dt Compression & Deduplication

- 1. Overview
- 2. Deduplication Workloads
  - 2.1. Predefined Deduplication Workloads
  - 2.2. dt Deduplication Specifics

1. Overview

This document will describe the different methods and workloads available to perform compression and deduplication data patterns.

As many of you know, I am a proponent of using multiple I/O tools to provide overall storage verification. To that end, there are a number of tools available depending on your test requirements, including: *iozone*, *fio*, *vdbench*, and others that are excellent tools and provide options for compression and deduplication *percentages*. Therefore, I have not devoted much effort into replicating what other tools already provide, but instead focused on better triage and testing options.

That said, dt **does** provide ways to verify both these features. Years ago, when tape devices with compression were introduced, I created a pattern file that included compressible and non-compressible data patterns. This custom pattern file was used to provide the then tape compression verification. Based on the pattern file, I created a pattern file suitable for testing compression and deduplication, and documented how folks could create their own custom pattern files. Nonetheless, folks generally desire percentages for both, not just some level of coverage. 😒

FYI: I'll use the term *storage efficiency* to refer to compression and deduplication features.

2. Deduplication Workloads

Beyond using repeatable data patterns, another method I've seen implemented with file systems, is to create multiple files with the same data in whatever block size required to trigger deduplication. These tools usually refer to this method as the *data factor*. For example, creating two identical files generates a 2x data factor.

With this knowledge, I developed dt workloads to mimic this same style of deduplication.

2.1. Predefined Deduplication Workloads

I've added several predefined dt workloads to make it easier for folks to use, without knowing all the dt options required. The ones recently added include:

```
robin@DESKTOP-SBC6MG3 ~
$ dt workloads dedup
Valid Workloads:

    dt_dedup: Deduplication Pattern
        workload=dt_file_system_dedup
bufmodes=cachereads,buffered,unbuffered,cachewrites

    longevity_file_dedup: Longevity File System w/Dedup Workload
        workload=longevity_common min_limit=1m max_limit=2g incr_limit=vary
dispose=keep flags=direct notime=close,fsync oflags=trunc maxdatap=75 threads=4
pf=x:\noarch\dtdata\pattern_dedup

    longevity_disk_dedup: Longevity Direct Disk w/Dedup Workload
        workload=longevity_common capacityp=75 slices=4
pf=x:\noarch\dtdata\pattern_dedup

    dt_dedup_common: Deduplication Common Options (template)
        dsize=4k min=8k max=1m incr=vary enable=raw,reread,log_trailer,syslog
history=5 enable=history_timing logprefix='%seq %nos %et %prog (j:%job t:%thread): '
keepalivet=5m runtime=-1 onerr=abort noprogt=30s noprogtt=5m notime=close,fsync
end_delay=60 enable=secsdelay stopon=C:\temp\stopit

    dt_dedup_pattern_file: Deduplication Pattern File (template)
        pf=x:\noarch\dtdata\pattern_dedup

    dt_dedup_data_pattern: Deduplication Data Pattern (template)
        pattern=iot prefix='%U@%h'

    dt_disk_dedup: Direct Disk Deduplication Workload
        workload=dt_dedup_common,dt_dedup_pattern_file capacityp=75 slices=4
iodir=vary prefix='%s@%h'

    dt_file_system_dedup: File System Deduplication Workload 2x Data Factor
        workload=dt_dedup_common,dt_dedup_data_pattern dispose=keep flags=direct
maxdatap=75 files=2 limit=2g maxdatap=75 threads=4 pattern=iot prefix='%U@%h'

    disk_dedup: Direct Disk Deduplication
        workload=dt_disk_dedup


robin@DESKTOP-SBC6MG3 ~
$
```

Note: The dedup pattern file (*pattern_dedup)* can also be used with file systems, but please know this will **not** provide high data validation by itself. The dt pattern files are in the data/ subdirectory along with a script named *makededup* to help create custom deduplication files. You will need to specify the pattern file option (pf=) with the path to the desired pattern file.

But as stated earlier, we **do** have alternate tools for providing compression and deduplication percentages, so they can and should be used if % is desired.

2.2. dt Deduplication Specifics

For file system dedup, the same pattern data is written to each file. As shown, the prefix string includes a unique identifier (UUID), host name, and the IOT data pattern to provide some level of data validation. This same data is written to multiple files, two (2) files as defined above. Increasing the number of files will provide higher deduplication (data factor). Variable file sizes are **not** used, so the exact data limit is written to each file.

For disk dedup, a pattern file is used along with the serial number and host name, to generate the same data across each slice, per disk, per host. Please know, the dedup pattern file provides both compressible and duplicated data. Custom pattern files can be created.

FYI: This is the closest dedup workloads possible with dt options today, even though not ideal.

Please Note: I do realize folks often desire compression and deduplication percentages (%) offered by *iozone, fio,* or *vdbench*, but now that I'm retired and no longer have access to intelligent storage arrays with these features, it's unlikely they'll be added (sorry).