■ Related papers: summary (key contribution), merits, demerits

No	Category	Year	Conf & Journal	Title	Summary (Key contribution)	Strong Point	Weak Point
1	Conf	1999	ATC	**Improving Application Performance through Swap Compression**	The performance of large applications tends to be poor due to the high overhead added by the swapping mechanism. The same problem may be found in highly-loaded multi-programmed systems where many of the running applications have to use the swap space in order to be able to execute at the same time. Furthermore, those large applications might not be able to run on laptop or home computers as their resources are usually smaller than the ones found in an office system. In this paper, they present a solution to both problems that they have implemented in the Linux kernel. The idea consists of compressing the swapped pages and keeping them in a swap cache whenever possible.	compressing	memroy capacity
2	Journal	2005	IWSSPS	**FASS : A Flash-Aware Swap System**	This paper mainly focuses on the swap system running over flash memory. they discuss design and implementation of flash-aware swap system, called FASS, where the kernel manages flash memory-based swap space directly without FTL. This approach can utilize kernel information and flash memory states to optimize the system.	nand aware	nand speed
3	Journal	2008	ICCSA	A New Linux Swap System for Flash Memory Storage Devices	This paper presents a new Linux swap system considering flash memory as a swap storage. Especially they focus on the garbage collection performance and reduce the number of erasures and the number of data copies due to garbage collection.	reduce nand io	nand speed
4	Conf	2009	OLS	**Transcendent Memory (tmem) and Linux**	Transcendent Memory (tmem for short) is a new approach to optimize RAM utilization in a virtual environment. Underutilized RAM from each guest, plus RAM unassigned to any guest (fallow memory), is collected into a central pool. Indirect access to that RAM is then provided by the VMM through a carefully crafted, page-copy-based interface. Linux kernel changes are required but are relatively small and not only provide valuable information to the VMM, but also furnish additional "magic" memory to the kernel, provide performance benefits in the form of reduced I/O, and mitigate some of the issues that arise from ballooning/hotplug.	ballooning/hotplug (tmem pool, pre-cache, preswap, compcache)	memory capacity
5	Conf	2010	ATC	FlashVM: Virtual Memory Management on Flash	With the decreasing price of flash memory, systems will increasingly use solid-state storage for virtual-memory paging rather than disks. FlashVM is a system architecture and a core virtual memory subsystem built in the Linux kernel that uses dedicated flash for paging. FlashVM focuses on three major design goals for memory management on flash: high performance, reduced flash theyar out for improved reliability, and efficient garbage collection. FlashVM modifies the paging system along code paths for allocating, reading and writing back pages to optimize for the performance characteristics of flash.	nand performance	nand cost
6	*Journal*	*2010*	*Korea conf*	*A flash memory swap system for mobile computers*	*They study proposes a new Linux swap system called PASS (process-aware swap system), which allocates the different flash memory blocks to each process. Trace-driven experimental resut show that PASS outperforms existing linux swap system with existing garbage collection schemes in terms of grabage collection cost.*	*process aware*	*nand speed (Korean conf.)*
7	Journal	2010	TCE	**Swap Space Management Technique for Portable Consumer Electronics with NAND Flash Memory**	In order to manage swap space efficiently, this paper presents a novel garbage collection policy for the portable consumer electronics with a swap system. The proposed policy has three features important in NAND flash memory based swap systems: (1) long endurance of NAND flash memory, (2) quick garbage collection, and (3) low energy consumption.	nand aware	nand speed

8	Journal	2010	TCE	An efficient garbage collection for flash memory-based virtual memory systems	This paper proposes a novel garbage collection technique which exploits data redundancy between the main memory and flash memory in flash memory-based virtual memory systems. Compared to the previous approach, our proposed scheme takes into consideration the locality of data to minimize the garbage collection overhead. In addition, by considering the computational overhead of the garbage collection algorithm, we also propose an adaptive scheme which can minimize the computational overhead with marginal I/O performance degradation.	minimize garbage collection of nand	nand speed
14	Conf	2011	FAST	Fast: Quick application launch on solid-state drives	Application launch performance is of great importance to system platform developers and vendors as it greatly affects the degree of users' satisfaction. The single most effective way to improve application launch performance is to replace a hard disk drive (HDD) with a solid state drive (SSD), which has recently become affordable and popular. A natural question is then whether or not to replace the traditional HDD-aware application launchers with a new SSD-aware optimizer.	SSD-aware optimization	SSD is targeted to server
