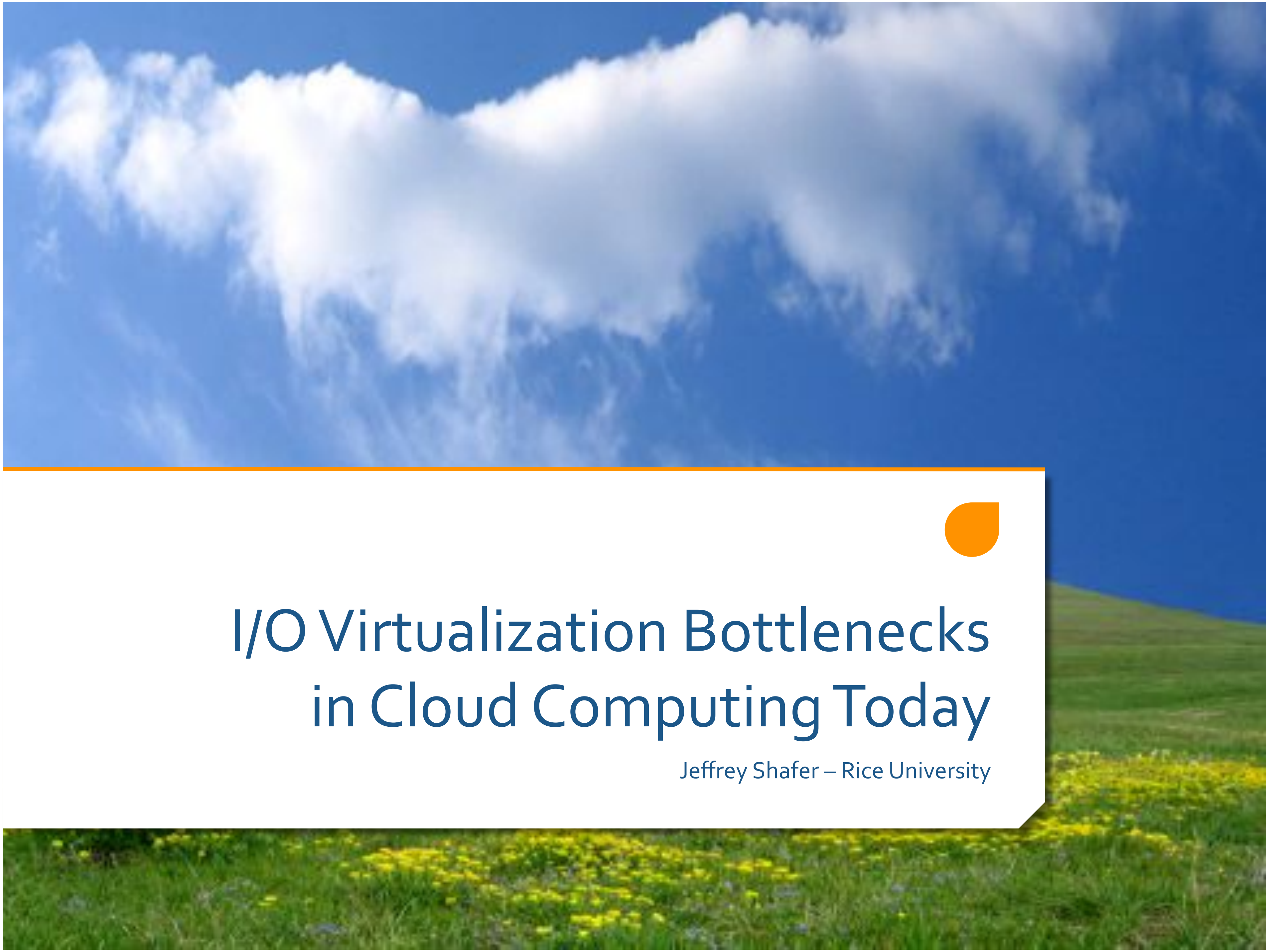


I/O Virtualization Bottlenecks in Cloud Computing Today

Jeffrey Shafer – Rice University

Cloud Computing

- Hot topic in popular media
- Builds upon established (trusted?) virtualization technology
  - Install applications not onto native machine, but into virtual machine images
  - Treat datacenter as a generic computing resource
    - Start / stop / migrate application images on demand
- Take the next step with **cloud computing**
  - Out-source the generic datacenter
    - Let someone else manage it
    - Pay only for what you use
  - Pioneered by Amazon Elastic Compute Cloud (EC2)
    - "Platform as a service" abstraction

Private Cloud Computing

- What if I'm not ready to trust the cloud?
  - Security concerns
    - Who has access to my data?
  - Performance / quality of service concerns
    - How many other customers are sharing the same server, network, or storage array?
  - Vendor lock-in
    - What if Amazon raises prices?
- Private cloud computing
  - Build your own cloud "behind the firewall"

