Параллельные

Евгений Балдин задумывается о будущем программирования и приходит к выводу, что без параллелизма ему не жить.

Наш эксперт

Евгений Балдин
Физик, который действительно знает, что такое нехватка вычислительных ресурсов.

Я думаю, что все, кто читает эти строки, как-то умеют программировать: в современном мире уметь использовать компьютер эффективно, то есть переваливать на него тупую и нудную работу – это необходимость. Ну, или хотя бы знают, что такое алгоритм – набор инструкций, описывающий порядок действия исполнителя. Другое дело, что почти все, за исключением очень небольшого числа узких специалистов, под словом «алгоритм» понимают традиционную последовательную парадигму, где слово «порядок» подменяет слово «последовательность». Пора это менять, и чем раньше это случится и чем с большим числом людей – тем лучше.

Этот текст является введением в цикл статей, который я затеял с целью ликбеза по параллельным технологиям. Этим технологиям нам волей-неволей придется как-то обучиться, так как иного способа повысить скорость выполнения программы в обозримом будущем не предвидится. В настольных компьютерах и телефонах теперь растет не частота процессора, а число ядер. Проблема освоения параллельных алгоритмов теперь встает в полный рост почти перед любым программистом – и будет стоять до тех пор, пока отсутствуют волшебные распараллеливающие компиляторы.

Основная проблема с параллелизацией уже существующих и хорошо проверенных на практике алгоритмов заключается в том, что далеко не все их можно разбить на независимо выполняемые фрагменты. Даже если дать современному компилятору сто тысяч процессоров, он не сможет в общем случае без подсказки программиста добиться сколько-нибудь значимого прироста производительности. Поэтому на сегодня вменяемая параллелизация лежит на человеческом уровне, то есть почти все приходится делать вручную. Это как возврат ко временам, когда оптимизирующие компиляторы с языков высокого уровня были не способны соревноваться с программистами на ассемблере. Со временем это поменяется, но для этого нужно создать хорошо подготовленное и квалифицированное сообщество, готовое писать, тестировать, да и просто использовать новые инструменты и понятия.

Идея цикла появилась у меня, когда я ради интереса забрел в Институт вычислительной математики и математической геофизики на двухдневную «Школу по параллельному программированию гибридных кластеров», где вкратце знакомили с технологиями MPI и CUDA. Вторую часть практических занятий я частично проигнорировал, так как был занят уговариванием ведущих занятия Михаила Остапкевича и Константина Калгина включиться в процесс ликвидации безграмотности окружающих, и меня в частности, на более литературном уровне.

Я ни в коем случае не буду единственным автором статей цикла, так как не являюсь специалистом в этом вопросе. Как и в случае цикла по свободной системе анализа R, который публиковался в Linux Format с 2008 по 2010 годы, статьи будут писать разные люди. Я же буду выполнять роль координатора и «причесывателя текста». В случае с R такая стратегия привела в 2012 году к изданию первой толстой книги по R в России «Наглядная статистика. Используем R!» за авторством А.Б. Шипунова и шести его сподвижников (я тоже удостоился чести стать одним из них). Есть надежда, что и в случае этого цикла тоже удастся собрать материал для большой и серьезной, хоть и популярной, литературной формы.

Предварительный план цикла статей ни в коем случае не будет истиной в последней инстанции и легко поменяется в момент, когда кого-то из участников процесса осветит светлая и интересная идея, а также если внезапно число участников увеличится. Порядок статей тоже весьма примерный: статьи могут как

Нам нет преград?

Гордон Мур, один из основателей компании Intel, почти 50 лет назад указал на экспоненциальное развитие вычислительной техники, однако шесть лет назад он также указал на ограничения, которые невозможно преодолеть современной индустрии: скорость света и размер атома. Что же может спасти «Закон Мура»? Ответ – параллельные вычисления!

технологии: Старт

разделяться, если информации слишком много, так и сливаться, если ее не хватает.

Часть потенциальных авторов статей пока еще не в курсе, что они будут что-то писать, и мне еще предстоит им об этом сообщить. Я также совершенно не против, если найдутся энтузиасты, которые сообщат об этом мне раньше, чем я найду их. Проще со своими вне всякого сомнения ценными предложениями, критикой и советами связаться со мной по электронной почте E.M.Baldin@inp.nsk.su. Все будет принято с благодарностью.

