# Tuning Python Applications Can Dramatically Increase Performance

**Vasilij Litvinov**

**Software Engineer, Intel**

# 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 © 2015, 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



Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Agenda

Why do we need Python optimization?

How one finds the code to tune?

Overview of existing tools

An example

Intel® VTune Amplifier capabilities and comparison

Q & A

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Why do we need Python optimization?

- Python is used to power wide range of software, including those where application performance matters
- Some Python code may not scale well, but you won't know it unless you give it enough workload to chew on
- Sometimes you are just not happy with the speed of your code

All in all, there are times when you want to make your code run faster, be more responsive, *(insert your favorite buzzword here)*.

So, you need to **optimize** (or tune) your code.

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# How one finds the code to tune – measuring vs guessing

- **Hard stare** = Often wrong

- **Logging** = Analysis is tedious

- **Profile** = Accurate, Easy

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Not All Profilers Are Equal

There are different profiling techniques, for example:

- **Event-based**
  - Example: built-in Python cProfile profiler

- **Instrumentation-based**
  - Usually requires modifying the target application (source code, compiler support, etc.)
  - Example: line_profiler

- **Statistical**
  - Accurate enough & less intrusive
  - Example: vmstat, statprof

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Optimization Notice

intel

# Most Profilers – High Overhead, No Line Info

| Tool | Description | Platforms | Profile level | Avg. overhead |
|---|---|---|---|---|
| cProfile (built-in) | • Text interactive mode: “pstats” (built-in)<br>• GUI viewer: RunSnakeRun<br>• Open Source | Any | Function | 1.3x-5x |
| Python Tools | • Visual Studio (2010+)<br>• Open Source | Windows | Function | ~2x |
| PyCharm | • Not free<br>• cProfile/yappi based | Any | Function | 1.3x-5x (same as cProfile) |
| line_profiler | • Pure Python<br>• Open Source<br>• Text-only viewer | Any | Line | Up to 10x or more |

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Example performance hogs

<table>
<tr><th>Task</th><th>Slow way</th><th>Faster way</th></tr>
<tr><td>Concatenate a list</td><td><pre>s = ''
for ch in some_lst:
    s += ch</pre></td><td><pre>s = ''.join(some_lst)</pre></td></tr>
<tr><td colspan="3" align="center">reason</td></tr>
<tr><td>Remove some value from a list</td><td><pre>while some_value in lst:
    lst.remove(some_value)</pre></td><td><pre>while True:
    try:
        lst.remove(some_value)
    except ValueError:
        break</pre></td></tr>
<tr><td colspan="3" align="center">reason</td></tr>
</table>

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Python example to profile: **demo.py**

```
class Encoder:
    CHAR_MAP = {'a': 'b', 'b': 'c'}
    def __init__(self, input):
        self.input = input

    def process_slow(self):
        result = ''
        for ch in self.input:
            result += self.CHAR_MAP.get(ch, ch)
        return result

    def process_fast(self):
        result = []
        for ch in self.input:
            result.append(self.CHAR_MAP.get(ch, ch))
        return ''.join(result)
```

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Python sample to profile: **run.py**

```
import demo
import time

def slow_encode(input):
    return demo.Encoder(input).process_slow()

def fast_encode(input):
    return demo.Encoder(input).process_fast()

if __name__ == '__main__':
    input = 'a' * 10000000 # 10 millions of 'a'
    start = time.time()
    s1 = slow_encode(input)
    slow_stop = time.time()
    print 'slow: %.2f sec' % (slow_stop - start)
    s2 = fast_encode(input)
    print 'fast: %.2f sec' % (time.time() - slow_stop)
```

