Flashy Prefetching for High-Performance Flash Drives

Ahsen J. Uppal, Ron C. Chiang, H. Howie Huang

Department of Electrical and Computer Engineering

The George Washington University

MSST '12 - April 19, 2012

Introduction

- SSDs are becoming very popular due to high bandwidth and low latency
- Scientific and enterprise data requirements continue to grow
- Traditional data prefetching, if inappropriately controlled, is likely interfere with normal I/O requests and result in lower application performance.
- Our approach is different...

Flashy Prefetching

- Consists of:
  - Accurate prediction of application needs in runtime
  - Adaptive feedback-directed prefetching that scales with application needs
  - Being considerate to underlying storage devices

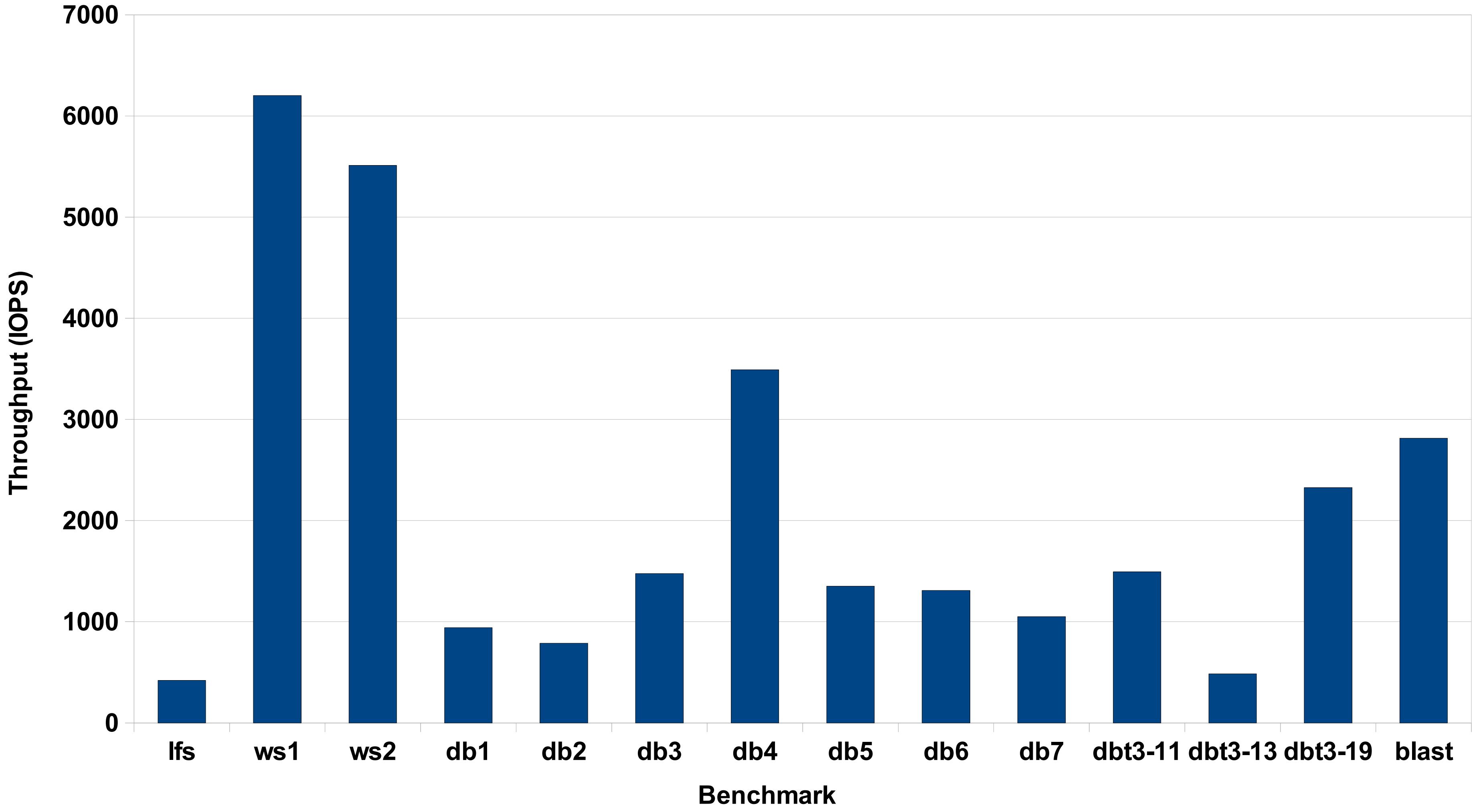
Issues with Traditional Prefetching

- Focused on hard drive and conservative with the amount prefetched
- If too aggressive:
  - it can take shared I/O bandwidth from application data accesses.
  - useful cached data may become evicted while main memory would be filled with mispredicted (and unneeded) data
- Not tuned for the storage device and apps, some devices can support higher prefetch rates and app needs vary.

