Article
Bob Binstock · Apr 26, 2021 9m read

Scaling Cloud Hosts and Reconfiguring InterSystems IRIS

Like hardware hosts, virtual hosts in public and private clouds can develop resource bottlenecks as workloads increase. If you are using and managing InterSystems IRIS instances deployed in public or private clouds, you may have encountered a situation in which addressing performance or other issues requires increasing the capacity of an instance's host (that is, *vertically scaling*).

One common reason to scale is insufficient memory. As described in Memory Management and Scaling for InterSystems IRIS in the *Scalability Guide,* providing enough memory for all of the entities running on the host of an InterSystems IRIS instance under all normal operating circumstances is a critical factor in both performance and availability. In one common scenario, as the workload of an InterSystems IRIS instance increases, its working set becomes too large to be held by the instance's allocated database cache. This forces some queries to fall back to disk, greatly increasing the number of disk reads required and creating a major performance problem. Increasing the size of the cache solves that problem, but if doing so would leave insufficient memory remaining for other purposes, you also need to increase the total physical memory on the host to avoid pushing the bottleneck to another part of the system.

Fortunately, scaling a virtual host is typically a lot easier than scaling hardware. This post discusses the two stages of the process:

- Scaling the cloud host's resources

You can change the resource specification of a virtual host on AWS, GCP, and Azure, using the platform's command line, API, or portal. VMWare vSphere allows you to easily change a number of resource parameters for a VM through its vSphere Client interface..

- Reconfiguring InterSystems IRIS to take advantage of the scaled resources

There are a number of ways to reconfigure InterSystems IRIS to take advantage of scaled host resources. This document describes the use of the configuration merge feature, which merges new parameter values, specified in a merge file, into an instance's CPF. Configuration merge is an easy and effective method because it lets you address only the configuration settings you want to modify, make multiple changes to an instance's configuration in one operation, and easily make the same set of changes to multiple instances.

The procedures described here are manual, but in production they would very likely be automated, for example using a script that would apply a specific merge file in an accessible location to a list of instances.

Scaling Cloud Host Resources

Public cloud platforms provide a range of resource templates to choose from that specify CPU, memory, network interfaces, and other resources for virtual hosts (storage is provisioned and sized separately). To resize a host, you change the template selected when the host was provisioned to one that specifies more of the resources you want to increase. On Amazon Web Services, the resource template is called an instance type; for example, the **t3.large** instance type specifies 2 CPUs and 8 GB of memory. On Google Cloud Platform it's a machine type, such as the **e2-standard-2** (which also includes 2 CPUs and 8 GB), and on Microsoft Azure it's a size (the **StandardB2ms** calls for the same 2 CPUs and 8 GB). By redefining the instance type, machine type, or size of an existing public cloud host, you can scale its resource specifications. In a VMware vSphere private cloud, you can use the vSphere Client interface to the vCenter Server management console to directly modify one or

more individual resource settings of an existing virtual machine. (You can also simultaneously scale groups of hosts on each platform.)

The following sections provide brief examples of resizing individual virtual hosts on the various platforms, with links to the documentation for all available methods. Please note that these methods (APIs, command line interfaces, and portal interfaces) are offered and maintained by the cloud vendors, and the examples included here are for informational purposes, to illustrate how easily you can adapt InterSystems IRIS to take advantage of increased resources

AWS

To modify the instance type of an AWS host (called an *instance*, not to be confused with an InterSystems IRIS instance), you can use the **modify-instance-attribute** CLI command, as shown in the following example:

```
$ aws ec2 describe-instances --instance-ids i-01519f663af48a55e
{
   "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0abcdef1234567890,
            "InstanceId": "i-1234567890abcdef0,
            "InstanceType": "m5n.large",
            ...
$ aws ec2 stop-instances --instance-ids i-01519f663af48a55e
{
    "StoppingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            ...
$ aws ec2 describe-instances --instance-ids i-01519f663af48a55e
{
   "Instances": [
        {
            ...
            "State": {
            "Code": 80,
            "Name": "stopped"
            }
            ...
$ aws ec2 modify-instance-attribute --instance-ids i-01519f663af48a55e \
      --instance-type "{\"Value\": \"m5n.xlarge\"}"
$ aws ec2 start-instances --instance-ids i-01519f663af48a55e
{
    "StartingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            ...
$ aws ec2 describe-instances --instance-ids i-01519f663af48a55e
{
   "Instances": [
```

```
{
    "AmiLaunchIndex": 0,
    "ImageId": "ami-0abcdef1234567890,
    "InstanceId": "i-1234567890abcdef0,
    "InstanceType": "m5n.xlarge",
    ...
```

You can also make this change using the ModifyInstanceAttribute AWS API call or the AWS EC2 console.

GCP

To modify the machine type of a GCP host (also known as an *instance*), you can use the gcloud CLI to stop, modify, and restart the instance. For example, you would use the following commands to change the machine type of an instance named **scalingTest** to **n1-highmem-96**:

```
$ gcloud compute instances stop scalingTest
$ gcloud compute instances set-machine-type scalingTest --machine-type n1-highmem-32
$ gcloud compute instances start scalingTest
```

You can also make this change using the Google Cloud Console or GCP API.

