ЭТАПЫ И ПРИМЕР 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 прямого хода метода Гаусса

(варианты 10 и 11 Лаб 4 (Гаусс) MPI)

Дан алгоритм прямого хода метода Гаусса с использованием расширенной матрицы:

do $k = 1, n-1$
  do $i = k+1, n$
    do $j = k+1, n+1$

$$a(i,j) = a(i,j) - \frac{a(i,k)}{a(k,k)}\, a(k,j)$$

    enddo
  enddo
enddo

Требуется разработать параллельный алгоритм согласно варианту 10 или варианту 11 (варианты отличаются только коммуникационными операциями).

Тайлинг: r_1=1 (цикл k глобальный не разбиваемый),

r_2= n–1 (цикл i локальный не разбиваемый),

Q_3 – параметр, $r_3 = \left\lceil \frac{n}{Q_3} \right\rceil$; s-координата: j;

коммуникации (вар. 10): бродкаст столбца, содержащего ведущий элемент.

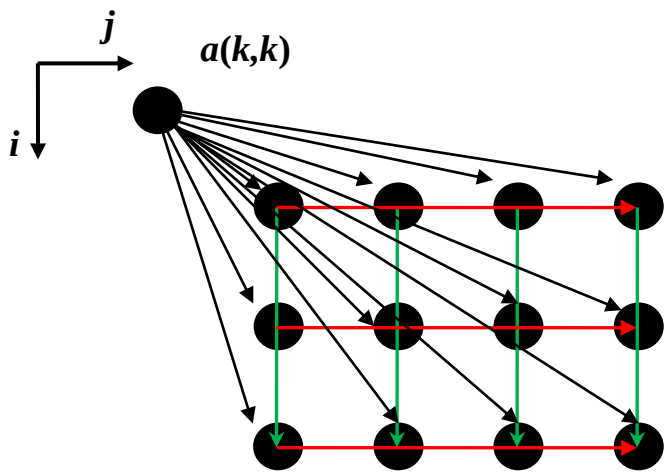
коммуникации (вар. 11): трансляция столбца, содержащего ведущий элемент.

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.

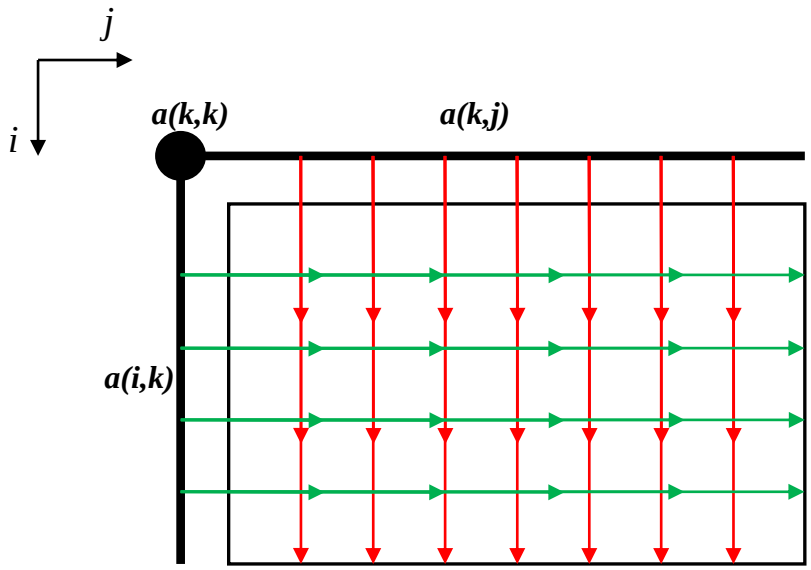
Рассмотрим этапы получения псевдокода.