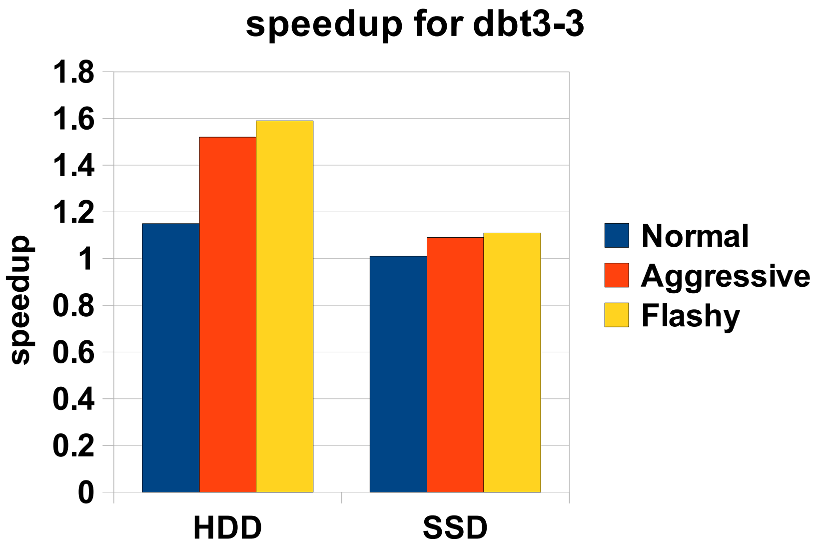
Challange 1: SSDs Differ

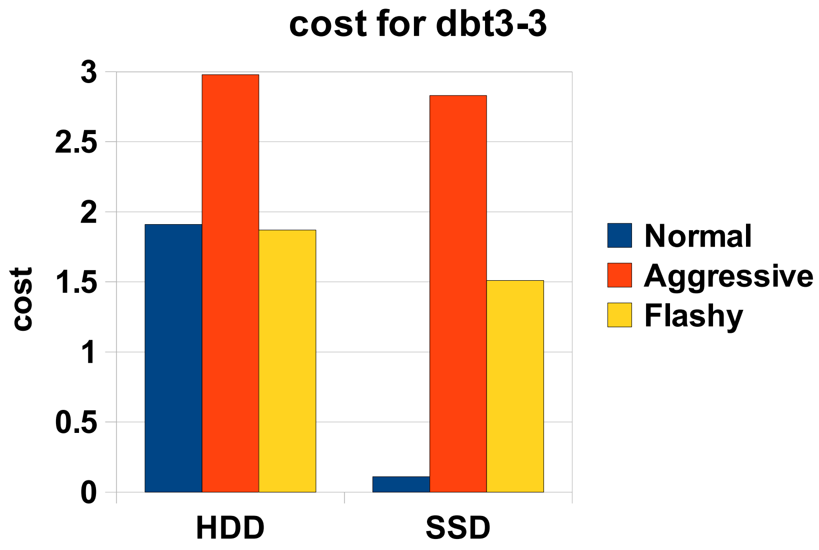
	SSD1	SSD2	SSD3	SSD4
Vendor	OCZ Vertex	OCZ Vertex 2	Intel X-25M	Intel 510
Capacity (GB)	120	120	80	120
Flash Type	MLC	MLC (34nm)	MLC (34nm)	MLC (34nm)
Controller	Indilinx	SandForce	Intel	Marvell
Read BW (MB/s)	250 (max)	285 (max)	250 (seq)	450 (seq)
Write BW (MB/s)	180 (max)	275 (max)	100 (seq)	210 (seq)
Latency (us)	100	100	65	65
Measured Read BW (MB/s)	170	170	265	215
Measured Write BW (MB/s)	180	203	81	212

