Real-Time Scheduling Support in Ultrix-4.2 for Multimedia Communication[*]

Tom Fisher
fisher@tenet.Berkeley.EDU

The Tenet Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720
and
International Computer Science Institute
1947 Center Street
Berkeley, California 94720

Abstract. Advanced workstations are now being equipped to handle various types of multimedia applications such as audio and video. The quality of these images and sounds depends highly on the timely management of multimedia data. In order to provide such management, support for real-time process scheduling must exist. In this paper we describe the fundamental mechanisms implemented in the Ultrix4.2 kernel that provide the real-time scheduling support needed for multimedia applications. Our primary goal is to reduce and bound the overall delay in the response times to both internal as well as external events. To achieve this goal we have modified the Ultrix4.2 kernel to be preemptible at highly specific locations and have added a small subset of data structure locks to maintain data consistency. Work has also been done towards reducing and bounding the time spent resolving lock conflicts. The end result is that we have a kernel capable of the real-time processing necessary for multimedia applications. Finally, we demonstrate and validate through experimentation that the above claim is true.

1 Introduction

When we talk about real-time in the Tenet Group in a broad sense we are referring to guaranteed performance, however in this paper, real-time as in real-time scheduling, refers to scheduling processes under bounded response times. In

---

[*] This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

order to achieve such bounded response times there needs to exist greater control over scheduling behavior. To achieve this greater control several areas of real-time functionality have been added to the Ultrix4.2 kernel. First, we discuss in detail the implementations in each of these areas. Then we proceed to validate the additional functionality through measurements. We note that in order to achieve valid and accurate measurements, all areas of the entire system must be vigorously and continuously exercised. It is only through this type of exhaustive measurement process do we establish guarantees about the work.

In addition, it must be understood that we cannot totally solve this problem of bounding the existing delays by software alone. We do not attempt to reduce nor bound the delays caused by interrupt processing and/or DMA processing, even though both types of high priority activities in the lower half of the kernel may produce very noticeable delays. To adequately bound such delays would require hardware support.

The rest of this paper is organized as follows: Section-2 states our objective. Section-3 describes and discusses our implementation. Section-4 describes our performance measurements. And, section-5 gives our conclusions.

2 Objective

Scheduling latency is defined in the context of this paper to be the delay between the time when an event (or request) takes place and the time when execution to handle (or respond) to that event actually occurs. The primary goal of our work is to reduce and bound the duration of this delay. In particular, the target is to reduce this delay to below $3ms$ for close to %100 of the time while the machine is operating under loaded conditions.

3 The Implementation

The fundamental mechanisms that make up the real-time scheduling features added to the Ultrix4.2 kernel are included in 5 areas: real-time process establishment, preemption points, data structure locks, an implemented scheme for resolving lock conflicts, and a separate real-time callout mechanism. This combined functionality provides for the scheduling control needed to reach the above stated objective.

3.1 Real-Time Process Establishment

The notion of a real-time user process has been introduced into the Ultrix kernel. In this context, a real-time process is simply a normal user process that has explicitly been accelerated to real-time status. It is this real-time status, which any user-process in the system is equally able to gain, that we would like to describe. In order to obtain this real-time status several things must occur:

Admission control. First of all, the kernel uses a simple admission control scheme to accept or reject a process's request to obtain real-time status. The scheme, at this point, is straight forward and simply allows only 4 (user-requested) real-time processes to exist at any one time, and rejects all others. It is possible, however, for the system to accelerate specific processes, and exceed this limit. This is done to reduce and limit the duration of time spent handling lock conflicts and will be described later (see section 3.4). In the future, we expect this admission control scheme to take into account multiple user-defined real-time priorities as well as user-defined resource requirements (or requests). The user process, under this new scheme, would then negotiate and haggle with the kernel until an agreement, if any, is made about the nature of the real-time status to be allocated.

Initialization. After an approval (or agreement) has been made, the process's pages are locked in memory, an entry in the system wide real-time process table is allocated and initialized, and the user/kernel process priorities are changed to a static real-time UNIX priority (see next section). And finally, a global flag is set signaling the kernel that execution of a real-time process is now pending and that preemptions are now enabled. Once a preemption has occurred and the real-time execution is under way, the global flag is turned off, preemptions are then disabled and the kernel proceeds to operate as normal.

Assignment of Priorities. A real-time process has associated with it several priorities. These are a static UNIX priority and 3 kinds of real-time priorities.

When a process is first accelerated to real-time status it is assigned a static UNIX priority. This priority, referred to as the process's real-time UNIX priority, defines the overall position of the priority of real-time processing in relation to all other high priority activities in the system.

In order to distinguish between real-time processes of various urgency we have adopted the notion of global, inherited and effective priorities from SunOS5.0 [Khanna92]. The use of these priorities is further described in section 3.4.

3.2 Preemption Points

Preemption points are highly specific locations in the kernel where a process may choose to give up its execution of the CPU in favor of a higher priority process. This approach is much simpler for us than putting locks on all data structures and making the entire kernel fully preemptible. We do not say that this is the correct approach, but only that it's the simplest way for us to achieve our needs which are to satisfy the requirements of multimedia applications. Other work has successfully been done to make the kernel fully preemptible, for example in OSF-1.0, SunOS-5.0, and HPUX.

Initially our