**УДК 681.3.06**

А. М. ЛАРГИНА (Технологический институт ВНУ им. В. Даля, Северодонецк)

# О РЕАЛИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ПРИЛОЖЕНИЙ РЕАЛЬНОГО ВРЕМЕНИ

Рассмотрены системы параллельного программирования для реализации приложений реального времени в управлении железнодорожным транспортом. Обосновывается выбор наиболее оптимальной системы программирования.

*Ключевые слова:* параллельная вычислительная система, система параллельного программирования, отказоустойчивость

## Введение

Развитие интеллектуальных транспортных систем [1] предполагает применение передовых информационных и вычислительных технологий, например, приведенных в [2]. Для реализации задач реального масштаба времени (принятие оперативных решений на основе моделирования, мониторинг условий движения, задачи планирования и организации перевозок) необходимы мощные вычислительные ресурсы.

Одним из предпочтительных по стоимости вариантов организации высокоскоростных вычислений является параллельная вычислительная система с кластерной структурой [3]. В качестве вычислительных узлов в кластере обычно используются персональные компьютеры, SMP-серверы. Сетевая операционная система имеет в своем составе средства передачи сообщений между компьютерами по линиям связи. На основе этих сообщений поддерживаются синхронизации доступа к разделяемым ресурсам, обнаружение отказов и динамическая реконфигурация системы [4].

Основным требованием к вычислительным процессам реального времени, реализующим функции обеспечения безопасности, является устойчивость к сбоям вычислительных ресурсов. Традиционным способом обеспечения отказоустойчивости при выполнении последовательной задачи является ее перезапуск на исправном ресурсе с некоторой контрольной точки. Более сложным является обеспечение отказоустойчивости параллельной задачи в системах без синхронного резервирования (к которым принадлежат кластеры). Процессы параллельной задачи реализуются на разных узлах и динамически взаимодействуют между собой. В этом случае простой перезапуск процесса с контрольной точки может привести либо к потере сообщений, отправленных другому процессу, но не полученных им в результате сбоя, либо к посылке повторных сообщений вследствие повторного выполнения уже исполненного фрагмента программы, либо и того и другого вместе. Для создания отказоустойчивых приложений необходимы соответствующие инструментальные средства. В связи с этим возникает задача сравнительного анализа и выбора системы параллельного программирования, наиболее адекватной требованиям функциональности и отказоустойчивости.

Целью данной статьи является оценка систем параллельного программирования для применения в системах реального времени на железнодорожном транспорте.

Для этого рассмотрим наиболее представительные системы, нашедшие промышленное применение.

## Обзор систем параллельного программирования

Существует достаточно много систем программирования, например [5–12], позволяющих достаточно эффективно создавать параллельные приложения. Наиболее предпочтительны системы со следующими свойствами:

– удобство программирования (универсальный интерфейс программирования; наличие библиотек подпрограмм для использования в программах на языке C++; скрытость от пользователя проблем коммуникаций);

– имеющие хорошие средства распараллеливания приложений (возможность порождения несколько процессов, взаимодействующих между собой; наличие эффективных коммуникаций, адаптирующихся к структуре физической системы с минимизацией времени обмена данными);

– содержащие встроенные механизмы обеспечения отказоустойчивости приложения при выходе из строя какого-либо из компонентов параллельной системы. Кратко рассмотрим некоторые из них.

### Параллельная Виртуальная Машина

PVM (Parallel Virtual Machine) – представляет собой надстройку над операционной системой UNIX и может использоваться на различных аппаратных платформах: ПЭВМ; локальная сеть, включающая в себя суперкомпьютеры с параллельной архитектурой; универсальные ЭВМ, графические рабочие станции и пр. [5]. PVM представляет собой набор библиотек и утилит, предназначенных для разработки, отладки параллельных программ и управления конфигурацией виртуальной вычислительной машины. PVM предоставляет программисту единый виртуальный многопроцессорный вычислительный комплекс.