9	Journal	2012	TCE	Compressed Memory Swap for QoS of Virtualized Embedded Systems	This paper presents an in-memory compressed swap device (CSW) for the virtualized consumer electronics environment. It swaps out only the memory of third-party applications in response to memory pressure on the main applications, to ensure its quality of service.	memroy based swap	memory space
10	Journal	2012	TCE	Flash-aware linux swap system for portable consumer electronics	This paper proposes a flash-aware Linux swap system, called FLSS, which adopts Linux kernel 2.6 to manage flash memory-based swap space directly without FTL. We introduce: 1) a partial block alignment scheme to perform an efficient swap-in read-ahead algorithm, 2) a swap-aware victim block selection method and the redefined concept of hot page and cold page to design a swap-aware garbage collection policy called SACATA, and 3) the notion of overage as well as the notion of frozen applied to blocks to exploit a wear leveling-aware block management scheme.	swap-aware garbage collection	nand speed
11	Journal	2012	TCE	Greedy page replacement algorithm for flash-aware swap system	This paper presents a greedy page replacement algorithm, called GDLRU, for flash-aware swap system. In order to reduce the number of flash page write operations, GDLRU introduces a clean-aware victim page selection method called CPS which evicts clean page preferentially. If there is no clean page, CPS evicts the dirty page with the least dirty data preferentially. To further reduce the number of flash page write operations, GDLRU also introduces a clean-aware victim page update scheme called CPU which only writes back the dirty flash pages within the victim dirty page	nand aware	nand speed
12	Conf	2012	MobiSys	Fast app launching for mobile devices using predictive user context	This paper has designed and built FALCON to remedy slow app launch. FALCON uses contexts such as user location and temporal access patterns to predict app launches before they occur. FALCON then provides systems support for effective app-specific prelaunching, which can dramatically reduce perceived delay. FALCON uses novel features derived through extensive data analysis, and a novel cost-benefit learning algorithm that has strong predictive performance and low runtime overhead. Trace-based analysis shows that an average user saves around 6 seconds per app startup time with daily energy cost of no more than 2% battery life, and on average gets content that is only 3 minutes old at launch without needing to wait for content to update	Trace-based analysis, app-specific prelaunching	Does not handle memry contension
13	Conf	2013	SysTor	Linux Block IO: Introducing Multi-queue SSD Access On Multi-core Systems	In this work, we demonstrate that the block layer within the operating system, originally designed to handle thousands of IOPS, has become a bottleneck to overall storage system performance, specially on the high NUMA-factor processors systems that are becoming commonplace. We describe the design of a next generation block layer that is capable of handling tens of millions of IOPS on a multi-core system equipped with a single storage device.	Multi-queue	Not small scale
15	Journal	2014	EMSoft	**Building High-Performance Smartphones via Non-Volatile Memory: The Swap Approach**	They revisit swapping for smartphones with fast, byte-addressable, non-volatile memory (NVM) technologies. Instead of using flash, they build the swap area with NVM, to allow high performance without sacrificing user experience. Based on NVM's high performance and byte-addressability, they show that a copy-on-write swap-in scheme can achieve even better performance by avoiding unnecessary memory copy operations.	reduce mem copy	nvm cost

16	Journal	2014	ISLPED	DR. Swap: Energy-Efficient Paging for Smartphones	They propose DR. Swap, an energy-efficient paging design to reduce energy consumption in smartphones. they adopt emerging energy-efficient non-volatile memory (NVM) and use it as the swap area. Utilizing NVM's byte-addressability, we propose direct read which guarantees zero-copy for readonly pages in the swap area.	energy-efficent paging with nvm	nvm cost
17	Journal	2014	SIGPLAN Notice	VSWAPPER: A Memory Swapper for Virtualized Environments	This paper addresses these problems by implementing VSWAPPER, a guest-agnostic memory swapper for virtual environments that allows efficient, uncooperative overcommitment. With inactive ballooning, VSWAPPER yields up to an order of magnitude performance improvement. Combined with ballooning, VSWAPPER can achieve up to double the performance under changing load conditions.	performance in virt	swap io speed
