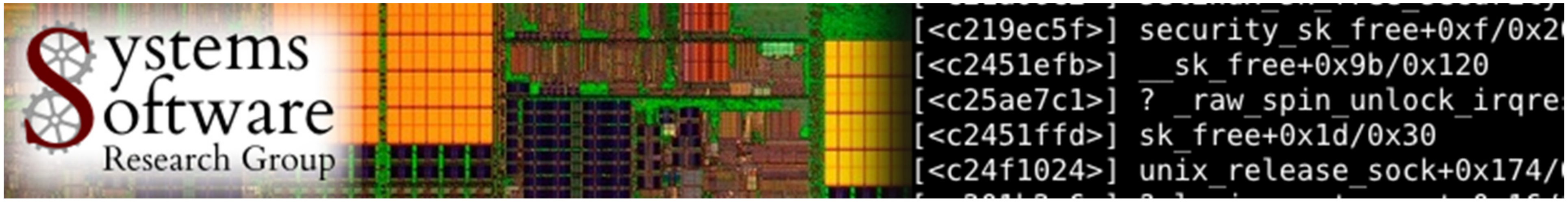


```
[<c219ec5f>] security_sk_free+0xf/0x2
[<c2451efb>] __sk_free+0x9b/0x120
[<c25ae7c1>] ? _raw_spin_unlock_irqre
[<c2451ffd>] sk_free+0x1d/0x30
[<c24f1024>] unix_release_sock+0x174/
```

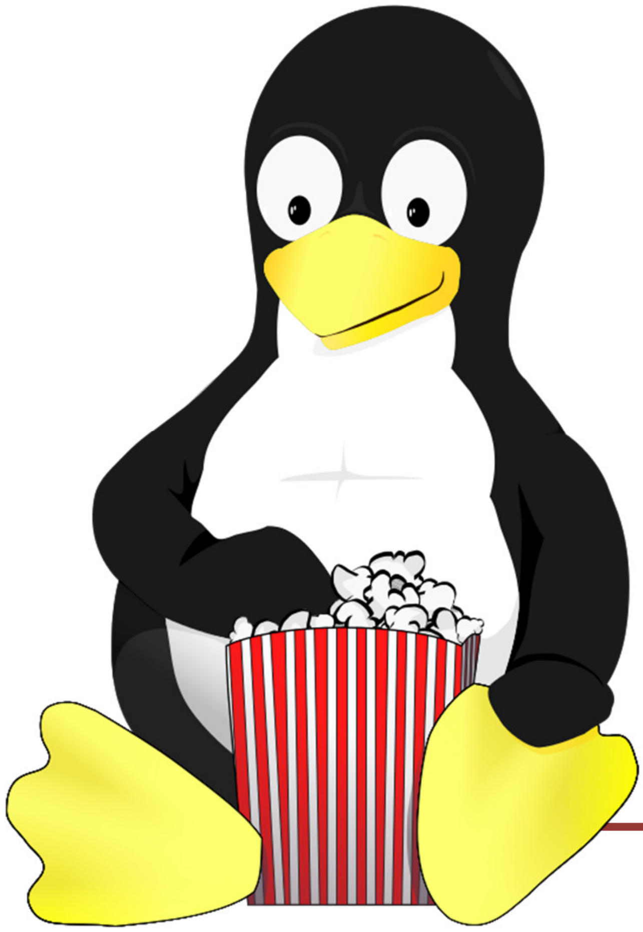
Popcorn Project: Replicated-kernel Linux

Antonio Barbalace

antoniob@vt.edu

Systems Software Research Group at Virginia Tech

http://ssrg.ece.vt.edu



ece Bradley Department of Electrical & Computer Engineering

VTLUUG meeting , May 2nd 2013

VirginiaTech Invent the Future

Popcorn Linux

- A replicated-kernel OS (multi-kernel)
- Based on Linux 3.2.14
- Supports the x86-64 arch (an x86-32 patch exists)
- Project started summer 2012
- Website: www.popcornlinux.org
- Patches available on the website
- Git repositories on *TO BE ANNOUNCED*
- Requires multiprocessor (multi-core) hardware

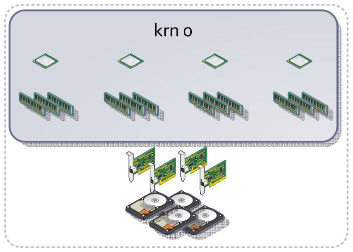
Pervasive Multiprocessor Hardware Technology

- Today's computing hardware is multiprocessor (multi-core)
  - from embedded to server-class computers and in high performance computing (HPC)
- Different processors (cores) share access to a common memory
- Various multiprocessor topologies exist
  - fat tree, mesh, torus, ..
- Linux supports multiprocessor hardware as a symmetric multiprocessing (SMP) operating system (OS)

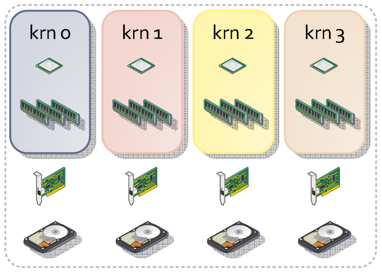
Systems Software Research Group

VirginiaTech Invent the Future

A Single Operating System (resource point of view)



