NetTopos: una plataforma de cálculo distribuido diseñada para el Simulador de Sistemas de Energía Eléctrica SimSEE

Pablo Alfaro, Ruben Chaer, *Member, IEEE*

***Abstract*-- El objetivo del desarrollo de NetTopos fue brindar una plataforma de comunicación para coordinar la ejecución de procesos en distintas computadoras de una red. En particular poder distribuir el cálculo de las simulaciones de sistemas de energía eléctrica en una red de computadoras. Como consignas del diseño, se impuso que fuera de uso sencillo, eficiente en la comunicación entre los procesos y el poder correr en cualquier equipo con sistema operativo Windows o Linux.**

***Index Terms*-- Distributed computing, Power system simulation.**

I. Introducción

UNA simulación con SimSEE involucra dos etapas importantes candidatas a ser distribuidas. La primera etapa es la de Optimización, en la que utilizando un algoritmo de Programación Dinámica Estocástica se calcula lo que es la "Política de Uso" de los recursos del sistema. La segunda etapa es la de Simulación que se lleva a cabo usando la política de uso calculada durante la optimización.

La Simulación es básicamente un proceso que se ejecuta en forma lineal, paso a paso realizando sorteos para simular diferentes crónicas de las diferentes fuentes de incertidumbre. Mediante la simulación de muchas crónicas se obtienen resultados en forma estadística. Cada crónica puede ser simulada en forma independiente de las demás por lo que la distribución de esta etapa resulta sencilla. Es el mismo proceso que se corre en los diferentes nodos de NetTopos cada uno con una semilla aleatoria de inicio diferente. Cada fin de crónica de simulación cada proceso informa al nodo que lanzó la ejecución que dispone del cálculo de una crónica más. De esta forma, cuando la suma de las crónicas ya ejecutadas en el conjunto de nodos es superior o igual al número de crónicas especificadas, se detiene el cálculo. Las diferentes velocidades de cálculo de los nodos queda así automáticamente regulada y cada uno correrá la cantidad de crónicas que pueda.

La etapa de Optimización es más compleja de distribuir dado que no es posible desacoplar el cálculo por crónicas como en la simulación. En la Programación Dinámica Estocástica el cálculo se realiza por pasos comenzando por el último paso de tiempo y realizando para cada paso de tiempo varios barridos del espacio de estado (discretizado). En un mismo paso de tiempo se realizan múltiples barridos pues con cada sorteo asociado a la simulación de crónicas de los datos aleatorios se realiza un barrido entero y se acumulan los resultados para promediar. Al terminar con el barrido de un paso, se pasa al anterior (se comienza por el ultimo) que repite el proceso y utiliza los valores del paso siguiente. Distribuir la simulación por el lado de que en cada nodo se resuelvan las etapas con diferentes sorteos implicaría una mayor comunicación pues en cada paso se debería comunicar un espacio de estados completo por sorteo. Entonces, la distribución del cálculo se realiza particionando la discretización del espacio de estados, asignando a cada nodo la resolución de la función de costo futuro de un sector del espacio.

El particionado del espacio y la distribución del cálculo se deben realizar en cada paso de tiempo y se debe aportar a cada nodo la información correspondiente a los cálculos realizados por el resto de los nodos en la etapa de cálculo previa que necesitan para la solución de la etapa actual. En la distribución del cálculo a todos los nodos se les da la misma semilla aleatoria, para que todos calculen con las mismas crónicas de aportes, roturas de máquinas, etc. A diferencia de la simulación, en la Optimización, en cada paso cada nodo debe barrer el sector del espacio de estado que le corresponde calcular para cada crónica y luego promediar los valores antes de informar los resultados de su cálculo.

II. Descripción de la Plataforma NetTopos

La plataforma NetTopos consiste en una serie de nodos, conectados entre sí a través de una red y con capacidad de comunicarse para poder coordinar su trabajo. Su arquitectura está dividida en dos capas: la capa de comunicación y la capa de aplicación.

