Peridigm

DLR-IB-FA-BS-2019-33

Peridigm Installation Guide

For Peridigm versions $\geq$ 1.4.1

Martin Rädel

DLR German Aerospace Center
Composite Structures and Adaptive Systems
Structural Mechanics
Braunschweig

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

DLR German Aerospace Center
Composite Structures and Adaptive Systems
Structural Mechanics
Dr. Tobias Wille
Lilienthalplatz 7
38108 Braunschweig
Germany
Tel: +49 531 295-3012
Fax: +49 531 295-2232
Web: http://www.dlr.de/fa/en

Martin Rädel
Tel: +49 531 295-2048
Fax: +49 531 295-2232
Mail: martin.raedel@dlr.de

Document Identification:

Report number	DLR-IB-FA-BS-2019-33
Title	Peridigm Installation Guide
Subject	Peridigm
Author(s)	Martin Rädel
With contributions by . .	Christian Willberg
Filename	Peridigm_Installation_Guide.tex
Last saved by	DLR\raed_ma
Last saved on	30th January 2019

Document History:

Version 0.0.1	Initial draft	04.02.2016
Version 0.0.2	Updated draft with extended MPI support	11.02.2016
Version 0.0.3	Resolved encoding issues, added FAQ	12.02.2016
Version 0.0.4	Added virtual machine chapter	14.04.2016
Version 0.0.5	Added FETranslator chapter	21.04.2016

Contents

List of Figures

List of Listings

1. About *Peridigm*

1.1. Description

Peridigm is an open-source computational peridynamics code developed at Sandia National Laboratories for massively-parallel multi-physics simulations. It has been applied primarily to problems in solid mechanics involving pervasive material failure. *Peridigm* is a C++ code utilizing foundational software components from Sandia's *Trilinos* project and is fully compatible with the Cubit mesh generator and *ParaView* visualization code.

Peridigm development began under the Physics & Engineering Models element of the US DOE's Advanced Simulation and Computing (ASC) program. The project was led by Michael Parks and managed by John Aidun. Subsequent funding has been provided by the US DOE through the ASC, ASCR, and LDRD programs.

1.2. Adresses

Official homepage:	https://peridigm.sandia.gov/
Source code repository:	https://github.com/peridigm/peridigm
Release snapshots:	Must be requested via Download Registration Form on https://peridigm.sandia.gov/content/download-registration-form
User guide [1]:	http://www.sandia.gov/ djlittl/docs/PeridigmV1.0.0.pdf

1.3. Hardware requirements

Currently, there are no hardware requirements available or known. Publications, reports and presentations continuously mention the massive parallelization possbile and presumably also necessary to run *Peridigm*.

Information on the performance of *Peridigm* on different system will be added as soon as it is available.

1.4. Features

1.4.1. Example 1

The simulation of impact and brittle fracture displayed Figure 1.1 was achieved using explicit transient dynamics, the linear peridynamic solid constitutive model, short-range force contact, and a critical stretch bond failure law.

Peridynamics provides a natural framework for capturing pervasive material failure and fracture.

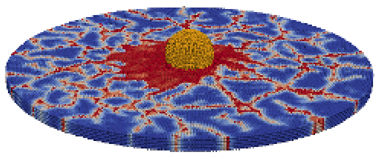


Figure 1.1.: **Impact failure**

1.4.2. Example 2

Peridigm is capable of performing explicit dynamic, implicit dynamic, and quasi-static time integration.

The tensile test simulation presented in Figure 1.2 was attained using an elastic correspondence constitutive model and quasi-static time integration. Pre- and post-processing were carried out using Sandia's *Cubit* mesh generator and *ParaView* visualization code.

Figure 1.2.: **Tensile test**

1.4.3. Example 3

The fragmentation of an expanding cylinder, shown in Figure 1.3, was simulated using the linear peridynamic solid constitutive model and critical-stretch bond failure rule. Initial velocities for each node in the discretization were specified via user-supplied analytic expressions.

Peridigm utilizes the RTCompiler function parser to process C-style expressions for the specification of input parameters, including initial and boundary conditions.

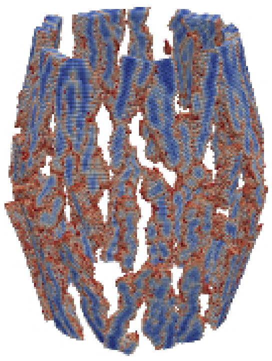


Figure 1.3.: **Fragmentation**

1.5. License

As of 04.02.2016 *Peridigm* is distributed under the three-term BSD license.

The current license terms can be obtained from:
`https://peridigm.sandia.gov/content/license`.

1.6. Peridigm working environment

Peridigm is a standalone tool for the handling of peridynamic models. Sandia National Labs suggests the use of *CUBIT* as the mesh generator and pre-processor and *ParaView* for post-processing. However, *CUBIT* is not publicly available free of charge except for U.S. government agencies. Any visualization package capable of reading and displaying ExodusII-format [2] data may be used to visualize the output of a Peridigm simulation. The creators of *Peridigm* use *ParaView*.

This installation guide focusses on the installation of the core program *Peridigm*. Several libraries are required as well as some basic tools *Peridigm* or one of its libraries is dependent on. All required tools for the use of *Peridigm* in a working environment are shown in Figure 1.4.

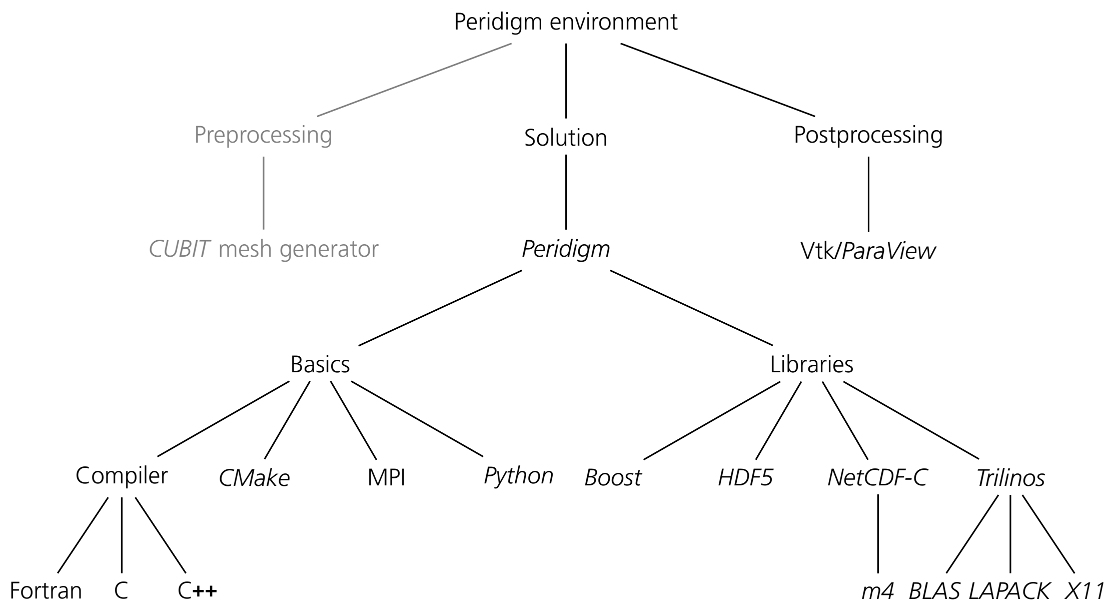


Figure 1.4.: **Peridigm working environment tree**

The installation guide features descriptions to the installation of all basic tools, libraries and *Peridigm* itself.

1.7. Dependencies

Below the dependencies between the individual tools and packages are shown.

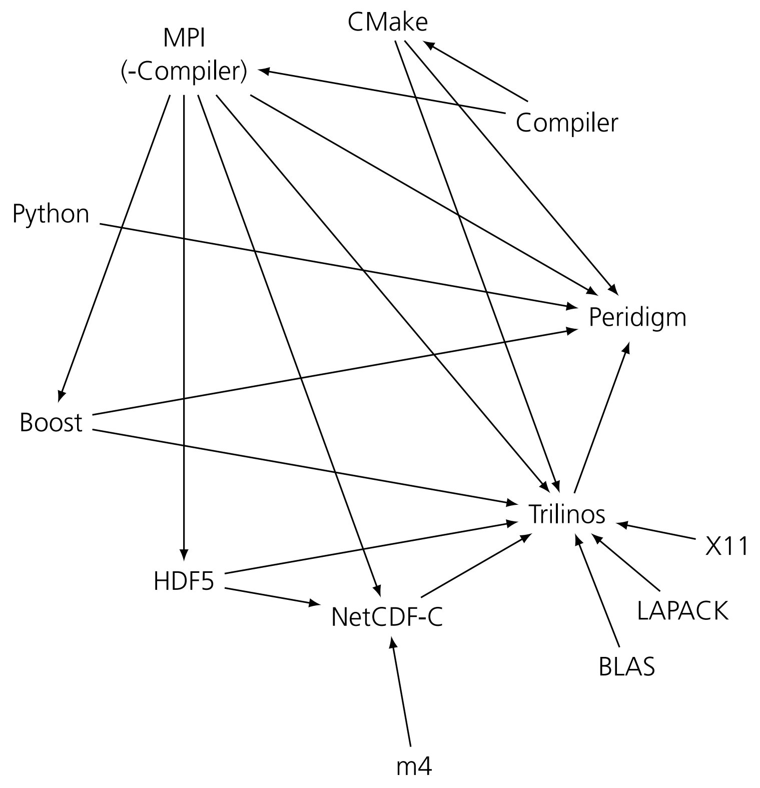


Figure 1.5.: **Peridigm package dependencies**

2. Linux in a virtual machine

If no native Linux-based operating system is available on a device it is possible to install a Linux distribution inside a virtual machine. The device used to install the virtual machine in is called the host in the description.

In case you have a native Linux operating system on your device you can skip the following steps and go to chapter 3.

2.1. Download Linux Distribution

For *Peridigm* we need a Linux distribution for the installation inside the virtual machine. Here, *openSUSE* is used. To download the latest version of *openSUSE* go to:

https://en.opensuse.org/Main_Page

In the top bar of the homepage go to the download section and choose *Latest stable release*. On the newly loaded page choose to download the DVD image as installation medium and perform the download onto your device. Beware, the image is quite big.

2.2. Install the virtual machine

2.2.1. Download and install virtualization software

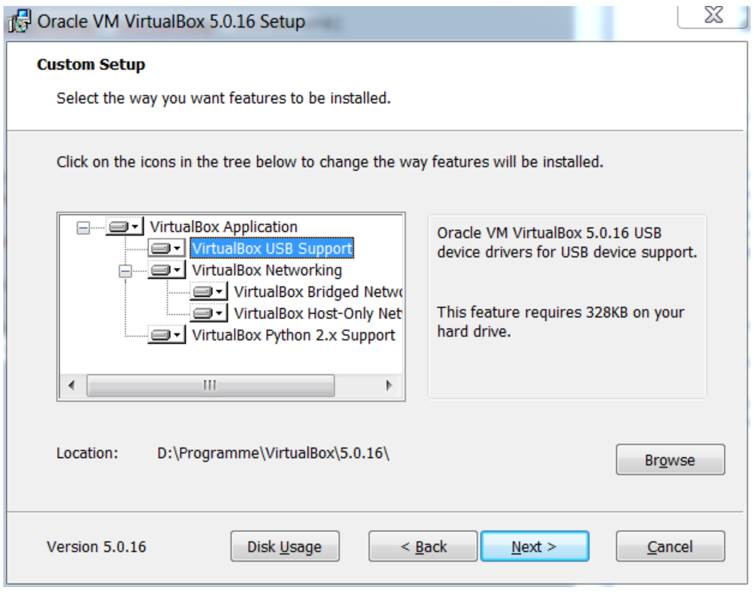
For the current case *VirtualBox* is chosen as the virtualization environment. *VirtualBox* is available for enterprise and home use and is freely available for both use-cases as Open Source Software under the terms of the GNU General Public License (GPL) version 2. The following chapter describes the setup of the then current version of *openSUSE* in a *VirtualBox* on your host operation system. *VirtualBox* is currently available for

- Windows
- OSX
- Linux
- Solaris

host operating systems. The current release of *VirtualBox* is available from:

https://www.virtualbox.org/

Go to the download section and choose the installer for your host operating system. The following description is valid for a Windows host and is tested for the 64bit variant of Windows 7. After the download execute the binary installer and follow the instructions of the setup wizard. In the custom setup step simply let all selected items enabled. Afterwards simply complete the wizard.



Oracle VM VirtualBox 5.0.16 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

```
VirtualBox Application
    VirtualBox USB Support
    VirtualBox Networking
        VirtualBox Bridged Netwo
        VirtualBox Host-Only Net
    VirtualBox Python 2.x Support
```

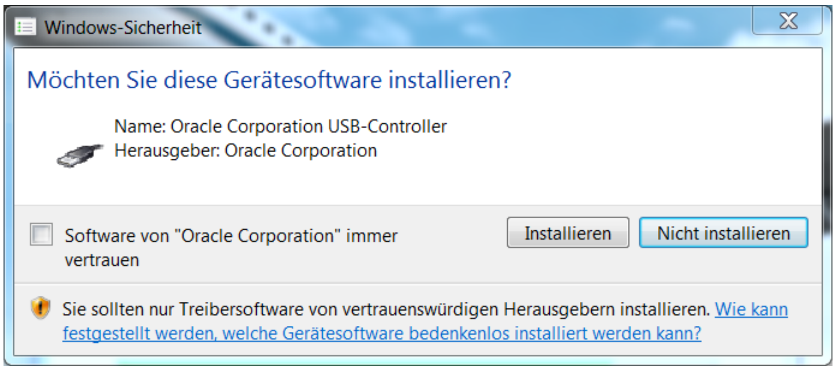
Oracle VM VirtualBox 5.0.16 USB device drivers for USB device support.

This feature requires 328KB on your hard drive.

Location: D:\Programme\VirtualBox\5.0.16\

Version 5.0.16

During the installation simply answer the following questions with *install* to allow access on the devices for the virtual machine.


Windows-Sicherheit

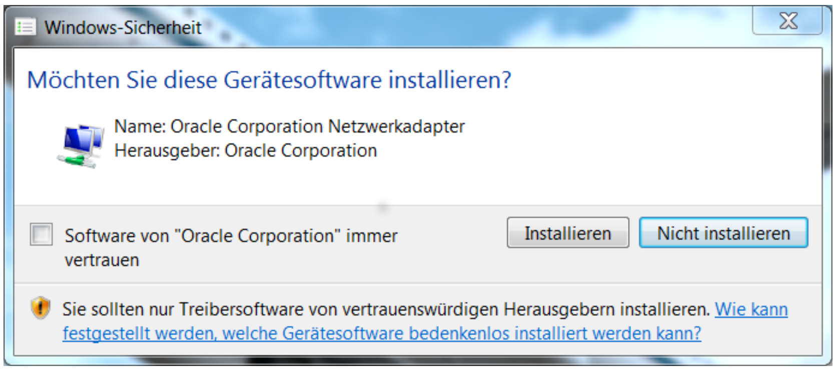
Möchten Sie diese Gerätesoftware installieren?

Name: Oracle Corporation USB-Controller
Herausgeber: Oracle Corporation

Software von "Oracle Corporation" immer vertrauen

Sie sollten nur Treibersoftware von vertrauenswürdigen Herausgebern installieren. Wie kann festgestellt werden, welche Gerätesoftware bedenkenlos installiert werden kann?

