UNIVERSIDAD NACIONAL DE COLOMBIA

Diagnóstico de Seguridad Mediante la Solución de Flujos de Potencia Post-Contingencia con Cómputo Paralelo

Autor:

María del Pilar Buitrago Villada

Universidad Nacional de Colombia

Facultad de Ingeniería y Arquitectura

Departamento de Ingeniería Eléctrica, Electrónica y Computación

Manizales, Colombia

2015

UNIVERSIDAD NACIONAL DE COLOMBIA

Security Diagnosis Through Post-Contingency Power Flow Solution with Parallel Computing

Author:

María del Pilar Buitrago Villada

Universidad Nacional de Colombia

Facultad de Ingeniería y Arquitectura

Departamento de Ingeniería Eléctrica, Electrónica y Computación

Manizales, Colombia

2015

Diagnóstico de Seguridad Mediante la Solución de Flujos de Potencia Post-Contingencia con Cómputo Paralelo

María del Pilar Buitrago Villada

Tesis presentada como requisito parcial para optar al título de:

Magister en Ingeniería – Ingeniería Eléctrica

Director

Carlos Edmundo Murillo Sánchez

Codirector

Francisco Abel Roldán Hoyos

Línea de Investigación

Sistemas Eléctricos de Potencia

Grupo de Investigación:

Potencia, Energía y Mercados – GIPEM

Universidad Nacional de Colombia

Facultad de Ingeniería y Arquitectura

Departamento de Ingeniería Eléctrica, Electrónica y Computación

Manizales, Colombia

2015

A Dios.

A mis padres José Uriel y Martha Cecilia.

A mis hermanos Juan Pablo y Ana María.

A mis hermanos en la fe.

Agradecimientos

En primer lugar gracias a Dios, por permitirme encontrar ángeles en mí camino durante todo este periodo de estudio.

A los profesores Carlos Edmundo Murillo Sánchez y Francisco Abel Roldán Hoyos por sus valiosos aportes y tiempo dedicado en la dirección de esta tesis. A ellos muchas gracias por la paciencia y confianza que depositaron en mí y sobre todo por sus enseñanzas que contribuyeron a mi crecimiento personal y profesional.

Un agradecimiento especial a mi director, por poner al servicio de éste trabajo su *cluster* de cómputo, esencial para la consecución de resultados.

A los jurados, Jaime Quintero Restrepo y Jorge Fernando Gutiérrez Gómez, por sus sugerencias y aportes que han permitido mejorar el documento final de tesis.

A la Universidad Nacional de Colombia por brindarme la oportunidad de trabajar en el Laboratorio de Electricidad y Electrónica, y por su patrocinio durante cada semestre académico.

A *Cornell University*, por facilitar su computador multinúcleo, permitiendo complementar los análisis realizados.

A mi familia y a mis hermanos por su apoyo incondicional.

Resumen

Una extensión a la formulación matemática y al esquema de solución del problema de análisis estático de contingencias de primer orden (N-1), incorporando el concepto de barra *slack* distribuida por AGC es propuesta. En la formulación clásica del estudio de flujo de carga, basada en el método de Newton-Raphson, se adicionaron ecuaciones de potencia activa que involucran los generadores pertenecientes al AGC, mediante sus factores de participación.

Se desarrolló una aplicación para análisis de contingencia en paralelo, diseñada con base en un esquema de asignación dinámica de contingencias, desarrollado bajo tres estrategias, una basada en la disponibilidad de las unidades de procesamiento, otra en la priorización de tareas y una más priorizando tareas y procesadores.

Las pruebas se realizaron empleando dos plataformas de cómputo, una compuesta por un *cluster* heterogéneo de 6 nodos y la otra por un computador de 12 núcleos. Como sistemas de prueba se eligieron los sistemas de potencia de 30, 118, 300 y 2383 barras, con 32, 196, 447 y 2681 contingencias simples respectivamente, involucrando la desconexión de líneas y generadores.

Palabras clave: Cálculo del análisis de sistemas de potencia, control automático de generación, método de Newton, procesamiento multinúcleo, programación paralela, prototipos de software, seguridad del sistema de potencia.

Abstract

An extension to the mathematical formulation and solution scheme of the first order (N-1) static contingency analysis incorporating the concept of distributed slack bus by AGC is proposed. In the classic formulation of load flow study, based on the Newton-Raphson method, equations were added to balance the active power imbalances involving the AGC generators through their participation factors.

An application was developed for parallel contingency analysis, designed using a dynamic scheme for allocation of contingencies to worker nodes in a cluster. Three assignment strategies were developed, one based on the availability of processing units, another using task priority and one more in prioritizing both tasks and processors.

The tests were performed in two computing platforms, one composed of six heterogeneous nodes and the other being a twelve core computer. As test cases, systems with 30, 118, 300 and 2383 buses were chosen with 32, 196, 447 and 2681 simple contingencies respectively. The contingencies involved both line and generator outages.

Keywords: Automatic Generation Control (AGC), multicore processing, Newton method, parallel programing, power system analysis computing, power system security, software prototyping.

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

La seguridad en sistemas eléctricos de potencia es la capacidad de un sistema de resistir perturbaciones repentinas sin consecuencias graves [1]. Esto involucra estudios que evalúen el estado de operación del sistema cuando una contingencia ocurre, previniendo fallas en cascada que conduzcan a apagones.

Una contingencia es la falla o interrupción inesperada de uno o varios elementos del sistema (generador, línea de transmisión, interruptor, etc.) [2]. El análisis de contingencias es un estudio que simula cada contingencia posible antes de que suceda, evalúa los resultados post-contingencia y comprueba si existen violaciones en los límites operacionales de los componentes del sistema.

La identificación de contingencias críticas es muy importante durante la fase de planeación. En esta etapa *off-line*, el tiempo de cálculo no es un factor importante, y más allá del estudio de contingencias de primer orden (N-1), pueden llevarse a cabo estudios de contingencias de orden superior (N-k). Durante la fase de operación, los tiempos de cálculo deben reducirse drásticamente, por lo que sólo se estudian contingencias de primer orden normalmente [3].

En un sistema de gran escala, el número de posibles contingencias es muy grande y la ventana de tiempo para analizar los puntos conflictivos y tomar acciones preventivas (pre-contingencia) y correctivas (post-contingencia) apropiadas es limitada [4].

Existen tres formas básicas de ejecutar muy rápido un análisis de seguridad [5]: estudiar el sistema de potencia con un algoritmo aproximado pero muy rápido, seleccionar sólo los casos importantes para análisis detallado, o usar un sistema de cómputo con múltiples procesadores o procesadores vectoriales para ganar velocidad. La primera forma ha sido estudiada y usada por muchos años de múltiples maneras [6]-[11]; con métodos de sensibilidad lineal, métodos de flujo de potencia DC [5], entre otros, siendo este último el más usado. A pesar de su fácil implementación, eficiente solución y optimización, es inherentemente aproximado y su precisión es muy dependiente del sistema y del

caso [12]. Además, está limitado a aplicaciones orientadas al flujo de potencia activa, donde los efectos de los flujos de potencia reactiva y las magnitudes de tensión de barra no son considerados.

La segunda forma, la selección de contingencias basada en índices de desempeño y en filtrado de contingencias, ha sido documentada en [3], [13]-[21]. Las listas de contingencias derivadas de estos, y que definen los casos de contingencia para análisis detallado, pueden presentar errores de clasificación, dejando muy abajo en la lista casos que pueden ser críticos [22]. Esto se conoce como efecto de enmascaramiento [23]. A pesar de los esfuerzos para mejorar la precisión de los índices de desempeño, algunas malas clasificaciones son inevitables [24].

Otros estudios reportados combinan el uso de un algoritmo aproximado con la selección de contingencias como en [25]-[32].

La última de las formas referidas más arriba, relacionada con el incremento del poder de cálculo, ha experimentado el desarrollo de redes de cómputo, de *cluster* de computadores, de equipos multinúcleo, y el uso de unidades de procesamiento gráfico o GPU (*Graphic Processing Unit*) en aplicaciones de cálculo de propósito general.

La revisión del estado del arte de cómputo de alto desempeño para la solución de problemas en sistemas de potencia es presentada en [33] [34]. Se destaca la tendencia hacia las computadoras multinúcleo y la necesidad del cómputo paralelo para utilizar estos equipos.

Dado que el análisis de contingencias es un proceso altamente paralelizable, puesto que es equivalente a resolver múltiples problemas de flujo de potencia, con muy poca comunicación entre cada caso de contingencia, son encontradas en la literatura como propuestas para su implementación en paralelo: el uso de plataformas de procesamiento distribuido [35], equipos multinúcleo [36], *cluster* [37] [38], combinación de *cluster* con múltiples procesadores [39] [40], y GPU [41]. Algunas otras [36] [42] [43], consideran de gran interés el estudio de balance de carga computacional.

Un análisis de contingencia generalmente se realiza mediante la ejecución de un flujo de potencia para cada caso de desconexión de los elementos del sistema. En el flujo de potencia clásico, existe una suposición irreal bajo la cual todas las pérdidas de potencia del sistema son asumidas por el generador en la barra *slack* única [44]. El propósito de ésta suposición es simplemente disminuir la complejidad del sistema de ecuaciones simultáneas no lineales resultante, al aprovechar una

dependencia escalonada de las variables a encontrar: primero se encuentran magnitudes de tensiones y sus ángulos fasoriales, y después algunas inyecciones específicas remanentes.

Entonces, el concepto de barra *slack* única, es sencillamente una escogencia ingenieril, cuya característica de compensación de pérdidas de potencia activa no tiene relación física con alguna barra de generación real. A diferencia de lo supuesto por el modelo con barra *slack* única, ningún generador real cuenta con capacidad de generación ilimitada. Por lo tanto, a menos que la barra *slack* sea el equivalente de una gran red, el modelo con barra *slack* concentrada no es realista [45].

En la operación actual de los sistemas de potencia no existe una barra *slack* única, en su lugar, muchos generadores distribuidos geográficamente asumen ésta función. Las barras que conectan generadores cuyas salidas de potencia activa sean ajustables o con AGC (*Automatic Generation Control*), pueden seleccionarse como barras *slack*, distribuyendo entre ellas los errores de potencia a través de algunas reglas, como los "*factores de participación*".

Según [46], los factores de participación son coeficientes que pueden determinarse basados en: la inercia de las máquinas, la característica de regulación de velocidad de los gobernadores o el despacho económico. Otros los relacionan con las características de las turbinas en cada barra de generación en la asignación de carga [47], con el uso de costo combinado y criterios de confiabilidad [48], o con la salida programada del generador [49].

Para tener en cuenta lo anterior, se hace necesario un modelo de flujo de carga con barra *slack* distribuida. Este modelo ha sido reportado tanto en estudios relacionados con sistemas de distribución [50]-[54], así como en sistemas de transmisión, especialmente para despacho económico [48], [49], [55]-[59].

Así mismo, el análisis de contingencias usando el modelo con barra *slack* distribuida, ha sido empleado como herramienta para calcular los índices de sensibilidad de inyección de potencia activa, orientado a la expansión en transmisión y generación de sistemas eléctricos en mercados desregulados [60]; y para caracterizar la respuesta de los generadores a las pérdidas de potencia activa y calcular la salida de potencia reactiva en el punto de colapso, orientado al control de frecuencia y tensión en el estudio de seguridad de sistemas de potencia interconectados [61]. Estos estudios reportaron beneficios del uso del modelo con barra *slack* distribuida, argumentado que éste representa de forma más real el funcionamiento del sistema de potencia.

De acuerdo con los antecedentes mencionados, en este documento se presenta una extensión a la formulación y al esquema de solución del problema de análisis estático de contingencias de primer orden (N-1), incorporando el concepto de *slack* distribuido mediante AGC. Igualmente, con el fin de hacer frente al problema de rapidez de ejecución del análisis de contingencias, se implementó la solución propuesta en un *cluster* de cómputo heterogéneo y en un computador multinúcleo, empleando un esquema de asignación dinámica de contingencias, desarrollado bajo tres estrategias, una basada en la disponibilidad de los trabajadores (núcleos de los procesadores), otra en la priorización de tareas y otra más en la priorización de tareas y trabajadores.

Para la verificación del desempeño de la implementación sugerida, se efectúo un análisis de la velocidad de procesamiento variando dos aspectos, que son el tamaño del sistema de potencia y el número de trabajadores en cada una de las dos opciones de hardware disponibles. Adicionalmente, en los casos con características de velocidad creciente con el número de procesadores usados, se calculó la aceleración máxima obtenida. Los resultados de las pruebas efectuadas, evidenciaron la superioridad en el desempeño del computador multinúcleo sobre el *cluster* heterogéneo.

El documento presenta en su *Introducción* la identificación e interés por el tema, así como una descripción del contenido. A continuación, el *Capítulo 2* abarca los conceptos básicos del análisis de contingencias, en particular lo relacionado con el modelo de flujo de potencia, e introduce la formulación matemática para el modelo de flujo de potencia propuesto, con *slack* distribuido mediante AGC.

Una revisión del estado del arte y de las herramientas de cómputo paralelo empleadas en el desarrollo del software para análisis de contingencias en paralelo, es realizada en el *Capítulo 3*; mientras que en el *Capítulo 4*, se expone la metodología adoptada para el desarrollo del software para análisis de contingencias, así como los resultados de la evaluación de su desempeño al usar sistemas de prueba con diversos tamaños. Finalmente, en el *Capítulo 5* se presentan las principales conclusiones, y se vislumbran trabajos futuros.

2. Formulación del esquema de solución del problema de flujo de potencia incorporando el concepto de barra *slack* distribuida mediante AGC

Un estudio de flujo de potencia determina la potencia activa y reactiva a través de las líneas de transmisión así como la tensión compleja en las barras de un sistema eléctrico de potencia, de acuerdo con unos esquemas de carga y generación preestablecidos. Con frecuencia, el método de Newton para la solución de ecuaciones es empleado para este propósito debido a su rapidez de convergencia numérica cuadrática.

En ausencia de un mejor criterio, la barra con la mayor capacidad de generación es arbitrariamente elegida como barra *slack,* y es la encargada de asumir todas las pérdidas y las variaciones en el suministro de potencia. Éstas últimas se asocian con incrementos de carga o con eventos que impidan que la carga sea servida, como pérdida de la capacidad de generación del sistema o desconexión de líneas de transmisión, las que pueden ser consecuencia de una contingencia.

En la práctica, los desbalances de potencia se distribuyen entre los generadores con características de regulación de generación, de acuerdo con los factores de participación, de tal forma que la variación del suministro de potencia activa de cada generador sea proporcional al cambio de la generación total del sistema. Esta labor se puede hacer manualmente o de forma automática a través del AGC.

Un modelo de flujo de potencia que incluya el concepto de barra *slack* distribuida mediante AGC, será desarrollado con miras a su posterior aplicación al análisis de contingencias.

2.1 Formulación clásica del problema de flujo de potencia AC

Matemáticamente, el problema de flujo de potencia consiste en la solución de un conjunto de ecuaciones algebraicas no lineales, planteadas como ecuaciones de balance de potencia nodal en función de las inyecciones de generación (P_g, Q_g) y las tensiones complejas de barra ($|V|$, δ), satisfaciéndose simultáneamente las condiciones de restricción de las barras.

Las ecuaciones (2.1) y (2.2) representan las ecuaciones de balance de potencia nodal, en sus componentes activa y reactiva respectivamente.

$$g_{P_i}(\delta, |V|) = P_{i,calc} - P_{i,prog} = P_{i,calc} + P_d - P_g = 0 \quad (2.1)$$

$$g_{Q_i}(\delta, |V|) = Q_{i,calc} - Q_{i,prog} = Q_{i,calc} + Q_d - Q_g = 0 \quad (2.2)$$

Las funciones $P_{i,calc}$ y $Q_{i,calc}$, que representan a las ecuaciones de flujo de potencia en forma polar, y cuya deducción se puede encontrar en [62], son

$$P_{i,calc} = \sum_{j=1}^{n} |V_i V_j Y_{ij}| \cos\left(\theta_{ij} + \delta_j - \delta_i\right) \quad (2.3)$$

$$Q_{i,calc} = \sum_{j=1}^{n} |V_i V_j Y_{ij}| \sin\left(\theta_{ij} + \delta_j - \delta_i\right), \quad (2.4)$$

donde los Y_{ij} son los coeficientes complejos de la matriz de admitancia nodal de la red.

Convencionalmente, una sola barra de generación (barra *slack*) sirve como referencia del ángulo de tensión y asume el balance global de potencia activa. En ésta, la magnitud y el ángulo de la tensión son conocidos. Las otras barras de generación se identifican como barras de tensión controlada (PV), donde la magnitud de la tensión y la inyección de potencia real son especificadas. Las barras sin generación o sin control de tensión se identifican como barras de carga (PQ), con los valores de potencia activa y reactiva dados. Para cada barra hay dos ecuaciones como las ecuaciones (2.1) y (2.2), en las que se especifican dos de las cuatro variables y se calculan las dos restantes.

El conjunto de ecuaciones a resolver está conformado por las ecuaciones (2.1) para las barras PV y PQ, y las ecuaciones (2.2) para las barras PQ. La Tabla 2-1 resume el problema de flujo de potencia, la que es similar a la presentada en [62].

Tabla 2-1: Resumen del problema de flujo de potencia

Tipo de Barra	**No. de Barras**	**Cantidades especificadas**	**No. de ecuaciones disponibles**	**No. de variables de estado δ_i, $\|V\|$**
Slack (i =1)	1	δ_1, $\|V_1\|$	0	0
PV ($i = 2, \dots, n_{pv} + 1$)	n_{pv}	P_i, $\|V_i\|$	n_{pv}	n_{pv}
PQ ($i = n_{pv} + 2 \dots n_b$)	n_{pq}	P_i, Q_i	$2n_{pq}$	$2n_{pq}$
Totales	$n_b = 1 + n_{pv} + n_{pq}$	$2n_b$	$n_{pv} + 2n_{pq}$	$n_{pv} + 2n_{pq}$

Luego de resolver el sistema para hallar los ángulos de tensión en las barras diferentes a la *slack* y las magnitudes de tensión en las barras PV, una ecuación de balance de potencia activa (ecuación (2.1)) puede usarse para calcular la potencia inyectada por el generador en la barra *slack*. Similarmente, las n_{pv} + 1 ecuaciones de balance de potencia reactiva restantes dan la inyección de potencia reactiva de todos los generadores.

2.1.1 El método de Newton

En el problema de flujo de potencia es necesario resolver un sistema de ecuaciones con ciertas condiciones iniciales. Mientras las ecuaciones sean no lineales, debe emplearse un esquema iterativo.

El método de Newton es aplicable a cierto sistema de ecuaciones no lineales, si la matriz Jacobiana correspondiente puede ser evaluada y si una aproximación inicial suficientemente buena es posible [63]. La ventaja principal de este método es que el número de iteraciones requerido para obtener una solución es independiente del tamaño del sistema y es computacionalmente rápido.

Las ecuaciones no lineales a resolver son del tipo (2.1) y (2.2), de modo que el sistema de ecuaciones es de la forma:

$$F_{(\delta,|V|)} = \begin{bmatrix} g_{P_{PV}}(\delta_{PV+PQ}, |V_{PQ}|) \\ g_{P_{PQ}}(\delta_{PV+PQ}, |V_{PQ}|) \\ g_{Q_{PQ}}(\delta_{PV+PQ}, |V_{PQ}|) \end{bmatrix} = 0 \qquad (2.5)$$

El problema originalmente no lineal es transformado en una secuencia de problemas lineales cuyas soluciones se aproximan a la solución del problema original, basado en la aproximación por series de Taylor de primer orden del sistema representado por (2.5).

$$F_{(\delta,|V|)} = F_{(\delta^0,|V|^0)} + \left[J_{(\delta^0,|V|^0)}\right]\left[\begin{bmatrix}\delta \\ |V|\end{bmatrix} - \begin{bmatrix}\delta \\ |V|\end{bmatrix}^0\right] \cong 0 \qquad (2.6)$$

Donde la matriz de coeficientes de la ecuación (2.6) es la matriz Jacobiana que se expresa como:

$$\left[J_{(\delta,|V|)}\right] \equiv \begin{bmatrix}\left[\frac{\partial g_P}{\partial \delta}\right] & \left[\frac{\partial g_P}{\partial |V|}\right] \\ \left[\frac{\partial g_Q}{\partial \delta}\right] & \left[\frac{\partial g_Q}{\partial |V|}\right]\end{bmatrix} \qquad (2.7)$$

Una expresión recurrente o iterativa se desarrolla a partir de la ecuación (2.6):

$$\begin{bmatrix}\delta_{PV+PQ}^{k+1} \\ |V_{PQ}|^{k+1}\end{bmatrix} = \begin{bmatrix}\delta_{PV+PQ}^{k} \\ |V_{PQ}|^{k}\end{bmatrix} - \begin{bmatrix}\left[\frac{\partial g_P}{\partial \delta}\right]^k & \left[\frac{\partial g_P}{\partial |V|}\right]^k \\ \left[\frac{\partial g_Q}{\partial \delta}\right]^k & \left[\frac{\partial g_Q}{\partial |V|}\right]^k\end{bmatrix}^{-1} * \begin{bmatrix} g_{P_{PV}} \\ g_{P_{PQ}} \\ g_{Q_{PQ}}\end{bmatrix}^k \qquad (2.8)$$

$$\begin{bmatrix}\Delta\delta_{PV+PQ} \\ \Delta|V_{PQ}|\end{bmatrix} = \begin{bmatrix}\delta_{PV+PQ} \\ |V_{PQ}|\end{bmatrix}^{k+1} - \begin{bmatrix}\delta_{PV+PQ} \\ |V_{PQ}|\end{bmatrix}^{k} = -\begin{bmatrix}\left[\frac{\partial g_P}{\partial \delta}\right]^k & \left[\frac{\partial g_P}{\partial |V|}\right]^k \\ \left[\frac{\partial g_Q}{\partial \delta}\right]^k & \left[\frac{\partial g_Q}{\partial |V|}\right]^k\end{bmatrix}^{-1} * \begin{bmatrix} g_{P_{PV}} \\ g_{P_{PQ}} \\ g_{Q_{PQ}}\end{bmatrix}^k \qquad (2.9)$$

Si δ y |V| convergen después de algunas iteraciones, de acuerdo con una tolerancia pre-especificada, sus valores finales son aceptados como la solución al problema de flujo de potencia.

2.2 *Slack* distribuido mediante AGC

El AGC es un sistema de supervisión y control, en tiempo real, que ajusta automáticamente la generación de potencia activa de un pequeño número de unidades de generación distribuidas geográficamente dentro de un área de control, como respuesta al desbalance de potencia real del sistema.

Durante la operación normal del sistema de potencia, se pueden identificar cuatro tareas con el propósito de AGC [64]: i) igualar la generación con la carga, ii) reducir las desviaciones de frecuencia a cero, iii) distribuir la generación total entre varias áreas de control cumpliendo con los flujos de interconexión programados, y iv) distribuir la generación de área individual entre sus fuentes de generación minimizando los costos operativos.

La primera tarea está asociada con el sistema primario o control de velocidad del gobernador, que responde proporcionalmente a las desviaciones de frecuencia local y normalmente lleva a cero el rango de cambio de la frecuencia dentro de una ventana de tiempo de varios segundos. Las últimas tres tareas son logradas por controles suplementarios dirigidos desde los centros de control del área. La segunda y tercera tareas están asociadas clásicamente con la función de regulación, o control de carga-frecuencia, mientras que la cuarta es asociada con la función de despacho económico del AGC [65].

Las características físicas, tanto del sistema de potencia interconectado como de las unidades de generación, dan forma a la naturaleza del problema de AGC. De acuerdo con [65], una característica del sistema que es fundamental para el problema de AGC, es que los generadores a través del sistema, responden naturalmente a los cambios de carga sin importar donde ocurran. Ésta es conocida como la respuesta primaria o natural y es llevada a cabo por los gobernadores de los generadores.

