PDUG

Gdańsk, 2-3 czerwca 2026

UNDERSTANDING DB2 IO

direct IO, file system caching, prefetching
and page cleaning

Kamil Kuduk

IBM Polska
kamil.kuduk@pl.ibm.com

POLISH DB2 USERS GROUP

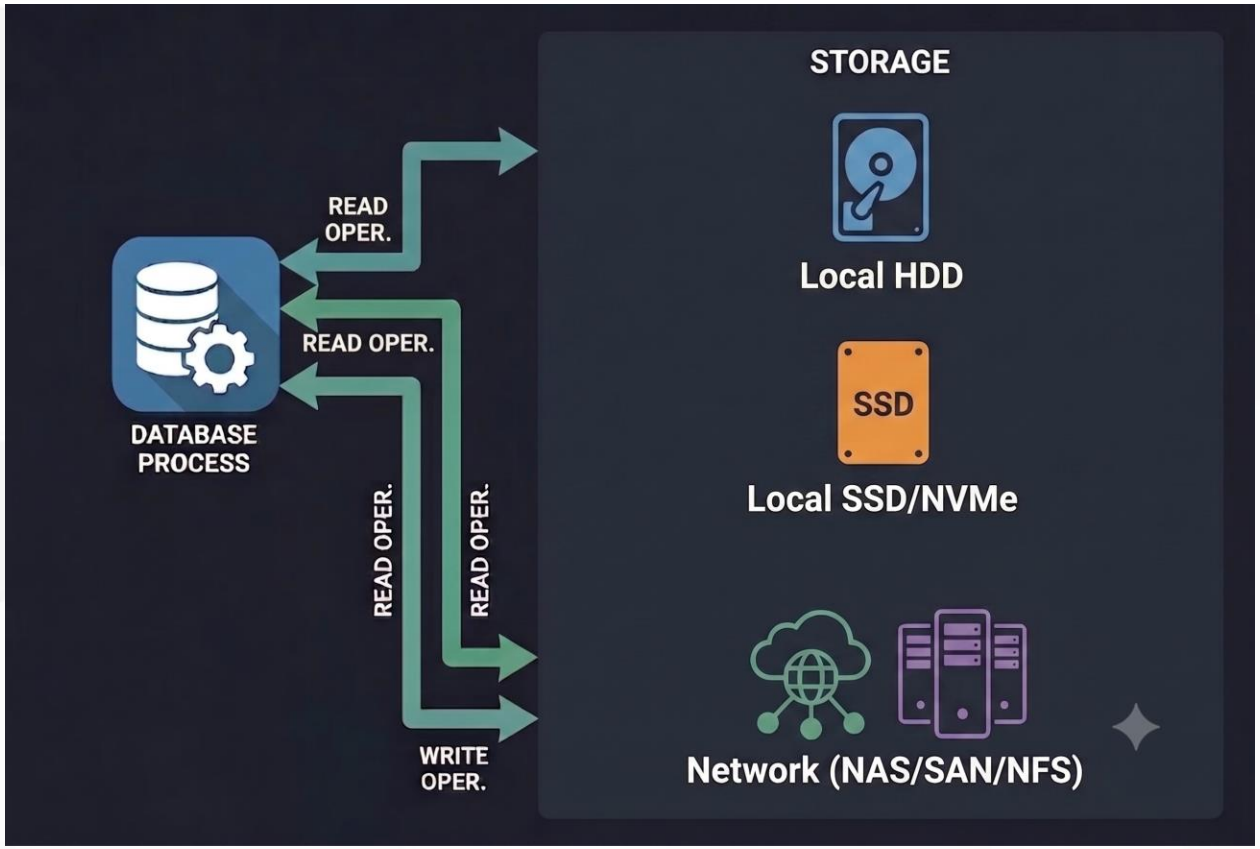
AGENDA

PDUG

1. WHAT IS THE IO?
2. STORAGE LAYERS ON MODERN (LINUX/UNIX) SYSTEMS
3. HOW MUCH TIME AN IO TAKES?
4. DIRECT VS BUFFERED IO
5. DB2 DATA STORAGE
6. SYNCHRONOUS VS ASYNCHRONOUS IO
7. PREFETCHING
8. PAGE CLEANING
9. DIRECT READS/WRITES IN DB2
10. BACKUP, LOGGING AND OTHER IOS

