УДК 004.4'242

ОПТИМИЗАЦИЯ ВРЕМЕННОЙ ЛОКАЛЬНОСТИ ДАННЫХ ПРИ АВТОМАТИЧЕСКОМ РАСПАРАЛЛЕЛИВАНИИ ЛИНЕЙНЫХ ПРОГРАММ

Лебедев А.С.

ФГБОУ ВПО «Рыбинский государственный авиационный технический университет» имени П.А. Соловьева, Рыбинск, Россия (152934, Рыбинск, ул. Пушкина, 53), e-mail: tementy@gmail.com

Рассматривается задача поиска расписания для линейной программы, оптимизирующего временную локальность данных, как задача многокритериального выбора в условиях полной определенности. Разработан метод, опирающийся на техники модели многогранников, и позволяющий минимизировать задержку использования данных для каждой информационной зависимости, исходя из системы предпочтений лица, принимающего решение (ЛПР) относительно качества оптимизации для каждой информационной зависимости. Система предпочтений ЛПР задается набором весовых коэффициентов для применения техники линейной свертки. Поиск оптимальной по Парето альтернативы расписания сводится к задаче линейного целочисленного программирования. Разработанный метод позволяет точнее осуществлять задание системы предпочтений ЛПР (в особенности для программ со слабой параметризацией или с ослабленной параметризацией при Just-In-Time компиляции) по сравнению с распространенным на момент исследования подходом, реализованным в компиляторе Pluto. Применение метода иллюстрируется примером распараллеливания алгоритма LU-разложения. Параллельный вариант алгоритма, полученный с помощью разработанного метода, показывает лучшее быстродействие по сравнению с результатом работы компилятора Pluto.

Ключевые слова: автоматическое распараллеливание, модель многогранников, локальность данных, многокритериальная оптимизация.

OPTIMIZING DATA REUSE LATENCIES DURING AUTOMATIC PARALLELIZATION OF LOOP NESTS WITH STATIC CONTROL FLOW

Lebedev A.S.

Soloviev Rybinsk State Aviation Technical University, Rybinsk, Russia (152934, Rybinsk, street Pushkina, 53), e-mail: tementy@gmail.com

The affine scheduling problem for linear programs is considered in context of data reuse latency optimization as a multi-objective optimization problem with no uncertainty. The method presented was developed to minimize reuse latency for each data dependence using polyhedron model techniques on the assumption of subjective preferences of the decision maker about optimization quality for every dependence. Subjective preferences of the decision maker are defined by weighting coefficients in linear form of scalarized multi-objective problem. Finding Pareto-optimal solution reduces to solving of integer linear programming problem. The method developed allows to specify subjective preferences of the decision maker more precisely (notably for weakly parameterized programs or for programs with artificially de-emphasized parameterization during Just-In-Time compilation) in contrast with modern and popular method implemented in Pluto compiler. Application of the method is illustrated with parallelization of LU-decomposition algorithm. Parallel version of the algorithm obtained by using presented method outperforms the result obtained by using Pluto compiler.

Keywords: automatic parallelizing, polyhedron model, data locality, multi-objective optimization.

С увеличением количества вычислительных элементов в современных процессорах вычислительные системы приобретают все большую степень параллелизма. Разработка вычислительного программного кода, способного эффективно задействовать возможности аппаратуры, всегда являлась сложной задачей. Сегодня одним из актуальных подходов решения данной задачи является автоматическое распараллеливание программ.

Большинство вычислительно емких научных и инженерных приложений тратят значительную часть времени исполнения на вложенности циклов, часто отвечающие

критериям линейности программ. Модель многогранников [8] предоставляет мощный математический аппарат, упрощающий анализ и преобразование таких программ с целью улучшения быстродействия путем повышения параллелизма и оптимизации локальности данных при вычислениях.

Методы модели многогранников широко используются в статической компиляции. Код, написанный на языке высокого уровня (чаще всего это C или FORTRAN), оптимизируется для параллельного выполнения и компилируется для целевой архитектуры.

Распараллеливание в модели многогранников осуществляется в пять этапов.

