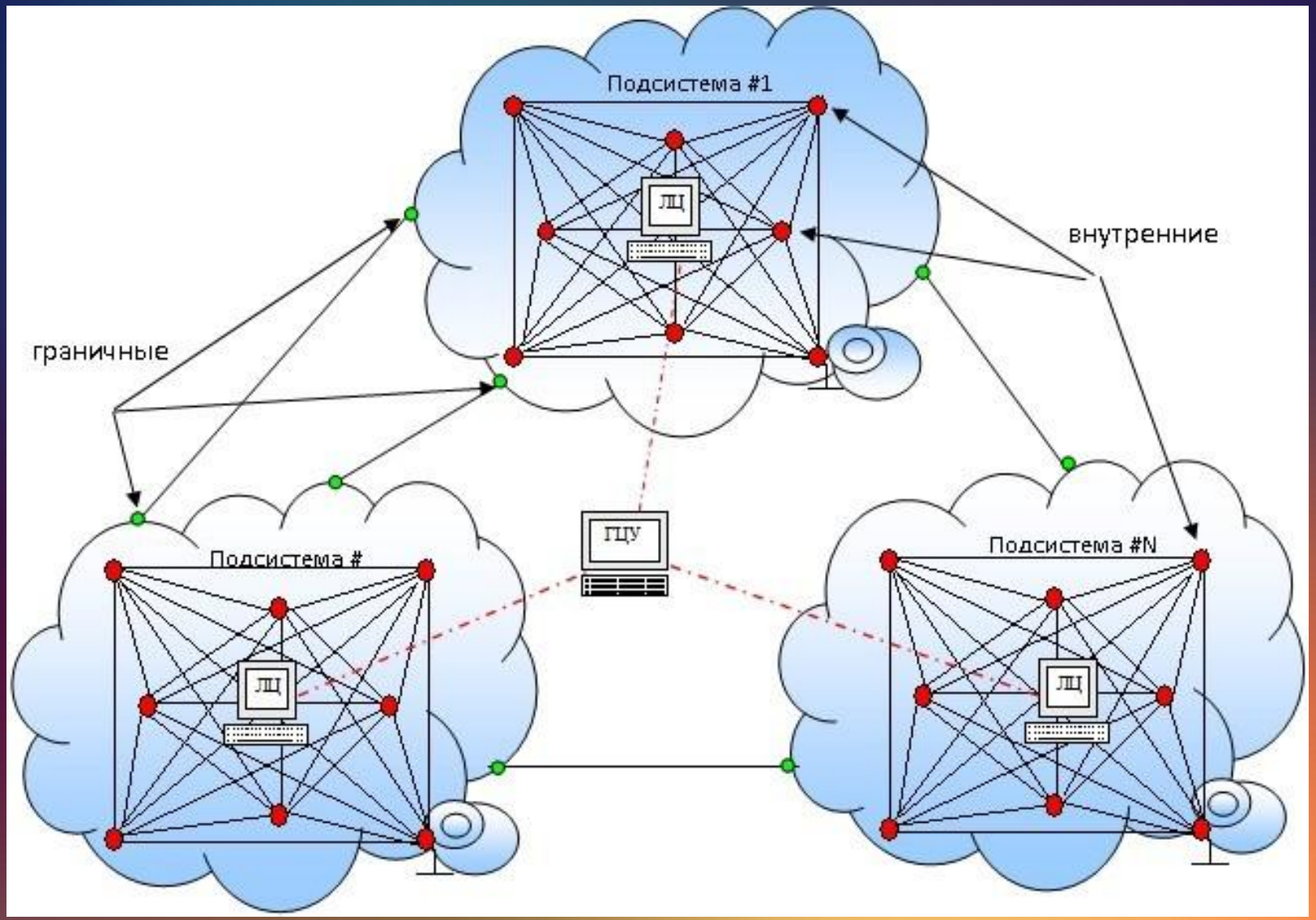
- ПОВОЛЖОСКИЙ ГОСУДАРСТВЕНИЙ ТЕХНОГОГИЧЕСКИЙ УНИВЕРИСТЕТ

- **ПРЕЗЕНТАЦИЯ НА ТЕМУ:Распределенные и параллельные вычисления**
- **ВЫПОЛНИЛ:Олтиев Н**

Параллельные вычисления

- Параллельные вычисления — способ организации компьютерных вычислений , при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно). Термин охватывает совокупность вопросов параллелизм в программировании, а также создание эффективно действующих аппаратных реализаций.


