インテル® ソフトウエア開発製品 利用方法

Revision 6.1KS

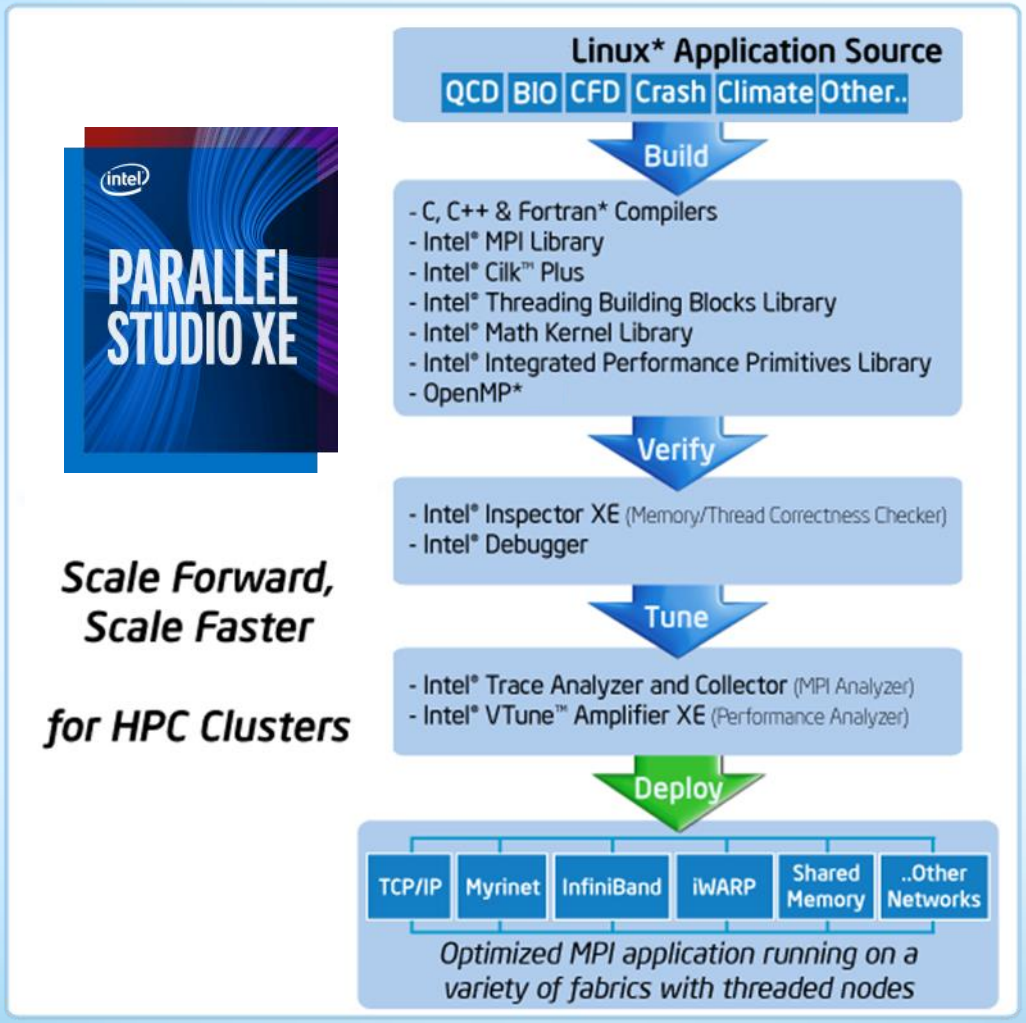
エクセルソフト株式会社

黒澤 一平

XLSOFT

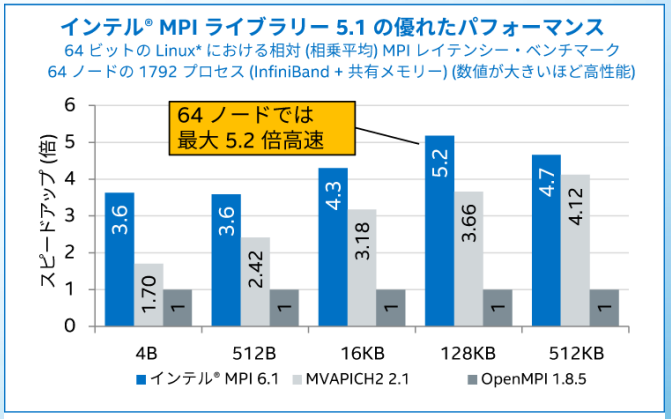
インテル® Parallel Studio XE Cluster Edition 概要

インテル® Parallel Stduio XE 2016 Cluster Edition
クラスターシステム向けの総合開発ツール



それぞれのフェーズに対応する
ハイパフォーマンス・ツール

- 高速化に特化したC、C++、Fortran コンパイラー
- 最適化されたライブラリー
- エラー検出ツール、デバッガー
- 性能解析ツール



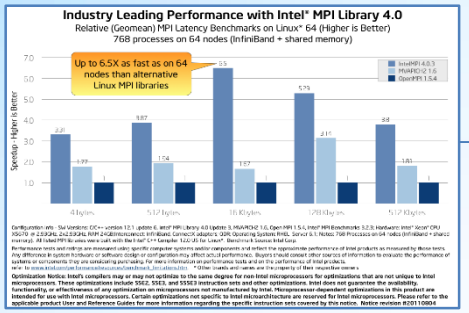
インテル® Parallel Studio XE 2016 の構成詳細

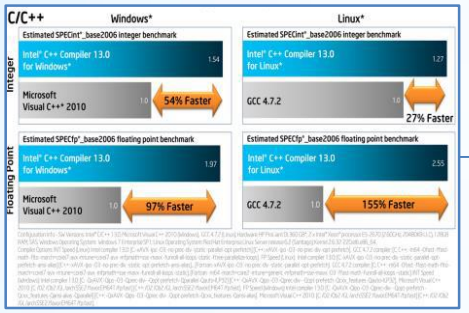
フェーズ	製品名	機能	利点
ビルド	 **インテル® MPI ライブラリー**	ハイパフォーマンスな MPI ライブラリー	• ハイパフォーマンス、スケーラビリティー、インターコネクトの独立性、実行時のファブリック選択、アプリケーション・チューニングを実現
	 **インテル® Composer XE**	C/C++ 、Fortran コンパイラーとパフォーマンス・ライブラリー • インテル® スレッディング・ビルディング・ブロック • インテル® Cilk™ Plus • インテル® インテグレーテッド・パフォーマンス・プリミティブ • インテル® マス・カーネル・ライブラリー	• マルチコアと将来のメニーコアのパフォーマンスおよびスケーラビリティーを引き出すアプリケーションを開発するためのソリューション
検証	 **インテル® Inspector XE**	コードの品質を高める動的なメモリー/スレッド化解析とスタティック・セキュリティー解析	• 生産性とコードの品質を高め、コストを削減し、早期にメモリー/スレッド/セキュリティーの問題を発見 • 各クラスターノードで MPI に対応
検証 & チューニング	 **インテル® トレース・アナライザー/コレクター**	アプリケーションの正当性と動作を理解するための MPI パフォーマンス・プロファイラー	• MPI プログラムのパフォーマンスを解析し、並列アプリケーションの動作と通信パターンを視覚化して、hotspot を特定
チューニング	 **インテル® VTune™ Amplifier XE**	アプリケーションのパフォーマンスとスケーラビリティーを最適化するパフォーマンス・プロファイラー	• 従来の推測作業を排除し、短時間で容易にパフォーマンスとスケーラビリティーのボトルネックを特定 • 各クラスターノードで MPI に対応
並列化、ベクトル化	 **インテル® Advisor**	並列化とマルチスレッド化のアドバイザー	• マルチスレッド化すべき場所を特定 • マルチスレッド化性能予測 • ベクトル化状況を解析

