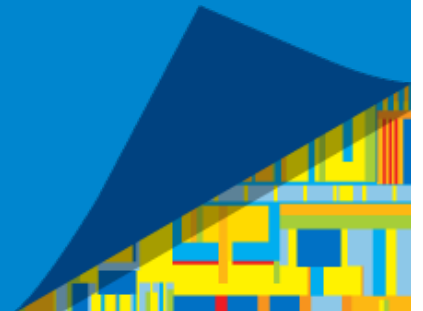




Vectorization Advisor: getting started



Before you analyze

Run GUI or Command Line

Set-up environment

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

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

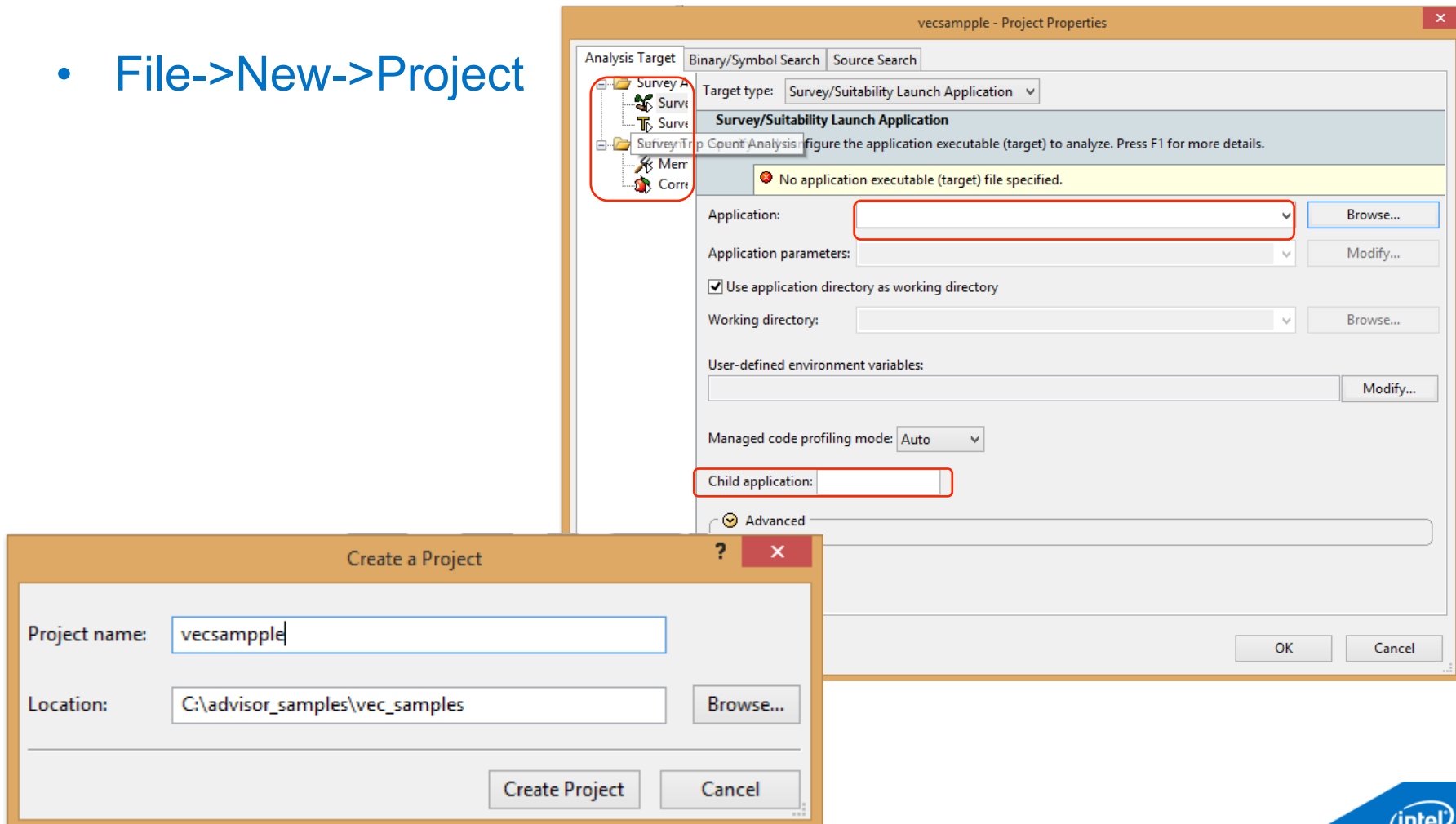
Min compilation switches (ICC example):