slow: 9.15 sec = 1.00x
fast: 3.16 sec = 1.00x

No profiling overhead - a baseline for tools' overhead comparison

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# cProfile + pstats UI example

```
> python -m cProfile -o run.prof run.py
> python -m pstats run.prof

run.prof% sort time
run.prof% stats
Tue Jun 30 18:43:53 2015    run.prof

         30000014 function calls in 15.617 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    9.597    9.597   10.268   10.268 demo.py:6(process_slow)
        1    3.850    3.850    5.302    5.302 demo.py:12(process_fast)
 20000000    1.267    0.000    1.267    0.000 {method 'get' of 'dict' objects}
 10000000    0.790    0.000    0.790    0.000 {method 'append' of 'list' objects}
        1    0.066    0.066    0.066    0.066 {method 'join' of 'str' objects}
        1    0.038    0.038    5.340    5.340 run.py:7(fast_encode)
        1    0.009    0.009   15.617   15.617 run.py:1(<module>)
        1    0.000    0.000   10.268   10.268 run.py:4(slow_encode)
```

slow: 10.27 sec = 1.12x
fast: 5.34 sec = 1.69x



Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Optimization Notice

# cProfile + RunSnakeRun

Run Snake Run: run.prof

File View View Type

Percent functions

| Name | Calls | RC... | Local | /Call |
|---|---|---|---|---|
| process_slow | 1 | 1 | 7.91... | 7.91.. |
| process_fast | 1 | 1 | 3.82... | 3.82.. |
| <method 'get' of 'dict' objects> | 2000... | 20... | 1.16... | 0.00.. |
| <method 'append' of 'list' objects> | 1000... | 10... | 0.70... | 0.00.. |
| fast_encode | 1 | 1 | 0.10... | 0.10.. |
| <method 'join' of 'str' objects> | 1 | 1 | 0.07... | 0.07.. |
| <module> | 1 | 1 | 0.00... | 0.00.. |
| slow_encode | 1 | 1 | 0.00... | 0.00.. |
| <module> | 1 | 1 | 0.00... | 0.00.. |
| __init__ | 2 | 2 | 0.00... | 0.00.. |
| Encoder | 1 | 1 | 0.00... | 0.00.. |
| <method 'disable' of '_lsprof.Profiler' objects> | 1 | 1 | 0.00... | 0.00.. |
| | 0 | 2 | 0.00... | 0.00.. |

process_slow@demo.py:6 [8.515s]

process_fast@demo.py:12 [5.16

<method 'app

<method 'g

Callees | All Callees | Callers | All Callers | Source Code

| Name | Calls | RC... | Local | /Call | Cum | /Call | File | L... | Directory |
|---|---|---|---|---|---|---|---|---|---|
| <met... | 2000... | 20... | 1.16... | 0.00... | 1.16... | 0.00... | ~ | 0 | |

__init__@demo.py:3 [0.000s]

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# cProfile in PyCharm

demo - [C:\Users\ \PycharmProjects\demo] - demo.pstat - PyCharm 4.5.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

demo.pstat | demo.py | run.py | demo.pstat

Statistics | Call Graph

| Name | Call Count | Time (ms) | | Own Time (ms) | |
|---|---|---|---|---|---|
| process_slow | 1 | 10069 | 64.2% | 9423 | 60.1% |
| process_fast | 1 | 5560 | 35.5% | 4036 | 25.7% |
| <method 'get' of 'dict' objects> | 20000000 | 1272 | 8.1% | 1272 | 8.1% |
| <method 'append' of 'list' object | 10000000 | 831 | 5.3% | 831 | 5.3% |
| <method 'join' of 'str' objects> | 1 | 65 | 0.4% | 65 | 0.4% |
| fast_encode | 1 | 5601 | 35.7% | 40 | 0.3% |
| run.py | 1 | 15678 | 100.0% | 7 | 0.0% |
| slow_encode | 1 | 10069 | 64.2% | 0 | 0.0% |
| <time.time> | 3 | 0 | 0.0% | 0 | 0.0% |
| demo.py | 1 | 0 | 0.0% | 0 | 0.0% |
| __init__ | 2 | 0 | 0.0% | 0 | 0.0% |
| Encoder | 1 | 0 | 0.0% | 0 | 0.0% |

Run

```
C:\Python27\python.exe "C:\Program Files (x86)\JetBrains\PyCharm 4.5.3\helpers\profiler\run_profiler.py" 127.0.0.1 49605 C:/U:
Starting cProfile profiler

slow: 10.07 sec
fast: 5.60 sec
Snapshot saved to C:\Users\ \.PyCharm40\system\snapshots\demo.pstat
```