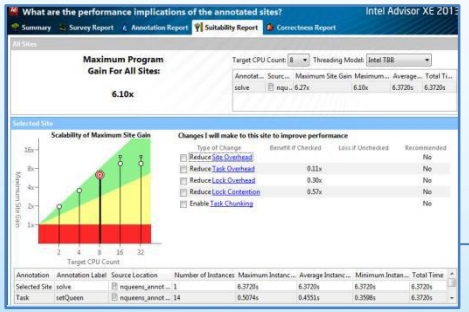
インテル® Composer XE に含まれるライブラリー

製品名	機能	利点
インテル® マス カーネル ライブラリー (インテル® MKL)	ハイパフォーマンスな数値演算ライブラリー ・LAPACK, ScaLAPAC ・BLAS, Sparse BLAS, PBLAS ・PARDISO ・FFT など	• 高速化、並列化されており、プロセッサーの性能を最大限活用することができる
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 (インテル® IPP)	ハイパフォーマンスなマルチメディアライブラリー • 信号処理、音声処理 • 画像、動画処理 • データ処理 • 暗号化	• 基関数を組み合わせるだけで様々な高速な関数を作成することができる
インテル® スレッディング ビルディング ブロック (インテル® TBB)	C++言語用のマルチスレッド化対応テンプレートライブラリー	• マルチスレッド化のための抽象化されたテンプレート、コンテナー、およびクラスを使用することでハイパフォーマンスなマルチスレッドアプリケーションの作成を実現

それぞれの性能、インターフェース







インテル® MPI ライブラリー

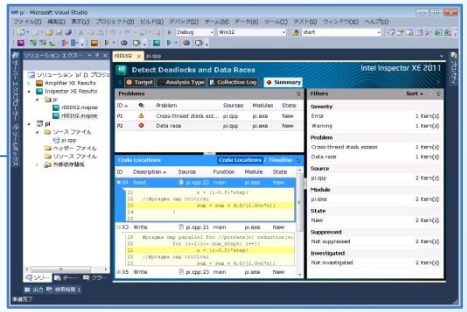
インテル® Composer XE

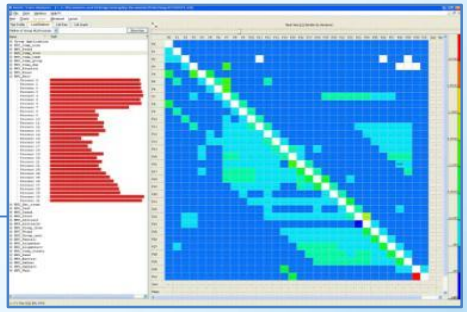
インテル® Inspector XE

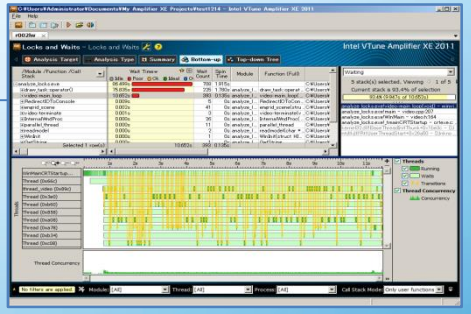
インテル® トレース・アナライザー/コレクター

インテル® VTune™ Amplifier XE

インテル® Advisor XE





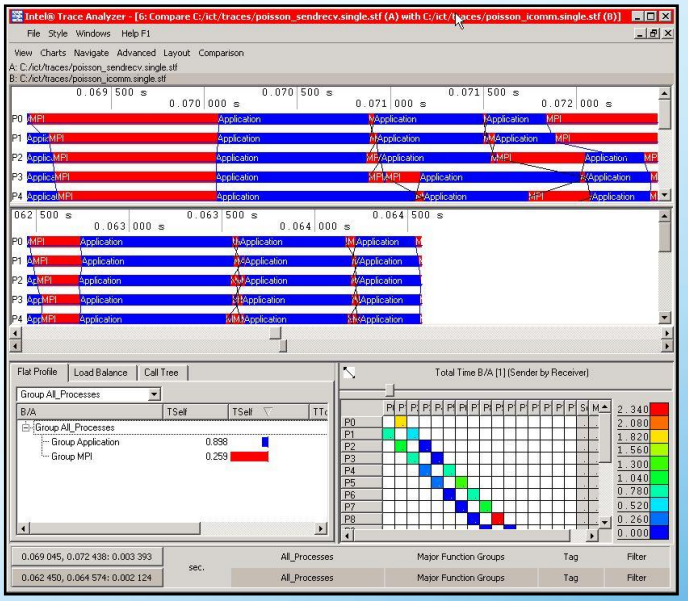


インテル® Trace Analyzer/ Collector の使用方法

インテル® Trace Analyzer / Collectorの概要

- インテル® Trace Analyzer/Collector
  - MPI アプリケーションの動作やパフォーマンス問題を視覚化
  - プロファイリング統計とロードバランス、通信 hotspot の解析

- 機能
  - イベントベースのアプローチ
  - MPI 関数とユーザーコードを分離
  - 低いオーバーヘッド
  - 複数のプロファイルの比較
  - 強力な集計機能およびフィルタリング機能
  - MPI 正当性検証



インテル® Trace Analyzer/ Collector の動作テスト1 実行時実装

- コンパイルは通常通り行う

- 実行 (stf ファイルの生成)

  ＞ mpiexec.hydra -trace -n 8 ./test.exe

- インテル Trace Analyzer の起動

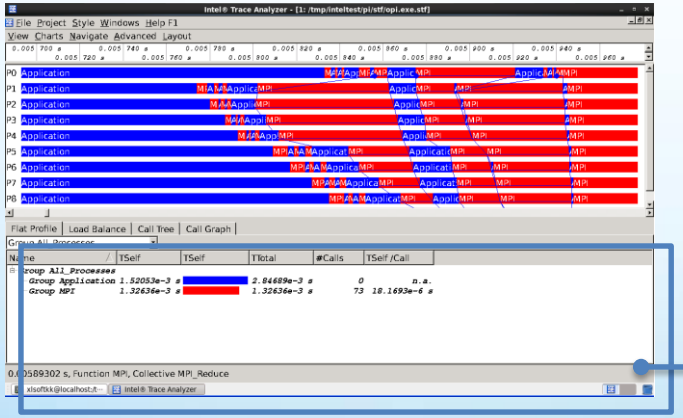
  ＞ traceanalyzer text.exe.stf

  または、以下コマンドでGUI起動後に stfファイルを開く

  ＞ traceanalyzer

インテル® Trace Analyzer/ Collector の動作テスト2

- stf ファイルを開いていない場合: [File]-[Open] より stfファイルを選択
- タイムラインを表示: [Charts]-[Event Timeline]
- タイムライン表示部分をマウスドラッグして拡大
- タイムラインの表示を変更することで Function Profile に表示される内容がフィルタリングされることを確認する



