Network Security with High Performance Storage for Big Data & HPC Applications

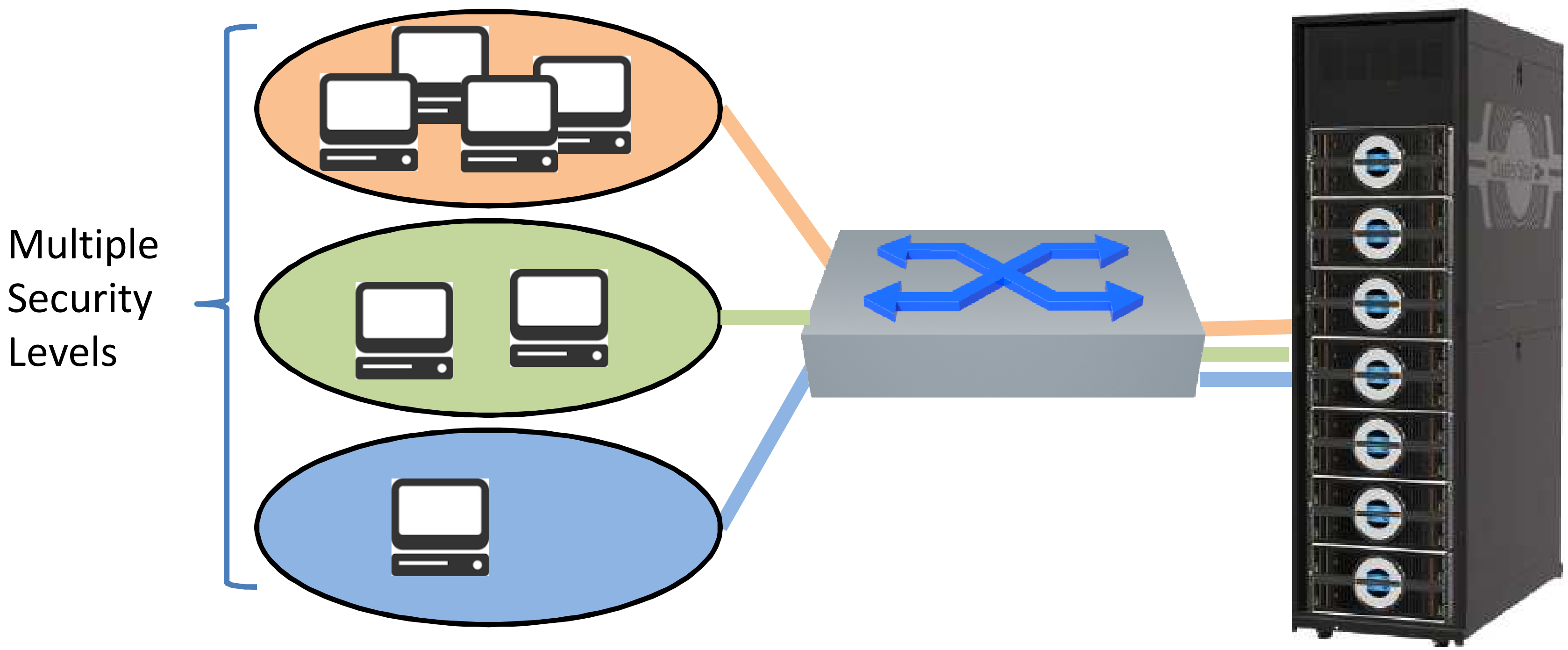
Kevin Deierling

Mellanox Technologies

Agenda

- Data Center Evolution
- Virtualization, MLS, Multi-Host
- User Space I/O, Virtualization, & RDMA
- SELinux Networking
- Securing RDMA
  - InfiniBand & RoCE in SELinux environment

Massive, High Performance MLS Storage



- Scalable storage for HPC & Big Data
- High Performance Lustre w (RDMA)
- Secure, integrated, scale out parallel file system
- Requires holistic security across OS, Network & Storage

