КОМПЬЮТЕРНЫЕ НАУКИ

УДК 519.6

РАЗРАБОТКА ЭФФЕКТИВНЫХ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ

Д.Ю. Тюгин, А.А. Куркин

Нижегородский государственный технический университет им. Р.Е. Алексеева
603950, Нижний Новгород, ул. Минина, 24

Аннотация: В статье рассматривается комплексный подход к задаче повышения производительности вычислений с применением параллельных технологий. Показано применение современных инструментов для распараллеливания, поиска ошибок и «узких» мест в алгоритмах на примере задачи моделирования внутренних гравитационных волн в замкнутых бассейнах. Разработан и реализован параллельный алгоритм расчета коэффициента захвата внутренних волн. Проведена оценка его эффективности.
Ключевые слова: параллельные вычисления, численные методы, внутренние волны в океане.

При исследовании таких крупномасштабных явлений как генерация и распространение внутренних волн в океане большую роль играет численное моделирование. Использование данных натурных измерений и математических моделей в совокупности с мощными вычислительными системами позволило исследовать подобные гидрофизические процессы. Тем не менее, масштабность таких процессов требует высокопроизводительных вычислений. Особенно это важно при реализации программно-аналитических комплексов для получения экспресс-оценок тех или иных параметров исследуемых процессов. Для подобных оценок необходимо иметь возможность быстро получить результаты моделирования по заданным параметрам. Одним из методов решения подобных задач является применение параллельных технологий.

Авторами был разработан многоцелевой программный комплекс IGWResearch [1 – 3] для исследования распространения и трансформации внутренних гравитционных волн, позволяющий также рассчитывать локализацию энергии данного типа волн в замкнутых бассейнах. Такие расчеты позволяют делать экспресс-оценки акваторий и выделять безопасные, с точки зрения волнового воздействия, области. Тем не менее, производительность комплекса была недостаточной и время вычислений на сетках данных с высоким разрешением доходило до нескольких недель.

Современные средства автоматического распараллеливания ещё не вышли на достаточный уровень, при котором не требуется участие человека в разработке параллельного алгоритма. С другой стороны уже существуют средства поддержки цикла распараллеливания исходного кода, например, Intel Parallel Studio XE 2013. Данный набор программ позволяет провести полный анализ от поиска места в коде для распараллеливания до анализа полученной производительности, что существенно сокращает время и затраты на разработку.

Математическая модель. Рассмотрим одну из математических моделей лежащую в основе программного комплекса [1]. Рефракционная модель позволяет описать распространения волн в слабо неоднородном океане в рамках лучевой теории [4 – 7]. Уравнения, описывающие распространение монохроматической волны с произвольной частотой, имеют вид [7 – 9]:

$$\frac{d\vec{R}}{ds} = \frac{c^2}{\omega}\vec{k}, \qquad \frac{d\vec{k}}{ds} = -(\nabla c)|\vec{k}|, \tag{1}$$

где $\vec{R} = \{x,y\}$ координаты радиус вектора точки на луче; $c = c(x, y)$ – фазовая скорость внутренних волн; $\omega = ck$ – частота длинных волн; $\vec{k} = \{k_x, k_y\}$ – волновой вектор, $k = \sqrt{k_x^2 + k_y^2}$ – модуль волнового вектора; s – время на луче; $\nabla = \left\{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right\}$ – дифференциальный оператор.

Решение системы дифференциальных уравнений (1) выполнено методом Рунге-Кутта 5-го порядка с модификацией Мерсона. Внешние условия задаются матрицей распределения фазовой скорости, рассчитанной в рамках слабонелинейной теории на основе международных атласов гидрологических параметров WOA [10], GDEM [11] и модельных данных RCO [12]. Результатом расчета являются траектории лучей.