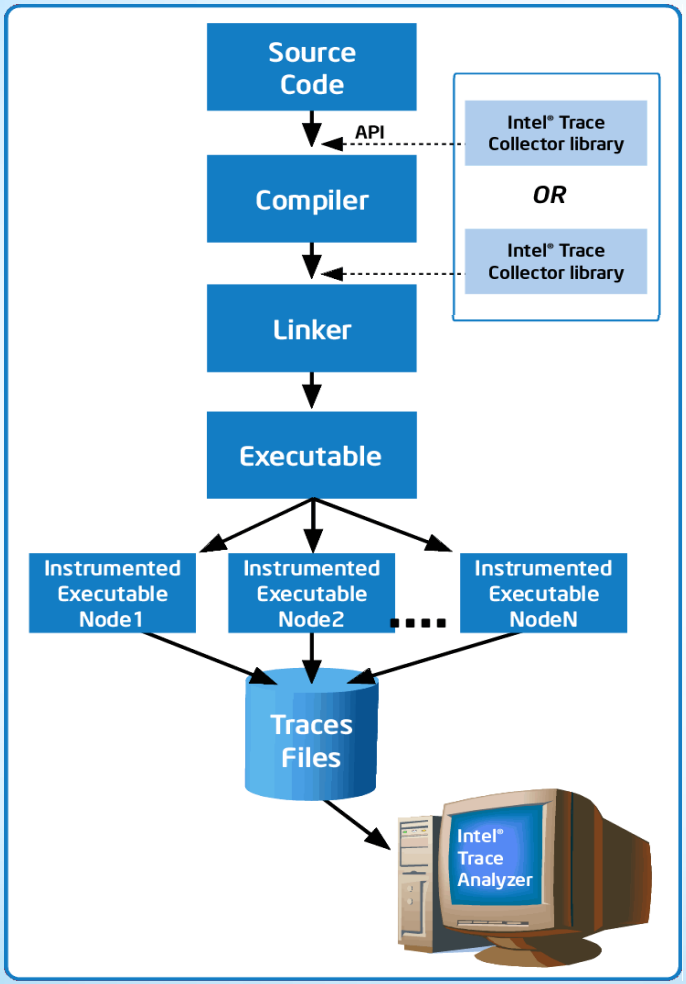
Function Profile

Tips: ショートカットキー

拡大(ズームイン):	i
縮小(ズームアウト):	o
ズームリセット:	r

必須

インテル® Trace Analyzer とインテル® Trace Collector



TC インテル® Trace Collector

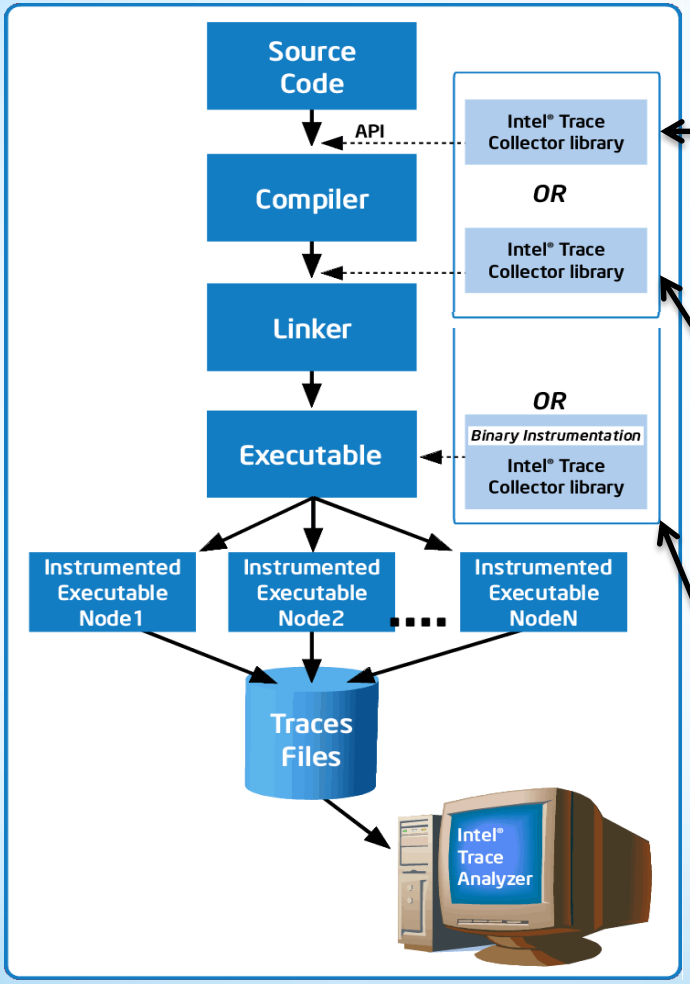
- MPI アプリケーションの解析を主目的とした情報収集ツール
- API、コンパイル時で実装することができ、解析情報は動的に収集される

TA インテル® Trace Analyzer

- インテル® Trace Collector が収集した解析情報を統計的に表示するツール
- 関数別、ユーザーコード/MPI別などにソートし、タイムライン表示やソースコード表示により解析結果を確認することができる

インテル® Trace Collector (TC) の実装

以下のいずれかの方法で実装



・組み込み関数の使用

ソースコードに組み込み関数を挿入してコンパイルすることで、TCが組み込まれたバイナリーが生成される。設定や環境変数により設定を変更することができる。

・コンパイル/リンク時

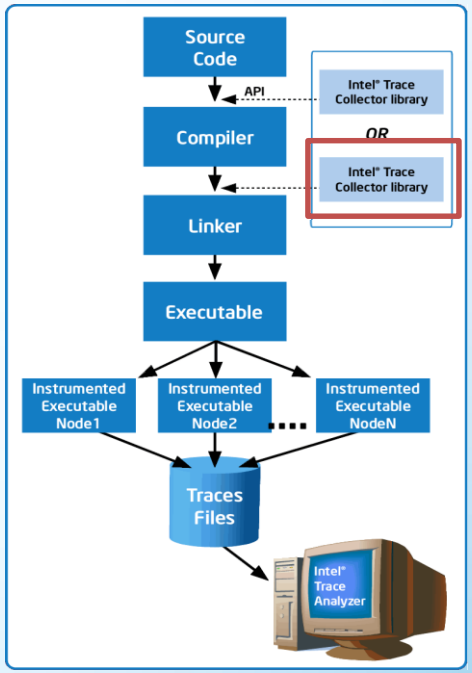
コンパイル時にTCに対応させるオプションを指定することができる。コンパイル時または後から環境変数によって動作を設定することができる。

・実行時

実行時に -trace オプションを指定することで解析可能。

コンパイル時のインテル® Trace Collector 実装 1

- コンパイル時に実装する目的
  - コンパイル時実装の場合、実行ファイルを実行するたびにトレースファイルが生成される
  - 入力データを複数パターン試す場合など、コンパイルせずに何度か実行する場合に、実装時間を削減することができる



ソースファイル → コンパイル時に実装 → 通常どおりの実行でトレースファイルが生成

コンパイル時のインテル® Trace Collector 実装 2 (バージョン 9.0 以降)

- コンパイル フォーマット

```
> mpiicc   sample.c   -trace -profile=vt
> mpiicpc  sample.cpp -trace -profile=vt
> mpiifort sample.f   -trace -profile=vt

> mpiicc sample.c  -trace -profile=vt
```

トレースの種類を決めるプロファイル・オプション

-profile=	概要
-lVT	トレースファイルアプリケーション終了時に生成する
-lVTfs	トレースファイルを随時生成する、Fale-safe 方式
-lVTmc	正確性チェックを追加 (エラー検出)

実行は通常通り

```
> mpiexec.hydra -n 8 ./a.out
```

コンパイル時のインテル® Trace Collector 実装 3 (バージョン 8.1 以前の利用方法、9.0 でも利用可能)

- TCライブラリーの選択

```
> mpiicc sample.c -L$VT_LIB_DIR -lVT
```

-lVT を以下のいずれかに変更することでトレース方法を変更することができる

ライブラリー		概要
-lVTnull	libVTnull.so	擬似ライブラリー。実行時にトレースファイルを生成しない
-lVT	libVT.so	トレースファイルアプリケーション終了時に生成する
-lVTfs	libVTfs.so	トレースファイルを随時生成する、Fale-safe 方式
-lVTmc	libVTmc.so	正確性チェックを追加 (エラー検出)
-lVTcs	libVTcs.so	シングルプロセスのみ生成してトレースファイルを生成

Table 1.4 Usage of Components
http://software.intel.com/sites/products/documentation/hpc/ics/itac/81/ITC_Reference_Guide/hh_goto.htm#1_2_System_Requirements_and_Supported_Features.htm

ソースビューの表示方法

1) –g オプションを指定してコンパイルし、実行時に環境変数、VT_PCTRACE を設定する。または事前に export (setenv) も可

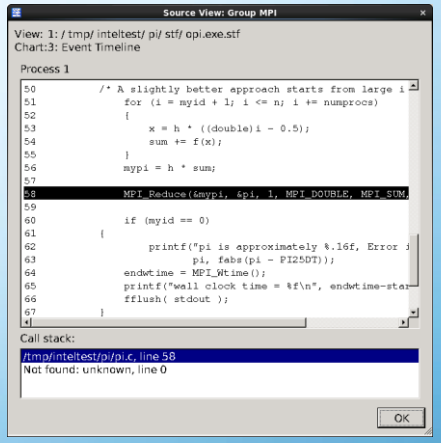
```
>mpiicc -g sample.c -o sample.exe -trace -O2
>mpiexec.hydra -genv VT_PCTRACE=5 -n 8 ./sample.exe
```

