УДК.004.27; 004.25; 004.382.2

Ю.С. Яковлев, Е.В.Елисеева

Основные принципы и методика распределения приложений в сложных компьютерных системах типа «Процессор–в–памяти»

С учетом особенностей архитектурно-структурной организации КС типа «Процессор–в–памяти» или *PIM*-системы, предложены блок-схема методики распределения приложений, основанная на новых принципах и математической модели распределения приложений, а также блок-схема контроллера, перечень пакетов прикладных программ и исходной информации, необходимых для ее реализации.

Taking into account the features of architecturally-structural organization KS of the «Processor–in–memory» type or PIM-systems, suggested are: a block diagram of the technique of distributing the applications as the basis of new principles and a mathematical model of the distribution of applications, as well as the controller block diagram, a list of packages of the applied programs and the initial information necessary for its realization.

З урахуванням особливостей архітектурно-структурної організації КС типу «Процессор–в–пам'яті» або *PIM*-системи, запропоновано блок-схему методики розподілу застосувань, яка заснована на нових принципах і математичній моделі розподілу застосувань, а також блок-схему контролера, перелік пакетів прикладних програм і початкової інформації, необхідних для її реалізації.

Введение. Появившийся вследствие развития интегральной технологии новый класс распределенных компьютерных систем типа «Процессор–в–памяти» («*Processor–in–memory*») или *PIM*-системы обладает в сравнении с классическими КС архитектурно-структурными особенностями, что позволяет в петафлопсном диапазоне решать задачи, плохо поддающиеся решению на *КС* с классической архитектурой [1, 2].

Так как *PIM*-системы строятся на основе больших интегральных схем (БИС) памяти путем размещения в них логических элементов для обработки информации, и эта концепция сохраняется в настоящее время, то одна из основных проблем обеспечения эффективной работы системы – управление памятью, включающая решение таких задач, как распределение памяти и приложений (пользовательских задач) по процессорам, размещение данных [3].

Естественно, что на решение этих задач накладывают отпечаток архитектурно-структурные особенности *PIM*-систем, это существенно сказывается на распределении приложений по процессорам, поскольку взаимосвязь между процессорами *PIM*-системы представляет собой многоуровневую иерархическую организацию: хост-машина – набор чипов – ведущий процессор (ВП) на каждом чипе – процессорные ядра (ПЯ), размещенные на том же чипе.

Известные подходы к распределению приложений в *PIM*-системах [4–6] основываются либо на так называемой итерационной технологии трансформации циклов, либо используют в качестве основной единицы анализа – оператор в цикле [4, 5]. При этом в основе подхода принята упрощенная модель *PIM*-системы: модель содержит по одному ВП и ПЯ (фактически *PIM*-система, размещенная на одном кристалле, содержит один ВП и множество ПЯ). Кроме того, использование оператора в цикле в качестве основной единицы анализа ставит под сомнение объединение и планирование операторов, находящихся вне циклов, а это означает, что не все операторы могут быть проанализированы на предмет их параллельного выполнения.

Известные методы распределения приложений по процессорам *PIM*-системы сложны и трудоемки, при этом сходимость алгоритма распределения – серьезная проблема. Отсутствуют пока и публикации принципов и методик распределения приложений, а также средств их аппаратной поддержки (в частности – контроллеров).

Авторами статьи предлагается подход к распределению приложений для *PIM*-систем, включая основные принципы, методику и структуру контроллера для поддержки реализации предложенной методики с учетом особенностей архитектурно-структурной организации *PIM*-систем.

Основные принципы распределения приложений в *PIM*-системе

Эти принципы учитывают прежде всего особенности архитектурно-структурной организа-

ции *PIM*-систем, а также приложений, обладающих свойствами широкого распараллеливания алгоритмов при их реализации, массовым обращением к памяти и часто – необходимостью обработки данных большой разрядности в петафлопсном диапазоне производительности. К ним могут быть отнесены следующие:

• *Многоступенчатое разделение приложения и распределение его фрагментов* (*модулей*) *по процессорам* в соответствии с многоуровневой архитектурно-структурной организацией *PIM*-системы: первый уровень разделения – системный (хост-машина – набор чипов); второй уровень для каждого чипа – узловой (ведущий процессор ВП – набор процессорных ядер ПЯ, представленный в виде эквивалентного ПЯ*; третий – блочный.