Un sistema interconectado consta de áreas de control de generación, conectadas por líneas de transmisión llamadas líneas de interconexión. Cada área de control es responsable de atender su propia carga, bien sea mediante sus propias unidades de generación o comprando potencia a otras áreas de control.

Cada área intenta regular su error de control de área (ACE, por sus siglas en inglés) a cero, al mantener los flujos de potencia en las líneas de interconexión y la frecuencia del sistema en sus valores programados, mientras satisface su propia carga. Esto se conoce como el control suplementario del AGC. Las áreas de control externas (en las que no sucede el cambio de carga) no inician ningún control suplementario mientras el área individual responde a su ACE y abastece su carga, reajustando la frecuencia del sistema [65].

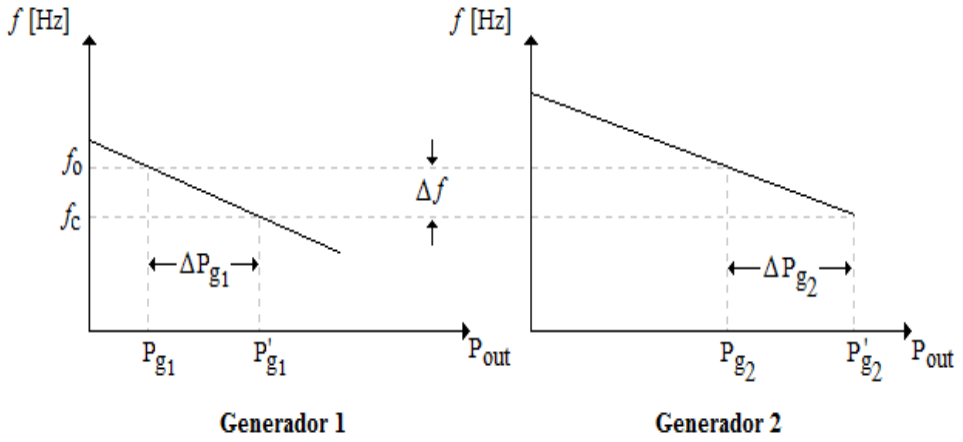
Sin embargo, todas las áreas de control contribuyen a la restauración de energía consumida por el cambio de carga inicial, suministrada desde la energía cinética almacenada en la masa rotativa de la turbina/generador, mientras los gobernadores responden para igualar la carga actual y la generación. La respuesta primaria y suplementaria del AGC no están, en realidad, ampliamente separadas a pesar de que esto sigue siendo un objetivo de diseño [65].

Aunque la respuesta primaria del gobernador no es directamente una parte del AGC, si es importante en el problema global. De hecho, de las cuatro tareas mencionadas anteriormente se puede deducir que el AGC permite que la respuesta primaria del sistema ocurra naturalmente [65].

2.2.1 Operación en paralelo de generadores con características de regulación de velocidad.

Si varios generadores, con gobernadores que tengan características de regulación de velocidad con respecto a la frecuencia (*speed-frequency droop*) son conectados al sistema de potencia, debe existir una frecuencia única en la que compartirán el cambio de carga [66]. La Figura 2-1 muestra las características de regulación de velocidad de dos generadores. Inicialmente, estos operan a frecuencia nominal f_0, con salidas de potencia Pg_1 y Pg_2. Un incremento de carga ΔP_L causa que los generadores se desaceleren y los gobernadores incrementan la salida de potencia hasta que alcancen una nueva frecuencia de operación común f_c.

Figura 2-1: Distribución de carga de dos generadores en paralelo con gobernadores con características de regulación de velocidad. Fuente: **[66]**.



La cantidad de carga asumida por cada generador dependerá de las características de regulación de velocidad, en este caso indicadas por R_1 y R_2:

$$\Delta P_{g1} = P'_{g1} - P_{g1} = -\frac{\Delta f}{R_1} \quad (2.10)$$

$$\Delta P_{g2} = P'_{g2} - P_{g2} = -\frac{\Delta f}{R_2} \quad (2.11)$$

Dividiendo la ecuación (2.10) entre la ecuación (2.11), se obtiene una expresión en función de Pg_1 y Pg_2,

$$\frac{\Delta P_{g1}}{\Delta P_{g2}} = \frac{R_2}{R_1} \quad (2.12)$$

De forma equivalente,

$$R_1\left(P'_{g1} - P_{g1}\right) = R_2\left(P'_{g2} - P_{g2}\right) \quad (2.13)$$

Según lo anterior, para mantener la frecuencia del sistema dentro de unos límites establecidos, se debe distribuir el incremento de carga entre cada generador, en proporción a sus características de regulación de velocidad R, de tal forma que éste parámetro se puede asumir como el factor de

participación (β) para cada unidad de generación. Por lo tanto, la expresión (2.12) puede escribirse como:

$$\beta_1(P'_{g1} - P_{g1}) = \beta_2(P'_{g2} - P_{g2}) \qquad (2.14)$$

2.3 Integración del concepto de *slack* distribuido al problema clásico de flujo de potencia

Sea $g_0\ (\delta,\ |V|,\ P_G,\ Q_G)$ una solución del problema de flujo de carga del sistema de potencia en ausencia de contingencias, y sea $\widehat{g_0}\ (\hat{\delta},\ |\hat{V}|,\ \hat{P}_G,\ \hat{Q}_G)$ la solución de un nuevo problema, que considera alguna contingencia simple, a partir de $g_0\ (\delta,\ |V|,\ P_G,\ Q_G)$, tal que:

1) $\widehat{g_0}$ presenta cambios con respecto a g_0 (líneas o generadores).
2) Los generadores presentes conservan su *setpoint* de tensión (dentro de su capacidad reactiva).
3) Un conjunto de los generadores, agrupados en su salida de potencia activa en el vector $P_{G_{AGC}}$, pueden efectivamente cambiar su salida activa de manera conjuntamente proporcional dentro de su capacidad, en función de sus factores de participación (β). La sumatoria de los factores de participación debe ser igual a la unidad.

   Las barras en las que se ubican estos generadores, cambiarán su nombre de barras PV a barras AGC y las barras con generación pero sin AGC continuaran siendo las barras PV.

Es de esperar que el comportamiento de la respuesta de los generadores bajo AGC, guarde similitud con el comportamiento de los gobernadores de los generadores y en consecuencia se pueden plantear ecuaciones donde el reajuste de la potencia activa de los generadores AGC sea proporcional a unos factores de participación (en función de su potencia nominal) de manera análoga a como la variación de la repartición de potencia de los generadores, es efectuada por sus respectivos gobernadores de manera proporcional a su estatismo.

Por lo tanto, se pueden plantear ecuaciones como la Ecuación (2.14):

$$\hat{g}_{fp}(P_{AGC}) = \left(P_{G_{ref}} - \hat{P}_{G_{ref}}\right) - \alpha_i\left(P_{G_i} - \hat{P}_{G_i}\right) = 0, \quad i = 2, \dots, n_{AGC}; \qquad ref = 1 \qquad (2.15)$$

Donde, $\alpha_i = \beta_i/\beta_{ref}$ es un factor de participación relativo. Se asume que la barra 1 sirve como referencia angular.

Entonces, después de cada iteración no sólo las magnitudes y ángulos de las tensiones deben ser actualizados, sino también la potencia activa generada por cada unidad participante en el AGC.

El conjunto de ecuaciones del flujo de potencia en sus nuevas variables es:

$$\hat{g}_P(\hat{\delta}, |\hat{V}|, \hat{P}_G) = \begin{bmatrix} \hat{g}_{P_{AGC}}(\hat{\delta}, |\hat{V}|, \hat{P}_{AGC}) \\ \hat{g}_{P_{PV}}(\hat{\delta}, |\hat{V}|, \hat{P}_{g_{PV}}) \\ \hat{g}_{P_{PQ}}(\hat{\delta}, |\hat{V}|) \end{bmatrix} \quad (2.16)$$

$$\hat{g}_Q(\hat{\delta}, |\hat{V}|, \hat{Q}_G) = \begin{bmatrix} \hat{g}_{Q_{AGC+PV}}(\hat{\delta}, |\hat{V}|, \hat{Q}_G) \\ \hat{g}_{Q_{PQ}}(\hat{\delta}, |\hat{V}|) \end{bmatrix} \quad (2.17)$$

Lo anterior se refleja en el sistema de ecuaciones, representado por:

$$\hat{F}(\hat{\delta}, |\hat{V}|, \hat{P}_G) = \begin{bmatrix} \hat{g}_{P_{AGC}}(\hat{\delta}, |\hat{V}|, \hat{P}_{AGC}) \\ \hat{g}_{P_{PV}}(\hat{\delta}, |\hat{V}|, \hat{P}_{g_{PV}}) \\ \hat{g}_{P_{PQ}}(\hat{\delta}, |\hat{V}|) \\ \hat{g}_{Q_{PQ}}(\hat{\delta}, |\hat{V}|) \\ \hat{g}_{fp}\ (\hat{P}_{AGC}) \end{bmatrix} \quad (2.18)$$

En el que $\hat{g}_{fp}(\hat{P}_{AGC})$ representa el cambio proporcional de la salida de potencia activa de los generadores participando en AGC, de acuerdo con la Ecuación (2.15).

Las incógnitas para el flujo de carga con *slack* distribuido, agrupadas en el vector *x* son:

$$x = \begin{bmatrix} \hat{\delta}_{nb-1} \\ |\hat{V}_{PQ}| \\ \hat{P}_{G_{AGC}} \end{bmatrix} \quad (2.19)$$

La matriz Jacobiana extendida está dada por la forma:

$$J = \begin{bmatrix} \frac{\partial \hat{g}_{P_{AGC}}}{\partial \hat{\delta}_{nb-1}} & \frac{\partial \hat{g}_{P_{AGC}}}{\partial |\hat{V}_{PQ}|} & \frac{\partial \hat{g}_{P_{AGC}}}{\partial \hat{P}_{AGC}} \\ \frac{\partial \hat{g}_{P_{PV+PQ}}}{\partial \hat{\delta}_{nb-1}} & \frac{\partial \hat{g}_{P_{PV+PQ}}}{\partial |\hat{V}_{PQ}|} & \frac{\partial \hat{g}_{P_{PV+PQ}}}{\partial \hat{P}_{AGC}} \\ \frac{\partial \hat{g}_{Q_{PQ}}}{\partial \hat{\delta}_{nb-1}} & \frac{\partial \hat{g}_{Q_{PQ}}}{\partial |\hat{V}_{PQ}|} & \frac{\partial \hat{g}_{Q_{PQ}}}{\partial \hat{P}_{AGC}} \\ \frac{\partial \hat{g}_{fp}}{\partial \hat{\delta}_{nb-1}} & \frac{\partial \hat{g}_{fp}}{\partial |\hat{V}_{PQ}|} & \frac{\partial \hat{g}_{fp}}{\partial \hat{P}_{AGC}} \end{bmatrix} \quad (2.20)$$

Las columnas y filas resaltadas en la matriz Jacobiana extendida, corresponden a los términos adicionales provenientes del flujo de carga con *slack* distribuido.

Donde,

$$\frac{\partial \hat{g}_{fp}}{\partial \hat{\delta}_{nb-1}} = 0_{(n_{AGC}-1 \times n_B-1)}$$

$$\frac{\partial \hat{g}_{fp}}{\partial \hat{V}_{PQ}} = 0_{(n_{AGC}-1 \times n_{PQ})}$$

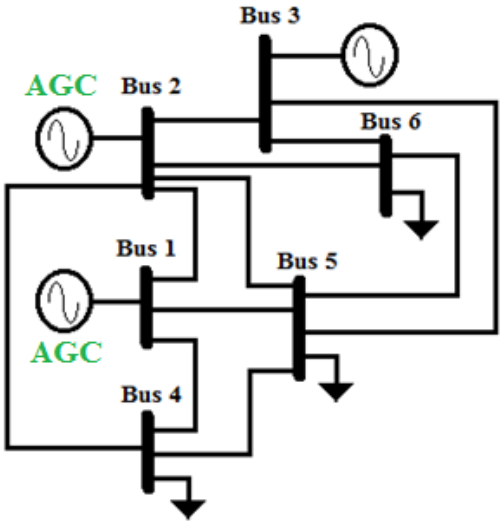
$$\frac{\partial \hat{g}_{Q_{PQ}}}{\partial \hat{P}_{AGC}} = 0_{(n_{PQ} \times n_{AGC})}$$

2.3.1 Ejemplo de aplicación de la formulación matemática del flujo de potencia con *slack* distribuido

El sistema de potencia 6 barras, presentado en la Figura 2-2, es empleado para ejemplificar la formulación matemática del flujo de potencia con *slack* distribuido por AGC. En este sistema de potencia con tres barras de generación, dos se designan con el propósito de AGC, y la referencia angular es provista por el generador en la barra 1.

Las cantidades especificadas para las barras AGC son el ángulo de la barra de referencia y las magnitudes de tensión; para las barras PV son las potencias activas y las magnitudes de tensión; y para las barras PQ las potencias activas y reactivas.

Figura 2-2 Sistema de potencia de 6 barras **[5]**



De acuerdo con lo anterior, para éste ejemplo, las incógnitas para el flujo de carga con *slack* distribuido, agrupadas en el vector *x* son:

$$x = \begin{bmatrix} \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ V_4 \\ V_5 \\ V_6 \\ P_{G_1} \\ P_{G_2} \end{bmatrix}$$

Para cada barra del sistema se definen dos ecuaciones de balance de potencia nodal, como las Ecuaciones (2.1) y (2.2), en sus componentes activa y reactiva. Adicionalmente, se definen n_{AGC-1} ecuaciones como la Ecuación (2.15), que representen el cambio proporcional de la salida de potencia activa de los generadores participando en AGC.

La Ecuación (2.15) relaciona por parejas, la potencia activa generada por los generadores bajo AGC, por lo tanto, en éste ejemplo se tendrá una sola ecuación de este tipo.

$$g_{fp} = \left(P_{G_1} - \hat{P}_{G_1}\right) - \alpha_1\left(P_{G_2} - \hat{P}_{G_2}\right) = 0$$

Donde, $\alpha_1 = {\beta_2}/{\beta_1}$ es el factor de participación relativo con β_1 y β_2 representando los factores de participación del AGC de las barras 1 y 2; $\hat{P}_{G_1}$ y $\hat{P}_{G_2}$ son las potencias activas generadas, en el estado pre-contingencia, por los generadores en las barras 1 y 2 respectivamente.

El conjunto de ecuaciones de flujo de potencia está representado por:

$$F = \begin{bmatrix} g_{P_1} \\ g_{P_2} \\ g_{P_3} \\ g_{P_4} \\ g_{P_5} \\ g_{P_6} \\ g_{Q_4} \\ g_{Q_5} \\ g_{Q_6} \\ g_{fp} \end{bmatrix}$$

Una expresión iterativa se construye a partir de este conjunto de ecuaciones:

$$\begin{bmatrix} \Delta\delta_{AGC+PV+PQ-1} \\ \Delta V_{PQ} \\ \Delta P_G \end{bmatrix} = \begin{bmatrix} \delta_{AGC+PV+PQ-1} \\ V_{PQ} \\ P_G \end{bmatrix}^{k+1} - \begin{bmatrix} \delta_{AGC+PV+PQ} \\ V_{PQ} \\ P_G \end{bmatrix}^{k} = - \begin{bmatrix} \left[\frac{\partial g_{P_{AGC}}}{\partial \delta_{nb-1}}\right]^k & \left[\frac{\partial g_{P_{AGC}}}{\partial |V_{PQ}|}\right]^k & \left[\frac{\partial g_{P_{AGC}}}{\partial P_{AGC}}\right]^k \\ \left[\frac{\partial g_{P_{PV+PQ}}}{\partial \delta_{nb-1}}\right]^k & \left[\frac{\partial g_{P_{PV+PQ}}}{\partial |V_{PQ}|}\right]^k & \left[\frac{\partial g_{P_{PV+PQ}}}{\partial P_{AGC}}\right]^k \\ \left[\frac{\partial g_{Q_{PQ}}}{\partial \delta_{nb-1}}\right]^k & \left[\frac{\partial g_{Q_{PQ}}}{\partial |V_{PQ}|}\right]^k & \left[\frac{\partial g_{Q_{PQ}}}{\partial P_{AGC}}\right]^k \\ \left[\frac{\partial g_{fp}}{\partial \delta_{nb-1}}\right]^k & \left[\frac{\partial g_{fp}}{\partial |V_{PQ}|}\right]^k & \left[\frac{\partial g_{fp}}{\partial P_{AGC}}\right]^k \end{bmatrix}^{-1} * \begin{bmatrix} g_{P_{AGC}} \\ g_{P_{PV}} \\ g_{P_{PQ}} \\ g_{Q_{PQ}} \\ g_{FP} \end{bmatrix}^k$$

La matriz Jacobiana extendida debe contener: dos columnas adicionales, en concordancia con las dos nuevas variables P_{G1} y P_{G2}, y dos filas adicionales correspondientes a la ecuación ∂g_{fp} y a la ecuación de balance de potencia activa nodal de la barra de referencia. La matriz Jacobiana está dada por la forma:

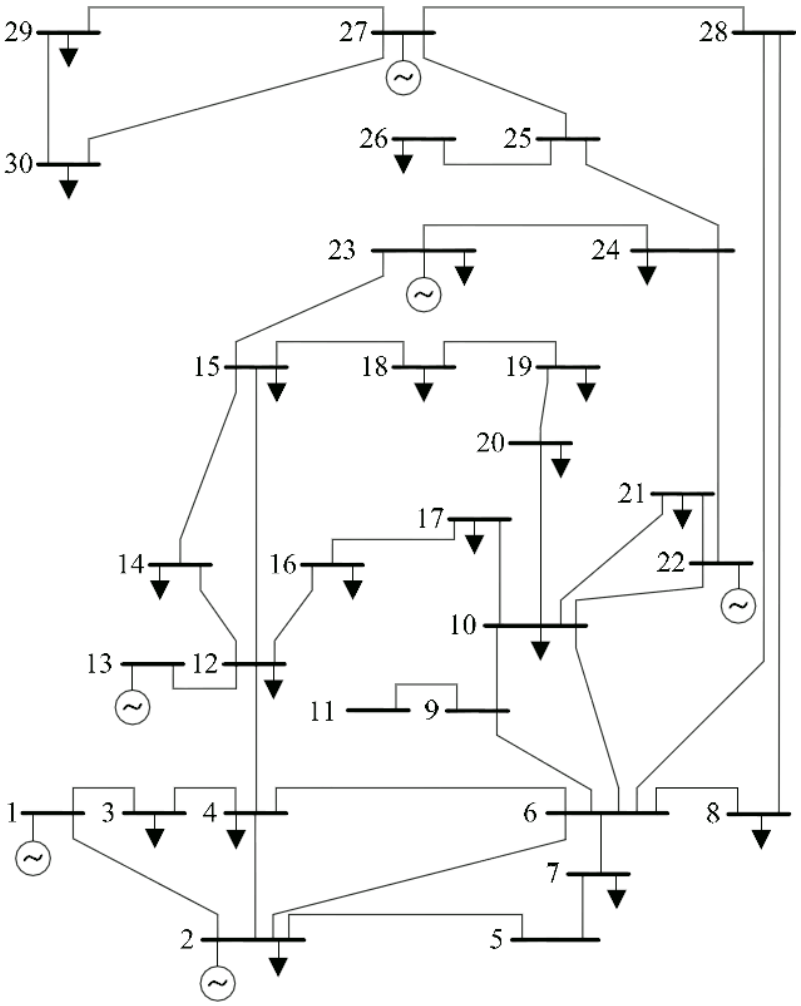
$$J = \begin{bmatrix}
\frac{\partial g_{P_1}}{\partial \delta_2} & \frac{\partial g_{P_1}}{\partial \delta_3} & \frac{\partial g_{P_1}}{\partial \delta_4} & \frac{\partial g_{P_1}}{\partial \delta_5} & \frac{\partial g_{P_1}}{\partial \delta_6} & \frac{\partial g_{P_1}}{\partial V_4} & \frac{\partial g_{P_1}}{\partial V_5} & \frac{\partial g_{P_1}}{\partial V_6} & \frac{\partial g_{P_1}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{P_1}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{P_2}}{\partial \delta_2} & \frac{\partial g_{P_2}}{\partial \delta_3} & \frac{\partial g_{P_2}}{\partial \delta_4} & \frac{\partial g_{P_2}}{\partial \delta_5} & \frac{\partial g_{P_2}}{\partial \delta_6} & \frac{\partial g_{P_2}}{\partial V_4} & \frac{\partial g_{P_2}}{\partial V_5} & \frac{\partial g_{P_2}}{\partial V_6} & \frac{\partial g_{P_2}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{P_2}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{P_3}}{\partial \delta_2} & \frac{\partial g_{P_3}}{\partial \delta_3} & \frac{\partial g_{P_3}}{\partial \delta_4} & \frac{\partial g_{P_3}}{\partial \delta_5} & \frac{\partial g_{P_3}}{\partial \delta_6} & \frac{\partial g_{P_3}}{\partial V_4} & \frac{\partial g_{P_3}}{\partial V_5} & \frac{\partial g_{P_3}}{\partial V_6} & \frac{\partial g_{P_3}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{P_3}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{P_4}}{\partial \delta_2} & \frac{\partial g_{P_4}}{\partial \delta_3} & \frac{\partial g_{P_4}}{\partial \delta_4} & \frac{\partial g_{P_4}}{\partial \delta_5} & \frac{\partial g_{P_4}}{\partial \delta_6} & \frac{\partial g_{P_4}}{\partial V_4} & \frac{\partial g_{P_4}}{\partial V_5} & \frac{\partial g_{P_4}}{\partial V_6} & \frac{\partial g_{P_4}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{P_4}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{P_5}}{\partial \delta_2} & \frac{\partial g g_{P_5}}{\partial \delta_3} & \frac{\partial g_{P_5}}{\partial \delta_4} & \frac{\partial g_{P_5}}{\partial \delta_5} & \frac{\partial g_{P_5}}{\partial \delta_6} & \frac{\partial g_{P_5}}{\partial V_4} & \frac{\partial g_{P_5}}{\partial V_5} & \frac{\partial g_{P_5}}{\partial V_6} & \frac{\partial g_{P_5}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{P_5}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{P_6}}{\partial \delta_2} & \frac{\partial g_{P_6}}{\partial \delta_3} & \frac{\partial g_{P_6}}{\partial \delta_4} & \frac{\partial g_{P_6}}{\partial \delta_5} & \frac{\partial g_{P_6}}{\partial \delta_6} & \frac{\partial g_{P_6}}{\partial V_4} & \frac{\partial g_{P_6}}{\partial V_5} & \frac{\partial g_{P_6}}{\partial V_6} & \frac{\partial g_{P_6}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{P_6}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{Q_4}}{\partial \delta_2} & \frac{\partial g_{Q_4}}{\partial \delta_3} & \frac{\partial g_{Q_4}}{\partial \delta_4} & \frac{\partial g_{Q_4}}{\partial \delta_5} & \frac{\partial g_{Q_4}}{\partial \delta_6} & \frac{\partial g_{Q_4}}{\partial V_4} & \frac{\partial g_{Q_4}}{\partial V_5} & \frac{\partial g_{Q_4}}{\partial V_6} & \frac{\partial g_{Q_4}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{Q_4}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{Q_5}}{\partial \delta_2} & \frac{\partial g_{Q_5}}{\partial \delta_3} & \frac{\partial g_{Q_5}}{\partial \delta_4} & \frac{\partial g_{Q_5}}{\partial \delta_5} & \frac{\partial g_{Q_5}}{\partial \delta_6} & \frac{\partial g_{Q_5}}{\partial V_4} & \frac{\partial g_{Q_5}}{\partial V_5} & \frac{\partial g_{Q_5}}{\partial V_6} & \frac{\partial g_{Q_5}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{Q_5}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{Q_6}}{\partial \delta_2} & \frac{\partial g_{Q_6}}{\partial \delta_3} & \frac{\partial g_{Q_6}}{\partial \delta_4} & \frac{\partial g_{Q_6}}{\partial \delta_5} & \frac{\partial g_{Q_6}}{\partial \delta_6} & \frac{\partial g_{Q_6}}{\partial V_4} & \frac{\partial g_{Q_6}}{\partial V_5} & \frac{\partial g_{Q_6}}{\partial V_6} & \frac{\partial g_{Q_6}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{Q_6}}{\partial \hat{P}_{G_2}} \\
\frac{\partial g_{fp}}{\partial \delta_2} & \frac{\partial g_{fp}}{\partial \delta_3} & \frac{\partial g_{fp}}{\partial \delta_4} & \frac{\partial g_{fp}}{\partial \delta_5} & \frac{\partial g_{fp}}{\partial \delta_6} & \frac{\partial g_{fp}}{\partial V_4} & \frac{\partial g_{fp}}{\partial V_5} & \frac{\partial g_{fp}}{\partial V_6} & \frac{\partial g_{fp}}{\partial \hat{P}_{G_1}} & \frac{\partial g_{fp}}{\partial \hat{P}_{G_2}}
\end{bmatrix}$$

Después de cada iteración, las magnitudes y los ángulos de las tensiones y la potencia activa generada por cada unidad participante en el AGC deben ser actualizados.