Seagate
ClusterStor

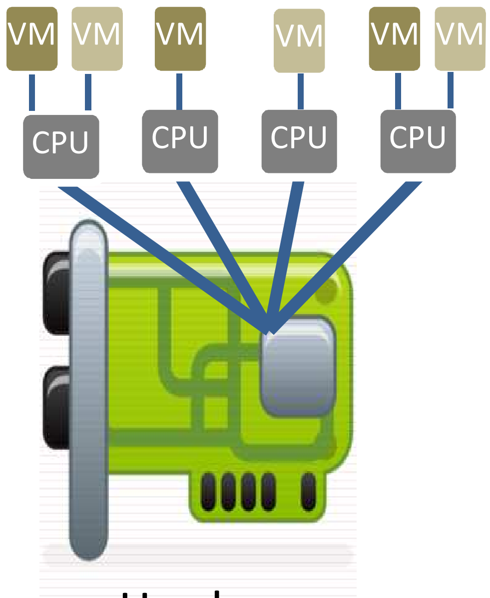
Secure
Storage
Appliance

Mellanox
TECHNOLOGIES

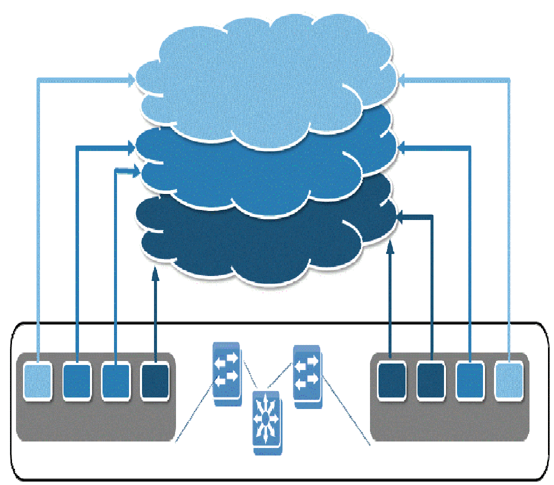
Virtualization & Security

- Virtualization, MLS, & user space I/O creates new security concerns
  - VM Hypervisors, containers
  - I/O virtualization: Needed for performance
- Hardware offload actually helps security
  - Embedded offloads
    - Hardware monitors & enforces policies set by secure SELinux processes
  - Secure user space I/O needs
    - Secure hardware configuration to implement policy
    - Hardware memory translation and protection enforcement!

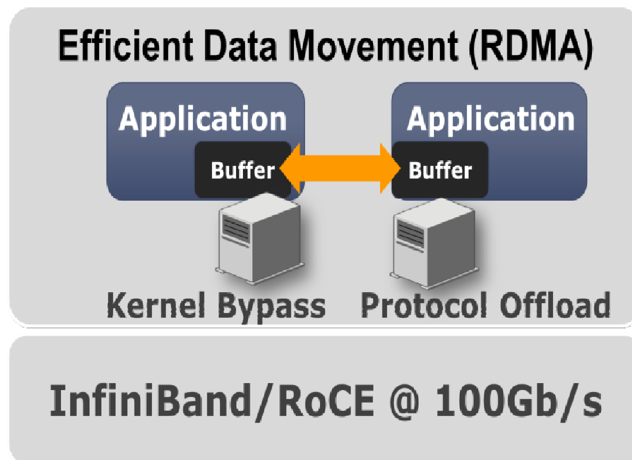
Efficient Data Movement



Hardware
Flow Steering



Partitions for
Multi-Level Security



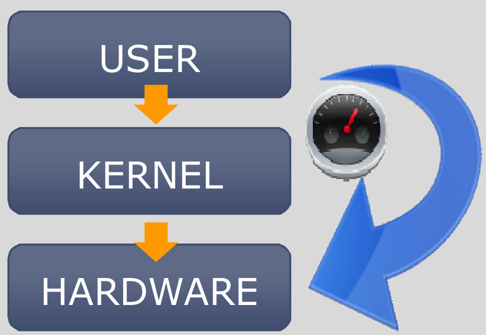
Efficient Data Movement
With RDMA

- Efficient Data Movement
  - Advanced Flow Steering Engine
  - Virtual network acceleration (VXLAN, NVGRE, GENEVE)
  - RDMA – Efficient Data Exchange - Low Latency, Low CPU Overhead

