Decentralized Distributed Systems

Tycoon

Fernando Rodríguez Haro
June 8th, 2006
Outline

• Introduction
• Architecture
• Baseline experiments
• Deploy of Tycoon
• Experiments
• Conclusions
Tycoon is a market-based system for managing compute resources in distributed clusters like PlanetLab, the Grid, or a Utility Data Center (UDC).

- A market based distributed resource allocation system based on proportional share (PS).
  - Incorporate auctions for the economic mechanism.

Tycoon = PS + auctions

\[
PS = r \left( \frac{W_i}{\sum W_i} \right)
\]
• The resources are allocated to the users who value them most (*economically efficient* way).

• Provide a mechanism to encourage users to truthfully reveal those values.
First prototype

• Implements a PS like algorithm.
  – Allow users to differentiate the value of their jobs
  – Its resource acquisition latency is limited only by communication delays
  – It imposes no manual bidding overhead on users.

• Xen for virtualization.
  – Each providing Tycoon host runs an auctioneer process that multiplexes the local physical resources for one or more virtual hosts.
  – Xen is a virtual machine monitor for x86 that supports execution of multiple guest operating systems with unprecedented levels of performance and resource isolation.
    • http://www.cl.cam.ac.uk/Research/SRG/netos/xen/
Scalability

• Current design scales to 500 hosts and 24 simultaneous active users (or any other combination with a product of 12,000) (Based on limited tests)

• Overhead
  – Hosts
  – Network
Service Model Abstraction (SMA)

- Users use a limited budget of *credits* to bid for resources.
- The user submits a bid \((h, r, b, t)\) to the auctioneer
  - Auctioneers are independent
  - Credits are not spent at the time of the bid.
  - Auctioneers calculates \(b/t\) for each bid \(i\) and resource \(r\)
- User must utilize the resource to burn the credits.
  - User uses scp to upload the program(s).
  - User uses ssh to run a program.
  - *Continuous* bid until cancelled or user runs out of money.
Advantages of the SMA

• Auctioneers can differentiate themselves by the quality of resources offered.
• User agents have an active role.
• The SMA permits an efficient, scalable, secure and reliable core infrastructure.

• Interface
  – Users have commands to create accounts, fund accounts, change bid interval and get status of account.
  – The bid interval defaults to 10,000,000 seconds.
Architecture (SLS and Bank)

• 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.
  – Uses signed messages.
  – PKI infrastructure.
Architecture (Auctioneer)

- **Management of local resources.**
  - Virtualization

- **Collection of bids from users**
  - **Fund** protocol and **set_interval** method.

- **Allocation of resources to users according to their bids.**
  - For each user $i$, the auctioneer knows the local balance $b_i$, and the bidding interval $t_i$. The auctioneer calculates the bid as $b_i/t_i$.
  - Consider a resource with total size $R$ over some period $P$.
  - The amount of resource allocated to user $i$ over $P$, is

\[
 r_i = \frac{b_i}{t_i} \sum_{j=0}^{n-1} \frac{b_j}{t_j} \cdot R.
\]
The amount that $i$ pays per second is

$$s_i = \min \left( \frac{q_i}{r_i}, 1 \right) \frac{b_i}{t_i}.$$ 

where $q_i$ is the amount of the resource that $i$ actually consumes during $P$.

• **Advertisement of the availability of local resources.**
  – The total amount available and the total amount spent (for each resource).

$$\sum_{j=0}^{n-1} s_i.$$
Tycoon’s distributed architecture

- Hosts do independent allocation
- 3 is relatively expensive, 4 is less expensive alternative
Baseline experiments

<table>
<thead>
<tr>
<th>Processor Variety</th>
<th>CPU</th>
<th>Memory</th>
<th>Disk</th>
<th># nodes</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium III</td>
<td>1 GHz</td>
<td>2 GB</td>
<td>32 GB SCSI</td>
<td>4</td>
<td>US</td>
</tr>
<tr>
<td>Mobile Pentium III</td>
<td>900 MHz</td>
<td>512 MB</td>
<td>40 GB IDE</td>
<td>8</td>
<td>UK</td>
</tr>
<tr>
<td>Pentium III</td>
<td>550 MHz</td>
<td>256 MB</td>
<td>10 GB IDE</td>
<td>2</td>
<td>US</td>
</tr>
<tr>
<td>Pentium II</td>
<td>450 MHz</td>
<td>128 MB</td>
<td>10 GB IDE</td>
<td>6</td>
<td>UK</td>
</tr>
</tbody>
</table>

Test cluster

User increasing his share at 190 seconds by decreasing the bidding interval
Baseline experiments

A low priority job with small share getting lower throughput when a high priority job arrives.

