Quantifying benefits of offloading data management to storage devices

Jianshen Liu, Philip Kufeldt, Carlos Maltzahn

Computer Science and Engineering, University of California, Santa Cruz

This project is funded in part by NSF OAC-1836650, CNS-1764102, CNS-1705021, and the Center for Research in Open Source Software (cross.ucsc.edu).

Abstract

- **Problem**
  It is difficult to quantify the benefits of offloading data management from hosts to storage devices.
- **Approach**
  Quantify the benefits based on a reference point in order to formalize a fair comparison.
- **Data Management**
  Manage data layout and placement in storage devices to ensure its durability and availability (e.g., compaction, deduplication, scrubbing, redundancy, recovery, rebalancing).
- **MBWU (pronounced "MibeeWu")**
  Media-based Work Unit, as a reference point, is defined by a combination of a storage device and workload and measured in IOPS.

Trends in Technology

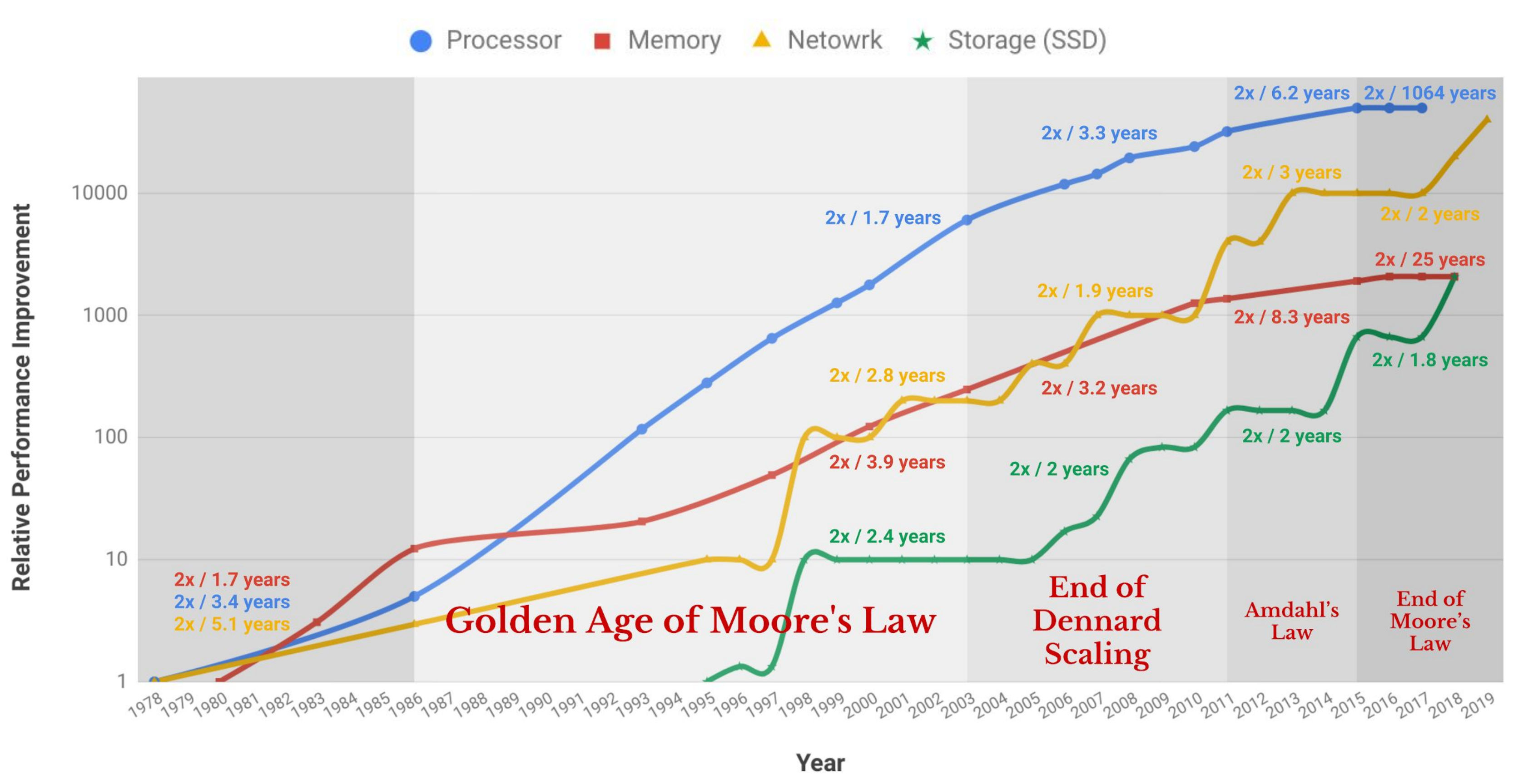


Fig. 1: Relative Performance for microprocessor, memory, network and storage (SSD). Performance of processors is relative to the VAX 11/780 as measured by the SPEC integer benchmarks. Performance of memory, network and storage refer to bandwidth.