Информационная структура алгоритма. В правой части оператора присваивания на каждом вхождении массива a происходит использование вычисленного на предыдущем шаге k элемента массива (при k=1 используются входные данные): $a(i,j)$ – использование прежнего значения обновляемого элемента, $a(i,k)$ – использование элемента столбца, содержащего ведущий элемент, $a(k,j)$ – использование элемента строки, содержащей ведущий элемент, $a(k,k)$ – использование ведущего элемента. Таким образом, вхождение массива a в левую часть оператора и вхождения в правую часть порождают зависимости.

Изобразим схематично зависимости операций k-го шага, порождаемые ведущим элементом $a(k,k)$, вычисленном на (k–1)-м шаге:



Изобразим теперь схематично зависимости операций k-го шага, порождаемые строками и столбцами (вычисленными на (k–1)-м шаге), содержащими ведущий элемент:



Укажем итерации, порождающие зависимости.

$S_1(k–1,i,j)\rightarrow S_1(k,i,j)$: данное $a(i,j)$, вычисленное на итерации $(k–1,i,j)$, является аргументом $a(i,j)$ для вычислений на текущей итерации (k,i,j);

$S_1(k–1,i,k)\rightarrow S_1(k,i,j)$: $a(i,k)$, вычисленное на итерации $(k–1,i,k)$, является аргументом для вычислений на текущей итерации (k,i,j);

$S_1(k–1,k,j)\rightarrow S_1(k,i,j)$: $a(k,j)$, вычисленное на итерации $(k–1,k,j)$, является аргументом для вычислений на текущей итерации (k,i,j);

$S_1(k–1,k,k)\rightarrow S_1(k,i,j)$: ведущий элемент $a(k,k)$, вычисленный на итерации $(k–1,k,k)$, является аргументом для вычислений на текущей итерации (k,i,j).

Обоснуем корректность тайлинга (для любого варианта). Достаточные условия допустимости тайлинга выполняются: для любой зависимости $S_\alpha(I)\rightarrow S_\beta(J)$ и любой координаты с одинаковым номером, её значение в J не меньше, чем в I (с учётом $j>k$, $i>k$); условия $\beta\geq\alpha$ выполняются (имеется только один оператор).

Тайлинг. Цикл с параметром k глобальный не разбиваемый, цикл с параметром i локальный не разбиваемый. Разобьем цикл с параметром j. Обозначим через Q_3 число итераций в глобальном цикле, а через r_3 (наибольшее) число итераций в локальном цикле; $r_3=\left\lceil \frac{n}{Q_3} \right\rceil$. Получим

do $k^{gl}= 1, n–1$
$k=k^{gl}$
  do $i= k+1, n$
    do $j^{gl} = 0, Q_3–1$
    do $j = \max(2+j^{gl}r_3, k+1), \min(1+(j^{gl}+1)r_3, n+1)$

$$a(i,j)=a(i,j) – \frac{a(i,k)}{a(k,k)}\ a(k,j)$$

    enddo
  enddo
  enddo

enddo

После перестановки циклов с параметрами i и j^{gl} получим

do k^{gl}= 1, n–1
  do j^{gl} = 0, Q_3–1
    // Начало тайла Tile(k^{gl},0,j^{gl})
    k=k^{gl}
    do i= k+1, n
      do j = max(2+$j^{gl}r_3$, k+1), min(1+(j^{gl}+1)r_3, n+1)

$$a(i,j)=a(i,j) - \frac{a(i,k)}{a(k,k)}\ a(k,j)$$

      enddo
    enddo
    // Конец тайла Tile(k^{gl},0,j^{gl})
  enddo
enddo

Таким образом,

do k^{gl}= 1, n–1
  do j^{gl} = 0, Q_3–1
    Tile(k^{gl},0,j^{gl})
  enddo
enddo

где Tile(k^{gl},0,j^{gl}) имеет вид

    k=k^{gl}
    do i= k+1, n
      do j = max(2+$j^{gl}r_3$, k+1), min(1+(j^{gl}+1)r_3, n+1)

$$a(i,j)=a(i,j) - \frac{a(i,k)}{a(k,k)}\ a(k,j)$$

      enddo
    enddo

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=j^{gl} – номер процесса):

