Effects of Virtualization on a Scientific Application

Running a Hyperspectral Radiative Transfer Code on Virtual Machines

Anand Tikotekar, Geoffroy Vallée, Thomas Naughton,
Hong Ong, Christian Engelmann & Stephen L. Scott

Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA

Anthony M. Filippi

Department of Geography
Texas A&M University
College Station, TX, USA

Premise:

Investigate the use of virtual machines for a real-world scientific application.

Goals:

1. Provide some insight for scientists interested in employing virtualization in their research.
2. Increase our understanding of application performance on VMs, and the associated tools currently available.

Background

- Prior work looking at *Hydrolight*
  - Summer project to aid running on cluster
  - Reduce wall-clock time with low investment

- HydroHPCC tools
  - Tools developed to support Hydrolight use on cluster
  - Decrease overhead in simulation input preparation
  - Add tools to help automation/batch-parallel execution
    - Leverage C3 with SSH to run simulations

Application Overview

- Hydrolight (Sequoia Scientific, Inc.)
  - Radiative-transfer numerical model
  - Determines radiance distribution within/leaving a water body
    - Ex. parameters: water depth, wavelength, wind speed, etc.
- Variety of uses
  - Underwater visibility studies
  - Remote-sensing mission planning & algorithm evaluation
  - Enhancing understanding of physical processes

Hydrolight Simulation Properties

- Simulations
  - Each is single execution for given set of model parameters
  - Binary name: `maincode.exe`
  - Parameters: 2 input files
    - "Iroot.txt" & "root.for“
  - HydroHPCC manages job startup, compile/re-link, execution & output

- Previous work performed 2,600 simulations on a small cluster
  - Generate training data for ANN (artificial neural network)
  - Wall-clock time: ~3.5 hrs (natively without profiling)
  - Time breakdown: ~50% with time > 9min

- Simplification for Experimentation
  - Simulation times consistent across executions
  - Select single experiment (input parms) from 10min group

Outline

- Discuss methodology & experimentation
- Observations & future work
- Summary

Methodology

- Simulations / Profiling
  – Select single experiment from 2,600 set
    - ~10min wall-clock native
  – OProfile for both native & Xen platforms
- Timing Note
  – Profiling focused on Hydrolight
  – Wall-clock timings for HydoHPCC
    - startup, linking/execution, data processing

OAK RIDGE National Laboratory

Experimental Environment

- XTORC Cluster
  - 2Ghz Pentium IV [<64 nodes]
  - 768MB memory
  - 100Mb FastEthernet
  - Fedora Core 5 (FC5)
  - Linux 2.6.16.33 (both native & para-virtualized)
  - Xen 3.0.4
  - OProfile 0.9.1
- Simulations*
  - GNU Fortran G77 3.2.3 (FFLAGS=-O3)
  - Bottom type: Red Algae
  - Depth: 10.0 m
  - Chl. concentration: 10.0 mg m^-3

* Note, refer to paper citation #4 for complete parameter details.

Profiler Settings

- Profiler
  - OProfile 0.9.1
  - Xenoprof
    - OProfile user-level patch
    - Xen 3.0.4 includes other aspects

- Run Parameters:

```
opcontrol --start --separate=kernel \
  --event=GLOBAL_POWER_EVENTS:100000:1:1:1 \
  --event=ITLB_REFERENCE:100000:2:1:1 \
  --event=INSTR_RETIRED:100000:1:1:1 \
  --event=MACHINE_CLEAR:100000:1:1:1 \
  --vmlinux=/opt/vmlinux_location/vmlinux
```

Profiler Settings (2)

Example:

"--start" start data collection

"--separate=kernel " separate shared library profiles per-application plus kernel profiles

"--event=GLOBAL_POWER_EVENTS:100000:1:1:1 "

event: GLOBAL_POWER_EVENTS

reset counter: 100000

h/w unitmask: 1

profile kernel: 1 (true)

profile userspace: 1 (true)