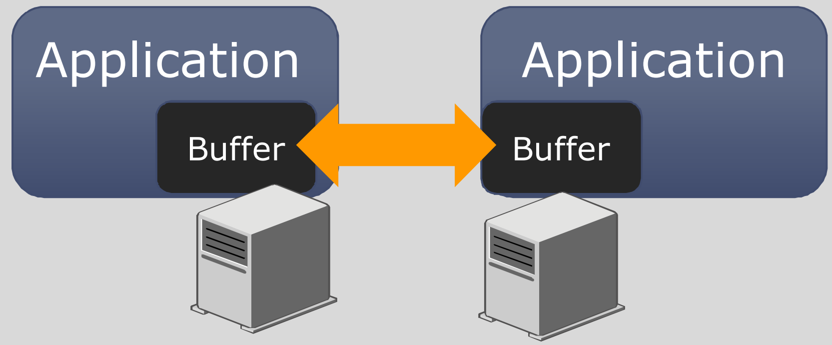
Mellanox TECHNOLOGIES

RDMA: Critical for Performance

ZERO Copy



Remote Data Transfer



Kernel Bypass

Protocol Offload

Low Latency, High Performance Data Transfers

InfiniBand - 100Gb/s

RoCE* – 100Gb/s

* RDMA over Converged Ethernet

User space I/O needs hardware memory protection!

Mellanox TECHNOLOGIES

Introduction

- SELinux is a Mandatory Access Control (MAC) scheme for Linux
  - Central policy is loaded upfront into the kernel
  - Applications cannot override or modify this policy
- Benefits
  - Differentiate a user from the applications that the user runs
  - Restrict application access only to what is required to perform its task
  - Allow granular policy segregation
  - Example
    - Run 2 instances of a Web Server: “top-secret” and “standard”
    - Each server can only
      - Receive traffic from specific network interfaces
      - Open sockets on specific ports
      - Serve files from specific directories
      - Communicate only with specific peer addresses
- Type enforcement is the main security mechanism used by SELinux

Mellanox TECHNOLOGIES

Type Enforcement (TE)

- Applies to all user-visible kernel entities
  - E.g., processes, files, IPC objects, sockets
- Each entity is associated with:
  - A security descriptor
    - Assigned upon creation, modified based on policy
  - A class and a set of operations
    - Stems from the type of object
    - E,g., a socket can send() and recv()
- TE defines what a <subject> can do on an <object> based on their security descriptors
  - Specified by a policy of access rules, enforced when accesses are made
- Security descriptors
  - Identify the user, role, type, and optionally security level+class of an object
  - Specified by a variable-length string: “user:role:type[:level]”
- Policy rules
  - Specify which source tag can access which target tag and for what operations
    - E.g., “allow source_t target_t:class { [op1] [op2] … }”
  - Typically, only the ‘type’ (a.k.a ‘Domain’) portion of the tag is mentioned

Fundamental RDMA SELinux Support

- RDMA network security shall be based on partitioning
  - Host kernels control the association of P_Key values with security descriptors

- Object labeling
  - Each QP shall be associated with a security descriptor
    - Inherited by the creating process in the absence of a specific policy
  - Each RDMA_ID shall be associated with a security descriptor
    - Inherited by the creating process in the absence of a specific policy
  - P_Key value labeling
    - Associates a P_Key value with a security descriptor
    - System object descriptors are a good example (like network interfaces or nodes)
      - "system_u:object_r:rdma_partition_default_t"
      - "system_u:object_r:rdma_partition_topsecret_t"

Mellanox TECHNOLOGIES

Fundamental RDMA SELinux Support (cont.)

