Высокопроизводительные вычислительные системы

1. Варфоломеев В.А., Лецкий Э.К., Шамров М.И., Яковлев В.В. Высокопроизводительные вычислительные системы на железнодорожном транспорте. –М.:ГОУ «Учебно-методический центр по образованию на железнодорожном транспорте», 2010
2. Варфоломеев В.А., Лецкий Э.К., Шамров М.И., Яковлев В.В. "Архитектура и технологии eServer zSeries", 2005 (www.intuit.ru)
3. Эбберс М., О'Брайен У., Огден Б. Введение в современные мэйнфреймы: основы z/OS – www.ibm.com
4. Варфоломеев В.А. Работа пользователя z/OS в среде ISPF/PDF.- М.:МИИТ, 2008

Вычислительная система (ВС)

- - взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации
- - комплекс средств вычислительной техники, содержащий не менее двух основных процессоров или ЭВМ с единой системой управления, имеющих общую память, единое математическое обеспечение ЭВМ и общие внешние устройства

Производительность ВС

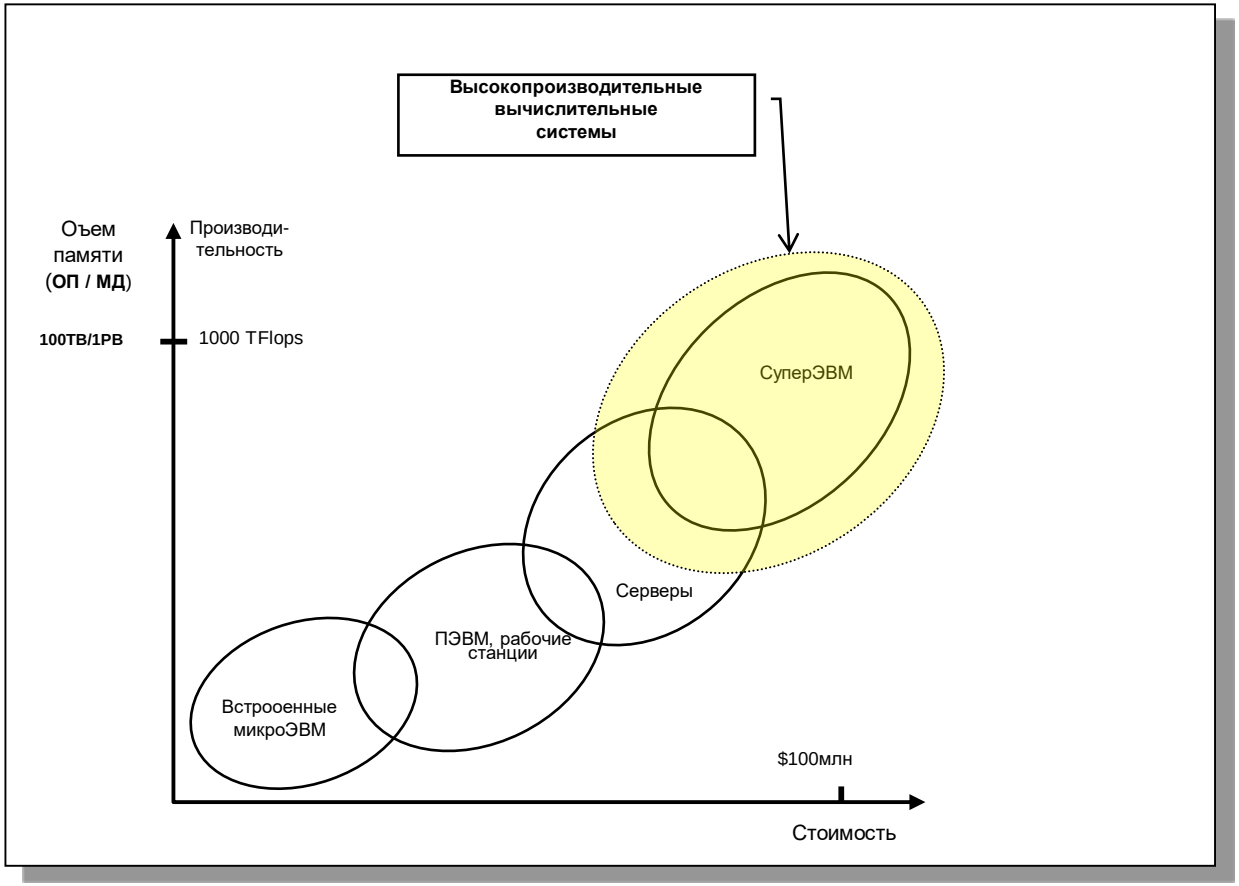
Производительность ВС = V/T

V – объем выполненной работы, T – время выполнения

Виды выполняемых работ:
- Пакетные вычисления без в/в;
- Пакетные вычисления с в/в;
- Обработка транзакций;
- Смешанный

