УДК 004.27

КЛИМЕНКО И.А.,
ЖАБИНА В.В.

ОБЕСПЕЧЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ ПОТОКОВЫХ СИСТЕМ НА ОДНОТИПНЫХ ВЫЧИСЛИТЕЛЬНЫХ МОДУЛЯХ

Предлагаются методы автоматической реконфигурации потоковых вычислительных систем с различной организацией сред формирования команд. Показана возможность исправления ошибок, вызванных отказом вычислительных модулей, на аппаратном уровне без дополнительного расхода команд. Процесс вычислений продолжается, пока в системе остается хотя бы один работоспособный вычислительный модуль.

Methods of automatic reconfiguration for data-flow systems with different instruction generation environment configuration are proposed. Possibility to fix errors caused by computation unit failure without additional instruction use at hardware level is shown. Computation process is active until there is at least one operable computation unit.

Введение

Для систем управления процессами и объектами в реальном времени важными характеристиками является быстродействие и надежность. Как известно, возможным подходом к ускорению обработки информации является распараллеливание вычислительных процессов на различных уровнях. В системах реального времени зачастую возникает необходимость реализации алгоритмов с мелкозернистой структурой. К ним относятся, например, алгоритмы интерполяции функций, расчета траектории объектов, преобразования координат в многомерном пространстве. Ускорения вычислений в этом случае можно добиться распараллеливанием вычислений на уровне операций. Использование вычислительных систем, в которых распределение заданий между вычислительными узлами осуществляется под управлением программы с использованием статических средств программирования, неэффективно ввиду большого числа обмена данными между узлами, выполняющими отдельные операции. Объединение операций, связанных по данным, в программный модуль уменьшает число пересылок данных, но снижает возможности распараллеливания вычислений.

В работе [1] показано, что эффективным подходом к ускорению реализации мелкозернистых алгоритмов является использование динамических средств распределения операций между вычислительными узлами, основанных на потоковой модели вычислений (вычислений, управляемых потоком данных). В потоковых системах, реализующих указанную модель вычислений, распределение операций осуществляется динамически в процессе вычислений на аппаратном или микропрограммном уровне. При подготовке задачи нет необходимости в прямой форме описывать процедуры синхронизации процессов и обмена данными между ветвями алгоритмов. Все это создает предпосылки для ускорения обработки информации.

Эффективным подходом к проблеме повышения надежности систем и достоверности результатов обработки информации является обеспечение отказоустойчивости систем. Наиболее экономичными с точки зрения аппаратуры являются программные способы повышения отказоустойчивости. Однако для систем реального времени зачастую временные затраты при таком подходе оказываются неприемлемыми. В данном случае более эффективными являются методы динамической реконфигурации систем при отказе оборудования. Проблеме обеспечения отказоустойчивости систем посвящено множество публикаций, в том числе, монографий [2-4]. Однако, эта проблема не разработана для систем, управляемых потоком данных, ввиду специфики организации вычислительных процессов в таких системах.

Таким образом, разработка методов и средств обеспечения отказоустойчивости вычислительных потоковых систем, в которых обеспечивается динамическое распределение заданий между вычислительными узлами и динамическая реконфигурация при отказе оборудования является актуальной задачей.

Модель вычислений, управляемых потоком данных

К первым работам в области организации систем, управляемых потоком данных, можно отнести публикации [5-8]. Несмотря на особенности архитектуры потоковых систем, они объединены общей концепцией, связанной со способом формирования и порядком выполнения команд.

В системах, управляемых потоком данных, команды выполняются не в заданной программой последовательности, а по мере поступления полной информации о команде и операндах, то есть определяющим для выполнения команды является доступность данных, а не заданный программой порядок выполнения команд.

Данные для вычислений могут быть подготовлены на основе потокового графа, вершины которого определяют операции, а дуги – потоки данных. Таким образом, связь между командами осуществляется только по данным.

Операции описывается информационным словом, которое называют актором (*actor*). В общем случае он может быть представлен в виде следующего кортежа

$$A = \langle I, F, N, P \rangle, \qquad (1)$$

где I – идентификатор актора; F – функция преобразования данных; N – имя актора, для которого передается результат в качестве операнда, а P – совокупность признаков данного операнда.

В свою очередь, данные описываются в следующем виде

$$D = \langle I, Q, N, P \rangle, \qquad (2)$$

где Q – значение операнда.

