УДК 519.682.3 Дата подачи статьи: 29.09.14
DOI: 10.15827/0236-235X.109.043-048

ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ МАТЕМАТИЧЕСКОЙ БИБЛИОТЕКИ INTEL MKL В ПАРАЛЛЕЛЬНЫХ ПРОГРАММАХ НА ЯЗЫКЕ T++ ДЛЯ T-СИСТЕМЫ С ОТКРЫТОЙ АРХИТЕКТУРОЙ

(Работы выполнены в рамках Программы фундаментальных научных исследований ОНИТ РАН «Архитектурно-программные решения и обеспечение безопасности суперкомпьютерных информационно-вычислительных комплексов новых поколений» и НИР «Методы и программные средства разработки параллельных приложений и обеспечения функционирования вычислительных комплексов и сетей нового поколения»)

В.А. Роганов, научный сотрудник, var@pereslavl.ru;
А.А. Кузнецов, научный сотрудник, tonic@pereslavl.ru;
Г.А. Матвеев, ведущий инженер-исследователь, gera@prime.botik.ru;
В.И. Осипов, к.ф.-м.н., научный сотрудник, val@pereslavl.ru
(Институт программных систем им. А.К. Айламазяна РАН, ул. Петра I, 4а, г. Переславль-Залесский, 152021, Россия)

Реализация параллельных вычислений в большей мере является проблемой программного обеспечения. Самый распространенный подход к разработке параллельных программ основан на использовании программных пакетов типа MPI (*Message Passing Interface,* интерфейс передачи сообщений). При этом такой подход требует от разработчика большого объема знаний, а также значительных временных затрат на разработку и отладку параллельных программ. В разработанной в ИПС РАН Т-системе реализован подход, при котором большая часть решений по распараллеливанию принимается динамически в процессе выполнения программ. Входной язык для Т-системы – язык Т++, а приложения, разработанные для Т-системы, являются Т-приложениями или Т-программами.

В статье дается краткий обзор продуктов и компонентов математической библиотеки Intel Math Kernel Library (Intel MKL), которая содержит большой набор математических функций и может использоваться в параллельных Т-приложениях для ускорения процесса счета и достижения максимальной производительности. В работе рассматриваются режимы использования математической библиотеки Intel MKL на кластерах, имеющих процессоры Intel Xeon и один или несколько ускорителей (сопроцессоров) Intel Xeon Phi компании Intel. Приводится несколько демонстрационных примеров использования библиотеки на языках Си и Т++. В работе показано, как использование математической библиотеки влияет на эффективность выполнения параллельных Т-программ. Все эксперименты проводились на энергоэффективном суперкомпьютере «РСК Торнадо ЮУрГУ» Южно-Уральского государственного университета.

Ключевые слова: *язык программирования Т++, OpenTS, Т-система, параллельное расширение Си++, суперкомпьютеры, математическая библиотека Intel MKL.*

Известно, что Т-система представляет собой концепцию автоматического динамического распараллеливания, в которой сочетаются наиболее удачные черты функционального программирования, dataflow-систем и традиционных языков и методов программирования [1–2].

Современная реализация концепции Т-системы (OpenTS [3]) обладает открытой и масштабируемой архитектурой, легко адаптируемой к стремительно меняющимся аппаратным платформам современных суперкомпьютеров. Поддерживаемый системой OpenTS входной язык программирования Т++ является синтаксически и семантически гладким расширением языка программирования Си++ [4–7].

Математическая библиотека Intel MKL

Библиотека Intel Math Kernel Library (Intel MKL) – это математическая библиотека, которая содержит большой набор математических функций и может использоваться в параллельных приложениях для ускорения процесса счета и достижения максимальной производительности [8].

Библиотека Intel MKL, адаптированная для процессоров компании Intel, предлагает высокооптимизированные и многопоточные функции, которые реализуют многие типы операций. Библиотека работает как на 32-разрядных (IA-32), так и на 64-разрядных (Intel 64) архитектурах.