Data Source:
- John L. Hennessy and David A. Patterson. 2017. Computer Architecture, Sixth Edition: A Quantitative Approach (6th ed.).
- Allen Samuels. The Consequences of Infinite Storage Bandwidth. Engineering Fellow, Systems and Software Solution. April 21, 2016
- https://en.wikipedia.org/wiki/List_of_interface_bit_rates#Dynamic_random-access_memory

When performance improvement of network and storage significantly outstrips the improvement of microprocessor and memory, what can we do?

By 2020, if 20% of memory bandwidth on a host is used for storage I/O, an CPU socket can only serve less than 4 SSDs (Fig. 2).

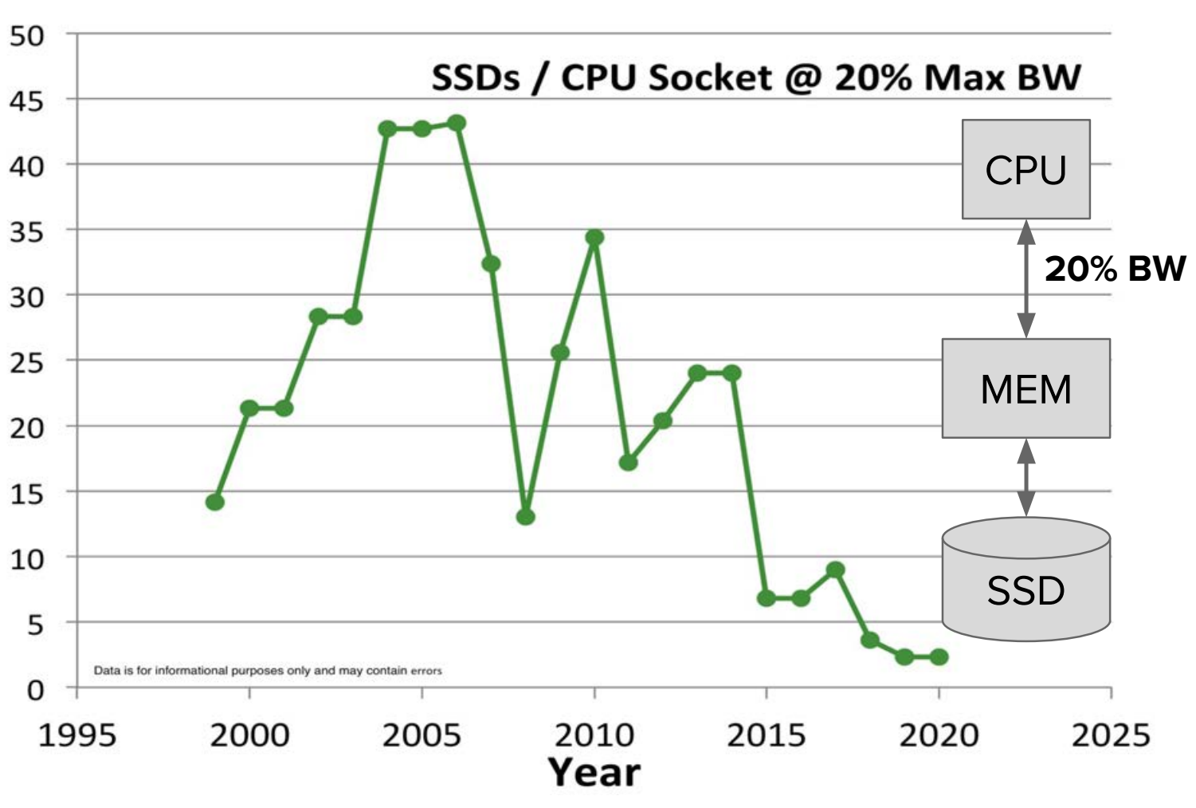


Fig. 2: Trend of SSDs/ CPU Socket.
Source: Allen Samuels. The Consequences of Infinite Storage Bandwidth. Engineering Fellow, Systems and Software Solution. April 21, 2016

We need to reduce the traffic between host I/O bus and storage devices.

