УДК 681.3
DOI: 10.15827/2311-6749.17.4.13

СЕМЕЙСТВО СИСТЕМ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ БОРТОВЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

В.В. Балашов, к.ф.-м.н., hbd@cs.msu.su
(Московский государственный университет имени М.В. Ломоносова, Ленинские горы, г. Москва, 119991, Россия)

В работе рассмотрены задачи планирования вычислений, возникающие при проектировании бортовых вычислительных систем реального времени, сочетающих модульную и федеративную архитектуры. Приведены подходы к планированию вычислений и информационного обмена в таких системах. Описано семейство инструментальных систем поддержки проектирования БВС РВ, реализующих эти подходы. Предложена схема совместного применения этих инструментальных систем.

Ключевые слова: *системы реального времени, планирование вычислений и информационного обмена, автоматизация проектирования.*

В современных *бортовых вычислительных системах реального времени* (БВС РВ) авиационного, морского и космического назначения широко используются два вида архитектур – федеративная и интегрированная модульная [1].

В соответствии с *интегрированной модульной архитектурой* (ИМА) система строится из набора стандартизованных вычислительных модулей, объединенных коммутируемой средой передачи данных с поддержкой виртуальных каналов (FC-AE-ASM-RT, AFDX). Далее будем называть такие БВС РВ модульными. Аппаратные ресурсы одного вычислительного модуля в модульной БВС РВ могут разделяться между различными функциональными программами (подсистемами), каждая из которых состоит из набора функциональных задач.

В БВС РВ с федеративной архитектурой для каждой подсистемы отводится свой вычислитель, на котором не могут выполняться функциональные задачи других подсистем. Привязка вычислительной нагрузки к аппаратным компонентам ВС фиксирована, а сами компоненты связаны между собой кабельной сетью, построенной в соответствии с рядом унаследованных стандартов и включающей в себя, в частности, каналы «точка-точка» и каналы с централизованным управлением.

Унаследованные БВС РВ в ходе модернизации могут оснащаться высокопроизводительным «ядром» – вычислителем с архитектурой ИМА [2], при этом ряд периферийных подсистем сохраняют федеративную архитектуру. В новых БВС РВ также не всегда целесообразно применение чистой модульной архитектуры. Это связано с невысокой производительностью модульной ВС по сравнению со спецвычислителями на ряде задач по обработке больших массивов данных, для решения которых требуется интенсивный обмен данными между процессорами [3]. Кроме того, некоторые периферийные компоненты БВС РВ не требуют модернизации (для новой БВС РВ могут быть позаимствованы из ранее созданных систем) и/или в соответствии со спецификой управляемого объекта должны быть пространственно распределены (например, контроллеры турбин самолетных двигателей или рулей высоты должны располагаться рядом с управляемыми объектами); замена таких компонентов в рамках модернизации или при создании новой БВС РВ, в т.ч. перемещение их функциональности и реализующего ее ПО в стандартизованные и централизованно расположенные вычислительные модули, не представляется целесообразной.

В связи с перечисленными факторами все более многочисленными становятся БВС РВ, сочетающие федеративную и модульную архитектуры. В данной работе рассматривается ряд задач поддержки проектирования вычислительных систем этого класса (далее будем обозначать их просто БВС РВ), а также разработанные в *Лаборатории вычислительных комплексов* (ЛВК) факультета ВМК МГУ алгоритмы и инструментальные средства решения этих задач. В первую очередь внимание уделяется задачам, связанным с планированием вычислений и конфигурированием *среды передачи данных* (СПД) для обеспечения выполнения функциональных задач и информационного обмена в реальном времени, с учетом технологических ограничений целевой ВС.

Задачи поддержки проектирования БВС РВ

При проектировании БВС РВ, сочетающих федеративную и модульную архитектуры, необходимо решить следующие задачи, имеющие отношение к планированию вычислений:

1) выделение функциональности, которая должна быть реализована в модульном ядре БВС РВ, и определение состава подсистем с федеративной архитектурой, реализующих оставшуюся функцио-

нальность (в дальнейшем будем рассматривать эти подсистемы и отведенные под них вычислители как «черные ящики»);

2) определение структуры вычислительной нагрузки на ядро БВС РВ, включая набор вычислительных задач и информационные связи между ними;

3) оценка вычислительных и сетевых ресурсов ядра (количества модулей, количества и типов процессорных ядер, объема памяти, количества и пропускной способности каналов информационного обмена), необходимых для выполнения вычислительной нагрузки в реальном времени, с соблюдением директивных сроков;

4) определение структуры вычислительных и сетевых ресурсов ядра, включая топологию межмодульной СПД (задача структурного синтеза);

5) распределение вычислительной нагрузки по модулям и процессорным ядрам в составе ядра БВС РВ с учетом интенсивности межзадачного информационного обмена и критичности задержек при доставке различных сообщений;

6) конфигурирование СПД на основе коммутаторов, с построением системы виртуальных каналов и расчетом их параметров;

7) построение расписаний выполнения вычислительных задач на процессорных ядрах в составе ядра БВС РВ;

8) построение расписаний информационного обмена по каналам, связывающим ядро БВС РВ с периферийными подсистемами, имеющими федеративную архитектуру.

Задача 1 может быть решена при помощи подходов, предложенных в [3, 4]. Задача 2 должна решаться индивидуально для каждой БВС РВ с учетом характера производимых вычислений; в рамках исследований, проводимых коллективом с участием автора, ее результат считается известным. Задачи 3 и 4 относятся к перспективным направлениям исследований. Задачи 5–8 более подробно рассматриваются далее; описываются их постановки, схемы алгоритмов решения, а также инструментальные средства поддержки решения.

Задача распределения вычислительной нагрузки по модулям и процессорным ядрам

В БВС РВ с архитектурой ИМА вычислительные задачи каждой функциональной подсистемы группируются в раздел. Каждый раздел привязывается к одному процессорному ядру. К одному и тому же ядру могут быть привязаны несколько разделов. Обмен данными между задачами внутри раздела осуществляется через общую память. Обмен данными между задачами разных разделов осуществляется через механизм сообщений. Сообщения между разделами, привязанными к ядрам одного модуля, передаются через память модуля. Сообщения между разделами, привязанными к ядрам разных модулей, передаются через сеть.

Задача распределения вычислительной нагрузки имеет следующую постановку.

Исходные данные

Описание ВС с модульной архитектурой (в данном случае – ядра БВС РВ):

– набор вычислительных модулей;

– для каждого модуля число и типы процессоров;

– для каждого типа процессора число ядер (все ядра в процессоре однотипны), задержки на переключение контекста между задачами;

– для каждого процессорного ядра максимально допустимое значение его загрузки.

Описание вычислительной нагрузки (задачи, сообщения):

– набор вычислительных задач;

– для каждой задачи период, приоритет, наихудшая (максимальная) длительность выполнения (возможно, различная для различных типов процессоров);

– набор разделов;

– для каждого раздела набор задач, принадлежащих ему (наборы не пересекаются), набор процессорных ядер, к которым раздел может быть привязан – при помощи этого ограничения может быть учтена потребность задач раздела в функциях, доступных только на специализированных модулях (например, графический сопроцессор на модуле графического контроллера);

– набор сообщений;

– для каждого сообщения задача-отправитель, задача-получатель, размер (в байтах); период передачи сообщения равен периоду задачи-отправителя.

Также для сообщения может быть задан признак синхронности. Задача-получатель синхронного сообщения может стартовать в очередной свой период только после получения этого сообщения; периоды задачи-отправителя и задачи-получателя синхронного сообщения должны совпадать.

Выходные данные (решение): привязка разделов к процессорным ядрам.

Ограничения на решение:

– каждый раздел должен быть привязан к процессорному ядру, допустимому для привязки этого раздела;

– для каждого процессорного ядра его загрузка не должна превышать заданного ограничения;

– каждый раздел должен быть привязан к единственному ядру.

Данная задача может быть рассмотрена как задача оптимизации. В качестве *минимизируемого критерия* предлагается рассматривать нагрузку на СПД, вычисляемую как суммарный объем сообщений, передаваемых через СПД за интервал планирования (равный наименьшему общему кратному периодов задач). Сообщение передается через СПД только в случае, если задача-отправитель и задача-получатель привязаны к различным модулям ВС; в противном случае сообщение передается через локальную память модуля. Таким образом, для различных распределений задач по модулям и процессорным ядрам нагрузка на СПД различается, и необходимо найти оптимальное допустимое (не нарушающее ограничений) распределение.

Задача конфигурирования СПД на основе коммутаторов с поддержкой виртуальных каналов

Эта задача решается после задачи распределения вычислительной нагрузки по модулям и процессорным ядрам. При известном распределении вычислительной нагрузки известен также набор передаваемых через СПД сообщений, и для каждого сообщения – модуль-отправитель и модуль-получатель. Модули рассматриваются в качестве абонентов СПД, каждый из которых подключен к порту одного из коммутаторов. Механизм виртуальных каналов в СПД обеспечивает предсказуемость задержек при передаче сообщений между модулями.

Задача имеет следующую постановку [5].

Исходные данные:

– набор сообщений;

– для каждого сообщения:

- размер;
- период передачи (равен периоду задачи-отправителя);
- абонент-отправитель;
- набор абонентов-получателей;
- максимально допустимая длительность передачи сообщения с начала выдачи абонентом-отправителем до получения всеми абонентами-получателями;
- опционально: максимально допустимый джиттер передачи сообщения (разность между максимальной и минимальной длительностью передачи сообщения).

В сетях AFDX и FC-AE-ASM-RT, используемых в современных ВС с архитектурой IMA, на абоненте-отправителе сообщение разбивается на кадры, и СПД выполняет передачу кадров абонентам-получателям по маршруту, соответствующему виртуальному каналу, к которому относится сообщение. Сообщение считается принятым абонентом-получателем, если приняты все его кадры.

Максимально допустимая длительность передачи сообщения через СПД оценивается исходя из периода выполнения задачи-отправителя; в дальнейшем (в случае неуспешного построения расписания выполнения задач) она может уточняться в сторону снижения, то есть усиления ограничения.

Выходные данные (решение):

– набор виртуальных каналов;

– для каждого виртуального канала:

- абонент-отправитель;
- набор абонентов-получателей;
- маршрут от отправителя к получателям через коммутаторы и каналы СПД (при наличии нескольких получателей маршрут имеет вид дерева);
- максимальный размер кадра, передаваемого по данному виртуальному каналу;
- для сетей AFDX – эталонный интервал времени (bandwidth allocation gap – BAG) между отправками оконечной системой в канал последовательных кадров данного виртуального канала;

– для каждого сообщения, передаваемого через СПД: верхние оценки задержки джиттера передачи и длительности передачи.

