Accelerating performance of the HPC electron collisions R-matrix code PFARM on the Xeon Phi

AG Sunderland(1), G Corbett(1), M Lysaght(2) and M Plummer(1)

(1) Scientific Computing Department, STFC Daresbury Laboratory, UK

(2) Novel Technologies Activity and the Intel Parallel Computing Centre, ICHEC Dublin, Ireland
R-matrix Theory

- Basis of computer programs that describe a wide range of atomic, molecular and optical processes. Numerically very stable
- Ab initio solution of full Schrodinger equation using CI
- Successful in treating a wide range of collision phenomena
  - Scattering of electrons, positrons or photons with atomic and molecular targets
  - Multiphoton interactions with atoms (and now/soon molecules)
- The PFARM code developed for atoms has recently been adapted for molecular codes
- Developed by CCP2/CCPQ. Optimization projects - dCSE, PRACE.
- **Real world applications include:**
  - Astrophysics: stars, interstellar medium (shocks)
  - Atmospheres, atomic and molecular plasmas (nuclear fusion, laser-produced plasmas, lighting)
  - Radiation damage to DNA (electron collisions with DNA bases)
The R-matrix method

• Configuration space divided into ‘inner’ and ‘outer’ regions by a sphere
• Inside: all electron (lepton) calculation, CI, exchange, spherical tensor algebra, Hamiltonian formation and diagonalization (with non-vanishing orbitals on the boundary)
• Outside: multipole potentials (from ‘inside’), coupled differential equations, propagation to asymptotic region, possible frame transformations
• Inside: energy-independent; outside: energy-dependent

Partition of Configuration Space
The R-matrix method

- Configuration space divided into ‘inner’ and ‘outer’ regions by a sphere
- Inside: all electron (lepton) calculation, CI, exchange, spherical tensor algebra, Hamiltonian formation and diagonalization (with non-vanishing orbitals on the boundary)
- Outside: multipole potentials (from ‘inside’), coupled differential equations, propagation to asymptotic region, possible frame transformations
- Inside: energy-independent; outside: energy-dependent

The parallelization of the code maps closely to this partitioning
PFARM: external and asymptotic regions
Baluja-Burke-Morgan (BBM)-based Implementation

2 Stage Parallelization of BBM approach in the external region:

- EXDIG Program (Modern Fortran):
  - Diagonalize Sector Hamiltonian matrices using ScALAPACK PDSYEVX (Blacs-based Data decomposition).

- EXAS Program (Modern Fortran):
  - For each scattering energy propagate using 3 functional groups:
    - Generate initial R-Matrix PDGEMM (Data decomposition).
    - Propagate R-Matrix across each sector in pipeline (Control decomposition). DGEMM, DGETRF, DGEMM
    - Calculate thermally averaged collision strengths. Serial S.V.D. (Task Farmed).
EXAS Stage

Serial, OpenMP and MPI versions

Internal Region

External Region

Asymptotic Region

Parallel Diagonalizations of Large Symmetric Sector Hamiltonian Matrices
EXAS Stage

- Outer code PFARM, scales to 1000s of cores: now used with both atomic inner region and UKRmol
  - full parallel diagonalization (ScaLAPACK), multiple MPI task propagation and pipelining:
Optimized code – overall 150% performance improvement on 8132 cores (I/O and diag improvement)
Candidates for Offloading

• Four dense linear algebra operations identified as candidates for offloading to Xeon Phi:
  – Matrix Multiply in EXAS (dgemm)
  – Linear Solver in EXAS (dgetrf)
  – Singular Value Decomposition in EXAS (dgesvd)
  – Symmetric Eigensolver in EXDIG (dsyevd)
MKL & MAGMA

• Intel® Math Kernel Library (MKL)
  – A library of highly optimized, extensively threaded math routines including BLAS library, LAPACK, ScaLAPACK, sparse solvers, Fast Fourier Transforms library, vector math, and more.

