Persistent Process with Log-Structured File System

叶 萌　　河野 真治
Meng YE　　Shinji KONO

琉球大学 理工学研究科 情報工学専攻
Department of Information Engineering, University of the Ryukyus

概 要

We design persistent processes on top of the BSD/OS Log-Structured File System(LFS). It can be used as a fast hibernation mechanism for a Note PC. The swap area of the process is put on the LFS, which enables faster recovery time and safer system off. This requires extra copy from the changing process memory space to the swap space, which is done by special daemon process.

1 Introduction

The concept of persistence may be defined as the attribute of data which specifies its period of existence. In conventional Operating System, services are separated into files and processes, and only files are persistent. This separation works well for simple applications such as editors, but it is no longer suitable for complex service such as Presentation tools.

For example, a Presentation may contains several pictures and text files. Each pictures and text files are coupled together by the tool and these couplings are meaningless without particular tool. Since the presentation itself has complex states such as it-is-in-presentation or it-is-in-edit. Giving persistency for each components is not useful in this case.

A conventional solution is that the application programmer defines complex file format including pictures, text and application states as different objects. Each object is associated with other objects by the object attributes. But the result of these solutions are plenty of different complex file formats. This is a disaster.

Persistent process gives another solution. Instead of define complex file formats, an Operating System provides a persistent process flame work. We can communicate with persistent application with conventional inter- application communication such as cut and paste, or print. In systems that support persistence, if an inter-process communication protocol is agreed, the application programmer don't have to manage the movement of data between different data formats. We don't need complex file formats, only the proper set of inter-application communication protocols are necessary.

Persistent process is important from other point view. Many of Note PC supports hibernation mechanism, which enables complete power off during off-duty to save battery power. It is usually implemented by registers and memory copy to the disk of Note PC. But it requires hundreds megabytes of copy which takes several minutes. In case of failure, for example, battery off during disk copy, entire saved state is lost. The persistent process architecture is used as a hibernation mechanism for a Note PC. Based on this architecture, the current processes exist in physical memory and virtual memory are recovered easily and quickly in the event of system failure. Copying to the disk is performed during normal computation idling instead of the last minutes before battery empty.

Recovery from hibernation can be faster by using persistent process. Because we don't have to restore entire process memory, OS can leave unnecessary persistent image in the disk. Conventional hibernation mechanism requires copy of entire physical memory which may be 128M-bytes or more.

This paper describes a new persistent processes architecture based on LFS. LFS is a file system based on logging structure. It provides a safe storage for The resilient persistent processes data. The storage enables faster recovery time and safer system off.

A daemon is designed to manage the movement of process data between the memory and the swap area put on the LFS. It supports full address space i.e. it supports persistent virtual memory.

2 Persistent Process

Persistence describes something that exists beyond its expected lifetime or that lasts after program completion. It is a mechanism to trigger saving or restoring the object state, either automatically or on a command.

Persistence is usually implemented by preserving the state of an object between executions of the program. To preserve the state of the object, the object is put on some kind of long-term storage media (usually a disk). This copy can be a partial or incremental copy, but the saved states have to be consistent, which means it have to be equal to the copy of entire process memory. This timing is called checkpoint and we call the saved image snapshot. The persistence of process is different from the LOAD and SAVE behavior. Persistent processes are removed from active state and its saved image are copied to the file system. This is called FREEZE. Frozen persistent processes become active by attaching saved image to the swap and active process list. This is called MELT. Instead of LOAD or SAVE, persistent processes are FREEZE or MELT.

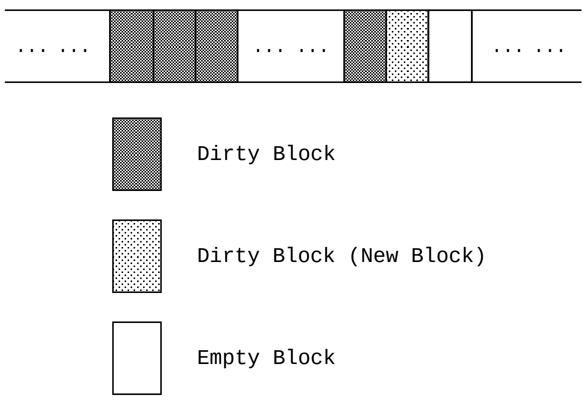

図1 Log-Structured File System

3 Swap on LFS

Basic idea of this persistent process flame work is to use swap space which implement the imaging from memory to Log-Structured File System (LFS is used in the left part).

