Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ (лекция 4)

А.С.Антонов

Вед. н.с. НИВЦ МГУ, к.ф.-м.н.

asa@parallel.ru

ВМК МГУ, 2014

Способы сдачи зачёта

1. *Зачёт в форме беседы с преподавателем. После 20 декабря, конкретные даты и аудитории будут объявлены позже. Вопросы к зачёту будут выложены позже.*
2. *Выполнение практического задания. Заявка – до 1 октября, выполнение – до 1 декабря.*
3. *Зачёт в форме электронного тестирования посредством системы http://sigma.parallel.ru. Около 15 декабря, конкретные дата и время будут объявлены позже.*

Вся информация на сайте http://parallel.ru/vmk14

Выполнение практического задания

- *Задание выполняется индивидуально.*
- *Требуется реализовать на суперкомпьютере «Ломоносов» параллельную версию некоторого алгоритма. Алгоритм выбирается на основе текущей научной работы.*
- *Заявка посылается на адрес asa@parallel.ru до 1 октября. Заявка должна содержать обоснование выбора алгоритма для параллельной реализации. На основе заявки оформляется доступ на суперкомпьютер «Ломоносов»*
- *Параллельная реализация должна быть выполнена с использованием технологии MPI. Для полученной реализации нужно провести исследование сильной масштабируемости. В случае плохой масштабируемости требуется определить основные мешающие факторы.*

Выполнение практического задания

- *Отчёт по заданию посылается на адрес asa@parallel.ru Срок представления отчёта – 1 декабря. Отчёт должен включать:*
  - *обоснование выбора алгоритма для параллельной реализации;*
  - *краткое математическое описание выбранного алгоритма;*
  - *описание выбранной схемы распараллеливания;*
  - *результаты запусков на суперкомпьютере «Ломоносов»;*
  - *график сильной масштабируемости с пояснениями;*
  - *объяснение полученных результатов.*
- *К отчёту должен прилагаться текст программы с краткими комментариями.*

Выполнение практического задания

- *Собранные результаты должны быть объяснимы и повторяемы.*
- *В отчёте должны содержаться сведения о программно-аппаратной среде, в которой получены результаты (компьютер, компилятор с использованными опциями оптимизации, библиотеки и т.д.)*
- *Каждый прогон программы с новыми параметрами рекомендуется выполнять несколько раз с последующим усреднением результата (для избавления от случайных выбросов).*
- *Количество процессов рекомендуется задавать в виде $p=2^n$, $n=0, 1, 2, \ldots, k$, где k определяется доступными ресурсами.*

Основные показатели эффективности и масштабируемости параллельных программ

Основные обозначения:

- p — число процессоров (процессорных ядер).
- T_1 — время работы программы на одном процессоре.
- T_p — время работы программы на p процессорах.

Основные показатели эффективности и масштабируемости параллельных программ

- *Ускорение* (*speedup*) $S = T_1/T_p$, где T_p — время исполнения распараллеленной программы на p процессорах, T_1 — время исполнения исходной программы.
- В идеальном случае (отсутствие накладных расходов на организацию параллелизма) получаем $S = p$ – *линейное ускорение*.
- *Суперлинейное ускорение* $S > p$.

Основные показатели эффективности и масштабируемости параллельных программ

- *Эффективность реализации* программы R_{max}/R_{peak} определяется как отношение реальной производительности R_{max} к пиковой производительности R_{peak}.
- Поскольку пиковая производительность недостижима на практике, эффективность реализации программы всегда меньше единицы.
- Чем ближе этот показатель к единице, тем лучше для пользователя, поскольку говорит о том, что более эффективно задействованы ресурсы компьютера.

Основные показатели эффективности и масштабируемости параллельных программ

- *Эффективность распараллеливания* E = S/p определяет среднюю долю времени выполнения параллельного алгоритма, в течение которого процессоры реально используются для решения задачи.
- Оценка качества распараллеливания предполагает получение наилучших (максимальных) значений ускорения и эффективности распараллеливания.
- Получение большого ускорения за счёт большого числа процессоров зачастую приводит к снижению эффективности распараллеливания.

Основные показатели эффективности и масштабируемости параллельных программ

- *Стоимость (cost)* вычислений $C = pT_p$.
- $T_0 = pT_p - T_1$ — суммарные *накладные расходы (total overhead)*. При увеличении p значение, как правило, возрастает.
- $T_p = (T_1 + T_0)/p$
- $S = T_1/T_p = pT_1/(T_1 + T_0)$

Основные показатели эффективности и масштабируемости параллельных программ

- $E = S/p = T_1/(T_1+T_0)=1/(1+T_0/T_1)$
- Если T_1 фиксировано, то при увеличении числа процессоров p эффективность распараллеливания E, как правило, уменьшается за счёт роста накладных расходов T_0.
- Если фиксировано число процессоров p, то эффективность распараллеливания E можно увеличить, увеличивая сложность решаемой задачи T_1.

Основные показатели эффективности и масштабируемости параллельных программ

- *Масштабируемость* (*scalability*) — способность системы увеличивать свою производительность при добавлении ресурсов (обычно аппаратных).
- Система называется *масштабируемой*, если она способна увеличивать производительность пропорционально дополнительным ресурсам.

Основные показатели эффективности и масштабируемости параллельных программ

- Масштабируемость можно оценить через отношение прироста производительности системы к приросту используемых ей ресурсов.
- Чем ближе это отношение к единице, тем масштабируемость лучше.

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость:

- Компьютера или его компонент (например, коммуникационной сети).
- Алгоритмов безотносительно к компьютеру.
- Параллельных программ относительно данного компьютера.

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость компьютера:

- *Вертикальная масштабируемость* (*масштабируемость вглубь, scale up*) – возможность замены платформы, в которой функционирует система, на новую, обладающую большей производительностью.
- *Горизонтальная масштабируемость* (*масштабируемость вширь, scale out*) – возможность увеличения производительности системы за счет добавления дополнительных программных или аппаратных средств.

