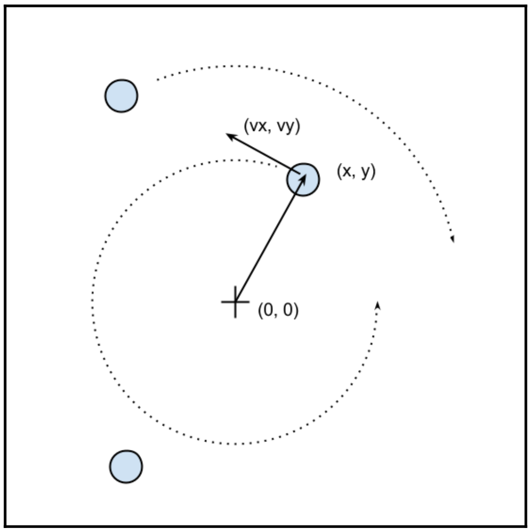
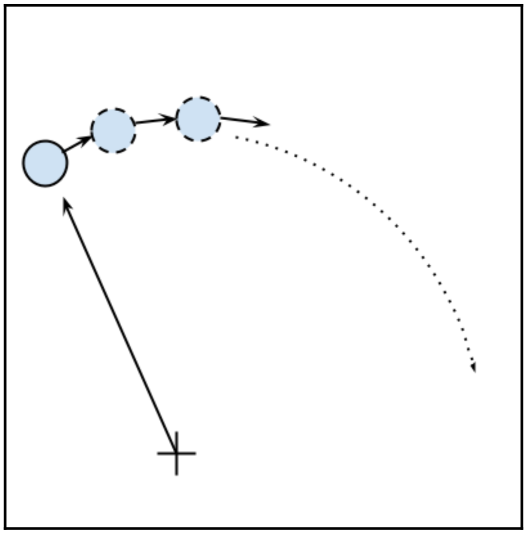
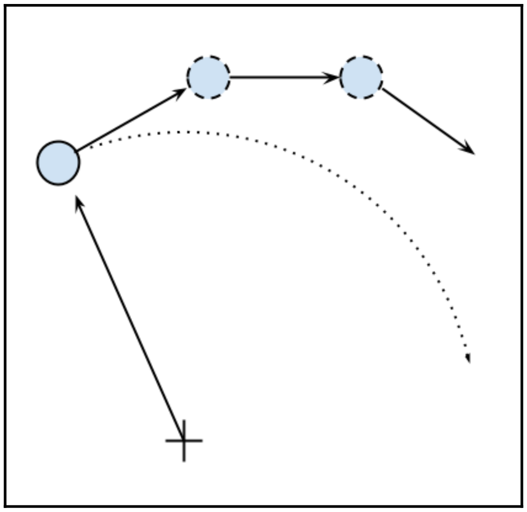
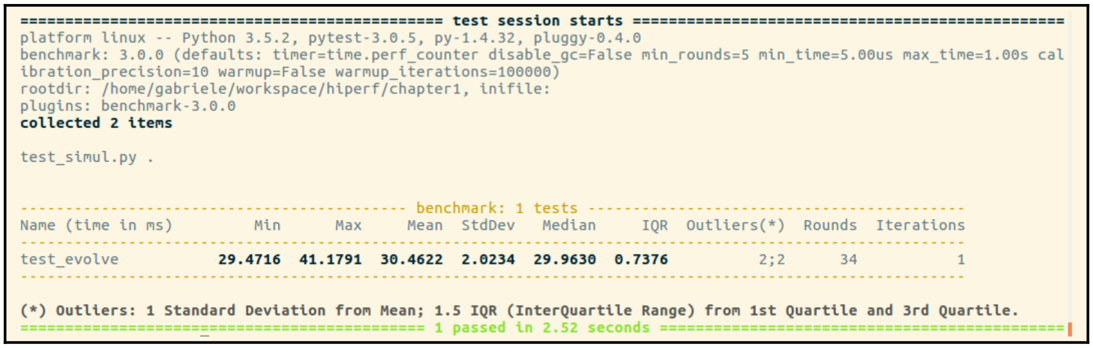
1
Graphic Bundle

Chapter 1: Benchmarking and Profiling

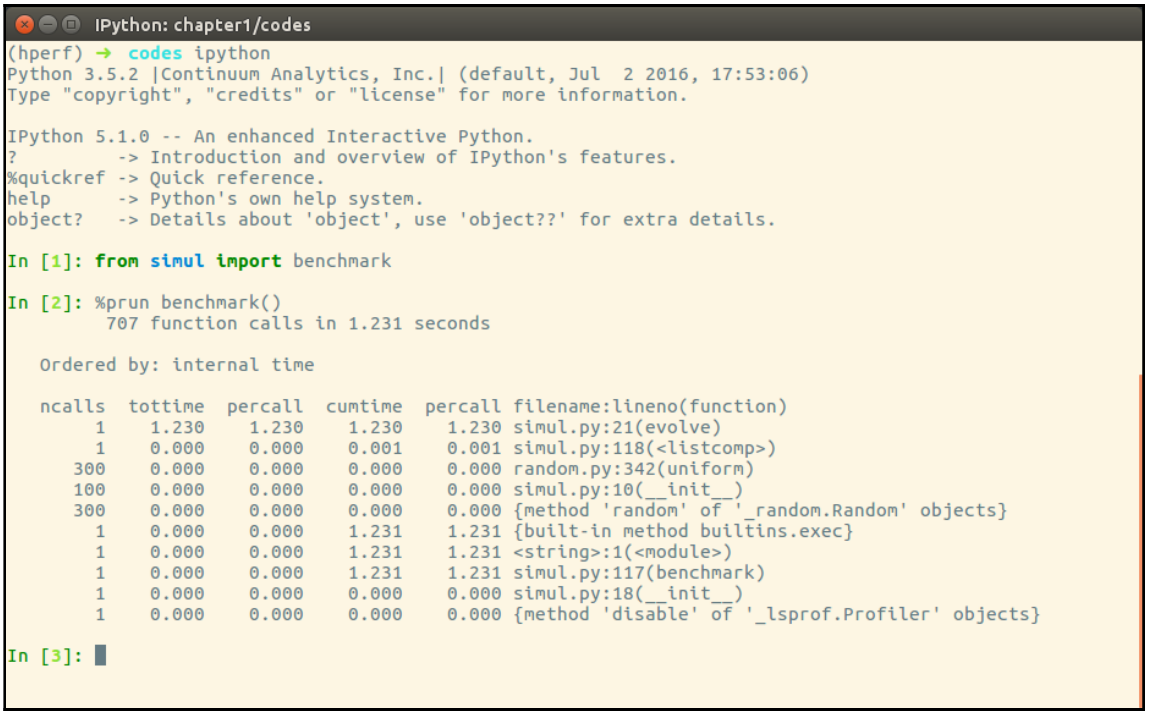






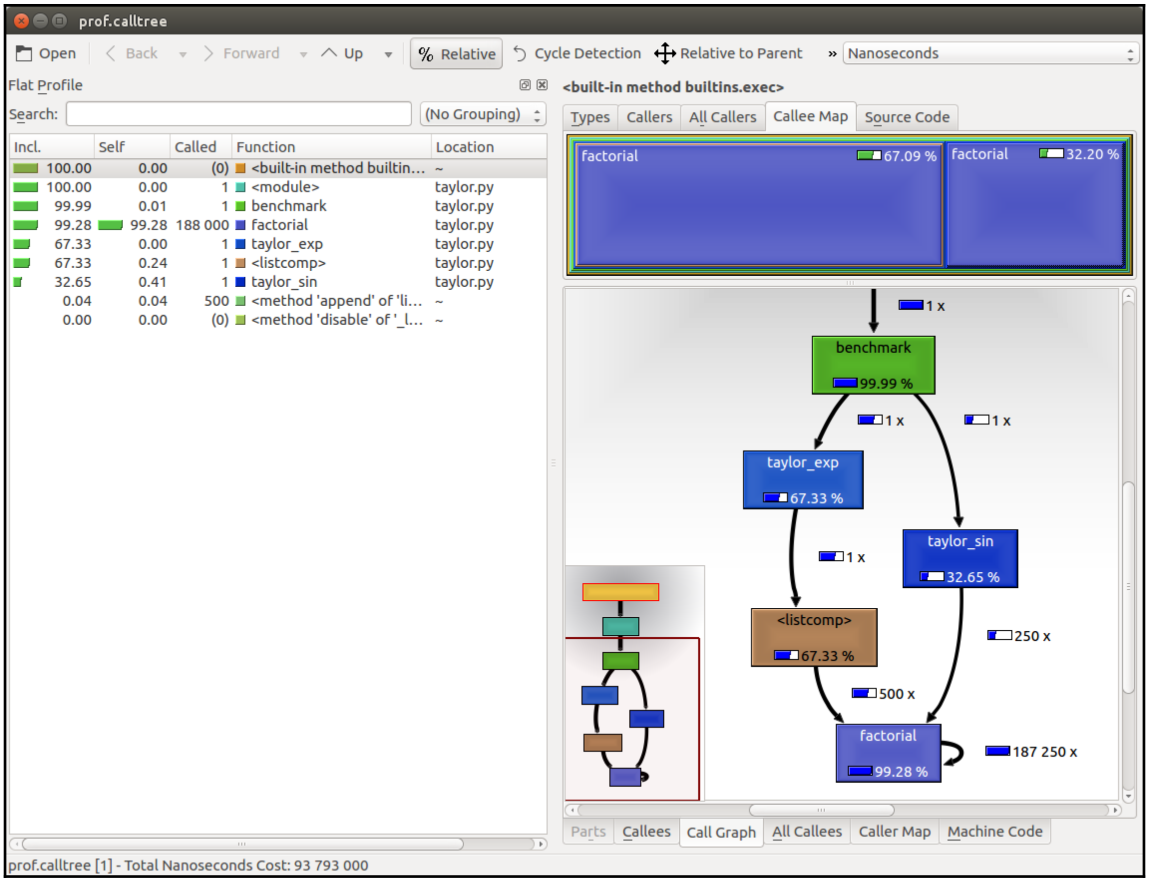


```
=============================================== test session starts ================================================
platform linux -- Python 3.5.2, pytest-3.0.5, py-1.4.32, pluggy-0.4.0
benchmark: 3.0.0 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=5.00us max_time=1.00s cal
ibration_precision=10 warmup=False warmup_iterations=100000)
rootdir: /home/gabriele/workspace/hiperf/chapter1, inifile:
plugins: benchmark-3.0.0
collected 2 items

test_simul.py .


------------------------------------------- benchmark: 1 tests ------------------------------------------
Name (time in ms)         Min       Max      Mean  StdDev    Median     IQR  Outliers(*)  Rounds  Iterations
---------------------------------------------------------------------------------------------------------
test_evolve           29.4716   41.1791   30.4622  2.0234   29.9630  0.7376          2;2      34           1
---------------------------------------------------------------------------------------------------------

(*) Outliers: 1 Standard Deviation from Mean; 1.5 IQR (InterQuartile Range) from 1st Quartile and 3rd Quartile.
============================================= 1 passed in 2.52 seconds =============================================
```



