VM2 3.2

Quick Start: Installation

VeraChem LLC

VeraChem has been issued a patent **(USPTO Patent No. 8,140,268)** for the VM2 method.

Contact:

For information regarding VM2 software package licensing contact VeraChem LLC at sales@verachem.com

For technical support contact VeraChem LLC at support@verachem.com

For general enquiries contact VeraChem LLC at info@verachem.com

VM2 3.2 Package Installation

1 Obtaining the VM2 package, and package choices

1.1 Commercial licensing

To obtain the VM2 package for commercial use contact sales@verachem.com.

Commercial licensing available includes one and two year licenses. Multi-site licenses are available.

1.2 Trial license

To obtain a trial license for the VM2 package contact sales@verachem.com

Free three-month licenses are available for users to trial the fully functional parallel processor enabled VM2 package.

1.3 Academic licensing

To obtain the VM2 package for academic use contact info@verachem.com

Provide your name, position, and institution, and outline in general terms your intended use of the software.

1.4 Package choices

The following table shows the various packages available and their capabilities:

VM2 Package	Parallelization	Maximum atom count	
		Whole Molecular System	Mobile Atoms
Host+Ligand	Serial, MPI	-	1000
Protein+ligand, Full suite	Serial, MPI MPI+OpenMP, MPI+CUDA, MPI+OpenMP+CUDA	10000	3000
Full suite - large count version	Serial, MPI MPI+OpenMP, MPI+CUDA, MPI+OpenMP+CUDA	40000	5000

Full suite: host+ligand and protein+ligand functionality

2 Operating systems and hardware

The VM2 package currently runs on Linux desktops, workstations, and clusters. It can also take advantage of GPU acceleration.

2.1 Linux workstations

The serial, MPI, and MPI-OpenMP VM2 packages can be installed on a workstation with two gigabytes of RAM per compute core or more. It is recommended that a minimum of 8 CPU cores is available for computation. The Red Hat Enterprise Linux 7, 8, and 9 operating system is supported and several other Linux distributions are known to work, including Ubuntu versions 18-24.

2.2 Linux desktops

These VM2 packages can also run on commodity desktop PCs that have adequate memory, though recommended use would be for smaller calculations (ligand, hosts, host-ligand complexes), with dedicated workstations more suitable for the more computationally demanding protein and protein-ligand complex calculations.

2.3 Linux clusters

The MPI and MPI-OpenMP VM2 packages can run across clusters of workstations (or clusters of commodity machines in the case of Beowulf clusters). Given that the MPI parallelization schemes are not communication bound, slower Ethernet interconnects are adequate, though parallel MPI also works with the faster InfiniBand interconnects if present.

2.4 Linux workstations and clusters with NVIDIA GPU acceleration

The MPI-CUDA and MPI-OpenMP-CUDA VM2 packages can take advantage of NVIDIA GPUs (Fermi and Kepler architectures) for acceleration of parts of its algorithm. This includes use of multi-GPU workstations and clusters of workstations each with multiple GPUs.

2.5 OSX

VM2 is not currently available for OSX.

2.6 MS Windows

VM2 is not currently available for MS Windows.

3 Installation procedure

3.1 Download the VM2 package

After downloading the VM2 package

`vcCompChem_3.2_*.tar.bz2`

uncompress and untar it in in the location of your choice, e.g.

```
tar xvf vcCompChem_3.2_*.tar.bz2
```

This will create the directory

`./vcCompChem_3.2`

in the directory you are currently in.

3.2 License files

Copy your license files named `vm2_license.LIC` and `vm2FreeEnergy.lic` into the /exe directory:

`./vcCompChem_3.2/exe`

3.3 Environment variables for installation

These installation instructions assume the bash shell is being used. Place the following shell commands and environment variable settings in your **.bashrc** file, which should then be sourced prior to running the installation script. You may use another default shell as you wish, as long as the equivalent command/same environment variables are set.

