Package 'gpuinfo'

September 26, 2026

Title Lightweight Hardware and GPU Compute Detection

Version 0.1.0

Description Detects central processing unit and graphics processing unit hardware and reports the apparent availability of 'CUDA', 'Metal', 'ROCm', and 'OpenCL' compute backends. Detection uses operating-system information, documented platform interfaces, and optional command-line utilities, without requiring a GPU framework, 'Python', or a vendor software development kit. Backend interpretation follows the official 'CUDA' <https://docs.nvidia.com/cuda/cuda-driver-api/>, 'Metal' <https://developer.apple.com/documentation/metal>, 'ROCm' <https://rocm.docs.amd.com/>, and 'OpenCL' <https://registry.khronos.org/OpenCL/> documentation. Missing hardware, drivers, libraries, and utilities are handled safely.

License MIT + file LICENSE

URL https://github.com/tkcaccia/gpuinfo

BugReports https://github.com/tkcaccia/gpuinfo/issues

Encoding UTF-8

NeedsCompilation yes

Depends R (>= 3.6.0)

Suggests jsonlite, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Author Stefano Cacciatore [aut, cre] (ORCID: <https://orcid.org/0000-0001-7052-7156>)

Maintainer Stefano Cacciatore <tkcaccia@gmail.com>

Repository CRAN

Date/Publication 2026-09-26 16:40:08 UTC

Contents

`gpu_sitrep` *Create a GPU diagnostic report*

Description

Creates a structured diagnostic report suitable for an issue, bug report, or hardware-validation record. When printed, the report includes the R and operating-system versions, CPU, visible GPUs, CUDA components, other backend results, and the selected fallback backend.

Usage

```
gpu_sitrep(format = c("text", "json"))
```

Arguments

`format`	Format used by the object's `print()` method. `"text"` displays a human-readable report. `"json"` displays the complete JSON representation and requires the suggested `jsonlite` package.

Details

The function performs the same read-only detection as `hardware_info()`. It does not install software, initialize a framework, change device visibility, or transmit the report. Review the output before sharing it because system and environment details may identify the computer or execution environment.

Value

A visible `gpuinfo_sitrep` object containing the structured output of `hardware_info()`. The object can be inspected with normal list operations; assigning it to a variable produces no console output.

See Also

`hardware_info()`

Examples

```
report <- gpu_sitrep()
report$cpu$vendor
print(report)

if (requireNamespace("jsonlite", quietly = TRUE)) {
    print(gpu_sitrep("json"))
}
```

`hardware_info` *Collect hardware information*

Description

Creates a single structured snapshot of the CPU, visible GPUs, compute backends, execution environment, and validation evidence. This is the main entry point when more detail than the `has_*()` predicates is required.

Usage

```
hardware_info(as = c("list", "json"))
```

Arguments

`as`	Output representation. `"list"` returns native R objects. `"json"` returns a pretty-printed JSON character scalar and requires the suggested `jsonlite` package. Missing values become JSON `null` values.

Details

Detection is read-only and framework-independent. It describes the hardware and system software visible to the current R process; it does not determine whether torch, TensorFlow, or another package was compiled with GPU support. Containers, schedulers, permissions, and device-visibility environment variables can affect the result.

Value

A nested list containing R, system, CPU, GPU, backend, environment, and package-validation details when `as = "list"`; otherwise a JSON character scalar containing the same information.

Returned components

The report contains the following named components:

r R version and platform.

system Operating-system name, release, version, and architecture.

cpu CPU model, vendor, architecture, logical-core count, and OS.

gpu One row per detected visible GPU, including device id, vendor, model, reported memory, backend, and NVIDIA compute capability when known. An empty data frame means that no visible GPU was detected.

accelerators A vendor-neutral accelerator table with memory and backend-status information.

capabilities Per-device precision, memory, and backend capability information. Unknown capabilities are represented by `NA`.

cuda NVIDIA GPU, driver, runtime, Toolkit, `nvcc`, versions, and apparent CUDA usability. The CUDA version advertised by `nvidia-smi` is a driver-supported maximum, not necessarily an installed Toolkit.

metal macOS platform, Apple silicon, device, framework, support, and apparent Metal usability.

rocm AMD GPU, kernel driver, runtime, utilities, version, and apparent ROCm usability.

opencl Loader, platforms, devices, vendors, memory, FP64 support, and apparent OpenCL availability.

available_backends Detected usable backends, always including the CPU fallback.

best_backend The first usable backend in the package priority order CUDA, Metal, ROCm, OpenCL, then CPU.

environment Container, CI, WSL, scheduler, and device-visibility information. Values are reported without changing the environment.

validation Real-hardware validation evidence shipped with this package version. This describes maintainer testing, not the current host.

