УДК 629.78.018

Ю.А. КУЗНЕЦОВА, Е.В. СОКОЛОВА

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

АДАПТИВНОЕ УПРАВЛЕНИЕ СЕРВЕРОМ ДАННЫХ В SCADA-СИСТЕМАХ

Показано, что требование постоянной поддержки актуальности и целостности данных, т.е. обновление информации до истечения директивных сроков, приводит к решению оптимизационной задачи управления OPC-серверами в клиент-серверных системах, реализующих технологию OPC (OLE for Process Control). Таким образом достигаются наиболее важные критерии в системе реального времени – реактивность и предсказуемость. Сформулированы ограничения на абсолютную и относительную корректность данных, за счет чего адаптивное управление запросами в клиент-серверных системах реального времени может быть достигнуто в результате решения задачи динамического программирования. Стратегия планирования запросов представляет собой NP-сложную задачу, решить которую можно, применив метод ветвей и границ.

Ключевые слова: *системы реального времени, АСУ ТП, SCADA, OLE for Process Control, адаптивное управление, метод ветвей и границ.*

Введение

С развитием промышленности резко возросла потребность в высокоэффективных и высоконадежных автоматизированных системах управления технологическими процессами. Данная потребность обусловлена следующими факторами:

– возросшие требования к повышению качества технологического процесса;

– рост дефицита природных ресурсов;

– появление мощных, компактных, недорогих измерительных и управляющих устройств;

– повышение степени автоматизации производства и перераспределение функций между человеком и аппаратурой.

Современные автоматизированные системы управления технологическими процессами (АСУ ТП) представляют собой аппаратно-программные комплексы, которые выполняют следующие основные функции:

– сбор информации от объекта управления;

– передача, преобразование и обработка информации;

– формирование управляющих команд и выполнение их на управляемом объекте.

Нижний уровень АСУ ТП занимается непосредственно управлением технологическими процессами и оборудованием, а верхний уровень – представляет собой системы диспетчерского управления.

SCADA (Supervisory Control And Data Acquisition – оперативное диспетчерское управление и сбор данных) отвечает за процесс сбора информации в режиме реального времени с удаленных точек для обработки, анализа и возможного управления удаленными объектами [1].

При применении аппаратуры нижнего уровня различных производителей, а также шинных систем промышленного назначения с их разнообразными протоколами и интерфейсами особенно важно наличие стандартизованного связующего звена между SCADA-уровнем и уровнем контролеров. Таким связующим звеном является технология OPC (OLE for Process Control – OLE для управления процессами, Object Linking and Embedding – связывание и встраивание объектов).

АСУ ТП содержит огромное количество управляемых и регулируемых данных и все больше нуждается в решении задач, удовлетворяющих требованием реального времени. Система должна оперативно реагировать на внешнее событие, а ее данные – быть достоверными и непротиворечивыми, а также не вступающими в конфликт друг с другом. Для корректного решения задач такого рода необходимо применять технологию адаптивного управления.

1. Обзор литературы

1.1. Основные причины создания OPC

Клиентские программы взаимодействуют с различными устройствами ввода-вывода через соот-

ветствующие драйверы независимых разработчиков. Но при этом возникают следующие проблемы [2]:

– каждая программа диспетчеризации должна иметь драйвер для конкретного устройства АСУ;

– возникают конфликты между драйверами различных разработчиков, что приводит к тому, что какие-то режимы или параметры работы оборудования не поддерживаются всеми разработчиками программного обеспечения;

– модификации оборудования могут привести к потере функциональности драйвера;

– конфликты при обращении к устройству – различные программы диспетчеризации не могут получить доступ к одному устройству одновременно из-за использования различных драйверов.

1.2. Преимущества технологии OPC

OPC (OLE for Process Control) – промышленный стандарт, созданный консорциумом всемирно известных производителей оборудования и программного обеспечения при участии Microsoft, описывающий интерфейс обмена данными между устройствами управления технологическими процессами [3]. Главной целью было предоставить разработчикам систем диспетчеризации некоторую независимость от конкретного типа контроллеров.

OPC был разработан для обеспечения доступа клиентской программы к нижнему уровню технологического процесса в наиболее удобной форме. Широкое распространение технологии OPC в промышленности имеет следующие преимущества [4]:

– независимость в применении систем диспетчеризации от используемого в конкретном проекте оборудования;

– разработчики программного обеспечения не должны постоянно модифицировать свои продукты из-за модификации оборудования или выпуска новых изделий;

– заказчик получает свободу выбора между поставщиками оборудования, а также имеет возможность интегрировать это оборудование в информационную систему предприятия, которая может охватывать всю систему производства, управления и логистики.

