Deploy and evaluation of a market-based resource allocation system

Fernando Rodríguez Haro
frodrigu@ac.upc.edu
Universitat Politècnica de Catalunya
Report for Decentralized Distributed Systems, spring semester, 2006

Abstract

Resource allocation systems have been widely studied for a long time, in particular there are algorithms that implements economic mechanisms in order to manage the resource allocation problem. Recent efforts have implemented different approaches to assure a fair assignment of resources among a set of jobs that demands CPU processing cycles. One of them is Tycoon, a distributed market-based resource allocation system that is sponsored by HP and available to test and download. We mainly focus on deploy, prepare and run a set of experiments to evaluate the platform and make some conclusions about the economic algorithm that is based on.

1 Introduction

The allocation of resources in a Grid computing environment is a difficult task, the key issue is to implement a set of distributed mechanisms that perform the assignment and management of the shared resources in all of the participating nodes of the Grid (service providers or producers) among clients (service consumers). ¿How can we assign (or reassign) the bunch of resources to a set of jobs?, obviously every one has different requirements for every kind of resources and mostly expressed in urgency or deadline of execution.

In a distributed environment, these requirements must to be mapped (re-expressed) to specific desired resources like CPU, disk space, memory, among others. To solve this problem different approaches have been proposed and implemented; one of them takes ideas from the economic market theory and different projects are trying to implement them.

An interesting project sponsored by HP is Tycoon [1], a distributed market-based resource allocation system. Tycoon incorporates an auction as the economic mechanism where users need to set values (buy resources) that match the requirements of their jobs.

To these days, Tycoon is a prototype implemented for managing CPU cycles and uses Linux as developing platform and Xen[2] for the virtualization of CPU. Xen is a virtual machine monitor for x86 that supports execution of multiple guest operating systems with unprecedented levels of performance and resource isolation.

2 Description of the problem

The goal of this work is to deploy and evaluate Tycoon, a market based distributed resource allocation system based on *proportional share* that implements *auctions* as the economic mechanism.

Why test Tycoon?

Many approaches of resource allocation systems have been proposed, traditional non-economic approach and those that use economic mechanisms. Despite the number of

proposed algorithms few of them have been implemented and available to test. In [3] key design issues are debated about the approach of use markets to solve resource allocation problems.

Following we describe the reasons that encourage us to choose tycoon.

Availability
Tycoon has been developed in recent years and is available to download and install. Source code is public via mercurial version control system and modifications can be submitted to be added in the main repository.

Simplicity
There is a separation between resource managers (auctioneers) and user requirements.

Evolving
There are recent research studies that focus on improve or extend key core components of Tycoon system, a time optimization algorithm presented in [4] extends functionality of Tycoon.

Support
Tycoon is a project initially sponsored by HP and more recently has the participation of CERN the world's largest particle physics laboratory. Two of the core components of the systems, the bank and the SLS are offered as a public service for whom are willing to test Tycoon.

2.1 Architecture of Tycoon
Tycoon has four components. The Service Location Service (SLS), Bank, Auctioneers and Agents.

Service Location Service (SLS). Used by auctioneers (to advertise resources every 30') and agents (to locate resources).

Bank. Transfer funds from a client's account to a provider's account and makes use of signed messages.

Auctioneer: One for every node and manage local resources with no information share between auctioneers.

Agent: Acts in behalf of users and interpret user's preferences and compute the bids on the machines to maximize the user's utility.

3 Deploy of Tycoon
The information presented in this chapter is for reader convenience and follows a format of a condensed guide of the steps performed during the installation of clients an auctioneer. Most of the parts can be found quite documented in [6,7].

3.1 Tycoon clients

Install packages and be sure that the install process completes with success.

```
yum -y -c http://tycoon.hpl.hp.com/~tycoon/dl/yum/tycoon.repo install tycoon_client
```

In order to retrieve the list of service providers, tycoon clients needs to use a SLS, for this purpose HP offers a SLS in `tycoon-sls.hpl.hp.com` which is preconfigured in `/etc/tycoon/tycoon.conf` and ready to use (and recommended) with the default installation. Also you can setup your own SLS changing `SLSHostName` field and downloading the apropiate package.

