Solaris Multithreading Drives World's Fastest Printer

CASE STUDY

Solaris™ multithreading technology is at the heart of the world's fastest duplex PostScript™ printer, a 180-impressions-per-minute printing application launched in 1993 by the Australian postal service. Australia Post's EDIPost project needed a printing solution that would cut down its physical mail handling volume by sorting and transmitting customers' computer-based mail records across WANs for printing on demand at multiple local sites. The project specification called for high-speed PostScript printers attached to SPARCstations™ which were capable of 180 duplex (printing both sides of a sheet in one pass) impressions per minute.

An ImageFast 180 printer from Delphax Systems of Canton, Massachusetts was chosen, but it did not recognize the PostScript data stream. So Delphax and its Australian reseller Computer Resources Company embarked on a collaborative development effort. They brought in Sydney-based Uniq Professional Services, a consulting firm with considerable expertise in UNIX® development, who selected a JAWS PostScript Raster Image Processor (RIP) from 5D Solutions in the UK. The task of the team was to develop software to acquire and spool the PostScript files, schedule jobs, and integrate the RIP with the printer controller. The speed of the printer controller was the most critical element.

Designing for Throughput with Multithreading

In full duplex 180 impression per minute printing, the printer needs to receive a page image about every 300 milliseconds. A missed page can cause desynchs and blown images, and completely break down the printer's flow. It was clear that a multithreading approach would best enable overlapping the various tasks involved. One thread would figure out where the printer application should go, and another could handle the driver, diving down into and returning back from the driver (essentially, everything but I/O time).

Between the RIP and printer controller reside a set of mmapped files (mmap is a facility that lets a file be mapped into memory and treated like normal memory). The RIP sees the mmaps as arrays of bytes, "blank canvasses" to rasterize onto; the printer controller sees them as slots containing images to be put onto the printer. The RIP sends messages that essentially say something like "This page comprises this slot and this slot, and has these printing options."

Because an image may need to be rasterized multiple times, and because a given page may be printed simplex or duplex, the printer controller has three threaded subsystems. The message thread accepts the message and

SunSoft
A Sun Microsystems, Inc. Business

into a common page list. The I/O threads are structured with a page thread on top and work threads below it which assemble the page and issue the page commands to the serial threads. There is one serial thread for input and one for output. The output thread manages sending packets out to the printer, putting checksums around packets and handling acknowledgments. The input thread parses the incoming status from the printer.

Uniq's Kevin Sheehan explains, "After the page thread issues serial commands to tell the printer what kind of image is ready for printing and which print engine will be printing it, the work threads have to get those images to the parallel port quickly. For example, in a duplex print cycle, when page 0 comes in, the RIP rasterizes the images into the mmap files, then sends a message to the printer process. The message thread picks it up and creates a page structure with the list of pages to be printed, the number of copies, how many sides, what colors, etc. The page thread picks up this list, sends serial commands to the serial threads, and creates work structures; the work threads pick up those structures and dive into the driver.

"In this way, the page 0 image is going out as the paper is moving through the engine. When the paper hits the duplex engine, one thread has a write in for the first half. When that write I/O is complete, its thread comes back for the next task, while another work thread has already issued the write for the second half. It is never necessary to wait for a process to wake up and come back into the driver."

Because Solaris threads are so inexpensive to create and use, there was never any concern about putting artificial limits on the number of threads in the program. And because the number of threads is independent of the number of CPUs, the entire program could run on a uniprocessor SPARCstation™. Indeed, a single SPARCstation handles the user interface, raster image processing and all of the printer controller operations.

A New Programming Paradigm

"I don't think we could have delivered the necessary performance without Solaris multithreading, says Sheehan. "Threads gave us a different programming paradigm that simplified the development process. Typically in this kind of application there is a giant loop that asks 'Do I have any messages, is there any I/O to do, are there serial processes to run,' etc. The programmer must deal with all of these at once.

"With threads, this loop is broken into small modules: a message module, an I/O module and a serial module, each with two or three routines that call among themselves. We could also use the same architecture between threads and between processes. In some cases, when some code couldn't be written as a thread, we could still use the same code as a process. The development process was essentially looking at small pieces and the interfaces between them. With threads we can make our machines scream for mercy, especially in applications which combine computation and high-speed output.

"Once you start writing with Solaris threads, you can't go back to the old methods," Sheehan concludes. "Take the page thread. If we had to write that non-threaded it would have meant the classic control loop above. The original control loop code for was over 200 lines. Now the page thread is about 20 lines, the message thread about six lines, and the main serial thread about 20-30 lines. We were able to complete all the coding for this project in about one month, which would have been impossible without multithreading."

"This project shows what can be accomplished in a short time with a knowledgeable customer, expert collaboration by all of the players, and the right technology," says Bob Rushby, Vice President of Product Development at Delphax. "Not only have we satisfied Australia Post, we have also strengthened our product line with a PostScript version of our ImageFast Series 180 printer."

For more information about how you can take advantage of multithreading, please send email to Threads-Hotline@Sun.COM, or call 415 336-3032.

SunSoft
A Sun Microsystems, Inc. Business

2550 Garcia Avenue, Mountain View, CA 94043-1100 USA ● (415) 960-3200 ● Fax (415) 961-6070

Printed in USA 10/94 94337-001