18	Journal	2014	TCE	Compressed and shared swap to extend available memory in virtualized consumer electronics	This paper proposes a new swap mechanism for virtualized CE devices with flash memory. This proposed mechanism reduces memory consumption by compressing and sharing unused pages. This swap mechanism stores the unused page in memory of another VM, to increase the available memory of the original VM. The proposed swap mechanism is implemented on the Xen hypervisor and Linux.	compressing unused pages	decompressing cost
19	Journal	2014	TCE	**Swap-aware garbage collection algorithm for NAND flash-based consumer electronics**	In order to reduce the energy consumption, this paper proposes a swap-aware garbage collection algorithm for NAND flash-based consumer electronics. The proposed algorithm focuses on reducing the garbage collection overhead and improving the endurance of NAND flash memory.	energy aware garbage colleciton	nand speed
20	Conf	2015	VEE	GPUswap: Enabling Oversubscription of GPU Memory through Transparent Swapping	In this paper, they present GPUswap, a novel approach to enabling oversubscription of GPU memory that does not rely on software scheduling of GPU kernels. GPUswap uses the GPU's ability to access system RAM directly to extend the GPU's own memory. To that end, GPUswap transparently relocates data from the GPU to system RAM in response to memory pressure. GPUswap ensures that all data is permanently accessible to the GPU and thus allows applications to submit commands to the GPU directly at any time, without the need for software scheduling.	GPU memory for system	memory space shortage
21	Conf	2015	HPCC	SwapBench: The Easy Way to Demystify Swapping in Mobile Systems	Mobile systems such as smartphones and tablets are re-adopting swapping—a mature but by rarely used OS feature—to extend memory capacity without adding more DRAM, especially low-end devices.This paper proposes an evaluation framework, SwapBench, to appraise swap schemes and focus on two important but overlooked metrics: application launch and switch. Cross-validation with microbenchmarks shows that SwapBench is accurate. Based on the findings from SwapBench, we further discuss the impacts of different approaches to swapping in mobile systems.	swap evaluation framework	No contribution
22	Conf	2015	ATC	**Memory-Centric Data Storage for Mobile Systems**	To improve both app responsiveness and energy efficiency, this paper proposes MobiFS, a memory-centric design for smartphone data storage. This design no longer exercises cache writeback at short fixed periods or on file synchronization calls. Instead, it incrementally checkpoints app data into flash at appropriate times, as calculated by a set of app/user-adaptive policies. MobiFS also introduces transactions into the cache to guarantee data consistency. This design trades off data staleness for better app responsiveness and energy efficiency, in a quantitative manner.	Flash-centric Incremental checkpointing	Not handle Launching time
23	Conf	2015	MobiSys	Reducing smartphone application delay through read/write isolation	They observe that reads experience up to 626% slowdown when blocked by concurrent writes for certain workloads. Additionally, we show the asymmetry of the slowdown of one I/O type due to another, and elaborate the speedup of concurrent I/Os over serial ones. We use this obtained knowledge to design and implement a system prototype called SmartIO that reduces the application delay by prioritizing reads over writes, and grouping them based on assigned priorities. SmartIO issues I/Os with optimized concurrency parameters.	Prioritize reads over writes, and grouping them	Concurrent wite cases
24	Conf	2016	ATC	An Evolutionary Study of Linux Memory Management for Fun and Profit	they present a comprehensive and quantitative study on the development of the Linux memory manager. The study examines 4587 committed patches over the last five years (2009-2015) since Linux version 2.6.32. Insights derived from this study concern the development process of the virtual memory system, including its patch distribution and patterns, and techniques for memory optimizations and semantics. Specifically, they find that the changes to memory manager are highly centralized around the key functionalities, such as memory allocator, page fault handler and memory resource controller.	memory manager	memory space shortage
25	Conf	2016	ccgrid	CloudSwap: A Cloud-Assisted Swap Mechanism for Mobile Devices	They propose CloudSwap, a fast and robust swap mechanism for mobile devices to enable the memory noblivious application caching. The key idea of CloudSwap is to use the fast local storage as a cache of read-intensive swap pages, while storing prefetch-enabled, write-intensive swap pages in a cloud storage. To preserve the lifespan of the local storage, CloudSwap minimizes the number of writes to the local storage by storing the modified portions of the locally swapped pages in a cloud.	fast local storage for read-intensive swap pages	cloud swap's speed and security

26	Journal	2016	EMSoft	The Design of an Efficient Swap Mechanism for Hybrid DRAM-NVM Systems	This paper analyzes the data accesses features of different applications . Then, a swap mechanism, called Refinery Swap, is proposed to improve the performance of the system, reduce energy consumption, and increase the lifetime of NVM simultaneously. Refinery Swap presented two algorithms to exploit the data access features of applications and the characteristics of different kinds of memory medias.	reduce mem copy	nvm cost
