«Моделирование управления в распределенных системах сетями Петри»

Тема: «Распределённые системы»

Распределенная система – это система, в которой обработка информации сосредоточена не на одной вычислительной машине, а распределена между несколькими компьютерами.

Существует шесть основных характеристик распределенных систем.

1. *Совместное использование ресурсов.* Распределенные системы допускают совместное использование как аппаратных (жестких дисков, принтеров), так и программных (файлов, компиляторов) ресурсов.
2. *Открытость.* Это возможность расширения системы путем добавления новых ресурсов.
3. *Параллельность.* В распределенных системах несколько процессов могут одновременно выполнятся на разных компьютерах в сети. Эти процессы могут взаимодействовать во время их выполнения.
4. *Масштабируемость***.** Под масштабируемостью понимается возможность добавления новых свойств и методов.
5. *Отказоустойчивость.* Наличие нескольких компьютеров позволяет дублирование информации и устойчивость к некоторым аппаратным и программным ошибкам. Распределенные системы в случае ошибки могут поддерживать частичную функциональность. Полный сбой в работе системы происходит только при сетевых ошибках.
6. *Прозрачность.* Пользователям предоставляется полный доступ к ресурсам в системе, в то же время от них скрыта информация о распределении ресурсов по системе.

Распределенные системы обладают и рядом недостатков.

1. *Сложность*. Намного труднее понять и оценить свойства распределенных систем в целом, их сложнее проектировать, тестировать и обслуживать. Также производительность системы зависит от скорости работы сети, а не отдельных процессоров. Перераспределение ресурсов может существенно изменить скорость работы системы.
2. *Безопасность*. Обычно доступ к системе можно получить с нескольких разных машин, сообщения в сети могут просматриваться и перехватываться. Поэтому в распределенной системе намного труднее поддерживать безопасность.
3. *Управляемость*. Система может состоять из разнотипных компьютеров, на которых могут быть установлены различные версии операционных систем. Ошибки на одной машине могут распространиться непредсказуемым образом на другие машины.
4. *Непредсказуемость*. Реакция распределенных систем на некоторые события непредсказуема и зависит от полной загрузки системы, ее организации и сетевой нагрузки. Так как эти параметры могут постоянно изменятся, поэтому время ответа на запрос может существенно отличаться от времени.

Из этих недостатков можно увидеть, что при проектировании распределенных систем возникает ряд проблем, которые надо учитывать разработчикам.

1. *Идентификация ресурсов*. Ресурсы в распределенных системах располагаются на разных компьютерах, поэтому систему имен ресурсов следует продумать так, чтобы пользователи могли без труда открывать необходимые им ресурсы и ссылаться на них. Примером может служить система URL (унифицированный указатель ресурсов), которая определяет имена Web-страниц.
2. *Коммуникация*. Универсальная работоспособность Internet и эффективная реализация протоколов TCP/IP в Internet для большинства распределенных систем служат примером наиболее эффективного способа организации взаимодействия между компьютерами. Однако в некоторых случаях, когда требуется особая производительность или надежность, возможно использование специализированных средств.
3. *Качество системного сервиса*. Этот параметр отражает производительность, работоспособность и надежность. На качество сервиса влияет ряд факторов: распределение процессов, ресурсов, аппаратные средства и возможности адаптации системы.
4. *Архитектура программного обеспечения*. Архитектура ПО описывает распределение системных функций по компонентам системы, а также распределение этих компонентов по

процессорам. Если необходимо поддерживать высокое качество системного сервиса, выбор правильной архитектуры является решающим фактором.

Задача разработчиков распределенных систем состоит в том, чтобы спроектировать программное и аппаратное обеспечение так, чтобы предоставить все необходимые характеристики распределенной системы. А для этого требуется знать преимущества и недостатки различных архитектур распределенных систем. Выделяется три типа архитектур распределенных систем.

