Параллельная обработка данных.

Воеводин Владимир Валентинович.

Top 500. org

также рекомендуется посетить

playboy.com

penthouse.com

hustler.com

badgirls blog.com

Лекция 1

05.09

1 IBM Blue Gene, > 131 000 ядер. 2^{17}
PowerPC 440

> 280 Tflops

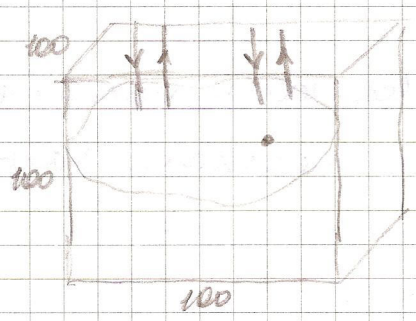
4 SGI, Altix, 10160 ядер
Intel Itanium

> 50 Tflops ОП 20ТБ, Хард 440 ТБ
Ленточные архивы 10 ПБ

10. NEC Earth Simulator 5120 ядер
NEC SX-6

> 35 Tflops ОП 10 ТБ, диски ~250ТБ
Лент. ~1,5 ПБ

Нейрс. компания



10^6 точек сетки

5-20 ф-ий

200-1000 операций на каждую ф-ию

100 - 1000 шагов по времени

$10^6 \cdot 10 \cdot 5 \cdot 10^2 \cdot 5 \cdot 10^2 = 2{,}5 \cdot 10^{12}$ операций

1999 - начало проекта

1 чип ~ 1 Gflops

1 кристалл ~ 32 процессора

1 плата ~ 64 кристалла

1 стойка ~ 8 плат

проект ~ 64 стойки → 1 Pflops

1948г EDSAC 2000 Hewlett Packard, Бирмингам

такт $2 \cdot 10^{-6}$ ~1500раз $1,3 \cdot 10^{-9}$

производительность 100 оп/с ~$2 \cdot 10^9$раз $192 \cdot 10^9$ оп/с

Изменение тактовой частоты происходит за счет изменения элементной базы

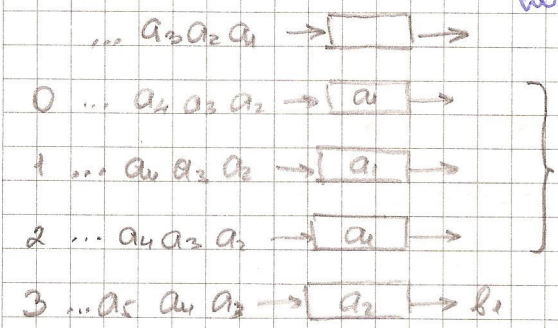
Основы ПОД:

- параллельность
- конвейерность

ФУ – функциональное устройство

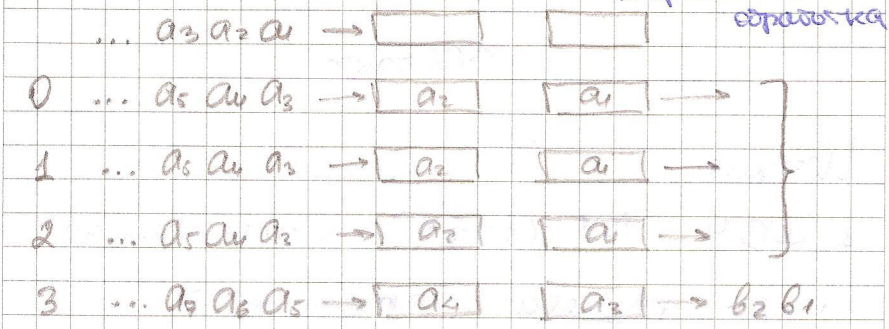
300 аргументов, ФУ $a \to b$ 3 такта

последовательная обработка

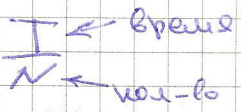


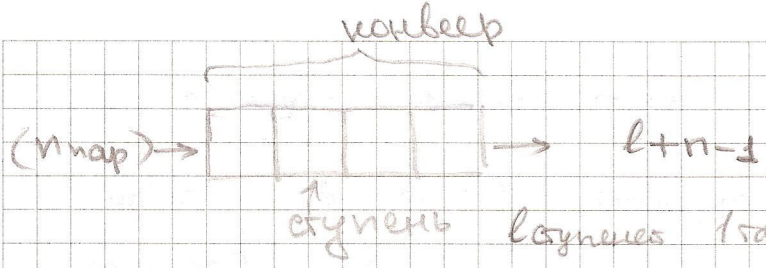
900 тактов

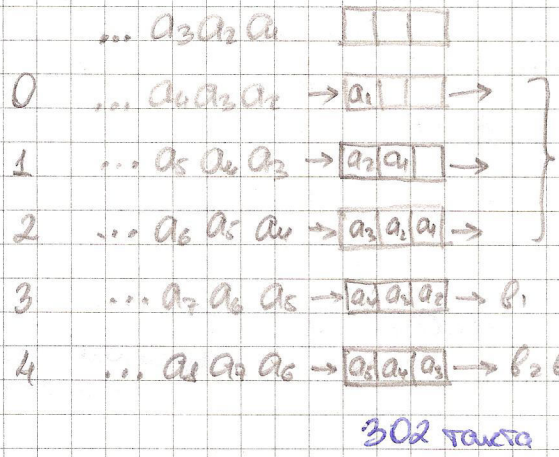
параллельная обработка



450 тактов







302 такта

История:

1 IBM 701 (1953) - параллельная память, разрядно-параллельная арифметика.

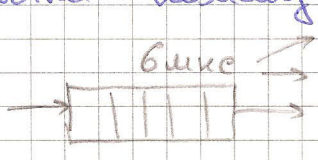
IBM 704 (1955)

