Lámpsakos | N°. 16 | pp. 44-50 | julio-diciembre | 2016 | ISSN: 2145-4086 | Medellín–Colombia

Algoritmo para el descubrimiento del modelo organizacional utilizando el patrón paralelo segmentación de cauce

Algorithm for the discovery of organizational model using cauce pipeline parallel pattern

Alex Rivero Botta
Instituto Superior Politécnico José Antonio Echeverría
La Habana, Cuba
ariverob@ceis.cujae.edu.cu

Ronny Álvarez Pérez
Instituto Superior Politécnico José Antonio Echeverría
La Habana, Cuba
ralavarezp@ceis.cujae.edu.cu

Humberto Díaz Pando, PhD.
Instituto Superior Politécnico José Antonio Echeverría
La Habana, Cuba
hdiazp@ceis.cujae.edu.cu

Lester Guerra Denis, Ing.
Instituto Superior Politécnico José Antonio Echeverría
La Habana, Cuba
lguerra@ceis.cujae.edu.cu

(Recibido el 25-05-2016. Aprobado el 16-11-2016)

Estilo de Citación de Artículo:
H. Diaz, L. Guerra, J. Fernández, E. Acevedo, "Algoritmo para el descubrimiento del modelo organizacional utilizando el patrón paralelo segmentación de CAUCE", Lámpsakos, no. 16, pp 44-50, 2016
DOI: http://dx.doi.org/10.21501/21454086.1969

Resumen. Los avances alcanzados con las tecnologías de la información y las comunicaciones han resultado en un crecimiento de todos los datos almacenados y/o intercambiados electrónicamente. Las técnicas de minería de procesos son capaces de extraer conocimiento de los registros de eventos comúnmente disponibles en los sistemas de información actuales. Asimismo, el procesamiento paralelo es un tipo de procesamiento de la información, que permite que se ejecuten varios procesos concurrentemente, logrando impresionantes poderes de cálculo.

El trabajo que a continuación se presenta es un diseño del algoritmo de minería de procesos para el descubrimiento del modelo organizacional utilizando el patrón paralelo segmentación de cauce, aprovechando las operaciones que en su diseño secuencial se ejecutan independientemente. Los experimentos realizados y los resultados obtenidos con las pruebas de hipótesis no paramétricas de Mann-Whitney arrojan que la solución propuesta disminuye los tiempos de ejecución en relación a su variante secuencial.

Palabras clave: modelo organizacional, patrón paralelo segmentación de cauce.

Abstract. The progress made with the information technology and communications have resulted in a growth of all data stored and / or exchanged electronically. The process mining techniques are able to extract knowledge from the common-mind records events available in the current information systems. Likewise, parallel processing is a type of information processing, which allows multiple processes to run concurrently, achieving impressive powers of calculation.

The work presented below is an algorithm design process mining Organizational Miner using the pipeline parallel pattern, according to its design operations in sequential run independently. The experiments conducted and results obtained with non-parametric tests Mann-Whitney hypothesis that the proposed solution yield decreases execution times relative to its sequential variant.

Keywords*:* organizational model, pipeline parallel pattern.

1. INTRODUCCIÓN

Los avances alcanzados con las tecnologías de la información y las comunicaciones han permitido recopilar y almacenar gran cantidad de información de todo tipo de procesos: industriales, empresariales, bancarios y publicitarios, entre otros. El registro de los eventos de un proceso, en los actuales sistemas de información, hace posible aplicar técnicas de minería de procesos, capaces de extraer conocimiento de ellos.

Con el transcurso del tiempo, son cada vez mayores los datos almacenados en estos sistemas, propiciándose un escenario favorable para el descubrimiento de modelos en una organización. Desafortunadamente, un crecimiento de los datos implica un mayor tiempo de procesamiento, careciendo, en muchos casos, de la operatividad necesaria para la extracción de conocimientos y la toma de decisiones que este implica.

El procesamiento paralelo es un tipo de procesamiento de la información que permite que se ejecuten varios procesos concurrentemente, logrando impresionantes poderes de cálculo.

El trabajo que a continuación presentamos propone un diseño del algoritmo para el descubrimiento del modelo organizacional basado en el patrón de computación paralela Segmentación de Cauce.

2. DESARROLLO

La minería de procesos es una técnica de administración de procesos que permite analizar los procesos de negocios de acuerdo con un registro de eventos. A través de esta actividad se desea extraer conocimiento desde los registros de evento de los procesos almacenados por los sistemas de información [1].

