```
from typing import Any



from clusterops import (
    execute_on_gpu,
    execute_on_multiple_gpus,
    list_available_gpus,
    execute_with_all_cpu_cores,
    execute_on_cpu,
)
from swarms.utils.loguru_logger import initialize_logger

logger = initialize_logger(log_folder="clusterops_wrapper")



def exec_callable_with_clusterops(
    device: str = "cpu",
    device_id: int = 1,
    all_cores: bool = True,
    all_gpus: bool = False,
    func: callable = None,
    enable_logging: bool = True,
    *args,
    **kwargs,
) -> Any:
    """
```

Executes a given function on a specified device, either CPU or GPU.

This method attempts to execute a given function on a specified device, either CPU or GPU. It logs the device selection and the number of cores or GPU ID used. If the device is set to CPU, it can use all available cores or a specific core specified by `device_id`. If the device is set to GPU, it uses the GPU specified by `device_id`.

Args:

device (str, optional): The device to use for execution. Defaults to "cpu".

device_id (int, optional): The ID of the GPU to use if device is set to "gpu". Defaults to 0.

all_cores (bool, optional): If True, uses all available CPU cores. Defaults to True.

all_gpus (bool, optional): If True, uses all available GPUs. Defaults to False.

func (callable): The function to execute.

enable_logging (bool, optional): If True, enables logging. Defaults to True.

*args: Additional positional arguments to be passed to the execution method.

**kwargs: Additional keyword arguments to be passed to the execution method.

Returns:

Any: The result of the execution.

Raises:

ValueError: If an invalid device is specified.

Exception: If any other error occurs during execution.

"""

if func is None:

raise ValueError("A callable function must be provided")

```
try:
    if enable_logging:
        logger.info(f"Attempting to run on device: {device}")
    device = device.lower()

    if device == "cpu":
        if enable_logging:
            logger.info("Device set to CPU")

        if all_cores:
            if enable_logging:
                logger.info("Using all CPU cores")
            return execute_with_all_cpu_cores(
                func, *args, **kwargs
            )

        if device_id is not None:
            if enable_logging:
                logger.info(
                    f"Using specific CPU core: {device_id}"
                )
            return execute_on_cpu(
                device_id, func, *args, **kwargs
            )
```

```
    elif device == "gpu":
        if enable_logging:
            logger.info("Device set to GPU")

        if all_gpus:
            if enable_logging:
                logger.info("Using all available GPUs")
            gpus = [int(gpu) for gpu in list_available_gpus()]
            return execute_on_multiple_gpus(
                gpus, func, *args, **kwargs
            )

        if enable_logging:
            logger.info(f"Using GPU device ID: {device_id}")
        return execute_on_gpu(device_id, func, *args, **kwargs)

    else:
        raise ValueError(
            f"Invalid device specified: {device}. Supported devices are 'cpu' and 'gpu'."
        )

except ValueError as e:
    if enable_logging:
        logger.error(
            f"Invalid device or configuration specified: {e}"
        )
```

```
    raise
except Exception as e:
    if enable_logging:
        logger.error(f"An error occurred during execution: {e}")
    raise
```