(a) **Install USB-Controller**


Windows-Sicherheit

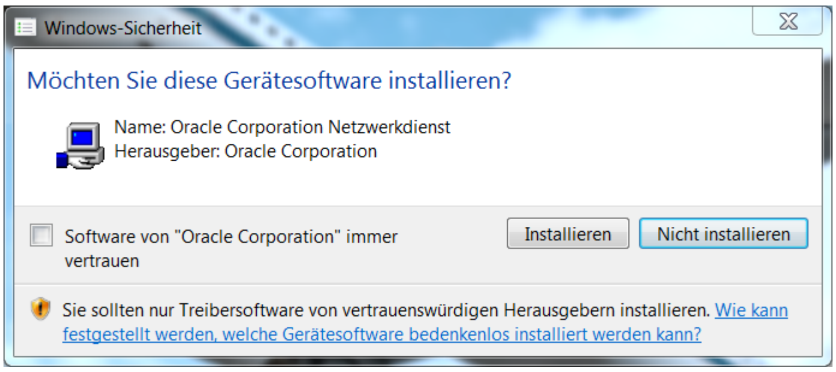
Möchten Sie diese Gerätesoftware installieren?

Name: Oracle Corporation Netzwerkadapter
Herausgeber: Oracle Corporation

Software von "Oracle Corporation" immer vertrauen

Sie sollten nur Treibersoftware von vertrauenswürdigen Herausgebern installieren. Wie kann festgestellt werden, welche Gerätesoftware bedenkenlos installiert werden kann?

(b) **Install network adapter**


Windows-Sicherheit

Möchten Sie diese Gerätesoftware installieren?

Name: Oracle Corporation Netzwerkdienst
Herausgeber: Oracle Corporation

Software von "Oracle Corporation" immer vertrauen

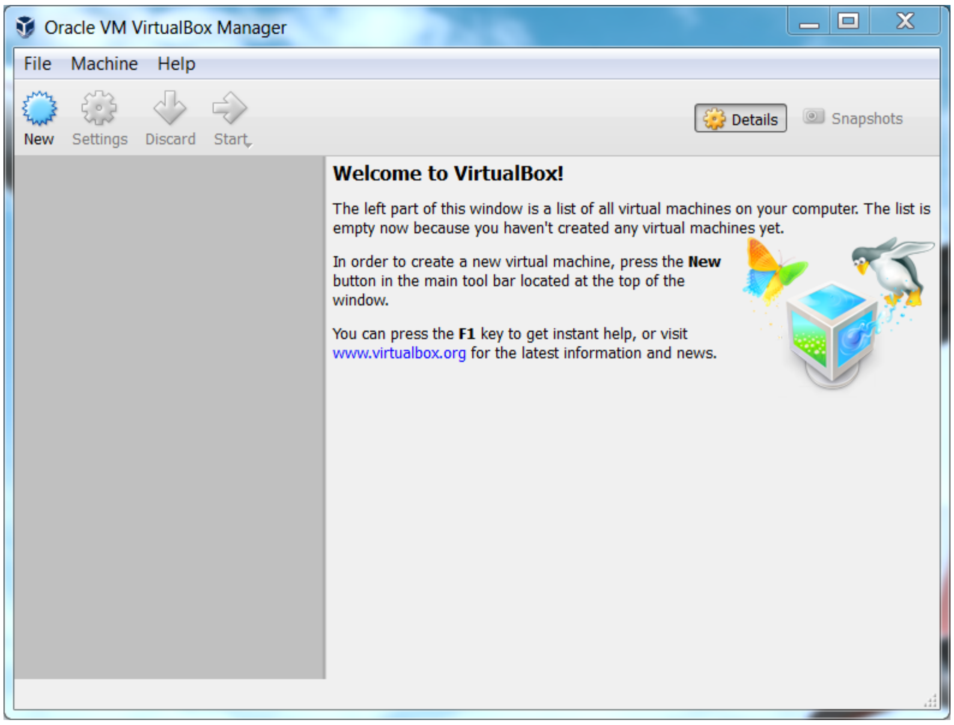
Sie sollten nur Treibersoftware von vertrauenswürdigen Herausgebern installieren. Wie kann festgestellt werden, welche Gerätesoftware bedenkenlos installiert werden kann?

(c) **Install network service**

Figure 2.1.: **Allow *VirtualBox* access to devices**

2.2.2. Create the virtual machine

After the installation is complete start the *VirtualBox* Manager.



Welcome to VirtualBox!

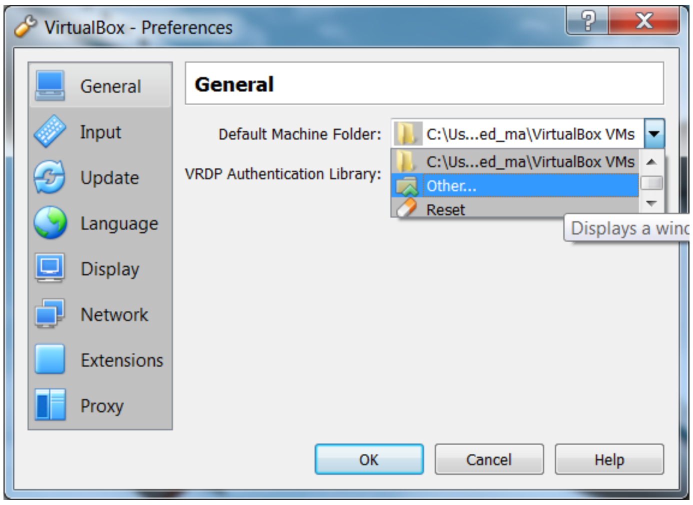
The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

Before we create a new virtual machine

- From the menu bar: Click *File*
- Click *Preferences*
- In the *General* tab select the *Default Machine Folder* to be a folder on a harddrive with sufficient amount of free memory to handle your virtual machine size, here 200GB



General

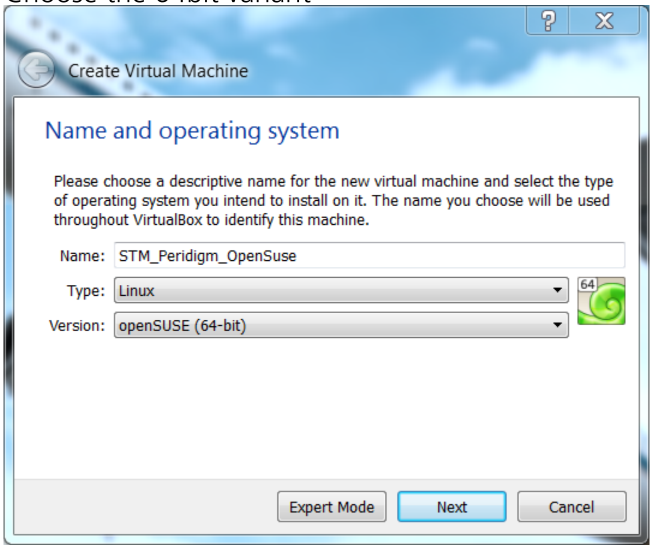
Default Machine Folder: C:\Us...ed_ma\VirtualBox VMs

VRDP Authentication Library:

- Close the Preferences windows with a click in *Ok*

Now we can create a new virtual machine. Click *New* in the *VirtualBox* Manager main window. A dialog appears that guides you through the setup of the virtual machine.

1. Select the virtual machine name, type and version
   - Choose a name that includes the version & the selection automatically jumps to your choice
   - Choose the 64bit variant



Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

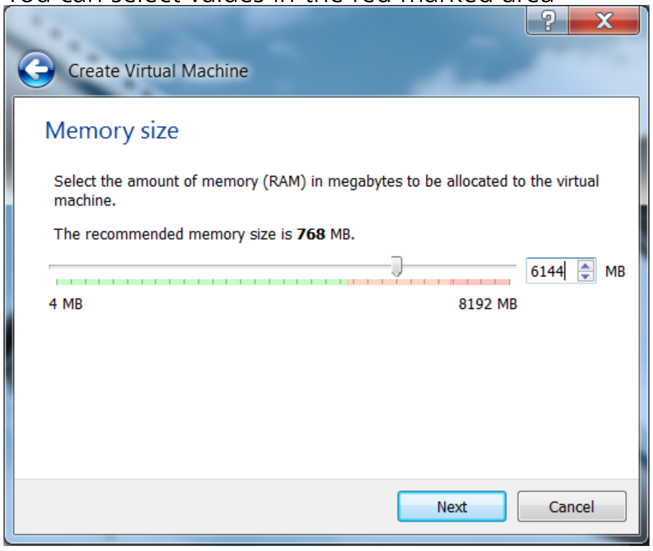
Name: STM_Peridigm_OpenSuse

Type: Linux

Version: openSUSE (64-bit)

   - A new folder with your chosen name is created in the directory you specified in the *General* tab of the *Preferences* toolbar menu
   - Click *Next*

2. Set the amount of RAM you grant your virtual machine
   - The more RAM *Peridigm* has available the better
   - But your host operating system also still needs some RAM left to work with
   - Choose an integer multiple of 1024 as amount of RAM
   - Leave the host operating system at least 2GB of RAM
   - You can select values in the red marked area

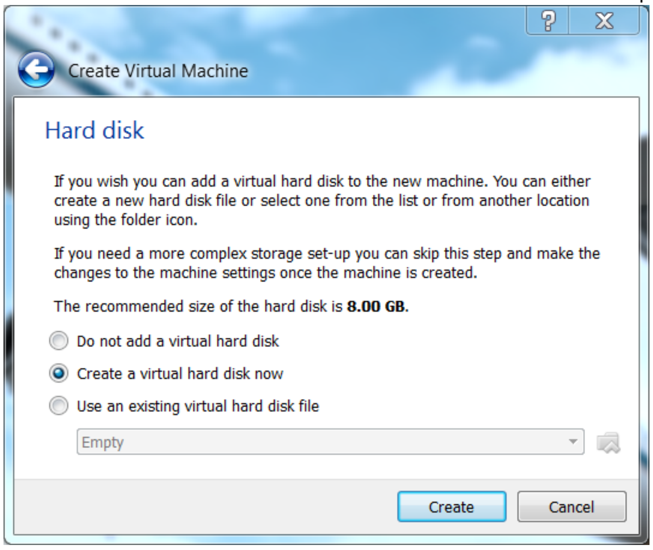


Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **768** MB.

   - Click *Next*
3. Create a virtual hard disk to install your distribution on
   - Here we select the *Create a virtual hard disk now* option



Hard disk

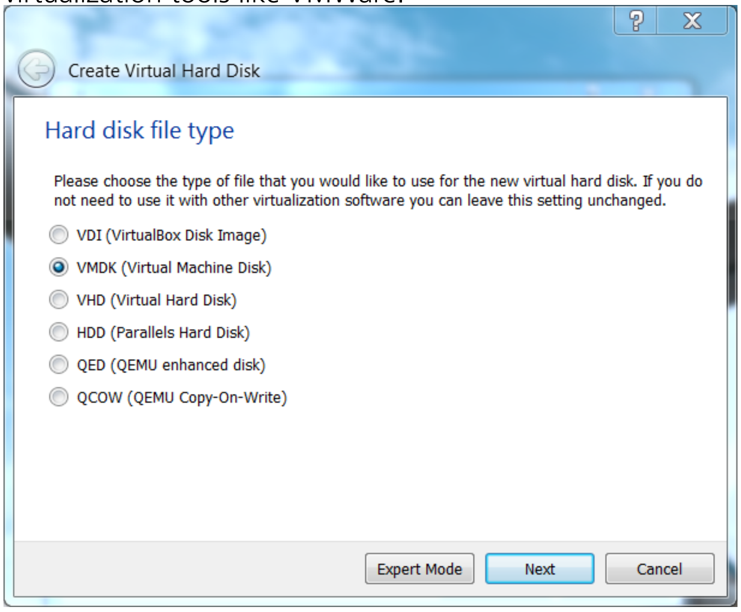
If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **8.00 GB**.

Do not add a virtual hard disk
[✓ Create a virtual hard disk now]
Use an existing virtual hard disk file

- Click *Create*
- Select *VMDK (Virtual Machine Disk)* to allow use of this virtual hard disk in other virtualization tools like VMWare.

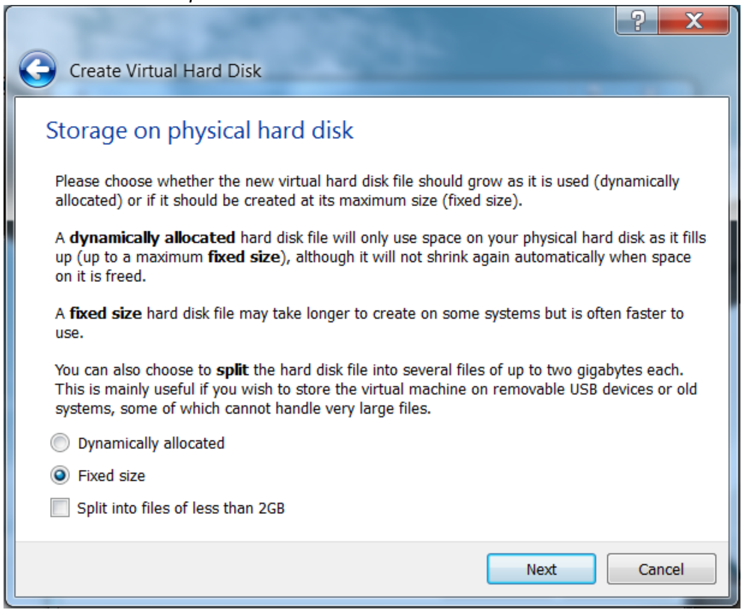


Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- ○ VDI (VirtualBox Disk Image)
- ● VMDK (Virtual Machine Disk)
- ○ VHD (Virtual Hard Disk)
- ○ HDD (Parallels Hard Disk)
- ○ QED (QEMU enhanced disk)
- ○ QCOW (QEMU Copy-On-Write)

- Click *Next*
- Select *Fixed size*
- Do not select *Split into files less than 2GB*



Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

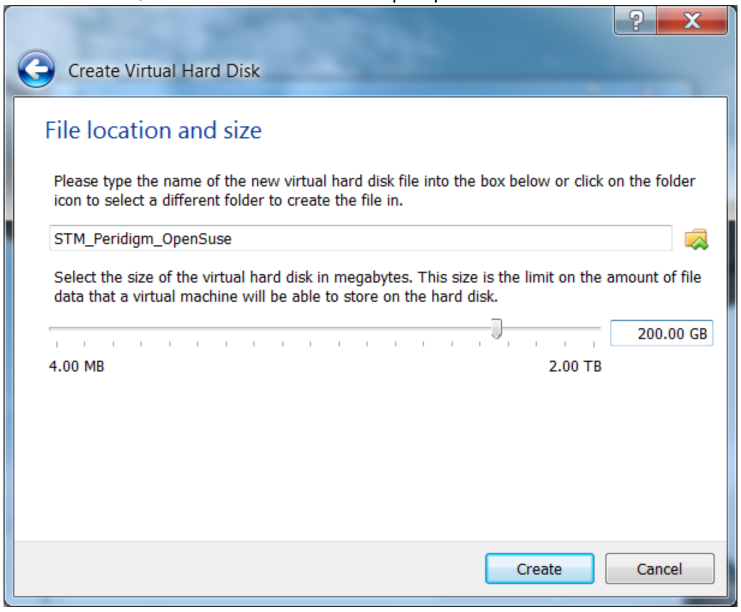
A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

You can also choose to **split** the hard disk file into several files of up to two gigabytes each. This is mainly useful if you wish to store the virtual machine on removable USB devices or old systems, some of which cannot handle very large files.

