Towards Database Virtualization for Database as a Service

Aaron J. Elmore, Carlo Curino,
Divyakant Agrawal, Amr El Abbadi

[aelmore,agrawal,amr] @ cs.ucsb.edu
ccurino @ microsoft.com

CAVEAT:
Representative not exhaustive

Virtualization: An Overview

Fifty+ Years of Virtualization

Virtualized Processing

Multiplex a single physical processing unit among multiple processing tasks.

Virtualized Memory

Physical memory shared among multiple logical address spaces

Logical address spaces are not constrained by the size of the physical memory

Contiguity and isolation of each logical address space even though the physical allocation is fragmented and partial

Fifty+ Years of Virtualization

Virtualized I/O

A logical I/O unit that may correspond to a partition on a physical disk or to a multi-disk RAID volume exported via a networked storage array.

Virtualized Network

A virtual private network that provides isolation via encryption even when the real data transfer is on the shared public network infrastructure.

Virtualization Principles

Stated simply

The art of decoupling the logical from the physical.

Implementation

Introduce a level of indirection between the abstract and the concrete

Incurs performance overhead/penalty

Such indirection, however, has proven to be extremely powerful and versatile

Multi-programming, Virtual Memory, File systems, etc.

Virtualization Origins: 1960s

Multi-tasking/Timesharing systems

- Leo III Computer by J. Lyons & Co. from UK in 1961
- CTSS (Compatible Time Sharing System) USA in 1961
- Project MAC at MIT
- Later: IBM System/360, MULTICS, and numerous others …

Virtual Memory

- VM concept developed at TU-Berlin in 1956
- First VM implementation in ATLAS at University of Manchester in 1959
- Widely adopted in the commercial systems in the sixties: IBM, Burroughs, and numerous others …

The Notion of Virtual Machine

Multi-tasking and Virtual Memory Management made the notion of "Virtual Machine" a first-class concept in operating system design.

The goal was to execute multiple processing tasks on a single physical machine while giving the perception that each task has a dedicated Virtual Machine.

Individual processing tasks could not distinguish between the physical machine and the virtual machine except perhaps the speed of execution.

Virtualization in IBM Systems

In late 60s – IBM had system implementations that allowed multiple Operating Systems to co-exist on a single machine ➔ pre-cursor to modern-era virtualization.

However, this concept did not become a mainstream feature since it was not required in most installations and perhaps due to business needs.

➔ Virtualization, as we understand now, remained dormant for the next two decades.

Virtualization Renaissance?

In the late nineties, UNIX and WINDOWS based servers proliferated in the enterprise context.

Furthermore, multiple versions and releases of UNIX was common since applications and services were based on a specific version/release of an Operating System.

In the same vein, commoditization of Hardware platforms led to servers and machines with varying capabilities.

➔ significant complexity for system administration in the enterprise context

Enterprise Server Infrastructure Administration

Proliferation of Hardware:

Machines with different configurations acquired over different time-periods

Proliferation of Operating Systems:

Multiple UNIX/LINUX implementation from different vendors
Multiple versions/releases of OS implementations from the same vendors
Microsoft became a dominant player in the server market

Proliferation of Applications and Services:

Large number of vendors for specialized applications/services
Application/services tied to different generations of OSs

➔ Basically, led to system administration nightmare

System Administration Realities

Need to ensure that a server/machine is configured with the correct OS for a given application/service

Hardware failure necessitated a highly manual task of configuring the Application+OS on a new/replacement hardware platform.

In general, the new replacement HW platform may not be identical ➔ long service disruptions.

Highly under-utilized server infrastructure since services running on different OSs could not be consolidated.

➔ the old forgotten concept now became an absolute necessity

The Need

On-the-fly Service Migration

Service Consolidation from multiple machines to a single machine

Resource Isolation

What is Virtualization?

Virtualization defined:

Decoupling of an Operating System from the underlying Hardware.

Virtualization ⟷ Cloud Computing?

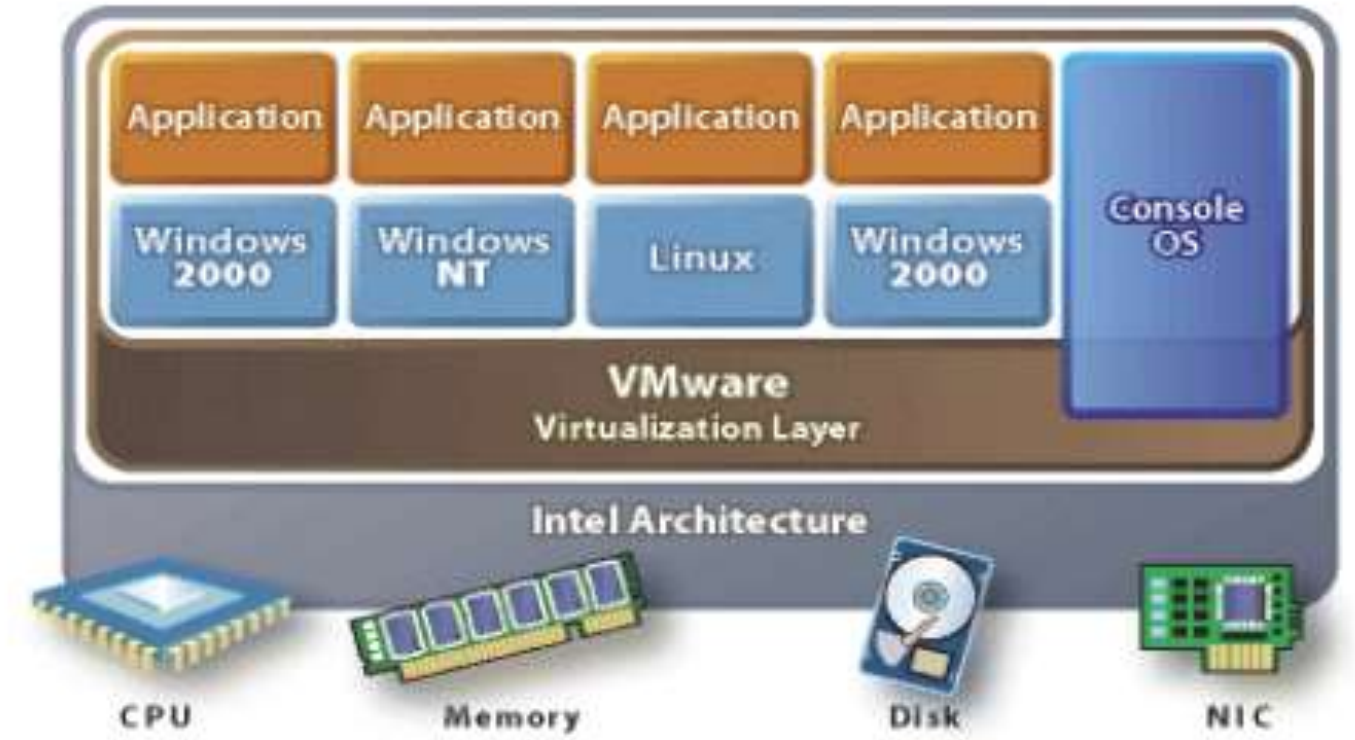
Virtualization benefits:

Mobility of OS services from one HW platform to another

Consolidation of services on heterogeneous OSs to a single HW platform

Virtualization visualized

Image by Carl Waldspurger



Virtualization Nuances

Full Virtualization

Guest OS have full access to the underlying machine:

- Hypervisor isolates the Guest OSs from each other
- Virtualization support at the HW level
- No modification to the OS binaries

Para Virtualization:

Guest OS interface with the Hypervisor:

- Some modification to the Guest OS binaries
- No virtualization support needed at the HW level

Types of Hypervisor

Type I Hypervisor:

Hypervisor run on the bare metal
Also referred to as native Hypervisor

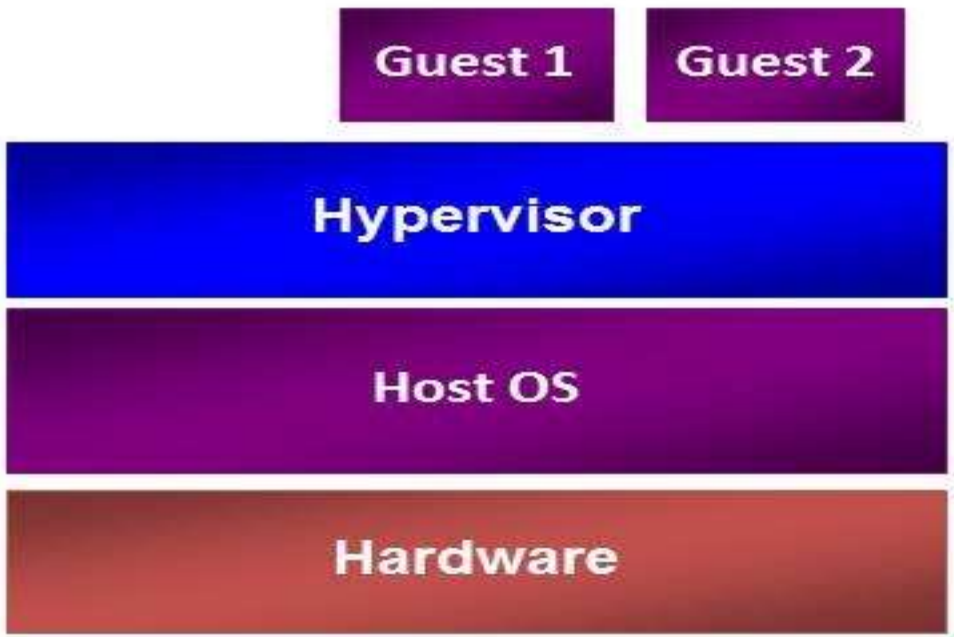
Type II Hypervisor:

Hypervisor hosted on an underlying OS
Also referred to as hosted Hypervisor

Hypervisor Design:

Two approaches

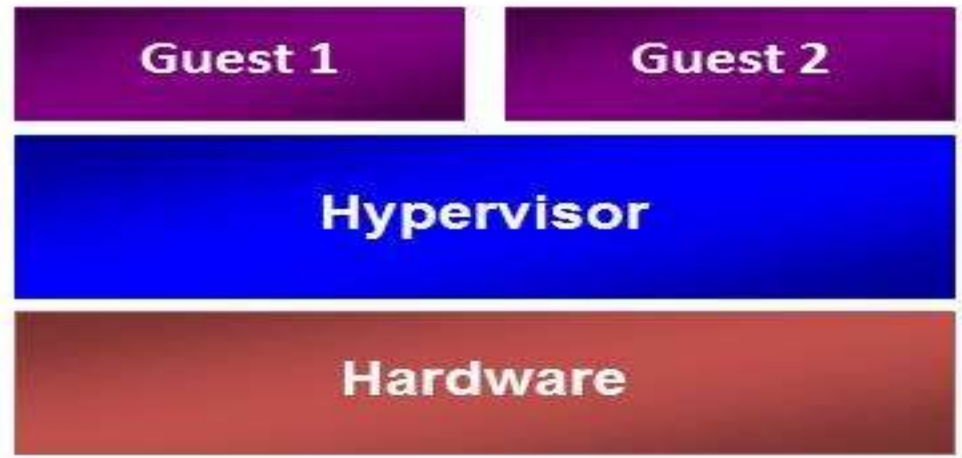
Type 2 Hypervisor



Examples:

- Virtual PC & Virtual Server
- VMware Workstation
- KVM

Type 1 Hypervisor



Examples:

- Hyper-V
- Xen
- VMware ESX

Road Map

Today:

- Process Virtualization
  - Resource Sharing and Isolation
  - Migration and Load Balancing
- Storage Virtualization

Tomorrow:

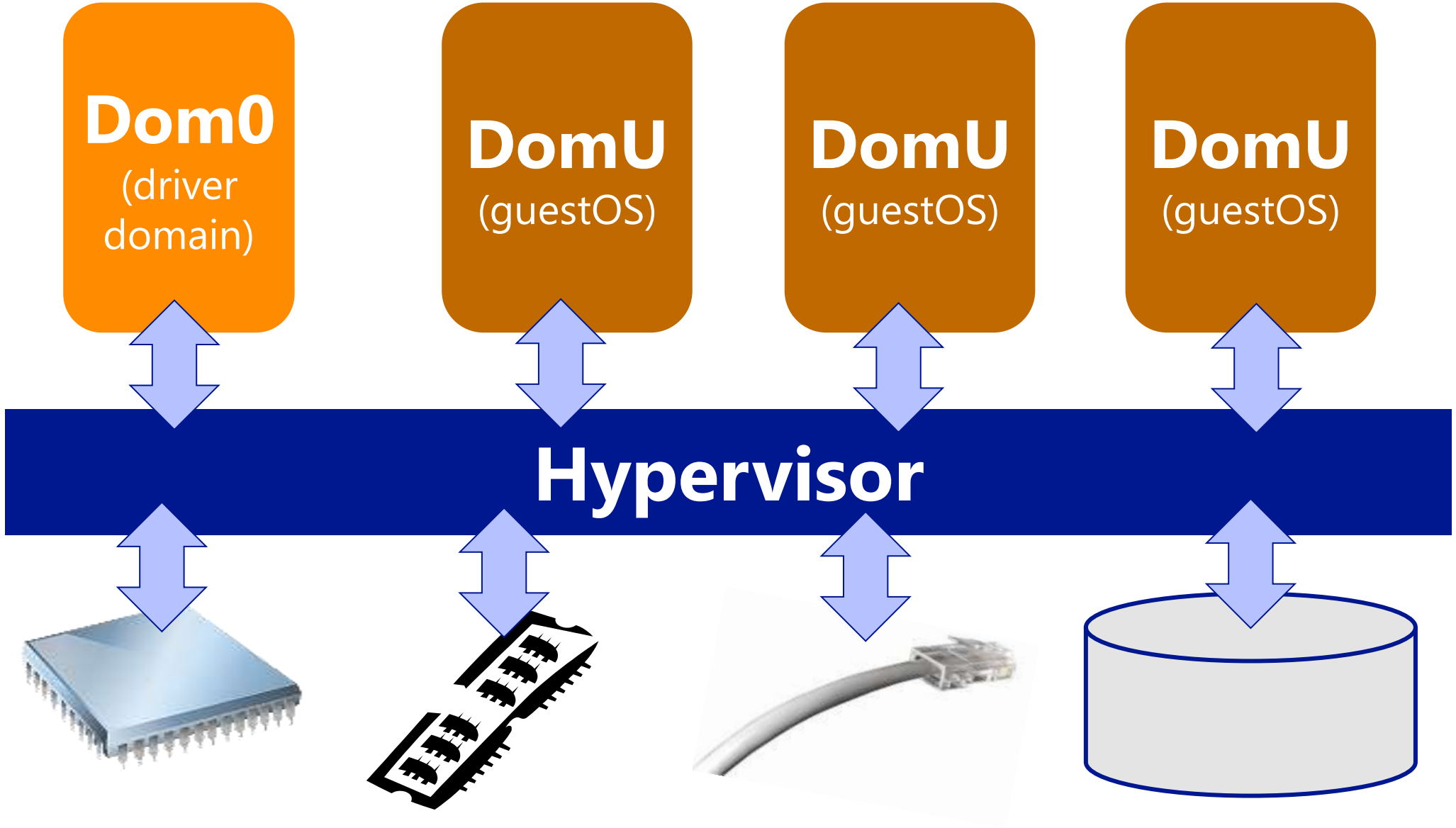
- Database Virtualization

Resource Allocation

"Better virtualization through sharing"

Xen Architecture (Bare-Metal)

[Barham et al. SOSP 2003]



Resource Allocation

Have to allocate how Hypervisor shares resources between VMs (CPU, memory, I/O, …)

- Can statically allocation of some resource (MHz , MB).

Better to have some flexibility in how resource allocation is controlled.