1.3. Недостатки технологии OPC

Использование OPC-серверов, разработанных организацией-поставщиком оборудования, снижает стоимость и повышает качество разработки, но при этом имеет следующие недостатки:

– принцип сокрытия информации – инкапсуляция алгоритмов обработки в OPC-серверах приводит к тому, что многие их принципиально важные свойства и характеристики могут быть установлены только экспериментальным путем из-за неизвестных архитектурных решений и особенностей реализации OPC-серверов [5];

– структура и основные параметры программного обеспечения (ПО) реального времени (РВ), например, конфигурация программных клиентов, пропускная способность, протоколы и т.д., выбираются на основе предварительных оценок необходимой производительности. Возникает риск неадекватной разработки ПО РВ – либо выявляется нехватка производительности и пропускной способности возникающих потоков данных, либо выясняется, что в систему «заложен» излишний запас вычислительной мощности, поэтому разработчики систем идут по пути заведомого увеличения ресурсов системы, что не всегда приводит к желаемым результатам [6];

– обработка информации в СРВ привязана к событиям в контролируемой системе, а интенсивность этих событий в некоторых ситуациях может изменяться в десятки-сотни раз. Пропускная способность каналов, требуемая производительность процессора должны гарантировать корректность работы и в этом случае [7].

2. Постановка задачи

Создание алгоритмов, самоадаптирующихся к условиям работы, состоянию системы и внешней среды, является одним из основных технических решений, способных парировать перечисленные выше недостатки.

В общем случае ПО может реализовать один из трех способов адаптации: параметрический, алгоритмический и ресурсный.

Создание адаптирующихся систем широко распространяется в последнее время и является одним из основных направлений повышения качества программного обеспечения.

Целью работы является построение математических моделей для адаптивного управления сервером данных в системах супервизорного диспетчерского управления и сбора данных, а также разработка прототипа программного обеспечения, что, в свою очередь, позволит обеспечить высокое качество процессов управления объектами SCADA-системы за счет управления режимами ввода информации.

3. Адаптивный метод управления вводом информации

Адаптивный метод предполагается использовать в SCADA-системах на основе технологии OPC,

которая является связующим звеном между системами автоматизации и программами на ПК. Механизмы обмена информацией между OPC-клиентом и OPC-сервером должны быть эффективными по критериям потребляемых ресурсов и обеспечивать актуальность, целостность, полноту данных, характеризующих состояние объекта управления, так как программное обеспечение систем автоматизированного управления относится к классу систем реального времени, корректность работы алгоритмов которых зависит от соответствия временным ограничениям.

3.1. Формальная постановка задачи адаптивного управления

Ограничения. В системах реального времени очень важна работа с устаревающими данными, т.е. данными, которые с течением времени теряют свою значимость. Поэтому в расчетах необходимо обеспечить актуальность данных.

Формально объект данных можно описать при помощи кортежа:

$$\mathrm{Data_i} = \left\langle \mathrm{Value_i}, \tau_{\mathrm{query_i}}, \tau_{\mathrm{answer_i}}, \Delta\tau_i \right\rangle, \qquad (1)$$

где $\mathrm{Value_i}$ – значение i-го объекта данных в некоторый момент времени, находящийся в интервале $\tau_{\mathrm{query_i}} \div \tau_{\mathrm{answer_i}}$ от момента формирования запроса до момента получения ответа;

$\Delta\tau_i$ – длина интервала, в течение которого значение i-того объекта данных допускается считать корректным.

В любой момент времени функционирования ПО для интервала планирования

$$t \in \left]t_{\mathrm{current}},\ \ t_{\mathrm{current}} + \Delta t\right]$$

на все множество данных Data(t), изменяющееся во времени, должно выполняться множество ограничений на **абсолютную корректность данных**:

$$\forall \mathrm{Data_i} \in \mathrm{Data(t)}: t\text{-}\tau_{\mathrm{query_i}} < \Delta\tau_i . \qquad (2)$$

Корректность входных данных задачи-преобразователя будет обеспечена, если выполнено условия **относительной корректности** этих данных:

$$\begin{aligned} &\forall j,k, \mathrm{Data_j} \in \mathrm{Data}_{\mathrm{sensor}_{\mathrm{Task}}\,i} \in \mathrm{Data}_{\mathrm{in}_{\mathrm{Task}i}}, \\ &\mathrm{Data_k} \in \mathrm{Data}_{\mathrm{sensor}_{\mathrm{Task}}\,i}: \\ &\max\left(\tau_{\mathrm{answer}_{\mathrm{Data_j}}}, \tau_{\mathrm{answer}_{\mathrm{Data_k}}}\right) - \\ &-\min\left(\tau_{\mathrm{query}_{\mathrm{Data_j}}}, \tau_{\mathrm{query}_{\mathrm{Data_k}}}\right) \le \Delta\tau_{\mathrm{Task}_{i_{j,k}}} \end{aligned} \qquad (3)$$