- ○ Dynamically allocated
- ● Fixed size
- ☐ Split into files of less than 2GB

- Click *Next*
- Setup virtual hard disk name and size
- By default the name is identical to the virtual machine name
- If you click the folder button you can change the location of the virtual hard disk file. By default it is saved in your virtual machine folder in the directory specified in the *General* tab of the *Preferences* toolbar menu
- Set the size of the virtual hard disk to the amount you want and have free, here 200GB. Despite being saved in binary format, *Peridigm* result files can be quite big and for explicit time integration a lot of them are created. Despite a Linux distribution does not need nearly as much hard disk space as a Windows installation, at least 120GB are proposed as virtual hard disk size.



Create Virtual Hard Disk

File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

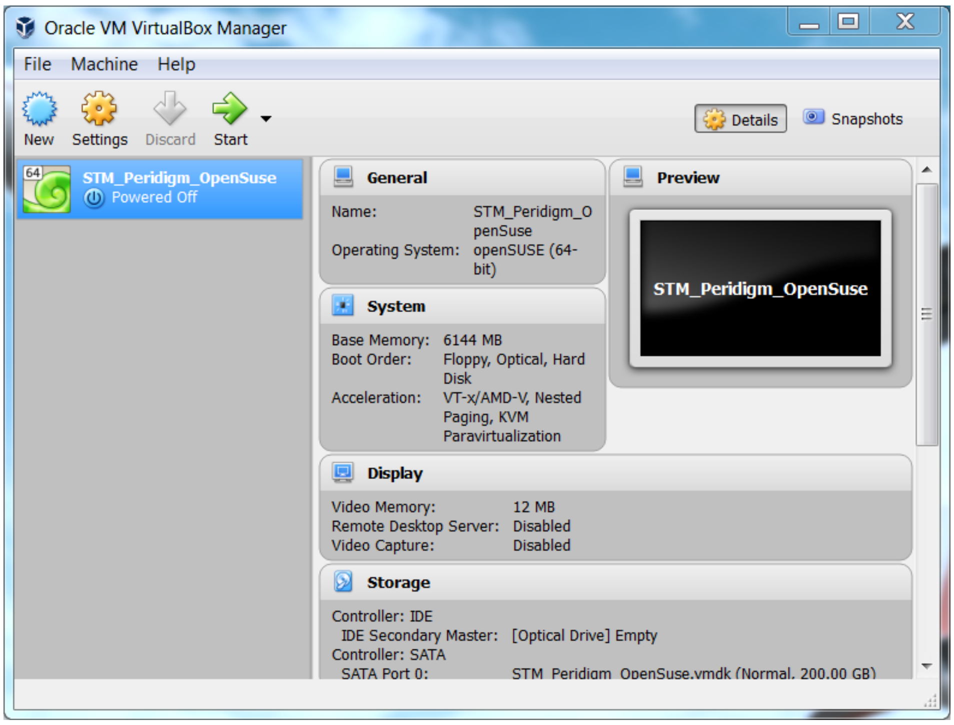
STM_Peridigm_OpenSuse

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

200.00 GB

4.00 MB 2.00 TB

- Click *Create*
- Wait for the creation to finish
- Afterwards you have a new virtual machine



General

Name: STM_Peridigm_OpenSuse
Operating System: openSUSE (64-bit)

System

Base Memory: 6144 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

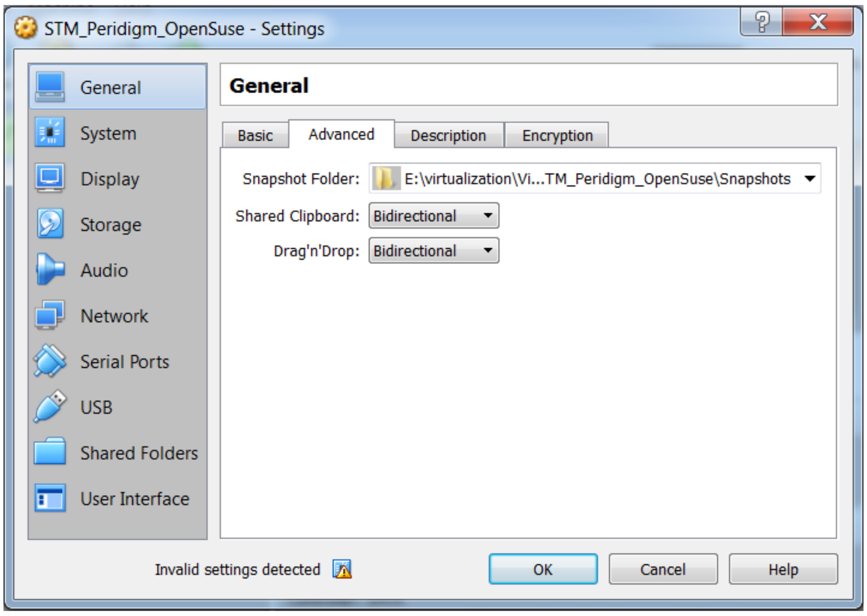
Display

Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: STM_Peridigm_OpenSuse.vmdk (Normal, 200.00 GB)

4. Before we install the Linux distribution inside the virtual machine, we have to configure some settings
   - Select the newly created virtual machine and click *Settings*
   - In the *General* tab
     - Select *Advanced*
     - Select: *Shared Clipboard* Bidirectional
       Drag'n'Drop Bidirectional

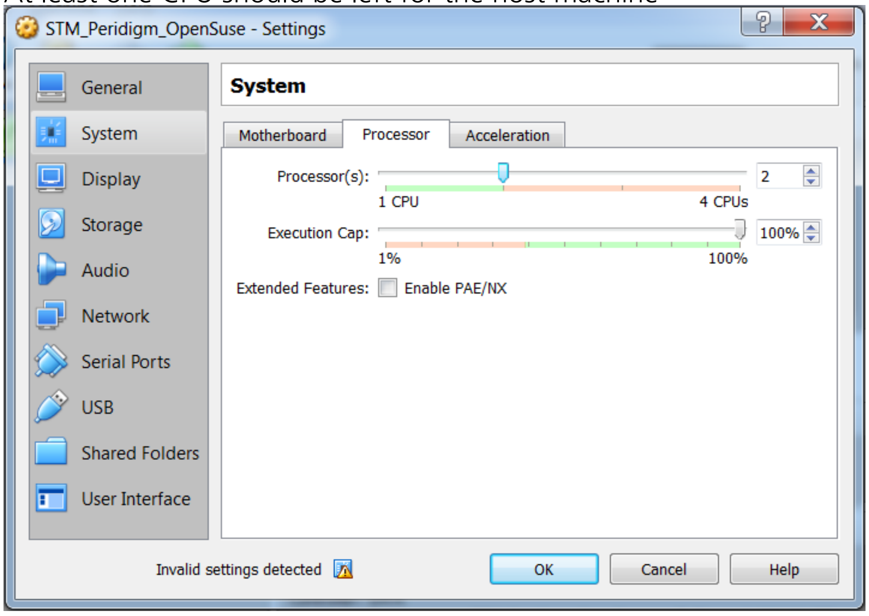

STM_Peridigm_OpenSuse - Settings

General

Snapshot Folder: E:\virtualization\Vi...TM_Peridigm_OpenSuse\Snapshots
Shared Clipboard: Bidirectional
Drag'n'Drop: Bidirectional

Invalid settings detected

- Ignore the warning about *Invalid settings detected* if this is only a RAM issue
- In the *System* tab
  - Select *Processor*
  - Set *Processor(s)* to your preferred value
  - At least one CPU should be left for the host machine


STM_Peridigm_OpenSuse - Settings

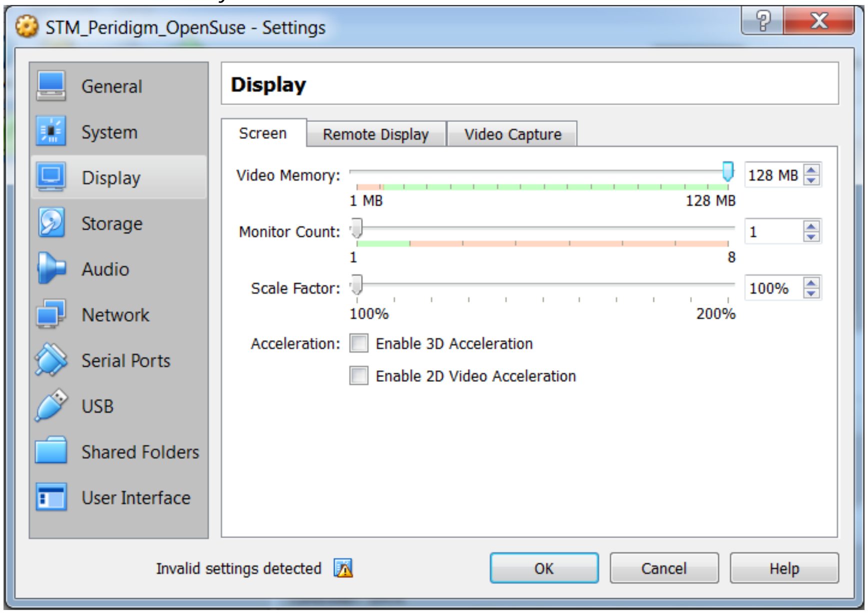
System

Processor(s): 2 (1 CPU – 4 CPUs)
Execution Cap: 100% (1% – 100%)
Extended Features: Enable PAE/NX

Invalid settings detected

- In the *Display* tab

- Select *Screen*
- Set *Video Memory* to maximum



Video Memory: 128 MB (1 MB – 128 MB)
Monitor Count: 1 (1 – 8)
Scale Factor: 100% (100% – 200%)
Acceleration: ☐ Enable 3D Acceleration
☐ Enable 2D Video Acceleration
Invalid settings detected

- Click *OK*

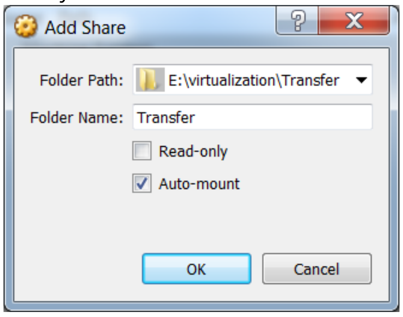
All modifications to the virtual machine preferences in step 4 can be modified after the installation of the virtual machine distribution in case the virtual machine is shut down.

2.2.3. Create a shared folder between host and virtual machine

1. In the host operating system:
   - Open a Windows Explorer
   - Create a shared folder for the file exchange between the host and the virtual machine operating system anywhere it suits you or use an existing one, here the shared folder is `E:\virtualization\Transfer`
   - Right-Click the newly created folder and click *Properties*
   - In the *Shared Folders/Freigabe* tab
     - Click on *Freigabe*
     - In the dialog appearing click on the combobox arrow and select *Anyone/-Jeder*
     - Click *Freigabe*
     - Click *Close*

- Be aware that the folder is visible in your whole network

2. Inside the virtual box manager:
   - Select the newly created virtual machine and click *Settings*
   - In the *Shared Folders* tab
     - Click the + folder button on the right
     - In *Folder Path:* create or add a transfer folder between your host system and your virtual machine



Add Share

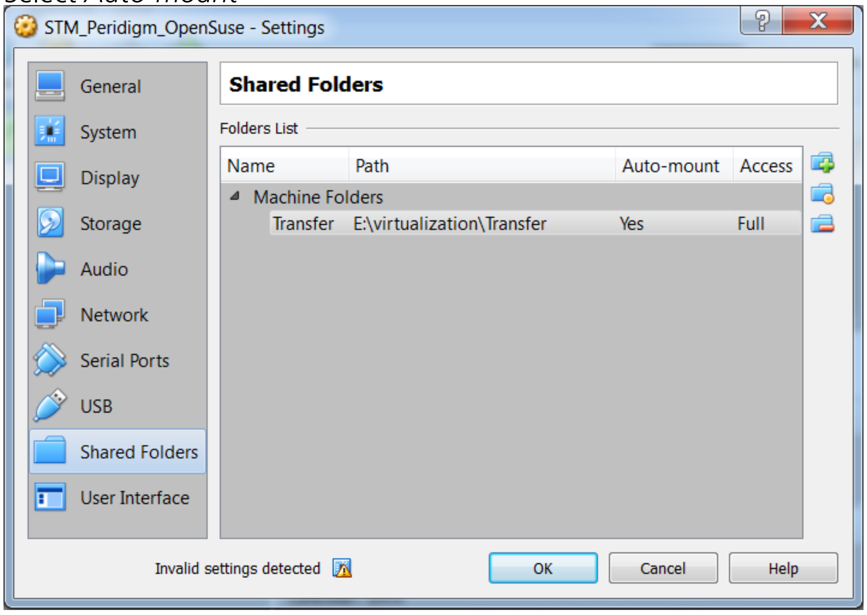
Folder Path: E:\virtualization\Transfer

Folder Name: Transfer

☐ Read-only

☑ Auto-mount

     - Select *Auto-mount*



STM_Peridigm_OpenSuse - Settings

Shared Folders

Folders List

Name	Path	Auto-mount	Access
Machine Folders			
Transfer	E:\virtualization\Transfer	Yes	Full

Invalid settings detected

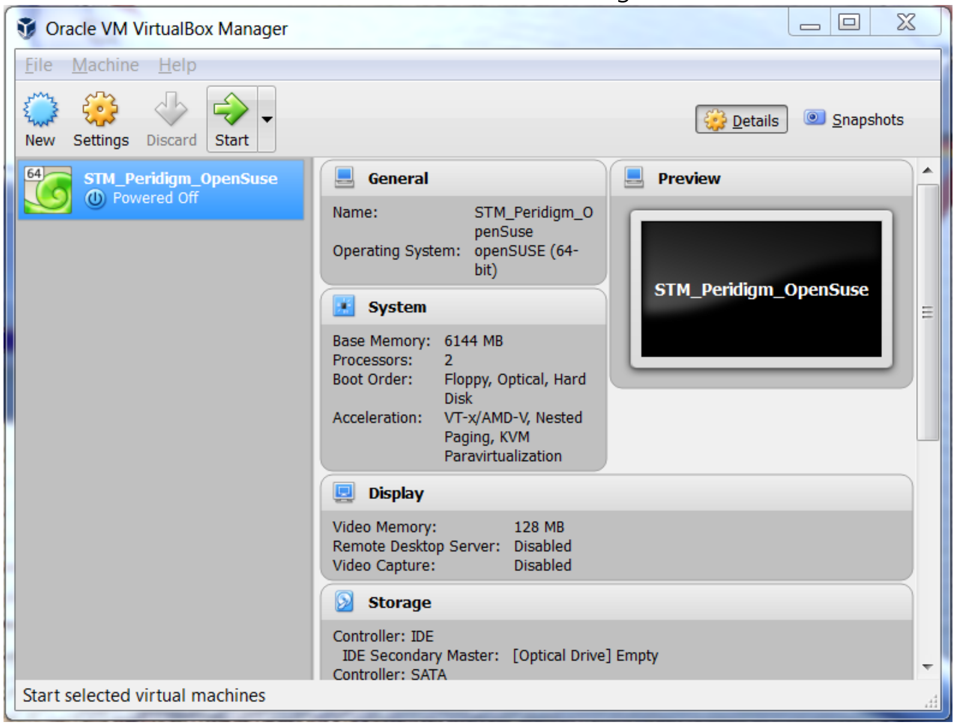
   - Click *OK*

2.2.4. Install the operating system in the virtual machine

Now, we can install the virtual machine operating system:

1. Start the installation

    ↗ Select the virtual machine in the *VirtualBox* Manager and click *Start*



General

Name: STM_Peridigm_OpenSuse
Operating System: openSUSE (64-bit)

System

Base Memory: 6144 MB
Processors: 2
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display