Пиковая производительность измеряется в MIPS или FLOPS

Класификация ВС



Области применения ВВС

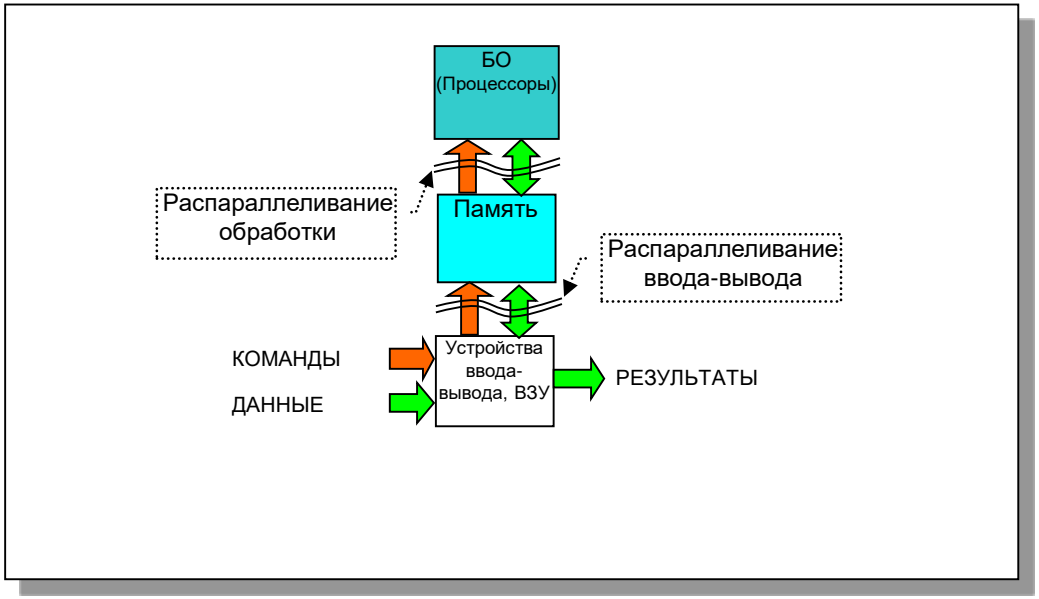
- *управление крупномасштабными техническими системами, в том числе транспортными, экономическими и др.*
- *моделирование сложных объектов с целью экономии средств на натурные эксперименты*
- *прогноз погоды и моделирование изменения климата*
- *синтез новых материалов*
- *сейсморазведка*
- *электро- и гидродинамика*
- *и др.*

www.top500.org

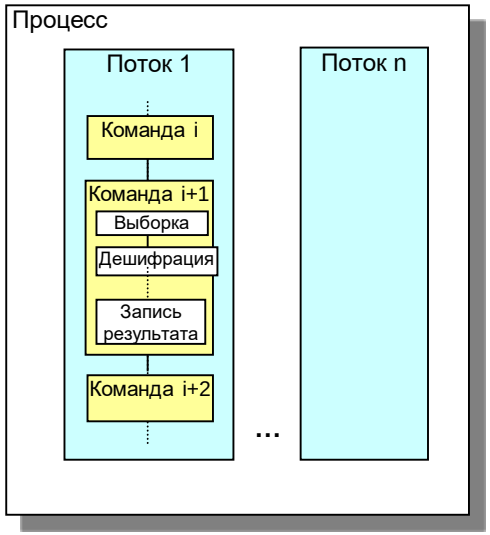
Пути повышения производительности ВС

- Совершенствование элементной базы аппаратных компонентов
- Использование архитектурных решений, основанных на распараллеливании вычислительного процесса

Распараллеливание вычислительного процесса



Гранулярность потоков команд



Модели распараллеливания

- SPMD (Single Program / Multiple Data) - параллельное исполнение нескольких копий одной программы в разных узлах системы над разными потоками данных.
- Конвейер - предусматривает поступление данных в первый процесс, после исполнения которого обработанные данные передаются во второй процесс, а в первый процесс загружаются новые данные и т.д. В результате при длинных потоках данных одновременно в работе несколько процессов.
- “Разделяй и властвуй ” - запуск одного процесса, который порождает другие процессы с передачей им части работы. Новые процессы в свою очередь запускают следующие процессы и т.д.
- “Портфель задач” (Replicated worker) - использование очереди, из которой процессы получают задачи для выполнения. После завершения задачи процесс получает из очереди новую задачу. Если задача порождает новые задачи, они включаются в общую очередь.

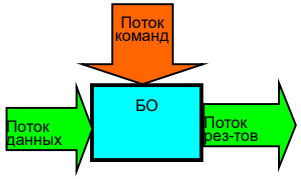
Закон Амдала

$$S \leq 1/(f+(1-f)/n)$$

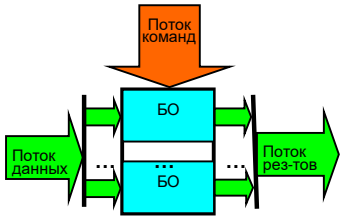
- S – коэффициент ускорения вычислений;
- n – число трактов обработки – параллельно функционирующих обрабатывающих устройств (узлов) ВС: АЛУ, процессоров, ЭВМ
- f – часть алгоритма, не допускающая распараллеливания ($0 \leq f \leq 1$)

Классификация способов параллельной обработки по Флинну.

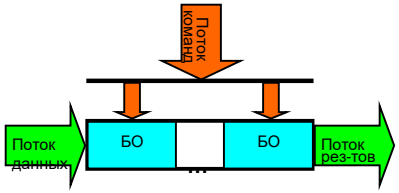
а) одиночный поток команд и одиночный поток данных **(SISD)**



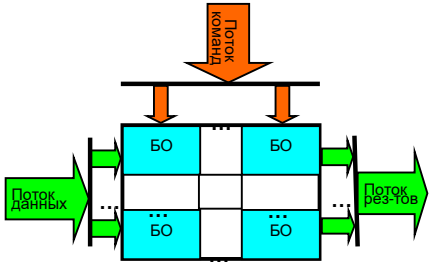
б) одиночный поток команд и множественный поток данных (**SIMD**)



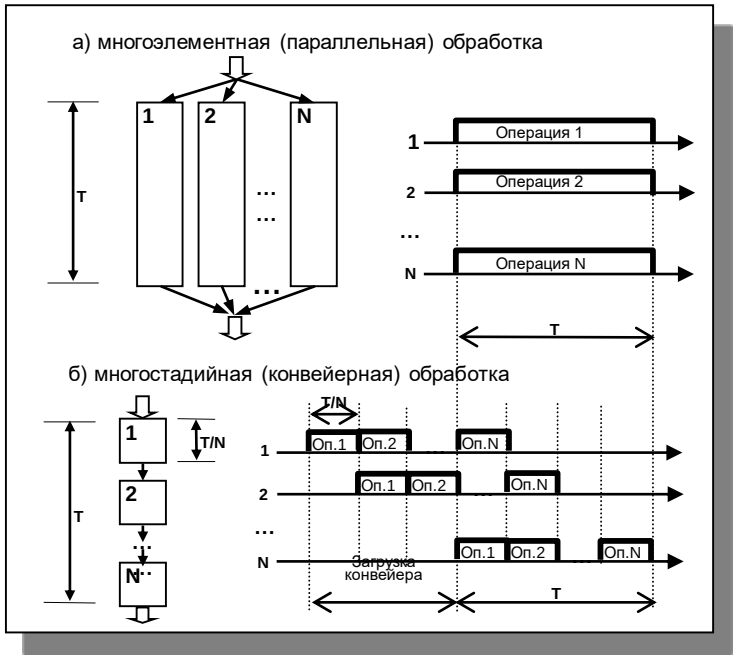
в) множественный поток команд и одиночный поток данных (**MISD**)