На основе данных траекторий выполняется построение карты распределения энергии внутренних гравитационных волн. Для этого вводится понятие локального коэффициента захвата. Из каждого узла сетки данных фазовой скорости выпускается 200 лучей (дальнейшее увеличение числа лучей не вносит значительного влияния на конечный результат), при достижении лучами областей с малыми значениями фазовой скорости расчет прекращается. По координатам конечных точек на лучах определяется ближайший узел сетки. Число конечных точек в каждом узле сетки суммируется в результирующую матрицу. Коэффициент захвата определяется формулой (2)

$$K_{ij} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij} n_{ij}}{R \cdot N} \cdot 100\%, \tag{2}$$

где K_{ij} – величина коэффициента захвата для точки (i, j) исходной сетки, C_{ij} – значение фазовой скорости в точке, n_{ij} – число конечных точек лучей в ячейке сетки точки (i, j), R – общее число лучей, N – общее число точек сетки

Последовательный алгоритм. Последовательный алгоритм расчета коэффициента захвата (2) представлен на рис. 1.

Загружаются исходные данные в виде матрицы фазовой скорости с шагом в 1/15 и 1/30 градуса по широте и долготе соответственно. Она представляет собой сетку данных. Для каждой узловой точки этой сетки моделируется распространение волны. Выбирается число лучей для представления волны (в данном случае 200), задаются координаты исходной точки (Refraction2::setPoint(x,y)) и матрица значений фазовой скорости c(x,y), задается направление луча, выполняется решение системы уравнений (1) методом Рунге-Кутта для первого луча (RefRaySolver::init(1)), затем для второго и т.д. Данные траекторий для текущей точки сохраняются в результирующий контейнер.

Проводится шаг алгоритма для следующей точки. После расчета всех точек коэффициент захвата вычисляется на основе полученных траекторий по формуле (2).

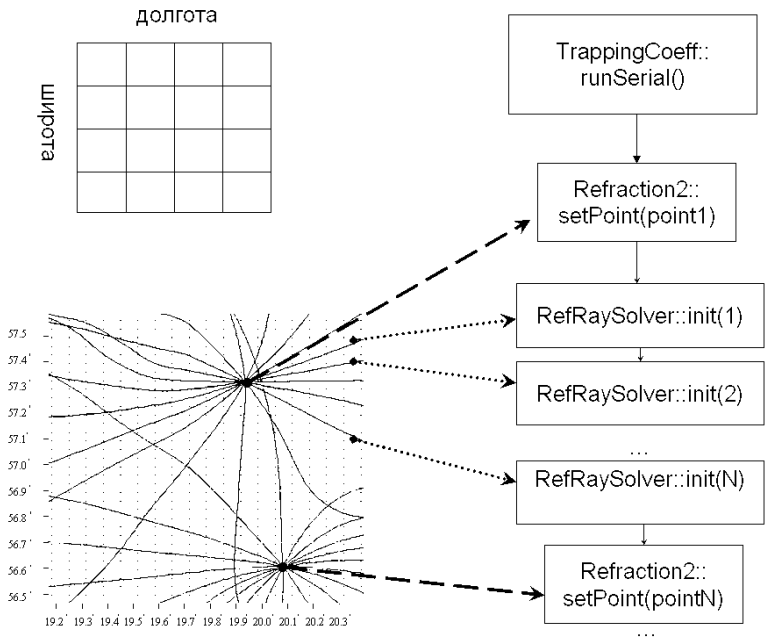

Рис. 1. Схема последовательного расчета коэффициента захвата (2)

Как видно из алгоритма, расчет каждого луча в каждой точке независим и может быть проведен отдельно. Большое число точек и лучей требует длительного времени расчета, и данная схема имеет большой потенциал для распараллеливания. Стоит отметить, что вычисление одного луча не является слишком длительной операцией, поэтому параллелизация по точкам предполагается более эффективной.

Для поддержки процесса параллелизации кода в Intel® Parallel Studio XE 2013 имеются все необходимые инструменты, которые позволяют существенно снизить временные и человеческие ресурсы на поиск мест распараллеливания, оценки эффективности, а также поиск и анализ узких, с точки зрения производительности, мест в коде.

