

Introduction to profiling

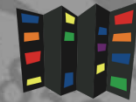
Martin Čuma

*Center for High Performance
Computing University of Utah
m.cuma@utah.edu*

- Profiling basics
- Simple profiling
- Open source profiling tools
- Intel development tools
 - Advisor XE
 - Inspector XE
 - VTune Amplifier XE
 - Trace Analyzer and Collector
- Interpreted languages profiling
- <https://www.surveymonkey.com/r/7PFVFCY>



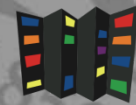
- Evaluate performance
- Find the performance bottlenecks
 - inefficient programming
 - memory I/O bottlenecks
 - parallel scaling



- Hardware counters
 - count events from CPU perspective (# of flops, memory loads, etc)
 - usually need Linux kernel module installed
- Statistical profilers (sampling)
 - interrupt program at given intervals to find what routine/line the program is in
- Event based profilers (tracing)
 - collect information on each function call

- Time program runtime
 - get an idea on time to run and parallel scaling
- Serial profiling
 - discover inefficient programming
 - computer architecture slowdowns
 - compiler optimizations evaluation
 - gprof
 - Trick how to get gprof to work in parallel:
<http://shwina.github.io/2014/11/profiling-parallel>

- Vendor based
 - AMD CodeAnalyst
- Community based
 - perf
 - hardware counter collection, part of Linux
 - oprofile
 - profiler
 - drawback – harder to analyze the profiling results



- HPC Toolkit
 - A few years old, did not find it as straightforward to use
- TAU
 - Lots of features, which makes the learning curve slow
- Scalasca
 - Developed by European consortium, did not try yet

- We have a 2 concurrent users license
- Tools for all stages of development
 - Compilers and libraries
 - Verification tools
 - Profilers
- More info

<https://software.intel.com/en-us/intel-parallel-studio-xe>

<https://www.chpc.utah.edu/documentation/software/intel-parallelXE.php>



- Intel Parallel Studio XE 2018 Cluster Edition
 - Compilers (C/C++, Fortran)
 - Distribution for Python
 - Math library (MKL)
 - Data Analytics Acceleration Library (DAAL)
 - Threading library (TBB)
 - Vectorization or thread design and prototype (Advisor)
 - Memory and thread debugging (Inspector)
 - Profiler (VTune Amplifier)
 - MPI library (Intel MPI)
 - MPI analyzer and profiler (ITAC)

- Serial and parallel profiler
 - multicore support for OpenMP and OpenCL on CPUs, GPUs and Xeon Phi
- Quick identification of performance bottlenecks
 - various analyses and points of view in the GUI
- GUI and command line use
- More info

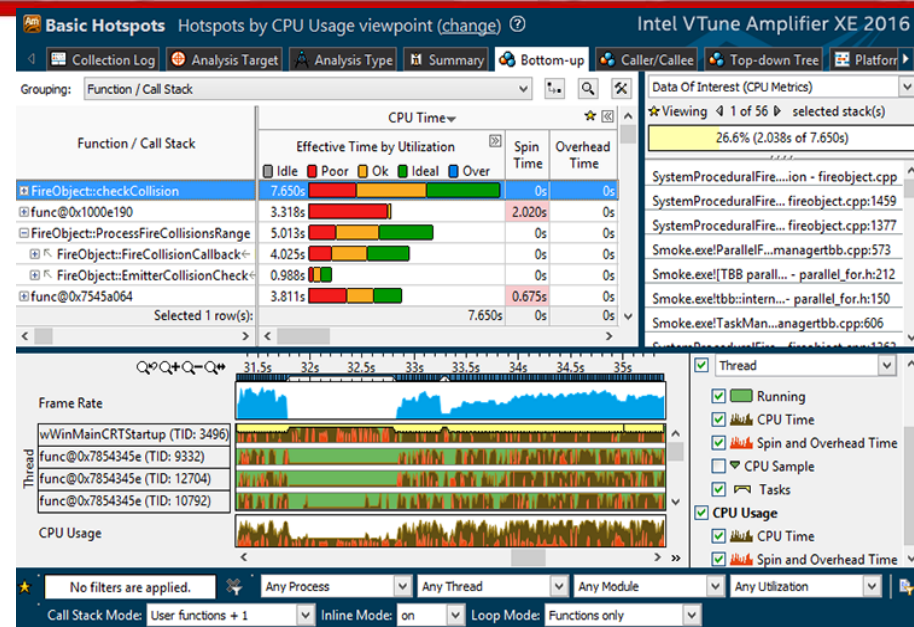
<https://software.intel.com/en-us/intel-vtune-amplifier-xe>