2.4 Comparación entre los resultados del flujo de carga con barra *slack* concentrada y el flujo de carga con barra *slack* distribuida

El sistema de potencia IEEE de 30 barras (Figura 2-3), fue elegido para efectuar la comparación entre los resultados del flujo de potencia con barra *slack* concentrada y con barra *slack* distribuida, tomando como parámetros de interés las tensiones complejas de nodo y los flujos de potencia en las líneas de transmisión.

Figura 2-3 Diagrama unifilar del sistema de prueba IEEE 30 barras **[67]**



Se realizaron varias combinaciones para el *slack* distribuido con las diferentes barras de generación del sistema y sus resultados se compararon con los arrojados por un estudio de flujo de potencia con *slack* concentrado en la barra 27, siendo ésta la barra *slack* originalmente propuesta en la literatura. Entre las combinaciones probadas, la conformada por las barras 1, 2 y 27 se eligió para mostrar las variaciones en los parámetros de interés, dado que presentó los cambios más apreciables.

Las Figura 2-4 y 2-5 ilustran las variaciones porcentuales en las magnitudes y ángulos de la tensión de barra de los resultados con *slack* distribuido en comparación con el *slack* concentrado.

Las Figuras 2-6 y 2-7 muestran las variaciones en los flujos de potencia activa y reactiva en las líneas de transmisión cuando se usa el modelo con *slack* distribuido, en comparación con el modelo de *slack* concentrado.

De los resultados se observó que:

1. Las magnitudes de tensión no varían con el modelo de barra *slack* usado, como se evidencia en la Figura 2-4.

2. Se presentan cambios apreciables en los ángulos de tensión (Figura 2-5), como el aumento del 137% en la barra 1 y la disminución del 303% en la barra 2, ambas pertenecientes al *slack* distribuido.

3. Variación del flujo de potencia activa en la mayoría de las líneas de transmisión del sistema de potencia, como lo muestra la Figura 2-6, sin importar si la línea conecta en alguno de sus extremos con alguna barra perteneciente al *slack* distribuido. Como ejemplo de lo anterior, se observó una disminución cercana al 40% en los flujos de potencia activa en la línea 6-28.

4. Variación del flujo de potencia reactiva en algunas líneas de transmisión del sistema de potencia, como lo muestra la Figura 2-7, sin importar si la línea conecta en alguno de sus extremos con alguna barra perteneciente al *slack* distribuido. Como ejemplo de lo anterior, se observó un aumento cercano al 20% en los flujos de potencia reactiva en la línea 25-27.

Se puede inferir que distribuir el *slack* entre múltiples barras de generación, tiene un impacto tanto en los ángulos de tensión nodal como en los flujos de potencia activa en las líneas de transmisión del sistema y que variables tales como la magnitud de la tensión nodal y los flujos de potencia reactiva no se ven afectados por el *slack* distribuido.

Figura 2-4: Variación porcentual en las magnitudes de las tensiones de barra con *slack* distribuido en comparación con el modelo con *slack* concentrado.

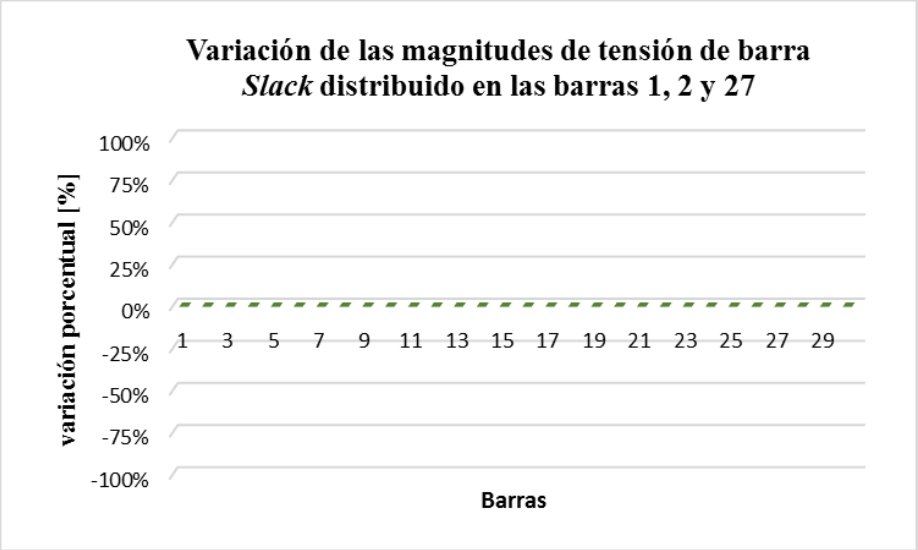


Figura 2-5: Variación porcentual en los ángulos de las tensiones de barra con *slack* distribuido en comparación con el modelo con *slack* concentrado.

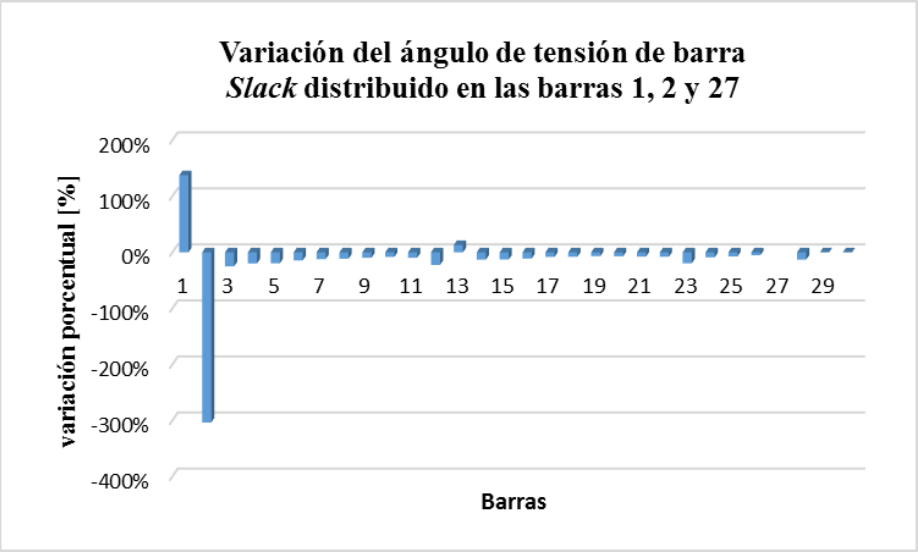


Figura 2-6: Variaciones en los flujos de potencia activa en las líneas de transmisión usando el modelo con *slack* distribuido en comparación con el modelo con *slack* concentrado.

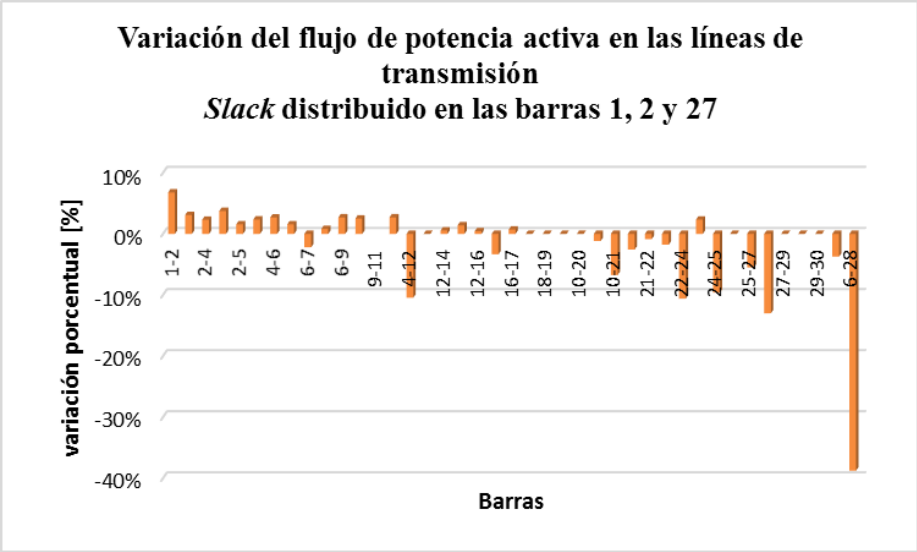
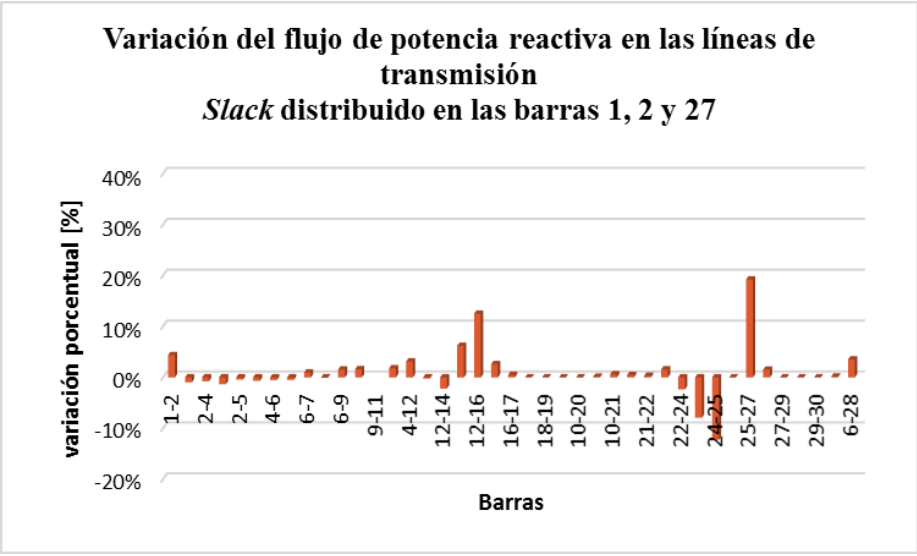


Figura 2-7: Variaciones en los flujos de potencia reactiva en las líneas de transmisión usando el modelo con *slack* distribuido en comparación con el modelo con *slack* concentrado.



2.5 Comentarios

La anterior representación matemática del problema de flujo de carga, en términos de la barra *slack* distribuida, será usada en la formulación para el análisis de contingencias. Él modelo con barra *slack* distribuida refleja de forma mucho más realista la operación del sistema que el modelo con barra *slack* única.

Ante la desconexión de una gran unidad de generación, es ilógico suponer que una sola barra de generación tenga la capacidad de asumir el total de la generación de dicha unidad, adicional a la potencia que ésta ya debía producir. En su lugar, este déficit es cubierto por los generadores del sistema que puedan ajustar su salida de potencia activa, como en efecto sucede en la realidad, teniendo cuidado de no exceder sus límites operacionales.

De acuerdo con la comparación entre los modelos con *slack* concentrado y *slack* distribuido realizado, se pudo establecer que éste último tiene un mayor impacto sobre los ángulos de las tensiones nodales y los flujos de potencia activa en las líneas de transmisión del sistema eléctrico de potencia.

3. Análisis masivo de contingencias usando cómputo paralelo

Una revisión desde la década de los 90 de las aplicaciones y las tendencias del cómputo paralelo en varios estudios de sistemas de potencia, incluyendo el análisis de contingencias, se presenta en [33]. Las tendencias generales encontradas fueron clasificadas en dos categorías; la primera relacionada con la forma en la que un problema específico puede ser descompuesto, paralelizado e implementado y la segunda relacionada con las herramientas de software y hardware usado para resolver problemas de altas dimensiones. Las tendencias computacionales destacadas comprenden el uso de redes de cómputo, equipos multinúcleo y GPU.

Igualmente, en [34] se exponen las preferencias en hardware y software para cómputo de alto desempeño observadas en sistemas de potencia, resaltando la tendencia hacia los computadores multinúcleo y la necesidad de cómputo paralelo para usarlos. Las aplicaciones para estimación de estado y análisis de contingencias fueron desarrolladas con cómputo de alto desempeño, mostrando para estas aplicaciones una mejora en su rendimiento.

Algunos ejemplos de la aplicación de cómputo paralelo en análisis de contingencias, combinando software, hardware y algoritmos matemáticos para su solución, serán expuestos a continuación. Adicionalmente, en este capítulo se realiza una introducción a las herramientas de hardware y software empleadas en el desarrollo de la aplicación propuesta para análisis de contingencias en paralelo. Un glosario de términos de cómputo paralelo es presentado en el Anexo A.

3.1 Panorama general de las implementaciones para análisis de contingencias basadas en cómputo paralelo

Un método para análisis completo de seguridad usando cuatro estaciones de trabajo IBM fue llevado a cabo en [35]. La lista con los casos de contingencia (simple y múltiple), fue sometida a un pre-filtrado basado en el índice de desempeño; sólo los casos con violaciones en límites operacionales, fueron sometidos a un procedimiento de evaluación de contingencia en paralelo, a través del flujo

de potencia desacoplado rápido. En la prueba se emplearon dos sistemas brasileros de 371 y 1663 barras, encontrando aceleraciones cercanas a 2,5 para el primero y 3,5 para el segundo.

El análisis de contingencias usando MPI (*Message Passing Interface*) y PVM (*Parallel Virtual Machine*), en cinco microcomputadores Pentium IV, de 2,3 GHz, y 532 MB de RAM, fue realizado en [37]. El algoritmo implementado, tanto en forma serial como paralela, fue validado en dos sistemas eléctricos brasileros de 486 barras / 533 líneas y 810 barras / 1340 líneas, para contingencias simples y múltiples. Los resultados mostraron aceleraciones cercanas a 3 y a 4 respectivamente.

Un estudio de análisis de contingencia para desconexión de líneas, resuelto por el método de evolución diferencial (método estocástico derivado de algoritmos genéticos), en un *cluster* con 37 nodos de cómputo, con 2 CPU (3,40 GHz y 2 GB de memoria) por nodo, es presentado en [38]. MATPOWER y *Parallel Computing Toolbox* de MATLAB® fueron usados como herramientas para la simulación de 403 casos de contingencia en el sistema de prueba IEEE 300 barras. Se observó que con 40 procesadores se consiguió una aceleración relativa de 25.

En [39] se presenta un método para análisis de contingencias usando MPI, en un grupo Linux de alto desempeño, con cerca de 2048 nodos de cómputo con procesadores Intel Xeon E5472 (de 2 y 4 núcleos) y 3,0 GHz, 16 GB RAM y 250 GB de disco duro. Los sistemas de prueba IEEE de 14, 30, 118, 162 y 300 barras, fueron considerados para análisis de desconexiones de línea simple. Cada análisis usó un modelo de flujo de potencia AC completo para el cálculo posterior del índice de desempeño. Se reportaron aceleraciones de 2,25, 4,22 y 25,82 para los sistemas de 118, 162 y 300 barras, y desempeño desacelerado para sistemas pequeños como el de 14 y 30 barras (0,04 y 0,35 respectivamente).

Como lo señala [34], el desafío en la aplicación de estudios para análisis de contingencias en paralelo está en el balance óptimo del tiempo computacional entre todos los procesadores, conocido como balance de carga computacional. Este tema también ha sido estudiado, como se verá a continuación.

Una aproximación computacional para análisis de contingencias N-k, usando los 8192 procesadores de la arquitectura multiproceso Cray XMT, y un grupo Linux de 192 nodos de cómputo, con 8 núcleos por nodo, fue reportado en [40]. Un algoritmo de trabajo compartido, que localiza dinámicamente un número seleccionado de contingencias en cada nodo esclavo, basado en la

disponibilidad de cada procesador, fue probado usando entre 200 y 17000 contingencias. Para la solución iterativa de las ecuaciones usaron el método de Newton.

En [36] se implementó el análisis de contingencias basado en el estándar OpenMP (*Open Multi Processing*), empleando dos configuraciones de hardware: un sistema HP Core i7 con 4 núcleos, 6GB de RAM y 750GB de disco duro, y un equipo IBM con 8 núcleos y 16GB de RAM. Se ejecutaron contingencias N-1, teniendo en cuenta la asignación dinámica y estática de tareas, en dos sistemas de potencia reales de 61177 barras/ 78600 líneas y un sistema WECC (*Western Electricity Coordinating Council*) de 21000 barras/ 23000 líneas.

En [42] se presenta un estudio sobre esquemas para balance de carga computacional en análisis masivo de contingencias. Estos incluyeron esquemas de balance de carga estático y dinámico, basados en un solo contador de tareas. Se alcanzaron aceleraciones superiores a 500 veces, usando el sistema de prueba WECC de 14000 barras y contingencias N-1 y N-2, en computadores paralelos con 512 procesadores, usando MPI y el método de Newton para la solución del flujo de potencia.

Debido a que la escalabilidad de estos esquemas dinámicos de balance de carga, con un gran número de procesadores, puede estar limitada por la congestión del contador, un esquema de contador múltiple desarrollado en un sistema con 10240 núcleos fue propuesto en [43].

3.2 Algunas plataformas de cómputo paralelo

Un sistema de cómputo paralelo puede estar conformado por una sola máquina con múltiples procesadores (y múltiples núcleos), o por un conjunto de computadores conectados mediante una red de comunicación dedicada, conocido como *cluster*.

Dichos sistemas, están determinados por unos componentes básicos característicos para cómputo de alto desempeño, tales como: elementos de procesamiento (multinúcleo y multiprocesador), memoria (compartida o distribuida), red o interconexión, y almacenamiento. Estos a su vez, se usan para representar la capacidad de un sistema de cómputo paralelo, haciendo referencia al número de nodos de cómputo, el número de procesadores disponibles con la cantidad de núcleos correspondientes, y la memoria total disponible en el sistema.

Como ejemplo, un computador multinúcleo de alta gama generalmente está compuesto por 8 núcleos o más situados en una tarjeta madre tipo servidor con múltiples *sockets* (típicamente dos), y usa módulos de memoria de decenas de GB (entre 32 y 64 GB) [34].

El *cluster* consistente en un grupo de computadores independientes e interconectados, trabajando juntos como un solo recurso de cómputo integrado [68]. El *cluster* es una de las herramientas computacionales preferidas, en virtud a que los computadores usados pueden mejorar fácilmente su capacidad agregando memoria o procesadores adicionales, es factible integrar nuevos equipos al *cluster* y el ancho de banda de las comunicaciones también se puede incrementar.

Los computadores o estaciones de trabajo que hacen parte del *cluster* se conocen como nodos. Un nodo puede tener uno o múltiples procesadores y cada procesador varios núcleos, así como memoria, dispositivos de entrada/salida y sistema operativo independientes, y están conectados a través de una red de área local (LAN). El código principal de la aplicación corre en uno de tales nodos, configurado como el "nodo principal" (*node head*). Los demás nodos esperan mientras el "nodo principal" les asigna trabajo, lo ejecutan y regresan a éste los resultados, de tal forma que cada uno trabaja en la misma tarea individual pero de forma paralela.

Para que un *cluster* funcione, es necesario que los computadores estén conectados y comunicados a través de redes de alta velocidad, responsables de transmitir paquetes de datos entre los nodos del sistema. Estas redes operan bajo estándares como Ethernet, Fast Ethernet, Gigabit Ethernet, Myrinet, Infiniband, SCI, entre otras. Adicionalmente, la red puede tener topología multiruta con enrutamiento estático y directo como en los *clusters Beowulf*.

En el mismo sentido, se requiere que los nodos tengan una imagen unificada del sistema, lo que es provisto por el *middleware* del *cluster*. El *middleware* es una capa que reside entre el sistema operativo y las aplicaciones, e incluye archivos del sistema del *cluster*, ambientes de programación, administrador de trabajos y sistemas de programación. Para administración del sistema y propósitos de programación de trabajo, el *middleware* corre "demonios" o servicios en cada nodo suministrando la información requerida a la herramienta de administración o al programador de trabajos [69].

Típicamente, los *clusters* son homogéneos, es decir, cuentan con arquitecturas similares y tienen el mismo sistema operativo. Sin embargo, también se puede formar un *cluster* a partir de equipos con

diferentes arquitecturas y/o sistemas operativos. Esto quiere decir que se puede combinar el uso de servidores, computadores de escritorio y equipos portátiles para formar un *cluster*. Aunque esta solución es aparentemente práctica, impone una dificultad en el balance de carga computacional [68].

El balance de carga es una configuración que consiste en la distribución de la carga de trabajo entre los diferentes recursos, con el fin de optimizar su uso, evitar la sobrecarga de algunos de ellos, y minimizar el tiempo de respuesta, mejorando de esta forma el desempeño total de la aplicación. Existen dos tipos de balance de carga, el estático en el que se distribuye el trabajo entre los procesadores antes de la ejecución del programa; y el dinámico en el que se distribuye el trabajo entre los procesadores que estén ejecutando poco o ningún trabajo, durante la ejecución del programa.

Por otro lado, para construir programas de aplicación en paralelo existen herramientas de programación que proveen lenguajes, librerías y depuradores de corrección y desempeño. Entre estos entornos de programación de aplicaciones, dos paradigmas han sido ampliamente usados.

El primero, la interface PVM, provee un conjunto de librerías que permiten ver el nodo como una máquina virtual, facilitando un ambiente para la transferencia de mensajes, administración de tareas y recursos, y notificación de fallas [68]. El segundo, la interface MPI, no es un lenguaje de programación completo pero si una librería aumentada que permite a los usuarios de C y Fortran acceder librerías para transferencia de mensajes entre procesos concurrentes en nodos de procesadores separados pero interconectados [69].

3.3 Programación paralela en MATLAB® [70]

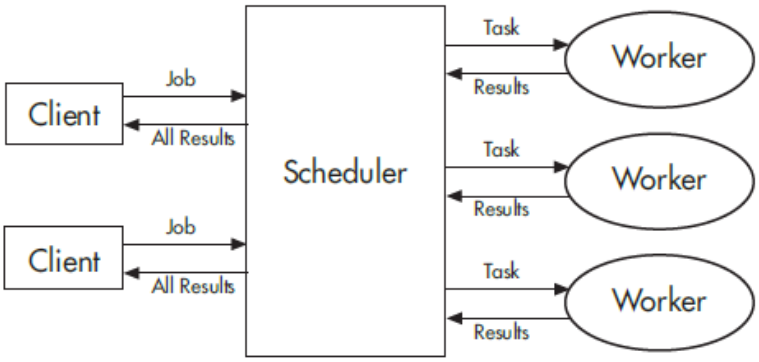
MATLAB® dispone de dos *toolbox* para ejecutar trabajo paralelo en computadores multinúcleo, GPUs y *cluster*: *Parallel Computing Toolbox* (PCT) y *Distributed Computer Server* (DCS).

