Optimization of the Selected Quantum Codes on the Cray X1

L. Bolikowski, F. Rakowski, K. Wawruch,
M. Politowski, A. Kindziuk, W. Rudnicki,
P. Bala, M. Niezgodka

June 30, 2004

Abstract

In this paper we present optimization of the selected Quantum Chemistry applications on Cray X1, and performance of selected bioinformatic code created from scratch for Cray X1. In particular we have optimized some density functional – molecular dynamics codes used for the investigation of the properties of molecular and biomolecular systems, and Smith-Waterman algorithm with appropriate filter.

1 Introduction

ICM is Interdisciplinary Centre for Mathematical and Computational Modeling, located at Warsaw University in Poland. To serve its purpose for the scientists using our supercomputers, ICM has worked on porting and optimizing selected Quantum Chemistry applications on Cray X1.

The applications we have selected for development are:

- **DFTB** – *Dense Functional – Tight Binding Method* code;
- **VASP** – *Vienna Ab-Initio Simulation Package*;
- **GROMOS** – molecular dynamics code;
- **CHARMM** – *Chemistry at Harvard Macromolecular Mechanics*, a program for macromolecular energy, minimization and dynamics calculation;
- **GAMESS** – *General Atomic and Molecular Electronic Structure System*, ab-initio package;
- **SIESTA** – *Spanish Initiative for Electronic Simulations with Thousands of Atoms*, ab-initio package.

ICM, in addition to porting and optimizing existing code, performs in-house development of new bioinformatic codes. The software created form a scratch at ICM is Smith-Waterman algorithm implementation with appropriate filters, for protein sequences.

Due to different priorities and complication level of the codes we have worked on, we may assign the following statuses to the applications:

Package	Status	Description
DFTB	Production, some development	Good MSP performance. Multistreaming optimizations and SSP version in development.
VASP	Production, some development	Very good SSP performance. Multistreaming and parallel optimizations in development.
GROMOS	Ported, partially optimized	Reasonable SSP performance. SSP, MSP and parallel optimizations in development.
GAMESS	Ported, partially optimized	Poor performance. Vectorization partially completed.
SIESTA	Ported, partially optimized	Poor performance. Vectorization partially completed.
CHARMM	Ported, not optimized	Very poor performance. Profound code modifications required for vectorization.

2 DFTB

The very first code ported and optimized for Cray X1 by ICM development team is DFTB, *Density Functional – Tight Binding Method* code. The DFTB code is written in Fortran, by prof Th. Frauenchaim on University of Paderborn. It serves for the approximate quantum calculations. This is the very fast quantum potential generator which do not calculate the Hamilton and Overlap matrix elements at each step of the SCF cycle, but the convergence is reached on the level of Mullikan Charges.

The model system for test calculations was an active site of PKA kinase, consisted of about 400 atoms. The single point energy calculations were performed, and the convergence was reached after 20 SCF cycles.

The performance results of the DFTB code on Cray X1 were compared to the single-CPU results of PC with Intel Pentium IV 2.66 GHz. The shortest user time achieved was 619s.

The straightforward recompilation (MSP mode) lead to very poor performance on Cray X1. The user execution time (97% of elapsed time), 471s, was not competitive to the standard PC computers, though floating point performance, 336Mflops, suggested wide area for improvements (the theoretical peak of 12.8Gflops).

The profiling of the straightforwardly compiled DFTB show, that `atomenerg` procedure utilizes 82% of computation time.

The vectorization and multistreaming of the code allowed to reduce the user execution time to 48s, with total one MSP performance of 3213Mflops. In comparison to the unoptimized code, the execution time was reduced ten times and gave results 15 times better than on reference PC. The improvement was obtained by the redesign of the loop in the `atomenerg` procedure. All loops were redesigned to longer vector length and to provide better data alignment for memory access.

After further profiling it turned out, that all procedures that utilize more than 4% of the calculation time are Linear Algebra procedures, already optimized for Cray.

We are planning to release SSP version of the code, and further work to improve multistreaming, as we are achieving 25% of the peak performance.