2 IBM 709 (1958) - независимые процессоры ввода/вывода.

3 IBM STRETCH (1961) -

Одновременная выборка команд +
+ расслоение памяти

4 ATLAS (1963г.) - конвейерная обработка команд



① выбор команды
② вычисление адреса операнда
③ выборка операнда
④ выполнение операции

5. CDC-6600 (1964г.) - независимые функциональные устройства 10 МГц
такт - 100 нс, 23 М оп/с

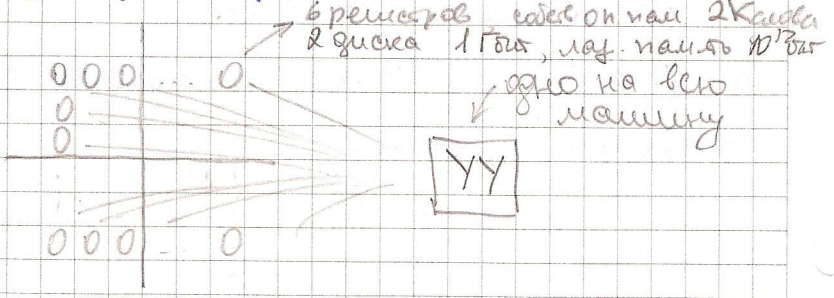
6. CDC-7600 (1969) 8 МГц
такт - 27,5 нс, 10-15 М оп/с

- пиковая производительность - теоретическая
- реальная производительность

12.09 Лекция 2.

7. ILLIAC-IV (1967–1974)

матрица процессоров 16×16:



УУ - устройство управления

Синхронно работающая матрица процессоров

время такта 40 нс

Производительность 1 Gflops } задумывалось

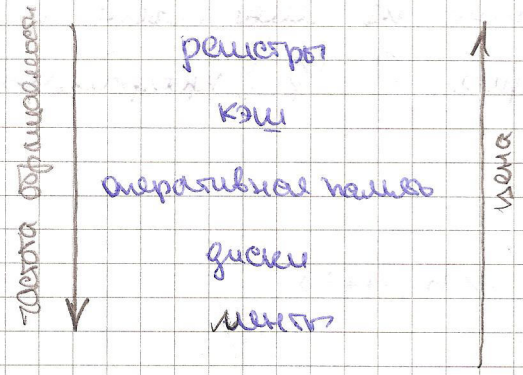
1967 - начало

1971 - изготовили I квадрант

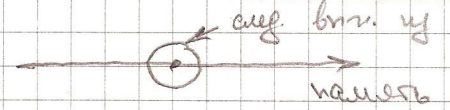
1974 - сдали в эксплуатацию I кв.

Получилось: 80 нс, 50 Mflops

Иерархия памяти.



- Локальность внимания



- локальность использования данных

Закон Амдала: p - процессоров

f - доля последовательных операций,

$0 \le f \le 1$

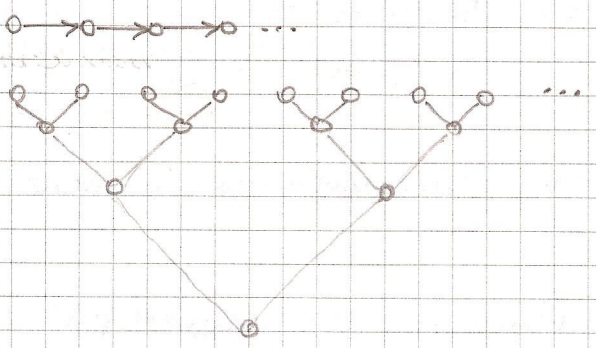
$$\frac{T_1}{T_p} = S \le \frac{1}{f + \frac{1-f}{p}}$$

$f = 0 \Rightarrow S \le p$

$f = 1 \Rightarrow S \le 1$

Чтобы ускорить выполнение программы в p раз, нужно не менее чем в p раз ускорить $(1 - 1/p)$ программы

①
```
s = 0;
for (i = 0; i < n; ++i)
  s += A[i];
```

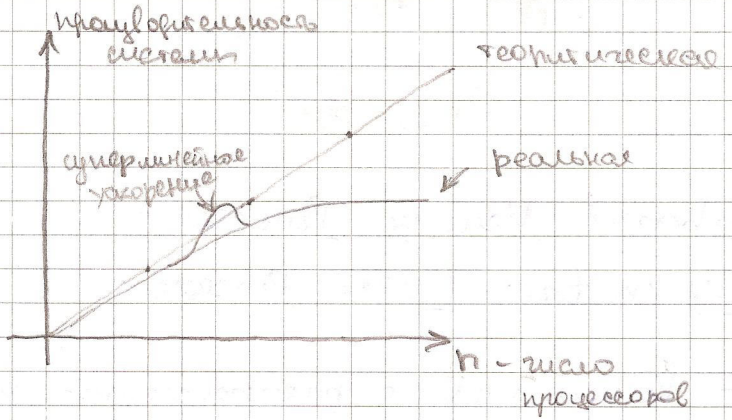


Эти два алгоритма разные (например ошибки округления)

Выбор баланс вычислительной нагрузки в зависимости от производительности вычислительных узлов.

Время тратится также на передачу данных.

