MAX PLANCK
COMPUTING & DATA FACILITY
MPCDF

Bits & Bytes

No.221, April 2026

Computer Bulletin of the Max Planck Computing and Data Facility (MPCDF)*
https://docs.mpcdf.mpg.de/bnb

High-performance Computing

Resource limits at login nodes of the HPC-Systems

In order to maintain the responsiveness of the login nodes on the HPC machines, per-user resource limits were introduced on *Raven*[1] and also on *Viper*[2] in late 2024. Over time, even stricter limits had to be enforced, actually.

The following tables summarize the current limits for raven and viper, respectively.

	raven[01-02]i	raven[03-04]i
cores	2	6
memory	50 GB	100 GB
tasks	768	1536

	viper[01-02]i	viper[03-06]i	viper[11-12]i	viper13i
cores	2	6	2	6
memory	50 GB	100 GB	50 GB	100 GB
tasks	768	1536	768	1536

As a consequence, especially because of the limitation of tasks (which actually is the sum of Unix processes and threads per user), some programmes might fail with

```
fork: retry: Resource temporarily unavailable
```

In such cases, please double check your process list for stale sessions that might have not terminated properly (`ps auxH`), and clean those up using standard tools like `kill`.

Furthermore, we'd encourage users to explicitly choose a login node based on the above table, and no longer rely on our legacy DNS aliases `raven` or `raven-i` (likewise for *viper*). These legacy aliases will be withdrawn in late 2026.

In case you require more powerful interactive sessions to analyze data on *Raven* and *Viper*, we strongly recommend to use our RVS service at *Raven* and *Robin*. The latter meanwhile has access to the filesystems `/raven/ptmp`, `/viper/ptmp1` (*Viper-CPU*), `/viper/ptmp2` (*Viper-GPU*) and `/nexus/posix0`.

For further insights on potential impacts on parallel builds and GitLab CI jobs on the login nodes, please also refer to our initial article on this topic in issue 217 of Bits&Bytes (December 2024)[3].

Christian Guggenberger

Software News

AMD software

Version 7.2.1 of the ROCm software stack has been installed on *Viper-GPU*. The corresponding module is called `rocm/7.2`.

On *Viper-CPU*, a new OpenMPI module has been installed for the AOCC compiler. Users can now compile Fortran codes with MPI using AOCC Flang by loading the modules `aocc/5.1 openmpi/5.0`. The MPI wrapper scripts follow the standard naming scheme, that is, `mpif90` for Fortran. Also modules providing HDF5 and NetCDF are available for this software stack: `hdf5-mpi/1.14.1` and `netcdf-mpi/4.9.2`.

Tobias Melson

Nvidia HPC SDK

A new module `nvhpcsdk/26` has been started with the Nvidia HPC SDK 26.1. Minor version updates will follow during the year without changing the module name. Update to 26.3 has already been done. The previous `nvhpcsdk/25` is now frozen on version 25.11. Be aware

*Editors: Dr. Renate Dohmen & Dr. Markus Rampp, MPCDF
[1]https://docs.mpcdf.mpg.de/doc/computing/raven-user-guide.html#resource-limits
[2]https://docs.mpcdf.mpg.de/doc/computing/viper-user-guide.html#resource-limits
[3]https://docs.mpcdf.mpg.de/bnb/217.html#resource-limits-on-the-hpc-machines

that the only CUDA version installed with `nvhpcsdk/26` is CUDA 13.1. It can be explicitly loaded with the module `cuda/13.1-nvhpcsdk_26` to set the CUDA relevant paths. With the change to CUDA 13+, some deprecations and API changes have been done. For further details, refer to the CUDA 13 release notes[4].

Tilman Dannert

New ELPA version 2026.02

The latest ELPA eigensolver library[5] release (version 2026.02.001) brings further performance improvements for GPU-based computations. In particular, ELPA 1-stage GPU tridiagonalization and backtransformation are now both ~10% faster, which leads to overall ~10% speedup for the standard eigenproblems.

As announced in the previous issue 220 of Bits&Bytes[6], the bugfix number is now dropped from the ELPA module name. For example, the new release is available as `module load elpa/mpi/standard/gpu/2026.02`.

