Simulador para la evaluación de algoritmos para la gestión de recursos compartidos en sistemas distribuidos

Simulator for the evaluation of algorithms for the management of shared resources in distributed systems

David Luis la Red Martínez
Universidad Nacional del Nordeste, Corrientes, Argentina
lrmdavid@exa.unne.edu.ar
ORCID: 0000-0003-2038-6468

Federico Agostini
Universidad Nacional del Nordeste, Corrientes, Argentina
fagostini@conicet.gov.ar

Julio César Acosta
Universidad Nacional del Nordeste, Corrientes, Argentina
julioa@exa.unne.edu.ar

Stella Gerzel
Universidad Nacional del Chaco Austral, Sáenz Peña, Argentina
stellagerzel@uncaus.edu.ar

Leandro Latyn
Universidad Nacional del Chaco Austral, Sáenz Peña, Argentina
leandrolatyn@uncaus.edu.ar

doi: https://doi.org/10.36825/RITI.10.20.006

Recibido: Diciembre 31, 2021
Aceptado: Marzo 25, 2022

Resumen: En los sistemas de procesamiento distribuido, especialmente cuando los procesos constituyen grupos colaborativos, es necesario que, al hacerse la asignación de recursos a los procesos, se tomen decisiones basadas en acuerdos respecto del acceso a recursos; las decisiones pueden estar relacionadas con la realización de determinada actividad que requiera o no la sincronización de los procesos, requiriendo el uso de recursos compartidos en la modalidad de exclusión mutua. Así surge el siguiente interrogante: ¿Cuáles son los modelos de decisión y los operadores de agregación que habrá que generar incorporando la perspectiva cognitiva global a los modelos clásicos para hacer más inteligente la toma de decisiones en la gestión de recursos y grupos de procesos teniendo en cuenta la autorregulación? ¿Cómo se implementarán los algoritmos de los distintos modelos de decisión? ¿Cómo validar los nuevos algoritmos propuestos comparándolos entre sí y con los algoritmos tradicionales? Para ello es necesario disponer de un simulador que implemente los algoritmos tradicionales y los nuevos propuestos y permita observar su comportamiento y resultados ante diferentes tipos de cargas de trabajo. Se consideran modelos clásicos para acceder a recursos compartidas en la modalidad de

exclusión mutua utilizando regiones críticas al algoritmo centralizado, al algoritmo distribuido de Lamport, Ricart y Agrawala, al algoritmo de anillo de fichas, entre otros. Se presenta un prototipo de simulador para la evaluación del desempeño de los nuevos modelos de decisión y operadores de agregación propuestos frente a los principales modelos tradicionales.

Palabras clave: *Sistemas Operativos, Sistemas Distribuidos, Gestión de Recursos y Procesos, Simulador de Algoritmos de Gestión de Recursos y Procesos, Modelos de Decisión.*

Abstract: In distributed processing systems, especially when processes constitute collaborative groups, it is necessary that, when allocating resources to processes, decisions are made based on agreements regarding access to resources; decisions may be related to the performance of a certain activity that requires or not the synchronization of processes, requiring the use of shared resources in the mode of mutual exclusion. Thus, the following question arises: What are the decision models and aggregation operators that will have to be generated by incorporating the global cognitive perspective to the classical models to make more intelligent decision making in the management of resources and groups of processes taking into account self-regulation? How will the algorithms of the different decision models be implemented? How to validate the new proposed algorithms by comparing them with each other and with traditional algorithms? For this, it is necessary to have a simulator that implements the traditional and the new proposed algorithms and allows to observe their behavior and results under different types of workloads. Classical models are considered for accessing shared resources in the mutual exclusion mode using critical regions to the centralized algorithm, the distributed algorithm of Lamport, Ricart and Agrawala, the token ring algorithm, among others. A prototype simulator is presented to evaluate the performance of the proposed new decision models and aggregation operators against the main traditional models.

Keywords: *Operating Systems, Distributed Systems, Resource and Process Management, Resource and Process Management Algorithm Simulator, Decision Modeling.*

1. Introducción

La proliferación de sistemas informáticos, muchos de ellos distribuidos, en los cuales existen múltiples procesos que cooperan para el logro de una determinada función, hace necesario disponer de modelos de decisión inteligentes que permitan gestionar la asignación de recursos a los procesos intervinientes en los distintos grupos de procesos, considerando el estado global del sistemas distribuido en cuanto a la carga de procesamiento actual, como así también la carga adicional estimada para atender las asignaciones requeridas de recursos a procesos, respetando además la exclusión mutua en el acceso a recursos compartidos.

Es especialmente significativo el caso del acceso a recursos compartidos por parte de distintos procesos, que pueden estar operando en equipos distribuidos, donde el acceso requerido a dichos recursos debe hacerse en la modalidad de acceso exclusivo y con exclusión mutua. El problema consiste entonces en lograr una adecuada gestión de recursos compartidos (entre varios procesos) en sistemas distribuidos de procesamiento (varios procesos ejecutando en distintos nodos, donde los procesos pueden ser parte de la misma aplicación o de aplicaciones distintas), donde puede haber recursos compartidos que sólo se pueden acceder en la modalidad de acceso exclusivo (de a un proceso a la vez) y con exclusión mutua (cuando el recursos compartido ha sigo asignado a un proceso, no puede ser asignado a otro hasta que el primero lo libera).

Para dar solución a la problemática planteada se han propuesto distintos métodos. Ejemplos de lo mencionado se encuentran en [1] y [2], donde se describen los principales algoritmos de sincronización en sistemas distribuidos, en [3], donde se presenta una solución eficiente y tolerante a fallas para el problema de la exclusión mutua distribuida, en [4], [5] y en [6], donde se presentan unos algoritmos para gestionar la exclusión mutua en redes de computadoras, en [7], donde se describen los principales algoritmos de sincronización en sistemas distribuidos, en [8], donde se detallan los principales algoritmos para la gestión distribuida de procesos, los estados globales distribuidos y la exclusión mutua distribuida. Estos temas y otros relacionados también han sido tratados en [9] y [10], etc.

Los modelos de decisión disponibles en la actualidad y generalmente aplicables en los sistemas distribuidos se basan en algoritmos de intercambio de permisos que intentan lograr un acuerdo de todos los procesos

intervinientes para realizar determinadas acciones, como el acceso a un área de memoria compartida a la que se debe acceder en la modalidad de exclusión mutua.

Asimismo, se han generado nuevos modelos de toma de decisiones en grupos de procesos distribuidos [14], que se han ampliado considerando la aplicación de métodos de imputación de datos para aquellos casos de datos de control faltantes, por ejemplo, como consecuencia de problemas en las comunicaciones entre los procesos, y *fuzzyficación* de variables para dar soporte a situaciones donde no es posible o conveniente expresar valores exactos. Estas nuevas propuestas consideran como casos particulares a las soluciones más conocidas y generalmente aplicadas.

Debido a lo indicado precedentemente, surgió la necesidad de desarrollar un entorno de simulación de los distintos algoritmos, los actualmente utilizados y los nuevos propuestos, para diferentes cargas de trabajo, a los efectos de evaluar su funcionamiento, siendo el objetivo de este artículo describir el simulador desarrollado.

Este trabajo se ha estructurado de la siguiente manera: en la sección 2 se describen brevemente algunos trabajos previos; en la sección 3 se explica la metodología utilizada comenzando con una sucinta reseña de las estructuras de datos, la descripción del prototipo de simulador, la indicación de las principales etapas del algoritmo empleado y de los cálculos realizados para los distintos escenarios planteados, para luego mostrar algunos aspectos de la navegación a través de las pantallas; en la sección 4 se incluye un análisis de resultados para los distintos escenarios de simulación con comentarios y discusiones, para luego finalizar con las conclusiones, la mención de las líneas futuras de trabajo, los agradecimientos y las referencias.

2. Trabajos previos

Algunas publicaciones relacionados con simuladores se presentan en [11], donde se analiza el grado de utilidad y usabilidad de un simulador, desarrollado por el profesor de la asignatura y sus alumnos, para la enseñanza de la programación de sistemas; en [12], donde se propone el aprendizaje del funcionamiento de un sistema operativo mediante simulaciones interactivas; en [13] se presenta un simulador que permite a los usuarios implementar rutinas para ampliar varias funcionalidades como gestión de la memoria, el procesamiento de control y cualquier otra función del sistema operativo. En todos los casos mencionados precedentemente se han considerado algoritmos conocidos y generalmente utilizados por los sistemas operativos.

Las publicaciones relacionadas con los nuevos modelos de decisión propuestos son las siguientes:

En [14] se describe detalladamente un nuevo operador de agregación para generar una lista ordenada de asignación de recursos a procesos en sistemas distribuidos incluyendo restricciones de exclusión mutua, pero no se desarrolla un simulador. El artículo ha sido tomado como referencia en varios trabajos posteriores que profundizaron sus desarrollos teóricos y originales propuestas, pero no incluyeron un simulador para evaluarlas con gran número de datos.

