Oracle® Solaris Studio 12.4: 概要

Part No: E57208
2014 年 12 月

ORACLE®

目次

このドキュメントの使用方法

- **概要** – この製品で利用できる多数のツール、コンパイラ、およびプログラミングライブラリについて説明します。このマニュアルでは、これらのツールの使用方法に関する詳細情報は提供していませんが、開発者がそれらをどのように併用して開発中のソフトウェアアプリケーションを編集、構築、および解析できるかを示しています。
- **対象読者** – アプリケーション開発者、システム開発者、設計者、サポートエンジニア
- **必要な知識** – なし

製品ドキュメントライブラリ

製品ドキュメントライブラリは http://docs.oracle.com/cd/E37069_01 にあります。

システム要件および既知の問題は、『Oracle Solaris Studio 12.4: リリースノート 』に記載されています。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info (聴覚に障害をお持ちの場合は http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs) を参照してください。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

Oracle Solaris Studio 12.4 の概要

Oracle Solaris Studio ソフトウェアは、Oracle Solaris™ および Linux プラットフォームでの C、C++、および Fortran 開発用のソフトウェア開発ツールセットであり、SPARC® プロセッサや x86 および x64 プロセッサを搭載したマルチコアシステムをサポートしています。

Oracle Solaris Studio ソフトウェアの紹介

Oracle Solaris Studio は 2 つのツールスイートから構成されています。コンパイラスイートと分析スイートです。各スイートのツールは相互に連携して、単独、マルチスレッド、または分散型のアプリケーションの開発向けに最適化された開発環境を提供するように設計されてします。

Oracle Solaris Studio は、SPARC または x86 および x64 プラットフォーム上の Oracle Solaris 10 または Oracle Solaris 11、あるいは x86 および x64 プラットフォーム上の Oracle Linux で実行される C、C++、および Fortran アプリケーションを開発するために必要なすべてのものを備えています。コンパイラおよび分析ツールは、Oracle Sun システム上でのアプリケーションの動作を最適にするように設計されています。

特に、Oracle Solaris Studio コンパイラおよび分析ツールは、新しいマルチコア CPU (SPARC T5、SPARC M5、SPARC M6、SPARC M10、SPARC M10+、Intel Ivy Bridge、および Haswell プロセッサを含む) および古いプロセッサ (SPARC T4、SPARC T3、Intel® Xeon®、および AMD Opteron™ プロセッサを含む) の機能を活用するように設計されています。Oracle Solaris Studio を使用すると、これらのプラットフォームで並列および同時実行のソフトウェアアプリケーションをより簡単に作成できます。

Oracle Solaris Studio のコンポーネントには次のものが含まれます。

- グラフィカル環境でのアプリケーション開発用の IDE。Oracle Solaris Studio IDE は、ほかの複数の Oracle Solaris Studio ツールを統合しています。
- コマンド行で、または IDE 経由でコードをコンパイルするための C、C++、および Fortran コンパイラこれらのコンパイラは、Oracle Solaris Studio デバッガ (`dbx`) と緊密に連携す

るように設計され、特定のプロセッサ向けにコードを最適化するためのオプションを備えています。

- アプリケーションに高度なパフォーマンスおよびマルチスレッド機能を追加するライブラリ。
- コマンド行で、または IDE 経由で分散コンピューティング環境でのコードを構築するための make ユーティリティー (`dmake`)
- コマンド行で、IDE 経由で、または独立したグラフィカルインタフェース (`dbxtool`) 経由でコードのバグを検出するためのデバッガ (`dbx`)。
- コンパイル時にコード内の静的コードエラーを検出し、実行時にメモリーアクセスエラーおよびコードカバレージエラーを検出するためのコードアナライザツール。
- DTrace などの Oracle Solaris テクノロジを採用し、コマンド行またはグラフィカルインタフェース経由で、デバッグによって検出できないコード内の問題箇所を検出するために使用できるパフォーマンスアナライザツール。
- マルチスレッドプログラムを検査して、データの競合およびデッドロックの原因となるプログラミングエラーを検出するためのスレッドアナライザ。

これらのツールを併用すると、Oracle Sun システム上で実行される Oracle Solaris で高いパフォーマンスが得られるようにアプリケーションを構築、デバッグ、および調整できます。このドキュメントでは、各コンポーネントについて詳しく説明します。

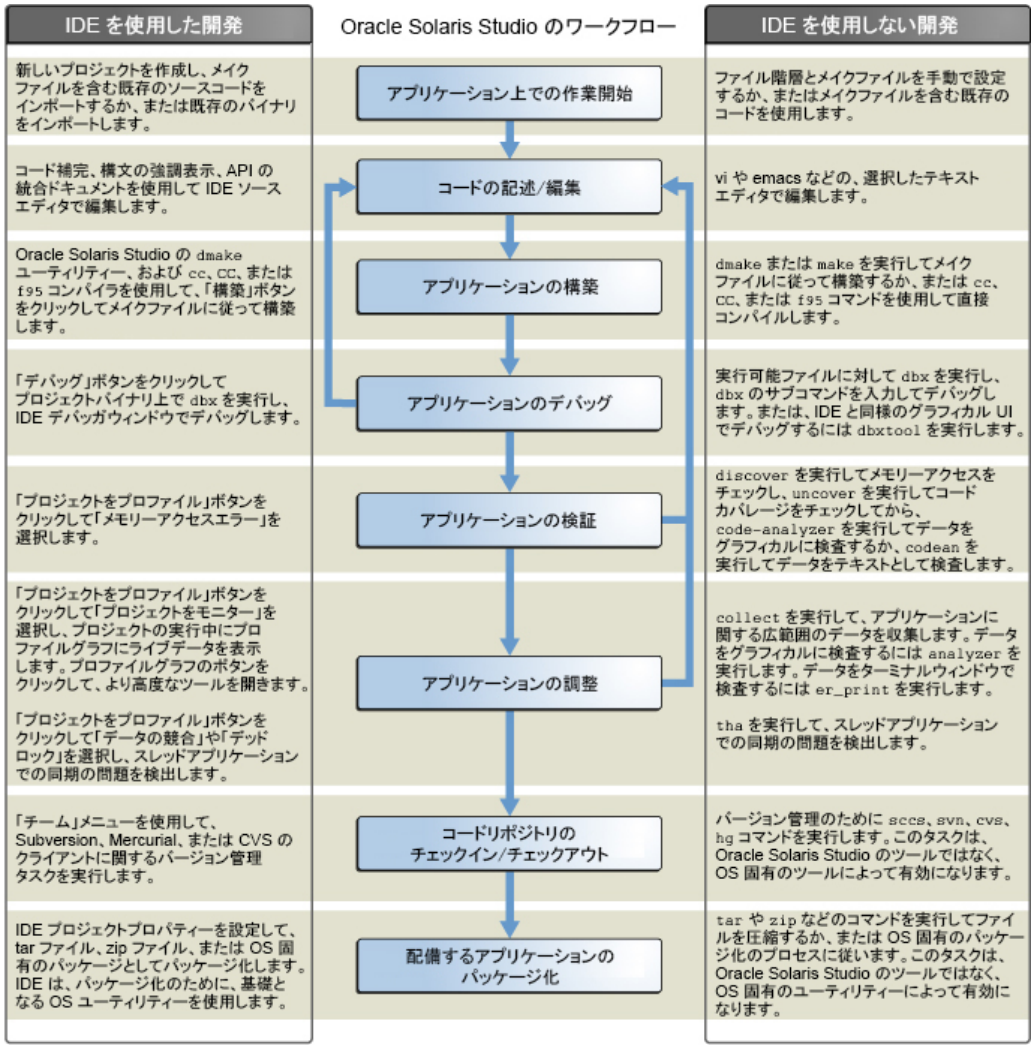
Oracle Solaris Studio の開発者ワークフロー

Oracle Solaris Studio は、Oracle Solaris で実行されるアプリケーションを開発者が作成するために役立つツールを提供しています。これらのツールは、多くの開発タスクを管理するグラフィカル IDE を必要とする開発者、およびソフトウェア開発のあらゆる側面を独自の方法で制御しようとする開発者をサポートできます。

これらのツールは任意の組み合わせで使用できるように設計されているため、必ずしも IDE またはコマンド行を使用する必要はありません。IDE でプロジェクトを作成していても、必要に応じてコマンド行から `dmake` または `make` を使用してプロジェクトのソースを構築できます。IDE で作成したプロジェクトのバイナリに対して、パフォーマンスアナライザを使用できます。IDE はプロジェクトファイルとソースファイルを別個に保管するため、依存関係は存在しません。

熱心な Emacs または vi ユーザーの場合、使い慣れた環境を引き続き使用して IDE を無視しても、Oracle Solaris Studio のコンパイラおよびパフォーマンスツールを採用して、Oracle Sun ハードウェア上の Oracle Solaris でのアプリケーションの動作を最適にできます。

次の図は、グラフィカル IDE を使用して開発する場合と使用しないで開発する場合の、Oracle Solaris Studio ツールでの開発者ワークフローを示しています。



このドキュメントの残りの部分では、Oracle Solaris Studio ソフトウェアのコンポーネントと、それらのコンポーネントがどのように統合されているかについて説明し、それらの使用方法を簡単に示します。

Oracle Solaris Studio IDE

Oracle Solaris Studio IDE はオープンソースの統合開発環境である NetBeans IDE を基にしています。Oracle Solaris Studio IDE には、コア NetBeans IDE、NetBeans C/C++ プラグインモジュール、およびオープンソースの NetBeans IDE では使用できない Oracle Solaris Studio の追加統合コンポーネントが含まれます。

Oracle Solaris Studio IDE では、Oracle Solaris Studio の C、C++、および Fortran コンパイラ、`dmake` 構築ユーティリティー、および `dbx` デバッガを使用します。また、IDE は、Oracle Solaris Studio パフォーマンスユーティリティーを透過的に使用して実行中のプロジェクトのデータを収集するグラフィカルプロファイリングツールも備えています。