In addition, the 2025.06.001 module has received a bugfix update, adding support for multi-architecture Nvidia GPU builds and fixing a bug in the tridiagonal solver. This and all previous versions are still available with their full version numbers, e.g., `module load elpa/mpi/standard/gpu/2025.06.001`. All available ELPA modules can be queried with `find-module elpa`.

Petr Karpov, Tobias Melson, Andreas Marek

Fortran support added to structured diff and merge tools via incremental parsing

Difftastic[7] and Mergiraf[8] are syntax-aware diff and merge tools that operate on concrete syntax trees instead of raw text. Difftastic computes structural diffs by parsing source code with Tree-sitter[9], an incremental parsing library that generates concrete syntax trees for many programming languages. Mergiraf builds on the same Tree-sitter infrastructure to perform structured, language-aware merges with improved conflict resolution. Tree-sitter provides fast, error-tolerant parsers and a uniform AST interface, enabling both tools to reason about code structure rather than line-based changes. Therefore the tools can align changes based on actual language structure (functions, expressions, blocks) rather than lines chosen by complicated algorithms, making diffs far more accurate and readable as shown in the screenshot in Figure 1 of Difftastic displaying differences between two Fortran files. Changes are displayed side-by-side by default, because they are no longer line-based. Note how changes are resolved almost to character level and embellished with basic syntax highlighting in bold and italic.

This structure awareness also enables smarter merges that reduce spurious conflicts and preserve intent, unlike traditional line-based tools that often misinterpret reformatting or code movement as substantive changes. Difftastic and Mergiraf can be used directly in Git as diff and merge drivers, respectively.

Recently, Fortran language support has been added by MPCDF to both Difftastic[10] and Mergiraf[11] on their respective development branches via a Tree-sitter Fortran grammar[12].