Основные показатели эффективности и масштабируемости параллельных программ

- *Вычислительная сложность задачи* W – количество основных вычислительных шагов лучшего последовательного алгоритма, необходимых для решения задачи на одном процессоре.
- W является некоторой функцией от размера входных данных.
- Если для простоты предположить, что каждый основной вычислительный шаг выполняется за единицу времени, то получим $W = T_1$.

Основные показатели эффективности и масштабируемости параллельных программ

Примеры:

- Для сложения N чисел:

$W = N-1$

- Для скалярного произведения векторов:

$W = 2N-1$

- Для перемножения матриц:

$W = N^2(2N-1)$

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельной программы определяется относительно конкретного компьютера и показывает, как изменяются динамические характеристики данной программы при использовании бóльших вычислительных ресурсов.

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельных программ:

- *Сильная масштабируемость* (*strong scaling*) — зависимость производительности R от количества процессоров p при фиксированной вычислительной сложности задачи (W = const).

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельных программ:

- *Масштабируемость вширь (wide scaling)* – зависимость производительности R от вычислительной сложности задачи W при фиксированном числе процессоров (p = const).

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельных программ:

- *Слабая масштабируемость* (*weak scaling*) — зависимость производительности R от количества процессоров p при фиксированной вычислительной сложности задачи в пересчёте на один процессор (W/p = const).

Основные показатели эффективности и масштабируемости параллельных программ

- Нужна метрика масштабируемости.
- Для многих задач при увеличении вычислительной сложности задачи W (а, следовательно, и времени T_1) эффективность распараллеливания E растёт.
- Если при одновременном увеличении числа процессоров p и вычислительной сложности задачи W эффективность распараллеливания E остаётся прежней, данную задачу на данном компьютере можно считать *масштабируемой*.

Основные показатели эффективности и масштабируемости параллельных программ

Функция изоэффективности **(***isoefficiency function***)**:

- Определим, в какой степени должна увеличиваться вычислительная сложность задачи W в зависимости от числа процессоров p, чтобы эффективность распараллеливания E оставалась постоянной. Эта характеристика будет показывать масштабируемость конкретной вычислительной системы.
- Чем меньше необходимая степень роста W для поддержания нужного уровня эффективности распараллеливания, тем более масштабируемой является система.

Основные показатели эффективности и масштабируемости параллельных программ

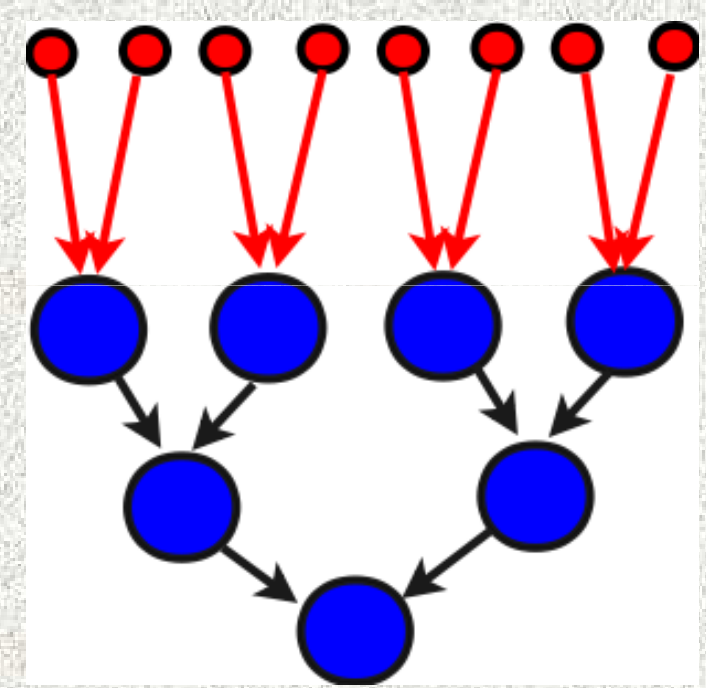
Функция изоэффективности **(***isoefficiency function***)**:

- $E = 1/(1+T_0/T_1) = 1/(1+T_0/W)$.
- $W = E/(1-E)*T_0 = K*T_0$, где $K = E/(1-E)$
- Получившаяся зависимость размера задачи, необходимой для достижения заданной эффективности распараллеливания, от числа процессоров – *функция изоэффективности (isoefficiency function)*.

Основные показатели эффективности и масштабируемости параллельных программ

Пример:

- Суммирование методом сдваивания (каскадная схема).



Основные показатели эффективности и масштабируемости параллельных программ

Пример:

- Пусть для сложения n чисел используется p процессоров
- $W = T_1 \approx n$
- $T_p \approx n/p + 2\log_2 p$
- $S = T_1/T_p = n/(n/p+2\log_2 p) = p/(1+2p\log_2 p/n)$
- $E = S/p = 1/(1+2p\log_2 p/n)$

Основные показатели эффективности и масштабируемости параллельных программ

Пример:

- $C = pT_p = p(n/p+2\log_2 p) = n+2p\log_2 p$
- $T_0 = pT_p - T_1 = 2p\log_2 p$
- $W = KT_0 = 2Kp\log_2 p = \Theta(p\log_2 p)$
- При увеличении числа процессоров от p до p' для поддержания постоянной эффективности распараллеливания E необходимо увеличить размер задачи в $(p'\log_2 p')/(p\log_2 p)$ раз.