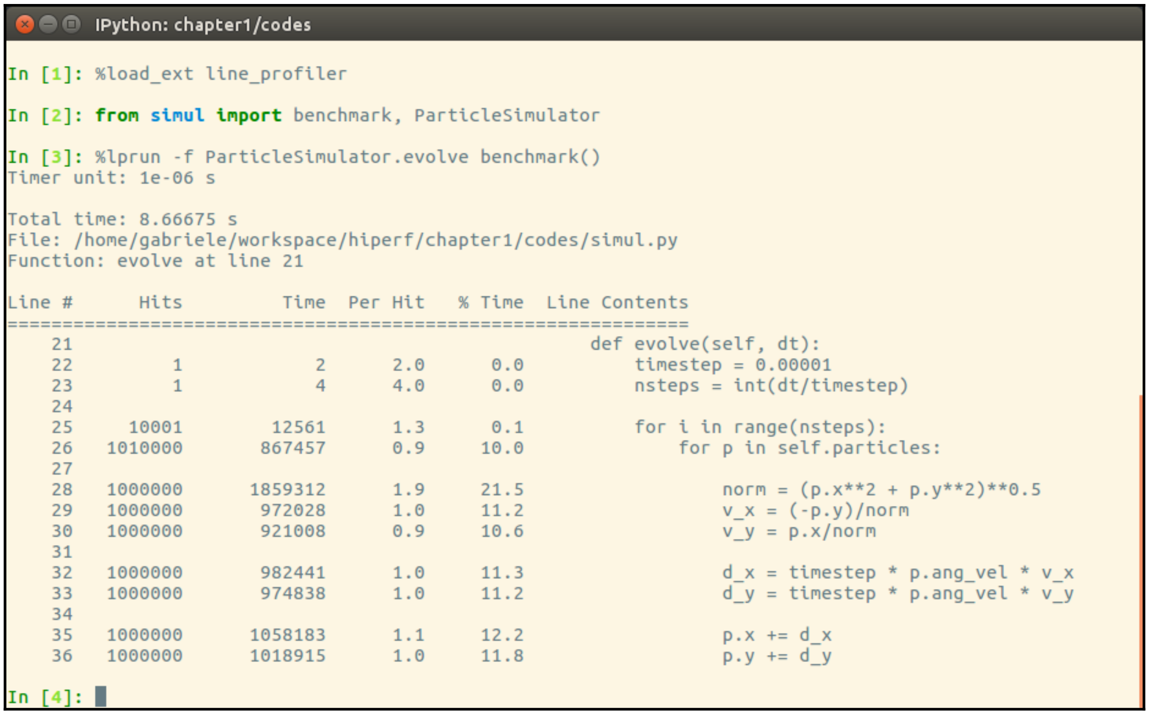
IPython: chapter1/codes

```
(hperf) ➜  codes ipython
Python 3.5.2 |Continuum Analytics, Inc.| (default, Jul  2 2016, 17:53:06)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from simul import benchmark

In [2]: %prun benchmark()
         707 function calls in 1.231 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    1.230    1.230    1.230    1.230 simul.py:21(evolve)
        1    0.000    0.000    0.001    0.001 simul.py:118(<listcomp>)
      300    0.000    0.000    0.000    0.000 random.py:342(uniform)
      100    0.000    0.000    0.000    0.000 simul.py:10(__init__)
      300    0.000    0.000    0.000    0.000 {method 'random' of '_random.Random' objects}
        1    0.000    0.000    1.231    1.231 {built-in method builtins.exec}
        1    0.000    0.000    1.231    1.231 <string>:1(<module>)
        1    0.000    0.000    1.231    1.231 simul.py:117(benchmark)
        1    0.000    0.000    0.000    0.000 simul.py:18(__init__)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}

In [3]: 
```

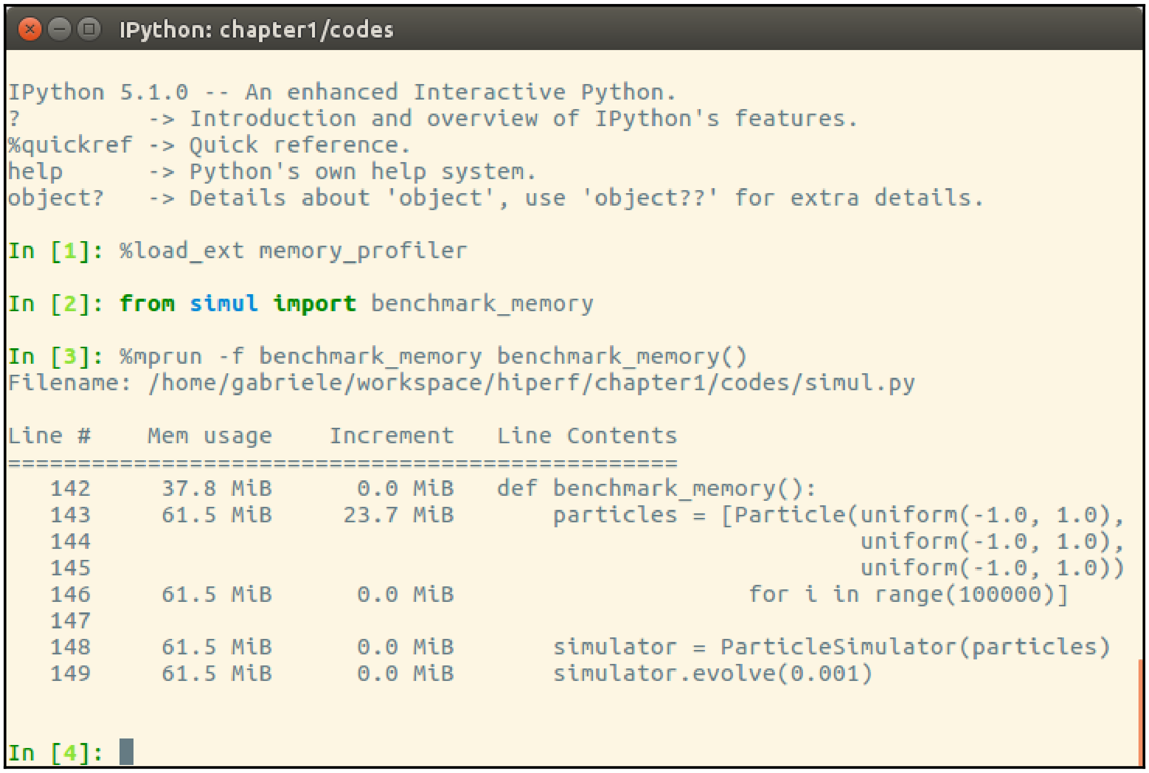


Incl.	Self	Called	Function	Location
100.00	0.00	(0)	<built-in method builtin…	~
100.00	0.00	1	<module>	taylor.py
99.99	0.01	1	benchmark	taylor.py
99.28	99.28	188 000	factorial	taylor.py
67.33	0.00	1	taylor_exp	taylor.py
67.33	0.24	1	<listcomp>	taylor.py
32.65	0.41	1	taylor_sin	taylor.py
0.04	0.04	500	<method 'append' of 'li…	~
0.00	0.00	(0)	<method 'disable' of '_l…	~

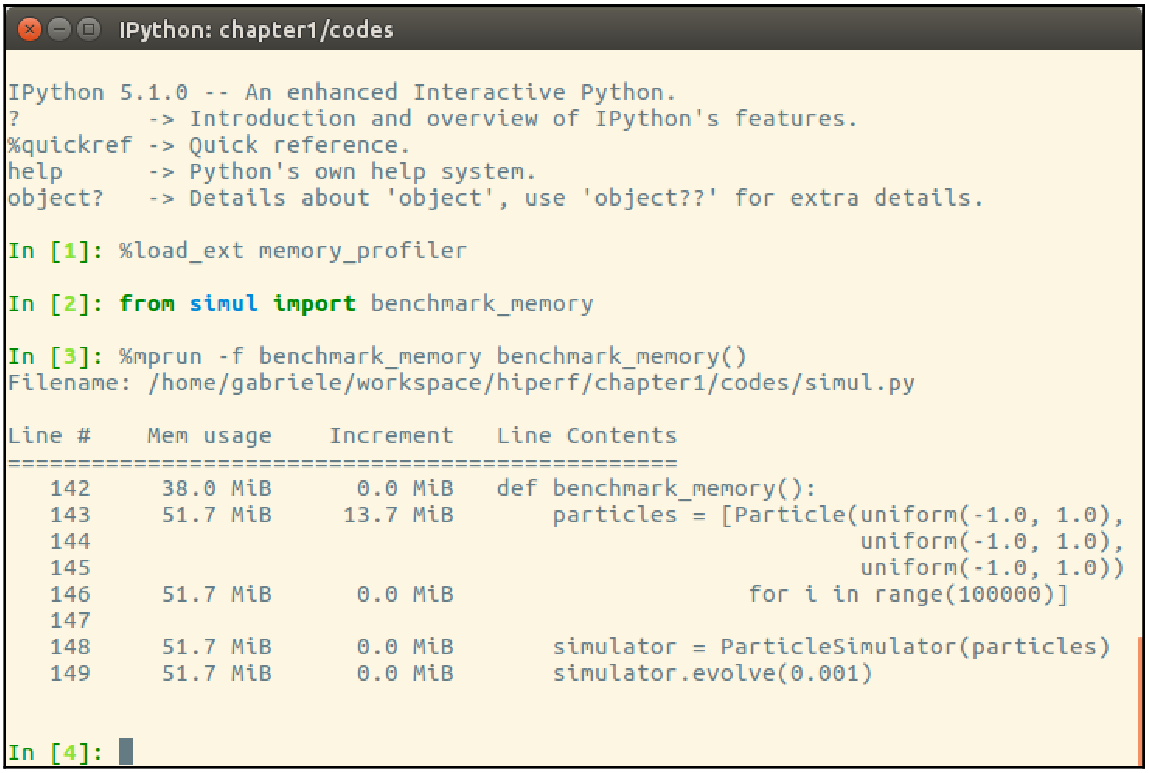
prof.calltree [1] - Total Nanoseconds Cost: 93 793 000


IPython: chapter1/codes

```
In [1]: %load_ext line_profiler

In [2]: from simul import benchmark, ParticleSimulator

In [3]: %lprun -f ParticleSimulator.evolve benchmark()
Timer unit: 1e-06 s

Total time: 8.66675 s
File: /home/gabriele/workspace/hiperf/chapter1/codes/simul.py
Function: evolve at line 21

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    21                                           def evolve(self, dt):
    22         1            2      2.0      0.0      timestep = 0.00001
    23         1            4      4.0      0.0      nsteps = int(dt/timestep)
    24
    25     10001        12561      1.3      0.1      for i in range(nsteps):
    26   1010000       867457      0.9     10.0          for p in self.particles:
    27
    28   1000000      1859312      1.9     21.5              norm = (p.x**2 + p.y**2)**0.5
    29   1000000       972028      1.0     11.2              v_x = (-p.y)/norm
    30   1000000       921008      0.9     10.6              v_y = p.x/norm
    31
    32   1000000       982441      1.0     11.3              d_x = timestep * p.ang_vel * v_x
    33   1000000       974838      1.0     11.2              d_y = timestep * p.ang_vel * v_y
    34
    35   1000000      1058183      1.1     12.2              p.x += d_x
    36   1000000      1018915      1.0     11.8              p.y += d_y

In [4]: 
```



