Linux 2.6 performance improvement through readahead optimization

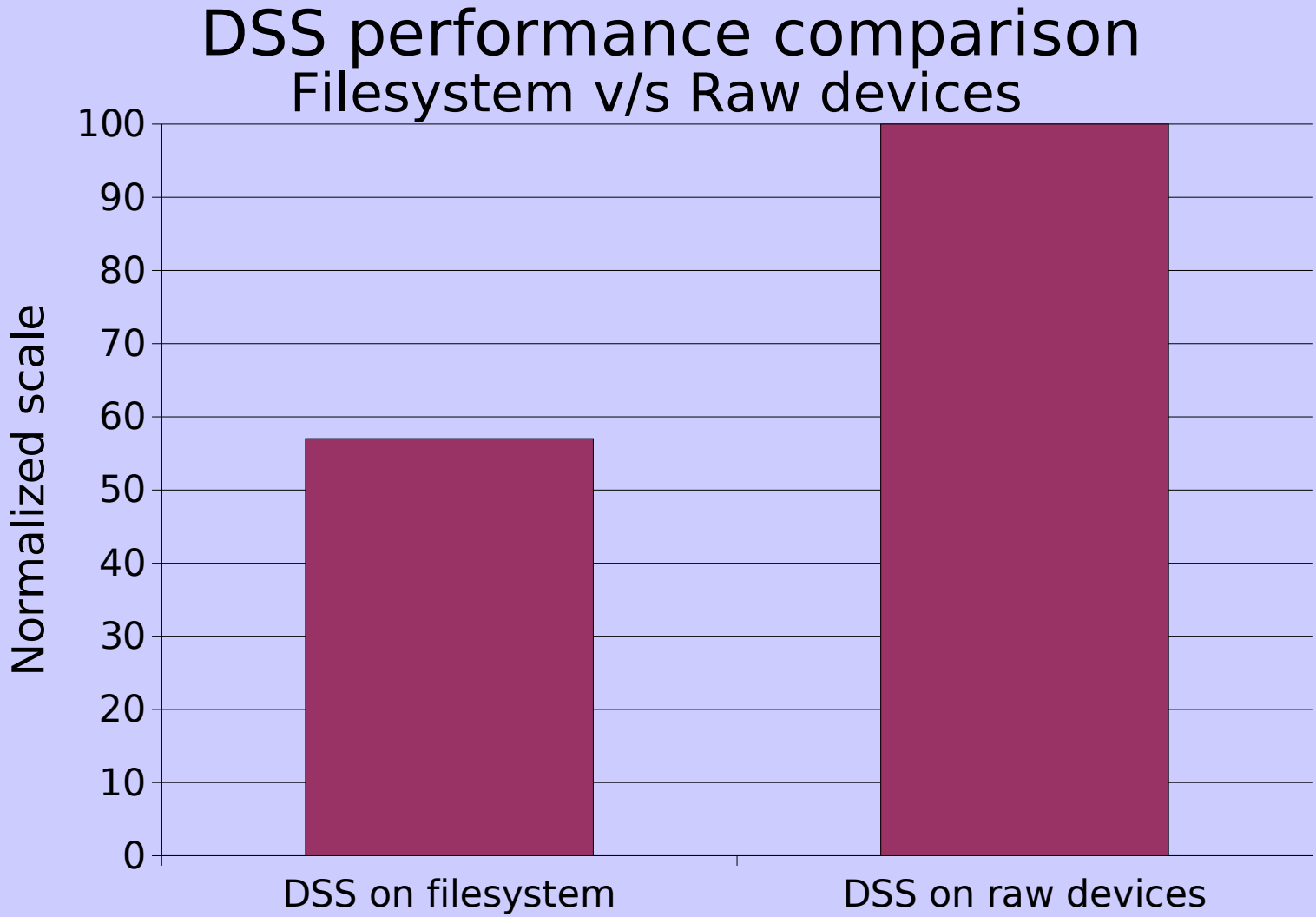
Ram Pai
(linuxram@us.ibm.com)

Badari Pulavarty
(badari@us.ibm.com)

Mingming Cao
(mcao@us.ibm.com)

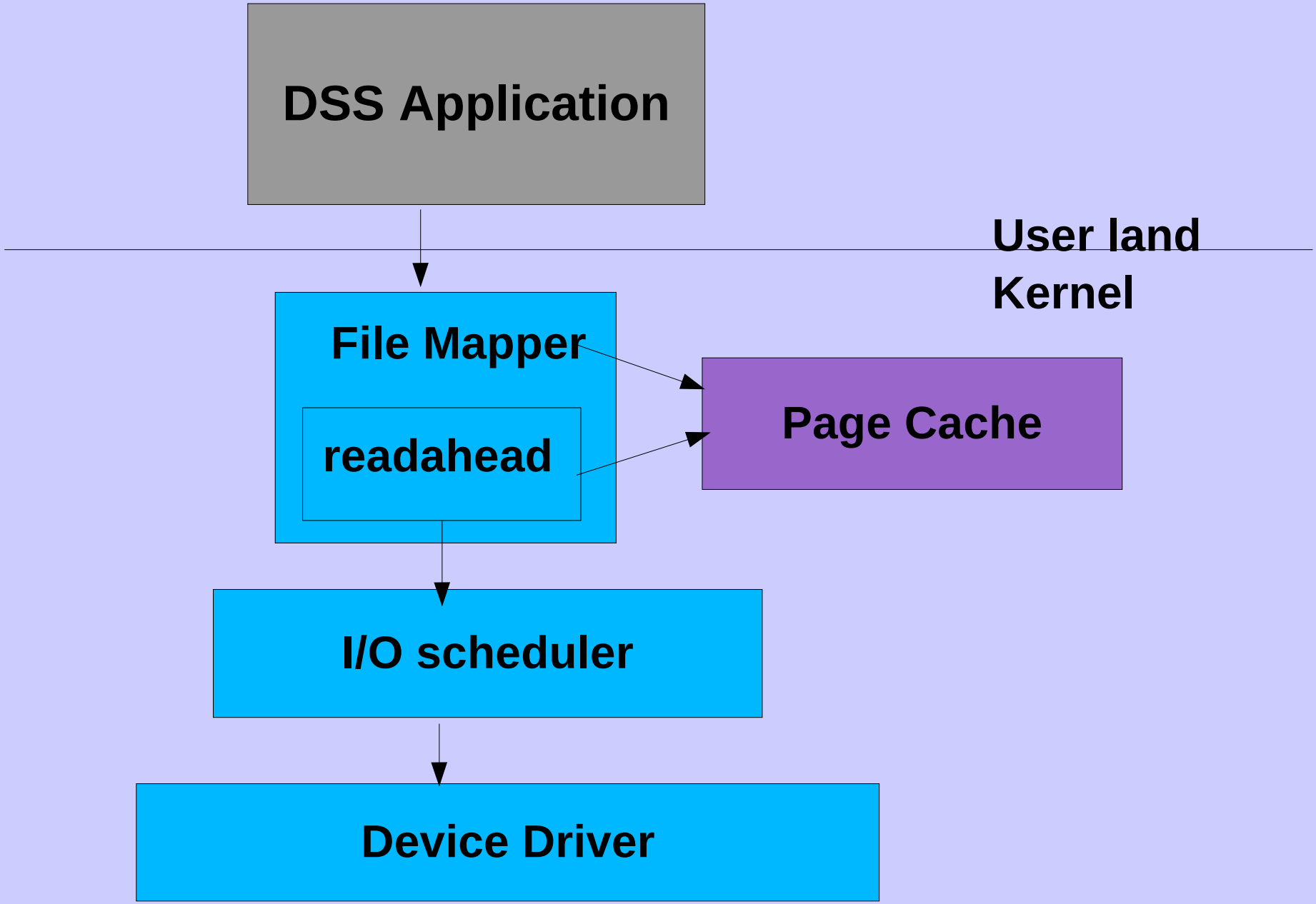
LTC-IBM

What is the problem?

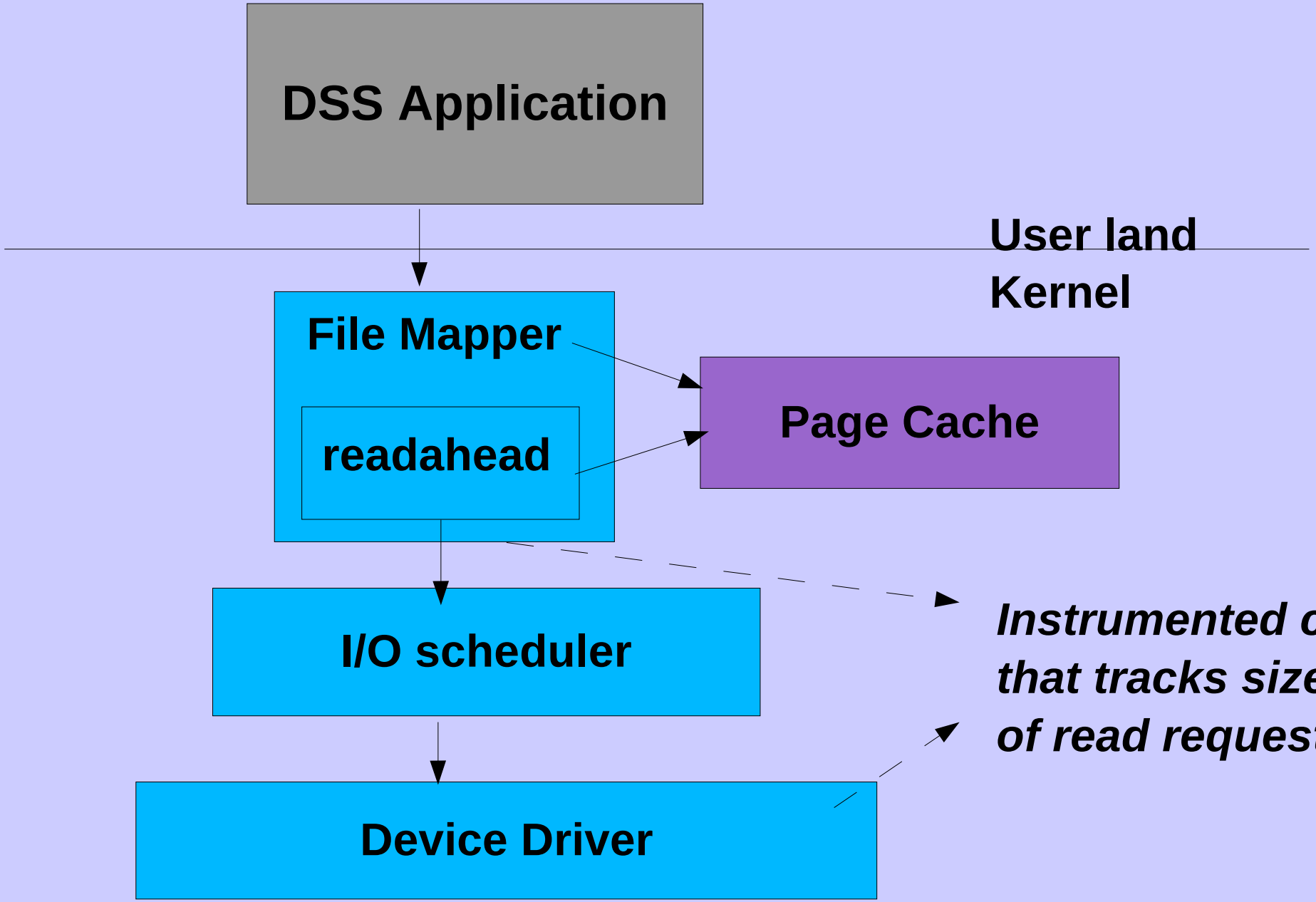


DSS: Decision Support System

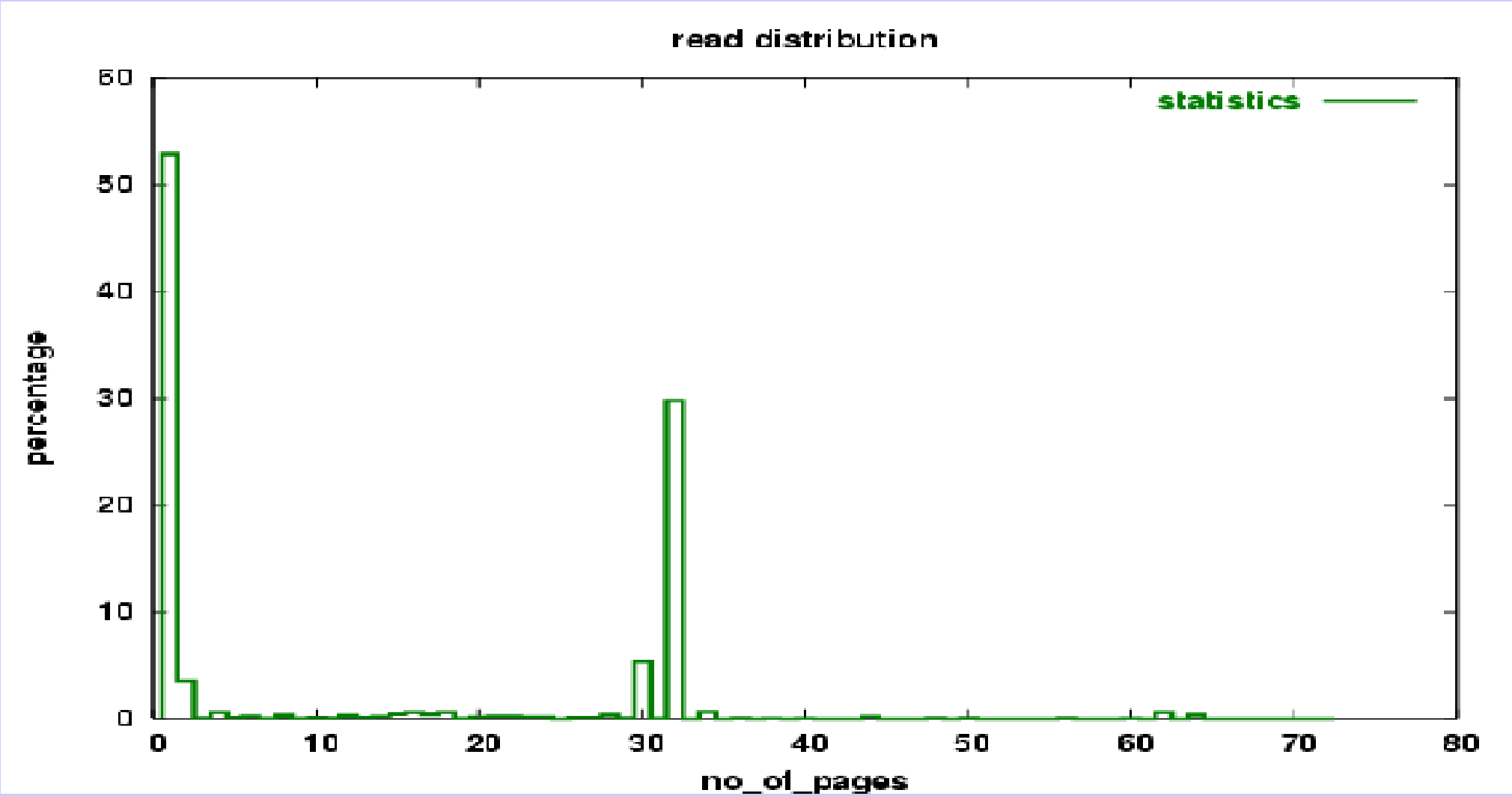
I/O architecture



Where is the bottleneck?



Results of instrumentation



NOTE: application is generating mostly 256K random reads

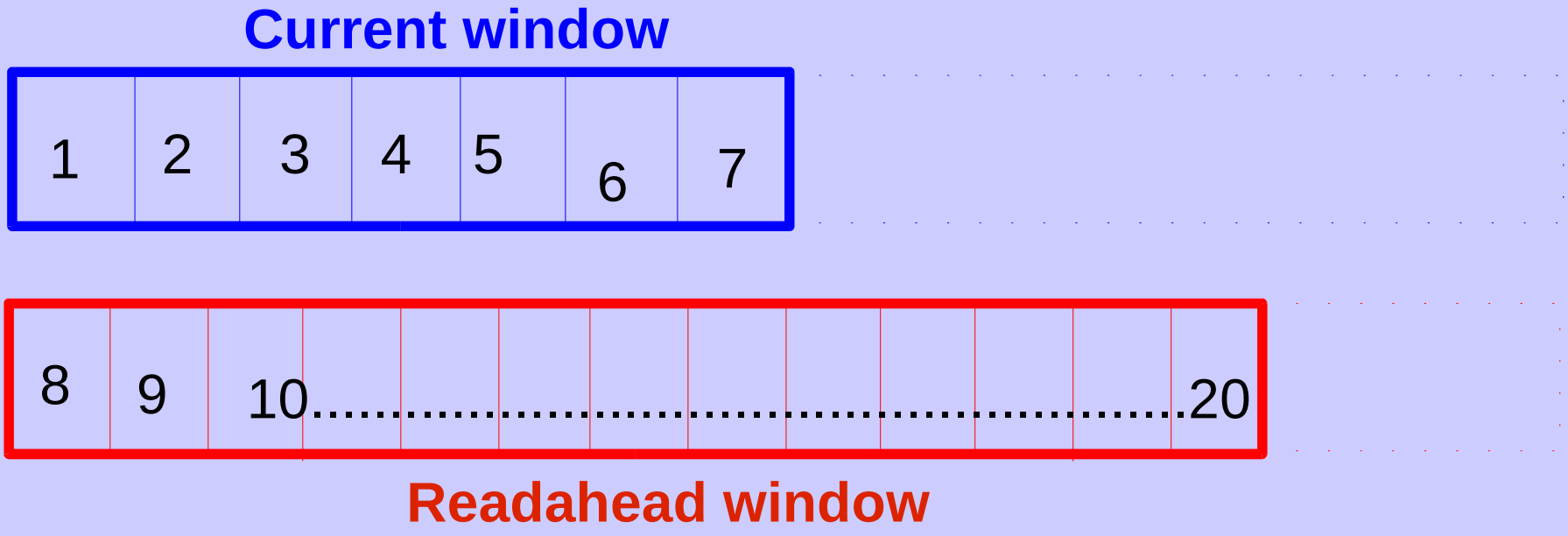
Results of the instrumentation cont..

- **I/O scheduler could not merge requests because ....**
- **It received one-page read requests 50% of the time because...**
- **READAHEAD did not READ AHEAD!**
- **Result: Synchronous 1 page reads most of the time.**

READAHEAD! What is its job?

- **Predict future read requests using past request patterns.**
- **Asynchronously bring future read requests into the page cache.**
- **Maximize page-cache hit!**
- **Minimize synchronous page read.**
- **Minimize waste of resources.**
- **Maximize size of the asynchronous read requests.**

Readahead algorithm in 2.6.0

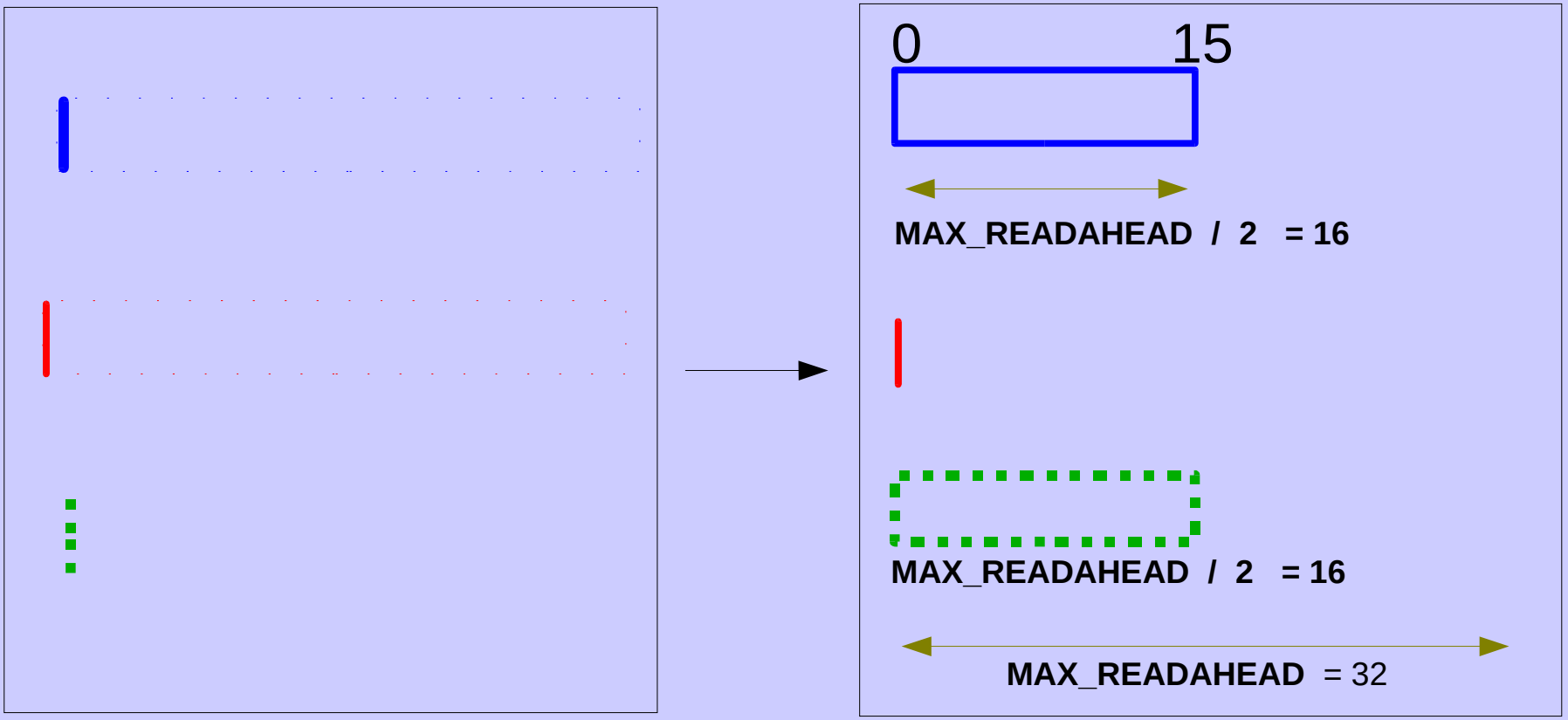


2.6.0 readahead cont..

- **Current window** tracks pages that satisfy the current read request.
- **Readhead window** tracks pages that satisfy future read requests.
- **Next window size** tracks the number of page frames in the next **readahead window** or the next **current window.**

2.6.0 readahead cont..

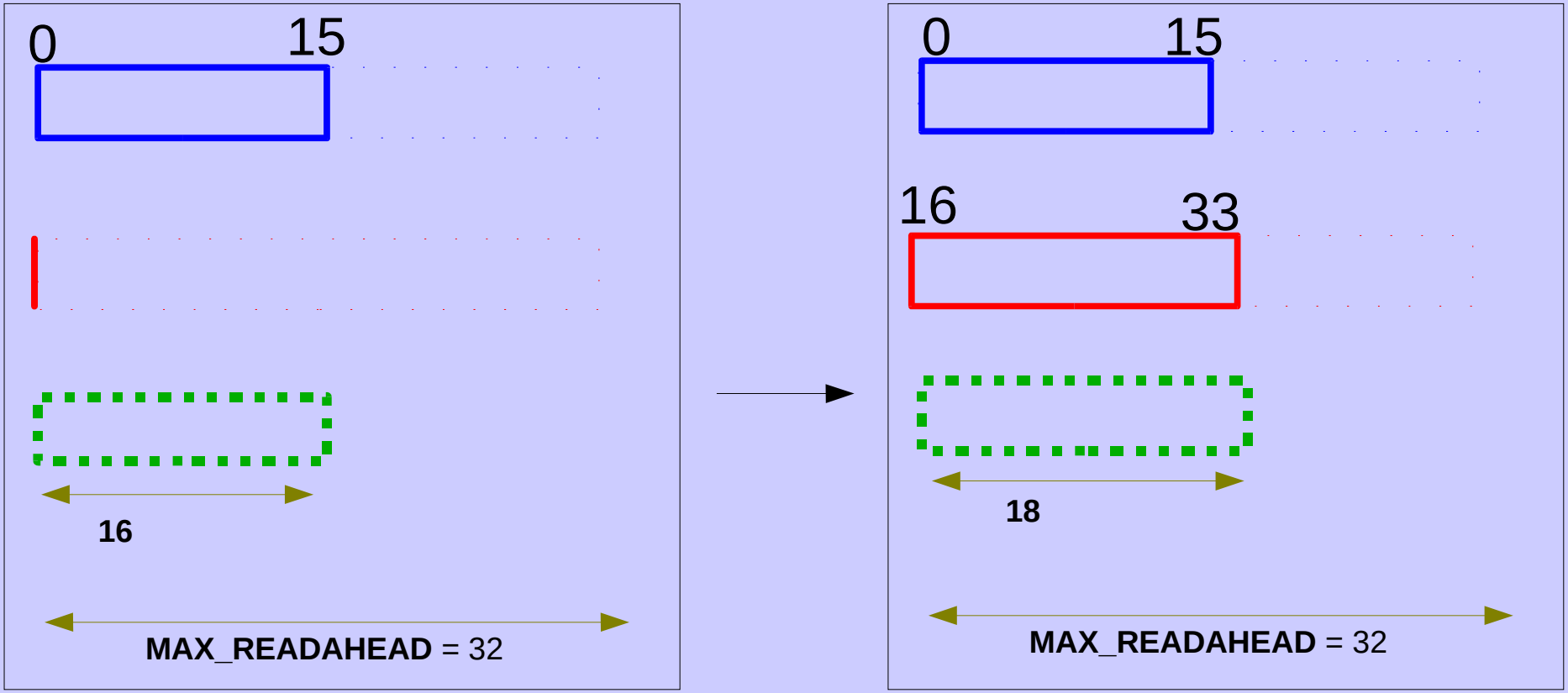
Case 1: read request for page 0 size 1



Note: **current window** is populated

2.6.0 readahead cont..

