Розглянуто деякі системи розпаралелювання: векторизатор KAP, векторизуючий транслятор Fortran-77VP, системи розпаралелювання і SUIF і OPC. Описано їх принципи роботи, внутрішні уявлення, можливості по распаралелюванню і роботи в діалоговому режимі

Ключові слова: KAP, BPC, SUIF, розпаралелювання, система розпаралелювання

Рассмотрены некоторые системы распараллеливания: векторизатор KAP, векторизующий транслятор Fortran-77VP, системы распараллеливания и SUIF и OPC. Описаны их принципы работы, внутренние представления, возможности по распараллеливанию и работы в диалоговом режиме

Ключевые слова: KAP, OPC, SUIF, распараллеливание, система распараллеливания

Some systems of paralleling are considered: KAP vectorizer, Fortran-77VP vectorized translator, systems of parallelizing and SUIF and OPS. Their principles of operation, internal representation, ability to work in parallel in interactive mode are described

Keywords: KAP, OPS, SUIF, paralleling, parallelizing system

УДК 004.057.6, 004.4'4, 519.681.3

ОГЛЯД РОЗПАРАЛЕЛЮЮЧИХ СИСТЕМ

І.М. Сиволовський
Викладач
Кафедра інформаційних систем і медичних технологій
Національний університет ім. Богдана Хмельницького
бул. Шевченка, 79, м. Черкаси, Україна, 18000
Контактний тел.: (047) 266-70-75, 093-575-40-32
E-mail: grom193@gmail.com

1. Вступ

Проблема прискорення швидкодії виконання будь-якої програми завжди була дуже гострою. Існує декілька шляхів вирішення цієї проблеми:

– оптимізація алгоритму програми;
– прискорення роботи комп'ютерної системи.

Прискорення роботи комп'ютерної системи за рахунок збільшення частоти роботи процесорного елементу можливе тільки до деякого порогового значення. Це зумовлено насамперед обмеженнями фізичних законів при використанні існуючих на даний час технологій створення процесорних елементів. Тому одним з виходів з цієї ситуації стало створення обчислювальних систем з кількома процесорними елементами. Спочатку такі системи, здебільшого, використовувались науковцями, але зараз сучасний персональний комп'ютер має в своєму складі декілька обчислювальних елементів (ядер або процесорів). Ці ядра можуть виконувати одночасно (паралельно) декілька задач. Паралельне виконання декількох задач прискорює роботу комп'ютера, але не прискорює виконання звичайної програми. Для прискорення роботи програми на багатоядерній або багатопроцесорній системі необхідно провести оптимізацію коду даної програми. В цьому випадку оптимізація коду програми – це пошук або створення коду, який можна виконувати незалежно (паралельно), іншими словами провести розпаралелювання.

З огляду на кількість видів багатопроцесорної техніки і на їх архітектурні відмінності, стає зрозумілим, що одна і та сама паралельна програма або взагалі не буде виконуватися на деяких видах таких систем або її виконання буде неефективним. Виходячи з цього можна зробити висновок, що розпаралелювання необхідно робити з урахуванням архітектури багатопроцесорної паралельної комп'ютерної системи.

Ручна оптимізація коду програми є ефективною в руках професіонала, але це довготривалий процес. Тому було розроблено і розробляється велика кількість інструментів для автоматизованого проведення розпаралелювання коду програми. Кожна з таких систем має свої обмеження та недоліки.

2. Основна частина

До систем розпаралелювання належать: векторизатори (PTRAN, KAP, PFC, CFT, FTN 200, Fortran-77/VP, FORT 77/HAP для HitachiS-810, компілятори фірми Convex, Pascal-XT), розпаралелюючі компілятори та транслятори (PascalABC.NET, Intel C++ Compiler, gcc); інструментальні засоби динамічного та статичного аналізу коду програми на можливість розпаралелювання (PTOOL, R^n, ParaScope, PAT,SUIF, PTOPP, PEPP).

KAP [1, 2] – це сімейство препроцесорів-векторизаторів, кожен член якого налаштований на конкретну об'єктну машину.