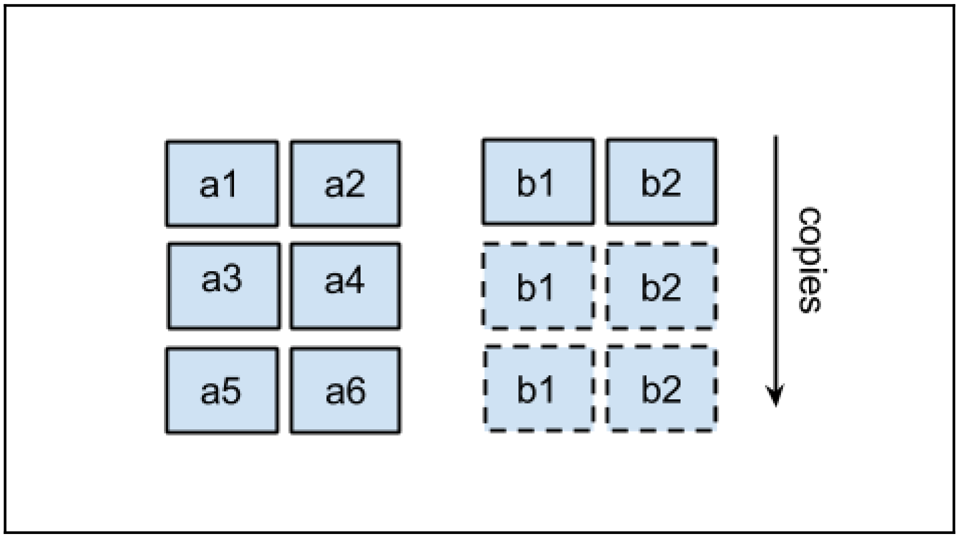
```
IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: %load_ext memory_profiler

In [2]: from simul import benchmark_memory

In [3]: %mprun -f benchmark_memory benchmark_memory()
Filename: /home/gabriele/workspace/hiperf/chapter1/codes/simul.py

Line #    Mem usage    Increment   Line Contents
================================================
   142     37.8 MiB      0.0 MiB   def benchmark_memory():
   143     61.5 MiB     23.7 MiB       particles = [Particle(uniform(-1.0, 1.0),
   144                                                       uniform(-1.0, 1.0),
   145                                                       uniform(-1.0, 1.0))
   146     61.5 MiB      0.0 MiB                    for i in range(100000)]
   147
   148     61.5 MiB      0.0 MiB       simulator = ParticleSimulator(particles)
   149     61.5 MiB      0.0 MiB       simulator.evolve(0.001)


In [4]:
```

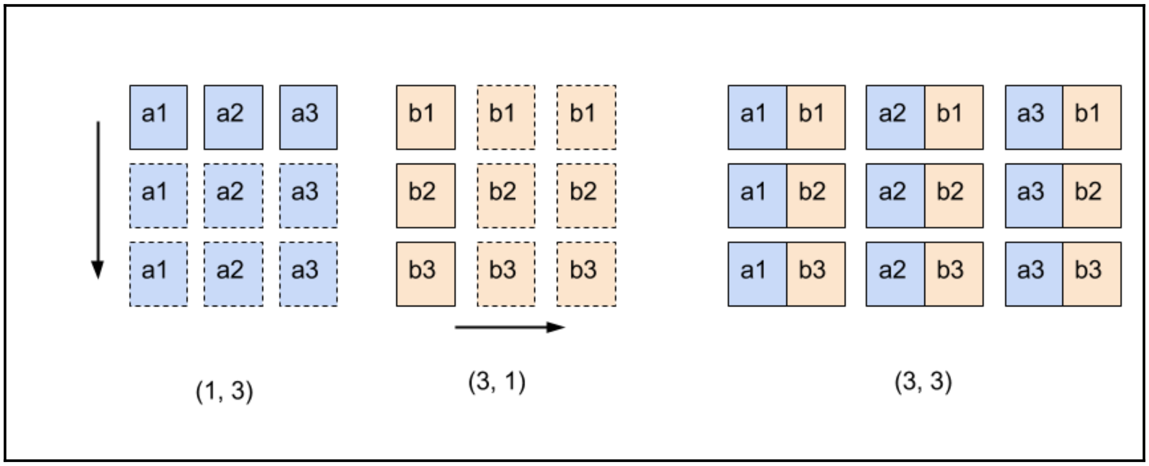


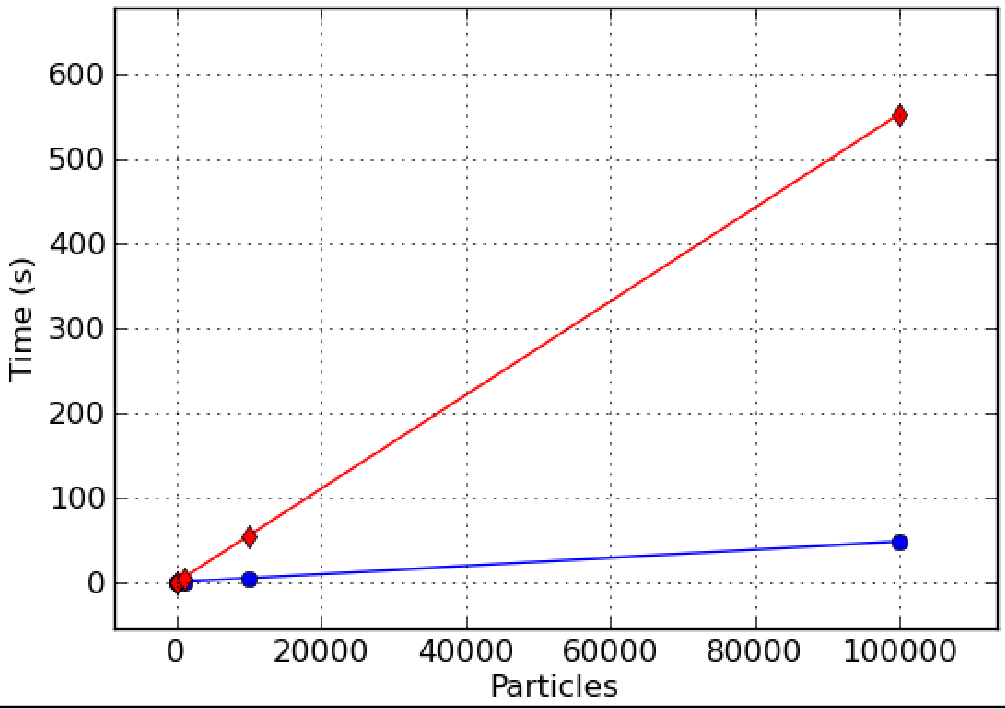
IPython: chapter1/codes

```
IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: %load_ext memory_profiler

In [2]: from simul import benchmark_memory

In [3]: %mprun -f benchmark_memory benchmark_memory()
Filename: /home/gabriele/workspace/hiperf/chapter1/codes/simul.py

Line #    Mem usage    Increment   Line Contents
================================================
   142     38.0 MiB      0.0 MiB   def benchmark_memory():
   143     51.7 MiB     13.7 MiB       particles = [Particle(uniform(-1.0, 1.0),
   144                                                       uniform(-1.0, 1.0),
   145                                                       uniform(-1.0, 1.0))
   146     51.7 MiB      0.0 MiB                    for i in range(100000)]
   147
   148     51.7 MiB      0.0 MiB       simulator = ParticleSimulator(particles)
   149     51.7 MiB      0.0 MiB       simulator.evolve(0.001)


In [4]:
```

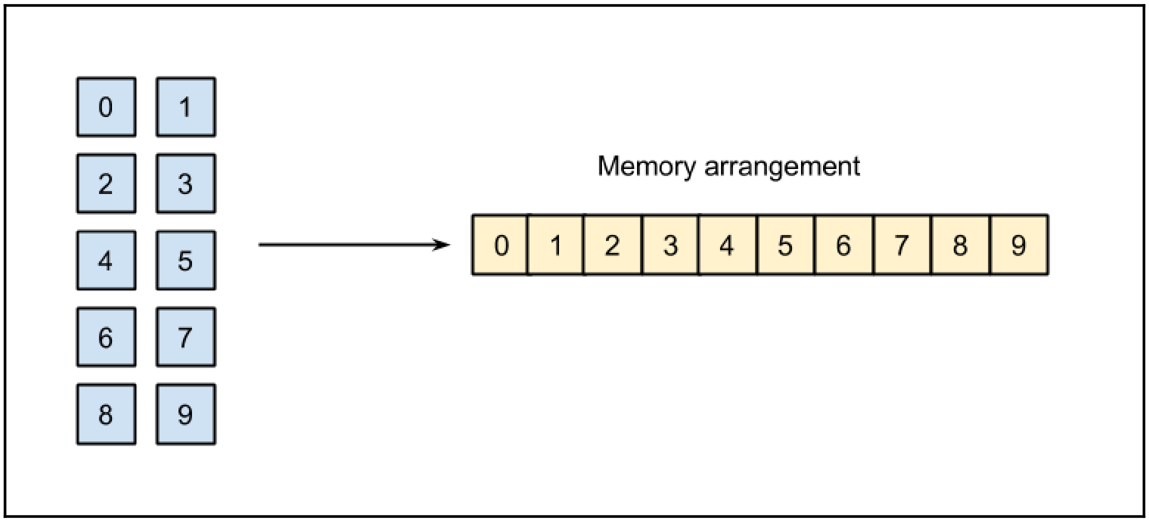
Chapter 3: Fast Array Operations with NumPy and Pandas