In order to make bank transactions, tycoon clients needs to use a Bank, for this purpose HP offers a Bank service in `tycoon-bank.hpl.hp.com` which is preconfigured in `/etc/tycoon/tycoon.conf` and ready to use (and recommended) with the default installation. Also you can setup your own Bank changing `BankHostName` field and downloading the apropiate package.

Besides SLS and Bank, tycoon client also has to communicate with auctioneers to buy and consume resources. To allow the communication firewall(s) (Linux and/or corporative) must be configured to open ports, see next table and make it properly.

port number	protocol	rule	Tycoon component
25955	TCP	out	SLS
8899	TCP	out	Bank
24571	TCP	out	Auctioneer

Another way to traverse corporate or ISP firewalls is to configure an HTTP Proxy to allow communication with each Tycoon component, this can be done in `/etc/tycoon/tycoon.conf` changing `HTTPProxy` field to your desired HTTP proxy. Before doing this consult your network administrator in order to assure you are not violating your corporate security policies and explain what are you doing.

3.2 Bank accounts

Resource consumers and resource providers use the bank for transactions, clients and auctioneers need a bank account to spent credits and charge credits respectively. The next steps allows anyone of them to create a bank account.

Assuming you have a linux account “usuario” at host “haro” then you should generate a ssh public key and configure to log into Tycoon machines.

```
[usuario@haro ~]$ ssh-keygen -t dsa
[usuario@haro ~]$ cat  ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
[usuario@haro ~]$ chmod 600  ~/.ssh/authorized_keys
```

Now you must generate some configuration files for Tycoon with next command adding your email address and the account name desired to use on remote machines.

```
[usuario@haro ~]$ tycoon user setup ferharo@gmail.com ferharo
~/.ssh/id_dsa.pub
```

Notice that a directory structure (.tycoon) has been created in your home with some configuration files.

Now you must submit your email address and your bank_account_public_key (located in `~/.tycoon/ferharo@gmail.com/bank_account_public_key`) to your Tycoon system administrator (in this case Kevin Lai whom is responsible of the architecture and development of tycoon project), you can also describe (optional) the application you want to run, all of this in `http://tycoon.hpl.hp.com/wiki/TycoonAccountForm`. A confirmation by email will let you know that your account is ready to use.

Now verify that your client can connect to the Bank by asking for your account balance.

```
[usuario@haro ~]$ tycoon bank get_balance
Account balance: 100
```

and finally verify that your client has connection with SLS by asking the list of auctioneers.

```
[usuario@haro ~]$ tycoon get_host_list
IP Address
----------
204.123.32.58
204.123.32.42
..... etc
```

3.3 Tycoon auctioneers

Install packages and be sure that the install process completes with success.

```
yum -y -c http://tycoon.hpl.hp.com/~tycoon/dl/yum/tycoon.repo install tycoon_aucd_xen3
```

List your boot directory and take note of files that:

Start with initdr and have the word xen.
Start with vmlinux and have the word xen.
Also the file that starts with xen-3

```
[root@haro ~]$ ls /boot
config-2.6.11-1.1369_FC4
config-2.6.16-xen3_86.1_fc4
grub
initrd-2.6.11-1.1369_FC4.img
initrd-2.6.16-xen3_86.1_fc4.img
initrd-2.6-xen.img
System.map-2.6.11-1.1369_FC4
System.map-2.6.16-xen3_86.1_fc4
vmlinux-syms-2.6.16-xen3_86.1_fc4
vmlinuz-2.6.11-1.1369_FC4
vmlinuz-2.6.16-xen3_86.1_fc4
vmlinuz-2.6-xen
xen-3.0.2-2.gz
xen-3.0.gz
xen-3.gz
xen.gz
xen-syms-3.0.2-2
```

note: if you see some extra files with similar names notice that they are just symlinks.

Update the GRUB configuration in /boot/grub/grub.conf. Warning: procede with care, a mistake can disallow booting your linux installation.

