时间）视图用于分析分配的持续时间。有关 "Heap"（堆）视图的更多信息，请参见性能分析器中的 "Help"（帮助）。

图 3-11 新建堆视图



性能分析器的其他更改

性能分析工具包含下列其他增强功能。

- 现在可以使用以下方法指定存储性能分析器用户设置的路径：使用 `analyzer` 命令的 `-u` 或 `--userdir` 参数。
- 比较和聚集实验现在变得更易执行。您可以在绝对值或增量之间选择，以显示比较实验时度量值的变化。
- 按照共享库或 Java 类聚集性能度量有了极大的改进，且支持它的对话框已从 "Show/Hide/API-only"（显示/隐藏/仅 API）更改为 "Set Library and Class Visibility"（设置库和类可见性）
- "Print"（打印）选项已被 "Export to a file"（导出到文件）所取代，支持更多输出。在某些视图中，现在可以导出为 ASCII 文本表、分隔符分隔列表或 HTML 表。所有导出均导出到运行性能分析器的计算机的文件系统中。如果在远程模式下运行，则将数据导出到远程计算机上的文件中。
- 用于时钟分析、MPI 跟踪和堆跟踪的名称已更改。有关详细信息，请参见 `collect`(1) 手册页。
- 性能分析器显示 Java 线程名称和线程组，您可以用它们进行过滤。
- 性能分析器现在支持连接到正在运行的进程并对其进行分析。选择 "File"（文件）> "Profile Running Process"（分析运行的进程），或者单击 "Welcome"（欢迎）屏幕上的 "Profile Running Process"（分析运行的进程）。以前只能使用 `collect` 命令或 `dbx collector` 命令连接到运行的进程。
- 现在可以从性能分析器分析 Oracle Solaris 内核。选择 "File"（文件）> "Profile Kernel"（分析内核），或者单击 "Welcome"（欢迎）屏幕上的 "Profile Kernel"（分析内核）。以前只能使用 `er_kernel` 命令分析内核。
- 性能分析器提供了一种新的方法来持久地保存设置。当您退出性能分析器时，如度量和视图等大部分设置是持久性的，因此下次打开同一个实验时，该实验仍会显示为上次关闭时的情景。您可以将选定的设置保存在配置文件中，并在从 "Open Experiment"（打开实验）对话框打开实验时将此配置应用于相同或不同的实验。性能分析器还可以将设置保存到 `.er.rc` 文件中，以供 `er_print` 读取。

命令行工具的更改

本节介绍各个命令行性能分析工具的更改。

对数据收集工具的更改

数据收集工具包括 `collect` 命令、`dbx collector` 命令和 `er_kernel` 命令。这些工具全都用于分析程序，以收集数据和创建实验，性能分析器或 `er_print` 可以读取此类数据和实验。

这些工具有如下通用的更改：

- 硬件计数器和堆栈展开支持新的处理器：SPARC T5、SPARC M5、SPARC M6、SPARC 64 X、SPARC 64 X+、Intel Ivy Bridge 和 Haswell。
- 即使未指定硬件计数器分析，缺省情况下也会启用时钟分析。
- 将归档设置为 on 与设置归档副本的方法一样。对于 `collect` 和 `er_kernel`，这意味着 `-A on` 现在与 `-A copy` 一样。 对于 `dbx collector`，这意味着 `collector archive on` 与 `collector archive copy` 一样。
- Oracle Solaris 上可以分析的线程最大数现为 32768。

`collect` 实用程序更改

`collect` 实用程序是在运行时分析应用程序的工具，可以收集数据和创建实验，且性能分析器或 `er_print` 可以读取此类数据和实验。

`collect` 实用程序在此发行版中进行了如下更改：

- 新 `-i` 标志支持 I/O 跟踪。
- 只要目标是 JVM，缺省情况下就会启用 Java 分析。现在，不再需要指定 `-j on`。
- 现在 Linux 系统支持使用 `-P` 选项从运行的进程收集数据。请注意，这仅适用于单线程应用程序。
- 现在 Linux 支持使用 `-c` 选项收集计数数据。
- 硬件计数器处理现在支持多个 `-h` 参数和缺省的计数器集。您可以将环境变量 `SP_COLLECTOR_HWC_DEFAULT` 设置为缺省情况下启用硬件计数器分析。
- 在 SPARC 和 x86 系统上，缺省情况下为精确计数器启用基于硬件计数器的内存空间分析。捕获内存地址时不再需要加号 (+)。有关更多信息，请参见“内存空间分析方面的改进” [25]。
- `collect -F =expr` 不再匹配进程沿袭的表达式。
- 使用 `-P` 连接到 Java 程序时，必须指定 `-j on`。

`dbx collector` 的更改

`dbx collector` 是 `dbx` 调试器的子命令，可用于收集性能数据。有关更多信息，请参见 `collector`(1) 手册页。

除了所有数据收集工具通用的更改外，本发行版中的 `dbx collector` 命令还有如下更改：

- `detach` 或实验终止后，可以启动另一个实验。
- 对于单线程本机应用程序，Linux 系统支持连接操作。有关连接限制的更多信息，请参见《Oracle Solaris Studio 12.4：发行说明 》中的“dbx attach 分析 (collect -P)”。

- `dbx collector` 支持以下新命令：
  - `collector iotrace on` – 指定打开 I/O 跟踪。
  - `collector duration` – 指定运行实验的时间范围。
  - `collector java` – 指定是否收集 Java 分析数据。缺省值为 `off`。分析 Java 时，应当通过连接或启动 JVM 来使用此命令。
  - `collector pausesig` – 指定用于暂停或恢复数据收集的信号。
  - `collector samplesig` – 指定用于记录样例的信号。
  - `collector hwprofile addcounter` – 为硬件计数器溢出分析指定其他计数器。

`er_kernel` 实用程序更改

`er_kernel` 命令分析 Oracle Solaris 内核，并生成可在性能分析器或 `er_print` 中检查的实验。

除了所有数据收集工具通用的更改外，本发行版中的 `er_kernel` 实用程序还有如下更改：

- 用户子实验中的时钟分析度量会记录下来，就像在 `collect` 实验中一样，但只会记录用户 CPU 时间和系统 CPU 时间，并不记录等待时间。
- 在内核创建者实验中报告的硬件计数器度量的名称以 `K_` 开头。用户子实验中的度量使用的名称与 `collect` 实验中的度量的名称相同。
- 对于精确计数器和安装了 DTrace 1.8 版或更高版本的系统，Solaris SPARC 上支持内核数据空间分析。
- 现在会记录创建者实验的时钟分析，就像在 `collect` 实验中一样，除非它们只有内核 CPU 时间度量。缺省情况下不启用硬件计数器分析。
- 内核调用堆栈的记录有所改进。

`er_print` 实用程序更改

`er_print` 实用程序会生成纯文本版本、可由性能分析器显示的数据视图。输出显示标准输出中。

`er_print` 实用程序在此发行版中进行了如下更改：

- IO 跟踪数据可在三个新报告中使用：`ioactivity`、`iodetail` 和 `iocallstack`。在 `ioactivity` 报告中，数据按文件名聚集。在 `iodetail` 报告中，数据在打开时由每个文件描述符分隔。在 `iocallstack` 中，数据按照通用调用堆栈聚集。
- 堆跟踪数据可在新报告中使用：`heap` 和 `heapstat`。使用 `heap` 输出按通用调用堆栈聚集的所有分配和泄漏。使用 `heapstat` 输出堆使用情况统计信息的摘要，包括整个进程存在期间的峰值使用情况。

- 当您分析应用程序以收集 cycles 和 insts 硬件计数器时，新的派生度量 CPI（Cycles Per Instruction，每指令周期数）和 IPC（Instructions Per Cycle，每周期指令数）可用。CPI 和 IPC 可以指定为度量。
- 新报告 overview 显示实验的摘要信息。
- object_list 显示的内容现包括索引（如其他数据视图的 PC 中所示）、指向对象的全路径及其共享库函数的可见性设置。
- 现在，er_print 在任何其他受支持的体系结构上运行时，可以读取在任何体系结构上记录的实验。
- 新的 printmode 命令，支持将 ASCII 表单输出为以前的版本 (string = table)，或输出为分隔符分隔的列表（string = *X*，X 可以是任何单字符分隔符）或输出为 HTML 格式的表 (string = html)。printmode 命令可在 er.rc 文件中。
- 如果在用户 er.rc 处理过程中，又或者使用输入命令或脚本中的命令指定了 machinemodel，或者如果正在装入的实验已经记录 machinemodel，则会创建特定于计算机的内存对象。
- er_print 显示 Java 线程名称和线程组，并可用它们进行过滤。
- 比较模式支持将比较数据显示为绝对值或增量，以揭示各个实验间的度量值变化。
- .er_rc 文件中不再允许使用 setpath 指令。如果使用前一个发行版创建的 .er_rc 文件中有 setpath 指令，则应将 setpath 行更改为 addpath。

其他命令的更改

er_archive 命令在此发行版中进行了如下更改：

- er_archive（和所有其他工具）可以处理符号信息记录在副文件中的可执行文件。
- er_archive 现在仅复制共享对象，不会生成归档文件；且会在不提示的情况下忽略 -A 标志。
- er_archive 支持用 -s *type* 标志指定归档或源、对象和辅助文件。如果 *type* 为 all，则归档所有可以找到的源文件；如果 *type* 为 used，则只归档实验所引用的源文件。
- er_archive 会检查环境变量 SP_ARCHIVE_ARGS，使用该变量可以指定 -s 和 -m 参数。如果在记录实验时设置了 SP_ARCHIVE_ARGS 变量，则自动运行的 er_archive 可以执行源归档操作。手动运行 er_archive 时，可以处理该变量。