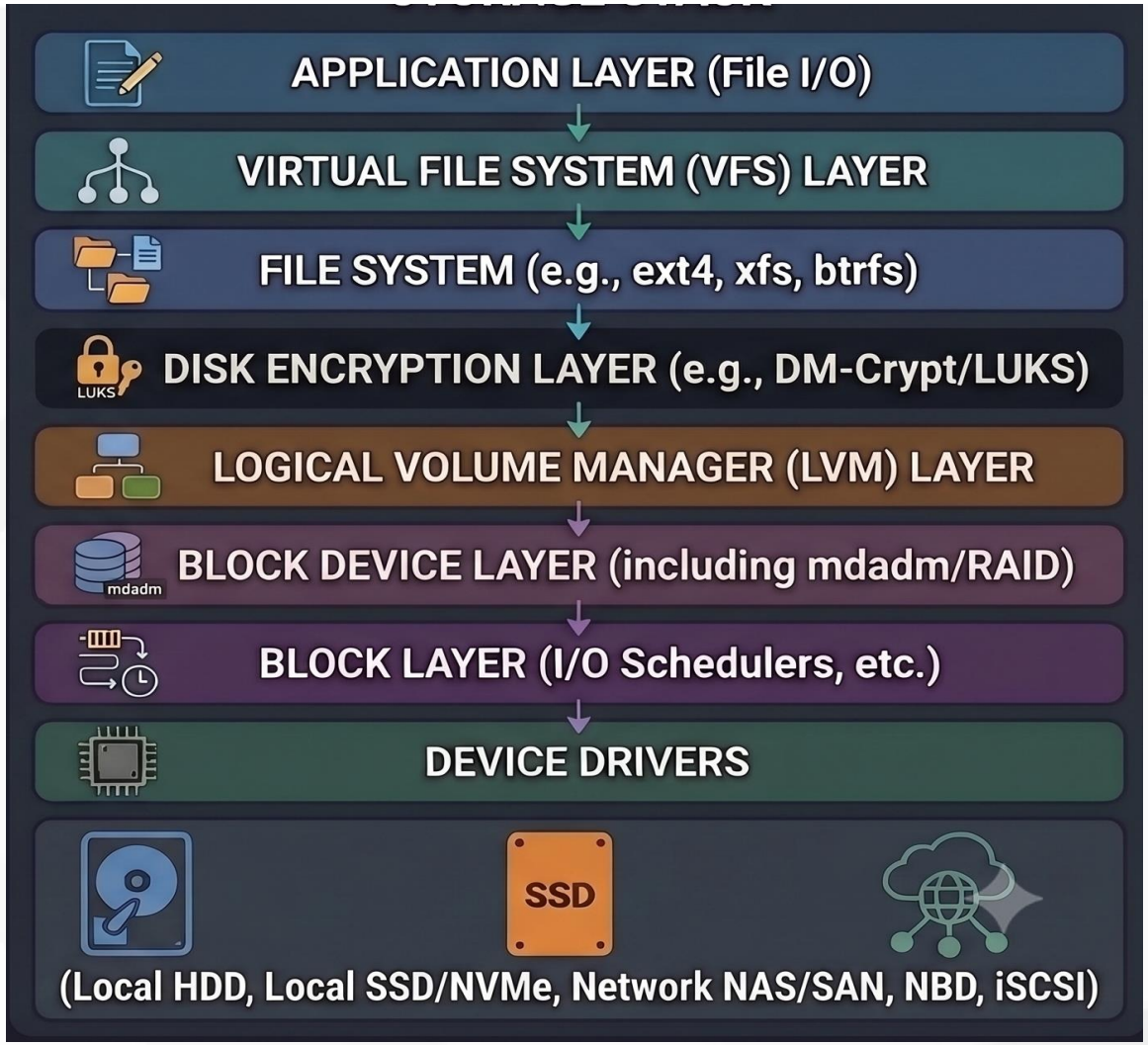
WHAT IS THE IO?