1. Анализ зависимостей операций по данным. Рассматриваются потоковые зависимости (чтение после записи), антизависимости (запись после чтения), зависимости через выход (запись после записи).
2. Поиск для каждой операции функции расписания, сохраняющей логический порядок операций в соответствии с зависимостями по данным.
3. Поиск для каждой операции функции размещения, ставящей в соответствие операции индекс виртуального процессора.
4. Разбиение индексного пространства виртуальных процессоров — сопоставление виртуальных процессоров физическим.
5. Генерация параллельной программы по алгоритму [1] в соответствии с пространственно-временным преобразованием, заданным функциями расписания и размещения, и разбиением.

Широко применяемыми на практике стали классические методы поиска расписаний, ориентированные на минимизацию задержки расписания [6, 7] (поиск расписаний с наибольшим параллелизмом), и современный подход [3], ориентированный на минимизацию задержки использования данных (поиск расписаний, обеспечивающих временную локальность данных для более эффективного использования кэшей процессоров).

Подход к оптимизации временной локальности данных, предложенный в работе [3] и реализованный в компиляторе Pluto, демонстрирует эффективность на практике [2]. Однако, метод, его реализующий, не лишен следующего конструктивного недостатка: поскольку задача сводится к поиску лексикографического минимума на многограннике [5], возникает вопрос о порядке следования переменных в целевом векторе, или, в терминах теории принятия решений, вопрос о предпочтениях ЛПР (лица, принимающего решение) относительно качества оптимизации для каждой зависимости по данным. Кроме того, в такой постановке задачи оптимизации невозможно задать одинаковый приоритет различным зависимостям, поскольку невозможно поставить две переменные на одну и ту же позицию в целевом векторе.

Метод, предложенный в настоящей работе, базируется на той же идее сокращения задержки использования данных для более эффективного использования кэшей процессоров, но предоставляет более гибкие средства задания предпочтений ЛПР, не исключая из рассмотрения оптимальные решения, достижимые по методу [3].

1. Базовые понятия модели многогранников

Модель многогранников рассматривает класс линейных программ и полагается на поиск расписаний в аффинном виде. Эти программы удовлетворяют следующим ограничениям:

- единственным типом исполнительного оператора может быть оператор присваивания, правая часть которого является арифметическим выражением;
- все повторяющиеся операции описываются только с помощью циклов DO языка FORTRAN (либо эквивалентными циклами for языка C); структура вложенности циклов может быть произвольной; шаги изменения параметров циклов всегда равны +1;
- допускается использование условных и безусловных операторов перехода, передающих управление «вниз» по тексту; не допускаются побочные выходы из циклов;
- все индексные выражения переменных, границы изменения параметров циклов и условия передачи управления задаются неоднородными формами, линейными по совокупности параметров циклов и внешних переменных программы; все коэффициенты линейных форм являются целыми числами;
- внешние переменные программы всегда целочисленные; конкретные значения внешних переменных известны только во время работы программы.

Обобщенный граф зависимостей — это ориентированный мультиграф, каждая вершина которого представляет множество соответственных операций (динамических экземпляров одной и той же инструкции). Каждая операция принадлежит только одной вершине. Ребро графа представляет временное ограничение на две операции, соответствующие начальной и конечной вершине (принцип причинности для информационно зависимых операций u и v: $u \to v \Rightarrow \theta(v) > \theta(u)$, где θ – логическое время). В обобщенном графе зависимостей могут быть петли и циклы. Каждая вершина помечается описанием соответствующих операций, а каждое ребро — описанием соответствующих отношений зависимости.

Каждой вершине X соответствует ее домен – многогранник D_X на множестве $\mathbf{Q}^{p_X}$, где p_X – размерность ее итерационного пространства (количество циклов в программе, включающих X). Каждая операция представляется как $\langle i, z; X \rangle$, где $i \in D_X$ – целочисленный вектор индекса итерации, z – целочисленный вектор внешних переменных программы.