En [15] se propone un innovador modelo de decisión para sincronización de procesos en sistemas distribuidos que parte del operador de agregación presentado en [14], pero no se incluye un simulador para evaluar su desempeño.

En [16] se pone el énfasis en el modelado matemático de un operador de agregación para la gestión de recursos en sistemas distribuidos, centrándose en los aspectos teóricos del mismo, pero no en un simulador para evaluarlo.

En [16] y [18] se presenta un nuevo modelo de consenso para la asignación de recursos en sistemas distribuidos partiendo del operador de agregación mencionado en [14] aplicándolo a diferentes escenarios de procesamiento distribuido y a la gestión de tráfico en redes de datos, poniéndose el énfasis en la descripción de los escenarios y en la asignación de paquetes de datos a rutas de transmisión, pero no se incluye un simulador al efecto.

En [17] y [20] se presentan modelos de decisión innovadores para la gestión de procesos en nuevos escenarios de sistemas distribuidos, con una detallada descripción de estos y ejemplos ilustrativos, pero no se menciona un simulador para evaluarlos.

En [21], [22] y [18] se desarrollan propuestas innovadoras para la gestión del tráfico en redes de datos partiendo del operador de agregación propuesto en [14] pero adaptado para brindar un orden de asignación de paquetes de datos a rutas de transmisión considerando el estado de carga de los nodos y de los enlaces además de la prioridad inicial de los paquetes, pero no se incluye un simulador.

En [19], [20] y [21] se proponen métodos innovadores para la asignación de recursos compartidos a procesos distribuidos, especialmente con requerimientos estrictos de consenso para dicha asignación, incluyéndose la gestión de grupos de procesos colaborativos, pero en ningún caso se describe un simulador.

En [22] y [23] se proponen interesantes modelos de decisión y operadores de agregación para la gestión de procesos y recursos utilizando lógica difusa contemplando balanceo dinámico de la carga de trabajo. En estas propuestas es destacable la generalización teórica que se hace del operador de agregación presentado en [14], al que se adapta para soportar diferentes conjuntos de etiquetas lingüísticas y 2-tuplas como datos de entrada en vez de sólo datos numéricos, pero sin incluir un simulador.

En [24] se consideran los desafíos fundamentales de regular la carga y la programación en arquitecturas orientadas al servicio (SOA) cumpliendo los objetivos de rendimiento extremo a extremo según las métricas acordadas, lo cual se considera altamente positivo, pero sin mencionar un simulador que permita evaluar la propuesta.

En [25] se estudia el equilibrio de carga de trabajo en entornos de computación en la nube mediante la migración de máquinas virtuales buscando minimizar el costo de dicha migración, lo cual se considera sumamente positivo para su utilización en grandes centros de datos, pero sin indicar un simulador al efecto.

En [26] se presenta un simulador específico para arquitecturas de sistemas distribuidos paralelos con multinúcleos, contemplando aspectos de distribución de carga y sincronización en un entorno con cientos o miles de núcleos. Esta propuesta, que se considera sumamente interesante, está acotada a escenarios específicos y por el momento no es generalizable.

En [27] se describe un entorno de simulación distribuida llamado COSSIM, que permite simular procesadores, periféricos y redes, y que está orientado a sistemas ciber-físicos. Esta propuesta también es muy interesante pero acotada a sistemas ciber-físicos y por lo tanto no es genérica.

En [28] se presenta la simulación en NS3 del problema de cuello de botella compartido que ocurre en el protocolo MP-TCP; este trabajo es sumamente interesante y se concentra en la transmisión de datos, pero no en la gestión de recursos y procesos en sistemas distribuidos.

De lo enunciado precedentemente surge la conveniencia de desarrollar un simulador específico, primeramente, para los modelos de decisión y operadores de agregación propuestos en [14] y sus derivaciones, permitiendo posteriormente la incorporación de otros métodos para la gestión de recursos y procesos en sistemas distribuidos, llegando finalmente a un simulador genérico capaz de soportar diferentes escenarios y soluciones que vayan apareciendo y que se incorporen al mismo.

3. Metodología

En esta sección se indicarán las estructuras de datos y los escenarios soportados por el simulador, como así también los cálculos requeridos para estos y las pantallas de navegación correspondientes.

3.1. Estructuras de datos y escenarios soportados

El sistema de matrices de datos utilizado es el descrito en [14]. Se trata de grupos de procesos distribuidos en nodos de procesos que acceden a recursos críticos compartidos en la modalidad de exclusión mutua distribuida, debiendo decidirse, ante la demanda de recursos por parte de los procesos, cuáles serán las prioridades para asignar los recursos a los procesos que los requieren.

Se ha desarrollado una aplicación que permite simular un entorno de sistemas distribuidos, en el que existen un conjunto de nodos que comparten recursos, y que interactúan entre sí intercambiando información de control necesaria para gestionar las asignaciones de recursos a procesos. Se considera que existe un *Runtime* (software de tiempo de ejecución) en cada nodo, que actúa como interfaz entre el sistema operativo y las aplicaciones, uno de estos nodos, actúa como nodo central y es el encargado de recibir toda esta información y tomar el control de la gestión de esta.

La simulación se hace a partir de la configuración de los parámetros de entrada, consistentes en recrear un entorno distribuido con un conjunto de nodos, los cuales tendrán distintos niveles de carga computacional, que alojarán a distintos recursos y procesos (independientes o grupales). En este entorno se plantea un macrociclo, que consiste en la recopilación de información de requerimientos de asignaciones de recursos a procesos y de estado de los distintos nodos, que se resuelve con los operadores de agregación específicos.

El operador propuesto en [14] consta de diferentes etapas, las primeras 4, tienen que ver con la información necesaria para resolver la asignación de recursos a procesos: a) Cálculo de la carga computacional actual de los nodos; b) Establecimiento de las categorías de carga computacional y de los vectores de pesos asociados a las mismas; c) Cálculo de las prioridades o preferencias de los procesos teniendo en cuenta el estado del nodo (se las calcula en cada nodo para cada proceso); d) Cálculo de las prioridades o preferencias de los procesos para acceder a los recursos compartidos disponibles (se las calcula en el administrador centralizado de recursos compartidos).

Una vez obtenidos los datos iniciales, se podrán representar 4 escenarios, considerando procesos independientes o grupos de procesos, y aplicando niveles estrictos de consenso o no. Esto permitirá determinar el orden en que se asignarán los recursos y a qué proceso será asignado cada uno de ellos (Fig. 1) La salida del primer escenario (E1) (caso general) sirve de entrada para el segundo (E2) cuando éste se presenta. La salida del segundo escenario (E2) sirve como entrada para el tercer escenario (E3) en caso de presentarse. Si se presentara el cuarto escenario (E4), se tomarán como entrada los datos del segundo paso del escenario 1 (E1).

3.2. Descripción de los cálculos realizados por el simulador para cada escenario

La premisa general para todos los escenarios es que se trata de grupos de procesos distribuidos en nodos de procesos que acceden a recursos críticos compartidos en la modalidad de exclusión mutua distribuida, debiendo decidirse cuáles serán las prioridades para asignar los recursos a los procesos que los requieren.

Figura 1. Escenarios posibles del simulador.

3.2.1. Primer escenario (Función de Asignación para Sistemas Distribuidos - FASD)

Consiste en que los procesos accedan a recursos compartidos en la modalidad de exclusión mutua *pudiendo constituir grupos de procesos* (los procesos independientes son considerados grupos unitarios); los procesos no requieren sincronización (estar activos en sus respectivos procesadores en un mismo lapso de tiempo) y *sin exigencias estrictas de consenso* para lograr el acceso (no se requiere un consenso para asignar de manera consecutiva los recursos solicitados por un proceso o grupo de procesos, es decir, que iniciada la secuencia de asignación de recursos a un proceso, la misma puede ser interrumpida para asignar recursos a otro proceso).

En la Fig. 2 se muestran 3 etapas correspondientes a la resolución de las asignaciones. La primera es obtener la Función de Asignación para Sistemas Distribuidos (FASD), que consiste en realizar rondas de asignaciones, en la que los recursos disponibles se asignan de manera tal, que se asegura la exclusión mutua, es decir, asignando un determinado recurso a un solo proceso a la vez, en cada iteración. La segunda etapa consiste en ordenar la lista final obtenida, de manera tal, que se asignen los recursos de mayor a menor prioridad, esto se denomina Función de Asignación para Sistemas Distribuidos Ordenada (FASDO). La tercera etapa es la concatenación de todas las iteraciones hasta que no queden solicitudes pendientes, esto se denomina Función de Asignación de Sistemas Distribuidos Concatenada (FASDC).

