Προχωρημένη Κατανεμημένη Υπολογιστική

ΗΥ623

Διδάσκων –
Δημήτριος Κατσαρός

@ Τμ. ΗΜΜΥ
Πανεπιστήμιο Θεσσαλίας

Διάλεξη 3η

Virtualization

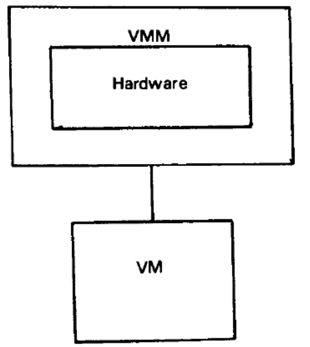
Concepts

Definitions

- Virtualization
  - A layer mapping its visible interface and resources onto the interface and resources of the underlying layer or system on which it is implemented
  - Purposes
    - Abstraction – to simplify the use of the underlying resource (e.g., by removing details of the resource's structure)
    - Replication – to create multiple instances of the resource (e.g., to simplify management or allocation)
    - Isolation – to separate the uses which clients make of the underlying resources (e.g., to improve security)
- Virtual Machine Monitor (VMM)
  - A virtualization system that partitions a single physical "machine" into multiple virtual machines.
  - Terminology
    - Host – the machine and/or software on which the VMM is implemented
    - Guest – the OS which executes under the control of the VMM

Origins - Principles

Fig. 1. The virtual machine monitor.



"an *efficient, isolated duplicate* of the real machine"

- Efficiency
  - Innocuous instructions should execute directly on the hardware
- Resource control
  - Executed programs may not affect the system resources
- Equivalence
  - The behavior of a program executing under the VMM should be the same as if the program were executed directly on the hardware (except possibly for timing and resource availability)

Formal Requirements for Virtualizable Third Generation Architectures

Gerald J. Popek
University of California, Los Angeles
and
Robert P. Goldberg
Honeywell Information Systems and
Harvard University

Virtual machine systems have been implemented on a limited number of third generation computer systems, e.g. CP-67 on the IBM 360/67. From previous empirical studies, it is known that certain third generation computer systems, e.g. the DEC PDP-10, cannot support a virtual machine system. In this paper, model of a third-generation-like computer system is developed. Formal techniques are used to derive precise sufficient conditions to test whether such an architecture can support virtual machines.

Communications of the ACM, vol 17, no 7, 1974, pp.412-421

Origins - Principles

Instruction types

- Privileged
  an instruction traps in unprivileged (user) mode but not in privileged (supervisor) mode.
- Sensitive
  - Control sensitive –
    attempts to change the memory allocation or privilege mode
  - Behavior sensitive
    - Location sensitive – execution behavior depends on location in memory
    - Mode sensitive – execution behavior depends on the privilege mode
- Innocuous – an instruction that is not sensitive

Theorem

For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

Signficance

The IA-32/x86 architecture is not virtualizable.

Origins - Technology

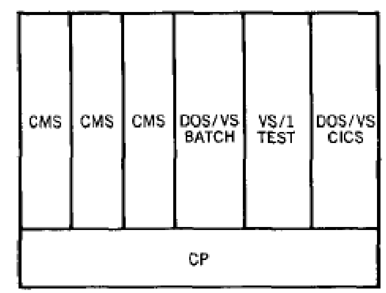
VM/370—a study of multiplicity and usefulness

by L. H. Seawright and R. A. MacKinnon

The productivity of data processing professionals and other professionals can be enhanced through the use of interactive and time-sharing systems. Similarly, system programmers can benefit from the use of system testing tools. A systems solution to both areas can be the virtual machine concept, which provides multiple software replicas of real computing systems on one real processor. Each virtual machine has a full complement of input/output devices and provides functions similar to those of a real machine. One system that implements virtual machines is IBM's Virtual Machine Facility/370 (VM/370).[1]

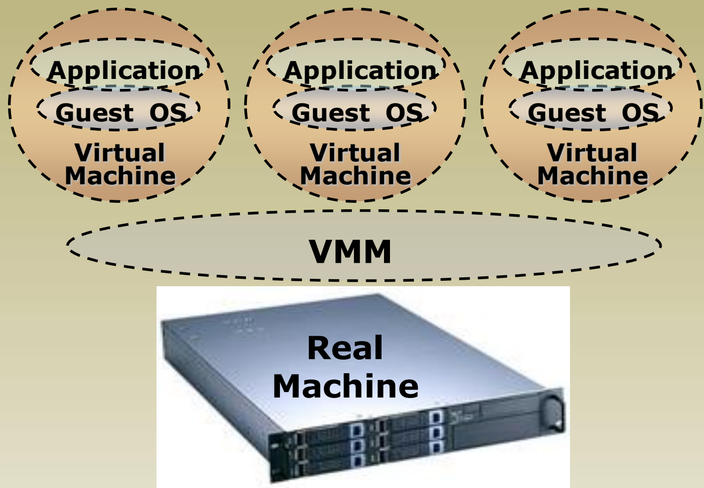
IBM Systems Journal, vol. 18, no. 1, 1979, pp. 4-17.

Figure 1 A VM/370 environment

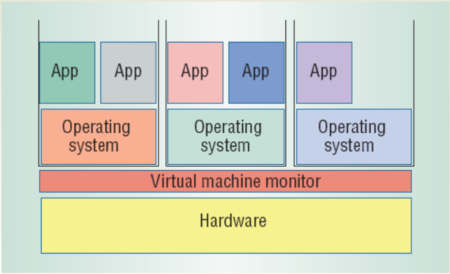


- Concurrent execution of multiple production operating systems
- Testing and development of experimental systems
- Adoption of new systems with continued use of legacy systems
- Ability to accommodate applications requiring special-purpose OS
- Introduced notions of “handshake” and “virtual-equals-real mode” to allow sharing of resource control information with CP
- Leveraged ability to co-design hardware, VMM, and guestOS