Библитека включает в себя следующие продукты и компоненты для работы с линейной алгеброй: BLAS, SparseBLAS, PBLAS, LAPACK/ ScaLAPACK, PARDISO, Iterative sparse solvers.

BLAS (Basic Linear Algebra Subprograms) – математическая библиотека, содержащая набор функций для реализации операций линейной алгебры (обработка матриц и векторов).

Операции линейной алгебры разделены на три уровня: 1-й уровень – векторно-векторные операции, 2-й уровень – матрично-векторные операции, 3-й уровень – матрично-матричные операции. На библиотеке BLAS построен тест Linpack.

SparseBLAS (Sparse Basic Linear Algebra Subroutines) – расширение библиотеки BLAS для работы с разреженными векторами и матрицами.

PBLAS (Parallel BLAS) – параллельная, основанная на использовании технологии MPI библио-

тека, адаптированная для кластеров с распределенной памятью.

LAPACK (Linear Algebra PACKage) – пакет, содержащий набор подпрограмм для решения линейных алгебраических уравнений. Предназначен для работы с плотными матрицами.

ScaLAPACK – пакет, представляющий собой набор параллельных подпрограмм для решения задач линейной алгебры.

PARDISO (PARallel DIrect Solver) – библиотека, предназначенная для решения систем линейных уравнений с большими разреженными матрицами.

Iterative sparse solvers – итерационные решатели для разреженных матриц.

FFTs – модули быстрого преобразования Фурье.

ClusterFFT – кластерная реализация библиотеки быстрого преобразования Фурье, которая содержит следующие компоненты:

– функции математической статистики: эксцесс распределения, коэффициент вариации, квантили и порядковая статистика, минимум/максимум, вариация/ковариация;

– аппроксимация (выравнивание) и интерполяция данных: сплайны, интерполяция и др.;

– другие компоненты: тригонометрические, логарифмические функции, взятие корня, генераторы случайных чисел, функции округления и т.д.

Библиотека Intel MKL хорошо оптимизирована для параллельной работы на кластерных установках, имеющих на узлах многоядерные процессоры с ускорителями Intel Xeon Phi, и на SMP-системах (системах с общей памятью). Библиотека поддерживает как гетерогенный режим вычислений, когда код библиотеки выполняется одновременно на хост-процессоре узла и на ускорителе (сопроцессоре), так и однородный режим вычислений, когда код библиотеки выполняется одновременно только на сопроцессорах либо только на хост-процессорах. Для использования библиотеки Intel MKL приложение может быть собрано компилятором Intel с указанием опции «-mkl».

Доступны три способа (режима) использования математической библиотеки Intel MKL:

– Automatic Offload mode – буквально «автоматическая выгрузка», используется при гетерогенном режиме вычислений;

– Offload mode supported by Compiler – выгрузка с поддержкой от компилятора; в этом режиме процесс выгрузки (переноса части вычислений) вычислительного кода на ускоритель задается в программе с помощью директив компилятора;

– Native mode – код библиотеки выполняется одновременно только на ускорителях (сопроцессорах).

Автоматическая выгрузка. Данный способ является самым простым в использовании математической библиотеки Intel MKL на кластерах, имеющих процессоры от Intel и один или несколько ускорителей (сопроцессоров). При использовании данного режима не требуется значительного изменения кода программы. Библиотека сама автоматически принимает решение об использовании ускорителя при вызове функций библиотеки на этапе выполнения программы. Решение принимается, исходя из текущей загрузки процессора и ускорителя, количества входных данных и параметров функции. Например, если в момент вызова какой-то функции ускоритель загружен другой работой, функция будет выполнена на хост-процессоре. В противном случае функция будет выполнена на ускорителе. Таким образом, библиотека Intel MKL выполняет автоматическую балансировку нагрузки на систему. Нагрузка на систему может как распределяться автоматически, так и задаваться программистом для достижения эффективного использования вычислительных ресурсов кластера.

Для включения режима необходимо вызвать функцию mkl_mic_enable(). Она позволяет автоматически выполнять (выгружать) на ускорителе Intel Xeon Phi код математических функций библиотеки Intel MKL.