А теперь, чтобы было что критиковать, план рассказов на почти год вперед:

» **Февраль** Общее введение в цикл статей. А также – общая информация на тему, как запускать программы на типичном кластере. Параллельными вычислениями можно заниматься и дома, но в случае серьезных вычислений придется прибиться к чему-то более серьезному – например, к ближайшему университету.

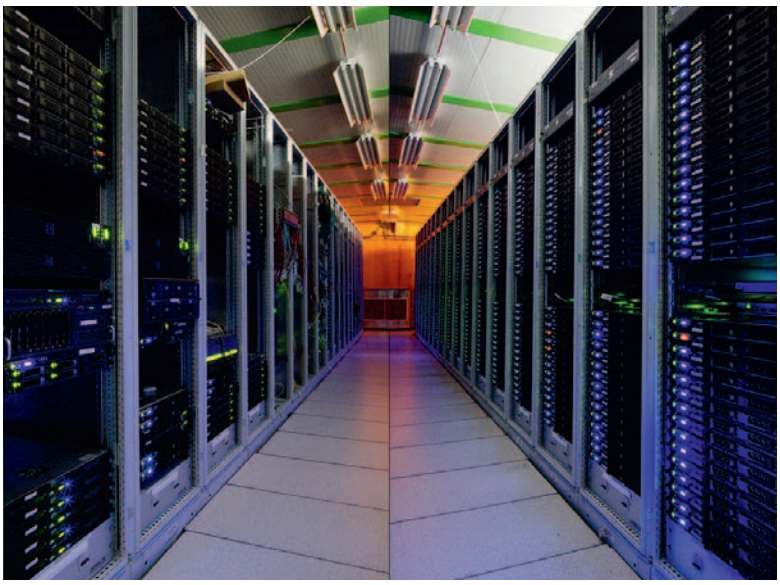
» **Март** MPI – стандарт для обмена информации между процессами, выполняющими одну и ту же задачу. Хорошо подходит для современных «не сильно» многоядерных процессоров, хотя и ориентирован в основном на системы с распределенной памятью. На примере игры клеточных автоматов будет показано, как можно разделить программу на два потока. Главное – следить за границами разделения!

» **Апрель-май** CUDA – то, что следует применять, если вычисления нужно проделать здесь и сейчас. Для тренировки можно использовать современные карты NVIDIA, а за относительно разумные деньги можно получить киловаттную печку с парой тысяч ядер на борту. Минусы – закрытое программное обеспечение, привязанное к одному-единственному вендору, но зато работает здесь и сейчас. Значительная тема.

» **Июнь** OpenCL – открытое стандартное окружение для написания параллельных программ. Это средства пока еще не так эффективно, как CUDA, зато понемногу проникает во все сферы, включая мобильные телефоны и планшеты. Также, в отличие от CUDA, OpenCL не привязано к графическим ускорителям и может использоваться на классических многопроцессорных системах, и даже кое-кем используется для программирования FPGA (ПЛИС – программируемая логическая интегральная схема). Возможно, это будущее параллельного программирования.

> «На сегодня параллелизация лежит на человеческом уровне.»

» **Июль** вычисления@home – наверняка все слышали про SETI@home. Это тоже метод, позволяющий ускорить свои вычисления. И хотя тут, скорее всего, важнее социальная инженерия, но и параллельное программирование тоже важно. Я как-то уже приставал к разработчиком самого крупного российского проекта этого рода – SAT@home, с предложением написать статью для **LXF**. Попробую в этот раз быть поубедительней. Как минимум, возьму электронное интервью.