- Traffic labeling
  - Uses network labeling (labels are carried on the wire)
  - P_Key values are used as the network label
- Policies
  - Allow a QP or RDMA_ID to be associated with a P_Key value
  - Example: “allow hpc_default_t rdma_partition_default_t : rdma_partition { modify }”, where
    - ‘hpc_default_t’ is the QP / RDMA_ID domain (type) inherited from the creating process
    - ‘rdma_partition_default_t’ is a partition security descriptor domain
    - ‘rdma_partition’ indicates that the subject is of partition type
    - ‘modify’ indicates that the QP is allowed to modify to reference the corresponding partition tag
- Enforcement
  - QP partitioning is enforced at all times
    - Upon creation, a violation shall result in an immediate error
    - At runtime, any violation due to policy or P_Key value changes shall transition the QP into error
  - RDMA-ID
    - All ingress/egress CM MADs shall be checked according to the partition policy
    - Any violation shall result in an immediate packet drop

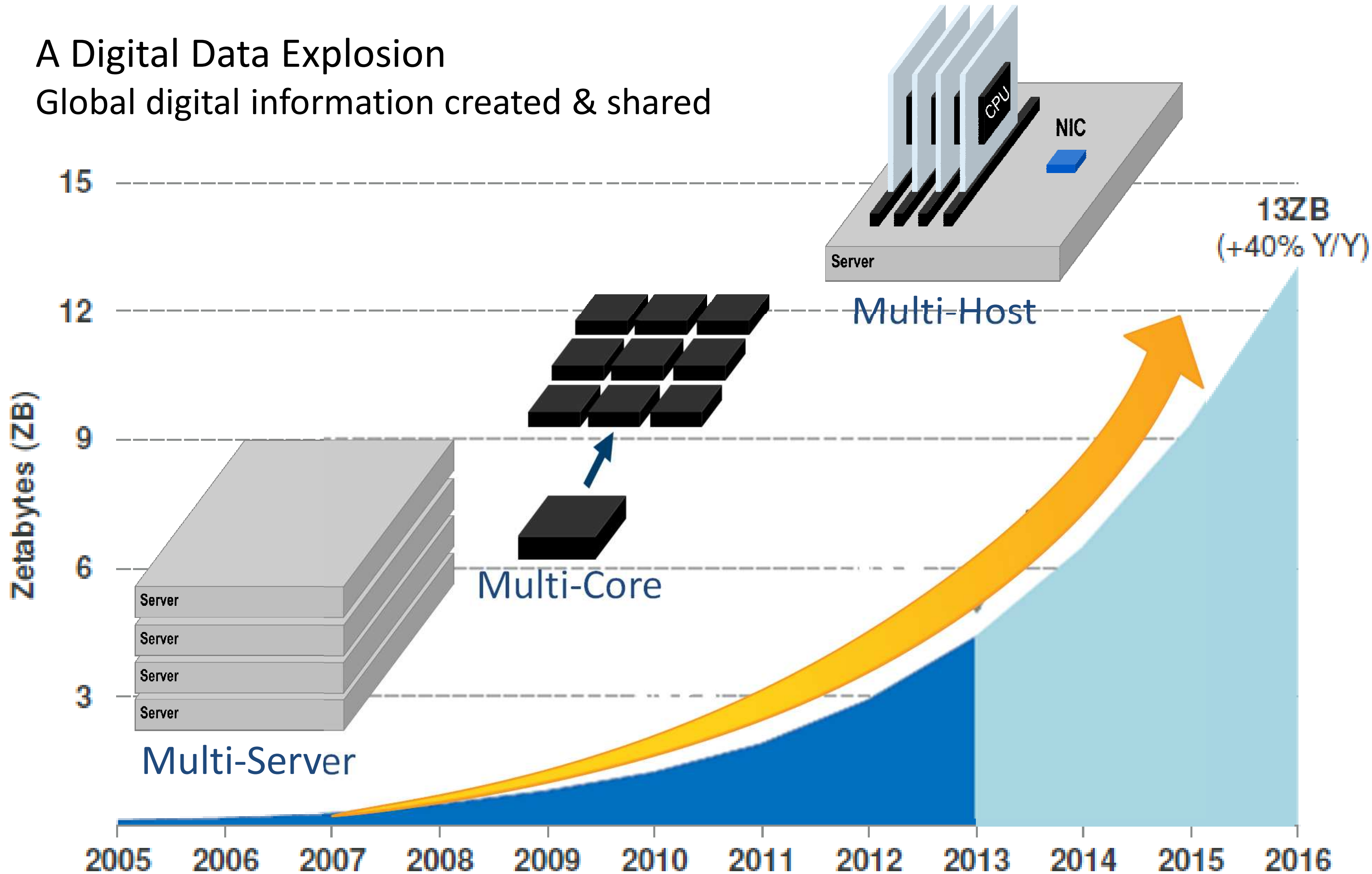
Thank You!

