УДК 629.78.018

І.Б. Туркін, Є.В. Соколова, Т.С. Нікітіна

Національний аерокосмічний університет ім. Н.Є. Жуковського "ХАІ", Харків

ДИНАМІЧНЕ ПЛАНУВАННЯ ЗАПИТІВ У КЛІЄНТ-СЕРВЕРНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ, ЯКІ РЕАЛІЗУЮТЬ ТЕХНОЛОГІЮ «OLE FOR PROCESS CONTROL»

Показано, що ключові властивості реактивності та передбачуваності в клієнт-серверних системах реального часу, які реалізують технологію OPC (OLE for Process Control) можуть бути досягнуті під час розробки програмного забезпечення, який реалізує алгоритмічний та ресурсний способи адаптації. Сформульовані обмеження на актуальність даних та часову цілісність інформації дозволяють розглядати процес планування запитів у клієнт-серверних системах реального часу як задачу динамічного програмування, розв'язання якої дає політику керування запитами.

системи реального часу, технологія клієнт-сервер, АСУТП, SCADA система, OPC

ВЯтуС

Програмне забезпечення систем реального часу (ПЗ РЧ) потребує забезпечення абсолютної коректності результату за умови обмеження часу, під час якого його потрібно одержати. Врахування часових обмежень потребує від програмних засобів реактивності та передбачуваності.

В сукупності ці властивості характеризують здібність програмно-апаратних засобів реагувати на зміни в контрольованій системі з достатньою швидкістю.

Більшість систем реального часу для диспетчерського управління та збору даних SCADA-систем (Supervisory Control and Data Acquisition System) проектуються за єдиною методикою з застосуванням стандартних типових компонент, архітектурних рішень, технологій обробки, прикладного програмного забезпечення та інструментарію. Загальна схема SCADA-систем включає три рівня:

– нижній – безпосереднє обладнання, технічні пристрої, вимірювальні прилади, локальні бази даних реального часу, з якими пов'язані програмовані логічні контролери, керовані в свою чергу локальними міні-ЕОМ;

– транспортний – різні за архітектурою мережі передачі даних;

– верхній – сервери додатків і баз даних, архіви накопичування даних, обробники подій та команд управління, диспетчерські пульти).

Нижній контролерний рівень містить різні датчики для збору інформації про перебіг технологічного процесу, електропривод та виконуючі механізми для реалізації регулюючих та управляючих впливів, які постачають інформацію локальним логічним контролерам, що програмуються (PLC — Programming Logical Controoller).

Оскільки інформація в контролерах заздалегідь обробляється та частково використовується на місці, суттєво знижуються потреби до пропускної спроможності каналів зв'язку. Інформація з локальних контролерів спрямовується до мережі диспетчерського пункту безпосередньо, або через контролери верхнього рівня.

Верхній диспетчерський рівень містить, перед усим, одну або декілька станцій управління, які являють собою автоматизовані робочі місця диспетчерів/операторів. Тут же можуть бути розміщені сервер бази даних, робочі місця (комп'ютери) для фахівців і т. інш. Станції управління призначені для відображення ходу технологічного процесу та оперативного управління. Ці задачі покликані вирішувати SCADA-системи. Під час застосування апаратури нижнього рівня різних виробників, а також шинних систем промислового призначення з їхніми різноманітними протоколами та інтерфейсами особливо важлива наявність стандартизованої сполучної ланки між SCADA-рівнем і рівнем контролерів. Такою сполучною ланкою є технологія OPC(OLE for Process Control), що забезпечує універсальний механізм обміну даними між датчиками, виконуючими механізмами, контролерами, пристроями зв'язку з об'єктом і системами подання технологічної інформації, оперативного диспетчерського керування, а також системами керування базами даних.

Специфікація OPC [0], розроблена для роботи під керуванням ОС Windows, визначає кілька моделей доступу до даних нижнього рівня: синхронний, асинхронний доступ і метод підписки. OPC, як процедура забезпечення цілісного доступу до виробничих даних, дає наступні переваги:

– виробники компонентів систем автоматизованого керування повинні поставляти своїм клієнтам тільки деякий набір стандартизованих програмних засобів, призначених для сполучення відповідних апаратних компонентів із програмним забезпеченням;

– для розробників програмного забезпечення відпадає необхідність написання нових драйверів, якщо внаслідок модернізації деякого апаратного компонента змінюється набір функцій доступу до його даних;

– замовники одержують більшу можливість вибору під час конфігурування і підбору апаратних засобів розв'язання їхніх задач автоматизації.

Програми-клієнти, яким потрібні дані з АСУ, запитують їх через стандартизований інтерфейс OPC для здійснення зв'язку з будь-якими підтримуючими технологію OPC-серверами. Загальноприйнята схема «клієнт-сервер» дозволяє підключити велику кількість клієнтів до одного сервера, і навпаки – використання одним клієнтом різних OPC-серверів.

