ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆԻ ԳԻՏԱԿԱՆ ՏԵՂԵԿԱԳԻՐ
УЧЕНЫЕ ЗАПИСКИ ЕРЕВАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Բնական գիտություններ 1,2001 Естественные науки

Математика

УДК 681.3.068

И.Е. БОЯХЧЯН, С.К.ШУКУРЯН

НАСТРОЙКА ЛОКАЛЬНЫХ МУЛЬТИКОМПЬЮТЕРОВ НА ЗАДАЧИ ПОЛЬЗОВАТЕЛЕЙ

В статье рассматривается проблема отображения задач пользователей на компьютерные системы. Приводятся обзор существующих средств настройки на разных уровнях представления компьютерной системы, а также их сравнительный анализ по ряду характеристик. В качестве компьютерных систем рассматриваются локальные мультикомпьютеры, называемые также кластерами рабочих станций.

Проблемы отображения задач пользователя на компьютерную систему. Компьютерная система естественным образом может быть представлена в виде многоуровневой иерархической структуры, каждый уровень которой соответствует некоторой виртуальной или реальной системе, например, архитектуре, операционной системе, среде программирования и т.д. Такое представление приводит к проблеме нахождения оптимального представления для каждой конкретной задачи пользователя в рамках данного уровня, то есть наиболее полного использования свойств и возможностей уровня. Эта проблема может быть решена путем адаптации уровня к конкретной поставленной задаче за счет добавления новых (программных и аппаратных) возможностей или модификаций имеющихся.

С каждым уровнем компьютерной системы обычно сопоставляется следующая тройка: ***поведение, структура и реализация*** [1, 2]. ***Поведение*** определяет интерфейс взаимодействия компьютерной системы с внешней средой, ***структура*** задает множество логически взаимосвязанных компонентов, которые составляют систему, а ***реализация*** представляет собой описание физического строения уровня. Например, для архитектурного уровня ***поведением*** является алгоритм/множество алгоритмов функционирования компьютерной системы, отображающих входные данные в выходные, ***структурой*** – множество процессоров, память, коммуникационные и сетевые каналы, внешние устройства и т.д., а ***реализацией*** – множество микросхем, плат, кабелей и т.д.

Задача или проблема, отображаемая на компьютерную систему, в конечном счете должна быть представлена в виде многоуровневой иерархической структуры, каждому уровню которой соответствует другая тройка – ***поведение, структура*** и ***программно-аппаратная реализация***.

Сформулируем две важные проблемы, связанные с отображением данной задачи на компьютерную систему [3].

Проблема 1. Построить для данной задачи компьютерную систему, которая обеспечит оптимальное по некоторому критерию отображение задачи на построенную систему.

Проблема 2. Получить для данной задачи ее оптимальное по некоторым критериям отображение на существующую компьютерную систему.

Исследованию поставленных проблем для различных уровней компьютерных систем посвящено значительное количество работ. В частности, способы решения обеих проблем для уровня микрокода приводятся в [4].

В настоящей работе рассматриваются методы решения второй проблемы для специального класса компьютерных систем, называемых локальными мультикомпьютерами (ЛМК) или кластерами рабочих станций [5–7].

Современные подходы к построению параллельных компьютерных систем могут быть представлены двумя большими направлениями.

Первое направление определяется специально разработанными для конкретных приложений массово-параллельными компьютерными системами, такими, как **Thinking Machines CM-5** [8], **Intel Paragon** [9], **IBM SP-2** [10]. Эти системы содержат тысячи вычислительных элементов (ВЭ), связанных с помощью специального оборудования, что обеспечивает высокую скорость коммуникаций между ВЭ.

Второе направление связано с использованием существующих вычислительных структур в качестве параллельных систем. К таким структурам относятся кластерные системы и, в частности, кластеры рабочих станций, объединенных локальной вычислительной сетью [5-7]. Вычислительными элементами в таких системах являются обычно рабочие станции, или серверы, подобно вычислительным серверам, файл-серверам и т.д. В качестве сети обычно используется установленная ранее локальная вычислительная сеть, такая, как Ethernet, Fast Ethernet [11], ATM [12] или FDDI ring [13]. Наиболее часто используемой операционной системой является Unix со стеком протоколов TCP/IP [14,15].