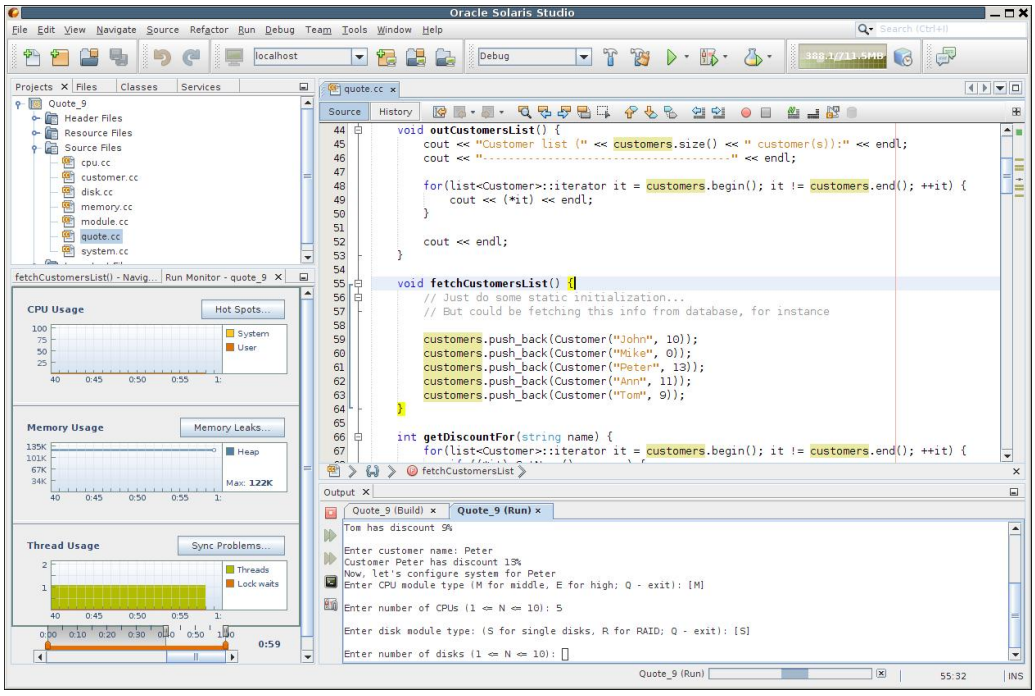
Oracle Solaris Studio IDE を使用すると、テキストエディタおよびコマンド行を使用して開発する場合に比べていくつかの利点があります。

- **コード編集**。構文の強調表示、コード補完、コード要素間のナビゲーション、API の統合ドキュメントおよびマニュアルページにより、コードでの作業効率を高めることができます。
- **コード検査**。何らかのコードを理解しようとしているときや、バグの根本原因を探索しているときは、「シンボルへ移動」、「使用状況を検索」、「クラス」ウィンドウ、「インクルードの階層」、「コールグラフ」などの IDE の機能が役に立つ場合があります。
- **リファクタリング**。プロジェクト内での変数または演算のすべての使用箇所を検索し、すべての出現箇所でその変数または演算の名前を変更して、製品全体でリファクタリングすることができます。リファクタリングを実行する前に、分割画面の UI で変更をプレビューして、個別または一度に承認できます。
- **リモート開発**。リモートサーバーにインストールされている Oracle Solaris Studio のコンパイラおよびツールを使用しているときに、デスクトップシステムで IDE、dbxtool、コードアナライザ、およびパフォーマンスアナライザを実行できます。ツールはリモートサーバー上で実行されると同時に、一般的なリモート表示ソリューションよりもはるかに優れた応答時間で、ローカルシステム上で実行されている IDE またはその他のツールに表示されます。

IDE を起動するには、次のコマンドを入力します。

```
% solstudio
```

次の図は、プロジェクトのモニタープロファイリングツールを表示して、Quote というサンプルプロジェクトを実行している IDE を示しています。



C または C++ または Fortran のプロジェクトは、ソースファイルと、ソースファイルをコンパイルおよびリンクし、結果のプログラムを実行する方法についての関連情報のグループです。IDE では、プログラムが単一のソースファイルに含まれる場合でも、常にプロジェクト内で作業します。IDE は、メイクファイルとメタデータファイルを含むプロジェクトフォルダにプロジェクト情報を格納します。ソースディレクトリが物理的にプロジェクトフォルダ内に存在する必要はありません。

各プロジェクト (既存のバイナリから作成されたプロジェクトを除く) には、IDE がプロジェクトを構築できるようにメイクファイルが必要です。プロジェクトのメイクファイルを IDE によって生成することも、IDE の外部で以前に作成されたメイクファイルを使用することもできます。すでにメイクファイルを含む既存のソース、または構成スクリプトを実行したときにメイクファイルを構築する既存のソースからプロジェクトを作成できます。

プロジェクトの構築、実行、およびデバッグを行うには、ツールバーのボタンをクリックするか、またはメニューコマンドを選択します。IDE は、デフォルトで Studio の C、C++、および Fortran コンパイラ、`dmake`、および `dbx` を使用するように事前構成されています。ただし、システム上に GNU コンパイラが存在し、PATH に含まれている場合、IDE はそれらを容易に検出できます。プロジェクトのプロパティーで「ツールコレクション」を設定することによって、GNU ツールコレクションを使用できます。

Oracle Solaris Studio IDE の使用方法を知るには、IDE の「ヘルプ」メニューを使用するか、F1 キーを押してアクセスできる IDE の統合ヘルプを参照してください。また多くのダイアログボックスに、そのダイアログボックスの使用方法に関する情報を表示するための「ヘルプ」ボタンがあります。

『Oracle Solaris Studio 12.4: IDE クイックスタートチュートリアル 』では、IDE の使用を開始する方法を説明しています。さらに、NetBeans IDE に関するチュートリアル「C/C++ Learning Trail」も、Oracle Solaris Studio IDE の使用方法を知るために役立ちますが、ユーザーインタフェースおよび機能にいくつか違いがあります。特に、デバッグについての NetBeans のドキュメントは Oracle Solaris Studio IDE には当てはまりません。

Oracle Solaris Studio のコンパイラ

Oracle Solaris Studio ソフトウェアには、次の機能を備えた C、C++、および Fortran コンパイラが付属しています。

- C、C++、および Fortran プログラミング言語の最新の規格に準拠しています。
- 指定されたコマンド行オプションに従って、特定のオペレーティングシステム、プロセッサ、アーキテクチャー、メモリーモデル (32 ビットおよび 64 ビット)、浮動小数点演算などを対象とするコードを生成します。
- シリアルソースコードに対して自動並列化を実行し、マルチコアシステムで強化されたパフォーマンスを発揮するバイナリを生成します。
- アプリケーションおよび配備環境に合わせて、コマンド行オプションで指定できる方法で最適化されたコードを生成します。
- ほかの Oracle Solaris Studio ツールによるデバッグまたは解析を強化するためのバイナリを準備します。
- これらの機能を指定するために、すべてのコンパイラで同じコマンド行オプションを使用します。

コンパイル済みのコードを最適化して速度を高め、プロセッサの命令セットおよび機能を最大限に活用するために使用できる Oracle Solaris Studio コンパイラのオプションには、次のものがあります。

-O*n*	1 から 5 までの数にできる *n* によって示される最適化のレベルを指定します。最適化レベルが高いほど、実行時のパフォーマンスの高いバイナリが作成されます。

`-fast`	実行可能コードの速度を高めるために最適なコンパイルオプションの組み合わせを選択します。`-fast` は最大限の実行時パフォーマンスを得る目的で実行可能ファイルを調整するための出発点として、効果的に使用できます。
`-g`	`dbx` によるデバッグおよびパフォーマンスアナライザによる分析のために、バイナリ内に追加情報を生成します。`-g` オプションを指定してコンパイルすると、注釈付きソース、関数情報、およびプログラムのコンパイル時にコンパイラが実行した最適化と変換について説明するコンパイラの注釈メッセージの表示など、パフォーマンスアナライザの機能を最大限に利用できます。

Oracle Solaris Studio コンパイラは、コードの理解に役立つ情報をほかのコンパイラよりもはるかに多く提供します。最適化を行うと、コンパイラは、コードに対して実行した変換、並列化の障害、ループの繰り返しの実行回数などについて説明する注釈を挿入します。コンパイラの注釈は、パフォーマンスアナライザなどのツールで表示できます。

C コンパイラ

Oracle Solaris Studio の C コンパイラは、『*Programming Language - C (ISO/IEC 9899:1999)*』規格、『*Programming Languages-C (ISO/IEC 9899:1990)*』規格、および『*Programming Language - C and ISO/IEC 9899:2011*』規格の一部に準拠しています。C コンパイラは、OpenMP 4.0 共有メモリー並列化 API もサポートしています。

C コンパイルシステムはコンパイラ、アセンブラ、およびリンカーから構成されます。`cc` コマンドは、コマンド行オプションを使用して手順を別々に実行しないかぎり、これらのコンポーネントをそれぞれ自動的に起動します。

cc コマンドの構文

`cc` コマンドの構文です。

`cc [`*compiler-options*`] `*source-files*` [-L`*dir*`] [-l`*library*`]...`

`cc -flags` と入力すると、可能なすべてのコンパイラオプションの短い説明を表示できます。

ソースファイル名は `.c`、`.s`、`.S`、または `.i` で終えることができます。名前がこれらの接尾辞のいずれかで終わらないファイルは、リンクエディタに渡されます。

必要に応じて、ソースファイル名の後ろに `-L`*dir* オプションを指定して、リンカーがライブラリを検索するリストにディレクトリを追加したり、`-l`*library* オプションを指定して、リンカーの検索ライブ

ラリのリストにオブジェクトライブラリを追加できます。-L オプションは、コマンド行で、関連付けられているライブラリより前にある必要があります。

リンクエディタはデフォルトで、a.out という名前の動的にリンクされた実行可能ファイルを生成します。- o *filename* オプションを使用すると、別の実行可能ファイル名を指定できます。-c オプションを使用すると、ソースファイルをコンパイルし、オブジェクト (.o) ファイルを作成しますが、リンクしないでおくことができます。

test.c という名前のソースファイルをコンパイルして、a.out という名前の実行可能ファイルを生成するには:

```
% cc test.c
```

ソースファイル test1.c および test2.c をコンパイルして、test という名前の実行可能ファイルにリンクするには:

```
% cc -o test test1.c test2.c
```

2 つのソースファイルを別々にコンパイルして、それらを実行可能ファイルにリンクするには:

```
% cc -c test1.c
% cc -c test2.c
% cc test1.o test2.o
```

C のドキュメント

C コンパイラおよび cc コマンドとそのオプションの使用方法についての詳細は、『Oracle Solaris Studio 12.4: C ユーザーガイド 』および cc(1) のマニュアルページを参照してください。新機能および変更された機能については、『Oracle Solaris Studio 12.4 リリースの新機能 』を参照してください。問題と回避策、およびコンパイラの制限事項と互換性の問題については、『Oracle Solaris Studio 12.4: リリースノート 』を参照してください。

C++ コンパイラ

Oracle Solaris Studio C++ コンパイラ (CC) は、*ISO International Standard for C++, ISO/IEC 14882:2011, Programming Language — C++* および *ISO International Standard for C++, ISO IS 14822:2003, Programming Language — C++* をサポートしています。この CC コンパイラは、OpenMP 4.0 共有メモリー並列化 API もサポートしています。OpenMP 4.0 API は Oracle Solaris Studio 12.4 に含まれています。

C++11 のサポートの詳細は、『Oracle Solaris Studio 12.4 リリースの新機能 』の「C++11 標準のサポート」を参照してください。

C++ コンパイラ (CC) は、指定されたコマンド行オプションに従って、特定のオペレーティングシステム、プロセッサ、アーキテクチャー、メモリーモデル (32 ビットおよび 64 ビット)、浮動小数点演算などを対象とするコードを生成します。コンパイラは、シリアルソースコードを自動的に並列化して、マルチコアシステムでよりよいパフォーマンスとなるバイナリを生成したり、ほかの Oracle Solaris Studio ツールによる拡張されたデバッグまたは分析のためのバイナリを準備することもできます。このコンパイラは GNU C/C++ 互換性機能もサポートしています。

C++ コンパイラは、フロントエンド、オプティマイザ、コードジェネレータ、アセンブラ、テンプレートのプリリンカー、リンクエディタから構成されています。CC コマンドは、コマンド行オプションでほかの指定をしないかぎり、これらのコンポーネントをそれぞれ自動的に起動します。

CC コマンドの構文

CC コマンドの構文です。

CC [*compiler-options*] *source-files* [-L*dir*] [-l *library*]...

