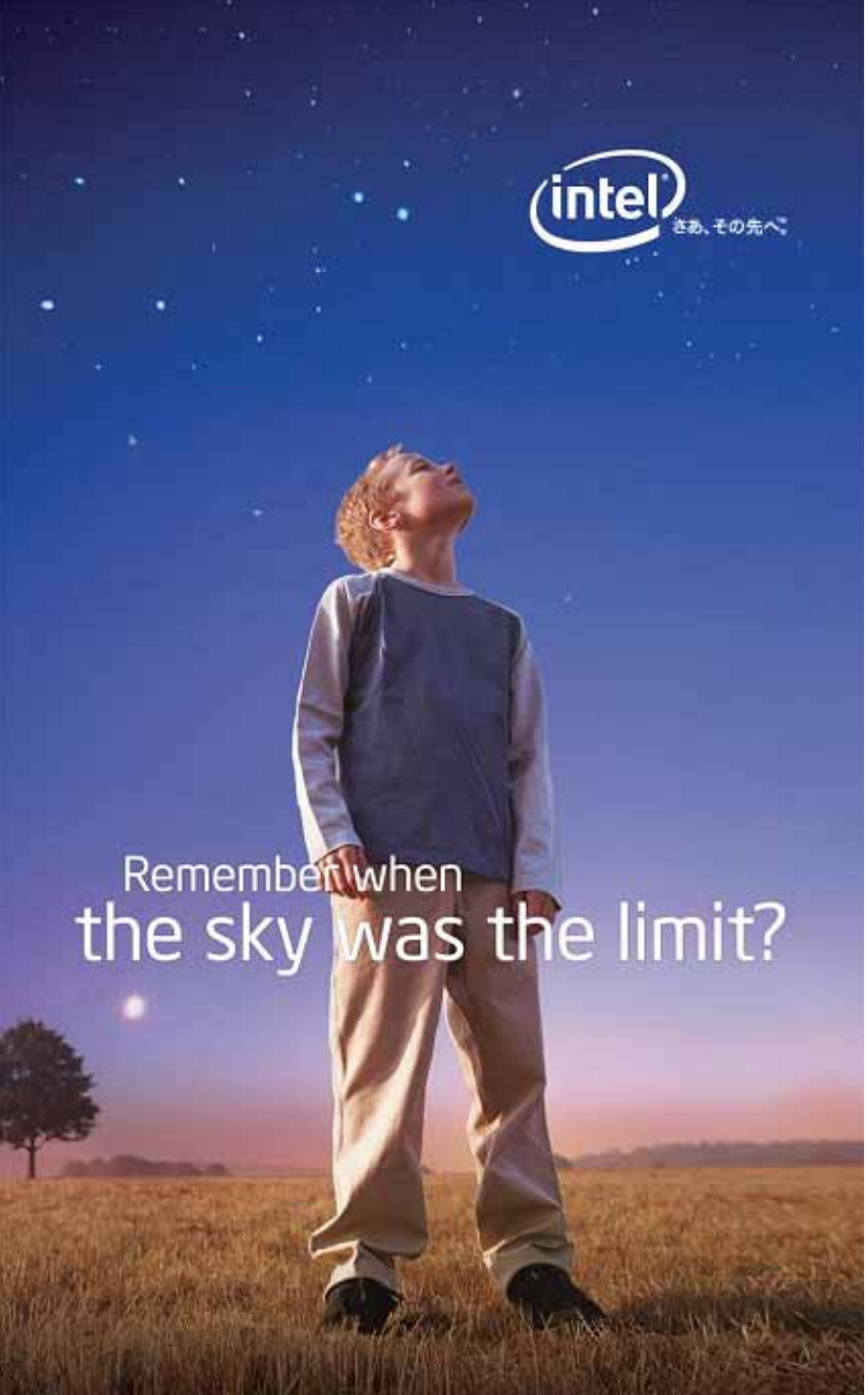


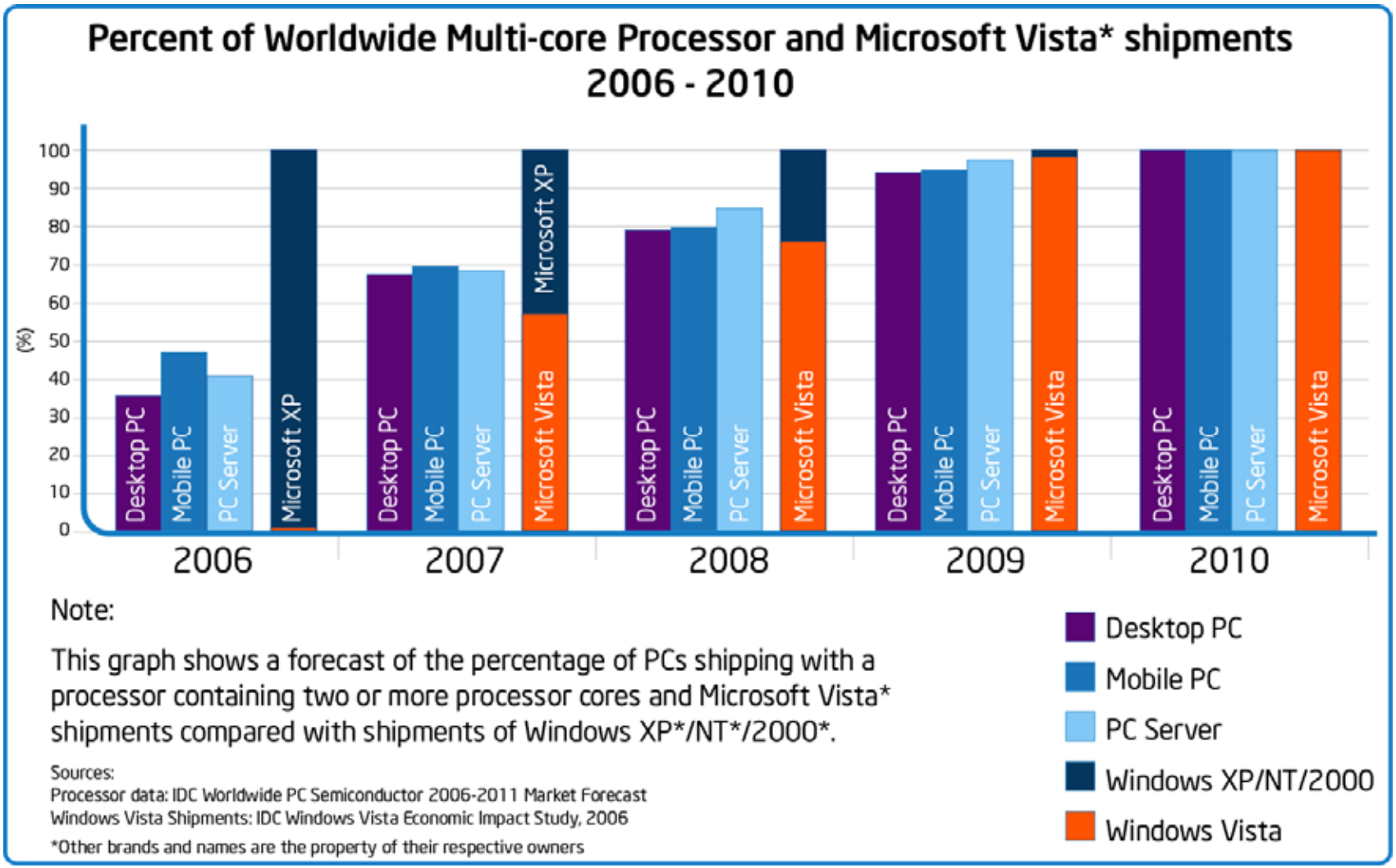
マルチコア・プロセッサーの能力を最大限に引き出すアプリケーション

インテル® ソフトウェア開発製品

製品の概要

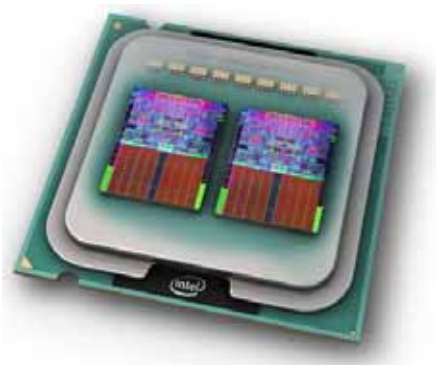
intel
Software
Products

世界的な動向と予測



インテル® Core™2 アーキテクチャー チューニングに最適

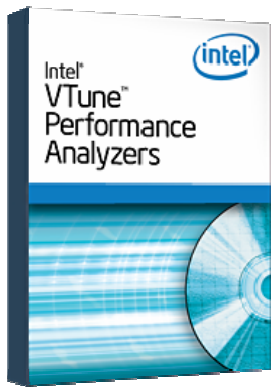
インテル® Core™2 Duo プロセッサー
2006/07/27 発表



インテル® Core™2 Quad プロセッサー
2006/11/14 発表

インテル® Core™2 プロセッサーの新機能

- ストール・サイクル・アカウンティング
- より優れたイベントで的確な分析
- precise イベントの追加



インテル® VTune™ パフォーマンス・アナライザーは、これらの新機能にアクセスし、より優れた解析を提供

「VTune™ アナライザーを使用したインテル® Core™2 プロセッサーのサイクルの要因分析 (サイクル・アカウンティング) は、1 年以内にチューニング方法論のスタンダードになるだろう。」