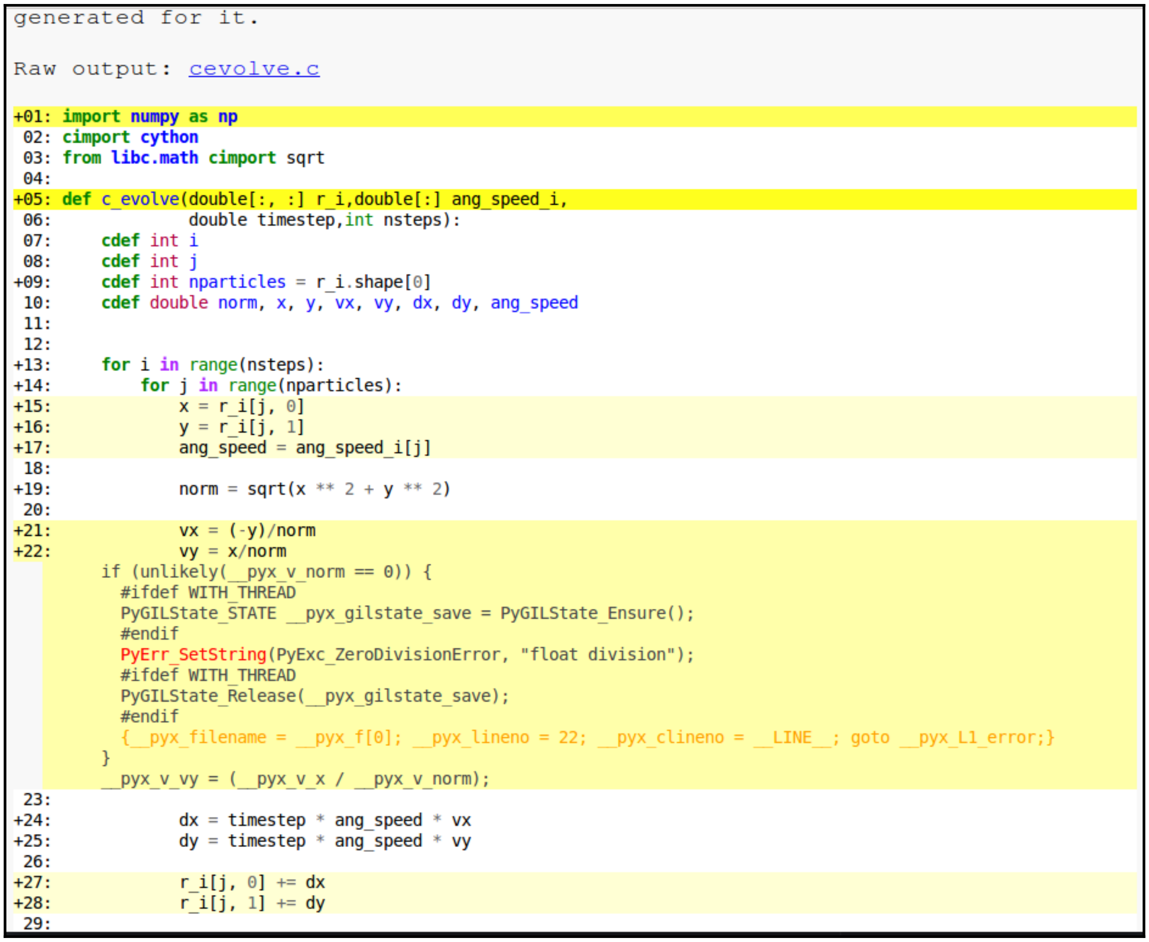




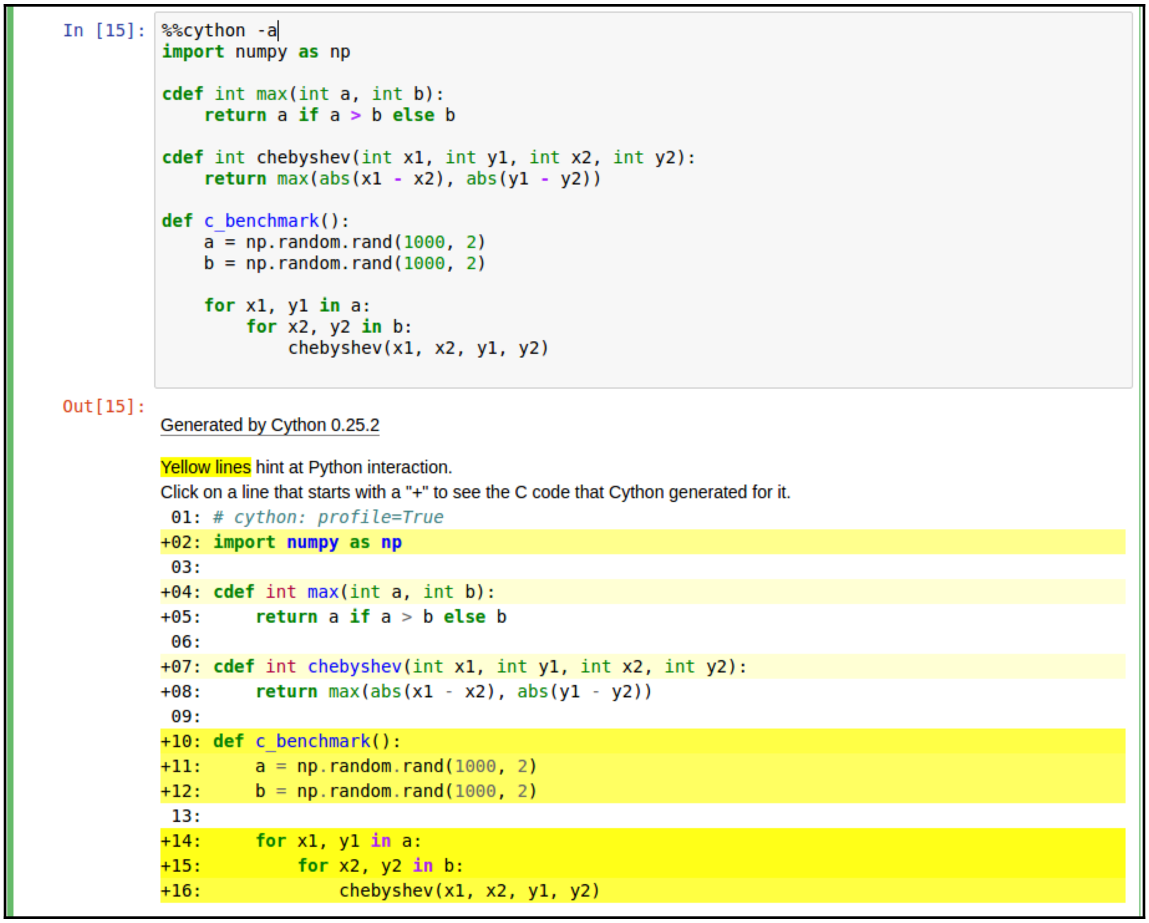

Time (s)
600
500
400
300
200
100
0
0
20000
40000
60000
80000
100000
Particles

Chapter 4: C Performance with Cython





```
generated for it.

Raw output: cevolve.c

import numpy as np
cimport cython
from libc.math cimport sqrt

def c_evolve(double[:, :] r_i,double[:] ang_speed_i,
             double timestep,int nsteps):
    cdef int i
    cdef int j
    cdef int nparticles = r_i.shape[0]
    cdef double norm, x, y, vx, vy, dx, dy, ang_speed


    for i in range(nsteps):
        for j in range(nparticles):
            x = r_i[j, 0]
            y = r_i[j, 1]
            ang_speed = ang_speed_i[j]

            norm = sqrt(x ** 2 + y ** 2)

            vx = (-y)/norm
            vy = x/norm
    if (unlikely(__pyx_v_norm == 0)) {
      #ifdef WITH_THREAD
      PyGILState_STATE __pyx_gilstate_save = PyGILState_Ensure();
      #endif
      PyErr_SetString(PyExc_ZeroDivisionError, "float division");
      #ifdef WITH_THREAD
      PyGILState_Release(__pyx_gilstate_save);
      #endif
      {__pyx_filename = __pyx_f[0]; __pyx_lineno = 22; __pyx_clineno = __LINE__; goto __pyx_L1_error;}
    }
    __pyx_v_vy = (__pyx_v_x / __pyx_v_norm);

            dx = timestep * ang_speed * vx
            dy = timestep * ang_speed * vy

            r_i[j, 0] += dx
            r_i[j, 1] += dy

```



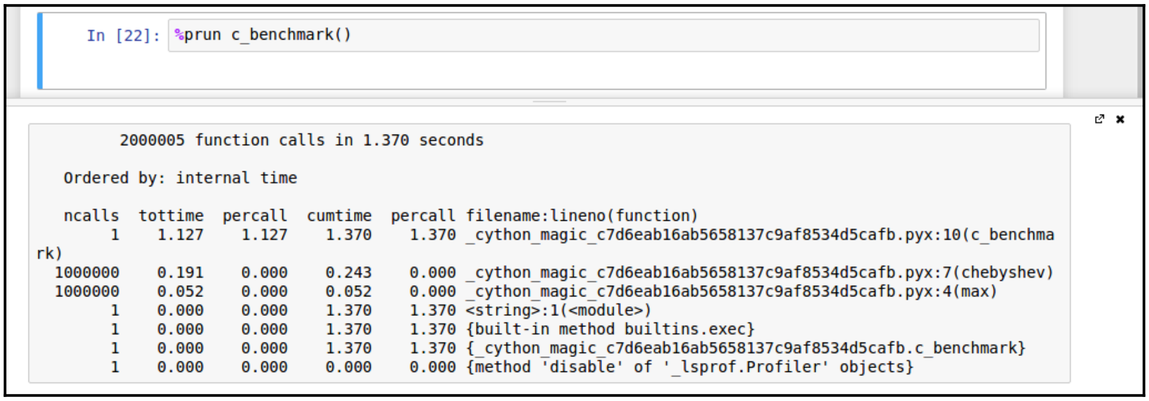
In [15]:
```
%%cython -a
import numpy as np

cdef int max(int a, int b):
    return a if a > b else b

cdef int chebyshev(int x1, int y1, int x2, int y2):
    return max(abs(x1 - x2), abs(y1 - y2))

def c_benchmark():
    a = np.random.rand(1000, 2)
    b = np.random.rand(1000, 2)

    for x1, y1 in a:
        for x2, y2 in b:
            chebyshev(x1, x2, y1, y2)
```

Out[15]:

Generated by Cython 0.25.2

Yellow lines hint at Python interaction.
Click on a line that starts with a "+" to see the C code that Cython generated for it.

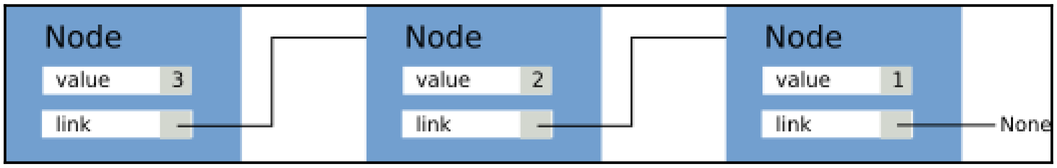
```
 01: # cython: profile=True
+02: import numpy as np
 03:
+04: cdef int max(int a, int b):
+05:     return a if a > b else b
 06:
+07: cdef int chebyshev(int x1, int y1, int x2, int y2):
+08:     return max(abs(x1 - x2), abs(y1 - y2))
 09:
+10: def c_benchmark():
+11:     a = np.random.rand(1000, 2)
+12:     b = np.random.rand(1000, 2)
 13:
+14:     for x1, y1 in a:
+15:         for x2, y2 in b:
+16:             chebyshev(x1, x2, y1, y2)
```