La primera capa, de comunicación, está compuesta por los Topos, procesos que actúan como colas de mensajes de un nodo, "escuchando" mensajes de aplicaciones remotas ejecutando en otros nodos y comunicándolos a la aplicación

Este trabajo fue realizado en el marco del convenio de cooperación ADME-Facultad de Ingeniería. R. Chaer está en ADME y Facultad de Ingeniería (rch@adme.com.uy). P. Alfaro está en ADME (palfaro@adme.com.uy)

local correspondiente, y una API de comunicación cuyas principales funciones son las siguientes:

registrarAplicacion: provee un identificador de aplicación único para el nodo en que este ejecutando. Los identificadores de aplicación globales para la red quedan compuestos por este identificador y el identificador del nodo (IP) en el que está ejecutando la aplicación.

comunicar: envía un comunicado a otra aplicación de la red. Este tipo de comunicado no espera a que la aplicación procese el mensaje. Al verificar que el mensaje fue entregado retorna el control a la aplicación de origen. Es útil para iniciar la ejecución de varias tareas en distintos nodos a la vez.

comunicarTareaSincrona: envía un comunicado a otra aplicación de la red y detiene a la aplicación origen hasta que la aplicación destino comunique su respuesta. Es útil para realizar tareas cortas y que necesiten una respuesta instantánea como preguntar el estado de un nodo.

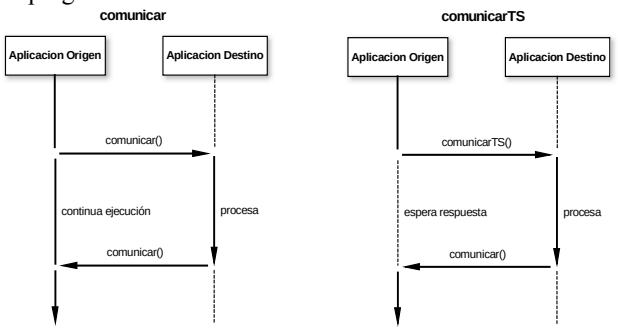

Fig. 1. Flujo de ejecución en los distintos métodos de comunicación

Sobre esta capa de comunicación se añade la capa de aplicación. En esta capa se encuentran las aplicaciones proveyendo servicios. Un ejemplo de aplicación es FileManager. Está aplicación permite crear y escribir archivos o abrirlos y leerlos, todo remotamente.

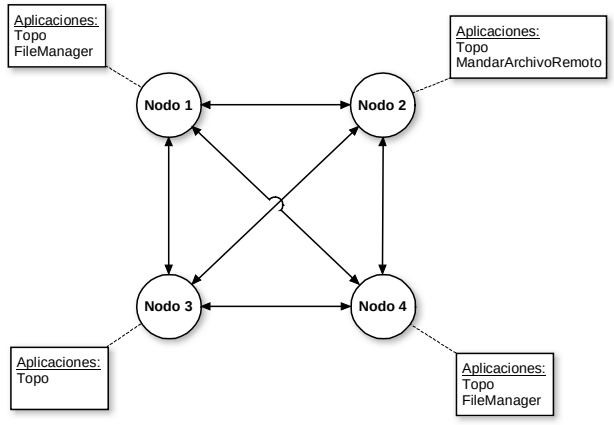

Fig. 2. Esquema de una red NetTopos

En muchas implementaciones típicas de cálculo distribuido una aplicación controla la ejecución mientras que varias aplicaciones distribuidas en la red realizan el cálculo en sí. Este será el caso de la optimización distribuida, en la que habrá una aplicación central, el Orquestador y varias aplicaciones de cálculo, los OptDis. La comunicación en nuestro caso será únicamente entre el Orquestador y los distintos OptDis. Los OptDis no se comunicaran entre sí.

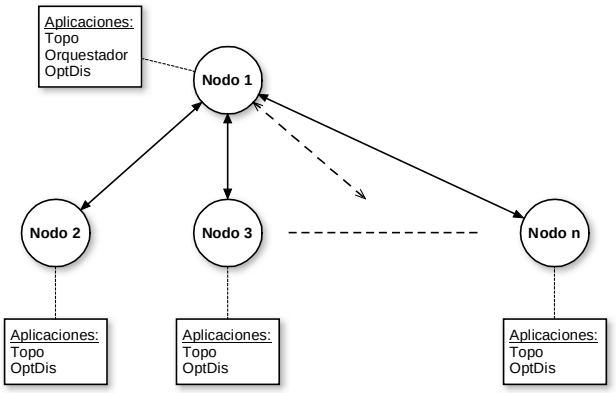

Fig. 3. Esquema de una red OptDis

III. Simulación de un Sistema de Energía Eléctrica

Esta sección pretende introducir una noción de los problemas de cálculo a los que uno se enfrenta al realizar la simulación de un sistema de energía eléctrica con el simulador SimSEE. Si el lector desea ver un estudio completo del problema puede referirse a la Tesis de Maestría del Ing. Rubén Chaer[1].

