jedox.

Jedox 7.0

Jedox GPU Accelerator
Fact Sheet

Jedox 7.0

Jedox GPU Accelerator

Fact Sheet

Dated: 28-Nov-2016

Table of Contents

1 Introduction

Jedox GPU Accelerator uses the computational power of NVIDIA GPUs to speed up calculations in the Jedox OLAP Server. This document provides information on hardware and system requirements, as well as how to install and use the Jedox GPU Accelerator.

2 System requirements

2.1 Hardware requirements

Jedox GPU Accelerator requires CUDA-capable NVIDIA Tesla™ GPUs with Compute Capability 2.0 or higher. If you are unsure about the compute capability of your NVIDIA device, please check the following website and/or contact your graphics card vendor:
https://developer.nvidia.com/cuda-gpus

The minimum recommended GPUs are NVIDIA Tesla™ C20xx/M20xx devices (code-named “Fermi”). More recent models of the NVIDIA Tesla™series (code-named “Kepler”, NVIDIA Tesla™ K20, K40, and K80) are faster and future-proof. In particular, the Tesla™ K40 offers 12 GB of graphics memory, and the Tesla™ K80 offers 24 GB of graphics memory. Current mainboards offer up to 8 PCIe slots and can thus house up to 8 GPUs. If you use more than one GPU, all cards must be of the same type.

As of Jedox 7.0, non-Tesla™ GPUs are supported for testing purposes (viable support); thus, any CUDA-capable NVIDIA GPU can be used. Note that non-Tesla™ GPUs have a driver-side CUDA kernel execution timeout (normally 5 seconds) that can be reached in extremely computation-intensive operations, such as aggregations or dimension filters. This timeout might be configurable, but Jedox does not provide any support for these types of configuration changes.

2.1.1 Purchasing GPU hardware

While it is possible to buy GPUs and other server components separately, Jedox strongly recommends that customers buy pre-configured GPU hardware from a specialized dealer to avoid incompatibilities or hardware instability. If you would like to start with one GPU but keep your upgrade options open, we recommend telling this to your hardware dealer, who can ensure you will have an appropriate mainboard (with multiple PCIe slots) and an adequate power supply. We recommend connecting the graphics card to undivided 16-bit PCI slots. We also recommend connecting all graphics cards on a single PCIe bus controller.

2.2 Operating system requirements

2.2.1 Windows

Jedox GPU Accelerator is only available for the 64-bit version of Jedox Suite and hence also requires a 64-bit version of Windows. **Windows Server 2012 R2** is recommended. Note that Windows Server 2012 (without R2) is not compatible with CUDA versions >= 7.5 as used in Jedox 6 and Jedox 7.

2.2.2 Linux

Jedox GPU Accelerator will run on the 64-bit versions of the following Linux distributions: CentOS, RHEL, Fedora, and on any Linux distribution based on kernel 2.6.32 or later. If you are using a Linux version not mentioned above, we would be grateful for your feedback regarding Jedox GPU Accelerator on that system.

3 Installation

Please install the NVIDIA driver (Windows) or CUDA Toolkit (Linux) **before** installing or updating the Jedox Suite.

3.1 Drivers

The following table lists the CUDA Toolkit versions used in latest Jedox releases and the minimum required driver versions:

CUDA Toolkit version	Jedox version	Minimum driver version (Windows)
8.0	7.0	369.30
7.5	6.0 SR2, 6.0 SR3	352.39

3.2 Driver installation

3.2.1 Windows

Use of the latest driver is recommended. To install NVIDIA graphics drivers, first download the driver, then start the driver installation process:
http://www.nvidia.com/Download/index.aspx

If the minimum required driver version as shown in the table above is not yet available for your specific GPU, use the driver that comes with the respective CUDA Toolkit installation (see Linux installation for download link; install in advanced mode and only perform the driver installation in the setup process).

After driver installation, make sure that TCC (Tesla Compute Cluster) mode is enabled for each Tesla™ GPU. To check, start the tool nvidia-smi (comes with the driver installation) via command prompt:

```
cd C:\Program Files\NVIDIA Corporation\NVSMI
nvidia-smi.exe
```

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 369.30                 Driver Version: 369.30                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf   Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080   WDDM  | 0000:02:00.0     Off |                  N/A |
| 27%   40C    P8     7W / 180W |    522MiB /  8192MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20c          TCC  | 0000:03:00.0     Off |                    0 |
| 30%   41C    P8    17W / 225W |      0MiB /  4726MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

For each GPU, the driver mode is shown to the right of its name. In the screenshot, the GTX™ card runs in WDDM driver mode and the Tesla™ card runs in TCC driver mode. If you see any Tesla™ card running in WDDM mode, change its mode with following command:

```
nvidia-smi -g 0 -dm 1
```

The -g option refers to the ID that appears to the left of the GPU name (change accordingly). Reboot the system.