• *Максимальное соответствие набора команд* (*операций*) *процессоров ВП и ПЯ набору операций, необходимых для реализации соответствующих фрагментов* (*модулей*) *приложения.* Например, если в наборе команд ВП есть команда умножения, а в наборе команд ПЯ она отсутствует, то фрагмент (модуль) приложения, имеющий множество команд умножения, приписывается для реализации на ВП. Если же одна и та же команда присутствует в наборах команд ВП и ПЯ, то принадлежность соответствующего фрагмента приложения для реализации на ВП или ПЯ определяется по принципу минимального «параметрического веса» этого фрагмента.

• *Минимальный* «*параметрический вес*», под которым понимается оценка времени реализации одного и того же фрагмента (модуля) алгоритма на ВП и отдельно на ПЯ, при этом данный фрагмент приписывается для рабочей реализации процессору с минимальным оценочным временем (весом).

• *Базовая процедура исходного разделения приложения на фрагменты* (*модули*, *которые определим как основные* – $М_{осн}$). Такой процедурой может быть, например выявление и анализ циклов, которые могут быть вложенными, при этом каждый уровень вложенности может содержать только один цикл и несколько операторов. Для идентификации этих модулей проводится соответствующая маркировка циклов в приложении, которые не содержат вызовов подпрограмм, приводящих к циклам, а также условных операторов, включающих циклы.

• *Зависимость* (*связность*) *по данным фрагментов* (*модулей*) *приложения*, *отдельных подзадач и вложенных пакетов прикладных программ* ($ПП_{вл}$). За ее отсутствием фрагменты (модули) разделенного приложения, подзадачи и $ПП_{вл}$ могут выполняться параллельно. Следует стремиться, чтобы взаимосвязь по данным между фрагментами (модулями) приложения осуществлялась в основном итоговыми значениями результатов их реализации.

• *Подбор размеров фрагментов* (*модулей*) *при распределении приложения по процессорам.* Малый размер фрагментов (модулей) приложения приводит к высоким накладным затратам, связанным с поддержкой когерентности КЭШ и с привязкой кода операций к модулям, готовым для выполнения. В связи с этим можно сформировать так называемые составные модули ($М_{сост}$) путем объединения некоторых основных модулей $М_{осн}$ с соседними операторами, используя принцип «параметрического веса» модуля, а также принцип операторной зависимости (связности) по данным. При этом наполнение модулей $М_{осн}$ операторами осуществляется в основном по принципу операторной зависимости (связности) по данным. С другой стороны, большие по размеру модули затрудняют их размещение в распределенные области свободной памяти. В этом случае применяют декомпозицию модулей, также используя принцип «параметрического веса» и принцип связности по данным.

• *Учет при распределении приложения ограничений на отдельные параметры PIM-системы*, определяемые особенностями ее архитектурно-структурной организации и возможностями интегральной технологии (например, ограничение на систему команд ПЯ и емкость банка памяти, подключенного к ПЯ), что в значительной степени определяет функциональную

* Все ПЯ набора размещены на одном чипе с ВП.

загрузку каждого ПЯ и уменьшает размер фрагментов (модулей) распределения приложения.

• *Контроль и обеспечение баланса загрузки процессоров на каждом этапе распределения фрагментов* (*модулей*) *приложения.* При этом критерием загрузки каждого процессора используется время реализации всех приписанных ему фрагментов (модулей) приложения, т.е. их «параметрический вес». Загрузка процессоров сбалансированна, если время реализации части приложения на ВП приблизительно равно времени реализации соответствующих частей приложения на всем наборе ПЯ на том же чипе, а время реализации частей приложения на одном чипе примерно одинаково для всех чипов. Это дает возможность при соблюдении принципа независимости по данным параллельно обрабатывать распределенные части приложения как внутри каждого чипа, так и во всей *PIM*-системе, содержащей множество чипов.

Таким образом, согласно перечисленным принципам, все ПЯ будут наделены соответствующими модулями $\mathrm{M}_{\text{ПЯ}}$, и при условии их параллельного выполнения, что и предполагалось с самого начала, эквивалентное время выполнения этих модулей на ПЯ будет равно максимальному значению времени выполнения одного модуля в сравнении с другими, т.е. $T(\mathrm{M}_{\text{ПЯ}})_{\text{экв}} = T(\mathrm{M}_{\text{ПЯ}})_{\max}$. При этом время выполнения каждого модуля рассчитывается как произведение количества операторов на время их реализации плюс время, затраченное на коммутацию информации между модулями, на обращение в локальную память за данными и время задержки работы ПЯ при неудачном обращении процессора в память типа КЭШ (при наличии такого типа памяти для каждого ПЯ).

