

Popcorn Linux Tutorial @ SOSP2019

Antonio Barbalace, Pierre Olivier, Binoy Ravindran



SOSP2019 Schedule

Sunday
Breakfast 7:30 AM - 9:00 AM

Workshops morning pre-break session 9:00 AM - 10:30 AM

Break 10:30 AM - 11:00 AM

Workshops morning pre-break session 11:00 AM - 12:30 PM

Tutorial Schedule

- 9:00AM – 9:30AM Welcome/Introduction
- 9:30AM – 10:30AM Popcorn Tutorial (1, 2a, 2b, 3a)
- 11:00AM – 11:30AM Popcorn Tutorial (3b, 4)
- 11:30AM – 12:15PM Popcorn Hexo Tutorial
- 12:15PM – 12:30PM Concluding Remarks

Tutorial Format

- Short Lectures
- Interactive Labs
- **Popcorners** in the room to help with the Labs
 - Tong Xing (Stevens)
 - Wei Wang (Stevens)
 - Xiaoguang Wang (VT)

Before Starting

- Popcorn Linux Deployments
 - Two physical machines (not for this tutorial)
 - Two virtual machines
 - On local machine
 - On remote machine
 - One virtual machine **with** two virtual machines
 - On local machine

Before Starting – Two Virtual Machines, Local

- Suggested to Linux users (**only**)
- Create the VMs from scratch (may take several hours)
 - [~~https://github.com/systems-nuts/popcorn_kernel/wiki/Build-Img-guide~~](https://github.com/systems-nuts/popcorn_kernel/wiki/Build-Img-guide)
- Download the VM images, or copy them from Popcorn's USB sticks
 - <https://www.cs.stevens.edu/~txing1/x86.img.zip>
 - <https://www.cs.stevens.edu/~txing1/x86.img.zip.md5>
 - <https://www.cs.stevens.edu/~txing1/arm.img.zip>
 - <https://www.cs.stevens.edu/~txing1/arm.img.zip.md5>
 - https://www.cs.stevens.edu/~txing1/QEMU_EFI.fd

Download
now!

Before Starting – Two Virtual Machines, Remote

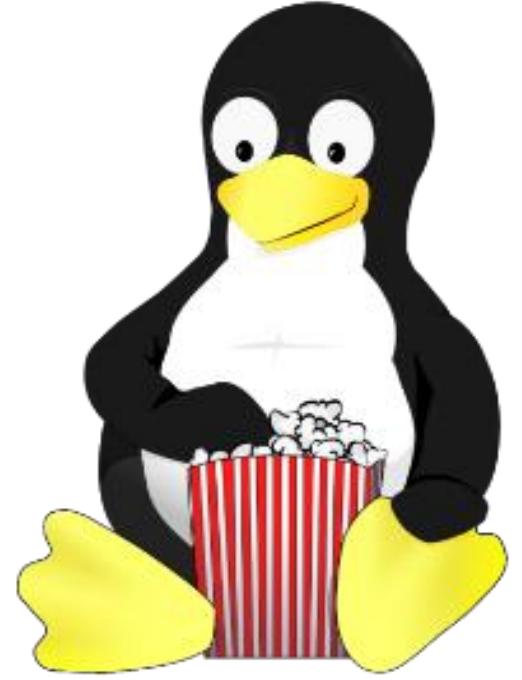
- Suggested to non-Linux users
 - Remote access will be **available for a limited time after the tutorial**
- Requires a web browser
 - Tested on Firefox, Chrome, MS Edge
- Open two (2) browser windows/tabs at
 - **VT LINK**
- We distribute user/password combinations (now)
 - Each user should connect to an ARM VM and a x86 VM

Connect
now!

Before Starting – Single Virtual Machine

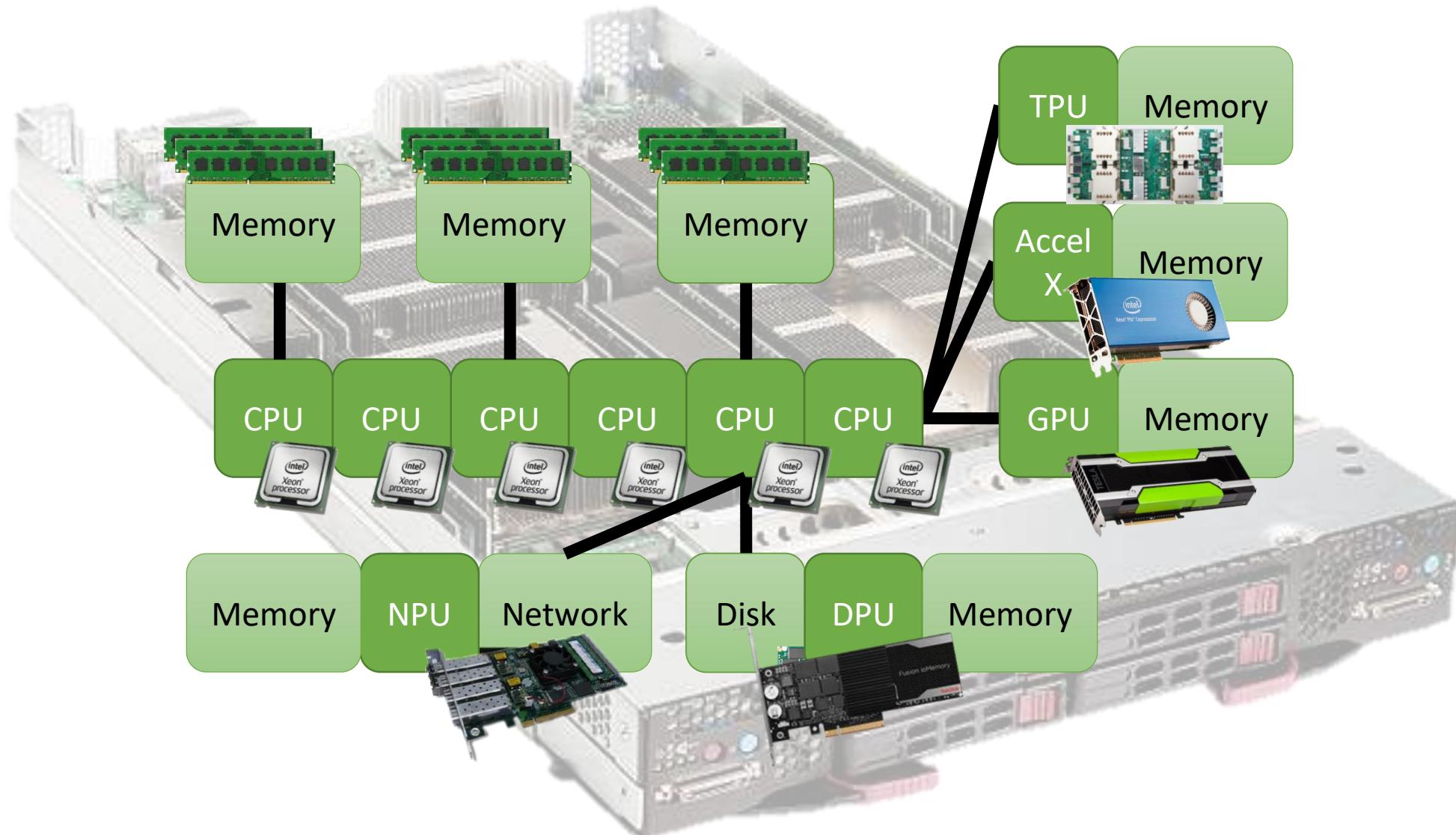
- Suggested to non-Linux users
- Requires Oracle Virtualbox
 - <https://www.virtualbox.org/>
- Download the VM image, or copy it from Popcorn's USB sticks
 - <https://www.cs.stevens.edu/~txing1/popcorn-box.ova>
- Import the VM image in Virtualbox
 - https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html

Download
and import
now!

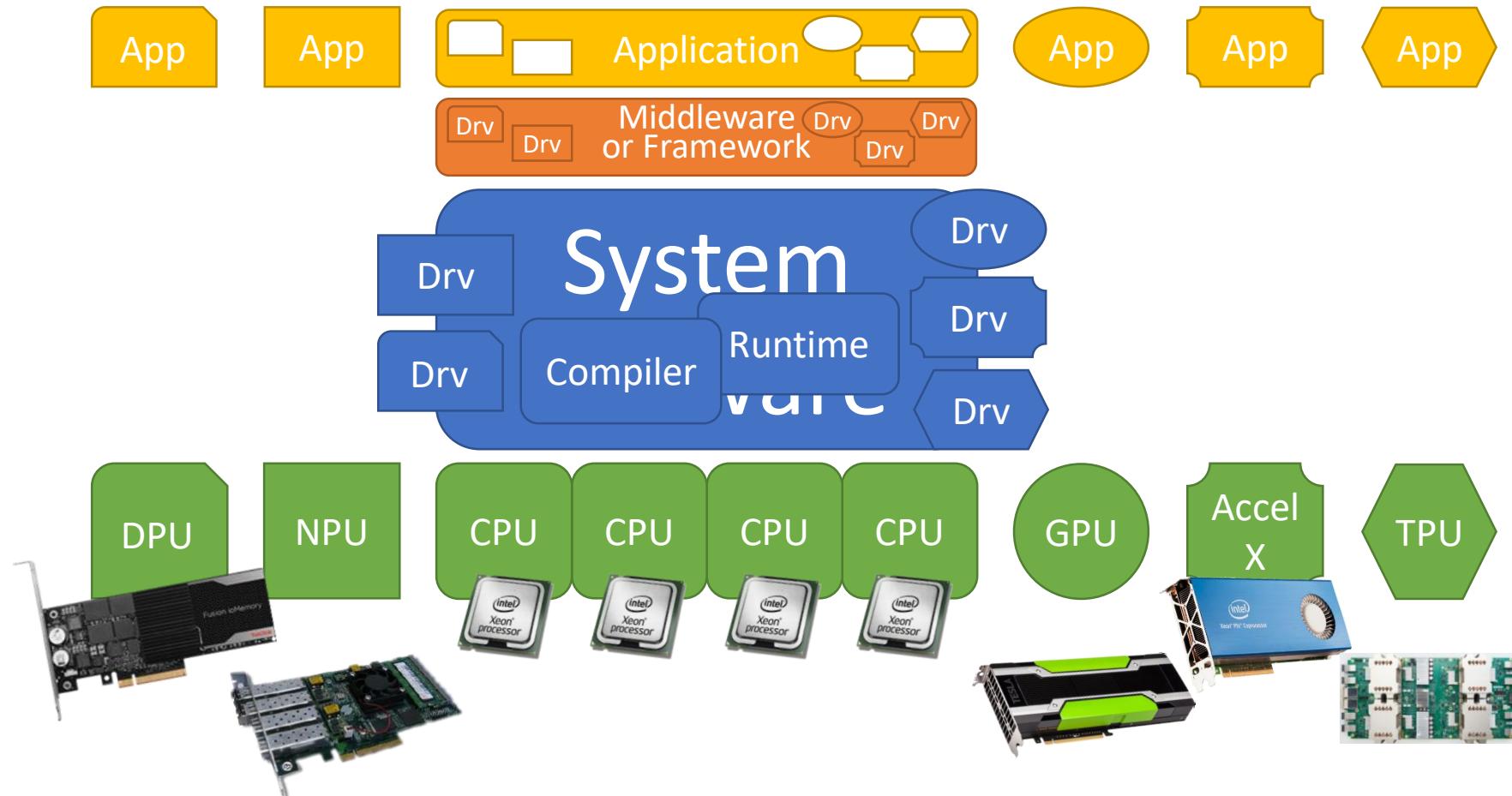


Popcorn Linux Introduction

Modern Hardware Landscape



Modern Software Landscape



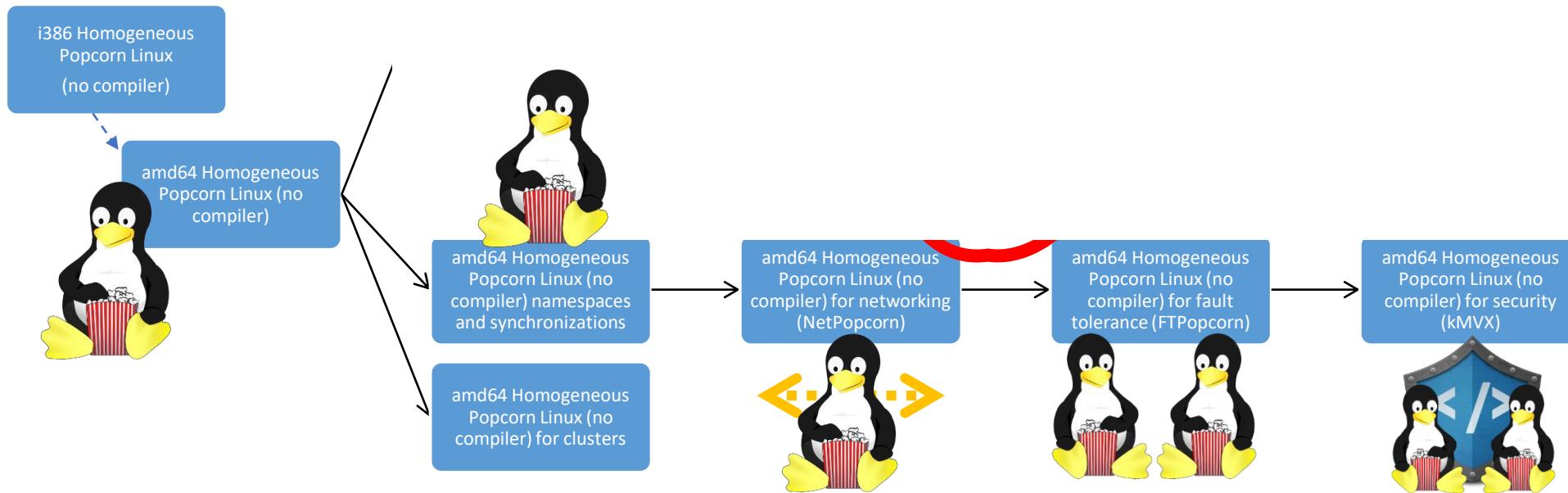
The Popcorn Linux Project

- Goals
 - Ease programmability of modern (complex) hardware
 - Full exploitation of all available (costly) resources
 - For all
- Foundational Principles
 - Seamlessly running application across processing units of different ISA
 - Transparently from the programmer, support SMP code
 - Advanced resource scheduling/mapping
 - For high-performance, low-power consumption, fault-tolerance, security, etc.
 - Based on production-grade OS (Linux) and compiler (LLVM)

The Popcorn Linux Project (continue)

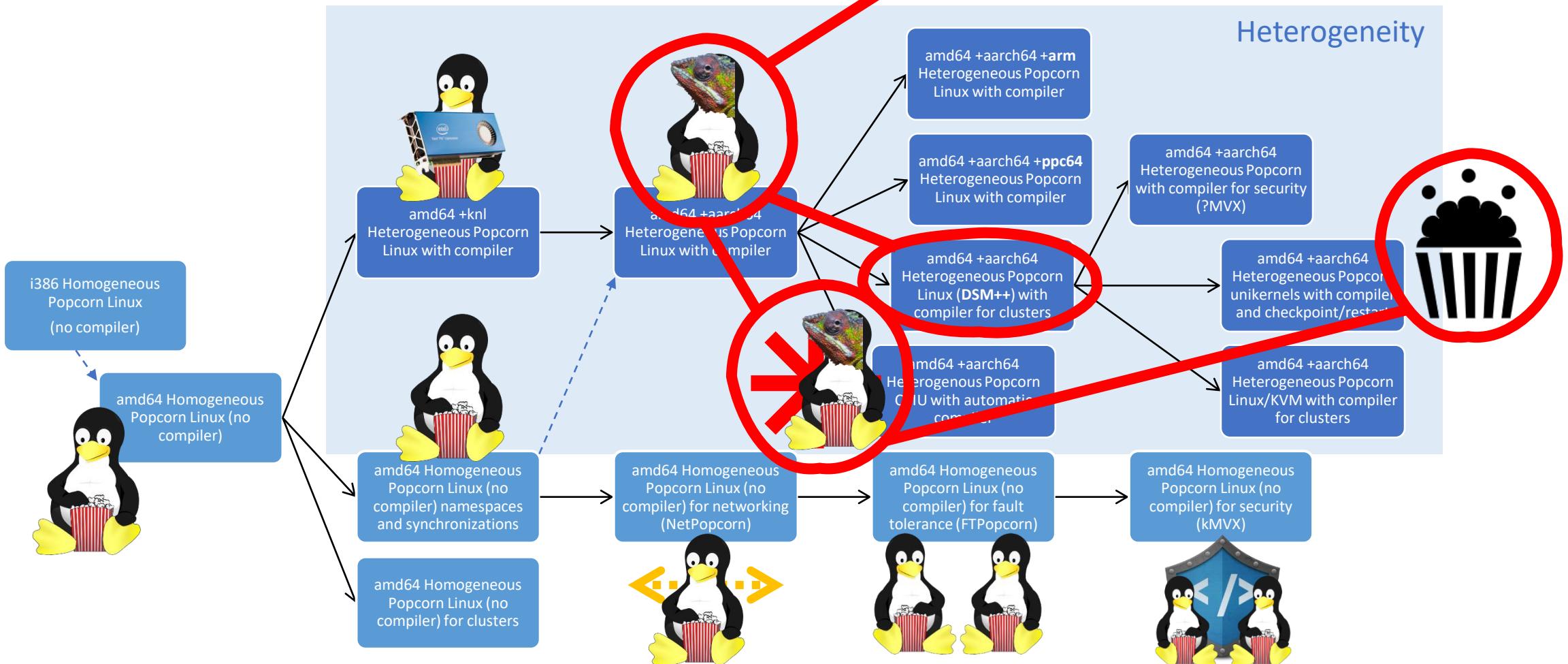
- **Brief History**
 - 2012-beginning initial idea (VT)
 - 2012-mid start first prototype
 - 2012-end first OS-only prototype demonstrated to stakeholders (x86 multiple-kernel)
 - 2014-beginning Xeon-Xeon Phi prototype demonstrated to stakeholders
 - 2015-mid first ARM-x86 prototype demonstrated to stakeholders
 - 2015-end ARM-x86 prototype demonstrated @ SOSP2015
 - ...

Popcorn Family of Projects



Popcorn Family of Projects

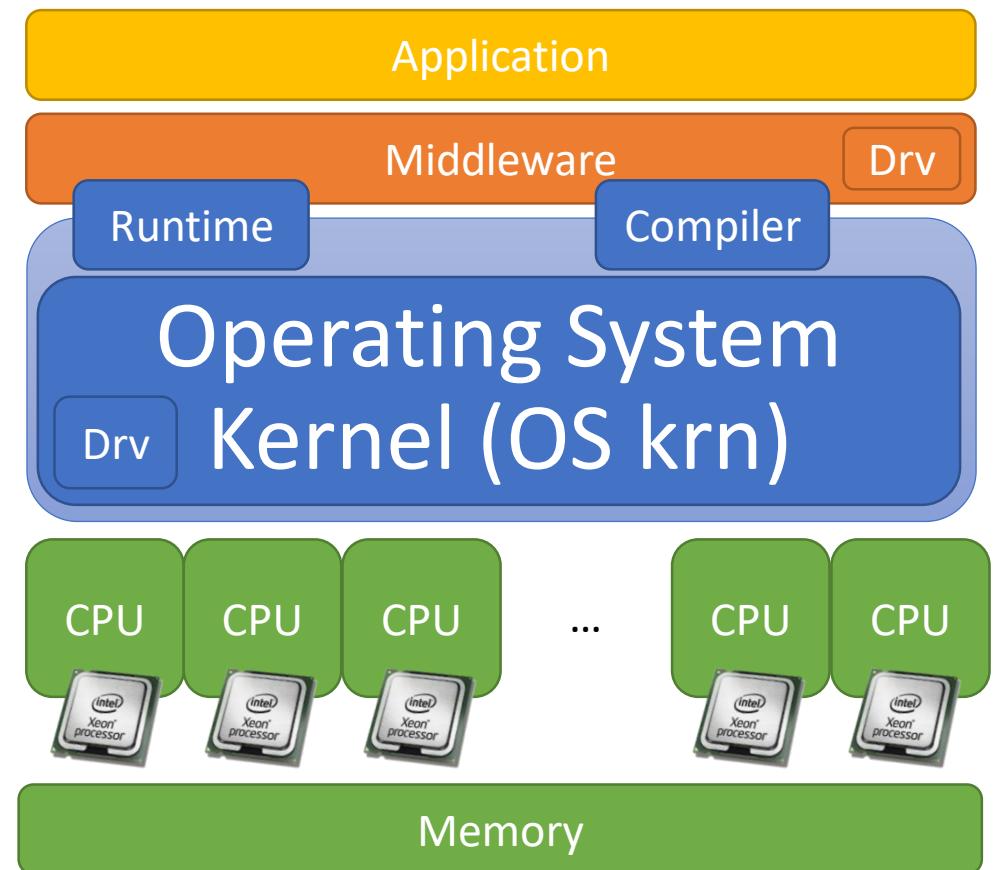
Focus of this tutorial



What is Linux?

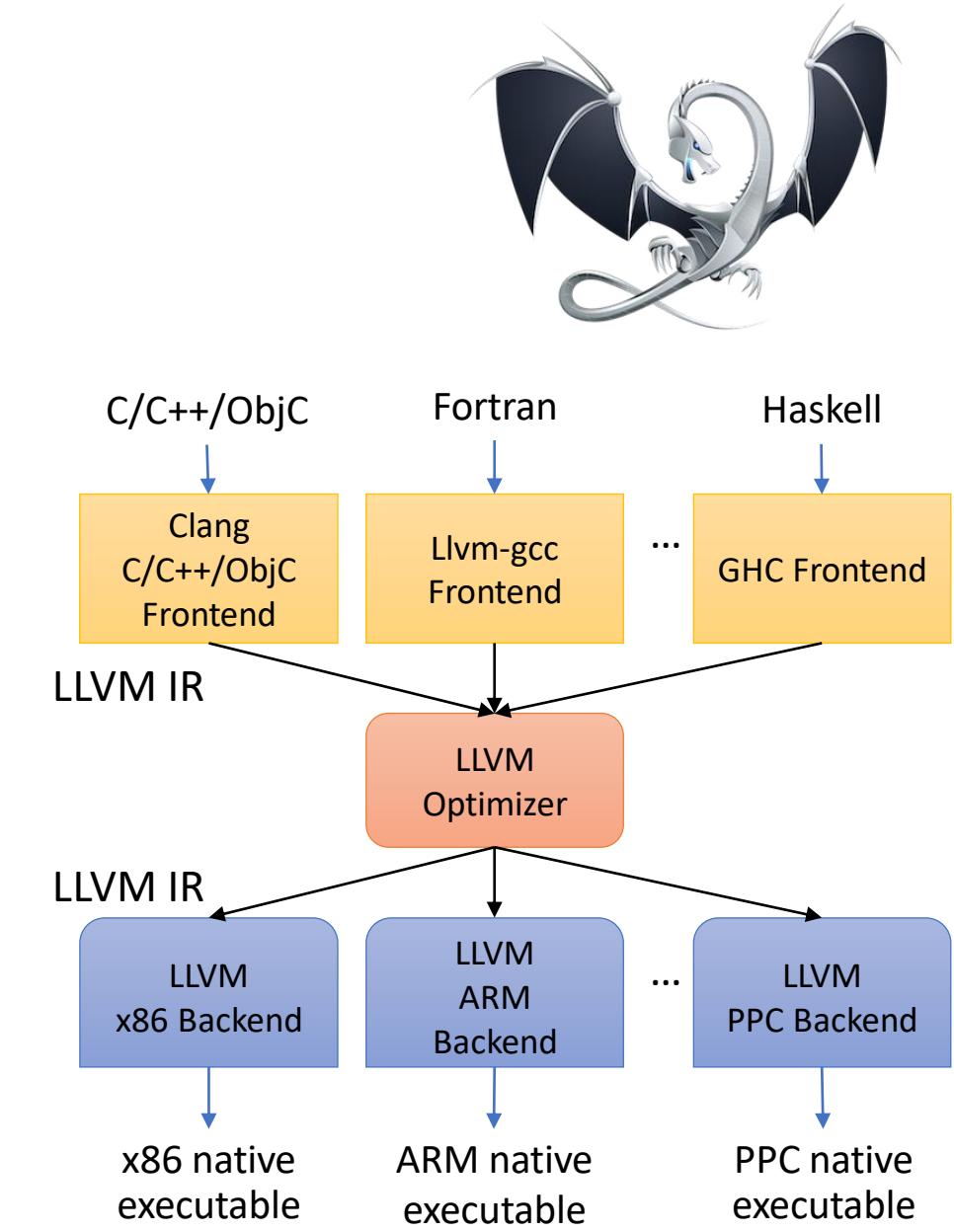


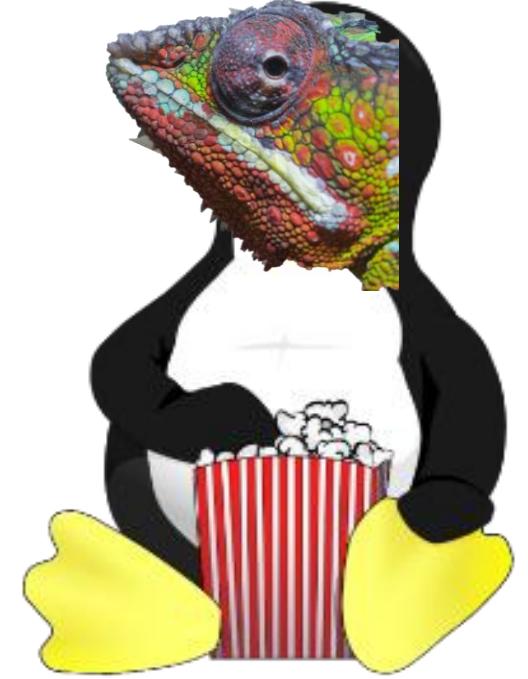
- Monolithic SMP OS
 - **Monolithic**: all OS services in the OS kernel
 - **SMP**: exploits shared memory, and multiple CPUs; targets homogeneous CPUs
- Widely adopted
 - Servers
 - Home/office
 - Embedded/IoT
- Large software ecosystem
 - Applications
 - Developer tools



What is LLVM?

