**ЭТАПЫ И ПРИМЕР (МММ) ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ЗЕРНИСТЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ**

**И ОБМЕНА ДАННЫМИ**

Этапы получения псевдокода параллельного зернистого алгоритма для реализации на суперкомпьютерах с распределенной памятью (исходный последовательный алгоритм задан гнездом циклов, автоматизация распараллеливания не используется):

* Информационная структура алгоритма: выявление (не обязательно формализованное) информационных зависимостей между операциями.
* Тайлинг (не нарушающий порядок выполнения зависимых операций).
* Запись параллельных зернистых вычислительных процессов (без распределения массивов между процессами и указания обменных операций): псевдокод уровня глобальных циклов, псевдокод уровня операций тайла. Детальное понимание распределения вычислений.
* Распределение входных и выходных данных (следует из распределения вычислений).
* Общее не формализованное представление о работе параллельного алгоритма, об обмене данными и выводе результатов вычислений.
* Выделение массивов. Приватизация (если возможно) массивов.
* Запись (псевдокод) тайла с выделенными массивами.
* Оптимизация вычислений в тайлах (например, введение новых массивов, оптимизация работы с кэшами, вычисление границ цикла вне цикла).
* Детальное понимание коммуникаций. Структурирование коммуникаций (например, бродкаст, трансляция).
* Псевдокод параллельного зернистого алгоритма, включающий пересылку процессам входных данных, коммуникационные операции, вывод результатов вычислений.

**Пример: параллельный алгоритм перемножения квадратных матриц**

(варианты 1 и 2 ЛабМММ)

Дан *ijk*-алгоритм перемножения двух квадратных матриц порядка *N*:

do *i =* 1*, N*

do *j =* 1*, N*

*S*1(*i,j*)*:* *c*(*i,j*) *=*0

do *k =* 1*, N*

*S*2(*i,j,k*)*:* *c*(*i,j*) *= c*(*i,j*) *+ a*(*i,k*) *b*(*k,j*)

enddo

enddo

enddo

Требуется разработать параллельные алгоритмы согласно варианту 1 и варианту 2 (варианты отличаются только коммуникационными операциями).

Тайлинг: *r*1 = 1*,* цикл *i* глобальный не разбиваемый,

*Q*2– параметр, *r*2,

*r*3 *=* *N*; цикл *k* локальный не разбиваемый;

s-координата: *j*;

коммуникации (вар. 1): трансляция части (согласованной с тайлом) *A,*

коммуникации (вар. 2): бродкаст части (согласованной с тайлом) *A*.

Бродкаст (одновременное распространение) – это передача данного группе процессоров, в которых данное одновременно (на одной итерации) используется как аргумент. Трансляция – это передача данного от процессора к процессору в случае, если элемент массива используется в разных процессорах по очереди.

Рассмотрим этапы получения псевдокода.

**Информационная структура алгоритма.** Зависимости алгоритма задаются функциями

(*i,j,*1), ,

(*i,j,k*), .

**Тайлинг.** Цикл с параметром *i* является глобальным не разбиваемым, цикл с параметром *k* является локальным не разбиваемым (в задании так сказано). Разобьем цикл с параметром *j*. Через *Q*2обозначеночисло итераций в глобальном (т.е. в первом новом) цикле, а через *r*2 обозначено (наибольшее) число итераций в локальном (т.е. во втором новом) цикле; *r*2. Получим

do *igl =* 1*, N*

*i* = *igl*

do *jgl =* 0, *Q*2–1

do *j =* 1+*jglr*2, min((*jgl*+1)*r*2, *N*)

*S*1*:* *c*(*i,j*) *=*0

do *k =* 1*, N*

*S*2*:* *c*(*i,j*) *= c*(*i,j*) *+ a*(*i,k*) *b*(*k,j*)

enddo

enddo

enddo

enddo

Операторы *S*1 и *S*2 окружены одним и тем же набором глобальных циклов, поэтому имеется тайл только одного типа:

do *igl =* 1*, N*

do *jgl =* 0, *Q*2–1

Tile(*igl,jgl,*0)

enddo

enddo

где Tile(*igl,jgl,*0) имеет вид

*i* = *igl*

do *j =* 1+*jglr*2, min((*jgl*+1)*r*2, *N*)