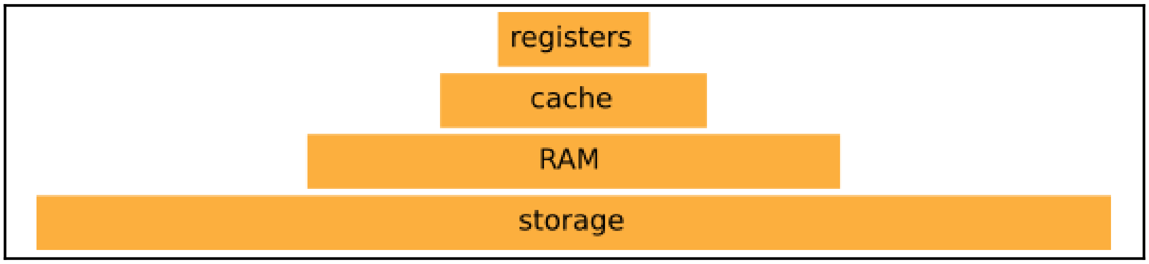
In [22]:
```
%prun c_benchmark()
```

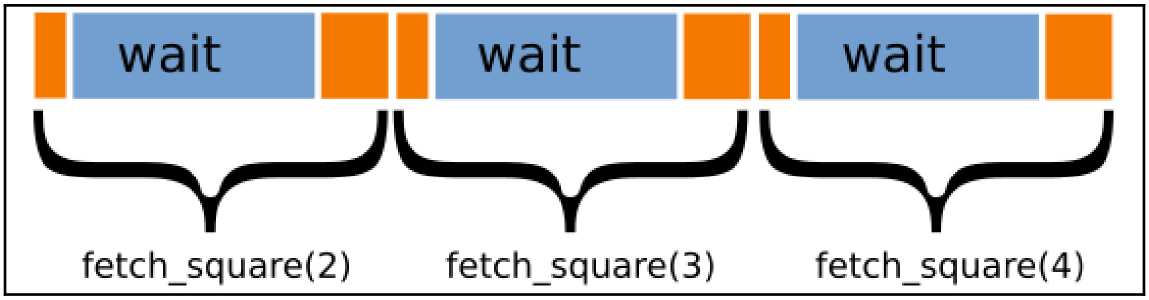
```
         2000005 function calls in 1.370 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    1.127    1.127    1.370    1.370 _cython_magic_c7d6eab16ab5658137c9af8534d5cafb.pyx:10(c_benchma
rk)
  1000000    0.191    0.000    0.243    0.000 _cython_magic_c7d6eab16ab5658137c9af8534d5cafb.pyx:7(chebyshev)
  1000000    0.052    0.000    0.052    0.000 _cython_magic_c7d6eab16ab5658137c9af8534d5cafb.pyx:4(max)
        1    0.000    0.000    1.370    1.370 <string>:1(<module>)
        1    0.000    0.000    1.370    1.370 {built-in method builtins.exec}
        1    0.000    0.000    1.370    1.370 {_cython_magic_c7d6eab16ab5658137c9af8534d5cafb.c_benchmark}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

