HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Service migration using virtualization

Tiina Niklander

In AMICT 2007, Petrozavodsk

Services

- To the USER
- One or more processes / threads working together

- User need not to know
  - Exact location
  - Exact structure (central vs. distributed)

Migration

- Goal: need to move running process from one machine to another
- Idea over 20 years old
- Policy vs mechanism

- Three phases:
  - Detach from source environment
  - Transfer with context
  - Attach to destination environment

Migration

- Migration decision based on policy
- How?
  - Checkpoint – restart
  - Stop – copy – continue
- Why?
  - Load sharing
  - Load balancing
  - Availability, fault tolerance
  - Management decision (source reboot)

Traditional Migration Restrictions

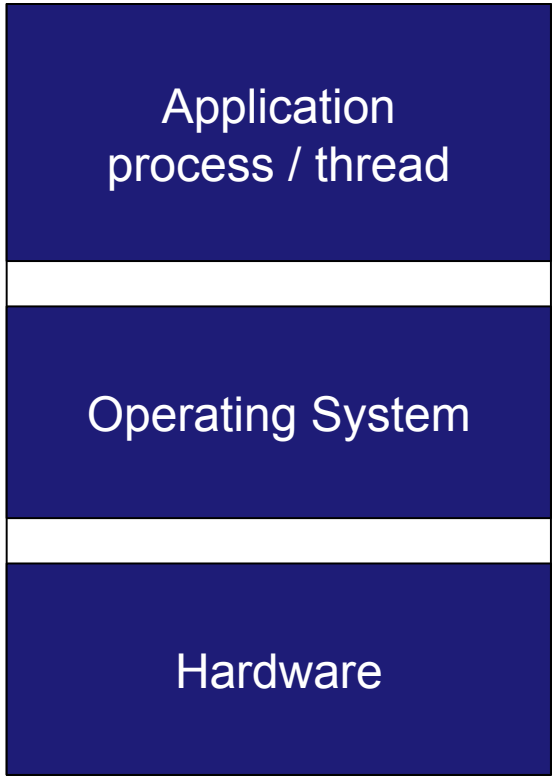
- Identical machines
  - Same machine code
  - Similar resources
  - Same OS
    - Version
    - Configuration (for example libraries)
- Goal of the restrictions:
  - Keep the running environment similar

Virtualisation goals:

- Hides hardware details
  - Gives more unified view over different hardware
- Combines application with a spesific os to a box or virtual machine
  - Allows different OS configuration for different applications on one machine
- Using emulations could even reduce the restriction of same machine code

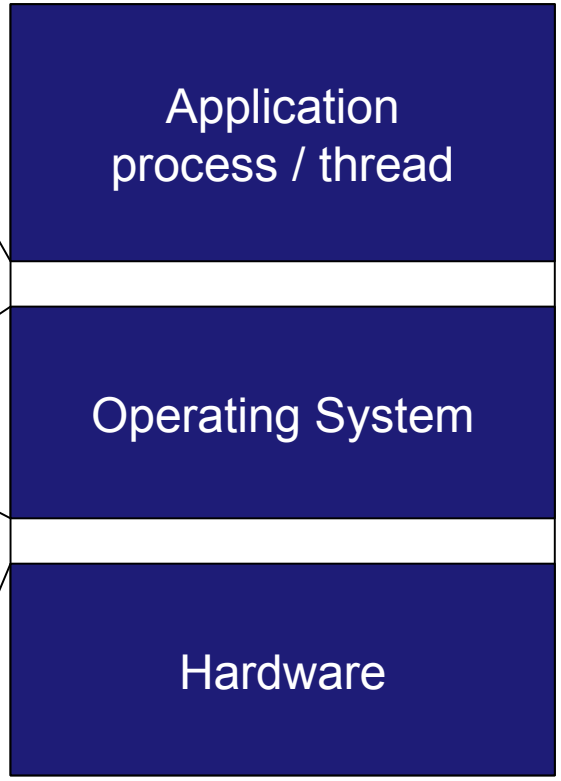
”Classical” view of structure

- Services running as user-level applications
- On top of the Operating System
- Using hardware and other resources via OS



