



Lenovo ThinkAgile CP Oversubscription (Technical Brief)



Models: CP 4000, CP 6000

Note

Before using this information and the product it supports, be sure to read and understand the safety information and the safety instructions, which are available at the following address:

http://thinksystem.lenovofiles.com/help/topic/safety_documentation/pdf_files.html

In addition, be sure that you are familiar with the terms and conditions of the Lenovo warranty for your solution, which can be found at the following address:

<http://datacentersupport.lenovo.com/warrantylookup>

First Edition (March 2020)

© Copyright Lenovo 2019, 2020.

LIMITED AND RESTRICTED RIGHTS NOTICE: If data or software is delivered pursuant to a General Services Administration "GSA" contract, use, reproduction, or disclosure is subject to restrictions set forth in Contract No. GS-35F-05925.

Contents

Contents	i	CPU oversubscription	5
Oversubscription in ThinkAgile CP	1	Memory oversubscription	5
How oversubscription works in ThinkAgile CP	1	Monitoring CPU and memory.	5
Oversubscription metrics	1	Enabling and disabling memory oversubscription.	6
Oversubscription at migration zone level	2	Service files	6
Oversubscription at the node level.	3	Enable and disable memory oversubscription	6
Oversubscription considerations	5		

Oversubscription in ThinkAgile CP

Oversubscription in ThinkAgile CP refers to various methods by which more resources than are available on the physical host can be assigned to the virtual machines (VMs) that host supports. In general, administrators can oversubscribe processing, memory, and storage resources in virtual machines.

Overprovisioning of processing, memory, and storage can help you maximize resource utilization in your virtual environment. However, with oversubscription, administrators are basically assigning to VMs more resources than are available on the host. In other words, if all of the VMs suddenly requested access to all of their allocated resources, the host would not have enough resources to service these requests. While resource oversubscription increases VM density, it carries some risks. Once an oversubscribed resource is exhausted, stability issues and major performance problems can affect all workloads running on the host server. Therefore, it is important to overprovision carefully and wisely.

Note: Subscription planning should include planning for high-availability events. For example, if a physical compute node is stopped because of maintenance or for failure, the application instances running on it will need to be allocated onto other available resources. If the subscription factor is too high, some or all of the instances will not be able to be run.

How oversubscription works in ThinkAgile CP

When compute oversubscription is used, the ThinkAgile CP hypervisor must invoke processor scheduling to distribute processor time to the VMs, or application instances, that need it.

For example, if we oversubscribe by 5:1, then each physical processor core is supporting five vCPUs, causing a drop in performance.

Note: In ThinkAgile CP, we refer to a VM as an application instance.

To handle and mitigate the effects of memory oversubscription, ThinkAgile CP uses a technique called memory ballooning. A memory ballooning driver is installed inside the guest VM. Idle or unused memory pages are assigned to the driver. The balloon driver then “pins” those pages and reports this back to the hypervisor. If the host becomes low on physical memory, guest memory pages are assigned to the balloon driver. The host can then reclaim these memory pages to address the needs of other VMs that need the RAM. When a VM has RAM to spare, it can transparently share that RAM with other VMs on the same host. This enables the host to achieve higher levels of VM density.

The ballooning process brings to RAM a sort of thin provisioning capability. The ballooning process does require some processing overhead, which is imperceptible. Swapping to disk is used as a last resort. Swapping is a process by which the hypervisor moves the least-used memory pages to disk. Those memory pages are still accessible, but when they are required, they must be retrieved from disk. Swapping noticeably degrades the host’s overall performance.

Oversubscription metrics

ThinkAgile CP offers oversubscription at two levels.

- **Migration zone** – Migration zone oversubscription is set as virtual data centers are created.
- **Node** – Node-level oversubscription is set as application instances are provisioned inside virtual data centers.

Both levels are largely static quantities whose values are a function of the virtual data centers created and the applications provisioned. Additionally, ThinkAgile CP also maintains the runtime load at the migration zone

and node levels. These levels are dynamic quantities whose values change with time and depend on what resources the applications actually consume when running. The migration zone level runtime load is a weighted average of the various node level runtime loads. ThinkAgile CP maintains both compute and memory loads at the migration zone and node levels.

Note: When a VM, or application instance, is first started in ThinkAgile CP, it consumes 100% of its provisioned memory resources. Then, over time (usually within a few minutes), the memory usage will scale back to what the instance is using. Because of this, an instance can only be started on a compute node if that node has enough available memory for the instance's provisioned memory.

