Tree Cache Learning
Or, What I Did this Summer

Jack Weinstein
Argonne National Laboratories

Normal Cache Behavior

- No Caching
  - Each basket request is a separate file transaction
- Caching
  - Cache misses are file transactions
  - No cache fills until after learn phase
    - Basket requests are separate file transactions while learning

Motivation

- Current best for learn phase is N file transactions for each of N branches used
- Can't make good guesses at branch usage
- Few large reads are less expensive than many small reads
- A single large read is not much more expensive than a single smaller read
  - Latency is the dominating factor
- Goal: reduce file read calls for learn phase

Testing

- group.test.hc.NTUP_TOPJET
- ~4000 branches, flat NTuples
- “Large” clusters
- Rewritten
  - Auto-flush 666 entries
  - Baskets sorted by branch
  - Baskets sorted by entry

Testing

- Files on NFS storage
- ROOT macro reads all entries of tree
- Reads a subset of branches
- Learn Entries left as default 100 (far below first cluster boundary)

Changes already in ROOT Trunk

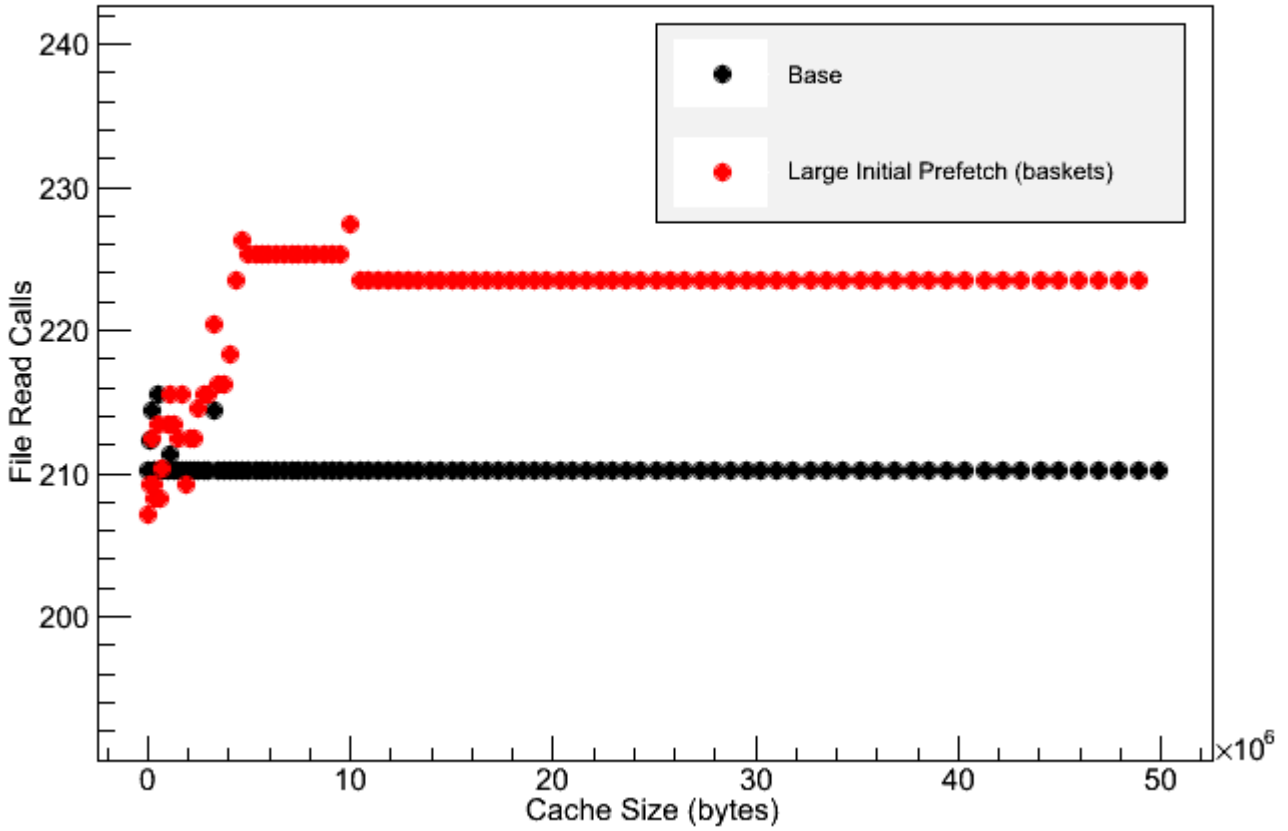
- Added TTreeCache::Enable() and Disable()
  - Duplicate / extraneous calls to TTreeCache::ReadBuffer
  - TFile::fReadCache
- Extraneous cache clear / fill after learn phase

Learning Phase Strategies

- Large Initial Prefetch
  - Large, single read
  - Data from beginning of Tree
- Neighboring Data Prefetch
  - On basket request, prefetch adjacent data on disk
  - Exploit physical locality of related branches
- By baskets
  - Add baskets similarly to cache fill
- By raw data blocks
  - Read blocks from disk, basket or not
  - On block request, check contained in read block

Prefetching by Baskets

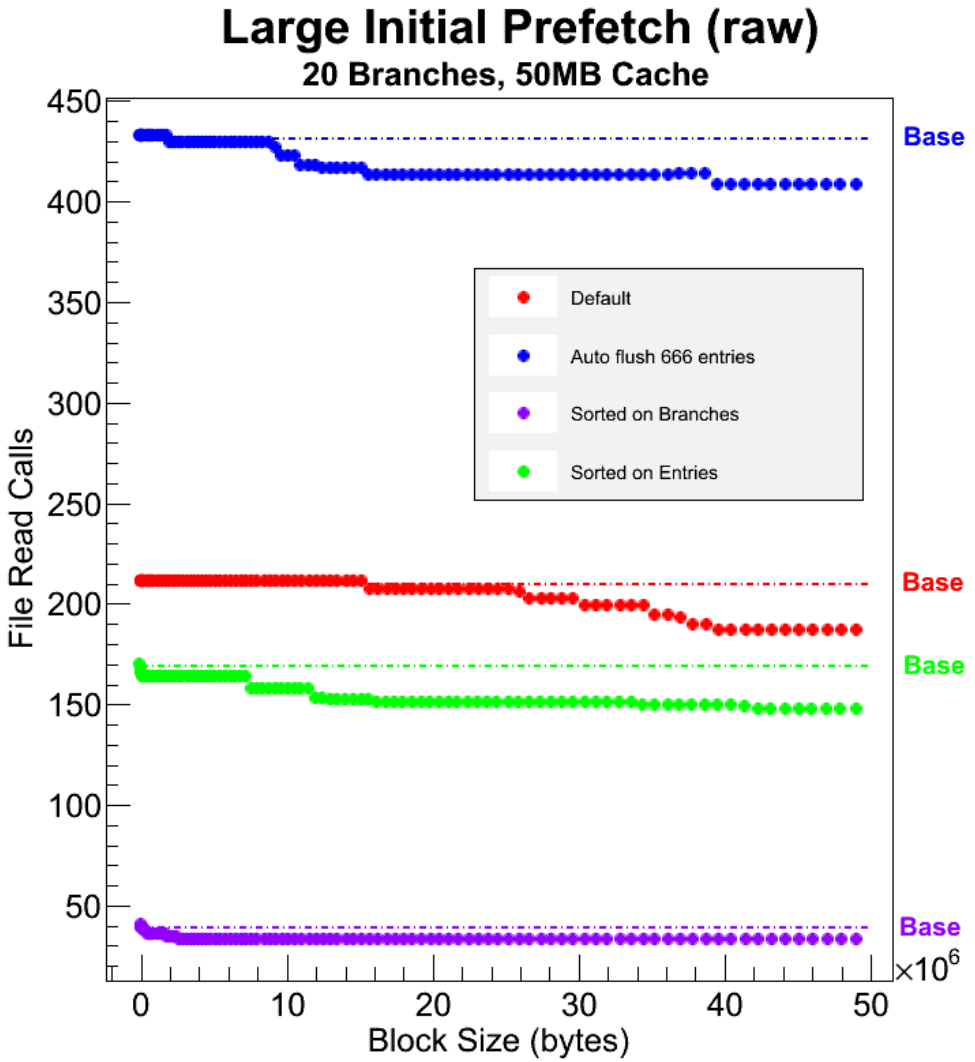
- Iterate over baskets of tree branches, add to cache
- Works well for cache fill – but not for the learn phase, wide in branches and shallow in baskets
- Small cache compared to branches and cluster concerning
- Too many fragmented reads
- Looks like: raw block size = cache size