Un trabajo (*job*) es alguna gran operación que necesita realizarse en la sesión de MATLAB®. Éste es dividido en segmentos llamados tareas, las cuales pueden o no ser idénticas. La sesión en la que el trabajo y sus tareas son definidos es llamada la sesión cliente.

El planificador de trabajo MJS (MATLAB® *Job Scheduler*) es el proceso que coordina la ejecución de los trabajos y la evaluación de sus tareas. El MJS distribuye las tareas para su evaluación en los trabajadores (*workers)*.

Un trabajador es una sesión de MATLAB® en la que se realiza la tarea computacional. El *toolbox* PTC provee a la sesión cliente con un grupo de trabajadores locales, mientras el *toolbox* DCS es indispensable para contar con múltiples trabajadores en un *cluster* remoto. La interacción entre sesiones en cómputo paralelo es ilustrada en la Figura 3-1.

Figura 3-1: Interacción entre sesiones en cómputo paralelo. Fuente: **[70]**



Una gran red puede incluir muchos MJS así como muchas sesiones cliente. Cualquier sesión cliente puede crear, correr e ingresar trabajo en cualquier MJS, pero un trabajador es registrado con y dedicado a solamente un MJS al tiempo.

El conjunto de trabajadores, tanto locales como remotos, reservados para correr un trabajo en paralelo es llamado un p*arallel pool*. Éste conjunto permite, de ser necesario, la comunicación entre los trabajadores y tiene el tiempo de vida del trabajo que esté ejecutando. Sólo se puede tener acceso a un único *parallel pool* a la vez en la sesión cliente.

Los comandos provistos por estos *toolbox* para la ejecución de trabajo en paralelo incluyen, lazos *for* paralelos (*parfor*), un programa múltiples datos (*spmd*), y programación multitarea.

El comando *spmd,* se ha elegido para el desarrollo de la aplicación de que trata este trabajo, ya que por sus características, se adapta a los requerimientos y particularidades del análisis de contingencias.

3.3.1 *Single program multiple data* (spmd)

El comando *spmd* (*single program multiple data*), permite intercalar sin problemas programación serie y paralela. Éste es usado cuando se requiere comunicación o sincronización entre los trabajadores. Todos los trabajadores ejecutando una sentencia *spmd*, son conscientes unos de otros, lo que permite controlar directamente la comunicación y la transferencia de datos entre ellos.

Un bloque idéntico de código (*single program*) se ejecuta en todos los trabajadores de un *pool*, cada uno de los cuales puede acceder a un conjunto diferente de datos (*multiple data*). Los trabajadores usados dentro de una sentencia *spmd* tienen un único valor de identificación conocido como *labindex***,** el cual permite correr código específicamente en ciertos trabajadores.

3.3.2 Funciones de transferencia de mensajes

Aunque en MATLAB® la transferencia de mensajes esté basada en MPI [71], el usuario no tiene que implementar éstas especificaciones directamente, en su lugar usa abstracciones de funciones de alto nivel equivalentes a algunas funciones en MPI.

Para la comunicación entre los diferentes trabajadores, se disponen de instrucciones que permiten: enviar datos a un trabajador (*labSend*), recibir datos de un trabajador (*labReceive*), enviar y recibir datos simultáneamente (*labSendReceive*), comprobar si un mensaje está listo para ser recibido (*labProbe*), sincronizar la transferencia de mensajes para temporizar (*labBarrier*), consultar en el entorno la identificación del trabajador (*labindex*) y consultar la cantidad de trabajadores que componen el *cluster* (*numlab*), entre otros.

4. Implementación y resultados del algoritmo para el cómputo en paralelo del análisis de contingencias con *slack* distribuido

La aplicación para análisis de contingencias en paralelo se diseñó con base en el paradigma de programación maestro/esclavo, donde un trabajador es designado como el maestro, y los demás trabajadores como esclavos.

El trabajador maestro tiene por funciones: determinar el estado de las tareas (no enviado, pendiente por recibir datos, terminada), distribuir las tareas entre los trabajadores esclavos, revisar continuamente el estado de estos (disponible, envío de resultados, ocupado), recibir los resultados enviados por los trabajadores esclavos, y controlar los criterios de finalización del proceso. Por su parte, los trabajadores esclavos tienen como funciones ejecutar cada tarea correspondiente a un flujo de carga completo y regresar al trabajador maestro los resultados obtenidos.

Complementariamente, se empleó un esquema de asignación dinámica de contingencias, desarrollado bajo tres estrategias: una basada en la disponibilidad de cada trabajador, otra en la priorización de tareas y una más priorizando tareas y trabajadores. La asignación dinámica de tareas en cómputo paralelo tiene como finalidad el uso óptimo del tiempo computacional de todos los procesadores.

4.1 Diseño de la aplicación para cómputo paralelo

Las funciones correspondientes al cálculo del caso base, clasificación de las tareas y de los trabajadores, la etapa de post-procesamiento y la presentación de los resultados, se realizan en la sesión de trabajo principal. El programa para análisis de contingencias con *slack* distribuido se puede consultar en el Anexo B y un ejemplo de la presentación de los resultados de dicho programa se muestra en el Anexo C.

Cada uno de los núcleos de los procesadores, tanto del computador multinúcleo como de cada computador en el *cluster*, son tratados como un trabajador por el PCT de MATLAB®.

El cálculo de cada caso de contingencia, se efectúa en paralelo en cada uno de los trabajadores según las tres estrategias de asignación dinámica de carga computacional, descritos a continuación.

4.1.1 Estrategia A: Asignación de contingencias basada en la disponibilidad de los trabajadores

El trabajador maestro monitorea el estado de las tareas por ejecutar. Si encuentra alguna cuyo estado sea "no enviado", revisa el estado de los trabajadores esclavos para determinar cuál se encuentra "disponible", y entonces asignarle el número de la tarea, cambiando su estado a "ocupado".

En caso contrario, el trabajador maestro verifica si el estado del trabajador esclavo es "envío de resultados", indicando que éste ya ejecutó una tarea y ha enviado resultados, procediendo a recibirlos y cambiar su estado a "disponible". La Figura 4-1 presenta la estrategia descrita.

4.1.2 Estrategia B: Asignación de contingencias basada en la priorización de tareas

La lista de tareas se ordena de forma descendente según el tiempo promedio que cada una consume (para diez mediciones de tiempo de ejecución), tomando como base los tiempos medidos previamente durante la ejecución serial del programa para análisis de contingencias diseñado. De esta forma, las contingencias que consume más tiempo se ubican en las primeras posiciones de la lista. Por lo demás, el procedimiento lógico seguido es similar al de la estrategia A, representada por la Figura 4-1.

4.1.3 Estrategia C: Asignación de contingencias basada en la priorización de tareas y trabajadores

Además de clasificar y ordenar las tareas como en la estrategia B, se efectúa una clasificación de los trabajadores según su velocidad de procesamiento.

El trabajador maestro monitorea el estado de las tareas por ejecutar. Si encuentra alguna cuyo estado sea "no enviado", revisa el estado de los trabajadores esclavos para determinar cuál se encuentra

"disponible". Luego determina si la tarea está entre las que consumen más tiempo y se identifica sí el trabajador "disponible" está entre los más "veloces"; de tal forma que la tarea que consume más tiempo sea asignada a un trabajador "veloz", y la tarea que consume menos tiempo a un trabajador "menos veloz". El número de la tarea se asigna al trabajador y su estado cambia a "ocupado".

En caso contrario, el trabajador maestro verifica si el estado del trabajador esclavo es "envío de resultados", indicando que éste ya ejecutó una tarea y ha enviado resultados, procediendo a recibirlos y cambiar su estado a "disponible". La Figura 4-2 presenta la estrategia descrita.

Figura 4-1: Estrategia A para el balance de carga computacional

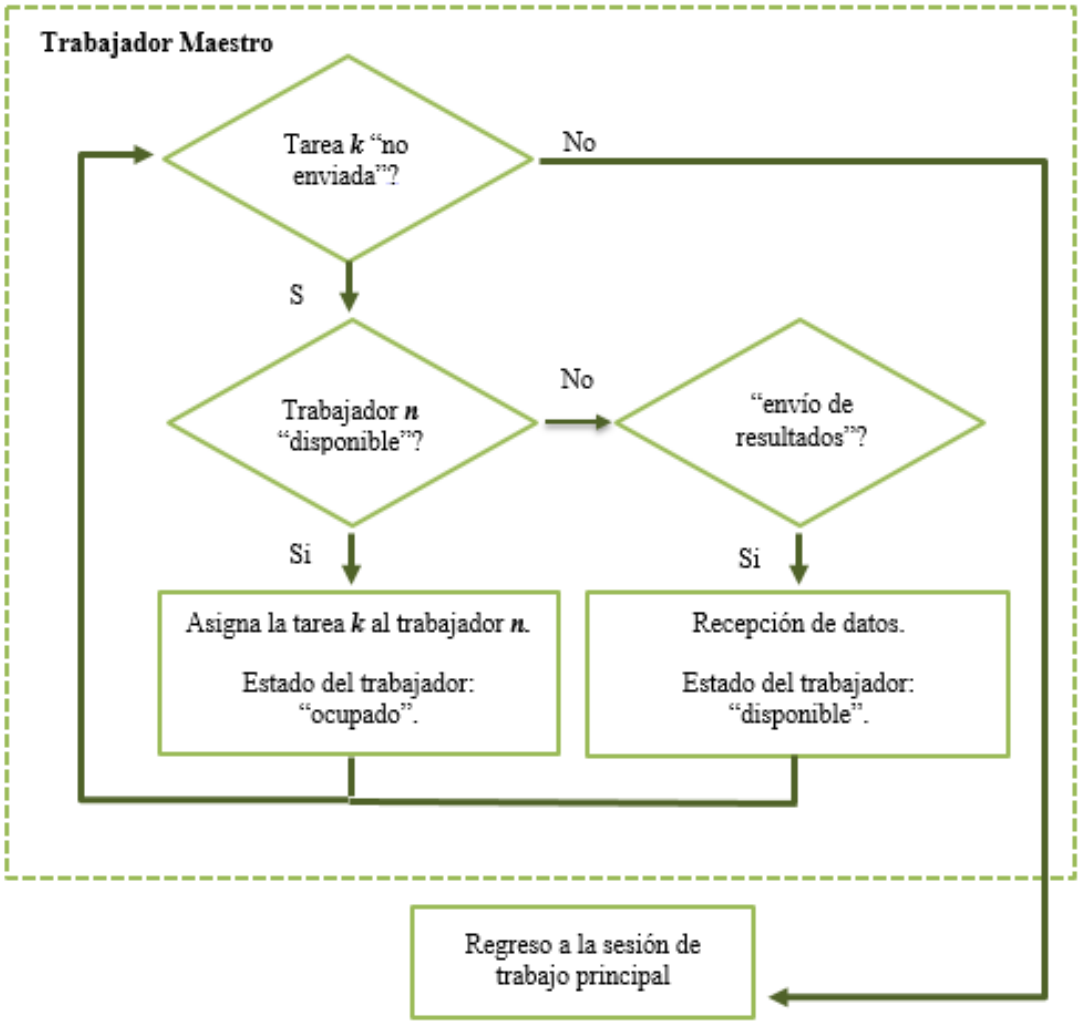
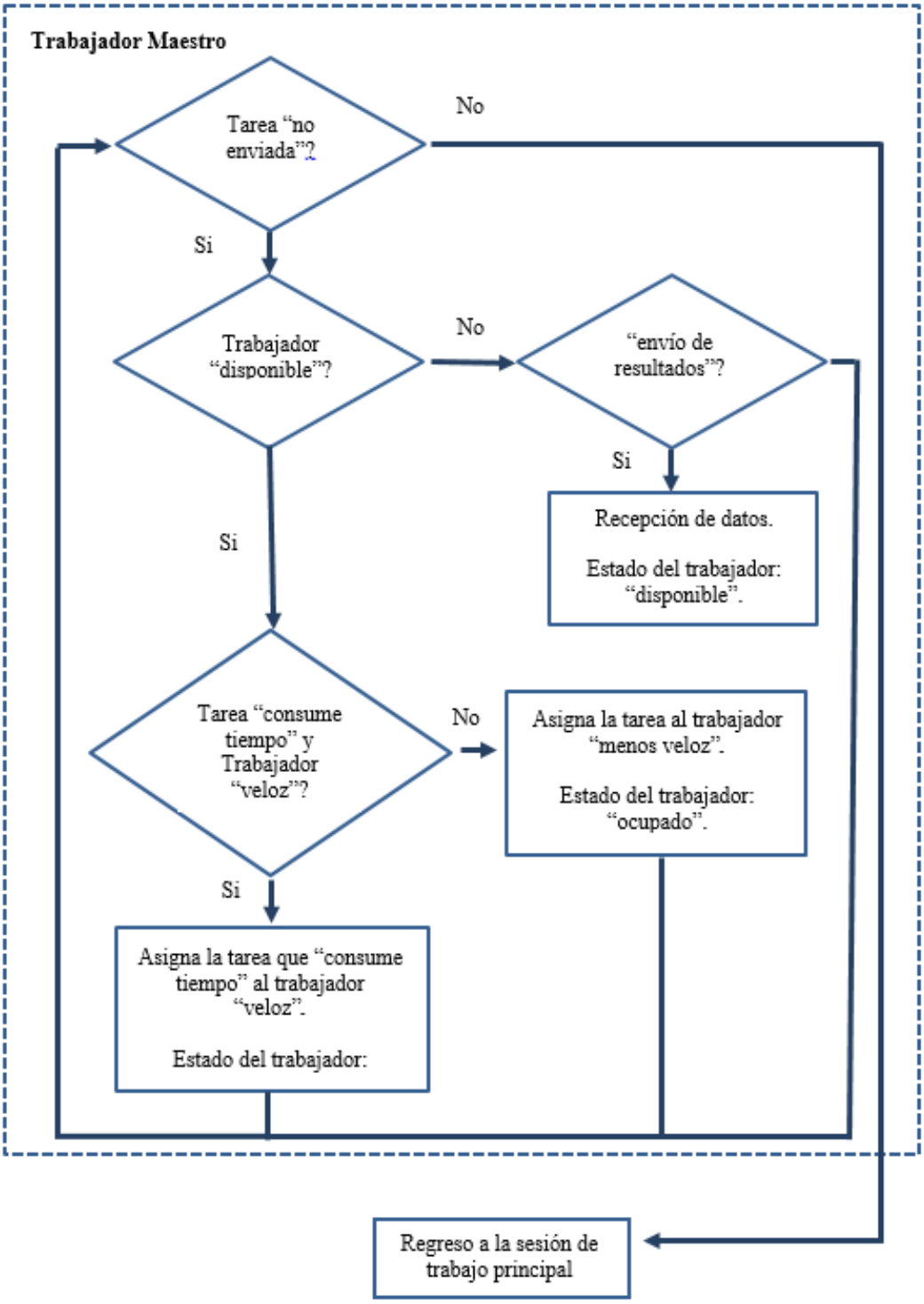


Figura 4-2: Estrategia C para el balance de carga computacional



4.1.4 Especificaciones del hardware y software empleados

Se empleó un *cluster* heterogéneo de 6 nodos con 12 núcleos (trabajadores), y un computador multinúcleo con 12 núcleos (trabajadores), cuyas características correspondientes son relacionadas en la Tabla 4-1 y Tabla 4-2 respectivamente.

La comunicación de los computadores en el *cluster* se realiza a través de una red de área local de 1Gbps, bajo el estándar Ethernet.

Tabla 4-1: Características del *cluster* heterogéneo

NODO	PROCESADOR	MEMORIA	DISCO DURO	SISTEMA OPERATIVO
1 (Head)	Core i7- 3770, 3,4 GHz, cuatro núcleos	4GB	2TB	Windows 7
2	Core 2 Duo, 2,10 GHz, dos núcleos	4GB	500GB	Windows 7
3	Core 2 Duo, 1,83 GHz, dos núcleos	4GB	500GB	Windows XP
4	Core i7, dos núcleos en máquina virtual	2GB	500GB	Windows XP
5	Centrino, 1,8 GHz, un núcleo	2GB	320GB	Windows XP
6	Centrino, 1,8 GHz, un núcleo	2GB	160GB	Windows XP

Tabla 4-2: Características del computador multinúcleo

PROCESADOR	MEMORIA	DISCO DURO	SISTEMA OPERATIVO
2 x Intel Xeon XE5, seis núcleos c/u, 2,93 GHz	64GB	8TB	OSX

Para la implementación del algoritmo se usó MATLAB (versión R2013a 32 bits en Windows y 64 bits en OSX), MATPOWER [72] (versión 5.0b1), y las herramientas para cómputo paralelo *Parallel Computing Toolbox* y *Distributed Computer Server* de MATLAB.

4.2 Resultados de la Simulación

Con el fin de estudiar el desempeño del algoritmo para análisis de contingencias en paralelo con *slack* distribuido, se efectuaron pruebas en cuatro sistemas de potencia de diferentes tamaños, empleando las tres estrategias de asignación dinámica de tareas expuestos en el apartado anterior. Las características de los sistemas bajo prueba se presentan en la Tabla 4-3.

Tabla 4-3: Características de los sistemas de potencia objeto de prueba

No. Barras	No. Contingencias	
	Generadores	Líneas
30	3	29
118	47	149
300	62	385
2383	320	2361

Se analizaron contingencias simples de desconexión de línea o de generador, estableciendo como criterios para la definición de los casos de contingencia a estudiar: la desconexión de líneas que no aíslen los generadores bajo AGC, la desconexión de líneas que no aíslen algún nodo del sistema de potencia, y la desconexión de generadores que no participen en AGC. Estos criterios tienen como finalidad evitar la eliminación de algún generador del sistema de potencia participando en AGC, o tener sistemas en isla.

El desempeño de la aplicación en paralelo se evalúa en términos de la velocidad, calculada como la inversa del tiempo de ejecución en paralelo. Para el cálculo de la velocidad, se consideraron solo los tiempos máximo, promedio y mínimo, debido a que corresponden a los casos más extremos (máximo y mínimo) y al caso más general (promedio). Por lo tanto, cada figura representará la velocidad calculada a partir de estos tiempos. En el Anexo D se encuentran las tablas de soporte de dichas figuras. Las pruebas de efectividad se realizaron en cada uno de los cuatro sistemas de prueba, variando el número de procesadores usados, en ambas plataformas de cómputo.

En los casos con buen desempeño, definidos por sus características de velocidad creciente con el número de procesadores usados, se calculó la aceleración máxima obtenida. La aceleración se calcula

como la relación entre el tiempo serial y el tiempo en paralelo, ambos tiempos calculados como el promedio de diez mediciones de tiempo.

4.2.1 Evaluación de la implementación en el *cluster*

Las pruebas se realizaron, en cada uno de los cuatro sistemas de potencia, y para cada uno de las tres estrategias de asignación de contingencias, variando el número de trabajadores. En las Figuras 4-3 a 4-5, se presentan las velocidades máxima, mínima y promedio, obtenidos para las estrategias A, B y C respectivamente, y en la Figura 4-6 se presentan las velocidades promedio de cada estrategia en cada sistema de prueba.

Figura 4-3: **Resultados del *cluster*. Número de trabajadores *vs* Velocidad – Estrategia A**

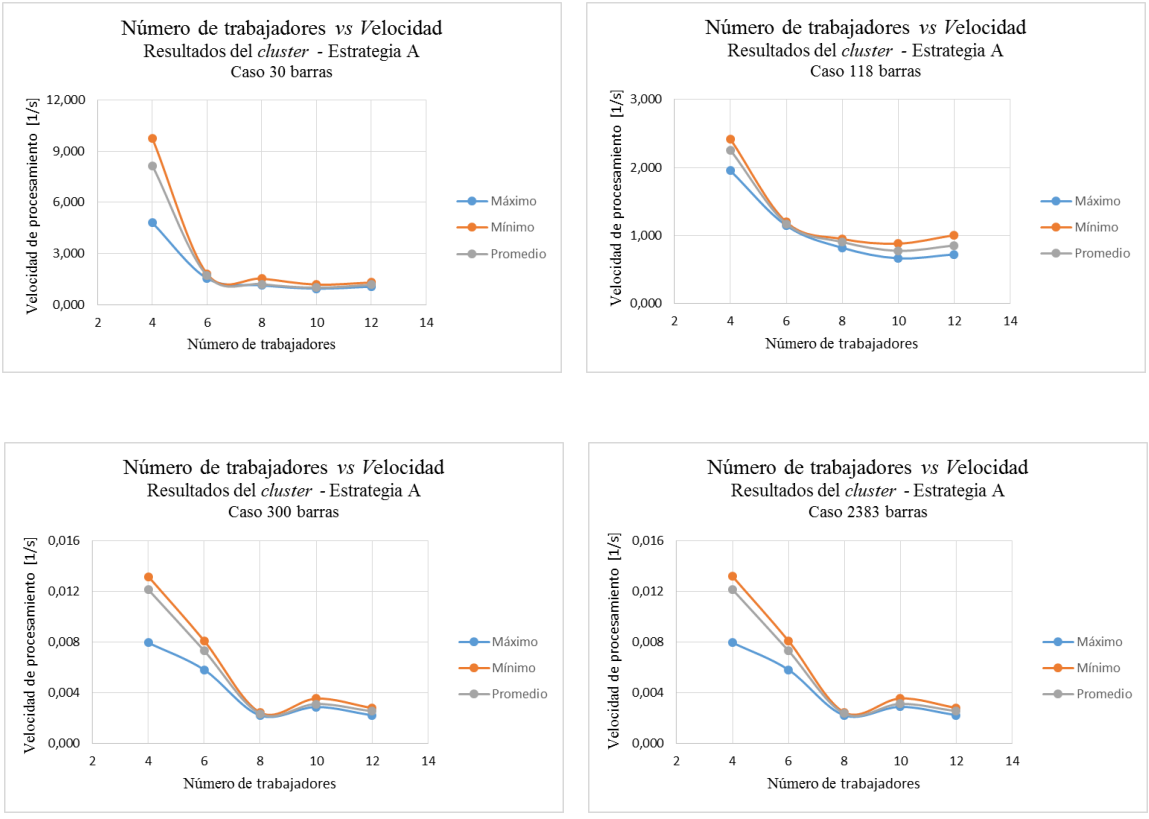


Figura 4-4: Resultados del *cluster*. Número de trabajadores *vs* Velocidad – Estrategia B

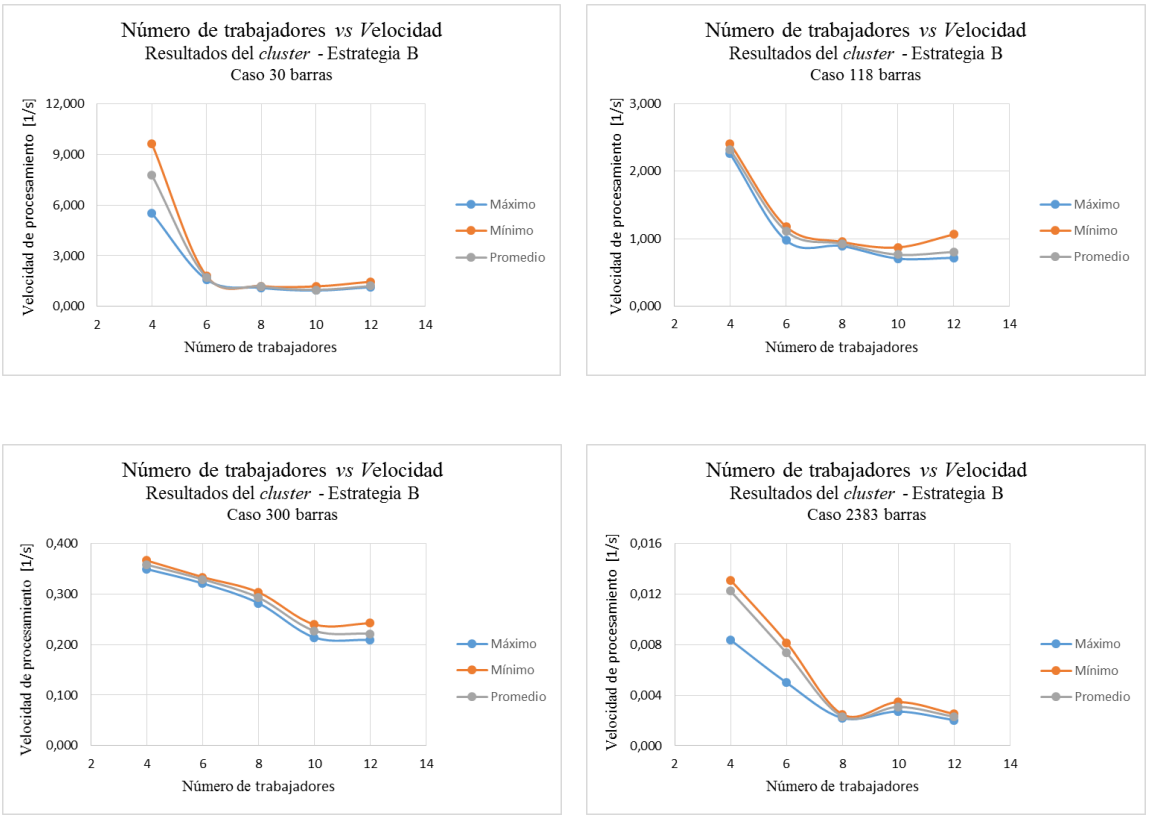


Figura 4-5: Resultados del *cluster*. Número de trabajadores *vs* Velocidad – Estrategia C

