Hierarchical Data Prefetching in Multi-Tiered Storage Environments

Hariharan Devarajan, Anthony Kougkas, Xian-He Sun
hdevarajan@hawk.iit.edu, akougkas@iit.edu, sun@iit.edu

SCALABLE COMPUTING SOFTWARE LABORATORY

ILLINOIS INSTITUTE OF TECHNOLOGY

Overview

HFetch is a data-centric prefetching decision engine that utilizes system-generated events, while leveraging the presence of multiple tiers of storage, to perform timely hierarchical data placement. HFetch can boost operations by up to **50%.** Compared to other prefetching solutions, HFetch is **10-35%** faster.

HFetch Highlights

- Hierarchical
- *Data-centric*
- Uses System-Push
- Highly Scalable
- Low Application Overhead

Related Work

A. Kougkas, H. Devarajan, X.-H. Sun, "I/O Acceleration via Multi-Tiered Data Buffering and Prefetching", Journal of Computer Science and Technology, 2019, (accepted to appear)

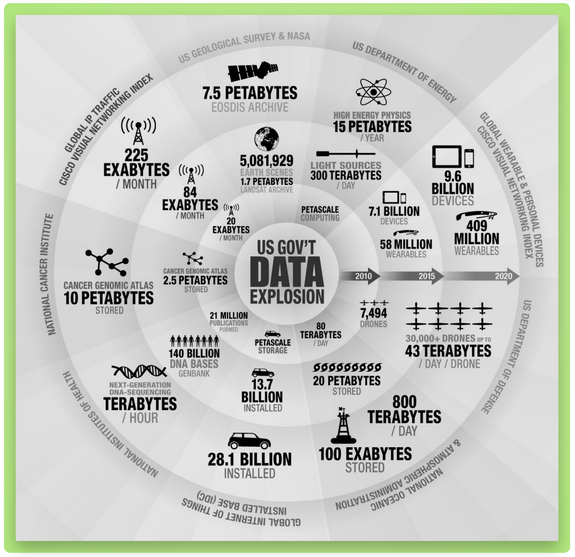
Chen, Yong, Huaiyu Zhu, Philip C. Roth, Hui Jin, and Xian-He Sun. "Global-aware and multi-order context-based prefetching for high-performance processors." The International Journal of High Performance Computing Applications 25, no. 4 (2011): 355-370.

Byna, Surendra, Yong Chen, and Xian-He Sun. "Taxonomy of data prefetching for multicore processors." Journal of Computer Science and Technology 24, no. 3 (2009): 405-417.

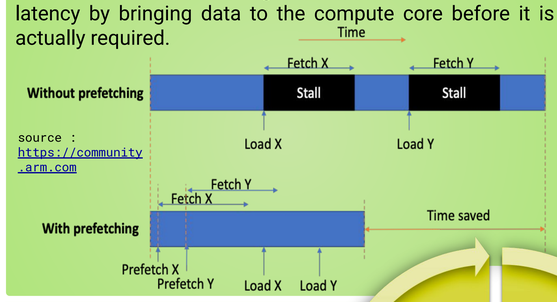
SCAN ME

tinyurl.com/hfetch

Get HFetch on Bitbucket
bitbucket.org/scs-io/hfetch



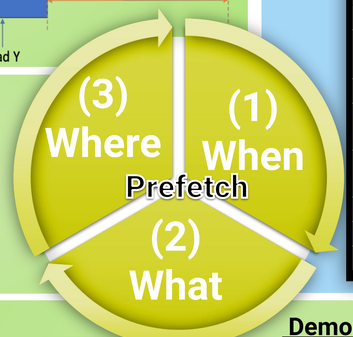
Definition: Data Prefetching is used to reduce the access latency by bringing data to the compute core before it is actually required.



Challenges

- Resource Utilization
- Support Multiple Tiers
- Application-agnostic

Data Centric

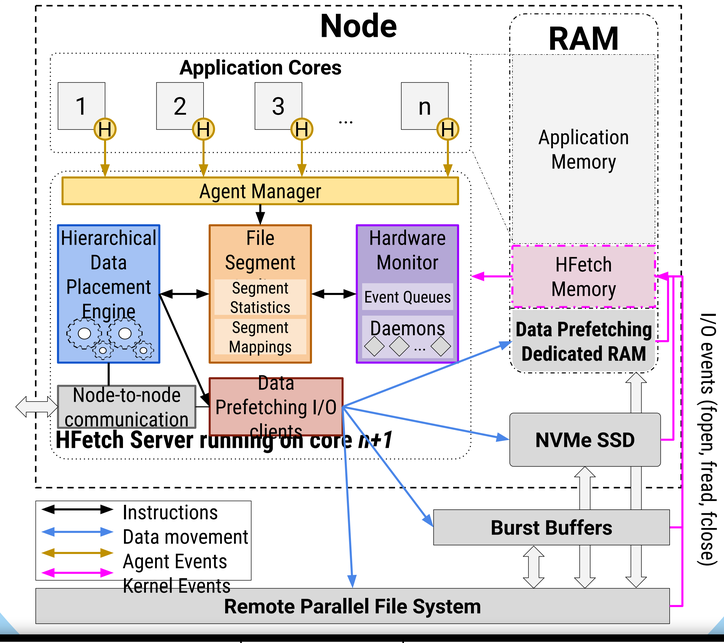


HFetch Design

Initializes client connections. Collects from agents beginning and end of prefetching epoch (fopen to fclose).

