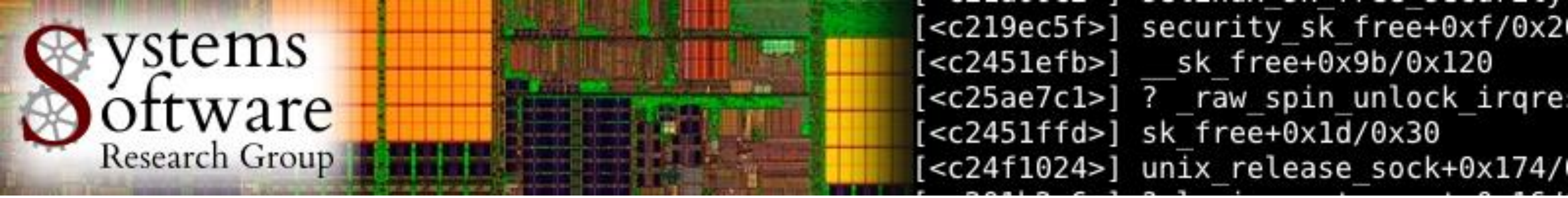


Breaking the Boundaries in Heterogeneous-ISA Datacenters

Antonio Barbalace, **Robert Lyerly**, Christopher Jelesnianski, Anthony Carno, Ho-Ren Chuang, Vincent Legout and Binoy Ravindran
Systems Software Research Group at Virginia Tech
ssrg.ece.vt.edu

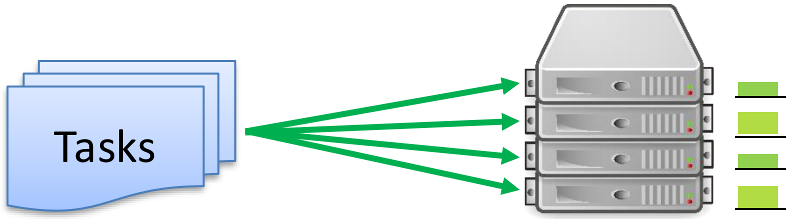
Highlight Paper
Presented on April 12th, 2017 at ASPLOS in Xi'an China

Introduction

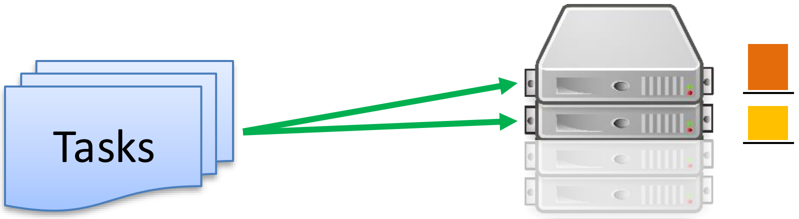
- Datacenters are integrating heterogeneous-ISA servers

Introduction

- Energy proportionality – get compute performance proportional to the amount of energy spent
- Current energy-reduction techniques migrate workloads between servers
  - Load balancing – spread workload evenly across available servers
  - Consolidation – group workload on minimal number of machines, idle or power down others



Load-balancing



Consolidation

Introduction

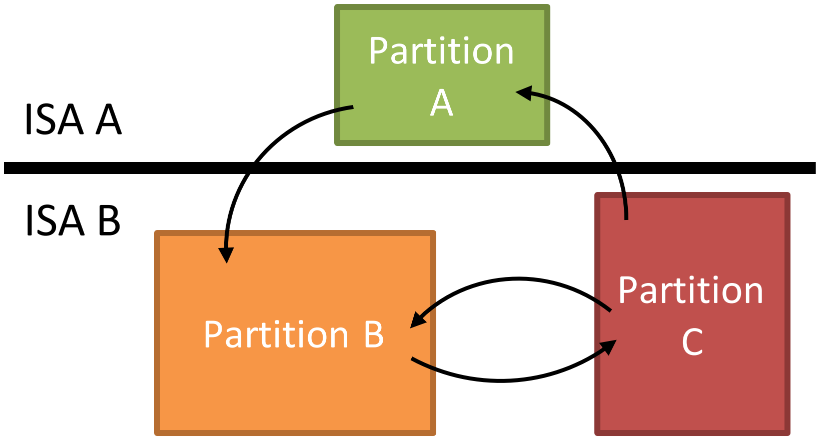
- Natively-compiled stateful applications, e.g., HPC and key-value stores, are increasingly being run in datacenters

How can existing energy management techniques be applied to these applications in heterogeneous-ISA datacenters?

