Some quick tips to speed up your analysis

- Optimizing code
- Vectorization
- RDataFrame
What’s wrong with this?

Hint: there are two speed issues, and two style issues

```python
# Compute the vector product of the two velocities
i = 10
while j < i+1:
    k = i + 3
    s[j] += k*j
    j += 1
```
```python
# Add a displacement to the distance array
interval = 10
limit = userNumber + 1
scale = interval + 3
while j < limit:
    distance[j] += scale*j
    j += 1
```
Vectorization
This means taking loops over arrays and splitting up the operations onto multiple processors

• C++
  • Just add “-O3” to the compilation command; e.g.
    ```
    g++ -O3 myprog.cxx -o myprog
    ```

• Python
  • Python vectorization involves full or partial conversion of Python into C code; e.g., Cython and Numba, which are Python compilers (and are available on the Nevis particle-physics systems).
    • Numba: http://numba.pydata.org/
    • Cython: https://cython.org/
  • With numpy, there may be a simpler way...
import math, numpy as np

# Create a couple of 1000x1000 2D arrays filled with 1’s
data = np.ones(shape=(1000,1000), dtype=np.float)
sqdata = data

# This is slow:
for i in range(1000):
    for j in range(1000):
        data[i][j] *= 2.0
        sqdata[i][j] = math.sqrt(data[i][j])

# This is fast. We’re using numpy’s definition of “*” and sqrt.
data *= 2.0
sqdata = np.sqrt(data)
RDataFrame

An RDataFrame lets you do column-wise operations on an ntuple or TTree, instead of the row-wise operations you’ve done so far in the tutorial.

You can do ntuple analysis in a more “spreadsheet-like” way.

RDataFrame is a ROOT class. You can look it up in the ROOT web site (as you did with THD). It works with both C++ and Python.

<table>
<thead>
<tr>
<th>Row</th>
<th>event</th>
<th>ebeam</th>
<th>px</th>
<th>py</th>
<th>pz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>150.14</td>
<td>14.33</td>
<td>-4.02</td>
<td>143.54</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>149.79</td>
<td>0.05</td>
<td>-1.37</td>
<td>148.60</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>150.16</td>
<td>4.01</td>
<td>3.89</td>
<td>145.69</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>150.14</td>
<td>1.46</td>
<td>4.66</td>
<td>146.71</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>149.94</td>
<td>-10.34</td>
<td>11.07</td>
<td>148.33</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>150.18</td>
<td>17.08</td>
<td>-12.14</td>
<td>143.10</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>150.02</td>
<td>5.19</td>
<td>7.79</td>
<td>148.59</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>150.05</td>
<td>7.55</td>
<td>-7.43</td>
<td>144.45</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>150.07</td>
<td>0.23</td>
<td>-0.02</td>
<td>147.78</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>149.96</td>
<td>1.21</td>
<td>7.27</td>
<td>146.99</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>149.92</td>
<td>5.35</td>
<td>3.98</td>
<td>140.70</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>149.88</td>
<td>-4.63</td>
<td>-0.08</td>
<td>147.91</td>
</tr>
</tbody>
</table>