Challange 2: Applications Differ

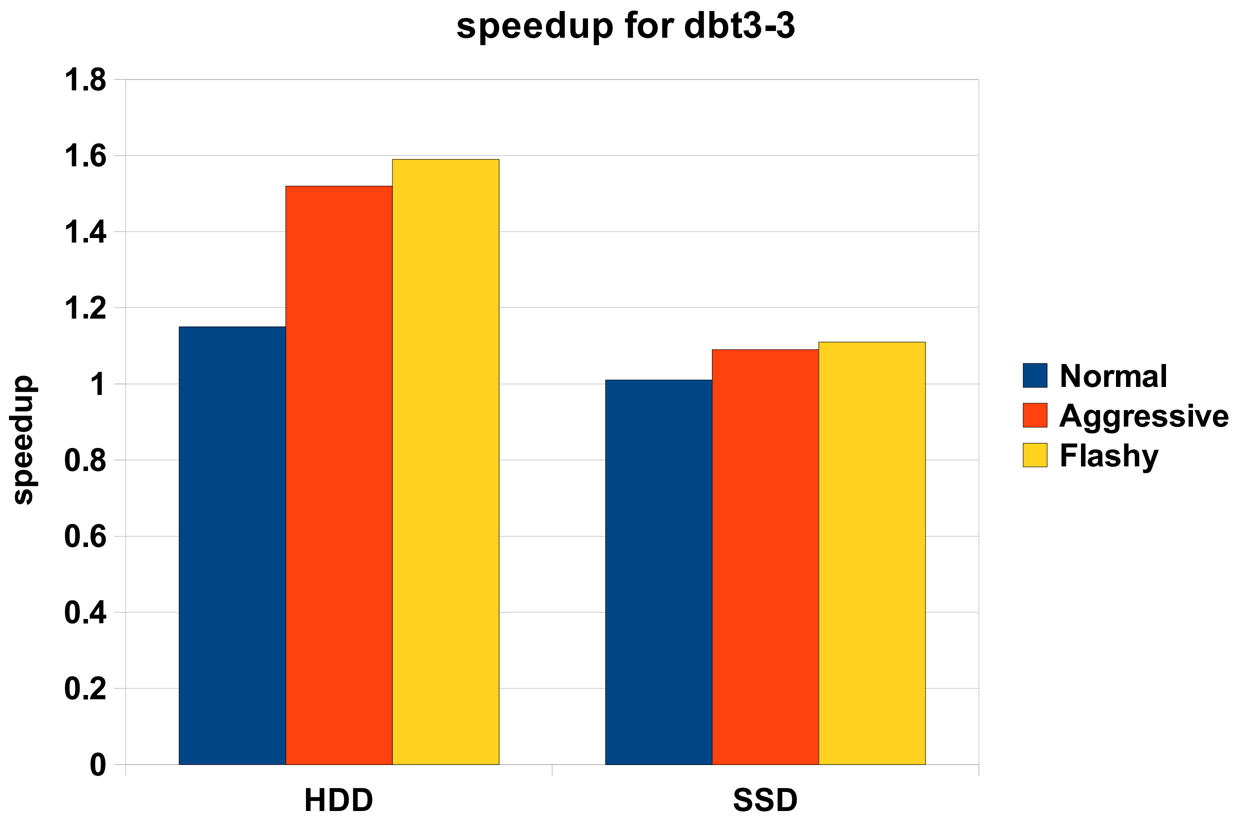


Challenge 3: Prefetching for HDDs and SSDs Differs





Challenge 3: Prefetching for HDDs and SSDs Differs



Flashy Prefetching

- Address the three challanges:
  - Control prefetching based on drive performance. Prefetch based on an app's measured rate up to a max per disk.
  - Control prefetching based on prefetching performance. Increase aggressiveness scale factor when a benefit is observed and decrease when there is no benefit.
  - Enable prefetching for multiple simultaneous accesses to take advantage of parallel access on SSDs. Must be aware of application context (process id, device id, block id).

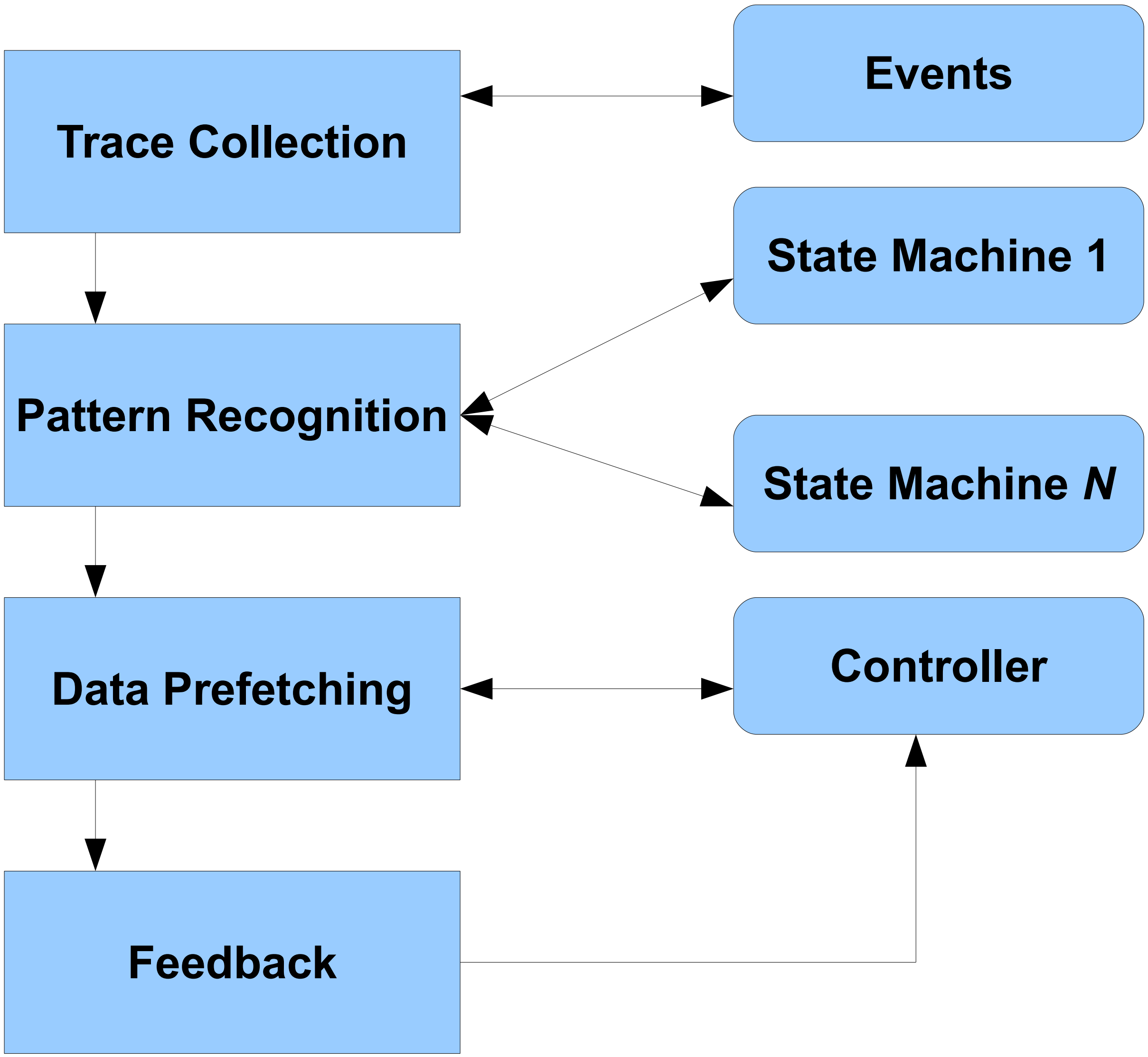
Prototype

- We have implemented a real system, called *prefetchd*, in Linux and evaluated it on four different SSDs.
- The results show 65-70% prefetching accuracy and an average 20% speedup on LFS, web search engine traces, BLAST, and TPC-H like benchmarks across various storage drives.

