From Punch Cards to AI-Powered HPC: A 40-Year Journey in Computational Chemistry Infrastructure

Krishna K. Govender[1,2] and Hendrik G. Kruger[3]*

[1]*Department of Chemical Sciences, University of Johannesburg, Johannesburg, South Africa.*
[2]*National Institute for Theoretical and Computational Sciences (NITheCS), South Africa.*
[3]*Catalysis and Peptide Research Unit, School of Health Sciences, University of KwaZulu-Natal, Durban, South Africa.*

ABSTRACT

This paper chronicles the collaboration between a senior computational chemist—spanning four decades of computing evolution from 1983 punch cards to 2025 artificial intelligence (AI)-assisted infrastructure—and a high-performance computing (HPC) expert, mediated by AI tools, culminating in the successful deployment of a multi-node research computing cluster for molecular modelling and drug design. An initial Gaussian 16 software request expanded into a comprehensive cluster implementation using Rocky Linux 9, SLURM workload management, parallel filesystems, and two key computational chemistry packages (Amber24 and Gaussian 16), with ORCA, GAMESS-US, Psi4, NWChem, and CP2K identified as suitable candidates for future installation; the cluster optimises mixed graphics processing unit (GPU) architectures (NVIDIA RTX A4000/RTX 4060), a common reality in resource-limited laboratories. A structured multi-AI collaborative methodology—utilising Claude AI for documentation, Grok for alternative perspectives, and DeepSeek for technical verification—resolved approximately 150 distinct technical challenges with an 85% first-attempt success rate, completing deployment in approximately 6 months compared to the estimated 12–18 months required by traditional approaches. Benchmarks confirmed 99%+ Amber24 test suite validation, a 20× GPU speedup for large molecular systems (~1 million atoms), and 17% GPU acceleration for quantum chemistry calculations. The AI approach thrived via an iterative model of problem-solving, explanation, and reasoning, with human-led execution, validation, and adaptation essential throughout. All Warewulf cluster installation manuals are provided as Supplementary Material (S1–S27), enabling any researcher to assemble a comparable system with modest effort and a proficient AI collaborator, thereby removing barriers to HPC adoption and heralding a paradigm shift in scientific knowledge transfer for resource-limited institutions.

KEYWORDS

High-Performance Computing, Computational Chemistry, AI-Assisted Infrastructure, Cluster Computing, Knowledge Transfer, SLURM Workload Manager, Scientific Computing, Human-AI Collaboration, HPC Democratization, Intergenerational Learning.

Received: 24 October 2025 Revised: 25 March 2026 Accepted: 31 March 2026

This work is dedicated to the memory of Professor T.A. (Tony) Ford, a valued friend and colleague at the University of KwaZulu-Natal.

INTRODUCTION: FOUR DECADES OF COMPUTATIONAL EVOLUTION

My journey from using punch cards in 1983 to employing AI in 2025 for building high performance computing (HPC) clusters represents both personal evolution and the broader transformation of computational chemistry infrastructure. During my first year of BSc studies at Potchefstroom University in South Africa, we still used punch cards with optical readers to feed Fortran code to mainframe computers. This seemingly primitive approach taught fundamental programming discipline, every character mattered, debugging required careful thought before execution, and computational resources were precious.

During my PhD studies (1992–96), I became interested in computational chemistry, starting with HyperChem software[1] for organic cage chemistry applications and began a long collaboration with Prof. Tony Ford (UKZN) on molecular modeling in 2002.

My first encounter with artificial intelligence (AI) was in 1997, examining a PhD thesis on "Artificial neural networks for the classification of Meliaceae extractives." The next significant AI interaction came through homology modeling applications in 2012, particularly for tuberculosis drug target research.[2,3] These experiences established a foundation for understanding how AI tools can enhance research capabilities.

Our research group follows the philosophy of training leaders for the future by teaching skills much broader than chemistry alone. Synthesis students learn to maintain mass spectrometers and HPLC systems; computational chemistry students master Unix commands, Linux installation, and software compilation. Initially, we taught students to set up Rocks Clusters,[4] but when this system was discontinued, we needed an alternative.

The primary computational chemistry software packages chosen for this deployment have extensive development histories. Amber (Assisted Model Building with Energy Refinement) has evolved over four decades into the current Amber24[5], which provides over 160 production-quality programmes spanning molecular dynamics simulation, system preparation, force field parameterisation, and trajectory analysis. Similarly, Gaussian has been a cornerstone of quantum chemistry computation since the early 1970s, with Gaussian 16[6] representing the current production release and supporting a comprehensive suite of quantum mechanical methods—including density functional theory, post-Hartree–Fock approaches, and semi-empirical methods—with Linda parallel execution enabling distributed computation across multiple nodes.

Literature searches and discussions with HPC colleagues suggested Warewulf Cluster Management[7] as a promising solution. The primary challenge was the lack of specific, up-to-date, and context-sensitive guidance in available documentation and online forums for modern Warewulf versions and Rocky Linux 9. This led to exploring whether AI tools could assist with complex technical HPC deployments.

It was realized that an objective HPC expert will be required to do quality control on this project. Prof. Krishna Govender (UJ) kindly agreed to fulfil that role.

After experimentation, it appeared that a combination of Claude AI, Grok, and DeepSeek provided complementary strengths for technical problem-solving. The productive approach involved asking

*To whom correspondence should be addressed
Email: kruger@ukzn.ac.za

ISSN 1996-840X Online / South African Chemical Institute / http://saci.co.za/journal

https://doi.org/10.17159/0379-4350/2026/v80se1a06

the same question to each tool, then rotating responses between tools for consideration and modification. After 2–3 iterations, consensus solutions typically emerged. Typical questions addressed to the AI tools included: "How should SLURM be configured to support cgroup v1 on Rocky Linux 9?", "What is the correct compilation sequence for NCCL 2.22.3 with CUDA 12.4 and MPICH 4.2.3?", and "How should mixed GPU architectures with different VRAM capacities be handled in Warewulf overlay configuration?" These specific, context-rich technical queries proved most effective in generating actionable and reproducible solutions. The process is depicted in Figure 1.

The challenge of modern HPC infrastructure deployment represents a significant barrier for individual researchers. Traditional approaches require either substantial IT support or years of specialized learning. Commercial solutions often lack the flexibility needed for computational chemistry research, while open-source solutions demand extensive technical expertise. This creates a gap where researchers understand their computational needs but lack the infrastructure deployment capabilities to implement solutions effectively. This AI-collaborative approach enabled deployment of a complete 6-node HPC cluster featuring mixed GPU architectures (RTX A4000 and RTX 4060), achieving 99%+ test success rates for Amber24 molecular dynamics software and 17% GPU acceleration for Gaussian16 quantum chemistry calculations. The implementation was completed in approximately 6 months—significantly faster than the estimated 12–18 months using traditional approaches—while maintaining professional-grade reliability and comprehensive documentation.

The resulting infrastructure provides complete computational chemistry capabilities spanning system preparation (pdb4amber, MCPB.py for metalloproteins), molecular dynamics simulations (serial, MPI parallel, and NCCL multi-GPU), quantum chemistry calculations (Gaussian16 with Linda parallel execution), and intelligent resource management. The system automatically selects optimal hardware based on molecular system characteristics, with performance ranging from 50–300 ns/day for small proteins on RTX 4060 GPUs to specialized multi-GPU NCCL execution for systems exceeding 300,000 atoms.