If you want to make Xen booting by default change first line from 0 to 1.

```
default=1
timeout=5
splashimage=(hd0,6)/boot/grub/splash.xpm.gz
hiddenmenu
title Fedora Core (2.6.11-1.1369_FC4)
      root (hd0,6)
      kernel /boot/vmlinuz-2.6.11-1.1369_FC4 ro root=LABEL=/1 rhgb quiet
      initrd /boot/initrd-2.6.11-1.1369_FC4.img
title Xen (Xen 3.0.2-2)
      root (hd0,6)
      kernel /boot/xen-3.0.2-2.gz
      module /boot/vmlinuz-2.6.16-xen3_86.1_fc4 ro root=LABEL=/1 rhgb quiet
      module /boot/initrd-2.6.16-xen3_86.1_fc4.img
```

Copy lines of your default boot configuration for Fedora Core, change name of files according to your previous notes and save grub.conf. Also note that we have changed the word `initrd` to `module`.

Finish the installation

```
iptables -A INPUT -s \! 127.0.0.1 -p tcp --dport 8001:8002 -j REJECT
iptables -A INPUT -s \! 127.0.0.1 -p tcp --dport 9601:9699 -j REJECT
service iptables save
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Reboot and verify that Xen kernel is loaded : `uname -a`

```
2.6.16-xen3_86.1_fc4 #1 SMP Thu Apr 13 08:26:52
```

Auctioneers needs to communicate with SLS and Bank and for this purpose must to be set the configuration in the same way as was done with the client.

A resource provider has an owner and is necessary to create a bank account and make the necessary changes in configuration files for this purpose.

Then first follow the procedure to create a bank account for the auctioneer's owner. We are going to configure the account ferharo (previously created) as owner of the auctioneer.

Copy the owner's bank account key pair (located in `~/.tycoon/ferharo@gmail.com/`) to `/etc/tycoon` and rename them like following.

```
ferharo@gmail.com_bank_private_key
ferharo@gmail.com_bank_public_key
```

Copy the owner's public key to /etc/tycoon/admin_public_key

```
ferharo@gmail.com_bank_public_key
```

Set the auctioneer's owner

Change (or add) the `UserName` option in /etc/tycoon/tycoon_aucd.conf.

```
UserName = "ferharo@gmail.com"
```

Configure firewall(s) (Linux and/or external) to open ports

25955	TCP	out	SLS
8899	TCP	out	Bank
24571	TCP	in	Auctioneer

If necessary configure a HTTP Proxy

`/etc/tycoon/tycoon.conf`

Change `HTTPProxy`, "http://proxy.hpl.hp.com:8088"

Now you can stop, start, restart, or see the status of auctioneer with.

```
service tycoon_aucd [stop, start, restart, status]
```

Check that port 24571 is listed and listening

```
netstat -t -a
```

3.4 Preparing client accounts for experiments.

In order to differentiate and compare results of job execution lets create two client tycoon accounts following the procedure described in 3.2.

For user "`usuario1`" at host "`76...`" with email `fernando@haro.ac.upc.es` and account on remote machine `fernando.`

```
[usuario1@76 ~]$ ssh-keygen -t dsa
[usuario1@76 ~]$ cat .ssh/id_dsa.pub >> .ssh/authorized_keys
[usuario1@76 ~]$ chmod 600 .ssh/authorized_keys
[usuario1@76 ~]$ tycoon user setup fernando@haro.ac.upc.es fernando
~/.ssh/id_dsa.pub
```

For user "usuario2" at host "76..." with email `frodrigu@ac.upc.es` and account on remote machine `frodrigu.`

```
[usuario2@76 ~]$ ssh-keygen -t dsa
[usuario2@76 ~]$ cat .ssh/id_dsa.pub >> .ssh/authorized_keys
[usuario2@76 ~]$ chmod 600 .ssh/authorized_keys
[usuario2@76 ~]$ tycoon user setup frodrigu@ac.upc.es frodrigu
~/.ssh/id_dsa.pub
```

For accounts creation in tycoon Grid we can follow the usual procedure submitting email address and bank_account_public_key for each user to tycoon system administrator and wait for confirmation. Or we can use the already account created ferharo to create this two extra accounts and fund them (from ferharo credits) as follows.

Creating account with email and bank_account_public_key of usuario1

```
[usuario@haro ~]$ tycoon bank create_account fernando@haro.ac.upc.es
                  20 bank_account_public_key
Creating new bank account.
Created bank account with initial deposit of 20
```

Creating account with email and bank_account_public_key of usuario2