Поддерживаются языки C/C++ и FORTRAN, имеются средства сопряжения библиотек PVM и с другими языками, такими как Perl, Java.

В PVM реализованы следующие принципы:

– управление конфигурацией виртуальных вычислительных ресурсов может осуществляться любым пользователем системы;

– единицей параллелизма является задача, состояние которой изменяется в процессе работы программы. Задача в определенные промежутки времени может выполнять вычисления, переходить к обмену данными и т. д.;

– использование явной модели обмена сообщениями;

– поддержка всех типов неоднородности;

– поддержка симметричных многопроцессорных систем.

При выполнении программы порождается несколько процессов, взаимодействующих между собой посредством обмена сообщениями. Обмен и другие операции реализуются обращениями к подпрограммам библиотек. Система PVM состоит из двух основных компонентов. Первым является процесс pvmd3 (демон – в терминологии ОС UNIX), который запускается на всех компьютерах, входящих в состав виртуальной машины. Эти процессы работают без связи с терминалом, находясь в состоянии "сна" до тех пор, пока какой-нибудь процесс не обратится к соответствующей службе ОС. Вторым компонентом PVM является набор библиотек, реализующих те или иные операции. В параллельных программах, написанных на языке C++, используются функции из библиотеки Hbpvm3. Модель программирования PVM включает как SPMD, так и MPMD-модели [3], причем обе они равноправны. Все это дает возможность эффективно использовать ресурсы параллельной вычислительной системы.

### Язык ADA

Ada — структурный, модульный, объектно-ориентированный язык программирования, содержащий высокоуровневые средства программирования параллельных процессов [6]. Является официальным языком программирования министерства обороны США. Основное назначение языка Ada – разработка больших программных систем реального времени для встроенных компьютерных систем. Это не исключает его использования для решения параллельных вычислительных задач.

Помимо раздельной компиляции модулей и обеспечения иерархической секретности спецификаций, в этом языке была реализована поддержка параллельного программирования в виде встроенных средств поддержки многозадачности.

Программа на языке Ada состоит из множества задач, выполняющихся одновременно и независимо от остальных. Механизмы межзадачного обмена данными и синхронизации основаны на высокоуровневых концепциях «рандеву» и использовании защищенных объектов. В спецификации задачи публикуются различные входы (entry) в задачу, в которых она готова ожидать обращения к ней от других задач. В процессе взаимодействия одна из задач рассматривается как сервер, а вторая – как клиент, причем задача-сервер не может быть инициатором начала взаимодействия.

Механизмы параллелизма языка Ada – это защищенные типы, которые поддерживают синхронизированный доступ к разделяемым данным, а также оператор requeue, обеспечивающий планирование и синхронизацию в зависимости от аргументов вызова.

Защищенные функции обеспечивают доступ только для чтения к скрытым переменным; следовательно, функцию могут вызвать одновременно несколько задач. Защищенные процедуры обеспечивают исключительный доступ к скрытым переменным для чтения и записи. Защищенные точки входа похожи на защищенные процедуры, но имеют еще часть when, которая определяет логическое условие синхронизации. Защищенная процедура или точка входа в любой момент времени может выполняться только для одной вызвавшей ее задачи. Вызов защищенной точки входа приостанавливается, пока условие синхронизации не станет истинным и вызывающая задача не получит исключительный доступ к скрытым переменным. Условие синхронизации не может зависеть от параметров вызова.

**Система с передачей сообщений MPI**

MPI (Message Passing Interface) представляет собой библиотеку функций, предназначенную для поддержки работы параллельных процессов в терминах передачи сообщений [7]. Система MPI является основным средством программирования таких высокопроизводительных мультикомпьютеров, как Cray T3D, Cray T3E, IBM SP2 и других. Ее реализации представляют собой библиотеки подпрограмм, которые могут использоваться в программах на языках C/C++ и FORTRAN. В настоящее время принята новая версия спецификации – MPI-2.

