In [46]:

```
# import libraries and show multiple cores/engines
from IPython import parallel
clients = parallel.Client()
clients.block = True
print clients.ids
```

```
[0, 1, 2, 3]
```

In [47]:

```
# import rest of the libraries
import numpy as np
import random
import os
import sys
```

In [98]:

```
# create a view object
dview = clients.direct_view()

# trials per core
N = 10

# save 40 random prices
random_prices = []

# generate random stock prices
for i in range(N):
    %px import numpy as np
    %px rand = np.random.random()*5+10
    random_prices.append(rand)
    rand = dview.gather('rand')
    print rand
```

```
[12.3242254626159, 11.566305120131927, 12.154693361363048, 12.620842376204873]
[13.542706550545166, 10.13313002624486, 13.782977061362859, 13.254420830263099]
[13.622786966861462, 11.544954267061943, 14.114578950981455, 13.171697071533416]
[13.578091970657066, 12.992950621375446, 14.915559500229016, 11.373960058008779]
[12.01738883456564, 11.88719903221514, 13.77926810461216, 11.25680711229165]
[12.078873665309045, 11.401193152140737, 13.536411992693038, 11.593223472585008]
[13.778466699860601, 10.418754514187542, 12.83433452231462, 13.525308220176788]
[13.85224432102109, 10.542745158982155, 12.05134731953343, 13.204400814029025]
[11.373104789145245, 10.19183672639443, 14.763939566197678, 13.716894026861258]
[12.56200952142981, 13.342093305452858, 14.818018591855967, 13.504686025411473]
```

In [99]:

```
# find mean of the random prices distribution
mu = np.mean(random_prices)
```

In [92]:

```
# find std dev of the random_prices distribution
sigma = np.std(random_prices)
sigma
```

Out[92]:

```
1.5411953225771 44
```

In [100]:

```
# hold monte carlo simulated prices
mc_simulated_prices = []

# compute mu and sigma for the random prices
%px mu = np.mean(rand)
%px sigma = np.std(rand)

for j in range(len(random_prices)):
        %px import numpy as np
        %px import random
        %px mc_rand = random.gauss(mu, sigma)
        mc_simulated_prices.append(dview.gather('mc_rand'))

print "Simulated Prices Using Monte Carlo on Parallel Engines: ", mc_simulated_pr
ices
```

```
Simulated Prices Using Monte Carlo on Parallel Engines:  [[12.562009521429809, 13
.342093305452858, 14.818018591855967, 13.504686025411473], [12.562009521429809, 1
3.342093305452858, 14.818018591855967, 13.504686025411473], [12.562009521429809,
13.342093305452858, 14.818018591855967, 13.504686025411473], [12.562009521429809,
 13.342093305452858, 14.818018591855967, 13.504686025411473], [12.562009521429809
, 13.342093305452858, 14.818018591855967, 13.504686025411473], [12.56200952142980
9, 13.342093305452858, 14.818018591855967, 13.504686025411473], [12.5620095214298
09, 13.342093305452858, 14.818018591855967, 13.504686025411473], [12.562009521429
809, 13.342093305452858, 14.818018591855967, 13.504686025411473], [12.56200952142
9809, 13.342093305452858, 14.818018591855967, 13.504686025411473], [12.5620095214
29809, 13.342093305452858, 14.818018591855967, 13.504686025411473]]
```

In [101]:

```
mc_simulated_prices_sorted = np.sort(mc_simulated_prices)
print "Sorted Monte Carlo Prices", mc_simulated_prices_sorted
```

```
Sorted Monte Carlo Prices [[ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]
 [ 12.56200952  13.34209331  13.50468603  14.81801859]]
```

In [102]:

```
first_percentile = np.percentile(mc_simulated_prices_sorted,1)
print "VAR: 99% of the time prices should stay above ", first_percentile
```

```
VAR: 99% of the time prices should stay above  12.5620095214
```

In []:

```

```