Программные системы и вычислительные методы
Правильная ссылка на статью:
Зеленский А.А., Грибков А.А. Конфигурирование память-ориентированной системы управления движением // Программные системы и вычислительные методы. 2024. № 3. DOI: 10.7256/2454-0714.2024.3.71073 EDN: TTQBBA URL: https://nbpublish.com/library_read_article.php?id=71073

Конфигурирование память-ориентированной системы управления движением

Зеленский Александр Александрович

ORCID: 0000-0002-3464-538X

кандидат технических наук

ведущий научный сотрудник; Научно-производственный комплекс "Технологический центр"

124498, Россия, г. Москва, пл. Шокина, 1, строение 7

zelenskyaa@gmail.com

Грибков Андрей Армович

ORCID: 0000-0002-9734-105X

доктор технических наук

ведущий научный сотрудник; Научно-производственный комплекс "Технологический центр"

124498, Россия, г. Москва, пл. Шокина, 1, строение 7

andarmo@yandex.ru

Статья из рубрики "Параллельные алгоритмы решения задач вычислительной математики"

DOI:

10.7256/2454-0714.2024.3.71073

EDN:

TTQBBA

Дата направления статьи в редакцию:

19-06-2024

Дата публикации:

25-07-2024

Аннотация: В статье исследуются возможности конфигурирования цикла управления, т.е. определения распределения интервалов времени, необходимых для выполнения

отдельных операций управления, по потокам исполнения, обеспечивающего реализуемость управления. Параллельное выполнение операций управления, там, где это допускается алгоритмом управления, в случае успешного конфигурирования цикла управления позволяет существенно снизить его длительность. Объектом исследования в данной статье являются системы управления с объектно-ориентированной архитектурой, предполагающей комбинированную вертикально-горизонтальную интеграцию функциональных блоков и модулей, распределяющих между собой все задачи управления. Данная архитектура реализуется посредством акторной инструментальной модели с использованием метапрограммирования. Такие системы управления наилучшим образом обеспечивают сокращение длительности цикла управления за счет параллельного выполнения вычислительных и других операций управления. Рассматриваются несколько подходов к конфигурированию цикла управления: без оптимизации, с комбинаторной оптимизацией по времени, с комбинаторной оптимизацией по ресурсам системы. Также достижение конфигурации, близкой к оптимальной, может быть достигнуто за счет использования адаптивного конфигурирования. Исследования показывают, что задача конфигурирования цикла системы управления имеет несколько вариантов решения. Практическое получение решения задачи конфигурирования в случае комбинаторной оптимизации связано с существенными сложностями, обусловленными высокой алгоритмической сложностью задачи и большим объемом потребных вычислений, быстро растущим по мере увеличения числа операций на этапах цикла управления. Возможным средством преодоления этих сложностей является использование стохастических методов, резко снижающих потребный объем вычислений. Также существенное снижение сложности задачи конфигурирования цикла системы управления можно добиться при использовании адаптивного конфигурирования, имеющего два варианта реализации. Первый вариант - это конфигурирование цикла системы управления в реальном времени. Второй вариант - это определение квазиоптимальной конфигурации на основе многократного конфигурирования с разными исходными данными и последующего сравнения получаемых результатов.

Ключевые слова:

система управления, память-ориентированная, конфигурирование, оптимизация, цикл, элементы, операции управления, потоки исполнения, адаптивный, методы сортировки

Исследование выполнено при поддержке Российского научного фонда по гранту No 24-19-00692, http://rscf.ru/project/24-19-00692/

Введение

Проблематика управления движением в реальном времени, необходимого для современных промышленных роботов, станков и другого технологического оборудования, складывается из нескольких взаимосвязанных составляющих. Центральной из них является проблема реализуемости управления, заключающаяся в определении (и обеспечении) возможности управления движением с быстродействием, необходимым для обеспечения заданной точности воспроизводства скорости и траектории движения. Исследования, проведенные авторами, показали [1, 2], что проблема реализуемости управления обуславливает крайне жесткие требования к быстродействию систем управления сложными объектами. К таким сложным объектам управления, в частности, относятся прецизионное технологическое оборудования с

большим числом одновременно управляемых осей (пять осей и более).

Одной из специфических особенностей управления движением в реальном времени является требование высокого быстродействия системы управления, а не высокой производительности. Производительность системы управления (как и любой вычислительной машины) определяется как среднее число вычислительных операций (или иных операций управления) за единицу времени, причем этот расчет обычно выполняется для достаточно больших объемов вычислений (операций управления) за значительный интервал времени.

Управление в реальном времени в большинстве случаев реализуется в виде сравнительно небольшого набора операций, однако этот набор операций должен быть выполнен за чрезвычайно малый интервал времени – цикл управления, не превышающий единицы миллисекунд, а для наиболее сложных объектов управления сокращающийся до микросекунд. В результате критичным для реализуемости управления становится обеспечение параметра быстродействия системы управления – величины, обратной длительности цикла управления. В более точной формулировке быстродействие – это количество единиц сложности объекта, обслуживаемых системой управления за единицу времени.

