



# *Intel Software tools: Advisor, VTune™, ITAC*

**Presenter: Heinrich Bockhorst Intel  
FZ Jülich, June 6<sup>th</sup> 2016**



# Agenda

- Advisor XE : enable and improve Vectorization
- VTune™ Amplifier XE: single node performance analysis
- Intel Trace Analyzer and Collector (ITAC): MPI analysis



# *Vectorization Advisor: getting started*



# Before you analyze

*Run GUI or Command Line*

## Set-up environment

- Linux: `source <install-dir>/advixe-vars.sh`

## Run GUI or Command Line:

- **advixe-gui**
- **advixe-cl** `-collect survey -project-dir C:\myadvisor mult.exe`

# *Intel Compiler 16 or 17 – enables more data in Survey*

Min compilation switches (ICC example):

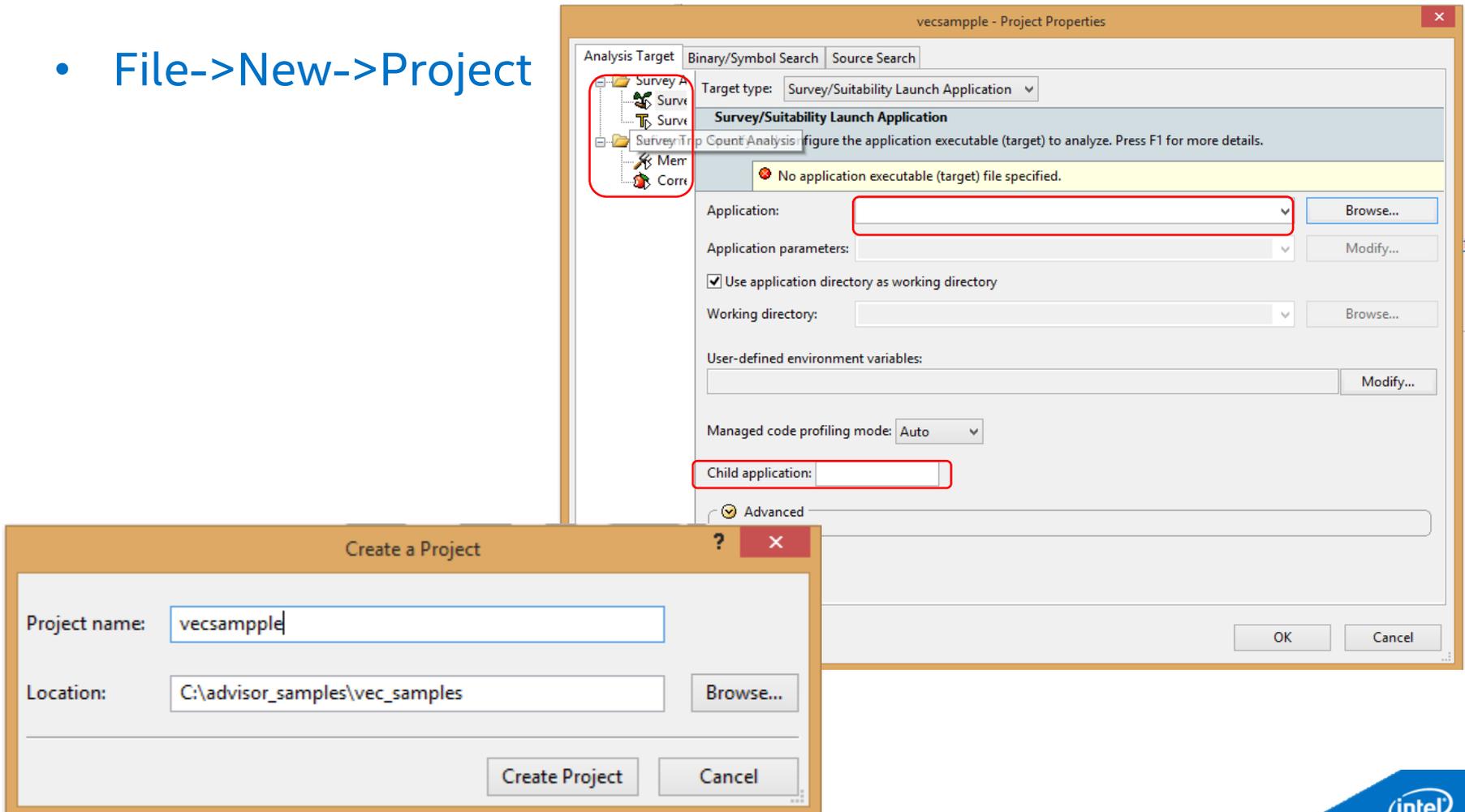
**-O2 -g -xHost**

More aggressive optimization: **-O3**

# GUI: Before you analyze

## Create Project

- File->New->Project



# Analyze what loops you are spending your time in and how they have been vectorized

**1. Survey Target**  
Explore where to add efficient vectorization and/or threading.  
▶ Collect [Icons]  
Command Line

**1.1 Find Trip Counts**  
Find how many iterations are executed.  
▶ Collect [Icon]  
Command Line

**Mark Loops for Deeper Analysis**  
Select loops for deeper analysis. Correctness: Ad. Patterns and: - There are...

**Where should I add vectorization and/or threading parallelism?** Intel Advisor XE 2016

Summary | **Survey Report** | Refinement Reports | Annotation Report | Suitability Report