27	Journal	2016	TCE	**In-Memory File System with Efficient Swap Support for Mobile Smart Devices**	This paper propsoes a new in-memory file system is devised to improve the current swap mechanism in the legacy in-memory file system, whereby strip-based layouts with separated file-swap partitions are used and unnecessary file-I/O overheads are eliminated. To evaluate the proposed in-memory file system, experiments with the file-I/O benchmark were conducted. The experiment results show that, compared to the current swap scheme, the proposed swap scheme improves the overall performance by ten times on average.	in-memory swap	memory space
28	COnf	2016	MicroCom	Optimize In-Kernel Swap Memory By Avoiding Duplicate Swap Out Pages (4 pages)	In-kernel memory swapping is a Linux feature which creates RAM based swap area and provides a form of virtual memory compression. It increases performance by using a compressed block device in RAM for paging instead of disk. This paper proposes a new mechanism to avoid duplicate compressed pages swapped to the In-RAM swap area. Experiments have assured increase in the available physical memory and there by improved the performance of applications even at low memory conditions	swap compression	ram limitation
29	Others	2016	lwn.net	Making swapping scalable https://lwn.net/Articles/704478/	If swapping can be made fast enough, the performance penalty for overcommitting memory becomes insignificant, leading to better utilization of the system as a whole.	Skip read ahead for unreferenced swap slots, Add cache for swap slots allocation	Focus on cloud environment
30	Journal	2016	TECS	SmartLMK: A memory reclamation scheme for improving user-perceived app launch time	This paper proposes a novel memory reclamation scheme called SmartLMK. SmartLMK minimizes the impact of the process-level reclamation on user experience. The worthiness to keep an app in memory is modeled by means of user-perceived app launch time and app usage statistics. The memory footprint and impending memory demand are estimated from the history of the memory usage. Using these values and memory models, SmartLMK picks up the least valuable apps and terminates them at once	app launch time and app usage statiscis analysis	additional cost to collect app usage statistics
31	Conf	2017	FAST	FlashBlox: Achieving Both Performance Isolation and Uniform Lifetime for Virtualized SSDs	they propose utilizing flash parallelism to improve isolation bettheyen virtual SSDs by running them on dedicated channels and dies. Furthermore, they offer a complete solution by also managing the theyar. they propose allowing the theyar of different channels and dies to diverge at fine time granularities in favor of isolation and adjusting that imbalance at a coarse time granularity in a principled manner.	parallelism	ssd speed
32	Conf	2017	NSDI	Efficient Memory Disaggregation with Infiniswap	This paper describes the design and implementation of INFINISWAP, a remote memory paging system designed specifically for an RDMA network. INFINISWAP opportunistically harvests and transparently exposes unused memory to unmodified applications by dividing the swap space of each machine into many slabs and distributing them across many machines' remote memory. Because one-sided RDMA operations bypass remote CPUs, INFINISWAP leverages the pothevr of	remote paging system	swap speed
33	Journal	2017	IEEE ACCESS	Maintaining Application Context of Smartphones by Selectively Supporting Swap and Kill	This article proposes a selective swap scheme that classifies applications based on their context-saving characteristics, and controls the number of processes involved in swap by monitoring system situations and application characteristics.	selective swap& kill	monitoring cost
34	Journal	2017	ISCE	Performance study and analysis of D-SWAP for mobile communication networks	In this paper, a mathematical analysis model is developed to demonstrate the performance of the newly proposed D-SWAP power-saving mechanism. The analyzed performance focuses on the average power consumption and average packet delay. The analysis and simulation results demonstrate that the proposed D-SWAP can concurrently satisfy the trade-off requirements of low average power consumption and low average packet delay.	power saving	swap speed
35	Conf	2017	DAC	**SmartSwap: High-Performance and User Experience Friendly Swapping in Mobile Systems (6 pages)**	Most mobile systems have limited memory space, which in turn affects user satisfaction. For example, application response time could become longer due to limited memory capacity. Swapping is an effective way to extend memory capacity, but often lead to poor performance in smartphones. This paper proposes predictive process-level swapping that predicts the most rarely used (MRU) applications and dynamically swaps processes ahead-of-time. Trace-based evaluations show up to 30% increase in application launch performance compared to the worst case brought by flash-based swap.	predictive process-level swap technique (Most rarely used (MRU) and Ahead-of-time swap)	swap speed, nand lifespan

