ORACLE
SOLARIS

Oracleテクニカル・ホワイト・ペーパー
2011年12月

Oracle Solaris Studioを使用したエンタープライズ・アプリケーションの開発

ORACLE

エグゼクティブ・サマリー

あらゆる業界の組織は、競争力のある立場を維持し、企業の目的を達成するために、強力でパフォーマンスの高いアプリケーションに依存しています。これらの重要なアプリケーションから最大の価値を引き出すには、あらゆる要素が連動して最適に機能するように設計されたプラットフォームが必要です。オラクルは、最高のパフォーマンスを実現できるよう最適化された統合テクノロジー・スタック（アプリケーションから、オペレーティング環境、仮想化ツールと管理ツール、ストレージとネットワーク・コンポーネントにいたるまで）を提供する唯一の企業です。そして、この統合プラットフォームを利用するアプリケーションの作成に必要なすべてのツールを開発者に提供するのは、Oracle Solaris Studioです。

はじめに

開発者は、企業向けのミッション・クリティカルなアプリケーションの作成にあたり、多くの課題に直面しています。特に、アプリケーション・パフォーマンスの最適化、プラットフォーム間の効率的かつ信頼性の高いアプリケーションの開発/実行、およびスケーラブルかつセキュアなアプリケーションの迅速な生成の必要性は重要な課題です。ツールの多くは単一目的のソリューションです。オープンソースの開発ツールを使用して間に合わせる組織もありますが、これにより混合環境が生まれ、開発者にとって効率や生産性を低下させる原因となります。

また、今日のシステムやプロセッサにはアプリケーション・パフォーマンスを大幅に向上できるさまざまな機能がありますが、開発者がこれらの機能を十分活用できるコードを作成しないかぎり利用することはできません。主要チップ・ベンダーからのマルチコアCPU（インテルXeon、AMD Opteron、SPARCなどのプロセッサ）の発売に伴い、現在および将来のプラットフォームに使用できるパラレルおよび並行ソフトウェア・アプリケーションを開発者がより簡単に作成できるツールが求められています。

Oracle Solaris Studioは、最高峰のエンタープライズ・オペレーティング・システムであるOracle Solaris向けに作成されており、業界をリードする開発環境です。Oracle Solaris Studioは、完全に統合された、機能豊富な開発プラットフォームであり、スケーラブルでセキュアな、信頼性の高いエンタープライズ・アプリケーションを短時間かつ低リスクで生成するために必要な、あらゆる開発者向けツールを搭載しています。また、C、C++、Fortranのコンパイラや高度なツールを搭載しているため、開発期間を大幅に短縮できます。

例を挙げると、Oracle Solarisの動的トレーシング（DTrace）テクノロジーに基づく新しいDLightビジュアル・プロファイリング・ツールや強力なスタンドアロンのdbxtool GUIデバッガによって、コード・エラーを視覚化し、マルチスレッド・コードを簡単にデバッグできます。高度なツール（パフォーマンス・ライブラリ、パフォーマンス・アナライザ、スレッド・アナライザ、メモリ・エラー検出ツール、コード・カバレッジ・ツール、コード・アナライザ、統合開発環境など）がコンパイラやデバッガと連携して機能し、マルチスレッドの高パフォーマンス・アプリケーションの構築、デバッグ、分析、およびチューニングのエンド・ツー・エンド・プロセス向けに最適化された開発プラットフォームを提供しています。開発者はこの統合環境独自の機能を使用することで、プラットフォームの機能とパフォーマンスを効果的に活用しながら、アプリケーション開発のプロセスを高速化かつ簡素化できます。

Oracle Solaris Studioの概要

Oracle Solaris Studioは、緊密に統合された生産的な環境で高品質/高パフォーマンスなアプリケーションを作成するのに必要なあらゆる機能を開発者向けに提供します。Oracle Solaris Studioを使用すれば、SPARCおよびSPARC64プロセッサ・ベースのOracleサーバー・システムであっても、インテル・プロセッサ搭載のx86およびx64システムであっても、開発者はOracleのシステムおよびソフトウェアに搭載された独自の機能を活用して、より効率的かつ強力なアプリケーション・パフォーマンスを実現できるようになります。

Oracle Solaris Studio統合開発環境

アプリケーション開発のプロセスが複雑化しているため、開発者は、統合され、最適化されたプラットフォームで作業する必要があります。Oracle Solaris Studioには、受賞歴のあるNetBeans IDEに基づくフル機能の統合開発環境（IDE）が含まれており、内蔵のコンパイラやデバッガと共に使用できるよう調整されています。Oracle Solaris Studioのツールのポートフォリオについては、図1を参照してください。

図1：次世代のOracle Solaris Studio IDEには、アプリケーション開発用の高度なツールがシームレスに統合されています。

Oracle Solaris Studioには、クロス・プラットフォームのデスクトップ、エンタープライズ、およびWebアプリケーションを作成するための豊富なツールが用意されています。またC、C++、およびFortranアプリケーションの機能とパフォーマンスを作成、編集、構築、デバッグするためのモジュールが含まれています。GUIで重要なのは、アプリケーション開発で重要なさまざまな領域を可視化できる多様な機能です。たとえば、次のような機能があります。

- 選択した関数からコールされる複数の関数、または選択した関数をコールする複数の関数のツリー・ビューを表示するCall Graphウィンドウ
- ソースおよびヘッダー・ファイルの階層を調べるInclude Hierarchyウィンドウ
- プロジェクトのソース・コードでクラス、関数、変数、マクロ、またはファイルが使用されるすべての場所を示すUsagesウィンドウ
- コード入力時の構文エラーおよびセマンティック・エラーの自動ハイライト
- コード補完
- 機能や変数に直接ジャンプしたり、含まれるファイルに移動したり、タイプを検索したりすることができるGoToメニュー項目

オラクルは開発者の生産性を上げるため、Oracle Solaris Studio IDEに次の機能を追加しました。

- プロファイリング、データ・レースとデッドロック検出、メモリ・アクセス・チェック、および静的エラー・チェック用ツールのIDEへの統合
- 既存バイナリをIDEにインポートし、そこからプロジェクトを作成可能
- リモート開発（リモート・ホストまたはネットワーク共有ファイル・システムでの構築、実行、デバッグ、プロファイリングなど）のオプション。リモート・ファイル・システムの参照とリモート・ホストとの組込みターミナル接続のオープンが含まれます。
- Microsoft Windows、Mac OS X、またはLinuxデスクトップ・システムでインストールおよび実行でき、リモート・ホスト上のOracle Solaris StudioコンパイラおよびツールにアクセスできるIDEのデスクトップ・ディストリビューションの生成
- C++集中型コードの強力なサポート（テンプレートおよび特殊化のナビゲーションなど）

Oracle Solaris Studio IDEでは、プログラマが実行するすべての手順（コード生成から編集、コンパイル、デバッグ、チューニングまで）を統合することで、高パフォーマンスなアプリケーションを簡単かつ迅速に構築できます。詳しくは、『*Oracle Solaris Studio 12.3 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。Oracle Solaris Studio IDEについては、図2を参照してください。

