УДК 629.78.018

**И.Б. ТУРКИН, Е.В. СОКОЛОВА, Ю.А. ШЕПЕТОВ, Т.С. НИКИТИНА**

***Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина***

# ПРАКТИЧЕСКИЕ АСПЕКТЫ РЕАЛИЗАЦИИ ДИНАМИЧЕСКОГО ПЛАНИРОВАНИЯ ЗАПРОСОВ В КЛИЕНТ-СЕРВЕРНЫХ СИСТЕМАХ НА ОСНОВЕ OPC

Показано, что ключевые для систем реального времени свойства реактивности и предсказуемости в клиент-серверных системах, реализующих технологию OPC могут быть достигнуты при разработке программного обеспечения, реализующего алгоритмический и ресурсный способ адаптации. Сформулированные ограничения на актуальность данных и временную целостность информации позволяют рассматривать процесс планирования запросов в клиент-серверных системах реального времени как задачу динамического программирования, решение которой дает политику управления запросами. С учетом разработанных архитектурных решений программного обеспечения выполнен качественный анализ проблемных вопросов.

**системы реального времени, технология клиент-сервер, АСУТП, SCADA система, OPC**

## Введение

В настоящее время большинство систем реального времени диспетчерского управления и сбора данных SCADA-систем (Supervisory Control And Data Acquisition System) проектируются по общей трех уровневой схеме: контроллеры, сети передачи данных, серверы приложений. Стандартизованным связующим звеном между аппаратурой нижнего уровня и интерфейсами выступает технология OPC (OLE for Process Control), которая обеспечивает универсальный механизм обмена данными. Спецификация OPC [1] определяет три модели доступа к данным нижнего уровня и в качестве процедуры обеспечения целостного доступа к производственным данным имеет свои преимущества и недостатки. Математическая постановка задачи динамического программирования для адаптивного управления запросами в клиент-серверных системах реального времени была описана ранее [2]. Рассмотрим практические аспекты реализации этой задачи.

## Классификация задач

Классифицируя задачи по критичности к выполнению требований реального времени [3] выделим три группы:

– жесткого реального времени, когда использование устаревших данных эквивалентно катастрофе.

Просроченные данные из этой группы использовать нельзя, необходимо предпринять действия по парированию аварийной ситуации. Эти действия могут основываться на активации соответствующих запросов для обновления исходных объектов-данных, либо предполагать старт служебных задач, обеспечивающих приемлемый выход из сложившейся ситуации;

– мягкого реального времени, когда обработка устаревших данных может быть доведена до завершения;

– крепкого реального времени, когда обработка устаревших данных нецелесообразна, поскольку вызывает лишь расход системных ресурсов.

– С другой стороны по критерию активности задачи могут быть классифицированы также на три группы:

– активные плановые задачи, для которых определена периодичность их исполнения и/или из-

вестны объекты данные, изменение которых должно вызывать эти задачи на исполнение;

– пользовательские задачи, которые исполняются один раз, будучи инициализированными внешними по отношению к рассматриваемой системе актерами;

– неактивные, но готовые к активации задачи. В категорию активных плановых они могут быть переведены пользовательскими, либо плановыми задачами.

Активные в системе задачи с точки зрения их важности можно поделить на два класса – пользовательские, которые являются наиболее важными, и прочие. Для учета важности задачи используется механизм приоритетов. Задачи с более высоким приоритетом имеют преимущество перед более низкоприоритетными транзакциями.

## Архитектура системы

Динамическое планирование запросов невозможно в существующей децентрализованной архитектуре «много OPC-клиентов – много OPC-серверов», поскольку предполагает формирование и исполнение единой стратегии запросов. Следовательно, необходим переход к архитектуре «много OPC-клиентов – один субсервер – много OPC-серверов» (рис. 1).

В рамках предлагаемой архитектуры реализуются паттерны, широко применяемые в СРВ и системах критического назначения [4]:

– микроядро, обеспечивающее взаимодействие сервисных и прикладных задач, путем предоставления минимально необходимого множества служб и интерфейсов;

– канальная обработка информации достигается путем последовательного исполнения задач, предоставляющих свои результаты через менеджер данных;

– шина данных, предоставляющая механизм для единообразной передачи информации и т.д.

Для обеспечения надежности и функциональной безопасности в состав субсервера могут быть реализованы паттерны монитора и сторожевого таймера.