David Levinthal 氏
シニア・テクニカル・コンサルティング・エンジニア

より豊かなPC体験に向けて

インテル® Core™2 Extreme プロセッサー QX6800

- パワー・ユーザーおよびゲーム・ユーザーに先進の性能を提供するクアッドコア・プロセッサー
- 2.93GHz，1066MHz FSB，8MB L2キャッシュ
- 2007年4月10日発表

4月11日発表:
クアッドコア、Windows Vista*に対応する新しいインテルのツール群

Think Parallel !! **並列化の準備はできていますか？**

- インテル® VTune™ パフォーマンス・アナライザー 9.0 for Windows, Linux
- インテル® スレッド・プロファイラー 3.1 for Windows, Linux
- インテル® スレッディング・ビルディング・ブロック 1.1 for Windows, Linux & Mac OS
- インテル® スレッドチェッカー 3.1 for Windows , Linux

インテル® ソフトウェア開発製品で実現する並列化

インテル® コンパイラー
- インテル® プロセッサー上でアプリケーション性能向上を実現し、開発生産性を向上

インテル® VTune™ パフォーマンス・アナライザー
- パフォーマンス上のボトルネックをいち早く発見

インテル® パフォーマンス・ライブラリー
- 高度に最適化、スレッド化されたマルチメディア用および科学技術演算用のライブラリー

インテル® スレッド化ツール
- スレッディングのエラーを発見し、スレッド化アプリケーションを最適化して性能を最大化

インテル® スレッディング・ビルディング・ブロック
- マルチスレッド・アプリケーション開発でパフォーマンス向上とスケーラビリティーを簡素化する C++ テンプレート・ベースのランタイム・ライブラリー

インテル® クラスターツール
- クラスターベースのアプリケーションの作成、分析、最適化、そして実装を支援

クロスプラットフォームをサポート

Intel® Software Development Products

Currently Available ●

		Itanium 2	Core 2 Duo / Centrino	Pentium 4 / Xeon / V/V				
		Operating systems		Operating Systems				
		Windows*	Linux*	Windows	Linux	Mac OS* X		
		Development Environments		Development Environments				
		Visual Studio*	GCC*	Visual Studio	GCC	XCode*		
Compilers	C++	●	●	●	●	●		
	Fortran	●	●	●	●	●		
Performance Analyzers	VTune™ Performance Analyzer	●	●	●	●			
Performance Libraries	Integrated Performance Primitives	●	●	●	●	●		
	Math Kernel Library	●	●	●	●	●		
	Threading Building Blocks		●	●	●	●		
Threading Analysis Tools	Thread Checker			●	●			
	Thread Profiler			●				
Cluster Tools	MPI Library		●		●			
	Trace Analyzer and Collector		●		●			
	Math Kernel Library Cluster Edition	●	●	●	●			
	Cluster Toolkit		●		●			

サーバーから携帯 / ワイヤレス機器まで、さまざまな
インテル® プラットフォーム向けのアプリケーション開発をサポート

インテル® C++ コンパイラーおよび Fortran コンパイラー

ソフトウェアをトップスピードで実行

マルチコア・プロセッサーのサポート

- **自動並列化、および OpenMP* をサポート**

OS 固有のサポート

- Windows*
  - Microsoft* Visual Studio* **との統合互換**
  - Microsoft* Visual C++* **とのソースおよびオブジェクト・コード互換**
- Linux*
  - GCC (C++ Linux) **とのコマンドライン互換**
  - GCC 4.0 **とのソースコードおよびバイナリー互換**
  - Eclipse 3.1.1/CDT 3.0 **との統合をサポート**
- Mac OS* X
  - GCC **とのコマンドライン互換**
  - XCode **開発環境に統合**

サポートするインテル® プロセッサー

- 32 **ビット・プロセッサー、インテル®** 64 **対応、およびインテル®** Itanium® 2 **プロセッサー・ファミリー**
- **ストリーミング** SIMD **拡張命令をサポート** (SSE2 **と** SSE3)

AMD Opteron* **や** Athlon* **などの** AMD **プロセッサーをサポート**

インテル® コードカバレッジとインテル® テスト優先化ツールを同梱

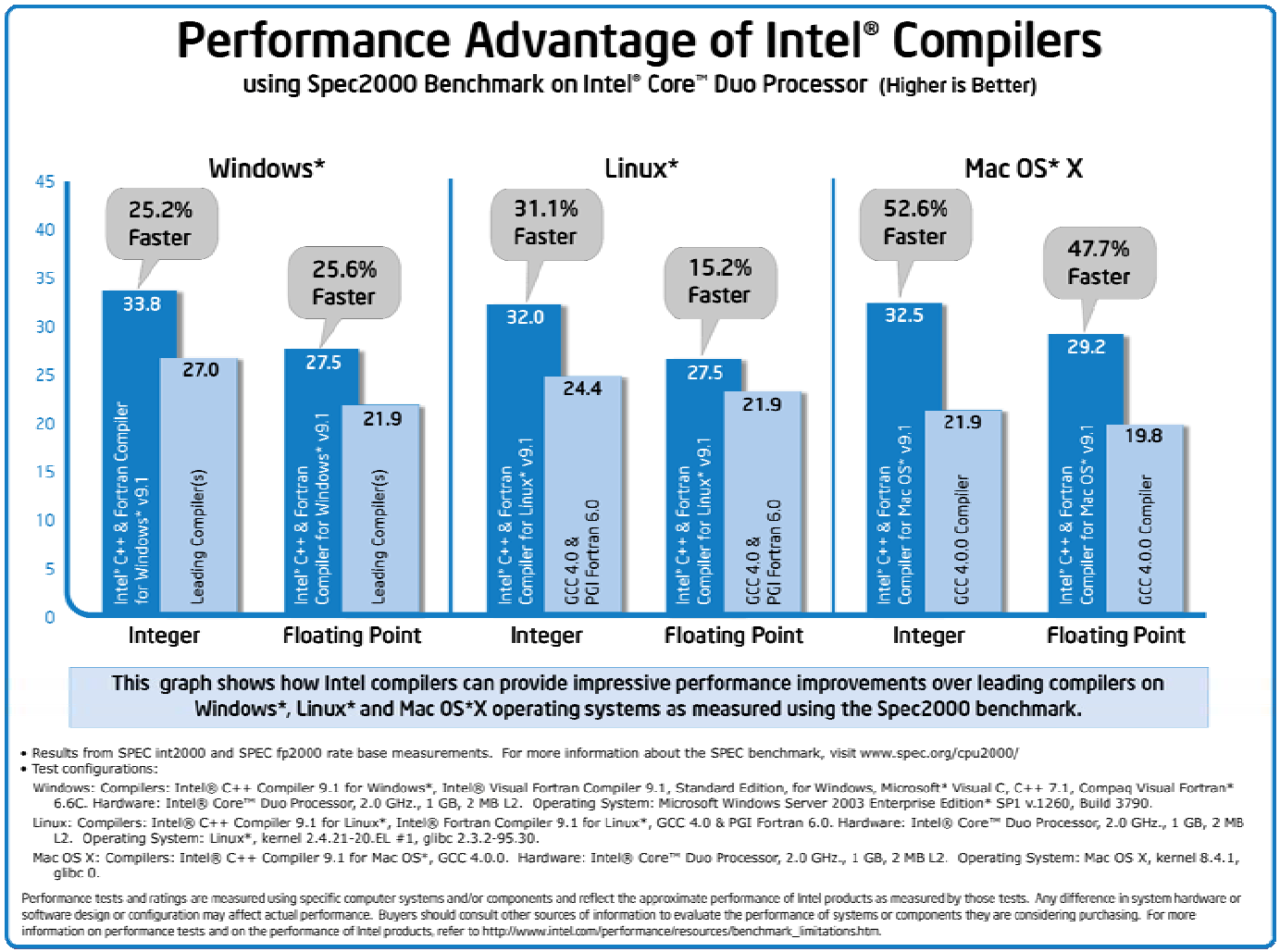
サポート	Windows	Linux	Mac	64 ビット	マルチコア	AMD*
インテル® C++ コンパイラー	√	√	√	√	√	√
インテル® Fortran コンパイラー	√	√	√	√	√	√

「インテル® C++ コンパイラー Linux* 版を導入して、当社の Computational Fluid Dynamics (CFD) ソフトウェアに対して標準ベンチマークを実施したところ、GCC コンパイラーを使用した場合と比べ、パフォーマンスが 9% から 37% に著しく向上しました。しかも、インテル® C++ コンパイラー Linux 版は、技術的な問題もなくスムーズに、当社の開発環境に導入できました。」

Dipankar Choudhury 博士
CTOFluent Inc.

intel

SPEC* CPU2000 V1.2、インテル® Core™ Duo プロセッサー Windows*、Linux*、Mac OS* X



- Results from SPEC int2000 and SPEC fp2000 rate base measurements. For more information about the SPEC benchmark, visit www.spec.org/cpu2000/
- Test configurations:

Windows: Compilers: Intel® C++ Compiler 9.1 for Windows*, Intel® Visual Fortran Compiler 9.1, Standard Edition, for Windows, Microsoft* Visual C, C++ 7.1, Compaq Visual Fortran* 6.6C. Hardware: Intel® Core™ Duo Processor, 2.0 GHz., 1 GB, 2 MB L2. Operating System: Microsoft Windows Server 2003 Enterprise Edition* SP1 v.1260, Build 3790.