У версії KAP для системи Sequent досліджуються такі особливості циклів: чи є в даному циклі критичний інтервал, чи є оператори IF або CALL, чи велика кількість ітерацій, чи містить цикл операції редукційного типу. Слід зазначити, що за наявності в тілі циклу умовних операторів та/або викликів зовнішніх процедур навіть автоматичний динамічний аналіз не завжди дає надійні оцінки. У цих випадках

залишається лише емпіричне вивчення динамічних профілів різних варіантів коду програми, вибір програмістом найкращого варіанту і повідомлення про цей вибір транслятору за допомогою спеціальних директив.

KAP/205 [1, 3] – векторизатор для Cyber 205 – призначений для полегшення ефективного виконання програм на Cyber 205, так як векторизатор універсального типу не вирішує всіх проблем, що постають перед користувачами Cyber 205. Векторизатор сприймає програми на мові Фортран 200 і векторизує DO-цикли. При цьому KAP/205 використовує векторні позначення мови Фортран 200, вбудовані векторні функції і спеціальні виклики Q8 в асемблерному коді. Векторизатор видає повідомлення та питання, що стосуються тих частин програми, які в подальшому можна буде поліпшити за додатковою допомогою користувача.

Адаптація KAP для Cyber 205 полегшується завдяки модульним принципам побудови KAP. На одній з перших стадій роботи KAP запускається розпізнавач кандидатів. На цій стадії аналізу DO-циклів з'ясовується, чи є операції всередині них легальними кандидатами на векторизацію. Векторизатор KAP/205 позначає оператори, що включають операції які не можуть бути векторизовані (наприклад операції типу doubleprecision). Таким же чином позначаються оператори READ і WRITE. Ті DO-цикли, які повністю складаються з невекторизованих операторів, позначаються як повністю послідовні і більше не досліджуються.

KAP/ST-100 [2, 4] – векторизуючий препроцесор – складається з сканера, розпізнавача кандидатів (на векторизацію), розпізнавача процесів і власне векторизатора. Розпізнавач кандидатів шукає і позначає DO-цикли, які є кандидатами для векторизації, наприклад відкидається будь-який оператор, що містить операнди з подвійною точністю. Кожен DO-цикл перевіряється; якщо DO-цикл не має операторів-кандидатів, то весь цикл позначається як непридатний і до нього не буде застосовуватися векторизація.

З метою поліпшення векторизації KAP/ST-100 виконує такі загальні перетворення, як розпізнавання індуктивних змінних і перетворення скалярів у масиви. Крім того, розпізнаються деякі IF-конструкції, які обчислюють максимум серед компонент вектора або знаходять індекс цього максимуму.

З метою знаходження найкращого варіанту векторизації виконуються такі перетворення, як перестановка і розподіл циклів. Для перевірки коректності цих перетворень використовується граф залежностей за даними.

Fortran-77VP – це векторизуючий варіант транслятора стандартної мови Фортран 77 для векторних процесорів сімейства FACOMVP японської фірми Fujitsu [1, 5].

Аналогічно іншим системам векторизація тут заснована на побудові графа залежностей усередині ітеративних циклів. Транслятор може частково векторизувати цикл, тобто розщепити його тіло на частини, що векторизуються та що невекторизуються, підставити текст зовнішньої Фортран-процедури на місце звернення до неї з тіла циклу. Можливий діалог з користувачем в процесі налагодження програми. За допомогою спеціальних директив, що вставляються в текст програми, користувач може повідомити транслятору:

- про відсутність залежності між елементами певного масиву;
- про кількість ітерацій циклу;
- про відносну кількість спрацьовувань альтернатив логічного умовного оператора IF.

Іншими словами, за допомогою операторів управління оптимізацією, оформлених як коментарі, може зазначатися, наприклад, коефіцієнт істинності умови; рекомендована довжина векторів при виконанні оператора циклу, що дозволяє найкращим чином використовувати можливості регістрів; спосіб примусової векторизації циклів, які носять очевидний рекурентний характер через присутності невідомих величин, і, навпаки, примусовий перехід до скалярної форми, коли заздалегідь відомо, що довжина циклу мала.