Goals of Resource Allocation

Minimize under-utilized resources

Performance isolation

Support flexible resource sharing

Controls

Shares

Specify importance by allocating shares proportionately

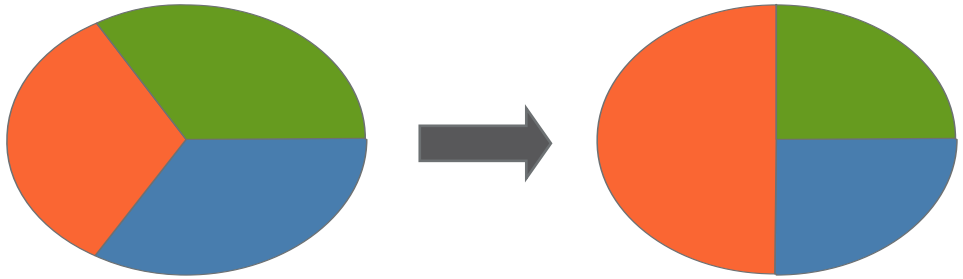
Reservation

Minimum amount (MHz, MB) guaranteed. Even if overloaded

Limit

Maximal amount (MHz, MB) guaranteed. Even if underloaded

Benefits of Share Based



Change shares for **VM**

Dynamic reallocation

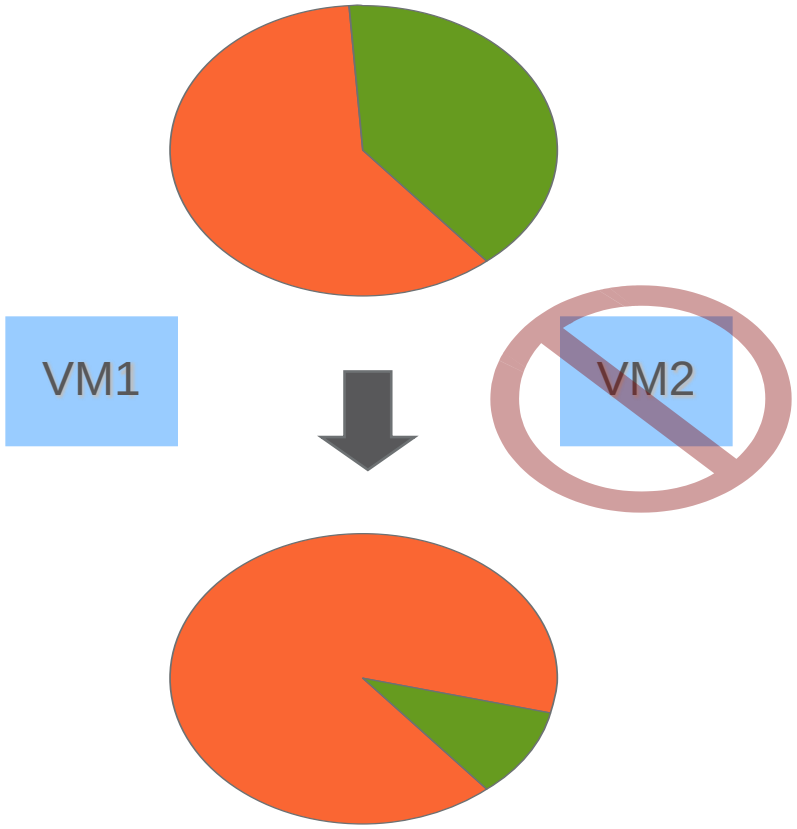
Slide by Waldspurger

Proportional-Share Scheduling

Simplified virtual-time algorithm

Virtual time = usage / shares
Schedule VM with smallest virtual time

Reservation Example



Total capacity

1800 MHz reserved

1200 MHz available

Admission control

2 VMs try to power on

Each reserves 900 MHz

Unable to admit both

VM1 powers on

VM2 not admitted

Slide by Waldspurger

Other Issues

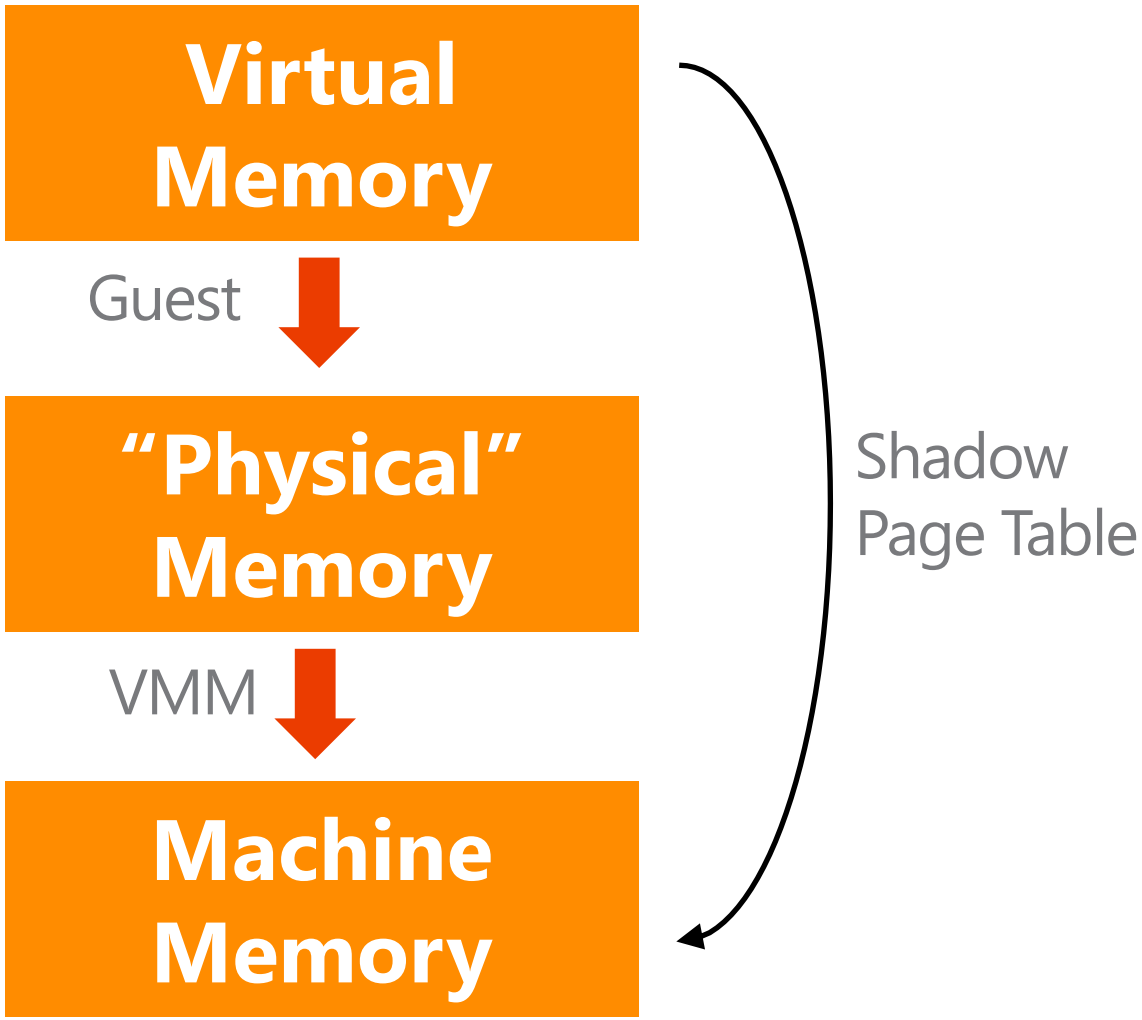
Measuring Usage

Multi-core/ inter-processor load balancing

Cache affinity

Interrupt balancing

Virtualized Memory Management



Extra layer of memory indirection by having hypervisor map memory pages.

VMs are allocated a static amount of the machines memory.

Memory Resource Management in VMware ESX Server

Carl A. Waldspurger
VMware, Inc.

OSDI 2002

Memory Resource Management in VMware ESX

Focus

Improve hypervisor's memory management sharing

Key Contributions

Memory ballooning

Page sharing to minimize wasted memory

(Limiting idle pages)

Memory Reclamation

How to implicitly reclaim memory without using direct hooks?

Inject ballooning process which inflates / deflates.

Not quick and not reliable, but lightweight

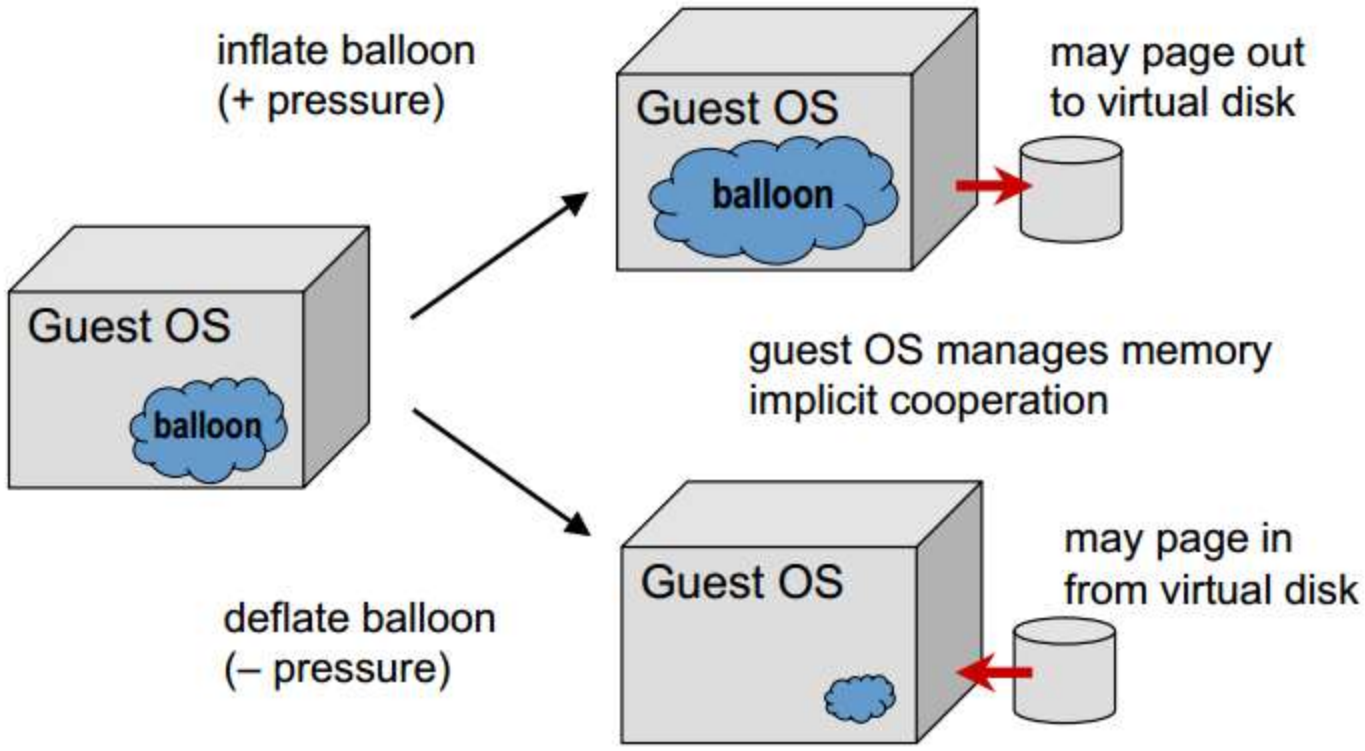


Image by Waldspurger

Sharing Memory

Possible duplicate memory pages (OS, apps, data, etc)

Transparent page sharing requires OS hooks (copy on write)

Content based sharing. Hash to ID possible matches.

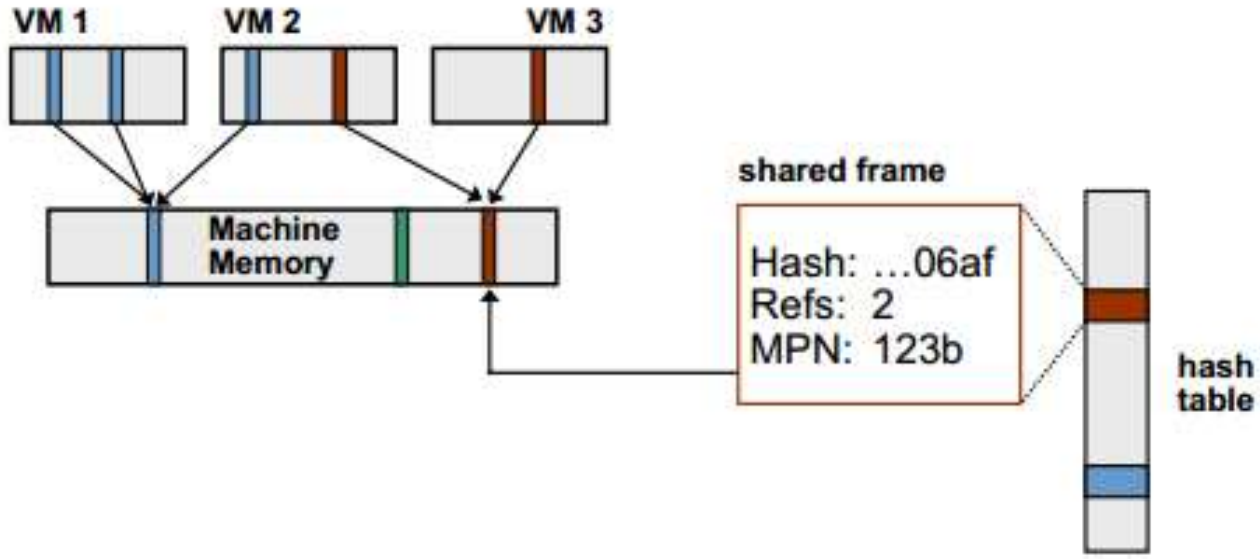


Image by Waldspurger

Other Issues

Other resources need sharing (network, disk).

Will talk later on disk related issues.

Shift gears to multi-machine resource allocation…

Live Migration

"Migration for fault tolerance, load balancing, and system maintenance"

Virtualization in Distributed Environments

What is different? Why is it desirable?

Narrow VMM interface minimizes the residual dependencies

Clean migration of in-memory state (both kernel and user level)

Separation of concerns: users (full control over software) and operators (full control over infrastructure)

Live Migration of Virtual Machines

Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen
Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield

University of Cambridge, University of Copenhagen

NSDI 2005

Live Migration of Virtual Machines

Focus

- Want to migrate a live VM state (non-persistent state)
- Minimize disrupting services

Key Contributions

- Canonical live migration for VM
- Optimizations minimize downtimes and disruption

Live Migration Design Space

Three memory transfer approaches:

- Push
- Stop-and-Copy
- Pull

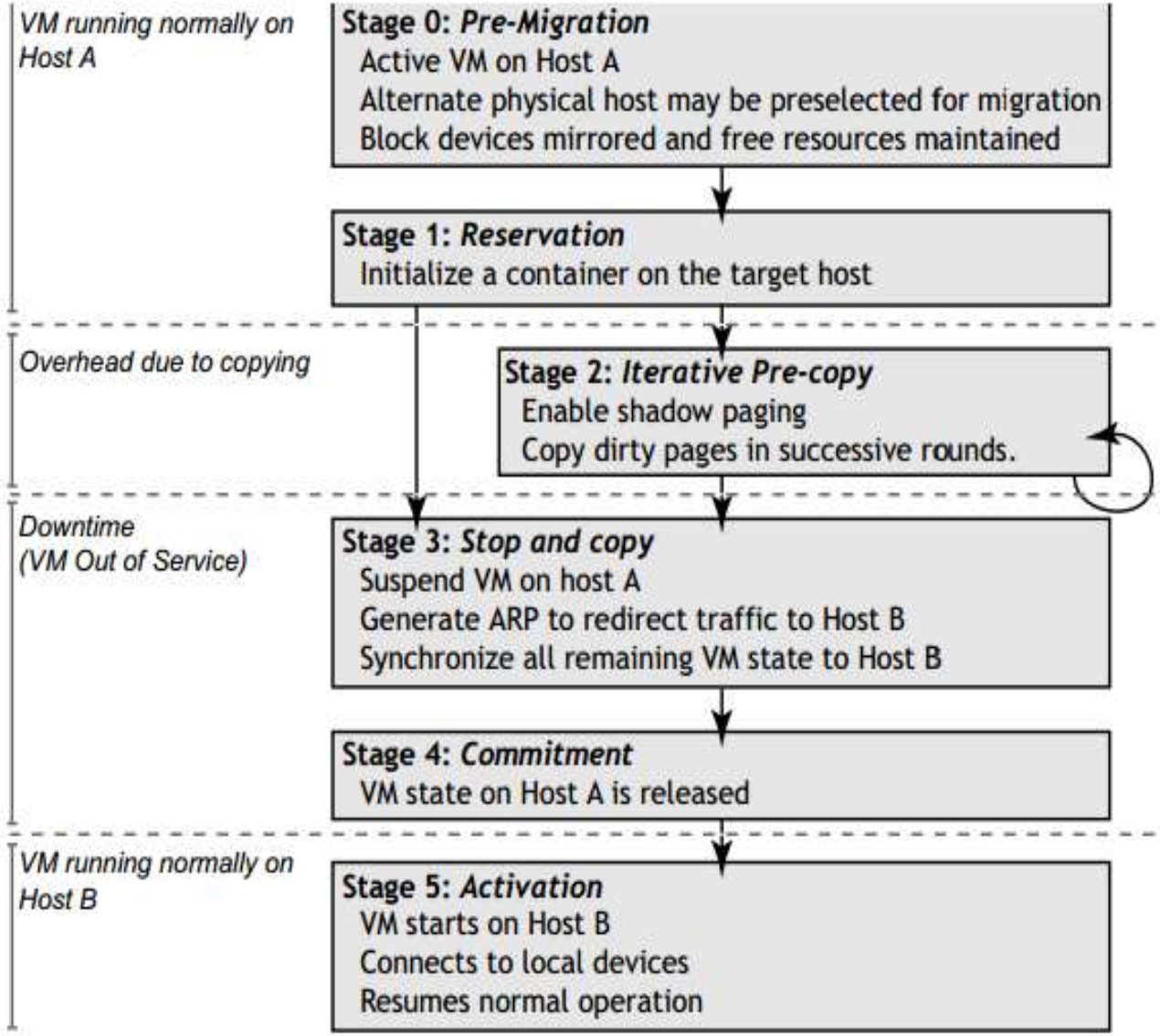
Implementation

Pre-copy migration

- **Bounded** iterative push phase
  - Rounds
  - *Writable Working Set*
- Short stop-and-copy phase

Careful to avoid *service degradation*

Migration Timeline



Technical challenges: Handling Local Resources

Open network connections

- Migrating VM can keep IP and MAC address.
- Broadcasts ARP new routing information

Local Storage

- Network-attached Storage

Technical challenges: Writable Working Set

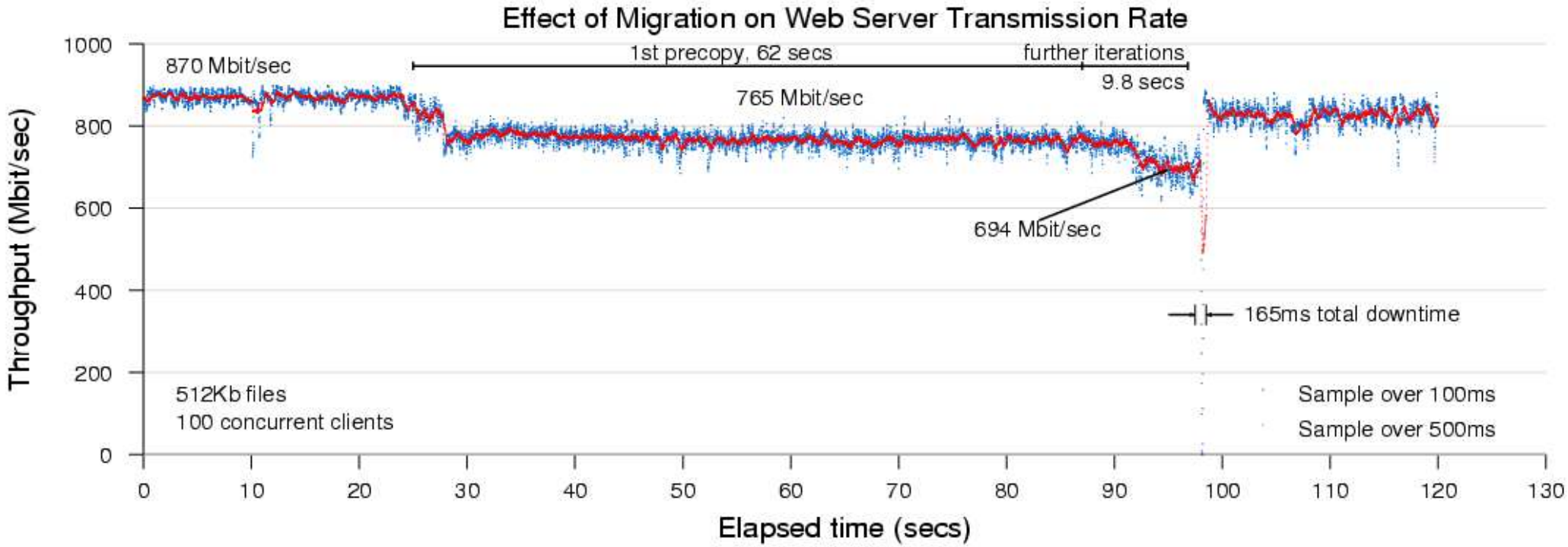
Overhead in live migration is a consistent copy of updating state

How much time to spend on iterative pre-copy?

Use Xen's shadow page tables

- Writes are trapped by hypervisor
- Can use this to determine WWS

Migrating a Simple Web Server



Leveraging Virtualization over Multi-Datacenters

Current challenges in the context of Cloud Computing:

- Large-scale outages resulting in data-center failures
- Planned shutdowns of data-centers

Internet Services/applications that run 24X7:

Need to migrate live services/applications from one data-center to another

➔ VM migration over WAN

Live Wide-Area Migration of Virtual Machines Including Local Persistent State

Robert Bradford, Evangelos Kotsovinos, Anja Feldmann, Harald Schiöberg

Deutsche Telekom Laboratories

Virtual Execution Environments 2007

Live Migration over WAN with persistent state

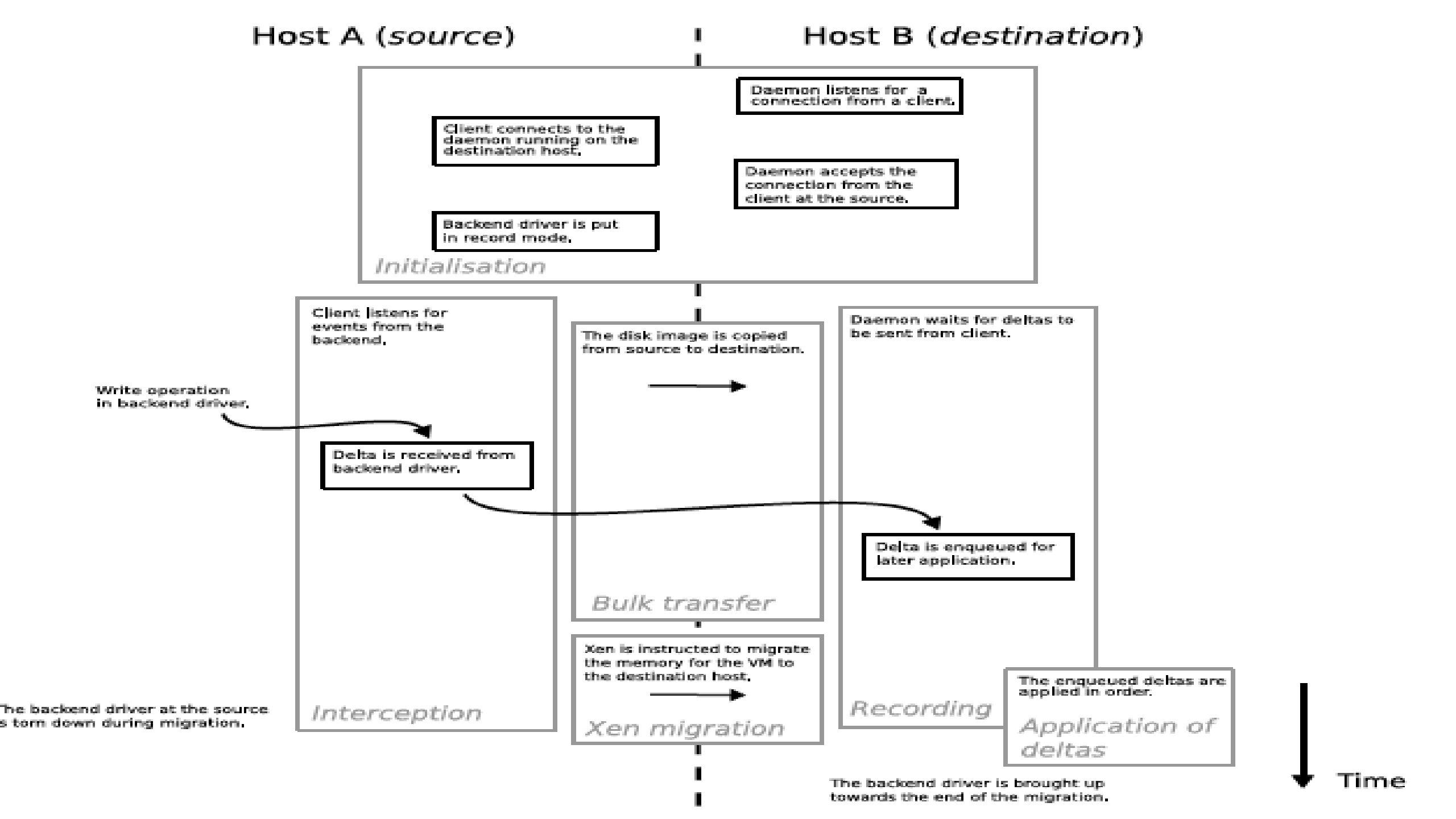
Focus

Extending live migration to manage migrating over wide area

Key Challenges

Ability to migrate local persistent (block) storage

Migrate open connections over WAN ← lightweight network redirection combined with new address via dynDNS


Host A (source)
Host B (destination)
Daemon listens for a connection from a client.
Client connects to the daemon running on the destination host.
Daemon accepts the connection from the client at the source.
Backend driver is put in record mode.
Initialisation
Client listens for events from the backend.
The disk image is copied from source to destination.
Daemon waits for deltas to be sent from client.
Write operation in backend driver.
Delta is received from backend driver.
Delta is enqueued for later application.
Bulk transfer
Xen is instructed to migrate the memory for the VM to the destination host.
The backend driver at the source is torn down during migration.
Interception
Xen migration
Recording
The enqueued deltas are applied in order.
Application of deltas
The backend driver is brought up towards the end of the migration.
Time

Main Takeaways

VM Migration over WAN requires:

- Transfer of network connection (Tunneling and dynDNS)
- Transfer of persistent storage (capture block writes)
- Throttle write heavy workloads

The paper reports that this is indeed feasible:

- Migration of a web server over WAN in less than a minute.

Many Other VM Migration Techniques

Checkpoint and recovery

Compression

Much more

Data-center Administration Challenges

Servers multiplexed across multiple applications:

- Many apps per server and components distributed across multiple servers
- Dynamic workload fluctuations caused by incremental growth, time-of-day effects, and flash crowds.