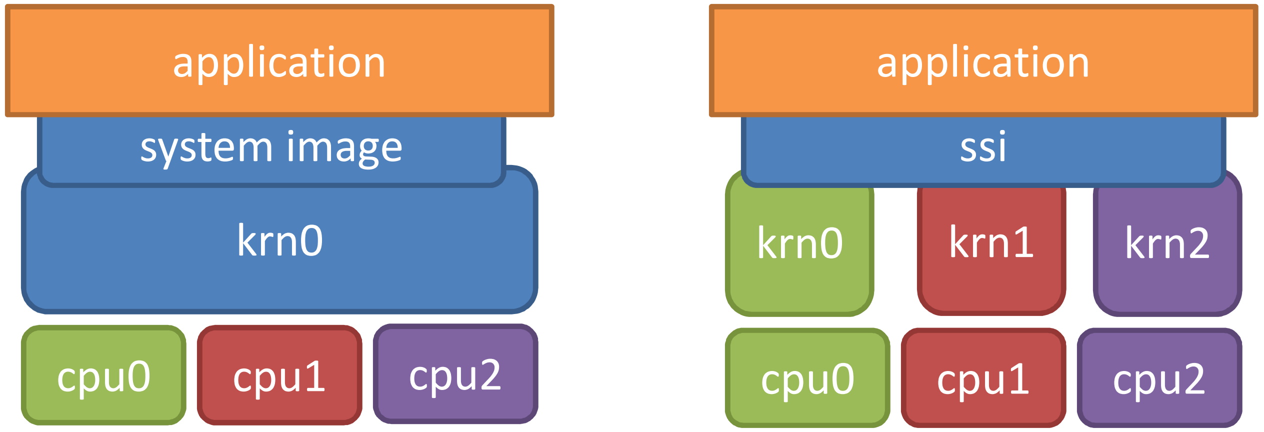
- Traditional SMP operating systems, like vanilla Linux, are single kernel image (monolithic)
  - The **kernel state** must be accessed concurrently by the different CPUs to be kept consistent
  - Performance penalties for accessing resources



- A replicated-kernel operating system is built on top of multiple instances of a kernel image
  - Every kernel maintains **its own state** concerning its own resources
  - There is no performance penalty to access local resources

A Single Operating System (application point of view)

- An application running on a traditional SMP OS is expecting a **predefined environment**
  - In order to interact with other processes
  - In order to interact with different threads
- In a replicated-kernel OS this property is guaranteed by providing the applications with a **single system image** (ssi)
  - An application written for a traditional SMP OS can run transparently on the new OS (recompilation is not required)



Do similar projects exist?

- Research projects, custom kernels
  - Hurricane, Stanford (1992)
  - Hive, Stanford (SGI IRIX UNIX mod, 1995)
  - Disco/Cellular Disco, Stanford (1997)
  - Barrelfish, ETH Zurich, MS Research (2009)
  - FOS, MIT CSAIL (2009)

virtualization

multikernel

- Previous attempts on Linux (to replace virtualization):
  - Twin Linux **(no source-code, and GPL?)**
  - Linux Mint **(no source-code again!)**
  - SHIMOS (Single Hardware with Independent Multiple Operating Systems) **(same story …)**
  - coLinux (do you really want Linux and **MS Windows** to coexist?)

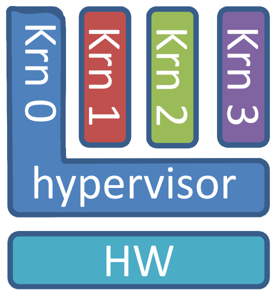
The dates reported refers to the year of publication or their representative papers, not the start of the project.

Virtualization or Replicated-kernel?

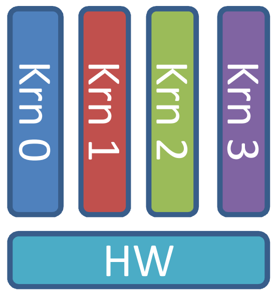
- Virtualization Technologies reborn at the end of 90s from a replicated-kernel project: Disco/Cellular Disco
- Disco/Cellular Disco was the last of a series of Operating System for emerging multiprocessor computers
  - Disco paper's title: "Running Commodity Operating Systems on Scalable multiprocessors"
  - Its predecessor, Hive, was implemented as a variant or IRIX, UNIX from SGI
  - Hive paper's title "Hive: Fault Containment for Shared-Memory Multiprocessors"
  - The predecessor of Hive was called Hurricane
- Disco/Cellular Disco evolved into a company called VMware

Peer Kernels

Virtualization



Peer kernels

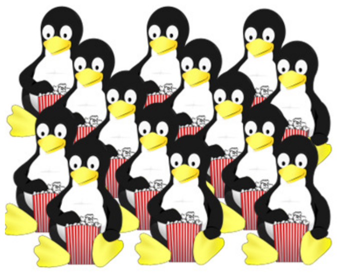


- All kernel instances are peers that reside within different resource partitions
  - This is different from virtualization where there is a **host** (privileged) and many **guest** kernels
  - Virtualization allows time-sharing resources at OS granularity (temporal and space partitioning)
- From each kernel the user can:
  - Boot any other kernel
  - Run any service
  - Control any of the hardware devices, if they are included in the kernel's resource partition

Multikernel or Replicated-kernel?