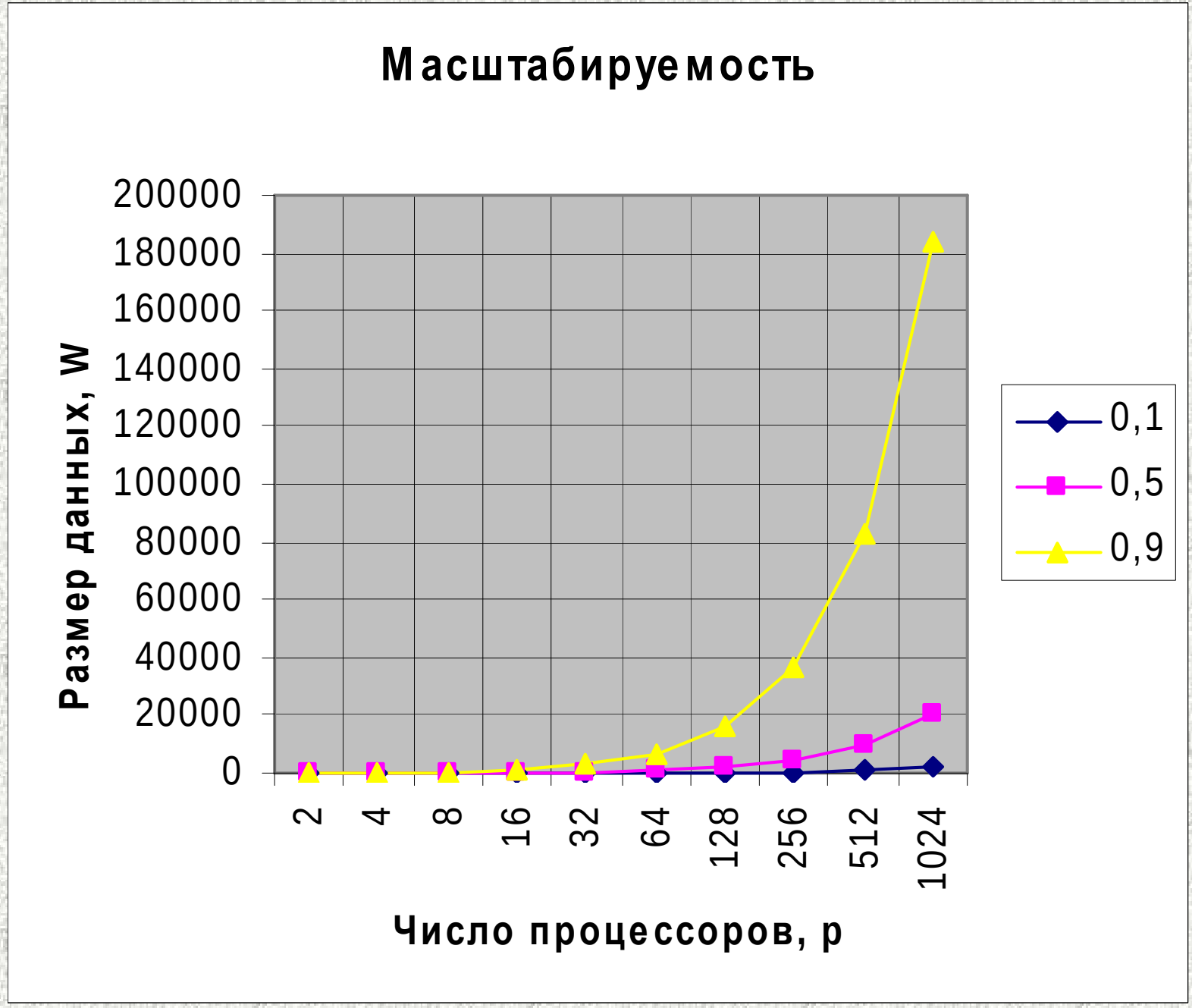
Основные показатели эффективности и масштабируемости параллельных программ

Пример:

Пусть E = 0.5, тогда K = E/(1-E) = 1.

- Пусть p = 16, тогда $W = 2Kp\log_2 p = 128$.
- Пусть p = 64, тогда $W = 2Kp\log_2 p = 768$.
- Пусть p = 1024, тогда $W = 2Kp\log_2 p = 20480$.