El optimizador del SimSEE busca obtener una política de operación del sistema que minimice el costo esperado. Definamos las siguientes variables:

x es el estado de nuestro sistema (por ejemplo la cantidad de agua en un embalse)

u_k son las entradas de control (por ejemplo turbinar o no turbinar agua en una central)

r_k son las entradas aleatorias (por ejemplo los aportes al embalse por lluvias).

Para cada paso de tiempo k tendremos una función de costo $CE(x,u_k,r_k,k)$ a partir de la cual podemos definir el costo futuro como:

$$CF(x,U_k,R_k,k)=\sum_{j=k}^{\infty} q^{j-k}\cdot CE(x_j,u_j,r_j,j)$$
$$U_k=\{u_k,u_{k+1},.....\} \quad (1)$$
$$R_k=\{r_k,r_{k+1},.....\}$$

Donde q es un factor de depreciación del futuro (el dinero en el futuro vale menos que el dinero hoy), $U_k=\{u_k,u_{k+1},.....\}$ es una realización de las entradas de control desde el paso k en adelante, $R_k=\{r_k,r_{k+1},.....\}$ es una realización de las entradas no controladas desde el paso k en adelante y x_{j+1} viene dada por la función de evolución de estados f

$$x_{j+1} = f(x_j, u_j, r_j, j) \quad (2)$$

Observando la sumatoria con la que definimos el costo futuro, vemos que podemos realizar la definición en forma recursiva:

$$CF(x,U_k,R_k,k) = CE(x,u_k,r_k,k) + q \cdot CF(x',U_{k+1},R_{k+1},k+1) \quad (3)$$
$$x' = f(x,u_k,r_k,k)$$

El problema del optimizador será encontrar la serie U_k que minimice el valor esperado de $CF(x,U_k,R_k,k)$ para todas las realizaciones de R_k. Llamemos $CF(x,k)$ al valor del mínimo costo futuro que es posible obtener usando la mejor política de operación, cuando partimos en la etapa *k* desde el estado x. El problema resultante es hallar:

$$CF(x,k) = \left\langle \min_{u_k} \{CE(x,u_k,r_k,k) + q \cdot CF(x',k+1)\} \right\rangle_{r_k}$$
$$s.a. \quad (4)$$
$$x' = f(x,u_k,r_k,k)$$
$$g(x,u,r,k) \le 0$$

Donde $g(x,u,r,k) \le 0$ representa las restricciones sobre el dominio de las u_k.

Como lo que suceda a partir de la etapa k+1 no afecta lo que sucede en la etapa k el problema es separable en pasos y conociendo el valor de $CF(x,k+1)$ para todo x podemos resolver el valor de $CF(x,k)$.

SimSEE dispone de la capacidad de optimizar el uso de ciertos recursos modelados como variables de estado por medio de la programación dinámica estocástica y aproximaciones lineales a la función de costo futuro conociendo su valor en una discretización de su dominio.

Para modelar los estados se utilizan variables de estado. Para cada variable de estado que se quiera considerar se da un rango de valores posibles y una discretización del mismo. Al producto cartesiano de estas discretizaciones le llamamos "frame" y a cada combinación de valores posibles de todas las variables de estado le llamaremos estrella.

A continuación se muestra el frame producido por las discretizaciones de dos variables de estado.

