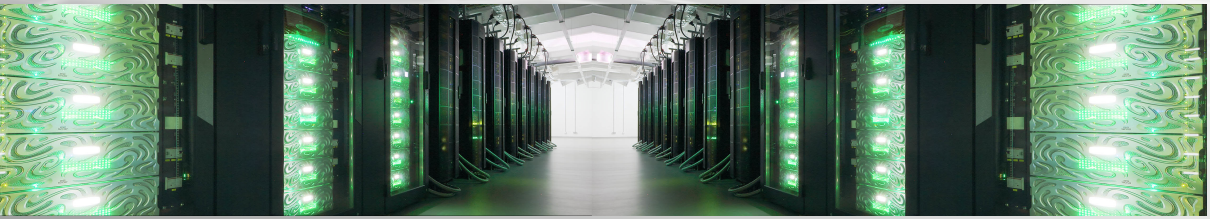
GWDG
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Are Nix or Guix the Answer to Software Provisioning on HPC Clusters?

An Analysis of Software Provisioning Technologies

Denzel Wilson Thoppil(denzel.wilsonthoppil@gwdg.de)
Prof. Dr. Julian Kunkel (julian.kunkel@gwdg.de)
Dr. Patrick Höhn (patrick.hoehn@gwdg.de)



November 13, 2026

Table of Contents

1 Introduction

2 Environment Validation

3 Comprehensive Analysis

4 Conclusion

About GWDG

- Research IT service provider for Max Planck Society and University of Göttingen
- National research infrastructure roles:
  - National HPC center within NHR consortium (NHR-Nord@Göttingen)
  - National AI service center (KISSKI)
  - HPC service partner for DLR and Max Planck institutes
  - Participant in NFDI (National Research Data Infrastructure)
- Serves 86 Max Planck institutes and 30,000+ researchers across diverse domains
- Infrastructure: 2000+ heterogeneous nodes supporting varied research workflows

The Challenge

- Software provisioning on HPC systems is challenging
- Must fulfill potentially conflicting needs of many users
- Ordinary package managers (apt, dnf) cannot provide solutions
  - Cannot handle multiple versions simultaneously
  - Lack reproducibility guarantees
  - System-wide modifications cause conflicts

Research Goals

- Many technologies exist to manage software in HPC environments
- Investigate five current technologies:
  - EasyBuild, Conda, Spack, Nix and Guix
- Find their benefits and drawbacks
- **Today's focus**: Are Nix or Guix the answer?

GWDG HPC Infrastructure

- Multiple cluster islands grouped by purpose and hardware generation:
  - **SCC** (Scientific Compute Cluster): General research computing
  - **NHR**: National HPC resources for German universities
  - **Emmy**: General CPU nodes (Intel Skylake/Cascade Lake/Sapphire Rapids)
  - **Grete**: GPU-accelerated nodes (Nvidia V100/A100/H100)
  - Institution-specific islands: CIDBN, FG, SOE, and others
- Heterogeneous hardware across generations and purposes
- **Validation scope**: Tested software package coverage on Emmy and Grete clusters

Coverage Results

	GRETE			EMMY		
Package Manager	**Total**	**Open Source**	**Proprietary**	**Total**	**Open Source**	**Proprietary**
Spack (current)	163	159	4	104	97	7
EasyBuild	129	125	4	82	77	5
Conda	119	119	0	85	84	1
Nix	117	117	0	91	90	1
Guix	106	106	0	88	87	1

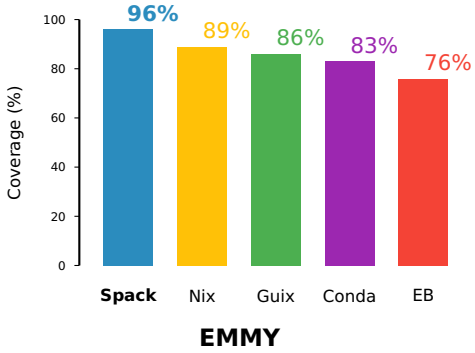
GRETE: 174 modules (166 open source, 8 proprietary)
EMMY: 115 modules (101 open source, 14 proprietary)