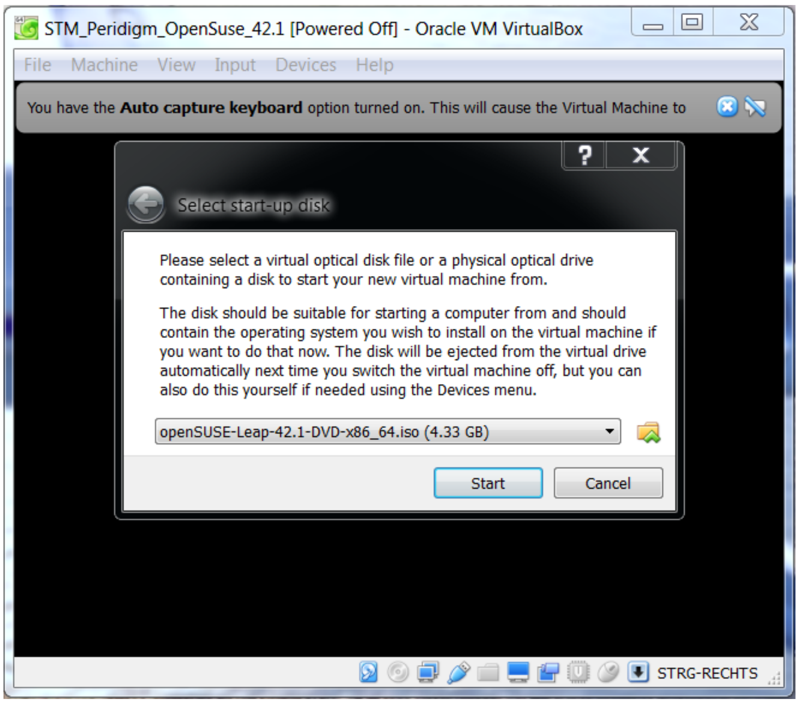
Video Memory: 128 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

Controller: IDE
 IDE Secondary Master: [Optical Drive] Empty
Controller: SATA

Start selected virtual machines

    ↗ Select the *openSUSE* image from section 2.1


STM_Peridigm_OpenSuse_42.1 [Powered Off] - Oracle VM VirtualBox
File Machine View Input Devices Help
Select start-up disk
Start
Cancel
STRG-RECHTS

You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to

Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

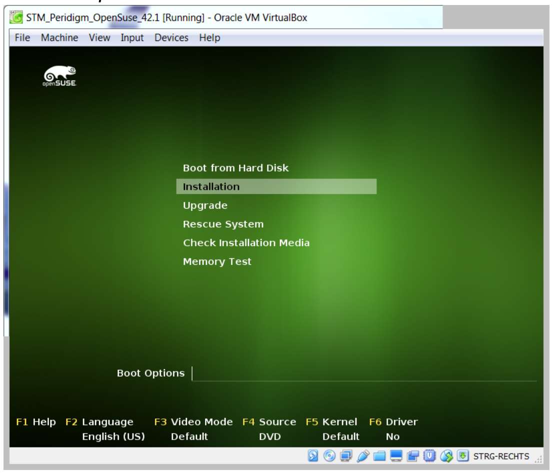
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

openSUSE-Leap-42.1-DVD-x86_64.iso (4.33 GB)

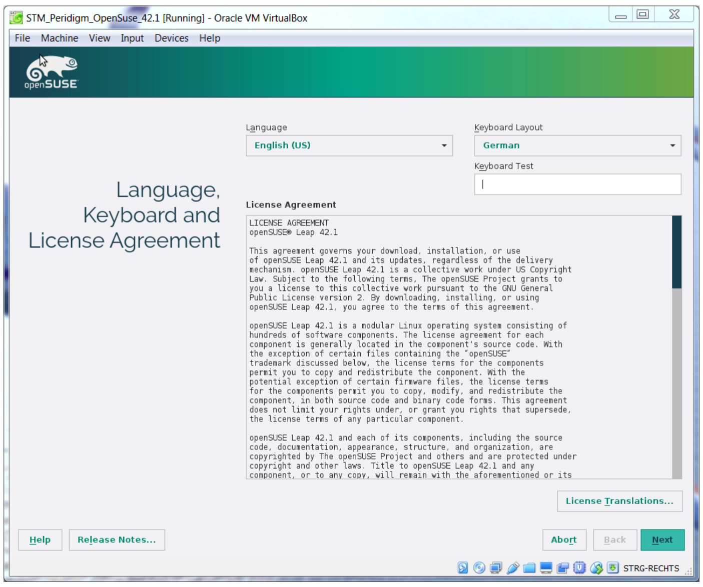
- Click *Start*

2. Setup the installation

- In the *openSUSE* boot menu select *Installation*


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
openSUSE
Boot from Hard Disk
Installation
Upgrade
Rescue System
Check Installation Media
Memory Test
Boot Options
F1 Help F2 Language English (US) F3 Video Mode Default F4 Source DVD F5 Kernel Default F6 Driver No
STRG-RECHTS

- Set the Language and Keyboard layout to your preferred option


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
openSUSE
Help
Release Notes...
Abort
Back
Next
STRG-RECHTS

Language, Keyboard and License Agreement

Language: English (US)

Keyboard Layout: German

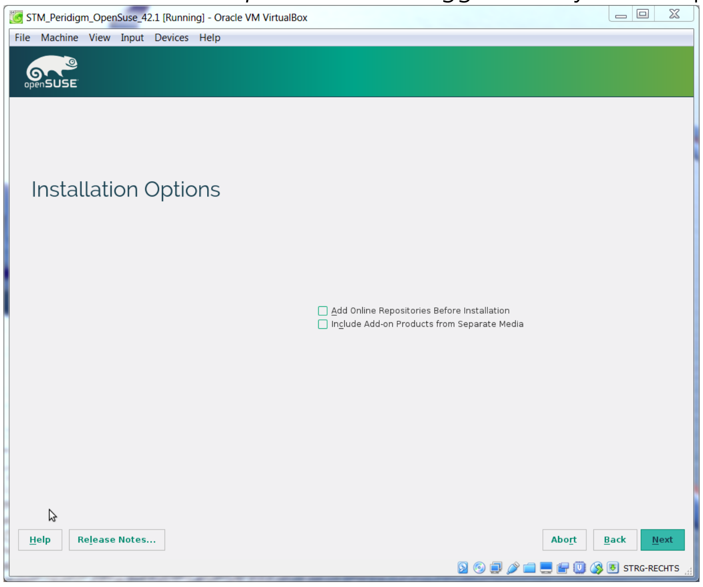
Keyboard Test

License Agreement

```
LICENSE AGREEMENT
openSUSE® Leap 42.1

This agreement governs your download, installation, or use
of openSUSE Leap 42.1 and its updates, regardless of the delivery
mechanism. openSUSE Leap 42.1 is a collective work under US Copyright
Law. Subject to the following terms, The openSUSE Project grants to
you a license to this collective work pursuant to the GNU General
Public License version 2. By downloading, installing, or using
openSUSE Leap 42.1, you agree to the terms of this agreement.

openSUSE Leap 42.1 is a modular Linux operating system consisting of
hundreds of software components. The license agreement for each
component is generally located in the component's source code. With
the exception of certain files containing the "openSUSE"
trademark discussed below, the license terms for the components
permit you to copy and redistribute the component. With the
potential exception of certain firmware files, the license terms
for the components permit you to copy, modify, and redistribute the
component, in both source code and binary code forms. This agreement
does not limit your rights under, or grant you rights that supersede,
the license terms of any particular component.

openSUSE Leap 42.1 and each of its components, including the source
code, documentation, appearance, structure, and organization, are
copyrighted by The openSUSE Project and others and are protected under
copyright and other laws. Title to openSUSE Leap 42.1 and any
component, or to any copy, will remain with the aforementioned or its
```

License Translations...

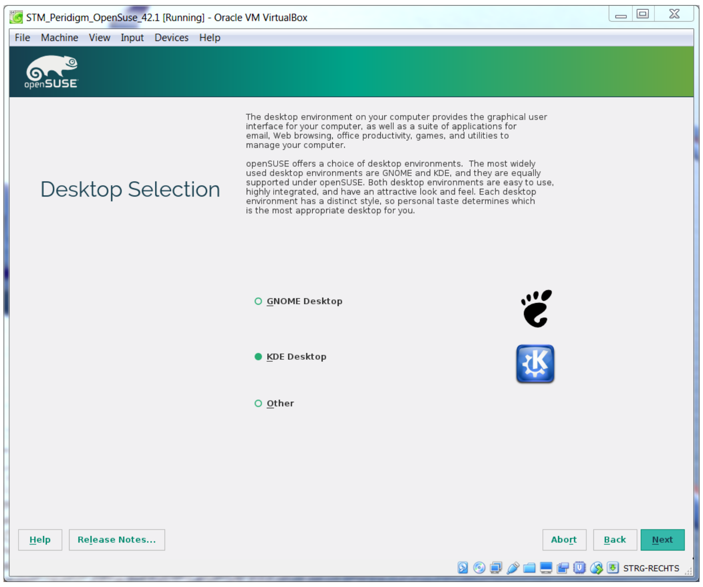
- Click *Next*
- In the *Installation Options* to not toggle on any of the options


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
openSUSE
Help
Release Notes...
Abort
Back
Next
STRG-RECHTS

Installation Options

☐ Add Online Repositories Before Installation
☐ Include Add-on Products from Separate Media

- Click *Next*
- Use the *Suggested Partition* and click *Next*
- Click on the Map to select your country to set *Clock and Time Zone* and click *Next*
- Use *KDE Desktop* in *Desktop Selection* and click *Next*


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
openSUSE
Help
Release Notes...
Abort
Back
Next
STRG-RECHTS

Desktop Selection

The desktop environment on your computer provides the graphical user interface for your computer, as well as a suite of applications for email, Web browsing, office productivity, games, and utilities to manage your computer.

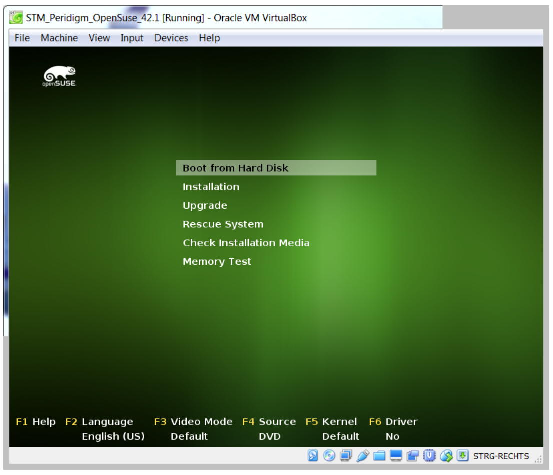
openSUSE offers a choice of desktop environments. The most widely used desktop environments are GNOME and KDE, and they are equally supported under openSUSE. Both desktop environments are easy to use, highly integrated, and have an attractive look and feel. Each desktop environment has a distinct style, so personal taste determines which is the most appropriate desktop for you.

- GNOME Desktop
- [✓ KDE Desktop]
- Other

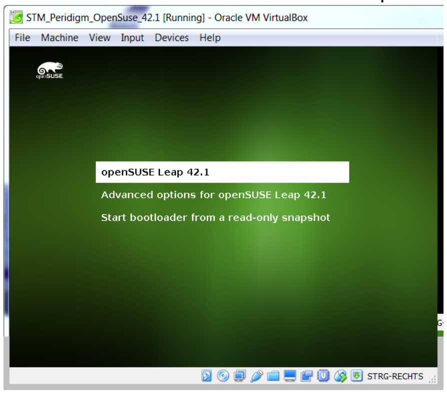
- Setup the first user name & password
  - You can use the same user and root password if you are and will always be the only user of your virtual machine
  - User: *Username* stm
    Password 13112
  - Admin: *Password* dlr-fa-13112-bs
  - Click next
- In the next windows click *Install*

3. After installation

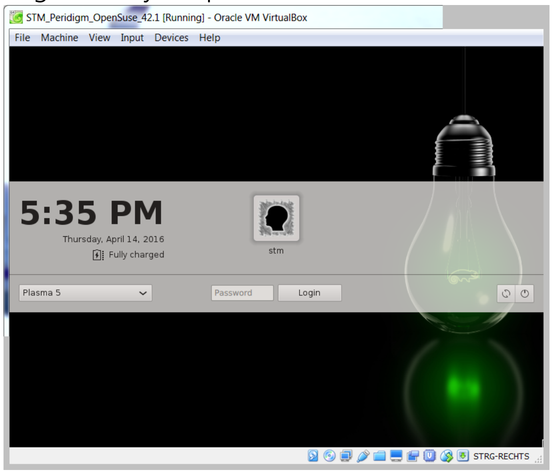
- After the installation is complete virtual system restarts. Now select *Boot from Hard Disk*


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Boot from Hard Disk
Installation
Upgrade
Rescue System
Check Installation Media
Memory Test
F1 Help F2 Language English (US) F3 Video Mode Default F4 Source DVD F5 Kernel Default F6 Driver No
STRG-RECHTS

→ Select the normal version and press Enter


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
openSUSE Leap 42.1
Advanced options for openSUSE Leap 42.1
Start bootloader from a read-only snapshot
STRG-RECHTS

→ Login with your password


STM_Peridigm_OpenSuse_42.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
5:35 PM
Thursday, April 14, 2016
Fully charged
stm
Plasma 5
Password
Login
STRG-RECHTS

→ Open a shell

- Login as root user, perform

  ```
  zypper refresh
  zypper update
  ```

  And select yes to perform an operating system update
- Restart the virtual machine operating system

Ta-daa you have a linux installation inside of a virtual machine.

2.2.5. User modifications to use shared folders

In order for the Linux users to use the virtual machine shared folders

1. Open *YaST2*
2. In the *Security and Users* tab open User and Group Management
3. Select the user and click *Edit*
4. Go to *Details* tab
5. On the right select *users* and *vboxsf* as *Additional Groups* and click *OK*
6. Restart your virtual system

The shared folders are mounted under `/media/`. In the current case the single shared folder is accessible under `/media/sf_transfer/` .

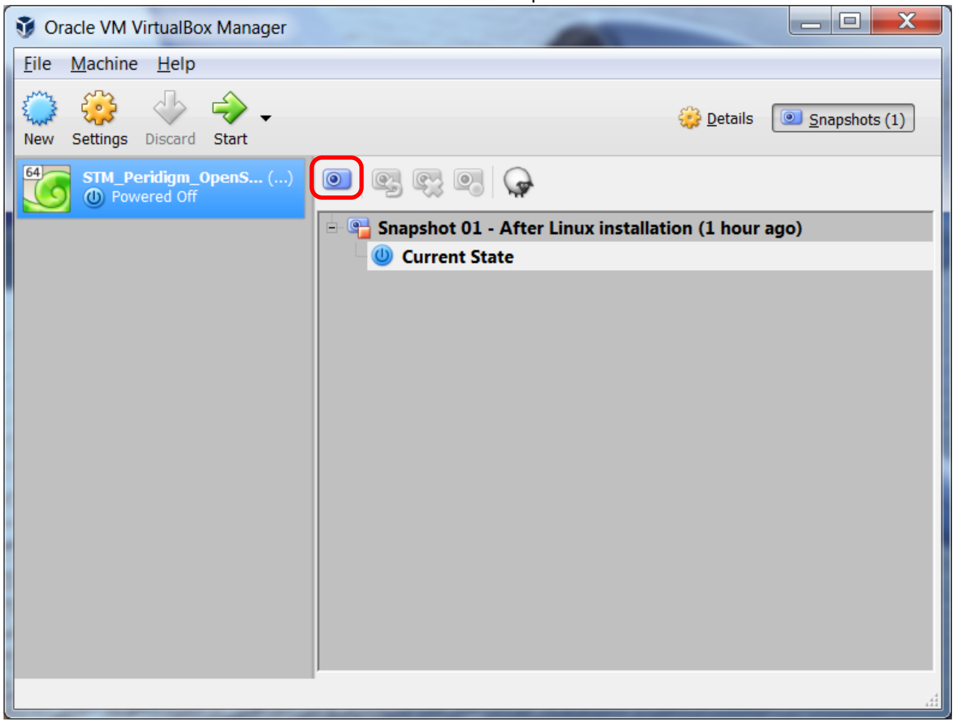
2.2.6. Save the virtual machine state

After the operating system installation it is recommended to save a snapshot of the current virtual machine state. Thus, it is always possible to reset your virtual machine to this state in case anything goes wrong during the *Peridigm* installation.