VMMs Rediscovered

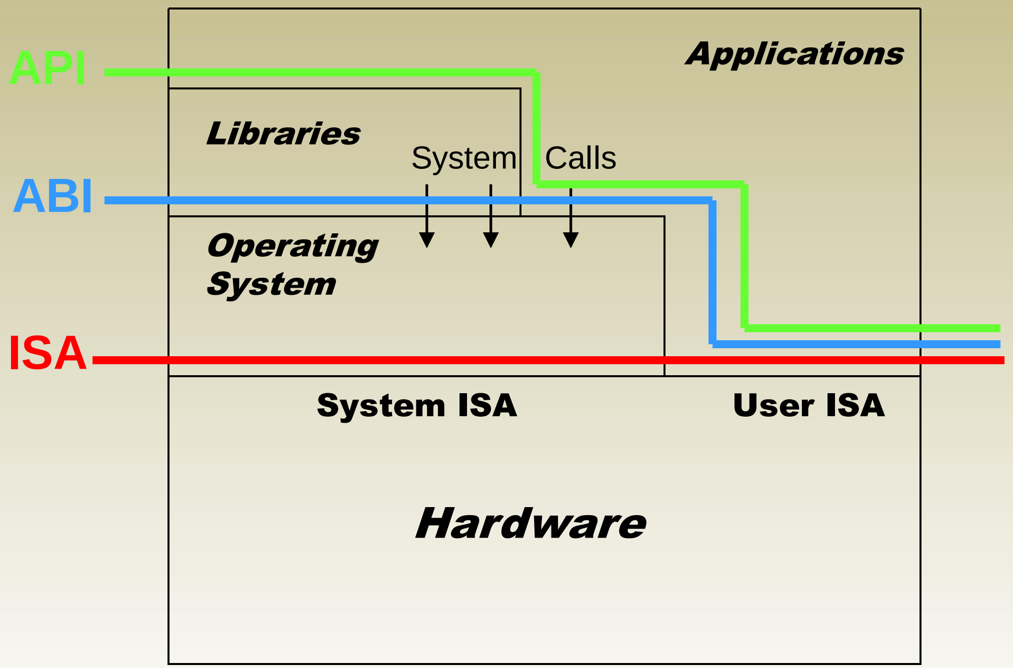


- Server/workload consolidation (reduces “server sprawl”)
- Compatible with evolving multi-core architectures
- Simplifies software distributions for complex environments
- “Whole system” (workload) migration
- Improved data-center management and efficiency
- Additional services (workload isolation) added “underneath” the OS
  - security (intrusion detection, sandboxing,…)
  - fault-tolerance (checkpointing, roll-back/recovery)



Architecture & Interfaces

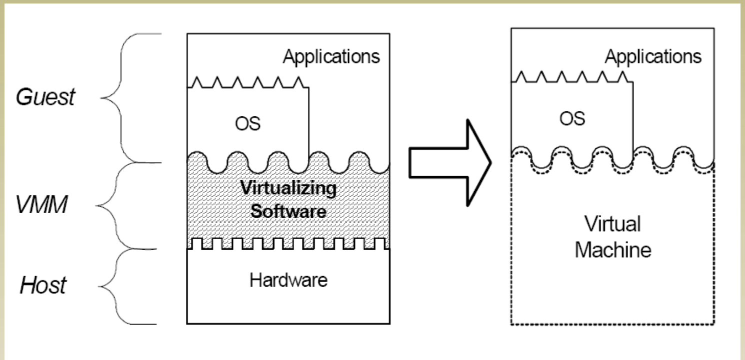
- Architecture: formal specification of a system's interface and the logical behavior of its visible resources.



- API – application binary interface
- ABI – application binary interface
- ISA – instruction set architecture

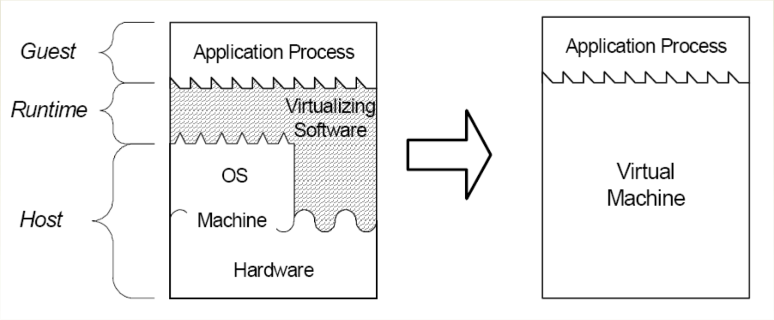
VMM Types

- System



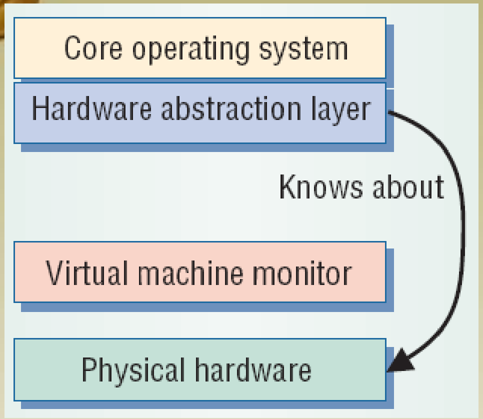
- Provides ABI interface
- Efficient execution
- Can add OS-independent services (e.g., migration, intrustion detection)

- Process

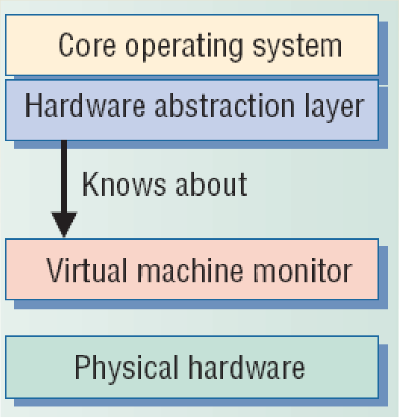


- Provdes API interface
- Easier installation
- Leverage OS services (e.g., device drivers)
- Execution overhead (possibly mitigated by just-in-time compilation)