- Read and write operation
- Uses permanent storage
- Ensures ACID transactions
- "Everything is a file"
- Buffering for efficiency



STORAGE LAYERS ON MODERN (LINUX/UNIX) SYSTEMS

- Files on VFS
- File system (xfs, ext4)
- LVM
- LUKS (optional)
- Block layer
- Device drivers
- Hardware storage



HOW MUCH TIME AN IO TAKES?

L1 cache reference	<1 ns		
Branch mispredict	5 ns		
L2 cache reference	7 ns		(14x L1 cache)
Mutex lock/unlock (Db2 latch)	25 ns		
Main memory reference	100 ns		(20x L2, 200x L1)
Send 1K bytes over 1 Gbps network	10,000 ns	10 us	
Read 4K randomly from SSD*	150,000 ns	150 us	(~1GB/s SSD)
Read 1 MB sequentially from memory	250,000 ns	250 us	
Round trip within same datacenter	500,000 ns	500 us	
Read 1 MB sequentially from SSD*	1,000,000 ns	1,000 us	1 ms ~1GB/s SSD
HDD seek/random read	4,000,000 ns	4,000 us	4 ms 8x local net
Read 1 MB sequentially HDD	20,000,000 ns	20,000 us	20 ms(80x mem, 20x SSD)
Send packet Poland->USA->Poland	100,000,000 ns	100,000 us	100 ms

HOW MUCH TIME AN IO TAKES?

Logical(in memory) page read:	0.001 ms (1 us)
Acquiring a lock (no wait):	0.002 ms (2 us)
Acquiring a lock on pureScale:	0.020 ms (20 us)
Latch wake up (semop) 0.010-0.100 ms	**(10-100 us)**
Physical read from disk:	~0.2-4 ms
Commit	~0.3-1 ms
Commit with HADR in SYNC	~1-3 ms
Dynamic SQL compilation:	>2 ms (up to mins)
Blink of an eye	100-400 ms
TBSCAN on 1 GB table:	300 ms (also CPU cost)
1 GB prefetch from NVMe	~500 ms
1 GB prefetch from SSD	~1000-2000 ms
1 GB prefetch from HDD	~6000 ms

HOW MUCH TIME AN IO TAKES?

```
# READ
dd if=./myfile of=/dev/zero bs=16k count=1
16384 bytes (16 kB, 16 KiB) copied, 2,3387e-05 s, 701 MB/s

dd if=./myfile of=/dev/zero bs=16k count=1000
16384000 bytes (16 MB, 16 MiB) copied, 0,002366 s, 6,9 GB/s

# WRITE
dd if=/dev/zero of=./myfile bs=16k count=1
16384 bytes (16 kB, 16 KiB) copied, 0,000126951 s, 129 MB/s

dd if=/dev/zero of=./myfile bs=16k count=1000
16384000 bytes (16 MB, 16 MiB) copied, 0,00963649 s, 1,7 GB/s
```

HOW MUCH TIME AN IO TAKES? (WRITES)

```
dd if=/dev/zero of=./myfile bs=16k count=1000
16384000 bytes (16 MB, 16 MiB) copied, 0,00963649 s, 1,7 GB/s

dd if=/dev/zero of=./myfile bs=16k count=1000 conv=fsync
16384000 bytes (16 MB, 16 MiB) copied, 0,017311 s, 946 MB/s

dd if=/dev/zero of=./myfile bs=16k count=1000 oflag=dsync,direct
16384000 bytes (16 MB, 16 MiB) copied, 1,70336 s, 9,6 MB/s

dd if=/dev/zero of=./myfile bs=160k count=100 oflag=dsync,direct
16384000 bytes (16 MB, 16 MiB) copied, 0,256296 s, 63,9 MB/s

dd if=/dev/zero of=./myfile bs=1600k count=10 oflag=dsync,direct
16384000 bytes (16 MB, 16 MiB) copied, 0,0427372 s, 383 MB/s
```