- New generation compiler toolchain
 - **Modular**: easy to extend
 - **Based on IR**: supports multiple programming languages
 - **Multiarch**: native compiler support for multiple architectures
 - No need for multiple compiler toolchains
- Widely adopted
 - Small projects
 - Large projects (Linux kernel, iOS apps, etc.)
- Large number of optimizations





Heterogeneous Popcorn Linux

Increasing Heterogeneity per-Socket/Platform

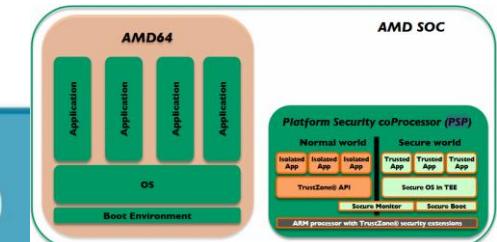
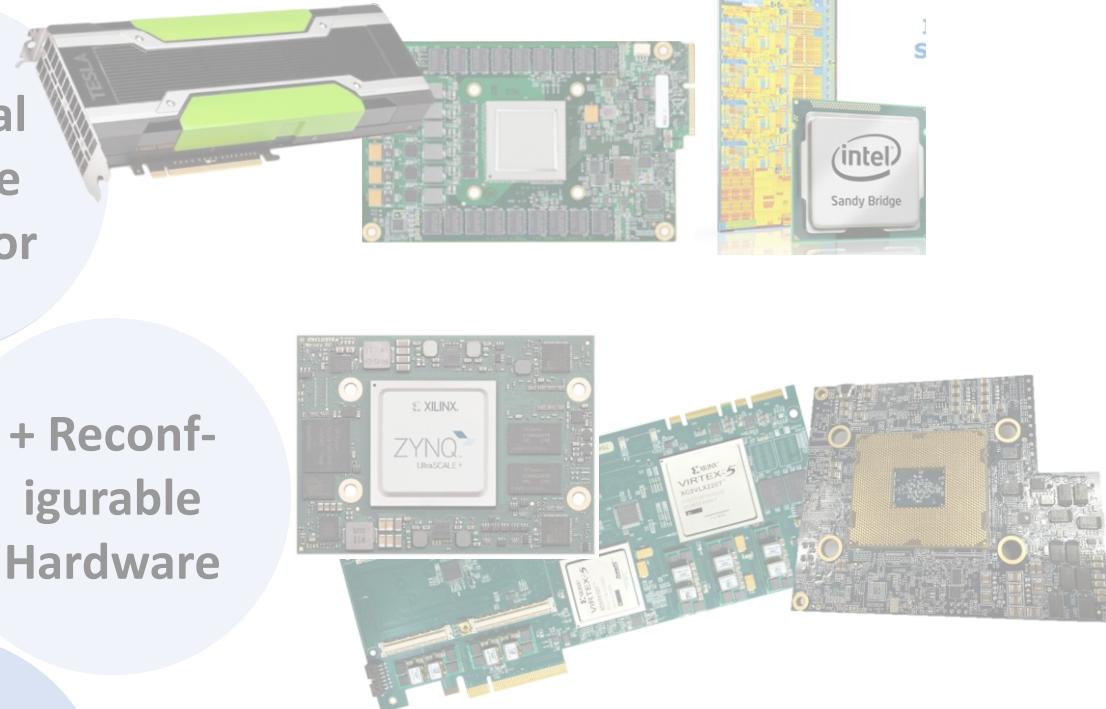


General
Purpose
Processor

+ Special
Purpose
Processor

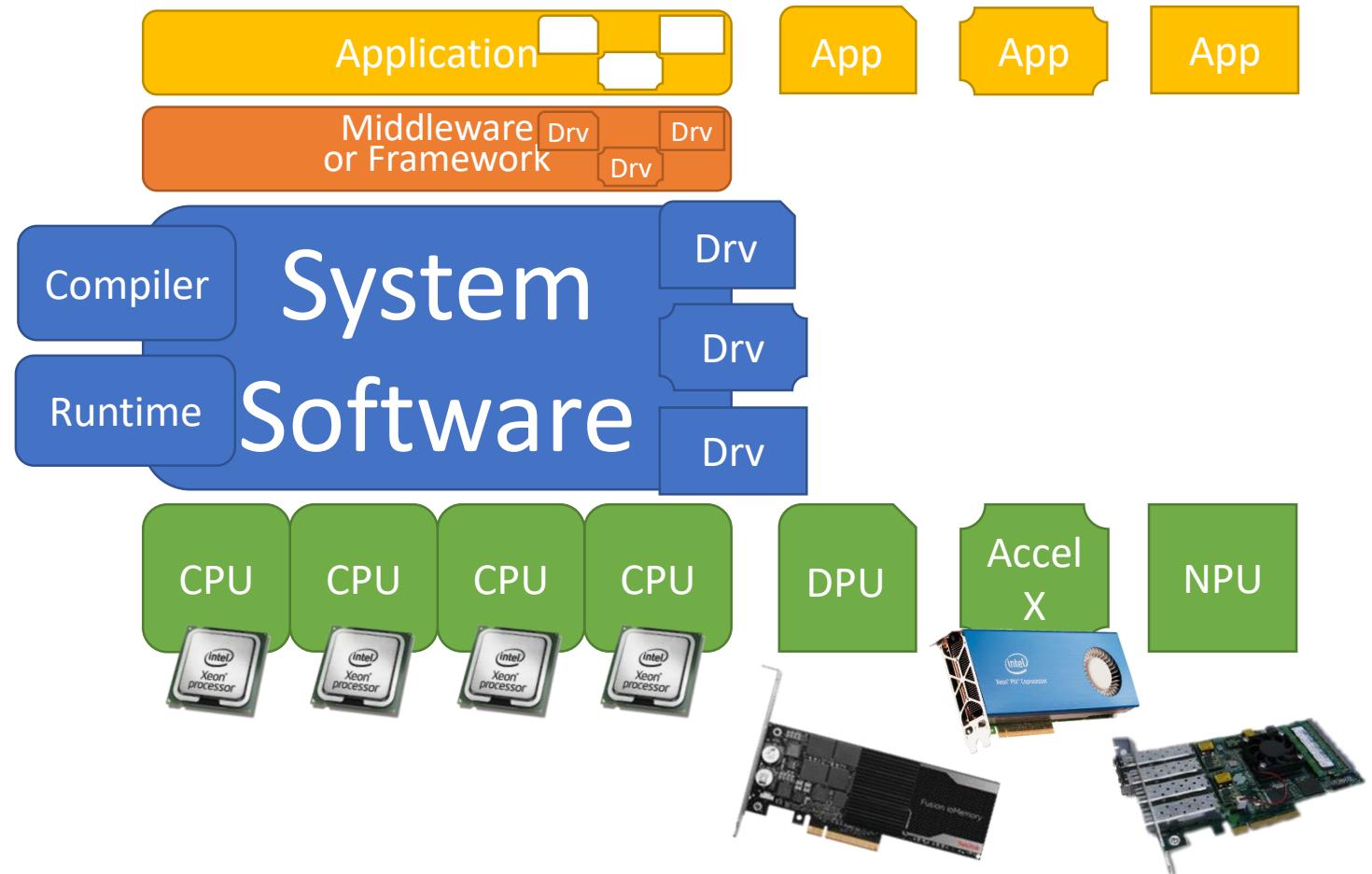
+ Reconf-
igurable
Hardware

+ General
Purpose
Processor



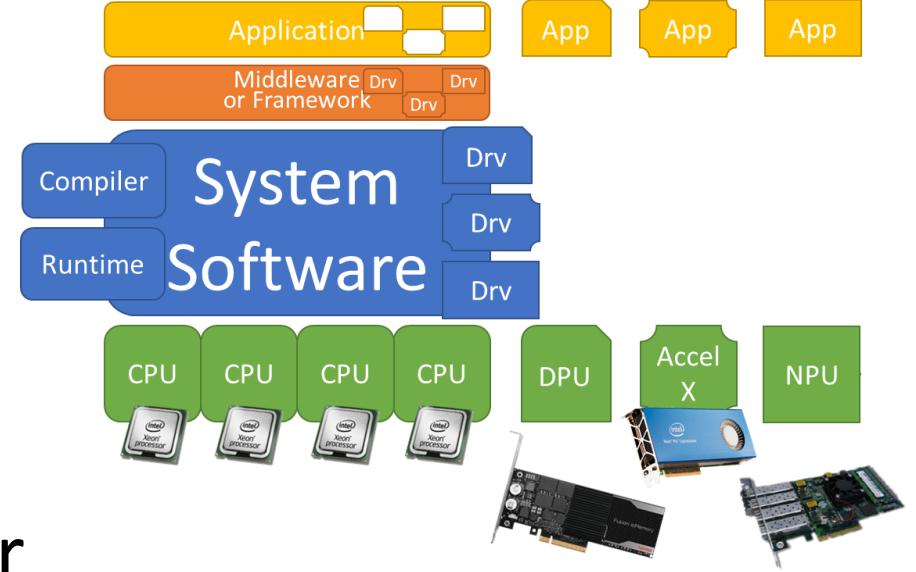
Classic Software for Heterogeneous Hardware

- Software runs on CPUs
- Other processing units cannot run the same software as the CPUs
- Programmer (strictly) partitions the application
- Each partition runs only on a predefined processing unit
- Supporting OS drivers, runtime, compilers



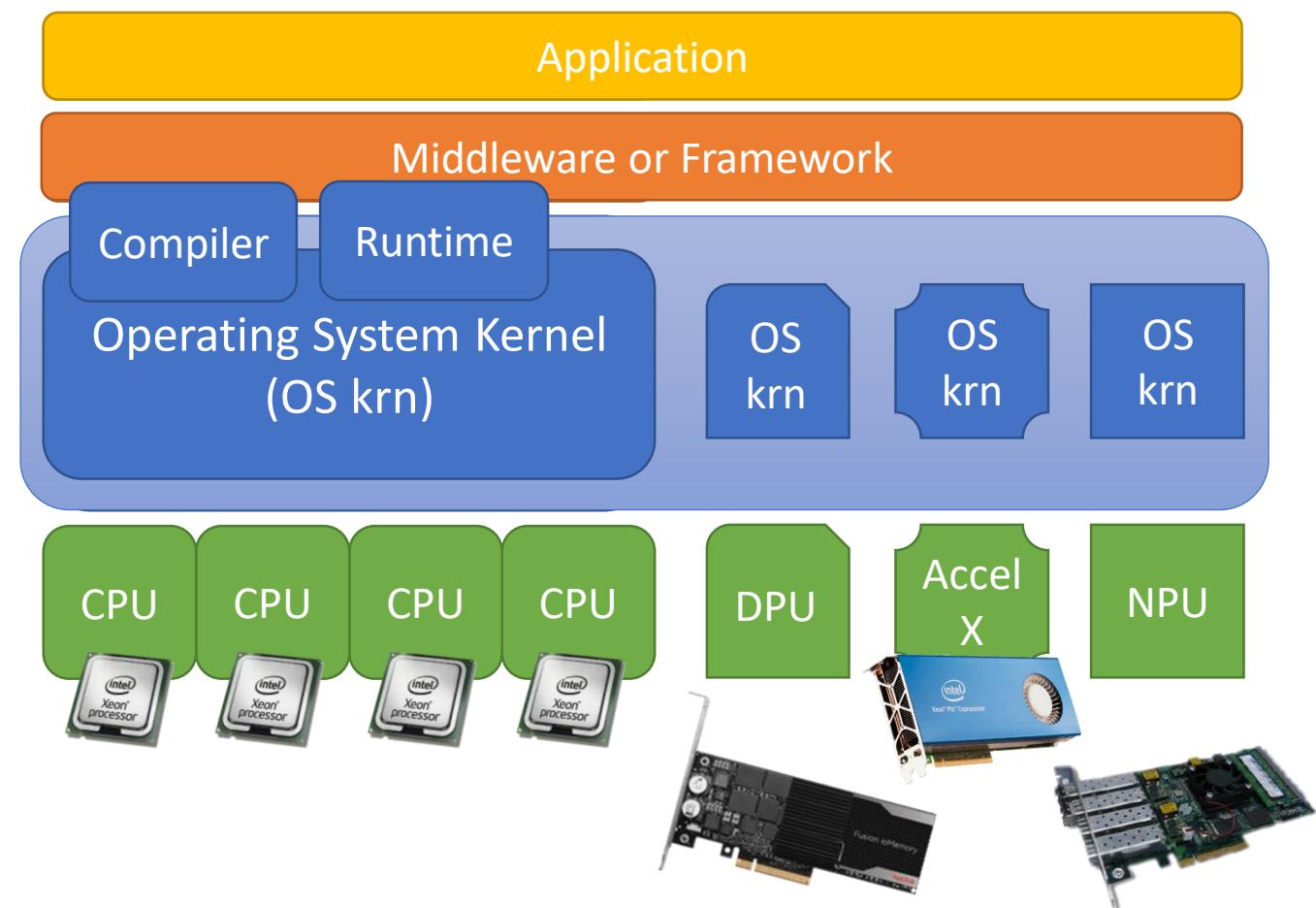
What Are the Problems?

- For each hardware component
 - Modify **all software layers**
 - Issues for the application's programmer
 - Hard to program
 - Difficult to port to a new platform
 - Poor resource utilization (performance, energy efficiency, determinism, etc.)
 - Resource distribution
 - Resource dependencies (and fault-tolerance)
 - Resource sharing (and security)

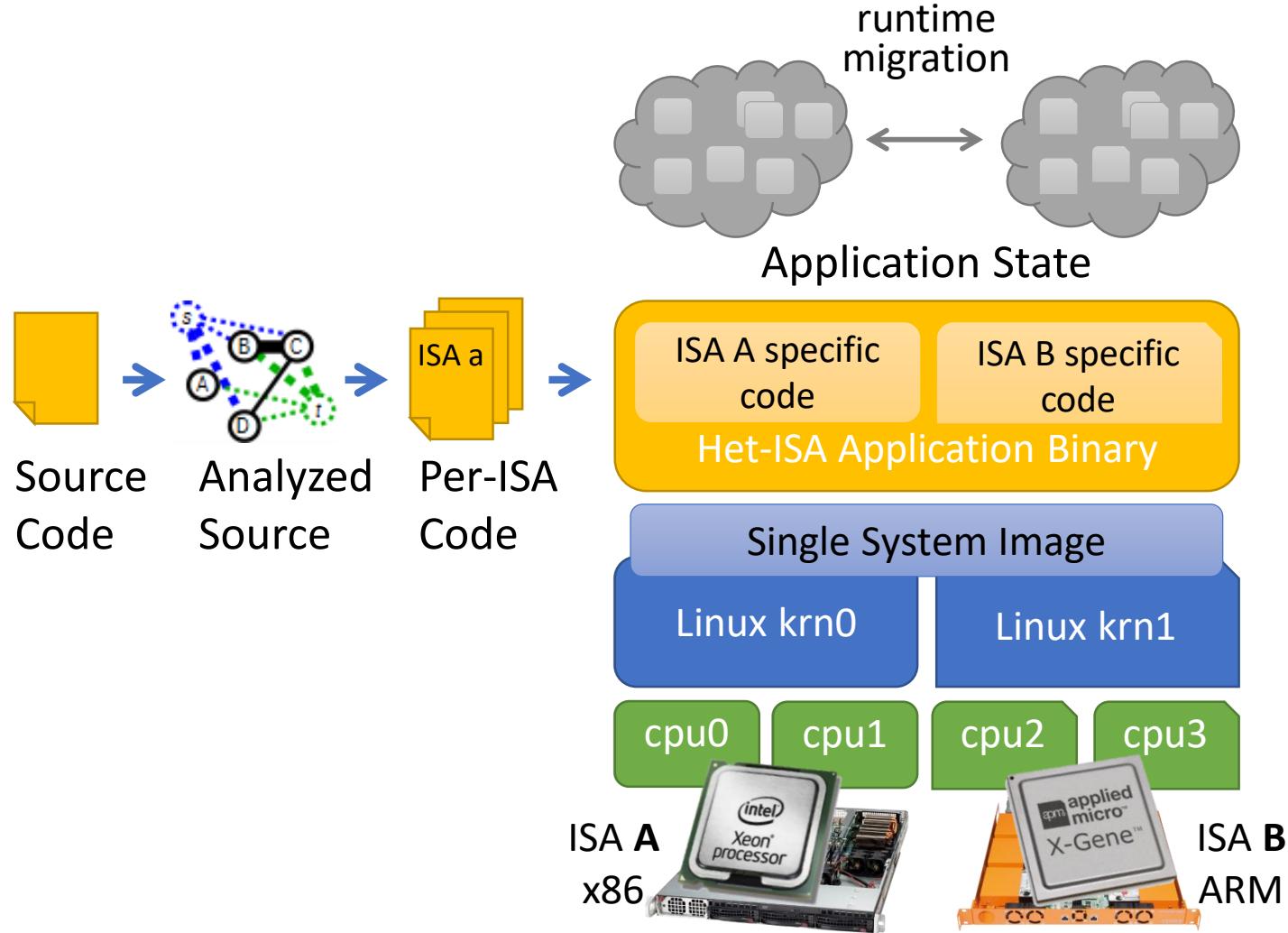


New Software for Heterogeneous Hardware

- The **OS** extends among all processing units
- The **compiler** builds applications software to run among all processing units
- The **runtime** supports all processing units
- **Programmers** don't have to partition the application, which may run everywhere

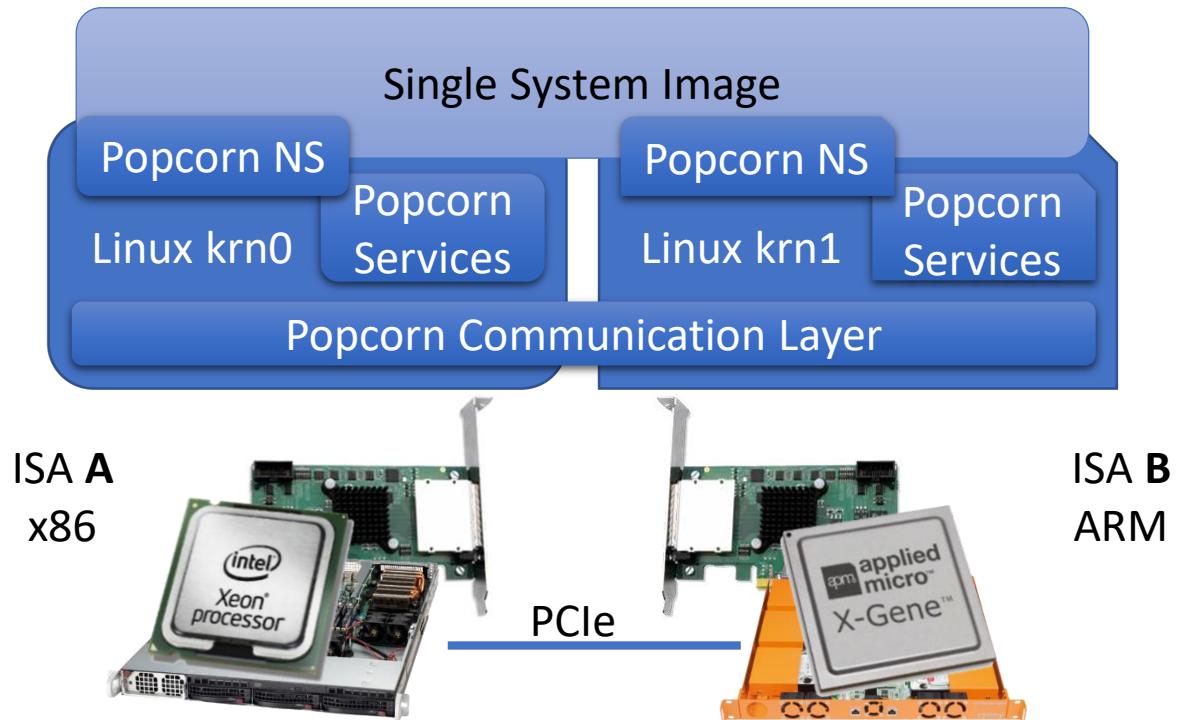


Heterogeneous Popcorn Linux



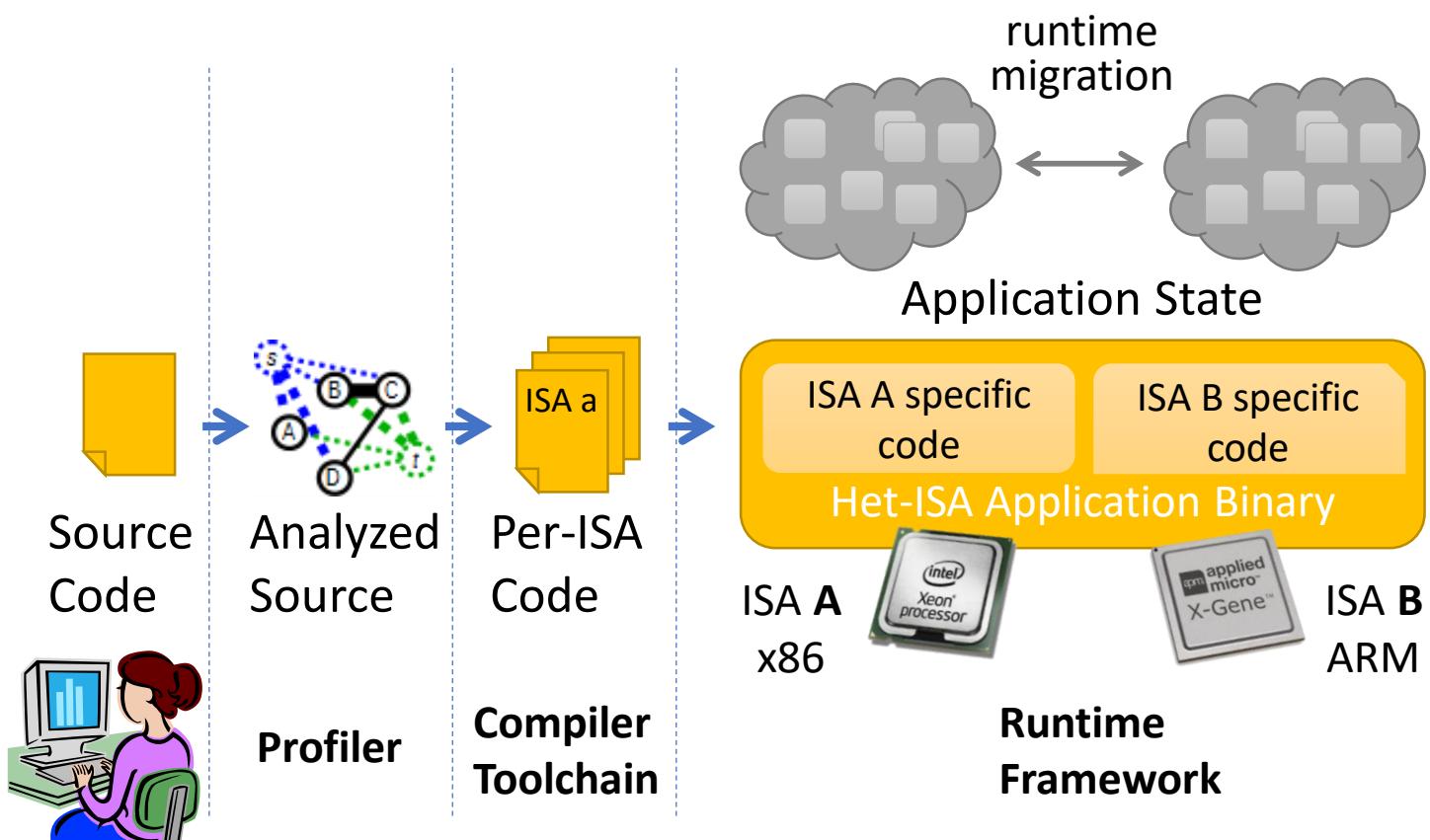
- **Runtime**
 - Runtime ISA execution migration
 - State transformation
 - Based on **musl**
- **Compiler Framework**
 - Offline analysis
 - One binary per ISA
 - Based on **LLVM**
- **Replicated-kernel Operating System**
 - One kernel per ISA
 - Distributed systems services
 - Based on **Linux**