"--vmlinux=/opt/vmlinux_location/vmlinux " un-stripped kernel image

** Note, we focused on GLOBAL_POWER_EVENTS & ITBL_REFERENCE in order to focus on the actual time spent by the application and its relationship with the ITLB miss rate.*

OAK RIDGE National Laboratory

OProfile Events

- GLOBAL_POWER_EVENTS: time during which processor is not stopped
- ITLB_REFERENCE: translations using the instruction translation lookaside buffer; 0x02 ITLB miss
- INSTR_RETIRE: retired instructions; 0x01 count non-bogus instructions which are not tagged
- MACHINE_CLEAR: cycles with entire machine pipeline cleared; 0x01 count a portion of cycles the machine is cleared for any cause

Experiments

- Ran application on 3 platforms
  - Native
  - HostOS (dom0)
  - VM (domU)
- Focus on user (T_{usr}) & system (T_{sys})
  - Samples pertaining to app image=`maincode.exe`
  - Compare Native to Virtual
  - NOTE: VM values for T_{sys} are incomplete
    - Runs on HostOS (dom0) are complete

OAK RIDGE National Laboratory

OProfile sampling

- Register NMI
- Generate interrupt & record context
- Dereference symbols from context
- Example:

```
samples    image name      app name        symbol name
9877360    maincode.exe    maincode.exe    rhotau_
    ....
140760     libm-2.4.so     maincode.exe    cos
    ....
10129      vmlinux         maincode.exe    init_pmtmr
```

Gathering Data

- Add OProfile calls to HydroHPCC
- For each platform (native, hostOS, VM)
    1. Run single simulation on multiple nodes
    2. Gather results/output
    3. Run post-processing scripts
    4. Record stats
- Post-processing scripts
    - Extract data specific to "`maincode.exe`"

Post-processing heuristics

image name	app name	Time Class
maincode.exe	maincode.exe	T_{usr} ($T_{usr.app}$)
lib*	maincode.exe	T_{usr} ($T_{usr.lib}$)
ld-*	maincode.exe	T_{usr} ($T_{usr.lib}$)
oprofiled	*	T_{sys} ($T_{sys.prof}$)
oprofile.ko	*	T_{sys} ($T_{sys.prof}$)
*.ko	maincode.exe	T_{sys} ($T_{sys.os}$)
vmlinux	maincode.exe	T_{sys} ($T_{sys.os}$)
xen-syms	maincode.exe	T_{sys} ($T_{sys.vmm}$)
anon(XXXX...)	maincode.exe	*untrackable*

Table 1. Post-processing heuristics applied to OProfile data shown using '$\star$' as a wildcard matching anything.