Каждому ребру e, исходящему из вершины $\sigma(e) = X$ в $\delta(e) = Y$, ставится в соответствие многогранник R_e на множестве $\mathbf{Q}^{p_X + p_Y}$ такой, что условие причинности должно быть

наложено на динамические экземпляры операций $\langle i,z;X\rangle$ и $\langle j,z;Y\rangle$ тогда и только тогда, когда для составного вектора (i,j) выполняется $(i,j)\in R_e$.

Пусть f и g – индексные функции ячейки памяти, к которой обращаются конфликтующие операции, тогда p_e называют глубиной зависимости:

$$(x,y)\in R_e \equiv f(x)=g(y)\wedge(x[1..p_e]=y[1..p_e])\wedge(x[p_e+1]<y[p_e+1]).$$

Расписание для обобщенного графа зависимостей есть неотрицательная функция $\theta:\Omega\to\mathbf{N_0}$ такая, что $\forall e\in E: x\in D_{\sigma(e)}, y\in D_{\delta(e)}, (x,y)\in R_e \Rightarrow \theta(\delta(e),y)\geq\theta(\sigma(e),x)+1$.

2. Минимизация задержки использования данных в модели многогранников

Пусть обобщенный граф зависимостей анализируемой программы содержит m вершин S_i, $i=1,\ldots,m$ и n ребер e_j, $j=1,\ldots,n$. Требуется найти расписание θ, минимизирующее задержку использования данных для каждого ребра e_j, $j=1,\ldots,n$:

$$\max_{(x,y)\in R_{e_j},x\in D_{\sigma(e_j)},y\in D_{\delta(e_j)}}\left(\theta(\delta(e_j),y)-\theta(\sigma(e_j),x)\right)\to\min.$$

Требования приводят к следующей формулировке задачи многокритериальной оптимизации. Определим множество исходов (совпадающее со множеством альтернатив):

$$\Theta=\left\{\theta\left|\begin{array}{l}\left(\theta(S_i,x)=v_i^T x+v_i^0\geq 0, i=1,\ldots,m, x\in D_{S_i}\right)\wedge\\ \left(x\in D_{\sigma(e_j)}, y\in D_{\delta(e_j)}, (x,y)\in R_e\Rightarrow\theta(\delta(e_j),y)\geq\theta(\sigma(e_j),x)+1,\ j=1,\ldots,n\right)\end{array}\right.\right\}.$$

Здесь v_i – вектор-столбец из p_{S_i} элементов, v_i^0 – скаляр (значения принадлежат множеству $\mathbf{N_0}$), $\theta(S_i,x)$ – аффинная функция расписания для инструкции S_i ($x\in D_{S_i}$).

Показатели качества решения задаются n функциями $f_j:\Theta\to\mathbf{N}$, $j=1\ldots n$, которые требуется минимизировать: $f_j(\theta)=\max_{(x,y)\in R_{e_j},x\in D_{\sigma(e_j)},y\in D_{\delta(e_j)}}\left(\theta(\delta(e_j),y)-\theta(\sigma(e_j),x)\right)\to\min$.

Будем искать эффективный вектор $f=\left(f_1\ \ldots\ f_n\right)^T$ как оптимальное по Парето решение, применив технику линейной свертки: $\sum_{j=1}^{n}\alpha_j f_j\to\min, \sum_{j=1}^{n}\alpha_j=1, \alpha_j\geq 0, j=1,\ldots,n$, где α_j – весовые коэффициенты, отражающие предпочтения ЛПР относительно качества оптимизации для каждой зависимости по данным. Такая формулировка позволяет точнее отразить предпочтения ЛПР в силу большей гибкости относительно [3].

Для программ со слабой параметризацией, или в случае Just-In-Time компиляции, когда значения внешних переменных программы становятся известными, становится возможной приоритезация зависимостей на основе частоты возникновения ситуаций доступа к одной

ячейке памяти: $\alpha_j = \#R_{e_j} \Big/ \sum_{k=1}^{n} \#R_{e_k}$, где $\#R_{e_j}$ – количество точек с целочисленными координатами внутри многогранника зависимостей R_{e_j}, которое может быть вычислено с помощью алгоритма Клаусса [4], реализованного в библиотеке PolyLib [9]. Такая постановка реализует естественную эвристику: ЛПР заинтересовано в наилучшем качестве оптимизации для тех зависимостей, которые чаще возникают.