Основные показатели эффективности и масштабируемости параллельных программ

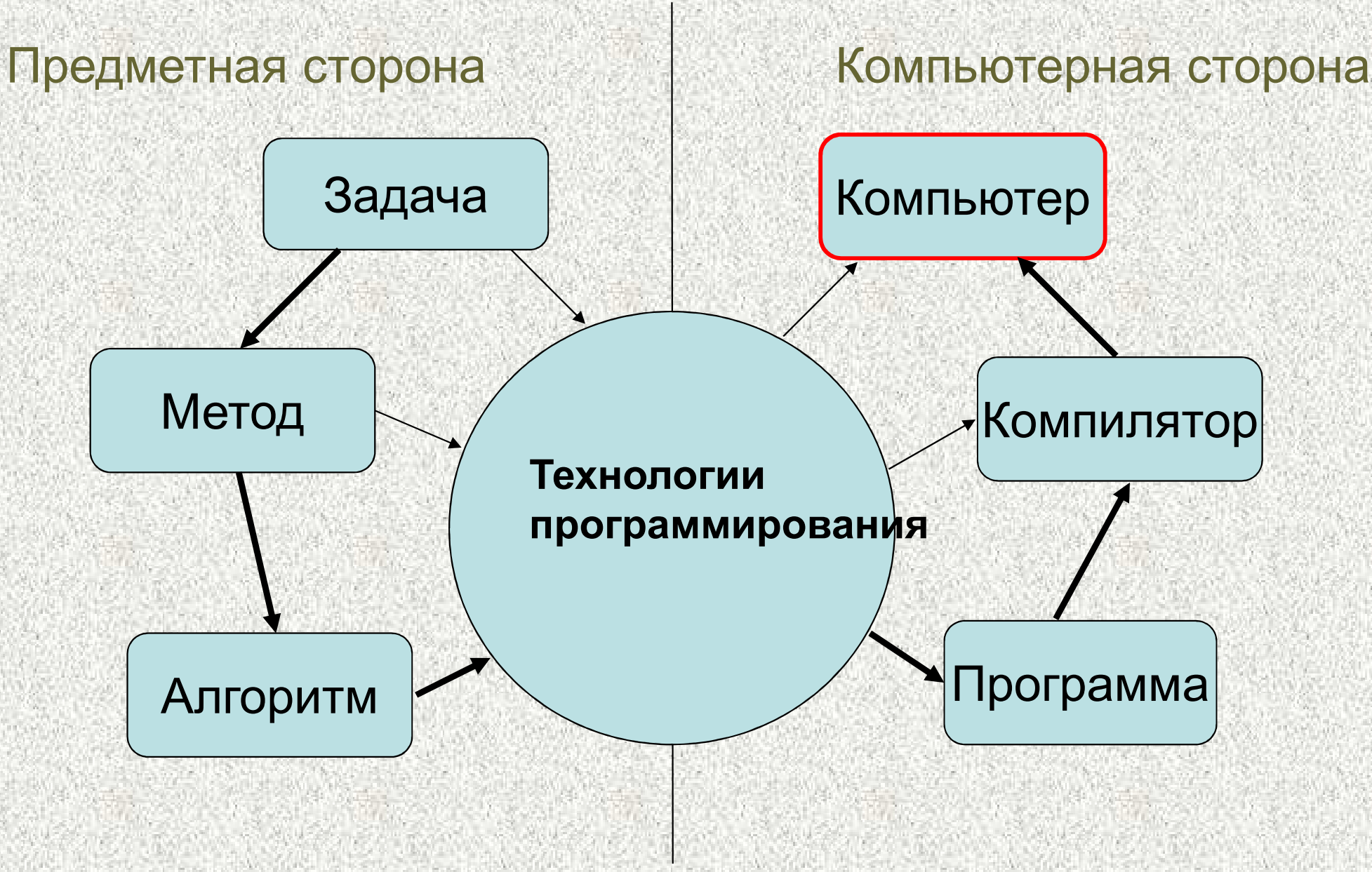


Основные показатели эффективности и масштабируемости параллельных программ

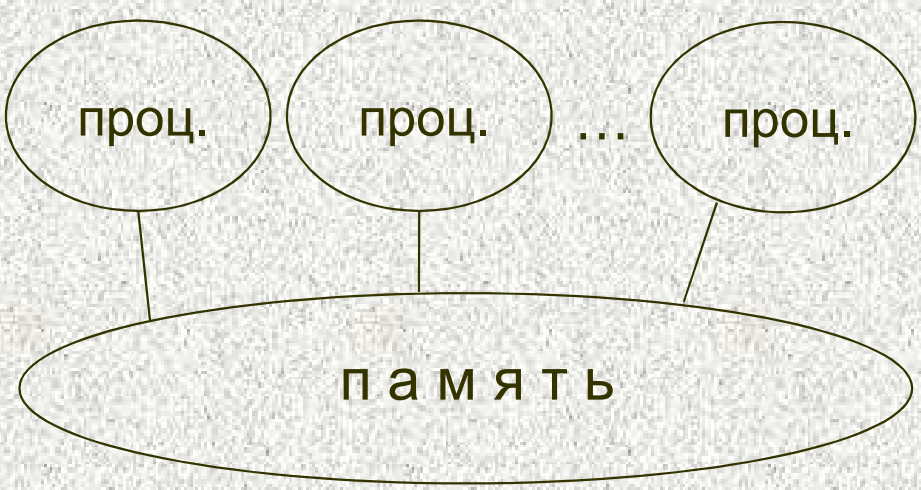
Основные помехи масштабируемости параллельных программ:

- Закон Амдала (последовательные части программы).
- Накладные расходы на коммуникации (латентность, пропускная способность).
- Неравномерность загрузки (load balancing) процессоров.
- Предел декомпозиции данных.

Решение задачи на компьютере



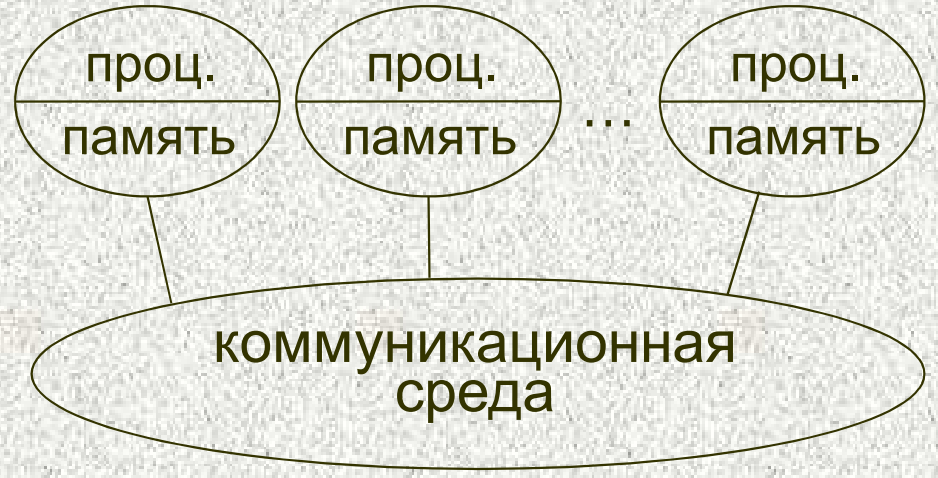
Компьютеры с общей памятью



SMP-компьютеры, два варианта расшифровки:
Shared Memory Processors
Symmetric MultiProcessor