На аппаратном уровне потоковая вычислительная система состоит из среды формирования команд, нескольких вычислительных модулей и блока управления. Связь между структурными компонентами и вычислительной системой более высокого уровня осуществляется через коммутационную среду. Различные аппаратные решения определяют различный способ формирования и активизации команд. В систему поступают акторы и данные. С использованием определенных средств и алгоритмов формируются команды, которые выполняются в свободном вычислительном модуле.

Постановка задачи

Несмотря на общую модель вычислений, потоковые системы имеют существенные отличия, связанные со способом активизации команд. Существуют различные подходы к структурной организации СФК. Известны методы формирования команд в СФК на основе ассоциативной памяти (АП) [7] и памяти с произвольным адресным доступом к ячейкам (ППД) [9]. Целью работы является исследование возможности обеспечения отказоустойчивости потоковых систем на однотипных ВМ, использующих разные механизмы формирования команд.

Автоматическая реконфигурация потоковых систем с АП

В потоковых системах подготовка задачи осуществляется без учета количества вычислительных модулей. Это создает предпосылки в случае отказа ВМ продолжать вычисления до тех пор, пока в системе не останется хотя бы один работоспособный ВМ. Учитывая, что методы обеспечения отказоустойчивости памяти разработаны достаточно хорошо (см., например, [10]), основное внимание уделяется аппаратным средствам автоматической реконфигурации системы при отказе ВМ. При этом основной проблемой является восстановление информации о команде, утерянной в связи с отказом ВМ.

Пусть для определенности ВМ выполняют двуместные операции (для одноместных может быть введен фиктивный операнд). Тогда команды из акторов (1) и данных (2) формируются в формате $\langle A, D_1, D_2 \rangle$.

Организация системы поясняется укрупненной структурной схемой (рис. 1). В состав системы входят вычислительные модули (ВМ), среда формирования команд (СФК) на базе ассоциативной памяти (АП), регистры для временного хранения информации (РВ), блок управления (БУ) буферная память (БП) и коммутационные среды (КС).

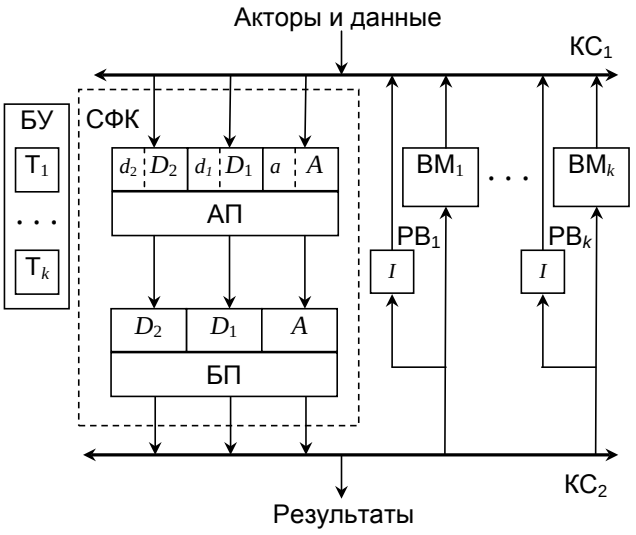

Рис. 1. Организация системы с АП

Формирование команд производится следующим образом. Акторы и данные в любом порядке поступают через $КС_1$ в СФК. С использованием процедуры адресной записи акторы и данные для i–й команды записываются в одну ячейку АП по адресу I_i, который в соответствии с (1) и (2) совпадает для всех объектов одной команды. Одновременно с записью актора и данных устанавливаются признаки их наличия в соответствующих разрядах ячейки памяти (d_2, d_1, a). Значения признаков $d_2 d_1 a = 111$ указывает на наличие в АП готовой команды для ВМ. Готовая команда считывается из АП с помощью процедуры ассоциативного чтения и переписывается в БП типа FIFO. Цикл безадресного чтения данных из БП значительно меньше цикла ассоциативного чтения из АП, что дает возможность ускорить вычисления.

Свободный $ВМ_j$ принимает команду из БП и приступает к ее выполнению. Адрес I_i на время выполнения команды сохраняется в $РВ_j$. Одновременно с этим сбрасываются в нулевое состояние признаки в соответствующей ячейке АП и блокируется запись новой информации в указанную ячейку. Пока признаки снова не будут установлены в единицу, команда считается неготовой и не может быть считана повторно другим ВМ. Вместе с этим в БУ запускается таймер $Т_j$ на время, которое достаточно для выполнения самой продолжительной команды. Число таймеров должно быть равно числу ВМ.

