An Operating System State Pausing Approach to Virtual Storage Device and Physical Computer System Co-Simulation

國立交通大學
National Chiao Tung University

Ming-Ju Wu (吳明儒)
Advisor: Chun-Jen Tsai
2015/01/08

About Me

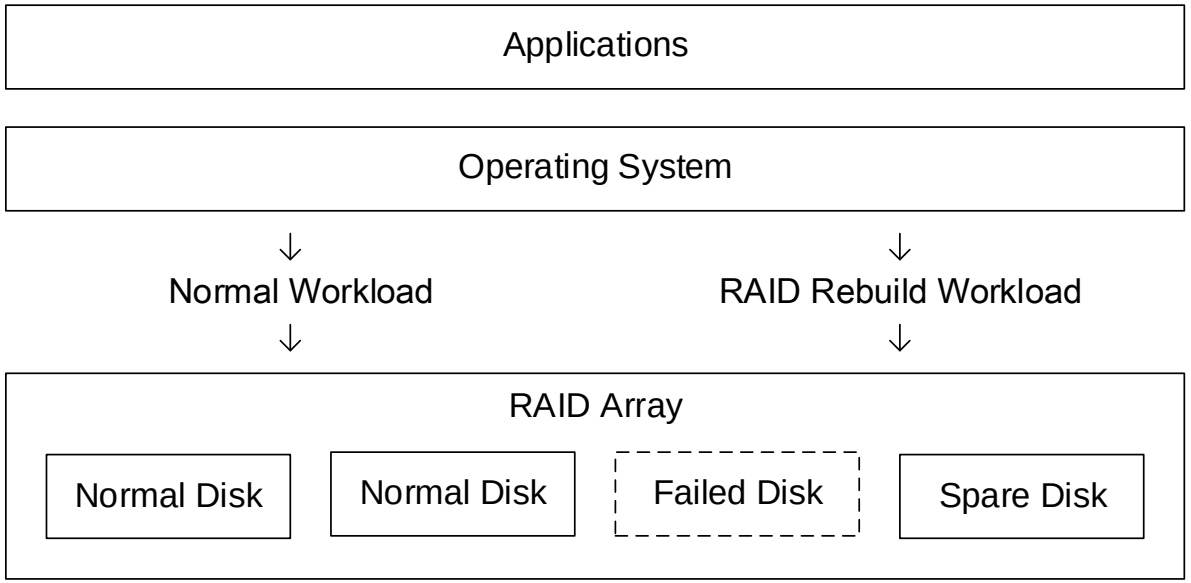
- Publications
  - **Ming-Ju Wu** and Chun-Jen Tsai, "A Storage Device Emulator for System Performance Evaluation," ACM Transactions on Embedded Computing Systems, U.S.A., Volume 14 Issue 4, Article 82, October 2015, 27 pages.
  - **Ming-Ju Wu**; Yi-Tseng Chen; Chun-Jen Tsai, "Hardware-assisted syntax decoding model for software AVC/H.264 decoders," IEEE International Symposium on Circuits and Systems (ISCAS), 2009.
  - **Ming-Ju Wu**; Yan-Ting Chen; Chun-Jen Tsai, "Dynamic pipeline-partitioned video decoding on symmetric stream multiprocessors," IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2015.
- Education
  - 國立交通大學 資訊科學與工程研究所 2006 – 2009
    - 碩士論文：SoC architecture for MPEG Reconfigurable Video Coding Framework
    - 指導教授：蔡淳仁
  - 國立交通大學 資訊工程學系 2002 – 2006
- Work Experience
  - 聯發科技 運算平台技術開發處 2014/12 – Present
  - 工業技術研究院 晶片中心、雲端中心 2009 – 2013

Outline

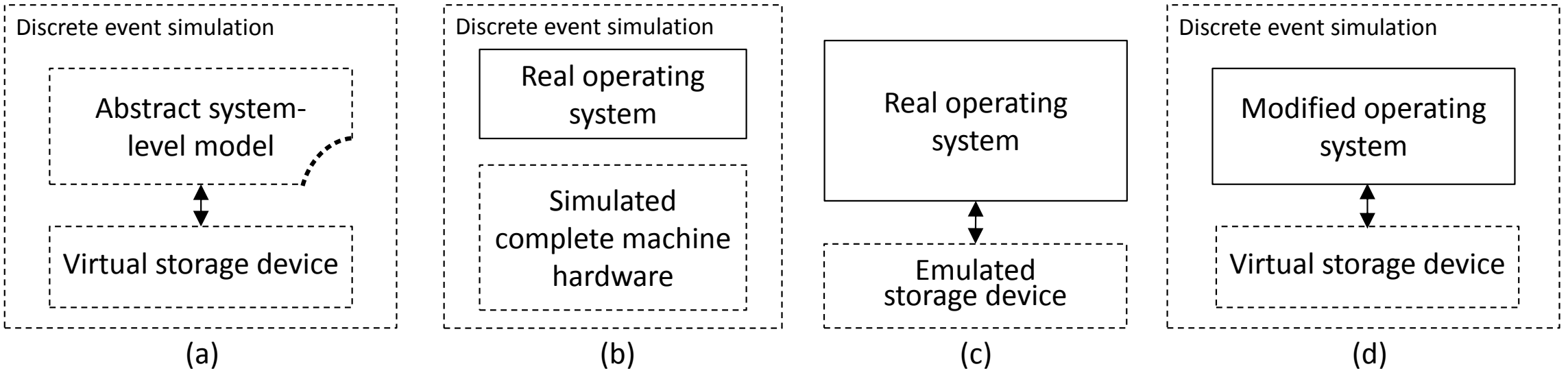
- Motivation
- Operating System State Pausing
  - Enables Virtual Storage Device and Physical Computer System Co-Simulation
- Two Works based on the OS State Pausing Concept
  - A Full-System Co-Simulator
  - A Storage Device Emulator
- Conclusions & Future Work

Motivation

- I have some ideas for improving online RAID array rebuild speeds
- Needed an environment for carrying out experimentations
- Ideally, the environment should simulate all the behaviors of the OS and application programs



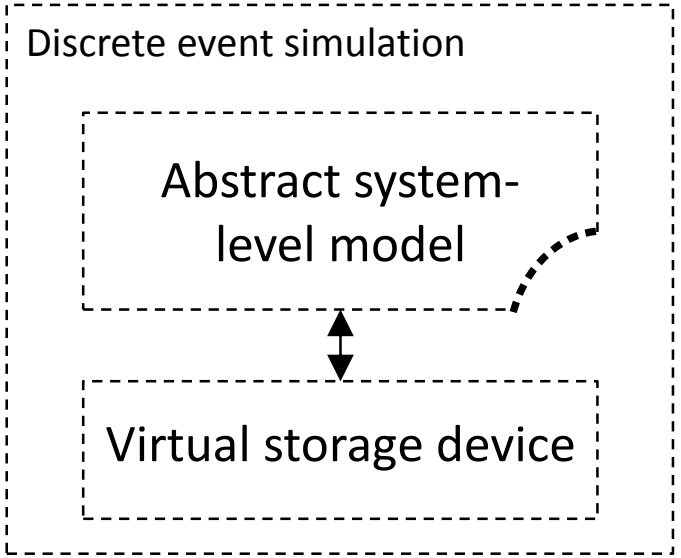
Full-System Performance Evaluation Approaches



a) Simulation with abstract system-level models
b) Complete machine simulation
c) Storage emulation
d) Proposed approach: Virtual storage device and physical computer system co-simulation

Simulation with Abstract Models

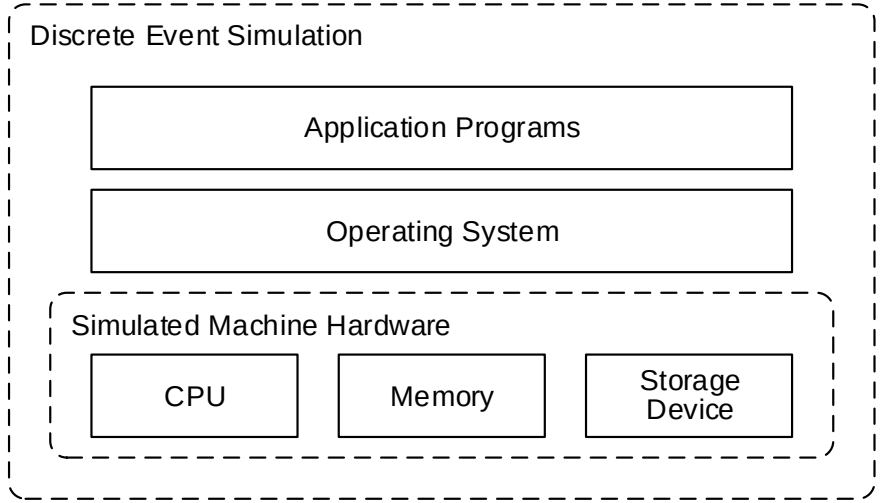
- Pro: Can provide fast preliminary answers
- Makes simplifying assumptions
- Can lead to wrong design conclusions [Thekkath et al. 1994; Ganger and Patt 1998]
  - Experts in the networking field also have similar observations [Wang 2007]
- Cannot run real-world applications



- Chandramohan A. Thekkath, John Wilkes, and Edward D. Lazowska. 1994. **Techniques for file system simulation.** Softw. Pract. Exper. 24, 11 (November 1994)
- Gregory R. Ganger and Yale N. Patt. 1998. **Using System-Level Models to Evaluate I/O Subsystem Designs**. IEEE Trans. Comput. 47, 6 (June 1998).
- S.Y. Wang, C.L. Chou, C.C. Lin, "**The Design and Implementation of the NCTUns Network Simulation Engine**", Simulation Modelling Practice and Theory, 15 (2007)

