L4FIFO: Task Communication for DROPS

Cheng Guanghui, Nicholas MC Guire, Li Lian, Zhou Qingguo
Distributed and Embedded System Lab
School of Information Science and Engineering
Lanzhou University
Tianshui South Road 222,Lanzhou,P.R.China
cheng.guanghui@gmail.com

Abstract

The Dresden Real-Time Operating Systems Project (DROPS) is a research project aiming at the support of applications with Quality of Service requirements[1]. In the DROPS project a key component is L4Linux, the Linux server on top of the L4 microkernel[2]. Now DROPS could provide to concurrent execution of many instances of L4Linux like Xen but there is no feasible interface to implement communication between several instances of l4linux or l4linux and real-time task. In this paper the author design an efficient and safety communication protocol named l4fifo for these communications. L4fifo is an implementation of FIFO protocol for DROPS system mainly considering about efficiency and safety.

1 Introduction

DROPS originally aimed to contruct distributed real-time operating system with guaranteeing a certain level of services to applications. Different from RTLinux/RTAI and other real-time extension of time sharing system the DROPS is subject for real-time capacity from scratch based on L4 microkernel. Later L4Linux is introduced as a key component into DROPS system and the DROPS could also co-exist with real-time tasks and non real-time task cocurrently like RTLinux/RTAI. Similary both of them could use modified Linux named L4Linux as a time-sharing system for general purpose.

Fiasco is a L4 implementation by TU-Dresden operating systems group and now it could fufill the L4 version 2 standard. In contrast with L4Ka::Pistachio and NICTA::Pistachio-embedded Fiasco is real-time capacity with preemptibiltiy. Fiasco is the base in the whole DROPS system.

L4Linux could provide the full compatibility for Linux (now support x86 and ARM) and have a little performance loss comparing with native Linux[3][4]. Typically in the DROPS system L4Linux coud be executed concurrently as many instances with only limitation of hardware performances.

Although under-constructed L4Env is a programming environment for application development on top of the L4 microkernel family and it could provided such as System Resource Manager(roottask), Dataspace management (dm_phys, dm_mem,dm_generic), Linux Device Driver Enironment (dde_linux), a full Uclibc support, etc. But compared to popular POSIX it is not enough and it is necessary to provide more feasible and more standard interface for developers[5].

This paper is organized as following. In the section 2 we describle the l4fifo architecture.The lock-free mechanism will be shown in the section 3 and Section 4 describles the pinned memory. Section 5 discusses dataspace protection and In the section 5 there is a future work description.

2 L4FIFO Architecture

From the abstraction we know l4fifo design must consider the safety and efficiency because the original aim of L4 is for how to make operating system more safety and how to make the microkernel have the least performance loss[6]. Therefore, communication itself is not first but it is indeed the basic function of

l4fifo and many methods or mechanism will be used to purse the safety and efficiency.

In the l4 architecture, anthing is server which could provide service for anything calling this server as following.

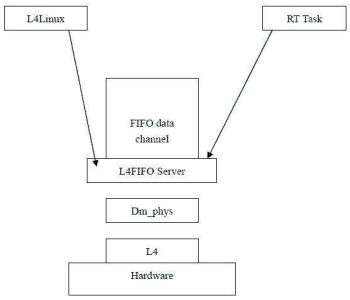

FIGURE 1: *Architecture of L4FIFO*

3 Lock free

Mutual exclusion has been a well-known resolution for sharing resources in the computer system for a long time and but lock-based algorithm will have more performance loss comparing to lock-free algorithms. In the real-time system the lock-based algorithms will cause the priority inversion problem especially in the communication between different prioriry processes. Nonblocking or lock-free algorithms could provide provide full preemptability and allow for multi-CPU concurrency. In addtion, priority inversion could be avoided[7]. These features cause lock-free algorithms very interesting for real-time systems. Especially L4/Fiasco is a lock-free-based real-time microkernel[7].

L4fifo implementation is based on lock-free mechanism. Basically all the lock-free algorithms base on compare-and-swap on top of CAS (x86/32 ,x86/64) or CAS2(x86/64)[7]. The choice of CAS or CAS2 is independent on the complexity of data structure and the target architecture. In the l4fifo what we need to compare and swap is the length of address which directly decides to use CAS or CAS2.