Opportunities:
- Embedded platforms have better price/performance
- Embedded processors are getting more powerful
- Domain-specific hardware is the future! [1,2]

Data Management is ideal for offloading:
- It is data intensive
- It is responsible for a significant fraction of host-to-device traffic

Platform Comparison

Conventional servers (as host platforms) and storage devices (as embedded platforms) are hard to compare due to their significant differences in software and hardware design.

Benefit evaluations in existing research generally run into performance comparison problems because:

- Evaluation results do not isolate impacts from different implementations of data management functionality.
- Modification of the device firmware changes storage device performance

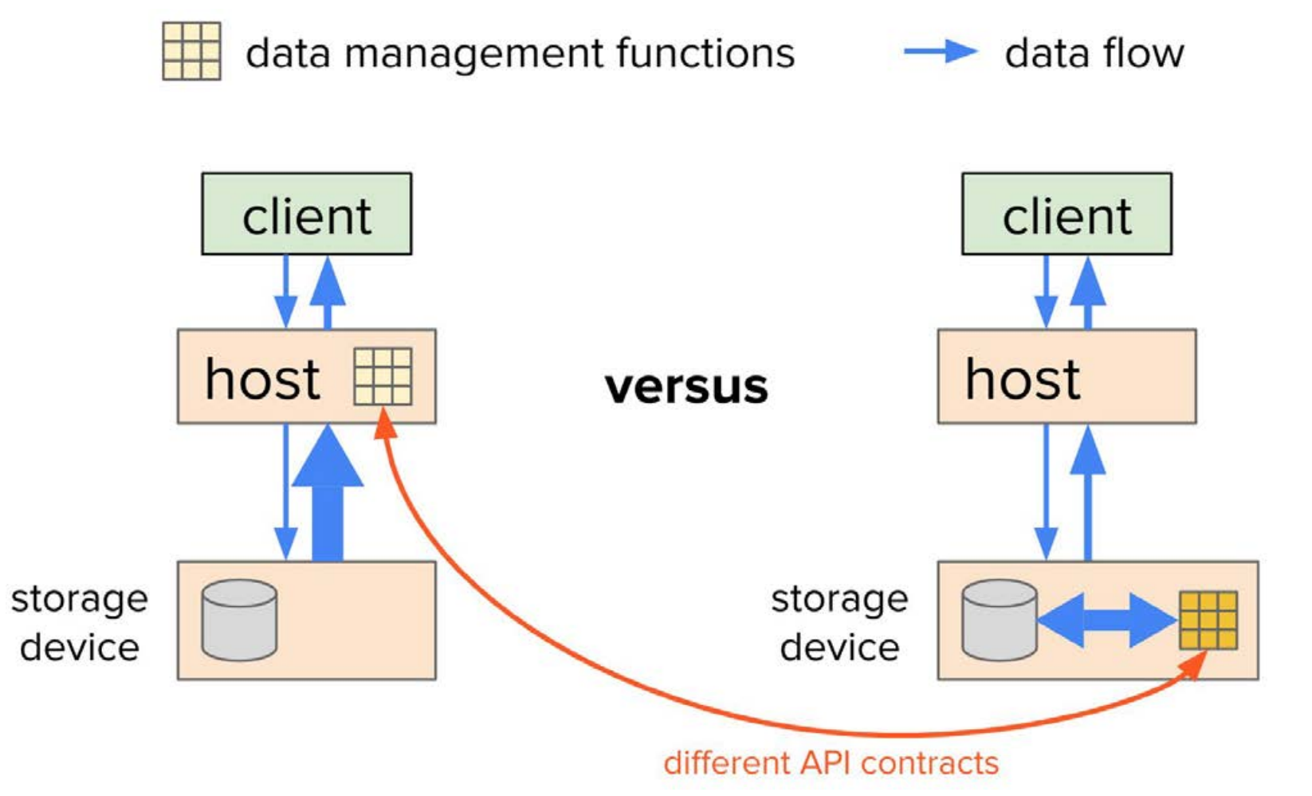


Fig. 3: Benefit evaluation of existing research

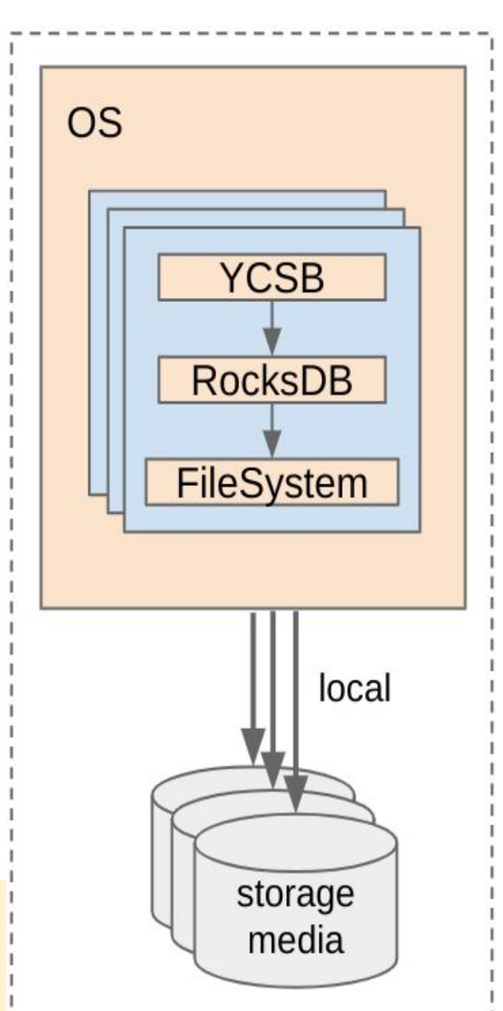
MBWU is dependent only on the **storage device** and **workload** making a reference point that is truly **independent** from the **platform**.

- The value of a single MBWU should not be throttled by any system resources on the platform. (e.g., CPU, memory).
- MBWU specifies a performance reference point for normalizing capabilities of different platforms for running a specific workload.

The capabilities of different platforms can be evaluated in terms of the number of MBWUs they can generate.

- Host platforms may be powerful enough to generate multiple MBWUs before hitting other bottlenecks.