CC -flags と入力すると、可能なすべての CC コンパイラオプションの短い説明を表示できます。

ソースファイル名は .c、.C、.cc、.cxx、.c++、.cpp、または .i で終えることができます。名前がこれらの接尾辞のいずれかで終わらないファイルは、オブジェクトファイルまたはライブラリとして扱われ、リンクエディタに渡されます。

必要に応じて、ソースファイル名の後ろに -L*dir* オプションを指定して、リンカーがライブラリを検索するリストにディレクトリを追加したり、-l*library* オプションを指定して、リンカーの検索ライブラリのリストにオブジェクトライブラリを追加できます。-L オプションは、コマンド行で、関連付けられているライブラリより前にある必要があります。

デフォルトでは、ファイルは指定された順序でコンパイルおよびリンクされ、a.out という名前の出力ファイルが生成されます。- o *filename* オプションを使用すると、別の実行可能ファイル名を指定できます。-c オプションを使用すると、ソースファイルをコンパイルし、オブジェクト (.o) ファイルを作成しますが、リンクしないでおくことができます。

test.C という名前のソースファイルをコンパイルして、a.out という名前の実行可能ファイルを生成するには:

```
% CC test.c
```

2 つのソースファイル `test1.c` および `test2.C` を別々にコンパイルしてから、それらを `test` という名前の実行可能ファイルにリンクするには:

```
% CC -c test1.c
% CC -c test2.C
% CC -o test test1.o test2.o
```

C++ のドキュメント

C++ コンパイラおよび `CC` コマンドとそのオプションの使用方法についての詳細は、『Oracle Solaris Studio 12.4: C++ ユーザーズガイド 』および `CC`(1) のマニュアルページを参照してください。新機能および変更された機能については、『Oracle Solaris Studio 12.4 リリースの新機能 』を参照してください。問題と回避策、およびコンパイラの制限事項と互換性の問題については、『Oracle Solaris Studio 12.4: リリースノート 』を参照してください。

Fortran 95 コンパイラ

Oracle Solaris Studio の Fortran コンパイラは、マルチプロセッサシステム上の Oracle Solaris 向けに最適化されています。このコンパイラは、自動的かつ明示的なループ並列化を実行できるため、プログラムをマルチプロセッサシステム上で効率的に実行できます。

この Fortran コンパイラは、Fortran77、Fortran90、および Fortran95 規格との互換性を備え、OpenMP 4.0 をサポートしています。

`f95` コマンドは、Oracle Solaris Studio Fortran コンパイラを起動します。

`f95` コマンドの構文

`f95` コマンドの構文です。

`f95` [*compiler-options*] *source-files*... [`-l`*library*]

コンパイラオプションはソースファイル名より前に配置します。`f95 -flags` と入力すると、可能なすべてのコンパイラオプションの短い説明を表示できます。

ソースファイル名は、`.f`、`.F`、`.f90`、`.f95`、`.F90`、`.F95`、または `.for` で終わる 1 つまたは複数の Fortran ソースファイル名にする必要があります。

必要に応じて、ソースファイル名の後ろに `-l`*library* オプションを指定して、オブジェクトライブラリをリンカーの検索ライブラリのリストに追加できます。

2 つのソースファイルから Fortran プログラムをコンパイルするためのサンプルコード:

```
% f95 -o hello_1 foo.f bar.f
```

同じプログラムを別個のコンパイルおよびリンク手順でコンパイルするには:

```
% f95 -c -o bar.o bar.f
% f95 -c  -o foo.o foo.f
% f95 -o hello_1 bar.o foo.o
```

同じプログラムをコンパイルして、`libexample` というライブラリにリンクするには:

```
% f95 -o hello_1 foo.f bar.f -lexample
```

Fortran のドキュメント

Fortran 95 コンパイラおよび `f95` コマンドとそのオプションの使用方法についての詳細は、『Oracle Solaris Studio 12.4: Fortran ユーザーズガイド 』および `f95`(1) のマニュアルページを参照してください。新機能および変更された機能については、『Oracle Solaris Studio 12.4 リリースの新機能 』を参照してください。問題と回避策、およびコンパイラの制限事項と互換性の問題については、『Oracle Solaris Studio 12.4: リリースノート 』を参照してください。

C/C++/Fortran ライブラリ

Oracle Solaris Studio コンパイラは、オペレーティングシステムのネイティブライブラリを利用します。Oracle Solaris オペレーティングシステムは、C 実行時ライブラリ `libc` や C++ 実行時ライブラリ `libCrun` を含む多数のシステムライブラリを備えており、これらは `/usr/lib` にインストールされています。`intro`(3) のマニュアルページでは、各ライブラリについて説明し、各ライブラリの詳細情報を示す追加のマニュアルページの参照先を示しています。このページを表示するには `man intro.3` と入力します。

`/usr/lib` システムライブラリをリンクするには、コンパイラで適切な `-l` オプションを使用します。たとえば、`libmalloc` ライブラリをリンクするには、リンク時に `cc` および `CC` コマンド行で `-lmalloc` を指定します。

Fortran、C、および C++ の実行時ライブラリは、オペレーティングシステムで提供されているライブラリに加えて、Oracle Solaris Studio でも提供されています。そのようなライブラリの例としては、`libsunmath` および `libmopt` 数学ライブラリがあります。

Fortran 実行時ライブラリは、オペレーティングシステムではなく Oracle Solaris Studio で提供されます。

Fortran プログラムでは、C インタフェースを持つ Oracle Solaris の `/usr/lib` ライブラリも使用できます。C-Fortran インタフェースの詳細は、『*Fortran プログラミングガイド*』を参照してください。

ライブラリのリンクについての詳細は、Oracle Solaris ドキュメントの『*リンカーとライブラリ*』を参照してください。

並列プログラミング用の OpenMP 4.0

OpenMP は、C、C++、および Fortran で共有メモリー並列アプリケーションを記述するためのアプリケーションプログラミングインタフェース (API) です。コンパイラディレクティブ、ライブラリルーチン、および環境変数で構成されています。

OpenMP でのプログラミングには、次の利点があります。

- 最新のマルチコアアーキテクチャーでプログラムのパフォーマンスが劇的に向上することがあります。
- OpenMP は多数のコンパイラでサポートされているため、プログラマは移植性があるコードを簡単に記述できます。
- プログラミングの労力が少なくなります。プログラマは、既存のプログラムの並列化できるコードを特定して、それを並列化するプラグマを追加します。
- プログラマはコードを漸増的に並列化できます。

コンパイラの OpenMP のサポートを利用するには、OpenMP 指令および関数を使用してコードのセクションを並列化し、コンパイル時に `-xopenmp` オプションを使用します。詳細は、『Oracle Solaris Studio 12.4: OpenMP API ユーザーズガイド 』を参照してください。

大量の計算を行うプログラムのための Oracle Solaris Studio Performance Library

Oracle Solaris Studio Performance Library は、線形代数や大量の数値計算を伴う問題を解くための一連の最適化された高速数学サブルーチンです。Oracle Solaris Studio Performance Library は、Netlib (http://www.netlib.org) から入手できるパブリックドメ

インのサブルーチンのコレクションに基づいています。Oracle はこれらのパブリックドメインのサブルーチンを強化し、Oracle Solaris Studio Performance Library としてバンドルしました。

Oracle Solaris Studio Performance Library のルーチンは、マルチコアおよびマルチプロセッサ (MP) の Oracle システムでのアプリケーションのパフォーマンスを改善できます。多くのルーチンには、ベースとなっている Netlib ライブラリには存在しない SPARC および x86 に固有の最適化、および OpenMP を使用した並列化が行われています。標準の Fortran インタフェースのほかに、完全な一連の C インタフェースも含まれています。

Oracle Solaris Studio Performance Library は、ほかのライブラリをリンクするために使用される -l スイッチではなく、-library スイッチでアプリケーションにリンクされます。

Performance Library ルーチンを使用する Fortran ソースをコンパイルするには:

```
% f95 -dalign filename.f -library=sunperf
```

Performance Library をコンパイルしてデータの整列を制御するために -dalign オプションが使用されているため、このオプションは必須です。

Performance Library ルーチンを使用する C または C++ ソースをコンパイルするには:

```
% cc filename.c -library=sunperf
% CC filename.cpp -library=sunperf
```

Oracle Solaris Studio Performance Library のないシステムにアプリケーションを配備できるように、静的にコンパイルおよびリンクするには、オプション -library=sunperf および -staticlib=sunperf を使用する必要があります。

Oracle Solaris Studio Performance Library の使用方法についての詳細は、情報ライブラリの PDF 形式の『Oracle Solaris Studio 12.4: Performance Library のユーザーズガイド』を参照してください。ライブラリの各関数およびサブルーチンのマニュアルページについては、Oracle Solaris Studio のマニュアルページのセクション 3p を参照してください。Oracle Solaris Studio Performance Library の新機能および変更された機能については、『Oracle Solaris Studio 12.4 リリースの新機能 』を参照してください。

アプリケーションの構築用の dmake ユーティリティー

dmake ユーティリティーは、メイクファイルで定義されているソフトウェアプロジェクトターゲットを構築するための、make(1) との互換性を持つコマンド行ツールです。dmake は、グリッド、分散、並

列、または逐次モードでターゲットを構築できます。標準的な make(1) ユーティリティーを使用している場合は、dmake への切り替えに伴ってメイクファイルに変更を加える必要があるとしても、変更はわずかです。dmake は、make ユーティリティーのスーパーセットです。

dmake はメイクファイルを解析し、並行して構築可能なターゲットを特定し、.dmakerc ファイルで指定した多数のホストにそれらのターゲットの構築を分散します。

プロジェクトのメイクファイルにあるターゲットを使用して Oracle Solaris Studio IDE でプロジェクトを構築して実行すると、IDE はデフォルトで dmake を使用します。IDE を介して、メイクファイルの個別のターゲットを dmake で実行することもできます。必要に応じて、代わりに make を使用するように IDE を構成できます。

コマンド行から dmake を使用する方法、および .dmakerc ファイルの作成方法については、『Oracle Solaris Studio 12.4: 分散メイク (dmake) 』または dmake(1) のマニュアルページを参照してください。