Modify the environment variable **VCHOME** to reflect the location of the directory resulting from the tar file extraction above.

```
ulimit -s unlimited
export VCHOME=/home/<user_name>/vcCompChem_3.2
export VM2HOME=$VCHOME
export VCPYTHON=$VCHOME/exe/vc_python
export VM2PYTHON=$VCPYTHON
```

Per-user python site-packages, data, and scripts directories (see https://peps.python.org/pep-0370/) can conflict with the correct operation of VM2 as well as other third party environments you may set up using Conda (such as AmberTools) This is because they circumvent the isolation of virtual environments. Per-user installations are usually located in:

```
~/.local/lib/python2.6/site-packages
~/.local/lib/python2.6
```

If you have such installations you will need to disable python's use of them prior to running VM2.
You can do this by setting the following environment variable:

```
export PYTHONNOUSERSITE=1
```

A key indication that you have a per-user local conflict are log files containing python tracebacks that reference packages installed in the .local directories listed above. During correct operation of VM2 no packages in those directories are used and therefore they should not appear in any logs.

3.4 Requirements for installation

Red Hat Enterprise Linux 9 (RHEL9)

- VM2 3.0 fully supports RHEL9
- It may be necessary to install packages such as tcsh and g77 before proceeding
- Also, zlib-devel.x86_64 might be required to compile python and gcc-c++.x86_64 for the extensions

To check for already installed libraries:

```
yum list zlib-devel
yum list gcc
yum list g++
```

If uncertain, carry out the following installations, which will ensure the necessary libraries are present:

```
sudo yum install zlib-devel
sudo yum groupinstall "Development Tools"
```

CentOS/Oracle Linux/Other Versions of RHEL

- VM2 3.0 has been routinely installed and run on CentOS 7, Oracle Linux 8 systems, and RHEL 8.
- The installed libraries requirements are the same as for RHEL8 (see above)

Ubuntu

- Ubuntu versions 18-24 are known to work.
- The following packages may be required to compile python and the included extensions: bzip2, zlib1g-dev, vflib3, build-essential.

To check which packages are already installed on your system:

```
apt list --installed
```

If uncertain, carry out the following installations, which will ensure the necessary libraries are present:

```
apt install bzip2
apt install zlib1g-dev
apt install vflib3
apt install build-essential
```

In some cases it may be necessary to upgrade the apt software itself:

```
apt update
apt upgrade
```

This is usually necessary only for entirely new installations of the operating system.

3.5 Installation script

Carry out the following sequence of commands to complete the installation:

```
cd vcCompChem_3.2
cd build
./install_vcCompChem.sh
```

The installation will take several minutes. At the conclusion of the installation steps ***an automated test set will run***, which will also take several minutes to complete. Results of these automated tests will be found in `vcCompChem_3.0/build` directory:

```
test_install_<date and time stamp>.log
```

(e.g. test_install_2024_09_28_17:39:11.log)

If any of the tests fail, please check that the **VCHOME** and/or **VCPYTHON** environment variable(s) are set correctly (see **3.3** above). Check this by typing:

```
echo $VCHOME

echo $VCPYTHON
```

Relevant information may also be found in the log files generated in `vcCompChem_3.0/build/` directory:

```
python_install.log
extensions_install.log
vc_install.log
```

In particular, missing prerequisites (see **3.4** above) can result in installation and test failures with errors noted in those logs. For example, if the build-essential package is not installed on Ubuntu you will see the following message in extensions_install.log:

```
gcc: fatal error: cannot execute ‘cc1plus’: execvp: No such file or directory
compilation terminated.
error: command 'gcc' failed with exit status 1
```

Please contact VeraChem for support at support@verachem.com if you have any questions or difficulties with installation or encounter any test failures.

