Profiling and optimizing RAM and CPU in RMG-Py

Richard West

Northeastern.edu/comocheng

“Premature optimization is the root of all evil.”

- Donald Knuth, Turing Award Lecture 1974

“Premature optimization is the root of all evil.”

- Donald Knuth, Turing Award Lecture 1974

Not an excuse to write bad code!

1. Get it working
2. Get it working right
3. Test it’s working right
4. Profile it
5. Improve the slow parts, then go to 3

How Python does memory management

Reference counting...

...with a garbage collector.

Memory Allocator is in cPython...

...and in the OS.

Every object has a dictionary of attributes...

...unless you use __slots__ or Cython.

What makes Python slow?

Attribute lookups!

Slow

```
for species1 in long_list:
    for species2 in long_list:
        distance = ( species1.molecule[0].atom[0].x -
                     species2.molecule[0].atom[0].x )
```

Faster

```
for species1 in long_list:
    x1 = species1.molecule[0].atom[0].x
    for species2 in long_list:
        distance = x1 - species2.molecule[0].atom[0].x
```

Slow

```
newlist = []
for word in oldlist:
    newlist.append(word.upper())
```

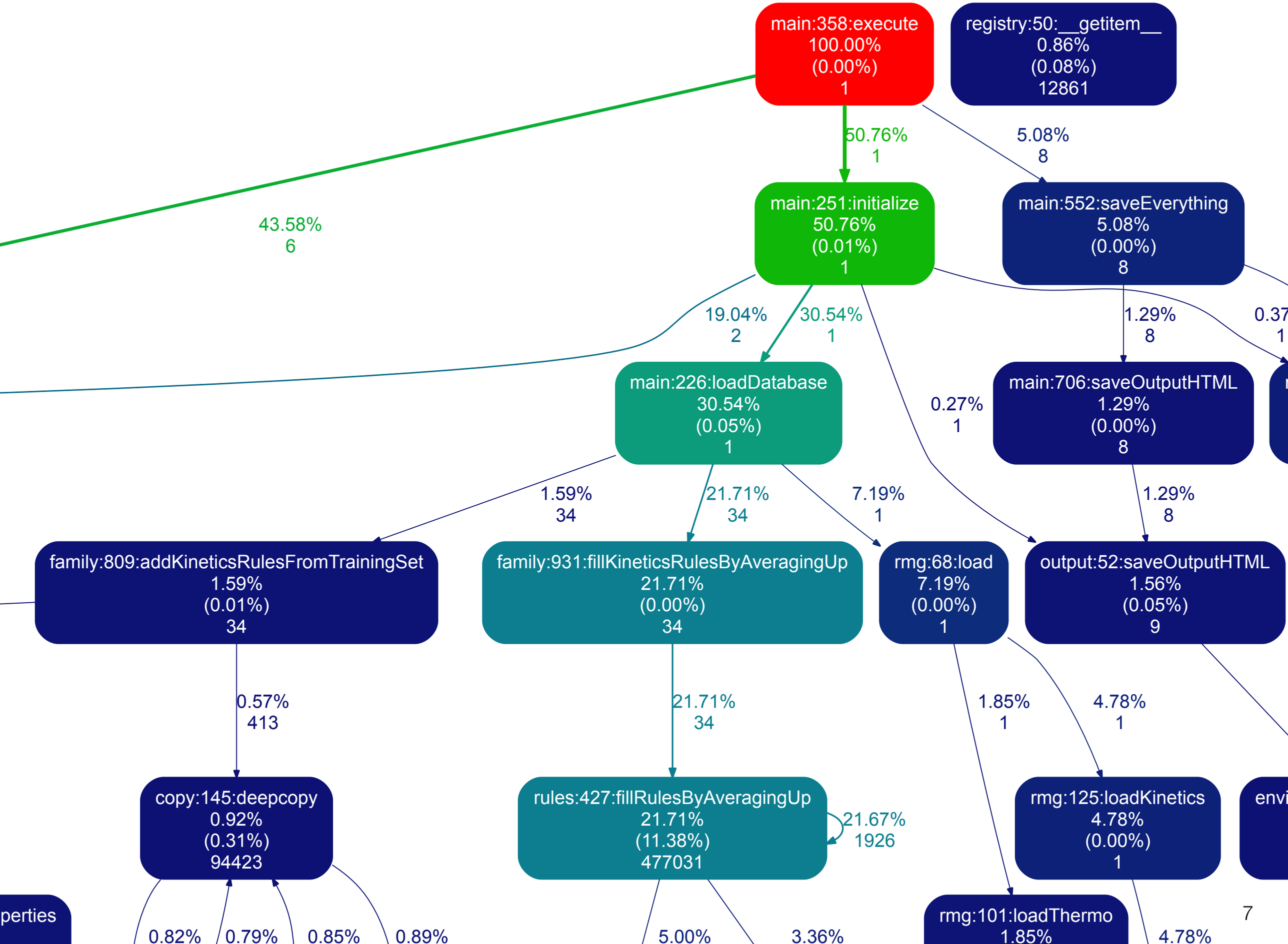
Faster

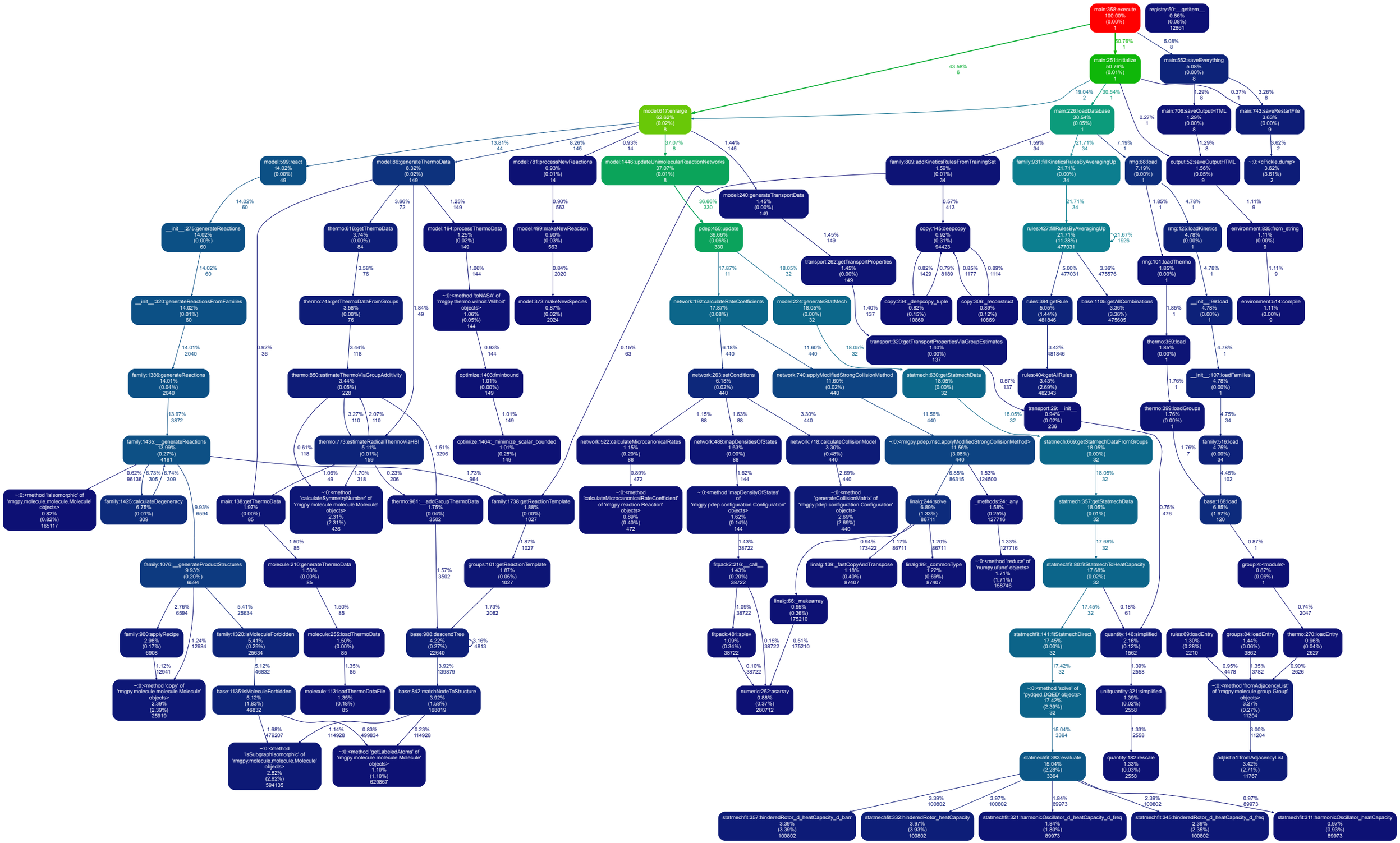
```
upper = str.upper()
newlist = []
append = newlist.append
for word in oldlist:
    append(upper(word))
```

