Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Cálculo multiprocesado de SASA

Organización del Computador II: TP Final

2015

Adaptación de calculo sasa a multiprocesamiento MIMD

| Esteban Rey 657/10 estebanlucianorey@gmail.com |

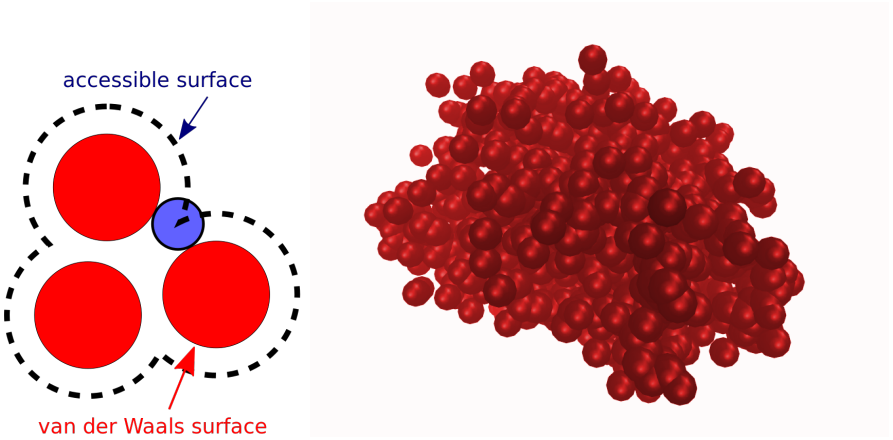
Índice

1. Objetivo

Basado en los resultados del benchmarking del anexo, en el cual se genera una fuerte mejora en el tiempo de ejecución de los algoritmos al pasarlos de programación SISD a MIMD, se busca reproducir la mejora en el calculo de la medición SASA de una molécula

2. Introducción: El calculo sasa de una molécula

Introducido por Lee y Richards en 1971 [1], el calculo del área accesible por un solvente (SASA en inglés) es normalmente calculado mediate el algoritmo propuesto por Shrake y Rupley en 1973 [3]. El mismo calcula la superficie mediante el rodamiento de una esfera de radio determinado (usualmente 1,4Å asemejando al radio de una molécula de agua) sobre la molécula objetivo y sumando la distancia recorrida.



(a) Las esferas rojas representan al atomo por su radio atómico, la linea punteada representa por donde la esfera de 1,4Å (azul) puede pasar, osea la medición SASA del conjunto de átomos

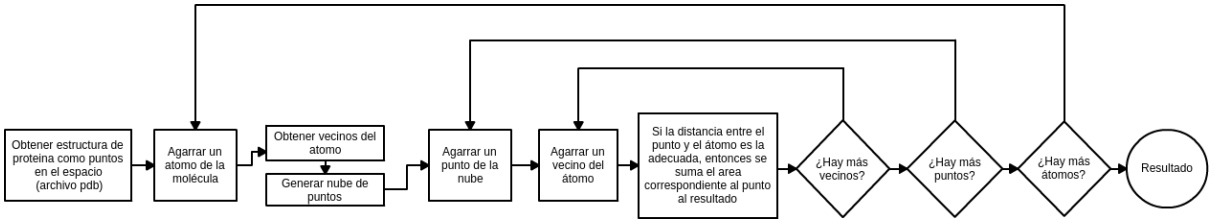
Desde otro punto de vista, lo que se hace es: dado cada atomo perteneciente a la molecula, se lo representa con una esfera del mismo radio que su radio atomico (superficie de Van Der Walls). A estos átomos se les incrementa el radio en 1.4 Å formando nuevas esferas. Si se suma la superficie de todas estas nuevas esferas y se le resta el area de las intersecciones entre ellas entonces obtenemos el calculo SASA de la molécula.

3. Implementación del algoritmo de Shrake y Rupley

Para discretizar el problema, la superficie de Van Der Walls de cada atomo se representa con una nube de puntos y la intersección con otro átomo se calcula mediante la distancia euclídea entre cada punto y los centros de los demás átomos. Si la distancia es menor al radio del otro átomo, más el tamaño de la molécula de agua (diámetro), entonces el punto se descarta de la suma.

Para generar la nube de puntos se intenta utilizar el algoritmo que genere la distribución más uniforme, para esto contamos con 3 distribuciones distintas:

- "Disco Ball"de Dave Russins aporta la distribución de puntos más densa, no obstante no se le puede especificar a priori la cantidad de puntos a utilizar en la discretización
- El método de Saff y Kuijlaars [2] el cual permite la parametrizacion de puntos, pero sacrifica la calidad de la distribución de los mismos.
- El método "Golden Section Spiral.[el] cual hace uso del numero áureo y logra una mejor distribución de puntos que el de Saff pudiendo determinarse la cantidad de puntos a usar. Este es el método usado en el trabajo.



(b) Algoritmo de SASA

El algoritmo presenta 2 niveles en los que se puede paralelizar: en una primera instancia, separar el calculo sasa de cada atomo en threads. La segunda instancia involucra el calculo sasa en cada átomo: en la etapa de seleccionar los puntos de la nube de puntos que deben considerarse para el area. Como hay que calcular la distancia de cada punto a cada atomo para decidir si sirve o no, podemos de alguna forma, paralelizar el cálculo.