Current Approaches

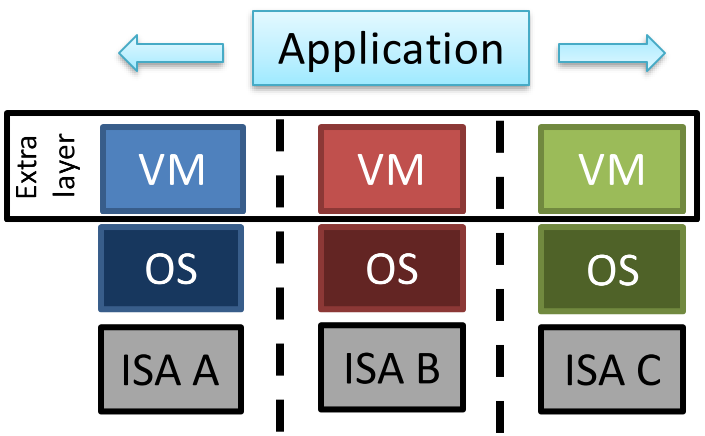
Message Passing Interface (MPI)

+ **High performance**
– **Complex code development/refactoring**
– **Hardcoded application partitions**



ISA Virtualization

- Managed languages, e.g., Java
  - **Rewrite application from scratch**
  - **Performance overheads**
- Dynamic binary translation, e.g., QEMU
  + **Run unmodified binaries**
  - **Order of magnitude slowdown**

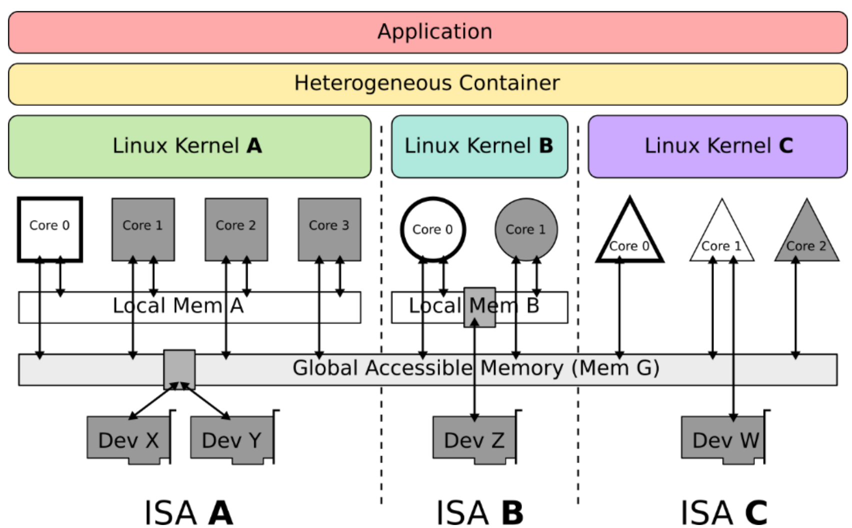


Solution

- System software stack for **migrating** compiled applications between heterogeneous-ISA servers
  - **Replicated-kernel OS** for thread and data migration
  - **Compiler** for creating a mostly-common virtual address space, generating metadata about ISA-specific execution state
  - **Runtime** for transforming ISA-specific execution state
- Allow developers to write shared memory compiled applications and leverage heterogeneity
  - Legacy code works too!