Elapsed time: 15.47s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops				Instruction Set Analysis	
						Vecto...	Estim...	Vector Length	Compiler Estimated Gain	Traits	Data Types
[Icon] [lo...]	1 Assumed ...	14.030s	14.030s	Scalar	vector dependence ...						Float64
[Icon] [lo...]		0.985s	15.015s	Scalar	outer loop was not a ...						Float64
[Icon] [lo...]		0.000s	15.035s	Scalar	loop with function c...						Float64

**2.1 Check Memory Access Patterns**  
Identify and explore complex memory accesses for marked loops. Fix the reported problems.  
▶ Collect [Icon]  
Command Line

**2.2 Check Memory Access Patterns**  
Identify and explore complex memory accesses for marked loops. Fix the reported problems.  
▶ Collect [Icon]  
Command Line

# Same approach for trip counts

**1. Survey Target**  
Explore where to add efficient vectorization and/or threading.  
▶ Collect [Icons]  
Command Line

**1.1 Find Trip Counts**  
Find how many iterations are executed.  
▶ Collect [Icons]  
Command Line

**Mark Loops for Deeper Analysis**  
Select loops in the Survey result for  
Correc  
Patt  
- The

**2.1 Check Loop Dependencies**  
Identify dependencies for marked loops. Fix the reported problems.  
▶ Collect [Icons]  
Command Line

**2.2 Check Memory Access Patterns**  
Identify and explore complex memory accesses for marked loops. Fix the reported problems.  
▶ Collect [Icons]  
Command Line

Click Collect

Trip Counts				
Median	Min	Max	Call Count	Iteration Duration
<b>50</b>	<b>50</b>	<b>50</b>	<b>101000000</b>	<b>&lt; 0.0001s</b>
101	101	101	1000000	< 0.0001s
1000000	1000000	1000000	1	< 0.0001s

Loops	🔥	Vector Issues	Self Time	Total Time	Trip Counts					Loop Type
					Median	Min	Max	Call Count	Iteration Duration	
▶ [loop at Multiply.c:55 in matvec]	<input type="checkbox"/>	💡 1 Assumed de...	14.030s	14.030s	50	50	50	101000000	< 0.0001s	Scalar
▶ [loop at Multiply.c:44 in matvec]	<input type="checkbox"/>		0.985s	15.015s	101	101	101	1000000	< 0.0001s	Scalar
▶ [loop at Driver.c:145 in main]	<input type="checkbox"/>		0.000s	15.035s	1000000	1000000	1000000	1	< 0.0001s	Scalar

# Specify loops for deeper analysis

**Ad** Where should I add vectorization and/or threading parallelism?

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 15.47s Vectorized Not Vectorized FILTER: All Modules All Sources

Loops		Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?
[loop at Multiply.c:55 in matvec]	<input checked="" type="checkbox"/>	1 Assumed ...	14.030s	14.030s	Scalar	vector dependence p ...
[loop at Multiply.c:44 in matvec]	<input checked="" type="checkbox"/>		0.985s	15.015s	Scalar	outer loop was not a ...
[loop at Driver.c:145 in main]	<input checked="" type="checkbox"/>		0.000s	15.035s	Scalar	loop with function c ...

# Deeper analysis

## Check dependencies

**1. Survey Target**  
Explore where to add efficient vectorization and/or threading.  
▶ Collect [Folder Icon]  
Command Line

**1.1 Find Trip Counts**  
Find how many iterations are executed.  
▶ Collect [Folder Icon]  
Command Line

**Mark Loops for Deeper Analysis**  
Select loops in the Survey result for Correctness and/or Memory Access Patterns analysis.  
– There are NO marked loops –

**2.1 Check Correctness**  
Identify and explore loop-carried dependencies for marked loops. Fix the reported problems.  
▶ Collect [Folder Icon]  
Command Line  
– Nothing to analyze –

**2.2 Check Memory Access Patterns**  
Identify and explore complex memory accesses for marked loops. Fix the reported problems.  
▶ Collect [Folder Icon]  
Command Line

Click Collect

We marked 3 loops for a dependency analysis. Two of the loops had no dependencies. One of the loops has Read-After-Write dependency and can't be vectorized.

**Check memory access patterns in your application**

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Site Location	Loop-Carried Dependencies	Strides Distribution	Access Pattern	Site Name
loop at Driver.c:145 in main	✔ No dependencies found	No information available	No information available	loop_site_6
loop at Multiply.c:44 in matvec	✔ No dependencies found	No information available	No information available	loop_site_10
loop at Multiply.c:55 in matvec	✘ RAW:1	No information available	No information available	loop_site_8

# Deeper analysis

## Memory Access Pattern analysis

Stride distribution

**1. Survey Target**  
Explore where to add efficient vectorization and/or threading.

▶ Collect  

Command Line

---

**1.1 Find Trip Counts**  
Find how many iterations are executed.

▶ Collect 

Command Line

---

**Mark Loops for Deeper Analysis**  
Select loops in the Survey result for Correctness and/or Memory Access Patterns analysis.  
– There are NO marked loops –

---

**2.1 Check Correctness**  
Identify and explore loop-carried dependencies for marked loops. Fix the reported problems.

▶ Collect 

Command Line

💧 – Nothing to analyze –

---

**2.2 Check Memory Access Patterns**  
Identify and explore complex memory accesses for marked loops. Fix the reported problems.