Для каждого процесса Pr_p, $0 \le p \le Q_3-1$:

do k^{gl}= 1, n–1
  Tile(k^{gl},0,p)
enddo

Операции тайла Tile(k^{gl},0,p):

$k=k^{gl}$
do $i=k+1, n$
  do $j = \max(2+p\,r_3, k+1), \min(1+(p+1)r_3, n+1)$

$$a(i,j)=a(i,j) - \frac{a(i,k)}{a(k,k)}\ a(k,j)$$

  enddo
enddo

В нулевом процессе Pr_0 осуществляются все вычисления алгоритма, для которых $\max(2, k+1) \le j \le r_3+1$; в процессе Pr_1 осуществляются вычисления, для которых $\max(r_3+2, k+1) \le j \le 2r_3+1$. В процессе Pr_p, кроме (Q_3-1)-го процесса, осуществляются все вычисления алгоритма, для которых $\max(2+p\,r_3, k+1) \le j \le 1+(p+1)r_3$; в процессе с номером Q_3-1 осуществляются вычисления алгоритма, для которых $\max(2+(Q_3-1)r_3, k+1) \le j \le \min(1+Q_3 r_3, n+1)$.

Распределение входных и выходных данных. Соответственно распределению вычислений происходит распределение между процессами столбцов исходной расширенной матрицы. Процессу Pr_0 распределяются столбцы с номерами j, для которых $2 \le j \le r_3+1$; процессу Pr_1 распределяются столбцы j, $r_3+2 \le j \le 2r_3+1$. Произвольному процессу Pr_p распределяются столбцы j такие, что $2+p\,r_3 \le j \le \min(1+(p+1)r_3, n+1)$. Первый столбец распределяется нулевому процессу. Распределение входных данных осуществляет нулевой процесс. Распределение выходных данных такое же, как и распределение входных данных (элементы на главной диагонали и выше главной диагонали являются преобразованными, элементы ниже главной диагонали считаются нулевыми).

Общее представление о работе параллельного алгоритма и об обмене данными. Напомним в общих чертах действия k-го шага прямого хода (k=1,2,…,n–1). На k-м шаге осуществляется преобразование матрицы, приводящее к обнулению k-го столбца расширенной матрицы A ниже главной диагонали. Соответствующие операции над k-м столбцом в реальности не выполняются, выполняются операции со столбцами расширенной матрицы, начиная с $(k+1)$-го и заканчивая $(n+1)$-м (эти операции можно рассматривать как операции со строками, начиная с $(k+1)$-й и заканчивая n-й). Всего выполняется $n-1$ шагов. Для выполнения обратного хода потребуется «верхний треугольник» преобразованной матрицы A.

На k-м шаге обозначим через c столбец, содержащий ведущий элемент; это k-й столбец преобразуемой матрицы A. Процесс, хранящий столбец c, выполняет операции со строками, начиная с $(k+1)$-й и заканчивая n-й, своей части матрицы A; если c – последний столбец, приписанный процессу Pr_p, то $k=(p+1)r_3+1$, в цикле do $j=\max(2+p\,r_3, k+1), \min(1+(p+1)r_3, n+1)$ нижняя граница больше верхней, вычислений нет. Далее процесс пересылает (если он не последний по номеру) столбец c (функциональное значение имеют

только элементы с индексами от $k+1$ до n) всем последующим процессам (вар. 10) или следующему процессу (вар. 11). После этого процесс готов перейти к шагу $k+1$.

Любой последующий процесс: получает столбец c от процесса, хранящего столбец c, (вар. 10) или от предыдущего процесса (вар. 11), выполняет операции со строками в своей части матрицы A, начиная с $(k+1)$-й и заканчивая n-й. Затем, процесс пересылает (только вар. 11, при бродкасте пересылать не надо) столбец c (функциональное значение имеют только элементы с индексами от $k+1$ до n) следующему процессу. После этого процесс готов перейти к шагу $k+1$.