Согласно методологии, разработанной авторами с соавторами, сложность объекта управления зависит от числа типов элементов и среднего количества элементов одного типа в системе, числа типов связей и среднего количества связей одного типа в системе, числа контролируемых параметров, посредством которых описывается состояние отдельного элемента системы, а также числа отслеживаемых состояний контролируемого параметра [1].

Быстродействие системы управления движением сложным объектом ограничено действием трех групп факторов, получивших образные названия стена мощности (the power wall), стена памяти (the memory wall) и стена частоты (the frequency wall) [3]. Для практического преодоления этих трех «стен» и повышения в результате быстродействия системы управления движения должны быть решены три задачи, имеющие решение за счет совершенствования архитектуры вычислительных машин: во-первых, необходимо уменьшить объем обрабатываемого потока данных за счет использования параллельных вычислений; во-вторых, необходимо увеличить скорость передачи данных между элементами вычислительной машины за счет применения технологий обработки в памяти (PIM)[4] или обработки вблизи памяти (NMC)[5]; в-третьих, необходимо устранить очереди при одновременном обращении к одной памяти нескольких вычислительных устройств за счет физического разделения памяти между устройствами.

Решать указанные три задачи позволяет память-ориентированная архитектура системы управления движением, при которой данные в процессе вычислений не перемещаются между процессором и памятью, а остаются в памяти, в которую интегрируется процессор. Ключевым отличием память-ориентированной архитектуры от традиционной (процессорно-ориентированной) заключается в отказе от организации управления, при которой оно разделяется на несколько уровней (например, стратегический, тактический и исполнительный). Вместо этого имеет место комбинированная вертикально-горизонтальная интеграция функциональных блоков и модулей, распределяющих между собой все задачи управления.

На программно-алгоритмическом уровне реализация память-ориентированной архитектуры обеспечивается акторной инструментальной моделью, в которой каждый

актор (элемент акторной модели) соответствует элементу системы управления движением, который находит свое отражение в виде задержки и цикле системы управления. Причем этот элемент может быть как реальным, соответствующим функциональному модулю в составе системы управления движения, так и виртуальным, формируемым в памяти (общей или локальной для отдельного функционального модуля) для выполнения частной вычислительной задачи, преобразования данных или генерации управляющих команд для периферийных устройств. Условию максимальной автономности акторов, асинхронно обменивающихся данными в процессе совместной реализации функции управления, соответствует программная реализация системы управления движением на основе метапрограммирования [6]. В этом случае каждый из акторов генерируется по необходимости в процессе управления в виде отдельной программы, запускаемой в общей памяти или в локальной памяти отдельного функционального модуля.

Практическое решение проблемы реализуемости управления движением сложного объекта сводится к распараллеливанию выполнения операций управления и, соответственно, к определению оптимального взаимного распределения затрат времени на отдельные операции с учетом установленных алгоритмических ограничений по последовательности их выполнения. Заметим, что именно с наличием таких ограничений связано существование закона Амдала [7].

Задачей данной статьи является решение указанной задачи оптимального распределения затрат времени на отдельные операции в рамках цикла управления. Кроме того, необходимо определить возможные алгоритмы практической реализации оптимизационных расчетов в том случае, если их объем окажется существенным.

Подходы к конфигурированию цикла системы управления

На рис. 1 приведена диаграмма оптимизированного цикла системы управления промышленного робота [2]. Определение оптимального взаимного распределения затрат времени на отдельные операции представляет собой задачу конфигурирования цикла системы управления движением. Данная задача может быть решена на основе трех основных подходов, каждый из которых, по мнению авторов, имеет право на существование.

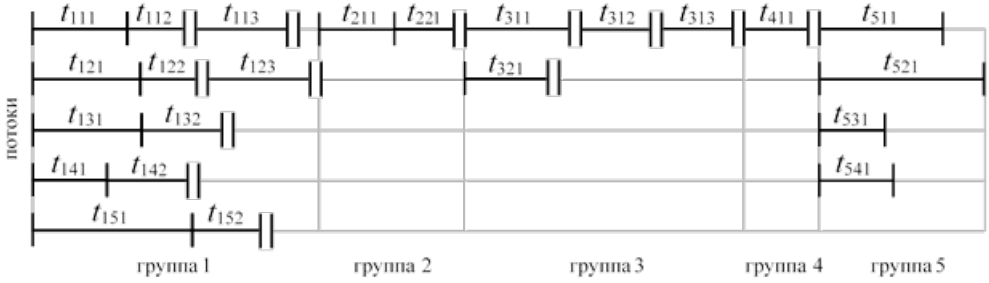

Рисунок 1. Диаграмма оптимизированного цикла СУ промышленного робота:

t_{111}, t_{121}, t_{131}, t_{141}, t_{151} — задержки коммуникационной сети при передаче данных от датчиков и сенсоров в модули очувствления; t_{112} — задержка системы технического зрения, СТЗ (без ИНС); t_{113} — задержка искусственной нейронной сети (ИНС) при обработки данных для СТЗ; t_{122} — задержка системы распознавания речи, СРР (без ИНС); t_{123} — задержка ИНС при обработки данных для СРР; t_{132}, t_{142}, t_{152} — задержки