3.2.2. Segundo escenario (Función de Asignación para Sistemas Distribuidos Concatenada Ordenada – FASDCO)

Consiste en que los procesos accedan a recursos compartidos en la modalidad de exclusión mutua pudiendo constituir grupos de procesos (los procesos independientes son considerados grupos unitarios); los procesos no requieren sincronización (estar activos en sus respectivos procesadores en un mismo lapso de tiempo) y deben tener estrictos requisitos de consenso para acceder a los recursos (se requiere un acuerdo para asignar consecutivamente los recursos solicitados por un proceso, es decir, una vez iniciada la secuencia de asignación de recursos, no se puede interrumpir, hasta que el proceso activo libere los recursos).

En la Fig. 3, se muestran las etapas correspondientes a la resolución de las asignaciones para el escenario 2. La primera de ellas consiste en calcular la *Prioridad Final Global del Proceso (PGFP)*, que considera el promedio de prioridades de las asignaciones de recursos que solicita cada proceso, en todas las rondas. La siguiente etapa consiste en ordenar el vector *PGFP* y obtener la *Prioridad Final Global del Proceso Ordenada (PGFPO)*.

El orden final de asignaciones de recursos a procesos, ordenadas por procesos de mayor a menor prioridad, se denominada *Función de Asignación para Sistemas Distribuidos Concatenada Ordenada – FASDCO*.

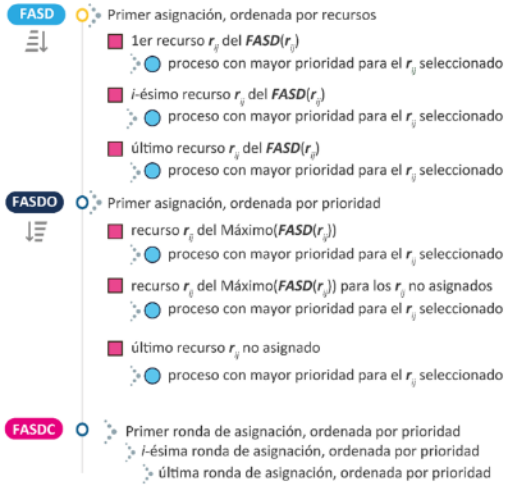


Figura 2. Pasos para obtener la FASD, FASDO y FASDC.

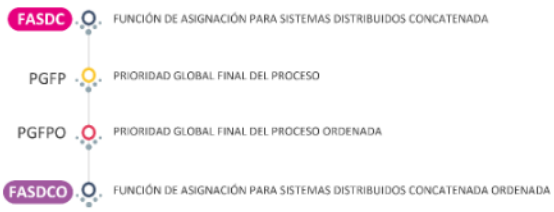


Figura 3. Cálculo de la FASDC a la FASDCO.

Aunque para los recursos los subíndices son distintos, no necesariamente serán recursos distintos, sino que pueden representar a un mismo recurso que se asigna varias veces en las distintas rondas, pero siempre al mismo proceso p_{ek}. La ubicación en la tabla *FASDCO* dependerá de la ubicación en el vector *PGFPO*.

La modalidad con la que trabaja internamente el simulador se puede observar en la Fig. 4, donde la representación de los r_{ij} indica los recursos (cuyo primer subíndice representa al nodo donde se encuentra y el segundo subíndice al propio número de recurso) que se le asignan al proceso p_{ek} (cuyo primer subíndice representa al nodo donde se encuentra y el segundo subíndice al propio número de proceso) en cada ronda.

3.2.3. Tercer escenario (Función de Asignación para Sistemas Distribuidos Ordenada por Grupo Compatible - FASDOGC)

Consiste en que los procesos accedan a recursos compartidos en la modalidad de exclusión mutua *constituyendo grupos de procesos* (los procesos independientes son considerados grupos unitarios y con prioridad grupal cero); los procesos no requieren sincronización (estar activos en sus respectivos procesadores en un mismo lapso de tiempo) y *deben tener estrictos requisitos de consenso* para acceder a los recursos (se requiere un acuerdo para asignar consecutivamente los recursos solicitados por todos los procesos de cada grupo, es decir, una vez iniciada la secuencia de asignación de recursos al grupo, no se puede interrumpir para otorgársela a otro grupos de procesos, hasta que el proceso activo en cada grupo libere los recursos).

En la Fig. 5, se muestran las etapas correspondientes a la resolución de las asignaciones para el escenario 3. La tabla con la *Función de Asignación para Sistemas Distribuidos Concatenada Ordenada (FASDCO)*, se utilizará como punto de partida para generar una nueva función y obtener una nueva lista ordenada de procesos y recursos, pero considerando las prioridades de grupos de procesos.

El primer paso consiste en obtener la *Prioridad Global de Grupo Ordenada*, la cual ordenará los grupos de procesos en forma descendente, según la prioridad promedio calculada de cada uno, en todas sus asignaciones.

El resultado, consiste en obtener la tabla denominada *Función de Asignación para Sistemas Distribuidos Ordenada por Grupo – FASDOG*. Se establecerá el orden de asignación por prioridades de grupo de los procesos para acceder a sus recursos y se establecerá el orden en el que se asignará cada uno.

Se designarán rondas de asignaciones a grupos, y para respetar la condición de exclusión mutua, habrá que establecer subrondas de asignaciones para aquellos procesos que no generen incompatibilidad en el acceso a los recursos, y otras subrondas para atender las asignaciones de recursos a procesos que no han sido atendidos en la subronda anterior, por estar en situación de incompatibilidad. El resultado de esta operación se denomina *Función de Asignación para Sistemas Distribuidos Ordenada por Grupo Compatible (FASDOGC).* Aquí se establecerá el orden de asignación por prioridades de grupo, de los procesos para acceder a sus recursos y se establecerá el orden en el que se asignará cada uno, considerando subrondas de asignación para procesos que no generen incompatibilidad en el acceso al recurso.

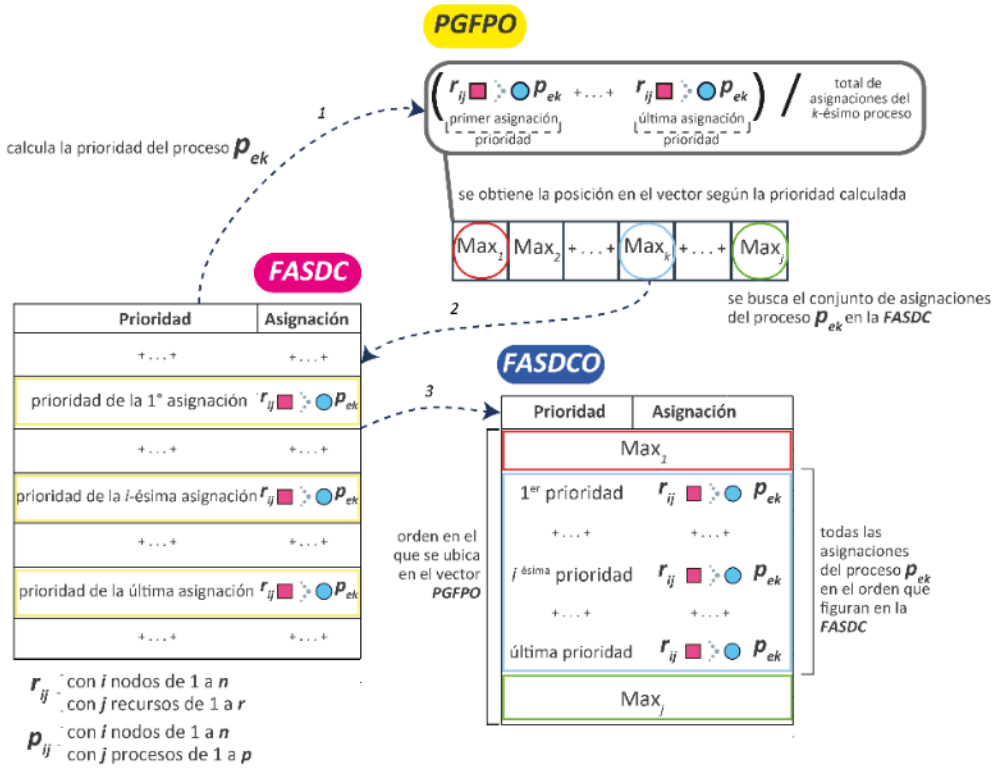


Figura 4. Cálculo de prioridades para el proceso p_{ek} con mayor prioridad en PGFPO.