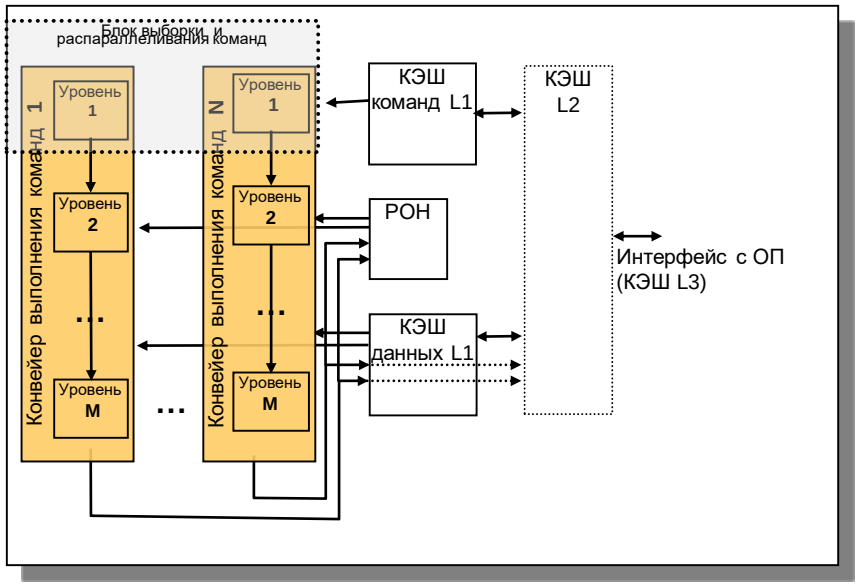
г) множественный поток команд и множественный поток данных (**MIMD**)



Способы распараллеливания работы устройств



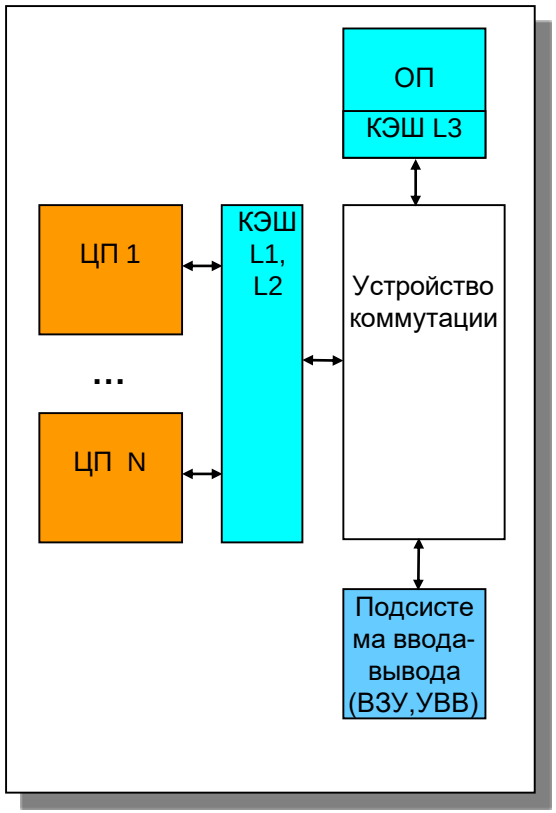
Параллелизм на уровне процессора



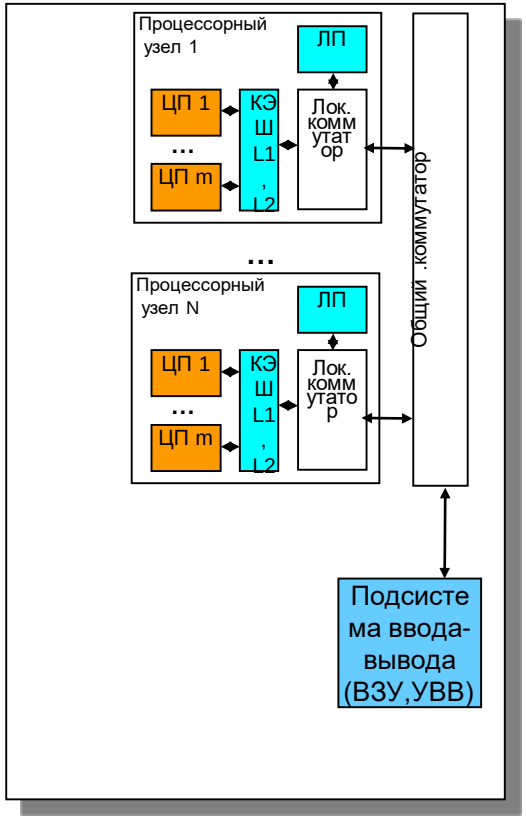
Типы процессоров:

- Ковейерный (N=1, M>1)
- Суперконвейерный
- Суперскалярный (N>1, M>1)
- Векторный