アプリケーションのデバッグ用のツール

Oracle Solaris Studio には、アプリケーション内のエラーの検出に役立つ dbx デバッガが付属しています。

dbx は、対話型でソースレベルの、コマンド行ベースのデバッグツールです。これを使用して、C、C++、または Fortran プログラムを制御下に置いた状態で実行し、停止したプログラムの状態を調べることができます。dbx により、プログラムの動的な実行をすべて制御でき、パフォーマンスデータとメモリーの使用状況の収集、メモリーアクセスのモニタリング、およびメモリーリークの検出も行えます。

dbx では次のタスクを実行できます。

- クラッシュしたプログラムのコアファイルを調べる
- ブレークポイントを設定する
- プログラムをステップ実行する
- 呼び出しスタックを調べる
- 変数と式を評価する
- 実行時検査を使用して、メモリーアクセスの問題とメモリーリークを検出する
- 修正と継続を使用して、ソースファイルを変更して再コンパイルし、プログラム全体を再構築せずに実行を継続する

dbx デバッガは、コマンド行で、Oracle Solaris Studio IDE 経由でグラフィカルに、または dbxtool という別個のグラフィカルインタフェース経由で使用できます。

それぞれのユーザーインタフェースで dbx を使用する方法については、次のセクションを参照してください。

- 21 ページの「コマンド行での dbx」
- 22 ページの「IDE での dbx」
- 23 ページの「dbxtool での dbx」

コマンド行での dbx

dbx を起動するための dbx コマンドの基本構文です。

dbx [*options*] [*program-name*|-] [*process-ID*]

dbx セッションを開始して、デバッグするプログラム test をロードするには:

```
% dbx test
```

dbx セッションを開始して、プロセス ID 832 ですでに実行されているプログラムに接続するには:

```
% dbx - 832
```

dbx セッションが開始されると、dbx がデバッグ対象プログラムのプログラム情報をロードします。dbx は次に、準備状態となって待機し、C または C++ プログラムの main() 関数など、プログラムのメインブロックを表示します。(dbx) コマンドプロンプトが表示されます。

(dbx) プロンプトにコマンドを入力できます。一般に、まず stop in main などのコマンドを入力してブレークポイントを設定してから、run コマンドを入力してプログラムを実行します。

```
(dbx) stop in main
(4) stop in main
(dbx) run
Running: quote_1
(process id 5685)
(dbx)
```

ブレークポイントで実行が停止したとき、step や next などのコマンドを入力してコードをシングルステップ実行したり、print および display を入力して式と変数を評価したりできます。

dbx ユーティリティーのコマンド行オプションの詳細は、dbx(1) のマニュアルページを参照してください。

コマンドリファレンスセクションを含む dbx の使用方法についての詳細は、『Oracle Solaris Studio 12.4: dbx コマンドによるデバッグ 』を参照してください。(dbx) コマンド行に help と入力しても、dbx コマンドやその他のトピックについて知ることができます。

新機能および変更された機能のリストについては、『Oracle Solaris Studio 12.4 リリースの新機能 』を参照してください。

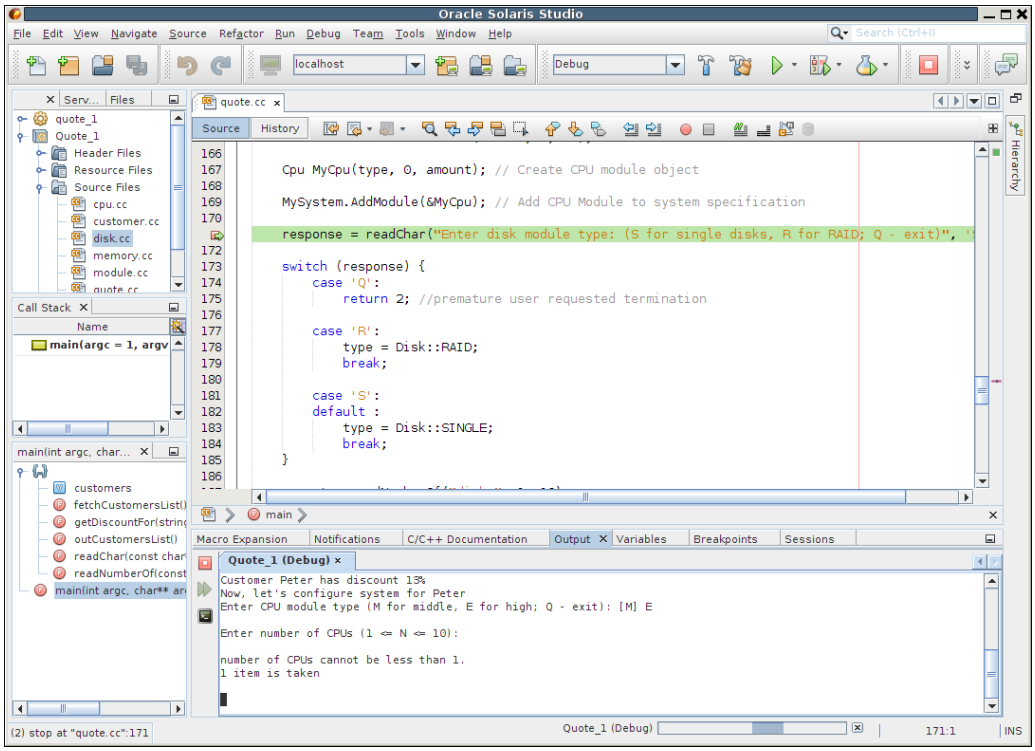
現在のリリースの dbx の既知の問題、制限事項、および互換性の問題については、『Oracle Solaris Studio 12.4: リリースノート 』を参照してください。

IDE での dbx

プロジェクトを開き、ソースにブレークポイントを作成して、「デバッグ」ボタンをクリックすることによって、Oracle Solaris Studio IDE で dbx を使用できます。IDE では、プログラムをステップ実行するためのメニューオプションおよびボタンを使用でき、デバッグウィンドウの完全なセットが用意されています。

IDE は、アプリケーションを構築する場合と同様に、アプリケーションをプロジェクトとしてデバッグします。IDE を使用して、IDE プロジェクトに関連付けられていない実行可能ファイルをデバッグすることもできます。

次のスクリーンショットでは、IDE のサンプルプロジェクトの 1 つが dbx で実行されています。「デバッグ」メニューのコマンドまたは IDE ウィンドウの右上にあるボタンを使用して、デバッガを制御できます。「デバッグ」のコマンドおよびボタンを使用すると、IDE が dbx にコマンドを発行し、さまざまなデバッグウィンドウに出力を表示します。



この図では、デバッガがブレークポイントで停止していて、出力ウィンドウにプログラムの相互の関連性が表示されています。「変数」や「ブレークポイント」などの一部のデバッガウィンドウも表示されていますが、選択されていません。「ウィンドウ」->「デバッグ」メニューを選択して、さらにデバッグウィンドウを開くこともできます。デバッグウィンドウの 1 つに、dbx との相互の関連性を示す「デバッガコンソール」ウィンドウがあります。「デバッガコンソール」ウィンドウで (dbx) プロンプトにコマンドを入力することもできます。

IDE での dbx の使用方法についての詳細は、IDE の統合ヘルプおよび『Oracle Solaris Studio 12.4: IDE クイックスタートチュートリアル 』を参照してください。

dbxtool での dbx

IDE とは別ですが、同様のデバッグウィンドウおよびエディタを備えたグラフィカルツールである dbxtoolを経由して dbx を使用することもできます。IDE と異なり、dbxtool はプロジェクトを使用しないため、C、C++、または Fortran の実行可能ファイルまたはコアファイルのデバッグに使用できます。

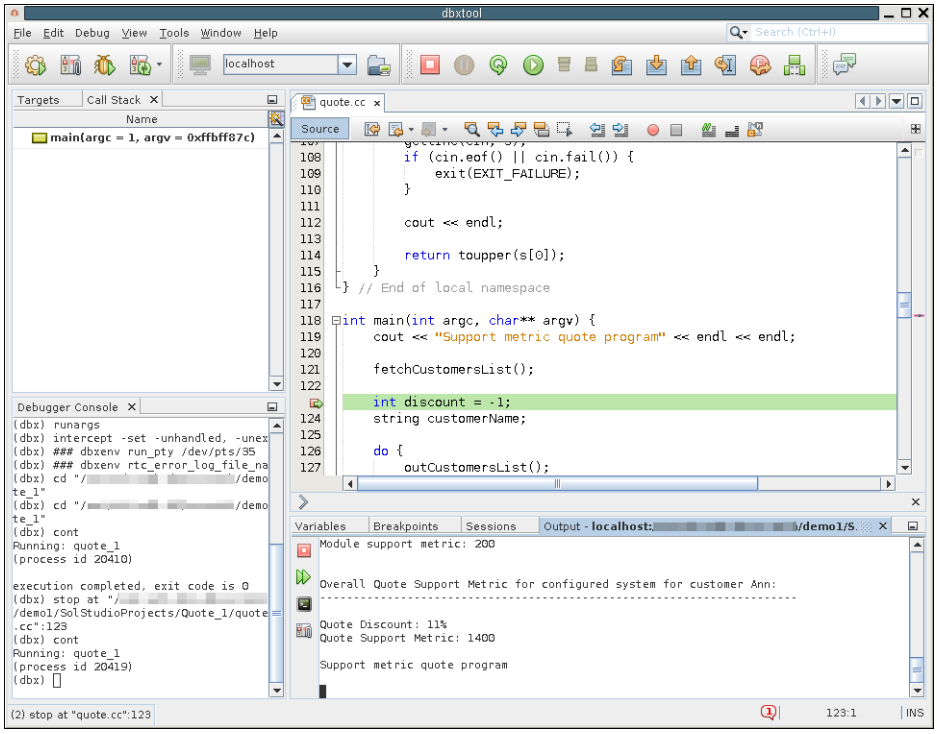
dbxtool を起動するには、次を入力します。

```
% dbxtool executable-name
```

実行可能ファイル名を省略して、代わりに dbxtool 内から指定することもできます。

IDE の場合と同様、dbxtool のツールバーボタンをクリックするか、「デバッグ」メニューのオプションを使用して、dbx にコマンドを発行できます。「デバッガコンソール」ウィンドウで (dbx) プロンプトにコマンドを入力することもできます。

次の図では、dbxtool で quote_1 プログラムに対して dbx が実行されています。「デバッガコンソール」ウィンドウが選択され、(dbx) プロンプトと、ユーザーの選択に応じて dbxtool によって入力されたコマンドが表示されています。



dbxtool の使用方法については、dbxtool (1) のマニュアルページまたは dbxtool の統合ヘルプを参照してください。『Oracle Solaris Studio 12.4: dbxtool チュートリアル 』では、dbxtool の使用方法を説明しています。

アプリケーションの検証用のツール

Oracle Solaris Studio は、アプリケーションの安定性を検証するためのツールを備えています。次のツールは、動的分析、静的分析、およびコードカバレージ分析の組み合わせにより、メモリーリークやメモリーアクセス違反などのアプリケーションの脆弱性を検出します。