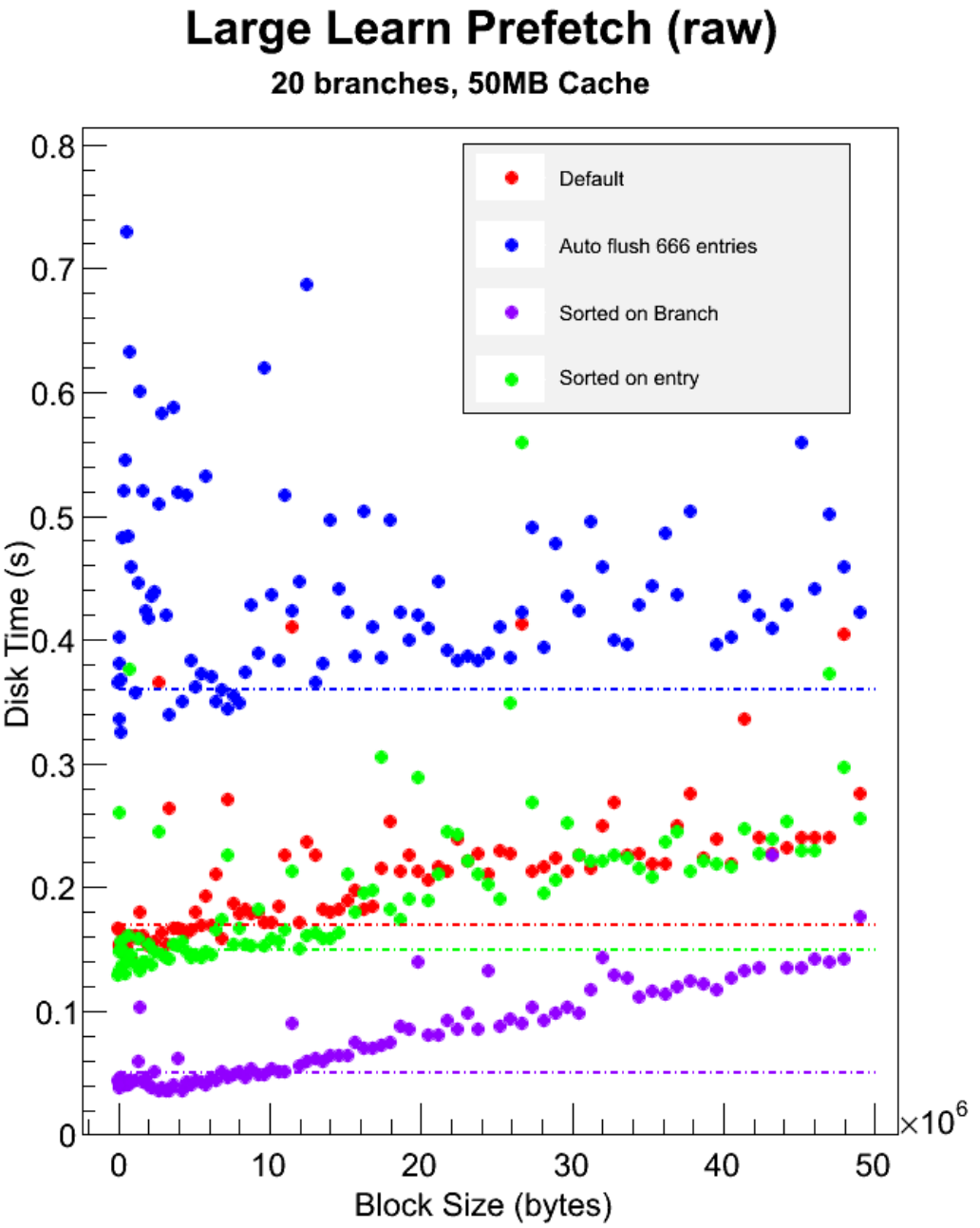


- 20 branches (not random)
- Default basket arrangement
- Base (no changes)
- Large initial prefetch, selecting baskets

Large Initial Prefetch as a Raw Block

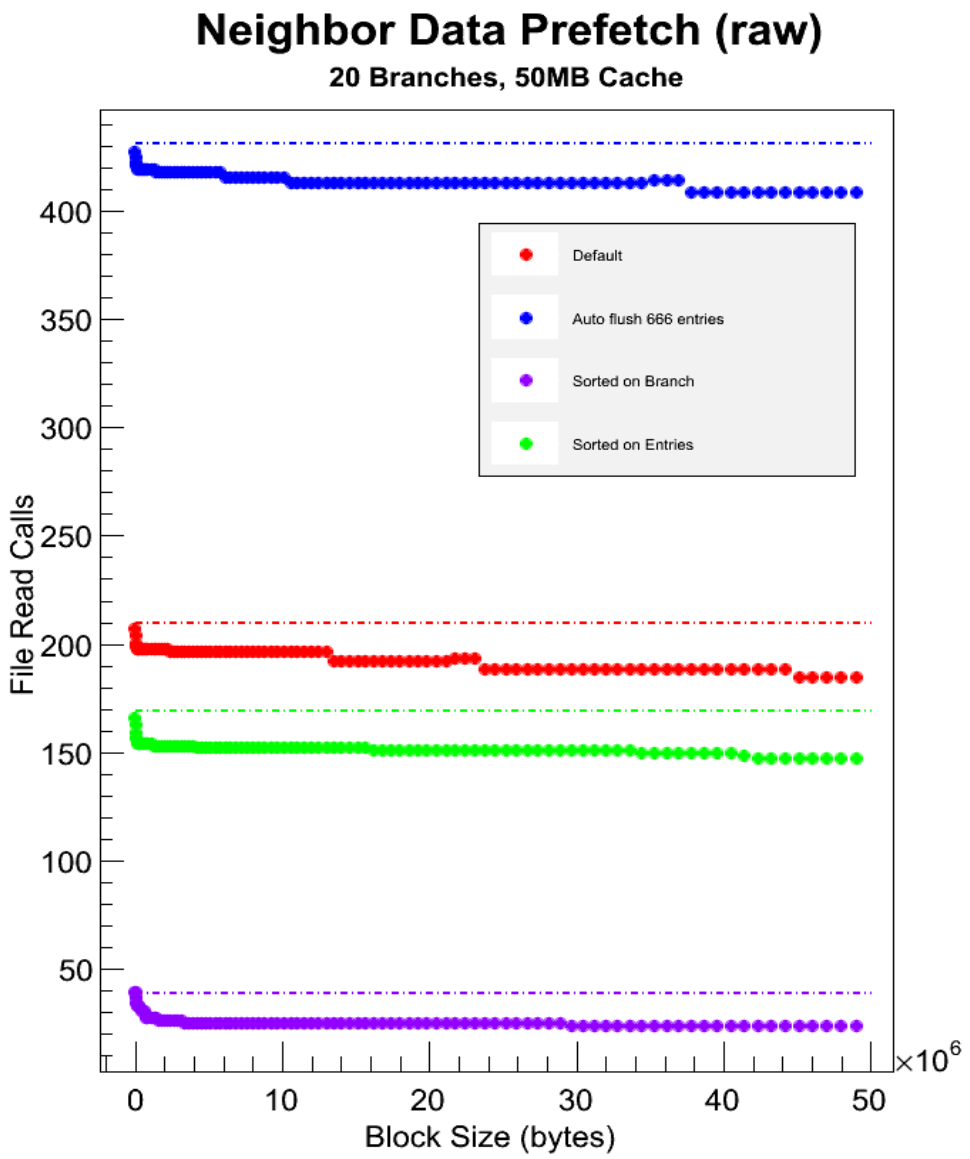
- Read a large block of data from the beginning of tree data
- No sorting, guaranteed single read
- Dealing with “nice” files. Trees are not entangled on disk
- Block size compared to cluster
  - Benefits from small initial cluster
- Possible to grab data beyond learn phase