Multikernel

Replicated-kernel



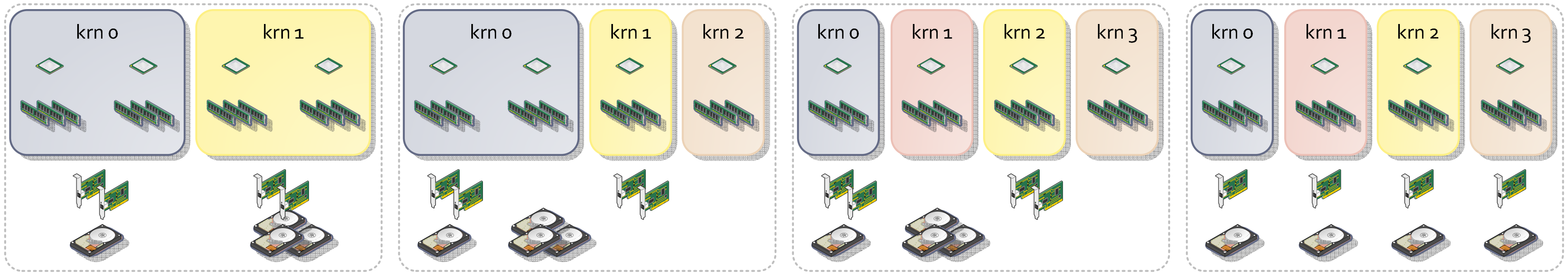
- Barrelfish paper: "The Multikernel: A new OS architecture for scalable multicore systems"
  - A multikernel OS was defined around the following principles:
    - Make all inter-core communication explicit
    - Make OS structure hardware-neutral
    - View state as replicated instead of shared
- The replicated-kernel (multi-kernel) OS architecture mixes concepts from the multikernel and distributed OS designs
  - In a replicated-kernel (similar to a multikernel):
    - Inter-kernel communication enables OS state consistency (a kernel can run on any subsets of cores)
    - Applications developed for a SMP OS transparently run on a replicated-kernel
    - Replication can also exploited at application level (heterogeneous ISA)
  - We believe that the replicated-kernel model can be applied to any traditional OS without ground-up redesign (like distributed OS)

Popcorn can be used as

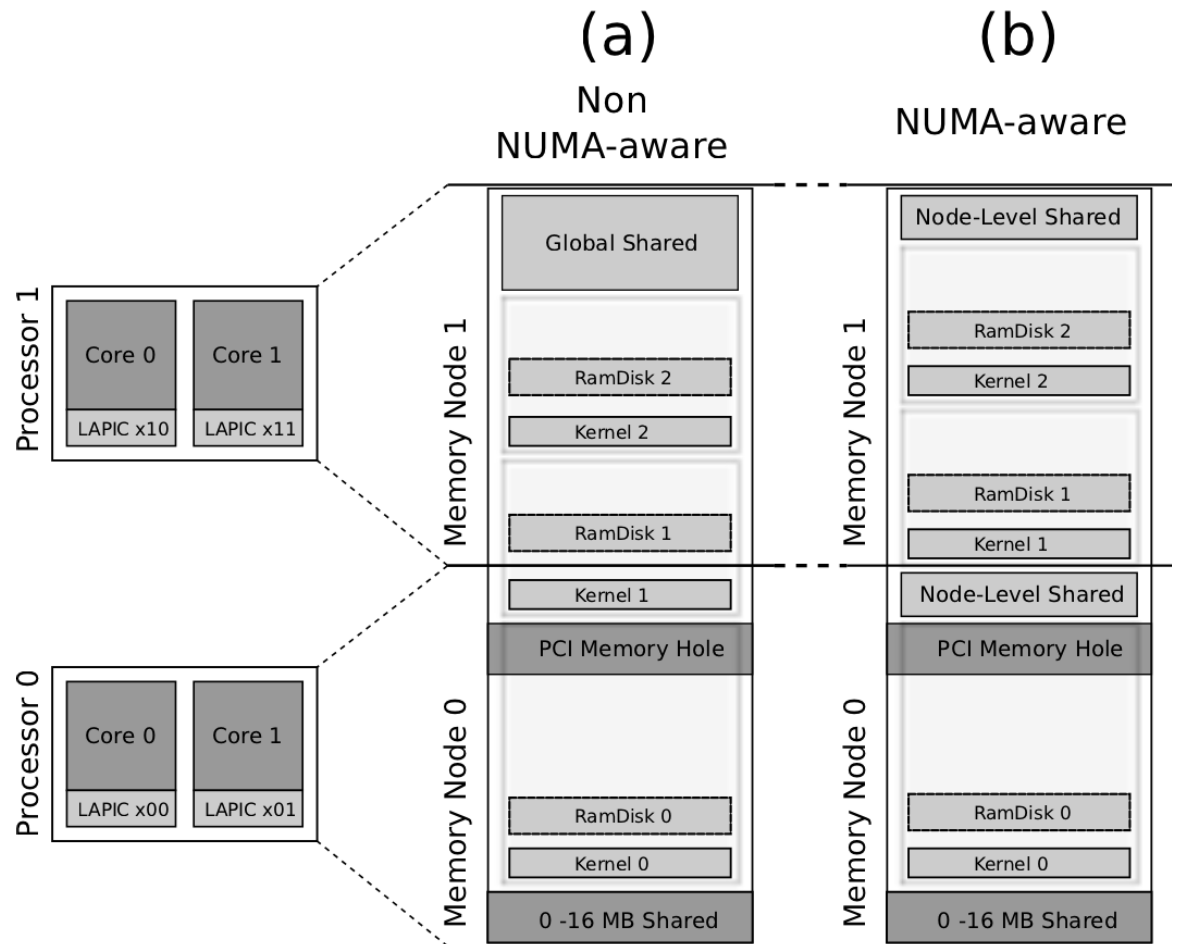
- A **single operating system** made up of multiple Linux kernels (replicated-kernel)
  - kernels are peers and must communicate and keep their state coherent
  - applications can run over all the hardware
- An **alternative to virtual machines** (federation of SMP OS, we compared with KVM)
  - kernels are independent from one another
  - applications can run on the resource partition assigned to that kernel

Abundant Resources

- Partitioning and clustering of hardware resources
- Static and dynamic resource partitioning and clustering



Example: Physical Memory Partitioning



- Basic approach: consider the whole memory as a single chunk that is globally accessible with the same latency cost (not NUMA-aware)
- NUMA-aware: the entire memory is subdivided between kernels based on the available memory and cores resident on each node. The system topology is given the greatest weight in deciding how to allocate private memory windows and shared memory areas