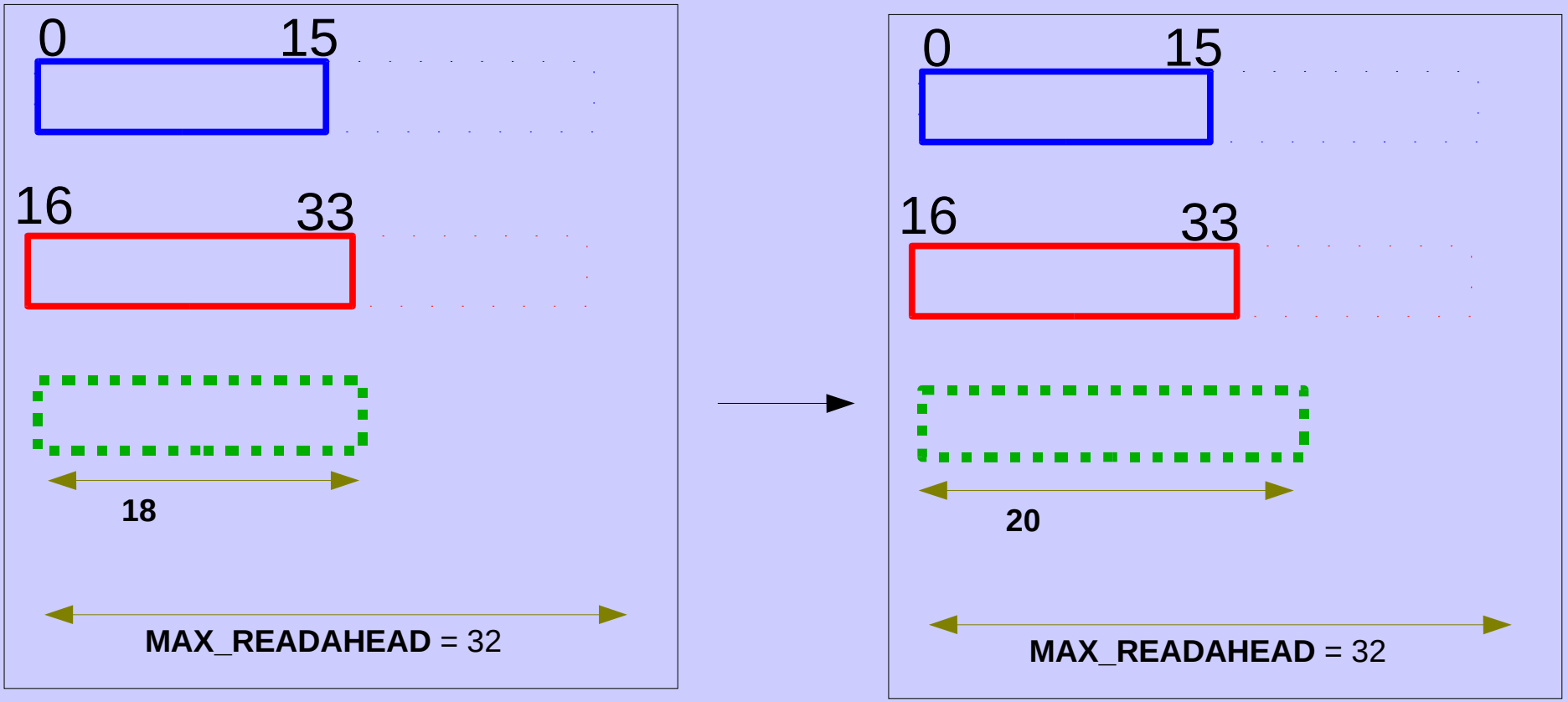
Case 2: read request for page 1



Note: **readahead window** is populated

2.6.0 readahead cont..

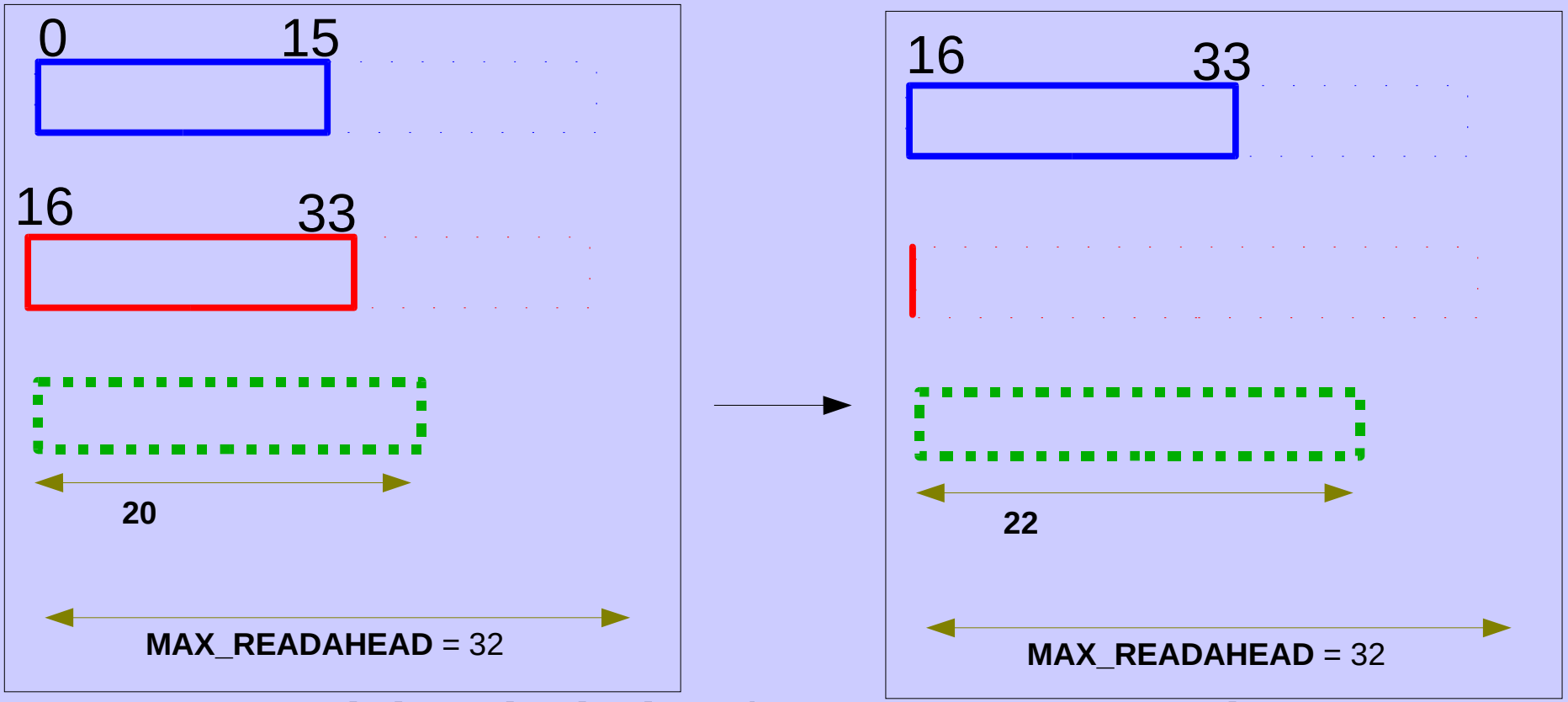
Case 3: read request for page 10



Note: **next window size** increments by 2

2.6.0 readahead cont..

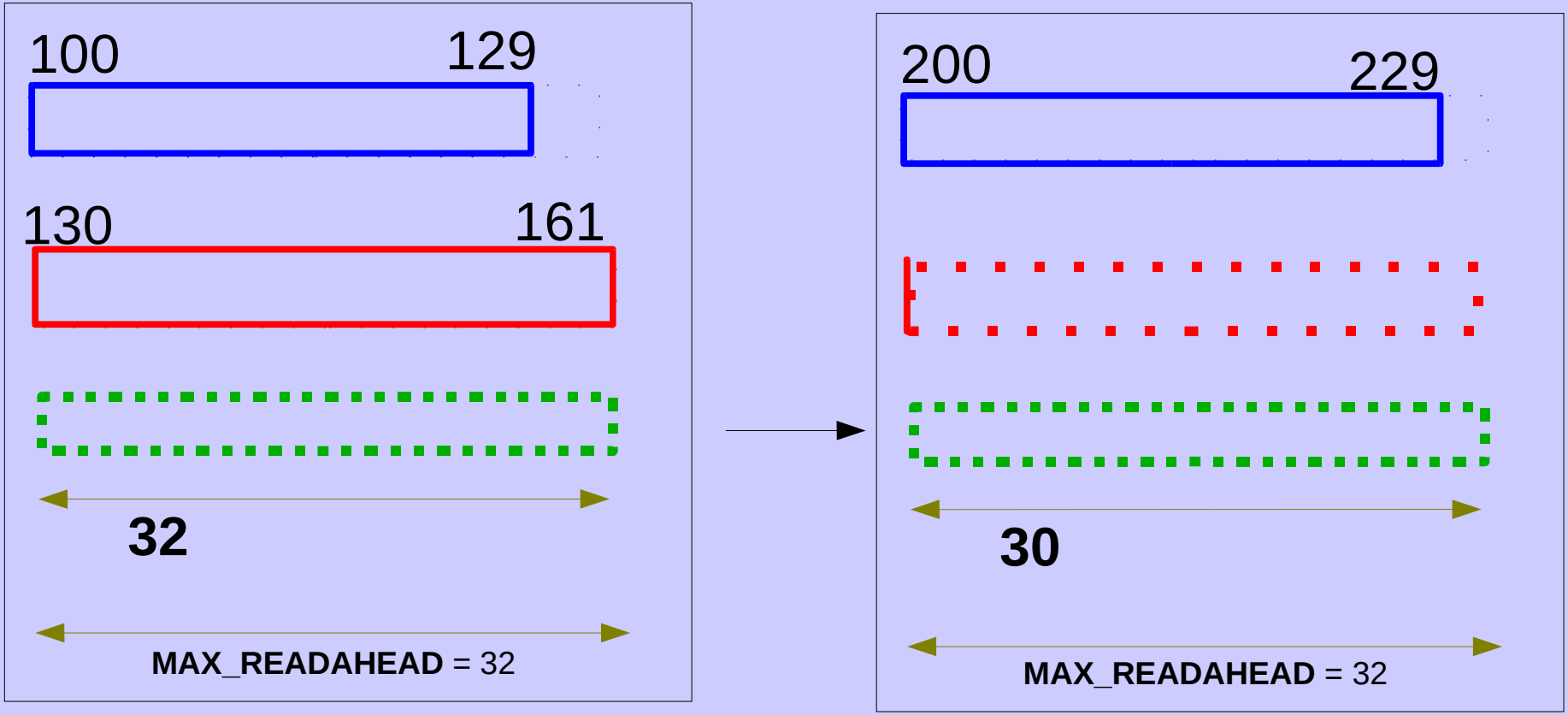
Case 4: read request for page 16



Note: **readahead window** becomes **current window**

2.6.0 readahead cont..

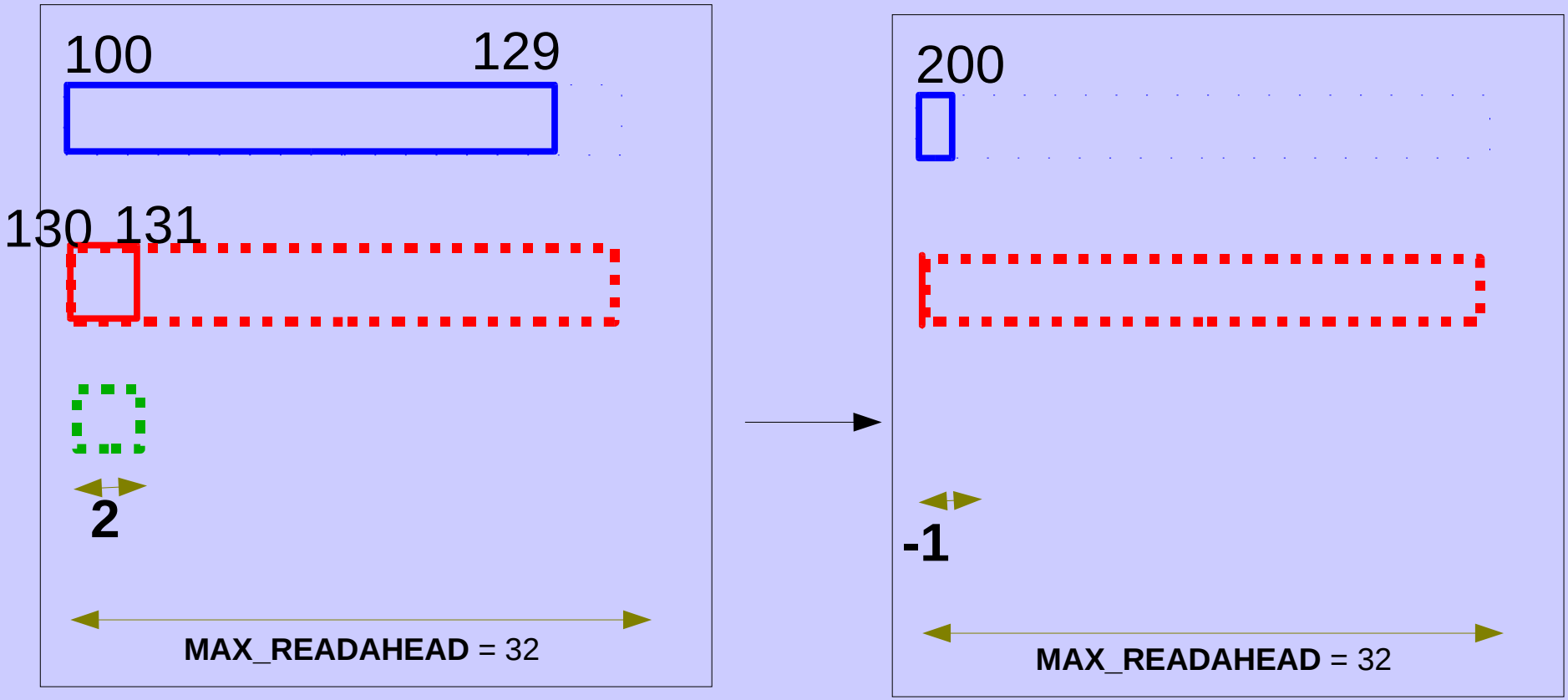
Case 5: read request for page 200



- **Next window size** decremented by 2
- **New current window** brought in with **next widow size** page frames
- **Readahead window** reset

2.6.0 readahead cont..

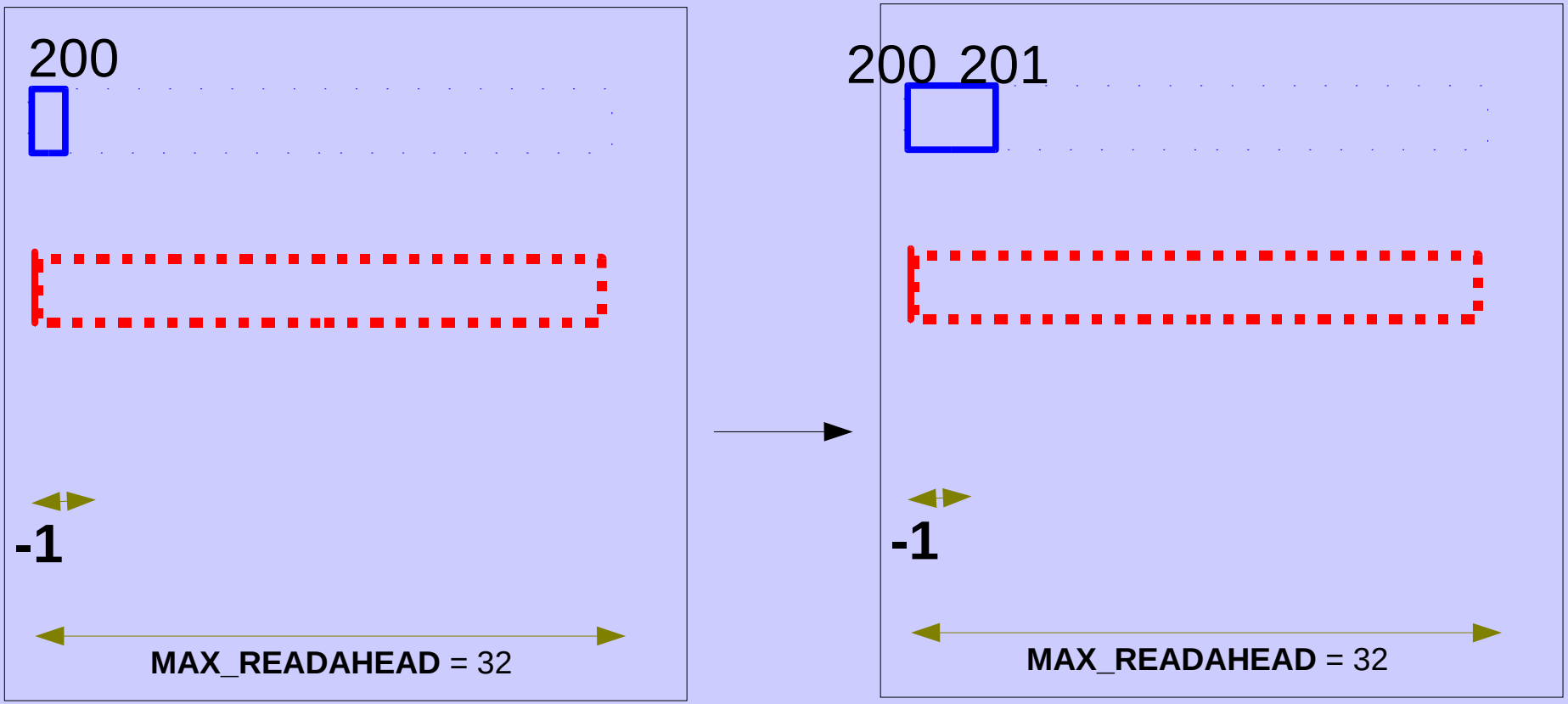
Case 6: read request for page 200



- Reading ahead stopped
- **Readahead window** reset
- Only the requested page brought in the **current window**

2.6.0 readahead cont..

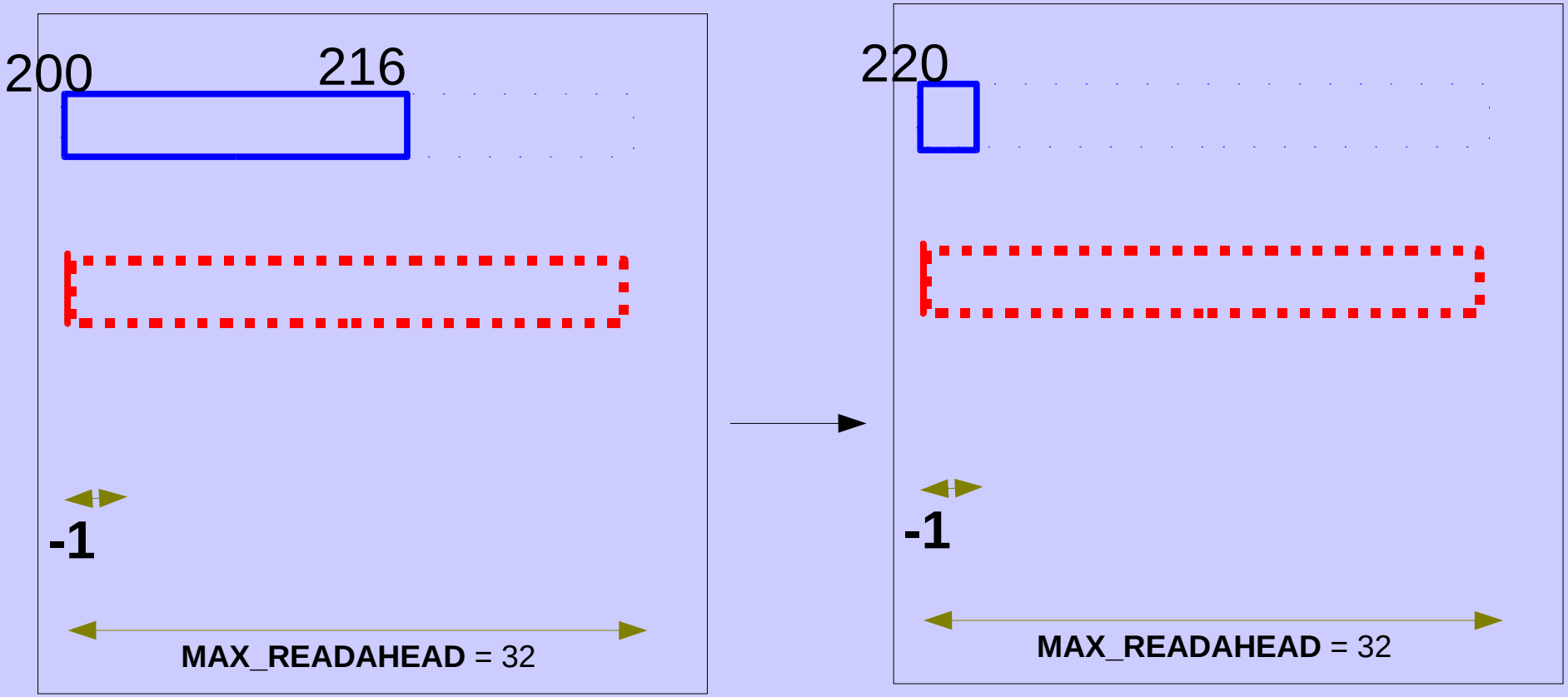
Case 7: read request for page 201



- Reading ahead stopped
- Only the requested page brought in the **current window**

2.6.0 readahead cont..

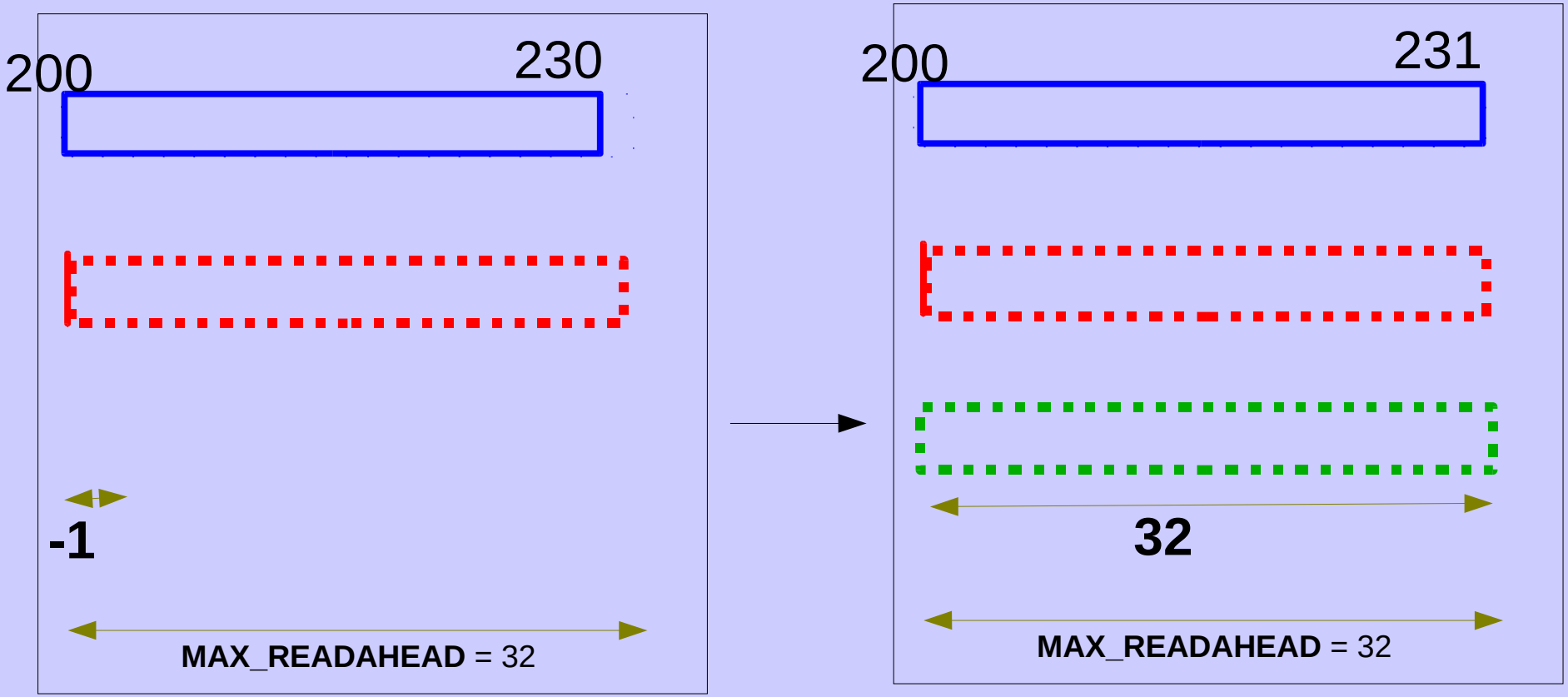
Case 8: read request for page 220



- Reading ahead stopped
- **current window** shrinks down to one page frame :-(

2.6.0 readahead cont..

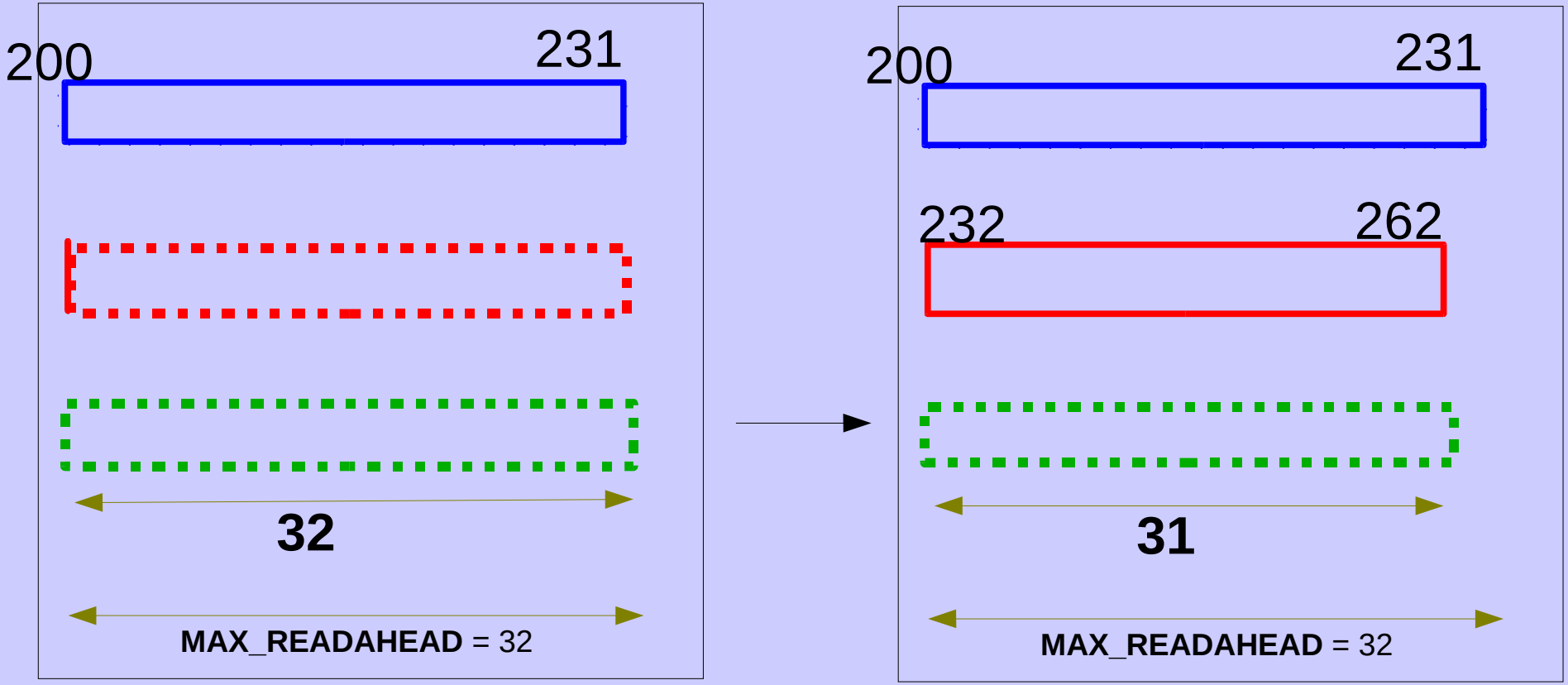
Case 9: read request for page 231



- Reading ahead mode resumed!
- **next readahead size** set to **MAX_READAHEAD!**

2.6.0 readahead cont..