За допомогою інтерактивного векторизатора можлива перебудова програми в діалоговому режимі. Програміст може ввести в систему відомості про те, які оператори циклу відносяться до тих що векторизуються, як підвищити швидкість обрахунку і т.д. Крім того, працюючи в інтерактивному режимі, програміст може змінювати текст вихідної програми, вводити оператори управління векторизацією та ін., контролюючи процес зміни програми на екрані.

Система SUIF (Stanford University Intermediate Format), яка розроблена в Стенфордському університеті, являє собою інфраструктуру для досліджень в області розпаралелюючих і оптимізуючих компіляторів.

SUIF була розроблена як платформа для дослідження методів компіляції для високопродуктивних машин. Ця потужна, модульна, гнучка, ясно документована і досить повна для компіляції великих тестових завдань система була успішно використана для проведення досліджень у таких областях, як скалярна оптимізація, аналіз залежностей в масивах даних, перетворення циклів з точки зору локальності і паралелізму, програмне зчитування з попередженням, планування команд. Дослідницькі проекти, що використовують SUIF, включають в себе декомпозицію глобальних даних і обчислень як для машин з загальною, так і з розподіленою пам'яттю, приватизацію масивів, міжпроцедурне розпаралелювання, ефективний аналіз покажчиків.

Виходячи з цілей створення, структура системи являє собою невелике ядро (kernel) плюс інструментарій (toolkit), що складається з різних компіляторних аналізів і оптимізацій. Ядро визначає проміжне представлення і інтерфейс між проходами компілятора. Цей інтерфейс завжди один і той же, так що проходи в інструментарії можуть бути легко включені, замінені і упорядковані.

Інструментарій SUIF містить велику кількість проходів. Препроцесор для Фортрану і ANSI C доступні для трансляції вихідних програм у SUIF, який включає паралелизатор, здатний автоматично знаходити паралельні цикли і породжувати розпаралелюваний код. Система SUIF забезпечує багато можливостей для підтримки розпаралелювання: аналіз залежностей за даними, розпізнавання ре-

дукції, велику кількість прийомів для покращення виявлення паралелізму і унімодулярні перетворення для збільшення паралелізму і локальності.

Коли компіляція доходить до стадії оптимізації та кодогенерації, високорівневі конструкції більше не потрібні і розширюються до операцій нижнього рівня. Результат такого розширення – простий список команд для кожної процедури. Інформація з високорівневого аналізу може бути легко приведена до низькорівневого представлення за допомогою анотацій. Наприклад, інформація про залежності за даними може бути доведена до команд планування за допомогою анотування команд завантаження і запису в пам'ять.

Компіляційна частина системи SUIF включає С- і Фортран-препроцесори, оптимізатор паралелізму і локальності на рівні циклів, що оптимізує MIPS-постпроцесор і безліч інструментів для розробки компіляторів.

Щоб досягти високої продуктивності на сучасних архітектурах, програми повинні ефективно використовувати ієрархію пам'яті комп'ютера, а також його здатність виконувати операції паралельно. Ключовим моментом компіляційного інструментарію SUIF є, таким чином, бібліотека і драйвер для здійснення оптимізації рівня циклів і локальності.

SUIF-паралелізатор переводить послідовні програми в паралельний код для машин із загальною пам'яттю. Компілятор породжує SPMD-програму, яка містить виклики бібліотеки часу виконання, що переноситься. Потім використовується SUIF-to-C-транслятор для конвертації SUIF в ANSI Ci.

SUIF-паралелізатор зроблений з багатьох різних проходів компілятора. По-перше, деяке число скалярних оптимізацій допомагає виявляти паралелізм. Останні операції включають протягування констант (constant propagation), протягування вперед (forward propagation), виявлення індуктивних змінних, згортання констант (constant folding) і аналіз приватизації скалярів (scalar privatization). Далі застосовуються унімодулярні перетворення циклів для реструктуризації коду з метою оптимізації та збільшення ступеня паралелізму. Нарешті, генератор паралельного коду породжує код з викликами з паралельної бібліотеки часу виконання.

У системі є аналізатор, який розпізнає редукції суми, добутку, мінімуму і максимуму. Це досить важке завдання – розпізнати редукції, які охоплюють кратні довільно вкладені гнізда циклів.