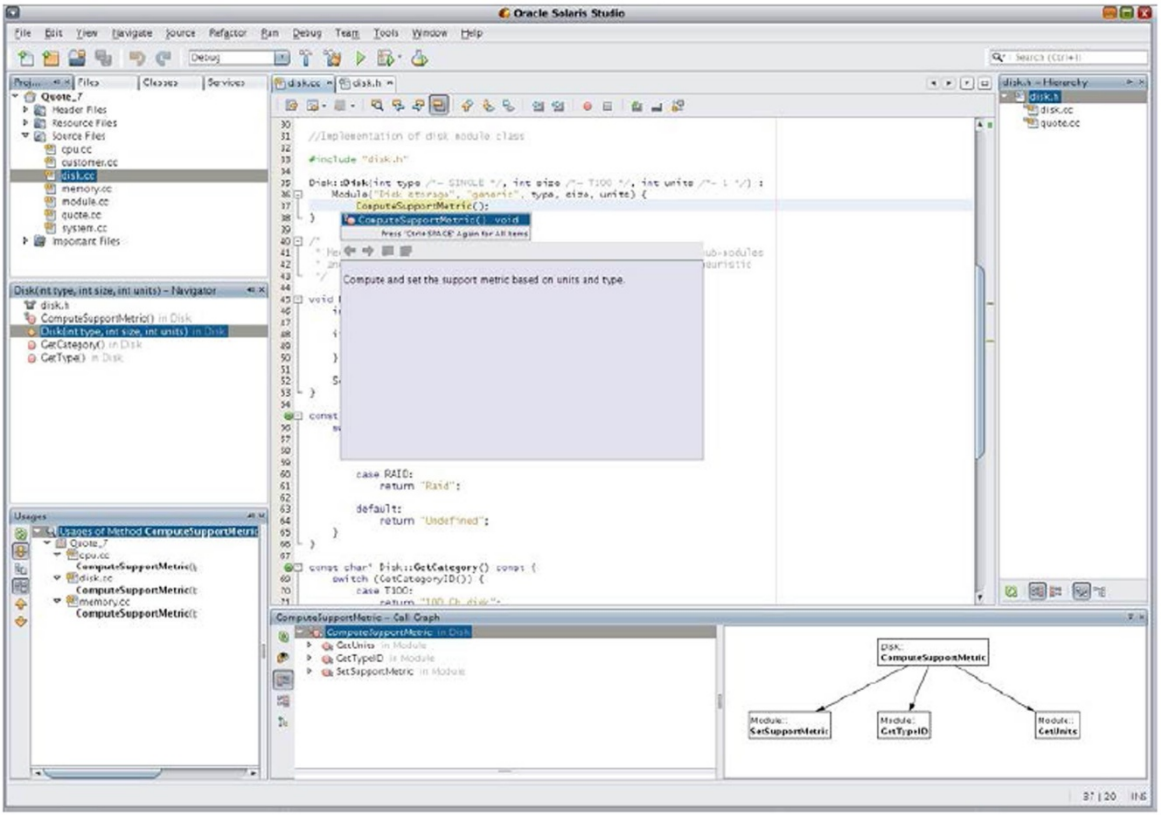

図2：Oracle Solaris Studio IDEには、開発者の生産性向上に役立つ強力なGUIが用意されています。

C、C++、およびFortranコンパイラによる、より優れた高速なアプリケーションの構築

プログラマが32ビットおよび64ビットの高パフォーマンス・アプリケーションを開発するには、高度なコンパイラとツールが必要です。また、最高の配置のパフォーマンスをもつ、最適化されパラレル化されたアプリケーションを開発できる必要があります。Oracle Solaris Studioは、最高パフォーマンスのパラレル化コンパイラを搭載した、現在のマルチスレッド開発における最先端のIDEです。またOracle Solarisおよび基盤となるハードウェア向けにチューニングされているため、Oracleシステム用の全体的な開発プラットフォームとして最も優れています。実際、Oracle Solaris StudioのC、C++、およびFortranコンパイラの記録的なアプリケーション・パフォーマンスは、他のオープンソース製品のパフォーマンスを常に上回っています。

プログラマはOracle Solaris Studioツールで、Oracleハードウェア製品ファミリー全体に移植可能な、効率的なアプリケーションを作成できます。このコンパイラは、SPARC、x64、およびx86プロセッサで動作するOracleシステム用の堅牢かつ高パフォーマンスなパラレル・コードを構築するための、信頼できる基盤となります。Oracle Solaris Studioコンパイラで使用可能な32ビット/64ビット・オプションの拡張セットを使用すれば、幅広いニーズに対応するようアプリケーションを構築できます。たとえば、あらゆるSPARCプロセッサで適切なパフォーマンスを出せるようアプリケーションをコンパイルできるため、SPARCプロセッサ搭載のすべてのOracleシステムで、アプリケーション・バイナリが共通化されます。

また、開発者は最新の64ビットUltraSPARCプロセッサの機能を利用する別のコンパイラ・オプションを使用することで、同じアプリケーションをコンパイルして最大パフォーマンスで動作させることが可能です。このためには、使用するプロセッサの種類を指定して、各種プロセッサの命令セット用に最適化されたコードをコンパイラが生成できるようにする必要があります。この結果、同じアプリケーション・ソース・コードを使用して各SPARCアーキテクチャ用に最適なアプリケーション・バイナリを作成でき、SPARC製品ライン全体のパフォーマンスを最大化できます。サポートされるプラットフォームのリストについては、http://www.oracle.com/technetwork/jp/server-storage/solarisstudio/overview/index.htmlを参照してください。

高度に最適化された自動パラレル化コンパイラの使用パフォーマンス向上の追求は、単なるプロセッサ速度への依存から、ハードウェアおよびソフトウェアの双方でのパラレル化へと変化しています。このためプログラマの間で、マルチスレッド・パラレル機能を持つアプリケーション開発への関心が高まっています。ハードウェアのパラレル化の好例として、プロセッサ（1チップに8個のコアと64個のスレッドを搭載したUltraSPARC T2プロセッサなど）が挙げられます。Oracle Solaris Studioのコンパイラは、マルチコアなマルチスレッド・ハードウェアの潜在能力を活用できるよう、最新のマルチコア・アーキテクチャ向けに最適化されています。またコンパイラの自動パラレル化に対応し、OpenMP（http://www.openmp.org/）およびMessage Passing Interface（MPI）のアプリケーション・プログラマ・インタフェース（API）をサポートしています。アプリケーションをマルチスレッド化することで、並行処理および再プログラムによって計算を分散できるタスクを識別し、マルチプロセッサ・システムでの効率を上げることができます。このようなアプリケーションのパラレル化プロセスによって、コンパイル済みプログラムを再計算し、マルチプロセッサ・システムの利点を最大限利用できます。

自動パラレル化では、コンパイラはループに焦点をあてます。複数のプロセッサにループの計算作業を分散します。この際、ソース・プログラムを変更する必要はありません。コンパイラによって、パラレル化するループおよびその分散方法が選択されます。コンパイラの主要な機能は、アプリケーションをパラレル化するために重要な機能を提供することです。たとえば次のような機能があります。

- ループ変換。コンパイラによっていくつかのループ再構築変換が実行され、プログラムでのループのパラレル化が改善されます。これらの変換の一部では、シングル・プロセッサのループの実行も改善できます。多くの場合ループには、並行して実行できない文が少数と、並行して実行できる文が多数含まれます。ループ・ディストリビューションによって、個別のループへの順次処理文の移動と、パラレル化可能な文の別のループへの収集が試行されます。ループ・ディストリビューションが常に有益または安全とは限りません。コンパイラの分析によって、ディストリビューションの安全性と有益性が判断されます。コンパイラは、ループ・ディストリビューションを慎重に実行するよう設計されています。
- 依存性の分析。一連の動作は、計算結果が実行順序に依存しない場合にのみ並行して実行できます。コンパイラでは、依存関係分析を実行して順序に依存しないループを検出できます。コンパイラによって、可能性のあるループ・パラレル化がすべて実行されるわけではない点に注意してください。パフォーマンスの向上と発生するオーバーヘッドを比較して、有益でないと判断される場合は、パラレル化されないループもあります。

詳しくは、『アプリケーションのパラレル・パフォーマンスを最適化する方法』を参照してください。

アプリケーション・パフォーマンスの向上

集中的な数学演算、数値アルゴリズム、数値的応用（行列乗算など）が、パフォーマンス上の問題をもたらす場合があります。Oracle Solaris Studioパフォーマンス・ライブラリは、サポートされているプラットフォーム・アーキテクチャを最大限に利用して最適なパフォーマンスを実現する高速演算サブルーチンおよび機能を収集したものです。開発者がこれらのルーチンを利用することで、演算の線形代数、高速フーリエ変換（FFT）、およびその他の数値集中型の問題を解決できます。Oracle Solaris Studioパフォーマンス・ライブラリのルーチンは、Fortran、C、およびC++プログラムからコール可能です。また、BLAS1、BLAS2、BLAS3、LAPACK、PBLAS、BLACS、FFTPACK、VFFTPACK、SPSOLVE、Sparse BLAS、およびSuperLUには、オラクルによるルーチンの拡張実装が含まれます。これらのルーチンの基本バージョンの多くは、元のNetlibライブラリで公開されており、入手できます。ただし、これらのライブラリは特定のシステム向けに高度に最適化およびパラレル化されているわけではありません。

Oracle Solaris Studioパフォーマンス・ライブラリには、これらのルーチンのスカラー設定およびパラレル設定の両方が含まれており、インタフェースや機能は元のNetlibライブラリと同じです。Oracle Solaris Studioパフォーマンス・ライブラリは静的および動的ライブラリ・バージョンの両方で使用可能で、プロセッサ・アーキテクチャのSPARCおよびx86/x64ファミリーを使用するシステム向けに最適化されています。このため、Netlibディストリビューションの代わりにOracle Solaris Studioパフォーマンス・ライブラリを使用することをリンク時に指定するだけで、ユーザー・コードの速度が自動的に大幅に向上します。このアプローチでは、ソース・コードの変更やアプリケーションの再コンパイルは不要です。