Questions

Data Center Evolution Over Time

A Digital Data Explosion

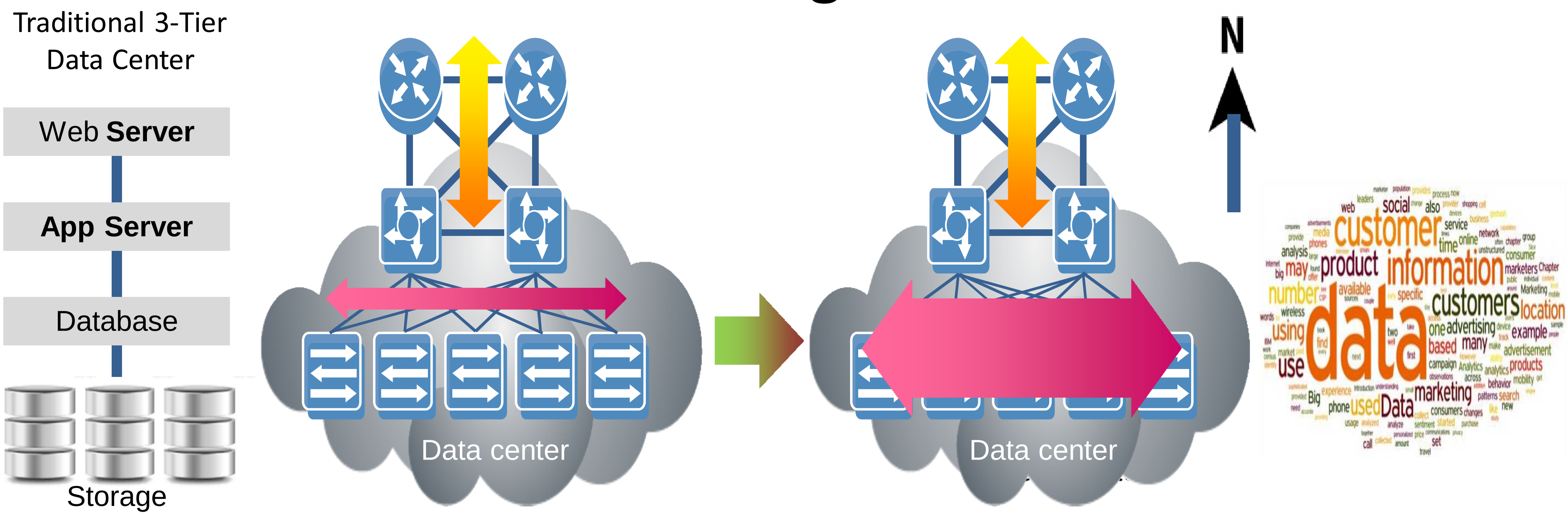
Global digital information created & shared