Case 10: read request for page 210. Pages 232 to 263 are already in the page cache



- Since all the **readahead window** pages already reside in page cache shrink **next readahead size** by 1!
- Populate the **readahead window**

Problems of 2.6.0 readahead

- Page-0 problem.
- Page-cache-hit on first read problem.
- Lots of wasted pages.

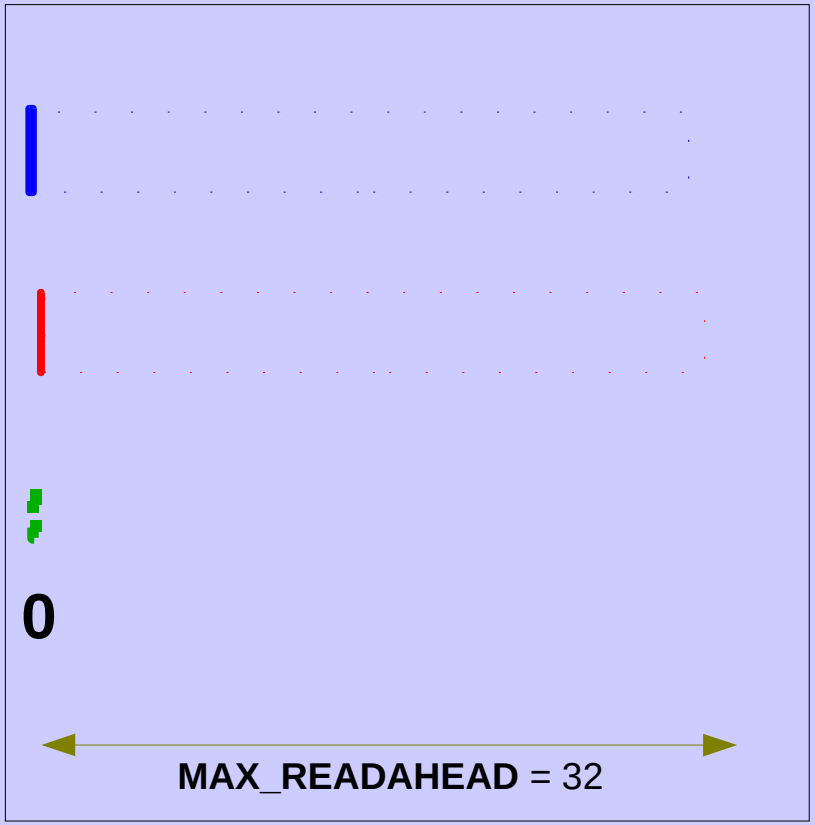
Problem-1 of 2.6.0 readahead

Consider the following file read pattern. 4-Page random read requests.

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31

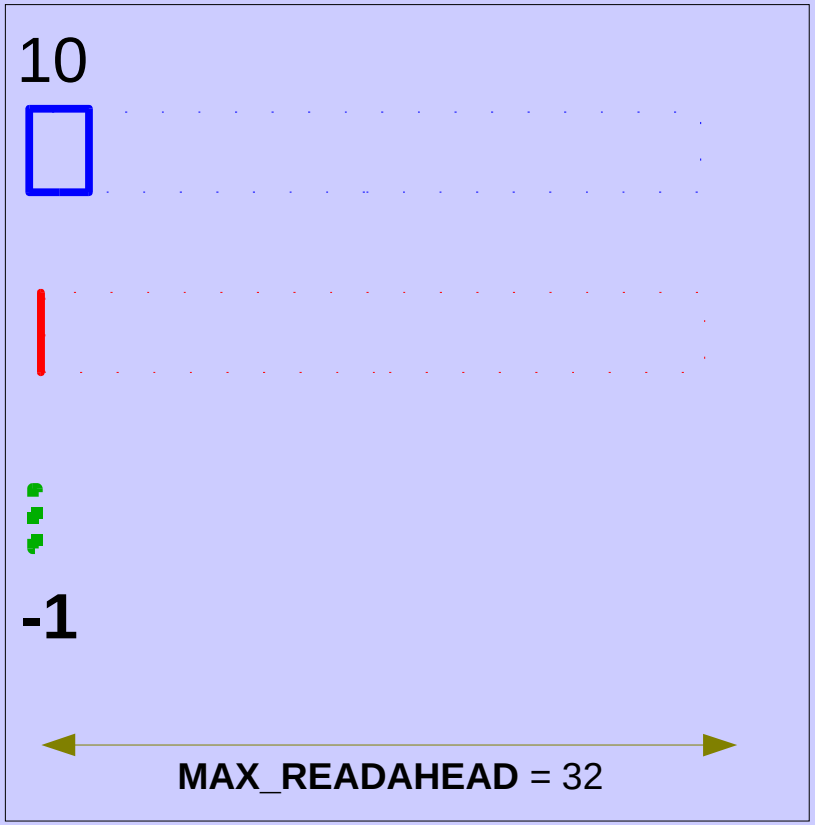
STEP 1: Initial state

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



STEP 2: Read page 10

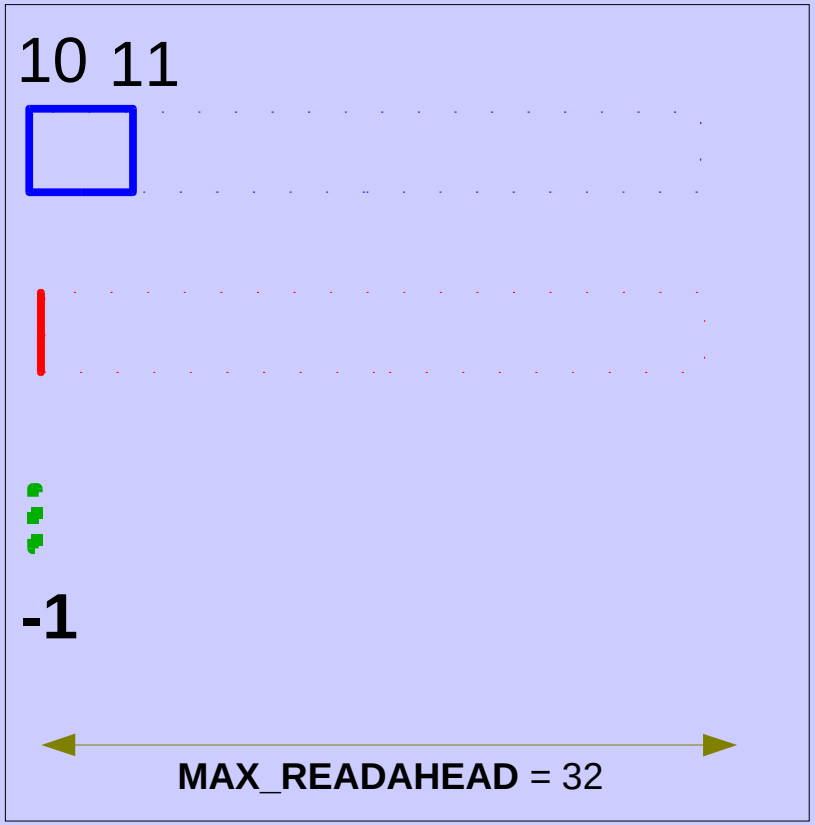
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 10

STEP 3: Read page 11

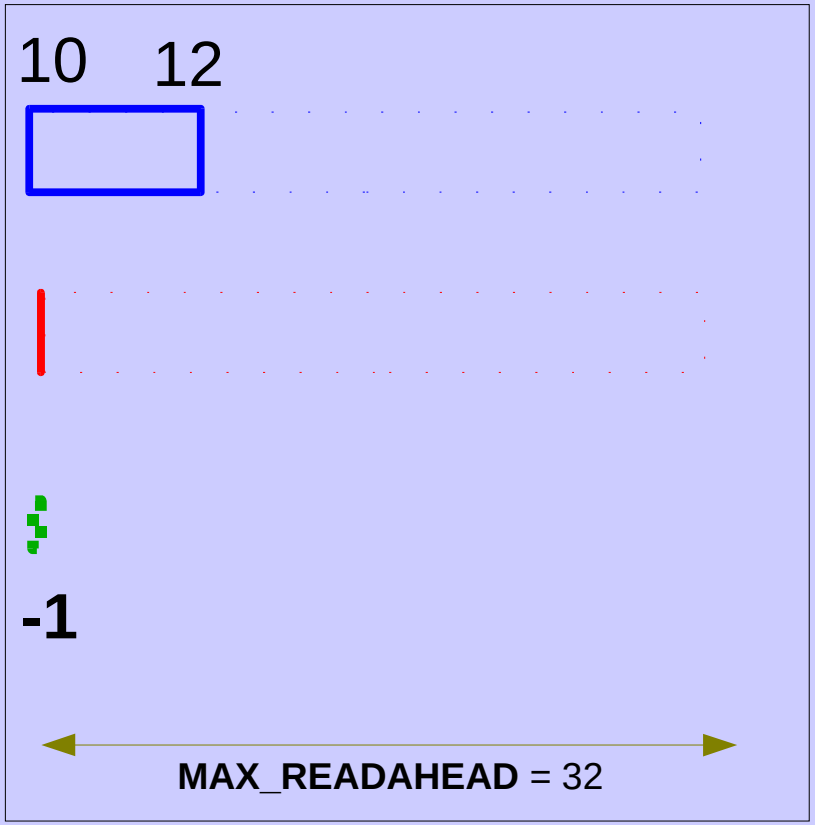
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 11

STEP 4: Read page 12

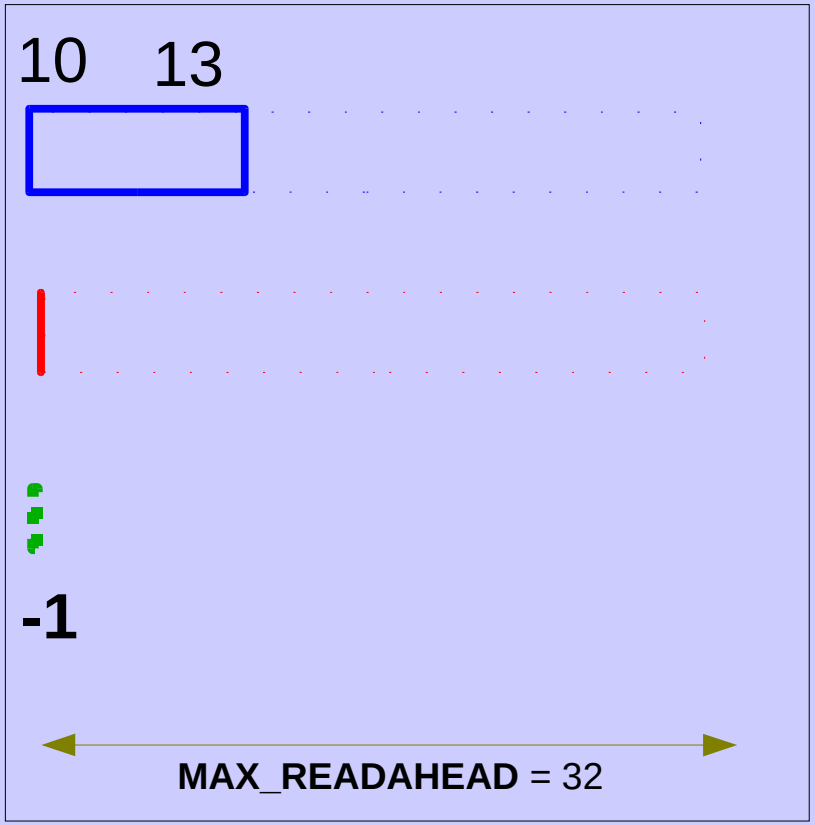
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 12

STEP 5: Read page 13

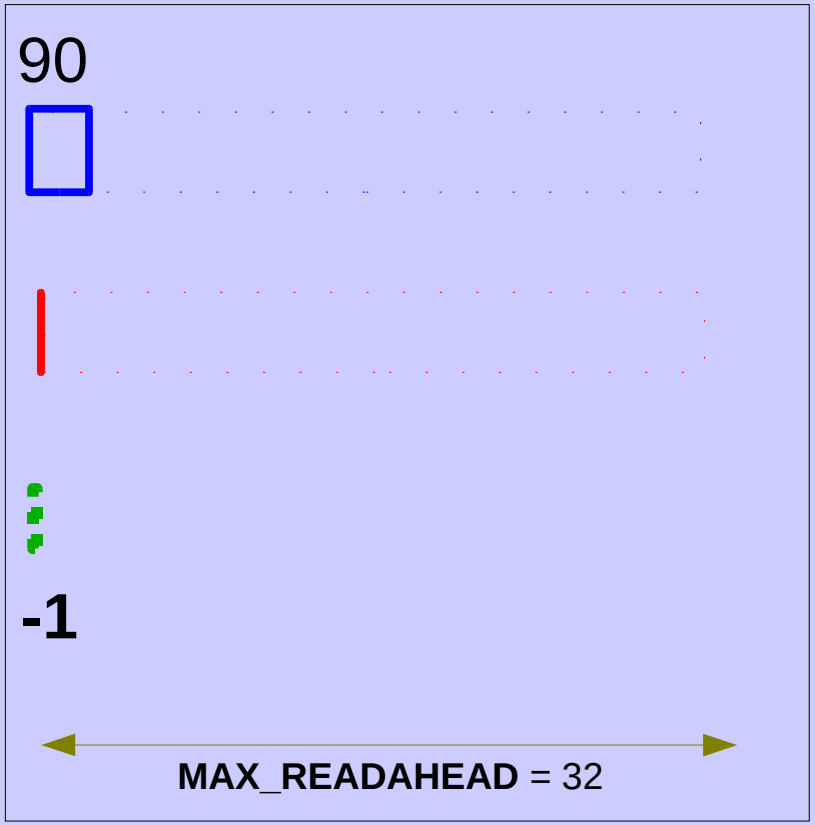
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 13

STEP 6: Read page 90

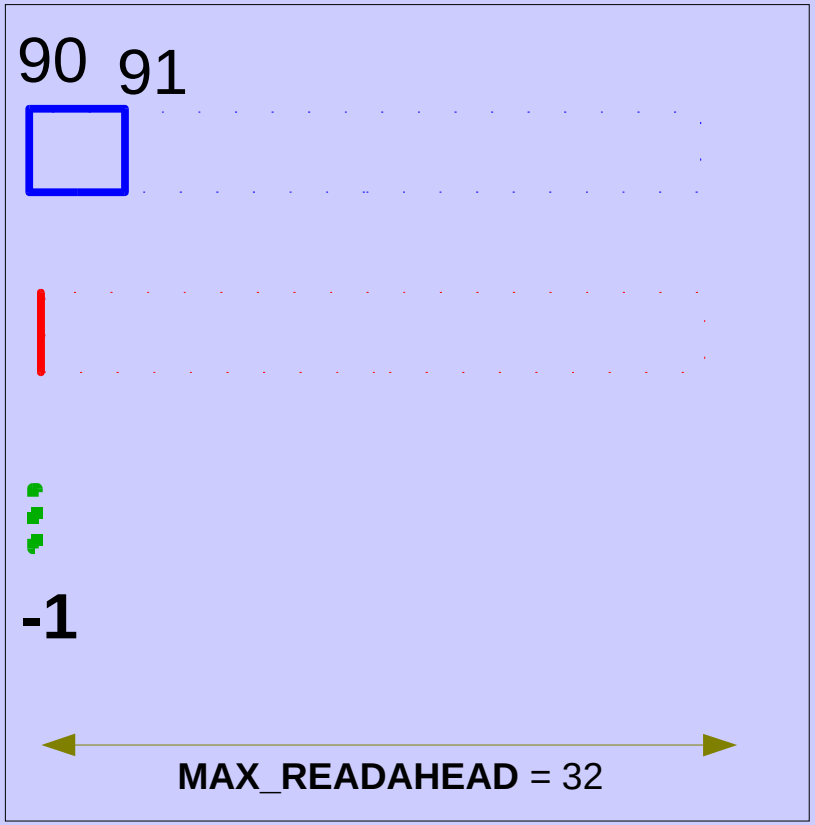
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 90

STEP 7: Read page 91

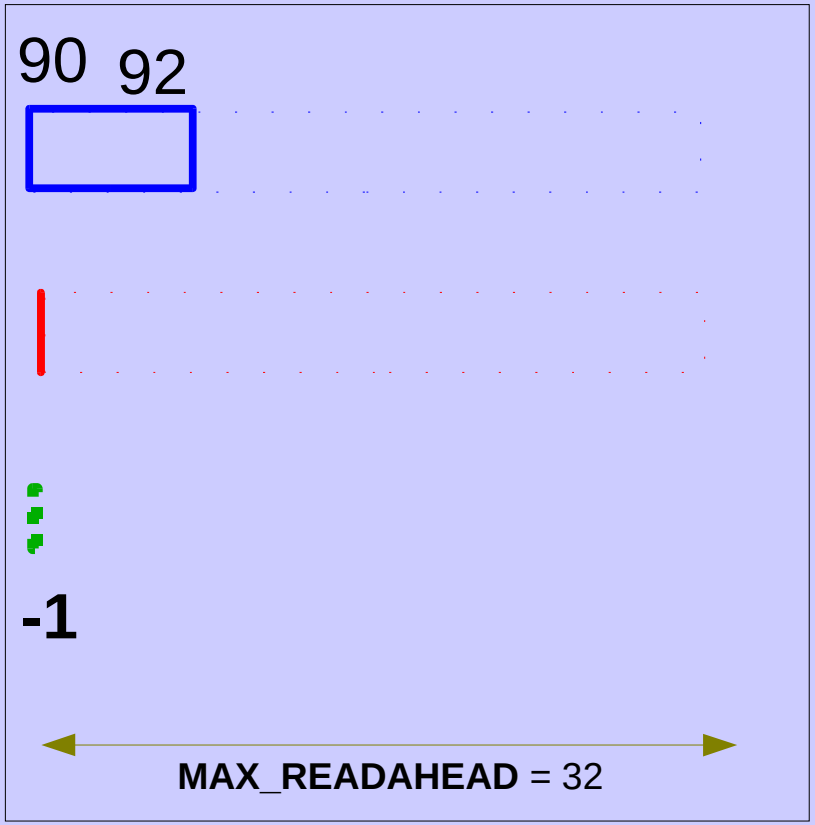
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 91

STEP 8: Read page 92

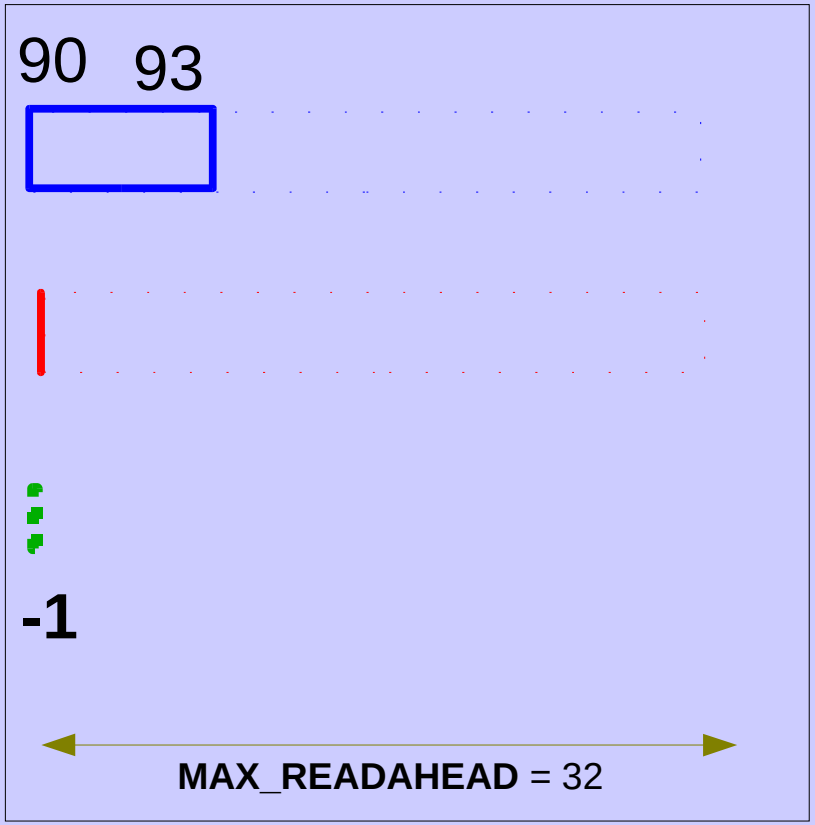
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 92

STEP 9: Read page 93

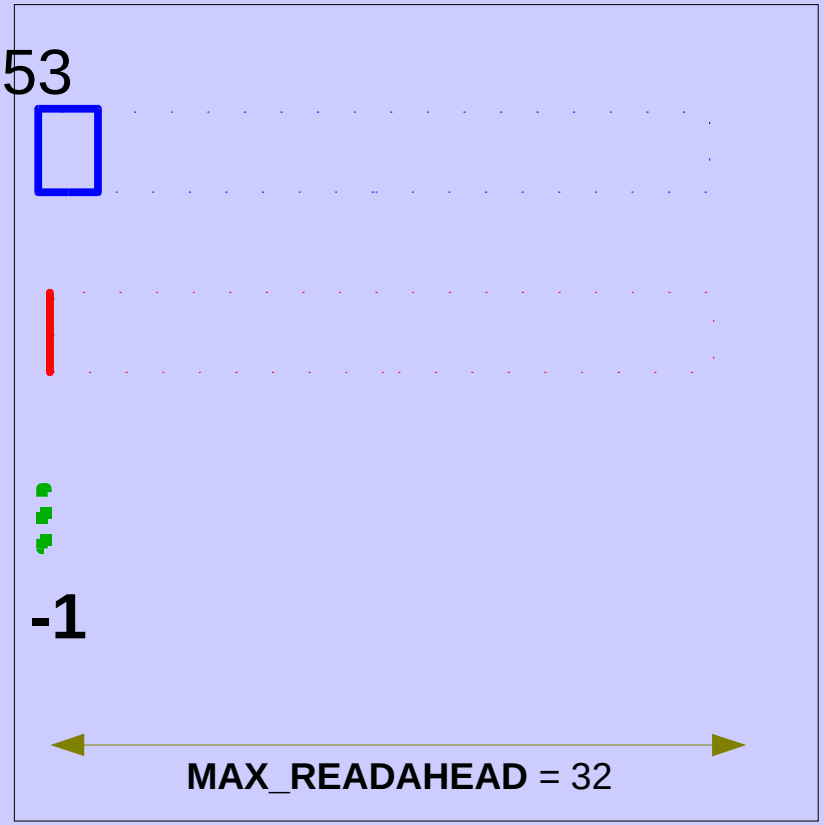
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 93

STEP 10: Read page 53

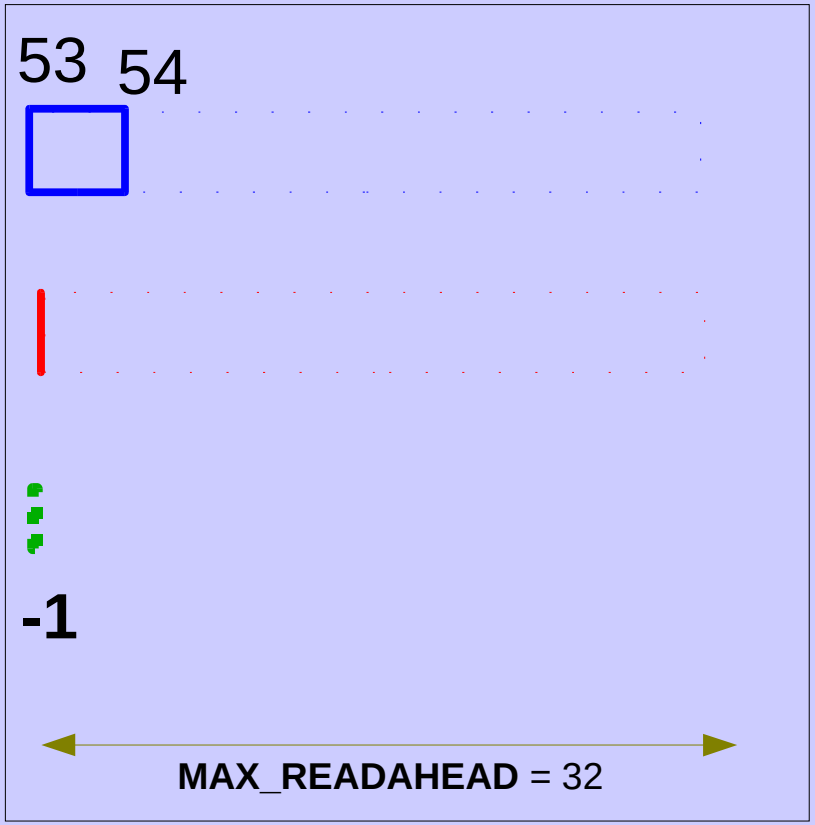
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 53

STEP 11: Read page 55

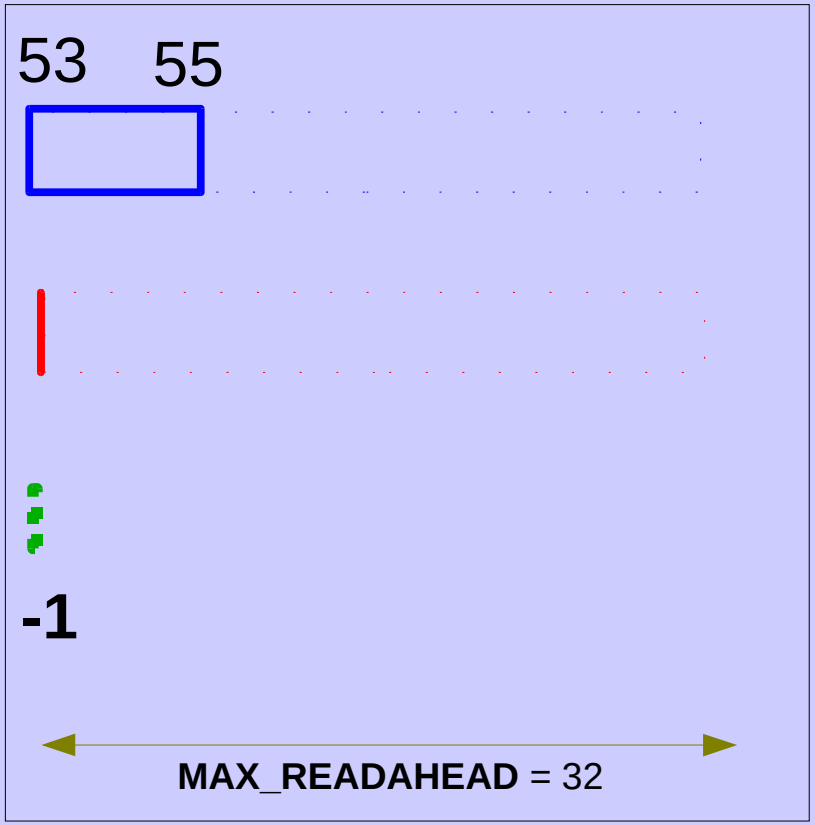
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 55