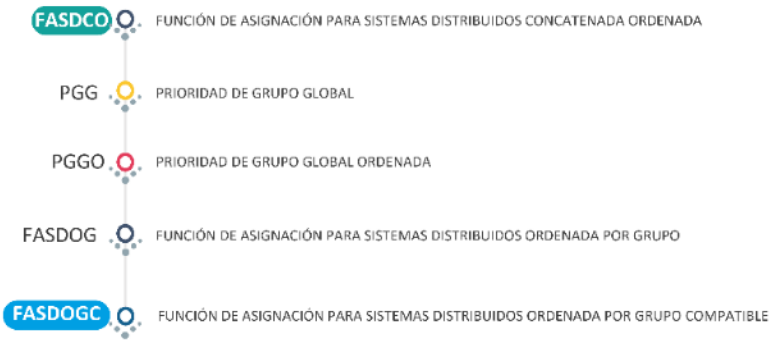


Figura 5. Cálculo de la FASDCO a la FASDOGC.

La modalidad con la que trabaja internamente el simulador se puede observar en la Fig. 6, que considera el promedio global de prioridades que cada grupo de procesos tiene sobre todos los recursos de todas sus asignaciones en las diferentes rondas *PGGO*, pero para la asignación global final del grupo, establece subrondas compatibles de asignación, dado que en un mismo grupo puede haber procesos que requieran un mismo recurso, por lo tanto, los procesos de mayor prioridad serán asignados en una primer subronda y los que generan incompatibilidad se asignarán en una siguiente subronda. Es decir, la elección del grupo de procesos al que se le otorgarán recursos se establece con el promedio global de prioridades en todas las asignaciones, pero el orden en el que se deben realizar esas asignaciones se establece por medio de subrondas compatibles de asignación.

La representación de los r_{ij}, indican los recursos (cuyo primer subíndice representa al nodo donde se encuentra y el segundo subíndice al propio número de recurso) que se les asignan a los procesos p_{ij} (cuyo primer subíndice representa al nodo donde se encuentra y el segundo subíndice al propio número de proceso) en cada ronda. Aunque para los recursos/procesos los subíndices son iguales, no necesariamente serán los mismos recursos/procesos, sino que pueden representar a distintos recursos/procesos que se asignan varias veces en las distintas rondas, pero siempre al mismo grupo g_h.

3.2.4. Cuarto escenario (Función de Asignación para Sistemas Distribuidos Ordenada por Grupo – FASDOG)
Consiste en que los procesos accedan a recursos compartidos en la modalidad de exclusión mutua *constituyendo grupos de procesos* (los procesos independientes son considerados grupos unitarios y con prioridad grupal cero); los procesos no requieren sincronización (estar activos en sus respectivos procesadores en un mismo lapso de tiempo) y *no requieren estrictos requisitos de consenso* para acceder a los recursos.

En la Fig. 7, se muestran las etapas correspondientes a la resolución de las asignaciones para el escenario 4. La tabla obtenida con la *Función de Asignación para Sistemas Distribuidos Ordenada (FASDO)*, se utilizará como punto de partida para generar una nueva función y obtener una nueva lista ordenada de procesos y recursos.

El primer paso consiste en obtener la *Prioridad de Grupo en la Asignación (PGA)*, que considera el promedio de las asignaciones de los procesos que pertenecen a cada grupo, seguidamente habrá que ordenar los grupos de procesos en forma descendente, según la prioridad promedio calculada de cada uno, para obtener la *Prioridad de Grupo en la Asignación Ordenada (PGAO)*.

El paso final consiste en obtener la tabla denominada *Función de Asignación para Sistemas Distribuidos Ordenada por Grupo – FASDOG*, donde se establecerá el orden de asignación a los grupos de procesos para que puedan acceder a los recursos solicitados y el orden en el que se asignará cada uno, dentro de cada grupo.

La modalidad con la que trabaja internamente el método considera que el grupo de procesos al cual se le otorgan los recursos se determina con el promedio global de prioridades en todas las asignaciones en la ronda PGGO, pero el orden en el que se deben realizar esas asignaciones respeta el de la tabla FASDO, para cada grupo de procesos.

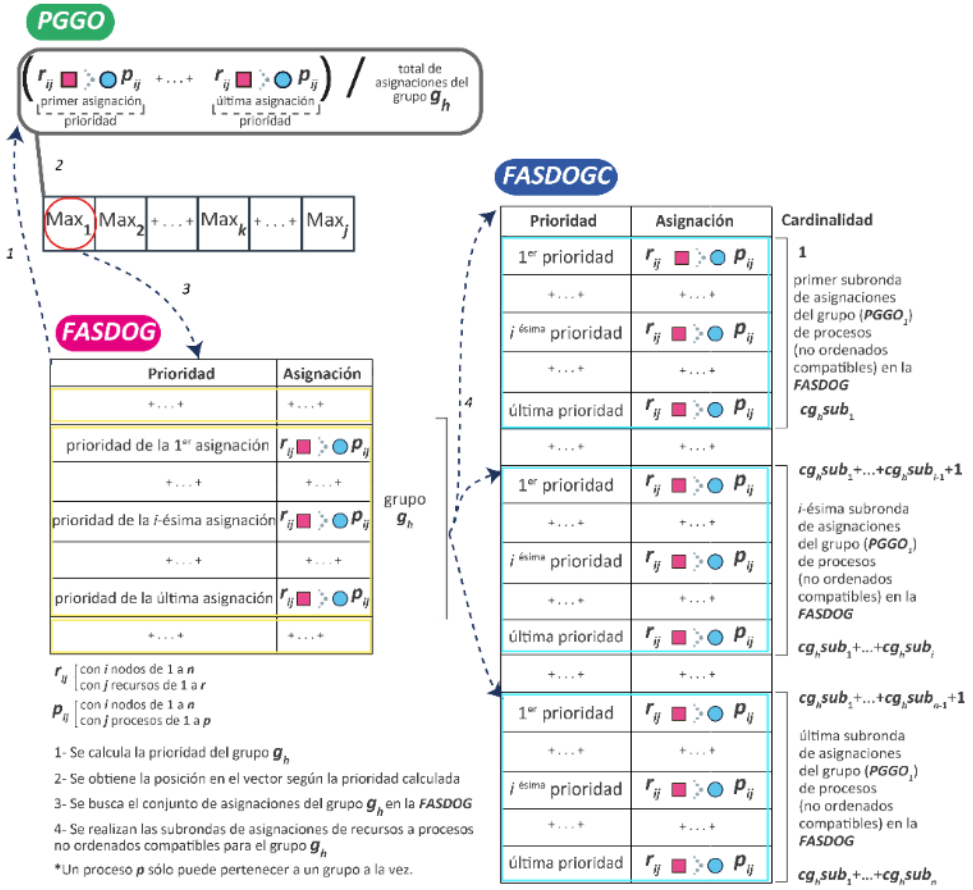


Figura 6. Listado final del grupo de mayor prioridad en la PGGO.

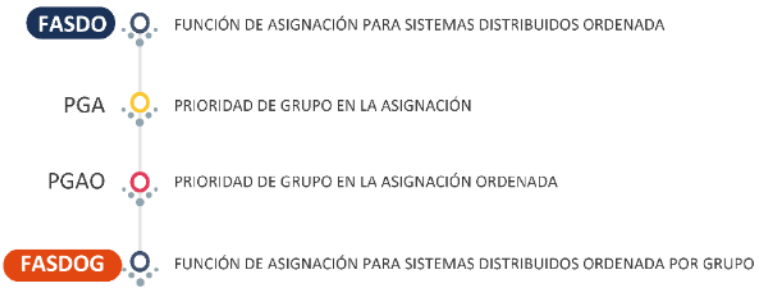


Figura 7. Cálculo de la FASDO a la FASDOG.

La representación de los recursos r_{ij}, indican los recursos (cuyo primer subíndice representa al nodo donde se encuentra y el segundo subíndice al propio número de recurso) que se les asignan a los procesos p_{ij} (cuyo primer subíndice representa al nodo donde se encuentra y el segundo subíndice al propio número de proceso) en cada ronda. Aunque para los recursos/procesos los subíndices son iguales, no necesariamente sean los mismos recursos/procesos, sino que pueden representar a distintos recursos/procesos que se asigna varias veces en las distintas rondas. Cabe destacar que los procesos deben pertenecer a un sólo grupo, ver Fig. 8.