В модели программирования, которую поддерживает MPI, программа порождает несколько процессов, взаимодействующих между собой с помощью обращений к подпрограммам передачи и приема сообщений. Спецификация MPI обеспечивает переносимость программ на уровне исходных кодов и большую функциональность. Поддерживается работа на гетерогенных кластерах и симметричных многопроцессорных системах.

Эффективность и надежность обеспечиваются: определением MPI операций не процедурно, а логически, т. е. внутренние механизмы выполнения операций скрыты от пользователя; использованием непрозрачных объектов в MPI (группы, коммуникаторы, типы и т. д.); хорошей реализацией функций передачи данных, адаптирующихся к структуре физической системы. Различают две модели параллельных вычислений: MPMD-модель (Multiple program – Multiple Data) и SPMD-модель (Single program – Multiple Data). В первом случае MPI-программа представляет собой совокупность автономных процессов, функционирующих под управлением своих собственных программ и взаимодействующих посредством стандартного набора библиотечных процедур для передачи и приема сообщений. Во втором случае (SPMD-модель) все процессы исполняют различные ветви одной и той же программы. Такой подход обусловлен тем обстоятельством, что задача может быть достаточно естественным образом разбита на подзадачи, решаемые по одному алгоритму.

Наличие в системе таких средств как: виртуальные топологии, коллективные взаимодействия, создаваемые пользователем типы данных и др., значительно повышают уровень параллельного программирования по сравнению с системами, у которых нет таких средств.

В проекте MPI-2 введены дополнительные по отношению к MPI, весьма важные возможности: поддержка механизма "клиент-сервер" для программирования многофункциональных задач; динамическое создание и уничтожение процессов (для работы в сетях); для работы с файлами есть архитектурно-независимый интерфейс (это необходимо для случая, если диск находится на одной ЭВМ, а процесс – на другой).

Важным достоинством MPI является наличие системы CMDE (Channel Memory based Dynamic Environment) [8], позволяющей решать проблему восстановления выполнения параллельного приложения после сбоя его узла на основе использования Памяти Каналов.

Система CMDE является динамическим программным окружением, состоящим из множества взаимосвязанных компонентов, каждый из которых запускается на отдельном узле параллельной системы или сети. Динамизм системы определяется обеспечением динамического включения компонентов в систему или исключения из неё.

Система CMDE состоит из двух подсистем: подсистемы низкоуровневых коммуникаций для библиотеки MPICH и подсистемы запуска и сопровождения параллельных MPI-программ. Первая подсистема состоит из одного компонента – коммуникационной библиотеки низкого уровня, реализующей всё множество высокоуровневых коммуникационных функций библиотеки MPICH. Вторая подсистема состоит из компонентов четырёх типов – Диспетчера, Сервера Памяти Каналов (СПК), Сервера Контрольных Точек (СКТ) и Исполнителя. В текущей реализации системы для выполнения поставленных задач необходимо существование одного Диспетчера, одного или более СПК, одного или более СКТ и двух или более Исполнителей.

Диспетчер обеспечивает добавление других компонентов в систему и отслеживает их отключение, формирует и поддерживает очередь приложений, назначает ресурсы приложению и перераспределяет ресурсы в случае сбоев.

Память Каналов разделяет процесс посылки и приёма сообщения между узлами и хранит сообщения в промежутке между посылкой и приёмом, а также сохраняет все сообщения, переданные через канал, начиная с момента последней контрольной точки процесса-получателя сообщений. Основными достоинствами системы является возможность асинхронного создания контрольных точек параллельными процессами на каждом узле и независимое восстановление параллельных процессов после сбоя.