Source: KPCB, IDC

Mellanox TECHNOLOGIES

Changing Traffic Patterns Requires Data Center Change



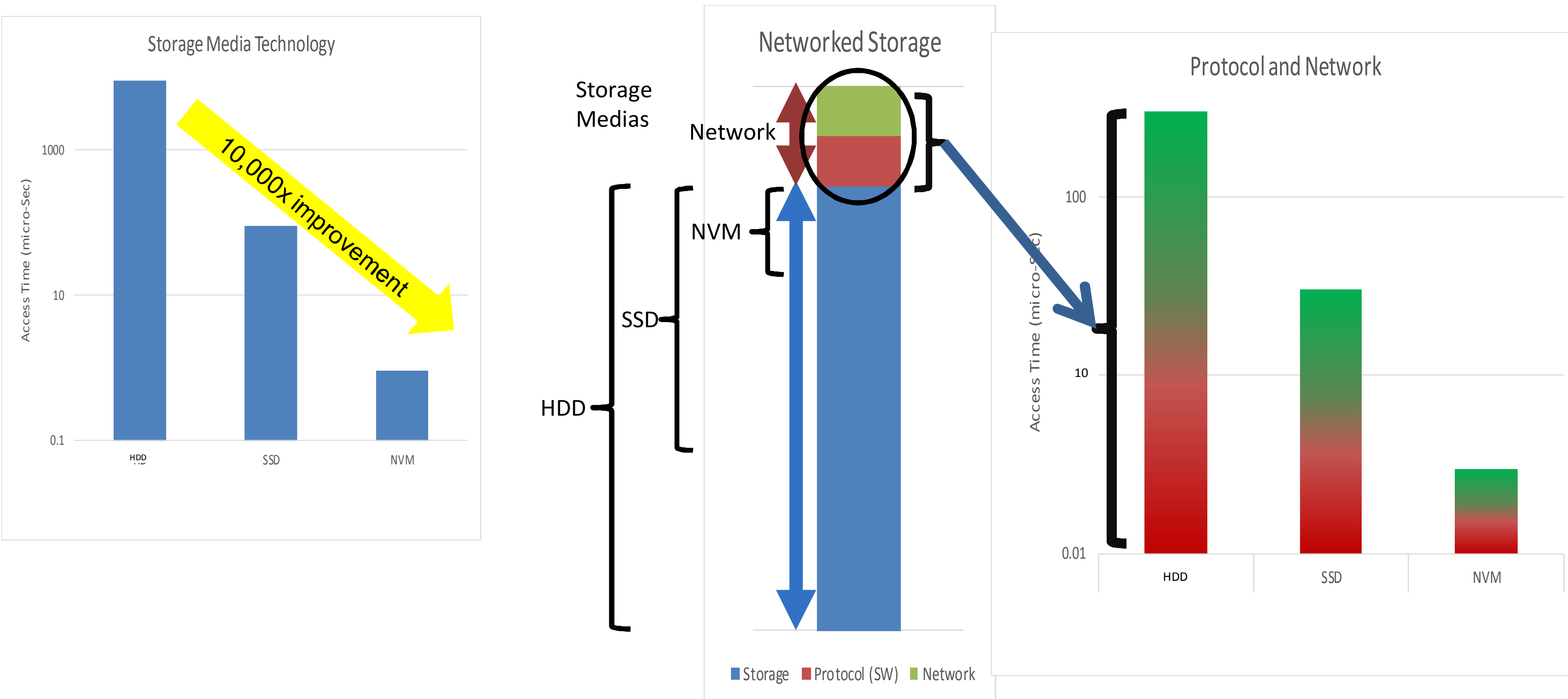
North-south traffic 80%

East-west traffic 70%

North-south traffic: Data forwarded between external users and internal servers. Typically data flows through the 3-tier architecture

East-west traffic: data forwarded between internal servers of the data center.