Выходные данные используются для программирования коммутаторов в составе СПД.

Пропускная способность СПД, резервируемая под виртуальный канал, рассчитывается на основе размеров и периодов сообщений, передаваемых через этот канал, максимального размера кадра, а для сетей AFDX – также параметра BAG.

Ограничения на решение:

– для каждого сообщения длительность его передачи через виртуальный канал не должна превышать заданного ограничения;

– для каждого канала в составе СПД суммарная пропускная способность, зарезервированная под проходящие через него виртуальные каналы, должна быть не ниже пропускной способности этого канала.

Под каналами имеются в виду каналы между коммутаторами, а также между абонентами и коммутаторами.

Данную задачу будем рассматривать как задачу удовлетворения ограничениям без введения оптимизируемого критерия.

Задача построения расписания выполнения вычислительных задач на процессорных ядрах

Эта задача решается после задачи распределения вычислительной нагрузки по модулям и процессорным ядрам (так как для каждого ядра должен быть определен набор привязанных к нему разделов), а также после задачи конфигурирования СПД (так как должны быть определены задержки на передачу сообщений через СПД).

В соответствии со стандартом ARINC 653 на системы с архитектурой ИМА, для каждого процессорного ядра должно быть построено статическое расписание окон – временны*х* интервалов, в рамках каждого из которых могут выполняться задачи одного конкретного раздела. Внутри окна работает динамический планировщик задач этого раздела. Для этой схемы планирования и будем рассматривать задачу построения расписания выполнения вычислительных задач.

Задача построения расписания имеет следующую постановку.

Исходные данные:

– исходные данные и результаты решения задачи распределения вычислительной нагрузки по модулям и процессорным ядрам;

– для каждого сообщения, передаваемого через СПД, верхняя оценка длительности его передачи через СПД;

– для модульной ВС в целом максимально допустимая и минимально допустимая длительность окна.

Задача имеет следующие *выходные данные (решение)*:

– для каждого ядра набор окон выполнения разделов;

– для каждого окна время начала, время завершения, раздел, сопоставленный окну (допустимы также «пустые» окна, к которым не привязан никакой раздел).

Ограничения на решение:

– окна для каждого ядра не должны перекрываться;

– к каждому окну привязан единственный раздел;

– длительность каждого окна должна быть не меньше минимально допустимой и не больше максимально допустимой;

– границы окон для всех ядер одного модуля должны совпадать (ограничение связано с тем, что на модуле работает планировщик окон, единый для всех ядер);

– расписание окон должно гарантировать выполнение всех задач в рамках директивных сроков (определяемых периодами задач) в «наихудшем» сценарии – при условии, что выполнение каждой задачи и передача каждого сообщения через СПД имеет длительность, равную соответствующей верхней оценке.

Данную задачу будем рассматривать как задачу удовлетворения ограничениям, без введения оптимизируемого критерия. Возможно также рассмотрение задачи как оптимизационной, с минимизацией числа переключений контекста; переключение контекста имеет место только при смене выполняемого на ядре раздела.

Задача построения расписаний информационного обмена по каналам, связывающим модульное ядро БВС РВ с периферийными подсистемами, имеющими федеративную архитектуру

Эта задача решается, как правило, независимо от описанных выше задач. В модульном ядре БВС РВ присутствует выделенный модуль ввода-вывода, отвечающий за обмен данными с периферийными подсистемами [1]. Этот обмен выполняется асинхронно с выполнением вычислительных задач на прочих модулях. Требуется обеспечить необходимую частоту обмена с периферийными подсистемами.

Типичными каналами информационного обмена между ядром БВС РВ и периферийными подсистемами являются каналы «точка-точка» (например, ARINC 429) и каналы с централизованным управлением (например, мультиплексный канал информационного обмена МКИО). Содержательная задача планирования информационного обмена имеет место только для каналов с централизованным управлением, обмен по которым идет под управлением контроллера канала (эту функцию выполняет модуль ввода-вывода в составе ядра). Остальные абоненты канала отрабатывают команды контроллера, выдавая и/или принимая сообщения.

Контроллер канала функционирует в соответствии с заданным статическим расписанием. Типичной схемой организации обмена по каналу МКИО является циклическая схема (рис. 1), в которой интервал планирования разбивается на интервалы равной длительности – подциклы [6]. В каждом из подциклов выполняется одна цепочка обменов, то есть последовательность передач сообщений, выполняемых без пауз. Эта схема определяется ограничениями используемых аппаратных контроллеров канала и управляющего ими системного ПО. В начале и конце каждого подцикла резервируется время, соответственно, для программирования контроллера на выдачу очередной цепочки и для повторного выполнения отдельных обменов в случае помех на канале.

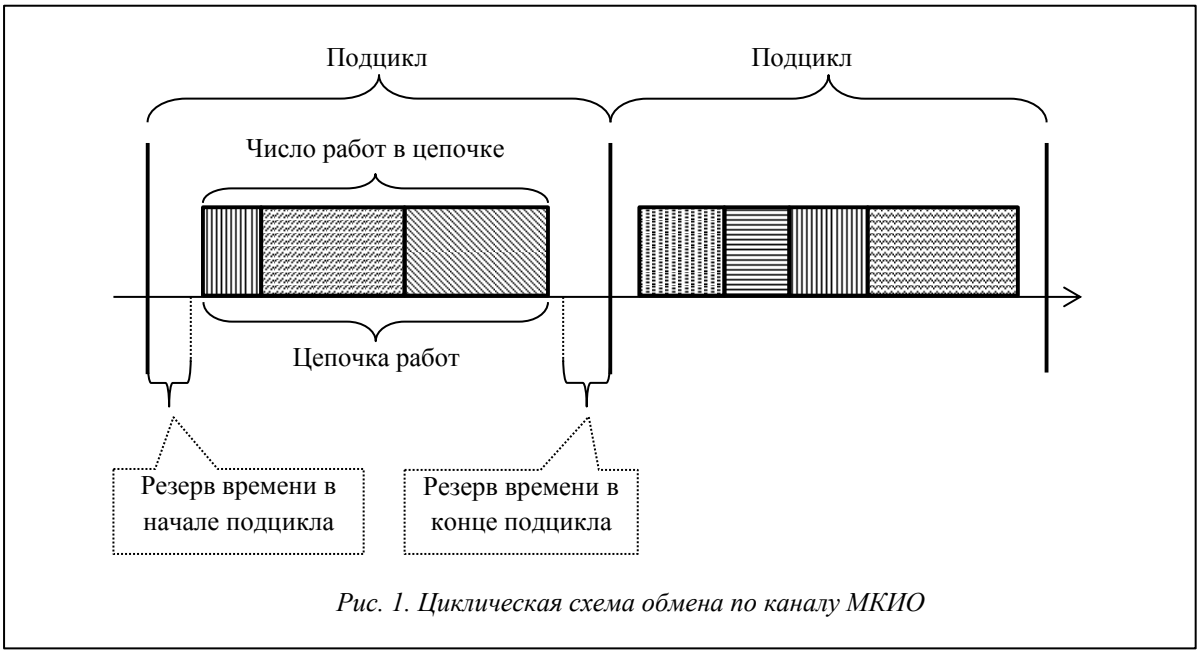

Рис. 1. Циклическая схема обмена по каналу МКИО

Задача построения расписания обмена по каналу с централизованным управлением имеет следующую постановку.

Исходные данные:

– набор сообщений;

– для каждого сообщения: период передачи, длительность передачи;

– значения технологических ограничений: длительность подцикла, резерв времени в начале подцикла, резерв времени в конце подцикла, максимально допустимое число обменов в цепочке.

Выходные данные (решение):

– набор цепочек обменов;

– для каждой цепочки:

- время старта;
- последовательность сообщений, передаваемых в рамках цепочки.

Ограничения на решение:

– каждое сообщение должно передаваться в рамках директивных сроков, определяемых его периодом (то есть не реже одного раза в период);

– должны выполняться технологические ограничения:

- в каждом подцикле должно выполняться не более одной цепочки обменов;
- зарезервированные интервалы в начале и в конце каждого подцикла не должны использоваться для выполнения обменов из цепочек;
- в каждой цепочке обмены должны следовать друг за другом без пауз;
- число обменов в каждой цепочке не должно превышать максимально допустимого.

Данную задачу будем рассматривать как задачу удовлетворения ограничениям без введения оптимизируемого критерия.

Существующие средства решения рассматриваемых задач

Задача распределения вычислительной нагрузки по модулям и процессорным ядрам схожа с *задачей о мультипликативном рюкзаке* (ЗМР). В рамках решения ЗМР необходимо распределить объекты по контейнерам так, чтобы суммарная стоимость распределенных объектов была максимальной, а емкость каждого из контейнеров не была превышена [7].

При распределении вычислительной нагрузки объектам соответствуют разделы, а контейнерам – процессорные ядра. Объему объекта соответствует вклад раздела в загрузку процессорного ядра; стоимости объекта – часть трафика между разделами, которая становится «внутренней» для модуля (не передаваемой через межмодульную СПД) при размещении объекта в контейнер, то есть при назначении раздела на процессорное ядро в составе модуля. У такой постановки задачи есть и отличия от классической ЗМР. Во-первых, объем объекта зависит от выбора контейнера, так как задача может иметь различные наихудшие времена выполнения для различных типов процессоров. Во-вторых, стоимость объекта зависит от содержимого контейнера, то есть от набора других объектов, размещенных в контейнер. В связи с этими отличиями существующие алгоритмы решения ЗМР не могут быть использованы «как есть».

В качестве основы для решения задачи распределения вычислительной нагрузки были выбраны следующие подходы к решению ЗМР:

– эвристический алгоритм с жадной схемой – распространенный способ быстрого поиска приемлемого решения для задачи с не очень жесткими ограничениями; пример использования такого алгоритма для решения ЗМР можно найти в [8, 9];

– метод ветвей и границ – точный метод с экспоненциальной сложностью относительно числа контейнеров и объектов, время работы и масштабируемость которого существенно зависят от мер по сокращению пространства перебора; применяется к ЗМР в [10];

– генетические алгоритмы – обладают хорошей масштабируемостью (в сравнении с методом ветвей и границ) и способностью избегать тупиков при поиске решения (в отличие от жадных алгоритмов); применяются к ЗМР в [7].