Механизмы обеспечения отказоустойчивости, реализованные в системе, заключаются в следующем. Выход из строя узла влечёт за собой перезапуск его подзадачи на другом узле при наличии свободных или приостановку выполнения подзадачи до появления такого свободного ресурса. Выход из строя СПК приводит к приостановке коммуникаций всех подзадач, связанных с этим сервером, и возобновление их работы после завершения процесса реконфигурации системы или перезапуск всех подзадач приложения при невозможности реконфигурации. Выход из строя СКТ приводит к перезапуску всех подзадач связанного с ним приложения с использованием другого СКТ (при его наличии), либо к приостановке приложения и его перезапуску при появлении нового СКТ. Выход из строя Диспетчера не отражается на работе запущенных приложений, однако после их завершения все компоненты системы завершают работу, а все задачи, находившиеся в очереди, должны будут быть вновь зарегистрированы с помощью клиента. Возможна реализация восстановления очереди задач при перезапуске Диспетчера на том же узле.

### Платформа Java

Технология Java разработана на основе платформенно-независимого, переносимого, объектно-ориентированного языка, обеспечивающего разработчиков инструментарием для создания решений, не зависящих от операционной системы и аппаратной платформы, на которых эти решения будут функционировать [9]. Независимость от платформы достигается за счет того, что уникальные характеристики каждой из поддерживаемых Java платформ, реализованы в виде оболочки, называемой Java Runtime Environment (JRE). Компилятор языка Java преобразует код в последовательность байт-кодов, которая ориентирована на одну из конкретных платформ в рамках JRE.

С точки зрения создания распределенных систем, наиболее интересным из состава платформы является интегрированный программный интерфейс RMI (Remote Method Invocation – удаленный вызов методов или процедур), CORBA (модель удаленного вызова объектов) и возможность присоединения таких интерфейсов как Jini и JavaSpaces, реализованных на языке Java.

RMI обеспечивает несколько интерфейсов для Java-объекта, каждый из которых характеризует определенное поведение этого объекта. RMI поддерживает также передачу объектов из одного адресного пространства в другое, для чего используется технология сериализации объекта [10].

Jini – это набор соглашений, специфицирующих методы автоматического взаимодействия и регистрации устройств, подключаемых к сети [11]. Jini-технология базируется на Java-технологии, Технология Jini, как и CORBA, относится к промежуточному программному обеспечению. В дополнении к решению проблем гетерогенности Jini предоставляет унифицированную вычислительную модель, используемую разработчиками распределенных приложений.

В качестве недостатка технологии Jini можно указать только поддержку одного языка программирования (Java). Технология JavaSpaces фактически базируется на технологиях RMI и Jini и часто ее рассматривают как продолжение технологии Jini для построения распределенных систем. Данная технология рассматривает приложение как совокупность процессов, согласованных через цепочки объектов внутри и вне одного из нескольких «пространств» [12].

## ВЫВОДЫ

Достоинства PVM – простота, наличие наследованного от ОС UNIX аппарата процессов и сигналов, а также возможность динамического добавления к группе вновь созданных процессов. Недостатки – относительно низкая производительность по сравнению с MPI и функциональная ограниченность.

Достоинства Ada – более высокая надежность по сравнению с другими языками. Ada содержит механизм работы с исключительными ситуациями (exception), который позволяет управлять ими во время исполнения. Недостаток – сложность доступа к компиляторам (обязательна сертификация Пентагона) и неразвитые средства взаимодействия с программами на других языках.

Достоинства Java – большое количество средств для разработки многопоточных приложений, развитый инструментарий по перехвату исключительных ситуаций, отсутствие таких сложных и неоднозначных инструментов, как указатели или ручное выделение памяти. Недостатки – медленный и большой код, ограничения лицензирования, непредсказуемые, в отношении реального времени, функциональные характеристики. Причинами такой непредсказуемости является отсутствие спецификации на алгоритм планирования потоков (как правило, используется алгоритм, основанный на разде-

лении времени). Кроме того, в Java любой поток может быть приостановлен в произвольный момент на время, необходимое для очистки памяти, что неприемлемо для систем реального времени

Достоинство MPI – обеспечение отказоустойчивости вычислений за счет применения системы CMDE. Библиотеки операций MPI являются не лицензионными и открытыми (с открытым исходным кодом), что важно для сертификации программного обеспечения. К недостатку можно отнести то, что MPI является специализированным и весьма сложным средством программирования, так спецификация на MPI-1 содержит 300 страниц, на MPI-2 – еще 500 (только отличия и добавления к MPI-1), и программисту для эффективной работы необходимо с ними ознакомиться. Сложность является ценой за компромисс между эффективностью и универсальностью.