Время $T(\mathrm{M}_{\text{ВП}})_{\text{экв}}$ реализации всех модулей, приписанных ВП, определяется произведением количества $N_{\text{ВП}}$ реализуемых последовательно операторов на время реализации каждого оператора плюс время, затраченное на обращение в память за данными, задержки работы ВП при неудачном обращении процессора в память типа КЭШ (при наличии такого типа памяти для ВП), плюс время на инициализацию связей между ВП и ПЯ при обмене данными и выдаче результатов реализации модулей от ПЯ к ВП. При этом необходимо стремиться, чтобы $T(\mathrm{M}_{\text{ВП}})_{\text{экв}} = T(\mathrm{M}_{\text{ПЯ}})_{\text{экв}}$, что обеспечивает возможность параллельного выполнения модулей ВП и ПЯ. Данные принципы положены в основу предложенной методики распределения приложений в *PIM*-системе. Базовые компоненты методики приведены на рис. 1, а укрупненная блок-схема методики показана на рис. 2 и 3.

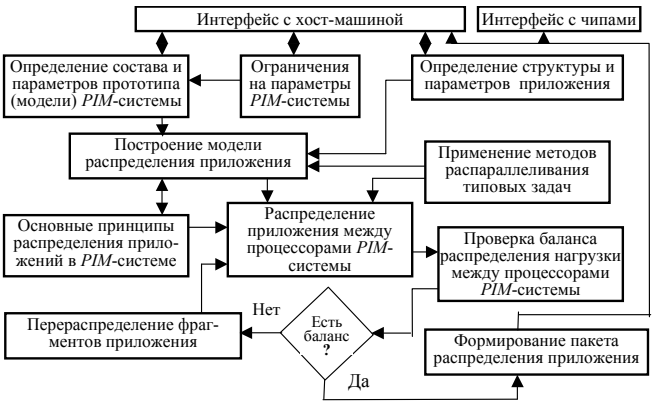


Рис. 1. Базовые компоненты методики распределения приложений по процессорам *PIM*-системы

Сущность всех компонентов данной блок-схемы раскрыта при отображении методики (рис. 2 и 3) и блок-схемы контроллера (рис. 4), за исключением блока «Построение модели распределения приложения». Методы распараллеливания типовых задач достаточно подробно изложены в [7, 8]. Поэтому на предложенном способе построения модели остановимся подробнее.

Построение модели распределения приложения и определение конфигурации *PIM*-системы

Учитывая особенности архитектурно-структурной организации *PIM*-системы, а также рис. 1, процедура распределения приложений может быть представлена в виде кортежа, который содержит следующие компоненты [9]:

$$R_{\text{РПЛ}} = (\mathrm{M}_{\text{РПЛ}}, R_F, \mathrm{B}_{\text{РПЛ}}, R_{\text{ТС}}, \mathrm{B}_{\text{ЗГ}}), \qquad (1)$$

где $\mathrm{M}_{\text{РПЛ}}$ – построение математической модели стратегии распределения приложений и описание ее компонентов; R_F – выделение признаков (функций), отличающих *PIM*-систему от КС с классической архитектурой при соответству-

ющей целевой функции распределения; $В_{РПЛ}$ – выбор способа распараллеливания вычислительных алгоритмов типовых задач; $R_{ТС}$ – формирование таблиц соответствия набора операций приложения и набора команд процессоров *PIM*-системы; $В_{ЗГ}$ – проверка баланса загрузки процессоров *PIM*-системы.