Linux: Compilers: Intel® C++ Compiler 9.1 for Linux*, Intel® Fortran Compiler 9.1 for Linux*, GCC 4.0 & PGI Fortran 6.0. Hardware: Intel® Core™ Duo Processor, 2.0 GHz., 1 GB, 2 MB L2. Operating System: Linux*, kernel 2.4.21-20.EL #1, glibc 2.3.2-95.30.

Mac OS X: Compilers: Intel® C++ Compiler 9.1 for Mac OS*, GCC 4.0.0. Hardware: Intel® Core™ Duo Processor, 2.0 GHz., 1 GB, 2 MB L2. Operating System: Mac OS X, kernel 8.4.1, glibc 0.

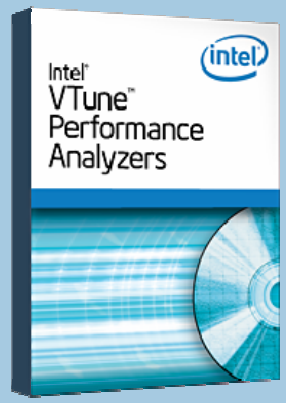
Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to http://www.intel.com/performance/resources/benchmark_limitations.htm.

インテル® VTune™ パフォーマンス・アナライザー 9.0

発見が困難なパフォーマンスのボトルネックを識別

「インテル® VTune™ パフォーマンス・アナライザーを使えば、これまで何日もかかっていた作業を 1 日以内に完了することができます。」

Randy Camp 氏
ソフトウェア R&D 担当副社長
MUSICMATCH Inc.

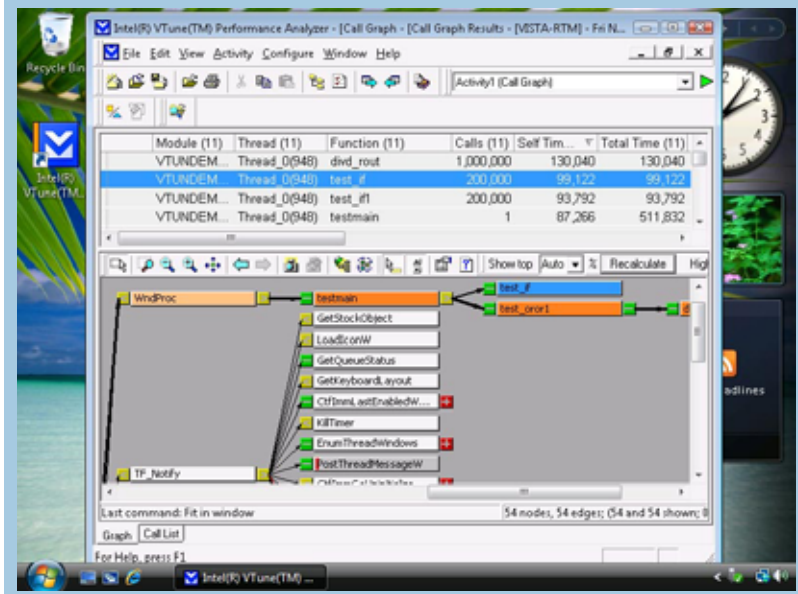


機能

- プロセスまたはスレッド並列コードのチューニング
- オーバーヘッドの少ないサンプリング
- グラフィカルなコールグラフ
- ソースまたはアセンブリーで結果を表示

新機能

- 新しいチューニング方法論
  - Core™2 Duo プロセッサーと Core™2 Quad プロセッサーのストール・サイクル・アカウンティング
- Windows: Microsoft Vista* 対応
- Linux: インテル® コンパイラーによる分析と直感的な hotspot ナビゲーター



Windows*	Linux*	Mac*	IA32	Intel64	IA64	マルチコア
√	√		√	√	√	√

インテル® マス・カーネル・ライブラリー 9.0

高度に最適化されスレッドセーフな関数群

マルチコア対応

スレッドセーフ

マルチプロセッサー・システムで優れたスケーリングを発揮

実行時に自動的にプロセッサーを識別

C および Fortran からの呼び出しをサポート

1つのパッケージですべてのインテル® プロセッサーに対応

ロイヤリティー無料の再配布権

- ✓ BLAS
- ✓ LAPACK
- ✓ ScaLAPACK
- ✓ スパースソルバー
- ✓ 高速フーリエ変換
- ✓ ベクトル演算

サポート	Windows*	Linux*	Mac OS*	64 ビット	マルチコア	AMD*
インテル® MKL	√	√	√	√	√	√

「インテル® マス・カーネル・ライブラリーの DGEMM ライブラリーを導入したところ、当社の標準ベンチマーク結果が 43％～71％ も高速化しました。これは実に驚くべき結果です。」

Matt Dunbar 氏
ソフトウェア・デベロッパー
ABAQUS, Inc

インテル® インテグレーテッド・パフォーマンス・プリミティブ

マルチコア機能向けに高度に最適化された関数

アプリケーションのソースコード

インテル® IPP のコードサンプル
- ビデオ/オーディオ/スピーチ・コーデックのサンプル
- イメージ処理と JPEG
- 信号処理
- データ圧縮
- .NET および Java* の統合

→ **迅速なアプリケーション開発**

API 呼び出し

インテル® IPP ライブラリー C/C++ API
- ビデオ・コーディング
- オーディオ・コーディング
- 音声コーディング
- 音声認識
- データ圧縮
- 暗号化
- マトリクス演算
- 信号処理
- イメージ処理
- JPEG コーディング
- コンピューター・ビジョン
- イメージカラー変換
- 文字列処理
- ベクター演算

→ **クロスプラットフォーム互換とコードの再利用**

スタティック・リンク/ダイナミック・リンク

インテル® IPP のプロセッサーに最適化されたバイナリー
- インテル® Core™ Duo および Core™ Solo プロセッサー
- インテル® Pentium® D デュアルコア・プロセッサー
- インテル® Pentium® M プロセッサー
- インテル® Pentium® 4 プロセッサー
- インテル® Xeon® プロセッサー
- インテル® Itanium® 2 プロセッサー
- Intel XScale® テクノロジーベース・プロセッサー

→ **卓越したパフォーマンス**

サポート	Windows*	Linux*	Mac*	64 ビット	マルチコア	AMD*
インテル® IPP	√	√	√	√	√	√

「良質で高解像度のビデオ映像を個人のデスクで視聴したり、または移動中に鑑賞できる機能が求められています。このような要望に応えるアプリケーションを提供しています。このアプリケーションの開発には、インテル® IPP とインテル® C++ コンパイラーがその最適化において重要な役割を果たしています。」

Bruce Rady 氏
社長
RadTIME, Inc.