Для отображения задачи на существующую компьютерную систему на разных уровнях используются специальные средства, расширяющие возможности соответствующего уровня. Ниже будет рассматриваться настройка на одном из следующих уровней:

- среды программирования. Средствами настройки здесь являются специальные операторы, функции и процедуры, которые добавляются к языку программирования с целью расширения его возможностей.
- операционной системы. На этом уровне возможности операционной системы расширяются путем добавления новых функций.
- архитектуры компьютерной системы. Здесь используются особенности структуры компьютерной системы. На этом уровне к системе могут добавляться новые компоненты, возможно изменение топологии системы и т.д.

Отображение задачи может происходить в двух режимах – ***статическом*** и ***динамическом***. При статическом все необходимые изменения соответствующего уровня компьютерной системы производятся заранее и вручную. Для динамического режима все места изменений определяются по ходу выполнения задачи, и изменения производятся автоматически, затем задача перезапускается на уже настроенной компьютерной системе.

Настройка на уровне операционной системы. Рассмотрения будут проводиться для задач, для которых существенным является их распараллеливание по ходу выполнения. Обычно эти задачи содержат большой объем вычислений, и целью распараллеливания является уменьшение времени их выполнения.

Параллельное программирование, очевидно, требует наличия соответствующей среды обслуживания, то есть возникает проблема создания специальной операционной системы, которая обеспечила бы управление передачей данных, взаимодействием процессов, динамическим реконфигурированием системы. Ниже рассматриваются наиболее известные на сегодняшний день операционные системы, которые поддерживают парадигмы параллельного программирования. Отметим, что все рассматриваемые системы поддерживают только режим статического отображения задачи на компьютерную систему.

PVM

PVM (Parallel Virtual Machine) [7,16–19] – это надстройка над операционной системой Unix, которая позволяет рассматривать разнородную компьютерную систему как единую виртуальную машину. Схема функционирования PVM [7] приведена на рис 1.

Основная составляющая управляющая часть PVM – PVMD (демон PVM) – поддерживает следующие средства настройки задач на заданную компьютерную систему:

- управление механизмами передачи сообщений с одновременной конвертацией форматов данных;
- распределение задач между компьютерами, входящими в состав виртуальной машины, при этом учитываются загруженность и потенциальные возможности каждого компьютера.

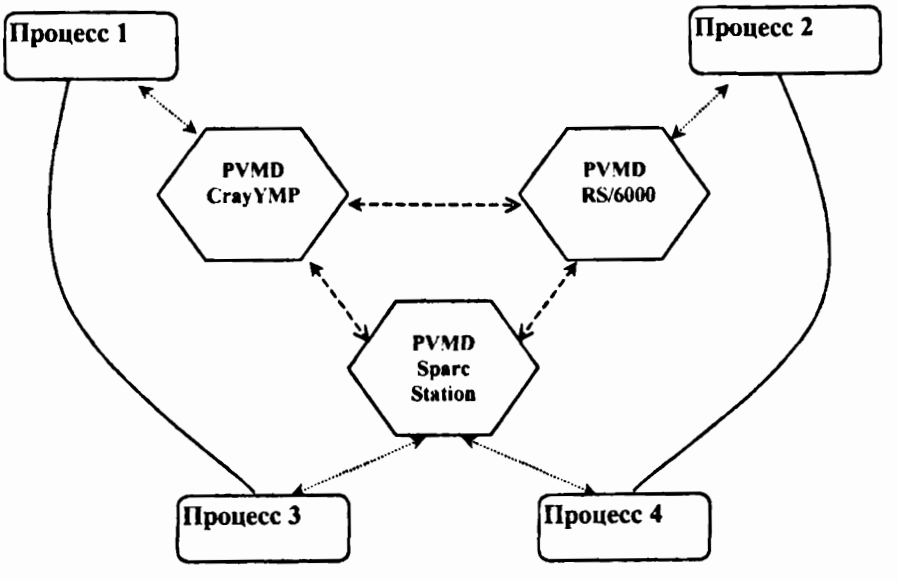

Рис.1. Схема функционирования PVM.