Разработанные на основе этих подходов алгоритмы более подробно рассмотрены далее.

Среди опубликованных подходов к конфигурированию СПД на основе коммутаторов с поддержкой виртуальных каналов следует отметить работы [11] и [12], в которых достаточно детально описаны предлагаемые алгоритмы (в отличие от описаний коммерческих систем, где алгоритмы не раскрываются). В работе [11] используется возможность задавать статический приоритет виртуальных каналов на портах коммутаторов для оптимизации времени передачи кадров. Такая возможность выходит за рамки стандарта AFDX и отсутствует во многих коммутаторах, в том числе в используемых на борту разрабатываемого в настоящее время отечественного авиалайнера МС-21. В работе [12] приведен оптимальный алгоритм проектирования виртуального канала для передачи по нему не более одного сообщения с заданными ограничениями на время выдачи последнего кадра в сеть (то есть время фактической передачи кадра по сети не учитывается). В работе также предложены процедура агрегации и алгоритм построения маршрутов виртуальных каналов на основе решения задачи Штейнера с помощью смешанного целочисленного программирования; предложенный алгоритм не учитывает ограничения на время и джиттер передачи сообщений через сеть, существенные при проектировании модульных БВС РВ; учитываются только ограничения на время выдачи сообщения в канал с оконечной системы.

При построении расписания выполнения вычислительных задач на процессорных ядрах БВС РВ, построенных по архитектуре ИМА, непосредственно строится расписание окон выполнения разделов (partition schedule, англ.). Для построенного расписания окон должна быть проведена проверка корректности, то есть того, что в наихудшем случае все вычислительные задачи будут выполнены в рамках директивных сроков.

Известен ряд работ по построению расписаний окон, в которых целевая вычислительная система и ее рабочая нагрузка рассматриваются на различных уровнях абстракции. В работе [13] рабочая нагрузка представляется в виде набора периодически выполняемых разделов, зависимости по данным между которыми не учитываются; отдельные задачи в составе разделов также не рассматриваются. Задача построения расписания решается методом удовлетворения ограничений (constraint satisfaction). В работе [14] добавляется учет зависимостей по данным между целыми разделами. Для построения расписания используется целочисленное линейное программирование. В работе [15] система рассматривается уже на уровне отдельных периодических задач, сгруппированных в разделы. При этом учет зависимостей между задачами очень ограничен – авторы предлагают выражать зависимости через явное задание директивных интервалов, не приводя методики расчета этих интервалов по графу зависимостей. Также имеет место требование кратности периодов. В работе [16] вводится сложная формальная модель для поддержки многоуровневого планирования, основанная на разделяемых ресурсах. При этом зависимость между задачами по данным не может быть выражена в рамках этой модели, так как не поддерживается задание ограничений на последовательность доступа к ресурсам (пример ограничения – «получатель может получить доступ к ресурсу «сообщение» только после отправителя»).

Практическое применение результатов перечисленных выше работ для построения расписаний окон затрудняется тем, что при разработке реальных систем с архитектурой ИМА необходимо учитывать все указанные выше факторы: характеристики отдельных задач, зависимости по данным, некратность периодов. Предложенные автором данной работы и его коллегами алгоритмы построения расписаний, учитывающие эти факторы, описаны в следующем разделе.

Следует также упомянуть алгоритмы построения статико-динамических расписаний, предложенные рабочей группой Института системного программирования РАН [17, 18]. Это переборные алгоритмы, использующие ряд фактов из теории чисел для ограничения области поиска. Алгоритмы не учитывают зависимости между задачами и основываются на предположении о кратности периодов задач. Также в постановке задачи требуется, чтобы каждая из них стартовала строго в начале своего периода, что не является требованием, характерным для реальных БВС РС.

Для проверки корректности уже построенных расписаний окон могут использоваться такие инструментальные системы, как Rapid RMA [19], Cheddar [20], MAST [21]. В их состав входят модули расширения (плагины), поддерживающие анализ корректности расписаний окон для ВС с архитектурой ИМА. При этом задачу построения расписания эти системы не решают.

Задача построения расписаний информационного обмена по каналам с централизованным управлением, соединяющим модульное ядро БВС РВ с периферийными устройствами, имеющими федеративную архитектуру, из четырех рассмотренных задач является наиболее близкой к классическим задачам построения расписаний [22]. В ее рамках планированию подлежат непосредственно выполняемые работы, а не их группы (разделы). Применимость общеизвестных алгоритмов построения расписаний к данной задаче затрудняется наличием технологических ограничений на расписание. Алгоритмы, минимизирующие суммарное запаздывание работ (например, описанные в [23]), не могут быть применены для построения расписания обмена по каналу с централизованным управлением, так как не поддерживают ограничения, не представимые в виде зарезервированных интервалов времени (например, ограничение сверху на число обменов в цепочке). Алгоритмы поиска максимального потока в сети [22, 24], а также ряд алгоритмов, основанных на ограниченном переборе или жадной стратегии [25–27], не могут быть применены, так как требуют поддержки вытеснения работ, отсутствующей на каналах передачи данных с централизованным управлением.

Предложенный в работе [28] жадный алгоритм не требует поддержки вытеснения работ, а также допускает расширение для поддержки технологических ограничений. Созданный на его основе алгоритм построения расписания обмена по каналу с централизованным управлением более подробно рассмотрен далее.

Разработанные инструментальные системы поддержки проектирования БВС РВ

САПР «Планировщик ИМА»

Для решения задачи распределения вычислительной нагрузки по модулям и процессорным ядрам, а также задачи построения расписания вычислений разработана САПР «Планировщик ИМА». Типовая последовательность действий при применении этой САПР следующая:

1) создание проекта: ввод или импорт из внешних источников (структурированных файлов) информации о структуре вычислительной системы с архитектурой ИМА (например, модульного ядра БВС РВ), а также о характеристиках рабочей нагрузки на ВС (набор задач и сообщений);

2) автоматическое построение привязки разделов к процессорным ядрам с минимизацией загрузки СПД и соблюдением ограничений на загрузку процессорных ядер;

3) при необходимости ручная корректировка привязки разделов к ядрам;

4) построение расписаний вычислений (то есть наборов окон) для процессорных ядер с учетом задержек на передачу сообщений через СПД;

5) при необходимости ручная корректировка границ окон;

6) проверка корректности построенного расписания вычислений посредством имитационного моделирования выполнения рабочей нагрузки с учетом заданных схем динамического планирования задач внутри разделов, длительностей выполнения задач, задержек на передачу сообщений через СПД;

7) формирование отчетов по исходным данным и результатам применения САПР для включения в документацию по системе ИМА;

8) экспорт построенного расписания вычислений во внешние файлы для включения в состав конфигурации операционной системы реального времени (ОС РВ).

Информация о задержках на передачу сообщений через СПД, необходимая для выполнения шага 4, берется из внешних источников, например, рассчитывается средствами конфигурирования СПД с учетом распределения вычислительной нагрузки по модулям ВС.

САПР «Планировщик ИМА» поддерживает инкрементальный режим распределения вычислительной нагрузки, при котором пользователем фиксируется привязка отдельных разделов к процессорным ядрам, и автоматически конструируемая полная привязка должна включать в себя заданную фиксированную часть. Это соответствует итеративному подходу к разработке ПО ВС с архитектурой ИМА, при котором вычислительная нагрузка на систему наращивается поэтапно.

В САПР «Планировщик ИМА» реализованы средства визуализации исходных данных и результатов работы САПР, а именно:

– набора задач и их зависимостей по данным (в виде графа зависимостей);

– привязки разделов к модулям и процессорным ядрам в их составе, а также интенсивности потоков данных между разделами;

– временной диаграммы работы целевой ВС (по результатам моделирования), включая моменты открытия и закрытия окон; старта, прерывания и завершения задач; выдачи и приема сообщений.

Снимки экрана, демонстрирующие перечисленные виды визуализации данных, приведены на рисунках 2–4.