(2)



При превышении реальной производительности теоретической → нет КЭШ памяти

Этапы решения задачи на парал. ВС:

Задача: → Метод → алгоритм

технология параллельного программирования

программа

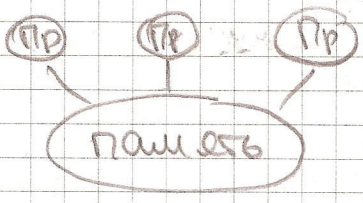
компилятор

компьютер

Лекция 3 26.09

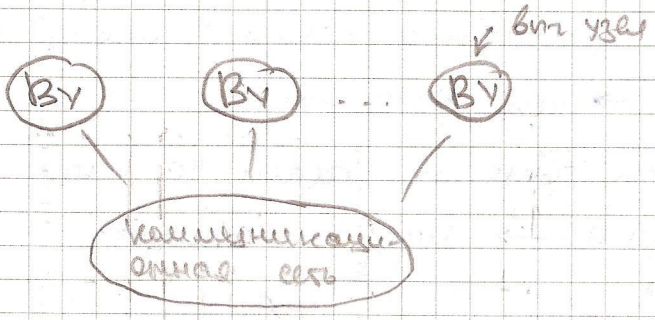
Архитектура параллельных ВС

- Компьютеры с общей памятью

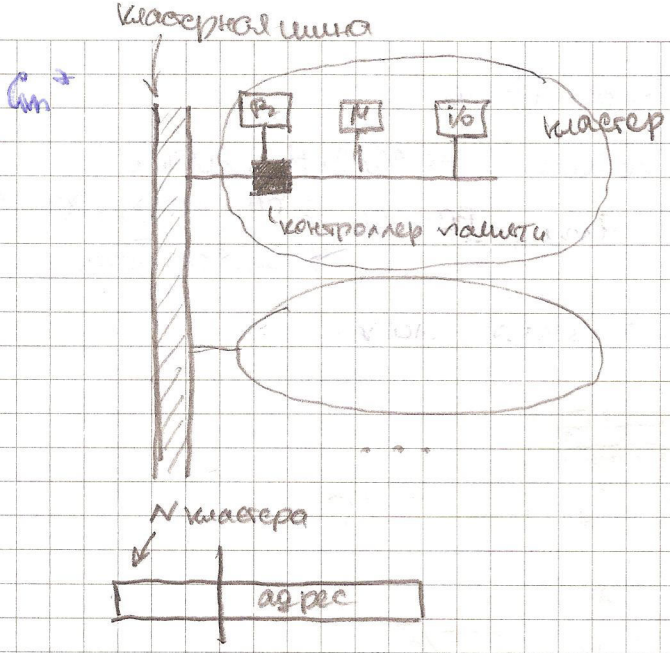


Shared Memory Processors
Symmetric Multi Processors

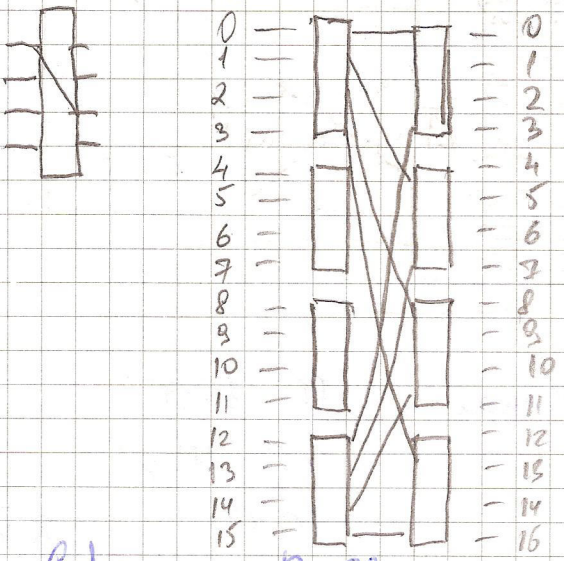
- Компьютеры с распределенной памятью



- NUMA (конец 70-х (н*))
Non Uniform Memory Access



- BBN Butterfly



Cache Coherence Problem

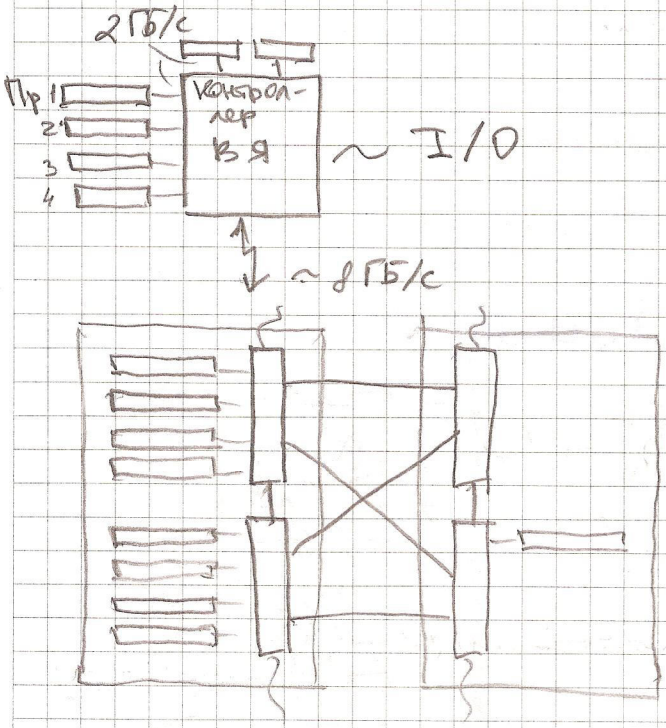
– HP Super Dome

по технологии ccNUMA, 2000г

до 64 проц, IA-64 → PA 8500, 8700, 8900

→ Intel Itanium 2

Вычислительная ячейка



до 64 проц

PA 8700

750 MHz

10 ч у

4 оп/такт

} 3 Гфлопс

=> 192 Гфлопса

Кэш 2,25 МБ — 1,5 МБ данные
→ 0,75 I команд

Причины снижения производительности:

1 Закон Амдала

2 cc-NUMA

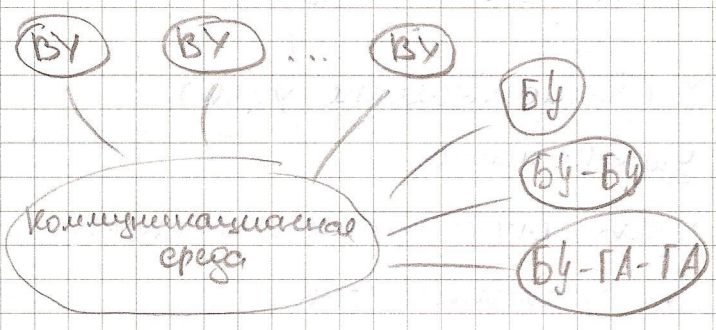
3 cc-NUMA

4 Распределение вычислительной нагрузки

5 Производительность процессора

и т.д

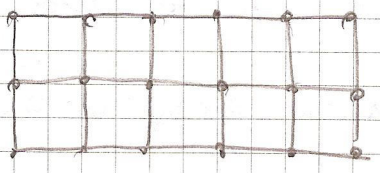
Лекция 4 03.10



Появился в начале 90-х годов

Это системы с массовым параллелизмом или массово-параллельные системы

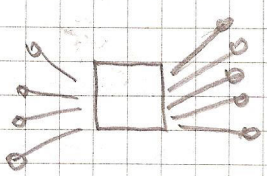
Intel Paragon



2860

IBM SP

Power



CRAY

разъёмы подробнее

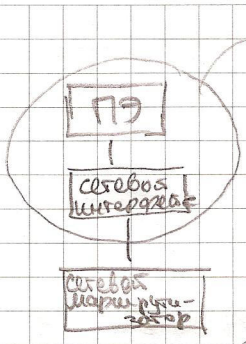
T3D (T3E) ... XT3, X3
95г 97г 2004г 2010

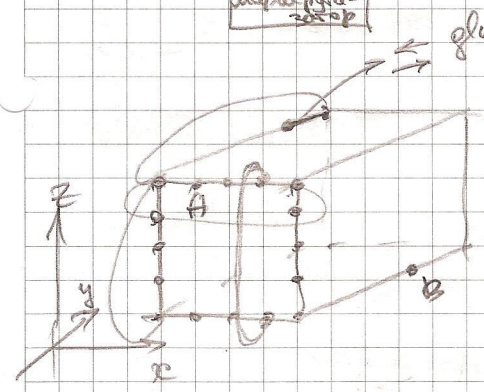
ВУ (вычислительные узлы):

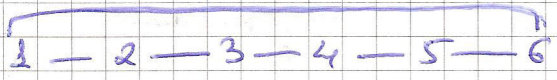
- управляющие	7
- узлы операционная система	5
- вычислительные узлы	260

CRAY T3D, T3E АЛЬФА
XT3 AMD

ПЭ - процессорный элемент







все кроме одной связи короткие

СКАУ ТЭС/1200 — 480 МБ/с

$A(x_A, y_A, z_A) \rightarrow B(x_B, y_B, z_B)$

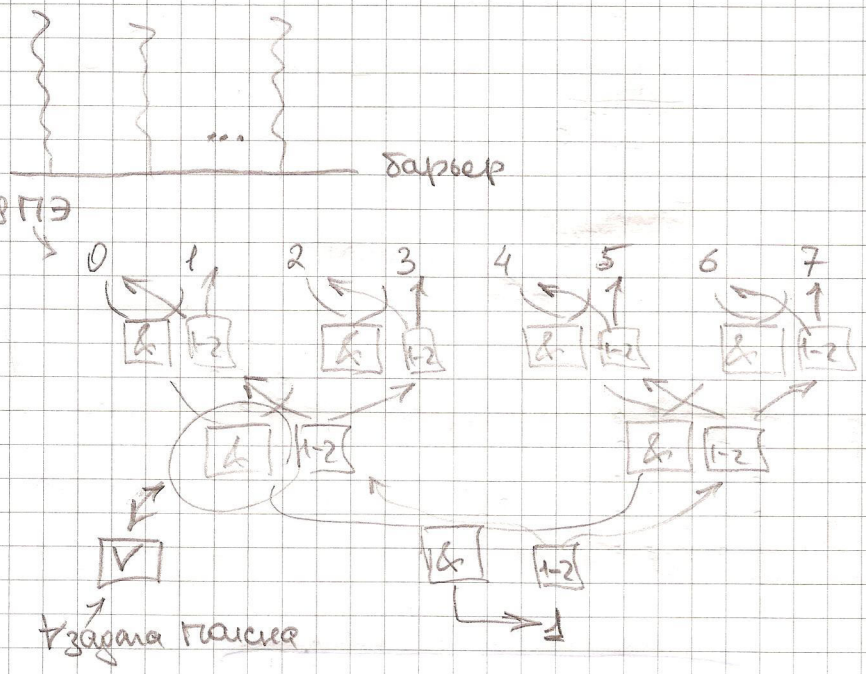
сначала $x_A \rightarrow x_K$

$y_A \rightarrow y_K$

$z_A \rightarrow z_K$

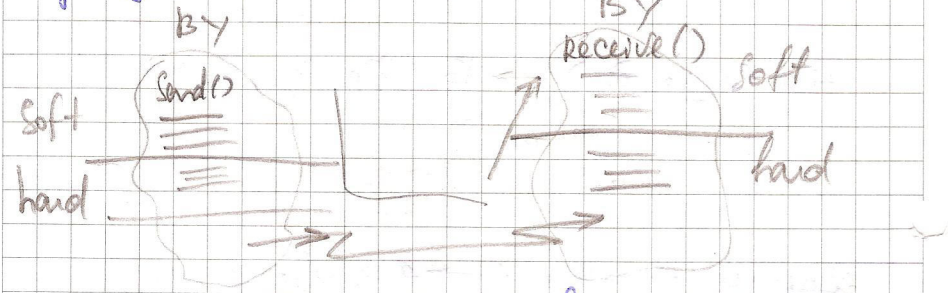
если из $A \rightarrow K$ и $K \rightarrow A$ одновременно,
то, как правило, маршруты не совпадают