В PVM добавлены специальные функции, позволяющие поддерживать динамическую реконфигурацию системы. Любая задача может самостоятельно добавить или удалить вычислительный элемент из виртуальной машины. Кроме того, любая задача может самостоятельно запустить или удалить произвольное количество задач.

Отображение задача–машина не обязательно является отображением один к одному: на каждой машине может быть запущено одновременно несколько процессов. Другая особенность – логическая взаимосвязанность процессов. Все запущенные в рамках одного процесса задачи могут напрямую обмениваться сообщениями. Обмен сообщениями происходит асинхронно: процесс, посылающий сообщение, не дожидается его приема адресатом. Поддерживаются также специальные средства для синхронизации процессов. Для процессов, запущенных на различных архитектурных платформах, при передаче сообщений производится автоматическая конвертация форматов данных.

MPI

MPI (Message Passing Interface) [18–21] – это среда для параллельных вычислений, которая содержит большую библиотеку функций передачи данных, используемых для широкого круга локальных мультикомпьютеров и массово-параллельных компьютеров. Однако MPI не поддерживает управление процессами и конфигурирование виртуальной машины. Таким образом, MPI можно рассматривать как промежуточный уровень между аппаратным уровнем и уровнем операционной системы. На сегодняшний день MPI является наиболее распространенной средой для параллельных вычислений, используемой, в частности, не только для исследовательских целей, но и для разработки коммерческих продуктов [22, 23].

P4

В P4 [24] средствами настройки являются утилиты и функции, разработанные для параллельного программирования. P4 поддерживает два режима программирования: модель разделяемой памяти (shared-memory model) и модель распределенной памяти (distributed-memory model). Для модели с распределенной памятью P4 поддерживает функции передачи данных. Управление процессами основано на использовании конфигурационного файла, который есть на каждой машине.

Настройка на уровне среды программирования. Очевидно, что разработка параллельных приложений требует наличия соответствующей среды программирования, которая поддерживает функции обмена данных, синхронизации процессов и т.д. В обзор включены существующие среды и языки параллельного программирования, которые могут быть непосредственно использованы (по крайней мере, в идейном смысле) для локальных мультикомпьютеров. Приводится также их сравнительная характеристика.

PVM

Для разработки программ, работающих в среде PVM, поставляется специальная библиотека функций, подключаемая к языкам программирования C и Fortran. Библиотека содержит разнообразные функции обмена сообщениями, конфигурирования системы, синхронизации,

инициации и терминации процессов. Поддерживаются модели программирования Master-Slave и SPMD (single-program, multiple-data) [16].

МАЯК

Язык МАЯК [2] был разработан для макроконвейерных компьютерных систем. Он имеет два уровня представления программ, каждый из которых поддерживается различными средствами операционной системы.

Первый уровень содержит обычные операции передачи данных и поддерживает также динамическое распараллеливание программ. Кроме того, поддерживаются обе модели памяти: разделяемая и распределенная. На этом уровне поддерживается только SPMD модель программирования.

Второй уровень поддерживает динамическую конфигурацию виртуальной машины, а также модель программирования Master-Slave.

Split-C

Split-C [25] – это расширение языка программирования C, специально разработанное для параллельного программирования. Split-C основан на модели разделяемой памяти, использующей глобальные указатели для обмена данными. Язык поддерживает только SPMD модель программирования и не поддерживает динамическое конфигурирование системы.

ORCHID

ORCHID [26, 27] – среда для параллельного программирования в разнородной компьютерной системе. ORCHID содержит большое множество утилит, реализованных на языке программирования C, с простым пользовательским интерфейсом. Наиболее используемые из них - это утилиты обмена данными, утилиты синхронизации задач, динамического распределения процессов и т.д. ORCHID поддерживает описание любой виртуальной топологии.

