import os

import shutil

import glob

import multiprocessing

def copy\_latest\_logs(server, src\_dir, dst\_dir):

# Sort files by modification time, and select the latest .log file

log\_files = sorted(glob.glob(f"\\\\{server}\\{src\_dir}\\\*.log"), key=os.path.getmtime, reverse=True)

if log\_files:

latest\_log\_file = log\_files[0]

\_, filename = os.path.split(latest\_log\_file)

dst\_file = os.path.join(dst\_dir, filename)

shutil.copy2(latest\_log\_file, dst\_file)

print(f"Copied latest log file {latest\_log\_file} from {server} to {dst\_file}")

else:

print(f"No log files found in {src\_dir} on {server}")

# Sort files by modification time, and select the latest .log.gz file

gz\_files = sorted(glob.glob(f"\\\\{server}\\{src\_dir}\\\*.log.gz"), key=os.path.getmtime, reverse=True)

if gz\_files:

latest\_gz\_file = gz\_files[0]

\_, filename = os.path.split(latest\_gz\_file)

dst\_file = os.path.join(dst\_dir, filename)

shutil.copy2(latest\_gz\_file, dst\_file)

print(f"Copied latest gzipped log file {latest\_gz\_file} from {server} to {dst\_file}")

else:

print(f"No gzipped log files found in {src\_dir} on {server}")

def search\_log\_files(log\_files, patterns):

for log\_file in log\_files:

with open(log\_file, 'r') as f:

lines = f.readlines()

for pattern in patterns:

matches = [line for line in lines if re.search(pattern, line)]

if matches:

print(f"Matches for pattern '{pattern}' in file {log\_file}:")

for match in matches:

print(match.strip())

def parallel\_copy\_and\_search(servers, src\_dir, dst\_dir, patterns):

pool = multiprocessing.Pool(processes=len(servers))

for server in servers:

pool.apply\_async(copy\_latest\_logs, args=(server, src\_dir, dst\_dir))

pool.close()

pool.join()

log\_files = glob.glob(f"{dst\_dir}\\\*.log\*")

pool = multiprocessing.Pool()

chunk\_size = len(log\_files) // multiprocessing.cpu\_count()

for i in range(0, len(log\_files), chunk\_size):

chunk = log\_files[i:i+chunk\_size]

pool.apply\_async(search\_log\_files, args=(chunk, patterns))

pool.close()

pool.join()