Maps file segments to layers of hierarchy based on score.

Allows node-to-node communications regarding metadata calls and data movement.



Calculates file segment statistics, such as access frequency, time of last access, and preceding segment access.

Discovers and monitors all available tiers of hierarchy for events. Events are either file accesses or tier remaining capacity.

Perform I/O calls to and from source tiers and destination tiers.

Client space			Kernel space	HFetch Server space					
Time	Applications		HFetch Agents		inotify_handle	Hardware Monitor	Auditor			Data Placement Engine Tiers: T1<T2<T3<T4
	#1	#2	Agent#1	Agent#2	(push to event queue)		Update Segment Statistics		Calculate Segment Score	
							Frequency	Recency	Sequence	
t0	fopen(f1, READ)	-	start_epoch(f1)	-		inotify_add_watch(f1)	[0,0,0,0]	[0,0,0,0]	null	[0.0,0.0,0.0,0.0,0.0]	[T4,T4,T4,T4]
t1	fopen(f2, WRITE)	-	IGNORE	-							
	-	fopen(f1, READ)	-	start_epoch(f1)		IGNORE					
t2	fread(f1,0,1)	-	-	-	f1,offset:0,size:1,t2	collect_event()	[+1,0,0,0]	[+t2,0,0,0]	prev->s0	[1.0,0.0,0.0,0.0,0.0]	[T1,T4,T4,T4]
t3	fread(f1,1,1)	fread(f1,0,1)	-	-	[{f1,offset:1,size:1,t3}, {f1,offset:0,size:1,t3}]	collect_event()	[+1,+1,0,0]	[+t3,+t3,0,0]	prev->[s0,s1]	[1.5,1.0,0.0,0.0,0.0]	[T1,T2,T4,T4]
t4	fread(f1,0,1)	fread(f1,1,2)	-	-	[{f1,offset:2,size:1,t4}, {f1,offset:1,size:2,t4}]	collect_event()	[+1,+1,+1,0]	[+t4,+t4,+t4,0]	prev->[s0,s1,s2]	[1.5,1.5,1.0,0.0,0]	[T1,T2,T2,T4]
t5	fread(f1,0,1)	-	-	-	f1,offset:0,size:1,t5	collect_event()	[+1,0,0,0]	[+t5,0,0,0]	prev->s0	[1.2,0.5,0.3,0.0]	[T1,T2,T3,T4]
t6	fclose(f1)	-	end_epoch(f1)	-		IGNORE					
t7	fclose(f2)	fclose(f1)	IGNORE	end_epoch(f1)		inotify_rm_watch(f1)					

Demonstrate the inefficiencies in application-centric prefetching

Access Characteristics:
- App A: Sequential
- App B: Strided
- App C: Repeated

Prefetching Approaches:
- Trace-based: KNOWAC
- ML-based: Stacker

Key Observations:
- Application-centric
  - Cache Pollution and Eviction
- Pseudo Hierarchical
  - Misses Pipelining
  - Misses Concurrency

Access Pattern				KNOWAC					Stacker				
Time step	App A	App B	App C	Fetch next	Hit RAM	Miss NVMe	Miss PFS	Cache Copy	Fetch next	Hit RAM	Miss NVMe	Cache Copy	Cache Eviction
t_0	1	1	5	$2^A, 3^B, 5^C$	-	Not Supported	$1^A, 1^B, 5^C$		$2^A, 3^B, 5^C$	-	$1^A, 1^B, 5^C$		
t_1	2	3	5	$3^A, 5^B, 5^C$	$2^A, 3^B$		5^C	✓	$3^A, 5^B, 5^C$	$3^B, 5^C$	2^A	✓	
t_2	3	5	5	$4^A, 1^B, 5^C$	$5^B, 5^C$		3^A	✗	$4^A, 1^B, 5^C$	$5^B, 5^C$	$2^A, 3^B, 5^C, 3^A$	✗✗	✗
t_3	4	1	5	$5^A, 3^B, 5^C$	$4^A, 1^B$		5^C	✓	$5^A, 3^B, 5^C$	$1^B, 5^C$	$2^A, 3^B, 5^C, 3^A, 5^B, 5^C, 4^A$	✗	✗✗
t_4	5	3	5	$6^A, 5^B, 5^C$	$5^B, 5^C$		3^A	✗	$6^A, 5^B, 5^C$	$3^B, 5^C$	$3^B, 5^C, 3^A, 5^B, 5^C, 4^A, 1^B, 5^C, 5^A$	✗	✗✗✗
t_5	6	5	5	-	$6^A, 5^B$		5^C	✓	-	$5^B, 5^C$	$3^A, 5^C, 4^A, 1^B, 5^C, 5^A, 3^B, 5^C, 6^A$	✗✗	✗✗✗
				Prefetch cache space wasted: 20% RAM					Prefetch cache space wasted: 20% RAM, 27% NVMe				
									Unnecessary evictions in 33% of all prefetching operations				

Results