Adding virtualization for binary-compatible applications

- Process level
  - Execute application in a box
  - Modify system calls
- System level
  - Execute OS in a box
  - Modify interrupts and traps



Process-level virtualization

- Main purposes
  - Migration, load balancing,
  - isolation of applications
- Homogeneous OS and hardware needed for migration
- Examples:
  - Zap [OSSN02]
  - vOS [BoDa02]

Migrating a virtualised process or process group (in Zap)

- Put the processes in POD with own virtualised name space
  - No resourse name conflicts
  - Mask out unallowed resources
  - Intercepts systems call tha would manipulate the resource identifiers
- Checkpoint-restart mechanism

Zap migration mechanism

- Suspend the POD
- Checkpoint the processes in POD
  - Zap virtualisation state
  - Process states: CPU registers, signal handlers, pending signals etc.
  - States of IPC mechanisms in POD
    - Data associated with external entities (such as sockets)
- Restart on the receiving machine
  - Recreate POD environment and its processes
- Uses Dynamic DNS to maintain name-to-IP

System-level virtualization

- Main purposes
  - Running multiple OS on one machine
  - Running services with different OS requirements in one machine
  - Load balancing, migration, isolation
- Examples:
  - Xen
  - VMware

System-level virtualisation

- To protect the underlying machine from uncontrolled modifications by the guest OS
- De-privileging
  - To reduce the priviledge of the guest OS
- Shadow structures
  - To protect the host machine from direct guest OS accesses
    - Page tables, priviledged registers

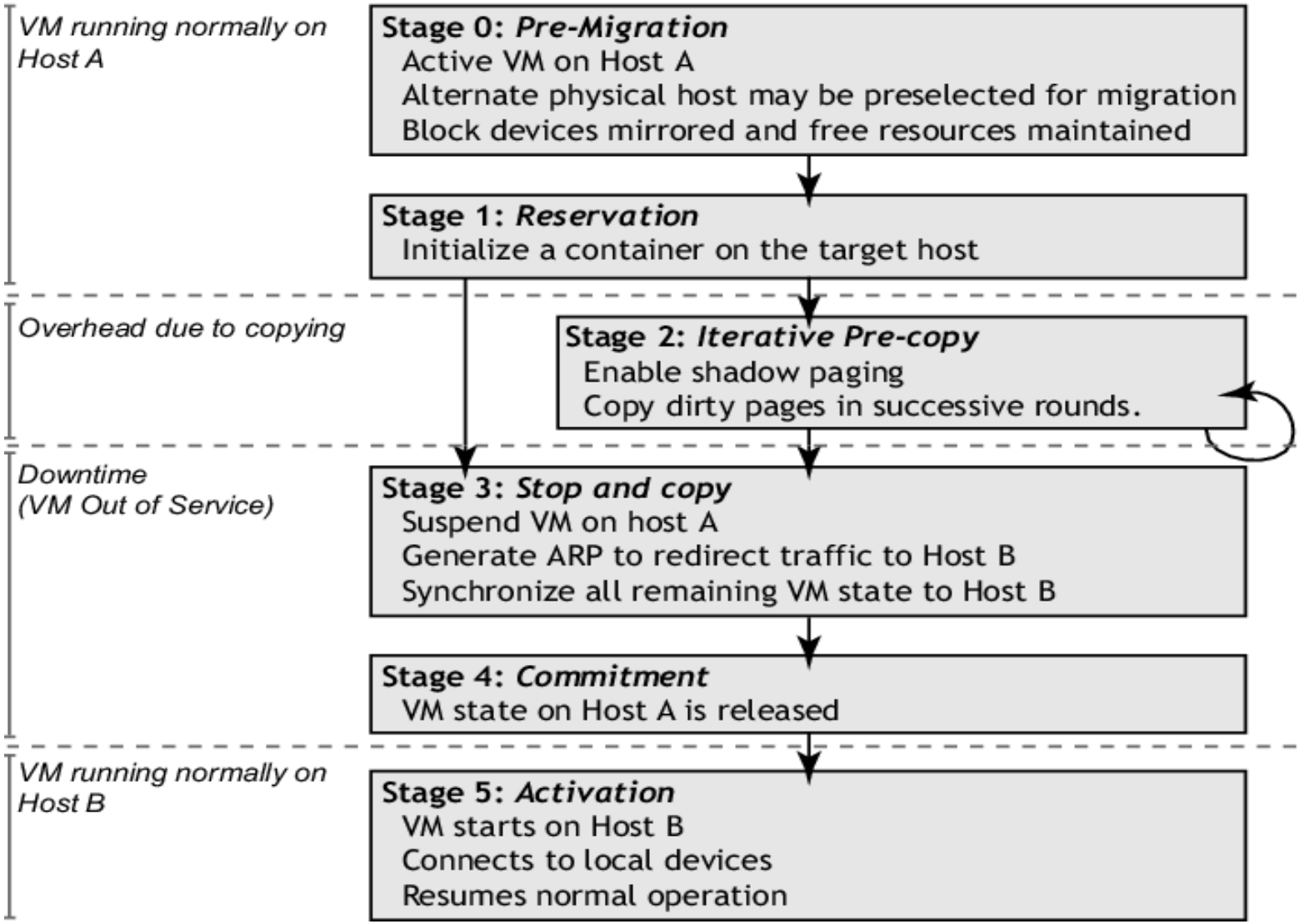
Execution of the OS in the virtual machine