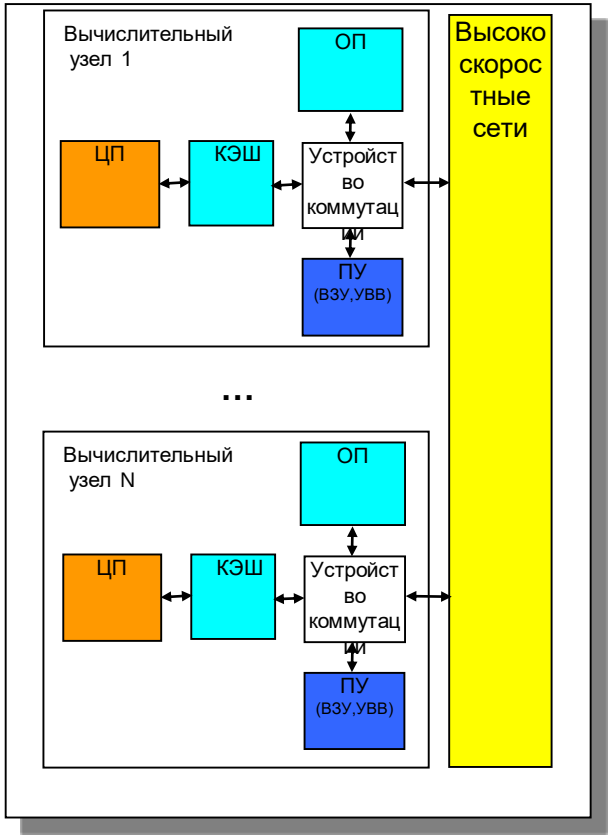
Параллелизм на уровне организации ВС



SMP-система с общей памятью



NUMA-система



Система с распределенной памятью

Кластеры

Кластер - группа компьютеров, объединённых высокоскоростными каналами связи и используемых как единый вычислительный ресурс (для решения одной задачи) (~1995)

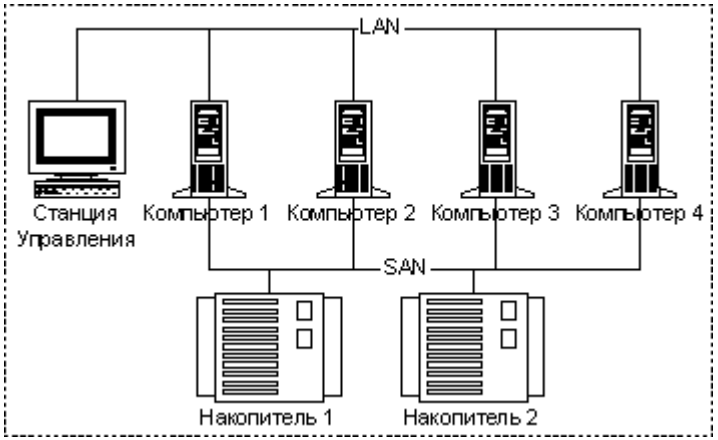
Узлы: однопроцессорные компьютеры, 2-4-процессорные SMP серверы

Преимущества:

- наиболее выгодное соотношение "цена / производительность"
- высокий уровень масштабируемости системы путем добавления или исключения стандартных вычислительных узлов
- свобода в выборе конфигурации
- высокая отказоустойчивость: при выходе из строя вычислительного узла его легко заменить без остановки системы

Типы кластеров:

- HPC (High-performance cluster)
- HAC (High-availability cluster)
- Load Balancing
- GRID



Кластеры

Особенности:

- необходимость использования специального ПО
- необходимость создания специальных прикладных программ
- высокие требования к инфраструктуре (энергопотребление (>100кВт), бесперебойное питание, помещения, кондиционирование, охлаждение, шумоизоляция, обслуживающий персонал)
- кадровая проблема (программисты, администраторы, технические специалисты)

Решения:

- Готовые кластер-компьютеры (Tyan Typhoon)
- Проектирование собственного кластера (установка на своей или чужой площадке)
- Аренда кластера

Этапы проектирования кластерной системы

Решение задач на основе параллельной системы (кластера) требует:

- учитывать на всех стадиях, что задача будет решаться на основе кластерной системы;
- помнить, что недостатки ранее принятых решений скажутся на эффективности работы программы;
- качественной реализации самой кластерной системы.

Компоненты кластера

- средства размещения и компоновки элементов кластера:
- вычислительные узлы;
- головной узел;
- хранилище данных или файл-сервер;
- сетевая инфраструктура;
- средства управления (KVM переключатель, сервисная сеть, компьютерный класс);
- источники бесперебойного питания.

Размещение компонентов кластера

- размещение в стойку, на стеллажах, на столах (компьютерный класс);
- выбор форм-фактора узлов (ширина=19”, высота – 1U÷4U(U~45мм) –ГОСТ 28601.1-90
- блейд-системы (blade) (U<40мм, до 100 узлов в корзине)

Выбор вычислительных узлов кластера

- тип процессора: Intel Xeon, Intel Itanium, AMD Opteron …, предпочтительно многоядерные;
- тактовая частота (с ростом частоты растет энергопотребление!);
- объем и быстродействие оперативной памяти;
- структура и объем кэш-памяти;
- наличие локальных дисков (при возникновении сетевых проблем, для файла подкачки, локальная память для промежуточных массивов)
- опционально: floppy и CD/DVD приводы, USB порт, COM-порт

Головной узел: взаимодействие с пользователями, компиляция программ, подготовка исходных данных, запуск программ

Файл-сервер или дисковое хранилище данных

Сетевая инфраструктура кластера

- **Коммуникационная сеть** – служит для обмена между параллельными процессами:

  - Ethernet 100 Мбит/с или 1Гбит/с (при малом числе обменов) (латентность 50-150мкс);

  - SCI (*Scalable Coherent Interface)* (5,6 Гбит/с, латентность 1,4 мкс);

  - Myrinet-2000/Myri-10G (до 10Гбит/с, латентность 2 мкс);

  - InfiniBand (до 30Гбит/с, латентность <1 мкс);

- **Транспортная (управляющая) сеть** – служит для передачи данных сетевой файловой системы и управления вычислительными узлами (использует отдельный комплект сетевых адаптеров!)