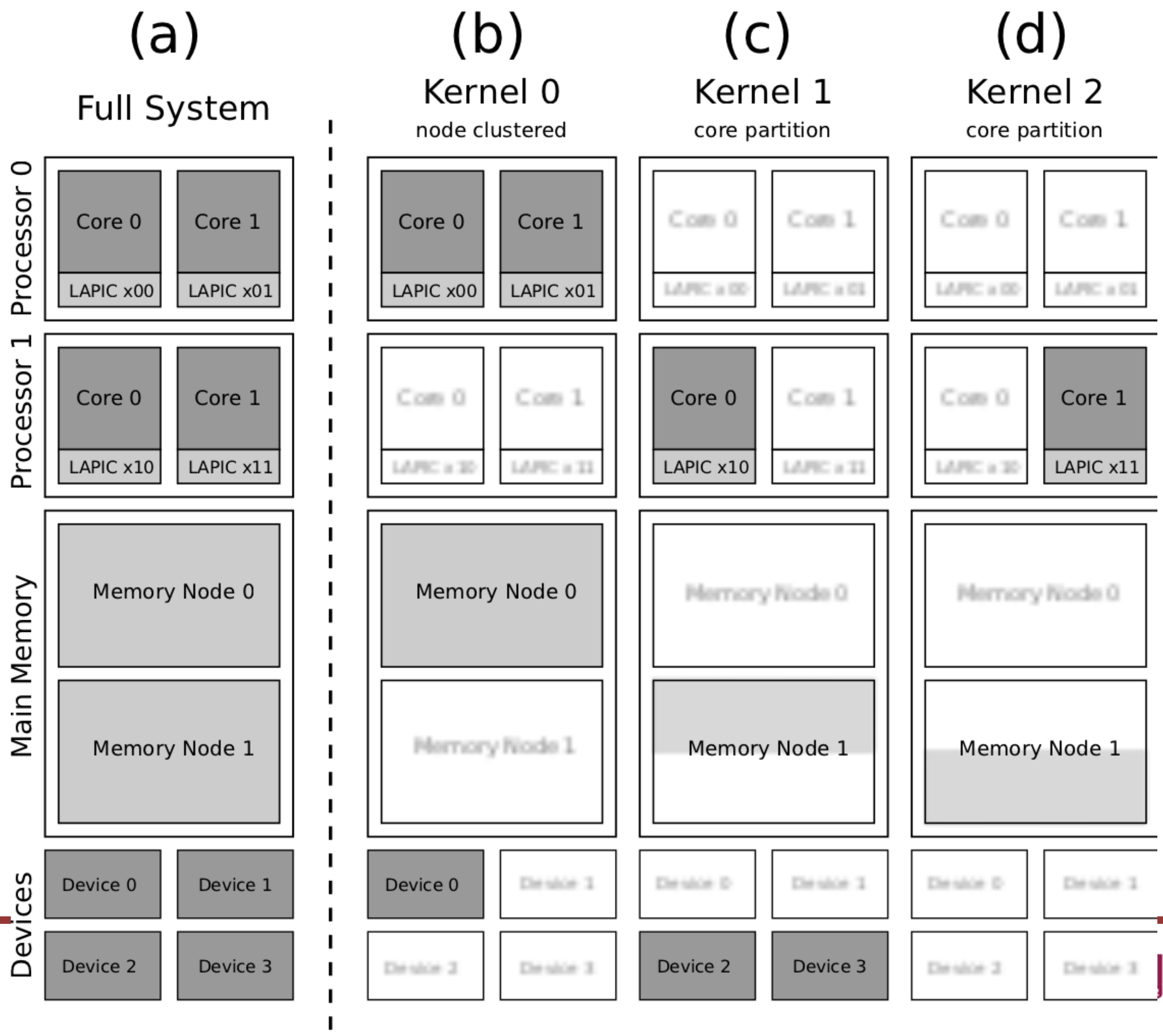
Systems Software Research Group

VirginiaTech Invent the Future

Exclusive Resources

- Not all hardware resources can be partitioned (or clustered) easily
- Hardware peripherals are usually **limited** in number and can not easily shared
- **Master/worker model** to access exclusive resources (plus messages)
- Alternative: exploit virtualization hardware solutions

Resource Subdivision Example



Systems Software Research Group

VirginiaTech Invent the Future

Replicated-kernel: Messaging

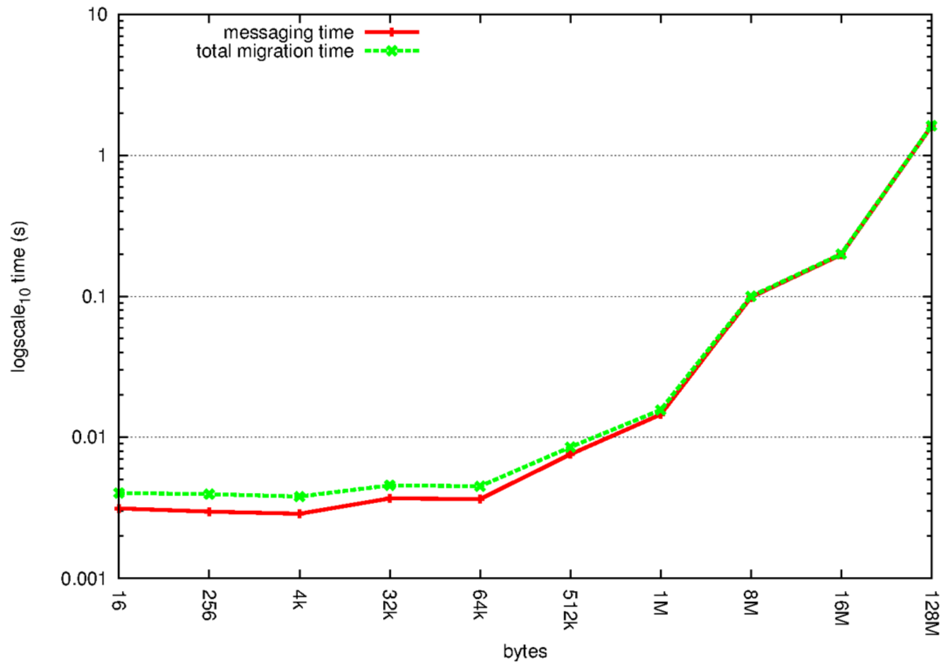
- Barrelfish and the microkernel community: "Communicating by message passing in a OS is not more expensive than using shared memory"
- In an SMP OS there is a single kernel status that is kept coherent using **shared data structures** with locking
- In a replicated-kernel, different kernel instances are kept coherent by means of **messages**
- This approach maps well to heterogeneous systems
  - Explicit messaging has the advantage of being easily marshaled or converted
  - Messages can be sent on shared memory or via any available hardware messaging peripherals

