System Technique: Backpointers

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

LFS Bradley, Akila, Sudarshan

Uses backpointers to keep track of live blocks throughout the file system. The imap stores the location of the data block. +

Wisckey Sudarsan, Yash, Shunmiao

Uses backpointers from value log to key for garbage collection

AFS Server has pointers to traverse directory structure. Local system stores callbacks ~~mode values~~, ~~pointers~~ locally to be able to quickly traverse the tree and improve performance by reducing server ~~to~~ load.

(kaushik. C, Peter)

1

System Technique: Background Work

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

+ In wisckey log, two pointers head and tail are maintained. Wisckey validates which of the key-value pair is valid by refering the key tree and as part of garbage collection moves it to head of the log. This process is performed in background.

Advantage: ① Does not interfer with foreground.

② Can be fine tuned with respect to foreground, so that it foreground requries more resources, background

'+' ⇒ use of version num in inode (lfs) can be limited to limit garbage collection overhead.

VENKATESH + ++ Checkpointing in journalling fs. } Akila, Sudarshan, Bradley.

+ ++ ADN in OptFS.

SSD Shunmiao, Sudarsan, Yash

Garbage collection of SSD aims to

1) pre-erase the blocks that are dangling to improve the write performance

2) compaction of data blocks in the background to reduce write amplification

3) wear-leveling

LFS garbage collection

System Technique: Layering

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

Coda's Venus presents the same interface when disconnected
Tim B. from the file servers. This simplifies app development, because
KanWu developers don't need to worry about which mode Coda is
✓ currently in. Similarly, users of the file system can
treat is as if it were a local FS.

Abhinav Floshgroph → Flashgroph used SAFS to cache the data and to
overlap I/O with computation. SAFS buffers edge lists from
+ SSD, so that Flashgroph can adapt to different cache hit rates.

RAID → Depending on the type of RAID used, the lowest
layer can have 'n' disks. But the top layer
Sudarshan, (File system) finds it as one logical disk. So
Akila, there is clear separation of layers, where FS is
Bradley. built on top of RAID.

System Technique: Indirection

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

System: LFS (Peter ε, Kaushik. C) + +

- Fixed checkpoint region contains i-map pointers
- i-map points to inodes.
- Allowed sequential writes to i-map instead of in-place updates which would have resulted in random writes and reduced performance.

System: Wisckey Tim B. Kan. KARTHIK

- Wisckey seperately stores key in LSM-tree while value in logs. However to ease the Garbage Collection, it also stores the corresponding key along with the value in the log.
- Thus, it avoids scanning of the whole LSM tree.
- By seperating keys & values it reduces the read & write amplification,

Previous

AFS - fid

SSDs - FTLs - indirection

FFS - inodes - file names

System Technique: Checksums/Parity opt+

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

→ FS Review group 6

+ Optimistic crash calculate checksum using both metadata and data.

By computing a checksum over the entire transaction and placing its value in J_c, the writes to J_M and J_c can be issued together, improving performance. Crash recovery can avoid replay of improperly committed transactions. → slightly unclear

FS - Review 8

Minh, Nikhita, Bidgit

RAID ≥ 4, 5 (Rotating Parity)

RAID 4 - single parity disk

RAID 5 - Rotating parity

+ RAID 4 uses parity to provide Relaibility, good sequential Performance

+ RAID 5 uses parity and parallelism to give better Seq and Random throughput

→ Group 9 [Karan, Brandon, Surya]

* Performance in terms of CPU usage goes down. As extra compute is required to compute checksums.

WiscKey → Can benefit from addition of Checksums. Possible because key and values are at different places. Checksum can bring them to a consistent state

Key, ptr, checksum over data

Disks: Checksums

System Technique: Optimism

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

YASH Sudarsan Shunmiao ✓ 1) OptFS ⇒ optimistic journaling ⇒ separate ordering & durability; sacrifice durability for performance. Benefit: higher performance.

