A. O. Vendik, A. V. Kren, T. G. Fomicheva
Saint Petersburg Electrotechnical University

AUTOMATED SYSTEM OF EDUCATIONAL PROCESS ORGANIZATION AT THE DEPARTMENT OF PHYSICAL EDUCATION AND SPORTS

An automated system developed at the Department of Computer Engineering and transferred to experimental operation at the Department of Physical Education and Sports, which makes it possible to simplify the work of the administration of the department in organizing the educational process, is described. The system is aimed at users of three categories – administrator of the department, teacher and student. It is aimed at the solution of the following tasks: formation of training groups on sports disciplines provided by the department, monitoring of attendance, ongoing monitoring of students' progress, and the formation of final semester assessments in the discipline «Physical Culture» for each student. Despite the fact that the system is being developed for the Department of Physical Education and Sports of Saint Petersburg Electrotechnical University them. VI Ulyanov (Lenin), it can serve as a basis for creating similar systems for other universities. The system is implemented as a web application based on the Content Management System (CMS) of WordPress, MySQL database management system (DBMS) is used for data storage. The system does not require the installation of additional software. To use the system, you need access to the Internet.

Computer-aided information system, database, database management system, web-application, content management system

УДК 004.9

В. С. Новопашин
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Организация распределенной общей памяти

Рассмотрена организация распределенной общей памяти, ее преимущества и недостатки. Проведено сравнение с методом передачи сообщений. Показаны способы обеспечения согласованности памяти каталога, установлены перспективные способы организации пространства памяти, организации доступа к адресной обработке данных компьютерными методами организации доведения и обработки информации. Рассмотрено применение динамических библиотек и механизмы копирования записей библиотек. Рассмотрены способы реализации распределенной общей памяти аппаратными и программными средствами, при реализации памяти программными средствами – использование виртуальной памяти при страничном подходе, применение процедур для организации доступа к разделяемым переменным при использовании метода разделяемых переменных, а также организация доступа к общим данным с помощью объектно-ориентированной дисциплины при объектном подходе. Рассмотрены принципы согласованности памяти для отслеживания и поддержания состояния блоков данных в узлах в памяти, составляющей систему. В базовом варианте будет отслеживаться как минимум три состояния между узлами для любого рассматриваемого блока в каталоге.

Распределенная память, общая память, распределенная общая память, методы организации, межпроцессорное взаимодействие, репликация чтения и записи

Под распределенной общей памятью (DSM – Distributed Shared Memory) подразумевается такая форма архитектуры памяти, в которой имеется возможность обратиться к физически разделенной памяти как к единому адресному пространству, разделенному логически. В этом случае под словом «общая» подразумевается не существование единой централизованной памяти, а то, что ее адресное пространство общее. Подобная архитектура означает, что на всех процессорах системы одно и то же место в памяти будет иметь один физический адрес. Поэтому существует такое понятие, как распределенное глобальное адресное пространство (DGAS – Distributed Partitioned Global Address Space). И в данном случае это понятие будет иметь примерно одинаковый смысл в довольно широком спектре как программных, так и аппаратных реализаций. В свою очередь, каждый узел такой системы не только име-

ет собственную неразделенную память, но и может получать доступ к общей памяти.

На основе выполненных теоретических и прикладных исследований системных связей и закономерностей функционирования информационных систем при обмене данными и организации доступа к данным [1], [2] были установлены перспективные способы организации пространства памяти, доступа к адресной обработке данных компьютерными методами доведения и обработки информации. Для формирования системных связей при обмене данными были установлены перспективные способы межпроцессорного взаимодействия и сохранения пространств памяти [3], [4], направления доступа к данным, способы представления данных, выполнения вычислительных задач различными архитектурами процессоров.

В аппаратном обеспечении общая память – это блок оперативной памяти (RAM – Random Access Memory), чаще всего очень большой, к которому могут обращаться несколько различных центральных процессоров (CPU – Central Processing Unit) в многопроцессорной компьютерной системе. На рис. 1 показана система с общей памятью, состоящая из трех процессоров.