1. *Архитектура клиент/сервер*. В этой модели систему можно представить как набор сервисов, предоставляемых серверами клиентам. В таких системах серверы и клиенты значительно отличаются друг от друга.
2. *Трехзвенная архитектура*. В этой модели сервер предоставляет клиентам сервисы не напрямую, а посредством сервера бизнес-логики.
3. *Архитектура распределенных объектов*. В этом случае между серверами и клиентами нет различий и систему можно представить как набор взаимодействующих объектов, местоположение которых не имеет особого значения. Между поставщиком сервисов и их пользователями не существует различий. Эта архитектура широко применяется в настоящее время и носит также название *архитектуры веб-сервисов*. Веб-сервис - это приложение, доступное через Internet и предоставляющее некоторые услуги, форма которых не зависит от поставщика (так как используется универсальный формат данных - XML) и платформы функционирования. В данное время существует три различные технологии, поддерживающие концепцию распределенных объектных систем. Это технологии EJB, CORBA и DCOM.

Тема: «Сети Петри»

В настоящее время математическим аппаратом анализа логических моделей является один из разделов теории графов – **теория сетей Петри**. Сети Петри впервые были введены Карлом Петри в 60-х годах прошлого столетия. С точки зрения общей теории они представляют собой *двудольный ориентированный мультиграф*. Далее даётся объяснение каждого из понятий, входящих в это определение. *Двудольный граф* – это граф, имеющий два типа вершин, причём вершины одного типа непосредственно могут соединяться только с вершинами другого типа. Применительно к сетям Петри, вершины одного типа носят название *позиции*, а другого – *переходы*. Обычно при графическом изображении сети Петри позиции обозначаются кружками, а переходы – отрезками.

Вершины любого графа соединяются посредством дуг. Если эти дуги имеют направление (представляются в виде стрелок), то граф называется *ориентированным*. Если допускается соединение вершин не одной, а несколькими дугами одинаковой направленности (такие дуги называются кратными), то граф является *мультиграфом*. Сеть Петри можно задать, нарисовав соответствующий ей граф. Однако это возможно лишь для относительно небольших (и, соответственно, достаточно простых) сетей. Сложные сети Петри часто задаются при помощи следующей четвёрки: $C = (P, T, I, O)$. Здесь C – структура Сети Петри; через P обозначено множество позиций, T – множество переходов. Символы I и O означают соответственно совокупности позиций, из которых осуществляются входы в переходы и выходы из них, т.е. таким образом задаётся направление дуг графа.

На рис. 1 приведён пример сети Петри, содержащей пять позиций и четыре перехода. Её описание четвёркой $C = (P, T, I, O)$ имеет следующий вид

$P = \{p_1, p_2\ p_3, p_4, p_5\}$, $T = \{t_1, t_2\ t_3, t_4\}$

$I(t_1) = \{p_1\}$, $O(t_1) = \{p_2, p_3, p_5\}$,

$I(t_2) = \{p_2, p_3, p_5\}$, $O(t_2) = \{p_5\}$,

$I(t_3) = \{p_3\}$, $O(t_3) = \{p_4\}$,

$I(t_4) = \{p_4\}$, $O(t_4) = \{p_2, p_3\}$.