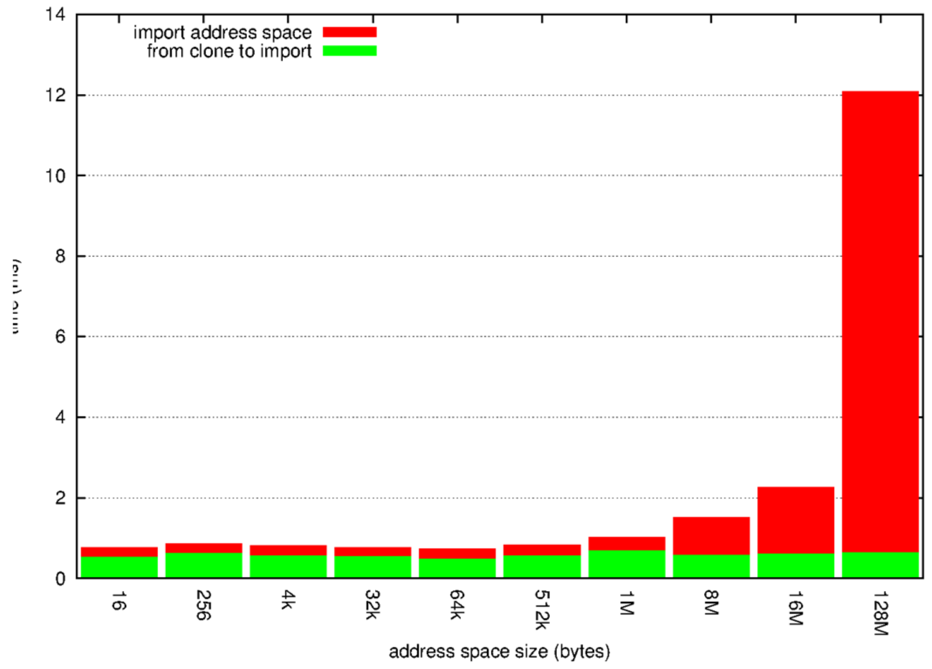
Inter-Kernel Process Migration

- Based on (the first release of) inter-kernel messaging
- The address space content of the application is at the **same physical address**
  - there is one copy of the memory of the process
  - all of the virtual mapping (**`vm_area_struct`** items) is copied from the source to the remote kernel
  - all of the page global directory (**`mm->pgd`**) mappings, with their flags, is copied
- On any kernel a **process server** is loaded at initialization to accept migrating processes

Results: Process Migration

- Initial results (25th March 2013) on process migration
- In Linux, **`sched_setaffinity()`** moves a process between cores in 600us on average (64x AMD Opteron 6274)