Oversubscription at migration zone level

Oversubscription at virtual data center creation time is calculated by category across a migration zone. Virtual data centers are allocated inside migration zones, and oversubscription is calculated after the virtual data centers are created.

For example: consider we have three nodes in a migration zone, each with 20 cores and 128 GB of memory. Node 1 and Node 2 are standard nodes (category = General) and Node 3 is a high-performance node (category = High_Perf). In total, we have 40 cores and 256 GB of General and 20 cores and 128 GB of High_Perf available. Virtual data centers are allocated from migration zones and storage pools, and application instances are provisioned inside virtual data centers.

Consider that we now create three virtual data centers from the three nodes in the migration zone, as shown in the table below. VDC1 is for development and test, VDC2 is for production and VDC3 is for DevOps. VDC1 is created with 16 cores and 160 GB of RAM of the General category, adequate for dev/test needs. VDC2 is created with 24 cores and 96 GB of General, and 24 cores and 160 GB of High_Perf. Some of the production application instances need nodes from the High_Perf category; others see the General nodes as adequate. Finally, VDC3 is created with 20 cores and 128 GB of General and 4 cores and 12 GB of High_Perf. In total, we have allocated 60 cores and 384 GB of General and 28 cores and 172 GB of High_Perf. The subscription factors for both cores and memory for the General category are 1.5. The subscription factors for the High_Perf category are 28/20 or 1.4 for cores, and 172/128 or 1.34 for memory. These subscription factors change if we allocate a new virtual data center, or if one or more of the existing virtual data centers change the number of cores or memory allocated to them. We use the term “oversubscribed” to refer to a situation when the subscription factor exceeds 1 (or the subscription percentage exceeds 100%).

It is useful to re-emphasize two points about the definition of subscription factors at allocation time. First, they do not apply to individual nodes; they apply to all nodes of a migration zone that are in a specific category. Second, the subscription factor is based on virtual datacenter allocations, not on application instances we have provisioned. In the above example, the High_Perf and General categories become oversubscribed as soon as the virtual data centers are allocated. Even if we have not yet started a single application in these virtual data centers, these categories can become oversubscribed, as the above example demonstrates.

Table 1. Virtual data center example

	GENERAL Category	HIGH_PERF Category
VDC1 (Dev/Test)	16 cores, 160 GB	
VDC2 (Production)	24 cores, 96 GB	24 cores, 160 GB
VDC3 (DevOps)	20 cores, 128 GB	4 cores, 12 GB
TOTAL in virtual data centers	60 cores, 384 GB	28 cores, 172 GB
TOTAL AVAILABLE	40 cores, 256 GB	20 cores, 128 GB
Subscription Factor	1.5 CPU, 1.5 RAM	1.4 CPU, 1.34 RAM

Oversubscription at the node level

Next, we provision application instances inside virtual data centers. An application instance must specify the virtual datacenter it runs in. It also specifies the amount of resources it needs, and whether those resources must come from a certain category or from any category.

Continuing the example from “[Oversubscription at migration zone level](#)” on page 2, we allocate a total of six (6) application instances. App1 is allocated inside VDC1 and needs 12 cores and 128 GB of General. App2 is allocated inside VDC2 and needs 20 cores and 64 GB of General. App3 is also allocated inside VDC2 and needs 20 cores and 128 GB of High_Perf. App4 is allocated inside VDC3 and needs 10 cores and 64 GB of any category. Since there are not enough resources of the High_Perf category, App4 is allocated in the General category. App5 in VDC3 needs five (5) cores and 32 GB of General. Finally, App6 in VDC3 needs four (4) cores and 12 GB of High_Perf. We still have room for application instances in VDC1 as long as they consume less than four (4) cores and 32 GB of General category or can run in any category. (These resources must be available in a single node.) Similarly, we still have room for application instances in Virtual Data Center2 as long as they consume less than four (4) cores and 32 GB of either category, and in VDC3 as long as they consume less than five (5) cores and 32 GB of General.

Refer to the following table for the example allocation of application instances to virtual data centers and what remains available in each virtual data center.