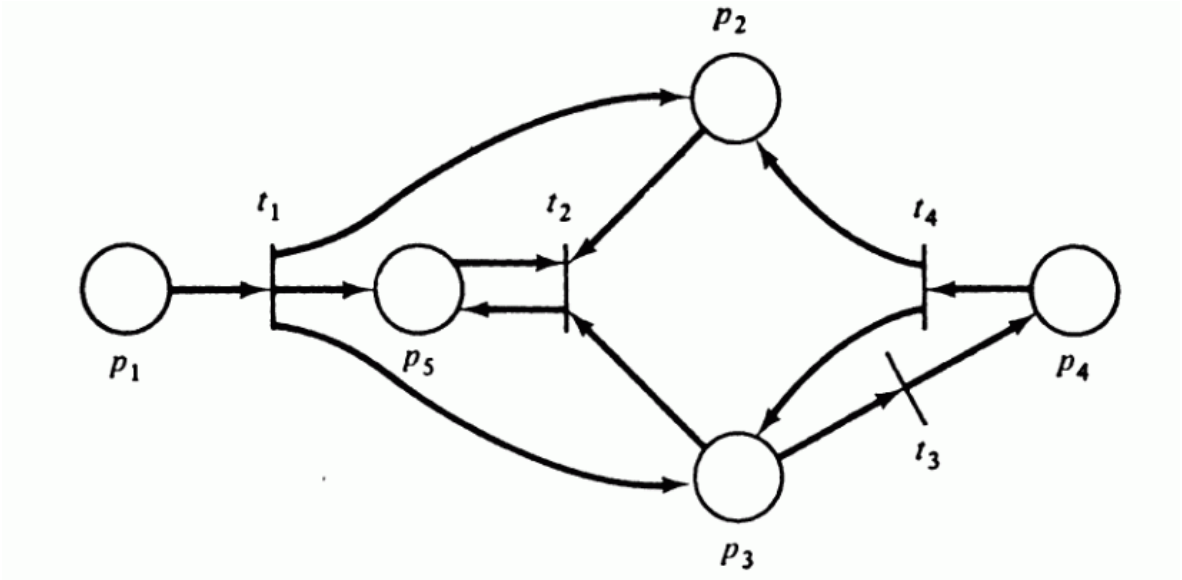


Рисунок 1: Пример сети Петри.

Для сети Петри вводится понятие *состояния*. Чтобы определить его рассмотрим следующую функцию $\mu : P \to N$, переводящую множество позиций во множество натуральных чисел *N*. Функция μ называется маркировкой сети Петри. Для наглядности на графе значения маркировки обозначаются точками, называемыми также *фишками*, внутри кружка, изображающего позицию. Например, если маркировка позиции p_2 равна трём, $\mu(p_2) = 3$, то в кружке, соответствующем p_2, ставится три фишки. Распределение фишек по позициям, т.е. маркировка, является состоянием сети.

Как и в любой другой системе, процесс в сети Петри определяется как смена её состояний. Таким образом, процессом в сети Петри является изменение маркировки. Оно происходит в результате запуска переходов, который осуществляется в соответствии со следующими правилами. При запуске перехода t_j из каждой его входной позиции забирается столько фишек, сколько дуг идёт из неё в рассматриваемый переход (эти фишки называются *разрешающими*). В выходные же позиции добавляются фишки в количестве, равном числу дуг, идущих из перехода в позицию. Например, при запуске перехода t_1 в сети Петри, изображённой на рис.1, из позиции p_1 будет удалена одна фишка, а в позиции p_2, p_3, p_5 добавлено по фишке.

Изменение значения маркировки в позиции p_i, вызванное запуском перехода t_j, определяется следующим *уравнением баланса фишек*:

$$\mu'(p_i) = \mu(p_i) - \{p_i, I(t_j)\} + \{p_i, O(t_j)\}, \qquad p_i \in P . \tag{1}$$

Здесь $\mu' = \delta(\mu, t_j)$ – новая маркировка в позиции p_i. Второй член в правой части (1) определяет убыль фишек во входных позициях, а третий – их прибыль в выходных позициях. Подчеркнём, что при запуске переходов фишки не перемещаются из входных позиций в выходные, а поглощаются в переходе и генерируются им в соответствии с числом входящих и выходящих дуг.

Тема: «Решение типовых задач управления в сетях Петри»

Задача о взаимном исключении. Под *взаимным исключением* здесь понимается метод разработки таких программ, в которых одновременно не более чем один процесс имеет доступ к разделяемому объекту данных. Участок кода, в котором осуществляется доступ к разделяемому объекту и который требует защиты от вмешательства других процессов, называется *критической секцией*.