Формально j-й запрос на ввод информации через k-й OPC-сервер с i-того измерительного преобразователя можно представить в виде:

$$\mathrm{Query}_{k,i,j} = \left\langle Z, \mathrm{Data}_{k,i,j} \right\rangle, \qquad (4)$$

а стратегию планирования запросов:

$$\mathrm{Queue}_{k,i,j} = \left\langle \mathrm{Query}_{k,i,0}, \ldots, \mathrm{Query}_{k,i,n_{k,i}} \right\rangle \qquad (5)$$

Тогда адаптивное управление запросами может быть достигнуто в результате решения задачи динамического программирования:

$$\pi^* = \arg\max_{\pi}\left[\min_{\tau=0..T}\left(n\left(\mathrm{Task}, \mathrm{Queue}\right)\right)\right], \qquad (6)$$

где T – интервал планирования запросов.

3.2. Анализ целевой функции

Требование постоянной поддержки актуальности и целостности данных, т.е. обновление информации до истечения директивных сроков приводит к решению оптимизационной задачи управления OPC-серверами. Рис. 1 показывает взаимосвязь понятий «актуальность данных» и «готовность к исполнению задач».

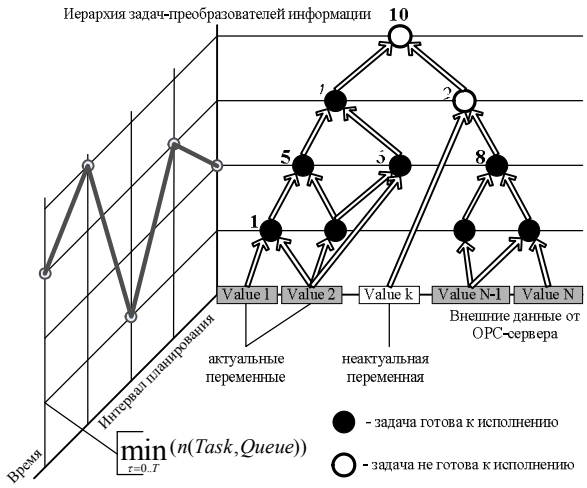

Рис. 1. Схема вычисления целевой функции

Как следует из рис. 1, задачи с номерами 1-8 в момент времени $\tau_i = 0$ могут быть непосредственно запущены на исполнение, так как все данные, необходимые для их работы, являются актуальными, а задачи 9, 10 – не готовы. В последующие моменты времени ситуация может измениться, что приведет к соответствующему изменению целевой функции $n\left(\mathrm{Task}, \mathrm{Queue}\right)$.

Рассмотрим более детально эту функцию. Она характеризует глубину вложенности множества готовых к исполнению элементарных задач-преобразователей информации, для которых необходимые данные являются актуальными в момент времени τ_i. Соответственно, ее значение характеризует эффективность выполнения запросов, то есть,

чем меньше значение функции, тем больше задач, неготовых к исполнению.

Время выполнения запроса зависит от количества запрашиваемых данных $Data_{sensor}$ и способа доступа Z к этим данным. Ввод информации из OPC-сервера в OPC-клиент может инициироваться сервером в режиме подписки или клиентом – в режимах синхронного либо асинхронного ввода. Обращение к данным может происходить как напрямую к устройству (device), так и через кэш (cache) OPC-сервера. Проведенные исследования показывают, что время обработки запроса может достигать 0,1 (с) при непосредственном обращении к устройству, в то время как скорость получения данных через кэш – 10^5 (тег/с).

Предположим, что затраты времени на получение информации либо через кэш, либо от устройства можно рассчитать по линейным моделям (7) и (8):

$$T_{cache} = \tau_{query} + \alpha_t \times N, \quad (7)$$

где τ_{query} – продолжительность операций инициирования обмена;

N – количество тегов в группе;

α_t – коэффициент пропорциональности для операции чтения из кэша.

$$T_{device} = \tau_{query} + \beta_t \times N, \quad (8)$$

где β_t – коэффициент пропорциональности для операции чтения с внешнего прибора.

При этом затраты времени на получение данных с внешнего устройства (9) и сумма затрат процессорного времени (10) не должны превышать интервал планирования T:

$$\sum T_{device} \le T, \ \forall device, \quad (9)$$

$$\sum T_{cache} \le k_{opc} \times T, \quad (10)$$

где k_{opc} – доля процессорного времени, которая тратится на обмен с OPC.