...ons on Java 7 if firewall is enabled. Command netsh advfirewall set... (5 minutes ago)

slow: 10.07 sec = 1.10x
fast: 5.60 sec = 1.77x

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# line_profiler results

```
Total time: 18.095 s
File: demo_lp.py
Function: process_slow at line 6
Line #      Hits     Time  Per Hit   % Time  Line Contents
==========================================================
     6                                       @profile
     7                                       def process_slow(self):
     8         1       14     14.0      0.0      result = ''
     9  10000001 10260548      1.0     23.3      for ch in self.input:
    10  10000000 33814644      3.4     76.7          result += self.CHAR_MAP.get(...
    11         1        4      4.0      0.0      return result

Total time: 16.8512 s
File: demo_lp.py
Function: process_fast at line 13
Line #      Hits     Time  Per Hit   % Time  Line Contents
==========================================================
    13                                       @profile
    14                                       def process_fast(self):
    15         1        7      7.0      0.0      result = []
    16  10000001 13684785      1.4     33.3      for ch in self.input:
    17  10000000 27048146      2.7     65.9          result.append(self.CHAR_MAP.get(...
    18         1   312611 312611.0      0.8      return ''.join(result)
```

slow: 24.32 sec = 2.66x
fast: 25.37 sec = 8.03x

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Python Tools GUI example

Performance_20150630(2).vsp - Microsoft Visual Studio

FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP

Performance_20150630(2).vsp | Performance_20150630.vsp | Performance_20150630(1).vsp | demo.py

Current View: Summary

**Instrumentation Profiling Report**

28.3 seconds of total execution time

Note:
Wallclock time (not CPU)

**Hot Path**

| Function Name | Elapsed Inclusive Time % | Elapsed Exclusive Time % |
|---|---|---|
| python.exe | 100.00 | 0.00 |
| run (module) | 100.00 | 0.07 |
| **run.slow_encode** | **60.13** | **0.00** |
| **run.fast_encode** | **39.79** | **0.00** |