Oracle Solaris Studioパフォーマンス・ライブラリでは、OpenMPまたはMPI APIを使用して、選択した関数をパラレル・コーディングすることでマルチコアまたはマルチプロセッサ・システムを利用します。これはOracle Solarisスレッド、POSIXスレッド、OpenMP、またはMPIを使用するプログラムで使用可能です。Oracle Solaris Studioパフォーマンス・ライブラリには、シングル・コアでパフォーマンスを大幅に向上できるようチューニングされた多くの主要なカーネル・ルーチンが含まれています。これにより、マルチコア、共有メモリ、および分散メモリ・パラレル環境でのパフォーマンスが自動的に向上します。多くのルーチンに対して、Oracle Solaris Studioパフォーマンス・ライブラリには、パラレル・パフォーマンスを向上するためのパラレル・アルゴリズムが含まれています。

開発者はOracle Solaris Studioパフォーマンス・ライブラリの次の機能を使用することで、アプリケーションのパフォーマンスを上げ、より効率的なコードを開発できます。

- 異なるライブラリ間で一貫性のあるAPI
- 拡張/新規追加された標準ルーチン（精度が向上している場合もあります）
- SPARCおよびx86/x64の特定の命令セット・アーキテクチャ向けに最適化
- 64ビット対応Oracle Solarisのサポート
- SPARCおよびx86/x64プラットフォームのパラレル処理コンパイラ・オプションのサポート
- 複数ある、プロセッサのハードウェア・オプションのサポート
- Oracle Solaris Studioパフォーマンス・ライブラリの簡易バージョンを作成したり、ルーチンが必要とするサイズまでライブラリを圧縮したりするオプションを提供する新しいカスタム・ライブラリ・ツール

Oracle Solaris Studioパフォーマンス・ライブラリについて詳しくは、http://docs.oracle.com/cd/E22054_01/index.html のユーザーズ・ガイドを参照してください。

アプリケーション・デバッグの簡素化

今日の開発者は、アプリケーションで問題が発生している箇所の特定という非常に困難な課題に直面しています。マルチコア・ハードウェア・プラットフォームで実行されるマルチスレッド・プログラムでは、アプリケーションのデバッグに手間と時間がかかってしまう場合があります。最適なパフォーマンスを実現するには、開発者がアプリケーション内の問題の内容と該当箇所を迅速に把握できる必要があります。

DiscoverとUncover

Oracle Solaris Studioには、アプリケーション開発における最も難しい局面を解決するための、メモリ・エラー検出ツール（Discover）とコード・カバレッジ・ツール（Uncover）という2つのツールがあります。

プログラム内のメモリ関連のエラーは、開発者にとって最もデバッグしにくいもので、プログラムの動作エラーの原因となることもよくあります。ソース・コード内のエラー箇所を見つけることは困難な場合があります。Discoverソフトウェアは、メモリ・アクセス・エラー検出用の高度な開発ツールです。Discoverは使いやすく、コンパイラが提供するあらゆるバイナリ（完全に最適化されたバイナリを含む）と連携して機能します。Discoverを使用するには、バイナリを1つのコマンドでインスツルメント化し、通常の方法で実行します。これにより、プログラム実行時にメモリ・アクセス・エラーを検知して動的にレポートされます。Discoverでは、実行中に検出されたメモリ異常のレポートが生成されます。このレポートは、テキスト・ファイルとして表示したり、WebブラウザでHTMLファイルとして表示したりすることができます。開発者は、Discoverが、実行されるコードに対して作用するという点に注意する必要があります。実行時に実行されない部分のユーザー・コードにあるエラーはレポートされません。

Discoverで検出可能なタイプのエラーとしては、たとえばプログラムによって配列が割り当てられ、初期化されずに1つの配列ロケーションからの読取りが試行されて、予想外の結果となるエラーなどがあります。Discoverで検出できるその他のエラーとしては、未割当てメモリの読取り/書込み、割当て済み配列バウンド以外でのメモリ・アクセス、解放されたメモリの不正使用、不正なメモリ・ブロックの解放、メモリ・リークなどがあります。Discoverは、Oracle Solaris 10 10/08オペレーティング・システム以降のSolaris 10 Update、Oracle Solaris 11 Express、Oracle Solaris 11を実行するシステム上の、Sun Studio 12、Sun Studio 12 Update 1、Oracle Solaris Express 6/10、Oracle Solaris Studio 12.2、Oracle Solaris 12.3コンパイラまたはGCC for Sun Systemsコンパイラ（バージョン4.2.0以降）でコンパイルされたバイナリで動作します。DiscoverからのHTMLレポートについては、図3を参照してください。

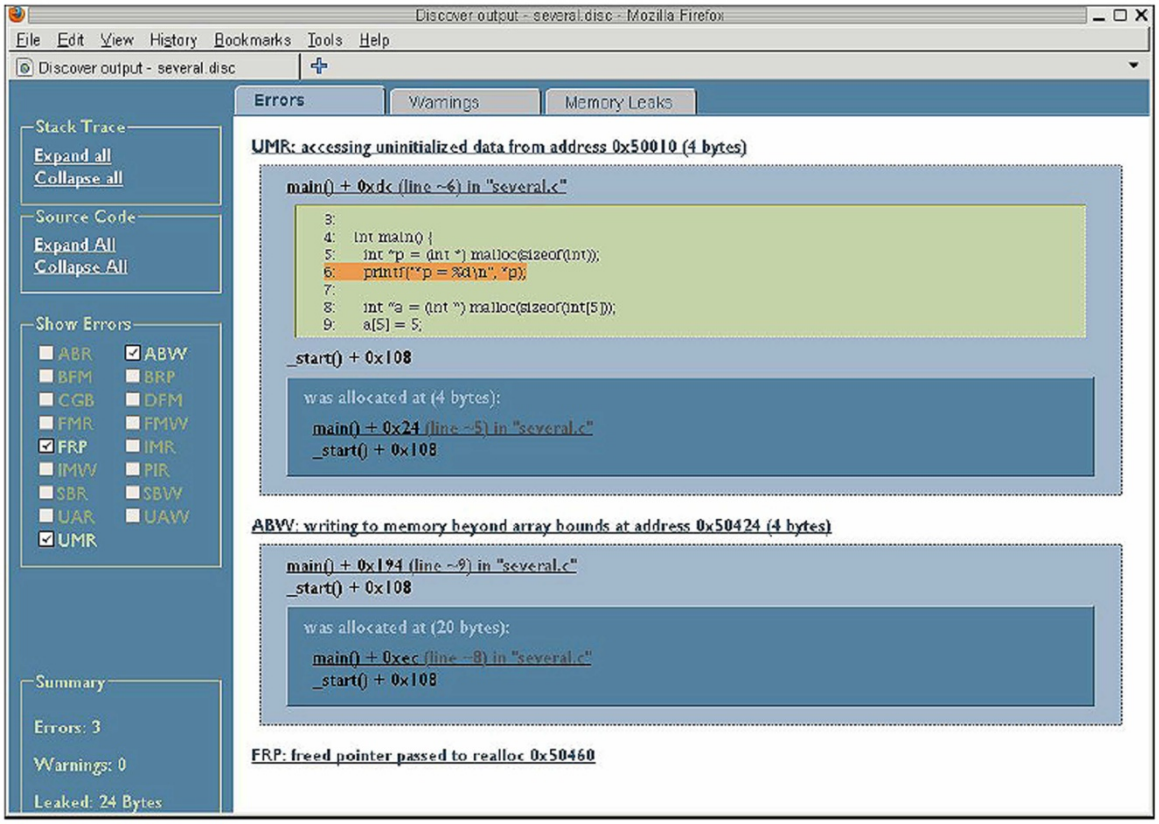

図3：開発者はDiscoverによって、見つけにくいメモリ・エラーを検出できます。

開発者にとって、新しいコードを本番使用前に徹底的にテストすることは非常に重要です。コード・カバレッジは、ソフトウェア・テスト・プロセスの一側面であり、テスト時のソース・コードの実行領域（または非実行領域）について情報を提供します。開発者はこの情報を得ることで、テスト・スイートを改善してより多くのコード部分をより効率的にテストできます。