модулей обработки данных от датчиков положения, скорости и др.; t_{211}, t_{221} — задержки ядра: интерпретатора (не в каждом цикле) и задержка при обработке данных от модулей очувствления; t_{311} — задержка модуля трансформации; t_{312} — задержка модуля интерполяции; t_{313} — задержка модуля эквидистантной коррекции; t_{321} — задержка модуля предварительного просмотра; t_{411} — задержка модуля разгона-торможения; t_{511}, t_{521}, t_{531}, t_{541} — задержки регуляторов исполнительных устройств; **| |** — задержка записи/чтения данных в общей памяти.

Первый подход, назовем его конфигурацией без оптимизации, основан на воспроизведении заданной конфигурации цикла управления с известным (определенным на основе экспертной оценки разработчика) распределением элементов (задержек отработки операций) по группам и потокам исполнения. Примером реализации такого подхода является приведенная на рис. 1 диаграмма оптимизированного цикла системы управления промышленного робота.

Если (для получения более простого представления) включить задержки чтения/записи данных в общей памяти в задержки соответствующих элементов, то длительность цикла системы управления в данном случае определяется следующим образом:

$$\tau = \sum_{u=1}^{u_{\max}} \max\left(\bigcup_{p=1}^{p_{u\max}} \sum_{j=1}^{m_{up}} t_{uj}^{(p)} \right), \qquad (1)$$

где $u_{\max}$ — число групп операций в цикле управления; $p_{u\max}$ — количество параллельных потоков обработки данных в u-ой группе; m_{up} — количество элементов в p-ом параллельном потоков обработки данных в u-ой группе.

Ресурсы, потребные для реализации данной конфигурации, должны быть не меньше суммарных ресурсов, необходимых всем элементам во всех потоках исполнения в каждый момент времени цикла управления.

Второй подход, назовем его конфигурацией с комбинаторной оптимизацией по времени, основан на конфигурировании цикла управления из множества элементов, разбитых на группы (определяемые последовательностью и взаимосвязанностью элементов в рамках алгоритма управления), но не распределенных по потокам исполнения. В процессе оптимизации устанавливается такое распределение по потокам исполнения, при котором длительность цикла управления будет минимальной.

Рассмотрим порядок оптимизации цикла управления по времени.

u-я группа операций формируется из множества элементов $A_u = \{a_{u1}, a_{u2}, \ldots a_{uj}, \ldots a_{un(u)}\}$, каждый из которых $a_{ui} = \{t_{ui}, w_{ui}\}$ характеризуется временем t_{ui}, необходимым для выполнения операции управления, соответствующей данному элементу, и ресурсами w_{ui}, необходимыми для реализации данного элемента. В качестве ресурсов элемента может рассматриваться потребный объем памяти и/или вычислительная мощность (число элементарных операций, необходимых для выполнения рассматриваемой операции управления, за единицу времени ее выполнения).

Элементы a_{ui} не могут быть задействованы в произвольном порядке. Попарное определение порядка задействования элементов задается в виде матрицы отношений

$$K_u = \begin{Bmatrix} k_{11} & k_{12} & \dots & k_{1n_u} \\ k_{21} & k_{22} & \dots & k_{2n_u} \\ \dots & \dots & \dots & \dots \\ k_{n_u 1} & k_{n_u 2} & \dots & k_{nn_u} \end{Bmatrix}, \quad (2)$$

в которой каждый элемент $k_{ij} \in \{-1,0,1\}$ определяет последовательность элементов i и j: «–1» – j потом i, «0» – порядок не имеет значения, «1» – i потом j.

Из элементов u-ой группы формируется выборка для отдельного p-го потока, в которой каждый ее элемент отожествляется с элементом из множества u-й группы:

$$a_{uj}^{(p)} \equiv a_{ui}, \quad (3)$$

причем элементы не повторяются

$$\forall h \neq j : a_{uh}^{(p)} \not\equiv a_{ui} \quad (4)$$

и реализуются в правильном порядке относительно всех предшествующих по времени элементов во всех (ранее определенных) потоках, включая свой,

$$\forall p, d : K_u \left(i\left(a_{u(j_d-1)}^{(d)} \right), i\left(a_{uj_p}^{(p)} \right) \right) \neq -1, \quad (5)$$

где величина j_d для потока d определяется исходя из условия

$$\sum_{q=1}^{j_d} t_{uq}^{(d)} > \sum_{q=1}^{j_p} t_{uq}^{(p)} > \sum_{q=1}^{j_d-1} t_{uq}^{(d)},$$

$i(a_{uj}^{(p)})$ – значение номера элемента в исходном множестве u-ой группы, соответствующее элементу $a_{uj}^{(p)}$ в выборке для p-го потока.

