Article
Tony Pepper · May 25, 2016 5m read

Open Exchange

Random Read IO Storage Performance Tool

New Tool Available

Please see PerfTools IO Test Suite for a later version of the Random Read IO tool.

Purpose

This tool is used to generate random read Input/Output (IO) from within the database. The goal of this tool is to drive as many jobs as possible to achieve target IOPS and ensure acceptable disk response times are sustained. Results gathered from the IO tests will vary from configuration to configuration based on the IO sub-system. Before running these tests ensure corresponding operating system and storage level monitoring are configured to capture IO performance metrics for later analysis.

Methodology

Start with a small number of processes and 10,000 iterations per process. Use 100,000 iterations per process for all-flash storage arrays. Then increase the number of processes, e.g. start at 10 jobs and increase by 10, 20, 40 etc. Continue running the individual tests until response time is consistently over 10ms or calculated IOPS is no longer increasing in a linear way.

As a guide, the following response times for 8KB and 64KB Database Random Reads (non-cached) are usually acceptable for all-flash arrays:

- Average <= 2ms
- Not to exceed <= 5ms

The tool requires an empty pre-expanded IRIS.DAT database to be at least double the size of memory in the server and at least four times the storage controller cache size. The database needs to be larger than memory to ensure reads are not cached in file system cache.

The tool uses the ObjectScript VIEW command which reads database blocks in memory so if you are not getting your expected results then perhaps all the database blocks are already in memory.

Specifications and Targets

Complete the following table with your environment specifications and targets:

Specification	Example
Storage	Storage array specification
Physical Server	CPU, Memory specification
Virtual machine	Red Hat Enterprise Linux 7 24 vCPU, 40GB vRAM

Specification	Example
Database size	200GB
Shared memory	Allocated 26956MB shared memory using Huge Pa buffers, 1000MB routine buffers
Target IOPS	2000
Target response time	<=5ms

Installation

Download the **PerfTools.RanRead.xml** tool from GitHub here.

Import **PerfTools.RanRead.xml** into USER namespace.

```
USER> do $system.OBJ.Load("/tmp/PerfTools.RanRead.xml","ckf")
```

Run the Help method to see all entry points. All commands are run in %SYS.

```
USER> do ##class(PerfTools.RanRead).Help()

InterSystems Random Read IO Performance Tool
--------------------------------------------
do ##class(PerfTools.RanRead).Setup(Directory,DatabaseName,SizeGB,LogLevel)
    - Creates database and namespace with the same name. The log level must be in the
 range of 0 to 3, where 0 is “none” and 3 is “verbose”.
do ##class(PerfTools.RanRead).Run(Directory,Processes,Iterations)
    - Run the random read IO test.
do ##class(PerfTools.RanRead).Stop()
    - Terminates all background jobs.
do ##class(PerfTools.RanRead).Reset()
    - Deletes all random read history stored in ^PerfTools.RanRead*
do ##class(PerfTools.RanRead).Export(directory)
    - Exports a summary of all random read test history to tab delimited text file.
```

Setup

Create an empty (pre-expanded) database called ZRANREAD approximately twice the size of the memory of the physical host to be tested. Ensure empty database is at least four times the storage controller cache size. You can create manually or use the following method to automatically create a namespace and database.

```
USER> do ##class(PerfTools.RanRead).Setup("/usr/iris/db/zranread","ZRANREAD",100,1)

Creating 100GB database in /usr/iris/db/zranread/
Database created in /usr/iris/db/zranread/
Run %Installer Manifest...
2016-05-23 13:33:59 0 PerfTools.RanRead: Installation starting at 2016-05-23 13:33:59
, LogLevel=1
2016-05-23 13:33:59 1 CreateDatabase: Creating database ZRANREAD in /usr/iris/db/zran
```

```
read// with resource
2016-05-23 13:33:59 1 CreateNamespace: Creating namespace ZRANREAD using ZRANREAD/ZRA
NREAD
2016-05-23 13:33:59 1 ActivateConfiguration: Activating Configuration
2016-05-23 13:34:00 1 EnableEnsemble: Enabling ZRANREAD
2016-05-23 13:34:00 1 ActivateConfiguration: Activating Configuration
2016-05-23 13:34:00 0 PerfTools.RanRead: Installation succeeded at 2016-05-23 13:34:0
0
2016-05-23 13:34:00 0 %Installer: Elapsed time 1.066633s
Database /usr/iris/db/zranread/ ready for testing.
do ##class(PerfTools.RanRead).Run(directory,processes,iterations) e.g.
do ##class(PerfTools.RanRead).Run("/usr/iris/db/zranread/",1,10000)
```

Run

Execute the Run method increasing the number of processes and taking note of the response time as you go.
If the tests are too quick or the results are not as expected then increase the number of iterations to 10000.

```
USER> do ##class(PerfTools.RanRead).Run("/usr/iris/db/zranread",20,10000)

InterSystems Random Read IO Performance Tool
----------------------------------------
Starting 20 jobs in the background.
To terminate jobs run:  do ##class(PerfTools.RanRead).Stop()
Waiting for jobs to finish.....................
Random read background jobs finished.
20 processes (1000 iterations) average response time = 7.18ms
Calculated IOPS = 2787
```