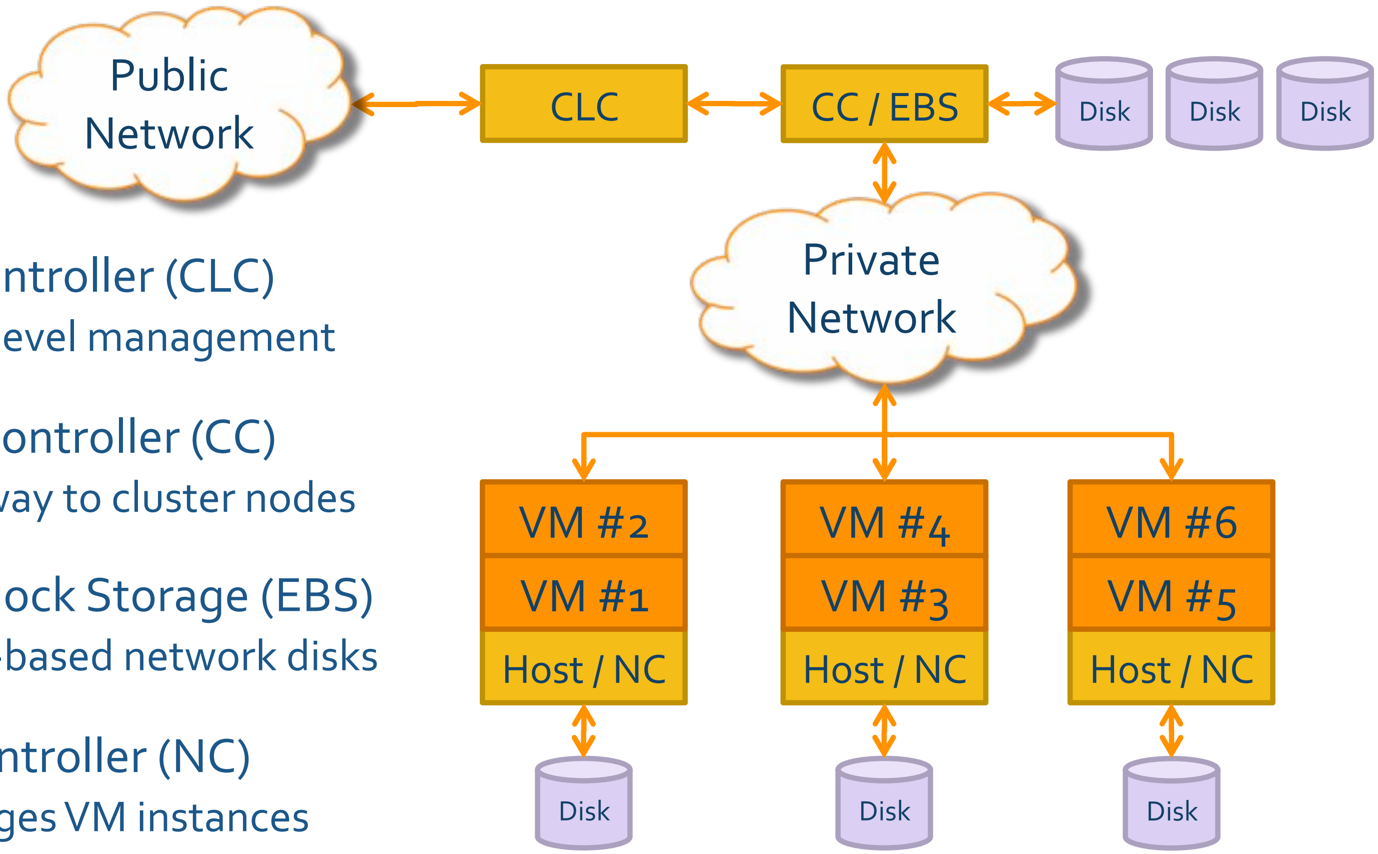
Eucalyptus

- Allows creation of private clouds
- Open-source cloud computing framework
  - Linux-centric: Many distributions, KVM or Xen virtualization
- API compatible with Amazon platform
  - Allows re-use of common administrative tools
  - Allows private clouds to burst to public clouds if desired
- Ubuntu Enterprise Cloud – New in version 9.10
  - Pre-packaged Eucalyptus installation
  - 30 minute "cloud-in-a-box"

Today's Talk – Data-Intensive Computing

- How well does this cloud computing platform run data-intensive applications?
  - Hadoop – open-source MapReduce framework written in Java
    - Convenient way to parallelize computation across a cluster
    - Target applications: web indexing, data mining, log file analysis, machine learning, scientific simulation, etc…
    - Commonly run in a cloud environment by those who can't afford a dedicated cluster (or don't need one full-time)