Поиск области кода для параллелизации. Для поиска места распараллеливания использовался программный продукт Intel® Advisor XE. Данный профилировщик позволяет построить распределение времени, затраченного на выполнение отдельных участков кода, а также подсказать, на что стоит обратить внимание при поиске мест параллелизации.

На рис. 2 представлен результат профилировки программного комплекса IGWResearch при выполнении расчета коэффициента захвата.

Древовидное представление позволяет наглядно проанализировать время выполнения различных участков кода. По результатом профилировки Intel® Advisor XE предоставляет подсказку: место кода, где параллелизация даст наиболее эффективный результат. В данном случае это место соответствует функции TrappingCoeff::runSerial(), реализующей расчет волны для каждой географической точки.

Function Call Sites and Loops	Total Time %	Total Time	Self Time	Source Location
⊟Total	100.0%	116.7622s	0s	
⊟_RtlUserThreadStart	100.0%	116.7622s	0s	
⊟__RtlUserThreadStart	100.0%	116.7622s	0s	
⊟BaseThreadInitThunk	100.0%	116.7622s	0s	
⊟_endthreadex	97.8%	114.1726s	0s	threadex.c:292
⊟_endthreadex	97.8%	114.1726s	0s	threadex.c:314
⊟QThread::setPriority	97.8%	114.1726s	0s	
⊟TrappingCoeff::run	97.8%	114.1726s	0s	trappingcoeff.cpp:180
⊟TrappingCoeff::runSerial [loop]	97.8%	114.1417s	0s	trappingcoeff.cpp:119
⊟TrappingCoeff::runSerial	97.4%	113.6890s	0s	trappingcoeff.cpp:127
⊟Refraction2::run	97.4%	113.6890s	0s	refraction2.cpp:177
⊟Refraction2::calcRays [loop]	97.4%	113.6890s	0s	refraction2.cpp:100
⊟Refraction2::calcRays [loop]	97.4%	113.6890s	0s	refraction2.cpp:105
⊟Refraction2::calcRays	96.8%	112.9876s	0s	refraction2.cpp:107
⊟RefRaySolver::makeStep	85.5%	99.7751s	0s	refraysolver.cpp:286
⊟RungeKutt5::process [loop]	84.9%	99.1198s	0s	rungekutt5.cpp:61
⊟RungeKutt5::process	77.2%	90.1676s	0s	rungekutt5.cpp:97
⊞RefRaySolver::f	26.5%	30.8879s	0s	refraysolver.cpp:176
⊞RefRaySolver::f	26.3%	30.7309s	0s	refraysolver.cpp:177
⊞RefRaySolver::f	17.2%	20.1230s	0s	refraysolver.cpp:170

Рис. 2. Результаты профилировки программой Intel® Advisor XE

Параллельный алгоритм. На основе области кода для параллелизации была составлена схема параллельного алгоритма. Данная схема предполагает, что расчеты точек будут проводиться параллельно, (рис. 3).