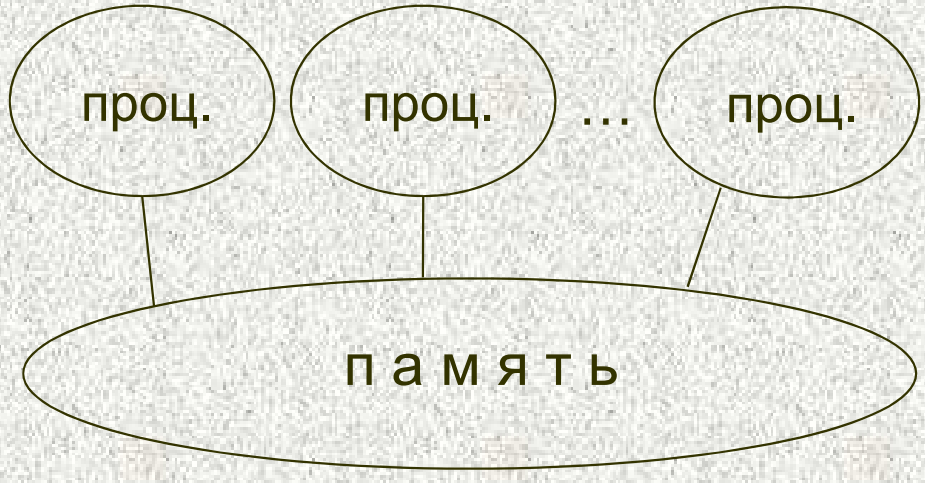
В SMP-компьютерах все, кроме процессоров, в одном экземпляре: образ операционной системы, память, подсистема ввода-вывода…

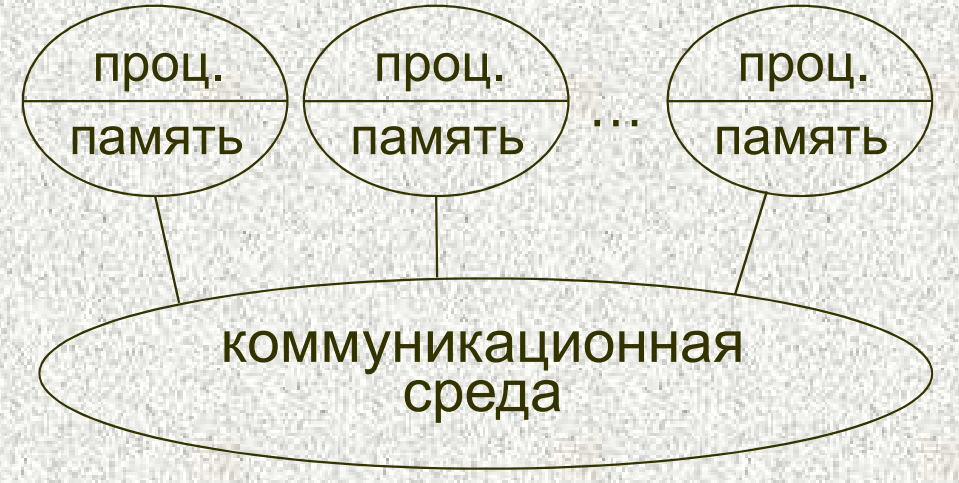
Компьютеры с распределенной памятью



Компьютеры с распределенной памятью состоят из вычислительных узлов, каждый из которых является полноценным компьютером со своей памятью, ОС, устройствами ввода-вывода и т.п., взаимодействующих друг с другом через коммуникационную среду.

Компьютеры с общей и распределенной памятью





+ относительная простота параллельного программирования,
– сложность увеличения числа процессоров (роста производительности)

– сложность параллельного программирования,
+ относительная простота увеличения числа процессоров (роста производительности)

Высокопроизводительные компьютеры и две задачи параллельных вычислений

Две задачи параллельных вычислений:

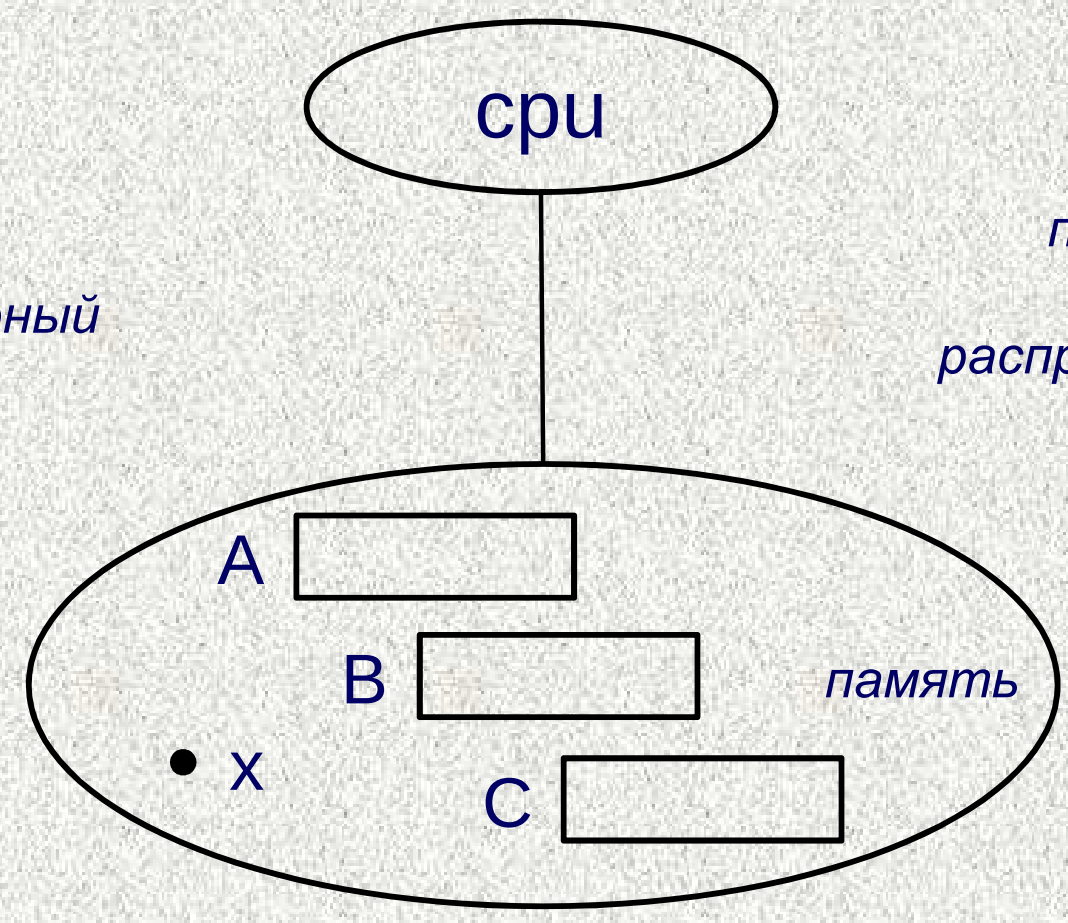
- *построение вычислительных систем с максимальной производительностью:*
  - *это компьютеры с распределенной памятью*

- *эффективное программирование параллельных вычислительных систем:*
  - *это компьютеры с общей памятью.*

Почему компьютеры с распределенной памятью программировать сложнее, чем компьютеры с общей памятью?