Для каждого ребра e_j наложим ограничения на функции расписания инструкций $\sigma(e_j)$ и $\delta(e_j)$, затем составим систему из всех этих ограничений:

$$\begin{cases} \theta(\delta(e_j), x_{\delta(e_j)}) - \theta(\sigma(e_j), x_{\sigma(e_j)}) - 1 \geq 0,\ j = 1, \ldots, n \\ \theta(\sigma(e_j), x_{\sigma(e_j)}) - \theta(\delta(e_j), x_{\delta(e_j)}) + f_j \geq 0,\ j = 1, \ldots, n \end{cases}.$$

Первое ограничение обусловлено принципом причинности, второе задает лимит задержки использования, который требуется минимизировать. Линеаризуем систему ограничений с помощью классической техники на основе леммы Фаркаша.

Лемма Фаркаша. Пусть D – непустой многогранник, определённый с помощью p аффинных неравенств (граней): $a_k x + b_k \geq 0,\ k = 1, \ldots, p$. Аффинная форма ψ неотрицательна в любой точке $x \in D$ тогда и только тогда, когда она является неотрицательной аффинной комбинацией: $\psi(x) \equiv \lambda_0 + \sum_{k=1}^{p} \lambda_k (a_k x + b_k),\ \lambda \geq 0$. Неотрицательные константы λ_k называются множителями Фаркаша.

Для каждой инструкции S_i выписывается шаблон функции расписания, неотрицательной внутри ее домена: $\theta(S_i, x_{S_i}) = \mu_{S_i,0} + \sum_{k=1}^{p_{D_{S_i}}} \mu_{S_i,k} (a_{S_i,k} x_{S_i} + b_{S_i,k})$, где $\mu_{S_i,k} \geq 0$ – множители Фаркаша, а $p_{D_{S_i}}$ неравенств $a_{S_i,k} x_{S_i} + b_{S_i,k} \geq 0$ определяют домен D_{S_i}, $i = 1, \ldots, m$.

Применим лемму Фаркаша для каждого ограничения:

$$\theta(\delta(e_j), x_{\delta(e_j)}) - \theta(\sigma(e_j), x_{\sigma(e_j)}) - 1 \equiv \lambda_{e_j,0} + \sum_{k=1}^{p_{R_{e_j}}} \lambda_{e_j,k} \left(c_{e_j,k} y_{e_j} + d_{e_j,k} \right)$$

$$\theta(\sigma(e_j), x_{\sigma(e_j)}) - \theta(\delta(e_j), x_{\delta(e_j)}) + f_j \equiv \lambda'_{e_j,0} + \sum_{k=1}^{p_{R_{e_j}}} \lambda'_{e_j,k} \left(c_{e_j,k} y_{e_j} + d_{e_j,k} \right)$$

Здесь $\lambda_{e_j,k} \geq 0$ и $\lambda'_{e_j,k} \geq 0$ – множители Фаркаша, а $p_{R_{e_j}}$ неравенств $c_{e_j,k} y_{e_j} + d_{e_j,k} \geq 0$ определяют многогранник зависимостей R_{e_j}, $y_{e_j} = \begin{pmatrix} x_{\sigma(e_j)} \\ x_{\delta(e_j)} \end{pmatrix}$ – составной вектор индексов для обеих инструкций $\sigma(e_j)$ в $\delta(e_j)$, $j = 1,\ldots,n$. В итоге система ограничений примет вид:

$$\begin{cases} \mu_{\delta(e_j),0} + \sum\limits_{k=1}^{p_{D_{\delta(e_j)}}} \mu_{\delta(e_j),k}(a_{\delta(e_j),k} x_{\delta(e_j)} + b_{\delta(e_j),k}) - \mu_{\sigma(e_j),0} - \sum\limits_{k=1}^{p_{D_{\sigma(e_j)}}} \mu_{\sigma(e_j),k}(a_{\sigma(e_j),k} x_{\sigma(e_j)} + b_{\sigma(e_j),k}) - 1 = \\ \lambda_{e_j,0} + \sum\limits_{k=1}^{p_{R_{e_j}}} \lambda_{e_j,k}\left(c_{e_j,k} y_{e_j} + d_{e_j,k}\right), j = 1,\ldots,n \\ \mu_{\sigma(e_j),0} + \sum\limits_{k=1}^{p_{D_{\sigma(e_j)}}} \mu_{\sigma(e_j),k}(a_{\sigma(e_j),k} x_{\sigma(e_j)} + b_{\sigma(e_j),k}) - \mu_{\delta(e_j),0} - \sum\limits_{k=1}^{p_{D_{\delta(e_j)}}} \mu_{\delta(e_j),k}(a_{\delta(e_j),k} x_{\delta(e_j)} + b_{\delta(e_j),k}) + f_j = \\ \lambda'_{e_j,0} + \sum\limits_{k=1}^{p_{R_{e_j}}} \lambda'_{e_j,k}\left(c_{e_j,k} y_{e_j} + d_{e_j,k}\right), j = 1,\ldots,n \\ y_{e_j} = \begin{pmatrix} x_{\sigma(e_j)} \\ x_{\delta(e_j)} \end{pmatrix}, j = 1,\ldots,n \\ \mu_{S_i,k} \geq 0, i = 1,\ldots,m; k = 0,\ldots,p_{D_{S_i}} \\ \lambda_{e_j,k} \geq 0, j = 1,\ldots,n; k = 0,\ldots,p_{R_{e_j}} \\ \lambda'_{e_j,k} \geq 0, j = 1,\ldots,n; k = 0,\ldots,p_{R_{e_j}} \end{cases}$$

Приравнивание соответственных множителей при компонентах векторов x_{S_i} в левых и правых частях ограничений позволяет оставить в системе только множители Фаркаша μ, λ, λ' и переменные-ограничители f.

Поиск Парето-оптимального решения задачи многокритериальной оптимизации сводится к решению задачи линейного целочисленного программирования с целевой функцией C, которую требуется минимизировать при сформированной системе ограничений (без участия компонентов векторов x_{S_i} в качестве переменных):

$C(\mu, \lambda, \lambda', f) = \sum\limits_{j=1}^{n} \beta_j f_j \to \min, \beta_j \geq 0, j = 1,\ldots,n$, где β_j – весовые коэффициенты, отражающие предпочтения ЛПР относительно качества оптимизации для каждой зависимости по данным. В случае слабой параметризации программы или Just-In-Time компиляции целесообразно взять $\beta_j = \alpha_j \sum\limits_{k=1}^{n} \# R_{e_k} = \# R_{e_j}$, что обеспечит больший приоритет зависимостям, которые чаще возникают.

Решение задачи ЛЦП может быть осуществлено, например, методом ветвей и границ, реализованным в библиотеке GNU Linear Programming Kit [10]. Подстановка значений μ в

соответствующие шаблоны функций расписания для каждой инструкции дает решение задачи поиска расписания θ.

3. Распараллеливание LU-разложения

Рассмотрим алгоритм LU-разложения с его информационными зависимостями (рисунок 1) и найдем два расписания, используя разработанный метод.

```
for (int k = 0; k < N; k++) {
  for (int j = k + 1; j < N; j++)
    A[j * N + k] /= A[k * N + k]; //S1
  for (int i = k + 1; i < N; i++)
    for (int j = k + 1; j < N; j++)
      A[i * N + j] -= A[i * N + k] * A[k * N + j]; //S2
}
```

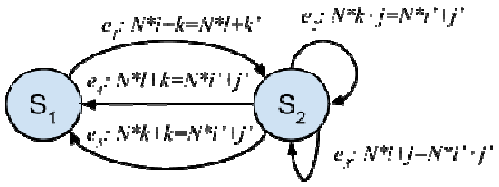

Рис. 1. Код LU-разложения на языке C и обобщенный граф зависимостей

Ослабим параметризацию программы, зафиксировав $N = 2^{10}$. Изменяя коэффициенты в целевой функции задачи ЛЦП можно управлять задержкой использования данных: например, для $C(\mu, \lambda, \lambda', f) = \sum_{j=1}^{5} \#R_{e_j} f_j$ функции расписания примут вид $\theta_{S_1} = 2k$ и $\theta_{S_2} = 2k+1$ (при этом $f = (1,2,2,1,1)^T$), а для $C(\mu, \lambda, \lambda', f) = f_1 + 10000 f_2 + 10000 f_3 + f_4 + f_5$ получаем $\theta_{S_1} = 2k$ и $\theta_{S_2} = k + j$ (при этом $f = (1023,1,1,1,1)^T$; именно такое расписание находит компилятор Pluto [2]).