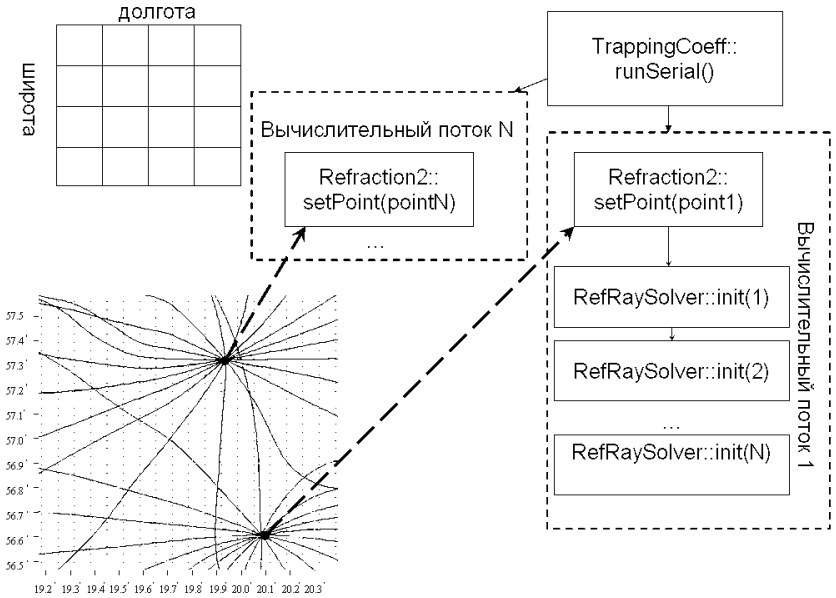

Рис. 3. Схема параллельного алгоритма

Следующий этап – вставка псевдоинструкций (аннотаций) параллелизации (см. приложение), этот этап позволил посмотреть, как поведет себя программа с точки зрения производительности если её распараллелить в обозначенных местах. Таким образом, не затрачивая время на значительную модификацию кода, можно оценить

какой прирост производительности будет получен, и при необходимости изменить место распараллеливания.

Для данной оценки Intel® Advisor XE имеет модель предсказания. При профилировке разработанного алгоритма она показала ускорение в восемь раз на восьми ядрах, что является фактически близким к идеальному, рис. 4

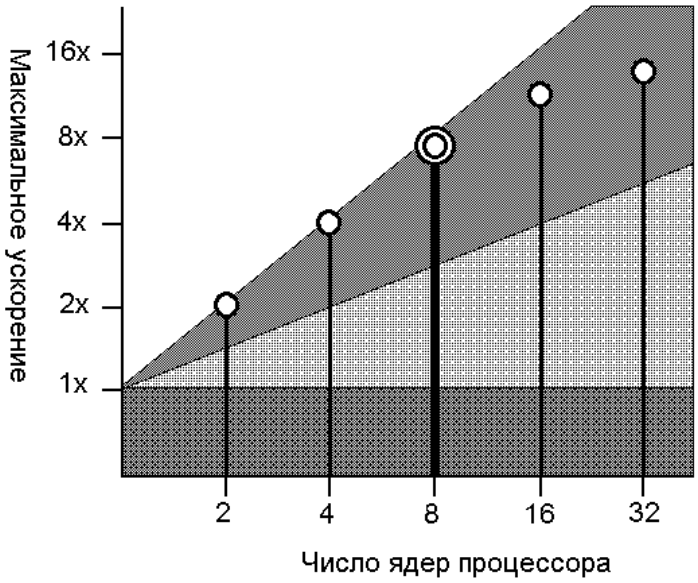

Рис. 4. Модель предсказания эффективности параллельного алгоритма

Также модель предсказания позволяет ответить на вопрос: что будет если использовать разные библиотеки для параллелизации (Intel® TBB, Intel® Cilk, Intel® OpenMP) или различное число ядер. Необходимо учитывать, что результаты модели носят приближенный характер, тем не менее, ошибочную схему можно сразу обнаружить на данном этапе.

Поиск ошибок памяти и потоков. После внесения параллельных мест, в программе могут появиться неочевидные ошибки, такие как гонки данных. Такие ошибки сложно обнаружить без соответствующих инструментов, и следствием таких ошибок могут стать неверные результаты расчетов. Для поиска ошибок в Intel® Advisor XE есть тип анализа проверка на корректность.

Проверка показала наличие ошибок при сборе данных. В данном случае при сохранении результатов траектории каждого луча происходит подсчет количества конечных точек лучей, общее количество сохраняется в матрице предварительных результатов, на основе которых происходит построение коэффициента захвата. Для устранения ошибки был введен объект мьютекс для синхронизации потоков (см. приложение). На данном этапе он также может быть создан с помощью аннотаций. Для реализации параллельной версии (см. приложение) был выбран набор библиотек Intel® Threading Building Block (TBB [13]). Данный инструмент также входит в Intel® Parallel Studio XE 2013.

Для проверки правильности работы параллельных алгоритмов и поиска возможных ошибок был применен продукт Intel® Inspector XE. Проверка на корректность Intel®

Advisor XE работает только с аннотациями, таким образом, после перевода исходного кода с псевдоинструкций на использование конкретной библиотеки для распараллеливания, требуется дополнительная проверка. Можно сразу воспользоваться Intel® Inspector XE, но с другой стороны Intel® Advisor XE позволяет проверить корректность ещё на этапе проектирования параллельного кода.