Программирование на общей и распределенной памяти

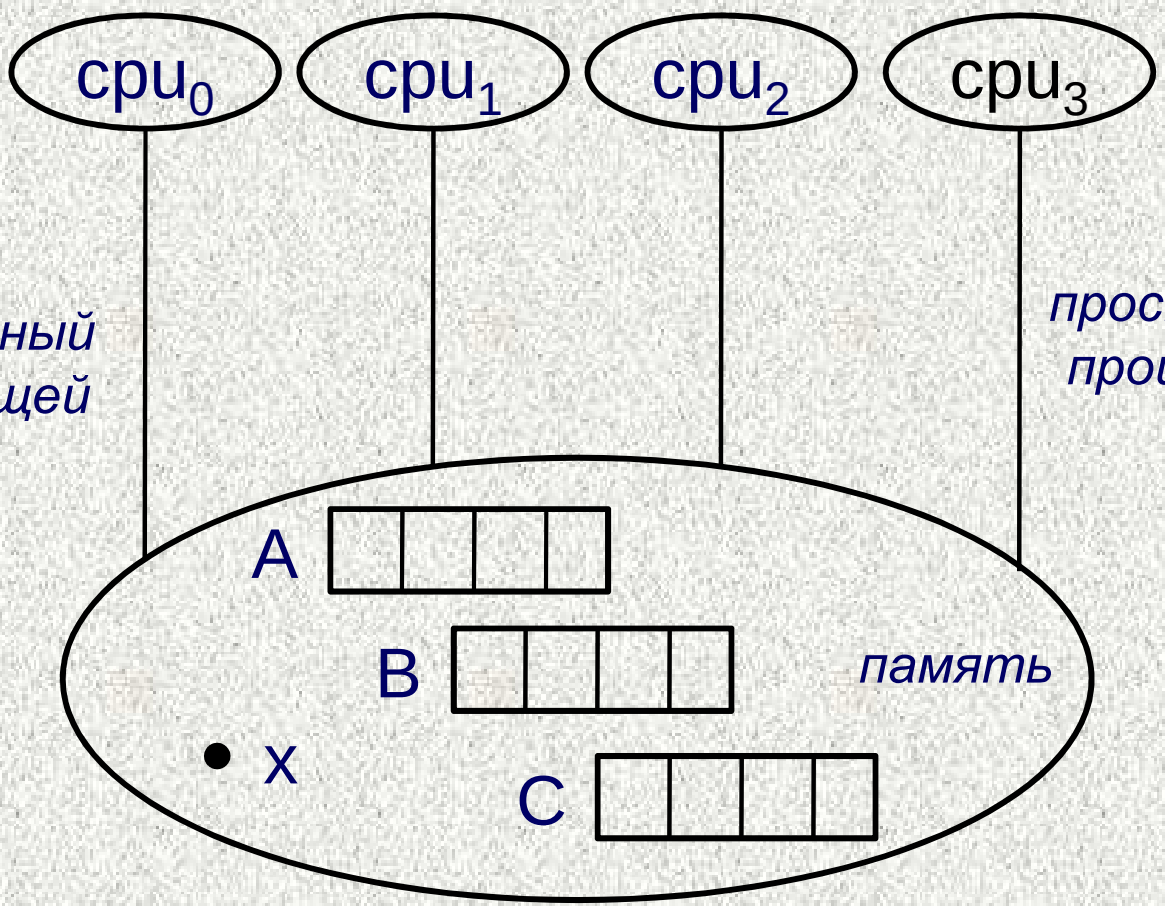
Однопроцессорный компьютер



Одно адресное пространство, один процессор, с распределением работы проблем нет.

