Processor Capacity Reserves:
Operating System Support for Multimedia Application

Clifford W. Mercer, Stefan Savage, and Hideyuki Tokuda

Real-Time Computing and Communications Lab.
Hanyang University
Kangwon Lee
2006. 11. 13

Contents

Introduction

❒ Motivation

- Multimedia applications have timing requirements that cannot generally be satisfied with using the time-sharing scheduling algorithms of general purpose operating systems

❒ Goal

- To design a processor reservation strategy for supporting continuous media applications

Introduction

❒ Processor capacity reservation mechanism

① Applications request processor capacity reservations
② Reservation has been granted by the scheduler
③ The application is assured of the availability of processor capacity

- Applications are also free to increase their reservations at any time during execution, and they are always free to decrease their reservations

Reservation strategy

- The reservation strategy must:
  - provide some means for application programs to specify their processor requirements **(capacity specification)**
  - evaluate the processor requirements of new programs to decide whether to admit them or not **(admission control)**
  - schedule programs consistently with the admission control policy **(scheduling)**
  - accurately measure the computation time consumed by each program to ensure that programs do not overrun their reservations **(reservation enforcement)**

Admission control and scheduling

- A scheduling framework based on the rate of program progress provides an effective environment for implementing processor reservation
- Rate of program
  - Periodic
    - The rate can be determined from the period that the programmer has in mind and the computation time during that period
  - Non-periodic
    - The rate arises from delay requirements

Admission control and scheduling

- The rate alone does not fully specify the timing attributes of a program
  - Ex) A program could specify that it requires 30% of the processor time to run successfully on a given machine
- We have three values which describe the processor requirement for a program, and two of these are required to specify a processor percentage

$$\rho = \frac{C}{T}.$$

ρ : process percentage

C : computation time

T : real-time period

Admission control and scheduling

- Delay for a non-periodic program which executes at a given rate

$$D = \frac{S}{\rho}$$

  - Using the largest acceptable delay yields the smallest acceptable rate of execution

Admission control and scheduling

❒ Admission control under fixed priority scheduling

- **Assumptions of RM**
  - Programs are periodic, and the computation during one period must finish by the end of the period (its deadline) to allow the next computation to start
  - The computation time of each program during each period is constant
  - Programs are preemptive with zero context switch time
  - Programs are independent, i.e. programs do not synchronize or communicate with other programs

Admission control and scheduling

❒ Admission control under fixed priority scheduling

- All of the programs will successfully meet their deadlines and compute at their associated rates if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1).$$

- When n is large, $n(2^{1/n} - 1) = \ln 2 \simeq .69$

Admission control and scheduling

❒ Admission control under fixed priority scheduling

- How to increase the Utilization of RM
  - using unreserved time for unreserved background computations.
  - using the exact analysis to determine whether a specific collection of programs can be scheduled successfully
- On average, task sets can be scheduled up to 88% utilization

Admission control and scheduling

❒ Admission control under dynamic priority scheduling

- The earliest deadline (ED) scheduling policy is effective for scheduling periodic programs such as our continuous media programs
- Under the same assumptions outlined in the section on rate monotonic scheduling, all programs will successfully meet their deadlines under earliest deadline scheduling if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1.$$

Admission control and scheduling

- **Discussion**
  - Earliest deadline scheduling seems preferable to the rate monotonic scheduling
  - The 69% bound for RM is pessimistic, but 88% is a more realistic bound
  - The reservation bound of rate monotonic can be 100% for the special case when all periods are harmonic
  - The amount of unreserved computation time of about 5-10% may be necessary to avoid scheduling failures due to
    - inaccuracy in the computation time measurement and enforcement mechanisms
    - the effects of critical regions and other synchronization and communication among programs

Reservation enforcement and reserves

- Measurement
  - Enforcement mechanism monitors processor usage by measuring the time each program is executing on the processor
  - It charges this computation time against the reserve associated with the program

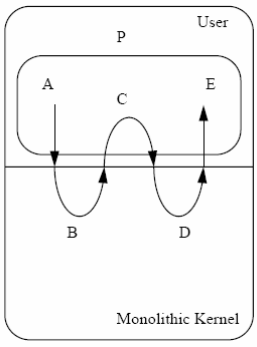
- Control policy
  - Programs which have not yet consumed their reservation take precedence over unreserved programs
  - If there is unreserved processor time available, unreserved programs can take advantage of the extra processor time

Reservation enforcement and reserves

- In many operating systems, the processor usage for each process is recorded in a per process logical clock
  - logical clock = user time + system time
- The combination of these two values is an accurate long-term measure of the computation time used by the process