Помимо ошибок работы с потоками Intel® Inspector XE позволяет также обнаружить ошибки работы с памятью. Например, утечки памяти или выход за границы массива и т.д. В случае, когда расчет выполняется очень долго, утечка памяти может проявить себя спустя значительное время, в системе закончится свободная память и расчет не сможет быть продолжен. Если рассматривать большие задачи, когда вычисления могут длиться несколько дней, время фактически будет потеряно, в случае с арендой кластера это уже не только временные, но и финансовые потери.

На рис. 5 показан результат профилировки программой Intel® Inspector XE (анализ памяти).

Problems

ID ▲	Problem	Sources	Modules	Object Size	State
⊞P3	Memory leak	tools.cpp	IGWResearch.exe	34832	New
⊞P4	Memory leak	tools.cpp	IGWResearch.exe	3117464	New
⊞P5	Memory leak	tools.cpp	IGWResearch.exe	34832	New
⊞P6	Memory leak	tools.cpp	IGWResearch.exe	24939712	New
⊞P11	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P12	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P13	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P14	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P15	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P16	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P17	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P18	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P19	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P20	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P21	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P22	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P23	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P24	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P25	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New
⊞P26	Memory leak	rungekutt5.cpp	IGWResearch.exe	320	New

Filters Sort

Severity	
Error	20 item(s)
Problem	
Memory leak	20 item(s)
Source	
rungekutt5.cpp	16 item(s)
tools.cpp	4 item(s)
Module	
IGWResearch.exe	20 item(s)
State	
New	20 item(s)
Suppressed	
Not suppressed	20 item(s)
Investigated	
Not investigated	20 item(s)

Рис. 5. Результаты анализа на ошибки памяти

Всего было обнаружено 20 диагностик с типом утечка памяти. Для того чтобы не разбирать ошибки в прикладных модулях (в данном случае часть ошибок была обнаружена в библиотеке Qt), можно отфильтровать список по названию модуля программы. Все ошибки были исправлены, повторный запуск анализа подтвердил их отсутствие.

Также была найдена гонка данных связанная с отрисовкой результатов и их расчетом. В данном случае ошибка некритична, так как визуализация результатов в процессе расчета носит ознакомительный характер, вследствие того, что коэффициент захвата считается только целиком. Иными словами, точная карта коэффициента захвата может быть получена только после окончания расчета всех точек и нормализации матрицы результатов.

Анализ достигнутой производительности. Для анализа результатов были проведены замеры времени выполнения параллельного и последовательного алгоритмов. Замеры проводились на двухпроцессорной машине с процессорами Intel® Xeon® 5570 2.93 ГГц, 16 Гб оперативной памяти, ОС Windows 7.

