ПРИМЕНЕНИЕ МНОГОПОТОЧНОСТИ В ГРАФИЧЕСКОЙ СИСТЕМЕ ISOGRAPH

Ю.В. Алферов, В.В. Копейкин

Гидрометеорологический научно-исследовательский центр Российской Федерации
alferov@mecom.ru, v.v.kopeykin@mail.ru

В последние годы практически все новые персональные компьютеры оснащаются многоядерными процессорами. Нередки случаи, когда в процессоре персонального компьютера 4, 6, а теперь даже можно встретить 12 вычислительных ядер. Это говорит о том, что на компьютере одновременно, не мешая друг другу, могут выполняться несколько процессов или потоков команд (в англоязычной литературе — threads).

Часто вычисления в сложных и долго исполняемых задачах распараллеливают таким образом, чтобы различные расчеты выполнялись бы несколькими потоками команд одновременно, обмениваясь при этом, если это нужно, результатами своих вычислений. Тем самым достигается ускорение решения задачи во много раз. Выполнение подобных расчетов требует особой техники программирования с использованием специальных библиотек процедур (например, **MPI — Message Passing Interface) или директив компилятора (например, OpenMP —** Open Multi-Processing), облегчающих организацию параллельных процессов или потоков и коммуникации между ними. Даже в этом случае в настоящее время, как правило, необходим индивидуальный подход к каждой задаче. Использование же систем автоматического распараллеливания, встроенных в компилятор и не требующих «ручного» управления, чаще всего не приводит к удовлетворительным результатам.

Упомянутые средства для упрощения организации многопоточных приложений предназначены для решения вычислительных задач на языках программирования Фортран или Си/Си++. Если же программа написана на другом алгоритмическом языке и касается области компьютерной графики, то не остается другого пути организации потоков, кроме «ручного» способа, путем вызова соответствующих программ операционной системы для инициализации и завершения потоков, их синхронизации и установления взаимодействия между ними.

В течение нескольких лет используется и совершенствуется разработанная в Гидрометцентре России графическая программная система Isograph [1, 2] для визуализации карт метеорологических данных. Для этой системы актуальна проблема использования возможностей современной компьютерной архитектуры для ускорения решения стоящих перед системой задач.

Наиболее вычислительно затратной операцией при визуализации данных является расчет карты, особенно если объем данных значителен, как, например, визуализация прогностического поля какого-либо метеоэлемента модели атмосферы COSMO (около полумиллиона точек на одном уровне) [4]. Это связано, например, с расчетом координат точек на изолиниях, двойным преобразованием координат (сначала из координат проекции поля модели COSMO в географические, а затем в координаты проекции карты [1]). К сожалению, каждый вид данных и каждый вид их изображения диктует свою процедуру расчета. Как правило, расчет выполняется достаточно быстро, за приемлемое для интерактивной системы время, и тогда попытка распараллелить такие вычисления может привести к неоправданному усложнению программы, влекущему появление дополнительных трудно отлаживаемых ошибок и чрезмерное усложнение процедуры сопровождения программы. Кроме того, распараллеленная программа при недостаточном объеме вычислений (то есть «полезной нагрузки») может привести и к замедлению по сравнению с однопоточной программой, что связано с «накладными расходами» на организацию потоков, коммуникации и синхронизацию.

Для равномерного распределения вычислительной нагрузки по ядрам процессора можно было бы также распараллелить расчеты изображений для различных слоев карты, определяемых разными данными, например рассчитывать в разных потоках береговую линию, реки, изолинии поля метеоэлемента, наноску пуансонов метеонаблюдений для одной карты. Однако время расчетов для столь разнородных слоев существенно различается, что приводит к значительному дисбалансу нагрузки на разные потоки. Балансировка загрузки процессоров для столь разнородных, а потому и разной продолжительности потоков может оказаться весьма непростой.

