

UF Research Computing: Overview Programming tools



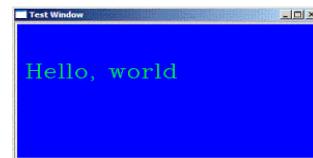
Ying Zhang
yingz@ufl.edu
4/2/2015

UF | Research Computing
Information Technology
Home of High-Performance Computing and **HiPerGator**

UF | Information Technology www.it.ufl.edu

What Is Computer Program

- ▶ A sequence of instructions
 - Code
- ▶ Written in a computer language
 - C, Fortran, Java, or Python
- ▶ Perform specified tasks
 - Algorithm



UF | Information Technology www.it.ufl.edu

Programming Environment

- ▶ Software setup for writing, compiling and executing programs
 - Compilers
 - C, C++, Fortran, etc
 - Interpreter
 - Matlab, Perl, Python, PHP, etc.
 - Other tools
 - Debuggers
 - Profilers
 - Performance analyzers

UF | Information Technology www.it.ufl.edu

Serial VS Parallel

- ▶ Serial Computing
 - Instructions are executed sequentially
 - On a single processor
- ▶ Parallel Computing
 - A problem is broken into independent components and can be solved concurrently
 - Each Components execute simultaneously on different processor or threads



UF | Information Technology www.it.ufl.edu

Programming Tools at RC

- ▶ Compilers
- ▶ Debuggers
- ▶ Performance profilers



UF | Information Technology www.it.ufl.edu

Compilers

- ▶ Compilers available
 - Intel Compiler Suite
 - Online documentation <https://software.intel.com/en-us/articles/new-user-compiler-basic-usage>
 - Modules
 - module spider intel
 - GNU Compiler Collection
 - Online documentation <https://gcc.gnu.org/onlinedocs/gcc-4.7.4/gcc/>
 - Modules
 - Module spider gcc



UF | Information Technology www.it.ufl.edu

Intel Compiler Suite

- For serial programs
 - `module load intel/2013`
- Usage
 - Compile C code: `icc`
 - e.g.: `icc -O1 mytest.c -o mytest`
 - Compile C++ code: `icpc`
 - e.g.: `icpc -g mytest.cpp -o mytest`
 - Compile Fortran code: `ifort`
 - e.g.: `ifort -O2 mytest.f90 -o mytest`

UF | Information Technology

www.it.ufi.edu

Intel Compiler Suite

- For MPI parallel programs
 - Modules
 - `module load intel/2013 openmpi/1.6.5`
- Usage
 - Compile C code: `mpicc`
 - e.g.: `mpicc -O1 mytest.c -o mytest`
 - Compile C++ code: `mpicxx`
 - e.g.: `mpicxx -g mytest.cpp -o mytest`
 - Compile Fortran code: `mpif77, mpif90`
 - e.g.: `mpif90 -O2 mytest.f90 -o mytest`
- For OpenMP programs
 - `module load intel/2013`
 - `icc/icpc/ifort -openmp mytest.c/.cpp/.f90 -o mytest`

UF | Information Technology

www.it.ufi.edu

Intel Compiler Suite

- Basic compiler options
 - `-g`: debugging
 - `-O#`: optimization
 - `-O0`: no optimization
 - `-O1`: optimize for speed without increase code size
 - `-O2`: default, optimize for speed, e.g. vectorization
 - `-O3`: high-level optimizer, prefetch, loop unrolling
- Support for target: SSE2, SSE3, SSE4.1, SSE4.2, AVX
 - `-xtarget`: for Intel architecture only
 - `-mtarget`: for any architecture supporting target, recommended on HiPerGator

