ЛАБОРАТОРНАЯ РАБОТА № 5

ИССЛЕДОВАНИЕ МОДЕЛЕЙ ВЗАИМОДЕЙСТВИЯ РАСПРЕДЕЛЕННО ВЫПОЛНЯЮЩИХСЯ ПРОЦЕССОВ

Цель работы: исследовать алгоритмическое построение методов взаимодействия распределено выполняющихся процессов**.**

1. Теоретическое введение.

Наряду с известными моделями взаимодействия распределенных процессов (взаимодействие «производитель-потребитель», «взаимодействие равных», «клиент-сервер») существует целый ряд важных моделей, к которым могут быть отнесены следующие:

1) модель «зонд-эхо»;
2) модель «распределенны семафоров»;
3) модель передачи маркера.

1.1. **Модель «зонд-эхо»**

Топология вычислительного кластера (топология определяет взаимодействие вычислительных процессов), построенного для распределенных вычислений, может быть представлена в виде графа с узлами – процессами, и ребрами – каналами передачи данных. «Зонд» - это сообщение, передаваемое узлами графа своему преемнику, «эхо» - это последующий ответ преемника родительскому узлу. Модель «зонд-эхо» используется как для построения топологии кластера, так и для широковещательной рассылки сообщений по всем узлам (процессам) кластера (процесс, выполняющийся на узле S должен разослать сообщения процессам, выполняющимся на всех остальных узлах).

Основным требованием при широковещательной рассылке является необходимость того, чтобы сообщение попадало на каждый узел (в каждый процесс) только один раз. Для обеспечения этого требования узел, выполняющий рассылку, должен иметь два специальных средства, обеспечивающих эту рассылку – это информация о топологии кластера (взаимодействии распределено выполняющихся процессов) и об его (кластера) **остовном** дереве. Топология может быть представлена в виде симметричной матрицы, где элемент i-ой строки и j-го столбца равен 1, если i-ый и j-ый узлы связаны между собой каналами передачи данных, в противном случае – 0.

Топология строится на основе выполнения широковещательных рассылок между узлами кластера эхо-запросов и получения эхо-ответов (построение топологии рассматривается далее). Для эффективной рассылки узел должен построить и использовать остовное дерево. Корнем остовного дерева является узел, инициирующий рассылку. Остовное дерево отличается от топологии сети тем, что в нем исключены ребра топологии, дублирующие связи между узлами. Отличие остовного дерева от топологии кластера комментирует рис. 1.1.

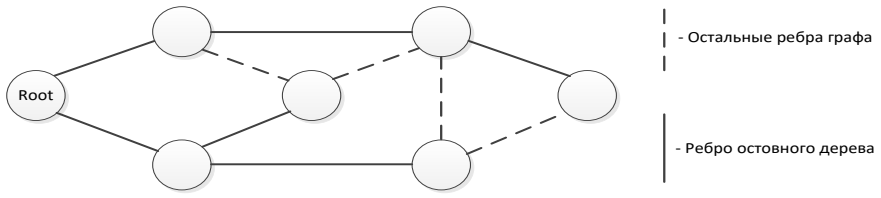

Рисунок 1.1 – Остовное дерево кластера

Остовное дерево интерпретируется соответствующей матрицей, построенной на основе матрицы топологии, из которой исключены единичные элементы, **дублирующие связи между узлами**.

Процесс рассылки широковещательных сообщений выполняется корневым узлом остовного дерева (т.е. процессом, реализующим широковещательную рассылку). При этом между узлами (процессами) кластера первоначально выполняется рассылка остовного дерева. Рассылка остовного дерева (предваряющая рассылку широковещательных сообщений) необходима для того, чтобы другие узлы топологии знали, куда им необходимо далее пересылать сообщение.