`discover`	コード内のメモリーアクセスエラーの検出に役立つコマンド行ユーティリティーです。
`uncover`	テストによってカバーされないアプリケーションコードの領域を示すコマンド行ユーティリティーです。
コードアナライザ	C および C++ コンパイラによって収集された静的コードエラーデータと、`discover` および `uncover` によって収集されたデータを分析するグラフィカルツールです。コードアナライザでは、静的エラーデータを動的メモリーアクセスエラーデータおよびコードカバレージデータと統合することで、ほかのエラー検出ツールを単独で使用した場合には見つからないアプリケーション内のエラーを検出できます。
`codean`	コードアナライザと同様の機能を提供するコマンド行ユーティリティー。

メモリーエラーを検出するための `discover` ツール

メモリーエラー探索ツール (`discover`) は、プログラム内のメモリーアクセスエラーを検出するための高度な開発ツールです。`-g` を指定してバイナリをコンパイルすると、`discover` はエラーおよび警告をレポートする際にソースコードおよび行番号情報を表示できるようになります。

`discover` ユーティリティーは簡単に使用できます。`-g` を指定してバイナリをコンパイルしてから、そのバイナリに対して `discover` コマンドを実行して計測機構を組み込みます。その後、計測機構の組み込まれたバイナリを実行して、`discover` レポートを生成します。`discover` レポートは、HTML 形式、テキスト形式、またはその両方で要求できます。レポートにはメモリーエラー、警告、およびメモリーリークが表示され、各エラーまたは警告についてソースコードとスタックトレースを表示することもできます。

`discover`(1) のマニュアルページにある次の例は、メモリーアクセスエラーを検出するための `discover` レポートを生成するために、実行可能ファイルを準備し、計測機構を組み込み、実行する方法を示しています。`discover` のコマンド行の `-w` オプションはレポートをテキストとして書き出すことを示し、`-o` オプションは出力を画面に表示することを示します。

```
% cc -g -O2 test.c -o test.prep
% discover -w - -o test.disc test.prep
```

```
% ./test.disc
ERROR (UMR): accessing uninitialized data from address 0x5000c (4 bytes) at:
     foo() + 0xdc  <ui.c:6>
          3:    int *t;
          4:    foo() {
          5:     t = malloc(5*sizeof(int));
          6:=>   printf("%d0, t[1]);
          7:    }
          8:
          9:    main()
     main() + 0x1c
     _start() + 0x108
    block at 0x50008 (20 bytes long) was allocated at:
     malloc() + 0x260
     foo() + 0x24  <ui.c:5>
          2:
          3:    int *t;
          4:    foo() {
          5:=>   t = malloc(5*sizeof(int));
          6:     printf("%d0, t[1]);
          7:    }
          8:
     main() + 0x1c
     _start() + 0x108

     ***************** Discover Memory Report *****************

1 block at 1 location left allocated on heap with a total size of 20 bytes

    1 block with total size of 20 bytes
     malloc() + 0x260
     foo() + 0x24  <ui.c:5>
          2:
          3:    int *t;
          4:    foo() {
          5:=>   t = malloc(5*sizeof(int));
          6:     printf("%d0, t[1]);
          7:    }
          8:
     main() + 0x1c
     _start() + 0x108
```

詳細は、discover(1) のマニュアルページと『Oracle Solaris Studio 12.4: Discover および Uncover ユーザーズガイド 』を参照してください。

コードカバレージを測定するための uncover ツール

uncover はコードカバレージを測定するためのコマンド行ツールです。このツールは、アプリケーションの実行時に実行されるアプリケーションコードの領域と、実行されず、テストによってカバーされない領域を示します。Uncover は、テスト時により多くのコードがカバーされるように、

テストスイートにどの関数を追加する必要があるかの判断に役立つ統計およびメトリックを含むレポートを生成します。

uncover は、Oracle Solaris Studio コンパイラで構築された任意のバイナリで機能しますが、バイナリが最適化なしで構築されている場合に最適に機能します。-g を指定してバイナリをコンパイルすると、uncover はコードカバレージについてレポートする際にソースコードおよび行番号情報を表示できるようになります。

バイナリをコンパイルしたら、そのバイナリに対して uncover を実行します。uncover は、計測コードが追加された新しいバイナリを作成し、プログラムのコードカバレージデータが格納される *binary*.uc という名前のディレクトリも作成します。計測機構の組み込まれたバイナリを実行するたびに、コードカバレージデータが収集され、*binary*.uc ディレクトリに格納されます。

パフォーマンスアナライザで実験データを表示することも、uncover レポートを HTML として生成し、Web ブラウザで表示することもできます。

次の例は、コードカバレージを調べるための uncover レポートを生成する目的で、実行可能ファイルを準備し、計測機構を組み込み、実行する方法を示しています。最適化されたバイナリは test であり、計測機構が組み込まれた、やはり test という名前のバイナリに置換されます。

```
% cc -g -O2 test.c -o test
% uncover test
% test
```

実験ディレクトリは test.uc であり、計測機構の組み込まれた test の実行時に生成されるデータが含まれます。test.uc ディレクトリには、計測機構のない test バイナリのコピーも含まれます。

パフォーマンスアナライザで実験を表示するには:

```
% uncover test.uc
```

ブラウザの HTML ページで実験を表示するには:

```
% uncover -H test.html test.uc
```

詳細は、uncover(1) のマニュアルページと『Oracle Solaris Studio 12.4: Discover および Uncover ユーザーズガイド 』を参照してください。

統合エラーチェック用のコードアナライザツール

Oracle Solaris Studio コードアナライザは、コードの統合分析を実行できるグラフィカルツールです。コードアナライザでは、ほかのツールで収集した 3 種類の情報を使用します。

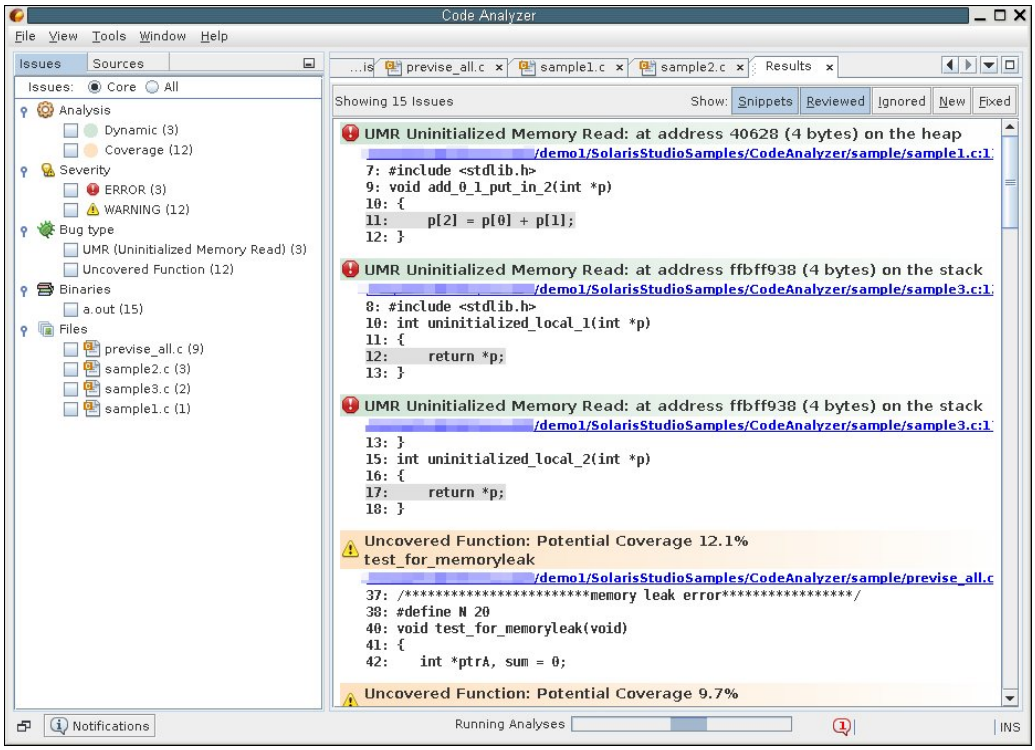
- Oracle Solaris Studio の C または C++ コンパイラでアプリケーションをコンパイルし、`-xanalyze=code` オプションを指定した場合に実行される静的コード検査。
- `discover` で `-a` オプションを使用してバイナリに計測機構を組み込んだあと、その計測機構が組み込まれたバイナリを実行した場合に実行される動的メモリーアクセス検査。
- `uncover` でバイナリに計測機構を組み込んで、その計測機構が組み込まれたバイナリを実行したあと、収集されたカバレージデータに対して `-a` オプションを指定して Uncover を実行した場合に実行されるコードカバレージ検査。

これらのツールのいずれかまたは任意の組み合わせを使用して準備したバイナリに対して、コードアナライザを使用できます。ただし、3 種類のデータを統合して表示すると、コードを最も明確に調べることができ、よりセキュリティーと堅牢性の高いアプリケーションを作成できます。

次の例は、事前に `discover` および `uncover` で準備した `a.out` という名前のバイナリに対してコードアナライザを実行する方法を示しています。

```
% code-analyzer a.out
```

次の図では、`a.out` バイナリで検出された問題がコードアナライザに表示されています。



詳細は、コードアナライザの統合ヘルプ、『Oracle Solaris Studio 12.4: コードアナライザユーザーズガイド 』、および『Oracle Solaris Studio 12.4: コードアナライザチュートリアル 』を参照してください。

統合チェック用の codean ツール

codean コマンド行ユーティリティーを使用して、コンパイラ、discover、および uncover によって収集されたデータからレポートを生成することもできます。codean ツールはコードアナライザと同様の機能を提供しますが、グラフィカル環境を利用できないシステムの場合、またはコマンド行を使用する場合に使用できます。codean ツールは自動化スクリプトでも使用でき、コードアナライザツールでまだ利用できないいくつかの機能があります。

詳細は、codean(1) のマニュアルページ、『Oracle Solaris Studio 12.4: コードアナライザユーザーズガイド 』、および『Oracle Solaris Studio 12.4: コードアナライザチュートリアル 』を参照してください。

アプリケーションのパフォーマンスを調整するためのツール

Oracle Solaris Studio ソフトウェアはアプリケーションの動作を調べるために使用できるいくつかのツールを備えているため、アプリケーションのパフォーマンスを調整できます。

パフォーマンスツールには次のものが含まれています。

- **パフォーマンスアナライザおよび関連ツール。**問題がパフォーマンスに影響を及ぼすコード内の位置を識別するために役立つ、高度なパフォーマンスツールおよびユーティリティーのセットです。
- **簡易パフォーマンス最適化ツール (SPOT)。**パフォーマンスアナライザツールと連携し、それらのツールによって収集されたデータをレポートする Web ページを生成します。
- **IDE のプロファイリングツール。**IDE 内からプロジェクトのパフォーマンスを検査できます。

パフォーマンスアナライザツール

Oracle Solaris Studio ソフトウェアには、相互に連携する一連の高度なパフォーマンスツールおよびユーティリティーが備わっています。コレクタ、パフォーマンスアナライザ、スレッドアナライザ、および `er_print` ユーティリティーは、コードのパフォーマンスを評価し、潜在的なパフォーマンスの問題を識別し、問題が発生するコードの部分を特定するために役立ちます。これらのツールをまとめてパフォーマンスアナライザツールといいます。