How to profile RMG-Py for CPU use

$ python rmg.py -p hexadiene/input.py

```
MODEL GENERATION COMPLETED

The final model core has 9 species and 8 reactions
The final model edge has 140 species and 317 reactions

RMG execution terminated at Wed Jan 22 14:42:49 2014
================================================================================
                                Profiling Data
================================================================================
Sorted by internal time
Wed Jan 22 14:42:49 2014    /Users/rwest/Code/RMG-Py/examples/rmg/1,3-hexadiene/RMG.profile

         25196711 function calls (24439766 primitive calls) in 46.146 seconds

   Ordered by: internal time
   List reduced from 2468 to 25 due to restriction <25>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
477031/34    5.251    0.000   10.022    0.295 rules.py:427(fillRulesByAveragingUp)
   100802    1.812    0.000    1.831    0.000 statmechfit.py:332(hinderedRotor_heatCapacity)
        2    1.668    0.834    1.671    0.835 {cPickle.dump}
   100802    1.563    0.000    1.563    0.000 statmechfit.py:357(hinderedRotor_d_heatCapacity_d_barr)
   475605    1.549    0.000    1.549    0.000 base.py:1105(getAllCombinations)
      440    1.422    0.003    5.334    0.012 {rmgpy.pdep.msc.applyModifiedStrongCollisionMethod}
   594135    1.300    0.000    1.300    0.000 {method 'isSubgraphIsomorphic' of 'rmgpy.molecule.molecule.Molecule'
objects}
    11767    1.252    0.000    1.577    0.000 adjlist.py:51(fromAdjacencyList)
   482343    1.242    0.000    1.582    0.000 rules.py:404(getAllRules)
      440    1.240    0.003    1.240    0.003 {method 'generateCollisionMatrix' of
'rmgpy.pdep.configuration.Configuration' objects}
    25919    1.104    0.000    1.104    0.000 {method 'copy' of 'rmgpy.molecule.molecule.Molecule' objects}
       32    1.101    0.034    8.042    0.251 {method 'solve' of 'pydqed.DOFD' objects}
```

