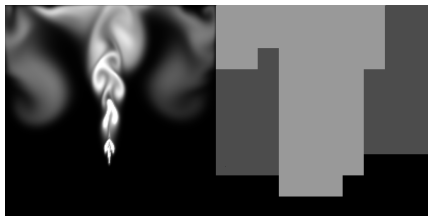
Refinamiento Adaptivo de Código

César Sabater



24 de enero de 2017

Presentación de Tesina

Introduccion

- Muchas aplicaciones de **cómputo intensivo** realizan **cálculos aproximados**
- Algunas Razones:
  - para simular objetos o fenomenos del mundo real
  - trabajan con sensores limitados en precision
  - estan sometidas a un deadline de tiempo
  - calculan resultados preliminares

Introduccion

Algunos ejemplos de estas aplicaciones son:

- Simulaciónes Físicas
- Sensores de entrada, algoritmos de procesamiento
- Decodificación de Video en tiempo real
- Predicción de Terremotos
- Aplicaciones de exploracion de geofisica

Introduccion

Normalmente, el desarrollo de estos algoritmos puede dividirse en **dos estapas**:

- Desarrollo de un kernel de computo ideal
  - ajuste del algoritmo
  - debugueo
- Optimizacion a una version de produccion
  - ajustando el nivel de precision para realizar calculos precisos solo donde es necesario
  - escalar al tamano real del problema
  - satisfacer un deadline

Introduccion

Optimizad un kernel ideal tiene complicaciones:

- es complejo
- lleva tiempo
- conduce a codigos menos mantenibles
- hay que hacerlo nuevamente cuando hay cambios grandes en la estrategia

Esto se podria reducir con un **enfoque automatico de compilacion**

Introduccion

Existen enfoques automaticos de optimizacion para computo intensivo:

- manteniendo la semantica original
  - paralelizacion
  - localidad de datos
  - vectorizacion

  bien abordados por los compiladores
- modificando la semantica original
  - relajacion de dependencias
  - modificiacion o eliminacion de calculos
  - usando conocimiento especifico de dominio

  escasamente abordados por los compiladores

Introduccion

Para aplicaciones que calculan resultados aproximados, un enfoque automatico de optimizacion que modifique la semantica original es adecuado siempre que asegure una **precision aceptable**.

Introduccion

Nuestro Objetivo: Busacar formas de optimizar codigos que computan aproximaciones

- de forma **automatica**
- **modificando la semantica** original
- asegurando una **precision aceptable**

Introduccion

Para realizarlo utilizaremos

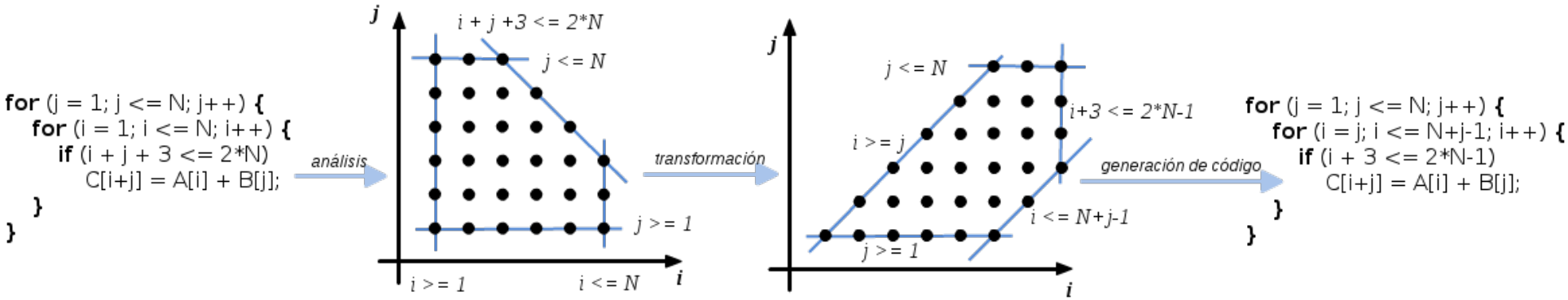
- El **Modelo Poliedrico** para realizar transformaciones agresivas de codigo
- **Conocimiento de Dominio Especifico** suministrado por el usuario
  - que guie la transformacion de codigo
  - mantenga una precision aceptable
- Un enfoque **adaptivo**
  - realiza calculos complejos solamente donde es necesario
  - ahorra computaciones que no impactan en el resultado
  - transforma el codigo de forma dinamica

