Deduping & Compressing your Primary Oracle Data

Is it worth the trouble?

Bart Sjerps - Oracle Specialist - EMEA
Asad Samdani – Oracle Specialist - EMEA
http://bartsjerps.wordpress.com

DELL EMC

Agenda

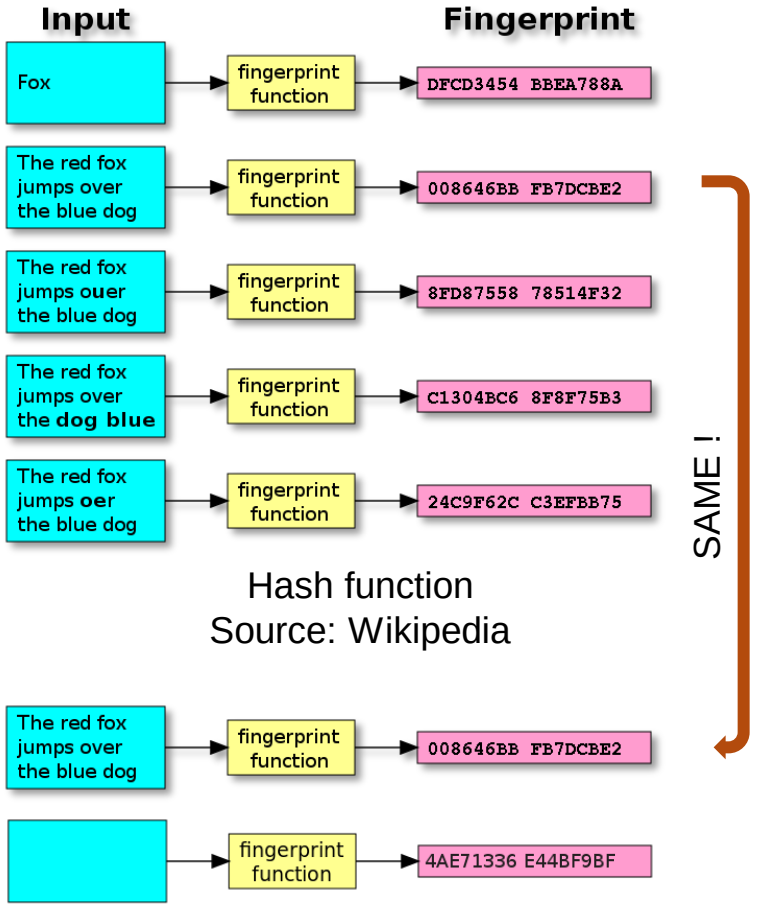
1. Short intro to All-Flash Arrays
2. What is Hash Allocation
3. Storage Compression
4. Introduction to QDDA and a look under the covers
5. Oracle datafiles
6. Live Demo

Modern All-Flash arrays

- Virtual (thin) provisioning
  - Blocks without content don't take up raw capacity
  - Blocks no longer used can be "zeroed" or discarded via special I/O call (i.e. "trim")
- Deduplication
  - Multiple chunks of identical data only get stored once and tracked by pointers
  - Chunk size varies
  - Multiple similar copies of databases hardly need additional space ("free" copies)
  - Typically implemented using a hashing algorithm (i.e. SHA-2)
- Compression
  - Chunks of data can be compressed
  - Compression algorithm varies (often LZ4 or similar)
  - Compressed block allocation algorithm varies, we assume "Bucket compression" (used in XtremIO)
- All these facilities are implemented inline (directly upon incoming write) or as post-processing jobs
  - Pros and cons to both methods, some arrays do a bit of both

Hash allocation

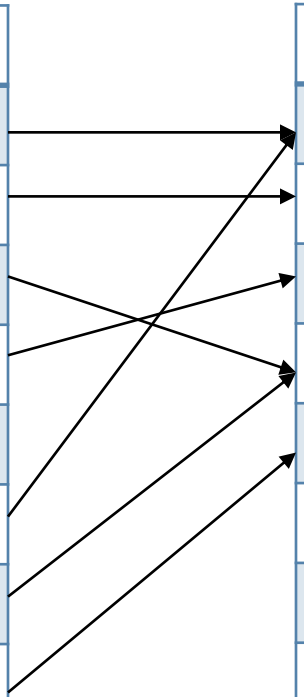
- Dedupe capable storage systems use hash algorithms to find duplicate blocks
  - Which hash algorithm is less important except for product engineers
- Zero blocks all have the same hash
  - Zero blocks don't get tracked, special case
  - How much raw capacity do we need to store 1GB of zeroed blocks?
- Doesn't matter HOW data gets there
  - Large or small I/O
  - Partial updates
  - DB restores, snapshots, host copy
  - As long as alignment is the same and data is the same