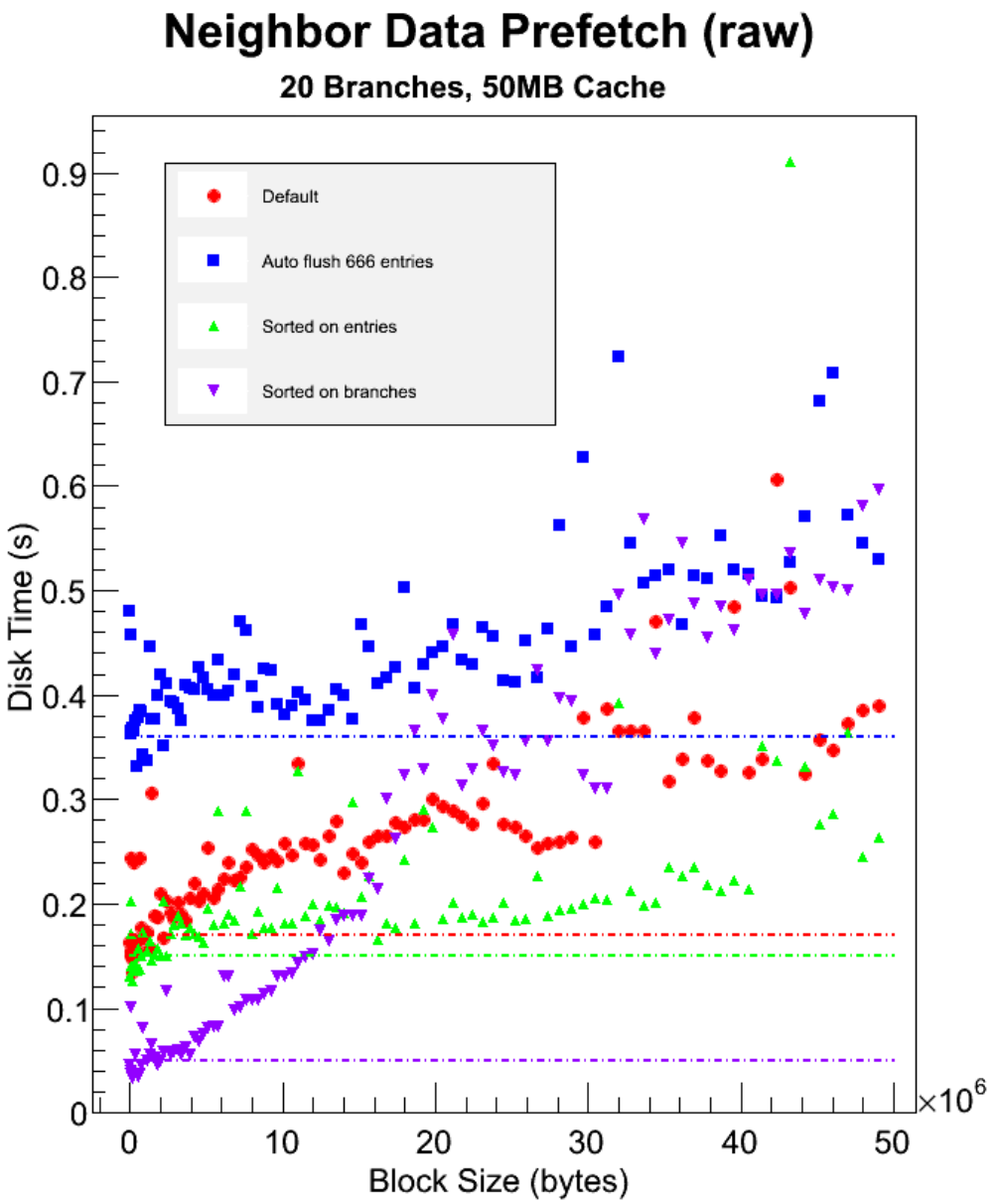

Large Initial Prefetch (raw)
20 Branches, 50MB Cache
File Read Calls
Block Size (bytes)
×10^6
Default
Auto flush 666 entries
Sorted on Branches
Sorted on Entries
Base
Base
Base

