Article
Murray Oldfield · Mar 8, 2016 8m read

InterSystems Data Platforms and performance – Part 1

Your application is deployed and everything is running fine. Great, hi-five! Then out of the blue the phone starts to ring off the hook – it ' s users complaining that the application is sometimes ' slow ' . But what does that mean? Sometimes? What tools do you have and what statistics should you be looking at to find and resolve this slowness? Is your system infrastructure up to the task of the user load? What infrastructure design questions should you have asked before you went into production? How can you capacity plan for new hardware with confidence and without over-spec'ing? How can you stop the phone ringing? How could you have stopped it ringing in the first place?

A list of other posts in this series is here

This will be a journey

This is the first post in a series that will explore the tools and metrics available to monitor, review and troubleshoot systems performance as well as system and architecture design considerations that effect performance. Along the way we will head off down a quite a few tracks to understand performance for Caché, operating systems, hardware, virtualization and other areas that become topical from your feedback in the comments.

We will follow the feedback loop where performance data gives a lens to view the advantages and limitations of the applications and infrastructure that is deployed, and then back to better design and capacity planning.

It should go without saying that you should be reviewing performance metrics constantly, it is unfortunate the number of times customers are surprised by performance problems that would have been visible for a long time, if only they were looking at the data. But of course the question is - what data? We will start the journey by collecting some basic Caché and system metrics so we can get a feel for the health of your system today. In later posts we will dive into the meaning of key metrics.

There are many options available for system monitoring – from within Caché and external and we will explore a lot of them in this series.

To start we will look at my favorite go-to tool for continuous data collection which is already installed on every Caché system – ^pButtons.

To make sure you have the latest copy of pButtons please review the following post:

https://community.intersystems.com/post/intersystems-data-platforms-and-performance-%E2%80%93-how-update-pbuttons

Collecting system performance metrics - ^pButtons

The Caché pButtons utility generates a readable HTML performance report from log files it creates. Performance metrics output by pButtons can easily be extracted, charted and reviewed.

Data collected in the pButtons html file includes;

- Caché set up: with configuration, drive mappings, etc.
- mgstat: Caché performance metrics - most values are average per second.

- Unix: vmstat and iostat: Operating system resource and performance metrics.
- Windows: performance monitor: Windows resource and performance metrics.
- Other metrics that will be useful.

pButtons data collection has very little impact on system performance, the metrics are being collected already by the system, pButtons simply packages these for easy filing and transport.

To keep a baseline, for trend analysis and for troubleshooting it is good practice to collect a 24-hour pButtons (midnight to midnight) every day for a complete business cycle. A business cycle could be a month or more, for example to capture data from end of month processing. If you do not have any other external performance monitoring or collection you can run pButtons year-round.

The following key points should be noted:

- Change the log directory to a location away from production data to store accumulated output files to avoid disk full problems!
- Run an operating system script or otherwise compress and archive the pButtons file regularly, this is especially important on Windows as the files can be large.
- Review the data regularly!

In event of a problem needing immediate analysis pButtons data can be previewed (collected immediately) while metrics continue to be stored for collection at the end of the days run.

For more information on pButtons including preview, stopping a run and adding custom data gathering please see the *Caché Monitoring Guide* in the most recent Caché documentation:

http://docs.intersystems.com

The pButtons HTML file data can be separated and extracted (to CSV files for example) for processing into graphs or other analysis by scripting or simple cut and paste. We will see examples of the output in graphs later in the next post.

Of course if you have urgent performance problems contact the WRC.

Schedule 24 hour pButtons data collection

^pButtons can be started manually from the terminal prompt or scheduled. To schedule a 24-hour daily collection:

1. Start Caché terminal, switch to %SYS namespace and run pButtons manually once to set up pButtons file structures:

```
%SYS>d ^pButtons Current log directory: /db/backup/benchout/pButtonsOut/
Available profiles:
     1  12hours     - 12 hour run sampling every 10 seconds
     2  24hours     - 24 hour run sampling every 10 seconds
     3  30mins      - 30 minute run sampling every 1 second
     4  4hours      - 4 hour run sampling every 5 seconds
     5  8hours      - 8 hour run sampling every 10 seconds
     6  test        - A 5 minute TEST run sampling every 30 seconds
```

Select option 6. for test, 5 minute TEST run sampling every 30 seconds. Note your numbering may be different, but the test should be obvious.

During the run, run a Collect^pButtons (as shown below), you will see information including the runid. In this case " 20160303_1851_test " .

```
%SYS>d Collect^pButtons
Current Performance runs:
    20160303_1851_test
        ready in 6 minutes 48 seconds nothing available to collect at the moment.
%SYS>
```

Notice that this 5 minute run has 6 minutes and 48 seconds to go? pButtons adds a 2 minute grace period to all runs to allow time for collection and collation of the logs into html format.