Если $K_u = -1$, то все элементы, уже включенные в выборку (от 1 до j), переставляются исходя из условия (5).

Каждый элемент в выборке для p-го потока, как и в исходном множестве u-ой группы, определяется потребными временем и ресурсами:

$$a_{uj}^{(p)} = \left\{ t_{uj}^{(p)}, w_{uj}^{(p)} \right\}. \quad (6)$$

Состав выборки для p-го потока u-ой группы представляет собой множество из m_{up} элементов

$$A_u^{(p)} = \left\{ a_{u1}^{(p)}, \dots a_{uj}^{(p)}, \dots a_{um_{up}}^{(p)} \right\}, \quad (7)$$

причем сумма элементов всех потоков равна числу элементов nu в исходном множестве u-ой группы и элементы в различных потоках (множествах) не повторяются:

$$\sum_{p=1}^{p_{u\max}} m_{up} = n_u, \forall d \neq p : \nexists \left(A_u^{(p)} \wedge A_u^{(d)} \right). \quad (8)$$

Последнее условие обеспечивается автоматически ввиду последовательного (без пропусков) включения элементов из исходного множестве u-ой группы в выборки для 1-

го, 2-го и последующих потоков.

Для реализации дальнейшей оптимизации необходимо задать функцию суммарных ресурсов, задействованных элементами во всех потоках *u*-ой группы в заданный момент времени:

$$w_u(t) = \sum_{p=1}^{p_{u\max}} w_{uj}^{(p)}, \tag{9}$$

где номер соответствующего элемента *j* определяется из выражения

$$\sum_{q=1}^{j} t_{uq}^{(p)} > t.$$

Теперь можно определить длительность части цикла управления, соответствующей выполнению операций *u*-ой группы в виде множества A_u с заданной последовательностью элементов:

$$\tau_u = \max\left(\bigcup_{p=1}^{p_{u\max}} \sum_{j=1}^{m_{up}} t_{uj}^{(p)}\right), \tag{10}$$

причем должно выполняться условие

$$w_u\left(t \le \tau_u\right) \le w_{u\max}.$$

Следует обратить внимание, что зависимости (9) и (10) не предполагают возможности произвольных задержек между элементами для изменения сочетаний элементов разных потоков для минимизации потребных ресурсов.

Для получения минимального времени отработки операций *u*-ой группы необходимо провести аналогичный анализ для множества A_u с другими последовательностями элементов (число вариантов перестановок $N_u = n_u!$) и другими значениями числа элементов в потоках.

Число вариантов перестановок быстро растет по мере роста числа элементов в множестве. Например, если при $n_u = 10$ (реализуемом при объединении некоторых связанных элементов, см. рис. 1) имеем приемлемое значение $N_u = n_u! = 3.6 \times 10^6$, то при $n_u = 15$ имеем $N_u = n_u! = 1.3 \times 10^{12}$, а при $n_u = 20$ имеем $N_u = n_u! = 2.4 \times 10^{18}$.

Значения числа элементов в потоках для группы *u* могут быть заданы в виде множества

$$M_{uj} = \left\{ m_{u1}^{(j)} \quad m_{u2}^{(j)} \quad \dots \quad m_{up}^{(j)} \quad \dots \quad m_{up_{\max}}^{(j)} \right\}, \tag{11}$$

где $p_{\max}$ – число потоков в *u*-ой группе, причем

$$\sum_{p=1}^{p_{\max}} m_{up}^{(j)} = n_u.$$

Число вариантов множества M_{uj} представляет собой количество упорядоченных разбиений числа n_u элементов в исходном множестве *u*-ой группы и равно $j_{\max} = 2^{n_u-1}$, что в случае $n_u = 10$ дает $j_{\max} = 512$, в случае $n_u = 15$ – $j_{\max} = 1.6 \times 10^4$, а в случае $n_u = 20$ – $j_{\max} = 5.2 \times 10^5$.

Минимальное время отработки операций u-ой группы:

$$\tau_u^{\min} = \min\left[\bigcup_{i=1,j=1}^{N_u, j_{\max}} \tau_u\left(A_{ui}, M_{uj}\right)\right], \quad (12)$$

где M_{uj} – j-й (из $j_{\max}$) вариант множества M_{uj}, A_{uj} – i-я (из N_u) перестановка исходного множества A_u.

Исходя из приведенных выше зависимостей, число вариантов, задаваемых $\tau_u(A_{ui},M_{ui})$, равно $N_u \cdot j_{\max} = n_u! \cdot 2^{n_u-1}$ и составляет в зависимости от числа элементов в исходном множестве u-ой группы от 1.8×10^9 при $n_u = 10$ до 2.1×10^{16} при $n_u = 15$ и 1.2×10^{24} при $n_u = 20$. Последние числа вариантов слишком велики и не могут быть на практике проанализированы.