• Matrix Algebra on GPU and Multicore Architectures (MAGMA)
  – similar to LAPACK but for heterogeneous/hybrid architectures, starting with current "Multicore+GPU" systems.

http://icl.cs.utk.edu/magma/index.html
Offloading in MKL

• Code with highly parallel phases
• Code runs on Xeon Host until a sufficiently computationally heavy region reached
• Data transfer to Phi and execution runs there
• Data transferred back to Host
• Auto or user defined

Image modified from:
• Slidecast 3/3 – PRACE Summer School on Code Optimisation for Multi-Core and Intel MIC Architectures – Workshop on MIC
• Intel MIC Architecture – Intel MIC HW/SW Architecture
Offloading in MAGMA

- Client/Server model
- Server must be active on the Phi before offloading can occur
- Small non-parallelizable tasks are scheduled on the host, whilst larger, more parallelizable tasks, (e.g. Level 3 BLAS), are scheduled on the Intel Xeon Phi.
- Unlike MAGMA-GPU, no supplier-provided Fortran interfaces

Image modified from:
- Slidecast 3/3 – PRACE Summer School on Code Optimisation for Multi-Core and Intel MIC Architectures – Workshop on MIC
- Intel MIC Architecture – Intel MIC HW/SW Architecture
DGEMM Performance on Intel Xeon Phi

DGEMM Performance on Intel Xeon Phi co-processor SE10/7120 at DL

DL Xeon Phi co-processor SE10/7120
2x8 cores on Host
61 cores on Xeon Phi
DGETRF & DGESVD Performance on Intel Xeon Phi

EXAS matrices too small at present to gain advantage from offloading DGETRF & DGESVD

DL Xeon Phi co-processor SE10/7120
2x8 cores on Host
61 cores on Xeon Phi
DSYEV (Eigensolver) Performance on Intel Xeon Phi

DSYEV Performance (GFlops) on Intel Xeon Phi co-processor SE10/7120 at DL

- MKL Xeon Host (32,C,F)
- MAGMA Xeon Phi (240,C,F)
- MAGMA Xeon Phi (60,C,F)

Matrix Dimensions vs. GFlops for various matrix dimensions.
Performance of EXDIG with Xeon Phi acceleration using MAGMA

The Xeon host calculations are run using MKL v 1.1 with 32 threads and the Xeon Phi MAGMA v1.1.0 calculations use 240 threads.
EXAS on Xeon Phi

- Matrices generally too small for effective offloading

- Collapse a pipeline communicator and do the work for the pipeline on the Phi (shared memory parallelism), hosts can continue to run alongside use standard distributed pipelines
  - Replacement coding strictly localized with a clear interface to the main code within the pipelining modules

- Dominated by dense linear algebra operations. Originally undertaken with MKL (serial, shared memory tasks, distributed memory)

- The new version of EXAS is fully heterogeneous, ie hosts and Intel Phis perform separate work simultaneously
Schematic of original EXAS implementation for a single pipeline (top) and schematic of new EXAS implementation enabled for Fionn Xeon Phi machine (bottom)
Performance analysis of original implementation of EXAS (left) and new implementation of EXAS on Xeon Phi (right) using the Intel Trace Analyzer and Collector (ITAC) profiler
Summary

- Optimised Intel Xeon Phi port of PFARM (EXDIG) incorporating MAGMA MIC for accelerated parallel eigensolvers. (~2x speed-up overall)

- A new version of PFARM (EXAS), restructured for accelerated R-matrix propagation pipelining. Tested and tuned on the Intel Xeon Phi and also applicable to GPUs (M.L. expects speed-up once communication bottlenecks reduced)

- Detailed analyses of MKL and MAGMA MIC numerical library routines performance on Intel Xeon Phi architectures.

- EXAS undergoing further optimization: currently a host in one functional group offloads to a Phi in another group with slow comms
  - Fully flexible MPI/OpenMP version – Distribute complete multiple functional groups efficiently across Host/Phi, exploit OpenMP 4.0 task model
  - Preparation for Knights Landing
  - MAGMA MIC, MKL Offloading for PDGEMM (large rectangular matrices)