Edoardo Aprà

Software needs for Quantum Chemistry Software

Acknowledgments

- Part of this work is funded by the U.S. Department of Energy, Office of Advanced Scientific Computing Research and by the division of Basic Energy Science, Office of Science, under contract DE-AC05-00OR22725 with Oak Ridge National Laboratory.
- This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory under contract DE-AC05-00OR22725.
Outline

- NWChem and Madness introduction
- List of Software requirements

Why Was NWChem Developed?

- Developed as part of the construction of the Environmental Molecular Sciences Laboratory (EMSL) at Pacific Northwest National Lab
- Designed and developed to be a highly efficient and portable Massively Parallel computational chemistry package
- Provides computational chemistry solutions that are scalable with respect to chemical system size as well as MPP hardware size
What is NWChem used for?

- Provides major modeling and simulation capability for molecular science
  - Broad range of molecules, including biomolecules, nanoparticles and heavy elements
  - Electronic structure of molecules (non-relativistic, relativistic, structural optimizations and vibrational analysis)
  - Increasingly extensive solid state capability (DFT, plane-wave, CPMD)
  - Molecular dynamics, molecular mechanics

Molecular Science Software Molecular Science Software Group

Interface Between the User and the Software

User

Science

Computer

NWChem

Global Arrays

Interface with the Science

Interface with the Computer
GA Tools Overview

- **Shared memory model** in context of distributed dense arrays
- **Complete environment** for parallel code development
- **Compatible with MPI**
- **Data locality control** similar to distributed memory/message passing model
- **Compatible with other libraries**: ScaLapack, Peigs, ...
- **Parallel and local I/O library** (Pario)
  - *De-Facto* Standard communication library for Quantum Chemistry Software (NWChem, Molpro, Molcas, MPQC)
  - Other fields expressed interest: Fusion, Nuclear Physics, Astrophysics, Climate

**Structure of GA**

Application programming language interface

- Fortran 77
- C
- C++
- Python
- Babel

Global Arrays and MPI are completely interoperable. Code can contain calls to both libraries.

- **Message Passing**
  - Global operations
  - Portable 1-sided communication
  - put, get, locks, etc

- **system specific interfaces**
  - LAPI, GM, threads, Q5net, IB, SHMEM, Portals
NWChem Architecture

- **Generic Tasks**
  - Object-oriented design
  - abstraction, data hiding, APIs

- **Molecular Calculation Modules**
  - Parallel programming model
  - non-uniform memory access, Global Arrays, MPI

- **Infrastructure**
  - GA, Parallel I/O, RTDB, MA, Peigs, ...

- **Program modules**
  - communication only through the database
  - persistence for easy restart

---

Gaussian DFT computational kernel
Evaluation of XC potential matrix element

\[ \rho(x_q) = \sum_{\mu\nu} D_{\mu\nu} \chi_\mu(x_q) \chi_\nu(x_q) \]
\[ F_{\lambda\sigma} \pm = \sum_{\alpha} w_\alpha \chi_\lambda(x_q) \nabla_{\rho(x_q)}^x \rho(x_q) \chi_\sigma(x_q) \]

my_next_task = $\text{SharedCounter()}$

\[
\begin{align*}
\text{do } i=1, \text{max}_i \\
\text{if}(i \text{.eq.} \text{my_next_task}) \text{ then} \\
\text{call ga\_get()} \\
\text{(do work)} \\
\text{call ga\_acc()} \\
\text{my_next_task} = \text{SharedCounter()}
\end{align*}
\]

endif

enddo

barrier()

Both **GA operations** are greatly dependent on the communication **latency**
Multiresolution Adaptive Numerical Scientific Simulation

Ariana Beste¹, George I. Fann¹, Robert J. Harrison¹,², Rebecca Hartman-Baker¹, Shinichiro Sugiki¹
¹Oak Ridge National Laboratory
²University of Tennessee, Knoxville