The Operating System



- **Single System Image**
 - Was based on Popcorn namespaces (NS)
 - Creates a single operating environment
 - Migrating app sees the same OS
 - Extends Linux among nodes
- **Distributed OS Services**
 - Task (thread and process) migration
 - Native code migration
 - Distributed shared memory
 - Distributed file system
- **Inter-kernel Communication Layer**
 - Performance critical component
 - low-latency and high-throughput
 - Exclusively kernel-space
 - Single format among ISAs

The Compiler/Runtime



- **Profiler**
 - Performance and power profiles
 - Function and sub-function granularity
 - Output performance and power code indicators
- **Compiler/Linker Toolchain**
 - Output heterogenous-ISA binary (native)
 - Common address space (includes TLS)
 - Insert migration points (fun boundaries)
 - Add state transformation metadata
- **Runtime Framework**
 - Support task migration
 - Implements state transformation
 - Stack-transformation (rewriting)
 - Register-transformation

Heterogeneous Popcorn Linux ...

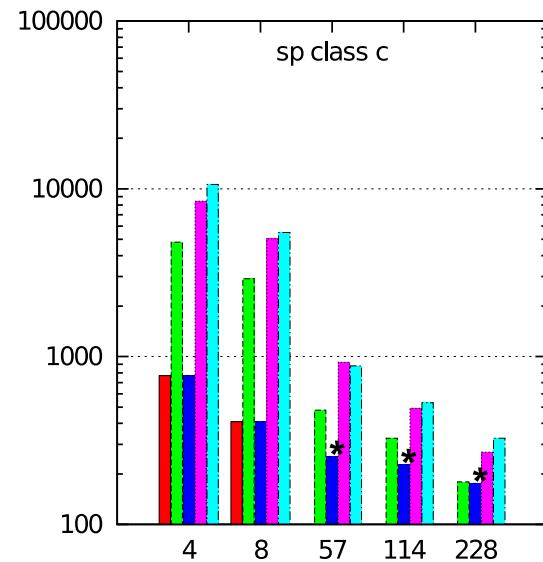


- **Enables immediate support of heterogeneous-ISA hardware**
 - Ease programmability (minimal to no code changes)
 - Portable across different hardware (and legacy support)
- **Minimal runtime overheads**
 - Application's changes demonstrated no observable overheads
 - Operating systems services' changes may show synchronization overheads
- **Improves hardware resource utilization**
 - Enabling Runtime vs static decision making

Runtime vs Static Decision Making

- On heterogeneous-ISA [1]
 - Up to 3.5x more performant than other heterogeneous frameworks

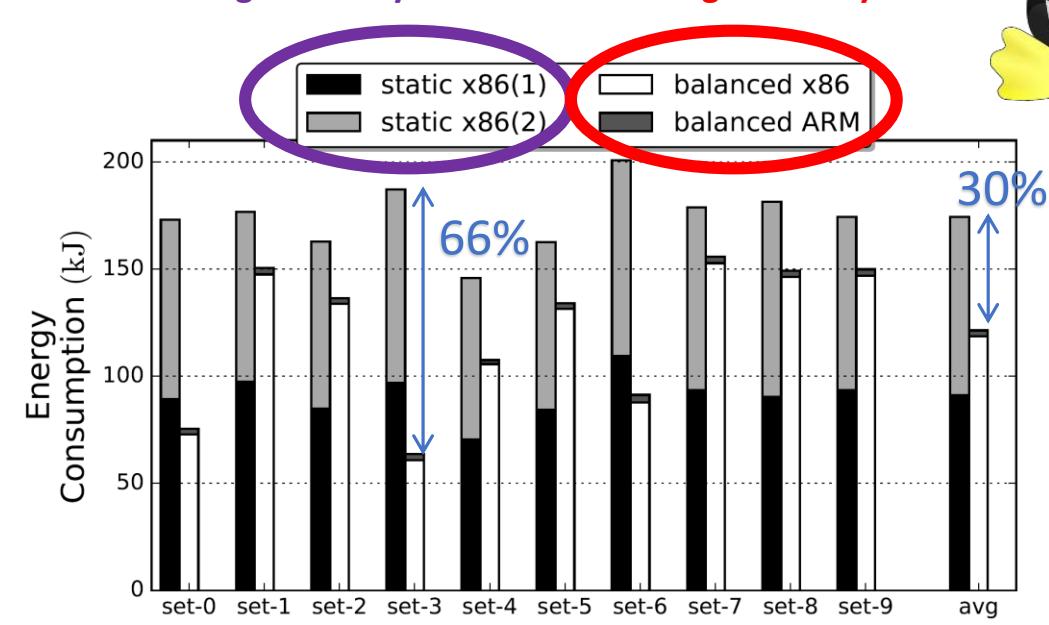
Xeon XeonPhi Popcorn
OpenCL Offloading migration *



46%
faster
53%
faster
3.5x
faster

- On fully heterogeneous-ISA [2]
 - Up to 66% better energy consumption for bursty arrivals

Homogeneous System Heterogeneous System



[1] "Bridging the Programmability Gap in Heterogeneous-ISA Platforms" A. Barbalace et al., EuroSys '15

[2] "Breaking the Boundaries in Heterogeneous-ISA Datacenters" A. Barbalace et al., ASPLOS '17

Software Versions and Supported Hardware



- Heterogeneous Popcorn Linux (latest stable release)
 - Linux kernel version 4.4.137
 - clang/LLVM 3.7.1
- Supported Platforms
 - x86-64 (**x86** 64bit)
 - Any hardware that can run Linux kernel version 4.4.137
 - aarch64 (**ARM** 64bit)
 - Any hardware that can run Linux kernel version 4.4.137
 - E.g., Cavium ThunderX

The Popcorn Tutorial

Popcorn Tutorial Breakdown

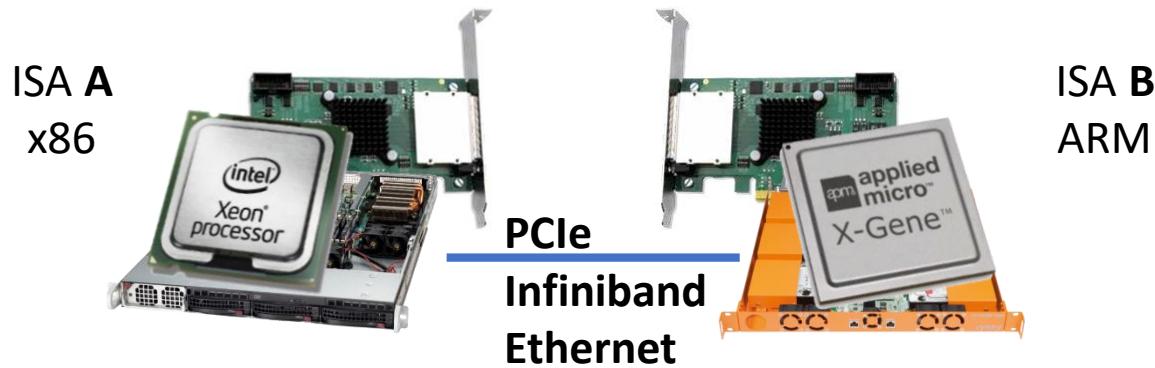
- Part 1
 - Environment Setup
- Part 2a
 - Enter Popcorn Linux
- Part 2b
 - Delving into Popcorn Linux
- Part 3a
 - The Popcorn Compiler Framework
- Part 3b
 - Dissecting Popcorn Compiler Framework
- Part 4
 - Pthread and Scheduling

Tutorial Part 1

Environment Setup

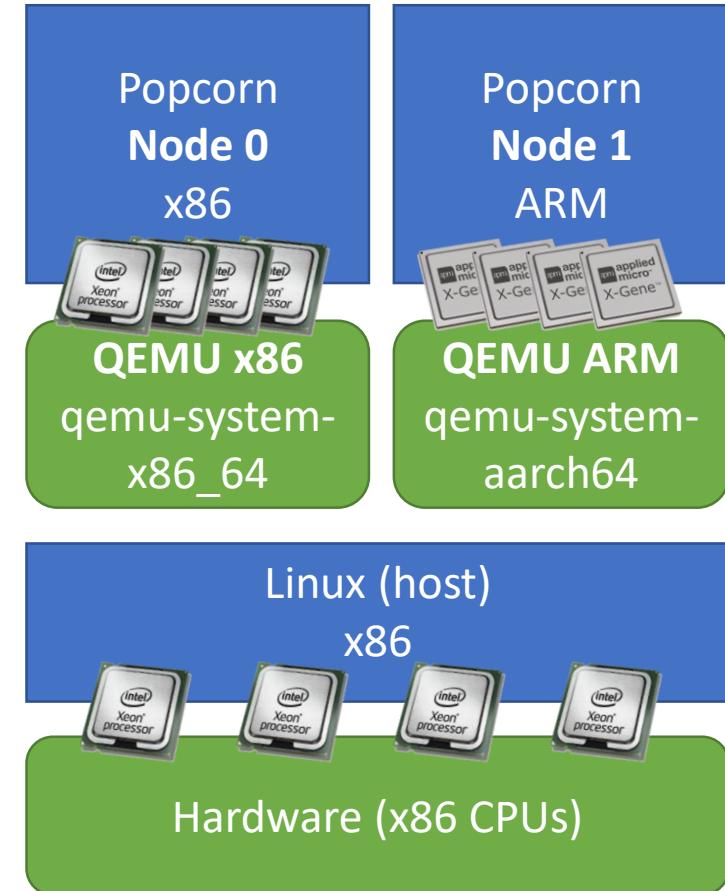
Classic Popcorn Deployments

- Two or more **machines**
 - Mounting heterogeneous-ISA processing units
 - Interconnected via PCIe, Infiniband, Ethernet, ...
- Two or more **processing units**
 - With heterogeneous-ISA
 - In the same machine
 - Interconnected via PCIe, QPI, HT, CCI, IF, ...



Tutorial Popcorn Deployment

- **Two VMs** (local/remote/in one VM)
 - x86 (x86-64) VM
 - ARM (aarch64) VM
- **x86-64 VMM (host)**
 - Hardware acceleration for x86 VM
 - ARM VM emulated
- VMs **communicate** via Ethernet
- **Interact** with VMs via
 - Serial line
 - SSH

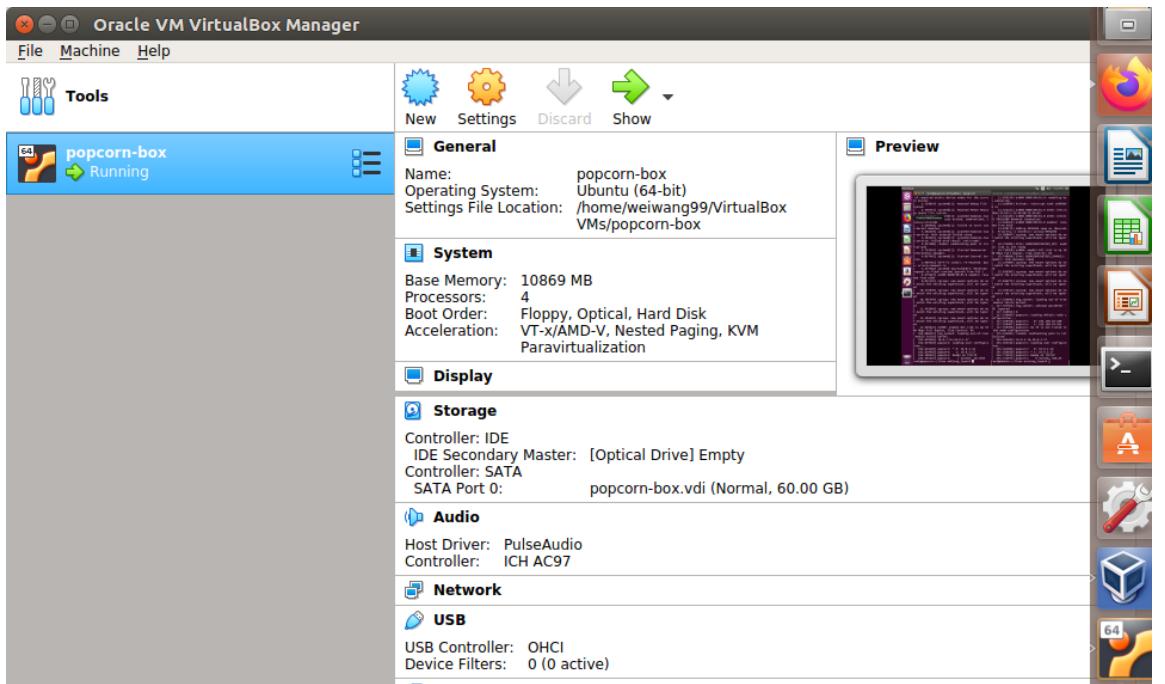


Notes on the Deployment

- Examples and commands based on Ubuntu
 - Should work on other distributions
 - Should work on Windows, OSX, and BSD
- All needed software is included in the VMs
 - No need to download/install/compile additional software
- The following slides illustrate the rest of the setup

Tutorial Setup – One Virtual Machine

- Import the VM and start it



The screenshot shows a terminal window titled 'Terminal' with the command 'root@popcorn-VirtualBox: ~/popcorn'. The terminal displays the boot log of the 'popcorn' virtual machine. The log starts with the message 'of required static device nodes for the current kernel.' and continues through various system initialization steps, including mounting file systems, starting services like systemd and journald, and configuring network interfaces (eth0, enp0s1). It also shows messages related to cgroups and kernel module loading. The log concludes with the prompt 'root@popcorn:~/linux-arm/msg_layer#'. The desktop environment visible in the background includes icons for a trash can, a terminal, and several application icons.

```
of required static device nodes for the current kernel.
[  3.529824] systemd[1]: Mounted Debug File System.
[  3.538741] systemd[1]: Mounted POSIX Message Queue File System.
[  3.560424] systemd[1]: systemd-modules-load.service: Main process exited, code=exited, status=1/FAILURE
[  3.569458] systemd[1]: Failed to start Load Kernel Modules.
[  3.582429] systemd[1]: systemd-modules-load.service: Unit entered failed state.
[  3.583345] systemd[1]: systemd-modules-load.service: Failed with result 'exit-code'.
[  3.637288] random: nonblocking pool is initialized
[  3.752943] systemd[1]: Started Nameserver information manager.
[  4.057492] systemd[1]: Started Journal Service.
[  5.089781] EXT4-fs (sda1): re-mounted. Options: errors=remount-ro
[  5.473782] systemd-journald[87]: Received request to flush runtime journal from PID 1
[  6.975925] e1000 0000:00:03.0 enp0s3: renamed from eth0
[  8.021124] cgroup: new mount options do not match the existing superblock, will be ignored
[  10.275078] cgroup: new mount options do not match the existing superblock, will be ignored
[  10.785794] cgroup: new mount options do not match the existing superblock, will be ignored
[  11.541064] cgroup: new mount options do not match the existing superblock, will be ignored
[  12.201644] cgroup: new mount options do not match the existing superblock, will be ignored
[  12.866922] e1000: enp0s3 NIC Link is Up 100 Mbps Full Duplex, Flow Control: RX
[  182.808525] msg_socket: loading out-of-tree module taints kernel.
[  182.819420] 10.0.2.16,10.0.2.17
[  182.819420] popcorn: Loading user configuration...
[  182.819420] popcorn: * 0: 10.0.2.16
[  182.819420] popcorn: 1: 10.0.2.17
[  183.590552] popcorn: Ready on TCP/IP
[  183.591535] popcorn: 1 joined, aarch64
root@popcorn:~/linux-arm/msg_layer#
root@ubuntu:~/linux-arm/msg_layer#
```

Tutorial Setup – Two Virtual Machines, Remote

- Open two (2) browser windows/tabs at
 - **VT LINK**
 - And login

Tutorial Setup – Two Virtual Machines, Local

- Next slides/steps cover this, but ...

Slide/Step needed for ...	Two VMs, Local	Two VMs, Remote	One VM
Host Machine Prerequisites	Yes	no	no
Prepare the host machine	Yes	no	no
Setup the host network bridge	Yes	no	no
Prepare the images	Yes	no	no
Start the x86 VM	Yes	no	Yes
Start the ARM VM	Yes	no	Yes
Check VMs connectivity	Yes	no	Yes
Connect to the VMs via ssh	Yes	no	Yes
Check each VM is up and running	Yes	Yes	Yes

Host Machine Prerequisites

- Two VMs local
 - Linux
 - Ubuntu/Debian
 - At least 50 GB of storage
 - Compressed images ~10GB total
 - Uncompressed images ~20GB each
 - Suggested 8 GB of RAM
 - 2GB per VM
 - Building from scratch requires up to 32GB (x86 VM)
- One VM (local)
 - At least 60 GB of storage
 - *.ova image ~22GB
 - Suggested 8 GB of RAM

Prepare the host machine

1. Download

- x86 VM image, and md5 checksum
- ARM VM image, and md5 checksum
- ARM EFI image

2. Install QEMU and bridge utilities

```
$ sudo apt-get update
```

```
$ sudo apt-get install qemu
```

```
$ sudo apt-get install bridge-utils
```

(only for Two VMs local)

Setup the host network bridge #1

- Make one of the host's network adaptors as a bridged interface, and tap VMs' NICs to the bridged interface
1. Edit */etc/network/interfaces* to setup host's eth0 as a bridged interface using DHCP

```
$ sudo vim /etc/network/interfaces
```

Before

```
...  
auto eth0  
iface eth0 inet dhcp
```

After

```
...  
auto eth0  
iface eth0 inet manual  
auto br0  
iface br0 inet dhcp  
    bridge_ports eth0  
    bridge_stp off  
    bridge_fd 0  
    bridge_maxwait 0
```

Setup the host network bridge #2

2. Edit */etc/sysctl.conf* to enable IP forwarding

```
$ sudo vim /etc/sysctl.conf
```

- Uncomment or add the following lines

```
net.ipv4.ip_forward=1
```

```
net.ipv4.conf.default.rp_filter=0
```

```
net.ipv4.conf.all.rp_filter=0\
```

3. Reload the configuration

```
$ sudo sysctl -p /etc/sysctl.conf
```

4. Restart the network subsystem

```
$ sudo /etc/init.d/networking restart
```

Prepare the images

1. Check MD5 checksum for each downloaded file

```
$ md5sum FILE_PATH > FILE_PATH.md5
```

```
$ diff FILE_PATH.md5 DOWNLOADED.md5
```

2. Uncompress the images

```
$ unzip x86.zip
```

```
$ unzip arm.zip
```

(not needed if remote access)

Start the x86 VM

- Launch QEMU on a terminal

```
$ sudo qemu-system-x86_64 \
--enable-kvm -m 2048 -smp 4 --nographic \
-netdev tap,id=x86nic,ifname=tap0 -device e1000,netdev=x86nic,mac=52:54:00:12:34:59 \
-hda x86.img
```

<https://www.cs.stevens.edu/~txing1/qemu-x86.sh>

- QEMU Options

- Sudo is necessary to use KVM and access tap0
- More RAM than 2GB can be specified with –m, more than 4 cores can be specified with –smp
- MAC must be specified to avoid MAC clashes

- **USER: root**

- **PWD: popcorn**

- **Exit QEMU with “Ctrl+A X”**

(not needed if remote access)

Start the ARM VM

- Launch QEMU on a terminal

<https://www.cs.stevens.edu/~txing1/qemu-arm.sh>

```
$ qemu-system-aarch64 \
  -cpu cortex-a57 -smp 4 -m 2048 -M virt -bios QEMU_EFI.fd --nographic \
  -netdev tap,id=armnic,ifname=tap1 -device e1000,netdev=armnic,mac=52:54:00:12:34:58 \
  -drive if=none,file=arm.img,id=hd0 -device virtio-blk-device,drive=hd0
```

- QEMU Options

- Sudo is necessary to use KVM and access tap1
- More RAM than 2GB can be specified with –m, more than 4 cores can be specified with –smp
- MAC must be specified to avoid MAC clashes

- USER: root
- PWD: popcorn
- Exit QEMU with “Ctrl+A X”

(not needed if remote access)

Check VMs connectivity

- QEMU serial line on terminal
- Detect the IP address on each VM with
 - \$ ifconfig
- Check connectivity pinging the host and each other

- On x86

```
$ ping ARM_IPADDR  
$ ping HOST_IPADDR
```

- On ARM

```
$ ping x86_IPADDR  
$ ping HOST_IPADDR
```

(not needed if remote access)

Connect to the VMs via ssh

- On the host
- Open a terminal and connect to the x86 via ssh
`$ ssh root@x86_IPADDR`
- Open another terminal and connect to the ARM via ssh
`$ ssh root@ARM_IPADDR`