Замечание: пересылку столбца c следующему процессу можно производить и до выполнения операций со строками.

Результаты вычислений передаются нулевому процессу. Нулевой процесс формирует преобразованную расширенную матрицу (с нулевыми элементами ниже главной диагонали).

Выделение массивов. Приватизация массивов. Матрицу, составленную из столбцов матрицы A, назначенных p-му процессу, обозначим A_p, $0 \le p \le Q_3-1$; элементы матрицы A_p будем обозначать $ap(i,j)$. Столбцы матрицы A_0 – это столбцы матрицы A с номерами $1,\ldots,r_3+1$. Столбец с номером 0 (это первый столбец матрицы A) имеет только матрица A_0. Столбцы матрицы A_p, $1 \le p \le Q_3-2$, – это столбцы матрицы A с номерами $p\cdot r_3+2,\ldots,(p+1)r_3+1$. Столбцы матрицы A_p, $p=Q_3-1$, – это столбцы матрицы A с номерами $(Q_3-1)r_3+2,\ldots,n+1$. Таким образом, $ap(i,jp)=a(i,j)$, $1 \le jp \le r_3$, $p\,r_3+2 \le j \le (p+1)r_3+1$, $jp=j-p\cdot r_3-1$. Матрица A_0 имеет еще столбец с номером 0.

Массив A_p, $0 \le p \le Q_3-1$, приватизируется процессом Pr_p.

Найдем номер процесса p, хранящего k-й столбец матрицы A (этот столбец транслируется на k-м шаге).

Процесс с номером p хранит столбцы с номерами j $(2 \le j \le n+1)$ такими, что $p\cdot r_3+2 \le j \le (p+1)r_3+1$. Отсюда получаем $\frac{j-1}{r_3}-1 \le p \le \frac{j-2}{r_3}$. Так как p является целым числом, то $p=\left\lceil \frac{j-r_3-1}{r_3} \right\rceil = \left\lfloor \frac{j-2}{r_3} \right\rfloor$.

Таким образом, процесс с номером $p=\left\lfloor \frac{k-2}{r_3} \right\rfloor$ хранит k-й столбец матрицы A $(k>1)$; первый столбец $(k=1)$ хранит нулевой процесс. В матрице A_p этот столбец имеет номер $kp=k-p\cdot r_3-1$. Этот транслируемый столбец обозначен буквой c. Заметим, что если на k-м шаге k-й столбец матрицы A есть последний столбец матрицы A_p, $(kp=k-p\cdot r_3-1=r_3$, т.е. $k=(p+1)r_3+1)$ то $\mathrm{Tile}(k^{gl},0,p)$, $k=k^{gl}$, не порождает вычислительных операций, но порождает коммуникационную операцию пересылки столбца c.

Запись тайла с выделенными массивами. Операции тайла Tile(k^{gl},0,p) для процесса с номером $p=\left\lfloor \frac{k-2}{r_3} \right\rfloor$ (p=0, если k=1), $k=k^{gl}$ (этот процесс хранит ведущий элемент):

$k=k^{gl}$
$c(k)=ap(k,k-p\cdot r_3-1)$ // $c(k)$ – это $a(k,k)$ на шаге k
do $i=k+1, n$
  $c(i)=ap(i,k-p\cdot r_3-1)$ // $c(i)$ – это $a(i,k)$ на шаге k
  do $j = \max(2+p\cdot r_3, k+1), \min(1+(p+1)r_3, n+1)$
    $jp=j-p\cdot r_3-1$
    $ap(i,jp)=ap(i,jp) - \frac{c(i)}{c(k)}\cdot ap(k,jp)$
  enddo
enddo

Напомним Tile(k^{gl},0,p):

$k=k^{gl}$
do $i=k+1, n$
  do $j = \max(2+p\cdot r_3, k+1), \min(1+(p+1)r_3, n+1)$
    $a(i,j)=a(i,j) - \frac{a(i,k)}{a(k,k)}\ a(k,j)$
  enddo
enddo