После успешного выполнения команды результат из $ВМ_j$ через $КС_1$ записывается в АП по адресу N_i. Вместе с этим сбрасывается таймер $Т_j$ и разблокируется адресная запись в ячейку с адресом I_i, то есть эта ячейка может быть использована для формирования другой команды с соответствующим именем.

В случае отказа $ВМ_j$ срабатывается таймер $Т_j$ (заканчивается лимит времени на выполнения команды). $ВМ_j$ отключается от КС и команда активируется путем установки признаков $d_2 d_1 a = 111$ в ячейке АП с адресом I_i, который сохранялся в $РВ_j$. Свободный ВМ считывает эту команду и она выполняется повторно, как указано выше.

Поскольку подготовка алгоритма к реализации для потоковой системы выполняется без учета конкретного числа ВМ в системе, то имеется возможность продолжения вычислений до тех пор, пока в системе будет оставаться хотя бы один работоспособный ВМ.

Основным недостатком системы является сложность ассоциативной памяти и большая длительность цикла ассоциативного поиска данных, что ограничивает объем памяти и снижает производительность системы.

Трудность реализации ассоциативных запоминающих устройств большого объема приводит к необходимости их эмуляции с применением других технических средств, например, специализированных процессоров [6], что позволяет увеличить объем памяти, но существенно снижает производительность, поскольку процесс эмуляции требует больших временных затрат. Кроме того, при такой организации системы невозможно использовать ячейки АП после считывания команды для других команд, что требуется, например, при итерационных вычислениях и вычислениях в конвейерном режиме. Все это снижает эффективность параллельных вычислений.

Реконфигурация систем с ППД

Уменьшение объема оборудования для активизации акторов и ускорения циклов обращения к памяти для формирования команд может быть достигнуто с использованием СФК на основе механизма адресного чтения и записи [1, 9] с использованием ППД. Возможны различные варианты организации систем.

Первый вариант. Укрупненная структура системы с ППД показана на рис. 2. Рассмотрим особенности активации команд для такой системы. Команда начинает формироваться в ячейках ППД, два разряда которых (d и a) являются признаками, показывающими наличие в ППД одного операнда D_1 и актора. Запись указанных объектов для i–й команды производится по адресу I_i. При поступлении из $КС_1$ второго операнда D_2 для данной команды, он записывается непосредственно в соответствующие разряды регист-

ра буферной памяти (БП) типа *FIFO*, куда одновременно из ППД переписываются A и D_1.

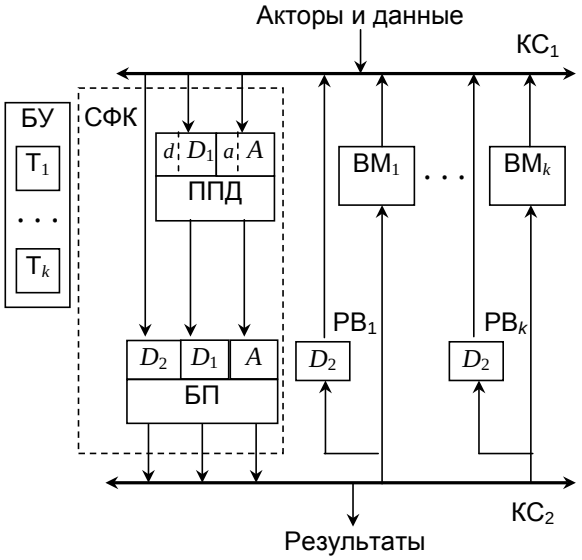

Рис. 2. Организация системы с ППД по первому варианту

Таким образом, в БП формируется готовая для выполнения команда, которая затем через $КС_2$ передается в свободный ВМ. При этом содержимое соответствующей ячейки ППД сохраняется для обеспечения повторного формирования команды в случае отказа ВМ. Одновременно с поступлением команды $\langle A, D_1, D_2 \rangle$ в $ВМ_j$ ее часть D_2, которая не сохраняется в ППД, записывается в регистр временного хранения $РВ_j$ на входной шине соответствующего вычислителя. При успешном выполнении операции, то есть когда время ее выполнения не превысило лимит времени выполнения команды в $ВМ_j$ установленный соответствующим таймером $Т_j$ блока управления БУ, результат операции поступает через $КС_1$ в ППД по адресу N_i, а содержимое соответствующей ячейки ППД по адресу I_i обнуляется.

