Rev. Int. Mét. Num. Cálc. Dis. Ing.
Vol. **25, 4,** 299-312 (2009)

Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Optimización de código para un simulador de estructuras aporticadas y su implementación como una herramienta productiva

Richard G. Espinoza, Julio Flórez-López

Facultad de Ingeniería, Centro de Investigación en Matemáticas Aplicadas (CIMA)
Universidad de Los Andes, Av. Alberto Carnevalli, Núcleo La Hechicera, Mérida-Venezuela
Tel.: 58 274 2401111 ext. 3715; Fax: 58 274 2402971
email: richardg@ula.ve; iflorez@ula.ve

Germán Larrazábal

Centro Multidisciplinario de Visualización y Cómputo Científico (CeMViCC)
Facultad de Ciencias y Tecnología (FACYT), Universidad de Carabobo (UC)
Av. Montes de Oca, N° 120-267, Edificio FACYT, Valencia-Venezuela
Tel./Fax: 58 241 6005000 ext. 315196
email: glarraza@uc.edu.ve

Resumen

En este trabajo se presenta la optimización del programa ***procesador***, utilizado por la herramienta **Portal de pórticos** (http://portaldeporticos.ula.ve). Este es un programa de elementos finitos escrito en FORTRAN-90, basado en la **Teoría del Daño Concentrado**. Este programa permite el análisis y la simulación numérica de estructuras de concreto armado, que son sometidas a sismos u otro tipo de solicitaciones. Con la finalidad de disminuir el tiempo de cómputo del análisis de la estructura, se acopló la biblioteca ***UCSparseLib*** (***U****niversity of* ***C****arabobo* ***Sparse Lib****rary*), y para hacer aun más eficiente, mejorar el rendimiento de esta herramienta de cálculo se implementa un manejador de colas en el cluster donde esta alojado el portal, el manejador se diseña mediante **Simulación de Eventos Discretos**. Los resultados experimentales muestran una reducción del tiempo del tiempo de cómputo del análisis en más del 80 % para los casos de pruebas. Por último se realiza una implementación utilizando ***OpenMP***, en un computador de memoria compartida.

Palabras clave: *Elementos finitos, resolución de sistemas de ecuaciones lineales, OpenMP, Simulación de Eventos Discretos.*

CODE OPTIMIZATION OF A FRAMED STRUCTURES SIMULATOR AND ITS IMPLEMENTATION AS A PRODUCTIVE TOOL

Summary

In this paper the optimization of the program "Processor" is presented. The "processor" is a finite element program used by the tool "Portal of Damage" (http://portaldeporticos.ula.ve), coded in Fortran-90, and based in the lumped damage mechanics. This program allows the simulation and modeling of the inelastic behavior and damage in reinforced concrete structures under static or dynamic loads. To decrease the computing time of the structure analysis, the library ***UCSparceLib*** (University of Carabobo Sparse) was included, and to make even more efficient, to improve the performance of the Processor is implemented a queue manager in the cluster where the Portal is lodged. The queue manager is designed under discrete events simulation paradigm. The experimental results show a reduction in the computing time of more than 80 % for the examples. Finally, an implementation using ***OpenMP*** is done in a shared memory computer.

Keywords: *Finite elements, lineal equation resolution system, OpenMP, simulation of discrete events.*

 ISSN: 0213–1315 Recibido: Nov. 2008 Aceptado: Abril 2009

INTRODUCCIÓN

En los últimos años se ha extendido ampliamente, el uso de programas computacionales en los procesos de análisis y diseño en ingeniería. Particularmente en ingeniería estructural, los programas de análisis cubren un campo de aplicaciones que va desde las estructuras aporticadas, con muros, hasta la inclusión de los pisos o coberturas laminares que pueden ser modelados con elementos finitos apropiados.

El portal de pórticos[1,2,3] es una de estas herramientas para el análisis y la simulación de estructuras aporticadas, este portal es un programa de elementos finitos basado en la Teoría del Daño Concentrado[4,5,6,7], esta teoría combina: La Mecánica de la fractura, Mecánica del Daño y el concepto de Rótula Plástica. Esta teoría es usada para modelar la conducta de pórticos de concreto armado. El programa usado en el portal de pórticos (*procesador*), es un programa secuencial escrito en FORTRAN-90, que se encuentra alojado en el centro de cómputo de alto rendimiento (CeCalCULA) en un cluster de pentium IV, el acceso a esta herramienta se efectúa através de la dirección Web http://portaldeporticos.ula.ve

Es bien sabido que el núcleo computacional que consume mayor cantidad de tiempo de CPU en los paquetes de simulación numérica en ingeniería es el módulo que resuelve los sistemas de ecuaciones lineales. En este caso particular, al hacer un análisis del tiempo de CPU utilizado en resolver el sistema de ecuaciones lineales se observó que en el mejor de los casos, es el 70 % del tiempo total de cómputo del análisis de la estructura, se observa adicionalmente que la matriz de coeficientes que se genera al aplicar el método de los elementos finitos, tiene una gran cantidad de elementos nulos, lo que indica que puede ser considerada una matriz tipo *sparse*.

En virtud de estas consideraciones, en este trabajo se estudió e implementó en el programa ***procesador*** la biblioteca ***UCSparseLib*** (***U****niversity of* ***C****arabobo* ***Sparse Lib****rary*)[8], cuya estructura de datos para el almacenamiento y manipulación de matrices sparse (dispersas) está basada en el formato compacto de almacenamiento ***CRS*** (***C****ompressed* ***R****ow* ***S****torage*)[9,10,11] el cual ha demostrado ser eficiente en las aplicaciones donde ha sido utilizado.

Después de realizar la implementación de la biblioteca ***UCSparseLib*** se estudió la posibilidad de aprovechar la arquitectura de cluster. La razón por la que no se utilizó la programación paralela en computadores de memoria distribuída es motivado a que la comunicación entre los diferentes procesos consume un gran porcentaje del tiempo total de cómputo del análisis, para resolver este problema se diseño mediante ***Simulación de Eventos Discretos*** (*SED*)[12], un manejador de cola y se implementó mediante el software ***SGE*** (Sun Grid Engine)[13].