Complete Machine Simulation

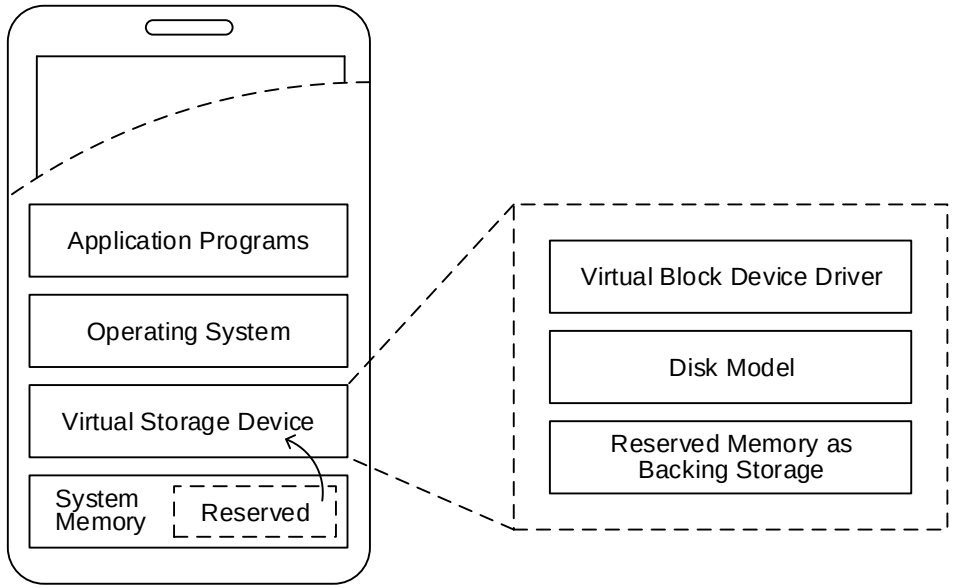
- Constructs a simulation model of the target machine hardware in enough detail
- The real OS and application programs are executed on the simulation model
- Pro: Anything that can be modeled can be simulated
- Con:
  - Simulation speed [Rosenblum et al. 1995]
  - Modeling effort [Gibson et al. 2000; Gutierrez et al. 2014]
  - Lack of detailed hardware specification for proprietary SoCs [Eklov et al. 2011]



- Mendel Rosenblum, Stephen A. Herrod, Emmett Witchel, and Anoop Gupta. 1995**. Complete Computer System Simulation: The SimOS Approach**. IEEE Parallel Distrib. Technol. 3, 4 (December 1995)
- Jeff Gibson, Robert Kunz, David Ofelt, Mark Horowitz, John Hennessy, and Mark Heinrich. 2000**. FLASH vs. (Simulated) FLASH: closing the simulation loop**. SIGARCH Comput. Archit. News 28, 5 (November 2000), 49-58.
- Anthony Gutierrez, Joseph Pusdesris, Ronald G. Dreslinski, Trevor Mudge, Chander Sudanthi, Christopher D. Emmons, Mitchell Hayenga, and Nigel Paver. 2014. **Sources of error in full-system simulation**. Performance Analysis of Systems and Software (ISPASS), 2014
- Butko, A.; Garibotti, R.; Ost, L.; Sassatelli, G., "**Accuracy evaluation of GEM5 simulator system,**" Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012
- David Eklov, Nikos Nikoleris, David Black-Schaffer, and Erik Hagersten. 2011. **Cache Pirating: Measuring the Curse of the Shared Cache**. In Proceedings of the 2011 International Conference on Parallel Processing (ICPP'11)

Local Virtual Storage Device Emulation

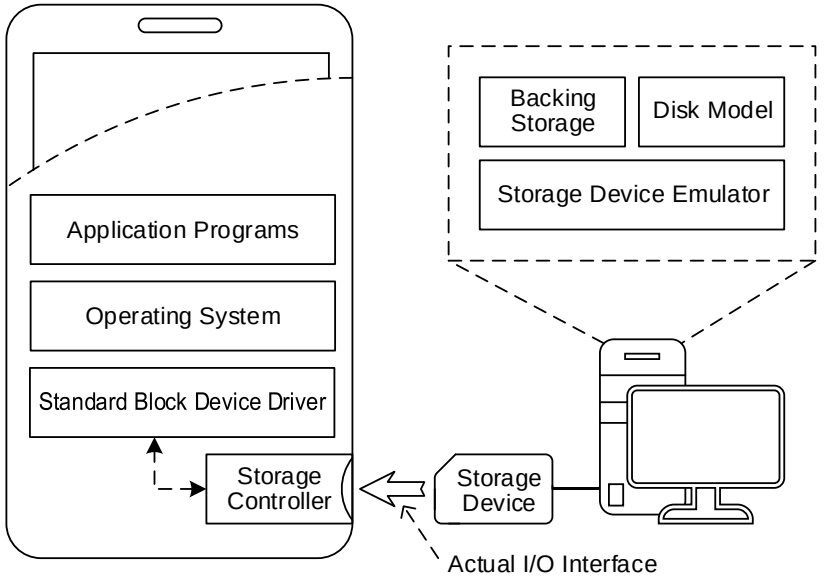
- Simulates a virtual block device to the OS at the block device driver level
- Uses RAM as the backing storage
- I/O service timings provided by the disk model
- Problems:
  - Storage size limited by RAM size
  - Uses target system's CPU for disk model simulation
  - Everything needs to run in real-time
    - Disk model simulation speed and backing storage speed must be faster than the target storage device



- John Linwood Griffin, Jiri Schindler, Steven W. Schlosser, John S. Bucy, and Gregory R. Ganger. 2002**. Timing-accurate storage emulation**. In Proceedings of the 1st USENIX conference on File and storage technologies (FAST'02).
- Kaoutar El Maghraoui, Gokul Kandiraju, Joefon Jann, and Pratap Pattnaik. 2010. **Modeling and simulating flash based solid-state disks for operating systems**. In Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering (WOSP/SIPEW '10).
- Ying-Chieh Lee, Chin-Ting Kuo, and Li-Pin Chang. 2012. **Design and Implementation of a Virtual Platform of Solid-State Disks**. IEEE Embed. Syst. Lett. 4, 4 (December 2012), 90-93.

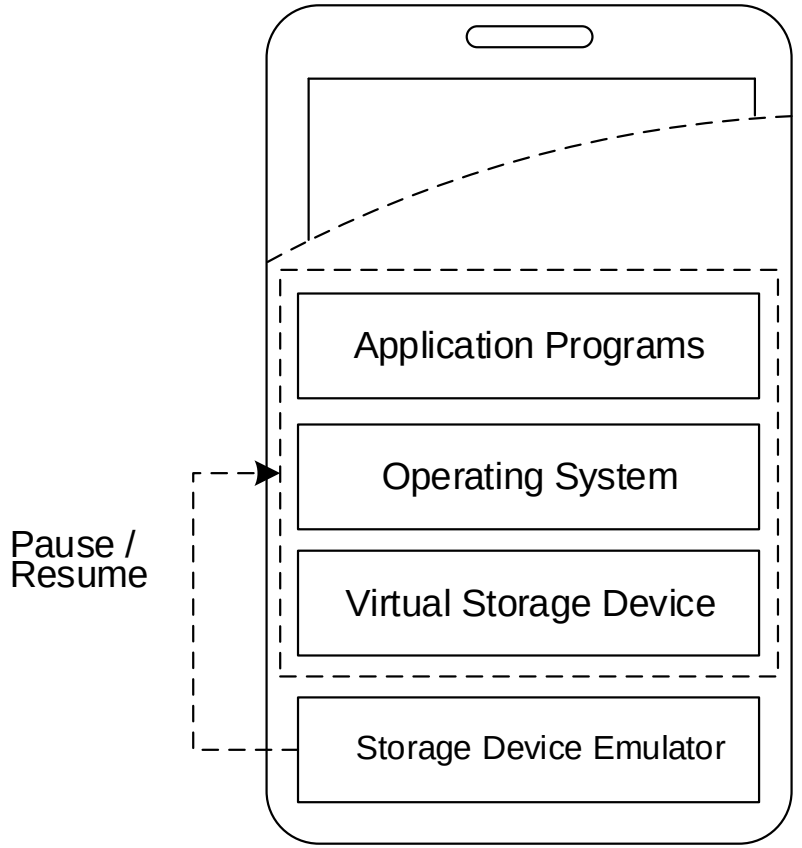
Remote Actual Storage Device Emulation

- Simulates a storage device to the OS over an actual I/O interface
- Backing storage is located on the remote server
- The target system's CPU is not utilized for disk model simulation
- Problems:
  - Performance limited by the actual I/O interface
  - Everything needs to run in real-time



Proposed Approach: Virtual Storage Device and Physical Computer System Co-Simulation

- Much of the problems associated with the conventional storage device emulation approaches are due to everything runs in the real-time domain
- The main idea of this dissertation:
  - Pausing the state of the OS while storage device emulator is busy
  - The OS will be operating in the discrete-time domain



Operating System State Pausing

- The “state” of an OS can be viewed as composed of the combined state of:
  - All the processes running in the OS
  - The current values of the hardware timers used by the OS
- The state of the OS can be paused by:
  - Pausing the hardware counters by writing to their control registers
  - Prevent CPU from doing work for the OS
    - Introduce a high priority blocker thread
    - For SMP systems, use Inter Processor Interrupt (IPI) to block the other CPUs
- All events are delivered to the OS according to the “virtual time”
  - E.g., I/O completion signals

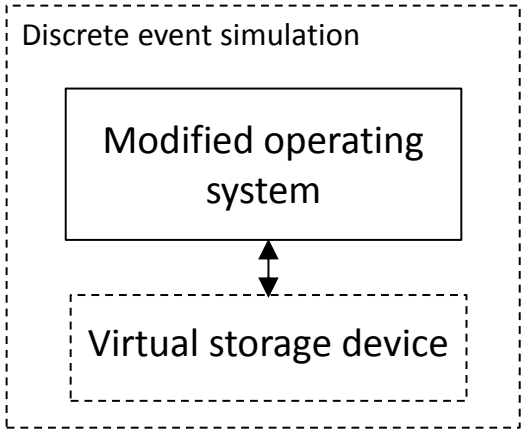
Advantages Over the Conventional Approaches

