A System for Migrating Computing Environments

Zap

Steven Osman
Dinesh Subhraveti
Gong Su
Jason Nieh
Benefits of Migration

- Dynamic Load Balancing
- Mobility
- Data Access Locality
- Improved Administration
- Fault Resilience
Clustered System Approach

- Single system image across a cluster
- Good for load-balancing
  Examples include, MOSIX, Sprite
- May leave dependency on previous host
- May be new operating system or significant kernel changes
Middleware/Language Approach

• Object-based approach using special programming language or middleware
  Examples include, Abacus, Emerald, Globus, Legion, Rover

• Requires applications to be rewritten
User-level Approach

- No operating system changes
- Good for long-running applications
  Examples include, Condor, CoCheck, Libckpt, MPVM

- Does not support many common operating system services
Virtual Machine Monitor Approach

- Support any operating system
- No application changes

Example, using VMware for migration

- Must migrate the whole operating system
- Potentially higher overhead
Introducing Zap

- Transparent migration
- Unmodified legacy applications
- Networked applications
- Commodity operating system
- Minimal operating system changes
- Leaves nothing behind
- Low overhead
Outline

• Background & Motivation
• Difficulties of Migration
• Zap components
  ● Virtualization
  ● Migration
• Experimental Results
• Conclusion
int iChildPID;

if (iChildPID=fork()) {
    /* parent does some work */
    waitpid(iChildPID);
} else {
    /* child does some work */
    exit(0);
}
Resource Consistency Problem

Host A

PID 10

PID 20

Host B

PID 30

PID 40

Parent invoked waitpid(20)
Resource Conflict Problem

Host A

PID 10

PID 20

Host B

PID 10

PID 20

Resources May Conflict With Other Processes
Resource Dependency Problem

Parent and child depend on each other
Problem Recap

Resource consistency
- Names can’t change

Resource conflict
- Names can’t be duplicates

Resource dependency
- Migration must be complete
Solution

- Group processes into a POD (Process Domain) that has a *private virtual* namespace
- PODs can contain one process, one group of processes, or a whole user session
- PODs are migrated as a unit
- Solves
  - Resource consistency problem
  - Resource conflict problem
  - Resource dependency problem
Zap Architecture

Zap combines
- Thin virtualization layer
- Checkpoint/restart mechanism

Checkpoint/restart offers:
- Easier to implement than demand paging
- Leaves nothing behind
- Suspend sessions
- Easily configure and clone environments
- Dynamic system configuration
What Should Zap Virtualize?

- Process identifiers (PID)s
- Inter-process communication (IPC) keys
- Memory
- File system structure
- Network connections
- Devices
PID and IPC Key Virtualization & Migration

- Create unique namespace for the POD
- Names are virtualized
- When entering a system call, replace POD virtual identifiers with real ones
- When exiting a system call, replace real return values with POD virtual ones
- Mask out identifiers that do not belong to the POD
Memory Virtualization & Migration

- Like IPC, create unique shared memory namespace
- Modern architectures support virtual memory

Thank you modern architectures!

Migration optimization: Move only data pages, code pages can be remapped
File System Virtualization & Migration

- Some filenames can’t conflict: 
  /var/run/httpd.pid

- Some directories have unique configuration: 
  /etc

- Some directories depend on the current processes 
  /proc
File System Virtualization & Migration

- Create a directory structure for POD
- Use network file systems
- Create private POD directories
  - Good for /tmp, /var & POD specific configuration
- Private /proc directory
- Private /dev directory
Use chroot() to map POD root directory
Networking Virtualization & Migration

• Two network addresses:
  ● Persistent internal address
  ● Host-dependent external address

• For connection migration:
  ● Transport layer sees virtual address
  ● Network layer sees real address
  ● Transport layer independent
  ● Initial virtual address is real address
Virtual Networking

Application

Transport ADDR. 1

1 → 2

= 3 → 2

Network ADDR. 3

1 → 2

= 2 → 3

Network ADDR. 2

Transport ADDR. 2

2 → 1

= 2 → 3

Application
Device Virtualization & Migration

Device migration is hard

- Pseudo Terminal
- Sound Device
- CDRW During a Recording Session
- Electron Microscope
Device Migration & Virtualization

Pseudo Terminal → Virtual device configuration+data

Sound Device → Virtual device configuration

Recording CDRW → Migrate later

Electron Microscope → Communicate with original host
Device Migration & Virtualization

Unsupported devices do not appear in a POD’s /dev

Zap currently supports pseudo terminals, ensuring their names are consistent after migration (e.g. /dev/pts/2)
Zap Implementation

- Developed for Linux 2.4
- Zap design enables
  - Loadable kernel module
  - No need to rebuild the kernel
- Intercept system calls for virtualization
Zap Implementation

User space

User Processes

ZAP Virtualization

System Calls

Kernel

Zap Migration

kernel space
Virtualization Cost

- Created micro-benchmarks
  - PID calls (getpid)
  - IPC calls (shmget/ctl, semget/ctl)
  - Process creation calls (fork, execve, exit)
- Used macro-benchmarks
  - Apache
  - Build Linux kernel
  - Volano
Virtualization Results

Normalized Performance

Vanilla Linux
with POD
in POD
VMware
Virtualization Results

- Zap incurs low overhead
Migration Cost – VNC Session
Migration Cost – Apache

- Apache 2.0.35
- Default configuration
Migration Cost – Time

Latency (seconds)

VNC Checkpoint | VNC Restart | Apache Checkpoint | Apache Restart

0.963 | 1.641 | 26.824 | 0.811 | 3.117 | 3.184 | 0.373 | 2.342 | 27.396

Latency (seconds)
Migration Cost

- Zap can be fast
- <1 second checkpoint/restart times
- Includes Zap networking round-trip
Zap

- Offers transparent migration of legacy and network applications
- Introduces PODs
  - Consistency
  - Conflict free
  - Avoids Unwanted dependencies
- Leaves nothing behind
- Fast and lightweight
For more information…

• Zap computing
  http://www.ncl.cs.columbia.edu/research/migrate

• Network Computing Laboratory
  http://www.ncl.cs.columbia.edu/
Future Work

• Secure migration
  • Trusted images, POD sandbox, etc.
• Generalized device support
• Migration policies
• Heterogeneity
• Contextualization
• Resource management