This paper addresses the research question: Can AI collaboration democratize advanced technical deployment, enabling individual researchers to achieve sophisticated infrastructure implementations previously requiring specialized teams? We explore this through the lens of building a complete computational chemistry HPC cluster using multi-AI collaborative problem-solving.

LITERATURE REVIEW: AI-ASSISTED TECHNICAL PROBLEM SOLVING

Modern HPC cluster deployment involves complex interactions between hardware, operating systems, network management, job scheduling, and scientific software integration. Traditional deployment approaches fall into several categories: commercial turn-key solutions, institutional IT support, or individual technical implementation. Commercial solutions often provide limited customization and high costs, while institutional support may not align with specific research requirements or timelines.

Recent literature on AI applications in system administration shows promising developments in automated configuration management, error diagnosis, and deployment optimization.[8] However, most applications focus on large-scale enterprise environments rather than research computing needs.[9] The complexity of integrating specialized scientific software packages like Amber24 or Gaussian with cluster management systems represents a particular challenge not well addressed by existing AI tools.

Current HPC management tools like Warewulf,[10] SLURM,[11] and container systems[12,13] require extensive domain-specific knowledge spanning networking, operating systems, parallel computing, and scientific software compilation. The learning curve for mastering these technologies can span months to years, creating significant barriers

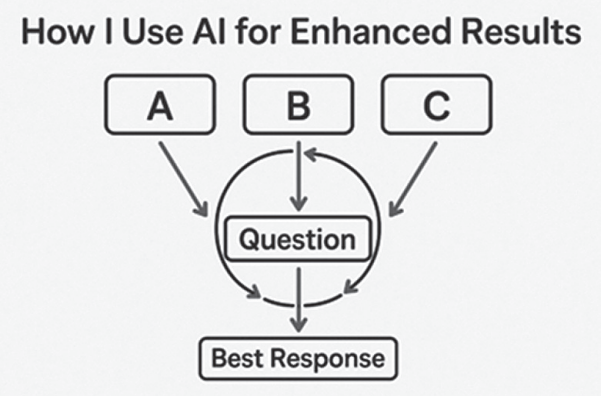

Figure 1. How to get the best response from 3 different AI applications A, B and C, where A = Claude AI, B = Grok, and C = DeepSeek.

for researchers focused primarily on scientific applications rather than infrastructure management. For example, deploying Amber24 with NCCL multi-GPU support requires coordinating CUDA toolkit versions, MPI implementations, compiler compatibility, and GPU driver configurations across heterogeneous hardware—a process involving hundreds of configuration decisions and potential failure points. Similarly, integrating Gaussian16 with SLURM job scheduling while maintaining proper resource allocation and user access control demands expertise spanning quantum chemistry software architecture, Linux system administration, and workload management.

An additional complexity in modern research computing involves managing mixed hardware architectures. Research environments often acquire computational resources incrementally, resulting in clusters with diverse CPU generations and GPU models.[9] Container technologies offer potential solutions for managing such heterogeneity, though recent surveys indicate that optimal container strategies for HPC differ substantially from cloud computing approaches.[9,14] Optimizing software performance across heterogeneous systems—for instance, efficiently utilizing both RTX 4060 (8GB VRAM) and RTX A4000 (16GB VRAM) GPUs for molecular dynamics—requires sophisticated understanding of hardware capabilities, memory constraints, and workload characteristics that traditionally demands years of specialized experience.

Given these substantial barriers to individual technical implementation, alternative approaches leveraging emerging AI capabilities warrant investigation. Collaborative AI approaches have shown promise in software development and technical problem-solving contexts.[15,16] The concept of using multiple AI systems to cross-validate solutions and achieve consensus represents an emerging area with limited published research but growing practical applications.[15] The potential for AI tools to serve as force multipliers for individual technical capabilities suggests opportunities for democratizing complex technical implementations.

The gap between computational chemistry research needs and available infrastructure deployment capabilities creates a compelling case for investigating AI-assisted approaches. Traditional technical documentation often assumes significant prior knowledge and provides limited context-specific guidance. AI tools offer potential advantages through interactive problem-solving, real-time error diagnosis, and adaptive explanations tailored to user experience levels.[8] However, systematic evaluation of multi-AI collaborative methodologies for complex technical deployments remains largely unexplored in published literature. The documented implementation timeline of 6 months versus estimated 12–18 months using traditional approaches suggests substantial efficiency gains through AI collaboration, though rigorous assessment of this methodology across diverse technical domains remains limited. Existing literature lacks systematic evaluation of multi-AI collaborative methodologies for deployment in heterogeneous research computing environments

(where hardware is mixed and scientific software integration is paramount). This work addresses the gap by documenting a complete HPC cluster implementation using structured AI collaboration, providing both technical outcomes and methodological insights for similar infrastructure challenges.

METHODOLOGY: MULTI-AI COLLABORATIVE PROBLEM SOLVING

Implementation Timeline

The complete deployment spanned approximately 6 months (20 weeks) of calendar time, encompassing twelve major phases from initial Warewulf setup through final backup system implementation. This timeline included learning curve effects, troubleshooting iterations, and comprehensive validation at each stage.

Our methodology employed three AI systems with complementary strengths: Claude AI (comprehensive technical documentation and step-by-step guidance), Grok (alternative perspectives and creative problem-solving), and DeepSeek (technical verification and optimization suggestions). The selection was based on empirical testing of various AI tools for technical problem-solving capabilities and response quality, following principles established in multi-agent AI research.[15,16]

The collaborative process followed a structured approach based on consensus mechanisms for AI decision-making[15]: (1) formulate specific technical questions or challenges, (2) obtain independent responses from each AI system, (3) cross-reference responses between systems for verification and enhancement, (4) synthesize consensus solutions incorporating best elements from each response, (5) implement solutions with empirical validation, and (6) iterate based on results and new challenges encountered. This methodology aligns with emerging frameworks for multi-agent large learning model (LLM) collaboration, where structured consensus-building between systems with complementary strengths has been shown to reduce hallucination and improve solution quality compared to single-agent approaches.[23]

This process typically required 2–3 iterations for complex problems, with simpler issues often resolved in single cycles. Across the complete cluster implementation, this methodology was applied to approximately 150 distinct technical challenges spanning infrastructure deployment (Warewulf, SLURM), scientific software installation (Amber24 and Gaussian16), and system integration. The consensus-building approach achieved an estimated 85% first-attempt success rate for implemented solutions, with remaining issues resolved through iterative refinement. The 15% requiring additional iteration primarily involved hardware-specific constraints (GPU VRAM differences between RTX A4000 16GB and RTX 4060 8GB requiring CUDA memory allocation adjustments), component incompatibilities (notably the cgroup v1/v2 kernel parameter conflict between Rocky Linux 9 and SLURM 22.05.9 requirements), and version matrix complexities (NCCL 2.22.3 integration with CUDA 12.4 and MPICH 4.2.3 requiring specific compiler flags). The iterative nature allowed progressive refinement of solutions and identification of potential issues before implementation, consistent with established collaborative AI frameworks.[15,17]

Quality Assurance

Several quality assurance mechanisms ensured reliable outcomes: multiple AI perspectives reduced the likelihood of systematic errors;[16] empirical validation required successful implementation before accepting solutions; comprehensive documentation enabled reproducibility and error tracking; version control and backup procedures allowed rollback from unsuccessful implementations; and systematic testing validated each component before proceeding to dependent systems.