To create a snapshot:

1. Open the *VirtualBox* Manager
2. In the upper right corner select *Snapshots*

3. Click the little camera button to take a snapshot of the current virtual machine state


Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
Details Snapshots (1)
STM_Peridigm_OpenS... (...)
Powered Off
Snapshot 01 - After Linux installation (1 hour ago)
Current State

3. *Peridigm* Linux installation

3.1. Tested combinations

3.1.1. CPU-Architecture & operating system

3.1.1.1. System 1 - STM-Laptop

Hardware:	CPU:	Name:	Intel Core i7 M620
		Architecture:	x86_64
		Cores:	4
		Threads:	4
		Clock rate:	2.67GHz
	RAM:	Amount:	8Gb
Software:	OS:	Name:	openSUSE
		Version:	42.1
		Type:	64bit

3.1.1.2. System 2 - Virtual machine

Hardware:	Host:	CPU:	Name:	Intel Core i7-4600U
			Architecture:	x86_64
			Cores:	2
			Threads:	4
			Clock rate:	2.1GHz
		RAM:	Amount:	8Gb
	VM:	CPU:	Cores:	2
		RAM:	Amount:	6Gb
Software:	Host:			VirtualBox 5.0
	VM:	OS:	Name:	openSUSE
			Version:	42.1
			Type:	64bit

3.1.1.3. System 3 - STM-Cluster

Hardware:	CPU:	Name:	Intel Xeon E5-2407
		Architecture:	x86_64
		Number:	8
		Cores:	4
		Threads:	4
		Clock rate:	2.2GHz
	RAM:	Amount:	32Gb
Software:	OS:	Name:	SUSE Linux Enterprise Server
		Version:	11
		Patchlevel:	3
		Type:	64bit

3.1.2. Library and system combinations

	Setup		1	2	3	4
Basics	System		STM-Laptop	VirtualBox	VirtualBox	STM-Cluster
	Compiler	Fortran	gcc-fortran 4.8.5	gcc-fortran 4.8.5	gcc-fortran 4.8.5	gcc-fortran 4.8.5
		C	gcc 4.8.5	gcc 4.8.5	gcc 4.8.5	gcc 4.8.5
		C++	gcc-c++ 4.8.5	gcc-c++ 4.8.5	gcc-c++ 4.8.5	gcc-c++ 4.8.5
	CMake		3.4.3	3.4.3	3.5.1	3.5.2
	MPI	*Open MPI*	1.10.2	1.10.2	1.10.2	1.10.4
		MPICH	-	-	-	-
	Python		2.7.11	2.7.9	2.7.9	2.6.9
Libraries	*Boost*		1.55.0	1.55.0	1.60.0	?
	HDF5		1.8.16	1.8.16	1.10.0	?
	NetCDF-C		4.4.0	4.4.0	4.4.0	4.4.1
	Trilinos		12.4.2	12.4.2	12.6.1	?
	Peridigm		1.4.1	1.4.1	1.5.0	1.4.1

3.2. Basics

The following section describes the installation of the required basic packages for the *Peridigm* libraries. The subsections are in the order required for a proper installation.

3.2.1. Preliminary remarks

3.2.1.1. Installation shell

The following description is valid for an installation using the `bash` as shell. If you use any other shell, e.g. `ksh`, `csh` or their decendants you have to modify the environment variable parts of this guide or simply type `bash` and Enter in your `tcsh` to switch shells.

3.2.1.2. Download directories

During the course of this installation guide it will be necessary to download several source code packages. In the instructions the key `$DOWNLOAD_DIR` is the identifier for the download directory. The scripts in the appendix of this documents assume that

```
$DOWNLOAD_DIR=/usr/local/src/
```

You are free to choose any other folder as the download directory. If you do so, you have to modify the source code path in the scripts in the appendix accordingly.

3.2.1.3. `PATH` variable for user defined installation directories

It is assumed that the current installations are performed for all users of the device. Thus, the global installation directory `/usr/bin/` is used.

However, the installation directory for each tool can be changed, e.g. with

```
./configure --prefix="/home/$USER/TOOL"
```

If this is done, the path to the executables have to exported to the `PATH` variables.

```
export PATH="$PATH:/home/$USER/TOOL/bin"
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/home/$USER/TOOL/lib/"
```

This has to performed for each individual installation directory.

NEVER EVER FORGET THE `$PATH:` AND `$LD_LIBRARY_PATH:` IN THE BEGINNING OR A NEW EMPTY PATH VARIABLE WILL BE CREATED AND ADDED TO. YOU WILL NOT BE ABLE TO USE ANYTHING USEFUL ON YOUR DEVICE BECAUSE ALL ALIASES WILL BE DELETED.

3.2.1.4. Installation user

All installations are performed as `root` user. To make sure that the end user is allowed to use the installed programs make sure that they have the necessary permissions. The default installation directory for installations with a package manager or *zypper* is `/usr/bin`. You can check if the permissions are correct by opening a terminal, navigation to `/usr/bin` and the command

```
ls -l | grep PARTOFTOOLABBREVIATION
```

For `ls -l | grep gcc` the result might look something like this:

```
lrwxrwxrwx 1 root root      7  5. Feb 14:07 cc -> gcc-4.8
lrwxrwxrwx 1 root root      7  5. Feb 14:07 gcc -> gcc-4.8
-rwxr-xr-x 1 root root 755680 29. Okt 18:02 gcc-4.8
lrwxrwxrwx 1 root root     10  5. Feb 14:07 gcc-ar -> gcc-ar-4.8
-rwxr-xr-x 1 root root  27136 29. Okt 18:02 gcc-ar-4.8
lrwxrwxrwx 1 root root     10  5. Feb 14:07 gcc-nm -> gcc-nm-4.8
-rwxr-xr-x 1 root root  27136 29. Okt 18:02 gcc-nm-4.8
lrwxrwxrwx 1 root root     14  5. Feb 14:07 gcc-ranlib -> gcc-ranlib-4.8
-rwxr-xr-x 1 root root  27144 29. Okt 18:02 gcc-ranlib-4.8
```

In the first column the current rights are specified. The first symbol shows if the current entry is a link or not. The nine symbol afterwards are the three individual rights for user, group and other. The three individual rights are r-read, w-write and x-execute.

If the permissions are not set correctly, consult the documentation of *chmod*.

3.2.2. Root .bashrc file

You usually login as a normal user to *openSUSE* but changes to the system are performed as root user.

During this installation guide modifications of the .bashrc, a hidden file in the user home directory are requested. For the installation process as root user it is necessary that these modifications also take effect for the root user.

To achieve this a symbolic link is created in the root home directory to the modified .bashrc in the ordinary user directory. To achieve this open a console and perform the following steps

```
su                                              # Switch to root user
cd                                              # Change to root home
ln -sf /home/$USERNAME/.bashrc .                # Create symbolic link
```

This way, the root .bashrc file is always an identical copy of the one of the user with the name `$USERNAME` here.

3.2.3. Fortran & C & C++-compiler

Peridigm as well as *Python* python require an acceptable C or C++-compiler. *Trilinos* additionally needs a Fortran-compiler. Here, the free *GNU Compiler Collection* versions, short *GCC* are used. The current release and further informations can be found on

https://gcc.gnu.org/

Currently, there are two main versions available, *GCC*, which is basically *GCC* version 4.8, as well as *GCC5*. The installation of the used *Python* currently seems not to work with *GCC5*. Additionally, *Trilinos* needs a compiler that is `C++11` compliant and thus needs *GCC* version **4.7.2 or later**. Therefore *GCC* version 4.8 is used. If using Intel compilers, version 13 or later is required by *Trilinos*.

Normally, the *GCC* repository is already part of an *openSUSE* distribution. To check the availability of the *GCC*-repository in your *openSUSE* distribution open a terminal as root and use the following command to get a list of all repositories.

```
zypper repos
```

Installation with *YaST2*

To install the Fortran-, C- and C++-compilers of *GCC* with the package manager perform the following steps:

1. Open *YaST2*
2. Click on *Install software*
3. Go to the *Search* tab
4. Search for GCC
5. Check *gcc-fortran*, *gcc* and *gcc-c++*
6. Click on apply

Installation from source

ToDo

Installation with *openSUSE*-repository

To use *zypper* open a terminal as root. Use the following commands to install Fortran-, C- and C++-compilers of *GCC* from the repositories. Answer the questions if installation shall continue with yes.

```
zypper install gcc-fortran
zypper install gcc
zypper install gcc-c++
```

The installation usually is performed to `/usr/bin/`. If another installation directory is used, it has to be made sure, that this directory is part of the `$PATH`-variable. To check if this is the case, open a terminal type

```
echo $PATH
```

The installation directory has to be an entry of the printed string.

3.2.4. *CMake*

CMake is cross-platform free and open-source software for managing the build process of software using a compiler-independent method. It is maintained by Kitware. The official homepage of is *CMake*

https://cmake.org/

Trilinos release 12.4 and higher use a *CMake* build system, which requires *CMake* version **2.8.11 or newer**.

Installation from source

In order to install *CMake* from the official or any other binary source open a terminal and login as root. Change directory to the designated download folder, e.g. `/usr/local/src/` and perform the following steps:

```
cd $DOWNLOAD_DIR
wget http://www.cmake.org/files/v3.4/cmake-3.4.3.tar.gz # Download
tar xvfz cmake-3.4.3.tar.gz                             # unzip
cd cmake-3.4.3                                          # go into directory
./configure --prefix=/usr/local/bin/cmake-3.4.3 > configure_cmake.log 2>&1
make > make_cmake.log 2>&1                              # build
make install > make_install_cmake.log 2>&1              # install
```

If you want to see the available configuration options, run the command below in the terminal.

```
./configure --help
```

In order to configure the installation directory of *CMake* before installation, run the command below

```
./configure --prefix=/opt/cmake
```

After installation without any errors you can verify the installation by running the command below:

```
/usr/local/bin/cmake-3.4.3/bin/cmake -version
```

The output should look something like below (depending upon *CMake* version you are installing).

```
cmake version 3.4.3
```

Afterwards, the *CMake*-directory has to be added to the `PATH` environment variable

```
export PATH=$PATH:/usr/local/bin/cmake-3.4.3/bin
```

Installation with *openSUSE*-repository

For the installation of the build process manager *zypper* visit:

http://software.opensuse.org/download.html?project=server%3Airc&package=cmake

You can either choose the 1-Click-installation or add the repository and install manually. For the latter login to a terminal as `root` and type

```
zypper addrepo http://download.opensuse.org/repositories/server:irc/
    openSUSE_Leap_42.1/server:irc.repo
zypper refresh
zypper install cmake
```

3.2.5. MPI

For parallel computations on multiple cores an implementation of the Message Passing Interface is required. Only **one** of the the following possibilities is required. The current implementation uses *Open MPI*.

3.2.5.1. *Open MPI*

Open MPI is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. The current version and further information can be found at

https://www.open-mpi.org

Installation with *YaST2*

To install *Open MPI* with the package manager perform the following steps:

1. Open *YaST2*
2. Click on *Install software*
3. Go to the *Search* tab
4. Search for Open MPI
5. Check Open MPI
6. Click on apply

Installation with *openSUSE*-repository

The *Open MPI*-repository is part of the openSUSE distribution. Therefore, it can be directly installed from the system repositories.

To use *zypper* open a terminal as root. Use the following commands to install *Open MPI* from the repositories. Answer the questions if installation shall continue with yes.

```
zypper install openmpi
```

Installation from source

The Gzipped tarball source files can be obtained from

https://www.open-mpi.org/software/ompi/

In the subfolder choose the version of your liking. Change directory to the designated download folder, e.g. `/usr/local/src/` and perform the following steps:

```
cd $DOWNLOAD_DIR
wget https://www.open-mpi.org/software/ompi/v1.10/downloads/openmpi
    -1.10.2.tar.gz
tar xvfz openmpi-1.10.2.tar.gz                  # unzip
cd openmpi-1.10.2                               # go into directory
./configure --prefix=/usr/local/lib/openmpi-1.10.2 > configure_openmpi.
    log 2>&1
make > make_openmpi.log 2>&1                    # build
make altinstall > make_install_openmpi.log 2>&1
```

If no previous version of *Open MPI* exists use `make install` instead of `make altinstall`.

Set the `PATH` variables

Unfortunately the *Open MPI* installation does not work out of the box. You need to set the `PATH` and `LD_LIBRARY_PATH` variables and edit a configuration file first.

The `LD_LIBRARY_PATH` must be set so that *mpi4py* can find the *Open MPI* libraries.

In bash do for 32-bit

```
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/lib
```

or 64-bit

```
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/
    lib64
```

We recommend you add this line to your `.bashrc` file in case you use *Bash* or call `setenv` and edit the `.cshrc` file if you use a C shell so that the variable is set correctly for all sessions. For the modification of user `.bashrc` file for all libraries, please consult section A.7.

3.2.5.2. *MPICH*

MPICH is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard.

Use the operating system distribution

ToDo

Use 1-click install

Go to

https://software.opensuse.org/package/mpich

and choose 1-Click-Install or download the rpm-file from the source.

Installation with *openSUSE*-repository

ToDo

Installation from source

Try

```
cd $DOWNLOAD_DIR
zypper si -d mpich2                              # install the build deps
                                                 # for the previous version
wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz
tar xvfz mpich-3.2.tar.gz                        # unzip
cd mpich-3.2                                     # go into directory
./configure > configure_mpich.log 2>&1
make > make_mpich.log 2>&1                       # build
make altinstall > make_install_mpich.log 2>&1   # install
```

3.2.6. *Python*

Use the operating system distribution

Python is already part of an openSUSE standard installation since also system components require python. The installed version can be shown in the terminal by the command

```
python -V
```

Packages are available for both *Python* 2.7 as well as *Python* 3.x. A parallel installation if *Python* 2 and *Python* 3 possible without problems or package conflicts.

To update the python distribution to the newest available state in the OS repositories, open a terminal, login as `root\verb` and use the following command

```
zypper update python
```

Additionally, `python-devel` is required, so

```
zypper install python-devel
```

Perform a new installation

If no initial version of *Python* is present in the operating system it is necessary to download the source and install the source. For the latest or required version of *Python* visit

http://www.python.org/download/

For the installation, open a terminal and change directory to `/home/USERNAME/bin` for a single-user installation or `/usr/local/bin` for an installation for all users

```
cd $DOWNLOAD_DIR
wget https://www.python.org/ftp/python/2.7.11/Python-2.7.11.tgz # Download
tar xvfz Python-2.7.11.tgz                               # unzip
cd Python-2.7.11                                         # go into directory
./configure
make                                                     # build
make altinstall                                          # install
```

Afterwards, you are free to delete the downloaded Gzipped source tarball, here

```
cd $DOWNLOAD_DIR
rm Python-2.7.11.tgz                                     # delete
```

3.3. System libraries

3.3.1. Necessary libraries

Install or update the following system libraries in advance of the *Peridigm* library installation.

```
zypper install libbz2-devel
zypper install zlib-devel

zypper install m4

zypper install blas
zypper install lapack
zypper install libX11-devel
```

3.3.2. Libraries that are not supposed to be installed yet

Open *YaST2* and go to *Software Management*. In the *Search* field type *HDF5* and look whether an older version than the *HDF5* version you want to use is already installed on the system.