3.1. Paralelización con SIMD

Para acotar la cantidad de átomos a considerar en las distancias, filtramos aquellos que consideremos 'vecinos' al átomo que estamos analizando, osea aquellos que se encuentren a una distancia menor o igual a la suma de los radios de los 2 átomos más el diámetro de la molécula de agua. Tomemos a p como las 3 coordenadas de uno de los n puntos de la nube de un átomo y a v como a las 3 coordenadas de uno de los m vecinos del átomo. Construimos entonces una matriz P de 3n x 3m floats: cada fila representara a cada punto de la nube, y estarán repetidas sus coordenadas m veces. Luego creamos una matriz V de iguales dimensiones en donde en cada fila estan los m vecinos y se repiten en las n filas:

$$\forall i \in [1,n], p_i = (x_i, y_i, z_i) \wedge \forall j \in [1,m], v_j = (x_j, y_j, z_j)$$

$$P = \begin{bmatrix} p_1 & p_1 & ... & p_1 \\ p_2 & p_2 & ... & p_2 \\ \vdots & \vdots & ... & \vdots \\ p_n & p_n & ... & p_n \end{bmatrix}; V = \begin{bmatrix} v_1 & v_2 & ... & v_m \\ v_1 & v_2 & ... & v_m \\ \vdots & \vdots & & \vdots \\ v_1 & v_2 & ... & v_m \end{bmatrix}$$

Elevando al cuadrado cada elemento de la matriz C = A - B obtenemos en cada elemento de la matriz la distancia al cuadrado de la distancia de todos los puntos con todos los vecinos (llamando C^2 a dicha matriz):

$$C^2 = \begin{bmatrix} (p_1 - v_1)^2 & (p_1 - v_2)^2 & ... & (p_1 - v_m)^2 \\ (p_2 - v_1)^2 & (p_2 - v_2)^2 & ... & (p_2 - v_m)^2 \\ \vdots & \vdots & & \vdots \\ (p_n - v_1)^2 & (p_n - v_2)^2 & ... & (p_n - v_m)^2 \end{bmatrix}$$

Si tomamos a C_x^2, C_y^2, C_z^2 como la división por coordenadas de C^2, cada una conteniendo n x m floats, y las sumamos, obtenemos en cada elemento de la matriz resultante, la distancia al cuadrado de cada punto contra cada vecino del átomo.

Solo resta evaluar que distancias se encuentran a una distancia mayor igual a 1,4 Å. Para ello le restamos una matriz de floats conteniendo en cada casillero dicha distancia y marcamos con un 0 los resultados negativos y con 1 a los positivos. De esta forma, si sumamos todos los elementos de la matriz resultante, obtendremos la cantidad de puntos de la nube que nos sirven para el cálculo. Finalmente se multiplica ese numero por un coeficiente relacionado con el radio atomico del atomo que estamos analizando, y se obtiene el sasa del mismo.

Esta forma de dividir los datos nos permite aplicar SIMD sin mayores problemas y solo requiriendo el set de instrucciones SSE.

4. Resultados

La aplicación de MIMD sobre el algoritmo resulto 3 veces más rápido que la versión no paralelizada, compiladas con el flag -O3 de gsc. El solo requerir compatibilidad con el set de instrucciones SSE nos permite el uso del algoritmo en maquinas no tan actualizadas (Pentium 3 en adelante)

5. Alcance de la mejora

La estructura del algoritmo original, incluido en la carpeta del tp, fue extraído de una biblioteca de funciones hechas en python. Al desarrollar el algoritmo SISD para utilizar de base para la versión MIMD se modificó la estructura del mismo en la forma explicada en la sección anterior. Este cambio solo, logro un speed up de AGREGAR SPEEDUP, gracias a el modo de escribir los ciclos para aprovechar el desenrollado de los mismos por parte del compilador, entre otras tantas optimizaciónes que se agregan al compilar con el flag -O3. A esta versión, y no a la original, fue a la que se le agregaron las instrucciones SSE. Se decidió calcular el speed up de MIMD frente a la versión modificada y no frente a la original para reflejar lo mejor posible el cambio de SISD a MIMD excluyendo las mejoras por reescritura de código.

6. Instalación y ejecución

Para ejecutar el proyecto, solo basta con ejecutar el archivo installNrun.sh, el script instalará las dependencias necesarias: De no funcionar se deben seguir los siguientes puntos:

- Descargar (o usar los provistos en el comprimido) e instalar, la biblioteca de python Biopython desde http://biopython.org/wiki/Download,notar que como dependencia tiene a la biblioteca numpy.
- Compilar asa.asm con la siguiente instrucción: nasm -felf64 -g -F dwarf asa.asm
- Instalar proyecto con la instrucción:sudo python mimd-setup.py install

Para poder correr la captura de tiempos del algoritmo SISD versus MIMD, en donde se encuentra un ejemplo de aplicación, se debe hacer lo siguiente:

- Instalar el plugin correspondiente al código de asa calculado con SISD: sudo python c-setup.py install
- Asegurar la presencia del archivo ¡CARPETA TP¿/datasource/proteina.pdb
- Ejecutar el tp: python tp.py

Referencias

[1] B. Lee and F. Richards. The interpretation of protein structures: Estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379 – IN4, 1971.

[2] E. Saff and A. Kuijlaars. Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11, 1997.

[3] A. Shrake and J. Rupley. Environment and exposure to solvent of protein atoms. lysozyme and insulin. *Journal of Molecular Biology*, 79(2):351 – 371, 1973.