3 VASP

VASP, *Vienna Ab-Initio Simulation Package*, provides ab-initio quantum-mechanical molecular dynamics simulation tools. It is developed at Vienna University in Fortran 77 with MPI parallel extensions.

The standard VASP package is ported to Cray C90/J90 vector supercomputers and initially did not compile on Cray X1. After porting the code to Cray X1 and obtaining working executables, it turned out that performance is significantly lower than on reference PC (Intel Pentium IV 2.66GHz).

The optimization of the code involved loops' redesign to allow vectorization (removing data dependencies though profound changes of loops' structure), to allow longer vector length and to provide better data alignment for memory access. VASP does not have well defined computational kernel: depending on the kind of test and options used, the computation time is scattered through many procedures.

After vectorization, we have obtained performance on one SSP processor from 800Mflops to 1582Mflops on real life tests, that is from 25% to 50% of the SSP peak performance. The MSP speedup, in comparison to SSP performance, ranges from 20% to 150%, depending on the parts of the code used by the test, as we have not completed multistreaming of some parts of the code. The predicted speedup after completing multistreaming is 3 times the SSP performance.

The parallel scalability with MPI communication library is quite poor. On 2 SSP processors average speedup is 1.71x (ranging from 1.66x to 1.98x), on 4 SSP processors – 2.60x (ranging from 1.60x to 3.38x), and breaking down on more than 4 SSP processors.

We are going to rewrite communication to Co-Array Fortran to reduce communication overhead to improve, as the result, scalability, and to enable scaling above 4 SSP processors.

4 GROMOS

The GROMOS is well know molecular dynamics package for biomolecular systems. The code is written in FORTRAN 77.

The most time consuming part of the ode is evaluation of the nonbonded interactions between atom pairs. In order to reduce computational effort, the GROMOS uses pair-list technique, eg. the list of interacting pairs is evaluated and than used for evaluation of the forces instead of looping over all atom pairs in each time step.

The pair-list is evaluated less frequent than forces (usually every 10th time-step). We have performed 500 time steps for 55000 atoms system.

Vector Code

The vector version of the nonbonder routines exists and have been used on Cray Y-MP. This version has been compiled and evaluated. Since the code is relatively old, the `ISRCHEQ` was used from the GROMOS library.

The SSP code, after vectorization, achieved performance of 531Mflops (99.21% of the vector operations) and average vector length of 46. The MSP executable achieved performance of 636Mflops (99.11% of the vector operations) and average vector length of 31, as the development already done do not include multistreaming optimization.

Parallel Code

We have used parallel version of the code, parallelized by the distribution of the calculation of the nonbonded forces and pair list generation. The parallel version has been developed by P. Bala and T. W. Clark using PFortran and Co-Array Fortran languages. In particular, Co-Array Fortran version has been used on X1.

The scalability of the parallel code is summarized in the table below. The test consisted of 500 steps.

# of SSP	Elapsed time [s]	Ratio
1	799.299	1.0
2	498.299	1.6
4	320.413	2.5
8	235.492	3.4

Performed parallelization can be improved by significant changes of the code, like implementation of domain decomposition instead of force decomposition.

The second issue is single node performance. The MD codes do not fit well to the vector architecture because most of the work is performed in short loops. 74% of the time is spent for calculation of nonbonded forces, and 12% for SHAKE subroutine.

The scalar/vector code performance requires further improvements. The code is reasonably well vectorized (95% of vector operations), though the overall performance is low due to short loops and low vector load (26). The performance achieved is 147Mflops.

Unfortunately there is no single computational kernel, the computations time is distributed over large number of code lines located in the different loops. Further improvement of the performance requires significant changes in the code. The short loops cannot be removed by the simple code rearrangements, the changes in the algorithm used for evaluation of nonbonded forces must be applied.

Much better performance of the vector code suggests attempt to use it also in parallel version, but this requires some work.

5 GAMESS

GAMESS, *General Atomic and Molecular Electronic Structure System*, is an ab-initio quantum chemistry package developed by Gordon Research Group at Iowa State University.