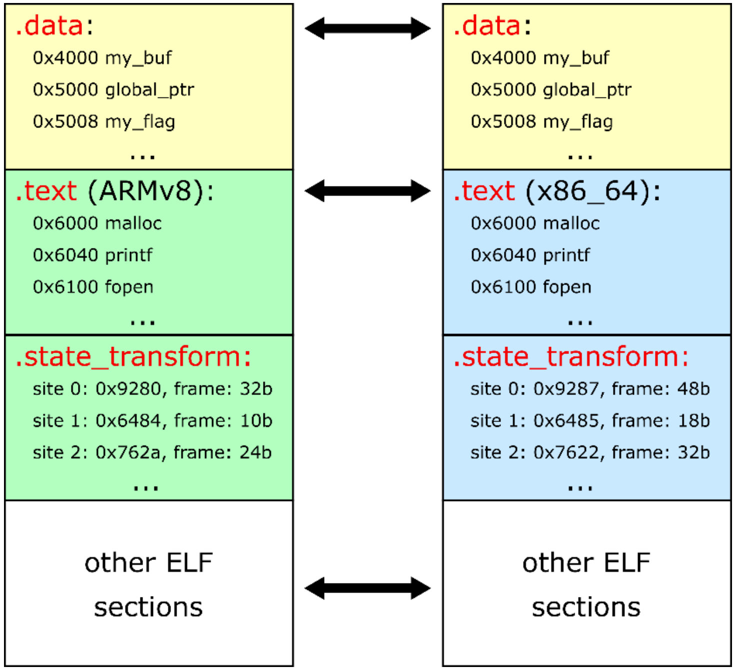
System Software Architecture

- **Heterogeneous Containers** – cross-ISA sub-environment
  - Built on top of Popcorn Linux, a replicated-kernel OS
    - Run one kernel per-ISA
    - OS services are distributed & kept coherent using message passing
  - Kernels coordinate to provide cross-ISA thread & state migration



System Software Architecture

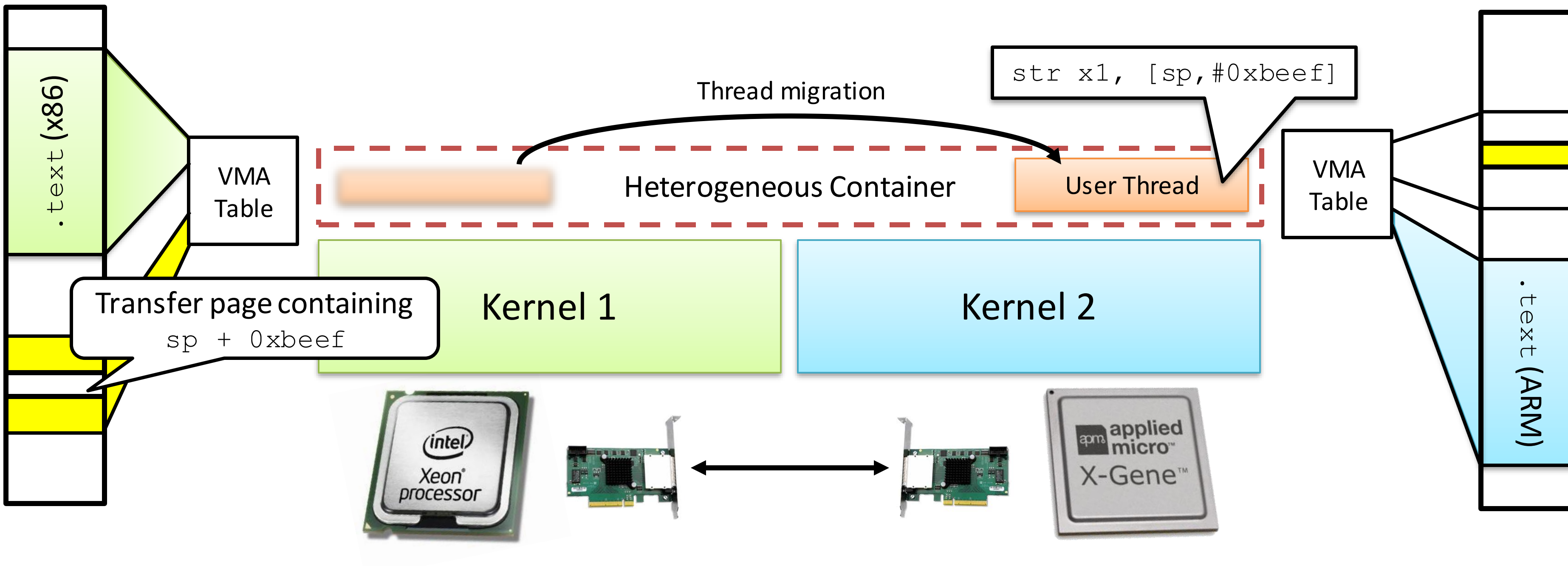
- **Multi-ISA Binaries** – migratable across ISAs
  - Application source compiled once per ISA
    - Single `.data` section, multiple `.text` sections (one per-ISA)
  - Minimize inter-ISA state transformation costs for cross-ISA migration
    - Global data (`.data`), code (`.text`) and thread-local storage aligned across all compilations
    - State transformation metadata added to binary for translating registers/stack between ISA-specific formats