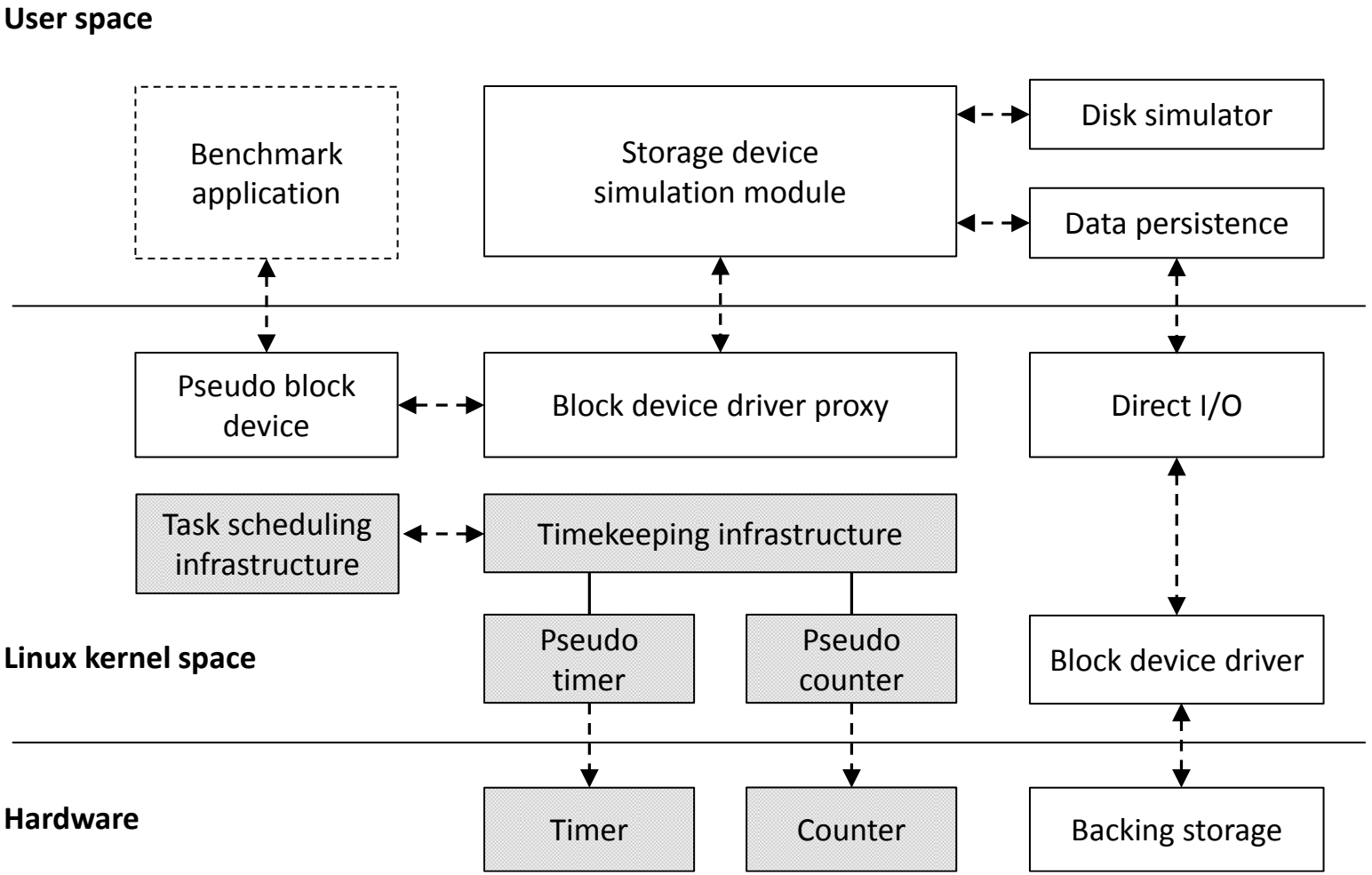
- Compared to complete machine simulation:
  - By running the OS on the real hardware, good system model fidelity and fast simulation speed can be achieved
- Compared to conventional real-time storage device emulation:
  - Due to OS state pausing, makes simulating large and fast storage devices possible
- Can speedup the entire simulation process by fast-forwarding the OS state
  - Up to 45x speedup is observed in the experimental setup

A Full-System Co-Simulator

for storage subsystem evaluation

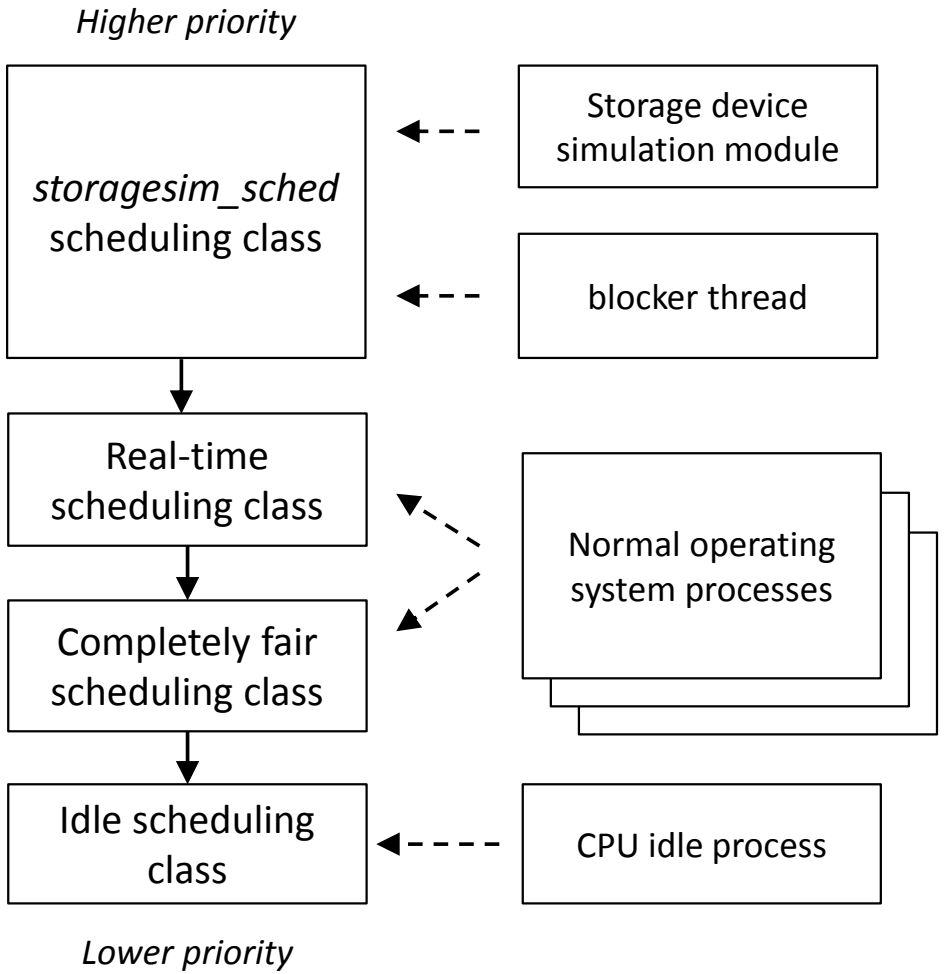
Overview of the Proposed Full-System Co-Simulator





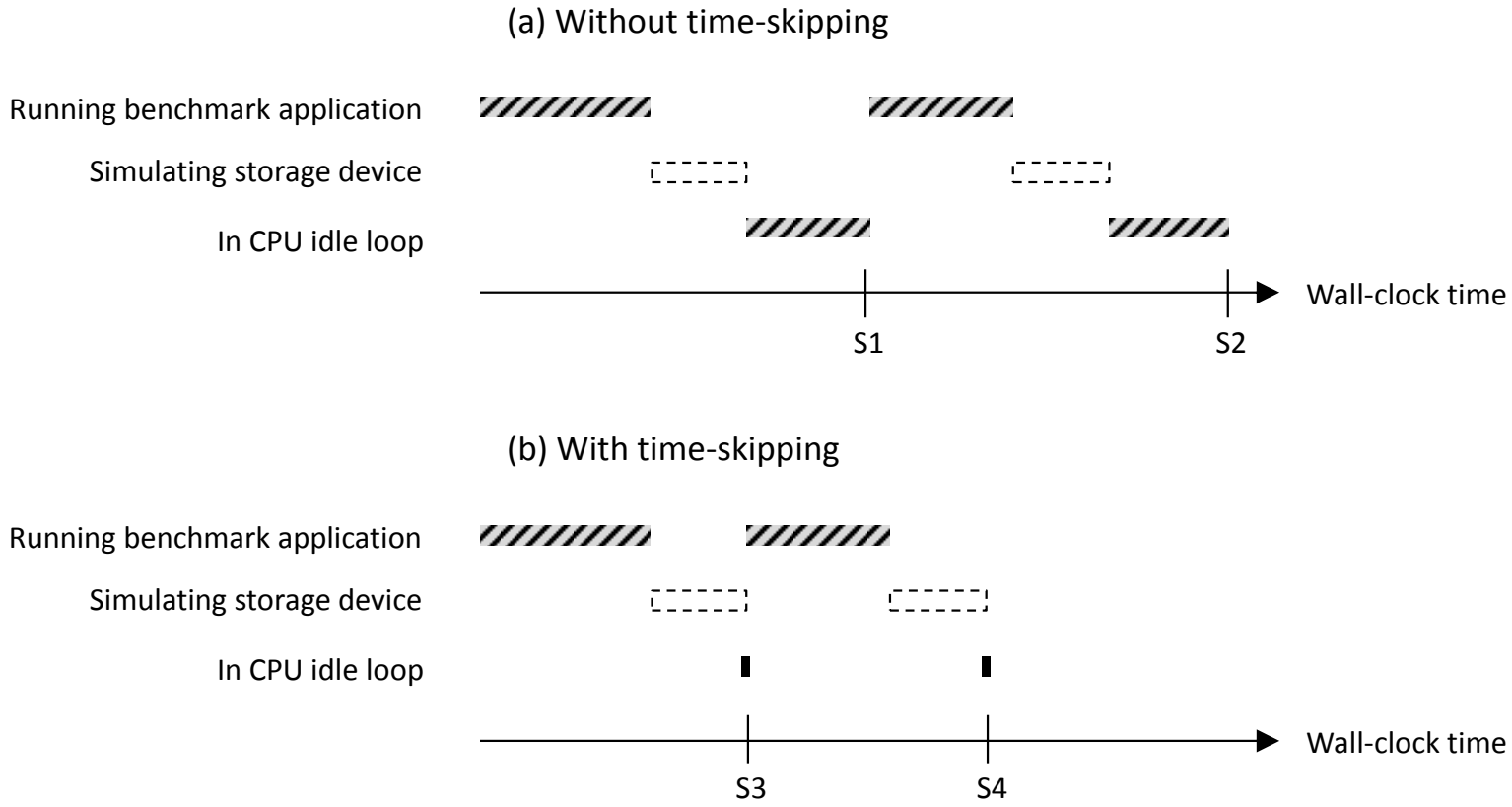
Modified Task Scheduler to Support OS State Pausing

- The storage device simulation module is run as the highest priority thread in the system
  - Can block the CPU that it is on from performing works for the normal OS processes
- A blocker thread is design for:
  - Blocking the other CPUs in SMP systems
    - IPI is used for triggering the blocker threads on the other CPUs