See Also

`gpu_sitrep()`, `has_gpu()`, `has_cuda()`, `has_metal()`, `has_rocm()`, `has_opencl()`

Examples

```
info <- hardware_info()
names(info)
info$cpu
info$gpu
info$cuda$usable

if (requireNamespace("jsonlite", quietly = TRUE)) {
    json <- hardware_info("json")
}
```

---

has_cuda *Test whether CUDA appears usable*

Description

Returns `TRUE` when an NVIDIA GPU is visible through an apparently working NVIDIA driver. Detection uses a dynamically loaded CUDA driver library and, when available, read-only `nvidia-smi` queries.

Usage

```
has_cuda()
```

Details

The CUDA Toolkit and nvcc compiler are not required: applications can use a GPU through a driver and a bundled runtime without a local Toolkit. A positive result does not imply that torch, TensorFlow, or another R package was built with CUDA support. Inspect `hardware_info()$cuda` to distinguish GPU, driver, runtime, Toolkit, compiler, and version information.

Value

One non-missing logical value: TRUE when CUDA appears usable, otherwise FALSE.

See Also

hardware_info(), has_gpu()

Examples

```
has_cuda()
hardware_info()$cuda
```

---

has_gpu *Test whether a GPU is detected*

Description

Returns TRUE when at least one graphics processor is visible to the current R process. Detection combines native vendor-library queries with portable operating-system and command-line fallbacks when available.

Usage

```
has_gpu()
```

Details

This tests hardware visibility, not whether any particular compute backend or R framework can use the device. A GPU with a missing driver can therefore be detected while all backend predicates remain FALSE. Conversely, a container or scheduler can hide a physical GPU from the process.

Value

One non-missing logical value: TRUE when a visible GPU is detected, otherwise FALSE.

See Also

hardware_info(), has_cuda(), has_metal(), has_rocm(), has_opencl()

Examples

```
has_gpu()
```

---

has_metal *Test whether Metal appears usable*

Description

Returns TRUE on macOS when a Metal-capable graphics device is visible. Detection uses the Metal framework when it can be loaded and supplements it with system_profiler and Apple-silicon platform information.

Usage

```
has_metal()
```

Details

This reports operating-system and hardware capability, not framework-specific support such as torch MPS. A positive result therefore does not guarantee that another R package can execute work through Metal. Detailed fields are available in hardware_info()$metal.

Value

One non-missing logical value: TRUE when Metal appears usable, otherwise FALSE. It is FALSE on non-macOS platforms.

See Also

hardware_info(), has_gpu()

Examples

```
has_metal()
hardware_info()$metal
```

has_opencl *Test whether OpenCL is available*

Description

Returns TRUE when an OpenCL loader is present, at least one platform can be enumerated, and at least one compute device is visible. Detection uses the OpenCL library directly and can supplement it with the read-only clinfo utility.

Usage

```
has_opencl()
```

Details

OpenCL devices can be GPUs, CPUs, or other accelerators, so this predicate is not synonymous with has_gpu(). It also does not guarantee that a particular R package has OpenCL support. Inspect hardware_info()$opencl for platform, device, vendor, memory, and FP64 details.

Value

One non-missing logical value: TRUE when OpenCL appears available, otherwise FALSE.

See Also

hardware_info(), has_gpu()

Examples

```
has_opencl()
hardware_info()$opencl
```

has_rocm *Test whether ROCm appears usable*

Description

Returns TRUE when an AMD GPU is visible and a usable ROCm/HIP runtime path is apparent. Detection can use the HIP runtime library, /dev/kfd, rocminfo, rocm-smi, hipcc, and common ROCm installation locations.

Usage

```
has_rocm()
```

Details

Merely finding ROCm files is insufficient: the predicate requires compatible visible hardware and runtime evidence. A positive result does not guarantee that another R package was compiled with ROCm support. Inspect `hardware_info()$rocm` for the individual detection fields and version.

Value

One non-missing logical value: `TRUE` when ROCm appears usable, otherwise `FALSE`.

See Also

`hardware_info()`, `has_gpu()`

Examples

```
has_rocm()
hardware_info()$rocm
```

Index