This is the case if you have *octave-forge-netcdf* installed on your system. If you do not really use this package you can uninstall it in *YaST2* together with the installed *HDF5* libraries.

If you really need *octave-forge-netcdf* which comes with *HDF5*-1.8.15 and have it installed already you can skip the installation of *HDF5* described here and change the rest of the installation to use version 1.8.15 of *HDF5*.

3.4. Libraries

3.4.1. *Boost*

Boost provides free peer-reviewed portable C++ source libraries. *Boost* libraries are intended to be widely useful, and usable across a broad spectrum of applications. For more informations and the current release visit

http://www.boost.org/

Peridigm requires *Boost*, version **1.37 or later**, including the *regex* and *unit_test* compiled libraries. *Boost* installations on many systems include header files only. This is not sufficient, the required libraries must be compiled and installed. To ensure proper execution of *Peridigm* and its unit tests, add the *Boost* directory `$INSTALL_DIR/lib` to your `LD_LIBRARY_PATH` (Linux) and/or `DYLD_LIBRARY_PATH` (Mac) environment variables.

Use 1-click install

There is a *RPM Package Manager* file available on the *openSUSE* homepage. However, *Boost* requires some additional libraries to be compiled specifically. Therefore, it is not tested if the following installation with the 1-click install option is sufficient for *Peridigm*. It is recommended to use the installation using the source files from a Gzipped tarball as described in the next paragraph.

Go to

https://software.opensuse.org/package/boost

and choose 1-Click-Install or download the rpm-file from the source. Be cautious, the version offered is not necessarily up to date. Consult

http://www.boost.org/

for the current release.

Installation from source

The current release of *Boost* is available from

http://www.boost.org/

and sourceforge:

http://sourceforge.net/projects/boost/files/boost/

Originally, I tried to install the then-current version 1.60. Unfortunately, this led to compilation errors in combination with the current version of the *GCC*-compiler. After searching for a solution of the issue, version 1.55 is recommended.

First, the additional libraries `libbz2-devel` and `zlib1g-dev` have to be installed in section 3.3 before we can install *Boost*. Additional a new root shell must be opened to load the new environment variable additions for *Open MPI*.

```
cd $DOWNLOAD_DIR
wget http://sourceforge.net/projects/boost/files/boost/1.55.0/
    boost_1_55_0.tar.gz
tar xvfz boost_1_55_0.tar.gz                        # unzip
cd boost_1_55_0                                     # go into directory
```

Afterwards, create the *Boost* build script as described in section A.1. In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created install script and execute it with

```
./install_boost-1.55.0.sh > install_boost.log 2>&1
```

The printout of the installation is written to `install.log`. It should be checked if all components of *Boost* are compiled correctly.

Afterwards, the *Boost*-directory has to be added to the `LD_LIBRARY_PATH` environment variable

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/boost-1.55.0/lib
```

3.4.2. *HDF5*

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. For further information visit https://www.hdfgroup.org/HDF5/

HDF5 version **1.8.9 or newer** is required by *NetCDF-C* and the *SEACAS Trilinos* package. *HDF5* should be configured with the `--enable-parallel option`.

Installation with *openSUSE*-repository

HDF5 is available in an *openSUSE*-repository and can be installed using *zypper*. However, it is recommended to use the manual install with the Gzipped tarball to make sure the correct options are set.

```
zypper addrepo http://download.opensuse.org/repositories/home:ocefpaf/
    openSUSE_Tumbleweed/home:ocefpaf.repo
zypper refresh
zypper install hdf5
```

Installation from source

The *HDF5* source code is available from

https://www.hdfgroup.org/HDF5/

Download the source code for your platform

```
cd $DOWNLOAD_DIR
wget http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.16.tar.gz
tar xvfz hdf5-1.8.16.tar.gz                           # unzip
cd hdf5-1.8.16                                        # go into directory
```

Afterwards, create the *HDF5* build script as described in section A.2. In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created install script and execute it with

```
./install_hdf.sh > install.log 2>&1
```

Afterwards, the *HDF5*-directory has to be added to the `PATH` and `LD_LIBRARY_PATH` environment variable

```
export PATH=$PATH:/usr/local/bin/hdf5-1.8.16/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/hdf5-1.8.16/lib64
```

3.4.3. *NetCDF-C*

NetCDF-C (Network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. Distributions are provided for Java and C/C++/Fortran. For more information visit

http://www.unidata.ucar.edu/software/netcdf/

NetCDF-C is required by the *Trilinos SEACAS* package. *NetCDF-C* should be configured with the `--disable-netcdf-4` and `--disable-dap` options.

Installation from source

The *NetCDF-C* source code is available from the *NetCDF-C* homepage. Download the source code for your platform

```
cd $DOWNLOAD_DIR
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4.4.0.tar.gz
tar xvfz netcdf-4.4.0.tar.gz                       # unzip
cd netcdf-4.4.0                                    # go into directory
```

Prior to compiling *NetCDF-C*, it is recommended that you modify the file `netcdf.h` in `$DOWNLOAD_DIR/netcdf-4.4.0/include/` to better support large-scale *Peridigm* simulations. Modify the following `#define` statements in the `netcdf.h` file. Change the values to match what is given below.

```
#define NC_MAX_DIMS      65536
#define NC_MAX_ATTRS      8192
#define NC_MAX_VARS     524288
#define NC_MAX_NAME        256
#define NC_MAX_VAR_DIMS      8
```

Afterwards, create the *NetCDF-C* build script as described in section A.3. In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created install script and execute it with

```
./install_netcdf.sh > install.log 2>&1
```

Due to an apparent glitch in the *NetCDF-C* installer, in some cases it may be necessary to manually copy the file `$DOWNLOAD_DIR/netcdf-4.4.0/include/netcdf_par.h` from the source distribution into the installation `include` subdirectory.

```
cp $DOWNLOAD_DIR/netcdf-4.4.0/include/netcdf_par.h /usr/local/bin/netcdf
    -4.4.0/include/
```

Afterwards, the *NetCDF-C*-directory has to be added to the `PATH` and `LD_LIBRARY_PATH` environment variable

```
export PATH=$PATH:/usr/local/bin/netcdf-4.4.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/netcdf-4.4.0/lib64
```

3.4.4. *Trilinos*

The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. A unique design feature of Trilinos is its focus on packages. For more information visit

https://trilinos.org/

A number of *Trilinos* packages are required by *Peridigm*. The *Trilinos* source code distribution includes the full set of *Trilinos* packages, each of which may be activated or deactivated using *CMake* build options, as described below. It is recommended that Makefiles be created by running `cmake` from the command line, as opposed to using the `ccmake` GUI.

The current release of *Trilinos* can be obtained from the download section of the *Trilinos* homepage. The download needs a short registration with a valid email-address. The download link is likely to be not reachable without the registration.

```
cd $DOWNLOAD_DIR
wget http://trilinos.csbsju.edu/download/files/trilinos-12.4.2-Source.tar
    .gz
tar xvfz trilinos-12.4.2-Source.tar.gz
```

Trilinos does not allow the use of the directory with the source-files for the further progress of the installation. Therefore, create a new folder

```
mkdir trilinos-12.4.2
```

and copy the file from section A.4 to the new folder. Change the line for the *Open MPI*- and the *Trilinos*-source-directory (last line) if necessary.

In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created path and execute it with

```
./cmake_trilinos.cmake > cmakeopts.log 2>&1
```

Once *Trilinos* has been successfully configured, it can be compiled and installed as follows:

```
make -j 4
```

If there occur any errors during the compilation of *Trilinos* visit section B. For comiling with `make -j 4` more than 8GB of RAM are necessary. If you do not know if there were any compilation errors have occured due to the long duration of the compilation process, repeat

```
make
```

after the original compilation with `make -j 4`. Only failed compilations are repeated. Afterwards perform

```
make install
```

The final installation can be found in the folder specified in `CMAKE_INSTALL_PREFIX:PATH` in the script from section A.4.

Afterwards, *Trilinos* has to be added to the `PATH` and `LD_LIBRARY_PATH` environment variables to later use the *Trilinos* decomposition features for the model decomposition for calculation on multiple processors.

```
export PATH=$PATH:/usr/local/bin/trilinos-12.4.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/trilinos-12.4.2/
    lib
```

3.5. *Peridigm*

3.5.1. Download

3.5.1.1. Download the official release

The current official release of *Peridigm* can be obtained from the download section of

https://peridigm.sandia.gov/

The download needs a short registration with a valid email-address. Download the `tgz` file of your preferred *Peridigm* version to `$DOWNLOAD_DIR`. Unpack the archive:

```
cd $DOWNLOAD_DIR
tar xvfz Peridigm_1.4.1.tgz
```

3.5.1.2. Download the latest master version from *GitHub*

The *Peridigm* repository is available from *GitHub* and can be downloaded from:

https://github.com/peridigm/peridigm

To obtain the latest master release go to that homepage and select the *master* branch in the top left and click on *Download zip* in the top right corner as shown by the red rectangles in Figure 3.1 or click this link.

Unpack the archive:

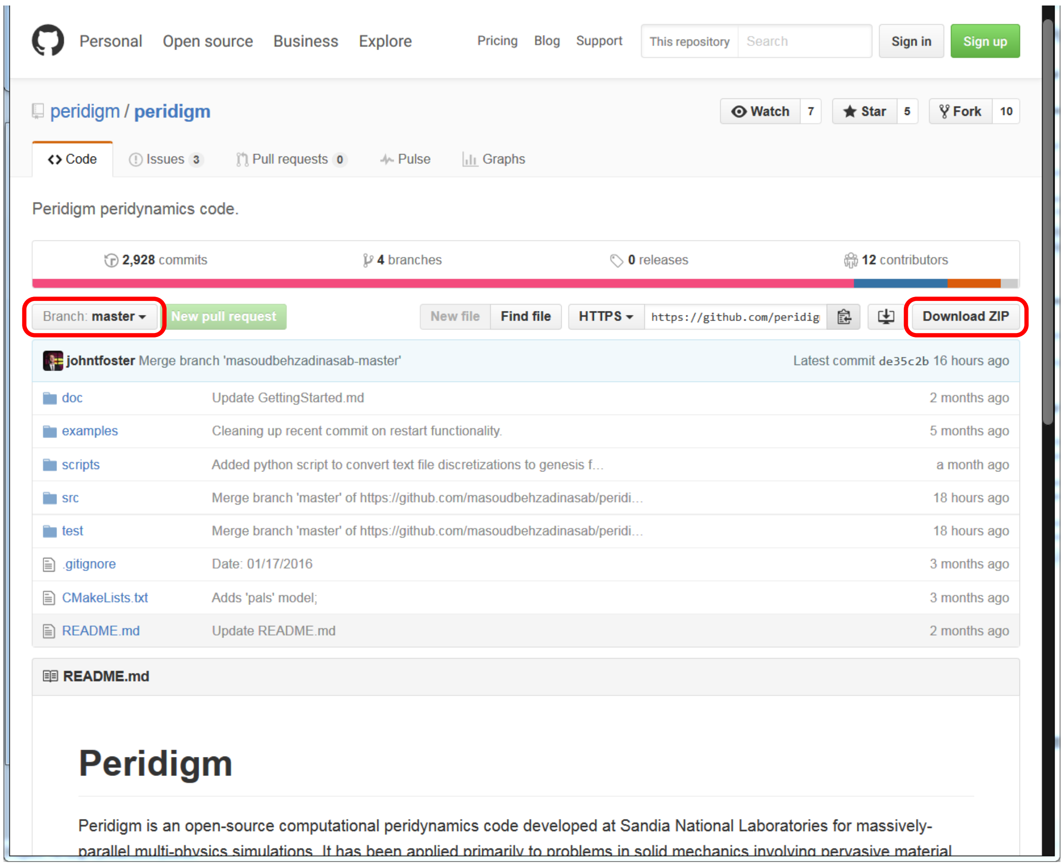
```
cd $DOWNLOAD_DIR
unzip peridigm-master.zip
mv peridigm-master Peridigm-1.4.1-Source
```

3.5.1.3. Checkout the latest version from *GitHub*

With svn

Use *kdesvn* to checkout the latest version. Checkout

https://github.com/peridigm/peridigm.git/trunk



peridigm / peridigm

Peridigm peridynamics code.

2,928 commits 4 branches 0 releases 12 contributors

https://github.com/peridig

johntfoster Merge branch 'masoudbehzadinasab-master' Latest commit de35c2b 16 hours ago

doc	Update GettingStarted.md	2 months ago
examples	Cleaning up recent commit on restart functionality.	5 months ago
scripts	Added python script to convert text file discretizations to genesis f…	a month ago
src	Merge branch 'master' of https://github.com/masoudbehzadinasab/peridi…	18 hours ago
test	Merge branch 'master' of https://github.com/masoudbehzadinasab/peridi…	18 hours ago
.gitignore	Date: 01/17/2016	3 months ago
CMakeLists.txt	Adds 'pals' model;	3 months ago
README.md	Update README.md	2 months ago

README.md

Peridigm

Peridigm is an open-source computational peridynamics code developed at Sandia National Laboratories for massively-parallel multi-physics simulations. It has been applied primarily to problems in solid mechanics involving pervasive material

Figure 3.1.: *Peridigm GitHub* repository screenshot

With git

Use *git* on the cluster:

```
git clone https://github.com/peridigm/peridigm.git
```

3.5.2. Compiling & installation

Peridigm utilizes the *CMake* build system. It is recommended that Makefiles be created by running `cmake` from the command line, as opposed to using the `ccmake` GUI.

The installation here is described for the then official *Peridigm* version 1.4.1. The steps for version 1.5 are identical besides the changes in folder names.

Peridigm does not allow the use of the directory with the source-files for the further progress of the installation. Therefore, create a new folder

```
mkdir Peridigm-1.4.1
```

and copy the file from section A.5 to the new folder. Change the lines for the library paths and the *Peridigm*-source-directory (last line) if necessary.

In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created path and execute it with

```
./cmake_peridigm.cmake > cmakeopts.log 2>&1
```

Once *Peridigm* has been successfully configured, it can be compiled and installed as follows:

```
make -j 4
make install
```

The default location for the created binary is `/usr/local/bin/`. In case you want to create the binary at a different location add the following line to the script from section A.5

```
-D CMAKE_INSTALL_PREFIX=/PATH/TO/DESTINATION
```

After installation make sure to change permissions of the installation directory for the necessary users and groups.

3.5.3. After building and installing

[1] recommends to run `ctest` in your build directory after building and installing *Peridigm* to run all tests and confirm that you have a clean build. Alternatively, *Peridigm* offers an own test suite which is used here.

To be able to execute the test you first have to temporarily change the source and installation folder owner. Thus perform the following commands before executing the test.

```
chown -R $USERNAME:$GROUPNAME Peridigm-1.4.1
chown -R $USERNAME:$GROUPNAME Peridigm-1.4.1-Source
```

Before starting the test-suite as a normal user, open a new shell or use an existing one and re-register your username to load the current state of the `.bashrc`-file:

```
su - $USERNAME
```

The *Peridigm* test suite is then run from the terminal as non-root user as follows:

```
make test
```

Remember to revert the modifications to the *Peridigm* directory ownership after it is assured that all tests are passed with:

```
chown -R root:root Peridigm-1.4.1
chown -R root:root Peridigm-1.4.1-Source
```

3.5.4. Install a local version on the cluster

1. Login to the STM cluster and move to a directory of your convenience
2. Clone your local Peridigm version from GitHub with `git clone` or download the master zip-file and unpack, see section 3.5.1.1
3. Allocate cluster-node exclusively for building

   ```
   salloc --exclusive
   ```
4. Load build and library environment

   ```
   . /cluster/software/slurm/etc/env.d/mpibuild.sh
   . /cluster/software/slurm/etc/env.d/peridigm.sh
   ```
5. Change directory to the folder, where the unpacked Peridigm source folder is located

   ```
   cd ~/src/peridigm
   ```
6. Create a build-directory and go to the new directory

```
mkdir peridigm-build
cd peridigm-build
```

7. Save the following lines in the file `cmake_peridigm.cmake`. Change the path in the code to the correct location.