Oracle Solaris Studioコード・カバレッジ・ツール（Uncover）は、アプリケーションのコード・カバレッジを測定するための、シンプルで使いやすいコマンドライン・ツールです。Uncoverにはuncoverageと呼ばれる独自の機能があり、テストされていないバイナリ内にある主要な機能領域を迅速に検索できます。Uncoverによってレポートされるカバレッジ情報は、機能、文、基本ブロック、または命令レベルです。

Uncoverは、マルチスレッドおよびマルチプロセスの安全性を確保するため、バイナリのインスツルメント化、テストの実行、および結果の表示を行う簡単なプロシージャを提供します。Uncoverではカバレッジ・テスト用の特別なビルドが不要であるため、開発者は本番用の通常のバイナリまたは最適化されたバイナリを使用して、ツールを簡単に使用できます。多くのプログラムでは、コードのインスツルメント化の際に速度が低下しますが、インスツルメント化されていないコードに関連するUncoverでは、パフォーマンスへの影響はわずかです。パフォーマンス・アナライザに表示されるUncoverからのカバレッジ・レポートについては、図4を参照してください。

DiscoverおよびUncoverについて詳しくは、『*Oracle Solaris Studio 12.3 DiscoverおよびUncoverユーザーズ・ガイド*』を参照してください。

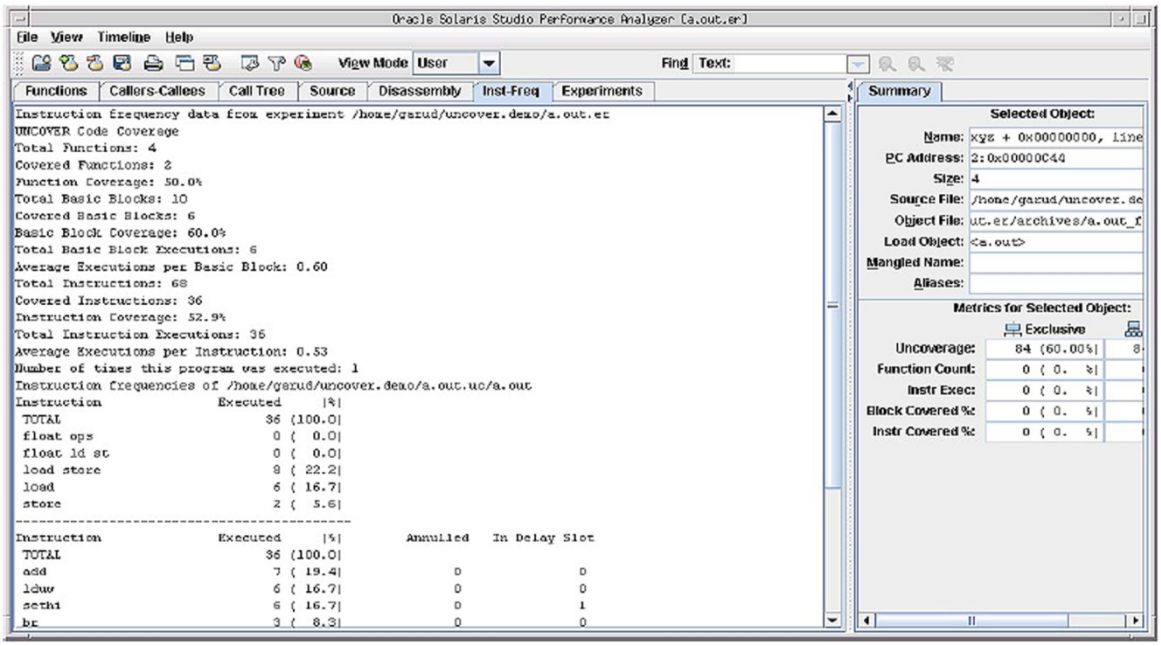

図4：Oracle Solaris Studioコード・カバレッジ・ツールでは、コード・カバレッジを命令レベルで測定できます。

コード・アナライザ

Oracle Solaris StudioコードアナライザはOracle SolarisのCおよびC++アプリケーションの開発者がセキュアで堅牢な、高品質のソフトウェアを開発できるように設計された、統合ツール・セットです。コード・アナライザによって、コンパイラが収集した静的コード・エラー・データと、Discoverが収集した動的メモリ・アクセス・エラー・データおよびUncoverが収集したコード・カバレッジ・データが統合されます。コード・アナライザでは、統合したエラー・データを分析することで、独立して動作する他のエラー検出ツールでは検出できないコード内のエラーを見つけることができます。また、コード内の核心となる問題（修正すれば他のほとんどのエラーを解決できるような問題）を特定できます。

特別なコンパイラ・オプションでコードをコンパイルした場合、コンパイラによって静的コード・エラー（配列バウンド以外での読取り/書込み、メモリ解放の重複、解放されたメモリの読取り/書込み、無限の空ループ、メモリ・リーク、および初期化されていないメモリからの読取りなど）が検出されます。エラー・データは、コード・アナライザで使用できるようディレクトリに保存されます。Discoverでコードをインスツルメント化し実行すると、メモリ・アクセス・エラー・データが追加されます。Uncoverでコードをインスツルメント化し実行すると、コード・カバレッジ・データが追加されます。

コード・アナライザのGUIでは、1種類、2種類、または3種類すべてのデータを分析および表示できます。このため開発者は、問題の詳細（エラー・パスやコール・スタックなど）を表示し、スタック内の関数コールから関連するソース・コード行にジャンプし、コード内で関数が使用されている箇所をすべて見つけ出し、関数のコール・グラフを表示して、データを調べることができます。また、各エラー・タイプについての参考情報（コード例や、可能性のあるエラー原因など）も表示されます。開発者は検出されたすべてのエラーを表示することも、他のエラーの原因である可能性が高い核心となる問題のみを表示することもできます。コア・アナライザに表示されるエラー・データについては、図5を参照してください。

詳しくは、マニュアル『*Oracle Solaris Studio 12.3コード・アナライザ・ユーザーズ・ガイド*』と『*Oracle Solaris Studio 12.3コード・アナライザ・チュートリアル*』を参照してください。