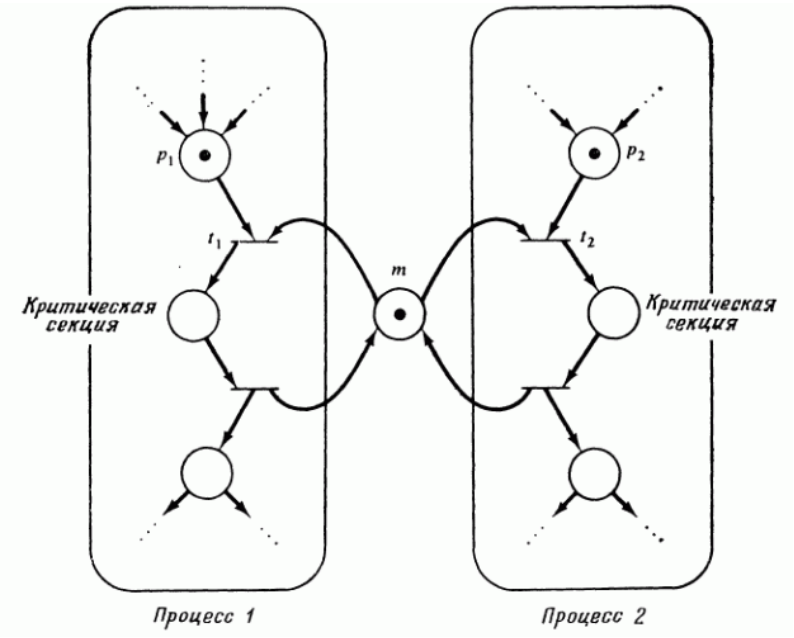
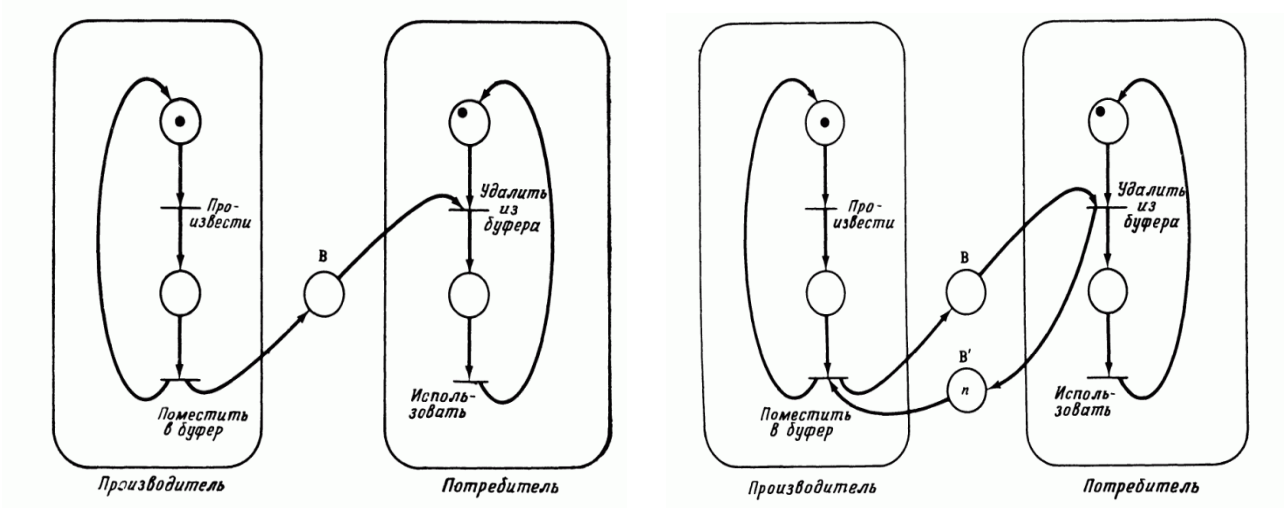


Рисунок 2: Решение задачи о взаимном исключении.

На рис. 2. представлено решение рассматриваемой задачи посредством сети Петри. Пусть, что процесс 1 входит в критическую секцию. Это означает, что запускается переход t_1, при этом забираются фишки из позиций p_1 и m. Для того, чтобы процесс 2 вошёл в критическую секцию, должен запуститься переход t_2. Однако он заблокирован, поскольку отсутствует фишка в позиции m. Появится она там только после того, как процесс 1 покинет критическую секцию.