В подобных системах могут применяться различные виды архитектуры, а именно:

– неравномерный доступ к памяти (NUMA – Non-Uniform Memory Access);

– унифицированный доступ к памяти (UMA– Uniform Memory Access);

– архитектура только кэш-памяти (COMA – Cache-Only Memory Architecture).

Архитектура NUMA подразумевает организацию неоднородного доступа к памяти, так как время доступа к той или иной части памяти может варьироваться в зависимости от расположения процессора относительно этой памяти, т. е. доступ к удаленным модулям памяти будет происходить медленнее, нежели к локальным.

В отличие от архитектуры с неравномерным доступом к памяти, в архитектуре с унифицированным доступом физическая память используется всеми процессорами совместно, так как в ее основе лежит одна шина. Соответственно, прежде чем процессор сможет считать информацию из памяти, ему необходимо сначала проверить, свободна ли шина, и только если она свободна – производить какие-либо действия. В противном случае процессор вынужден ожидать ее освобождения. Очевидно, что при малом количестве процессоров доступ к шине легко управляем, однако при большом числе процессоров от пропускной способности шины будет зависеть производительность всей системы, а процессоры могут довольно часто простаивать.

В то же время, архитектура только кэш-памяти подразумевает, что у каждого процессора его основная память используется как кэш-память. Поэтому при подобной организации памяти у страниц не имеется собственных фиксированных машин.

С точки зрения программирования система общей памяти относительно проста, так как всеми процессорами совместно используется одно представление данных, а связь между процессорами, в свою очередь, не уступает по быстродействию доступу к памяти, сосредоточенной в одном месте. Однако для систем с общей памятью проблемой становится то, что многим процессорам необходим достаточно быстрый доступ к памяти и, скорее всего, наличие кэш-памяти, что обусловливает следующие сложности:

– снижается время доступа, так как при попытке получения доступа к одной ячейке памяти возникают конфликты, а при попытке доступа к смежным ячейкам может произойти ложный обмен. С точки зрения масштабируемости компьютеры с общей памятью недостаточно хороши, именно поэтому в большинстве таких компьютеров имеется не более десяти процессоров;

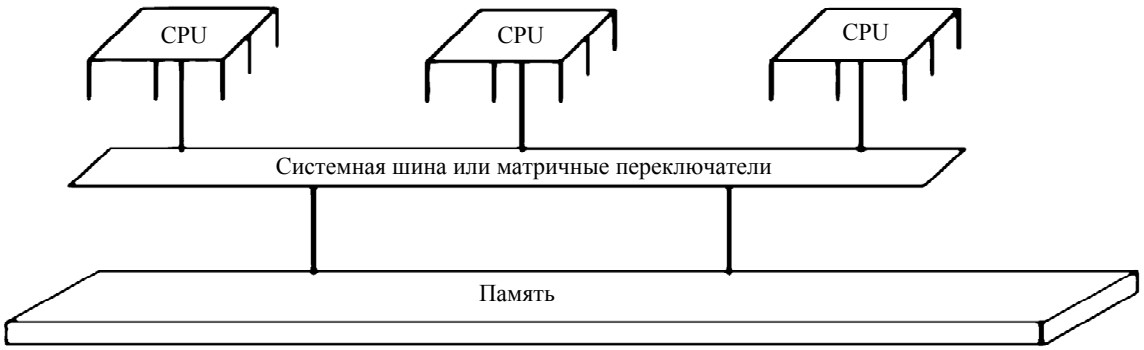

Рис. 1

– данные не согласованы, поэтому каждое обновление кэша информацией, которая в этот момент может использоваться другим процессором, необходимо отражать на других процессорах. В противном случае процессорам придется работать с несогласованными данными. Подобные протоколы обеспечения когерентности кэша при должной реализации способны сделать доступ к совместной информации нескольких процессоров чрезвычайно высокопроизводительным; в то же время, при высоких нагрузках они могут стать узким местом для производительности.

