Linux Cluster Architecture

by

Alex Vrenios

(Shameless Plug)

English

Chinese

Slide # 1
Copyright © 2003 Alexander Vrenios
Overview:

- Why would anyone want to build a cluster system?
- Computer Architecture Review: UPs through Clusters
- Gathering the PC computer hardware (on the cheap!)
- Connecting the node computers into a local area network
- Local and remote software development file structure
- Configuring relevant Linux OS files for internetworking
- Subtasks and sockets for local or network programming
- The design of our master-slave cluster server software
- Internal and external performance monitoring and tuning
Why would anyone want to build a cluster system?

- **Hobbyists:**
  It’s a new and interesting pathway to experience! (And how many of your friends have a cluster server anyway?)

- **Professionals:**
  Sophisticated systems are often developed in parallel, meaning the hardware won’t be ready when you want to test your software. Having a test bed will get you past the hardware independent bugs, and put you in a position to polish your product when the platform is finally ready.

- **Managers:**
  This is all bleeding edge stuff; you’ll want to prepare for the issues your people might face and the questions they might ask. Experience gives you the insight you will need.

- **Academics:**
  Analyze data from a live system, when you can, simulation can be over-simplified and its results can be misleading.
Computer Architecture Review:

- Uniprocessor or UP

*SISD*: Single instruction, single data stream.

The typical PC is a uniprocessor.

* Flynn proposed this taxonomy - some other configurations follow...
Linux Cluster Architecture

- **Array or Vector Processor**

  SIMD: Single Instruction, multiple data stream.

  (ILLIAC IV, IBM 390, DSPs, etc.)

- **Pipeline Processor**

  MISD: Multiple instruction, single data stream?

  (Some say there is no MISD.)

- **Multiprocessor or MP**

  MIMD: Multiple instruction, multiple data streams.

  Mainframe, Workstation, etc. (Mostly for the very wealthy!)
The MIMD is so interesting that gets its own taxonomy:

- **UMA**: Uniform Memory Access
  - Tightly-coupled multiprocessor. All CPUs access instructions and data at the same transfer rate.

- **NUMA**: Non-uniform memory access
  - VME Chassis, some Beowulf Clusters, and many embedded processor systems.
  - Plug-in boards, for example, each with a CPU and some local memory.

- **NORMA**: No (hardware) Remote Memory Access (rare distinction)
  - Older Beowulf Clusters, Distributed Shared Memory Systems (IVY), and some Modern day cluster computers.
  - PCs, whose “personality” may be molded by its software!
Network Block Diagram:

Research testbed:

- External Monitor
  - Desktop PC (Near-real-Time Performance)
- Interactive Client
  - Query Generator & Response Monitor
- Cluster Server
  - Node 0
  - Node 1
  - Node 2
  - Node 3
  - Node 4
  - Node 5
  - Node 6
  - Node 7
- NFS
- Client queries & responses

Ethernet
Linux Cluster Architecture

CHAOS: Cheap Array of Outmoded Systems:

• Finding and networking $N$ personal computers:

  - 4-way KV Switches (may be optional)
  - Network Activity Monitoring
  - Surge Protectors
  - Development System
  - Books!
  - and NFS File Server
Gathering PC Computer Hardware:

- Small computer stores (Renaissance Computer, e.g.)
- Newspaper and club and organization newsletter ads
- Family, friends and neighbors (closets, garage sales)
- Large corporations? (hospitals, Am Exp, Mot, etc.)
- Computer salvage outlets:

  ![Diagram]

  - ASU Salvage
  - University
  - Pima 101
  - Rio Salado
Connecting the Node PCs into a LAN:

- **IP address** (unique for each node)
- **Host name** (unique for each node)
- **MAC address** (not shown)
- **Domain name**
One way to organize your files for software development:

1. You create a make file under src

   ```
   pgm: 
   gcc -I../inc/ -o../bin/pgm pgm.c
   ```

2. You issue a make command under src

   ```
   alpha:/home/alex/src> make pgm
   ```

3. Make reads in the data

4. Make creates an executable under bin
Remote User File Structure:

Network File System (NFS): the *illusion of locality* via remote-mount points
Linux OS Networking Files:

- First, file /etc/hosts belongs on all the network nodes:
  127.0.0.1  localhost  localhost.chaos.org
  10.0.0.1   alpha     alpha.chaos.org
  ...
  10.0.0.5   omega     omega.chaos.org

- Next, file /etc/exports on 10.0.0.1, the NFS server named alpha:
  Server ———> /home (rw)

- Finally, file /etc/fstab on every node except the server named alpha:
  Clients ———> 10.0.0.1:/home /home nfs rw 0 0
  /dev/hda1 swap swap defaults 0 0
  /dev/hda2 / ext2 defaults 1 1
  none /proc proc defaults 0 0
  /dev/fd0 /mnt/floppy ext2 noauto 0 0
Internetworking Services: Operation

The *illusion of locality* via internetworking services

1. You issue `rcat`
2. `rcat` sends udp packet to service
3. `rcatd` gets control and uses a `cat` command
4. `rcat` lists the file contents on screen
Internetworking Services: Configuration (using inetd)

- Add a line to file `/etc/services` on each remote-server node:
  
  ```
  rcatd  5000/udp  # remote-cat UDP service on port 5000
  ```

- Add a line to file `/etc/inetd.conf` on each remote-server node:
  
  ```
  rcatd  dgram  udp  wait  alex  /home/alex/bin/rcatd
  ```

- Reconfiguration: `omega:/root> killall -HUP inetd`

Sequence of events:

1. Client process sends a UDP packet to server’s port 5000
2. Daemon (inetd) starts process at `/home/alex/bin/rcatd`
3. Service reads incoming UDP packet data from “keyboard”
Internetworking Services: Configuration (using xinetd)

- File `/etc/xinetd.d/rcatd` on each (xinetd) remote-server node:

  ```
  service rcatd {
    port = 5000
    socket_type = dgram
    protocol = udp
    wait = yes
    user = alex
    server = /home/alex/bin/rcatd
    only_from = 10.0.0.0
    disable = no
  }
  ```

- Reconfiguration: `omega:/root> /etc/rc.d/init.d/xinetd restart`

(Refers to name of service)

(Means 10.0.0.*)
Distributed Systems C-Language Skills:

Subtasking on a single processor
(details are in the book)

• Shared memory and semaphores
• Signal header/structure
• Handler initialization
• Signal processing and handler re-initialization
Linux Cluster Architecture

Distributed Systems C-Language Skills:

- Internetworking headers/structures
- Socket initialization
- Remote service “discovery”
- Reliable communications
  (topic discussions, anyway)
Master-Slave Cluster Server - Overview:

- **Slave**
  - **Beta**
  - **Gamma**
- **Master**
  - **Alpha**
  - **Delta**
- **Smart Client**
- **Monitor**

Queries and responses are handled between the smart client and the monitors. Performance and statistics are monitored and managed by the monitors.
Master-Slave Cluster Server - Initialization:

Broadcast starts *slave* tasks...

Local subtasks contact slaves...

- Master starts local subtask, one for each registering remote slave.
- All tasks start a *perform* subtask.
Master-Slave Cluster Server - Operation:

Perform tasks send performance info to monitor…

queries are processed by first available slave, via s# subtask.
### Linux Cluster Architecture

#### Performance Statistics – Pseudo-file Text*

- **CPU Utilization in /proc/stat** - Running Jiffy Counts in each State
  
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu</td>
<td>1256</td>
<td>0</td>
<td>1566</td>
</tr>
</tbody>
</table>

- **Disk Reads and Writes in /proc/stat** - Running I/O Counts
  
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>disk_rio</td>
<td>1270</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>disk_wio</td>
<td>1337</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Note that the exact meaning of proc file content can be OS release dependent: see `man proc`*
Performance Statistics - More Pseudo-file Text:

- Memory Utilization in /proc/meminfo - Current Values
  
  Mem:  14942208  13713408  1228800  ...