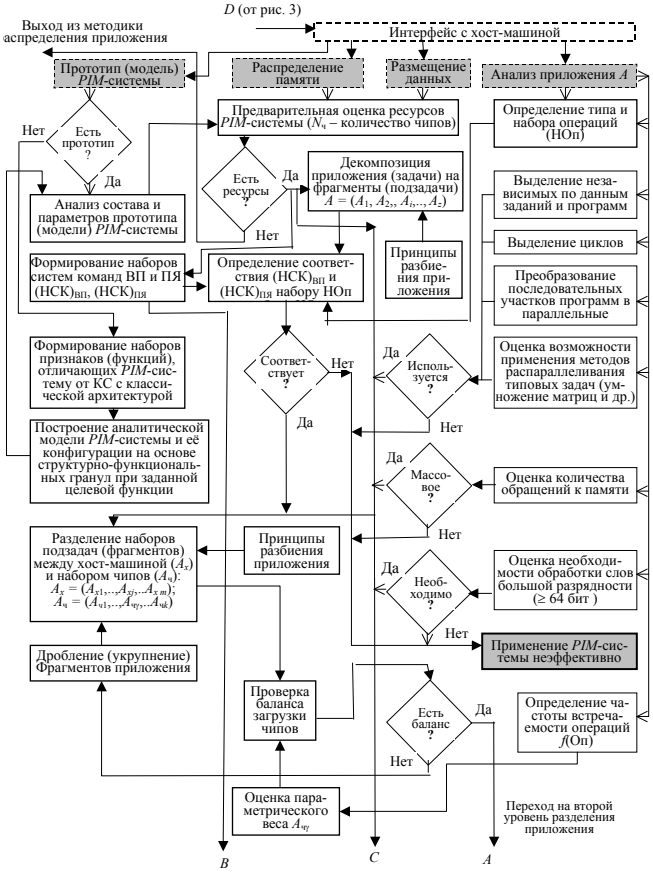


Рис. 2. Укрупненная блок-схема первого (системного) уровня методики распределения приложения для *PIM*-систем

С учетом (1), а также базовых компонентов методики распределения приложений по процессорам *PIM*-систем, математическая модель стратегии разделения приложений может быть представлена следующим образом:

$$E_{РПЛ} = \{(\forall a_{РПЛ} \in A_{РПЛ})[\exists R_{РПЛ} \in \xi]: (f(R_{РПЛ}) = W_{РПЛ}, W_{РПЛ} \in \Omega_{РПЛ}) \Rightarrow F_{РПЛ}\}|_{Z_{ОГР}}, \quad (2)$$

где $F_{РПЛ}$ – целевая функция распределения приложения (например, *повышение производительности*, *масштабирование параллелизма*, *эффективное использование ресурсов*), т.е. для всех значений параметров $a_{РПЛ}$ приложения $A_{РПЛ}$ существует способ разделения приложения $R_{РПЛ}$,

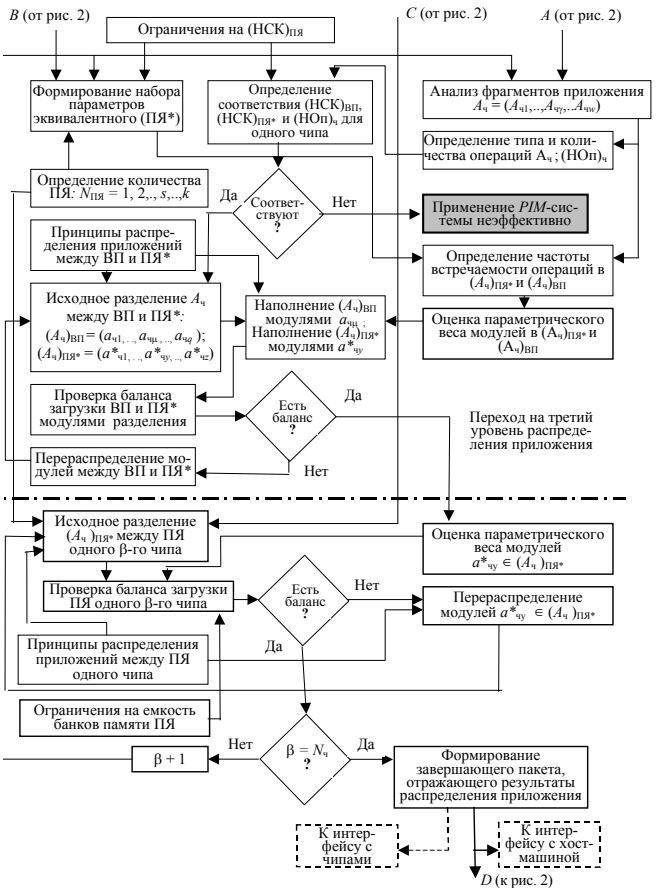


Рис. 3. Укрупненная блок-схема второго и третьего уровней методики распределения приложения для *PIM*-систем

являющийся составной частью стратегии распараллеливания приложений ξ, такой, что реализация этого способа $f(R_{РПЛ})$, представленного через параметры множества $\Omega_{РПЛ}$, отражающие признаки (функции), отличающие *PIM*-системы от классических КС, приводит к выполнению заданной целевой функции $F_{РПЛ}$ разделения приложения при ограничениях $Z_{ОГР}$ на параметры *PIM*-системы.