Henri Menke

[4] https://docs.nvidia.com/cuda/archive/13.0.0/cuda-toolkit-release-notes/index.html
[5] https://elpa.mpcdf.mpg.de
[6] https://docs.mpcdf.mpg.de/bnb/220.html#new-elpa-module-version-scheme
[7] https://difftastic.wilfred.me.uk/
[8] https://mergiraf.org/
[9] https://tree-sitter.github.io/tree-sitter/
[10] https://github.com/Wilfred/difftastic/pull/951
[11] https://codeberg.org/mergiraf/mergiraf/pulls/717
[12] https://github.com/stadelmanma/tree-sitter-fortran

```
sample_files/fortran_2.f90 --- 1/2 --- Fortran
 1 ! A simple Fortran module for numerical integration
 2 module integration
 3     implicit none
 4
 5 contains
 6
 7     ! Trapezoidal rule integration
 8     function trapezoid_f(f, a, b, n) result(integral)
 9         implicit none
10         real, external :: f
11         real, intent(in) :: a, b
12         integer, intent(in) :: n
13         real :: integral
14
15         real :: h, x
16         integer :: i
17
18         h = (b - a) / real(n)
19         integral = 0.5 * (f(a) + f(b))
20
21         do i = 1, n-1
22             x = a + i * h

sample_files/fortran_2.f90 --- 2/2 --- Fortran
26         integral = integral * h
27     end function trapezoid_f
28
29     ! Simpson's rule integration
30     function simpson_f(f, a, b, n) result(integral)
31         implicit none
32         real, external :: f
33         real, intent(in) :: a, b
34         integer, intent(in) :: n
35         real :: integral
36
37         real :: h, x
38         integer :: i
39
40         if (mod(n, 2) /= 0) then
41             print *, "Error: n must be even for Simpson's rule"
42             integral = 0.0
43             return
44         end if
45
46         h = (b - a) / real(n)
47         integral = f(a) + f(b)
48
49         do i = 1, n-1
50             x = a + i * h
51             if (mod(i, 2) == 0) then
52                 integral = integral + 2.0 * f(x)
53             else
54                 integral = integral + 4.0 * f(x)
55             end if
56         end do
57
58         integral = integral * h / 3.0
59     end function simpson_f
60
61 end module integration
62
63 program main
64     use integration
65     implicit none
66     real :: a, b, result_trapezoid, result_simpson
67
68     a = 0.0
69     b = 3.14159
70
71     result_trapezoid = trapezoid_f(sin, a, b, 1000)
72     result_simpson = simpson_f(sin, a, b, 1000)
73
74     print *, "Trapezoid rule result: ", result_trapezoid
75     print *, "Simpson's rule result: ", result_simpson
76
77 end program main
```

```
 1 ! A Fortran module for numerical integration
 2 ! Added more integration methods and error handling
 3 module integration
 4     use, intrinsic :: iso_fortran_env, only: dp => real64
 5     implicit none
 6     private
 7     public :: trapezoid_f, simpson_f, gaussian_f
 8
 9 contains
10
11     ! Trapezoidal rule integration with double precision
12     function trapezoid_f(f, a, b, n) result(integral)
13         implicit none
14         real(dp), external :: f
15         real(dp), intent(in) :: a, b
16         integer, intent(in) :: n
17         real(dp) :: integral
18
19         real(dp) :: h, x
20         integer :: i
21
22         if (n < 1) then
23             print *, "Error: n must be positive"
24             integral = 0.0_dp
25             return
26         end if
27
28         h = (b - a) / real(n, kind=dp)
29         integral = 0.5_dp * (f(a) + f(b))
30
31         do i = 1, n-1
32             x = a + i * h
36         integral = integral * h
37     end function trapezoid_f
38
39     ! Simpson's rule integration with double precision
40     function simpson_f(f, a, b, n) result(integral)
41         implicit none
42         real(dp), external :: f
43         real(dp), intent(in) :: a, b
44         integer, intent(in) :: n
45         real(dp) :: integral
46
47         real(dp) :: h, x
48         integer :: i
49
50         if (mod(n, 2) /= 0) then
51             print *, "Error: n must be even for Simpson's rule"
52             integral = 0.0_dp
53             return
54         end if
55
56         h = (b - a) / real(n, kind=dp)
57         integral = f(a) + f(b)
58
59         do i = 1, n-1
60             x = a + i * h
61             if (mod(i, 2) == 0) then
62                 integral = integral + 2.0_dp * f(x)
63             else
64                 integral = integral + 4.0_dp * f(x)
65             end if
66         end do
67
68         integral = integral * h / 3.0_dp
69     end function simpson_f
70
71     ! Gaussian quadrature integration (3-point)
72     function gaussian_f(f, a, b) result(integral)
73         implicit none
74         real(dp), external :: f
75         real(dp), intent(in) :: a, b
76         real(dp) :: integral
77
78         real(dp) :: x(3), w(3), midpoint, half_width
79
80         ! 3-point Gaussian quadrature weights and points
81         x = [ -0.7745966692414834_dp, 0.0_dp, 0.7745966692414834_dp ]
82         w = [ 0.5555555555555556_dp, 0.8888888888888888_dp, 0.5555555555555556_dp ]
83
84         midpoint = 0.5_dp * (a + b)
85         half_width = 0.5_dp * (b - a)
86
87         integral = half_width * (w(1) * f(midpoint + half_width * x(1)) + &
88                                  w(2) * f(midpoint + half_width * x(2)) + &
89                                  w(3) * f(midpoint + half_width * x(3)))
90     end function gaussian_f
91
92 end module integration
93
94 program main
95     use integration
96     use, intrinsic :: iso_c_binding, only: c_double
97     implicit none
98     real(dp) :: a, b, result_trapezoid, result_simpson, result_gaussian
99     integer :: n
100
101     a = 0.0_dp
102     b = acos(0.0_dp) * 2.0_dp  ! 2*pi
103     n = 1000
104
105     result_trapezoid = trapezoid_f(sin_dp, a, b, n)
106     result_simpson = simpson_f(sin_dp, a, b, n)
107     result_gaussian = gaussian_f(sin_dp, a, b)
108
109     print *, "Integration bounds: [", a, ", ", b, "]"
110     print *, "Number of intervals: ", n
111     print *, "Trapezoid rule result: ", result_trapezoid
112     print *, "Simpson's rule result: ", result_simpson
113     print *, "Gaussian quadrature result: ", result_gaussian
114
115 contains
116
117     ! Double precision sine function
118     function sin_dp(x) result(y)
119         real(dp), intent(in) :: x
120         real(dp) :: y
121         y = sin(x)
122     end function sin_dp
123
124 end program main
```

Figure 1: A screenshot of Difftastic displaying differences between two Fortran files.