Подсистема #1
ЛЦ
внутренние
граничные
ГЦУ
Подсистема #
ЛЦ
Подсистема #N
ЛЦ

История появления

- Параллельное вычисление развилось из более ранней работы над железными дорогами и телеграфией, от 19-го и в начале 20-го века и некоторой даты условий к этому периоду, такими как семафоры. Они возникли, чтобы обратиться к вопросу того, как управлять многократными поездами на той же самой системе железной дороги (избегающий столкновений и максимизирующий эффективность) и как обращаться с многократными передачами по данному набору проводов (повышающий эффективность), такой как через мультиплексирование с разделением времени (1870-е).

- Научное исследование параллельных алгоритмов началось в 1960-х, с приписанным то, чтобы быть первой бумагой в этой области, определив и решив взаимное исключение.

Сложность при проектировании параллельных программ

- Обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, разделяемых между процессами.

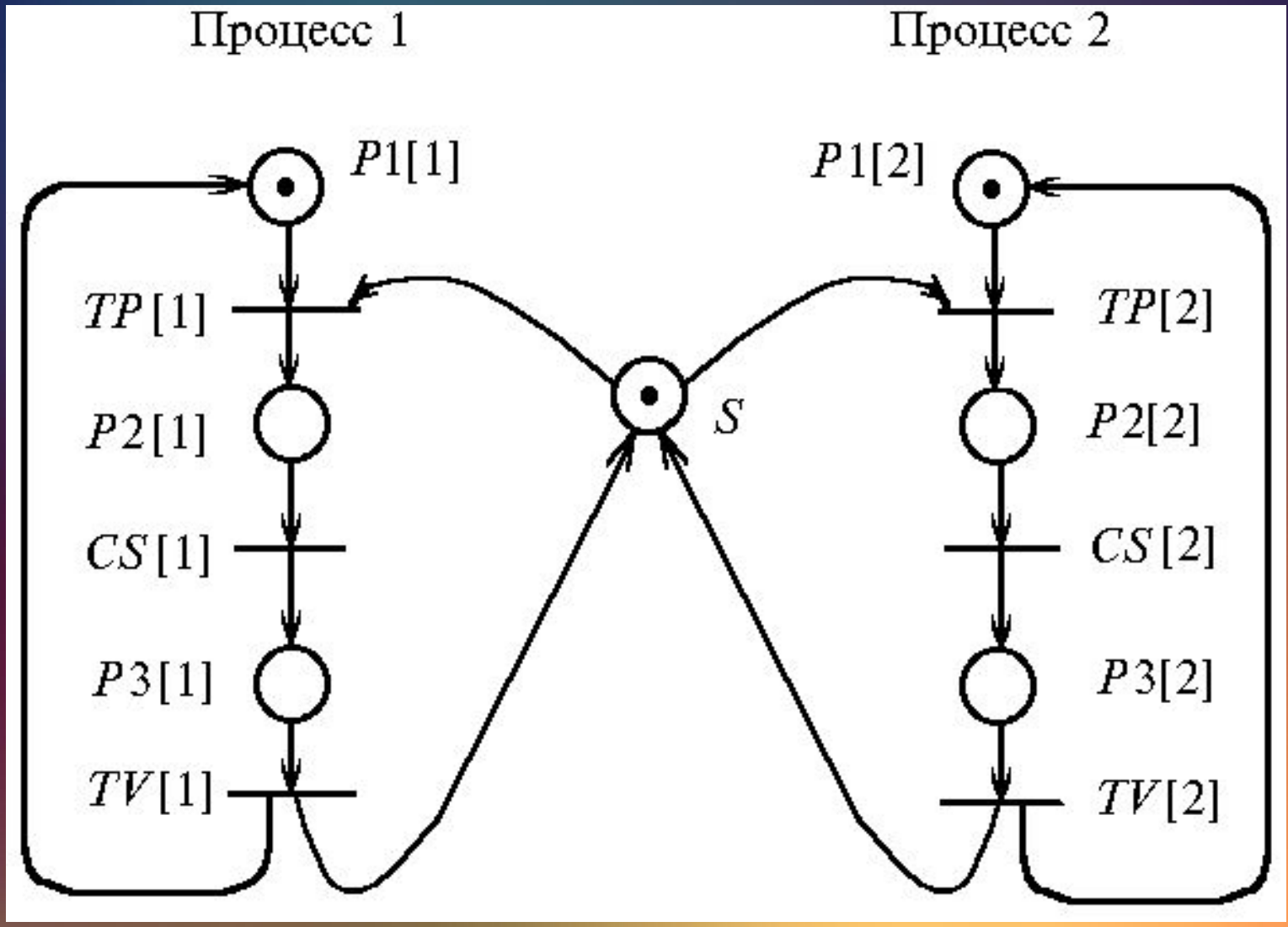
Основная сложность при проектировании параллельных программ

- Обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, разделяемых между процессами.