```
cmake \
-D CMAKE_BUILD_TYPE:STRING=Release \
-D CMAKE_INSTALL_PREFIX=/home/USER/peridigm-build \
-D CMAKE_CXX_FLAGS:STRING='-O2 -Wall -std=c++11 -pedantic -Wno-long-
   long -ftrapv -Wno-deprecated' \
/home/USER/peridigm-master
```

8. Call CMake via terminal with the given code:

```
./cmake_peridigm.cmake
```

9. Make

```
make -j 8
```

10. Test

```
make test
```

11. Create the executable

```
make install
```

12. Exit salloc shell

```
exit
```

13. The executable is now located at `~/src/peridigm/peridigm-build/bin`
14. For how to execute the local version using the cluster queuing-system look at the Peridigm User Guide

A complete script for cloning Peridigm from GitHub, compiling and installing on the cluster can be found in subsection A.5.2.

3.5.5. Use of *Docker*

http://johntfoster.github.io/posts/peridigm-without-building-via-Docker.html

4. Running *Peridigm*

A dedicated description on how to run *Peridigm* can be found in the *Peridigm* Users Guide which is part of the same repository as this document.

5. Install *ParaView*

ParaView is an open source multiple-platform application for interactive, scientific visualization. The current release and further informations can be found on

http://www.paraview.org/

ParaView was developed to analyze extremely large datasets using distributed memory computing resources.

5.1. Linux installation

Use 1-click install

Go to

https://software.opensuse.org/download/package?project=science&package=paraview

and choose 1-Click-Install or download the rpm-file from the source.

Installation with *openSUSE*-repository

ParaView is part of the *openSUSE* Science Repository. The repository can be included into the package management.

```
zypper addrepo http://download.opensuse.org/repositories/science/
   openSUSE_Leap_42.1/science.repo
zypper refresh
zypper install paraview
```

Installation from source

The source code of the current *ParaView* release is available from the download section of the *ParaView* homepage. For an installation using the source code and *CMake* please consult

http://www.paraview.org/Wiki/ParaView:Build_And_Install

5.2. Windows installation

Go to the download section of the *ParaView* homepage and download the binary installers for your windows operating system architecture. Afterwards, simply run the executable installer and follow the instructions.

6. Install everything for *FETranslator*

FETranslator is a *Java*-based tool to translate models between finite element software. *FE-Translator* implements the conversion of meshes from commercial FE tools into the format that *Peridigm* is capable of using as a discretization for the creation of peridynamic collocation points.

In order to use the *FETranslator* an implementation of the *Java* Runtime Environment (JRE) is necessary. To translate the mesh into binary format the tool *ncgen* from *NetCDF-C* is required.

6.1. Linux

6.1.1. Java

openSUSE comes with a pre-installed version of the *openJDK* which is a free and open source implementation of the Java Platform, Standard Edition (Java SE). *openJDK* should be perfectly capable of running *FETranslator*. To see if and which version of *Java* is installed on your system open a shell and type:

```
java -version
```

However, since additions and changes to the *FETranslator* can be necessary, a *Java*-capable IDE is required. The Oracle *Java* Development Kit (JDK) offers an integrated solution with the JRE and *NetBeans* as IDE.

6.1.1.1. Install only *Java* development kit (JDK)

1. Go to: http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Click on *Java Platform (JDK)*
3. Accept the License Agreement
4. Open a shell and type `lsb_release -a` and check your operating system architecture (32bit: i586; 64bit: x86_64)

5. Click on the according `rpm` file for your Linux version (here: jdk-8u91-linux-i586.rpm for 32bit or jdk-8u91-linux-x64.rpm for 64bit, we use 64bit)
6. In the dialog choose *Save File* and save the file somewhere convenient on your system
7. Open a root shell or a normal shell and switch to root user with

```
su -
```

8. Change directory to the folder where the RPM file is located
9. Type

```
zypper install jdk-8u91-linux-x64.rpm
zypper install update-alternatives
update-alternatives --install /usr/bin/java java /usr/java/jdk1.8.0
    _91/bin/java 1065
update-alternatives --install /usr/bin/javac javac /usr/java/jdk1.8.0
    _91/bin/javac 1065
update-alternatives --install /usr/bin/jar jar /usr/java/jdk1.8.0_91/
    bin/jar 1065
update-alternatives --install /usr/bin/javaws javaws /usr/java/jdk1
    .8.0_91/bin/javaws 1065
update-alternatives --config java
java -version
nedit /home/$USERNAME/.bashrc &
```

10. Add `export JAVA_HOME=/usr/java/jdk1.8.0_91/` to the `.bashrc`-file

6.1.1.2. Install *Java* development kit (JDK) with *NetBeans*

1. Go to: http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Click on *NetBeans with JDK*
3. Accept the License Agreement
4. Open a shell and type `lsb_release -a` and check your operating system architecture (32bit: i586; 64bit: x86_64)
5. Click on the according `sh` file for your Linux version (here: `jdk-8u91-nb-8_1-linux-x64.sh`)
6. In the dialog choose *Save File* and save the file somewhere convenient on your system
7. Open a root shell or a normal shell and switch to root user with

```
su -
```

8. Change directory to the folder where the .sh file is located
9. Change the installer file's permissions so it can be executed:

```
chmod u+x <installer-file-name>
```

10. Type

    ```
    ./<installer-file-name>
    ```

11. In the installation wizard:
    a) At the Welcome page of the installation wizard, click *Next*.
    b) At the JDK Installation page, specify the directory where to install the JDK, here `/usr/local/java/jdk1.8.0_91`, and click *Next*.
    c) At the *NetBeans* IDE Installation page, do the following:
       - Specify the directory for the *NetBeans* IDE installation (here `/usr/local/java/netbeans-8.1`)
       - Accept the default JDK installation to use with the IDE or specify another JDK location.
    d) Accept the default JDK installation to use with the IDE or specify another JDK location.
    e) Click *Next*
    f) Review the Summary page to ensure the software installation locations are correct.
    g) Click *Intall* to begin the installation.
    h) At the Setup Complete page, provide anonymous usage data if desired, and click *Finish*
    i) When the installation is complete, you can view the log file, which resides in the following directory: `~/.nbi/log`.

12. Type

    ```
    zypper install update-alternatives
    update-alternatives --install /usr/bin/java java /usr/local/java/jdk1
        .8.0_91/bin/java 1065
    update-alternatives --install /usr/bin/javac javac /usr/local/java/
        jdk1.8.0_91/bin/javac 1065
    update-alternatives --install /usr/bin/jar jar /usr/local/java/jdk1
        .8.0_91/bin/jar 1065
    update-alternatives --install /usr/bin/javaws javaws /usr/local/java/
        jdk1.8.0_91/bin/javaws 1065
    update-alternatives --config java
    java -version
    nedit /home/$USERNAME/.bashrc &
    ```

13. Add

    ```
    export JAVA_HOME=/usr/java/jdk1.8.0_91/
    ```

and

```
export PATH=$PATH:/usr/local/java/netbeans-8.1/bin
```

to the `.bashrc`-file

14. Start a new shell with `su - $USERNAME` and type `java -version` to see if the correct version is active
15. Start a new shell with `su - $USERNAME` and type `netbeans &` to start the IDE
16. Perform update inside the IDE if asked for

If problems occur during any `update-alternatives --install` try

```
update-alternatives --install /usr/bin/java java /usr/local/java/jdk1.8.0
    _91/bin/java 1
update-alternatives --install /usr/lib64/browser-plugins/javaplugin.so
    javaplugin /usr/local/java/jdk1.8.0_91/jre/lib/amd64/libnpjp2.so 1 --
    slave /usr/bin/javaws javaws /usr/local/java/jdk1.8.0_91/bin/javaws
update-alternatives --install /usr/bin/javac javac /usr/local/java/jdk1
    .8.0_91/bin/javac 1 --slave /usr/bin/jar jar /usr/local/java/jdk1.8.0
    _91/bin/jar
```

Now you can set the *Java* priorities with:

```
update-alternatives --config java
update-alternatives --config javac
update-alternatives --config javaplugin
```

6.1.2. *NetCDF-C*

The *NetCDF-C*-tool *ncgen* is required to convert the ascii mesh file into the binary format readable by *Peridigm*. *NetCDF-C* should already be installed to use *Peridigm*. If the additions to the `PATH`-variable from subsection 3.4.3 are set, no further actions have to be performed.

6.2. Windows

6.2.1. Java

1. Go to: http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Click on *NetBeans with JDK* for Development Kit and Netbeans or just *Java Platform (JDK)*
3. Perform the installation

6.2.2. *NetCDF-C*

To test the *FETranslator* under Windows it is necessary to have *ncgen* available. *ncgen* is available as part of pre-built *NetCDF-C* libraries. To install the latest release

1. Go to: http://www.unidata.ucar.edu/software/netcdf/
2. Click *Pre-built Windows Binaries for the latest version of NetCDF-C*
3. Go to *Latest Release (NetCDF-C X.Y.Z)*, here NetCDF-C 4.4.0
4. Download the executable matching your system, here netCDF4.4.0-NC4-64.exe
5. Execute the installer
6. Add the `bin` folder of the installation path, here `D:\Programme\netCDF 4.4.0\` to the Windows `PATH`-Variable:
    a) Open the *Windows Control Panel* (Systemsteuerung)
    b) Open *System*
    c) Click *Advanced System Settings* (Erweiterte Systemeinstellungen)
    d) In the *Advanced* tab open *Environment Variables*
    e) Under *User variables for USERNAME* select `PATH`
    f) Click *Edit*
    g) Under *Value of the variable* add the path to the `bin` folder of the *NetCDF-C* installation separated by a semicolon (;), here: `;D:\Programme\netCDF 4.4.0\bin\`
    h) Click *OK* multiple times

Now you can use *ncgen* in a command-window:

```
ncgen.exe -o $OUTPUTFILENAME.g $INPUTFILENAME.g.ascii
```

Bibliography

[1] Michael L. Parks et al. *Peridigm Users Guide v1.0.0*. Tech. Report SAND2012-7800. Sandia report. Albuquerque, New Mexico 87185 and Livermore, California 94550, USA, 2012. URL: http://www.sandia.gov/~djlittl/docs/PeridigmV1.0.0.pdf.

[2] Larry A. Schoof and Victor R. Yarberry. *Exodus II: A Finite Element Data Model*. SAND92-2137, UC-705. Sandia report. Albuquerque, New Mexico 87185 and Livermore, California 94550, USA, 1994.

Appendices

A. Build-scripts for Libraries

In the following sections, the build scripts for the libraries for *Peridigm* are collected. These are Bash-scripts or *CMake*-files.

The scripts are taken from https://peridigm.sandia.gov/ and modified slightly if necessary.

The files are provided with UTF-8 encoding. Please modify to your needs if necessary.

A.1. *Boost*

A.1.1. *Boost* 1.55.0

Open a text editor, copy the following code into a file and save as `install_boost-1.55.0.sh`

Listing A.1: **Install script for *Boost* 1.55.0**

```
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Run the Boost bootstrap script
./bootstrap.sh

# add using mpi to user-config.jam
echo "using mpi ;" >> tools/build/v2/user-config.jam
cp tools/build/v2/user-config.jam ~/

# Compile and install Boost using the Boost's bjam build system
./b2 install --prefix=/usr/local/lib/boost-1.55.0/
```

Alternatively, you can download the script from within this document.

A.1.2. *Boost* 1.60.0

Open a text editor, copy the following code into a file and save as `install_boost-1.60.0.sh`

Listing A.2: **Install script for *Boost* 1.60.0**

```
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Run the Boost bootstrap script
./bootstrap.sh

# add using mpi to project-config.jam
echo "using mpi ;" >> project-config.jam

# Compile and install Boost using the Boost's bjam build system
./b2 install --prefix=/usr/local/lib/boost-1.60.0/
```

Alternatively, you can download the script from within this document.

A.2. *HDF5*

Open a text editor, copy the following code into a file and save as `install_hdf.sh`

Listing A.3: **Install script for *HDF5***

```
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Configure HDF5
./configure --prefix=/usr/local/bin/hdf5-1.8.16/ --enable-parallel

# Make and install HDF5
make -j 4
make install
```

Alternatively, you can download the script from within this document.

A.3. *NetCDF-C*

Open a text editor, copy the following code into a file and save as `install_netcdf.sh`

Listing A.4: **Install script for *NetCDF-C***

```
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Configure NetCDF
./configure --prefix=/usr/local/bin/netcdf-4.4.0/ --disable-netcdf-4 --
    disable-dap

# Make, test, and install NetCDF
make -j 4
make check
make install
```

Alternatively, you can download the script from within this document.

A.4. *Trilinos*

Open an editor, copy the following code into the file and save as `cmake_trilinos.cmake`. The final line marks the path to the *Trilinos* source directory, which is named `$DOWNLOAD_DIR` in the documentation.

Listing A.5: ***CMake* script for *Trilinos***

```
rm -f CMakeCache.txt
rm -rf CMakeFiles/

cmake -D CMAKE_INSTALL_PREFIX:PATH=/usr/local/bin/trilinos-12.4.2/ \
-D MPI_BASE_DIR:PATH="/usr/local/lib/openmpi-1.10.2/" \
-D CMAKE_CXX_FLAGS:STRING="-O2 -std=c++11 -pedantic -ftrapv -Wall -Wno-
   long-long" \
-D CMAKE_BUILD_TYPE:STRING=RELEASE \
-D Trilinos_WARNINGS_AS_ERRORS_FLAGS:STRING="" \
-D Trilinos_ENABLE_ALL_PACKAGES:BOOL=OFF \
-D Trilinos_ENABLE_Teuchos:BOOL=ON \
-D Trilinos_ENABLE_Shards:BOOL=ON \
-D Trilinos_ENABLE_Sacado:BOOL=ON \
-D Trilinos_ENABLE_Epetra:BOOL=ON \
-D Trilinos_ENABLE_EpetraExt:BOOL=ON \
-D Trilinos_ENABLE_Ifpack:BOOL=ON \
-D Trilinos_ENABLE_AztecOO:BOOL=ON \
-D Trilinos_ENABLE_Amesos:BOOL=ON \
-D Trilinos_ENABLE_Anasazi:BOOL=ON \
-D Trilinos_ENABLE_Belos:BOOL=ON \
-D Trilinos_ENABLE_ML:BOOL=ON \
-D Trilinos_ENABLE_Phalanx:BOOL=ON \
-D Trilinos_ENABLE_Intrepid:BOOL=ON \
-D Trilinos_ENABLE_NOX:BOOL=ON \
-D Trilinos_ENABLE_Stratimikos:BOOL=ON \
-D Trilinos_ENABLE_Thyra:BOOL=ON \
-D Trilinos_ENABLE_Rythmos:BOOL=ON \
-D Trilinos_ENABLE_MOOCHO:BOOL=ON \
-D Trilinos_ENABLE_TriKota:BOOL=OFF \
-D Trilinos_ENABLE_Stokhos:BOOL=ON \
-D Trilinos_ENABLE_Zoltan:BOOL=ON \
-D Trilinos_ENABLE_Piro:BOOL=ON \
-D Trilinos_ENABLE_Teko:BOOL=ON \
-D Trilinos_ENABLE_SEACASIoss:BOOL=ON \
-D Trilinos_ENABLE_SEACAS:BOOL=ON \
-D Trilinos_ENABLE_SEACASBlot:BOOL=ON \
-D Trilinos_ENABLE_Pamgen:BOOL=ON \
-D Trilinos_ENABLE_EXAMPLES:BOOL=OFF \
-D Trilinos_ENABLE_TESTS:BOOL=ON \
-D TPL_ENABLE_Matio:BOOL=OFF \
-D TPL_ENABLE_HDF5:BOOL=ON \
-D HDF5_INCLUDE_DIRS:PATH="/usr/local/bin/hdf5-1.8.16/include" \
-D HDF5_LIBRARY_DIRS:PATH="/usr/local/bin/hdf5-1.8.16/lib" \
-D TPL_ENABLE_Netcdf:BOOL=ON \
-D Netcdf_INCLUDE_DIRS:PATH="/usr/local/bin/netcdf-4.4.0/include" \
-D Netcdf_LIBRARY_DIRS:PATH="/usr/local/bin/netcdf-4.4.0/lib" \
-D TPL_ENABLE_MPI:BOOL=ON \
-D TPL_ENABLE_BLAS:BOOL=ON \
-D TPL_ENABLE_LAPACK:BOOL=ON \
-D TPL_ENABLE_Boost:BOOL=ON \
-D Boost_INCLUDE_DIRS:PATH="/usr/local/lib/boost-1.55.0/include" \
-D Boost_LIBRARY_DIRS:PATH="/usr/local/lib/boost-1.55.0/lib" \
-D CMAKE_VERBOSE_MAKEFILE:BOOL=OFF \
-D Trilinos_VERBOSE_CONFIGURE:BOOL=OFF \
/usr/local/src/trilinos-12.4.2-Source/
```

Alternatively, you can download the script from within this document.

A.5. *Peridigm*

A.5.1. *CMake* script for *Peridigm*

Open an editor, copy the following code into the file and save as `cmake_peridigm.cmake`. The final line marks the path to the *Peridigm* source directory, which is named `$DOWNLOAD_DIR` in the documentation.

Listing A.6: ***CMake* script for *Peridigm***