SONiC

SONiC (Shared-Memory Net-interconnected Computer) [28] основан на принципах объектно-ориентированного программирования, и его главная особенность – использование ***копируемых разделяемых объектов*** (Replicated Shared Objects), которые имеют в своем описании операции передачи данных и синхронизации процессов. Другой основной компонент - ***сервис расписаний*** (Scheduling Service) – позволяет резервировать необходимое количество циклов ЦПУ для параллельных вычислений на каждом вычислительном элементе. Третий компонент – ***сервис удаленного выполнения*** (Remote Execution Service) – управляет инициализацией и терминацией параллельных процессов. Наконец удобный графический интерфейс пользователя позволяет определить однородную виртуальную машину, основанную на кластерах рабочих станций.

В таблице приводятся сравнительные характеристики рассмотренных выше систем параллельного программирования по их функциональным возможностям.

Система	Поддержка модели разделяемой памяти	Поддержка модели распределенной	Автоматическое распределение процессов	Поддержка виртуальной топологии	Динамическая конфигурация	Поддержка SPMD модели программ	Поддержка модели Master-Slave программ	Поддержка разнородной системы
MAYAK	+	+	+	-	+	+	+	-
PVM	-	+	+	-	+	+	+	+
SPLIT-C	+	-	+	-	-	+	-	-
ORCHID	-	+	+	+	-	-	+	+
MPI	-	+	-	-	-	+	-	+
SONiC	+	-	+	-	-	+	-	-

Настройка на уровне архитектуры.

Архитектура VLIW. Архитектура VLIW (Very Long Instruction Word) [29–32] – это специальная архитектура, которая позволяет распараллеливать программу на уровне команд процессора.

Процессоры, имеющие такую архитектуру, обрабатывают за такт одно длинное командное слово. Из кэша команд выбирается очередное командное слово, состоящее из нескольких независимых примитивов, и отправляется на параллельное выполнение. Эти возможности VLIW процессоров используются специальными компиляторами, генерирующими код, в котором последовательно сгруппированы параллельно выполняемые команды. Логика управления работой процессора с архитектурой VLIW довольно проста, так как не требуется ни динамического распределения, ни переупорядочивания операций (как в большинстве современных суперскалярных процессоров), достаточно разместить на процессоре больше регистров и функциональных устройств.

В настоящее время проводятся работы по созданию компиляторов, транслирующих объектный код, сгенерированный для традиционных архитектур, на архитектуру VLIW. Подробный обзор работ, рассматривающих проблемы проектирования программного обеспечения для архитектуры VLIW и, в частности, указанных компиляторов, приведен в работе [33]. Основные принципы архитектуры VLIW использованы в новейших разработках архитектуры микропроцессоров IA-64 [34] компаний Intel и HP и Power4 [35] компании IBM.

Кластерные системы. Вычислительная кластерная система – это совокупность компьютеров, объединенных в рамках некоторой сети для решения конкретной задачи. Каждый компьютер или ВЭ работает под управлением своей копии операционной системы. Мощность ВЭ может меняться в пределах одного кластера, что позволяет собирать неоднородные кластерные системы.

Один из первых примеров кластерных систем – кластер Beowulf [33, 36, 37], был разработан в 1994 году. Он состоит из 17 процессоров Intel 486DX4/100 *МГц*. На каждом узле установлено 3 сетевых Ethernet адаптера. Для работы в такой конфигурации были разработаны специальные драйверы, распределяющие трафик между доступными сетевыми картами.

Другой пример кластерных систем – кластер the HIVE (Highly-parallel Integrated Virtual Environment) [37–38]. Он состоит из четырех подкластеров и объединяет 332 процессора. В 1998 году был построен суперкомпьютер Avalon [33, 37, 39], который первоначально состоял из 68 процессоров, а затем был расширен до 140. На каждом узле было установлено 256 *Мбайт* оперативной памяти, жесткий диск на 3 Гбайт и сетевой адаптер Fast Ethernet. В апреле 2000 года для биомедицинских исследований был разработан кластер Velocity+ [37,40], состоящий из 64 ВЭ с двумя процессорами Pentium III/733 *МГц* и 2 *Гбайт* оперативной памяти каждый. Эти, а также другие современные кластерные системы подробно рассматриваются в [41,42].