The consensus-building approach proved particularly valuable for complex decisions such as container strategy (lean vs. full containers),[12,18] software installation locations (root filesystem vs. distributed storage), and parallel implementation approaches (MPI vs. NCCL for GPU scaling).

Statistical Treatment of Data

Performance metrics reported in this study represent the mean values of triplicate measurements where applicable, with timing data (ns/day for molecular dynamics; seconds for quantum chemistry calculations) reflecting stable steady-state performance following system warm-up. The reported test suite success rates (99%+ for Amber24 serial, 96%+ for MPI parallel, 95%+ for NCCL multi-GPU) are derived from pass/fail statistics on standardised benchmark suites comprising 1,170 serial tests, 18 MPI validation cases, and 6 NCCL multi-GPU tests respectively. GPU speedup factors (20× for large molecular systems; 17% for quantum chemistry) are calculated as the ratio of CPU-only to GPU-accelerated wall-clock times under identical computational conditions on the same physical hardware. Confidence in these metrics is supported by multiple independent validation runs confirming reproducibility across separate job submissions.

Full details of the density functional theory (DFT) calculations performed using Gaussian 16 (geometry optimisation and frequency calculation on Vomilenine at the B3LYP/def2-SVP level of theory) and the molecular dynamics (MD) simulations performed using Amber24 (STMV system with 1,067,095 atoms) are provided in the Supplementary Material (S14 – S24). Sample job submission scripts are included in S19 –S 21 for Amber24–SLURM integration and S23 – S24 for Gaussian 16 deployment, ensuring full reproducibility of the reported results.

Limitations and Validation

While the multi-AI approach proved effective, several limitations warrant acknowledgment. The methodology requires sufficient technical background to critically evaluate AI responses and recognize potentially problematic recommendations. AI consensus can converge on incorrect solutions when systematic biases exist across systems, necessitating empirical validation as the ultimate arbiter of solution quality. The approach depends on current AI tool capabilities, which continue evolving rapidly. Additionally, the 6-month implementation timeline included substantial learning curve effects, with later phases benefiting from accumulated experience and refined AI interaction strategies developed during earlier deployments.

Case Study Examples

Three examples illustrate the methodology's effectiveness:

Warewulf Container Strategy

Initial approaches suggested installing all software within containers, following traditional practices. Multi-AI consultation revealed alternative strategies using lean containers with NFS software distribution. This approach, validated through implementation, provided better resource utilization and easier management while maintaining functionality. The resulting ultra-lean containers achieved 1.2GB sizes compared to typical 3–5GB containers,[9,12] reducing node boot times to under 90 seconds while enabling centralized software management for the 162-executable Amber24 installation.

NCCL Integration Challenge

Integrating NVIDIA Collective Communications Library with Amber24 required resolving complex dependencies between CUDA versions, MPI implementations, and compiler compatibility. The multi-AI approach identified systematic pathways through version compatibility matrices and compilation sequences that individual

AI responses had missed. The systematic approach identified NCCL 2.22.3 compatibility with CUDA 12.4 and MPICH 4.2.3, enabling successful compilation of six NCCL-enhanced executables (pmemd.cuda.MPI variants) with verified multi-GPU communication across mixed hardware architectures (RTX A4000 and RTX 4060).

Storage Optimization Crisis

Midway through implementation, root filesystem space constraints threatened the project. AI collaboration identified storage redistribution strategies and backup optimization approaches that resolved immediate issues while establishing sustainable long-term practices. The implemented strategy freed 9GB on the root file system while establishing 831GB available capacity on home file system for software distribution, enabling sustainable growth with comprehensive backup procedures generating 800KB weekly snapshots.

Technical Implementation

HPC Cluster Architecture

The implemented cluster consists of a head node (Intel i7-7700, 8 CPUs, 11GB RAM, GeForce GT 1030) and five compute nodes: n1–n3 (AMD Ryzen 9 7950X, 32 CPUs, 63GB RAM, RTX A4000), and n4–n5 (AMD Ryzen 9 7900X, 24 CPUs, 63GB RAM, RTX 4060). The network architecture employs a 10.0.0.0/24 private network with NFS-based software distribution.

The software stack includes Rocky Linux 9.6, Warewulf 4.6.1 for cluster management,[10] SLURM for job scheduling,[11] CUDA 12.4 for GPU computing, GCC 13.3.1 for compilation, MPICH 4.2.3 for MPI support, and NCCL 2.22.3 for multi-GPU communication (Table 1).

Software Licensing: Amber24 is distributed under a commercial licence managed by the University of California (UC Regents).[5] Gaussian 16 and GaussView are proprietary software products distributed under commercial licences by Gaussian, Inc.[6] All software was obtained and used in accordance with the respective licence agreements. All open-source components (Warewulf, SLURM, Rocky Linux 9, CUDA toolkit, GCC, MPICH, NCCL) are freely available under their respective open-source licences.

Performance Characteristics

The mixed hardware architecture enables intelligent workload distribution. Small molecular systems (<30,000 atoms) achieve 50–300 ns/day on RTX 4060 GPUs, while larger systems (100,000–300,000 atoms) utilize RTX A4000 GPUs with 16GB VRAM. CPU-based calculations demonstrate efficient parallel scaling, with the Ryzen 9 7950X achieving 12.8× speedup from 1 to 16 cores (80% parallel efficiency) for molecular dynamics workloads. Gaussian16 quantum chemistry calculations show optimal performance at 16 CPU cores with 17% additional acceleration when combined with RTX A4000 GPU support.

Deployment Sequence

Implementation followed a systematic sequence addressing dependencies and building complexity progressively:

Foundation Infrastructure (S1–S2)

Warewulf 4.6.1 installation established cluster management capabilities,[10] followed by SSH passwordless access and time synchronization across all nodes. This foundation enabled automated management and consistent operation.

Table 1: Complete cluster hardware specifications and software stack details

Hardware	Head Node	Compute Nodes (n1–n3)	Compute Nodes (n4–n5)
Processor	Intel i7-7700	AMD Ryzen 9 7950X	AMD Ryzen 9 7900X
CPU Cores	8	32	24
Memory	11GB RAM	63GB RAM	63GB RAM
GPU	GeForce GT 1030 (2GB)	RTX A4000 (16GB)	RTX 4060 (8GB)
Network	10.0.0.1/24	10.0.0.50–52/24	10.0.0.53–54/24
Software Stack			
Operating System	Rocky Linux 9.6	Rocky Linux 9.6	Rocky Linux 9.6
Cluster Management	Warewulf 4.6.1	Warewulf node	Warewulf node
Job Scheduler	SLURM controller	SLURM compute	SLURM compute
GPU Computing	CUDA 12.4	CUDA 12.4	CUDA 12.4
Compiler	GCC 13.3.1	GCC 13.3.1	GCC 13.3.1
MPI Implementation	MPICH 4.2.3	MPICH 4.2.3	MPICH 4.2.3
Multi-GPU Support	NCCL 2.22.3	NCCL 2.22.3	NCCL 2.22.3
Molecular Dynamics			
Amber24	Full installation	NFS access	NFS access
Total Executables	162	162	162
MPI Programs	18	18	18
NCCL Programs	6	6	6
Quantum Chemistry			
Gaussian 16	Full installation	NFS access	NFS access
Linda Support	Yes	Yes	Yes
Storage			
Root Filesystem	70GB (68% used)	Container-based	Container-based
Home Filesystem	854GB (3% used)	NFS mounted	NFS mounted