Для включения режима можно воспользоваться и переменной окружения MKL_MIC_ENABLE, установив ее значение в «1».

Библиотека Intel MKL предоставляет функции для поддержки выполнения программ на ускорителях Intel Xeon Phi:

```
#include "mkl.h"
rc = mkl_mic_enable();
rc = mkl_mic_set_workdivision(MKL_TARGET_MIC,
MKL_MIC_DEFAULT_TARGET_NUMBER, 0.0);
rc = mkl_mic_set_workdivision(MKL_TARGET_MIC, 0,
MKL_MIC_AUTO_WORKDIVISION);
rc = mkl_mic_get_workdivision(MKL_TARGET_MIC, 0,
&wd);
rc = mkl_mic_set_max_memory(MKL_TARGET_MIC, 0,
mem_size);
rc = mkl_mic_set_device_num_threads(MKL_TARGET_
MIC, 0, num_threads);
rc = mkl_mic_set_offload_report(1);
rc = mkl_mic_free_memory(MKL_TARGET_MIC, 0);
rc = mkl_mic_disable();
```

Семейство функций mkl_mic_set_workdivision() позволяет задавать желаемое распределение нагрузки между хост-процессором и ускорителями Intel Xeon Phi. Дополнительные функции управления распределением нагрузки позволяют определить объем вычислительной работы в виде пропорции, которая будет распределена между хост-процессором и ускорителями Intel Xeon Phi. Объем работы задается вещественным числом и может изменяться в пределах от 0,0 до 1,0. Например, установка объема работы для хост-процессора в значение 0,5 означает, что нагрузка будет распределена следующим образом: половина вычислительной работы (50 %) будет выполнена на хост-процессоре, а вторая половина – на ускорителях (если в системе больше одного ускорите-

ля). Или, например, установка объема работы в значение 0,25 для ускорителя означает, что четверть вычислительной работы (25 %) будет выполнена на ускорителе, а остальная часть работы (75 %) будет отдана хост-процессору. Отметим, что такое указание распределения нагрузки на систему носит лишь рекомендательный характер.

Существуют также переменные окружения, с помощью которых можно управлять нагрузкой на систему. Например, в данной переменной окружения для нулевого ускорителя установлен объем работы в 25 % от общей нагрузки:

MKL_MIC_0_WORKDIVISION=0,25.

Для отключения режима используется функция mkl_mic_disable():

```
#include "mkl.h"
rc = mkl_mic_disable();
```

Выгрузка с поддержкой от компилятора. В этом режиме процесс выгрузки (переноса) вычислительного кода на ускоритель Intel Xeon Phi задается в программе с помощью директив компилятора.

Рассмотрим пример:

```
#pragma offload target(mic) \
   in(maxfct, mnum, mtype, phase) \
   in(n) \
   in(nrhs) \
   in(msglvl) \
   out(error) \
   in(pt:length(64)) \
   in(p_a:length(18)) \
   in(p_ia:length(9)) \
   in(p_ja:length(18)) \
   in(iparm:length(64)) \
   in(p_b:length(8)) \
   in(p_x:length(8)) \
   out(p_x:length(8) alloc_if(0))
   {
       MKL_INT idummy;
       pardiso(pt, &maxfct, &mnum, &mtype, &phase, &n,
      p_a, p_ia, p_ja, &idummy, &nrhs,
              iparm, &msglvl, p_b, p_x, &error);
   }
```

В данном примере дано описание входных и выходных данных для передачи в математическую функцию pardiso(), код которой будет выполнен на ускорителе. Функция pardiso() используется для решения большой разреженной линейной системы уравнений, поддерживает работу с различными типами матриц.

Следует отметить, что данный способ не гарантирует увеличения производительности вычислений.

Выполнение кода только на ускорителях. В данном режиме код библиотеки выполняется только на ускорителях Intel Xeon Phi, исключая хост-процессоры. Во время выполнения команды mpirun (mpiexec.hydra) создается однородная подсеть из вычислительных узлов, содержащих только ускорители, которые взаимодействуют между собой и обмениваются данными посредством MPI-сообщений.