Максимальная сложность данного оптимизационного алгоритма соответствует нотации $O(n!\cdot 2^{n-1}\cdot n^2)$, где n^2 – сомножитель, связанный с сортировкой выборки перебором из исходного множества u-ой группы. Данный характер сложности оказывает критическое негативное влияние на практическую реализуемость полного перебора всех возможных вариантов. Такая ситуация является типичной для задач комбинаторной оптимизации.

Средством решения данной задачи комбинаторной оптимизации при большом числе элементов в группе является использование стохастических методов со случайной генерацией последовательности элементов в исходном множестве A_u для u-ой группы и множества M_{uj} значений числа элементов в потоках этой группы. Для повышения эффективности случайной выборки можно целенаправленно обеспечивать существенное различие последовательностей элементов, разбиение групп на потоки с сопоставимым числом элементов, ограничивать вариативность числа потоков исходя из доступных ресурсов для всей группы и средних значений ресурсов отдельных элементов, собирать потоки из элементов исходного множества не последовательно, а параллельно (поочередно добавляя элементы в разные потоки) и т.д. В результате N_u и $j_{\max}$ не будут зависеть от n_u (будут примерно постоянными, устанавливаемыми исходя из экспертной оценки разработчика), сложность алгоритма существенно снизится и может соответствовать сложности сортировки, имеющей в случае простого перебора нотацию $O(n^2)$.

Критерием оценки репрезентативности случайной выборки может служить монотонность изменений расчетных значений вблизи минимального значения длительности цикла управления.

Длительность цикла управления определяется следующим образом:

$$\tau = \sum_{u=1}^{u_{\max}} \tau_u^{\min}. \quad (13)$$

Третий подход, назовем его конфигурацией с комбинаторной оптимизацией по ресурсам, также, как и второй, основан на конфигурировании цикла управления из множества элементов, разбитых на группы, но не распределенных по потокам исполнения. В процессе оптимизации устанавливается такое распределение по потокам исполнения, при котором длительность цикла управления не превысит заданную, а потребные ресурсы будут минимальными.

Рассмотрим порядок оптимизации цикла управления по ресурсам.

Для данного варианта оптимизации справедливы формулы (2-10), полученные нами ранее для варианта оптимизации по времени.

Ресурсы, потребные для реализации всех операций *u*-ой группы цикла управления:

$$w_u = \max\left[w_u\left(t = 0 \ldots \tau_u\right)\right]. \quad (14)$$

При реализации оптимальной по ресурсам конфигурации цикла управления (минимальные) потребные ресурсы составят

$$w = \max\left(\bigcup_{u=1}^{u_{\max}} \min\left[\bigcup_{i=1,\,j=1}^{N_u,\,j_{\max}} w_u\left(A_{ui}, M_{uj}\right)\right]\right). \quad (15)$$

причем длительность цикла управления не должна превысить заданное максимальное значение:

$$\tau = \sum_{u=1}^{u_{\max}}\left[\max\left(\bigcup_{p=1}^{p_{u\max}} \sum_{j=1}^{m_{up}} t_{uj}^{(p)}\right)\right] \le \tau_{\max}.$$

Конфигурация цикла управления с оптимизацией по времени или ресурсам – оптимизационные задачи, относящаяся к группе NP-полных задач. Ограничениями задачи конфигурирования цикла управления с комбинаторной оптимизацией являются, соответственно, допустимые (вычислительные) ресурсы системы управления или длительность цикла управления.

Ввиду высокой алгоритмической сложности рассматриваемых оптимизационных задач ни один из типовых оптимизационных алгоритмов, используемых для комбинаторной оптимизации, не может в полной мере ее воспроизвести. В частности, не удается построить оптимизационные алгоритмы на взвешенных ориентированных графах, например алгоритме Декстры, алгоритме Флойда-Уоршелла и др. Необходимость использования взвешенных ориентированных графов обусловлена тем, что элементы конфигурации цикла управления имеют веса и ограничения по последовательности. Проблема в построении оптимизационных алгоритмов связана с тем, что в известных алгоритмах вес элементов (ребер графа) влияет на построение маршрута локально, определяя последовательность пары вершин, между тем в нашей оптимизационной задаче ребра графа (время или ресурсы элемента) хотя и имеют вес, но он сказывается лишь в сочетании с весами других ребер графа. В результате один и тот же вес ребра в разных позициях конфигурации цикла управления оказывает разное влияние на длительность цикла управления или потребные для управления ресурсы.

Существующие алгоритмы оптимизации, однако, могут быть использованы для решения локальных задач в рамках комбинаторной оптимизации. В частности, использование алгоритмов сравнения при упорядочивании выборки для *p*-го потока *u*-ой группы позволяет снизить время выполнения алгоритма сортировки с $O(n^2)$ до $O(n \log n)$ [8, с. 206-207].