Log-Structured File System improved the file system performance by storing all file system data in a single, continuous log. It is optimized for writing without seek. It is also optimized for reading, because all data within files are stored continuously on disk.

LFS stores the disk data blocks in a single, continuous log structure. In LFS, the disk is laid out in segments. Each segment consists of a summary block followed by data blocks and inode blocks. LFS is described by a superblock similar to the one used by FFS. The super block is replicated and occurs as the first block of each segments.

The swap space of a persistent process is a file on LFS. Unlike normal swap, this swap must contains complete snapshot of the process. Of course we don't have to copy read-only part or shared program text. Since LFS provides con-

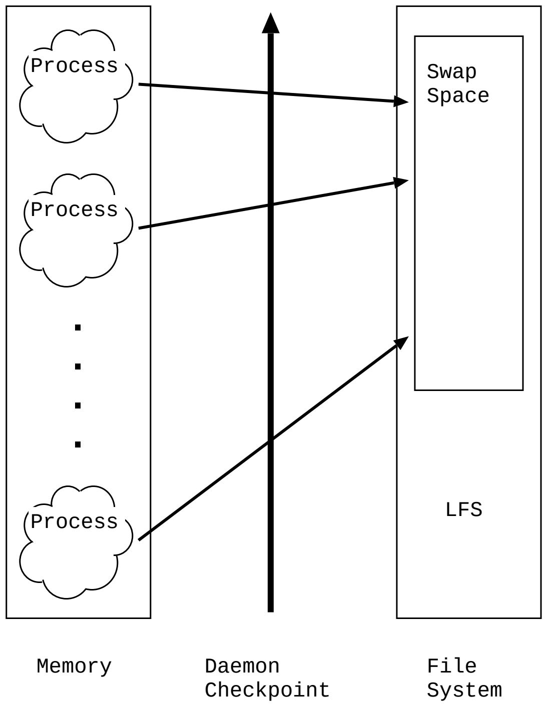

図2 Persistent Process on LFS

tagious write, it is suitable for snapshot. But if we keep track every small modification of process memory, system becomes very slow.

4 Checkpoint Daemon

Snapshot is done by several minutes, like updated in Unix. A special daemon is used to implement checkpoint for memory space. It copies the modified part of persistent process memory space to the swap space on LFS. We use two special signals. Checkpoint daemon send weak snapshot signal. The weak snapshot signal do checkpoint when the persistent process is idle, or the process is running, checkpoint will not be proceeded. Another signal is force snapshot. It do checkpoint after process suspension. Snapshot and process activity are racing each other. If a process modifies large amount memory continuously, the process have to be stopped to make consistent snapshot.

5 Freeze, Melt and Recovery

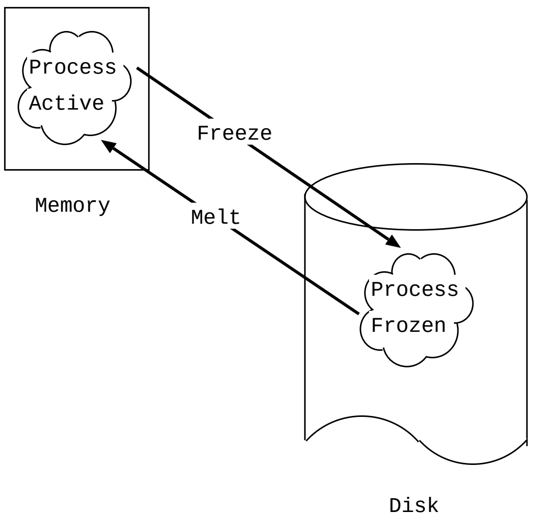

図3 Freeze and Melt

Freeze and Melt is somewhat like Suspend and Resume. Suspend signal remove a process from active queue and the process becomes suspend state. Freeze also remove the process from active queue, but it also removed from process queue itself. Checkpoint are performed and its image are no more active swap. It becomes a file. It is incompatible with other OS like binary command. To make a copy among different OS's, some Inter-application communication protocol is required.