System-level Design Approaches

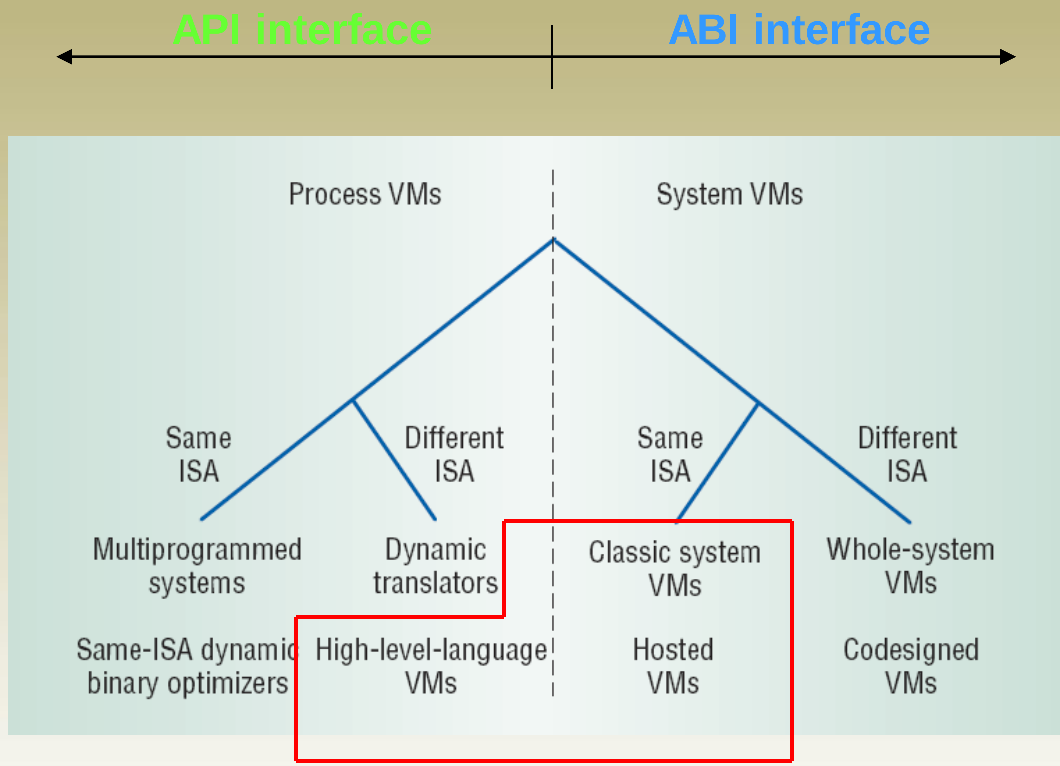


- Full virtualization (direct execution)
  - Exact hardware exposed to OS
  - Efficient execution
  - OS runs unchanged
  - Requires a "virtualizable" architecture
  - Example: VMWare



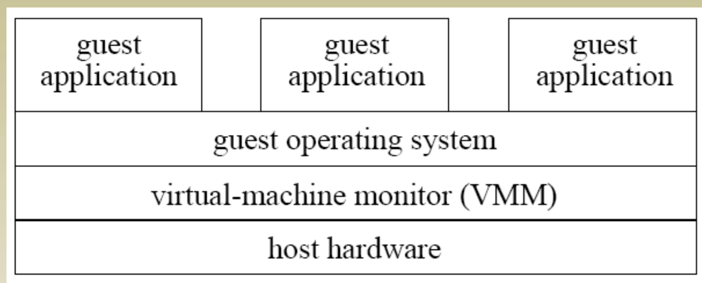
- Paravirtualization
  - OS modified to execute under VMM
  - Requires porting OS code
  - Execution overhead
  - Necessary for some (popular) architectures (e.g., x86)
  - Examples: Xen, Denali

Design Space (level vs. ISA)

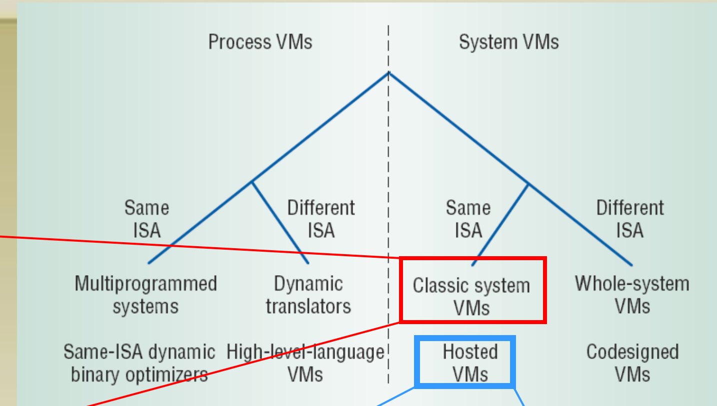


- Variety of techniques and approaches available
- Critical technology space highlighted

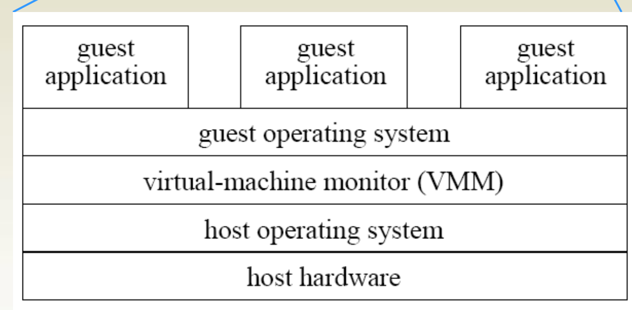
System VMMs



Type 1



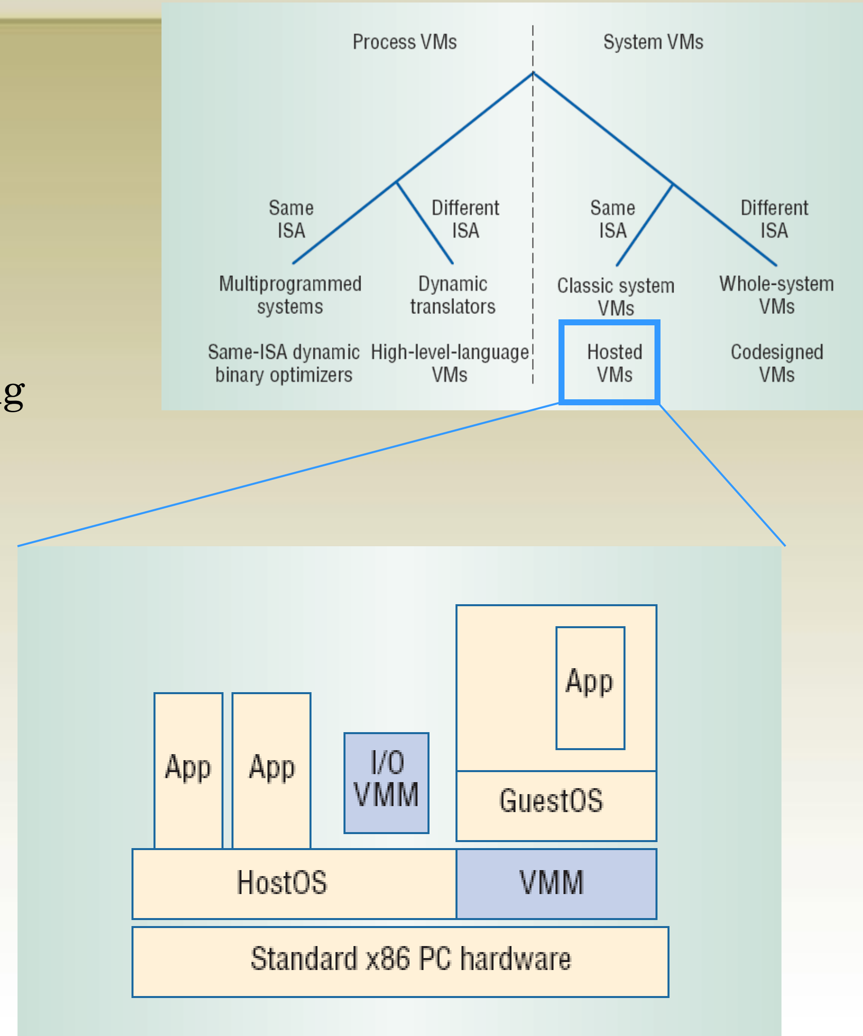
- Structure
  - Type 1: runs directly on host hardware
  - Type 2: runs on HostOS
- Primary goals
  - Type 1: High performance
  - Type 2: Ease of construction/installation/acceptability
- Examples
  - Type 1: VMWare ESX Server, Xen, OS/370
  - Type 2: User-mode Linux