Використання OPC-серверів, розроблених організацією-постачальником устаткування, знижує вартість і підвищує якість розробки, але при цьому має наступні недоліки:

1. Принцип приховування інформації - інкапсуляція алгоритмів обробки в OPC-серверах приводить до того, що значна більшість їх принципово важливих властивостей і характеристик може бути встановлена тільки експериментальним шляхом завдяки невідомим архітектурним рішенням і особливостям реалізації OPC-серверів.

2. Структура і основні параметри ПО РЧ, наприклад, конфігурація програмних клієнтів, пропускна спроможність, протоколи і т. і., вибираються на основі попередніх оцінок необхідної продуктивності. Ризик неадекватної розробки ПО РЧ виникає завдяки недостачі продуктивності і пропускної здатності виникаючих потоків даних, або завдяки зайвим запасам обчислювальної потужності, тому розроблювачі систем свідомо збільшують їх ресурси, що не завжди приводить до необхідних результатів.

3. Обробка інформації в СРЧ зв'язана з подіями у контрольованій системі, а інтенсивність цих подій у деяких ситуаціях може змінюватися в десяткисотні разів. Пропускна здатність каналів, необхідна продуктивність процесора повинні гарантувати коректність роботи і в цьому випадку.

Створення алгоритмів, що самоадаптуються до умов роботи, стану системи і зовнішнього середовища, є одним з головних технічних рішень, здатних парирувати перераховані недоліки [0]. Таке ПЗ може реалізувати один із трьох способів адаптації [0]:

– параметричний, коли зміна в часі алгоритмів функціонування відбувається за рахунок підстроювання параметрів. Прикладом можуть бути нейронні мережі, що набувають знання;

– алгоритмічний, який базується на перемиканні з одного алгоритму на інший;

– ресурсний, який зводиться до більш ефективного використання ресурсів системи.

У цей час одним з головних напрямків підвищення якості ПЗ є створення систем, що адаптуються (adaptive – адаптивних) [0], тобто здатних автоматично пристосуватися до умов, що змінюються, за рахунок зміни алгоритмів функціонування, перерозподілу ресурсів, при незмінному складі і структурі.

Актуальність даних і часова цілісність інформації

Дані, що характеризують стан зовнішніх об'єктів, безупинно міняються в часі. У кожному разі програма використовує дані, що характеризують стан об'єкта в минулому, і з часом цей минулий стан може значно відрізнятися від сьогодення. Тому в розрахунках необхідно забезпечити актуальність даних.

Формально об'єкт даних можна описати за допомогою кортежу

$$\mathrm{Data}_i = \left\langle \mathrm{Value}_i, \tau_{\mathrm{querry}_i}, \tau_{\mathrm{answer}_i}, \Delta\tau_i \right\rangle,$$

де Value_i – значення i-го об'єкта даних у деякий момент часу, що перебуває в інтервалі $\tau_{\mathrm{querry}i} \div \tau_{\mathrm{answer}i}$ від моменту формування запиту до моменту одержання відповіді, $\Delta\tau_i$ – довжина інтервалу, протягом якого значення i-го об'єкта даних допускається вважати коректним.

До розгляду вводиться саме інтервал часу, оскільки в результаті пакетного, найбільш часто застосовуваного запиту на введення інформації повертається пакет даних, супроводжуваний оцінкою часу всього пакета, а не окремих даних, які в нього входять.

У будь-який момент часу функціонування ПЗ для інтервалу планування $t \in]t_{\mathrm{current}}, t_{\mathrm{current}} + \Delta t]$ на всю множину даних $\mathrm{Data}(t)$, що змінюється в часі, повинна виконуватися множина обмежень на абсолютну коректність даних:

$$\forall \mathrm{Data}_i \in \mathrm{Data}(t) : -\tau_{\mathrm{querry}_i} < \Delta\tau_i .$$

Крім обмежень на абсолютну коректність даних повинна виконуватися вимога актуальності підмножин даних, використовуваних у певних задачах, як тих що виконуються, так і запланованих до виконання з початком деяких подій, пов'язаних зі зміною стану зовнішнього середовища. Найпростішим прикладом таких задач є розрахунок агрегованих значень, наприклад, визначення максимального значення із множини або розрахунок миттєвої витрати рідини або газу, використовуючи оперативні дані про тиск, його перепади на діафрагмі і температурі.