- **Сервисная сеть** – используется для администрирования вычислительных узлов

Примеры кластерных систем

Место установки / Условное название кластера	**г. Томск, ТГУ/СКИФ Cyberia**	**МИИТ/ Т-4700**
Год установки кластера	2007	2008
Количество узлов в кластере	283	64
Количество процессоров на узел	2	2
Тип процессоров	Intel Xeon 5150, 2,66 ГГц	AMD Opteron 2356, 2,3ГГц
Объём оперативной памяти на узел	4 ГБ	16 Гб
Тип коммуникационной сети	InfiniBand	InfiniBand
Тип транспортной сети	Gigabit Ethernet	Gigabit Ethernet
Тип сервисной сети	ServNet	IPMI
Количество жёстких дисков на узел	1	
Форм-фактор узлов	1U	
Управляющий узел? Форм-фактор?	Да, 2U	
Объём файлового хранилища	10 ТБ	60ТБ
Размеры и число стоек	14 стоек 42U, 19": (6 под UPS)	
Суммарная потребляемая мощность	115 кВт	60кВт
Площадь помещения	72 кв.м.	
Версия ОС	SuSe Linux Enterprise Server 10, Microsoft Windows Compute Cluster Server 2003	SuSe Linux Enterprise Server 10
Производительность пиковая / HPL	12/9 Тфлопс	4,7/3,7Тфлопс

Программное обеспечение вычислительного кластера

- **Операционная система** – Linux (стандарт де-факто), Windows Compute Cluster Server 2003;
- **Инструментальное ПО** для автоматизации установки образов ОС на узлы и настройки их на работу в кластере – Rocks, OSCAR;
- **Средства синхронизации времени** (xntp сервер);
- **Средства мониторинга состояния UPS** (NUT);
- **Средства синхронизации системных файлов** (passwd, shadow, hosts,…);
- **Средства безопасности** (включить firewall, для доступа к кластеру использовать ssh, запретить telnet, ftp, samba);
- **Компиляторы** , учитывающие особенности архитектуры используемых процессоров;
- **Средства разработки параллельных программ** (MPI, LAM, OpenMP, Linda, mpC)
- **Средства тестирования**: mpi-benchsuite, Intel MPI Benchmarks
- **Системы управления заданиями** (Cleo, LSF, SunGridEngine, LoadLeveler): - распределение заданий пользователей, квотирование и бюджетирование ресурсов;
- **Специализированное ПО** для решения прикладных задач в различных областях.

По рзелульаттам илссеовадний одонго анлигйсокго унвиертисета, не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве. Галвоне, чотбы преавя и пслоендяя бквуы блыи на мсете. Осатьлыне бкувы мгоут селдовтаь в плоонм бсепордяке, все-рвано ткест чтаитсея без побрелм. Пичрионй эгото ялвятеся то, что мы не чиатем кдаужю бкуву по отдльенотси, а все солво цликеом!

MPI –Message Passing Interface

- Наиболее распространенная технология программирования для параллельных компьютеров (Си, Фортран) (версии 1.1, 2.0)
- Поддержка моделей MIMD и SIMD
- Свободно распространяемое ПО: MPICH (MPICH2)