Способы синхронизации параллельного взаимодействия:

- 1) Взаимодействие через разделяемую память:
- на каждом процессоре мультипроцессорной системы запускается поток исполнения, который принадлежит одному процессу. Потоки обмениваются данными через общий для данного процесса участок памяти. Количество потоков соответствует количеству процессоров. Потоки создаются либо средствами языка (например, в Java или C#, C++ (начиная с C++11), C (начиная с C11)), либо с помощью библиотек явно (например, в C/C++ с помощью PThreads), либо декларативно (например, с помощью библиотеки OpenMP), или автоматически встроенными средствами компилятора (например, High Performance Fortran). Данный вид параллельного программирования обычно требует какой-то формы захвата управления (мьютексы, семафоры, мониторы) для координации потоков между собой.

Синхронизация Процесора



- 2) Взаимодействие с помощью передачи сообщений : на каждом процессоре многопроцессорной системы запускается однопоточный процесс, который обменивается данными с другими процессами, работающими на других процессорах, с помощью сообщений. Процессы создаются явно, путем вызова соответствующей функции операционной системы, а обмен сообщениями — с помощью библиотеки (например, реализация протокола MPI), или с помощью средств языка (например, High Performance Fortran, Erlang или occam). Обмен сообщениями может происходить асинхронно, либо с использованием метода «рандеву», при котором отправитель блокирован до тех пор, пока его сообщение не будет доставлено. Асинхронная передача сообщений может быть надёжной (с гарантией доставки) либо ненадёжной.

- 3) Гибридный способ: на многопроцессорных системах с распределённой памятью (DM-MIMD), где каждый узел системы представляет собой мультипроцессор с общей памятью (SM-MIMD), можно использовать гибридный метод программирования. На каждом узле системы запускается многопоточный процесс, который распределяет потоки между процессорами данного узла. Обмен данными между потоками на узле осуществляется через общую память, а обмен данными между узлами — через передачу сообщений. В этом случае количество процессов определяется количеством узлов, а количество потоков — количеством процессоров на каждом узле. Гибридный способ программирования более сложен (требуется особым образом переписывать параллельную программу), но наиболее эффективен в использовании аппаратных ресурсов каждого узла многопроцессорной системы.

Программные инструменты параллелизма

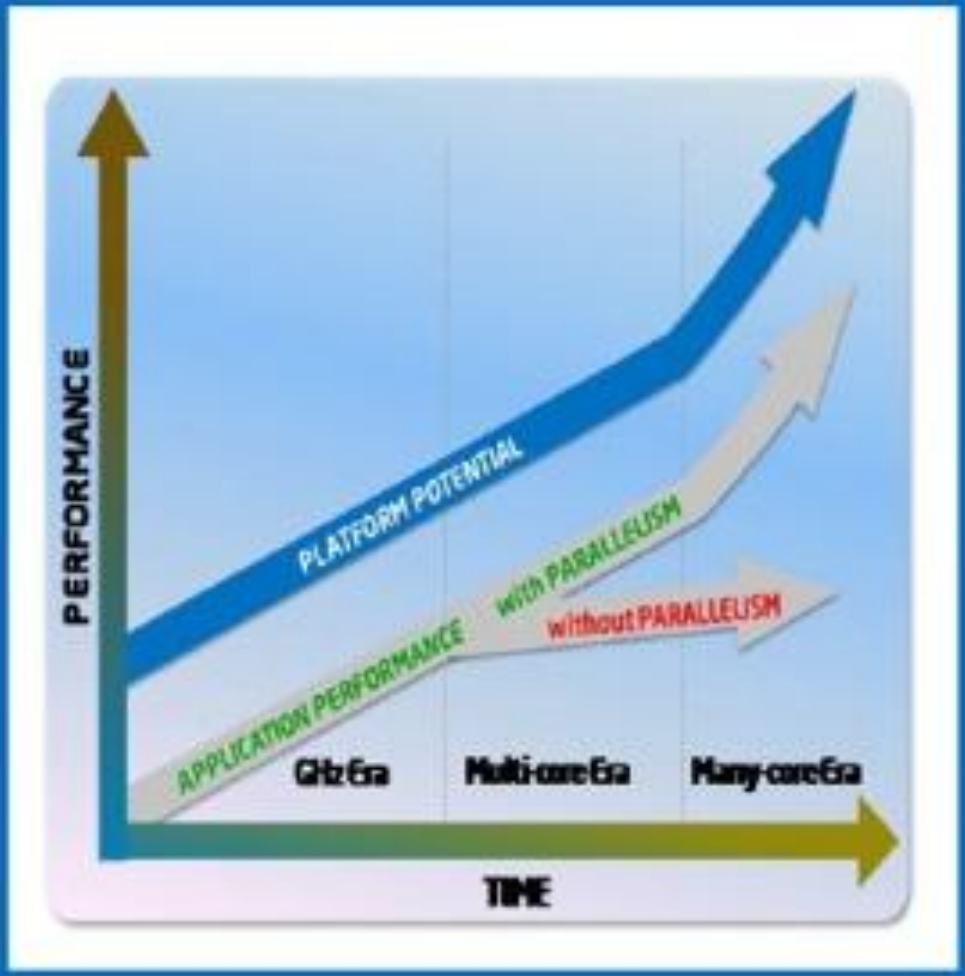
- OpenMP — стандарт интерфейса приложений для параллельных систем с общей памятью.
- POSIX Threads — стандарт реализации потоков (нитей) выполнения.
- Windows API — многопоточные приложения для C++.
- PVM (Parallel Virtual Machine) позволяет объединить разнородный (но связанный сетью) набор компьютеров в общий вычислительный ресурс.
- MPI (Message Passing Interface) — стандарт систем передачи сообщений между параллельно исполняемыми процессами.

