在科学计算中使用 NIX

实战经验与挑战

陈浩南 – 2025-08-09

目录

自我介绍

关于我自己

- 现居厦门，学生，搞物理的。
- 在联邦宇宙活动。
- 希望和大家成为朋友！

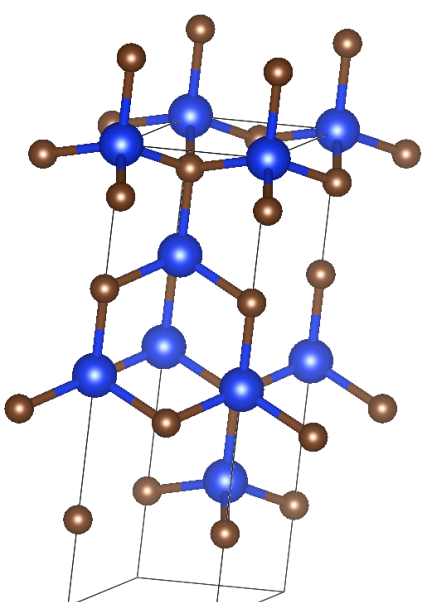
扫码看主页

使用过的发行版

- 桌面：2018 年起主力使用 Linux，Deepin → Arch → Gentoo → NixOS (2023-05-28)
- 个人服务器：OpenWRT → Ubuntu → Arch → Gentoo → NixOS
  - 著名作品：xmurp-ua
- 科学计算：2020 年硕士入组，Gentoo + Ubuntu → NixOS (2023-09)

科研方向

- 组内：宽禁带半导体（氮化物、碳化硅等）。
- 个人：第一性原理计算，兼职管理服务器。
  - **第一性原理**计算：仅依赖最基础的物理原理（量子力学），原则上不引入经验值。



$$\hat{H}\varphi=\hat{E}\varphi \longrightarrow$$

稳定结构?

电学/热学/光学性质?

缺陷/杂质?

科学计算在实践中面临的困难

科学计算特点

（*更专业报告见明天的《高性能计算》@VickieGPT。）

- 相比于爱好者折腾/软件开发的环境，科学计算的环境有**两个特别需求**：
  - 计算量大 → **高性能**。
  - 使用者缺乏 CS 技能 → **易于使用和维护**。

矛盾：**高性能** → 复杂的软硬件配置 → **难以实现** **易于使用和维护**。

科学计算现状

高性能 和 **易于使用和维护** 的矛盾，导致充满妥协：

- **不存在**的包管理：手动收集依赖、修改 Makefile
- **老旧**的基础环境：10+ 年前
- **随意**的编程习惯：无视标准、能用就行
- **闭源**的编译器：Intel (OneAPI)，Nvidia (HPC SDK)
- **混乱**的用户权限：账户多人共用

（*仅身边统计学，没有 diss 别的组的意思）

使用 NIX 搭建科学计算环境

Nix / NixOS 的优势

- 编译软件超方便
  - 超多软件包：Nix: 100k+, Arch: 70k+, Ubuntu: 39k+, CentOS: 2k+
  - 方便打新包/打补丁。
- 配置服务不易错
  - SLURM、NFS 等都有现成模块。
  - 多机器/服务/用户共用配置。

Nix 的优势：将 **编译/配置过程** 抽象成 **可复现、可版本管理的代码**，从而重复使用，用 **机器**的自动化运作代替**人力**的劳动。

用 Nix 解决问题

- **不存在**的包管理：用 Nix 提供依赖/打包
- **老旧**的基础环境：关我 `/nix/store` 什么事?
- **随意**的编程习惯：Nix 保证如果今天能用，那么一年后也能用。
- **闭源**的编译器：打包进 Nix 了
- **混乱**的用户权限：系统级环境/复用 hm 配置

配置细节

- 组内小集群：NixOS。
- 超算（无 root 权限）：使用 Nix 安装小工具（e.g. gnuplot）。
  - 编译器、MPI、VASP 等已经使用传统方式安装，下一次更新时再考虑使用 Nix。