对实验的更改

性能数据实验有如下更改：

- 实验格式已更改，且版本号现为 12.4。
- 此发行版的性能分析工具可以读取以下实验版本：

10.2	使用低于 Oracle Solaris Studio 12.3 的发行版创建
12.3	使用已发行的 Oracle Solaris Studio 12.3 创建
12.4	使用此 Oracle Solaris Studio 12.4 创建

如果您尝试使用低于 10.2 版的实验，工具会显示消息，指明必须使用早期版本的工具读取此实验。

- 时钟分析度量已经过重新组织，以显示更多详细信息，有些度量已重命名，且设置了不同的缺省值。有关更多详细信息，请参见 `collect` 手册页，该页面现在列出了度量缩写词及名称。

♦ ♦ ♦ 第 4 章

代码分析工具

代码分析工具套件可检测常见的编码错误（包括内存泄漏和访问违规），使开发者可以更快地编写错误更少的更好代码。

本章介绍此 Oracle Solaris Studio 发行版中代码分析工具的新增功能和更改的功能，其中包含以下各节：

关于代码分析工具

代码分析工具使用静态、动态和代码覆盖分析来检测许多常见的编码错误（包括内存泄漏和内存访问违规），从而帮助应用程序变得更加可靠。Previse 在编译时执行静态分析，Discover 在应用程序运行时执行动态分析，以确定代码质量问题。Uncover 工具分析代码覆盖数据以提供有关测试套件未涵盖的函数的信息，并告知如何通过覆盖这些函数受益。

使用代码分析器图形工具或新增的 `codean` 命令行实用程序可以查看三种类型的分析，以便全面了解应用程序的漏洞，从而提高应用程序的正确性和可靠性。

新增的命令行代码分析器工具 codean

利用新实用程序 `codean`，可以使用命令行查看 Previse、Discover 和 Uncover 生成的代码分析数据，而不必使用代码分析器图形工具。`codean` 工具提供类似于代码分析器的功能，但可在没有图形环境的系统上使用，也可供偏好命令行的用户使用。`codean` 工具还可用于自动化脚本，并且有一些在代码分析器工具中尚未提供的功能。

codean 工具有以下独特功能：

- 查看 HTML 和文本格式的结果。
- 保存工具报告供与被检查的二进制文件的新版本比较。
- 将当前报告与保存的报告比较以仅显示新错误。请参见“使用 --whatisnew 选项” [38]。
- 将当前报告与保存的报告比较以仅显示已修复的错误。请参见“使用 --whatisfixed 选项” [38]。
- 显示多次动态错误检查运行发现的所有错误。例如，如果某个测试套件运行时使用了 Discover 检测的二进制文件，则将所有数据汇总在一个文本报告中。
- 为某个目录下的所有报告编译一个摘要 HTML 页面。请参见“使用 codean 命令生成摘要 HTML 页面” [39]。

有关 codean 的更多信息，请参见 codean(1) 手册页。

使用 --whatisnew 选项

使用 --whatisnew 选项可通过与以前保存的工具报告比较，生成仅包含新错误的报告。例如，使用代码分析工具可创建采用工具时源代码库状态的冻结副本。然后，可使用 --whatisnew 确保对源代码库的继续更改不会形成新的安全漏洞。

以下是使用 --whatisnew 以仅显示新错误的一个示例：

```
%codean --whatisnew a.out
STATIC report of a.out showing new issues:
Compare the latest results against a.out.analyze/history/09:58:35May152013...
MEMORY LEAK 1 : 1 block left allocated on heap with a total size of 400 bytes
    sample1()  <sample1.c : 20>
        17:    {
        18:          global = (int *)malloc(100);
        19:          int *p = malloc(100*sizeof(int));
        20:=>          int *q = malloc(100*sizeof(int));
        22:          add_0_1_put_in_2(p);-
PREVISE SUMMARY:
    0 new error(s), 0 new warning(s), 1 new leak(s) in total
```

使用 --whatisfixed 选项

--whatisfixed 选项可以生成仅显示在以前缓存的工具报告中发生的错误的报告，从而补充了 --whatisnew 选项的不足。开发团队或质量保证团队可以使用此选项来跟踪安全错误修复。

以下是使用 --whatisfixed 以仅显示修复的错误的一个示例：

```
% codean --whatisfixed a.out
STATIC report of a.out showing fixed issues:
Compare the latest results against a.out.analyze/history/10:02:05May152013...
MEMORY LEAK 1 : 1 block left allocated on heap with a total size of 400 bytes
(Warning: Source files have changed. Source code shown below may not be accurate.)
    sample1()  <sample1.c : 20>
        17:    {
        18:          global = (int *)malloc(100);
        19:          int *p = malloc(100*sizeof(int));
        20:=>          int *q = malloc(100*sizeof(int));
        21:          free(q);
PREVISE SUMMARY:
    0 fixed error(s), 0 fixed warning(s), 1 fixed leak(s) in total
```

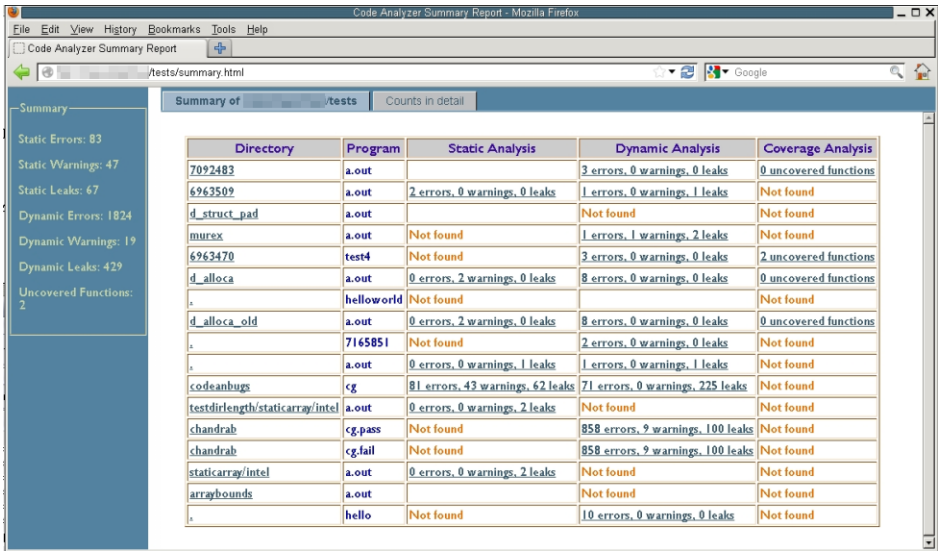
使用 codean 命令生成摘要 HTML 页面

可以创建一个摘要报告，其中包含以前对某个目录树中的多个二进制文件运行的代码分析工具生成的所有信息。该 HTML 页面将多个二进制文件的静态、动态和覆盖报告中的所有错误整理成一个简洁的表。

例如，如果要在顶级目录 `./tests` 中创建一个 HTML 页面，则可使用 `codean ./tests`。

下图是生成的包含 `/tests` 目录下所有报告的 `summary.html` 页面的一个示例。

图 4-1 摘要 html 报告



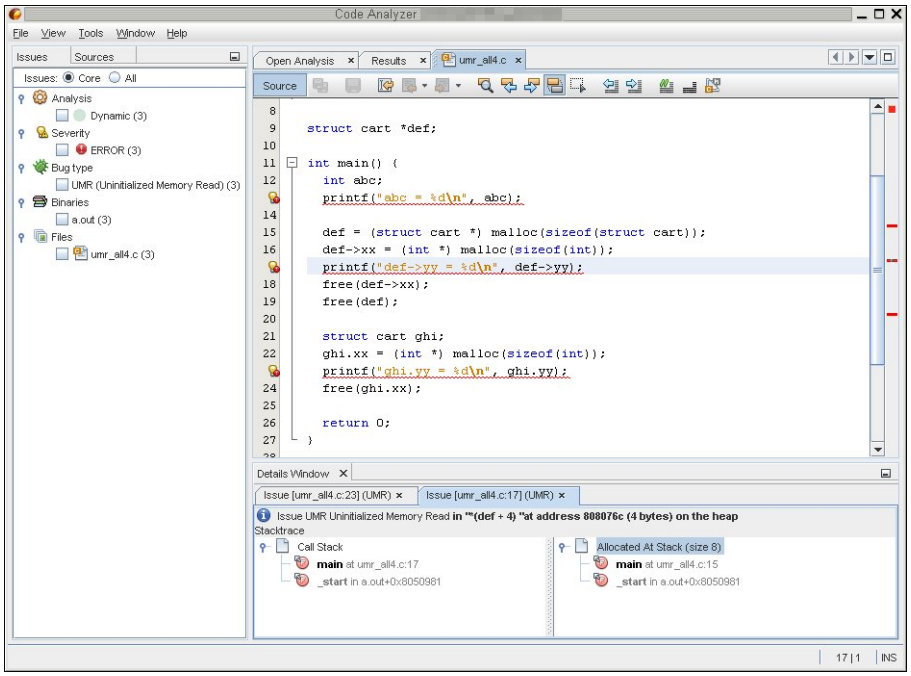
代码分析器更改

所有代码分析器组件工具中都增加了以下常用功能。有关更多信息，请参见代码分析器中的 "Help"（帮助）和《Oracle Solaris Studio 12.4：代码分析器用户指南》。