Эра параллелизма

Векторизация (на уровне процессора)

Распараллеливание на общей памяти (на уровне одной машины)

Распараллеливание на распределенной памяти (кластерные системы)



Преимущества параллельного вычисления

- Увеличенная прикладная пропускная способность – параллельное выполнение параллельной программы позволяет числу задач, выполненных в определенном периоде времени увеличиваться.
- Высокий живой отклик для ввода/вывода – input/output-intensive заявления главным образом ждет входа или произвел операции, чтобы закончить. Параллельное программирование позволяет время, которое было бы проведено, ожидая, чтобы использоваться для другой задачи.
- Более соответствующая структура программы – некоторые проблемы и проблемные области подходящие к представлению как параллельные задачи или процессы.

Реализация параллельных систем

Производительность компьютеров росла экспоненциально, начиная с 1945 года и до настоящего момента (если брать средний показатель за каждые 10 лет). Компьютерная архитектура претерпела значительные изменения, пройдя путь от последовательной до параллельной.

Производительность компьютера непосредственно зависит от времени, требующегося на выполнение основных функций и количество этих основных операций, которые могут быть выполнены одновременно. Время выполнения одной простейшей инструкции в конечном итоге ограничено.

Несложно сделать вывод, что нельзя ограничиваться увеличением скорости лишь за счет тактовой частоты процессоров. Зависимость от процессоров в конечном итоге заводит в тупик. Другая стратегия в этой области – использование внутреннего параллелизма в чипе процессора. Но такая технология очень дорога. Современные суперкомпьютеры основываются в большей степени на идее использование большого количества относительно не дорогих уже имеющихся процессоров.