▶ Collect 

Command Line

**Check memory access patterns in your application**

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Site Location	Loop-Carried Dependencies	Strides Distribution	Access Pattern	Site Name
loop at Driver.c:145 in main	✔ No dependencies found	100% / 0% / 0%	All unit strides	loop_site_6
loop at Multiply.c:44 in matvec	✔ No dependencies found	85% / 15% / 0%	Mixed strides	loop_site_10
loop at Multiply.c:55 in matvec	✘ RAW:1	74% / 26% / 0%	Mixed strides	loop_site_8

Memory Access Patterns Report Correctness Report

ID	Stride	Type	Source	Nested Function	Modules	Alignment
P3		Parallel site information	Driver.c:145		matrix_vector_multiplication_c.exe	
P9	0	Unit stride	Driver.c:157		matrix_vector_multiplication_c.exe	
P10	0	Unit stride	Multiply.c:39	matvec	matrix_vector_multiplication_c.exe	
P12	0	Unit stride	Multiply.c:44	matvec	matrix_vector_multiplication_c.exe	
P14	0; 1	Unit stride	Multiply.c:45	matvec	matrix_vector_multiplication_c.exe	

```

43     int i, j;
44
45     for (i = 0; i < size1; i++) {
46         b[i] = 0;
47     }
    
```

Click Collect

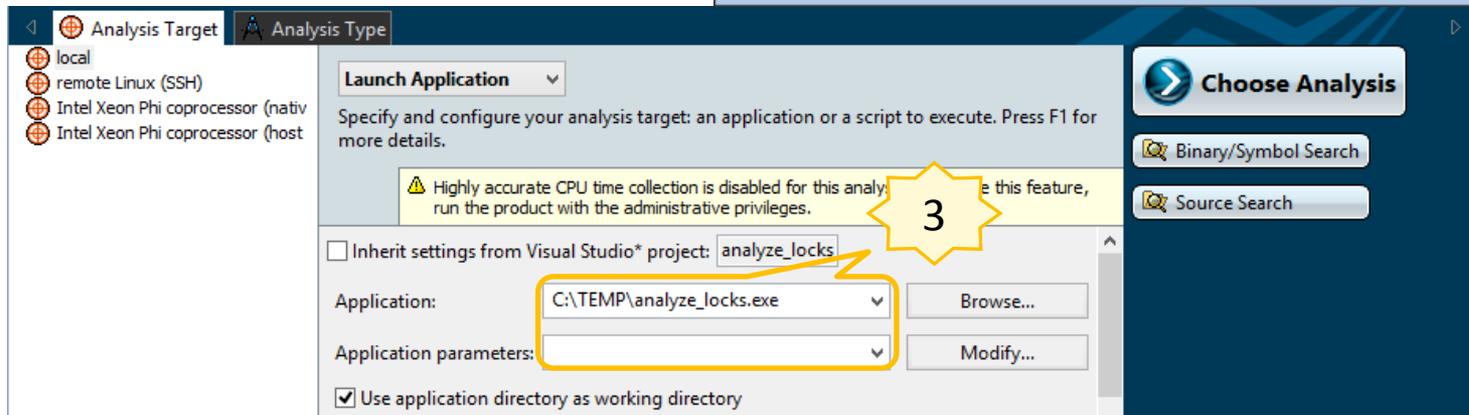
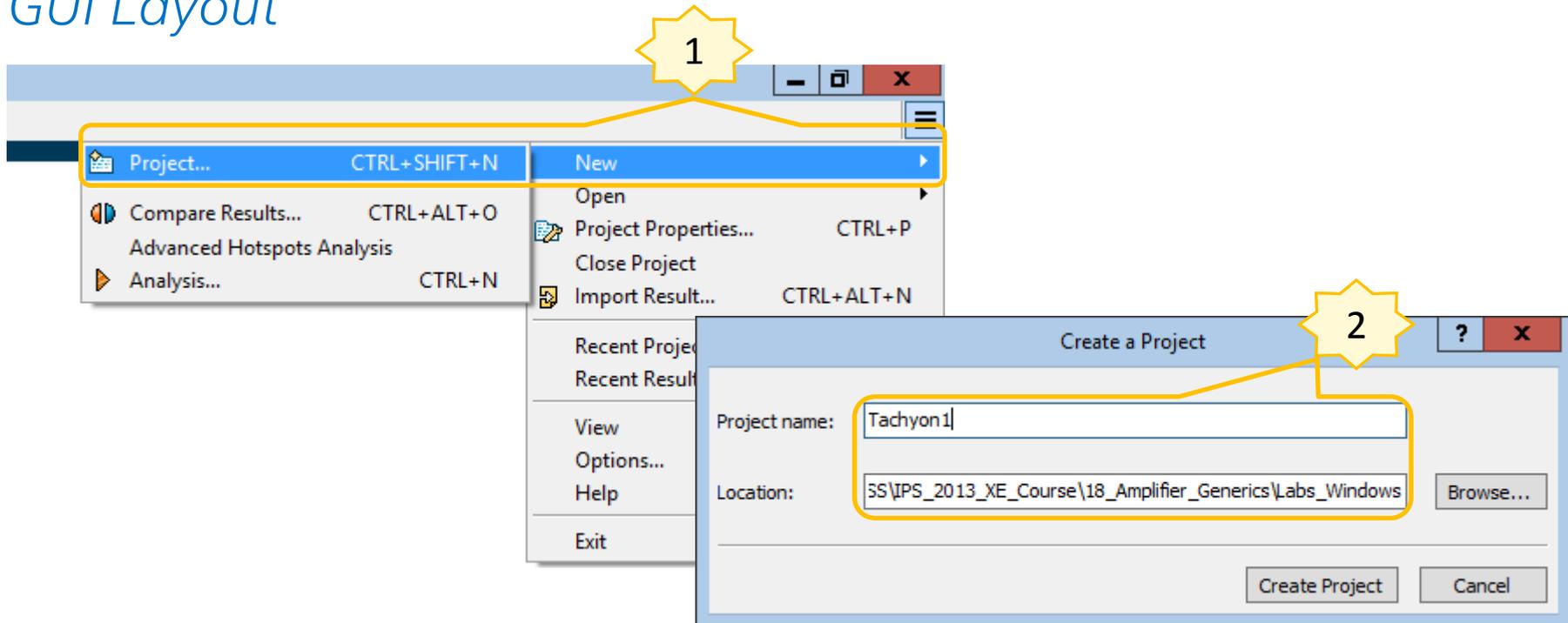