Одной из особенностей системы Isograph является то, что каждая карта открывается в отдельном окне, а количество таких карт-окон не ограничено. Многооконный режим работы с Isograph наиболее востребован. Причем, как правило, совокупность карт включает однотипные изображения, различающиеся либо данными за разные сроки, либо полями прогнозов разных заблаговременностей. Запрос на пересчет всех таких карт часто возникает одновременно, например при изменении размеров окон, связанных с выстраиванием карт рядами («черепицей»). Таким образом, включение всех расчетов для одной карты в один поток может позволить сэкономить время для расчета такого комплексного изображения, состоящего из нескольких карт. К тому же вычисления в таких потоках сами собой оказываются достаточно сбалансированными в силу природы используемых данных. Это и решено было сделать на первых порах. Схема структуры потоков Isograph и их взаимодействия (пример некоторого состояния) приведена на рис. 1.

Программная система Isograph реализована на языке программирования Delphi Object Pascal. Стандартный набор графических средств этого языка разрабатывался давно, и эти средства таковы, что не могут корректно работать в условиях нескольких взаимодействующих потоков [5]. Видимо, это связано с наличием некоторого количества управляющих глобальных переменных, за которыми при работе нескольких потоков возникают гонки с непредсказуемыми результатами.

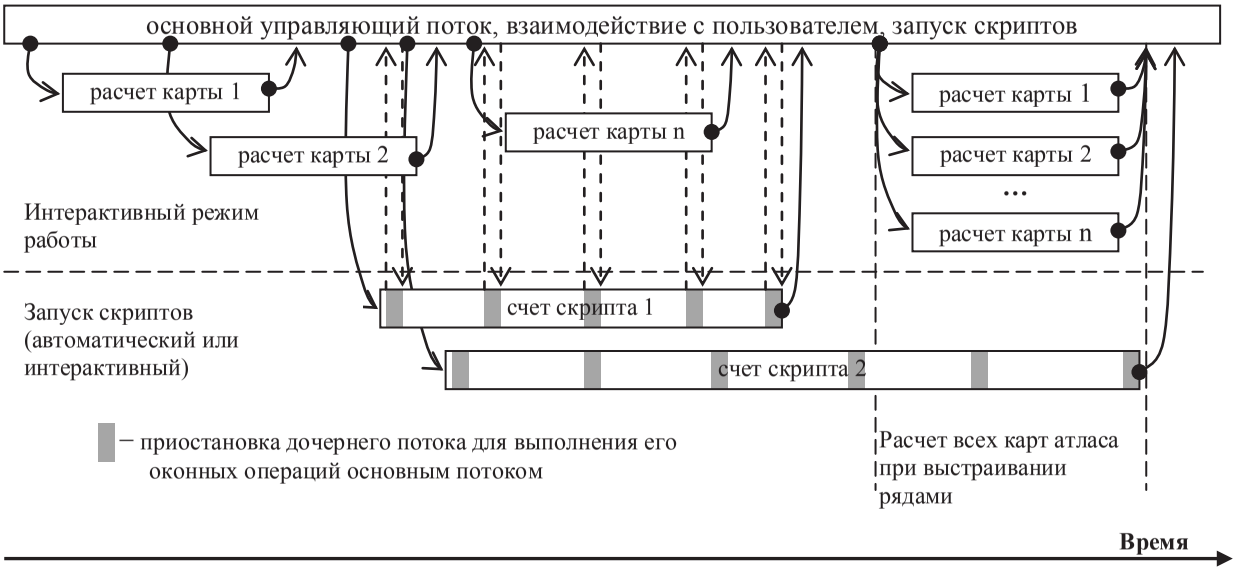

Рис. 1. Схема взаимодействия потоков в программной системе Isograph.

В средствах программирования Delphi существует класс TThread, способствующий упрощению управления потоками Windows. Трудности работы с оконным интерфейсом и, более широко, с графикой в Delphi преодолеваются следующим образом. Класс TThread содержит метод Synchronize, который является «оберткой» для какой-либо графической процедуры. Фактически вызов этого метода обеспечивает выполнение указанной графической процедуры в контексте главного управляющего потока в подходящий, выбранный для этого операционной системой момент. Таким образом, в выбранной в Isograph модели многопоточности можно признать правильным такой способ программирования, при котором расчеты проводятся длинными слитными интервалами времени, которые лишь изредка прерываются для выполнения нужных графических операций.