Dentro de las técnicas de minería de procesos existen varias clasificaciones, las mismas son:

- Descubrimiento: No existe un modelo a priori, es decir, sobre la base de un log de eventos se construye un modelo de proceso que puede ser descubierto basado en eventos de bajo nivel [1].

- Análisis de Conformidad: En el análisis de conformidad hay un modelo a priori. Este modelo se compara con el registro de eventos y se analizan las discrepancias entre el registro y el modelo. Comprobación de conformidad puede ser utilizada para detectar las desviaciones para enriquecer el modelo [1].

Extensión: En los algoritmos de tipo extensión también existe un modelo a priori. Este modelo se amplía con un nuevo aspecto o perspectiva, es decir, el objetivo no es para comprobar la conformidad, sino para enriquecer el modelo. Un ejemplo es la extensión de un modelo de proceso con los datos de rendimiento [1].

2.1. Algoritmo para el descubrimiento del modelo organizacional

El algoritmo para el descubrimiento del modelo organizacional es un algoritmo de minería de procesos del tipo descubrimiento cuyo objetivo es procesar un log de eventos para generar un modelo donde queden representados los originadores de tareas en un proceso dado, relacionados con las tareas que los mismos comienzan. Para ello los datos son cargados desde una base de datos existente en la red del centro.

Cada evento del registro tiene un nombre del originador, que es la persona o sistema que realiza la tarea. Los eventos además contienen tareas que no son más que actividades que pueden ser ejecutadas por uno o más originadores. La salida de dicho algoritmo es un modelo que contiene una lista de originadores y cada originador contiene una lista de tareas ejecutadas por él.

Un pseudocódigo de dicho algoritmo aparece en la Figura 1.

```
ALGORITMO OrgMiner (log: Registro de Eventos)
   Iterar sobre el log
     Mientras existan eventos
        SI evento.originador no está en modelo
           Añadir originador al modelo y su tarea
        SINO
           SI evento.tarea no está en la lista de tareas
              del originador.
              Añadir tarea a la lista de tareas del
              originador.
   FIN Iterar
   Devolver modelo
FIN Algoritmo
```

Fig. 1. Pseudocódigo secuencial del algoritmo para el descubrimiento del modelo organizacional

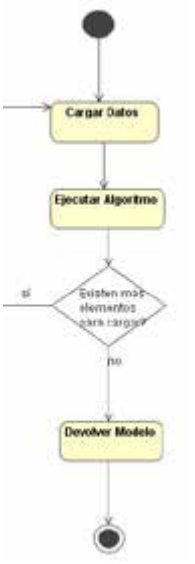

Fig. 2. Diagrama de flujo del algoritmo Minería Organizacional visto como 2 etapas

Para poder realizar este procesamiento es necesario que el algoritmo cargue los datos de un registro de eventos. Debido al tamaño de estas fuentes de información en la actualidad, es necesario que la carga de los datos se haga por intervalos; ya que en algún momento no será posible cargar todos los datos de una sola vez en la memoria interna de la computadora.

Debido a experiencias anteriores se conoce que la carga de los datos representa un factor importante que influye en el tiempo de ejecución del algoritmo de manera significativa. Por ello la presente investigación se centra en ver el algoritmo como un proceso de 2 etapas, la primera, la carga de los datos y la segunda el procesamiento de los mismos. Esta óptica puede ser comprendida de una mejor manera a través de la Figura 2.

2.2. Patrones de computación paralela

Los patrones de programación paralela son una combinación recurrente de distribución de tareas y acceso a datos que resuelve un problema específico dentro del diseño de algoritmos paralelos. Estos patrones a menudo están compuestos o surgen de patrones de diseño no paralelos. Los mismos no están sujetos a arquitecturas de hardware, lenguajes de programación o sistemas, por lo que pueden ser implementados para distintas combinaciones de los elementos anteriormente descritos [2].

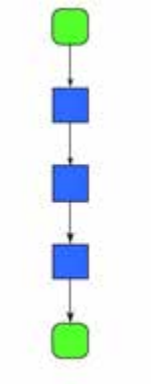

Fig. 3. Esquema del Patrón Segmentación de Cauce. Tomado de [2]

2.2.1. Patrón Segmentación de Cauce

Dentro de los patrones de programación paralela se encuentra el patrón Segmentación de Cauce. El mismo consiste en una cadena de procesos conectados de forma tal que la salida de cada elemento en la cadena es la entrada del próximo. Permiten la comunicación y sincronización entre procesos. Es común el uso de buffer de datos entre elementos consecutivos [2].