2) TA を起動して、sample.stf を開く

3) タイムライン表示、[Charts]-[Event Timeline] の任意の箇所を右クリックして [Details on Function] をクリック

4) [Show Source] ボタンをクリック

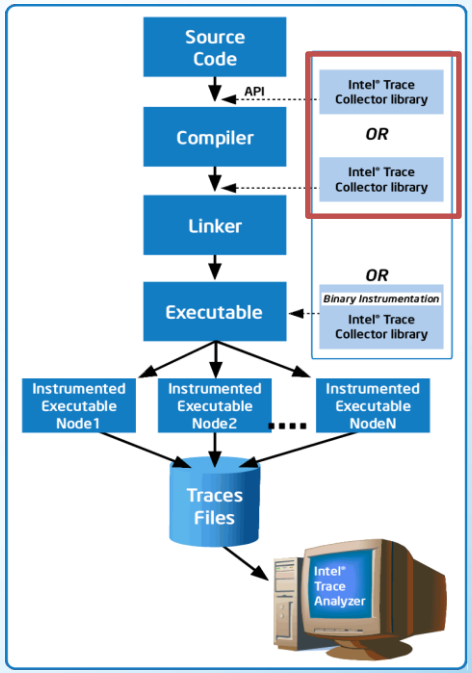
5) 該当箇所のソースコードが表示される



環境変数 VT_PCTRACE を設定することで、コールスタックのプログラムカウンター(PC)を記録し、ソースコードの位置と分析結果を紐付ける

組み込み関数 1

- 組み込み関数を使用する目的
  - トレースに必要な時間が長い場合や、大量のMPI関数をコールする際に、データを見やすくし、トレースファイルのサイズを小さくすることができる
  - 任意の箇所に関数を配置することができるため、高い柔軟性がある



ソースファイルに組み込み関数を追加 ▶ コンパイル時に実装 ▶ 通常どおりの実行でトレースファイルが生成

インテル® Trace Analyzer の基本機能

- マニュアル、Tutorial_Analyzing_MPI_Application.htm を参照

- トレース済みのサンプルファイル

  <Install dir>/examples/traces/

  poisson_sendrecv.single.stf:　MPI_sendrecv を用いたブロック通信

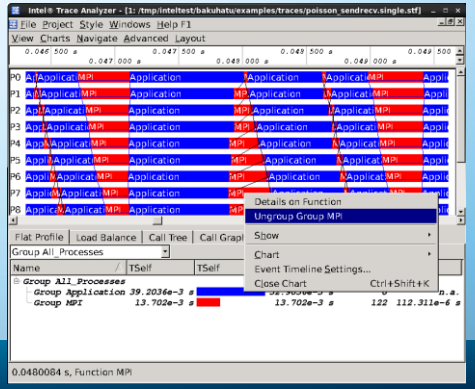
  poisson_icomm.single.stf:　ブロック無し通信への改良版

  インテル® Trace Analyzer で上記ファイルを開く

インテル® Trace Analyzer の基本機能

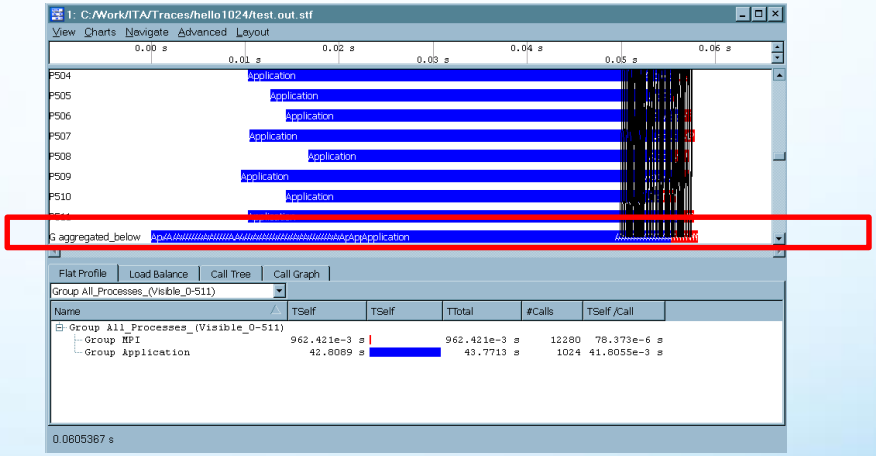
- タイムラインを表示 [Charts]-[Event Timeline]
  - グラフを拡大してアプリケーション (青色部分)とMPI (赤色部分)の関係を確認する
    - MPIタイムの比率が大きすぎないか
    - 各プロセスが均一か

- 問題の可能性のある箇所の詳細を確認
  - Event Timeline の赤色部分を右クリックし、[MPI Ungroup Group MPI] を選択すると使用されている MPI 関数の詳細が表示される



Advanced Aggregation: データの集約 1

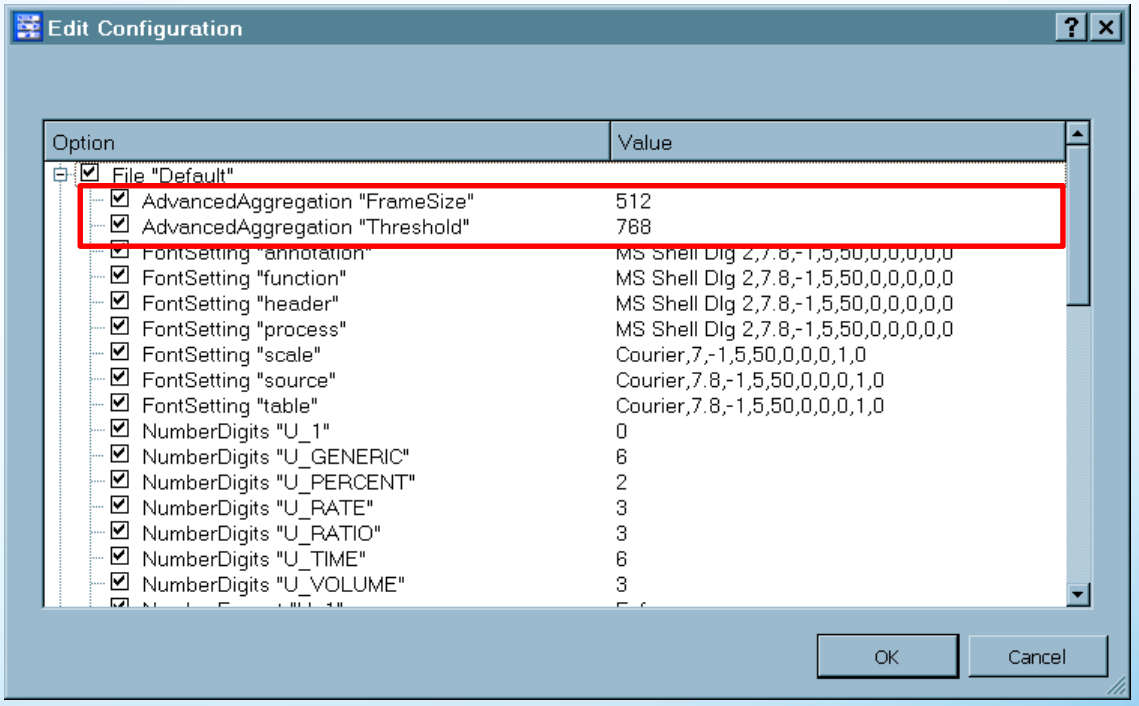
- インテル® Trace Analyzer 8.0.3 より実装された Advanced Aggregation 機能は、大量のデータを特定の事象ごとに集約して表示
  - プロセス数に応じてデータを集約する
  - 1000プロセス以上のデータを扱うことができ、初期値では 768 以上のプロセスは自動的に集約表示される



集約されたプロセスは纏められ、
1プロセスのように表示される

Advanced Aggregation: データの集約 2

- 自動集約される閾値は 変更可能
  [File]-[Edit Configuration] にて表示されるダイアログで設定



参考: MPI エラーの検出方法

システムB,C で使用可能
(システムA,D,Eでは使用不可)

MPI エラーの検出方法1

- Message Checking ライブラリーを使用してデッドロック、不正なメモリーアクセス、競合状態、MPIエラー等を自動検出

```
> mpiicc    sample.c    -trace -profile=vtmc
```

vtmc	Message Checking ライブラリーを指定

```
>mpiexec.hydra -genv VT_CHECK_TRACING on -genv VT_DEADLOCK_TIMEOUT 20s
 -genv VT_DEADLOCK_WARNING 25s -n 8 ./a.out
```