Table 2. Allocation of application instances to VDCs and what remains available in each VDC

	Compute Categories			
Migration Zone Allocation	40	256	20	128
VDC Allocations	General		High Perf	
	Cores	RAM	Cores	RAM
VDC1	16	160	0	0
App1	12	128		
Free	4	32		
VDC2	24	96	24	160
App2	20	64		
App3			20	128
Free	4	32	4	32
VDC3	20	128	4	12
App4	10	64		
App5	5	32		
App6			4	12
Free	5	32	0	0
Subscription Factor	1.5	1.5	1.4	1.34375

To support even more application instances, infrastructure admins may allow the size of existing virtual data centers to become larger. For example, if an application instance that consumes more than four (4) cores and 32 GB of General needs to be started in Virtual Data Center1, this cannot be allowed until Virtual Data Center1 is made larger. Infrastructure admins may also allow new virtual data centers to be created; for example, they may allow a new Virtual Data Center4.

The infrastructure admin can look at the actual utilization of the running application instances to decide whether to allow these new allocations. Even though the subscription factor for the General category in this example is 1.5, if it turns out that the actual utilization of the General application instances (that is, application instances running on General nodes) is quite low, the admin can allow more virtual data centers to be created using General compute and General GBs and let the subscription factor go even higher. On the other hand, if the actual utilization of General application instances is very high, further virtual data centers needing General compute and General GBs may not be permitted. The infrastructure admin must make this decision carefully. Once the infrastructure admin allows a virtual data center of a certain size (cores, GBs) to be created, the virtual data center administrator for that virtual data center expects to be able to fully use those cores and GBs in running applications.

CAUTION:

If an infrastructure admin allows utilizations to run high, there is a danger that a virtual data center admin cannot start an application in their virtual data center, even with enough resources allocated. This situation must be avoided.

Since application instances must run on nodes, assume App1 and App4 run on Node 1, App2 and App5 run on Node 2, and App3 and App6 run on Node 3 as shown in the following table. This table shows the node-level oversubscription after the application instances have been provisioned. SF refers to subscription factor.

Table 3. Example assignment of application instances to nodes

	Apps	Cores	GB
Node 1	App1	20	128
	App4	10	64
	Used	30	192
	Avail	20	128
	SF	1.5	1.5
Node 2	App2	20	64
	App5	5	32
	Used	25	96
	Avail	20	128
	SF	1.25	0.75
Node 3	App3	20	128
	App6	4	12
	Used	24	140
	Avail	20	128
	SF	1.2	1.09375

The ThinkAgile CP user interface shows the application instances assigned to each node, as well as the core and memory requirements of each application instance. The subscription factor per node is not shown in the UI, but can be calculated by summing up the needs of each application instance on that node.

The ThinkAgile CP user interface also shows the load on each node. The load is different than the subscription factor.; the load is a function of the actual usage of the running application instances and can be lower than the subscription factor. It also varies with time. The load also includes the resources taken up by the hypervisor. A new application instance can be started on a node as long as the load is low enough and can accommodate the needs of the new instance, even if the subscription factor is high.

In our example, Node 1 has a compute subscription factor of 1.5 and a memory subscription factor of 1.5. The compute subscription factor is acceptable in our example. However, a memory subscription factor of 1.5 is higher than we generally recommend.

Oversubscription considerations

Oversubscription must be done with care. The conservative way to ensure that your application performs as intended is to not oversubscribe at all. Many administrators follow this approach. However, others consider this approach to be too conservative. Some administrators are comfortable with over committing CPU and storage resources, but not with overcommitting RAM. This is because the real-world consequences of over committing RAM are much worse and more unpredictable than overcommitting CPU.

How much oversubscription should be used depends on workload. An approach that works for one customer and one workload may perform poorly for another. Furthermore, the age of the processor is also a factor. Newer processors can support higher levels of CPU oversubscription than older processors. Therefore, it is not possible to provide general guidelines that work safely for all customers, all workloads, and all processor types.

Note: If you are going to oversubscribe resources, we recommend that you oversubscribe storage and CPU resources before you oversubscribe memory resources.

CPU oversubscription

For CPU oversubscription, a subscription factor of three (3), or subscription percent of 300%, or below generally works for a range of workloads.

Whether you can go above 3 is workload dependent. For example, simple HTTP server workloads may be able to run with subscription factors as high as 10 (1,000%). We recommend you test your workload for a minimum of a week, under production level conditions, to determine the best subscription factor for your workload.

Memory oversubscription

For memory oversubscription, we recommend that you start with a subscription factor of 1 (100%).

If you want better memory utilization, we have found from limited testing that a subscription factor of 1.2 (120%) works adequately for some workloads. We do not recommend using anything higher than 1.2 without careful testing.