Для устранения таких узких мест можно использовать системную шину (FrontSideBus), сети Omega, матричные переключатели или же Hyper Transport.

Бывают случаи, когда процессорная архитектура может объединять с общей памятью один тип процессоров, а разные, как, например, графические и центральные, процессоры. Подобная системная архитектура называется гетерогенной. В ней и блок управления памятью ввода-вывода графического процессора, и блок управления памятью центрального процессора должны иметь некоторые определенные характеристики – к примеру, наличие общего адресного пространства.

Рассмотрим, что в таком случае подразумевается под общей памятью с точки зрения программного обеспечения:

– способ, когда обмен данными организовывается между одновременно работающими программами называется межпроцессорным взаимодействием (IPC – Inter-Process Communication). В этом случае один из процессов создает область в оперативной памяти, к которой впоследствии могут обращаться другие процессы;

– метод, при котором доступ направляется не к самим фрагментам данных, а к их копиям с явной поддержкой программы или отображением виртуальной памяти называется методом сохранения пространства памяти (XIP – execution-in-place, с выполнением на месте). Применение подобного метода часто можно наблюдать при реализации общих библиотек, либо при выполнении на месте.

В случае если процессы, совместно использующие память, работают на отдельных процессорах, а базовая архитектура не согласована с кэшем, могут возникнуть некоторые проблемы. Для их избегания необходимо с осторожностью выполнять все процессы и производить их на одной машине, что, в свою очередь, несколько усложняет масштабирование такой системы. В то же время, такой способ связи – очень быстрый, так как все процессы могут обращаться к области общей памяти. Другие механизмы IPC, например CORBA или доменные сокеты Unix, не могут похвастаться схожим быстродействием.

Динамические библиотеки, как правило, хранятся в памяти один раз и сопоставляются с несколькими процессами, что позволяет дублировать только страницы. Страницы, в свою очередь, настраиваются для отдельного процесса. Это связано с тем, что символ может разрешаться в разных страницах по-разному. Реализуется это с помощью прозрачного копирования при записи, т. е. сперва при попытке записи происходит копирование страницы и только потом разрешается запись в личную копию.

Под распределенной памятью обычно понимается многопроцессорная система, где у каждого процессора имеется его личная память. В связи с этим вычислительные задачи имеют возможность использовать только локальные данные, а при необходимости получения удаленных данных этой задаче потребуется взаимодействовать с нужными для ее выполнения процессорами. В противоположность такому поведению можно поставить мультипроцессор, имеющий общую память, так как в нем все имеющиеся процессоры используют общее пространство памяти. В таком случае процессоры могут даже не иметь сведений о том, где находятся данные, однако следует избегать возникновения условий гонки, а также возможны некоторые проблемы с производительностью.

Система с распределенной памятью, как правило, состоит из процессора и памяти, и между всеми процессорами имеется тот или иной способ общения, который позволяет программам различных процессоров взаимодействовать между собой. Соединение, обеспечивающее связь между процессорами, может быть реализовано через коммутационную сеть, которая будет обеспечена с помощью отдельного оборудования, либо через двухточечные соединения. В подобном случае при определении масштабирования такой системы ключевым фактором может служить топология сети. В свою очередь, связи между узлами можно реализовать несколькими способами:

– с применением сетевых ссылок на заказ;

– с применением двухпортовой памяти;

– с помощью одного из стандартных сетевых протоколов, например Ethernet.