SSE: Streaming SIMD Extensions; AVX: Advanced Vector Extension

UF | Information Technology

www.it.ufi.edu

Intel Compiler Suite

Compiler Based Vectorization

| Feature | Extension |
|---|-------------------------|
| Intel® Streaming SIMD Extensions 2 (Intel® SSE2) as available in initial Pentium® 4 or compatible non-Intel processors | sse2 |
| Intel® Streaming SIMD Extensions 3 (Intel® SSE3) as available in Pentium® 4 or compatible non-Intel processors | sse3 |
| Supplemental Streaming SIMD Extensions 3 (SSSE3) as available in Intel® Core™2 Duo processors | ssse3 |
| Intel® SSE4.1 as first introduced in Intel® 45nm Hi-K next generation Intel Core™ micro-architecture | sse4.1 |
| Intel® SSE4.2 Accelerated String and Text Processing instructions supported first by Intel® Core™ i7 processors | sse4.2 |
| Extensions offered by Intel® ATOM™ processor : Intel® SSSE3 (I1) and MOVBE instruction | sse3_atom |
| Intel® Advanced Vector Extensions (Intel® AVX) as available in 2nd generation Intel Core processor family – code name Sandy Bridge | AVX |
| Intel® Advanced Vector Extensions (Intel® AVX) as code-name Ivy Bridge and in code-name Haswell (available only in compilers v13+, fall 2012) | CORE-AVX-I CORE-AVX2 |

UF | Information Technology

www.it.ufi.edu

GNU Compiler Collection (GCC)

- For serial programs
 - Module
 - `module load gcc/4.7.2`
 - Usage
 - Compile C code: `gcc`
 - e.g.: `gcc -O1 mytest.c -o mytest`
 - Compile C++ code: `g++`
 - e.g.: `g++ -g mytest.cpp -o mytest`
 - Compile Fortran code: `gfortran`
 - e.g.: `gfortran -O2 mytest.f90 -o mytest`

UF | Information Technology

www.it.ufi.edu

GNU Compiler Collection

- For MPI parallel programs
 - Modules
 - `module load gcc/4.7.2 openmpi/1.6.5`
 - Usage
 - Compile C code: `mpicc`
 - e.g.: `mpicc -O1 mytest.c -o mytest`
 - Compile C++ code: `mpicxx`
 - e.g.: `mpicxx -g mytest.cpp -o mytest`
 - Compile Fortran code: `mpif77, mpif90`
 - e.g.: `mpif90 -O2 mytest.f90 -o mytest`
- For OpenMP programs
 - Module
 - `module load intel/2013`
 - Usage
 - `gcc/g++/gfortran -fopenmp mytest.c/.cpp/.f90 -o mytest`

UF | Information Technology

www.it.ufi.edu

GNU Compiler Collection

- ▶ Basic compiler options
 - **-g**: debugging
 - **-O#**: optimization
 - **-O0**: default, reduce compilation time
 - **-O1**: reduce code size and execution time
 - **-O2**: default, optimize for speed
 - **-O3**: O2 + prefetching, loop unrolling, vectorization, etc.
 - **-Ofast**: O3 + ffast-math and other Fortran related optimization
 - Support for: SSE2, SSE3, SSE4.1, SSE4.2, AVX
 - **-mfma=cpu-type**: generic, native, corei7, core-avx-i ...
 - **-msse**, **-msse2**, **-msse3**, **-msse4.1**, **-mavx** ...

UF | Information Technology

www.it.ufl.edu

Debuggers

- ▶ Debugging
 - Locate and remove errors or abnormalities
- ▶ Debugger available on HiPerGator
 - Allinea DDT
 - Intel Inspector XE
 - Valgrind



UF | Information Technology

www.it.ufl.edu

Allinea DDT

- ▶ A graphical distributed debugging tool
 - Serial code
 - Multithreaded code
 - Multiprocess code
 - C, C++, Fortran, CUDA
- ▶ Module required
 - **module load ddt**
 - Latest version: 5.0