VT_CHECK_TRACING on	Message Checking のトレーシングをオン
VT_DEADLOCK_TIMEOUT 20s	停止した場合のタイムアウト指定 (20秒)
VT_DEADLOCK_WARNING 25s	停止した場合の警告を表示。デッドロックやロードインバランスの発見に有効 (25秒)

MPI エラーの検出方法2

デッドロックの検出例

- サンプルコードを入手
  http://www.shodor.org/refdesk/Resources/Tutorials/BasicMPI/deadlock.c

- libVTmc.so を使用してコンパイルし、デッドロックによる停止に対応できる設定を行い、トレースする (または、後記の実行時の実装などでも可能)

```
> mpiicc -g deadlock.c -o deadlock.exe -L${VT_LIB_DIR} -lVTmc ${VT_ADD_LIBS}

>mpiexec.hydra -genv VT_CHECK_TRACING on -genv VT_DEADLOCK_TIMEOUT 20s -genv
VT_DEADLOCK_WARNING 25s -n 2 ./deadlock.exe 0 80000
```

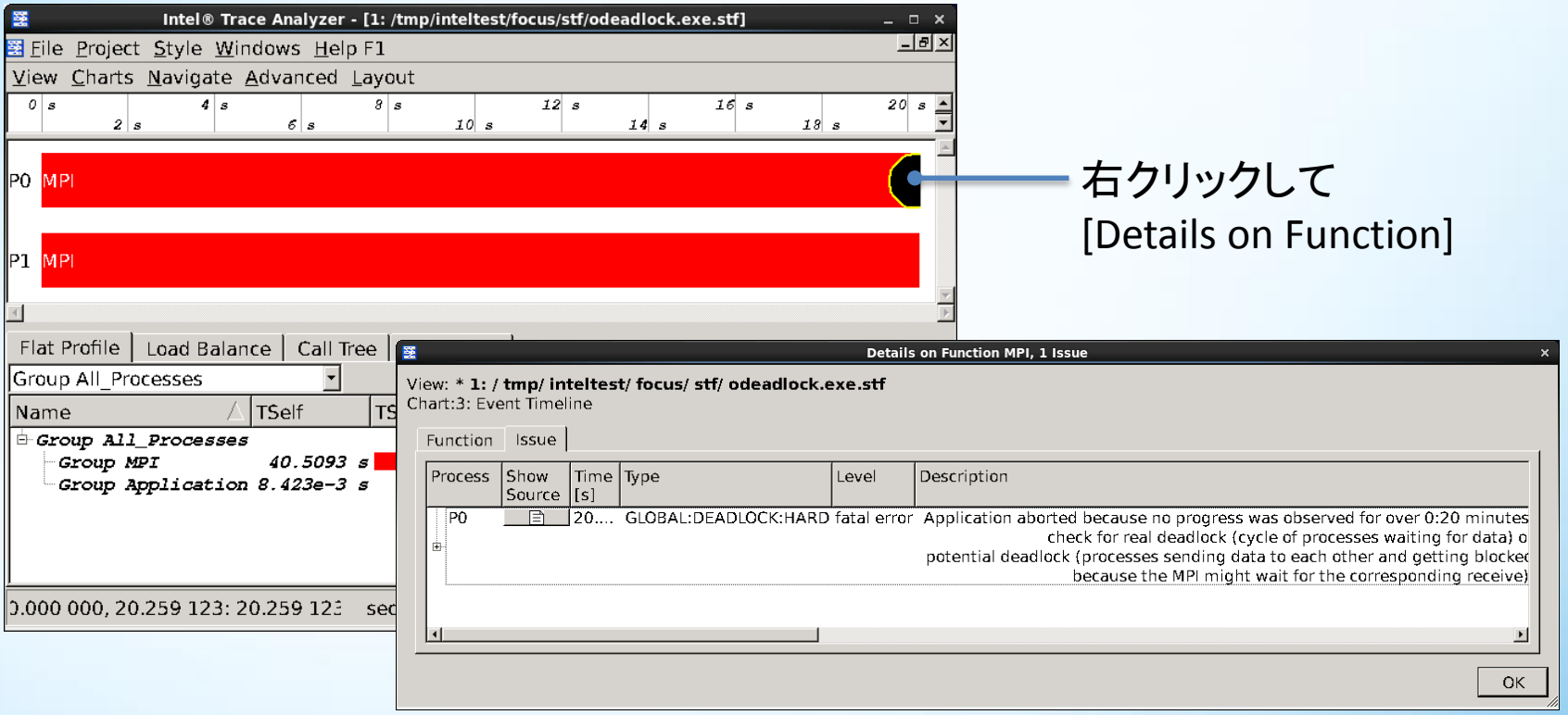
```
[0] ERROR: no progress observed in any process for over 0:20 minutes, aborting application
[0] WARNING: starting emergency trace file writing

[0] ERROR: GLOBAL:DEADLOCK:HARD: fatal error
[0] ERROR:    Application aborted because no progress was observed for over 0:20 minutes,
[0] ERROR:    check for real deadlock (cycle of processes waiting for data) or
[0] ERROR:    potential deadlock (processes sending data to each other and getting blocked
[0] ERROR:    because the MPI might wait for the corresponding receive).
[0] ERROR:    [0] no progress observed for over 0:20 minutes, process is currently in MPI call:
[0] ERROR:       MPI_Recv(*buf=0x7fff1cdb2474, count=800000, datatype=MPI_INT, source=1,
tag=999, comm=MPI_COMM_WORLD, *status=0x7fff1d0bf874)
[0] ERROR:       main (/tmp/inteltest/focus/deadlock.c:49)
[0] ERROR:       (/lib64/libc-2.12.so)
[0] ERROR:       (/tmp/inteltest/focus/odeadlock.exe)
```

```
[0] ERROR:    [1] no progress observed for over 0:20 minutes, process is currently in MPI call:
[0] ERROR:       MPI_Recv(*buf=0x7fffa4bd4ce4, count=800000, datatype=MPI_INT, source=0,
tag=999, comm=MPI_COMM_WORLD, *status=0x7fffa4ee20e4)
[0] ERROR:       main (/tmp/inteltest/focus/deadlock.c:49)
[0] ERROR:       (/lib64/libc-2.12.so)
[0] ERROR:       (/tmp/inteltest/focus/odeadlock.exe)
[0] INFO: Writing tracefile odeadlock.exe.stf in /tmp/inteltest/focus/stf

[0] INFO: GLOBAL:DEADLOCK:HARD: found 1 time (1 error + 0 warnings), 0 reports were suppressed
[0] INFO: Found 1 problem (1 error + 0 warnings), 0 reports were suppressed.

APPLICATION TERMINATED WITH THE EXIT STRING: Hangup (signal 1)
```

実行時に表示されるメッセージ

MPI エラーの検出方法3

TA による表示



右クリックして
[Details on Function]

タイムライン表示、[Charts]-[Event Timeline] の黒丸を右クリックして
[Details on Function] をクリックし、[Issue] タブをクリックすると詳細が
表示される。

インテル® VTune™ Amplifier XE によるパフォーマンス解析

インテル® VTune™ Amplifier XE

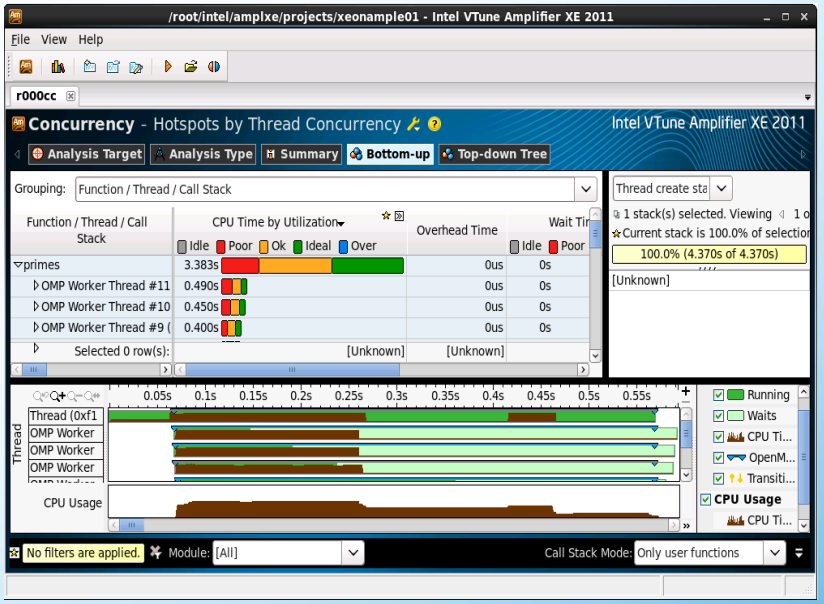
- パフォーマンス向上のための解析ツール
  ソースコードの修正や、特別なビルドなしに該当箇所を特定