      - Equivalent to Amazon Elastic MapReduce product
- Summarize out-of-the-box performance and configuration options
- Discuss ways to increase performance

Eucalyptus Architecture *(Simplified)*



- Cloud Controller (CLC)
  - High-level management
- Cluster Controller (CC)
  - Gateway to cluster nodes
- Elastic Block Storage (EBS)
  - Block-based network disks
- Node Controller (NC)
  - Manages VM instances

Data-Intensive Computing Performance

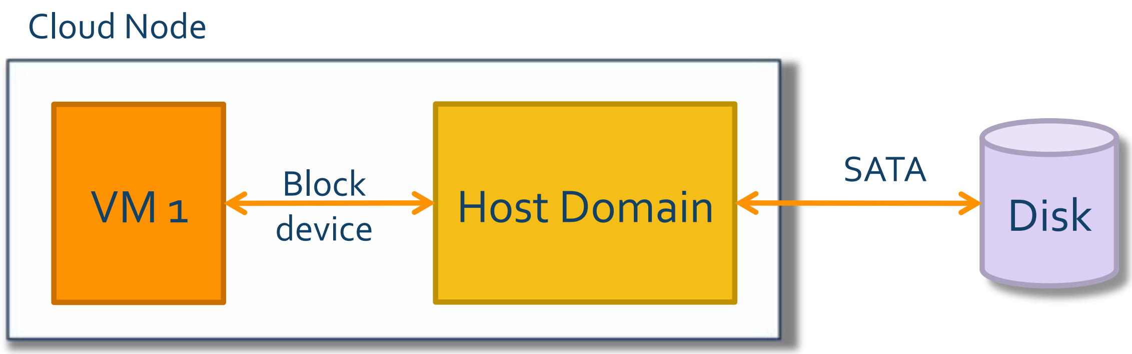
- Hadoop MapReduce framework
  - 10GB read/write tests (streaming sequential access)
  - Used local storage (disk attached to same node)
  - Measured execution time (seconds)

Environment	Write (s)	Read (s)
Non-Virtualized	113	116
Virtualized	6826	196

- Virtualized system not CPU limited (> 90% idle)
  - Storage bandwidth the bottleneck?