HOW MUCH TIME AN IO TAKES? (READS)

```
dd if=./myfile of=/dev/zero bs=16k count=1000
16384000 bytes (16 MB, 16 MiB) copied, 0,002366 s, 6,9 GB/s

dd if=./myfile of=/dev/zero bs=16k count=1000 iflag=direct
16384000 bytes (16 MB, 16 MiB) copied, 0,0749547 s, 219 MB/s

dd if=./myfile of=/dev/zero bs=16k count=1000 iflag=direct,sync
16384000 bytes (16 MB, 16 MiB) copied, 0,0743046 s, 220 MB/s

dd if=./myfile of=/dev/zero bs=160k count=100 iflag=direct
16384000 bytes (16 MB, 16 MiB) copied, 0,0218998 s, 748 MB/s

dd if=./myfile of=/dev/zero bs=1600k count=10 iflag=direct
16384000 bytes (16 MB, 16 MiB) copied, 0,0110049 s, 1,5 GB/s
```

DB2 DATA STORAGE

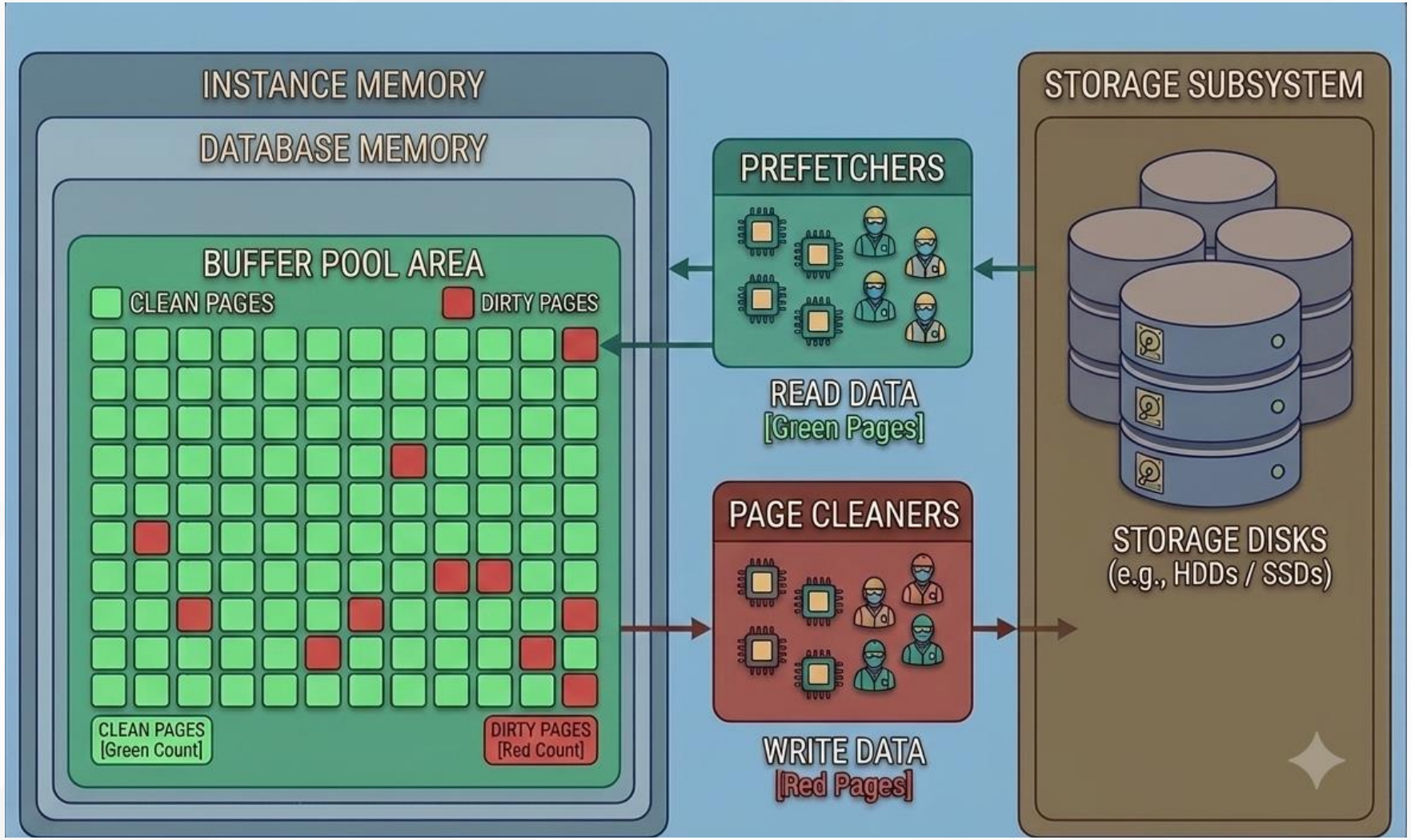
```
Tablespace Configuration:
Id     Type Content Pagesz ExtentSz Prefetch BufID  FSC  Name
0      DMS  Regular 4096   4        4        1      Def  SYSCATSPACE
1      SMS  SysTmp  4096   32       32       1      On   TEMPSPACE1
2      DMS  Large   4096   32       32       1      Def  USERSPACE1
3      DMS  Large   4096   32       32       1      Def  TS_DATA

Containers:
TspId Type    TotalPgs   Container
0     File    32768      /db2data/NODE0000/MYDB/T0000000/C0000000.CAT
1     Path    1          /db2data/NODE0000/MYDB/T0000001/C0000000.TMP
2     File    8192       /db2data/NODE0000/MYDB/T0000002/C0000000.LRG
3     File    1957888    /db2data/NODE0000/MYDB/T0000003/C0000000.LRG
```