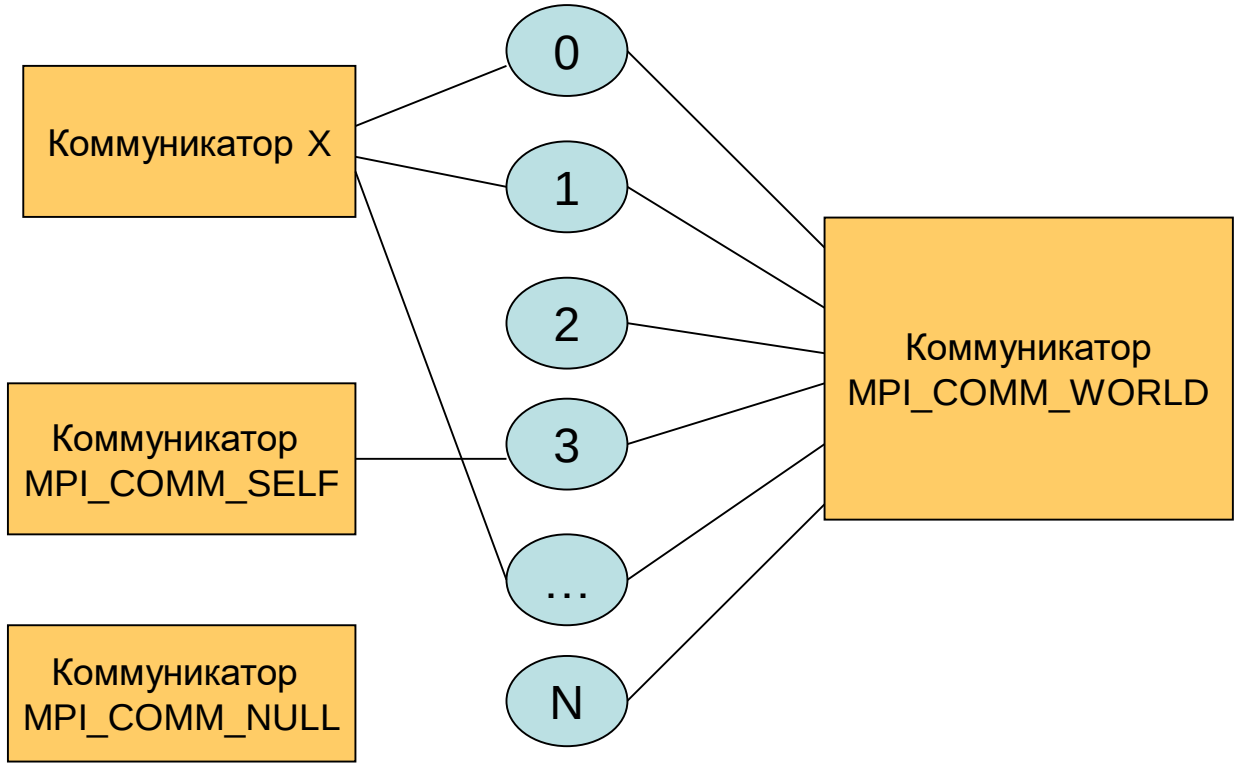
www.mpi-forum.org – стандарт MPI

www.mcs.anl.gov/research/projects/mpich2/index.php - программное обеспечение MPI

parallel.ru/tech/tech_dev/mpi.html - учебные материалы

Концепция MPI

- MPI программа – множество параллельных взаимодействующих процессов
- Все MPI процессы изолированы в собственных адресных пространствах и исполняют один и тот же код (SIMD)
- Атрибуты процесса – коммуникатор и номер в коммуникаторе
- Взаимодействие процессов основано на обмене сообщениями через коммуникаторы. Атрибуты сообщения: номер процесса-отправителя, номер процесса-получателя, идентификатор (тег) сообщения



Базовые функции MPI

MPI_Init() - начало блока параллельных вычислений

MPI_Finalize() – окончание блока параллельных вычислений

MPI_Comm_size() – количество параллельных процессов

MPI_Comm_rank() – идентификатор текущего процесса

MPI_Send() – отправить сообщение

MPI_Recv() – получить сообщение

…

Структура MPI программы

```
#include “mpi.h”
main(int argc, char **argv)
{
…
// Инициализация параллельной части программы (аргументы
   передаются всем процессам)
MPI_Init(&argc, &argv);
…
//  Завершение параллельной части программы
MPI_Finalize();
…
}
```

Подготовка и запуск MPI программы

- Компиляция+Линковка

mpiCC –o программа программа.f

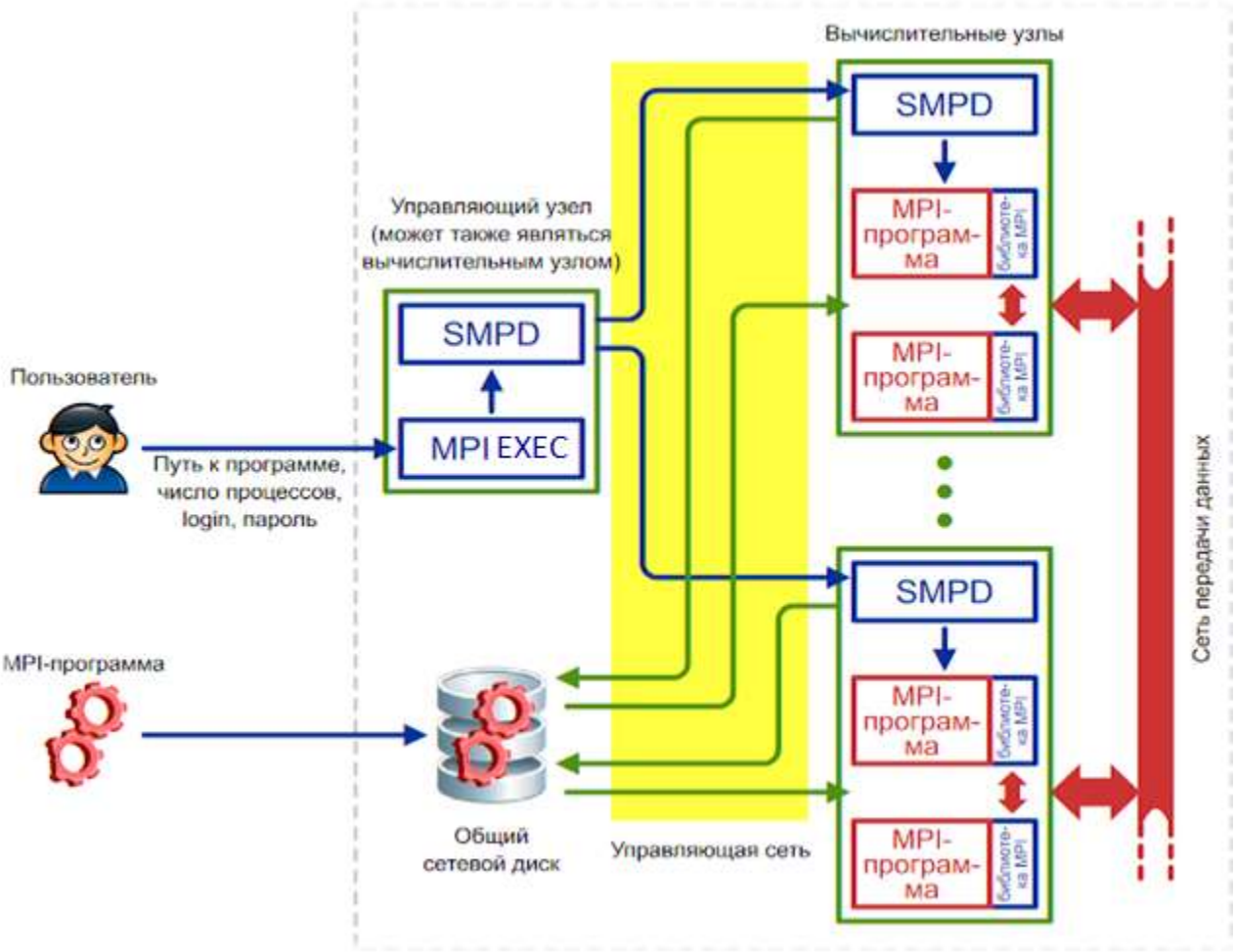
- Запуск

mpiexec -np N программа аргументы

- Поиск и конфигурирование доступных хостов

mpiconfig

Взаимодействие программных компонентов MPICH



Пример1

```
#include “mpi.h”
main(int argc, char **argv)
{
   int me, size;
   MPI_Init (&argc, &argv);
   //  получить количество запущенных процессов (size) в группе по
   //  умолчанию (MPI_COMM_WORLD)
   MPI_Comm_size (MPI_COMM_WORLD, &size);
   //  получить номер текущего процесса (me)
   MPI_Comm_rank (MPI_COMM_WORLD, &me);
   print (“Я #”,me,” из”, size);
   MPI_Finalize();
}
```

```
> mpiexec –np 4 myprog1
Я  #1 из 4
Я  #3 из 4
Я  #0 из 4
Я  #2 из 4
```