# *VTune™ Amplifier XE: getting started*



# Creating a Project

## GUI Layout



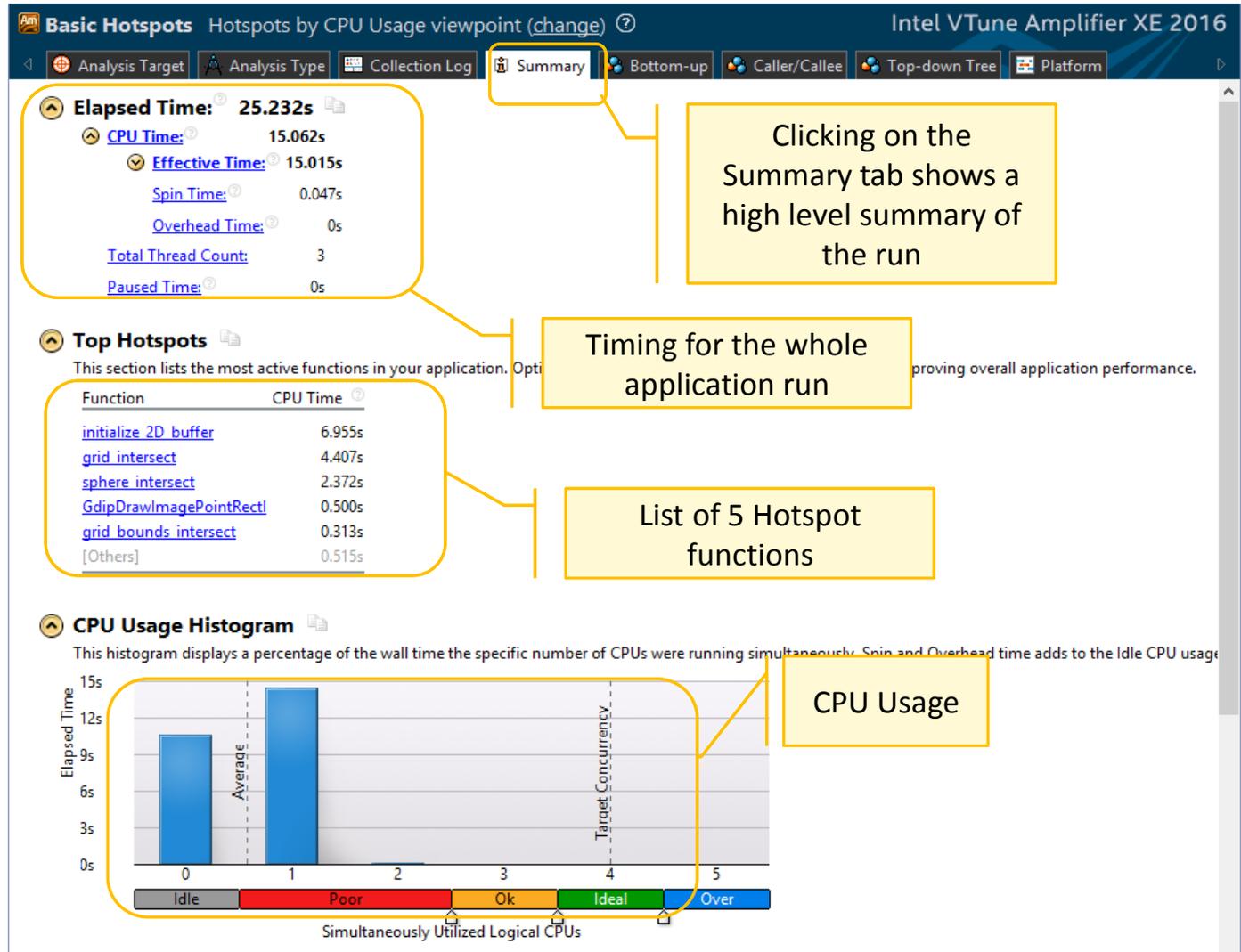
# Selecting type of data collection

## GUI Layout

The screenshot shows the Intel VTune Amplifier XE 2016 interface. The main window is titled "Choose Target and Analysis". On the left, there is a tree view under "Analysis Target" showing various analysis types. A yellow callout bubble points to this tree, stating "All available analysis types". The tree includes categories like "Algorithm Analysis", "Microarchitecture Analysis", "Platform Analysis", and "Custom Analysis". A specific item, "My custom hotspot", is highlighted. A second yellow callout bubble points to the "Start" and "Start Paused" buttons on the right, stating "Different ways to start the analysis". A third yellow callout bubble points to a context menu that is open over the "My custom hotspot" item, listing options like "Copy from Current", "New Hardware Event-based Sampling Analysis", and "New User-mode Sampling and Tracing Analysis", with a callout stating "Helps creating new analysis types". A fourth yellow callout bubble points to a "Command Line..." button at the bottom right, stating "Copy the command line to clipboard". The interface also shows a "My comments..." section and a "Choose Target" button.