Contenido

- **Transformacion de Codigo: Modelo Poliedrico**
- Herramienta estatica: Spot
- Refinamiento Adaptivo de Codigo
- Experimentos con Simulacion de Fluidos
- Conclusiones y Trabajo Futuro

Modelo Poliedrico

- Es un modelo matematico-computacional para la optimizacion y paralelizacion automatica de programas
- muy poderoso para la representacion y transformacion estructural de programas
- Representa bucles matematicamente con **relaciones afines**
  - dominios de iteracion
  - relaciones de orden
  - relaciones de acceso



Dominios de Iteracion

- **instancia de statement**: una ejecucion individual de un stamement S
- $S(i', j')$ es la instancia de S para los valores $i = i'$ y $j = j'$
- el **dominio de iteracion** queda definido por el conjunto de valores del vector $\binom{i}{j}$
- el dominio de iteracion se puede representar con un **poliedro**

```
     for (i = 0; i < 2 * N - 1; i++)
S1:      z[i] = 0;

     for (i = 0; i < N; i++)
          for (j = 0; j < N; j++)
S2:           z[i+j] = x[i] + y[j];
```

Figura: multiplicacion de matrices

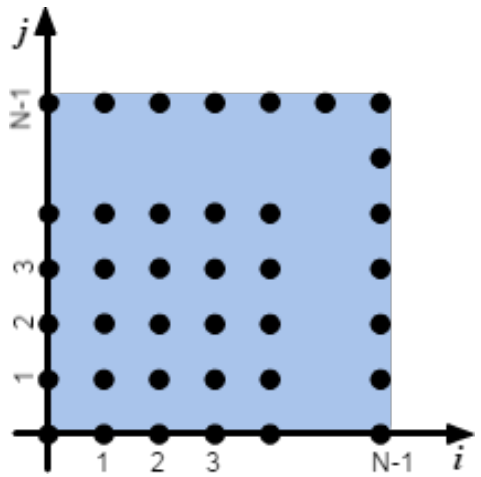


Figura: instancias de $S2$

Representacion poliedrica: Dominios de Iteracion

inecuaciones del dominio de iteracion de $S2$:

- $i \geq 0$
- $i < N$
- $j \geq 0$
- $j < N$

ecuacion matricial:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \\ N \\ 1 \end{pmatrix} \geq \vec{0}$$

- representacion poderosa y compacta
- restricciones:
  - no hay esquema general para soportar flujo de control dinamico
  - cotas de bucles y condicionales deben ser funciones afines de iteradores exteriores y parametros

Representacion poliedrica: Relaciones de Ordenamiento y Posicion

- especifican **orden y lugar** de cada instancia
- **orden**: posicion temporal de ejecucion con respecto otras instancias
- **lugar**: en que procesador deben ejecutarse
- a cada instancia se le asigna una **fecha logica**

$$\theta_{S_1}(N,i) = \begin{pmatrix} 0 \\ i \end{pmatrix}$$

$$\theta_{S_2}(N,i,j) = \begin{pmatrix} 1 \\ i \\ j \end{pmatrix}$$

```
     for (i = 0; i < 2 * N - 1; i++)
S1:      z[i] = 0;

     for (i = 0; i < N; i++)
         for (j = 0; j < N; j++)
S2:          z[i+j] = x[i] + y[j];
```

- los statements se ejecutan en orden lexicografico
- las funciones de ordenamiento deben ser afines

Representacion poliedrica: Relaciones de Ordenamiento y Posicion

relaciones afines:

$$\theta_{S_1}(N)=\left\{ (i)\to\begin{pmatrix} t_1 \\ t_2 \end{pmatrix}\in\mathbb{Z}\times\mathbb{Z}^2 \;\middle|\; \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \end{bmatrix}\begin{pmatrix} t_1 \\ t_2 \\ i \\ N \\ 1 \end{pmatrix}=\vec{0} \right\},$$

$$\theta_{S_2}(N)=\left\{ \begin{pmatrix} i \\ j \end{pmatrix}\to\begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}\in\mathbb{Z}^2\times\mathbb{Z}^3 \;\middle|\; \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix}\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ i \\ j \\ N \\ 1 \end{pmatrix}=\vec{0} \right\}$$

Representacion Poliedrica: Funciones de Acceso