UF | Information Technology

www.it.ufl.edu

Allinea DDT

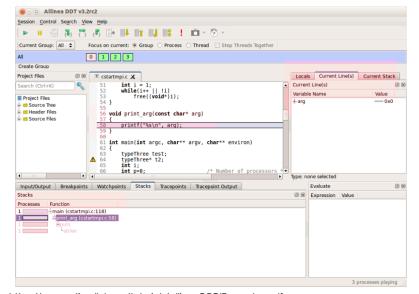
- ▶ Flow control
 - Control program progress
 - Static analysis at thread and process level
 - Identify and fix problems
- ▶ Data monitoring
 - Track variables
 - Detect memory errors
 - Check data calculation
 - Trace points

UF | Information Technology

www.it.ufl.edu

Allinea DDT

Flow control and data monitoring

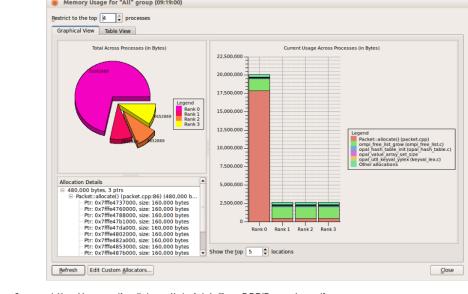
Source: <https://computing.llnl.gov/tutorials/allineaDDT/Examples.pdf>

UF | Information Technology

www.it.ufl.edu

Allinea DDT

Memory usage analysis

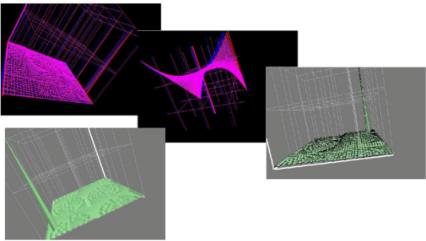
Source: <https://computing.llnl.gov/tutorials/allineaDDT/Examples.pdf>

UF | Information Technology

www.it.ufl.edu

Allinea DDT

Multi-Dimensional Array Viewer



Source: <https://www.allinea.com/user-guide/forge/ViewingVariablesAndData.html>

UF | Information Technology www.it.ufl.edu

Valgrind

- ▶ A memory mismanagement detector
- ▶ Detect
 - Memory leak
 - Deallocation errors
 - Use uninitialized memory
 - Read/Write freed memory
 - Mismatched allocation/deallocation and syntax
 - malloc/new/new[] vs free/delete/delete[]
- ▶ Cons
 - No bound checking for static arrays
 - Execution specific

UF | Information Technology www.it.ufl.edu

Valgrind



- ▶ Required modules
 - module spider valgrind
 - For serial programs
 - module load gcc/4.7.2 valgrind
 - For parallel programs
 - module load intel/2013 openmpi/1.6.5 valgrind
 - module load gcc/4.7.2 openmpi/1.6.5 valgrind
- ▶ Usage
 - For serial programs,
 - e.g.: valgrind --leak-check=yes myprog input
 - For parallel programs
 - e.g.: mpirun -np 4 valgrind --leak-check=yes myprog input

UF | Information Technology www.it.ufl.edu

Valgrind

- ▶ Example


```
#include <stdlib.h>
void f(void)
{
    int *x = malloc(10 * sizeof(int));
    x[10] = 0;           // problem 1: heap block overrun
}
int main(void)
{
    f();
    return 0;
}
```
- ▶ Valgrind output


```
==19182== Invalid write of size 4
==19182==   at 0x804838F: f (example.c:6)
==19182==   by 0x80483AB: main (example.c:11)
==19182== Address 0x1BA45050 is 0 bytes after a block of size 40 alloc'd
==19182==  by 0x1BFF5FCD: malloc (vg_replace_malloc.c:130)
==19182==  by 0x8048385: f (example.c:5)
==19182==  by 0x80483AB: main (example.c:11)
```

Source: <http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare>

UF | Information Technology www.it.ufl.edu

Performance Profiling and Evaluation

- ▶ Dynamic program analysis that reveals features in the program and interactions between software and hardware
 - The frequency and duration of function calls & loops
 - Calculation speed
 - Memory usage
 - Communication pattern
 - Multithreading analysis
 - Load balance and efficiency
- ▶ Goal:
 - Code optimization