Таким образом, с учетом вышесказанного, можно сделать вывод, что наиболее предпочтительными из рассмотренных систем для реализации параллельных вычислительных задач в реальном времени являются системы программирования MPI.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. Гапонович, В. А. Разработка и продвижение интеллектуальных транспортных систем [Текст] / В. А. Гапонович, И. Н. Розенберг // Ж.-д. трансп. – 2011. – № 4. – С. 5–11.
2. Панченко, С. В. Применения грид технологий в системах безопасного управления железнодорожным транспортом [Текст] / С. В. Панченко, Н. Г. Панченко, С. Л. Пархоменко // Інформаційно-керуючі системи на залізничному транспорті. – 2011. – № 2. – С. 114–117.
3. Столлинг, В. Операционные системы, 4-е изд. [Текст] : пер. с англ. – М. : Изд. дом «Вильямс», 2002. – 848 с.
4. Рязанцев, О. И. Організація обчислювальних процессів в комп'ютерних системах [Текст] / О. И. Рязанцев, А. М. Ларгіна. – Луганьск : Вид-во СНУ ім. В. Даля, 2006. – 608 с.
5. Geist, A. Vaidy Sunderam PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing [Text] / A. Geist, A. Beguelin, J. Dongarra. – MIT Press, 1994.
6. Burns, A. Concurrent and Real-Time Programming In Ada [Text] / A. Burns, A. Wellings. – Cambridge University Press, 2005.
7. MPI: A Message-Passing Interface Standard, Version 2.2. Message Passing Interface Forum. High Performance Computing Center Stuttgart (HLRS) [Text]. – 2009. – 647 p.
8. Selikhov, A. CMDE: A Channel Memory Based Dynamic Environment for Fault-Tolerant Message Passing Based on MPICH-V Architecture [Text] / A. Selikhov, C. Germain // Parallel Computing Technologies. Lecture Notes in Computer Science. – 2003. – Vol. 2763/2003. – P. 528–537.
9. Sun Microsystems: Java Remote Method Invocation Specification. Sun Microsystems, Inc., Palo Alto, California [Text], 1999.
10. Sun Microsystems: Java 2 Platform Standard Edition. Sun Microsystems, Inc., Palo Alto, California [Text], 2002.
11. Edwards, W. K. Core Jini [Text] / W. K. Edwards; 2nd ed. – Sun Microsystems Press, 2001. – 1004 p.
12. Freeman, E. JavaSpaces. Principles, patterns and practice [Text] / E. Freeman, S. Hupfer, K. Arnold. Addison-Wesley Publ. Company, 2001. – 364 p.



Поступила в редколлегию 15.03.2012.
Принята к печати 19.03.20122.



А. М. ЛАРГІНА


# ПРО РЕАЛІЗАЦІЇ ПАРАЛЕЛЬНИХ ПРОГРАМ РЕАЛЬНОГО ЧАСУ


Розглянуто системи паралельного програмування для реалізації програм реального часу в управлінні залізничним транспортом. Обґрунтовується вибір найбільш оптимальної системи програмування.



*Ключові слова:* паралельна обчислювальна система, система паралельного програмування, відмовостійкість



A. M. LARGINA


# ON THE PARALLEL REAL-TIME APPLICATIONS


We consider a system of parallel programming to implement real-time applications in the railway transport management. The choice of optimal programming system is substantiated.



*Keywords:* parallel computing system, system of parallel programming, fault tolerance