Crowdsourcing code tuning with PeachPy.io

Marat Dukhan, Anirudh Gubba, Richard Vuduc
mdukhan3@gatech.edu, agubba3@gatech.edu, richie@cc.gatech.edu

Introduction

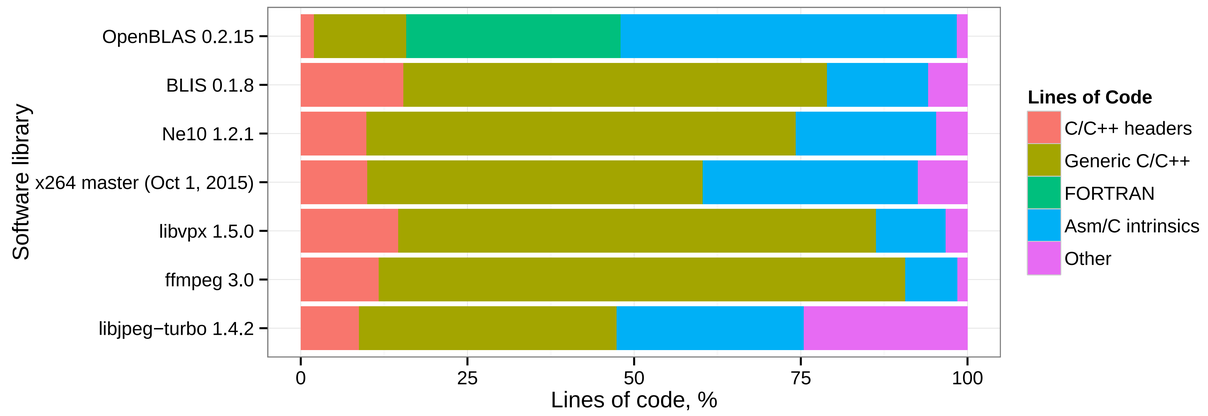
High-performance scientific libraries include a large portion of optimized assembly code. Maintaining processor-specific optimized assembly and keeping it up-to-date requires rare low-level optimization skills. In practice, a very small pool of potential contributors is further reduced by the lack of access to target hardware, and complicated software configurations required for scientific projects.
We develop infrastructure to increase productivity of performance tuning experts:

- PeachPy, a Python-based DSL for assembly metaprogamming. PeachPy automates boring and error-prone parts of assembly programming, such as register allocation and adaption of code to different ABIs.
- WebPeachPy, a Python interpreter with PeachPy that runs inside a Web browser.
- WebRunner, a REST server that safely loads and executes compiled assembly kernels.
- PeachPy.io, an in-browser IDE for assembly code tuning and analysis.

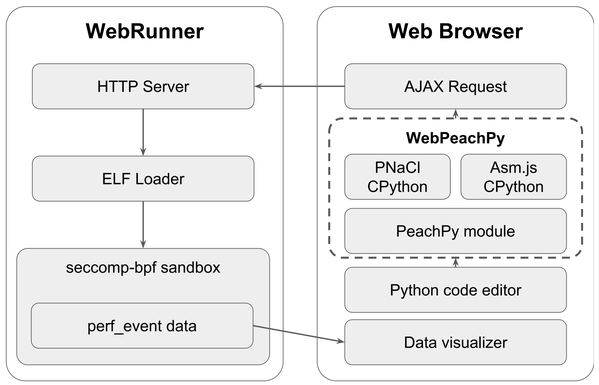
PeachPy Example: SGEMM

```
ymm_c = [[YMMRegister() for n in range(nr)] for m in range(mr // 8)]
ymm_a = [YMMRegister() for m in range(mr // 8)]
ymm_b_n = YMMRegister()

VZEROALL()

with Loop() as loop:
        for m in range(mr // 8):
                VMOVAPS(ymm_a[m], [reg_b + m * YMMRegister.size])

        for n in range(nr):
                VBROADCASTSS(ymm_b_n, [reg_b + n * float_.size])
                for m in range(mr // 8):
                        VFMADD231PS(ymm_c[m][n], ymm_b_n, ymm_a[m])

        ADD(reg_a, mr * float_.size)
        ADD(reg_b, nr * float_.size)

        DEC(reg_k)
        JNE(loop.begin)
```