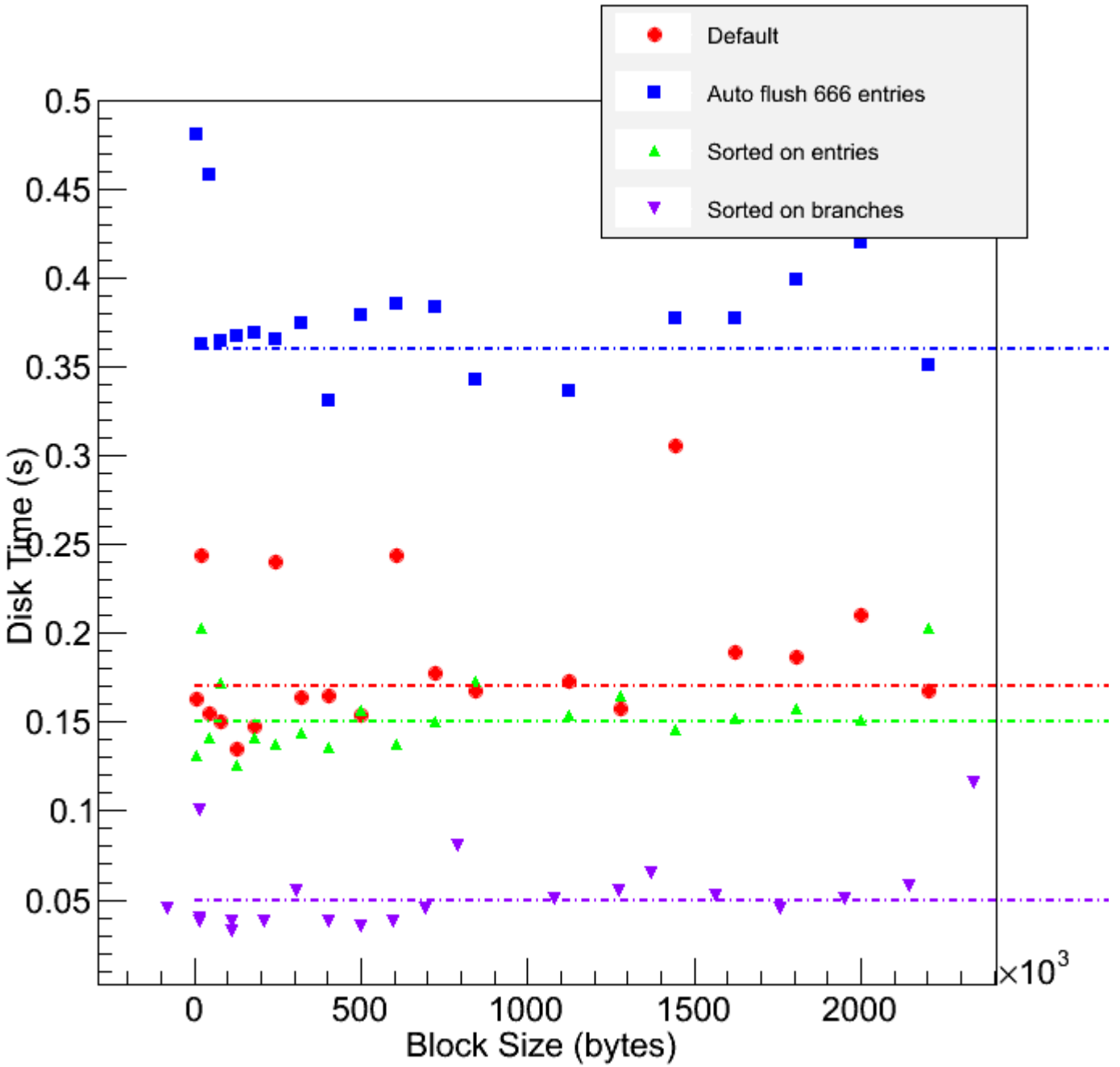

Large Learn Prefetch (raw)
20 branches, 50MB Cache
Disk Time (s)
Block Size (bytes)
×10^6
Default
Auto flush 666 entries
Sorted on Branch
Sorted on entry

Neighbor Data Prefetch as a Raw Block

- During learn phase, before cache miss, grab sequential block
- Exploit physical locality of related baskets
- Similar to TFile readahead
  - Don't know next read, no gap to fill
- Smaller blocks are sufficient to reduce reads
- Read overhead increases with branches used

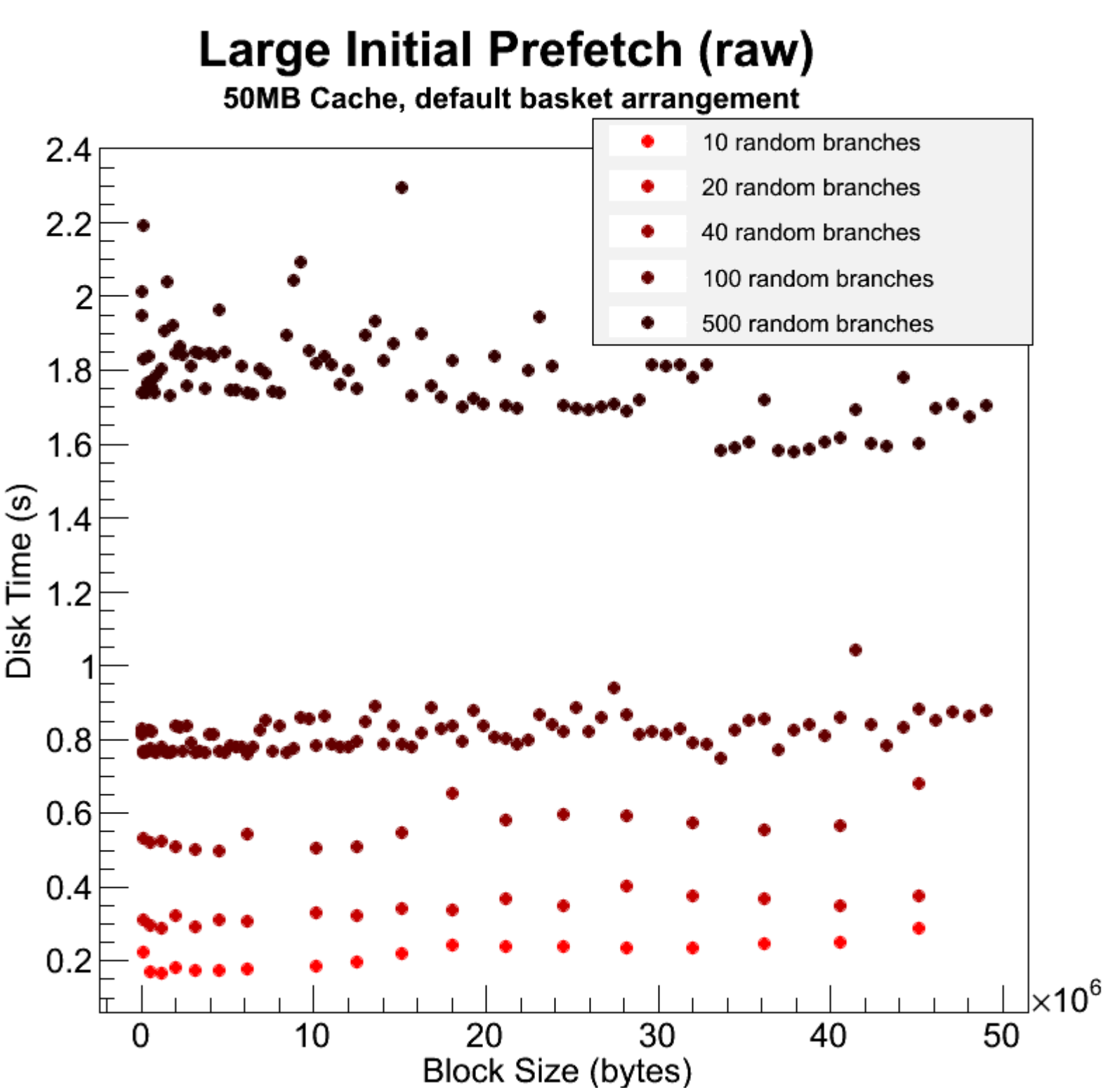

Neighbor Data Prefetch (raw)
20 Branches, 50MB Cache
Default
Auto flush 666 entries
Sorted on Branch
Sorted on Entries
File Read Calls
400
350
300
250
200
150
100
50
0
10
20
30
40
50
×10^6
Block Size (bytes)


Neighbor Data Prefetch (raw)
20 Branches, 50MB Cache
Default
Auto flush 666 entries
Sorted on entries
Sorted on branches
Disk Time (s)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
×10^6
Block Size (bytes)


Default
Auto flush 666 entries
Sorted on entries
Sorted on branches
Disk Time (s)
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
500
1000
1500
2000
×10^3
Block Size (bytes)

With More/Different Branches

- Greater number of random branches
  - Read baskets get closer
  - File read calls decrease more sharply
- Neighbor data prefetch makes more overhead reads