Por último, y con la finalidad de contar con una herramienta que pueda hacer uso de las diferentes arquitecturas de computadoras, se estudió la forma de aplicar computación paralela en máquinas de memoria compartida usando una división del pórtico por pisos con el fin de intentar disminuir aun más el tiempo de cómputo del análisis, con este fin se implementó esta idea mediante el software ***OpenMP***[14,15].

MODELO DE DAÑO CONCENTRADO PARA PÓRTICOS DE CONCRETO ARMADO

En el modelo de daño concentrado se estudian las **Deformaciones Generalizadas** y los **Esfuerzos Generalizados** de un pórtico plano (Figura 1a) que están dados por $\Phi^t = (\phi_i, \phi_j, \delta)$ y $\mathbf{M}^t = (m_i, m_j, n)$ respectivamente, donde ϕ_i, ϕ_j son **las rotaciones** en los extremos i y j del miembro, δ es **la deformación axial**, así como m_i, m_j son **los momentos flectores** y n es l**a fuerza axial** (Figuras 1b, 1c). Los principales fenómenos que ocurren en un pórtico de concreto armado son: *la cedencia del refuerzo* y *el agrietamiento del concreto*, para estudiar estos fenómenos se adopta el modelo que considera que todos

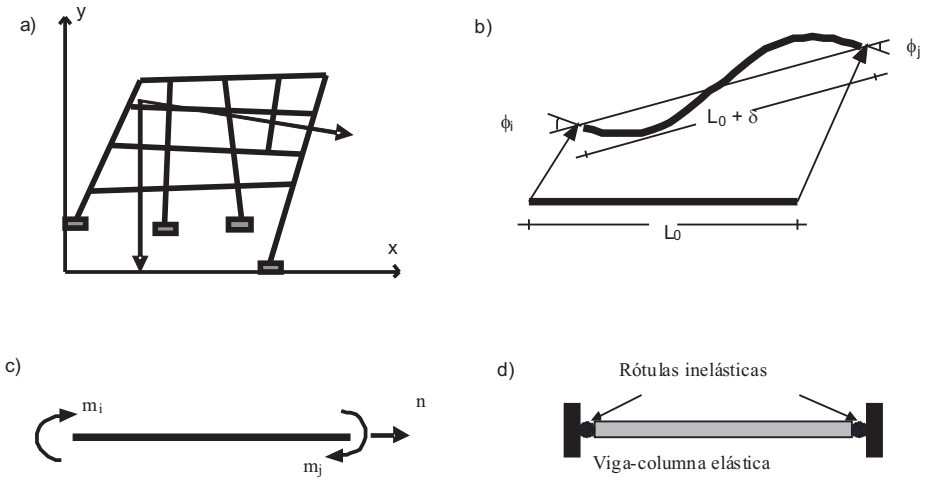

Figura 1. Variables en la teoría del daño

los fenómenos inelásticos se concentran en las rótulas plásticas o inelásticas de un miembro formado por el ensamblaje de una viga-columna elástica y dos rotulas en los extremos[4] (Figura 1d).

La teoría de pórticos elasto-plásticos se desarrolla introduciendo una variable interna que denota las deformaciones plásticas generalizadas $(\Phi^p)^t = \left(\phi_i^p, \phi_j^p, 0\right)$ donde ϕ_i^p y ϕ_j^p son las **rotaciones plásticas** de las rótulas en los extremos i y j y la mecánica del daño concentrado por la introducción de parámetros internos $D = (d_i, d_j)$ llamados daño, que están relacionados con las rótulas plásticas (Figura 1d), () y () son variables que miden la intensidad de las micro-grietas y pueden tomar valores entre 0 y 1, es decir, $0 \leq d_i \leq 1$ y $0 \leq d_j \leq 1$.

Con estas consideraciones el modelo de daño concentrado usado para simular pórticos de concreto armado[4,5,6,7] se describen usando el conjunto de ecuaciones que se muestran a continuación:

- Ley de estado.

$$\{\Phi - \Phi^p\} = [F(D)]\, M = \begin{bmatrix} \frac{F_{11}^0}{(1-d_i)} & F_{12}^0 & 0 \\ F_{21}^0 & \frac{F_{22}^0}{1-d_j} & 0 \\ 0 & 0 & F_3^0 \end{bmatrix} M \tag{1}$$

donde $[F(D)]$ es la **matriz de flexibilidad** de un elemento de concreto armado agrietado

- Leyes de evolución del daño.

Energía de Deformación (W^*) y Tasa de Disipación de energía (G)

$$W^* = \frac{1}{2}\mathbf{M}^t\{\Phi - \Phi^p\} = \tfrac{1}{2}\mathbf{M}^t\left[F\left(D\right)\right]\mathbf{M} \tag{2}$$

$$G^t = (G_i, G_j) = \left(\frac{\partial W^*}{\partial d_i}, \frac{\partial W^*}{\partial d_j}\right) = \left(\frac{m_i^2 F_{11}^0}{2(1-d_i)^2}, \frac{m_j^2 F_{22}^0}{2(1-d_j)^2}\right) \tag{3}$$

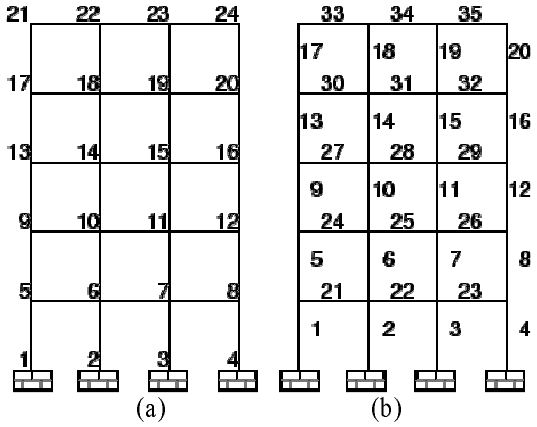

Figura 2. Ejemplo de numeración; a) de los nodos y b) de los elementos

donde G_i y G_j representan las tasas de disipación de energía para las rótulas i y j respectivamente.