Related Views: Call Tree Functions

**Functions With Most Individual Work**

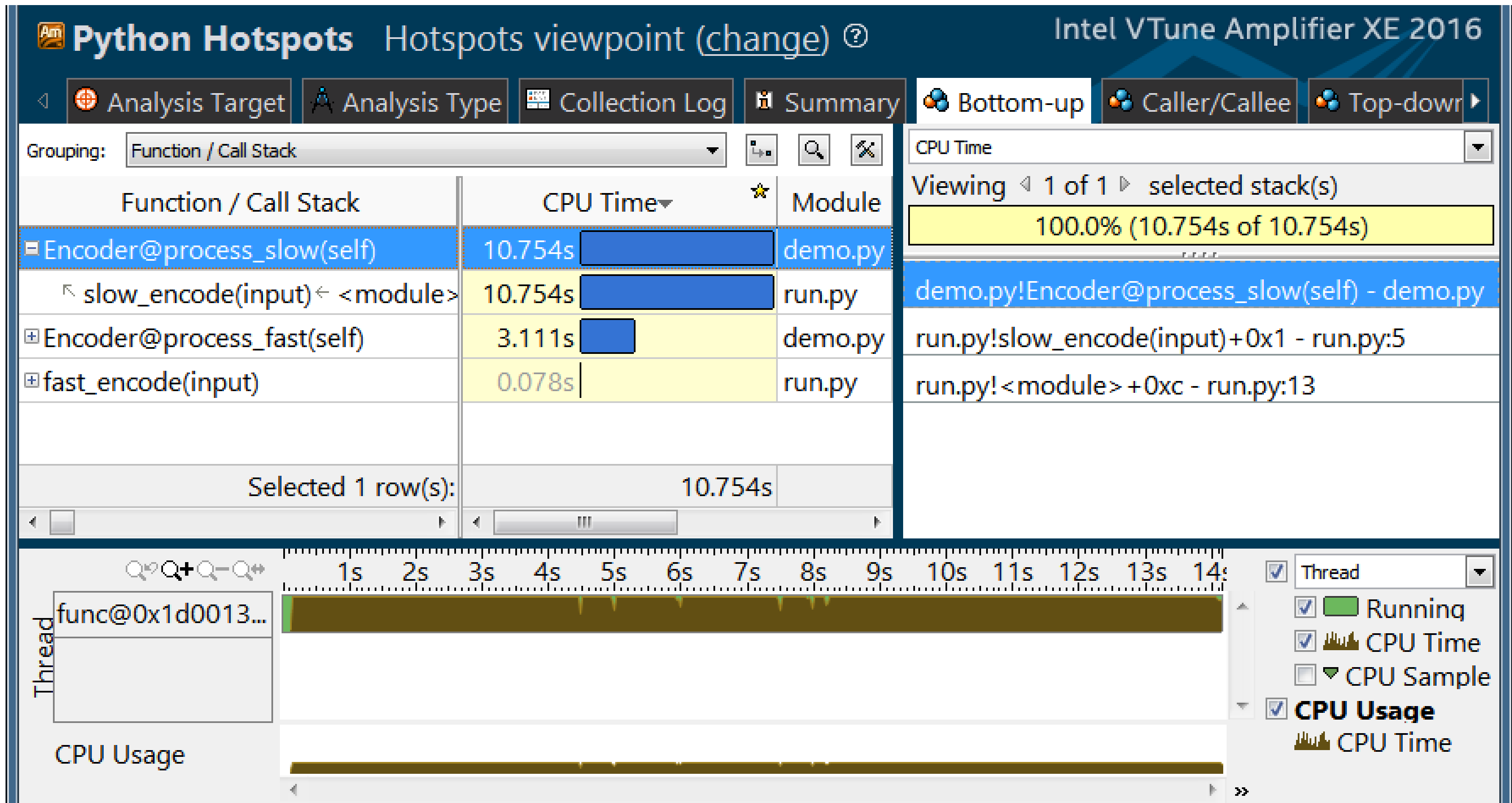
| Name | Exclusive Time % |
|---|---|
| demo.Encoder.process_slow | 56.49 |
| demo.Encoder.process_fast | 26.94 |
| dict.get | 9.12 |
| list.append | 7.00 |
| str.join | 0.39 |

Report
- Show Trimmed Ca
- Compare Reports
- Export Report Dat
- Save Analyzed Re
- Filter Report Data
- Toggle Full Screer
- Set Symbol Paths

slow: 17.40 sec = 1.90x
fast: 12.08 sec = 3.82x

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Intel® VTune Amplifier example




slow: 10.85 sec = 1.19x
fast: 3.30 sec = 1.05x


Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Optimization Notice

# Intel® VTune Amplifier – source view

Python Hotspots Hotspots viewpoint (change) Intel VTune Amplifier XE 2016

Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform | demo.py

Source | Assembly | Assembly grouping: Address

| S. | Source | CPU Time |
|---|---|---|
| 1 | class Encoder: | |
| 2 | CHAR_MAP = {'a': 'b', 'b': 'c'} | |
| 3 | def __init__(self, input): | |
| 4 | self.input = input | |
| 5 | | |
| 6 | def process_slow(self): | |
| 7 | result = '' | |
| 8 | for ch in self.input: | |
| 9 | result += self.CHAR_MAP.. | 10.754s |
| 10 | return result | |
| 11 | | |
| 12 | def process_fast(self): | |
| 13 | result = [] | |

CPU Time

Viewing 1 of 1 selected stac

100.0% (10.754s of 10.754s)

demo.py!Encod...lf) - demo.py

run.py!slow_en...0x1 - run.py:5

run.py!<modul...c - run.py:13



Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Optimization Notice

# Intel® VTune Amplifier: Accurate & Easy

## Line-level profiling details:

- Uses sampling profiling technique
- Average overhead ~1.1x-1.6x (on certain benchmarks)