Эксперименты на четырехъядерном процессоре Intel Core i7 930 показывают ускорение от распараллеливания в 3,96 раза для $\theta_{S_1} = 2k$ и $\theta_{S_2} = 2k+1$, и ускорение в 3,21 раза для $\theta_{S_1} = 2k$ и $\theta_{S_2} = k + j$, что свидетельствует об эффективности разработанного метода и целесообразности более точного задания системы предпочтений ЛПР.

Заключение

В ходе исследования была показана необходимость совершенствования способа задания системы предпочтений ЛПР для того, чтобы повысить качество оптимизации временной локальности данных при автоматическом распараллеливании линейных программ. Предложен метод поиска расписания, минимизирующего задержку использования данных для каждой информационной зависимости. Рассмотрение задачи поиска расписания как задачи многокритериального выбора в условиях полной определенности позволяет задать линейную свертку критериев качества, точнее отражающую систему предпочтений ЛПР и потому позволяет находить более удачные (в плане повышения быстродействия синтезированной параллельной программы) расписания, чем распространенный на момент

исследования метод [3], опирающийся на технику поиска лексикографического минимума на многограннике.

Работа выполнена в рамках проекта РФФИ №14-37-50316 «Исследование и разработка методов автоматического распараллеливания программ для гетерогенных вычислительных систем и систем с распределенной памятью» при финансовой поддержке Фонда.

Список литературы

1. Bastoul C. Code generation in the polyhedral model is easier than you think //Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques. – IEEE Computer Society, 2004. – С. 7-16.
2. Bondhugula U. et al. A practical automatic polyhedral parallelizer and locality optimizer //ACM SIGPLAN Notices. – 2008. – Т. 43. – №. 6. – С. 101-113.
3. Bondhugula U. et al. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model //Compiler Construction. – Springer Berlin Heidelberg, 2008. – С. 132-146.
4. Clauss P. Counting solutions to linear and nonlinear constraints through Ehrhart polynomials: Applications to analyze and transform scientific programs //Proceedings of the 10th international conference on Supercomputing. – ACM, 1996. – С. 278-285.
5. Feautrier P. Parametric integer programming //RAIRO Recherche opérationnelle. – 1988. – Т. 22. – №. 3. – С. 243-268.
6. Feautrier P. Some efficient solutions to the affine scheduling problem. I. One-dimensional time //International journal of parallel programming. – 1992. – Т. 21. – №. 5. – С. 313-347.
7. Feautrier P. Some efficient solutions to the affine scheduling problem. Part II. Multidimensional time //International journal of parallel programming. – 1992. – Т. 21. – №. 6. – С. 389-420.
8. Lengauer C. Loop parallelization in the polytope model //CONCUR'93. – Springer Berlin Heidelberg, 1993. – С. 398-416.
9. Loechner V. PolyLib: A library for manipulating parameterized polyhedra. – 1999. URL: http://icps.u-strasbg.fr/polylib/ (дата обращения: 26.11.14).
10. Makhorin A. GNU linear programming kit //Moscow Aviation Institute, Moscow, Russia. – 2012. URL: http://www.gnu.org/software/glpk (дата обращения: 26.11.14).

Рецензенты:

Хачумов В.М., д.т.н., профессор, зав. лабораторией интеллектуального управления, ФГБУН Институт системного анализа Российской академии наук, г. Москва;

Цирлин А.М., д.т.н., профессор, зам. директора по научной работе ФГБУН Институт программных систем им. А.К. Айламазяна Российской академии наук, Ярославская обл., с. Веськово.