The frozen persistent process is melted to be active process. The process is put in active process queue and the process image file is attached to the process swap. There is no instant copy of entire process image which can be very large. Copy is done by demand driven and virtual memory mechanism do the job. We don't have to restore a process behind a window immediately.

In case of system failure or recovery from hibernation, all persistent process are melted from its LFS swap. It is not necessary the latest one. The latest consistent snapshots are recovered. Partially snapshot-ed process are easily checked. Since these are written in contiguous segment of the disk.

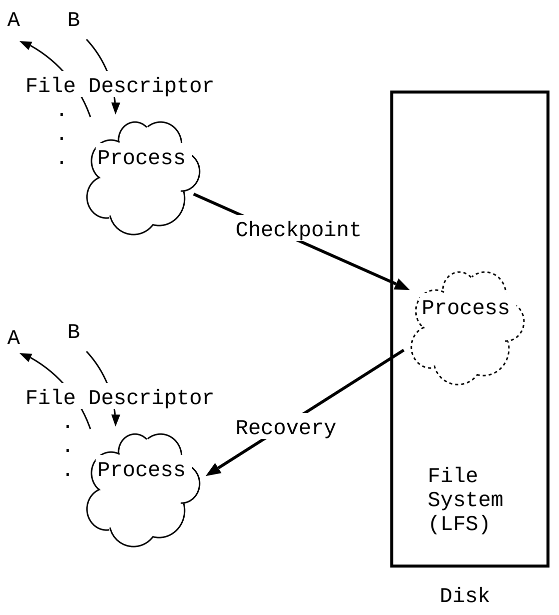

図4 **File Descriptor Reconnection**

6 File Descriptor Reconnection

After melt or recovery, resuming activity is not enough. Meaningful process have to communicate with other process, such as X-Window and its clients. Unix uses File Descriptor to describe the file opened by the current process. The file may be all kinds of formats - i.e. a file exists in the disk, pipe, network, I/O, etc. We have to reconnect all these connection. Connections should be preserved among persistent process. Connection between persistent process and non-persistent process can be lost.

To connect file descriptors, procfs is used to store the states of process. `/proc/1234/fds/3` means file descriptor No. 3 in process No. 1234. If it is connected files it becomes a symbolic link to the target such as a file or other file descriptor. Status of file descriptor including pseudo tty status have to be included in the snapshot of the persistent process.

When daemon stores memory by checkpoint, the File Descriptor of each process should be stored, so that the File Descriptor can be rebuilt in the event of recovery.

7 Fast Recovery for Hibernation

Many UNIX operating systems use Fast File System. When FFS write a data to disk block, several pieces of information need to be modified: the block, the inode, the block map, and the location of the last allocation. If the system crashes, the FFS must rebuild the entire file system state, including the block map and meta data. And FFS need to check the file system structure and block pointer. Traditionally, fsck is the agent that performs those operations in FFS.

LFS adds the new data block to the end of the log. So data in LFS are never overwritten. LFS used a checkpoint approach to store current file system state.

The traditional Unix file systems store all data in files randomly, so the contiguous data in the file may be stored in the uncontiguous disk blocks. In the event of system crash, the hibernation mechanism on top of traditional Unix file system spends much time to seek the data describe the last memory state. In LFS, all data in files are stored contiguously on LFS, so the data seeking takes less time than traditional Unix file system. Taking advantage of it, a fast hibernation mechanism can be implemented for Note PC.

8 Conclusion

Persistent process with LFS can provides uniform interface for complex application. It also makes fast hibernation and fast recovery. We are implementing Persistent Process Framework on top of BSD/OS.

参考文献

[1] Seltzer, M., Bostic, K., McKusick., M., Staelin, C. A Log-Structured File System for UNIX, Proceedings of the 1993 Winter Usenix Conference.

[2] David Hulse, and Alan Dearle, A Log-Structured Persistent Store, Proceedings 19th Australasian Computer Science Conference, 1996.

[3] Mendel Rosenblum, The Design and Implementation of a Log-Structured File System (Kluwer International Series in Engineering and Computer Science), Kluwer Academic Pub, ISBN:0792395417.