PCET's
Pimpri
Chinchwad
University, Pune

Learn | Grow | Achieve

Name	
Checked By	Dr.Avinash Golande
Sign	
Date	
Grade	

EXPERIMENT NO. 01

- **Aim:** Write a Python script to simulate resource scaling scenarios (horizontal vs. vertical scaling) using simple process/thread creation and control.

- **Objective:**
  - ➢ To understand the concept of elasticity in cloud computing.
  - ➢ To study the differences between horizontal scaling and vertical scaling.
  - ➢ To simulate the effect of scaling using threads and processes in Python.
  - ➢ To measure the performance difference between no scaling, horizontal scaling, and vertical scaling.
  - ➢ To relate the local simulation with real-world cloud infrastructure (like AWS, Azure, GCP).

- **Resources used :-** PC, Laptop**, VS** code, Jyputer

- **Theory :-**

  ➢ **What is Scaling?**

  Scaling refers to increasing or decreasing the system's ability to handle workload. It helps maintain performance under varying loads.

  ➢ **Vertical Scaling:**

  Vertical scaling (or **scaling up**) increases the resources of a **single machine**, like more CPU cores or more memory. It's useful when the system is not distributed and the application runs on a single server.
  Example: Upgrading a virtual machine (VM) from 2 vCPUs to 8 vCPUs.

  ➢ **Horizontal Scaling:**

  Horizontal scaling (or **scaling out**) increases performance by **adding more machines** or nodes to handle the load. It typically involves load balancers and multiple instances.

Example: Running multiple copies of a web server behind a load balancer.

- **Threads and Processes in Scaling**

  To simulate or implement scaling locally (without real cloud infrastructure), we use **threads** and **processes** in programming:

  - **Threads:**
    1. Threads run within the same process and share memory space.
    2. Good for **I/O-bound** operations (e.g., file read/write, API requests).
  - **Processes:**
    1. Processes run in **separate memory spaces**, ideal for **CPU-bound** tasks.
    2. In Python, we often use the multiprocessing module for this.
    3. Closer to how cloud VMs or containers behave (each having isolated resources).

  In Cloud Terms:

  1. **Vertical scaling** is like making each thread or process faster (by giving it more CPU/RAM).
  2. **Horizontal scaling** is like running multiple threads or processes in parallel to divide the load

  .