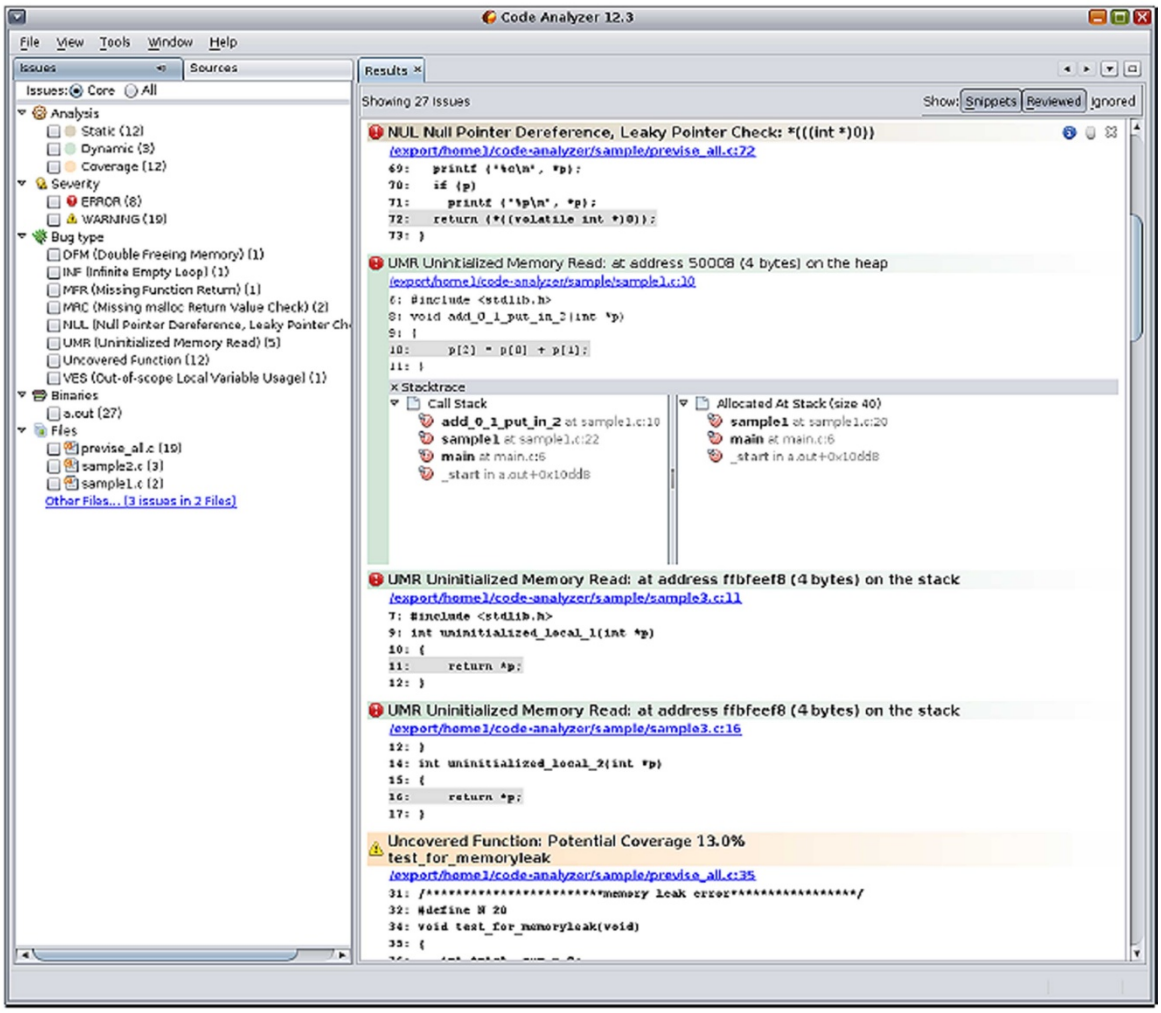

図5：コード・アナライザに表示されるエラー・データ

dbxデバッガとGUIベースのdbxtool

Oracle Solaris Studioには、dbxとdbxtoolという、2つの強力なデバッグ・ソリューションがあります。dbxtoolは、dbxのスタンドアロンのグラフィック・ベース・バージョンです。dbxは長い年月をかけて有効性が証明された対話型デバッグ・ツールで、制御された状態でプログラムを実行し、停止したプログラムの状態を調査する機能があります。dbxによって、開発者はプログラムの動的実行（パフォーマンス・データの収集など）を完全に制御できます。dbxを使用すると、実行可能ファイル、コア・ファイル、さらには（実行中のプロセスに接続することで）実行中のプログラムでも迅速かつ簡単にデバッグできます。また、マシン命令レベルでのデバッグにも使用できます。

スクリプト対応のdbxデバッガはマルチスレッド対応で、Oracle SolarisスレッドまたはPOSIXスレッドを使用するマルチスレッド・アプリケーションをデバッグできます。ユーザーは、各スレッドのスタック・トレースを検証したり、すべてのスレッドを再開したり、特定のスレッドの使用または不使用を選択したり、スレッド間を移動したりすることができます。またデバッガでは、"修正と続行"がサポートされます。これは変更後にプログラム全体の再コンパイルやデバッグ・セッションの再起動を行うことなく、ソース・ファイルを再リンクするプロセスです。プログラムの再構築や、デバッガの最初からの再開といった時間のかかる手順が不要となるため、プログラムをより迅速に機能させることができます。また、開発者はdbxによって重要なメモリ情報（使用率、リーク、アクセス箇所など）を入手できるため、難しいバグを突き止めることができます。

dbxデバッガには、コマンドライン、IDE、またはdbxtool（dbxのすべての機能にアクセスできるスタンドアロンのグラフィカル・インタフェース）からアクセスできます。dbxtoolのグラフィカル・インタフェースについては、図6を参照してください。dbxtoolとIDE実装のおもな違いは次の2点です。

- dbxtoolでは、dbxコマンドラインと同様に、デバッガに入りコマンドを実行してからアプリケーションのデバッグを開始できます。
- dbxtoolでは、IDEプロジェクトは認識されません。dbxコマンドラインと同様にファイル名が使用されます。

dbxについて詳しくは、マニュアル『*Oracle Solaris Studio 12.3：dbxコマンドによるデバッグ*』を参照してください。dbxtoolについて詳しくは、『*Oracle Solaris Studio 12.3：dbxtoolチュートリアル*』を参照してください。

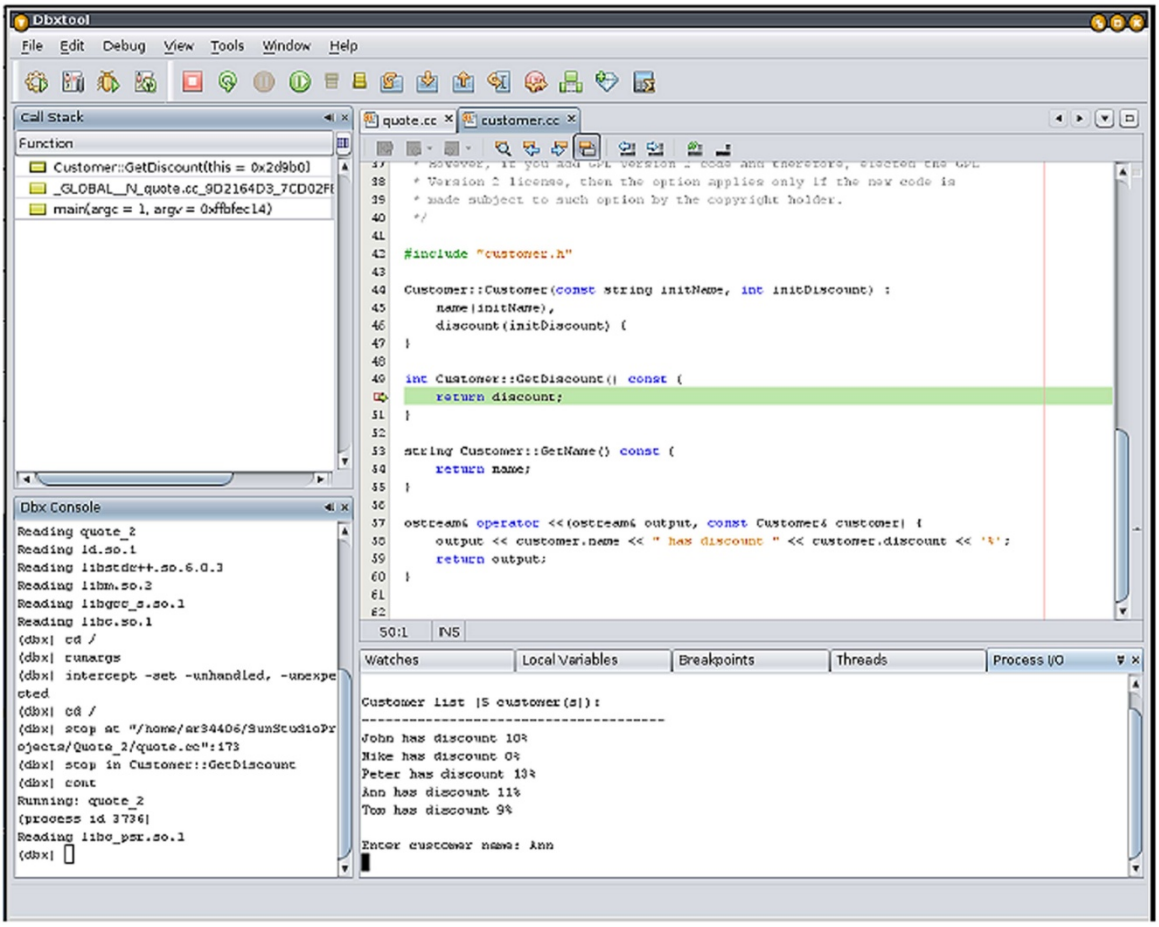

図6：スタンドアロンのdbxtoolによるアプリケーションのデバッグ

実行時検査

メモリ関連の欠陥があるアプリケーションをデバッグする場合、アプリケーション開発者は、ソース・コードのサイズを増大させずに、またはソース・コードをインスツルメント化せずに、メモリ使用率に関するデータを監視および収集し、見つけにくいメモリの問題を検出できる効率的な方法が必要です。実行時検査はOracle Solaris Studioの内蔵デバッグ機能で、開発フェーズ中に、メモリ・アクセスやメモリ・リークなどのアプリケーションのランタイム・エラーを自動的に検出します。エラーが検出されると、デバッガによってプログラム実行が中断され、関連するソース・コードが表示されます。このため開発者がバグを検出時に修正でき、開発の生産性とアプリケーションの品質が大幅に向上します。