Operating System

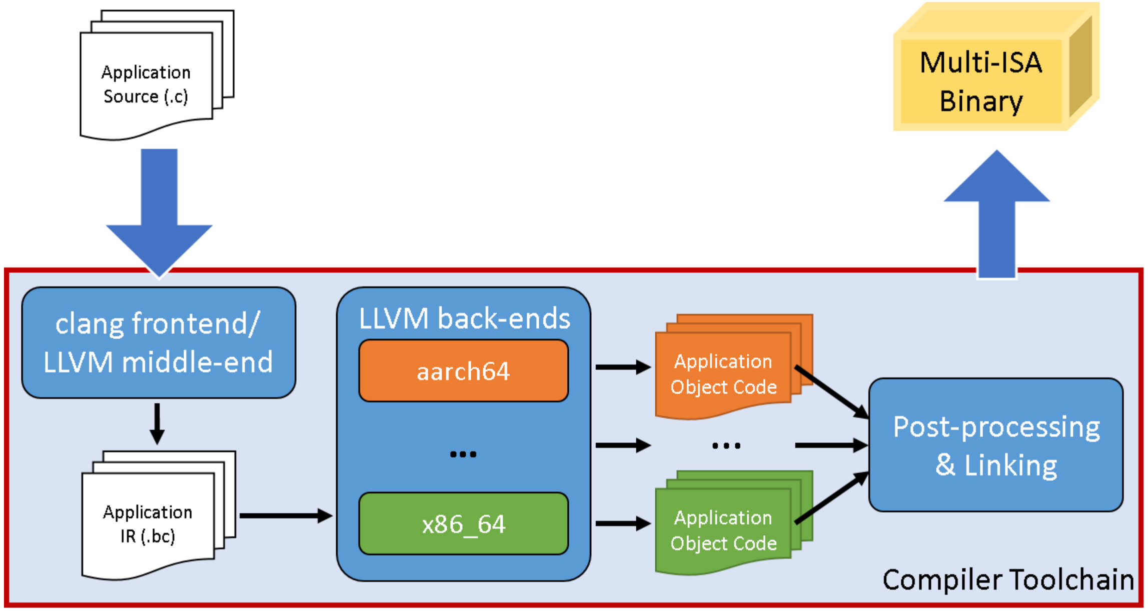
- Thread migration & heterogeneous continuations
  - Kernels cooperate to migrate user-space thread contexts between ISAs
  - Kernel maps user-space PC, SP and FBP registers between ISAs
- On-demand page migration
  - Migrate memory pages between kernels as they are accessed by the application
    - Extend the page fault handler
  - Memory region aliasing for ISA-specific sections (e.g., `.text`)

Operating System



Compiler Toolchain

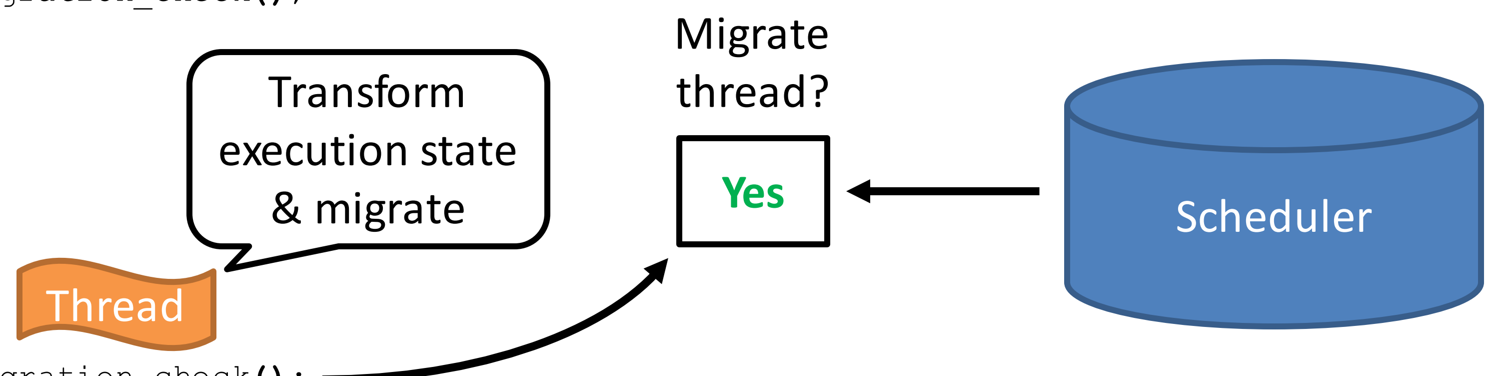
- Built on top of clang/LLVM
  - clang/LLVM 3.7.1, GNU gold 2.27 (~7k LoC)
  - Virtual address space alignment tool (~1.5k LoC)
  - State transformation runtime linked into application (~5k LoC)