In the l4fifo because the reader or writer could operate the same address in a short period so that the other doesn't know whether the target address has been modified. So the reader or writer needs to check the end address of reading or writing last time and if they are not equal the last address could overwrite the present address and the last address is correct. The following is CAS code segment in the IA-32 Arechitecture.

```
#define CAS(adr, ov, nv) ({                          \
    __typeof__(ov) ret;                              \
    __asm__ __volatile__(                            \
    "cmpxchg %3 %1"                                  \
    :"=a"(ret), "=m"(*(volatile char *)(adr))        \
    :"a"(ov), "r"(nv)                                \
    );                                               \
    ret;                                             \
    })
```

4 Pinned memory

fifo is used for communications between different tasks and it has to take frequent writing and overwriting with application execution. In common when the memory demand is sufficiently slow the system will move part of memory image to the hard disk and when the system is free the removed the memory image will be back to the system. After the frequent-used memory belong int to the l4fifo is moved to the hard disk the next l4fifo operation has to wait for the reinitialization of l4fifo memory. In this case fifo operation will be slow especially the system is very busy.

A good method is to enable these memory pages to be maitained in the system all the time. Of course if there are too many pinned memory segments in the system the swap partition has a bad efficiency with the replacement of memory region.That means that once the memory is mapped to a client, no pagefaults will occur accessing that memory.

Luckily, in the l4env packages, the dm_phys could provide pinned memory directly. We could use dm_phys to create a 16M dataspaces subject for l4fifo.

```
dm_phys Cpool=1,0x00100000,0x00300000,0x00400000,fif
```

5 Dataspace protection

Dataspace is a concept from SawMill virtual memory framework in the context of L4 microkernel and it is an unstructed data container[8]. For examples dataspaces are files, anonymous memory, frame buffer, etc.

Acess to any form memory is provided via dataspaces which are unstructed data containters for any type of data. Dataspaces are provided by dataspaces managers. Clients call dataspaces manager to create data space and obtain a data space descriptor. A descriptor can be used to map dataspaces from one address space to another using IPC. Access rights can be restricted in such mapping operations. Dataspace managers construct from other dataspaces of which some may represent physical memory[7].

When the fifo client requests data from fifo server the dataspace manager provides a mapping

to the client and the client could proceed to use the data.dataspace is better to be sure the safety of data acess.

6 Future work

This paper is only a design technical report and all the implementation of l4fifo is not finished yet. So in the future the first work is to get an available l4fifo not a thesis. And then we should have a benchmark about the l4fifo and check its efficiency.

References

[1] *http://os.inf.tu-dresden.de/drops/overview.html*

[2] *http://os.inf.tu-dresden.de/L4/LinuxOnL4/overview.shtml*

[3] *The Performance of 08-Kernel-based Syst* H.Haretig, M. Hohmuth, J. Liedtke, S. oberg, J. Wolter Appeared at 16th SOSP,, Technicl University Dresden

[4] *General Performance Assesement of the L4/Fiasco Micro-kernel* Cheng Guanghui, Zhou Qingguo*, Nicholas Mc Guire, Wu Wenzhong ,Appeared at 7th RTLinux Workshop 2006, Lanzhou, China

[5] *http://os.inf.tu-dresden.de/l4env/doc/index.xml*

[6] *Can we make operating systems reliable and secure?*, Tanenbaum, A.S. HerderJ.N. Bos, H,Dept. of Comput. Sci., Vrije Univ., Amsterdam, Netherlands IEEE Xplore

[7] *Pragmatic Nonblocking Synchronization for Real-Time Systems*, Michael Hohmuth Hermann H01rtig Proceedings of the General Track: 2002 USENIX Annual Technical Conference /bibitempaper 8 *The SawMill framework for virtual memory diversity* Aron, M.; Yoonho Park; Jaeger, T.; Liedtke, J.; Elphinstone, K.; Deller, L. Computer Systems Architecture Conference, 2001. ACSAC 2001. Proceedings. 6th Australasian

[8] *PCI 9656BA Data Book Version 1.0*,2003, PLX Technology, Inc.

[9] *RTLinux3.1 Getting Started with RTLinux*,2001, FSM Labs, Inc.

[10] *Advanced filesystem implementers guide, introducing XFS*, Daniel Robbins,01 Jan 2002,Gentoo TechnologiesInc.