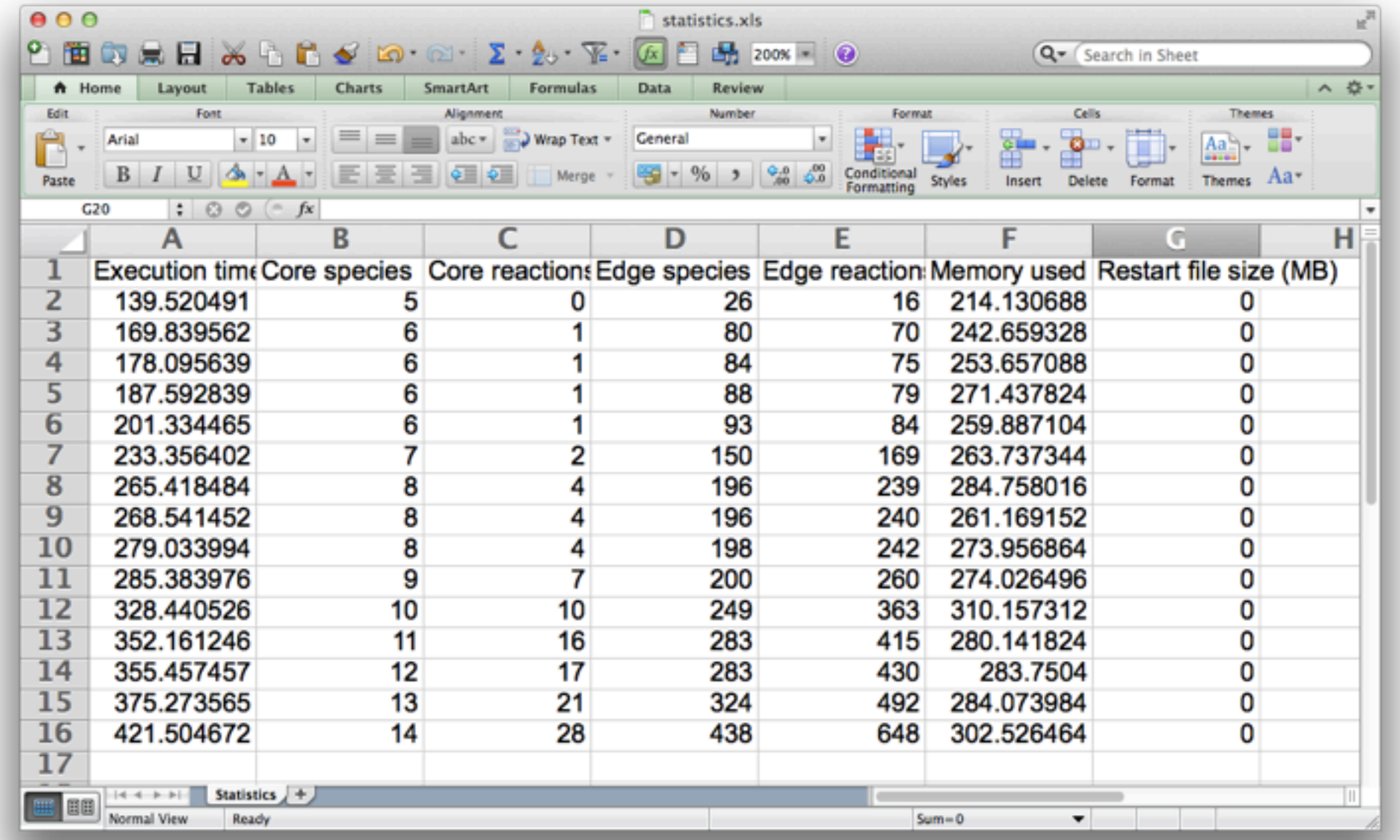

main:358:execute
100.00%
(0.00%)
1
registry:50:__getitem__
0.86%
(0.08%)
12861
50.76%
1
5.08%
8
43.58%
6
main:251:initialize
50.76%
(0.01%)
1
main:552:saveEverything
5.08%
(0.00%)
8
19.04%
2
30.54%
1
1.29%
8
0.37
1
main:226:loadDatabase
30.54%
(0.05%)
1
main:706:saveOutputHTML
1.29%
(0.00%)
8
0.27%
1
1.59%
34
21.71%
34
7.19%
1
1.29%
8
family:809:addKineticsRulesFromTrainingSet
1.59%
(0.01%)
34
family:931:fillKineticsRulesByAveragingUp
21.71%
(0.00%)
34
rmg:68:load
7.19%
(0.00%)
1
output:52:saveOutputHTML
1.56%
(0.05%)
9
0.57%
413
21.71%
34
1.85%
1
4.78%
1
copy:145:deepcopy
0.92%
(0.31%)
94423
rules:427:fillRulesByAveragingUp
21.71%
(11.38%)
477031
21.67%
1926
rmg:125:loadKinetics
4.78%
(0.00%)
1
perties
0.82%
0.79%
0.85%
0.89%
5.00%
3.36%
rmg:101:loadThermo
1.85%
4.78%