Если при выполнении операции произошел отказ $ВМ_j$, то есть лимит времени выполнения операции, заданный соответствующим $Т_j$, был превышен, $ВМ_j$ считается отказавшим и блокируется (отключается от $КС_1$ и $КС_2$), а значение D_2 из $РВ_j$ снова передается в СФК. Поскольку значения A и D_1 утеряны не были (соответствующая ячейка ППД не была изменена), то команда повторно записывается в БП, что дает возможность реализовать следующую успешную попытку выполнения команды в исправном ВМ.

К недостаткам описанного метода следует отнести необходимость хранения компонентов команды в ячейках ППД до полного завершения выполнения данной команды, в том числе, с учетом времени реконфигурации системы при отказе ВМ и повторного выполнения команды. Это замедляет вычисления, если соответствующую ячейку памяти необходимо использовать для других команд, например, для очередной итерации или очередной реализации алгоритма в конвейерном режиме.

Второй вариант. Для устранения указанного недостатка предлагается в РВ записывать не только операнд D_2, а всю команду из БП (рис. 3) [11]. При этом принцип формирования команды остается прежним. После записи данных в БП освобождается место в ППД для записи следующих данных. По аналогии с предыдущим вариантом, при отказе $ВМ_j$ он будет блокирован, а данные из $РВ_j$ будут переданы обратно непосредственно в БП, что даст возможность реализовать следующую попытку выполнения команды.

Таким образом, при таком подходе ресурсы системы используются более эффективно. Нет необходимости сохранять фрагменты команды в СФК. Ячейки ППД могут использоваться для других алгоритмов или другой итерации. Это в свою очередь разрешает совмещать вычислительные процессы не только на уровне команд, но и на уровне алгоритмов, что обеспечивает сокращение времени преобразования информации.

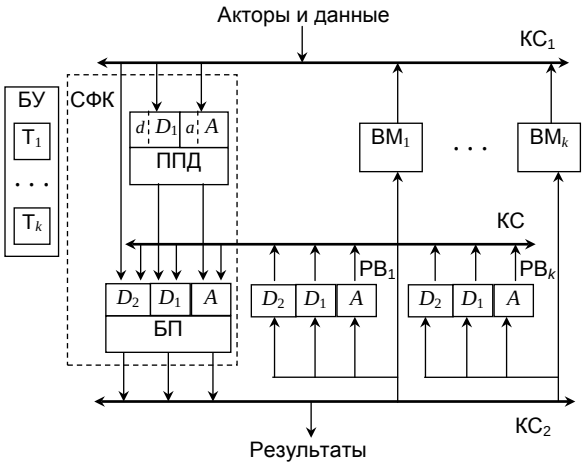

Рис. 3. Организация системы с ППД по второму варианту

Тем не менее, с точки зрения затрат времени на реконфигурацию системы, при отказе ВМ данный подход, как и рассмотренные ранее, является недостаточно эффективным. Действительно, время ожидания срабатывания таймера может быть гораздо больше, чем время, реально необходимое для выполнения команды. Это вносит дополнительную временную задержку при реконфигурации системы.

Третий вариант. Для решения вышеуказанной проблемы формат команды, передаваемой в ВМ, преобразуется к виду: $\langle A, D_1, D_2, T \rangle$, где T – время ожидания результата выполнения конкретной команды (рис. 4). Фактически T и A могут рассматриваться как один информационный объект. Часть РВ выполнена в виде таймера, который запускается непосредственно с записью команды в РВ и ВМ. При этом необходимость таймеров в БУ отпадает.

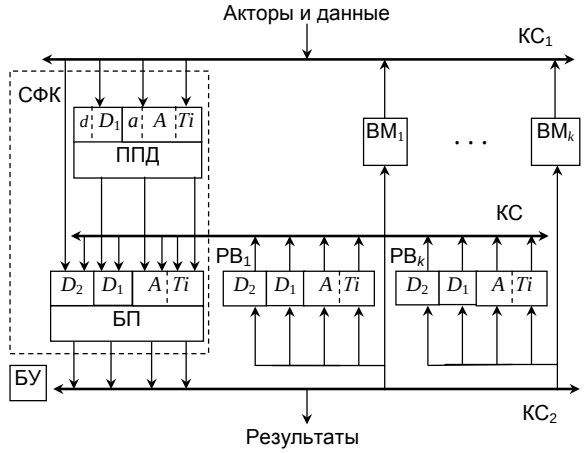

Рис. 4. Организация системы с ППД по третьему варианту

Сравнительная оценка систем с разной организацией СФК

Механизмы повторного выполнения команды при отказе ВМ, описанные во втором и третьем варианте организации систем с ППД, могут быть применены и для системы с АП (рис. 1), то есть системы с АП и ППД имеют сравнимые функциональные возможности. В связи с этим целесообразно сравнить системы по быстродействию и аппаратурным затратам.

