NWChem: Description and Software Architecture

Karol Kowalski
Pacific Northwest National Lab
karol.kowalski@pnnl.gov

Sriram Krishnamoorthy
Pacific Northwest National Lab
sriram@pnnl.gov

Q: How many developers are involved and how is the development structured? (co-located team, post-docs, community contributions, etc.)

A co-located team of 5 (+ 2 post-docs) and many contributors

Q: Language: Fortran 77, C

Q: *Lines of code:* Several million lines of code

Q: Primary methods:

Gaussian-basis Density Functional Theory
Time-dependent Density Functional Theory
Plane-wave Density Functional Theory
Many-body perturbation theory
Coupled Cluster
Multi-scale approaches (QM/MM, embedding methods)

Q: Types of problems/domains/science application problems

NWChem addresses a wide range of computational chemistry problems including atmospheric chemistry, biochemistry, catalysis, and energy conversion vs. energy storage.

Q: Scale of resources commonly used for production runs

Depending on usage needs, production runs range in scale from desktops to leadership class computing facilities. In large-scale user facilities, runs can vary in scale from single node to several hundred thousand cores.

Q: Supercomputers regularly used

EMSL Cascade, NERSC Edison, OLCF Jaguar/Titan, TACC

Q: Libraries/tools for prototyping and production science campaigns

Both prototyping and production runs employ the same set of libraries and tools: blas, scalapack, Global Arrays, MPI, OpenMP/OpenACC, and CUDA.

Q: Describe efforts to develop code (application, library, etc.) portable across

diverse architectures.

Overall code builds on MPI/GA/OpenMP and is portable across compute platforms. Compute-intensive parts are either OpenMP loops or calls to optimized libraries such as blas and lapack. Further, whole-program optimizations (operation minimization, communication minimization, etc.) are done through code-generation and topology-aware algorithm design.

Q: Where were the abstractions?

One-sided communication, global arrays, tensors, tensor contraction expressions, multi-dimensional FFTs, parallel eigen solvers, etc.

Q: How much code re-use was possible? If something was not possible, please describe why.

We are working towards combining an evolutionary and revolutionary approach to preparing the code for future platforms. The evolutionary approach includes OpenMP and data-flow annotations that expose additional information to be exploited by parallel programming models while preserving the base semantics of the algebraic operations. The revolutionary approach involves the design of new algorithms and abstractions that enable runtime introspection and adaptation. These new algorithms and implementations are being designed in an incremental fashion to ensure correctness at every step of the development process.

Q: What successes have you had with performant code across difference architectures? Were the same algorithms applicable at all across the architectures?

We have had considerable success with achieving good performance across multiple supercomputing platforms. This is evidenced by the fact that NWChem had the first quantum many-body codes that scaled to full machine scales and utilized NVIDIA GPUs and Intel MICs. However, we expect additional effort for the upcoming supercomputing platforms due to deeper memory hierarchies, continued increase in compute-to-communication and compute-to-memory ratios, and changes in the nature of compute resources (heterogeneity and wider SIMD units).

Q: What approaches did you reject and why? What was the leading contender rejected?

While attractive, we found that end-to-end code generation precluded evolutionary development, correctness testing, and incorporation of optimizations based on expert knowledge. Therefore, we are pursuing a more evolutionary approach.

Q: What is your greatest fear going to exascale for application portability and functionality?

Beyond performance challenges, the most fundamental challenge is sustained funding. Over the years, we have made significant advances in our understanding of the algorithms required to efficiently implement the key methods. These ideas have been prototyped and published in peer-reviewed venues. However, securing sufficient funding to translate these ideas and prototypes into production quality implementations has been a major challenge.