SYNCHRONOUS VS ASYNCHRONOUS IO

- In Db2, the IO is considered synchronous vs async from the perspective of the coordinator EDU (agent) for the SQL
- We want as little synchronous IO as possible
- Write ahead log (WAL) is always synchronous
- Db2 is using both synchronous (*pread/pwrite*) as well as async IO interfaces (*io_submit/listio/IOCP*)
- Async IO infrastructure consists of bufferpools (IO caching layer) and "helper threads": page cleaners and prefetchers

SYNCHRONOUS VS ASYNCHRONOUS IO



PREFETCHING

- Larger, parallel reads
- A single IO size is equals to: page_size * extent_size
- Number of parallel requests depend on number of containers (storage paths) or DB2_PARALLEL_IO setting
- Done in the background by db2pfchr EDUs (NUM_IOSERVERS)
- Used also by backup and LOB reads/writes

PREFETCHING

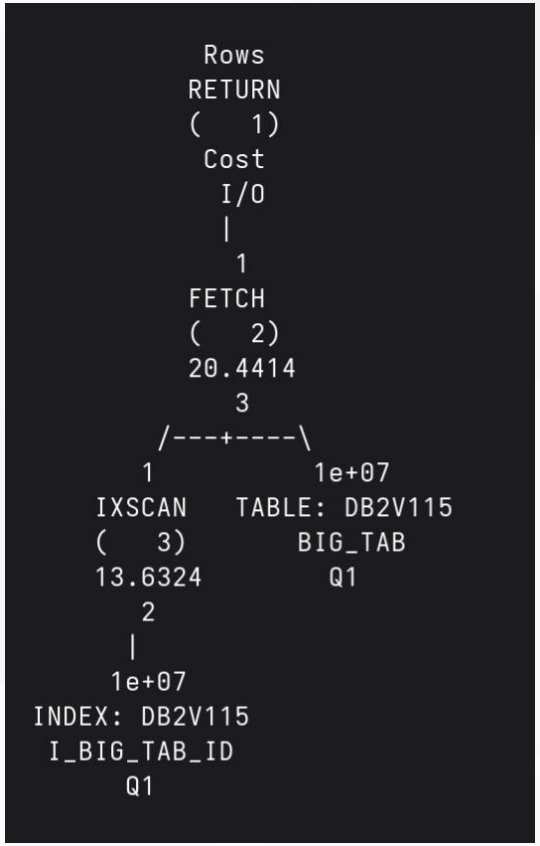
```
Id     Type Content PageSz ExtentSz Auto Prefetch BufID Name
0      DMS  Regular 8192   4        Yes  4        1     SYSCATSPACE
1      SMS  SysTmp  8192   32       Yes  32       1     TEMPSPACE1
2      DMS  Large   8192   32       Yes  32       1     USERSPACE1

db2set "DB2_PARALLEL_IO=*:10"

Id     Type Content PageSz ExtentSz Auto Prefetch BufID Name
0      DMS  Regular 8192   4        Yes  40       1     SYSCATSPACE
1      SMS  SysTmp  8192   32       Yes  320      1     TEMPSPACE1
2      DMS  Large   8192   32       Yes  320      1     USERSPACE1
```