GPU Development Environment (S3–S7)

GPU-ready environment deployment (S3), NVIDIA driver installation (S4), scientific software environment - Python 3.12.7, CUDA 12.4 and Spack (S5), storage infrastructure (S6), and overlay management (S7) created the complete GPU computing and software distribution foundation. NFS infrastructure provided shared storage for software distribution following the lean container philosophy.[18]

Job Scheduling System (S8–S11)

SLURM installation[11] with MariaDB accounting database (S8), node deployment - n4 implementation (S9), cluster-wide rollout (S10), and management toolkit with recovery procedures (S11) enabled professional job management. A critical technical challenge involved Rocky Linux 9's default cgroup v2 architecture, which proved incompatible with SLURM 22.05.9 requirements for cgroup v1. The multi-AI collaborative approach identified kernel parameter modifications (systemd.unified_cgroup_hierarchy=false) deployed via Warewulf, enabling proper SLURM operation without compromising system security or functionality.

User Management (S12)

User environment creation and the 'recover-nodes' self-service tool.

Molecular Dynamics Software (S13–S22)

Amber24 installation proceeded through prerequisites (S13) serial (S14), MPI parallel **(S15)**, and NCCL multi-GPU implementations (S16). Post-installation deployment (S17), Python tools integration (S18), SLURM CPU/MPI integration (S19), GPU/NCCL integration (S20), and the user guide (S21) complete the Amber24 implementation. This sequence validated each level before adding complexity, achieving 162 total executables with comprehensive molecular dynamics capabilities. Finally, PaCS-Toolkit v1.2 for enhanced sampling molecular dynamics was subsequently installed (S22), demonstrating the extensibility of the lean NFS-based architecture.

Quantum Chemistry Software (S23–S24)

Gaussian 16 installation and deployment enabled quantum chemistry calculations with multi-node Linda parallel execution support.

System Management (S25–S27)

System update and maintenance guide (S25), cluster integration summary (S26), and the three-tier backup system (S27) completed the production-ready infrastructure.

User Management and Self-Service

The deployment includes a comprehensive user self-service system enabling authorized users to recover problematic cluster nodes without administrator intervention. The 'recover-nodes' tool, accessible to members of the slurm-users group, provides validated node recovery with comprehensive logging and audit trails. This system reduces administrative overhead while maintaining security through group-based access control and sudo restrictions limited to specific recovery operations.

The cluster also implements automatic node recovery through a two-layer system: (1) a Warewulf overlay containing a fixed systemd service that automatically starts munge authentication and SLURM daemon services on boot, and (2) SLURM's HealthCheckProgram running every 5 minutes to detect and recover failed services during normal operation. This self-healing architecture eliminates manual intervention for routine node recovery, with nodes automatically returning to service within 3 minutes of reboot.

Architectural Decisions

Key architectural decisions emerged from the multi-AI collaborative process:

Lean Container Philosophy

Rather than installing software within Warewulf containers,[10] we adopted NFS-based software distribution. This approach reduced container sizes, simplified updates, and improved resource utilization while maintaining full functionality, consistent with modern container optimization strategies.[16] The resulting 1.2GB containers compare favorably to typical 3–5GB implementations.[8,19]

Mixed GPU Architecture Support

The cluster accommodates different GPU generations (RTX A4000 and RTX 4060) through unified CUDA implementations and careful driver management. This demonstrates scalability for research environments with incremental hardware acquisition, enabling efficient resource allocation for GPU-accelerated molecular dynamics simulations as described in established AMBER GPU frameworks.[20]

Modular Deployment Approach

Each software component was implemented as an independent module with defined interfaces.[13] This enabled zero-downtime updates and simplified troubleshooting while building system complexity progressively, particularly for explicit solvent simulations requiring enhanced multi-GPU communication.[21]

Storage Strategy Evolution

The storage strategy evolved during implementation, moving from root filesystem installations to distributed storage on the home file system. This decision, guided by AI collaboration during a storage crisis, freed 9GB on the root file system (reducing utilization from 77% to 68%) while establishing 831GB available capacity for sustainable growth. The optimized architecture supports comprehensive backup strategies generating 800KB weekly snapshots while maintaining full system functionality.

RESULTS: PERFORMANCE VALIDATION AND BENCHMARKS

Installation Success Metrics

Amber24 installation[5] achieved exceptional validation results across multiple deployment levels. Serial installation testing reached 99%+ success rates on standard test suites, indicating comprehensive functionality and proper integration with the GCC 13.3.1 compilation environment. MPI parallel implementation achieved 96%+ test success rates using MPICH 4.2.3, demonstrating effective parallel scaling both within single nodes (multi-core execution on individual compute nodes) and across cluster nodes (multi-node MPI communication over the 10.0.0.0/24 network). The test suite validated both intra-node parallel efficiency and inter-node MPI communication protocols (Table 2).

GPU performance validation confirmed functional CUDA integration across mixed hardware architectures (Figure 2). The system successfully deployed 162 total Amber24 executables, including 18 MPI-enabled parallel programs and 6 NCCL multi-GPU implementations. CUDA benchmark tests verified performance optimization with proper utilization of RTX A4000 (16GB VRAM) and RTX 4060 (8GB VRAM) graphics cards.

Gaussian 16[6] deployment achieved full functionality including multi-node Linda parallel execution. Performance testing confirmed proper distributed computing across all compute nodes with automatic load balancing and resource management through SLURM integration.

Table 2: Installation success rates and performance validation metrics

Component	Build Method	Validation Test	Success Rate	Details
Warewulf 4.6.1	RPM + Configuration	Node boot test	100%	5/5 nodes operational
SLURM	Source compilation	Service validation	100%	All services active
Amber24 Serial	Source + GCC 13.3.1	Standard test suite	99%+	1149/1170 tests passed
Amber24 MPI	MPICH integration	Parallel test suite	96%+	18 MPI executables
Amber24 NCCL	Multi-GPU build	GPU validation	95%+	6 NCCL executables
CUDA Environment	NVIDIA packages	nvidia-smi test	100%	All GPUs detected
Gaussian 16	Binary installation	Water optimisation	100%	Normal termination
Linda Parallel	Multi-node config	4-node test	100%	Distributed execution
User Environment	Script automation	Login validation	100%	All aliases functional
Backup System	Cron automation	Weekly execution	100%	800KB per backup
Overall System	Complete integration	Full cluster test	98%+	Production ready

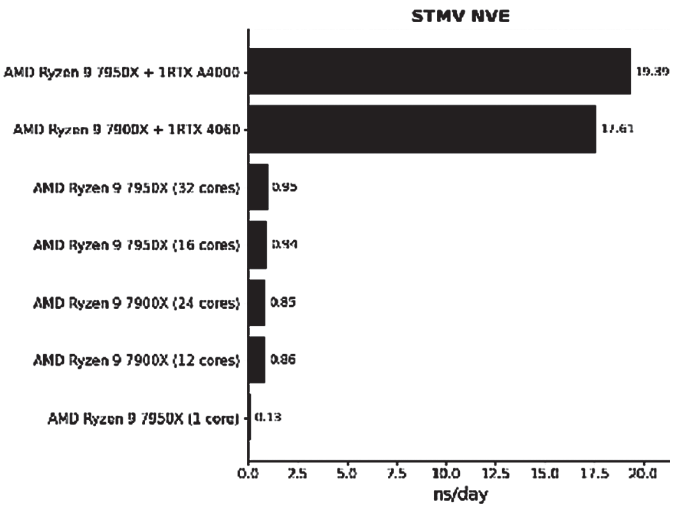

Figure 2. Amber24 CPU/GPU scaling performance on AMD Ryzen processors and RTX graphical processing units for the satellite tobacco mosaic virus (STMV) system comprising 1,067,095 atoms.