UF | Information Technology www.it.ufl.edu

Performance Profiling & Evaluation Tools

- ▶ Allinea MAP
 - module load ddt
- ▶ TAU
 - module load intel/2013 tau
 - module load intel/2013 openmpi/1.6.5 tau
- ▶ Intel Toolkit
 - VTune Amplifier XE
 - Advisor XE

UF | Information Technology www.it.ufl.edu

Allinea MAP

- ▶ A graphical distributed profiler
 - Detect bottlenecks at source level
 - Support
 - Serial, multithreaded, and multiprocess code
 - C, C++, Fortran 90
 - Performance metrics
 - Memory usage
 - Floating Point calculations
 - MPI call performance
 - Easy to use, no instrumentation or recompilation

UF | Information Technology www.it.ufl.edu

The screenshot shows a graphical user interface for the Allinea MAP profiler. It includes a timeline at the top, a memory usage analysis section with a bar chart, and a detailed source code analysis window showing MPI calls and their performance metrics.

UF | Information Technology www.it.ufl.edu

TAU

- ▶ Tuning and Analysis Utilities
- ▶ Integrated Performance Toolkit
 - Performance profiling and tracing
 - Both serial and parallel programs
 - Source code instrumentation
 - Adds probes to collect performance data
 - Manual or automatic
 - Comprehensive data analysis and management
 - Include hardware performance counters via PAPI
 - Performance data visualization
 - Paraprof
 - Open source
 - <http://www.cs.oregon.edu/research/tau/home.php>

UF | Information Technology www.it.ufl.edu

TAU

- ▶ Performance analysis
 - Time consumption – identify bottlenecks
 - At source level: loops, functions calls
 - I/O, communication, computation
 - Memory usage
 - Allocation/deallocation, cache misses
 - Floating point operations
 - FLOP rate
 - Cross process comparisons
 - Load balancing and scaling

UF | Information Technology www.it.ufl.edu

TAU

- ▶ Identify potential bottlenecks

The screenshot displays a call graph titled "TAU INPUT: NEMO-3D-LINEAR-TAU". It shows various function calls and their execution times, with a color scale indicating time consumption. The legend indicates values from 0.000 to 100.000 seconds.

UF | Information Technology www.it.ufl.edu

TAU

- ▶ How much time is spent in each routine?

The screenshot shows a horizontal bar chart titled "Value: Exclusive". It lists various routines and the time spent in them, such as LEO_JKSWEEP1 (9647.318), LEO_BIGVECT (435.713), and so on. The legend indicates values from 0 to 2600+ seconds.

UF | Information Technology www.it.ufl.edu

TAU

- Program call graph

Source: Sameer Shende, TAU Performance System Tutorial

UF | Information Technology www.it.ufl.edu

TAU ParaProf 3D Profile Browser

Source: Sameer Shende, TAU Performance System Tutorial

UF | Information Technology www.it.ufl.edu

Intel Parallel Studio

| Phase | Productivity Tool | Feature | Benefit |
|--------------------------|-----------------------------------|--|--|
| Advanced Parallel Design | Intel® Advisor XE | Analyze existing code base and find opportunities for parallelization. | Easier analysis and performance heuristics, find compute hotspots and check for parallelization strategies. |
| Advanced Build and Debug | Intel® Composer XE | C/C++ and Fortran compilers, performance libraries, and parallel models | Application performance, scalability and quality for current multicore and future many-core systems. |
| Advanced Verify | Intel® Inspector XE | Memory & threading error checking tool for higher code reliability & quality | Increases productivity and lowers cost, by catching memory and threading defects early |
| Advanced Tune | Intel® VTune™ Amplifier XE | Performance Profiler to optimize performance and scalability | Removes guesswork, saves time, makes it easier to find performance and scalability bottlenecks. Combines ease of use with deeper insights. |