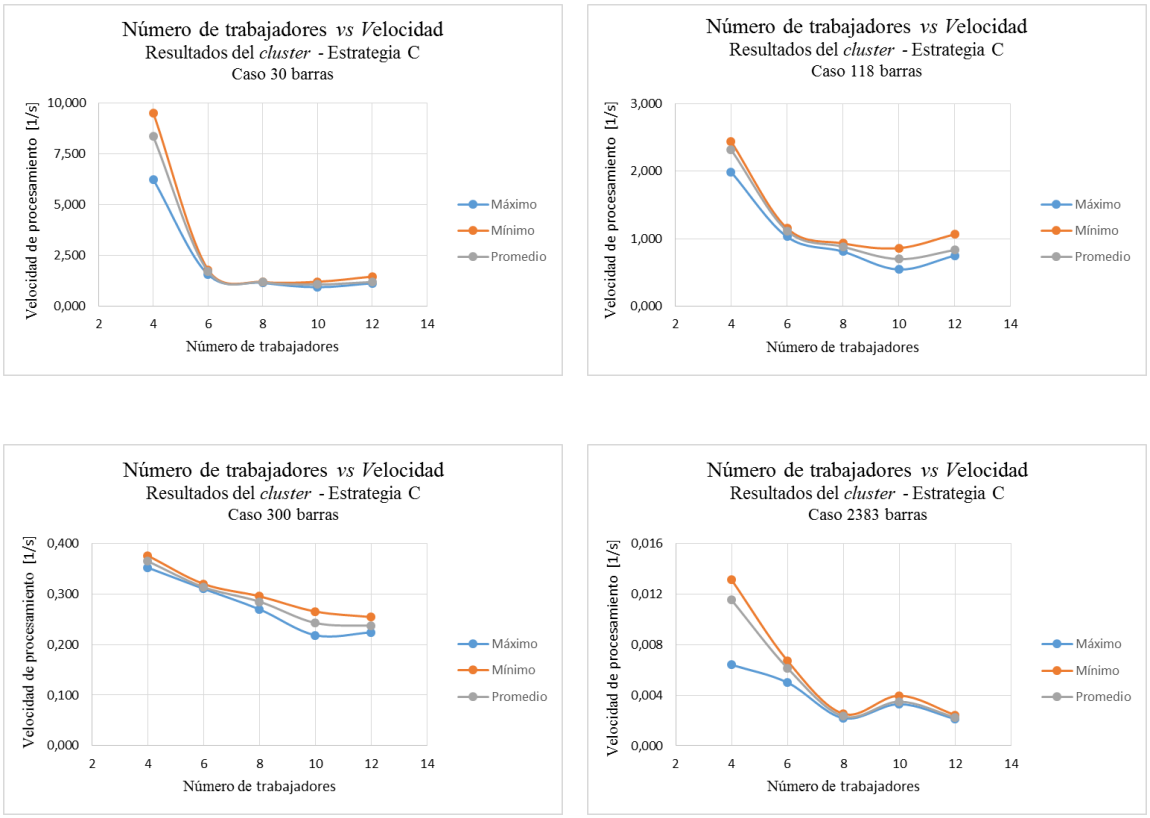
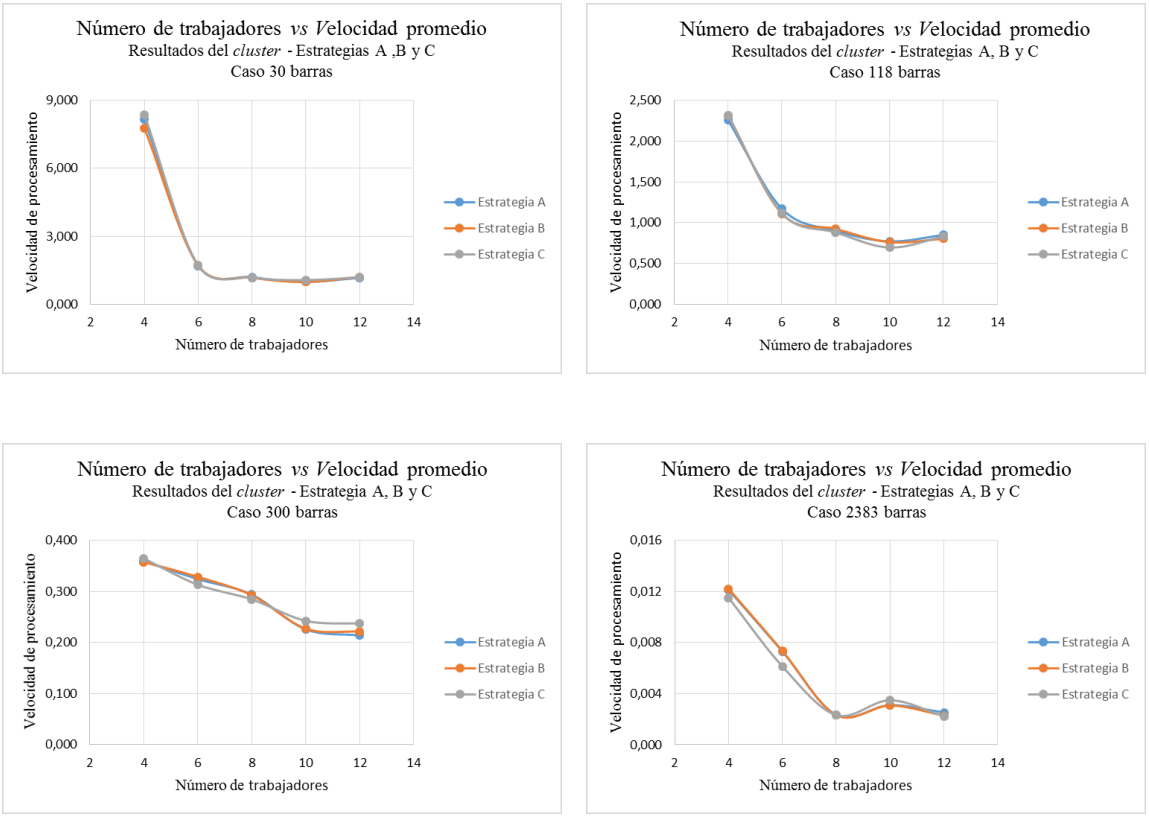


Figura 4-6: Resultados en el *cluster*. Número de trabajadores *vs* Velocidad promedio de cada estrategia



El desempeño del *cluster* está muy condicionado por las capacidades de comunicación y por la asimetría en la velocidad de los procesadores de cada equipo. Para este caso particular, la diferencia entre el equipo más rápido (*head*) y el equipo más lento del *cluster*, en cuanto a velocidad de procesamiento, es muy grande. Además, el mal desempeño de los nodos menos rápidos no es solo numérico sino también de comunicaciones.

Al correr el programa se observó que cuando se abre un *MatLabpool*, el MJS siempre asigna los nodos en orden decreciente de rapidez (numérica y de comunicación), es decir, los nodos más lentos son los primeros que se descartan. Entonces, cuando se indica el número de nodos con los que se desea trabajar, siempre se tomarán los más rápidos dentro del *cluster*. Esto hace que la adición marginal de capacidad de cómputo sea decreciente al incrementar los nodos, y posiblemente incluso el promedio de capacidad también sea decreciente al aumentar la cantidad de nodos del *cluster*.

En efecto, las Figuras 4-3 a 4-6 muestran que, en general, el desempeño disminuye a medida que se aumenta el número de trabajadores, independientemente de la estrategia de balance de carga computacional aplicada y del tamaño del sistema bajo prueba.

También se advierte la similitud en el comportamiento de las tres estrategias de asignación dinámica de carga computacional, asociada con la asignación de los nodos en orden decreciente de rapidez efectuada por el MJS, y a que el tiempo que consume cada tarea (contingencia) es muy similar.

De los resultados se puede concluir que no es recomendable trabajar con un *cluster* demasiado heterogéneo. Las ganancias obtenidas al aumentar la capacidad de procesamiento, son superadas por las limitaciones en comunicación que impactan de forma negativa el tiempo de cálculo total.

4.2.2 Evaluación de la implementación en el computador multinúcleo

Al igual que con el *cluster*, las pruebas se realizaron en cada uno de los cuatro sistemas de potencia, y para cada una de las tres estrategias de asignación de contingencias, variando en número de trabajadores. En las Figuras 4-7 a 4-9, se presentan las velocidades máxima, mínima y promedio, obtenidos para las estrategias A, B y C, y en la Figura 4-10 se presentan las velocidades promedio de cada estrategia en cada sistema de prueba.

Figura 4-7: Resultados del computador multinúcleo. Número de trabajadores *vs* Velocidad – Estrategia A

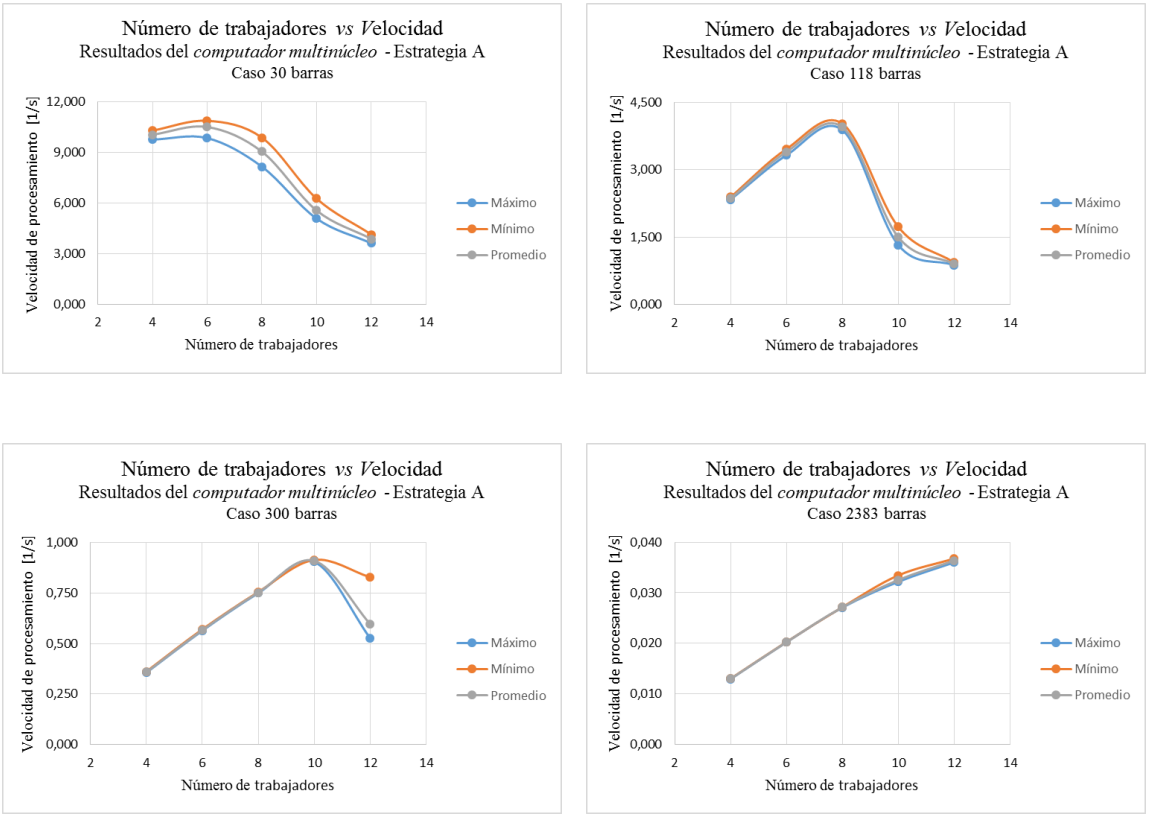
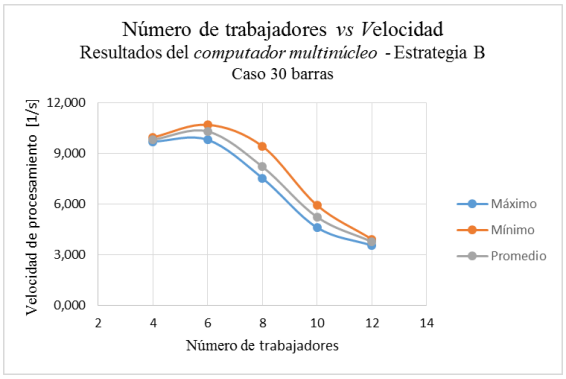
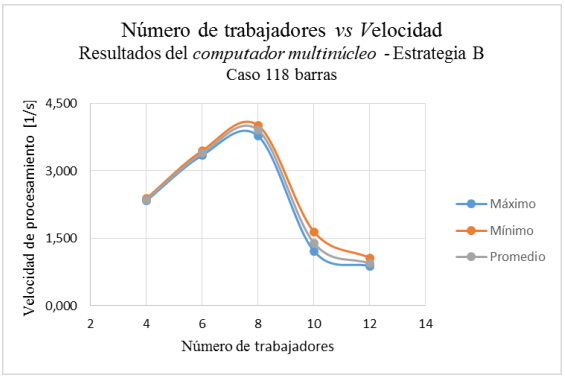
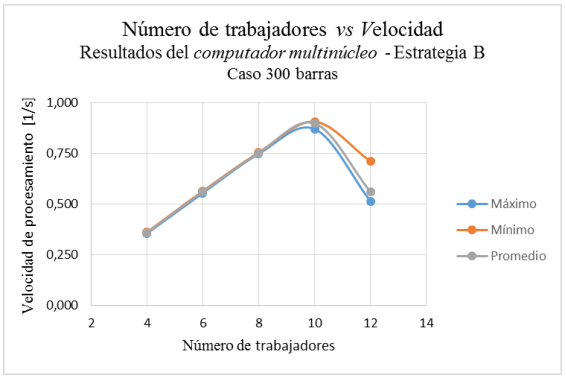


Figura 4-8: Resultados del computador multinúcleo. Número de trabajadores *vs* Velocidad – Estrategia B







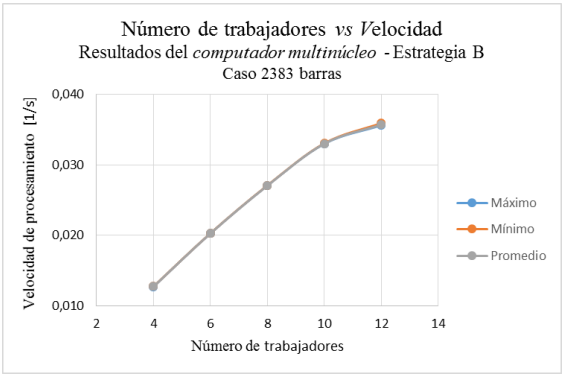


Figura 4-9: Resultados del computador multinúcleo. Número de trabajadores *vs* Velocidad – Estrategia C

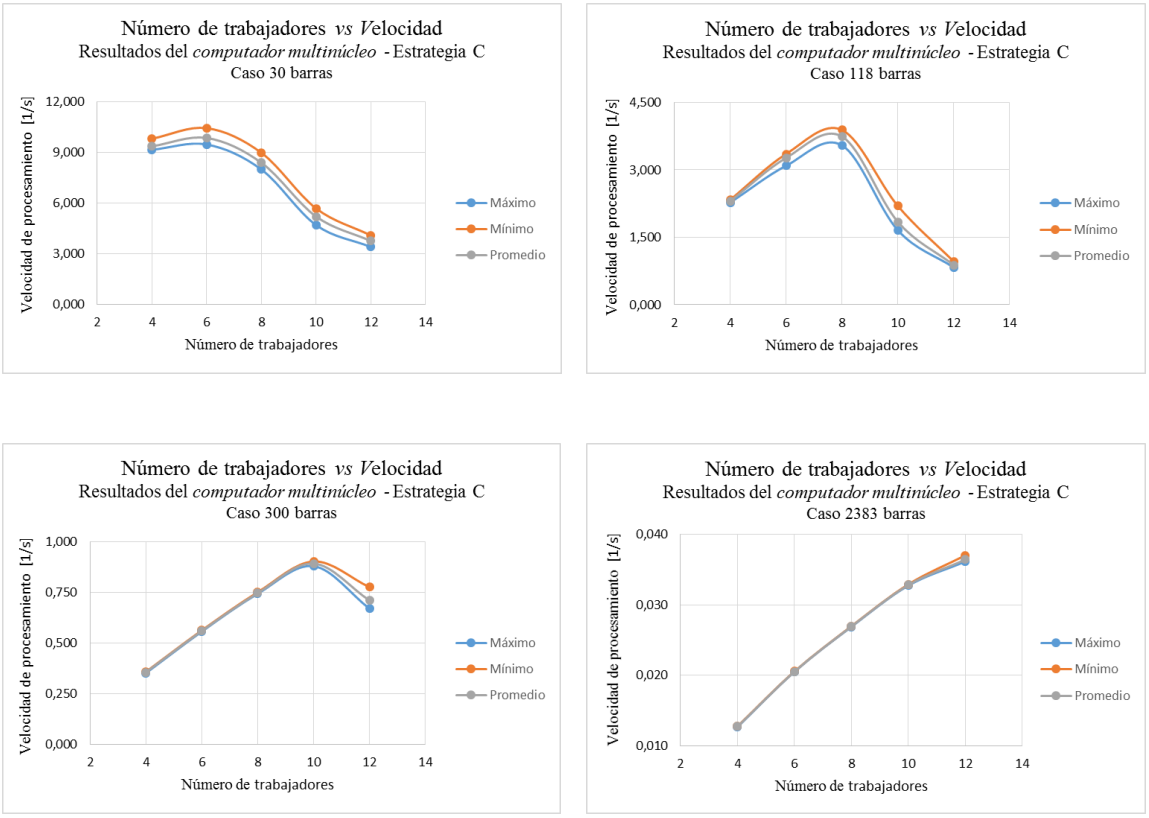
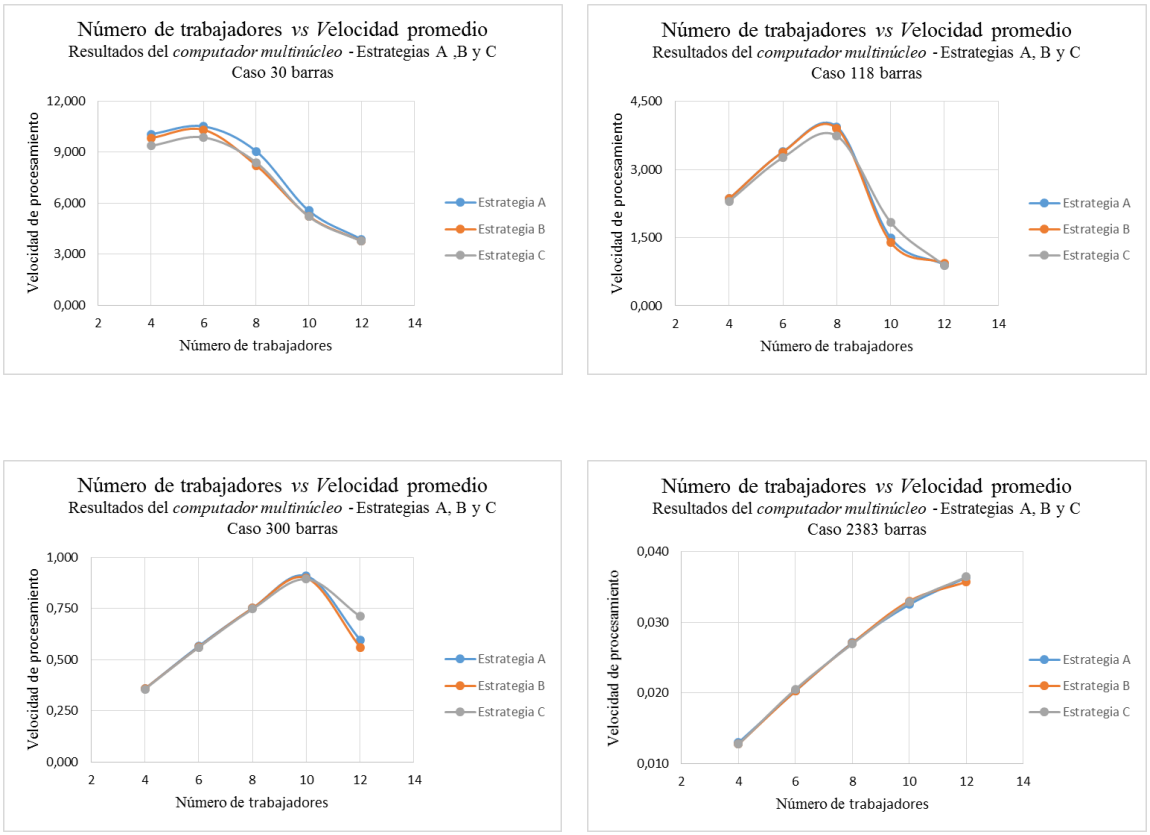


Figura 4-10: Resultados del computador multinúcleo. Número de trabajadores *vs* Velocidad promedio de cada estrategia



Las Figuras 4-7 a 4-10 muestran que para el sistema de 2383 barras, el de mayor tamaño, la velocidad de procesamiento tiende a aumentar con el número de trabajadores usados. Por el contrario, para sistemas pequeños, como en el caso de 30 barras, la velocidad de procesamiento tiende a disminuir a medida que se incrementa el número de procesadores.

Un comportamiento intermedio se observó en los sistemas de 118 y 300 barras, en los que inicialmente se presentó un crecimiento en la velocidad a medida que se aumentaba el número de trabajadores, alcanzando un punto de saturación luego del cual se experimentó un comportamiento desacelerado.

En cuanto a la aceleración, se observó que para todos los sistemas de prueba se consiguieron aceleraciones con respecto a su tiempo de ejecución serial, de acuerdo con el número de procesadores

usados. En la Tabla 4-4, se presentan las aceleraciones máximas alcanzadas para cada sistema eléctrico de prueba indicando el número de procesadores usados.

Tabla 4-4: Valores calculados de aceleración de cómputo obtenidos con el computador multinúcleo

Sistema de prueba	**Número de procesadores usados**	**Aceleración máx. obtenida por estrategia de asignación de tareas**		
		A	**B**	**C**
30 barras	6	1,09	1,10	1,04
118 barras	8	3,67	3,62	3,49
300 barras	10	6,95	6,89	6,81
2383 barras	12	7,82	7,69	7,75

De la Tabla 4-4 se puede inferir que para la aplicación en cómputo paralelo, la aceleración incrementa con el tamaño del caso de prueba, con un número de procesadores diferente para cada caso. Se percibe que el menor valor de aceleración (1,10) corresponde al sistema más pequeño, es decir, el de 30 barras, mientras que el mayor valor de aceleración (7,82) fue para el sistema de prueba más grande, de 2383 barras, coincidiendo con la tendencia en el comportamiento presentado por la velocidad de procesamiento.