- Embedded platforms may only be able to generate a fraction of an MBWU.

Fig. 4: Evaluate MBWUs with multiple storage devices



We can now compare the total cost of ownership of these platforms based on the MBWUs (e.g., $/MBWU, kW·h/MBWU, m²/MBWU).

MBWU is useful for evaluating the benefits of any data management offloading to storage devices. (e.g., management of key/value data, full-memory encryption for persistent-memory technologies)

Evaluation

To demonstrate the use of this **platform-agnostic measurement methodology**, we evaluate the benefits of offloading key-value data management to storage devices. Configurations of the two platforms used in our experiment are shown in Table I.

Before running the workload, we precondition the SSDs with two different preconditioning configurations (one covers the first **75%** of LBAs and the other covers the entire LBA space) to illustrate the sensitivity of the evaluation results. The workload is a steady load containing key-value, read-and-write requests generated by **YCSB** to the underlying **RocksDB** that in turn uses the SSD. The detail evaluation process is shown in Fig. 6.

Platform	# of SSDs	CPU	Memory	Cost
Host	8	24-cores	64 GB	$4,100
Embedded	1	hexa-core	4 GB	$120

TABLE I: Configuration of Two Platforms

The key-value **workload** used in our experiment is:

- Key is 16 bytes, value is 4 KiB
- The read/write ratio is 50%
- The popularity of keys follows a Zipf distribution
- The size of dataset is 40 GiB.
- The number of key-value operations is 10,444,959 (~40 GiB).