Oracle Solaris Studio の C、C++、および Fortran コンパイラのオプションを使用して、ハードウェアと、プログラムのパフォーマンスを向上させる高度な最適化テクニックをターゲットにすることができます。パフォーマンスアナライザツールは、Oracle Sun ハードウェア上でコンパイラとともに使用できるようにも設計されていて、Oracle Sun マシン上で実行する場合のプログラムのパフォーマンスを向上させるのに役立ちます。

パフォーマンスアナライザツールでは、収集されるデータをより精密に制御し、データをより深く検査し、プログラムとハードウェアの相互の関連性を調べることができます。パフォーマンスアナライザツールは、現在の Oracle Sun ハードウェアで実行されている、複雑なコンピュートインテンシブアプリケーション向けに設計されており、それらのアプリケーションでテスト済みです。

パフォーマンスアナライザツールは、OpenMP 並列アプリケーションおよび MPI ベースの分散アプリケーションのプロファイリング機能も備えているため、それらのテクノロジをアプリケーションで効果的に使用しているかどうかを判別できます。

パフォーマンスアナライザツールを使用するには、2 つの手順を実行する必要があります。

1. パフォーマンスアナライザでターゲットアプリケーションをプロファイルするか、collect コマンドを使用してターゲットアプリケーションからパフォーマンスデータを収集します。
2. パフォーマンスアナライザのグラフィカルツール、er_print コマンド行ユーティリティー、またはスレッドアナライザのグラフィカルツールでデータを調べて、マルチスレッドアプリケーションでのデータの競合およびデッドロックを検出します。

パフォーマンスデータを収集してアプリケーションをプロファイルする

コレクタは、プロファイリングを使用し、関数呼び出しをトレースすることによってパフォーマンスデータを収集します。このデータには、呼び出しスタック、マイクロステートアカウンティング情報 (Oracle Solaris プラットフォームのみ)、スレッド同期遅延データ、ハードウェアカウンタのオーバーフローデータ、MPI (Message Passing Interface) 関数呼び出しデータ、メモリー割り当てデータ、およびオペレーティングシステムとプロセスのサマリー情報が含まれる可能性があります。コレクタは C、C++、および Fortran プログラムのあらゆる種類のデータと、Java プログラミング言語で記述されたアプリケーションのプロファイリングデータを収集できます。コレクタは、collect コマンドを使用して、またはパフォーマンスアナライザの「アプリケーションのプロファイル」ダイアログから、あるいは dbx デバッガの collect サブコマンドを使用して実行できます。

Oracle Solaris Studio IDE のプロファイリングツールも、コレクタを使用して情報を収集します。

collect コマンドでデータを収集するには:

```
% collect [collect-options] executable executable-options
```

collect コマンドにオプションを付けて、収集するデータの種類を指定できます。たとえば、-i on オプションを指定すると、コレクタが入出力トレースを実行します。実行可能ファイルの後ろに引数を指定することによって、ターゲットの実行可能ファイルに引数を渡すことができます。

コレクタは、デフォルトで test.1.er という名前のデータディレクトリを作成しますが、コマンド行で別の名前を指定できます。test.1.er ディレクトリは実験と呼ばれ、その名前はツールによって実験として認識されるように、常に .er で終わる必要があります。

次のコマンドは、synprog プログラムに対して collect を使用する方法を示しています。

```
% collect synprog

Creating experiment database test.1.er (Process ID: 11103) ...
00:00:00.000  ===== (11103) synprog run
```

```
00:00:00.005  ===== (11103) Mon  22 Sep 14  17:05:51 Stopwatch calibration
  OS release 5.11 -- enabling microstate accounting 5.11.
        0.000096 s.  (22.4 % of 0.000426 s.) -- inner
 N = 1000, avg = 0.096 us., min = 0.090, max = 0.105
        0.000312 s.  (67.0 % of 0.000466 s.) -- outer
 N = 1000, avg = 0.312 us., min = 0.307, max = 0.457
00:00:00.006  ===== (11103)  Begin commandline
 icpu.md.cpu.rec.recd.dousl.gpf.fitos.uf.ec.tco.b.nap.sig.sys.so.sx.so
00:00:00.006  ===== (11103) start of icputime
    3.003069 wall-secs.,   2.978360 CPU-secs., in icputime
00:00:03.009  ===== (11103) start of muldiv
    3.007489 wall-secs.,   2.997647 CPU-secs., in muldiv
00:00:06.017  ===== (11103) start of cputime
    3.002315 wall-secs.,   2.989407 CPU-secs., in cputime
00:00:09.019  ===== (11103) start of recurse
    3.082371 wall-secs.,   3.069782 CPU-secs., in recurse
    ...
    (output edited to conserve space)
    ...
```

データは `test.1.er` ディレクトリに格納され、パフォーマンスアナライザまたは `er_print` を使用して表示できます。

ダウンロード可能なサンプルアプリケーションにパフォーマンスアナライザを使用するステップごとの手順については、『Oracle Solaris Studio 12.4: パフォーマンスアナライザチュートリアル 』を参照してください。

プロファイリングアプリケーション、およびコレクタの使用方法の詳細については、パフォーマンスアナライザの「ヘルプ」メニュー、『Oracle Solaris Studio 12.4: パフォーマンスアナライザ 』マニュアル、および `collect`(1) のマニュアルページを参照してください。

パフォーマンスアナライザでパフォーマンスデータを調べる

パフォーマンスアナライザを使用すると、アプリケーションの動作について深い理解が得られ、コード内の問題のある部分を見つけることができます。パフォーマンスアナライザは、もっともシステムリソースを使用している関数、コードセグメント、およびソース行を特定します。パフォーマンスアナライザは、シングルスレッド、マルチスレッド、およびマルチプロセスのアプリケーションをプロファイルし、アプリケーションのパフォーマンスを改善できる箇所を特定するために役立つプロファイリングデータを提供できます。

パフォーマンスアナライザは `analyzer` コマンドで実行できます。パフォーマンスアナライザを起動するための `analyzer` コマンドの基本構文です。

```
% analyzer [experiment-list]
```

experiment-list は、コレクタによって収集された実験の 1 つまたは複数のファイル名です。複数の実験をロードする場合は、名前をスペースで区切って指定します。パフォーマンスアナライザは、複数の実験に対して起動された場合、デフォルトでは実験データをまとめますが、コマンド行で実験名の前に -c オプションを指定すると、実験を比較するためにも使用できます。

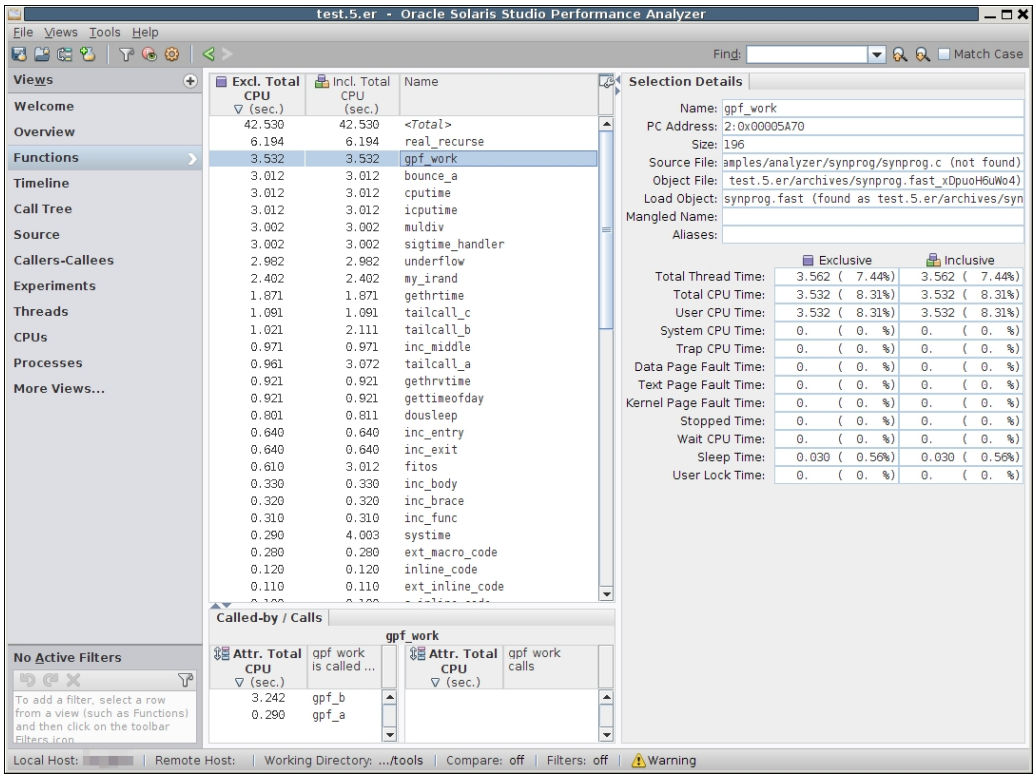
コマンド行に実験を指定しないと、パフォーマンスアナライザの「ようこそ」画面が表示され、そこから作業を開始できます。

パフォーマンスアナライザで実験 `test.1.er` を開くには:

```
% analyzer test.1.er
```

実験の初期のビューは「概要」であり、プログラムによって使用された時間およびリソースの簡単な概要が表示され、パフォーマンスデータのビューに表示するパフォーマンスメトリックを選択できます。

次の図は、`synprog` の例で行われた `test.1.er` 実験についての、パフォーマンスアナライザの「関数」ビューを示しています。「関数」ビューには、`synprog` プログラムの各関数によって使用される CPU 時間が表示されます。関数 `gpf_work` をクリックすると、`gpf_work` 関数のリソース使用状況の詳細が右側の「選択の詳細」タブに表示されます。「関数」ビューの下部の「呼び出し元/呼び出し回数」領域には、`gpf_work` によって呼び出された関数が表示され、呼び出しをダブルクリックすると「関数」ビューのそれらの関数に移動できます。



パフォーマンスアナライザの使用方法については、『Oracle Solaris Studio 12.4: パフォーマンスアナライザ 』マニュアル、パフォーマンスアナライザの統合ヘルプ、および `analyzer`(1) のマニュアルページを参照してください。

ダウンロード可能なサンプルアプリケーションにパフォーマンスアナライザを使用するステップごとの手順については、『Oracle Solaris Studio 12.4: パフォーマンスアナライザチュートリアル 』を参照してください。

`er_print` ユーティリティーでパフォーマンスデータを調べる

`er_print` ユーティリティーは、「タイムライン」表示、「MPI タイムライン」表示、および「MPI チャート」表示を除き、パフォーマンスアナライザで提供される表示のほとんどをプレーンテキストで提供します。