Паралелізатор виконує аналіз залежностей за даними, використовуючи SUIF-бібліотеку залежностей. Аналізатор залежностей базується на алгоритмі Майдана і складається з серії швидких точних тестів, кожен з яких застосовується в обмеженій області. Його останній тест – алгоритм виключення Моцкін-Фур'є, розширеного для вирішення цілочислових задач. Аналізатор також здатний обробляти деякі прості нелінійні доступи до масивів.

Відкрита распаралелююча система (ВРС, [9]) призначена для автоматичного розпаралелювання програм з процедурних мов програмування на паралельні комп'ютери. За допомогою ВРС можуть бути розроблені: распаралелюючі компілятори або конвертори в мови високого рівня, діалогові распаралелюючі системи, програми для вивчення паралельного програмування і т.д. Перетворення з бібліотеки ВРС можна комбінувати для різних архітектур. У ВРС автоматично визначаються цикли ParDO, тобто цикли, ітерації яких можуть виконуватися незалежно [7]. Автоматично визначаються цикли що конвеєризуються, автоматично розраховується синхронізація конвеєрних обчислень. Основними частинами ВРС є: внутрішнє уявлення, графові моделі програм, бібліотека оптимізуючих/розпаралелюючих перетворень програм, додаткові функції компіляції.

Внутрішнє уявлення (ВУ) – це структура даних, що зберігає повну інформацію про програму, а також бібліотека сервісних функцій, що полегшують виконання перетворень програм [8]. ВУ у ВРС є універсальним, тобто дозволяє зберігати програми на декількох процедурних мовах програмування: С, Фортран, Паскаль; слугує базисом для побудови, аналізу інформаційних залежностей і написання перетворень програм, при цьому схожі властивості процедурних мов уніфікуються, а специфічні зберігаються; розширюваним для підключення компіляторів переднього плану з нових мов програмування до ВРС; зручним для генерації машинного коду різних цільових архітектур.

В вузлах внутрішнього уявлення використовуються спеціальні позначки, які дозволяють передавати інформацію між різними частинами ВРС. Також до складу ВРС входять налагоджувальні засоби і засоби перевірки цілісності структури ВУ під час роботи програми.

ВУ Відкритої распаралелюючої системи є зв'язковою деревоподібною структурою і складається з чотирьох дерев: типів, ідентифікаторів, виразів і операторів. Кожне дерево ВУ зберігає інформацію про відповідні частини програми. ВУ спроектовано таким чином, що при перетворенні програми зберігається синтаксична і семантична цілісність.

ВРС дозволяє у діалоговому режимі проводити ланцюги перетворень, аналізувати можливості розпаралелювання програм, автоматично виводячи на екран графові моделі програм.

У Відкритій розпаралелюючій системі реалізовані і візуалізовані наступні графові моделі програм: граф інформаційних зв'язків, граф обчислень, гратчастий граф програми (РГА), керуючий граф програми та граф викликів підпрограм.

Перетворення у ВРС діляться на групи, залежно від типів фрагментів вихідного тексту, до яких вони застосовуються При виконанні перетворення перевіряються умови еквівалентності, засновані на графі інформаційних зв'язків або на гратчастому графі. У ВРС реалізовані змішані обчислення і стандартизація виразів, що дозволяє підвищити широту застосування перетворень. Відкрита розпаралелююча система може виконувати наступні перетворення: підстановка вперед, включаючи інтервальну підстановку з оцінкою похибки, винесення загальних підвиразів; розтягування скалярів, розщеплення вершин, перестановка операторів, канонізація циклів, розбиття циклів, злиття циклів, перестановка циклів, гніздування циклів, розрив ітерацій. ВРС містить перетворення, засновані на гратчастих графах, еквівалентність яких не може бути заснована на традиційному графі програмних залежностей:

розщеплення багатовимірних гнізд циклів у вигляді послідовності тісних гнізд; підстановка індексних змінних; експансія масивів, на основі розщеплення у вигляді послідовності тісних гнізд циклів.