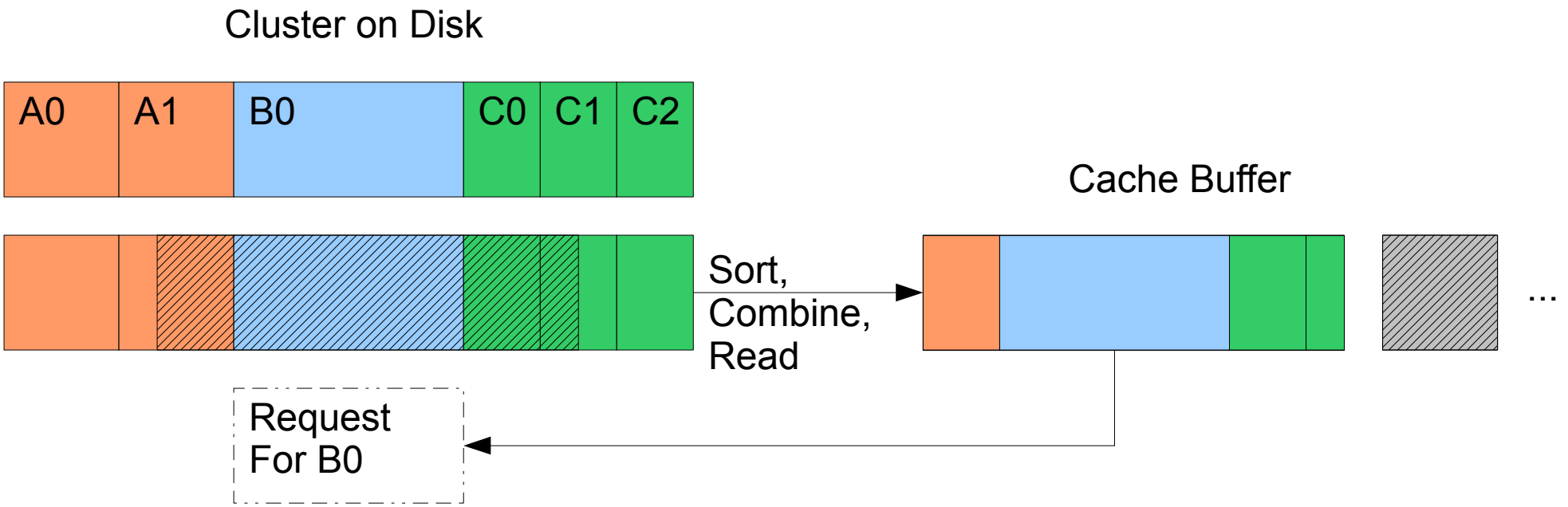

Large Initial Prefetch (raw)
50MB Cache, default basket arrangement
10 random branches
20 random branches
40 random branches
100 random branches
500 random branches
Disk Time (s)
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
10
20
30
40
50
×10^6
Block Size (bytes)

Conclusions

- Neighbor Data Prefetch works well for small block sizes
  - Sharp decrease in read calls with block size
- Large Initial Prefetch works well for “large” blocks compared to cluster size
  - Constant overhead disk time for fixed block sizes
  - Slower decrease in read calls
- Most cases, trade read calls for disk time

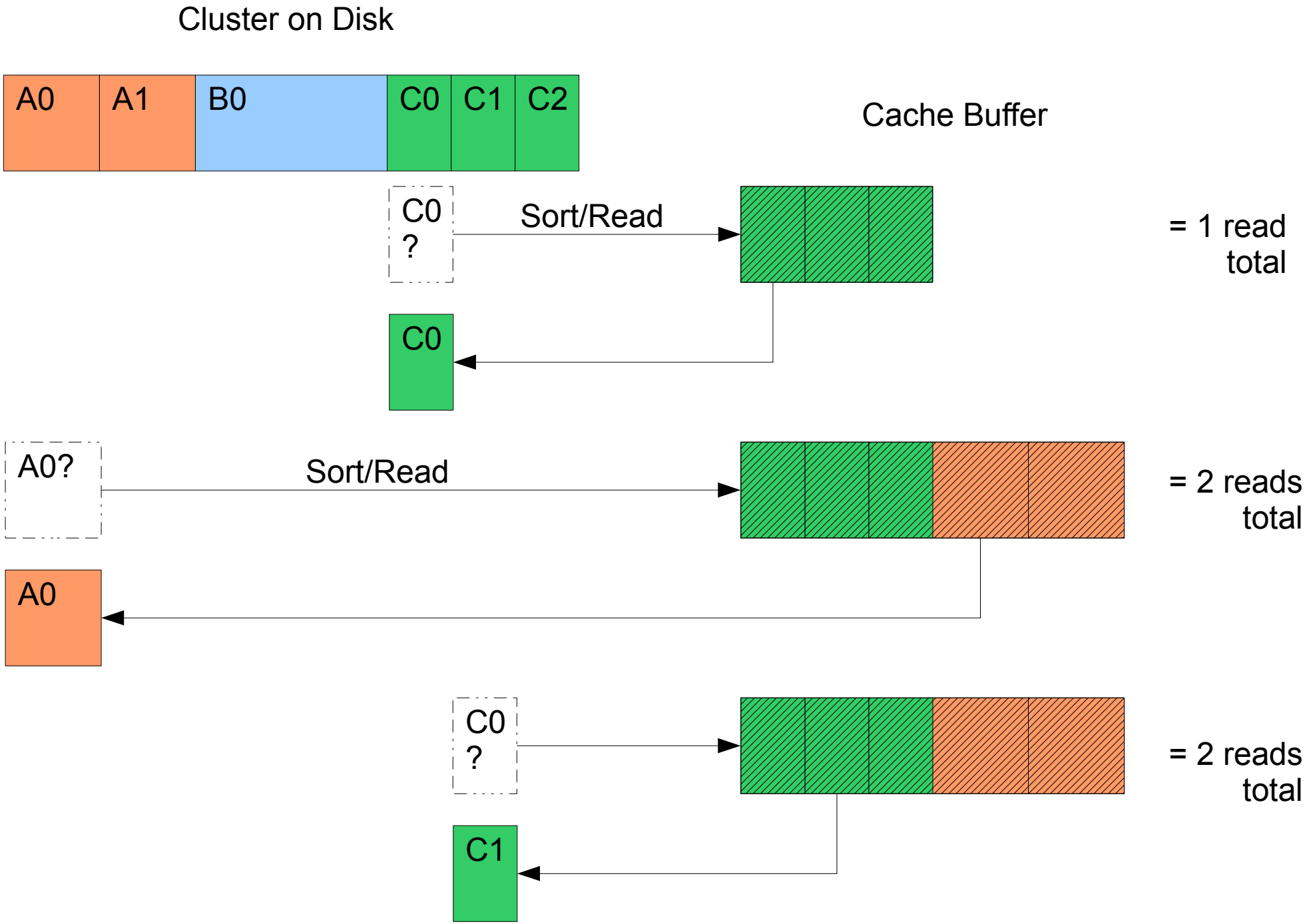
ReadBuffer Overload

- TTreeCache::ReadBufferExtNormal
  - Overloads TFileCacheRead::ReadBufferExtNormal
  - Extends functionality