$$A_i = B_i + C_i * x, \quad i = 1, \ldots, n$$

Программирование на общей и распределенной памяти

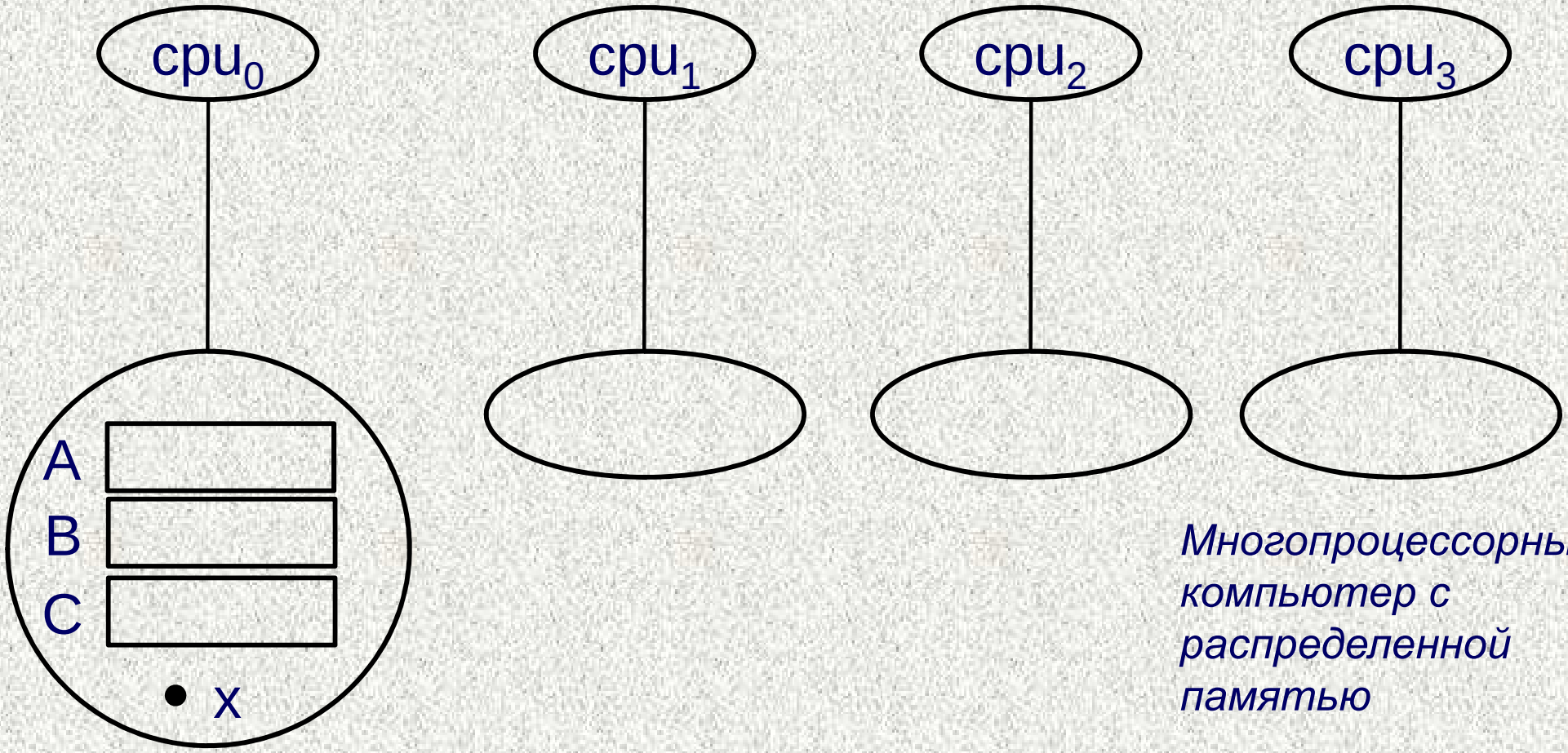


Многопроцессорный компьютер с общей памятью

Единое адресное пространство, много процессоров, каждый выполняет свою часть работы.

$$A_i = B_i + C_i * x, \quad i = 1, \ldots, n$$

Программирование на общей и распределенной памяти

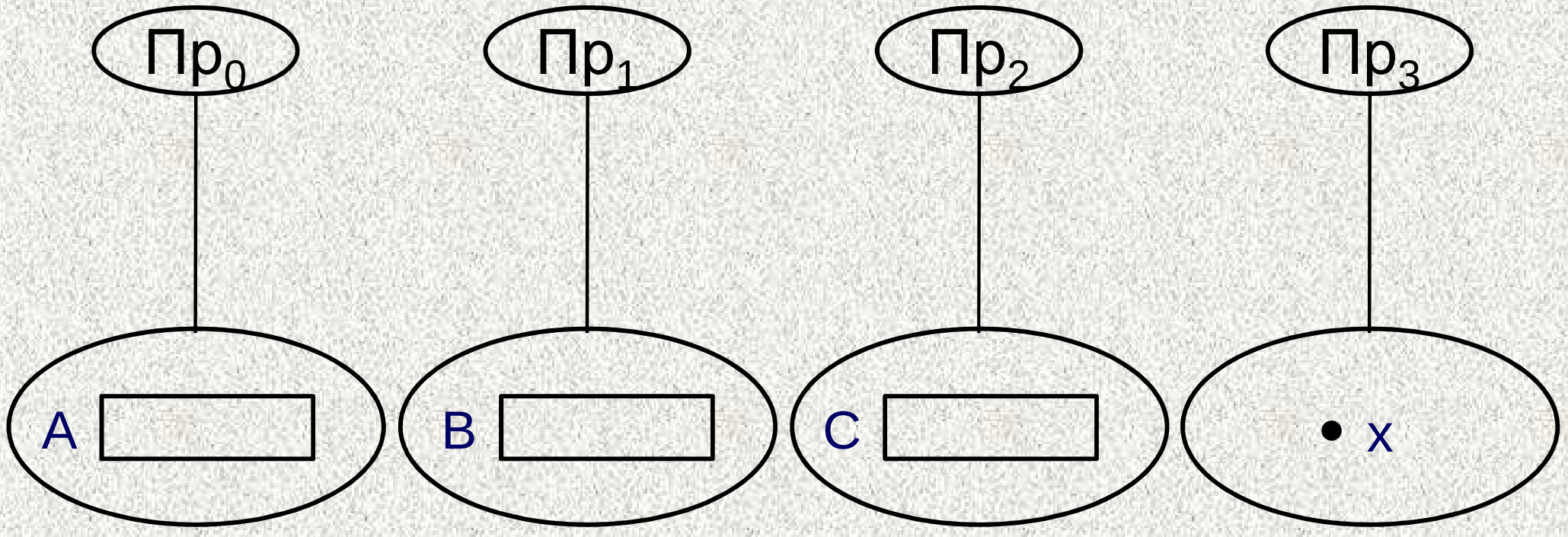


Многопроцессорный компьютер с распределенной памятью

Различные адресные пространства, плохое распределение данных, низкая локальность данных, много пересылок, низкая производительность.

