2N-4

ADAPTING THREADS ON SPP(Self synchronizing Parallel Program) PROCESSES IN A UNIX ENVIRONMENT

Bernady O. Apduhan, Nobuhisa Tokuzaki, Tatsuhiro Ohshima,
*Itsujiro Arita

Dept. of Computer Science, Kyushu Institute of Technology
Tobata Campus, Kitakyushu City
*Faculty of Computer Science and Systems Engineering
Kyushu Institute of Technology
Iizuka Campus, Iizuka City

1. INTRODUCTION

The SPPs are parallel programs with a high-speed synchronizing mechanism utilizing FIFO(First In First Out) queue[4], to decrease the overheads due to the synchronization of operations in a multiprocessor system. We apply the concept of SPP in the design of the distributed operating system of our HYPHEN B-16 Multi-microprocessor System project.

In the sections that follow, we describe the abstraction of SPP processes, adapting the thread concept, in a UNIX environment.

2. THE SELF SYNCHRONIZING PARALLEL PROGRAM

A sequential program, with consideration to its data-dependency relation and by applying appropriate rules, can be transformed into a precedence graph, known as Self synchronizing Parallel Program[3]. SPPs are represented in terms of processes, segments and tasks. A task is a unit of execution in SPP and specifies a sequence of operations that are executed continously without interruption. A segment is a unit of allocation, i.e., all the tasks included in a segment must be allocated to the same processor module. The program structure of SPP is shown in Figure 1.

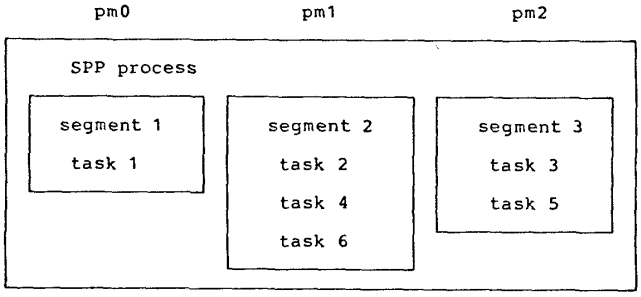

Figure 1. The Program Structure of SPP.

In order to schedule the execution of the tasks, each processor module has its own FIFO queue where the entry addresses of the tasks to be run are placed or removed by two special purpose instructions; namely, 'parallel branch' and 'exchange task', which operate on the FIFO queues.

3. THE SPP PROCESSES IN A UNIX ENVIRONMENT

3.1 The Driving Forces

The development of the new SPP process abstraction was motivated by the increasing complexity of distributed and multiprocessor environments and the imminent needs of modern applications. In the conventional UNIX, many different system calls are involved in process control, resulting in high overhead on the part of the operating system.

3.2 The Abstraction Environment

The thread is the basic unit of CPU utilization in the Mach kernel[1], which is roughly equivalent to an independent program counter operating within a task. It is the specification of an execution state within a task. All threads within a task share access to all resources. Mach allows multiple threads to exist(execute) within a single task. A task is generally a high overhead object(much like a process), whereas a thread is relatively low overhead object. On tightly coupled shared memory multiprocessors, multiple threads within the same task may execute in parallel. Thus, an application can use the full parallelism available, while incurring only a modest overhead on the part of the kernel.

The above mentioned features and capabilities of threads, make it so attractive to our multi-microprocessor system project. The adaptation structure is shown in Figure 2. The

threads correspond to the SPP segments. The thread concept allows multiple SPP tasks to exist within a single SPP segment, but one SPP segment may contain many SPP tasks. For multiprocessor systems, threads utilizes the wait system call for synchronization. In the HYPHEN B-16 Multi-microprocessor System, the 'EXchange Task' instruction is used and multiple SPP tasks from one SPP segment may be executed in parallel. An SPP task may be created, destroyed, suspended and removed.

Likewise, SPP segments may also be suspended or resumed. An SPP task may only execute when both it and its SPP segment are resumed. Resuming an SPP segment does not cause all component SPP tasks to begin executing, but only those SPP tasks which are not suspended.

4. CONCLUSION

With this new abstraction of SPP processes, we anticipate to gain substantial performance benefits. As of this writing, the verification and development on the above mentioned model of SPP processes are well under way on the Apollo DOMAIN, before its implementation to the HYPHEN B-16 hardware. The on-going development of this new model of SPP processes has lead to the key aspects and considerable refinements in the design of the HYPHEN B-16 Operating System.

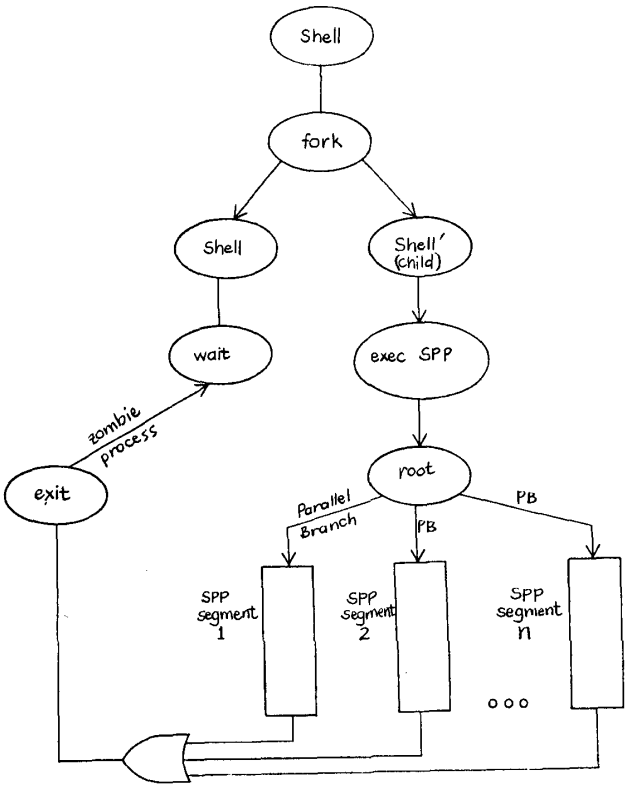

Figure 2. The Adaptation Structure.

REFERENCES

[1] Accetta, M., et al., "Mach: A New Kernel Foundation for UNIX Development", in Proceedings of USENIX 1986 Summer Conference, pp.93-112.

[2] Apduhan, B.O., et al., "The Implementation of SPP(Self synchronizing Parallel Program) Kernel Functions in a UNIX Environment", 1987 Denki Kankei Gakkai Kyushu Shibu Rengoukai Taikai Rombunshou, Okinawa, October 1987.

[3] Arita, I., "On a Parallel Program with Synchronizing Mechanism using FIFO Queue(I) -Self synchronizing Parallel Program", Trans. IPSJ, Vol.24, No.2, 1983(in Japanese).

[4] Arita, I., Sueyoshi, T., " On a Parallel Program with Synchronizing Mechanism using FIFO Queue(III) - Execution Control Mechanism", Trans. IPSJ, Vol.24, No.6, 1983(in Japanese).

[5] Baron, R.V., et al., " MACH Kernel Interface Manual ", Dept. of Computer Science, Carnegie Melon University, January 1987.