Потребность в пересчете карты возникает только при наступлении нескольких событий: изменение состава данных для карты, изменение параметров изображений некоторых данных или изменение масштаба карты при модификации размера окна или при приближении некоторого района. Однако перерисовка карты по команде Windows происходит гораздо чаще, например при закрытии или сдвиге какого-то вышележащего окна, перекрывавшего полностью или частично карту. Такие события для ускорения перерисовки окна карты не должны приводить к ее пересчету. Для решения этой задачи первичный вывод карты после ее расчета ведется не напрямую в окно, а первоначально на битовое изображение (bitmap) окна, которое играет роль буфера окна, хранится в памяти компьютера совместно с окном и выводится на экран всякий раз, когда поступает команда Windows о его перерисовке. К сожалению, использованный первоначально для этого стандартный класс Delphi TBitmap оказался не свободным от указанных выше недостатков, связанных с работой потоков. По этой причине он был заменен аналогичным потокозащищенным классом из библиотеки Graphics32 (http://www.graphics32.org), развиваемой международной группой разработчиков как свободно распространяемое программное обеспечение в рамках лицензии Mozilla Public License (MPL).

Итак, многопоточный режим расчета карт выглядит следующим образом. При необходимости пересчета карты создается дополнительный поток, в котором происходит расчет карты во временный буфер (битовое изображение). На время выполнения этой операции окно карты становится неактивным, чтобы исключить вероятность появления событий, которые повлекут за собой повторный пересчет. О том, что с этим окном работать пока нельзя, пользователя информирует специальный индикатор (рис. 2). По завершении расчета карты её изображение передается из временного буфера в окно карты, и это окно вновь становится активным. Важным обстоятельством является то, что во время долгого расчета, выполняемого автономными потоками, пользователь системы имеет возможность создавать новые карты или работать с теми картами, которые в данный момент не рассчитываются и поэтому не блокированы. В связи с тем, что все оконные операции выполняются монопольно основным потоком, открытие какого-либо модального диалога во время расчета совокупности карт блокирует вывод этих карт на экран, так как управляющий поток блокируется модальным диалогом до того момента, пока пользователь системы его не закроет. Это обстоятельство также надо иметь в виду при работе с системой.

Потоки точно таким же образом используются также при печати карт и при экспорте карт в файлы графического формата.

Для иллюстрации эффективности предложенного решения рассмотрим пример. Тест состоял из восьми одинаковых карт изолиний поля модели COSMO. Тест проводился на компьютере с процессором Intel® Core™ i7-2600K, имеющим 4 ядра и использующим технологию Hyper Threading, при которой на каждом вычислительном ядре одновременно выполняются по два потока команд, в результате чего операционной системой этот процессор воспринимается имеющим 8 ядер. Расчет одной карты выполнялся за 10 секунд. Время замерялось по внешнему секундомеру. Расчет восьми карт занял 26 секунд, то есть в 3 раза быстрее, чем этого можно было бы ожидать в однопоточном режиме. Графики утилизации процессора при расчете восьми карт приведены на рис. 3. Видно, что из-за прерываний, связанных с