3.2.2 Linux

See NVIDIA CUDA Toolkit Installation below.

3.2.3 NVIDIA CUDA Toolkit Installation (Linux only)

To install NVIDIA CUDA Toolkit 8.0, first download the executable for one of the supported Linux distributions:

https://developer.nvidia.com/cuda-toolkit

Make binary file executable with

```
chmod +x cuda_8.0.27_linux.run
```

Then stop the Xserver by changing to run level 3 with

```
init 3
```

Note: You have to install the Linux kernel headers and build tools of your distribution first to run setup.

If “nouveau” module is loaded on startup, this module has to be blacklisted. Then run executable as ‘root’ with

```
./cuda_8.0.27_linux.run
```

For more information about driver installation on Linux, please read the NVIDIA CUDA Installation Guide for Linux:

http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html

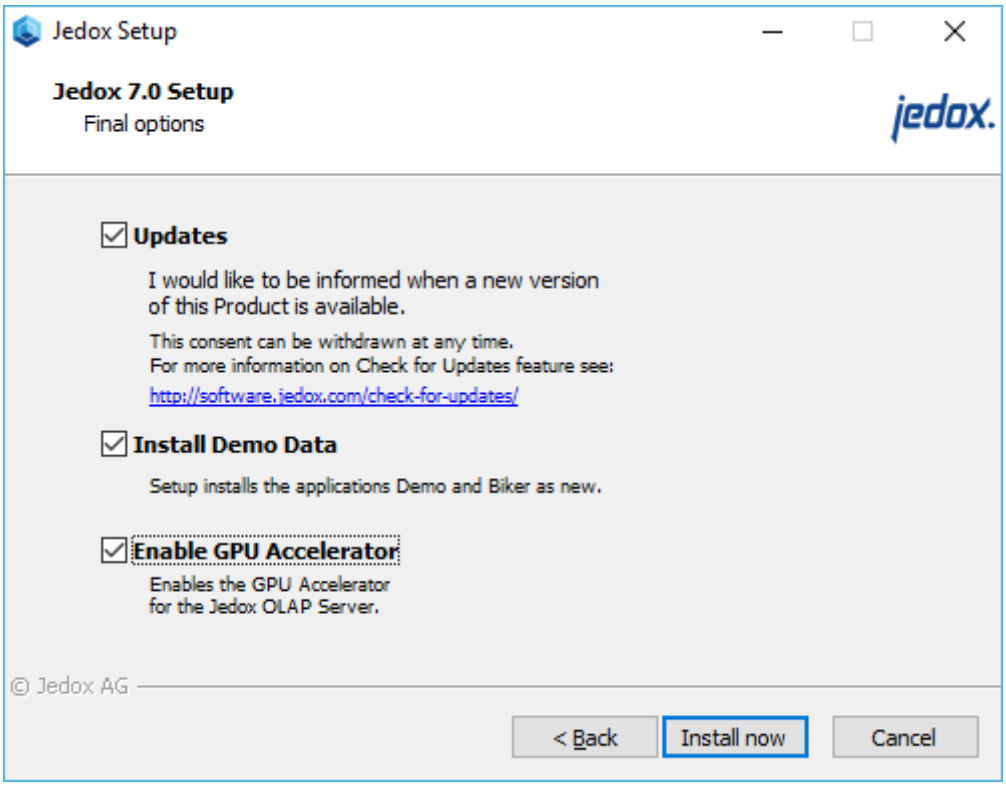
3.3 Jedox Suite installation

3.3.1 Windows

Use the Windows Jedox Setup file from the official Jedox website. During the setup, select the following setup type:

- **Advanced setup**

In setup dialog "Final options" select "Enable GPU Accelerator". This option is only selectable if a supported GPU card was found:



Activation will be done automatically during the installation process.

3.3.2 Linux

Download the Jedox Suite installation archive (Jedox_Suite_7.0.tar) and stop any running Jedox Suite instance by executing stopserver.sh. At this point we recommend backing up existing files. Extract all files in the tar archive to a temporary directory. Install it by executing

```
install.sh
```

Choose the GPU option at installation time. Start the server with startserver.sh.