実行時検査ではマルチスレッド・コードがサポートされているため、より複雑なマルチスレッド・アプリケーションのデバッグに有用な手段となります。Oracle Solaris Studioの実行時検査はすべての言語で動作し、再コンパイル、再リンク、Makefileの変更が不要です。Oracle Solaris Studioのユーザーは、実行時検査を使用しながら、パフォーマンス・データの収集以外のすべてのデバッグ操作を実行できます。

個々のプログラム・モジュールを開発しながら、開発サイクルの早い時期に実行時検査を使用することをお勧めします。実行時検査を段階的に使用して各モジュールを個別にチェックすることで、1回で処理するエラーの個数を減らすことができます。このアプローチによって、すべてのモジュールを完全なプログラムに統合した際に新たに生じるエラーが少なくなります。検出されるエラーがゼロまで減った場合は、モジュールの変更時にのみ実行時検査を再実行する必要があります。

詳しくは、マニュアル『*Oracle Solaris Studio 12.3：dbxコマンドによるデバッグ*』の第9章「実行時検査」を参照してください。

スレッド・アナライザ

スレッド・アナライザは、一般的ではありますがデバッグが非常に難しい、マルチスレッド・コード内の問題を特定するためのツールです。スレッド・アナライザには強力なスレッド分析機能が搭載されており、複数スレッドにまたがるプログラムの実行を分析し、データ・レースやデッドロックの状態を検出します。

データ・レース状態は、1つのプロセス内の複数のスレッドが同じメモリ・ロケーションに同時にアクセスし、1つ以上のスレッドが書込み操作を試行し、そのスレッドが排他的ロックを使用せずにそのメモリへのアクセスを制御している場合に発生します。データ・レースはマルチスレッド・コードによくある落とし穴で、後でプログラムを実行する際に不正または予想外の結果をもたらす原因となる場合があります。またその場合、問題の原因の特定が難しくなります。スレッド・アナライザによって、マルチスレッド・プログラムの実行時のデータ・アクセスが追跡され、見つかったデータ・レースがレポートされます。このツールではデータ・レースが自動的に検出されるため、マルチスレッド・プログラミングのタスクが簡単になります。

デッドロックはマルチスレッド・コードのもう1つの落とし穴で、あるスレッドが一方のスレッドによって保有されているリソースを待機してブロックされ、2番目のスレッドが最初のスレッドによって保有されているリソースを待機してブロックされた場合に発生します。スレッド・アナライザによって、ロックの不適切な使用によるデットロックを検出できます。このツールでは、プログラムがハングする原因となる実際のデッドロックと、特定の実行時にはプログラムをハングさせない可能性のある潜在的なデッドロックを検出できます。

スレッド・アナライザのインタフェースはマルチスレッド・プログラムの分析用に合理化されており、Races、Deadlocks、Dual Source、Race Details、およびDeadlock Detailsのタブが表示されます。スレッド・アナライザでは、POSIXのスレッドAPI、Oracle Solarisオペレーティング・システムのスレッドAPI、OpenMPディレクティブ、またはこれらを組み合わせて記述されたコードがサポートされます。スレッド・アナライザのインタフェースについては、図7を参照してください。

スレッド・アナライザについて詳しくは、マニュアル『*Oracle Solaris Studio 12.3：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。

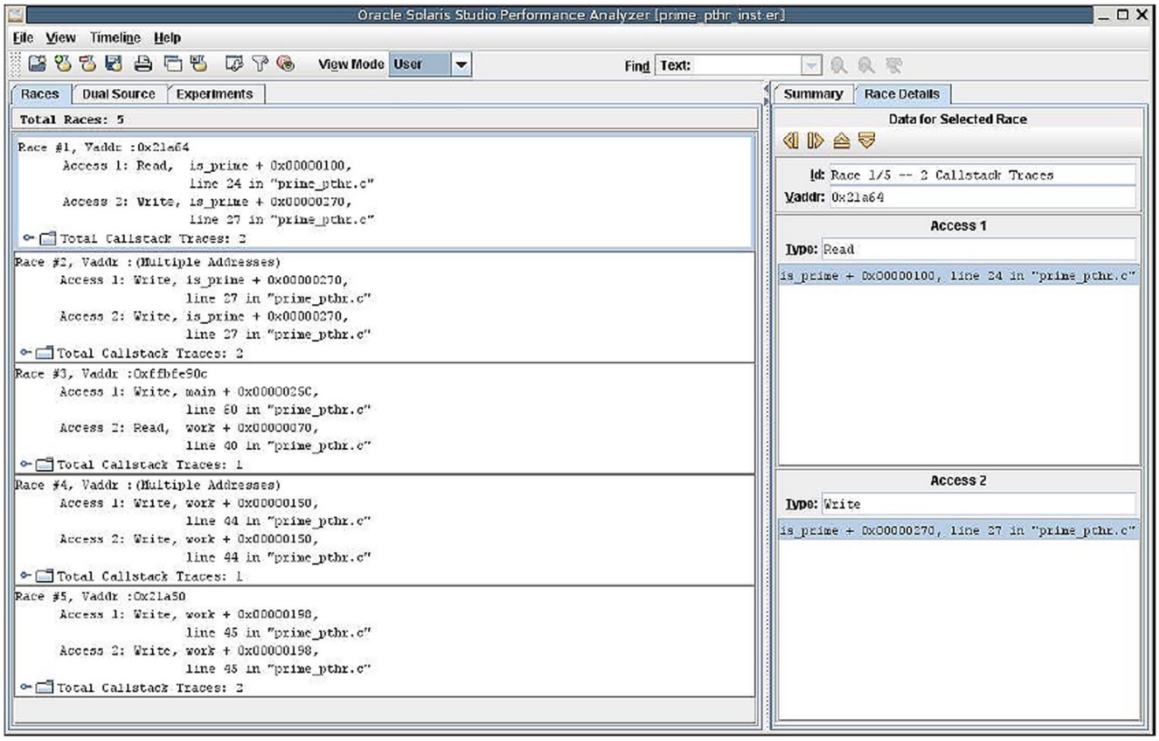

図7：スレッド・アナライザは、データ・レースやデッドロックなどの非常に難しいマルチスレッド・コード・エラーのトラブルシューティングに役立ちます。

アプリケーションの動作について

開発者は、アプリケーションの特定の動作の原因を特定しようとする場合、多くの困難な問題に遭遇します。Oracle Solaris Studioには、アプリケーションをスレッド・レベルで分析し、全体的なパフォーマンス分析を実行するためのツールが含まれています。

パフォーマンス・アナライザ

マルチスレッド・アプリケーションが複雑化するにつれ、開発者がアプリケーション・パフォーマンスを監視したり、問題のある領域を特定したりすることが難しくなっています。パフォーマンス・アナライザには、アプリケーション・コードのパフォーマンスの評価、パフォーマンスの潜在的問題の識別、および問題が発生しているコード内の場所の特定に役立つツールが含まれています。パフォーマンス・アナライザは、C、C++、Fortran、Java、またはこれらの言語を組み合わせて、OpenMP、MPI、またはPthreadsプログラミング・モデルを使用して開発されたアプリケーションをサポートします。

開発者は通常、プロファイリング・ツール（`prof`および`gprof`）を使用して、他のモジュールをコールしているプログラミング・モジュールや他のモジュールからコールされているモジュールを一覧表示することで、プログラムを分析します。データが収集され、`gprof`コマンドによってデータ結果が解析されます。この分析によって、関数が過度に再帰的でパラレル化が必要であるかどうかを把握できます。ただし、これらのプロファイリング・ツールは複雑なプログラムのチューニングには不向きです。開発者は累積時間を計算する際、`gprof`の分析結果のみに依存するべきではありません。

gprofは、すべての関数のコールに同じ時間がかかることを前提としているためです。この前提は正しくないため、誤ったデータが出る原因となります。

開発者は、profとgprofの代わりにOracle Solaris Studioツールであるコレクタとパフォーマンス・アナライザを組み合わせて、アプリケーション・パフォーマンス・データを収集および分析できます。これらのツールによって、一般的に使用されているプロファイリング・ツール（profおよびgprof）より、柔軟、詳細、かつ正確な分析を実行できますが、これらのツールはgprofで発生しうるエラーを対象にしていません。いずれのツールも、コマンドラインまたはグラフィカル・ユーザー・インタフェースから使用できます。

コレクタ・ツールおよびパフォーマンス・アナライザ・ツールを組み合わせて使用することで、次のデータを入手できます。

- プログラムが消費する使用可能なリソース量
- リソースを最も多く消費している関数またはロード・オブジェクト
- リソースの消費に関連しているソース行および命令
- プログラム実行における、このポイントへの到達方法
- 関数オブジェクトやロード・オブジェクトが消費しているリソース