-O2 -g -xHost

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

1.1 Find Trip Counts
Find how many iterations are executed.
▶ Collect

Mark Loops for Deeper Analysis
Select loops for deeper analysis. Correctness, Patterns and more. There are 10 loops marked for analysis.

2.1 Check Loop Dependencies
Identify and explore complex loop dependencies for marked loops. Fix the reported problems.

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

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	
						Vector Length	Compiler Estimated Gain	Traits	Data Types	
Identify and explore complex loop dependencies for marked loops. Fix the reported problems.	1 Assumed ...	14.030s	14.030s	Scalar	vector dependence ...					Float64
reported problems.		0.985s	15.015s	Scalar	outer loop was not a ...					Float64
▶ Collect		0.000s	15.035s	Scalar	loop with function c...					Float64

Same approach for trip counts

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

▶ Collect

Command Line

Click Collect

1.1 Find Trip Counts
Find how many iterations are executed.

▶ Collect

Command Line

Trip Counts				
Median	Min	Max	Call Count	Iteration Duration
50	50	50	101000000	< 0.0001s
101	101	101	1000000	< 0.0001s
1000000	1000000	1000000	1	< 0.0001s

Mark Loops for Deeper Analysis
Select loops in the Survey result for

Correct Pattern - The

2.1 Check Dependencies
Identify 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

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 [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 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 [Icons]
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 [Icons]
Command Line

Check memory access patterns in your application

Summary Survey Report Refinement Reports Animation 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	

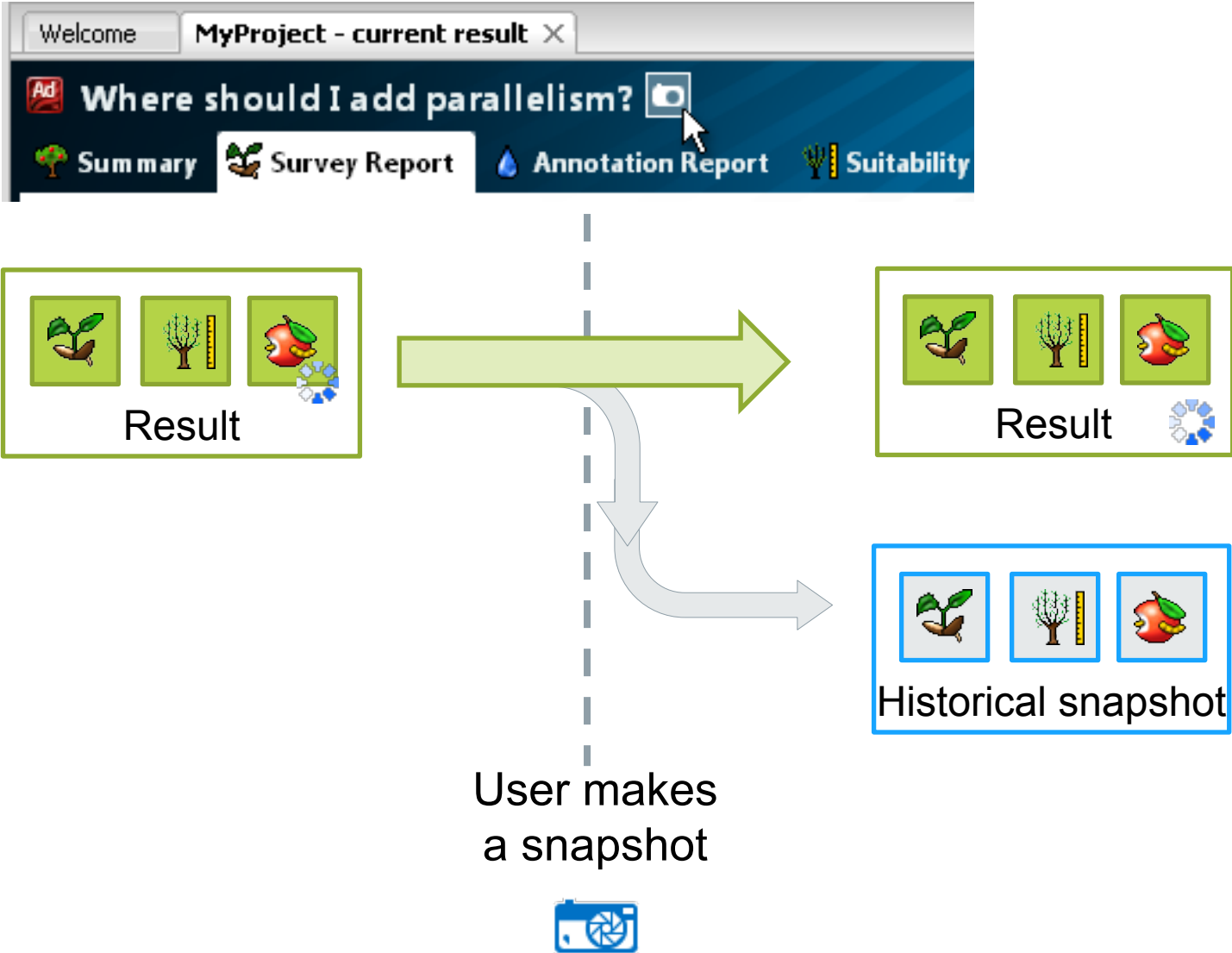
```

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