STEP 12: Read page 56

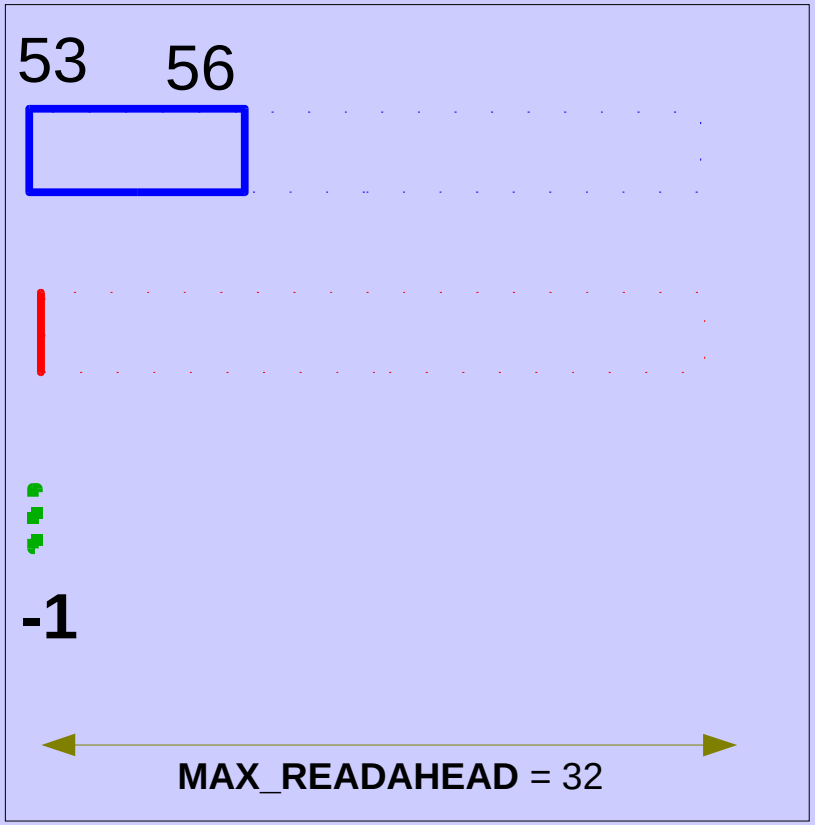
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 56

STEP 13: Read page 56

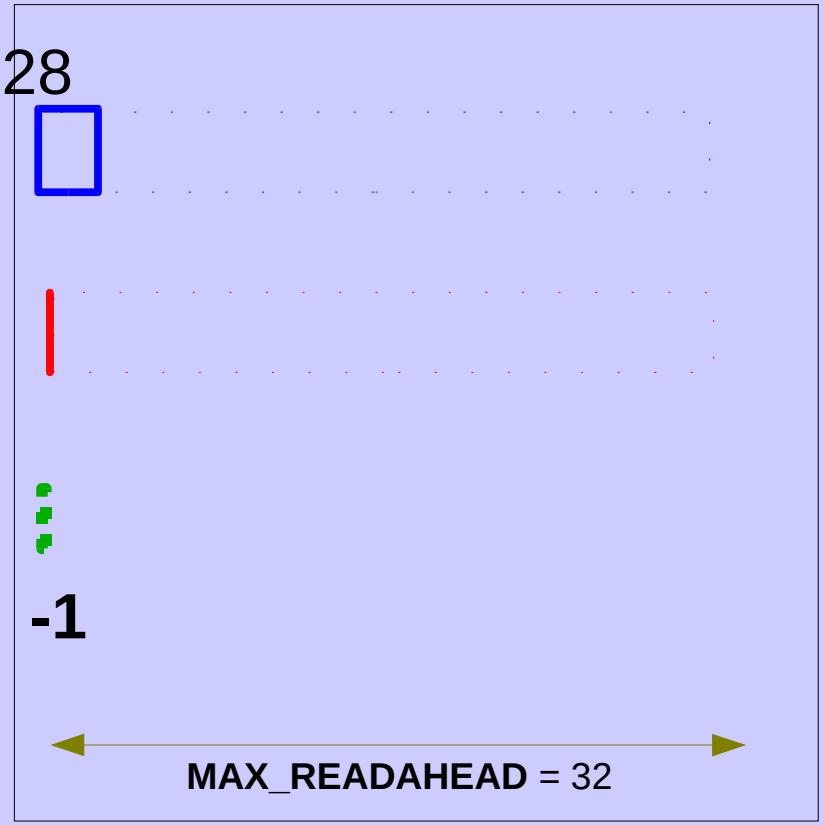
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 56

STEP 14: Read page 28

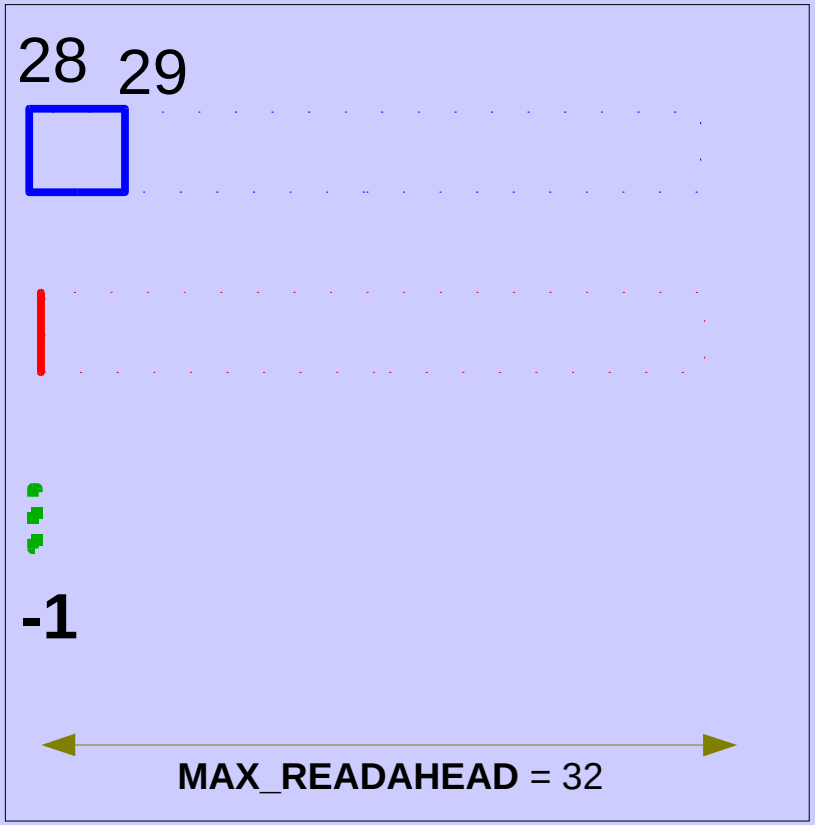
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 28

STEP 15: Read page 29

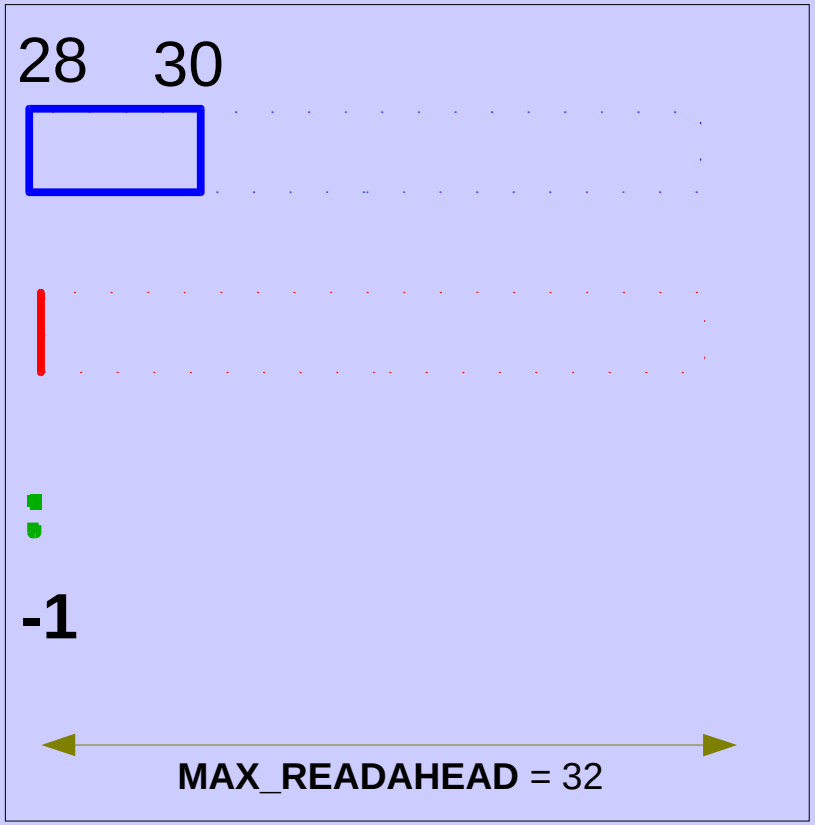
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 29

STEP 16: Read page 30

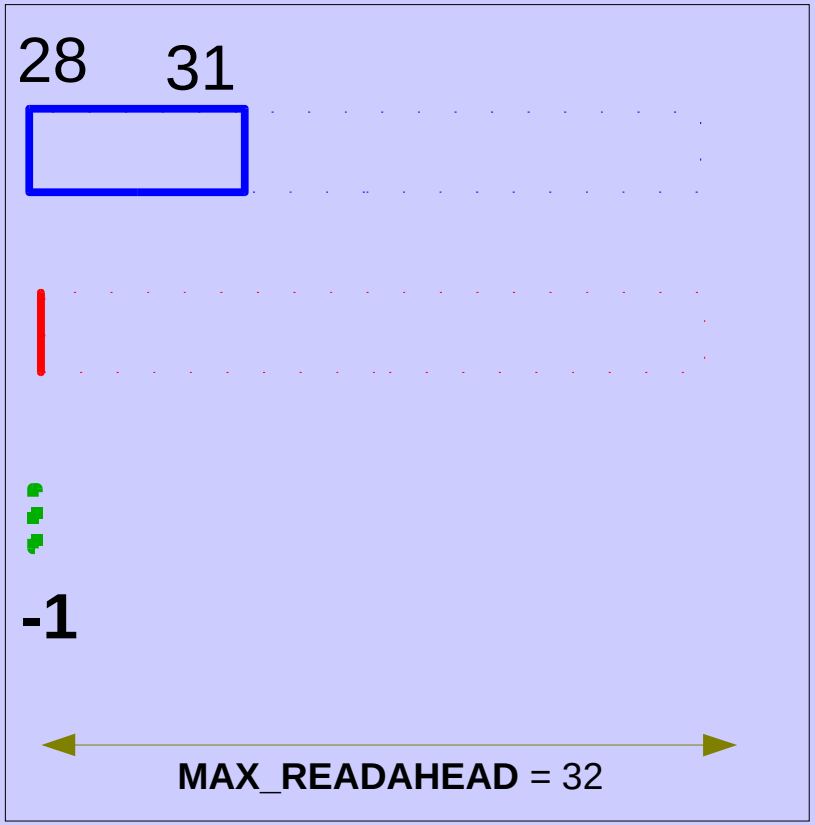
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 30

STEP 18: Read page 31

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



- NOTE: synchronous read of page 31

Where is the bug?

- **If file is accessed from offset 0, file access is sequential?**
- **In the readahead off-mode any miss in contiguity is penalized heavily by resetting the current window.**

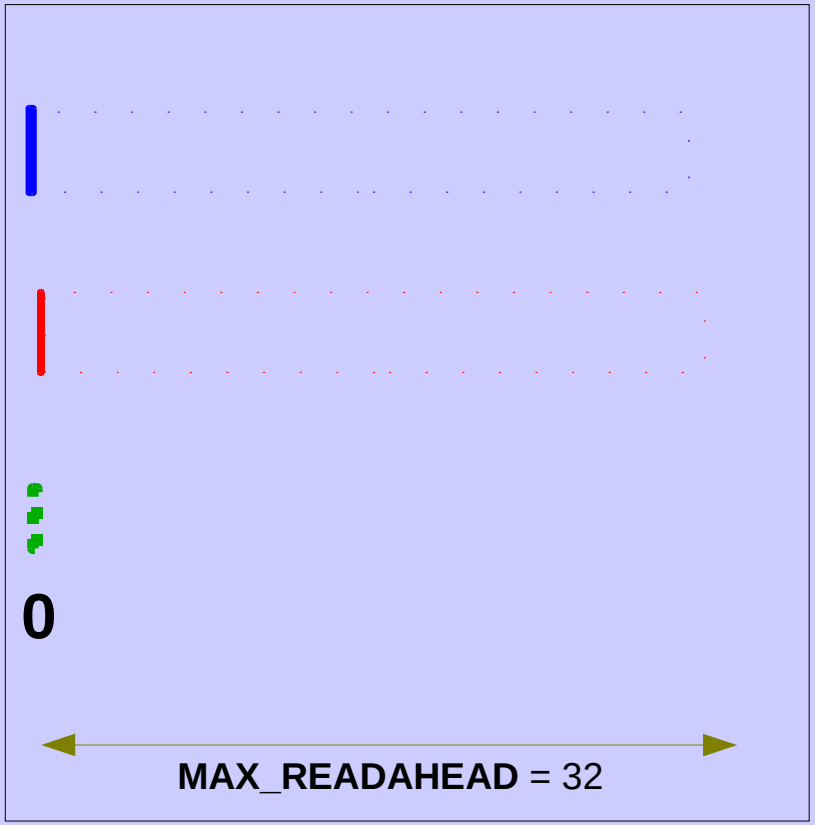
Problem-2 of 2.6.0 readahead

- If the pages read in the **current-window** the first time, all reside in the page-cache reada-head is turned-off!

Consider the following file read pattern. 4-Page random read requests.

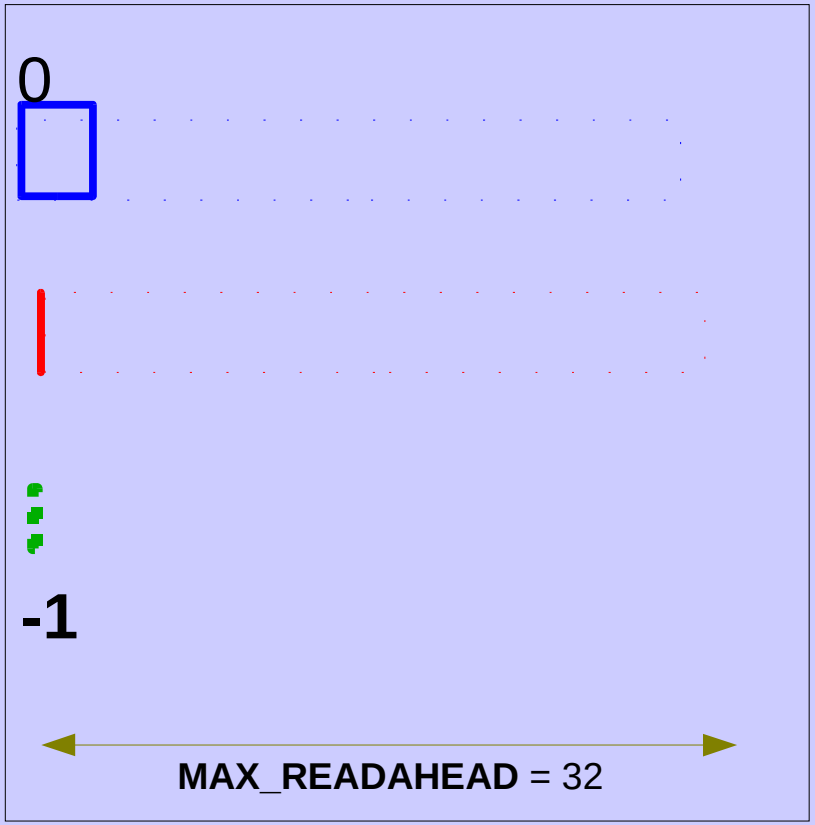
0, 1, 2, 3
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31

STEP 1: Initial state



0
1
2
3
4
5
6
7
8
...
...
15

STEP 2: Read page 0



0
1
2
3
4
5
6
7
8
...
...
15

Problem-3 of 2.6.0 readahead

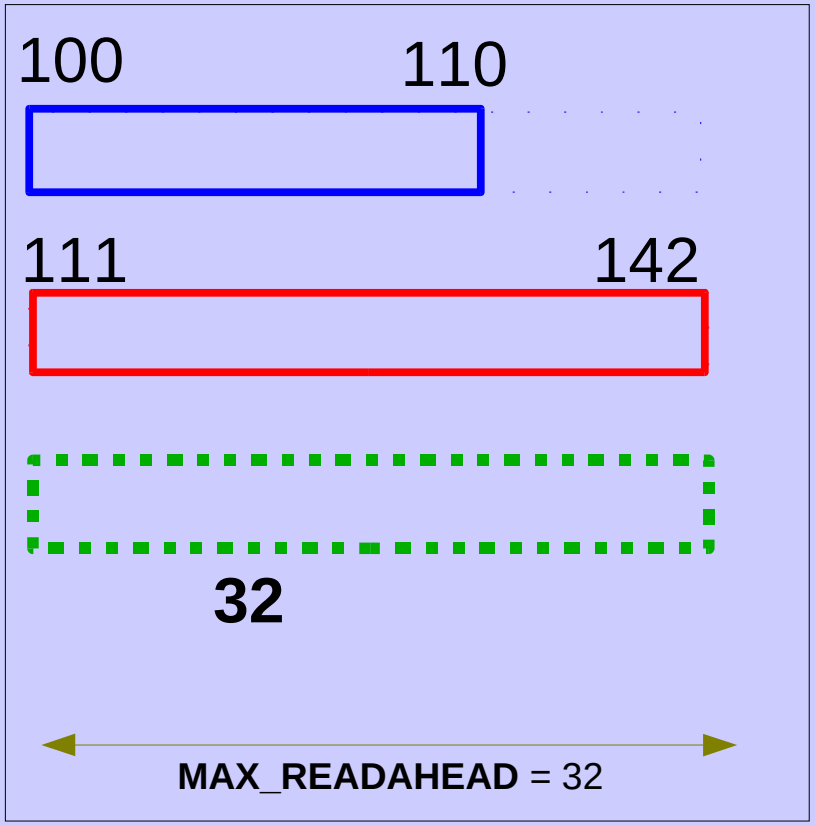
- Lots of wasted pages, wasted bandwidth, wasted resources!

Consider the following file read pattern. 4-Page random read requests.