Определение компоненты R_F основано на *выделении признаков* (*функций*), *отличающих PIM-систему от КС с классической архитектурой при соответствующей целевой функции распределения* $F_{РПЛ}$. Для этого мы считаем целесообразным использовать таблицы параметров для разработанной ранее модели *PIM*-системы [9, 10], основанной на использовании теории нечетких множеств и теории гранулирования.

Если при распределении приложения определяется функция цели $F_{РПЛ}$ – повышение производительности системы (этот параметр в [9,

10] обозначен β), то для обеспечения высокой производительности *PIM*-система должна иметь набор средств и способов, обеспечивающих соответствующие параметры (признаки, функции), отличающие ее от *КС* с классической архитектурой (см. табл. 1).

Таблица 1. Набор средств и способов, обеспечивающих сверхвысокую производительность и отличающих *PIM*-систему от *КС* с классической архитектурой

Параметр	Средства и способы, обеспечивающие признаки (функции), отличающие *PIM*-систему от *КС* с классической архитектурой
s_1	Широкая полоса пропускания по каналу процессор-память (существенно больше в сравнении с классическими *К*С)
s_4	В качестве *ВП* в составе *PIM*-чипа используются как серийные, так и специализированные процессоры
s_5	В качестве средств коммутации внутри чипа используются скоростные коммутаторы (селекторы выбора), а между чипами – высокоскоростная межчиповая сеть
w_1	Модифицированная операционная система ($ОС_I$), например, модифицированная *Unix*, *Linux* и т.п.
W_6	Программные средства для поддержки работы межчиповой коммутационной сети – *МЧКП* (*ПО/МЧКП*) и иерархической системы памяти (*ПО/ПАМ*) в различных режимах
w_{10}	Набор системных сервисных программ – *ПО* разбиения алгоритма на гранулы трех уровней: системного, узлового и блочного
c_1	Управление процессами *PIM*-системы основано на концепции потоковой обработки информации и управляемого сообщением вычисления
c_2	Поддержка управляемого сообщением вычисления основана на концепции пакетов, содержащих значения параметров и спецификаторы действий, а также дополнительные поля, необходимые для транспортировки, обнаружения ошибок, маршрутизации и управления контекстом
c_8	Используется мультипроцессирование для быстрого удовлетворения входящих запросов на обслуживание и обеспечения перекрытия по времени выполнения вычислений, коммуникаций и доступа к памяти
p_8	*PIM* интегрируются в очень больших количествах, обеспечивая возможность параллелизма и распределенных вычислений, чем их традиционные кластерные копии
p_{11}	Эффективность коммуникации делает системы *PIM* лучшими для мелкомодульного распределенного вычисления, чем их кластерные копии
a_1	Возможность разделения реализуемого алгоритма на три уровня: системный, узловой и блочный
a_3	Наличие часто повторяющихся участков алгоритма, реализация которых требует максимальных ресурсов памяти и производительности
a_5	Высокая степень параллелизма алгоритма решаемой задачи с помощью ресурсов одного кристалла (чипа)
a_6	Масштабирование параллелизма за счет наращивания *PIM*-чипов (до нескольких тысяч чипов)

Функционально-структурная гранула при этом имеет вид:

$$\Gamma(\beta) = (s_1, s_4, s_5, w_1, w_6, w_{10}, c_1, c_2, c_8, p_8, p_{11}, a_1, a_3, a_5, a_6), \quad (3)$$

где $s_j \in S$, $w_q \in W$, $c_h \in C$, $p_r \in P$, $a_i \in A$, при этом S – структурно-архитектурный срез модели *PIM*-системы [9, 10]; W– срез программного обеспечения; C– срез управления процессами внутри чипа; P – пользовательский срез; A – алгоритмический срез; j, q, h, r, i, k определяют номера элементов множеств, а их количество определяет мощности m_j, m_q, m_h, m_r, m_i, m_k. Если же используется в качестве функции цели $F_{\text{РПЛ}}$ – масштабирование параллелизма (этот параметр в [9, 10] обозначен b), то соответствующая гранула имеет вид:

$$\Gamma(b) = (s_1, w_{12}, c_1, p_8, p_{11}, a_1, a_2, a_3, a_4, a_6). \quad (4)$$

Аналогично могут быть получены гранулы для других значений функций цели.