Hotspot
処理に時間を要する箇所を特定

Concurrency
マルチスレッドの並列性を表示

Lock & Wait
スレッド間通信などの情報

Hardware Event
プロセッサーで生じたイベントを表示

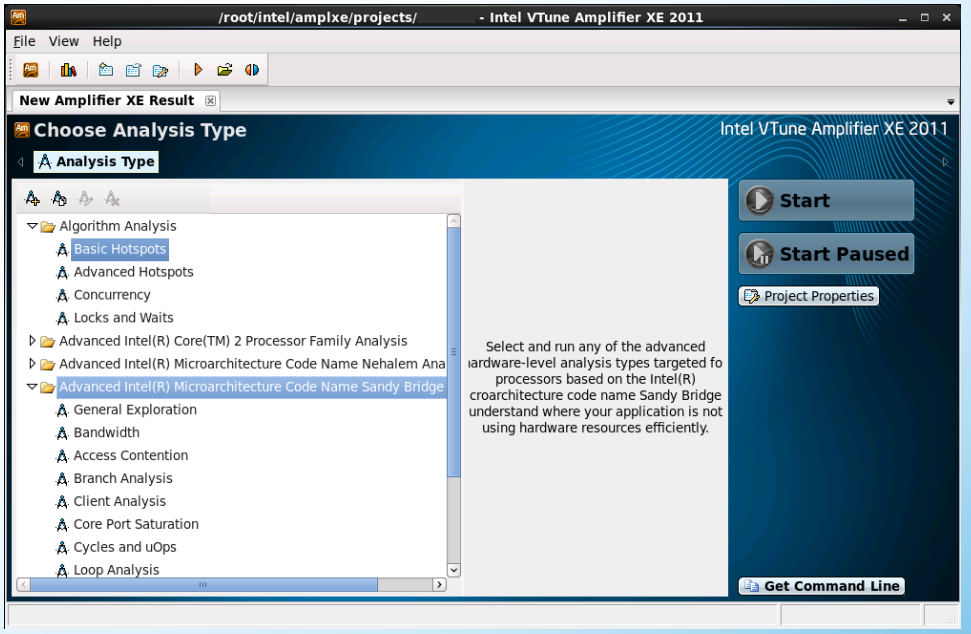


プリセットされた解析タイプ

- GUI およびコマンドラインから、プリセットされた解析タイプを選択するだけで、主要な情報を取得することができる

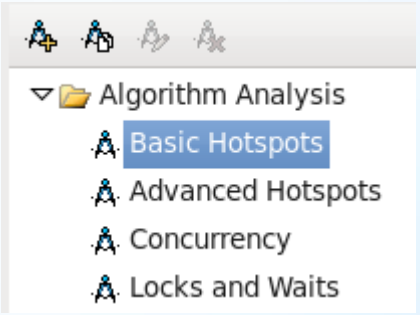
<Analysis Type> 例：
hotspots
concurrency
locksandwaits

advanced-hotspots
general-exploration
bandwidth



GUI は、初めての利用者でも使用できるように、選択箇所の説明が表示される

解析タイプ



◆ Advanced Hotspots (-collect advanced-hotspots)

- CPU の処理に時間を要している箇所を検出する
- オーバーヘッドが少ない
- 関数コールスタック情報を収集しない
- 論理コア単位の動作状況も表示できる

◆ Basic Hotspots (-collect hotstpots)

- CPU の処理に時間を要している箇所を検出する（オーバーヘッドは約5%）
- 実行中のインストラクション・ポインター(IP)、関数コールスタック情報などを収集する
- 同時実行コア数のヒストグラム表示ができる
- アプリケーションまたはプロセスの検出を行う（システム全体の検出はできない）

◆ Concurrency (-collect concurrency)

- 詳細なスレッド動作の検出を行う
- スレッドの同時利用状況、スレッド間の同期状態などの情報を収集する
- アプリケーションまたはプロセスの検出を行う（システム全体の検出はできない）

◆ Locks and Waits (-collect locksandwaits)

- アイドル状態の検出を行う
- スリープ処理や同期オブジェクト、I/O 処理待ち状態などの情報を収集する
- アプリケーションまたはプロセスの検出を行う（システム全体の検出はできない）

プロセッサーイベントに関する解析

1. 解析タイプ“General Exploration”からサンプリングをスタートする
   コマンドラインの場合には (-collect general-exploration)

2. 解析が完了したら、ビューポイントを“（General Exploration”に設定し、
   “Summary”ウィンドウから「Hardware Event-based Metrics」を確認する

   - この 評価基準は、インテルアーキテクトによって定められたイベント比率であり、評価基準の閾（しきい）値が定義されている

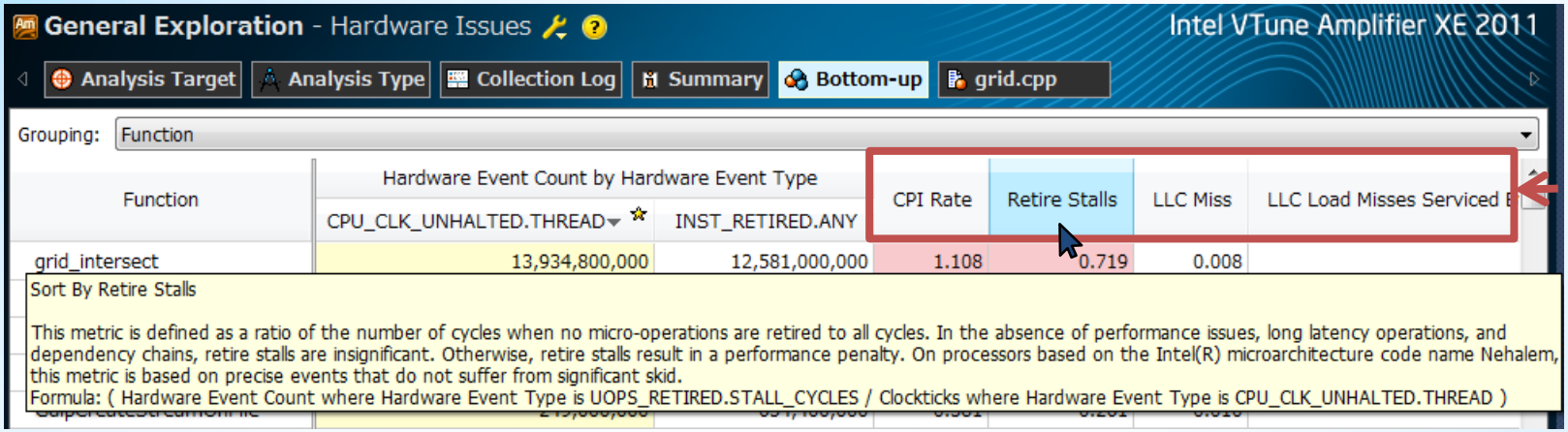
3. 関数単位の Metrics 内容を確認するために、Bottom-up ボタンをクリック

   - 5%以上のCPU 使用率をもつ関数はホットスポットと認識され、またその基準値を超えている場合は、潜在的な問題と見なされ、ピンクでマークされる
   - 各項目にマウスポインターでフォーカスすると、その項目の内容とイベント比率を求める公式が表示される
   - ピンクでマークされている箇所にマウスポインターでフォーカスすると、問題修正のヒントや設定された基準の閾値が表示される