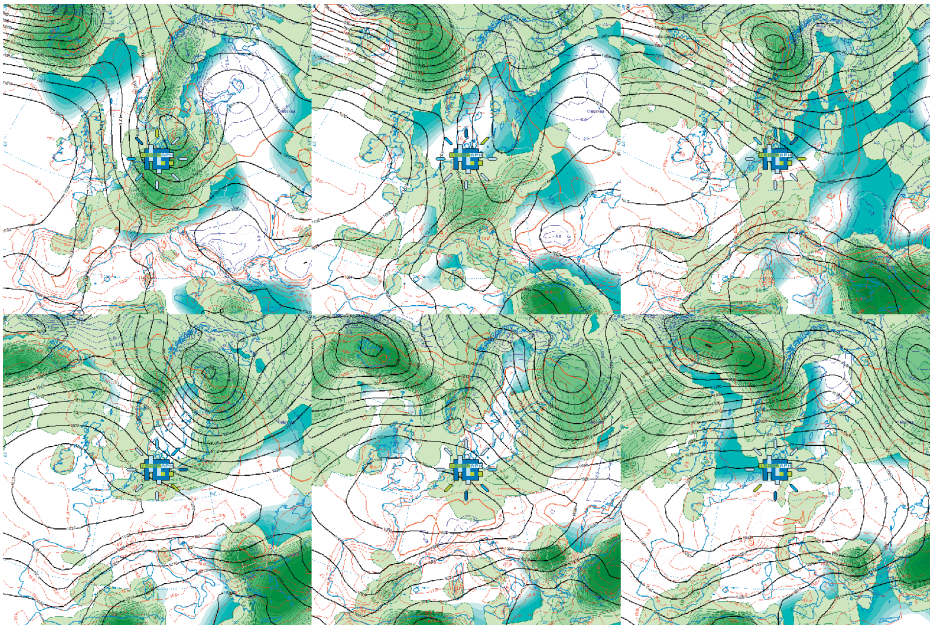

Рис. 2. Расчет совокупности карт при выстраивании их «черепицей». Карты временно заблокированы. Индикатор блокировки − в центре каждой карты.

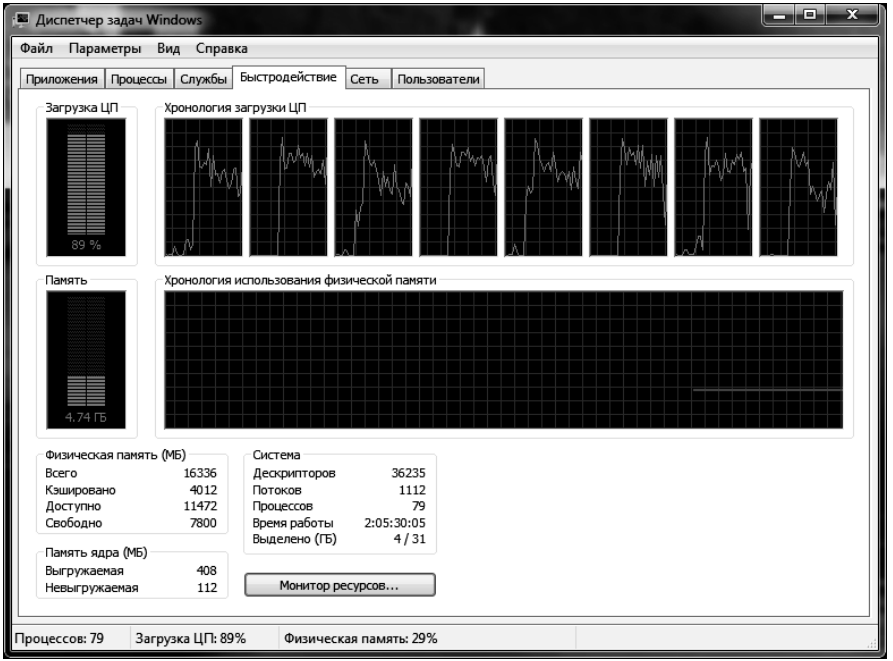

Рис. 3. Утилизация процессора при расчете восьми карт.

выполнением графических операций, мощности процессора используются не на 100 %. По этой причине, а также благодаря вытесняющей многозадачности, реализованной в операционной системе Windows, расчет десяти таких карт на том же процессоре занял 33 секунды.

При использовании многопоточного режима работы очень существенным оказывается следование правилу программирования, сформулированному выше. Так, мы видоизменили тест, о котором шла речь в предыдущем абзаце, таким образом, что на карту наносились не только изолинии, но и закрашенные изоконтуры. Тогда одна карта рассчитывалась в течение 13 секунд, а восемь карт − 117 секунд, то есть в 9 раз дольше, тогда как при однопоточном режиме можно было бы ожидать замедления только в 8 раз. Этот факт объясняется тем, что при расчете изолиний мы накапливаем вычисленные координаты их точек в списках точек, а затем, когда картина полностью получена, единовременно выводим графическое изображение всех изолиний. При закрашивании изоконтуров использован не вполне эффективный алгоритм, при котором мы рассматриваем последовательно ячейки сетки поля, и фрагменты изоконтуров, попадающие в ячейку, выводятся каждый раз, как только они найдены. Поэтому возникает большое количество графических операций, перемежающихся с расчетами, которые, как мы помним, связаны с прерываниями дочернего потока.