`er_print` ユーティリティーを使用して、関数、呼び出し元と呼び出し先、呼び出しツリー、ソースコードリスト、逆アセンブリリスト、標本収集情報、データ空間データ、スレッド分析データ、および実行統計のパフォーマンスメトリックを表示できます。

er_print コマンドの一般的な構文です。

```
% er_print -command experiment-list
```

1 つまたは複数のコマンドを指定して、表示するデータの種類を示すことができます。*experiment-list* は、コレクタによって収集された実験の 1 つまたは複数のファイル名です。er_print は、複数の実験について起動された場合、デフォルトでは実験データをまとめますが、実験を比較するためにも使用できます。

次の例は、プログラムの関数情報を表示するためのコマンドを示しています。表示されている出力は、このドキュメントの前のセクションにあるパフォーマンスアナライザのスクリーンショットで使用されたのと同じ実験のものです。

```
%  er_print -functions test.1.er
Functions sorted by metric: Exclusive Total CPU Time

Excl.     Incl.      Name
Total     Total
CPU sec.  CPU sec.
50.806    50.806     <Total>
 5.994     5.994     so_burncpu
 5.914     5.914     real_recurse
 3.502     3.502     gpf_work
 3.012     3.012     sigtime_handler
 3.002     3.002     bounce_a
 3.002     3.002     cputime
 3.002     3.002     icputime
 2.992     2.992     sx_burncpu
 2.992     2.992     underflow
 2.792     2.792     muldiv
 2.532     2.532     my_irand
 1.831     1.831     gethrtime
 1.031     1.991     tailcall_b
 0.961     0.961     inc_middle
 0.961     0.961     tailcall_c
 0.941     0.941     gethrvtime
 0.941     0.941     gettimeofday
 0.911     2.902     tailcall_a
 0.801     0.801     dousleep
 0.650     0.650     inc_entry
 0.640     0.640     inc_exit
 0.480     3.012     fitos
 0.330     0.330     inc_func
 0.320     0.320     inc_body
 0.320     0.320     inc_brace
 0.290     4.003     systime
 0.260     0.260     ext_macro_code

      lines deleted
```

er_print を起動するときに実験名を指定して、コマンドを省略すると、er_print を対話的に使用することもできます。(er_print) プロンプトにコマンドを入力できます。

er_print ユーティリティーの詳細は、『Oracle Solaris Studio 12.4: パフォーマンスアナライザ 』マニュアルおよび er_print(1) のマニュアルページを参照してください。

スレッドアナライザでマルチスレッドアプリケーションのパフォーマンスを分析する

スレッドアナライザは、マルチスレッドプログラムを検査するためのパフォーマンスアナライザの特殊バージョンです。スレッドアナライザは、POSIX スレッド API、Oracle Solaris スレッド API、OpenMP ディレクティブ、またはこれらの組み合わせを使用して作成されたコード内でデータの競合やデッドロックの原因となっているマルチスレッドプログラミングのエラーを検出できます。

スレッドアナライザは、マルチスレッドプログラム内で 2 つの一般的なスレッドの問題を検出します。

- 1 つのプロセス内の 2 つのスレッドが、排他ロックを保持せずに共有メモリーの同じ場所に同時にアクセスし、それらのアクセスの少なくとも 1 つが書き込みである場合に発生するデータの競合。
- 2 つ以上のスレッドが互いのタスクの完了まで待機しているためにブロックされている場合に発生するデッドロック。

スレッドアナライザはマルチスレッドプログラムの分析向けに簡素化されていて、パフォーマンスアナライザの「競合」、「デッドロック」、および「デュアルソース」のデータビューのみが表示されます。OpenMP プログラムについては、「OpenMP 並列領域」ビューと「OpenMP タスク」ビューも表示されます。

ソースコードまたはバイナリコードでのデータの競合も検出できます。どちらの場合も、コードに計測機構を組み込んで、必要なデータを収集できるようにする必要があります。

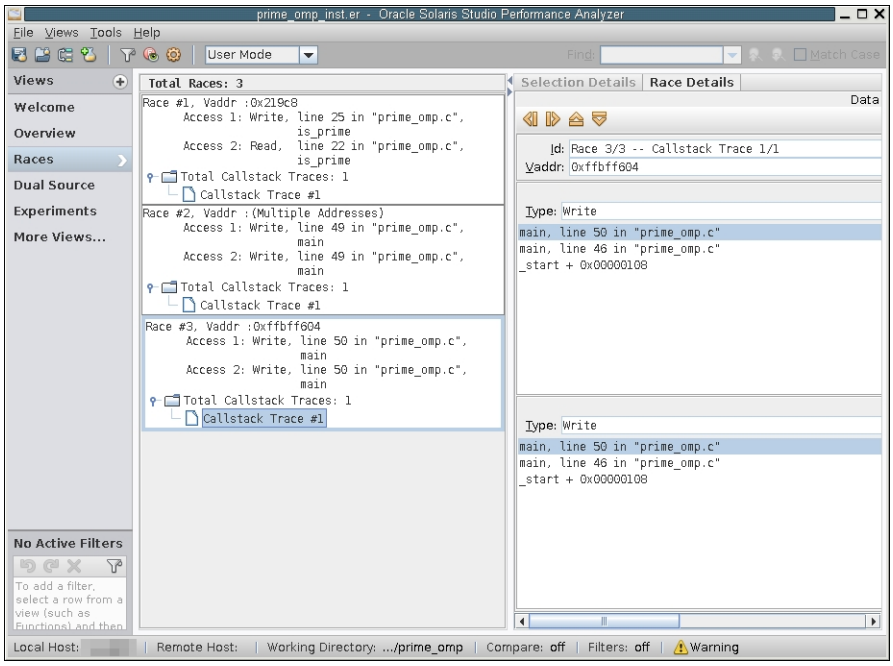
スレッドアナライザを使用するには:

1. データの競合を解析するため、コードに計測機構を組み込みます。ソースコードの場合は、コンパイル時に -xinstrument=datarace コンパイラオプションを使用します。バイナリコードの場合は、discover -i datarace コマンドを使用して、計測機構が組み込まれたバイナリを作成します。

デッドロックの検出には、計測機構は不要です。

2. データ競合データを収集するための `-r race` オプション、デッドロックデータを収集するための `-r deadlock` オプション、または両方の種類のデータを収集するための `-r all` オプションを付けた `collect` コマンドを使用して、実行可能ファイルを実行します。
3. `tha` コマンドでスレッドアナライザを起動するか、`er_print` コマンドを使用して結果の実験を表示します。

次の図は、OpenMP プログラムで検出されたデータの競合と、データの競合をもたらす呼び出しスタックが表示されている「スレッドアナライザ」ウィンドウを示しています。



スレッドアナライザの使用方法については、`tha`(1) のマニュアルページおよび『Oracle Solaris Studio 12.4: スレッドアナライザユーザーズガイド』を参照してください。

簡易パフォーマンス最適化ツール (SPOT)

簡易パフォーマンス最適化ツール (SPOT) は、アプリケーションでのパフォーマンスの問題を診断する際に役立ちます。SPOT はアプリケーションに対して一連のパフォーマンスツールを実行

し、それらのツールによって収集されたデータをレポートする Web ページを生成します。それらのツールを SPOT から独立して実行することもできます。

SPOT は Oracle Solaris Studio パフォーマンスアナライザを補完するものです。パフォーマンスアナライザは、アプリケーションの実行で時間がかかった箇所を通知します。ただし、特定の状況下では、アプリケーションの問題の診断に役立つ情報がさらに必要な場合があります。SPOT はこのような状況で助けになります。

SPOT は `collect` ユーティリティーをそのツールの 1 つとして使用します。SPOT は、`er_print` ユーティリティー、および `er_html` という追加のユーティリティーを使用して、プロファイルデータを Web ページとして表示します。

SPOT を使用する前に、SPOT ツールがコード行にパフォーマンス情報をマッピングできるように、`-O` オプションで何らかのレベルの最適化を指定し、`-g` オプションでデバッグ情報を含めてアプリケーションバイナリをコンパイルする必要があります。

SPOT を使用すると、アプリケーションを起動するか、すでに実行されているアプリケーションに接続して、パフォーマンスデータを収集できます。

SPOT を実行してアプリケーションを起動するには:

```
% spot executable
```

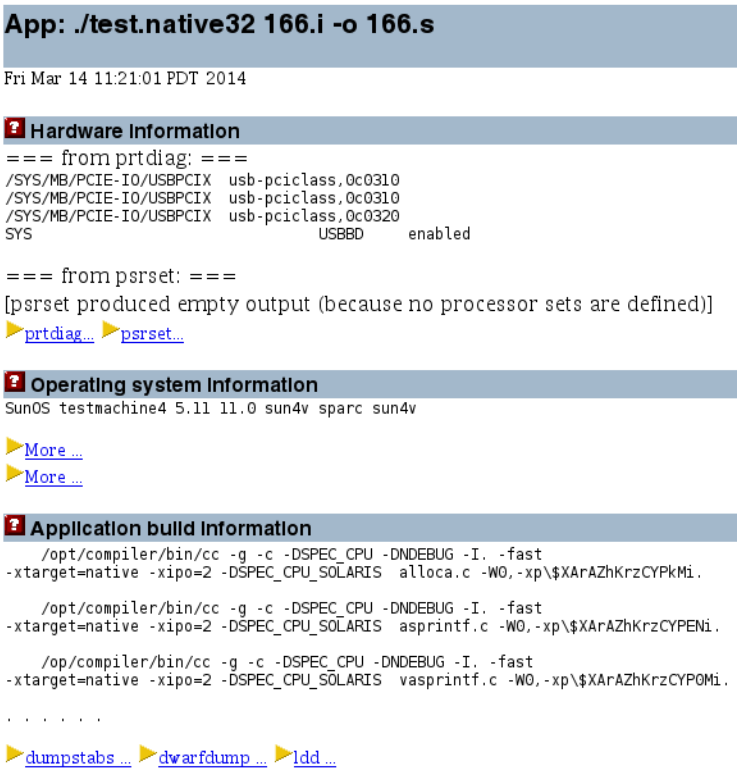
すでに実行されているアプリケーションに対して SPOT を実行するには:

```
% spot -P process-id
```

SPOT は、アプリケーションの毎回の実行についてのレポートのほか、異なる実行からの SPOT データを比較するレポートも生成します。

特定の PID に SPOT を使用すると、複数のツールがその PID に順に接続されて、レポートを生成します。

次の図は、SPOT が実行されたシステムと、アプリケーションがコンパイルされた方法についての情報を示す SPOT 実行レポートの一部を示しています。このレポートには、詳細情報を含むほかのページへのリンクがあります。



SPOT レポートの Web ページは、コンパイルされたすべてのデータを容易に調べられるように、相互にリンクされています。

詳細は、Oracle Solaris Studio 12.2 ドキュメントライブラリ () の『Oracle Solaris Studio 12.2: Simple Performance Optimization Tool (SPOT) User's Guide』を参照してください。

