OPTIMIZING TOMOPY
Performance analysis of grid reconstruction
Benchmarking: make_data.py

```python
import numpy as np, tomopy
obj = tomopy.shepp3d(size=512)
ang = tomopy.angles(750) # Generate uniformly spaced tilt angles.
sim = tomopy.project(obj, ang) # Calculate projections.
np.save('projection.npy', sim)
np.save('angles.npy', ang)
```

Checking dimensions and type of the projection data:

```python
In [1]: import numpy as np
In [2]: sim = np.load('projection.npy')
In [3]: sim.shape
Out[3]: (750, 512, 720)
In [4]: sim.dtype
Out[4]: dtype('float32')
```
Tomopy out of the box

conda create --name tomopy_nomkl \
   -c dgursoy \
   nomkl tomopy pyfftw fftw numpy scipy numexpr pywavelets \
   scikit-image ipython ipython-notebook astropy \
   python=3.5

conda create --name tomopy \
   -c dgursoy \
   tomopy pyfftw fftw numpy scipy numexpr pywavelets \
   scikit-image ipython ipython-notebook astropy \
   python=3.5
import numpy as np, tomopy, time as t

def timeit(func, named_args, kwargs):
    t0 = t.time()
    r = func(*named_args, **kwargs)
    t1 = t.time()
    return (t1 - t0, r)

sim = np.load('projection.npy')
ang = np.load('angles.npy')

# Reconstruct object:
recon_time, rec = timeit(tomopy.recon, (sim, ang), dict(algorithm='gridrec'))

print("Reconstruction time: {0:.3f}".format(recon_time))
## Performance times

```plaintext
(knl)$ numactl -p 1 \
   python recon_bench.py

<table>
<thead>
<tr>
<th>Platform</th>
<th>Version</th>
<th>Threads</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL</td>
<td>nomkl</td>
<td>256</td>
<td>47.696</td>
</tr>
<tr>
<td>KNL</td>
<td>mkl</td>
<td>256</td>
<td>98.56</td>
</tr>
<tr>
<td>KNL</td>
<td>mkl</td>
<td>256</td>
<td>12.965</td>
</tr>
</tbody>
</table>

must set OMP_NUM_THREADS=1

KMP_AFFINITY=disabled

(hsw)$ python
recon_bench.py

<table>
<thead>
<tr>
<th>Platform</th>
<th>Version</th>
<th>Threads</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>nomkl</td>
<td>32</td>
<td>4.246</td>
</tr>
<tr>
<td>HSW</td>
<td>mkl</td>
<td>32</td>
<td>11.356</td>
</tr>
<tr>
<td>HSW</td>
<td>mkl_seq</td>
<td>32</td>
<td>3.294</td>
</tr>
</tbody>
</table>

MKL_THREADING_LAYER=SEQUENTIAL
```
Hotspots tomopy_nomkl on KNL
Hotspots tomopy_nomkl on HSW
Building tomopy toolchain with icc

- Created recipes to build essential components with icc targeting common-avx512 architecture
- Small changes to tomopy itself
  - removed –lm in setup.py, vectorized code in phantom.py
  - changes to gridrec.c to enable vectorization
- Modules are fftw, pyfftw, tomopy, dxchange, dxfile, olefile are built locally
- Modules numpy, scipy, scikit-image are conda-installed from intel channel
- Other modules (pywavelets, etc) taken from dgrursoy channel
- netCDF4 and atropy were pip or conda installed
Tomopy recipe: build.sh

```bash
#!/bin/bash

export CC=icc
export LDSHARED="icc -shared"

$PYTHON setup.py config

C_INCLUDE_PATHS="$PREFIX/include" \ 
LD_LIBRARY_PATH="$PREFIX/lib" \ 
CFLAGS="-m64 -fomit-frame-pointer -pthread -qopenmp -fPIC -fp-model fast2 -O3 -xCORE-AVX2 -xCOMMON-AVX512 -I$PREFIX/include $CFLAGS" \ 
LDFLAGS="-L$PREFIX/lib $LDFLAGS" \ 
$PYTHON setup.py build_ext --inplace

$PYTHON setup.py install --old-and-unmanageable
```

Compile tomopy using icc targeting both HSW and KNL, enabling vectorization.

Recipes are available on cori.

