**ПРИМЕР ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ЗЕРНИСТЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ**

**И ОБМЕНА ДАННЫМИ**

(вариант 21 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 параллельный алгоритм согласно варианту 21.

Тайлинг: *r*1 – параметр,

*r*2– параметр,

*Q*3 – параметр, *r*3;

s-координата: *k*;

коммуникации: трансляция части (согласованной с тайлом) *C*.

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

(*i,j,*1), ,

(*i,j,k*), .

**Тайлинг.** Разобьем все три цикла (циклы с параметрами *i*, *j*, *k*); *Q*1, *Q*2. Получим

do *igl =* 0, *Q*1–1

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

do *jgl =* 0, *Q*2–1

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

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

do *kgl =* 0*, Q*3–1

do *k* = 1+ *kglr*3, min((*kgl*+1)*r*3, *N*)

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

enddo

enddo

enddo

enddo

enddo

enddo

Цикл с параметром *kgl* не может быть внутренним по отношению к локальным циклам. Кроме того, его не должно быть в окружении оператора *c*(*i,j*)*=*0, так как в окружении этого оператора первоначально не было цикла с параметром *k*. Поэтому распределим циклы с параметрами *i* и *j* между выполняемыми операторами и осуществим необходимую перестановку локальных и глобальных циклов:

do *igl =* 0, *Q*1–1

do *jgl =* 0, *Q*2–1

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

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

enddo

enddo

do *kgl =* 0*, Q*3–1

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

do *k* = 1+ *kglr*3, min((*kgl*+1)*r*3, *N*)

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

enddo

enddo

enddo

enddo(*k gl*)

enddo(*j gl*)

enddo(*igl*)

Операторы *S*1 и *S*2 окружены разными наборами глобальных циклов, поэтому требуются тайлы двух типов:

do *igl =* 0, *Q*1–1

do *jgl =* 0, *Q*2–1

Tile1(*igl,jgl*)

do *kgl =* 0*, Q*3–1

Tile2(*igl,jgl,kgl*)

enddo(*k gl*)

enddo(*j gl*)

enddo(*igl*)

Вычисления тайла первого типа Tile1(*igl,jgl*):

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

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

enddo

enddo

Вычисления тайла второго типа Tile2(*igl,jgl,kgl*):

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

do *k* = 1+ *kglr*3, min((*kgl*+1)*r*3, *N*)

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

enddo

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*.

**Запись параллельных зернистых вычислительных процессов.** Из условия следует, что *Q*3 число процессов, предназначенных для реализации алгоритма. Единый для каждого из *Q*3 процессов псевдокод параллельного алгоритма (без учета операций обмена данными) можно записать следующим образом (*p=k gl* номер процесса):

Для каждого процесса Pr*p*, 0*p**Q*3–1:

do *igl =* 0, *Q*1–1

do *jgl =* 0, *Q*2–1

if *p=*0 Tile1(*igl,jgl*)

Tile2(*igl,jgl,p*)

enddo(*j gl*)

enddo(*igl*)

В нулевом процессе Pr0 осуществляются все операции *S*1(*i,j*) и все вычисления алгоритма, для которых 1*k**r*3; в процессе Pr1 осуществляются вычисления, для которых *r*3*+*1*k*2*r*3. Далее в процессе Pr*p*, кроме, возможно, (*Q*3–1)-го процесса, осуществляются все вычисления алгоритма, для которых 1+*p* *r*3*k*(*p*+1)*r*3; в процессе с номером *Q*3–1 осуществляются вычисления алгоритма, для которых 1+(*Q*3–1)*r*3*k*min(*Q*3*r*3, *N*).

**Распределение входных и выходных данных.** Соответственно распределению вычислений происходит распределение между процессами элементов матриц *A* и *B*; согласно заданию варианта 21 элементы матрицы *C* назначаются процессам динамически. Произвольному процессу Pr*p* распределяются столбцы матрицы *A* и строки матрицы *B* с номерами с1+*p* *r*3 по min((*p*+1)*r*3, *N*). Нулевой процесс Pr0 отправляет процессам Pr*p*, *p>*0, «свои» части, обозначим их *Ap* и *Bp*, матриц *A* и *B*. Результаты вычислений – вся матрицы *C* – окончательно подсчитывается в Pr*Q*3-1.

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

**Выделение массивов.** **Приватизация массивов.** *Ap* – матрицs размера *N×r*3и *Bp* – матрицы размера *r*3*×N* – приватизируются процессом Pr*p*, 0*p**Q*2–1. *CP* – матрица размера *r*1*×r*2 используется для трансляции. Результирующая матрица *C* формируется в процессе Pr*Q*3-1.

**Запись тайла с выделенными массивами.** Напомним вид тайла Tile1(*igl,jgl*):

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

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

enddo

enddo

Tile1(*igl,jgl*) с выделенными массивами:

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

*ip=i–igl* *r*1

*jp= j–jgl* *r*2

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

enddo

enddo

Напомним вид тайла Tile2(*igl,jgl,p*):

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

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

do *k* = 1+ *p* *r*3, min((p+1)*r*3, *N*)

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

enddo

enddo

enddo

Tile2(*igl,jgl,p*) с выделенными массивами:

do *i =* 1 + *igl r*1,min((*igl +*1)*r*1, *N*)

*ip=i–igl* *r*1

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

*jp= j–jgl* *r*2

do *k* = 1+ *p* *r*3, min((p+1)*r*3, *N*)

*kp= k–p* *r*3

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

enddo

enddo

enddo

**Оптимизация вычислений в тайлах.** Оптимизациювычислений **(**например, вычисление границ цикла следует выполнять вне цикла**)** рассматривать не будем. В контрольной работе этот пункт можно опустить.

**Структурирование коммуникаций.** Трансляцию данных опишем непосредственно при записи псевдокода.

**Псевдокод параллельного зернистого алгоритма.**

Для каждого процесса Pr*p*, 0*p**Q*3–1:

{if *p=*0 сформировать матрицы *Aq*, 0*q**Q*3–1,

send(Pr*q*; *Aq*; *N×r*3), 1*q**Q*3–1,

сформировать матрицы *Bq*, 0*q**Q*3–1,

send(Pr*q*; *Bq*; *r*3*×N*), 1*q**Q*3–1}

if *p>*0 receive(Pr0; *Ap*; *N×r*3), receive(Pr0; *Bp*; *r*3*×N*)

do *igl =* 0, *Q*1–1

do *jgl =* 0, *Q*2–1

if *p=*0 Tile1(*igl,jgl*)

if *p>*0 receive(Pr*p-*1; *CP*; *r*1*×r*2)

Tile2(*igl,jgl,p*)

if *p<Q*3–1 send(Pr*p+*1; *CP*; *r*1*×r*2)

if *p=Q*3–1 используя *CP* сформировать часть матрицы *C*

enddo(*j gl*)

enddo(*igl*)