コレクタによって、3種類のデータ（プロファイリング、トレーシング、およびグローバル・データ）が収集されます。プロファイリングによって、プロファイリング・クロックまたはハードウェア・カウンタのオーバーフローによってトリガーされるイベントと、プロセス・コール・スタックを含むイベントが記録されます。トレーシング・データによって、さまざまなライブラリ関数のラッパーの介在に基づいてイベントが記録されます。また、コールに関する情報（コールにかかった時間、パラメータ、プロセス・コール・スタックなど）も記録されます。グローバル・データやサンプリングは、さまざまなシステム・ルーチンのコールによる情報取得によって収集されます。プロファイリング・データとトレーシング・データには、特定のイベントに関する情報が含まれており、いずれのタイプのデータもパフォーマンス・メトリックに変換されます。グローバル・データはメトリックに変換されませんが、特定の時間セグメント中のプログラム実行の概要を提供するために使用されます。

コレクタによって収集できるデータのタイプには、次のようなものがあります。

- **クロックベースのプロファイリング・データ。**クロックベースのプロファイリングは、収集されるデフォルト・データであり、プログラムで時間がかかる箇所の統計サンプリングを示します。Oracle Solarisでは、このデータにユーザーCPU時間、システムCPU時間、ロック待機時間、CPU待機時間などが含まれます。Linuxでは、ユーザーCPU時間だけをカーネルから取得できます。
- **ハードウェア・カウンタ・オーバーフロー・プロファイリング・データ。**ハードウェア・カウンタでは、キャッシュ・ミス、キャッシュ・ストール・サイクル、浮動小数点演算、ブランチ予測ミス、CPUサイクル、および実行された命令などのイベントが追跡されます。ハードウェア・カウンタ・オーバーフロー・プロファイリングは、スレッドが動作しているCPU上の所定のハードウェア・カウンタがオーバーフローした場合に、プロファイル・パケットを記録するプロセスです。コレクタによって、プロファイル・パケット中のオーバーフロー値とカウンタ・タイプが記録されます。カウンタはリセットされ、カウントが続行されます。ハードウェア・カウンタのオーバーフロー・プロファイリングは、CPUに送られる情報またはCPUから取得される情報の流れに関する問題を診断するのに便利です。使用可能な具体的なハードウェア・カウンタは、マシン内の特定のCPUチップに依存します。

- **同期待機トレーシング・データ。**マルチスレッド・プログラムでは、別々のスレッドで実行されるタスクの同期が原因で、プログラム実行が遅延する場合があります（あるスレッドが他のスレッドによってロックされているデータへのアクセスを待機しなければならない場合）。ロックの待機にかかる時間は、待機時間と呼ばれます。これらの同期遅延は、Oracle Solarisまたはpthreadスレッド関数へのコールのトレースによって収集されます。また、これらのイベントの収集および記録プロセスは、同期待機トレーシングと呼ばれます。この情報を使用して、スレッド競合の有無や、パフォーマンス向上のためプログラム修正が必要かどうかを判断できます。
- **ヒープ・トレーシング（メモリ割当て）データ。**ヒープ・トレーシングでは、コレクタが標準Cライブラリのメモリ割当て関数（malloc、realloc、memalign）と割当て解除関数（free）に介入することで、メモリ割当ておよび割当て解除リクエストがトレースされます。これらのメモリ割当ておよび割当て解除関数が適切に管理されていない場合は、非効率なデータ使用やプログラム・パフォーマンスの低下の原因となります。ヒープ・トレーシング・データの収集は、プログラム内のメモリ・リークの識別や、非効率なメモリ割当て箇所の特定に役立ちます。
- **SPARCでのデータ領域プロファイリング。**データ領域プロファイリングでは、イベントの原因となるデータオブジェクト参照に対して、メモリ関連イベントが発生している命令ではなく、メモリ関連イベント（キャッシュ・ミスなど）がレポートされているデータが収集されます。データ領域プロファイリングは、SPARCアーキテクチャ用にコンパイルされたCプログラムで実行できます。Linuxシステムでは使用できません。
- **カーネル・プロファイリング。**コマンドer_kernelでは、クロック・カウンタ・プロファイリングまたはハードウェア・カウンタ・プロファイリングを使用して、Oracle Solarisカーネルのプロファイリングを実行できます。これにより、ユーザー・プログラムに対するOracle Solarisの動作状況を把握できます。また、ユーザー・プロセスを同時に測定できます。

パフォーマンス・アナライザ・ツールでは、コレクタが記録したデータの表示、データの処理、およびプログラム、関数、ソース行、命令レベルでのさまざまなパフォーマンス・メトリックの表示が行われます。また時間の関数として、RAWデータもグラフィカル形式で表示されます。

詳しくは、マニュアル『*Oracle Solaris Studio 12.3：パフォーマンス・アナライザ*』を参照してください。パフォーマンス・アナライザについては、図8を参照してください。

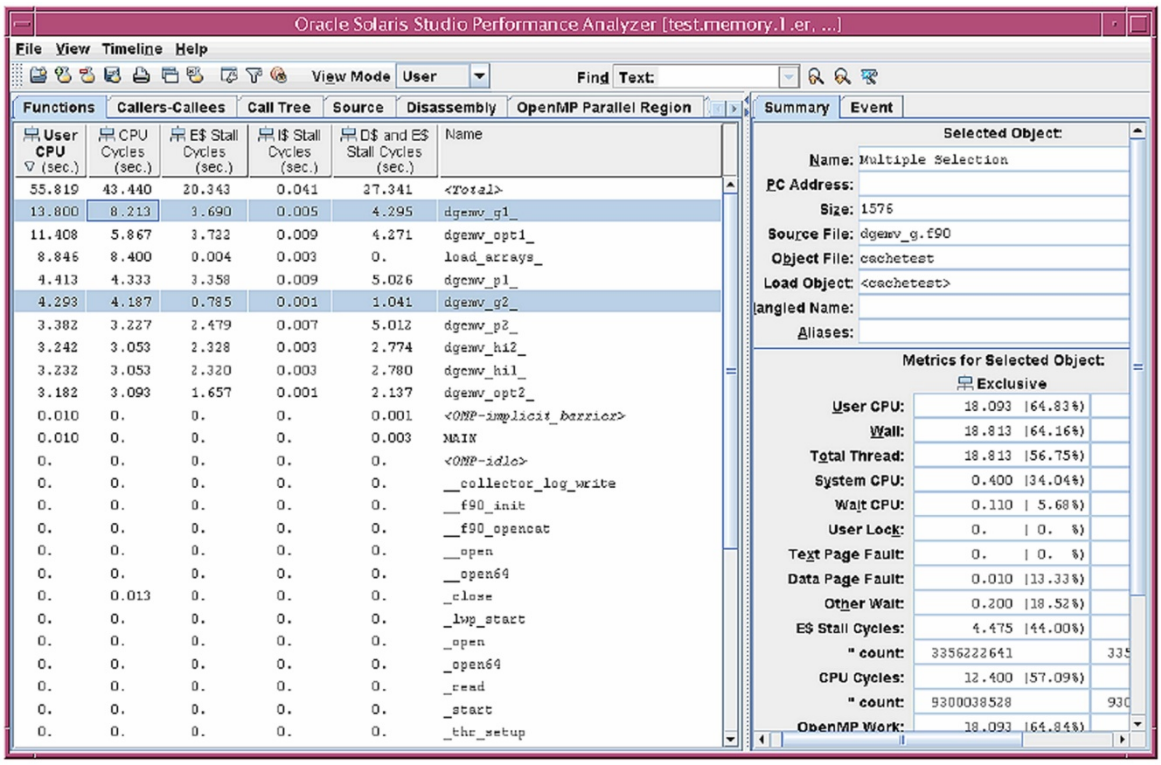

図8：パフォーマンス・アナライザには、有用なプログラム分析が表示されます。

DLight

システム・プロファイリング・ツールでは、開発者がシステムの動作状態を調査および把握したり、多くのソフトウェア・レイヤーにわたるパフォーマンスの問題を識別したり、システムやアプリケーションの異常動作の原因を特定したりすることができます。DLightは、Oracle SolarisプラットフォームのOracle Solarisの動的トレーシング（DTrace）テクノロジーを使用してアプリケーションおよびシステム・プロファイリングを統合するシンプルなドラッグ・アンド・ドロップ・インタフェースを搭載した、新しいGUIツールです。