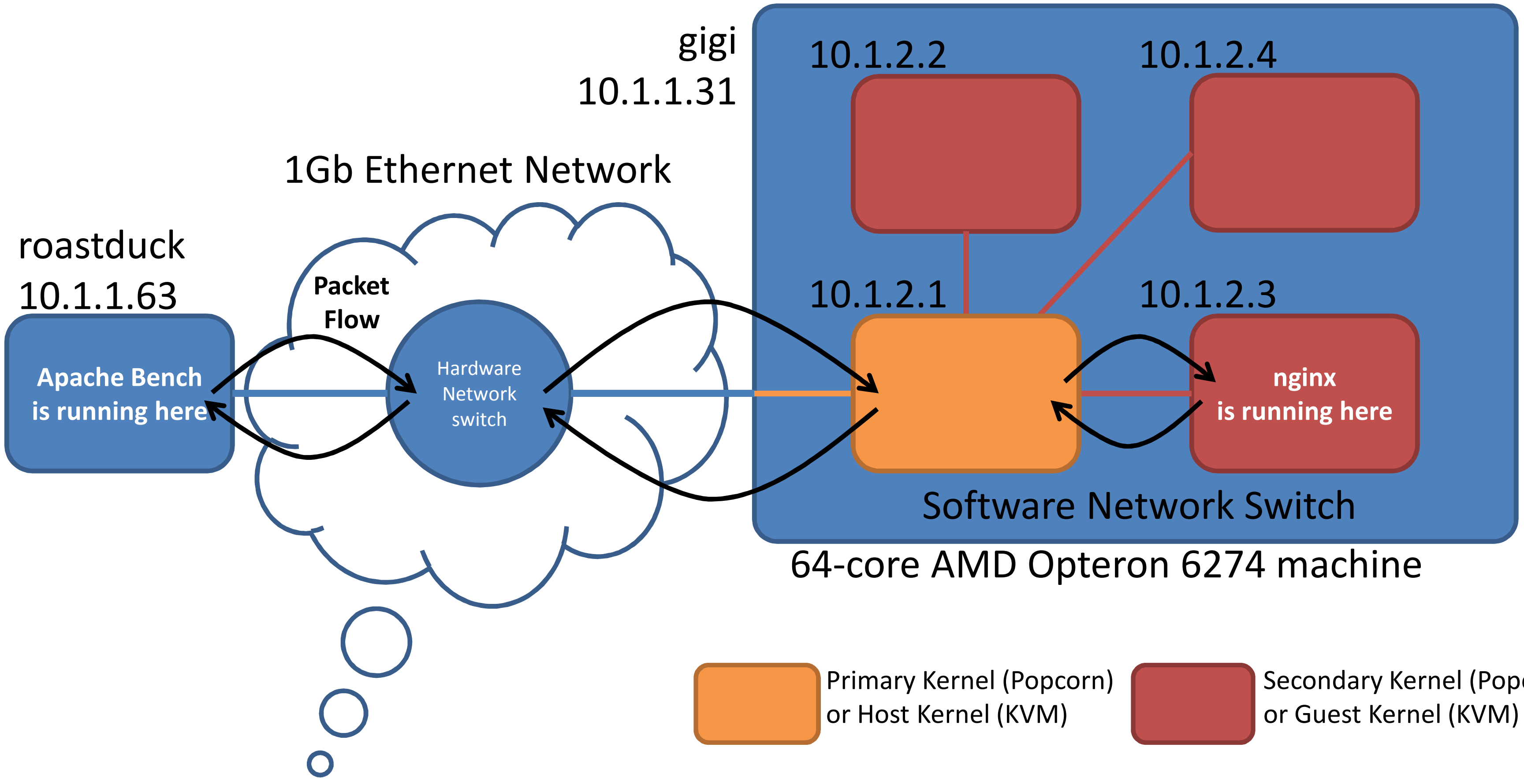




Inter-Kernel Virtual Network Switch

- First version based on Linux TUN/TAP interface, runs in **userspace**
- Second version based on Linux TUN/TAP interface in **kernel-space**
- Current version is **hybrid** interrupt and polling mechanism based on Linux NAPI (kernel-space)
  - Fast ring buffers in shared memory between client and server
  - Under high traffic, using polling reduces the overhead of servicing thousands of interrupts per second

Apache Bench Test Setup



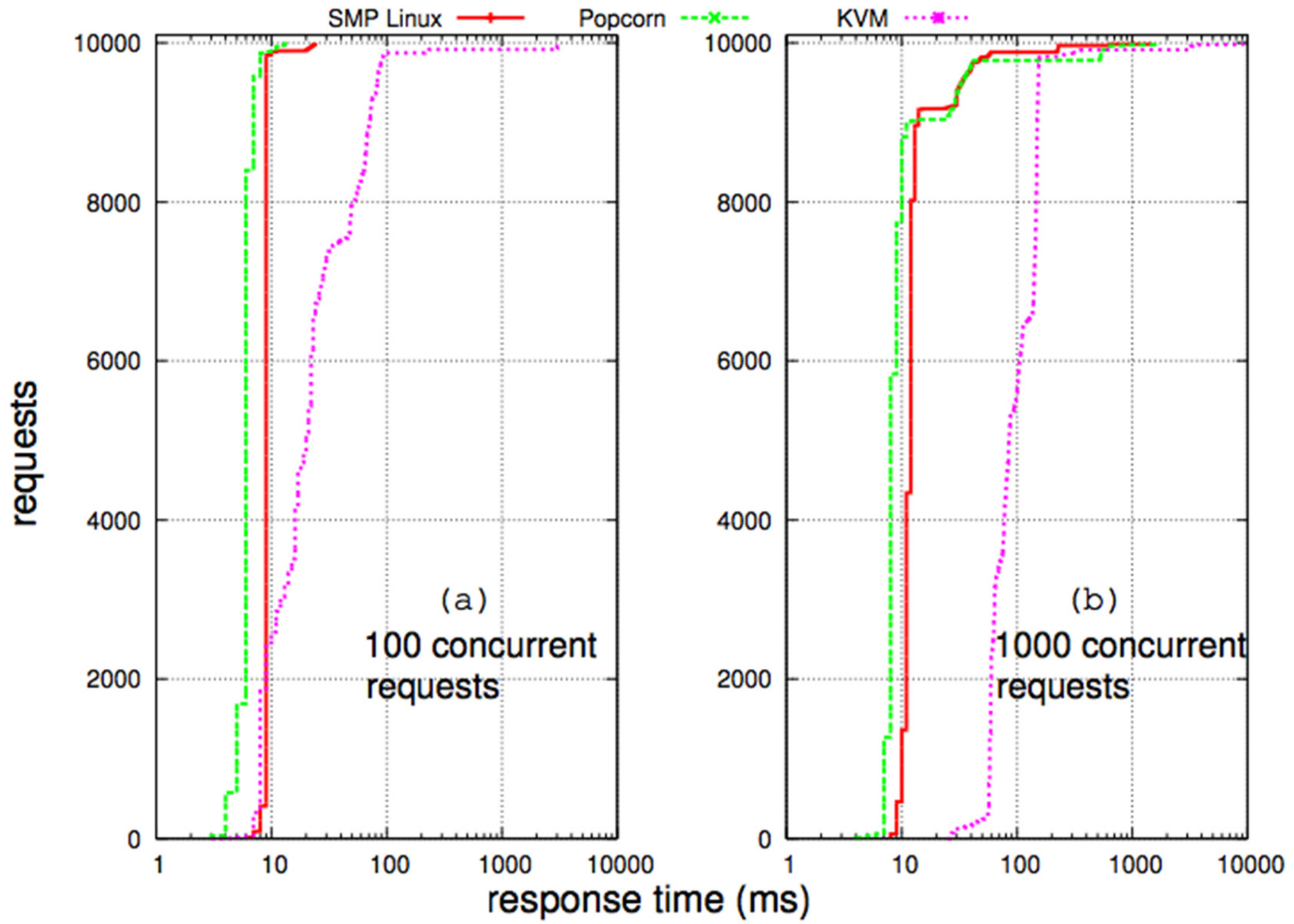
🟧 Primary Kernel (Popcorn) or Host Kernel (KVM)