Este proceso consiste en una secuencia lineal de etapas (segmentos de código de un algoritmo mayor). Cada etapa realiza una transformación en los datos y la pasa a la siguiente para continuar en el procesamiento de los mismos. La Figura 3 muestra un esquema de dicho patrón. En esta figura los elementos verdes representan los datos de entrada y la salida del algoritmo. Los elementos azules representan las diferentes etapas en las que es dividido el mismo.

Existen 2 tipos de tuberías, las series y las de estados paralelos. En las series en cada etapa se procesan los datos de uno en uno pero varias etapas se ejecutan al mismo tiempo por lo que se dice que ocurre paralelismo de tareas. En el segundo tipo las etapas procesan datos en paralelo y al igual que en las series varias etapas están activas al mismo tiempo por lo que se dice que ocurre paralelismo de datos y de tareas. La velocidad con la que se ejecutarán ambos tipos se verá limitada por el tiempo que demore la ejecución de la etapa más lenta del algoritmo.

2.2.2. Diseño del algoritmo haciendo uso del patrón Segmentación de Cauce

Como fue descrito anteriormente el algoritmo es enfocado de manera simple como un proceso de carga y procesamiento. De ahí surgen las dos etapas que tendrá la segmentación de cauce. La primera donde se cargarán los datos desde el registro de eventos y la segunda etapa recibirá los datos cargados en la etapa anterior y los procesará con el objetivo de obtener un modelo donde queden representados los originadores con la lista de tareas que origina cada uno de ellos. De esta manera quedará conformada una tubería serie ya que no contendrá ninguna etapa paralela.

En la primera iteración solo se cargarán datos debido a que no se puede ejecutar el algoritmo en paralelo porque todavía no existen datos para procesar. En las próximas iteraciones sí se ejecutan ambas etapas en paralelo hasta llegar a la última, donde sólo se ejecutará el algoritmo ya que se han cargado todos los datos existentes en la base de datos.

En la Figura 5 se muestra un pseudocódigo de la versión paralela del algoritmo.

```
ALGORITMO paralelo (log: Registro de Eventos)
CargarDatos
    Repetir numIteraciones veces
        Ejecutar en paralelo
            EjecutarAlgoritmo
            CargarDatos
        Fin Ejecutar en Paralelo
    Fin Repetir
EjecutarAlgoritmo
Devolver modelo
Fin Algoritmo
```

Fig. 4. Pseudocódigo paralelo del algoritmo para el descubrimiento del modelo organizacional utilizando el patrón Segmentación de Cauce.

De esta manera el diseño secuencial que fue examinado en la Figura 3 quedaría de la siguiente manera.

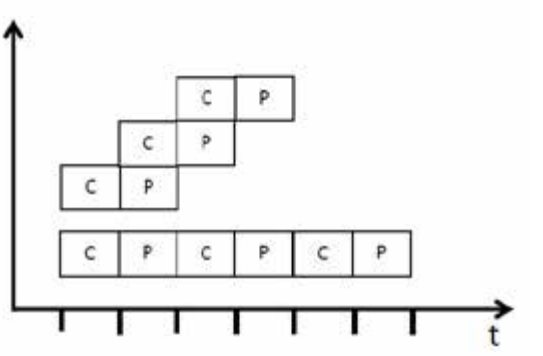

Fig. 5. **Representación de la ejecución paralela del algoritmo.**

Mediante el análisis de esta figura se podría suponer que esta implementación debe reducir los tiempos de ejecución de la versión secuencial del mismo. En el siguiente epígrafe validaremos esta hipótesis.

2.3. Experimentos realizados

Primeramente se realizó un experimento donde se hicieron 10 ejecuciones de las versiones paralela y secuencial, con el objetivo de comprobar si se logró disminuir los tiempos de ejecución del mismo, realizando intervalos de carga de 100 000 elementos en cada iteración. Luego se realizó un segundo experimento, donde se aumentaron los intervalos de carga a 500 000 para comprobar el impacto que tiene este factor en los tiempos de ejecución del algoritmo. Todos los experimentos fueron realizados sobre una base de datos existente en la red del centro, la cual tiene un total de 1 408 539 elementos en la tabla de donde se obtienen los datos para la ejecución del algoritmo.

Estos experimentos fueron realizados en una computadora con las siguientes características:

Procesador Intel(R) Core(TM)2 Duo CPU E7300 @ 2.66GHz 2.67GHz con 2 hilos de ejecución.

2GB de Memoria RAM.

Además para cada uno de los experimentos se calcularon las siguientes métricas de rendimiento:

Aceleración. La aceleración (*Speed Up*) es la relación entre el tiempo de ejecución utilizando solo un procesador y el tiempo de ejecución utilizando varios procesadores [3]. La aceleración representa cuantas veces es más rápido un programa paralelo con res-

pecto a uno secuencial, resolviendo ambos el mismo problema. La aceleración se calcula como se define en (1).

$$S=\frac{T_S}{T_p} \qquad (1)$$

Donde:

S: Aceleración.