К числу лучших по времени и памяти алгоритмов сортировки относятся [9]: merge sort (сортировка слиянием [8, с. 181-192]), heapify и heap sort (пирамидальная сортировка [10]) – лучшее, среднее и худшее время $O(n \log n)$, память $O(n)$; quik sort (сортировка Хоара) — лучшее и среднее время $O(n \log n)$, худшее время $O(n^2)$, память $O(\log n)$; tree sort (сортировка с помощью двоичного дерева) – лучшее время $O(n)$, среднее и худшее

время $O(n \log n)$, память $O(n)$; timsort (гибридный алгоритм сортировки, сочетающий сортировку вставками и сортировку слиянием) и binary insertion sort (гибридный алгоритм сортировки, сочетающий сортировку вставками с бинарным поиском места вставки [11]) – лучшее время $O(n)$, среднее и худшее время $O(n \log n)$, память $O(n)$.

Адаптивное конфигурирование цикла системы управления

Наряду с рассмотренными вариантами комбинаторной оптимизации конфигурации цикла управления по времени или ресурсам, также может быть использовано адаптивное конфигурирование цикла управления. Это конфигурирование, как и рассмотренные варианты оптимизации, применяется к группам элементов, а итоговые результаты получаются после объединения полученных для них результатов. При этом в каждый момент времени задействуются максимально доступные ресурсы, обеспечивая тем самым минимизацию времени исполнения всех элементов группы.

В каждый момент времени нам известны элементы, реализуемые на каждом потоке u-ой группы $\{a_u^{(1)},a_u^{(2)},...a_u^{(p)},...a_u^{(p_{\max})}\}$, причем каждый элемент взят из исходного множества элементов u-ой группы, после чего в исходном множестве он помечается как использованный. Это реализуется заданием для элементов a_{ui} в исходном множестве, наряду со временем и ресурсами, параметра $z_{ui} \in \{1,0\}$ «использован» / «не использован».

Когда время выполнения какого-либо из элементов заканчивается, на его место (в тот же поток) назначается неиспользованный элемент из исходного множества, для которого выполняются два условия. Согласно первому условию, назначаемый q-ый элемент должен быть задан к выполнению не позже любого из оставшихся незадействованных элементов, т.е.

$$\forall 1 \le j \le n_u, j \ne i\left(a_u^{(q)}\right), z_{uj} = 0 : K_u\left(i\left(a_u^{(q)}\right), j\right) \ne -1, \qquad (16)$$

а согласно второму условию, назначаемый q-ый элемент должен обладать максимальной потребностью в ресурсах из всех оставшихся неиспользованными в исходном множестве, но при этом суммарные потребные ресурсы всех потоков (включая поток назначаемого элемента) не должны превышать допустимого максимального значения, т.е.

$$w_u^{(q)} = \max\left(\bigcup_{i=1}^{n_u} w_{ui}, z_{ui} = 0, \sum_{p=1}^{p_{\max}} w_u^{(p)} \le w_u^{\max}\right). \qquad (17)$$

Если после назначения нового элемента в системе остаются ресурсы, то по тому же алгоритму назначается еще один элемент, для которого добавляется новый поток.

После завершения выполнения элементов 1-ой группы запускается выполнение элементов 2-ой группы и т.д. Цикл управления завершается тогда, когда заканчиваются элементы во множествах всех групп.

Запуск адаптивного конфигурирования осуществляется с заданных экспертной оценкой разработчика стартовых элементов (т.е. элементов, выполняемых, согласно алгоритмы управления, первыми). Потоки добавляются пока задействованные во всех потоках ресурсы меньше имеющихся в наличии в системе.

Для реализации адаптивного конфигурирования, которое в общем случае не гарантирует минимизации длительности цикла управления, могут быть использованы указанные выше алгоритмы оптимизации сортировки. Они применимы как для реализации условия (16),

так и (17).

Адаптивное конфигурирование может быть использовано для решения двух различающихся задач: во-первых, для распределения элементов цикла управления в процессе управления, т.е. конфигурирования в реальном времени; во-вторых, для определения квазиоптимальной конфигурации цикла управления на основе многократного адаптивного конфигурирования и сравнения его результатов.

В случае адаптивного конфигурирования в реальном времени частота переопределения конфигурации ограничена интервалом времени, необходимым для его расчета. Вероятно, этот интервал времени много больше длительности цикла управления, т.е. переопределение конфигурации цикла управления происходит один раз за сотни или даже тысячи циклов управления. По мнению авторов, это является допустимым, поскольку в том случае, когда вследствие корректирования алгоритма управления требуется переопределение времени выполнения элементов, оно происходит за время, соответствующее (например, для систем управления движением) существенным изменениям объекта управления (например, перемещениям рабочего органа).

Многократное адаптивное конфигурирование цикла управления предполагает его выполнение с разными перестановками исходного множества элементов заданной группы.