- 目前在 Oracle Enterprise Linux 上可用。
- 通过在代码分析器组件检测引擎中采用 64 位技术，改进了企业应用程序支持。
- 极大地改进了检测引擎运行时。
- 由于大量减少了引擎内存占用，可以在小型计算机上检测大型应用程序。
- 增加了对 SPARC T5 和 M5 处理器的支持。
- 增加了对 Intel Ivy Bridge 处理器的支持。
- 在动态错误报告中增加了显示变量名称信息。

下图显示的代码分析器 GUI 包含了一个显示变量名称的代码示例。

图 4-2 包含变量名称的代码分析器 GUI 屏幕抓图



通常，有关代码分析器的更多信息，请参见代码分析器中的 "Help"（帮助）、《Oracle Solaris Studio 12.4：代码分析器用户指南》、《Oracle Solaris Studio 12.4：代码分析器教程》和 `code-analyzer` (1) 手册页。

新增 Previse 静态分析功能

此发行版中为 Previse 静态分析工具增加了以下功能：

- 现在，要收集静态代码错误，可以在编译代码时使用 `-xprevise` 选项，而不再使用已废弃的 `-xanalyze=code` 选项。
- 现在，可在 Oracle Enterprise Linux 上使用静态分析。
- 通过消除误报极大地提高准确性。

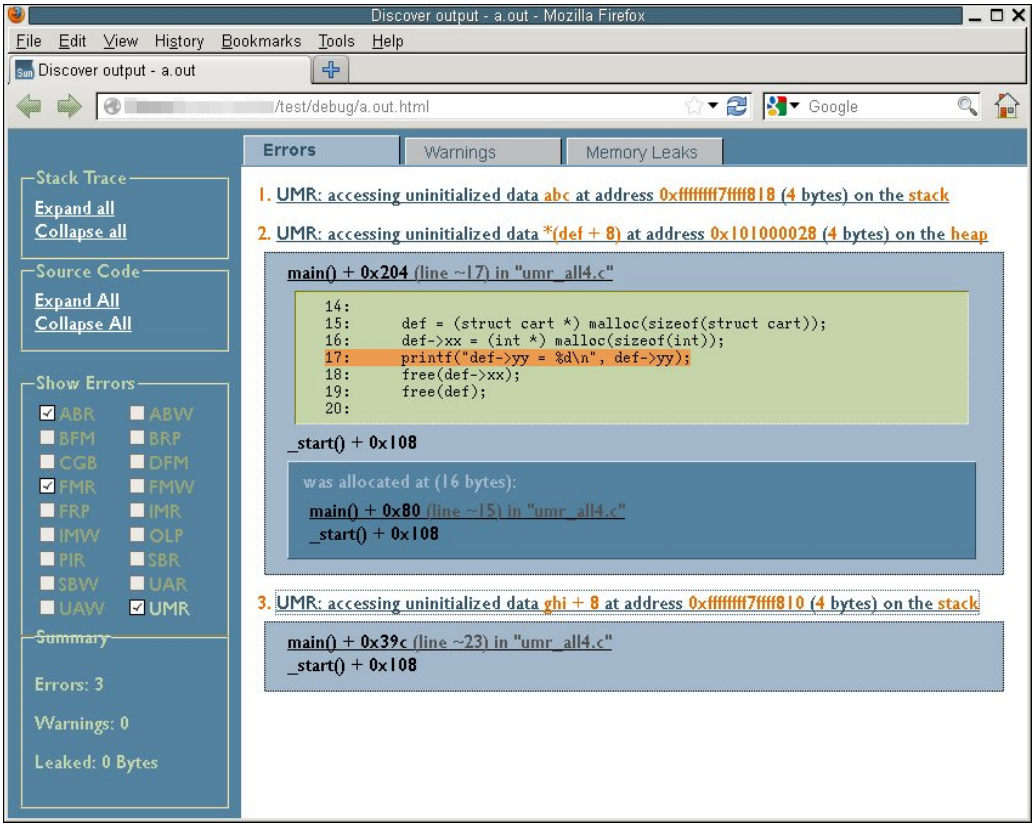
新增 Discover 功能

此发行版的 Discover 内存分析工具中增加了以下功能。有关更多信息，请参见 `discover`(1) 手册页和《Oracle Solaris Studio 12.4：Discover 和 Uncover 用户指南》。

- Discover 可通过使用 `-c[- | lib[:scope...] | file]` 选项检查可执行文件或库的某些部分。有关更多信息，请参见《Oracle Solaris Studio 12.4：Discover 和 Uncover 用户指南》中的“检查库或可执行文件的部分”。
- 新的 Discover API 可按需显示内存泄漏及内存使用情况。有关更多信息，请参见“新增 Discover API” [42]。
- 现在，可在 Oracle Enterprise Linux 上使用 Discover。
- Discover 可使用 `-F both` 选项跟踪和收集子进程和父进程中的内存访问数据。这是新的缺省值。
- Discover 错误报告支持在一个测试套件中多次运行目标二进制文件。错误报告格式可与新的命令行实用程序 `codean` 配合使用。
- 新增对使用 `mmap(2)` 分配的代码进行内存错误检查的功能。
- 改进了 HTML 报告，可以突出显示变量名称、行号和地址。
- 在发生了内存损坏的情况下，现在可在内存错误中显示变量名称。
- 缺省情况下，Discover 现在可以捕获静态类型数组超出边界错误。有关更多信息，请参见《Oracle Solaris Studio 12.4：Discover 和 Uncover 用户指南》中的“内存访问错误和警告”。
- 现在，使用 `-i datarace` 选项可以报告通过使用 Discover 检测二进制文件所检测到的争用的双栈跟踪。
- 增加了对大文件的支持。

下图的示例使用 Discover 生成了一个报告，其中包含突出显示的 HTML 以及变量名称。

图 4-3 Discover HTML 突出显示和变量名称示例



新增 Discover API

代码分析工具中新增了六个 Discover API。当应用程序使用的内存太多时，可以在应用程序结束前使用这些 API 精确定位发生的内存泄漏。这些 API 还可以定位所有使用中的内存块。

可以将这些 API 直接插入到应用程序源代码中，也可以在调试会话过程中随时调用它们，以报告从某个点开始的新增内存使用量和内存使用总量，以及新增的内存泄漏量和内存泄漏总量。对于长时间运行的或者一直运行的企业服务器应用程序，这些 API 尤其有用。

代码分析工具中增加了以下 API：

- `discover_report_all_inuse ()`
- `discover_mark_all_inuse_as_reported ()`
- `discover_report_unreported_inuse ()`
- `discover_report_all_leaks ()`
- `discover_mark_all_leaks_as_reported ()`
- `discover_report_unreported_leaks ()`

例 4-1 使用 `discover_report_unreported_leaks ()` API

以下是在使用 Discover 运行二进制文件以及使用 `dbx` 执行 API 之后使用 `discover_report_unreported_leaks ()` API 的一个示例。

```
%discover -w - a.out
% dbx a.out
 (dbx) stop in foo2
 (dbx) run
 (dbx) call discover_report_unreported_leaks()
 ******** discover_report_unreported_leaks() Report ********

  1 allocation at 1 location left on the heap with a total size of 1 byte

      LEAK 1: 1 allocation with total size of 1 byte
          foo1() + 0x5e  <api_example.c:7>
                   4:    #include <discoverAPI.h>
                   5:
                   6:    void foo1() {
                   7:=>    char *x = (char *) malloc(sizeof(char));
                   8:      *x = 'a';
                   9:      printf("x = %c\n", *x);
                  10:      /*free(x);*/
          main() + 0x1a  <api_example.c:21>
                  18:    }
                  19:
                  20:    int main() {
                  21:=>    foo1();
                  22:      foo2();
                  23:      return 0;
                  24:    }
          _start() + 0x71

  ************************************************************
```

有关每个 Discover API 以及如何使用它们的更多信息，请参见 Discover 头文件以及《Oracle Solaris Studio 12.4：Discover 和 Uncover 用户指南》中的“discover API”。