Барьерная синхронизация



- доступные процессоры
- доступные сетевые технологии
- доступное ПО

} Вычислительные кластеры



- Латентность – интервал времени, за который передается "нулевое" сообщение

- пропускная способность сети

GE lat ~ 50-130 мкс

пр. сп ~ 1 Гбит/с

---

lat ~ 3-7 мкс

пр. сп ~ 10 Гбит/с

и_4

1 Закон Амдала

2 Латентность

3 Скорость передачи данных

4 Асинхронность передачи данных } send()

5 Балансировка вычислительной нагрузки } Recieve()

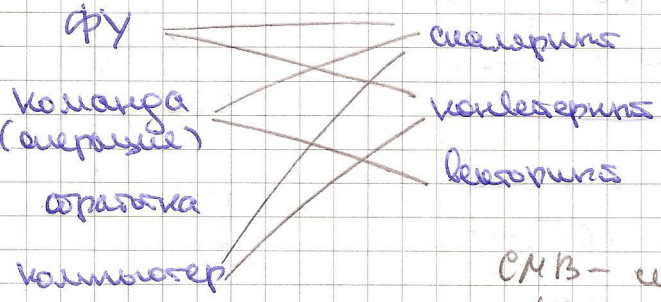
5 Трансляционная прозрачность

- масштабность
- распределенность
- динамичность
- неоднородность
- различие административной принадлежности

Лекция 5 10.10

Векторно-конвейерни компютри

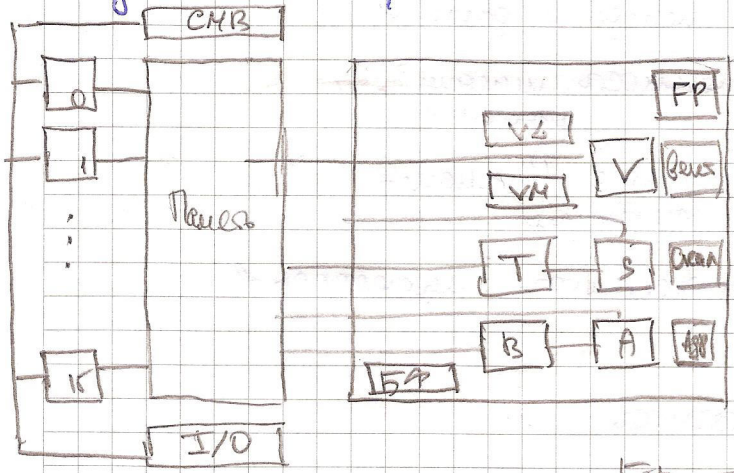
1976 – CRAY-1



СМВ – секция микропроцесорни библиотеки

CRAY C90 90-е год

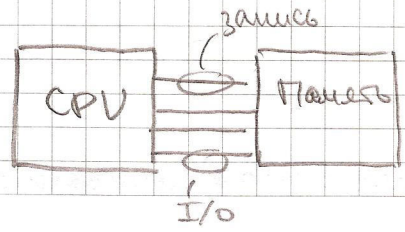
до 16 CPUs ; 4.1 ns (~ 250 MHz)



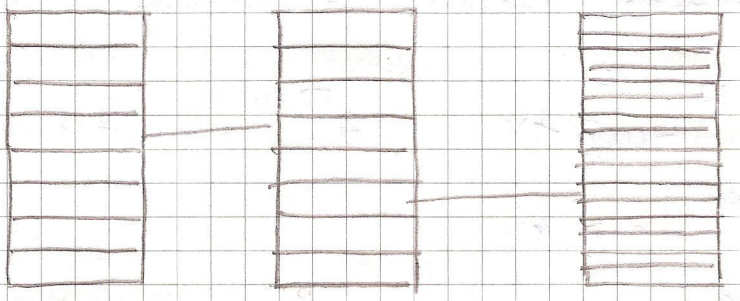
V4 – 8p

VM – 128p

БР – буфер команди



до 8 скалк до 8 векторов до 16 банков

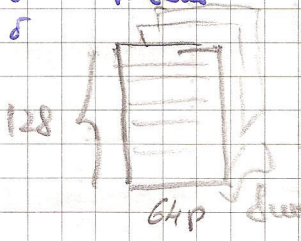


0 ~ 0c 0n 0б
1 ~ 1c 0n 0б
...
7 ~ 7c 0n 0б
8 ~ 0c 1n 0б
9 ~ 1c 1n 0б
...
63 ~ 7c 7n 0б
64 ~ 0c 0n 1б
65 ~ 1c 0n 1б
...

Регистровая структура

	Основная	Вспомогательная
A-адр.	8 × 32р	B ~ 64 × 32р
S-скал	8 × 64р	T ~ 64 × 64р

V-вект



ФУ: – конв.
– неделис.
– сдвиг конвейера 1 – такт

1. Адресная ФУ – 2 вх., целая., скал., 32р
2. Скалярная – 4 вх., целая, скал., 64р

3 Векторная - 5-fma, целочисл., вещ, 64р

4 Вещест. арифметика - 3шт, вещ, скал/вект, 64р

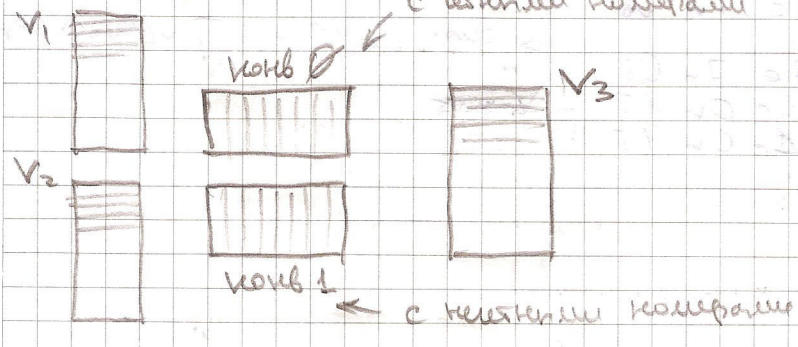
$A_s^{I,32}$ - адр ФУ $A_v^{FP,64}$ - вещ ФУ