Virtualization advantage:

- The ability to flexibly remap physical resources to virtual servers to handle workload dynamics
- Manual detection and migration lacks the agility to respond to sudden changes in workload dynamics

Data-center Automation using Virtualization

Goal:

Automate the task of monitoring and detecting hotspots

Determining a new mapping of physical and virtual resources to mitigate the problem of hotspots

Initiating the necessary VM migrations to realize this new mapping

Black-box and Gray-box Strategies for Virtual Machine Migration

Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif

University of Mass. Amherst, Intel

NSDI 2007

Black- and Gray-box Strategies for VM Migration

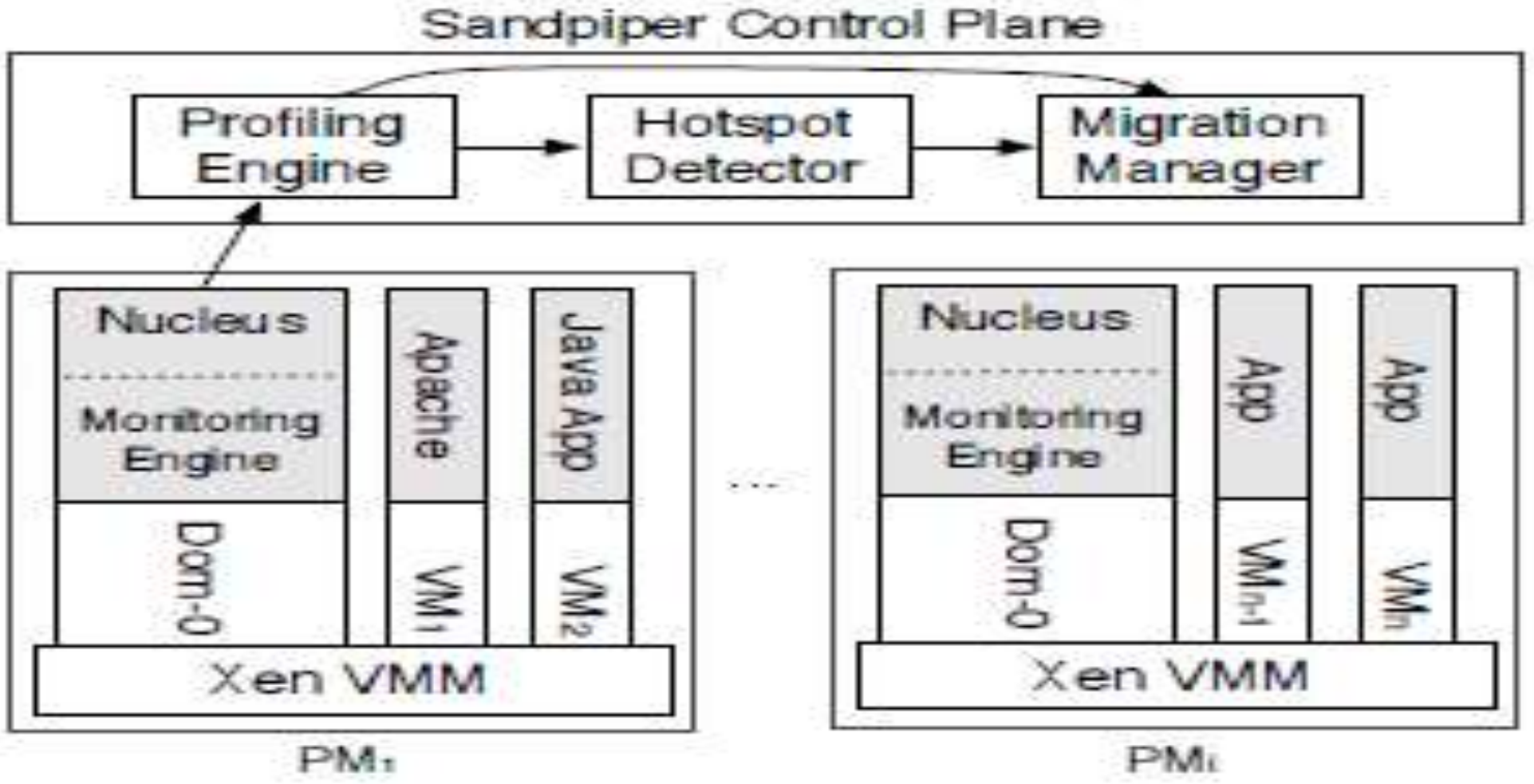
Focus

- Method to monitor and detect hotspots
- Resolving overloaded servers

Key Contributions

- Black- and gray-box monitoring
- Identifying and migrating VMs to alleviate overloaded servers

Sandpiper Architecture



Black-Box Monitoring

Infer VM workload from "VM external" observations:

- **domain-0** (driver domain)

Interested in monitoring during each interval:

- CPU Usage
- Network Usage
- Memory
- ~~Disk~~ Assumed to be external

Black-box Monitoring

CPU

Each VM CPU usage is derived by scheduling events

Ignores VM overhead of disk IO and network CPU

Network

Hypervisor implements the network interface driver (direct)

Memory

Cannot directly tell memory **usage**

Geiger to derive working set from swaps

Resource Provisioning

Challenge when CPU is saturated, how to determine what each VM needs?

Limitation of black box model.

Gray-box Monitoring

Black-box is good when we cannot peak inside VM.

Certain environments this OK, not always feasible.

Lightweight monitoring daemon inside VM to watch:

/proc for CPU, memory, network, etc

Application level stats (needs hooks) e.g. request rate, service time, etc

Generate Profile

Use black or gray parameters to build profile over sliding window W

Build a probability distribution for resources and time series of observations.

Profile to determine what resources are needed.

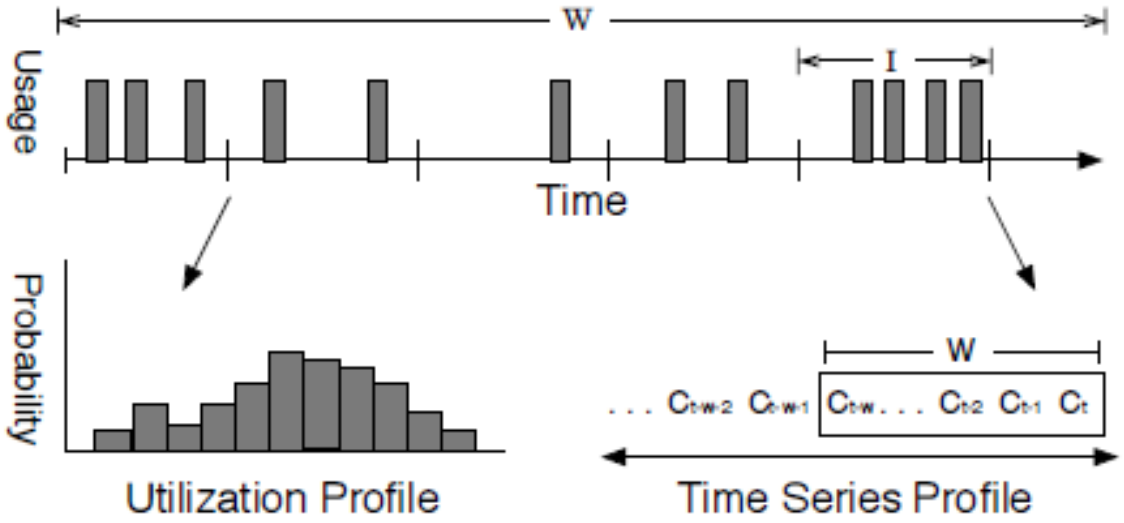


Figure 2: Profile generation in Sandpiper

Hotspot Detection

Thresholds or SLAs are violated for **sustained period.**

K/N most recent observations in violation

Auto-regressive prediction on time series to predict violation

Want to mitigate sustained resource that does not appear to be trending down.

Migration

Migrate from hottest to coldest which

Minimize the overhead of migration

Start with migrating VM with highest usage to size (RAM) ratio

- Maximize the load being migrated with minimizing overhead
- Ensure destination has capacity for profiled resources

Relieving Hotspot

Want to avoid creating new hotspots.

Need to find a spot to migrate the VM to. Use volume as a heuristic to account for multiple resources.

$$Vol = \frac{1}{1 - cpu} * \frac{1}{1 - net} * \frac{1}{1 - mem}$$

Might need to swap VMs if lacking capacity. Requires a 3rd server and migration. Limit use of swaps.

Other Projects

VMWare DRS (Dist. Resource Scheduler)

Quotas Specified

Violin and Virtuoso

Scientific computing. Light SLA

Shirako

Application resource driven

Virtualized Storage

Motivation

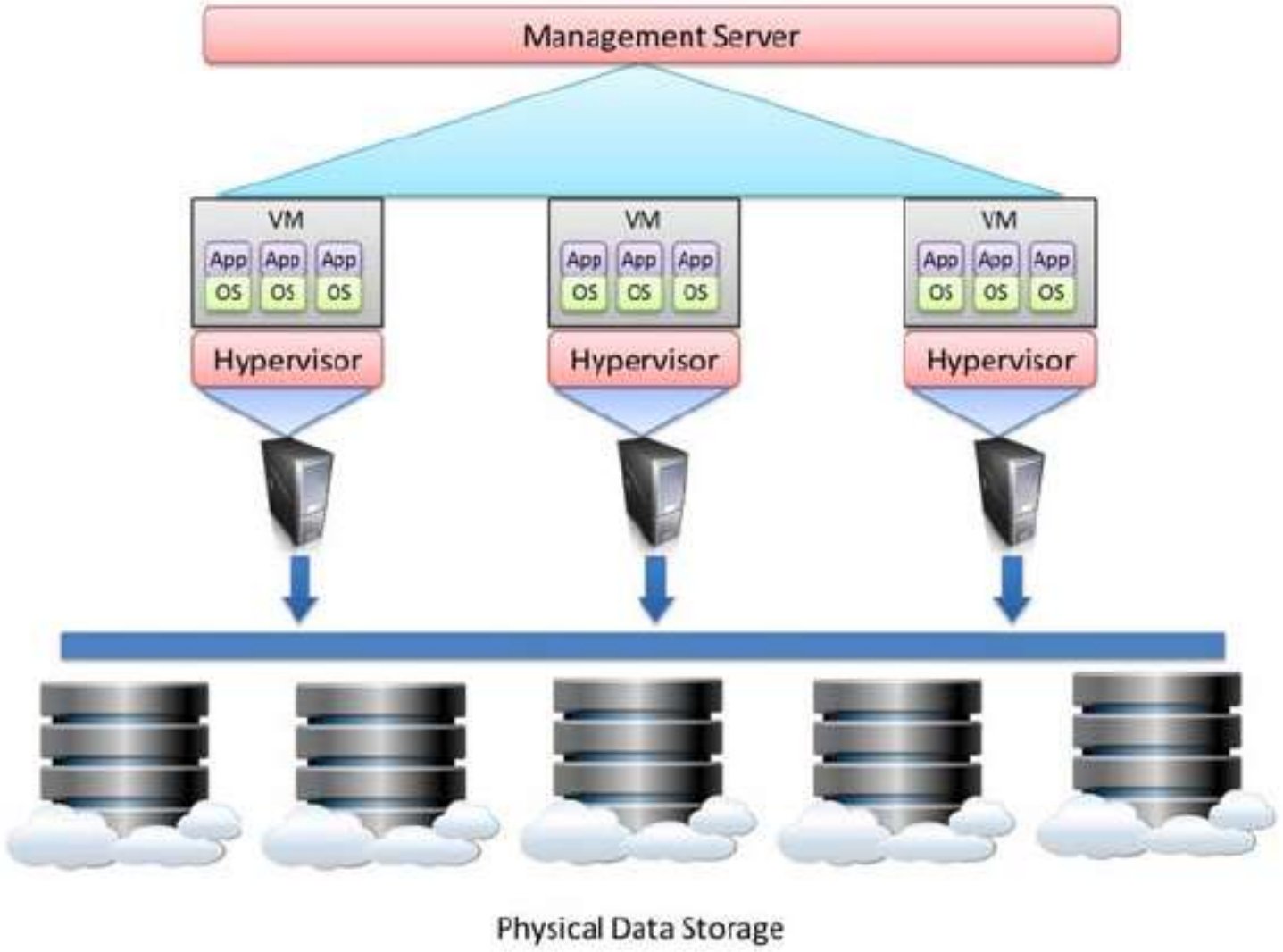
We know how to do automated load balancing of CPU and Memory resources across a cluster of servers using live migration.