Взаимодействие «конфигурационная база данных – планировщик запросов» обеспечивает настройку системы за счет доступа к представленным в табличной форме индивидуальным параметрам опроса каждого сигнала: тип данных, адрес в контроллере, периоды периодических опросов значений и изменений и т.п. Стандартный интерфейс OPC Custom обеспечивает функционирование компонентов и объектов OPC. Интерфейсы «субсервер – задачи» расширяют функциональность стандартного OPC интерфейса за счет возможности регистрации периодических задач, подлежащих исполнению, списков сенсорных и порожденных данных, заданных временных ограничений, состава задач, активация которых возможна при наступлении внешних событий.

Работа алгоритма планировщика запросов основана на динамической идентификации основных параметров, характеризующих затраты времени на взаимодействие с доступными OPC-серверами, в том числе на операции:

– создания/удаления связей с данными (то есть создание/удаление тегов);

– изменения состояния групп;

– изменения состояния связей с данными (тегами);

– синхронное/асинхронное чтение (запись) данных;

– подписки, когда субсервер передаёт OPC-серверу список интересующих его переменных, а сервер затем регулярно присылает клиенту информацию об изменившихся переменных из этого списка.

Результаты динамической идентификации используются для организации адаптивного управления запросами. Кэширование данных, реализованное в OPC-серверах, может не применяться, так как реализуется менеджером данных субсервера. Рассмотрим некоторые особенности реализации алгоритма планировщика.

## Выбор периода подписки на группу переменных

Для подписки клиента на группу переменных, OPC-спецификация предусматривает возможность указание параметров подписки, в том числе: перечня переменных (тегов) в группе, максимально допустимого периода оповещения об изменениях переменных и процентной величины порога изменения переменных. Зарегистрировав подписанную группу, OPC-сервер обязан отсылать клиенту информацию о текущих значениях переменных группы, если превышено время, указанное в качестве периода оповещения, либо любая из переменных изменилась больше, чем было задано порогом изменения. Таким образом, обновление данных по подписке выполняется периодически и поддерживает актуальность данных.

Несмотря на всю привлекательность указанного режима обмена информацией, он не гарантирует высокие и надежные показатели по следующим причинам:

– выбор большой величины периода оповещения опасен тем, что в течение какого-то времени сенсорные данные, могут оказаться не абсолютно корректными и, следовательно, не могут быть использованы ни одной задачей, малый период оповещения может привести к перегрузке системы;

– OPC-сервера разных производителей реализуют механизм подписки по-своему. Так, синхронизированные функции обновления тега со стороны уровня, обеспечивающего коммуникацию с внешними устройствами, в универсальных OPC-серверах фирмы Fastwel появились только, начиная с версии 3.0 [5]. В более ранних версиях обновление информации, а, следовательно и, оповещение OPC-клиента интерфейсной частью сервера производилось только по истечении временного интервала;

– в случае синхронного обновления тега со стороны уровня, обеспечивающего коммуникацию с внешними устройствами, возникает иная проблема.

Определенные события во внешней системе могут привести к одновременному изменению значений нескольких тегов. При этом оповещение OPC-клиента интерфейсной частью сервера произойдет многократно, по мере того, как после поступления информации от внешних устройств, будет поступать во внутренние таблицы данных сервера. Таким образом, мы приходим к требованию отслеживать режим поступления обновлений. Для решения этой задачи можно предложить блокировать OPC-сервером передачу сообщений об изменении значений в объектах-элементах на период записи всего множества поступивших обновленных значений.

## Поддержка менеджером данных многоверсионности переменных

Перечисленные выше ограничения на актуальность и временную целостность данных приводят к достаточно интересному эффекту: и предыдущее, и текущее значение объекта данных может удовлетворять абсолютному ограничению на актуальность, зато относительное ограничение, характеризующие допустимые временные рассогласования между этим объектом и какими-то иными, может выполняться только для предыдущего значения объекта данных. Следовательно, в таком случае корректным является использование только предыдущего значения обновляемого элемента данных.

Поэтому менеджер данных должен хранить короткую предысторию объектов данных, сохраняя все их версии, удовлетворяющие абсолютным ограничениям. Из-за возможности существования нескольких версий одного и того же элемента данных

менеджер должен предоставить задаче набор данных, укомплектовав его такими их наиболее свежими версиями, которые не нарушают относительную целостность.