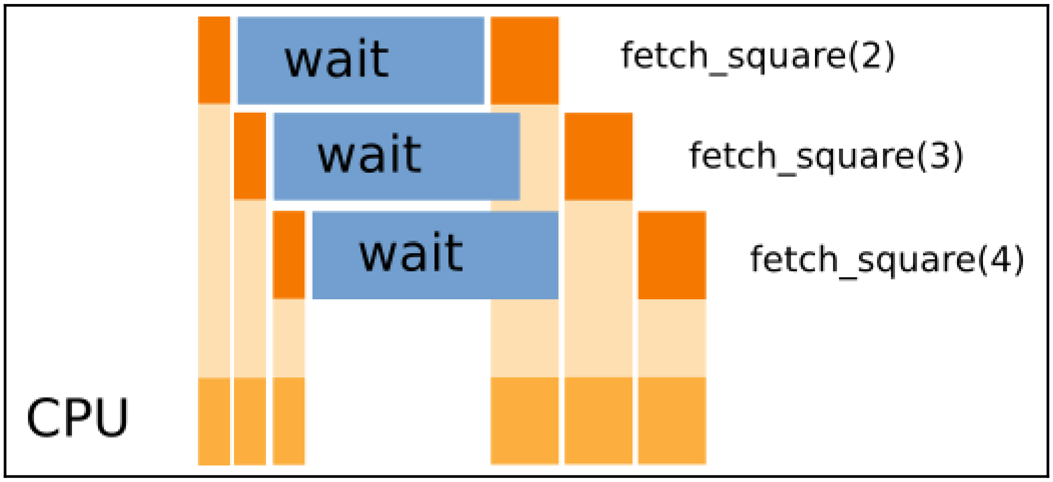
Chapter 5: Exploring Compilers

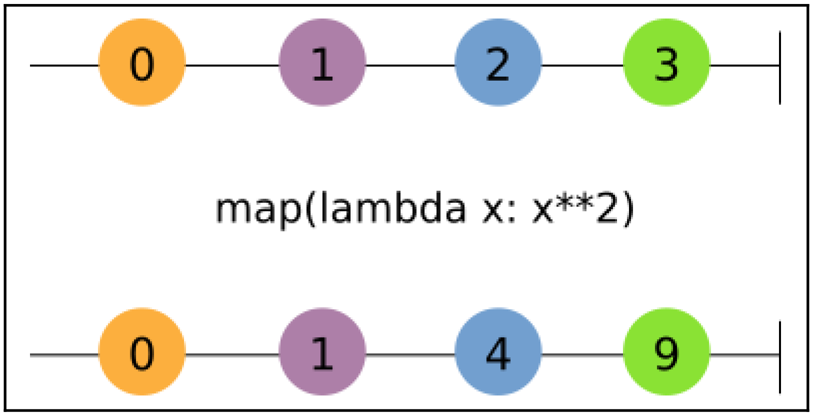


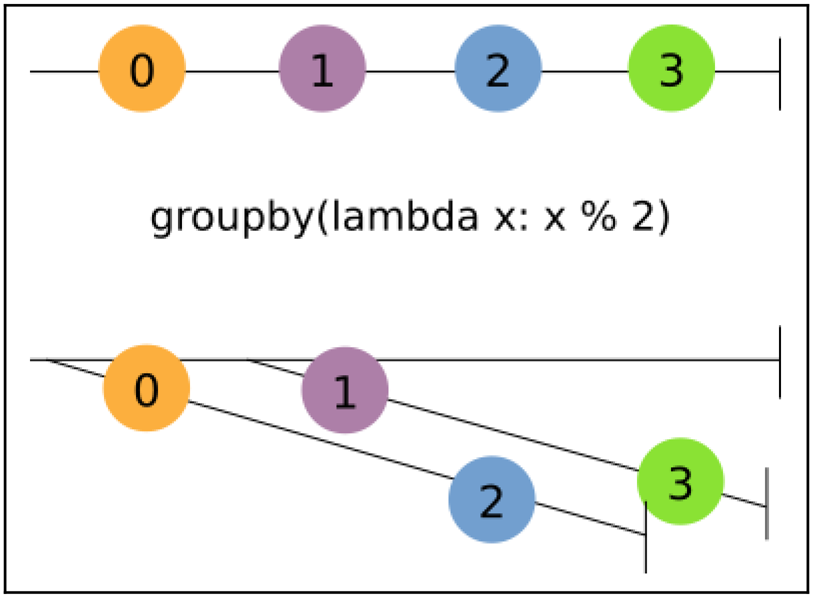
Chapter 6: Implementing Concurrency

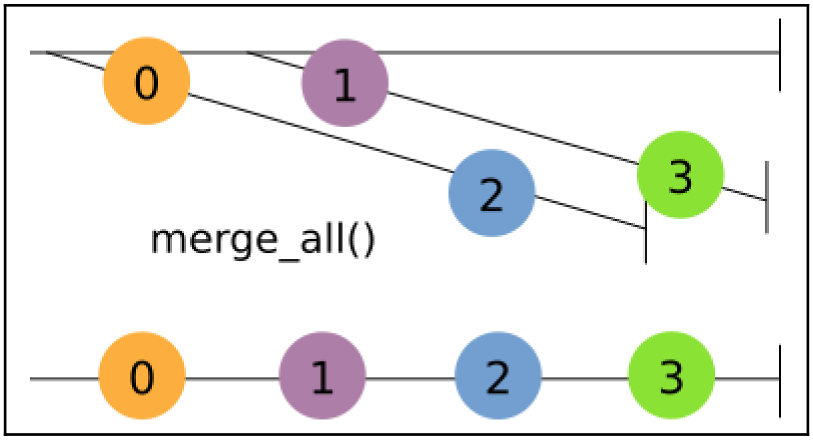


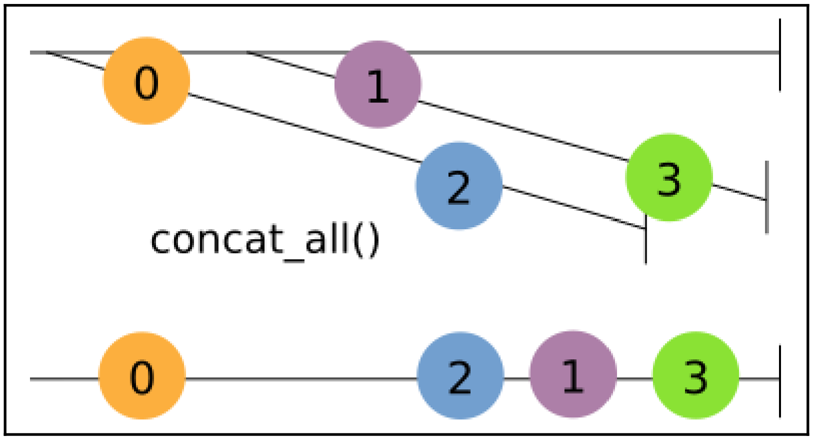


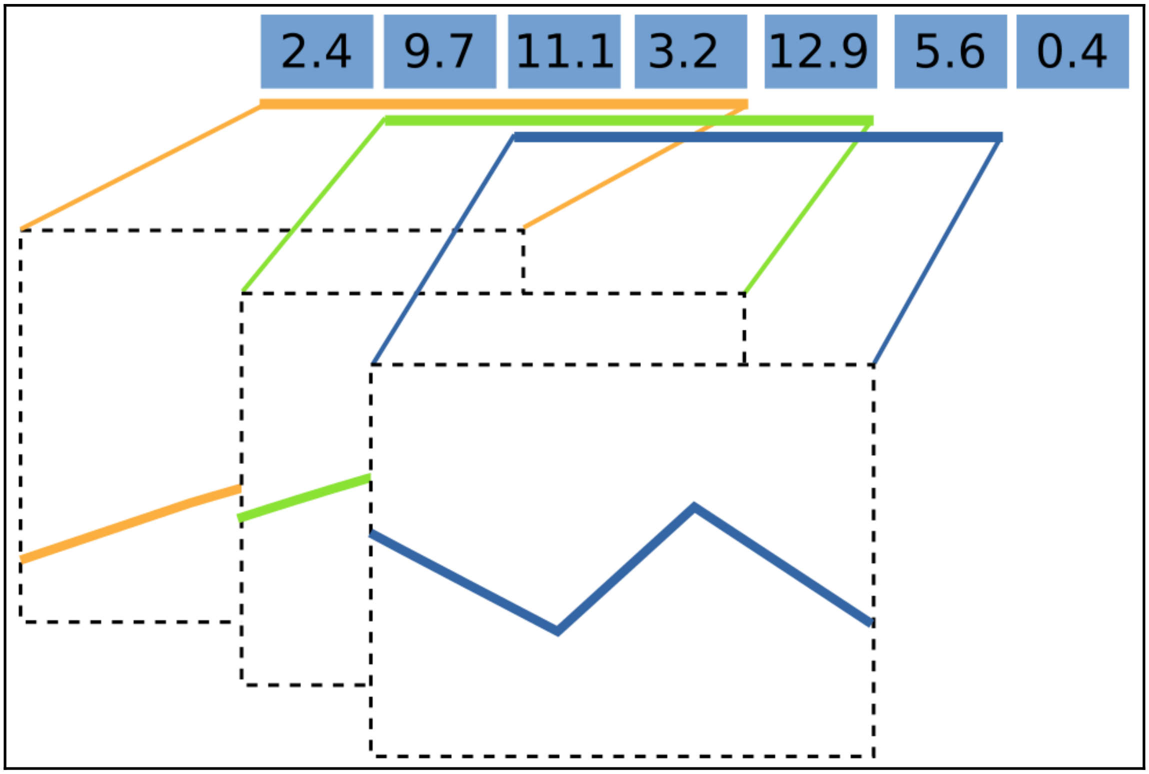



0
1
2
3
map(lambda x: x**2)
0
1
4
9

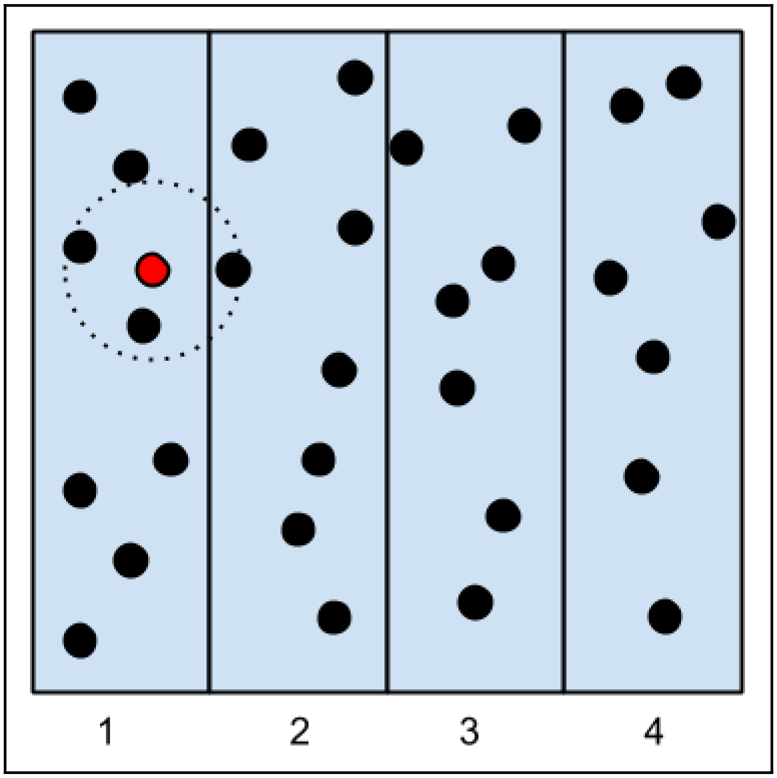

0
1
2
3
groupby(lambda x: x % 2)
0
1
2
3

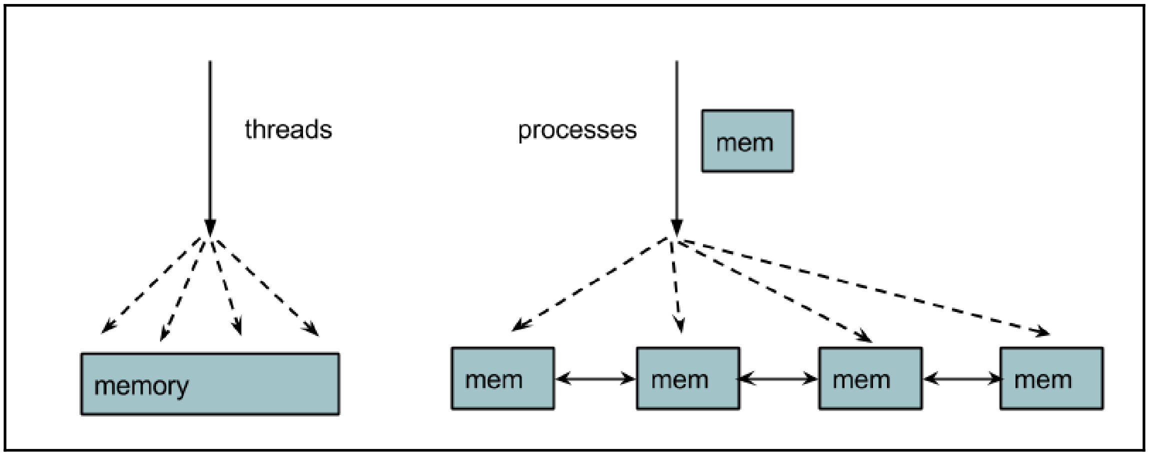

0
1
2
3
merge_all()
0
1
2
3

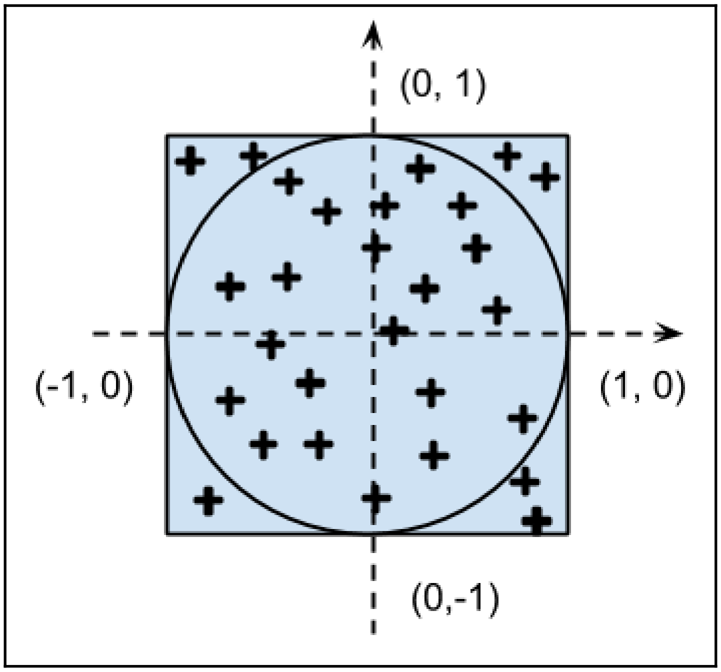

0
1
2
3
concat_all()
0
2
1
3

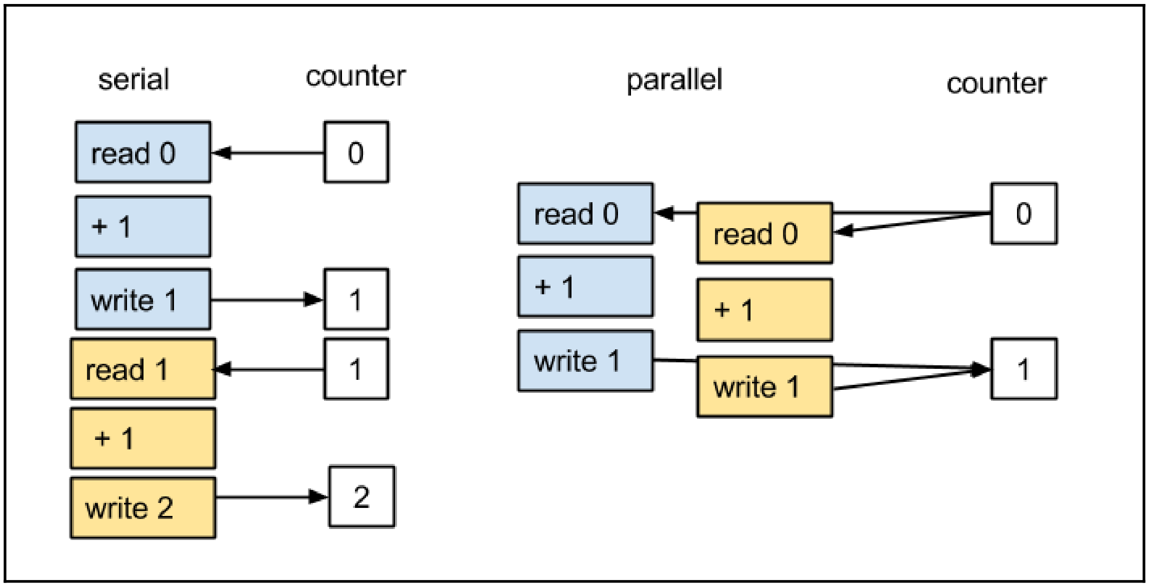

2.4
9.7
11.1
3.2
12.9
5.6
0.4

Chapter 7: Parallel Processing

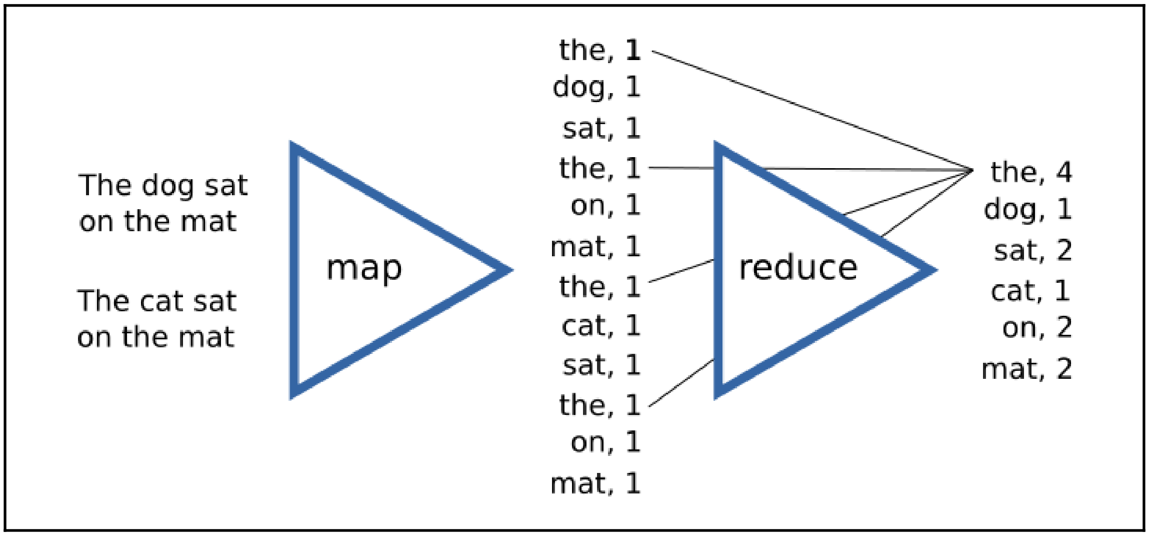


