The "Fast-network, slow-host" challenge

Wenji Wu, Phil DeMar, Liang Zhang
Fermilab

Single-core CPU performance began to plateau in the mid-2000s, in large part due to Dennard scaling issues. In response, the semiconductor industry introduced a new generation of advanced computer architectures that exploited parallelism to continue Moore's Law-like growth in processing power. Among this new generation of processing technologies are multicore, manycore, and hybrid mixtures of the two technologies. These technologies are now commonly found in a wide spectrum of computer hardware, ranging from the large supercomputers in the DOE Office of Science's computing facilities, the data servers in data centers, and even mobile devices. Multicore and manycore have become the norm for high-performance computing.

Network speeds have continued to grow. Within the data center, server performance growth is driving 10GE connections for host systems, which in turn pressure data center backbone networks, paving the way for 100GE network links. 40GE host connections loom on the horizon. NICs supporting 40GE are now available in the market place, with 100GE NICs predicted to be available ~2018.

Because networks are getting faster and CPU cores are not, it is increasingly difficult for a single core to keep up with the high-speed link rates. Multi-queue NIC and multicore technologies are widely used to address the speed mismatch between CPU cores and high-speed networks. At data reception, a multi-queue NIC is logically partitioned into multiple queues, with each queue tied to a distinct core of a multicore system. Packets are distributed across the queues using a hardware-based traffic-steering mechanism. Each core handles a portion of the overall traffic. A basic assumption associated with this approach is that the hardware-based balancing mechanism is capable of evenly distributing the incoming traffic among cores. Typically, a NIC's traffic-steering mechanism distributes packets to cores based on a per-flow policy that assigns packets of the same flow to the same core. Such a traffic-steering mechanism can significantly improve network-processing performance. However, because network traffic is bursty by nature, this method of traffic steering can lead to a load imbalance condition in which certain cores become overloaded while others remain idle. In a worst-case scenario, a single core would be flooded with all the network traffic at wire speed (at least in short time scales). Such flooding would lead to buffer overflow, and result in packet loss. The so-called "fast-network, slow-host" problem would occur. In high-speed networks, packet drops would bring serious performance problems, especially for TCP.

The "fast-network, slow-host" problem is fundamental and inevitable when 40/100GE NICs run on multicore systems. If not properly addressed, serious performance bottlenecks within the end systems (i.e. hosts) will almost certainly ensure. The DOE is working toward deploying terabit networks in support of distributed extreme-scale data movement. Existing backbone networks are now being upgraded with ultra-scalable 100-gigabit technologies. Terabit networks need 40GE/100GE-capable end systems to efficiently move data on and off the network.

To address the "fast-network, slow-host" problem, we have identified a list of open questions that need to be answered:

(1) How to reduce traffic burstiness in high-speed networks? Should the traffic burstiness be addressed at layer 2, layer 3, or layer 4? Or, should we consider a cross-layer and holistic approach?

(2) How to improve end systems' capability in handling bursty network traffic? We believe that advanced NIC steering mechanisms, and OS offloading mechanisms should be designed to distribute incoming traffic across cores to avoid overloading particular core(s).

(3) Existing TCP implementations have been optimized for in-sequence packet processing. They leverage the concept of a fast path to maximize the performance of TCP packet processing. However, this fast path concept may not fit the multicore architecture well because it does not facilitate intra-flow packet processing across cores in parallel. How should we remedy this deficiency? Can we design a *multi-core-friendly TCP* or *a new transport layer mechanism* that facilitates both intra-flow and inter-flow parallelism on packet processing on multicore systems?