IDE のプロファイリングツール

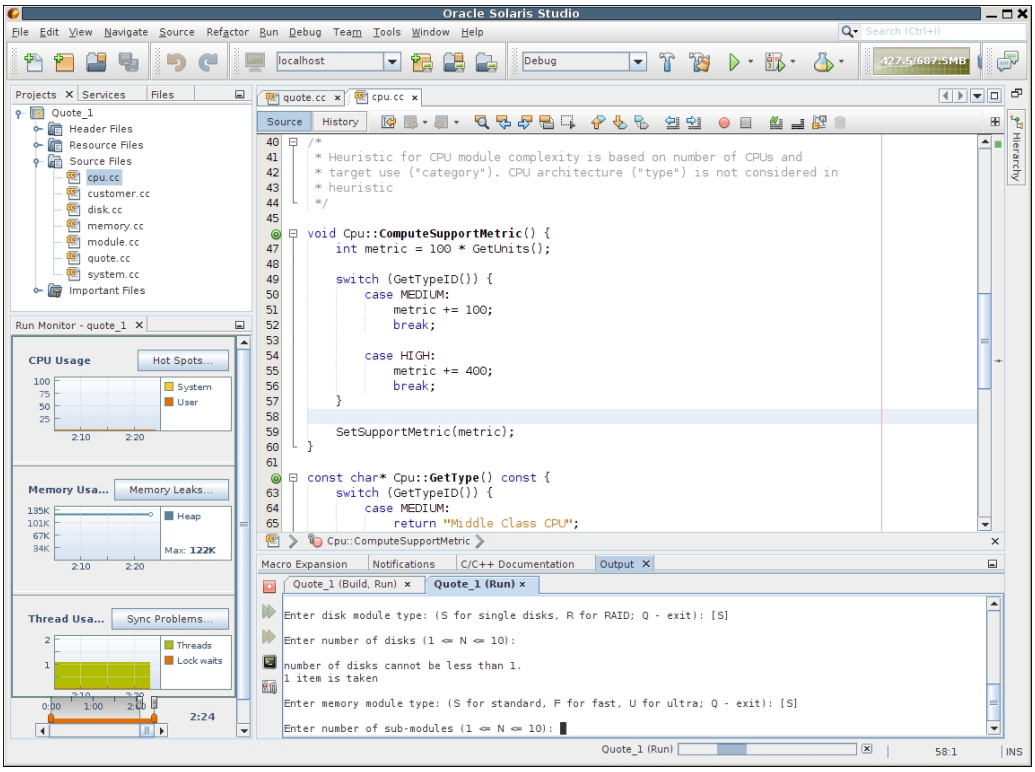
Oracle Solaris Studio IDE は、IDE 内で実行されているプロジェクトのパフォーマンスを検査できる対話型のグラフィカルプロファイリングツールを提供しています。プロファイリングツールは、Oracle Solaris Studio のユーティリティーおよびオペレーティングシステムのユーティリティーを使用してデータを収集します。

プロファイリングツールは、「プロジェクトをプロファイル」ボタン  から使用できます。

プロジェクトのモニター	プログラムのリソース使用状況のサマリーを確認できるグラフが表示されます。
メモリーアクセスエラー	実行されているプログラムを分析して、メモリーアクセスエラーおよびメモリーリークを検出します。
データ競合とデッドロックの検出	実行されているプログラムを分析して、実際に発生したデータの競合や潜在的なデータの競合、およびスレッド間のデッドロックを検出します。

プロジェクトをプロファイルして「プロジェクトをモニター」を選択すると、「実行モニター」ウィンドウが開いて、CPU 使用率、メモリー使用状況、およびスレッド使用量に関する低インパクトツールの出力が表示されます。

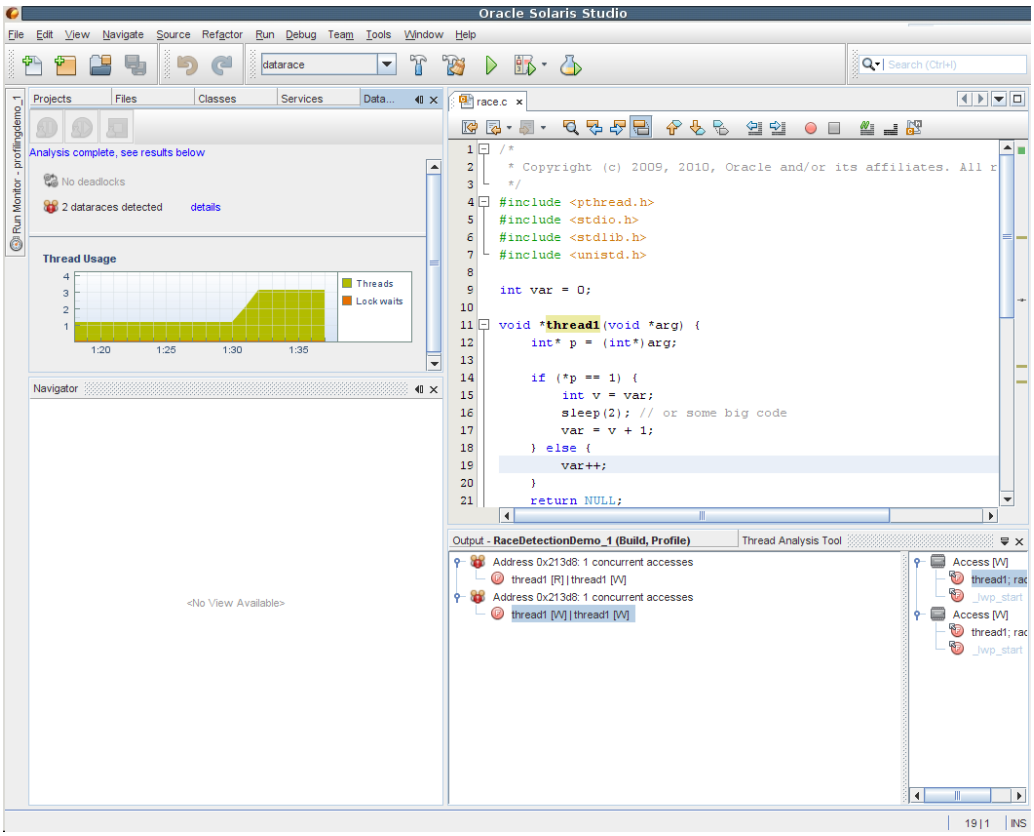
次の図は、「実行モニター」ツールが表示されている IDE を示しています。



より詳細なプロファイリングのための追加ツールは、システムおよびアプリケーションへのパフォーマンス上の影響が大きいため、プロジェクトのモニターを実行したときにそれらのツールは自動的には実行されません。高度なツールは実行モニタープロファイリングツールにリンクされていて、ホットスポット、メモリーリーク、および同期の問題を表示するためのボタンをクリックすることによって簡単に起動できます。

「データの競合とデッドロックの検出」ツールで使用されているベースとなるテクノロジは、このドキュメントで説明するスレッドアナライザと同じです。このツールは、スレッド化されたプログラムに計測機構を組み込んだあと、そのプログラムを実行中に解析して、スレッド間での実際のデータ競合およびデッドロックと潜在的なデータ競合およびデッドロックを検出します。このツールを起動するには、「プロジェクトをプロファイル」ボタンクリックし、「データの競合やデッドロック」を選択し、データ収集のオプションを指定して、「開始」をクリックします。

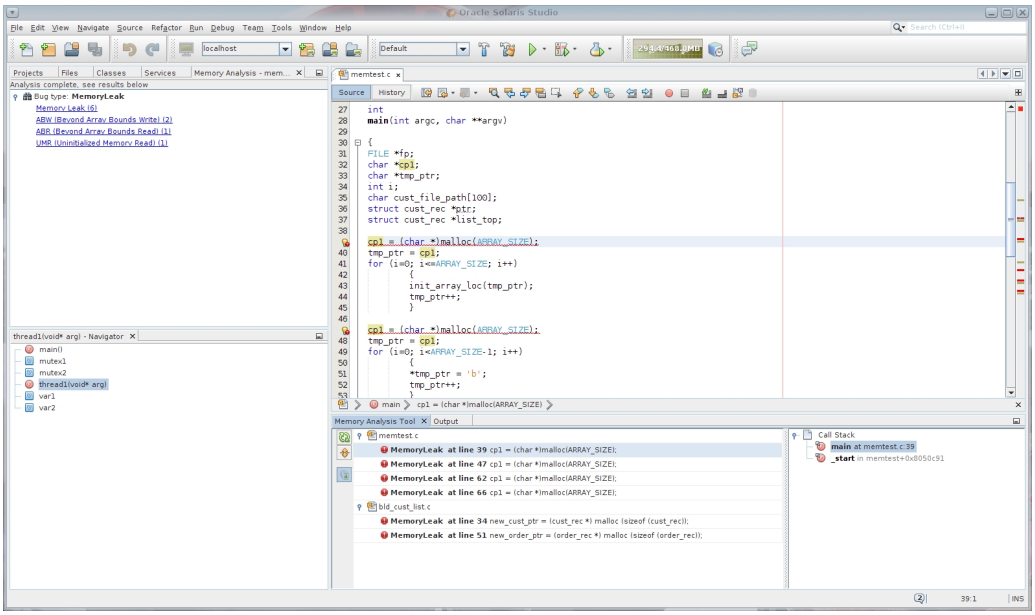
次の図は、データの競合を検出したあとの「データの競合とデッドロックの検出」ツールを示しています。



「データの競合の検出」ウィンドウの「詳細」リンクをクリックすると、「スレッドの詳細」ウィンドウが開き、データの競合の発生箇所が表示されます。「スレッドの詳細」ウィンドウでスレッドをダブルクリックして、問題が発生するソースファイルを開き、影響を受けるコード行に移動できます。

「メモリーアクセスエラー」ツールで使用されるベースとなるテクノロジは、前述の `discover` と同じです。このツールは、プログラムに計測機構を組み込んでから、プログラムを実行中に解析して、メモリーアクセスエラーおよびメモリーリークを検出します。このツールを起動するには、「プロジェクトをプロファイル」ボタンクリックし、「メモリーアクセスエラー」を選択し、データ収集のオプションを指定して、「開始」をクリックします。各種のメモリーアクセスエラーが「メモリー解析」ウィンドウに表示されます。エラーの種類をクリックすると、その種類のエラーが「メモリー解析ツール」ウィンドウに表示され、各エラーの呼び出しスタックを見ることができます。

次の図は、メモリーアクセスエラーを検出したあとの「メモリーアクセスエラー」ツールを示しています。



プロファイリングツールの使用方法については、F1 キーを押すか、IDE の「ヘルプ」メニューを使用してアクセスできる IDE の統合ヘルプを参照してください。ヘルプの「目次」タブで「C/C++/Fortran アプリケーションのプロファイル」、「データの競合とデッドロックの検出」、および「プロジェクト内のメモリーアクセスエラーの検出」を選択してください。

詳細情報

詳細は、Oracle Technology Network の Oracle Solaris Studio の製品ページを参照してください。Oracle Solaris Studio をさらに活用するために役立つホワイトペーパー、技術的な参考資料、トレーニングおよびサポートの情報、コミュニティフォーラムおよびブログが見つかります。