*S*1*:* *c*(*i,j*) *=*0

do *k =* 1*, N*

*S*2*:* *c*(*i,j*) *= c*(*i,j*) *+ a*(*i,k*) *b*(*k,j*)

enddo

enddo

Обоснуем корректность тайлинга (для любого варианта). Имеются зависимости *S*1(*i,j*)→*S*2(*i,j,*1), *S*2(*i,j,k*–1)→*S*2(*i,j,k*). Достаточные условия допустимости тайлинга выполняются: для любой зависимости *S*α(*I*)→*S*β(*J*) имеет место β≥α и, если у *I* и *J* есть координата с одинаковым номером, её значение в *J* не меньше, чем в *I*. Заметим, что в данном конкретном случае (варианты 1 и 2) порядок выполнения операций при тайлинге не изменяется, поэтому обосновать допустимость тайлинга можно без анализа зависимостей.

**Запись параллельных зернистых вычислительных процессов.** Из условия следует, что *Q*2 число процессов, предназначенных для реализации алгоритма. Единый для каждого из *Q*2 процессов псевдокод параллельного алгоритма (без учета операций обмена данными) можно записать следующим образом (*p=jgl* номер процесса):

Для каждого процесса Pr*p*, 0*p**Q*2–1:

do *igl =* 1*, N*

Tile(*igl,p,*0)

enddo

Операции тайла Tile(*igl,p,*0):

*i* = *igl*

do *j =* 1+*p* *r*2, min((*p*+1)*r*2, *N*)

*S*1(*i,j*)*:* *c*(*i,j*) *=*0

do *k =* 1*, N*

*S*2(*i,j,k*)*:* *c*(*i,j*) *= c*(*i,j*) *+ a*(*i,k*) *b*(*k,j*)

enddo

enddo

В нулевом процессе Pr0 осуществляются все вычисления алгоритма, для которых 1*j**r*2; в процессе Pr1 осуществляются вычисления, для которых *r*2*+*1*j*2*r*2. В процессе Pr*p*, кроме, возможно, (*Q*2–1)-го процесса, осуществляются все вычисления алгоритма, для которых 1+*p* *r*2*j*(*p*+1)*r*2; в процессе с номером *Q*2–1 осуществляются вычисления алгоритма, для которых 1+(*Q*2–1)*r*2*j*min(*Q*2*r*2, *N*).

**Распределение входных и выходных данных.** Соответственно распределению вычислений происходит распределение между процессами элементов матриц *B* и *C* (согласно заданиям вариантов 1 и 2 элементы матрицы *A* назначаются процессам динамически). Произвольному процессу Pr*p* распределяются столбцы матрицы *B* с номерами с1+*p* *r*2 по min((*p*+1)*r*2, *N*). В Pr*p* вычисляются столбцы матрицы *C* с теми же номерами. Распределение входных данных осуществляет нулевой процесс. Процесс Pr*p*, *p>*0, отправляет процессу Pr0 «свою» часть вычисленной матрицы *C*. «Cвою» (для процесса Pr*p*) часть матрицы *B* обозначим *Bp*, «свою» часть вычисленной матрицы *C* обозначим *Cp*. Нулевой процесс формирует всю вычисленную матрицу *C*.

**Общее представление о работе параллельного алгоритма и об обмене данными.** Эти рассуждения здесь не приводим (в лабораторной работе этот пункт тоже можно опустить).

**Выделение массивов.** **Приватизация массивов.** *Bp* и *Cp*, 0*p**Q*2–1, – матрицы размера *N×r*2 – приватизируются процессом Pr*p*. *AP* – вектор размера *N* – формируется (при каждом фиксированном *igl*) в процессе Pr0 из строк массива *a*(*i,k*). Вектор *AP* нужен (используется, служит) для трансляции или бродкаста.