```
[usuario@haro ~]$ tycoon bank create_account frodrigu@ac.upc.es 50
                  bank_account_public_key
Creating new bank account.
Created bank account with initial deposit of 50
```

Now use commands `tycoon bank get_balance` and `tycoon get_host_list` to test new tycoon clients connection with Bank and SLS

3.5 Using Tycoon.

Now lets see some commands that clients can execute on tycoon.

3.5.1 Usuario1 commands

a) retrieving the list of auctioneers

```
[usuario1@76 ~]$ tycoon get_host_list
```

b) select one auctioneer from list and create a virtual machine on it to execute jobs.

```
[usuario1@76 ~]$ tycoon create_account 62.57.49.76 1
Creating host account(s) (may take several minutes)...
62.57.49.76 SSH port number: 60478
62.57.49.76 has booted.
62.57.49.76 created account with initial deposit of 1
```

c) log into remote tycoon machine with tycoon_ssh.

```
[usuario1@76 ~]$ tycoon_ssh fernando@62.57.49.76
```

d) Copy files (programs) into remote tycoon machine with tycoon_scp.

```
[usuario1@76 ~]$ tycoon_scp -r program 62.57.49.76
```

e) See bank transactions of usuario1 (fernando).

```
[usuario1@76 ~]$ tycoon bank get_history
```

f) When you log into a remote tycoon machine you actually log into your VM. Here you can install whatever package you want.

```
[fernando@fernando ~]yum install octave
Setting up Install Process
Setting up repositories
tycoon                    100% |=========================|  951 B
00:00
....
Installed: octave.i386 6:2.9.5-1.fc4
Dependency Installed: Glide3.i386 0:20050815-1.fc4 Glide3-l.....
Complete!
```

3.5.2 Usuario2 commands

a) retrieving the list of auctioneers

```
[usuario2@76 ~]$ tycoon get_host_list
```

b) select one auctioneer from list and create a virtual machine on it to execute jobs.

```
[usuario2@76 ~]$ tycoon host create_account 62.57.49.76 1
Creating host account(s) (may take several minutes)...
62.57.49.76 SSH port number: 15070
62.57.49.76 has booted.
62.57.49.76 created account with initial deposit of 1
```

c) log into remote tycoon machine with ssh (or putty in windows).

```
[usuario2@76 ~]$ ssh -l frodrigu -p 15070 62.57.49.76
```

d) Copy files (programs) into remote tycoon machine with scp (or WinSCP).

```
[usuario2@76 ~]$ scp -P 15070 program frodrigu@62.57.49.76:program
```

e) Ask for account status

```
[usuario2@76 ~]$ tycoon host get_account_status 62.57.54.178 user_name
balance expiration deposited
ip_address: 62.57.54.178 {'user_name': 'frodrigu@ac.upc.es',
'balance': 0.99999999369902492, 'deposited': '1', 'expiration':
2149380922L}
```

e) Shutdown your VM on auctioneer.

```
[usuario2@76 ~]$tycoon host shutdown 62.57.54.178
Shutdown virtual machine on 62.57.54.178. Graceful: True
```

e) Boot your VM on auctioneer.

```
[usuario2@76 ~]$tycoon host boot 62.57.54.178
62.57.54.178 has booted.
```

e) Delete account on auctioneer and return remaining funds to your bank account.

