**《并行计算》课程实验报告**

**实验1：基于华为云的并行环境搭建和使用**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 常添 | | 院系 | | 计算学部 | | | 学号 | | 2022111699 | |
| 任课教师 | | 张伟哲 | | | | 指导教师 |  | | | | |
| 实验地点 | | 格物楼213 | | | | 实验时间 | 2024年9月20日 | | | | |
| 实验课表现 | | 出勤、表现得分 | |  | | 实验报告  得分 |  | | 实验总分 | |  |
| 操作结果得分 | |  | |
| **一、实验目的** | | | | | | | | | | | |
| 要求：需分析本次实验的基本目的，并综述你是如何实现这些目的的？  本次实验的主要目的是在华为云环境下，搭建并使用并行计算环境，通过集群环境运行并行程序，掌握集群环境的配置和并行计算的基本原理。具体实验目标包括：  1. 了解华为云的使用：通过实际操作，熟悉华为云的弹性云服务器的配置与管理流程。  2. 掌握集群环境的搭建：通过购买和配置多台云服务器，搭建并行计算环境，实现节点之间的通信和任务分配。  3. 编写并运行并行程序：编写一个简单的MPI并行程序（如Hello World程序和PI值计算程序），并在集群环境中运行，体会并行计算的效率与性能提升。  4. 比较串行与并行性能差异：通过对比串行和并行程序在不同处理器数目下的运行时间，分析并行计算的优势和性能瓶颈。 | | | | | | | | | | | |
| **二、实验内容** | | | | | | | | | | | |
| 该部分填写在实验过程中，你都完成了哪些工作。  1. 登录华为云并购买服务器：  - 通过华为云官网登录账户，并购买了三台弹性云服务器ECS，选择了openEuler 20.03镜像作为操作系统，以确保软件的兼容性和稳定性。  2. 配置服务器环境：  - 完成了三台服务器的配置，包括设置服务器名称、密码，并为三台服务器配置了私有IP地址。  - 记录了每台服务器的私有IP地址，以便在后续的集群配置中使用。  3. 远程登录服务器并创建用户：  - 使用CloudShell远程登录到每台云服务器，并通过命令创建了新的用户，为该用户设置密码并赋予root权限。  4. 配置SSH免密登录：  - 配置了集群中的三台服务器之间的免密登录，通过生成本地密钥并将其复制到其他服务器，实现了服务器之间的无密码通信。  - 通过编辑/etc/hosts文件，添加了三台服务器的IP和主机名映射，确保各服务器能够正常通信。  5. 安装MPI并配置环境变量：  - 在每台服务器上安装了必要的依赖包（如GCC编译器和MPICH），并通过修改环境变量，配置了MPI的路径，使得MPI相关命令可以在系统中正常使用。  6. 编写并行程序：  - 创建了一个简单的MPI并行程序mpi\_hello\_world.c，该程序使用MPI机制打印出各个节点的处理器名称和进程编号。  - 编写了Makefile用于自动编译该程序，并成功在集群环境中进行了编译。  7. 主机配置文件的设置：  - 编写了主机配置文件config，指定了每台服务器的计算资源分配情况（每台服务器2个处理器核心），为后续的并行计算做好准备。  8. 并行程序的运行与监测：  - 在主节点（ecs-hw-0001）上运行了并行程序，通过mpirun命令在集群中启动了程序，并成功得到了每个处理节点的输出，验证了程序的正确性。  - 通过查看程序的输出结果，确认了并行计算任务在集群中的正常运行。  9. 编写并行PI计算程序：  - 编写了一个用于计算PI值的并行程序pi\_mpi.c，并通过MPI机制将计算任务分配给多个节点进行并行处理。  - 在集群环境中分别编译并运行了串行代码和并行代码，并对不同进程数下的程序运行时间进行了记录。  10. 性能测试与结果分析：  - 逐步增加并行进程数，观察程序的执行时间如何随着进程数的增加而变化。通过实验，记录了程序的执行时间，并通过截图保存结果。 | | | | | | | | | | | |
| **三、实验结果** | | | | | | | | | | | |
| **文本  描述已自动生成** | | | | | | | | | | | |
| **四、思考题** | | | | | | | | | | | |