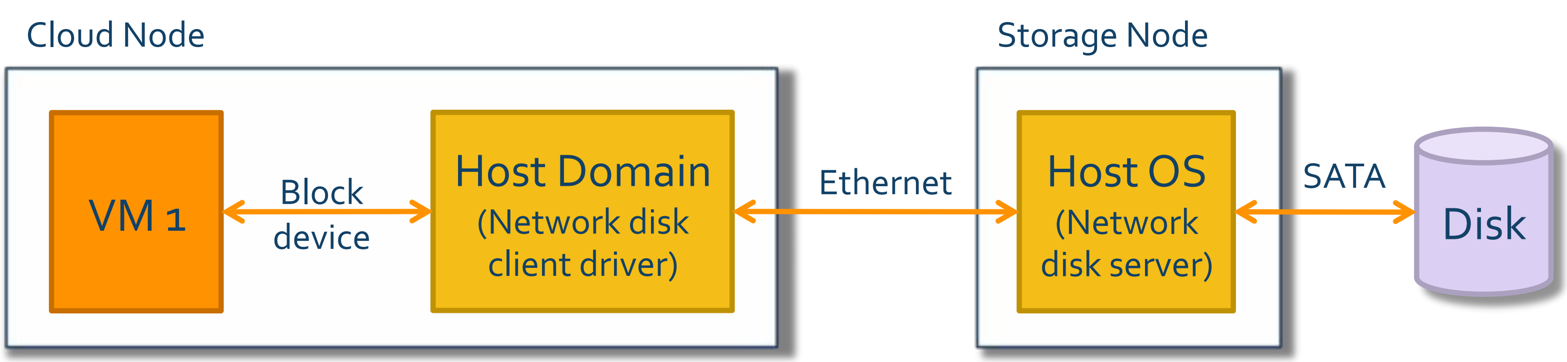
Eucalyptus Storage Options

- Choice #1: Local storage
  - File on local disk mapped into guest domain
  - Equivalent to Amazon local instance storage



