ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

# Approach to Manage Resources in Linux

Prashu Kumar Jain

***Abstract:- The paper reviews of research are an effort to develop a resource management framework for Linux. The framework consists of a basic infrastructure that allows resources such as CPU, Memory, Disk I/O to be controlled and monitored. It investigates the development of resource management, the alternative solutions proposed and the challenges faced for developing such a framework The paper describes the most essential concepts, such as an infrastructure, controllers, feedback and monitoring of resources. Each resource controller has its own set of problems to solve. Since the work is being done for a large community of Linux users, the addition of a new feature such as this, should not impact existing users or users who are not interested in using resource management. The paper looks at two commonly used resources, CPU and memory. The resource control and monitoring mechanisms developed for them is described in great detail in two separate chapters devoted to them***

***Keywords:*** **Network Intrusion Detection, Data Mining, Honeypots, User Authentication.**

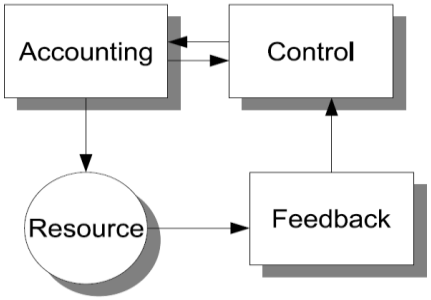
## I. INTRODUCTION

As Virtualization is a key need for large enterprises. It allows enterprises to partition their workloads without adding the overhead of additional hardware, space and system administration. Each virtual system has a unique role to play, for example, an enterprise might configure its system into two virtual systems. It might use one of the virtual systems for use in production and the other for use in a test environment. The test system might be in use by developers working on the next release of the product. The production system needs to use all available resources and the test system should receive either a small percentage of available resources or unused resources. To achieve this goal, resource management is used. In the example above, the system administrator might configure the system to provide 80% of the CPU to the production system and 20% to the test system. Resources are usually administrated by grouping tasks and associating them with resources. For example all tasks in a system belonging to a particular customer could be grouped together and then they may be assigned a certain percentage (say 30%), based on the contract with the customer. Accounting is needed to monitor report usage. Control consists of providing guarantees and limits on resource usage. A guarantee ensures that a certain percentage of a resource is always available to the group of tasks. A limit ensures that the usage (monitored by accounting the resource usage) does not exceed a certain amount. A point us towards the need for sharing resources yet achieving isolation. A group of tasks are isolated as a container. Each container has a set of resources associated with it, examples include CPU, Memory and Disk I/O. Each of these resources is shared across the operating system, but in an unfair manner. A certain container might be business critical and thus requires more resources than another lower priority container. The isolation is achieved by setting limits and guarantees for each resource. Traditional UNIXes provide limited resource control. They provide a per task POSIX interface called rlimit (2). rlimit however does not meet the needs of resource management for the following reasons —

1. It is per task based, most resource management Requirements need to control a group of unrelated Tasks.
2. In the case of some of the resources (Resident Set Size (RSS)), rlimit is not enforced thus; there is a Need for developing a new resource management Framework and resource controllers to meet the ever increasing need for resource management in the Enterprise.

## II. RESOURCE MANAGEMENT ARCHITECTURE (EXISTING)

In this section, we discuss an architecture that is applicable to most resource management solutions that have been proposed so ar. In figure, the resource manager is shown in modular form as components. The components include an accounting subsystem, a control subsystem and a feedback subsystem. The resource in the diagram represents a managed resource.




**1 Accounting**

The accounting subsystem tracks the resource usage of each container. In the case of a CPU controller (we shall define a controller later in section 4.1), the accounting subsystem would track the cpu usage of each task and then track the usage of the group of tasks (container). For renewable resources like CPU time, it is required to define an interval over which accounting takes place. In other words, the accounting must be restarted at the end of the interval.

**2 Controls**

The control subsystem implements resource control for the group of tasks. If the system administrator limited the CPU usage of a group to say 10% of the total CPU bandwidth available, the controller would jump into action and prevent the group of tasks from using more than 10% of the bandwidth.

**3 Feedback**

Feedback tells us whether the group of tasks is making forward progress as per their resource requirements and allocations. A system administrator might initially assign 10% of the CPU to a critical business application. A system administrator might then need feedback from the system indicating how the tasks (applications) are fairing with their current resources. If the feedback indicates that the tasks end

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

up waiting for CPU most of the time, the administrator knows that it is time to boost the CPU bandwidth of the tasks. The form in which feedback is provided depends on the resource being controlled. In the case of CPU, the feedback would include the following parameters

• Time on Run queue
• Time waiting on the Run queue

## III. PROBLEMS IN EXISTING SYSTEM

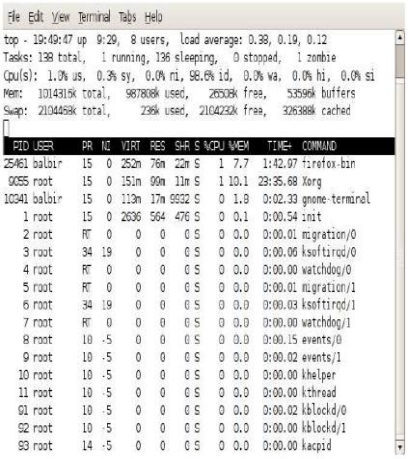
Good time on the run queue and minimal waiting time on the run queue are good indications of a CPU hungry task making forward progress.