```

Click Collect

Snapshot concept



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

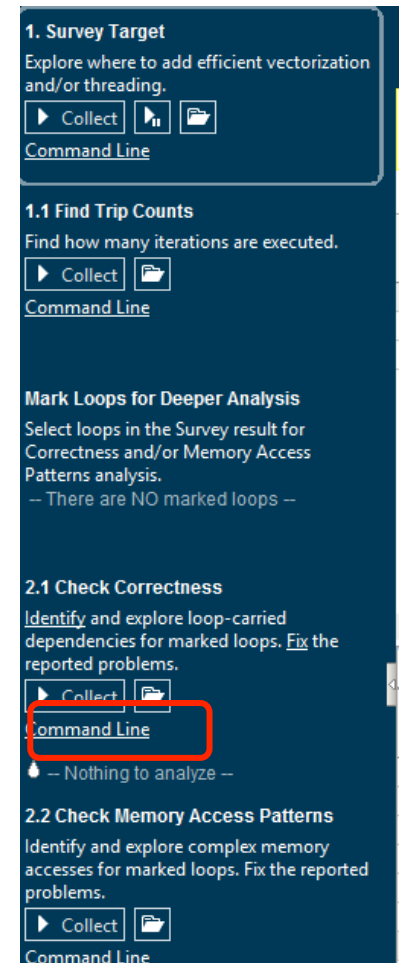
2) Then run

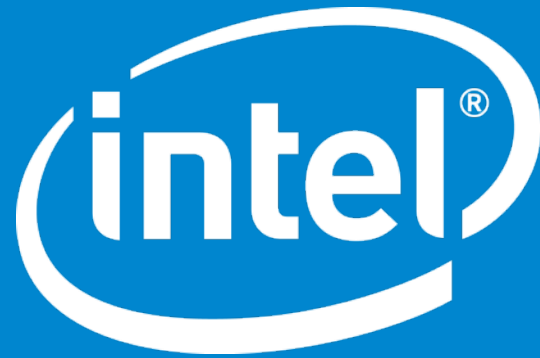
advixe-cl –collect dependencies

–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