As with CPU oversubscription, we recommend that you test your workload for a minimum of a week, under real-world conditions, to determine the right subscription factor level and whether a subscription factor higher than 1.2 is safe.

Monitoring CPU and memory

In the ThinkAgile CP user interface, the dashboard shows you CPU and memory oversubscription levels for each migration zone.

It also shows you CPU and memory loads for each migration zone over a period of your choosing (last hour, last 12 hours, last day, last 7 days, last month, 3 months, 1 year, etc.). If the load remains below 50% over a full 7 days of production level running at a given oversubscription level, that should be a safe oversubscription level for you.

We recommend that you watch CPU and memory node loads continually, not just during the recommended one-week test period. If they start to approach 100% at any time, ThinkAgile CP provides ways for you to move application instances to other nodes with lower loads.

Enabling and disabling memory oversubscription

Memory oversubscription (also known as memory ballooning) is a service daemon that handles dynamic memory allocation across virtual machines (VMs) running on a compute node.

This service allows VMs to release memory when it is no longer needed and, thus, ensures that at any given time no VMs have more memory allocated than is actually consumes. The benefit is more efficient use of available memory and improvements in workload performance. For more information, see [Chapter 1 “Oversubscription in ThinkAgile CP” on page 1](#).

Service files

The service daemon consists of several files, which are deployed during the installation and upgrade of the `tacp-compute-controller` service.

The service daemon consists of the following files:

- `tacp-memory-oversubscription.service` - Systemd unit file that contains the definition of the memory oversubscription service. Gets deployed to `/usr/lib/systemd/system/`
- `tacp-memory-scheduler` - The main script that implements the memory over-subscription functionality. Gets deployed to `/usr/bin/`
- `Domains.py`, `ErrorCodes.py`, `execute.py`, `VM.py` - A set of helper scripts used by the `tacp-memory-scheduler` script. Gets deployed to `/usr/share/tacp/memory-oversubscription/`

Enable and disable memory oversubscription

By default, the memory oversubscription daemon is NOT enabled.

Enable or disable memory oversubscription on the compute block

This section covers the procedure required to enable or disable compute block memory oversubscription.

Note: If your hardware stack has multiple compute controllers, you must enable the memory oversubscription service on each compute node separately.

Step 1. To start the service:

- Run: `sudo systemctl start tacp-memory-oversubscription.service`
- To ensure that the service is up and running, run: `sudo systemctl status tacp-memory-oversubscription.service`
 - `tacp-memory-oversubscription.service` - ThinkAgile CP Dynamic Memory Oversubscription Service
Loaded: loaded (`/usr/lib/systemd/system/tacp-memory-oversubscription.service`; disabled; vendor preset: disabled)
Active: **active (running)** since Tue 2918-12-04 EST; 1s ago
Main PID: 47861 (`tacp-mem`)
Tasks: 1
CGroup: `/system.slice/tacp-memory-oversubscription.service`
└─47861 `/usr/bin/python /usr/bin/tacp-memory-scheduler -i 1 -r 300 -m 25 -f 10`
- Enable the memory oversubscription service so that the service starts on reboot: `sudo systemctl enable tacp-memory-oversubscription.service`

Step 2. To stop the service:

- Run: `sudo systemctl stop tacp-memory-oversubscription.service`
- To ensure that the service is not running, run: `sudo systemctl status tacp-memory-oversubscription.service`
 - `tacp-memory-oversubscription.service` - ThinkAgile CP Dynamic Memory Oversubscription Service

```
Loaded: loaded (/usr/lib/systemd/system/tacp-memory-oversubscription.service; disabled;
        vendor preset: disabled)
Active: inactive (dead)
```

- c. Disable the memory oversubscription service: `sudo systemctl disable tacp-memory-oversubscription.service`

Enable and disable memory oversubscription within the VMs

You can enable or disable memory oversubscription with the VMs.

To enable and disable memory oversubscription within the VMs, perform these steps.