Results

The results for each run are saved in USER in SQL table PerfTools.RanRead. Run the following SQL query to see a summary of results.

```
SELECT RunDate,RunTime,Database,Iterations,Processes,
    {fn ROUND(AVG(ResponseTime),2)} As ResponseTime,
    {fn ROUND(AVG(IOPS),0)} As IOPS
FROM PerfTools.RanRead
GROUP BY Batch
```

To export the result set to a tab delimited text file run the following:

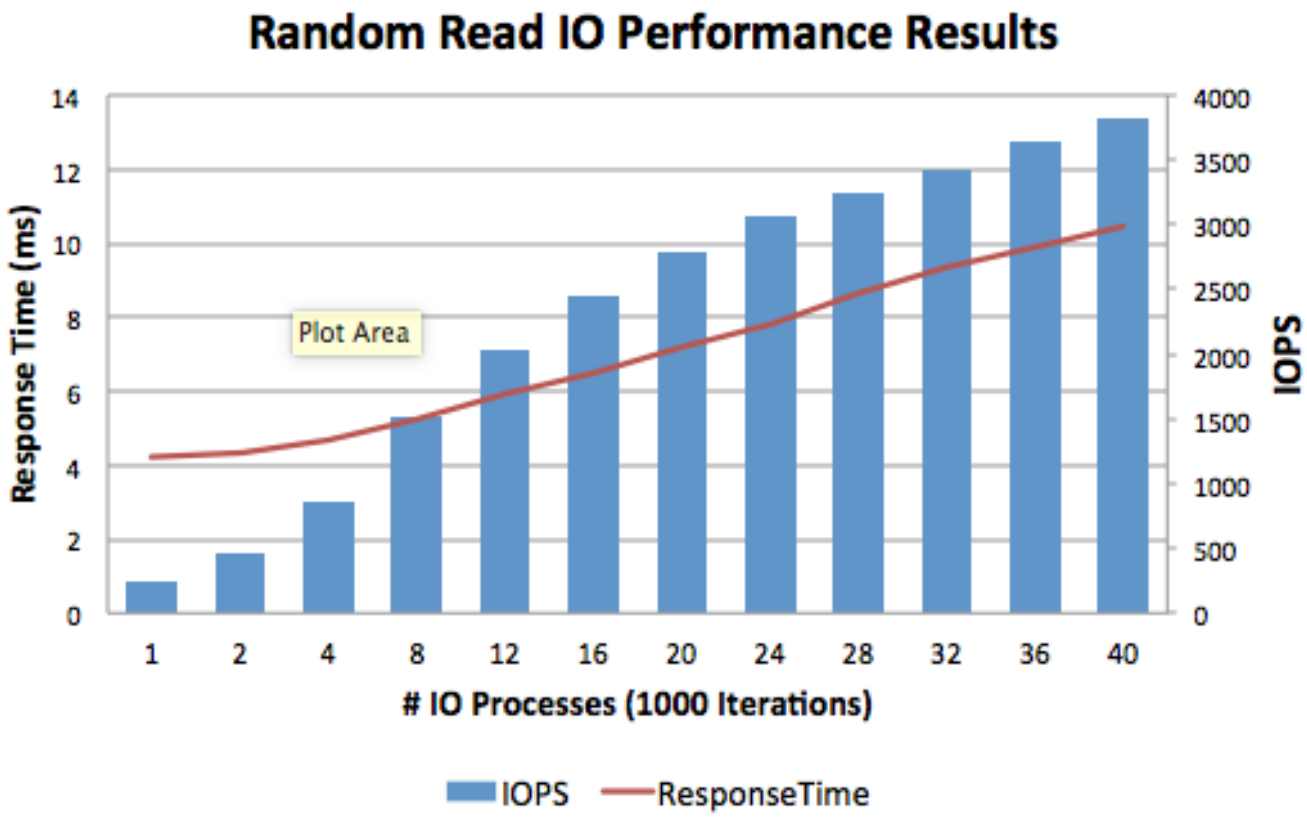
```
USER> do ##class(PerfTools.RanRead).Export("/usr/iris/db/zranread/")

Exporting summary of all random read statistics to /usr/iris/db/zranread/PerfToolsRan
Read_20160523-1408.txt
Done.
```

Analysis

Open the exported text file in Excel, then copy and paste into the PerfToolsRandomReadAnalysisTemplate.xlsx

spreadsheet for charting.



The sample spreadsheet can be downloaded from GitHub here.

Clean Up

After finished running the tests purge the history by running:

```
%SYS> do ##class(PerfTools.RanRead).Reset()
```

#Best Practices #Databases #Performance #InterSystems IRIS
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/random-read-io-storage-performance-tool