インテル® スレッドチェッカー 3.1

スレッド化のエラーを正確にピンポイント

機能

- 発見の困難な競合状態やデッドロックを検出
- ソースコード行のエラーの正確な位置を指摘
- 回帰テスト実行用にバッチスクリプトを統合
- Windows と Linux のコマンドライン・インターフェイス
- 再コンパイルせずに標準的なデバッグビルドで動作

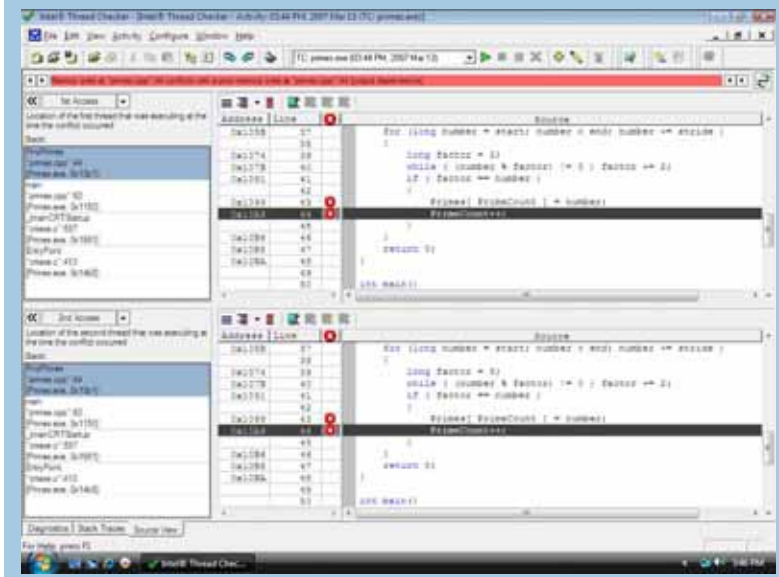
新機能

- パフォーマンスの最適化による迅速な分析
- Microsoft* Vista* 対応

Windows*	Linux*	Mac*	IA32	Intel64	IA64	マルチコア
√	√		√	√		√

「スレッドチェッカーがなければ、これほど早く、効率的にネットワークをセットアップして実行させることはできなかったでしょう。スレッドチェッカーは本当にすばらしいツールで、この製品なしにはマルチスレッド・コードの開発は考えられません。」

Doug Service 氏、テクノロジー開発担当ディレクター
Chris Stark 氏、ソフトウェア・エンジニア
Ritual Entertainment



インテル® スレッド・プロファイラー 3.1
スレッド化の非効率な場所をピンポイント

機能

- 並列レベルでアプリケーションを表示し、コアが十分に活用されているかを確認
- スレッド関連のオーバーヘッドが与えているパフォーマンスの影響を特定
- アクティブまたはインアクティブなスレッドを識別
- VTune™ アナライザー Windows* 版に同梱
- スレッディング・ビルディング・ブロック API 対応

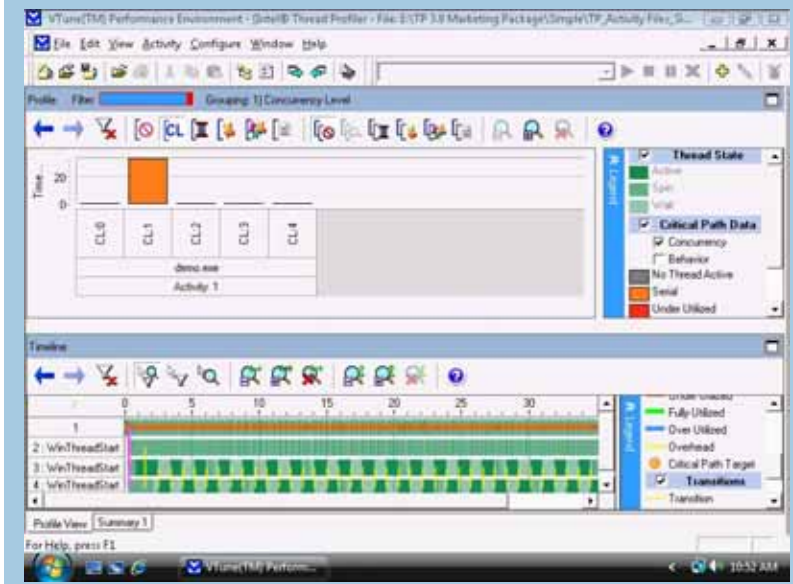
新機能

- 使用方法は簡単 - カスタム設定を呼び出すだけ
- すぐに使用可能 - ユーザーが選択できるスタック・ウォーキング
- Microsoft* Vista* 対応

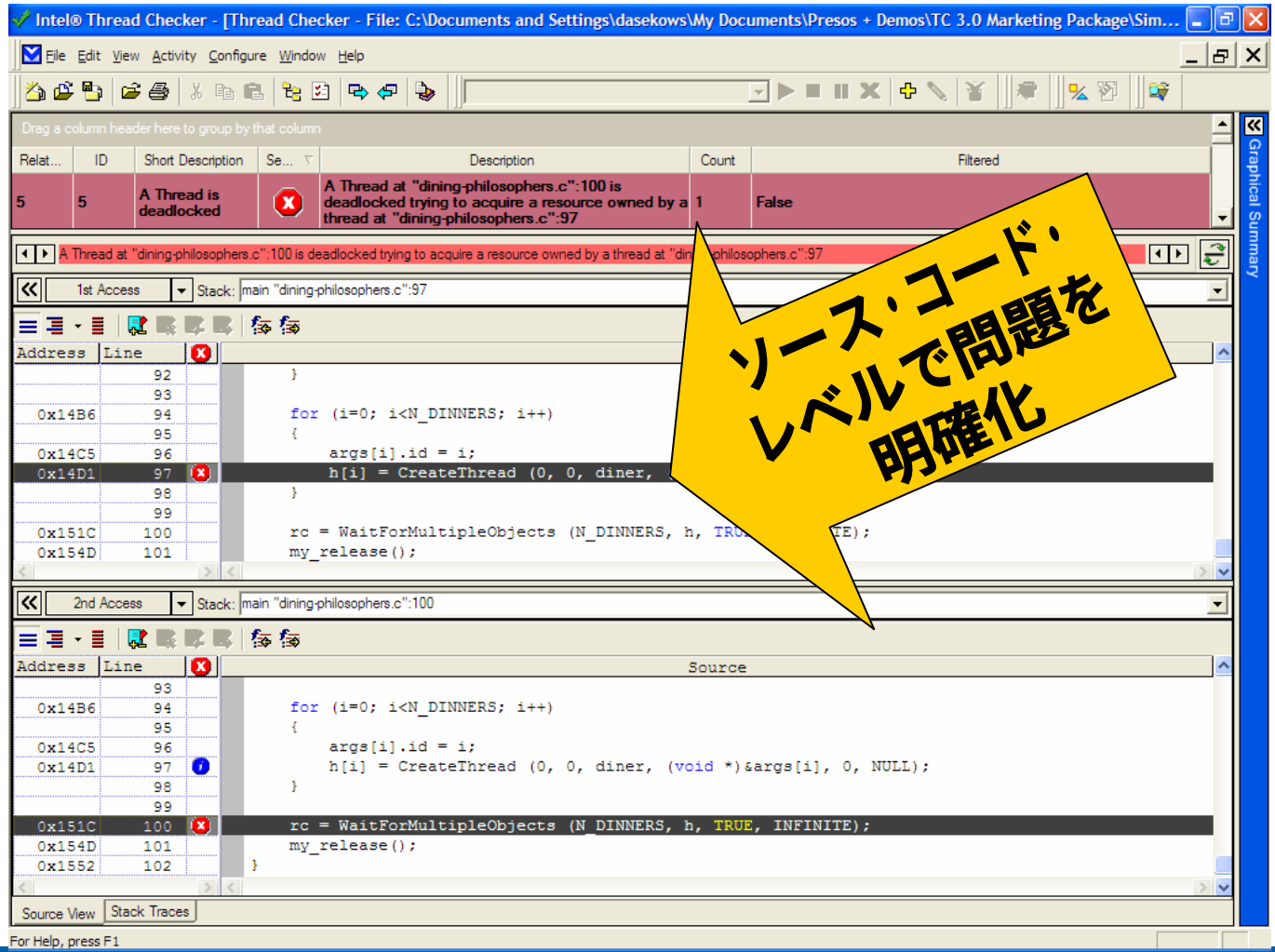
Windows*	Linux*	Mac*	IA32	Intel64	IA64	マルチコア
√	√		√	√		√

「インテル® スレッド・プロファイラーは、スレッド化コードに存在するボトルネックの分析に非常に役立ちました。インテル® スレッド・プロファイラーでは、問題のある部分が素早くピンポイントで検出され、速度低下の理由が表示されるため、コードを再構築してスレッド・パフォーマンスを向上させることができました。」