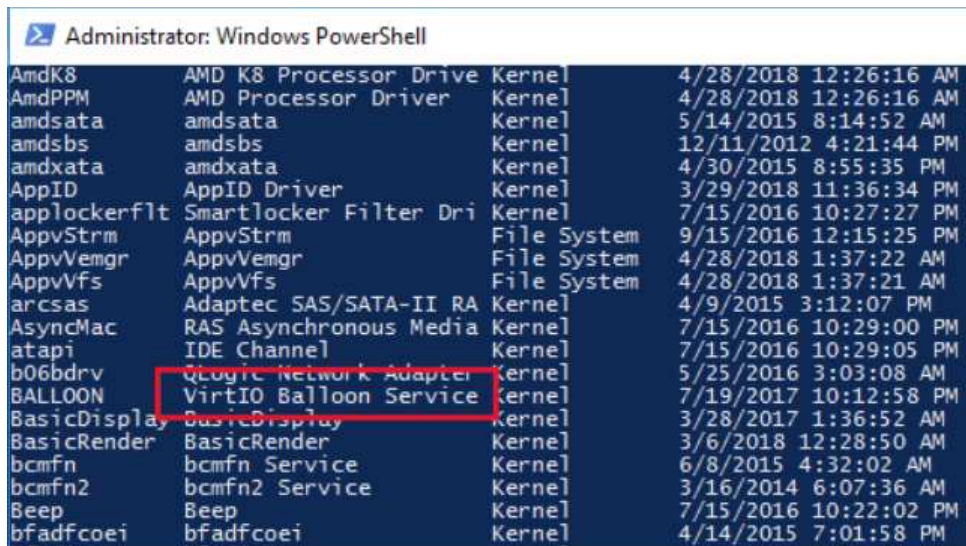
- **Linux:** No action is needed for Linux VMs. Memory oversubscription is supported automatically once it is enabled on the compute block.
- **Windows:** Perform the following procedure.

Note: Templates provided by the Lenovo Cloud Marketplace already include the necessary drivers. You must install them manually only if you are preparing your own custom template.

Note: The Windows Memory Ballooning service is NOT installed by default in templates provided by Lenovo Cloud Marketplace. You must install it manually for both ThinkAgile CP and your custom templates.

Step 1. Ensure that the correct drivers are installed within the guest operating system (OS) as follows:

- a. Open PowerShell with administrator privileges.
- b. Run: `driverquery`
- c. Ensure that VirtIO drivers are installed. If the drivers are not present, install them manually (see step 3).



```
Administrator: Windows PowerShell
AmdK8 AMD K8 Processor Drive Kernel 4/28/2018 12:26:16 AM
AmdPPM AMD Processor Driver Kernel 4/28/2018 12:26:16 AM
amdsata amsata Kernel 5/14/2015 8:14:52 AM
amdsbs amsbs Kernel 12/11/2012 4:21:44 PM
amdxata amdxata Kernel 4/30/2015 8:55:35 PM
AppID AppID Driver Kernel 3/29/2018 11:36:34 PM
applockerflt Smartlocker Filter Dri Kernel 7/15/2016 10:27:27 PM
AppvStrm AppvStrm File System 9/15/2016 12:15:25 PM
AppvVemgr AppvVemgr File System 4/28/2018 1:37:22 AM
AppvVfs AppvVfs File System 4/28/2018 1:37:21 AM
arcsas Adaptec SAS/SATA-II RA Kernel 4/9/2015 3:12:07 PM
AsyncMac RAS Asynchronous Media Kernel 7/15/2016 10:29:00 PM
atapi IDE Channel Kernel 7/15/2016 10:29:05 PM
b06bdrv QLogic Network Adapter Kernel 5/25/2016 3:03:08 AM
BALLOON VirtIO Balloon Service Kernel 7/19/2017 10:12:58 PM
BasicDisplay BasicDisplay Kernel 3/28/2017 1:36:52 AM
BasicRender BasicRender Kernel 3/6/2018 12:28:50 AM
bcmfn bcmfn Service Kernel 6/8/2015 4:32:02 AM
bcmfn2 bcmfn2 Service Kernel 3/16/2014 6:07:36 AM
Beep Beep Kernel 7/15/2016 10:22:02 PM
bfadfcoei bfadfcoei Kernel 4/14/2015 7:01:58 PM
```