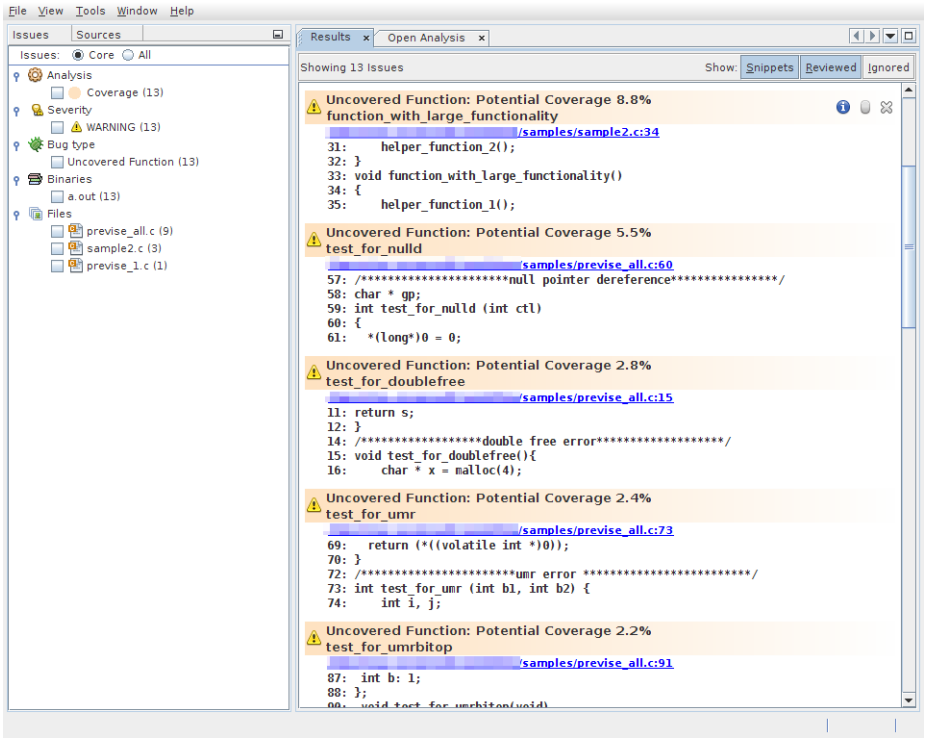
新增的 Uncover 功能

此发行版的 Uncover 代码覆盖工具中增加了以下功能。

- 现在，可在 Oracle Enterprise Linux 上使用 Uncover。
- 提高了 C++ 应用程序的覆盖报告中的正确性。
- 支持在长时间运行的服务器应用程序中常见的异常进程终止。
- 减少了 Uncover 检测和运行时的时间和内存占用开销。
- 现在，Uncover 可以在程序运行期间获取覆盖数据和分析数据。
- 增加了对大文件的支持。

下图中是通过代码分析器 GUI 使用 Uncover 的一个示例：

图 4-4 代码分析器 GUI 中的 Uncover 结果示例



有关更多信息，请参见 `uncover`(1) 手册页和《Oracle Solaris Studio 12.4：Discover 和 Uncover 用户指南》。

♦ ♦ ♦ 第 5 章

5 调试工具

Oracle Solaris Studio 提供了命令行 dbx 调试器以及用于使用 dbx 的 dbxtool 图形工具。调试器还集成到 IDE 中。有关使用 IDE 进行调试的更多信息，请参见第 6 章 Oracle Solaris Studio IDE。

本章包含有关调试工具中新增功能的以下主题：

关于 dbx 调试器

dbx 调试器是一个交互式源代码级事后和实时调试工具。可以在命令行通过 dbxtool 图形界面和在 Oracle Solaris Studio IDE 中使用该调试器。dbx 调试器是可脚本化并且多线程感知的。

新增和更改的 dbx 功能

以下是 dbx 中添加或更改的功能。有关更多信息，请参见《Oracle Solaris Studio 12.4：使用 dbx 调试程序》、dbx (1) 手册页和 dbx 帮助文件。

- dbx 的启动时间大大减少，企业应用程序最多可以提高 7 倍。
- dbx 的新编译器和链接程序选项。有关更多信息，请参见“用于支持调试的新编译器和链接程序选项” [46]。
- 新嵌入的 Python 解释器，通过它可以在 Oracle Solaris 上为 C 和 C++ 表达式编写美化输出过滤器。有关更多信息，请参见“使用 Python 的美化输出” [48]。
- 新 dbxenv 变量 output_pretty_print_mode，确定使用的美化输出机制。如果设置为 call，则使用调用样式的美化输出器。如果设置为 filter，则使用基于 Python 的美化输出器。如果设置为 filter_unless_call，则先使用调用样式的美化输出器。

- 新 dbxenv 变量 filter_max_length。将此 dbxenv 变量设置为通过美化输出过滤器转化为数组的序列最大长度。
- 添加了对 C++11 标准的支持。
- 添加了对 C11 标准的支持。
- 添加了对以下编译器选项的支持。有关这些选项的更多信息，请参见：“编译器中的更改” [63]。
  - -g1
  - -xdebuginfo
  - -xglobalize
  - -xpatchpadding

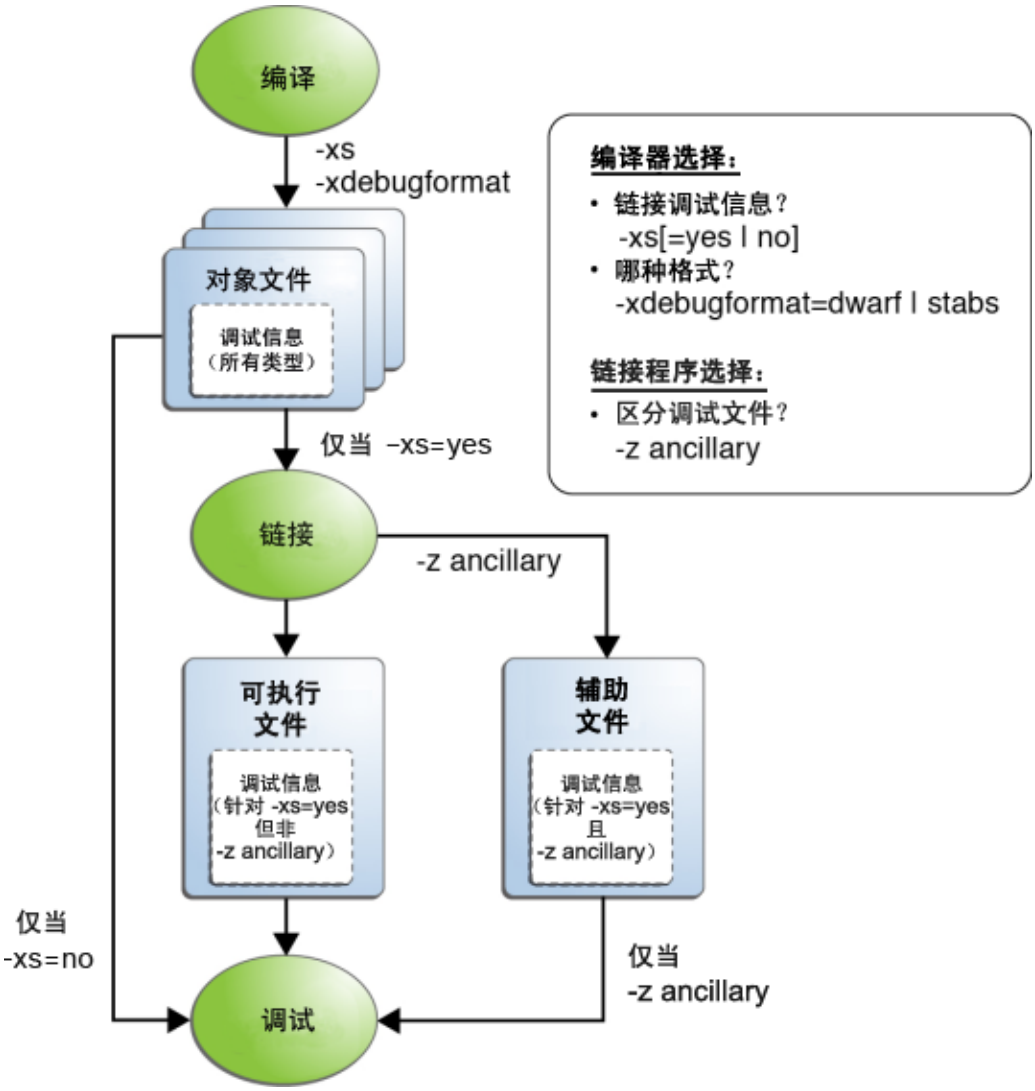
有关更多信息，请在 dbx 下发出 help changes 命令来访问 dbx 帮助文件。

用于支持调试的新编译器和链接程序选项

使用新编译器和链接程序选项，用户可更自由地生成和使用调试信息。编译器为 DWARF 生成索引，这与索引 stabs 类似。该索引始终存在，并可在使用 DWARF 进行调试时加快 dbx 启动速度并带来其他改进。

下图显示了不同类型的调试信息及其位置，特别突出显示了调试数据所在的位置：

图 5-1 调试信息流



索引 DWARF (-xs[={yes|no}])

缺省情况下，DWARF 已装入可执行文件。新索引可使用 -xs=no 选项将 DWARF 保留在对象文件中。这样可生成更小的可执行文件和更快的链接。对象文件必须保留，以便进行调试。这与 stabs 的工作原理类似。

独立的调试文件 (-z ancillary[=*outfile*])

Oracle Solaris 11.1 链接程序可以将调试信息发送到单独的辅助文件，同时生成可执行文件。在必须移动、安装或归档所有调试信息的环境中，独立的调试文件非常有用。可执行文件既可独立运行，也可由用户使用其独立的调试文件副本进行调试。

dbx 继续支持使用 GNU 实用程序 objcopy 将调试信息提取到独立的文件中，但与 objcopy 相比，使用 Oracle Solaris 链接程序具有以下优点：

- 独立的调试文件作为链接的副产物生成
- 由于过大而难以作为一个文件链接的程序可作为两个文件链接为两个文件

注 - 使用索引 DWARF 和独立的调试文件具有仅复制索引的效果，其仅将要链接的信息传送到独立的调试文件。这通常不是非常有用，除非索引使用的少量空间非常关键。

最大程度地减少调试信息

-g1 编译器选项旨在实现已部署应用程序的最小可调试性。使用此选项编译应用程序时，将生成文件、行号以及系统认为对事后调试至关重要的简单参数信息。有关更多信息，请参见编译器手册页和编译器用户指南。