Очевидно, що за результатами роботи цих елементарних задач-перетворювачів $\mathrm{Task}_i \in \mathrm{Task}$ будуть породжуватися об'єкти даних – $\mathrm{Data}_{\mathrm{out}_{\mathrm{Task}_i}}$, що мають у загальному випадку тимчасові обмеження. Отже, необхідно розглядати розширену множину об'єктів даних, як об'єднання двох непересічних підмножин даних, сенсорних – $\mathrm{Data}_{\mathrm{sensor}_{\mathrm{Task}}}$, отриманих із зовнішнього світу, і тих, що обчислюються – $\mathrm{Data}_{\mathrm{calculated}}$, тобто породжуваних роботою задач. Тоді кожна задача може розглядатися як перетворювач:

$$\forall \mathrm{Task}_i \in \mathrm{Task}, \mathrm{Task}_i :$$

$$: (\mathrm{Data}_{\mathrm{sensor}\mathrm{Task}_i}, \mathrm{Data}_{\mathrm{in}_{\mathrm{task}_i}}) \longrightarrow \mathrm{Data}_{\mathrm{out}_{\mathrm{Task}_i}};$$

$$\mathrm{Data}_{\mathrm{in}_{\mathrm{task}_i}} \in \mathrm{Data}_{\mathrm{calculated}}, \mathrm{Data}_{\mathrm{out}_{\mathrm{out}_{\mathrm{task}_i}}} \in$$

$$\in \mathrm{Data}_{\mathrm{calculated}}, \mathrm{Data}_{\mathrm{in}_{\mathrm{task}_i}} \bigcap \mathrm{Data}_{\mathrm{out}_{\mathrm{Task}_i}} = \varnothing.$$

Коректність вхідних даних задачі-перетворювача буде забезпечена, якщо виконані умови відносної коректності цих даних:

$$\forall j, k, \mathrm{Data}_j \in \mathrm{Data}_{\mathrm{sensor}\mathrm{Task}_i} \bigcup \mathrm{Data}_{\mathrm{in}_{\mathrm{task}_i}},$$

$$\mathrm{Data}_k \in \mathrm{Data}_{\mathrm{sensor}_{\mathrm{Task}_i}} \bigcup \mathrm{Data}_{\mathrm{in}_{\mathrm{Task}_i}} :$$

$$: \max(\tau_{\mathrm{answer}_{\mathrm{Data}_j}}, \tau_{\mathrm{answer}_{\mathrm{Datak}}}) -$$

$$- \min(\tau_{\mathrm{querry}_{\mathrm{Dataj}}}, \tau_{\mathrm{querry}_{\mathrm{Datak}}}) \le \Delta\tau_{\mathrm{Task}_{i_{j,k}}},$$

де $\Delta\tau_{\mathrm{Task}_{i_{j,k}}}$ – призначена максимально припустима величина, що характеризує припустиму тимчасову неузгодженість пари вхідних даних під час розв'язання i-ої задачі.

Для всіх даних, які одержані у результаті розв'язання задач, як і у випадку сенсорних даних повинні бути призначені значення інтервалів часу, протягом якого ці дані можна вважати коректними – $\Delta\tau_i$, при цьому обчислення атрибутів-оцінок часу обчислюється за формулами:

$$\forall \mathrm{Data}_j \in \mathrm{Data}_{\mathrm{out}_{\mathrm{Task}_i}} : \begin{cases} \tau_{\mathrm{querry}_{\mathrm{Data}_j}} = \\ = \min\limits_{\mathrm{Data}_{\mathrm{sensor}_{\mathrm{Task}_i}} \bigcup \mathrm{Data}_{\mathrm{in}_{\mathrm{Task}_i}}} \left(\tau_{\mathrm{querry}}\right); \\ \tau_{\mathrm{answer}_{\mathrm{Data}_{\mathrm{out}_{\mathrm{Task}_i}}}} = \\ = \max\limits_{\mathrm{Data}_{\mathrm{sensor}_{\mathrm{Task}_i}} \bigcup \mathrm{Data}_{\mathrm{in}_{\mathrm{Task}_i}}} \left(\tau_{\mathrm{answer}}\right). \end{cases}$$

Таким чином, запуск задачі на виконання повністю коректний, якщо задоволені абсолютні обмеження на актуальність вхідних об'єктів-даних, а також відносні обмеження, що характеризують припустимі тимчасові неузгодженості між ними. Витрати часу на розв'язання задачі, тобто породження вихідних даних ураховуються приписуванням поточного часу як атрибут часу відповіді.

Наведені вище обмеження в сукупності утворюють критерії придатності вигляду

$$G: \bigcap_{i=1}^{m} (y_{ij} \in \{y_{ij}^{д}\}), [j = 1,..,n],$$

в якому $y_{ij}[i = 1,..,m; j = 1,..,n]$ – показник i -ої властивості j - го об'єкту, $\{y_{ij}^{д}\}$ – множина припустимих значень тимчасових оцінок об'єктів даних.