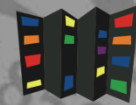
- Source the environment module load vtune
- Run VTune
`amplxe-gui` – graphical user interface
`amplxe-cl` – command line (best to get from the GUI)

Can be used also for remote profiling (e.g. on Xeon Phi)

- Tuning guides for specific architectures

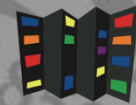
<https://software.intel.com/en-us/articles/processor-specific-performance-analysis-papers>





- Vectorization advisor
 - Identify loops that benefit from vectorization, what is blocking efficient vectorization and explore benefit of data reorganization
- Thread design and prototyping
 - Analyze, design, tune and check threading design without disrupting normal development
- More info

<http://software.intel.com/en-us/intel-advisor-xe/>



- Source the environment module load advisorxe

- Run Advisor
advixe-gui – graphical user interface
advixe-cl – command line (best to get from the GUI)
- Create project and choose appropriate modeling
- Getting started guide

<https://software.intel.com/en-us/get-started-with-advisor>

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

Summary Survey Report Refinement Reports Annotation Report Suitability Report

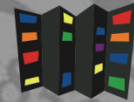
Elapsed time: 54.44s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops	
							Vecto...	Efficiency
[loop at stl_algo.h:4740 in std::tr ...]		0.170s	0.170s	12; 4	Scalar	non-vectorizable loop ins ...	AVX	~100%
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem ..	0.170s	0.170s	12	Vectorized (B		AVX	
[loop at loopstl.cpp:2449 in s ...]		0.020s	0.020s	4	Remainder			
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but ...		
[loop at loopstl.cpp:3509 in s2 ...]	1 High vector register ...	0.160s	0.160s	12	Expand	Expand	AVX	~69%
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem ..	0.150s	0.150s	125; 4	Expand	Expand	AVX	~96%
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX	~100%
[loop at stl_numeric.h:247 in std ...]	1 Assumed dependency ...	0.150s	0.150s	49	Scalar	vector dependence preve ...		



- MPI profiler
 - traces MPI code
 - identifies communication inefficiencies
- Collector collects the data and Analyzer visualizes them
- More info