По результатам *j*-го адаптивного конфигурирования формируется множество

$$B_{uj} = \left\{ b_{uj1}, \ldots b_{ujn}, \ldots b_{ujn_u} \right\}, \tag{18}$$

образованное элементами $b_{ujn} = \{u, i, p\}$, где $\{u, i, p\}$ – номер группы, номер элемента в исходном множестве элементов *u*-ой группы и номер потока для *n*-го элемента множества B_{uj}.

Интервал времени, необходимый для выполнения элементов *p*-го потока *u*-ой группы, определяется из условия

$$\forall b_{ujn} = \left\{ u, i, p \right\}, 1 \le n \le n_u : \tau_{uj}^{(p)} = \sum\nolimits_{i=1}^{n_u} t_{ui}. \tag{19}$$

Длительность части цикла управления, за которую выполняются все элементы *u*-ой группы, соответствующая *j*-ой реализации адаптивного конфигурирования, определяется из условия

$$\tau_{uj} = \max\left(\bigcup\nolimits_{p=1}^{P_{\max}} \tau_{uj}^{(p)} \right). \tag{20}$$

Наименьшая длительность части цикла управления, за которую выполняются все элементы *u*-ой группы, из всех $j_{\max}$ реализаций адаптивного конфигурирования:

$$\tau_u = \min\left(\bigcup\nolimits_{j=1}^{j_{\max}} \tau_{uj} \right). \tag{21}$$

Наименьшая длительность цикла управления, получаемая по результатам многократного адаптивного конфигурирования:

$$\tau = \sum\nolimits_{u=1}^{u_{\max}} \tau_u. \tag{22}$$

Выводы

Резюмируем проведенное в статье исследование:

1. Управление сложными объектами в реальном времени требует для обеспечения своей реализуемости высокого быстродействия системы управления. Высокое быстродействие предполагает возможность выполнения ограниченного (сравнительно небольшого) набора операций управления за малый интервал времени – цикл управления, не превышающий единицы миллисекунд или даже (для наиболее сложных объектов управления) микросекунд.

2. Требуемое сокращение длительности цикла управления может быть достигнуто за счет параллельного выполнения вычислительных и других операций управления. Наилучшим образом это обеспечивается при использовании память-ориентированной архитектуры системы управления, реализуемой посредством акторной инструментальной модели, программируемой средствами метапрограммирования.

3. На практике задача обеспечения реализуемости управления сводится к конфигурированию цикла управления, т.е. определению взаимного распределения затрат времени на различные операции управления по потокам исполнения согласно заданным ограничениям по их последовательности и доступным вычислительным ресурсам системы.

4. Возможны три подхода к конфигурированию цикла управления: без оптимизации с конфигурацией, заданной на основе экспертной оценки разработчика, а также конфигурирование с комбинаторной оптимизацией по времени или по ресурсам системы. Комбинаторная оптимизация (в обоих случаях) характеризуется крайне высокой алгоритмической сложностью и на практике должна быть упрощена введением стохастических методов.

5. Менее точным и достоверным подходом к конфигурированию цикла системы управления является адаптивное конфигурирование, которое может реализоваться в реальном времени, либо, при многократном использовании, в качестве инструмента определения квазиоптимальной конфигурации цикла системы управления.

Библиография

1. Зеленский А.А., Кузнецов А.П., Илюхин Ю.В., Грибков А.А. Реализуемость управления движением промышленных роботов, станков с ЧПУ и мехатронных систем. Часть 1 // Вестник машиностроения. 2022. №11. С. 43-51
2. Зеленский А.А., Кузнецов А.П., Илюхин Ю.В., Грибков А.А. Реализуемость управления движением промышленных роботов, станков с ЧПУ и мехатронных систем. Часть 2 // Вестник машиностроения. 2023. №3. С. 213-220
3. Cell Broadband Engine Programming Tutorial. Version 2.0. IBM Systems and Technology Group, December 15, 2006. URL: https://arcb.csc.ncsu.edu/~mueller/cluster/ps3/CBE_Tutorial_v2.0_15December2006.pdf
4. Ghose S., Hsieh K., Boroumand A., Ausavarungnirun R., Mutlu O. Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions. 2018. URL: https://arxiv.org/abs/1802.00320
5. Singh G., Chelini L., Corda S., Awan A.J., Stuijk S., Jordans R., Corporaal H., Boonstraz A. Near-Memory Computing: Past, Present, and Future. August 2019, Microprocessors and Microsystems 71. URL: https://www.researchgate.net/publication/335028505_Near-Memory_Computing_Past_Present_and_Future
6. Зеленский А.А., Ивановский С.П., Илюхин Ю.В., Грибков А.А. Программирование

