Xilinx Vivado Slurm Cluster Configuration

Conversation Export • Technical Reference Guide

USER

I am building a Xilinx Vivado slurm cluster. I would like users to be able to submit their jobs and have it run on an arbitrary version of Vivado, but I think I need to use LMOD to configure the various tool chains. Can you explain how I could do that?

ASSISTANT

To allow users to select and run arbitrary versions of Xilinx/AMD Vivado on a Slurm cluster, you need three core components working together: **shared software installation**, **Lmod modulefiles**, and **Slurm environment initialization**.

1. Cluster Prerequisites & Storage Layout

All Slurm compute nodes (and head nodes) must see the exact same file system paths.

- **Shared Directory Structure:** Install all Vivado versions onto shared storage (e.g., NFS, Lustre, or CephFS) mounted identically on all nodes.
  - Example tool directory: `/apps/xilinx/Vivado/2022.2`, `/apps/xilinx/Vivado/2023.2`, etc.
  - Example modulefiles directory: `/apps/modulefiles/Core/vivado/`
- **License Server Configuration:** Ensure floating licenses are configured via a centralized license server (`2100@lic-server.example.com`) reachable by all compute nodes.

2. Writing the Lmod Lua Modulefile

Lmod uses Lua scripts to manipulate environment variables cleanly (`setenv`, `prepend_path`). While Vivado provides a `settings64.sh` file, hard-coding environment variables in a pure Lua modulefile is faster and avoids subshell execution overhead during Slurm job scheduling.

Create directory: `/apps/modulefiles/Core/vivado/`
Create module file: `/apps/modulefiles/Core/vivado/2023.2.lua`

```
-- /apps/modulefiles/Core/vivado/2023.2.lua
help([[
Loads Xilinx / AMD Vivado Design Suite 2023.2.
]])

whatis("Name: Xilinx Vivado")
whatis("Version: 2023.2")
whatis("Category: FPGA EDA Tools")

local version = "2023.2"
local base    = "/apps/xilinx/Vivado/" .. version

-- Set Core Vivado Environment Variables
setenv("XILINX_VIVADO", base)
setenv("XILINX_HLS",    base .. "/Xilinx_HLS") -- required if Vitis/HLS tools are included

-- Configure Executable Path and Libraries
prepend_path("PATH",            base .. "/bin")
prepend_path("LD_LIBRARY_PATH", base .. "/lib/lnx64.o")

-- Configure License Server (Adjust port and hostname for your site)
setenv("XILINXD_LICENSE_FILE", "2100@lic-server.example.com")
-- setenv("LM_LICENSE_FILE",    "2100@lic-server.example.com")

-- Optional: Prevent Vivado from spawning excessive GUI/web telemetry threads on compute
nodes
setenv("XIL_SILENT_RUN", "1")
```

3. Configuring Lmod for Slurm Compute Nodes

Slurm batch jobs execute in non-interactive subshells by default. Ensure Lmod is initialized automatically on compute nodes:

1. **System-wide Profile:** Make sure `/etc/profile.d/z00_lmod.sh` (or `/etc/profile.d/lmod.sh`) is sourced globally on all compute nodes.
2. **Default MODULEPATH:** Export your custom module directory in the global environment or inside `/etc/lmod/modulespath`:

   ```
   echo "/apps/modulefiles/Core" >> /etc/lmod/modulespath
   ```

3. **Slurm Job Environment Export:** Verify `/etc/slurm/slurm.conf` does not aggressively strip user environments unless intended. Ensure `PropagateResourceLimits=ALL` or standard environment propagation is active.

4. Example Slurm Job Submission Script

Users can now specify the exact Vivado version in their Slurm batch submission script using standard `module load` commands:

```
#!/bin/bash
#SBATCH --job-name=vivado_synth
#SBATCH --output=vivado_%j.log
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=32G
#SBATCH --time=04:00:00

# 1. Initialize Lmod (if non-login shell)
source /etc/profile.d/lmod.sh

# 2. Load the requested Vivado version
module purge
module load vivado/2023.2

# 3. Verify tool version in log
vivado -version

# 4. Execute Vivado in non-interactive batch mode
vivado -mode batch -source run_synthesis.tcl -tclargs --jobs ${SLURM_CPUS_PER_TASK}
```

5. Helpful Cluster Management Practices

- **Set a Default Version:** Set a symbolic link or Lmod default so `module load vivado` picks your cluster's stable baseline:

  ```
  cd /apps/modulefiles/Core/vivado
  ln -s 2023.2.lua default
  ```

- **Slurm Generic Resources (GRES) for Licenses:** If you have limited Vivado license tokens, consider tracking licenses in `slurm.conf` via `Licenses=vivado_keys:10` so Slurm queues jobs when license limits are reached rather than having Vivado jobs fail mid-run.

USER

If the user has a project checked out on their local machine, is there a way to submit the project over the network or is it required that the project exist on the nfs /home share in the SLURM cluster?

ASSISTANT

It is **not strictly required** for the project to pre-exist on the cluster's NFS share, but because Slurm compute nodes can only access local/networked storage attached to the cluster, the project files **must land on the cluster storage before execution begins**.

1. Direct Transfer & Remote Submit (Best User Experience)

Option A: Standard scp / rsync Wrapper Script

Write a small helper script (e.g., `cluster-submit`) on local machines that syncs the workspace to the cluster's NFS `/home` or `/scratch` directory and executes `sbatch` via SSH.

```
#!/bin/bash
# Local wrapper script: cluster-submit
REMOTE_USER="alice"
CLUSTER_HOST="cluster-login.example.com"
REMOTE_DIR="/home/${REMOTE_USER}/jobs/$(basename $PWD)_$1"

# 1. Create remote job directory on NFS
ssh ${REMOTE_USER}@${CLUSTER_HOST} "mkdir -p ${REMOTE_DIR}"

# 2. Sync local project files (excluding large build artifacts/git metadata)
rsync -avz --exclude='.git/' --exclude='*.runs/' ./ ${REMOTE_USER}@${CLUSTER_HOST}:$
{REMOTE_DIR}/

# 3. Submit job on remote cluster via SSH
ssh ${REMOTE_USER}@${CLUSTER_HOST} "cd ${REMOTE_DIR} && sbatch run_vivado.sh"
```

2. Recommended Workflow Pattern for Vivado

Vivado generates large temporary build directories (`.runs` , `.Xil` , IP caches) during synthesis and implementation. Pushing all that intermediate data across NFS during runtime can severely degrade cluster I/O.

The Golden Pattern for Vivado on Slurm:

1. Local Workstation → Push lightweight sources (`.v` , `.sv` , `.tcl`) via rsync.
2. NFS Shared Storage → Slurm assigns compute node.
3. Compute Node Local SSD (`/tmp` or `/scratch_local`) → Copy sources to fast local node SSD and run Vivado.
4. NFS Shared Storage → Copy back ONLY final bitstreams/reports.

Summary Strategy Matrix

Approach	Setup Effort	Performance	Best Used For
Direct NFS (/home)	None	Low (NFS locking/IOPS bottleneck)	Quick tests, small projects
rsync + Local Scratch (/tmp)	Low (Basic wrapper)	**Very High** (Local SSD IOPS)	**Production FPGA Builds**
Git / CI/CD Pipeline	Medium	High	Automated build servers

USER

Can you email this chat to me at sbruno1@unm.edu?

ASSISTANT

I don't have the capability to send emails directly, so I can't send this chat to **sbruno1@unm.edu**. However, you can easily save or share this conversation yourself via copy/paste or the share link.