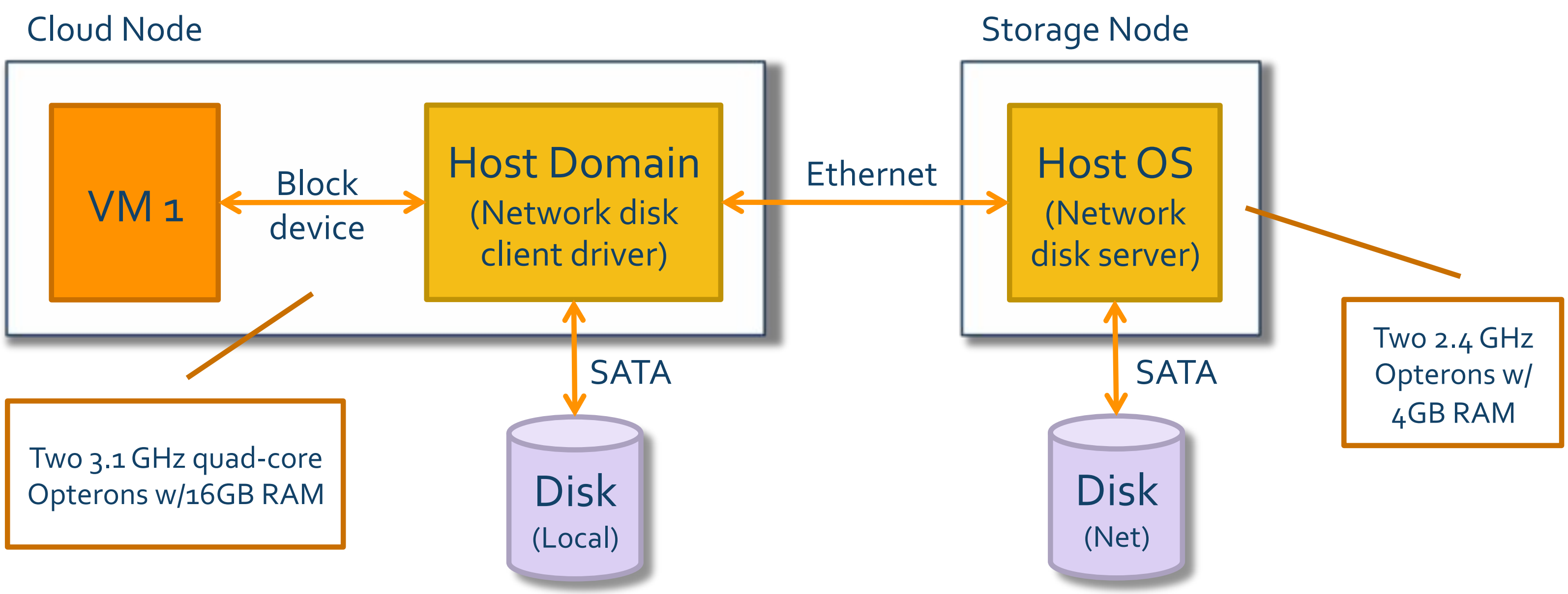
Eucalyptus Storage

- Choice #2: Network storage
  - File on remote disk (on storage server)
  - Network disk server exports file across the network
    - ATA over Ethernet protocol
    - Lightweight encapsulation of ATA requests, non-routable
  - Host domain runs device driver to access network storage
  - Abstraction: To clients, storage is still local
  - Equivalent to Amazon Elastic Block Storage (EBS)