Reservation enforcement and reserves

- **Monolithic Kernel Accounting**



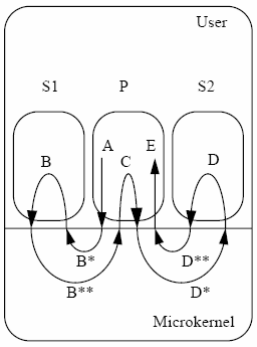
P(total usage)=A+B+C+D+E

Pu(user usage)=A+C+E

Ps(system usage)=B+D

Reservation enforcement and reserves

❒ Microkernel Accounting



◆**The monolithic system accounting scheme**

P(total usage)=A+B+C+D+E

Pu(user usage)=A+C+E

Ps(system usage)=B+D

◆**The microkernel accounting scheme**

P = A + B + B* + B** + C + D* + D** + D + E,

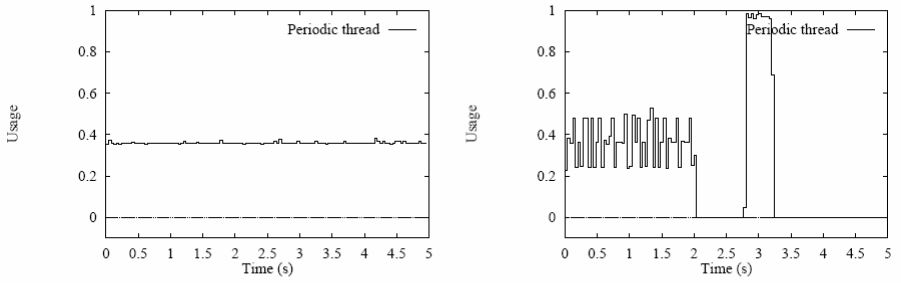
Performance evaluation

❒ Environment

- **Real-Time Mach (version MK78)**
- **Gateway 2000 33MHz 486 based machine with 16MB RAM**
- **Alpha Logic STAT! timer board**
- **CMU UNIX server (version UX39)**

Performance evaluation

❒ Unreserved periodic thread

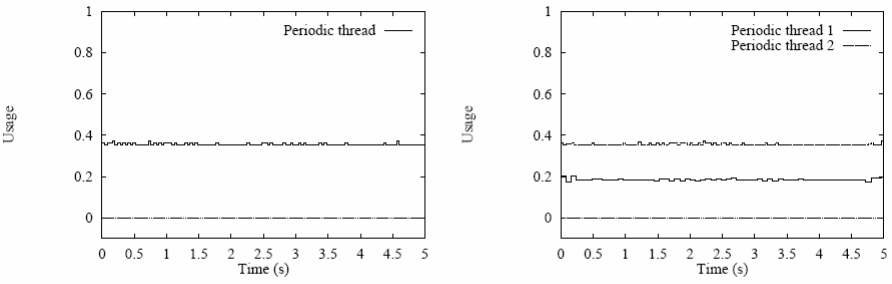


(a) No competition (b) With competition

Figure 3: Unreserved periodic thread without and with competition

Performance evaluation

❒ Reserved periodic thread

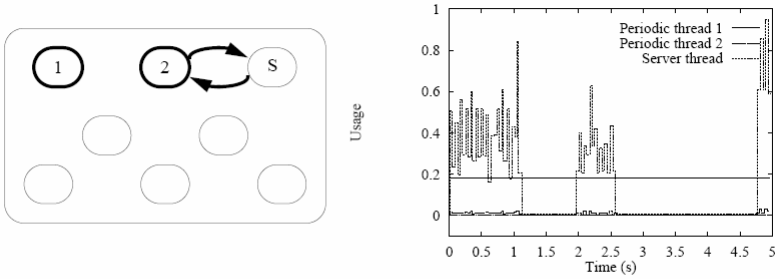


(a) One reserved thread (b) Two reserved threads

Figure 4: Reserved periodic thread(s) with competition from unreserved threads

Performance evaluation

❒ Server invocation with no reservation

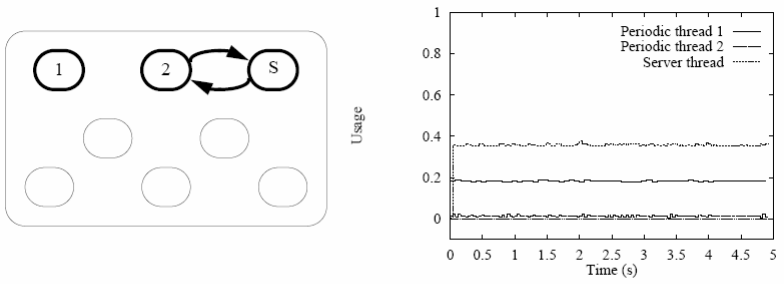


(a) Task set structure (b) Measured usage behavior

Figure 5: Uncoordinated client/server reserves

Performance evaluation

❒ Server invocation with reservation



(a) Task set structure (b) Measured usage behavior

Figure 6: Integrated client/server reserves

Conclusion

- Scheduling framework
  - Based on computation rates, it provides an effective way to specify processor requirements
  - Both RM and EDF scheduling algorithms are suitable for implementing reservation in this framework

- Accounting mechanism
  - It is well suited for the microkernel architecture
  - It tracks processor time used by individual threads which call on user-level servers to perform system services.