Пересылка сообщения между узлами остовного дерева осуществляется с использованием уже существующих каналов передачи данных, созданных предварительно для обмена сообщениями между процессами. Построению остовного дерева предшествует процедура построения топологии кластера, которая заключается в следующем. Каждому узлу известна лишь его локальная топология (т.е. связи с его соседями). Задача состоит в том, чтобы собрать воедино все локальные топологии и построить общую топологию кластера. Агоритм построения топологии кластера состоят в следующем:

1) каждый узел (начиная с инициатора рассылки) посылает сообщение-зонд (запрос на топологию) своим соседям. В зонде указывается источник этого зонда (Рис. 1.2);

Тип сообщения (зонд)	Идентификатор источника

Рисунок 1.2 – Формат зонда

2) узлы-преемники, получив зонд, ретранслируют его далее, но не посылают эхо-ответ до тех пор, пока не придёт эхо-ответ от их преемников (при ретрансляции указывается уже другой источник зонда);
3) так как источник зонда (начальный или последующий) знает, сколько у него каналов рассылки зондов, из них он и будет ожидать возврат эхо-ответов;
4) процесс рассылки зонда продолжается до тех пор, пока он не достигнет узлов, не имеющих соседей, либо, если топология циклическая, на узел не придет пустой эхо-ответ, что определяет отсутствие необходимости дальнейшей ретрансляции зонда;
5) в топологии с циклами, если узел получает последующие зонды во время ожидания эхо-ответа, он сразу отсылает пустой эхо-ответ на тот родительский узел, который послал этот зонд;
6) получив эхо от соседних узлов, узел объединяет эти ответы со своим множеством соседей и отсылает это сообщение родительскому узлу; таким образом, суммарное сообщение должно содержать некоторую часть топологии сети, идентифицируемую массивом топологии (его часть заполняется узлами по мере движения снизу вверх по сети)(Рис. 1.3);

Тип сообщения (эхо)	Источник ответа	Массив топологии

Рисунок 1.3 – Формат эхо-ответов

7) инициировавший рассылку узел обобщает эхо-ответы, поступившие от соседей, строит дерево смежности, остовное дерево и пересылает широковещательно сообщение по сети.

1.2. Модель «распределенные семафоры»

Данная модель позволяет осуществлять разделение общего ресурса между распределенными процессами при их выполнении. При реализации механизма распределенных семафоров можно не только разграничивать доступ к ресурсу, но и формировать очередь процессов, ожидающих этого доступа. Возможность организации очереди связана с понятием логических часов. Логические часы – это целочисленный счетчик, увеличивающийся при реализации события. У каждого процесса есть свой логический счетчик, имеющий нулевое начальное значение. В каждом передаваемом сообщении, используемом в алгоритме распределенных семафоров, выделено поле для временной метки. Изменение значения счетчика осуществляется в соответствии со следующими правилами:
1) передавая сообщение, процесс присваивает его метке текущее значение логических часов (переменная lc (localclock)) и увеличивает lc на 1;
2) получая сообщение с меткой времени ts, процесс присваивает своей переменной lc значение, максимальное из lc и $ts + 1$, затем увеличивает свою переменную lc на 1.

По аналогии со стандартными семафорами распределенные семафоры оперируют с P- и V- операциями. При синхронизации доступа к ресурсу процессы обмениваются сообщениями требуемого формата (Рис.1.4).

Идентификатор процесса	Дескриптор типа операции (P-или V-)	Метка времени

Рисунок 1.4 – Формат сообщения при синхронизации доступа к ресурсу

При рассылке сообщений, процесс их генерирующий, сохраняет их временные метки. Так как очереди P-операций всеми процессами обрабатываются одновременно, каждый процесс знает, когда он может занять общий ресурс (т.е. когда в очереди обрабатывается именно его операция P).