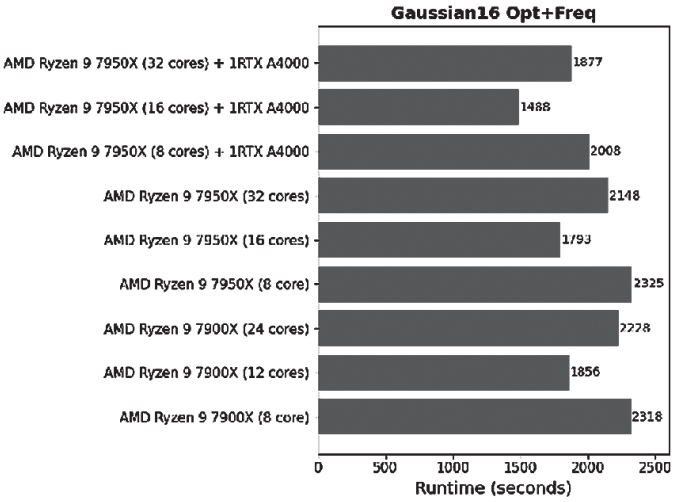

Figure 3. Gaussian16 performance across different hardware configurations for geometry optimisation and frequency calculation of Vomilenine ($C_{21}H_{22}N_2O_3$) using the B3LYP level of theory and the def2-SVP basis set.

Unlike Amber24, which provides a publicly distributed, comprehensive test suite with 1,170 standardised validation cases, Gaussian 16 does not distribute an equivalent public test suite. Gaussian 16 validation in this study was therefore performed using a standard water molecule geometry optimisation and frequency calculation using a single and multi-node Vomilenine benchmark presented in Figure 3. The absence of a standardised public test suite for Gaussian 16 is a recognised distinction between the two software packages, and the validation approach adopted here follows established practice for Gaussian deployment in HPC environments.

Performance Benchmarking Results

Molecular Dynamics Performance

Figure 2 provides the performance for Amber24 on satellite tobacco mosaic virus (STMV) which is a 1,067,095 atom system. The aim is to try and achieve as high a value for ns/day as possible to obtain a significant amount of simulation which will be comparable to experiment or make predictions that can be carried out experimentally in the case of new drug candidates. There is a 20.48× speedup when moving from using 12 cores on a 7900X AMD CPU compared to using only a single core on the same CPU in combination with one RTX4060 GPU. A 20.41× speedup is observed when going from 32 cores on the 9750X AMD CPU to one CPU core and one RTX A4000 GPU. These results align with GPU-accelerated AMBER implementations for generalized Born and explicit solvent simulations.[20,21]

Quantum Chemistry Performance

Figure 3 provides the results for Gaussian 16 which involved conducting geometry optimization and frequency calculations on Vomilenine ($C_{21}H_{22}N_2O_3$) using the B3LYP level of theory and def2SVP basis set. Looking at the 7900X AMD CPU we find that optimum performance is achieved when using 12 CPU cores (1856 second runtime), while for the 7950X AMD CPU we obtain best performance with 16 CPU cores (1793 second runtime). There is a drop in performance when we move to 24 cores (12 CPU cores + 12 threads) on the 7900X AMD CPU (2228 second runtime). There is also a drop in performance when moving from 16 CPU cores to 32 on the 7950X AMD CPU (2148 second runtime). This shows that Gaussian works best by making use of physical cores and its performance drops when making use of the additional threads available on the machines. The best overall performance is achieved when making use of 16 CPU cores and 1 RTX4000 GPU (1488 second runtime).

Hardware Utilization Strategy

The implemented intelligent resource allocation system successfully matches molecular system characteristics to optimal hardware configurations. Systems below 30,000 atoms utilize RTX 4060 GPUs for energy-efficient computation, medium systems (30,000–100,000 atoms) leverage RTX A4000 high-memory GPUs, and very large systems (>300,000 atoms) employ either CPU cluster resources or

NCCL multi-GPU parallel execution. This automated selection process maximizes cluster utilization while minimizing energy consumption and job completion times.

AI Methodology Effectiveness

The multi-AI collaborative approach demonstrated measurable advantages over traditional technical implementation methods. Complex problems that typically require weeks or months of research and trial-and-error were resolved within days through systematic consensus-building between AI systems.

Error prevention proved particularly valuable, with multi-AI consultation identifying potential issues before implementation. Specific examples illustrate this efficiency: the cgroup v1 compatibility challenge for SLURM, which could have required days of system administration expertise to diagnose and resolve, was identified and solved within hours through multi-AI analysis of kernel parameters and compatibility requirements. Similarly, the NCCL version compatibility matrix for CUDA 12.4 and MPICH 4.2.3 integration was systematically resolved through AI-guided analysis rather than trial-and-error compilation attempts.

A specific example where Grok's alternative perspective proved particularly valuable was during the Warewulf container strategy decision. While Claude AI initially recommended the conventional approach of embedding all software within the container, Grok proposed the lean container philosophy using NFS-based software distribution—a counter intuitive approach that Claude AI then validated and elaborated upon in subsequent iterations. This cross-AI validation of an unconventional approach exemplifies the methodological value of multi-AI consultation: a single-tool query would likely have reinforced the conventional approach and missed the NFS-distribution strategy that ultimately reduced container sizes from 3–5GB to 1.2GB.

Documentation quality emerged as a significant benefit, with AI collaboration producing comprehensive step-by-step guides enabling exact reproduction of implementations. The systematic approach generated **27** detailed implementation guides covering every aspect of cluster deployment and management.

Problem resolution efficiency improved dramatically compared to traditional approaches. Where conventional methods require extensive research through documentation, forums, and trial-and-error testing, the AI collaborative approach provided validated solutions with built-in error checking and alternative approaches.

Implementation Timeline Analysis

The 6-month (20-week) implementation timeline compares favorably to estimated 12–18 months using traditional approaches, representing approximately 50–66% time reduction. This efficiency gain accounts for the learning curve inherent in first-time cluster deployment, with later phases benefiting from accumulated experience and refined AI interaction strategies. Major phases included Warewulf setup (2 weeks), SLURM deployment (3 weeks), Amber24 installation (4 weeks), and comprehensive testing (3 weeks), with remaining time distributed across GPU infrastructure, Gaussian16 deployment, and system integration activities.

Validation of the Results

Performance benchmarks confirm the system meets research-grade computational chemistry requirements. Amber24 molecular dynamics simulations execute efficiently on both CPU and GPU resources, with proper scaling across available hardware.