Función de resistencia al agrietamiento (R) y Función de Fluencia de una rótula plástica con daño y endurecimiento (f_i).

$$R(d_i) = G_{cr} + q\frac{\log(1-d_i)}{1-d_i} \quad y \quad f_i = \left|\frac{m_i}{1-d_i} - c\phi_i^p\right| - m_y \leq 0 \tag{4}$$

donde G_{cr} y q son parámetros que dependen de las propiedades del material, mientras que c y m_y son propiedades que dependen del elemento.

El criterio de Griffth para la rótula i es:

$$\begin{cases} \dot{d}_i = 0 & si \quad G_i < R(d_i) \quad o \quad \dot{G}_i < \dot{R}(d_i) \\ \dot{d}_i > 0 & si \quad G_i = R(d_i) \quad y \quad G_i = \dot{R}(d_i) \end{cases} \tag{5}$$

Donde $\dot{d}_i$ representa el incremento de daño en la rótula i y $R(d_i)$ es su función de resistencia.

ANÁLISIS DEL PÓRTICO

Cuando se desea realizar el análisis de un pórtico de n-nodos y m-elementos (Figura 2), se debe resolver un sistema de ecuaciones diferenciales ordinarias no lineales de la forma

$$\sum_{b=1}^{m} [B]_b^t \{M(U)\}_b + [m]\left\{\ddot{U}\right\} - \{P\} = 0 \tag{6}$$

donde $[m]$ es la matriz de masa del pórtico y $[B]$ es la matriz de transformaciones.

Así usando una discretización del tiempo de simulación [0,T] en un conjunto $(0, t_1, t_2, \cdots, T)$ y una aproximación para $\{\ddot{U}\} \cong \left\{\ddot{U}\left(U^r, U^{r-1}\right)\right\}$ por algún método numérico, el sistema de ecuaciones se puede expresar como $\{L(U^r)\} = \sum_{b=1}^{m} [B]^t \{M(U)\}_b + [m]\left\{\ddot{U}(U^r, U^{r-1})\right\} - \{P^r\} = 0$, esto implica que para encontrar la solución en el instante de tiempo t_r se debe conocer la solución en el instante t_{r-1}. Dado que $\{L(U)\}$=0 es un sistema no lineal se usa el método de Newton para resolverlo. Así $\left\{L(U_{k+1}^r)\right\} \cong \{L(U_k^r)\} + \left[\frac{\partial L}{\partial U}\right]_{\{U\}=\{U_k^r\}} \left(\{U_{k+1}^r\} - \{U_k^r\}\right) = 0$ y para determinar los desplazamientos en el instante de tiempo t_r y en la

iteración k+1, $\{U^r_{k+1}\}$ será necesario resolver el sistema de ecuaciones lineales

$$\{L(U^r_k)\} + \left[\frac{\partial L}{\partial U}\right]_{\{U\}=\{U^r_k\}} \{\Delta U^r_{k+1}\} = 0 \quad \text{y} \quad \{U^r_{k+1}\} = \{U^r_k\} + \{\Delta U^r_{k+1}\} \tag{7}$$

para ello se determinarán $\{L(U^r_k)\}$ y $\left[\frac{\partial L}{\partial U}\right]_{\{U\}=\{U^r_k\}}$ que son llamadas matrices globales. Sin embargo, para encontrar estas matrices, será necesario resolver los m problemas locales que consisten en determinar las matrices locales $\{M(U^r_k)\}$ y $\left[\frac{\partial M}{\partial U}\right]_{\{U\}=\{U^r_k\}}$ para cada elemento de la estructura y así poder ensamblar las matrices globales

$$\sum_{b=1}^{m} [B]^t \{M(U^r_k)\} \quad y \quad \sum_{b=1}^{m} \left([B]^t \left[\frac{\partial M}{\partial U}\right]_{\{U\}=\{U^r_k\}}\right) \tag{8}$$

y por consiguiente resolver el sistema

$$\{L(U^r_k)\} + \left[\frac{\partial L}{\partial U}\right]_{\{U\}=\{U^r_k\}} \{\Delta U^r_{k+1}\} = 0 \tag{9}$$

Este es un sistema de ecuaciones lineales de la forma $Ax{=}b$ cuya matriz de coeficientes tiene dimensiones $3n{\times}3n$, puesto que se estudian 3 desplazamientos por nodo y n representa el número de nodos de la estructura.

El programa *procesador* utilizado en el portal de pórticos utiliza el análisis de las ecuaciones descritas anteriormente, el modelo de la teoría del daño concentrado, y el método de elementos finitos, en un programa secuencial desarrollado en FORTRAN-90. Este programa tiene implementado un método directo para resolver el sistema de ecuaciones lineales generado a través de la discretización utilizada.

OPTIMIZACIÓN DEL CÓDIGO

Para realizar la optimización, primero se usó la herramienta ***gprof*** para identificar las zonas del código que consumen mayor cantidad de tiempo.