Type 2

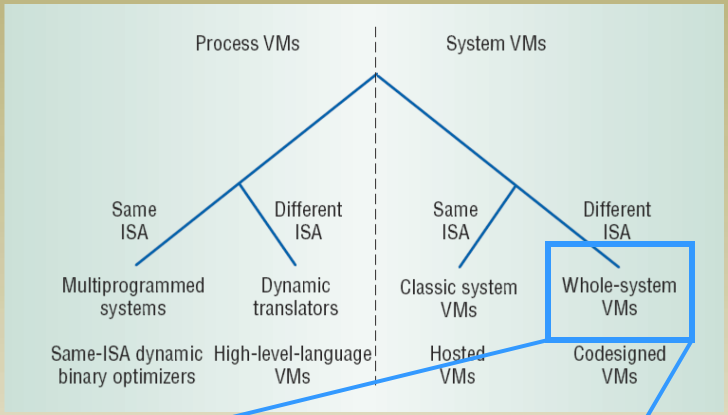
Hosted VMMs

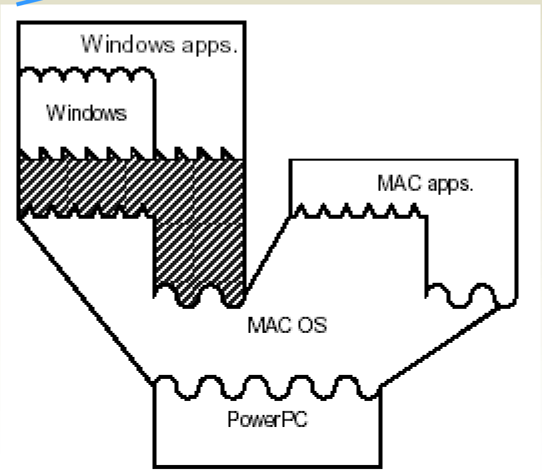
- Structure
  - Hybrid between Type1 and Type2
  - Core VMM executes directly on hardware
  - I/O services provided by code running on HostOS
- Goals
  - Improve performance overall
  - leverages I/O device support on the HostOS
- Disadvantages
  - Incurs overhead on I/O operations
  - Lacks performance isolation and performance guarantees
- Example: VMWare (Workstation)



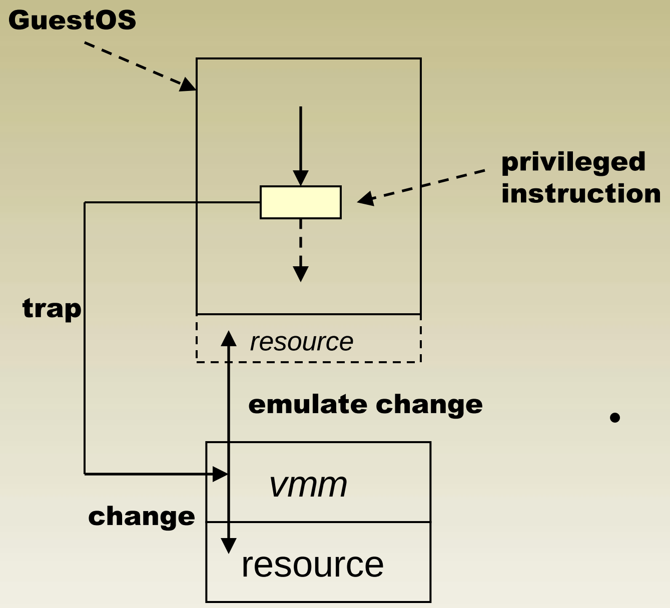
Whole-system VMMs

- Challenge: GuestOS ISA differs from HostOS ISA
- Requires full emulation of GuestOS and its applications
- Example: VirtualPC





Strategies



- De-privileging
  - VMM emulates the effect on system/hardware resources of privileged instructions whose execution traps into the VMM
  - aka trap-and-emulate
  - Typically achieved by running GuestOS at a lower hardware priority level than the VMM
  - Problematic on some architectures where privileged instructions do not trap when executed at deprivileged priority
- Primary/shadow structures
  - VMM maintains “shadow” copies of critical structures whose “primary” versions are manipulated by the GuestOS
  - e.g., page tables
  - Primary copies needed to insure correct environment visible to GuestOS
- Memory traces
  - Controlling access to memory so that the shadow and primary structure remain coherent
  - Common strategy: write-protect primary copies so that update operations cause page faults which can be caught, interpreted, and emulated.