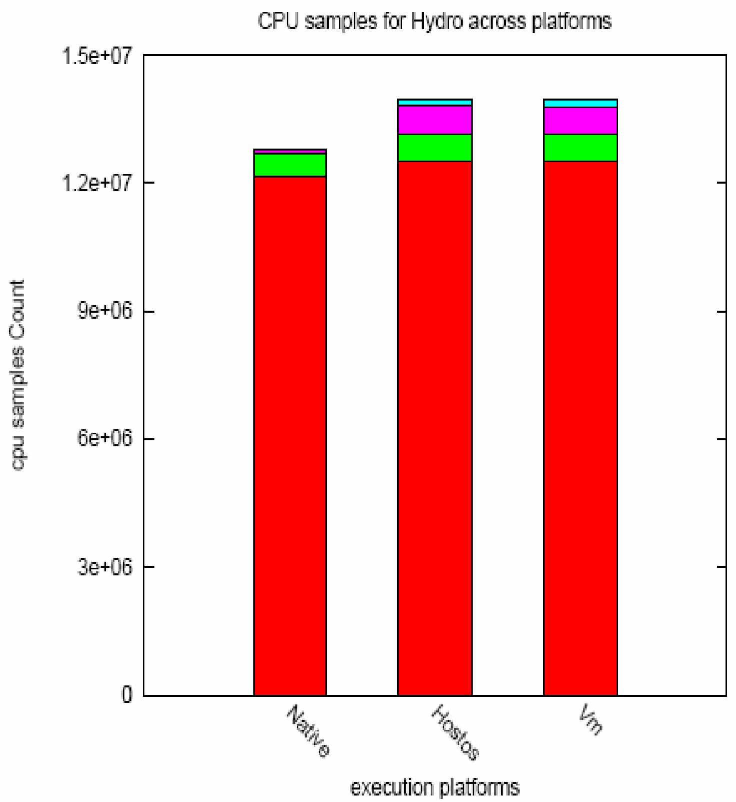
user

system

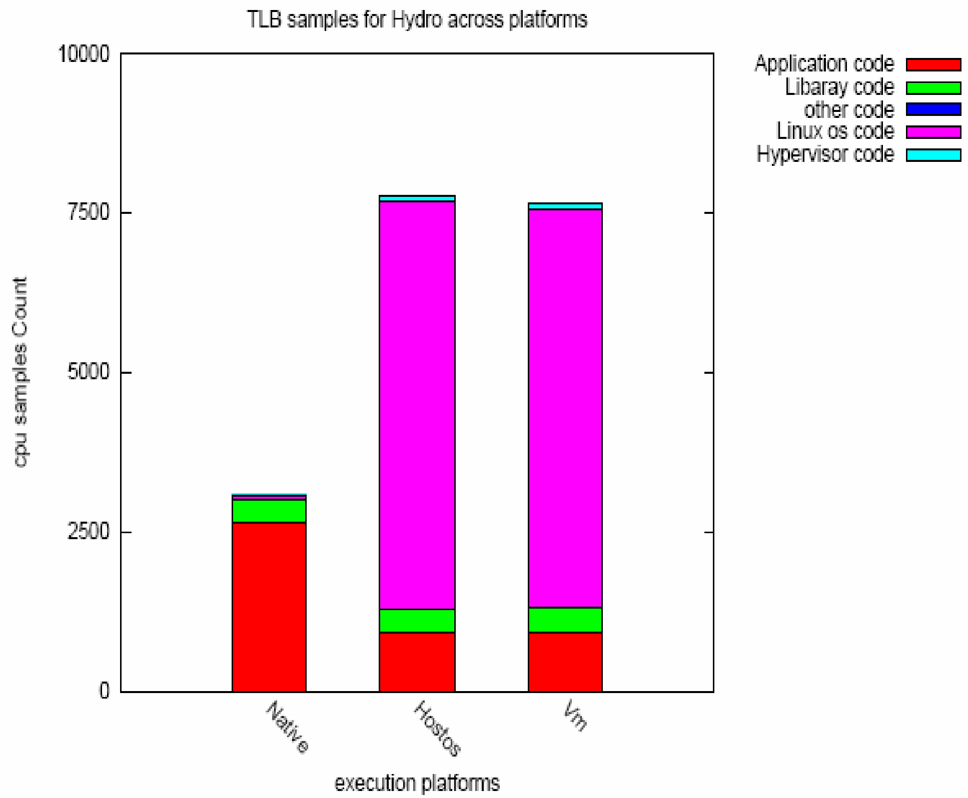
```
samples     image name      app name        symbol name
9877360     maincode.exe    maincode.exe    rhotau_
    ....
140760      libm-2.4.so     maincode.exe    cos
    ....
10129       vmlinux         maincode.exe    init_pmtmr
```

Platform avgerages 20 runs

CPU (GLOBAL_POWER_EVENTS)