4 Installed VM2 package structure

The installed VM2 package directories of interest are:

```
$VCHOME/documentation
$VCHOME/exe             <--- VM2 workflow, helper tools, and calc engine exes
$VCHOME/lib
$VCHOME/tests
$VCHOME/tutorials
$VCHOME/examples
```

4.1 Documentation

The `$VCHOME/documentation` directory contains useful guides to the VM2 package, its functionality and tools:

```
./getting_started_with_VM2_3.2.pdf
./VM2_3.0_quick_start_installation.pdf
```

4.2 VM2 workflow executable

The VM2 workflow executable `VM2.pyc` is located in the `$VCHOME/exe` directory.

Also present is a useful workflow script generator tool:

`generate_vm2_workflow_scripts.sh`

4.3 VM2 workflow helper tool executables

A set of workflow helper software tools are present in the `$VCHOME/exe` directory. The most commonly used of these are:

VCharge: assignment of partial atomic charges

VCharge provides fast, easy access to accurate partial charges for virtually any drug-like compound. As input it requires an sdf/mol file. In addition to the Linux command line version, `Vcharge.pyc`, supplied with this package, a GUI version is available.

VConf: 2D to 3D and small molecule conformational search

VConf is a standalone conformational search application, which processes an SD file of drug-like compounds containing an initial 2D or 3D conformation of each molecule. In addition to the Linux command line version, `Vconf.pyc`, supplied with this package, a GUI version is available.

VMap: superimpose ligands on a reference molecule

VMap identifies the maximum common substructure (MCS) shared by the reference molecule and each of a series of ligands, and then minimizes the RMSD of the MCS by dihedral rotations. The reference format can be PDB (e.g. co-xtal ligand), SDF, MOL, or CRD. The ligand series is supplied as an SDF. In addition to the Linux command line version, `Vmap.pyc`, supplied with this package, a GUI version is available.

prm2top: AMBER formatted input data files to VM2 input data files

This tool `prm2top.pyc` given AMBER formatted prmtop and inpcrd files, outputs VM2 input coordinate, topology, and parameter data files i.e. the crd, top, and mol files required to run the VM2 calculation engine.

4.4 VM2 calculation engine executables

The VM2 calculation engine executables present in the `$VCHOME/exe` directory depend on the licensing level -- see the Package choices section above.

Licensing Level	Executables Present
Host + ligand	VC_CompChemPackage_serial.exe VC_CompChemPackage_mpi.exe
Protein + ligand, Full Suite, Full suite - large	VC_CompChemPackage_serial.exe VC_CompChemPackage_mpi.exe VC_CompChemPackage_mpi_openmp.exe VC_CompChemPackage_mpi_openmp_cuda.exe

4.5 Additional 3rd-party-format converter executables

mmo2top: Schrodinger mmo file to VM2 input data files

This tool `mmo2top.pyc` given a Schrodinger .mmo file, outputs VM2 input coordinate, topology, and parameter data files i.e. crd, top, and mol files.

psf2top: CHARMM formatted input data files to VM2 input data files

This tool `psf2top.pyc` given a CHARMM formatted psf file and mol/sd file, outputs VM2 input coordinate, topology, and parameter data files i.e. crd, top, and mol files. This tool requires that Discovery Studio Visualizer be installed and the environment variable **VCDSPATH** be set to the location of the CHARMm forcefield files in the particular installation of Discovery Studio Visualizer e.g. : `export VCDSPATH=$HOME/DiscoveryStudio_2016/share/forcefield/CHARMm`

4.6 Supplied libraries

The following run time libraries are supplied in `$VCHOME/lib` :

```
/intel      <--- required Intel math, linear algebra, and parallel processing libraries (MPI,
OpenMP)
/cuda       <--- required Nvidia CUDA libraries for running on GPUs
/magma      <--- required linear algebra libraries for running on GPUs
```