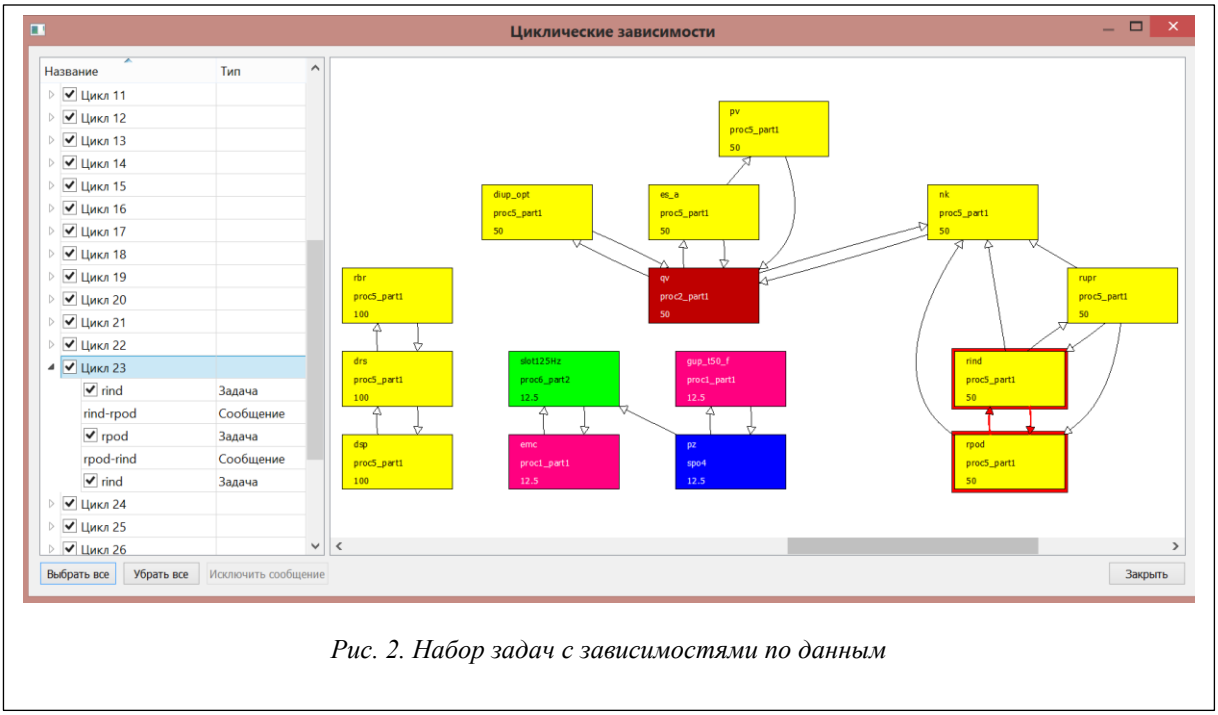

Рис. 2. Набор задач с зависимостями по данным

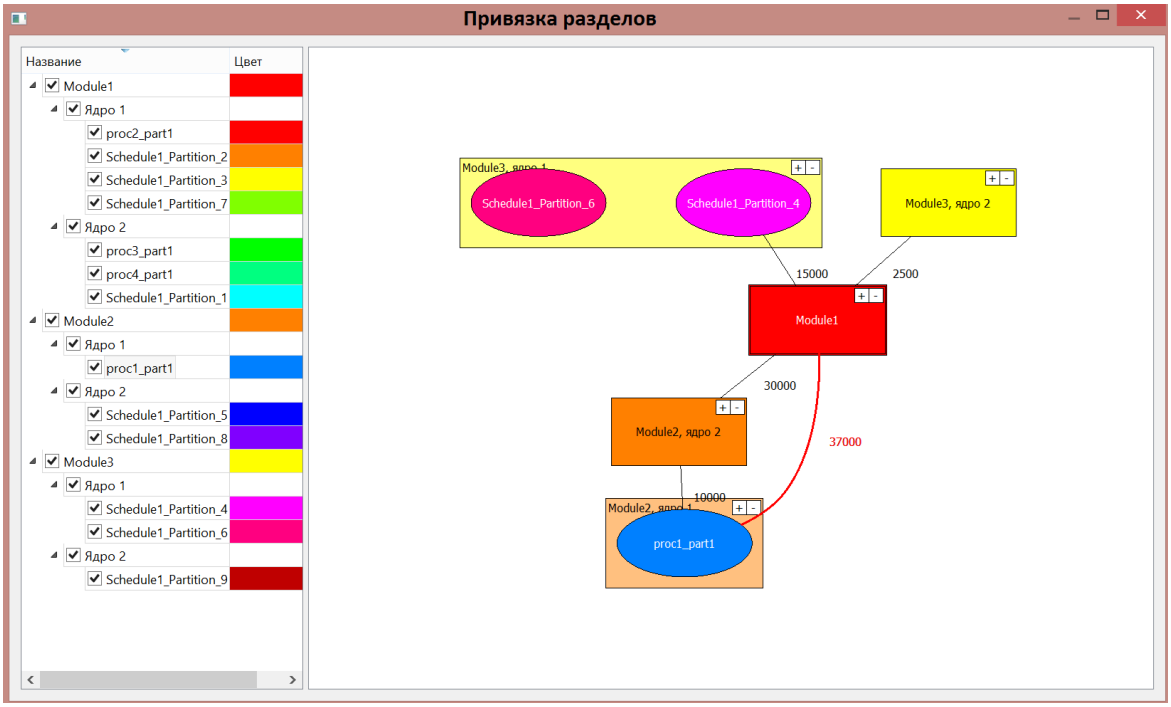

Рис. 3. Привязка разделов к модулям и ядрам

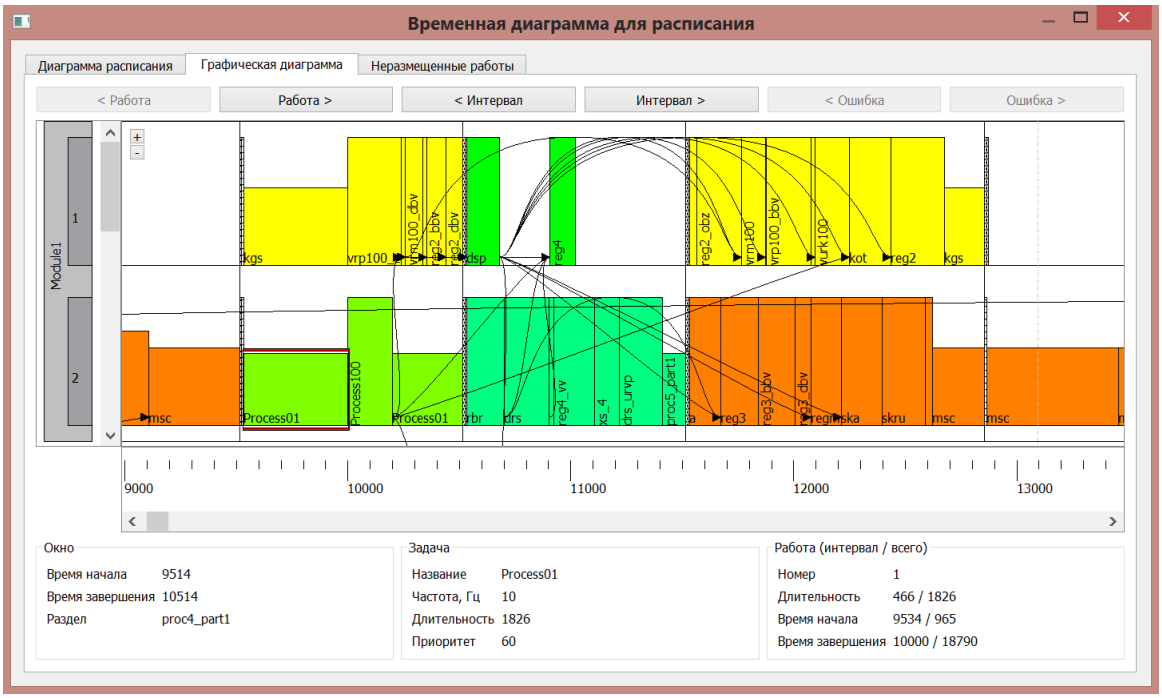

Рис. 4. Расписание вычислений на ядрах ВС

САПР «Планировщик ИМА» реализована на языке C++ и функционирует под управлением ОС Windows или Linux. На рисунке 5 показана структура САПР – набор подсистем и информационные связи между ними.

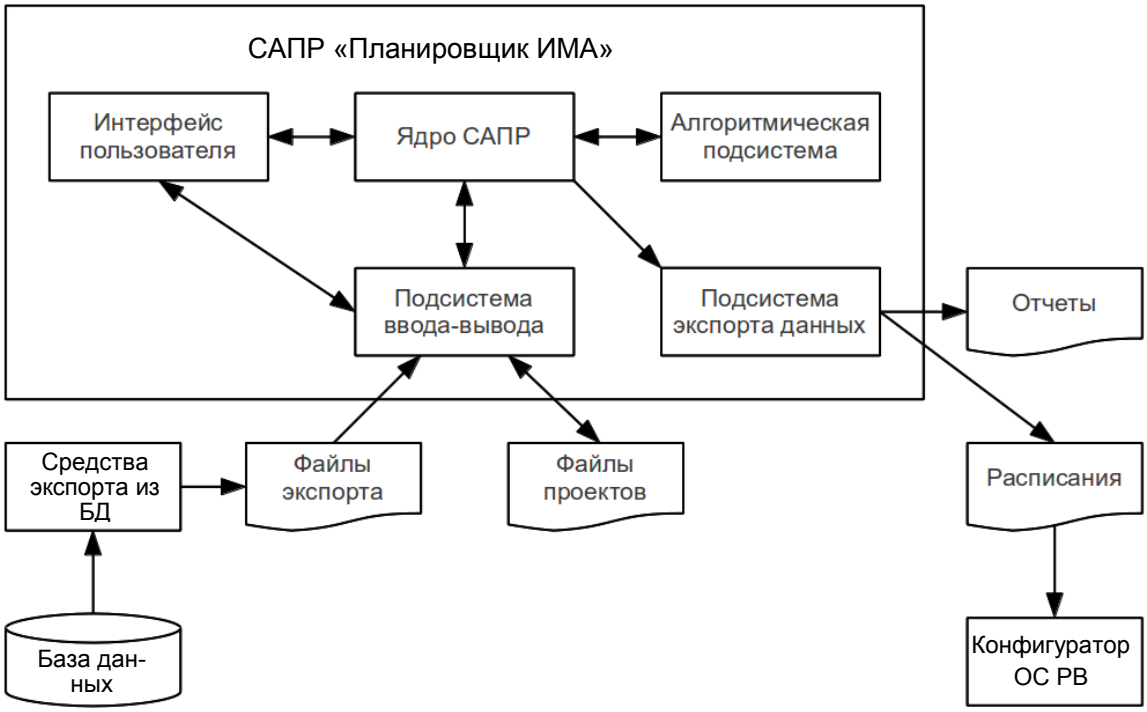

Рис. 5. Структура САПР «Планировщик ИМА»

Для распределения вычислительной нагрузки (разделов) по модулям и процессорным ядрам в САПР «Планировщик ИМА» реализованы алгоритмы, основанные на подходах, описанных ранее: эвристиче-

ский алгоритм с жадной схемой, метод ветвей и границ, генетические алгоритмы. Каждый из реализованных алгоритмов минимизирует одну и ту же целевую функцию – суммарный объем сообщений, передаваемых через СПД между модулями на протяжении интервала планирования.

Жадный алгоритм стремится минимизировать целевую функцию посредством назначения наиболее интенсивно взаимодействующих разделов на процессорные ядра одних и тех же модулей, пока это возможно без нарушения ограничений на загрузку ядер. В качестве локального критерия при выборе очередного раздела и модуля для его размещения служит объем трафика, который становится «внутренним» для модуля при размещении раздела на одно из процессорных ядер этого модуля. В случае невозможности привязать раздел ни к одному из ядер выбранного модуля производится попытка перераспределить разделы между ядрами модуля с тем, чтобы высвободить на одном из них достаточное количество процессорного времени.

Реализованный жадный алгоритм с элементами ограниченного перебора позволяет быстро (за единицы секунд для реалистичных ВС) найти решение задачи распределения рабочей нагрузки. Его недостатком является то, что для высоконагруженных систем и/или систем с большим (несколько десятков и более) числом процессорных ядер и разделов алгоритм находит существенно неоптимальные решения или вообще не находит допустимых решений.