Задача о производителе/потребителе. В задаче о производителе/потребителе также присутствует совместно используемый ресурс, но в этом случае он точно определен и является буфером. Процесс-производитель создает продукт, который помещается в буфер. Потребитель ждет, пока продукт не будет помещен в буфер, удаляет его оттуда и использует. Сеть, моделирующая эту задачу, представлена на рис. 3,*а*.



а) *б*)

Рисунок 3: Сеть Петри для задачи о производителе/потребителе (а), и её варианта с ограниченным буфером (б).

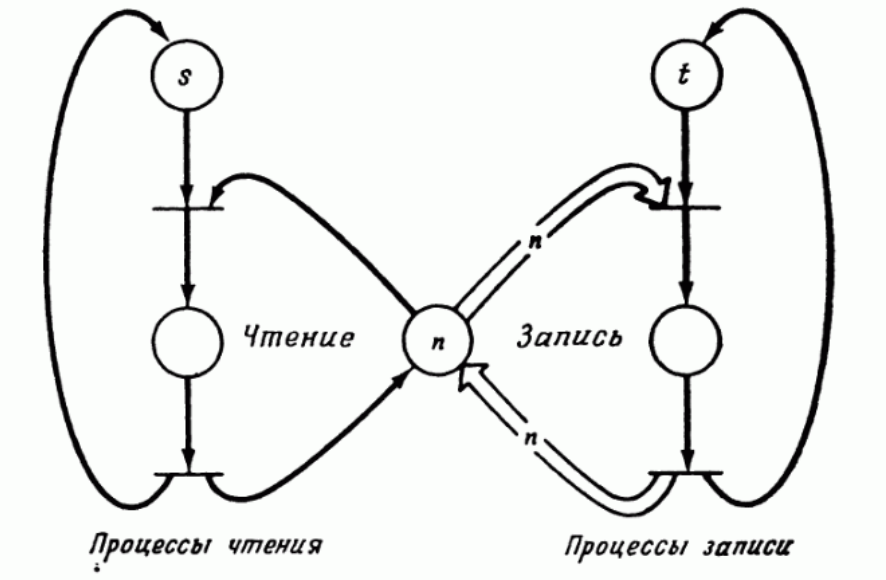


Рисунок 4: Задача о чтении/записи. Число процессов чтения ограничено величиной n. Первоначально имеются s процессов чтения и t процессов записи.

В другом варианте задачи используется буфер ограниченного размера. Следовательно, производитель не может постоянно работать с той скоростью, которая ему нужна, а вынужден ждать, если потребитель работает медленно и буфер заполнен. На рис. 3,*б* показано решение этой проблемы. Ограниченному буферу сопоставляются две позиции B и B', представляющие соответственно количество заполненных и пустых ячеек в буфере. Первоначально B' имеет n фишек, а B фишек не имеет. При бездействии потребителя, по мере работы производителя фишки будут удаляться из позиции B' и скапливаться в позиции B. Когда в B' не останется фишек переход, соответствующий событию “поместить в буфер” будет заблокирован, поскольку B' является его входной позицией.

Задача о чтении/записи. Существует несколько вариантов задачи о чтении/записи, однако их основная структура остается неизменной. Имеются процессы двух типов: процессы чтения и процессы записи. Все процессы совместно используют общий файл или переменную. Процессы чтения не изменяют объект в отличие от процессов записи. Таким образом, процесс записи должен исключать все другие процессы чтения и записи, в то время как несколько процессов чтения могут иметь доступ к разделяемым данным одновременно. Задача состоит в определении структуры управления, которая не приведет к тупику и не допустит нарушения критерия взаимного исключения. На рис. 4 иллюстрируется решение задачи в том случае, когда число процессов записи составляет t, а количество процессов чтения ограничено величиной n и равно $s \leq n$. Принцип исполнения сети на рис. 4 такой же, как и сети на рис. 2. Проблема возникает в том случае, если количество процессов чтения не ограничено. Это связано с тем, что в сетях Петри нет механизма, который бы осуществлял проверку на отсутствие фишек в неограниченной позиции.