main:358:execute
100.00%
(0.00%)
1
registry:50:__getitem__
0.86%
(0.08%)
12861
43.58%
6
50.76%
1
5.08%
8
main:251:initialize
50.76%
(0.01%)
1
main:552:saveEverything
5.08%
(0.00%)
8
model:617:enlarge
62.62%
(0.02%)
8
main:226:loadDatabase
30.54%
(0.05%)
1
main:706:saveOutputHTML
1.29%
(0.00%)
8
main:743:saveRestartFile
3.63%
(0.00%)
9
19.04%
2
30.54%
1
1.29%
8
0.37%
1
3.26%
8
0.27%
1
13.81%
44
8.26%
145
0.93%
14
37.07%
8
1.44%
145
1.59%
34
21.71%
34
7.19%
1
1.29%
8
3.62%
2
model:599:react
14.02%
(0.00%)
49
model:86:generateThermoData
8.32%
(0.02%)
149
model:781:processNewReactions
0.93%
(0.01%)
14
model:1446:updateUnimolecularReactionNetworks
37.07%
(0.01%)
8
family:809:addKineticsRulesFromTrainingSet
1.59%
(0.01%)
34
family:931:fillKineticsRulesByAveragingUp
21.71%
(0.00%)
34
rmg:68:load
7.19%
(0.00%)
1
output:52:saveOutputHTML
1.56%
(0.05%)
9
~:0:<cPickle.dump>
3.62%
(3.61%)
2
model:240:generateTransportData
1.45%
(0.00%)
149
14.02%
60
3.66%
72
1.25%
149
0.90%
563
36.66%
330
0.57%
413
21.71%
34
1.85%
1
4.78%
1
1.11%
9
__init__:275:generateReactions
14.02%
(0.00%)
60
thermo:616:getThermoData
3.74%
(0.00%)
84
model:164:processThermoData
1.25%
(0.02%)
149
model:499:makeNewReaction
0.90%
(0.03%)
563
pdep:450:update
36.66%
(0.06%)
330
copy:145:deepcopy
0.92%
(0.31%)
94423
rules:427:fillRulesByAveragingUp
21.71%
(11.38%)
477031
21.67%
1926
rmg:125:loadKinetics
4.78%
(0.00%)
1
environment:835:from_string
1.11%
(0.00%)
9
1.45%
149
transport:262:getTransportProperties
1.45%
(0.00%)
149
rmg:101:loadThermo
1.85%
(0.00%)
1
14.02%
60
3.58%
76
1.06%
144
0.84%
2020
17.87%
11
18.05%
32
0.82%
1429
0.79%
8189
0.85%
1177
0.89%
1114
5.00%
477031
3.36%
475576
4.78%
1
1.11%
9
__init__:320:generateReactionsFromFfamilies
14.02%
(0.01%)
60
thermo:745:getThermoDataFromGroups
3.58%
(0.00%)
76
~:0:<method 'toNASA' of 'rmgpy.thermo.wilhoit.Wilhoit' objects>
1.06%
(0.05%)
144
model:373:makeNewSpecies
0.87%
(0.02%)
2024
network:192:calculateRateCoefficients
17.87%
(0.08%)
11
model:224:generateStatMech
18.05%
(0.00%)
32
copy:234:_deepcopy_tuple
0.82%
(0.15%)
10869
copy:306:_reconstruct
0.89%
(0.12%)
10869
rules:384:getRule
5.05%
(1.44%)
481846
base:1105:getAllCombinations
3.36%
(3.36%)
475605
__init__:99:load
4.78%
(0.00%)
1
environment:514:compile
1.11%
(0.00%)
9
1.84%
49
0.93%
144
1.40%
137
3.42%
481846
1.85%
1
4.78%
1
family:1386:generateReactions
14.01%
(0.04%)
2040
thermo:850:estimateThermoViaGroupAdditivity
3.44%
(0.05%)
228
optimize:1403:fminbound
1.01%
(0.00%)
149
transport:320:getTransportPropertiesViaGroupEstimates
1.40%
(0.00%)
137
thermo:359:load
1.85%
(0.00%)
1
0.92%
36
0.15%
63
14.01%
2040
3.27%
110
2.07%
110
1.01%
149
6.18%
440
11.60%
440
18.05%
32
0.57%
137
4.78%
1
network:263:setConditions
6.18%
(0.02%)
440
network:740:applyModifiedStrongCollisionMethod
11.60%
(0.02%)
440
statmech:630:getStatmechData
18.05%
(0.00%)
32
rules:404:getAllRules
3.43%
(2.69%)
482343
__init__:107:loadFamilies
4.78%
(0.00%)
1
13.97%
3872
transport:29:__init__
0.94%
(0.02%)
236
thermo:399:loadGroups
1.76%
(0.00%)
1
1.15%
88
1.63%
88
3.30%
440
11.56%
440
18.05%
32
1.76%
1
4.75%
34
family:1435:__generateReactions
13.99%
(0.27%)
4181
thermo:773:estimateRadicalThermoViaHBI
5.11%
(0.01%)
159
optimize:1464:_minimize_scalar_bounded
1.01%
(0.28%)
149
network:522:calculateMicrocanonicalRates
1.15%
(0.20%)
88
network:488:mapDensitiesOfStates
1.63%
(0.00%)
88
network:718:calculateCollisionModel
3.30%
(0.48%)
440
~:0:<rmgpy.pdep.msc.applyModifiedStrongCollisionMethod>
11.56%
(3.08%)
440
statmech:669:getStatmechDataFromGroups
18.05%
(0.00%)
32
family:516:load
4.75%
(0.00%)
34
0.62%
96136
6.73%
305
6.74%
309
0.61%
118
1.51%
3296
0.89%
