



Л) – коэффициенты для вычисления новых корней. Рассмотрим начальный момент, когда в строке  $i-1$  вычисляется корень  $x_{i-1}^{k+1}$  (рис. 1,а). Для этого предположим, что каким-то образом ранее вычислены суммы, охватывающие отмеченный стрелками участок коэффициентов:  $S_{n,i-1} = \sum_{j=1}^{i-2} c_{i-1,j} \times x_j^{k+1}$  и  $S_{n,i-1} = \sum_{j=i}^m c_{i-1,j} \times x_j^k + d_{i-1}$ . Тогда  $x_{i-1}^{k+1} = S_{n,i-1} + S_{n,i-1}$ . Полученное значение этого корня через коммуникационную систему рассылается всем процессорам выше и ниже процессора строки  $i-1$ .

Теперь рассмотрим последовательность операций в процессоре строки  $i$ . В этой строке уже сформировано значение суммы  $s_{n,i} = \sum_{j=1}^{i-2} c_{i,j} \times x_j^{k+1}$  (здесь  $s$  – маленькое). Такие же суммы уже вычислены и во всех нижележащих процессорах для своих коэффициентов. Далее в строке  $i$  необходимо выполнить следующие две операции:  $S_{n,i} = S_{n,i} + c_{i,i-1} \times x_{i-1}^{k+1}$  и  $x_i^{k+1} = S_{n,i} + S_{n,i}$ . Эти операции на уровне строки  $i$  требуют времени  $t = t_1 + t_2 + t_3$ . Все операции на уровнях  $i+1, i+2$  и так далее до уровня  $m$  будут повторяться, поэтому полное время  $T^m$  параллельного решения СЛАУ методом Гаусса-Зейделя без учета обменов будет  $T^m = m \times t$ .

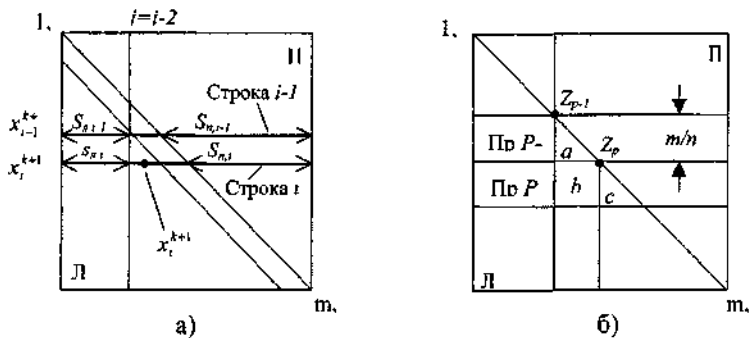


Рис. 1. Организация параллельных вычислений СЛАУ методом Гаусса-Зейделя, а)  $n=m$ ; б)  $n < m$ .

Как уже говорилось, значение  $x_{i-1}^{k+1}$  посылается также и в процессоры, лежащие выше строки  $i-1$ . Эти процессоры уже освободились в данной итерации от вычисления корней  $x_i^{k+1}$ , где  $l < i-1$ , поэтому они используются для вычисления в каждой  $l$ -ой строке частных сумм

$$S_{n,l}^{k+1} = \sum_{j=1}^{i-2} a_{l,j} \times x_j^{k+1}. \text{ Поэтому одновременно с вычислением последнего кор-}$$

ны  $x_m^{k+1}$  в правой части уже будут получены все суммы  $S_{n_i}$  для следующей итерации.

Приведенная выше ситуация для  $n=m$  использовалась для пояснения принципа распараллеливания. Реально  $n < m$  или даже  $n \ll m$ . Рассмотрим процесс параллельных вычислений для двух смежных процессоров  $p-1$  и  $p$  (рис. 1,б). Каждый из процессоров вычисляет  $m/n$  корней  $x_i^{k+1}$ .

Для каждой строки в процессоре  $p-1$  как и прежде (рис. 1,а) на основе вычисленных ранее корней  $x_i^{k+1}$  сформированы суммы  $S_j$  и  $S_n$ . Таким образом, для получения  $m/n$  корней в зоне процессора  $p-1$ , ему необходимо произвести вычисления в треугольнике  $a$ . Эти корни через коммуникационную систему будут переданы в процессор  $p$ , который вычислит очередные  $m/n$  корней. Назовем шагом вычислений процесс вычислений на участке одного процессора. Следовательно, сумма затрат времени на всех  $n$  шагах и составит время решения СЛАУ  $T_{1-2} = m \times t_m$ .

Пусть время одного шага  $t_m$  состоит из времени перехода от точки  $z_{p-1}$  до точки  $z_p$ . Чтобы процессор  $p$  мог начать вычисления в точке  $z_p$ , должны выполняться следующие действия:

- Ожидание вычисления первой части корней треугольника  $a$ :

$$t_{\text{ож}} = \left(\frac{m}{n \cdot k}\right)^2 \cdot \frac{1}{2} \cdot t$$

Здесь:  $t = t_1 + t_2$ ,  $k = 1, \dots, m/n$  – доля корней, передаваемых от процессора  $p-1$  в процессор  $p$  в первой посылке. При  $k=1$  передача производится только после вычисления всех корней процессором  $p-1$ ; в случае  $k=m/n$  передача осуществляется после вычисления каждого очередного корня.