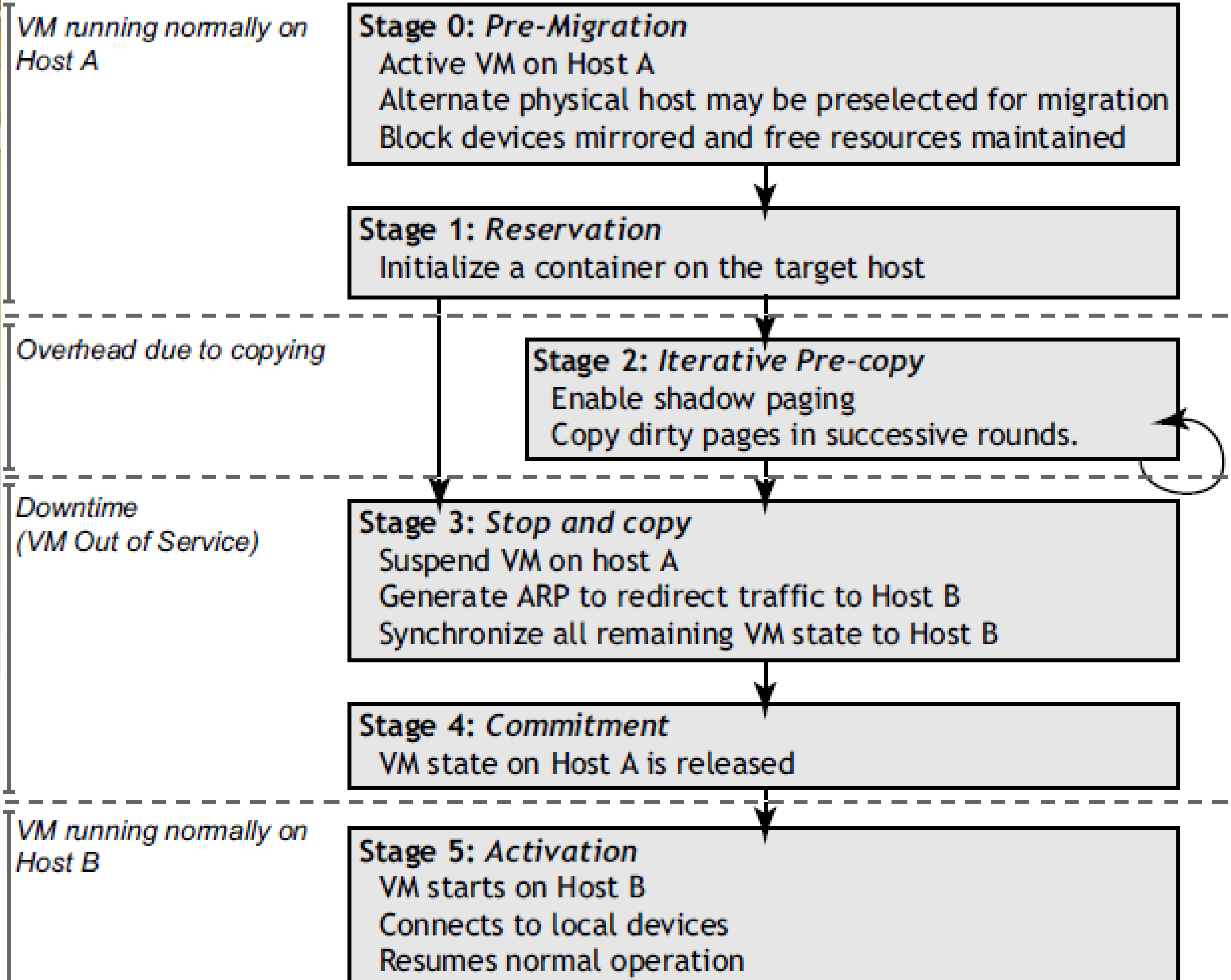
Live migration of Virtual Machines

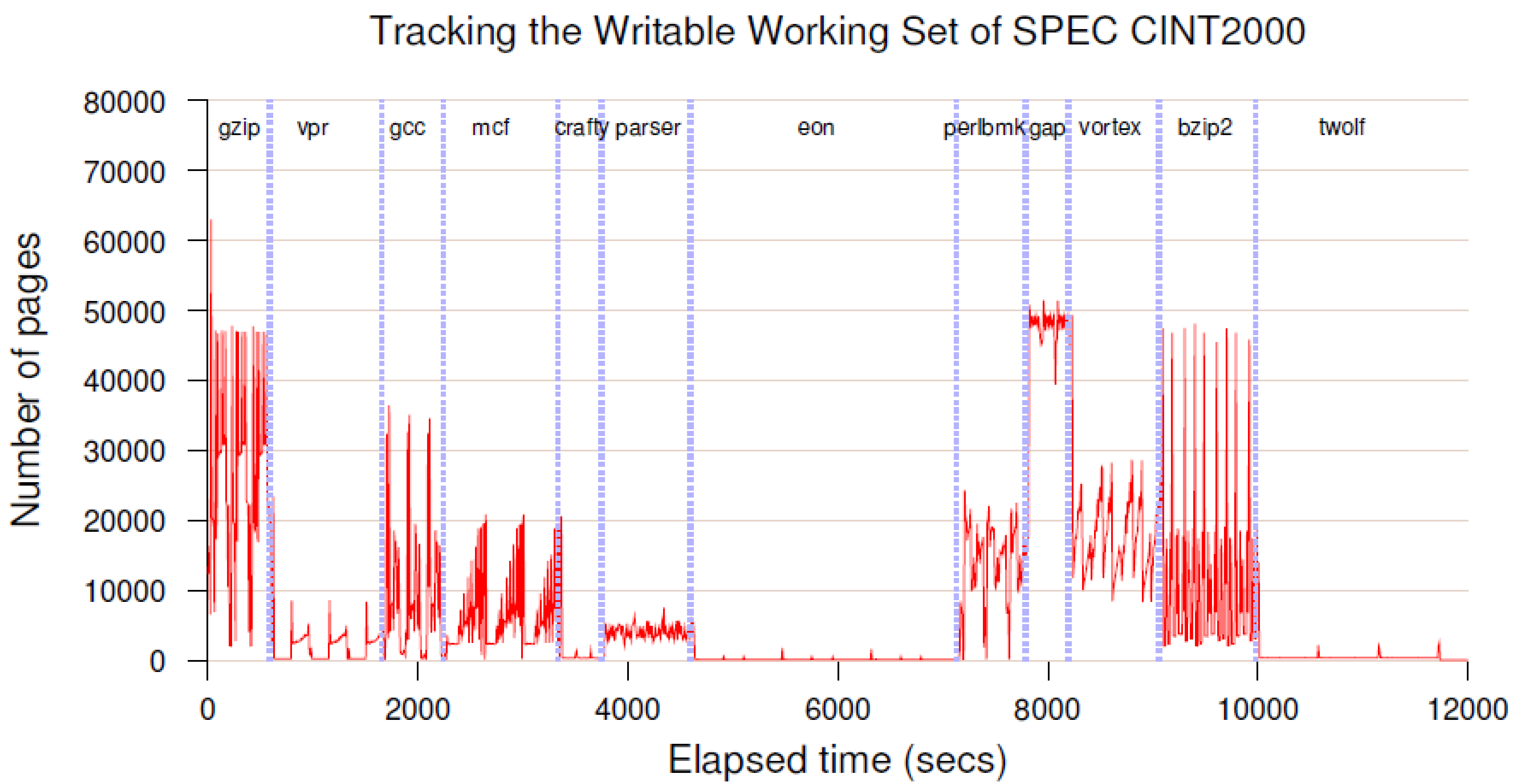
Introduction

- OS virtualization
  - Data centers
  - Cluster computing
- Live OS migration
  - Avoid problem of "residual dependencies"
  - In-memory state can be transferred in a consistent and efficient way
  - Separation of concerns between Users and Operator of a data center or cluster
  - Separation of hardware and software considerations, and consolidating clustered hardware into a single coherent management domain