Для высоконагруженных систем со средним (20–25) числом процессорных ядер и разделов в САПР «Планировщик ИМА» применяется алгоритм на основе метода ветвей и границ. Этот алгоритм пошагово достраивает распределение вычислительной нагрузки, неявно обходя дерево частичных решений. Корнем дерева является пустое частичное решение. Дуге от одной вершины к другой соответствует назначение некоторого раздела на некоторое процессорное ядро. Дуги из вершин уровня N соответствуют назначению N-го раздела (в некоторой фиксированной нумерации). K-я дуга из вершины соответствует назначению ядра на K-е ядро в составе ВС (для сквозной нумерации ядер). Поддерево отсекается и не подлежит обходу, если все решения в его составе гарантированно не лучше какого-либо из ранее найденных корректных решений; необходимо отметить, что достраивание частичного решения может только ухудшить значение целевой функции (за счет добавления новых сообщений, передаваемых через СПД), в связи с чем значение целевой функции в промежуточной вершине (корне поддерева) является оценкой снизу для всех значений этой функции в вершинах поддерева и может быть использовано для принятия решения об отсечении поддерева.

Метод ветвей и границ является точным, но его применимость существенно зависит от качества сужения области поиска, которое, в свою очередь, зависит от порядка обхода дерева частичных решений. Для повышения качества сужения области поиска разделы упорядочиваются в соответствии с критерием, похожим на используемый в жадном алгоритме. Список разделов составляется так, что на N-ю позицию ставится раздел, наиболее интенсивно суммарно взаимодействующий с (N-1) разделами, уже включенными в список. Этот критерий определяет порядок следования ярусов в дереве.

В алгоритме на основе метода ветвей и границ также используется жадный критерий для выбора дуги из текущей вершины: дуги упорядочиваются по убыванию взаимодействия (то есть объема трафика) между текущим назначаемым разделом и разделами, ранее привязанными к ядру, соответствующему концу дуги. Здесь текущий раздел определяется ярусом вершины, а ранее привязанные разделы соответствуют пути от корня дерева до этой вершины.

Недостатком данного алгоритма является его низкая масштабируемость: при числе разделов и процессорных ядер, большем 25, время работы алгоритма превышает 10^5 с (больше 1 суток). Для поддержки перспективных ВС с архитектурой ИМА (от 50 до 100 разделов и ядер) необходима лучшая масштабируемость.

В настоящее время коллектив ЛВК ведет работы по улучшению масштабируемости алгоритма на основе метода ветвей и границ. В то же время неплохие показатели точности и особенно масштабируемости продемонстрировал *генетический алгоритм* (ГА) распределения вычислительной нагрузки. Общая схема ГА описана в [29, 30].

Чтобы применить ГА к конкретной оптимизационной задаче, необходимо определить следующие его атрибуты:

- функция выживаемости;
- кодировка решения;
- операции селекции, скрещивания и мутации;
- условие останова.

В качестве функции выживаемости была выбрана целевая функция задачи распределения вычислительной нагрузки, а именно суммарный объем сообщений, передаваемых через СПД на протяжении интервала планирования. Решение задачи кодируется в виде массива, номер элемента в котором соответствует номеру раздела, а значение элемента – номеру ядра, на которое назначен раздел. Данная кодировка выбрана из соображений простоты операций скрещивания и мутации. Селекция осуществляется по турнирной схеме, которой в меньшей мере, чем пропорциональной и рулеточной схемам, свойственна

преждевременная сходимость к локальному оптимуму. Используется операция однородного скрещивания, позволяющая популяции быстрее (по сравнению с одноточечным скрещиванием) распространяться на ранее не охваченные области множества решений.

Чтобы избежать частого формирования недопустимых решений (с превышением допустимой загрузки процессорных ядер), используется сочетание простой мутации (случайно выбранный раздел перемещается на другое ядро) и мутации обмена (если при простой мутации происходит перегрузка ядра-получателя раздела, другой раздел перемещается с ядра-получателя на исходное ядро). Алгоритм останавливается после выполнения заранее заданного числа итераций. Перечисленные выше виды операций также описаны в [29].

Для повышения скорости работы ГА на многоядерных компьютерах реализована островная схема алгоритма [31], при которой популяция разделяется на несколько групп; в каждой группе выполняется фиксированное количество обычных итераций ГА; затем осуществляется миграция решений между группами. Эти шаги повторяются циклически, пока не будет достигнуто заданное ограничение на суммарное число обычных итераций.

Построение расписаний окон для процессорных ядер ВС осуществляется по следующей схеме:

– при помощи алгоритма планирования с фиксированными приоритетами для заданного интервала планирования строится вспомогательное многопроцессорное статическое расписание выполнения задач (на каждом ядре – для задач тех разделов, которые привязаны к этому ядру); при этом учитываются синхронные зависимости по данным и задержки на передачу сообщений между модулями через СПД;

– параллельно со вспомогательным расписанием на основании моментов старта и завершения задач из него строится расписание окон: новое окно на процессорном ядре начинается в момент старта задачи из раздела, не совпадающего с разделом, к которому относится предыдущая выполнявшаяся на этом ядре задача;

– при построении вспомогательного расписания учитывается ограничение на длительность окна: на интервале, меньшем или равном минимальной длительности окна, недопустимо выполнение на одном ядре задач разных разделов.

Также при работе алгоритма, построенного по этой схеме, учитываются ограничение на максимальную длительность окна и требование совпадения границ окон для всех процессорных ядер модуля. Для обоих указанных случаев в алгоритме предусмотрено открытие новых окон.

Формализованное описание используемого в САПР «Планировщик ИМА» алгоритма построения окон приведено в работе [32]. Для проверки корректности построенного расписания вычислений посредством имитационного моделирования выполнения рабочей нагрузки применяется как встроенная в САПР имитационная модель, так и формализованная модель [33] на основе аппарата временных автоматов с остановкой таймеров.

САПР «Планировщик ИМА» прошла апробацию при планировании вычислений в модульном ядре БВС РВ самолета-истребителя 5-го поколения [2]. Количественные характеристики вычислительной нагрузки и целевой ВС допускали применение алгоритма на основе метода ветвей и границ, таким образом, было найдено оптимальное распределение вычислительной нагрузки по модулям и процессорным ядрам. Расписание окон также было построено успешно. Жадный алгоритм показал не более чем 15 %-ное отклонение от оптимального решения по значению целевой функции.

Для исследования масштабируемости алгоритмов распределения вычислительной нагрузки были выполнены прогоны алгоритмов на синтетических наборах исходных данных, автоматически сформированных на основе данных по реальной БВС РВ. Исследование показало, что, начиная с количества разделов и процессорных ядер, равного 25±1, отклонение решения, находимого жадным алгоритмом, от оптимального, становится существенным (более 20 %, и в дальнейшем резко возрастает). Примерно на такой же размерности целевой системы становится неприемлемо высоким время работы алгоритма на основе метода ветвей и границ. В то же время ГА сохраняет приемлемую длительность работы (порядка 20–30 с при размере популяции в 1 000 решений и ограничении на число итераций, равном 1 000) для всех рассмотренных наборов входных данных. Будучи рандомизированным алгоритмом, ГА продемонстрировал достаточно высокую стабильность – разброс значений целевой функции на решениях, находимых для одного и того же набора входных данных, не превышал 15 % для систем с размерностью (числом разделов и ядер) до 150. Для наборов входных данных, для которых известно (найдено методом ветвей и границ) оптимальное решение, ГА находил решение, отклоняющееся от оптимального не более чем на 10 %. Результаты исследования масштабируемости ГА и алгоритма на основе метода ветвей и границ показаны на рисунке 6.

При всей перспективности ГА необходимо отметить, что в силу своей рандомизированности и невозможности предсказать точность получаемого решения (возможны только вероятностные оценки) этот алгоритм воспринимается скептически разработчиками реальных БВС РВ. В связи с этим основным направлением дальнейших исследований является повышение масштабируемости алгоритма на основе метода ветвей и границ.

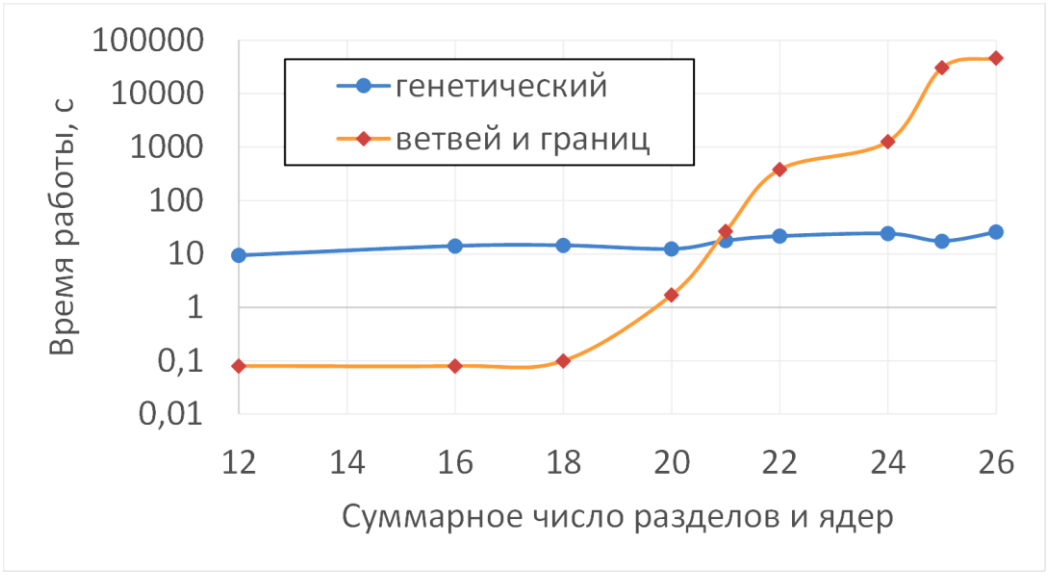

Рис. 6. Масштабируемость алгоритмов распределения вычислительной нагрузки

САПР AFDX

Для решения задачи конфигурирования СПД на основе коммутаторов с поддержкой виртуальных каналов коллективом ЛВК разработана инструментальная система САПР AFDX. Типовая последовательность действий при применении этой САПР следующая:

1) создание проекта: ввод или импорт из внешних источников (структурированные файлы) информации о структуре бортовой сети на основе коммутаторов с поддержкой виртуальных каналов (поддерживаются сети AFDX, но возможно расширение функциональности САПР для поддержки других аналогичных сетей, в т.ч. FC-AE-ASM-RT), а также о характеристиках рабочей нагрузки на сеть (набор сообщений);

2) автоматическое построение набора виртуальных каналов (включая их маршруты) с соблюдением ограничений на времена передачи и джиттеры сообщений;

3) при необходимости ручная корректировка параметров и/или маршрутов виртуальных каналов;

4) экспорт построенного набора виртуальных каналов во внешние файлы для последующей настройки коммутаторов и оконечных систем AFDX.