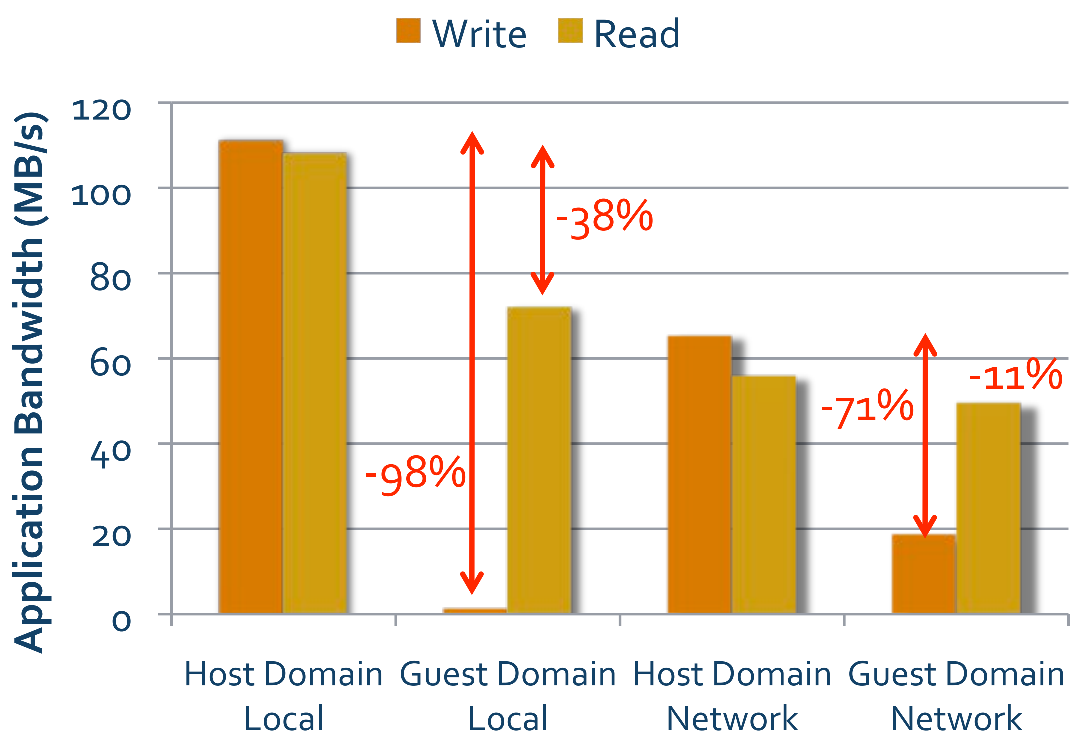
Experiment Setup

- Ubuntu Enterprise Cloud with KVM
- 500 GB Seagate SATA hard drives
- 1 guest per machine – Want to be disk-bound, not compute-bound
- 1 disk per guest – Data-intensive applications "share" poorly



Eucalyptus Storage Performance

- DD synthetic test
  - Same access pattern as Hadoop (sequential access, 64kB requests)
  - Minimal CPU overhead
- Non-virtualized host domain for comparison purposes



Eucalyptus Storage Performance

- The default configuration performs poorly
  - Have we already solved this problem?
  - Did Eucalyptus just choose some poor defaults?
- Expanded the test scope
  - Hypervisors: KVM, Xen
  - I/O virtualization:
    - Full virtualization: SCSI device
    - Para-virtualization: Virtio (for KVM), XVD (for Xen)
  - Sparse file, full file, and full disk backing options
- More data beyond application bandwidth
  - Disk request size, queue depth, and utilization
  - Measured in host domain to quantify disk *efficiency*

Write Bandwidth / Local Storage

VMM	Driver	Bandwidth (MB/s)	Request Size (kB)	Queue Size (Elements)	% Util Disk
None	*N/A*	*111*	*512*	*140*	*100%*
KVM (*)	SCSI / **sparse file**	1.3	15	0.9	90%
KVM	SCSI / **full file**	62.6	128	0.82	81%

(*) = Default configuration

- Causes of poor initial write performance
  - Sparse file backing / expansion overhead
  - Small (15kB) disk requests
- **Pre-allocating backing file** on disk increases bandwidth
- Tradeoff – Starting guests takes much longer

Write Bandwidth / Local Storage

VMM	Driver	Bandwidth (MB/s)	Request Size (kB)	Queue Size (Elements)	% Util Disk
None	*N/A*	*111*	*512*	*140*	*100%*
KVM	**SCSI** / full file	62.6	128	0.82	81%
KVM	**Virtio** / full file	87.0	490	42	100%

- **Para-virtualized drivers** increase disk efficiency in KVM and Xen (not shown)
- Large requests (> 350kB)
  - Page cache in guest domain and/or device driver aggregates multiple 64kB application requests
- Multiple outstanding requests (> 3.0)
  - Synchronous writes are committed to guest page cache and immediately return
  - Requests queued in OS and committed to disk in a batch
- Tradeoff - Requires guest OS support

Write Bandwidth / Local Storage

VMM	Driver	Bandwidth (MB/s)	Request Size (kB)	Queue Size (Elements)	% Util Disk
None	*N/A*	*111*	*512*	*140*	*100%*
KVM (*)	SCSI / sparse file	1.3	15	0.9	90%
KVM	SCSI / full file	62.6 ↓	128	0.82	81%
KVM	SCSI / **disk**	71.5	128	0.57	64%
KVM	Virtio / full file	87.0 ↓	490	42	100%
KVM	Virtio / **disk**	110	512	60	100%
Xen	SCSI / full file	58.4 ↓	498	142	100%
Xen	SCSI / **disk**	65.8	126	0.87	86%
Xen	XVD / **disk**	102	350	3.0	100%