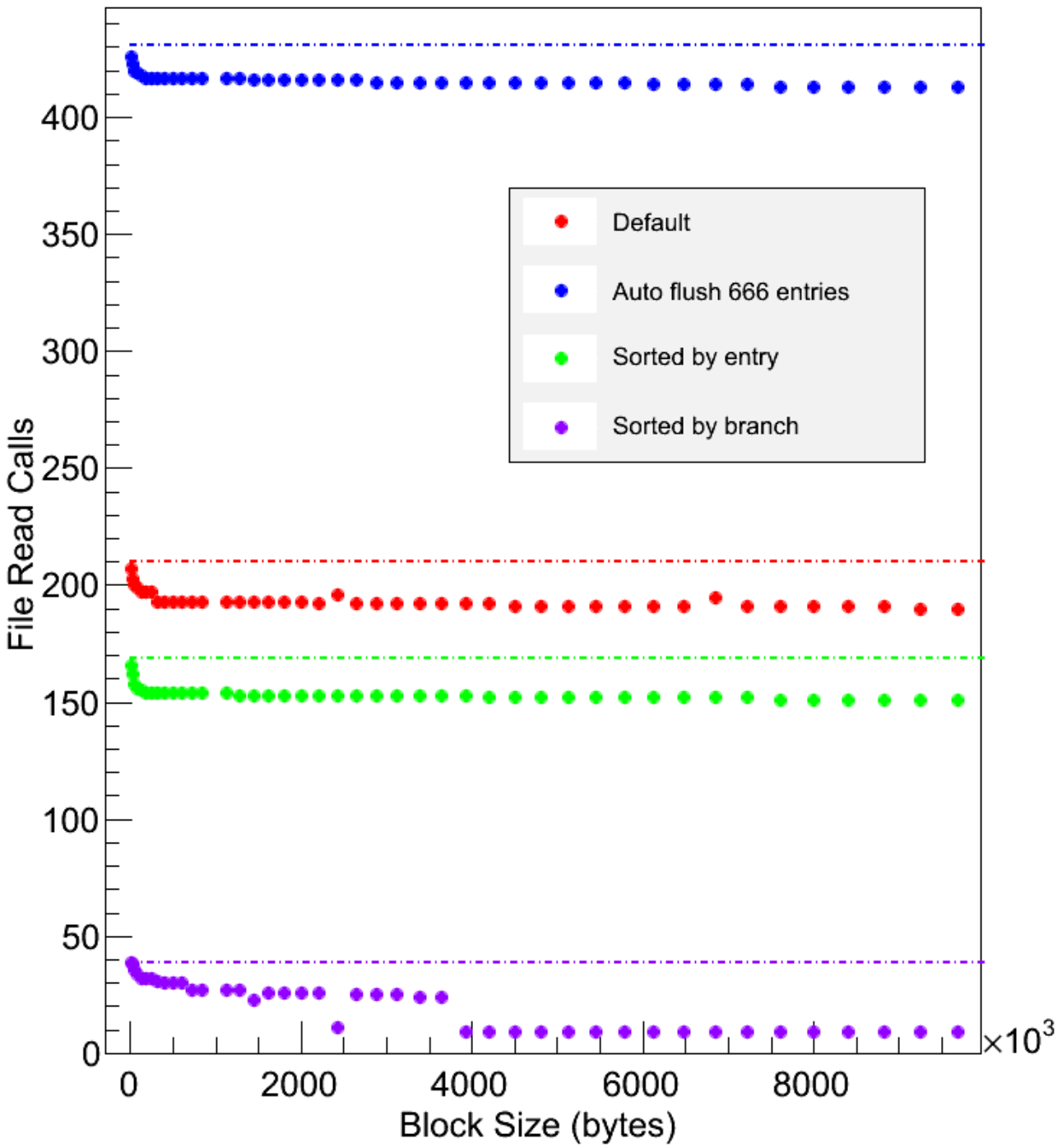
Afterthought

- It would be nice to be able to read data into the cache without clearing the cache
  - Recycle reads
  - Would work well with neighboring data prefetch
  - Could mix large initial prefetch with neighboring data prefetch


Cluster on Disk
A0
A1
B0
C0
C1
C2
Cache Buffer
C0
?
Sort/Read
= 1 read
total
C0
A0?
Sort/Read
= 2 reads
total
A0
C0
?
= 2 reads
total
C1

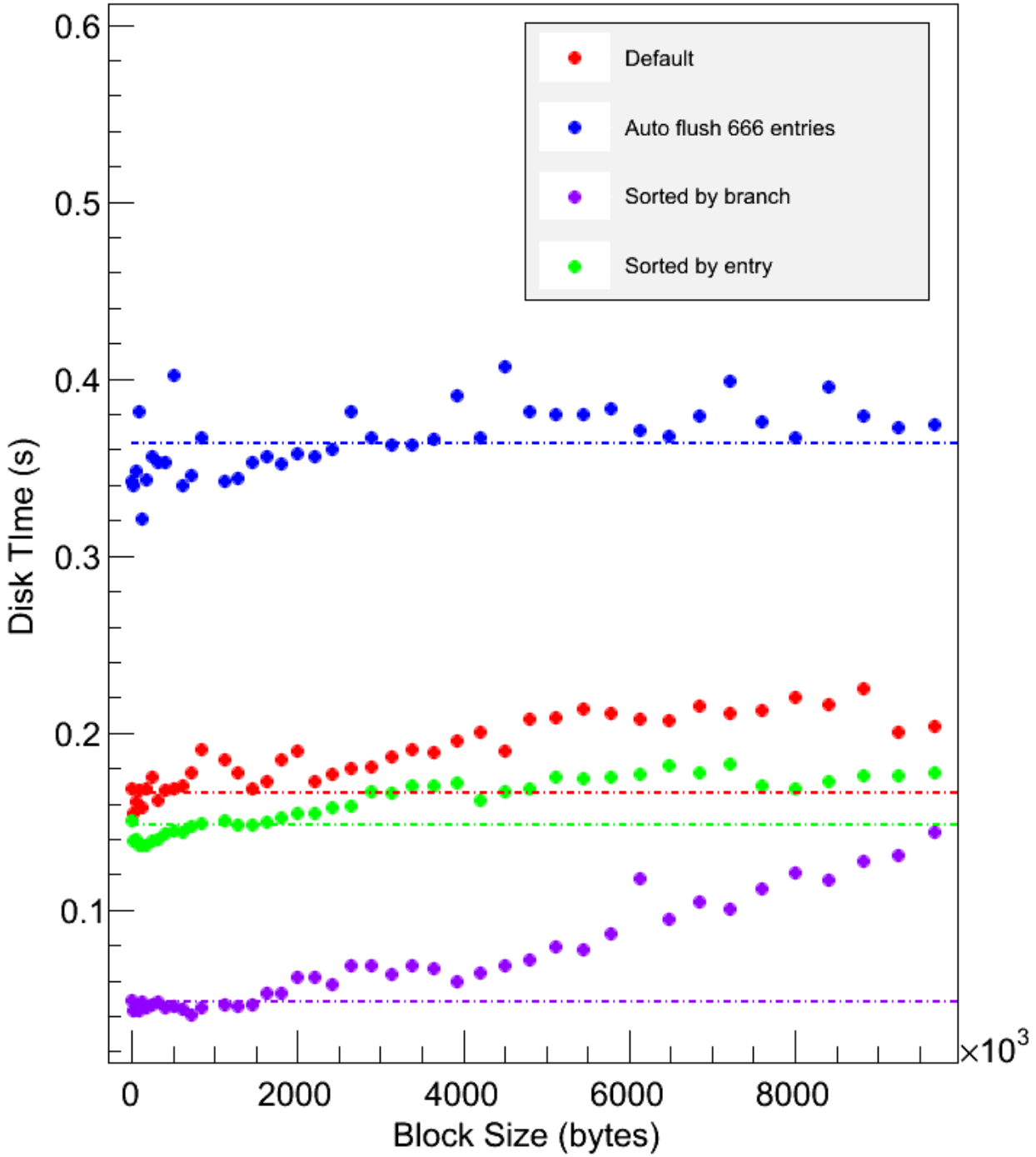
Neighbor Data Prefetch with Cache Modifications

- Don't clear cache (until after learn phase, before cache fill)
- Don't throw away learn phase reads
- Overhead in bytes read is never more than the cache size
- Larger decrease in disk reads
- Slight decrease in overall disk time for small block sizes


Neighbor Data Prefetch (raw)
20 branches, 50MB Cache, with Cache Modifications
File Read Calls
Block Size (bytes)
0
50
100
150
200
250
300
350
400
0
2000
4000
6000
8000
×10^3
Default
Auto flush 666 entries
Sorted by entry
Sorted by branch