Compiler Toolchain

- Insert **migration points** into code
  - Can only transform stack at *equivalence points*
    - Direct mapping of execution state between ISA-specific formats
  - Scheduler cannot migrate threads at arbitrary points, must signal threads to initiate migration process

```
void foo() {
    migration_check();
```



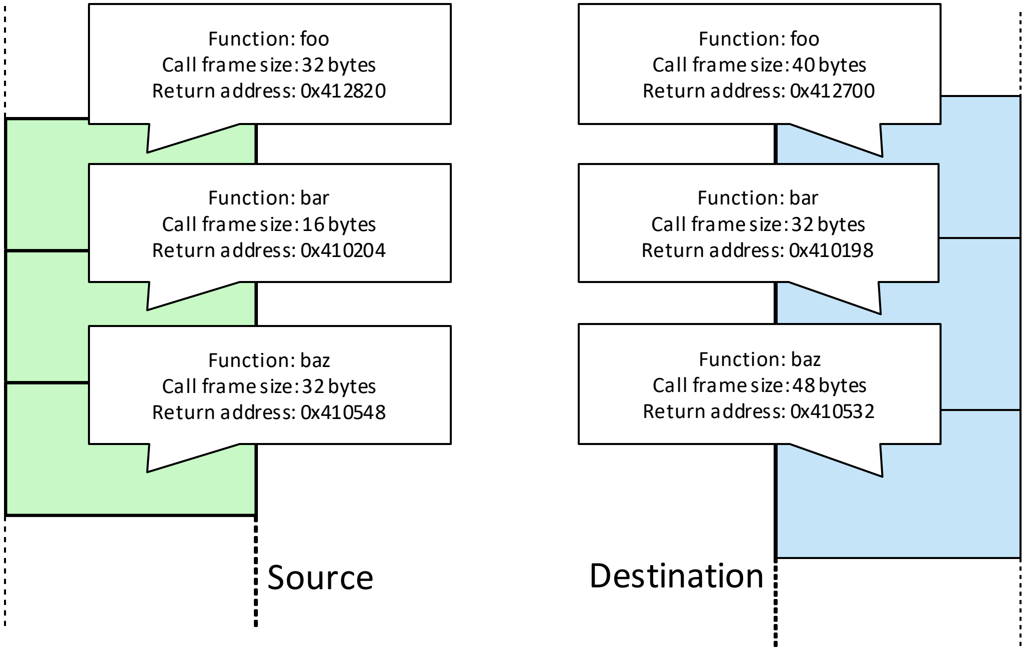

```
    migration_check();
}
```

State Transformation Runtime

- Transform registers & stack between ISA-specific formats
- Runtime transforms state before migration
  - Attaches to a thread's registers/stack
  - Reads compiler metadata describing function activation layouts
  - Rewrites stack in its entirety from source to destination ISA format
- After transformation, runtime invokes migration
  - Passes destination ISA's register state and stack to OS's thread migration service