Отметим, что формирование такого типа гранул при определенных целевых установках и манипуляция с ними дает возможность при отсутствии прототипа предварительно оценить требуемую конфигурацию, свойства и функции *PIM*-системы, для которой в дальнейшем формируется стратегия, алгоритм распределения и размещения приложения по процессорам. Например, если поставить задачу *повышения быстродействия только за счет масштабирования параллелизма*, то эквивалентная гранула будет иметь вид:

$$\Gamma(\beta / b) = \Gamma(\beta) \cap \Gamma(b) = (s_1, c_1, p_8, p_{11}, a_1, a_3, a_6). \quad (5)$$

В соответствие с этим, *PIM*-система (в сравнении с КС с классической архитектурой) должна иметь (табл. 1):

- широкую полосу пропускания по каналу процессор-память, существенно большую в сравнении с классическими КС (параметр s_1) ;
- потоковую обработку информации и управляемое сообщением вычисления (параметр c_1);
- интеграцию *PIM* в очень больших количествах, обеспечивая большую возможность параллелизма и распределенных вычислений, чем их кластерные копии (параметр p_8);
- более мелкомодульное распределенное вычисление, чем в традиционных кластерных копиях, и эффективную коммуникацию (параметр p_{11});
- возможность разделения реализуемого алгоритма на три уровня: системный, узловой и блочный (параметр a_1);

• наличие часто повторяющихся участков алгоритма, реализация которых требует максимальных ресурсов памяти и производительности (параметр a_3);

• масштабирование параллелизма путем наращивания *PIM*-чипов (параметр a_6).

Методика распределения приложений для *PIM*-систем

Используя в качестве исходного базиса предложенные выше принципы и математическую модель, авторами разработана методика распределения приложений в *PIM*-системе, укрупненная блок-схема которой приведена на рис. 2 и 3. При этом в качестве исходной информации используется следующая:

♦ параметры имеющейся *PIM*-системы (прототипа), для которой должно быть распределено приложение; если же прототип отсутствует, то определяется конфигурация и основные свойства *PIM*-системы путем построения ее математической модели, а также модели распределения приложения при заданной целевой функции, используя теорию нечетких множеств и теорию гранулирования;

♦ параметры приложения, подлежащего распределению, которые определяются либо на хост-машине, либо на другой компьютерной системе с соответствующими характеристиками с помощью типовых пакетов прикладных программ;

♦ наличие необходимых ресурсов по емкости памяти как *PIM*-системы в целом, так и отдельного банка памяти и всех банков памяти каждого чипа с учетом ограничений;

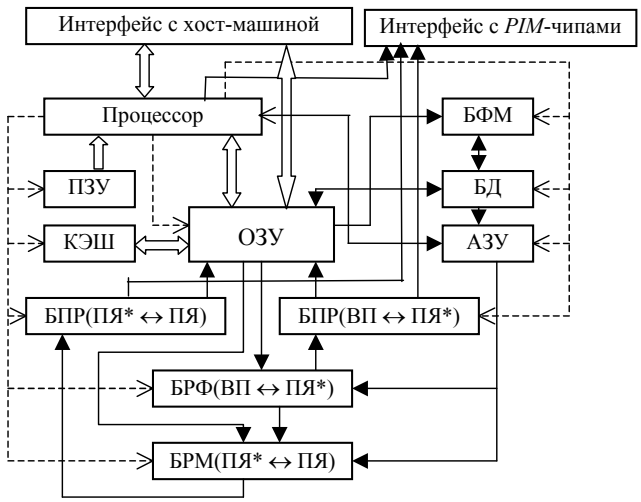
♦ наличие необходимых ресурсов по производительности всех процессоров и их количеству с учетом ограничений на систему команд ПЯ;

♦ выполненные процедуры распределения областей памяти и размещения в них данных для всей *PIM*-системы;

♦ средства программной (пакеты прикладных программ – ПП) либо аппаратной (контроллер распределения приложения), либо программно-аппаратной (комбинированной) поддержки для реализации методики распределения приложения (см. рис. 2 и 3).

Учитывая, что для построения контроллера могут быть использованы коммерческие изделия (микропроцессор, оперативная и постоянная памяти и др.), а также тот факт, что контроллер берет на себя нагрузку от других процессоров *PIM*-системы по распределению приложения, повышая тем самым эффективность их использования, принимаем комбинированный способ распределения приложений с применением контроллера. Укрупненная блок-схема такого контроллера приведена на рис. 4.

Информация, необходимая для распределения приложения, а также средства программной поддержки процессов распределения согласно блок-схеме, приведенной на рис. 2 и 3, отражены соответственно в табл. 2 и 3.