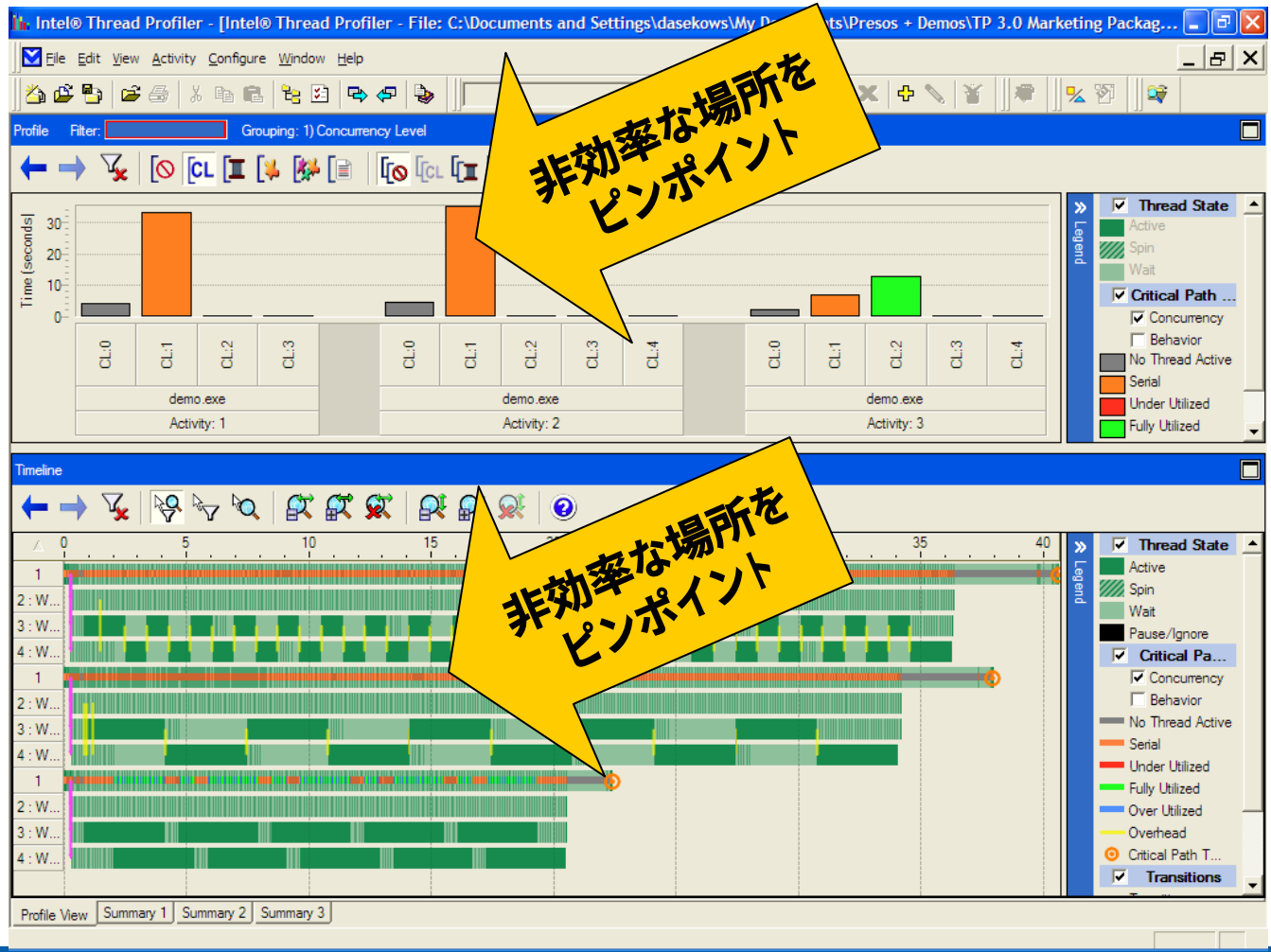
Martin Watt 氏
ソフトウェア・アーキテクト
Alias



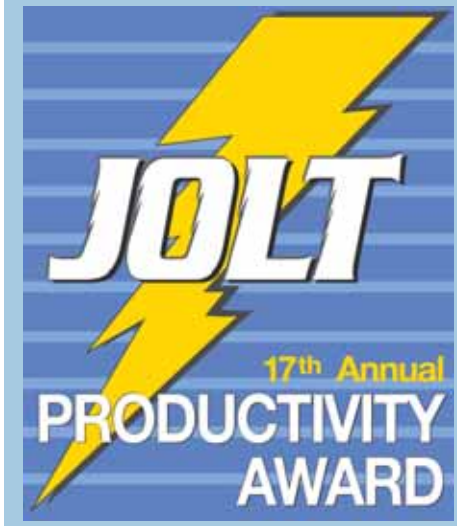
インテル® スレッドチェッカー Windows* 版
スレッドの問題をピンポイントで特定

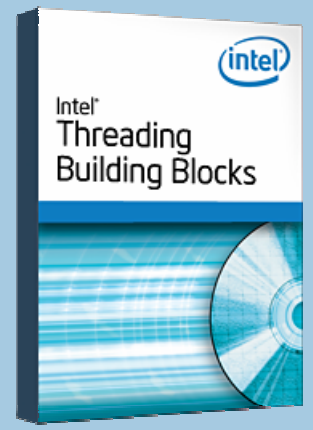


インテル® スレッド・プロファイラー
非効率なスレッディング問題をピンポイント



インテル® スレッディング ビルディング・ブロック 1.1
C++ で並列化





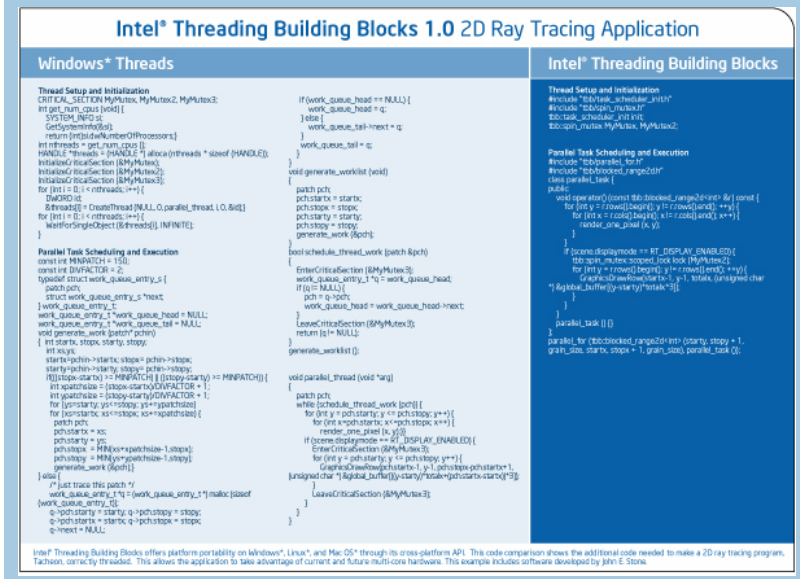
機能

- スレッドではなく、馴染みのあるタスク処理を使用する C++ ランタイム・ライブラリー
- ハイレベルな抽象化により、パフォーマンスを損なうことなくわずかなコードの追加でスレッド化
- 利用できるコア数に応じて適切にパフォーマンスを向上
- スレッド・ライブラリー API は Linux、Windows、Mac OS 間でのプラットフォーム間のポータビリティーを提供
- すべての C++ コンパイラーで動作 (例: Microsoft、GNU、インテル)

新機能

- より効率的な並列アルゴリズムのための自動粒度計算
- Microsoft* Vista* 対応
- Mac OS X* のネイティブ 64 ビットのフルサポート

Windows*	Linux*	Mac*	IA32	Intel64	IA64	マルチコア
√	√	√	√	√	√	√



等価な Windows スレッド機能と並べて比較すると、インテル® スレッディング・ビルディング・ブロックがスレッド化の実装を大幅に単純化していることがわかる

わずかなコードの追加で並列化を実現
例: 2 次元レイ・トレーシング・アプリケーション

Windows スレッド

スレッドのセットアップと初期化

```
CRITICAL_SECTION MyMutex, MyMutex2, MyMutex3;
int get_num_cpus (void) {
    SYSTEM_INFO si;
    GetSystemInfo(&si);
    return (int)si.dwNumberOfProcessors;}
int nthreads = get_num_cpus ();
HANDLE *threads = (HANDLE *) alloca (nthreads * sizeof (HANDLE));
InitializeCriticalSection (&MyMutex);
InitializeCriticalSection (&MyMutex2);
InitializeCriticalSection (&MyMutex3);
for (int i = 0; i < nthreads; i++) {
    DWORD id;
    &threads[i] = CreateThread (NULL, 0, parallel_thread, i, 0, &id);}
for (int i = 0; i < nthreads; i++) {
    WaitForSingleObject (&threads[i], INFINITE);
}
```