Hash allocation (2) – dedupe and thin in action

Offset	Data block	Hash
0	The	A4704FD3
1	Quick	809B7A80
2	Brown	ED63FC91
3	Fox	DE7B8FDC
4		00000000
5	The	A4704FD3
6	Brown	ED63FC91
7	Dog	C935D187
8		00000000

Logical LUN: 9 blocks



Hash	Data	Ref count
A4704FD3	The	1
809B7A80	Quick	1
DE7B8FDC	Fox	1
ED63FC91	Brown	2
C935D187	Dog	1
Free		
Free		
Free		
Free		

Raw pool: 5 blocks

Compression (older implementations)

Data (8K)	Compressed Bytes (ex)
The	3144
Quick	5643
Fox	8192
Brown	986
Dog	7016
Jumps	2130
Over	…
Lazy	

8K Block	Chunk 1	Chunk 2	Free
1	3144	986	(4062)
2	5634	2130	(428)
3	8192		
4	7016		(1176)
5	…	…	….

- Need to track offsets within blocks
- Need to uncompress entire block if we update
- Sometimes move unrelated parts of the block elsewhere leaving unusable gaps
- Requires garbage collection / defrag

Bad performance!

Compression (into buckets)

Data (8K)	Compressed Bytes (ex)
The	3144
Quick	5643
Fox	8192
Brown	986
Dog	7016
Jumps	2130
Over	…
Lazy	

8K Block	Buckets 2K, 4K and 8K			
1	2K	2K	2K	2K
2	4K		4K	
3	8K uncompressed			
4	4K		4K	
5	8K uncompressed			
6	8K uncompressed			

- Fixed offsets
- Only compress/uncompress buckets
- Never move entire block
- Free gaps can be re-used
- But… anything over 4K does not compress?

Good performance!

A final word on DellEMC arrays

- DellEMC XtremIO X2
  - Moved to 16K blocksize and much more bucketsizes (1K, 2K … 16K)
  - Much better compression ESPECIALLY for Oracle
  - Compression algorithm = CDERS (proprietary but much like LZ4)
  - Tradeoff: some blocks no longer dedupe due to 8K database/filesystem/dataset allocation (very minor drawback)
  - All operations ALWAYS inline (direct effect)
- DellEMC VMAX AFA 1st gen
  - Had compression on VMAX All-Flash
  - Compression chunk size 128K sorted into 8K, 16K … 128K, BUT:
  - Not all bucket sizes are always configured
  - Post-process
  - Not all data is compressed (hot data remains uncompressed)
- DellEMC VMAX AFA NG
  - Now also does dedupe @ 128K !
  - Compression algorithm deflate (same as GZIP): Higher compression ratio but more CPU overhead (there are always tradeoffs)
- Other DellEMC platforms
  - Varies by product

How to analyze YOUR data for dedupe/compress savings?

- “Commercial” or vendor provided analyze tools?
  - Sends obscure binary data back to the mothership (may not be appropriate)
  - Then wait a bunch of minutes and get a PowerPoint with a summary mailed to you
  - No details, no insight in where the numbers came from
  - Good for official vendor supported sizing
- “Freeware” on the internet
  - There are a few but limited (such as fixed blocksize) and require registration, still an obscure binary
  - Many focus on files instead of disks, some focus on backup vs primary data
  - Mostly closed-source
- Decided to write my own tool
  - Customer in South Africa complaining they didn’t get the data reduction promised by sales (…)
  - Said I could do it in a day (Well, … it was a long day…)
  - Called it QDDA (Quick & Dirty Dedupe Analyzer)

The Quick & Dirty Dedupe Analyzer

Figure out how much efficiency savings are possible

- Linux tool to scan disks, files and data streams for duplicate blocks
  - Any data, including Oracle
  - C++ with SQLite, LZ4, MD5
  - Multi-threaded (fast!)
  - Free (Open Source, GPL license)
- Can safely scan a running system
  - No need to run as root
  - Bandwidth throttle by default
- Configurable blocksize and array definitions
- Bonus
  - Compression and thin analysis
  - Run your own queries against the data

[QDDA Landing page]

The best thing about being me… There are so many “me”s.