1 Resource Management should be supported for a group of tasks
2. A task should be able to move across resource groups
3. Setting of resource limits for a group of tasks should be supported

## IV. SYSTEM FRAMWORK (PROPOSED WORK)

Feedback is a very important component of resource management. Feedback can help the administrator or a system monitor decide if the resources allocated to a certain group are sufficient. Feedback can be obtained from the operating system in several forms. The most important forms are covered and a new form of feedback called delay accounting developed for resource management is described. Delay accounting has been written by the author and accepted into the mainline Linux kernel.

**1 Top:** A common utility used for tracking and monitoring resource usage is top (1). Top is an interactive utility that displays tasks and information about the %CPU, %MEM of each task. As shown in figure, the utility displays CPU idle time, available memory, swap number of tasks running, etc. It can also sort the output by any of the displayed fields. Although top provides good feedback, it is an interactive program and requires somebody to monitor the system, understand the output and then apply the feedback. One way to automate the feedback is to get the information programmatically from the same place that top gets its information. Linux provides a /proc file system, where all process specific information is made available. top also gets its information from /proc




**2 Getrusage**

Getrusage (2) is a system call that provides the information about the following parameters of a task Integral value implies that the final value is a product of the execution time of the task and the value represented. In the case of ru ixrss for example, the total shared memory size is multiplied by the execution time of the task.

## V. IMPLEMENTATION ALTERNATIVES

There are several alternatives for implementing the infrastructure, they implement various interface mechanisms discussed earlier

**1 Bean counters**

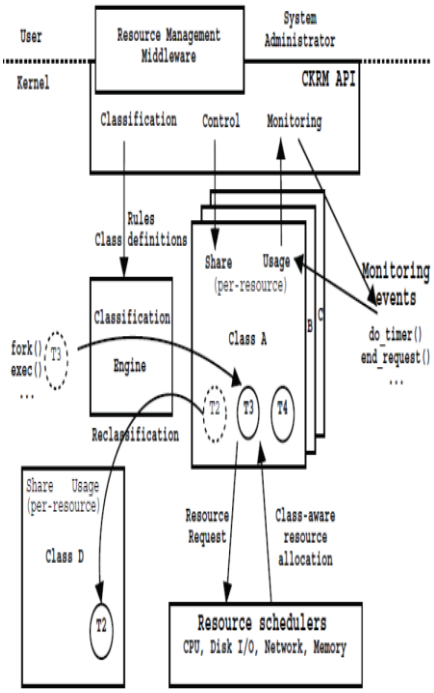
Bean counters were developed and implemented by Alan Cox [10]. They were later enhanced by OpenVZ developers [15]. Bean counters implement a small counter that is referenced counted. Figure shows what a beancounter looks like. The bean counter contains accounting information for resource (how much of it used, etc) and limit information, which dictates the hard and soft limits for the resource. Figure shows the architecture of the bean Counter implementation. The implementation uses a global hash table of all beancounters. Each beancounter has a unique identifier (a number). The number is input to the hashing function which determines where the bean counter will be placed. Each task on the system has two members fork bc and exec bc which point to the bean counter to which the task belongs. Fork BC points to the parent's bean counter, where as exec bc points to the current bean counter which is being charged for the resources that the task consumes. During fork, both fork bc and exec bc are inherited from the parent. Bean counters provide system calls to –

1Set/Change limits of a bean counter
2. Create a new bean counter
3. Get information from the bean counter about its resource usage

**2 Aggregated Beancounters**

The limitation of bean counters is that does not allow movement of tasks. To address this limitation Balbir [11] developed aggregated bean counters. Figure shows the architecture of aggregated bean counters. Each aggregated bean counter as the name suggests is an aggregation of other bean counters. Under this scheme, each task is now associated with one bean counter. A group of bean counters form an aggregated bean counter. Each aggregated bean counter like the bean counter has a set of limits and resource usage statistics for the aggregation. In the figure, task T1 is associated with bean counter B1 and aggregated bean counter A1. The figure shows a task T5 of the task group moving. It moves from aggregated bean counter A1 to A2 (the movement is shown by dotted lines). When the task moves, its associated bean counter B5 also moves to A2. Resource control and monitoring is now applied to the aggregated bean counter, instead of the individual bean counter.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012




### 3 Class based Kernel Resource Management

CKRM (Class based Kernel Resource Management) [8] was proposed to the Linux Kernel Community in 2003. CKRM provides an infrastructure very similar to the AIX 5L Workload Manager [14]. CKRM consists of an infrastructure to write controllers and group tasks. CPU, Memory and Disk I/O controllers were developed for CKRM. Figure shows the CKRM architecture. CKRM supports file system based hierarchical configuration. Control and monitoring of resource parameters. CKRM also provides an additional component called the Classification Engine. As stated previously, the container and resource hierarchy is inherited by the child on fork. The classification engine classifies the task to a class on fork or exec based on a pre-defined set of rules set by the system administrator. The system administrator could for example, classify all cron jobs to be automatically moved a particular class.