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31

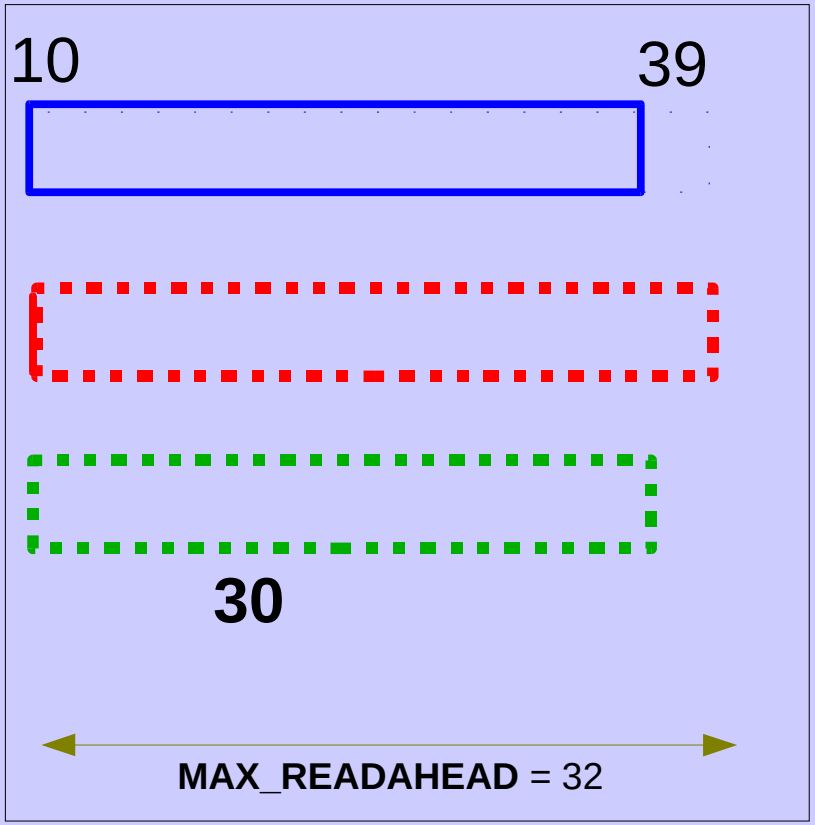
STEP 1: Initial state

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



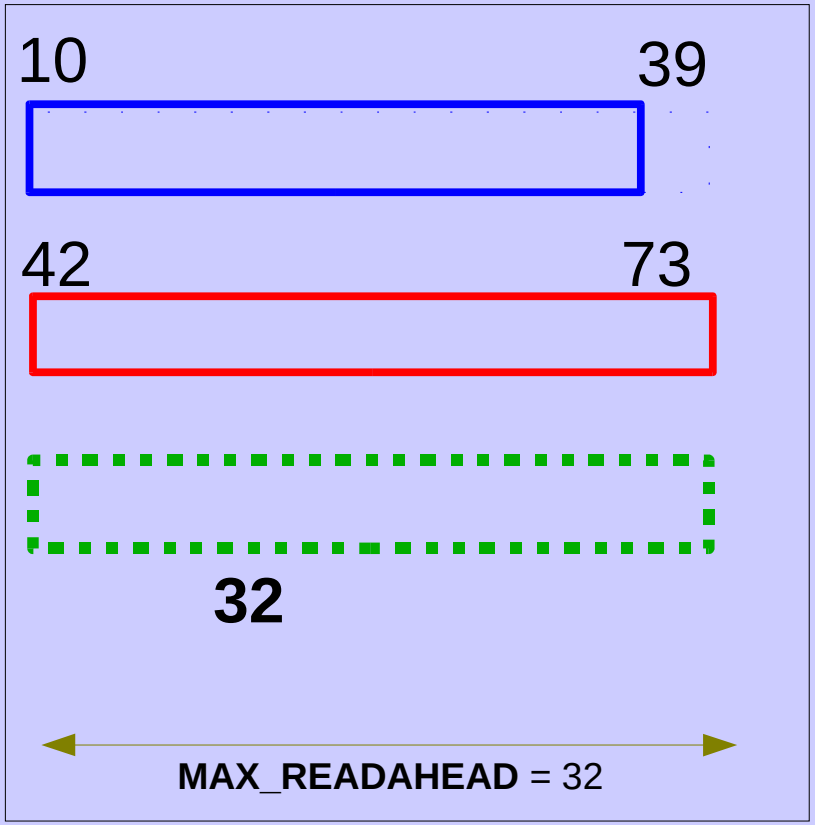
STEP 1: request for page 10

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



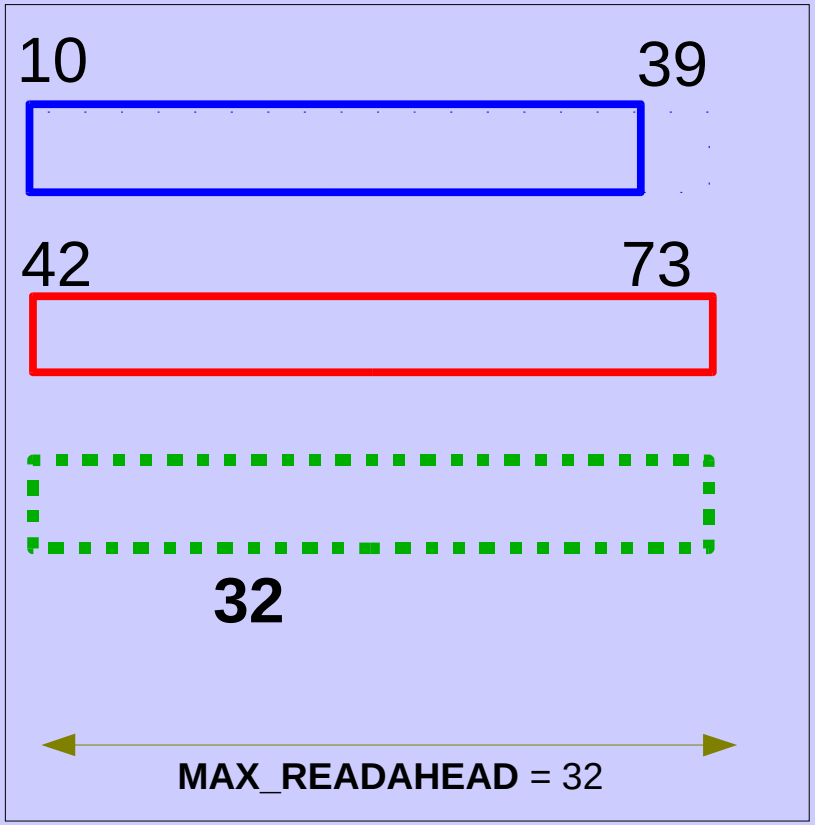
STEP 2: request for page 11

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



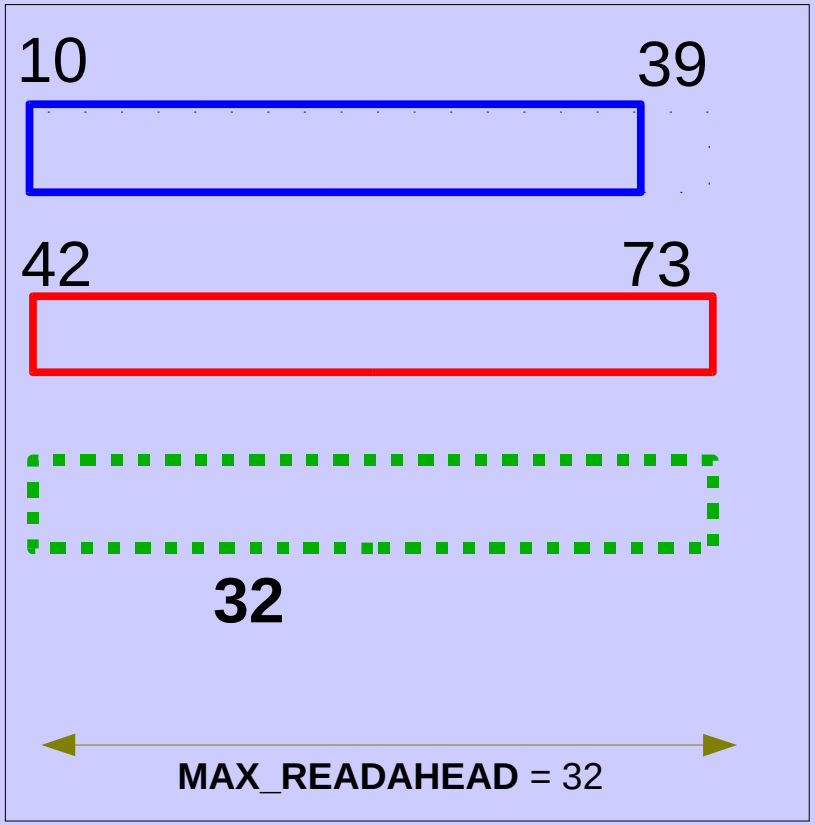
STEP 3: request for page 12

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



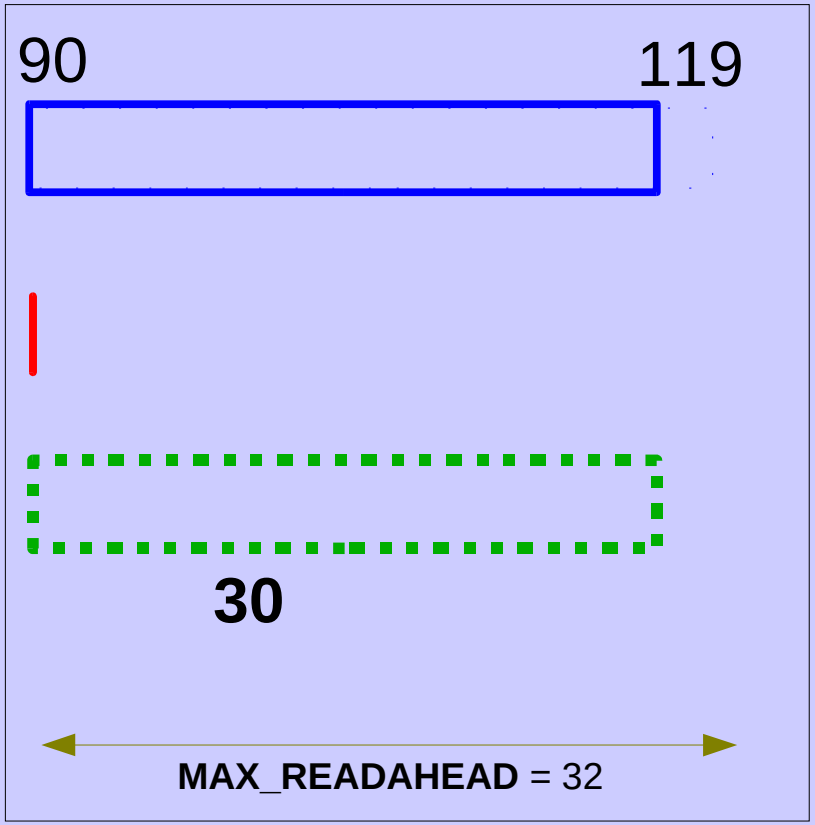
STEP 4: request for page 13

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



STEP 5: request for page 90

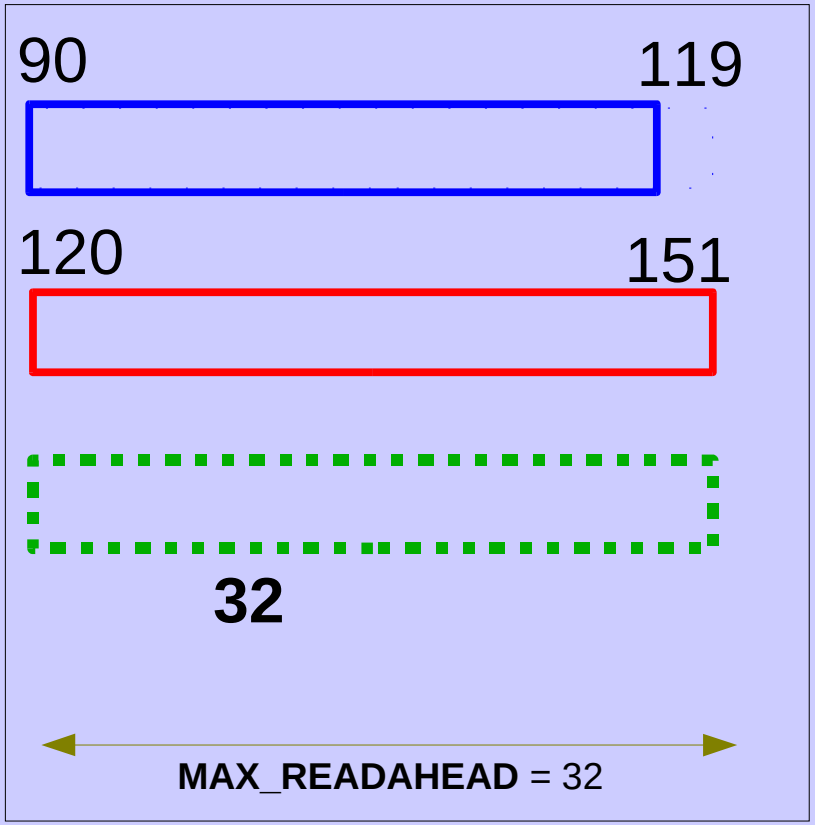
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



WASTED 58 pages!

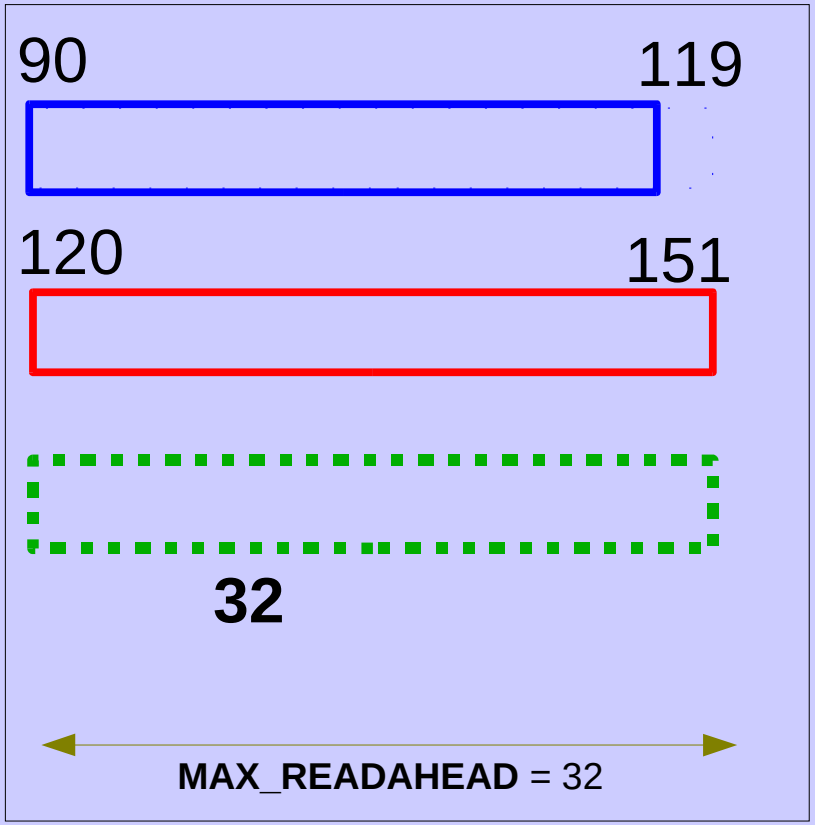
STEP 6: request for page 91

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



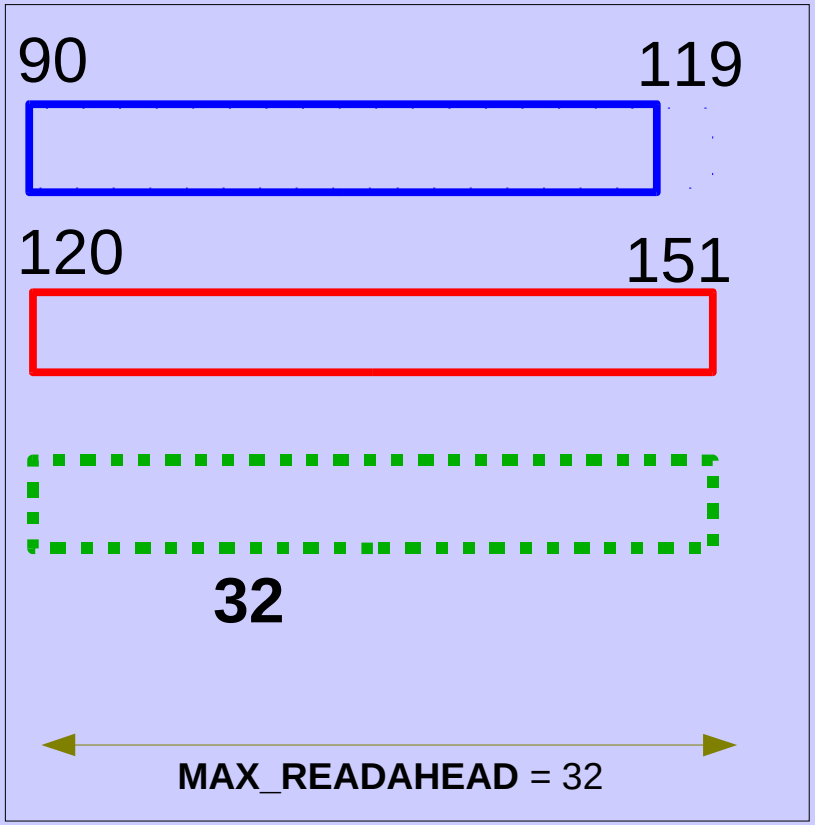
STEP 7: request for page 92

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



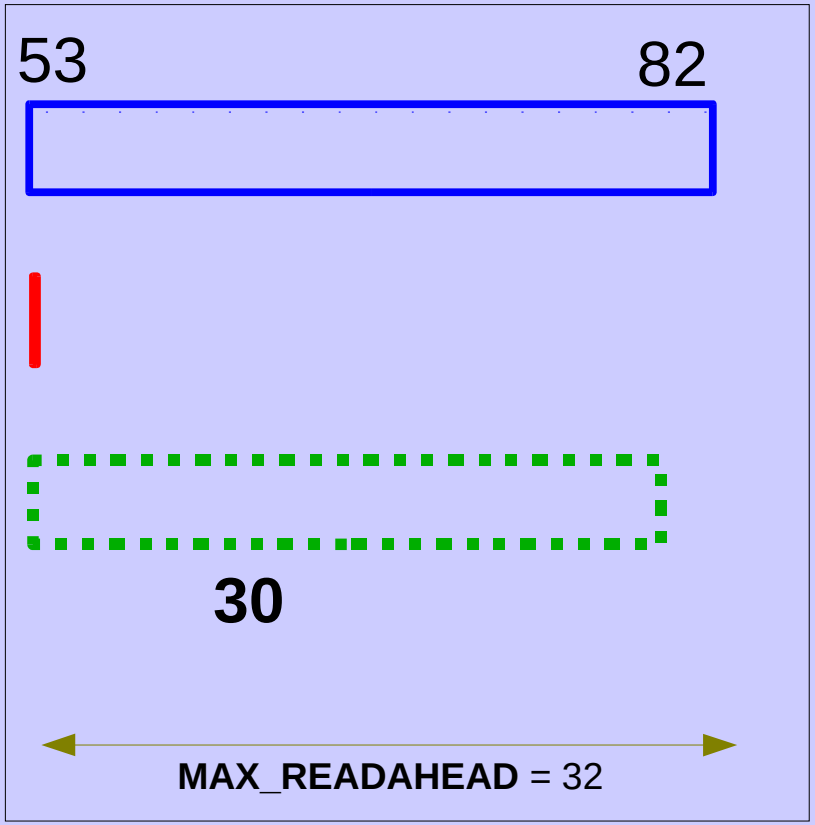
STEP 8: request for page 93

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



STEP 9: request for page 53

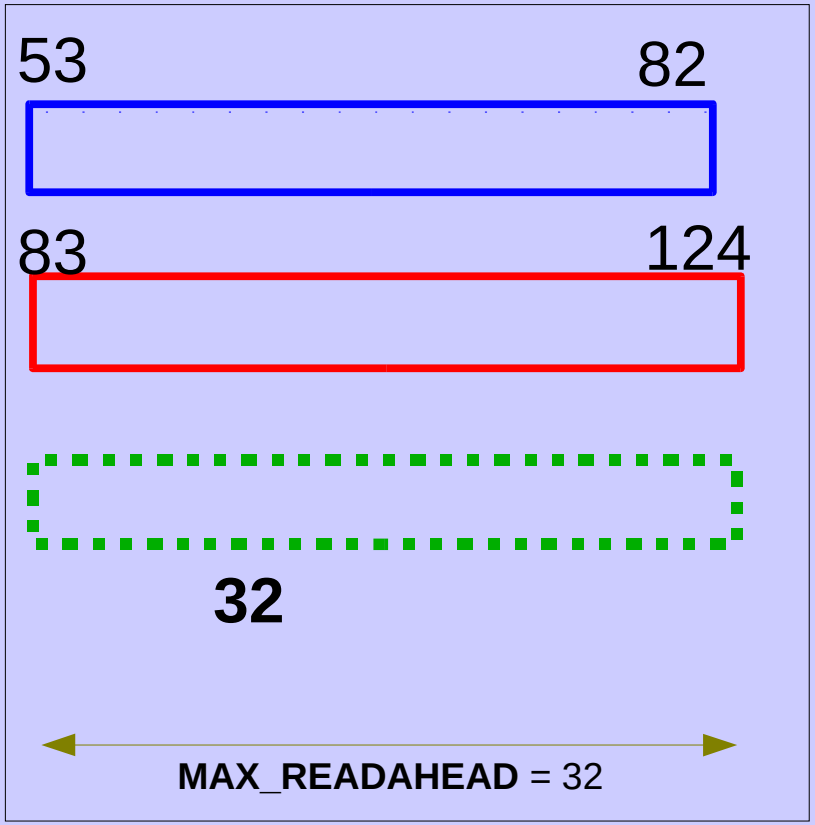
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



WASTED 58 pages!

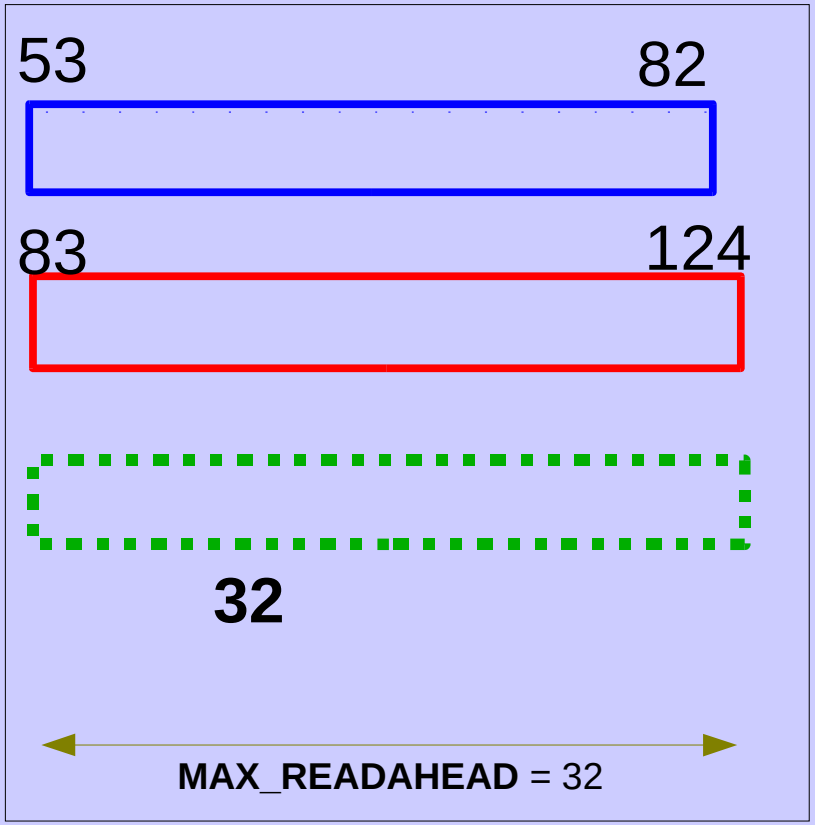
STEP 10: request for page 54

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



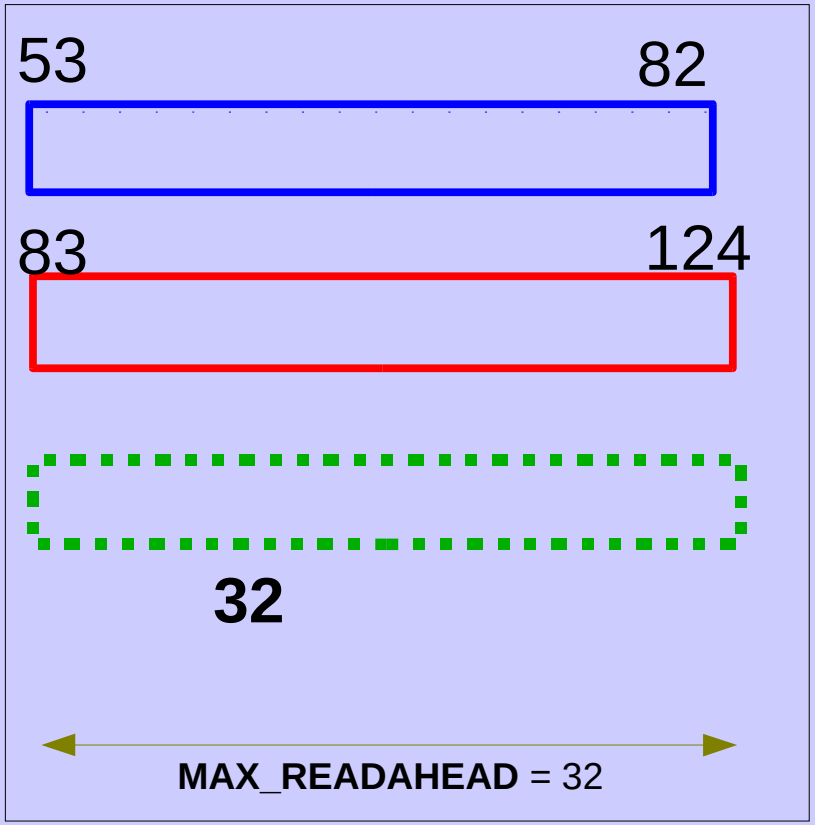
STEP 11: request for page 55

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



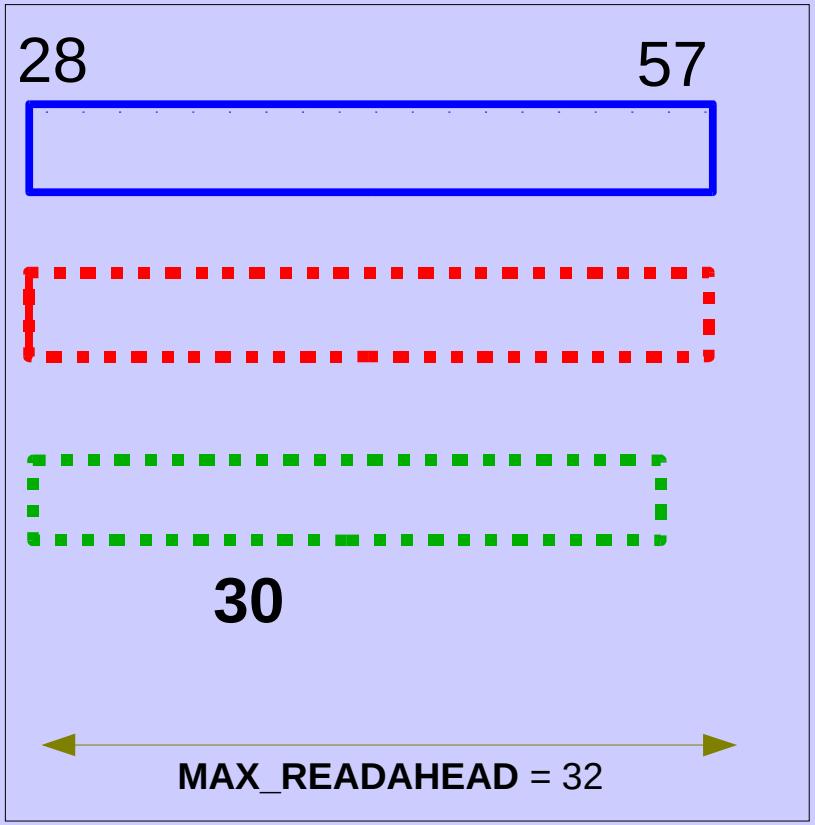
STEP 12: request for page 56

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



STEP 13: request for page 28

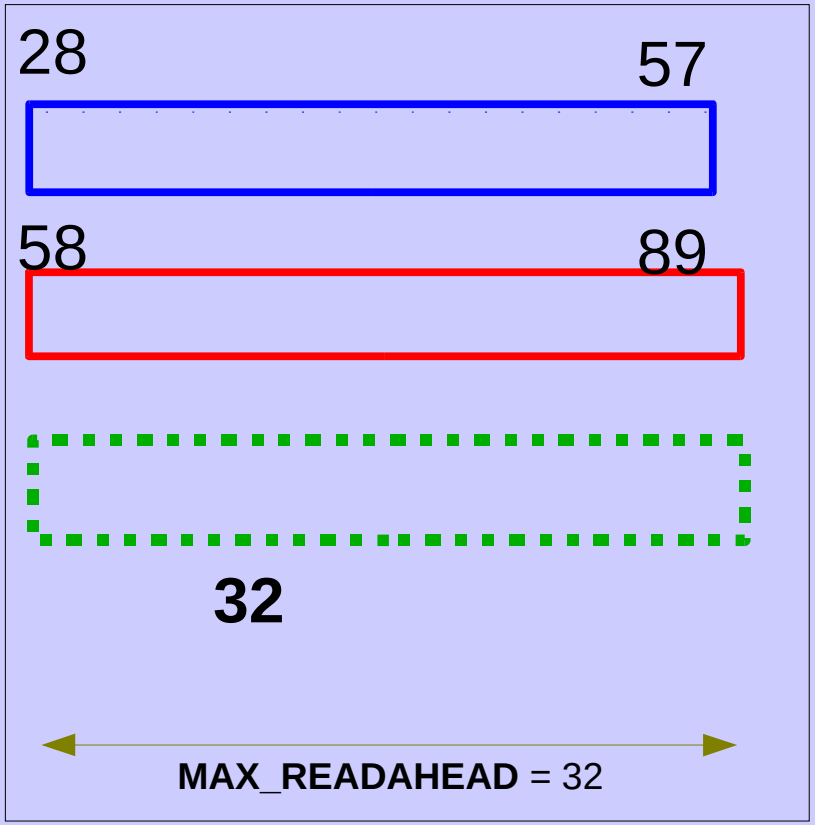
10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



WASTED 58 pages!

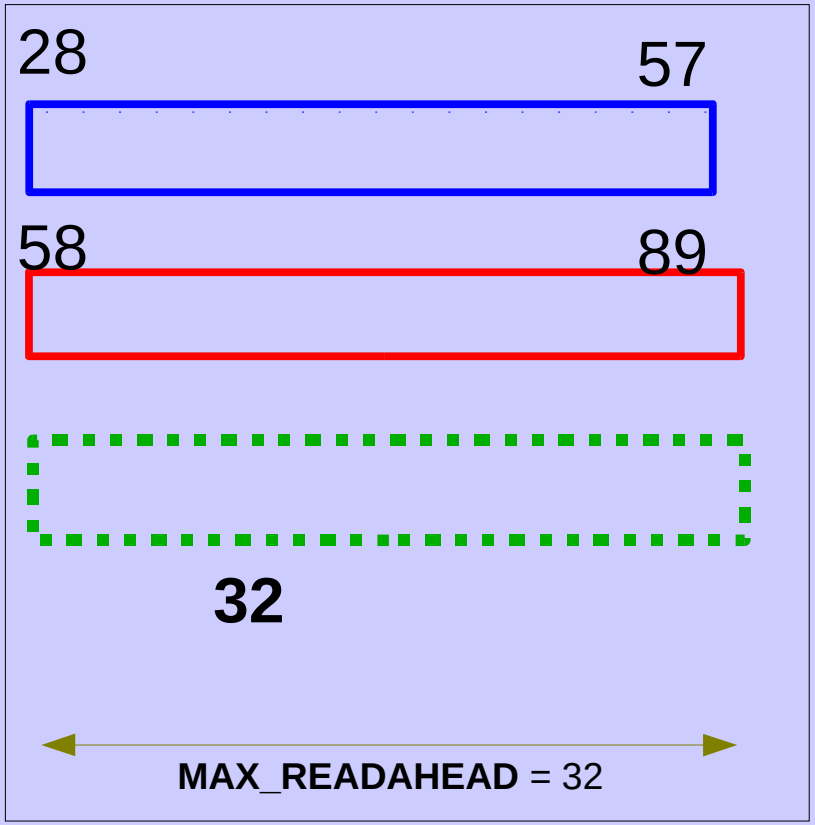
STEP 14: request for page 29

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



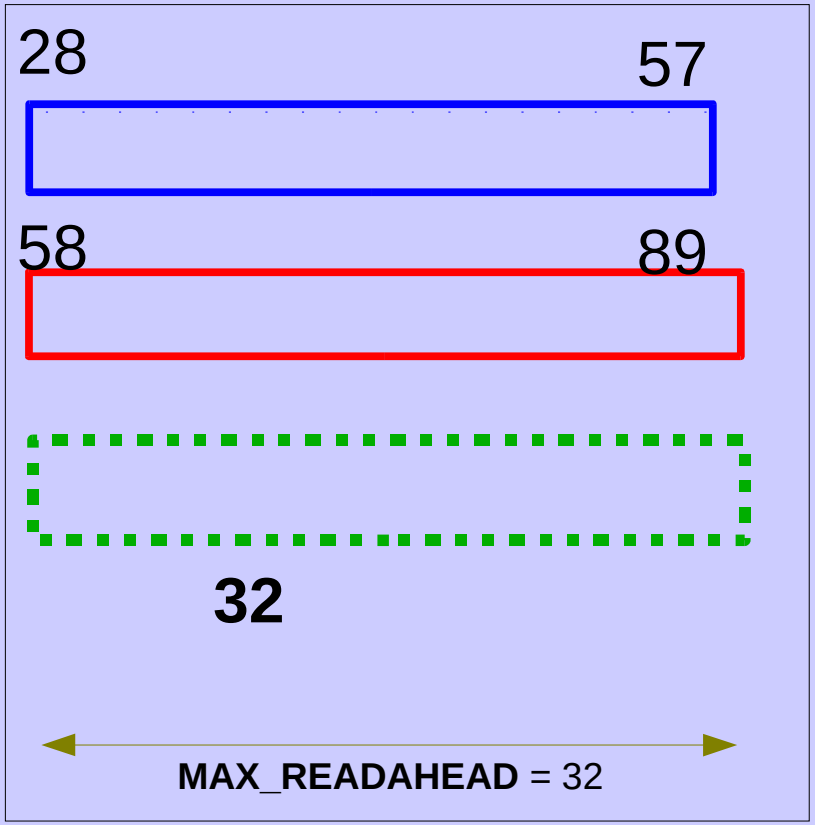
STEP 15: request for page 30

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



STEP 16: request for page 31

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31



2.6.7 Readahead algorithm

- No penalty if reads don't start from page 0
- Readahead mode is not switched off for the first read request even if all the pages are found in the page cache.
- The average size of read request is tracked.
- The average size determines the size of the next current window.
- Only if average size is greater than MAX_READAHEAD, readahead window is populated.