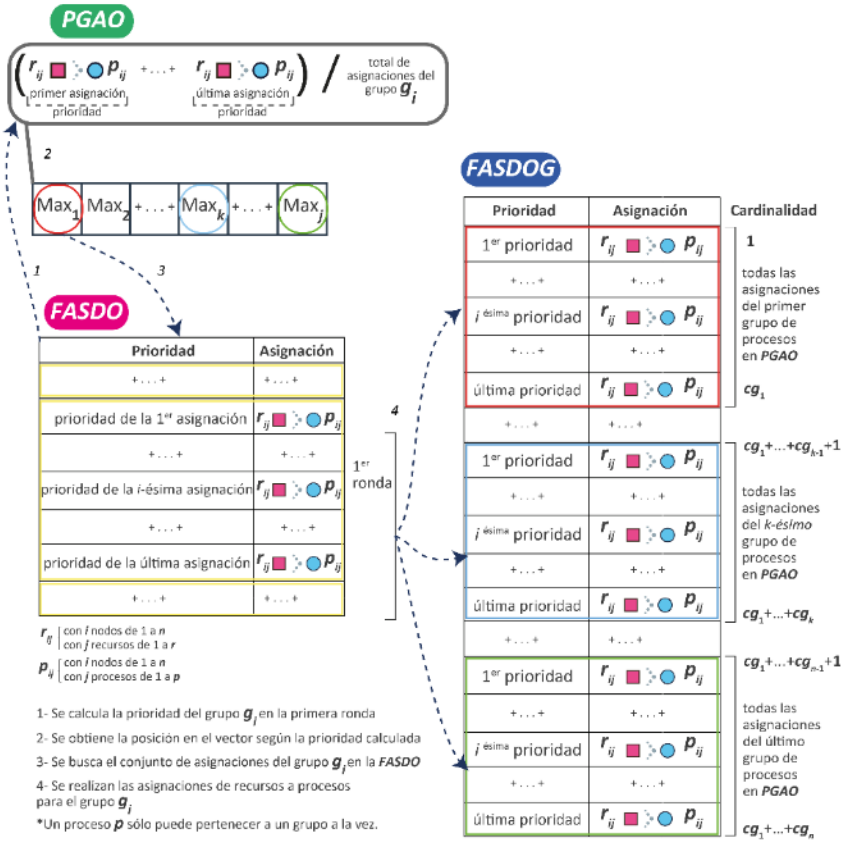


Figura. 8. Cálculo desde la FASDO a la FASDOG en la primera ronda.

3.3. Navegación a través de las pantallas del simulador

En la primera pantalla (Fig. 9), se encuentra el menú principal, donde se puede acceder a los diferentes escenarios desarrollados (se tiene previsto incorporar más escenarios en los próximos meses). También, se pueden visualizar el conjunto de pasos iniciales para gestionar la información de control principal, que sirven como entrada para cada uno de los escenarios.

Figura 9. Pantalla principal de la plataforma.

Esta información consiste en una cronología que se debe seguir para completar los datos de entrada, realizar los primeros procesamientos de información proveniente de los distintos nodos, que el nodo principal, a través de su

Runtime, deberá procesar para obtener una lista ordenada final de asignación de recursos a procesos. Este último paso mencionado, dependerá del escenario elegido.

Cada uno de los pasos mencionados a continuación, son guardados en una base de datos MySQL, con el objetivo de poderlos usar en los distintos escenarios, incluso la posibilidad de generar valores históricos correspondientes a cada macrociclo (Fig. 10).

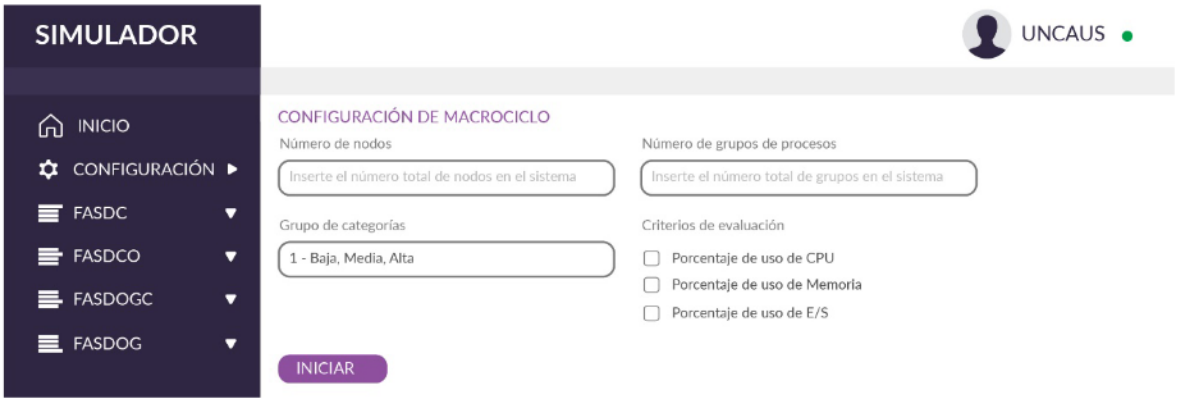


Figura 10. Configuración de macrociclo. Primer paso.

El siguiente paso es la configuración del vector de pesos asociado a los criterios de evaluación de carga computacional. Es necesario calcular el vector de pesos final que se utilizará en el proceso de agregación para determinar el orden o la prioridad de acceso a los recursos (Fig. 11).

Se debe seleccionar el número de procesos y recursos existentes en el sistema, se podrán asignar manualmente o de forma aleatoria. Además, se deberán introducir la cantidad de recursos y procesos disponibles en cada nodo, esto también tendrá una opción aleatoria (Fig. 12).

Otras pantallas están relacionadas con la asignación de los procesos a los grupos, los pedidos de recursos por parte de los procesos y los cálculos de las preferencias en cada nodo.

Una vez obtenidos todos los datos de entrada, se procede a la resolución de los distintos escenarios, consistentes en obtener la lista final ordenada de asignaciones de recursos a procesos.

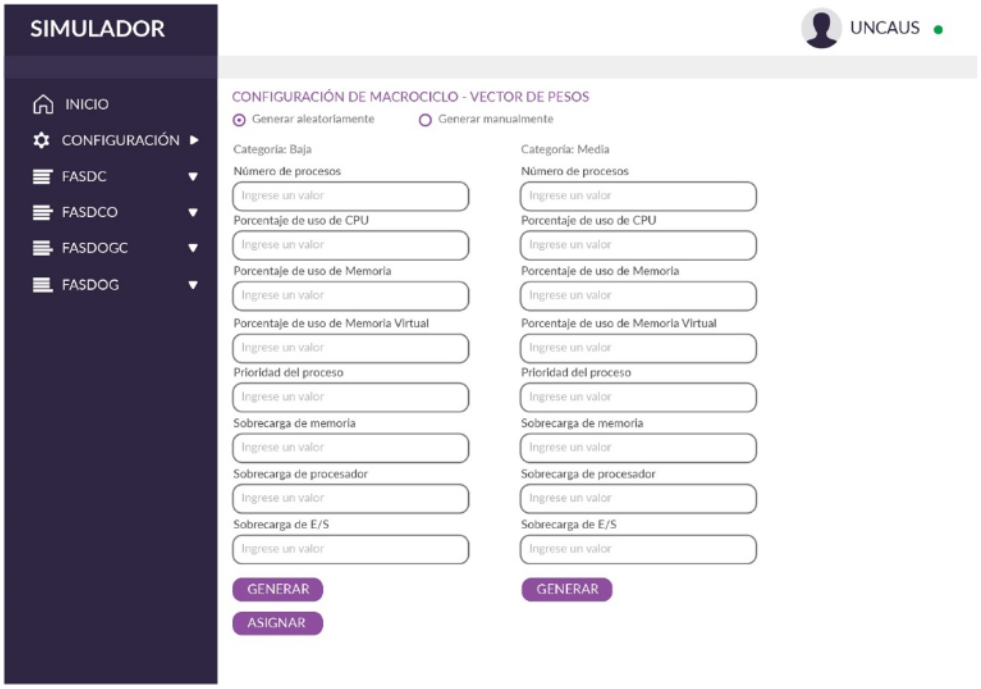


Figura 11. Configuración del vector de pesos.

Figura 12. Configuración de la carga de recursos y procesos.

Para los distintos escenarios, previamente habrá que configurar los datos de entrada según lo comentado en el apartado anterior y se deberán realizan diferentes cálculos:

1. El primer caso consiste en completar los parámetros de configuración, para generar los datos de entrada, que serán almacenados en la base de datos.
2. A continuación, se deberá seleccionar el escenario deseado, que permitirá obtener la Función de Asignación para Sistemas Distribuidos para cada caso, la cual aplicará los cálculos establecidos en el operador de agregación correspondiente.
3. Dependiendo del tipo de escenario se usará un determinado algoritmo, en aquellos que no consideren requisitos de consenso, se trabajará por rondas de resolución, para asegurar la exclusión mutua en el acceso de los recursos, los resultados se darán en cada iteración, guardando estos en la base de datos, que se irá cargando a medida que se ejecutan las diferentes rondas. En el caso del algoritmo utilizado para cuando se consideran requisitos de consenso, este tomará la información correspondiente a la concatenación de todas las rondas de asignaciones calculadas previamente, los resultados se almacenarán en la base de datos, luego de aplicar los cálculos globales.
4. En cada escenario, se generarán distintos vectores, que posteriormente habrá que ordenarlos, según cada premisa (beneficiar procesos o grupos de procesos).
5. Mientras aún queden solicitudes por atender, se podrá seguir realizando iteraciones y cálculos, para ello, en cada iteración se irán marcando aquellas solicitudes que han sido atendidas. Siempre se obtendrá una lista final con todas las asignaciones de recursos a procesos ordenadas, según la premisa establecida en cada escenario.