$A_s^{I,64}$ - скал ФУ

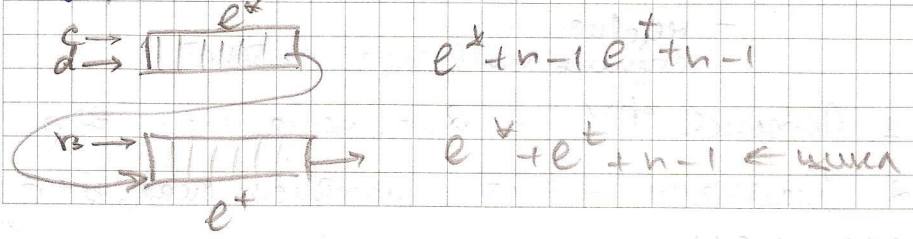
$A_v^{I,64}$ - вект ФУ

$A_s^{FP,64}$ - вещ ФУ

A^v $V_1, V_2 \rightarrow V_3$



$A = B + C * d$



4 on 1 core 4,1 ns

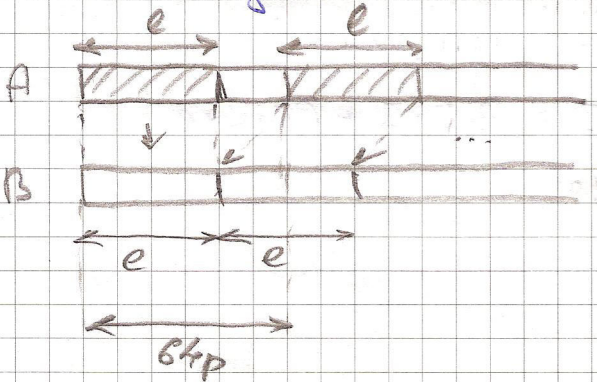
Пиковая производительность ~ 10^9 flops

"4"

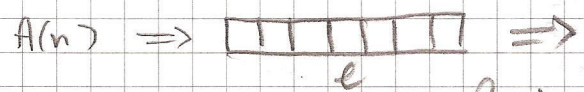
1. Все ФУ конвейерные
2. Все ФУ независимые
3. Векторная обработка
4. Дублирование вычислительных ФУ.
5. Зацепление ФУ
6. 16 независимых процессоров

Лекция 6 17.10

1 Закон Амдала

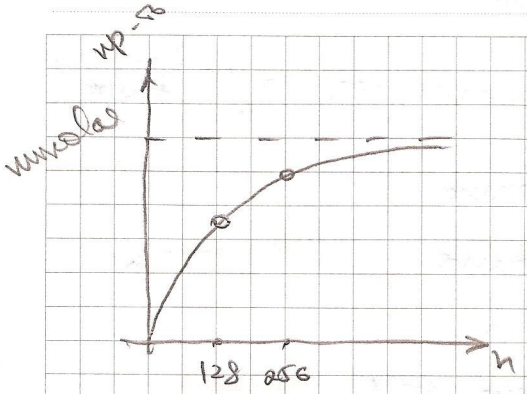


2



$l + n - 1 + 6$

подготовка вычислительного конвейера



Чем больше длина вектора, тем ближе к максимальной производительности.

Если n мало, то сильно влияют b и б

2 Кратность размера конвейера

3 Синхронизация векторных команд

```
for (i=0; i<n; i++)
    A[i] = B[i] + C[i]*e
```

n	Mflops
1	7
4	27
64	301
128	433
129	364
150	413
256	548
257	491
...	...

4 Конфликты в памяти

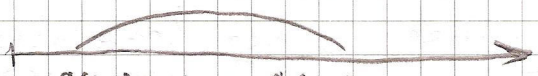
```
for (i=0; i<1000; i+=k)
    A[i] = B[i] + C[i]*e
```

k	Mflops
1	705
2	444
4	274
8	142
16	84
32	44
64	22
128	22

```
do i=1,n
  do j=1,n
    do k=1,n
      A(i,j,k) = A(i,j,k) + B(i,i) * V(k,j)
    enddo
  enddo
enddo
```

5 Mflops

$h = 40 \times 40 = 1600 = 64 \times 25$



$\rightarrow A(40,40,100)$

надо $\Rightarrow$ 41, 41, 100

5. Узкий граф CPU – память

6 Сбалансированность использования ЦУ.

7 Необходимость использования векторных регистров

```
do j = 1, 120
   do i = 1, n
      d(i) = d(i) + s * p(i, j-1) + t * p(i, j)
```

Чтение: $d(i)$, $p(i, j-1)$, $p(i, j)$ | $120 \times 3 = 360$

Запись: $d(i)$ | 120

Развёртка внешнего цикла, множитель 2:

```
do j = 1, 120, 2
   do i = 1, n
      d(i) = d(i) + s * p(i, j-1) + t * p(i, j)
                  + s * p(i, j) + t * (p, j+1)
```

Чтение: $d(i)$, $p(i, j-1)$, $p(i, j)$, $p(i, j+1)$ | $4 \times 60 = 240$

Запись: $d(i)$ | 60

8 Оптимальное число векторных регистров

3 Параллелизм на уровне машинных команд

Суперскалярные архитектуры

VLIW Very long Instruction Word

EPIC Explicitly Parallel Instruction Computing

Лекция 24.10

Технологии параллельного программирования

- возможность создания эффективных параллельных программ
- возможность быстро создавать программы
- переносимость программ
- стоимость

0. Распараллеливающий компилятор
1. Спецкомментарии
2. Расширения существующих языков программирования
3. Специальные языки программирования
   - Оссат - для транспьютеров
   - НОРМА, ЯПМРАН

   декларативные языки программирования
4. Библиотеки и интерфейсы
   - MPI
   - PVM
   - Shmem

5. Параллельные предм. библиотеки
   - PBLAS
   - ScaLAPACK
   - ATLAS
   - MKL
   - FFTW, & FFT Back
   - PETSc

6. Спец. пакеты пр-м
   - GAMESS

Linda

корпение ("p", 3, 1.5)

ПК (кр-во корпений)

out ("p", 3, 1.5);

in ("p", out i, 1.5);

$\equiv$

$i = 5$

in ("p", i, 1.5);

read id val ("p", f(k), ...)

