Сетевая ОС является основой информационной вычислительной сети. Поскольку каждый компьютер сети работает самостоятельно, под сетевой ОС в широком смысле понимается совокупность ОС отдельных компьютеров, взаимодействующих с целью обмена сообщениями и разделения ресурсов по единым правилам – протоколам. В более узком смысле сетевая ОС – это ОС отдельного компьютера, обеспечивающая его возможность работать в сети. В сетевой ОС отдельного компьютера можно выделить следующие средства:

1. Средства управления собственными ресурсами. Сюда входят функции распределения ОП между процессами, планирования и диспетчеризации процессов, управлении периферийными устройствами и тд.
2. Средства предоставления собственных ресурсов в общее пользование. Это серверная часть ОС. Эти средства обеспечивают ведение справочников имен, сетевых ресурсов, обработку и управления очередями запросов, удаленного доступа к файловой системе, БД, и периферийным устройствам.
3. Средства запроса доступа к удаленным ресурсам других компьютеров и их использование. Это клиентская часть ОС. Эта часть ОС выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей и осуществляет прием ответов от серверов так, что для приложения и пользователя выполнение локальных и удаленных запросов выглядит совершенно одинаково.
4. Средства коммуникации, с помощью которых происходит обмен сообщениями в сети. Эта часть ОС обеспечивает адресацию и буферизацию сообщений, выбор их маршрута передачи по сети, надежность передачи и другие функции, то есть является средством транспортировки сообщений.

В зависимости от назначения компьютера и выполняемых им функций в его ОС может отсутствовать клиентская или серверная части. Если есть 2 компьютера, один из которых выполняет функции сервера, а второй функции клиента, то их взаимодействие происходит следующим образом: клиентская часть ОС получает все запросы, поступающие от приложений и анализирует их. Если выдан запрос к ресурсу данного компьютера, то он переадресовывается соответствующей подсистеме локальной ОС. Если же это запрос к удаленному ресурсу, то он направляется в сеть. При этом клиентская часть ОС преобразует запрос из локальной формы в сетевой формат и передает его транспортной подсистеме, которая отвечает за доставку сообщений указанному серверу. Серверная часть ОС второго компьютера принимает запрос от транспортной подсистемы, преобразует его и передает его для выполнения совей локальной части, которая в свою очередь передает его соответствующему приложению. После того, как от приложения получен результат, сервер обращается к транспортной подсистеме и направляет ответ клиенту, выдавшему запрос. Клиентская часть ОС преобразует результат в соответствующий формат и адресует его тому приложению, которое выдало запрос. В зависимости от того, как распределены сетевые функции между компьютерами, сетевые ОС делятся на одно ранговые и двух ранговые или сети с выделенными серверами. Если компьютер предоставляет свои ресурсы другим пользователям сети, он выполняет функции сервера, а компьютер, обращающийся к его ресурсам, является клиентом. Если выполнение серверных функций является основным назначение компьютера, то он называется выделенным сервером. На выделенных серверах устанавливаются ОС. Оптимизированные для выполнения необходимых серверных функций. В сетях с выделенными серверами используются сетевые ОС, в состав которых входят несколько вариантов ОС, отличающихся возможностями серверных частей.

Распределенная ОС (РОС) – это совокупность независимых компьютеров, которая представляется пользователю как единая система. Распределенная ОС имеет единый глобальный меж процессный коммуникационный механизм, глобальную схему контроля доступа, одинаковое видение файловой системы и единую очередь процессов, ожидающих выполнения.

Многопроцессорные системы обладают рядом определенных свойств, которые отсутствуют в однопроцессорных вычислительных машинах. Например, в таких системах часто встречается удержание процессом Spin блокировки – циклической блокировки, или низкоуровневый примитив синхронизации, применяемый в многопроцессорных системах для реализации взаимного исключения, исполнения критических участков кода с использованием цикла активного ожидания. При этом другие процессоры, ждущие освобождение Spin блокировки, просто теряют время в циклах ожидания пока этот процесс не будет запущен снова и не отпустит блокировку. Чтобы решить данную проблему в некоторых системах применяется так называемое умное планирование, при котором процесс, захватывающий Spin блокировку, устанавливает флаг, демонстрирующий, что в данный момент он обладает Spin блокировкой. Когда процесс освобождает Spin блокировку, он также очищает и флаг. Таким образом, планировщик не останавливает процесс, удерживающий Spin блокировку, а наоборот, дает ему еще немного времени, чтобы тот завершил выполнение критической секции и, например, отпустил mutex.