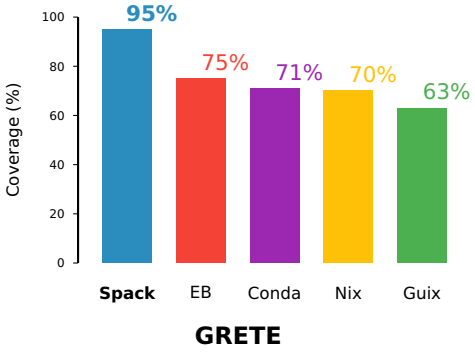
Package Examples: Universal Availability

Package	EB	Conda	Spack	Nix	Guix
Compilers & Languages					
GCC	■	■	■	■	■
LLVM	■	■	■	■	■
Python	■	■	■	■	■
Julia	■	■	■	■	■
Rust	■	■	■	■	■
Build Tools					
CMake	■	■	■	■	■
Git	■	■	■	■	■
Proprietary Software (Integration wrappers only)					
MATLAB	■		■		
ANSYS	■		■		

= Available in official repositories; Proprietary requires separate licensing

Final Scores





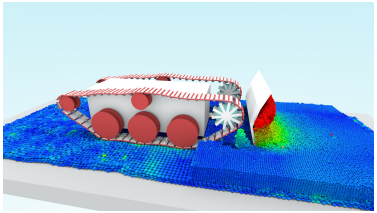
The Migration Question

What would switching to Nix/Guix actually cost?

- Coverage gaps revealed by validation:
  - Emmy: 7-10% gap
  - Grete: 25-32% gap
- Migration implications:
  - Operators must manually package missing software
  - Some HPC tools need porting (AMD compilers, performance analyzers)
  - Significant effort required
- **Question**: What benefits would justify this migration effort?

Case Studies

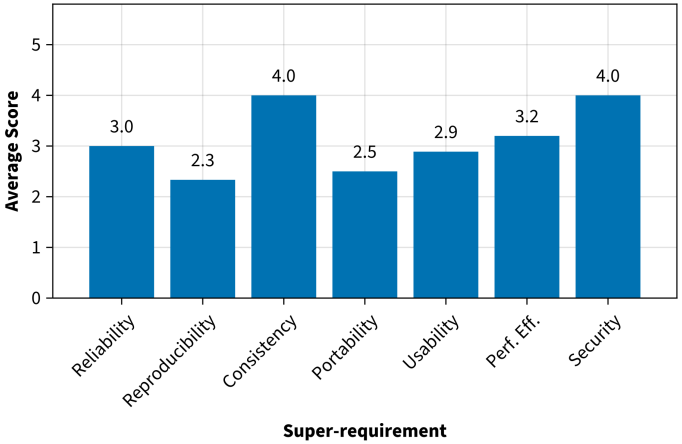
- Case studies conducted to analyze all five tools:
  1. Package Project Chrono library (complex C++ dependencies, configurable builds)
  2. Integrate into local user environments
- Evaluation metrics: reliability, reproducibility, configurability, usability



Comparison Matrix - Sample

Super-requirement	Sub-requirement	EasyBuild	Conda	Spack	Nix	Guix
Reliability	Run-time Stability	4	4	4	5	5
	Reliable Dependency Management	3	4	3	5	5
	Rollback Functionality	2	5	2	5	5
Reproducibility	Rebuildability	4	4	3	5	5
	Bitwise Reproducibility	2	2	2	4	5
	Long-Term Reproducibility	1	2	4	4	5
[...]						