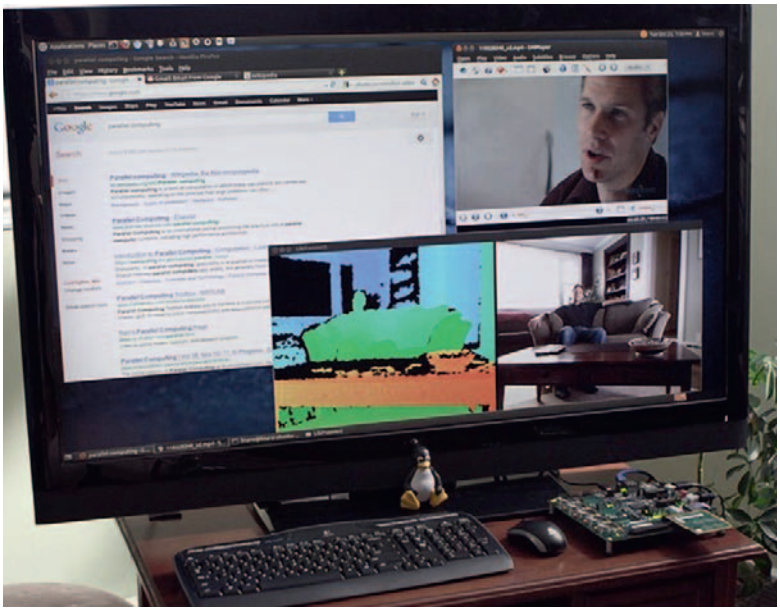


› **Один из кластеров в CERN (фото Андрея Зайцева). Параллелизация тут идет на уровне данных, но ресурсов нужно много.**

» **Август** GRID – параллелить можно не только алгоритмы, но и данные. Большой адронный коллайдер выдает просто гигантское число независимых событий, каждое из которых можно обрабатывать на своем личном процессоре. Да, тут все ПО однопоточное, но умение держать всю эту ораву разношерстных ячеек в узде пришло не само собой, и этот опыт требует осмысления.

» **Сентябрь** Динамический виртуальный вычислительный кластер – позволяет наплевать на особенности кластерной инфраструктуры реальных мощностей и относительно легко перенести свое родное и привычное окружение на другие железные рельсы. Это из серии «голь на выдумки хитра», или – как использовать университетские мощности, ничего не меняя в своем ПО.

Кроме статей о программировании, есть желание описать «железные» платформы, на которых можно посчитать что-то параллельным образом. В каком порядке они появятся и между какими статьями вставятся, в значительной степени зависит от того, в какой момент будет получен доступ к «телу» и сколько времени »



Плата Parallella и TuX. Возможное будущее домашних параллельных вычислений, пока еще не «причесанное».

Итак: есть задумки для 12 статей. Какие-то из них наверняка не удадутся, но я надеюсь, что им на смену в сообществе энтузиастов возникнут новые идеи!

ПО для работы с кластером

Безусловно, заниматься параллельным программированием можно и дома на коленке. Более того, все примеры перед публикациями будут опробованы на домашнем компьютере. Да, можно на карманные деньги купить распоследнюю Nvidia ® Tesla ® и заняться добычей биткойнов, или обработкой ну очень больших фотоснимков. Но серьезные вычислительные мощности и дорогие платформы обычно хранятся за пределами уютных домашних квартир, а именно на специализированных кластерах.

В качестве примера я приведу Информационно-вычислительный центр Новосибирского государственного университета (http://www.nusc.ru/). Там работают довольно грамотные специалисты, поэтому можно с удовольствием побродить по страничкам центра, поглядеть на статистику, почитать документацию, осознать, что до стопроцентной загрузки центру весьма далеко... и это нормально. Пиковые мощности ни в коем случае не должны быть сравнимы со средней загрузкой. Если вы живете в крупном городе, то у вас поблизости гарантированно есть подобный центр, куда можно сходить/договориться об использовании. Если вы студент университета, то следует просто пойти на соответствующий спецкурс. Некоторые из вычислительных центров ведут коммерческую деятельность, торгуя мощностями в обмен на разного рода ресурсы. В крайнем случае, можно прикупить немножко «Амазонского облака». В общем, всегда возможно найти тот или иной способ прицепиться к кластеру. Что же вас там ожидает?

> «Всегда возможно найти способ прицепиться к кластеру.»

Во-первых, следует осознавать, что все кластеры работают под управлением того или иного дистрибутива GNU/Linux. В Сети ходит довольно смешной рассказ, как одна большая и мягкая фирма с помпой открывала кластер в Томском университете под своей проприетарной системой, ориентированной уйдет на его «препарирование». С моей стороны я обязательно расскажу про проект Parallella (http://www.parallella.org/) – многоядерный сопроцессор на плате с потреблением 5 Вт и стоимостью в 100 долларов. Это скорее образовательный проект, но с чего-то нужно же начинать работать дома. Другим кандидатом на «вивисекцию» безусловно является модуль Tesla от компании Nvidia – реально мощный калькулятор; правда, им нужно уметь пользоваться. Также есть надежда получить доступ к новому HPC-процессору от Intel® Xeon® Phi и даже к вычислительной системе на FPGA.