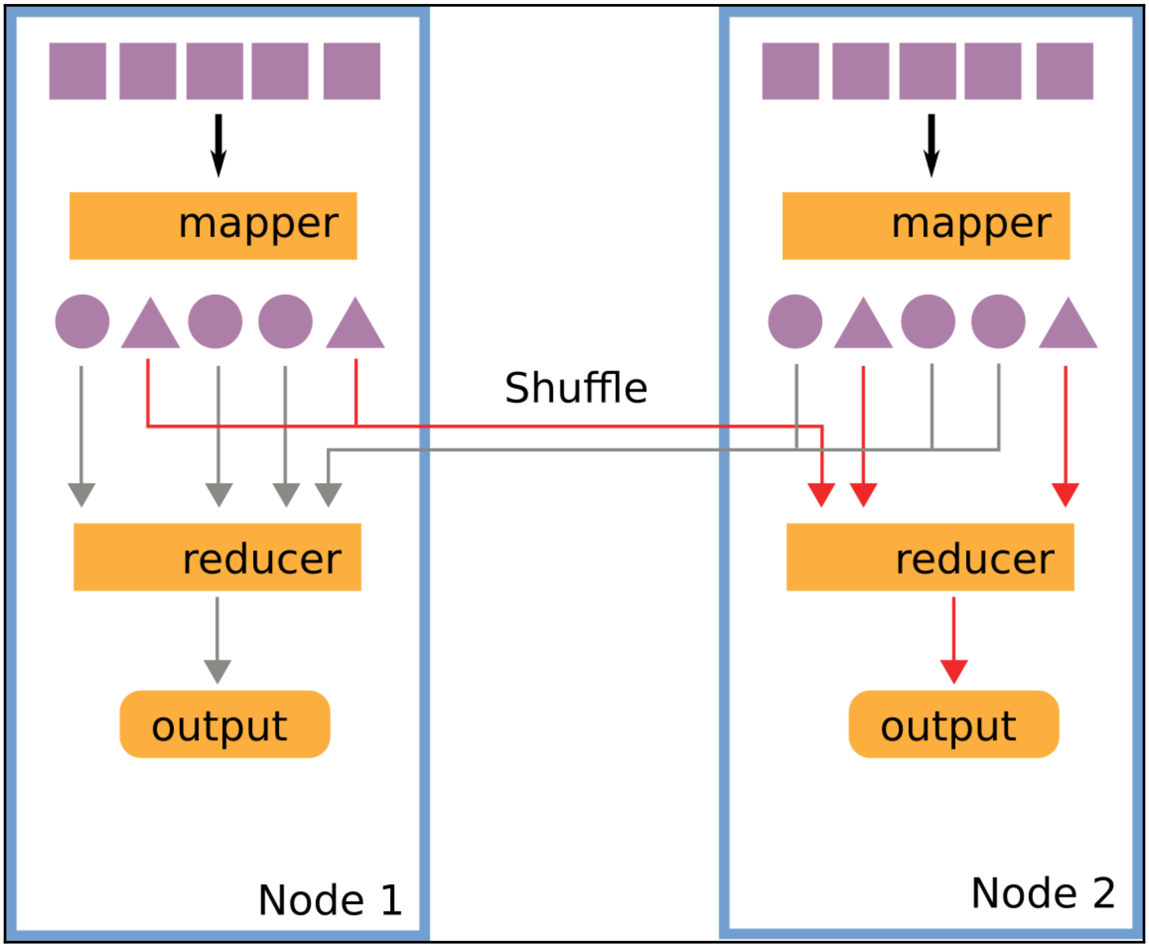

threads
processes
mem
memory
mem
mem
mem
mem

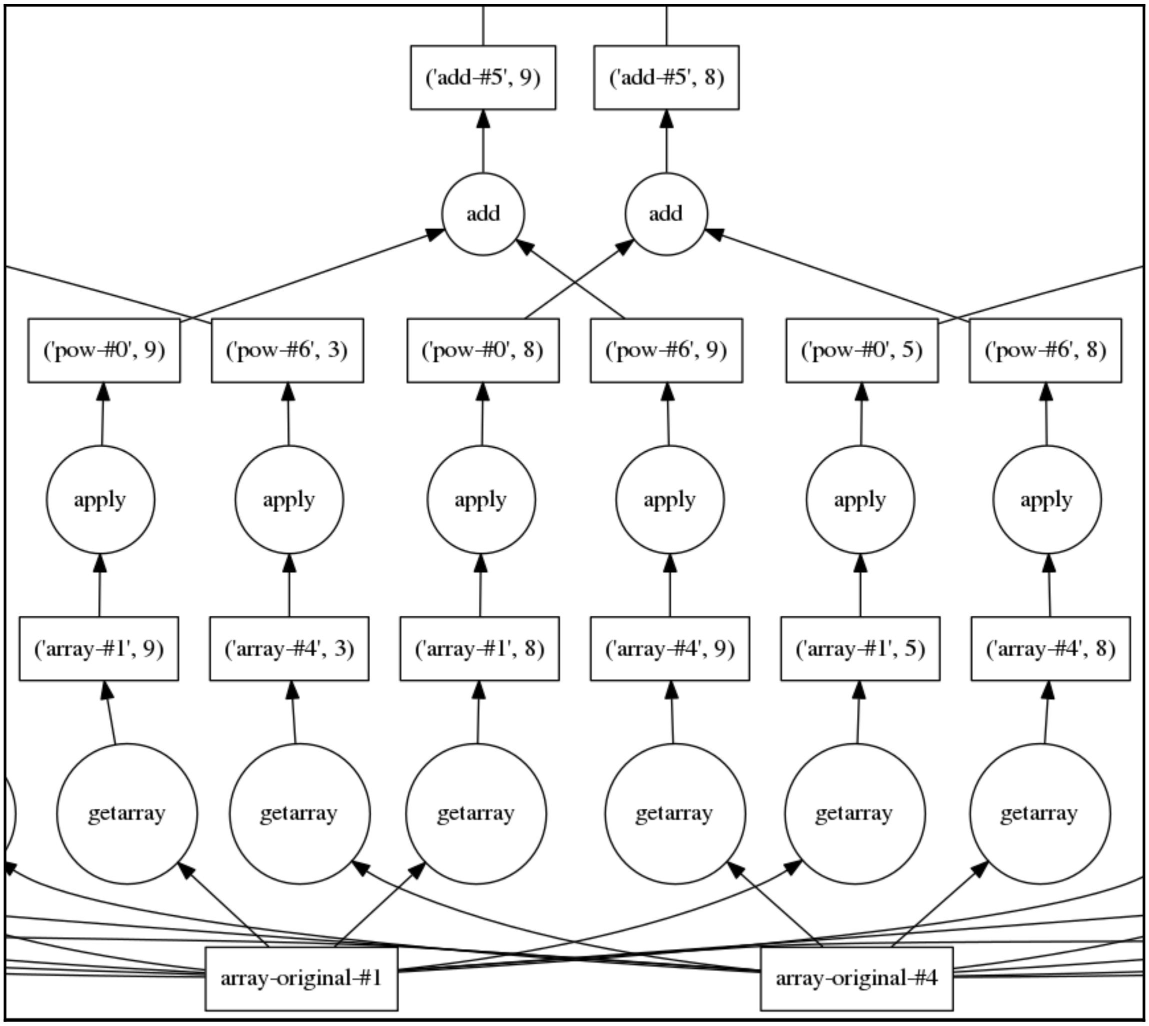

(0, 1)
(-1, 0)
(1, 0)
(0,-1)

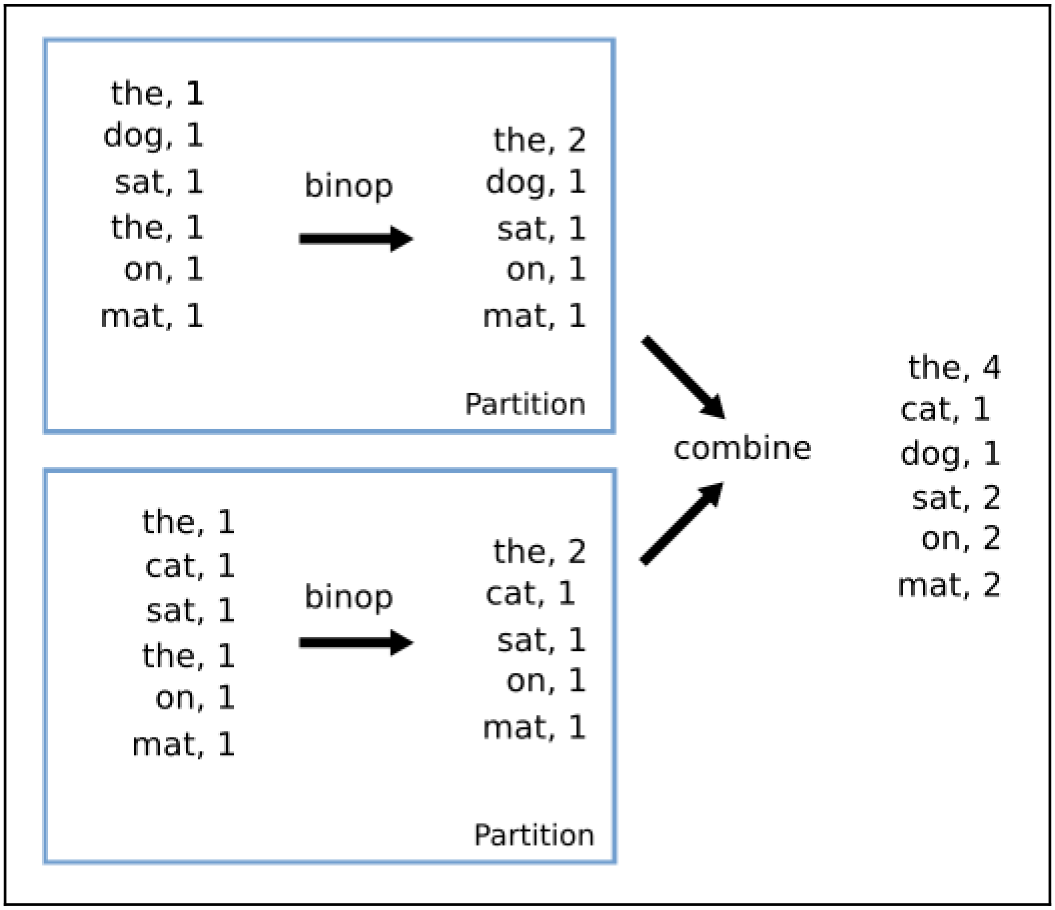

serial
counter
read 0
0
+ 1
write 1
1
read 1
1
+ 1
write 2
2
parallel
counter
read 0
read 0
0
+ 1
+ 1
write 1
write 1
1

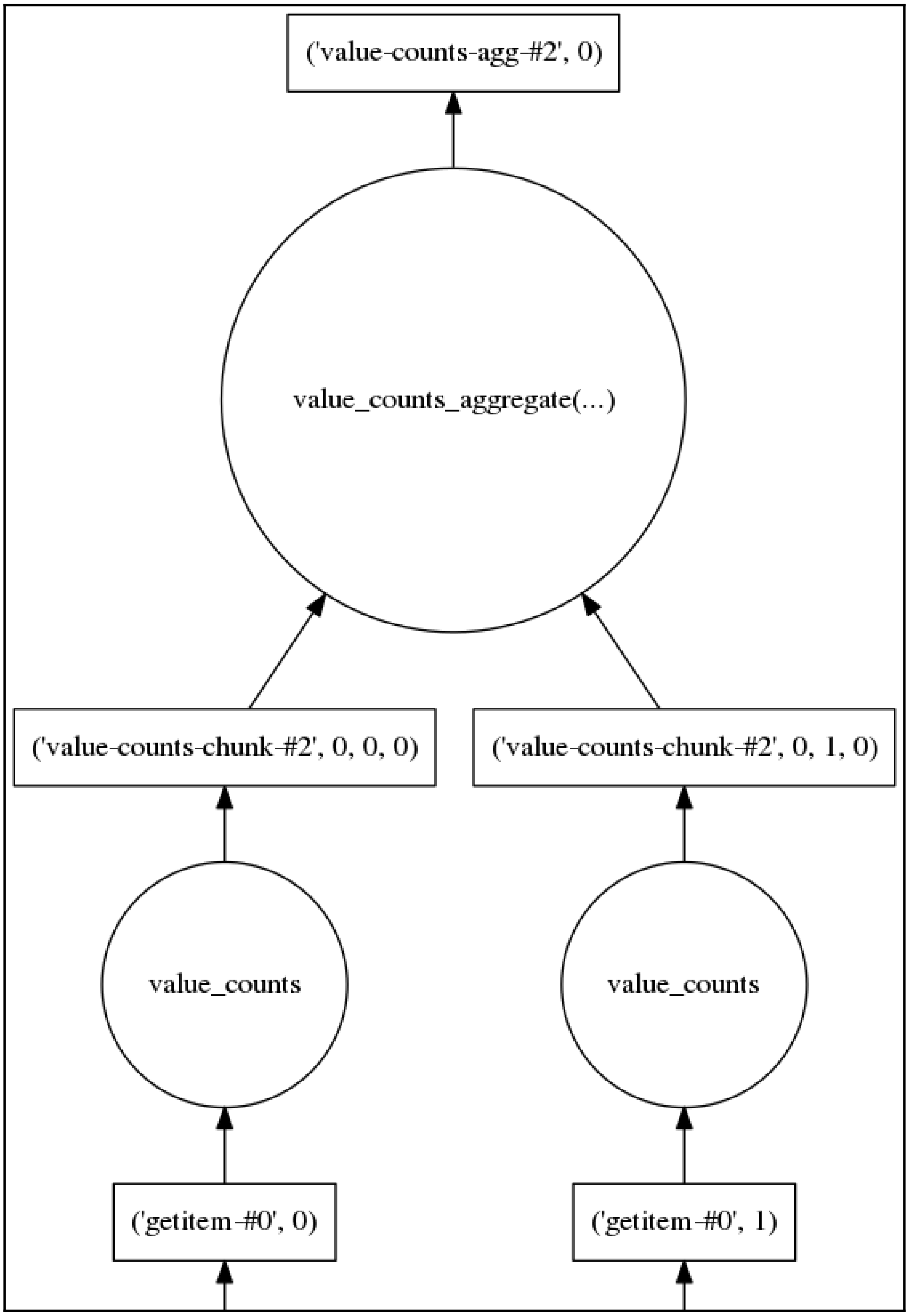
Chapter 8: Distributed Processing

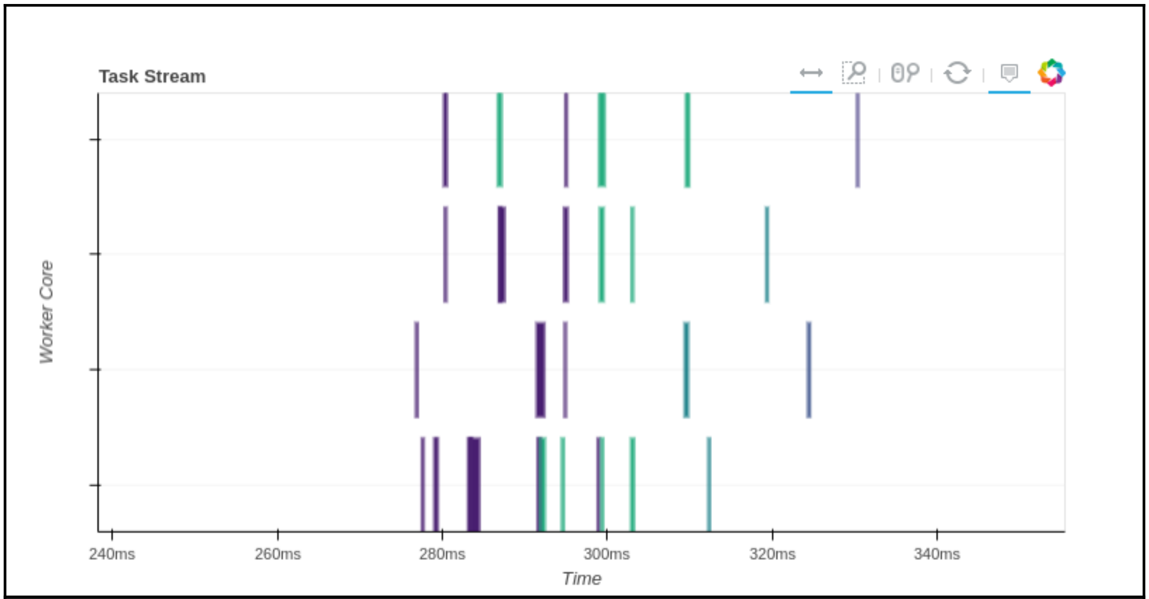


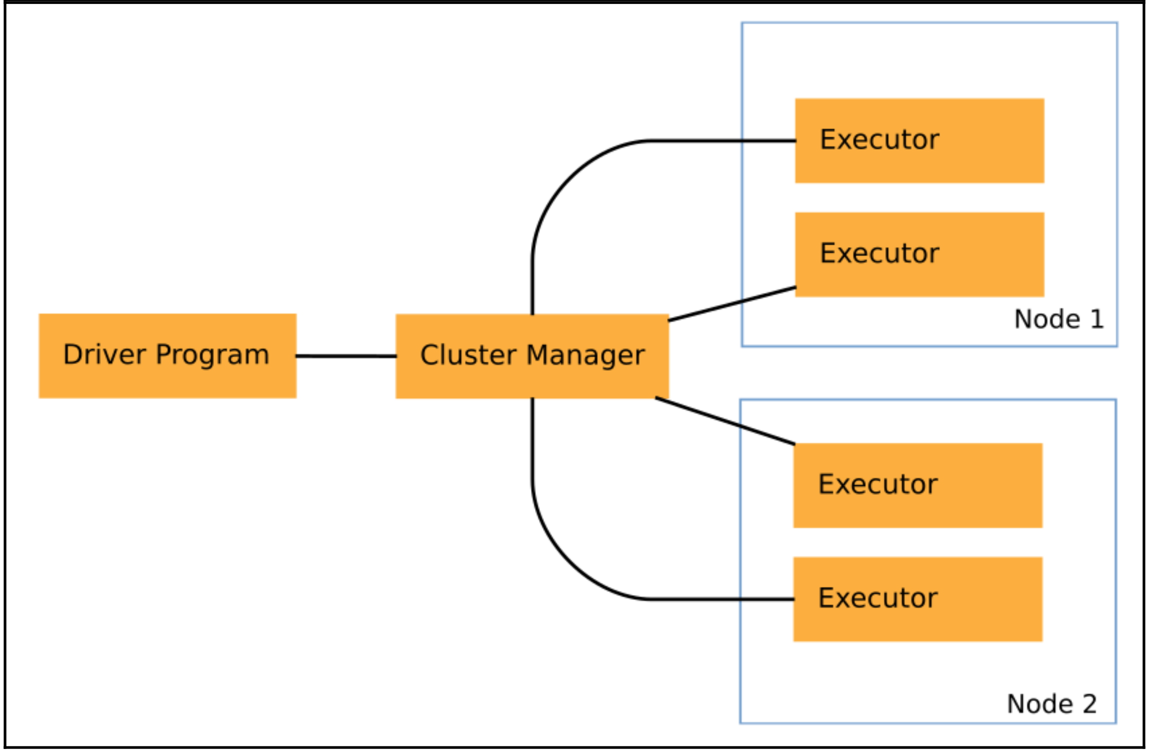

mapper
mapper
Shuffle
reducer
reducer
output
output
Node 1
Node 2

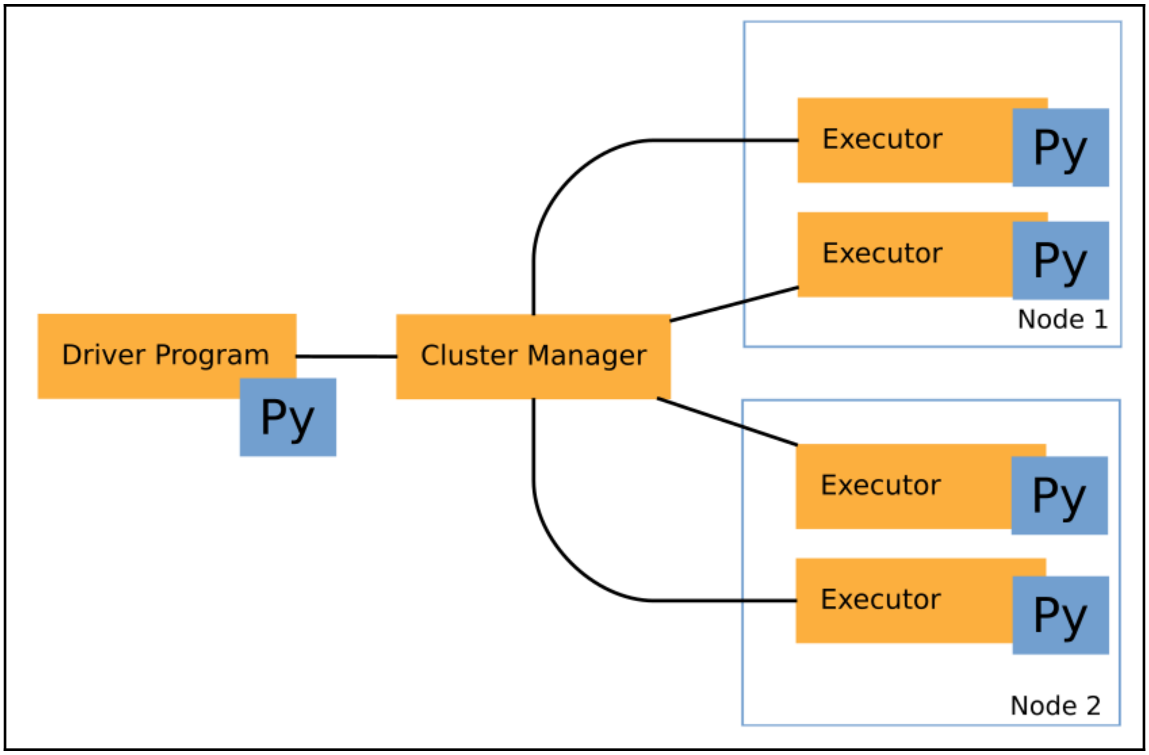

('add-#5', 9)
('add-#5', 8)
add
add
('pow-#0', 9)
('pow-#6', 3)
('pow-#0', 8)
('pow-#6', 9)
('pow-#0', 5)
('pow-#6', 8)
apply
apply
apply
apply
apply
apply
('array-#1', 9)
('array-#4', 3)
('array-#1', 8)
('array-#4', 9)
('array-#1', 5)
('array-#4', 8)
getarray
getarray
getarray
getarray
getarray
getarray
array-original-#1
array-original-#4


the, 1
dog, 1
sat, 1
the, 1
on, 1
mat, 1
binop
the, 2
dog, 1
sat, 1
on, 1
mat, 1
Partition
the, 1
cat, 1
sat, 1
the, 1
on, 1
mat, 1
binop
the, 2
cat, 1
sat, 1
on, 1
mat, 1
Partition
combine
the, 4
cat, 1
dog, 1
sat, 2
on, 2
mat, 2


('value-counts-agg-#2', 0)
value_counts_aggregate(...)
('value-counts-chunk-#2', 0, 0, 0)
('value-counts-chunk-#2', 0, 1, 0)
value_counts
value_counts
('getitem-#0', 0)
('getitem-#0', 1)


Task Stream
Worker Core
240ms
260ms
280ms
300ms
320ms
340ms
Time


Driver Program
Cluster Manager
Executor
Executor
Node 1
Executor
Executor
Node 2


Executor
Py
Executor
Py
Node 1
Driver Program
Py
Cluster Manager
Executor
Py
Executor
Py
Node 2