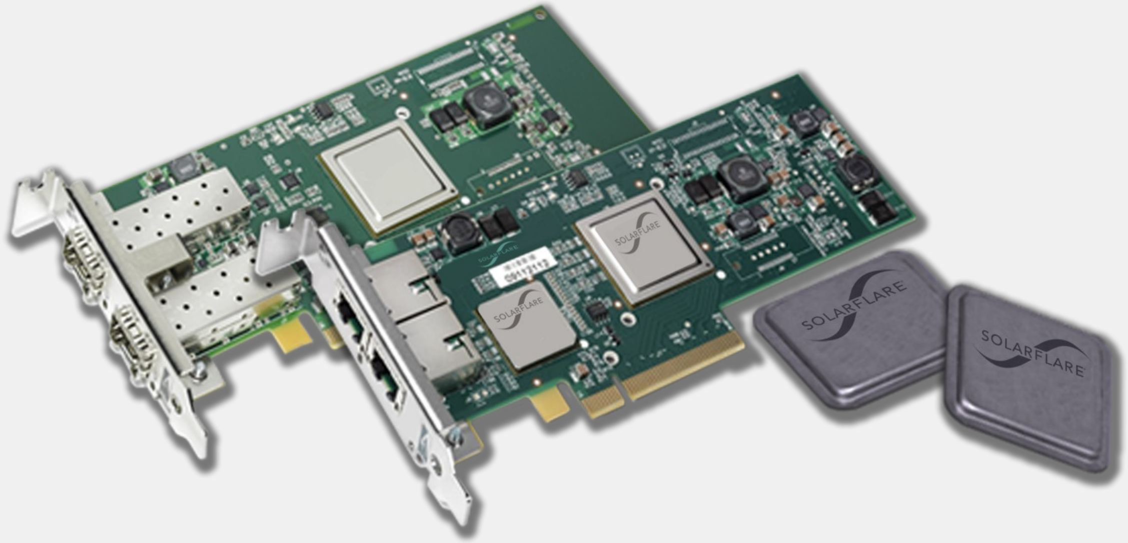

Managing multiple queues: affinity and other issues

Ben Hutchings - netconf 2011

Naive affinity settings

- Multiqueue driver allocates an IRQ to handle completions from each queue set (usually 1 RX and 1 TX)
- **CPU selection for socket user, RX packet processing and TX completions are independent:**
  - RX queue selection based on hardware flow hash
  - TX queue selection based on different software flow hash.
- IRQ affinity initially unset. May be spread out by irqbalanced, but without any understanding of multiqueue devices. **Effective scaling requires manual configuration.**
- Descriptor and event rings on NUMA node near device. **Reading rings can require inter-node traffic.**
- Software structures mostly allocated on NUMA node where probe was scheduled. **Do you feel lucky?**
- RX buffers usually allocated in same context as completions for the queue, on current NUMA node. This just works.

Later refinements to affinity

- Sockets have affinity based on scheduling of user thread; RX packet processing scheduled accordingly (RFS)
- RX flow steering puts completions and buffers near socket user (accelerated RFS; RX NFC/n-tuple)
- TX flow hashing can be overridden based on CPU (XPS)
- Driver can provide IRQ affinity hints which irqbalance will follow; probably not a good idea
- TX queue NUMA affinity inferred from XPS settings and used for qdisc allocation
- **Mostly dependent on manual configuration; disabled by default**

Simplifying affinity settings

Should be a simple way for administrator to set *all* affinities, preferably working with existing userland tools.

1. Userland sets IRQ affinities and queues follow
   - Accelerated RFS and proposed 'automatic XPS' infer queue affinity like this
   - Queue structures must not be reallocated automatically if device is up, as it will interrupt traffic. But many/most drivers free IRQs when device is down!
2. Userland sets queue affinities and IRQs follow
   - Networking core and ideally userland will need to be aware of queue-IRQ mapping
   - IRQ subsystem does not allow kernel users to set affinity. But we can set affinity hints.

RX queue selection

- Default behaviour of most drivers:
  - 1 IRQ and queue set per CPU (thread), within limits of device and IRQ controllers
  - RX flow hash spreads flows across all queues; therefore spreads packet processing across all CPUs with IRQ directed at them
  - Assuming IRQ affinity is spread out properly, this takes a little time from every CPU which is fair but can be wasteful
- Where steering is available, maybe better to limit flow hash to subset of queues?
  - Most devices have an indirection table which can be used for this (required by MS RSS spec)
  - RT users don't want random flows interrupting RT threads
  - RFS does a better job of assigning RX work to CPUs
  - All queues still available to accelerated RFS and RX NFC

Scaling policy

What could an administrator specify in a system policy for network device scaling?

- May limit number of IRQs and queue sets per device (absolute or *n* per package/core/thread)
- May limit number of RX queues to hash across

Networking core would calculate the actual numbers for drivers.

If networking core sets IRQ affinity hints:

- Specifying scale based on topology results in hints based on topology
- May specify global affinity mask for RX flow hashing; all hints would be subsets of this