Кинограмма напряжений показывает, как формируется поле напряжений. Начиная из центра широкой грани, волна концентрируется в середине волновода, поднимаясь к узкой грани (рис. 6, *а*, *б*, *в*).

При медленном приложении нагрузки, можно увидеть поле максимальных напряжений, возникающих в конструкции (см. рис. 5, *а*, *б*).

По результатам проведенных расчетов можно сделать следующие выводы:

1. Определены параметры экспериментальной установки, позволяющие получить максимальные K_y для заданных физико-механических характеристик и геометрии акустического трансформатора.

2. В ходе вычислительного эксперимента подтверждена прочность акустического трансформатора при K_y^{max}.

3. Узкая грань при работе изгибается, что приводит к падению эффективности технологического процесса при использовании волноводов.

Вад.В. Воеводин

Научно-исследовательский вычислительный центр МГУ им. М.В. Ломоносова

ХАРАКТЕРИСТИКИ РАБОТЫ С ПАМЯТЬЮ И ЭФФЕКТИВНОСТЬ РАБОТЫ ПРОГРАММ

На сегодняшний день существует много средств и технологий для повышения эффективности выполнения программ, однако в большинстве своем они ориентированы либо на работу в рамках конкретной аппаратуры, либо на оптимизацию определенного свойства программы. Представленная работа направлена на создание унифицированного подхода к изучению эффективности всего процесса отображения алгоритма на различные аппаратные платформы. Основной идеей является детальное описание трех этапов процесса отображения – алгоритмов, программ и их реализаций – с помощью набора характеристик, оценивая которые можно оценить и эффективность самого процесса

отображения. Некоторая часть данных характеристик является аппаратно-независимой, что позволяет абстрагироваться от выбранной платформы и исследовать эффективность процесса отображения алгоритма в общих терминах. Принципиальной позицией излагаемого подхода является то, что характеристики необходимо выделять на всех этапах отображения, начиная от свойств самого алгоритма и заканчивая характеристиками среды времени выполнения и аппаратуры. В докладе приводится описание общего процесса отображения, а также примеры характеристик на разных этапах.

На эффективность процесса отображения оказывают влияние многие факторы, и одним из самых важных среди них является эффективность работы с памятью. Организовать эффективную работу с памятью в программах непросто, поскольку на нее оказывает влияние множество различных параметров как самого алгоритма или программы, так и используемой аппаратуры. Память в современных вычислительных платформах имеет сложную структуру и иерархию, поэтому число параметров аппаратуры велико: размер и количество уровней кэш-памяти, объем и скорость чтения/записи в ОП, параметры работы контроллера памяти, размер строки, степень ассоциативности кэш-памяти, размер и строение TLB-буфера и т.д. Учитывать все эти параметры трудно, однако каждый из них может оказывать большое влияние на эффективность работы программы. Например, различие двух данных фрагментов кажется несущественным:

1)
```
for (i = 0; i < length; i+=1040)
        sum = sum + a[i];
```

2)
```
for (i = 0; i < length; i+=1024)
        sum = sum + a[i];
```

В обоих вариантах происходит суммирование элементов вещественного массива a длиной length, а единственное отличие заключено в шаге, с которым происходит выборка элементов массива – в первом случае шаг равен 1040, во втором – 1024. Проведенные на процессоре Intel Clovertown эксперименты по-

казали, что при длине массива в 2^{24} элементов первый вариант эффективнее второго в 8 раз! Причина – в ассоциативности кэш-памяти: во втором варианте требуемые данные постоянно попадают в одни и те же группы строк кэш-памяти, задействуя таким образом только малую ее часть, в то время как в первом варианте используется гораздо больший ее объем.