使用 Python 的美化输出

dbx 现在具有一种机制，从而可以在 Python 中编写美化输出过滤器。美化输出过滤器将值转化为 dbx 中的更可读形式。

在 dbx 命令行上，可以通过使用 print 的 -p 选项、display 和 watch 命令或通过键入 dbxenv output_pretty_print on 来启用美化输出。在 IDE 和 dbxtool 中，可以通过将 dbxenv 变量 output_pretty_print 设置为 on 来启用美化输出并且可以使用 "Watches"（监视）和 "Variables"（变量）窗口的上下文菜单中的 "Pretty Print"（美化输出）复选框。

系统已内置过滤器，用于在 C++ 标准模板库的 4 种实现中选择类。下表指定了库名和该库的编译器选项：

库的编译器选项	库名称
-library=Cstd（缺省值）	libCstd.so.1
-library=stlport4	libstlport.so.1
-library=stdcxx4	libstdcxx4.so.4.**
-library=stdcpp（使用 -std=c++11 选项时的缺省值）	libstdc++.so.6.*

下表指定哪些类过滤器可在 C++ 标准模板库中使用，以及是否可输出索引和分片：

类	索引和分片可用
string	不适用
pair	不适用
vector	是
list	是
set	是
deque	是
bitset	是
map	是
stack	是
priority_queue	是
multimap	是
multiset	是
tuple（仅 C++）	不适用
unique_ptr（仅 C++）	不适用

例 5-1 使用过滤器的美化输出

以下是一个在 dbx 中使用 print 命令时输出列表的输出示例：

```
(dbx) print list10
list10 = {
   __buffer_size = 32U
   __buffer_list = {
     __data_ = 0x654a8
   }
   __free_list   = (nil)
   __next_avail  = 0x67334
   __last     = 0x67448
   __node     = 0x48830
   __length   = 10U
   }
```

以下是与 dbx 中输出的列表相同的列表，但该列表使用的是美化输出过滤器：

```
(dbx) print -p list10
list10 = (200, 201, 202, 203, 204, 205, 206, 207, 208, 209)

(dbx) print -p list10[5]
list10[5] = 205

(dbx) print -p list10[1..100:2]
list10[1..100:2] =
[1] = 202
```

```
[3] = 204
[5] = 206
[7] = 208
```

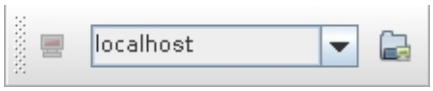
有关美化输出和调用样式的美化输出器的一般信息，请参见《Oracle Solaris Studio 12.4：使用 dbx 调试程序》中的“使用美化输出”和 `dbx` 帮助文件中的主题 `prettyprint`。

dbxtool 更改

dbxtool 是独立的调试器 GUI。Oracle Solaris Studio 12.3 中提供的 dbxtool 的所有功能仍会保留，但是 dbxtool 有与 Oracle Solaris Studio IDE 相似的新外观。

向 dbxtool 添加了以下功能：

- **"Debug Recent"（调试近来的）按钮** – dbxtool 现在具有 "Debug Recent"（调试近来的）按钮以及下拉调试历史记录列表，从而您可以选择使用不同参数的代码的不同运行。按 "Debug Recent"（调试近来的）按钮而不选择特定运行将会调试最近的目标和参数。
- **Code Assistance（代码帮助）** – 您调试代码时，dbxtool 在编辑器中自动启用了代码帮助。虽然您仍不能在 dbxtool 中直接重新编译代码，您可以通过编辑器中代码帮助的帮助来修复代码。要禁用代码帮助，请右键单击 "Projects"（项目）标签中的调试目标，然后取消选择 "Code Assistance"（代码帮助）选项。或者，您可以启动 dbxtool 并关闭代码帮助。有关代码帮助的更多信息，请参见“代码帮助改进” [55]。
- **`--disable-code-assistance` 选项** – 启动 dbxtool 时，如果不需要代码帮助或者如果内存使用有问题并且不需要代码帮助，可以指定 `--disable-code-assistance` 选项。
- **远程主机工具栏** – 现在可以选择使用远程主机工具栏在 dbxtool 中管理和添加远程主机，缺省情况下启用该工具栏。



- **"Projects"（项目）标签中的调试目标** – dbxtool 现在自动在一个位置显示所有调试目标。启动调试会话后，将在 "Projects"（项目）标签中创建和显示调试目标。目标信息存储在 `userdir` 目录中，并在 dbxtool 运行之间永久存在。展开调试目标可以显示根源目录中的文件夹和文件，它们来自构建的二进制文件。

♦ ♦ ♦ 第 6 章

Oracle Solaris Studio IDE

Oracle Solaris Studio IDE 为喜欢图形编程环境的用户集成了 Oracle Solaris Studio 的许多组件。

本章阐述下列主题：

关于 Oracle Solaris Studio IDE

Oracle Solaris Studio 提供了图形集成开发环境 (integrated development environment, IDE)，其生成于 NetBeans 平台之上，配置为使用 Oracle Solaris Studio C、C++ 和 Fortran 编译器、`dmake` 分布式 make 命令和 `dbx` 调试器。IDE 还与分析套件的一些分析器工具集成，从而您可以分析代码而不离开 IDE。

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 中的帮助。有关使用 IDE 基本功能的逐步说明，请参见《Oracle Solaris Studio 12.4：IDE 快速入门教程》。

新增和已更改的 IDE 功能

以下是 Oracle Solaris Studio IDE 中添加或更改的功能：

- **调试未在 IDE 项目中的可执行二进制文件（无项目调试）**。可以通过选择 "Debug"（调试）> "Debug Executable"（调试可执行文件）并指定可执行文件的

路径以及运行该可执行文件所需的任何参数或环境变量，来调试可执行文件。还可以导航到 "Favorites"（收藏夹）窗口中的可执行文件，右键单击该文件并选择 "Debug"（调试）。可执行文件不需要是 IDE 项目的一部分，但是，应通过用于生成可执行文件的源代码来查找可执行文件，这样调试器便可以查找调试信息。

- **C++11 支持。**如果代码使用 C++11 标准并且 C++ 编译器实现了 C++11 标准，则可以在 IDE 中启用 C++11 支持。这样，您可以为 "auto" 说明符等功能使用代码帮助。要为项目启用 C++11 支持，请右键单击该项目，选择 "Properties"（属性），然后选择 "Build"（生成）> "C++ Compiler"（C++ 编译器）> "C++ Standard"（C++ 标准）> "C++11"。要仅在单个文件上启用 C++ 支持，请右键单击该文件，选择 "Properties"（属性），然后选择 "C++ Compiler"（C++ 编译器）> "C++ Standard"（C++ 标准）> "C++11"。
- **内存使用情况改进。**大项目的内存使用情况降低了 50%。
- **更快搜索。**"Find Usages"（查找使用实例）快得多并且具有改进的界面。"Find Usages"（查找使用实例）现在在后台运行，这样在搜索大量文件时您还可以执行其他任务。结果立即显示在单独的 "Usages"（使用实例）面板中，并随着不断找到搜索结果继续显示它们。"Usages"（使用实例）面板提供进度指示器以及递增的实例数。您可以随时停止搜索，系统将保存截止至该时间点的搜索结果。您可以在搜索命中内容之间进行导航，将视图从逻辑更改为物理，并使用不同的设置再次运行 "Find Usages"（查找使用实例）。此外，还可以向搜索添加过滤器并在注释中搜索。
- **轻量级部分重新解析。**因为重新解析方面的改进，编辑具有大的相关项集的源文件现在要快得多。重新设置格式、在函数主体内进行编辑以及添加注释或空格等格式不再导致重新解析整个项目。某些更改会导致 IDE 重新解析整个项目。
- **调试器断点组。**您可以使用多个不同类别对断点分组，例如按文件、按项目、按类型、按语言等。在 "Window"（窗口）> "Debugging"（调试）> "Breakpoints"（断点）窗口中，单击 "Select breakpoint groups"（选择断点组）图标并选择断点组。将根据您的选择对断点进行排列。
- **窗口管理和分组。**除了单个窗口外，您还可以对窗口组执行操作。对于每个窗口所属的组，您可以将其最小化/还原、拖动到新位置、浮动在单独的窗口中或者停靠回 IDE 窗口。例如，通过单击组右侧的 "Minimize Window Group"（最小化窗口组）按钮，可以最小化左上部的 "Projects"（项目）、"Files"（文件）、"Classes"（类）和 "Services"（服务）窗口。在组的标签区域内单击右键来选择组的选项或者选择 "Window"（窗口）> "Configure Windows"（配置窗口）。用于控制窗口行为的选项位于 "Tools"（工具）> "Options"（选项）> "Appearance"（外观）> "Windows"（窗口）中。
- **将文件路径复制到剪贴板。**您可以通过将鼠标光标悬停在 IDE 中的任意文件上并按 Alt+Shift+L 组合键将文件路径复制到剪贴板中。
- **启动器功能。**请参见“IDE 中的新启动器功能” [53]。
- **代码编辑器的改进。**请参见“IDE 代码编辑器改进” [54]。
- **代码帮助的改进。**请参见“代码帮助改进” [55]。
- **面包屑。**请参见“使用面包屑导航” [57]。
- **IDE 的 "Action Items"（操作项）窗口现在与 C/C++ 项目配合使用。**"Action Items"（操作项）窗口显示您已经使用 `TODO` 或 `FIXME` 或 `Pending` 或 `<<<<<<<<<` 等注释进行了标记的源文件中的行。将在 "Tools"（工具）> "Options"（选项）> "Team"（组）> "Action Items"（操作项）中指定检测到的字符串。对于 C/C++ 项