En la Tabla I se muestra una salida típica de **gprof**, en la cual queda en evidencia que la subrutina rfssedpglobal (encargada de resolver el sistema global de ecuaciones lineales) es la que consume el mayor tiempo en la simulación y en segundo lugar la rutina rfssedp encargada de resolver los m sistemas locales de ecuaciones lineales en cada instante de tiempo, siendo m el número de elementos del pórtico.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ks/call  Ks/call  name
 73.19   5205.42  5205.42    21288     0.00     0.00  rfssedpglobal_
  8.17   5786.31   580.89        1     0.58     6.39  MAIN__
  5.55   6181.30   394.99                             for_cpstr
  2.68   6371.83   190.53 19412062     0.00     0.00  rfssedp_
  1.53   6480.95   109.12                             cvtas_a_to_t
  1.13   6561.63    80.68 293369500    0.00     0.00  cal_dano_
  0.82   6620.27    58.63 388110574    0.00     0.00  datosdefinidos_mp_truncar_
  0.63   6665.40    45.13 146684750    0.00     0.00  cal_f_
  0.51   6701.69    36.28  4470548     0.00     0.00  pro_3mat_
  0.43   6731.99    30.30 22170040     0.00     0.00  pro_2mat_
```

Tabla I. Salida del ***gprof*** para un pórtico de treinta pisos

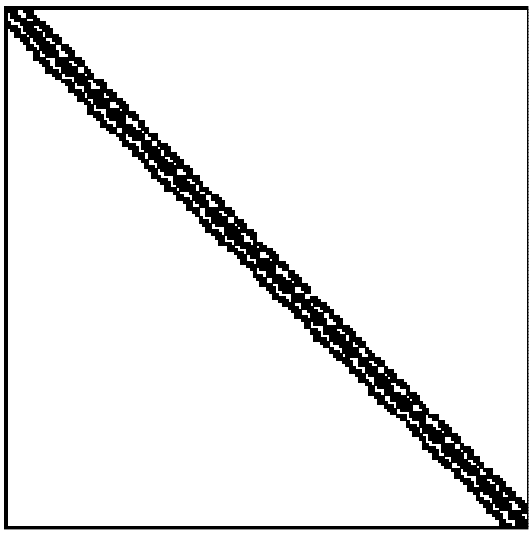

Figura 3. Forma de la matriz de coeficientes en el sistema $Ax=b$

Basados en la información dada por **gprof** se analizaron matrices de muestra en la simulación para medir el tiempo que tarda en resolverse el sistema $Ax=b$ en una iteración, si se usa una u otra subrutina, es decir, *UCSparseLib* y la utilizada en *procesador* la que llamaremos *normal*.

Los resultados obtenidos de este análisis muestran que la rutina que está implementada (normal) tarda 4.0 seg. en resolver el sistema aproximadamente, mientras que *UCSparseLib* tarda aproximadamente 0,01 seg.

En cuanto a la forma de la matriz, en la Figura 3, se muestra ésta para un pórtico de 30 pisos y cuyo tamaño es de 372x372 con un total de elementos no nulos de 1960 lo que representa aproximadamente un 2 % de entradas total y puede ser considerada una matriz tipo *sparse*.

Puesto que estos resultados son favorables para el uso de la biblioteca *UCSparseLib* se decidió implementar ésta al programa *procesador*.

ACOPLAMIENTO DE UCSparseLib AL PROGRAMA PROCESADOR

Para acoplar la biblioteca *UCSparseLib*[7] al programa *procesador* fue necesario resolver dos problemas, a saber:

1. El formato de almacenamiento que usa UCSparseLib, se basa en un formato compacto similar al ***CRS*** (***C****ompressed* ***R****ow* ***S****torage*) mientras que el formato usado en el *procesador* es un formato completo, es decir se almacena toda la matriz incluyendo los ceros.

2. El otro problema es el hecho que *UCSparseLib* está escrita en ANSI C y por tanto se creó un módulo para poder enlazar ésta con el programa *procesador* que está escrito en FORTRAN-90.

Para resolver estos inconvenientes, se estudió el formato de almacenamiento ***CRS*** (***C****ompressed* ***R****ow* ***S****torage*).

Este formato consiste en almacenar la matriz A de n filas por n columnas, en tres vectores (AN,JA,IA), donde AN es el vector de las entradas no nulas de A, JA es el vector que almacena el índice de la columna en la que se encuentra cada elemento no nulo e IA es el vector que indica el índice en AN del elemento que comienza cada nueva fila (Figura 4).

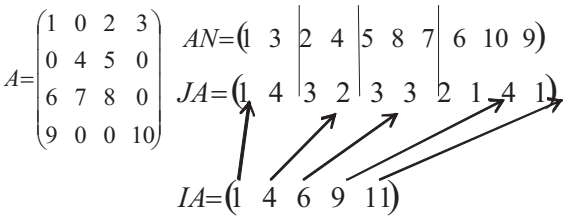

Figura 4. Formato CRS para una matriz 4x4 con 10 elementos no-nulos

Este formato se implementó en el programa *procesador* mediante la creación de una subrutina en FORTRAN-90 llamada *formatoCRS* y que toma como entrada una matriz A, la dimensión de esta y construye los vectores AN, AJ e IA del formato CRS, así como también cuenta la cantidad de entradas no nulas (nz) de la matriz A.

Para poder acoplar UCSparseLib con el programa *procesador* se creó un módulo en la biblioteca UCSparseLib, llamado portico.c que hace posible esta integración.

IMPLEMENTACIÓN DE UN MANEJADOR DE COLAS

Originalmente el programa *procesador* fue diseñado para trabajar en una sola máquina. Esto significa que todas las solicitudes que llegan através de la página Web deben compartir el procesador simultáneamente y ello provoca que se degrade el rendimiento, tanto del cálculo de las soluciones como de los otros servicios que este nodo debe mantener.