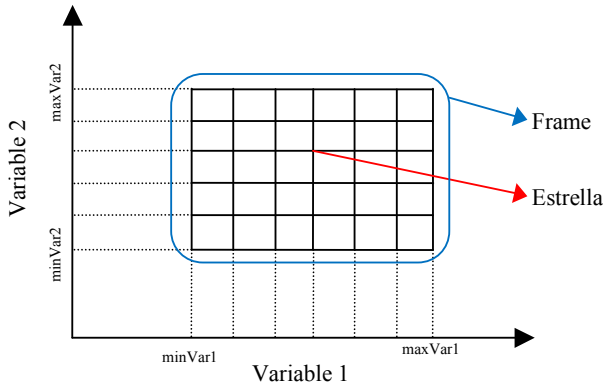

Fig. 4. Frame de Estados Compuesto Por Dos Variables

Si queremos cubrir todos los valores posibles de las variables de estado para todo el horizonte de simulación, tendremos un frame por cada paso de tiempo.

Al depender únicamente de los valores de $CF(x,k+1)$ para calcular el valor de $CF(x,k)$ podemos resolver el problema mediante la programación dinámica estocástica.

Nos posicionamos en el último paso de tiempo y le asignamos valor 0 a todos los estados, indicando que el costo del futuro luego del final de los tiempos es despreciable.

Luego hacemos un barrido en reversa desde el penúltimo paso de tiempo hacia el primero, barriendo el frame de estados calculando los valores de $CF(x,k)$ que utilizan los valores ya calculados de $CF(x,k+1)$. El pseudo-código del algoritmo es el siguiente:

para todo x hacer:
$$CF(x,k_{ultima+1}) = 0$$

para k desde k_{ultima} retrocediendo hasta 1 hacer:
para todo x hacer:
$$CF(x,k) = \left\langle \min_{u_k} \{CE(x,u_k,r_k,k) + q \cdot CF(x',k+1)\} \right\rangle_{r_k}$$
con $x' = f(x,u_k,r_k,k)$
y sujeto a $g(x,u,r,k) \le 0$

Resolver una estrella (4) se reduce a la resolución de un problema de programación lineal, en nuestro caso resuelto mediante el método simplex. Si bien la resolución de un problema de estos con el poder de cálculo de una computadora actual insume un tiempo pequeño, nos encontramos con un problema al querer abarcar el problema completo.

Sea D el producto cartesiano de las discretizaciones de las variables de estado. Como el valor de la función de costo futuro debe resolverse para cada punto dentro de D, el agregar una variable de estado más con N valores posibles a la

descripción del problema agrega tener que resolver N veces más problemas (todos los valores posibles de las demás variables de estado para cada uno de los N valores posibles de la nueva variable). La siguiente figura muestra como aumentaría el tamaño del frame de la Fig. 2 Si agregamos una nueva variable con 3 puntos en su discretización

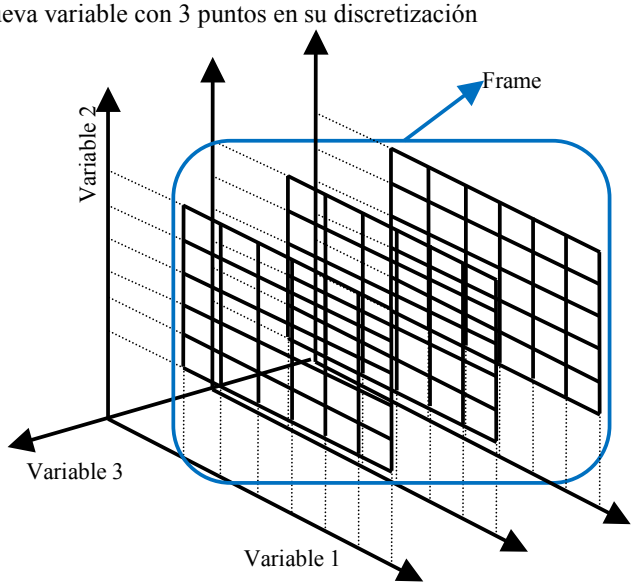

Fig. 5. Aumento del Tamaño del Frame de Estados al Agregar una Variable

Si a esto le sumamos que para resolver el problema estocástico debemos resolver cada punto de cada frame tantas veces como sorteos deseemos realizar, la cantidad de problemas que tendremos que resolver es |D| * nroSorteos * nroPasos. Supongamos que tenemos 5 variables de estado, tres de ellas con 10 valores posibles, una con 5 y otra con 3, que queremos realizar 10 sorteos y nuestra sala tiene un horizonte de simulación de 1500 pasos. La cantidad de problemas a resolver es 10^3*5*3*10*1500 = 225000000. La resolución de cada problema tarda en el orden de 0.2 milisegundos, por lo cual resolver el problema completo llevaría 12.5 horas, una cantidad poco deseable teniendo en cuenta que en general se busca hacer análisis de sensibilidad o pequeñas modificaciones al escenario de simulación y realizar varias optimizaciones para comparar resultados. Buscamos entonces mejorar este tiempo de ejecución.