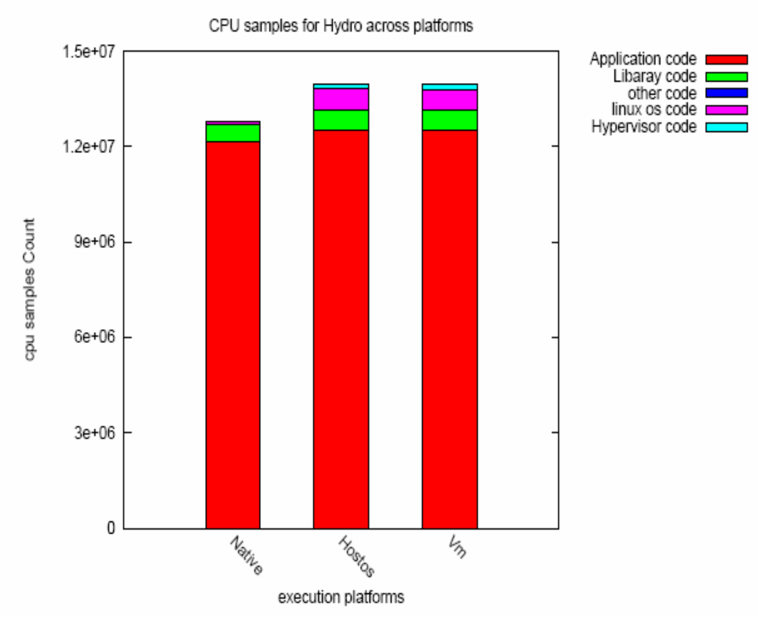


ITLB miss (ITLB_REFERENCES)



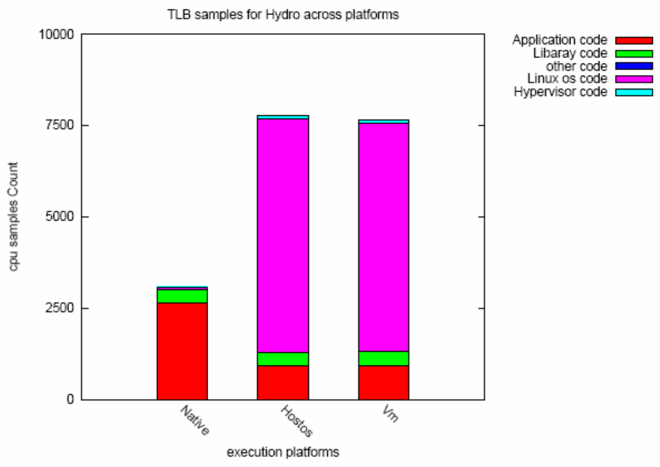
CPU time

- CPU time
  - Majority of time in user code (Native & VM)
  - T_{usr} roughly equiv. for Native & Virtual
  - VM has ~7K more system code samples than Native



ITLB miss

- ITLB miss
  - Virtual spends approx. 2x more in user code
  - N:V user vs system:
    - Native ~43x usr/sys   (3,007 / 69)
    - Virtual ~0.20 usr/sys   (1299 / 6,340)
  - Virtual user/system code, system ~5K more samples
  - VM has ~6.3K more system code samples than Native



Observations: Native vs Virtual

- All: CPU - approx. same time in app code
  – Confirms virtualization not hurt (cpu) user code
- Native: lower number of samples
  – Both user & system code
    - Except: Higher ITLB miss for user code on native
    - Note, Higher ITLB miss for system on virtual.

Observations: Native vs Virtual (2)

- Native: App wall-clock times more consistent
  - min/max: 690/697sec vs VM 763/790sec (Table 4)
- Wall-clock on virtual environment
  - w/o profile instrumentation ~8% > native
  - w/ profile instrumentation ~11% > native
    - Note: Profile only 1 event, drops to ~8% > native
    - Note: VM missing some system samples!
- Overall time to solution for 2,600 simulations
  - Virtual is roughly 8% higher than native
  - 36 nodes: Native: 2h 40m ; VM: 2h 55m

Observations: Native vs Virtual (3)

- Native: higher std. on system code
  - Both CPU & ITLB misses
  - Comment: Possibly an accounting / node issue?
    - 2-3 nodes report "`ide_outsw`" associated w/ differerent app image, so excluded by our method.
      - App name: "`vmlinux`" instead of "`maincode.exe`"

Platform	cpu avg	cpu std	% std	tlb avg	tlb std	% std
Native	92984	16908	18.18	69	10	14.49
HostOS	823407	21270	2.58	6475	123	1.89
VM	792183	23082	2.91	6340	134	2.11

Table 3. Average and standard deviations for system level (T_{sys}) CPU and ITLB miss samples from one experiment (file) over 20 runs. Note, the VM's T_{sys} only contains domU portion.