Для увеличения частоты попадания в кэш используют родственное планирование. Основная идея этого метода заключается в том, чтобы процесс был запущен на том же процессоре, что и в прошлый раз. Один из способов реализации данного метода состоит в использовании двухуровнего алгоритма планирования. В момент создания процесс назначается конкретному процессору, например, наименее загруженному в данный момент. Такое назначение процессов процессорам представляет собой верхний уровень алгоритма. В результате каждый процессор получает свой набор процессов. Действительно планирование процессов находится на нижнем уровне алгоритма. Оно выполняется отдельно каждым процессором при помощи приоритетов или других средств. Старание удерживать процессы на одном и том же процессоре максимизирует родственность кэша. Однако, если у какого-либо процессора нет работы, у загруженного работой процессора отнимается процесс и отдается менее загруженным процессорам. Двухуровневое планирование обладает тремя преимуществами:

1. Оно довольно равномерно распределяет нагрузку среди имеющихся процессоров
2. Оно по возможности использует преимущества родственности кэша
3. у каждого процессора при таком варианте планирования есть свой список процессоров. Минимизируется конкуренция за списки свободных процессов

Еще один подход к планированию в многопроцессорных системах может быть использован, если процессы связаны друг с другом каким-либо способом. Также случается, что один процесс создает множество потоков, работающих совместно. Простейший алгоритм работает следующим образом:

создается группа связанных процессов. В момент их создания планировщик проверяет, есть ли свободные процессоры, по количеству создаваемых процессов. Если свободных процессоров достаточно каждому процессу выделяется процессор, и все процессы запускаются одновременно. Если процессоров недостаточно, то ни один из процессов не запускается, пока не освободится достаточное количество процессоров. Каждый процесс выполняется на своем процессоре вплоть до завершения. Если процесс оказывается заблокированным операцией ввода\вывода, он продолжает удерживать процессор, который простаивает до тех пор, пока процесс не сможет выполнять свою работу.

При использовании этого алгоритма в любой момент времени множество процессоров статически разделяется на несколько подмножеств, на каждом из которых выполняются процессы одной задачи.

**Миграция процессов** – перенос процессов с одного процессора на другой, возможность исполнения процесса на любом процессе дает множество преимуществ:

1. процессы можно переносить на слабо загруженные процессоры и таким образом, уменьшать время реагирования и повышать производительность и пропускную способность
2. возможность миграции процессов повышает отказоустойчивость. Если есть программа, которая должна без сбоев выполнят вычисления, а узел, на котором она выполняется требуется отключить или он работает не устойчиво, и полученные программой результаты могут быть потеряны, миграция процессов позволит программе переместиться на другой узел и продолжить вычисления
3. возможность миграции позволяет улучшить разделения ресурсов, т. к. в больших системах некоторые ресурсы не продублированы во всех узлах
4. возможность миграции процессов повышает эффективность взаимодействия. Два процесса, взаимодействующих друг с другом, должны выполнятся на одном узле или близко расположенных узлах, чтобы уменьшить задержки при взаимодействии. Поскольку каналы связи часто формируются динамически можно воспользоваться миграцией процессов и сделать размещение процессов тоже динамическим.

**Тупик в распределенных системах**

Тупики в распределенных системах подобны тупикам в централизованных системах, только их сложнее обнаруживать и предотвращать. Иногда выделяют особый вид тупиков в распределенных системах – коммуникационные тупики. для борьбы с тупиками в распределенных системах используют следующие стратегии:

1. полное игнорирование проблемы
2. обнаружение тупиков:
   1. централизованное обнаружение тупиков заключается в распространении идеи графа размещения ресурсов на распределенную систему. Координатор строит единый граф для всех узлов системы и на нем выполняет редукцию или приведение.
   2. распределенное обнаружение тупиков. в этом методе инициирование задачи обнаружение тупика начинает процесс, который подозревает что система находится в тупике. Он посылает тем процессам, которые держат нужный ему ресурс сообщение, состоящие из 3-х информационных полей: 1 поле содержит номер процесса, инициировавшего поиск, и оно не меняется при дальнейшей пересылке; 2 и 3 поля содержат соответственно номер процесса, который зависит от ресурса и номер процесса, который держит этот ресурс. Если в конце концов, сообщение вернется тому процессу, который инициировал поиск, то он приходит к выводу, что система зациклена, то есть в ней присутствует взаимоблокировка, и он должен предпринять определенные действия по ликвидации тупика.
3. предотвращение тупиков в распределенных системах базируется на идее упорядочивания процессов по их временным меткам. Допустимой будет ситуация, когда старший процесс, то есть запущенный позже, ждет ресурсы, которые захватил младший процесс, то есть процесс, запущенный раньше. В обратном случае младший процесс долен отказаться от ожидания.