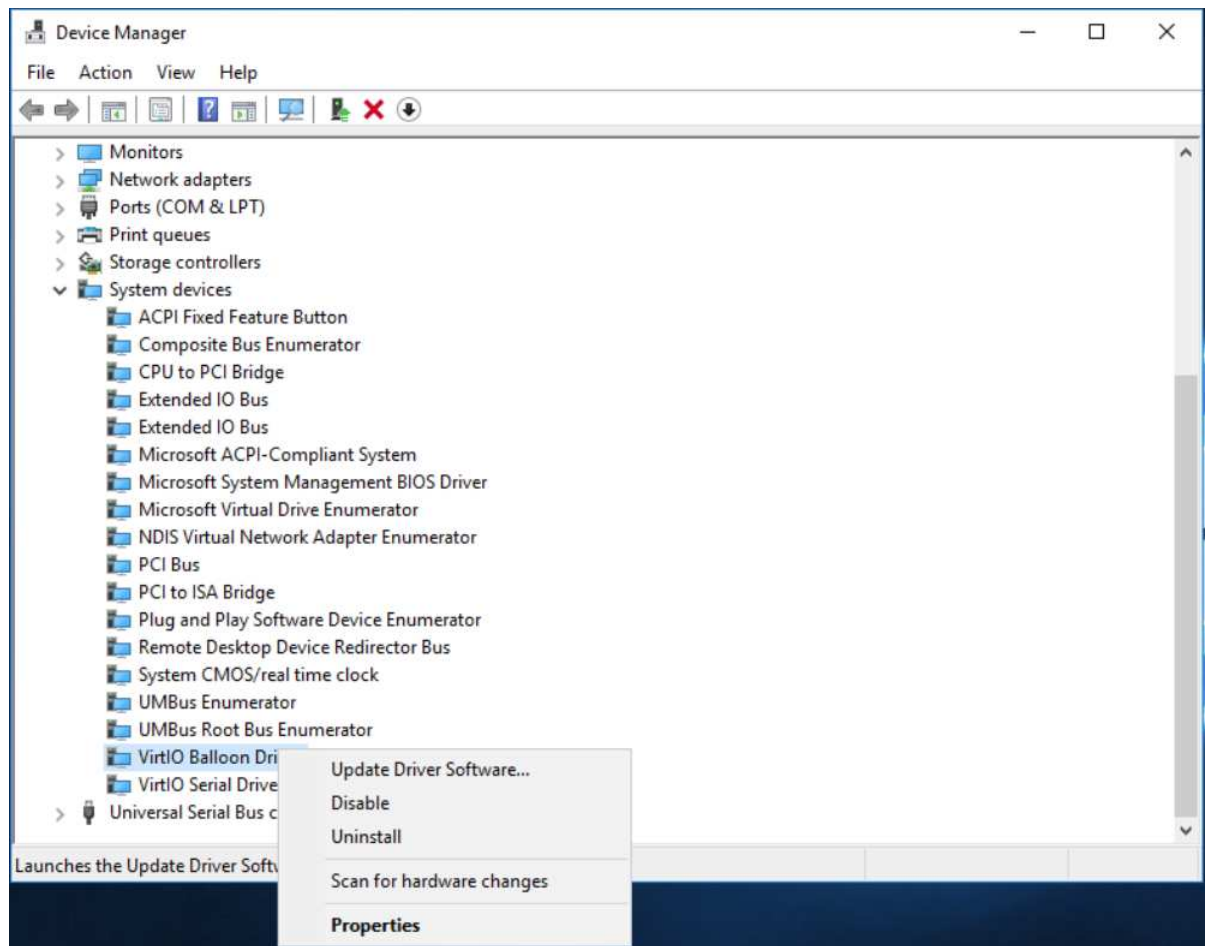
Step 2. Ensure that the memory ballooning service is installed and running as follows:

- a. Open PowerShell with administrator privileges.
- b. Run: `Get-Service`
- c. Ensure that the memory ballooning service is listed and is in a running state. If the service is not present, install it manually (see step 4). If the service is not running, start it.