```
[usuario2@76 ~]$tycoon host delete_account 62.57.54.178
Deleting account(s) (may take a minute)...
62.57.54.178 deleted account.
```

4 Experiments

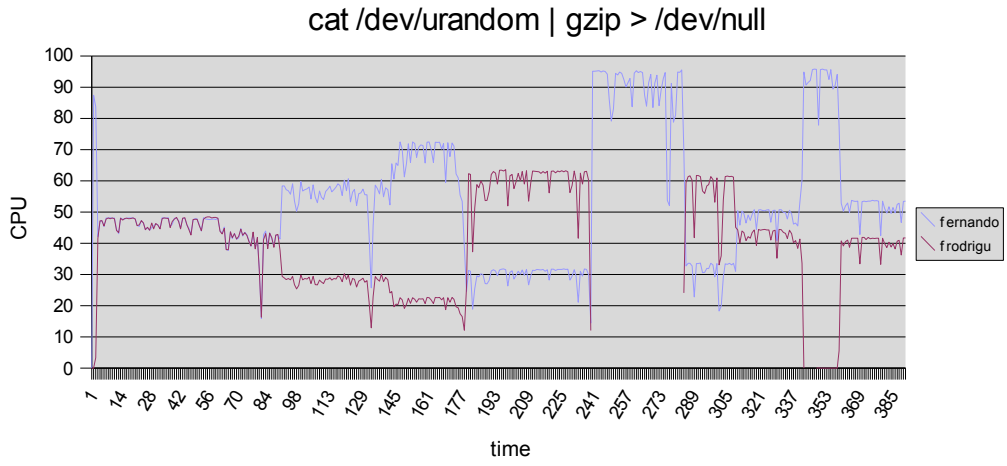
We have based our experiments in some of the main results that are published in [5], here we reproduce as close as possible a set of experiments in order to capture the main characteristics of those experiments and compare them with our results.

We measure the performance (CPU) of a tycoon auctioneer under two different scenarios.

In appendix A of this report you can see how was performed the monitoring (and log) of Xen domains performance during experiments.

4.1 Experiment one

Here we use `cat /dev/urandom | gzip > /dev/null` command to simulate jobs that makes use of intensive CPU processing. Both accounts were created with 1 credit and interval was left to default value 10000000 seconds.



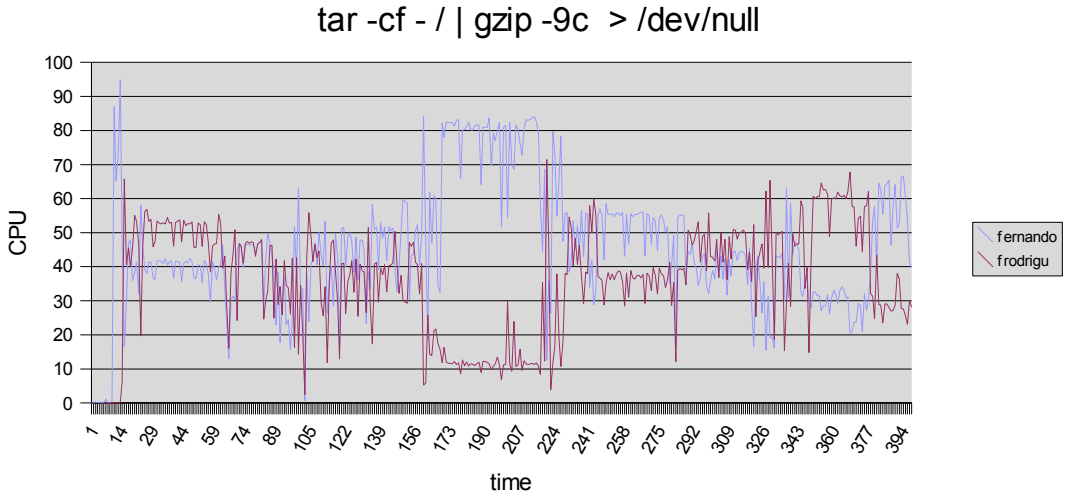
Here we can see changes in graph by analyzing next commands through time.

```
Time   Command
2      [frodrigu@frodrigu ~]$ cat /dev/urandom | gzip > /dev/null
4      [fernando@fernando ~]$ cat /dev/urandom | gzip > /dev/null
93     [usuario1@76 ~]$ tycoon host fund  62.57.49.76 1
146    [usuario1@76 ~]$ tycoon host fund  62.57.49.76 1
183    [usuario2@76 ~]$ tycoon host fund  62.57.49.76 5
242    [usuario1@76 ~]$ tycoon host bid 62.57.49.76 500
287    [usuario2@76 ~]$ tycoon host bid 62.57.49.76 500
312    [usuario2@76 ~]$ tycoon host bid 62.57.49.76 1000
344    [frodrigu@frodrigu ~]Ctr-C
363    [frodrigu@frodrigu ~]$ cat /dev/urandom | gzip > /dev/null
```

Note: Because of limited resources (just one laptop), tycoon clients “usuario1” and “usuario2” were configured in Domain-0 of auctioneer, and all of the commands were issued there causing a little extra overhead during change of contexts, this could be avoided if tycoon clients were configured in different PCs.

4.2 Experiment two

Here we use `tar -cf -/ | gzip -9c > /dev/null` command to simulate jobs that makes use of hard disk reading as well as CPU processing. Account fernando starts with 5 credits and frodrigu with 6 credits, and the intervals were set to 100000 seconds for both of them.



And now the commands that were used during this experiment.