4. 問題のある関数はダブルクリックしてソース/アセンブリコードで調査する

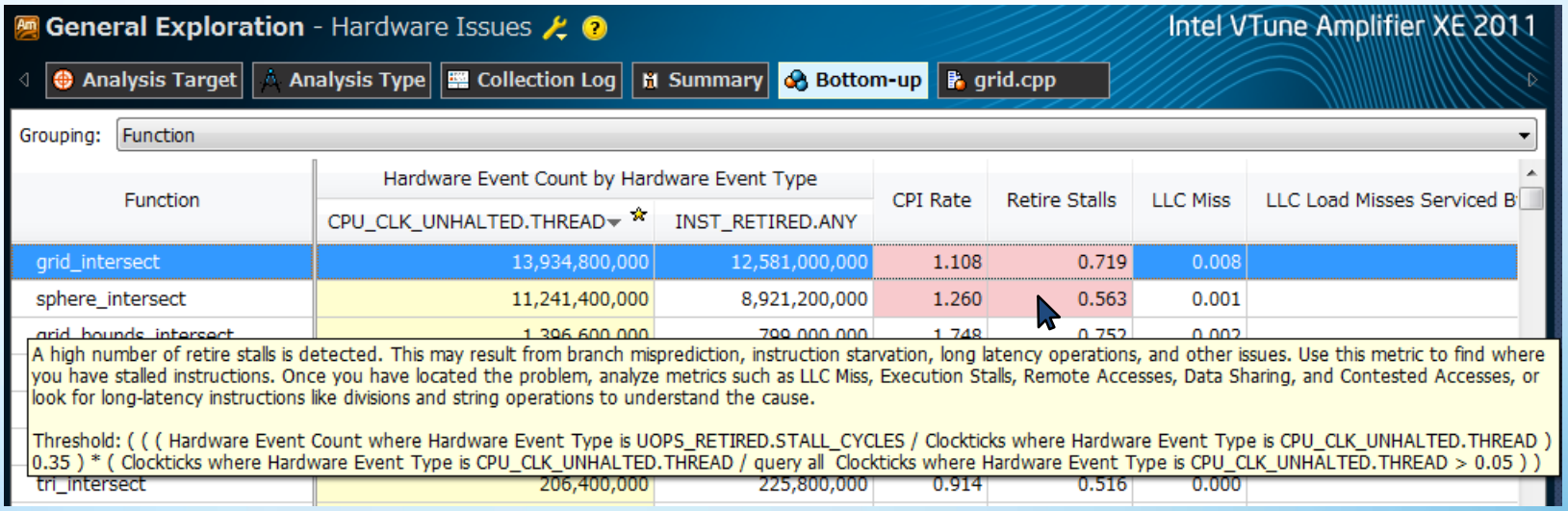
5. 発見された特定の問題に関する解析タイプを使用して更なる分析を行う

基準値に関する情報（マウスカーソルで表示）



基準項目

基準値の説明と公式

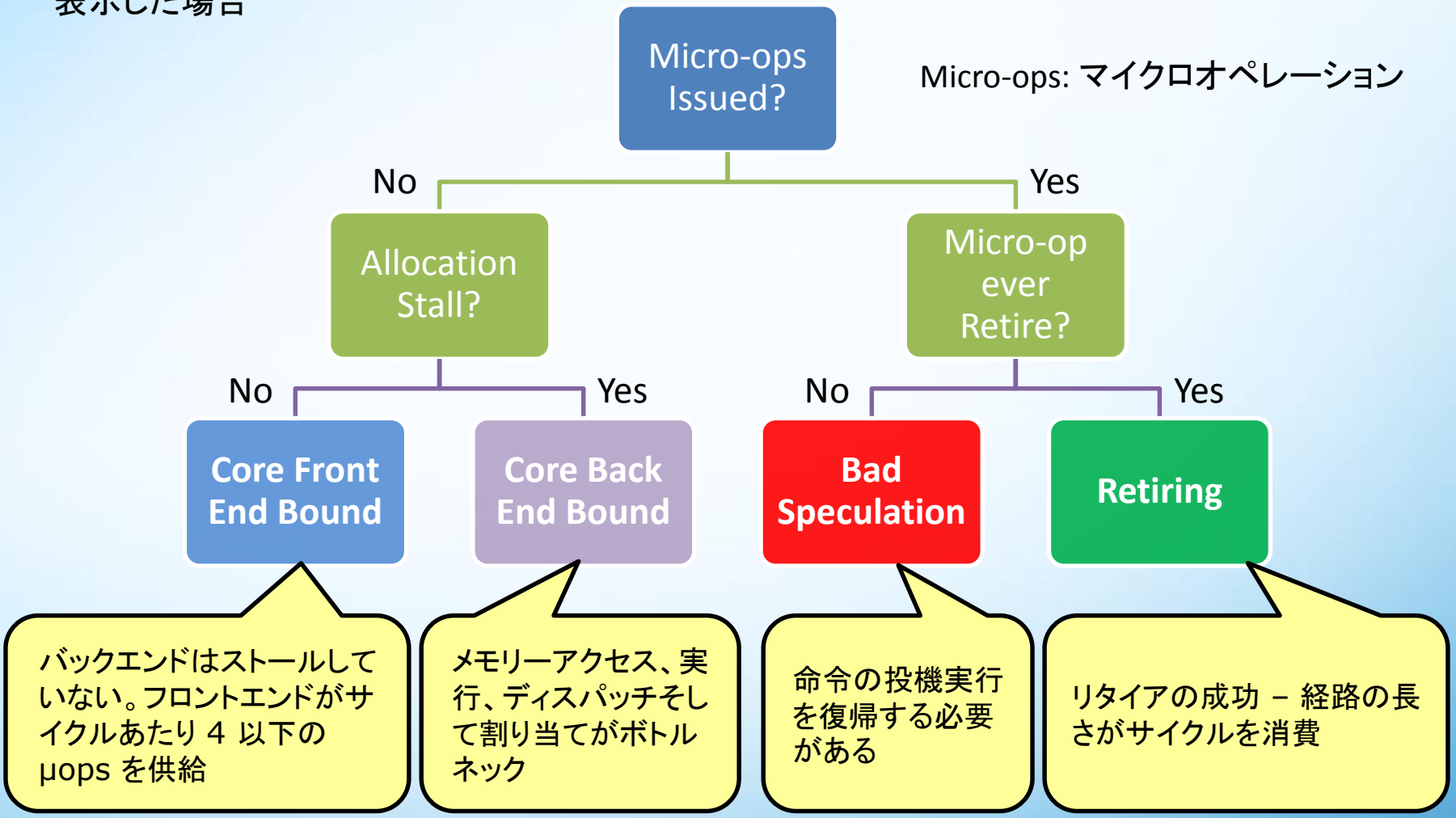


問題解決のヒントと閾値

※ 他Metrics は、Intel VTune Amplifier XE Help ドキュメントから、[Key Concepts] – [Performance Metrics] – [Hardware Event-based Metrics] に記載

プロセッサー イベント サンプリング 解析の考え方(手順)