IV. Optimización Distribuida

Como se mostró en el final de la sección 2 la optimización distribuida en NetTopos consta de dos tipos de aplicaciones:

Orquestador: se encarga de particionar el problema, ordenar y controlar la ejecución y acumular los resultados

OptDis: se encargan de realizar los cálculos instruidos por el orquestador y comunicarle los resultados

Cada paso de tiempo el Orquestador particiona el problema inicial y comunica a los OptDis que realicen los cálculos de los sub-problemas resultantes. Los OptDis realizan los cálculos y retornan sus resultados. El Orquestador colecciona estos resultados y mientras queden pasos de tiempo por resolver vuelve a ejecutar el ciclo.

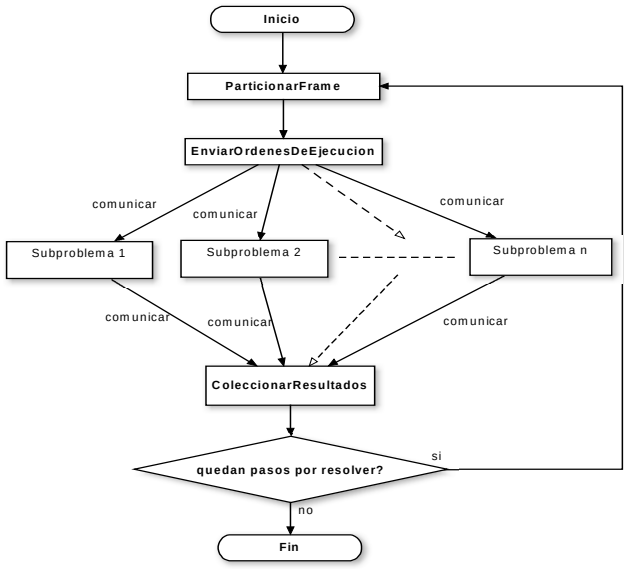

Fig. 6. Diagrama de flujo de una optimización distribuida

Siguiendo la metodología planteada en la sección 3, resolver la optimización de un sistema de energía eléctrica implica para cada paso de tiempo resolver el siguiente problema:

For i:= 1 to NEstrellas do

$$CF(x(i),k) = \left\langle \min_{u_k} \{CE(x(i),u_k,r_k,k) + q \cdot CF(x',k+1)\} \right\rangle_{r_k}$$

Donde: $x(i)$ es el punto i de la discretización del espacio de estados y $x' = f(x,u_k,r_k,k)$

Para obtener el valor esperado de las realizaciones de se utiliza el método de Montecarlo, sorteando distintas crónicas de las series, resolviendo el problema y luego promediando todos los resultados obtenidos.

For i:= 1 to NEstrellas do
  a:= 0;
  For s:= 1 to NSorteos do
    Sortear (r_k);
    $a := a + \min_{u_k} \{CE(x(i),u_k,r_k,k) + q \cdot CF(x',k+1)\}$

$CF(x(i),k) := a / NSorteos$

Particionar el problema en las distintas crónicas de sorteos representaría un aumento en la cantidad de comunicaciones necesarias debido a que para cada sorteo debería comunicarse un espacio de estados entero para acumular y promediar al terminar con los sorteos. Para evitar este problema se decidió particionar el frame de estados en rangos del espacio de estados, reduciendo el problema inicial a resolver n sub-

problemas concurrentes más pequeños de la siguiente forma:

For i:= estrellaInicial(rj) to estrellaFinal(rj) do
a:= 0;
For s:= 1 to NSorteos do
$Sortear\ \ (r_k)$;
$a := a + \min_{u_k} \{CE\ (x(i), u_k, r_k, k) + q \cdot CF\ (x', k+1)\}$
$CF\ (x(i), k) := a\ /\ NSorteos$

Donde: x(i) es el punto i de la discretización del espacio de estados y x'= f(x,uk,rk,k)

El resultado de este cálculo es únicamente los valores del costo futuro de las estrellas del rango, por lo que si d es la dimensión de un frame, al transmitir los n rangos se transmitirán únicamente las d estrellas, en lugar de d * NSorteos.