New Storage Media Creates Network Bottlenecks



Faster Networking, Protocol Offloads, & Bypass Required to Match NVM Performance

Mellanox TECHNOLOGIES

Containers

```
/* shocker: docker PoC VMM-container breakout (C) 2014 Sebastian Krahmer
 *
 * Demonstrates that any given docker image someone is asking
 * you to run in your docker setup can access ANY file on your host,
 * e.g. dumping hosts /etc/shadow or other sensitive info, compromising
 * security of the host and any other docker VM's on it.
 *
 * docker using container based VMM: Sebarate pid and net namespace,
 * stripped caps and RO bind mounts into container's /. However
 * as its only a bind-mount the fs struct from the task is shared
 * with the host which allows to open files by file handles
 * (open_by_handle_at()). As we thankfully have dac_override and
 * dac_read_search we can do this. The handle is usually a 64bit
 * string with 32bit inodenumber inside (tested with ext4).
 * Inode of / is always 2, so we have a starting point to walk
 * the FS path and brute force the remaining 32bit until we find the
 * desired file (It's probably easier, depending on the fhandle export
 * function used for the FS in question: it could be a parent inode# or
 * the inode generation which can be obtained via an ioctl).
 * [In practise the remaining 32bit are all 0 :]
 *
 * tested with docker 0.11 busybox demo image on a 3.11 kernel:
 *
 * docker run -i busybox sh
 *
 * seems to run any program inside VMM with UID 0 (some caps stripped);
```

Docker, Linux Containers (LXC), and security from Jérôme Petazzoni

Containers & Security: Oxymoron or Opportunity

- “Containers do not contain.”
  - Dan Walsh (Mr SELinux)
- Leak Threats
  - Filesystem, Namespace problems
- Fixes
  - Name space mapping to isolate UID
  - SELinux: containers security contexts

Docker, Linux Containers (LXC), and security from Jérôme Petazzoni

Mellanox TECHNOLOGIES

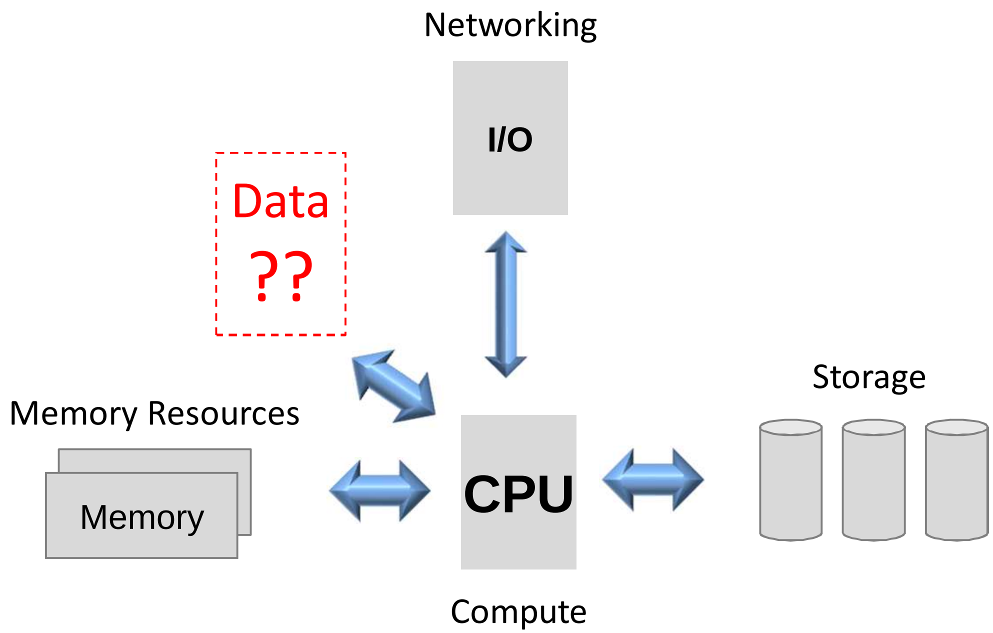
From Compute Centric to Data Centric Data Center (DCDC)

- Compute-centric architecture
  - CPU at the center with attached peripherals
  - Developed for transactional processing
    - Small, slow, fixed-format data
  - Data is an afterthought!
  - Not equipped for Big-Fast-Unstructured Data