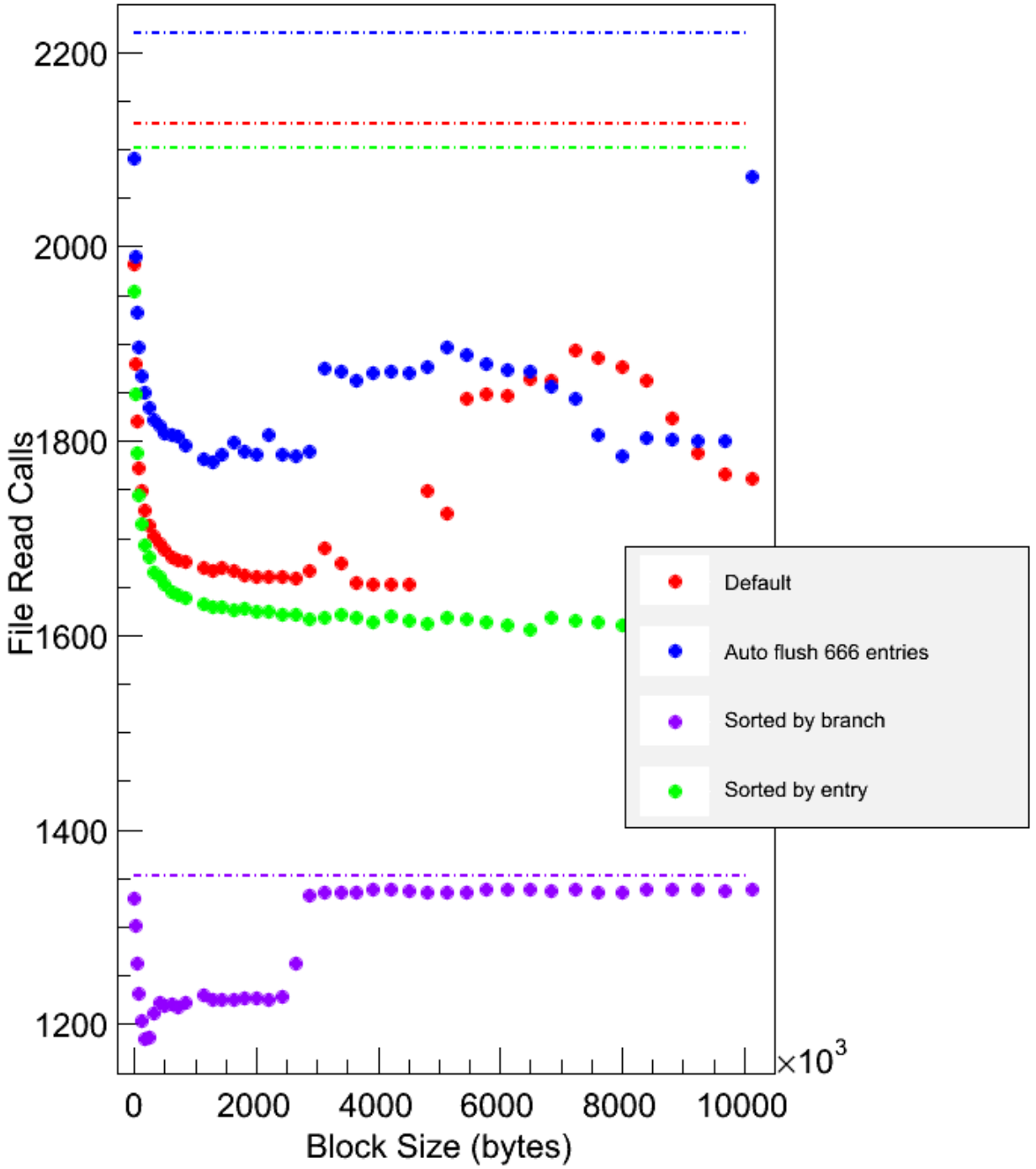
Neighbor Data Prefetch (raw)

20 branches, 50MB Cache, with Cache Modifications



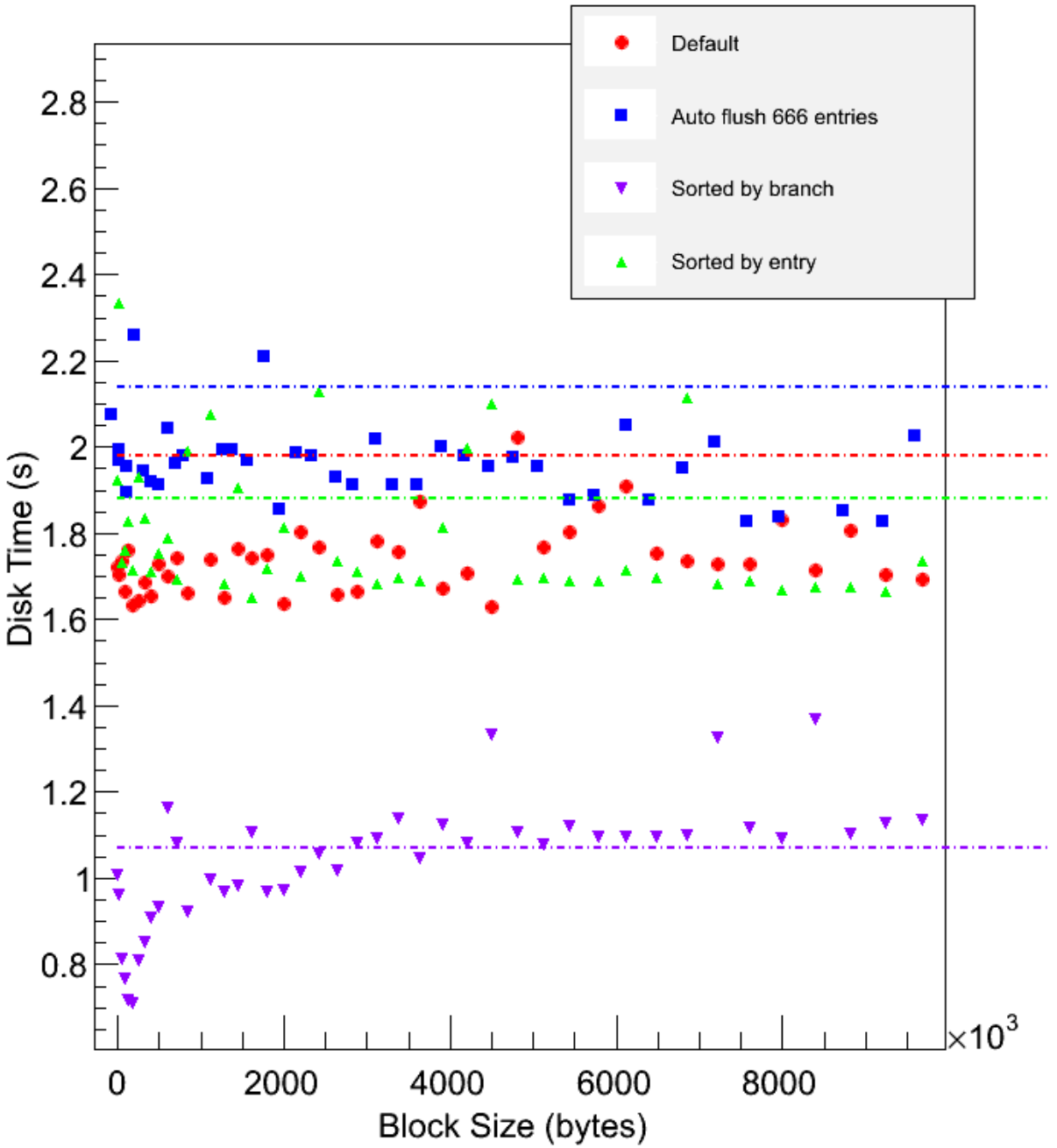
Neighbor Data Prefetch (raw)