Con la finalidad de hacer al portal una herramienta productiva y aprovechar la arquitectura de cluster donde se encuentra alojado, se diseña un manejador de colas eficiente mediante el uso de la **simulación de eventos discretos** (SED)[12], con este fin se desarrollo un modelo del sistema considerando tanto el nodo principal como los nodos de cálculo, y los diferentes regímenes de llegada de las solicitudes al portal. Es de señalar que en el modelo se considera que el nodo principal no realiza cálculos.

Modelo del sistema

En todo estudio de simulación de eventos discretos. El primer paso consiste en diseñar un modelo conceptual del sistema que describa totalmente el flujo de las diversas entidades, así como los procesos a los que éstas son sometidas.

Modelo conceptual

Las solicitudes llegan al sistema según una tasa variable de llegadas (determinada mediante análisis estadístico); de allí pasan a formar parte de una cola FIFO (**First In**, **First Out)** :primero en entrar, primero en salir, administrada por el manejador de colas. Las solicitudes entonces salen de la cola para ser atendidas por el primer nodo desocupado; si hay más de uno desocupado se asignaran a aquel cuya carga total acumulada sea menor. El tiempo de proceso de cada solicitud en los nodos está dado por una distribución exponencial obtenida también del análisis estadístico; una vez terminado el proceso de la solicitud esta sale del cluster.

Análisis Estadístico

Para realizar el análisis estadístico se tomaron registros de las solicitudes que llagaban al portal durante 12 semanas (84 días continuos) lo que dio lugar a 1036 observaciones.

Se realizó el análisis de la distribución de las llegadas de las solicitudes, obteniendo como resultado una distribución como sigue:

1. Para el comportamiento de los fines de semanas la distribución de llegadas es una distribución discreta, para los otros días de la semana la distribución es uniforme tomando en cuenta que la máxima llegada de trabajos en promedio se encuentra en 50 y la mínima en 1.

2. Las horas del día también se toman en cuenta en el modelo considerando que la mayor cantidad de trabajos llegan entre las 10 a.m. y 12 m.

El análisis de los tiempos de procesamiento se realiza de manera similar, obteniendo en este caso una única distribución de tipo exponencial[16] con media $\lambda = 0{,}00327$ mín /*solicitud*.

Es necesario señalar que el valor tan bajo del parámetro de la exponencial se debe a que en el periodo de observación un 50 % de las solicitudes requirió menos de 1 minuto de procesamiento.

Modelo Computacional

Una vez determinados los parámetros del modelo, el mismo se implementa en un programa usando el software de Simulació de Eventos Discretos Arena Versión 5.0[17]. El modelo consta de 4 bloques (Figura 5).

En el primero que a su vez es un submodelo, se implementa toda la lógica correspondiente a la llegada de las solicitudes ver Figura 6 en el segundo bloque se le asigna a cada solicitud un atributo que indica su tiempo efectivo de procesamiento usando para ello la distribución exponencial descrita anteriormente.

El tercer bloque modela el cluster y su cola asociada, para ello se construyó un conjunto de 4 recursos, donde cada uno de los recursos es un nodo de procesamiento. Por ultimo el bloque final se encarga de recoger las estadísticas y desechar las entidades.

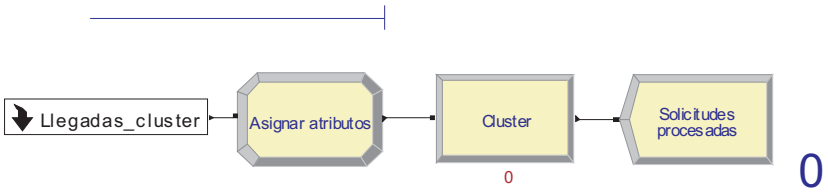

Figura 5. Modelo computacional

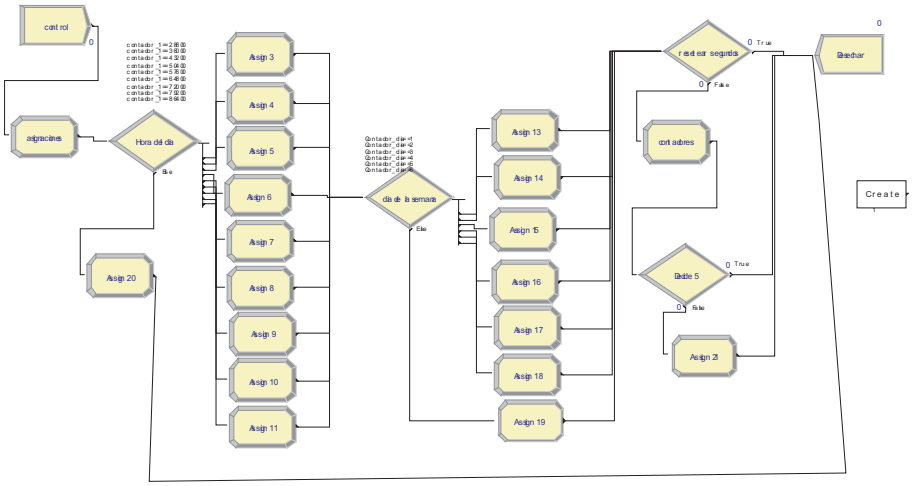

Figura 6. Sub-modelo de las llegadas al cluster

Verificación y validación

La verificación en Simulación de Eventos Discretos. Significa que el modelo no tenga errores lógicos o de programación y la validación consiste en observar que los resultados promedios de las simulaciones coinciden estadísticamente con la realidad.

Para la verificación se realizaron pruebas de "Caja Negra" donde se introdujo un número pequeño de solicitudes y se observó a la salida la misma cantidad procesada. También se realizaron pruebas de "Caja Blanca" cambiando algún parámetro estocástico del modelo por un valor fijo (determinista) y se observó que el comportamiento global del modelo reflejaba dicho cambio.

Para la validación del modelo se simuló el mismo durante 12 semanas y se tomaron 30 réplicas.