◆ 解析タイプ“General Exploration”でのサンプリング結果を“General Exploration”ビューポイントで表示した場合



インテル® VTune™ Amplifier XE を使用するための準備

- -g オプションを指定してコンパイル

  例: icc -g openmp_app.c -O2

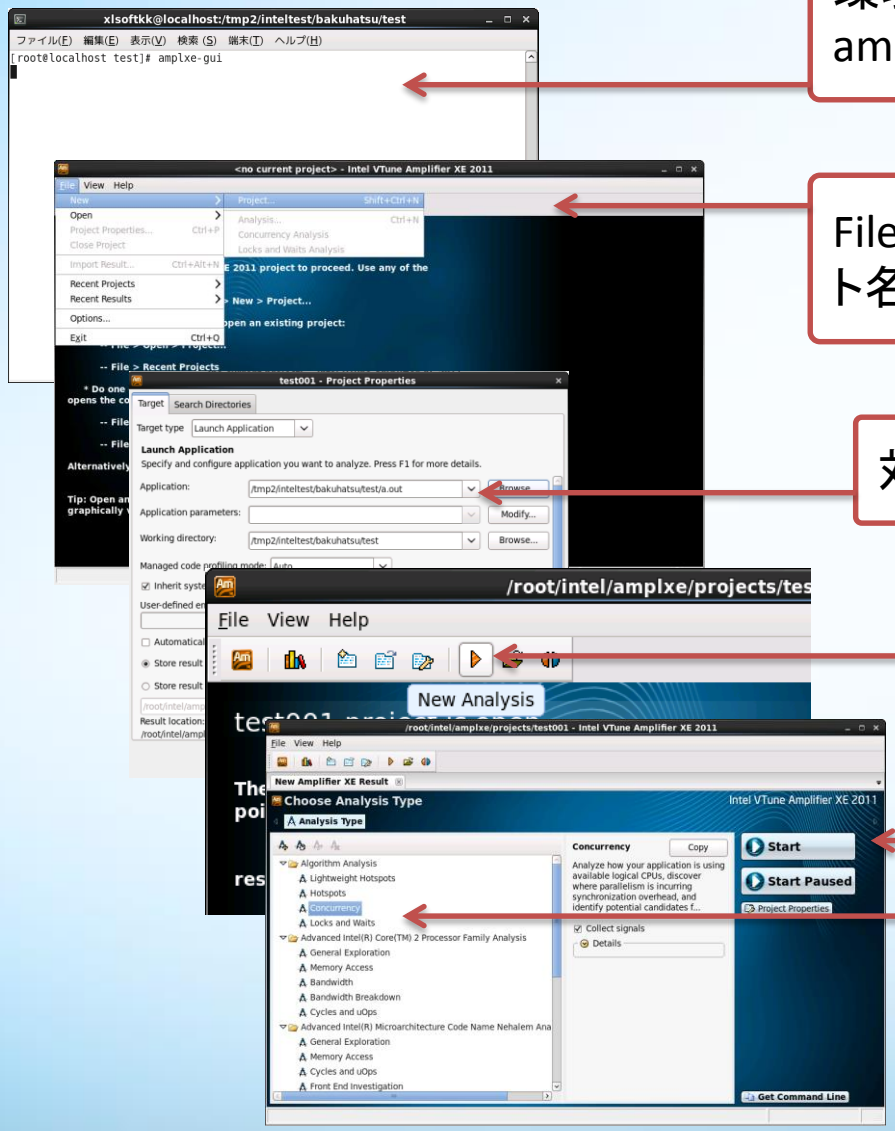
  ※-g オプションを指定してデバッグ情報を生成することで、
  エラー検出時にソースコードと連携して表示させることが出来るようになる。
  -g オプションを付けると最適化が無効になるため、-O2 や -O3 -xHOST などの
  最適化オプションを明示的に指定する。

- GUI または コマンドラインから実行する

  次ページより手順記載

必須

GUI からの利用手順



環境変数を設定し、次のコマンドを実行する
amplxe-gui

File > New > Project にて、任意のプロジェクト名を指定する

対象アプリケーションを指定する

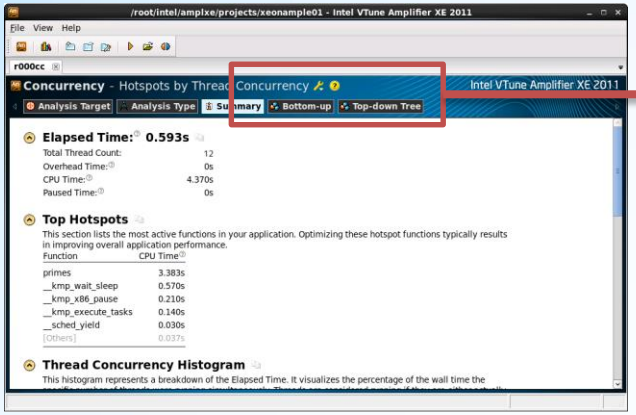
New Analysis ボタンを押す

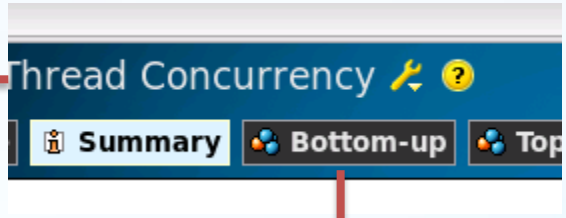
解析タイプを選択し、Start ボタンを押す※詳細については次ページ

必須

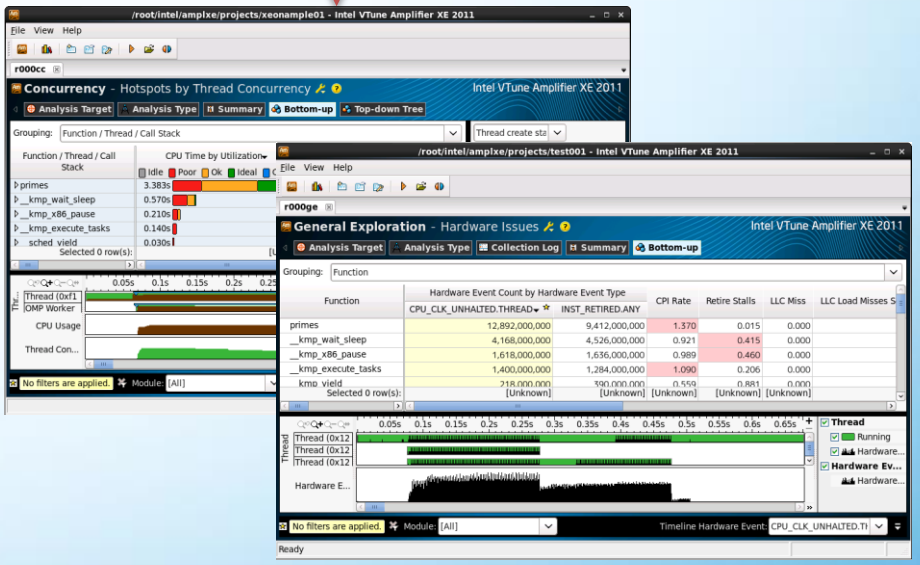
解析結果の操作方法

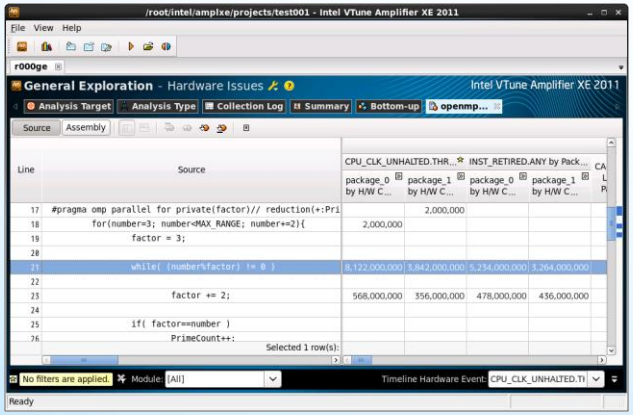
最初の画面ではサマリーが表示される





Bottom-up をクリック





ソースコード行ごとの情報が表示される

解析タイプにより異なる表示方法の
特定箇所をダブルクリック

必須

インテル® VTune™ Amplifier XE のコマンドライン
※MPI アプリケーションを解析する場合にはコマンドラインからの解析が必須

コマンドからインテル® VTune Amplifier XE を実行する

- 1プロセスのみの解析

  amplxe-cl -collect *<Analysis Type>* -- *<program>*

  例: amplxe-cl -collect snb_general-exploration -- ./a.out

- 複数プロセスの解析 (MPIアプリケーション)

  次ページ

必須

インテル® VTune™ Amplifier XE を使用して、複数プロセスの解析を行う場合

- インテル® VTune Amplifier XE は、一部のサンプリング方式では Performance Monitoring Unit (PMU) からデータを取得する

PMUを使用する解析タイプ例
<Analysis Type> =advanced-hotspots, general-exploration, bandwidth

以下のいづれかの方法で可能

```
# mpiexec.hydra -genvall -gtool "amplxe-cl -r <my_result> -collect
<analysis type>:all=exclusive" -n <n> <my_app> [my_app_ options]

# amplxe-cl -collect <Analysis Type> -- mpiexec.hydra -n 1 -host
hoge1 ./a.out …
  または、a.out : -n 1 -host hoge2

# mpiexec.hydra -host host001 -n 1 amplxe-cl -collect <Analysis Type>
-result_dir <dir name> -- <program>: -host host001 -n 15 <program>:
-host host 002 -n 1 amplxe-cl -collect <Analysis Type> -result_dir <dir
name> -- <program>: -host host002 -n 15 …
```


必須

最適化に関する注意事項

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Optimization Notice

最適化に関する注意事項

インテル コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル ストリーミング SIMD 拡張命令 2 (インテル SSE2)、インテル ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。

改訂 #20110804