36	Journal	2018	TECS	**Application-Aware Swapping for Mobile Systems**	This paper proposes a novel scheme to properly harness the swapping to mobile systems. They identify that a vast amount of I/O for swapping comes from the conflict of the traditional page-level approach of the swapping and the process-level memory management scheme tailored to mobile systems. Moreover, they find out that the current victim page selection policy is not effective due to the process-level policy.	app aware approach (김진수 교수랩)	swap speed
37	Others	2018	lwn.net	The final step for huge-page swapping https://lwn.net/Articles/758677/	The use of huge pages can improve the performance of the system significantly. This patch is merged in 4.14, further delayed the splitting until the huge page had actually been written to the swap device, again improving performance through better batching and by writing the entire huge page as a unit.	Improve P/F with huge pages	The splitting of huge pages
38	Journal	2018	IEEE ACCESS	A Hybrid Swapping Scheme Based On Per-Process Reclaim for Performance Improvement of Android Smartphones	This paper proposes a hybrid swapping scheme based on per-process reclaim that supports both secondary-storage swapping and zRAM swapping. It attempts to swap out all the pages in the working set of a process to a zRAM swap space rather than killing the process selected by a low-memory killer, and to swap out the least recently used pages into a secondary storage swap space. The main reason being is that frequently swapin/out pages use the zRAM swap space while less frequently swap-in/out pages use the secondary storage swap space, in order to reduce the page operation cost. Our scheme resolves both the response time and wear-out problems of secondary-storage swapping and zSWAP, and overcomes the size limitation of the zRAM swap	zRAM/zSWAP with per-process	SWAP cost (io scheduling, battery consumption) SWAP in/out cost
39	Conf	2018	ATC	**FastTrack: Foreground App-Aware I/O Management for Improving User Experience of Android Smartphones**	This paper proposes a foreground app-aware I/O management scheme, called FastTrack, that accelerates foreground I/O requests by 1) preempting background I/O requests in the entire I/O stacks including the storage device and 2) preventing foreground app's data from being flushed from the page cache.	foreground app-aware I/O management	emulating the storage device for evaluation
40	Journal	2019	IEEE ACCESS	ezswap: Enhanced Compressed Swap Scheme for Mobile Devices	To overcome the aforementioned problems and maximize the memory efficiency, they propose a compressed swap scheme, called enhanced zswap (ezswap), for mobile devices. ezswap accommodates not only anonymous pages, but also clean file-mapped pages.	dram-based swap Clean file-mapped-page Beneficial compression ratios	dram space
41	Conf	2020	ATC	**End the Senseless Killing: Improving Memory Management for Mobile Operating Systems (Marvin)**	This paper presents Marvin, a new memory manager for mobile platforms that efficiently supports swapping while meeting the strict performance requirements of mobile apps. Marvin's swap-enabled language runtime is co-designed with OS-level memory management to avoid common pitfalls of traditional swap mechanisms. Its key features are: (1) a new swap mechanism, called ahead-of-time (AOT) swap, which pre-writes memory to disk, then harvests it quickly when needed, (2) a modified bookmarking garbage collector that avoids swapping in unused memory, and (3) an object-granularity working set estimator.	AOT SWAP, Bookmarking, W/S Estimator	swap speed
42	Conf	2020	ATC	Effectively Prefetching Remote Memory with Leap	In this paper, they propose Leap, a prefetching solution for remote memory accesses due to memory disaggregation. At its core, Leap employs an online, majority-based prefetching algorithm, which increases the page cache hit rate. they complement it with a lighttheyight and efficient data path in the kernel that isolates each application's data path to the disaggregated memory and mitigates latency bottlenecks arising from legacy throughput-optimizing operations.	prefetching	memory space shortage
43	Conf	2020	SAC	An Efficient Garbage Collection in Java Virtual Machine via Swap I/O Optimization	Various applications, frameworks, and services are built on Java Virtual Machine (JVM) (e.g., Big data analytics) due to its crossplatform portability. Hotheyver, many of them suffer from long latency of Garbage Collection (GC) which also drops throughput, efficiency, and availability of the system. In this paper, they present a performance analysis of the existing GC policy in JVM. Based on the result of analysis, they propose an efficient GC scheme to improve the GC performance via swap I/O optimization to complement the existing GC policy. In this scheme, they selectively compact JVM heap by interacting with OS swap system during GC.	efficient GC scheme	swap speed