並列タスクのスケジューリングと実行

```
const int MINPATCH = 150;
const int DIVFACTOR = 2;
typedef struct work_queue_entry_s {
    patch pch;
    struct work_queue_entry_s *next;
} work_queue_entry_t;
work_queue_entry_t *work_queue_head = NULL;
work_queue_entry_t *work_queue_tail = NULL;
void generate_work (patch* pchin)
{  int startx, stopx, starty, stopy;
    int xs,ys;
    startx=pchin->startx; stopx= pchin->stopx;
    starty=pchin->starty; stopy= pchin->stopy;
    if(((stopx-startx) >= MINPATCH) || ((stopy-starty) >= MINPATCH)) {
      int xpatchsize = (stopx-startx)/DIVFACTOR + 1;
      int ypatchsize = (stopy-starty)/DIVFACTOR + 1;
      for (ys=starty; ys<=stopy; ys+=ypatchsize)
      for (xs=startx; xs<=stopx; xs+=xpatchsize) {
        patch pch;
        pch.startx = xs;
        pch.starty = ys;
        pch.stopx  = MIN(xs+xpatchsize-1,stopx);
        pch.stopy  = MIN(ys+ypatchsize-1,stopy);
        generate_work (&pch);}
} else {
      /* just trace this patch */
      work_queue_entry_t *q = (work_queue_entry_t *) malloc (sizeof
(work_queue_entry_t));
      q->pch.starty = starty; q->pch.stopy = stopy;
      q->pch.startx = startx; q->pch.stopx = stopx;
      q->next = NULL;
      if (work_queue_head == NULL) {
        work_queue_head = q;
      } else {
        work_queue_tail->next = q;
      }
      work_queue_tail = q;
    }
}
void generate_worklist (void)
{
    patch pch;
    pch.startx = startx;
    pch.stopx = stopx;
    pch.starty = starty;
    pch.stopy = stopy;
    generate_work (&pch);
}
bool schedule_thread_work (patch &pch)
{
    EnterCriticalSection (&MyMutex3);
    work_queue_entry_t *q = work_queue_head;
    if (q != NULL) {
      pch = q->pch;
      work_queue_head = work_queue_head->next;
    }
    LeaveCriticalSection (&MyMutex3);
    return (q != NULL);
}
generate_worklist ();


void parallel_thread (void *arg)
{
    patch pch;
    while (schedule_thread_work (pch)) {
      for (int y = pch.starty; y <= pch.stopy; y++) {
        for (int x=pch.startx; x<=pch.stopx; x++) {
          render_one_pixel (x, y);}}
      if (scene.displaymode == RT_DISPLAY_ENABLED) {
        EnterCriticalSection (&MyMutex3);
        for (int y = pch.starty; y <= pch.stopy; y++) {
          GraphicsDrawRow(pch.startx-1, y-1, pch.stopx-pch.startx+1,
(unsigned char *) &global_buffer[((y-starty)*totalx+(pch.startx-startx))*3]);
        }
        LeaveCriticalSection (&MyMutex3);
      }
    }
}
```

この例は、John E. Stone 氏によって開発されたソフトウェアを使用しています。

インテル® スレッディング・ビルディング・ブロック

スレッドのセットアップと初期化

```
#include "tbb/task_scheduler_init.h"
#include "tbb/spin_mutex.h"
tbb::task_scheduler_init init;
tbb::spin_mutex MyMutex, MyMutex2;
```

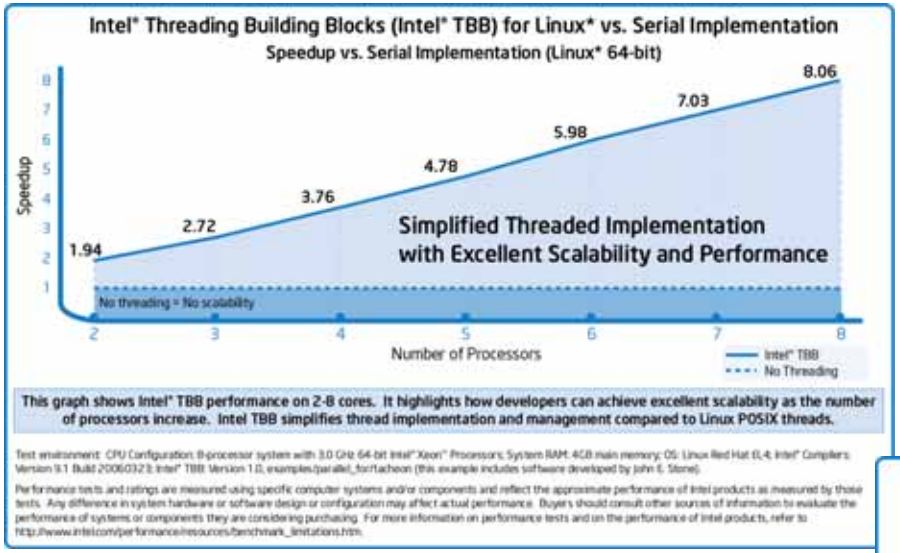
並列タスクのスケジューリングと実行

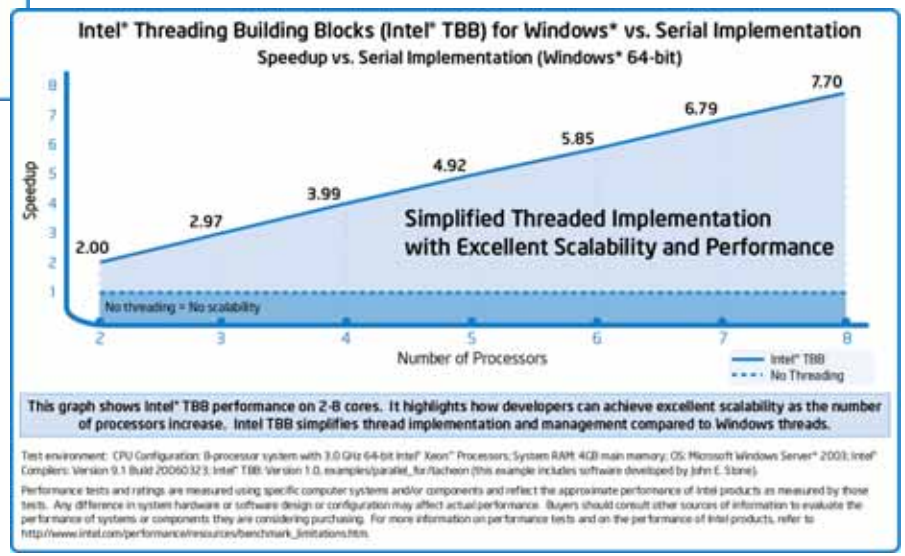
```
#include "tbb/parallel_for.h"
#include "tbb/blocked_range2d.h"
class parallel_task {
public:
    void operator() (const tbb::blocked_range2d<int> &r) const {
      for (int y = r.rows().begin(); y != r.rows().end(); ++y) {
        for (int x = r.cols().begin(); x != r.cols().end(); x++) {
          render_one_pixel (x, y);
        }
      }
      if (scene.displaymode == RT_DISPLAY_ENABLED) {
        tbb::spin_mutex::scoped_lock lock (MyMutex2);
        for (int y = r.rows().begin(); y != r.rows().end(); ++y) {
          GraphicsDrawRow(startx-1, y-1, totalx, (unsigned char
*) &global_buffer[(y-starty)*totalx*3]);
        }
      }
    }
    parallel_task () {}
};
parallel_for (tbb::blocked_range2d<int> (starty, stopy + 1,
grain_size, startx, stopx + 1, grain_size), parallel_task ());
```

スレッドをどう制御するかに注目するのではなく、行うべき処理に注目

インテル® スレッディング・ビルディング・ブロックは、クロスプラットフォームで利用可能な共通 API によって Windows*、Linux* および Mac OS* X でのプラットフォーム間でのポータビリティーを提示します。このコード比較は、2D レイ・トレーシング・プログラム (Tacheon) に正しくスレッド化するために必要とされる追加コードを示します。これにより、アプリケーションが現在と将来のマルチコア・ハードウェアを利用することを可能にします。

ネイティブスレッドを使用した場合とのスケーリング比
ベンチマーク: 2 次元レイ・トレーシング・アプリケーション





インテル® クラスターツール

クラスター・アプリケーションの作成、デバッグ、最適化

クラスター・アプリケーションの開発とパフォーマンスの向上を支援

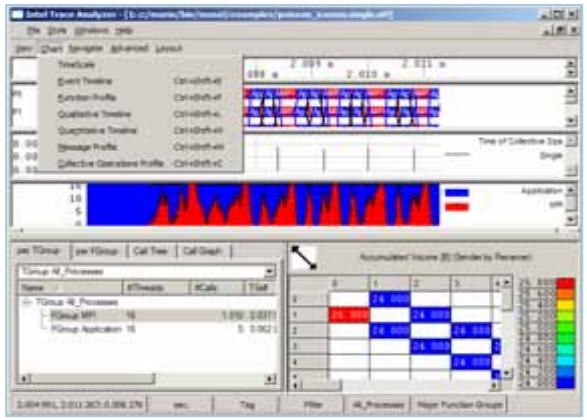
- 並列アプリケーションの作成、分析、最適化、そして実装を支援
- ネットワークの種類に依存しない MPI ライブラリー
- マルチコア・クラスター対応