Los resultados indican valores promedios con intervalos de confianza al 95 % menores que el 10 % de la media, esto indica que los resultados son estadísticamente válidos.

La implementación de este modelo de manejador de cola en el cluster se hace mediante el software ***SGE*** (Sun Grid Engine) con dos nodos en funcionamiento y se comparan los resultados con los de este modelo, observando que coinciden estadísticamente con los resultados reales.

IMPLEMENTACIÓN PARALELA USANDO OpenMP

Una vez terminada la primera etapa en donde se acopla *UCSparseLib* al programa y se logra una optimización del tiempo de procesamiento en un 80 %, se plantea como continuación optimizar el tiempo que consume la rutina *rfssedp*. Puesto que los sistemas de ecuaciones que resuelve esta rutina son 3x3, no se piensa en implementar *UCSparseLib* sino aprovechar la computación paralela para atacar este problema haciendo una división del pórtico por pisos y usando *OpenMP*[11,12]. El esquema utilizado para esta paralelización se muestra en la Figura 7, utilizando las directivas necesarias en el lazo encargado de la distribución de los elementos del pórtico y acoplamiento a la matriz global del sistema.

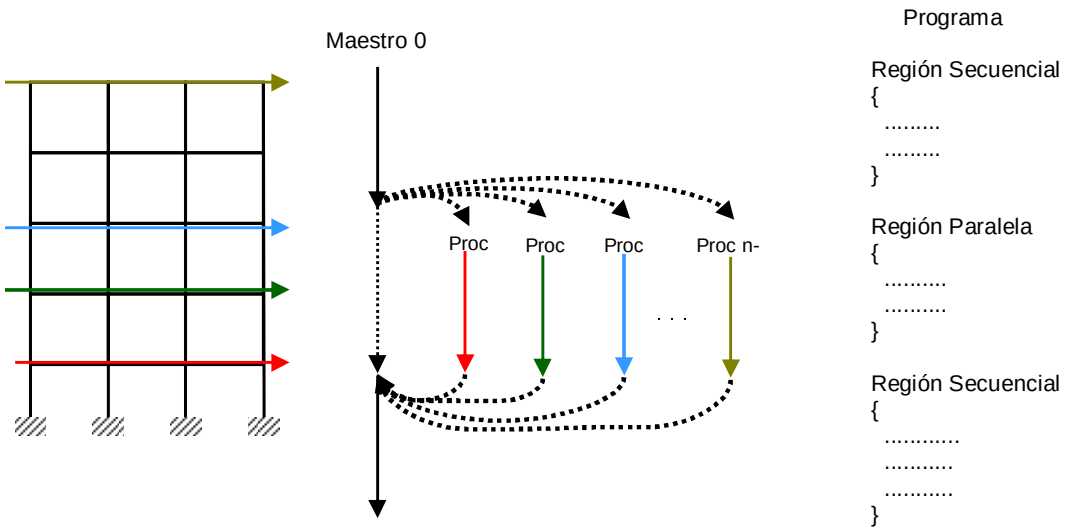

Figura 7. Esquema usado en la paralelización

RESULTADOS NUMÉRICOS

La rutina desarrollada se integró al programa *procesador* y se verificó su aplicabilidad usando algunos pórticos de concreto armado diseñados según las normas venezolanas, cada uno de estos ejemplos se introduce al programa *procesador* mediante un archivo con extensión .inp en el cual se describen las características del concreto, del refuerzo, las dimensiones de las vigas y columnas, la numeración de nodos y elementos, etc. La descripción más relevante de estos ejemplos desde el punto de vista de este trabajo se muestra en la Tabla II.

El número de pisos y el número de tramos nos da el número de nodos de la estructura por ejemplo, un pórtico de 5 pisos y 3 tramos tiene 24 nodos (Figura 2a) y el número de nodos nos da el tamaño de la matriz. Otro dato relevante es el sismo, el llamado elcents.amp es el registro de un sismo ocurrido en la ciudad de California, los otros son sismos sintéticos generados con los espectros de respuestas. Por último tenemos la duración de cada uno de los sismos.

No.	Nombre del Archivo	No. De Pisos	No. de Tramos	No. de Nodos	Tamaño de la Matriz	Nombre del Sismo	D. Sismo (seg)
1	Pórtico2	7	5	48	144x144	elcentns.amp	29
2	PórticoAyF	7	2	24	72x72	elcentns.amp	29
3	A2111	8	3	36	108x108	z3s2gb2.amp	29
4	Por_10	10	3	44	132x132	z2s2ga.amp	29
5	Por_12	12	3	52	156x156	z7s2gb2.amp	29
6	Por12_último	12	3	52	156x156	z7s2gb2.amp	29
7	A4111	16	3	68	204x204	z7s2gb2.amp	29
8	A4311	16	3	68	204x204	z3s2gb2.amp	29
9	Por_30z3	30	3	124	372x372	z3s2gb2.amp	29
10	Por_30cen	30	3	124	372x372	elcentns.amp	58

Tabla II. Descripción de los pórticos usados como ejemplos

		Normal		UCSpaseLib	
No.	Nombre	T. Total	% Ax=b	T. Total	% Ax=b
1	Pórtico2	9,07	17,07	7,91	1,06
2	PórticoAyF	3,71	4,57	3,70	1,04
3	A2111	6,44	9,01	6,40	1,13
4	Por_10	12,37	23,64	9,86	1,43
5	Por_12	17,57	20,84	15,40	1,21
6	Por12_último	17,73	20,65	15,35	1,21
7	A4111	24,48	35,87	17,23	1,42
8	A4311	14,05	35,80	9,69	1,49
9	Por_30z3	85,48	84,56	16,49	1,59
10	Por_30cen	173,10	85,49	28,76	1,77

Tabla III. Resultados obtenidos usando la rutina *normal* y al implementar *UCSparseLib*

Estos ejemplos se corrieron en el cluster donde está alojado el programa *procesador*, la solución en cuanto a los resultados del análisis usando una rutina u otra no difieren significativamente. Por consiguiente se compararon otros parámetros del análisis: el tiempo total de computo del análisis (**T. Total**) en cada uno de los casos, y el porcentaje de tiempo de CPU gastado en resolver el sistema de ecuaciones lineales (**% Ax=b**) para el caso donde se uso la rutina *normal* y cuando se cambió por *UCSparseLib* esto en cada uno de los ejemplos descritos en la Tabla II, los resultados obtenidos se muestran en la Tabla III. En esta tabla se observa cómo el porcentaje del tiempo empleado en resolver el sistema $Ax=b$ aumenta a medida que aumenta el tamaño de la matriz.

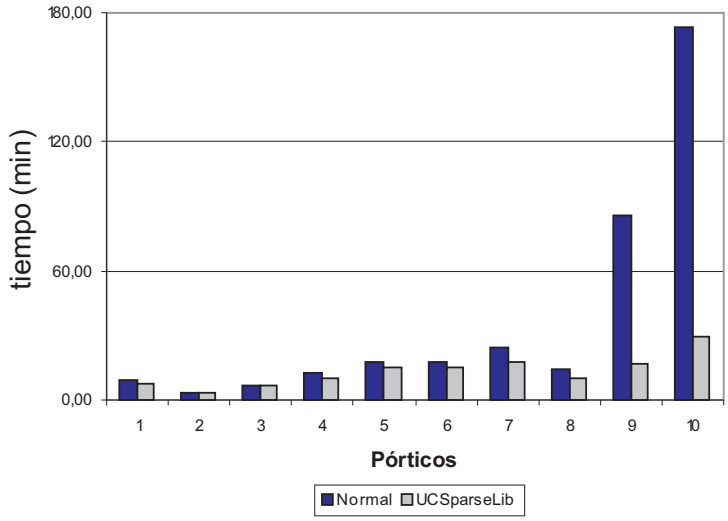

Figura 8. Comparación en el tiempo de CPU utilizado