Операции тайла Tile(k^{gl},0,p) для процесса с номером $p>\left\lfloor \frac{k-2}{r_3} \right\rfloor$ (p=0 при k=1), $k=k^{gl}$:

$k=k^{gl}$
do $i=k+1, n$
  do $j = \max(2+p\cdot r_3, k+1), \min(1+(p+1)r_3, n+1)$
    $jp=j-p\,r_3-1$
    $ap(i,jp)=ap(i,jp) - \frac{c(i)}{c(k)}\cdot ap(k,jp)$
  enddo
enddo

Оптимизация вычислений в тайлах. Деление $\frac{c(i)}{c(k)}$ можно производить вне цикла с параметром j. Операции тайла Tile(k^{gl},0,p) для процесса с номером $p=\left\lfloor \frac{k-2}{r_3} \right\rfloor$ (p=0, если k=1), $k=k^{gl}$:

$k=k^{gl}$
$c(k)=ap(k,k-p\cdot r_3-1)$
do $i=k+1, n$

$c(i)=ap(i,k-p\cdot r_3-1)$

$l=\dfrac{c(i)}{c(k)}$

do $j=\max(2+p\cdot r_3, k+1), \min(1+(p+1)r_3, n+1)$

$jp=j-p\,r_3-1$

$ap(i,jp)=ap(i,jp)-l\cdot ap(k,jp)$

enddo

enddo

Операции тайла Tile(k^{gl},0,p) для процесса с номером $p>\left\lfloor\dfrac{k-2}{r_3}\right\rfloor$, $k=k^{gl}$:

$k=k^{gl}$

do $i=k+1, n$

$l=\dfrac{c(i)}{c(k)}$

do $j=\max(2+p\cdot r_3, k+1), \min(1+(p+1)r_3, n+1)$

$jp=j-p\cdot r_3-1$

$ap(i,jp)=ap(i,jp)-l\cdot ap(k,jp)$

enddo

enddo

Отметим, что вычисление границ цикла лучше выполнять вне цикла.

Структурирование коммуникаций. Процесс с номером $p=\left\lfloor\dfrac{k-2}{r_3}\right\rfloor$ формирует на k-м шаге ($k>1$) столбец c; если $k=1$, то столбец c (столбец с номером 0 матрицы A_0) формирует нулевой процесс. Этот процесс Pr_p пересылает (если он не последний по номеру) столбец c (функциональное значение имеют только элементы с индексами от $k+1$ до n) процессам (вар. 10) Pr_q, $\left\lfloor\dfrac{k-2}{r_3}\right\rfloor+1\le q\le Q_3-1$, или (вар. 11) процессу с номером $\left\lfloor\dfrac{k-2}{r_3}\right\rfloor+1$.

Каждый процесс Pr_p, $\left\lfloor\dfrac{k-2}{r_3}\right\rfloor+1\le p\le Q_3-1$ ($1\le p\le Q_3-1$ при $k=1$) получает столбец c от процесса с номером $\left\lfloor\dfrac{k-2}{r_3}\right\rfloor$ или (вар. 11) от процесса Pr_{p-1}. После вычислений процесс Pr_p ($p\ne Q_3-1$) пересылает (только вар. 11, при бродкасте пересылать не надо) столбец c процессу Pr_{p+1}.

Бродкаст данных (вариант 10) или трансляцию данных (вариант 11) формально запишем далее в псевдокоде.

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 10.