Al igual que en las pruebas con el *cluster*, se observa que las tres estrategias de asignación dinámica de carga computacional proporcionaron resultados muy similares. Esto se debe a que las características de los procesadores son homogéneas y a que el tiempo que consume cada tarea (contingencia) es muy similar.

De lo anterior se puede establecer que el desempeño en el computador multinúcleo no depende tanto de las comunicaciones, pero sí de la competencia por el acceso a memoria, llegando a un punto en el que el desempeño cae al estar todos los núcleos tratando de acceder a memoria simultáneamente.

5. Conclusiones y trabajo futuro

5.1 Conclusiones

En este trabajo se presentó una extensión a la formulación y al esquema de solución del problema de análisis estático de contingencias de primer orden (N-1), incorporando el concepto de *slack* distribuido mediante AGC.

Con el fin de comparar los perfiles de tensión de las barras del sistema y los flujos de potencia en las líneas de transmisión para un sistema de prueba, al correr un flujo de carga con *slack* concentrado y un flujo de carga con *slack* distribuido, se eligió el sistema IEEE de 30 barras. Para este ejemplo, se pudo inferir que distribuir el *slack* entre múltiples barras de generación, tiene un impacto apreciable tanto en los ángulos de tensión nodal como en los flujos de potencia activa en las líneas de transmisión del sistema y que variables tales como la magnitud de la tensión nodal y los flujos de potencia reactiva no se ven afectados por el *slack* distribuido.

La formulación propuesta, con flujo de carga con *slack* distribuido por AGC, se implementó mediante herramientas de cómputo paralelo, y se diseñó con base en el paradigma de programación maestro/ esclavo y tres estrategias de asignación dinámica de tareas. La verificación del desempeño del programa se efectúo mediante un análisis de la velocidad de procesamiento, empleando dos plataformas de hardware: un *cluster* heterogéneo y un computador multinúcleo. Las pruebas se ejecutaron variando dos aspectos, que son el tamaño del sistema de potencia y el número de trabajadores en cada una de las dos opciones de hardware disponibles. Adicionalmente, en los casos con buen desempeño, definidos por sus características de velocidad creciente con el número de procesadores usados, se calculó la aceleración máxima obtenida.

Se observó un pobre desempeño de la aplicación en el *cluster* heterogéneo, debido a que está condicionado por las capacidades de comunicación y por la asimetría en la velocidad de cada equipo. Dado lo anterior, se pudo inferir que no es recomendable trabajar con un *cluster* demasiado

heterogéneo, puesto que la ganancia derivada del aumento de la capacidad de procesamiento se ve opacada por las limitaciones en comunicación que afectan el rendimiento total de la aplicación.

En cambio, los resultados para las pruebas con el computador multinúcleo, en la mayoría de los sistemas mostraron un comportamiento casi lineal en la velocidad hasta un determinado número de procesadores, mientras que en el sistema pequeño (30 barras) el desempeño fue siempre lento. La saturación en la curva velocidad *vs* número de procesadores observada en algunas pruebas, se debe a la competencia de los procesadores por el acceso a memoria. Dados los resultados de éstas pruebas, se calculó la aceleración de cómputo para cada caso de prueba encontrando aceleraciones máximas de 1,10 para el sistema de 30 barras, 3,67 para el sistema de 118 barras, 6,95 para el sistema de 300 barras y 7,82 para el sistema de 2383 barras, observando que la aceleración en la ejecución del análisis de contingencias tiene una relación directa con el tamaño del caso.

5.2 Principales contribuciones

- Se presentó la formulación de un método para flujo de potencia con barra *slack* distribuida mediante AGC, ilustrándola a través de un caso pequeño de forma que facilite su aplicación en casos con mayor número de barras.

- Se proporcionan estrategias y herramientas de cómputo aplicables al sector eléctrico, orientadas al mejoramiento de soluciones para estimación de la seguridad de un sistema eléctrico de potencia en tiempo real, gracias al procesamiento de gran cantidad de información a través del cómputo paralelo.

5.3 Trabajo futuro

- Extender la formulación matemática expuesta para el método de Newton a otros métodos como Newton Desacoplado, Newton Desacoplado Rápido, y DC.
- Considerar la aplicación del software desarrollado para el estudio de análisis de contingencias de orden superior (N-k).
- Explorar otros enfoques como el proporcionado por el flujo de potencia óptimo.
- Emplear GPUs masivamente paralelas para resolver el problema de análisis de contingencias con la formulación matemática expuesta, a través de un método de bajo consumo de memoria como el método de Gauss-Seidel.

A. Anexo: Glosario

Aplicación [73]*:* a menudo se refiere al programa que se está ejecutando y a los archivos y bases de datos con los que trabaja

Cómputo de alto desempeño – HPC (High Performance Computing) [34]: es el uso y aplicación de computadores paralelos para la solución de problemas científicos, de ingeniería y técnicos.

Conjunto Paralelo (*Parallel Pool*) [70]: es un conjunto de *workers* de MATLAB en un *cluster* o computador, limitados juntos por un tipo especial de trabajo tal que ellos pueden ser usados interactivamente y pueden comunicarse entre sí durante el tiempo de vida del trabajo.

Cluster [70]*:* es un conjunto de computadores que están conectados a través de una red y están destinados a un propósito común.

Clusters Beowulf [69]*:* es un *cluster* con nodos compuestos por computadores personales (PC) o pequeños multiprocesadores simétricos de PCs integrados por redes de área local, y ejecutan un sistema operativo de libre distribución.

Cluster heterogéneo: un *cluster* con máquinas diferentes, en términos de hardware y software.

Demonio (Daemon: Disk and execution monitor) [69]: Procesos o servicios del sistema que se ejecuta en segundo plano.

Ethernet [73]: red de área local que transmite a 10Mbps y puede conectar en total hasta 1024 nodos.

Imagen del sistema (*system image*) [73]: Vista de memoria del entorno operativo actual, incluyendo el sistema operativo y programas en ejecución.

LAN (Local Area Network) [73]: red de comunicaciones que sirve a usuarios dentro de un área geográficamente limitada.

Mensaje (*message*) [73]: En comunicaciones, un conjunto de datos que se transmite en una línea de comunicaciones, de la misma forma a como un programa se convierte en un trabajo cuando se ejecuta en la computadora, los datos se convierten en mensajes cuando se transmiten en una red de comunicación.

Message Passing Interface (MPI): grupo de rutinas estándar para paso de mensajes con la sintaxis y semántica bien definidos [37]. Es el medio por el cual los *workers* se comunican entre sí mientras corren tareas en el mismo trabajo [70].

Nodo [70]*:* computador que es parte de un *cluster*.

Nodo principal (head node) [70]*:* usualmente, el nodo del *cluster* designado para corree el programador de trabajos y manejo de licencia. Este es además útil para correr todos los procesos no relacionados con los *worker* en una sola máquina.

Núcleo [73]*:* Unidad de procesamiento independiente que hace parte del procesador de un computador, y es la unidad que lee y ejecuta las instrucciones de un programa. Múltiples núcleos pueden correr múltiples instrucciones al mismo tiempo.

OpenMP (Open Multi Processsing) [36]: es un estándar de comunicación paralelo usado para paralelismo multiproceso y memoria compartida para lenguajes como C/C++ o Fortran. Este consta de tres partes: directivas de compilador, rutinas de la librería *runtime* así como variables ambientales.

Parallel Virtual Machine (PVM) [37]*:* es un software que habilita el ambiente de cómputo para trabajar máquinas en paralelo. Permite la programación de aplicaciones en una red constituida de máquinas heterogéneas, dando la impresión de ser un gran computador paralelo. Está compuesto de dos partes, el "demonio" que corre en todas las máquinas que componen el *cluster*, y las librerías de interface que contienen rutinas para transferir mensajes, asignar procesos, coordinar tareas, modificar la máquina virtual, prevenir fallas en los procesos, etc.

Sistema Operativo [74]: Conjunto de rutinas encargadas de la supervisión de las operaciones de un sistema de computador (carga, situación en memoria, sucesión de operaciones, ensamblaje de las instrucciones, compilación de programas, etc.). El sistema operativo, en cada caso introduce automáticamente el programa apropiado.

Tarea [70]: segmento de trabajo que es evaluado por un *worker*.

Trabajador (Worker) [70]: Sesión de trabajo de MATLAB que realiza la tarea computacional.

Trabajo (*Job*) [70]: operación completa a gran escala realizada en MATLAB, compuesta de un conjunto de tareas.

B. Anexo: Programa para análisis de contingencias con *slack* distribuido por AGC

contingency.m

```
% contingency.m executes an serial and parallel contingency analysis
% (with distributed slack bus) for branches and gens no related to
% AGC gens

function [mean_s, mean_p, var_p, g] = contingency(casedata, mpopt)

%% define named indices into bus, gen, branch matrices
[PQ, PV, REF, NONE, BUS_I, BUS_TYPE, PD, QD, GS,...
 BS, BUS_AREA, VM, VA, BASE_KV, ZONE, VMAX, VMIN,...
 LAM_P, LAM_Q, MU_VMAX, MU_VMIN] = idx_bus;

[F_BUS, T_BUS, BR_R, BR_X, BR_B, RATE_A, RATE_B, RATE_C,...
    TAP, SHIFT, BR_STATUS, PF, QF, PT, QT, MU_SF, MU_ST,...
    ANGMIN, ANGMAX, MU_ANGMIN, MU_ANGMAX] = idx_brch;

[GEN_BUS, PG, QG, QMAX, QMIN, VG, MBASE, GEN_STATUS, PMAX, PMIN,...
    MU_PMAX, MU_PMIN, MU_QMAX, MU_QMIN, PC1, PC2, QC1MIN, QC1MAX,...
    QC2MIN, QC2MAX, RAMP_AGC, RAMP_10, RAMP_30, RAMP_Q, APF] = idx_gen;

%% default arguments
if nargin < 2
            mpopt = mpoption;
            mpopt.pf.tol = 1e-4;

      if nargin < 1
                  casedata= 'case2383wp_apf';
      end
 end

%% Read case
mpc = loadcase(casedata);
baseMVA = mpc.baseMVA;

qlim = mpopt.pf.enforce_q_lims;

%% Base case solution
```

```
base = runpf(casedata, mpopt);
base.broutid = [0 0 0 0];   % [task_id  elem_ id elem_out_id  worker_id]

branch = base.branch;
gen = base.gen;
bus = base.bus;

%% get idx_case for branch with no connection to AGC gens
idg = find(gen(:,APF)~=0 & gen(:,GEN_STATUS) == 1);
idb = gen(idg, GEN_BUS);

nbr = size(branch,1);
id_branch = (1:nbr)';
aux = [];

for j = 1:(length(idg))
    p = find((branch(:,F_BUS) == idb(j)) | (branch(:,T_BUS)== idb(j)));
    aux = [aux;p];
end

for j = 1 : (size(bus,1))
    isol = find((branch(:,F_BUS) == j) | (base.branch(:,T_BUS) == j));

    if (length(isol) == 1)
        aux = [aux; isol];
    end
end

id_branch(aux) = [];
nbrout = size(id_branch,1);
id_branch = [ones(nbrout,1) id_branch];

%% get index for gens out AGC (pv gens)
ipv = (1:size(gen,1))';
ipv(idg)=[];

ngenout = length(ipv);
id_gen = [zeros(ngenout,1) ipv];

%% Make a serial contingency analysis for outage of lines and gens
[resS, mean_s, var_s, mean_perC_s] = serial_CA(base, mpopt, id_branch,
id_gen) ;

%% Run parallel contingency analysis

MatLabpool  open
nwork = MatLabpool('size');

% ordered by descending order according to time
id_case = [[id_gen; id_branch] mean_perC_s'];
id_case = sortrows(id_case,-3);

% workers clasification
```

```
Worker_rate = worker_test(nwork);



for j = 1:10
  [resP, Timep] = parallel_CA_B(base, mpopt, id_case,
nwork,Worker_rate);
  g(j) = Timep;
end

mean_p  = mean(g);
var_p = var(g);

% select the print output from serial or parallel
results = resP;
Time = mean_p;

%% get Vm and Pf arrays from structure array "results"
n = ngenout + nbrout;

for i=1:n
    mpc = results(i);
    voltage_res(i,:)= mpc.bus(:,VM);
    branch_res(i,:)= abs(mpc.branch(:,PF) + 1i*mpc.branch(:,QF));
    C (i,:) = mpc.exitflag;
end

%% verify voltage limits in pq buses
pq  = find(bus(:, BUS_TYPE) == PQ);

lim_vmax = bus (pq,VMAX)';
lim_vmin = bus (pq,VMIN)';

[voltage_list] = v_list(voltage_res, lim_vmax, lim_vmin, n, pq);

%% verify loading limits in trasmission elements
br_lim = branch(:,RATE_A)';

[branch_list] = b_list(branch_res, br_lim, n, nbrout);

%% print resutls
id_br = [branch(:,F_BUS) branch(:,T_BUS)];
id_g = gen(:,GEN_BUS);

success = base.success;

print_conting(C, casedata, branch_list, voltage_list, Time, id_case,...
nbrout, id_br, id_g, br_lim, lim_vmax, lim_vmin, pq, success, qlim);
```

npfagc.m

```
% Solves the power flow using a full Newton's method take into account
% the participation factors of AGC generators.

function res = npfagc(base,mpopt)

%% default arguments
if nargin < 2
    mpopt = mpoption;
end

%% options
tol     = mpopt.pf.tol;
max_it  = mpopt.pf.nr.max_it;

%% unpack input structure
gen = base.gen;
bus = base.bus;
branch = base.branch;
baseMVA = base.baseMVA;
e_out_id = base.broutid(2);     % elem id : 0 = gen, 1 = branch;
br_out_id = base.broutid(3);    % elem idx

%% start counter
start_npf = tic;

%% Disconnect gen or branch
if e_out_id == 0
    gen(br_out_id, GEN_STATUS) = 0;
else
    branch(br_out_id, BR_STATUS) = 0;
end

%% initialize
converged = 0;
i = 0;
success=0;

%% convert to internal indexing
[i2e, bus, gen, branch] = ext2int(bus, gen, branch);

%% get bus index of each type of bus (ref, pv, pq) and gen index for
%% agc generators (agc_g)
[ref, agc, pv, pq, agc_g, pv_g] = n_bustypes(bus, gen) ;

%% initial values
V = bus(:, VM) .* exp(sqrt(-1) * pi/180 * bus(:, VA));
Va = angle(V);
Vm = abs(V);

p0 = gen(agc_g,PG)/baseMVA;
P = p0;
```

```
beta = gen(agc_g,APF);

repeat = 1;

while (repeat)
%% alpha vector
alpha(:,1) = beta/beta(1);
alpha(1) = [];

%% Sbus
Sbus = makeSbus(baseMVA, bus, gen);

%% Ybus
[Ybus, Yf, Yt] = makeYbus(baseMVA, bus, branch);

%% set up indexing for updating V and P
nagc = length(agc);
npv  = length(pv);
npq  = length(pq);

j1 = 1;         j2 = nagc - 1;
j3 = j2 + 1;    j4 = j3 + npv-1;
j5 = j4 + 1;    j6 = j5 + npq-1;
j7 = j6 + 1;    j8 = j7 + npq-1;
j9 = j8 + 1;    j10 = j9 + j2;

%% evaluate F(x0)
mis = V .* conj(Ybus * V) - Sbus;

mis_p = (P(1) - p0(1)) - alpha.*(P(2:nagc,1) - p0(2:nagc,1));

F = [   real(mis([agc; pv; pq]));
        imag(mis(pq));
        mis_p                   ];

%% check tolerance
normF = norm(F, inf);

if normF < tol
    converged = 1;
end


%% do Newton iterations
while (~converged && i < max_it)
    %% update iteration counter
    i = i + 1;

    %% evaluate Jacobian
    [dSbus_dVm, dSbus_dVa] = dSbus_dV(Ybus, V);

    j11 = real(dSbus_dVa([agc; pv; pq], [agc; pv; pq]));
    j11(:,1)=[];
```

```
j12 = real(dSbus_dVm([agc; pv; pq], pq));

j21 = imag(dSbus_dVa(pq, [agc;pv; pq]));
j21(:,1)=[];

j22 = imag(dSbus_dVm(pq, pq));

[dP_dPagc, dQ_dPagc] = dSbus_dPagc(agc, pq, pv);

j13 = dP_dPagc;
j23 = dQ_dPagc;

[dPafp_dVa, dPafp_dVm, dPafp_dPagc] = dPafp_dX(agc, pq, pv, alpha);

j31 = dPafp_dVa;
j32 = dPafp_dVm;
j33 = dPafp_dPagc;

% augmented jacobian
J = [  j11 j12 j13;
       j21 j22 j23;
       j31 j32 j33     ];

%% compute update step
dx = -(J \ F);

%% update voltage and Pagc
if nagc
      if ref
        agc1 = agc;
        ind = find (agc ==ref);
        agc1(ind) = [];
      end

    Va(agc1) = Va(agc1) + dx(j1:j2);
    P = P + dx(j9:j10);
end

if npv
    Va(pv) = Va(pv) + dx(j3:j4);
end

if npq
    Va(pq) = Va(pq) + dx(j5:j6);
    Vm(pq) = Vm(pq) + dx(j7:j8);
end


V = Vm .* exp(1j * Va);
Vm = abs(V);
Va = angle(V);
```

```
%% Sbus wiht new Pagc
gen(agc_g,PG) = P * baseMVA;
Sbus = makeSbus(baseMVA, bus, gen);

    %% evalute F(x)
    mis = V .* conj(Ybus * V) - Sbus;

    mis_p = (P(1) - p0(1)) - alpha.*(P(2:nagc,1) - p0(2:nagc,1));

    F = [   real(mis(agc));
            real(mis(pv));
            real(mis(pq));
            imag(mis(pq));
            mis_p           ];

    %% check for convergence
    normF = norm(F, inf);

    if normF < tol
        converged = 1;
    end

end

%% update PG for generators in AGC
 gen(agc_g,PG) = P * baseMVA;

 [bus,gen,branch] = pfsoln_agc(baseMVA,bus,gen,branch,Ybus,Yf,Yt,V);

end

end_npf = toc(start_npf);


%% convert back to original bus numbering
[bus, gen, branch] = int2ext(i2e, bus, gen, branch);

exitflag = [converged,i,success];
%% convert results to structure
res.bus = bus;
res.gen = gen;
res.branch = branch;
res.exitflag = exitflag;
res.time = end_npf;
res.broutid = base.broutid;
```

parallel_CA_A.m

```
% parallel code for contingency analysis with scheme A
function Time = parallel_CA_A(base, mpopt, id_case, nwork)

%% output array setup
a.bus = [];
a.gen = [];
a.branch = [];
a.exitflag = [];
a.time = 0;
a.broutid = [];

output = repmat(a, size(id_case,1), 1);

%% parallel computing

Worker_Status = zeros(nwork,1);
Worker_Status(1) = -2;
Task_Status = zeros(size(id_case,1),1);

startLoop = tic;

spmd (nwork)

  if labindex == 1    % master worker

      while any(Task_Status ~= 2)

          % Tasks
          i_task = find(Task_Status == 0);
          if ~isempty(i_task)
              i_task = i_task(1);
          end

          % Workers
          i_idle = find(Worker_Status == 0);
          if ~isempty(i_idle)
              i_idle = i_idle(1);
          end

          if ~isempty(i_task) && ~isempty(i_idle)
            Worker_Status(i_idle) = i_task;
            Task_Status(i_task) = 1;

            if (id_case(i_task,1) == 0)
                base.broutid = [i_task, 0, id_case(i_task,2), i_idle];
            else
                base.broutid = [i_task, 1, id_case(i_task,2), i_idle];
            end

            labSend(base, i_idle);
```

```
          elseif any(Task_Status ~= 2)

              [stat, i, tag] = labProbe;

               if any(stat == 1)

                   mpcm = labReceive(i);
                   idt = mpcm.broutid(1);
                   output(idt) = mpcm;

                   Worker_Status(i) = 0;
                   Task_Status(idt) = 2;

               end

          end
      end

      % if all tasks were finished then send an empty array
      exitC = [];
      labSend(exitC, (2:nwork));

  else          % slave worker

   mpc_in = 1;
    while ~isempty(mpc_in)

      while ~labProbe
      end

      mpc = labReceive(1);

      if isempty(mpc)
        break;
      end

      out = npfagc(mpc, mpopt);

      labSend(out, 1);

    end

  end

end

Time = toc (startLoop);
```

parallel_CA_B.m

```
% parallel code for contingency analysis with scheme B
function Time = parallel_CA_B(base, mpopt, id_case, nwork, Worker_rate)

%% output array setup
a.bus = [];
a.gen = [];
a.branch = [];
a.exitflag = [];
a.time = 0;
a.broutid = [];

output = repmat(a, size(id_case,1), 1);

%% parallel computing
Worker_Status = zeros(nwork,1);
master = find(Worker_rate(:,2) == 1);
Worker_Status(master) = -2;

Worker_rate = Worker_rate(:,2);
Task_Status = zeros(size(id_case,1),1);

startLoop = tic;

spmd (nwork)

  if labindex == 1    % master worker

      while any(Task_Status ~= 2)

          % Tasks
          i_task = find(Task_Status == 0);
          if ~isempty(i_task)
             i_task = i_task(1);
          end

          % Workers
          i_idle = find(Worker_Status == 0);
          if ~isempty(i_idle)
             i_idle = i_idle(1);
          end


          if ~isempty(i_task) && ~isempty(i_idle)
            Worker_Status(i_idle) = i_task;
            Task_Status(i_task) = 1;

            if (id_case(i_task,1) == 0)
                base.broutid = [i_task, 0, id_case(i_task,2), i_idle];
            else
                base.broutid = [i_task, 1, id_case(i_task,2), i_idle];
            end
```

```
            i_idle = Worker_rate(i_idle);
            labSend(base, i_idle);

          elseif any(Task_Status ~= 2)

             [stat, i, tag] = labProbe;

              if any(stat == 1)

                  mpcm = labReceive(i);
                  idt = mpcm.broutid(1);
                  output(idt) = mpcm;

                  pos = find(Worker_rate == i);
                  Worker_Status(pos) = 0;
                  Task_Status(idt) = 2;

              end
          end
      end

      % if all tasks were finished then send an empty array
      exitC = [];
      labSend(exitC, (2:nwork));

  else          % slave worker

   mpc_in = 1;
    while ~isempty(mpc_in)

      while ~labProbe
      end

      mpc = labReceive(1);

      if isempty(mpc)
        break;
      end

      out = npfagc(mpc, mpopt);

      labSend(out, 1);

    end

  end

end

Time = toc (startLoop);
```

parallel_CA_C.m

```
% parallel code for contingency analysis with scheme C
function Time = parallel_CA_C(base, mpopt, id_case, nwork, Worker_rate)

%% output array setup
a.bus = [];
a.gen = [];
a.branch = [];
a.exitflag = [];
a.time = 0;
a.broutid = [];

output = repmat(a, size(id_case,1), 1);

%% parallel computing
Worker_Status = zeros(nwork,1);
master = find(Worker_rate(:,2) == 1);
Worker_Status(master) = -2;

Worker_rate = Worker_rate(:,2);
Task_Status = zeros(size(id_case,1),1);

startLoop = tic;

spmd (nwork)

  if labindex == 1          % master worker

      while any(Task_Status ~= 2)

          % Tasks
          i_task = find(Task_Status == 0);
          if ~isempty(i_task)
              f_task = i_task(end);
              i_task = i_task(1);

              if (i_task == f_task)
                  f_task = [];
              end
          end

          % Workers
          i_idle = find(Worker_Status == 0);
          if ~isempty(i_idle)
              f_idle = i_idle(end);
              i_idle = i_idle(1);

            if ~isempty(f_task)
                if ( f_idle == i_idle)
                    f_idle = [];
                end
            end
```

```
    end


    if ~isempty(i_task) && ~isempty(i_idle)
      Worker_Status(i_idle) = i_task;
      Task_Status(i_task) = 1;

      if (id_case(i_task,1) == 0)
          base.broutid = [i_task, 0, id_case(i_task,2), i_idle];
      else
          base.broutid = [i_task, 1, id_case(i_task,2), i_idle];
      end

      i_idle = Worker_rate(i_idle);
      labSend(base, i_idle);

      if ~isempty(f_task)&& ~isempty(f_idle)
          Worker_Status(f_idle) = f_task;
          Task_Status(f_task) = 1;

        if (id_case(f_task,1) == 0)
            base.broutid = [f_task, 0, id_case(f_task,2), f_idle];
        else
            base.broutid = [f_task, 1, id_case(f_task,2), f_idle];
        end

          f_idle = Worker_rate(f_idle);
          labSend(base, f_idle);
      end

    elseif any(Task_Status ~= 2)

        stat  = 1;
        while stat
            [stat, i, tag] = labProbe;

             if any(stat == 1)
                  mpcm = labReceive(i);
                  idt = mpcm.broutid(1);
                  output(idt) = mpcm;

                  pos = find(Worker_rate == i);
                  Worker_Status(pos) = 0;
                  Task_Status(idt) = 2;
            end
        end
    end
end

% if all tasks were finished then send an empty array
exitC = [];
labSend(exitC, (2:nwork));
```

```
else          % slave worker

   mpc_in = 1;
    while ~isempty(mpc_in)

      while ~labProbe
      end

      mpc = labReceive(1);

      if isempty(mpc)
        break;
      end

      out = npfagc(mpc, mpopt);

      labSend(out, 1);

    end
  end
end

Time = toc (startLoop);
```

C. Anexo: Ejemplo de presentación de resultados de salida del programa

CONTINGENCY ANALYSIS for case30_apf

SUMMARY
Enforce limits, one-at-a-time bus type conversion.