САПР AFDX [34] реализована в виде программного прототипа и может быть доработана до полнофункционального программного инструмента за счет реализации функциональности генерации отчетов в требуемом формате, учета специфики конкретных моделей коммутаторов в составе целевой БВС РВ, а также реализации инкрементального режима построения виртуальных каналов без модификации заданного (ранее построенного) набора существующих каналов.

В САПР AFDX реализована функция визуального редактирования структуры бортовой сети, а также визуализации маршрутов построенных виртуальных каналов. Функциональность отображения и редактирования структуры сети AFDX продемонстрирована на рисунке 7.

САПР AFDX реализована на языках C++ (алгоритмическая подсистема, подсистема ввода-вывода) и Python (графический интерфейс) и функционирует под управлением ОС Linux. Структура САПР AFDX аналогична приведенной на рисунке 5 структуре САПР «Планировщик ИМА» с поправкой на отсутствие функциональности генерации отчетов и на вид экспортируемых данных (конфигурация сети вместо расписаний).

Для построения системы виртуальных каналов в сети AFDX (иными словами – для конфигурирования сети) САПР AFDX использует алгоритмы, описанные в [5]. Эти алгоритмы расширяют подходы, предложенные в [12], посредством учета ограничений на джиттер и длительность передачи сообщений через сеть. В случае, если после первоначального конфигурирования сети (включая определение маршрутов виртуальных каналов) эти ограничения не выполняются для некоторых виртуальных каналов, проводится ограниченный перебор с целью поиска альтернативных маршрутов для проблемных каналов, так, чтобы при использовании этих маршрутов ограничения соблюдались. Если ограниченный перебор не привел к полному соблюдению ограничений, запускается процедура агрегации каналов, имеющих одного и того же отправителя и хотя бы частично пересекающихся по маршрутам. Также для отдельно взя-

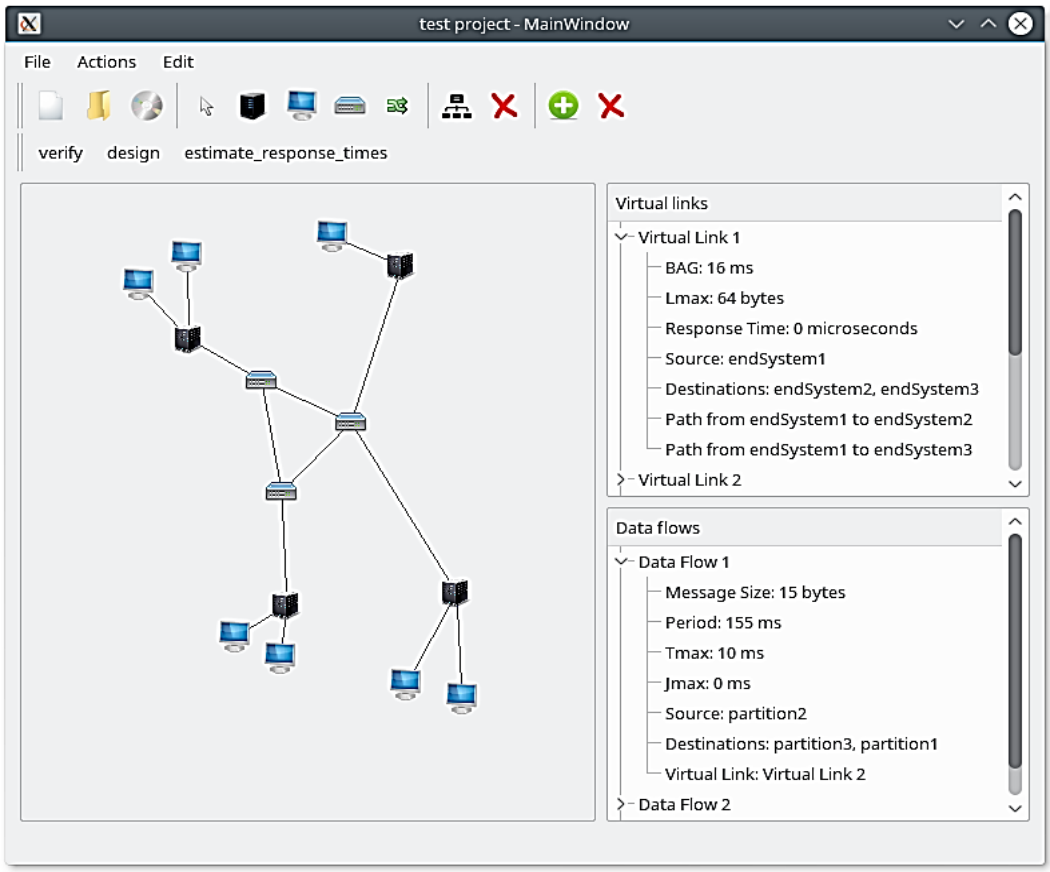

Рис. 7. Отображение и редактирование структуры сети AFDX

того канала поддерживается процедура перенастройки параметров (максимальный размер кадра, джиттер, длительность передачи через сеть), чтобы в случае превышения джиттером максимально допустимого значения сократить джиттер за счет некоторого увеличения длительности передачи сообщения через сеть или, наоборот, в случае недопустимо высокой длительности уменьшить ее за счет роста джиттера.

Алгоритмы, реализованные в САПР AFDX, были исследованы на различных сетевых топологиях (в т.ч. аналогичных используемым на реальных бортах, с порядка 10 коммутаторами и несколькими десятками оконечных систем) и на наборах сообщений с различными характеристиками (кратные и некратные периоды от 10 мс до нескольких секунд, размеры от 1 Кбайт до нескольких Мбайт). Алгоритмы показали высокую эффективность, в большинстве случаев построив корректные конфигурации для передачи всех сообщений. Наибольшее количество не включенных в конфигурацию сообщений (порядка 25 %) имели место в случае жестких ограничений на длительность передачи сообщений через сеть (до 1/10 периода сообщения). Наибольший вклад в эффективность алгоритма вносит процедура агрегации виртуальных каналов, при отключении которой доля сообщений, не вошедших в получаемую конфигурацию (то есть для которых не построены виртуальные каналы), резко возрастает – с 0 до десятков процентов для ряда наборов входных данных.

САПР МКИО

Для решения задачи построения расписаний информационного обмена по каналам с централизованным управлением, связывающим модульное «ядро» БВС РВ с периферийными подсистемами, коллективом ЛВК разработана инструментальная система САПР МКИО. Типовая последовательность действий при применении этой САПР следующая:

1) создание проекта: наполнение БД информацией о структуре бортовой сети (каналы передачи данных, их абоненты) и характеристиках рабочей нагрузки на каналы (наборы сообщений); в САПР МКИО поддерживается работа с каналами передачи данных МКИО, но возможно расширение функциональности САПР для поддержки других каналов с централизованным управлением, например FC-AE-1553 [35];

2) автоматическое построение расписания обмена данными, являющегося полным (включающим все работы) и корректным (удовлетворяющим всем ограничениям на корректность расписания);

3) при необходимости ручная корректировка расписания;

4) в случае, если нельзя построить полное и корректное расписание, автоматическая корректировка технологических ограничений таким образом, чтобы было возможным построение расписания с обновленными ограничениями;

5) формирование отчетов по исходным данным и результатам применения САПР для включения в документацию по БВС РВ;

6) экспорт построенного расписания вычислений во внешние файлы для включения в состав конфигурации ОС РВ.

САПР МКИО поддерживает визуальное отображение структуры построенного расписания, а также визуальное редактирование расписания. На рисунке 8 показан снимок экрана, демонстрирующий функциональность отображения и редактирования структуры расписания.

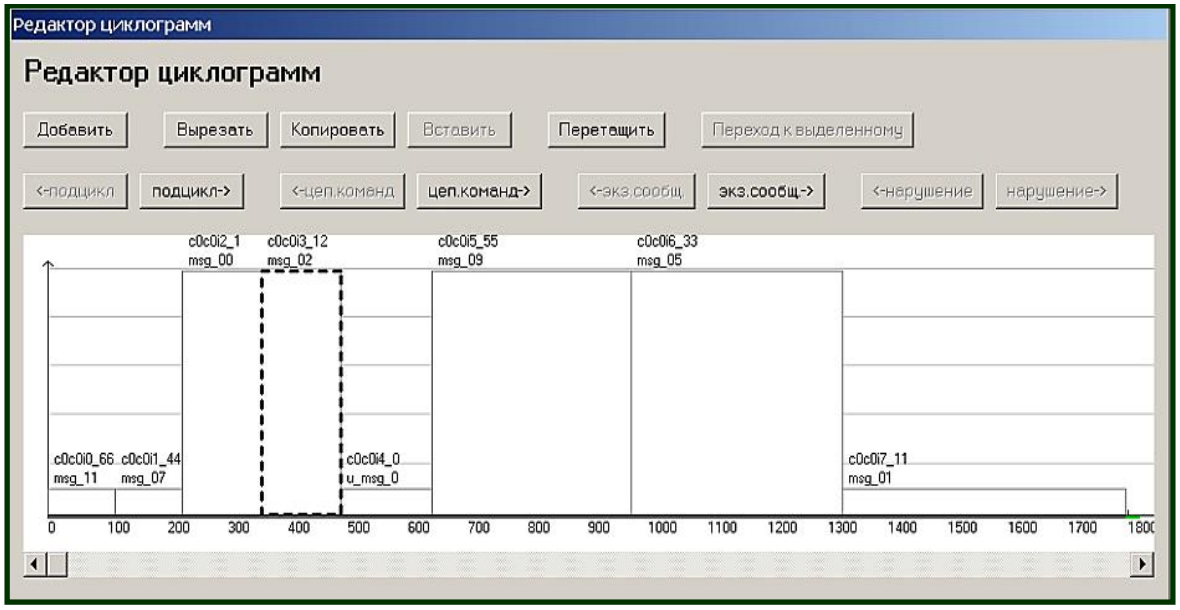

Рис. 8. Отображение и редактирование структуры расписания МКИО

САПР МКИО реализована на языке C++ и функционирует под управлением ОС Windows или Linux. Структура САПР МКИО аналогична приведенной на рисунке 5 структуре САПР «Планировщик ИМА» с поправкой на вид экспортируемых данных (расписание обмена данными вместо расписаний выполнения вычислений).

Для построения расписаний информационного обмена в САПР МКИО реализованы два алгоритма [36]: жадный, основанный на жадной схеме, и муравьиный, основанный на схеме муравьиных колоний [37].