The code is written in Fortran 77. After porting and some optimizations, we have achieved performance of 3-10% of peak SSP processor performance (the results for MSP processors were worse). As the code is very large, it was not yet fully vectorized. The current performance results of some real-life tests on single SSP processor are presented in the following table:

Test name	Performance [Mflops]	Description
exam22	240.547	UHF + UMP2 gradient
exam12	76.705	Closed shell DFT geometry optimization
exam32	366.249	Coupled cluster test
exam31	160.004	PCM test case

The performance of some artificial tests, utilizing best-vectorized parts of code, achieve over 1500Mflops on single SSP processor. It should be possible to optimize the code to achieve approximately 25% of peak performance, though it requires substantial amount of work.

The MSP version of the executable will be developed after completing vectorization.

6 SIESTA

SIESTA, *Spanish Initiative for Electronic Simulations with Thousands of Atoms*, ab-initio quantum chemistry package, is written in Fortran 90 with parallel version using MPI library.

After porting and removing bugs that caused numerical errors, we have obtained single SSP processor performance of 7-15% of peak. We are focusing now on improving vectorization of the code. After completing it, we will work on MSP version of the executables. We are going to investigate, if the parallel communication should be rewritten in Co-Array Fortran to improve scalability.

7 CHARMM

CHARMM, *Chemistry at Harvard Macromolecular Mechanics*, a program for macromolecular simulations, including energy minimization, molecular dynamics and Monte Carlo simulations, is a huge Fortran 77 code with parallel communication provided by the MPI library.

The version we have ported to X1 is CHARMM c29b1. We have created SSP binaries that works with some sample tests, though with poor performance (less than 100Mflops) and with some numerical errors on some tests.

The main problem with vectorizing CHARMM is that it includes multiple hardly vectorizable loop elements, like function calls and conditional expressions. The CHARMM code includes some vector elements, though they are outdated and sometimes intruded by some scalar additions.

We are focusing now on checking, what CHARMM functionality is not used by most of the users. We will remove corresponding parts of the code to simplify the whole code logical structure, and then we will vectorize the remaining part.

8 Smith-Waterman Algorithm Implementation

One of the scientific projects at ICM requires clustering of whole known sequence database (over 2 million sequences) to speed up searching the database and to find biologically significant similarities.

To achieve it, we have to create code that would be able to search through whole database in less than 1 hour on 1 SSP processor for average reference sequence.

Due to some biological reasons, we have decided to use Smith-Waterman algorithm. We are able to process 25M cells per second on 1 SSP (on the fastest PCs we can obtain performance of up to 7M cells per second).

As such processing speed does not allow us to reach our goal, we have developed filter that decreases the number of sequences that are analyzed by Smith-Waterman algorithm by the factor of 10. The filter, depending on the parameters, processes 80M to 160M cells per second on 1 SSP (on PC, up to 20M cells per second). We are able to achieve such processing speed thanks to *Bit Matrix Multiply* unit usage.

9 Summary

After working on multiple scientific codes on Cray X1, we may draw some final conclusions.

64 bits are still a problem

Most of the codes causes some problems when compiling in 64-bit mode. 32-bit mode on Cray X1 is great advantage to enable fast application porting. The fact of application's existence on any 64-bit platform does not imply, that it does not cause problems – differences in variable types' length between 64-bit architectures may allow some deeply hidden bugs to be emerging on Cray X1.

Cray to IEEE floating point transition is a problem

Changing of the numeric format for floating point numbers from Cray proprietary to the IEEE standard may cause problems. Many codes include special vectorized code variants, that are written for Cray PVP supercomputers. Such code may take advantage of Cray numeric format, leading to numeric errors after porting to Cray X1. Modifying such code may be difficult, as it usually is extremely optimized using some tricks and not documented or commented. Switching to scalar code variant is not acceptable solution, as causes significant performance decrease.

3 levels of parallelism are complicated

The three parallelism levels (inside MSP, inter-MSP, inter-node) require bigger programmer's knowledge, caution and experience, though enable creating cutting-edge optimal applications for Cray X1.

High sustained performance

As we have shown on examples in this article, it is possible to optimize existing scientific codes on Cray X1 to achieve 25-50% of peak performance on real-life jobs, with opportunities to achieve more than 50% after more optimization efforts.