2. IMPORTANT! Change pButtons log output directory – the default output location is the <cache install path>/mgr folder. For example on unix the path to the log directory may look like this:

```
do setlogdir^pButtons("/somewhere_with_lots_of_space/perflogs/")
```

Ensure Caché has write permissions for the directory and there is enough disk space available for accumulating the output files.

3. Create a new 24 hour profile with 30 second intervals by running the following:

```
write $$addprofile^pButtons("My_24hours_30sec","24 hours 30 sec interval",30,2880)
```

Check the profile has been added to pButtons:

```
%SYS>d ^pButtons
Current log directory: /db/backup/benchout/pButtonsOut/
Available profiles:
     1  12hours     - 12 hour run sampling every 10 seconds
     2  24hours     - 24 hour run sampling every 10 seconds
     3  30mins      - 30 minute run sampling every 1 second
     4  4hours      - 4 hour run sampling every 5 seconds
     5  8hours      - 8 hour run sampling every 10 seconds
     6  My_24hours_30sec- 24 hours 30 sec interval
     7  test        - A 5 minute TEST run sampling every 30 seconds

select profile number to run:
```

Note: You can vary the collection interval – 30 seconds is fine for routine monitoring. I would not go below 5 seconds for a routine 24 hour run (… ” ,5,17280) as the output files can become very large as pButtons collects data at every tick of the interval. If you are trouble-shooting a particular time of day and want more granular data use one of the default profiles or create a new custom profile with a shorter time period, for example 1 hour with 5 second interval (… ” ,5,720). Multiple pButtons can run at the same time so you could have a short pButtons with 5 second interval at running at the same time as the 24-hour pButtons.

4. Tip For UNIX sites review the disk command. The default parameters used with the 'iostat' command may not include disk response times. First display what disk commands are currently configured:

```
%SYS>zw ^pButtons("cmds","disk")
^pButtons("cmds","disk")=2
^pButtons("cmds","disk",1)=$lb("iostat","iostat ","interval"," ","count"," > ")
^pButtons("cmds","disk",2)=$lb("sar -d","sar -d ","interval"," ","count"," > ")
```

In order to collect disk statistics, use the appropriate command to edit the syntax for your UNIX installation. Note the trailing space. Here are some examples:

```
LINUX:     set $li(^pButtons("cmds","disk",1),2)="iostat -xt "
AIX:       set $li(^pButtons("cmds","disk",1),2)="iostat -sadD "
VxFS:      set ^pButtons("cmds","disk",3)=$lb("vxstat","vxstat -g DISKGROUP -i ","int
erval"," -c ","count"," > ")
```

You can create very large pButton html files by having both iostat and sar commands running. For regular performance reviews I usually only use iostat. To configure only one command:

```
set ^pButtons("cmds","disk")=1
```

More details on configuring pButtons are in the online documentation.

5. Schedule pButtons to start at midnight in the `Management Portal > System Operation > Task Manager`:

```
Namespace: %SYS
 Task Type: RunLegacyTask
 ExecuteCode: Do run^pButtons("My_24hours_30sec")
 Task Priority: Normal
 User: superuser
 How often: Once daily at 00:00:01
```

Collecting pButtons data

pButtons shipped in more recent versions of InterSystems data platforms include automatic collection. To manually collect and collate the data into an html file; In %SYS namespace, run the following command to generate any outstanding pButtons html output files:

```
do Collect^pButtons
```

The html file will be in the logdir you set at step 2 (if you did not set it go and **do it now!**). Otherwise the default location is the <Caché install dir/mgr>

Files are named <hostnameinstanceNamedatetimeprofileName.html> e.g. vsan-tc-db1H20152016021802 55test.html

Windows Performance Monitor considerations

If the operating system is Windows then Windows Performance Monitor (perfmon) can be used to collect data in synch with the other metrics collected. On older Caché distributions of pButtons, Windows perfmon needs to be configured manually. If there is demand from the post comments I will write a post about creating a perfmon template to define the performance counters to monitor and schedule to run for the same period and interval as pButtons.

Summary

This post got us started collecting some data to look at. Later in the week I will start to look at some sample data and what it means. You can follow along with data you have collected on your own systems. See you then.

http://docs.intersystems.com

#Best Practices #InterSystems Business Solutions and Architectures #Performance #System Administration #Caché #InterSystems IRIS #InterSystems IRIS for Health

Source URL:https://community.intersystems.com/post/intersystems-data-platforms-and-performance-%E2%80%93-part-1