Design and Implementation of Dynamic Adaptive File System for Mobile Computing

Michinobu Tanaka

School of Information Science,
Japan Advanced Institute of Science and Technology

February 14, 1997

Keywords: mobile computing, adaptive system, object graph, file system, APM, flash memory.

This paper presents a dynamic adaptive file system that changes resource management policies or modifies system components dynamically according to the condition of mobile computers.

In mobile computing environments, the condition of system services is changed dramatically due to many reasons. First, the PC Card Interface that is widely used in portable computers recently, enables us to change the configuration of a mobile computer by replacing such devices even when executing applications. Second, the speed of portable computers are slow, and they have small memory size and disk capacity, poor display resolution and size. Consequently, executable application numbers and types are restricted. Also, these limit the number of executing applications. And these days, there are so many kind of applications such as RDBMS, Multimedia, WWW, and these application require different services each other.

For these reasons, we expect the system that can change resource management policies or modify system components modules dynamically by consideration of what the system's hardware elements are, or on what an environment executing, or what an application is running. One example, it is important to save the battery consumption while mobile computing, because of executing as long as possible. So it is preferable that an application uses a resource management policy or an algorithm that reduces the number of time to

access to power-hangly disk or to send data. But while executing with AC-line, there is no need to save battery consumption, then it prefers using a high performance policy or algorithm.

For requirement of dynamic system changeability, our system is implemented by using *object graph* constructed with *composite objects*. Composite objects are organized with more then two objects that are connected with statically dependency, and they provide some sets of functions like as file services. It is called a *component object* that constructs a composite object. And the relation that's decided statically which object depends on another seems to be reference to an object. This relation represented by the graph is called *object graph*.

There are basic operations of object graph with composite objects.

- copy of object graph
- destruction of object graph
- replacement of object graph
- insertion of object graph
- deletion of object

Applying these operations to copy object graph from an original one, it can make a new object graph. Our system can dynamically change system components by using this object graph mechanism.

The main goal of this file system is to adapt to an environment change that can be dynamically changed. The environment is typical in mobile computing environments that the system provides services with saving battery resource, and while connected AC-line, the system provides services with high performance. For supporting adaptation to mobile computing environments, we define three system conditions; one is connected AC-line, and second is using battery charged high, and third is using battery charged low. To adapt each conditions, we provide three composite objects, and adaptation to some one is changing composite objects.

Considering power saving about a file system, there are two requirements. First, reducing disk spin time. There is a large difference in power consumption between a disk that is spinning and one that is not. To reduce spin time, our file system keeps changed data in physical memory as long as possible. But, if battery condition changes low, changed data must be written in storage device. Besides keeping data in memory, for reduce power consumption effectively, our file system uses a flash memory PC-Card as a part of disk. Because flash memory's electric power consumption is smaller then disk's one, it is expected that total electric power consumption is reduced.

Second requirement is page-in and page-out control. Virtual Memory subsystem provides a large address space more then physical memory. If the amount of physical memory is not enough, virtual memory move some pages into storage device like disk. Therefor this operation causes disk accesses, then to reduce spin time is not good. A summary of requirements for power saving, a file system has to control I/O between disk and memory.

For more efficient I/O control, an application specific resource management is necessary. So our file system enables application to change a resource management policy. For this, our file system is implemented as a library. It is not only flexible and extensible, but also high performance because this library file system is implemented on RT-Mach micro-kernel. And then to manage I/O control in detail, this file system is developed by making from object filing system, *Texas*. *Texas* developed in the University of Texas[1], is a persistent storage system for C++, providing high performance while emphasizing simplicity, modularity and portability.

To manage I/O control, we modified Texas by using *external memory pager*, that's memory manager extracted from kernel. Because the external memory pager can manage memory, our file system reduces disk access and controls paging by using this mechanism. Our file system controls the total memory size used by an application, and provides an application specific memory management.

An application linked with the dynamic adaptive file system library can behave adaptively in mobile computing environments.

[1]http://www.cs.utexas.edu/users/oops/