Contingency (N-1)	Converged? Yes/No	No. Iterat.	Load Viol.	Voltage Viol.	Max. branch [%]	Min. Voltage [p.u.]	Max. Voltage [p.u.]	Enforce P/Q limits Remarks
Generator 22	Yes	3	1	9	107.31	0.93	0.98	None
Line 10-20	Yes	2	2	3	108.88	0.93	0.99	None
Line 15-18	Yes	2	2	2	108.72	0.95	0.99	None
Line 21-22	Yes	2	1	2	107.92	0.95	0.99	None
Line 19-20	Yes	2	1	2	108.87	0.94	0.99	None
Line 4-6	Yes	2	2	1	107.29	0.95	0.99	None
Line 15-23	Yes	2	2	0	107.85	0.96	0.99	None
Line 10-22	Yes	2	2	0	113.42	0.96	0.99	None
Line 10-17	Yes	2	1	1	108.80	0.93	0.99	None
Line 6-7	Yes	2	1	1	109.10	0.94	0.99	None
Generator 13	Yes	3	2	0	105.94	0.95	0.99	None
Line 6-28	Yes	1	1	0	104.72	0.96	0.99	None
Line 8-28	Yes	2	1	0	134.76	0.96	0.99	None
Line 29-30	Yes	2	1	0	108.86	0.95	0.99	None
Line 24-25	Yes	2	1	0	107.15	0.96	0.99	None
Line 23-24	Yes	2	1	0	109.53	0.96	0.99	None
Line 22-24	Yes	2	1	0	108.49	0.96	0.99	None
Line 10-21	Yes	2	1	0	108.66	0.96	1.00	None
Line 18-19	Yes	2	1	0	108.77	0.96	0.99	None
Line 16-17	Yes	2	1	0	108.88	0.96	0.99	None
Line 14-15	Yes	1	1	0	108.82	0.96	0.99	None
Line 12-16	Yes	2	1	0	108.90	0.96	0.99	None
Line 12-15	Yes	2	1	0	109.17	0.96	0.99	None
Line 12-14	Yes	2	1	0	108.90	0.96	0.99	None
Line 4-12	Yes	1	1	0	108.65	0.96	0.99	None
Line 9-10	Yes	2	1	0	109.10	0.96	0.99	None
Line 6-10	Yes	1	1	0	108.93	0.96	0.99	None
Line 6-9	Yes	2	1	0	109.10	0.96	0.99	None
Line 6-8	Yes	3	0	1	132.58	0.86	0.99	None
Line 5-7	Yes	2	1	0	108.25	0.95	1.00	None
Line 3-4	Yes	2	1	0	108.55	0.96	1.00	None
Generator 23	Yes	2	1	0	108.00	0.96	0.99	None

* Transmission Elements Loading

Contingency	Element ID	Value [MVA]	Limit [MVA]	Loading [%]
Generator 22	Line 6-8	34.34	32	107.31
Line 10-20	Line 6-8	34.84	32	108.88
	Line 15-18	16.40	16	102.52
Line 15-18	Line 6-8	34.79	32	108.72
	Line 21-22	32.21	32	100.67
Line 21-22	Line 6-8	34.53	32	107.92
Line 19-20	Line 6-8	34.84	32	108.87
Line 4-6	Line 6-8	34.33	32	107.29
	Line 21-22	32.81	32	102.54
Line 15-23	Line 6-8	34.51	32	107.85
	Line 21-22	33.10	32	103.43
Line 10-22	Line 21-22	36.29	32	113.42
	Line 6-8	34.78	32	108.70
Line 10-17	Line 6-8	34.82	32	108.80
Line 6-7	Line 6-8	34.91	32	109.10
Generator 13	Line 6-8	33.90	32	105.94
	Line 21-22	32.14	32	100.44
Line 6-28	Line 6-8	33.51	32	104.72
Line 8-28	Line 6-8	43.12	32	134.76
Line 29-30	Line 6-8	34.83	32	108.86
Line 24-25	Line 6-8	34.29	32	107.15
Line 23-24	Line 6-8	35.05	32	109.53
Line 22-24	Line 6-8	34.72	32	108.49
Line 10-21	Line 6-8	34.77	32	108.66
Line 18-19	Line 6-8	34.81	32	108.77
Line 16-17	Line 6-8	34.84	32	108.88
Line 14-15	Line 6-8	34.82	32	108.82
Line 12-16	Line 6-8	34.85	32	108.90
Line 12-15	Line 6-8	34.94	32	109.17

Line 12-14	Line 6-8	34.85	32	108.90
Line 4-12	Line 6-8	34.77	32	108.65
Line 9-10	Line 6-8	34.91	32	109.10
Line 6-10	Line 6-8	34.86	32	108.93
Line 6-9	Line 6-8	34.91	32	109.10
Line 5-7	Line 6-8	34.64	32	108.25
Line 3-4	Line 6-8	34.74	32	108.55
Generator 23	Line 6-8	34.56	32	108.00

* Bus voltages magnitude

Contingency	Bus ID	Value [p.u.]	Vmin [p.u.]	Vmax [p.u.]
Generator 22	8	0.95	0.95	1.05
	9	0.95	0.95	1.05
	11	0.95	0.95	1.05
	18	0.94	0.95	1.05
	10	0.94	0.95	1.05
	17	0.94	0.95	1.05
	19	0.93	0.95	1.05
	20	0.93	0.95	1.05
	21	0.93	0.95	1.05
Line 10-20	18	0.94	0.95	1.05
	19	0.93	0.95	1.05
	20	0.93	0.95	1.05
Line 15-18	19	0.95	0.95	1.05
	18	0.95	0.95	1.05
Line 21-22	19	0.95	0.95	1.05
	21	0.95	0.95	1.05
Line 19-20	18	0.95	0.95	1.05
	19	0.94	0.95	1.05
Line 4-6	8	0.95	0.95	1.05
Line 10-17	17	0.93	0.95	1.05
Line 6-7	7	0.94	0.95	1.05
Line 6-8	8	0.86	0.95	1.05

D. Anexo: Tablas de resultados

Las Tablas D-1 a D-12, presentan los resultados de 10 mediciones de tiempo de cómputo paralelo en el *cluster*, para los diferentes sistemas de prueba y estrategias. Así mismo, para cada serie de medidas se muestra el cálculo del promedio y la desviación estándar para la población, sin tomar en cuenta la primera medida, puesto que ésta incluye los tiempos de precompilación de los archivos enviados, tarea que no es ejecutada en las medidas sucesivas.

Tabla D- 1: Tiempos de cómputo paralelo en el *cluster*. Caso 30 barras – Estrategia A

Medida	**Caso 30 barras - Estrategia A**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,2884	0,8405	1,2677	1,1828	1,2695
2	0,2086	0,5747	0,8833	1,0481	0,8672
3	0,1114	0,5771	0,8573	1,0594	0,8602
4	0,1105	0,5872	0,8536	1,0598	0,8605
5	0,1439	0,5567	0,8469	1,0363	0,9457
6	0,1036	0,5648	0,8433	0,8701	0,8578
7	0,1085	0,5882	0,8336	1,0399	0,8754
8	0,1027	0,5972	0,8371	1,0297	0,7929
9	0,1035	0,5918	0,8665	0,8492	0,8682
10	0,1120	0,6476	0,6535	1,0296	0,7696
$\bar{x}$	0,1227	0,5873	0,8306	1,0025	0,8553
σ	0,0326	0,0246	0,0642	0,0772	0,0474

Tabla D- 2: Tiempos de cómputo paralelo en el *cluster*. Caso 30 barras – Estrategia B

Medida	Caso 30 barras - Estrategia B				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,2962	0,8469	1,2145	1,2629	1,0454
2	0,1039	0,6255	0,9069	1,0433	0,8727
3	0,1811	0,5712	0,8436	1,0292	0,8658
4	0,1154	0,5749	0,8388	1,0301	0,8670
5	0,1079	0,5547	0,8490	1,0313	0,8689
6	0,1095	0,5858	0,8295	0,8411	0,8647
7	0,1500	0,6213	0,8378	1,0351	0,8636
8	0,1787	0,5772	0,8533	1,0600	0,6814
9	0,1078	0,5670	0,8482	1,0302	0,6842
10	0,1038	0,5662	0,8428	1,0310	0,8633
$\bar{x}$	0,1287	0,5826	0,8500	1,0146	0,8257
σ	0,0305	0,0232	0,0212	0,0621	0,0765

Tabla D- 3: Tiempos de cómputo paralelo en el *cluster*. Caso 30 barras – Estrategia C

Medida	Caso 30 barras - Estrategia C				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,2988	0,9045	1,2638	1,2829	1,1956
2	0,1060	0,5947	0,8482	0,8598	0,8282
3	0,1108	0,6392	0,8593	0,8715	0,8740
4	0,1074	0,5726	0,8483	0,8373	0,8856
5	0,1602	0,5560	0,8333	1,0681	0,8652
6	0,1115	0,5823	0,8396	1,0310	0,6858
7	0,1095	0,5761	0,8494	0,8366	0,8012
8	0,1050	0,5771	0,8523	1,0409	0,8676
9	0,1163	0,5720	0,8596	0,8316	0,8560
10	0,1497	0,5753	0,8703	1,0427	0,8650
$\bar{x}$	0,1196	0,5828	0,8511	0,9355	0,8365
σ	0,0193	0,0221	0,0104	0,0996	0,0585

Tabla D- 4: Tiempos de cómputo paralelo en el *cluster*. Caso 118 barras – Estrategia A

Medida	Caso 118 barras - Estrategia A				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,6101	1,0036	1,2637	1,5733	1,3906
2	0,4499	0,8556	1,1123	1,1390	1,1644
3	0,4346	0,8372	1,0915	1,1934	1,0101
4	0,4712	0,8517	1,0818	1,2201	1,3492
5	0,4371	0,8603	1,1042	1,2906	1,3890
6	0,4138	0,8423	1,1428	1,2939	1,3033
7	0,5124	0,8467	1,0565	1,3394	1,1651
8	0,4164	0,8579	1,0626	1,5106	1,0576
9	0,4251	0,8570	1,1076	1,3530	1,1375
10	0,4292	0,8744	1,2249	1,3120	0,9973
$\bar{x}$	0,4433	0,8537	1,1094	1,2947	1,1748
σ	0,0295	0,0103	0,0478	0,1013	0,1361

Tabla D- 5: Tiempos de cómputo paralelo en el *cluster*. Caso 118 barras – Estrategia B

Medida	Caso 118 barras - Estrategia B				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,5600	1,0641	1,2679	1,9498	1,4056
2	0,4425	0,8565	1,0707	1,4166	1,2518
3	0,4277	1,0256	1,0504	1,3230	1,3671
4	0,4220	0,8498	1,0748	1,2593	1,3163
5	0,4435	0,8686	1,0839	1,2839	1,1386
6	0,4159	0,8484	1,0648	1,3661	1,2681
7	0,4407	0,8570	1,1094	1,4004	1,3580
8	0,4364	0,9421	1,0749	1,3125	1,3934
9	0,4220	0,9141	1,1218	1,2844	0,9417
10	0,4345	0,9329	1,0758	1,1485	1,1549
$\bar{x}$	0,4317	0,8994	1,0807	1,3105	1,2433
σ	0,0095	0,0565	0,0208	0,0766	0,1361

Tabla D- 6: Tiempos de cómputo paralelo en el *cluster*. Caso 118 barras – Estrategia C

Medida	Caso 118 barras - Estrategia C				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,6468	1,0070	1,2464	1,8508	1,4729
2	0,4158	0,8952	1,1583	1,4239	1,3350
3	0,4317	0,8998	1,1548	1,3826	1,1225
4	0,4171	0,9727	1,1223	1,5672	1,2349
5	0,4372	0,8857	1,2357	1,4829	1,3315
6	0,4104	0,8855	1,0754	1,4369	0,9398
7	0,4168	0,8665	1,0924	1,3440	1,1719
8	0,4249	0,8847	1,0869	1,8413	1,2135
9	0,5035	0,8772	1,1069	1,1630	1,1581
10	0,4274	0,8966	1,1721	1,2652	1,3219
$\bar{x}$	0,4316	0,8960	1,1339	1,4341	1,2032
σ	0,0266	0,0288	0,0484	0,1822	0,1192

Tabla D- 7: Tiempos de cómputo paralelo en el *cluster*. Caso 300 barras – Estrategia A

Medida	Caso 300 barras - Estrategia A				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	2,9241	3,2471	3,7519	5,8488	5,4084
2	2,7576	3,0853	3,3173	4,2845	4,8658
3	2,7731	3,0850	3,5373	4,6523	4,8072
4	2,7850	3,0619	3,3057	3,8692	4,6897
5	2,7445	3,0305	3,4196	4,2599	4,7822
6	2,8075	3,0280	3,4997	4,7404	4,4687
7	2,7700	3,0417	3,3657	4,7961	4,7232
8	2,8026	3,1713	3,4045	4,7796	4,9060
9	2,7498	3,0934	3,4229	4,3499	4,4681
10	2,8536	3,1389	3,3337	4,0795	4,3388
$\bar{x}$	2,7826	3,0818	3,4007	4,4235	4,6722
σ	0,0324	0,0459	0,0753	0,3154	0,1884

Tabla D- 8: Tiempos de cómputo paralelo en el *cluster*. Caso 300 barras – Estrategia B

Medida	Caso 300 barras - Estrategia B				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	2,9240	3,1924	3,6147	4,7158	5,0805
2	2,7655	3,0141	3,3362	4,6730	4,5959
3	2,7553	3,0008	3,3619	4,1989	4,3987
4	2,8636	2,9992	3,5141	4,5561	4,6434
5	2,8151	3,0468	3,2975	4,1703	4,5386
6	2,8136	3,1158	3,3932	4,4502	4,3126
7	2,8196	3,0768	3,5480	4,3815	4,5384
8	2,7314	3,0081	3,4746	4,4455	4,7147
9	2,7426	3,0094	3,3461	4,3646	4,1295
10	2,8408	3,0980	3,4677	4,3615	4,7808
$\bar{x}$	2,7942	3,0410	3,4155	4,4002	4,5170
σ	0,0440	0,0426	0,0829	0,1490	0,1937

Tabla D- 91: Tiempos de cómputo paralelo en el *cluster*. Caso 300 barras – Estrategia C

Medida	Caso 300 barras - Estrategia C				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	3,0065	3,4699	3,8464	5,1680	4,9399
2	2,7493	3,1883	3,5072	3,9743	4,1904
3	2,7158	3,2202	3,4185	4,1158	3,9310
4	2,6603	3,1628	3,5778	4,5790	4,4401
5	2,7279	3,1906	3,5169	4,0548	4,1704
6	2,8168	3,1261	3,5528	4,3285	4,2114
7	2,8404	3,2175	3,7089	4,2278	4,4646
8	2,7624	3,2233	3,4168	4,0771	4,3863
9	2,7161	3,1780	3,5216	3,9580	4,0131
10	2,6789	3,1982	3,3847	3,7692	4,1470
$\bar{x}$	2,7409	3,1894	3,5117	4,1205	4,2171
σ	0,0557	0,0295	0,0936	0,2217	0,1735

Tabla D- 20: Tiempos de cómputo paralelo en el *cluster*. Caso 2383 barras – Estrategia A

Medida	Caso 2383 barras - Estrategia A				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	148,0070	200,1393	496,0669	378,4789	476,8833
2	125,7628	172,2241	452,3352	340,0406	414,1402
3	78,9531	151,8236	453,9163	282,7725	377,5787
4	77,8703	127,3465	412,3473	319,3858	358,2303
5	76,8428	142,8445	430,0947	296,0821	358,8368
6	76,0263	127,6126	418,2791	344,0694	381,0994
7	77,3849	135,9728	410,9061	326,0649	408,2138
8	76,0924	123,5588	436,8215	347,6331	432,4826
9	76,2200	123,6743	413,5382	314,8476	450,7889
10	75,8132	125,4974	416,5860	326,0559	372,2060
$\bar{x}$	82,3295	136,7283	427,2027	321,8835	394,8419
σ	15,3865	15,5166	16,0259	20,4694	31,1522

Tabla D- 11: Tiempos de cómputo paralelo en el *cluster*. Caso 2383 barras – Estrategia B

Medida	Caso 2383 barras - Estrategia B				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	139,6402	212,5132	468,4157	379,3295	512,2994
2	119,4960	200,3305	452,9244	317,8446	430,9871
3	78,5876	133,5333	436,8334	311,0696	401,2239
4	78,0247	133,2272	426,4425	289,4796	397,0516
5	77,2174	133,5541	455,4498	304,3365	420,9723
6	76,4841	125,5440	420,3023	369,1152	421,2037
7	76,6900	124,2618	430,0529	352,3010	437,1025
8	76,3576	122,5596	427,0169	326,6037	462,4208
9	76,4733	124,1538	402,6103	328,9398	449,5093
10	76,7744	128,0832	428,8574	31,2756	492,8812
$\bar{x}$	81,7895	136,1386	431,1655	292,3295	434,8169
σ	13,3507	23,0670	15,1833	95,0505	28,4818

Tabla D- 12: Tiempos de cómputo paralelo en el *cluster*. Caso 2383 barras – Estrategia C

Medida	**Caso 2383 barras - Estrategia C**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	185,8432	241,6397	483,5514	306,2322	477,2891
2	156,1576	200,0473	438,0306	291,0568	453,9505
3	83,8786	160,2075	407,1031	252,6570	412,0413
4	78,9938	158,6578	396,3094	252,5580	420,4720
5	77,1656	159,0105	396,0269	275,5282	436,3719
6	76,8245	153,7338	437,3595	299,5421	472,1261
7	78,7559	163,6377	428,1259	301,3669	456,3870
8	76,5214	153,2246	444,6679	303,3792	460,2599
9	76,2175	148,9324	457,6185	301,9802	448,0735
10	76,5438	168,0204	456,7605	300,7221	454,8810
$\bar{x}$	86,7843	162,8302	429,1114	286,5323	446,0626
σ	24,6302	14,2195	22,6633	19,8787	18,3719

Las Tablas D-13 a D-24, presentan los resultados de 10 mediciones de tiempo de cómputo paralelo en el computador multinúcleo, para los diferentes sistemas de prueba y estrategias. Así mismo, para cada serie de medidas se muestra el cálculo del promedio y la desviación estándar para la población, sin tomar en cuenta la primera medida, puesto que ésta incluye los tiempos de precompilación de los archivos enviados, tarea que no es ejecutada en las medidas sucesivas.