(*) = Default configuration

Write Bandwidth / Local Storage

- Full disk backing improves performance further over file backing
  - Para-virtualized drivers (in KVM) **comes within 1%** of non-virtualized disk bandwidth
- Tradeoff in flexibility – only one guest domain per disk partition
  - Acceptable for data-intensive computing applications
    - Storage performance is critical
- General-purpose cloud computing applications can continue to use file backing
  - Storage performance less critical
  - Sharing hardware between multiple guests is necessary for economic reasons

Read Bandwidth / Local Storage

VMM	Driver	Bandwidth (MB/s)	Request Size (kB)	Queue Size (Elements)	% Util Disk
None	*N/A*	*108*	*256*	*0.94*	*96%*
KVM (*)	SCSI / sparse file	71.9	225	1.1	96%
KVM	SCSI / full file	71.4	241	0.64	64%
KVM	SCSI / disk	70.5	256	0.7	68%
KVM	Virtio / full file	75.9	256	0.7	69%
KVM	Virtio / disk	76.2	256	0.5	57%
Xen	SCSI / full file	83.1	121	1.6	99%
Xen	SCSI / disk	42.8	7	22.4	99%
Xen	XVD / disk	94.8	64	2.2	99%

(*) = Default configuration

Read Bandwidth / Local Storage

- Unable to reach peak disk read bandwidth
  - Best observed configuration (Xen / XVD) has a 12% performance gap in this best-case test
  - Average configuration has a 30% performance gap
- Disk access patterns show the problem. Either:
  - Small request sizes (< 7kB) – Disk is used inefficiently
  - Small queue depths (< 0.7 requests) – Disk sits idle waiting for requests
- Challenge with synchronous I/O – application issues a 64kB read request and then waits for the data
  - Guest OS page cache *may* pre-fetch amount of additional data

Read Bandwidth

- Asynchronous I/O a solution for data-intensive computing?
  - Application can post many large read requests simultaneously
- Challenges in Hadoop / Linux
  - Asynchronous I/O in Linux only works in conjunction with O_DIRECT mode (bypasses page cache)
  - Neither feature is natively supported in Java
    - But we only have to implement it once!

Network Storage

- Local storage suitable for scratch purposes only
  - Example: storing temporary map/reduce keys
  - Deleted when guests are stopped
- Network storage necessary for persistent data in cloud environment
- Performance in host domain:

DD Application	Bandwidth (MB/s)
Write	65.2
Read	55.8

Network Storage

- Network storage bandwidth limited by ATA over Ethernet protocol
  - Degrades raw disk bandwidth by 40%+ just reaching the host domain
  - Simple request/response design, just like native ATA
- Potential optimizations not used in default Eucalyptus
  - Jumbo Ethernet frames (increase payload size of each ATA request)
  - At server application
    - Aggregate adjacent I/O requests to improve disk efficiency
    - O_DIRECT decreases CPU overhead
- What happens to bandwidth in the virtualized domain?

Bandwidth / Network Storage

- Virtualization increases latency to reach host domain and network driver, and degrades write bandwidth further