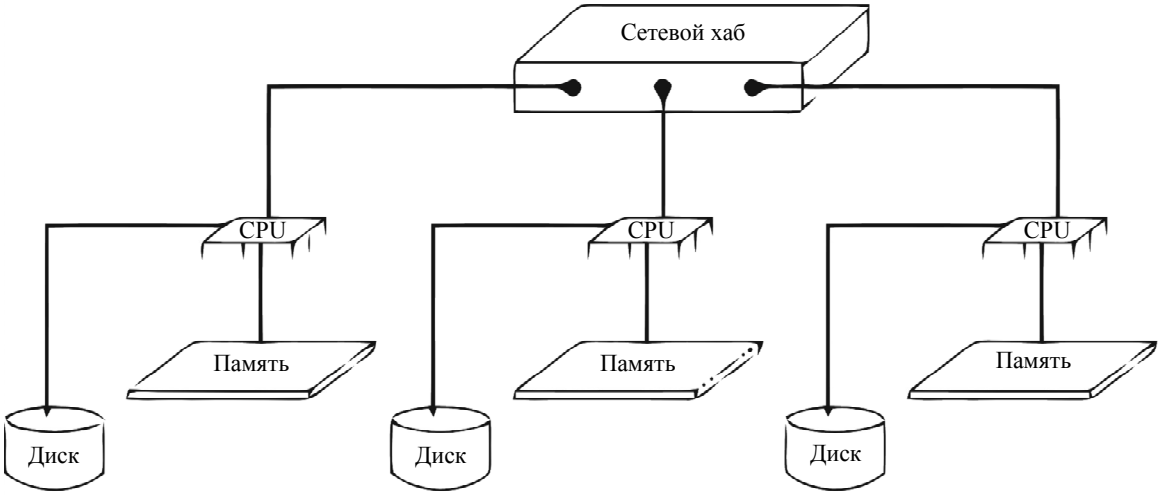

Рис. 2

На рис. 2 изображена система, имеющая распределенную память и состоящая из трех компьютеров.

Вопрос о том, каким образом лучше всего распределить данные по имеющейся памяти, – один из ключевых при программировании систем, которые используют распределенную память. В зависимости от решаемой проблемы данные могут распределяться статически или перемещаться по узлам. Данные могут быть перемещены по требованию или переданы на новые узлы заранее.

Например, если проблему можно описать как конвейер, в котором данные x обрабатываются впоследствии с помощью функций f, g, h и т. д. (результат равен $h(g(f(x)))$), то это можно выразить как проблему распределенной памяти, когда данные передаются сначала узлу, который выполняет f, который передает результат во второй узел, который вычисляет g, и, наконец, в третий узел, который вычисляет h. Такое преобразование также известно как систолическое вычисление.

Данные могут храниться статически в узлах, если большинство вычислений происходит локально, и только изменения на ребрах должны сообщаться другим узлам. Примером этого служит моделирование данных с использованием сетки, где каждый узел моделирует небольшую часть более крупной сетки. На каждой итерации узлы информируют все соседние узлы о новых данных ребра.

Довольно часто под мультикомпьютерной системой понимают такую, где существует распределенная общая память, которая в свою очередь имеет несколько независимых узлов и у каждого такого узла есть собственные модули памяти и между этими модулями имеется некоторая общая сеть. Реализация систем, в которых используется DSM, может быть как на уровне операционной системы, так и в виде некоторой программной библиотеки, которую можно рассматривать как расширение, применяемое к базовой архитектуре виртуальной памяти. Распределенная память может быть реализована с применением как программных, так и аппаратных средств – например, сетевых интерфейсов или схем согласованности кэша. Программные же средства могут подразумевать следующие подходы:

– страничный, при котором используется виртуальная память;

– разделяемых переменных, когда для доступа к этим переменным используются процедуры;

– объектный, при использовании которого с помощью объектно-ориентированной дисциплины осуществляется доступ к общим данным.