процесс

```
if (My_Id == 0) {...}
    out ("Next", 1);
    :
    for (i=0; i<N proc; i++)
        eval (...);
    :
    in ("Next", formal My_Id)
    out ("Next", My_Id+1);
```

```
C = A * B , N * N
    for (i=0; i<N; i++) {
        out ("A", i, <i-th col of A>);
        out ("B", i, <i-th row of B>);
    }
    for (i=0, i<N proc; i++)
        eval ("Proc", elem());
    out ("Next[0,0]", 1);
    elem() {
        in ("Next[i,j]", formal Next);
        if (Next < N*N)
            out ("Next[i,j]", Next+1);
```

```
Nrow = (Next-1)/N+1;
Ncol = (Next-1)%N+1;
read ("A", Nrow, formal row);
read ("B", Ncol, formal icol);
for (irow=0, irow<N
  out ("result", Nrow, Ncol, &dopod (row,col));

for (irow=0; irow<N; irow++)
  for (icol=0; icol<N, icol++)
      in ("result", irow+1, icol+1, formal
                          C[irow][icol]);

барьерная синхр
   out ("Forkamer", N);
---
   in ("Forkamer", formal Kar);
   Kar = Kar-1;
   if (Kar!=0) {
      out ("Forkamer", Kar);
      read ("Kamer");
}
```

```
else out ("Banner");
```

31.10

Технология программирования MPI.

SPMD – Single Program Multiple Data

MPI – Message Passing Interface

Fortran, C, C++

```
MPI_Init (int *argc, char ***argv)
MPI_Initialized (int *flag)
MPI_Finalize ()
MPI_Comm_size (MPI_Comm comm, int *size)
```

↑ число процессов в коммуникаторе

```
MPI_Comm_rank (MPI_Comm comm, int *rank)
```

↑ номер процесса в коммуникаторе

```
#include "mpi.h"
...
main (int argc, char **argv)
{ int size, n;
  ...
  MPI_Init (&argc, &argv);
  ...
  MPI_Comm_Size (MPI_COMM_WORLD, &size);
  MPI_Comm_Rank (MPI_COMM_WORLD, &n);
```

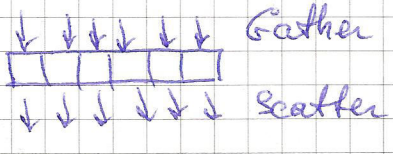
```
    printf ("%d %d\n", size, n);
    ...
    MPI_Finalize ();
}

MPI_Send (void *buf, int count,
          MPI_Datatype, int dest, int tag,
          MPI_Comm comm);
MPI_Recv ( —u— , MPI_Status *status);
MPI_Probe (...)
MPI_Get_count (...)
MPI_Bsend (...)
MPI_Ssend (...)
MPI_Rsend (...)
MPI_ISend (—u—, MPI_Request *request);
MPI_IRecv (..., MPI_Request *request);
MPI_Wait (*request, *status);
MPI_Test (*request, int *flag, *status);
MPI_Init_Send ( )
MPI_Init_Recv ( )   } => MPI_Startall (...)
```

Коллективные операции:

MPI_Bcast (void * buf, int count, dtype, root, comm)

MPI_Scatter

MPI_Gather



MPI_Reduce ( )

MPI_Reduce All ( )

MPI_MAX ( )

MPI_SUM ( )

MPI_Comm_split (comm, int color, int key, * new comm)

Лекция 28.11

parallel. ч

(N, A, F)

Расеш 2-мерная цикл:

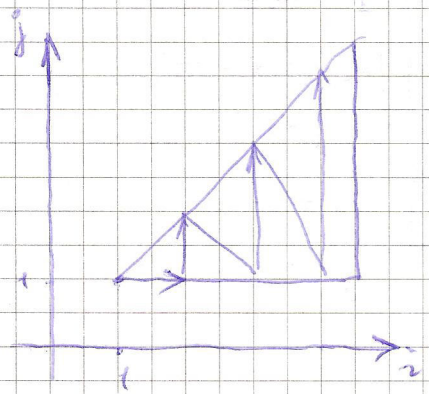
do i = 1, n

do j = 1, m

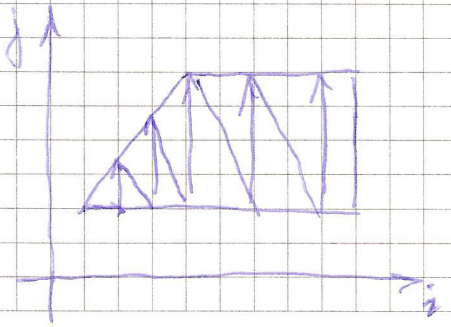
if (i ≥ j) S = S+1

endo

end do



Другое соотношение параметров m и n



SPARSKIT

1) $\begin{cases} 2 \le n \le m \\ \begin{cases} j \le i \le n \\ j \ge 2 \end{cases} \\ i_1 = i \\ j_1 = j - 1 \end{cases}$

2) $\begin{cases} 2 \le n \le m \\ \begin{cases} 2 \le i \le n \\ j = 1 \end{cases} \\ i_1 = i - 1 \\ j_1 = i - 1 \end{cases}$