В рассматриваемых системах используются разные структуры СФК. Поскольку БП могут иметь одинаковую организацию, то основное отличие СФК заключается в организации АП и ППД. Как известно, АП уступает в быстродействии ППД и требует для построения больших аппаратурных затрат. Это объясняется тем, что кроме адресных операций АП должна выполнять ассоциативный поиск ячеек по признакам. Это, в свою очередь, требует более сложной организации как структуры памяти в целом, так и запоминающих ячеек. Из-за этого увеличивается и цикл обращения к памяти.

Таким образом, системы с ППД содержат меньше оборудования и являются более быстродействующими, чем системы с АП.

Выводы

Предложенные методы реконфигурации и восстановления систем при отказе ВМ реализуются на аппаратном уровне без расхода дополнительных команд. Особенности архитектуры потоковых систем не требуют замены отказавшего модуля резервным путем его подключения в определенное место системы. Если все модули имеют одинаковые функциональные возможности, то команда, которая не выполнилась в отказавшем ВМ, повторно выполняется в другом ВМ, работающем в обычном режиме. В системе сохраняется полная информация для осуществления повторного выполнения команды. Таким образом, реконфигурация системы после отказа ВМ в основном сводится к отключению отказавшего модуля, что не требует больших затрат времени.

При построении систем может быть использован основополагающий принцип построения отказоустойчивых систем – принцип расширяемого ядра, защищенного аппаратными методами повышения надежности [12]. В качестве защищенного ядра в рассматриваемом случае может служить среда формирования команд. Данный компонент системы реализован на основе запоминающих устройств, для которых хорошо разработаны аппаратные методы контроля и исправления ошибок [10]. С учетом указанного предложенные методы автоматической реконфигурации потоковых вычислительных систем для разных сред формирования команд дают возможность продолжать вычисления до тех пор, пока в системе остается хотя бы один работоспособный модуль. Это повышает достоверность результатов вычислений, что является важным фактором, определяющим эффективность применения вычислительных систем.

Таким образом, для реализации алгоритмов с мелкозернистой структурой в реальном времени могут быть эффективно использованы потоковые вычислительные системы, обеспечивающие автоматическое распределение операций между вычислительными модулями и автоматическую реконфигурацию системы при отказах оборудования.

Список литературы

1. Жабин В.И. Архитектура вычислительных систем реального времени. – К.: ВЕК +, 2003. – 176 с.
2. Погребинский С.Б., Стрельников В.П. Проектирование и надежность многопроцессорных ЭВМ. – М.: Радио и связь, 1988. – 168 с.
3. Диллон Б., Сингх И. Инженерные методы обеспечения надежности систем. – М: Мир, 1983. – 318 с.
4. Коваленко А.Е., Гула В.В. Отказоустойчивые микропроцессорные системы. – К.: Техніка, 1986. – 150 с.
5. Dennis J. B., Missunas D. P. A preliminary architecture for basic data flow processor// Proc. 2nd Annual Symp. Comput. Stockholm, May 1975. N. Y. IEEE. – 1975. – P. 126 – 132.
6. Johnson D. Data flow machines threaten the program counter// Electronic Design. – 1980. – N 22. – P. 255 – 258.
7. Silva J.G.D., Wood J.V. Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
8. Watson R., Guard J. A practical data flow computer // Computer. – 1982. – Vol. 15, N 2.– P. 51 – 57.
9. Жабин В.И.. Организация вычислений в системах, управляемых потоком данных // Вісник Національного технічного університету України “Київський політехнічний інститут”, Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+. – 1998. – №31. – С. 44-51.
10. Огнев И.В., Сарычев К.Ф. Надежность запоминающих устройств. – М.: Радио и связь, 1988. – 223 с.
11. Дек. пат. №7727 України, МКВ G06F 15/16, 15/76. Обчислювальній пристрій / І.А. Жуков, В.І. Жабін, І.А. Клименко, В.В. Ткаченко (Україна). – №20040907712: заявлено 22.09.2004; опубл. 15.07.2005. Бюл. №7. – 9 с.
12. Авиженис А. Отказоустойчивость – свойство, обеспечивающее постоянную работоспособность цифровых систем // ТИИЭР (пер. с англ.). – 1978. – Т. 66, №10. – С. 5-25.

Поступила в редакцию 14.12.2009