- Trap-and-emulate
  - Trap all kernel mode system calls
  - Emulate their execution for the guest OS
- Replace emulation by interpretor
  - VMware: Binary translation
- Support on the machine level
  - More modes, new instructions to transfer between hypervisor and guest OS
  - IBM S370: interpretive execution mode
  - X86: guest mode

Migrating a virtual machine

- The virtual machine is already isolated from the environment
- Simple stop-copy-continue works, but has long delay
- Have to copy the whole virtual machine
  - Full memory copy

Migrating a virtual machine in Xen



Conclusion

- Migration based on existing virtualization mechanisms is feasible
- Migration of the state information still has some problems: f. ex. large file systems, networks connections.

References:

- Artsy, Y. and Finkel, R.: Designing a Process Migration Facility – The Charlotte Experience. IEEE Computer, September 1989. pp. 47-56
- Bradford, R., Kotsovinos, E., Feldmann A. and Schiöberg H.: Live Wide-Area Migration of Virtual Machines Including Local Persistent State. Proc. of VEE´07. ACM, 2007. pp. 169-179
- Potter, S. and Nieh, J.: Reducing Downtime Due to System Maintenance and Upgrades. Proc. of LISA'05. USENIX, 2005. pp. 47-62
- Osman, S., Subhraveti, D., Su, G., Nieh, J.: The Design and Implementation of Zap: A System for Migrating Computing Environments. Proc. of OSDI 2002. USENIX, 2002.

References:

- Boyd, T. and Dasgupta, P.: Process Migration: A Generalized Approach Using a Virtualizing Operating System. Proc of ICDCS'02. IEEE, 2002.
- Vasudevan, N. and Venkatesh, P.: Design and Implementation of a Process Migration System for the Linux Environment. 3rd Intl Conf on Neural, Parallel and Scientific Computing, 2006.
- Adams, K. and Agesen O.: A Comparison of Software and Hardware Techniques for x86 Virtualization. Proc of ASPLOS'06. ACM, 2006. pp. 2 – 13.
- Barham, P. et.al.: Xen and the Art of Virtualization. Proc of SOSP´03. ACM, 2003. pp. 164-177.
- Clark, C. et.al.: Live Migration of Virtual Machines. Proc of NSDI'05. USENIX, 2005. pp. 273-286