Used vectorization report (--qopt-report=5) to guide optimizations
Tomopy recipe, cont.

```
[ set version = "1.0.1" ]
[ set buildnumber = 6 ]
[ set iccver = "16.0.3" ] [unix or py35]

package:
  name: tomopy
  version: {{version}}

build:
  number: {{buildnumber}}
  features:
    - intel

source:
  git_url: https://github.com/tomopy/tomopy
  git_rev: master
  patches:
    - intel_changes.patch

requirements:
  build:
    - python
    - intelpython
    - icc_rt
    - setuptools
    - numpy
    - fftw
```

Patch represents diff between official

github.com/tomopy/tomopy.git

and branch feature/intelem of its fork

github.com/oleksandr-pavlyk/tomopy.git
Gist of optimizations

- Replace lroundf(x) with (int) roundf(x)
- Replace ceil(x) with ceilf(x), etc.
- Replace fabs(x) with fabsf(f)
- Apply vectorization pragmas
- Split one double loop to enable vectorization
Changes in gridrec.c

for(iu=iu1; iu<=iu2; iu++)
{
    rtmp = wtbl[1+roundf(fabsf(U-iu)*tb1spcg)];
    for(iv=iv1, k=0; iv<=ivh; iv++, k++)
    {
        const float convolv = rtmp*work[k];
        H[iu][iv] += convolv*Cdata1;
        H[pdim-iu][pdim-iv] += convolv*Cdata2;
    }
}

#pragma simd assert
for(iu=iu1; iu <= iu2; iu++)
{
    rtmp = wtbl[(int) roundf(fabsf(U-iu)*tb1spcg)];
    for(iv=iv1, k=0; iv<=ivh; iv++, k++)
    {
        const float convolv = rtmp*work[k];
        H[iu][iv] += convolv*Cdata1;
        H[pdim-iu][pdim-iv] += convolv*Cdata2;
    }
}

// assert( iu == pdim2 || iu > iu2 );
for( ; iu <= pdim2 & & iu <= iu2; iu++)
{
    rtmp = wtbl[(int) roundf(fabsf(U-iu)*tb1spcg)];
    for(iv=iv1, k=0; iv<=ivh; iv++, k++)
    {
        const float convolv = rtmp*work[k];
        H[iu][iv] += convolv*Cdata1;
        H[pdim-iu][pdim-iv] += convolv*Cdata2;
    }
}

#pragma simd assert
for( ; iu<=iu2; iu++)
{
    rtmp = wtbl[(int) roundf(fabsf(U-iu)*tb1spcg)];
    for(iv=iv1, k=0; iv<=ivh; iv++, k++)
    {
        const float convolv = rtmp*work[k];
        H[iu][iv] += convolv*Cdata1;
        H[pdim-iu][pdim-iv] += convolv*Cdata2;
    }
}
Building tomopy-recipe

#!/bin/bash
export CONDA_BLD_PATH=./conda-build
conda build -c intel -c dgorsoy --override-channels \
   --no-anaconda-upload --python 3.5 --numpy 1.11 tomopy-recipe
Using built tomopy

#!/bin/bash -x

export _ENV_=$1

conda create --name $_ENV_ -c intel numpy scipy scikit-image numexpr h5py hdf5 six ipython python=3.5 --yes
conda activate $_ENV_
conda install -c intel -c dguusoy --override-channels pywavelets tifffile edffile spefile --yes
conda install astropy --yes
pip install netCDF4

# install modules locally built with icc
pushd conda-build/linux-64
conda install \
   dxfchange-0.1.2-py35_intel_0.tar.bz2 \n   fftw-3.3.6-intel_1.tar.bz2 \n   pyfftw-0.10.4-py35_intel_0.tar.bz2 \n   dxfile-0.4.0-py35_intel_0.tar.bz2 \n   olefile-0.44.0-py35_intel_1.tar.bz2 \n   tomopy-1.0.1-py35_intel_6.tar.bz2
## Performance results

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>python recon_bench.py</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSW</td>
<td>optimized</td>
<td>32</td>
<td>1.343</td>
</tr>
<tr>
<td>KNL</td>
<td>optimized</td>
<td>256</td>
<td>2.492</td>
</tr>
</tbody>
</table>

```
KMP_AFFINITY=disabled numactl -p 1 python recon_bench.py
```
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