Для создания исполняемого файла, который может быть выполнен на ускорителе, используется ключ «-mmic». Унаследованный код, предназначенный для выполнения на CPU, может быть легко собран для выполнения на ускорителе. Для этого достаточно при сборке приложения указать ключ «-mmic».

Более подробные сведения о режимах использования библиотеки Intel MKL можно узнать в работе [9].

Примеры использования математической библиотеки Intel MKL в параллельных программах на языке T++

Первый набор T-программ. Рассмотрим несколько простых T-программ. Первая T-программа состоит из трех T-функций: главной T-функции tfun int main() и двух вспомогательных – t1() и t2(). Каждая вспомогательная T-функция выполняет умножение двух матриц простым способом, используя тройной вложенный цикл, а затем в результирующей матрице считает сумму абсолютных значений всех ее элементов. Данное число возвращается T-функцией в качестве выходного (tout) готового значения. Матрицы, участвующие в операции умножения, инициализируются случайным образом. Главная T-функция осуществляет вызов вспомогательных T-функций и ожидает завершения их выполнения. Она получает от вспомогательных T-функций значения сумм абсолютных значений элементов результирующих матриц и вычисляет минимальное из двух значений. Результат выводится на экран. Отметим, что данная T-программа не использует возможности математической библиотеки Intel MKL.

Компиляция и компоновка кода. Соберем нашу T-программу для выполнения на хост-узлах и для выполнения на ускорителях Intel Xeon Phi.

Сборка T-программы для хост-узла выполняется командой

```
$ t++ -opt t-simple_mult_sum_test.tcc -o t-simple_mult_
sum_test
```

Сборка T-программы для сопроцессора Intel Xeon Phi выполняется командой

```
$ t++ -opt -mmic t-simple_mult_sum_test.tcc -o t-simple_
mult_sum_test.mic
```

Запуск T-программы на выполнение. Все эксперименты будем проводить на суперкомпьютере «РСК Торнадо ЮУрГУ», разработанном группой компаний «РСК» [10].

Вычислительный кластер представляет собой 480 вычислительных узлов и два сервера (управления и доступа пользователей), объединенных между собой с помощью транспортной сети Infiniband и двух Ethernet-сетей (мониторинга и управления заданиями). Вычислительные узлы имеют сквозную нумерацию вида «nodeXXX» (где X – число от 001 до 480). Каждый вычислительный узел имеет в своем составе

– сопроцессор Intel Xeon Phi с именем вида «nodeXXX-mic0»; внутри вычислительного узла возможна адресация по «mic0»;
– сетевой интерфейс Infiniband (имя Ib0 в пределах узла или nodeXXX-ib0 в пределах кластера);
– сетевой интерфейс Ethernet (eth0).

Для запуска MPI-программ (в нашем случае Т-программ) на выполнение используется планировщик задач SLURM [11]. MPI-программы могут быть запущены на кластере как в интерактивном, так и в пакетном режимах. В интерактивном режиме пользователь либо использует для запуска команду srun, либо самостоятельно выделяет необходимое количество ресурсов (вычислительных узлов), а затем выполняет команду mpiexec.hydra. В пакетном режиме пользователь использует утилиту sbatch. Программа становится в очередь и будет выполнена, как только станут доступными необходимые для ее выполнения ресурсы. Результат выполнения программы будет записан в файл slurm<номер_задачи_в_очереди>.out.

Для управления версиями различных библиотек и пакетов на кластере установлен пакет Environmental Modules. Он позволяет гибко настраивать переменные окружения для пользовательских и пакетных задач. Пакет состоит из модулей. Каждый модуль содержит всю информацию, необходимую для настройки окружения под конкретное приложение. Например, настраиваются такие переменные окружения, как PATH, MANPATH, INCLUDE, LD_LIBRARY_PATH. Модули можно подгружать или выгружать вручную.