prefetchd

- The prototype for Linux systems:
  - Monitors application read requests
  - Predicts which pages are likely to be read in the near future
  - Loads those pages into the system page cache while attempting to not evict other useful pages
  - Monitors its success rate
  - Adjusts its aggressiveness accordingly
  - Does this for multiple simultaneous applications

Architecture



Trace Collection

- Trace collection (blktrace in Linux) accumulates disk events.
- Periodically, collected events are fed into Pattern Recognition which keeps counters (per context) for sequential, strided, and reverse accesses.
- If counts exceed a threshold, Data Prefetching issues prefetch requests (readahead in Linux) based on an aggressiveness scale factor up to a max per device.

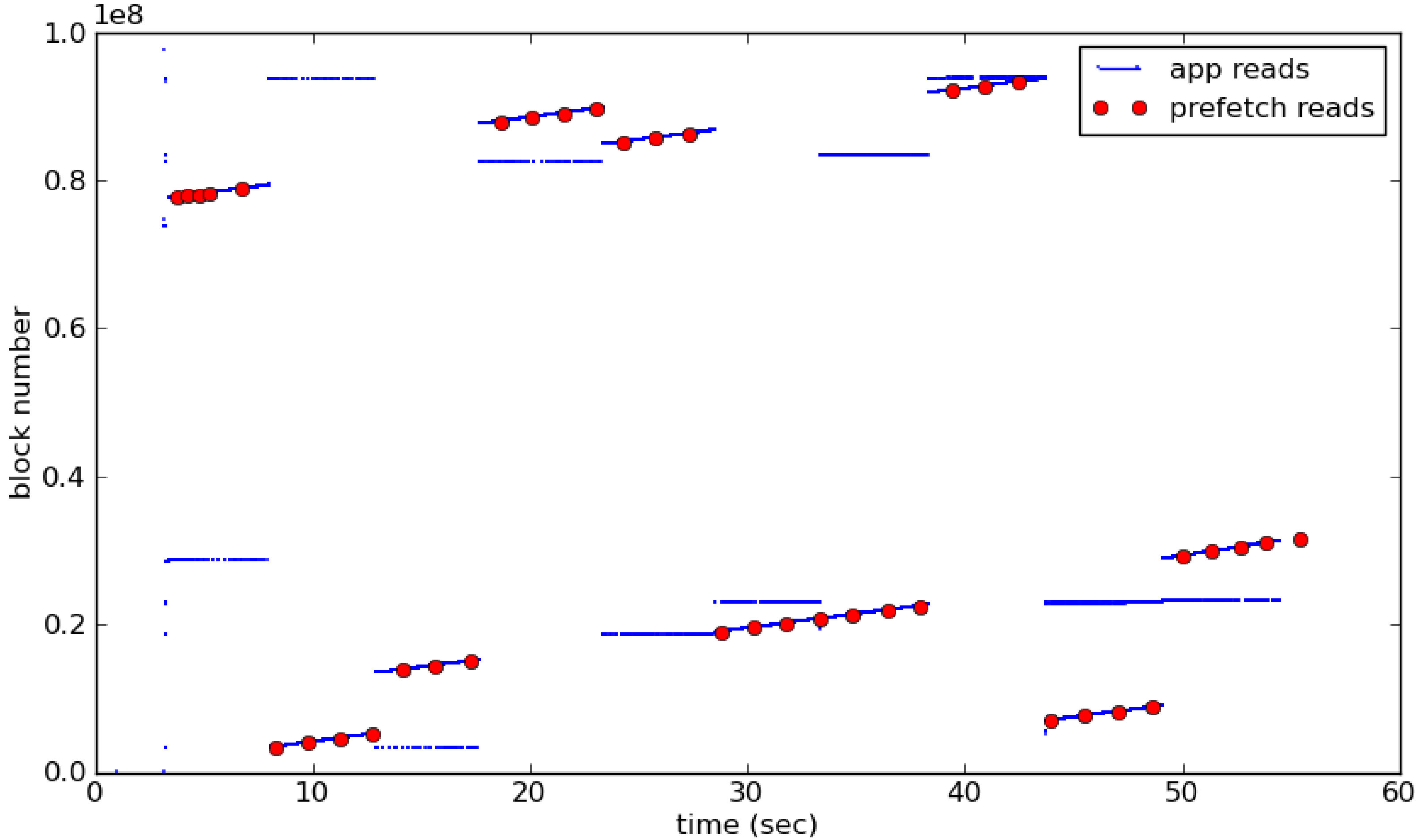
Feedback Monitoring

- Feedback adjusts the scale factor by comparing old prefetch requests to reads reaching disk.
  - If there are any linear, easily predictable reads that were not prefetched, and still reached disk, then the prefetching aggressiveness should be increased.
  - If there are no linear reads reaching the disk and the statistics show that the prefetching amount is more than what the applications are requesting, decrease the aggressiveness.