2.6.7 readahead

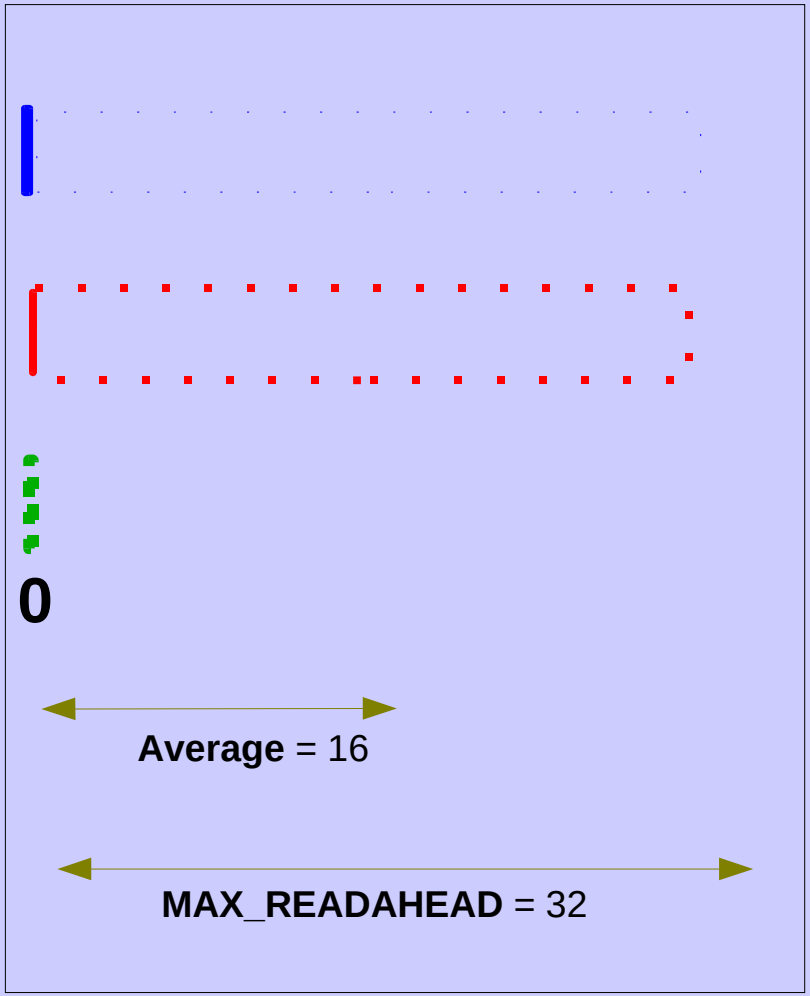
Consider the following file read pattern.

4-Page random read requests.