472
1.62%
144
2.69%
440
6.85%
86315
1.53%
124500
18.05%
32
1.76%
7
4.45%
102
~:0:<method 'isIsomorphic' of 'rmgpy.molecule.molecule.Molecule' objects>
0.82%
(0.82%)
165117
family:1425:calculateDegeneracy
6.75%
(0.01%)
309
main:138:getThermoData
1.97%
(0.00%)
85
~:0:<method 'calculateSymmetryNumber' of 'rmgpy.molecule.molecule.Molecule' objects>
2.31%
(2.31%)
436
thermo:961:__addGroupThermoData
1.75%
(0.04%)
3502
family:1738:getReactionTemplate
1.88%
(0.00%)
1027
~:0:<method 'calculateMicrocanonicalRateCoefficient' of 'rmgpy.reaction.Reaction' objects>
0.89%
(0.40%)
472
~:0:<method 'mapDensityOfStates' of 'rmgpy.pdep.configuration.Configuration' objects>
1.62%
(0.14%)
144
~:0:<method 'generateCollisionMatrix' of 'rmgpy.pdep.configuration.Configuration' objects>
2.69%
(2.69%)
440
linalg:244:solve
6.89%
(1.33%)
86711
_methods:24:_any
1.58%
(0.25%)
127716
statmech:357:getStatmechData
18.05%
(0.01%)
32
base:168:load
6.85%
(1.97%)
120
0.62%
96136
6.73%
305
6.74%
309
9.93%
6594
1.73%
964
1.06%
49
1.70%
318
0.23%
206
0.75%
476
0.87%
1
family:1076:__generateProductStructures
9.83%
(0.20%)
6594
molecule:210:generateThermoData
1.50%
(0.00%)
85
groups:101:getReactionTemplate
1.87%
(0.05%)
1027
fitpack2:216:__call__
1.43%
(0.20%)
38722
linalg:139:_fastCopyAndTranspose
1.16%
(0.40%)
87407
linalg:99:_commonType
1.22%
(0.69%)
87407
~:0:<method 'reduce' of 'numpy.ufunc' objects>
1.71%
(1.71%)
158746
statmechfit:80:fitStatmechToHeatCapacity
17.68%
(0.02%)
32
group:4:<module>
0.87%
(0.06%)
1
1.50%
85
1.57%
3502
1.87%
1027
0.94%
173422
1.17%
86711
1.20%
86711
1.33%
127716
17.68%
32
0.87%
1
2.76%
6594
5.41%
25634
1.24%
12684
1.50%
85
1.73%
2082
1.43%
38722
17.45%
32
0.18%
61
0.74%
2047
family:960:applyRecipe
2.98%
(0.17%)
6908
family:1320:isMoleculeForbidden
5.41%
(0.29%)
25634
molecule:255:loadThermoData
1.50%
(0.00%)
85
base:908:descendTree
4.22%
(0.27%)
22640
3.16%
4813
linalg:66:_makearray
0.95%
(0.36%)
175210
statmechfit:141:fitStatmechDirect
17.45%
(0.00%)
32
quantity:146:simplified
2.16%
(0.12%)
1562
rules:69:loadEntry
1.30%
(0.28%)
2210
groups:84:loadEntry
1.44%
(0.06%)
3862
thermo:270:loadEntry
0.96%
(0.04%)
2627
1.09%
38722
1.12%
12941
5.12%
46832
1.35%
85
3.92%
139879
fitpack:481:splev
1.09%
(0.34%)
38722
0.15%
38722
0.51%
175210
17.42%
32
1.39%
2558
0.95%
4478
1.35%
3782
0.90%
2626
0.10%
38722
~:0:<method 'copy' of 'rmgpy.molecule.molecule.Molecule' objects>
2.39%
(2.39%)
25919
base:1135:isMoleculeForbidden
5.12%
(1.83%)
46832
molecule:113:loadThermoDataFile
1.35%
(0.18%)
85
base:842:matchNodeToStructure
3.92%
(1.58%)
168019
numeric:252:asarray
0.88%
(0.37%)
280712
~:0:<method 'solve' of 'pydqed.DQED' objects>
17.42%
(2.39%)
32
unitquantity:321:simplified
1.39%
(0.02%)
2558
~:0:<method 'fromAdjacencyList' of 'rmgpy.molecule.group.Group' objects>
3.27%
(0.27%)
11204
15.04%
3364
1.33%
2558
3.00%
11204
1.68%
479207
1.14%
114928
0.83%
499834
0.23%
114928
~:0:<method 'isSubgraphIsomorphic' of 'rmgpy.molecule.molecule.Molecule' objects>
2.82%
(2.82%)
594135
~:0:<method 'getLabeledAtoms' of 'rmgpy.molecule.molecule.Molecule' objects>
1.10%
(1.10%)
628867
statmechfit:383:evaluate
15.04%
(2.28%)
3364
quantity:182:rescale
1.33%
(0.03%)
2558
adjlist:51:fromAdjacencyList
3.42%
(2.71%)
11767
3.39%
100802
3.97%
100802
1.84%
89973
2.39%
100802
0.97%
89973
statmechfit:357:hinderedRotor_d_heatCapacity_d_barr
3.39%
(3.39%)
100802
statmechfit:332:hinderedRotor_heatCapacity
3.97%
(3.93%)
100802
statmechfit:321:harmonicOscillator_d_heatCapacity_d_freq
1.84%
(1.80%)
89973
statmechfit:345:hinderedRotor_d_heatCapacity_d_freq
2.39%
(2.35%)
100802
statmechfit:311:harmonicOscillator_heatCapacity
0.97%
(0.93%)
89973