Для просмотра списка уже загруженных модулей нужно выполнить команду

```
$ module list
```

Перед запуском Т-программы на выполнение необходимо загрузить модуль launcher/mic. Для этого выполним команду

```
$ module load launcher/mic
```

Запустим Т-программу на двух узлах кластера в пакетном режиме. Для этого используем утилиту sbatch и скрипт symmetric_run_tcp.sh:

```
$ sbatch -N 2 -p quick symmetric_run_tcp.sh --ppn=1 --ppnmic=0 $HOME/t-simple_mult_sum_test
```

В команде используются следующие опции:

-N <количество узлов> – количество запрашиваемых вычислительных узлов;

-p [comm | quick] – очередь (comm – коммерческая, quick – тестовая);

--ppn=<количество MPI-процессов на каждом из узлов>;

--ppnmic=<количество MPI-процессов на каждом из ускорителей>.

Будут запущены два MPI-процесса: по одному на каждом узле.

Просмотреть очередь задач можно командой

```
$ squeue
```

Теперь запустим Т-программу на двух сопроцессорах Intel Xeon Phi. Для этого опять используем утилиту sbatch и скрипт symmetric_run_tcp.sh:

```
$ sbatch -N 2 -p quick symmetric_run_tcp.sh --ppn=0 --ppnmic=1 $HOME/t-simple_mult_sum_test
```

Будут запущены два MPI-процесса: по одному на каждом из ускорителей Intel Xeon Phi. Время выполнения Т-программы t-simple_mult_sum_test на двух хост-узлах и время выполнения на двух ускорителях приведено в таблице.

Возникает вопрос: можно ли как-то оптимизировать код и повысить эффективность и, как следствие, производительность вычислений. Попробуем поднять оптимизацию. Для этого соберем Т-программу с ключом «-O3».

Сборка Т-программы для выполнения на хост-узле выполняется командой

```
$ t++ -opt -O3 t-simple_mult_sum_test.tcc -o t-simple_mult_sum_test
```

Сборка Т-программы для сопроцессора Intel Xeon Phi – командой

```
$ t++ -opt -O3 -mmic t-simple_mult_sum_test.tcc -o t-simple_mult_sum_test.mic
```

Время выполнения Т-программы t-simple_mult_sum_test, собранной с ключом «-O3», на двух хост-узлах и время выполнения на двух ускорителях Intel Xeon Phi приведены в таблице.

Из полученных результатов можно сделать вывод: применение ключа оптимизации «-O3» сократило время выполнения Т-программы почти в два раза.

Рассмотрим следующую Т-программу. По функциональности она выполняет ту же работу, что и первая Т-программа, рассмотренная выше. Только теперь Т-программа будет использовать возможности математической библиотеки Intel MKL. Нам важно увидеть, насколько применение библиотеки Intel MKL сможет повысить эффективность выполнения Т-программы. Теперь Т-программа перемножения матриц будет использовать функцию cblas_dgemm(), а для вычисления суммы абсолютных значений всех элементов результирующих матриц – функцию cblas_dasum() из библиотеки подпрограмм для линейной алгебры BLAS (Basic Linear Algebra Subprograms) [12]. Для выделения памяти под входные и результирующие матрицы во вспомогательных Т-функциях будем использовать функцию mkl_malloc() из библиотеки Intel MKL. Она позволяет выделять память под объекты, выровненную по границе 64 байта, для повышения производительности вычислений.

Соберем нашу Т-программу для выполнения на хост-узлах и для выполнения на ускорителях Intel Xeon Phi. Для подключения математической библиотеки Intel MKL используем ключ «-mkl».

Сборка Т-программы для хостового узла выполняется следующим образом:

```
$ t++ -opt -mkl t-mkl_test.tcc -o t-mkl_test
```

Сборка Т-программы для сопроцессора Intel Xeon Phi:

```
$ t++ -opt -mmic -mkl t-mkl_test.tcc -o t-mkl_test.mic
```

Сравнительная таблица времени выполнения Т-программ

Comparison table T-programs execution time

Т-программа	Время выполнения Т-программы, сек.	
	На двух хост-узлах	На двух ускорителях Intel Xeon Phi
Первый набор		
t-simple_mult_sum_test, собранной с ключом «-O2»	4,02	327,9
t-simple_mult_sum_test, собранной с ключом «-O3»	2,29	176,5
t-mkl_test, собранной с ключом «-mkl», использующей возможности библиотеки Intel MKL	0,68	1,66
Второй набор		
t-simple_dot_product_test	3,84	7,02
t-mkl_dot_product_test, собранной с ключом «-mkl», использующей возможности библиотеки Intel MKL	3,15	6,39