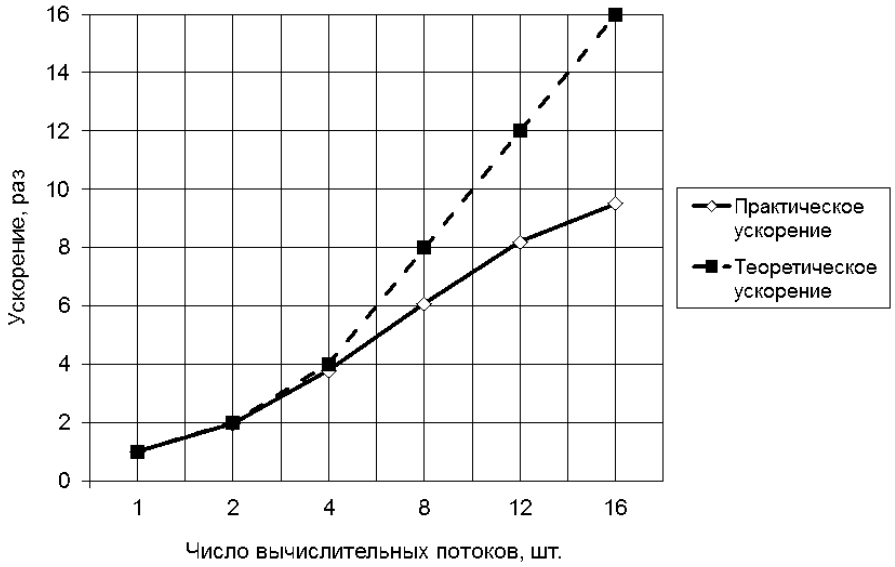

Рис. 6. Зависимость ускорения от числа потоков

На основании замеров был построен график ускорения (отношения времени выполнения последовательного алгоритма к времени выполнения параллельного), (рис.6).

Как видно из рис. 6, реальное ускорение становится меньше предсказанного (в идеальном случае – ускорение прямо пропорционально числу потоков). Причиной тому могут быть дополнительные синхронизации, которые добавляются в параллельную версию, накладные расходы библиотек Intel® TBB и т.п.

Замеры времени работы параллельной и последовательной версий могут показать ускорение, но в случае если оно не совпадает с ожидаемым, без соответствующих инструментов профилировки разобраться практически невозможно. Для того чтобы оценить результаты, распределение нагрузки по потокам и время их синхронизации, а также дальнейшие пути оптимизации служит инструмент Intel® VTune™ Amplifier XE.

Как видно на рис. 7, большую часть времени на 16 ядерной машине работало одновременно 16 потоков.

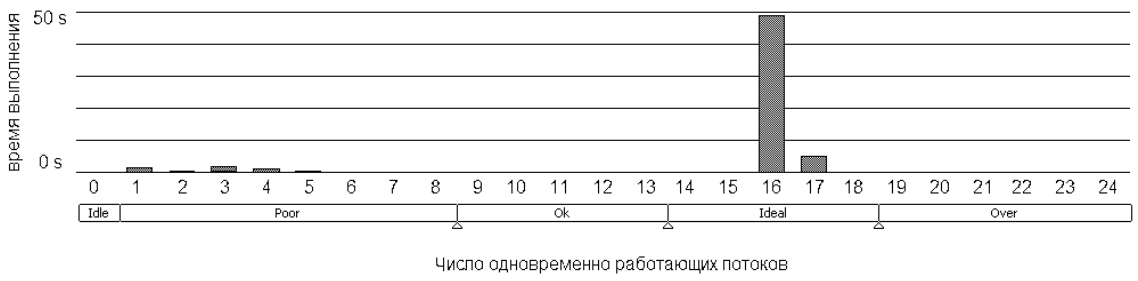

Рис. 7. Распределение по времени числа одновременно работающих потоков

Это хороший результат, показывающий, что почти всегда все потоки заняты работой. Для проверки времени синхронизации была запущена профилировка по анализу синхронизации между потоками, рис. 8.