Source: Intel Parallel Studio Webinar

UF | Information Technology www.it.ufl.edu

Intel Inspector XE

- Graphical software “Correctness Analyzer”
- Memory Error
 - Memory leaks
 - corruptions
 - allocation/deallocation
 - Inconsistent memory usage,
 - Illegal memory access
 - Uninitialized memory read
- Threading error
 - Data races
 - Deadlocks

UF | Information Technology www.it.ufl.edu

Intel Inspector XE

- Key features

| Feature | Details |
|---------------------|--|
| Analyses | <ul style="list-style-type: none"> Dynamic Memory Analysis Dynamic Threading Analysis |
| GUI | <ul style="list-style-type: none"> Microsoft Visual Studio IDE integration (2008, 2010, or 2012) Stand alone GUI on both Windows and Linux |
| Compilers supported | <ul style="list-style-type: none"> Microsoft® Visual® C++ .NET® Intel® Parallel Composer and Intel® Composer XE Gcc |
| OS | <ul style="list-style-type: none"> Windows XP, Vista, 7, 8 Linux (various distros) |
| Languages | <ul style="list-style-type: none"> C/C++ C# Fortran |

Source: Intel Parallel Studio Webinar

UF | Information Technology www.it.ufl.edu

Intel Inspector XE

Double click on Problem to navigate to source (next slide)

Powerful filtration feature

Code locations grouped into Problems to simplify results management

Source: Intel Parallel Studio Webinar

UF | Information Technology www.it.ufl.edu

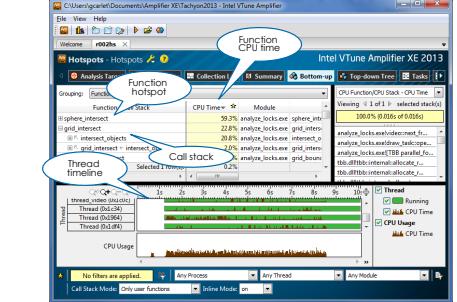
Intel VTune Amplifier XE

- Performance analyzer
- Bottleneck identification
- Source level performance data
- Thread profiling
- Hardware event based sampling
- Support
 - C/C++, Fortran, Java, .NET
 - GCC, Intel Compiler, Windows
 - Serial and parallel programs
 - Linux and Windows



UF | Information Technology www.it.ufi.edu

Intel VTune Amplifier XE

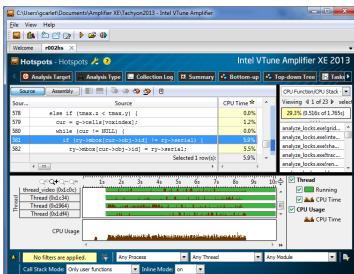


The screenshot shows the Intel VTune Amplifier XE interface. The main window displays a 'CPU Function CPU Stack' view. A call stack is highlighted with a yellow box, and a 'Function CPU Time' panel is shown above it. Below the stack, a 'Thread timeline' chart shows CPU usage over time for multiple threads. The interface includes various tabs like 'Hotspots', 'Analysis Target', 'Collection Log', 'Summary', 'Bottom-up', 'Top-down Tree', and 'Tasks'. A status bar at the bottom indicates 'Source: Intel Parallel Studio Webinar'.

UF | Information Technology www.it.ufi.edu

Intel VTune Amplifier XE

Source view – identify hotspots



The screenshot shows the 'Source view – identify hotspots' feature of Intel VTune Amplifier XE. It displays assembly code for a function named 'main' and a corresponding CPU usage chart below it. The chart shows CPU time distribution across multiple threads. The interface includes tabs for 'Hotspots', 'Analysis Target', 'Collection Log', 'Summary', 'Bottom-up', 'Top-down Tree', and 'Tasks'. A status bar at the bottom indicates 'Source: Intel Parallel Studio Webinar'.