We know how to do live migration between storage devices.

No online tools for **automated virtual disk placement** and IO load balancing across storage devices.

Initial placement of virtual disks and subsequent data migration across data stores needs to be based **in workload characteristics, device model and analytic formulation** to improve IO performance and utilization.

Storage Virtualization Architecture in Data Centers



Virtualized Storage Outline

Basil —passive modeling of workloads and storage devices.

Pesto —dynamic monitoring and modeling of workloads on devices.

Romano —construct a detailed workload specific model.

Soundararajan et al. —consider database applications specifically.

BASIL: Automated IO Load Balancing

Ajay Gulati, Chethan Kumar, Irfan Ahmad, and Karan Kumar

VMWare

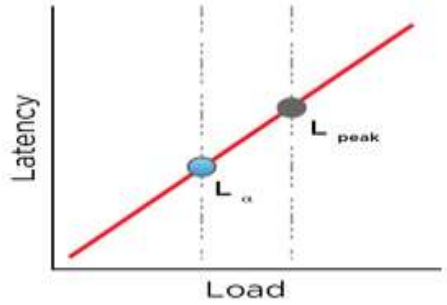
Fast 2010

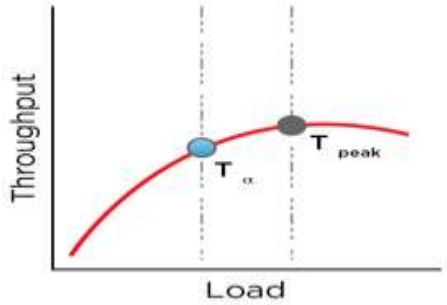
BASIL Approach

Black Box (vs white) and Relative Modeling (vs absolute)

Model workload and device independently.

Primary metric: Use **IO latency** which is linear in number of outstanding IO requests, unlike throughput and bandwidth.





Workload Characteristics

To a first approximation, the following parameters are inherent to the workload and independent of the underlying device:

- Average Outstanding IO (OIO)
- IO size
- Read %
- Random %

Latency = $(k_1+OIO)(k_2+IOsize)(k_3+Read\%)(k_4+random\%)$

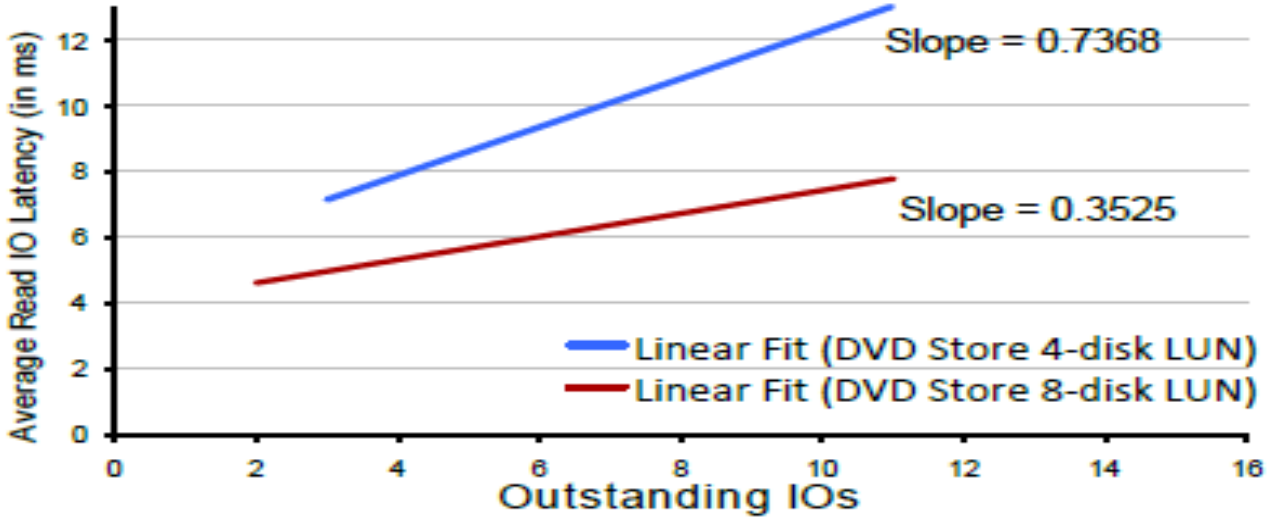
$K_1=1.3, k_2=51, k_3=0.4, k_4=0.6$ from empirical data

OIO dominates; IO size only when change is LARGE.

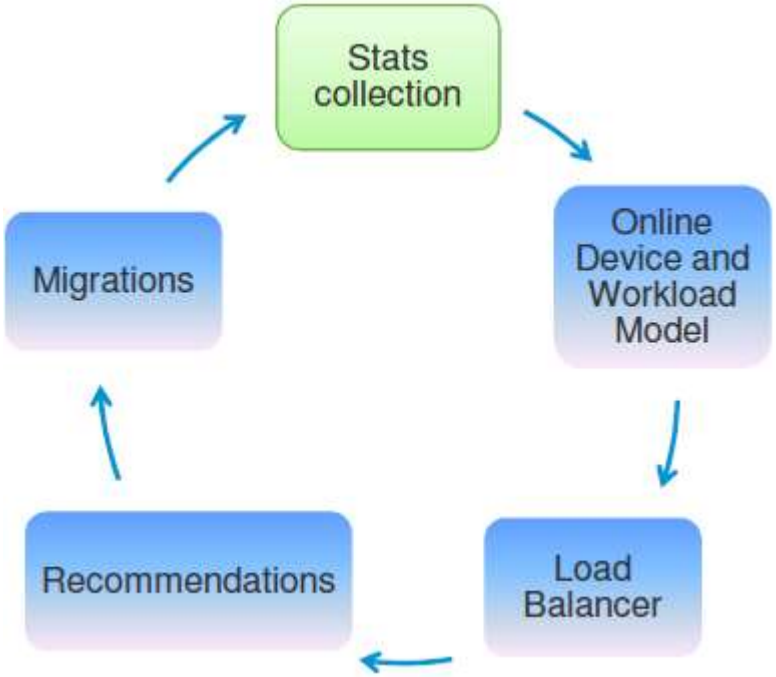
Storage Device Characteristics

Device performance can vary widely depending on number, type, etc of disk, but hidden from hosts.

P: inverse of the slope of the relationship between **average latency** and **outstanding IOs** for given device.



Load Balancing



Define **Normalized Load (NL)** on a device as

∑ workload on device/P

Assign workload to equalize **NL** across data stores

Iteratively move virtual disk from device with maximum NL to device with minimum NL (hottest to coldest).

Experiments: BASIL provides good average latency and high utilization.

Pesto: Online Storage Performance management in Virtualized Datacenter

Ajay Gulati, Ganesha Shanmuganathan, Irfan Ahmad, Carl A. Waldspurger, Mustafa Uysal

VMWare

SoCC 2011

Motivation

BASIL encountered several challenges in real production environments.

Uses passive observations, causing different models produced for same device over time.

A robust model requires long observation periods
(and might still f ail to observe a wide enough range)

No cost-benefit analysis.

Pesto provides IO load balancing with cost-benefits analysis using continuously updated storage performance models.

Pesto Approach

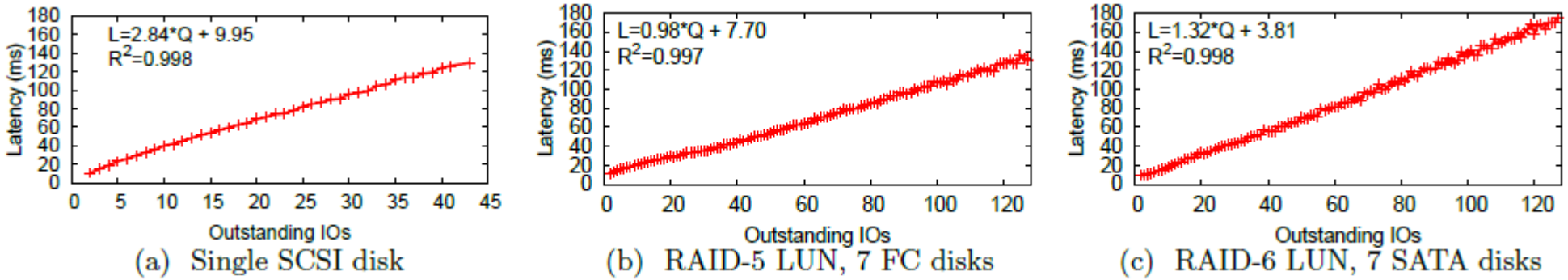
Uses **workload injector** to automatically generate performance models while the system is running.

Periodically collect stats on the way virtual disks are accessed on each datastore.

Compute **models** for various datastores and virtual disk stats.

Use stats to **load balance** using migration or new placements.

Pesto Workload Performance Model



(a) Single SCSI disk (b) RAID-5 LUN, 7 FC disks (c) RAID-6 LUN, 7 SATA disks

The LQ slope of a workload can predict the device throughput of that workload closely

Empirically and analytically derive linear relationship between **IO latency and Outstanding IO**

Latency = m * OIO + C (L = mQ +C)

m is the LQ slope and C its intercept

Peak Throughput = 1/m (using Little's law))

Philosophy: Approximate models obtained online (vs offline detailed models).

Performance Model Generation

Workload injector: run during idle periods to generate performance models periodically.

- Issues **random read IO**, and collects latency measurements; then compute slope.