4. Análisis de resultados

La plataforma ha sido desarrollada en el lenguaje PHP y el *framework* CodeIgniter. El resultado ha sido satisfactorio. Se trata de un programa con cientos de líneas de código, lo cual da idea de la complejidad de su desarrollo. El mismo fue probado exhaustivamente, ensayando cada tipo de escenario por separado. Los resultados obtenidos se contrastaron con los generados por planillas de cálculo para los mismos conjuntos de datos, las que no se muestran por razones de espacio. En las Figs. 9, 10, 11 y 12 se muestran las principales ventanas del simulador, las cuales permiten una cómoda interacción con el usuario. Existen 4 escenarios posibles (Figs. 1 y 9), de los cuales se indican a continuación sus principales funciones de asignación:

4.1. E1 - FASDC

La tabla final resulta de la denominada Función de Asignación para Sistemas Distribuidos Concatenada (Fig. 13).

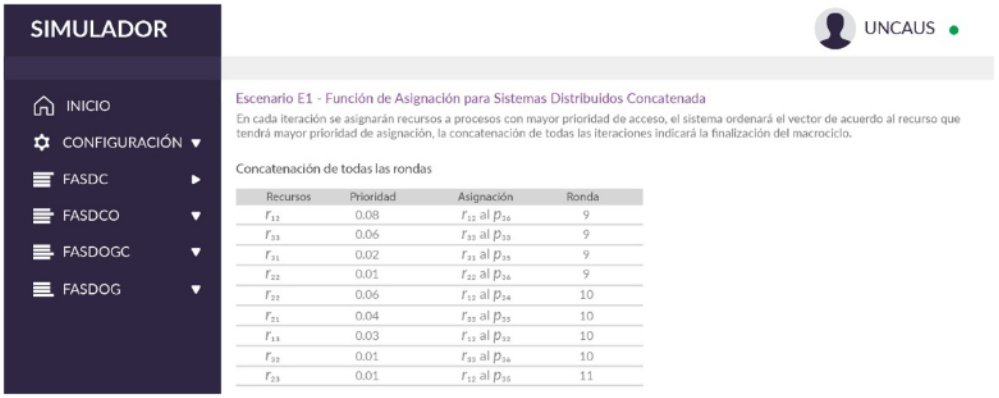


Figura 13. Resultado de las últimas rondas de la tabla concatenada.

4.2. E2 - FASDCO

De acuerdo con el orden de procesos obtenido (*PGFPO*), se construye una tabla denominada *Función de Asignación para Sistemas Distribuidos Concatenada Ordenada – FASDCO,* con las asignaciones de recursos a procesos, ordenada por proceso de mayor a menor prioridad (Fig. 14). El modelo propuesto incluye, como caso particular, un método que consiste en considerar la prioridad global de los procesos, en lugar de un grupo de variables de estado de cada nodo.

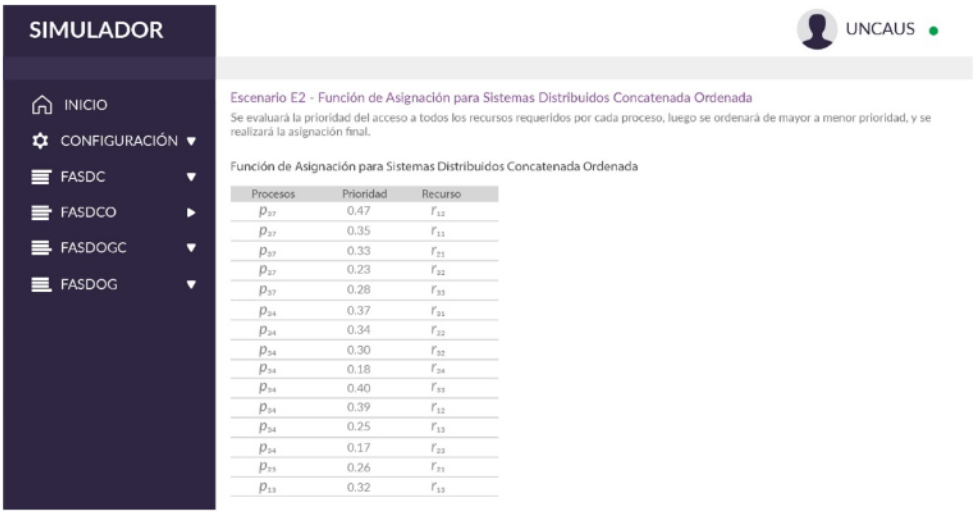
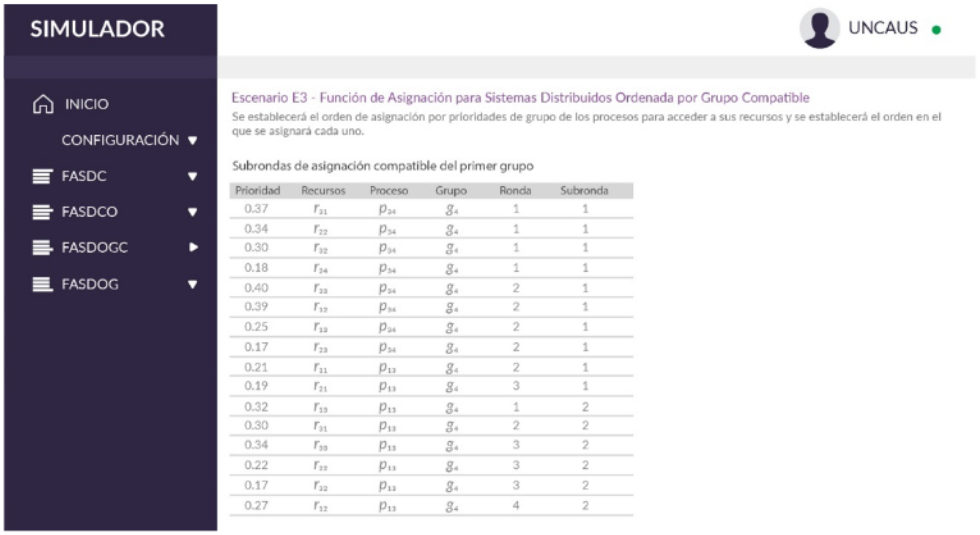


Figura 14. Función de Asignación para Sistemas Distribuidos Concatenada Ordenada.

4.3. E3 - FASDOGC

En la Fig. 15, se muestra una tabla donde se han ordenados los grupos de mayor a menor prioridad, y por cada grupo subrondas de asignación, para evitar la incompatibilidad en la asignación de recursos y asegurar la exclusión mutua. Esa tabla se denomina *Función de Asignación para Sistemas Distribuidos Ordenada por Grupo Compatible (FASDOGC).*



Prioridad	Recursos	Proceso	Grupo	Ronda	Subronda
0.37	r_{31}	p_{24}	g_4	1	1
0.34	r_{22}	p_{34}	g_4	1	1
0.30	r_{32}	p_{34}	g_4	1	1
0.18	r_{24}	p_{34}	g_4	1	1
0.40	r_{33}	p_{34}	g_4	2	1
0.39	r_{12}	p_{34}	g_4	2	1
0.25	r_{33}	p_{34}	g_4	2	1
0.17	r_{23}	p_{34}	g_4	2	1
0.21	r_{31}	p_{13}	g_4	2	1
0.19	r_{21}	p_{13}	g_4	3	1
0.32	r_{33}	p_{13}	g_4	1	2
0.30	r_{31}	p_{13}	g_4	2	2
0.34	r_{33}	p_{13}	g_4	3	2
0.22	r_{22}	p_{13}	g_4	3	2
0.17	r_{32}	p_{13}	g_4	3	2
0.27	r_{32}	p_{13}	g_4	4	2

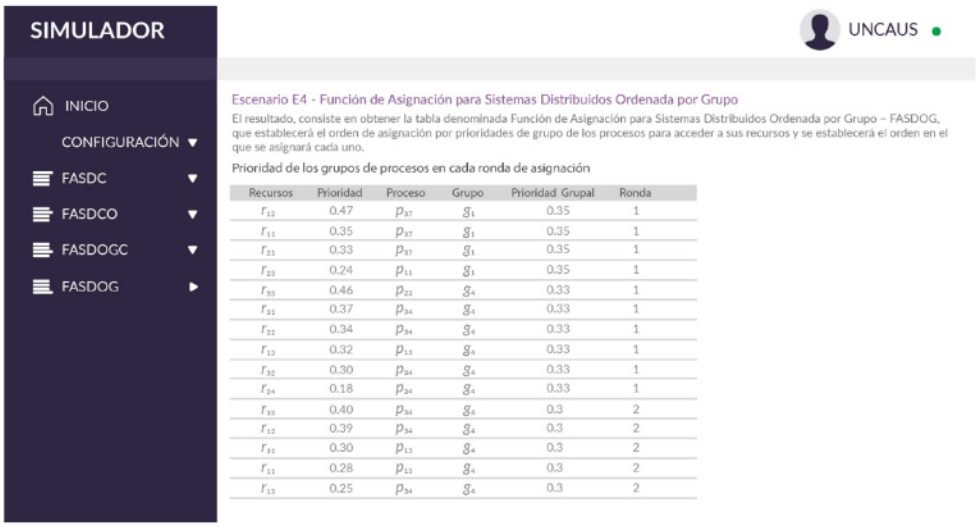
Figura 15. Subrondas de asignación compatible del primer grupo.