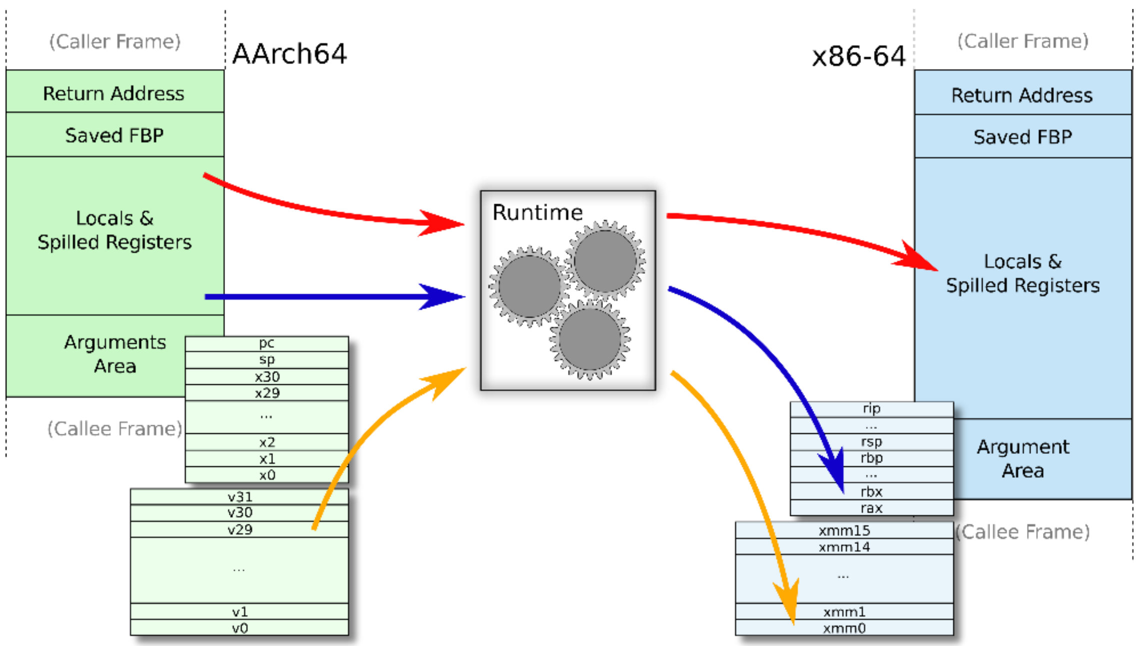
State Transformation Runtime

- Two phases to State Transformation
  1. Unwind current stack to find current live activations & size new stack
  2. Rewrite a frame at a time, from outermost frame inwards



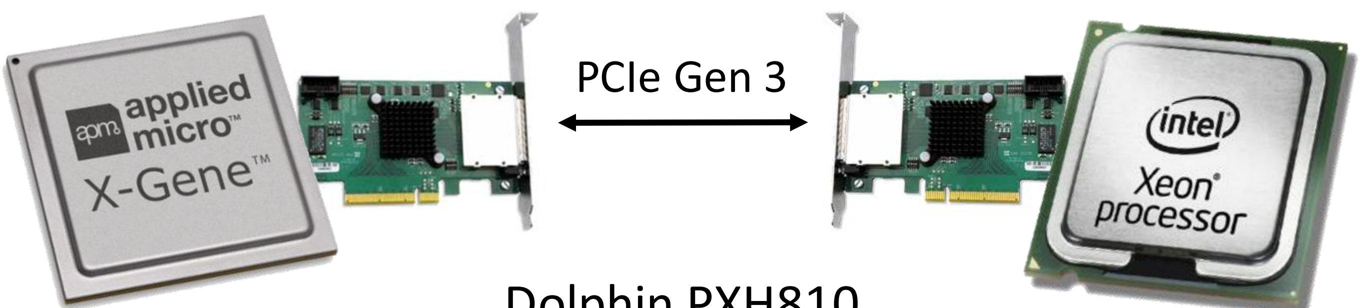
State Transformation Runtime

- Two phases to State Transformation
  1. Unwind current stack to find current live activations & size new stack
  2. Rewrite a frame at a time, from outermost frame inwards



Evaluation

- APM X-Gene 1
  - 8 cores @ 2.4GHz
  - 8MB LLC, 32GB RAM
  - 40nm process, 50W TDP
    - Measured via on-board sensor
    - Estimated power consumption scaled to 22nm using McPAT
- Intel Xeon E5-1650v2
  - 6 cores @ 3.5GHz (3.9GHz turbo)
    - Hyperthreading disabled
  - 12MB LLC, 16GB RAM
  - 22nm process, 130W TDP
    - Measured via RAPL



