

GPU Acceleration & Performance

Modern GPUs transform password cracking from hours to minutes. This lesson covers how to harness GPU parallel processing in Hashcat, tune performance, manage thermals, and troubleshoot common issues.

Learning Objectives

- Understand why **GPUs outperform CPUs** for hash cracking
 - Prepare your system with the correct **drivers and compute frameworks**
 - Use **device selection** and **workload profiles** to control performance
 - Configure **temperature limits** and **runtime constraints**
 - Troubleshoot **common GPU issues** in Hashcat
-

Why GPU?

CPUs are designed for complex sequential tasks with a few powerful cores. GPUs contain **thousands of smaller cores** optimized for parallel computation — perfect for hashing.

GPU Advantages

- **Parallel computation:** Thousands of cores process hashes simultaneously
- **High throughput:** 10–100x faster than CPU for most hash types
- **Efficient for large volumes:** Millions of hash candidates tested per second

Example: An NVIDIA RTX 4090 can compute ~160 billion MD5 hashes per second. A high-end CPU manages ~5 billion. That is a 32x difference on a single GPU.

Preparation

Before using GPU acceleration, your system must have the correct drivers and compute frameworks installed.

NVIDIA GPUs

- Install the latest **NVIDIA driver** from [nvidia.com](https://www.nvidia.com)
- Install **CUDA Toolkit** (includes OpenCL support)
- Verify with `nvidia-smi`

AMD GPUs

- Install **AMD Adrenalin drivers** or ROCm
- Ensure **OpenCL runtime** is available
- Verify with `clinfo`

Note: Hashcat supports OpenCL, CUDA, and HIP backends. CUDA is preferred for NVIDIA; OpenCL works for

both vendors.

Checking Available Devices

Use `hashcat -I` to list all available compute devices on your system.

```
# List all OpenCL/CUDA devices
hashcat -I
```

Sample output:

```
OpenCL Info:
=====
Platform ID #1
  Name.....: NVIDIA CUDA
  Vendor....: NVIDIA Corporation
  Device ID #1
    Name.....: NVIDIA GeForce RTX 4090
    Type.....: GPU
    Memory.....: 24564 MB
    OpenCL Version.: OpenCL 3.0 CUDA

Platform ID #2
  Name.....: Intel(R) OpenCL
  Device ID #2
    Name.....: Intel(R) Core(TM) i9-13900K
```

Device IDs matter. Use these IDs with `-d` to target specific devices.

Using GPU in Hashcat

The `-D` flag selects the device type. Use `-D 2` to target GPUs exclusively.

```
# Dictionary attack on GPU
hashcat -m 0 -a 0 -D 2 -o results.txt hashes.txt rockyou.txt
```

Flag	Value	Description
<code>-D 1</code>	CPU	Use CPU only
<code>-D 2</code>	GPU	Use GPU only
<code>-D 1,2</code>	CPU + GPU	Use both (not always optimal)

Selecting Specific GPUs

```
# Use only GPU devices 1 and 2  
hashcat -m 0 -a 0 -D 2 -d 1,2 hashes.txt rockyou.txt
```

Workload Profiles

The `-w` flag controls how aggressively Hashcat uses the GPU. Higher profiles yield faster speeds but can make the system unresponsive.

Profile	Name	Description	Use Case
<code>-w 1</code>	Low	Minimal GPU usage	Desktop remains usable
<code>-w 2</code>	Medium	Default balance	Background cracking
<code>-w 3</code>	High	Aggressive GPU usage	Dedicated cracking rig
<code>-w 4</code>	Maximum	Full GPU saturation	Headless server, no display

```
# High workload profile
hashcat -m 0 -a 3 -D 2 -w 3 hashes.txt ?a?a?a?a?a
```

Warning: Profile `-w 4` can cause display freezes and system instability on machines with a single GPU that also drives the display.

Temperature Control

Hashcat provides built-in thermal management to protect your hardware.

```
# Abort if GPU reaches 85C, throttle to stay under 75C
hashcat -m 0 -a 0 -D 2 \
  --gpu-temp-abort=85 \
  --gpu-temp-retain=75 \
  hashes.txt rockyou.txt
```

Temperature Flags

- `--gpu-temp-abort=N` — Immediately stop if any GPU reaches N degrees Celsius
- `--gpu-temp-retain=N` — Dynamically throttle speed to keep the GPU at or below N degrees

Note: Monitor temperatures in real-time with `nvidia-smi -l 1` (NVIDIA) or `watch -n1 sensors` (Linux).

Performance Tuning

Advanced flags for fine-tuning GPU performance:

Flag	Description	Notes
<code>-n</code>	Kernel accel (threads per block)	NVIDIA tuning; higher = more parallelism
<code>-u</code>	Kernel loops (work blocks)	AMD tuning; controls work granularity
<code>-T</code>	Kernel threads (vector width)	Number of vector elements per thread

Runtime Limits

```
# Run for at most 1 hour, show status every 60 seconds
hashcat -m 0 -a 3 -D 2 \
  --runtime=3600 \
  --status-timer=60 \
  hashes.txt ?a?a?a?a?a?a
```

Tip: Let Hashcat auto-tune first (`-w 3` without `-n` / `-u` / `-T`). Only manually tune if you understand your GPU architecture and need to squeeze out the last few percent.

Troubleshooting

Common GPU issues and solutions:

Low Hash Speed

- Check workload profile (`-w 3` or `-w 4`)
- Ensure GPU drivers are up to date
- Verify you are using `-D 2` (not CPU)

GPU Overheating

- Set `--gpu-temp-retain=75`
- Improve case airflow / cooling
- Reduce workload profile

Multi-GPU Issues

- Use `-d` to select specific devices
- Check for PCIe bandwidth bottlenecks
- Ensure matching driver versions

Driver / RAM Problems

- Reinstall CUDA or OpenCL runtime
 - Check GPU memory with `nvidia-smi`
 - Close other GPU-intensive applications
-

Lab & Homework

Lab Exercises

1. **Device check** — run `hashcat -I` and system tools
2. **GPU dictionary attack** — crack hashes with `-D 2`
3. **GPU mask attack** — brute-force with GPU
4. **Workload comparison** — test `-w 1` through `-w 4`
5. **Temperature limits** — set thermal safeguards
6. **CPU vs GPU** — compare performance

Homework Tasks

1. Benchmark all hash modes on your GPU
2. Optimize workload for a specific hash type
3. Document multi-GPU or CPU-vs-GPU results