# Summary View

## GUI Layout



# Bottom-Up View

## GUI Layout

**Basic Hotspots** Hotspots by CPU Usage viewpoint (change) Intel VTune Amplifier XE 2016

Analysis Target Analysis Type Summary Bottom-up Caller/Callee Top-down Tree Platform grid.cpp

Grouping: Function / Call Stack

Function / Call Stack	Effective Time by Utilization			
	Idle	Ok	Ideal	
grid_intersect	0s	0.64s	4.440s	0s
sphere_intersect	0s	0.096s	2.662s	0s
grid_bounds_intersect	0s	0.022s	0.085s	0.307s
GdipDrawImagePointRect	0s	0.042s	0.077s	0.215s
shader	0s	0s	0.037s	0.076s
Selected 1 row(s):	0s	0.463s	1.864s	4.440s

Data Of Interest (CPU Metrics)

Viewing 4 of 30 selected stack(s)

- analyze\_locks.exe... - shade.cpp:139
- analyze\_locks.exe...yze\_locks.cpp:106
- analyze\_locks.exe...yze\_locks.cpp:173
- analyze\_locks.exe...partitioner.h:257
- analyze...
- tbb.dll

Thread

- TBB Worker Thread (T...
- TBB Worker Thread (T...
- TBB Worker Thread (T...
- thread\_video (TID: 164...
- WinMainCRTStartup (...
- func@0x1800028c0 (TI...

CPU Usage

No filters are applied. Any Process Any Thread Any Module Utilization: A

Call Stack Mode: User functions + 1 Inline Mode: on Loop Mode: Functions only

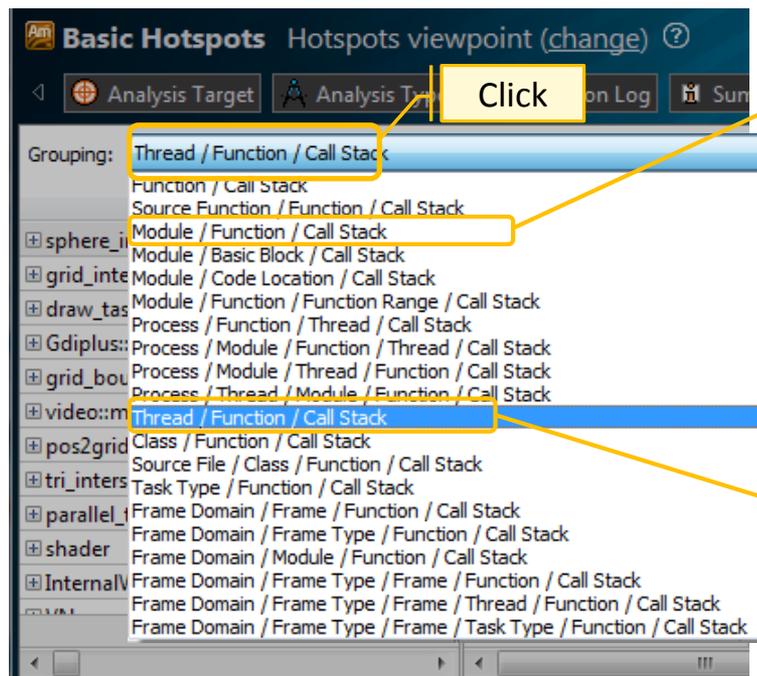
16

# Result Analysis

## GUI Concepts

### Groupings

- Each analysis type has many viewpoints
- Each viewpoint has pre-defined groupings
- Allows you to analyze the data in different hierarchies and granularities