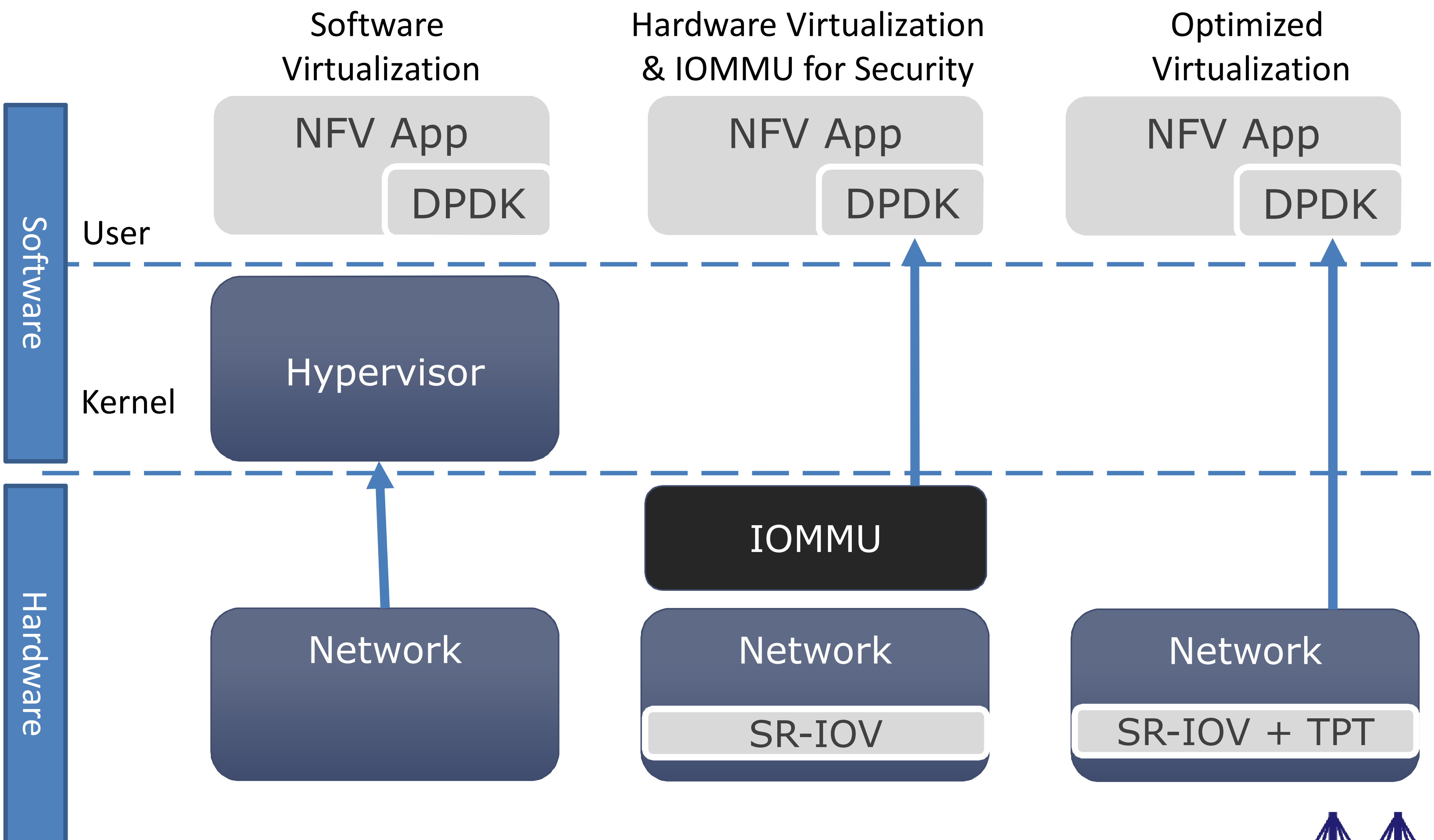
- Focus is on server-level optimization
  - Compute-centric optimization focus is the server
  - Secondary focus is the storage chassis

- A higher level view is huge advantage!
  - From compute to data centric architecture
  - Explicitly considers Big-Fast-Unstructured Data
  - Higher efficiency and better CapEx and OpEx

Compute Centric Center Architecture



Optimizing NFV & DPDK for Security



NFV: Network Function Virtualization
DPDK: Data Plane Development Kit

SR-IOV: Single Root I/O Virtualization
TPT: Translation Protection Table

Mellanox TECHNOLOGIES