OProfile Observations

- OProfile differences
  - Sampling for multiple events simultaneously
    - Native not noticeable effect
    - Virtual greatly increased the overhead (interference)
    - See future work
  - Lack full “context” in virtual
    - domU/dom0 – “`maincode.exe`” in domU context only

OAK RIDGE National Laboratory

Related Work

- HPC benchmarks & network apps/IO
  - Original Xenoprof developers [Menon:vee05]
  - Para-virt for HPC systems [Wolski:xhpc06]
  - VMM I/O bypass [Panda:ics06]
  - Xen & UML for HPC [Stanzione:cluster06]
- Some looked at real-world apps
  - Mainly systems perspective / developers
- Profiler tools
  - VIVA (UCSB) project's *VIProf* for JVM
  - Address issue of dynamic symbols (profiling context)

Future work

- Look into OProfile/Xenoprof
  - Single vs. Multi event samples
  - Guest context
- Investigate system side
  - Identify root causes
- Revise methodology
  - Improve VM system portions

Platform	Avg User	Avg System
Native	12009823	81559
HostOS	12781305	176387
VM	12697736	163400

Table 7. Average number of samples for profiling the single event GLOBAL_POWER_EVENTS, for one experiment (file) over 10 runs.

Platform	Avg User	Avg System
Native	22	7
HostOS	38	45
VM	38	35

Table 8. Average number of samples for profiling the single event ITLB_REFERENCE (miss_samples), for one experiment (file) over 10 runs.

Summary

- Analyzed scientific application & virtual env.
  - Hypspectral radiative transfer code (Hydrolight)
  - Wall-clock on virtual environment (4 events)
    - w/o profile instrumentation ~8% > native
    - w/ profile instrumentation ~11% > native
      - Profile only 1 event, drops to ~8% > native
- Tools for virtual environments
  - Still somewhat immature
  - Performance isolation issues
    - ex. OProfile sampling 4 vs. 1 event

Thank you

Questions?

