Link Aggregation: A Server Perspective

Shimon Muller

Sun Microsystems, Inc.

May 2007

Supporters

- **Howard Frazier, Broadcom Corp.**

Outline

- The Good, The Bad & The Ugly
- LAG and Network Virtualization
- Networking and Multi-Threading
- Summary

The Good

- **Potential for linear throughput scaling**
  - Assumptions:
    - Total throughput is an aggregation of multiple “conversations” (flows)
    - The “conversations” are uniformly distributed across the physical links
    - Packet ordering must be maintained at all times
- **Works well for applications where a large number of network flows is a given**
  - Web Tier servers
    - Thousands/Millions of flows
  - Statistical multiplexing works for almost any traffic distribution algorithm
  - No flow dominates the bandwidth

The Bad

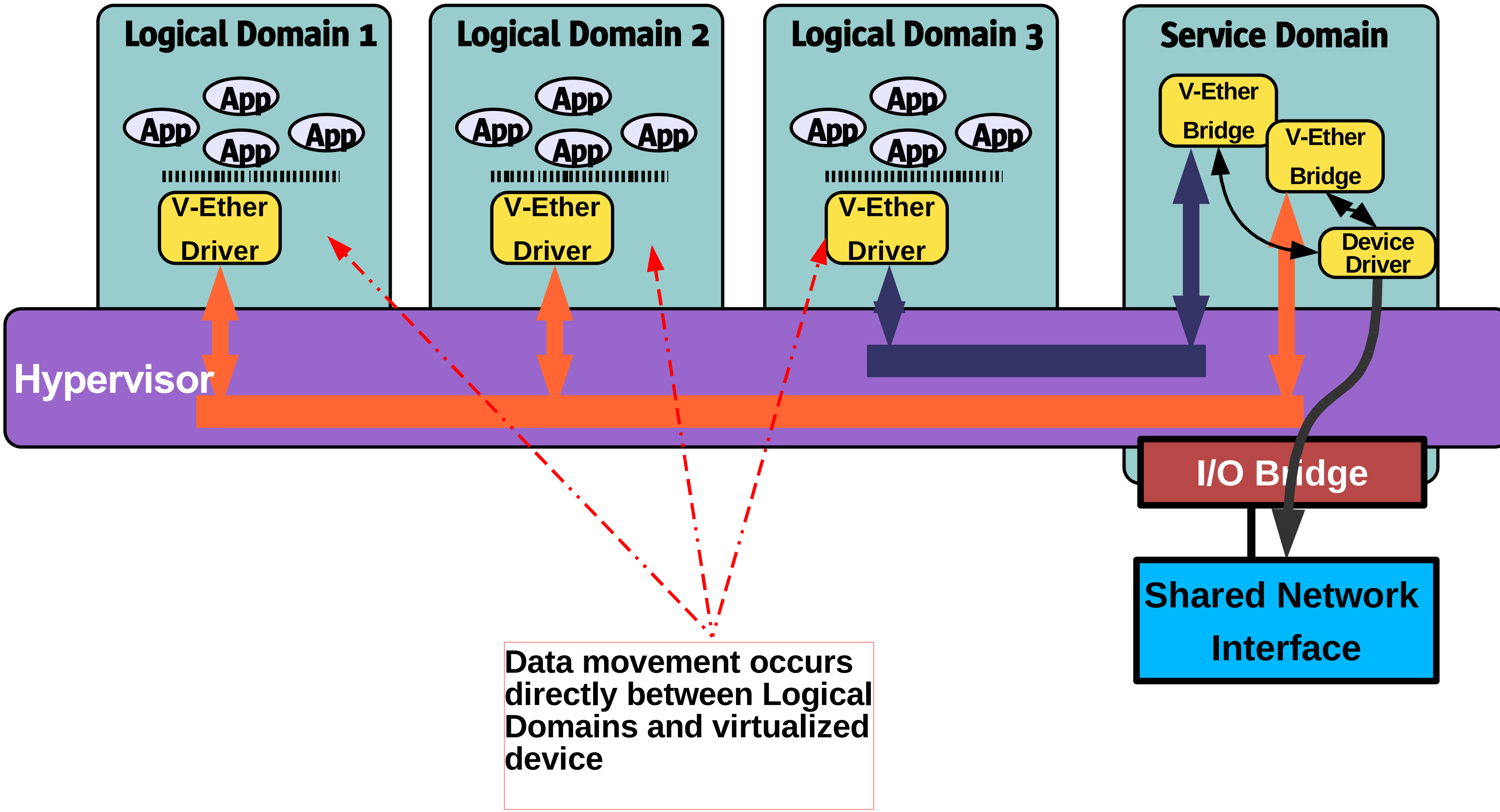
- **For some applications linear scaling is not a given**
  - Back-end Tier servers: Data Warehousing, Databases, OLTP, etc.
    - Dozens of connections at most
    - Can't assume statistical multiplexing
  - Need to dynamically manage the flow spreading over the LAG
    - Move flows around ---too complicated
- **Single flow throughput limited to the speed of a single phys. link**
  - Directly affects the performance of some Application Tier servers
  - Bulk data transfers: file servers, backup servers, etc.
- **The LAG distributor can have a performance impact**
  - On the host, typically implemented in the driver or just above it
  - Requires packet inspection
    - The deeper, the better spreading, but implies higher overhead
    - Duplicates protocol stack processing
- **Encrypted traffic...**
- **Layer violations...**