Para calcular los costos futuros del paso k es necesario conocer los valores de la función de costo futuro para cualquier estado resultante posible en el paso k+1. Esto hace que antes de que los OptDis puedan empezar a calcular el paso k se les debe transmitir la porción de espacio de estados que no calcularon en el paso k+1.

Juntando la partición de espacio de estados mencionada con el envío de los costos futuros no calculados el diagrama de flujo completo de la optimización distribuida es el siguiente:

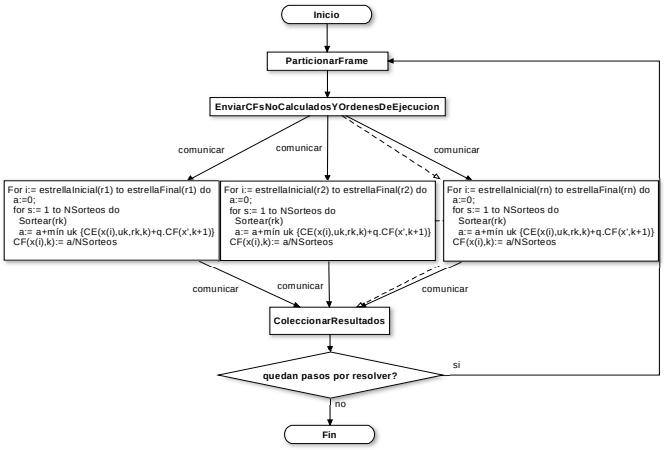

Fig. 7. Diagrama de flujo completo de una optimización distribuida

La partición del frame de estados juega un papel crucial en la reducción del tiempo de ejecución, dado que el tiempo de cálculo será proporcional a la porción de frame que el nodo tenga que calcular. Buscamos que todos los nodos demoren el mismo tiempo en realizar sus cálculos, de modo de no tener nodos ociosos. Si un nodo es más rápido se le asignara una mayor porción del espacio de estados, si es más lento, menor.

Para dar una indicación de la velocidad de cada nodo se dispone de una clase utilitaria a la que se le proporciona el peso relativo de la tarea que ejecuto un nodo y el tiempo que demoro en resolver dicha tarea. La clase se encarga de hacer el promedio móvil de las últimas n velocidades (pesoRelativo / tiempoEjecución), asignando este valor como velocidad del nodo. Luego el espacio de estados se reparte en forma proporcional a las velocidades de los nodos sobre el total de velocidad. A continuación se muestra para una corrida con 4 OptDis ejecutando en computadoras iguales como con la técnica mencionada rápidamente los pesos relativos asignados a los nodos convergen al valor correspondiente a la velocidad relativa del OptDis y los tiempos de ejecución tienden a igualarse.