Optimized libraries: code composition



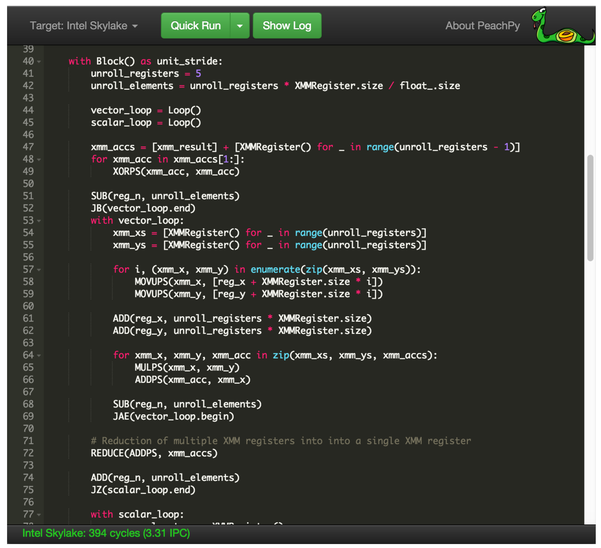
PeachPy.io Infrastructure



PeachPy Example: x264 Macro

```
                                %macro ABSW2 6
def absw2(dst1, dst2, src1, src2): ; dst1, dst2, src1, src2, tmp, tmp
  if function.target.has_ssse3:   %if cpuflag(ssse3)
    PABSW(dst1, src1)               pabsw   %1, %3
    PABSW(dst2, src2)               pabsw   %2, %4
  elif dst1 == src1:              %elifidn %1, %3
    tmp1, tmp2 = \                  pxor    %5, %5
      MMXRegister(), MMXRegiser()   pxor    %6, %6
    PXOR(tmp1, tmp1)                psubw   %5, %1
    PXOR(tmp2, tmp2)                psubw   %6, %2
    PSUBW(tmp1, dst1)               pmaxsw  %1, %5
    PSUBW(tmp2, dst2)               pmaxsw  %2, %6
    PMAXSW(dst1, tmp1)            %else
    PMAXSW(dst2, tmp2)              pxor    %1, %1
  else:                             pxor    %2, %2
    PXOR(dst1, dst1)                psubw   %1, %3
    PXOR(dst2, dst2)                psubw   %2, %4
    PSUBW(dst1, src1)               pmaxsw  %1, %3
    PSUBW(dst2, src2)               pmaxsw  %2, %4
    PMAXSW(dst1, src1)            %endif
    PMAXSW(dst2, src2)            %endmacro
```

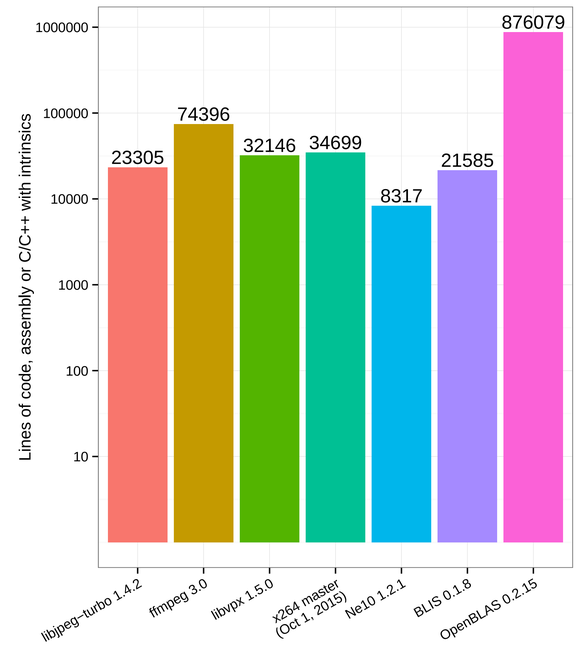
PeachPy.io Code Editor



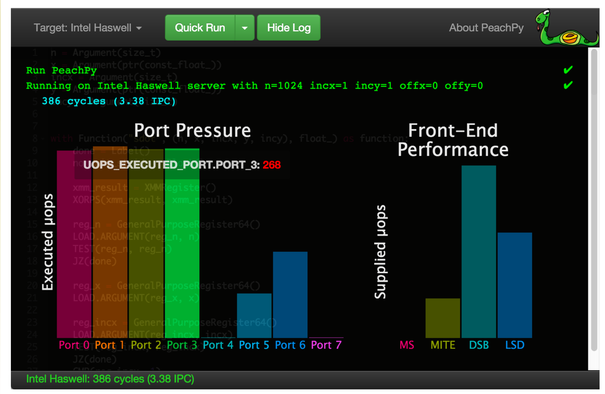
WebRunner Specification

```
<?xml version='1.0' encoding='utf-8'?>
<kernel name="sdot" namespace="blis">
        <query>
                <parameter name="n" type="uint64" default="1024" />
                <parameter name="incx" type="uint32" default="1" min="1" />
                <parameter name="incy" type="uint32" default="1" min="1" />
                <parameter name="offx" type="uint32" default="0" max="63" />
                <parameter name="offy" type="uint32" default="0" max="63" />
        </query>
        <call>
                <argument name="n" type="size_t" />
                <argument name="x" type="const float*" />
                <argument name="incx" type="size_t" />
                <argument name="y" type="const float*" />
                <argument name="incy" type="size_t" />
        </call>
</kernel>
```

Target-specific codes



PeachPy.io Code Analysis



Acknowledgements

This material is based upon work supported by the U.S. National Science Foundation (NSF) Award Number 1339745. We appreciate a hardware donation from AMD to support PeachPy.io Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of NSF or AMD.