Экспресс-интервью

Буквально на пару вопросов ответил начальник отдела разработок Adapteva **Роман Троган** [Roman Trogan]. Следует отметить, что введение было написано до этого электронного мини-интервью, так что его слова на текст не повлияли, но весьма удачно показали необходимость ликбеза в области параллельных вычислений.

LXF: Что нового вы привнесли в этот мир?

Роман Троган: Последние четыре года Adapteva занимается разработкой многоядерной архитектуры Epiphany. Разработанные нами чипы состоят из легко масштабируемого набора простых RISC-процессоров, объединенных в быструю сеть с общей разделяемой памятью. Для программирования Epiphany можно использовать C/C++. Сейчас основные усилия нашей группы разработчиков направлены на проект Parallella, основой которого и является наша многоядерная архитектура. Нашей целью является сделать параллельные вычисления вездесущими путем предоставления разработчикам и энтузиастам доступной, открытой и легко программируемой платформы.

LXF: Но зачем?

РТ: Мы считаем, что будущее компьютеров – за параллельными вычислениями. К сожалению, здесь и сейчас мало кто знает, как программировать параллельные системы, и это по праву считается весьма сложным занятием. Одной из причин такого состояния дел является отсутствие доступной и простой в эксплуатации параллельной платформы. Предоставляя сообществу проект Parallella, мы надеемся значительно поднять уровень образования в области параллельных вычислений.

Роман Троган намерен повысить осведомленность о параллельных вычислениях.

» **Пропустили номер?** Узнайте на с. 104, как получить его прямо сейчас.

на высокопроизводительное вычисления. Когда помпа утихла, а в центральных газетах были написаны соответствующие хвалебные статьи, системные администраторы на месте, хмыкнув, снесли эту систему и установили то, что годится для работы и привычно пользователям. В этой сфере GNU/Linux не нужно побеждать – он там есть и является стандартом.

Как следствие, доступ идет через SSH. Вас, скорее всего, попросят прислать открытую половинку ключа, получаемого командой **ssh-keygen**. Вход по паролю не приветствуется, так как пользователей много, а центры пытаются взломать непрерывно.

В силу большого числа пользователей, никто не даст вам в руки машину с дорогостоящим оборудованием, а попадете вы на один из гостевых компьютеров, где вы сможете собрать свою программу и поставить ее в очередь на выполнение. Надеюсь, вы читали «Понедельник начинается в субботу» братьев Стругацких и помните, чем занимался главный герой – Привалов: он заведовал вычислительным центром НИИЧаВо, до которого пользователи не допускались. Они только приносили свои пачки перфокарт, которые, в свою очередь, в машину загружали специальные техники в зависимости от наличия свободных ресурсов. Сейчас, конечно, не так все печально, и задачу грузят не техники, а специальный планировщик задач; но, поставив свою задачу в очередь, вы на нее никак повлиять уже не можете, а можете только ждать результатов.

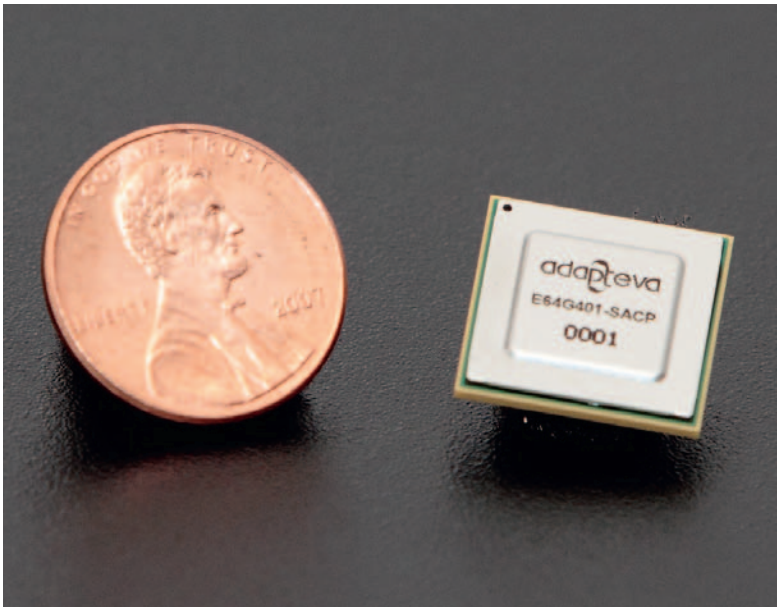
Планировщики задач могут быть как проприетарными, например, Altair PBS Pro, так и свободными, такими как TORQUE (форк OpenPBS) или Oracle Grid Engine (в девичестве Sun Grid Engine). В этом случае для работы с ними можно воспользоваться более-менее стандартными утилитами.

