

Activity 1

1.

Scale-Out: Scale-out refers to increasing a system's processing capacity by adding more nodes (such as servers or computers). This approach improves performance by distributing the workload. For example, using cluster computing or distributed database systems to handle high concurrent requests.

Scale-Up: Scale-up refers to enhancing the performance of a single node by adding more resources (such as CPU, memory). This approach is suitable when a single application or database needs more resources to handle a larger workload.

Differences:

Scale-Out is suited for distributed systems, easily scalable, and typically offers higher fault tolerance, as the failure of one node does not impact the entire system's availability.

Scale-Up is suited for single-machine systems, can respond faster to the needs of a single node, but has limited scalability and may face single points of failure.

2.

Distributed Database: A distributed database is a database system where data is stored across multiple physical locations. This system allows query requests to be distributed across nodes, improving data access efficiency and fault tolerance.

Example: Google Spanner is a globally distributed database that supports strong consistency and high availability.

Distributed Computing: Distributed computing refers to multiple computing devices working together to solve a computational problem. Tasks are divided into smaller sub-tasks, distributed across different computing nodes, and processed in parallel, then results are aggregated.

Example: Apache Hadoop is a distributed computing framework used for processing large-scale datasets.

Distributed File Systems: Distributed file systems store data across multiple physical devices, allowing users to access these distributed data as if they were on a single system.

Example: Google File System (GFS) is a distributed file system designed for large-scale data processing.

Distributed Applications: Distributed applications consist of multiple independent computing processes running on different machines, communicating over a network to work together. Each process may perform different tasks, collectively fulfilling the overall functionality of the application.

Example: An email system is a distributed application involving multiple servers (sending, receiving, and storing emails) that work together.

3.

Heterogeneity

Explanation: This challenge involves ensuring that a distributed system can operate across different platforms, hardware, and network environments. It requires the system to support various programming languages, operating systems, and hardware configurations.

Use Case: A web application developed using React Native should work seamlessly on both Android and iOS platforms without requiring significant changes in the codebase.

Transparency

Explanation: Transparency in distributed systems refers to hiding the complexity of the system from the users and making the system appear as a single, unified system. This includes abstracting the physical separation of computing nodes from the end users through virtualization or other means.

Use Case: In cloud computing, users interact with virtual machines without knowing the actual physical servers or locations where their data is being processed, maintaining the illusion of a single coherent system.

Openness

Explanation: Openness involves designing a system that is open for extension, interoperation, and compliance with existing standards. This ensures that the system can evolve and interact with other systems.

Use Case: An open-source database management system like MySQL, which allows developers to modify and extend its functionalities to meet specific needs, is an example of an open system.

Concurrency

Explanation: Concurrency involves managing the execution of multiple processes simultaneously while ensuring that shared resources are accessed in a synchronized and consistent manner.

Use Case: In a distributed banking system, multiple users might try to access and modify account balances simultaneously. Concurrency control mechanisms ensure that all transactions are processed correctly and that balances remain accurate.

Security

Explanation: Security in distributed systems involves protecting data, processes, and communication from unauthorized access and ensuring privacy, integrity, and trust within the system.

Use Case: An online payment system must implement strong encryption, authentication, and authorization mechanisms to protect sensitive financial information and ensure secure transactions.

Scalability

Explanation: Scalability refers to the ability of a system to handle growing amounts of work by adding resources such as additional nodes or servers. This includes both horizontal scaling (adding more machines) and vertical scaling (adding more power to existing machines).

Use Case: A social media platform like Twitter needs to scale horizontally by adding more servers

as the number of users and the volume of data increases, ensuring that the system remains responsive and available.

Fault Tolerance

Explanation: Fault tolerance involves designing the system to continue functioning even in the presence of failures. This is achieved through mechanisms like replication, leader appointment, and failover strategies.

Use Case: In a distributed database, data is often replicated across multiple servers. If one server fails, the system can automatically switch to another replica to maintain availability and prevent data loss.

Activity 2

1. Setting Up Virtual Machines (Master and Worker)

Created two virtual machines using VirtualBox, one for Master and one for Worker.

Installed Ubuntu 24.04 LTS on both machines.