Time-Skipping the CPU Idle Loop

- Fast-forward the system time when CPU is in idle
- OS state invariant
  - State S1 is the same as S3
  - State S2 is the same as S4

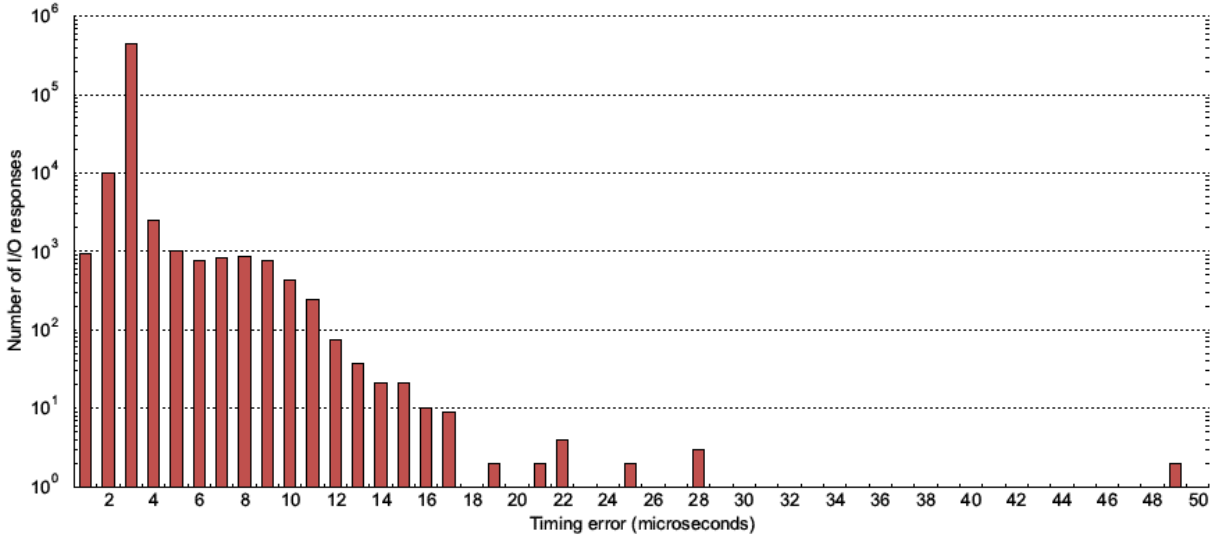


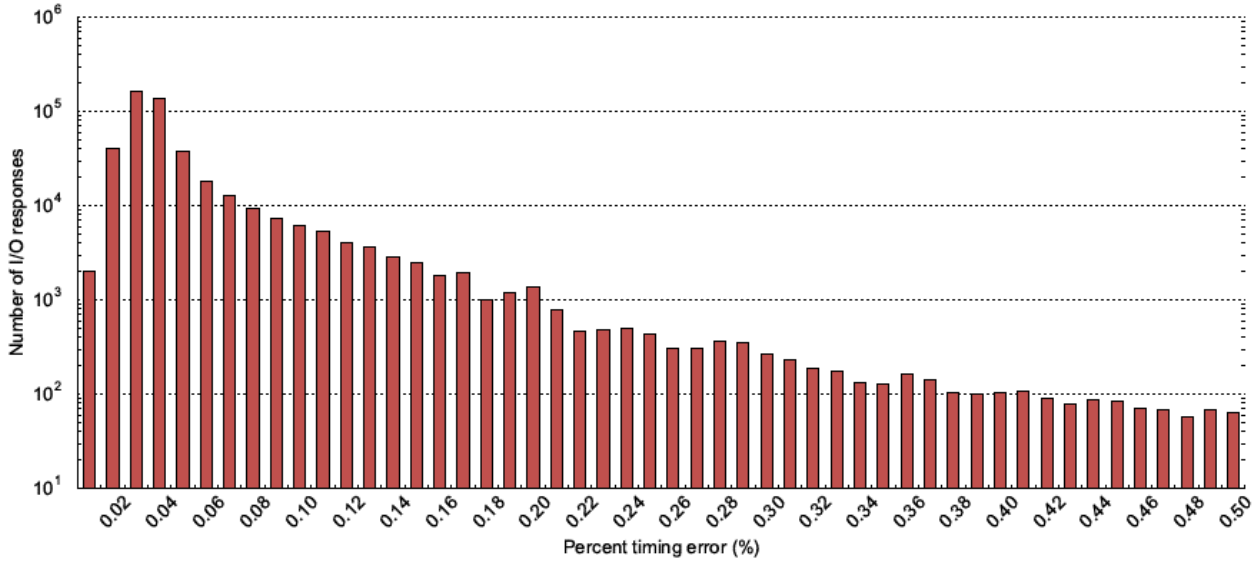
Experimental Setup

- Target system:
  - 3.1 GHz Intel Core i5-3210M processor
  - 16 GB of DDR3 Memory
  - 240 GB Intel 520 series SSD as backing storage
- Disk simulators:
  - DiskSim: simulating a 147GB Seagate Cheetah 15k.5 hard disk drive
  - Vesper: simulating a IBM 08K0383 36GB hard disk drive
- I/O workload generated using Vdbench
  - Read / Write / Read-Write(50% read, 50% write)
  - Random access(100% random) / Mixed access(20% sequential)
  - Small size(4KB) / Large size(128KB) / Uniform size[4KB, 128KB]

Timing Accuracy of the Simulated Storage Device

- I/O response time error can be caused by OS state pause/resume overhead
- The timing difference between the intended I/O response time and the OS observed time
  - Over 99.995% of the I/O responses measured have a timing error value of less than 20µs.
  - Over 99.98% of all the I/O responses measured have less than 0.5% timing error
- Griffin et al. [2002] has shown that performance prediction accuracy is good when over 99% of all emulated I/O responses have less than 2% error





Simulation Speed

Table 4.6: Simulation results for Vesper and RAM

Workload profile	Simulation result			
	I/O per second	Bandwidth (MB/sec)	Elapsed system time	vdbench CPU time (user / sys)
mixed / read / small	146.79	0.57	186.330	2.228 / 0.532
mixed / read / large	73.83	9.23	188.614	2.420 / 0.504
mixed / read / uniform	97.60	6.30	186.335	2.368 / 0.560
mixed / write / small	147.51	0.58	186.387	2.668 / 0.540
mixed / write / large	73.88	9.24	188.355	2.568 / 0.576
mixed / write / uniform	97.95	6.33	187.117	2.852 / 0.540
mixed / read-write / small	147.81	0.58	186.830	2.388 / 0.496
mixed / read-write / large	73.82	9.23	188.101	2.640 / 0.572
mixed / read-write / uniform	97.84	6.37	186.942	2.564 / 0.528

Remarks: The full-system co-simulator is configured to use Vesper as the disk simulator and RAM as the backing storage. The target device modeled is the 36GB IBM 08K0383 hard disk drive. Units are in seconds unless otherwise specified.

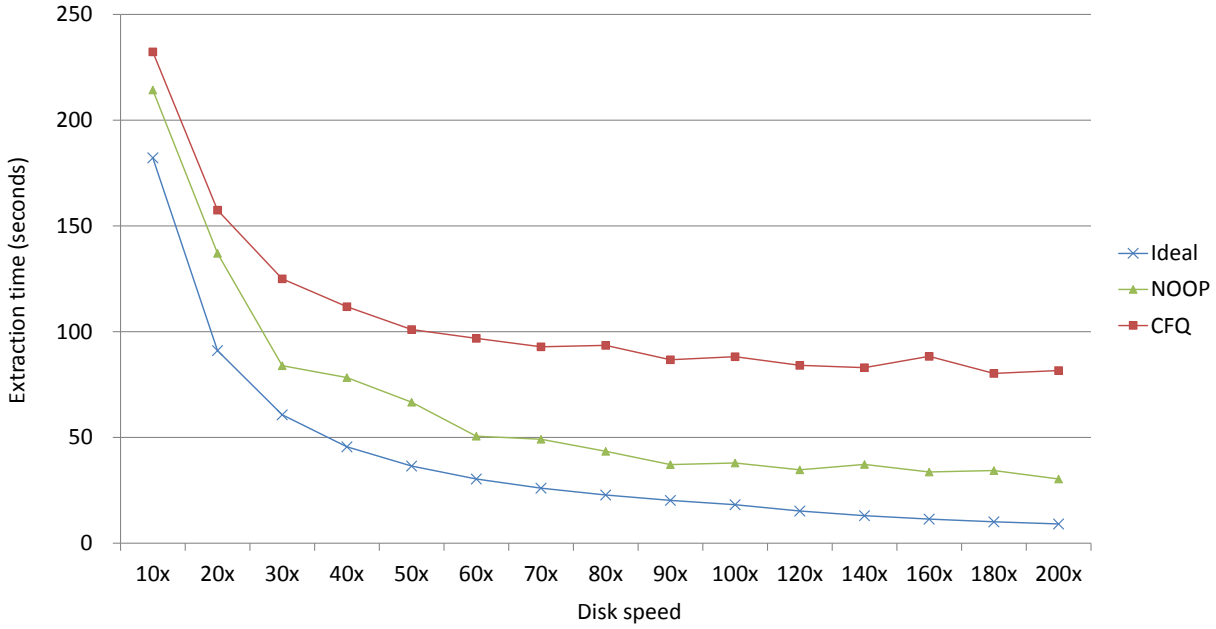
Table 4.7: Simulation speed for Vesper and RAM