4.4. E4 - FASDOG

Se muestra la tabla denominada *Función de Asignación para Sistemas Distribuidos Ordenada por Grupo – FASDOG*, donde están las asignaciones de recursos a procesos ordenadas por grupos de procesos, de mayor a menor prioridad en cada ronda (Fig. 16).

El modelo propuesto incluye, como caso particular, un método que consiste en considerar la prioridad de los grupos de procesos en cada ronda de asignación, en lugar de un grupo de variables de estado de cada nodo.

Al evaluar los resultados obtenidos con el simulador, se pudo verificar que la solución producida contempla un adecuado balanceo de carga de trabajo, según el sustento teórico utilizado para el desarrollo del simulador. Asimismo, permitió demostrar que la solución teórica propuesta [14] es más adecuada que algoritmos tradicionales que asignan recursos a procesos solamente en función de la prioridad de los procesos.



Recursos	Prioridad	Proceso	Grupo	Prioridad Grupal	Ronda
r_{12}	0.47	p_{37}	g_1	0.35	1
r_{11}	0.35	p_{37}	g_1	0.35	1
r_{21}	0.33	p_{37}	g_1	0.35	1
r_{23}	0.24	p_{11}	g_1	0.35	1
r_{33}	0.46	p_{22}	g_4	0.33	1
r_{31}	0.37	p_{24}	g_4	0.33	1
r_{22}	0.34	p_{34}	g_4	0.33	1
r_{13}	0.32	p_{13}	g_4	0.33	1
r_{32}	0.30	p_{34}	g_4	0.33	1
r_{24}	0.18	p_{24}	g_4	0.33	1
r_{33}	0.40	p_{34}	g_4	0.3	2
r_{12}	0.39	p_{34}	g_4	0.3	2
r_{31}	0.30	p_{13}	g_4	0.3	2
r_{13}	0.28	p_{13}	g_4	0.3	2
r_{13}	0.25	p_{34}	g_4	0.3	2

Figura 16. Prioridad de los grupos de procesos en cada ronda de asignación.

Como valor agregado al simulador, se le han añadido gráficos interactivos utilizando las librerías de amCharts 4 [29]. El mapa de calor representa la prioridad que tiene cada proceso de acceder a cada recurso, donde el color más intenso se relaciona con el valor más alto de prioridad (Fig. 17) [29].

Mediante el diagrama de Sankey se puede mostrar, por ejemplo, el flujo y la relación entre los recursos y procesos en las distintas rondas, por ejemplo, para el escenario 1 (Fig. 18) [29].

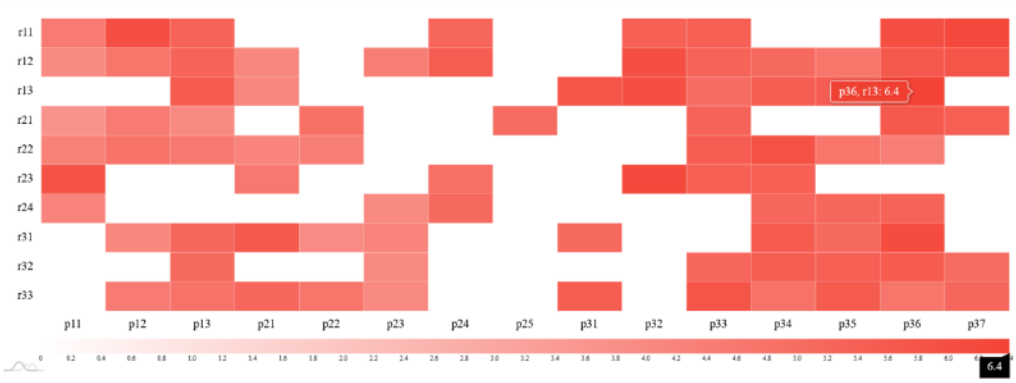


Figura 16. Mapa de calor de asignaciones de recursos a procesos.

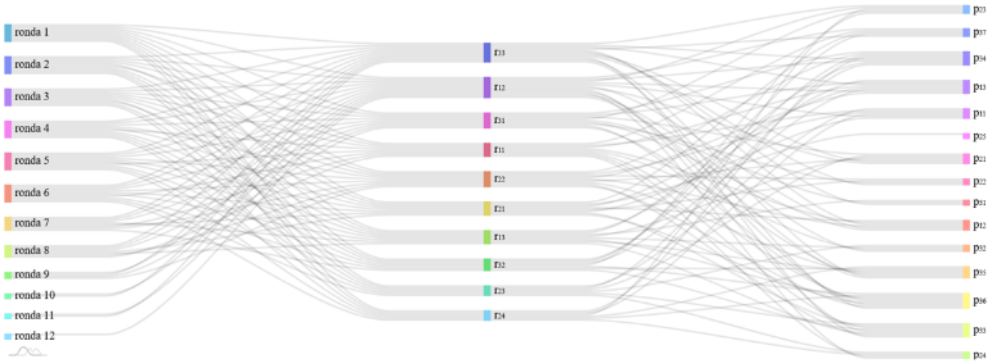


Figura 18. Gráfico de asignaciones de recursos a procesos por rondas.

Si bien un simulador no puede sustituir el trabajo directo con equipos, puede proveer en cambio, facilidad de acceso, independencia de topologías, equipos y protocolos, trabajar con diferentes tipos de escenarios personalizados, algunos de estos escenarios pueden ser configurados erróneamente o incompletos para corregirlos, y un punto clave es la visualización gráfica y los costes. Aunque no son reales, imitan de cerca la realidad. Otro punto a favor de los simuladores son las prácticas domiciliarias o de oficina, en cualquier horario y al ritmo propio del usuario.

Con la ayuda del correspondiente soporte informático, el modelo de simulación permite considerar complejas tareas interrelacionadas y proyectarlas mediante la realización de muchas combinaciones alternativas en cuestión de segundos. Además, la interacción de los recursos con los procesos se traduce en un gran número de pruebas simulando variedad de cargas de trabajos diferentes y de posibles resultados imposibles de abarcar y valorar sin la ayuda de un modelo de simulación computarizado.

Comparativas con los métodos tradicionales y resultados preliminares del uso de este simulador han sido utilizados en varias publicaciones [19, 20, 21, 22, 23], pero en ninguna de ellas se describió en detalle cómo funciona el simulador desarrollado, lo cual se ha hecho en este artículo.

5. Conclusiones

Se ha desarrollado un simulador en el que se han considerado distintos escenarios posibles para permitir al sistema predecir, comparar y optimizar el comportamiento de sus procesos simulados en un tiempo muy breve sin el coste ni el riesgo de llevarlos a cabo, haciendo posible la representación de los procesos, recursos y nodos en un modelo dinámico.

La utilización del simulador con la misma carga de trabajo para los diferentes escenarios planteados permitió corroborar que los resultados en cuanto al orden de asignación de recursos a procesos son diferentes, conforme a los distintos escenarios planteados.

Se han simulado diferentes cargas de trabajo en el sistema distribuido, bajo los diferentes escenarios propuestos, habiéndose obtenido resultados acordes con los modelos teóricos utilizados, que son el sustento del

simulador. Al evaluar los resultados obtenidos con el simulador, se pudo verificar que la solución producida contempla un adecuado balanceo de carga de trabajo, según lo previsto por el modelo teórico utilizado.

Asimismo, se pudo demostrar que la solución teórica propuesta es más adecuada que algoritmos tradicionales que asignan recursos a procesos solamente en función de la prioridad inicial de los procesos.

A futuro se tiene previsto trabajar en las siguientes líneas de investigación y agregado de funcionalidades al simulador:

1. Imputación de datos de control faltantes, por ejemplo, como consecuencia de problemas en las comunicaciones entre los nodos.
2. Fuzzyficación de variables para dar soporte a situaciones donde no es posible o conveniente expresar valores exactos. Se considerará utilizar lógica difusa con conjuntos de etiquetas lingüísticas de al menos 5 etiquetas.
3. Definición e incorporación de escenarios complejos, adaptándolos a las investigaciones más recientes sobre el tema que aborda el grupo de investigación.
4. Planeamiento de capacidad, es decir, la posibilidad de que para determinadas cargas de trabajo el simulador indique las características deseables de los diferentes nodos del sistema distribuido.
5. Incorporación de aspectos vinculados a la seguridad en la ejecución de los procesos, en el acceso a los recursos y en la comunicación entre los nodos.
6. Incorporación del soporte para sistemas de tiempo real, es decir, sistemas con requerimientos con tiempo perentorio de finalización de tareas.
7. Migración de procesos, por ejemplo, como consecuencia de la necesidad de requerimientos específicos, basados en las características del nodo donde se debería ejecutar el proceso.

