```
import pytest
from unittest.mock import patch, MagicMock
from clusterops import (
    list_available_cpus,
    execute_with_cpu_cores,
    list_available_gpus,
    execute_on_gpu,
    execute_on_multiple_gpus,
    execute_on_cpu,
)


# Example function to run
def sample_task(n: int) -> int:
    return n * n


# Mock the environment for pytest
@pytest.fixture
def mock_psutil():
    with patch("psutil.cpu_count", return_value=12):
        with patch("psutil.Process") as mock_process:
            mock_process.return_value.cpu_affinity = MagicMock()
            yield
```

```
@pytest.fixture
def mock_gputil():
    with patch("GPUtil.getGPUs") as mock_get_gpus:
        mock_get_gpus.return_value = [
            MagicMock(
                id=0,
                name="GPU 0",
                memoryFree=10000,
                memoryTotal=16000,
            ),
            MagicMock(
                id=1, name="GPU 1", memoryFree=8000, memoryTotal=16000
            ),
        ]
        yield


@pytest.fixture
def mock_ray():
    with patch("ray.init"), patch("ray.remote") as mock_remote, patch(
        "ray.get"
    ):
        mock_remote.return_value = MagicMock(return_value=sample_task)
        yield
```

```
# Test listing available CPUs
def test_list_available_cpus(mock_psutil):
    cpus = list_available_cpus()
    assert cpus == list(range(12)), "Should list 12 CPU cores."


# Test executing a function on a specific CPU
def test_execute_on_cpu(mock_psutil):
    result = execute_on_cpu(0, sample_task, 10)
    assert result == 100, "Expected task result to be 100."


# Test executing with multiple CPU cores
def test_execute_with_cpu_cores(mock_psutil):
    result = execute_with_cpu_cores(4, sample_task, 10)
    assert result == 100, "Expected task result to be 100."


# Test listing available GPUs
def test_list_available_gpus(mock_gputil):
    gpus = list_available_gpus()
    assert len(gpus) == 2, "Should list 2 available GPUs."
    assert gpus[0]["name"] == "GPU 0"
    assert gpus[1]["name"] == "GPU 1"
```

```
# Test executing on a specific GPU

def test_execute_on_gpu(mock_gputil, mock_ray):

    result = execute_on_gpu(0, sample_task, 10)

    assert result == 100, "Expected task result to be 100 on GPU 0."



# Test executing on multiple GPUs

def test_execute_on_multiple_gpus(mock_gputil, mock_ray):

    results = execute_on_multiple_gpus([0, 1], sample_task, 10)

    assert len(results) == 2, "Expected results from 2 GPUs."

    assert all(

        result == 100 for result in results

    ), "Expected task results to be 100 on all GPUs."
```