- Передача первой части корней из процессора  $p-1$  в процессор  $p$  через коммуникационную систему:

$$t_{\text{пер}} = t_n + \frac{m}{n \cdot k} \cdot t_{\text{ср}}$$

Здесь  $t_n$  – латентность (начальная задержка) коммуникатора,  $t_{\text{ср}}$  – время передачи одного слова коммуникационной системой.

- Вычисление прямоугольника  $b$ :

$$t_{\text{св}} = \left(\frac{m}{n}\right)^2 \cdot t$$

Так как треугольник  $c$  равен треугольнику  $a$ , то для процессора  $p+1$  и всех последующих вычисления выполняются аналогично.

Ускорение вычисляется по формуле  $R = T_{\text{послед}} / T_{\text{пар}}$ , где  $T_{\text{послед}} = m^2 \cdot t$  - время последовательного, а  $T_{\text{пар}} = n \cdot (t_{\text{сер}} + t_{\text{пер}} + t_{\text{св}})$  - время параллельного вычисления СЛАУ методом Гаусса-Зейделя, поэтому получаем:

$$R = \frac{m^2 \cdot t}{n \cdot \left[ \left( \frac{m}{n \cdot k} \right)^2 \cdot \frac{1}{2} \cdot t + \left( t_s + \frac{m}{n \cdot k} \cdot t_{\text{св}} \right) + \left( \frac{m}{n} \right)^2 \cdot t \right]}$$

Это выражение в стандартной форме приобретает вид:

$$R = \frac{1}{\frac{1}{n} \cdot \left( 1 + \frac{1}{2 \cdot k^2} \right) \cdot t + n \cdot \frac{1}{m^2} \cdot \frac{t_s}{t} + \frac{1}{m \cdot k} \cdot \frac{t_{\text{св}}}{t}}$$

Первый член знаменателя этого выражения соответствует процессорным вычислениям, два следующих члена определяют вклад коммуникационной системы - ее задержку и пропускную способность.

Приведенное выражение использовалось для оценки ускорения:

- Для небольшого кластера типа Beowulf, построенного на базе высокоскоростной сети Fast Ethernet с числом процессоров до 20. Расчет производился для:  $m=10^3$ ,  $t=50$  нс (учитываются промахи кэша, вспомогательные операции и др.),  $t_s=150$  мкс,  $t_{\text{св}}=1$  мкс.
- Для большого кластера типа СКИФ с быстродействующей сетью SCI и характеристиками:  $m=10^4$ ,  $t=50$  нс,  $t_s=2$  мкс,  $t_{\text{св}}=20$  нс.

В обоих случаях для пересылки вычисленных корней использовалась известная операция коллективного обмена *bcast*, обеспечивающая одновременную передачу данных от одного процессора всем другим.

Соответствующие графики представлены на рис. 2 и рис.3.

#### Выводы

1. Предложенный алгоритм обеспечивает предельно возможный уровень распараллеливания для метода Гаусса-Зейделя
2. Для кластера на базе сети Fast Ethernet низкие результаты объясняются в первую очередь очень большой латентностью коммуникационной системы - 150 мкс. Результаты будут существенно лучше для матриц большего размера.
3. Для кластера на SCI и больших матриц достигается почти линейный рост ускорения.

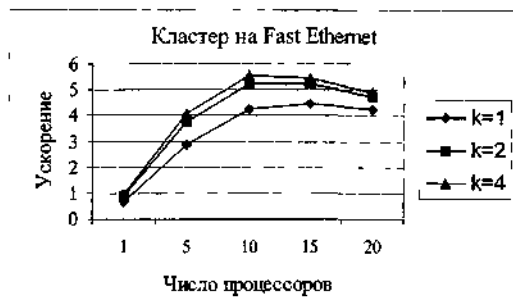


Рис.2. Ускорение для кластера типа Beowulf с сетью Fast Ethernet.



Рис.3. Ускорение для большого кластера с сетью SCI

### Литература

1. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. Пер. С англ. – М.: Мир, 1991. –367 с.
2. В.Д. Корнеев. Параллельное программирование в MPI. Новосибирск: Изд-во СО РАН, 2000. 213 с.
3. Г.И. Шпаковский, Н.В. Серикова. Программирование для многопроцессорных систем в стандарте MPI. Мн.: БГУ, 2002. 323 с.
4. Д.Л. Головашкин, О.Е. Горбунов. Параллельное решение СЛАУ методом Зейделя. Вестн. Самар. Гос. Техн. Ун-та. Сер. Физико-математические науки. 2004. №27