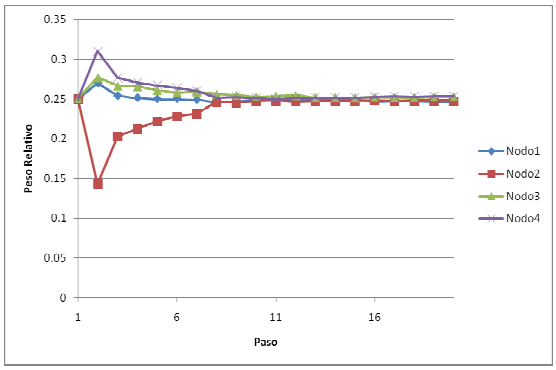

Fig. 8. Estabilización de los Pesos Relativos Asignados

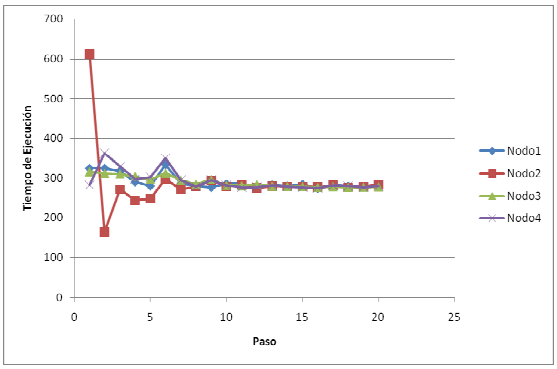

Fig. 9. Igualación de los Tiempos de Ejecución

V. Resultados Numéricos

Se realizaron diversas corridas representando el sistema uruguayo para los 3 meses entre el 13/09/2008 y el 13/12/2008, con un paso de tiempo horario, resultando en 2184 pasos de tiempo.

Se consideraron las centrales de Baygorria, Bonete, Palmar y Salto Grande, cada una con su embalse, con 5, 10, 5 y 5 discretizaciones respectivamente. Además se considero una variable de estado hidrológico para representar los aportes de lluvias con 5 discretizaciones, Resultando la cantidad total de estrellas por paso de tiempo 6250.

Las corridas se realizaron con 1 sorteo y con distintas cantidades de nodos.

A Continuación se presentan los resultados obtenidos:

TABLA I
RESULTADOS OBTENIDOS

Configuración	Tiempo de Ejecución[segundos]	Fracción del Tiempo Original
SimSEE	2456.469	100%
OptDisV2 1 Nodo	2474.16	100.72%
OptDisV2 2 Nodos	1259.7	51.28%
OptDisV2 3 Nodos	843.48	34.34%
OptDisV2 4 Nodos	666.8	27.14%
OptDisV2 5 Nodos	544	22.14%

El siguiente gráfico muestra los valores obtenidos contra el tiempo de ejecución si la disminución fuera lineal con la cantidad de nodos.

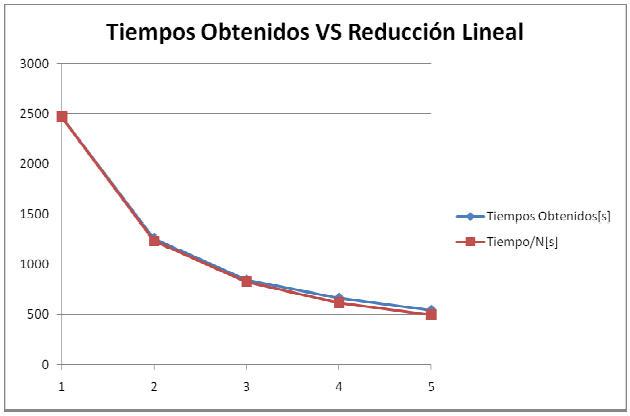

Fig. 10. Tiempos de Ejecución Obtenidos VS Reducción Lineal

Si bien los tiempos se ajustan bastante bien a una disminución lineal se puede ver sobre el final de las series una pequeña separación, producto del tiempo de comunicación. Vamos a buscar predecir cuales serán los tiempos de ejecución en función de la cantidad de nodos y determinar la cantidad óptima de nodos.

Al distribuir una aplicación, se intenta plantear el cálculo de forma de desacoplar lo más posible la comunicación necesaria entre los procesos y la distribución del cálculo sólo tiene sentido si el tiempo que se gana por la ejecución simultánea en los diferentes nodos es superior al tiempo que se agrega de comunicación entre los procesos. En nuestro caso el tiempo de resolución de un paso estará compuesto por el tiempo de cálculo de las estrellas y el tiempo agregado por las comunicaciones entre los OptDis y el Orquestador. Se definen las siguientes variables:

n: cantidad de OptDis en la red
d: número de estrellas por frame
s: cantidad de sorteos
t: tiempo de cálculo de una estrella
c: tiempo de comunicación de una estrella
k: tiempo de establecimiento de comunicación

Cada nodo calcula d / n estrellas, por lo que será necesario comunicarle las d – d / n estrellas que no calculo.