## Cross-platform:

- Windows and Linux
- Python 32- and 64-bit; 2.7.x, 3.4.x, 3.5.0 versions

## Rich Graphical UI

## Supported workflows:

- Start application, wait for it to finish
- Attach to application, profile for a bit, detach

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Low Overhead *and* Line-Level Info

| Tool | Description | Platforms | Profile level | Avg. overhead |
|---|---|---|---|---|
| Intel® VTune Amplifier | • Rich GUI viewer | Windows<br>Linux | Line | ~1.1-1.6x |
| cProfile (built-in) | • Text interactive mode: “pstats” (built-in)<br>• GUI viewer: RunSnakeRun (Open Source)<br>• PyCharm | Any | Function | 1.3x-5x |
| Python Tools | • Visual Studio (2010+)<br>• Open Source | Windows | Function | ~2x |
| line_profiler | • Pure Python<br>• Open Source<br>• Text-only viewer | Any | Line | Up to<br>10x or more |

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# We’ve Had Success Tuning Our Python Code

- One widely-used web page in our internally set up Buildbot service: 3x speed up (from 90 seconds to 28)
- Report generator – from 350 sec to <2 sec for 1MB log file
  - Distilled version was the base for demo.py
- Internal SCons-based build system: several places sped up 2x or more
  - Loading all configs from scratch tuned from 6 minutes to 3 minutes

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Sign Up with Us to Give the Profiler a Try & Check out Intel® Software Development Tools

- Technical Preview & Beta Participation – email us at scripting@intel.com
  - We're also working on Intel-accelerated Python (e.g.  NumPy/SciPy, etc.), which is currently in Tech Preview. Sign up!
- Check out Intel Developer Zone – software.intel.com
  - Check out Intel® Software Development tools
  - Qualify for Free Intel® Software Development tools
- Catch me on the conference if you have some questions, feature suggestions, etc.

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Free Intel® Software Development Tools

Academic Researcher ›
Intel Performance Libraries for academic research

Student ›
For current students at degree-granting institutions.

Educator ›
For use in teaching curriculum.

Open Source Contributor ›
For developers actively contributing to open source projects.

Visit us at https://software.intel.com/en-us/qualify-for-free-software

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Q & A

… and again:

- For Tech Preview and Beta, drop us an email at scripting@intel.com
- Check out free Intel® software – just google for "free intel tools" to see if you're qualified

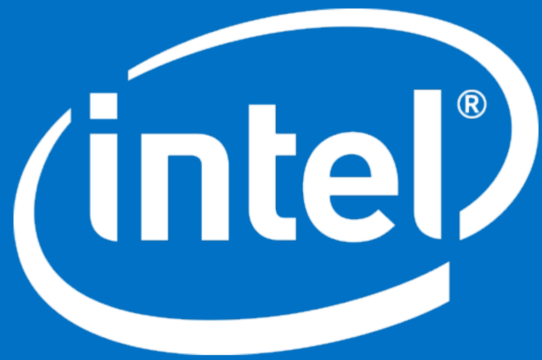
Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

# Performance Starts Here!

<table>
<tr><th>You are a:</th><th>Products Available++</th><th>Support Model</th><th>Price</th></tr>
<tr><td>Commercial Developer or Academic Researcher</td><td>Intel® Parallel Studio XE (Compilers, Performance Libraries & Analyzers)</td><td>Intel® Premier Support</td><td>$699** - $2949**</td></tr>
<tr><td>Academic Researcher+</td><td><u>Intel® Performance Libraries</u><br>Intel® Math Kernel Library<br>Intel® MPI Library<br>Intel® Threading Building Blocks<br>Intel® Integrated Performance Primitives</td><td rowspan="4">Forum only support</td><td rowspan="4">Free!</td></tr>
<tr><td>Student+</td><td rowspan="2">Intel® Parallel Studio XE Cluster Edition</td></tr>
<tr><td>Educator+</td></tr>
<tr><td>Open Source Contributor+</td><td>Intel® Parallel Studio XE Professional Edition</td></tr>
</table>

+Subject to qualification ++OS Support varies by product **Single Seat Pricing

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.


intel