Параллельная сетевая архитектура CNA. Кластеры рабочих станций имеют много преимуществ – таких, как низкая стоимость, простота эксплуатации, открытость архитектуры. В то же время показано, что они подходят для решения задач, не требующих интенсивного обмена данными, так как с увеличением количества используемых рабочих станций происходит резкое ухудшение временных показателей [41] (рис.2.).

Рис.2. Тенденция насыщения кластеров рабочих станций.

Для улучшения представления при использовании кластеров рабочих станций предлагается использовать CNA (Concurrent Network Architecture) – параллельную сетевую архитектуру [41–45], которая является расширением базовой кластерной архитектуры. Рабочие станции в CNA расположены в виде регулярной n-мерной решетки (рис.3.). Каждая рабочая станция подсоединена к n различным и независимым сетевым каналам и имеет n различных сетевых адресов. Каждый процессор в этой системе должен выполнять также задачи маршрутизации по всем сетевым каналам. Одним из главных вопросов в таких системах является определение необходимости добавления нового измерения в CNA. Дополнительные средства, наличие которых позволяет динамически добавить новое измерение в процессе распараллеливания задачи, описаны в [41].

В [42–44] решается задача выбора оптимального маршрута доставки сообщений с помощью специально разработанного демона коммуникаций, позволяющего эффективно использовать коммуникационные каналы CNA. При этом учитываются длина маршрута и загруженность каждого канала.

В [45] рассматривается задача оптимального отображения вычислительного процесса, представленного в виде графа без циклов, на линейную кластерную архитектуру, а также на CNA.

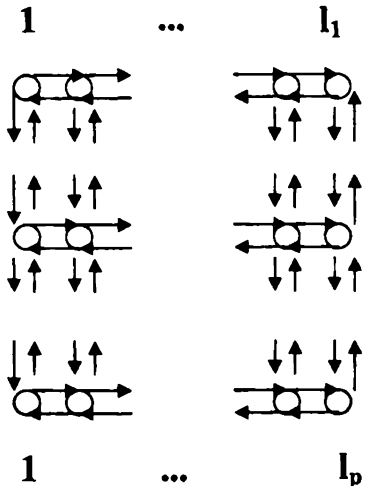

Рис.3. Двухмерная модель CNA.

Заключение. Ввиду очевидных трудностей формализации рассмотренных подходов к настройке ЛМК на задачи пользователя в рамках единой модели для их качественного сравнительного анализа единственно возможным методом на данном этапе представляется проведение экспериментов и выполнение сравнений на конкретных задачах пользователя.

В последующих публикациях будут рассмотрены настройка ЛМК на некоторые конкретные задачи пользователей в рамках описанных сред и оценка их эффективности.

Работа выполнена в рамках гранта CRDF #374100.

Кафедра алгоритмических языков *Поступила 03.10.2000*

ЛИТЕРАТУРА

1. **Gajski D.D., Frank Vahid, Sanjiv Narayan, Jie Song.** Specification and Design of Embedded Systems. PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994.
2. **Капитонова Ю.В., Летичевский А.А.** Математическая теория проектирования вычислительных систем, М.: Наука, 1988.
3. **Voevodin V.** Mathematical Foundation of Parallel Computing. Word Scientific Publishing, Singapore-New Jersey –Hong Kong, 1992.
4. **Шукурян С.К.** Исследование, разработка и реализация принципов многопрограммной адаптации вычислительных машин широкого применения к решаемым задачам. Автореф. дис. на соискание уч. ст. докт. физ.-мат. наук, Киев, Институт кибернетики АН Украины, 1990.
5. **Rajkumar Buyya.** High Performance Cluster Computing: Architectures and Systems. Prentice Hall PTR, NJ, USA, 1999.
6. **Greg Burns, Raja Daoud** and **James Vaigl.** LAM: An open cluster environment for MPI. In Proceedings of Supercomputing Symposium'94, pp.379-386, 1994.

7. **Schmidt B.K.** and **Sunderam V.K.** Empirical Analysis of Overheads in Cluster Environments 1994. Technical report available via FTP from: mathcs.emory.edu,/pub/vss/empanal.ps.