Пример 2

```
#include "mpi.h"
int main (int argc, char **argv)
{ char message[20];
  int myrank;
  MPI_Status status;  /* структура атрибутов сообщений    */
  MPI_Init (&argc, &argv);
  MPI_Comm_rank (MPI_COMM_WORLD, &myrank);
  if (myrank==0)      /* код процесса #0 */
    {
      strcpy (message, "Hello!");
      MPI_Send(message, strlen(message), MPI_CHAR, 1, 99, MPI_COMM_WORLD);
    }
  else  /* код процесса #1 */
    {
      MPI_Recv (message, 20, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
      print («Получено сообщение", message);
    }
  MPI_Finalize();
}
```

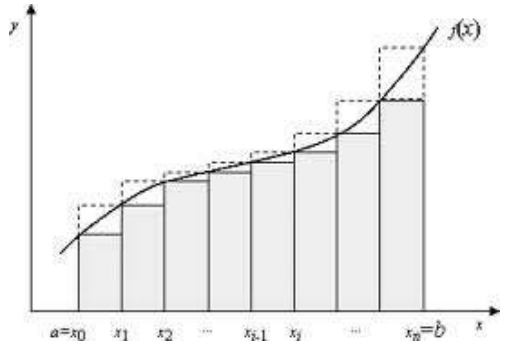
> mpiexec –np 2 myprog2
> Получено сообщение Hello!

Пример 3

```
#include “mpi.h”
main(int argc, char **argv)
{  int me, size;
   int SOME_TAG=0;
   MPI_Status status;
    . . .
   MPI_Init (&argc, &argv);
   MPI_Comm_rank (MPI_COMM_WORLD, &me);
   MPI_Comm_size (MPI_COMM_WORLD, &size);
   if ((me % 2)==0)  /* если номер процесса  me четный */
     { /* послать сообщение следующему процессу  с номером me+1 */
       if ((me+1) < size)
            MPI_Send (..., me+1, SOME_TAG, MPI_COMM_WORLD);
      }
   else  /* иначе, если номер процесса  нечетный, - принять сообщение */
        MPI_Recv (..., me-1, SOME_TAG, MPI_COMM_WORLD, &status);
   . . .
   MPI_Finalize();
}
```

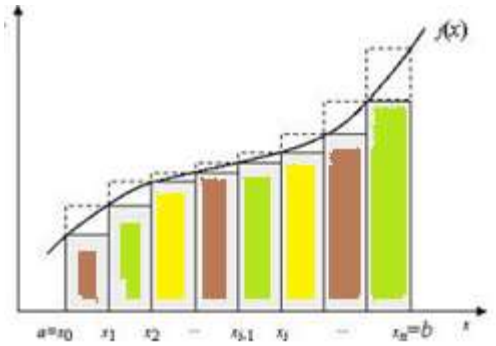
Пример 4: вычисление числа π

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$




```
int main ( int argc, char *argv[ ] )
{ int n, myid, numprocs, i;
  double mypi, pi, h, sum, x, t1, t2, PI25DT = 3.14159265358979323846264 3;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);     /* число процессов */
  MPI_Comm_rank(MPI_COMM_WORLD,&myid);          /* номер текущего процесса*/
  if (myid == 0)
    { printf ("Enter the number of intervals: (0 quits) ");
      scanf ("%d", &n);
      t1 = MPI_Wtime();
    }
  MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
//   for (i=0; i<numprocs; i++)
//       MPI_Send(&n, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
```

```
  if (n == 0) break;
  else
     {  h = 1.0/ (double) n;
        sum = 0.0;
        for (i = myid +1; i <= n; i+= numprocs)
            { x = h * ( (double)i - 0.5);
              sum += (4.0 / (1.0 + x*x));
             }
       mypi = h * sum;
 // Передача 0-му процессу и итоговое суммирование  частных сумм (mypi)
//  процессов в переменной pi
      MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
       if (myid == 0)
          { t2 = MPI_Wtime();
            printf ("pi is approximately %.16f. Error is %.16f\n",pi, fabs(pi - PI25DT));
            printf ("'time is %f seconds \n", t2-t1);
           }
       }
MPI_Finalize();
  return 0;
}
```



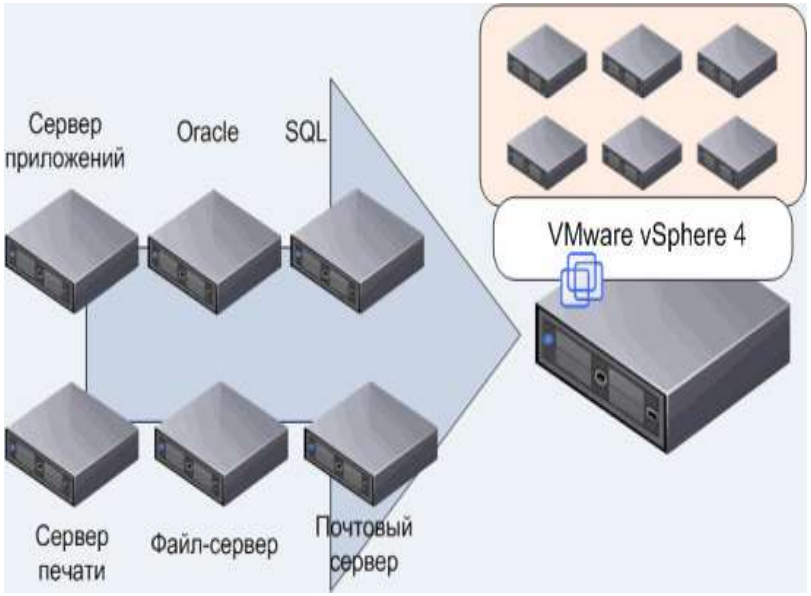
Виртуализация вычислений

Виртуализация вычислений — процесс (1) раздельного представления набора вычислительных ресурсов, или (2) их логического объединения, который даёт какие-либо преимущества перед оригинальной конфигурацией.