El tiempo de actualización de las estrellas que no haya calculado el nodo será proporcional a la cantidad de estrellas que no cálculo y se demora un tiempo fijo en establecer la comunicación, entonces:

$$T_{com} = \left(d - \frac{d}{n} \right) * c + k \tag{5}$$

El tiempo de cálculo de un nodo será proporcional a la cantidad de estrellas que tenga para resolver y a la cantidad de sorteos que tenga que realizar, de lo que resulta:

$$T_c = \frac{d}{n} * s * t \tag{6}$$

Finalmente el tiempo de respuesta de un nodo es el tiempo que tarda en devolver los resultados de sus cálculos y será proporcional a la cantidad de estrellas que haya resuelto y se demora un tiempo fijo en establecer la comunicación, entonces:

$$T_r = \frac{d}{n} * c + k \tag{7}$$

Con la estrategia implementada los tiempos de cálculo se superponen, pero los tiempos de comunicación se serializan (los tiempos de respuesta podrían no superponerse, pero asumiremos el peor caso en el cual no se superponen por simplicidad), resultando el tiempo de cálculo de un paso:

$$T_p = \left(\left(d - \frac{d}{n} \right) * c + k \right) * n + \frac{d}{n} * s * t + \left(\frac{d}{n} * c + k \right) * n \tag{8}$$

$$T_p = (d * c + 2k) * n + \frac{d}{n} * s * t$$

En esta fórmula se puede ver que si bien el tiempo de cálculo es inversamente proporcional a la cantidad de nodos, el tiempo de comunicación es directamente proporcional, por lo que agregar nodos indefinidamente terminaría redundando en un perjuicio y no en un beneficio como sería deseable.

Se realizaron medidas de las variables involucradas en el sistema y se obtuvieron los siguientes valores:

t = 0.00018 segundos/estrella
c = 7.83446E-07 segundos/estrella
k = 1.00188E-06 segundos

Con estos valores se obtuvieron las siguientes predicciones de tiempos:

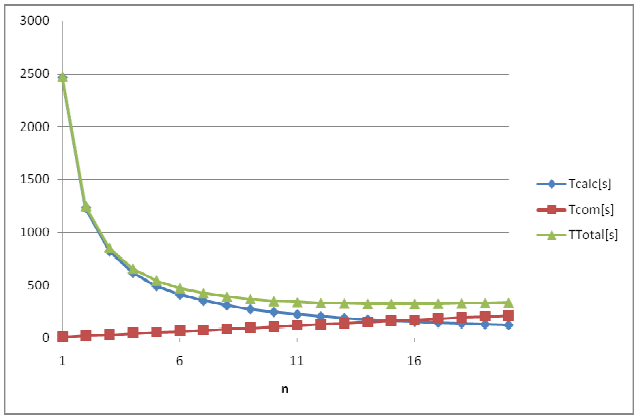

Fig. 11. Predicción de Tiempos de Cálculo, Comunicación y Total

Como se puede ver en la gráfica el límite teórico es alcanzado en 15 nodos, con una reducción al 13% del tiempo original. La ganancia real se da hasta los 10 OptDis en donde se obtiene una reducción al 14% del tiempo original.

VI. Conclusiones

La versión distribuida del SimSEE logró disminuir el tiempo de ejecución a 22% para un escenario con 5 computadoras. En términos prácticos, se logro reducir una corrida de 40 minutos a 9. Si bien no se elimina la maldición de la dimensionalidad se logra reducirla a niveles ejecutables. Esto permitirá realizar corridas considerando detalles como mínimos técnicos o estados de rotura de las máquinas que de otra forma demorarían demasiado.

Cabe aclarar además que la penalidad por la comunicación solo aplica si los procesadores se encuentran distribuidos a través de una red. Con el surgimiento de los equipos multi-core se podrían tener varios procesadores procesando cálculos con una sola comunicación. Para citar un ejemplo práctico el proyecto clúster del INCO contará con 8 nodos de cálculo comunicados a través de la red, cada uno con 8 procesadores. De esta forma el tiempo de cálculo se repartiría entre 64 procesadores, pero el tiempo de comunicación se multiplicaría únicamente por 8.

VII. Referencias

Disertaciones:

[1] R. Chaer. "SIMULACIÓN DE SISTEMAS DE ENERGÍA ELÉCTRICA," Tesis de Maestría, Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, UDeLaR, 2008.

Software:

[2] *Instituto de Ingeniería Eléctrica. (2008). SimSEE.* http://iie.fing.edu.uy/simsee/