How to profile RMG-Py for memory use

Just run it

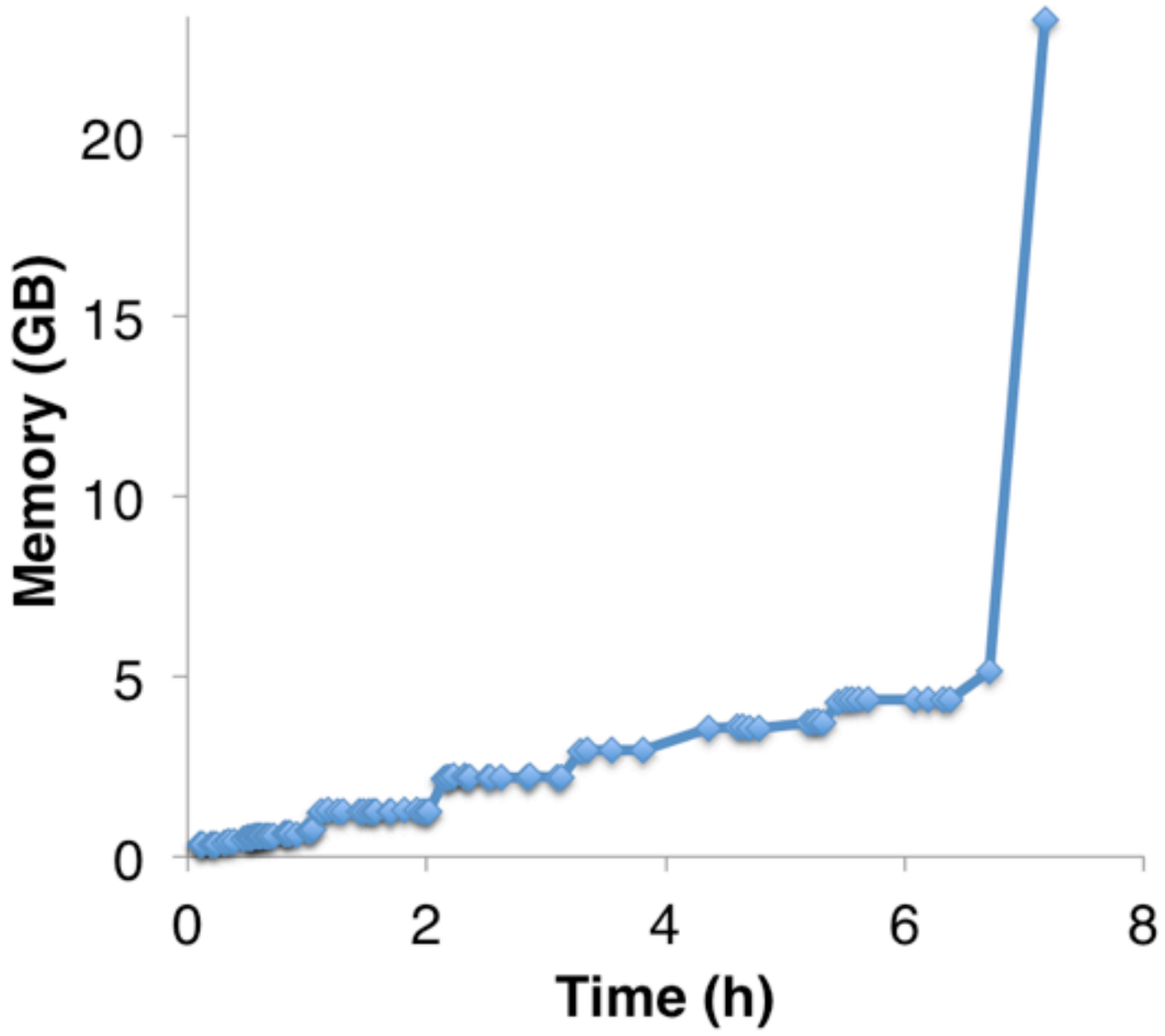
statistics.xls



	A	B	C	D	E	F	G	H
1	Execution time	Core species	Core reactions	Edge species	Edge reaction	Memory used	Restart file size (MB)	
2	139.520491	5	0	26	16	214.130688	0	
3	169.839562	6	1	80	70	242.659328	0	
4	178.095639	6	1	84	75	253.657088	0	
5	187.592839	6	1	88	79	271.437824	0	
6	201.334465	6	1	93	84	259.887104	0	
7	233.356402	7	2	150	169	263.737344	0	
8	265.418484	8	4	196	239	284.758016	0	
9	268.541452	8	4	196	240	261.169152	0	
10	279.033994	8	4	198	242	273.956864	0	
11	285.383976	9	7	200	260	274.026496	0	
12	328.440526	10	10	249	363	310.157312	0	
13	352.161246	11	16	283	415	280.141824	0	
14	355.457457	12	17	283	430	283.7504	0	
15	375.273565	13	21	324	492	284.073984	0	
16	421.504672	14	28	438	648	302.526464	0	
17								