3.4 Activation

When not specified otherwise, Jedox starts with all instances of the most powerful GPU found in the system according to the licensed number of devices (for questions on licensing, please contact Jedox). If only specific GPUs should be used, then those must be explicitly listed in the configuration file (palo.ini) in the following way:

```
...
device 83.0
device 84.0
...
```

The device IDs can be taken from the OLAP server log file after a successful startup:

```
...
... INFO: GPU: Device Count: 3 - CUDA driver / runtime version: 8.0 / 8.0
... INFO: GPU: Device 0 (Tesla K40c) has ID 83.0
... INFO: GPU: Device 1 (Tesla K40c) has ID 84.0
... INFO: GPU: Device 2 (Tesla K40c) has ID 85.0
...
```

3.5 Performance optimization

To further enhance the performance of the Jedox GPU Accelerator up to 100%, we recommend modifying the operating system's power options.

3.5.1 Windows

To prevent Windows from setting the system to power-saving mode, thereby slowing down the OLAP Server, follow these steps:

Go to Control Panel -> System and Security -> Power Options. Choose "High performance" (maybe hidden under "additional plans") or generate your own plan and enter the "Change advanced power settings" dialog. Set Processor power management -> Minimum processor state to "100%". In addition, you can set PCI Express -> Link State Power Management to "off".

3.5.2 Linux

For every CPU core with index <X>, run the following command to always set the CPU frequency to the maximum:

```
echo performance | tee /sys/devices/system/cpu/cpu<X>/cpufreq/scaling_governor

nvidia-smi -ac <memory,graphics> (e.g. 2000, 800)
```

This command specifies maximum <memory,graphics> clocks that define GPU speed while running applications on a GPU. Note that the command only works on Tesla devices from the Kepler+ family. See NVIDIA System Management Interface for details:

https://developer.nvidia.com/nvidia-system-management-interface

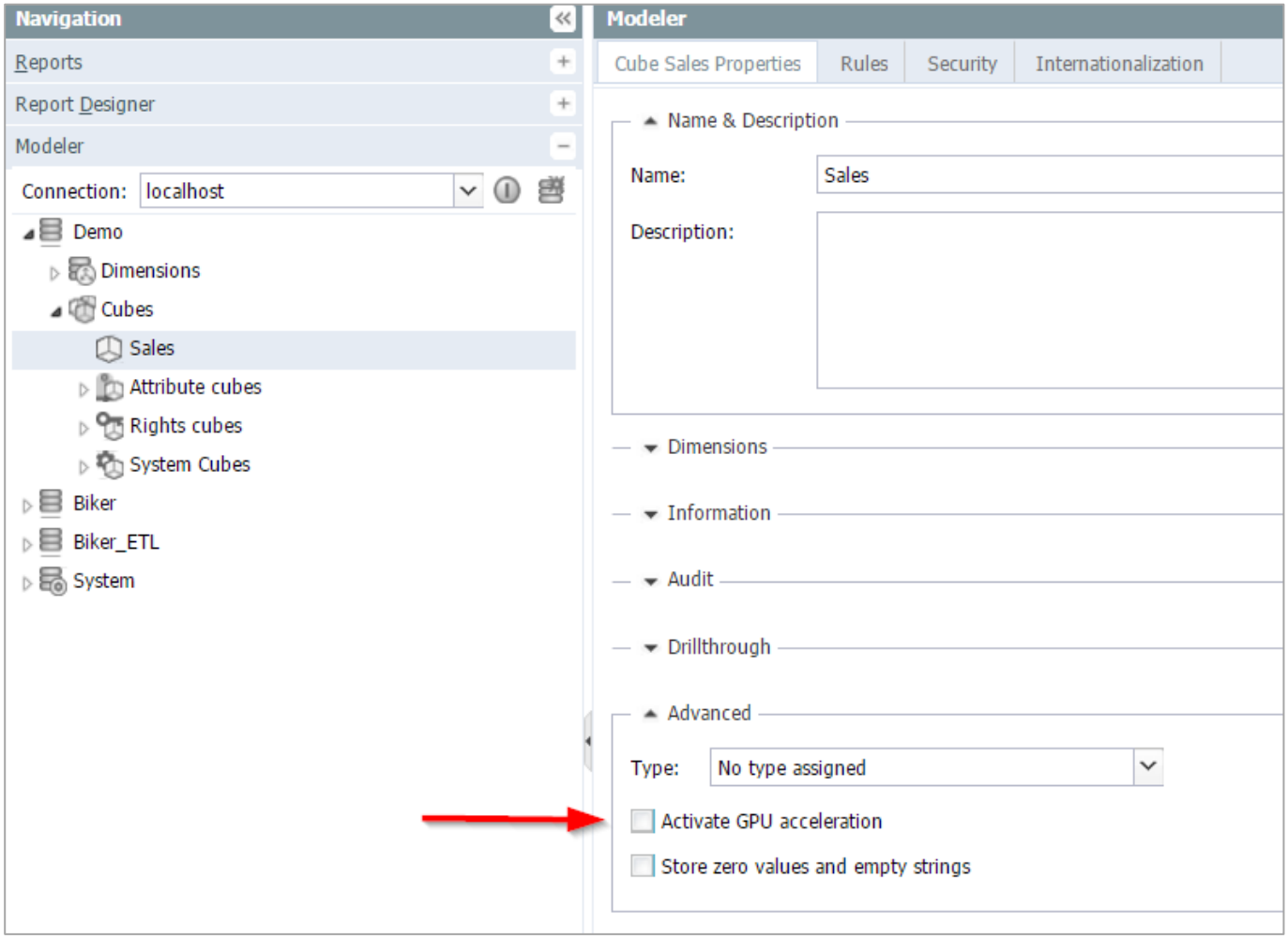
4 Using the Jedox GPU Accelerator

4.1 Cube conversion

Administrators can decide for each cube individually whether or not it should make use of GPU acceleration. While any cube (except system cubes) can be converted to a GPU cube, it makes sense especially for cubes with the following properties:

- **High numeric data volumes:** E.g. > 300k filled numeric cells.
- **Large consolidations**: Compute-intensive calculations, such as high-level consolidations or consolidations on large target areas.
- Cubes with **business rules** on base-cell level (B: rules), like arithmetical (+, -, *, /) or conditional rules (if/then/else), or rules across different cubes (PALO.DATA).
- Reports involving **dimension filters** (/dimension/dfilter).

To activate GPU acceleration for a specific cube, open the Modeler, select the cube, open the “Advanced” panel, and check the box for “Activate GPU acceleration”:



Note that speedups can only be expected when the majority of steps in the computation chain (e.g. multiplication step in rule, cell transformation step in PALO.DATA, aggregation) provide enough input cells to fully utilize the GPU hardware, i.e., thousands of cells or more.

4.1.1 Cube conversion to GPU memory

When not specified otherwise (see next section), each cube's numerical data storage will physically reside in GPU memory after cube conversion, which requires enough available memory on the GPU. If multiple GPUs are available, the data storage is distributed among the devices. Note that GPU memory is also required by the GPU engine during calculations; as a rule of thumb, make sure that converted cubes do not use more than half of the available GPU memory. Use nvidia-smi to view GPU memory consumption:

```
cd C:\Program Files\NVIDIA Corporation\NVSMI
nvidia-smi.exe -l
```

4.1.2 Cube conversion to host memory

As of Jedox 7.0, the cubes' numerical data storages can also reside in GPU format in conventional RAM, which allows for accelerating cubes that are larger than available GPU memory. The option can be switched on by adding palo.ini parameter

```
gpu-data-storage R
```

Note that a conversion of the cube to GPU format is still necessary and requires additional available RAM on the host system. As internally page-locked (pinned) memory is used, **system performance might degrade** whenever the system runs out of available memory.

4.2 Write-back

Writing back to GPU-accelerated cubes is fully supported.

4.3 Rules

Rules are GPU-accelerated under specific circumstances that depend on the query and the rules involved.

As of Jedox 7.0, most rule functions are supported on GPU. To find out whether all rules are supported for a specific cube, administrators can use the GPU Accelerator Advisor from Jedox Excel Add-in to list all rules, along with GPU support information. Find details here:

http://knowledgebase.jedox.com/knowledgebase/accelerator-advisor/

4.4 Dynamic engine selection

As of Jedox 6, each reading operation (e.g. aggregation, rule computation, dimension filter, etc.) is evaluated according to its impact on the CPU and GPU engine, and the best suited engine is chosen to do the actual computation. Dynamic engine selection can be switched off such that GPU does the computation whenever it supports the operation by using/adding "o" to the "engine-configuration" parameter (in palo.ini):

```
engine-configuration o
```