UF | Information Technology www.it.ufi.edu

Intel Advisor XE

- Threading design and prototyping
 - Analyze, design, tune the threading design in multithreaded program
 - Identify the code location for parallelism
 - Identify synchronization errors
 - Predict threading errors and scaling
 - Separate design and implementation
- Support
 - C, C++, Fortran, C#
 - Linux, Windows

UF | Information Technology www.it.ufi.edu

Intel Advisor XE

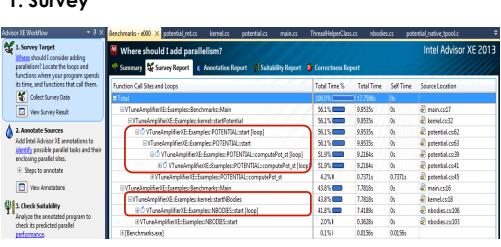
- Typical workflow
 - Survey
 - Add annotations
 - Model performance suitability
 - Check correctness
 - Add parallel framework



UF | Information Technology www.it.ufi.edu

Intel Advisor XE

1. Survey



The screenshot shows the 'Survey Report' section of Intel Advisor XE. It displays a table of benchmarks and their execution times. The table includes columns for 'Total Time %', 'Self Time', and 'Source Location'. Several rows are highlighted with red boxes, specifically pointing to 'VtuneApplfile1.Examples.Benchmark.Main', 'VtuneApplfile1.Examples.POTENTIAL.main', and 'VtuneApplfile1.Examples.POTENTIAL.computePhi_j'. A status bar at the bottom indicates 'Source: Intel Parallel Studio Webinar'.

UF | Information Technology www.it.ufi.edu

Intel Advisor XE

2. Add Annotation

The screenshot shows a C++ code editor with annotations like '#pragma omp parallel' and '#pragma omp for'. A context menu is open over the code, showing options like 'Annotation Wizard...', 'Annotate Site...', 'Annotate Task...', 'Annotate Data Loop Site...', 'Annotate Data Task...', 'Annotate Task Site...', 'Annotate Task End...', and 'Annotate Site End...'. An orange arrow points from the menu towards the code.

Source: Intel Parallel Studio Webinar
UF | Information Technology www.it.ufl.edu

Intel Advisor XE

3. Add Annotation

The screenshot shows the 'Estimated Overall Speed-up' report. It includes a graph of 'Scalability of Maximum Site Gain' and a table of 'Maximum Program Gain For All Site'. A blue box highlights 'Recommended Improvement' with options like 'Reduce Task Unbalance', 'Reduce Task Duplication', and 'Reduce Task Shifting'. Another blue box highlights 'Scalability Graph'.

Source: Intel Parallel Studio Webinar
UF | Information Technology www.it.ufl.edu

Intel Advisor XE

4. Check correctness

The screenshot shows the 'Correctness Report' with a section titled '4 Memory reuse conditions found!'. It lists memory reuse issues with annotations like '#pragma omp parallel' and '#pragma omp for'. A blue box highlights 'Observations help identify problem'.

Analyze your design for errors

Source: Intel Parallel Studio Webinar
UF | Information Technology www.it.ufl.edu

Intel Advisor XE

5. Add parallel framework

The screenshot shows the 'Potential program gain': 1.12x (8 CPUs, Microsoft TPL Threading Model) report. It includes a graph of 'Scalability of Maximum Site Gain' and a table of 'Maximum Program Gain For All Site'. A red box highlights 'Step 1: Add Parallel Framework'.

Potential program gain': 2.41x (8 CPUs, Microsoft TPL Threading Model)

Source: Intel Parallel Studio Webinar
UF | Information Technology www.it.ufl.edu

UF Research Computing

- ▶ Help and Support
 - <https://my.it.ufl.edu>
 - For any kind of question or help requests
 - <http://wiki.rc.ufl.edu>
 - Documents on hardware and software resources
 - Various user guides
 - Many sample submission scripts
 - <http://rc.ufl.edu>
 - Frequently Asked Questions
 - Account set up and maintenance

UF | Information Technology www.it.ufl.edu

8