Рассмотрим второй пример, который отражает иные особенности работы с кэш-памятью:

```
1) for (i = 1; i < length; i++) {
        cache_annil[i] = i;
        helper += cache_annil[i-1];
}
<основной алгоритм>

2) for (i = 1; i < length; i++)
        cache_annil[i] = i;
   for (i = 1; i < length; i++)
        helper += cache_annil[i-1];
   <основной алгоритм>
```

Предположим, нас интересует время работы некоторого основного алгоритма, без учета служебных операций. Перед этим алгоритмом необходимо очистить кэш-память от записанных туда ранее данных для более точного измерения времени выполнения алгоритма. Для этого используется массив cache_annil,причем длина массива length подобрана так, чтобы заполнить всю кэш-память. Отличие двух вариантов только в том, что в первом варианте оба действия (запись нового значения в массив cache_annil и прибавление предыдущего элемента к общей сумме) происходят в рамках одного цикла, а во втором – в разных. Однако это приводит к увеличению времени выполнения, и, как следствие, неверной оценке работы основного алгоритма в первом примере, причем различие

во времени может быть существенным, в зависимости вычислительной сложности самого алгоритма. Причина в том, что в первом варианте весь массив cache_annil просматривается за один проход, и после этого в кэш-памяти остаются модифицированные данные, которые в основной части программы приходится вытеснять, что, естественно, отражается на эффективности этой основной части. Во втором варианте этого не происходит, поскольку модифицированные данные вытесняются в оперативную память до основного алгоритма, во время второго цикла.

Данные примеры демонстрируют разнообразие факторов, которые могут влиять на эффективность процесса отображения. Важной особенностью данных факторов, или характеристик аппаратуры, является то, что чаще всего их нельзя изменять. Программу необходимо составлять таким образом, чтобы в ней учитывались особенности выбранной аппаратуры. А для этого необходимо выделить те характеристики алгоритма и программы, которые могут влиять на работу с памятью, после чего, изменяя их, можно подстраивать программу под аппаратуру. Примерами таких характеристик являются пространственная и временная локальности, общее число обращений в память и т.д. На их исследование и описание и направлена данная работа.

Составим формальное описание работы с памятью. На данный момент будем рассматривать последовательные программы. Работу программы с памятью можно представить в виде потока, который описывается последовательностью обращений к используемым в программе переменным в том порядке, в котором это происходит во время работы программы. Если задействован элемент массива или другой составной переменной, то учитывается именно этот элемент, а не вся составная переменная. Таким образом, в последовательность могут входить как скалярные переменные, так и составные (если в некоторой операции адресуется такая переменная целиком) или их элементы.

Если для работы с элементами составной переменной используется некоторый итератор, то для каждого конкретного зна-

чения итератора в поток обращений записывается отдельное обращение. Например, в случае итерирования массива, если в качестве индекса массива A используется некоторая переменная i, то в потоке обращений необходимо для каждого значения i подставлять это конкретное значение. Например, если в программе есть код

for (i=0;i<5;i++) a[i]=0,

то поток обращений к массиву A будет выглядеть следующим образом: a[0],a[1],a[2],a[3],a[4].

Каждое обращение в потоке A можно поменять на виртуальный адрес соответствующей переменной, получив тем самым поток виртуальных адресов. В этом случае происходит переход на другой уровень абстракции, поскольку работа происходит не с переменными, а с адресами. Далее, при замене виртуальных адресов на физические формируется поток физических адресов. Формулируя и рассматривая характеристики задачи на каждом из уровней (переменные, виртуальные и физические адреса), можно получать описания поведения задачи на разных уровнях абстракции и зависимости от конкретной аппаратуры. Уровень физических адресов позволяет получить наиболее точное описание, поскольку оперирует реальным расположением адресуемых данных в памяти во время выполнения программы, однако его сложнее учитывать, так как расположение данных может изменяться от запуска к запуску.

В докладе вводится более подробное описание данных уровней абстракции и отдельно рассматривается уровень переменных, в рамках которого формулируются некоторые характеристики работы с памятью, определяющие свойства пространственной и временной локальности данных. Далее приводится объяснение, каким образом данные характеристики могут влиять на эффективность работы с памятью, и показывается важность анализа этих характеристик для получения эффективного процесса отображения.

Работа выполнена при поддержке гранта РФФИ № 09-07-00168-а и Федерального агентства по науке и инновациям, государственный контракт № 02.740.11.0388.