- Packets Sent and Received in /proc/net/dev - Running I/O Counts
  
  lo:  80  0  0  0  0  80  0  0  0  0
  eth0:  115  0  0  0  0  68  0  0  0  0
Internal Performance Statistics - Display:

NEAR REAL-TIME CLUSTER PERFORMANCE STATISTICS

10Base2

-----ALPHA-----+           |           +-----BETA-----+
| Cpu  Mem     | | Cpu  Mem    |
|     7%  94%  |Rcvd 0     |  21 Rcvd |  28%  40% |
| Rio  Wio +-----------+-----------+ Rio  Wio |
|     1      0 |Sent 12 |  1 Sent 01 |     0      1 |
|10.0.0.1-----+           |           +10.0.0.2-----+

-----GAMMA-----+           |           +-----DELTA-----+
| Cpu  Mem     | | Cpu  Mem    |
|     2%  75%  |Rcvd 2     |       0 Rcvd|  5%  56% |
| Rio  Wio +-----------+-----------+ Rio  Wio |
|     4      0 |Sent 0  |  10 Sent 3 0 |     0      1 |
|10.0.0.3-----+           |           +10.0.0.4-----+

chaos.org

- Overall Network Loading -
  23 Pkts/sec
Performance Monitoring – External (displayed):

- Resource utilization reporting via a smart client process:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>OBSERVATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME (msec)</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>70</td>
</tr>
<tr>
<td>71</td>
<td>80</td>
</tr>
<tr>
<td>81</td>
<td>90</td>
</tr>
<tr>
<td>91</td>
<td>100</td>
</tr>
</tbody>
</table>

50 Total Observations

Average = 30 milliseconds ...what if you’re not happy with this level of performance?
Performance Tuning – Defining Execution Phases:

1. Status Table
2. S1
3. Transit times
4. Slave 1
5. Shared
6. DB
7. Transit times
8. S2
9. Response

Client

Ethernet

DB

Slave 2

SLAVE

MASTER

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE

SLAVE
Performance Tuning - Execution Phase Times:

![Graph showing initial MSI phase times with three time distributions: Expon, Pulse, and Sweep.]

Not a SW issue.

Leave the file open?

Found a bug!
Performance Tuning - Final Times:

Average Time (seconds)

- Expon
- Pulse
- Sweep

Dramatic reduction!

About a 10% improvement (not too bad)

See book for further details on statistical distributions.

Final MSI Phase Times

Execution Phases (Three Time Distributions)
Performance Tuning:

• The proof is in the pudding!

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>OBSERVATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME (msec)</td>
<td>10</td>
</tr>
</tbody>
</table>

---

Average = 25 milliseconds = 17% improvement!
Further Details are in the Book:

- Download all the C source code for free:
  
  http://www.samspublishing.com

  - Search on "Linux Cluster Architecture" or "Vrenios"
  - Click on the “Downloads” link in the book description

  1. Individual chapter examples are in zip files
  2. A complete user-files environment is in a tar.gz file

- Book Signings:

  Sep 8th          Borders Chandler, Sunday @ 2pm
  Sep 15th         Borders Arrowhead, Sunday @ 2pm
  Oct 25th         Barnes & Noble Arrowhead, Friday @ 7pm
Further Reading:

* Distributed Operating Systems, Andrew S. Tanenbaum (of MINIX and AMOEBA fame!), Prentice Hall, 1995

* Unix Distributed Programming, Chris Brown, Prentice Hall, 1994

* Advanced Programming in the UNIX Environment, W. Richard Stevens, Addison-Wesley, 1992

* Linux Programming White Papers, Rushling, et al, CoriolisOpen, 1999

Further details about the early development of my cluster are in:


* “CHAOS Part 2,” LinuxGazette.com, Alex Vrenios, December 1998
Linux Cluster Architecture

You’ve been a terrific audience!

Any questions?

Hurry out and buy this book!

Or this one…