目，"Action Items"（操作项）窗口还显示您最后生成项目时发生的编译错误和警告。

- **对版本控制系统的改进支持：**
  - Git 资源库支持。
  - 本地历史记录：恢复已删除的和新的 "History"（历史记录）标签。
  - 搁置更改：允许您保留本地更改（搁置）并开始使用不同功能。对于 Mercurial 和 Subversion，请参见 "Team"（团队开发）> "Shelve Changes"（搁置更改）菜单选项。
  - Mercurial 增强功能：对分支和标记以及队列的基本支持。
- **新的 solstudio 命令行选项。**可以使用 `--open-group` 和 `--close-group` 选项在 IDE 启动时打开和关闭项目。
- **书签更新。**可以使用 "Window"（窗口）> "IDE Tools"（IDE 工具）> "Bookmarks"（书签）打开 "Bookmarks"（书签）视图，可以使用 Ctrl-Shift-M 在您的文件中创建书签。
- **通过按 Ctrl-空格键在 "Search"（搜索）栏中自动完成。**在编辑器中进行搜索时，可以自动完成您的搜索词条，与您在编辑器中得到的一样。
- **“主项目”的概念不再用于大多数任务。**仍可以使用 "Run"（运行）> "Set Main Project"（设置主项目）设置主项目。
- **可以更容易地查找选项。**"Tools"（工具）> "Options"（选项）对话框提供一个搜索框来帮助更容易地查找选项。
- **工具栏界面改善。**主工具栏通过显示下拉列表（允许您访问不可见的按钮）指示一些工具栏项何时不可见。以前，如果您启用了太多工具栏，则无法看见所有工具栏。
- **使用现有代码编译项目的单个文件。**
- **每个项目的 C/C++ 格式样式。**在项目属性中选择特定于项目的格式样式时，可以指定 C 格式样式、C++ 格式样式和 C/C++ 头格式样式。

IDE 中的新启动器功能

您可以创建“启动器”，以便使用不同的参数从项目上下文菜单轻松地运行项目，或者从脚本启动项目。通常，当您从 IDE 中运行应用程序时，将执行在项目属性中指定为运行命令的可执行文件。如果创建了启动器，您可以指定要运行的多个命令，然后从上下文菜单中选择这些命令。还可以使用启动器进行调试。

要创建启动器，请转至 `nbproject/private` 文件夹并定制启动器文件 (`launchers.properties`)。

在 "New Launchers File"（新建启动器文件）对话框中，如果要在项目的 `nbproject/private` 子文件夹中存储启动器的定义，请选择选项 "File Privacy"（文件保密性）。如果项目与其他开发者共享，此选项很有用，尤其是在版本控制系统。您可以忽略 VCS 中的 `nbproject/private`，从而您共享项目时将不包括它。如果存在专用启动器文件，专用文件中的启动器将覆盖公共启动器文件中相同名称的启动器。

单击 "Finish"（完成），将在 IDE 编辑器中打开 `launchers.properties` 文本文件。可以指定要运行的命令，并显示要在 IDE 中为运行这些命令而显示的名称。例如，对于 IDE 的 C/C++ 样例应用程序调用的参数，您可以将以下内容添加到 `launchers.properties` 文件中：

```
launcher1.runCommand="${OUTPUT_PATH}" "arg 1" "arg 2" "arg 3" "arg 4"
launcher1.displayName=Four Args

launcher2.runCommand=../dist/Debug/OracleSolarisStudio-Solaris-x86/arguments_1 "arg 1"
launcher2.displayName=One Arg

launcher3.runCommand=/bin/sh runMyProgram.sh
```

文件 `runMyProgram.sh` 可能是一个脚本，例如用于设置环境变量或执行您需要的任何操作。

如果您要使用运行脚本的启动器调试应用程序，还必须为该启动器指定选项 `symbolFiles`，以便调试器可以调试应用程序而非用于运行脚本的 shell。对于上述 `launcher3` 示例，可按如下方式添加该选项：

```
launcher3.runCommand=/bin/sh runMyProgram.sh
launcher3.symbolFiles=${LINKER_OUTPUT}
```

添加完启动器选项后，保存 `launcher.properties` 文件。

然后可以通过右键单击项目并选择这些命令中的一个来运行这些命令。在上面的示例中，您可以右键单击，选择 "Run As"（运行方式）命令并选择命令的名称。例如，"Run As"（运行方式）> "Four Args" 或 "Debug As"（调试身份）> /bin/sh runMyProgram.sh。

IDE 代码编辑器改进

代码编辑器具有许多改进，包括下列各项：

矩形块选择

在编辑器中，可以通过按 Ctrl+Shift+R 组合键或单击编辑器工具栏中的 "Toggle Rectangular Selection"（开启/关闭矩形选择）图标来启用用于选择文本矩形块的模式。通过此模式，您可以选择文本中多个行的一部分，而不包括每行的开头或结尾。例如，可以选择注释块（而不包括开始的星号）并复制该块，或者可以在一个操作中已注释的代码的一部分中选择和删除所有星号，而不是逐行删除星号。例如，还可以从文本表中

复制、移动或删除列。如果在矩形选定文本中键入字符，将在每行上复制该字符，替代选定文本。

剪贴板历史记录

您可以查看复制到桌面剪贴板的最后九个文本缓冲区，并选择一个进行粘贴。将光标放在想要插入文本的位置，按 Ctrl+Shift+D 可打开包含剪贴板条目的弹出窗口。使用箭头键可导航剪贴板缓冲区，并在缓冲区列表下的窗口中查看全部内容。要粘贴缓冲区的内容，请键入缓冲区编号或者在选择缓冲区后按 Enter 键。请注意，此缓冲区包含从桌面上任意窗口复制过来的内容，而不仅仅是 IDE 的内容。

查找/替换增强功能

"Find/Replace"（查找/替换）功能现在作为一个整体位于编辑器窗口底部的 "Find"（查找）工具栏中，而不是一个单独的用于替换的对话框。"Replace"（替换）字段和按钮显示在 "Find"（查找）字段和按钮下的工具栏中。按 Ctrl+F 可激活 "Find"（查找）工具栏，按 Ctrl+H 可激活 "Replace"（替换）功能。

代码帮助改进

IDE 提供了关于代码帮助的许多改进，包括下列各项：

- “共享代码帮助高速缓存” [55]
- “代码帮助的新项目属性选项” [56]
- “搜索文件系统以查找 C/C++ 头文件” [56]

共享代码帮助高速缓存

解析 C/C++ 源代码时，IDE 会把解析结果存储到磁盘上的代码帮助高速缓存中。打开项目时，IDE 将检查高速缓存是否为最新的。如果高速缓存是最新的，则 IDE 不会解析项目，而仅会装入从代码帮助高速缓存导航代码所需的数据。

缺省情况下，代码帮助高速缓存位于 `${`*userdir*`}/var/cache` 文件夹中，其中 `${`*userdir*`}` 表示 IDE 用户目录。Oracle Solaris 中的用户目录位于用户的 `$HOME/.solstudio/ide-`*<release>* 中。用户目录中的高速缓存不能共享或复制到其他位置。

但是，如果将代码帮助高速缓存置于项目内部，则可以将其复制到其他计算机，前提是该计算机满足以下要求：

- 计算机的操作系统与代码解析所在的操作系统相同
- 项目使用的工具集合位于计算机上的相同位置

指示 IDE 将代码帮助高速缓存放在项目元数据内部：

1. 向以下任一项添加行 "`cache.location=nbproject/private/cache`"：
   - 项目属性文件 (`nbproject/project.properties`)
   - 私有属性文件 (`nbproject/private/private.properties`)

   项目属性文件和私有属性文件之间的差别在于：缺省情况下公共文件 (`nbproject/project.properties`) 通过版本控制系统在 IDE 中共享，而私有文件 (`nbproject/private/private.properties`) 则不是。因此，如果您修改了私有属性，则需要将私有属性文件与其他计算机上的相同文件同步。如果修改了项目属性文件，则版本控制系统可以自动将其与其他计算机上的相应文件同步。
2. 修改属性文件之后，关闭并重新打开项目。

   IDE 将解析项目，并将代码帮助高速缓存放入项目元数据的专用子目录中。
3. 关闭项目并归档 nbproject/private/cache 或将其复制到共享位置。

   如果未在复制或压缩之前关闭项目，某些数据将不会刷新到高速缓存。

代码帮助高速缓存即可复制到其他计算机上的其他项目中并进行使用，而不用等待 IDE 解析项目。如果要将高速缓存复制到的计算机上有一些较新的文件，则仅将解析较新的文件。

注 - 如果需要在运行不同操作系统或不同编译器的计算机之间共享代码帮助高速缓存，则必须为每个操作系统组合和编译器集合创建单独的高速缓存。

代码帮助的新项目属性选项

对于从现有源代码或从二进制文件创建的项目，IDE 目前提供了以下项目属性，使您可以更方便地在版本控制系统中使用项目。

瞬态宏	可以提供可变宏的列表（-D 选项），这些宏依赖于时间、日期或特定环境。这些宏值不会随项目的公共元数据一起存储。
用户环境变量	您可以提供环境变量列表，项目使用这些环境变量来传递特定于系统的路径。这些宏环境变量值不会随项目的公共元数据一起存储。对于现有代码或二进制文件中的项目，您可以指定要在存储项目元数据时使用的环境变量列表。当 IDE 存储编译器选项并且选项值与变量值一致时，将改为编写宏。