How to profile RMG-Py for memory use

Just run it

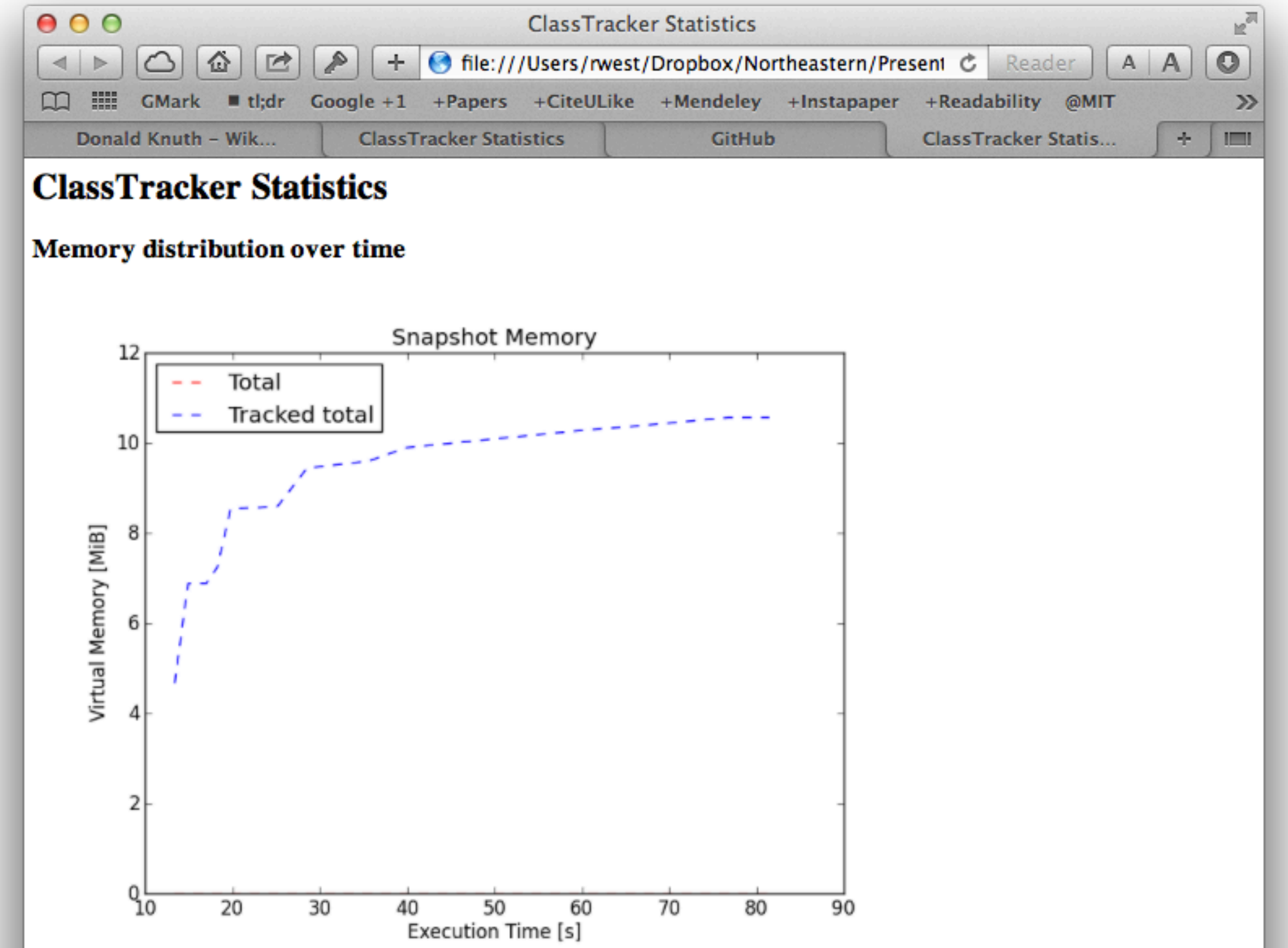


How to profile RMG-Py for memory use

Take your pick of toolkits, libraries, etc.

- Guppy
- Meliae
- Pympler

$ python rmg.py -m example/input.py



$ p

Snapshots statistics

2 core 5 edge species 13 seconds snapshot at 00:00:13.42

Total virtual memory assigned to the program at that time was 2.58 GB, which includes 12.14 KB profiling overhead. The ClassTracker tracked 4.66 MB in total. The measurable objects including code objects but excluding overhead have a total size of 0 B.

Class	Instance #	Total	Average size	Share
rmgpy.rmg.model.CoreEdgeReactionModel	1	4.66 MB	4.66 MB	0.18%
rmgpy.rmg.model.Species	7	3.12 KB	456 B	0.00%

Other

3 core 8 edge species 15 seconds snapshot at 00:00:14.94

Total virtual memory assigned to the program at that time was 2.61 GB, which includes 21.41 KB profiling overhead. The ClassTracker tracked 6.87 MB in total. The measurable objects including code objects but excluding overhead have a total size of 0 B.

Class	Instance #	Total	Average size	Share
rmgpy.rmg.model.CoreEdgeReactionModel	1	6.87 MB	6.87 MB	0.26%
rmgpy.rmg.model.Species	11	4.87 KB	453 B	0.00%

Other

4 core 8 edge species 16 seconds snapshot at 00:00:16.02

Other resources

SlideShare.net

e.g. http://www.slideshare.net/g3_nittala/profiling-and-optimization-14511764

Google.com

e.g. “Python memory profiling”