Grouping: Module / Function / Call Stack

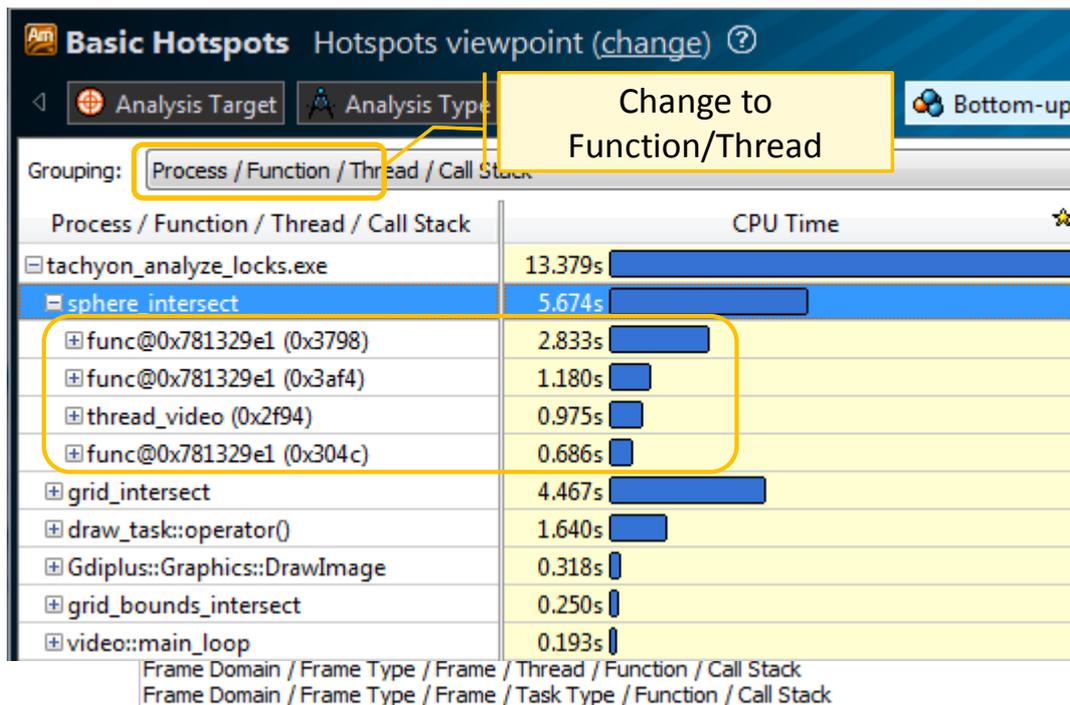
Module / Function / Call Stack	CPU Time
tachyon_analyze_locks.exe	13.367s
sphere_intersect	5.674s
grid_intersect	5.674s
grid_intersect	4.467s
intersect_objects	4.053s
grid_intersect	0.414s
draw_task::operator()	1.640s

Grouping: Thread / Function / Call Stack

Thread / Function / Call Stack	CPU Time
func@0x781329e1 (0x3798)	5.983s
func@0x781329e1 (0x3af4)	2.598s
thread_video (0x2f94)	2.384s
sphere_intersect	0.975s
grid_intersect	0.975s
intersect_objects	0.957s
shader	0.625s
trace	0.333s

# Viewpoints and Groupings

For example, pre-defined groupings can be used to determine load imbalance





*ITAC: getting started*



# Intel® Trace Analyzer and Collector

ITAC may be applied without touching the program or environment. One way to get a first trace is:

```
$ mpirun -trace -n <nprocs> ./test.x
```

Alternatively, just set the preload library and run without the `-trace` flag:

```
$ export LD_PRELOAD=libVT.so  
$ mpirun -f <hostfile> -n <nprocs> ./test.x
```

this is actually what the flag does internally. This methodology may be applied to situations with complex run scripts not knowing where the `mpirun` is actually executed.

Note: this does not work for statically linked Intel® MPI (not recommended).

# Viewing the trace file

ITAC will generate several files inside the directory where you started mpirun. Just start traceanalyzer in this directory:

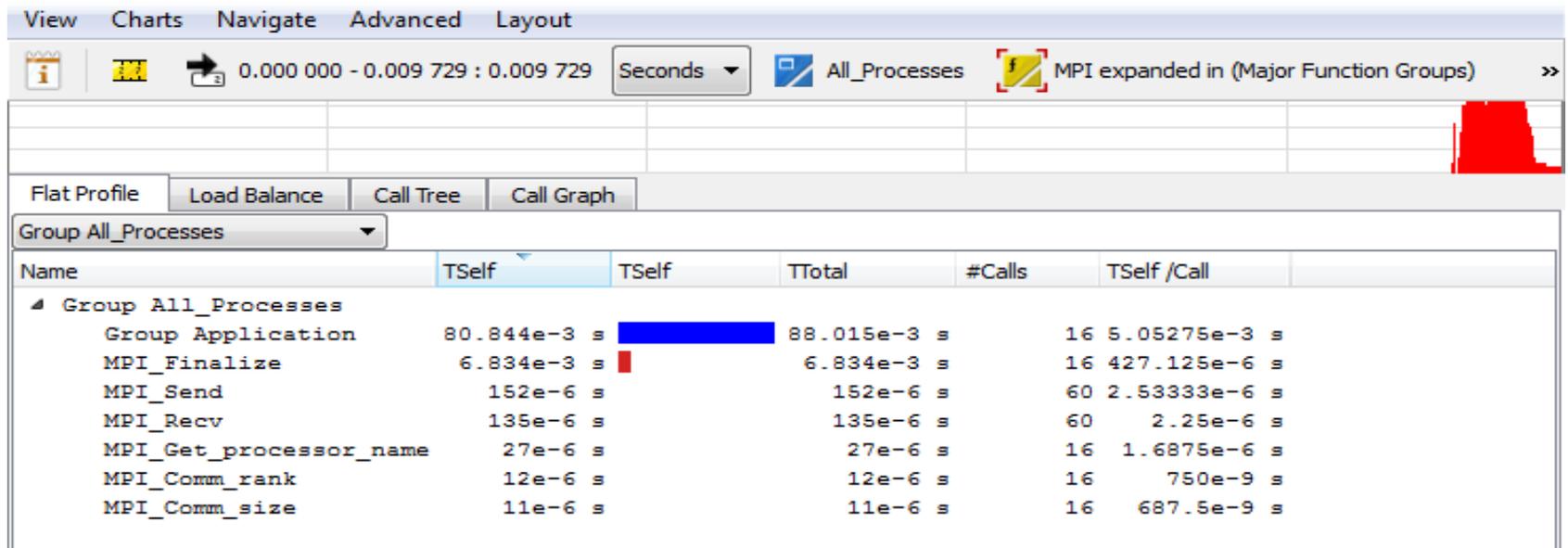
```
$ traceanalyzer test.x.stf
```

Alternatively there is a Windows version of traceanalyzer contained in the Linux ICS package.

# ITAC Function Profile

After starting ITAC a window showing a basic timing profile for MPI and Application will be displayed. Right click on the red **MPI** bar to show the profiling for each used MPI routine:

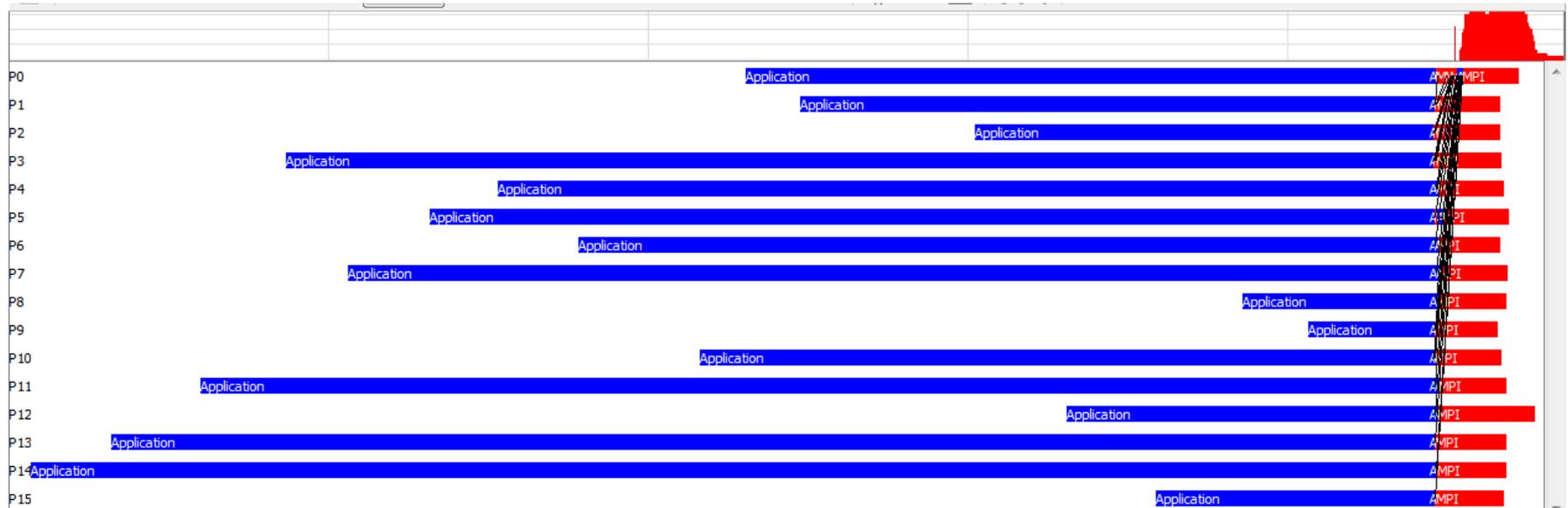
ungroup MPI



# ITAC Event Timeline

Most important view of ITAC is the Event Timeline. This shows the temporal development of MPI routines and messages:

Charts -> Event Timeline



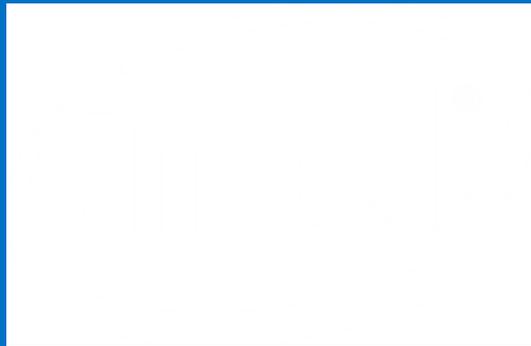
# ITAC MPI Correctness Checker

Correctness Checker validates MPI correctness. It uses another library but may be started like the ordinary ITAC:

```
$ mpirun -check -n <nprocs> ./test.x
```

or

```
$ export LD_PRELOAD=libVTmc.so  
$ mpirun -n <nprocs> ./test.x
```

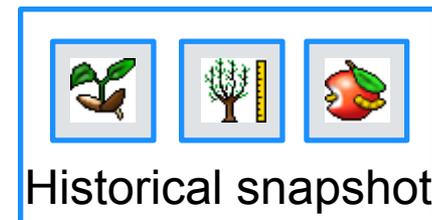
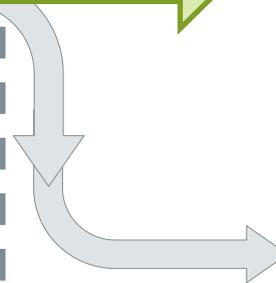




## *Vectorization Advisor: Backup*



# Snapshot concept



User makes  
a snapshot



# Command Line: Intel® Advisor XE

## Collect

```
advixe-cl --collect STEP --project-dir PROJ EXE
```

## Report

```
advixe-cl --report STEP --project-dir PROJ
```

Where **PROJ** is your Intel Advisor XE project

**STEP** is the specific type of report or collection you are trying to run. Could be: survey, tripcounts, map, dependencies

Need helps? -

```
advixe-cl --help
```

```
advixe-cl --help report
```

# Command Line: Intel® Advisor XE

## Collecting survey and tripcounts

```
advixe-cl --collect survey --project-dir ./advi -- mult.exe
```

```
advixe-cl --collect tripcounts --project-dir ./advi mult.exe
```

## Creating snapshot in command line, e.g:

```
advixe-cl --snapshot --project-dir ./advi --pack --cache-sources --  
cache-binaries -- /tmp/new_snapshot
```

## Viewing the results

```
advixe-gui ./advi
```

```
advixe-cl --report survey --project-dir ./advi
```

# Important notes

## Reporting (exporting) spreadsheet data to csv/xml/txt : example

```
advixe-cl --report survey -show-all-columns -format=csv --  
project-dir ./advi
```

- - Csv file will be created automatically and location will be reported

## Enabling experimental features (e.g. FLOPs)

```
$ export ADVIXE_EXPERIMENTAL=FLOPS
```

# Running Intel Advisor XE on a cluster

```
mpirun -n 10 advixe-cl -collect survey --project-dir ./my_proj  
./your_app
```

```
mpirun -n R advixe-cl -collect survey --project-dir ./my_proj  
./your_app : -np 9 ./your_app
```

Intel MPI-specific:

```
mpirun -n N -gtool "advixe-cl -collect STEP  
C:\myadvisor:R,R,R" mult.exe
```

Where STEP is the type of collection, N is the number of processes and R are ranks you want to run.

# Running dependency analysis using the command-line

1) First run a survey to get the ID of the loop you want to analyze

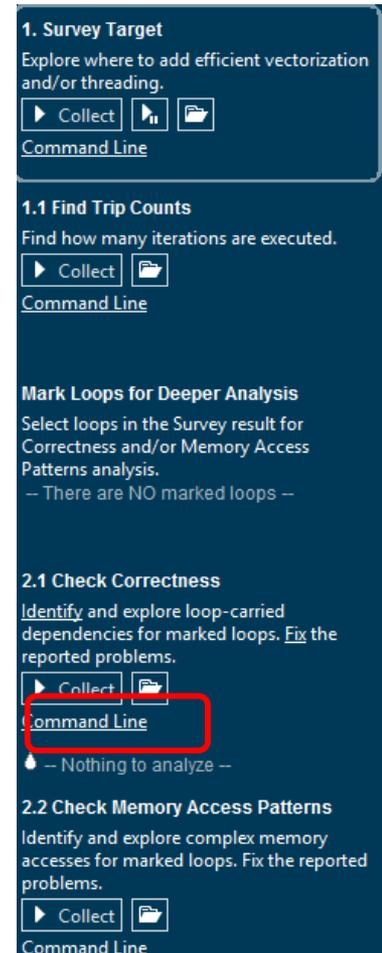
2) Then run

**advixe-cl** -collect dependencies

-mark-up-list ID -project-dir C:\myadvisor mult.exe

3) To display the results you can use the GUI or the command-line

**advixe-gui** C:\myadvisor



# Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2015v, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

## 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