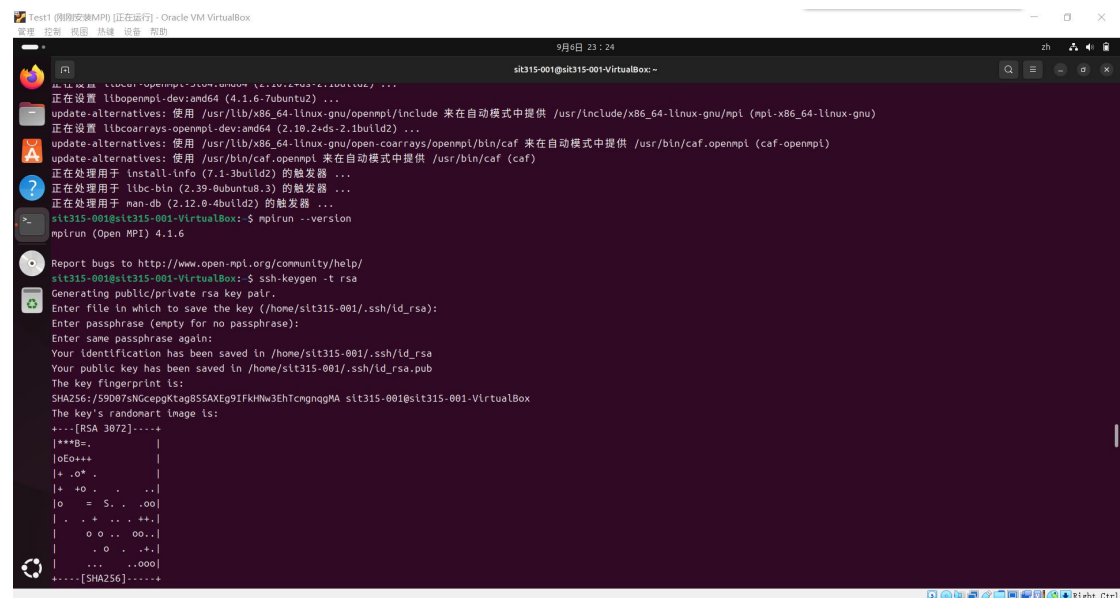
Configured network to use Bridged Adapter mode to place both VMs on the same network.

2. Installing Open MPI

Installed Open MPI on both Master and Worker VMs

3. Setting Up SSH Key-based Authentication

Generated an SSH key and copied it from the Master VM to the Worker VM to enable passwordless login



```
Test1 (Ubuntu 24.04 LTS) [正在运行] - Oracle VM VirtualBox
管理 控制 帮助 设置 帮助

9月4日 23:24
slt315-001@slt315-001-VirtualBox:~$

正在设置 libopenmpi-dev:amd64 (4.1.6-7ubuntu2) ...
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/openmpi/include 来在自动模式中提供 /usr/include/x86_64-linux-gnu/mpl (mpi-x86_64-linux-gnu)
正在设置 libcoarrays-openmpi-dev:amd64 (2.10.2+ds-2.1build2) ...
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/open-coarrays/openmpi/bin/caf 来在自动模式中提供 /usr/bin/caf.openmpi (caf-openmpi)
update-alternatives: 使用 /usr/bin/caf.openmpi 来在自动模式中提供 /usr/bin/caf (caf)
正在处理用于 install-info (7.1-3build2) 的触发器 ...
正在处理用于 libc-bin (2.39-0ubuntu8.3) 的触发器 ...
正在处理用于 man-db (2.12.0-4build2) 的触发器 ...
slt315-001@slt315-001-VirtualBox:~$ mpirun --version
mpirun (Open MPI) 4.1.6

Report bugs to http://www.open-mpi.org/community/help/
slt315-001@slt315-001-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/slt315-001/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/slt315-001/.ssh/id_rsa
Your public key has been saved in /home/slt315-001/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:59D07sNGcepgKtag855AXEg91FKHw3EhTcngnqMA slt315-001@slt315-001-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|**B+|
|oEo++|
|+ .o+|
|+ .o+ . .+|
|o = S. . .oo|
| . + . . .++|
|o o . . .oo+|
| . o . .+|
|... ..ooo|
+---[SHA256]-----+
```

```

sit315-001@sit315-001-VirtualBox:~$ ssh-copy-id sit315-001@192.168.1.146
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
sit315-001@192.168.1.146's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'sit315-001@192.168.1.146'"
and check to make sure that only the key(s) you wanted were added.

sit315-001@sit315-001-VirtualBox:~$

```

Successfully tested passwordless SSH login:

```

sit315-001@sit315-001-VirtualBox:~$ ssh sit315-001@192.168.1.146
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

扩展安全维护（ESM）Applications 未启用。

0 更新可以立即应用。

8 个额外的安全更新可以通过 ESM Apps 来获取安装。
可通过以下途径了解如何启用 ESM Apps: at https://ubuntu.com/esm

sit315-001@sit315-002-virtualbox:~$

```

4. Writing and Compiling the Hello MPI Program

Created the hello_mpi.cpp program and compiled it on both Master and Worker VMs

mpic++ -o hello_mpi hello_mpi.cpp

Creating Host File and Running the MPI Program content with IP with every node

Ran the MPI program with 4 processes using:

mpirun --hostfile hosts -np 4 ./hello_mpi

6.The output confirmed the MPI program ran successfully on both the Master and Worker VMs

```
sit315-001@sit315-001-VirtualBox:~$ mpirun --hostfile hosts -np 4 ./hello_mpi
Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Hello SIT315. You get this message from sit315-001-VirtualBox, rank 1 out of 4
Hello SIT315. You get this message from sit315-001-VirtualBox, rank 0 out of 4
Hello SIT315. You get this message from sit315-002-virtualbox, rank 2 out of 4
Hello SIT315. You get this message from sit315-002-virtualbox, rank 3 out of 4
sit315-001@sit315-001-VirtualBox:~$
```