搜索文件系统以查找 C/C++ 头文件

如果从现有源代码创建其源代码尚未生成并且不包含任何调试信息的项目，则在配置代码帮助时 IDE 可能会遇到问题。在这种情况下，您可以指定在 "Configure Code Assistance"（配置代码帮助）向导中使用一种特殊的模式，即在文件系统中搜索 C/C

++ 头文件。在这种模式下，IDE 会尝试在文件系统中搜索头文件，以解析执行失败的 include 指令。该向导会让您输入头文件的搜索路径。缺省情况下，该路径是项目的源根目录。

使用面包屑导航

通过面包屑，可以跟踪您在 IDE 内的位置。面包屑导航栏位于源代码编辑器窗口下方，为您显示与光标位置有关的嵌套元素。每个元素都标有一个图标，以标识其类型。有关图标及其含义的完整列表，请参见 IDE 中的 "Icons Used in the Classes and Navigator Windows"（类和导航器窗口中使用的图标）帮助页面。

要使用面包屑，请执行以下操作之一：

- 单击面包屑栏中的箭头以显示嵌套语句或成员的列表，然后选择一个在源代码编辑器中前往相应的位置。
- 单击面包屑栏中的项目可回顾光标位置的历史记录。
- 按 Alt+左箭头和 Alt+右箭头可以在光标位置的历史记录中前后移动。

♦ ♦ ♦ 第 7 章

7 OpenMP API 和线程分析器

本章介绍此 Oracle Solaris Studio 发行版中 OpenMP API 支持和线程分析器的更改。

OpenMP

本节讨论 OpenMP API 的新增功能和更新。

OpenMP 4.0 支持

此发行版支持 OpenMP API 版本 4.0 中引入的新功能，这是对 OpenMP API 标准语言规范的主要升级。此发行版中 C、C++ 和 Fortran 编译器支持的新 OpenMP 4.0 功能包括下列各项：

- **错误处理** – OpenMP 4.0 定义错误处理功能以改进存在运行时错误时 OpenMP 应用程序的弹性和稳定性。可以使用条件取消和用户定义的取消点安全地中止并行 OpenMP 执行。
- **线程相似性** – OpenMP 4.0 提供了机制来定义执行 OpenMP 线程的位置，实现更好的位置、更少的假共享以及更多内存带宽。
- **任务扩展** – OpenMP 4.0 提供了对基于任务的并行支持的多个扩展。任务可以组合来支持深度任务同步。通过任务相关性规范来支持任务到任务同步。
- **对 Fortran 2003 的支持** – Fortran 2003 标准添加了许多现代计算机语言功能。在 OpenMP 规范中具有这些功能，用户可以并行化符合 Fortran 2003 的程序。
- **按顺序一致的原子** – 添加了一个子句，从而在以原子方式访问特定存储位置时可以执行顺序一致性。
- **用户定义的规约** – 除了使用基本语言操作符和内部过程的规约，OpenMP 4.0 支持用户定义的规约。编程人员可以使用 `declare reduction` 指令定义定制规约；可以在 `reduction` 子句中指定这些规约。
- **新环境变量** `OMP_DISPLAY_ENV` – `OMP_DISPLAY_ENV` 环境变量可用于显示与 OpenMP 环境变量关联的内部控制变量 (Internal Control Variables, ICV) 的值。

注 - 在此发行版中，接受 OpenMP 4.0 设备和 SIMD 构造。但是，将在主机设备上执行所有代码，SIMD 构造可能不会导致使用 SIMD 指令。

有关详细信息，请参见《Oracle Solaris Studio 12.4：OpenMP API 用户指南》。

有关 OpenMP 4.0 功能的更多信息，请参见《OpenMP Application Program Interface Version 4.0, July 2013》 (http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf)（《OpenMP 应用程序接口版本 4.0，2013 年 7 月》）和《OpenMP 4.0.1 Examples, February 2014》 (http://openmp.org/mp-documents/OpenMP_Examples_4.0.1.pdf)（《OpenMP 4.0.1 示例，2014 年 2 月》）。

与 OpenMP 相关的增强功能

下列项是 Oracle Solaris Studio 中 OpenMP 的其他增强功能。

- **新的线程缺省数** – 用于执行并行区域的线程的缺省数量从两个线程更改为计算机上的可用内核数，上限为 32。
- **堆栈溢出检测和诊断** – 现有 C、C++ 和 Fortran 编译器选项 `-xcheck=stkovf` 已扩展，可以选择启用运行时错误诊断。

  语法如下所示：

  ```
  -xcheck=stkovf [:detect | :diagnose]
  ```

  如果指定了 `:detect`，则通过执行通常与错误关联的信号处理程序来处理检测到的堆栈溢出错误。

  如果指定 `:diagnose`，则通过捕获关联的信号来处理检测到的堆栈溢出错误并通过调用 stack_violation(3C) 来诊断错误。如果诊断到堆栈溢出错误，则会向 `stderr` 输出错误消息。这是没有指定任何内容时的缺省行为。

  有关 `-xcheck=stkovf` 编译器选项的更多信息，请参见 cc(1)、CC(1) 或 f95(1) 手册页。

线程分析器

线程分析器是一个非常强大的工具，可以分析多线程程序的执行并检测常见线程错误，例如数据争用和死锁。使用线程分析器，可以更容易地调试多线程应用程序，从而生产率更高。可以将线程分析器与使用以下标准和框架中的一个或组合编写的程序配合使用：

- POSIX 线程 API
- Oracle Solaris 线程 API
- OpenMP 指令

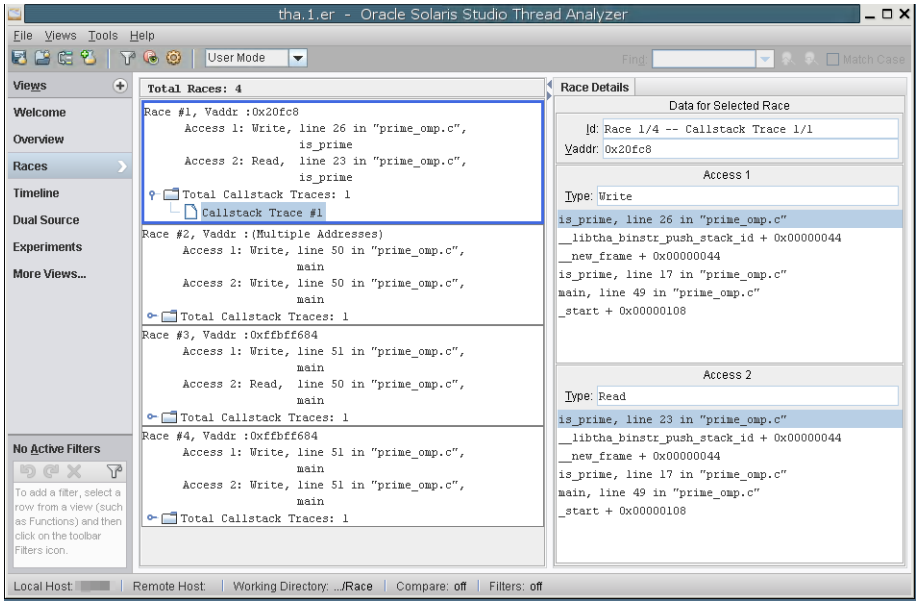
有关线程分析器的更多信息，请参见 `tha(1)` 手册页和《Oracle Solaris Studio 12.4：线程分析器用户指南》。

在 Oracle Solaris Studio 的此发行版中，添加了以下功能：

- 使用 Discover 执行数据争用检测的二进制文件检测时，将为数据争用访问显示全部调用堆栈。
- Oracle Solaris 10 中引入了对 `atomic_ops` API 的支持。
- 对 `collect -r terminate` 的支持，其在死锁检测处于打开状态并且发生实际死锁时强制终止进程。
- 线程分析器的用户界面已经重新设计，改进了数据显示并可以与性能分析器的用户界面一起导航。有关更多信息，请参见“性能分析器导航” [18]。

下图显示使用 `discover` 检测了二进制文件后，线程分析器如何显示双向调用堆栈。

图 7-1 线程分析器窗口



♦ ♦ ♦ 第 8 章

8 其他更改

本章介绍 Oracle Solaris Studio 软件的其他组件的新增和更改的功能。

编译器中的更改

以下部分介绍编译器中的更改，包括以下主题：

编译器通用的新增和更改的功能

下面列出了自上一发行版起对 C、C++ 和 Fortran 编译器进行的更改。可在编译器手册页中找到详细信息。特定于 C++ 编译器的更改已在第 2 章 C++ 编译器中进行详细介绍。

新硬件上的应用程序性能

每一个 Oracle Solaris Studio 发行版都包括对 Oracle Sun 硬件服务器的性能改进。此发行版中包括对 SPARC T5、SPARC M5、SPARC M6、SPARC M10 及 Intel Ivy Bridge 和 Haswell 编译器的扩展支持以及库性能改进（首先在特定于 Oracle Solaris Studio 12.3 1/13 平台的增强功能发行版中提供）。

- x86 上针对 Ivy Bridge 和 Haswell 处理器的新 `-xarch`、`-xchip` 和 `-xtarget` 值。
- 针对 SPARC T5、M5、M6 和 M10+ 处理器的新 `-xarch`、`-xchip` 和 `-xtarget` 值。
- 支持 Ivy Bridge 和 Haswell 汇编程序指令。