Виртуализация серверов —хостинг нескольких независимых операционных систем на одном компьютере.

Виртуальная машина — это программное окружение, которое представляется для «гостевой» операционной системы, как аппаратное.

Преимущества виртуализации



- эффективное использование ресурсов физического сервера
- увеличение скорости разворачивания сервера и независимость сервера от «железа».
- использование несовместимых приложений на одном компьютере.
- высокая надежность («падение» одного сервера не ведет к потере остальных серверов)
- балансировка нагрузки.
- возможность «засыпания“ серверов необходимых периодически

Полная виртуализация

z/VM, VMware VSphere, VMVare Workstation, Sun Virtualbox, Oracle VM, Microsoft Hyper-V

Паравиртуализация

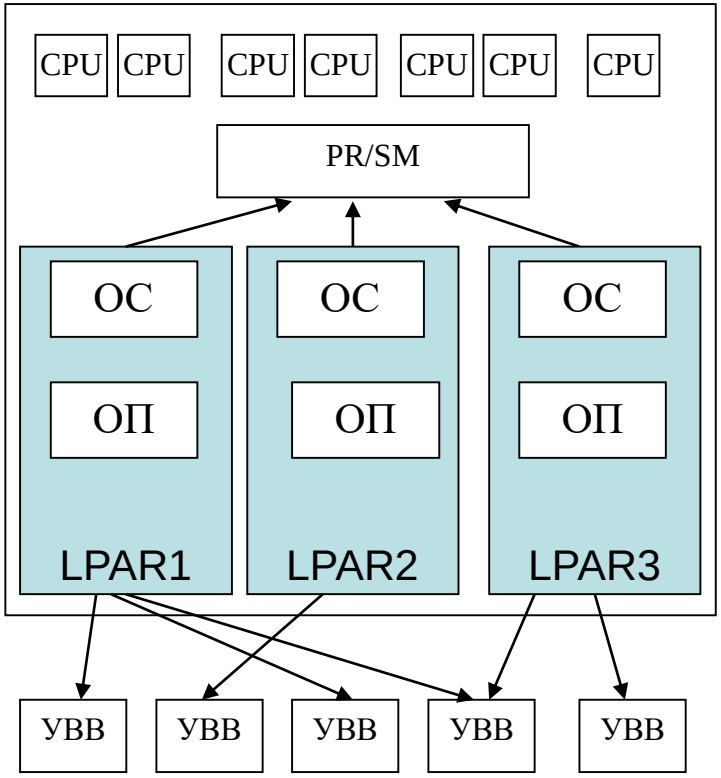
XEN, UML

Виртуализация уровня операционной системы

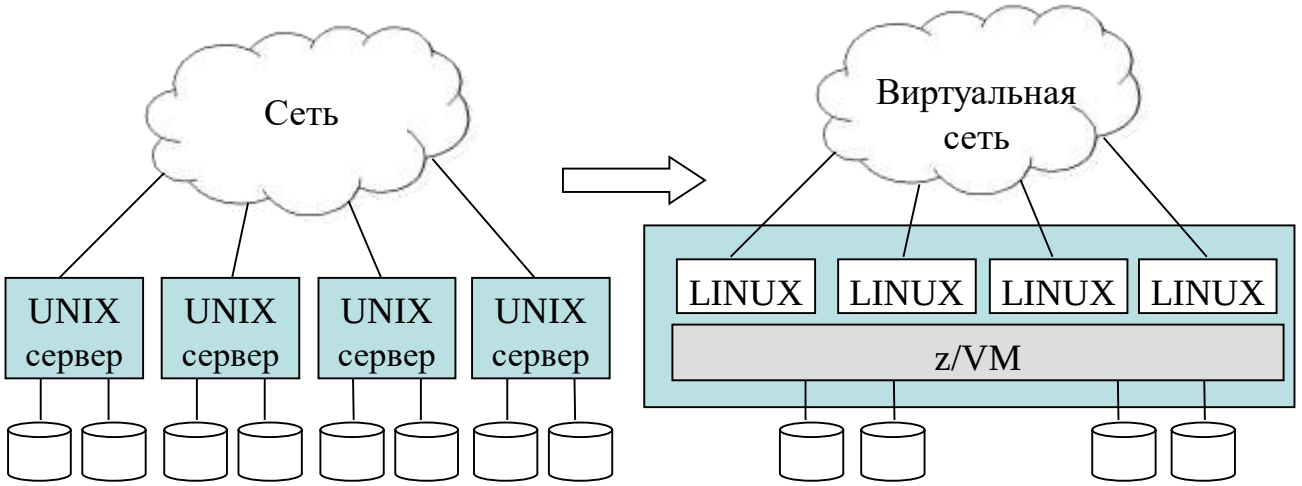
Linux-VServer, Open VZ

Операционная система z/VM

Аппаратная виртуализация



Консолидация серверов



• уменьшение расходов на администрирование и техническое обслуживание;
• снижение времени и расходов на установку и настройку ОС;
• повышение безопасности;
• использование технологии виртуальных сетей для организации взаимодействия серверов;
• использование общего дискового пространства;
• более эффективное использование процессорного времени и других ресурсов (снижение простоев, увеличение загрузки);
• гибкость - вертикальное и горизонтальное масштабирование;
• сокращение затрат на потребление электроэнергии и вентиляцию;

Реализация многоуровневой ИС на базе IBM System z