Azure

When you use the Azure CLI to modify the size of a Linux VM , you can view a list of the sizes available on the hardware cluster where the VM is hosted using the **list-vm-resize-options** command, for example:

```
az vm list-vm-resize-options --resource-
group testingGroup --name scalingTest --output table
```

You can then use the **resize** command to change the VM's size to one of the listed options, as shown. This command restarts the VM automatically.

```
az vm resize --resource-group testingGroup --name scalingTest --size Standard_E32d_v4
```

If the size you want to change the VM to is not available, you can deallocate the VM, which can then be resized to any size supported by the region and restarted; the commands involved are illustrated below:

```
az vm deallocate --resource-group testingGroup --name scalingTest
az vm resize --resource-group testingGroup --name scalingTest --size Standard_M128s
az vm start --resource-group testingGroup --name scalingTest
```

You can resize a Windows VM on Azure using either the Azure portal or Powershell commands.

vSphere

To resize a VMware vSphere VM, do the following:

1. Open vSphere Client or Web Client and display the VM inventory.
2. Right-click the VM you want to modify and select **Edit Settings.**
3. On the **Virtual Hardware tab**,
    - Expand **Memory** and change the amount of RAM configured for the VM.
    - Expand **CPU** and change the number of cores and optionally the number of cores per socket.

- Make any other desired changes to the hardware resources allocated to the VM.

Reconfiguring InterSystems IRIS for Scaled Resources

Once you have scaled the host, the next step is to reconfigure InterSystems IRIS to take advantage of the increased resources by changing one or more parameters in the instance's configuration parameter file (CPF). For example, to continue with the common scenario mentioned at the start of this post, now that you have increased the host's memory resources, you will want to take advantage of this by increasing the size of the InterSystems IRIS instance's database cache (which is done by changing the value of the globals parameter) so it can keep more data in memory.

An easy way to make such a change, and far the easiest and most repeatable way to make multiple changes to an instance's configuration in one operation or make the same changes to multiple instances, is to use the configuration merge feature, which is available on UNIX® and Linux systems. As described in Using Configuration Merge to Deploy Customized InterSystems IRIS Instances in *Running InterSystems Products in Containers* and Using the Configuration Merge Feature in the *Configuration Parameter File Reference*, configuration merge lets you specify a merge file containing the settings you want merged into an instance's CPF immediately prior to a restart. (In release 2021.1 you'll be able to do this on a running instance without restarting it.) Not only is this more convenient than editing an instance's CPF directly, but it is highly repeatable across multiple instances, and supports reliable change management by enabling you to keep an accurate record of changes simply by versioning the configuration merge files you apply.

To execute a configuration merge, you need to do the following:

1. Create the merge file with the parameters you want to modify.
2. Stage the merge file in a location accessible to the instance. If the instance you are modifying is in a container (which is likely on a cloud host), you can stage the file in the instance's durable %SYS directory (see Durable %SYS for Persistent Instance Data in *Running InterSystems Products in Containers*).
3. Specify the merge file's location using the **ISC_CPF_MERGE_FILE** environment variable before restarting the instance.

For example, continuing with the case of the database cache that needs updating, suppose you wanted a containerized instance's database cache size increased to 100 GB. The setting for this, in the **[config]** section of the CPF, would be **globals=102400**, which sets the database cache for 8-kilobyte blocks to 102,400 MB, or 100 GB. (As explained in the globals description in the *Configuration Parameter File Reference,* the parameter sets the size of the cache for multiple block sizes; if only one value is provided, however, it is applied to the 8-kilobyte block size, and **0** [zero] is assumed for the other sizes; **globals=102400** is therefore the equivalent of **globals=0,0,102400,0,0,0**.)

To make this change, you might do the following on the cloud host:

1. Create a configuration merge file, called for example **mergefile2021.06.30.cpf**, containing these lines:

```
[config]
globals=102400
```

2. Stage the merge file in the durable %SYS directory on the host's file system, which if you mounted the external volume **/data** as **/external** in the container and used the **ISC_DATA_DIRECTORY** variable to specify **/external/iris_durable** as the durable %SYS directory for the instance, would be **/data/iris_durable** .

3. Use the **docker exec** command on the host's command line to specify the variable and restart the instance with the **iris** command; if the instance's container is named **iris** and the instance is named **IRIS**, for example, the command would look like this:

```
docker exec iris ISC_CPF_MERGE_FILE=/data/iris_durable/mergefile2021.06.30.cpf iris s
top IRIS restart
```

4. When the instance is restarted, you can confirm the new **globals** setting with this command:

```
docker exec iris grep globals /data/iris_durable/iris.cpf
```

#AWS #Azure #Best Practices #Cloud #Google Cloud Platform (GCP) #Performance #System Administration #InterSystems IRIS #InterSystems IRIS for Health