Tabla D- 33: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 30 barras – Estrategia A

Medida	**Caso 30 barras – Estrategia A**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,2912	0,2906	0,3157	0,3510	0,3788
2	0,0971	0,0919	0,1150	0,1970	0,2701
3	0,0971	0,0949	0,1106	0,1800	0,2572
4	0,0976	0,1015	0,1124	0,1776	0,2451
5	0,0996	0,0953	0,1067	0,1597	0,2769
6	0,1003	0,0930	0,1019	0,1963	0,2706
7	0,1015	0,0951	0,1226	0,1908	0,2561
8	0,1024	0,0944	0,1089	0,1903	0,2466
9	0,0982	0,0958	0,1012	0,1671	0,2594
10	0,1013	0,0941	0,1153	0,1612	0,2419
$\bar{x}$	0,0995	0,0951	0,1105	0,1800	0,2582
σ	0,0019	0,0025	0,0064	0,0138	0,0117

Tabla D- 14: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 30 barras – Estrategia B

Medida	**Caso 30 barras – Estrategia B**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,2677	0,2595	0,2707	0,3003	0,3644
2	0,1030	0,0934	0,1245	0,1911	0,2819
3	0,1023	0,0957	0,1326	0,1732	0,2746
4	0,1009	0,0946	0,1319	0,1841	0,2781
5	0,1007	0,0964	0,1289	0,1768	0,2551
6	0,1018	0,0938	0,1245	0,1686	0,2635
7	0,1023	0,1001	0,1079	0,2170	0,2553
8	0,1023	0,1019	0,1060	0,2147	0,2598
9	0,1022	0,0982	0,1247	0,2073	0,2584
10	0,1004	0,0988	0,1141	0,1878	0,2625
$\bar{x}$	0,1018	0,0970	0,1217	0,1912	0,2655
σ	0,0008	0,0028	0,0094	0,0169	0,0095

Tabla D- 15: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 30 barras – Estrategia C

Medida	**Caso 30 barras – Estrategia C**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,2764	0,2742	0,2884	0,3103	0,3440
2	0,1018	0,1032	0,1229	0,1890	0,2644
3	0,1057	0,1015	0,1165	0,2129	0,2507
4	0,1069	0,0957	0,1124	0,2050	0,2941
5	0,1064	0,1022	0,1113	0,1894	0,2727
6	0,1071	0,1026	0,1163	0,1775	0,2483
7	0,1090	0,1023	0,1251	0,1984	0,2657
8	0,1073	0,1055	0,1247	0,1759	0,2448
9	0,1091	0,0986	0,1230	0,1955	0,2731
10	0,1069	0,1006	0,1200	0,1856	0,2813
$\bar{x}$	0,1067	0,1014	0,1191	0,1921	0,2661
σ	0,0020	0,0027	0,0049	0,0115	0,0153

Tabla D- 46: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 118 barras – Estrategia A

Medida	Caso 118 barras – Estrategia A				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,6386	0,5081	0,4658	0,8445	1,1553
2	0,4250	0,3010	0,2559	0,6525	1,0993
3	0,4267	0,2944	0,2545	0,7513	1,0716
4	0,4225	0,2924	0,2518	0,6501	1,1412
5	0,4185	0,2966	0,2564	0,5965	1,1045
6	0,4282	0,2921	0,2485	0,7543	1,0787
7	0,4266	0,2889	0,2523	0,5785	1,0669
8	0,4241	0,2938	0,2576	0,7147	1,1180
9	0,4193	0,2940	0,2501	0,6814	1,1287
10	0,4245	0,2974	0,2528	0,6231	1,1277
$\bar{x}$	0,4239	0,2945	0,2533	0,6669	1,1041
σ	0,0031	0,0033	0,0028	0,0600	0,0255

Tabla D- 57: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 118 barras – Estrategia B

Medida	Caso 118 barras – Estrategia B				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,5962	0,4713	0,4369	0,8650	1,1654
2	0,4285	0,2917	0,2615	0,7243	1,0583
3	0,4200	0,2988	0,2527	0,7440	1,0979
4	0,4195	0,2951	0,2547	0,6919	1,0347
5	0,4257	0,2964	0,2646	0,6254	1,1367
6	0,4249	0,2947	0,2531	0,8214	0,9394
7	0,4248	0,2960	0,2640	0,7093	1,0669
8	0,4245	0,2900	0,2495	0,6067	1,1340
9	0,4198	0,2953	0,2491	0,7258	1,1179
10	0,4253	0,2968	0,2519	0,7877	0,9801
$\bar{x}$	0,4237	0,2950	0,2557	0,7152	1,0629
σ	0,0030	0,0025	0,0057	0,0650	0,0647

Tabla D- 68: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 118 barras – Estrategia C

Medida	Caso 118 barras – Estrategia C				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	0,6175	0,4805	0,4800	0,7012	1,2109
2	0,4326	0,3103	0,2668	0,5945	1,2039
3	0,4335	0,2985	0,2719	0,4892	1,1233
4	0,4321	0,3060	0,2627	0,5720	1,1059
5	0,4400	0,2981	0,2735	0,5206	1,1274
6	0,4384	0,3039	0,2623	0,6035	1,1349
7	0,4342	0,3096	0,2577	0,5396	1,1192
8	0,4282	0,3233	0,2694	0,4549	1,1827
9	0,4311	0,3008	0,2819	0,5479	1,1553
10	0,4327	0,3038	0,2571	0,5574	1,0390
$\bar{x}$	0,4336	0,3060	0,2670	0,5422	1,1324
σ	0,0034	0,0073	0,0076	0,0454	0,0446

Tabla D-19: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 300 barras – Estrategia A

Medida	Caso 300 barras – Estrategia A				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	3,0091	1,9842	1,5487	1,3266	2,0445
2	2,8029	1,7704	1,3269	1,0999	1,5009
3	2,7985	1,7725	1,3332	1,0969	1,6572
4	2,7756	1,7667	1,3253	1,0966	1,8962
5	2,7787	1,7532	1,3300	1,0950	1,7841
6	2,7995	1,7750	1,3331	1,0996	1,5956
7	2,7829	1,7699	1,3288	1,1058	1,8681
8	2,7640	1,7677	1,3267	1,0992	1,7139
9	2,7931	1,7554	1,3256	1,0977	1,8512
10	2,8061	1,7633	1,3352	1,1069	1,2074
$\bar{x}$	2,7890	1,7660	1,3294	1,0997	1,6750
σ	0,0135	0,0070	0,0035	0,0038	0,2072

Tabla D- 70: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 300 barras – Estrategia B

Medida	Caso 300 barras – Estrategia B				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	2,9660	1,9507	1,5060	1,2894	2,0862
2	2,7695	1,7715	1,3321	1,1124	1,4062
3	2,7781	1,7739	1,3345	1,1056	1,8077
4	2,7915	1,8010	1,3303	1,1105	1,9297
5	2,7931	1,7835	1,3278	1,1039	1,6777
6	2,7969	1,7832	1,3238	1,1121	1,8525
7	2,7935	1,7744	1,3261	1,1072	1,6416
8	2,8167	1,7791	1,3368	1,1476	1,9434
9	2,7824	1,7805	1,3274	1,1055	1,8576
10	2,7966	1,7743	1,3305	1,1020	1,9253
$\bar{x}$	2,7909	1,7802	1,3299	1,1119	1,7824
σ	0,0126	0,0084	0,0039	0,0131	0,1672

Tabla D- 81: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 300 barras – Estrategia C

Medida	Caso 300 barras – Estrategia C				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	2,9749	1,9697	1,5242	1,2952	1,7107
2	2,8070	1,7746	1,3435	1,1257	1,4723
3	2,7906	1,7765	1,3355	1,1228	1,2868
4	2,8400	1,7857	1,3374	1,1191	1,3959
5	2,7923	1,7869	1,3439	1,1356	1,3683
6	2,8091	1,7868	1,3357	1,1167	1,4909
7	2,8003	1,7855	1,3393	1,1064	1,3928
8	2,8095	1,7801	1,3411	1,1151	1,4650
9	2,8104	1,7841	1,3303	1,1227	1,4448
10	2,8027	1,7937	1,3367	1,1091	1,3410
$\bar{x}$	2,8069	1,7838	1,3382	1,1192	1,4064
σ	0,0136	0,0055	0,0041	0,0083	0,0639

Tabla D- 92: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 2383 barras – Estrategia A

Medida	**Caso 2383 barras – Estrategia A**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	77,2796	49,4402	37,0390	31,1190	28,1819
2	76,8840	49,3940	36,8068	29,8943	27,7237
3	77,2560	49,3106	36,8467	30,9175	27,4746
4	76,9668	49,2779	36,8224	30,2102	27,7669
5	76,9403	49,3523	36,8002	31,0151	27,5895
6	76,7846	49,3261	36,8102	30,9329	27,2101
7	77,0929	49,3591	36,7923	31,0754	27,4282
8	76,8875	49,2921	36,8064	30,8364	27,2824
9	77,0035	49,2786	36,8682	30,7993	27,4476
10	77,1710	49,4422	36,9121	30,8653	27,5600
$\bar{x}$	76,9985	49,3370	36,8295	30,7274	27,4981
σ	0,1416	0,0525	0,0370	0,3771	0,1741

Tabla D- 23: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 2383 barras – Estrategia B

Medida	**Caso 2383 barras – Estrategia B**				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	78,5997	49,3157	36,9317	30,3273	28,0478
2	78,5806	49,2221	36,9227	30,2212	27,8102
3	78,4540	49,2415	36,8943	30,2181	28,0052
4	78,4431	49,2558	36,9031	30,2450	27,9491
5	78,5668	49,2226	36,8840	30,2482	28,0338
6	78,5288	49,2513	36,9060	30,2815	27,9568
7	78,5755	49,2212	36,9309	30,2702	28,0118
8	78,4352	49,2771	36,9134	30,2683	28,0427
9	78,3866	49,2400	36,9179	30,2705	28,0264
10	78,3932	49,2857	36,9053	30,2740	27,8316
$\bar{x}$	78,4849	49,2464	36,9086	30,2552	27,9631
σ	0,0740	0,0223	0,0136	0,0220	0,0819

Tabla D- 104: Tiempos de cómputo paralelo en el computador multinúcleo. Caso 2383 barras – Estrategia C

Medida	Caso 2383 barras – Estrategia C				
	Número de procesadores				
	4	**6**	**8**	**10**	**12**
	Tiempo [s]				
1	79,4333	48,7403	37,1601	30,5381	27,7143
2	78,2805	48,7042	37,0446	30,3491	27,6450
3	78,4714	48,6042	37,1271	30,4428	27,3985
4	78,2953	48,6522	37,1542	30,4546	27,4624
5	78,6307	48,7010	37,0605	30,4422	27,4621
6	78,4975	48,6661	37,1046	30,5115	27,4911
7	78,6177	48,6713	37,1131	30,4316	27,4222
8	78,6346	48,6973	37,0350	30,3833	27,5692
9	78,8337	48,7269	37,0435	30,4516	26,9818
10	78,5563	48,6135	37,1041	30,5135	27,4804
$\bar{x}$	78,5353	48,6707	37,0874	30,4422	27,4347
σ	0,1644	0,0394	0,0401	0,0499	0,1750

En la Tabla D-25, se relacionan los tiempos promedio de ejecución serial del análisis de contingencia en el computador multinúcleo, para cada sistema de prueba y cada una de las tres estrategias de asignación de carga computacional.

Tabla D- 115: Tiempos promedio de ejecución serial en el computador multinúcleo

Tamaño del sistema de prueba [barras]	tiempo serial promedio [s]		
	Estrategia A	**Estrategia B**	**Estrategia C**
30	0,1041	0,1069	0,1050
118	0,9288	0,9262	0,9319
300	7,6437	7,6580	7,6279
2383	212,6362	214,9964	213,8085

Bibliografía

[1] N. Balu, T. Bertram, A. Bose, V. Brandwajn, G. Cauley, D. Curtice, A. Fouad, L. Fink, M. G. Lauby, B. F. Wollenberg, and J. N. Wrubel, «On-line Power System Security Analsis,» *Proc. IEEE,* vol. 80, nº 2, pp. 262-280, Febrero 1992.

[2] IEEE, The Authoritative Dictionary of IEEE Standard Terms, 2000.

[3] C.I.F. Agreira, C.M.M. Ferreira, J.A.D. Pinto, F.P.M. Barbosa, «Contingency screening and ranking algorithm using two different sets of security performance indices,» *Proc. IEEE Power Tech Conference,* vol. 4, pp. 23-26, 2003.

[4] M. Shahidehpour, W. F. Tinney, Y. Fu, «Impact of Security on Power Systems Operation,» *Proc. IEEE,* vol. 93, nº 11, pp. 2013-2025, Noviembre 2005.

[5] A.J. Wood, B. F. Wollenberg , Power Generation, Operation and Control, 2 ed., New York: John Wiley and Sons Inc., 1996.

[6] M. S. Sachdev and S. A. Ibrahim, «A Fast Approximate Technique for Outage Studies in Power System Planning and Operation,» *IEEE Transaction on Power Apparatus and Systems,* vol. 93, pp. 1133-1141, Mayo 1974.

[7] F.C. Aschmoneit and J.F. Verstege, «An External System Equivalent for On-Line Steady-State Generator Outage Simulation,» *IEEE Transactions on Power Apparatus and Systems,* vol. 98, nº 3, Mayo 1979.

[8] M. K. Enns, J.J. Quada, B. Sackett, «Fast Linear Contingency Analysis,» *IEEE Transactions on Power Apparatus and Systems,* vol. 101, nº 4, pp. 783-791, 1982.

[9] A. Mohamed, «Performance Comparisons of AC Load-flow Techniques for Real Time Applicatios,» *Proc. IEEE,* vol. 138, nº 5, pp. 457-461, Septiembre 1991.

[10] A. Ozdemir, J. Y. Lim, C. Singh, «Branch Outage Simulation for MVAR Flows: Bounded Network Solution,» *IEEE Transactions on Power Systems,* vol. 18, nº 4, pp. 1523-1528, 2003.

[11] P. Sood, *Development of Improved dc Network Model for Contingency Analysis,* Arizona: Arizona State University, 2014.

[12] B. Stott, J. Jardim, and O. Alsac, «DC Power Flow Revisited,» *IEEE Transactions on Power Systems*, vol. 24, nº 3, pp. 1290-1300, 2009.

[13] S. Vemuri, R. E. Usher, «On-Line Automatic Contingency Selection Algorithms,» *IEEE Transactions on Power Apparatus and Systems*, vol. 102, nº 2, pp. 346-354, Febrero 1983.

[14] J. Kim, G. Maria, V. Wong, «Contingency Ranking and Simulation for On-Line Use,» *IEEE Transactions on Power Apparatus and Systems*, vol. 104, nº 9, pp. 2401-2407, 1985.

[15] V. Brandwajn, M.G. Lauby , «Complete Bounding Method for AC Contingency Screening,» *IEEE Transactions on Power Systems*, vol. 4, nº 2, pp. 724-729, 1989.

[16] M.M.M. El-Arini, S.A. Soliman and A.M. Al-Kandari, «A Fast Technique for Outage Studies in Power System Planning and Operation,» *Electric Power Systems Research,* vol. 32, pp. 203-211, 1995.

[17] C.A. Castro, A. Bose, E. Handschin, W. Hoffmann, «Comparison of Different Screening Techniques for the Contingency Selection Function,» *Electrical Power & Energy Systems,* vol. 18, nº 7, pp. 425-430, 1996.

[18] Sun Wook Kang, A. P. Meliopoulos, «Contingency Selection via Quadratized Power Flow Sensitivity Analysis,» *Proc. IEEE Power Engineering Society Summer Meeting,* vol. 3, pp. 1494-1499, 2002.

[19] A. Escobar y L.F. Gallego, «Análisis Estático de Contingencias de Potencia Activa en Sistemas Eléctricos de Potencia,» *Scientia et Technica,* vol. 10, nº 25, pp. 1-6, Agosto 2004.

[20] M. Abedi, M. Ehsan, Z. G. Jahromi, M. M. Jamei, «Utilization of Analytical Hierarchy Process in contingency ranking,» de *IEEE Power Systems Conference and Exposition*, Seattle, WA, 2009.

[21] P.R. Bijwe, D.P. Kothari and S.M. Kelapure, «An efficient approach for contingency ranking based on voltage stability,» *Electrical power & energy systems - Elsevier,* vol. 26, pp. 143-149, 2004.

[22] T.F. Halpin, R. Fischl and R. Fink, «Analysis of Automatic Contingency Selection Algorithms,» *IEEE Transactions on Power Apparatus and Systems,* vol. 103, nº 5, pp. 938-945, 1984.

[23] K.F. Schafer and J.F. Verstege, «Adaptive Procedure for Masking Effect Compensation in Contingency Selection Algorithms,» *IEEE Transactions on Power Systems,* vol. 5, nº 2, pp. 539-546, 1990.

[24] G.K. Stefopoulos, F. Yang, G.J. Cokkinides and A.P.S. Meliopoulos, «Advanced Contingency Selection Methodology,» *IEEE Proc.of the 37th Annual North American Power Symposium*, p. 67 – 73, 23-25 Octubre 2005.

[25] G. D. Irisarri and A. M. Sasson, «An Automatic Contingency Selection Method for On-Line Security Analysis,» *IEEE Transactions on Power Apparatus and Systems,* vol. 100, nº 4, pp. 1838-1844, 1981.

[26] R. Caglar, A. Ozdemir, F. Mekic, «Contingency Selection Based on Real Power Transmission Losses,» de *IEEE Electric Power Engineering, PowerTech Budapest*, Budapest, 1999.

[27] Z. J. Meng, Y. Xue and K. L. Lo, «A New Approximate Load Flow Calculation Method for Contingency Selection,» *IEEE PSCE,* pp. 1601-1605, 2006.

[28] R. Sunitha, K.R. Sreerama, T.M. Abraham, «Development of a Composite Security Index for Static Security Evaluation,» *IEEE TENCON,* pp. 1-6, 2009.

[29] H. Mori, D. Tanaka and J. Kanno, «A Preconditioned Fast Decoupled Power Flow Method for Contingency Screening,» *IEEE Transactions on Power Systems,* vol. 11, nº 1, pp. 357-363, 1996.

[30] A. Mohamed, G.B. Jasmon, «Realistic Power System Security Algorithm,» *Proc. IEEE,* vol. 135, nº 2, pp. 98-106, Marzo 1988.

[31] J. Hazra and A.K. Sinha, «A Risk Based Contingency Analysis Method Incorporating Load and Generation Characteristics,» *International Journal of Electrical Power and Energy Systems,* vol. 32, pp. 433-442, 2010.

[32] I. Musirin, T. K. A. Rahman, «Fast Automatic Contingency Analysis and Ranking Technique for Power System Security Assessment,» *Proc. IEEE Student Conference on Research and Development, Putrajaya, Malaysia,* pp. 231-236, 2003.

[33] R.C. Green, L. Wang, and M. Alam, «High Performance Computing for Electric Power Systems: Applications and Trends,» de *IEEE Power and Energy Society General Meeting*, San Diego, CA, USA, 2011.

[34] D. Chavarría-Miranda, Z. Huang, and Y. Chen, «High-Performance Computing (HPC): Application & Use in the Power Grid,» *IEEE Power and Energy Society General Meeting*, pp. 2-7, July 22-26, 2012.

[35] J.C. Mendes, O.R. Saavedra and S.A. Feitosa, «A Parallel Complete Method for Real-time Security Analysis in Power Systems,» *Electric Power System Research Elsevier*, vol. 56, pp. 27-34, 2000.

[36] F. Dong, X. Xu, and X. Zhang, «Parallel Contingency Analysis Solution based on OpenMP,» de *North American Power Symposium*, 2014.

[37] L. Balduino and A. Alves, «Parallel processing in a cluster of microcomputers with application in contingency analysis,» *IEEE/PES Transmission and Distribution Conference and Exposition: Latin America*, p. 285–290, November 2004.

[38] O. Ceylan, H. Dag, and A. Özdemir, «Parallel contingency analysis using diferential evolution based solution for branch outage problem,» de *5th International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control*, North Cyprus, 2009.

[39] G. A. Ezhilarasi and K. S. Swarup, «Parallel contingency analysis in a high performance computing environment,» de *International Conference on Power Systems - ICPS*, Kharagpur, India, 2009.

[40] I. Gorton, Z. Huang, Y. Chen, B. Kalahar, S. Jin, D. Chavarríıa-Miranda, D. Baxter and J. Feo, «A high-performance hybrid computing approach to massive contingency analysis in the power grid,» de *Proceedings IEEE International Conference on e-Science ESCCIENCE*, Washington, USA, 2009.

[41] A. Gopal, D. Niebur, and S. Venkatasubramanian, «DC Power Flow Based Contingency Analysis Using Graphics Processing Units,» *PowerTech 2007*, pp. 731-736, 2007.

[42] Z. Huang, Y. Chen and J. Nieplocha, «Massive Contingency Analysis with High Performance Computing,» de *Proceeding of IEEE PES General Meeting*, Calgary, Canada, July 26-30, 2009.

[43] Y. Chen, Z. Huang, and M. Rice, «Evaluation of Counter-Based Dynamic Load Balancing Schemes for Massive Contingency Analysis on Over 10,000 Cores,» de *Proceeding of IEEE High Performance Computing, Networking Storage and Analysis*, Salt Lake City, Utah, USA, November 12-16, 2012.

[44] W.F. Tinney and W.L. Powell, «Notes on Newton-Raphson methods for solution of AC power flow problema,» *BPA, Portland Oregon,* April, 1971.

[45] F. Milano, Power System Modelling and Scripting, New York: Springer, 2010.

[46] P.H. Haley and M. Ayres, «Super decoupled load flow with distributed slack bus,» *IEEE Trans. Power App. Syst,* Vols. %1 de %2PAS-101, pp. 104-113, 1985.

[47] M. Okamura, Y. Oura, S. Hayashi, K. Uemura and F. Ishiguro, «A new power flow model and solution method including load and generator haracteristics and effects of system control devices,» *IEEE Trans. Power App. Syst,* Vols. %1 de %2PAS-94, nº 3, pp. 1042-1050, 1975.

[48] A. Zobian and M. D. Ilic, «Unbundling of transmission and ancillary services. Part I. Technical issues,» *IEEE Transactions on Power Systems,* vol. 12, nº 2, pp. 539-548, 1997.

[49] J. Meisel, «System incremental cost calculations using the participation factor load-flow formulation,» *IEEE Transaction on Power Systems,* vol. 8, nº 1, pp. 357-363, 1993.

[50] Shiqiong Tong; Miu K.N., «A Network-Based Distributed Slack Bus Model for DGs in Unbalanced Power Flow Studies,» *IEEE Transactions on Power Systems,* vol. 20, nº 2, pp. 835-842, 2005.

[51] S. Tong; M. Kleinberg; and K. Miu, «A distributed slack busmodel and its impact on distribution system application techniques,» *Proc. IEEE International Symposium on Circuits and Systems (ISCAS),* vol. 5, p. 4743–4746, 2005.

[52] S. Tong and K.N. Miu, «Participation factor studies for distributed slack bus models in three-phase distribution power flow analysis,» *Proc. IEEE PES Transm. Distrib. Conf. Exhib., 2005/2006,* p. 92–96, 2006.

[53] S. Tong and K. Miu, «Slack bus modeling and cost analysis of distributed generator installations,» *J. Energy Eng,* vol. 133, nº 3, pp. 111-120, 2007.

[54] M.Z. Kamh and R. Iravani, «A Sequence Frame-Based Distributed Slack Bus Model for Energy Management of Active Distribution Networks,» *IEEE Transactions on Smart Grid,* vol. 3, nº 2, pp. 828 - 836, 2012.

[55] P. Yang, «Modified distributed slack bus load flow algorithm for determining economic dispatch in deregulated power systems,» *Proc. IEEE Power Eng. Soc. Winter Meeting,* p. 1226–1231, 2001.

[56] G.S. Jang, D. Hur, J.K. Park and S.H. Lee, «A modified power flow analysis to remove a slack bus with a sense of economic load dispatch,» *Electric power system research - ELSEVIER,* vol. 73, pp. 137 - 142, 2004.

[57] V.N. Bharatwaj; A. R. Abhyankar; and P. R. Bijwe, «Iterative DCOPF Model Using Distributed Slack Bus,» *IEEE Power and Energy Society General Meeting,* pp. 1 - 7, 2012.

[58] T. Wu; Z. Alaywan; and A. D. Papalexopoulos, «Locational Marginal Price Calculations Using the Distributed-Slack Power-Flow Formulation,» *IEEE Transactions on Power Systems,* vol. 20, nº 2, pp. 1188-1190, 2005.

[59] F. Milano, *Pricing System Security in Electricity Market Models with Inclusion of Voltage Stability Constraints,* Genova, Italia: University of Genova, 2003.

[60] I. Silverio, *The need to coordinate generation and transmission planning and to ensure a secure and efficient reactive power provision: two key aspects of the restructured electricity industry,* PhD Thesis in Ingegneria Elettronica, Informatica ed Elettrica, XIII Ciclo, Università Degli Studi Di Pavia, Italia, 2010.

[61] O. A. Mousavi, *On Voltage and Frequency Control in Multi-area Power System Security,* PhD Thesis, Faculté des Sciences et Techniques de l'Ingénieur, École Polytechniqe Fédérale de Lausanne, Suiza, 2014.

[62] J. J. Grainger and W.D. Stevenson, Análisis de Sistemas de Potencia, México: McGraw Hill, 1996.

[63] W.F. Tinney and C.E. Hart, «Power Flow Solutions by Newton's Method,» *IEEE Transactions on Power Apparatus and Systems,* vol. 86, nº 11, pp. 1449 - 1460, 1967.

[64] J. Arrillaga, C.P. Arnold, Computer Analysis of Power System, Chichester, England: John Wiley and Sons, inc., 1994.

[65] T. M., Athay, «Generation Scheduling and Control,» *Proceedings of the IEEE,* vol. 75, nº 12, pp. 1592-1606, 1987.

[66] D. Das, Electrical Power Systems, New Delhi: New Age International publishers, 2006.

[67] F. M. Gonzalez-Longatt. [En línea]. Available: http://fglongatt.org/OLD/Test_Case_IEEE_30.html. [Último acceso: 30 octubre 2015].

[68] M. Baker and R. Buyya, «Cluster Computing at a Glance,» [En línea]. Available: http://www.buyya.com/cluster/v1chap1.pdf. [Último acceso: 15 Abril 2015].

[69] M. Baekr, «Cluster Computing White Paper,» University of Portsmouth, U.K., 2000.

[70] MathWorks, *Parallel Computing Toolbox, User's Guide*, 2014.

[71] W. Gropp, E. Lusk and A. Skelllum, Using MPI: Portable Parallel Programming with the Message Passing Interface, MIT Press, 1994.

[72] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, «Matpower: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education,» *IEEE Transactions on Power Systems*, vol. 26, nº 1, pp. 12-19, Feb. 2011.

[73] A. Freedman, Diccionario de computación, Madrid, España: McGraw-Hill, 1993.

[74] A. Mejía Mesa, Diccionario Técnico Actualizado, Medellín, Colombia, 1989.