Перед запуском Т-программы предварительно установим переменную окружения LD_LIBRARY_PATH:

$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$MKLROOT/lib/mic:$MKLROOT/../compiler/lib/mic

В переменную окружения LD_LIBRARY_PATH добавлены пути для поиска динамических библиотек: libmkl_core.so, libmkl_intel_thread.so, libmkl_intel_lp64.so, libiomp5.so.

Время выполнения Т-программы t-mkl_test, использующей возможности математической библиотеки Intel MKL, на двух хост-узлах и время выполнения на двух ускорителях приведены в таблице.

Второй набор Т-программ. Рассмотрим следующий пример. Т-программа состоит из трех Т-функций: главной Т-функции tfun int main() и двух вспомогательных – t1() и t2(). Каждая вспомогательная Т-функция вычисляет скалярное умножение двух векторов размерности 134 217 728 чисел типа double, и результат возвращается в качестве выходного значения. При этом для умножения векторов используется обычный цикл. Результат возвращается в главную Т-функцию. Главная Т-функция запускает и ожидает завершения выполнения Т-функций и печатает сумму двух вычисленных сумм. Данная Т-программа не использует возможности математической библиотеки Intel MKL.

Сборка Т-программы для хост-узла выполняется командой

$ t++ -opt t-simple_dot_product_test.tcc -o t-simple_dot_product_test

Сборка Т-программы для сопроцессора Intel Xeon Phi – командой

$ t++ -opt -mmic t-simple_dot_product_test.tcc -o t-simple_dot_product_test.mic

Время выполнения Т-программы на двух хост-узлах и время выполнения на двух ускорителях приведены в таблице.

Следующий пример. Т-программа будет выполнять те же вычисления, что и Т-программа в предыдущем примере, но использовать возможности математической библиотеки Intel MKL. Теперь Т-программа для скалярного умножения двух векторов будет использовать функцию cblas_ddot() из библиотеки подпрограмм для линейной алгебры BLAS [12]. Для выделения памяти под входные векторы во вспомогательных Т-функциях будем использовать функцию mkl_malloc() из библиотеки Intel MKL. Она позволяет выделять память под объекты, выровненную по границе 64 байта, для повышения производительности вычислений.

Сборка Т-программы для хост-узла с ключом «-mkl» выполняется следующим образом:

$ t++ -opt -mkl t-mkl_dot_product_test.tcc -o t-mkl_dot_product_test

Сборка Т-программы для сопроцессора Intel Xeon Phi с ключом «-mkl»:

$ t++ -opt -mmic -mkl t-mkl_dot_product_test.tcc -o t-mkl_dot_product_test.mic

Время выполнения Т-программы, использующей возможности библиотеки Intel MKL, на двух хост-узлах и время выполнения на двух ускорителях представлены в таблице.

Из данных таблицы можно сделать вывод о том, что использование возможностей математической библиотеки Intel MKL позволяет без особых усилий увеличить быстродействие и эффективность счета параллельных Т-программ.

Литература

1. Абрамов С.М., Васенин В.А., Мамчиц Е.Е., Роганов В.А., Слепухин А.Ф. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Науч. сессия МИФИ-2001: сб. науч. тр. Т. 2. М., 2001. С. 34–235.

2. Абрамов С.М., Кузнецов А.А., Роганов В.А. Кросс-платформенная версия Т-системы с открытой архитектурой // Параллельные вычислительные технологии (ПаВТ'2007): тр. Междунар. науч. конф. (29 января–2 февраля 2007 г., Челябинск). Челябинск: Изд-во ЮУрГУ. Т. 1. С. 115–121.

