1 Profile

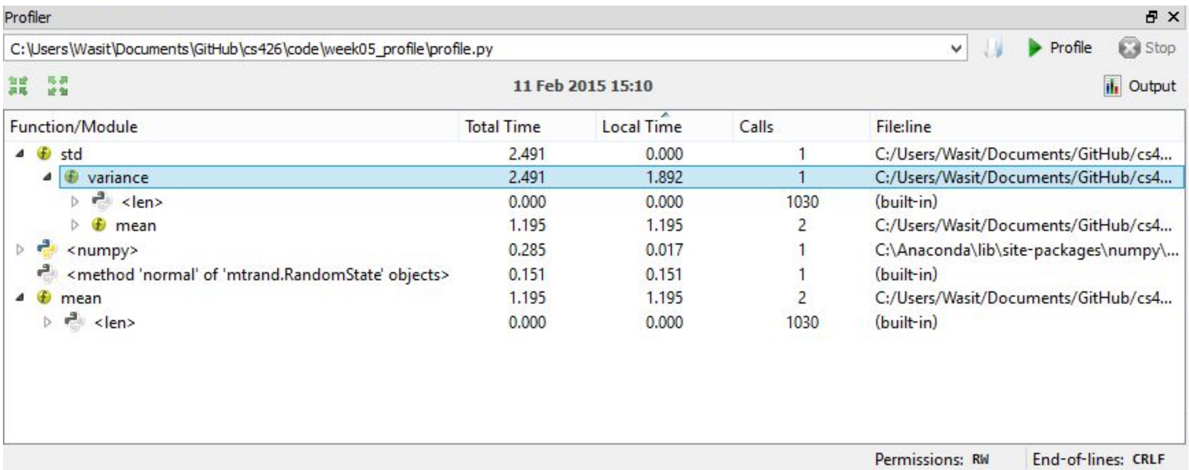
Profiling is a measurement of time and resources being used in a program. A program may consists of many sub-functions.

Total time is a total time that used by a function and any other sub-functions called by the function

Local time is time used by only the function excluding time of the sub-functions

Example

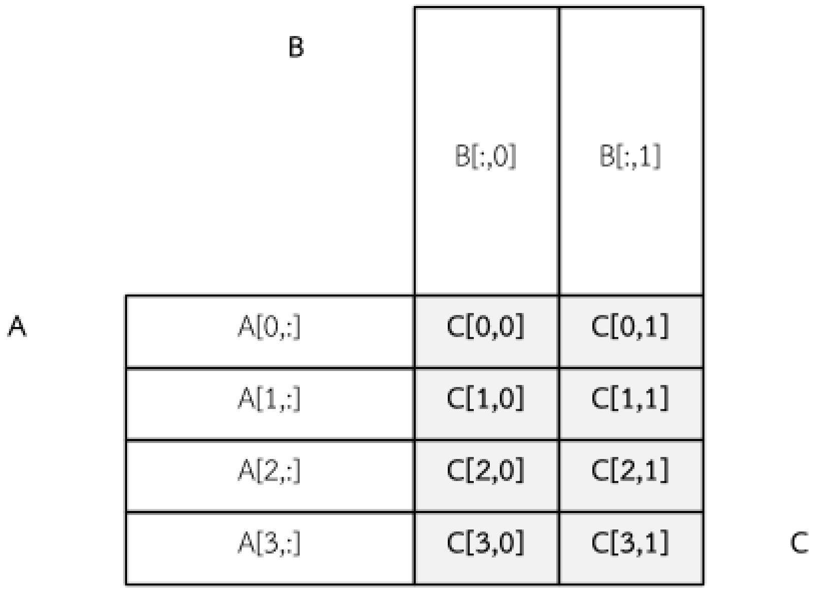
```
def mean(x):
    sum=0.0;
    for xi in x:
        sum=sum+xi
    return sum/len(x)

def variance(x):
    xbar=mean(x)
    sum=0.0;
    for xi in x:
        sum=sum+(xi-xbar)**2
    return sum/len(x)
def std(x):
    return variance(x)**0.5
if __name__ == "__main__":
    import numpy as np
    mu, sigma = 0, 0.1 # mean and standard deviation
    x = np.random.normal(mu, sigma, 1000000)
    print mean(x)
    print std(x)
```



Function/Module	Total Time	Local Time	Calls	File:line
std	2.491	0.000	1	C:/Users/Wasit/Documents/GitHub/cs4...
variance	2.491	1.892	1	C:/Users/Wasit/Documents/GitHub/cs4...
<len>	0.000	0.000	1030	(built-in)
mean	1.195	1.195	2	C:/Users/Wasit/Documents/GitHub/cs4...
<numpy>	0.285	0.017	1	C:\Anaconda\lib\site-packages\numpy\...
<method 'normal' of 'mtrand.RandomState' objects>	0.151	0.151	1	(built-in)
mean	1.195	1.195	2	C:/Users/Wasit/Documents/GitHub/cs4...
<len>	0.000	0.000	1030	(built-in)

2 Matrix Multiplication

In the example we are going to use ipython.parallel to find a product of A times B, where A and B are any matrices. Let us assume that have 8 cores computing system. The output C is the result of the product C= A*B.



The input matrices are divided into smaller blocks. In the case A and B are divided into 4 and 2 blocks, respectively. Therefore the output C consists of 8 blocks that comes from a number of blocks of A times a number of block of B. Then the 8 computation core are assigned to compute each block of C.

2.1 Psudo code

```
r_par is a number  of partitions of A
c_par is a number of partitions of B
distribute A and B to assigned cores
compute C[r,c]
gather C[r,c] from each core and assemble into C
```

```
import numpy as np
from IPython import parallel
from k_partition import partition
c=parallel.Client()
print c.ids
dv=c.direct_view()
dv.execute('import numpy as np')


x = np.linspace(0, 7, 8)
y = np.linspace(0, 7, 8)
B, A = np.meshgrid(x, y)


#rmax is a number of row of the output matrix
rmax=A.shape[0]
#cmax is a number of column of the output matrix
cmax=B.shape[1]

r_par=4
c_par=2
pr=partition(r_par,rmax)
pc=partition(c_par,cmax)
ri=0
for i in xrange(r_par):
    rf=ri+pr[i]
    x=i*c_par+np.arange(c_par)
    for j in x:
        c[j]['a']=A[ri:rf,:]
    print 'core: ',x
    print 'row ',ri,':',rf
    ri=rf
#print dv['a']

ci=0
for i in xrange(c_par):
    cf=ci+pc[i]
    x=np.arange(r_par)*c_par+i
    for j in x:
        c[j]['b']=B[:,ci:cf]

    print 'core: ',x
    print 'column ',ci,':',cf
    ci=cf
#print dv['b']
dv.execute('c=np.dot(a,b)')
```