Check each VM is up and running

- Check the number and type of processor on each VM with

```
$ cat /proc/cpuinfo
```

```
elliot@gigi: ~/popcorn
File Edit View Search Terminal Help
root@ubuntu:~# cat /proc/cpuinfo
processor : 0
BogoMIPS : 125.00
Features : fp asimd aes pmull sha1 sha2 crc32
CPU implementer : 0x41
CPU architecture: 8
CPU variant : 0x1
CPU part : 0xd07
CPU revision : 0

processor : 1
BogoMIPS : 125.00
Features : fp asimd aes pmull sha1 sha2 crc32
CPU implementer : 0x41
CPU architecture: 8
CPU variant : 0x1
CPU part : 0xd07
CPU revision : 0

processor : 2
BogoMIPS : 125.00
Features : fp asimd aes pmull sha1 sha2 crc32
CPU implementer : 0x41
CPU architecture: 8
CPU variant : 0x1
CPU part : 0xd07
CPU revision : 0

processor : 3
BogoMIPS : 125.00
Features : fp asimd aes pmull sha1 sha2 crc32
CPU implementer : 0x41
CPU architecture: 8
CPU variant : 0x1
CPU part : 0xd07
CPU revision : 0

root@ubuntu:~#
```

ARM

```
elliot@gigi: ~/popcorn
File Edit View Search Terminal Help
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall
      nx lm rep_good nopl pnpi vmx cx16 x2apic hypervisor lahf_lm retpoline kaiser tpr_shadow vnmi flexpriority ept vpid
bugs : cpu_meltdown spectre_v1 spectre_v2
bogomips : 5387.98
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 2
vendor_id : GenuineIntel
cpu family : 6
model : 6
model name : QEMU Virtual CPU version 2.5+
stepping : 3
microcode : 0x1
cpu_mhz : 2693.695
cache size : 4096 KB
physical id : 2
siblings : 1
core id : 0
cpu cores : 1
apicid : 2
initial apicid : 2
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall
      nx lm rep_good nopl pnpi vmx cx16 x2apic hypervisor lahf_lm retpoline kaiser tpr_shadow vnmi flexpriority ept vpid
bugs : cpu_meltdown spectre_v1 spectre_v2
bogomips : 5388.03
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 3
vendor_id : GenuineIntel
cpu family : 6
model : 6
model name : QEMU Virtual CPU version 2.5+
stepping : 3
microcode : 0x1
cpu MHz : 2693.695
cache size : 4096 KB
physical id : 3
siblings : 1
core id : 0
cpu cores : 1
apicid : 3
initial apicid : 3
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall
      nx lm rep_good nopl pnpi vmx cx16 x2apic hypervisor lahf_lm retpoline kaiser tpr_shadow vnmi flexpriority ept vpid
bugs : cpu_meltdown spectre_v1 spectre_v2
bogomips : 5388.03
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:
```

x86

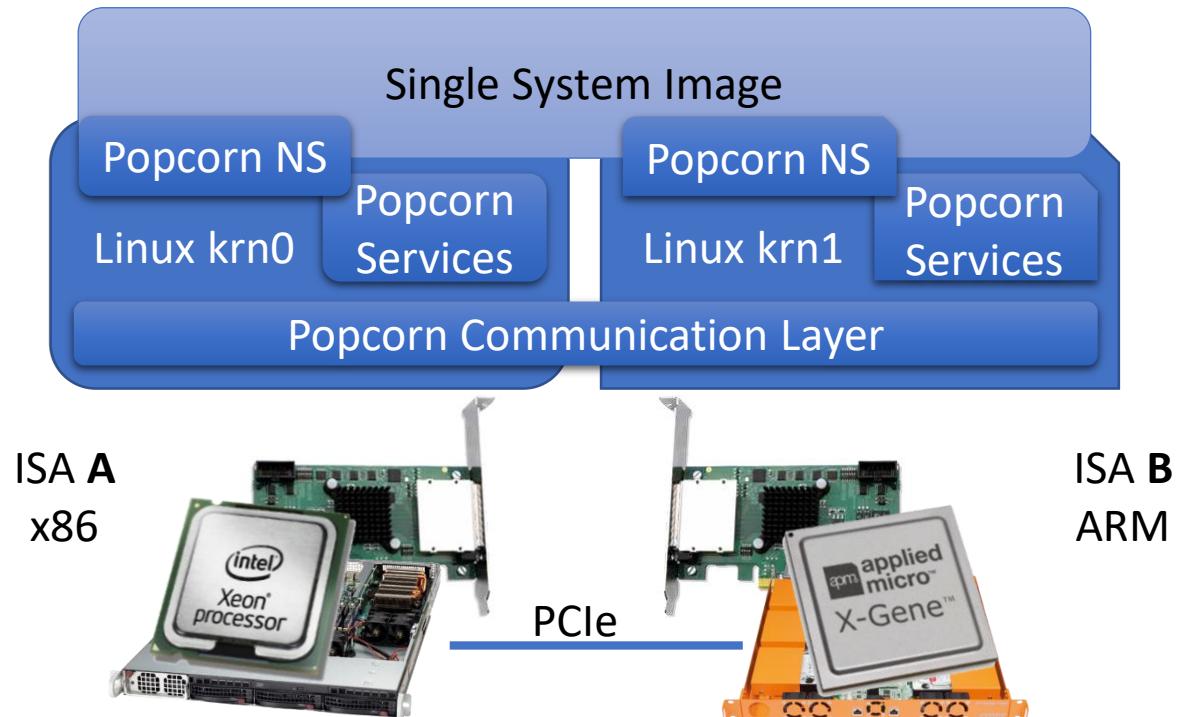
Tutorial Part 2a

Enter Popcorn Linux

(the Popcorn OS)

The Operating System

Discussed
before

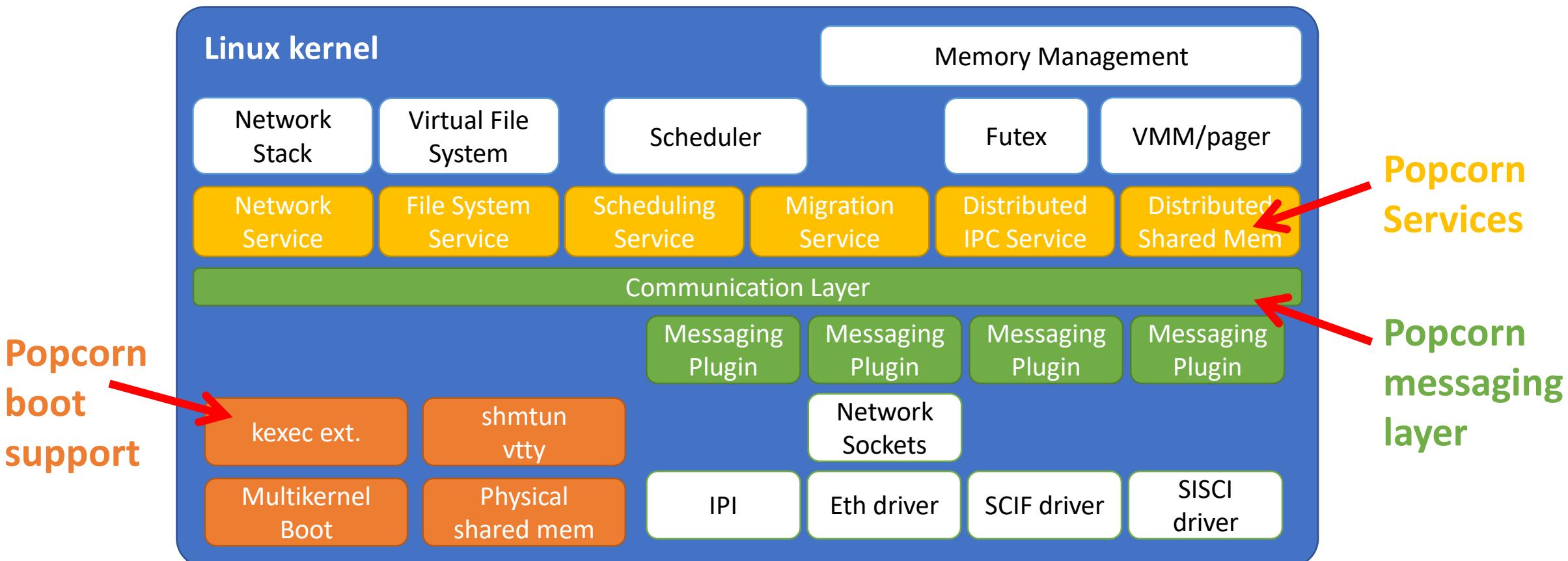


- **Single System Image**
 - Was based on Popcorn namespaces (NS)
 - Creates a single operating environment
 - Migrating app sees the same OS
 - Extends Linux among nodes
- **Distributed OS Services**
 - Task (thread and process) migration
 - Native code migration
 - Distributed shared memory
 - Distributed file system
- **Inter-kernel Communication Layer**
 - Performance critical component
 - low-latency and high-throughput
 - Exclusively kernel-space
 - Single format among ISAs

Software Architecture



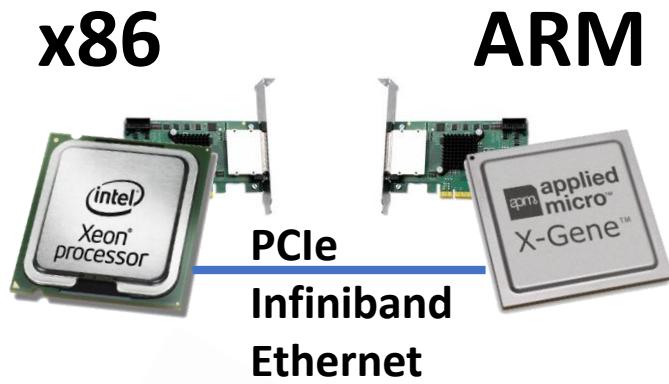
Application



Inter-kernel Messaging Layer



- Interconnects multiple (two or more) kernels via
 - Ethernet (TCP/IP)
 - Infiniband (RDMA)
 - Dolphin PCIe NTB (SISCI/DMA)
 - PCIe (Xeon-Phi SCIF/DMA, Broadcom Stringray DMA)
 - Shared memory (on SMP using IPI)
- Includes
 - Priority
 - Message routing
 - Fault-tolerance



OS Services and Single System Image



- Distributed OS services enable **inter-kernel application migration**
- **Single operating environment** illusion among kernels
 - A migrating application interacts with the **same operating environment**
 - As migration in SMP
 - CPU namespace
 - */proc/cpuinfo* lists all CPUs in the platform
 - Memory namespace
 - */proc/meminfo* lists all memory in the platform
 - PID namespace
 - File System namespace

Not fully supported by all versions

Overview

- Where are the sources?
- Make sure Popcorn kernel is running
- Load the messaging layer (socket)
- Make sure VMs are connected
- Explore shared resources

Where are the sources?

- Popcorn Linux kernel source code available in each VM
 - Self-hosted, no cross compiler is needed
- x86
 - /root/linux-x86
- ARM
 - /root/linux-arm

Make sure it is Popcorn Linux

- Check the Linux kernel version on each VM with

```
$ uname -a
```

- The Linux kernel version should include “popcorn”
4.4.173-popcorn+

```
root@ubuntu:~/linux-arm/msg_layer# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/linux-arm/msg_layer#
```

ARM

```
root@popcorn:~/linux-x86/msg_layer# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/linux-x86/msg_layer#
```

x86

Connect the VMs via Socket Popcorn Messaging

- x86

```
$ cd /root/linux-x86/msg_layer/  
$ sudo insmod ./msg_socket.ko ip=x86_IPADDR,ARM_IPADDR
```

- ARM

```
$ cd /root/linux-x86/msg_layer/  
$ sudo insmod ./msg_socket.ko ip=x86_IPADDR,ARM_IPADDR
```

- IP list

- node 0 (x86) goes first
- then node 1 (ARM)
- etc.

Make sure VMs are connected

- Check Popcorn peer kernels on each VM with

```
$ cat /proc/popcorn_peers
```

```
root@popcorn:~/linux-x86/msg_layer# cat /proc/popcorn_peers
* 0 192.168.10.100 NODE_IP
  1 192.168.10.101 NODE_IP
```

Shows an * on the current kernel

- Connection statistics are shown by

```
$ cat /proc/popcorn_stat
```

```
root@popcorn:~/linux-x86/msg_layer# cat /proc/popcorn_stat
      22          22  Total network I/O
    0.000        0.000  MB/s
      0            0  socket
```

Explore shared resources

- List all available CPUs on each VM with

```
$ cat /proc/cpuinfo
```

- Local CPUs are listed first
- Remote CPUs are listed after

The image shows two terminal windows side-by-side. Both windows have a dark background and white text. The left window is titled 'elliot@gigi: ~/popcorn' and the right window is titled 'elliot@gigi: ~/popcorn'. Both windows display the output of the command 'cat /proc/cpuinfo'. The output is organized into sections for local and remote CPUs. The first section lists local CPUs, followed by three sections for remote CPUs at addresses 0, 1, and 2. The right window is labeled 'ARM' in red text at the bottom right, and the left window is labeled 'x86' in red text at the bottom right.

```
elliot@gigi: ~/popcorn
File Edit View Search Terminal Help
bogomips      : 5387
TLB size       : 0 4K pages
clflush size   : 64
cache_alignment : 64
address sizes  : 40 bits physical, 48 bits virtual
****  Remote CPU at 0 ****
processor      : 1
vendor_id     : GenuineIntel
cpu_family    : 6
model         : 6
model name    : QEMU Virtual CPU version 2.5+
stepping       : 3
microcode     : 0x1
cpu MHz       : 2693
cache size    : 4096 kB
flags          : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge
                mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm rep_good n
                opl pnpi vmx cx16 x2apic hypervisor lahf_lm retpoline kaiser tpr_sh
                adow vnmi flexpriority ept vpid
bogomips      : 5387
TLB size       : 0 4K pages
clflush size   : 64
cache_alignment : 64
address sizes  : 40 bits physical, 48 bits virtual
****  Remote CPU at 1 ****
processor      : 0
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 1
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 2
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 3
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 3
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 3
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
root@ubuntu:~/linux-arm/msg_layer# 
elliot@gigi: ~/popcorn
File Edit View Search Terminal Help
cache size     : 4096 KB
physical id   : 3
siblings       : 1
core id       : 0
cpu cores     : 1
apicid        : 3
initial apicid: 3
fpu           : yes
fpu_exception  : yes
cpuid level   : 13
wp             : yes
flags          : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge
                mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm rep_good n
                opl pnpi vmx cx16 x2apic hypervisor lahf_lm retpoline kaiser tpr_sh
                adow vnmi flexpriority ept vpid
bugs          : cpu_meltdown spectre_v1 spectre_v2
bogomips      : 5387.56
clflush size   : 64
cache_alignment : 64
address sizes  : 40 bits physical, 48 bits virtual
power management:
****  Remote CPU at 1 ****
processor      : 0
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 1
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 2
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 3
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
****  Remote CPU at 1 ****
processor      : 3
model name    : ARMv8 Processor
BogoMIPS       : 125.00
Features        : fp asimd aes pmull sha1 sha2 crc32
CPU implementer: 0x41
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0xd07
CPU revision   : 0
root@popcorn:~/linux-x86/msg_layer# 
```

Tutorial Part 2b

Delving into Popcorn Linux

(Hello World!)

The Operating System – Task Migration

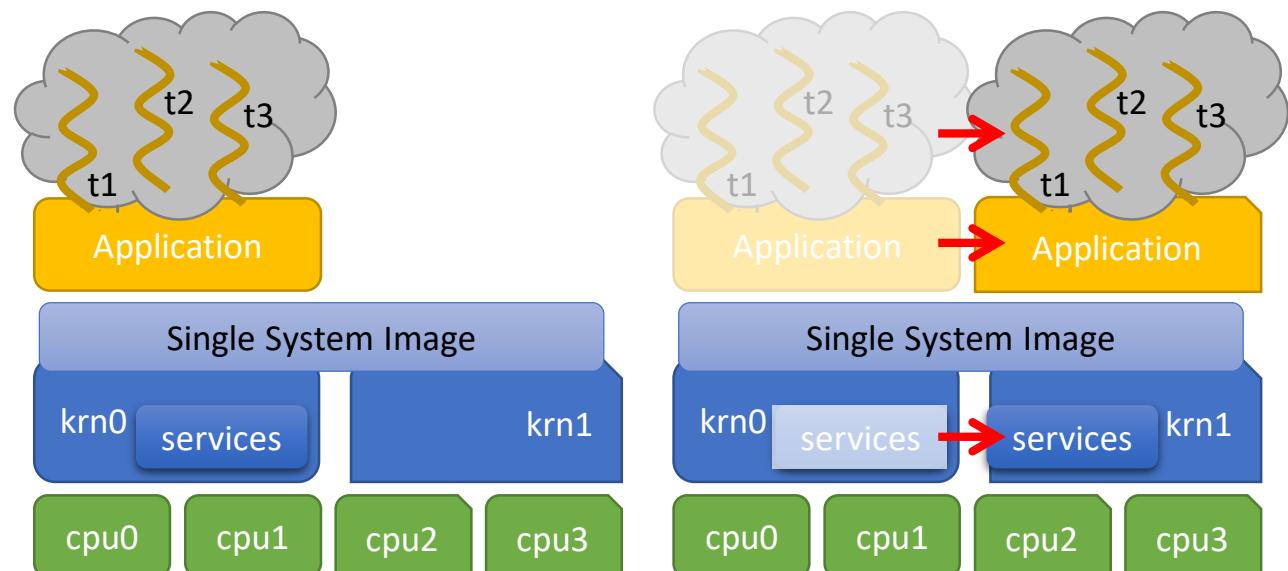


- Inter-kernel **thread** and **process** migration
- Execution Flow **Migration**
 - Native code migration – no intermediate rep. (e.g., no Java)
 - Maximum execution performance, supports all languages
 - Intra-kernel and inter-kernel shared memory, multi-threaded applications
 - POSIX/shared memory, OpenMP, MPI*, etc.
- Task State **Components**
 - User address space (e.g., .text, heap, stack, .data)
 - Operating system state (e.g., file descriptors)
 - **CPU state is ISA/microarchitecture dependent**



Process Migration

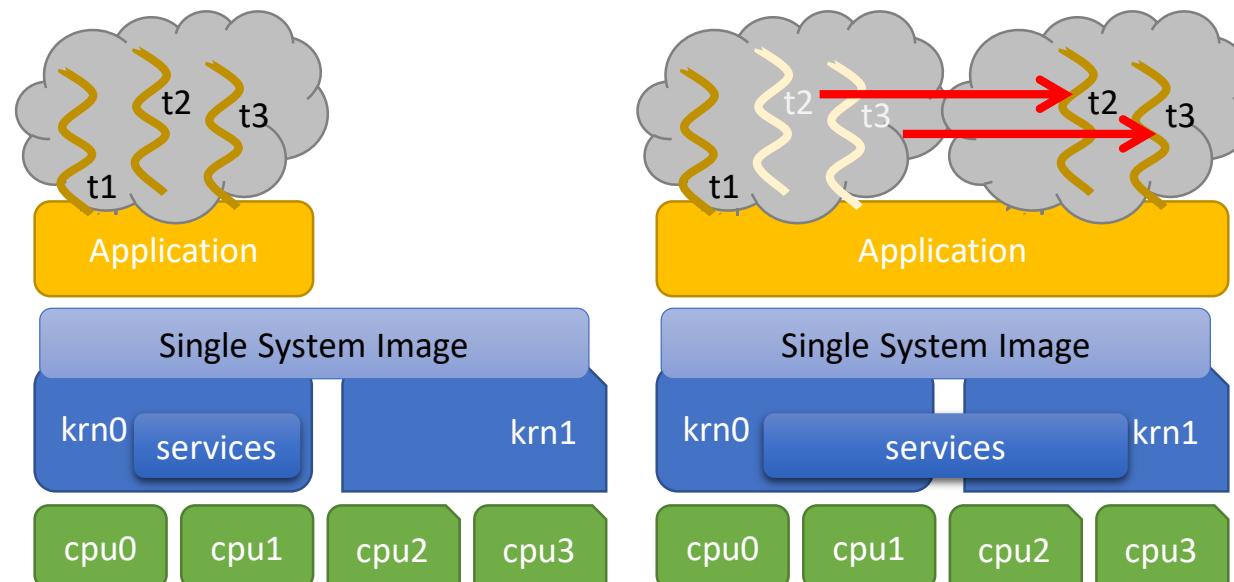
- Whole application is transferred between kernels
 - All threads, user- & kernel-state
- **No dependencies left on the origin kernel**



Thread Migration



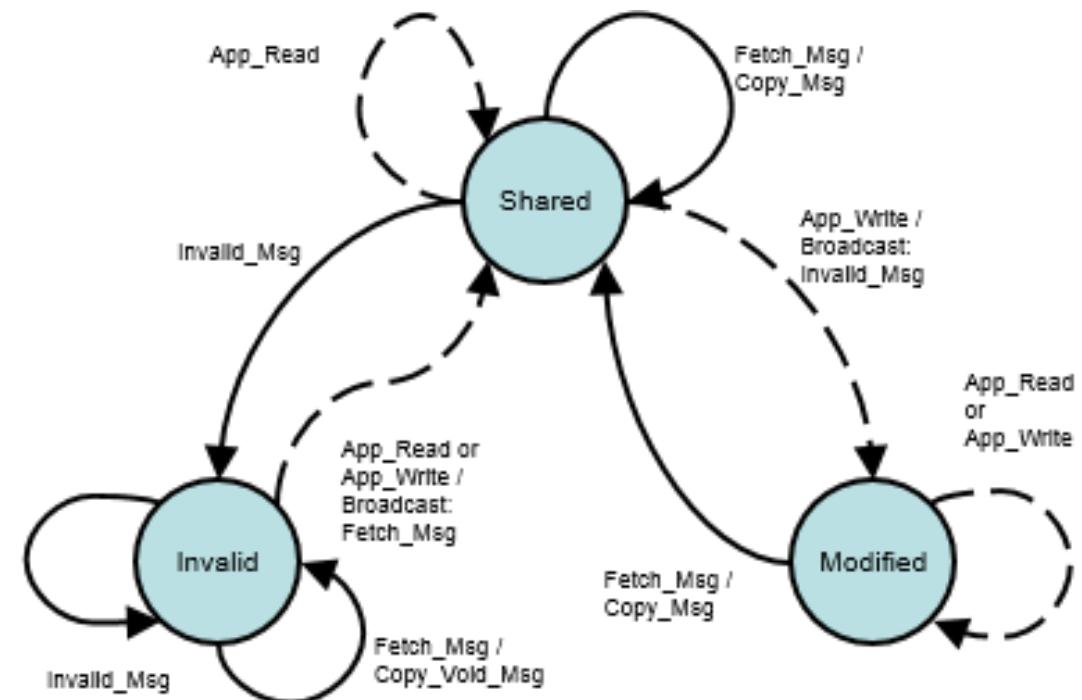
- Selected threads are transferred
 - Threads' state is transferred
- Kernels **coordinate** to maintain application state consistent



Thread Migration: Distributed Shared Memory Service



- Replicated virtual address space
- Kept consistent among kernels
- Page coherency protocol
 - Based on **Modified-Shared-Invalid (MSI)** cache coherency protocol
 - Memory page granularity instead of cache line granularity
 - Additional states to improve performance
 - Scaled from single node to multi-node



Overview

- Foundations of application development in Popcorn
- C code, Makefile, deploy and run, and advanced material of
 - Popcorn Hello World
 - Popcorn Hello World 2
 - Popcorn Hello World N
 - Popcorn Hello World S
 - Popcorn Hello World Id

Foundations of Application Development in Popcorn

- Writing anywhere
- Compiling on a single machine (x86), with one compiler (Popcorn LLVM)
 - Compiler sources in */root/popcorn_compiler*
 - Installed in the VM
 - No cross-compiler needed
- Makefile system developed for x86
- For each library (not lib C)
 - Need source code **for each target architecture**
 - Or the compiled library (*.a) **for each target architecture**
 - Popcorn binaries **are statically linked**
- Examples in */root/popcorn-tutorial* (installed in the VM)

Popcorn Hello World – C Code

```
#include <stdio.h>
#include "migrate.h"
int main (int argc, char* argv[])
{
    printf("hello, world (node %d)\n", current_nid());
    return 0;
}
```