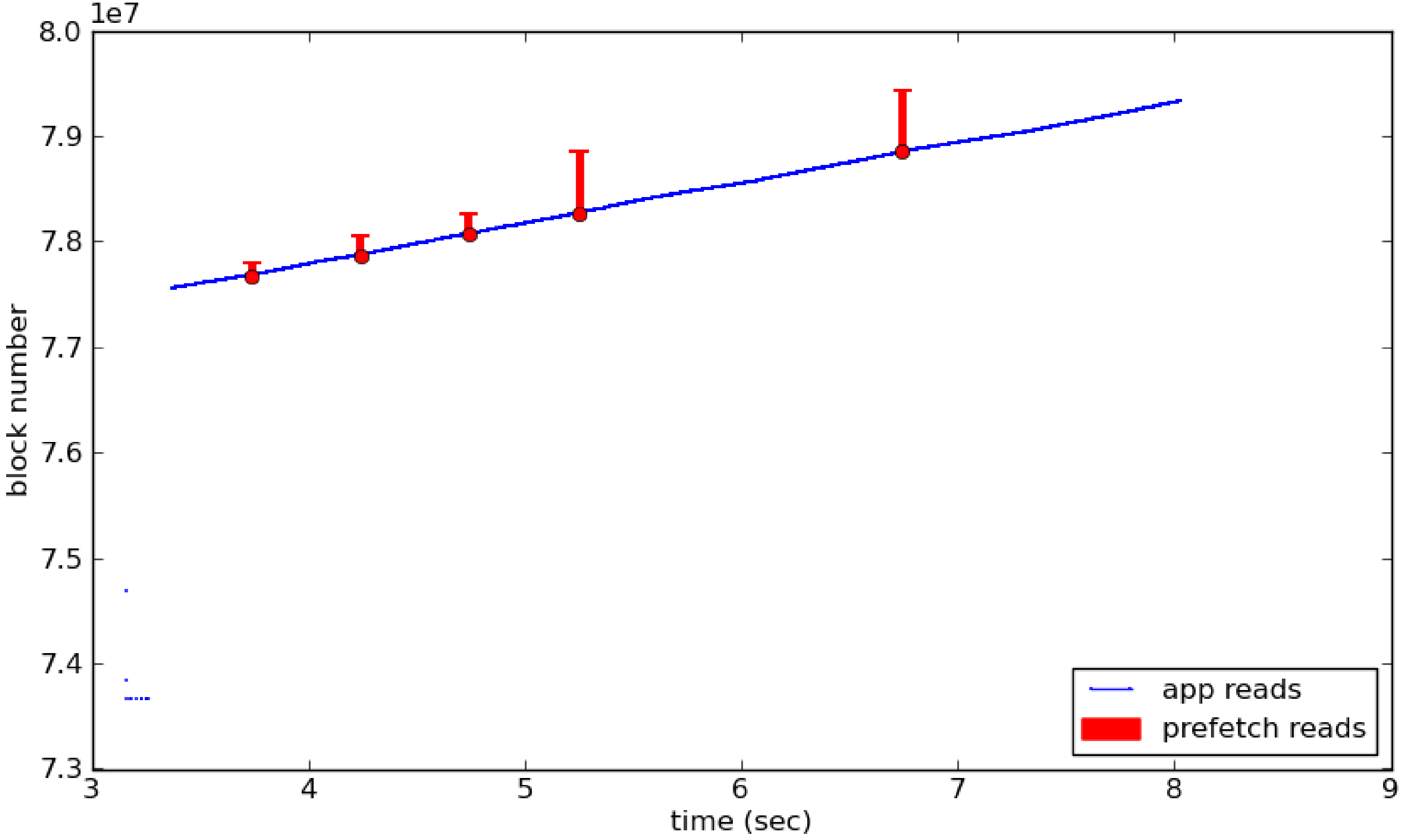
Evaluation Setup

- Evaluated several benchmarks: DBT3 (postgres queries), BLAST (bio), LFS (large file I/O), PostMark (email), WebSearch (traces)
- Runs on Linux kernel 2.6.28 on an Intel Core2 Quad CPU at 2.33 GHz and 8 GB RAM.
- We tested four SSDs and one hard drive
  - Used loopback device on top of a regular fs to use the faster VM cache instead of buffer cache