4.7 Validation tests

The validation tests included with the package are found in the directory `$VCHOME/tests`, which has the following high level scripts to invoke the tests:

```
./run_install_tests.bsh
./run_tools_tests.bsh
./run_vm2_tests.bsh
```

and the following directories containing lower level run scripts, and the tests themselves:

```
/scripts          <--- lower level scripts to run the tests
/tools            <--- tests for workflow helper tools
/vm2              <--- VM2 calculation engine tests
/workflow         <--- VM2 workflow test
```

4.8 Tutorials, example workflows, and VM2 setups for the Schindler and Gapsys Free Energy Benchmarks

Detailed tutorials that take you through the step-by-step set up of protein-ligand series VM2 binding free energy calculation workflows from scratch are available in the `$VCHOME/tutorials` directory:

```
$VCHOME/tutorials/protein_ligand/pl_tutorial_1
$VCHOME/tutorials/protein_ligand/pl_tutorial_2
```

Tutorial 1 is an example of a complete end-to-end automated workflow for HIV-1 Protease and 38 ligands, where the ligands are supplied as 2D structures. In this tutorial the protein and ligand force field parameterization steps are automated, as well as the ligand initial placements in the binding site.

Tutorial 2 is an example of a workflow where the user has previously carried out protein and ligand setup/parameterization and ligand placement within the binding site (using third party or in-house tools), and supplies parameter and other files e.g. prmtop, inpcrd, etc. to the workflow. The system is c-Met Kinase and 24 ligands.

It is highly recommended that you work through at least one of them before working on your own system.

The `$VCHOME/examples` directory includes an additional set of example VM2 workflows for calculation of protein-ligand series and host-guest series binding free energies.The protein-ligand workflow example directory structure is:

```
/examples/vm2/workflow/protein_ligand/vm2pkg_pl_cmet_workflows
/examples/vm2/workflow/protein_ligand/vm2pkg_pl_CoV-2_mp_workflows
/examples/vm2/workflow/protein_ligand/vm2pkg_pl_hivp_umass38_workflows
/examples/vm2/workflow/protein_ligand/vm2pkg_pl_hivp_umass5_workflows
```

The host-guest workflow example directory structure is:

```
/examples/vm2/workflow/host_guest/cb7_gilson_set
/examples/vm2/workflow/host_guest/cd_gilson_set
/examples/vm2/workflow/host_guest/cd_mobley_sets
/examples/vm2/workflow/host_guest/Sampl3
/examples/vm2/workflow/host_guest/Sampl4
/examples/vm2/workflow/host_guest/Sampl5
/examples/vm2/workflow/host_guest/Sampl6
```

Finally, the supporting information for (Gilson *et al.*, Rapid, Accurate, Ranking of Protein–Ligand Binding Affinities with VM2, the Second-Generation Mining Minima Method. *J. Chem. Theory Comput.* **2024**, *20* (14), 6328–6340. https://doi.org/10.1021/acs.jctc.4c00407.) Contains complete ready-to-run setups, with several run setting variations, for all of the protein-ligand systems in the Schindler (Schindler *et al.* . *J. Chem. Inf. Model.* **2020**, *60* (11), 5457–5474) and Gapsys (Gapsys *et al.* Chem. Sci. **2020**, *11* (4), 1140–1152) free energy benchmarks. This includes 20 systems and a total of 536 ligands. It can be downloaded at: https://zenodo.org/records/11660114

5 Running calculations

5.1 Environment variables

The following environment variables must be set before running a calculation.They can either be set in the user's **.bashrc** or, preferably, within a script used to launch the calculation. The actual values of OMP_NUM_THREADS and MKL_NUM_THREADS will depend on the type of parallel run being requested.

Note that for both the validation tests and the tutorials/workflow examples, as well as any runs that are set up and carried out with the workflow, the required environment variables are set for the user within the run scripts, so they do not have to be added to the user's **.bashrc** before running. The variables are presented here for informational purposes.

```
```

```
ulimit -s unlimited

INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

CUDA_LIBS=$VCHOME/lib/cuda:$VCHOME/lib/magma

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS:$CUDA_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m
```

Since other software besides VM2 may depend on existing MPI and CUDA configurations, care should be taken when setting the variables to ensure that they only affect the environment in which VM2 software is being run.