Design

- Migrating memory
    - Balancing Downtime and Total migration time
  - Push phase
  - Stop-and-copy phase
  - Pull phase
- Local resources
  - Connections to local devices (disks , network interfaces)
  - Single switched LAN
  - Network-Attached Storage


VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
Resumes normal operation


Tracking the Writable Working Set of SPEC CINT2000
Number of pages
80000
70000
60000
50000
40000
30000
20000
10000
0
gzip
vpr
gcc
mcf
crafty
parser
eon
perlbmk
gap
vortex
bzip2
twolf
0
2000
4000
6000
8000
10000
12000
Elapsed time (secs)

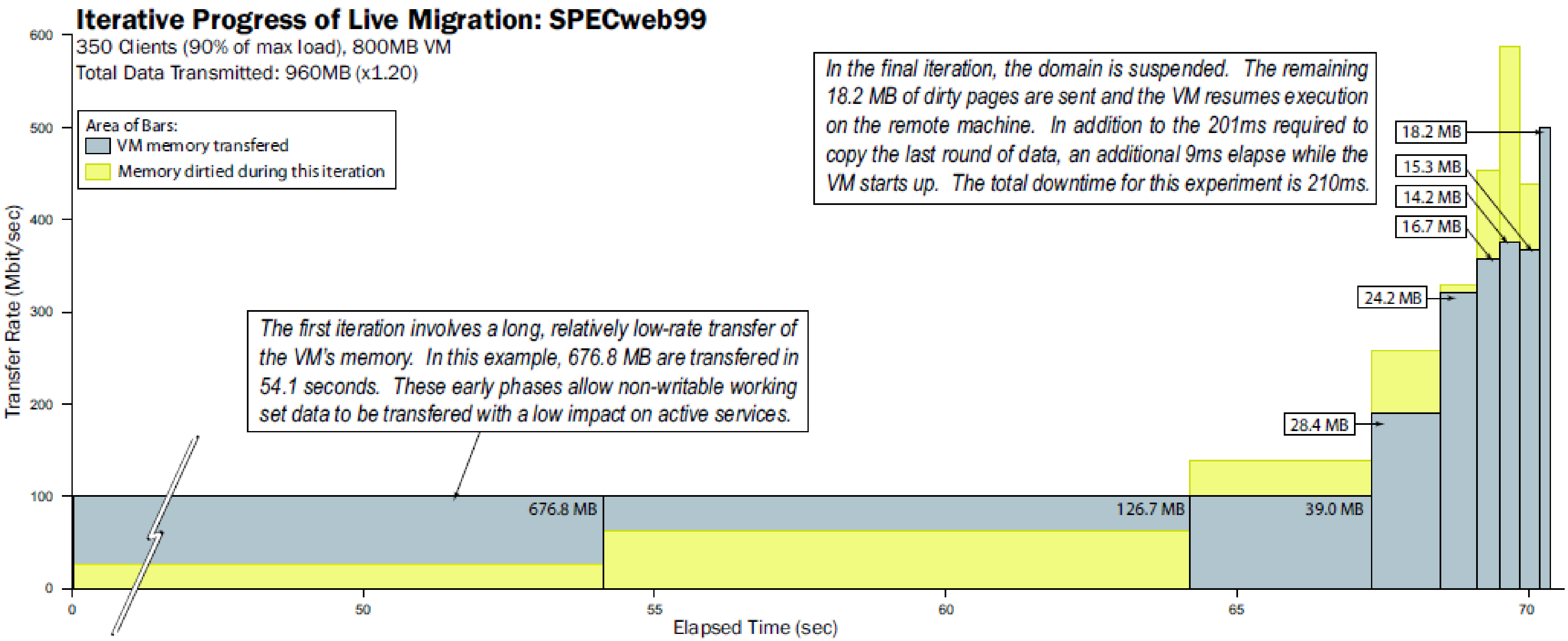


Figure 9: Results of migrating a running SPECweb VM.