Обозначения:

ОЗУ – оперативное (основное) запоминающее устройство;

ПЗУ – постоянное запоминающее устройство для хранения микропрограмм;

КЭШ – буферное ОЗУ типа КЭШ;

АЗУ – ассоциативное запоминающее устройство для хранения таблиц соответствия систем команд ВП и ПЯ набору операций приложения;

БД – база данных для хранения информации о процессорах, приложении и др.;

БФМ – блок формирования модели распределения приложения;

БРФ – блок распределения фрагментов приложения;

БРМ – блок распределения модулей фрагментов приложения;

БПР – блок формирования пакетов по результатам распределения приложения на различных этапах;

ВП – ведущий процессор чипа;

ПЯ – процессорные ядра, размещенные на чипе и подключенные к банкам памяти;

ПЯ* – эквивалентное ПЯ, используемое на втором этапе разделения приложения, с параметрами: емкость памяти равна сумме емкостей памяти всех банков памяти, размещенных на чипе, а время цикла работы равно времени цикла одного ПЯ, поделенное на количество ПЯ, при этом набор команд ПЯ* равен набору команд одного ПЯ.

Рис. 4. Укрупненная блок-схема контролера распределения приложений для *PIM*-системы

Таблица 2. Информация, размещенная в БД и необходимая для распределения приложения в *PIM*-системе

№ п/п	Наименование информации, размещенной в БД
1	Набор системы команд ВП
2	Набор системы команд ПЯ
3	Наборы признаков (функций), отличающих *PIM*-систему от КС с классической архитектурой
4	Математическая модель *PIM*-системы и описание ее компонентов
5	Математическая модель стратегии распределения приложения и описание ее параметров
6	Конфигурация *PIM*-системы при заданной целевой функции
7	Идентификаторы методов распараллеливания типовых задач (например, умножение матриц, операции на графе и др.)
8	Идентификаторы методов оценки «параметрического веса» фрагментов приложений
9	Идентификаторы методов дробления (укрупнения) фрагментов приложений
10	Ограничения на систему команд ПЯ
11	Ограничения на емкость банков памяти чипа
12	Ограничения на другие параметры *PIM*-системы согласно ТЗ

Исходная информация согласно табл. 2 передается хост-машиной через соответствующий интерфейс сначала в оперативно запоминающее устройство (ОЗУ) и после анализа и структурирования – в БД контроллера, а пакеты прикладных программ (табл. 3) также загружаются в ОЗУ хост-машиной.

Конструктивно такой контроллер может быть выполнен на отдельном кристалле с использованием при изготовлении кристалла схемотехнических и технологических решений коммерческих компонентов, в частности – микропроцессоров, памяти, базы данных, интерфейса и т.п. Например, в качестве интерфейса для связи контроллера с хост-машиной и чипами *PIM*-системы может быть использована стандартная шина *PCI*.

Заключение. Процедура распределения приложений в сложных компьютерных системах типа «Процессор–в–памяти» наряду с процедурами распределения памяти и размещения данных наиболее трудоемка и ответственна, поскольку от ее реализации зависит эффективность использования *PIM*-системы и, следовательно, ее производительность.

Особенности архитектурно-структурной организации *PIM*-системы в сравнении с *КС* с классической архитектурой, а также приложений отразились на выполнении этой процедуры: в основу ее реализации положены новые принципы распределения приложений, включающие многоуровневость в соответствие с уровнями организации системы; сопоставление параметров системы (в частности – системы команд процессоров) с соответствующими параметрами приложения, применение методов распараллеливания стандартных задач и др.

Таблица 3. Программная среда (пакеты прикладных программ) поддержки распределения приложений в *PIM*-системах