Se considera que la incorporación de estas funcionalidades sumadas a las ya existentes permitirá disponer de un simulador complejo, flexible y de fácil utilización para la investigación y la docencia.

6. Agradecimientos

Este trabajo se enmarca especialmente en el Proyecto de Investigación “Desarrollo de un simulador para la evaluación de algoritmos clásicos y nuevos para la gestión de recursos compartidos en sistemas distribuidos contemplando exclusión mutua”, PI N° 126, aprobado por Res. N° 011/20 CS de la Universidad Nacional del Chaco Austral (Argentina). Un agradecimiento especial para los integrantes del PI que colaboraron activamente en la revisión bibliográfica, Manuel Alejandro Ricardone, Cynthia Evelín Bruic y Hugo Haurech.

7. Referencias

[1] Tanenbaum, A. S. (1996). *Sistemas Operativos Distribuidos*. México: Prentice-Hall Hispanoamericana S.A.
[2] Tanenbaum, A. S. (2009). *Sistemas Operativos Modernos* (3ra. Ed.). México: Pearson Educación S. A.
[3] Agrawal, D., El Abbadi, A. (1991). An Efficient and Fault-Tolerant Solution of Distributed Mutual Exclusion. *ACM Transactions on Computer Systems,* 9 (1), 1-20. doi: https://doi.org/10.1145/103727.103728
[4] Ricart, G., Agrawala, A. K. (1981). An Optimal Algorithm for Mutual Exclusion in computer networks. *Communications of the ACM,* 24 (1), 9-17. doi: https://doi.org/10.1145/358527.358537
[5] Cao, G., Singhal, M. (2001). A Delay-Optimal Quorum-Based Mutual Exclusion Algorithm for Distributed Systems. *IEEE Transactions on Parallel And Distributed Systems*, 12 (12), 1256-1268. doi: https://doi.org/10.1109/71.970560
[6] Lodha, S., Kshemkalyani, A. (2000). A Fair Distributed Mutual Exclusion Algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 11 (6), 537-549. doi: https://doi.org/10.1109/71.862205
[7] La Red Martínez, D. L. (2004). *Sistemas Operativos*. Argentina: EUDENE.
[8] Stallings, W. (2005). *Sistemas Operativos* (5ta. Ed.). España: Pearson Educación S.A.
[9] Joshi, R., Holzmann, G. J. (2007). A Mini-Challenge: Build a Verifiable Filesystem. *Formal Aspects of Computing*, 19 (2), 269-272. doi: https://doi.org/10.1007/s00165-006-0022-3

[10]Alagarsamy, K. (2003). Some Myths About Famous Mutual Exclusion Algorithms. *ACM SIGACT News,* 34 (3), 94–103. doi: https://doi.org/10.1145/945526.945527

[11]Ayala Franco, E., Menéndez Domínguez, V. H., Curi Quintal, L. F., Castellanos Bolaños, M. E. (2019). Usabilidad de un simulador para la enseñanza de la Programación de Sistemas. *Abstraction & Application*, 26, 56-72.

[12]Pamplona Roche, S., Medinilla Martínez, N. (2012). Aprendizaje de sistemas operativos mediante simulaciones interactivas. Trabajo presentado en *Jornadas de Enseñanza de la Informática*, Ciudad Real, España.

[13]Ayala, E., Madera, F. A., Basto, L. (2017). Operating System Simulator to Translate Assembler Code to Machine Code. *International Journal of Computer Trends and Technology (IJCTT),* 51 (1), 58-63. doi: https://doi.org/10.14445/22312803/IJCTT-V51P109

[14]La Red Martínez, D. L. (2017). Aggregation Operator for Assignment of Resources in Distributed Systems. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 8 (10), 406-419. doi: https://dx.doi.org/10.14569/IJACSA.2017.081053

[15]La Red Martínez, D. L., Acosta, J. C., Gerzel, S. M., Rambo, A. R. (2017). New decision making models of processes synchronization in distributed systems. Trabajo presentado en *International Conference on Communication and Electronic Information Engineering* (pp. 65-71). Guangzhou, China. doi: https://doi.org/10.2991/ceie-16.2017.9

[16]Agostini, F., La Red Martínez, D. L., Acosta, J. C. (2018). Modeling of the Consensus in the Allocation of Resources in Distributed Systems. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 9 (12), 26-36. Recuperado de: https://repositorio.unne.edu.ar/handle/123456789/30859

[17]La Red Martínez, D. L., Acosta, J. C., Agostini, F. (2018). Assignment of Resources in Distributed Systems. *Journal of Systemics, Cybernetics and Informatics*, 16 (5), 81-87. Recuperado de: http://www.iiisci.org/journal/sci/FullText.asp?var=&id=ZA872ME18

[18]Ríos, D. A., La Red Martínez, D. L. (2019). New Decision Model for Network Traffic Management. *International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC)*, 6 (2) 99-122.

[19]Agostini, F., La Red Martínez, D. L. (2019). Allocation of shared resources. Trabajo presentado en *14th Iberian Conference on Information Systems and Technologies (CISTI*) (pp. 1-6). Coimbra, Portugal. doi: https://doi.org/10.23919/CISTI.2019.8760901

[20]Agostini, F., La Red Martínez, D. L., Acosta, J. C. (2019). Assignment of Resources in Distributed Systems With Strict Consensus Requirements. Trabajo presentado en *10th International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC)* (pp. 90-95). Orlando, Florida, USA.

[21]Agostini, F., La Red Martínez, D. L., Fornerón Martínez, J. T. (2019). Nuevo Operador de Agregación para Grupos de Procesos. Trabajo presentado en *Conferencias IADIS (International Association for Development of the Information Society) Ibero-Americanas WWW/Internet* (pp. 223-230). Lisboa, Portugal.

[22]Fornerón Martínez, J. T., Agostini, F., La Red Martínez, D. L. (2020). Resource and Process Management with a Decision Model based on Fuzzy Logic. Trabajo presentado en *24th World Multi-Conference on Systemics, Cybernetics and Informatics* (pp 74-79). Orlando, Florida, USA.

[23]Fornerón Martínez, J. T., Agostini, F., La Red Martínez, D. L. (2020). Modelo de Decisión para Gestión de Procesos y Recursos en Sistemas Distribuidos con Balanceo de Carga de Trabajo. Trabajo presentado en *Conferencias IADIS (International Association for Development of the Information Society) Ibero-Americanas WWW/Internet* (pp. 197-201).

[24]Suresh, L., Bodik, P., Menache, I., Canini, M., Ciucu, F. (2017). Distributed resource management across process boundaries. Trabajo presentado en *Symposium on Cloud Computing (SoCC)* (pp. 611-623). Santa Clara, California, USA. doi: https://doi.org/10.1145/3127479.3132020

[25]Qurani, M. O., Singh, R. (2018). Load Balancing for Virtual Resources Management in Data Center. Trabajo presentado en *8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 677-682). Noida, India. doi: https://doi.org/10.1109/CONFLUENCE.2018.8442843

[26]Miller, J. E., Kasture, H., Kurian, G., Gruenwald, Ch., Beckmann, N., Celio, C., Eastep, J., Agarwal, A. (2010). Graphite: A distributed parallel simulator for multicores. Trabajo presentado en *The Sixteenth International Symposium on High-Performance Computer Architecture* (pp 1-12). Bangalore, India. doi: https://doi.org/10.1109/HPCA.2010.5416635

[27] Tampouratzis, N., Papaefstathiou, I., Nikitakis, A., Brokalakis, A., Andrianakis, S., Dollas, A., Marcon., M., Plebani, E. (2020). A Novel, Highly Integrated Simulator for Parallel and Distributed Systems. *ACM Transactions on Architecture and Code Optimization*, 17 (1), 1-28. doi: https://doi.org/10.1145/3378934

[28] Valdivieso Pinzón, C. A., Cajas Guijarro, C. D., Mejía Navarrete, R. D., Bernal Carrillo, I. M. (2017). Simulación en NS3 del problema denominado cuello de botella compartido que se presenta en el protocolo MP-TCP. *Revista de Investigación en Tecnologías de la Información (RITI),* 5 (9), 70-79. Recuperado de: https://www.riti.es/ojs2018/inicio/index.php/riti/article/view/45

[29] Amcharts. (2021). *JavaScript Charts & Maps*. Recuperado de: https://www.amcharts.com