## VI. CONCLUSION

Resource Control is going to be a critical requirement in enterprises. Several operating systems have already started supporting this feature, with a good set of tools for management of the features supported. With the growth and spread of virtualization, system administrators will want to differentiate different instances based on the priority of hosting the container. In addition to that, not all applications are treated equal. Some applications or part of applications will be prioritized based on their criticality to the business, The implementation of resource management in Linux has been discussed extensively. Some parts of the code (such as delay accounting) are already available. The other parts are being developed. The wide choice of implementations of the resource control infrastructure has made it possible to discuss the selection of the most promising infrastructure. As the infrastructure stabilizes, controllers such as CPU and Memory controllers are being developed on top of the infrastructure. In this p, we looked at the scope of work and split it into phases

- Infrastructure
- Accounting
- Feedback
- Control

Each of the phases and the work done has been looked into in detail. Tests were run and data collected. The data collected shows some interesting results of the impact of the container limit on the performance of applications. The results show that limiting the container to close to 50% of its full memory requirement yields best results.

## REFERENCES

[1] Daniel, P. Bovet and Cesati Marco. Understanding the Linux Kernel. O' Reilly, November, 2005.

[2] Love, Robert. Linux Kernel Development. Novell Press, January, 2005.

[3] Vaddagiri, Srivatsa. " [RFC] Resource Management - Infrastructure choices". http://lkml.org/lkml/2006/10/30/49. October (2006).

[4] Singh, Balbir and Emelianov, Pavel. "Containers/Guarantees for resources". http://wiki.openvz.org/Containers/Guarantees for resources. November (2006).

[5] Singh, Balbir and Nagar, Shailabh. "Delay accounting patches". http://lwnnet/Articles/182133/ May (2006).

[6] Nagar, Shailabh. "Delay accounting performance". http://lkml.org/lkml/2006/3/23/141 March (2006).

[7] Moore, Paul and Hadi, Jamal. "Genetlink documentation". http://lwn.net/Articles/208755/ November (2006).

[8] Seetharaman, Chandra. "Class Based Resource Management".http://ckrm.sourceforge.net/ December (2006).

[9] Menage, Paul. "Generic Process Containers (V6)". http://lkml.org/lkml/2006/12/22/112 December (2006).

[10] Linux Weekly News. "Resource Beancounters". http://lwn.net/Articles/197433/ August (2006).

[11] Singh, Balbir. "Aggregated Beancounters". http://lwn.net/Articles/199938/ September (2006).

[12] Derr, Simon. "CPUSets Documentation". http://lwn.net/Articles/127936/ September(2006).6 About Author's

[13] Nagar, Shailabh Franke, Hubertus Choi, Jonghyuk Seetharaman, Chandra Kaplan, Scott Singhvi, Nivedita Kashyap, Vivek, Kravetz, Mike. "Class-based Prioritized Resource Control in Linux". Ottawa Linux Symposium Proceedings 1(2003).

[14] Castro, Sofia, Tezulas, Nurcan, Yu, BooSeon, Berg, Jrgen, Kim, HoHyeon, Gfroerer, Diana. AIX 5L Workload Manager. IBM Corporation (Redbook), 2001.

[15] Derr, Simon"Server Virtualization Open Source Project". http://lwn.net/Articles/127936/ September (2006).

[16] Riel, Rik Van. "Towards an O (1) VM". Ottawa Linux Symposium Proceedings 1(2003).

[17] Knuth, Donald. E. The Art of Computer Programming, Volume 1, Fundamental Algorithms Third Massachusetts: Addison-Wesley, 1997.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

[18] Corbet, John. "Trees I: Radix Trees". http://lwn.net/Articles/127936/ September (2006).

[19] Zijlstra, Peter. "RSS accounting". http://lkml.org/lkml/2006/10/10/130.

[20] Singh, Balbir and Menage, Paul. "Simple CPU accounting container subsystem". http://lkml.org/lkml/2007/2/12/90.

[21] Srinivasan, Vaidyanathan. "Containers: Page Cache Accounting and Control subsystem (v1)". http://lwn.net/Articles/224815/.

## **Author** Profile

Prashu Kumar Jain M.Tech from Gyan Vihar University Jaipur prashurock007@gmail.com and Phone No - +91 9887458060