5.2 AmberTools

AmberTools is not required for installation of the VM2 package, and the VM2 calculation engine is solely responsible for calculations of free energies. AmberTools does, however, provide a means for receptor and ligand structure preparation, atom typing, and forcefield parameter assignment. Some of the functionality in some of the tests, therefore, is dependent on AmberTools. In addition, the VM2 workflow can take advantage of AmberTools preparation, typing, and parameter assignment tools, so most of the workflow examples are dependent on it at the setup stage. Therefore, it is recommended that AmberTools be installed using Conda and the resulting environment be activated prior to running VM2 calculations. We recommend installing AmberTools via conda (use of the miniforge distribution is recommended) and activating the resulting environment prior to running VM2. If access to OpenFF (including the AM1BCC-GNN partial charge method) is also desired, it is possible to install a single environment containing both AmberTools and OpenFF (see below). Instructions for obtaining and installing AmberTools alone can be found here.

While versions up to 23 are available via conda-forge, versions 24 and above are only available on the special `ambertools-dac` channel (as of the date of this release). AmberTools installations from `ambertools-dac` include scripts that run when the environment is activated. Among other things, these scripts set the environment variable `PYTHONPATH`, which interferes with the VM2 workflow by overriding normal Python path resolution. If using AmberTools version 24 or above, you must unset this variable before running VM2:

```
unset PYTHONPATH
```

This does not appear to affect any of the AmberTools functions used by VM2, because the `PYTHONPATH` set by the AmberTools activation script is already present in the Python system path. The explicit setting of `PYTHONPATH` is therefore unnecessary.

In addition, the *reduce* module was removed in AmberTools version 26, so it is only available with AmberTools version 25 or earlier.

OpenFF

VM2 can also use the OpenFF toolkit to assign OpenFF parameters to ligands. This requires the `openff-toolkit` package and its associated dependencies to be installed and available. It is recommended that conda be used and the resulting environment activated when running VM2. By default, installation of `openff-toolkit` will also install AmberTools, so a single environment can cover both the AmberTools and OpenFF requirements. Installation instructions for OpenFF can be found here.

6 Running the validation tests

As described above, a set of validation tests is available in the `vcCompChem_3.0/tests` directory, and a subset of these tests is run automatically after installation. These tests are a basic confirmation of installation. It is recommended that the user ***run all the tests*** appropriate to their intended use of the package (e.g., hardware configurations) to confirm correct installation.

To run the quick set of installation tests only again use:

```
./run_install_tests.py
```

The following two shell scripts automate the full set of tests:

```
./run_tools_tests.bsh

./run_vm2_tests.bsh
```

These scripts will run the entire test suite, separated into VM2 helper tools/workflow tests and VM2 calculation engine tests, and validate the results. The quick set installation tests should take `1-2 minutes`, the VM2 helper tools/workflow tests approximately `30 minutes`, and the VM2 tests around `45 minutes`.

6.1 Helper tools validation tests

The supplied helper tool validation tests check that file format conversions for AmberTools and Maestro/Macromodel based system setups are functioning correctly. They also test that ligand preparation, atom typing, and forcefield parameter assignment via interface of the VM2 package with AmberTools is functioning. The VeraChem tools Vcharge, Vconf, Vrms, and Vfilter are also tested.

6.2 VM2 workflow validation test

This test invokes the Step 1 of the VM2 workflow for automated protein-ligand series binding free energy calculation. The test takes as input pre-generated prmtop/inpcrd files so is not dependent on an AmberTools installation to run.