In collaboration with

Gregory Beylkin⁴, Fernando Perez⁴, Lucas Monzon⁴, Martin Mohlenkamp⁵ and others
⁴University of Colorado
⁵Ohio University

harrisonrj@ornl.gov

Multiresolution chemistry objectives

- Complete elimination of the basis error
  - One-electron models (e.g., HF, DFT)
  - Pair models (e.g., MP2, CCSD, …)
  - Bound and continuum states on equal footing
- Correct scaling of cost with system size
- General approach
  - Readily accessible by students and researchers
  - Higher level of composition
  - Direct computation of chemical energy differences
- New computational approaches
  - Fast algorithms with guaranteed precision
MADNESS Runtime Objectives

- Scalability to 1+M processors ASAP
- Runtime responsible for
  - scheduling and placement,
  - managing data dependencies,
  - hiding latency, and
  - Medium to coarse grain concurrency
- Compatible with existing models
  - MPI, Global Arrays
- Borrow successful concepts from Cilk, Charm++, Python
- Anticipating next gen. languages

Key elements

- *Futures* for
  - hiding latency and
  - automating dependency management
- Global names and name spaces
- *Non-process centric* computing
  - One-sided messaging between objects
  - Retain place=process for MPI compatibility
- Dynamic load balancing
I/O Requirements

- I/O models in NWChem
  - local I/O (e.g. internal HD in a node of a cluster)
  - Parallel I/O
- I/O usage:
  - Checkpointing (size: \(O(N^2)\) where \(N\) is \(~10^4\))
  - Scratch data: compute and written once, read many times (size: ask much as we can get)
- MPI/IO
- HDF5
- Reliability (parallel filesystems …)

1-sided Libs req

- GA is used for the bulk of NWChem comm.
- ARMCI
- GPC
- Gasnet
MPI Requirements

- 64-bit integers (again!)
- Compatibility with other communication libraries in use.

Multicore programming

- Task based parallelism (Intel Threading Building Blocks ?)
- Compatible with MPI and other communication libraries
- Profiling & debugging tools
Linear Algebra - Serial

- Serial (e.g. Lapack/Blas)
  - Requirement: ease of installation
  - Optimized perf. Other large range of N
  - 64-bit integers (if possible)

Linear Algebra - Parallel

- Distributed arrays: interface with GA
- Installation
- Extensive use of dense symmetric eigensolver
  - MRRR algorithm in Scalapack (PDSYEVX by C. Voemel)
Optimization/Tuning

- Profiling tools
  - Gprof, TAU
  - PAPI
  - Code coverage & instrumentation

Quality/Assurance?

- QA suite in NWChem
  - Perl scripts
  - Reference output files
  - Checks results’ correctness
  - Database of performance on various HW?
Debugging

- Parallel debuggers
- Printf?

Thanks

- NWChem group and GA Group - PNNL
- Robert Harrison
- NCCS & CCS – ORNL
- Workshop organizers
NWChem HW and SW requirements

- low latency and high bandwidth for
  - I/O
  - communication
- availability of large amount of aggregate memory and disk
- flat I/O model using local disk
- 64-bit addressing (required by highly correlated methods)
- extensive use of Linear Algebra:
  - BLAS
  - FFT
  - Eigensolvers
- use of other scientific libraries (e.g. MASS, ACML_vec for exp and erfc)
NWChem Porting issues on the XT3

- Re-use of existing serial port for x86_64 (compilers)
- Communication library: GA/ARMCI
  - Two ARMCI ports: CRAY-SHMEM & Portals
- Made Pario library compatible with Catamount (using glibc calls)

NWChem Porting - II

- Stability issue with SHMEM port
  - Uncovered portal problem: Cray SHMEM group provided workaround
  - NWChem crashes the whole XT running large jobs
- Performance of SHMEM port
  - Latency ~ 10 μsec
  - BW
    - Contiguous put/get ~ 2GB/sec
    - Strided put/get ~ 0.3GB/sec
- ARMCI using Portal in progress