Coda + 2) assume that files won't be overwritten by multiple users; write conflicts are unlikely. This allows coda to implement reliability for disconnected systems.
(- Peter ε, kaushik. c)

Hmm. when conflicts do happen, you need to handle that

System Technique: Sorting

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

(Aarati Kakaraparthy) Zhewen Song Kevin Houck

Graph Chi: We have a list of ordered vertices, which we partition. Corresponding to each partition, we have a shard of in-edges. In each shard, we sort the edges by source. The benefit of doing this is, when out-edges corresponding to a shard are accessed in other shards, they are sequential. + +

~~[redacted]~~ LFS

+ (If there are workloads which write random block and then read sequential reads then Log structured file system does n't give performance.) (For these kind of workloads we can sort the blocks before writing.) -> LFS does not do this, not sure how this would work

Pradeep Sang Leon

Fs Review 8

Wisc Key

-> uses sorting of keys in the LSM tree

+ The technique of sorting keys is essential for fast searching and by separating keys and values and storing keys in-memory it is optimized for SSDs

System Technique: More work done more efficiently

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

1. In FlashGraph, the authors were able do more work by focusing on vertex rather than edges. + good balancing

They used a thread based model; partitioned across vertices. As soon as a thread was done, it ~~stealed~~ stole others work.

The key here is to load balance in an efficient way, by being flexible to re-assign partitions to different workers. [Surya, Karan, Brandon] '-' isn't clear. '-' (they do not do "more" work)

2. Wiskey: Uses prefetching for range queries. (Euclita, Pulkit) It leverages parallel I/O characteristics of SSDs to prefetch values from the vLog concurrently based on the access pattern of the range query. (∵ values are not sorted) '+'

3. X-Stream: It's edge-centric. It scans the edge list for multiple times (Zhewen Song) sequentially. It reads more than required, but this work is sequential, which is more efficient than the random access in SSDs or HDDs. way

System Technique: Overlap/Parallelism

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

+ Flash Graph: Group8: Minh; Nikhita; Bidyut

+ Overlapping: FG hides latency by overlapping computation with I/O

(x) + Parallelism: FG initiates many parallel I/Os and process data when its ready

+ Why it's useful: + get maximum performance out of SSD.

+ ⇒ Out-of-memory graph computation faster on SSD

RAID: Grp 9 [Brendan, Surya, Kaiwan]

(+) Parallelism RAIDs allow us to read multiple blocks of data at the same time across multiple disks. ~~Data is~~ It ~~use~~ takes advantage of having multiple disks. It speeds up read rates and improves crash consistency.

SSD – Unwritten Contract:

Use of NCQ depth and channels to serve requests in parallel

01 WiscKey

System Technique: Caching

1. Read the notes written by the previous groups for this technique. Please rate their response with a "+", a checkmark, or a "-"; if you see anything you disagree with or think is wrong or unclear, please politely correct.
2. For the System Technique listed above, think about each of the papers/systems you have read in CS 736. Pick one **system** that uses that technique in an interesting or important way (that has NOT been already described on this page).
   - In a few sentences, describe **how** your chosen system uses the technique.
   - Explain **why** the technique is **useful** in that system (i.e., the benefits of the technique).

If you are unable to think of a different system that uses this technique, you can instead (in order of preference):

- Describe how that technique could be **added** in a useful way to one of the systems
- Elaborate on the answer given previously by a different group; for example, what new **challenges** does this technique introduce that must now be solved?
- Pick a different technique you haven't considered yet (indirection, layering, checksums, caching, overlap/parallelism, background work, more work done more efficiently, optimism, or sorting), denote the technique, and answer this question for that technique.

Please alternate who in your group takes notes and write your first names by your notes.

1. AFS uses whole file caching on local disks of client workstations. (+)
Advantages: (i) Reduces network traffic and server load (ii) client only contacts server on open & close (iii) allows use of bulk data transfer protocols (iv) reduces refetch of data after reboots (v) simplifies cache manggment
{ Suchita, Pulkit }

2. X-stream: Use caching of vertices in CPU cache and edges in ram, or vertices in ram depending on graph size. This improves processing time by speeding up random accesses by using the faster memory tier. +
{ Kevin Houch }

3. Wisckey: The original LSM has significant read amplification, which make RAM as a cache less useful. Wisckey only has ~~ke~~ keys in the tree, which is small enough to store in RAM. With this caching, random access to keys ~~a~~ is much faster.
{ Leon Song Pradeep }

- Coda
- SSDs (Onwritten) - FTL
- LFS - imap