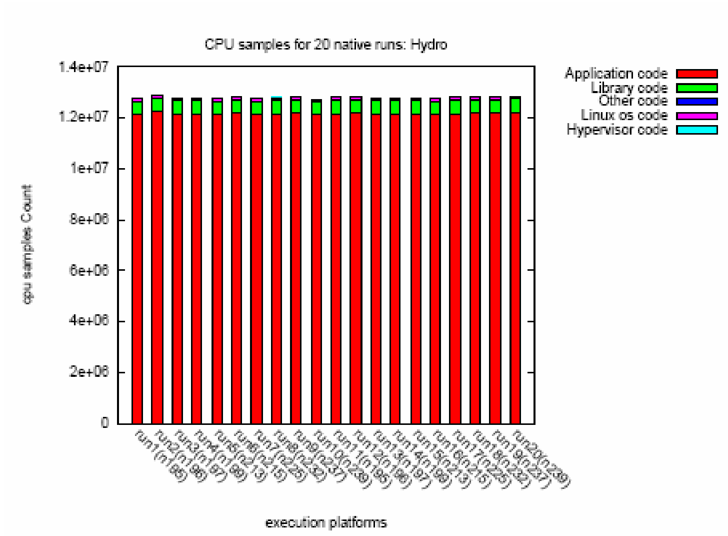
Acknowledgements

This research was supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

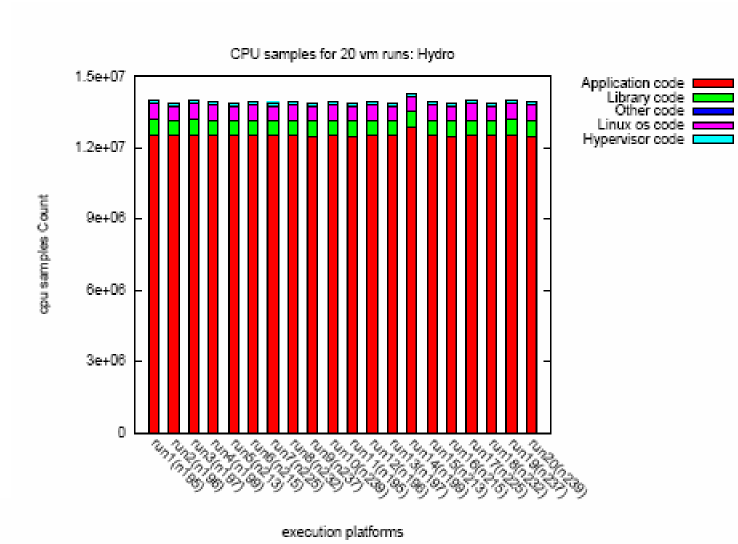
A. M. Filippi: This research was supported in part by an appointment to the U.S. Department of Energy (DOE) Higher Education Research Experiences (HERE) for Faculty at the Oak Ridge National Laboratory (ORNL) administered by the Oak Ridge Institute for Science and Education. A.M. Filippi also thanks Budhendra L. Bhaduri and Eddie A. Bright, Computational Sciences & Engineering Division, ORNL, for their support.