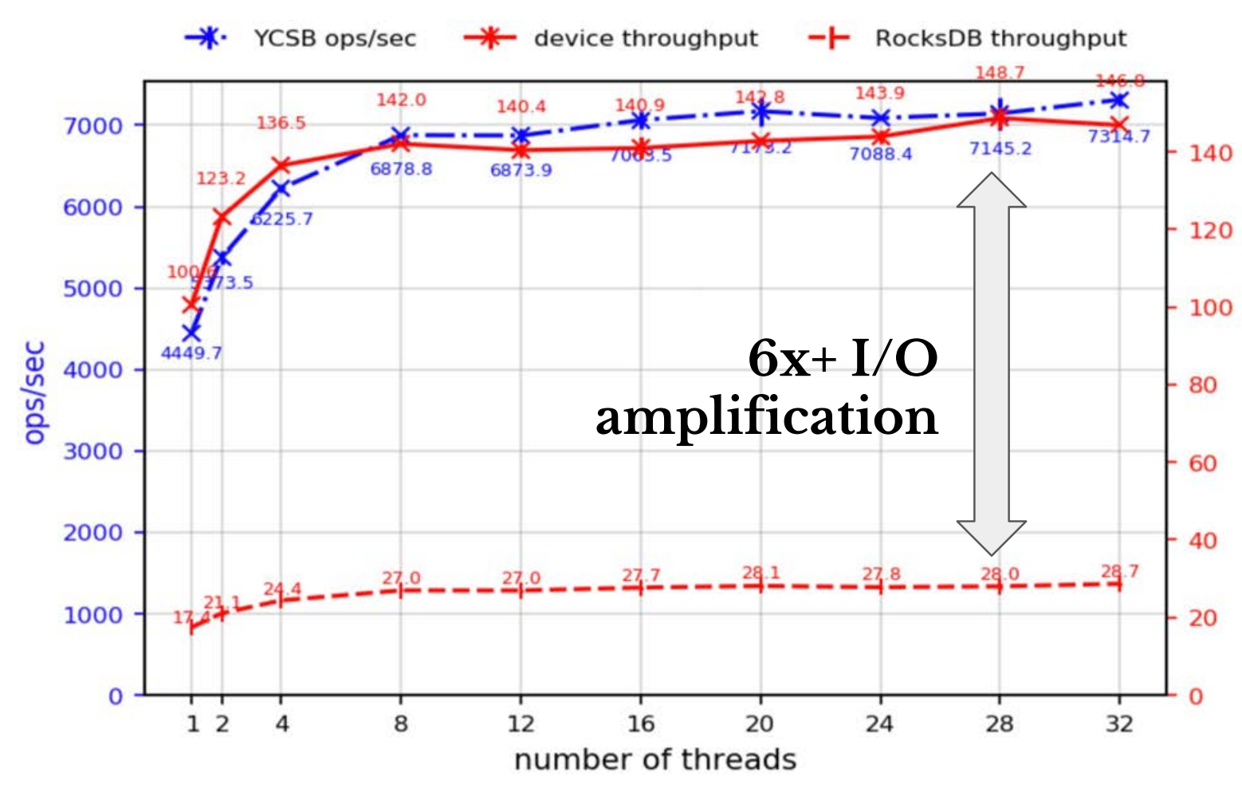


Fig. 5: Key-value data management occupies a significant fraction of host-to-device traffic

Precondition	75% LABs		100% LABs	
Platform	# of MBWUs	Current	# of MBWUs	Current
Host	6	1.73A	7.5	1.53A
Embedded	0.5	0.087A	0.5	0.08A

TABLE II: Evaluation Results

The evaluation results are shown in Table II. We have built a tool to automate the evaluation process. This tool allows to customize the characteristics of a workload with more options such as ratio of scan and read-modify-write operations.

With these numbers (Table II), for example, if the LBAs of the SSDs are 75% preconditioned, the performance of **12** embedded platforms is equivalent to the performance of **one** host platform. Therefore, the embedded platforms can save 65% of the cost per MBWU compared to running the same key-value workload on the host platform, and they can save 39.6% of energy per MBWU as well.