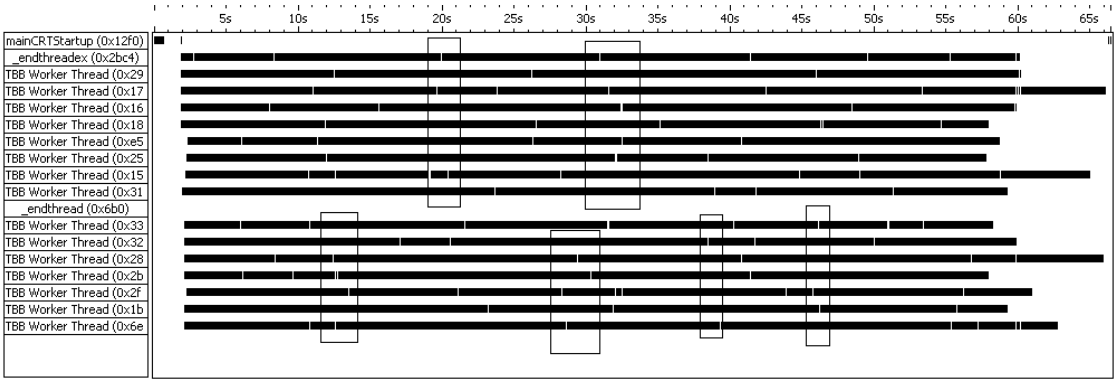

Рис. 8. Анализ времени выполнения и синхронизаций потоков

Как видно из диаграммы, синхронизации встречаются часто, но каждый раз задействованы не все потоки, и время синхронизации минимально. Зато время работы потоков оказалось различным. Таким образом, в самом конце работы часть потоков может закончить расчет раньше других. Такое ограничение вызвано неоднородностью задачи. Расчет каждой волны зависит от распределения фазовой скорости, которое в разных точках может быть различно.

Данный инструмент также полезен при более детальной и низкоуровневой оптимизации кода.

Применение Intel® Parallel Studio XE 2013 показало высокую эффективность при разработке параллельных алгоритмов. Инструменты Intel® Parallel Studio XE 2013 позволяют существенно облегчить и ускорить процесс параллелизации кода. Моделирование параллельной версии алгоритма без значительной модификации кода позволяет оценить ускорение на этапе проектирования параллельного алгоритма. Анализ ошибок позволяет найти неочевидные ошибки и предотвратить получение неверных результатов или нестабильной работы приложения. Без инструментов анализа производительности практически невозможно определить степень корректности параллельной реализации с точки зрения производительности.

Заключение

Разработанная реализация параллельного алгоритма позволила сократить время расчетов в 10 раз на 16 ядерной машине, что позволит в дальнейшем проводить расчеты на более точных сетках с большим числом данных.

Представленные результаты поисковой научно-исследовательской работы получены в рамках реализации:

1. Гранта «Применение Intel® Parallel Studio 2013 XE Beta в разработке эффективных параллельных алгоритмов ДОГОВОР № NN/R&D/56/2012 от 30/03/2012»
2. ФЦП «Научные и научно-педагогические кадры инновационной России на 2009 - 2013 годы»

Приложение

Вставки псевдоинструкции (аннотации) параллелизации

```
void TrappingCoeff::runSerial()
{       ...
  //Главный цикл по точкам, trgetSet – массив точек
ANNOTATE_SITE_BEGIN( start_sources );
  for(int i=0;i<targetSet.size();i++)
  {
    ANNOTATE_TASK_BEGIN( wave_propagation );
    double lon=targetSet.at(i).x;          //географическая широта
    double lat=targetSet.at(i).y;          //географическая долгота
    this->startPoint(lon,lat);             //визуализация начальной точки

    Refraction2 ref;                       //Создание объекта для решения системы (1)

    this->initRefObject(ref,lon,lat);      //задание начальных параметров
    ref.run();                             //Расчет
              ...
    this->pointEnd(lon,lat);               //визуализация конечной точки
    ANNOTATE_TASK_END();
    if(exit)
    {
      break;
    }
  }
ANNOTATE_SITE_END();
}
```

Вставка объекта мьютекс в исходный код

```
void Refraction2::collectEndPoints(double lon, double lat)
{
  if(endPointMap!=NULL)
  {
    ANNOTATE_LOCK_ACQUIRE(endPointMap);
    double v=endPointMap->getNearest(lon,lat);
    v=v+1;
    endPointMap->setNearest(lon,lat,v,true);
    ANNOTATE_LOCK_RELEASE( endPointMap);
  }
}
```

Реализация параллельной версии алгоритма

```
task_scheduler_init init;
parallel_for(blocked_range<int>(0,targetSet.size()),
ParallelTrappingCoeff(this,targetSet),auto_partitioner());

void ParallelTrappingCoeff::operator()(const blocked_range<int> & r)const
{
   Refraction2 ref;
   for( int i=r.begin(); i!=r.end(); ++i )//часть главного цикла
   {
      double lon=targetSet.at(i).x;
      double lat=targetSet.at(i).y;
      tc->pointStart(lon,lat);
      tc->initRefObject(ref,lon,lat);    //задание начальных параметров
      ref.run();                         //расчет распространения волны
           ...
      tc->pointEnd(lon,lat);
   }
}

//Сбор траекторий
void Refraction2::collectEndPoints(double lon, double lat)
{
   if(endPointMap!=NULL)
   {
        collectMutex.lock();
      double v=endPointMap->getNearest(lon,lat);
      v=v+1;
      endPointMap->setNearest(lon,lat,v,true);
        collectMutex.unlock();
   }
}
```