Dolphin PXH810
PCIe point-to-point connection, 64Gbps

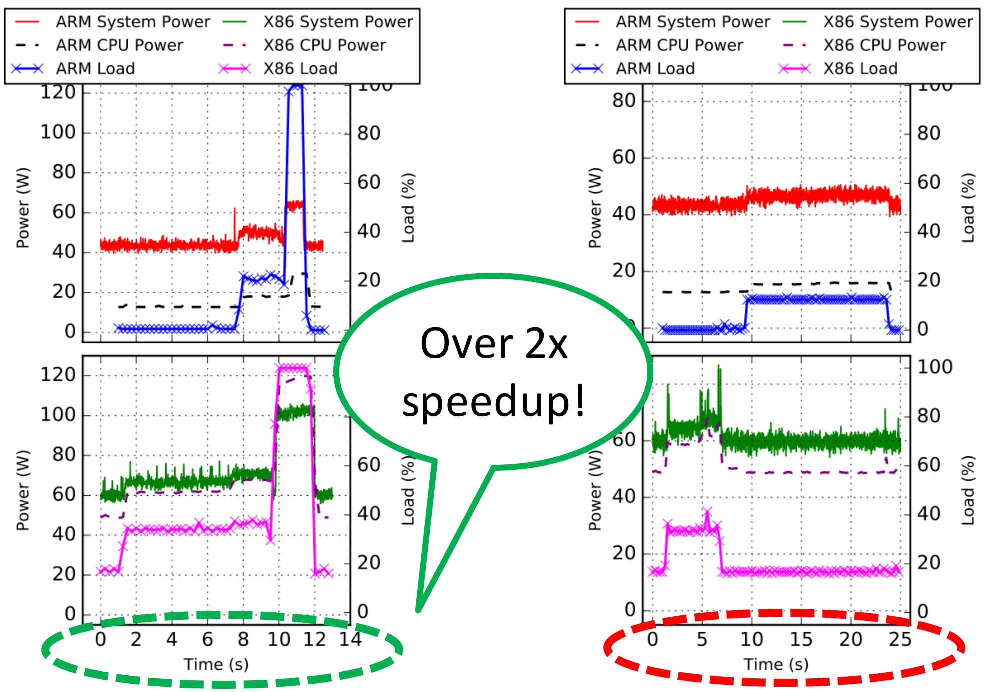
Evaluation

- Benchmarks
  - NAS Parallel Benchmarks (NPB), classes A, B & C
- Comparison: PadMig/Java
  - Source-to-source compiler inserts migration code into application
  - Migrates thread & data using Java reflection/serialization
- Scheduling
  - Periodic workload – each set consists of 5 waves of up to 14 jobs
    - Uniformly sampled from NPB (all classes)
    - Waves arrive every 60-240 seconds
  - Comparison against 2 x Intel Xeon E5-1650v2 w/o migration