Другой возможностью использования потоков в системе Isograph является автоматизированный расчет карт, запрограммированный с помощью скриптов. Возможность интерпретации скриптов (программ) для автоматического создания карт погоды была добавлена недавно [3]. Естественно было бы проводить расчеты в соответствии со скриптом в отдельном автономном потоке с тем, чтобы не нарушать интерактивного режима работы пользователя. Итак, при запуске каждого скрипта создается дочерний поток, который производит интерпретацию данного скрипта и создание, а в случае необходимости, и уничтожение карт в соответствии с заданной программой (см. рис. 1). Стоит напомнить, что для создания и уничтожения окна карты

необходима синхронизация этого потока с главным потоком Isograph при помощи метода Synchronize. Одновременно таким образом могут обрабатываться несколько скриптов.

Система Isograph не имеет собственного хранилища гидрометеорологических данных. Доступ к данным осуществляется через механизм программного шлюза, функция которого − преобразовать данные из формата хранения во внутренний формат системы (см. [2]). Предполагается, что программа-шлюз может быть написана пользователем системы Isograph для данных из своего хранилища (файла или базы данных). При одновременном доступе к данным одного источника из разных потоков происходит обращение к одной и той же программе-шлюзу, а если она не является реентерабельной, то возникает ситуация гонки за данными. Чтобы не обременять пользователя чрезмерными требованиями к такой программе, естественным решением этой проблемы явилось размещение фрагмента кода с обращением к шлюзу для доступа к данным в критической секции. То есть указанный фрагмент обрамляется специальными системными вызовами, после чего операционная система сама следит за тем, чтобы данный код единовременно исполнялся бы лишь одним потоком. Таким образом, все потоки, связанные с обработкой скриптов, получают доступ к гидрометеорологическим данным по очереди.

Все потоки, связанные с интерпретацией скриптов, заносятся в соответствующий список, за который отвечает главный поток системы Isograph, и содержатся в нём до момента их окончания и уничтожения. Пока данный список не пуст, Isograph не может быть закрыт. Но каждый и этих потоков может быть завершен досрочно по команде пользователя при помощи соответствующего пункта меню.

Таким образом, графическая система Isograph преобразована в многопоточную программу, что в большинстве случаев приводит к ускорению работы в многооконном режиме, а также позволяет организовать построение предопределенных карт в автоматическом фоновом режиме без блокировки интерактивной работы пользователя.

Список использованных источников

1. *Алферов Ю.В.* Принципы построения автоматизированной графической системы для визуализации полей метеоэлементов в научных исследованиях // Труды Гидрометцентра России. – 2000. – Вып. 334. – С. 180–189.

2. *Алферов Ю.В.* Автоматизированная графическая система для визуализации результатов численных прогнозов // Труды Гидрометцентра России. – 2003. – Вып. 338. – С. 119–124.

3. *Алферов Ю.В., Копейкин В.В.* Аспекты автоматизации в гидрометеорологической системе визуализации Isograph // Труды Гидрометцентра России. – 2011. – Вып. 346. – С. 17–27.

4. *Вильфанд Р.М., Ривин Г.С., Розинкина И.А.* Система COSMO-Ru негидростатического мезомасштабного краткосрочного прогноза погоды Гидрометцентра России: первый этап реализации и развития // Метеорология и гидрология. – 2010. – № 8. – С. 5–20.

5. *Осипов Д.* Delphi. Профессиональное программирование. – СПб; М.: Символ-Плюс, 2006. – 1056 с.

Поступила в редакцию 15.01.2015 г.