```
Time  Command
12    [fernando@fernando ~]$ tar -cf -/ | gzip -9c > /dev/null
17    [frodrigu@frodrigu ~]$ tar -cf -/ | gzip -9c > /dev/null
72    [usuario1@76 ~]$ tycoon host fund  62.57.49.76 1
109   [usuario1@76 ~]$ tycoon host fund  62.57.49.76 1
163   [usuario2@76 ~]$ tycoon host bid 62.57.49.76 500000
233   [usuario2@76 ~]$ tycoon host bid 62.57.49.76 300000
292   [usuario2@76 ~]$ tycoon host bid 62.57.49.76 200000
347   [usuario1@76 ~]$ tycoon host bid 62.57.49.76 400000
382   [usuario1@76 ~]$ tycoon host bid 62.57.49.76 100000
```

5 Conclusions

We can see in both experiments that tycoon is fair and fast to allocate resources, this is done as soon as an account change its bid interval or add credits to spend. The baseline experiments reported by tycoon team uses a rendering application to test tycoon cluster and the measure is done with frames/s, even though we measure CPU in our simulated jobs, we see a strong correlated behavior for the main characteristics discussed in the referenced work.

Auction as a market mechanism to allocate resources and Xen for virtualization conforms the platform of tycoon which is an integral solution for bidding for resources as well as allowing testing new algorithms under this framework.

Also, tycoon is continuously improved, Kevin Lai has confirmed that tycoon will implement memory management allowing clients to bid on memory resource.

References

1. [abstract] [PDF] [bibtex] Kevin Lai, Bernardo A. Huberman and Leslie Fine, "**Tycoon: A Distributed Market-based Resource Allocation Systems**", *Technical Report* *arXiv:cs.DC/0404013*, April 5, 2004.

2. Xen Project Home Page http://www.cl.cam.ac.uk/Research/SRG/netos/xen/

3. Jeffrey Shneidman et al. *"Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems"*

4. Anthony Sulistio and Rajkumar Buyya, *"A Time Optimization Algorithm for Scheduling Bag-of-Task Applications in Auction-based Proportional Share Systems"*.

5. [abstract] [PDF] [bibtex] Kevin Lai, Lars Rasmusson, Eytan Adar, Stephen Sorkin, Li Zhang and Bernardo A. Huberman, "**Tycoon: an Implementation of a Distributed Market-Based Resource Allocation System**", *Technical Report* *arXiv:cs.DC/0412038*, December 8, 2004.

6. Kevin Lai. Tycoon User's Manual. Version 0.3.0p33. http://www.hpl.hp.com/research/tycoon/documentation/users_manual_en/index.html

7. Kevin Lai. Tycoon Administrator's Guide. Version 0.3.0p33. http://www.hpl.hp.com/research/tycoon/documentation/admin_manual_en/index.html

Appendix A

Monitoring Xen Domains.

Listing domains

A general list can be retrieve with `xm list`, this command shows general information about every domain running. Domain-0 is the first running domain and is loaded during boot process, it has root privileges over Xen like create or destroy VMs.

```
[root@76 ~]# xm list
Name                                ID Mem(MiB) VCPUs State   Time(s)
Domain-0                             0      128     1 r-----  1609.6
fernando                             1      191     1 -b----   193.2
frodrigu                             2      192     1 -b----    11.0
```

If we want a detailed information report about a specific domain, then we can retrieve it with `xm list domainname --long`.