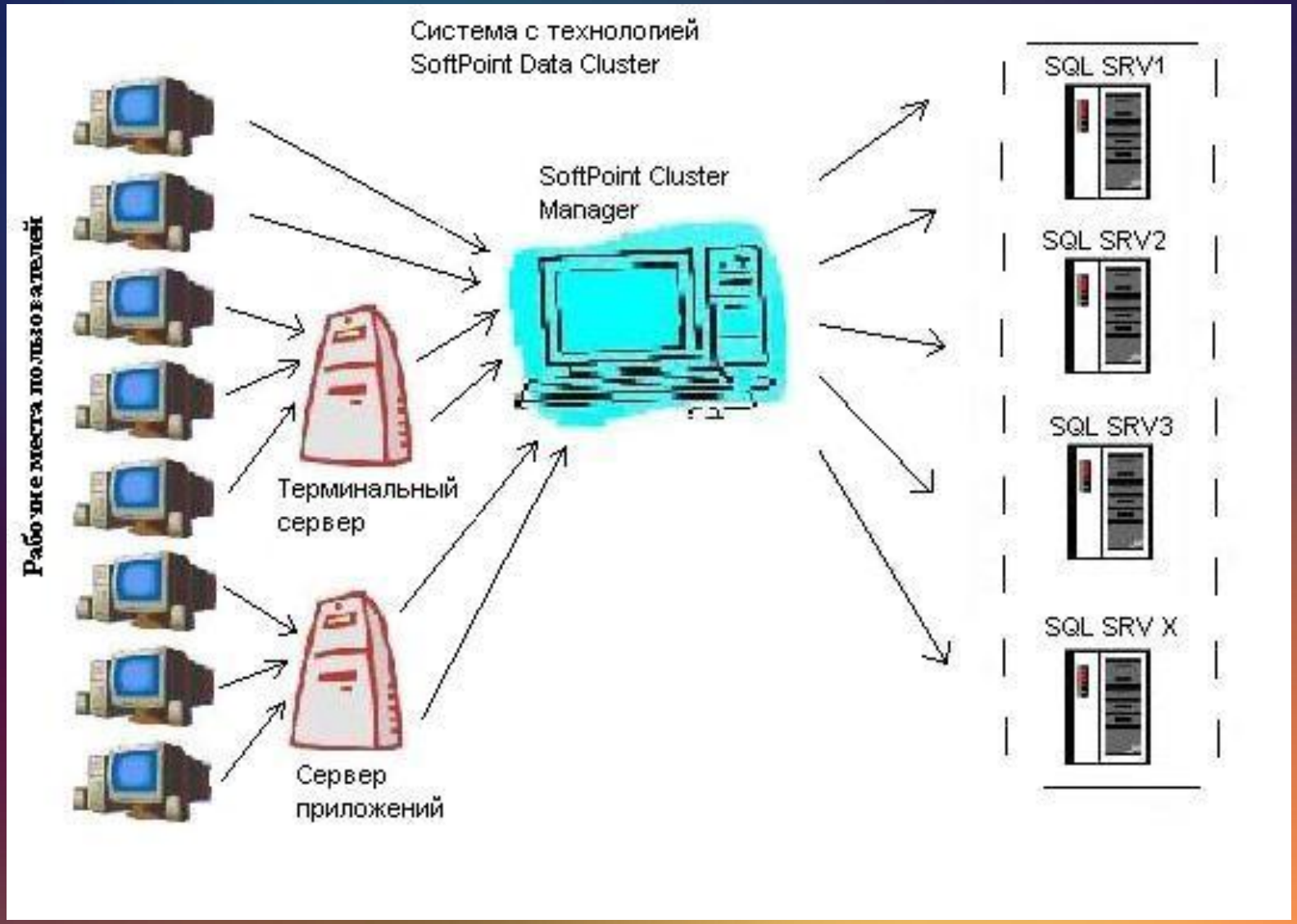
История развития распределенных вычислений

В 1978 году советский математик Виктор Глушков работал над проблемой макроконвейерных распределённых вычислений. Он предложил ряд принципов распределения работы между процессорами.На базе этих принципов им была разработана ЭВМ ЕС-2701.

В 1994 году Дэвидом Джиди была предложена идея по организации массового проекта распределённых вычислений, который использует компьютеры добровольцев (т. н. добровольные вычисления) — SETI@Home. Научный план проекта, который разработали Дэвид Джиди и Крейг Каснофф из Сиэтла был представлен на пятой международной конференции по биоастрономии в июле 1996 года.

Преимущества распределенных вычислений

- Распределенные вычисления идеально подходят для многих компаний. В модели "клиент-сервер" требования к аппаратному обеспечению для серверов намного меньше, чем для мейнфрейма. Это приводит к снижению начальных издержек. Поскольку каждая рабочая станция обладает своей собственной вычислительной мощью, она может работать независимо от сервера. А посредством использования множества серверов, локальных сетей, глобальных сетей, Интернет, системы распределенных вычислений могут простираться на весь мир. Сегодня пользователи могут работать с корпоративной системой со своих переносных компьютеров откуда угодно, даже в самолетах.


Система с технолопией
SoftPoint Data Cluster
Рабочие места пользователей
SoftPoint Cluster
Manager
Терминальный
сервер
Сервер
приложений
SQL SRV1
SQL SRV2
SQL SRV3
SQL SRV X

- Распределенные вычисления также обеспечивают и надежность. Если информация может дублироваться на нескольких серверах, то отключение одного сервера не предотвращает доступ к информации. Тщательное управление репликацией данных может гарантировать, что только какая-нибудь глобальная катастрофа может привести систему в нерабочее состояние. Резервные каналы связи обеспечивают устойчивость к отказам для систем, содержащих критическую информацию. Это было одной из причин, по которой военные изобретали систему распределенных вычислений
- распределенные вычисления позволяют испрользовать устаревшие компьютеры для сложных расчетов, которые они не могли бы делать. Например, старые машины на процессорах 386 могут получать доступ к ресурсам ваших серверов Windows 2000. Причем этот тип доступа прозрачен для пользователя. Разработчик может писать приложение только для одной операционной системы и не проверять, как оно работает на других системах. Это снижает стоимость разработки и программа становится дешевле для пользователя.

Спасибо за внимание