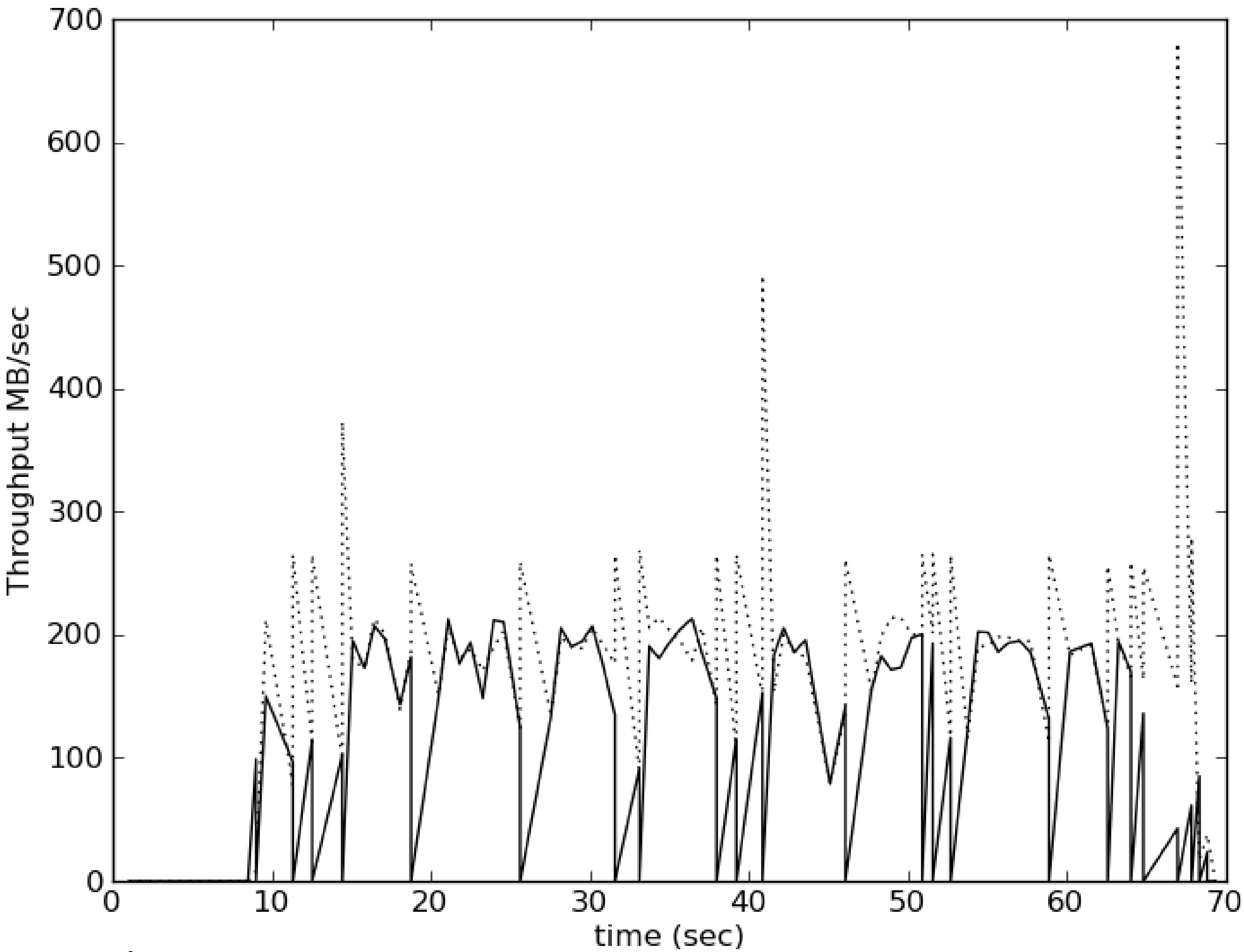
BLAST Trace Zoomed Out



BLAST Trace Zoomed In

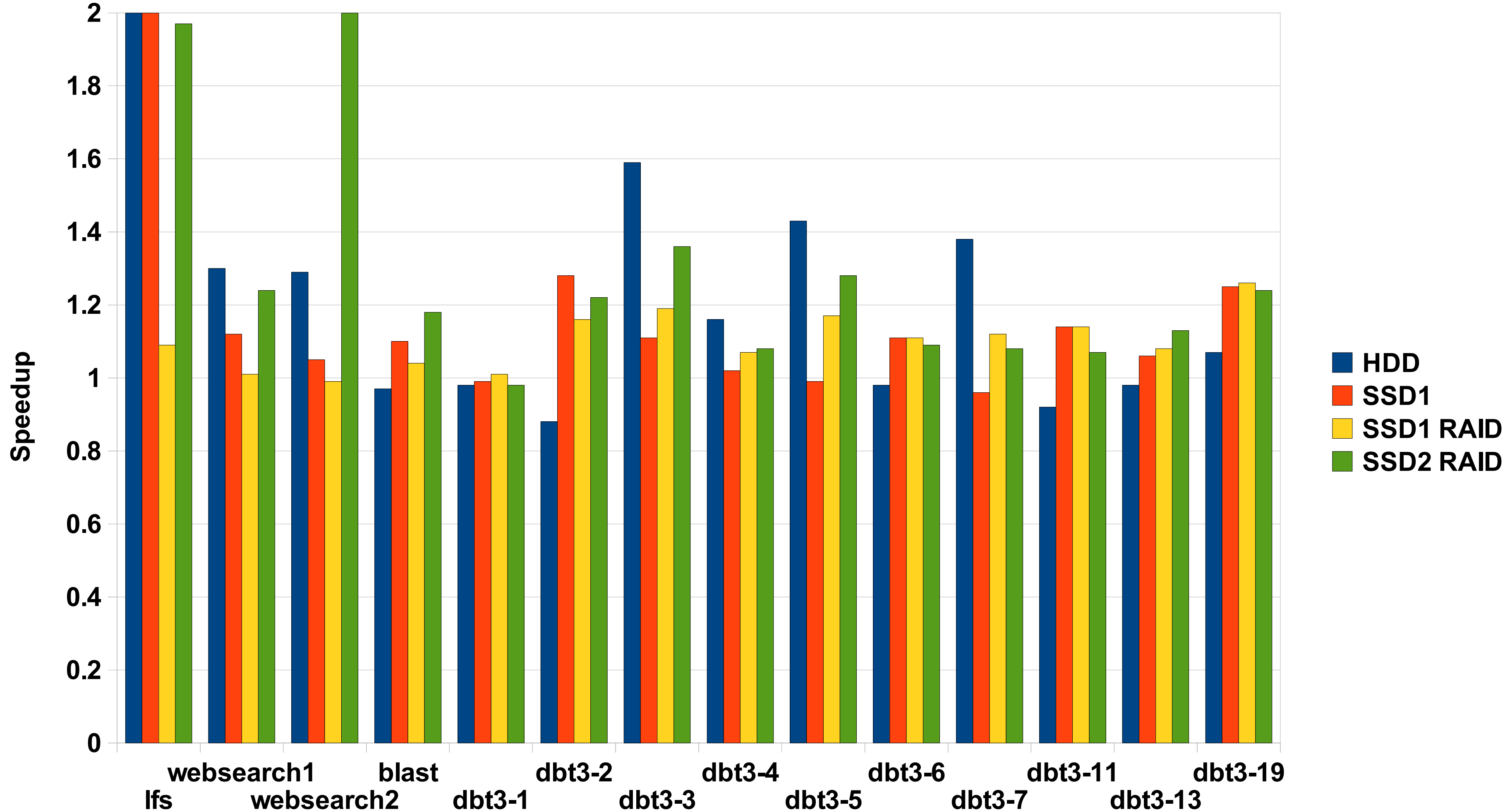


BLAST Aggressiveness Over Time



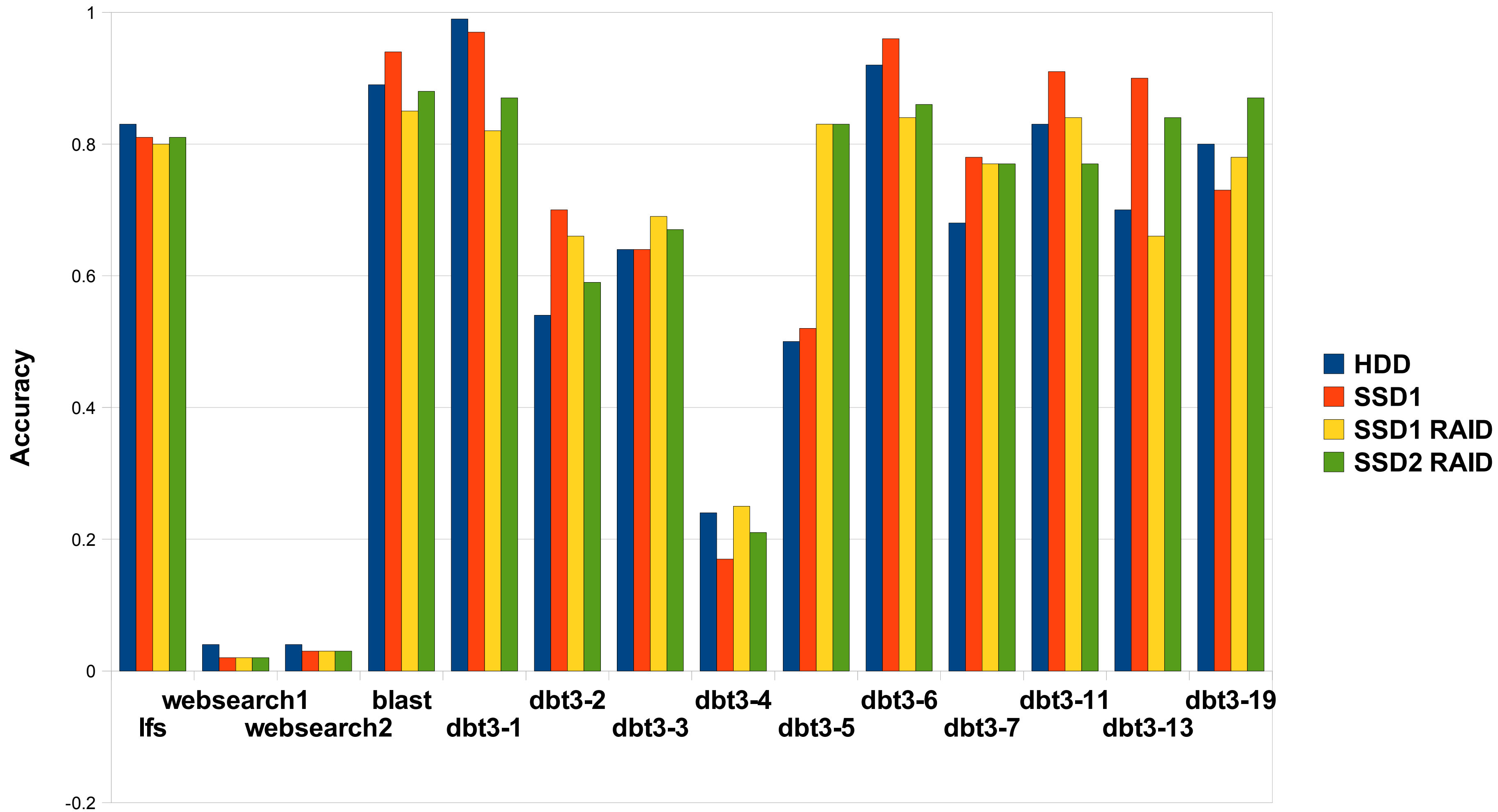
- Solid is app read rate
- Dashed is prefetchd read rate

Prefetching Performance



- Values above 2.0 are omitted (including for websearch2 at 2.02) to more clearly show variation between the other benchmarks.
- The speedup for SSDs tends to be lower than for the HDD because the potential speedup on an already-fast device is limited.