Будем полагать, что в текущий момент времени состояние системы полностью определено через статическую составляющую:

- иерархия задач;
- ограничения на актуальность;

и динамическую:

- текущее состояние переменных, т.е. известны сроки окончания их актуальности;
- состояние устройств, обслуживаемых OPC-серверами. Выделим два состояния: устройство свободно либо устройство обрабатывает запрос.

Если известны параметры линейной модели (8), то возможно определение срока завершения запроса и множества переменных, обновляемых в результате запроса.

Эта информация о состоянии системы является основой для решения задачи оптимизации.

Планированию запросов на интервале времени T соответствует график актуальности данных (рис. 2).

Рис. 2. График актуальности данных

Таким образом, актуальность данных на интервале планирования T представляет собой функцию планирования запросов на этом же интервале. Следовательно, функция $n(Task, Queue)$ также зависит от актуальности данных.

Найти решение для такой задачи очень сложно, так как она относится к классу NP-сложных задач, для которых пока не найдено быстрых алгоритмов решения, но проверка того, является ли данное решение правильным, проходит быстро. Время работы алгоритма решения такой задачи также сильно зависит от размера входных данных.

3.3. Метод ветвей и границ

Метод ветвей и границ (МВГ) представляет собой общий алгоритмический метод для нахождения оптимальных решений задач оптимизации. Метод является комбинаторным (алгоритм перебора) с отсевом подмножеств множества допустимых решений, не содержащих оптимальных решений. Этот метод можно использовать для решения некоторых NP-трудных задач.

Рассмотрим задачу дискретной оптимизации:

$$n(\tau) \to \min_{\tau \in T} \left(n(Task, Queue) \right), \quad (11)$$

где T – некоторое конечное множество из R^n.

Для решения данного класса задач часто используется метод ветвей и границ, в основе которого лежат следующие основные модули:

1) Построение дерева вариантов осуществляется на основе разбиения множеств на конечное число непересекающихся подмножеств. Ветвление множества происходит на очередном шаге в соответствии со сформулированным заранее правилом, характеризующее допустимое множество исследуемой задачи. Стратегия планирования запросов имеет

ограничения вида $\mathrm{Queue}_{\mathrm{OPC_k Device_{i,OPC_k}}}\left(\tau_j\right)\in\{0,1\}$, j = 1,…,n (т.е. является задачей с булевыми переменными – данные либо актуальны, либо нет).

2) Оценка функции n(τ) задачи (11) на множестве T – такое число ξ = ξ(T), что (τ)≥ξ, $\forall\,\tau\in$ T. К оценочным задачам предъявляются следующие требования: с одной стороны, их решение не должно занимать много времени, с другой – оценки, полученные с их помощью, должны быть как можно точнее. Оценки подмножеств не должны быть меньше оценки разветвленного множества.

3) Правило обхода дерева вариантов в процессе работы алгоритма называют стратегией обхода дерева. Существует множество различных стратегий. Наиболее распространенными являются три из них [8]: «по минимуму оценки», односторонний обход дерева вариантов и смешанная стратегия.

4) Основное правило сокращения перебора: если оценка множества больше рекордного значения, то такое множество вариантов отбрасывается. Множество может не подвергаться последующему ветвлению, если при решении оценочной задачи на данном множестве была получена точка, являющаяся допустимой в исходной задаче, или если допустимое множество оценочной задачи пусто.

5) Работа метода останавливается в соответствии с критерием оптимальности. Признаком останова является следующая ситуация: не осталось ни одного множества, которое может быть подвергнуто последующему ветвлению. Решением при этом является точка, в которой получено последнее рекордное значение.

3.4. Упрощенный алгоритм динамического планирования запросов

Предположим, что $T_{cache}<<T_{device}$, то есть суммарные затраты времени на получение информации от внешних устройств определяются, в основном, характеристиками этих устройств, а не продолжительностью обработки информации в процессоре. Тогда необходима максимальная загрузка устройств, т.е. $\sum T_{device} = T$, $\forall$device. Вместо планирования на интервале времени будем использовать последовательность действий. При этом каждое действие представляет собой запрос к одному устройству, который выполняется, если устройство свободно в данный момент. Учитывая модель (8), в этом запросе желательно увеличение количества переменных. Вместе с тем, учет сроков завершения актуальности переменных позволяет предложить следующий упрощенный алгоритм динамического планирования:

1) цикл по устройствам, готовым принять запрос;

2) цикл по переменным, начиная с тех, которые раньше теряют свою актуальность;

3) формируем проект запроса, оцениваем изменение функции $\mathrm{n}(\mathrm{Task},\mathrm{Queue})$;

4) если достигнута лучшая оценка $\mathrm{n}(\mathrm{Task},\mathrm{Queue})$, запомнить текущие значения Queue и n;

5) конец цикла по переменным;

6) конец цикла по устройствам.