— Agent Smith, The Matrix Reloaded

QDDA: How does it work? The basics

- Design
  - Written in C++ (best performance but tricky)
  - SQLite (embedded database, VERY fast and scalable, some limitations/caveats)
  - MD5 hashes
- Create an SQLite database
  - With a Key-Value store
  - Key is the hash, Value is the ref count *number of times we've seen the same block)
- Open a stream (file/disk/pipe)
  - Read a block
  - Calculate the hash of the block
  - Insert the hash in the key-value table or increase the reference count if it already exists
  - Repeat until no more blocks left
- Count the sum of hashes
  - This is how much capacity (in blocks) you would need after dedupe
- Later added compression
  - Calculate compressed size using LZ4 - so we now have a key-value-bytes table
  - Separate staging database because of performance
  - Lots of other improvements including multithreading
- Demonstrated to scan 7GB/s on a fast server (Dell 940)

And now: Oracle Data

Deduplication of Oracle data

- Oracle writes specific metadata to each datablock
  - So we expect no dedupe. True?
- What is it good for then? → Database copies!
  - Not important HOW the copies were made (storage snap, RMAN restore/image copy, host copy, …)
  - Every copy gets the same performance as the source
- DB cloning purposes
  - Refreshing test / dev / acceptance
  - Ad-hoc troubleshooting (“Firefighting”)
  - Creating Data Marts / staging areas (Data warehousing)
  - DB upgrade testing
  - Backup offload / instant restores
- **With dedupe, copies come for free !**

Compression of Oracle data

- Oracle data usually contains lots of compressible data
- Applies to all kinds of Oracle files
  - Datafiles, temp tables, redo/arch, rbs, backups …
- What about already compressed data?
  - BLOBs (images, documents, zip files)
  - Compressed tables
  - Hybrid Columnar Compression
- Encrypted?
  - No compress
  - Dedupes just fine

Demo Scenarios

1. Database (+DATA, +TEMP, +REDO)
2. Empty tablespace (datafile)
3. Full tablespace (Swingbench data, not compressed)
4. Full tablespace (Swingbench data, OLTP compressed)
5. ASM Diskgroup (after rebalance) - reclaim unused space
6. Backups

Performance

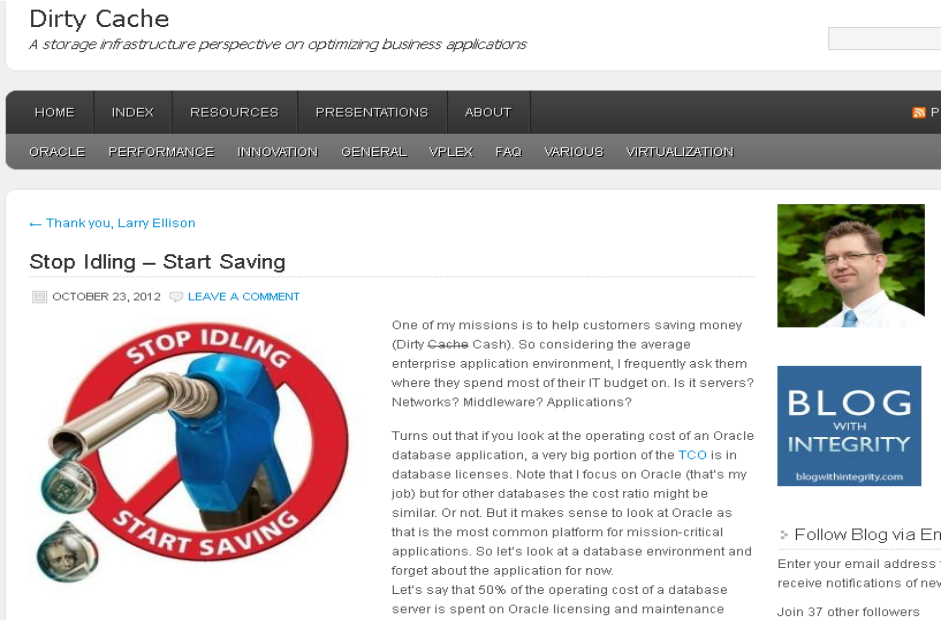
- Inline dedupe and compression on modern CPU is VERY fast
- Can be done AFTER I/O is complete (in protected data buffer)
  - Or post-process in some arrays
- On modern arrays, hash table sits entirely in memory (very fast lookups)
- Decompress upon read: small CPU overhead but less data to be retrieved -> better performance
- DB CPU is expensive (due to licensing) -> Better do compression elsewhere
- Array compresses/dedupes ALL data
  - Not just certain datafiles
- Copies are near unlimited, free and fast