3. Абрамов С.М., Кузнецов А.А., Роганов В.А. Кросс-платформенная версия Т-системы с открытой архитектурой // Вычислительные методы и программирование. 2007. Т. 8. № 1. Разд. 2. С. 175–180; URL: http://num-meth.srcc.msu.su/ (дата обращения: 04.09.2014).

4. Кузнецов А.А., Роганов В.А. Экспериментальная реализация отказоустойчивой версии системы OPENTS для платформы WINDOWS CCS // Суперкомпьютерные системы и их применение (SSA'2008): тр. 2-й Междунар. науч. конф. (27–29

октября 2008 г., Минск). Минск: Изд-во ОИПИ НАН Беларуси, 2008. С. 65–70.

5. OpenTS. Руководство программиста. URL: http://www.opents.net/index.php/ru/ruk-progr (дата обращения: 04.09.2014).

6. Степанов Е.А. Планирование в OpenTS – системе автоматического динамического распараллеливания // Информационные технологии и программирование: межвуз. сб. стат.; [под ред. В.А. Васенина, Д.Л. Ревизникова, Е.А. Роганова]. М.: Изд-во МГИУ, 2005. Вып. 2 (14). С. 31–42.

7. Кузнецов А.А., Роганов В.А. Поддержка отказоустойчивых хранилищ данных в системе OpenTS // Программные системы: теория и приложения. 2011. № 3 (7); С. 53–60. URL: http://psta.psiras.ru/read/psta2011_3_53-60.pdf (дата обращения: 04.09.2014).

8. Intel Math Kernel Library Documentation. URL: http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/ (дата обращения: 04.09.2014).

9. Использование библиотеки Intel MKL при программировании на сопроцессоре Intel Xeon Phi. URL: http://www.intuit.ru/studies/courses/10612/1096/lecture/22919 (дата обращения: 04.09.2014).

10. Суперкомпьютер «Торнадо ЮурГУ». URL: http://supercomputer.susu.ac.ru/computers/tornado/ (дата обращения: 04.09.2014).

11. Планировщик задач SLURM. URL: https://computing.llnl.gov/linux/slurm/documentation.html (дата обращения: 04.09.2014).

12. Intel Math Kernel Library Reference Manual. URL: https://software.intel.com/sites/products/documentation/doclib/iss/2013/mkl/mklman/index.htm (дата обращения: 04.09.2014).

DOI: 10.15827/0236-235X.109.043-048 Received 29.09.14

USING POSSIBILITIES OF INTEL MKL MATH LIBRARY IN PARALLEL PROGRAMS IN T++ LANGUAGE FOR A T-SYSTEM WITH AN OPEN ARCHITECTURE TO IMPROVE THEIR PERFORMANCE

***Roganov V.A.**, Research Associate, var@pereslavl.ru; **Kuznetsov A.A.**, Research Associate, tonic@pereslavl.ru;*
***Matveev G.A.**, Leading Engineer Researcher, gera@prime.botik.ru;*
***Osipov V.I.**, Ph.D. (Physics and Mathematics), Research Associate, val@pereslavl.ru*
(Program System Institute of RAS, Petr I St. 4a, Pereslavl-Zalesskiy, 152021, Russian Federation)

Abstract. The problem of parallel computing is a problem of software. The most common approach to developing parallel programs is based on using software packages such as MPI (*Message Passing Interface).* This approach requires a large amount of knowledge from developer. Developing and debuging parallel programs is also time consuming. The PSI RAS has developed a T-system which implements automatic dynamic parallelization at runtime. Input language for T-system is T++, and applications developed for the T-system are T-applications or T-programs. This paper provides an overview of products and components of Intel Math Kernel Library (Intel MKL), which contains a large set of mathematical functions and can be used in parallel T-applications to accelerate computation and get maximum performance. The paper considers the modes of using the Intel MKL mathematical library on clusters with Intel Xeon processors and one or more Intel Xeon Phi accelerators (coprocessors) from Intel company. There are demo examples of using the library in C and T++. The paper shows how the use of Intel MKL math library affects the efficiency of parallel T-programs. All experiments has been performed on "RSC Tornado SUSU" energy-efficient supercomputer of the South Ural State University.