Заключение

Ключевые для систем реального времени свойства реактивности и предсказуемости в клиент-серверных системах, реализующих технологию OPC, могут быть достигнуты при разработке адаптивного программного обеспечения, что в итоге способствует снижению затрат на аппаратную компоненту и повышению эффективности процессов управления технологическими процессами автоматизированных систем.

Литература

1. Блинов И.В. Системы диспетчерского управления и сбора данных (SCADA-системы) / И.В. Блинов // Мир компьютерной автоматизации. – ДонНТУ, 1999. – № 3. – С. 42-59 [Электрон. ресурс]. – Режим доступа к ресурсу: http://ankey.ru/tech/scada/intro.htm.

2. Давыдов В.Г. OPC-серверы с открытой архитектурой – средства взаимодействия компонентов в промышленной автоматизации / В.Г. Давыдов // Автоматизация в промышленности. – 2003. – №7. – С. 23-27.

3. Куцевич И. Стандарт OPC – путь к интеграции разнородных систем / И. Куцевич, А. Григорьев // PCWeek. – 2001. – №32. – С. 8 [Электрон. ресурс]. – Режим доступа к ресурсу: http://www.rtsoft.ru.

4. Руководство пользователя ИСБ Интеллект. Технология OPC. Основные принципы и преимущества [Электрон. ресурс]. – Режим доступа к ресурсу: http://www.itv.ru.

5. OPC Data Access Custom Interface Specification 3.0. – 2003.

6. Пономарев О.П. SCADA- системы: уч.-мет. пособие / О.П. Пономарев. – Калининград, 2006. – 78 с.

7. Димаки А.В. Интегрированные системы проектирования и управления: уч. пособие / А.В. Димаки. – Томск: Издательство Томск. гос. ун-та систем упр. и радиоэлектроники, 2005. – 170 с.

8. Чернышова Г.Д. Дискретная оптимизация: уч. пособие / Г.Д. Чернышова, И.Л. Каширина. – Воронеж, 2003. – 26 с.

Поступила в редакцию 19.01.2009

Рецензент: д-р техн. наук, проф. В.М. Вартанян, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

АДАПТИВНЕ УПРАВЛІННЯ СЕРВЕРОМ ДАНИХ В SCADA-СИСТЕМАХ

Ю.А. Кузнецова, Є.В. Соколова

Показано, що вимоги постійної підтримки актуальності та цілісності даних, тобто оновлення інформації до закінчення директивних строків, призводить до вирішення оптимізаційної задачі управління OPC-серверами в клієнт-серверних системах, які реалізують технологію OPC (OLE for Process Control). Таким чином досягаються найбільш важливі критерії в системі реального часу – реактивність та передбачуваність. Сформульовано обмеження на абсолютну та відносну коректність даних, за рахунок чого адаптивне управління запитами в клієнт-серверних системах реального часу можна отримати в результаті розв'язання задачі динамічного програмування. Стратегія планування запитів являє собою NP-складну задачу, вирішити яку можна, застосувавши метод гілок та меж.

Ключові слова: системи реального часу, АСУ ТП, SCADA, OLE for Process Control, адаптивне управління, метод гілок та меж.

ADAPTIVE MANAGEMENT BY DATA SERVER IN SCADA-SYSTEMS

Yu. A. Kuznetsova, E.V. Sokolova

This paper shows that requirement of steady support of data relevance and integrity that is of data update to expiration of directive terms leads to solution of optimization management problem by OPC-servers in client-server systems realizing the OPC (OLE for Process Control) technology. In that way the most important criteria in Real-Time system are responsiveness and predictability. The limitations for absolute and relational data reasonableness are formulated, due to this adaptive management by queries in client-server systems of real time can be reached as a result of dynamic programming problem decision. Scheduling strategy of queries is NP-hard task, which can be solved by using the branch and bounds method.

Keywords: Real-Time systems, ACS TP, SCADA, OLE for Process Control, adaptive management, the branch and bounds method.

Кузнецова Юлия Анатольевна – магистрант кафедры инженерии программного обеспечения, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: JK-Sv@yandex.ru.

Соколова Евгения Витальевна – ст. преп. кафедры инженерии программного обеспечения, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: Sev_ai@mail.ru.