доверенной память-центрической системы управления движением робототехнических и мехатронных систем // Вестник Московского авиационного института. 2022. Т. 29. №4. С. 197-210
7. Juurlink B., Meenderinck C. Amdahl's law for predicting the future of multicores considered harmful // ACM SIGARCH Computer Architecture News, 40 (2012), 2. pp.1-9
8. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск. М.: ООО "И.Д. Вильямс", 2018. 832 с.
9. Time Complexities of all Sorting Algorithms. Geeks for Geeks, 2023. URL: https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/
10. Heap Algorithms. Massachusetts Institute of Technology, 2010. URL: https://courses.csail.mit.edu/6.006/fall10/handouts/recitation10-8.pdf
11. Binary Insertion Sort. Geeks for Geeks, 2023. URL: https://www.geeksforgeeks.org/binary-insertion-sort/

Результаты процедуры рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Статья посвящена проблематике управления движением в реальном времени для промышленных роботов и другого технологического оборудования. Авторы фокусируются на разработке и оптимизации память-ориентированной архитектуры системы управления, обеспечивающей высокое быстродействие и точность управления.
В статье предложена методология, направленная на разработку и оптимизацию системы управления движением промышленных роботов с использованием память-ориентированной архитектуры. Центральная идея этой методологии заключается в том, что данные в процессе вычислений не перемещаются между процессором и памятью, а остаются в памяти, в которую интегрируется процессор. Это позволяет уменьшить задержки при передаче данных и повысить общую производительность системы за счет использования технологий обработки в памяти или обработки вблизи памяти.
Одним из ключевых аспектов методологии является акторная инструментальная модель, где каждый элемент системы управления движением представлен актором, соответствующим функциональному модулю системы. Акторы генерируются по необходимости в процессе управления в виде отдельных программ, запускаемых в общей памяти или в локальной памяти отдельного функционального модуля. Для программной реализации системы управления движением используется метапрограммирование, что позволяет генерировать и исполнять программы на основе текущих потребностей системы.
Конфигурирование цикла управления заключается в определении взаимного распределения затрат времени на различные операции управления по потокам исполнения согласно заданным ограничениям по их последовательности и доступным вычислительным ресурсам системы. Предложены три подхода к конфигурированию: без оптимизации, с комбинаторной оптимизацией по времени и с комбинаторной оптимизацией по ресурсам. В первом подходе воспроизводится заданная конфигурация цикла управления с известным распределением элементов по группам и потокам исполнения. Во втором подходе формируются циклы управления из множества элементов с целью минимизации длительности цикла управления. В третьем подходе распределение элементов по потокам исполнения осуществляется таким образом, чтобы минимизировать потребные ресурсы при соблюдении заданной длительности цикла управления.

Для решения задач комбинаторной оптимизации при большом числе элементов в группе применяются стохастические методы, которые обеспечивают снижение сложности алгоритмов и улучшают их практическую реализуемость. Также рассматривается адаптивное конфигурирование цикла управления, при котором задействуются максимально доступные ресурсы в каждый момент времени для минимизации времени исполнения всех элементов группы. Адаптивное конфигурирование может использоваться как в реальном времени, так и для многократного адаптивного конфигурирования с целью определения квазиоптимальной конфигурации цикла управления.

Актуальность исследования обусловлена необходимостью повышения быстродействия систем управления движением сложных объектов, таких как промышленные роботы и станки с ЧПУ. В условиях современной промышленности, где требуется высокая точность и скорость обработки данных, предложенные решения могут значительно улучшить производительность и надежность таких систем.

Научная новизна работы заключается в предложении памяти-ориентированной архитектуры системы управления движением, которая отличается от традиционной процессорно-ориентированной. Использование акторной модели и метапрограммирования для реализации этой архитектуры представляет собой инновационный подход к решению задач управления в реальном времени.

Статья написана в научном стиле, характеризующемся высокой точностью и логичностью изложения. Структура работы включает введение, обзор существующих проблем, описание предложенных методологий и подходов, экспериментальные результаты и выводы. Содержание статьи достаточно полное и охватывает все аспекты заявленной темы.

Выводы статьи подтверждают эффективность предложенных подходов к конфигурированию цикла управления. Авторы демонстрируют, что использование памяти-ориентированной архитектуры позволяет значительно сократить время выполнения операций управления и повысить общую производительность системы.

Статья будет интересна специалистам в области управления движением промышленных роботов, а также исследователям, занимающимся разработкой высокопроизводительных систем управления. Предложенные методы и результаты могут быть полезны для дальнейших исследований и практических разработок в этой области.

Рекомендации по доработке:

1. Предоставить более подробные результаты экспериментальных исследований, включая сравнение с альтернативными подходами и количественные показатели улучшений.
2. Углубленно описать алгоритмы, использованные для оптимизации конфигурирования цикла управления, и привести примеры их практической реализации.
3. Рассмотреть возможные ограничения предложенной архитектуры и условия, при которых её применение может быть неэффективным.

Заключение

Статья представляет собой значимый вклад в область управления движением промышленных роботов. Предложенные методологии и результаты исследований имеют высокую научную и практическую ценность. С учетом предложенных доработок, статья может быть рекомендована к публикации.

Рекомендуемое решение по статье: направить статью на доработку с последующей повторной рецензией.