Write

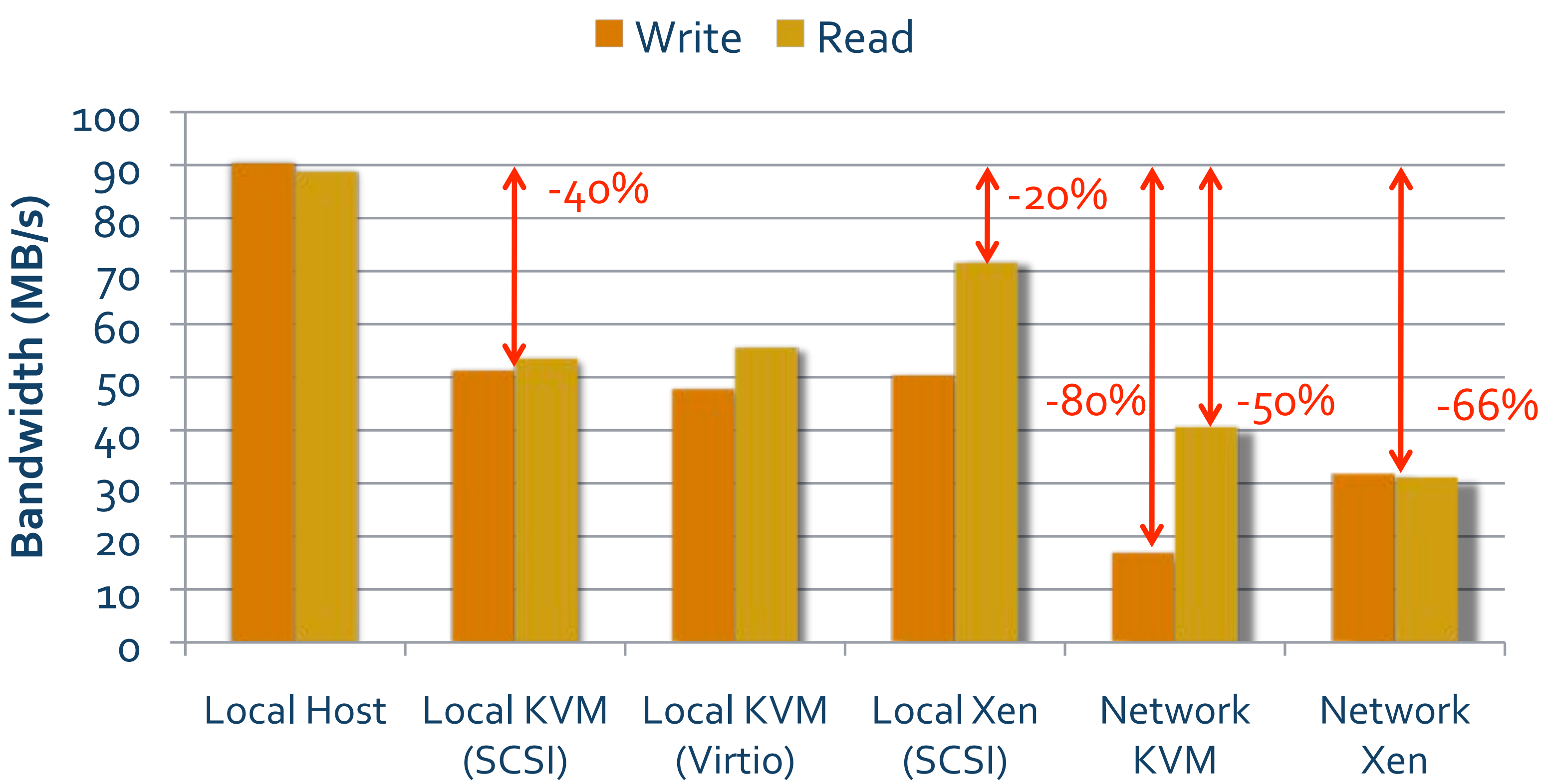
VMM	Driver	Bandwidth (MB/s)
None	*N/A*	*65.2*
KVM (*)	SCSI	19.9
KVM	Virtio	20.3
Xen	SCSI	26.5

(*) = Default configuration

Read

VMM	Driver	Bandwidth (MB/s)
None	*N/A*	*55.8*
KVM (*)	SCSI	49.5
KVM	Virtio	48.0
Xen	SCSI	51.4

Conclusions – Hadoop Summary



Conclusions

- Cloud computing framework degrades data-intensive computing applications significantly
- Configuration changes improve out-of-box performance while still maintaining API compatibility
  - Full backing files instead of sparse
  - Para-virtualized block I/O instead of fully-virtualized
- Future work needed to close performance gap
  - Improve network disk protocol implementation
  - Explore impact of asynchronous I/O on virtualized guest performance

Questions?

? ? ?