```
[root@76 ~]# xm list frodrigu --long
(domain
    (domid 1)
    (uuid 26cc695e-af65-e46b-b354-307c9b5a6d3e)
    (ssidref 0)
```

```
    (vcpus 1)
    (cpu_weight 1.0)
    (memory 192)
    (maxmem 384)
    (name frodrigu)
    (on_poweroff destroy)
    (on_reboot restart)
    (on_crash restart)
    (image
        (linux
            (kernel /home/frodrigu/vmlinuz-default)
            (ip
                10.20.9.246:1.2.3.4:10.20.9.245:255.255.255.252 ......
```

Monitoring domains

This can be done with xenmon.py, a performance monitoring tool developed by HP and accepted to be part of Xen 3.

Lets run xenmon by typing

```
[root@76 ~]# xenmon.py
xenbaked: no process killed
ms_per_sample = 100
Initialized with 1 cpu
CPU Frequency = 1594.71
ERROR: Failure to get trace buffer pointer from Xen (22 = Invalid
argument)
xenbaked: no process killed
```

In case errors happen a text window will show you something like this

```
CPU = 0        Last 10 seconds                                  Last 1 second
================================================================================
                  0.00%                                          0.00%



 CPU = 0        Last 10 seconds                                  Last 1 second
```

You can quit xenmon with 'q'

Now lets fix this problem with the following commands

```
[root@76 ~]# setsize 20
Failure to get tbuf info from Xen. Guess size is 0: Invalid argument
This may mean that tracing is not enabled in xen.
set_size Hypercall failure: Invalid argument

[root@76 ~]# tbctl 1
Tracing now enabled

[root@76 ~]# xenmon.py
```

Now you will be able to see a text window like following showing the domains running.

```
================================================================================================================
  0   853.43 ms   85.34%      28.86 ms/ex       80.58 ms    8.06%       245.52 us/ex      Gotten
  0                           4.30 ms/ex                                2.69 ms/ex        Allocated
  0   146.13 ms   14.61%      0.00 ns/io        911.49 ms   91.15%      0.00 ns/io        Blocked
  0   439.35 us   0.04%       14.86 us/ex       7.90 ms     0.79%       24.07 us/ex       Waited
  0                           29/s                                      328               Execution count
  0   0/s                     0/ex              0                       0.00/ex           I/O Count
  1   18.46 us    0.00%       71.83 us/ex       204.44 us   0.02%       70.18 us/ex       Gotten
  1                           500.00 us/ex                              500.00 us/ex      Allocated
```

```
1   999.98 ms   100.00%     0.00 ns/io        999.73 ms   99.97%      0.00 ns/io        Blocked
1   2.07 us     0.00%       8.05 us/ex        70.65 us    0.01%       24.25 us/ex       Waited
1                           0/s                                       2                 Execution count
1   0/s                     0/ex              0                       0.00/ex           I/O Count
2   21.98 us    0.00%       85.55 us/ex       122.48 us   0.01%       126.14 us/ex      Gotten
2                           500.00 us/ex                              500.00 us/ex      Allocated
2   999.76 ms   99.98%      0.00 ns/io        999.88 ms   99.99%      0.00 ns/io        Blocked
2   219.94 us   0.02%       855.99 us/ex      1.46 us     0.00%       1.50 us/ex        Waited
2                           0/s                                       0                 Execution count
2   0/s                     0/ex              0                       0.00/ex           I/O Count
31  146.48 ms   14.65%      4.88 ms/ex        918.41 ms   91.84%      2.77 ms/ex        Gotten
31                          994.01 ms/ex                              997.08 ms/ex      Allocated
31  0.00 ns     0.00%       0.00 ns/io        0.00 ns     0.00%       0.00 ns/io        Blocked
31  9.66 ms     0.97%       321.59 us/ex      81.34 ms    8.13%       244.93 us/ex      Waited
31                          30/s                                      332               Execution count
31  0/s                     0/ex              0                       0.00/ex           I/O Count
                99.99%                                    99.93%
```

In order to record this information in a file we can use option -n, also we can set the duration of monitoring, lets say for monitoring domains during 400 seconds we will use option -t 400. See this example.

```
[root@76 ~]# xenmon.py -t 400 -n
```

which will create a set of log files in the current directory.

`log-dom0.log, log-dom1.log, log-dom2.log, log-dom31.log.`

And finally use your preferred tool to analyze and graph the different metrics that can be extracted from these files.