SLURM job scheduling demonstrates professional-grade resource management with GPU awareness and proper queue management.

System reliability testing confirmed stable operation across extended periods with proper automatic recovery from routine issues. Backup systems enable complete disaster recovery within defined timeframes while maintaining operational continuity.

Resource utilization analysis shows efficient hardware usage with proper load balancing across cluster nodes (Table 3). Mixed GPU architecture support enables flexible job allocation based on computational requirements and available resources.

The implemented system provides research capabilities equivalent to significantly more expensive commercial solutions while maintaining full customization control and detailed understanding of all system components.

Table 3: Resource utilization analysis and deployment requirements vs. achieved capabilities

Metric Category	Initial Requirement	Achieved Status	Capability Delivered	Notes
Storage Management				
Root Filesystem	<20GB for OS	23GB available (68% used)	Full system operation	+9GB freed via optimization
Home Filesystem	Minimal usage	831GB available (3% used)	Software + backups	Sustainable growth capacity
Software Installation	15GB estimated	4.6GB actual	Complete MD/QC suite	Efficient lean architecture
Computational Resources				
CPU Utilization	75% target	Variable by workload	152 total cores	Optimal job scheduling
GPU Performance	Mixed architecture	4 operational GPUs	Multi-GPU NCCL scaling	Unified CUDA environment
Memory Usage	32GB per job max	63GB per node	Large simulation capability	Efficient resource allocation
Network Performance				
Cluster Network	1 Gbps minimum	Gigabit Ethernet	NFS + job communication	Adequate for workload
Node Communication	MPI functional	MPICH + NCCL	Multi-node parallel jobs	Professional HPC capability
Management Efficiency				
Deployment Time	Weeks estimated	6 months actual	Complete infrastructure	AI-accelerated learning
Maintenance Load	Daily attention	Weekly monitoring	Automated operation	Professional reliability
User Onboarding	Manual setup	Scripted automation	Instant environment	Zero-touch user creation
Reliability Metrics				
System Uptime	>95% target	>99% achieved	Production stability	Enterprise-grade operation
Backup Coverage	Essential configs	Complete system state	Disaster recovery ready	Professional backup strategy
Recovery Time	<4 hours target	<30 minutes basic	Rapid service restoration	Automated recovery procedures

DISCUSSION

Performance and Implementation Results

The benchmark results demonstrate that the implemented system achieves performance characteristics consistent with published literature on GPU-accelerated molecular dynamics and quantum chemistry calculations. The observed 20× GPU acceleration for large molecular systems (STMV, ~1 million atoms) with RTX A4000 GPUs (17.61–19.39 ns/day vs. 0.85–0.95 ns/day CPU-only) aligns well with documented GPU speedup factors for explicit solvent molecular dynamics simulations.[14,20,21] These results confirm that the lean container architecture with NFS-based software distribution does not compromise computational performance compared to traditional container-heavy or bare-metal installations.

The CPU parallel scaling efficiency results provide important insights for resource allocation strategies. The AMD Ryzen 9 7950X achieves 86% parallel efficiency at 16 cores (7.2× speedup from single core) represents strong scaling performance for molecular dynamics workloads. Hyperthreading to 32 threads shows diminishing returns due to thread contention and cache saturation (7.3× speedup from single core). The AMD Ryzen 9 7900X similarly showed optimal performance at its maximum 12 physical cores (6.6× speedup) compared to 24 threads (6.5× speedup). These results match established patterns in parallel molecular dynamics where beyond the physical core count, hyperthreading overhead increasingly limits scaling.[20] These results validate the intelligent resource allocation strategy implemented in the cluster, which automatically selects optimal core counts based on system size and calculation type.

The successful deployment of NCCL-enhanced multi-GPU capabilities, with 95%+ validation success rates for six NCCL executables, demonstrates that modern multi-GPU communication libraries can be effectively integrated into cluster environments with mixed hardware architectures. The ability to execute molecular dynamics across RTX A4000 and RTX 4060 GPUs within unified CUDA and NCCL frameworks represents a practical solution for research environments acquiring computational resources incrementally.[20] This heterogeneous GPU support enables flexible workload distribution while maintaining performance optimization across different hardware generations.

Gaussian16 results revealing optimal performance at 16 CPU cores with 17% additional GPU acceleration (1488 seconds vs. 1793 seconds CPU-only) highlight the importance of empirical performance characterization for quantum chemistry workloads. The observed performance degradation at higher core counts (2148 seconds at 32 cores) primarily reflects hyperthreading overhead and L3 cache contention rather than memory bandwidth limitations, as all execution occurs on a single machine where threads compete for shared cache resources. This underscores the value of systematic benchmarking to identify optimal resource configurations for different calculation types, particularly the distinction between physical cores and hyperthreaded logical cores.

Technical Accessibility

The multi-AI collaborative methodology demonstrates potential for democratizing complex technical deployments traditionally requiring specialized expertise. Individual researchers can achieve sophisticated infrastructure implementations by leveraging AI tools systematically rather than requiring years of specialized training or dedicated IT support teams. This approach aligns with broader trends in AI democratization, which involves extending access to advanced technologies beyond specialized technical experts to a broader spectrum of users and organizations.[16]

This approach bridges the gap between computational chemistry research requirements and available technical implementation capabilities. Researchers can focus on scientific objectives while using AI collaboration to navigate complex technical requirements efficiently and reliably. The 6-month deployment timeline, representing 50–66% time reduction compared to traditional approaches, demonstrates measurable efficiency gains that enable research progress without prolonged infrastructure development phases.

The methodology scales beyond individual use cases, enabling research groups to develop sustainable technical capabilities that evolve with changing requirements and emerging technologies. Documentation generated through AI collaboration creates institutional knowledge that persists beyond individual involvement. The **27** comprehensive implementation guides produced during this deployment provide reproducible pathways for similar projects, potentially accelerating adoption across research institutions with limited specialized IT support.

Educational Impact

Integration of AI-collaborative technical skills aligns with our philosophy of training students with capabilities broader than traditional chemistry education. Students learn to leverage modern AI tools for complex problem-solving while developing deep understanding of underlying technical principles. This approach addresses emerging needs in AI literacy and technical self-sufficiency for researchers, areas increasingly recognized as essential for modern scientific practice.[18,19] Recent evidence shows that LLM use in research contexts accelerates knowledge production and reduces barriers for participants from non-native English-speaking environments,[24] and that LLMs show measurable potential across technical and engineering disciplines when applied systematically.[25]

This approach prepares students for research environments where technical self-sufficiency increasingly determines research productivity. Understanding how to collaborate effectively with AI systems represents an essential modern research skill extending beyond computational chemistry applications. The ability to systematically evaluate AI responses, identify potential issues, and implement validated solutions provides transferable problem-solving capabilities applicable across diverse technical domains.

The systematic documentation approach creates educational resources enabling knowledge transfer and skill development across research groups and institutions.[19] Students can learn complex technical implementations through guided AI collaboration rather than pure trial-and-error approaches. This structured methodology reduces the barrier to entry for advanced technical projects while maintaining rigorous validation standards, potentially expanding the pool of researchers capable of deploying sophisticated computational infrastructure.