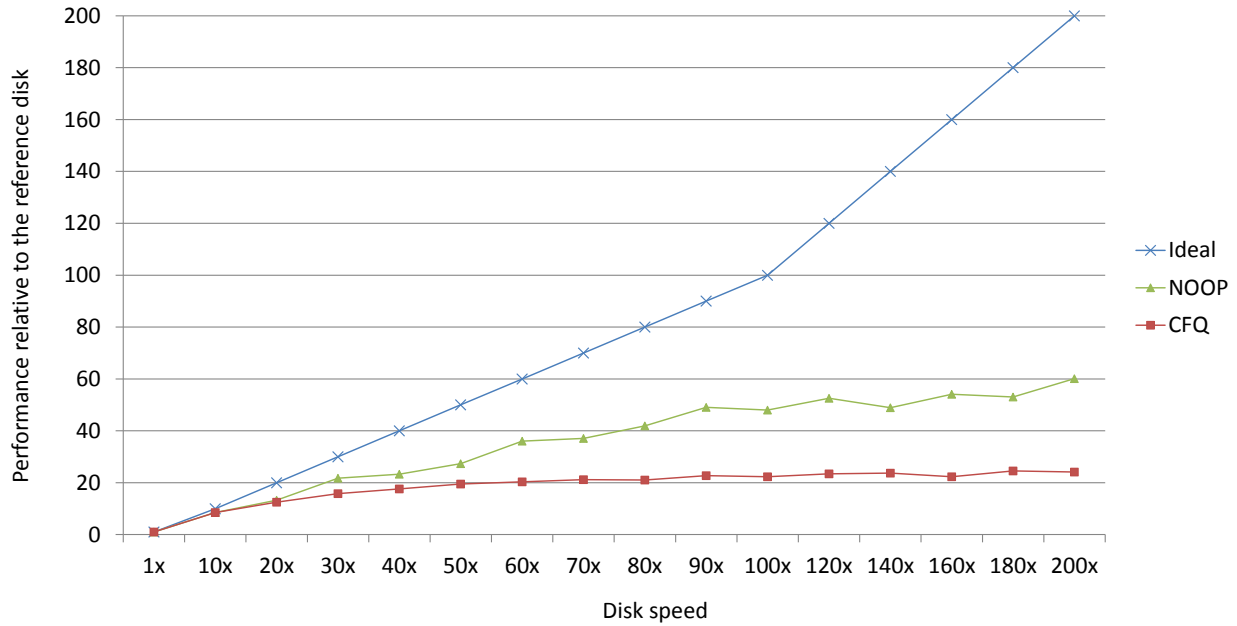
Workload profile	Simulation speed				
	Elapsed wall clock time	In paused mode	DiskSim busy (% of paused mode)	Backing storage busy	Speedup over real-time
mixed / read / small	4.052	0.689	0.036 (5.23%)	0.060	45.99×
mixed / read / large	5.117	1.618	0.033 (2.02%)	0.779	36.86×
mixed / read / uniform	4.784	1.283	0.034 (2.66%)	0.534	38.95×
mixed / write / small	4.562	0.727	0.035 (4.80%)	0.091	40.85×
mixed / write / large	5.941	2.188	0.031 (1.41%)	1.265	31.70×
mixed / write / uniform	5.691	1.683	0.033 (1.94%)	0.866	32.88×
mixed / read-write / small	4.214	0.719	0.036 (5.06%)	0.076	44.33×
mixed / read-write / large	5.712	1.900	0.033 (1.75%)	1.025	32.93×
mixed / read-write / uniform	5.254	1.500	0.034 (2.28%)	0.711	35.58×

Remarks: The full-system co-simulator is configured to use Vesper as the disk simulator and RAM as the backing storage. The target device modeled is the 36GB IBM 08K0383 hard disk drive. Units are in seconds unless otherwise specified.

Application 1 — Effects of Having Faster Storage Devices, 1/4

- Workload: Linux kernel source archive extraction
  - IBM 08K0383 36GB hard disk drive with ext3 file system
- Ideal: Assumes that the system performance scales linearly with disk speed





Application 1 — Effects of Having Faster Storage Devices, 2/4

- Why does the Ideal performance > using NOOP scheduler
- The workload became CPU bounded as the disk speed is increased.

Table 4.8: Simulation results for the NOOP scheduler

Disk speed	Predicted extraction time (seconds)	Speedup over reference disk	Ideal extraction time (seconds)	CPU utilization (% busy)
1× (reference)	1821.706	1.00	-	2.99
10×	214.287	8.50	1821.71	24.92
20×	137.064	13.29	182.17	39.80
30×	83.911	21.71	91.09	59.10
40×	78.298	23.27	60.72	60.46
50×	66.666	27.33	45.54	62.70
60×	50.580	36.02	36.43	69.29
70×	49.169	37.05	30.36	69.76
80×	43.451	41.93	26.02	75.60
90×	37.113	49.09	22.77	86.23
100×	37.949	48.00	20.24	83.43
120×	34.670	52.54	18.22	92.12
140×	37.269	48.88	15.18	95.77
160×	33.692	54.07	13.01	96.36
180×	34.360	53.02	11.39	95.85
200×	30.312	60.10	10.12	97.44

Application 1 — Effects of Having Faster Storage Devices, 3/4

- Why is the performance of using CFQ scheduler < using NOOP?
- At a first guess, maybe the CFQ scheduler is more CPU hungry than the NOOP scheduler?

Table 4.9: Simulation results for the CFQ scheduler

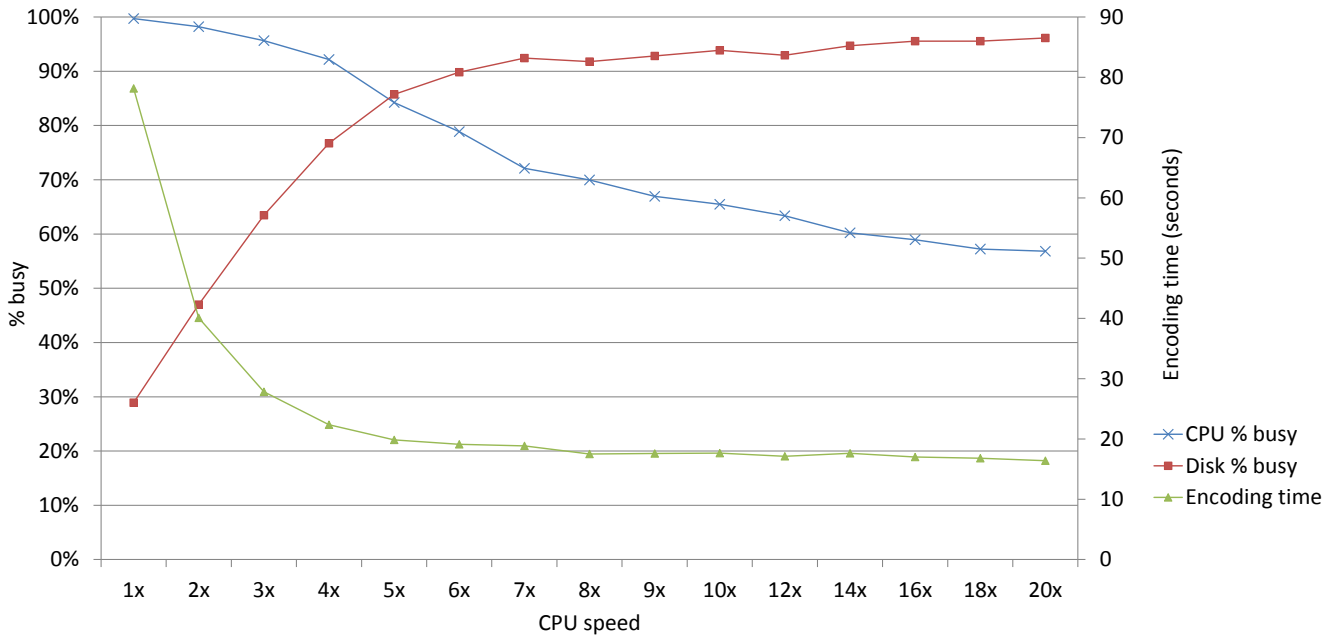
Disk speed	Predicted extraction time (seconds)	Speedup over reference disk	Ideal extraction time (seconds)	CPU utilization (% busy)
1× (reference)	1968.493	1.00	-	2.91
10×	232.285	8.47	196.85	24.03
20×	157.455	12.50	98.42	35.18
30×	124.959	15.75	65.62	43.09
40×	111.831	17.60	49.21	46.56
50×	100.979	19.49	39.37	49.19
60×	96.806	20.33	32.81	51.66
70×	92.814	21.21	28.12	51.01
80×	93.545	21.04	24.61	52.71
90×	86.753	22.69	21.87	52.93
100×	88.153	22.33	19.68	53.64
120×	84.121	23.40	16.40	54.43
140×	83.012	23.71	14.06	55.47
160×	88.324	22.29	12.30	53.98
180×	80.226	24.54	10.94	54.26
200×	81.584	24.13	9.84	55.26

Application 1 — Effects of Having Faster Storage Devices, 4/4

- Upon further investigation, I found that the CFQ scheduler has a default anticipatory scheduling behavior:
  - Disabling this anticipatory behavior makes the performance of using the CFQ scheduler comparable to the NOOP scheduler
- A good example of having the ability to explore the design space with realistic full-system models
  - For example, RAID rebuild performance can also be affected by the selected I/O scheduler

Application 2 — Video Encoding Performance on Systems with Different CPU Speeds

- Systems with difference CPU speeds can be simulated by manipulating the advancement of the pseudo system time
- Testing workload:
  - Generated by running 5 duplicates of the ffmpeg program (version 0.7.6) encoding the 300-frame standard MPEG CIF Foreman test sequence
  - Writing to the IBM 08K0383 36GB hard disk drive with ext3 file system
- Observation: Disk I/O becomes the bottleneck as the CPU speed increases



A Storage Device Emulator

for system performance evaluation

- Ming-Ju Wu and Chun-Jen Tsai. 2015. **A Storage Device Emulator for System Performance Evaluation**. *ACM Trans. Embed. Comput. Syst.* 14, 4, Article 82 (October 2015)

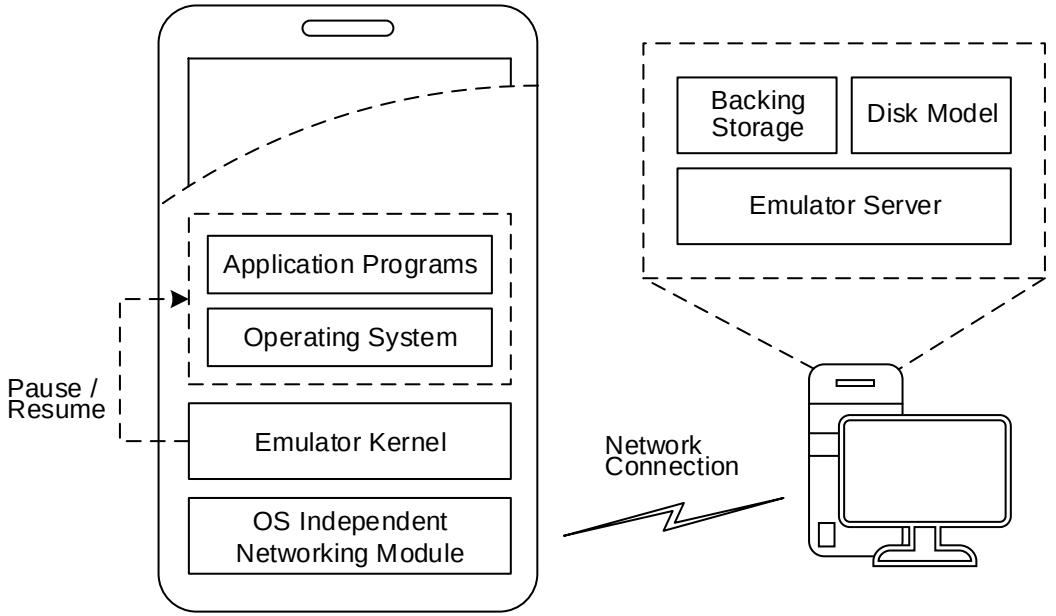
†Open sourced and is available at: http://www.cs.nctu.edu.tw/~cjtsai/research/nctusde/