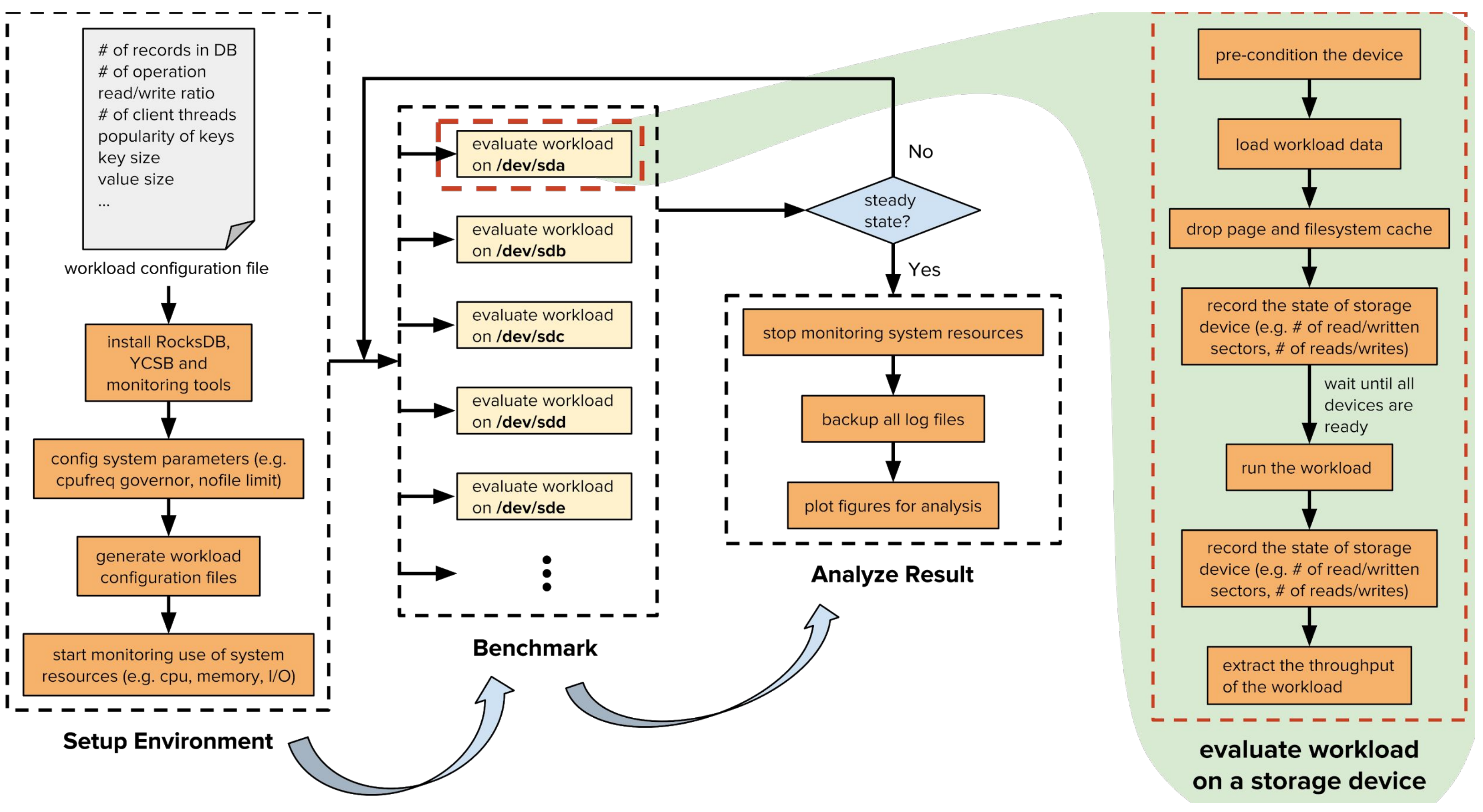

# of records in DB
# of operation
read/write ratio
# of client threads
popularity of keys
key size
value size
...
workload configuration file
install RocksDB, YCSB and monitoring tools
config system parameters (e.g. cpufreq governor, nofile limit)
generate workload configuration files
start monitoring use of system resources (e.g. cpu, memory, I/O)
Setup Environment
evaluate workload on /dev/sda
evaluate workload on /dev/sdb
evaluate workload on /dev/sdc
evaluate workload on /dev/sdd
evaluate workload on /dev/sde
Benchmark
steady state?
No
Yes
stop monitoring system resources
backup all log files
plot figures for analysis
Analyze Result
pre-condition the device
load workload data
drop page and filesystem cache
record the state of storage device (e.g. # of read/written sectors, # of reads/writes)
wait until all devices are ready
run the workload
record the state of storage device (e.g. # of read/written sectors, # of reads/writes)
extract the throughput of the workload
evaluate workload on a storage device

Fig. 6: Evaluation Process in Detail

Baskin Engineering UC SANTA CRUZ

CROSS CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

References

[1] Hennessy, John, and David Patterson. "A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enhanced."
[2] Patterson, David. "50 Years of computer architecture: From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set." Solid-State Circuits Conference-(ISSCC), 2018 IEEE International. IEEE, 2018.

Contact

Jianshen Liu (jliu120@ucsc.edu), UC Santa Cruz
Philip Kufeldt (pkufeldt@ucsc.edu), UC Santa Cruz
Carlos Maltzahn (carlosm@ucsc.edu), UC Santa Cruz