Part 1: The Head Node (Lenovo X1 Carbon)

Role: Gateway, Scheduler, Login Node, Remote Access Point.
OS: Ubuntu Server 24.04 LTS.

1. BIOS Settings (Restart > F1)

- **Config > Network > Wake on LAN: Enabled**.
- **Config > Power > Sleep State: Linux (S3)** (Crucial for stability).
- **Config > Thunderbolt 3: Security Level: No Security** (Ensures Ethernet dongle boots instantly).
- **Hyperthreading: ENABLED** (Keep this ON for multitasking).

2. OS Installation & User Setup (Crucial)

- **Install:** Ubuntu Server 24.04 (Minimized).
- **Create Default User:** The installer will force you to create a user (e.g., `admin`). This user will get `UID 1000`. **Do not use this user for the cluster.**
- **Create the Cluster User (UID 1100):**
  After installation, log in and run:

```
sudo useradd -u 1100 -m -s /bin/bash student
sudo passwd student  # Set a password
sudo usermod -aG sudo student  # Give sudo access
```

Now, log out and log back in as `student`.

3. Network Configuration (Static IP)

- Edit Netplan: `sudo nano /etc/netplan/01-netcfg.yaml`

```
network:
  version: 2
  renderer: networkd
  ethernets:
    # WiFi (Internet) - Fast DNS
    wlan0:
      dhcp4: true
      nameservers:
        addresses: [1.1.1.1, 8.8.8.8]
    # Ethernet (Cluster LAN) - Static Gateway IP
    eth0:
      addresses: [10.0.0.1/24]
      # NO Gateway, NO DNS. This is strictly local.
```

Apply: `sudo netplan apply`

4. Enable NAT (Internet Sharing)

- **Forwarding:** `sudo sysctl -w net.ipv4.ip_forward=1`
- **Routing Rules:**

```
sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
sudo iptables -A FORWARD -i wlan0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o wlan0 -j ACCEPT
```

- **Persist:** `sudo apt install iptables-persistent`

5. Remote Access

- **Tailscale:** `curl -fsSL https://tailscale.com/install.sh | sh` -> `sudo tailscale up`
- **NoMachine:** Download `.deb` -> `sudo dpkg -i nomachine*.deb`

Part 2: The Compute Nodes (Lenovo P340s)

Role: Raw Number Crunching.
OS: Rocky Linux 9 Minimal.

1. BIOS Settings (Restart > F1)

- **Power > Intelligent Cooling: Performance** (Max Fan).
- **Devices > Video Setup > IGD Memory: 32MB** (Minimum).
- **Advanced > CPU Setup > Hyperthreading: DISABLED** (Critical for AVX2 speed).
- **Power > After Power Loss: Power On**.

2. Installation & User (Matching UID)

- **Install:** Rocky Linux 9 Minimal.
- **User Creation:** Create user `student`.
- **IMPORTANT:** Click **"Advanced"** and manually set **UID to 1100**.
- *This now perfectly matches the Head Node user.*
- **Network (In Installer):**
- **IPv4:** Manual -> `10.0.0.2` (Node 1), `10.0.0.3` (Node 2), `10.0.0.4` (Node 3).
- **Gateway:** `10.0.0.1`
- **DNS:** `1.1.1.1, 8.8.8.8`

3. Post-Install Speed Tuning

- **Disable Firewall & SELinux:**

```
sudo systemctl disable --now firewalld
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
```

Part 3: The Cluster Fabric (Linking Them)

1. The "Internal DNS" (`/etc/hosts`)

- Edit `/etc/hosts` on **ALL 4 NODES**. Add:

```
10.0.0.1    headnode
10.0.0.2    node01
10.0.0.3    node02
10.0.0.4    node03
```

2. Passwordless SSH

- **On Head Node (as user `student`):**

```
ssh-keygen -t rsa
ssh-copy-id node01
ssh-copy-id node02
ssh-copy-id node03
```

3. Shared Storage (NFS)

- **Head Node (Server):**

```
sudo apt install nfs-kernel-server
sudo mkdir -p /mirror
sudo chown 1100:1100 /mirror
echo "/mirror 10.0.0.0/24(rw,sync,no_root_squash)" | sudo tee -a /etc/exports
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

- **Compute Nodes (Clients):**