En la Tabla III se observa cómo al usar la subrutina *normal* el porcentaje del tiempo empleado en resolver el sistema $Ax=b$ aumenta a medida que aumenta el tamaño de la matriz mientras que al usar *UCSparseLib* se aprecia cómo el porcentaje de tiempo usado en resolver el sistema de ecuaciones $Ax=b$ no presenta ese aumento en relación con el tamaño de la matriz.

En la Figura 8, se hace una comparación del tiempo total de la simulación utilizado en cada uno de los ejemplos y se observa cómo este disminuye considerablemente al hacer uso de la biblioteca *UCSparseLib*.

En cuanto a la implementación del manejador de cola los resultados obtenidos son los siguientes:

1. El modelo computacional se implementó en el software de simulación arena 5.0, considerando el cluster formado por 2 nodos de cálculo y al realizar una simulación por doce semanasde tiempo simulado se obtienen los siguientes resultados:

Número de solicitudes recibidas: 1086

El software muestra los resultados por promedio semanal, así que tenemos:

Un nodo es usado en promedio 52.333 veces a la semana y el otro 38 veces, sin embargo al observar el tiempo de uso de los nodos se tiene que un nodo es usado en el 56 % del tiempo y el otro en 44 %. Se presenta un máximo de cola de 9 trabajos sin embargo el tiempo máximo de espera es de menos de 3 segundos (esto motivado a que el mayor numero de solicitudes dura menos de un minuto).

2 La estrategia de manejo de colas diseñado en el modelo también se implemento en el cluster ***beauvoir.cecalc.ula.ve*** usando 2 nodos y considerando la misma cantidad de semanas de tiempo real. Se realizó el seguimiento con la finalidad de verificar y validar el modelo computacional. Los resultados estadísticos son los siguientes:

Número de solicitudes recibidas: 1025

Un nodo es usado en promedio 44.66 veces a la semana y el otro 40.75 veces, sin embargo al observar el tiempo de uso de los nodos se tiene que un nodo es usado en el 51 % del tiempo y el otro en 49 %. Se presenta un máximo de cola de 4 trabajos.

Los resultados reales coinciden estadísticamente con los de la simulación, esto permite pensar que la estrategia de manejo de colas es óptima.

Por último, con la finalidad de evitar la comunicación entre los procesos y poder determinar así si la paralelización del código mejora aun más los tiempos de computo del análisis, se realizó una prueba usando un computador de memoria compartida se corrieron todos los ejemplos en una *Sun Enterprice450* (4 cpu de 450 MHz y 4 GB de memoria ram). Los resultados se muestran en la Tabla IV.

Cabe resaltar que los valores en cuanto al tiempo total de la simulación cambian de una máquina a otra, sin embargo lo que importa aquí es la comparación entre las diferentes estrategias.

		Normal		UCSpaseLib		+OpenMp
No.	Nombre	T. Total	% Ax=b	T. Total	% Ax=b	T. Total
1	Pórtico2	24,66	20,17	21,49	1,58	18,62
2	PórticoAyF	10,09	7,68	10,07	1,04	8,95
3	A2111	17,49	12,46	17,39	1,09	15,43
4	Por_10	36,34	25,64	26,80	1,40	23,04
5	Por_12	47,77	22,44	41,86	1,22	36,61
6	Por12_último	48,19	23,05	41,72	1,21	36,68
7	A4111	66,54	38,87	46,84	1,40	39,82
8	A4311	38,19	37,50	26,33	1,39	22,36
9	Por_30z3	232,37	85,12	44,82	1,63	37,44
10	Por_30cen	470,54	85,84	78,17	1,72	64,98

Tabla IV. Resultados obtenidos usando la rutina *normal* y al implementar *UCSparseLib*

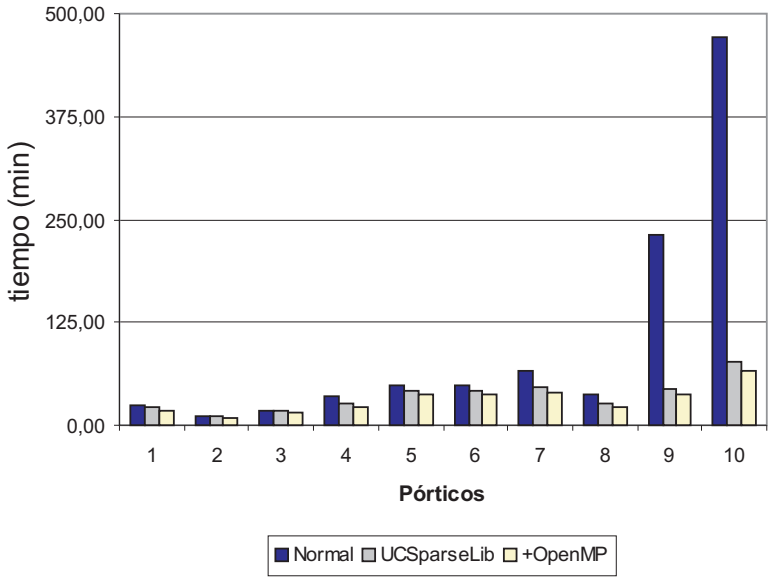

Figura 9. Comparación en el tiempo total utilizado por la simulación

En la Figura 9, se hace una comparación del tiempo total de la simulación utilizado en cada uno de los ejemplos y se observa cómo este disminuye considerablemente al hacer uso de la biblioteca *UCSparseLib* y luego al hacer uso de *OpenMP*, se nota que hay una nueva disminución del tiempo aunque en menor porcentaje.

CONCLUSIONES

En las gráficas y tablas mostradas se puede apreciar que el uso de la biblioteca *UCSparseLib* trae como principal ventaja la optimización del tiempo de la simulación, que en algunos casos logra una disminución de hasta el 80 % del tiempo antes de la implementación, esto hace que el uso del portal http://portaldeporticos.ula.ve para realizar el análisis y las simulaciones de estructuras aporticadas usando la teoría del daño concentrado sea más accesible a los usuarios del mismo.

