NAME

zpool-iostat - display logical I/O statistics for ZFS storage pools

SYNOPSIS

zpool iostat [[[**-c** *SCRIPT*] [**-lq**]]|**-rw**] [**-T u**|**d**] [**-ghHLnpPvy**] [*pool*<?>|[*pool vdev*<?>]|*vdev*<?>]
[*interval* [*count*]]

DESCRIPTION

Displays logical I/O statistics for the given pools/vdevs. Physical I/O statistics may be observed via iostat(1). If writes are located nearby, they may be merged into a single larger operation. Additional I/O may be generated depending on the level of vdev redundancy. To filter output, you may pass in a list of pools, a pool and list of vdevs in that pool, or a list of any vdevs from any pool. If no items are specified, statistics for every pool in the system are shown. When given an *interval*, the statistics are printed every *interval* seconds until killed. If **-n** flag is specified the headers are displayed only once, otherwise they are displayed periodically. If *count* is specified, the command exits after *count* reports are printed. The first report printed is always the statistics since boot regardless of whether *interval* and *count* are passed. However, this behavior can be suppressed with the **-y** flag. Also note that the units of **K**, **M**, **G**<?> that are printed in the report are in base 1024. To get the raw values, use the **-p** flag.

-c [*SCRIPT1*[,*SCRIPT2*]<?>]

Run a script (or scripts) on each vdev and include the output as a new column in the **zpool iostat** output. Users can run any script found in their *~/.zpool.d* directory or from the system */etc/zfs/zpool.d* directory. Script names containing the slash (/) character are not allowed. The default search path can be overridden by setting the **ZPOOL_SCRIPTS_PATH** environment variable. A privileged user can only run **-c** if they have the **ZPOOL_SCRIPTS_AS_ROOT** environment variable set. If a script requires the use of a privileged command, like smartctl(8), then it’s recommended you allow the user access to it in */etc/sudoers* or add the user to the */etc/sudoers.d/zfs* file.

If **-c** is passed without a script name, it prints a list of all scripts. **-c** also sets verbose mode (**-v**).

Script output should be in the form of "name=value". The column name is set to "name" and the value is set to "value". Multiple lines can be used to output multiple columns. The first line of output not in the "name=value" format is displayed without a column title, and no more output after that is displayed. This can be useful for printing error messages. Blank or NULL values are printed as a ’-’ to make output AWKable.

The following environment variables are set before running each script:

VDEV_PATH	Full path to the vdev
VDEV_UPATH	Underlying path to the vdev (*/dev/sd**). For use with device

mapper, multipath, or partitioned vdevs.

VDEV_ENC_SYSFS_PATH The sysfs path to the enclosure for the vdev (if any).

-T u|d Display a time stamp. Specify **u** for a printed representation of the internal representation of time. See time(1). Specify **d** for standard date format. See date(1).

-g Display vdev GUIDs instead of the normal device names. These GUIDs can be used in place of device names for the zpool detach/offline/remove/replace commands.

-H Scripted mode. Do not display headers, and separate fields by a single tab instead of arbitrary space.

-L Display real paths for vdevs resolving all symbolic links. This can be used to look up the current block device name regardless of the */dev/disk/* path used to open it.

-n Print headers only once when passed

-p Display numbers in parsable (exact) values. Time values are in nanoseconds.

-P Display full paths for vdevs instead of only the last component of the path. This can be used in conjunction with the **-L** flag.

-r Print request size histograms for the leaf vdev's I/O. This includes histograms of individual I/O (ind) and aggregate I/O (agg). These stats can be useful for observing how well I/O aggregation is working. Note that TRIM I/O may exceed 16M, but will be counted as 16M.

-v Verbose statistics Reports usage statistics for individual vdevs within the pool, in addition to the pool-wide statistics.

-y Normally the first line of output reports the statistics since boot: suppress it.

-w Display latency histograms:

total_wait	Total I/O time (queuing + disk I/O time).
disk_wait	Disk I/O time (time reading/writing the disk).
syncq_wait	Amount of time I/O spent in synchronous priority queues. Does not include disk time.
asyncq_wait	Amount of time I/O spent in asynchronous priority queues. Does not include disk time.
scrub	Amount of time I/O spent in scrub queue. Does not include disk time.
rebuild	Amount of time I/O spent in rebuild queue. Does not include disk time.

-l Include average latency statistics:

total_wait	Average total I/O time (queuing + disk I/O time).
disk_wait	Average disk I/O time (time reading/writing the disk).
syncq_wait	Average amount of time I/O spent in synchronous priority queues. Does not include disk time.
asyncq_wait	Average amount of time I/O spent in asynchronous priority queues. Does not include disk time.
scrub	Average queuing time in scrub queue. Does not include disk time.
trim	Average queuing time in trim queue. Does not include disk time.
rebuild	Average queuing time in rebuild queue. Does not include disk time.

-q Include active queue statistics. Each priority queue has both pending (**pend**) and active (**activ**) I/O requests. Pending requests are waiting to be issued to the disk, and active requests have been issued to disk and are waiting for completion. These stats are broken out by priority queue:

syncq_read/write	Current number of entries in synchronous priority queues.
asyncq_read/write	Current number of entries in asynchronous priority queues.
scrubq_read	Current number of entries in scrub queue.
trimq_write	Current number of entries in trim queue.
rebuildq_write	Current number of entries in rebuild queue.

All queue statistics are instantaneous measurements of the number of entries in the queues. If you specify an interval, the measurements will be sampled from the end of the interval.

EXAMPLES

Example 13: Adding Cache Devices to a ZFS Pool

The following command adds two disks for use as cache devices to a ZFS storage pool:

```
# zpool add pool cache sdc sdd
```

Once added, the cache devices gradually fill with content from main memory. Depending on the size of your cache devices, it could take over an hour for them to fill. Capacity and reads can be monitored using the **iostat** subcommand as follows:

```
# zpool iostat -v pool 5
```

Example 16: Adding output columns

Additional columns can be added to the **zpool status** and **zpool iostat** output with **-c**.

```
# zpool status -c vendor,model,size
  NAME     STATE  READ WRITE CKSUM vendor  model        size
  tank     ONLINE 0    0     0
  mirror-0 ONLINE 0    0     0
  U1       ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U10      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U11      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U12      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U13      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U14      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
```

```
# zpool iostat -vc size
              capacity     operations     bandwidth
pool        alloc   free   read  write   read  write  size
----------  -----  -----  -----  -----  -----  -----  ----
rpool       14.6G  54.9G      4     55   250K  2.69M
  sda1      14.6G  54.9G      4     55   250K  2.69M   70G
----------  -----  -----  -----  -----  -----  -----  ----
```

SEE ALSO

iostat(1), smartctl(8), zpool-list(8), zpool-status(8)