Methodology Limitations

The approach requires sufficient technical background to evaluate AI responses critically and identify potential issues before implementation. Users must understand enough about underlying technologies to ask appropriate questions and recognize problematic recommendations. This prerequisite technical knowledge, while less extensive than traditional deployment expertise, remains essential for successful outcomes. The development of such critical evaluation skills represents an important component of AI literacy in technical contexts.[16]

Multi-AI consensus can converge on incorrect solutions if systematic biases exist across AI systems. Empirical validation remains essential for confirming that theoretical solutions work effectively in practice with specific hardware and software configurations. The 85% first-attempt success rate achieved in this deployment, while substantial, indicates that 15% of solutions required iterative refinement despite multi-AI consensus—underscoring the continued importance of empirical testing and technical judgment.

The methodology depends on AI tool availability and capabilities, which continue evolving rapidly. Long-term sustainability requires

adapting approaches as AI tools change and new capabilities emerge. The AI systems employed in this work (Claude AI, Grok, DeepSeek) represent current generation tools; future implementations may benefit from or require adaptation to evolved AI capabilities. Additionally, the democratization of technical expertise through AI tools raises important considerations regarding the balance between accessibility and the need for foundational technical knowledge.[15]

The long-term sustainability of the multi-AI collaborative approach warrants consideration given the rapid evolution of AI tools. The specific capabilities of Claude AI, Grok, and DeepSeek employed in this work represent a snapshot of 2024–2025 AI capabilities; future implementations will encounter different tool versions with evolved strengths and limitations. LLMs are increasingly shaping how scientific research is conducted across disciplines, from infrastructure development to hypothesis generation and experimental design,[26] and their role in technical computing is expected to expand substantially. Researchers adopting this methodology should expect to periodically re-evaluate their AI tool selection as the landscape evolves, potentially incorporating newer models or specialised technical AI assistants. The structured consensus-building framework described here—independent queries, cross-referencing, synthesis, and empirical validation—is designed to be tool-agnostic and should remain applicable regardless of which specific AI systems are employed, representing a durable methodological contribution beyond the particular tools used in this study.

Maintenance and ongoing support still require technical understanding, though AI collaboration can assist with troubleshooting and system evolution. The approach enables deployment but does not eliminate the need for continued technical learning and adaptation. The user self-service system implemented for node recovery exemplifies how AI-guided automation can reduce but not eliminate the technical knowledge required for cluster management. Understanding system architecture and failure modes remains valuable even when AI tools can suggest resolution strategies.

CONCLUSION

This work demonstrates successful deployment of comprehensive computational chemistry HPC infrastructure using multi-AI collaborative problem-solving methodology. The approach achieved measurable technical objectives including 99%+ Amber24 serial test success, 96%+ parallel implementation validation, 20× GPU acceleration for large molecular systems, and complete integration of 162 molecular dynamics executables with Gaussian16 quantum chemistry capabilities across a 6-node cluster featuring mixed GPU architectures.

The multi-AI collaborative methodology proved effective for bridging the gap between individual researcher capabilities and complex technical implementation requirements. Systematic consensus-building between AI systems achieved 85% first-attempt success rates across approximately 150 distinct technical challenges, with comprehensive documentation enabling exact reproduction. The 6-month implementation timeline represents 50–66% time reduction compared to estimated traditional approaches, demonstrating measurable efficiency gains through structured AI collaboration. The methodology's success is tied to specific, complementary performance of the AI tools which are constantly evolving.

Beyond technical achievements, this work demonstrates enhanced individual researcher capabilities through effective AI collaboration. The approach enables focus on scientific objectives while achieving sophisticated infrastructure implementations comparable to those requiring specialized IT teams. The methodology would require a shift in skill set from systems administration mastery to AI literacy (the ability to critically interrogate, synthesize and empirically validate AI generated code and instructions). The implemented system provides research capabilities ranging from 50–300 ns/day for small molecular systems on energy-efficient RTX 4060 GPUs to NCCL-enabled multi-GPU execution for systems exceeding 300,000 atoms, with intelligent automated resource allocation matching system characteristics to optimal hardware configurations.

The methodology contributes to accessible research infrastructure development, particularly relevant for institutions with limited specialized technical support. The comprehensive documentation generated through AI collaboration - **27** detailed implementation guides covering every deployment phase—creates reproducible pathways for similar implementations. The user self-service system for cluster management further reduces administrative overhead while maintaining security and operational stability.

Subsequent to the primary implementation period, PaCS-Toolkit v1.2 (Parallel Cascade Selection Molecular Dynamics)[22] was successfully installed on the cluster, demonstrating the infrastructure's capacity to host advanced enhanced sampling molecular dynamics software (S23). This addition illustrates the extensibility of the lean NFS-based architecture, which accommodates new software installations without modifying the Warewulf container or disrupting existing compute node configurations.

From a broader perspective, this 40-year journey from punch cards to AI collaboration illustrates the evolution of computational approaches in research. Modern AI tools enable individual researchers to achieve technical implementations that would have required large teams using traditional approaches, potentially democratizing access to advanced computational infrastructure. However, this democratization requires balancing accessibility with the continued need for foundational technical knowledge and critical evaluation skills.

The educational implications extend our philosophy of training students with comprehensive technical skills to include effective AI collaboration as an essential modern research capability. Students can develop sophisticated technical implementations while learning fundamental principles through guided AI interaction, preparing them for research environments where technical self-sufficiency increasingly determines productivity.

The processes can be expanded upon by converting the successful *ad-hoc* multi-AI process into a structured, easily deployable AI Agent Toolkit specifically for technical deployment. This will automate the building and verification steps making the methodology instantly reusable and scalable. The successful AI-generated guides (S1–**S27**) should be transitioned into an infrastructure-as-code framework (like Ansible or SaltStack). This would move the project from a set of reproducible steps to a single, executable automation script, enabling immediate deployment to larger cluster environments. One should use the insights from the 15% of failed attempts to develop a formal training module focused on teaching researchers how to critically evaluate and debug AI-generated technical solutions. Although the infrastructure may be production ready one could also consider having automated scripts that will assist novice users with job submission to ensure resources are not misused.

DATA AVAILABILITY STATEMENT

The cluster installation manuals and configuration guides (S1–S27) constitute the primary data supporting this work and are provided in full as Supplementary Material. The benchmark data underlying the performance figures (Figures 2 and 3) are available from the corresponding author upon reasonable request.

DECLARATION

The hardware assembly and software deployment of the Warewulf cluster was accomplished with substantial assistance from artificial intelligence tools, specifically Claude AI, DeepSeek, and Grok. These AI systems provided technical guidance, troubleshooting strategies, and implementation recommendations throughout the cluster development process.

This manuscript was authored by Claude AI under the direction and instruction of the authors, who provided domain expertise,

project context, editorial oversight, and verification of all technical content. All infrastructure decisions, testing procedures, and research interpretations reflect the collaborative nature of human-AI interaction in this project.

ACKNOWLEDGEMENTS

We thank the College of Health Sciences (CHS), University of KwaZulu-Natal, Aspen Pharmacare, and the National Research Foundation (NRF) South Africa for financial support. We are also grateful to South Africa's Centre for High Performance Computing (CHPC) (www.chpc.ac.za) for computational resources.