Жадный алгоритм является развитием алгоритма, введенного в [28], но расширяет его за счет поддержки технологических ограничений на расписание обмена и различных критериев выбора очередной работы для включения в расписание. В ходе выполнения жадного алгоритма расписание достраивается итеративно, а работы размещаются на временной оси слева направо в порядке возрастания времени старта. На каждой итерации алгоритма определяется множество работ, которые могут стартовать в текущий рассматриваемый момент времени (точка планирования) без нарушения директивных сроков и технологических ограничений. Из этих работ выбирается одна в соответствии с заданным критерием. Выбранная работа включается в расписание с моментом старта, равным точке планирования. Сама точка планирования перемещается на момент завершения выбранной работы, то есть сдвигается вправо на длительность этой работы. Если для новой позиции точки планирования нет ни одной работы, которая может быть включена в расписание со стартом в этот момент времени (без нарушения директивных сроков и технологических ограничений), точка планирования сдвигается вправо до ближайшего момента времени, для которого существует хотя бы одна такая работа. По ходу выполнения алгоритма работы из первоначального входного перечня перемещаются в перечень либо включенных в расписание, либо не вошедших в расписание в связи с тем, что их директивные интервалы оказались заняты другими работами.

В качестве критериев выбора работ для включения в расписание используются (по выбору пользователя):

– наименьший директивный срок завершения $f(v)$(earliest deadline first, EDF [38]);

– наименьшее значение разности между директивным сроком завершения и сроком завершения при старте в текущей точке планирования (least slack first, LSF [39]);

– максимальная частота сообщения, которое соответствует работе (rate monotonic, RM [38]).

В основе муравьиного алгоритма лежит представление расписания в виде пути по полносвязному графу, вершины которого делятся на две категории:

– вершины, соответствующие работам;

– вершины, соответствующие подциклам (путь должен начинаться с вершины-подцикла).

Каждая из вершин входит в путь ровно один раз. Вершины-работы, следующие в пути за вершиной-подциклом, соответствуют работам (обменам по каналу) из состава цепочки обменов, выполняемой в этом подцикле. Порядок следования подциклов в расписании и порядок следования работ в подциклах определяются последовательностью соответствующих вершин в пути по графу. Моменты старта работ определяются по схеме, аналогичной описанному выше жадному алгоритму, с поправкой на то, что в качестве критерия для выбора очередной работы для включения в расписание используется последовательность вершин-работ в пути, а очередной подцикл открывается при прохождении вершины-подцикла. Кроме того, поскольку путь по графу должен включать все вершины, допускается условное включение в расписание работ с нарушениями директивных сроков или технологических ограничений. Критерием сравнения путей по графу является число таких работ (меньше–лучше).

Муравьиный алгоритм работает итеративно, на каждой итерации выполняя построение нескольких путей и сочетая при выборе очередной вершины пути:

– учет локального критерия выбора вершины;

– элемент случайности выбора вершины;

– феромонные метки (построенную на предыдущих итерациях числовую разметку дуг, входящих в наилучшие найденные пути; чем больше итераций прошло с момента постановки метки, тем большая доля метки «испаряется»).

Проведено экспериментальное исследование алгоритмов, реализованных в САПР МКИО. При исследовании использовались наборы данных по реальным БВС РВ авиационного и морского назначения, а также синтетические наборы данных, сформированных на их основе. Исследование продемонстрировало [36] существенную зависимость качества результатов, получаемых жадным алгоритмом, от критерия выбора работы для включения в расписание. На наборах данных по БВС РВ морского назначения, для которых специфична сложная структура директивных интервалов (директивные интервалы сообщений короче, чем периоды сообщений), при использовании критерия RM полные расписания строились только при низкой (до 35 %) загрузке канала, причем для загрузки 30–35 % полные расписания строились только для 25 % наборов сообщений. В зависимости от специфики исходных данных (кратность периодов, структура директивных интервалов) лучшие результаты показывал жадный алгоритм либо с критерием EDF, либо с критерием LSF. Но в случае наиболее сложной структуры директивных интервалов (в некоторых подциклах – по два директивных интервала для двух работ с паузой между этими интервалами) жадный алгоритм с любым из трех критериев показывал неудовлетворительный результат: не более чем для 15 % наборов входных данных строились полные расписания. Это связано с неспособностью указанных выше критериев выбора учесть, что работы с широкими директивными интервалами нужно откладывать на будущее, чтобы закрыть паузу между директивными интервалами других работ внутри подцикла и избежать нарушения требования непрерывности единственной цепочки обменов в подцикле.

Муравьиный алгоритм показал высокую адаптивность к специфике входных данных, построив для всех исследованных наборов сообщений полные расписания без нарушения заданных ограничений. Вместе с тем муравьиный алгоритм имеет ограничения по учету некоторых видов требований к расписанию: например, неясно, каким образом учесть требование, чтобы интервалы времени между последовательными обменами с передачей одного и того же сообщения не отклонялись от периода сообщения более, чем на заданную долю от этого периода.

Также необходимо отметить определенную сложность при объяснении логики работы муравьиного алгоритма пользователям САПР МКИО, в связи с чем для них предпочтительно использование жадного алгоритма с выбранным вручную критерием, наиболее подходящим для конкретного используемого набора сообщений.

Совместное использование разработанных инструментальных систем для поддержки проектирования БВС РВ

В рамках поддержки проектирования БВС РВ целесообразно совместно применять описанные в предыдущем разделе инструментальные системы. Определим последовательность применения этих систем для решения соответствующих задач проектирования БВС РВ.

Исходные данные для задачи построения расписания обмена данными с периферийными подсистемами по каналу с централизованным управлением не зависят от распределения вычислительной нагрузки по модулям и ядрам модульного ядра БВС РВ, расписаний выполнения вычислений на ядрах, а также от конфигурации сети, связывающей эти модули. Следовательно, эта задача может решаться без привязки к решению остальных рассмотренных ранее задач, а САПР МКИО применяться без непосредственного сопряжения с САПР «Планировщик ИМА» и САПР AFDX.

В то же время для расчета конфигурации коммутируемой сети необходима информация о распределении вычислительных задач по модулям. По этой информации определяются начальные и конечные абоненты, между которыми должны быть проложены виртуальные каналы; данные о выходных сообщениях задач-отправителей используются для определения интенсивности трафика по виртуальным каналам и, как следствие, требуемой пропускной способности этих каналов.

При определении границ окон для разделов, а также при проверке корректности расписания окон должны использоваться задержки передачи синхронных сообщений, определенные с учетом распределения задач по модулям и характеристик виртуальных каналов, проходящих через сеть. Передача сообщений между задачами, разделы которых расположены на одном модуле, осуществляется через оперативную память модуля с задержками, значительно меньшими, чем при передаче через сеть, и не зависящими от конфигурации сети.

С учетом сказанного последовательность решения рассмотренных ранее задач должна быть следующей:

1) построение привязки разделов к процессорным ядрам в составе модулей (решается средствами САПР «Планировщик ИМА»);

2) построение конфигурации сети передачи данных между модулями (решается средствами САПР AFDX или аналогичной САПР для другой используемой СПД на основе коммутаторов);

3) построение расписаний окон для процессорных ядер (решается средствами САПР «Планировщик ИМА»).

Последовательность применения указанных САПР должна соответствовать приведенной последовательности решения задач. Одним из решений является запуск САПР AFDX из САПР САПР «Планировщик ИМА» после решения задачи 1 с использованием результатов запуска при решении задачи 3.

Возможно также использование для решения задач 1, 2, 3 алгоритмов с обратной связью, которые в случае неуспешного решения одной из задач уточняют входные данные для предшествующей задачи. Например, неуспешное решение задачи 2 (построение конфигурации СПД) может быть обусловлено тем, что полученная при решении задачи 1 привязка разделов к ядрам, оптимальная с точки зрения минимизации суммарной загрузки сети, создает высокую нагрузку на канал от коммутатора к одному из модулей, в связи с чем длительность передачи сообщений от/к этому модулю не укладывается в требуемые ограничения. В таком случае разделы с этого модуля необходимо распределить на разные модули и для задачи 1 добавляется ограничение – недопустимо привязывать данную совокупность разделов к одному модулю. Задача 1 должна быть решена повторно с учетом этого ограничения.

В случае неуспешного решения задачи 3 (построение расписаний окон) возможен возврат к решению задачи 2 в случае, если выполнение цепочки задач с синхронными зависимостями не может быть осуществлено в рамках директивных сроков (задаваемых периодами задач) в связи с высокими длительностями передачи сообщений через сеть. В этом случае при повторном решении задачи 2 необходимо сократить суммарную задержку при передаче сообщений через соответствующие виртуальные каналы, например, за счет увеличения зарезервированной под эти каналы пропускной способности СПД.

Также при неуспешном решении задачи 3 возможен возврат к решению задачи 1 с целью:

- сокращения длительности передачи сообщений за счет группировки некоторых разделов на один модуль, что приведет к передаче сообщений между задачами этих разделов через оперативную память модуля и к обнулению задержек, связанных с передачей их через сеть;
- снижения нагрузки на ядро, расписание окон для которого не удалось построить в связи со слишком высокой загрузкой ядра – в этом случае при повторном решении задачи 1 необходимо уменьшить верхнее ограничение на загрузку этого ядра.

Разработка алгоритмов с обратной связью на основе ранее созданных алгоритмов планирования вычислений в модульных БВС РВ является перспективным направлением исследований; первые результаты работы приведены в [40].

На рисунке 9 показана диаграмма переходов между решением задач 1–3 в рамках схемы с обратной связью. На линиях обратной связи показаны цели, для достижения которых осуществляется переход на более ранние этапы планирования. Необходимо отметить возможность не последовательного, а одновременного решения задач 1–3 как минимум в части пошагового определения привязки разделов к модулям *вместе* с пошаговым построением конфигурации СПД. Исследование этой схемы является предметом работ на более отдаленную перспективу.