Keywords: T++ programming language, OpenTS, T-system, C++parallel extension, supercomputers, Intel MKL math library.

References

1. Abramov S.M., Vasenin V.A., Mamchits E.E., Roganov V.A., Slepukhin A.F. Dynamic program multisequencing based on parallel graph reduction. T-system new version software architecture. *Nauchnaya sessiya MIFI-2001. Vol. 2 Informatika i upravlenie. Informatsionnye tekhnologii. Setevye tekhnologii. Parallelnye setevye tekhnologii* [MEPhi-2001 scientific session. Informatics and Control. Informational technologies. Network technologies. Parallel Computing Technologies]. Collected papers. Moscow, 2001, pp. 234–235 (in Russ.).

2. Abramov S.M., Kuznetsov A.A., Roganov V.A. T-system cross-platform version with open architecture. *Trudy Mezhdunar. nauch. konf. "Parallelnye vychislitelnye tekhnologii (PAVT'2007)"* [Proc. of the Int. Scientific Conf. "Parallel Computational Technologies PCT'2007"]. Vol. 1, Chelyabinsk, 2007, pp. 115–121 (in Russ.).

3. Abramov S.M., Kuznetsov A.A., Roganov V.A. T-system cross-platform version with open architecture. *Vychislitelnye metody i programmirovanie* [Computational methods and programming]. 2007, vol. 8, no. 1, section 2, pp. 175–180, available at: http://num-meth.srcc.msu.su/ (accessed September 04, 2014).

4. Kuznetsov A.A., Roganov V.A. An experimental implementation of OPENTS system fault-tolerant version for WINDOWS CCS. *Trudy 2 Mezhdunar. nauch. konf. "Superkomputernye sistemy i ikh primenenie (SSA'2008)"* [Proc. of the 2nd Int. Scientific Conf. "Supercomputer Systems and Applications (SSA'2008)"]. Minsk, OIPI NAN Belarusi Publ., 2008, pp. 65–70 (in Russ.).

5. *OpenTS. Progarmmer's Guide*. Available at: http://www.opents.net/index.php/ru/ruk-progr (accessed September 04, 2014).

6. Stepanov E.A. Scheduling in OpenTS – system with automatic dynamic parallelization. *Informatsionnye tekhnologii i programmirovanie: mezhvuzovskiy sbornik statey* [Informational technologies and Programming: interuniversity collection of articles]. Pod redaksiey V.A. Vasenin, D.L. Reviznikov, E.A. Roganov (Eds.). Moscow, MGIU Publ., 2005, iss. 2 (14). pp. 31–42 (in Russ.).

7. Kuznetsov A.A., Roganov V.A. Supporting fault-tolerant data storage in OpenTS system. *Elektronny Nauchny Zhurnal: "Programmnye systemy: teoriya i prilozheniya"* [Electronic scientific journ.: "Program systems: theory and applications"]. 2011, no. 3 (7), pp. 53–60. Available at: http://psta.psiras.ru/read/psta2011_3_53-60.pdf (accessed September 04, 2014).

8. *Intel Math Kernel Library Documentation*. Available at: http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/ (accessed September 04, 2014).

9. *Ispolzovanie biblioteki Intel MKL pri programmirovanii na soprotsessore Intel Xeon Phi* [The use of Intel MKL library when programming in Intel Xeon Phi accelerator]. Available at: http://www.intuit.ru/studies/courses/10612/1096/lecture/22919 (accessed September 04, 2014).

10. *Supercomputer "Tornado YuUrGU"* [Supercomputer "YuUrGU Tornado"]. Available at: http://supercomputer.susu.ac.ru/computers/tornado/ (accessed September 04, 2014).

11. *Planirovshchik zadach SLURM* [SLURM Task Scheduler]. Available at: https://computing.llnl.gov/linux/slurm/documentation.html (accessed September 04, 2014).

12. *Intel Math Kernel Library Reference Manual*. Available at: https://software.intel.com/sites/products/documentation/doclib/iss/2013/mkl/mklman/index.htm (accessed September 04, 2014).