DSM-системы (подпрограммные) в качестве преимущества рассматривают гибкость в организации области общей памяти. Разделяемую память в качестве страниц с нединамическим размером позволяет осуществить подход на основе организации страниц. Так как объектно-ориентированный подход работает по другому принципу, он организует область разделяемой памяти как некоторое абстрактное пространство для хранения разделенных объектов, имеющих динамические размеры. Из этого вытекают проблемы, связанные с распределением доступа между потоками данных, которые не блокируют синхронизацию с масштабируемостью по количеству процессоров. Еще одна весьма распространенная реализация использует пространство кортежей, где общий элемент – кортеж.

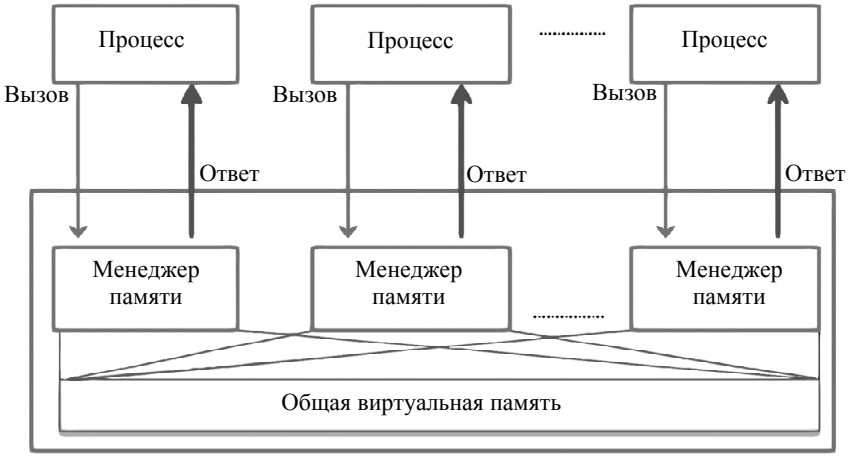

Рис. 3

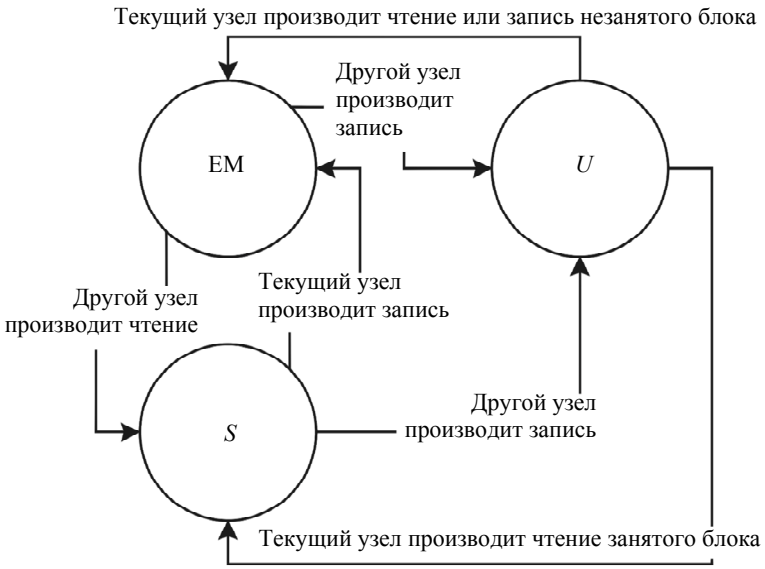

Рис. 4

Архитектура общей памяти может включать разделение памяти на несколько частей, которые распределены между узлами и основной памятью, или распределение всей памяти децентрализовано только между узлами. В соответствии с выбором последовательной модели протокол согласованности поддерживает когерентность памяти. На рис. 3 можно увидеть представление системы с распределенной общей памятью.

Из плюсов использования архитектуры с распределенной общей памятью можно выделить следующие:

– при наличии большого числа узлов не теряется легкость масштабирования;

– передача сообщений скрыта;

– появляется возможность обработки больших и сложных баз данных без репликации или отправки данных в процесс;

– использование многопроцессорной системы в таком случае обошлось бы дороже;

– виртуальная память становиться фактически не ограниченной;

– поддержка программ становится проще из-за общих интерфейсов программирования;

– программисты защищены от отправки или получения примитивов.