EasyBuild


Average Score
0
1
2
3
4
5
3.0
2.3
4.0
2.5
2.9
3.2
4.0
Reliability
Reproducibility
Consistency
Portability
Usability
Perf. Eff.
Security
Super-requirement

EasyBuild: Key Findings

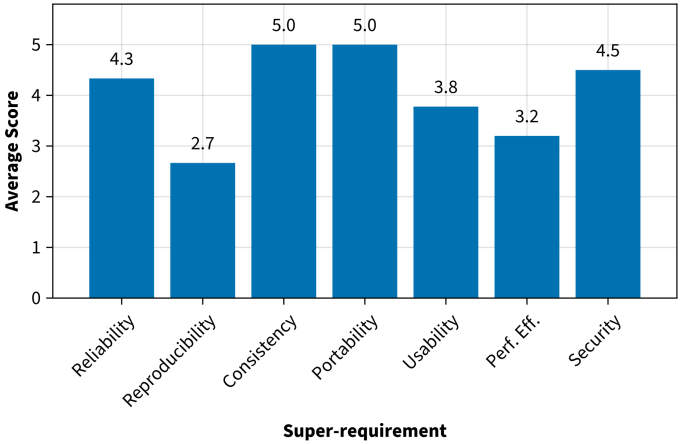
Strengths:

- Solid usability (simple provisioning, simple package definitions)
- Automatic hardware optimization

Weaknesses:

- Reliability issues (system-level dependencies, hard-coded configs)
- Limited functionality
- Poor storage efficiency
- Significant deployment time

Conda


Average Score
0
1
2
3
4
5
4.3
2.7
5.0
5.0
3.8
3.2
4.5
Reliability
Reproducibility
Consistency
Portability
Usability
Perf. Eff.
Security
Super-requirement

Conda: Key Findings

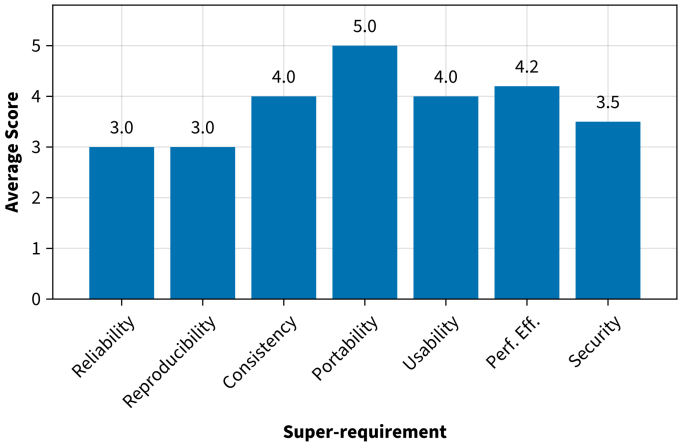
Strengths:

- Excellent portability (binary packaging, platform versatility)
- Broad environment support

Weaknesses:

- Performance limitations (hardware optimization complex)
- Complex package definitions (YAML + shell + batch files)

Spack


Average Score
0
1
2
3
4
5
3.0
3.0
4.0
5.0
4.0
4.2
3.5
Reliability
Reproducibility
Consistency
Portability
Usability
Perf. Eff.
Security
Super-requirement

Spack: Key Findings

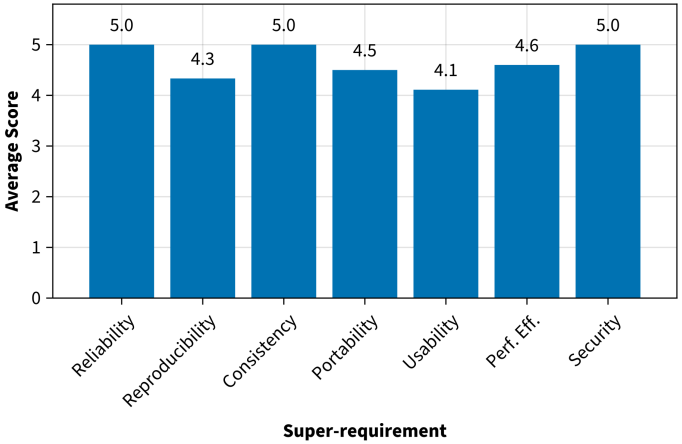
Strengths:

- Exceptional package definitions (Python-embedded DSL, simple, flexible, powerful)
- Performance-focused
- Solid environment support