This figure shows how a user that runs infrequently can receive more resources when he does run in comparison to a user that runs continuously.
Deploy of Tycoon
Tycoon client

• Install packages
  install tycoon_client

• Leave setup to use the standard SLS and Bank (recommended)
  – tycoon-sls.hpl.hp.com
  – tycoon-bank.hpl.hp.com

• Or install and configure our own SLS and Bank.
  – Installing and Configuring the Bank
  – Installing and Configuring the SLS

• Configure firewall(s) (Linux and/or external) to open ports
  – 25955 TCP out SLS
  – 8899 TCP out Bank
  – 24571 TCP out Auctioneer

• If necessary configure an HTTP Proxy
  – `/etc/tycoon/tycoon.conf`
  – Change HTTPProxy, "http://proxy.hpl.hp.com:8088"
Create a bank account

• 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

• Setup a Tycoon configuration
  
  [usuario@haro ~]$
  $ tycoon user setup ferharo@gmail.com ferharo

  ~/.ssh/id_dsa.pub

• Send email address and bank account public key file to Kevin
  
  ~/.tycoon/ferharo@gmail.com/bank_account_public_key

  Kevin will respond confirming the creation.

  – Note: since May 26th can be done via web.

• Verify Bank
  
  [usuario@haro ~]$
  $ tycoon bank get_balance

  Account balance: 100

• Verify SLS
  
  [usuario@haro ~]$
  $ tycoon get_host_list

  IP Address
  ------------
  204.123.32.58
  204.123.32.42
  ..... etc
• **Install packages**
  
  - `yum -y -c http://tycoon.hpl.hp.com/~tycoon/dl/yum/tycoon.repo install tycoon_aucd_xen3`

• **Update the GRUB configuration in /boot/grub/grub.conf (Warning)**

```bash
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/vmlinux-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/vmlinux-2.6.16-xen3_86.1_fc4 ro root=LABEL=/1 rhgb quiet
    module /boot/initrd-2.6.16-xen3_86.1_fc4.img
```
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
- Verify Xen kernel: `uname -a`
  - 2.6.16-xen3_86.1_fc4 #1 SMP Thu Apr 13 08:26:52

Configure SLS and Bank (same as client)
Tycoon Auctioneer

- Copy the owner's bank account key pair to `/etc/tycoon`
  - `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 an HTTP Proxy
  - `/etc/tycoon/tycoon.conf`
  - Change `HTTPProxy", "http://proxy.hpl.hp.com:8088"
- Auctioneer
  - `service tycoon_aucd [stop, start, restart, status]`
Create accounts for experiments

Tycoon account: fernando

Tycoon account: frodrigu

Tycoon account: ferharo
Using Tycoon

[usuario1@76 ~]$ tycoon get_host_list
[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

Usuario1 @ 76....
Tycoon account: fernando

[usuario2@76 ~]$ tycoon get_host_list
[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

 Usuario2 @ 76...
Tycoon account: frodrigu
Using Tycoon

Usuario1 @ 76....
Tycoon account: fernando

Usuario2 @ 76...
Tycoon account: frodrigu

```bash
[usuario1@76 ~]$ tycoon_ssh fernando@62.57.49.76
[usuario1@76 ~]$ tycoon_scp -r program 62.57.49.76
[usuario1@76 ~]$ tycoon bank get_history

[usuario2@76 ~]$ ssh -l frodrigu -p 15070 62.57.49.76
[usuario2@76 ~]$ scp -P 15070 program frodrigu@62.57.49.76:program
[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}
```
Using Tycoon

Usuário1 @ 76....
Tycoon account: fernando

[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 Glide3-1.....
Complete!

Usuário2 @ 76...
Tycoon account: frodrigu

[usuario2@76 ~]$ tycoon host shutdown 62.57.54.178
Shutdown virtual machine on 62.57.54.178. Graceful: True
[usuario2@76 ~]$ tycoon host boot 62.57.54.178
62.57.54.178 has booted.
[usuario2@76 ~]$ tycoon host delete_account 62.57.54.178
Deleting account(s) (may take a minute)...
62.57.54.178 deleted account.
Experiments (CPU)

cat /dev/urandom | gzip > /dev/null

[frodrigu@frodrigu ~]$ cat /dev/urandom | gzip > /dev/null
[fernando@fernando ~]$ cat /dev/urandom | gzip > /dev/null
[usuario1@76 ~]$ tycoon host fund 62.57.49.76 1
[usuario1@76 ~]$ tycoon host fund 62.57.49.76 1
[usuario2@76 ~]$ tycoon host fund 62.57.49.76 5
[usuario1@76 ~]$ tycoon host bid 62.57.49.76 500
[usuario2@76 ~]$ tycoon host bid 62.57.49.76 500
[usuario2@76 ~]$ tycoon host bid 62.57.49.76 1000
[frodrigu@frodrigu ~]Ctr-C
[frodrigu@frodrigu ~]$ cat /dev/urandom | gzip > /dev/null
Experiments (CPU and I/O)

