Shaun Ahmadian
603-223-641

FLASH REPORT

When choosing a web server architecture there are many choices and each has its own benefits and drawbacks. Probably the most simplistic architecture is the Multi-Process (MP) model. According to this model every new request that the server receives causes a new process to be instantiated. Each process is responsible for one particular client request and they are independent of one another; that is, if one process happens to block (stall) the other processes are not affected in anyway. Another benefit of this independence is not having to deal with synchronization issues because each process has its own "private" address space. However, such convenience has certain drawbacks. Due to the need for completely separate processes there is a substantial amount of memory and kernel overhead. Although the execution of the processes can be interleaved, there is an enormous cost associated with every context-switch that needs to be made (which can be a lot considering there might be a hundred or so processes/requests). Also, the MP model uses memory less efficiently assuming each process has its own cache.

On the other hand, the AMPED architecture does not exhibit such drawbacks. AMPED follows an "event dispatcher" design pattern. Unlike MP, AMPED has one main process running at all times that calls upon other functions to perform particular tasks. However, in the case of a disk access operation, helper processes (or threads) are utilized to prevent the blocking of the main process. By having a single process the AMPED architecture avoids the overhead costs of context switching. And since it uses one single cache it does not have to deal with synchronization issues. Since the model uses "helpers" there is an overhead associated with maintaining them. In addition, since these processes are separate from the main process there is a need for inter-process communication (so that the main process is notified once the file has been retrieved from the disk). The IPC is one key cost associated with disk access in AMPED which did not have to be dealt with in the MP architecture. But it must be noted that in MP IPC would have to be used to consolidate data produced by the various processes.

In terms of simplicity of implementation, MP is the likely choice. MP's model is very similar to the abstract model of a web server, with each process being an identical copy of the other. So as long as a person could write procedures dealing with each task in the model, they can be called sequentially in the main function. The only issue that would have to be dealt with would be data consolidation.

On the other hand, AMPED has a much more "elegant" design which makes it more difficult to program. The model resembles more of a state diagram than the "uniflow" diagram of the abstract model. Since the server behaves like a state machine, the "state" of each request has to be maintained in order to determine which task to perform next. The individual operations/states would be programmed similar to those in MP since they are modular in nature. Furthermore, the "helpers" need to be implemented along with some IPC protocol. The main process would have to do a lot of maintenance work dealing with the various operations and helpers. In some ways the AMPED implementation seems much more "customized" compared to MP; that is a lot of the code has to be written from scratch and it cannot be reused anywhere else.

No matter how cumbersome the implementation of AMPED might be, the increase in efficiency and faster response time are well worth the effort. The web server, unlike the Therac-25, is a *time-critical* system. The satisfaction of the customers is not determined by accuracy but the responsiveness of the system. Although complex systems are more error prone than simple ones, the speed gains of the AMPED model outweigh any "bugs" that might be introduced during implementation.