Weaknesses:

- Reliability issues (system dependencies, Misconfigurations)
- High storage utilization

Nix


Average Score
0
1
2
3
4
5
5.0
4.3
5.0
4.5
4.1
4.6
5.0
Reliability
Reproducibility
Consistency
Portability
Usability
Perf. Eff.
Security
Super-requirement

Nix: Key Findings

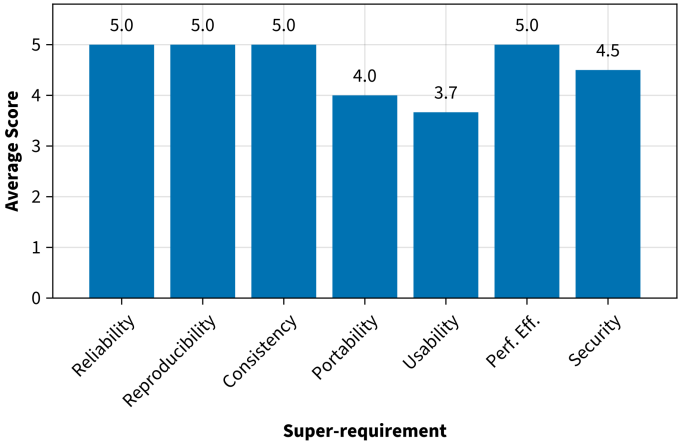
Strengths:

- Superior binary reproducibility
- Large generic software ecosystem
- Excellent storage efficiency
- Robust dependency management
- Security guarantees

Weaknesses:

- High complexity
  - Steep learning curve
  - Transitioning approaches
- Hardware optimization complex
  - Targets generic systems
  - Requires advanced knowledge
- Single version per channel

Guix


5.0
5.0
5.0
4.0
3.7
5.0
4.5
Average Score
0
1
2
3
4
5
Reliability
Reproducibility
Consistency
Portability
Usability
Perf. Eff.
Security
Super-requirement

Guix: Key Findings

Strengths:

- Similar to Nix
- Even better binary reproducibility
  - Software Heritage integration
- Advanced hardware optimization
- Provenance features

Weaknesses:

- Limited proprietary support
- Very steep learning curve
  - Uses Scheme dialect
- Limited platform coverage
- Smaller ecosystem
- Single version per channel

Key Findings from Validation

1. **Coverage gaps grow with cluster complexity**: EMMY 7-10% gap, GRETE 25-32% gap - larger clusters face bigger migration challenges
2. **Migration requires significant operator effort**: 12+ packages need manual porting including common HPC tools
3. **Binary reproducibility benefits serve minority of users**: 5% publish reproducible workflows, 95% expect `module load`
4. **Challenges with proprietary software**: Nix/Guix 0-7% vs Spack 50%

Recommendations

1. **Existing centers**: Continue with Spack/EasyBuild, migration challenges outweigh potential benefits
2. **New deployments**: Nix/Guix for small teams, Spack/EasyBuild for broad coverage
3. **Pragmatic path**: Hybrid approach - Spack/EasyBuild as primary, with selective Nix/Guix for reproducibility

Good Fit for Nix/Guix

Small teams, binary reproducibility-critical work, experts

Poor Fit

Large user base, traditional workflows, limited resources

Summary

- Thesis validated: Nix (4.47) and Guix (4.32) provide superior binary reproducibility - technical advantages are real
- Operator reality: 25-32% coverage gaps on larger clusters, most users don’t need binary reproducibility features
- **Are Nix/Guix THE answer?** Context-dependent - YES for small reproducibility-focused teams, NOT YET for general HPC centers
- Realistic path: Hybrid approach balances breadth (Spack/EasyBuild) with binary reproducibility (selective Nix/Guix)

Acknowledgments

- Henrik Jonathan Seeliger's master's thesis
- Survey participants from HPC community
- Development communities of all five tools

E-mail: denzel.wilsonthoppil@gwdg.de

Matrix Server: @dwilson:gwdg.de