🟥 Secondary Kernel (Popcorn) or Guest Kernel (KVM)

Systems Software Research Group

VirginiaTech Invent the Future

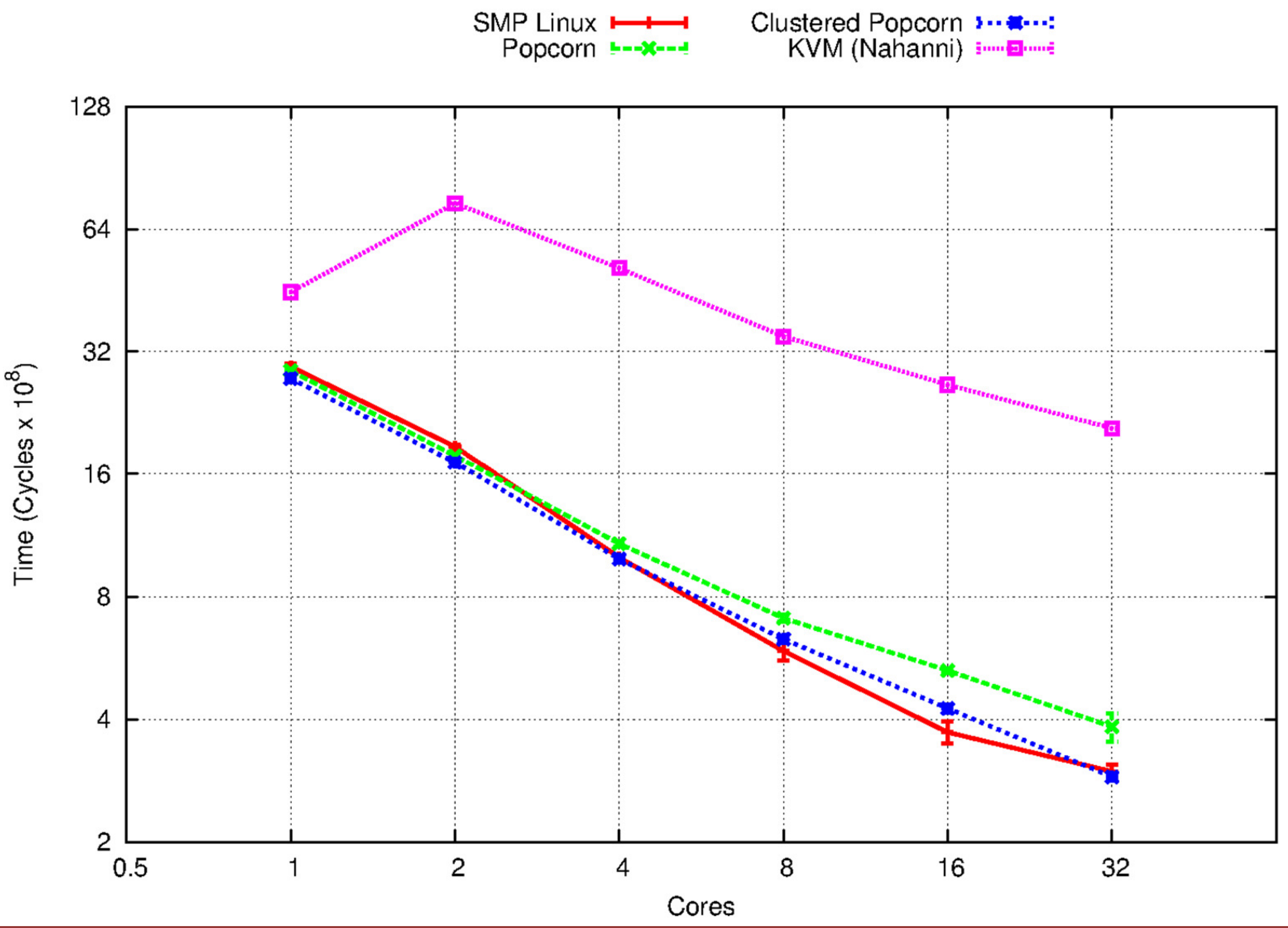
Results: Webserver, Apache Bench



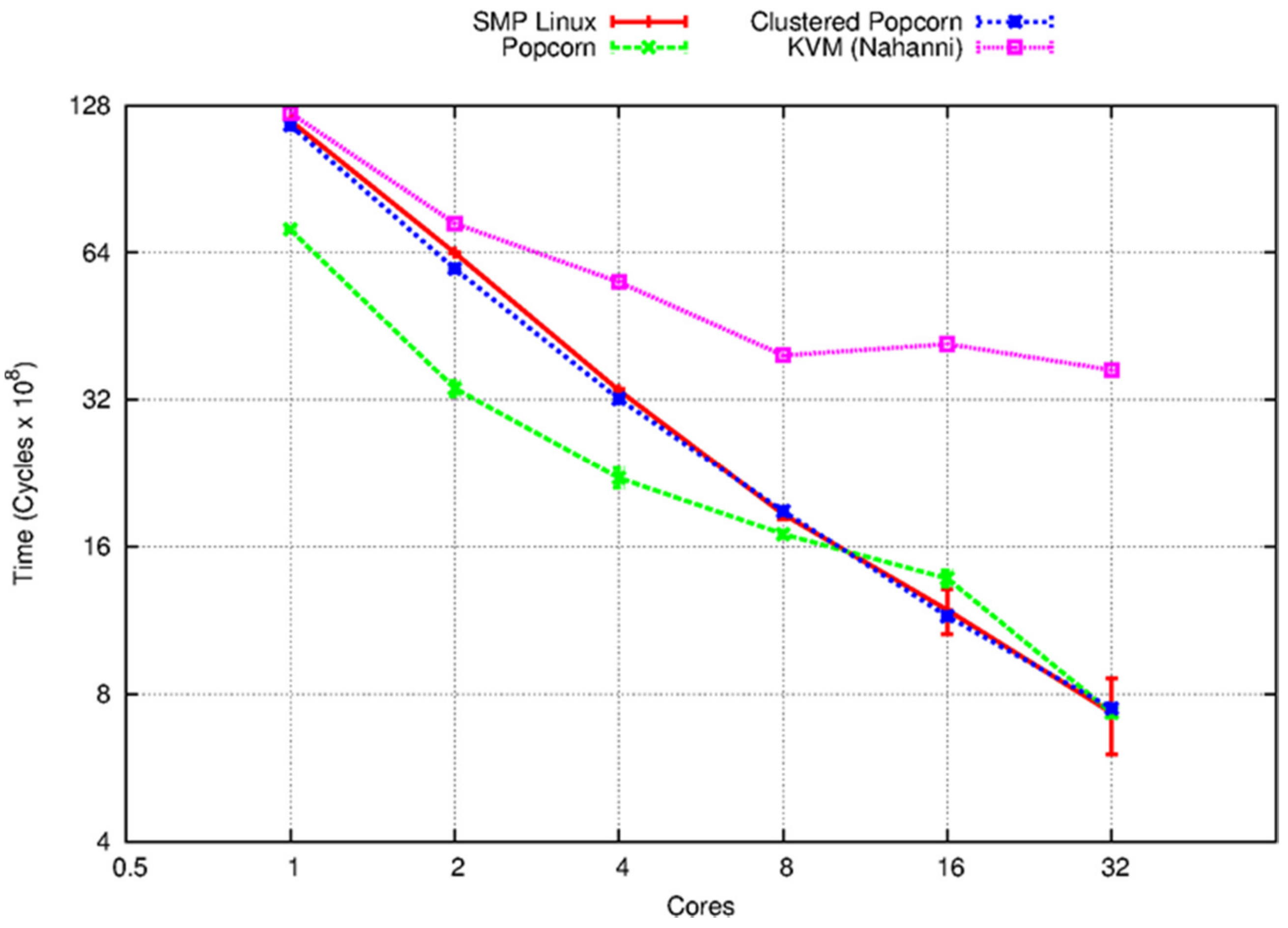
NPB MPI Setup

- We ran NASA Parallel Benchmarks (NPB) MPI version to verify that Popcorn **maintains the compute bound performance** of Linux
  - We compared it with KVM and SMP Linux
  - We tested one kernel per core configuration (partitioned Popcorn)
  - We tested one kernel per NUMA node configuration (clustered Popcorn)
- We used MPICH2 in all configurations (KVM uses an inter-VM shared memory called ‘Nahanni’)

Results: Compute Bound, IS-NPB



Results: Compute Bound, CG-NPB