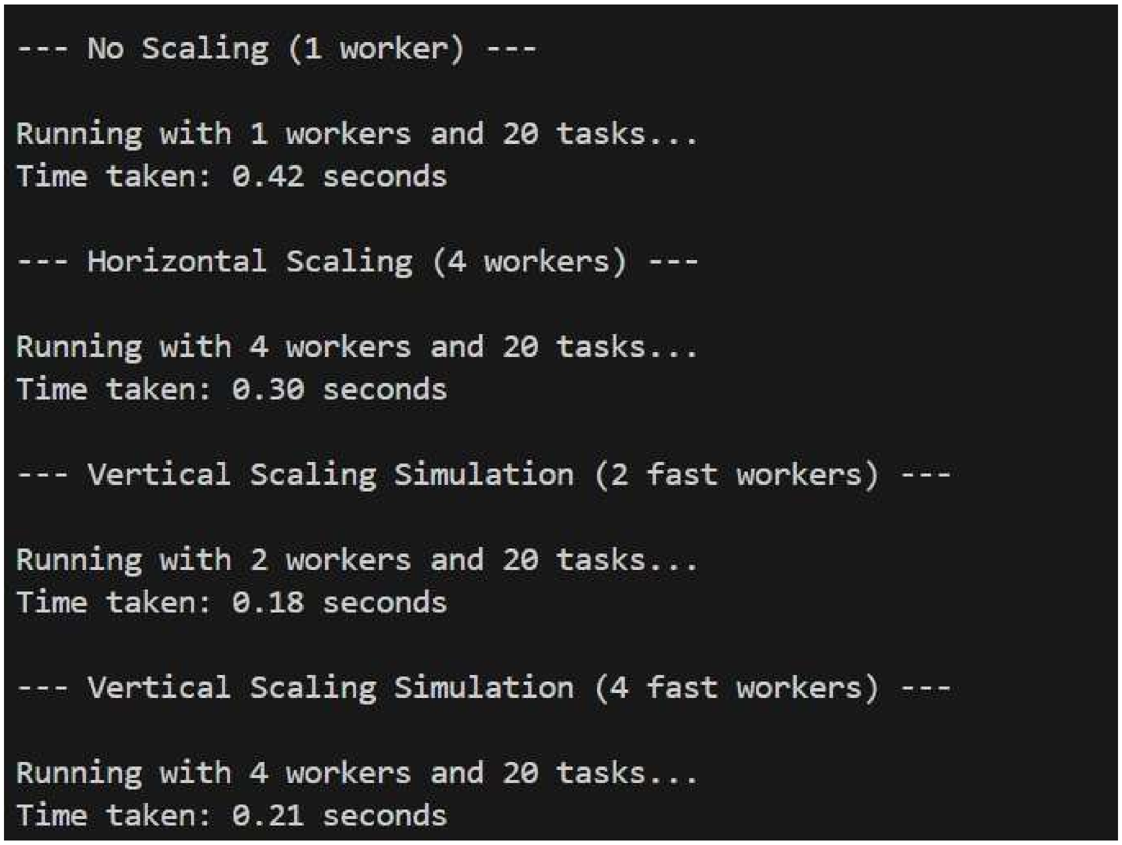
- **Algorithm**

1. Define total workload.
2. Process with a single thread/process (no scaling).
3. Add more threads or processes to simulate **horizontal scaling**.
4. Simulate **vertical scaling** by reducing task intensity or making each task faster.
5. Compare execution time across all scenario

- **CODE :-**

```
import time
from multiprocessing import Pool

# Simulate CPU-bound task
def simulate_task(task_id):
    total = 0
    for i in range(10**5):
        total += i * i
    return f"Task {task_id} completed"

def faster_task(task_id):
    total = 0
    for i in range(10**4):  # Less load = faster execution
        total += i * i
    return f"Fast Task {task_id} completed"

def simulate_workload(num_tasks, num_workers, use_fast_task=False):
    print(f"\nRunning with {num_workers} workers and {num_tasks} tasks...")
    task_func = faster_task if use_fast_task else simulate_task
    start_time = time.time()
    with Pool(num_workers) as pool:
        results = pool.map(task_func, range(num_tasks))
    end_time = time.time()
    print(f"Time taken: {end_time - start_time:.2f} seconds")
    return results

# Main execution
if __name__ == "__main__":
    num_tasks = 20

    print("\n--- No Scaling (1 worker) ---")
    simulate_workload(num_tasks, 1)

    print("\n--- Horizontal Scaling (4 workers) ---")
    simulate_workload(num_tasks, 4)

    print("\n--- Vertical Scaling Simulation (2 fast workers) ---")
    simulate_workload(num_tasks, 2, use_fast_task=True)

    print("\n--- Vertical Scaling Simulation (4 fast workers) ---")
    simulate_workload(num_tasks, 4, use_fast_task=True)
```

- **Output:-**



```
--- No Scaling (1 worker) ---

Running with 1 workers and 20 tasks...
Time taken: 0.42 seconds

--- Horizontal Scaling (4 workers) ---

Running with 4 workers and 20 tasks...
Time taken: 0.30 seconds

--- Vertical Scaling Simulation (2 fast workers) ---

Running with 2 workers and 20 tasks...
Time taken: 0.18 seconds

--- Vertical Scaling Simulation (4 fast workers) ---

Running with 4 workers and 20 tasks...
Time taken: 0.21 seconds
```

- **Conclusion:**
    1. Elasticity is critical in modern cloud applications.
    2. Vertical Scaling is simple and suitable for single-node apps.
    3. Horizontal Scaling is scalable and fault-tolerant, ideal for large distributed systems.
    4. Cloud platforms like AWS, Azure, and Google Cloud provide built-in tools for scaling easily