8.**Steele G.L.** An Overview of the Connection Machine Model CM-5, pp.87-103, Supercomputer'92, Mannheim, Springer, 1992.

9.**Bemmerl Th.** Paragon XPIS – The Road to TeraFLOPS. pp.82-86, Supercomputer'92, Mannheim, Springer, 1992.

10.**Hwang K., Degroot D.** Parallel Processing for Supercomputers and Artifical Intelligence. McGraw-Hill, 1989.

11.**Boggs D.R., Mogul J.C.** and **Kent C.A.** Measured Capacity of an Ethernet: Myths and Reality. Research Report 88/4. Digital Equipment Corporation, Western Research Laboratory, 1988.

12.**Lin M., Hsieh J., Thomas J., MacDonald J.** Distributed Network Computing over local ATM networks. University of Minnesota, Technical Report TR-94-17, 1994.

13.**Mirchandi S., Khanna R.** FDDI: Technology and Applications Wiley, New York, 1993.

14.**Stevens W.** Advanced Programming in the Unix Environment. Addison-Wesley,1993.

15.**Comer D.E.** Internetworking with TCP/IP, Volume I. Prentice Hall, 1991.

16.**Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., Sunderam V.** PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. The MIT Press,1994.

17.**Geist G.A.** Network Based Concurrent Computing on the PVM System. Concurrency:Practice and Experience 4(4), pp. 293-311,1992.

18.**Geist G.A.,Kohl J.A.** and **Papadopoulos P.M.** PVM and MPI: A comparison of features. Calculateurs Paralleles, 8(2),1996.

19. **William Gropp., Ewing Lusk.** Why are PVM and MPI so different? In Marian Burak, Jack Dongarra and Jerzy Wasniewski, editors, Recent Advances in Parallel Virtual Machine and Message Passing Interface, v. 1332 of Lecture Notes in Computer Science, pp.3-10, Springer Verlag, 1997, 4-th European PVM/MPI User's Group Meeting, Cracow, Poland, 1997.

20. **Message Passing Interface Forum**. MPI: A Message Passing Interface standard. Computer Science Dept. Technical Report CS-94-230, University of Tennessee, Knoxville, TN, April, 1994.(To appear in the International Journal of Supercomputer Applications, v. 8, N 3/4,1994).

21.**William Gropp., Ewing Lusk** and **Anthony Skjellum.** Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press,1994.

22.**Hempel R.** The MPI message-passing standard and its implementation on the NEC SX-4. In Shun Doi, editor. Proceedings of the NEC HPC Workshop, Tokyo, Japan, 1996.

23.**Gropp W., Lusk E.** Implementing MPI: The 1994 Implementors' Workshop. Argonne National Laboratory, Argonne, IL 61039, 1994.

24.**Butler R.** and **Lusk E.** Monitors, messages and clusters: The p4 parallel programming system. Technical Report Preprint MCS-P362-0493, Argonne National Laboratory, Argonne, IL,1993.

25.**David E. Culler., Andrea Dusseau, Seth Copen Goldstein, Arvind Krishnamurthy, Steven Lumetta, Thorsten von Eicken,** and **Katherine Yelick.** Parallel Programming in Split-C. The MIT Press, 1993.

26.**Volotis C., Manis G., Tsanakas P. and Papakonstaninou G.** Facilitating the Development of Portable Parallel Applications on Distributed Memory Systems.In Proceedings of the 1995 Massively Parallel Programming Models Conference, Berlin.

27.**Voliotis C., Manis G., Lekatsas H.,Tsanakas P.** and **Papakonstantinou G.** ORCHID: A portable platform for parallel programming. Techical report, National Technical University of Athens, 1994.

28. **Andreas Polze, Miroslaw Malek.** Network Computing with SONiC Journal of Systems Architecture, N 44, 1998.

29.**Moreno J.H., Moudgil M, Ebcioglu K. , Altman E.R., Hall B., Miranda R., Chen S.K., Polyak A.** Simulation/Evaluation Environment for a VLIW processor architecture. IBM Journal of Research and Development, v. 41, No. 3, May 1997, pp.287-302.