Для каждого процесса Pr_p, $0 \le p \le Q_3-1$:
{if $p=0$ сформировать матрицы A_q, $0 \le q \le Q_3-1$,
send(Pr_q; A_q; $n \times r_3$), $1 \le q \le Q_3-1$}
if $p>0$ receive(Pr_0; A_p; $n \times r_3$)
// Итерацию $k^{gl}=1$ распишем отдельно:
{if $p=0$ сформировать столбец c // это столбец с номером 0 матрицы A_0
bsend(Pr_q, $1 \le q \le Q_3-1$; c; n)}
if $p>0$ breceive(Pr_0; c; n)
Tile(1,0,p)
do $k^{gl}= 2, n-1$
{if $p=\left\lfloor \frac{k^{gl}-2}{r_3} \right\rfloor$ сформировать столбец // это столбец с номером
$k-p \cdot r_3-1$, $k=k^{gl}$, матрицы A_p
if $p<Q_3-1$ bsend(Pr_q, $\left\lfloor \frac{k^{gl}-2}{r_3} \right\rfloor +1 \le q \le Q_3-1$; c; n)}
if $p>\left\lfloor \frac{k^{gl}-2}{r_3} \right\rfloor$ receive(Pr_q, $q=\left\lfloor \frac{k^{gl}-2}{r_3} \right\rfloor$; c; n)
Tile(k^{gl},0, p)
enddo
if $p>0$ send(Pr_0; A_p; $n \times r_3$)
{if $p=0$ receive(Pr_q; A_q; $n \times r_3$), $1 \le q \le Q_3-1$,
сформировать вычисленную матрицу A}

Псевдокод параллельного зернистого алгоритма для варианта 11.

Для каждого процесса Pr_p, $0 \le p \le Q_3-1$:
{if $p=0$ сформировать матрицы A_q, $0 \le q \le Q_3-1$,
send(Pr_q; A_q; $n \times r_3$), $1 \le q \le Q_3-1$}
if $p>0$ receive(Pr_0; A_p; $n \times r_3$)

// Итерацию $k^{gl}=1$ распишем отдельно:

if $p=0$ сформировать столбец c (столбец с номером 0 матрицы A_0)

if $p>0$ receive(Pr_{p-1}; c; n)

Tile(1,0,p)

if $p<Q_3-1$ send(Pr_{p+1}; c; n)

do $k^{gl}= 2, n-1$

{if $p=\left\lfloor \frac{k^{gl}-2}{r_3} \right\rfloor$ сформировать столбец c

(столбец с номером $k-p\cdot r_3-1$, $k=k^{gl}$, матрицы A_p)}

if $p>\left\lfloor \frac{k^{gl}-2}{r_3} \right\rfloor$ receive(Pr_{p-1}; c; n)

Tile(k^{gl},0, p)

if $p<Q_3-1$ send(Pr_{p+1}; c; n)

enddo

if $p>0$ send(Pr_0; A_p; $n\times r_3$)

{if $p=0$ receive(Pr_q; A_q; $n\times r_3$), $1\le q\le Q_3-1$,

сформировать вычисленную матрицу A}

Параллельный алгоритм прямого хода без избыточных вычислений границ пустых тайлов

Как уже отмечалось, на k-м шаге прямого хода обнуляется k-й столбец расширенной матрицы A ниже главной диагонали, выполняются операции со столбцами, начиная с $(k+1)$-го. Для фиксированного j^{gl} тайл Tile(k^{gl},0,j^{gl}) ($k^{gl}=k$) не является пустым, если он содержит вычисления, преобразующие хотя бы один столбец матрицы. Поэтому, для фиксированного j^{gl}, неравного Q_3-1, верхнее граничное значение k можно взять таким, что вычисления тайла преобразуют только столбец с номером $1+(j^{gl}+1)r_3$. Получим $1+(j^{gl}+1)r_3=k+1$, откуда $k=(j^{gl}+1)r_3$. Таким образом, псевдокод вычислительных операций параллельного алгоритма, не имеющего избыточных вычислений границ пустых тайлов, можно записать следующим образом ($p=j^{gl}$ – номер процесса):

Для каждого процесса Pr_p, $0\le p\le Q_3-1$:

do $k^{gl} = 1, \min((p+1)r_3, n-1)$

Tile(k^{gl},0,p)

enddo