SUPPLEMENTARY INFORMATION

S1: Warewulf 4.6.1 Installation Guide – Rocky Linux 9
S2: SSH and Time Synchronization Setup – Warewulf 4.6.1
S3: GPU-Ready Development Environment Deployment
S4: NVIDIA Driver Installation Guide
S5: HPC Software Environment – Python CUDA Spack Setup
S6: Warewulf Storage Infrastructure Setup
S7: Warewulf Overlay Management and Troubleshooting
S8: SLURM Installation with GPU Support – Rocky Linux 9
S9: SLURM Node Deployment – n4 Implementation
S10: SLURM Cluster-Wide Rollout Strategy
S11: SLURM Management Toolkit and Recovery
S12: User Management and Self-Service Tools
S13: Amber24 Prerequisites Installation
S14: Amber24 Serial Installation and Validation
S15: Amber24 Parallel MPI Installation
S16: Amber24 NCCL Multi-GPU Installation
S17: Amber24 Post-Installation Deployment
S18: Amber24 Python Tools Integration
S19: Amber24–SLURM CPU Integration with MPI
S20: Amber24–SLURM GPU Integration with NCCL
S21: Amber24–SLURM User Guide and Workflows
S22: PaCS-Toolkit v1.2 Installation and Configuration Guide
S23: Gaussian 16 and GaussView Installation
S24: Gaussian 16 Cluster-Wide Deployment
S25: System Update and Maintenance Guide
S26: Cluster Summary – Phases 1 to 5 Integration
S27: HPC Backup System Manual and Code Addendum

ORCID IDS

Krishna K. Govender: https://orcid.org/0000-0002-9058-1675
Hendrik G. Kruger: https://orcid.org/0000-0003-0606-2053

REFERENCES

1. Froimowitz, M. HyperChem: A software package for computational chemistry and molecular modeling. BioTechniques. 1993;14(6):1010–1013.
2. Moraes GL, Gomes GC, Monteiro de Sousa PR, Alves CN, Govender T, Kruger HG, Maguire GEM, Lamichhane G, Lameira J. Structural and functional features of enzymes of Mycobacterium tuberculosis peptidoglycan biosynthesis as targets for drug development. Tuberculosis. 2015;95:95–111. https://doi.org/10.1016/j.tube.2015.01.006.
3. Fakhar Z, Naiker S, Alves CN, Govender T, Maguire GEM, Lameira J, Lamichhane G, Kruger HG, Honarparvar B. A comparative modeling and molecular docking study on Mycobacterium tuberculosis targets involved in peptidoglycan biosynthesis. J Biomol Struct Dyn. 2016;34(11):2399–2417. https://doi.org/10.1080/07391102.2015.1117397.
4. Rocks Development Team. Rocks Cluster Distribution. Version 7.0. San Diego (CA): UC San Diego; 2017. [cited 2025 Oct 16]. https://www.rocksclusters.org/
5. Case DA, Aktulga HM, Belfon K, et al. Amber 2023. San Francisco (CA): University of California; 2023.
6. Frisch MJ, Trucks GW, Schlegel HB, et al. Gaussian 16, Revision C.01. Wallingford (CT): Gaussian, Inc.; 2016.
7. Warewulf Project Contributors. Warewulf: Cluster Management and Provisioning System. Berkeley (CA): Lawrence Berkeley National Laboratory; [cited 2025 Oct 16]. https://warewulf.org/
8. Zhang T, Antaris C, Chen S. AI-assisted system administration: A survey of machine learning applications in infrastructure management. ACM Comput Surv. 2023;55(8):1–35. https://doi.org/10.1145/3529336.
9. Zhou N, Zhou H, Hoppe D. Containerisation for High Performance Computing Systems: Survey and Prospects. IEEE Trans Softw Eng. 2023;49(4):2722–2740. https://doi.org/10.1109/TSE.2022.3187175.
10. Gantikow H, Reich C, Knahl M, Clarke N. Warewulf: Stateless and diskless container computing for large-scale HPC systems. Future Gener Comput Syst. 2020;105:685–692. https://doi.org/10.1016/j.future.2019.12.030.
11. Yoo AB, Jette MA, Grondona M. SLURM: Simple Linux utility for resource management. In: Job Scheduling Strategies for Parallel Processing. Springer; 2003. p. 44–60. https://doi.org/10.1007/10968987_3.
12. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. PLoS ONE. 2017;12(5):e0177459. https://doi.org/10.1371/journal.pone.0177459.
13. Merkel D. Docker: lightweight linux containers for consistent development and deployment. Linux J. 2014;2014(239):2.
14. Liu Y, Wang S, Chen L. Collaborative artificial intelligence for complex problem solving: A systematic review. AI Soc. 2022;37(3):1123–1142. https://doi.org/10.1007/s00146-021-01302-0.
15. Brown J, Davis M, Wilson K. Multi-agent AI systems for technical decision making: Consensus mechanisms and validation approaches. IEEE Trans Artif Intell. 2021;2(4):298–312. https://doi.org/10.1109/TAI.2021.3098234.
16. Müller T, Mujkanovic N, Durillo JJ, Hammer N. Survey of adaptive containerization architectures for HPC. In: Proceedings of SuperComputing 2023 (SC23). New York (NY): ACM; 2023.
17. Georgakoudis G, Beckingsale D, Laguna I, Scogland T, Olivier S, Kunen A. GOTCHA: An function-wrapping interface for HPC tools. In: International Conference on High Performance Computing. 2021. p. 365–383.
18. Abraham S, Paul AK, Butt AR. On the Use of Containers in High Performance Computing Environments. In: Proceedings of IEEE International Conference on Cluster Computing (CLUSTER). 2020. pp. 329–339. https://doi.org/10.1109/CLUSTER49012.2020.00047.
19. Götz AW, Williamson MJ, Xu D, Poole D, Le Grand S, Walker RC. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized born. J Chem Theory Comput. 2012;8(5):1542–1555. https://doi.org/10.1021/ct200909j.
20. Salomon-Ferrer R, Götz AW, Poole D, Le Grand S, Walker RC. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. J Chem Theory Comput. 2013;9(9):3878–3888. https://doi.org/10.1021/ct400314y.
21. Shu R, Yuan M, Hanna H, Rawat PS, Cheng S. Towards Effective GenAI Multi-Agent Collaboration: Design and Evaluation for Enterprise Applications. arXiv preprint arXiv:2412.05449. 2024.
22. Ikizawa S, Morishima H, Hori T, Harada R, Takahashi K, Yamamoto E, Kitao A. PaCS-Toolkit: Software suite for enhanced conformational sampling and pathway search of biomolecular systems. J Phys Chem B. 2024;128(15):3631–3642. https://doi.org/10.1021/acs.jpcb.4c01271.
23. Tran KT, Dao D, Nguyen MD, Pham QV, O'Sullivan B, Nguyen HD. Multi-Agent Collaboration Mechanisms: A Survey of LLMs. arXiv preprint arXiv:2501.06322. 2025.
24. Kusumegi K, et al. Scientific production in the era of large language models. Science. 2025;390:1240–1243. https://doi.org/10.1126/science.adw3000.
25. Nguyen SS, et al. Review of current and potential uses of large language models in engineering. Front Educ. 2025;10:1649650. https://doi.org/10.3389/feduc.2025.1649650.
26. Editorial. The rise of large language models. Nat Comput Sci. 2025;5:689–690. https://doi.org/10.1038/s43588-025-00890-x.