Алгоритм синхронизации процессов с помощью распределенных семафоров состоит из следующих шагов:
1) при необходимости доступа к ресурсу процесс рассылает всем остальным процессам сообщение с дескриптором P-операции и меткой времени; при этом процесс источник сообщения с P-операцией размещает в очереди процессов к ресурсу свой идентификатор, с которым связана соответствующая временная метка; отсылка сообщения приводит к изменению значения логических часов источника.
2) принятие широковещательного сообщения (с P-операцией) приводит к изменению логических часов на всех хостах (за исключением источника); одновременно идентификатор процесса, запросившего ресурс (с соответствующей временной меткой) устанавливается в общую очередь процессов на обработку этим ресурсом; очередь сообщений должна быть упорядочена в соответствии с временной меткой; процессы, принявшие широковещательные сообщения с P-операцией, подтверждают его принятие, отсылая источнику сообщение ask, которое не изменяет временных меток ни на одном из хостов;
3) приняв от каждого из процессов кластера ask-сообщения, процесс-инициатер захвата ресурса обращается к своей локальной очереди идентификаторов процессов на обработку ресурсом, аналогично выполнив передачу сообщения ask, хосты кластера также обрабатывают очередь идентификаторов процессов на доступ к ресурсу; из этой очереди каждым процессом извлекается первый идентификатор (из головы очереди); тот процесс, идентификатор которого извлечен из очереди (первый процесс в очереди) выполняет обращение к ресурсу, все остальные процессы удаляют этот идентификатор из своих очередей; для синхронизации доступа на каждом хосте выделяется локальная переменная S (переменная семафор), определяющая доступность ресурса; при захвате ресурса операции с семафором S аналогичны модели с общей памятью, при этом обращение к очереди процессов выполняется только в случае S=1 (первоначальная не занятость ресурса и освобождение ресурса при выполнении V-операции);

4) если переменная $S = 1$, то всеми хостами выполняется P-операция (соответствующая первому в очереди идентификатору процессов), которая переводит значение S в 0; идентификатор процесса, соответствующий этой операции, удаляется из очереди; при $S = 0$, процесс доступа к ресурсу не получает; он может получить доступ к ресурсу только в случае выполнения V-операции процессом, освобождающим ресурс;
5) если процесс освобождает ресурс, он формирует сообщение с V-операцией, которое подтверждается всеми принявшими его процессами (сообщения ask), после чего каждый процесс изменяет свое локальное значение « распределенного» семафора *S= S+1;* после этого каждый из распределенных процессов обращается к своей локальной очереди и если она не пуста, то выполняет ее обработку с выделением ресурса первому в этой очереди процессу.

Реализация алгоритма взаимодействия процессов с использованием распределенных семафоров рассмотрена на Рис.1.5-1.13.

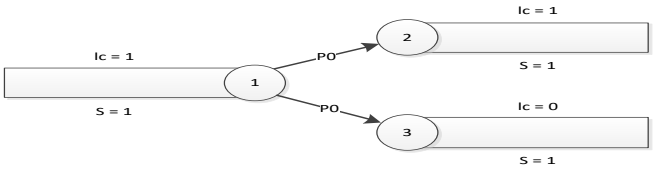

Рисунок 1.5 – Запрос захвата ресурса первым процессом

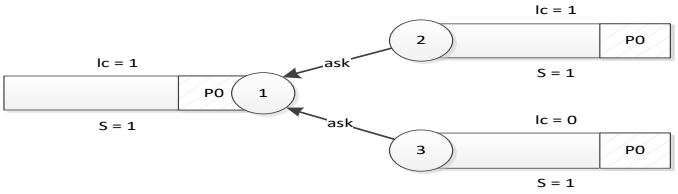

Рисунок 1.6 – Размещение сообщения P0 в очереди и отсылка подтверждающих сообщений ask(логические часы при принятии P-сообщения увеличиваются на 1)

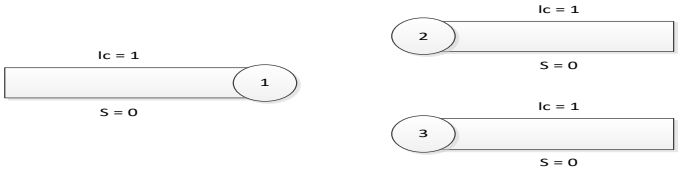

Рисунок 1.7 – Обработка очереди (P-операция), удаление P-операции из очереди, изменение значения переменной S

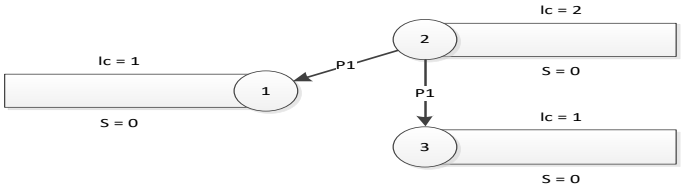

Рисунок 1.8 – Запрос вторым процессом ресурса. Отправка сообщений с идентификатором P-операции.