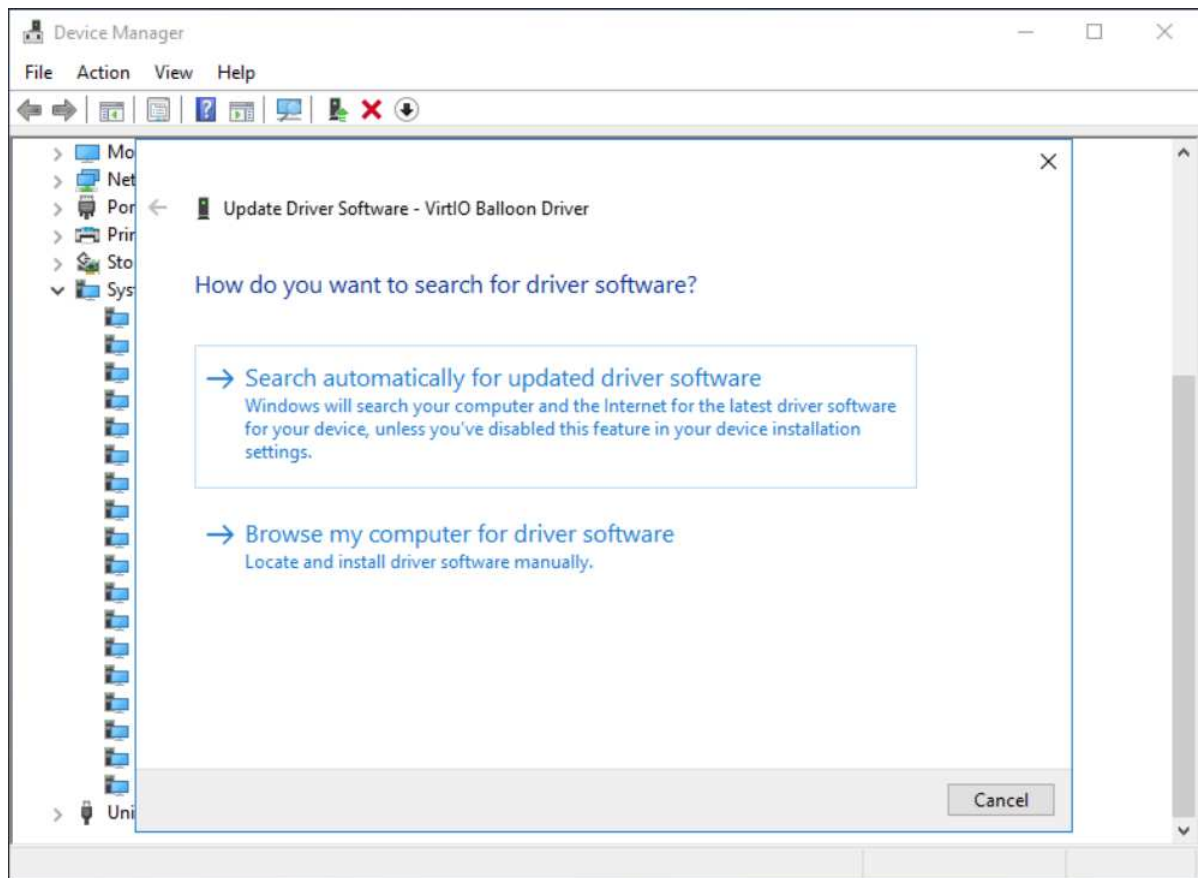
```
Administrator: Windows PowerShell
PS D:\virtio-win-0.1.141\Balloon\2k16\amd64> Get-Service

Status      Name                DisplayName
-----
Stopped    AJRouter            AllJoyn Router Service
Stopped    ALG                 Application Layer Gateway Service
Stopped    AppIDSvc            Application Identity
Running    Appinfo             Application Information
Stopped    AppMgmt             Application Management
Running    AppReadiness        App Readiness
Stopped    AppVClient          Microsoft App-V Client
Stopped    AppXSvc             AppX Deployment Service (AppXSVC)
Stopped    AudioEndpointBu... Windows Audio Endpoint Builder
Stopped    Audiosrv            Windows Audio
Stopped    AxInstSV            ActiveX Installer (AxInstSV)
Running    BalloonService      Balloon Service
Running    BFE                 Base Filtering Engine
Stopped    BITS                Background Intelligent Transfer Ser...
Running    BrokerInfrastru... Background Tasks Infrastructure Ser...
Stopped    Browser             Computer Browser
```