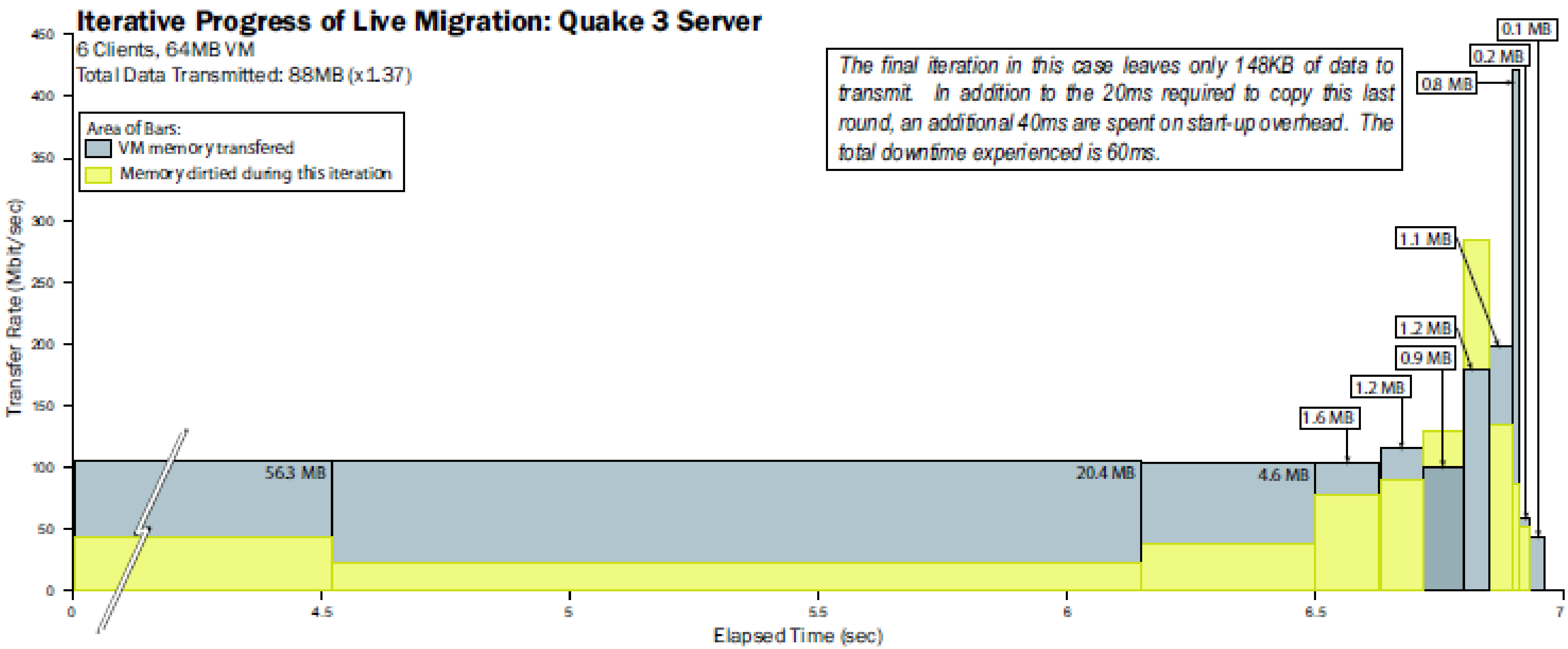


Figure 11: Results of migrating a running Quake 3 server VM.

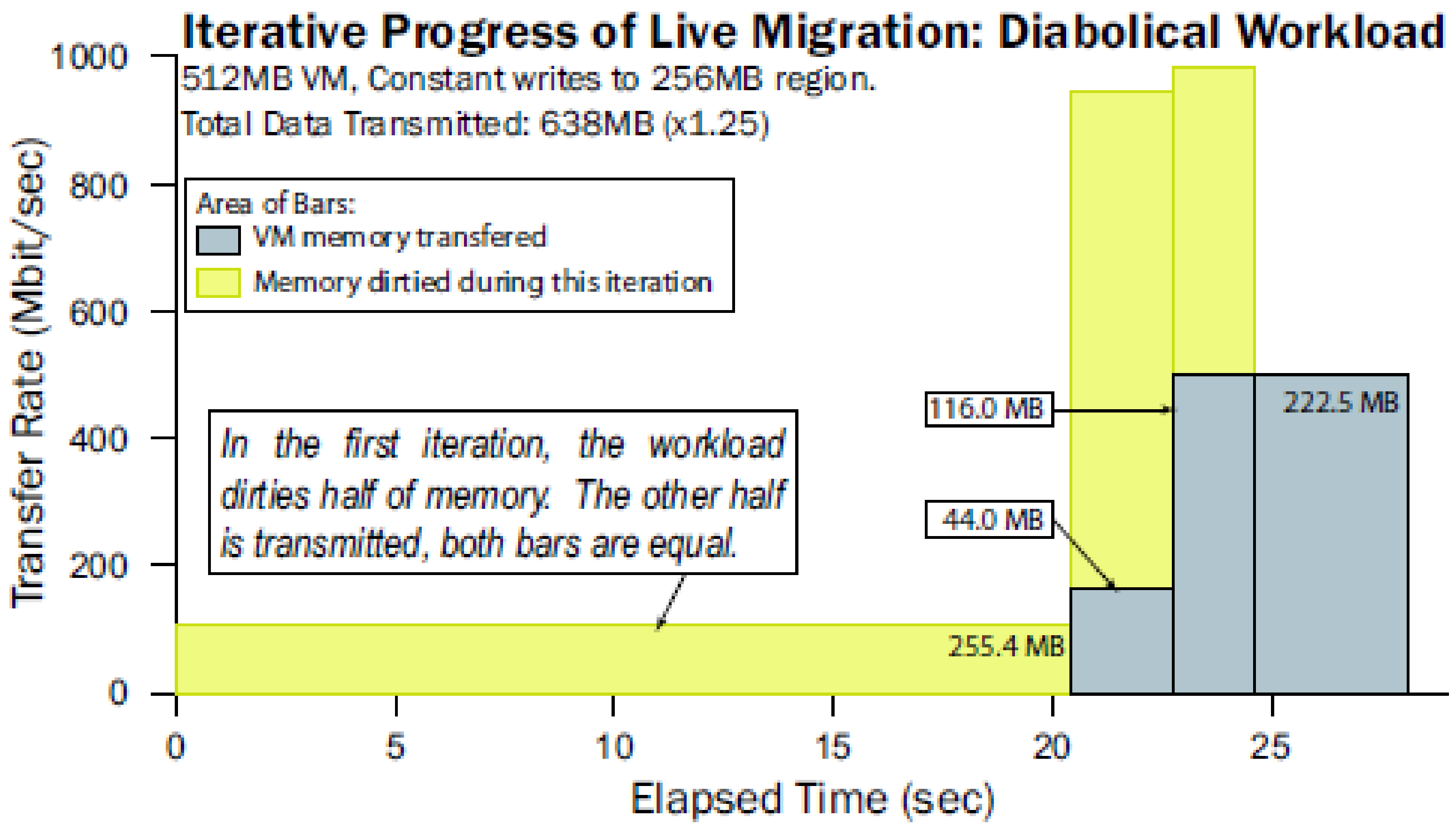


Figure 12: Results of migrating a VM running a diabolical workload.