Для формування критерію оптимальності врахуємо, що в процесі роботи на інтервалі планування $t \in]t_{current}, t_{current} + \Delta t]$ склад задач, що виконуються $Task_0$, може змінюватися з початком деяких подій, згрупованих у множину $Events$. Події цієї множини оброблюються активними задачами, які при виявленні події активізують деякі додаткові задачі-перетворювачі інформації. Тоді розширеною множиною елементарних задач-перетворювачів інформації рівня n будемо називати множину, яка обумовлена рекурсивно:

$$Task_0 = Task_{current},$$
$$Task_{i+1} = Task_i \bigcup Task_{events}(Events, Task_i),$$
$$\forall i, j / i \le j \Rightarrow Task_i \subset Task_j.$$

Поданій ієрархії множин задач буде відповідати ієрархія сенсорних об'єктів даних:

$$Data_{sensor\,Task_0} = Data_{sensor\,Task_{current}},$$
$$Data_{sensor\,Task_{i+1}} = Data_{sensor\,Task_i} \bigcup$$
$$\bigcup Data_{sensor\,Task_{events}(\mathbf{Events}, Task_i)}.$$

Сенсорним об'єктам даних відповідає також інша ієрархія, що пояснюється через джерела їхнього одержання – OPC-сервера. Кожний OPC-сервер одержує інформацію від вимірювальних перетворювачів (device), реалізованих як пристрій вводу-виводу на даній апаратній платформі, або доступних через мережу.

Отже, загальна множина сенсорних даних може бути розбита на неперетинаючі підмножини даних, побудовані з обліком ієрархії OPC-сервер, вимірювальні перетворювачі, сенсорні дані у вигляді:

$$Data_{sensor} = \bigcup_{OPC_k} \bigcup_{Device_{i,OPC_k}} Data_{OPC_k, Device_{i,OPC_k}}.$$

Вимірювальні перетворювачі з погляду на програмування реалізують фізичну паралельність, тому елементарний запит на введення інформації обмежений множиною даних, які можуть бути отримані з одного перетворювача. Тоді формально визначимо j-й запит на введення інформації через k-й OPC-сервер з i-го вимірювального перетворювача у вигляді двійки:

$$Querry_{k,i,j} = \langle Z, Data_{k,i,j} \rangle,$$

де Z – спосіб реалізації запиту, тобто спосіб доступу до інформації, визначений OPC-специфікацією, $Data_{k,i,j} \subset Data_{OPC_k, Device_{i,OPC_k}}$ – множина запитуваних даних.

З урахуванням введених вище позначень стратегія планування запитів на введення інформації в клієнт-серверних системах РЧ, що реалізують технологію OPC, може бути представлена двомірним масивом (за кількістю OPC-серверів і кількості вимірювальних перетворювачів, що обслуговуються кожним OPC-сервером) черг, що характеризують порядок виконання запитів:

$$Queue_{k,i} = \langle Querry_{k,i,0}, \cdots Querry_{k,i,n_{k,i}} \rangle.$$

Тоді адаптивне керування запитами в клієнт-серверних СРЧ, що реалізують технологію OPC, може бути досягнуте в результаті розв'язання задачі динамічного програмування, що формулюється як вибір такої політики (стратегії планування запитів) π^*, яка забезпечує максимум їхньої ефективності:

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

де $n(Task, Queue)$ – функція, яка характеризує глибину вкладеності множини елементарних задач-перетворювачів інформації, що досягається за результатами вибору стратегії планування запитів. Функція розраховується рекурсивно шляхом визначення актуальності даних і цілісності вихідної інформації для виконання розширеної множини елементарних задач-перетворювачів інформації рівня n.

Висновки

В результаті досліджень теоретично обґрунтована архітектура «багато OPC-клієнтів - один субсервер – багато OPC-серверів» для СРЧ і систем критичного призначення, яка дозволить розглядати процес планування запитів у клієнт-серверних СРЧ як задачу динамічного програмування, розв'язання якої дасть політику керування запитами.

Список літератури

1. OPC Data Access Custom Interface Specification 2.0. – 1998. – 344 p.

2. Растригин Л.А. Адаптация сложных систем. – Рига: Зинатне, 1981. – 288 с.

3. Черняк Л. Адаптируемость и адаптивность // Открытые системы. – 2004. – № 9 [Електрон. ресурс]. – Режим доступу: http://www.osp.ru/os/2004/09/030_print.htm.

4. Маккинли Ф., Саджади С.М., Кастен Э., Ченг Б. Композиционная адаптация программ // Открытые системы. – 2004. – № 9 [Електрон. ресурс]. – Режим доступу: http://www.osp.ru/os/2004/09 /020_print.htm.

Надійшла до редколегії 14.12.2007

Рецензент: д-р техн. наук, проф. І.В. Чумаченко, Національний аерокосмічний університет «ХАІ» ім. М.Є. Жуковського, Харків.