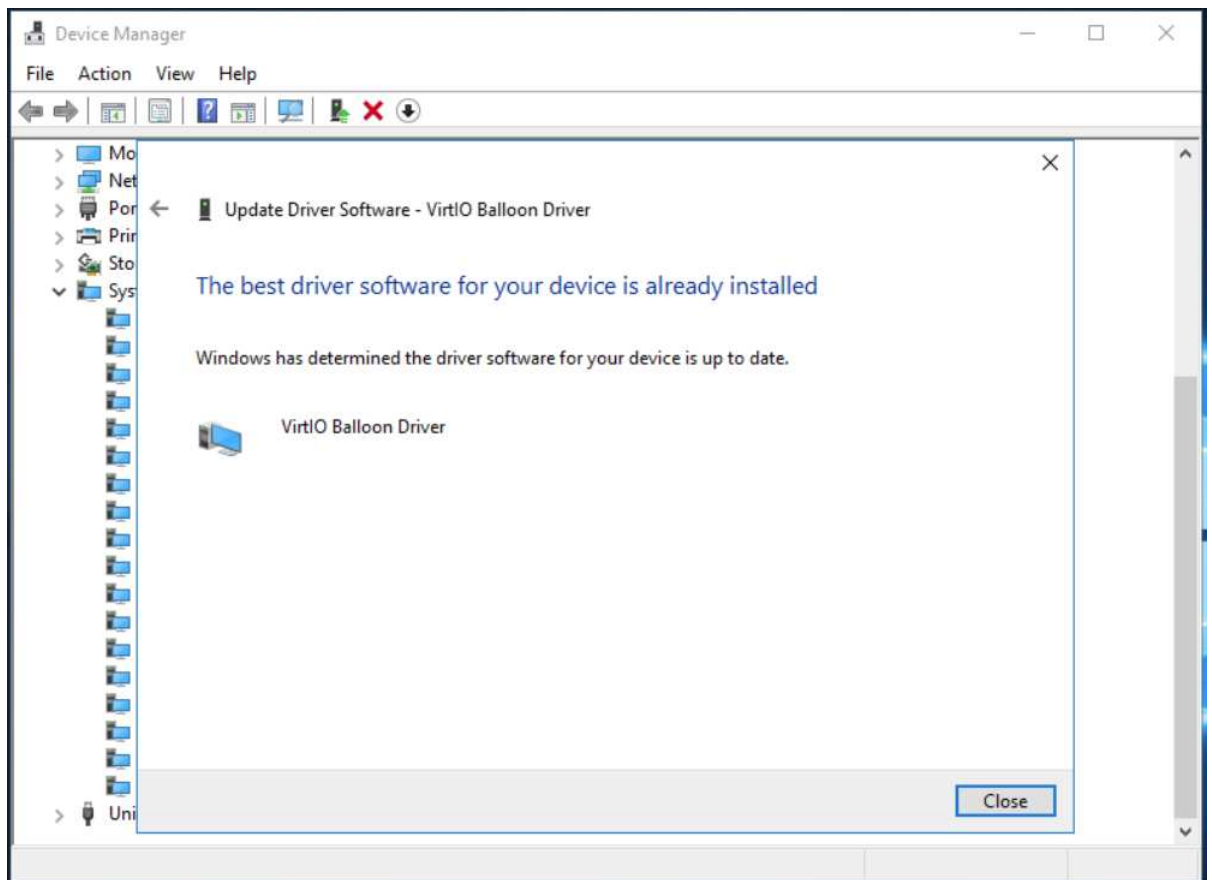
- Step 3. Install VirtIO drivers. Within the Windows guest OS, ensure that the ThinkAgile CP ISO is attached as follows:
- a. Open Device Manager.
 - b. Navigate to System Devices. (The option here might have a different name depending on your Windows OS version. For example, it is known as System Devices for Windows Server 2016.)
 - c. Right-click the VirtIO Balloon Driver.
 - d. Select **Update Driver Software**.



- e. Select **Browse my computer for driver software**.



- f. Provide the path to the CD drive and click **Next**.
- g. After the drivers are installed, you should receive a confirmation like the following.



Step 4. Install the Windows Memory Ballooning Service.

- a. Within the Windows guest OS, ensure that the ThinkAgile CP ISO is attached.
- b. Navigate to the folder <CDROM>:\virtio-win-0.1.141\Balloon\<WINDOWS>
- c. Copy the folder with the balloon service to a safe location on the C:\ drive. For example, you can use C:\Balloon\.

It is critical to copy the service into a reliable location before registering it. Otherwise, the service might become unavailable after the VM restart.

Note: Depending on the Windows version, the ISO might include both 32-bit (x86) and 64-bit (amd64) versions of the service. You should choose based on your VM processor architecture.

- d. Open the PowerShell window as the administrator, and navigate to the folder into which you copied the Balloon service.
- e. Execute `.\blnsvr.exe -i` to register the service within the OS. (See the image in the following step to see how the registration looks.)
- f. Follow the Windows instructions above to ensure that the memory ballooning service is now running.

```

Administrator: Windows PowerShell
PS D:\virtio-win-0.1.141\Balloon\2k16\amd64> .\blnsvr.exe -i
Service Installed
Service is starting...
Service RUNNING.
PS D:\virtio-win-0.1.141\Balloon\2k16\amd64>
  
```


Lenovo