In /root/popcorn-tutorial/helloworld

Popcorn Hello World - Makefile

```
BIN := popcorn-hello
```

```
SRC := popcorn-hello.c
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/helloworld`

Popcorn Hello World – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$ ./popcorn-hello
```

- On each node, it prints the node id (there is no migration)

```
root@ubuntu:~/popcorn-tutorial/helloworld# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld# ./popcorn-hello
hello, world (node 1)
root@ubuntu:~/popcorn-tutorial/helloworld# █
```

ARM

```
root@popcorn:~/popcorn-tutorial/helloworld# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworld# ./popcorn-hello
hello, world (node 0)
root@popcorn:~/popcorn-tutorial/helloworld# █
```

x86

In /root/popcorn-tutorial/helloworld

Popcorn Hello World 2 – C Code

```
#include <stdio.h>
#include “migrate.h”
int main (int argc, char* argv[])
{
    printf(“hello, world (node %d)\n”, current_nid());
    migrate(current_nid() ? 0 : 1, 0, 0);
    printf(“hello, world (node %d)\n”, current_nid());
    return 0;
}
```

In /root/popcorn-tutorial/helloworld2

Popcorn Hello World 2 - Makefile

```
BIN := popcorn-hello
```

```
SRC := popcorn-hello.c
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/helloworld2`

Same as before!

Popcorn Hello World 2 – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$ ./popcorn-hello
```

- It prints the node id before and after migration

```
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019  
aarch64 aarch64 aarch64 GNU/Linux  
root@ubuntu:~/popcorn-tutorial/helloworld# ./popcorn-hello  
hello, world (node 1)  
hello, world (node 0)  
root@ubuntu:~/popcorn-tutorial/helloworld#
```

ARM

```
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019  
x86_64 x86_64 x86_64 GNU/Linux  
root@popcorn:~/popcorn-tutorial/helloworld# ./popcorn-hello  
hello, world (node 0)  
hello, world (node 1)  
root@popcorn:~/popcorn-tutorial/helloworld#
```

x86

In /root/popcorn-tutorial/helloworld2

Popcorn Hello World 2 – Tracing Migrations

- x86 -> ARM
- Tracing information in the Linux kernel log buffer
 - \$ dmesg

```
root@ubuntu:~/popcorn-tutorial/helloworld# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld# dmesg
[ 5396.905684] remote_worker_main: [3040] for [2438/0]
[ 5396.905755] remote_worker_main: [3040] /root/popcorn-tutorial/helloworld/popcorn-hello
[ 5396.907629]
          ##### MIGRATED - [3041/1] from [2438/0]
[ 5396.970414] EXITED [3041] remote / 0x0
[ 5396.985796]
          TERMINATE [3040] with 0x0
[ 5396.985859] EXITED [3040] remote worker / 0x0
root@ubuntu:~/popcorn-tutorial/helloworld#
```

ARM

```
root@popcorn:~/popcorn-tutorial/helloworld# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworld# ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
root@popcorn:~/popcorn-tutorial/helloworld# dmesg
[ 5397.013636] ##### MIGRATE [2438] to 1
[ 5397.089208] process_remote_task_exit [2438] 0x0
[ 5397.089217] EXITED [2438] local / 0x0
[ 5397.089221] TERMINATE [3040/1] with 0x0
root@popcorn:~/popcorn-tutorial/helloworld#
```

x86

Popcorn Hello World 2 – Tracing Migrations

- ARM -> x86
- Tracing information in the Linux kernel log buffer
 - \$ dmesg

```
root@ubuntu:~/popcorn-tutorial/helloworld# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld# ./popcorn-hello
hello, world (node 1)
hello, world (node 0)
root@ubuntu:~/popcorn-tutorial/helloworld# dmesg
[ 5481.912327] ##### MIGRATE [3050] to 0
[ 5481.939068] process_remote_task_exit [3050] 0x0
[ 5481.939129] EXITED [3050] local / 0x0
[ 5481.939143] TERMINATE [2457/0] with 0x0
root@ubuntu:~/popcorn-tutorial/helloworld# □
```

ARM

```
root@popcorn:~/popcorn-tutorial/helloworld# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworld# dmesg
[ 5482.027650] remote_worker_main: [2457] for [3050/1]
[ 5482.027655] remote_worker_main: [2457] /root/popcorn-tutorial/helloworld/popcorn-hello
[ 5482.027825]
##### MIGRATED - [2458/0] from [3050/1]
[ 5482.049870] EXITED [2458] remote / 0x0
[ 5482.050775]
TERMINATE [2457] with 0x0
[ 5482.050786] EXITED [2457] remote worker / 0x0
root@popcorn:~/popcorn-tutorial/helloworld#
```

x86

Popcorn Hello World N – C Code

```
#include <stdio.h>
#include "migrate.h"
#define N 4
int main (int argc, char* argv[])
{ int i;
    printf("hello, world (node %d)\n", current_nid());
    for (i=0; i<N; i++) {
        migrate(get_nid() ? 0 : 1, 0, 0);
        printf("hello, world (node %d)\n", current_nid());
    }
    return 0;
}
```

In /root/popcorn-tutorial/helloworldN

Popcorn Hello World N - Makefile

```
BIN := popcorn-hello
```

```
SRC := popcorn-hello.c
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/helloworldN`

Same as before!

Popcorn Hello World N – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$ ./popcorn-hello
```

- It prints the node id before and after migration (multiple times)

```
root@ubuntu:~/popcorn-tutorial/helloworldN# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworldN# ./popcorn-hello
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
root@ubuntu:~/popcorn-tutorial/helloworldN# 
```

In /root/popcorn-tutorial/helloworldN

ARM

```
root@popcorn:~/popcorn-tutorial/helloworldN# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworldN# ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
root@popcorn:~/popcorn-tutorial/helloworldN# 
```

x86

Popcorn Hello World N – Tracing Migrations

- x86 -> ARM
- Tracing information in the Linux kernel log buffer
 - \$ dmesg

```
root@ubuntu:~/popcorn-tutorial/helloworldN# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworldN# dmesg
[ 7663.282534] remote_worker_main: [2942] for [3294/0]
[ 7663.282607] remote_worker_main: [2942] /root/popcorn-tutorial/h
elloworldN/popcorn-hello
[ 7663.291991]
        ##### MIGRATED - [2943/1] from [3294/0]
[ 7663.357785] ##### MIGRATE [2943] to 0
[ 7663.357877] EXITED [2943] remote / 0x200
[ 7663.372932]
        ##### MIGRATED - [2944/1] from [3294/0]
[ 7663.434347] ##### MIGRATE [2944] to 0
[ 7663.434427] EXITED [2944] remote / 0x200
[ 7663.442412]
        TERMINATE [2942] with 0x0
[ 7663.442471] EXITED [2942] remote worker / 0x0
root@ubuntu:~/popcorn-tutorial/helloworldN#
```

ARM

```
root@popcorn:~/popcorn-tutorial/helloworldN# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworldN# ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
root@popcorn:~/popcorn-tutorial/helloworldN# dmesg
[ 7657.890145] ##### MIGRATE [3294] to 1
[ 7657.969341] ### BACKMIG [3294] from [2943/1]
[ 7657.982583] ##### MIGRATE [3294] to 1
[ 7658.045895] ### BACKMIG [3294] from [2944/1]
[ 7658.053224] EXITED [3294] local / 0x0
[ 7658.053238] TERMINATE [2942/1] with 0x0
root@popcorn:~/popcorn-tutorial/helloworldN#
```

x86

Popcorn Hello World N – Tracing Migrations

- ARM -> x86
- Tracing information in the Linux kernel log buffer
 - \$ dmesg

```
root@ubuntu:~/popcorn-tutorial/helloworldN# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworldN# ./popcorn-hello
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
root@ubuntu:~/popcorn-tutorial/helloworldN# dmesg
[ 7766.617896] ##### MIGRATE [2964] to 0
[ 7766.650204] ### BACKMIG [2964] from [3303/0]
[ 7766.666374] ##### MIGRATE [2964] to 0
[ 7766.731095] ### BACKMIG [2964] from [3304/0]
[ 7766.738969] EXITED [2964] local / 0x0
[ 7766.738993] TERMINATE [3302/0] with 0x0
root@ubuntu:~/popcorn-tutorial/helloworldN# 
```

ARM

```
root@popcorn:~/popcorn-tutorial/helloworldN# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworldN# dmesg
[ 7761.230523] remote_worker_main: [3302] for [2964/1]
[ 7761.230530] remote_worker_main: [3302] /root/popcorn-tutorial/helloworldN/popcorn-hello
[ 7761.230713]
[ 7761.230713] ##### MIGRATED - [3303/0] from [2964/1]
[ 7761.261315] ##### MIGRATE [3303] to 1
[ 7761.261348] EXITED [3303] remote / 0x200
[ 7761.278513]
[ 7761.278513] ##### MIGRATED - [3304/0] from [2964/1]
[ 7761.342086] ##### MIGRATE [3304] to 1
[ 7761.342102] EXITED [3304] remote / 0x200
[ 7761.350794]
[ 7761.350794] TERMINATE [3302] with 0x0
[ 7761.350808] EXITED [3302] remote worker / 0x0
root@popcorn:~/popcorn-tutorial/helloworldN# 
```

x86

Popcorn Hello World S – C Code

```
#include <stdio.h>
#include <unistd.h>
#include "migrate.h"
#define N 4
int main (int argc, char* argv[])
{ int i;
  printf("hello, world (node %d)\n", current_nid());
  for (i=0; i<N; i++) {
    migrate(get_nid() ? 0 : 1, 0, 0);
    printf("hello, world (node %d)\n", current_nid());
    sleep(30);
  }
  return 0;
}
```

In /root/popcorn-tutorial/helloworldS

Popcorn Hello World S - Makefile

```
BIN := popcorn-hello
```

```
SRC := popcorn-hello.c
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/helloworldS`

Same as before!

Popcorn Hello World S – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$ ./popcorn-hello
```

- It prints the node id before and after migration (multiple times)

```
root@ubuntu:~/popcorn-tutorial/helloworld$ uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld$ ./popcorn-hello
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
root@ubuntu:~/popcorn-tutorial/helloworld$ 
```

In /root/popcorn-tutorial/helloworldS

ARM

```
root@popcorn:~/popcorn-tutorial/helloworld$ uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworld$ ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
root@popcorn:~/popcorn-tutorial/helloworld$ 
```

x86

Popcorn Hello World S – Tracing Processes

- x86 -> ARM
 - Process tracing information in /proc/popcorn_ps

TIP: use watch

```
Every 2.0s: cat /proc/popcorn_ps           Mon Oct 21 16:48:22 2019 root@popcorn:~/popcorn-tutorial/helloworld$ uname -a
R:    popcorn-hello  3981                  Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
          3982      0  3389      0      0          x86_64 x86_64 x86_64 GNU/Linux
          TOTAL      0      0
root@popcorn:~/popcorn-tutorial/helloworld$ ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
```

```
Every 2.0s: cat /proc/popcorn_ps           Mon Oct 21 16:48:52 2019  root@popcorn:~/popcorn-tutorial/helloworld$# uname -a
R:      popcorn-hello  3981                  TOTAL      0      0                               Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
                                                x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworld$# ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
hello, world (node 0)
```

ARM **x86**

```
Every 2.0s: cat /proc/popcorn_ps           Mon Oct 21 16:49:22 2019 root@popcorn:~/popcorn-tutorial/helloworld$# uname -a
R:      popcorn-hello  3981                 Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
          4070      0  3389      0      0             x86_64 x86_64 x86_64 GNU/Linux
          TOTAL      0      0
root@popcorn:~/popcorn-tutorial/helloworld$# ./popcorn-hello
hello, world (node 0)
```

Popcorn Hello World S – Tracing Processes

- ARM -> x86
 - Process tracing information in /proc/popcorn_ps
- ```
$ cat /proc/popcorn_ps
```

TIP: use  
watch

```
root@ubuntu:~/popcorn-tutorial/helloworld$ uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld$./popcorn-hello
hello, world (node 1)
hello, world (node 0)
```

```
Every 2.0s: cat /proc/popcorn_ps
Mon Oct 21 16:54:18 2019
R: popcorn-hello 3711
 3712 1 4170 0 0
 TOTAL 0 0
```

```
root@ubuntu:~/popcorn-tutorial/helloworld$ uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld$./popcorn-hello
hello, world (node 1)
hello, world (node 0)
hello, world (node 1)
```

```
Every 2.0s: cat /proc/popcorn_ps
Mon Oct 21 16:54:49 2019
R: popcorn-hello 3711
 TOTAL 0 0
```

**ARM**

```
root@ubuntu:~/popcorn-tutorial/helloworld$ uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworld$./popcorn-hello
hello, world (node 1)
```

```
Every 2.0s: cat /proc/popcorn_ps
Mon Oct 21 16:55:21 2019
R: popcorn-hello 3711
 3813 1 4170 0 0
 TOTAL 0 0
```

**x86**

# Popcorn Hello World S – /proc ? (x86 -> ARM)

```
root@ubuntu:~/popcorn-tutorial/helloworlds# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworlds# cat /proc/popcorn_ps
R: popcorn-hello 4554
 4555 0 4192 0 0
 TOTAL 0 0
root@ubuntu:~/popcorn-tutorial/helloworlds# ls /proc/4554/task/
4554 4555
root@ubuntu:~/popcorn-tutorial/helloworlds# cat /proc/4554/wchan
remote_worker_mainroot@ubuntu:~/popcorn-tutorial/helloworlds# cat
/proc/4555/wchan
hrtimer_nanosleeproot@ubuntu:~/popcorn-tutorial/helloworlds#
```

```
root@popcorn:~/popcorn-tutorial/helloworlds# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworlds# ./popcorn-hello
hello, world (node 0)
hello, world (node 1)
]
```

TIP: try  
**/proc/\*/wchan**

ARM

x86

# Popcorn Hello World S – /proc ? (ARM -> x86)

```
root@ubuntu:~/popcorn-tutorial/helloworlds# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworlds# ./popcorn-hello
hello, world (node 1)
hello, world (node 0)
[]
```

```
root@popcorn:~/popcorn-tutorial/helloworlds# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworlds# cat /proc/popcorn_ps
R: popcorn-hello 4650
 4651 1 4723 0 0
 TOTAL 0 0
root@popcorn:~/popcorn-tutorial/helloworlds# ls /proc/4650/task/
4650 4651
root@popcorn:~/popcorn-tutorial/helloworlds# cat /proc/4650/stack
[<fffffffff8109b9d0>] process_timeout+0x0/0x10
[<fffffffff8106f490>] check_preempt_curr+0x50/0x90
[<fffffffff81070cb4>] wake_up_new_task+0xf4/0x1a0
[<fffffffff810702d0>] default_wake_function+0x0/0x10
[<fffffffff8111586a>] remote_worker_main+0x21a/0x490
[<fffffffff81115650>] remote_worker_main+0x0/0x490
[<fffffffff81066b79>] kthread+0xc9/0xe0
[<fffffffff81066ab0>] kthread+0x0/0xe0
[<fffffffff813ec4c5>] ret_from_fork+0x55/0x80
[<fffffffff81066ab0>] kthread+0x0/0xe0
[<ffffffffffffffffff>] 0xfffffffffffffff
root@popcorn:~/popcorn-tutorial/helloworlds# cat /proc/4651/stack
[<fffffffff8109dacc>] hrtimer_nanosleep+0xbc/0x180
[<fffffffff8109cd70>] hrtimer_wakeup+0x0/0x30
[<fffffffff8109dbe6>] SyS_nanosleep+0x56/0x70
[<fffffffff813ec10a>] entry_SYSCALL_64_fastpath+0x1e/0x8e
[<ffffffffffffffffff>] 0xfffffffffffffff
root@popcorn:~/popcorn-tutorial/helloworlds# []
```

TIP: try  
`/proc/*/stack`

ARM

x86

# Popcorn Hello World Id – C Code

```
#include <stdio.h>
#include <unistd.h>
#include "migrate.h"
#define N 4
int main (int argc, char* argv[])
{ int i;
 printf("[%d,%d] hello, world (node %d)\n", getpid(), gettid(), current_nid());
 for (i=0; i<N; i++) {
 migrate(get_nid() ? 0 : 1, 0, 0);
 printf("[%d,%d] hello, world (node %d)\n", getpid(), gettid(), current_nid());
 }
 return 0;
}
```

In /root/popcorn-tutorial/helloworldId

# Popcorn Hello World Id - Makefile

```
BIN := popcorn-hello
```

```
SRC := popcorn-hello.c
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/helloworldId`

**Same as before!**

# Popcorn Hello World Id – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$./popcorn-hello
```

- It prints the node id before and after migration (multiple times)

```
root@ubuntu:~/popcorn-tutorial/helloworldId# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/helloworldId# ./popcorn-hello
[4944,4944] hello, world (node 1)
[4857,4858] hello, world (node 0)
[4944,4944] hello, world (node 1)
[4857,4859] hello, world (node 0)
[4944,4944] hello, world (node 1)
root@ubuntu:~/popcorn-tutorial/helloworldId#
```

In /root/popcorn-tutorial/helloworldId

**ARM**

```
root@popcorn:~/popcorn-tutorial/helloworldId# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/helloworldId# ./popcorn-hello
[4856,4856] hello, world (node 0)
[4941,4942] hello, world (node 1)
[4856,4856] hello, world (node 0)
[4941,4943] hello, world (node 1)
[4856,4856] hello, world (node 0)
root@popcorn:~/popcorn-tutorial/helloworldId#
```

**x86**

# Insights

- When a task migrates
  - It will create an additional task in the new kernel (helper thread)
- On the origin kernel
  - PID and TIDs never change
- On the remote kernel(s)
  - PID is fixed but TIDs change at every connection
- (This is not true for other verions)

# Tutorial Part 3a

## The Popcorn Compiler Framework

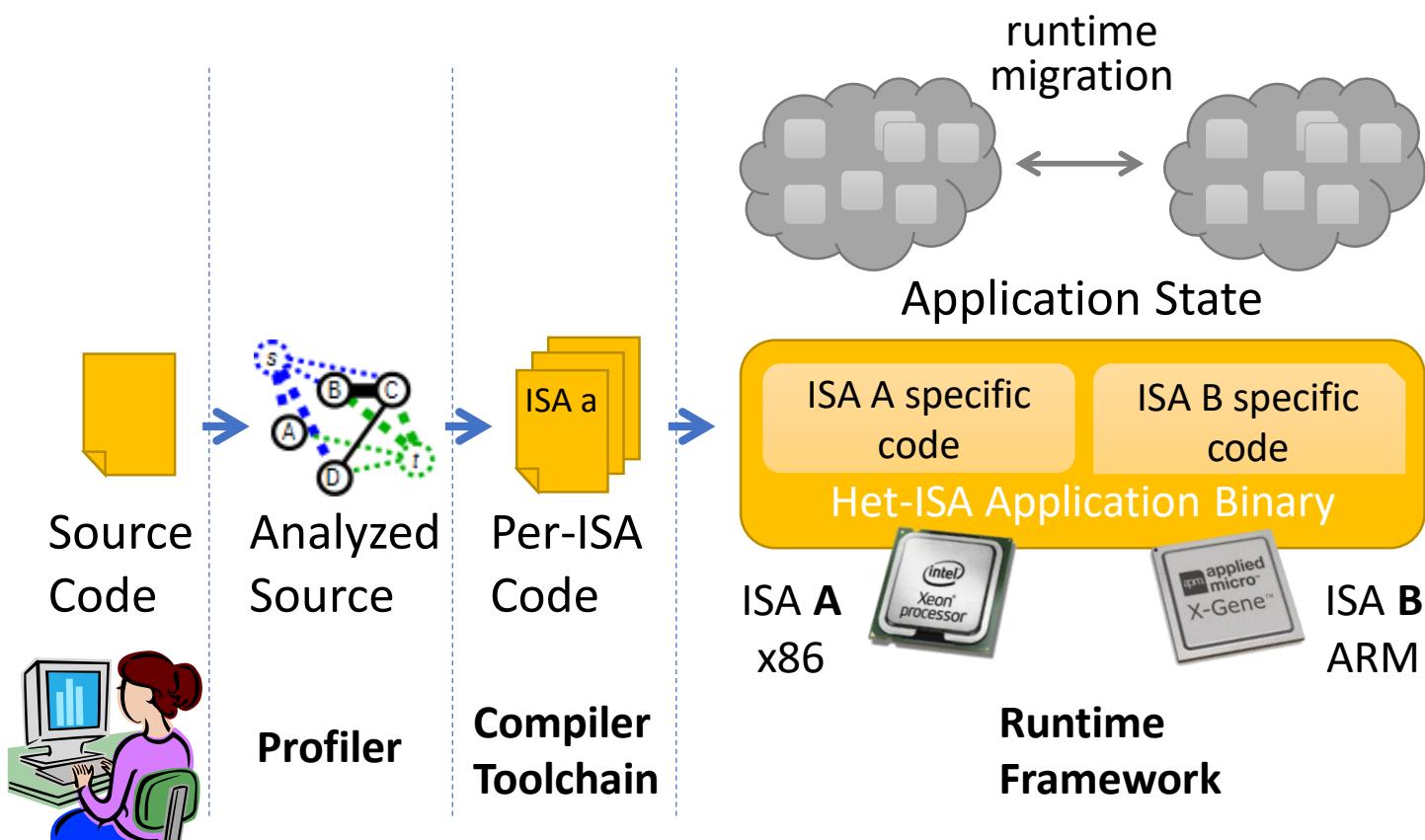
(aka Popcorn Compiler)

# Note

- For **homogeneous-ISA** deployments the compiler is not needed
- For **heterogeneous-ISA** deployments the Popcorn Linux Compiler framework (based on LLVM) is required

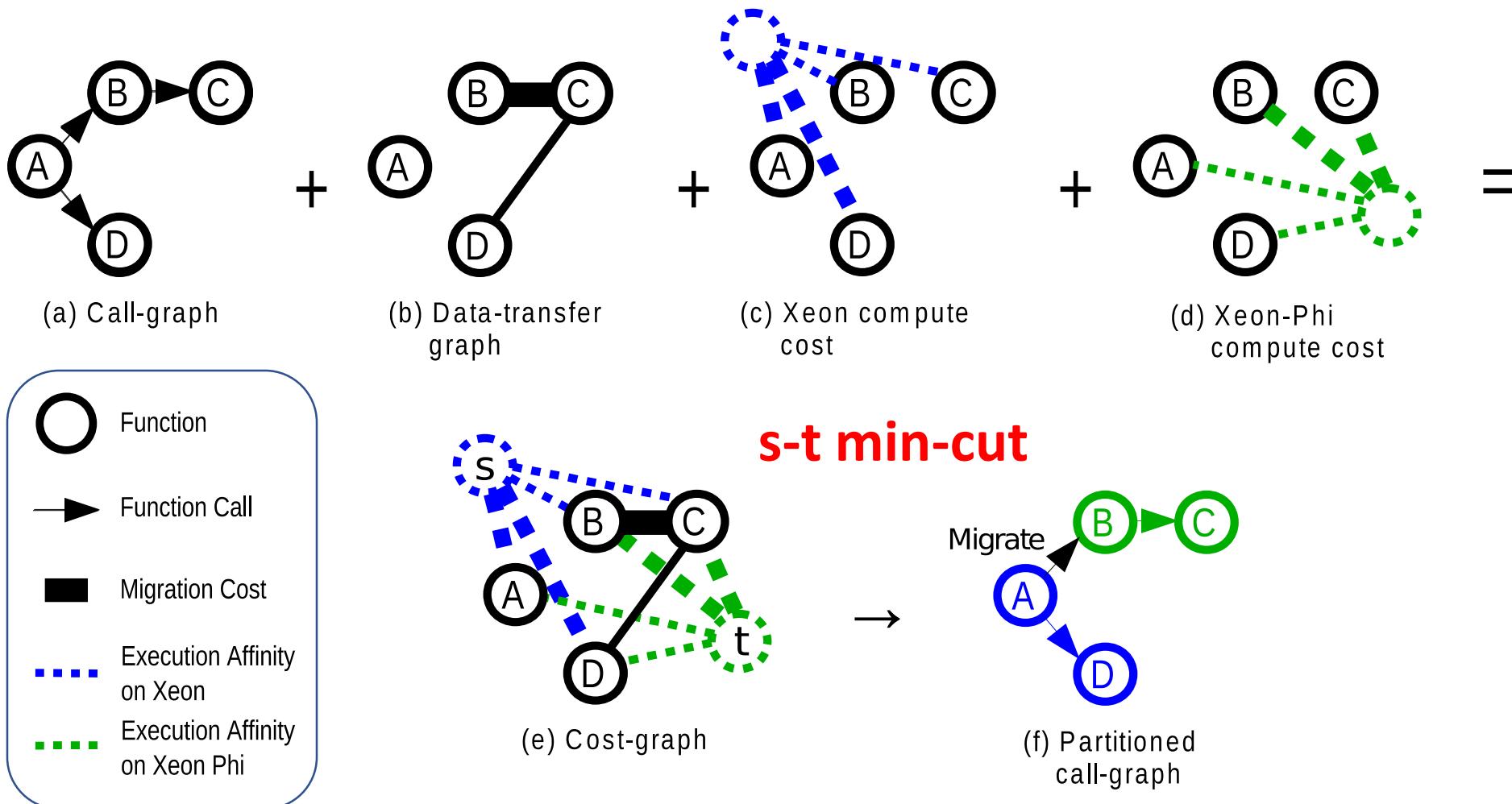
# The Compiler/Runtime

Discussed  
before



- **Profiler**
  - Performance and power profiles
  - Function and sub-function granularity
  - Output performance and power code indicators
- **Compiler/Linker Toolchain**
  - Output heterogenous-ISA binary (native)
    - Common address space (includes TLS)
    - Insert migration points (fun boundaries)
    - Add state transformation metadata
- **Runtime Framework**
  - Support task migration
  - Implements state transformation
    - Stack-transformation (rewriting)
    - Register-transformation