インテル® クラスター・ツールキット、完全な MPI ツール環境

- インテル® MPI ライブラリー
- インテル® トレース・アナライザー/コレクター
- インテル® MKL クラスター・エディション
- インテル® MPI ベンチマーク

インテル® コンパイラーのアドオンされたクラスター OpenMP*

- メモリー分散型の OpenMP* であり、クラスター OpenMP* と呼ばれる。64 ビット インテル® プロセッサー版が入手可能（Itanium® プロセッサーとインテル® 64 対応プロセッサー）



サポート	Windows*	Linux*	Mac*	64 ビット	マルチコア	AMD*
インテル® MPI ライブラリー		√		√	√	√
インテル® MKL クラスター・エディション	√	√		√	√	√
インテル® トレース・アナライザー/コレクター		√		√	√	
インテル® クラスター・ツールキット		√		√	√	
クラスター OpenMP* アドオン		√		√	√	√

「メッセージの統計表示機能は、通信を行っているプロセッサーがグリッド上に表示されて全体を概観でき、特に役立つ機能です。」

Dominic Holland 氏
SDSC

インテル® MPI ライブラリー 3.0

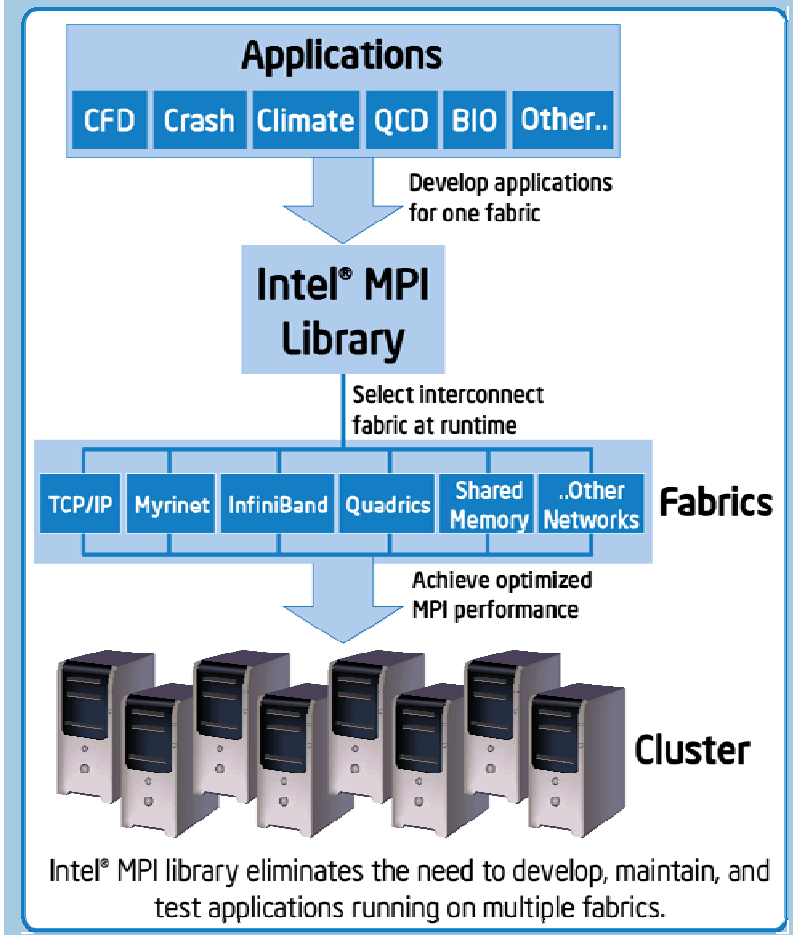
複数のネットワーク・ファブリックで動作するアプリケーションを実現するユニバーサルな MPI ソリューション

機能

- 簡単なインストールと設定
- 開発リソースを節約し、アプリケーションの品質を向上
- ジョブ・スケジューラーのサポート: PBS Pro*、Torque*、LSF* など
- デバッガーのサポート: IDB、DDT*、gdb、TotalView*
- 幅広く使用されている ANL MPICH2 がベース

新機能

- ファブリック自動選択
- 強化されたプロセスピンニング
- パフォーマンスの最適化とチューニング・オプション
- スレッドのフルサポート (MPI_THREAD_MULTIPLE)



「インテルの MPI とクラスターツールによって最良のクラスター開発環境が得られました。」

古石 貴裕 博士
戎崎計算宇宙物理研究室
独立行政法人 理化学研究所

インテル® トレース・アナライザー / コレクター 7.0

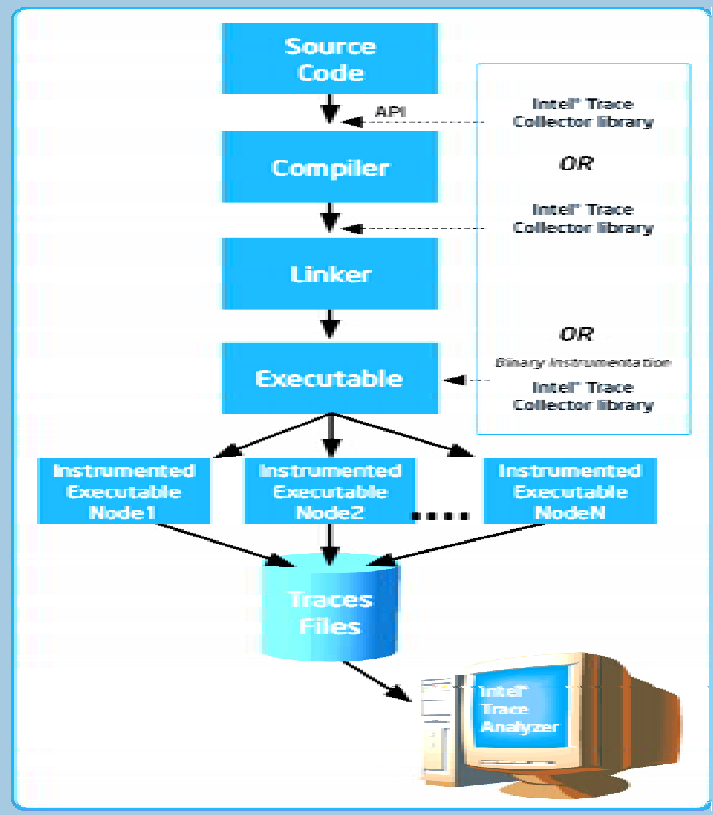
MPI アプリケーションの最高水準の解析ツール

機能

- 生産性とクラスター・アプリケーションの性能の向上
- きわめて少ない影響
- 時間とプロセッサーにおける優れたスケーラビリティー
- Linux* および Windows* 上の GUI

新機能

- 複数のトレースファイルの比較
- パフォーマンス・カウンターのタイムライン表示
- 新しいマクロレベルの表示と関数のフィルタリング
- GUI の改善と高速化
- MPI 検証 - ライブラリーの正当性の検証



インテル® トレース・アナライザー / コレクターは、FEKO 並列通信パターン、さらにメッセージ・パッシングの最適化に非常に役立ちました。その結果、ISV 電磁気クラスター・アプリケーションで優れたパフォーマンスを発揮させることができました。」

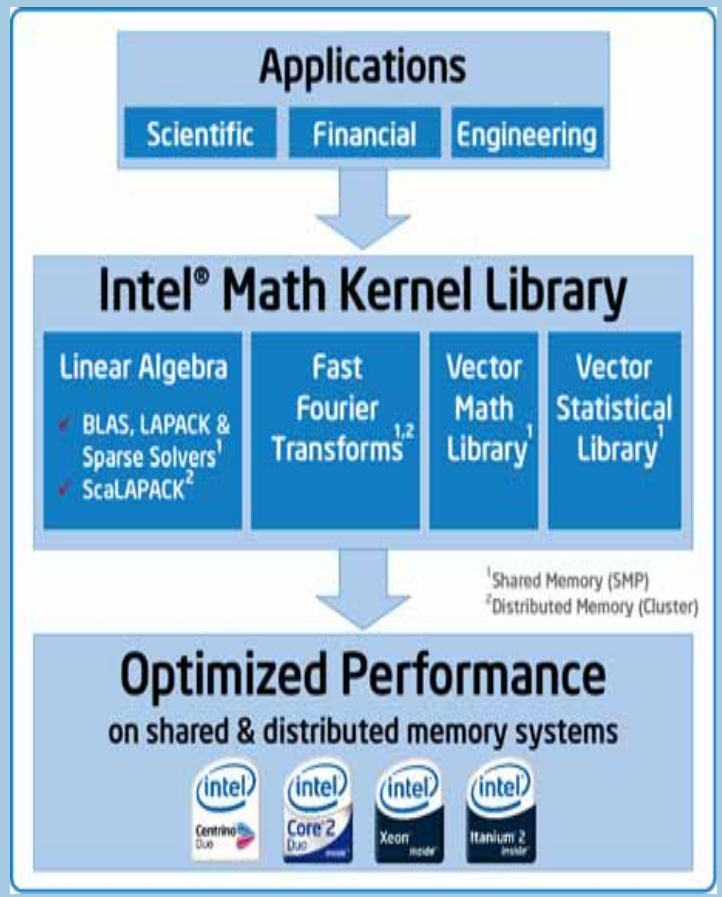
Ing. Ulrich Jakobus 博士
テクニカル・ディレクター

インテル® マス・カーネル・ライブラリー クラスター・エディション 9.0

パフォーマンスを最大限に引き出す高度に最適化された演算ライブラリー

新機能

- 新しいマルチコア・プロセッサー、インテル® Xeon® プロセッサー 5100 番台および 5300 番台向けに最適化
- 新しい VML 関数
  - floor、ceil、round、trunc、hypot など。
- 新しい FMGRES 反復法スパースソルバー
- Fortran と C の FFTW インターフェイス
- 新しい部分微分方程式ソルバー
  - Helmholtz 方程式、Poisson 方程式、Laplace 方程式
- 新しいユーザーズ・ガイドと Linux の man ページ