DataShare: Public link passwords

Since the migration of the DataShare service to Nextcloud last year, creating password protected public link has been a two step process: Initially the share is created without a password, which must then be set if desired in the *Customize link* dialog (see Figure 2). If there is an issue with saving the password - for example because it doesn't meet the password strength requirements - the displayed error can easily be overlooked and may lead to the share remaining without a password set at all. For this reason, new link shares will now always be created with a secure random password by default. The password can still be changed or removed if needed.

Florian Kaiser

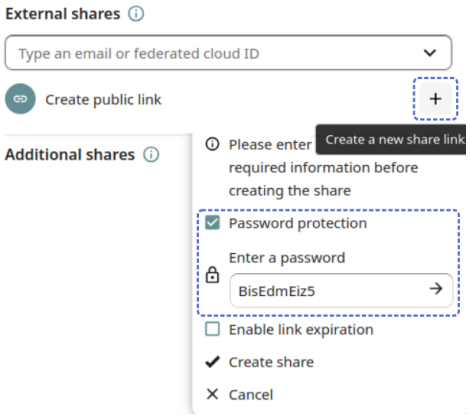

Figure 2: A screenshot of the new DataShare link share dialog

News & Events

AMD workshop

Coming up soon, MPCDF will run another AMD GPU workshop with a focus on the MI300A technology in *Viper-GPU*. This time, the workshop is split into two parts: A first part in collaboration with HLRS[13] (they also have AMD MI300A APUs in their Hunter system) with four half-day sessions of lectures and exercises in the afternoons of April 21st to 24th, 2026, and a second part with a hackathon on *Viper-GPU* from April 27th to 29th. The online lectures of the first part will be given by AMD, and the accompanying exercises will be done on AMD cloud resources. The detailed program and the registration link can be found on the HLRS website AMD Instinct GPU Training[14].

For the online hackathon on the MPCDF *Viper-GPU* system the week after, users are invited to apply with their code to bring in and to work on profiling and optimization aspects or specific porting issues, supported by experts from AMD and MPCDF. Participants are expected to have a good understanding of their code and they should also be familiar with the relevant parts of the lectures of the previous week, as there will be no introductory lectures for the week of the hackathon. Registration for this event has to be done separately at MPCDF AMD GPU Hackathon[15].

Tilman Dannert

Introduction to MPCDF Services

The next session of our introductory online course, which is designed to familiarize new users with the MPCDF compute and data services, will be held on April 30th, 2026, 14:00-16:30, online. No registration is necessary, you can just join with the link[16] published on our website. The link is only active at the time of the workshop.

Tilman Dannert

IT4Science Days 2026

Save-the-date: This year's IT4Science Days - including the "MPG DV-Treffen" - will take place from September 29th to October 1st at the MPG Faßberg campus in Göttingen. Further information as well as registration will become available through the website of the conference[17]. As usual, the "IT-Verantwortlichen Treffen" will start the day before (Monday, September 28). For reference, see also last year's meeting site[18].

Raphael Ritz

[13] https://www.hlrs.de
[14] https://www.hlrs.de/training/2026/gpu-amd
[15] https://plan.events.mpg.de/e/amd-gpu-hackathon-2026
[16] https://mpcdf-mpg-de.zoom-x.de/j/69500447868?pwd=5fQ3xerZMaMDh77215EaPyPVm4MGnM.1
[17] https://plan.events.mpg.de/event/670/
[18] https://plan.events.mpg.de/event/474/

RDA Deutschland Tagung 2026

The Research Data Alliance Germany had its yearly conference again at the "Geoforschungszentrum (GFZ)" in Potsdam February 24th-25th, 2026. Focus topics were the forthcoming "Forschungsdatengesetz" as well as the future of the National Research Data Infrastructure (NFDI). Further information including slides from most of the contributions are available from the conference website[19]. As in previous years, MPCDF helped to organize the event and contributed to the programm.

Raphael Ritz

[19] https://indico.desy.de/event/50156/overview