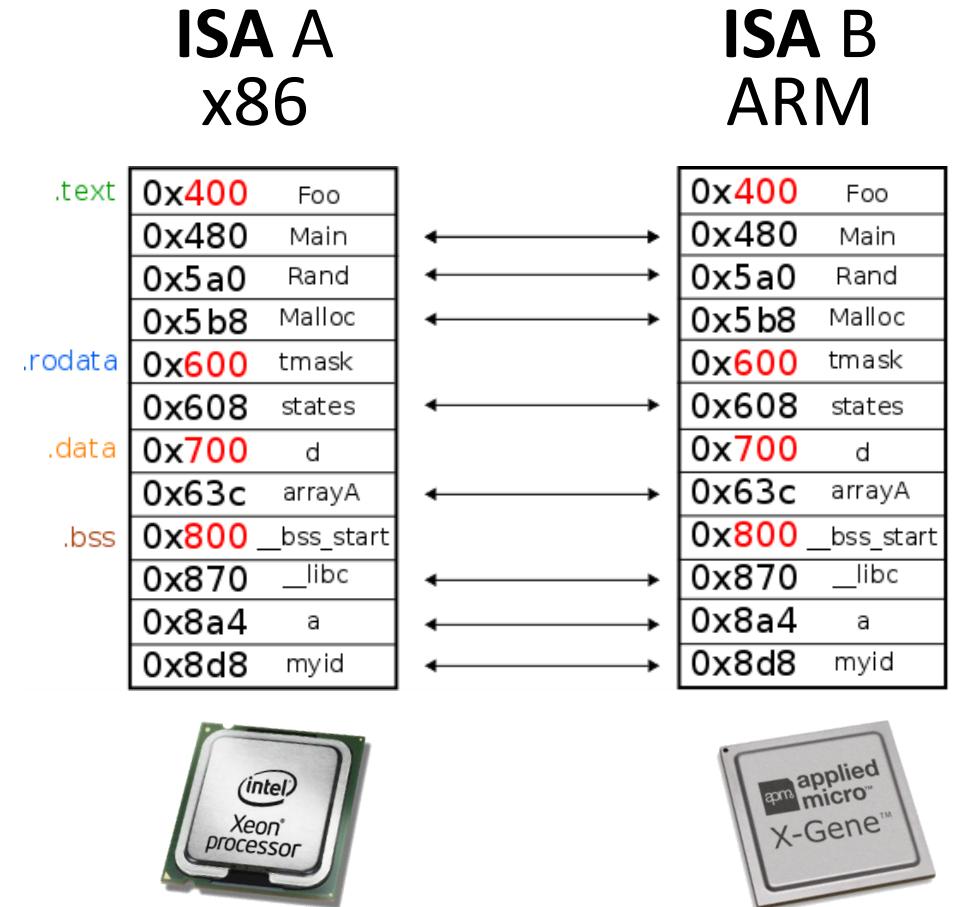
# Profiler – the case for Xeon-Xeon Phi



# Compiler/Linker Toolchain

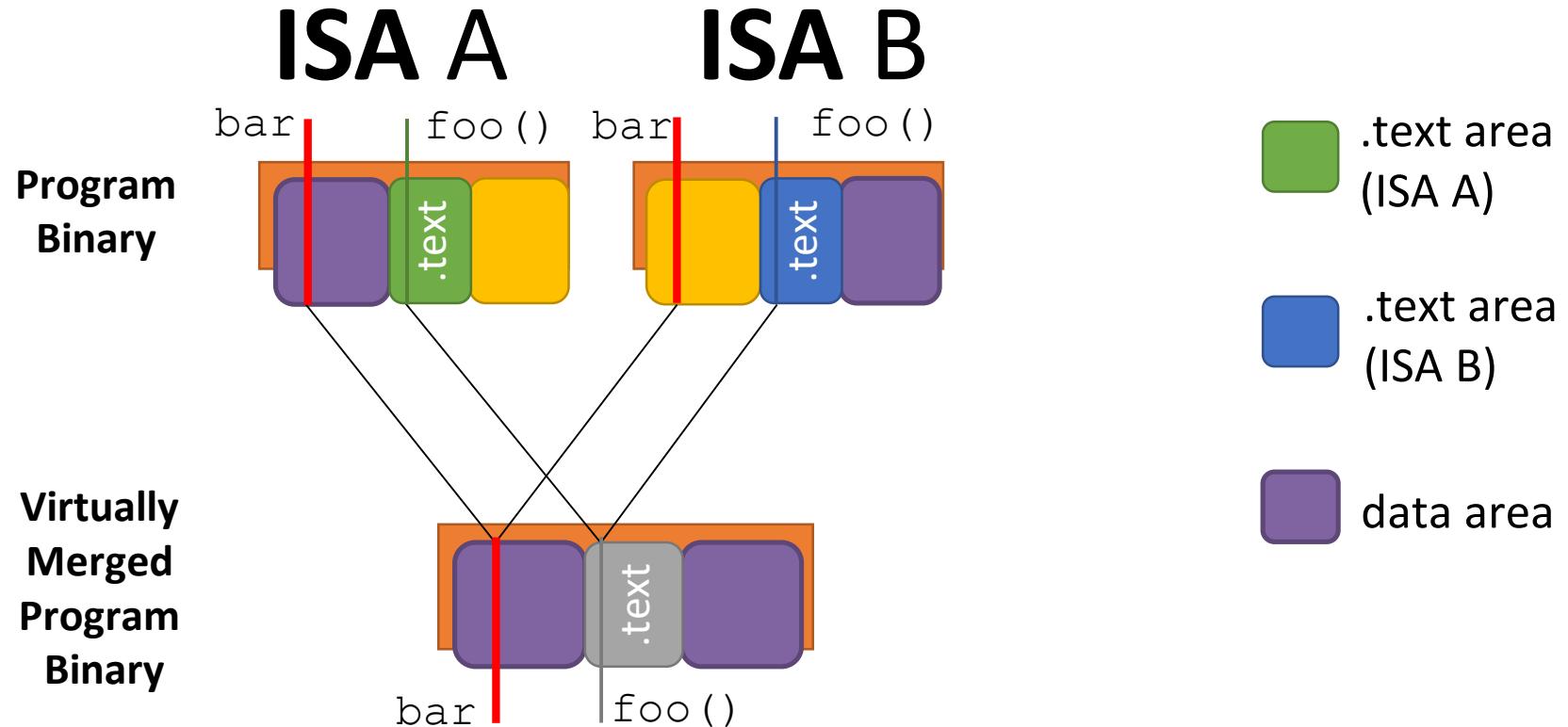


- Produces **program binaries** for each ISA
  - **Common address space**
    - Common type system
    - Each symbol at same virtual address on every ISA
    - No address space conversion!
  - **Common thread-local storage (TLS) layout**
    - x86\_64 layout forced
    - No TLS conversion!
  - Insert **migration points**
    - Cannot migrate at any instruction
  - Insert **state-transformation meta-data** in binaries
    - E.g., var properties, stack frame offsets



**Minimal runtime program state conversion!** (*stack and CPU state only*)

# Program Binaries at Runtime



# Migration Points – at Compile Time



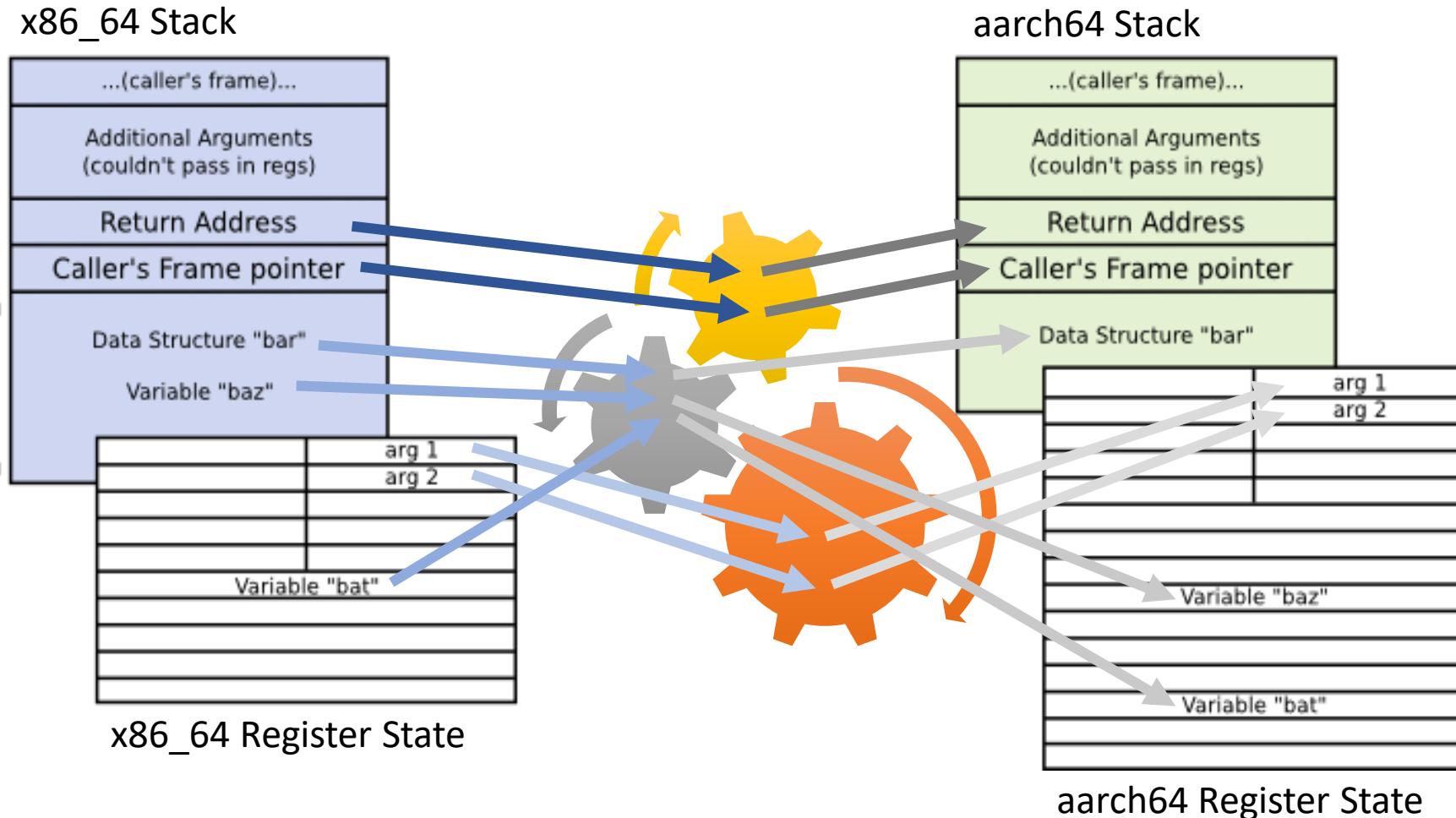
- Automatically inserted at **function boundaries** by compiler
  - Reduces program state conversion overheads
    - e.g., CPU registers
  - Circumvents the difficulty of finding program equivalence points
    - Sometimes impossible
- Migration points maybe also added
  - **Based on profiling**
    - Inside functions, but after outlining
    - Increases migration opportunities during application execution
  - **Manually**
    - Just use *migrate()*

# Migration Points – at Runtime



- Where a migration **can actually happen**
  - Based on a previous decision
    - Scheduler
    - Profiler
    - Developer
- When a migration point is reached
  1. Callback to check if a migration is needed
  2. Application state transformation: **stack transformation**
    - Converts current stack to destination architecture's ABI
    - Exploits DWARF frame information
    - All frames at once
  3. A migration is requested to the Operating System
    - CPU register re-mapping

# Runtime Migration – Stack Transformation



# Popcorn Migrate Me – C Code

```
#include <stdio.h>
#include <unistd.h>
#include "migrate.h"
#define N 30
void migrateme (int i) {
 printf("[%d,%d] iteration %d, node id %d\n", getpid(), gettid(), i, current_nid());
}
int main (int argc, char* argv[]) {
 int i;
 for (i=0; i<N; i++) {
 migrateme(i);
 sleep(1);
 }
 return 0;
}
```

In /root/popcorn-tutorial/migrateme

**NOTE: no call  
to migrate()**

# Popcorn Migrate Me - Makefile

```
BIN := popcorn-migrateme
```

```
SRC := popcorn-migrateme.c
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/migrateme`

# Popcorn Migrate Me – Will it Migrate?

- Check symbols in the executable binary

```
$ nm popcorn-migrateme
```

```
root@ubuntu:~/popcorn-tutorial/migrateme# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/migrateme# nm popcorn-migrateme | grep migrate
0000000000505380 T check_migrate
00000000005054a0 T clear_migrate_flag
00000000005053f0 T migrate
0000000000505610 T __migrate_fixup_aarch64
0000000000505690 T __migrate_fixup_powerpc64
000000000050559c T __migrate_fixup_x86_64
00000000007010f4 D __migrate_flag
0000000000501030 T migrateme
0000000000505440 T migrate_schedule
0000000000504de0 T __migrate_shim_internal
0000000000505540 t __migrate_sighandler
0000000000601000 R popcorn_migrateme__str__d_d_it
000000000051c120 T pthread_get_migrate_args
000000000051c10c T pthread_set_migrate_args
00000000005054c0 t __register_migrate_sighandler
0000000000601c9b R src_migrate__func__get_thread_pointer_get_thread
0000000000601c4f R src_migrate__func____migrate_shim_internal__migrate_
0000000000601c20 R src_migrate__str_1_0_____Unsu
0000000000601c41 R src_migrate__str_2_src_migrat
0000000000601c67 R src_migrate__str_3_Could_not_
0000000000601c81 R src_migrate__str_4_Could_not_
```

```
root@popcorn:~/popcorn-tutorial/migrateme# uname -a
Linux popcorn 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:50:01 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/migrateme# nm popcorn-migrateme | grep migrate
0000000000505380 T check_migrate
00000000005054a0 T clear_migrate_flag
00000000005053f0 T migrate
0000000000505610 T __migrate_fixup_aarch64
0000000000505690 T __migrate_fixup_powerpc64
000000000050559c T __migrate_fixup_x86_64
00000000007010f4 D __migrate_flag
0000000000501030 T migrateme
0000000000505440 T migrate_schedule
0000000000504de0 T __migrate_shim_internal
0000000000505540 t __migrate_sighandler
0000000000601000 R popcorn_migrateme__str__d_d_it
000000000051c120 T pthread_get_migrate_args
000000000051c10c T pthread_set_migrate_args
00000000005054c0 t __register_migrate_sighandler
0000000000601c9b R src_migrate__func__get_thread_pointer_get_thread
0000000000601c4f R src_migrate__func____migrate_shim_internal__migrate_
0000000000601c20 R src_migrate__str_1_0_____Unsu
0000000000601c41 R src_migrate__str_2_src_migrat
0000000000601c67 R src_migrate__str_3_Could_not_
0000000000601c81 R src_migrate__str_4_Could_not_
```

ARM

x86

# Popcorn Migrate Me – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$./popcorn-migrateme
```

- There is **no migration**

```
root@ubuntu:~/popcorn-tutorial/migrateme# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/migrateme# ./popcorn-migrateme
[1537,1537] iteration 0, node id 1
[1537,1537] iteration 1, node id 1
[1537,1537] iteration 2, node id 1
[1537,1537] iteration 28, node id 1
[1537,1537] iteration 29, node id 1
root@ubuntu:~/popcorn-tutorial/migrateme#
```

In /root/popcorn-tutorial/migrateme

ARM

```
root@popcorn:~/popcorn-tutorial/migrateme# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/migrateme# ./popcorn-migrateme
[1488,1488] iteration 0, node id 0
[1488,1488] iteration 1, node id 0
[1488,1488] iteration 2, node id 0
[1488,1488] iteration 28, node id 0
[1488,1488] iteration 29, node id 0
root@popcorn:~/popcorn-tutorial/migrateme#
```

x86

# Popcorn Migrate Me – How to migrate?

- x86 -> ARM
- Send signal 35 to the running application (need two terminals)

```
$ sudo kill -35 APPLICATION_PID
```

```
root@popcorn:~# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~# kill -35 1580
root@popcorn:~# █
```

x86

```
root@popcorn:~/popcorn-tutorial/migrateme# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/migrateme# ./popcorn-migrateme
[1580,1580] iteration 0, node id 0
[1580,1580] iteration 1, node id 0
[1580,1580] iteration 2, node id 0
[1580,1580] iteration 3, node id 0
[1580,1580] iteration 4, node id 0
[1580,1580] iteration 5, node id 0
[1580,1580] iteration 6, node id 0
[1580,1580] iteration 7, node id 0
[1580,1580] iteration 8, node id 0
[1580,1580] iteration 9, node id 0
[1580,1580] iteration 10, node id 0
[1611,1612] iteration 11, node id 1
[1611,1612] iteration 12, node id 1
[1611,1612] iteration 13, node id 1
[1611,1612] iteration 14, node id 1
[1611,1612] iteration 15, node id 1
[1611,1612] iteration 16, node id 1
```

x86

# Popcorn Migrate Me – How to migrate?

- ARM -> x86
- Send signal 35 to the running application (need two terminals)  
\$ sudo kill -35 APPLICATION\_PID

```
root@ubuntu:~/popcorn-tutorial/migrateme# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/migrateme# ./popcorn-migrateme
[1000,1000] iteration 0, node id 1
[1000,1000] iteration 1, node id 1
[1000,1000] iteration 2, node id 1
[1000,1000] iteration 3, node id 1
[1000,1000] iteration 4, node id 1
[1000,1000] iteration 5, node id 1
[1000,1000] iteration 6, node id 1
[1000,1000] iteration 7, node id 1
[1000,1000] iteration 8, node id 1
[1000,1000] iteration 9, node id 1
[1000,1000] iteration 10, node id 1
[848,849] iteration 11, node id 0
[848,849] iteration 12, node id 0
[848,849] iteration 13, node id 0
[848,849] iteration 14, node id 0
[848,849] iteration 15, node id 0
[848,849] iteration 16, node id 0
```

ARM

```
root@ubuntu:~# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~# kill -35 1000
root@ubuntu:~#
```

ARM

# Popcorn Migrate Me – How to migrate? (Again)

- x86 -> ARM -> x86

```
root@ubuntu:~# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019 aa
ranch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~# sshpass -p popcorn ssh root@192.168.10.48 "kill -35 2
040"
root@ubuntu:~# kill -35 2055
root@ubuntu:~# █
```

TRY: use ssh  
to signal

ARM

```
root@popcorn:~/popcorn-tutorial/migrateme# uname -a
Linux popcorn 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:50:01 EDT 20
19 x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/migrateme# ./popcorn-migrateme
[2040,2040] iteration 0, node id 0
[2040,2040] iteration 1, node id 0
[2040,2040] iteration 2, node id 0
[2040,2040] iteration 3, node id 0
[2040,2040] iteration 4, node id 0
[2040,2040] iteration 5, node id 0
[2040,2040] iteration 6, node id 0
[2040,2040] iteration 7, node id 0
[2054,2055] iteration 8, node id 1
[2054,2055] iteration 9, node id 1
[2054,2055] iteration 10, node id 1
[2054,2055] iteration 11, node id 1
[2054,2055] iteration 12, node id 1
[2054,2055] iteration 13, node id 1
[2054,2055] iteration 14, node id 1
[2054,2055] iteration 15, node id 1
[2054,2055] iteration 16, node id 1
[2054,2055] iteration 17, node id 1
[2054,2055] iteration 18, node id 1
[2040,2040] iteration 19, node id 0
[2040,2040] iteration 20, node id 0
[2040,2040] iteration 21, node id 0
[2040,2040] iteration 22, node id 0
```

x86

# Tutorial Part 3b

## Dissecting Popcorn Compiler Framework

(Advanced compiler topics)

# Compiling and Linking



- LLVM front-end
    - Emits IR
  - LLVM Optimizer
    - Run Popcorn passes
    - Emits IR
  - LLVM back-end
    - Emit Popcorn objects
- 
- Link ARM, and x86
    - Emits unaligned binaries
    - Emits symbol list
  - Aligner
    - Align Symbols
    - Emits linker scripts
  - Link ARM, and x86
    - Emits aligned binaries