PREFETCHING

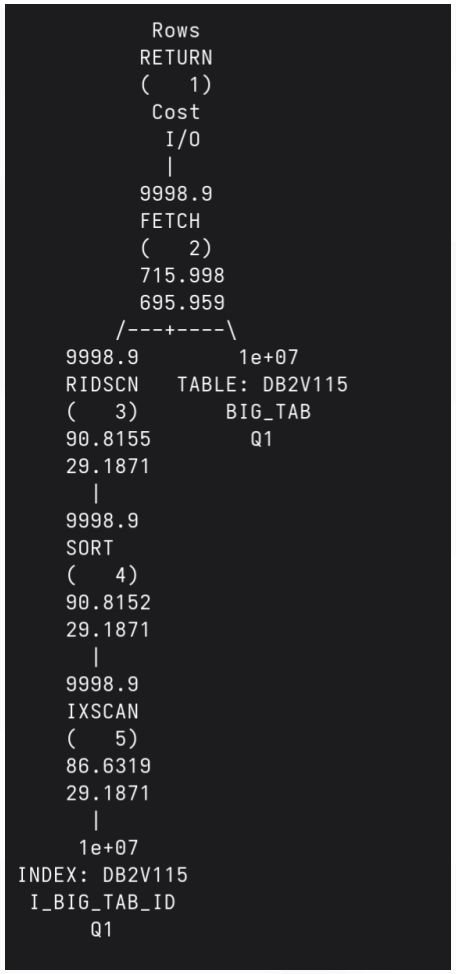
```
select
  *
from
  big_tab
where
  id = ?
```




```
select
  *
from
  big_tab
where
  id > 10000 and
  id < 20000
```

5) IXSCAN: (Index Scan)
   PREFETCH:
   SEQUENTIAL,READAHEAD

2) FETCH : (Fetch)
   MAX RIDS: 512
   PREFETCH: LIST



PREFETCHING

- **Sequential prefetching**

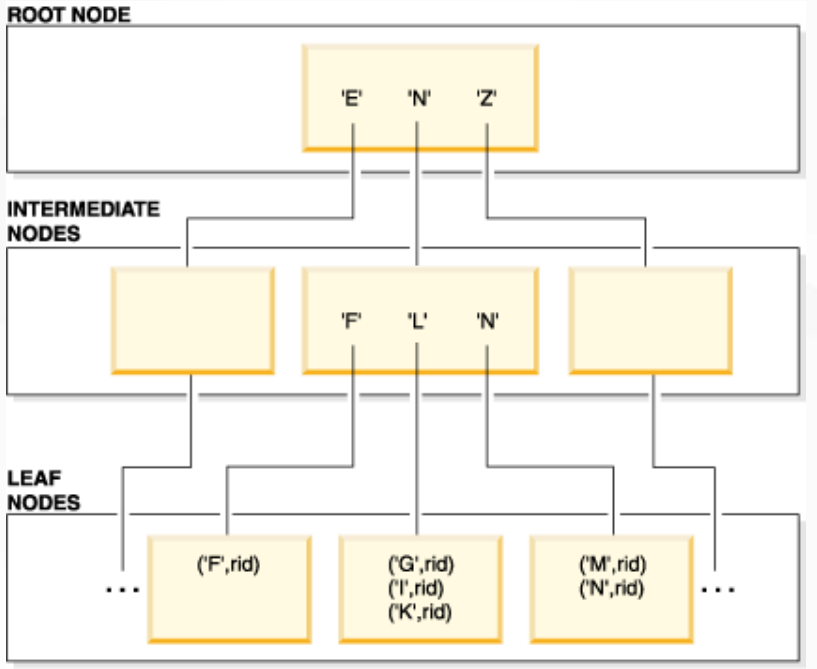
The prefetchers read consecutive pages. This is used if the database knows that all or most of the pages being read will be required, e.g. In TBSCAN access plan.

