ABSTRACT

A model for protocol offload is used to estimate the CPU utilization before and after TCP/IP protocol extraction. The model presented here is based on queuing theory and the Extensive Message Oriented (EMO) model. A modified version of Apache 2.2 Web server with raw sockets and a PC acting as the offload engine are used to implement the offload engine. Benchmarks will be run on our implementation to estimate the impact of protocol offload extraction and to refine our analytical model.

PROJECT DESCRIPTION

Protocol offload is the process of relieving the central processing unit (CPU) of a computer system, in whole or in part, of the processing of the end-to-end communication. Protocol processing using offload, is done in an entity outside main CPU called an Offload Engine (OE). When a node wants to transmit data from one place to another, the data transmitted is received by a network interface card (NIC), which passes every packet directly to the operating system (OS) running at the CPU for further processing. At the CPU, the OS runs the communication protocols software to retrieve the data contained within each packet. Also, in the CPU, the Network Application (NA) that handles the data is processed.

A different approach is used when offloading the protocol processing. The NIC passes the packets to the OE, which is entitled to process the communication protocols. In Figure 1 this OE is represented as a TCP Offload Engine (TOE). After each packet has been processed the data contained inside each packet is passed to the OS, then, the CPU processes the application that uses the data extracted. This approach, in some cases [6], maximizes the time applications spends running on the CPU. This is done since the CPU can be processing the data in the NA, while at the same time, the TOE process the incoming packets containing the next piece of data that the NA needs to continue using the CPU.

THE MODEL

Our model uses queuing theory. The network of queues is presented in Figure 2. It is assumed that the queue that process the packets (OE) it is in equilibrium and converges into a steady state. Our model can be adapted if the arrival process and service time has a closed form.

On a non-offload system the only queue that exists is the one represented by the CPU. This process the protocol and the NA.

The performance measure for our approach is the utilization of the CPU.

EXPERIMENTAL SETUP

A real implementation of an offload engine is underway using the test setup presented in Figure 3. The OS of the Master and the Slave server is Linux 2.6 (FC 6). The Master Server runs a modified version of Apache 2.2. This "hacked" version uses raw sockets to pass data and control messages to the Slave Server through an exclusive connection. This connection is via a 10 Gigabit per second Ethernet connection. This bypasses the TCP/IP stack between the servers. The Slave Server acts as a TCP/IP Offload Engine (TOE) [5,7,8].

FUTURE WORK

The implementation has been completed and tested. The user does not know if it is using a non-offload or offload approach when accessing the Web server. However, no benchmark has been run by the time this poster was prepared. The file generator of SpecWeb96 and the SURGE workload are under study for further testing. No data has been obtain yet for further analysis. The implementation needs a slight modification for avoiding interrupt pressure [2] and maximizing the offload engine capabilities. Also, we are expecting a slight modification of the analytical model.

REFERENCES