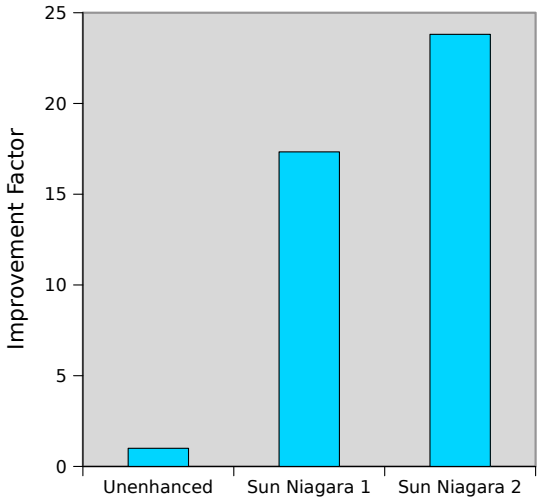
Parallel Snort: Or how to get 24× the throughput by using all your CPUs

www.world45.com

- Effectively parallelize Snort intrusion detection software.
- Up to 24× the performance of plain Snort on the same hardware.
- Takes full advantage of Sun Niagara processors and Logical Domains technology.
- All software: no FPGAs or specialised cards needed.

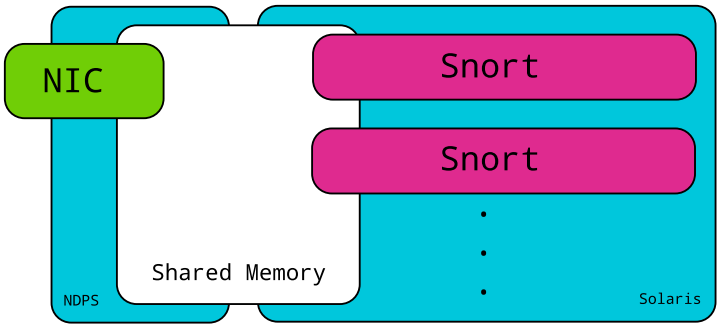
Snort (www.snort.org) is an open-source intrusion detection package that is very popular as a benchmark for network processing hardware. It is flexible and widely available, but has one fatal flaw: it is a program designed for the single-CPU world. This is a world we're fast leaving behind. 32 and 64 CPU machines are available today. With plain Snort 2.x, you can have, at most, one instance of Snort per network interface. On most hardware Snort isn't remotely fast enough to handle a full 1 Gb^{-1} interface, let alone make any headway into a 10 Gb^{-1} network. By using multiple CPUs World45 can meet this demand.

We use Sun Niagara processors which currently offer the maximum number of CPUs on one die. These have eight cores with 4 or 8 hardware threads each: effectively 32 or 64 CPUs (32 for the Niagara 1, 64 for the Niagara 2). By running multiple instances of Snort and using our packet provider to speed up I/O and to steer packet streams we can achieve the following results:



Since Snort 2.x is inherently single-threaded we have taken the approach of running multiple Snort instances in parallel and having a dedicated I/O provider with direct access to the hardware and a shared-memory interface to the Snort instances. The packet source resides inside a Sun Logical Domain - effectively bare Sparc hardware - without an operating system, but with direct access to the network hardware. The source can direct the NIC to deliver packets to a set of buffers in inter-domain shared

memory, and then tell the Snort instances where the packets are - eliminating all packet copying overhead and interrupt overhead. This is the fastest way to provide I/O.



The packet source also has to do processing to make sure that each TCP stream stays with a single Snort instance and that the load is evenly balanced between instances. The code to handle the shared memory and the packet queues is wrapped in a library that is interface-compatible with the pcap library, eliminating any need for modification of the main Snort code.

We conducted the tests with emulated I/O, detaching some cores from the domain where the Snort processes reside (running Solaris) to allow for this. Assuming that we can meet the I/O requirements, a Niagara 2.2 system can provide a factor of 24 improvement in throughput compared to a single Snort instance on the same hardware.

We considered multi-threading Snort internally. Despite its single-threaded design and over-use of global state there is potential within the current design (Snort 3, a complete redesign, will also be inherently multi-threaded). Our estimates suggest a factor of between 3 and 9 is possible with this approach, but needing the resources of 10 or more CPUs. The high end of this range can only be met by improved I/O, not using the OS-limited pcap library.