<https://software.intel.com/en-us/intel-trace-analyzer>



- Source the environment

`module load itac`

- Using Intel compilers, can compile with `-trace`

`mpiiifort -openmp -trace trap.f`

- Run MPI code

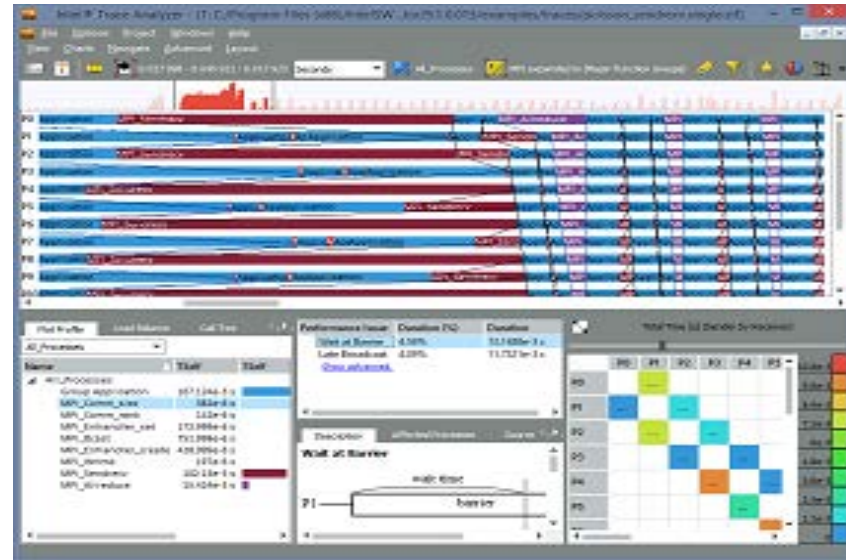
`mpirun -trace -n 4 ./a.out`

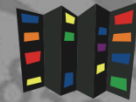
- Run visualizer

`traceanalyzer a.out.stf &`

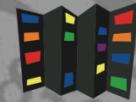
- CHPC site

<https://software.intel.com/en-us/get-started-with-itac-for-linux>

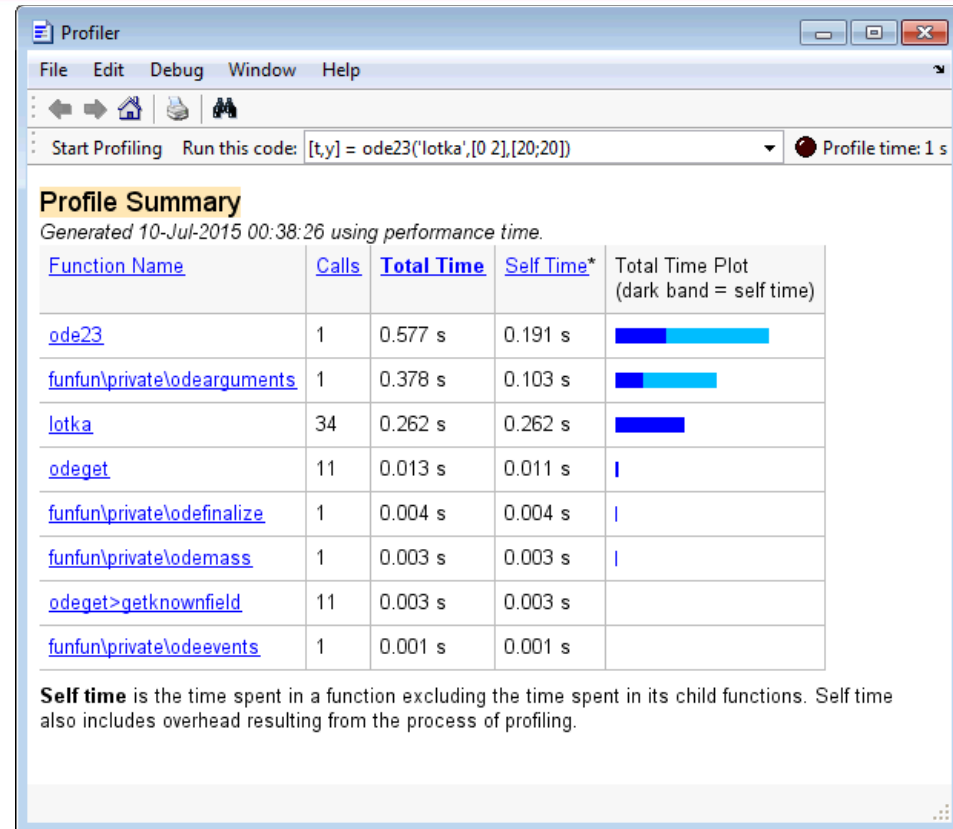




- With increased use of interpreted languages, their performance is becoming important
- Matlab
 - Profiling ecosystem in the IDE
- Python
 - Python modules or IDEs
- R
 - Profiling libraries or RStudio