Традиционно в Unix для управления очередями заданиями [Batch Queues] со стороны пользователя зарезервированы специальные команды: **qalter**, **qdel**, **qhold**, **qmove**, **qmsg**, **qrerun**, **qrls**, **qselect**, **qsig**, **qstat** и **qsub**. Как всякий древний стандарт, каждая из команд обросла кучей подробностей, ключиков, переменных окружения и условий применения. Обычно для работы хватает трех команд:

- **qsub** – запуск заданий;
- **qstat** – вывод статистики по заданиям в очереди;
- **qdel** – удаление задания из очереди.

Команде **qsub** нельзя просто подсунуть исполняемый файл. Оный нужно запускать из специально сформированного скрипта, где, кроме вызова самого файла, нужно передать системе информацию о параметрах задания. Например, типичная шапка моего задания в окружении Sun Grid Engine выглядит так:

```
#!/usr/bin/perl -w
#  SGE vars
# ------------------------------------------
# --        use perl                  --
#$ -S /usr/bin/perl
# ------------------------------------------
# --           batch name             --
#$ -N analyze-run-log
# ------------------------------------------
# --    What to redirect to where     --
#$ -cwd
#$ -o $JOB_NAME.$JOB_ID
#$ -j y
# ------------------------------------------
# --           Qeue list              --
#$ -q remote
# ------------------------------------------
# --           mail me                --
#$ -M E.M.Baldin@inp.nsk.su
#$ -m be
# ------------------------------------------
```



Многоядерный процессор от Adapteva.

Тут параметры передаются после комбинации символов #$. Скрипт не обязательно должен быть на *bash*, интерпретатор можно указать с помощью ключика **-S**, имя задания указывается с помощью ключика **-T**, в качестве рабочей директории объявляется текущая (**-cwd**), имя лог-файла (**-o**) строится из имени задачи и номера задания, имя очереди (**-q**) – remote, а все сообщения о начале и окончании (**-m**) выполнения задания велено отсылать на мой e-mail (**-M**). Тут нет никаких определений на тему, какой компьютер мне нужен и что на нем должно стоять, так что это задание поставится без разбора куда.

Если требуется уточнить, какие именно ресурсы нужны для выполнения задачи, например: четыре OpenMP-процесса, и каждому требуется по 2000 МБ ОЗУ, то строчка задания может выглядеть как-то так:

```
#PBS -l select=1:ncpus=4:ompthreads=4:mem=2000m
```

Подробности следует узнавать в документации, выложенной на страничке конкретного кластера. Все, что перечислено выше, можно задать и в командной строке **qsub**, но ключиков и параметров так много, что лучше их записывать в файле задания.

Чтобы понять, какие из заданий запущены, а какие бездельничают, можно воспользоваться утилитой *qstat*, только следует отфильтровать свои задания, так как их может быть очень много:

```
> qstat | grep baldin
 247236 0.55500 BHABHA-118 baldin   dr  12/29/2012 08:39:35
remote@sscc-142
 249219 0.55500 JPSI2MUMUN baldin   r   12/30/2012 13:23:51
remote@sscc-180
 249363 0.55500 JPSI2EE-12 baldin   qw  01/03/2013 15:06:19
```

Здесь три задачи за моим именем. Одна из них умерла (**dr**) и по какой-то причине не была удалена из записи, одна работает (**r**), а третья висит в очереди. Можно удалить одно из заданий:

```
> qdel  249219
baldin has registered the job 249219 for deletion
```

Это, безусловно, не единственный способ общения с очередью заданий. В частности, система управления очередями Simple Linux Utility for Resource Management (http://slurm.net), установленная на самом быстром на конец 2012 года компьютере из TOP500, имеет свой собственный набор команд для постановки задачи в очередь, получения статистики и передачи заданию сигналов от пользователя. К счастью, во всех случаях логика примерно одна и та же, поэтому обучиться ей по месту можно очень быстро. LXF