References

My Blog “Dirty Cache”
http://bartsjerps.wordpress.com

Everything Oracle @ EMC (community):
http://emc.com/everythingoracle

Outrun (Open Source software depot):
http://outrun.nl

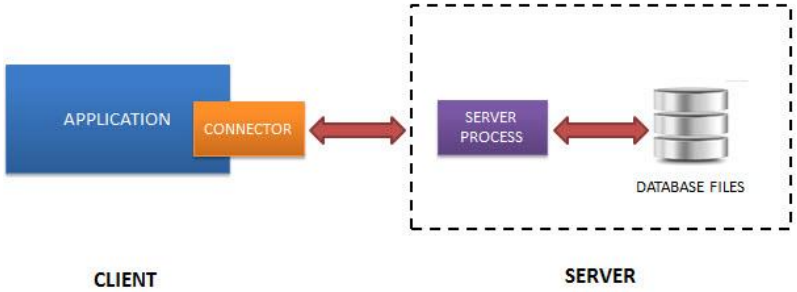


QDDA – Quick & Dirty Dedupe Analyzer
http://github.com/outrunnl/qdda
https://thecodeteam.com/projects/qdda/

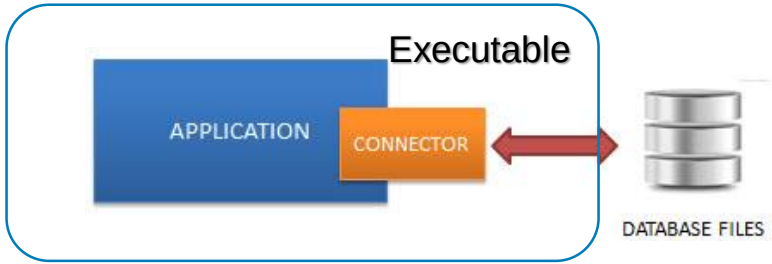
Questions?

Short into to SQLite

- Open Source (GPL)
- Becomes part of the executable (C++)
  - 1 large C file (~7M) linked with your program
  - No Network, server process, client processes
  - No security (handled by file permissions)
  - No Tablespaces (alternative: multiple SQLite files)
  - Administration (almost)
- Performance
  - Extremely fast (no overhead)
  - Writes are single threaded (!)
- Scalability
  - QDDA tested with ~ 20TB data @ 16K blocksize = 1 Billion rows
  - DB size: ~ 10 GB
  - Merge staging into final table → get some coffee ☺
- Limitations
  - Some…



Classic RDBMS i.e. Oracle, MySQL, …



Application with SQLite

SQLite database details

Hash (60 bits)	blocks	bytes
3032303536363739353438303936363O	1	6071
0	747	0
37383236363935323437303930323535	1	4133
34363731373937313637373234373535	2	8178
35333634373036353630313433323338	3	5945
38353234393732343932313036353435	1	13257
35373730393038373031363533373633	1	8663
38353234393732343932313036353435	2	10689
35373730393038373031363533373633	1	1588
37373132373932353435353532373036	1	1602
35333839383435373639363538353739	1	14868
…….	….	….

```
-- Primary table:
CREATE TABLE IF NOT EXISTS kv (
            hash unsigned integer primary key
,           blocks integer
,           bytes integer
) WITHOUT ROWID;

-- Staging table (separate SQLite file):
CREATE TABLE IF NOT EXISTS staging (
            id integer primary key autoincrement
,           hash integer
,           bytes integer
);

select count(*) from kv;    -- deduped blocks
select sum(blocks) from kv; -- original capacity

select blocks from kv
where hash=0;               -- zero blocks

-- Size of 1K compression bucket
select count(*) from kv
where bytes between 1 and 1024;
```

Oracle datafiles… do they compress?

- Short answer: Yes
- Empty tablespaces
  - Compress very well (16:1 on XtremIO X2)
  - Good alternative to real “thin provisioning” if you want to pre-allocate tablespaces
- Table data
  - Usually very good compression for regular datafiles (text/value based tables) -> expect 3:1
  - BLOB/CLOB -> not if they are already compressed (makes sense)
  - Indexes -> Yes
  - OLTP compressed data? YES! (but lower compression ratio, FYI OLTP “compression” isn’t compression)
  - HCC compressed data? No (already compressed)
- Other database files
  - Temp files / redo logs -> Yes
  - Archive logs -> Yes (copies of redo)
  - Backup files … Why do you keep them on primary storage? ☺ (but yes if not already compressed)

Oracle ASM

- Empty disk groups? -> Mostly zeroed so no compression but maximum thin savings
- Diskgroups after rebalance
  - Non-allocated AU's contain leftovers from previous data -> no dedupe, compression maybe -> Not optimal
  - ASRU (ASM Reclaim Utility)? -> can work but has corruption issues when running during ASM metadata operations, most DBAs avoid it due to the risk
  - ASMBalloon! (Work in progress) see https://github.com/outrunnl/asmballoon

Disk Alignment

Offset usually caused by old skool “fdisk” partitioning adding 63x512 bytes for boot sector & partition table (Still a problem even in modern Linux distros)
-> Use PARTED! See Disk alignment reloaded

Oops!