**Запись тайла с выделенными массивами.** Напомним вид тайла Tile(*igl,p,*0):

*i* = *igl*

do *j =* 1+*p* *r*2, min((*p*+1)*r*2, *N*)

*S*1(*i,j*)*:* *c*(*i,j*) *=*0

do *k =* 1*, N*

*S*2(*i,j,k*)*:* *c*(*i,j*) *= c*(*i,j*) *+ a*(*i,k*) *b*(*k,j*)

enddo

enddo

Tile(*igl,p,*0) с выделенными массивами:

*i* = *igl*

do *j =* 1+*p* *r*2, min((*p*+1)*r*2, *N*)

*jp = j – p* *r*2

*S*1(*i,j*)*:*  *cp*(*i,jp*) *=*0

do *k =* 1*, N*

*S*2(*i,j,k*)*:* *cp*(*i,jp*) *= cp*(*i,jp*) *+ ap*(*k*) *bp*(*k,jp*)

enddo

enddo

**Оптимизация вычислений в тайлах.** Оптимизациювычисленийрассматривать не будем. В лабораторной работе следует вычисления оптимизировать. Например, вычисление границ цикла следует выполнять вне цикла.

**Структурирование коммуникаций.** Трансляцию данных (вариант 1), бродкаст данных (вариант 2) опишем непосредственно при записи псевдокода.

Коммуникационную операцию получения массива данных будем представлять в виде

receive(Pr; *a*; *M*),

где первый аргумент обозначает процесс, в котором вычислялся массив, второй аргумент обозначает пересылаемый массив, третий аргумент указывает объем (число элементов) массива. Коммуникационную операцию отправки массива данных будем представлять в виде

send(Pr; *a*; *M*),

где первый аргумент обозначает процесс, которому потребуются вычисленные элементы массива, второй аргумент обозначает пересылаемый массив, третий аргумент указывает объем массива.

При использовании бродкаста будем употреблять breceive и bsend.

**Псевдокод параллельного зернистого алгоритма для варианта 1.**

Для каждого процесса Pr*p*, 0*p**Q*2–1:

{if *p=*0 сформировать матрицы *Bq*, 0*q**Q*2–1,

send(Pr*q*; *Bq*; *N×r*2), 1*q**Q*2–1}

if *p>*0 receive(Pr0; *Bp*; *N×r*2)

do *igl =* 1*, N*

if *p=*0 сформировать (из строки *a*(*igl,k*)) *AP*

if *p>*0 receive(Pr*p-*1; *AP*; *N*)

Tile(*igl,p,*0)

if *p<Q*2–1 send(Pr*p+*1; *AP*; *N*)

enddo

if *p>*0 send(Pr0; *Cp*; *N×r*2)

{if *p=*0 receive(Pr*q*; *Cq*; *N×r*2), 1*q**Q*2–1,

сформировать вычисленную матрицу *C*}

**Псевдокод параллельного зернистого алгоритма для варианта 2.**

Для каждого процесса Pr*p*, 0*p**Q*2–1:

{if *p=*0 сформировать матрицы *Bq*, 0*q**Q*2–1,

send(Pr*q*; *Bq*; *N×r*2), 1*q**Q*2–1}

if *p>*0 receive(Pr0; *Bp*; *N×r*2)

do *igl =* 1*, N*

{if *p=*0 сформировать (из строки *a*(*igl,k*)) *AP*.

bsend(Pr*q*, 1*q**Q*3–1; *AP*; *N*)}

if *p>*0 breceive(Pr0; *AP*; *N*)

Tile(*igl,p,*0)

enddo

if *p>*0 send(Pr0; *Cp*; *N×r*2)

{if *p=*0 receive(Pr*q*; *Cq*; *N×r*2), 1*q**Q*2–1,

сформировать вычисленную матрицу *C*}