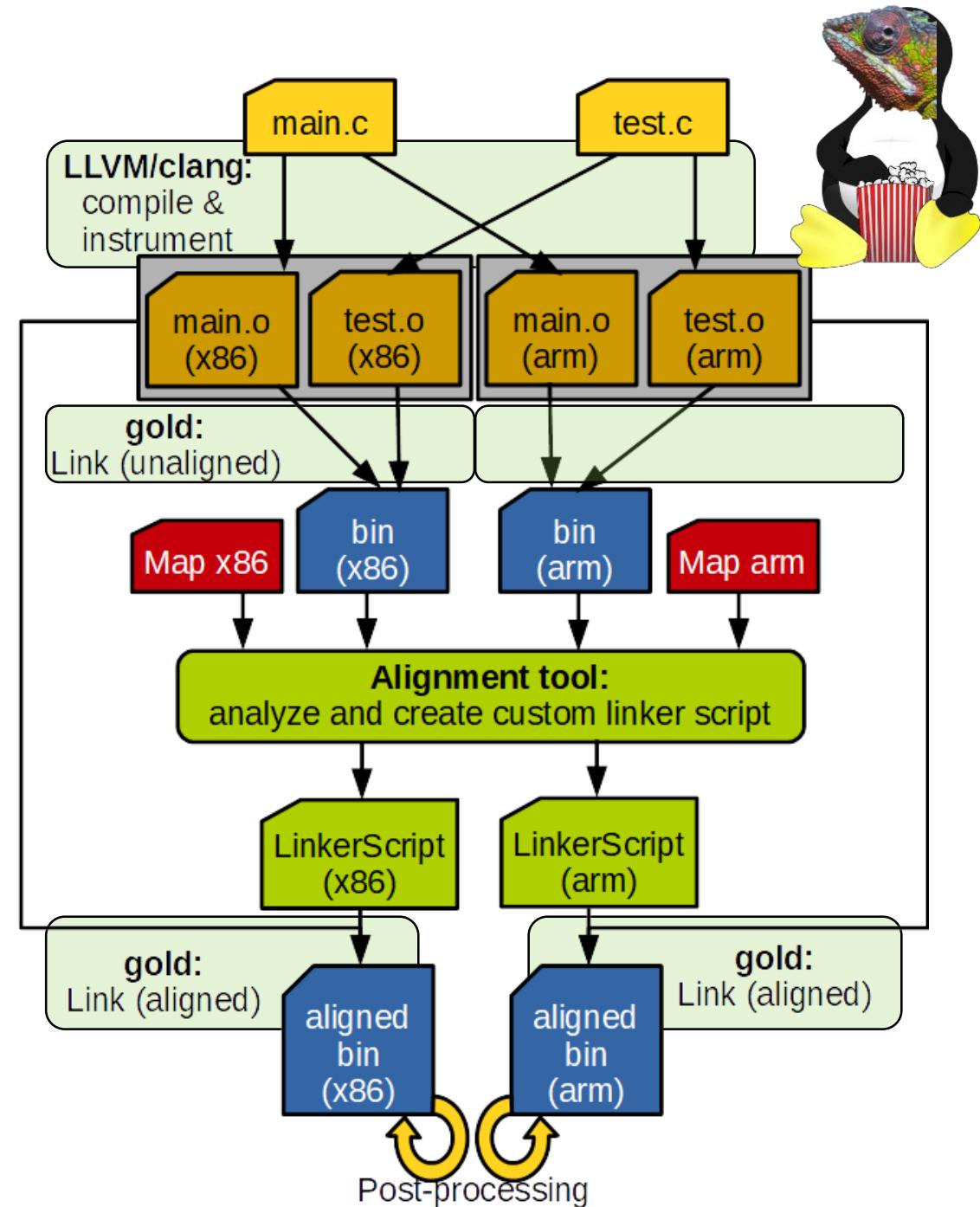
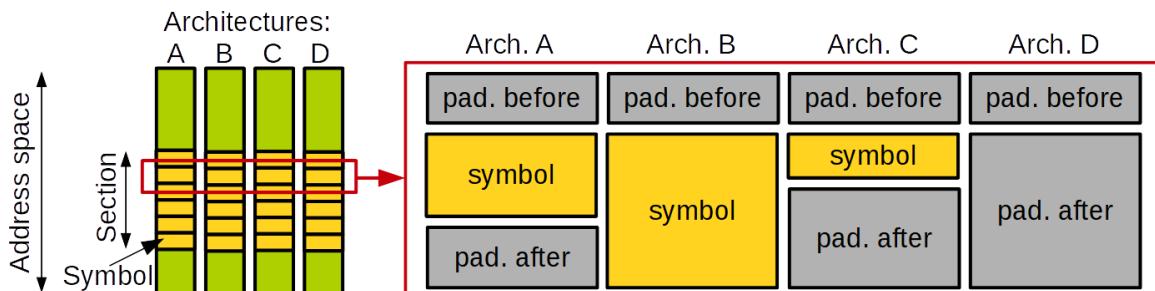
# The Compiler



- Based on LLVM
  - It compiles the same source file for multiple architectures at once
    - aarch64 and x86-64
  - Use *-popcorn-migratable*
  - Additionally
    - Need to link against Popcorn's lib C (musl), *-nostdinc -fno-common*
    - One section per function, one section per data, *-data-sections=1 -function-sections=1*
- *-popcorn-migratable* expands to multiple passes
  - *-insert-stackmaps*
  - *-popcorn-compat*
  - *-popcorn-instrument=migration*
  - *-migration-points*

# The Linker

- Based on gold
  - Enforces the same TLS layout
- Called twice
  - Firstly, without arguments
  - Secondly, with linker script
    - Produced by the Alignment tool



# NAS Parallel Benchmark (NPB) Examples

- Single thread
- Makefile modified to be compiled for Popcorn
- Integer Sort (**is**)
  - With explicit migration call
    - /root/popcorn-tutorial/npb-is
  - Compiler only migration points
    - /root/popcorn-tutorial/npb-is\_sig
- Discrete 3D fast Fourier Transform (**ft**)
  - With explicit migration call
    - /root/popcorn-tutorial/npb-ft
  - Compiler only migration points
    - /root/popcorn-tutorial/npb-ft\_sig

# NPB Examples – Makefile Fragment #1

```
Compiler
CC := $(POPCORN)/bin/clang
CXX := $(POPCORN)/bin/clang++
CFLAGS := -O0 -Wall -nostdinc -g
HET_CFLAGS := $(CFLAGS) -popcorn-migratable -fno-common -ftls-model=initial-exec

Linker
LD := $(POPCORN)/bin/ld.gold
LDFLAGS := -z noexecstack -z relro --hash-style=gnu --build-id -static
LIBS := /lib/crt1.o \
 /lib/libc.a \
 /lib/libopenpop.a \
 /lib/libmigrate.a \
 /lib/libstack-transform.a \
 /lib/libelf.a \
 /lib/libpthread.a \
 /lib/libc.a \
 /lib/libm.a

LIBGCC := --start-group -lgcc -lgcc_eh --end-group
```

```
Alignment
ALIGN := $(POPCORN)/bin/pyalign

Post-processing & checking
POST_PROCESS := $(POPCORN)/bin/gen-stackinfo
COMPRESS := $(POPCORN)/bin/compress
ALIGN_CHECK := $(POPCORN)/bin/check-align.py
STACKMAP_CHECK := $(POPCORN)/bin/check-stackmaps
```

# NPB Examples – Makefile Fragment #2

```
Locations
ARM64_POPCORN := $(POPCORN)/aarch64
ARM64_BUILD := build_aarch64
```

```
Generated files
ARM64_ALIGNED := $(BIN)_aarch64
ARM64_VANILLA := $(ARM64_BUILD)/$(ARM64_ALIGNED)
ARM64_OBJ := $(SRC:.c=_aarch64.o)
ARM64_MAP := $(ARM64_BUILD)/map.txt
ARM64_LD_SCRIPT := $(ARM64_BUILD)/aligned_linker_script_arm.x
ARM64_ALIGNED_MAP := $(ARM64_BUILD)/aligned_map.txt
```

```
Flags
ARM64_TARGET := aarch64-linux-gnu
ARM64_INC := -isystem $(ARM64_POPCORN)/include
ARM64_LDFLAGS := -m aarch64linux -L$(ARM64_POPCORN)/lib \
 -L$(ARM64_LIBGCC) \
 $(addprefix $(ARM64_POPCORN),$(LIBS)) $(LIBGCC)
```

**ARM**

```
Locations
X86_64_POPCORN := $(POPCORN)/x86_64
X86_64_BUILD := build_x86-64
```

```
Generated files
X86_64_ALIGNED := $(BIN)_x86-64
X86_64_VANILLA := $(X86_64_BUILD)/$(X86_64_ALIGNED)
X86_64_OBJ := $(SRC:.c=_x86_64.o)
X86_64_MAP := $(X86_64_BUILD)/map.txt
X86_64_LD_SCRIPT := $(X86_64_BUILD)/aligned_linker_script_x86.x
X86_64_ALIGNED_MAP := $(X86_64_BUILD)/aligned_map.txt
```

```
Flags
X86_64_TARGET := x86_64-linux-gnu
X86_64_INC := -isystem $(X86_64_POPCORN)/include
X86_64_LDFLAGS := -m elf_x86_64 -L$(X86_64_POPCORN)/lib \
 $(addprefix $(X86_64_POPCORN),$(LIBS)) \
 --start-group --end-group
```

**x86**

# NPB Examples – Makefile Fragment #3

```
%_aarch64.o: %.c
@echo "[CC] $<"
@$($CC) $(HET_CFLAGS) -c $(ARM64_INC) -o $(<..c=.o) $<
```

**ARM**

```
$(ARM64_VANILLA): $(ARM64_BUILD)/.dir $(ARM64_OBJ)
@echo "[LD] $@ (vanilla)"
@$($LD) -o $@ $(ARM64_OBJ) $(LDFLAGS) $(ARM64_LDFLAGS) -Map
$(ARM64_MAP)
```

```
$(ARM64_LD_SCRIPT): $(ARM64_VANILLA) $(X86_64_VANILLA)
@echo "[ALIGN] $@"
@$($ALIGN) --compiler-inst $(POPCORN) \
--x86-bin $(X86_64_VANILLA) --arm-bin $(ARM64_VANILLA) \
--x86-map $(X86_64_MAP) --arm-map $(ARM64_MAP) \
--output-x86-ls $(X86_64_LD_SCRIPT) --output-arm-ls $(ARM64_LD_SCRIPT)
```

```
$(ARM64_ALIGNED): $(ARM64_LD_SCRIPT)
@echo "[LD] $@ (aligned)"
@$($LD) -o $@ $(ARM64_OBJ) $(LDFLAGS) $(ARM64_LDFLAGS) -Map \
$(ARM64_ALIGNED_MAP) -T $<
```

```
%_x86_64.o: %_aarch64.o
```

**x86**

```
$(X86_64_VANILLA): $(X86_64_BUILD)/.dir $(X86_64_OBJ)
@echo "[LD] $@ (vanilla)"
@$($LD) -o $@ $(X86_64_OBJ) $(LDFLAGS) $(X86_64_LDFLAGS) -Map
$(X86_64_MAP)
```

```
$(X86_64_LD_SCRIPT): $(ARM64_LD_SCRIPT)
@echo "[ALIGN] $@"
```

```
$(X86_64_ALIGNED): $(X86_64_LD_SCRIPT)
@echo "[LD] $@ (aligned)"
@$($LD) -o $@ $(X86_64_OBJ) $(LDFLAGS) $(X86_64_LDFLAGS) \
-Map $(X86_64_ALIGNED_MAP) -T $<
```

# NPB IS – Explicit Migration

- x86 -> ARM -> x86

```
Every 2.0s: cat /proc/popcorn_ps Wed Oct 23 20:40:04 2019 root@popcorn:~/popc
R: is 11544
 11545 0 5857 4 17
 TOTAL 4 17
```

ARM

In /root/popcorn-tutorial/npb-is

```
root@popcorn:~/popcorn-tutorial/npb-is# uname -a
Linux popcorn 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:50:01 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/npb-is# ./is
[5857]
```

NAS Parallel Benchmarks (NPB3.3-SER) - IS Benchmark

Size: 8388608 (class A)
Iterations: 10

iteration  
1  
2  
3  
4  
5  
6  
7  
8  
9

10

IS Benchmark Completed
Class = A
Size = 8388608
Iterations = 10
Time in seconds = 32.86
Mop/s total = 2.55
Operation type = keys ranked
Verification = SUCCESSFUL
Version = 3.3.1
Compile date = 03 Nov 2017

Compile options:

CC = gcc
CLINK = \$(CC)
C\_LIB = -lm
C\_INC = -I../common
CFLAGS = -g -Wall -O3 -mcmodel=medium
CLINKFLAGS = -O3 -mcmodel=medium

Please send all errors/feedbacks to:
Center for Manycore Programming
cmp@aces.snu.ac.kr
http://aces.snu.ac.kr

```
[5857]
root@popcorn:~/popcorn-tutorial/npb-is#
```

# NPB IS – Explicit Migration

- ARM -> x86 -> ARM

```
root@ubuntu:~/popcorn-tutorial/npb-is# uname -a Every 2.0s: cat /proc
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/npb-is# ./is
[11676]

NAS Parallel Benchmarks (NPB3.3-SER) - IS Benchmark

Size: 8388608 (class A)
Iterations: 10

iteration
 1
```

ARM

In /root/popcorn-tutorial/npb-is

```
root@ubuntu:~/popcorn-tutorial/npb-is# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/npb-is# ./is
[11676]

NAS Parallel Benchmarks (NPB3.3-SER) - IS Benchmark

Size: 8388608 (class A)
Iterations: 10

iteration
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10

IS Benchmark Completed
Class =
Size = 8388608
Iterations =
Time in seconds =
Mop/s total =
Operation type =
Verification =
Version =
Compile date =
Compile options:
CC =
CLINK =
C_LIB =
C_INC =
CFLAGS =
CLINKFLAGS =

Please send all errors/feedbacks to:
Center for Manycore Programming
cmp@aces.snu.ac.kr
http://aces.snu.ac.kr

[11676]
root@ubuntu:~/popcorn-tutorial/npb-is#
```

# NPB IS – Implicit Migration

- Code as-is
- Current limitations
  - %f in *printf()* trigger segmentation faults
  - Avoid %f in *printf()*, use decimals
  - Application can migrate and terminate on any kernel
- Print the TID to know what thread to migrate

In /root/popcorn-tutorial/npb-is\_sig

# NPB FT – Explicit Migration

- X86 -> ARM -> x86

```
Every 2.0s: cat /proc/popcorn_ps Wed Oct 23 21:15:41 2019 root@popcorn:
R: ft 13137
 13138 0 7221 94 0
 TOTAL 94 0
```

ARM

In /root/popcorn-tutorial/npb-ft

```
root@popcorn:~/popcorn-tutorial/npb-ft# uname -a
Linux popcorn 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:50:01 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/npb-ft# ./ft

NAS Parallel Benchmarks (NPB3.3-SER-C) - FT Benchmark

Size : 64x 64x 64
Iterations : 6

[7221]
[13138]
Verification test for FT successful

FT Benchmark Completed.
Class = S
Size = 64x 64x 64
Iterations = 6
Linux popcorn+ 9.08
Time in seconds = 9.08
x86_64 x86_64 Mop/s total = 19.50
root@popcorn: Operation type = floating point
Verification = SUCCESSFUL
Version = 3.3.1
NAS Parallel Compile date = 03 Nov 2017

Size Compile options:
Iteration CC = gcc
 CLINK = $(CC)
 C_LIB = -lm
 C_INC = -I../common
 CFLAGS = -g -Wall -O3 -mcmmodel=medium
 CLINKFLAGS = -O3 -mcmmodel=medium
 RAND = randdp

Please send all errors/feedbacks to:
Center for Manycore Programming
cmp@aces.snu.ac.kr
http://aces.snu.ac.kr

root@popcorn:~/popcorn-tutorial/npb-ft#
```

# NPB FT – Explicit Migration

- ARM -> x86 -> ARM

```
root@ubuntu:~/popcorn-tutorial/npb-ft# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/npb-ft# ./ft
```

NAS Parallel Benchmarks (NPB3.3-SER-C) - FT Benchmark

```
Size : 64x 64x 64
Iterations : 6
```

```
[13231]
```

ARM

In /root/popcorn-tutorial/npb-ft

```
root@ubuntu:~/popcorn-tutorial/npb-ft# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/npb-ft# ./ft
```

NAS Parallel Benchmarks (NPB3.3-SER-C) - FT Benchmark

```
Size : 64x 64x 64
Iterations : 6
```

```
[13231]
[7400]
Verification test for FT successful
```

FT Benchmark Completed.

```
Class = S
Size = 64x 64x 64
Iterations = 6
Time in seconds = 3.07
Mop/s total = 57.76
Operation type = floating point
Verification = SUCCESSFUL
Version = 3.3.1
Compile date = 03 Nov 2017
```

Compile options:

```
CC = gcc
CLINK = $(CC)
C_LIB = -lm
C_INC = -I../common
CFLAGS = -g -Wall -O3 -mcmodel=medium
CLINKFLAGS = -O3 -mcmodel=medium
RAND = randdp
```

```
Please send all errors/feedbacks to:
Center for Manycore Programming
cmp@aces.snu.ac.kr
http://aces.snu.ac.kr
```

```
root@ubuntu:~/popcorn-tutorial/npb-ft#
```

# NPB FT – Implicit Migration

- Code as-is
- Current limitations
  - %f in *printf()* trigger segmentation faults
  - Avoid %f in *printf()*, use decimals
  - Application can migrate and terminate on any kernel
- Print the TID to know what thread to migrate

In /root/popcorn-tutorial/npb-ft\_sig

# Tutorial Part 4

# Pthread and Scheduling

(Orchestrate an application among multiple kernels)

# Pthread

- Popcorn supports Pthread
- Including synchronization primitives
  - *pthread\_create()*
  - *pthread\_join()*
  - ...
  - *pthread\_mutex\_init()*
  - *pthread\_mutex\_destroy()*
  - *pthread\_mutex\_lock()*
  - *pthread\_mutex\_unlock()*
- Popcorn uses its own lib C
  - Based on musl

# Overview

- C code, Makefile, Deploy and Run, and Advanced of
  - Popcorn Pthread 2
  - Popcorn Pthread S
  - Popcorn Migrate Me Pthread
  - Prime Sum
- Scheduling applications in Popcorn
  - User-space scheduler design, script, run
- Exercise

# Popcorn Pthread 2 – C Code

```
#define _GNU_SOURCE
#include <pthread.h>
#include <stdio.h>

#include "migrate.h"

void *thread(void *arg) {

 printf("thread, node id %d\n", current_nid());

 migrate((current_nid() ? 0 : 1), 0, 0);

 printf("thread, node id %d\n", current_nid());

 return NULL;
}
```

```
int main(void) {
 pthread_t pthread;
 if (pthread_create(&pthread, NULL, thread, 0) != 0) {
 perror("pthread_create");
 return 1;
 }

 printf("main, node id %d\n", current_nid());

 if (pthread_join(pthread, NULL) != 0) {
 perror("pthread_join");
 return 1;
 }

 return 0;
}
```

In /root/popcorn-tutorial/pthread2

# Popcorn Pthread 2- Makefile

```
BIN := popcorn-pthread
```

```
SRC := popcorn-pthread.c
```

```
LIBS := /lib/libpthread.a
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/pthread2`

# Popcorn Pthread 2 – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$./popcorn-pthread
```

- Only the newly created thread migrates

```
root@ubuntu:~/popcorn-tutorial/pthread2# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/pthread2# ./popcorn-pthread
main, node id 1
thread, node id 1
thread, node id 0
root@ubuntu:~/popcorn-tutorial/pthread2#
```

**ARM**

In /root/popcorn-tutorial/pthread2

```
root@popcorn:~/popcorn-tutorial/pthread2# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/pthread2# ./popcorn-pthread
main, node id 0
thread, node id 0
thread, node id 1
root@popcorn:~/popcorn-tutorial/pthread2#
```

**x86**

# Popcorn Pthread 2 – Tracing Migrations

- x86 -> ARM
- Tracing information in the Linux kernel log buffer
  - \$ dmesg

```
root@ubuntu:~/popcorn-tutorial/pthread2# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/pthread2# dmesg
[570.574100] remote_worker_main: [981] for [905/0]
[570.574187] remote_worker_main: [981] /root/popcorn-tutorial/pt
hread2/popcorn-pthread
[570.576411]
 ##### MIGRATED - [982/1] from [906/0]
[570.600954] EXITED [982] remote / 0x0
[570.608639]
 TERMINATE [981] with 0x0
[570.608693] EXITED [981] remote worker / 0x0
root@ubuntu:~/popcorn-tutorial/pthread2# █
```

ARM

```
root@popcorn:~/popcorn-tutorial/pthread2# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/pthread2# ./popcorn-pthread
main, node id 0
thread, node id 0
thread, node id 1
root@popcorn:~/popcorn-tutorial/pthread2# dmesg
[557.046722] ##### MIGRATE [906] to 1
[557.077372] process_remote_task_exit [906] 0x0
[557.077381] EXITED [906] local / 0x0
[557.084440] EXITED [905] local / 0x0
[557.084445] TERMINATE [981/1] with 0x0
root@popcorn:~/popcorn-tutorial/pthread2# █
```

x86

# Popcorn Pthread 2 – Tracing Migrations

- ARM -> x86
- Tracing information in the Linux kernel log buffer
  - \$ dmesg

```
root@ubuntu:~/popcorn-tutorial/pthread2# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/pthread2# ./popcorn-pthread
main, node id 1
thread, node id 1
thread, node id 0
root@ubuntu:~/popcorn-tutorial/pthread2# dmesg
[804.970913] ##### MIGRATE [999] to 0
[804.999560] process_remote_task_exit [999] 0x0
[804.999623] EXITED [999] local / 0x0
[805.007961] EXITED [998] local / 0x0
[805.007986] TERMINATE [922/0] with 0x0
root@ubuntu:~/popcorn-tutorial/pthread2#
```

ARM

```
root@popcorn:~/popcorn-tutorial/pthread2# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/pthread2# dmesg
[791.448004] remote_worker_main: [922] for [998/1]
[791.448009] remote_worker_main: [922] /root/popcorn-tutorial/pthread2/popcorn-pthread
[791.448162]
[791.448162] ##### MIGRATED - [923/0] from [999/1]
[791.473853] EXITED [923] remote / 0x0
[791.484141]
[791.484141] TERMINATE [922] with 0x0
[791.484152] EXITED [922] remote worker / 0x0
root@popcorn:~/popcorn-tutorial/pthread2#
```

x86

# Popcorn Pthread S – C Code

```
#define _GNU_SOURCE
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include "migrate.h"

#define N 4

void *thread(void *arg) {
 int i;
 printf("thread, node id %d\n", current_nid());
 for (i=0; i<N; i++) {
 migrate((current_nid() ? 0 : 1), 0, 0);
 printf("thread, node id %d\n", current_nid());
 sleep(30);
 }
 return NULL;
}
```

In /root/popcorn-tutorial/pthreadsS

```
int main(void) {
 pthread_t pthread;
 if (pthread_create(&pthread, NULL, thread, 0) != 0) {
 perror("pthread_create");
 return 1;
 }

 printf("main, node id %d\n", current_nid());

 if (pthread_join(pthread, NULL) != 0) {
 perror("pthread_join");
 return 1;
 }
 return 0;
}
```

# Popcorn Pthread S - Makefile

```
BIN := popcorn-pthread
```

```
SRC := popcorn-pthread.c
```

```
LIBS := /lib/libpthread.a
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/pthreadS`

# Popcorn Pthread S – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$./popcorn-pthread
```

- Only the newly created thread migrates

```
root@ubuntu:~/popcorn-tutorial/pthreads# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019 aa
rch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/pthreads# ./popcorn-pthread
main, node id 1
thread, node id 1
thread, node id 0
thread, node id 1
thread, node id 0
thread, node id 1
root@ubuntu:~/popcorn-tutorial/pthreads#
```

ARM

In /root/popcorn-tutorial/pthreads

```
root@popcorn:~/popcorn-tutorial/pthreads# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 20
19 x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/pthreads# ./popcorn-pthread
main, node id 0
thread, node id 0
thread, node id 1
thread, node id 0
thread, node id 1
thread, node id 0
root@popcorn:~/popcorn-tutorial/pthreads#
```

x86

# Popcorn Pthread S – Tracing Threads

- x86 -> ARM
- Send signal 35 to the running application (need two terminals)

```
$ sudo kill -35 APPLICATION_PID
```

|                                                        |                          |                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Every 2.0s: cat /proc/popcorn_ps                       | Tue Oct 22 16:22:48 2019 | root@popcorn:~/popcorn-tutorial/pthreads# uname -a<br>Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 20<br>19 x86_64 x86_64 x86_64 GNU/Linux<br>root@popcorn:~/popcorn-tutorial/pthreads# ./popcorn-pthread<br>main, node id 0<br>thread, node id 0<br>thread, node id 1 |
| R: popcorn-pthread 1069<br>1070 0 915 0 0<br>TOTAL 0 0 |                          |                                                                                                                                                                                                                                                                                         |

|                                      |                          |                                                                                                                                                                                                                                                                                                              |
|--------------------------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Every 2.0s: cat /proc/popcorn_ps     | Tue Oct 22 16:23:15 2019 | root@popcorn:~/popcorn-tutorial/pthreads# uname -a<br>Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 20<br>19 x86_64 x86_64 x86_64 GNU/Linux<br>root@popcorn:~/popcorn-tutorial/pthreads# ./popcorn-pthread<br>main, node id 0<br>thread, node id 0<br>thread, node id 1<br>thread, node id 0 |
| R: popcorn-pthread 1069<br>TOTAL 0 0 |                          |                                                                                                                                                                                                                                                                                                              |

x86

x86

|                                                        |                          |                                                                                                                                                                                                                                                                    |
|--------------------------------------------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Every 2.0s: cat /proc/popcorn_ps                       | Tue Oct 22 16:23:47 2019 | root@popcorn:~/popcorn-tutorial/pthreads# uname -a<br>Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 20<br>19 x86_64 x86_64 x86_64 GNU/Linux<br>root@popcorn:~/popcorn-tutorial/pthreads# ./popcorn-pthread<br>main, node id 0<br>thread, node id 0 |
| R: popcorn-pthread 1069<br>1155 0 915 0 0<br>TOTAL 0 0 |                          |                                                                                                                                                                                                                                                                    |