6.3 VM2 calculation engine validation tests

The tests for the VM2 calculation engine are located in vcCompChem_3.0/tests/vm2 . The test mpi_4 is run automatically after installation.

```
/mpi_16
/mpi_4
/mpi_8
/mpi_cuda
/mpi_openmp_8_2
/mpi_openmp_8_4
/mpi_openmp_cuda
```

Each test is named for a different configuration of mpi, openmp, and cuda. Most tests include scripts for use with PBS/Torque and when running interactively. The PBS scripts will need to be modified to match your computing environment, queue names, run time limits, etc.

Example output is provided in the reference subdirectory of each test. If you open either **.out** file, the time required for the test on our hardware will be found at the bottom of the file.

7 Running workflow examples

Once the VM2 package is successfully installed, and an Ambertools installed Conda environment activated, the workflow examples may be run as a more complete test of the installation and the run environment.

For an example of a protein-ligand series (HIV-1 protease + 5 ligands) workflow where there is a co-crystalized ligand available with the same scaffold as the ligand series, see the directories:

```
$VCHOME/examples/vm2/workflow/protein_ligand/vm2pkg_pl_hivp_umass5_workflows/vm2_coxtal_exampl
e

$VCHOME/examples/vm2/workflow/protein_ligand/vm2pkg_pl_hivp_umass5_workflows/rawdata_hivp_umas
s5_ad81template
```

Be sure to update the resource manager related variables in the in the `set_vc_workflow_control_vars.sh` file to match your own computing environment:

```
export QUEUETYPE='slurm'
export QUEUENAME='default'
```

As shipped, the examples are configured to use the SLURM resource manager and submit to whatever the default queue is for your environment. PBS is also supported (change QUEUETYPE to 'pbs') and if you do not have access to either SLURM or PBS you can select to run on the local machine, one calculation at a time, by changing QUEUETYPE to 'bsh'.

The examples are all set to use 12 cores for each calculation. You can change this by changing the following variables in the `set_vc_workflow_control_vars.sh`

```
export NUMMPIPROCS='12'
...
export NUMLIGMPIPROCS='12'
```

The first controls the number of cores used for calculations involving the protein (protein alone and complexes) and the second controls the number used for ligand alone calculations. Generally it is recommended that a minimum of 8 be used for good performance.

8 Tutorials and Setting up Systems

Two methods are provided to assist with setting up systems for VM2 calculations. The first uses pre-configured run templates (found in `$VCHOME/run_templates`) that match common use cases, such as computing binding energies for already-docked ligands or for congeneric ligand series that are not yet docked. Using a run template involves copying the appropriate template and supplying a few files describing your receptor and ligands, typically a PDB file for the receptor and MOL/SDF files for the ligands. In most cases, run templates are the quickest and simplest way to set up a new calculation. For cases that do not match any of the provided templates, the VM2 workflow script generator may be used. This text-based tool guides you through a series of questions to help configure your system, and offers greater flexibility, but may require more familiarity with VM2 than the run template approach.

Detailed step-by-step tutorials for both the run template and script generator approaches are provided in the `$VCHOME/tutorials` directory, covering the full setup of protein–ligand binding free energy calculations:

```
$VCHOME/tutorials/protein_ligand/run_templates/
    pl_run_template_1_predocked_ligands_tutorial/
    pl_run_template_2_congeneric_ligands_not_predocked_tutorial/

$VCHOME/tutorials/protein_ligand/script_generator/
    pl_tutorial_1/
    pl_tutorial_2/
```

`pl_tutorial_1` and `pl_run_template_2` are examples of setting up a complete end-to-end automated workflow for HIV-1 Protease and 38 ligands, where the ligands are supplied as 2D structures. In this tutorial the protein and ligand force field parameterization steps are automated, as well as the ligand initial placements in the binding site.

`pl_tutorial_2` and `pl_run_template_1` are examples of a workflow situation where the user has previously carried out ligand placement within the binding site (via docking or other methods using third party or in-house tools). The system is c-Met Kinase and 24 ligands.

It is highly recommended that you work through at least one of them before working on your own system.