Однако имеются и некоторые минусы использования архитектуры распределенной общей памяти:

– как правило, доступ более медленный, чем к нераспределенной общей памяти;

– необходима дополнительная защита от одновременного доступа к общим данным.

Схема работы узлов распределенной общей памяти (запись и чтение) и диаграмма состояний блока памяти в DSM представлены на рис. 4.

Блок «присвоен», если один из узлов имеет блок в состоянии EM.

Для обеспечения системе, организующей DSM, возможности отслеживания и поддержания состояния блоков данных в узлах памяти, из которых и состоит эта система, как раз и необходима согласованность памяти.

Базовый DSM будет отслеживать как минимум три состояния между узлами для любого данного блока в каталоге, обозначающие блок как: 1) некэшированный (*U*), 2) занятый (EM), 3) общий (*S*). Когда блоки попадают в организацию каталогов, они переходят из состояния *U* в состояние EM в начальном узле, затем состояние может перейти в *S*, когда другие узлы начнут читать блок.

В системе, в которой упор делается на использование домашнего узла, существует необходимость обрабатывать запросы и ответы между самими узлами. Ее можно избежать благодаря использованию DSM, что позволит выполнять по одной транзакции за раз, пока домашним узлом не будут получены ответы от всех имеющихся процессов, после чего он примет решение о том, что транзакция была завершена.

Преимущества данного подхода:

– невозможность «гонки данных»;

– простота реализации.

Недостаток:

– медленная буферизованная стратегия запроса-ответа, ограниченная домашним узлом.

В системе, ориентированной на запрос, DSM позволит узлам по желанию общаться друг с другом через домашний узел. Это означает, что несколько узлов могут пытаться запустить транзакцию, но это требует дополнительных вычислений для обеспечения согласованности.

Преимущество данного подхода:

– быстр в использовании.

Недостатки:

– не препятствует «гонке данных»;

– генерирует больше трафика.

DSM должен следовать определенным правилам модели согласованности, показанной на рис. 5, системы для обеспечения согласованности того, как организуется порядок чтения и записи среди узлов Op (1, 2, ... *n*).

Рассмотрим ситуацию, когда имеется некоторое количество процессов, обозначим это количество *N*, а количество операций с памятью, выполняемых для каждого *i*-го процесса, – O_i. В таком случае, если считать, что все операции выполняются последовательно, можно вычислить общее количество возможных операций. В итоге получим следующую формулу:

$$(O_1 + O_2 + \ldots + O_N)! / (M_1!\, M_2! \ldots M_N!).$$

Однако, ввиду того, что разрешенные чередования определяются согласованностью памяти для DSM, возникает проблема в определении правильности чередующихся операций.

Рассмотрим следующие типы алгоритмов репликации:

– репликация записи. В этом случае несколько различных узлов могут одновременно производить как чтение, так и запись;

– репликация чтения. Отличается от репликации записи тем, что хотя несколько различных узлов по-прежнему могут производить чтение одновременно, записывать что-либо может только один из них. Сами же запросы на запись обрабатываются некоторой прикладной программой или же аппаратным устройством.

Из преимуществ, которые может обеспечить репликация данных, стоит выделить следующие:

– уменьшение количества ошибок;

– уменьшение сетевого трафика;