# Popcorn Pthread S – Tracing Threads

- ARM -> x86
- Send signal 35 to the running application (need two terminals)  
\$ sudo kill -35 APPLICATION\_PID

```
root@ubuntu:~/popcorn-tutorial/pthreads# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019 aa
ranch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/pthreads# ./popcorn-pthread
main, node id 1
thread, node id 1
thread, node id 0
root@ubuntu:~/popcorn-tutorial/pthreads# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019 aa
ranch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/pthreads# ./popcorn-pthread
main, node id 1
thread, node id 1
thread, node id 0
thread, node id 1
```

ARM

```
Every 2.0s: cat /proc/popcorn_ps Tue Oct 22 16:25:22 2019
R: popcorn-pthread 968
 969 1 1272 0 0
 TOTAL 0 0
Every 2.0s: cat /proc/popcorn_ps Tue Oct 22 16:25:50 2019
R: popcorn-pthread 968
 TOTAL 0 0
Every 2.0s: cat /proc/popcorn_ps Tue Oct 22 16:26:22 2019
R: popcorn-pthread 968
 1070 1 1272 0 0
 TOTAL 0 0
```

ARM

# Popcorn Migrate Me Pthread – C Code

```
#define _GNU_SOURCE
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include "migrate.h"
#define N 30
void migrateme (int i) {
 printf("[%d,%d] iteration %d, node id %d\n",
 getpid(), gettid(), i, current_nid());
}
void *thread(void *arg) {
 int i;
 for (i=0; i<N; i++) {
 migrateme(i);
 sleep(1);
 }
 return NULL;
}
```

In /root/popcorn-tutorial/migratemep

```
int main(void) {
 int ii;
 pthread_t pthread;
 if (pthread_create(&pthread, NULL, thread, 0) != 0) {
 perror("pthread_create");
 return 1;
 }
 for (ii=0; ii<N/2 ;ii++) {
 printf("main, node id %d\n", current_nid());
 sleep(2);
 }
 if (pthread_join(pthread, NULL) != 0) {
 perror("pthread_join");
 return 1;
 }
 return 0;
}
```

**NOTE: no call to migrate()**

# Popcorn Migrate Me Pthread - Makefile

```
BIN := popcorn-migratemep
```

```
SRC := popcorn-migratemep.c
```

```
LIBS := /lib/libpthread.a
```

```
include /usr/local/bin/share/build/Makefile
```

In </root/popcorn-tutorial/migratemep>

# Popcorn Migrate Me Pthread – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

- (It may request the password based on the setup)

2. Execute the application on each VM with

```
$./popcorn-migratemep
```

- Only the newly created thread migrates

```
root@ubuntu:~/popcorn-tutorial/migratemep# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019 aa
ranch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/migratemep# ./popcorn-migratemep
main, node id 1
[1675,1676] iteration 0, node id 1
[1675,1676] iteration 1, node id 1
main, node id 1
[1675,1676] iteration 2, node id 1
[1675,1676] iteration 3, node id 1
main, node id 1
[1675,1676] iteration 4, node id 1
[1675,1676] iteration 5, node id 1
```

ARM

```
root@popcorn:~/popcorn-tutorial/migratemep# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 20
19 x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/migratemep# ./popcorn-migratemep
main, node id 0
[1688,1689] iteration 0, node id 0
[1688,1689] iteration 1, node id 0
main, node id 0
[1688,1689] iteration 2, node id 0
[1688,1689] iteration 3, node id 0
main, node id 0
[1688,1689] iteration 4, node id 0
[1688,1689] iteration 5, node id 0
```

x86

In /root/popcorn-tutorial/migratemep

# Popcorn Migrate Me Pthread – Migrate Threads

- x86 -> ARM
- Send signal 35 to the running application (need two terminals)

```
$ sudo kill -35 APPLICATION_PID
```

```
root@popcorn:~# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 2019 x
86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~# kill -35 1754
root@popcorn:~#
```

x86

```
root@popcorn:~/popcorn-tutorial/migrateme# uname -a
Linux popcorn 4.4.137-popcorn+ #3 SMP Fri Oct 18 18:30:18 EDT 20
19 x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/migrateme# ./popcorn-migrateme
main, node id 0
[1753,1754] iteration 0, node id 0
[1753,1754] iteration 1, node id 0
main, node id 0
[1753,1754] iteration 2, node id 0
[1753,1754] iteration 3, node id 0
main, node id 0
[1753,1754] iteration 4, node id 0
[1753,1754] iteration 5, node id 0
main, node id 0
[1753,1754] iteration 6, node id 0
[1753,1754] iteration 7, node id 0
main, node id 0
[1753,1754] iteration 8, node id 0
[1753,1754] iteration 9, node id 0
main, node id 0
[1753,1754] iteration 10, node id 0
[1730,1731] iteration 11, node id 1
[1730,1731] iteration 12, node id 1
main, node id 0
[1730,1731] iteration 13, node id 1
[1730,1731] iteration 14, node id 1
main, node id 0
```

x86

# Popcorn Migrate Me Pthread – Migrate Threads

- ARM -> x86
- Send signal 35 to the running application (need two terminals)

```
$ sudo kill -35 APPLICATION_PID
```

```
root@ubuntu:~/popcorn-tutorial/migratemp# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 2019 aa
ranch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/migratemp# ./popcorn-migratemp
main, node id 1
[1812,1813] iteration 0, node id 1
[1812,1813] iteration 1, node id 1
main, node id 1
[1812,1813] iteration 2, node id 1
[1812,1813] iteration 3, node id 1
main, node id 1
[1812,1813] iteration 4, node id 1
[1812,1813] iteration 5, node id 1
main, node id 1
[1812,1813] iteration 6, node id 1
[1812,1813] iteration 7, node id 1
main, node id 1
[1812,1813] iteration 8, node id 1
[1812,1813] iteration 9, node id 1
main, node id 1
[1812,1813] iteration 10, node id 1
[1757,1758] iteration 11, node id 0
[1757,1758] iteration 12, node id 0
main, node id 1
[1757,1758] iteration 13, node id 0
[1757,1758] iteration 14, node id 0
main, node id 1
```

```
root@ubuntu:~# uname -a
Linux ubuntu 4.4.137-popcorn+ #5 SMP Fri Oct 18 19:18:36 EDT 201
9 aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~# kill -35 1813
root@ubuntu:~# █
```

ARM

ARM

# How to Automatically Control Migrations?

- Popcorn Linux **kernel-space scheduler**
  - Monitors load, and/or power consumption
    - Balances load, and/or power consumption
  - Triggers migrations
    - By sending signals
- Custom **user-space scheduler**
  - Implementable as script or executable

# Popcorn Prime Sum – C Code #1

NOTE: no call  
to migrate()

```
#define _GNU_SOURCE
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <malloc.h>
#include "migrate.h"
#define MAX 400000
#define N 4
static pthread_mutex_t sum_mutex;
static int sum = 0;

int is_prime(unsigned int n) {
 int i;
 if (n <= 1) return 1;
 for (i = 2; i < n; i++)
 if (n%i == 0) return 0;
 return 1;
}
```

In /root/popcorn-tutorial/primesum

```
int prime_sum(int index, int range) {
 int i, cnt = 0, nid = current_nid();
 printf("[%d,%d] thread index %d, range %d (node %d)\n",
 getpid(), gettid(), index, range, nid);
 for(i = index; i < (index + range); i++) {
 int cnid = current_nid();
 if (cnid != nid) {
 printf("[%d,%d] thread index %d, range %d (node %d)\n",
 getpid(), gettid(), index, range, cnid);
 nid = cnid;
 }
 if (is_prime(i)) cnt++;
 }
 printf("[%d,%d] thread sum %d (node %d)\n",
 getpid(), gettid(), cnt, nid);
 return cnt;
}
```

# Popcorn Prime Sum – C Code #2

NOTE: no call  
to migrate()

```
void *thread(void *arg) {
 int index = *(int*) arg, cnt = 0;
 cnt = prime_sum(index, (MAX/N));
 pthread_mutex_lock(&sum_mutex);
 sum += cnt;
 pthread_mutex_unlock(&sum_mutex);
 return NULL;
}

int main(void) {
 int err, i, ret=0;
 int *index = (int *)malloc(sizeof(int) * N);
 pthread_t *pt = (pthread_t *)malloc(sizeof(pthread_t) * N);
 if (pthread_mutex_init(&sum_mutex, NULL) != 0) {
 perror("pthread_mutex_init");
 ret = 1;
 goto error_exit;
 }
}
```

In /root/popcorn-tutorial/primesum

```
for (i = 0; i < N; i++) {
 index[i] = (MAX/N) * i;
 err = pthread_create(&(pt[i]), NULL, thread, &(index[i]));
 if (err != 0) {
 perror("pthread_create");
 ret = 1;
 goto error_exit1;
 }
}
for (i = 0; i < N; i++)
 pthread_join(pt[i], NULL);
printf("sum is %d\n", sum);

error_exit1:
 pthread_mutex_destroy(&sum_mutex);
error_exit:
 free(pt);
 free(index);
 return 0;
}
```

# Popcorn Prime Sum - Makefile

```
BIN := popcorn-primesum
```

```
SRC := popcorn-primesum.c
```

```
LIBS := /lib/libpthread.a
```

```
include /usr/local/bin/share/build/Makefile
```

In `/root/popcorn-tutorial/primesum`

# Popcorn Prime Sum – Deploy and Run

1. Sync the produced executables to all Popcorn kernels

```
$ psync -p=popcorn
```

2. Execute the application on each VM with

```
$./popcorn-primesum
```

```
root@ubuntu:~/popcorn-tutorial/primesum# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/primesum# time ./popcorn-prime_sum
[1028,1029] thread index 0, range 100000 (node 1)
[1028,1030] thread index 100000, range 100000 (node 1)
[1028,1032] thread index 300000, range 100000 (node 1)
[1028,1031] thread index 200000, range 100000 (node 1)
[1028,1029] thread index 0, sum 9594 (node 1)
[1028,1030] thread index 100000, sum 8392 (node 1)
[1028,1031] thread index 200000, sum 8013 (node 1)
[1028,1032] thread index 300000, sum 7863 (node 1)
sum is 33862

real 2m10.857s
user 5m46.548s
sys 0m17.036s
root@ubuntu:~/popcorn-tutorial/primesum#
```

**ARM**

```
root@popcorn:~/popcorn-tutorial/primesum# uname -a
Linux popcorn 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:50:01 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/primesum# time ./popcorn-prime_sum
[944,945] thread index 0, range 100000 (node 0)
[944,946] thread index 100000, range 100000 (node 0)
[944,947] thread index 200000, range 100000 (node 0)
[944,948] thread index 300000, range 100000 (node 0)
[944,945] thread index 0, sum 9594 (node 0)
[944,946] thread index 100000, sum 8392 (node 0)
[944,947] thread index 200000, sum 8013 (node 0)
[944,948] thread index 300000, sum 7863 (node 0)
sum is 33862

real 0m7.754s
user 0m18.168s
sys 0m0.284s
root@popcorn:~/popcorn-tutorial/primesum#
```

**x86**

# User-space Scheduler – Design

- **Balances** the number of Popcorn **threads** on two kernels
  1. Checks the number of Popcorn threads for each kernel
  2. If the number is
    - **Same**
      - Sleeps and goes back to the beginning
    - **Higher** on remote
      - Migrates threads from remote
    - **Lower** on remote
      - Migrates threads to remote

# User-space Scheduler – Script #1

```
list myself and all connected popcorn nodes
MYSELF=`cat /proc/popcorn_peers | awk '/^*[\t]+[0-9]+[\t]+[0-9]+\. [0-9]+\.[0-9]+[\t]+/{print $3}`;
if [-z "$MYSELF"] ; then
 echo "Error: malformed /proc/popcorn_peers" ; exit 1 ; fi
ALL=`cat /proc/popcorn_peers | sed 's/*/ /' | awk '/^*[\t]+[0-9]+[\t]+[0-9]+\. [0-9]+\.[0-9]+[\t]+/{print $2}`
if [-z "$ALL"] ; then
 echo "Error: malformed /proc/popcorn_peers (all)" ; exit 1 ; fi

KEYWORD="popcorn"

continuously check the status of the system
while true ; do
 # list the running popcorn threads on the current machine
 THREADS=`ps -A -T | grep $KEYWORD | awk '{if (NR>1) {print $2}}`;
 CURRENTNUM=`printf "$THREADS" | wc -l`;
```

In `/root/popcorn-tutorial/util/scheduler.sh`

# User-space Scheduler – Script #2

```
Balance the number of threads for each machine (TODO extend the algo to more than 2 machines)
for CURRENT in $ALL ;
do
 # skip if myself
 if ["$CURRENT" = "$MYSELF"] ; then continue ; fi

 # list the current popcorn threads on remote
 REMOTES=`${PASSWORD}ssh root@$CURRENT ps -A -T | grep $KEYWORD | awk '{if (NR>1) {print $2}}'`;
 REMOTENUM=`printf "$REMOTES" | wc -l`;

 # already balanced
 if [$REMOTENUM -eq $CURRENTNUM] ; then sleep 1 ; break ; fi
```

In /root/popcorn-tutorial/util/scheduler.sh

# User-space Scheduler – Script #3

```
remote has more threads - move threads to current
if [$REMOTENUM -gt $CURRENTNUM] ; then
 let TIME=($REMOTENUM-$CURRENTNUM)/2
 for THREAD in $REMOTES ; do
 ${PASSWORD}ssh root@$CURRENT kill -35 $THREAD
 let TIME=$TIME-1
 if [$TIME -eq 0] ; then break ; fi
 done ; break;
fi
remote has less threads - move threads to remote
if [$REMOTENUM -lt $CURRENTNUM] ; then
 let TIME=($CURRENTNUM-$REMOTENUM)/2
 for THREAD in $THREADS ; do
 kill -35 $THREAD
 let TIME=$TIME-1
 if [$TIME -eq 0] ; then break ; fi
 done ; break;
fi
done
done
```

In </root/popcorn-tutorial/util/scheduler.sh>

# User-space Scheduler – Run

- Run prime\_sum on ARM
- Run scheduler on x86

```
root@ubuntu:~/popcorn-tutorial/primesum# uname -a
Linux ubuntu 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:56:19 EDT 2019
aarch64 aarch64 aarch64 GNU/Linux
root@ubuntu:~/popcorn-tutorial/primesum# time ./popcorn-prime_sum
[1959,1960] thread index 0, range 100000 (node 1)
[1959,1961] thread index 100000, range 100000 (node 1)
[1959,1962] thread index 200000, range 100000 (node 1)
[1959,1963] thread index 300000, range 100000 (node 1)
[1582,1583] thread index 0, range 100000 (node 0)
[1582,1583] thread index 0, sum 9594 (node 0)
[1582,1601] thread index 100000, range 100000 (node 0)
[1582,1601] thread index 100000, sum 8392 (node 0)
[1582,1609] thread index 200000, range 100000 (node 0)
[1582,1609] thread index 200000, sum 8013 (node 0)
[1582,1622] thread index 300000, range 100000 (node 0)
[1582,1622] thread index 300000, sum 7863 (node 0)
sum is 33862
```

```
real 1m2.067s
user 1m41.428s
sys 0m23.416s
root@ubuntu:~/popcorn-tutorial/primesum#
```

ARM

```
root@popcorn:~/popcorn-tutorial/util# uname -a
Linux popcorn 4.4.137-popcorn+ #6 SMP Tue Oct 22 21:50:01 EDT 2019
x86_64 x86_64 x86_64 GNU/Linux
root@popcorn:~/popcorn-tutorial/util# ./scheduler.sh -p=popcorn
[]
```

x86

# User-space Scheduler – Exercise

- The proposed scheduler balances the number of Popcorn threads per kernel
- How to implement a scheduler that balances the load (or the power) among nodes?
  - Certainly, only Popcorn threads/processes can migrate between nodes
- How to handle odd number of threads?
  - E.g., periodically move between nodes



Popcorn  
**WITH HEXO**

# Concluding Remarks

# Popcorn Linux - Questions

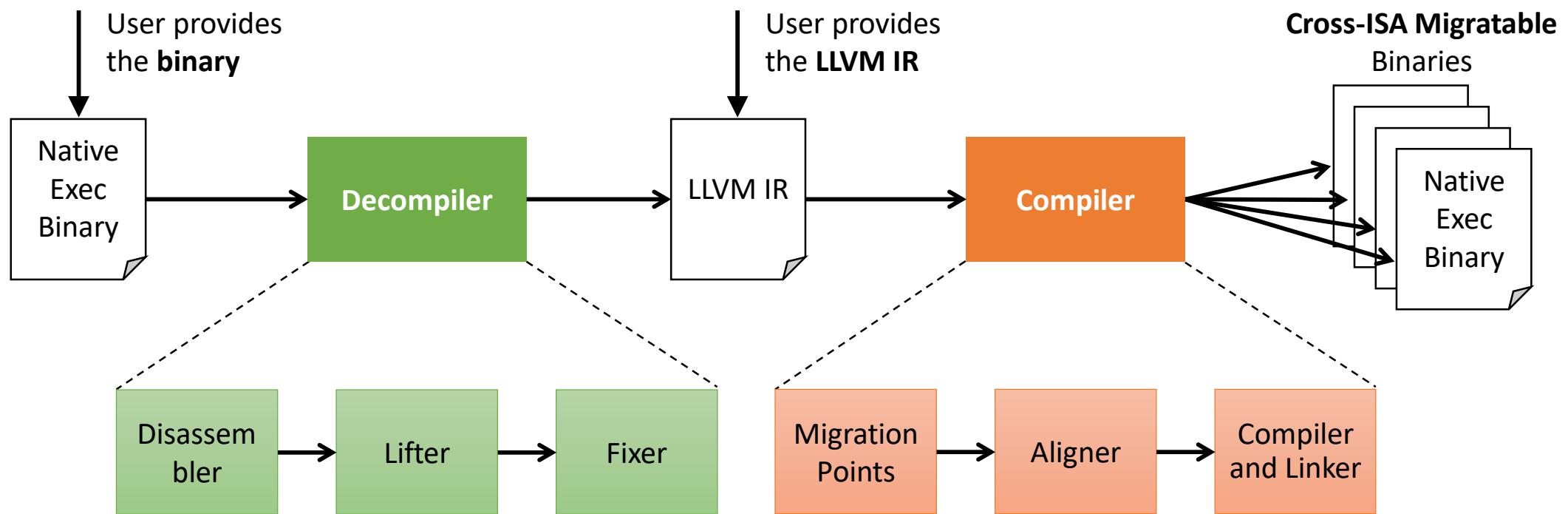


- **Usability? Somewhat easy**
  - No modifications required to source-files
  - Modifications required to the compilation scripts
- **Compatibility? Lacks support for ...**
  - Language virtual machines (but in principle it should work)
  - Server applications
  - Dynamically linked executable binaries
- **Deployability? Restrictive for large deployments**
  - Requires a specific compiler toolchain
  - Requires a specific Linux kernel version
    - That supports a handful of machines

# Welcome Edge-Containers! #1



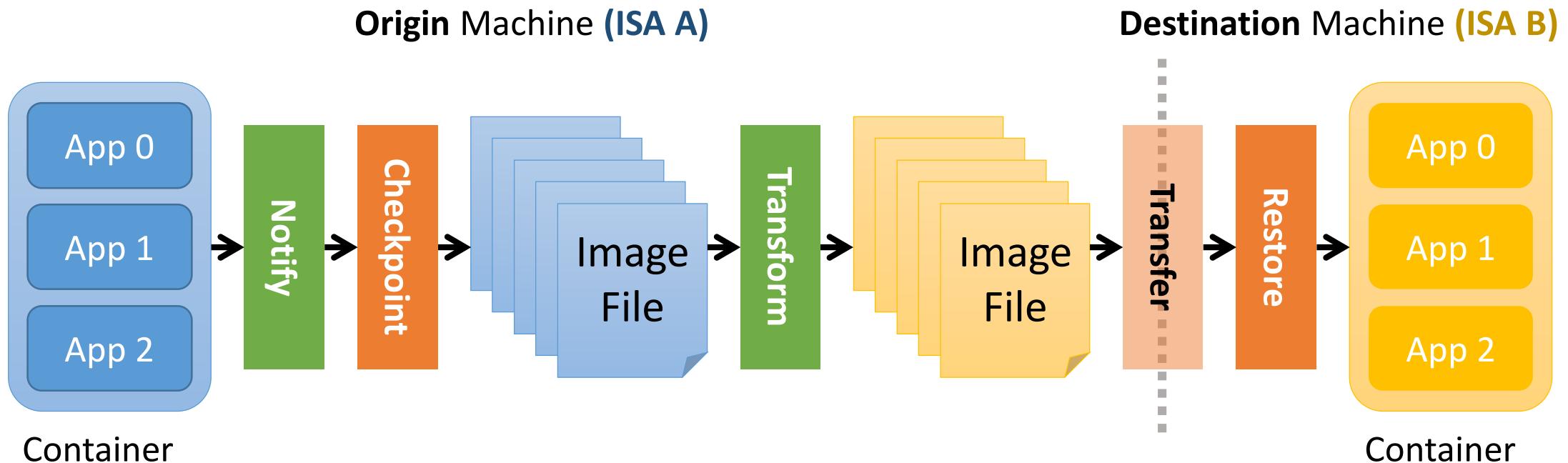
- Migrate applications among processors of diverse ISA
- **Without any modification to the application source code**
  - Applications are automatically transformed to run on multiple ISAs



# Welcome Edge-Containers! #2



- Migrate applications among processors of diverse ISA
- **Without any reliance of a specific Linux version (or kernel patches)**
  - Edge-Containers extends Linux's Checkpoint Restart in User Space



# Edge-Containers – Summary



- Enables programs migration between heterogeneous ISA CPUs
  - Forth and back
  - On any Linux kernel version that supports CRIU
  - Any binary
    - Currently FPU is not supported
- Simultaneous thread execution on multiple ISA CPUs requires Popcorn Linux kernel
  - But the automatic binary executable transformation can be exploited
- Will be released open source soon

# Current Research

- Multi-variant execution among heterogeneous-ISA units
  - Improving security leveraging ISA differences
- From Popcorn Linux to Popcorn KVM
  - Migrating an entire VM instead of just applications
- Shared Memory Multiple-kernel OS
  - Removing messaging at the base of multiple-kernel OSes
- **Come to our presentation @ PLOS (the first, after lunch)!**

# Some Research Directions

- DBT integration, “real” anytime migration
  - Enable scheduler-controlled migrations
- Extending to new architectures
  - ARM32, i386, RISC-V (internal effort), PowerPC
- Mitigating DSM overheads
  - Exploring hardware acceleration
- Unified stack layout
- Support for managed languages (Java, ...)
- Support for RUST
- Continuation vs Stack –based languages
- ...

[abarbala@stevens.edu](mailto:abarbala@stevens.edu)

[polivier@vt.edu](mailto:polivier@vt.edu)

[binoy@vt.edu](mailto:binoy@vt.edu)

[www.popcornlinux.org](http://www.popcornlinux.org)

Thanks!  
Q&A



Popcorn  
Linux