Systems Software Research Group

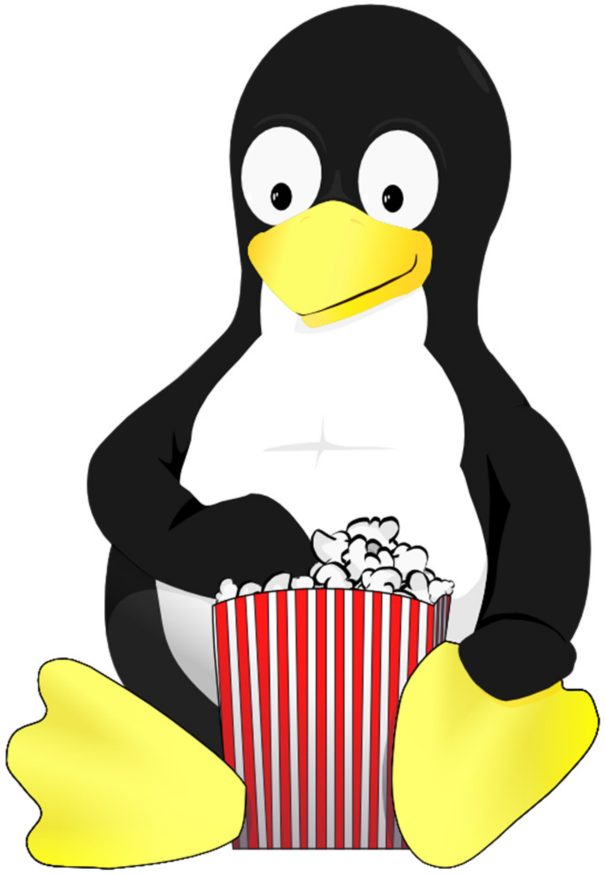
VirginiaTech Invent the Future

Adopting Popcorn (a summary)

- Many different kernels
- Every kernel is responsible of a subset of all physical resources (CPUs, memory, devices, etc.) present in the machine
- A single Operating System state is kept coherent across different kernels, not every kernel object must be kept coherent across all kernels (single OS configuration only)
- Applications can transparently use all physical resources present in the machine
- Applications on different kernels can communicate
- Applications see a Single System Image and can transparently migrate on different kernels (single OS configuration only)

Questions?

Team



www.popcornlinux.org