44	Conf	2020	ICCE	Enhanced Flash Swap: Using NAND Flash As a Swap Device with Lifetime Control	This paper proposes a new solution, Enhanced Flash Swap (EFS). EFS uses NAND flash as a swap device with a careful control of its lifetime. This paper introduces several techniques to address the issue of lifetime on flash-based swap, including deduplication, compression, buffering and physical block management.	nand aware	nand speed
45	Journal	2020	IEEE ACCESS	Analysis of Smartphone I/O Characteristics Toward Efficient Swap in a Smartphone	They present a new architecture that adopts non-volatile memory at the Android swap layer. Specially, as Android swap has bimodal data access characteristics, they identify and manage hot and cold data ef ciently by making use of precise admission control and replacement algorithms.	specify hot & cold data	nvm capacity (price)
46	Conf	2020	PACT	DeepSwapper: A Deep Learning Based Page Swap Management Scheme for Hybrid Memory Systems (2 pages, Poster)	This paper introduces DeepSwapper, a deep learning-based page swap management scheme that utilizes RNN to perform fast, energy-efficient, and temperature-aware page swapping in hybrid memory systems. DeepSwapper comprises of LSTM units of RNN model to predict the future memory accesses to guide its swap management scheme, a dynamic page swap management scheme that utilizes DRAM capacity efficiently by enabling hot pages in a swap group to be swapped with cold pages of another swap group, and a temperature-aware page swap management scheme, which first predicts the future writes to NVM pages and then, decides to migrate those pages with frequent writes in hot NVM banks to DRAM to enhance the NVM lifetime	Dynamic swap for RNN/LSTM, Temperature-aware	swap speed
47	Conf	2020	ATC	**Acclaim: Adaptive Memory Reclaim to Improve User Experience in Android Systems**	This paper shows that the current memory reclaim scheme cannot deliver its desired performance due to two key reasons: page re-fault, which occurs when an evicted page is demanded again soon after, and direct reclaim, which occurs when the system needs to free up pages upon request time. Unlike the server workload where the direct reclaim happens infrequently, multiple direct reclaims can happen in many common Android use cases. We provide further analysis that identifies the major sources of the high number of page re-faults and direct reclaims and propose Acclaim, a foreground aware and size-sensitive reclaim scheme. Acclaim consists of two parts: foreground aware eviction (FAE) and lightweight prediction-based reclaim scheme (LWP). FAE is used to relocate free pages from background applications to foreground applications. While LWP dynamically tunes the size and the amount of background reclaims according to the predicted allocation workloads.	Foreground-aware eviction, Lightweight prediction-based reclaim scheme (mobile-aware page reclaim)	memory capacity
48	Conf	2021	ATC	**ASAP: Fast Mobile Application Switch via Adaptive Prepaging**	This paper identifies conventional demand paging as the primary source of this inefficiency and proposes ASAP, a mechanism for fast application switch via adaptive prepaging on mobile devices. ASAP performs prepaging by combining i) high-precision switch footprint estimators for both file-backed and anonymous pages, and ii) efficient implementation of the preparing mechanism to minimize resource waste for CPU cycles and disk bandwidth during an application switch. Our evaluation using eight real-world applications on Google Pixel 4 and Pixel 3a demonstrates that ASAP can reduce the switch time by 22.2% and 28.3% on average, respectively (with a maximum of 33.3% and 35.7%, respectively), over the vanilla android 10.	Propose adaptive prepaging for mobile device. reducing latency of application switch to improve UX of smartphones	Predicting is difficult to do prepaging
49	Conf	2017	ATC	Improving user perceived page load times using gaze	We take a fresh look at Web page load performance from the point of view of user experience. Our user study shows that perceptual performance, defined as user-perceived page load time (uPLT) poorly correlates with traditional page load time (PLT) metrics. However, most page load optimizations are designed to improve the traditional PLT metrics, rendering their impact on user experience uncertain. Instead, we present WebGaze, a system that specifically optimizes for the uPLT metric.	user-perceived page loading	web-centric evaluation
50	Conf	2019	ATC	Asynchronous I/O stack: A low-latency kernel I/O stack for ultra-low latency SSDs	Today's ultra-low latency SSDs can deliver an I/O latency of sub-ten microseconds. With this dramatically shrunken device time, operations inside the kernel I/O stack, which were traditionally considered lightweight, are no longer a negligible portion. This motivates us to reexamine the storage I/O stack design and propose an asynchronous I/O stack (AIOS), where synchronous operations in the I/O path are replaced by asynchronous ones to overlap I/O-related CPU operations with device I/O.	Asynchronous I/O stack	nand speed