Motivation: Predicting the Performance of a Specific System

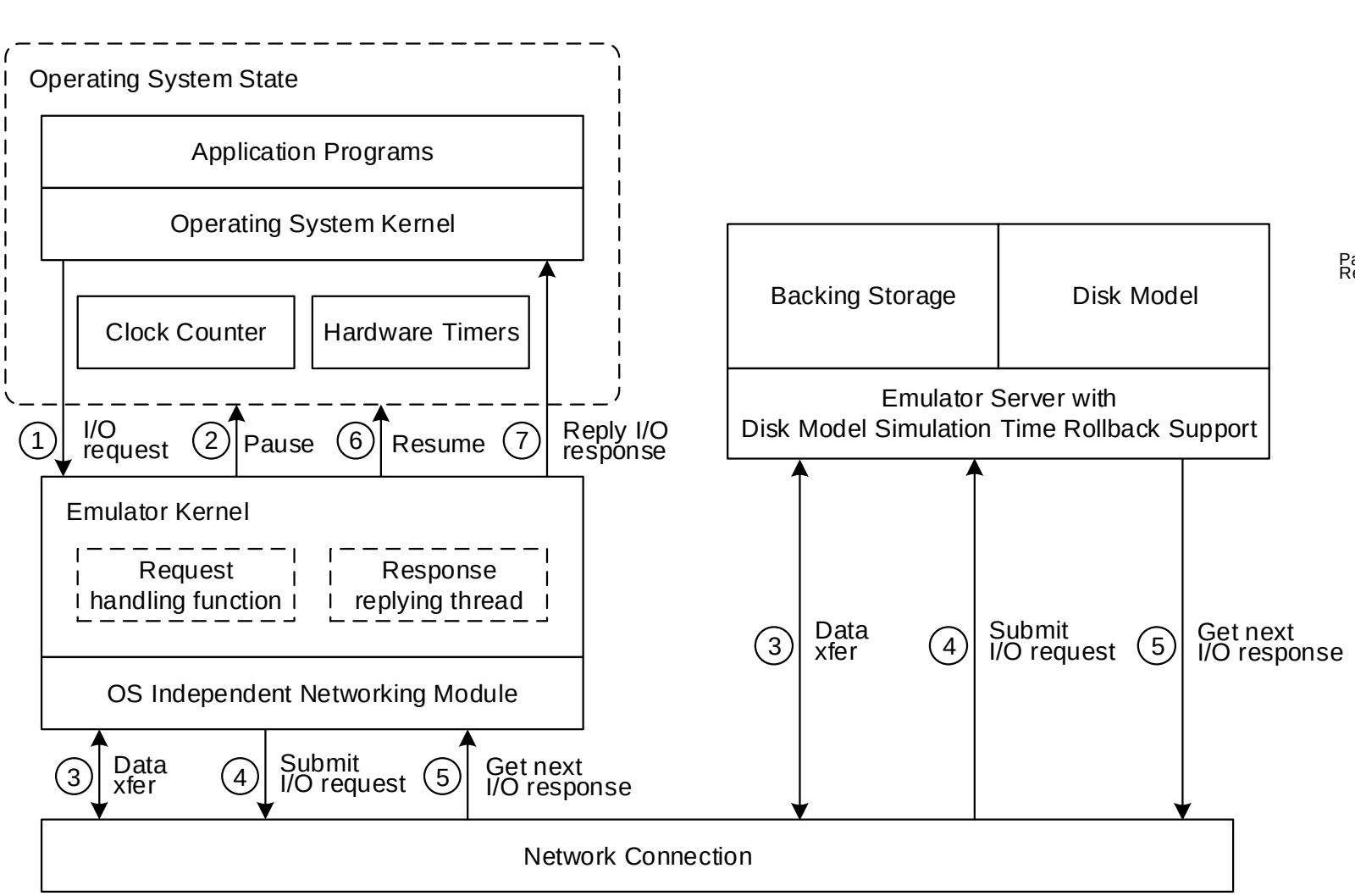
- The proposed full-system co-simulation environment is good for:
  - Generating realistic workloads towards the virtual storage device
  - Benchmark the virtual storage device with realistic system-level metrics
- However, the accuracy for using it to predict the performance of a specific system is not yet established
  - For example: Answering the question of what is the performance of a specific embedded system when equipped with a certain storage device?
- Things that can affect the behaviors of the target system:
  - When simulating the storage device:
    - CPU cache content interference
  - When performing OS state pausing:
    - The speed at which the OS can be paused/resumed
    - The resolution of the underlying timer hardware

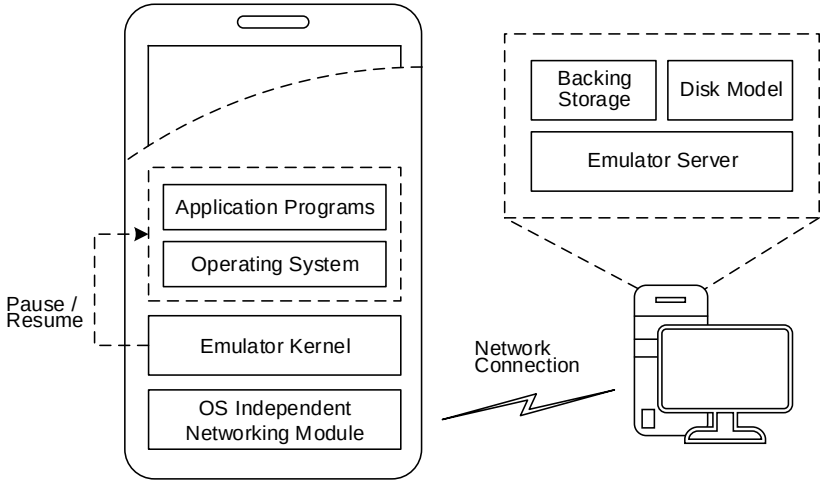
Proposed Storage Device Emulator

- The task of disk emulation is split into two parts:
  - An emulator kernel that emulates a storage device at the block device driver level
  - An emulator server that is responsible for data persistence and I/O timing simulation
- Emulator server runs on a separate computer
  - Minimizes cache interference to the target system
  - Communicates with emulator kernel using network connection
  - Communication takes place while the OS is paused
    - Virtual storage device performance will not be limited by the network speed



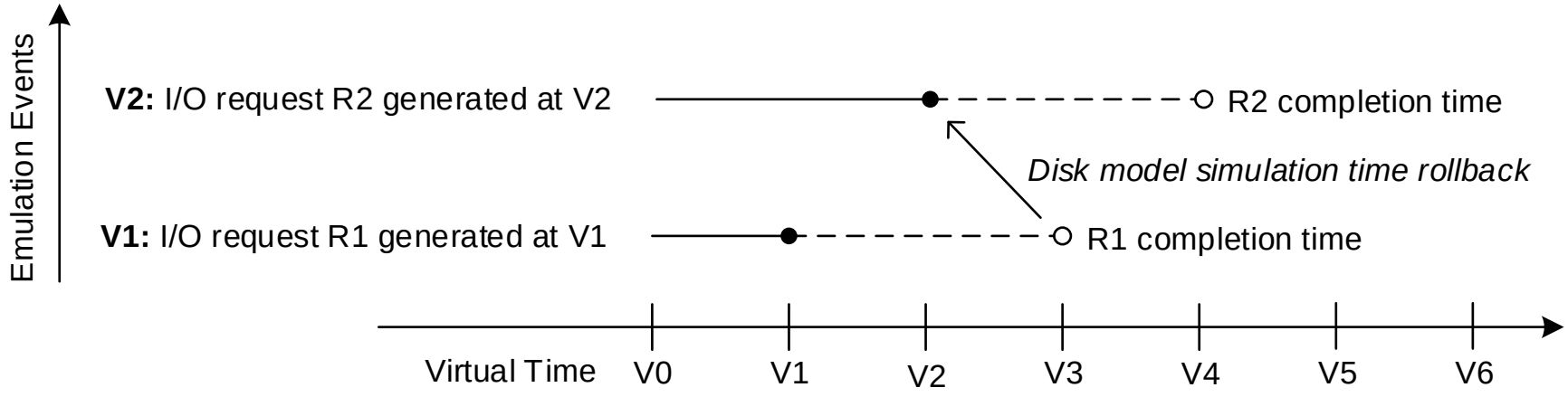
Proposed Storage Device Emulator – Detailed Operation





Emulator Server with Disk Model Simulation Time Rollback Support

- Disk model is always simulated to a “future” time to determine the next completed I/O
- The next completed I/O can change due to later I/O requests
  - I/O prioritization
  - Storage devices that support multi-channel accesses
- When a new I/O request is received, the state of the disk model is rolled-back and re-simulated



OS Independent Network Module

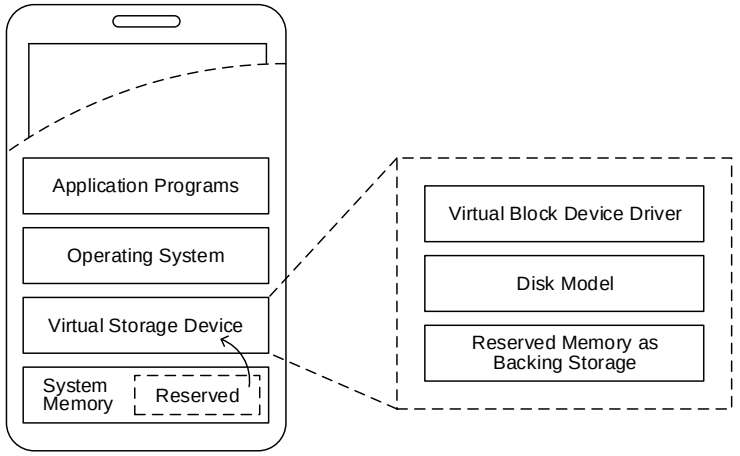
- An OS Independent Network Module is developed because:
  - Interrupts are disabled while OS is paused
  - Stock network device driver requires interrupt for proper functioning
    - A pure polling-based device driver is developed
  - Buffers used by the networking layer are allocated from non-cached memory region
    - To minimize cache interference
  - Hardware timers are also paused while OS is in paused state
    - TCP retransmission control will not work
    - Implemented our own polling-based retransmission protocal over UDP

