ADVANCED OPERATING SYSTEMS

TERM PROJECT WEEKLY REPORT

Under,

An-I Andy Wang

Shuanglong Zhang

By,

Akshara Dendi ([ad13r@my.fsu.edu](mailto:ad13r@my.fsu.edu))

Lehari Saggam ([ls13d@my.fsu.edu](mailto:ls13d@my.fsu.edu))

Sinduri Shyamala ([ss13w@my.fsu.edu](mailto:ss13w@my.fsu.edu))

**OPTION CHOSEN:** SURVEY PROJECT

**PROJECT TITLE:** A STUDY ON LINUX LOCK SYSTEMS.

**GOAL:** To do a detailed analysis on the Linux lock systems.

**TEAM MEMBERS:** Akshara Dendi ([ad13r@my.fsu.edu](mailto:ad13r@my.fsu.edu))

Lehari Saggam ([ls13d@my.fsu.edu](mailto:ls13d@my.fsu.edu))

Sinduri Shyamala ([ss13w@my.fsu.edu](mailto:ss13w@my.fsu.edu))

**DATE:** 3/11/2014

**SINDURI SHYAMALA’S CONTRIBUTION:**

**Distributed Lock Manager**

This week, my focus has been on studying lock manager. For all software distributed applications, a lock manager is provided in order to use all the resources in a proper order. When two or more processes compete for resources, a deadlock may occur when they try to share them at the same time. Hence the resources are shared in particular time slots and this can be done using the Distributed Lock Manager. In clustered file systems, the DLM has been playing a very major role in providing better performance. A clustered file system is where different processes belonging to one cluster can access the storage of any file in the same cluster. For LINUX, RedHat and Oracle have introduced this clustering. DLM is very much used in Virtual Machine Systems (VMS). This was used in Version 3. Various resources have been associated with the DLM. The locks in a particular database can be accessed completely and also individual parts based upon the requirement can be accessed by any process.

**Lock Modes**

The Virtual Machine System Cluster generally referred to as the VMSCluster has different lock modes. This classification of the different modes is based on the need to access a resource at a given point of time. A lock can be moved to higher or lower levels. The data of a resource can be destroyed when the resource is unlocked by the different processes.

* NL-Null Lock

In this mode of locks, the resource and its lock value block are saved. In situations where processes do not have to lock a resource, the values are still saved.

* PR-Protected Read

In this mode, the resource can be read by a process but changes cannot be made to the resource. No information can be added to it. This allows other processes to read the data and hence PR is a share lock.

* EX-Exclusive

Only one process can read and modify the resource at a given time. Other processes are denied access when a process is accessing it.

All the above-mentioned locks are used in Open DLM Standard and Implementation-OCFS2DLM. The other modes are:

* CR-Concurrent Read

One process is given access to one resource at a time. Only read is possible. Other processes do not have exclusive access.

* CW-Concurrent Write

Reading and updating is possible and other processes are denied exclusive access to the resource.

* PW-Protected Write

This mode helps in reading and updating a particular resource while not allowing other processes to read it.

Enqueueing is used to access to obtain a lock. Synchronous or asynchronous operation is possible. Deadlocks also occur in locking. This happens when two or more processes get lock on the same resource. OpenVMS DLM is used to check for deadlocks.

**LEHARI SAGGAM’S CONTRIBUTION:**

The topic on which I have been working last week is “futex”. Futex mechanism was introduced in 2.5.7 by Rusty Russell, Hubertus Franke, and Mathew Kirkwood. It is a fast and a lightweight kernel-assisted locking primitive built for user-space applications. Futex provides very fast uncontended lock acquisition and release. The futex state is saved in a user-space variable. Futex is the root of mutual exclusion constructs which are commonly used in threaded programming some of them are pthread mutexes, condvars, semaphores, rwlocks, and barriers. All this have been through reconstructive and cosmetic surgery over the last several years, and are now more effectively and more functional.

Futexes can be utilized store the state of a lock and give kernel waitqueue for tasks which are blocking on the lock. To lessen the syscall overhead, this state has to allow for atomic lock acquisition when the lock is uncontended. The state could be defined as unlocked and locked.

I have also read about all the kernel locks like Ticket lock, Giant lock. I can describe about all these locks once I complete all the papers on this.

We will be starting our actual project from this week. We are planning to start the comparison of the code of various versions of Linux locks. We will be simultaneously try to gather the information about them and will also work on our documentation by writing all the required information necessary to be added in the documentation.

**AKSHARA DENDI’S CONTRIBUTION:**

This week my study included kernel lock manager, locking in user context and locking between user softirqs, tasklets and timers.

Kernel lock manager is used to locate the locks and also assist the users. It has many advantages over current cache functionalities like extendible lock, built-in, cross nodes etc. we can also edit and configure the parameters using lock manager. Using it we can view list of lock screens, single lock details, add and manage locks, terminate the process etc. Using edit lock we can add and export the lock templates.

We use simple semaphore for in user context locking and it will not return the received signal if down() is used so we should mostly avoid it. There are few issues when softriqs is used with user context locking and work good for Uni-processor. When using locking between user and tasklets it runs on single CPU at once and using timers its similar to tasklets. When tasklet and timer want to share data with each other then we need consider two cases: same and different. Even when softriqs wants to share data with itself then also we need to considered two cases same and different softriqs.

References:

[1] <http://www.va.gov/vdl/documents/Infrastructure/Kernel/xu_8_0_sp_p608.pdf>

[2] <https://www.kernel.org/pub/linux/kernel/people/rusty/kernel-locking/>