- Compute **the slope in the model using least squares.**

Fine grain stats are measured on a per-virtual disk periodically, and collected by a manager server to compute percentiles based on workload metrics (OIOs, IO size, read% and randomness).

Dependence on IO workload

High variance in LQ slope based on workload: randomness, IO size, read%

Except in extreme cases, variance is limited to 15%.

Therefore, performance estimates are based on slope of benchmark closest to real workload.

LQ slope is representative of **overall** throughput even if workload is a **mix of multiple** IO streams.

Load balancing

IO load balancing: equalize latency across devices.

- Ignore moves where 'median' latencies are lower at source vs destination.

Cost-Benefit Analysis: Migration occurs based on estimates of both cost and benefits of migration from source to a destination.

Performance-based capacity planning (how many workloads on a given device)

- Use Latency thresholds and peak throughput policies.

Pesto improved throughput by 10% and reduced peak latency by up to 19%.

ROMANO: Autonomous Storage Management using Performance Prediction in Multi-tenant Data Centers

Nohhyun Park, Irfan Ahmad, and David J. Lilja

University of Minnesota and CloudPhysics

SoCC 2012

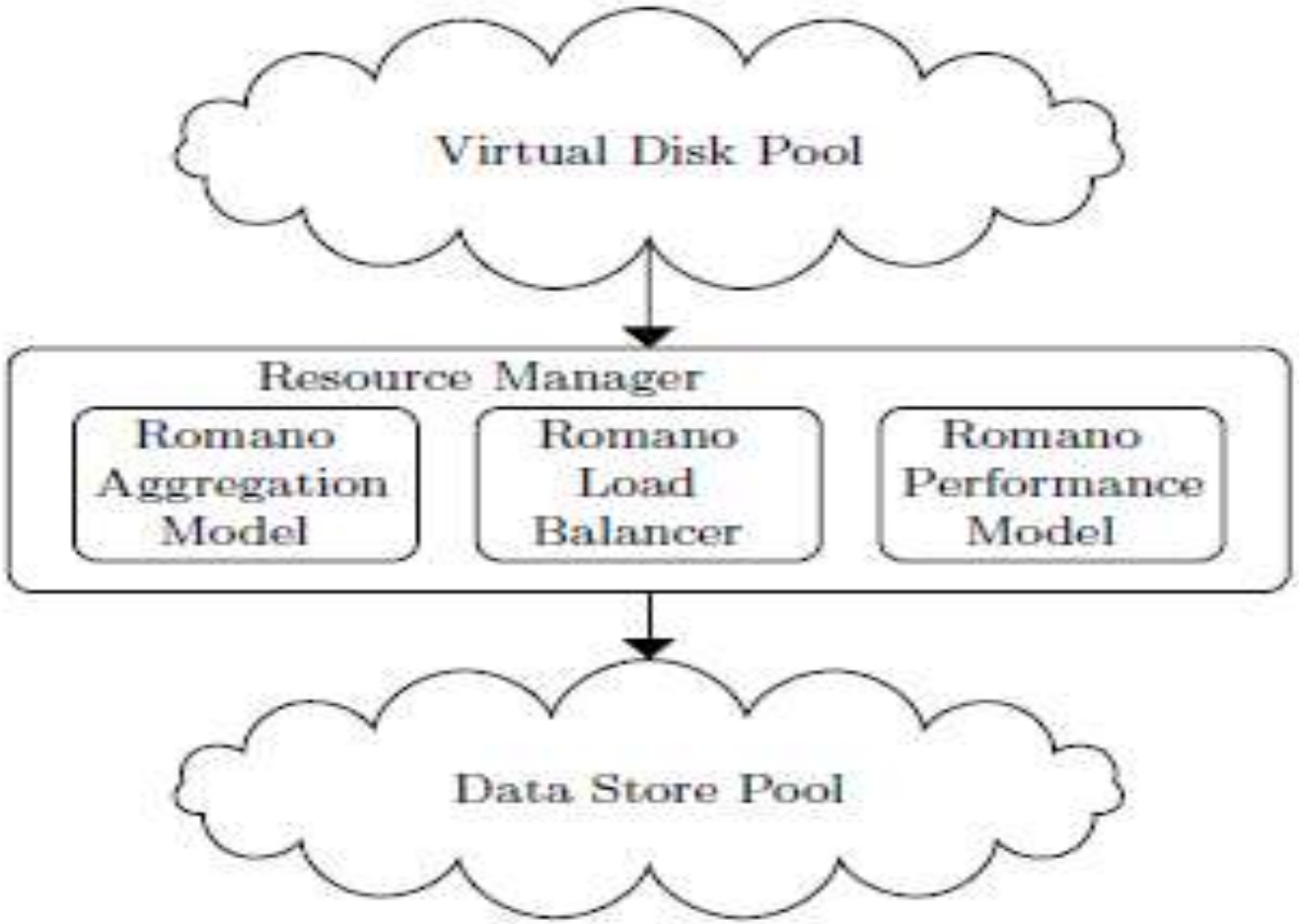
Motivation

Basil and Pesto mainly use heuristics based on empirical studies.

Romano constructs workload specific performance models of storage devices automatically.

Explicitly considers multiple heterogeneous workload on a single storage device.

High level View of Romano



Romano Performance Model

How to model storage performance to incorporate workload/device dependence as well as inherent noise?

A storage system's performance P depends on both the system S and the workload W:

$$P = f_P(S,W)$$

Using linear regression:

$$P = M_S * M_W + \varepsilon = M_P + \varepsilon$$

Where **P** is average IO latency and $\mathbf{M_P}$ is the desired performance model.

Approach: Measure **P** to determine $\mathbf{M_W}$ then solve for $\mathbf{M_S}$.

IO Workspace

Workload characteristics used to derive M_W:

$$random\%(X) = \{r | r \in \mathbb{I}, 0 \leq r \leq 100\}$$
$$read\%(R) = \{o | o \in \mathbb{I}, 0 \leq o \leq 100\}$$
$$IOsize(Z) = \{s | s \in 2^n B, n \in \mathbb{N}, 9 \leq n \leq 19\}$$
$$OutstandingIO(O) = \{i | i \in \mathbb{N}, 1 \leq i \leq 128\}$$

Space of experiments **TOO BIG**.

Romano uses **sampling**.

$$X = \{0\%, 25\%, 50\%, 75\%, 100\%\}$$
$$R = \{0\%, 25\%, 50\%, 75\%, 100\%\}$$
$$Z = \{1K, 2K, 4K, 8K, 16K, 32K, 64K\}$$
$$O = \{1, 2, 4, 8, 16, 32, 64\}$$

Derive $M_P = M_W * M_S$

Romano Aggregation Model

How to aggregate workloads so that the total work required can be predicted?

$$(X_{new}, R_{new}, Z_{new}, O_{new}) = (f_X(X_i, T_i), \frac{\sum_i R_i T_i}{\sum_i T_i}, \frac{\sum_i Z_i T_i}{\sum_i T_i}, \sum_i O_i)$$

Outstanding IO (O): addition

Read Ratio (R) and **Size** (Z): weighted (by throughput T) average based on ratio of requests.

Randomness (X): trickier, function of seeks within and across virtual disks, proportionally weighted by throughput (f_X).

Romano Load balancer

How Romano load balances heterogeneous workloads and systems?

Uses **Simulated Annealing** to Optimize global state rather than single storage migration.

Minimize **mean** and **maximum** latency.

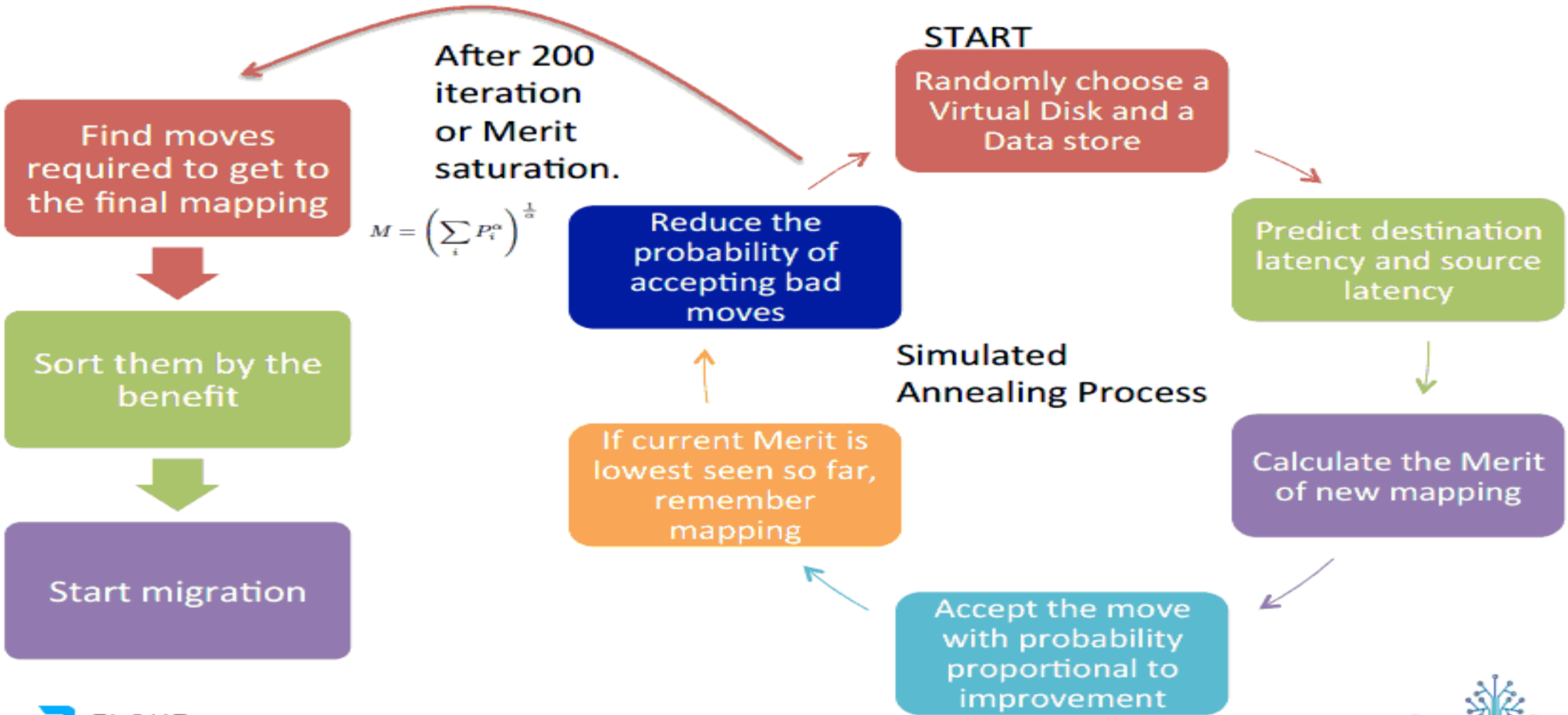
Load Balancing metric of Merit (M):

α controls weight of max latency.