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73

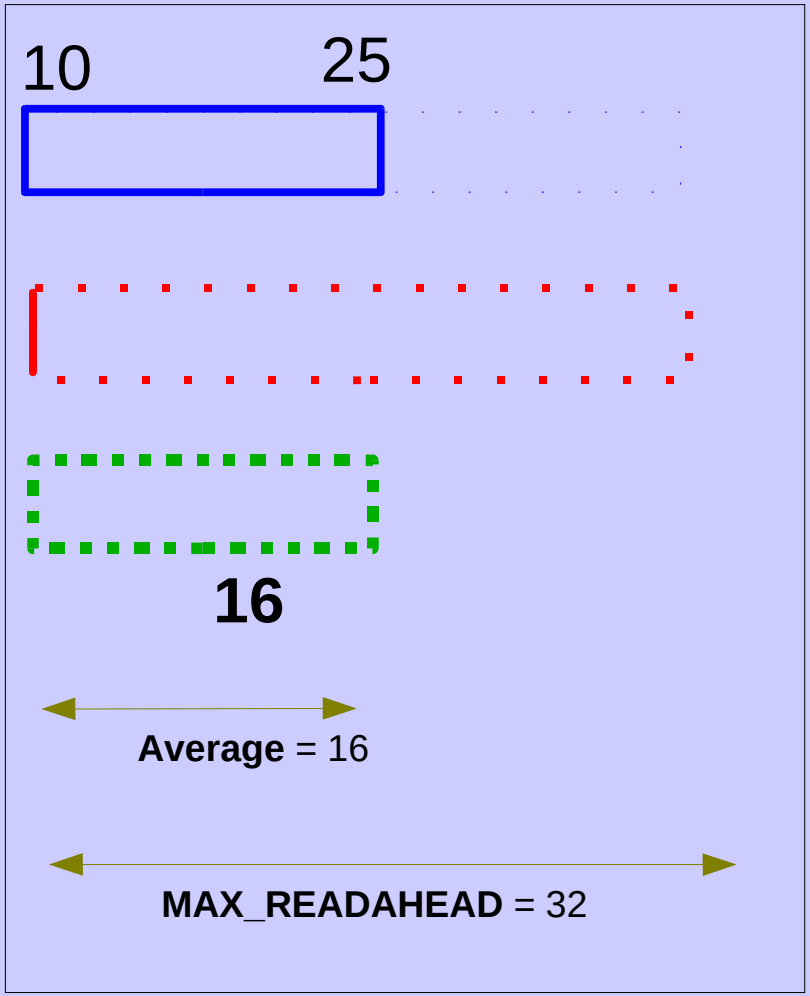
STEP 1: Initial state

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



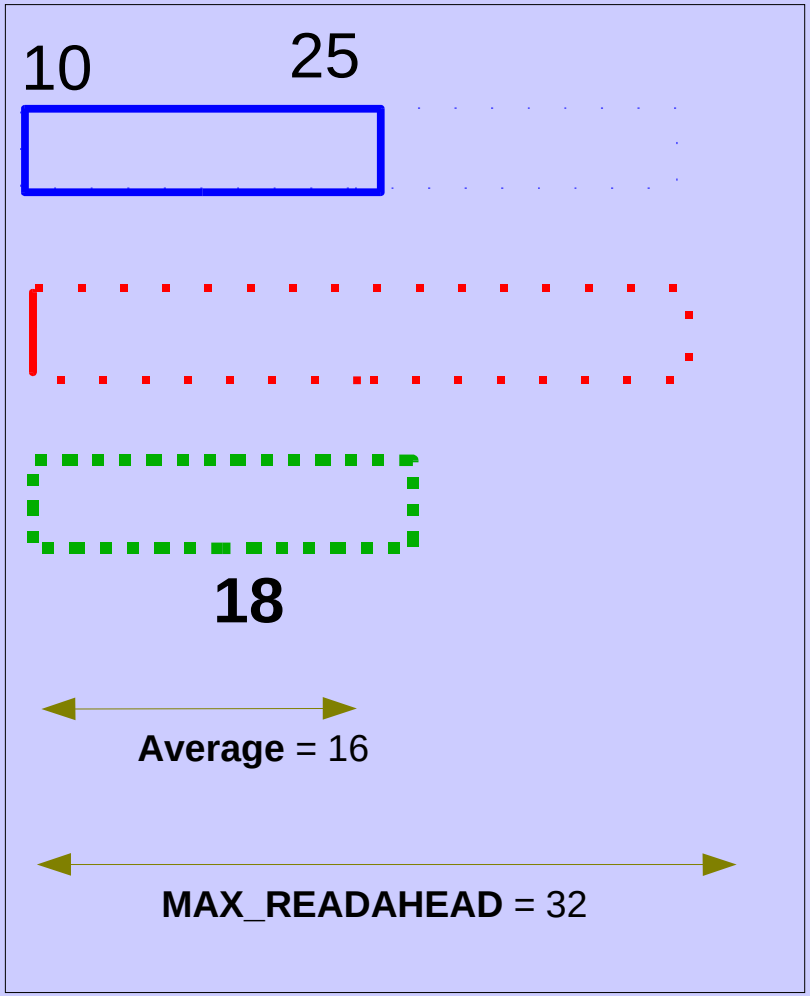
STEP 2: Request for page 10

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



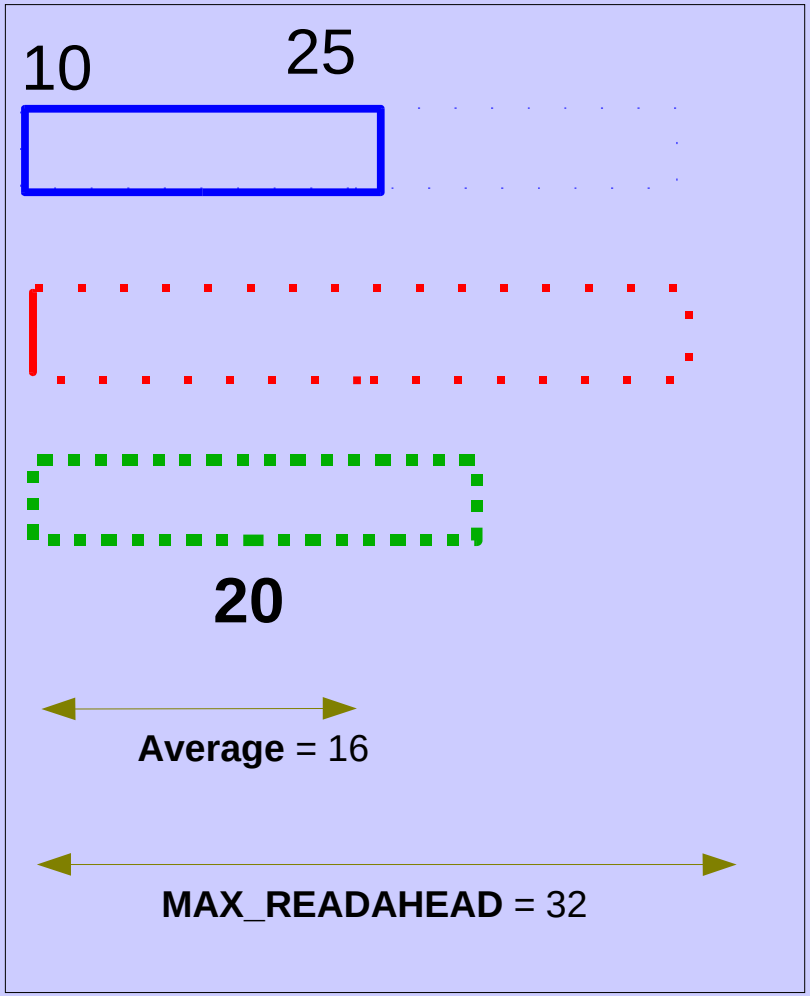
STEP 3: Request for page 11

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



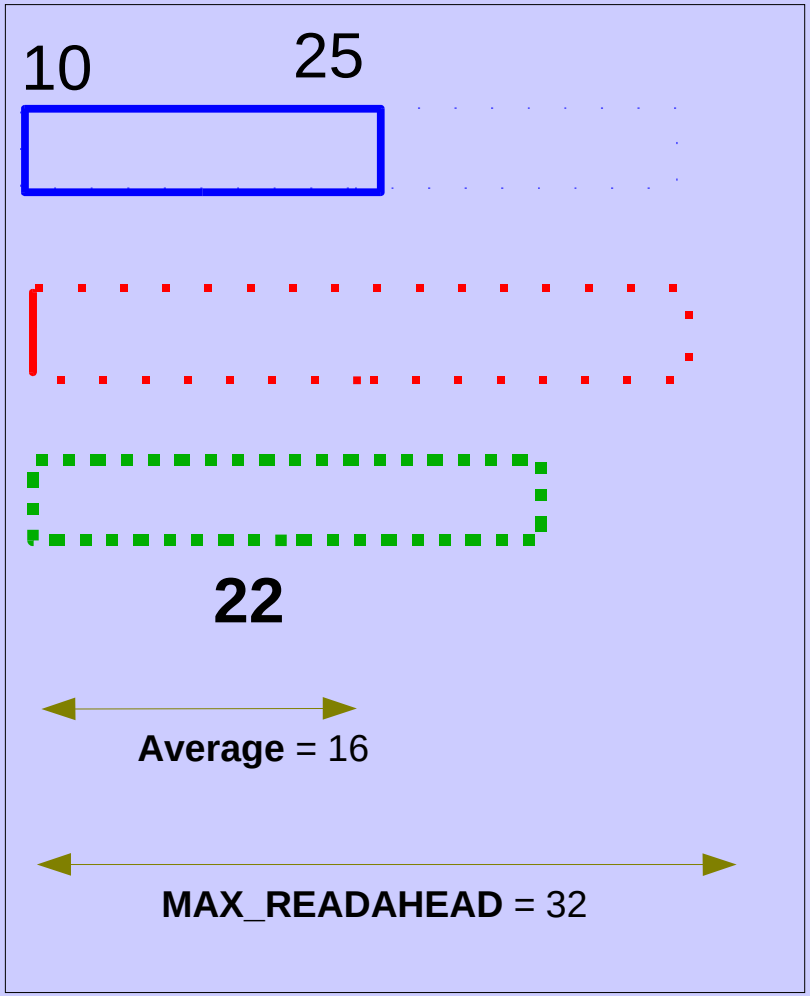
STEP 4: Request for page 12

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



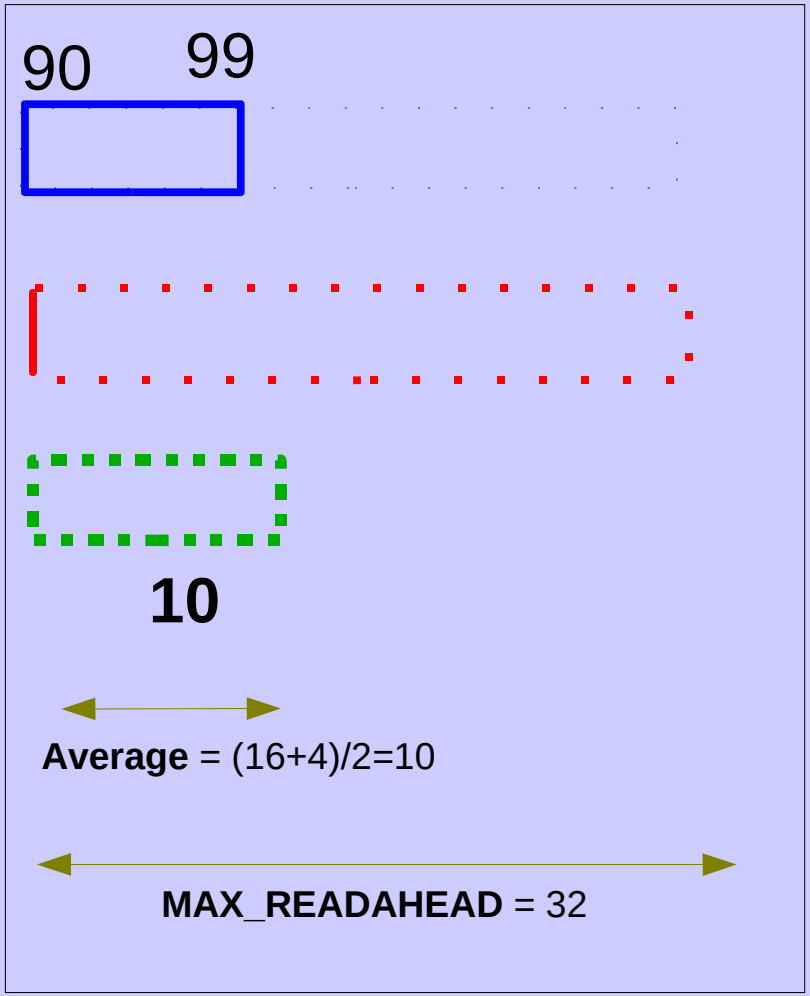
STEP 5: Request for page 13

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



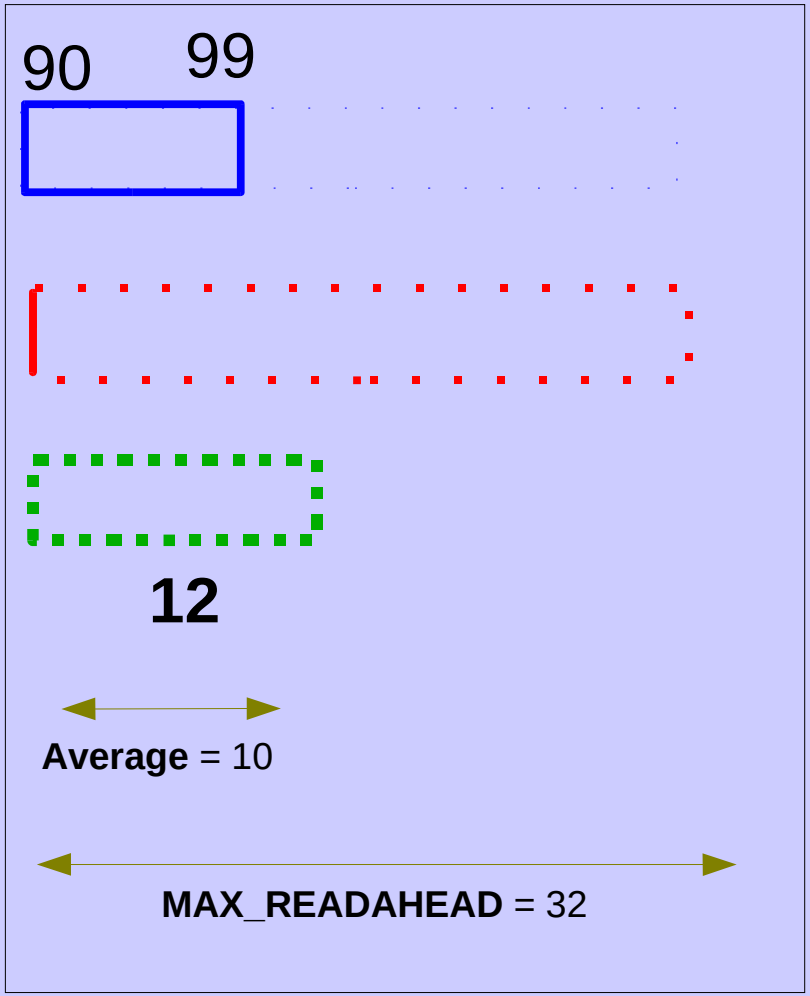
STEP 6: Request for page 90

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



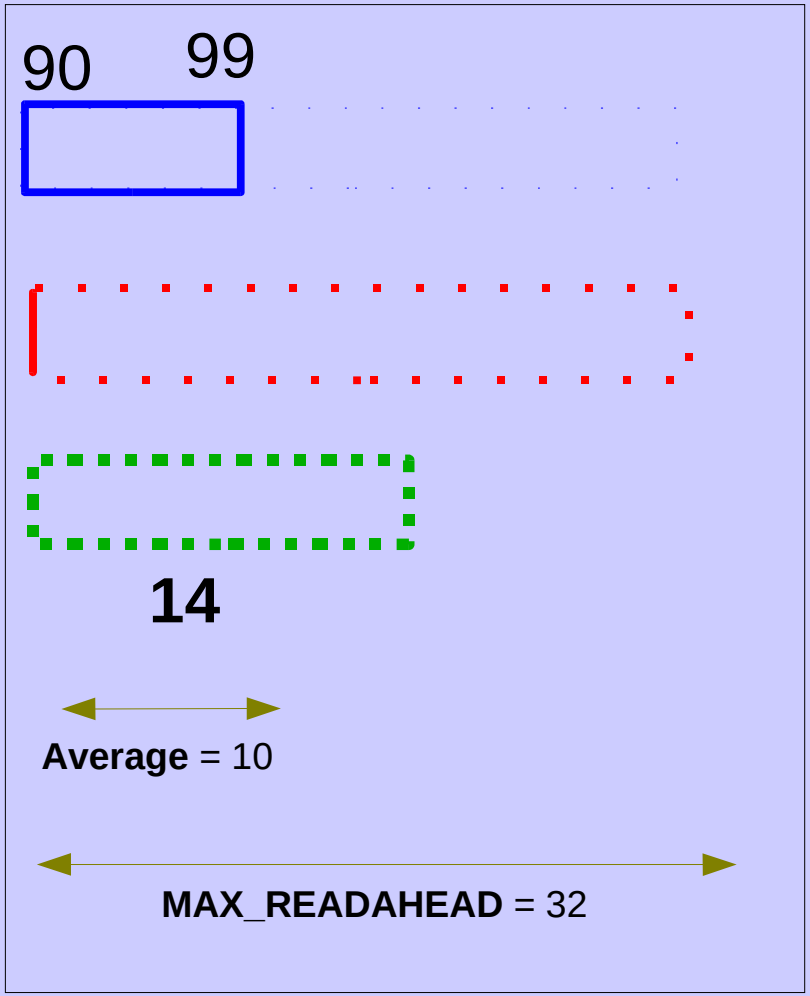
STEP 7: Request for page 91

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



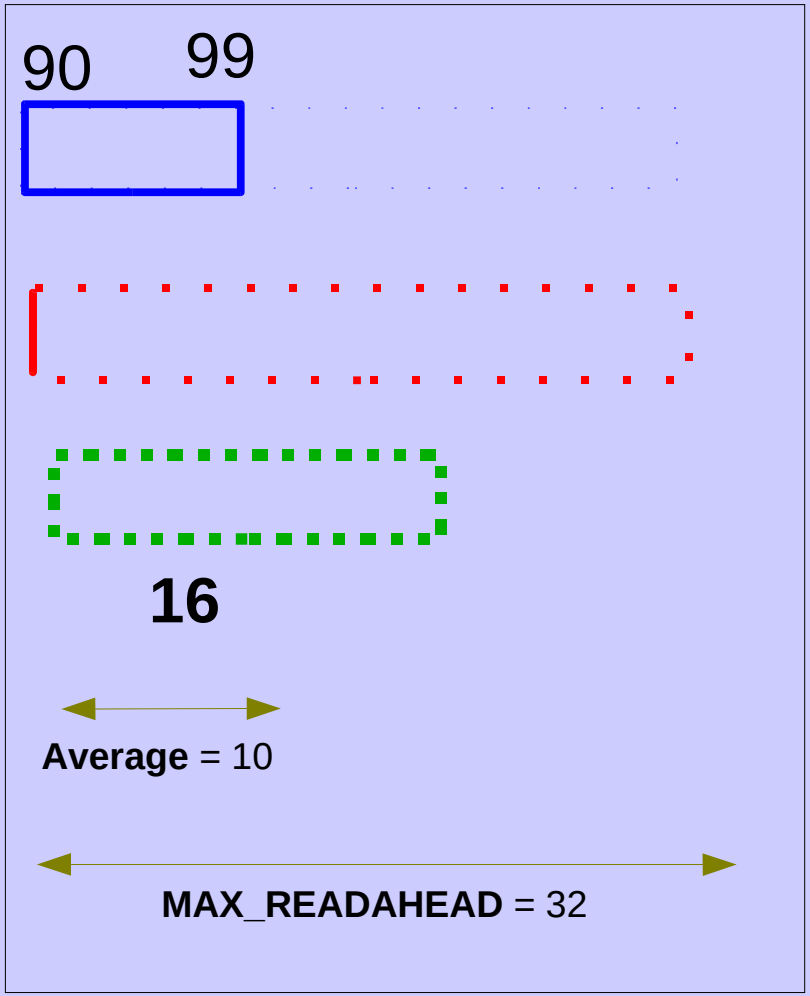
STEP 8: Request for page 92

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



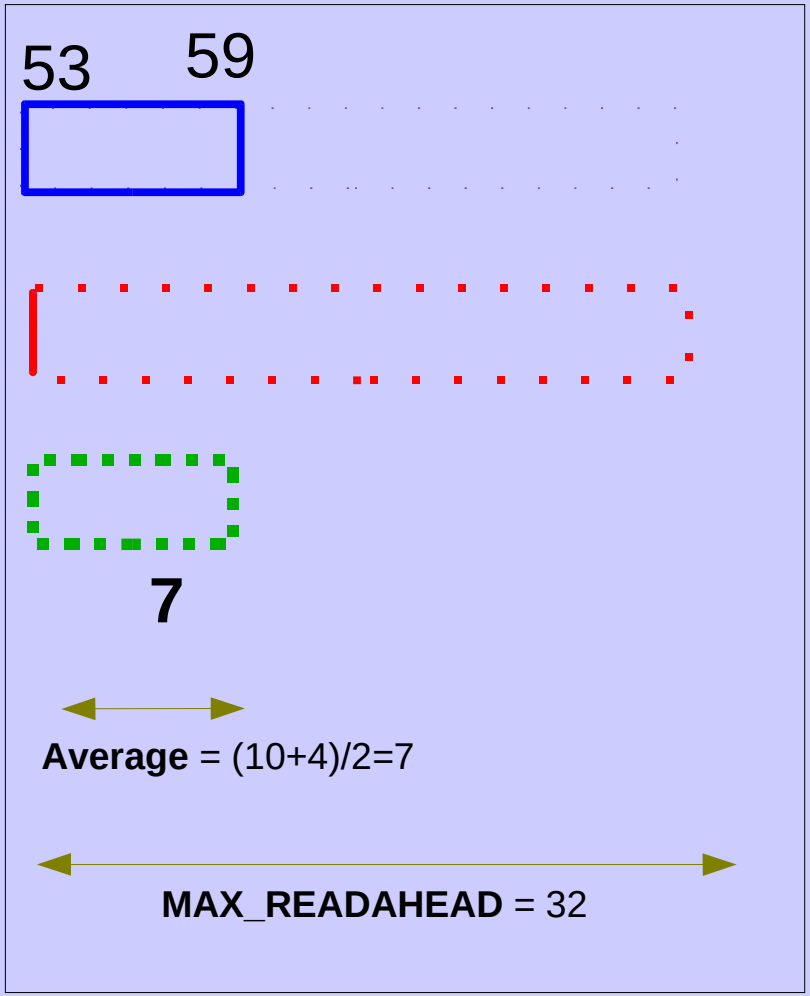
STEP 9: Request for page 93

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



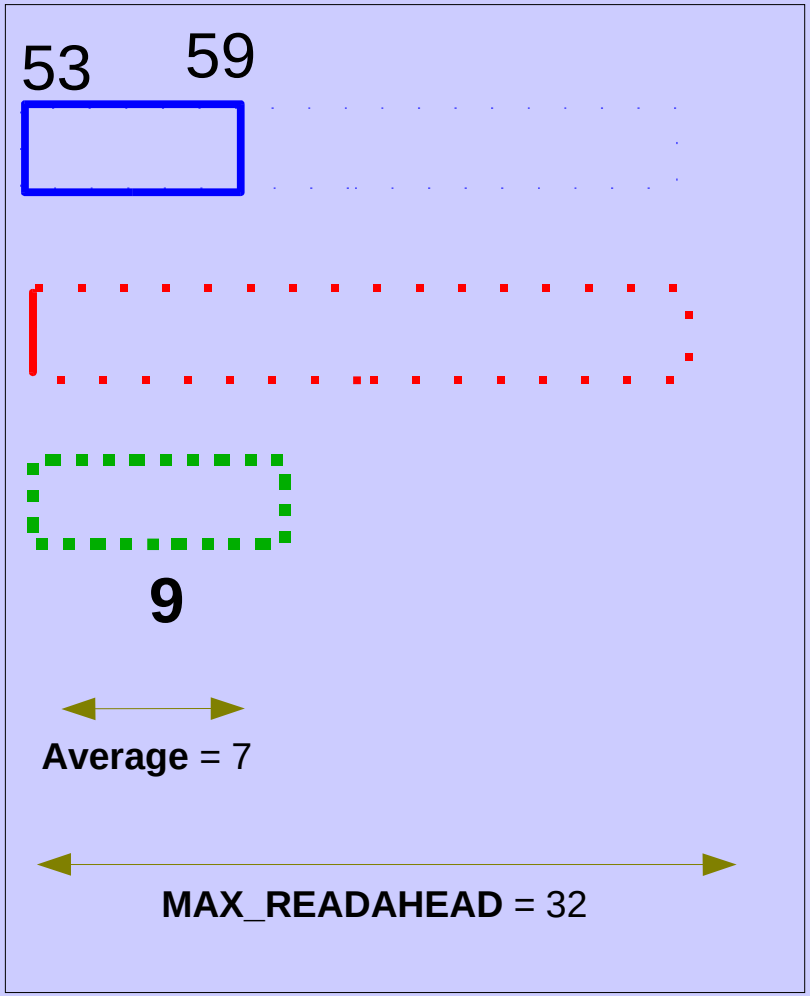
STEP 10: Request for page 53

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



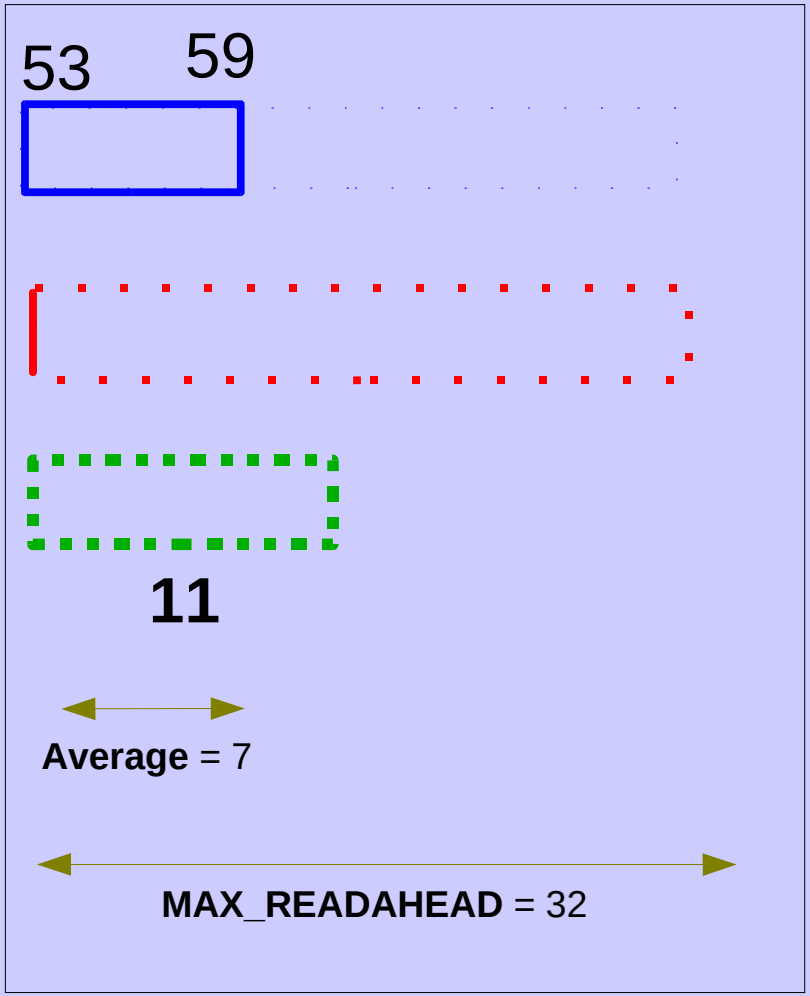
STEP 11: Request for page 54

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



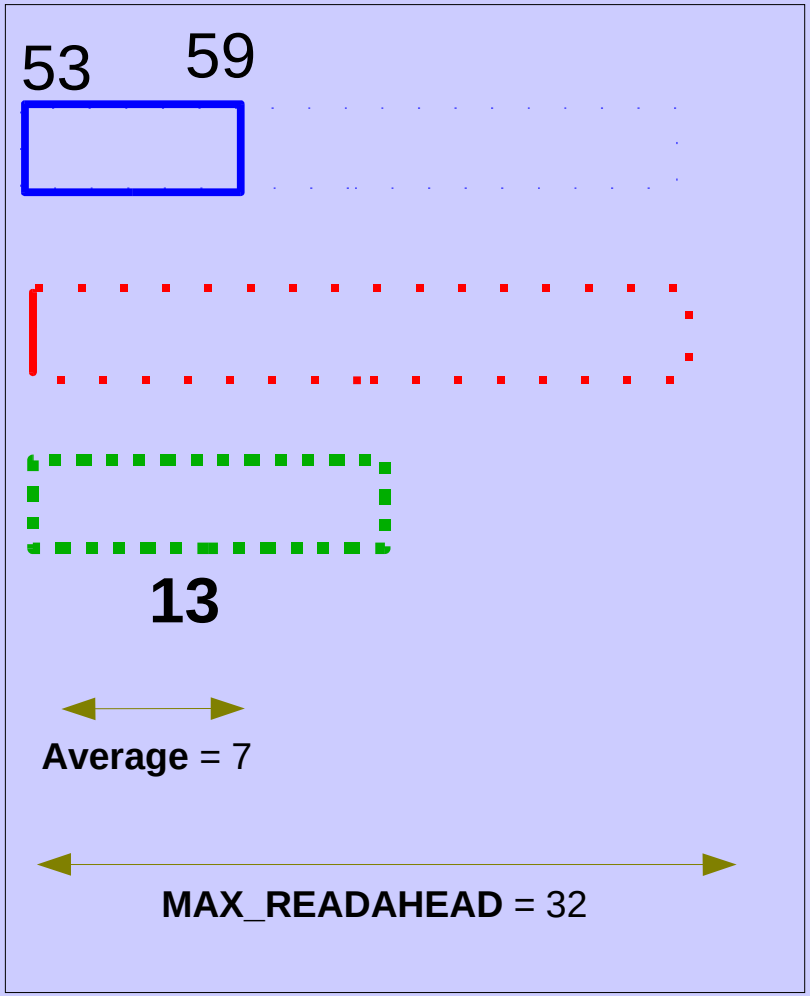
STEP 12: Request for page 55

10, 11, 12, 13
90, 91, 92, 93
53, 54, **55**, 56
28, 29, 30, 31
70, 71, 72, 73



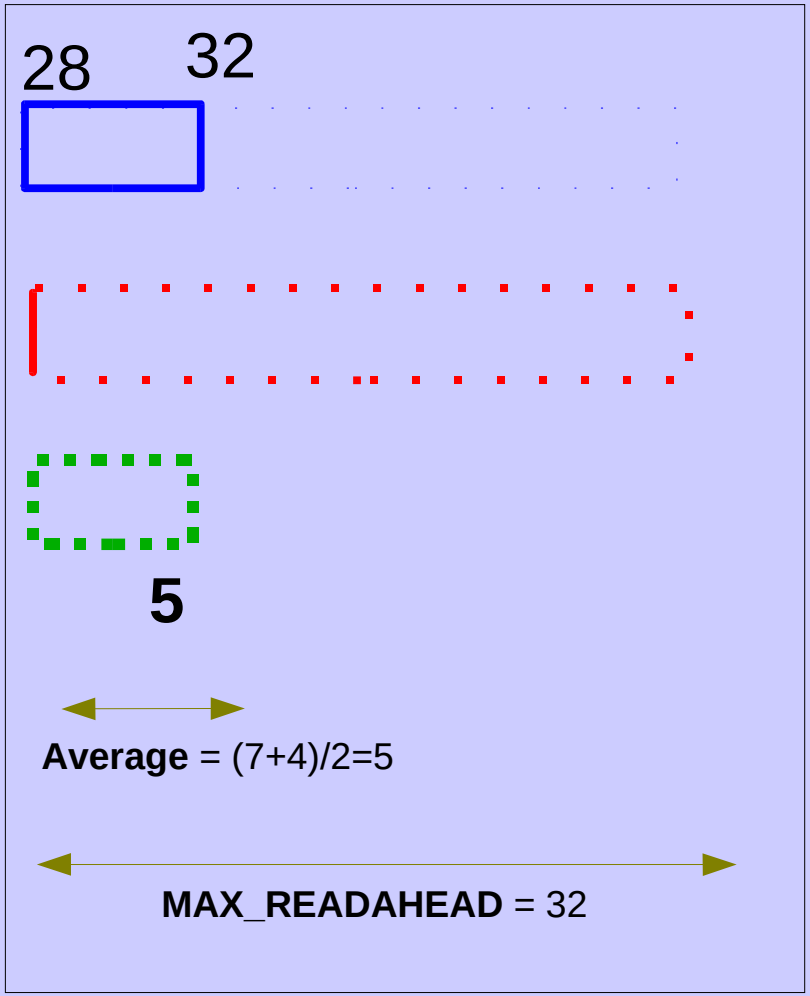
STEP 13: Request for page 56

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



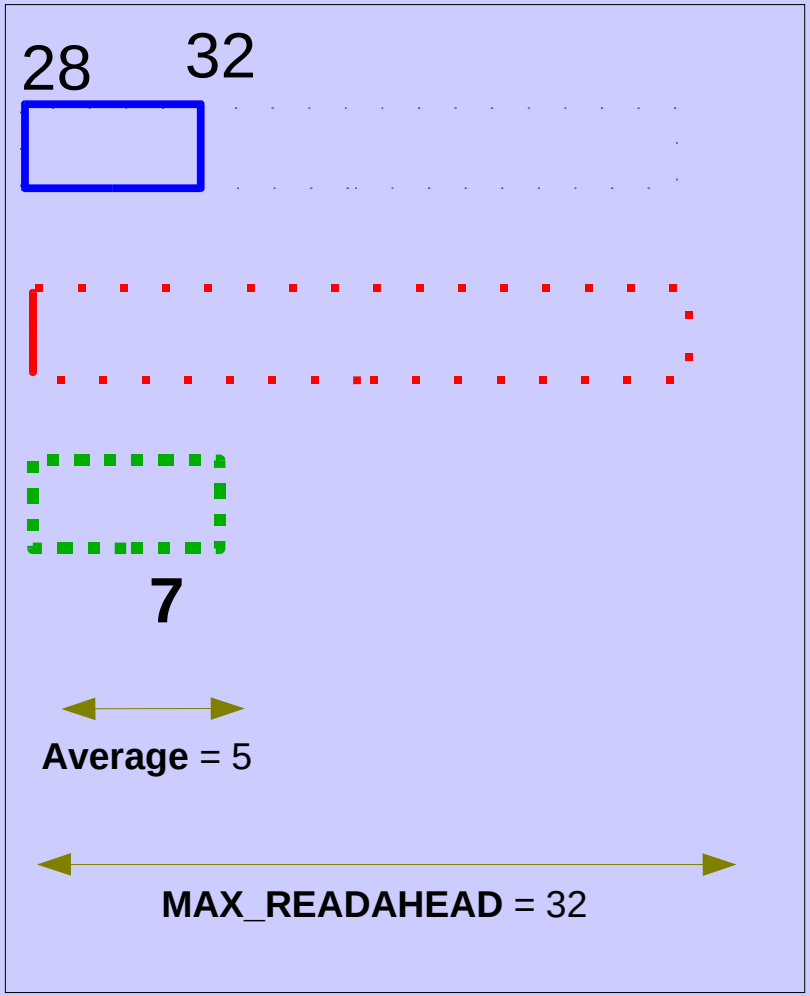
STEP 14: Request for page 28

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, **29, 30, 31**
70, 71, 72, 73