Puesto que los resultados obtenidos al utilizar el manejador de colas coinciden estadísticamente con los de la simulación, esto permite pensar que la estrategia de manejo de colas es óptima, así tenemos que al unir el manejador de colas en el cluster para enviar los trabajos secuenciales y la mejora que proporciona *UCSparseLib*, se obtiene como consecuencias el uso optimo de los recursos del cluster, así como la descarga de trabajo del nodo principal con el consiguiente aumento de su eficiencia y, lo más importante, la disminución del tiempo total de cómputo que cada análisis requiere para su ejecución.

Otro resultado interesante es el hecho de que al usar *UCSparseLib+OpenMP* se disminuye aun más el tiempo de cómputo del análisis. Es claro que al hacer uso de *OpenMP* el programa sólo se puede ejecutar en ambientes de memoria compartida, sin embargo la tendencia es tener cluster formados por nodos con multi-procesadores con lo que se podría aprovechar todo lo desarrollado en este trabajo.

Como trabajo futuro, se plantea adaptar el modelo en Arena a diversas situaciones tales como: Caída de algunos nodos, Llegadas de trabajos en otros horarios o con otras distribuciones de tiempo, la duración de procesamiento, número de nodos dentro del cluster, etc.; y de esta forma se podrían simular estos escenarios y predecir el comportamiento del cluster ante los mismos, además con estas simulaciones se podrian establecer entre otras cosas el número de nodos que debe tener el cluster para que la espera por procesamiento sea la menor posible, así como tambien cambiar la política de la cola.

REFERENCIAS

1 M.E. Marante, J. Redondo, B. Vera, M. Uzcátegui, J. Flórez López, A. Quero, L. Suárez, "*Portal de pórticos: Herramienta computacional para el análisis de estructuras aporticadas basadas en la teoría del daño concentrado*", ISBN: 980-6745-00-0. (2004)

2 R. Espinoza, J. Flórez-López, N. Jaramillo, M.E. Marante, A. Quero y L. Suarez "Ruptura of framed structures, lumped damage mechanics and Internet" *Latin American Journal of Solid and Structures*, Vol. **2**, pp. 29–39, (2005).

3 M.E. Marante, B. Vera, M. Uzcátegui, M. Puglisi, L. Núñez, J. Flórez López, "Un portal de cálculo para el evaluación de la vulnerabilidad sísmica por elementos finitos de estructuras aporticada", *Revis. Inter. Mét. Num. Cálc. Dis. Ing.*, Vol. **24**, N° 4, pp. 299–321, (2008).

4 J. Flórez López, "Simplified model of unilateral damage for RC frames", *J. Struc. Eng.*, ASCE, Vol. **121**, N° 12, pp. 1765–1772, December (1995).

5 J. Flórez López, "Frame analysis an continum damage mechanics", *Eur. J. Mech.,/Solids*, Vol. **17**, N° 2, pp. 269–283, (1998).

6 J. Flórez López, "Plasticidad y fractura en estructuras aporticadas", Monografías en Ingeniería Sísmica CIMNEIS, N° 35, (1999).

7 A. Cipollina y J. Flórez López, "Modelos simplificados de daño en pórticos de concreto armado", *Revis. Inter. Mét. Num. Cálc. Dis. Ing.*, Vol. **11**, N° 3, pp. 3–22. (1995).

8 G. Larrazábal, "UCsparseLib: Una biblioteca numérica para resolver sistemas lineales dispersos", Simulación Numérica y Modelado Computacional, SVMNI. pp. TC19-TC25, ISBN: 980-6745-00-0, (2004).

9 E. Montagne, A. Ekambaram, "An alternative compressed storage format for sparse matrices", Technical Report, Computer Sciences Department, University of Central Florida, (2003).

10 J. Dongarra, "*Templates for the Solution of Algebraic Eigenvalue Problems :A Practical Guide*", SIAM, Philadelphia, (2000).

11 G. Larrazábal, "Técnicas algebraicas de precondicionamiento para la resolución de sistemas lineales", PhD Thesis, Departamento de Arquitectura de Computadores (DAC), Universidad Politécnica de Catalunya (UPC), Barcelona, España. ISBN: 84-688-1572-1, (2002).

12 A.M. Law, W.D. Kelton. "*Simulation Modeling and Analysis, 3/e*", McGraw Hill. ISBN: 0070592926, (2000).

13 N1 Grid Engine 6 Collection, "*N1 Grid Engine 6 User's Guide*", http://docs.sun.com, (2005).

14 Forte Developer 7 Collection "Forte Developer 7: OpenMP API User's Guide". http://docs.sun.com. (2002).

15 OpenMP Version 2.5 Specification Released, "*OpenMP Application Program Interface*", http://www.openmp.org, May (2005).

16 J.E. Freund, R.E. Walpole, "*Estadística matemática con aplicaciones*", Prentice Hall Hispanoamericana, (1990).

17 W.D. Kelton, R.P. Sadowski, D.T. Sturrock, "*Simulation with Arena*", McGraw Hill Professional, ISBN: 0072919817, (2003).