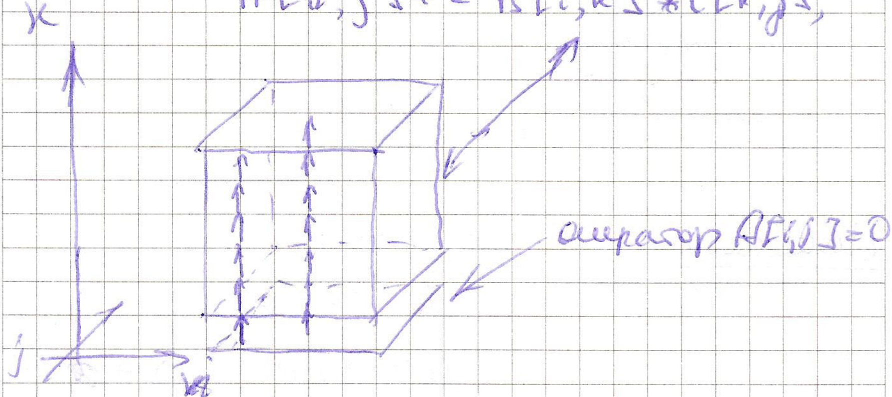
3) $\begin{cases} 2 \le m \le n-1 \\ \begin{cases} j \le i \le n \\ 2 \le j \le m \end{cases} \\ i_1 = i \\ j_1 = j - 1 \end{cases}$

4) $\begin{cases} 1 \le m \le n-1 \\ \begin{cases} 2 \le i \le m+1 \\ j = 1 \end{cases} \\ i_1 = i - 1 \\ j_1 = i - 1 \end{cases}$

5) $\begin{cases} 1 \le m \le n-1 \\ \begin{cases} m+1 \le i \le n \\ j = 1 \end{cases} \\ i_1 = i - 1 \\ j_1 = m \end{cases}$

A = BC   !! ~~~~~~

```
for (i = 0; i < n; ++i)
    for (j = 0; j < n; ++j)
        A[i, j] = 0;
        for (k = 0; k < n; ++k)
            A[i, j] += B[i, k] * C[k, j];
```



паралелізм

конечна — кількість паралельних вітвей жорстко фікс

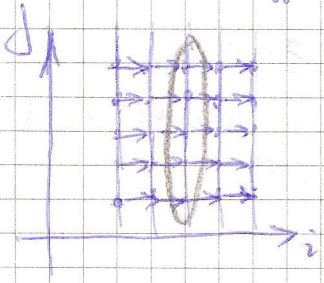
масовна — координатна, скошенна

Коорд: до $i=1,n$

до $j=1,n$

все итерации имеют зависимости

$a(i,j) = a(i-1,j) + x(i)$

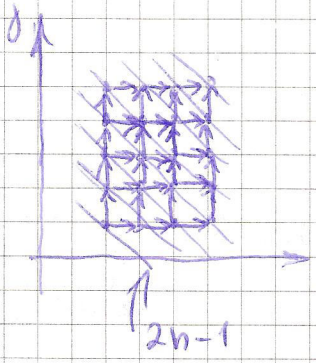


кол-во операторов меньше по времени // координ

Диагональ: до $i=1,n$

до $j=1,n$

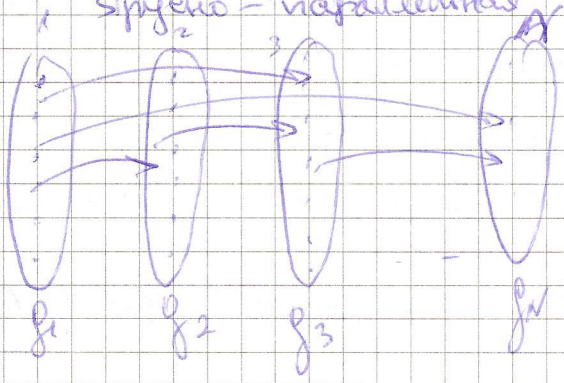
$a(i,j) = a(i-1,j) + a(i,j-1)$



$O(n)$

критическая часть!

Ярусно-параллельная форма



$$\left.\begin{array}{l} A \to B \\ A \in f_i \\ B \in f_j \end{array}\right\} \Rightarrow i < j$$

N – длина ярусно-// формы

Каноническая ЯПФ : если $\forall$ вершины $s \in f_k$ ведет хотя бы одна дуга в $k-1$

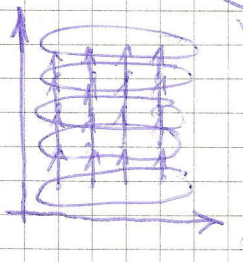
$\max\limits_k |f_k|$ – ширина ЯПФ

Элементарные преобразования циклов:

V do i = 1, n

(do j = 1, n

A(i,j) = A(i, j-1) + x(i)



↓ метод V⁺ для CRAY C90

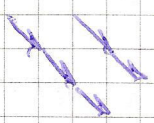
⇒ do j = 1, n

do i = 1, n

A(i,j) = A(i,j-1) + x(i)

– перестановка циклов

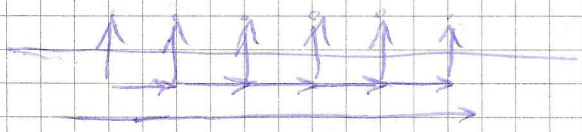
A(i,j) = A(i-1, j+1) + x(i)



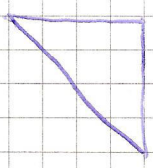
циклы имеют метод

– распределение циклов

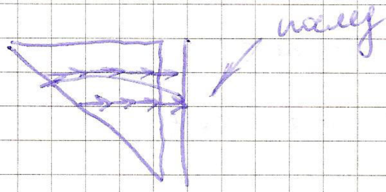
```
do  i=1, n
    a(i) = a(i-1) + s
    b(i) = b(i) + a(i) * t
enddo
```



```
=>  do  i=1, n
Se      a(i) = a(i-1) + s
Pq  (do  i=1, n
        b(i) = b(i) + a(i) * t
```



```
do  i=n, 1, -1
    s=0
    do  j=i+1, n
        s = s + A(i,j) * x(j)
    enddo
    x(i) = (b(i) - s) / A(i,i)
enddo
```



Заменим второй цикл на
n, 0+1, -1