The Ugly

- **Latency penalty**
  - > Dominates the performance of transactional applications
    - > Typically a request-response exchange, followed by a bulk data transfer
    - > Measured in single-digit microseconds
  - > Round-trip latency is directly proportional to the speed of the physical link
    - > Common scenario: a small packet (request/response) stuck after a large packet (bulk transfer)
    - > A typical LAG distributor will map all the packets to the same physical link
    - > Time to send a packet: 1.23usec for a 4x10Gb LAG vs. 0.31usec for a 40Gb link
  - > LAG distributor serialization point adds latency
    - > Packet inspection
    - > Mutex lock contention

- **The LAG distributor creates a serialization point for Tx traffic**
  - > Breaks the parallelism paradigm for multi-core/multi-threaded computing
  - > Breaks the network virtualization story
  - > Interferes with efficient network b/w provisioning, capacity planning and QoS
    - > Choice of suboptimal traffic spreading vs. sophisticated h/w

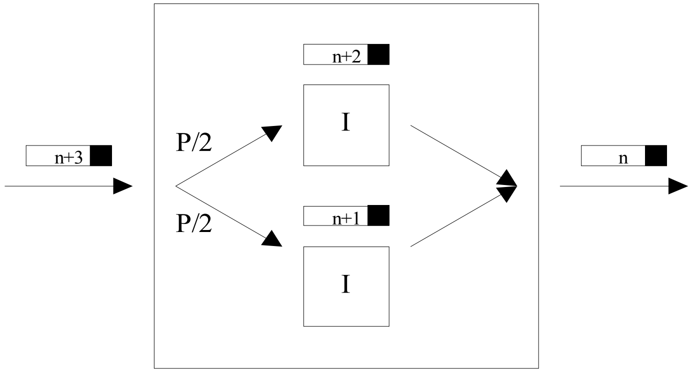
Network Interface Virtualization

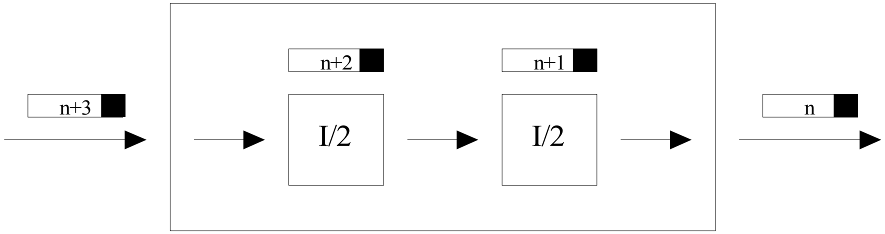


LAG and Network Virtualization

- **Network virtualization goals**
  - Pool all the networking resources in a system
  - Dynamically provision network resources to applications in compute domains with fine granularity and QoS
  - Enforce isolation between the compute domains
- **Network virtualization usage models**
  - Today network virtualization is done using the proxy model
    - All network traffic goes through the Service Domain
    - Creates a performance bottleneck
    - Breaks parallelism for network processing
  - Next generation of n/w virtualization will provide a direct path to shared NIC
- **The role of LAG**
  - Efficient LAG distribution algorithms require a complete view of all the network flows in the system
    - Implies the use of the proxy model
    - Doing LAG in Guest Domains will be suboptimal due to a limited number of flows
  - Complicates bandwidth provisioning and QoS
    - May need to split the bandwidth across multiple physical links

Networking in a Thread-Rich System





- An arbitrary combination of parallel and pipeline semantics
  - Can assign threads to do very specific chores with minimal latency
    - Use parallelism to improve latency and throughput
    - Use pipelining to improve throughput

LAG and Multi-Threading

- **New techniques for optimizing server network performance**
  - Typically will be a combination of parallelism and pipelining
  - Server performance can be optimized without re-writing applications
    - Use the threads to distribute the work intelligently
- **Multi-threading does not necessarily imply more n/w flows**
  - Nor is there a need to assume that for a faster pipe
    - No need to rely on statistics
    - Can use the threads to speed up the throughput of a single connection
  - It doesn't take too many network connections to saturate a 10Gb pipe today

Summary

LAG is a good thing...

...but not as good as a faster pipe!