DTraceは、Oracle Solarisに組み込まれた包括的な動的トレーシング機能です。ユーザー、管理者、開発者がこの機能を使用することで、ユーザー・プログラムやオペレーティング・システムの動作を、新しい独自のレベルで監視できます。Oracle Solarisのカーネル、ユーティリティ、およびその他のソフトウェア・コンポーネントには、数千ものトレーシング・ポイントやプローブが組み込まれているため、ユーザー指定プローブを動的にインスツルメント化して、データを記録したりシステムを詳しく調べたりすることができます。

他のプロファイリング・ツールとは異なり、DTraceによってターゲット・システムに対して変更が加えられることはありません。変更によってプロファイリングおよび非プロファイリング・コードの動作に差異が生じたり、エラー調査の異常につながったりする可能性があるためです。DTraceによって、テスト対象のシステムにパフォーマンス上の影響をほとんど与えずに、実際の本番システムをテストできます。DTraceは、セキュリティ、完全な安全性、およびエラー・チェックに主眼をおいて設計されているため、実行中のシステムに悪影響を与えることはありません。このため、必要に応じて安心して実際の本番システムで使用できます。DTraceを使用することで、アプリケーション、ユーザー・プロセス、およびオペレーティング・システム・カーネルの内部動作を安全かつ正確に把握でき、システムのトラブルシューティングや問題解決を簡単かつ迅速に行うことができます。

DLightでは、DTraceのデバッグおよびパフォーマンス分析機能を利用および視覚化しています。開発者はターゲット・アプリケーションを選択し、ターゲット・アプリケーションを実行しながら監視対象のDLightインスツルメンテーションを選択し、ツールによって返されるデータを分析できます。返されるデータを使用して、分析対象のアプリケーションの動作が正常になるまで、実験を再帰的に改良できます。DLightツールは、接続されているプロセスや実行可能ファイルに適用できます。DLightを使用した分析については、図9を参照してください。

詳しくは、『*Oracle Solaris Studio 12.3 DLight チュートリアル*』を参照してください。

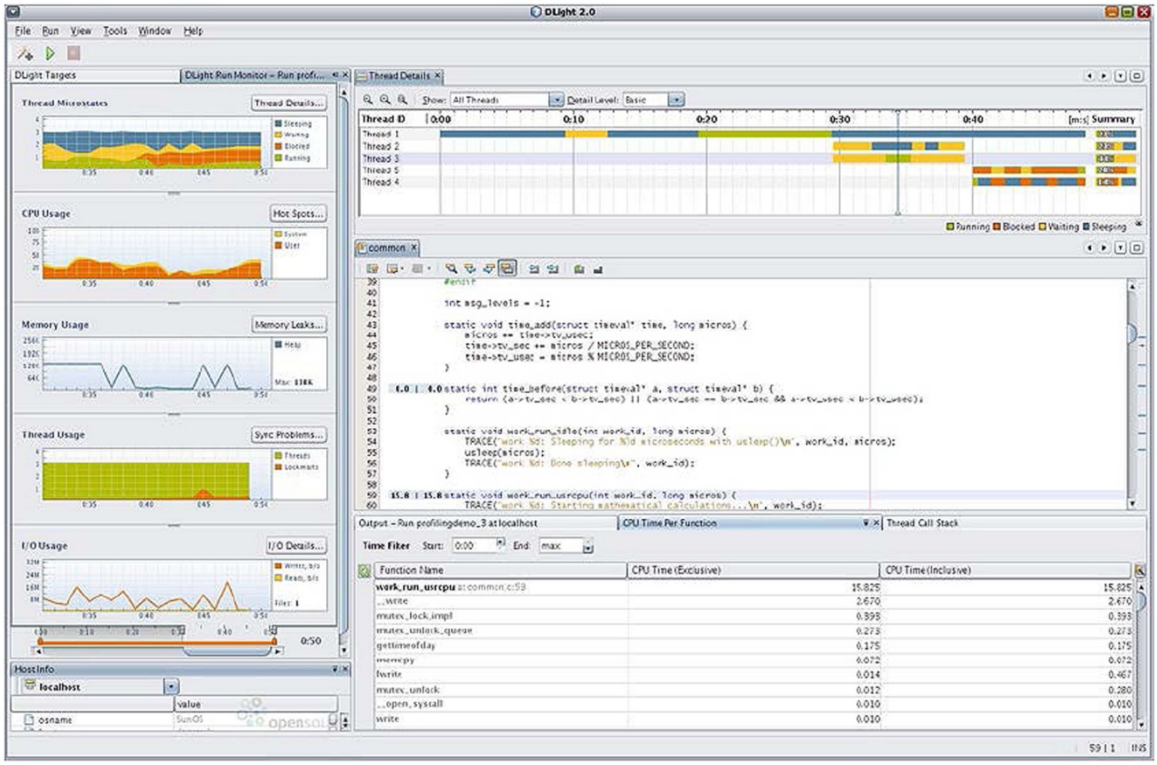

図9：Dlightでは、DTraceテクノロジーを使用してプログラムを詳細に分析します。

結論

強力なエンタープライズ・システムを利用している組織の場合、開発者はハードウェアを活用し、パラレル・アプリケーションのエンド・ツー・エンドの開発を合理化するために、高度なツールを必要とします。Oracle Solaris Studioは、すべてのOracleサーバーで実装を最適化できるよう設計されており、アプリケーション・パフォーマンスを最大限に最適化できる製品として評価されています。プログラマは、アプリケーションの開発サイクル全体を通して、連携していないさまざまなツールの代わりに、統合されたOracle Solaris Studioスイートを使用することで、基盤となるプラットフォームの利点を最大限に生かした高パフォーマンス・アプリケーションを迅速かつ簡単に作成できます。

謝辞

このホワイト・ペーパーの作成にあたり、次の方々に多大なご協力をいただきました。深く感謝いたします。Marty Itzkowitz、Nawal Copty、Leonid Lenyashin、Thomas Preisler、David Ford、Raj Prakash、Bhawna Mittal、Ann Rice、Don Kretsch

追加情報

Oracle Solaris Studioおよび関連するテクノロジーについて詳しくは、表1の参照先をご覧ください。

TABLE 1. REFERENCES

Oracle Solaris Studio ドキュメント	http://www.oracle.com/technetwork/jp/server-storage/solarisstudio/documentation/index.html
アプリケーション・パフォーマンスを分析および向上する方法	http://www.oracle.com/technetwork/jp/articles/servers-storage-dev/o11-145-studio-analyze-perf-1413944-ja.html
OTNのOracle Solaris Studio に関する情報	http://www.oracle.com/technetwork/jp/server-storage/solarisstudio/overview/index.html
“Solaris Studio Topics: Debugging”	http://www.oracle.com/technetwork/systems/debugging-apps/index.html
DTrace information	http://www.oracle.com/technetwork/systems/dtrace/dtrace/index.html
パフォーマンス・ライブラリ：ユーザーズ・ガイド	http://docs.oracle.com/cd/E22054_01/index.html
Oracle Solaris Studio 12.3: パフォーマンスアナライザ	http://docs.oracle.com/cd/E27069_01/html/E26456/index.html
Oracle Solaris Studio 12.3: スレッドアナライザ・ユーザーズガイド	http://docs.oracle.com/cd/E27069_01/html/E26458/index.html
Oracle Solaris Studio 12.3: dbx コマンドによるデバッグ	http://docs.oracle.com/cd/E27069_01/html/E26440/index.html
Oracle Solaris Studio 12.3 IDE クイックスタートチュートリアル	http://docs.oracle.com/cd/E27069_01/html/E26460/index.html
Oracle Solaris Studio 12.3 dbxtool チュートリアル	http://docs.oracle.com/cd/E27069_01/html/E26462/index.html
Oracle Solaris Studio 12.3 DLight チュートリアル	http://docs.oracle.com/cd/E27069_01/html/E26464/index.html
Oracle Solaris Studio 12.3 Discover および Uncover ユーザーズ・ガイド	http://docs.oracle.com/cd/E27069_01/html/E26444/index.html
Oracle Solaris Studio 12.3 コード・アナライザ・ユーザーズ・ガイド	http://docs.oracle.com/cd/E27069_01/html/E26466/index.html
Oracle Solaris Studio 12.3 コードアナライザ・チュートリアル	http://docs.oracle.com/cd/E27069_01/html/E26468/index.html

Oracle Solaris Studioを使用した
エンタープライズ・アプリケーションの開発
2011年12月

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口：
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together