- 支持 Ivy Bridge 和 Haswell 内部函数（可在 *solstudio-install-dir*`/lib/compilers/include/cc/sys</immintrin.h` 中找到）。
- x86 和 x64 体系结构上 `-xarch=generic` 的缺省值设置为 `sse2`。有关更多信息，请参见“x86 缺省浮点行为的更改” [64]。

x86 缺省浮点行为的更改

与以前的发行版相比较，Oracle Solaris Studio 12.4 在 x86 系统上使用缺省编译器值编译的程序中的浮点计算结果可能会稍有不同。即使在相同的硬件和操作系统上，也可能会得到不同的结果。这是因为对于所有地址空间模型和平台而言，现在的缺省指令集体系结构都是 SSE2。

在 Oracle Solaris Studio 12.4 和以前的发行版中，针对 32 位 Oracle Solaris 的缺省地址空间模型是 `-m32`。在 Linux 上，对于 64 位硬件的缺省值为 `-m64`。在所有平台上，您都可以分别使用 `-m32` 和 `-m64` 来对 32 位地址空间模型或 64 位地址空间模型进行编译。

编译器使用 `-xarch` 选项优化代码，以确定硬件中实现了哪些指令并因此适合进行代码生成。在以前的 Oracle Solaris Studio 发行版中，`-m32` 对应的缺省值是 `-xarch=386`，`-m64` 对应的缺省值是 `-xarch=sse2`。

对于 32 位寻址的 x86，如果未使用 `-xarch`、`-xnative` 或 `-fast` 指定或隐式指定任何代码生成选项，则代码生成选项将是 `-xarch=sse2` 而不是 `-xarch=386`。因此，使用浮点运算并且使用缺省 `-xarch` 编译的程序可能会生成不同的浮点结果。

新 x86 缺省 `-m32 -xarch=sse2` 与以前的缺省 `-m32 -xarch=386` 实现相同的 ABI。浮点操作数和结果在 x87 浮点寄存器中传递。但是，以下单精度和双精度浮点操作通常在 sse2 寄存器中执行。

```
+
-
*
/
sqrt
convert
```

仍将 x87 寄存器用于长双精度操作和硬件初等超越函数求值。

其他编译器更改

- 在 x86 上支持 `-xlinkopt`。针对为现代 Intel 处理器优化的大型企业应用程序进行了模块间、过程代码间排序优化。对于大型应用程序，在充分优化的二进制文件上可实现高达 5% 的性能提升。
- x86 的新编译器选项：`-preserve_argvalues` 将基于寄存器的函数参数的副本保存在堆栈中。

- 增强的 - xs 选项用于控制可执行文件大小与保留对象文件的需求之间的平衡以便进行调试。
- 在 Linux 上支持 -xanalyze 和 -xannotate。
- 支持 -fopenmp 作为 -xopenmp=parallel 的同义词。
- 支持 SPARC M10 值 -xchip=sparc64x、-xtarget=sparc64x、-xarch=sparcace。
- 支持 SPARC M10+ 值 -xchip=sparcxplus、-xtarget=sparc64xplus、-xarch=sparcaceplus。
- 支持 SPARC M10+ 处理器通用的 SPARC 指令集扩展的 -xarch=sparc4b 和 -xarch=sparc4c。
- 支持 x86 平台上的 Ivy Bridge 值 -xchip=ivybridge、-xtarget=ivybridge 和 -xarch=avx_i。
- 支持 x86 平台上的 Haswell 值 -xchip=haswell、-xtarget=haswell 和 -xarch=avx2。
- 编译器的新选项：
  - -g1 – 生成文件和行号以及在事后调试期间视为至关重要的简单参数信息。
  - -xdebuginfo – 控制发出多少调试和监测信息。
  - -xglobalize – 控制文件静态变量的全局化，但是不控制函数的全局化。
  - -xinline_param – 允许更改编译器用来确定何时内联函数调用的试探式方法。
  - -xinline_report – 可生成在编译器内联函数时写入标准输出的报告。
  - -xipo_build – 避免在初始传递期间通过编译器进行优化而仅在链接时优化，从而缩短编译时间。
  - -xkeep_unref – 保留未引用的函数和变量的定义。
  - -xpatchpadding – 在开始每个函数之前保留一个内存区域。
  - -xsegment_align – 使驱动程序在链接行上包括特殊映射文件。
  - -xthroughput – 指示当许多进程同时在系统上运行时运行应用程序。
  - -xunboundsym – 指定程序是否包含对动态绑定符号的引用。

C 编译器

C 编译器的更改包括“编译器通用的新增和更改的功能” [63]中介绍的更改，以及以下其他更改。

- 在 x86 上支持 -xregs=float。
- C 编译器的新选项：
  - -ansi – 等效于 -std=c89。
  - -pedantic – 强制严格遵循非 ANSI 构造的错误/警告。
  - -staticlib – 与 -library=sunperf 一起使用时，将与 Sun 性能库静态链接。
  - -std – 指定 C 语言标准。-std=c11 是缺省编译器模式。

  - `-temp` – 为临时文件定义目录。
  - `-xlang` – 覆盖 `-std` 标志指定的缺省 `libc` 行为。
  - `-xprevise` – 生成可使用代码分析器查看的源代码的静态分析。
- 支持 C11 功能：
  - `_Static_assert`
  - 匿名结构/联合
  - `_Noreturn` 函数断言
  - `_Thread_local` 存储说明符
  - `_Alignof` 运算符
  - `_Alignas` 对齐说明符
  - C11 指定的 UCN 中的字符集

有关更多信息，请参见 `cc` 手册页和《Oracle Solaris Studio 12.4：C 用户指南》。

Fortran 编译器

Fortran 编译器通过针对 Fortran77、Fortran90 和 Fortran95 标准的创纪录运行时性能和兼容性选项来支持技术和科学应用程序开发。包括大多数 Fortran 2003 功能和 OpenMP 4.0 支持。Fortran 编译器与 C 和 C++ 编译器使用相同的高性能代码生成技术，从而确保结果应用程序为最新的基于 SPARC 和 x86 的 Oracle 系统生成最高性能的并行代码。

Fortran 编译器更改包括"编译器通用的新增和更改的功能" [63]中所述的更改。

下面列出了此 8.7 发行版的 Fortran 编译器的新增和更改的功能。有关更多信息，请参见 `f95` (1) 手册页和《Oracle Solaris Studio 12.4：Fortran 用户指南》。

- `-xM` 选项可用于自动生成 makefile 相关项。与新的 `- keepmod=yes` 选项一起，它允许使用模块在 Fortran 应用程序上进行最优递增构建。新的 `- keepmod` 选项用于保留编译时不更改的模块。缺省为 `-xkeepmod=yes`，它取代每次创建新模块文件时的旧行为，即使没有对先前编译进行任何更改也是如此。
- 使用模块的应用程序的编译时间显著改进，不再出现由于模块处理导致的内存溢出。
- `#pragma ident` 可用于源文件中，标识编译的对象的源版本。
- 支持延迟类型参数（冒号）作为声明中使用的字符类型中的 LEN 类型参数。例如：

  ```
  character(LEN=:), pointer :: str
  ```
- 支持过程指针。
- 针对 `ISO_C_BINDING` 模块支持 Fortran 2003 函数 `C_F_PROCPOINTER ()`。`C_FUNLOC ()` 函数扩展为支持过程指针作为参数。
- 完全支持面向对象的 Fortran。现在已允许具有以下属性的类型限制过程：

  - GENERIC
  - DEFERRED
  - NON-OVERRIDABLE
  - PASS
  - NOPASS
- 支持 Fortran 2003 功能让派生类型和通用函数具有相同的名称。
- 支持 Fortran 2008 功能将 TARGET 对象传递到 INTENT(IN) 指针哑元。
- 将支持扩展到允许在 Fortran 2003 标准中指定的初始化表达式中使用所有基本内部函数（每个参数本身都是一个初始化表达式）。以前，在该上下文中使用的基本内部函数限于仅返回整数和字符类型的函数。
- 支持 -fserialio，该选项指定程序不能一次在多个线程中执行 I/O。

性能库更改

Oracle Solaris Studio 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。Oracle Solaris Studio 性能库以来自 http://www.netlib.org 上的 Netlib 的公共域子例程集合为基础。Oracle 增强了这些公共域子例程，并将其捆绑成 Oracle Solaris Studio 性能库。

在此发行版中，进行了以下更改：

- 优化了 SPARC T5、M5 和 M6 平台及 SPARC 64X+ 上的性能。
- Oracle Solaris Studio 性能库中的 LAPACK 升级为版本 3.4.2。在 Oracle Solaris Studio 性能库中实现了 LAPACK 3.4.2 中的所有新功能，包括以下功能：
  - 额外的精确迭代优化线性求解器 (3.2)
  - 新的快速且准确的 Jacobi SVD。(3.2)
  - 矩形全包装 (Rectangular Full Packed, RFP) 格式例程。(3.2)
  - Pivoted Cholesky。(3.2)
  - 用于开发快速单精度硬件的混合精度迭代优化子例程。(3.2)
  - 计算完整的 CS 分解。(3.3)
  - 3 级 BLAS 对称不定式求解和对称不定式转位。(3.3)
  - xGEQRT：QR 因式分解（改进了界面）。(3.4)
  - xGEQRT3：递归 QR 因式分解。(3.4)
  - xTPQRT：免通信 QR 顺序内核 (3.4)

  引入这些功能的 LAPACK 版本显示在圆括号中。

索引

Z