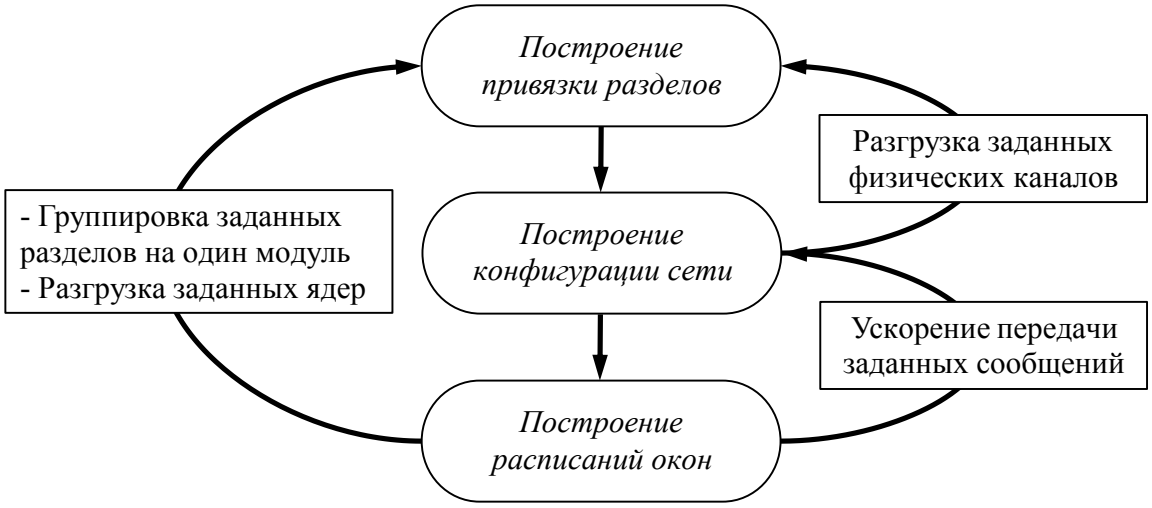

Рис. 9. Планирование вычислений с обратной связью

Заключение

В работе рассмотрен ряд задач планирования вычислений, возникающих при проектировании БВС РВ, сочетающих модульную и федеративную архитектуру. Приведены подходы к планированию вычислений в модульном ядре БВС РВ и информационного обмена между ядром и периферийными подсистемами, имеющими федеративную архитектуру. Описано семейство разработанных в ЛВК инструментальных систем поддержки проектирования БВС РВ, реализующих эти подходы. Предложена схема совместного применения этих инструментальных систем для согласованного решения соответствующих задач.

Следует отметить дальнейшие направления исследований:

– разработка подходов к разделению функциональности БВС РВ между модульным ядром и федеративными периферийными подсистемами, к определению структуры вычислительной нагрузки на ядро БВС РВ и к структурному синтезу ядра;

– повышение масштабируемости точных алгоритмов планирования вычислений, в первую очередь – алгоритма распределения вычислительной нагрузки по модулям и ядрам, основанного на методе ветвей и границ;

– реализация предложенной схемы планирования вычислений в модульном ядре БВС РВ с обратной связью между этапами планирования, в т.ч. необходимая доработка алгоритмов планирования и их сопряжение в комплекс;

– сопряжение между собой разработанных инструментальных систем поддержки планирования вычислений (САПР «Планировщик ИМА», САПР AFDX, САПР МКИО).

Итоговой целью работ является создание инструментальной системы (САПР), поддерживающей сквозное решение задач планирования вычислений, возникающих при создании БВС РВ, сочетающих модульную и федеративную архитектуры.

Работа выполнена при финансовой поддержке РФФИ, грант №17-07-01566.

Литература

1. Парамонов П.П., Жаринов И.О. Интегрированные бортовые вычислительные системы: обзор современного состояния и анализ перспектив развития в авиационном приборостроении // Науч.-технич. вестн. информ. технологий, механики и оптики. 2013. № 2. С. 1–17.

2. Многоядерный Т-50: на новом российском истребителе ИМА БК заменила «Багет» // Горизонты 2014. № 1. С. 10–13.

3. Костенко В.А. Возможные причины снижения эффективности использования аппаратных средств КБО с архитектурой интегрированной модульной авионики // Авиационные системы в XXI веке: тр. Всерос. науч.-технич. конф. М.: Изд-во ГосНИИАС, 2016. С. 221–225.

4. Костенко В.А., Смелянский Р.Л. Проблемы построения бортовых комплексов с архитектурой интегрированной модульной авионики // Радиопромышленность. 2016. № 3. С. 63–70.

5. Вдовин П.М., Костенко В.А. Организация передачи сообщений в сетях AFDX // Программирование. 2017. № 1. С. 5–20.

6. Смелянский Р., Костенко В., Балашов В., Балаханов В. Инструментальная система построения расписания обмена данными по каналу с централизованным управлением // Современные технологии автоматизации. 2011. № 3. С. 78–84.

7. Raidl G.R. The multiple container packing problem: a genetic algorithm approach with weighted codings. ACM SIGAPP Applied Computing Review. 1999, vol. 7, no. 2, pp. 22–31.

8. Crainic T.G., Perboli G. and Tadei R. Recent advances in multi-dimensional packing problems. New Technologies – Trends, Innovations and Researc, 2012, pp. 91–110.

9. Martello S., Toth P. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, 1990, 296 p.

10. Fukunaga A.S., Korf R.E. Bin completion algorithms for multicontainer packing, knapsack, and covering problems. Jour. of Artificial Intelligence Research, 2007, 28, pp. 393–429.

11. Frances F., Fraboul C., Grieu J. Using network calculus to optimize the AFDX network. Proc. ERTS 2006: 3rd Europ. Congr. on Embedded Real-Time Soft. 2006.

12. Al Sheikh A. et al. Optimal design of virtual links in AFDX networks. Real-Time Systems, 2013, vol. 49, no. 3, pp. 308–336.

13. GoltzH.-J., PiethN. A tool for generating partition schedules of multiprocessor systems. Proc. 23rd Workshop on (Constraint) Logic Programming, 2009, pp. 167–176.

14. Al Sheikh A., Brun O., and Hladik P.E. Partition scheduling on an IMA platform with strict periodicity and communication delays. Proc. 18th Intern. Conf. on Real-Time and Network Systems, 2010, pp. 179–188.

15. Easwaran A., Lee I., Sokolsky O., and Vestal S. A compositional framework for avionics (ARINC 653) systems. Proc. IEEE Real-Time Computing Systems and Applications, 2009, pp. 371–380.

16. Craveiro J.P. Real-time scheduling in multicore time- and space-partitioned architectures. Ph.D. Thes. Univ. of Lisbon, 2013.

17. Третьяков А.В. Автоматизация построения расписаний для периодических систем реального времени // Тр. ИСП РАН. 2012. Т. 22. С. 375–400.

18. Буздалов Д., Зеленов С., Корныхин Е., Петренко А., Страх А., Угненко А., Хорошилов А. Инструментальные средства проектирования систем интегрированной модульной авионики // Тр. ИСП РАН. 2014. Т. 26. Вып. 1. С. 201–230.

19. The Real-Time Analysis Experts. 2017. URL: http://www.tripac.com/ (дата обращения: 12.09.2017).

20. The Cheddar project: a free real time scheduling analyzer. 2017. URL: http://beru.univ-brest.fr/~singhoff/cheddar (дата обращения: 12.09.2017).

21. MAST: Modeling and Analysis Suite for Real-Time Applications. 2014. URL: http://mast.unican.es/ (дата обращения: 12.09.2017).

22. Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы. М.: Наука, 1984. 222 с.

23. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Минимизация суммарного запаздывания для одного прибора. М.: Изд-во ВЦ РАН, 2006. 150 с.

24. Federgruen A., Groenevelt H. Preemptive scheduling of uniform machines by ordinary network flow technique. Jour. of Management Science, 1986, vol. 32, no. 3, pp. 341–349.

25. Coffman E.G. (Ed.). Computer and job-shop scheduling theory. John Wiley & Sons, 1976.

26. Гуз Д.С., Фуругян М.Г. Составление расписаний выполнения заданий и загрузки памяти для однопроцессорных систем жесткого реального времени // Моделирование процессов управления. М: Изд-во МФТИ, 2004. С. 162–169.

27. Гуз Д.С., Фуругян М.Г. Планирование вычислений в многопроцессорных АСУ реального времени с ограничениями на память процессоров // Автоматика и телемеханика. 2005. № 2. С. 138–147.

28. Костенко В.А., Гурьянов Е.С. Алгоритм построения расписаний обменов по шине с централизованным управлением и исследование его эффективности // Программирование. 2005. № 6. С. 67–76.

29. Sivanandam S.N., Deepa S.N. Introduction to Genetic Algorithms. Springer Berlin Heidelberg, 2008, 442 p.

30. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М.: Физматлит, 2006. 320 с.

31. Whitley D., Rana S., Heckendorn R.B. The Island model genetic algorithm: on separability, population size and convergence. Jour. of Computing and Information Technology, 1999, no. 7, pp. 33–47.

32. Balashov V.V., Balakhanov V.A., KostenkoV.A. Scheduling of computational tasks in switched network-based IMA systems. Proc. Intern. Conf. on Engineering and Applied Sciences Optimization, NTUA, Athens, Greece, 2014, pp. 1001–1014.

33. Глонина А.Б. Обобщенная модель функционирования модульных вычислительных систем реального времени для проверки допустимости конфигураций таких систем. Суперкомпьютерные дни в России: тр. Междунар. конф. М.: Изд-во МГУ, 2017. С. 800–814.

34. Вдовин П.М., Костенко В.А. Инструментальная система конфигурирования виртуальных каналов в сетях AFDX // Авиационные системы в XXI веке: сб. тез. докл. Всерос. науч.-технич. конф. 2016. С. 221–225.

35. Information technology. Fibre Channel – Part 312: Avionics environment upper layer protocol MIL-STD-1553B Notice 2 (FC-AE-1553): Technical report TR 14165312:2009. ISO, 2009.

36. Balashov V., Balakhanov V., Kostenko V. et al. A technology for scheduling of data exchange over bus with centralized control in onboard avionics systems. Proc. Institution of Mechanical Engineers, Part G, Jour. of Aerospace Engineering, 2010, vol. 224, no. 9, pp. 993–1004.

37. Штовба С.Д. Муравьиные алгоритмы: теория и применение // Программирование. 2005. № 4. С. 3–18.

38. Liu C.L., Layland J.W. Scheduling Algorithms for Multiprogramming in a hard-real-time environment, Jour. of the ACM, 1973, vol. 20, no. 1, pp. 46–61.

39. Mok A.K. Fundamental design problems of distributed systems for the hard-real-time environment. Ph.D. Thesis. MIT, Cambridge, MA, USA, 1983.

40. Новоселов А.Д., Балаханов В.А. Построение статико-динамического расписания алгоритмом с обратной связью // Программные системы и инструменты. 2016. Тематич. сб. № 16. С. 108–118.