Proactive: continuously adjusting global st

$$M = \left(\sum_i P_i^{\alpha} \right)^{\frac{1}{\alpha}}$$

Load Balancing Algorithm



CLOUD PHYSICS

CRIS

Romano Summary

A **performance predictor** with workload and storage device characteristics as input.

Storage performance needs to deal with **heterogeneity**: different workloads interact differently with different devices.

Load balancing is **not a bin packing problem**, Romano uses a probabilistic approach to find a pseudo–optimal solution.

Dynamic Resource Allocation for Database Servers Running on Virtual Storage

Gokul Soundararajan, Daniel Lupei, Saeed Ghanbari, Adrian Daniel Popescu, Jin Chen, and Cristiana Amza

University of Toronto

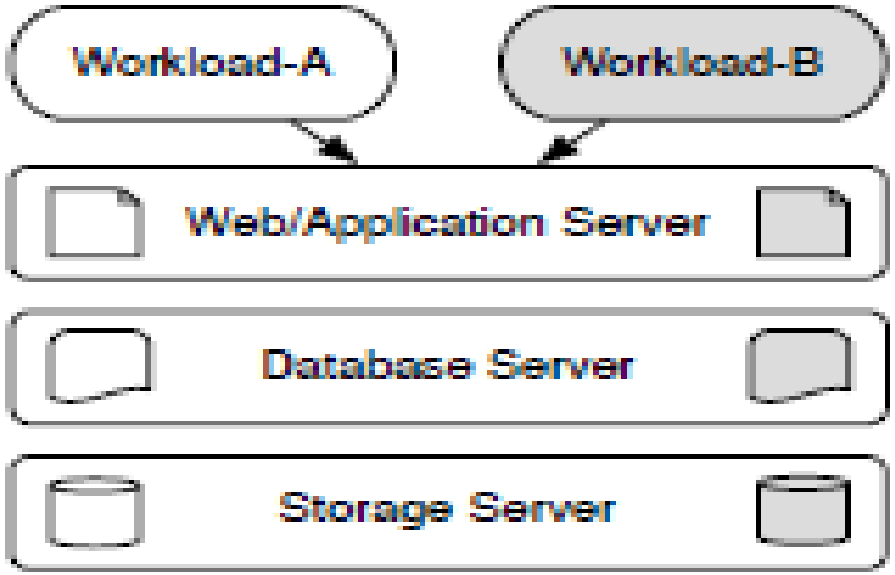
FAST 09

Data Center Infrastructure

Adopt a **holistic** approach.

Focus on interdependencies between resources:

- Storage and database caches
- Storage bandwidth



Overall goal:

- Effective resource partitioning for database servers running on virtual storage

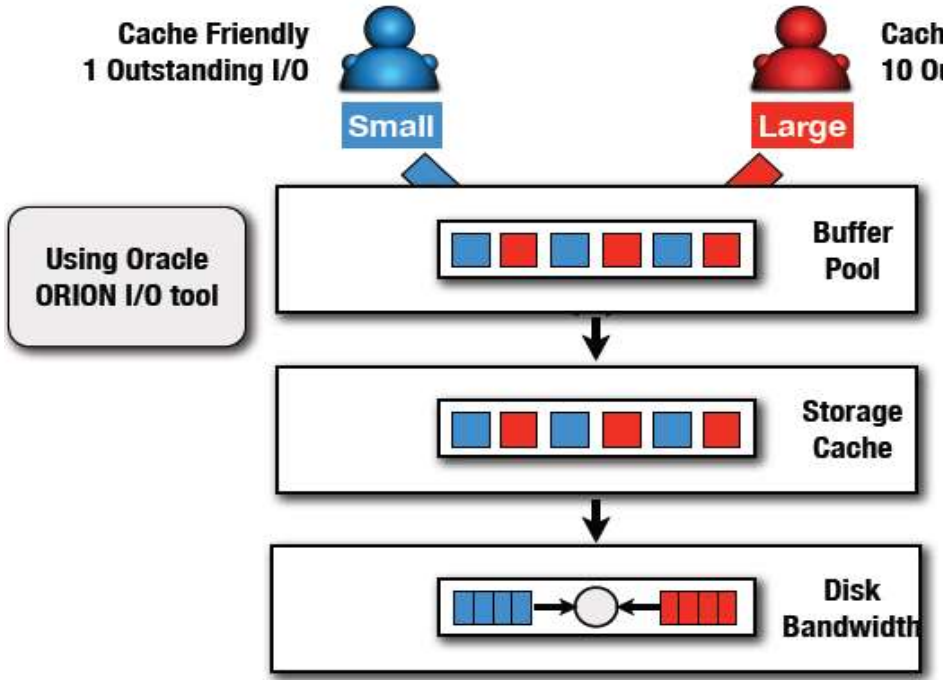
Approach:

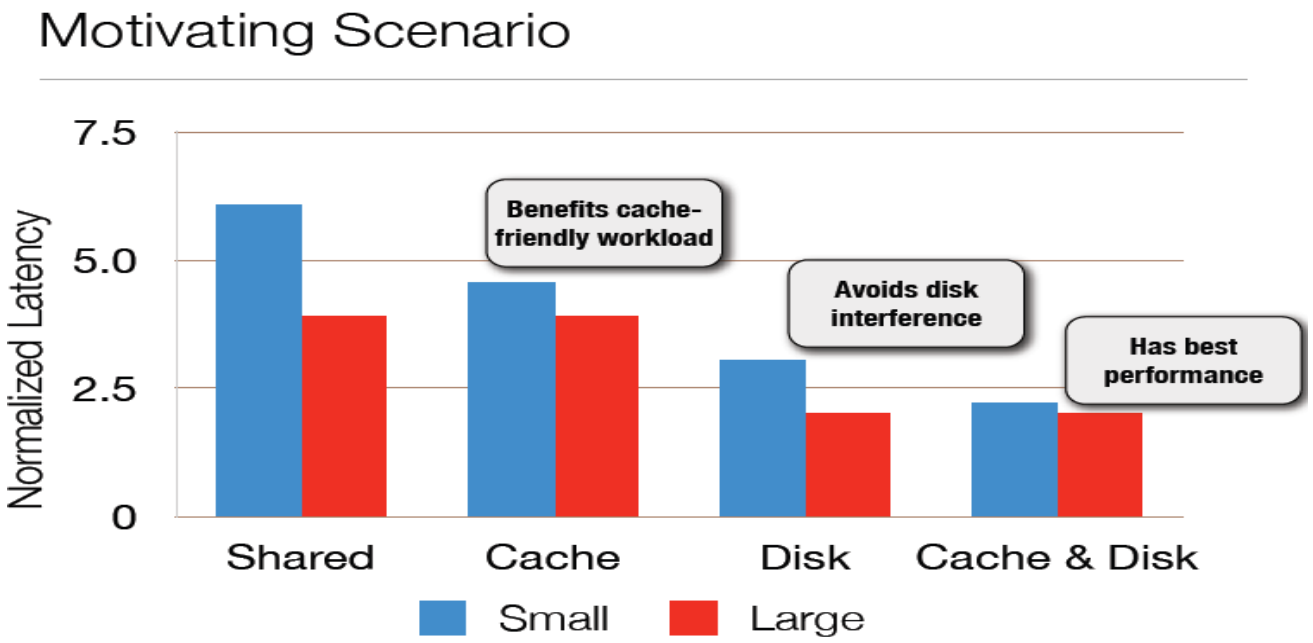
- Per-application performance model based on minimal statistics
- Experimental sampling and statistical interpolation for refinement

Simple Motivating Example

Storage Cache: Use Miss Ratio Curves algorithm

Storage Bandwidth: exhaustively find best.





Dynamic Multi-Resource Allocation

Goal: Partition m resources among n applications to minimize the sum of the data access latencies of all applications.

Per-Application Performance Model

Model cache hierarchy as a single-level cache.

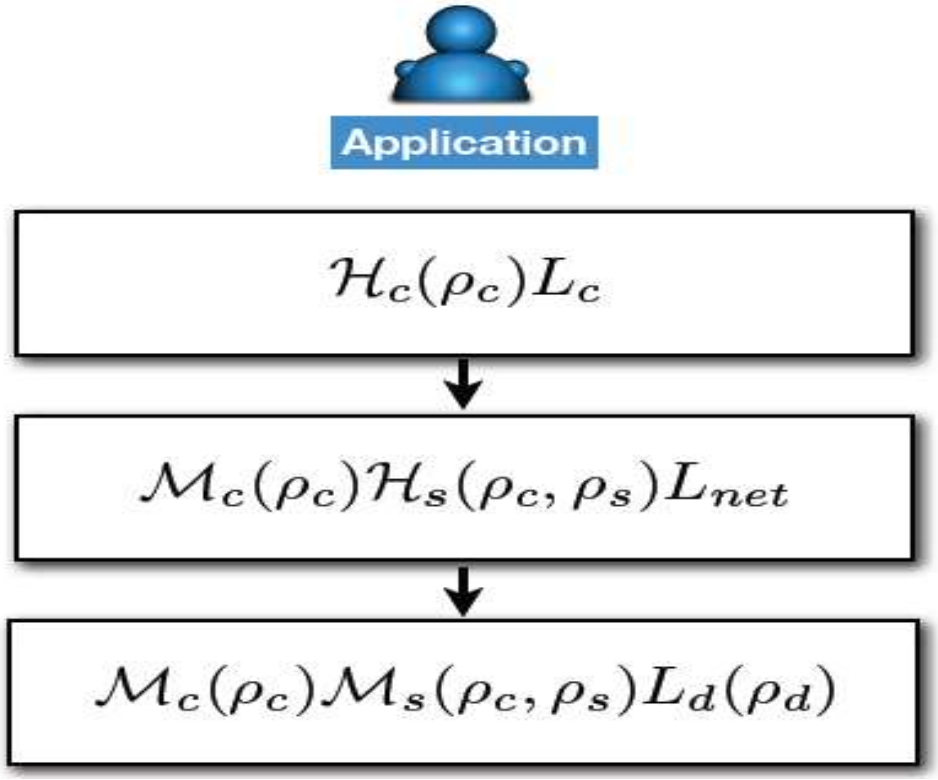
- Uncoordinated LRU cache hierarchy: max size
- Coordinated Demote cache hierarchy: sum of sizes.

Model disk as a closed loop system (interactive and constant number of users): rate of serviced requests roughly equals rate of incoming requests.

- Empirically derive baseline latency (full disk access) for an application. $L_d(1)$
- Given allocation of disk (ρ_d) model the expected latency :

$$L_d(\rho_d): L_d(1)/\rho_d$$

Putting it all together



Many simplifying assumptions introduce inaccuracies.

Fine Tuning:

Iteratively collect samples from different configurations then test using statistical regression.

Provides an end to end multi-resource allocation method based on a unified resource to performance model.

Virtual Storage

Is a major challenge, more complex than just dealing with CPU and Memory.

Storage is critical for databases, with more complex interactions.

Tomorrow we will deal with all this in the context of databases.