Results

- Comparison: migrating NPB IS with PadMig

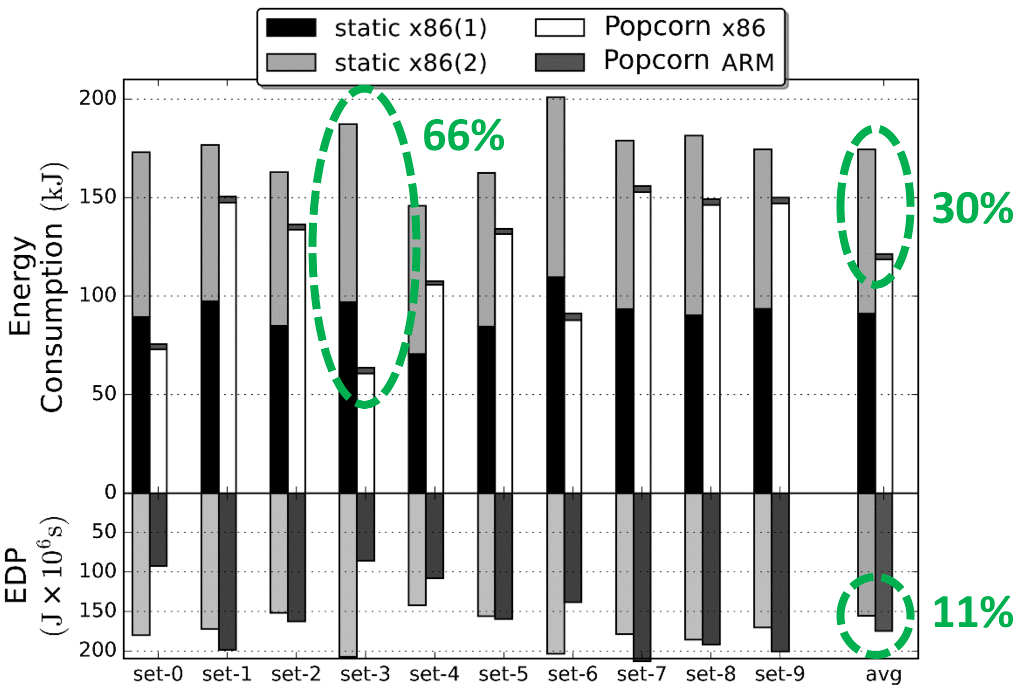
Popcorn Linux



PadMig/Java

Results

- Scheduling comparison to homogeneous setup

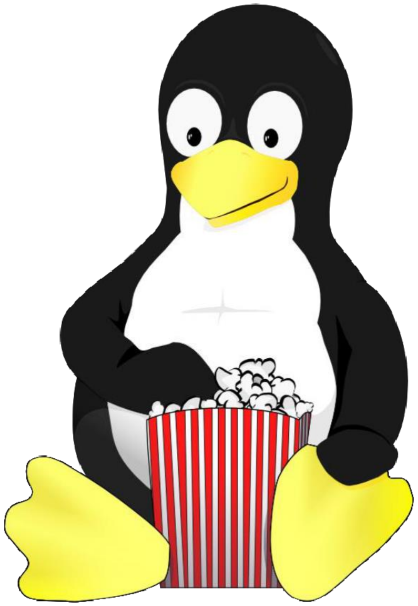


Conclusion

- Datacenters are adopting heterogeneous-ISA servers
- Proposed a full system software redesign to enable cross-ISA migration for compiled applications
  - Compiler builds multi-ISA binaries
  - OS enables cross-ISA thread and data migration
  - State transformation runtime converts ISA-specific data
  - Allows developers to use shared-memory programming model
- Implemented prototype & demonstrated effectiveness
  - Saved on average 30% and up to 66% energy for bursty workloads

More Information

- Popcorn Linux is open source and available online at http://popcornlinux.org



Acknowledgements

- The authors would like to thank Christopher Rossbach and Malte Schwarzkopf for their invaluable comments on an early version of the paper, and the anonymous reviewers for their insightful feedback

- Logos downloaded from:
  - Intel: https://commons.wikimedia.org/wiki/File:Intel-logo.svg
  - ARM: https://commons.wikimedia.org/wiki/File:ARM_logo.svg
  - Cavium: http://www.prnewswire.com/news-releases/cavium-announces-thunderx2-300276536.html
  - Applied Micro: https://www.forbes.com/sites/patrickmoorhead/2015/11/27/whats-the-significance-of-applied-micro-circuits-x-gene-3-and-x-tend-interconnect/#7f6797cb4384
  - Qualcomm: https://commons.wikimedia.org/wiki/File:Qualcomm-Logo.svg
  - OpenPOWER: https://commons.wikimedia.org/wiki/File:Openpower-logo-wht-bg_logo-wht-bg.jpg
  - IBM: https://commons.wikimedia.org/wiki/File:IBM_logo.svg
  - Google: https://commons.wikimedia.org/wiki/File:Google_2015_logo.svg
  - Rackspace: https://www.sec.gov/Archives/edgar/data/1107694/000110769416000063/rackspacelogoclra07.jpg

- This work is supported in part by ONR under grants N00014-13-1-0317 and N00014-16-1-2711, AFOSR under grant FA9550-14-1-0163, and NAVSEA/NEEC under grants 3003279297 and N00174-16-C-0018. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of ONR, AFOSR, and NAVSEA.

Questions?