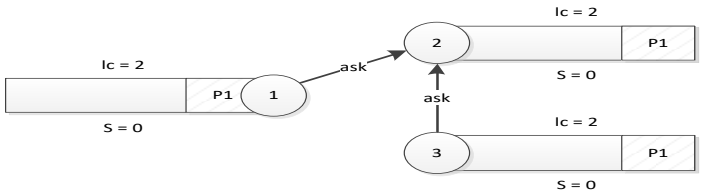

Рисунок 1.9 – Размещение полученного сообщения в очереди, подтверждение приемниками получения сообщения сигналом ask.

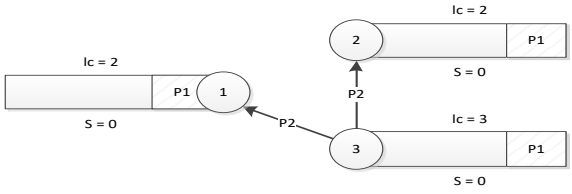

Рисунок 1.10 – Запрос третьим процессом ресурса. Отправка сообщения с идентификатором Р-операции.

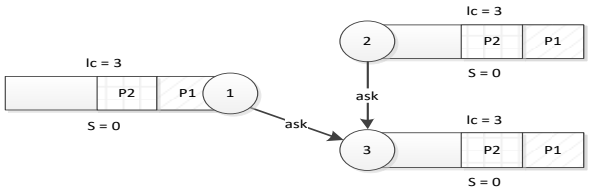

Рисунок 1.11 – Размещение Р-сообщения в очереди. Подтверждение принятия Р-сообщения

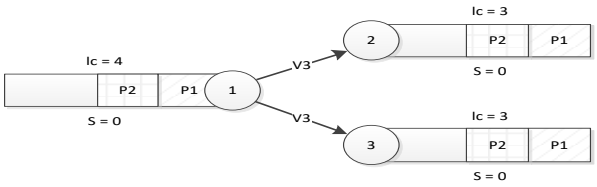

Рисунок 1.12 – Обработка третьим процессом значения переменной S. Посылка первым процессом сообщения V об освобождении ресурса.

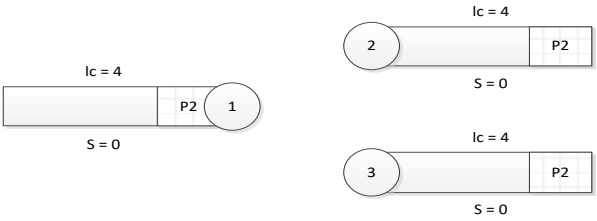

Рисунок 1.13 – Обработка очереди

Происходит изменение значения переменной S на 1, извлечениеоперации P1 (соответствующей второму процессу) из очереди, изменение переменной S на значение 0. Разрешение доступа процессу, сгенерировавшему операцию Р (второй процесс) доступа к ресурсу. Удаление операции P1, соответствующей второму процессу, использующему ресурс из очереди.

1.3. Модель передачи маркера

Алгоритм передачи маркера предназначен для синхронизации доступа распределенно выполняющихся процессов к общему ресурсу. Маркер – это сообщение специального вида, которое предназначено для передачи разрешения на доступ к ресурсу (маркер– сообщение, передача которого соответствует возможности доступа к общему ресурсу определенного процесса).

С каждым из распределено выполняющихся процессов, реализующих вычисления (процесс-*user),* сопоставлен специальный вспомогательный процесс, реализующий передачу маркера и синхронизирующий доступ к ресурсу (процесс-*helper*). Вспомогательные процессы образуют кольцо, т.е. первый процесс передает маркер второму процессу, он – третьему и т.д., N-ый процесс передает первому процессу. Алгоритм распределения ресурса с использованием передачи маркера комментирует рис. 1.14.