```
rm -f CMakeCache.txt
rm -rf CMakeFiles/

cmake \
-D CMAKE_BUILD_TYPE:STRING=Release \
-D Trilinos_DIR:PATH=/usr/local/bin/trilinos-12.4.2/lib/cmake/Trilinos/ \
-D CMAKE_C_COMPILER:STRING=/usr/local/lib/openmpi-1.10.2/bin/mpicc \
-D CMAKE_CXX_COMPILER:STRING=/usr/local/lib/openmpi-1.10.2/bin/mpicxx \
-D BOOST_ROOT=/usr/local/lib/boost-1.55.0/ \
-D CMAKE_CXX_FLAGS:STRING="-O2 -Wall -std=c++11 -pedantic -Wno-long-long
    -ftrapv -Wno-deprecated" \
/usr/local/src/Peridigm-1.4.1-Source
```

Alternatively, you can download the script from within this document.

A.5.2. Script for cloning *Peridigm* from GitHub and compiling on the STM-Cluster

Listing A.7: **Script for cloning *Peridigm* from GitHub and compiling on the STM-Cluster**

```
#!/bin/bash

#######################################
# Header                              #
#######################################
#
# Install Peridigm from github repository master
#
# Requirements:
#            Allocate exclusive node before using this script:
#            salloc --exclusive
#
# Revisions: 2017-12-22 Martin Raedel <martin.raedel@dlr.de>
#                       Initial draft
#
# Contact:   Martin Raedel,  martin.raedel@dlr.de
#            DLR Composite Structures and Adaptive Systems
#
#                                 __/|__
#                                /_/_/_/
#            www.dlr.de/fa/en      |/ DLR
#
#######################################
# Content                             #
#######################################

#--------------------------------------
# Variables
#--------------------------------------

# The directory where all the magic happens - should not exist in advance
basedir=$HOME'/Documents/Peridigm/20171222EC_/'

# Path from where to clone
githubclonepath='https://github.com/peridigm/peridigm.git'
```

```
# Internal directory names
builddir='build'
srcdir='src'

# File names
file_cmake='cmake_peridigm.cmake'
file_cmake_log='cmake_peridigm.log'
file_make_log='make.log'
file_make_test_log='make_test.log'
file_make_install_log='make_install.log'

file_peridigm_bin='Peridigm'

# Number of CPUs for make
make_cpus = 8

#---------------------------------------
# Script
#---------------------------------------

#-------------------
# Load build environment
#-------------------

echo 'Load build environment'
. /cluster/software/slurm/etc/env.d/mpibuild.sh
. /cluster/software/slurm/etc/env.d/peridigm.sh

#-------------------
# Folder structure
#-------------------

echo 'Create directory structure'
if [ -d ${basedir} ]; then  # Control will enter here if $DIRECTORY doesn
    't exist.
  echo 'Directory '${basedir}' already exists. Exit.'
  exit 0
fi

mkdir ${basedir}
cd ${basedir}
```

```
mkdir ${builddir}
mkdir ${srcdir}
cd ${srcdir}

#------------------
# Clone from GitHub
#------------------

echo 'Clone from GitHub'
git clone ${githubclonepath}
cd ../${builddir}

#------------------
# Create cmake file
#------------------

echo 'Create and execute cmake file'

# Fill file
echo 'rm -f CMakeCache.txt' >> ${file_cmake}
echo 'rm -rf CMakeFiles/' >> ${file_cmake}
echo '' >> ${file_cmake}
echo 'cmake \' >> ${file_cmake}
echo '-D CMAKE_BUILD_TYPE:STRING=Release \' >> ${file_cmake}
echo '-D CMAKE_INSTALL_PREFIX='${basedir}${builddir}' \' >> ${file_cmake}
echo '-D CMAKE_CXX_FLAGS:STRING="-O2 -Wall -std=c++11 -pedantic -Wno-long
    -long -ftrapv -Wno-deprecated" \' >> ${file_cmake}
echo ${basedir}${srcdir}'/peridigm/' >> ${file_cmake}

# Change permissions to make cmake-file executable
chmod u+x ${file_cmake}

# Execute cmake-file
./${file_cmake} > ${file_cmake_log} 2>&1

#------------------
# Make
#------------------

echo 'make'
make -j ${make_cpus} > ${file_make_log} 2>&1
```

```
echo 'make test'
make test > ${file_make_test_log} 2>&1
if grep -q Error ${file_make_test_log}; then
  echo '  make test contains failed tests'
else
  echo '  all tests passed'
fi

echo 'make install'
make install > ${file_make_install_log} 2>&1

#------------------
# Comment
#------------------

cd bin
echo 'Finished - Peridigm executable path':
readlink -f ${file_peridigm_bin}

#---------------------------------------
# Clean
#---------------------------------------
```

Alternatively, you can download the script from within this document.

A.6. Make a script executable

In order to use a script file in the shell for installation you must first make the text file executable. Therefore, open a terminal, change directory to the folder the individual script is located and make the script executable for the user with

```
chmod u+x $SCRIPTNAME.sh
```

A.7. Modifications of `.bashrc`

When an interactive shell that is not a login shell is started, bash reads and executes commands from `~/.bashrc`, if that file exists. You can find the `.bashrc` file in your user home directory `/home/$USER/` with `ls -al`.

The following listings shows a modified `.bashrc` file which includes the exportation of the significant libraries in the `$PATH` and `$LD_LIBRARY_PATH` environment variables. The header is not printed.

Be aware:

- In case you use a 32bit operating system, or in some cases also for 64bit operating system, the lib64-folders must be changed to lib.
- The entries to the `$PATH` and `$LD_LIBRARY_PATH` variables should be added step-by-step **after** the installation of the individual tool. Otherwise, the install scripts might find pre-compiled items and use these instead of creating new binaries with the current settings.

A.7.1. `.bashrc` for *Peridigm* 1.4.1

Listing A.8: **Modified .bashrc-file to set environment variables for *Peridigm* 1.4.1**

```
test -s ~/.alias && . ~/.alias || true

# add default lib paths
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib64

# add cmake
export PATH=$PATH:/usr/local/bin/cmake-3.4.3/bin

# add openMPI
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/
    lib64

# add boost
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/boost-1.55.0/lib
```

```
# add hdf5
export PATH=$PATH:/usr/local/bin/hdf5-1.8.16/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/hdf5-1.8.16/lib64

# add netcdf
export PATH=$PATH:/usr/local/bin/netcdf-4.4.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/netcdf-4.4.0/lib64

# add trilinos
export PATH=$PATH:/usr/local/bin/trilinos-12.4.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/trilinos-12.4.2/
    lib
```

You can download the file from within this document.

A.7.2. `.bashrc` for *Peridigm* 1.5

Listing A.9: **Modified .bashrc-file to set environment variables for *Peridigm* 1.5**

```
test -s ~/.alias && . ~/.alias || true

# add default lib paths
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib64

# add cmake
export PATH=$PATH:/usr/local/bin/cmake-3.5.1/bin

# add openMPI
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/lib

# add boost
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/boost-1.60.0/lib

# add hdf5
export PATH=$PATH:/usr/local/bin/hdf5-1.8.16/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/hdf5-1.10.0/lib64

# add netcdf
export PATH=$PATH:/usr/local/bin/netcdf-4.4.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/netcdf-4.4.0/lib64

# add trilinos
export PATH=$PATH:/usr/local/bin/trilinos-12.6.1/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/trilinos-12.6.1/
    lib
```

You can download the file from within this document.

B. FAQ

The error `g++ internal compiler error killed (program cc1plus)` occures while compiling with `make -j N`. What is that?

- The problem is probably caused by insufficient memory.
- Try typing `free -m` in a terminal while compiling to see the amount of free RAM
- Running `make -j 4` runs lots of process which use more memory. The problem above occurs when your system runs out of memory. In this case rather than the whole system falling over, the operating systems runs a process to score each process on the system. The one that scores the highest gets killed by the operating system to free up memory. If the process that is killed is cc1plus, gcc (perhaps incorrectly) interprets this as the process crashing and hence assumes that it must be a compiler bug. But it isn't really, the problem is the OS killed cc1plus, rather than it crashed.
- If this is the case, you are running out of memory. So run perhaps `make -j 2` instead. This will mean fewer parallel jobs and will mean the compilation will take longer but hopefully will not exhaust your system memory.

When I call `make test` after the *Peridigm* installation all multi-processor-tests fail. What is the problem?

- Make sure you do not execute `make test` as root user. It is not allowed to call `mpirun` as root. If you call `mpirun` as root, the job is cancelled automatically.
- In order to run the tests as a normal user make sure you set the permissions of the *Peridigm* installation folder to allow execution for group and others

Everything works fine, all *Peridigm* tests are passed, but when I call *decomp* I get a `***HDF5 library version mismatched error***` error. Why?

- It seems *decomp* finds another version of the *HDF5* library `libhdf5.so` than the one you use to compile *Trilinos* with.
- This can be caused by installation of other tools via the *YaST2* software management which puts libraries in `/usr/lib` or `/usr/lib64`. These are found in the current `LD_LIBRARY_PATH` before the *HDF5* version installed in this guide.
- Tools which bring their own version of *HDF5* are for example:

Tool/Package	*HDF5* version
octave-forge-netcdf	1.8.15

- ↗ If these packages are installed and you do not use them, uninstall them and the *HDF5* libraries as described in subsection 3.3.2. You do not have to recompile anything. Just try using *decomp* afterwards.
- ↗ In case you really really need the tool with the other *HDF5* there seems to be nothing left to to but to compile *Trilinos* using this version of *HDF5*.

I get errors when using the provided CMake-files, like `[...] command not found`. What can I do?

- ↗ This problem may arise if you import the appended scripts under Windows.
- ↗ The end-of-line character might be changed to `CR-LF` instead of `LF`
- ↗ To check this problem, open *Notepad++* and go to *Edit* → *EOL Conversion* → *Convert to Unix Format (LF)*
- ↗ Save and try using the script again

C. Useful Linux commands

Category	Command	Description
Working with archives		
	`tar xvfj [FILENAME].tar.bz2`	Extract tar.bz2 file
	`tar xvfz [FILENAME].tar.gz`	Extract tar.gz file
	`unzip [FILENAME].zip -d [DESTINATION_FOLDER]`	Extract zip file to folder
	`tar cvfz [ARCHIVENAME].tar.bz2 [FILENAMES]`	Pack files to tar.bz2 file
Files, directories, links		
	`head -n [NUM] [FILENAME]`	Print the first [NUM] lines of [FILENAME]
	`ln -sf source [TARGET]`	Creates a symbolic link
	`ls -al`	List files (visible and hidden) in current directory
	`tail -f [FILENAME]`	Print and follow last line of file
Jobs		
	`watch -n [SECONDS] [COMMAND]`	Repeat command automatically
Memory		
	`df -h`	display the amount of available disk space
	`du -sch *`	display the file and folder sizes of current directory
	`free -m`	display the amount of available RAM
Permissions		

continued . . .

…continued

Category	Command	Description
	`chmod [OPTIONS] mode file1 ...`	Change permissions of files or folders
	`chown [OPTIONS][USER][:[GROUP]] file1 ...`	Change owner and group of file or folder
System information		
	`cat /etc/*-release`	Find out My Linux Distribution Name and Version
	`cat /proc/cpuinfo`	Find out about CPU
	`cat /proc/meminfo`	Find out about RAM
	`lsb_release -a`	Find out about operating system

D. Non-necessary tools & scripts

The installation is described for *openSUSE* for version 42.1.

D.1. *NEdit*

NEdit, the Nirvana editor, is a text editor and source code editor for the X Window System. For the installation of the editor *NEdit* visit:

http://software.opensuse.org/download.html?project=editors&package=nedit

You can either choose the 1-Click-installation or add the repository and install manually. For the latter login to a terminal as `root` and type

```
zypper addrepo http://download.opensuse.org/repositories/editors/
   openSUSE_Leap_42.1/editors.repo
zypper refresh
zypper install nedit
```

D.2. RM-LaTeX

Download the package via the intranet (from within the DLR network or via a VPN connection):

teamsites.dlr.de/rm/latex/SitePages/Homepage.aspx

Follow the instructions given in `/doc/RM-LaTeX-Guide/RM-LaTeX-Guide.pdf`.

D.2.1. Linux

Before using `mktexlsr` set

```
chmod +t texmf/
```

and

```
chmod go+w texmf/
```

Due to some problems in the RM-LaTeX package meaningful use is only possible under Windows. If using with Linux do not use everything related to the package `dlrsecondpage`.

D.2.2. Windows

There should be no additional steps necessary.

E. This document

E.1. Repository

This document is part of the PeriDoc repository. The complete repository can be found at:

https://github.com/PeriDoX/PeriDoX

A tool implementing the *subversion* protocols can be used to access the repository as well as update the files or commit changes using command line options or a graphical user interface.

Following is a list of tested possible tools:

Operating system	Tool	Homepage
Windows	*TortoiseSVN*	https://tortoisesvn.net
Linux with KDE	*KDESvn*	http://kdesvn.alwins-world.de/

E.2. Typesetting

This document is typeset using the RM-LaTeX package, see section D.2.

The compilation is performed with `pdflatex` with the following options:

```
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %source  --
    extra-mem-top=60000000
```

E.3. Structure

This document is part of the PeriDoc repository. The following files are used from the PeriDoc_Common folder:

THIS PART IS TEMPORARILY OMITTED DUE TO A BUG IN THE PGFPLOTS & FOREST FPU LIBRARY. CHECK BACK AFTER THE NEXT MIKTEX UPDATE OF THE ACCORDING LIBRARIES. SEE: http://tex.stackexchange.com/questions/328972/presence-of-pgfplots-package-breaks-forest-environment-w-folder-option-en

BSD Documentation License