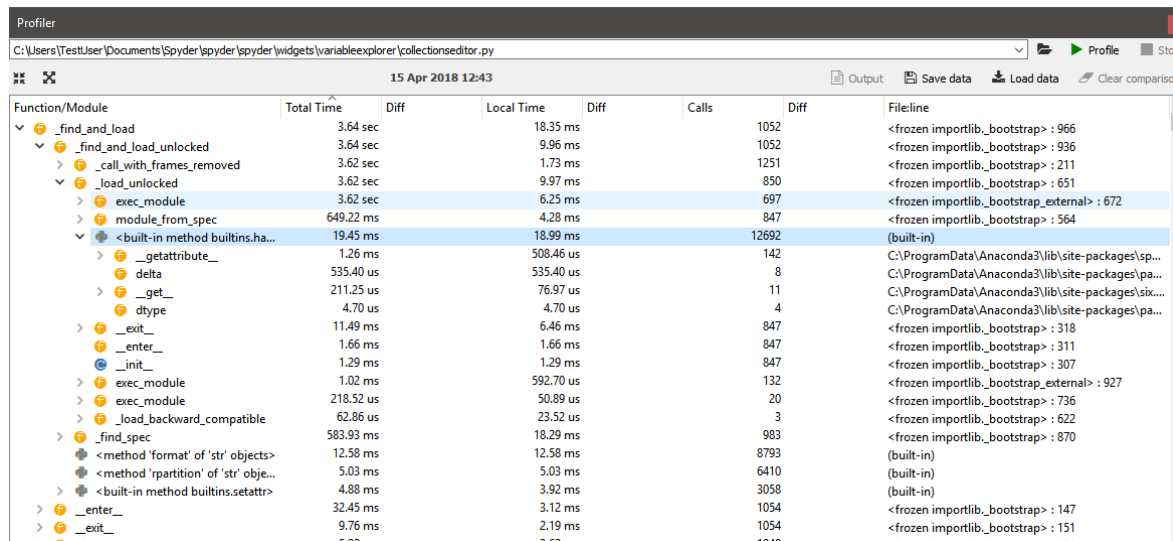
- `profile` command turns on/off profiling
- Profile is then displayed in the IDE
- Click on each function to show line-by-line profile



- Performance improvement strategies

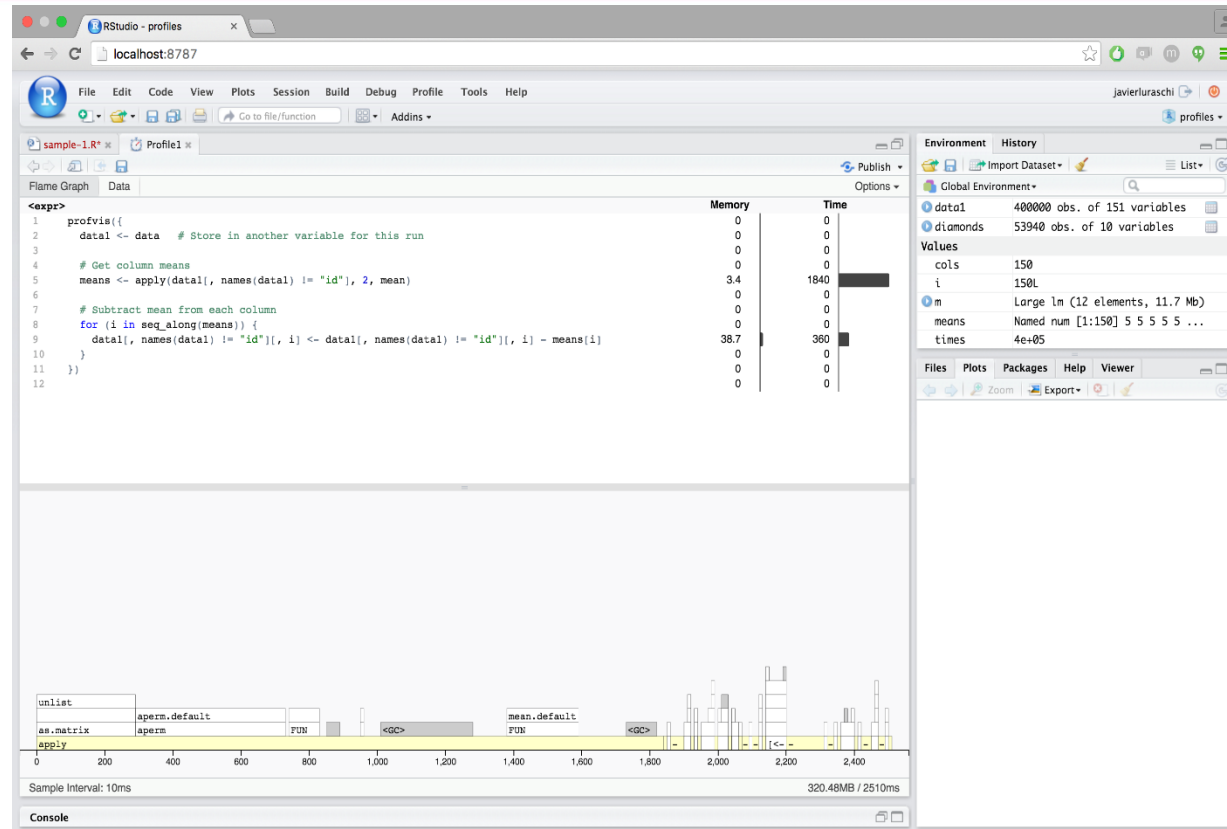
https://www.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html