配置细节

编译器、MPI

- Intel 编译器（OneAPI）：
  - 必需，因为对一些软件有优化，比 gfortran 更宽容。
  - 在 bscpkgs 基础上修改。
- Nvidia 编译器（HPC SDK）：
  - 必需，提供 nvfortran、QD 库等。
  - 自行打包（`nvhpcPackages.stdenv`）。
- OpenMPI：
  - 需要修改才能与闭源编译器兼容。（还没有提 pr）
  - NVIDIA 需要使用修改过的源代码。
- 提供 overlay 及示例：`github:CHN-beta/nix-hpc-test`。
- 仍有需要改进的地方（详见下文）。

配置细节

NixOS 上的其它软件和服务

- 文件系统：
  - NFS 共享 `/home`；Btrfs 透明压缩 RAID1。
  - hm 和 impermanence 需要单独处理（详见下文）。
- 队列系统：
  - 使用 SLURM。
  - MPI 尽量使用 OpenMPI、尽量用 srun 启动。
  - 提供 TUI 代替 sbatch。
- native 优化：
  - 针对硬件优化（设置 `hostPlatform.gcc.arch`、`oneapiArch`、`nvhpcArch`）。
  - nixpkgs 半年更新一次，期间仅 cherry-pick 个别提交。平衡“新”、“稳定”与编译整个系统需要的代价。

配置细节

超算上使用 Nix

- 没有 user namespace。proot 影响性能。
- 使用可读写的 store 目录。编译机上建立相同的目录，编译好再上传。
  - 不能设置 `real` 参数指向 `/nix/store`，否则会破坏编译机的 Nix 数据库。

```
# this will break the build machine's nix database
sudo nix build --store 'local?store=/data/gpfs01/jykang/.nix/
store&real=/nix/store' .#jykang
# this is safe
sudo nix build --store 'local?store=/data/gpfs01/jykang/.nix/
store&state=/data/gpfs01/jykang/.nix/state&log=/data/gpfs01/
jykang/.nix/log' .#jykang
```

一些开放性问题

闭源编译器打包

能工作但质量不高。

- 依赖 gcc：
  - 在非 FHS 环境下，用参数/环境变量指定 gcc。
  - gcc/gfortran 分包/wrap，如何合并得到“完整”的 gcc?
- 支持参数与 gcc 不同。需要仔细调整。

Call for help：很需要了解 stdenv 的同学的帮忙。

impermanence / hm on NFS

- `.bashrc` 等需禁用软连接、改为 bind mount。
- 父目录的父目录的所有者会出错（bug?）

构建时 FHS 环境（FHSStdenv）

- 在**构建时**提供 FHS 环境，以快速（但低质量地）打包上游提供了 installer 的软件?
- 这不美观，但很有用!
- 目前不能复用 stdenv 中 setup hook / xxxPhase。能否编写 `FHSStdenv`?

构建时 FHS 环境（FHSStdenv）

一个例子（曾经的 NVIDIA HPC SDK 打包）

```
let
  builder = buildFHSEnv
    { extraBwrapArgs = [ "--bind" "$out" "$out" ]; };
  buildScript = writeShellScript "build.sh" ''xxxxx'';
in stdenvNoCC.mkDerivation
{
  pname = "nvhpc";
  installPhase =
  ''
    mkdir -p $out
    ${builder}/bin/builder ${buildScript}
  '';
}
```

总结

理想 vs 现实

- 理想：
  - **软件开发者**使用 Nix，我们一键安装/配置；
  - **论文作者**使用 Nix，我们一键复现，略作修改继续研究；
  - **超算管理员**使用 Nix，必要时一键回滚，方便互相参考。
- 现实：
  - 在现实中没有遇到除我以外的科研人员使用 Nix。
  - 网上有一些。Barcelona Supercomputing Center（巴塞罗那，西班牙）开源了 bscpkgs，其中包含 Intel 闭源编译器。

展望

我认为 Nix 在科学计算中有很大的潜力，但：

- 使用不够广泛；我也不知道怎么办。
  - 明天上午《如何在公司里把 Nix 安利给同事》@Noa Virellia
- 缺少一些“垫脚石”，还需要开源社区努力（包括我在内）。