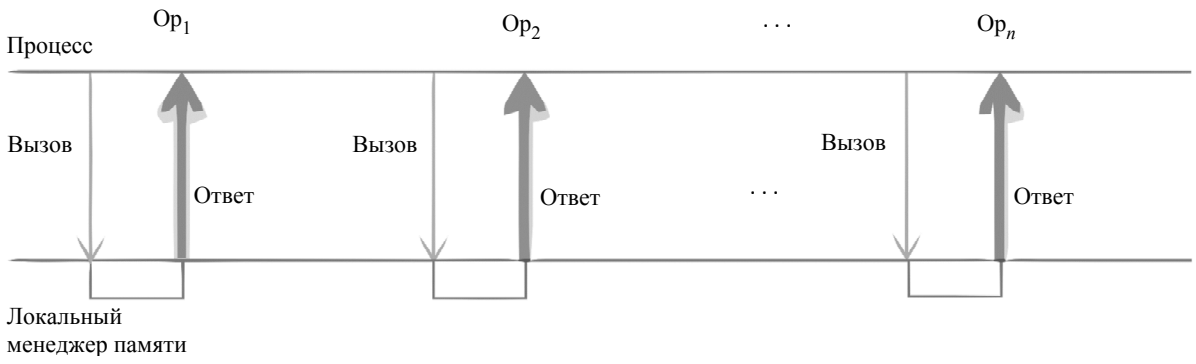

Рис. 5

– увеличение степени параллелизма.

В то же время, сохранение последовательности, как и согласованности, может превратиться в несколько более сложную задачу.

Таким образом, в ходе выполненного исследования были установлены перспективные способы организации пространства памяти, доступа к адресной обработке данных компьютерными методами организации доведения и обработки информации. Для организации системных связей при обмене данными установлены перспективные способы межпроцессорного взаимодействия и сохранения пространств памяти, рассмотрена организация направления доступа к данным, способы представления данных, организации вычислительных задач различными вариантами архитектурной организации процессоров.

Преимуществом общей памяти служит наличие некоего единого адресного пространства, в котором можно найти все данные.

Преимуществом распределенной памяти можно назвать ее способность исключить условия гонки и заставить программиста задумываться о распределении данных.

Преимущество распределенной общей памяти заключается в более легком проектировании машины, которая масштабируется с помощью алгоритма.

Распределенная общая память скрывает механизм общения, но не скрывает задержки общения.

СПИСОК ЛИТЕРАТУРЫ

1. Introduction to Data Mining / P-N. Tan, M. Steinbach, A. Karpatne, V. Kumar // Michigan Staite University, 2019.

2. Обзор и сравнение существующих методов управления доступом. URL: http:www.ict.nsc.ru/ws/YM2003/6312/ (дата обращения 22.05.2019).

3. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015.

4. Андреев А. М., Можаров Г. П., Сюзев В. В. Многопроцессорные вычислительные системы: теоретический анализ, математические модели и применение. М.: Изд-во МГТУ им. Н. Э. Баумана, 2011.

5. Solihin Yan. Fundamentals of Parallel Multicore Architecture. Boca Raton. Florida: Chapman and Hall CRC, 2015. P. 339–340.

6. El-Rewini H., Abd-El-Barr M. Advanced computer architecture and parallel processing. Wiley-Interscience, 2005. P. 77–80.

7. Sorin D. J., Hill M. D., Wood D. A. A Primer on memory consistency and cache coherence. Morgan & Claypool, 2011. P. 173–174.

V. S. Novopashin
Saint Petersburg Electrotechnical University

ORGANIZATION OF DISTRIBUTED SHARED MEMORY

The organization of distributed shared memory, its advantages and disadvantages are considered. A comparison is made with the message transmission method. The ways of ensuring the consistency of the catalog memory are shown, promising ways of organizing memory space, organizing access to address data processing by computer methods of organizing information delivery and processing are established. The use of dynamic libraries and mechanisms for copying library records is considered. The ways of implementing distributed shared memory by hardware and software are considered. When implementing memory by software, the page approach using the system's virtual memory, the method of shared variables using procedures for accessing shared variables, and the object approach with access to shared data through an object – oriented discipline are considered. The principles of memory consistency for tracking and maintaining the state of data blocks in nodes in the memory that makes up the system are considered. The basic version will track at least three States between nodes for any block under consideration in the directory.

Distributed memory, shared memory, distributed shared memory, organization methods, interprocessor interaction, read and write replication