Обозначение	Назначение ПП
$ПП_1$	Определение характеристик приложения: – типа и состава набора операций (On); – частоты встречаемости операций $f(On)$; – типа и количества циклов ($К_ц$); – количества обращений к памяти ($К_М$); – разрядности обрабатываемых данных ($R_Д$).
$ПП_2$	Определение характеристик прототипа (модели) системы: – типа и состава блоков С(Бл); – типа и состава системы команд ведущего процессора $(СК)_{ВП}$ и $(СК)_{ПЯ}$; – разрядности ведущего процессора ($R_{ВП}$) и процессорного ядра ($R_{ПЯ}$); – емкости банков памяти Q(БП), подключенных к ПЯ.
$ПП_3$	Оценка «параметрического веса» фрагментов приложения на уровнях разделения
$ПП_4$	Распараллеливание типовых задач (умножения матриц, операций на графе и др.)
$ПП_5$	Реализация блок-схемы общей методики разделения приложений и распределение его фрагментов по процессорам
$ПП_6$	Выделение из приложения независимых по данным подзадач и вложенных ПП
$ПП_7$	Распределение подзадач между хост-машиной ($A_Х$) и набором чипов ($A_ч$)
$ПП_8$	Дробление / укрупнение фрагментов разделения приложения
$ПП_9$	Выделение последовательных участков приложения, которые могут быть преобразованы в параллельные, и их преобразование
$ПП_{10}$	Проверка баланса загрузки процессоров чипа фрагментами приложения

Примечание: алгоритмы построения модели распределения приложения, распределение его фрагментов между ВП и ПЯ*, а также распределение модулей фрагментов между всеми ПЯ одного чипа рекомендуется реализовать аппаратно (рис. 4).

Предложенная методика распределения приложений может быть использована не только при наличии конкретного прототипа *PIM*-системы, но и при его разработке, для чего предлагается подход к построению конфигурации системы при заданной целевой функции с улучшенными параметрами в сравнении с классическими *КС*.

Методика может быть реализована либо программным способом с применением пакетов соответствующих программ в хост-машине, либо аппаратным с помощью специализированной *БИС* контроллера, либо комбинированным способом с применением контроллера на коммерческих компонентах, а также ограниченного набора информации в базе данных контроллера и пакетов прикладных программ, загружаемых в *ОЗУ* контроллера с хост-машины.

1. *Архитектурно*-структурная организация компьютерных средств класса «Процессор–в–памяти» / А.В. Палагин, Ю.С. Яковлев, Б.М. Тихонов и др. // Математичні машини і системи. – 2005. – № 3.– С. 3–16.
2. *Елисеева Е.В.*, *Яковлев Ю.С.* О концепции построения программной среды *PIM*-систем // УСиМ. – 2008. – № 4. – С. 58–67.
3. *Яковлев Ю.С.*, *Елисеева Е.В.* Основные задачи и методы реализации функций управления памятью в *PIM*-системах // Математичні машини і системи. – 2008. – № 2. – С. 47–62.
4. *Tsung-Chuan Huang, Slo-Li Chu.* A Parallelizing Framework for Intelligent Memory Architectures // Proc. of The Seventh Workshop on Compiler Techniques for High-Performance Computing; (Hsinchu, Taiwan, Mar. 15–16, 2001) (CTHPC 2001). – P. 96–10. – http:// parallel.iis.sinica.edu.tw/cthpc/7th/ 03CTHPC2001-Hardcopy.pdf
5. *Slo-Li Chu, Tsung-Chuan Huang.* Exploiting Application Parallelism for Processor–in–Memory Architecture // Proc. of 2003 National Comp. Symp. (Taichung, Taiwan, Dec. 18–19, 2003). – P. 2293–2303. – http:// dspace. lib.fcu.edu.tw/bitstream/2377/564/1/ OT_1022003305. pdf
6. *Yan Solihin.* Improving memory performance using intelligent memory. THESIS of Doctor of Philosophy in Comp. Science in the Graduate College of the Univer. of Illinois at Urbana-Champaign, 2002. – 102 p. – http// portal.acm.org/citation.cfm?id=936938&coll=&dl=& CFID=15151515&CFTOKEN=6184618.pdf
7. *Гергель В.П.*, *Стронгин Р.Г.* Основы параллельных вычислений для многопроцессорных вычислительных систем: Учеб. пособие. – Н. Новгород: Изд-во Нижегородского госуниверситета, 2003. – 82 с.
8. *Гергель В.П.* Теория и практика параллельных вычислений: Учеб. пособие. – М.: Интернет-Университет Информационных Технологий, 2007. – 423 с.
9. *Яковлев Ю.С.*, *Елисеева Е.В.* Математическая модель и основные положения стратегии распределения приложений для интеллектуальной памяти распределенных компьютерных систем // Математичні машини і системи. – 2009 – № 4. – С. 3–17.
10. *Елисеева Е.В.*, *Яковлев Ю.С.* Математическая модель функциональной среды *PIM*-системы на основе теории нечетких множеств и теории гранулирования // Там же. – 2009 – № 1. – С. 40–54.

Поступила 10.09.2009

Тел. для справок: (044) 526-3207, (Киев)

●