Backup slides

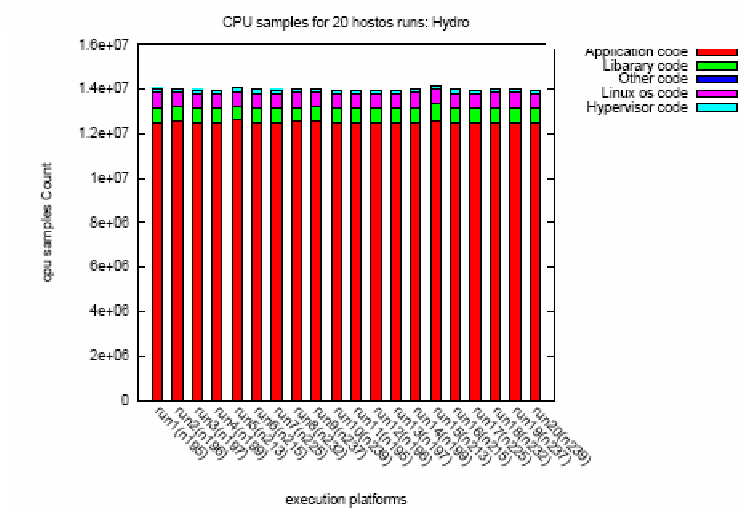
CPU: Native / HostOS / VM



Native



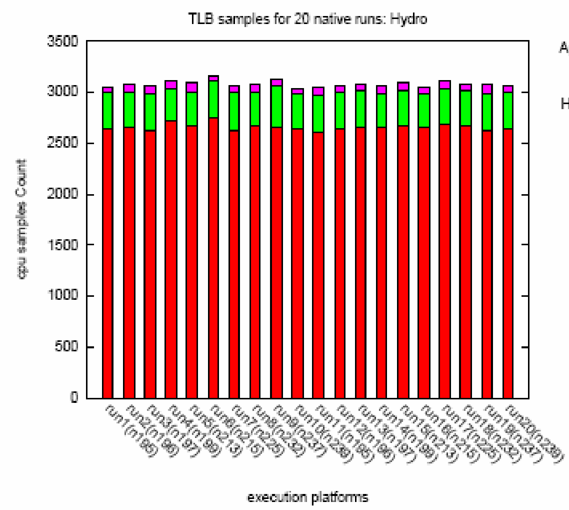
VM



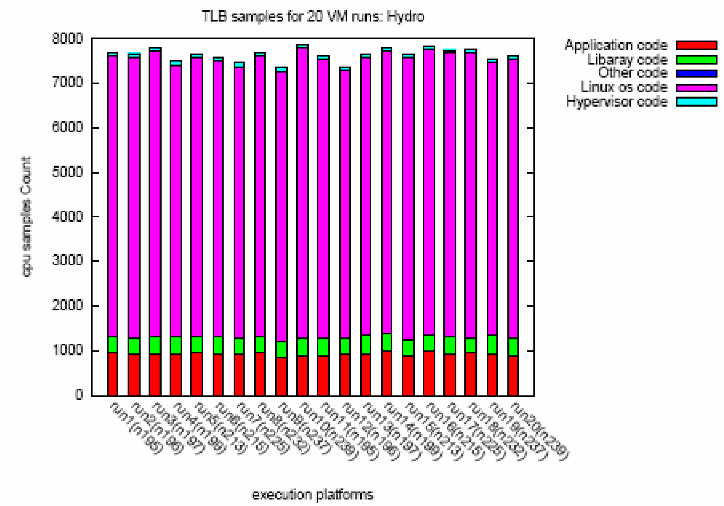
HostOS

OAK RIDGE National Laboratory

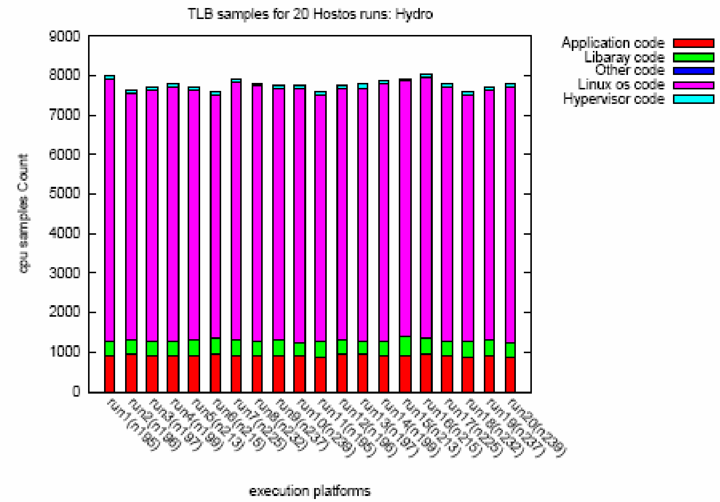
ITLB miss: Native / HostOS / VM



Native



VM



HostOS

Average & STD 20 runs, 4 events

Platform	cpu avg	cpu std	% std	tlb avg	tlb std	% std
Native	12687854	31740	0.25	3007	32	1.06
HostOS	13162659	50532	0.38	1296	37	2.85
VM	13156140	84812	0.64	1299	39	3.00

Table 2. Average and standard deviations for user level (T_{usr}) CPU and ITLB miss samples from one experiment (file) over 20 runs.

Platform	cpu avg	cpu std	% std	tlb avg	tlb std	% std
Native	92984	16908	18.18	69	10	14.49
HostOS	823407	21270	2.58	6475	123	1.89
VM	792183	23082	2.91	6340	134	2.11

Table 3. Average and standard deviations for system level (T_{sys}) CPU and ITLB miss samples from one experiment (file) over 20 runs. Note, the VM's T_{sys} only contains domU portion.

CPU time

Platform relation	Average
N:V T_{usr} / T_{usr}	0.96
N:N T_{usr} / T_{sys}	136
V:V T_{usr} / T_{sys}	16.6
N:V T_{sys} / T_{sys}	0.11

ITLB miss

Platform relation	Average
N:V T_{usr}/T_{usr}	2.31
N:N T_{usr}/T_{sys}	43.27
V:V T_{usr}/T_{sys}	0.20
N:V T_{sys}/T_{sys}	0.010

- Average of one experiment, 20 runs
- N=native, V=VM
- (Tsys on VM only domU)

Application Overview (2)

- Hydrolight execution characteristics
  - Sequential, deterministic, CPU-bound
  - I/O: initial input params, configurable output
    - Output file settings: KBs up to MBs
  - Majority of time in single user-space routine*
    - e.g., rhotau()

* Note: Based on the set of parameters/tests we performed.

OAK RIDGE National Laboratory