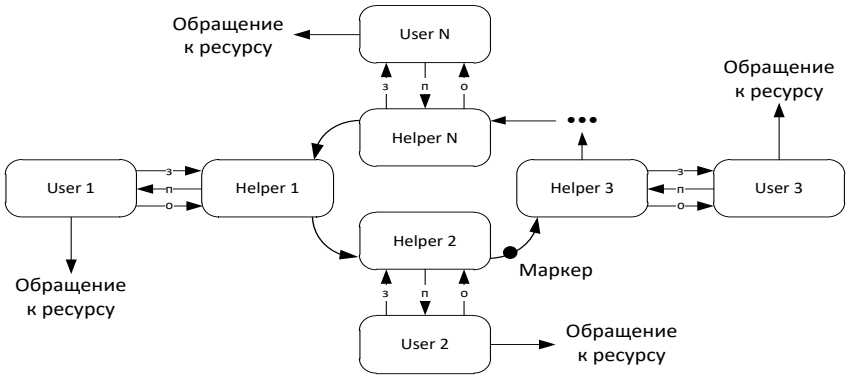

Рисунок 1.14 – Реализации алгоритма передачи маркера

На рис 1.14 использованы следующие обозначения:

- User – вычислительный процесс,
- Helper – вспомогательный процесс,
- З – запрос ресурса,

– П – подтверждение возможности использования ресурса,
– О – сигнал освобождения ресурса.

Алгоритм управления доступом процессов к ресурсу с использованием передачи маркера состоит из следующих шагов:

1) приняв маркер, процесс **`Helper[i]`** проверяет наличие запроса от вычислительного процесса на доступа к ресурсу; если процесс **`User[i]`** выполнил запрос, то процесс **`Helper[i]`** далее маркер не передает, выдает процессу **`User[i]`** разрешение на использование ресурса и ждет от него сигнала об освобождении;
2) в том случае, если процесс **`User[i]`** не осуществил запрос на использование ресурса, процесс **`Helper[i]`** передает маркер (т.е сообщение без аргумента) процессу **`Helper[i+1]`**.

Таким образом, реализуется механизм исключительного доступа к ресурсу лишь одного из группы процессов с использованием единственного сообщения маркера.

2. Задание на работу.

2.1. Вариант №1

Осуществить построение топологии кластера требуемого вида (рис. 2.1); выполнить широковещательную рассылку вводимого с клавиатуры сообщения от узла S на все остальные узлы. На узле, инициирующем рассылку, выводить (в виде матрицы) топологию и остовное дерево, на остальных хостах кластра после получения сообщения выводить номер хоста и сам текст сообщения.

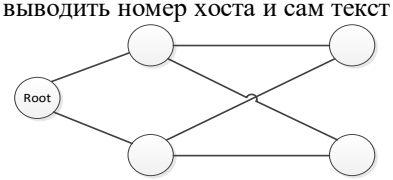

Рисунок 2.1 – Схема каналов взаимодействия процессов в кластере

2.2. Вариант №2

Распределено выполняются три процесса, каждый из которых вводит данные (целые числа) из локального файла по 5 элементов, выполняет суммирование 5 элементов и помещает полученную сумму в выходной файл. При этом все три процесса используют общую процедуру суммирования (процедура является общим разделяемым ресурсом), в которую передают массив элементов, обратно получают сумму. Выполнить синхронизацию доступа трех процессов к общему ресурсу (процедуре суммирования), используя модель распределенных семафоров.

2.3. Вариант №3

Реализовать задание второго варианта, используя модель передачи маркера. Выводить сообщения, комментирующие синхронизацию каждого вычислительного процесса (запрос и получение ресурса, освобождение ресурса), а также получение маркера в кольце вспомагательных процессов.

3. Контрольные вопросы.

3.1. Назовите особенности реализации и использования рассматриваемых моделей взаимодействия распределенных процессов.
3.2. Сформулируйте понятие топологии кластера и остовного дерева, определите форматы рассылаемых сообщений, алгоритмы построения топологии кластера и остовного дерева, алгоритм рассылки.
3.3. Сформулируйте алгоритм реализации механизма «Распределенных семафоров», назначение логических часов и очереди сообщений, определите форматы передаваемых сообщений.
3.4. Сформулируйте алгоритм реализации модели «передачи маркера».