Ts: Tiempo de ejecución del algoritmo secuencial.

Tp: Tiempo de ejecución del algoritmo paralelo.

Eficiencia. La eficiencia puede ser definida como la parte del tiempo que los procesadores dedican al trabajo útil, o sea, a las tareas. La eficiencia muestra cuán bien se han utilizado los procesadores. La fórmula para calcularla se define en (2).

$$E=\frac{S_p}{P}=\frac{T_S}{P\times T_p} \qquad (2)$$

Donde:

E: Eficiencia.

Sp: Aceleración asociada a la ejecución paralela del algoritmo.

P: Número de hilos de ejecución del procesador.

2.3.1. Experimento con intervalos de carga de 100 000 elementos

Para comprobar si la versión paralela del algoritmo logró reducir los tiempos de ejecución del mismo, se realizaron 10 ejecuciones de cada una de las versiones. Los tiempos resultantes de las mismas se muestran en la tabla 1.

Para comprobar si realmente se disminuyeron los tiempos de ejecución se realizó una prueba de hipótesis. La misma se describe a continuación.

TABLA 1. Tiempos de la ejecución de ambas versiones del algoritmo con intervalos de carga de 100 000 elementos

	Tiempo del algoritmo secuencial (s)	Tiempo del algoritmo paralelo (s)
	55.99	50.06
	55.56	50.20
	57.00	52.26
	59.06	50.49
	58.00	48.80
	56.68	48.83
	55.85	50.53
	60.54	48.54
	54.67	50.24
	55.06	48.47
Promedio	56.84	49.84
Máximo	60.54	52.26
Mínimo	54.67	48.47

Las hipótesis evaluadas fueron:

H_0: Ts = Tp
H_1: Ts > Tp
Con α = 0.05
Donde:
H_0: Hipótesis nula
H_1: Hipótesis alternativa
α: Máximo nivel de riesgo aceptable para rechazar una hipótesis nula verdadera.
Ts: Tiempo del algoritmo secuencial.
Tp: Tiempo del algoritmo paralelo.

Para probar las hipótesis se utilizó la prueba no paramétrica Mann-Whitney. Se seleccionó esta por no contar con evidencias de que los datos siguen una distribución normal. El resultado que devuelve esta prueba es un valor p, que indica la probabilidad de obtener la mediana de la muestra si H_0 es verdadera. Lo que se espera es que al ejecutar la prueba, la misma devuelva un valor $p < \alpha$, ya que si esto ocurre se rechaza H_0 y se concluye que el tiempo de ejecución secuencial es mayor que el paralelo.

La prueba fue realizada con el software estadístico Minitab. Los resultados de la ejecución de la misma con los valores mostrados en la Tabla 1, arrojó los resultados que se muestran en la Figura 6.

Como se muestra en la Fig 6, el valor p=0.0001 es menor que 0.05, por lo que se puede concluir que la versión paralela del algoritmo disminuye los tiempos de ejecución del mismo.

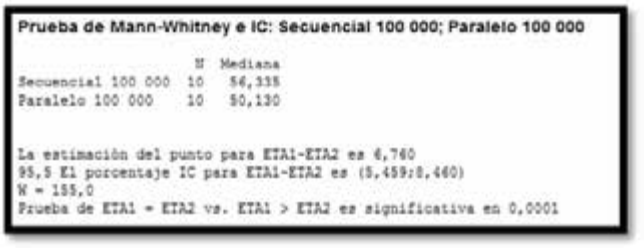

Fig. 6. Resultados de la ejecución de la prueba de Mann-Whitney para el experimento 1.

TABLA 2. Tiempos de la ejecución del algoritmo paralelo con intervalos de carga 100 000 y 500 000 elementos

	Tiempo del algoritmo secuencial (s)	Tiempo del algoritmo paralelo (s)
	50.06	23.50
	50.20	24.43
	52.26	25.46
	50.49	23.65
	48.80	23.27
	48.83	23.21
	50.53	23.07
	48.54	24.26
	50.24	22.96
	48.47	25.17
Promedio	49.84	23.89
Máximo	52.26	25.46
Mínimo	48.47	22.96

El cálculo de las métricas asociadas utilizando los tiempos promedio de ambas versiones arrojó los siguientes resultados:

S = 56.841/49.482 = 1.140423

E = 1.140423/2 = 0.57

2.3.2. Experimento con intervalos de carga de 500 000 elementos

Posteriormente se realizó un experimento para comprobar cómo un aumento el tamaño de los intervalos de carga podría influir en los tiempos de ejecución del algoritmo. Para ello se realizaron 10 ejecuciones de la versión paralela del mismo, con el objetivo de compararlos con su homólogo cargando elementos a intervalos de 100 000. La tabla de comparaciones resultante se muestra a continuación:

Para comprobar si realmente se disminuyeron los tiempos de ejecución se realizó una prueba de hipótesis. La misma se describe a continuación.