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

The Sandpiper A…

Determine:
What virtual servers should migrate
Where to move them
How much of a resource to allocate
the virtual servers after migration

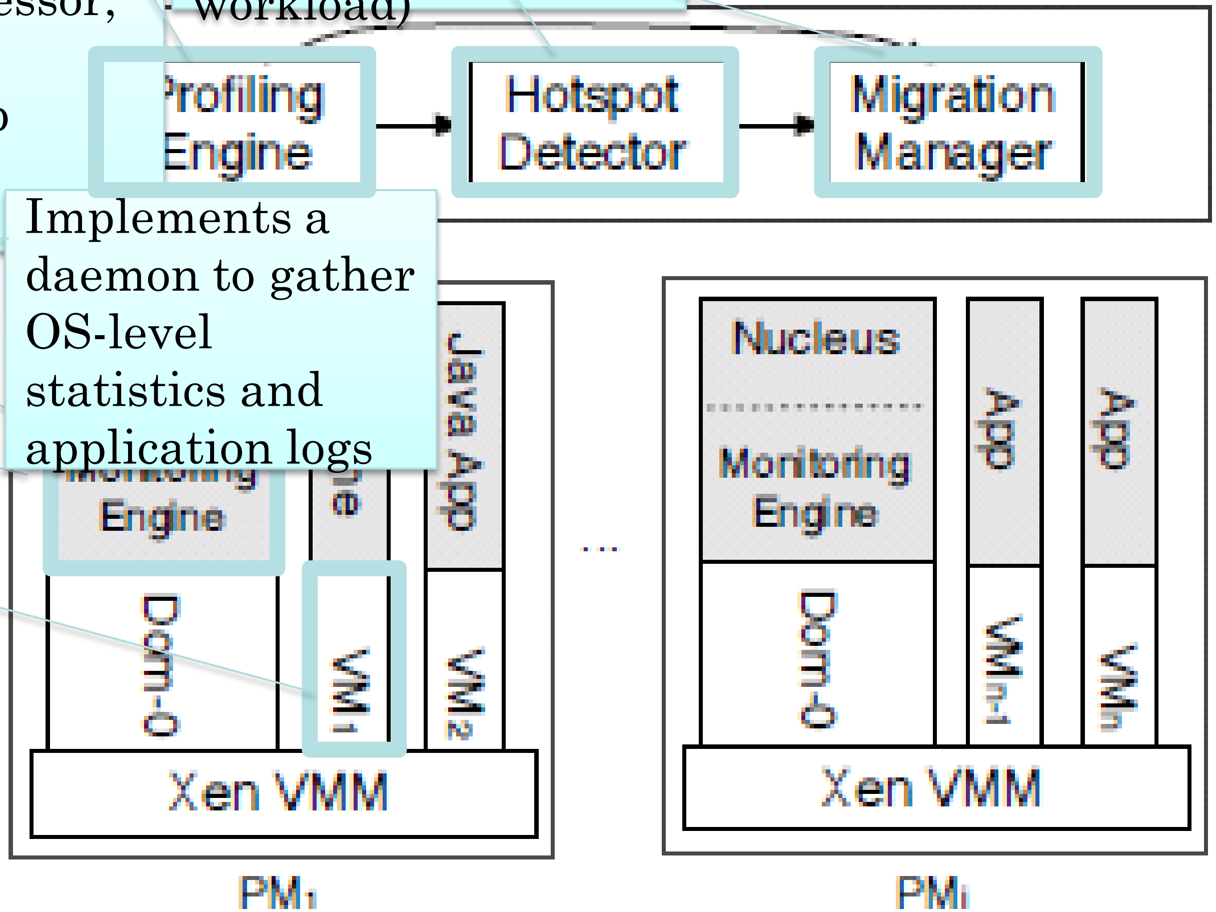
sustain period

Construct res…
usage profiles …
each virtual se…
(Predict PM
workload)

Gathering
resource
usage

Gathers processor,
network and
memory swap
statistics
for each VM

Implements a
daemon to gather
OS-level
statistics and
application logs



Profile Generation(2/2)

- Profile type:
  - Distribution profile:
    - The probability distribution of the resource usage over the window W.
  - Time series profile:
    - The temporal fluctuations and it is simply a list of all reported observations within the window W.

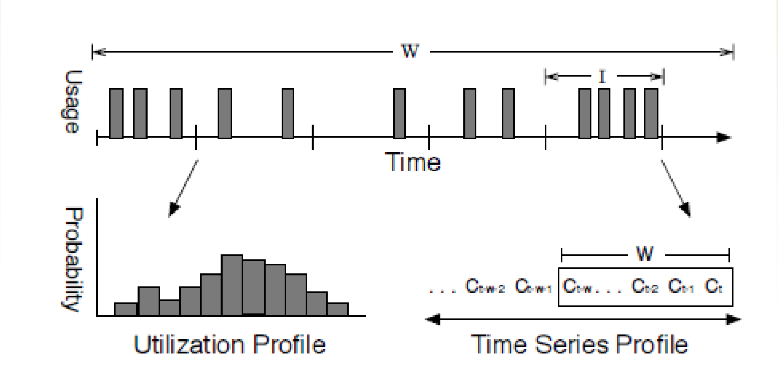


Figure 2: Profile generation in Sandpiper

Hotspot detection

- Goal:
  - Signaling a need for VM migration whenever SLA violations are detected.
- A hotspot is flagged only if thresholds or SLAs are exceeded for a sustained time.
  - at least k/n most recent observations and the next predicted value exceed a threshold.
    - Use time series profile
    - Formula: $\hat{u}_{k+1} = \mu + \phi(u_k - \mu)$
      $where\ u_k = k_{th} observation, \mu = mean\ of\ the\ time\ series,$
      $\emptyset = variration$ (auto-regressive family of predictors-AR(1).)

Hotspot mitigation (1/3)

- Hotspot mitigation algorithm:
  - Goal:
    - Determine which VM should be migrate to where to dissipate
  - Challenge:
    - NP-Hard – multi-dimensional bin packing problem
      - Bin=physical server, dimension=resource constraints
  - Solution:
    - A heuristic which solve:
      - Which overloaded VMs to migrate
      - Migrate to where such that migration overhead is minimized.
        - Migration overhead can not be neglect

Hotspot mitigation (2/3)

- Hotspot mitigation algorithm:
  - Intuition:
    - Move load from the most overloaded servers to the least-loaded servers,
    - minimize data copying incurred during migration
  - Volume: the degree of load along multiple dimensions in a unified fashion

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

    - where cpu, net and mem are the corresponding utilizations of that resource for the virtual or physical server

Hotspot mitigation (3/3)

- Hotspot mitigation algorithm:
  - volume-to-size ratio (VSR):
    - Volume/Size (Size=the memory size of the VM)
  - Migration decision:
    - Move highest VSR VM from the highest volume server and determines if it can be housed on the least volume physical server.
  - Swap decision (only consider 2-way swap):
    - Activate when simple migration cannot solve hotspot
    - Swap the highest VSR VM on the highest volume hotspot server with k lowest VSR VMs in lowest volume server
      - If a swap cannot be found, the next least loaded server is considered