- **Readahead prefetching**

The prefetchers use indices to determine pages to be read.

- **List prefetching**

The prefetchers read data pages to read data specified in a list of values. Typically in Db2 optimizer access plans we see several branches with an index access to retrieve a list of RIDs. After sorting, the query processing uses this list to prefetch the relevant data pages via the prefetching system.



PAGE CLEANING

- Involved with UPDATE/INSERT/DELETE activity
- Write modified pages to disk in batches of pages
- Writes in the background by db2pfchr EDUs (NUM_IOCLEANERS) using OS async IO interfaces
- Write as late as possible
- Configurable by PAGE_AGE_TRGT_MCR
- DB2_USE_ALTERNATE_PAGE_CLEANING (default in 12.1)

PAGE CLEANING

```
db2pd -alldbs -dirty summary
Bufferpool: 1
   Dirty pages %       : 96589 / 100000 (96.59% dirty)
   Bufferpool minbuflsn: 0000000000057241
Oldest page info:
TspID PPNum   ObjID OPNum Typ UFlag    Pincount  pgLtch
3     0       5     0     0   4194306  0         0x00007F64B1E7B5A8

db2pd -alldbs -cleaner
Page Cleaners:
Clnr Idx   State Cycle      Task  Pgs Gthr  IO outstnd Max AIO   Pgs Thrsh  Pgs D Stl  Pgs Skppd
0          Work  18294      Thrsh 6         0          32        28880      43801      73
1          Work  18337      D Stl 6         0          32        28545      44328      61
2          Work  18406      D Stl 6         0          32        29106      43980      70
3          Work  18318      D Stl 6         0          32        28908      43962      53
4          Work  18351      D Stl 6         0          32        28824      44112      78
5          Work  18336      D Stl 6         0          32        28926      43830      58
```

DIRECT READS/WRITES IN DB2

- LOBs/CLOBs/XML do not use bufferpools
- Use INLINE LENGTH whenever possible (and 32k pages to increase the possibility of inlining)
- For large LOBs, use "LONG IN" and specify a table space with FILE SYSTEM CACHING (helps with reads)
- LOAD and BACKUP utilities do not use bufferpools (UTILHEAP)

OTHER DB2 IO CODE PATHS

- Backup is not using DIO for write target unless DB2_BACKUP_USE_DIO=ON
- Db2 logging uses DIO for current log, other logs are opened (for read) in buffered mode
- Log archival is using buffered mode (AIX: "rbw" mount option on logarchmeth1 can be considered)
- SysTmp by default use FS cache (due to extend+truncate activity)
- 11.5 is using async for MIRRORLOG by default
- Fast preallocation might slow down writes (GPFS,JFS2)

OS FILE SYSTEM LEVEL CACHING

- O_DIRECT + O_CIO prevents OS from caching Db2 IOs (we don't want double buffering except from LOBs)
- High FS cache pressure (lots of buffered writes) can affect the overall system performance
- Current FS cache info in /proc/meminfo (AIX: *vmstat –v*)

```
Cached:          14638488 kB
Active(anon):    14149508 kB
Inactive(anon):        0 kB
Active(file):    3153140 kB
Inactive(file):  9998068 kB
```

RESOURCES

- Backup: https://www.redbooks.ibm.com/abstracts/tips1344.html#4
- Prefetching: https://www.ibm.com/support/pages/system/files/inline-files/Db2%20Prefetching%20v2.0.0_1.pdf