- `profile` and `cProfile` modules
 - Text based output, optional format with `pstats`, analysis with `Stats`
- Plethora of other tools
 - E.g. line profiling with `line_profiler`
- Some IDEs display profiles
 - Spyder



Function/Module	Total Time	Diff	Local Time	Diff	Calls	Diff	Fileline
✓ <code>_find_and_load</code>	3.64 sec		18.35 ms		1052		<frozen importlib_bootstrap> : 966
✓ <code>_find_and_load_unlocked</code>	3.64 sec		9.96 ms		1052		<frozen importlib_bootstrap> : 936
✓ <code>_call_with_frames_removed</code>	3.62 sec		1.73 ms		1251		<frozen importlib_bootstrap> : 211
✓ <code>_load_unlocked</code>	3.62 sec		9.97 ms		850		<frozen importlib_bootstrap> : 651
✓ <code>exec_module</code>	3.62 sec		6.25 ms		697		<frozen importlib_bootstrap_external> : 672
✓ <code>module_from_spec</code>	649.22 ms		4.28 ms		847		<frozen importlib_bootstrap> : 564
✓ <code><built-in method builtins.ha...</code>	19.45 ms		18.99 ms		12692		(built-in)
✓ <code>_getattr__</code>	1.26 ms		508.46 us		142		C:\ProgramData\Anaconda3\lib\site-packages\sp...
✓ <code>delta</code>	535.40 us		535.40 us		8		C:\ProgramData\Anaconda3\lib\site-packages\pa...
✓ <code>_get__</code>	211.25 us		76.97 us		11		C:\ProgramData\Anaconda3\lib\site-packages\six...
✓ <code>dtype</code>	4.70 us		4.70 us		4		C:\ProgramData\Anaconda3\lib\site-packages\pa...
✓ <code>_exit__</code>	11.49 ms		6.46 ms		847		<frozen importlib_bootstrap> : 318
✓ <code>_enter__</code>	1.66 ms		1.66 ms		847		<frozen importlib_bootstrap> : 311
✓ <code>_init__</code>	1.29 ms		1.29 ms		847		<frozen importlib_bootstrap> : 307
✓ <code>exec_module</code>	1.02 ms		592.70 us		132		<frozen importlib_bootstrap_external> : 927
✓ <code>exec_module</code>	218.52 us		50.89 us		20		<frozen importlib_bootstrap> : 736
✓ <code>_load_backward_compatible</code>	62.86 us		23.52 us		3		<frozen importlib_bootstrap> : 622
✓ <code>_find_spec</code>	583.93 ms		18.29 ms		983		<frozen importlib_bootstrap> : 870
✓ <code><method 'format' of 'str' objects></code>	12.58 ms		12.58 ms		8793		(built-in)
✓ <code><method 'rpartition' of 'str' obje...</code>	5.03 ms		5.03 ms		6410		(built-in)
✓ <code><built-in method builtins.setattr></code>	4.88 ms		3.92 ms		3058		(built-in)
✓ <code>_enter__</code>	32.45 ms		3.12 ms		1054		<frozen importlib_bootstrap> : 147
✓ <code>_exit__</code>	9.76 ms		2.19 ms		1054		<frozen importlib_bootstrap> : 151
✓ <code>_exit__</code>	5.22 ms		2.62 ms		1048		<frozen importlib_bootstrap> : 136

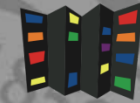
- Rprof function to profile
- summaryRprof to display
- RStudio has a profile interface called profviz



- Performance improvement strategies

<http://adv-r.had.co.nz/Profiling.html>

- Serial profilers
 - gprof, perf
- Intel tools
 - VTune, AdvisorXE, ITAC
- Interpreted languages profiling
 - Matlab profile
 - Python profile, Cprofile
 - R Rprof, profviz
- <https://www.surveymonkey.com/r/7PFVFCY>



- <https://www.surveymonkey.com/r/7PFVFCY>