Experimental Setup

- Target hardware platform
  - The ZedBoard development board
  - 667MHz dual-core ARM Cortex-A9 MPCore processor
  - 512MB of DDR3 main memory
  - 1Gbps Ethernet LAN
  - Timer hardware provides a timing resolution of approximately 3*ns*
- Kernel version 3.10.0 from the Xilinx source repository
  - With high-resolution timer capability enabled
    - Avoids “half tick” loss when OS state is paused
- Root file system: *nano* build of the Linaro release version 13.11

Reference System

- A reference system is used for measuring the disturbances that OS state pausing can cause
- Potential source of disturbances:
  - CPU blocker thread activation latency
  - Timer hardware resolution
  - CPU cache content interference
- The reference system comprises:
  - An unmodified Linux OS running on the same hardware platform
  - A RAMDISK storage device with artificial delay to simulate I/O processing time
    - The size of the RAMDISK is kept small to fit in memory
    - Very simple disk model to minimize CPU overhead



Storage Device Configurations

- Baseline storage device modeled after Transcend UHS Class 1 SD card
- Configuration ChN-xS means:
  - N: The number of concurrent processing channels
  - S: Speedup factor

Transfer Size	Random Workload		Sequential Workload	
	Read (ms)	Write (ms)	Read (ms)	Write (ms)
512 bytes	0.0051	1.1136	0.0019	0.0092
1 KB	0.0053	1.1123	0.0020	0.0095
2 KB	0.0054	1.1136	0.0023	0.0099
4 KB	0.0057	1.1601	0.0027	0.0115
8 KB	0.0063	1.2853	0.0033	0.0136
16 KB	0.0071	1.2579	0.0036	0.0035
32 KB	0.0096	2.2222	0.0068	0.0070
64 KB	0.0151	2.7027	0.0139	0.0137

Performance Numbers Used for the Baseline Storage Device

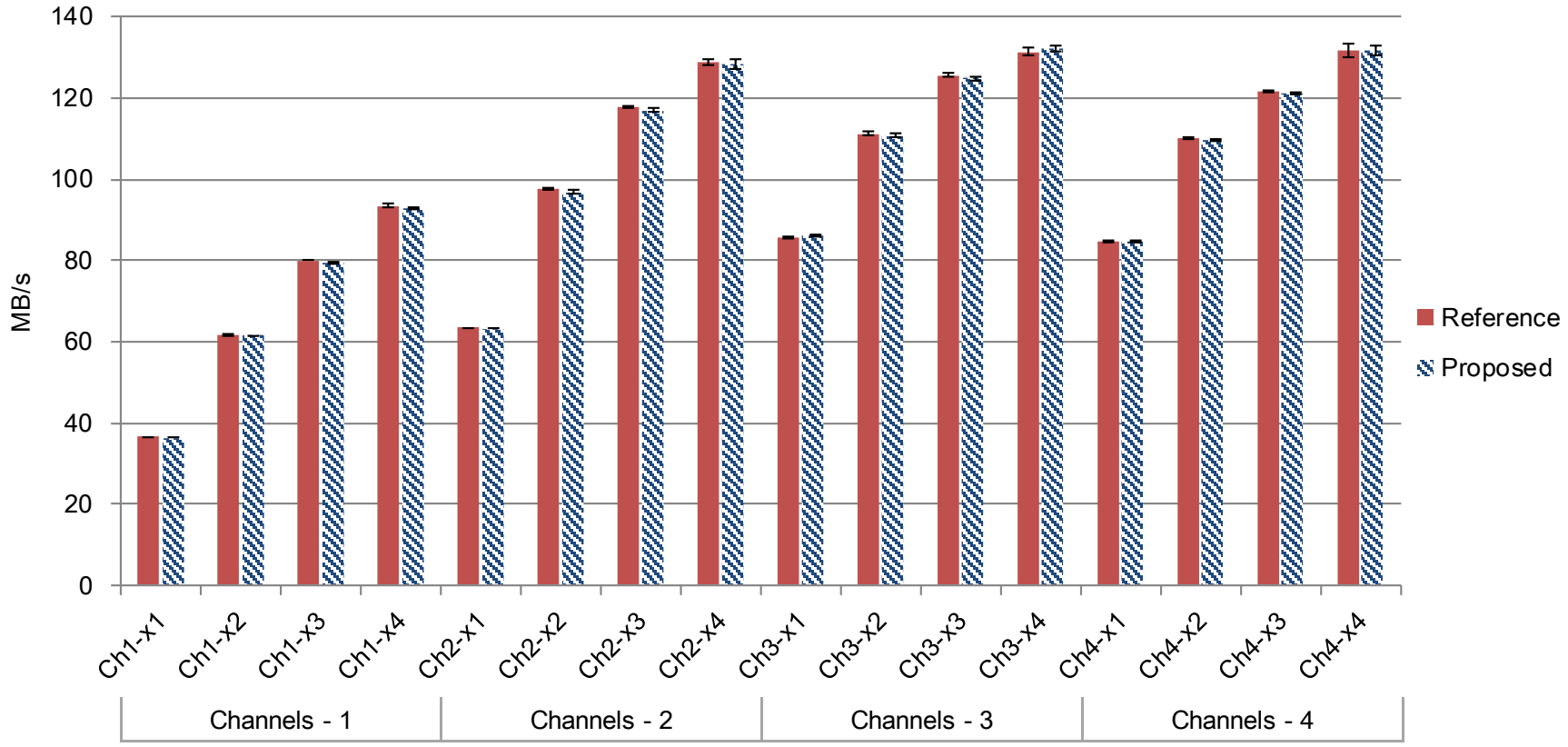
Experimental Results

- Automated testing using the auto-pilot automation framework
- Each number is average of at least 10 runs
  - And until the radius of the 95% confident interval is less than 5% of the mean
- Error bars represent the 95% confidence intervals of the means.
- All results from the proposed storage device emulator are within 2% differences to the results from the reference system.

Charles P. Wright, Nikolai Joukov, Devaki Kulkarni, Yevgeniy Miretskiy, and Erez Zadok. 2005. **Auto-pilot: a platform for system software benchmarking**. In *Proceedings of the annual conference on USENIX Annual Technical Conference* (ATEC '05). USENIX Association, Berkeley, CA, USA, 53-53.