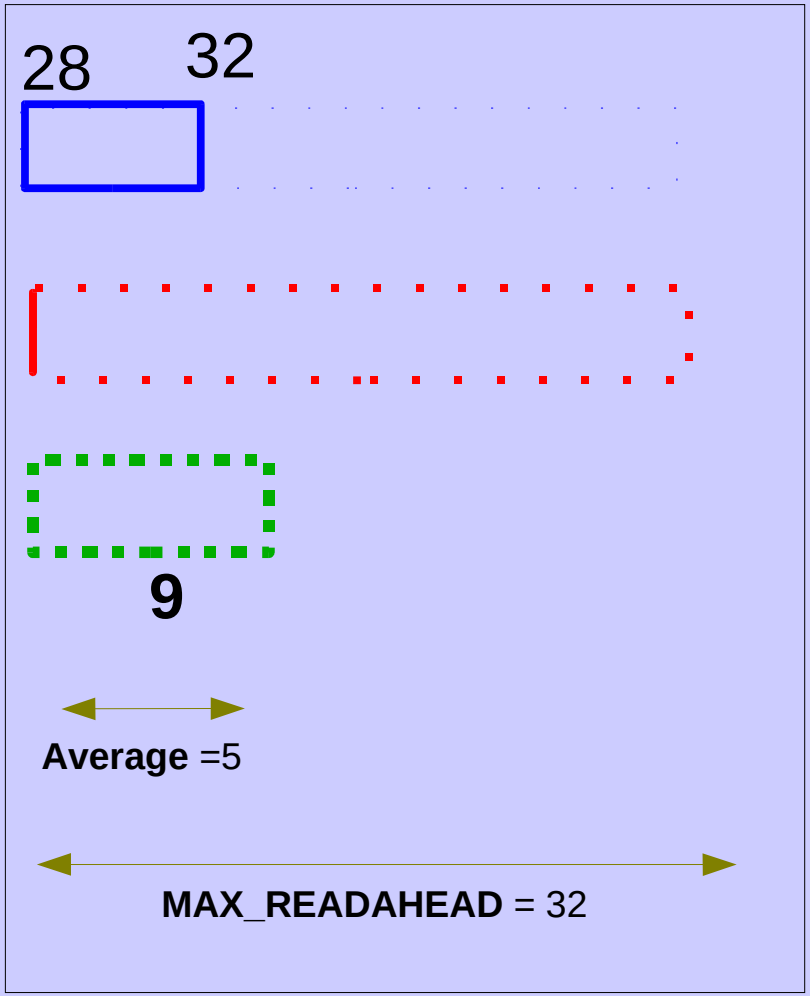
STEP 15: Request for page 29

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



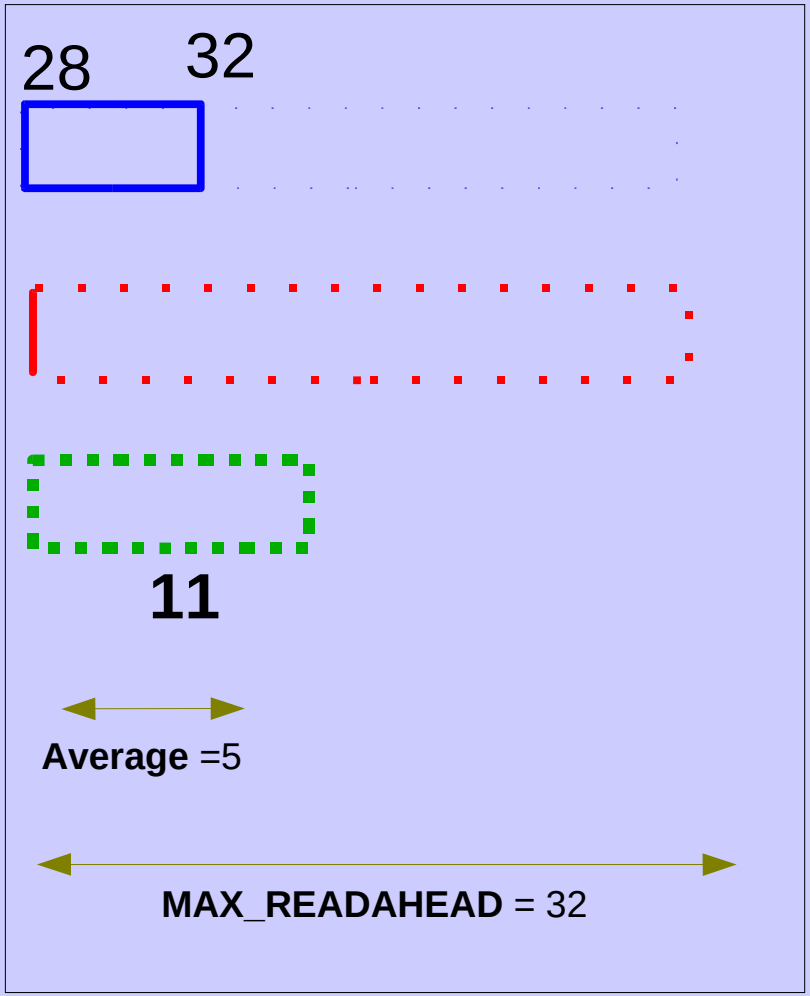
STEP 16: Request for page 30

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, **30**, 31
70, 71, 72, 73



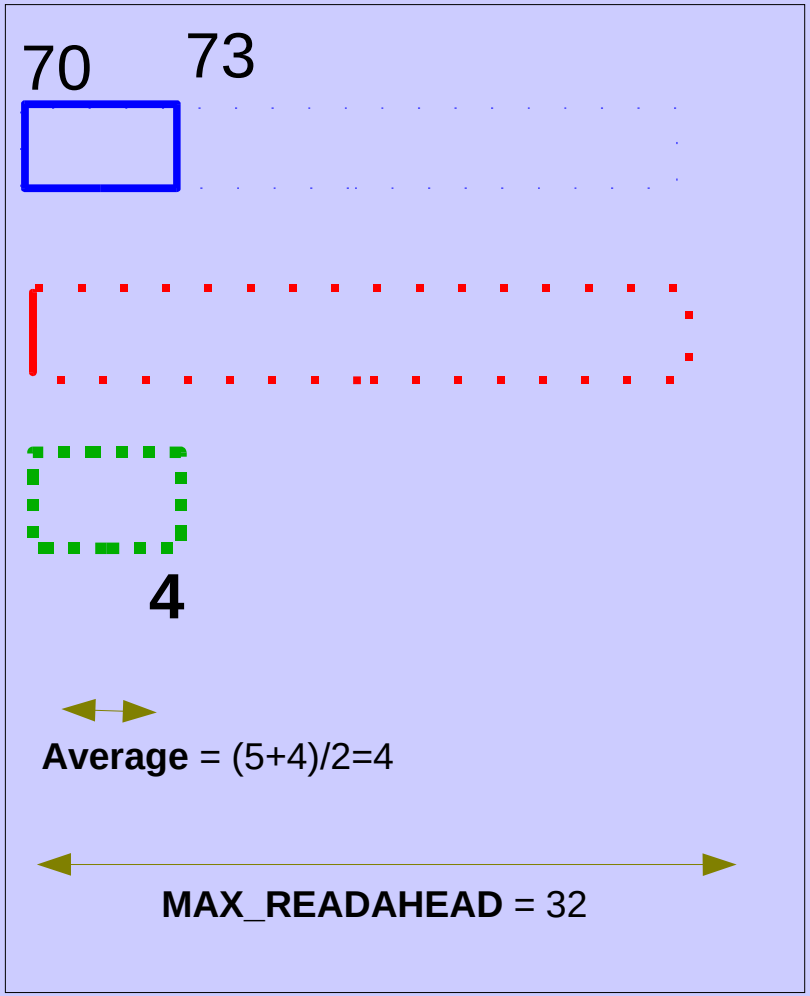
STEP 17: Request for page 31

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, **31**
70, 71, 72, 73



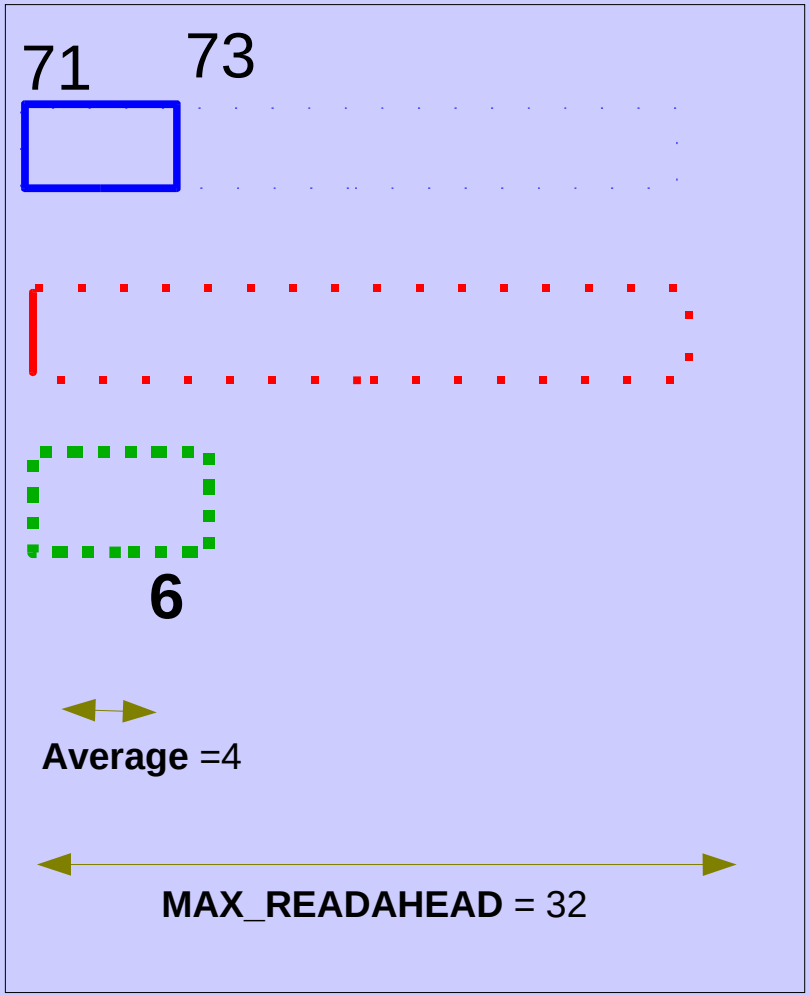
STEP 18: Request for page 70

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



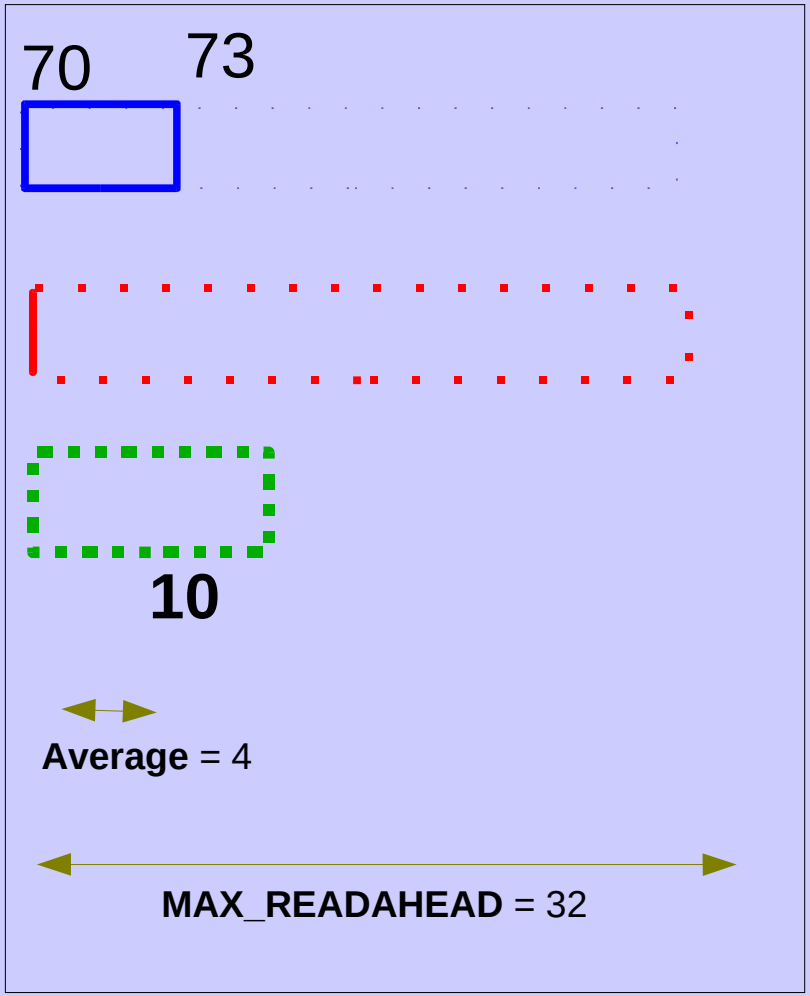
STEP 19: Request for page 71

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



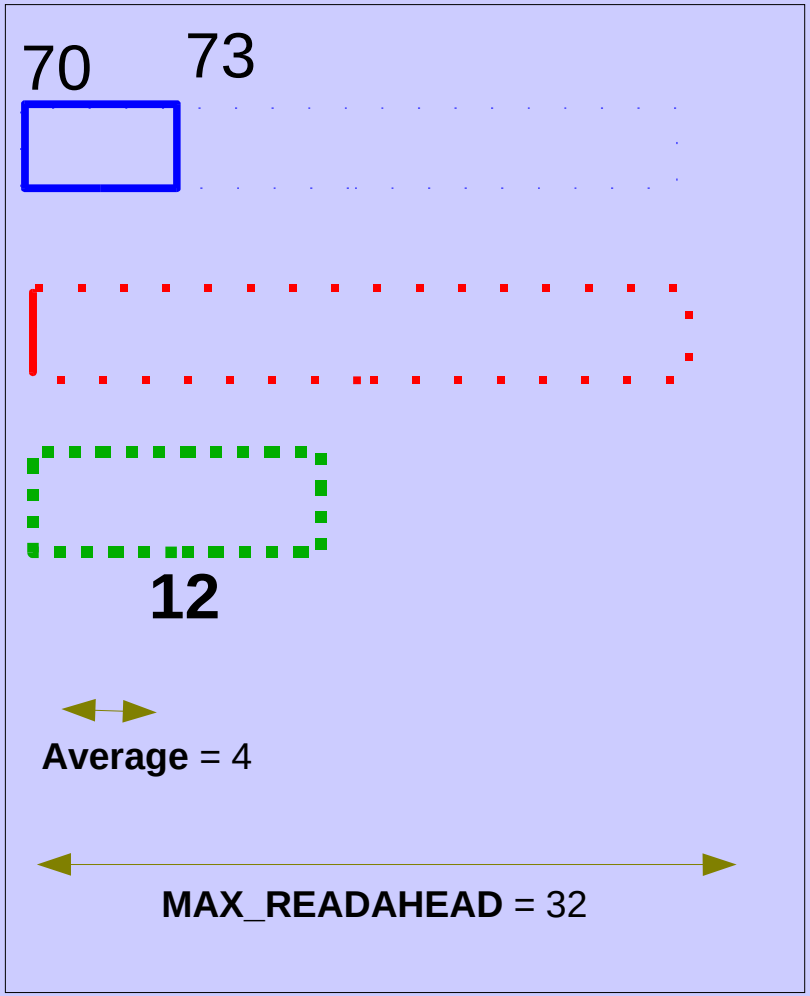
STEP 20: Request for page 72

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



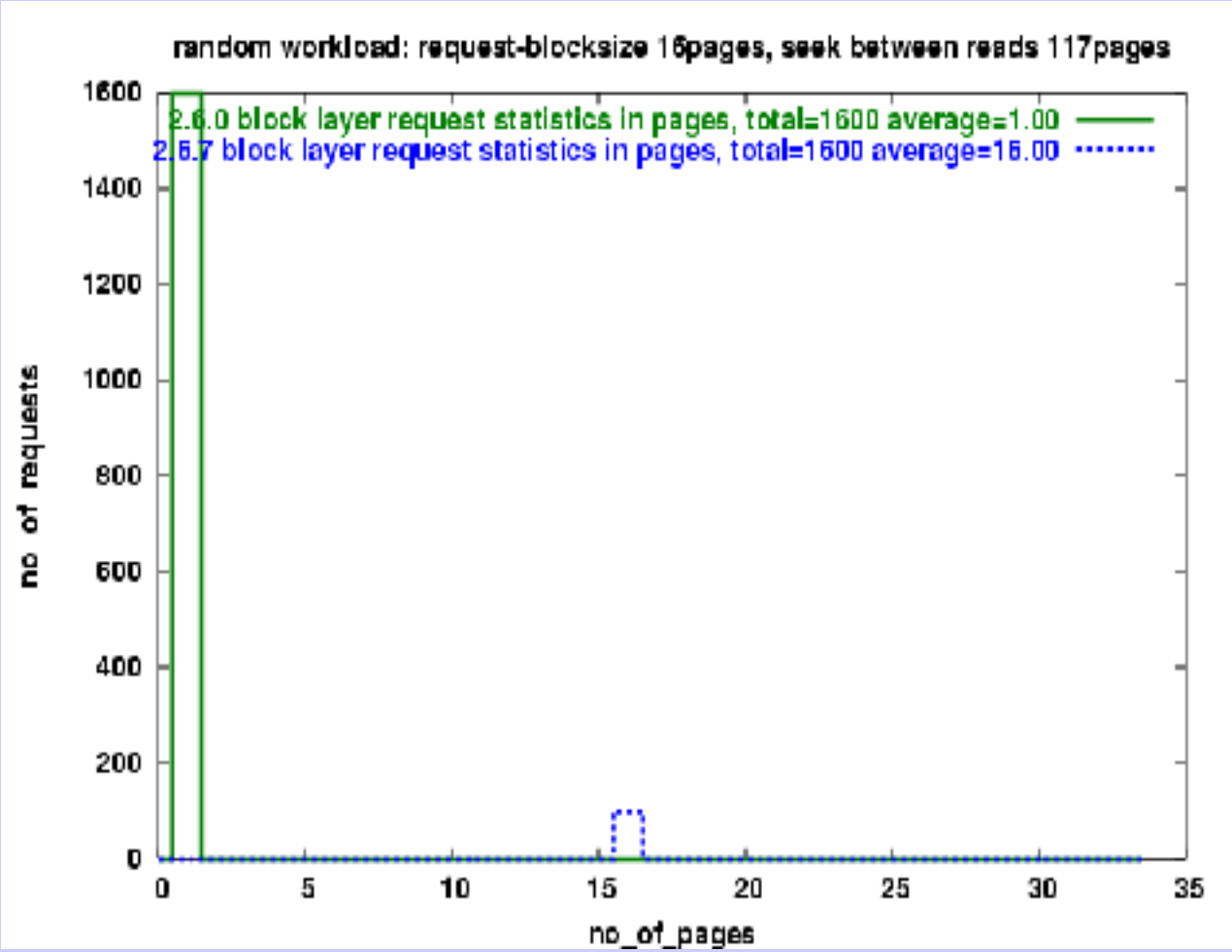
STEP 21: Request for page 73

10, 11, 12, 13
90, 91, 92, 93
53, 54, 55, 56
28, 29, 30, 31
70, 71, 72, 73



2.6.0 v/s 2.6.7

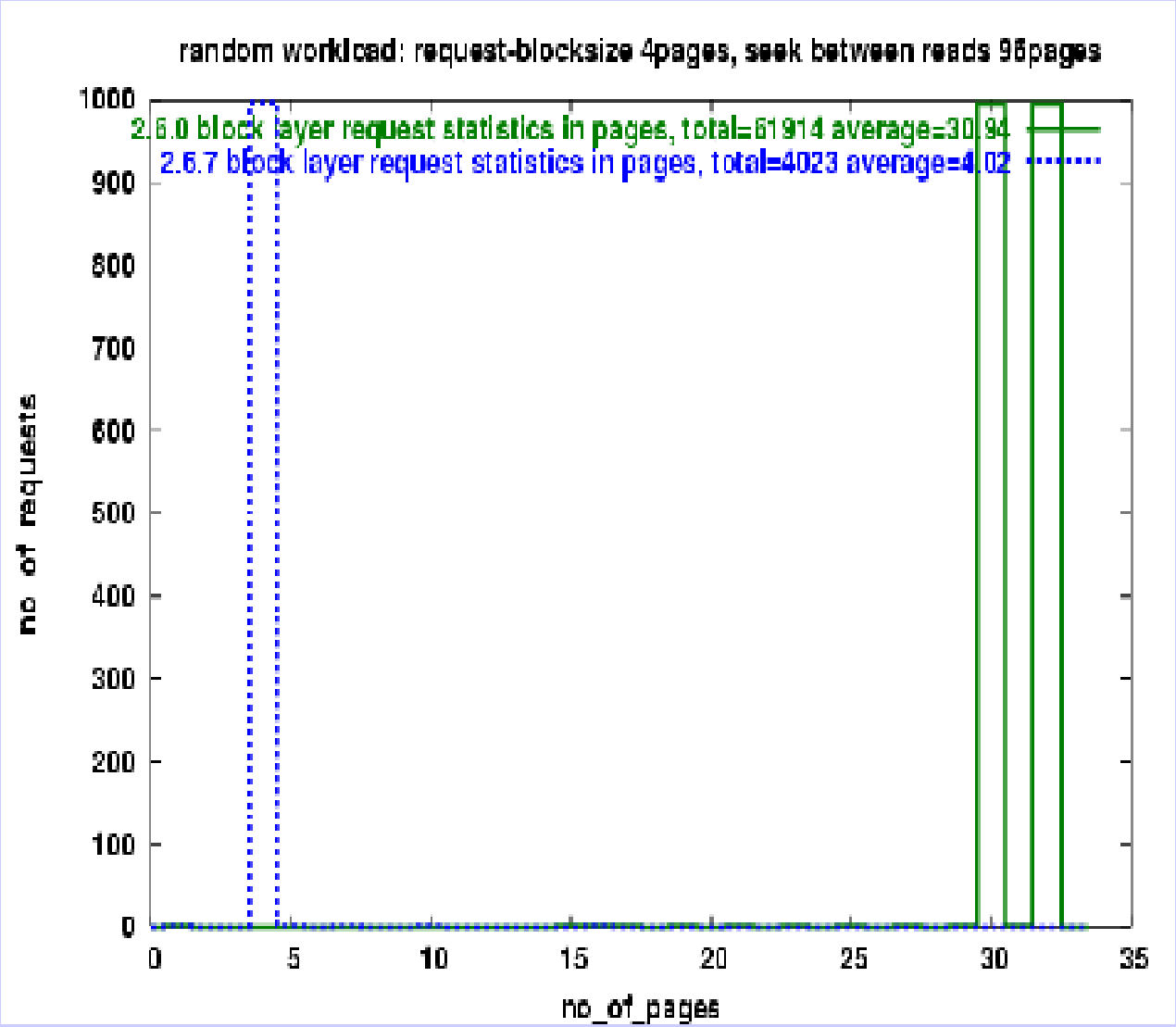
Application generates 16-page request followed by 117-page seek, repeated 100 times. First read request is not at offset 0.



	2.6.0	2.6.7
Average size	1	16
Pages read	1600	1600
Wasted pages	0	0
No of read requests	1600	100

2.6.0 v/s 2.6.7

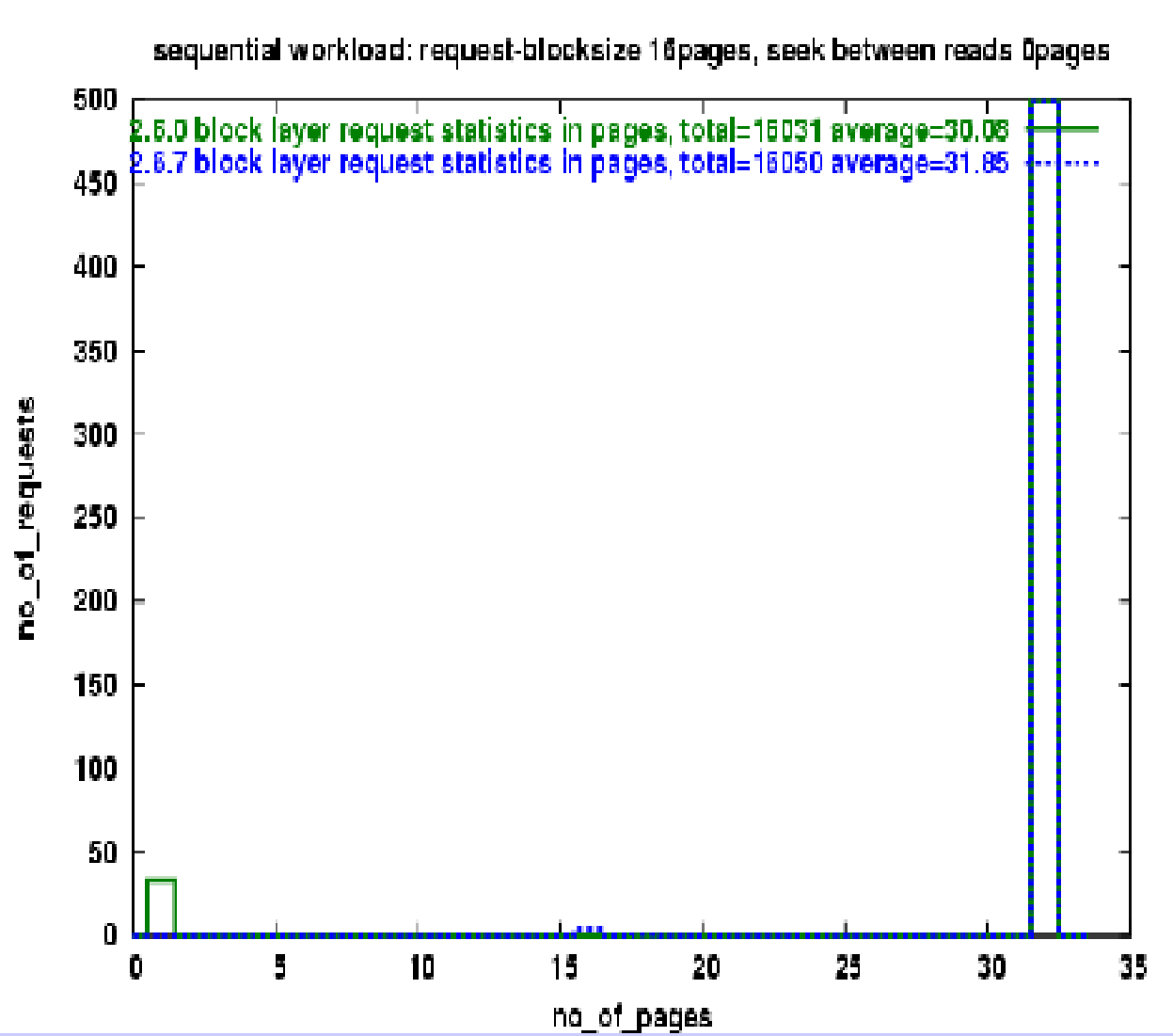
Application generates 4-page request followed by 96-page seek, repeated 1000 times.



	2.6.0	2.6.7
Average size	30.94	4.02
Pages read	61914	4023
Wasted pages	57914	23
No of read requests	2001	1001

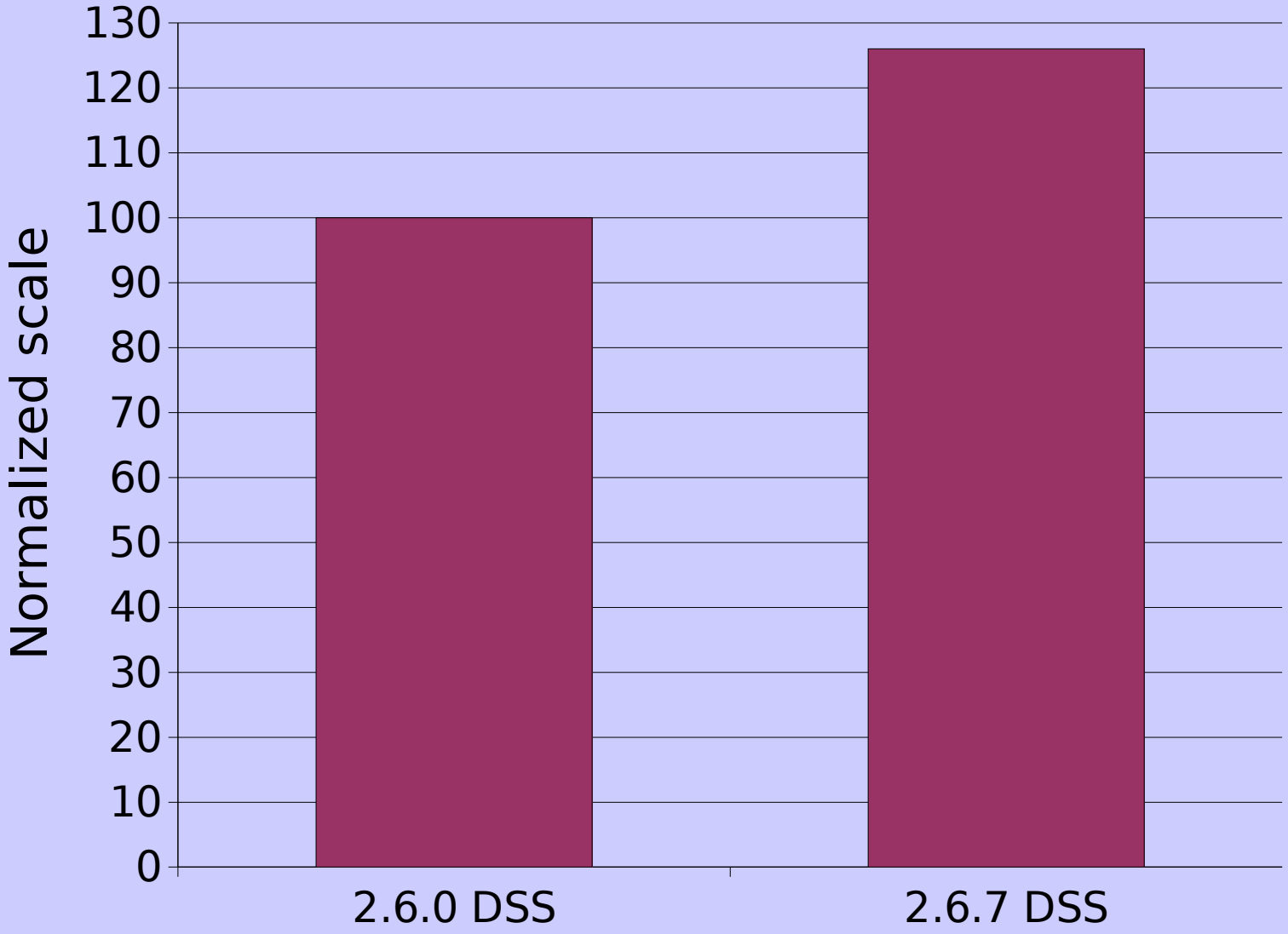
2.6.0 v/s 2.6.7

Application generates sequential 16-page request repeated 1000 times. [SEQUENTIAL READ PATTERN]

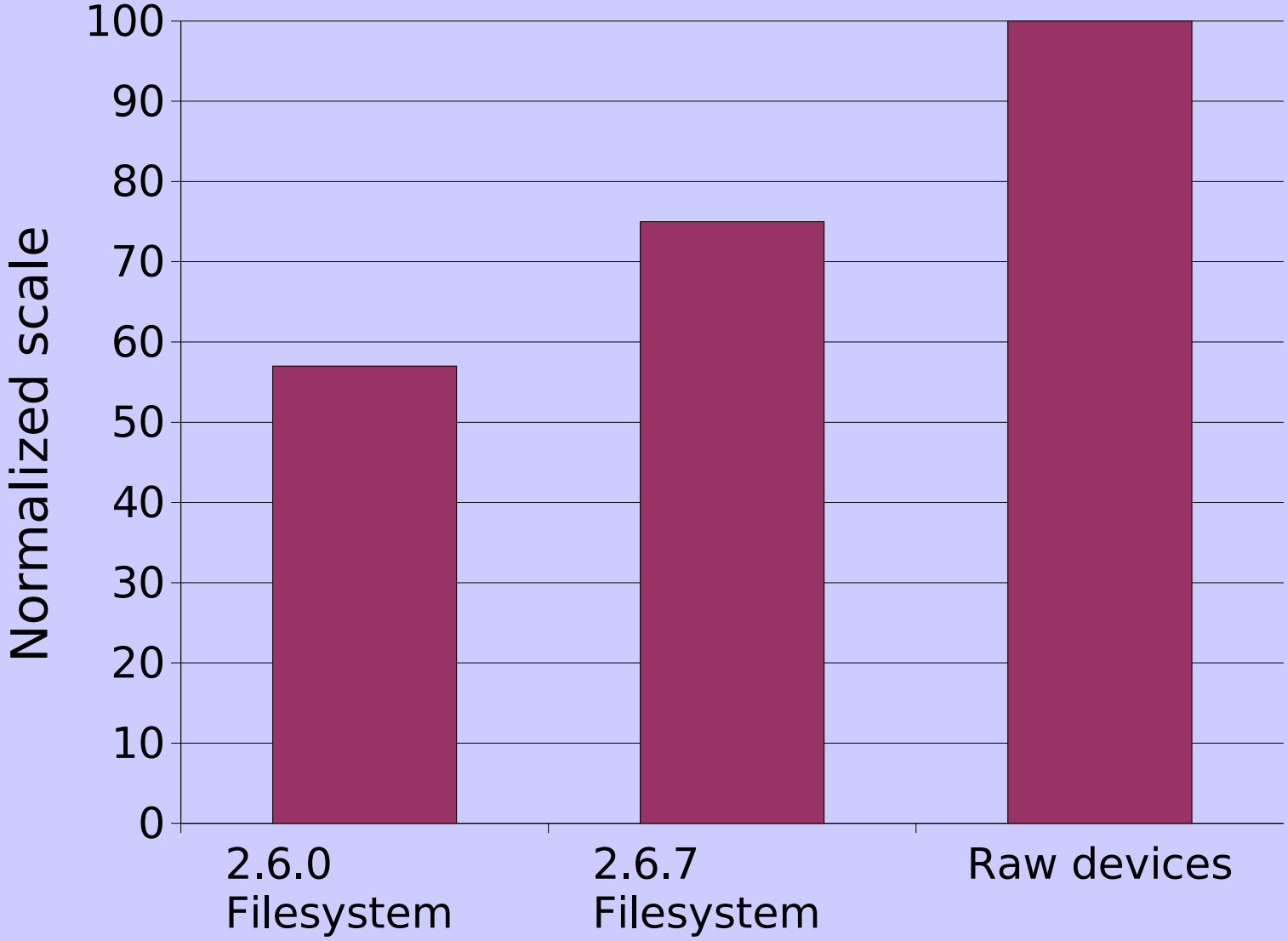


	2.6.0	2.6.7
Average size	30.08	31.85
Pages read	16031	16050
Wasted pages	31	50
No of read requests	533	504

DSS performance comparison



DSS performance comparison



Iozone on NFS comparison

	2.6.0 through-put in KB/sec	2.6.7 through-put in KB/sec
Random read	10056.49	19968.79
Random mix read	10053.37	21565.43
Reverse read	10125.13	20138.83
Stride read	7210.96	14461.63
Sequential read	14464.20	13614.49
Sequential re-read	14591.19	13715.94
Pread	11703.76	13668.21

Filesize = 4G, blocksize=128K

Current Limitations

- **Too large ramp-up time especially for sequential reads**
- **For sequential or large-random reads current-window pages and readahead-window pages can be brought together**
- **Slow read path(readahead-off mode) is too slow!**
- **Multiple threaded reads confuses the algorithm.**

Current Limitations cont..

- **Readahead algorithm is unaware of the size of the current read even though the File-Mapper knows about this.**
- **Does not adapt to page-cache pressure.**
- **MAX_READAHEAD default value is set to 32 which is too small for enterprise workloads.**

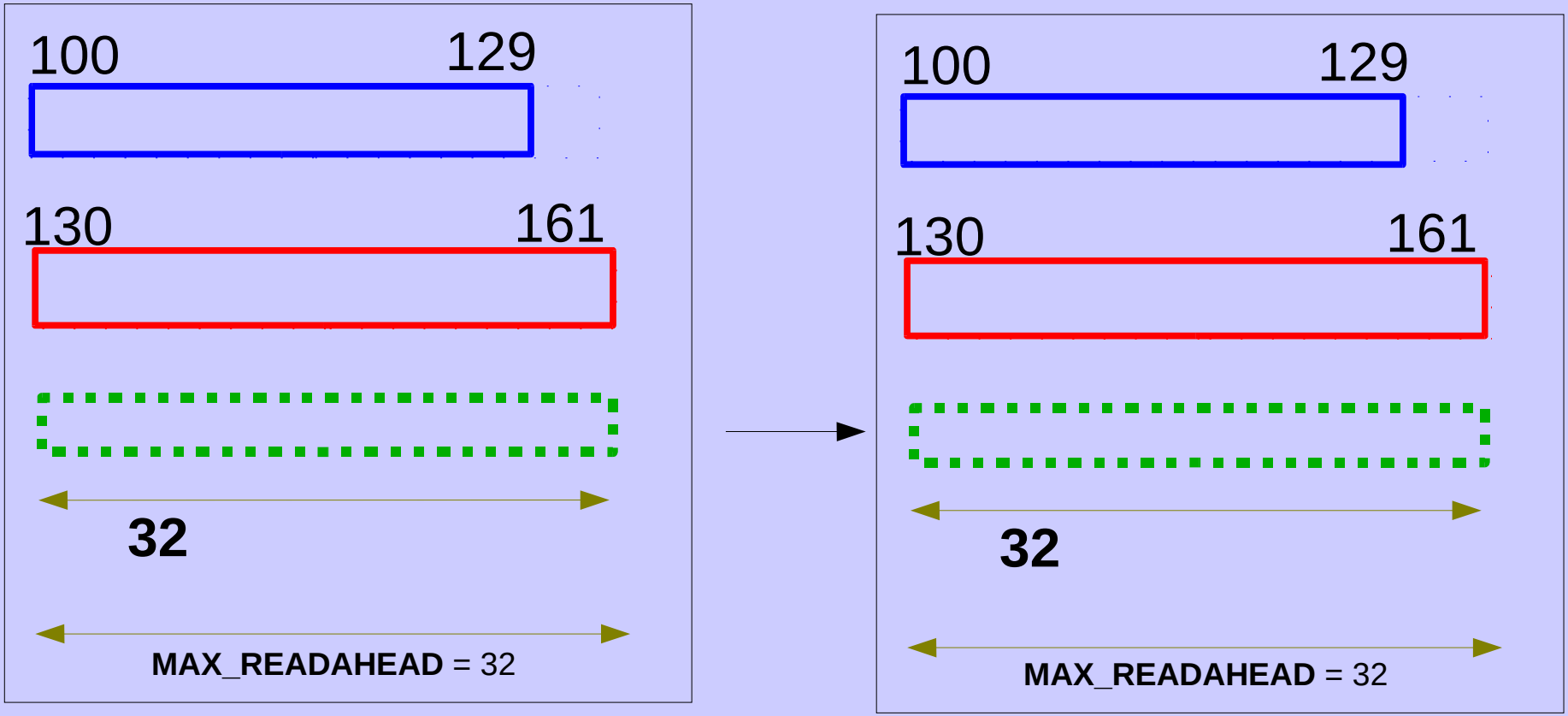
Legal Statement

- **This work represents the view of the authors and does not necessarily represent view of IBM.**
- **IBM is a registered trademark of International Business Machines Corporation in United States, other countries, or both.**
- **Linux is a registered trademark of Linus Torvalds.**
- **Other company, product, and service names may be trademarks or service marks of others.**

Questions?

2.6.0 readahead cont..

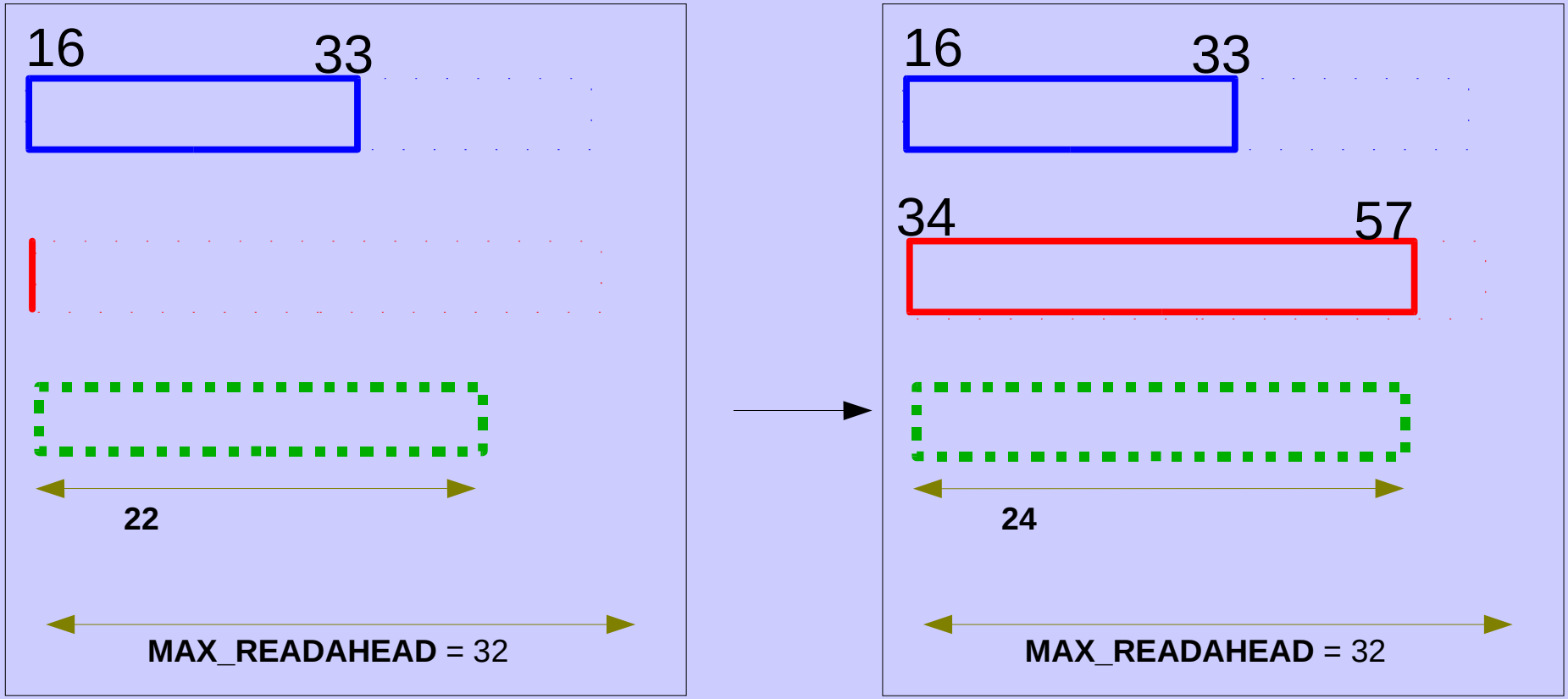
Case 6: read request for page 112



Note: **next window size** cannot exceed **MAX_READAHEAD**

2.6.0 readahead cont..

Case 5: read request for page 17



Note: **readahead window** is populated