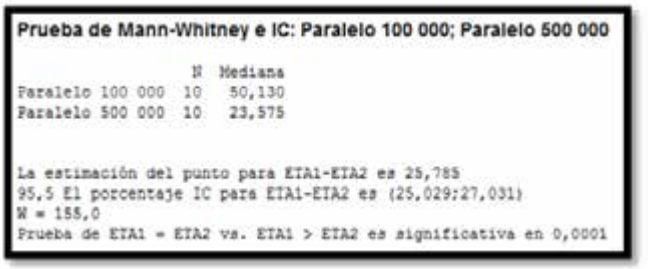

Fig. 7. Resultados de la ejecución de la prueba de Mann-Whitney para el experimento 2.

Las hipótesis evaluadas fueron:

H_0: $T_{100\,000} = T_{500\,000}$

H_1: $T_{100\,000} > T_{500\,000}$

Con $\alpha = 0.05$

Donde:

H_0: Hipótesis nula.

H_1: Hipótesis alternativa.

$T_{100\,000}$: Tiempo del algoritmo paralelo con intervalos de carga de 100 000 elementos.

$T_{500\,000}$: Tiempo del algoritmo paralelo con intervalos de carga de 500 000 elementos.

α: Máximo nivel de riesgo aceptable para rechazar una hipótesis nula verdadera.

Se realizó la prueba estadística descrita para el experimento anterior. El resultado de ejecutar la misma con los valores que se muestran en la Tabla 2 se muestra en la Figura 7.

Como se muestra en la Figura 7, el valor p=0.0001 es menor que 0.05, por lo que se puede concluir que cargar los datos en intervalos de 500 000 elementos disminuye los tiempos de ejecución del algoritmo.

2.3.3. Conclusiones de los experimentos

Los experimentos realizados han demostrado que aumentando el tamaño de los intervalos de carga de datos, se puede reducir en gran medida los tiempos de ejecución del algoritmo. Por tanto se recomienda realizar futuros estudios con tamaños mayores en los intervalos de carga de datos con el objetivo de determinar qué tamaño sería el óptimo para la

ejecución del mismo sobre registros de eventos de mayor dimensión. Además, estos experimentos han demostrado que mediante la implementación del algoritmo para el descubrimiento del modelo organizacional utilizando el patrón Segmentación de Cauce y el modelo de programación paralela OpenMP, se logran disminuir los tiempos de ejecución con respecto a la versión secuencial del mismo.

3. TRABAJOS FUTUROS

Los resultados expuestos indican que el uso del patrón de computación paralela Segmentación de Cauce, en combinación con el modelo de programación OpenMP, disminuye el tiempo de ejecución del algoritmo secuencial para el descubrimiento del modelo organizacional. Cabe entonces como trabajo futuro analizar qué otros patrones de computación paralela permiten diseñar el algoritmo, qué otros modelos de computación paralela se podrían utilizar para implementarlo, incluyendo los modelos de programación en memoria distribuida, y qué otras estructuras de datos permiten una gestión más eficiente de los datos, con el objetivo de disminuir, aun más, el tiempo de ejecución del algoritmo.

4. CONCLUSIONES

Los experimentos realizados y los resultados obtenidos con las pruebas de hipótesis no paramétricas de Mann-Whitney arrojan que la solución propuesta disminuye los tiempos de ejecución en relación a su variante secuencial.

Adicionalmente, se probó que aumentando el tamaño de la carga a 500 000 elementos también se reducen los tiempos de ejecución.

REFERENCIAS

[1] Wil van der Aalst, A.A., Ana Karla Alves de Medeiros, Franco Arcieri, Thomas: et al., *Manifiesto sobre Minería de Procesos.*

[2] Michael McCool, A.D.R., James Reinders. “Structured Parallel Programming Patterns for Efficient Computation”.

[3] **Cairós, S.G.:** *Paralelización de la obtención de datos de entrada del modelo de concentraciones de HYSPLIT. Serie de Notas Técnicas Digitales del Centro de Investigación Atmosférica de Izaña* 2009.

[4] McCool, M. “Structured Parallel Programming”, 2012.