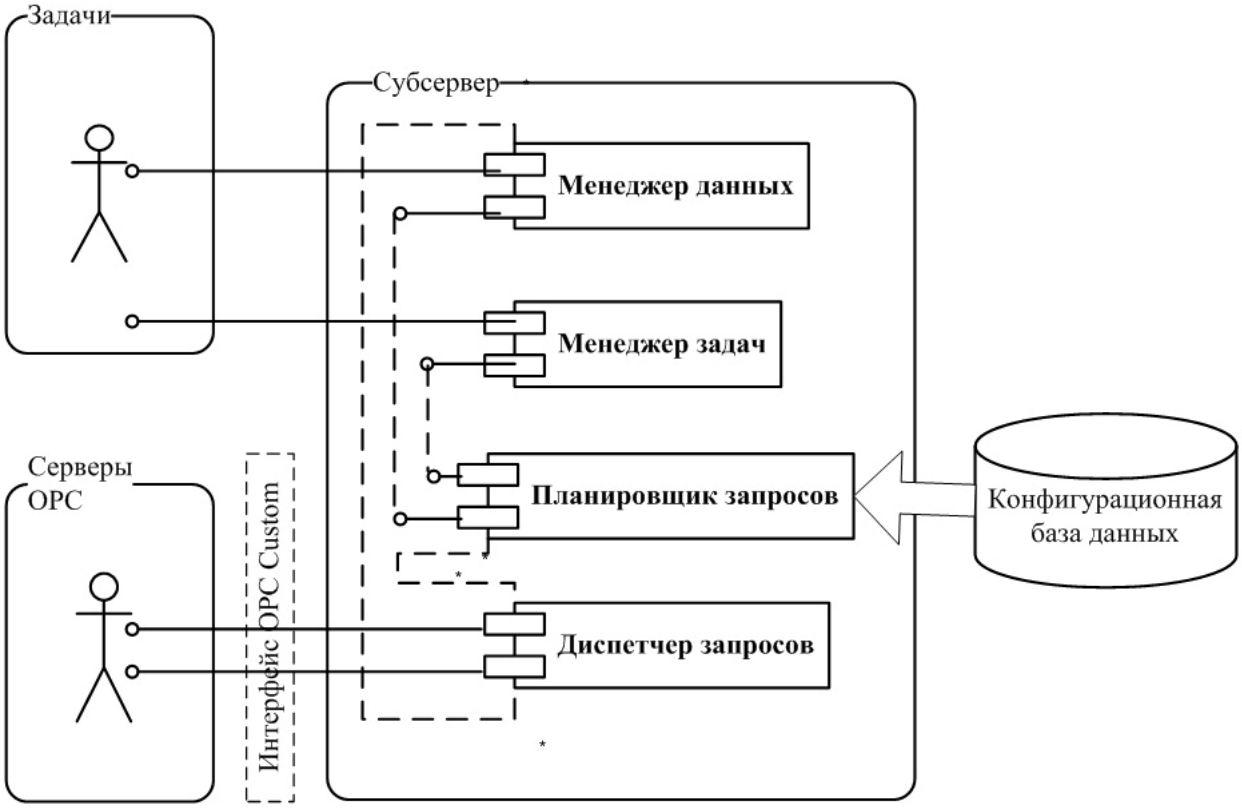


Рис. 1. Архитектура системы

## Выводы

В результате исследований теоретически обоснована архитектура «много OPC-клиентов – один субсервер – много OPC-серверов» для СРВ и систем критического назначения, которая позволит рассматривать процесс планирования запросов в клиент-серверных СРВ как задачу динамического программирования, решение которой даст политику управления запросами.

## Литература

1. OPC Data Access Custom Interface Specification 2.0. 1998.

2. Туркін І.Б., Соколова Є.В., Никитина Т.С. Динамічне планування запитів у клієнт-серверних системах реального часу, які реалізують технологію «OLE for Process Control». // Системи озброєння та військова техніка. – 2007. – № 4. – С. 45-49.

3. Laplante, Phillip A. Real-time Systems Design and Analysis: An Engineer's Handbook, 3rd ed. – 505 p.

4. Bruce Powel Douglass. Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Addison Wesley. – 2002. – 528 p.

5. Fastwel Universal OPC Сервер. Руководство пользователя.

*Поступила в редакцию 3.02.2008*

**Рецензент:** д-р техн. наук, проф. И.В. Чумаченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.