ЛИТЕРАТУРА

1. *Тюгин Д.Ю., Куркина О.Е., Куркин А.А.* Программный комплекс для численного моделирования внутренних гравитационных волн в мировом океане. // Фундаментальная и прикладная гидрофизика. 2011. № 4(2). С. 32 – 44.
2. *Тюгин Д.Ю., Куркина О.Е., Куркин А.А., Гиниятуллин А.Р.* Программа для ЭВМ «Моделирование внутренних гравитационных волн в рамках рефракционной модели», Россия, А.С. 2010615884 // Роспатент, 2010.
3. *Тюгин Д.Ю., Куркина О.Е., Куркин А.А., Гиниятуллин А.Р.* Программа для ЭВМ «Программный комплекс для исследования внутренних гравитационных волн», Россия, А.С. 2011612327 // Роспатент, 2011.

4. *Бреховских Л.М., Годин О.А.* Акустика слоистых сред. – М.: Наука, 1989. 416 с..
5. *Воронович А.Г.* Распространение внутренних и поверхностных гравитационных волн в приближении геометрической оптики // Известия АН СССР. ФАО. 1976. Т. 12. С. 519–523.
6. *Кравцов Ю.А., Орлов Ю.И.* Геометрическая оптика неоднородных сред. – М.: Наука, 1980. 303 с.
7. *Пелиновский Е.Н.* Нелинейная динамика волн цунами. – Горький: ИПФ АН СССР, 1982.
8. *Ле Блон П., Майсек Л.* Волны в океане. Т. 1, 2. – М.: Мир, 1981.
9. *Broutman D., Rottman J.W., Eckermann D.* Ray methods for internal waves in the atmosphere and ocean // Annual Review of Fluid Mech. 2004. V. 36. P. 233–253.
10. *T.P. Boyer, J.I. Antonov, H.E. Garcia, D.R. Johnson, R.A. Locarnini, A.V. Mishonov, M.T. Pitcher, O.K. Baranova, I.V. Smolyar*, 2006. World Ocean Database 2005. S. Levitus, Ed., NOAA Atlas NESDIS 60, U.S. Government Printing Office, Washington, D.C., 190 pp., DVDs.
11. *Teague W.J., Carron M.J., Hogan P.J.* A Comparison between the Generalized Digital Environmental Model and Levitus Climatologies // J. Geophys. Res. 1990. V. 95. C5. P. 7167 – 7183.
12. *Soomere T., Delpeche T. et. al.* Patterns of current-inducted transport in the surface layer of the Gulf of Finland // Boreal Environment Research. 2001. V. 16. P. 49 – 63.
13. Intel(R) Threading Building Blocks Reference Manual // URL: http://threadingbuildingblocks.org/uploads/81/91/Latest%20Open%20Source%20Documentation/Reference.pdf (дата обращения 01.07.2012)

DEVELOPMENT OF EFFECTIVE PARALLEL ALGORITHMS FOR SOLVING OF PROBLEMS OF NUMERICAL MODELING

D. Tyugin, A. Kurkin

Alexeev Nizhny Novgorod State Technical University
24, Minin Street, Nizhny Novgorod, 603950, Russia

Abstract: In this paper complex approach for solving performance problem in computing based on parallel technologies is considered. Demonstrated applying modern tools for parallelization, errors and “bottlenecks” catching in the problem of modeling internal gravity waves in closed basins. Parallel algorithm of trapping coefficient computing is designed and implemented. The algorithm performance is estimated.
Key words: parallel computing, numerical methods, internal waves in the ocean.