Prefetching Accuracy

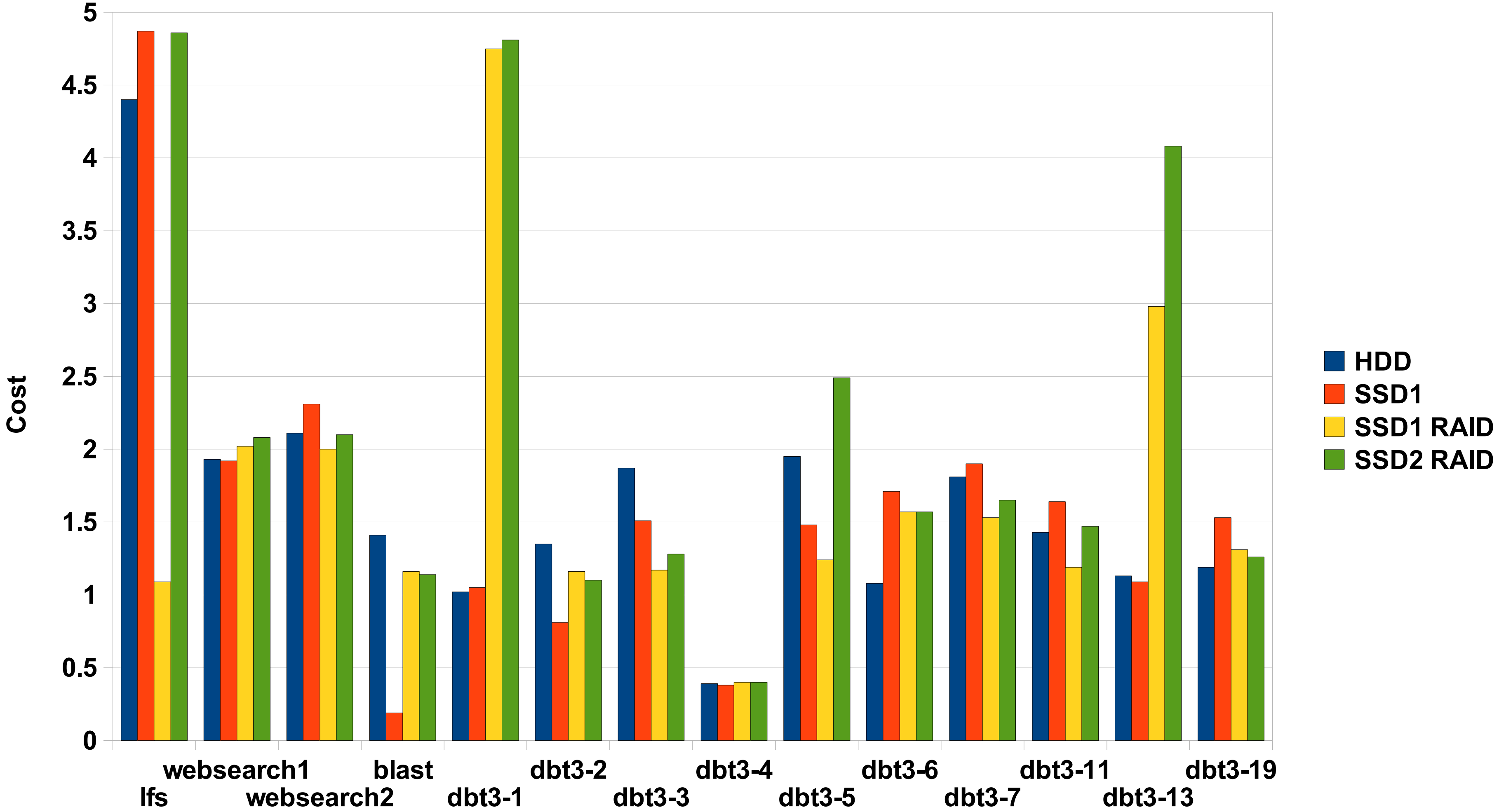


- Accuracy = Amount prefetched and subsequently used by the app / total amount used by the app. Range is between 0 and 1.

Prefetching Cost

- Cost = Amount of Prefetched Data / Amount of Application-Requested Data
- Potentially large, but it's ok as long as we don't saturate the drive bandwidth and don't evict useful data from the cache.
- Prefetchd will increase aggressiveness until the marginal benefit approaches zero.

Prefetching Cost



Conclusion

- We have designed and implemented a data prefetcher for emerging high-performance storage devices, including SSDs that:
  - detects application access patterns
  - retrieves data to match both drive characteristics and application needs
- The prototype is able to achieve a 20% speedup and a 65-70% prefetching accuracy on average.

Acknowledgments

- We would like to thank the anonymous reviewers for their feedback and suggestions.

- This work is in part supported by the National Science Foundation under grants OCI-0937875 and IOS-1124813.

Questions

Thank you.