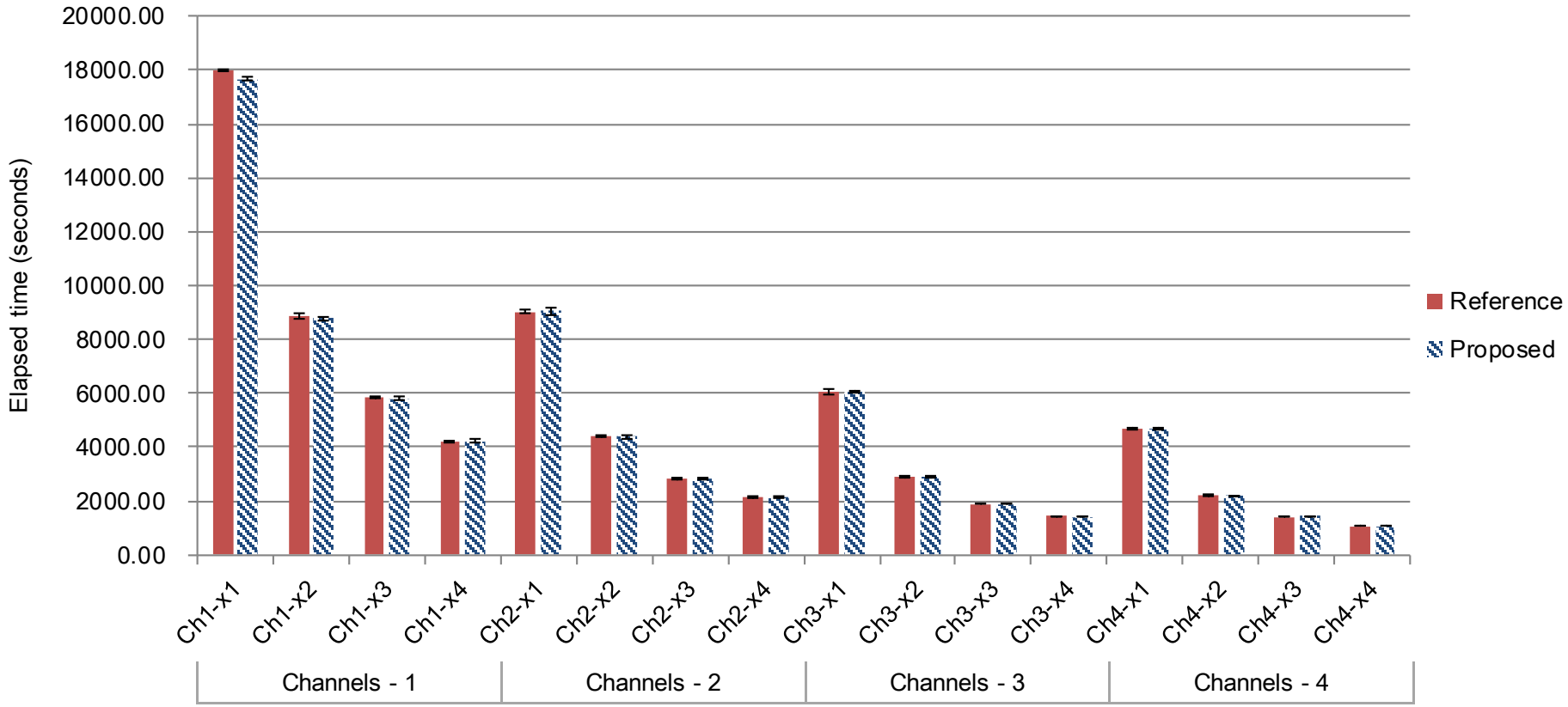
Sequential Read Bandwidth Comparison

- Sequential read bandwidth measured using the *hdparm* utility



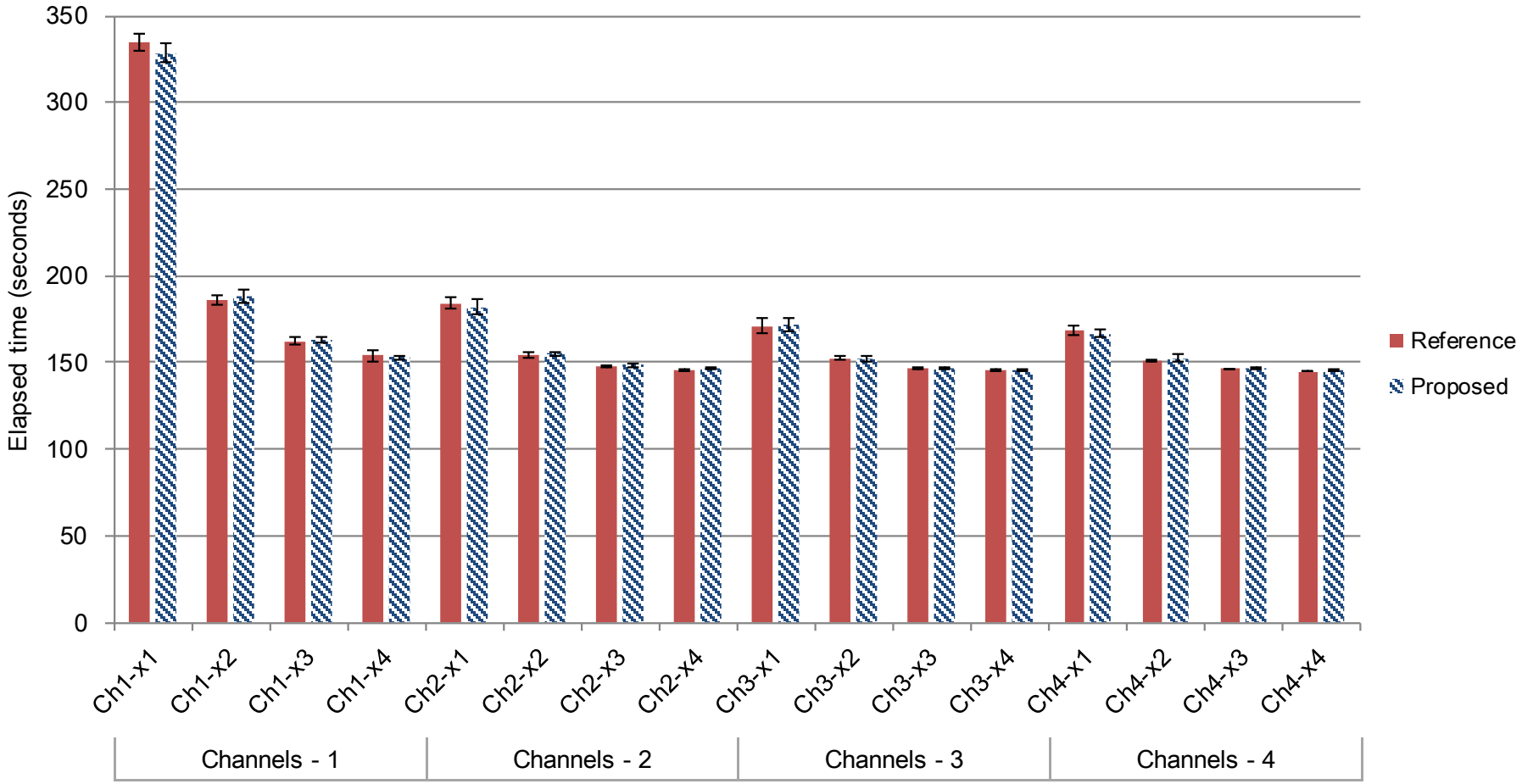
Postmark Workload Comparison

- Ext3 filesystem. noop scheduler and CFQ scheduler.
- 8 concurrent postmark threads to generate a total of 100,000 transactions over a set of 10,000 files



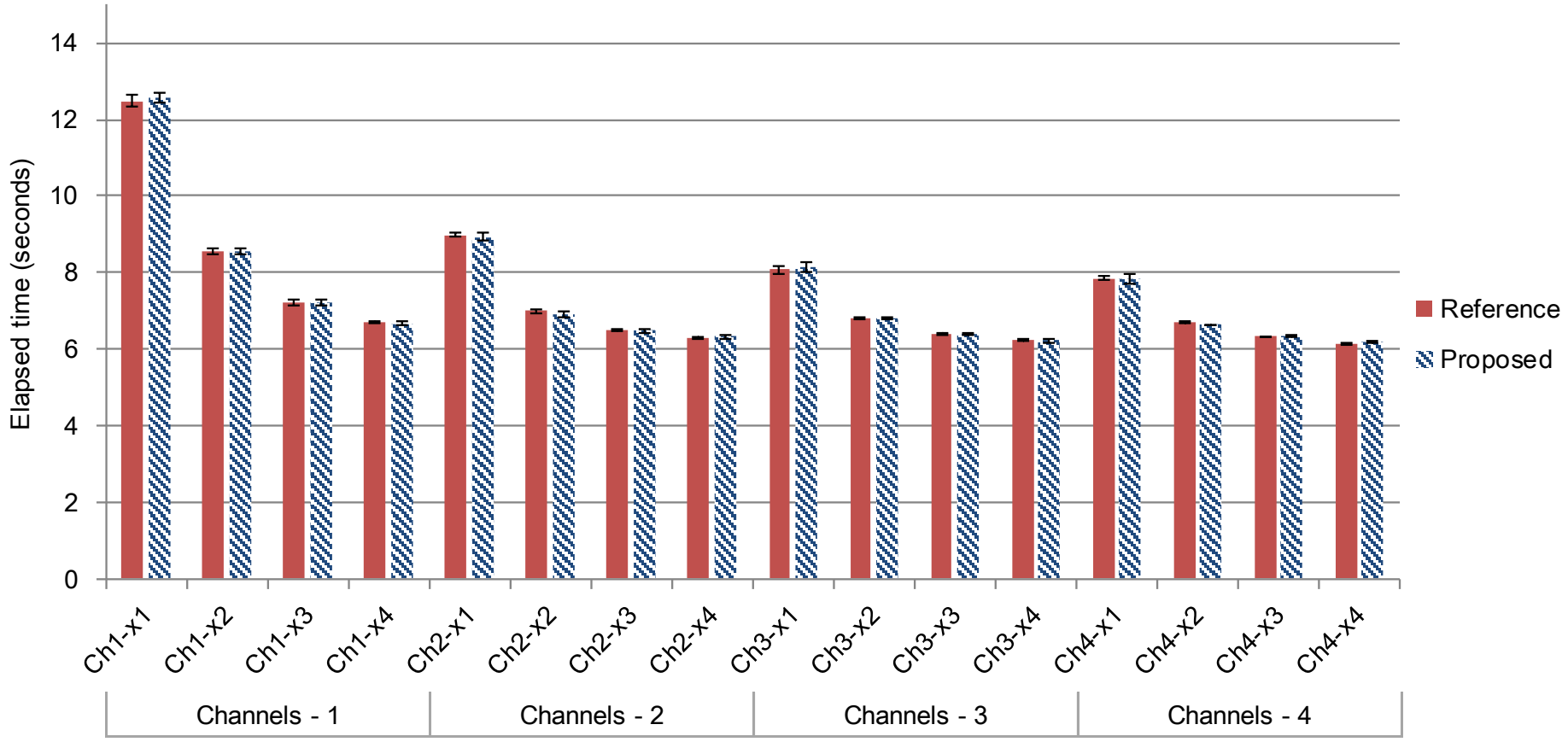
Kernel Source Extraction and Word Count Comparison

- Ext4 filesystem. noop scheduler and CFQ scheduler.
- Extracts *linux-3.10.tar.bz2 and then do word count*



Video Encoding and Decoding Comparison

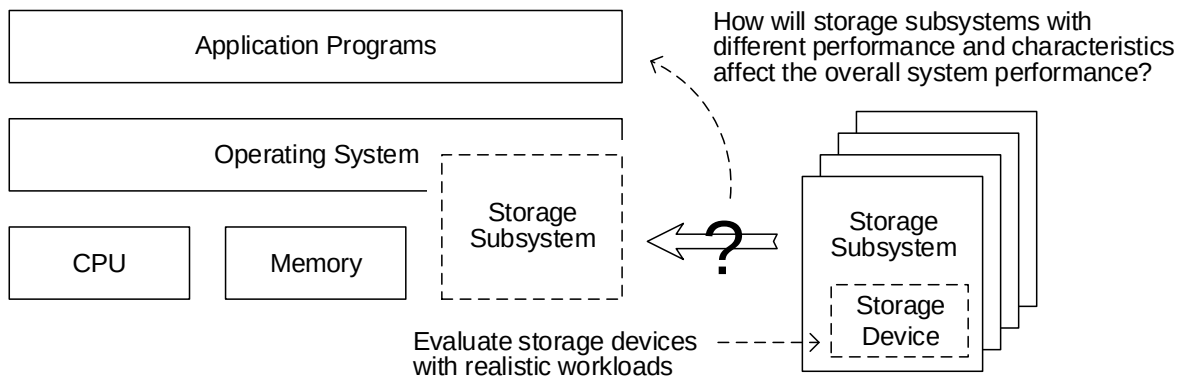
- Ext3 filesystem. noop scheduler and CFQ scheduler.
- CIF resolution foreman video sequence is first copied to the ext3 filesystem. Encoded to the MPEG2 format. Then decoded back to the raw format.



Conclusions and Future Work

Conclusions

- This dissertation proposed a novel OS state pausing approach that enables the co-simulation of virtual storage device and physical computer system
- Designed and implemented two co-simulation environments
- In general, the co-simulation environments can:
  - Be used for evaluating storage subsystem designs with realistic workloads and system-level metrics
  - be used for predicting the overall system performance when different storage devices are available



Future Work

- Support for emulating multiple storage devices
  - Can support for studying software RAID array behaviors
- Pausing the system state from a lower level
  - For example, by stopping the clocks of the digital circuits
- Simulate the thermal behaviors of the target system