| **思考题1：简述华为云环境下集群的搭建过程。**  在华为云环境下搭建集群的过程如下：  1.购买云服务器：首先通过华为云官网购买多台弹性云服务器（ECS），根据实验需求，选择合适的操作系统镜像（如openEuler）以及计算资源配置（如CPU核数和内存）。  2.配置服务器网络：在每台服务器上记录其私有IP地址，并在/etc/hosts文件中手动配置服务器名称与IP地址的映射，确保各个服务器之间可以通过IP地址或主机名互相访问。  3. SSH免密登录配置：生成本地SSH密钥，并将公钥复制到其他集群节点上，通过SSH免密登录配置，确保各个服务器之间可以无密码互相通信。这一步至关重要，因为它确保了后续的并行任务可以跨服务器执行，而无需手动输入密码。  4.安装并配置MPI：在每台服务器上安装MPI相关的依赖包，并配置环境变量（如MPI\_ROOT等），确保MPI的相关命令可以在所有节点上正常运行。MPI负责管理集群中的并行计算和任务分发。  5.主机配置文件设置：在集群主节点上创建主机配置文件，定义各个服务器及其处理器核的数量，这个文件会在运行MPI程序时使用，确保程序可以正确分配任务到不同的节点。  6.同步程序文件：通过scp命令将主节点的程序文件和配置文件复制到其他节点，确保所有节点上都有相同的程序文件和配置。  7.运行并行程序：通过mpirun命令在集群上运行并行程序，指定使用的进程数和配置文件，集群中的多个节点会协同处理任务，实现并行计算。  **思考题2：集群之间如果彼此不配置信任秘钥，程序能否正常运行？**  如果集群之间不配置信任秘钥，程序通常不能正常运行。并行计算依赖于各个节点之间的通信，而MPI通过SSH协议在节点之间传递消息。如果没有配置免密登录，每次节点之间通信时都会要求手动输入密码，无法自动化执行分布式计算任务，从而导致程序无法正确运行。因此，配置信任秘钥（免密登录）是保证集群中并行程序能够顺利执行的关键步骤。  **思考题3：hello-world程序多次运行，结果一样吗？**  在正常情况下，hello-world程序多次运行时，结果是一样的。这是因为hello-world程序本质上是一个简单的并行程序，它的输出只是每个处理节点的ID、进程编号以及总进程数。这些信息在每次运行时都是固定的，因此输出也应该是相同的，除非发生了意外情况（如网络故障或节点故障）。  **思考题4：PI并行代码，增加更多的进程数（超过每个机器的CPU核数），性能如何变化？**  当并行代码的进程数增加，尤其是超过每台机器的CPU核数时，程序的性能可能不会继续提升，甚至可能下降。这是因为：  1.上下文切换开销：当进程数超过CPU核数时，操作系统需要在多个进程之间切换，而每次切换都会带来额外的开销，增加了程序的运行时间。  2.资源竞争：过多的进程会导致CPU、内存、I/O等资源的竞争，进程之间的资源争夺会影响并行程序的效率。  3.通信开销：在并行计算中，进程之间需要通过网络通信进行数据交换。如果进程数过多，通信开销会显著增加，反而可能导致并行效率降低。  因此，增加进程数能够提升性能的前提是进程数不要超过机器的物理核数。当进程数超过核数时，性能可能会出现瓶颈甚至下降。 | | | | | | | | | | | |
| **五、实验心得体会** | | | | | | | | | | | |
| 1. 云平台的应用与优势：  在本实验中，我使用了华为云的弹性云服务器（ECS）来搭建并行计算环境。云平台的优势在于灵活的资源配置和按需扩展能力，让我们能够快速获得所需的计算资源，无需自己配置物理服务器。这让我更加意识到云计算在实际并行计算中的应用场景和潜力。  2. 并行环境的搭建细节：  通过本次实验，我更加清晰地理解了搭建并行计算环境所需的步骤和细节。特别是SSH免密登录配置和MPI安装配置环节，使我认识到集群中节点间通信的核心在于保证各节点的互联互通和信任关系，免密登录至关重要，否则集群运行将受阻。  3. MPI并行程序的实现与优化：  实验中，我编写并运行了基于MPI的Hello World程序和PI计算程序。这不仅让我掌握了MPI并行程序的编写方法，还通过多次运行和调试过程，了解到如何根据节点数量合理分配任务，以提高程序的并行效率。我深刻体会到并行计算不仅在硬件资源上有要求，还需要对算法和程序进行优化，以减少通信开销并提高负载均衡性。  4. 并行计算的效率提升与瓶颈：  实验中，通过对比串行程序与并行程序在不同处理器数量下的运行时间，我真实地体验到了并行计算在大规模任务中的效率提升。然而，随着进程数的增加，尤其是超过机器的物理核数后，性能提升开始变得有限，甚至出现下降。这让我认识到并行计算的瓶颈不仅来自计算资源，还包括网络通信、上下文切换等其他开销。因此，在实际应用中，合理的并行任务分配和资源使用非常关键。 | | | | | | | | | | | |
| 指导教师评语：  日期： | | | | | | | | | | | |