$$A_i = B_i + C_i * x, \quad i = 1, \ldots, n$$

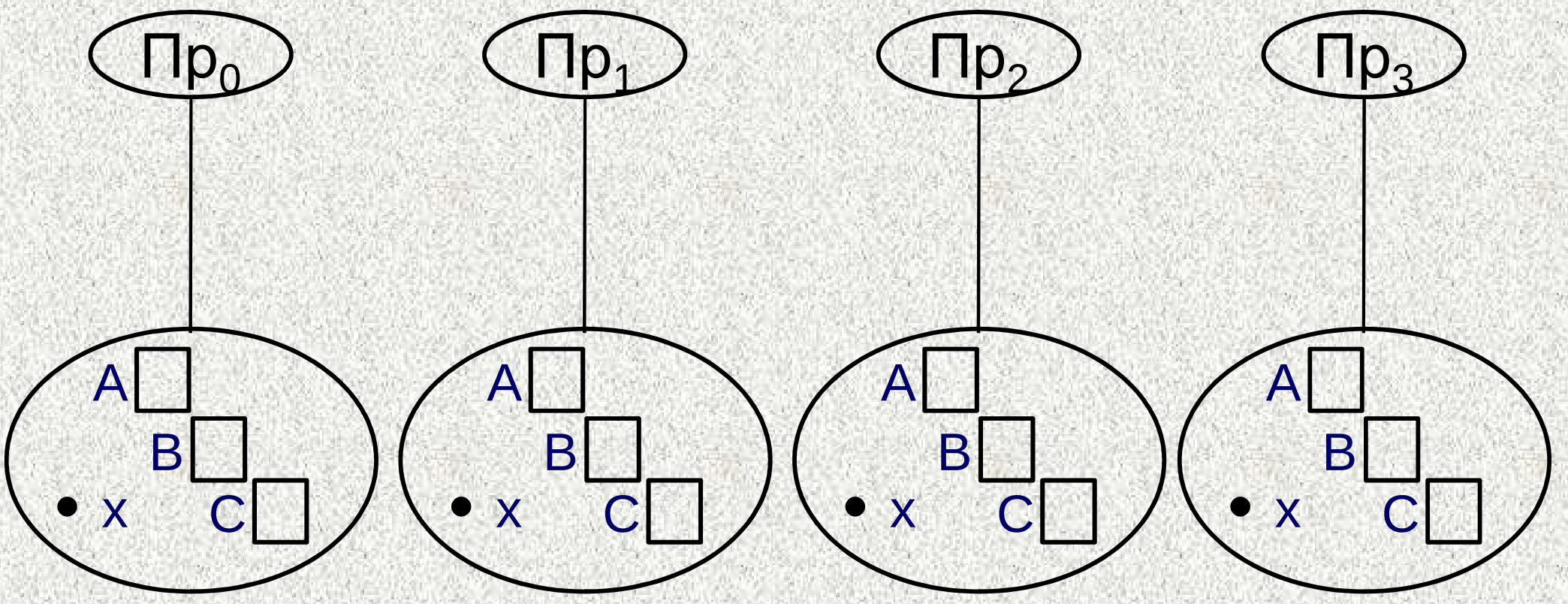
Программирование на общей и распределенной памяти



Различные адресные пространства, плохое распределение данных, низкая локальность данных, много пересылок, низкая производительность.

$$A_i = B_i + C_i * x, \quad i = 1, \ldots, n$$

Программирование на общей и распределенной памяти



Различные адресные пространства, хорошее распределение данных, высокая локальность данных, мало пересылок, высокая производительность.

$$A_i = B_i + C_i * x, \quad i = 1, \ldots, n$$

Программирование на общей и распределенной памяти

Последовательный алгоритм:

```
N=1000;
for(i=0; i<N; ++i)
   A[i]=B[i]+C[i]*x;
for(i=0; i<N; ++i)
   A[i]=(A[i]+C[i])*B[N-i-1];
```

На последовательном компьютере все данные (массивы A, B, C и скаляр x) лежат в общей памяти, различий в классах переменных нет.

Программирование на общей и распределенной памяти

Оба цикла являются параллельными, их итерации могут быть распределены между процессорами. Распределение вычислений обязательно должно быть согласовано с распределением данных!

На параллельном компьютере с общей памятью распределение данных не требуется (память общая), нужно распределить операции (итерации циклов). Возможных распределений очень много.

Программирование на общей и распределенной памяти

Первый вариант (блочное распределение итераций):

```
N=1000;
n_pr=4;
size=N/n_pr;
ibegin=proc_id*size;
iend=ibegin+size;
for(i=ibegin; i<iend; ++i)
   A[i]=B[i]+C[i]*x;
for(i=ibegin; i<iend; ++i)
   A[i]=(A[i]+C[i])*B[N-i-1];
```

Возникает понятие классов переменных. В данном случае переменные proc_id, i, ibegin, iend должны быть локальными (своя копия на каждом процессоре), а переменные A, B, C, x должны быть распределёнными (одна копия данных для всех процессоров).

Программирование на общей и распределенной памяти

Другой вариант (циклическое распределение итераций):

```
N=1000;
n_pr=4;
for(i=proc_id; i<N; i+=n_pr)
   A[i]=B[i]+C[i]*x;
for(i=proc_id; i<N; i+=n_pr)
   A[i]=(A[i]+C[i])*B[N-i-1];
```

Переменные proc_id, i должны быть локальными (своя копия на каждом процессоре), а переменные A, B, C, x должны быть распределёнными (одна копия данных для всех процессоров).

Программирование на общей и распределенной памяти

На параллельном компьютере с распределённой памятью помимо распределения операций требуется также распределение данных по локальным модулям памяти разных процессоров и организация обменов данными.

```
N=1000;
n_pr=4;
N1=N/n_pr;
<обмен 1> // рассылка частей массивов A, B, C и скаляра x
for(i=0; i<N1; ++i)
   A[i]=B[i]+C[i]*x;
<обмен 2> // обмен частями B: процессы 0<->3 и 1<->2
for(i=0; i<N1; ++i)
   A[i]=(A[i]+C[i])*B[N-i-1];
<обмен 3> // пересылка результирующего массива A
```

Для эффективного программирования компьютеров с распределенной памятью необходимо:

1. *Найти в программе ресурс параллелизма.*
2. *Распределить данные по модулям памяти вычислительных узлов.*
3. *Согласовать распределение данных с параллелизмом вычислений.*
4. *Организовать необходимые пересылки данных.*

Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ
(лекция 4)

А.С.Антонов

Вед. н.с. НИВЦ МГУ, к.ф.-м.н.

asa@parallel.ru

ВМК МГУ, 2013