「インテル® マス・カーネル・ライブラリーで特筆すべきは、乱数生成におけるブロック分割オプションです。これは並列アプリケーションで非常に役立ちます。」

Mike Giles 氏
オックスフォード大学教授

インテル® プレミアサポート

インテルの専門家が直接提供するテクニカルサポート

Intel® Premier Support

インテル® ソフトウェア開発製品には、
1 年間の無制限のプレミアサポートが付属

インテル® プレミアサポート:

- **インテル® ソフトウェア開発製品の全製品のサポート**
- **インテル® プレミアサポート Web サイトのオンライン アクセス**
- **問題の送信と状況の確認**
- **製品のアップデートと関連ダウンロード**
- **FAQ、ユーザーフォーラム、その他の情報**

「サポートの登録は簡単でした。必要なときにインテルのサポートが受けられるという安心感は貴重です。」

Rob Hoffmann 氏
マーケティング・ディレクター
NewTek, Inc

マルチスレッド・アプリケーションの開発に包括的で業界をリードするソリューション

システム上で実行されている
アプリケーションの状態を
視覚化

解析

高度に最適化されたコンパ
イラーでスケーラブルな
ソリューションを実現

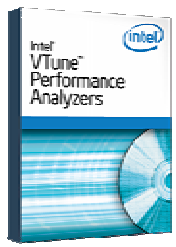
導入

マルチスレッド特有の
問題点を検出

デバッグ

パフォーマンスとスケーラ
ビリティーに基づいて
チューニング

チューニング



MPI の正当性の検証ツール

まとめと次のステップ

インテル® ソフトウェア開発製品
アプリケーションの並列化に必須な製品

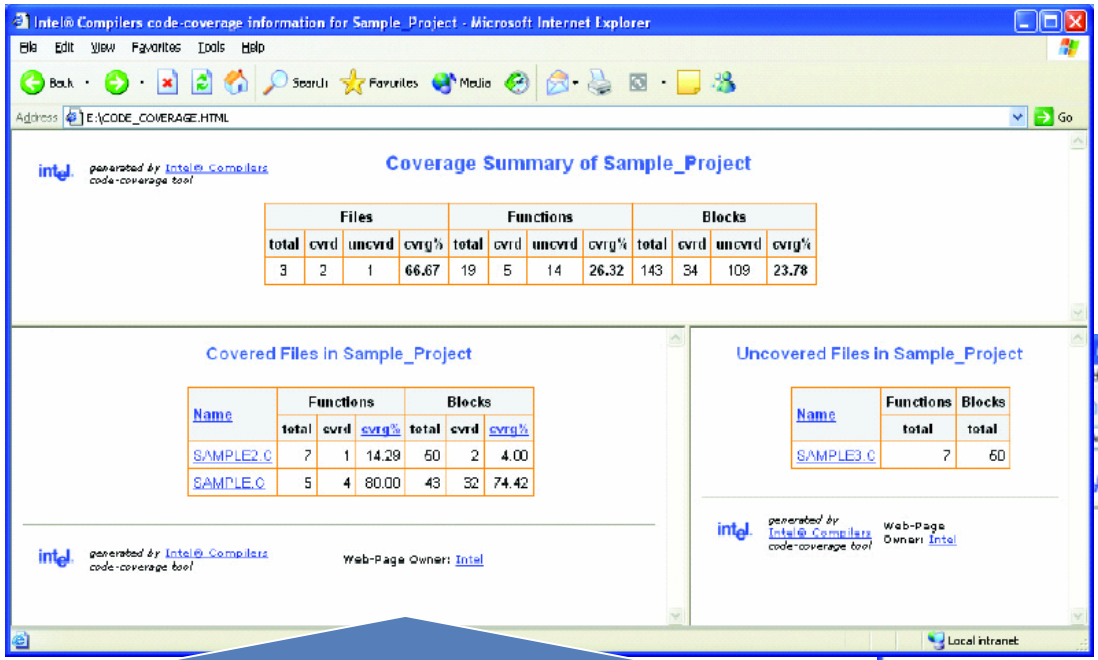
- マルチスレッド化および MPI クラスターを含む並列プログラムの分析、導入、デバッグ、チューニングをサポート
- 既存のビルドプロセスをサポート
- ソースおよびバイナリー互換を提供
- クロスプラットフォーム（プロセッサー、OS ）をサポート

次のステップ:

是非製品をご評価ください。
www.intel.co.jp/jp/software/products/

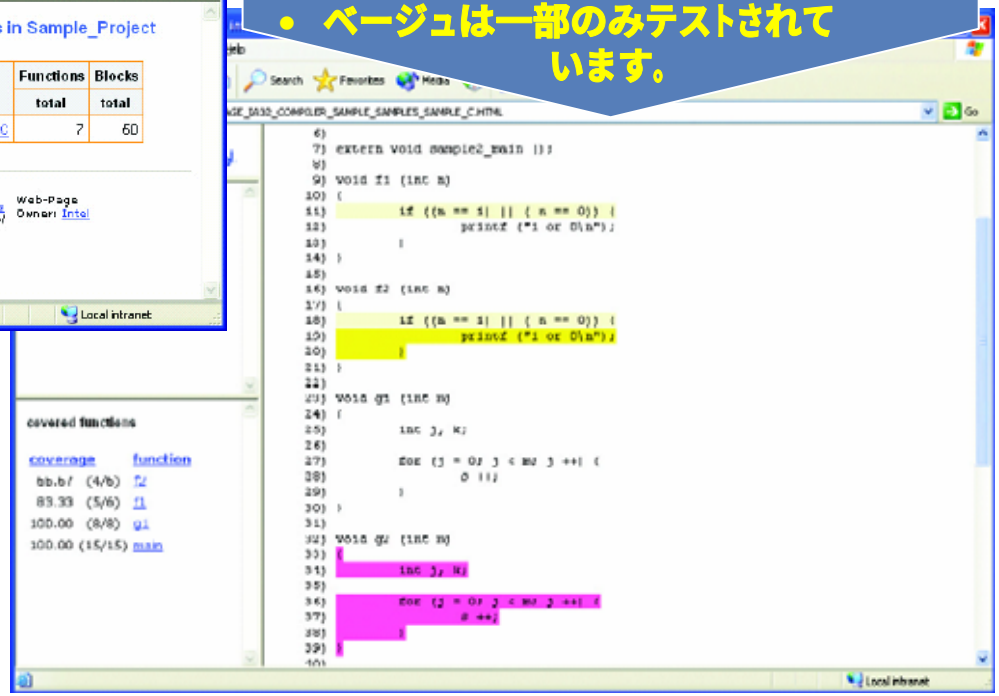
予備

インテル® コード・カバレッジ・ツール



SAMPLE.C をクリックすると、テストされたコードがハイライトされて表示されます。

- この例では、ピンクでハイライトされたコードは一度も実行されず、
- 黄色でハイライトされたコードは実行されますが、開発者によってセットアップされたテストではテストされず、
- ベージュは一部のみテストされています。



プロジェクトのコード・カバレッジ・サマリーの例です。このテストでは、3 つのうち 2 つのモジュール、19 のうち 5 つの関数、143 のうち 34 のブロックがテストされたことを示しています。ファイル SAMPLE.C では、5 つのうち 4 つの関数がテストされています。

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。
製品に付属の売買契約書『Intel's Terms and conditions of Sales』に規定されている場合を除き、インテルはいかなる責を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他、知的所有権を侵害していないことへの保証を含む) に関しても一切責任を負わないものとします。

インテル製品は、予告なく仕様が変更されることがあります。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。