3. Висновки

Всі існуючі на сьогоднішній день системи розпаралелювання мають ряд недоліків і відрізняються один від одного за деякими критеріями. До основних глобальних недоліків можна віднести не універсальність за вхідними даними, вхідними даними в своїй більшості слугують програми на таких мовах, як Сі та Фортран; обмеженість архітектур для яких створюється паралельний код; візуалізація; високий рівень програміста-оператора.

При візуалізації роботи програми та проведенні аналізу коду на можливість розпаралелювання, використовуються різноманітні графи, наприклад, граф інформаційних зв'язків, граф обчислень, гратчастий граф програми (РГА), керуючий граф програми та граф викликів підпрограм.

Таке розмаїття графів може викликати певні складнощі при роботі з ними. Для обробки графів в коді програми необхідно використовувати їх математичне представлення, матриці. При великій кількості графів буде велика кількість матриць, кожна з яких буде зберігати певну інформацію про алгоритм, який аналізується.

Більш доцільним може бути використання однієї матриці, в котрій буде зберігатися вся інформація необхідна для проведення розпаралелювання, для візуалізації вихідного коду програми доцільно було би використовувати мережі Петрі, які дозволять оцінити еквівалентність перетворення та перевірити вихідний код на відсутність dead lock'ів, зациклень та інших виключно-помилкових ситуацій.

Література

1. Евстигнеев, В.А. Анализ циклов: выбор кандидатов на распараллеливание [Текст] / В.А. Евстигнеев, И.Л. Мирзуитова - Новосибирск : Ин-т систем информатики им. А.П. Ершова СО РАН, 1999. - 48 с.; 20 см.
2. Черняев, А.П. Программные системы векторизации и распараллеливания Фортран-программ для некоторых векторно-конвейерных ЭВМ (обзор) [Текст] // Программирование. - 1991. - N 2. - С. 53-68.
3. Хьюзон, К. KAP/205: усовершенствованный векторизатор типа текст-текст для суперкомпьютера Cyber 205 [Текст]/ Хьюзон К., Мак Т., Дейвис Д. и др. // Векторизация программ: теория, методы, реализация: Сб. статей/ Под.ред. Г.Д.Чинина. - М.: Мир, 1991. - С. 217-235.
4. Мак, Т. KAP/ST-100: Фортран-транслятор для присоединенного процессора ST-100 [Текст] / Т. Мак, К. Хьюзон, Д. Дейвис и др. // Векторизация программ: теория, методы, реализация: Сб. статей/ Под.ред. Г.Д.Чинина. - М.: Мир, 1991. - С. 202-216.
5. Миура, К. СуперЭВМ фирмы Fujitsu: векторная система FACOM [Текст] / К. Миура // СуперЭВМ. Аппаратная и программная организация. - М.: Радио и связь, 1991. - С. 166-183.
6. Wilson, R.P. SUIF: An infrastructure for research on parallelizing and optimizing compilers [Текст] / R.P. Wilson, R.S. French, C.S. Wilson a.o. // SIGPLAN Not. - 1994. - Vol. 29, N 12 - P. 31-37.
7. Шульженко А.М. Автоматическое определение циклов ParDo в программе [Текст] / А.М. Шульженко // Известия вузов. Северокавказский регион. Естественные науки. Приложение 11'05. с. 77-88.
8. Петренко, В.В. Внутреннее представление OPC3 [Текст] / В.В. Петренко // Научный сервис в сети Интернет: технологии распределенных вычислений: Труды Всероссийской научной конференции. (19-24 сентября 2005 г., г. Новороссийск). - М.: Изд-во МГУ, 2005, с. 106-108.
9. Штейнберг Б.Я., Нис З. Я., Петренко В. В., Черданцев Д. Н., Штейнберг Р. Б., Шульженко А. М. Состояние и возможности Открытой распараллеливающей системы (лето 2006 г.) / Шестая международная конференция "Перспективы систем информатики". Рабочий семинар "Наукоемкое программное обеспечение". 28-29 июня 2006 г. Новосибирск, Академгородок, Россия. Информационный бюллетень [Електронний документ]. \www/ URL: http://ops.rsu.ru/download/works/ops20-06_nsk.pdf - 18.03.2012 Загл. с экрана.