30.**Ebcioglu K., Altman E., Sathaye S., Gschwind M.** Execution-Based Scheduling for VLIW Architectures. Proc. Europar '99 (P. Amestoy, P. Berger, M. Dayde, I. Duff, V. Fraysse, L. Giraud, D. Ruiz, eds.), pp. 1269-1280, Lecture Notes in Computer Science 1685, Springer Verlag, 1999.

31.**Евстигнеев В.А.** Некоторые особенности программного обеспечения ЭВМ с длинным командным словом (обзор). Программирование, N2, 1991.
32.**Ellis J.R.** BULLDOG: A compiler for VLIW architectures. MIT Press, 1986.
33.**Корнеев В.В.** Параллельные вычислительные системы. М.Нолидж, 1999.
34.**Кузьминский М.** Краткий обзор IA-64. Открытые системы, N9-10, 1999.
35.**Кузьминский М.** Будущее архитектуры Power4. Открытые системы N4, 2000.
36. http://www.beowulf.org. Официальный сайт проекта Beowulf.
37.**Андреев А., Воеводин В., Жуматий С.** Кластеры и суперкомпьютеры – близнецы или братья? Открытые системы, N5-6, 2000.
38.http://newton.gs.fc.nasa.gov/thehive/.
39.Introduction Avalon. http://www.eklektix.com/lwn/980507/a/avalon.html45.
40.http://www.lobos.nih.gov.
41. **Shoukourian, S.K.** and **Tavangarian D.** Implementation of Algorithms on Computer Systems Using a Concurrent Network Architecture. In Proceedings of the 1996 Simulation Multiconference ("High Performance Computing 96" conference), SCS, USA, 1996.
42.**Klein, M., Hipper G.** and **Tavangarian D.** Performance in Workstation Clusters with a Concurrent Network Architecture. In Proceedings of "High Performance Computing 95" conference , USA,1995.
43. **Tavangarian D., Klein M., Hipper G.** Parallel Computing in Workstation Clusters using a Concurrent Network Architecture. ISMM 94, Washington D.C., 1994.
44. **Klein M., Hipper G., Tavangarian D.** Communication Performance Measurements in a Concurrent Network Archtecture. HPC'95, Phoenix, Arizona, April 1995.
45.**Geoletsyan H.G., Shoukourian S.K., Tavangarian D.** Fast algorithms of tuning computer systems based on a concurrent network architecture. In Proceedings of HIGH PERFORMANCE COMPUTING '98, SCS International Advanced Simulation Technologies Conference ASTC'98, Boston, USA, April 1998, pp.163-167.

Ի.Ե. ԲՈՅԱԽՉՅԱՆ, Ս.Կ. ՇՈՒՔՈՒՐՅԱՆ

ԼՈԿԱԼ ՄՈՒԼՏԻՀԱՄԱԿԱՐԳԻՉՆԵՐԻ ՀԱՐՄԱՐԵՑՈՒՄԸ ՕԳՏԱԳՈՐԾՈՂՆԵՐԻ ԽՆԴԻՐՆԵՐԻ ՀԱՄԱՐ

Ամփոփում

Հոդվածում դիտարկվում է օգտագործողների խնդիրների վրա կոմպյուտերային համակարգերի արտապատկերման պրոբլեմը: Քերվում է գոյություն ունեցող միջոցների վերլուծությունը կոմպյուտերային համակարգի տարբեր մակարդակների համար, կատարվում է նաև այդ միջոցների համեմատություն ըստ ֆունկցիոնալ հնարավորությունների: Որպես կոմպյուտերային համակարգ դիտարկվում է լոկալ մուլտիհամակարգիչների դասը, որոնք այլ կերպ կոչվում են աշխատանքային կայանների կլաստերներ:

I.E. BOYAKHCHYAN, S.K. SHOUKOURIAN

TUNING OF LOCAL MULTICOMPUTERS TO SPECIFIC TASKS

Summary

The problem of tuning of computer systems to specific tasks is considered. A review of existing tools that provide tuning on different levels for a specific class of computer systems named local area multicomputers is presented. A comparative analysis of considered tools is given according to their functional characteristics.