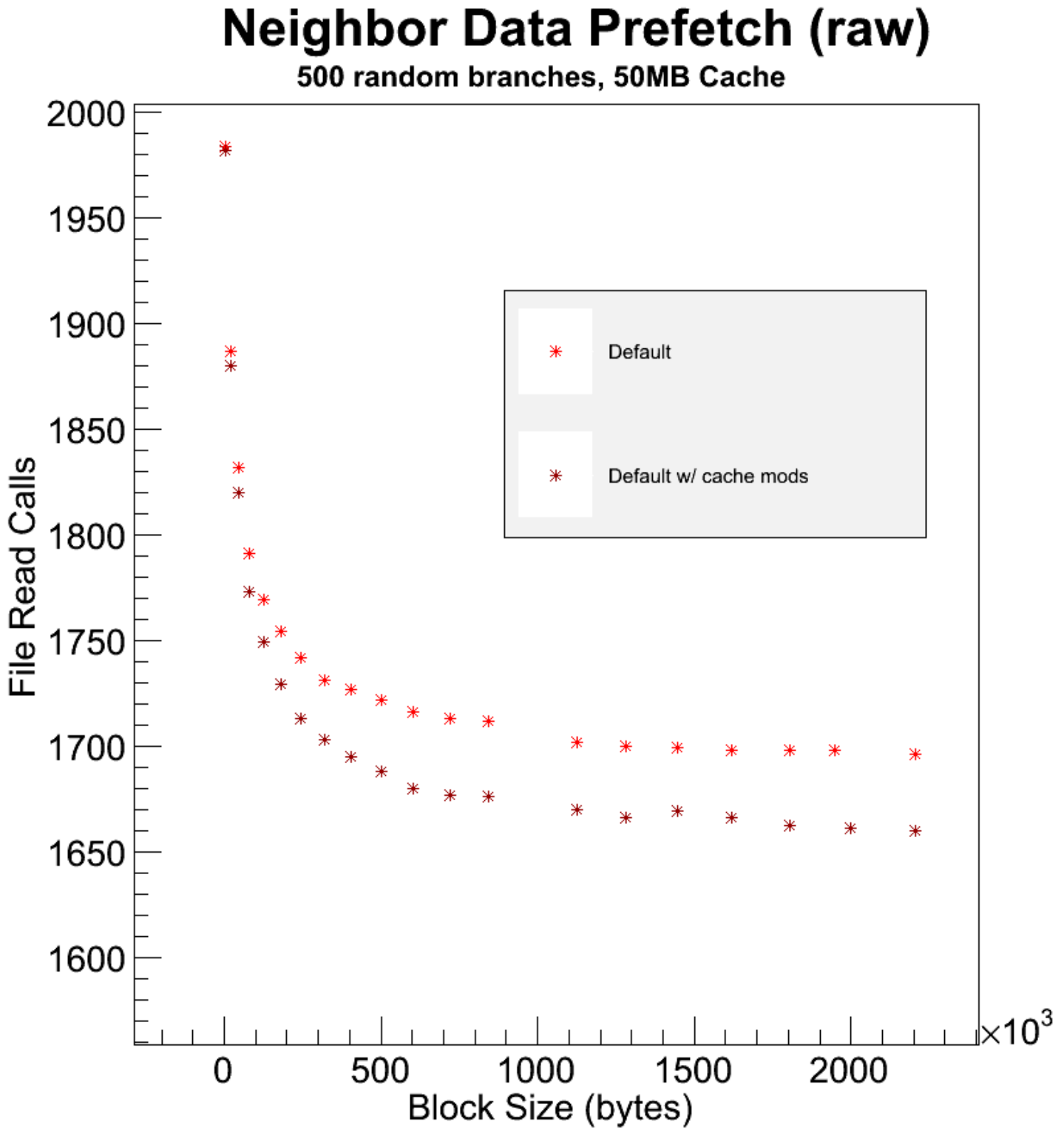
500 random branches, 50MB Cache, with Cache Modifications

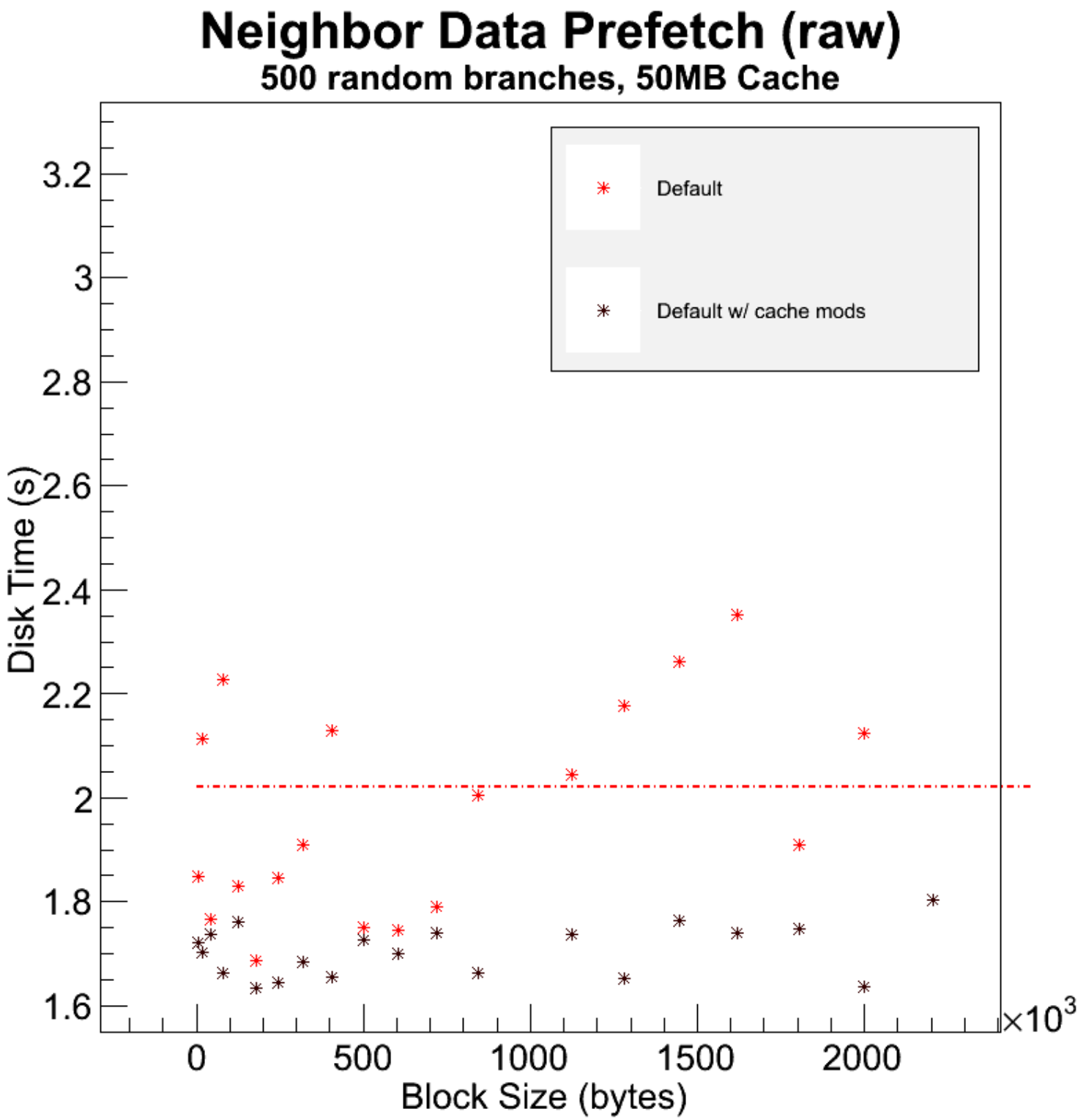


Neighbor Data Prefetch (raw)

500 branches, 50MB Cache, with Cache Modifications




Neighbor Data Prefetch (raw)
500 random branches, 50MB Cache
File Read Calls
2000
1950
1900
1850
1800
1750
1700
1650
1600
0
500
1000
1500
2000
×10^3
Block Size (bytes)
Default
Default w/ cache mods


Neighbor Data Prefetch (raw)
500 random branches, 50MB Cache
Default
Default w/ cache mods
Disk Time (s)
3.2
3
2.8
2.6
2.4
2.2
2
1.8
1.6
0
500
1000
1500
2000
×10^3
Block Size (bytes)

Neighbor Data Prefetch (raw)

500 random branches, 50MB Cache