Initial bids: Fernando 5, frodrigu 6

[usuario1@76 ~]$ tycoon host bid 62.57.49.76 100000
[usuario1@76 ~]$ tycoon host bid 62.57.49.76 100000
[usuario1@76 ~]$ tycoon host fund 62.57.49.76 1
[usuario1@76 ~]$ tycoon host fund 62.57.49.76 1
[usuario2@76 ~]$ tycoon host bid 62.57.49.76 500000
[usuario2@76 ~]$ tycoon host bid 62.57.49.76 300000
[usuario2@76 ~]$ tycoon host bid 62.57.49.76 200000
[usuario1@76 ~]$ tycoon host bid 62.57.49.76 400000
[usuario1@76 ~]$ tycoon host bid 62.57.49.76 100000

tar -cf - | gzip -9c > /dev/null
[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

[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

......
Xen: XenMon

[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

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

[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
### Xen: XenMon

<table>
<thead>
<tr>
<th>Domain</th>
<th>Time (ms)</th>
<th>%</th>
<th>Time (ns/io)</th>
<th>%</th>
<th>Time (us/ex)</th>
<th>%</th>
<th>Time (ms)</th>
<th>%</th>
<th>Time (ns)</th>
<th>%</th>
<th>Execution count</th>
<th>I/O Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>853.43</td>
<td>85.34%</td>
<td>28.86</td>
<td>80.58%</td>
<td>245.52</td>
<td>Gotten</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4.30</td>
<td>0.43%</td>
<td>0.00</td>
<td>0.00%</td>
<td>0.00</td>
<td>Allocated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>146.13</td>
<td>14.61%</td>
<td>14.86</td>
<td>7.90%</td>
<td>24.07</td>
<td>Waited</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>29</td>
<td>0.29%</td>
<td>29/s</td>
<td>328</td>
<td>0.00/ex</td>
<td>Execution count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0/s</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0/0</td>
<td>I/O Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.46</td>
<td>0.00%</td>
<td>500.00</td>
<td>500.00</td>
<td>126.14</td>
<td>Gotten</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>999.98</td>
<td>100.00%</td>
<td>999.73</td>
<td>999.97%</td>
<td>1.50</td>
<td>Waiting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.07</td>
<td>0.02%</td>
<td>2/s</td>
<td>2</td>
<td>0.00/ex</td>
<td>Execution count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21.98</td>
<td>0.00%</td>
<td>85.55</td>
<td>122.48</td>
<td>0.00/ex</td>
<td>I/O Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>999.76</td>
<td>99.98%</td>
<td>999.88</td>
<td>99.99%</td>
<td>0.00</td>
<td>Allocated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>219.94</td>
<td>0.02%</td>
<td>855.99</td>
<td>1.46%</td>
<td>0.00</td>
<td>Blocked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0/s</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0/0</td>
<td>Waiting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>146.48</td>
<td>14.65%</td>
<td>4.88</td>
<td>181.48</td>
<td>0.00/ex</td>
<td>Gotten</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>994.01</td>
<td>99.98%</td>
<td>997.08</td>
<td>99.98%</td>
<td>0.00/ex</td>
<td>Allocated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0.00</td>
<td>0.00%</td>
<td>0.00</td>
<td>0.00%</td>
<td>0.00</td>
<td>Blocked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>9.66</td>
<td>0.97%</td>
<td>321.59</td>
<td>81.34%</td>
<td>0.00</td>
<td>Waiting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>0.00%</td>
<td>30/s</td>
<td>332</td>
<td>0.00/ex</td>
<td>Execution count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0/s</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0/0</td>
<td>I/O Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

log files in the current directory:

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

- Resource allocation
- Continuous Bids
- Virtualization
- Users
- Experiments
Tycoon team

Kevin Lai (Architecture and development)
Lars Rasmusson (Development and virtualization)
Thomas Sandholm (Development, Grid Integration, Predictability)

• Developers
  – Browse source (www)
  – Getting source via *Mercurial*: a fast, lightweight Source Control Management system designed for efficient handling of very large distributed projects.
  – Programming language: Python.

• Tycoon support
  – Users mailing list
    • [http://groups.google.com/group/Tycoon-Users](http://groups.google.com/group/Tycoon-Users)
More references

- Tyler Close, “Credit transfer within market-based resource allocation infrastructure”, HPL-2006-5 report

- Chee Shin Yeo and Rajkumar Buyya, “A taxonomy of market-based resource management systems for utility-driven cluster computing”


- Jeffrey Shneidman, Chaki Ng, et.al. “Why Markets Could (But Don’t Currently) Solve Resource Allocation Problems in Systems”