```
sudo dnf install nfs-utils
sudo mkdir -p /mirror
echo "headnode:/mirror /mirror nfs defaults 0 0" | sudo tee -a /etc/fstab
sudo mount -a
```

Part 4: The Brains (Slurm Scheduler)

1. Authentication (Munge)

- **Install:** `sudo apt install munge` (Head) / `sudo dnf install munge` (Compute).
- **Key Sync:** Copy `/etc/munge/munge.key` from Head to all Compute Nodes.
- **Permissions:** `sudo chown munge:munge /etc/munge/munge.key` (All nodes).
- **Start:** `sudo systemctl enable --now munge` (All nodes).

2. Slurm Controller (Head Node)

- **Install:** `sudo apt install slurmctld`
- **Config (`/etc/slurm/slurm.conf`):**
  Note: ClusterName updated to 'threeidiots'.

```
ClusterName=threeidiots
SlurmctldHost=headnode
MungeSocketPath=/var/run/munge/munge.socket.2
SlurmUser=slurm
NodeName=node[01-03] CPUs=8 RealMemory=16000 State=UNKNOWN
PartitionName=debug Nodes=node[01-03] Default=YES MaxTime=INFINITE State=UP
```

- **Start:** `sudo systemctl enable --now slurmctld`

3. Slurm Client (Compute Nodes)

- **Install:** `sudo dnf install slurm-slurmd`
- **Config:** Copy `slurm.conf` from Head Node to `/etc/slurm/` .
- **Start:** `sudo systemctl enable --now slurmd`

Part 5: The Speed (Intel OneAPI)

1. Install Packages

- **Head Node:** Follow Intel docs to add `apt` repo -> `sudo apt install intel-basekit intel-hpckit` .
- **Compute Nodes:** Follow Intel docs to add `yum` repo ->
  `sudo dnf install intel-basekit intel-hpckit` .

2. Activate Environment (Crucial)

- Add to `~/.bashrc` on **ALL 4 NODES**:

```
source /opt/intel/oneapi/setvars.sh > /dev/null 2>&1
```

Part 6: Verification & Launch

1. Check Node Status

- On Head Node: `sinfo`
- **Success:** `PARTITION: debug | NODES: 3 | STATE: idle`

2. The "Three Idiots" Test Script

Create `idiot_test.sh` in `/mirror`:

```
#!/bin/bash
#SBATCH --job-name=idiot_check
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=1
#SBATCH --output=result.out

# Load Intel Environment
source /opt/intel/oneapi/setvars.sh

# Run MPI Hostname Check
echo "Reporting from cluster: threeidiots"
mpirun -n 3 hostname
```

- **Submit:** `sbatch idiot_test.sh`
- **Verify:** `cat result.out` (Should show `node01`, `node02`, `node03`).

Part 7: Benchmarking (Proving It)

1. The "Latency" Test (OSU Micro-Benchmarks)

Goal: Prove your network is optimized.

- **Download & Compile (Run on Head Node):**

```
wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-7.3.tar.gz
tar xzf osu-micro-benchmarks-7.3.tar.gz
cd osu-micro-benchmarks-7.3
./configure CC=icc CXX=icpc
make
```

- **Run:**

```
mpirun -n 2 -hosts node01,node02 ./mpi/pt2pt/osu_latency
```

- **Result:** Look for the **0 Byte** latency. Lower is better.

2. The "Raw Compute" Test (DGEMM)

Goal: Prove AVX2 optimization.

- Create `speed_test.c` in `/mirror`:

```
#include <stdio.h>
#include <mkl.h>
#define SIZE 2048
int main() {
    double *A, *B, *C;
    A = (double *)mkl_malloc(SIZE*SIZE*sizeof(double), 64);
    B = (double *)mkl_malloc(SIZE*SIZE*sizeof(double), 64);
    C = (double *)mkl_malloc(SIZE*SIZE*sizeof(double), 64);
    // Fill A and B...
    for(int i=0; i<SIZE*SIZE; i++) { A[i]=1.0; B[i]=2.0; }

    double s_initial = dsecnd();
    cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, SIZE, SIZE, SIZE, 1.0, A, SIZE, B, SI
    double s_elapsed = dsecnd() - s_initial;

    printf("Matrix Mul Time: %.5f seconds\n", s_elapsed);
    mkl_free(A); mkl_free(B); mkl_free(C);
}
```

- **Compile (The Cheat Code):**

```
icx -O3 -xHost -qmkl speed_test.c -o speed_test
```

- **Run:** `srun -N 3 ./speed_test`
- **Result:** Compare this number to a generic compile to show your speed gains.