Simulación paralela basada en agentes de sociedades precolombinas: Relaciones entre clanes

Alfonso Imbernon Garcia

Resum— La utilización de paralelismo para resolver problemas de simulación es algo habitual. En este proyecto se ha desarrollado un modelo en paralelo que permita simular, a partir de agentes, la sociedad precolombina en la patagonia. El modelo no partía de cero y en este proyecto se le ha añadido funcionalidades que simulen la vida del ser humano y la relación entre los clanes. Para finalizar se ha comprobado las mejoras en el rendimiento del modelo paralelo.

Paraules clau— ABM, cluster, computación paralela, agente, FLAME, CRAG.

Abstract— The use of parallelism to solve problems of simulation is something common. In this project i have developed a model based in parallel that let simulate, using agents, the Pre-Columbian society of Patagonia. The model was starting and in this project have been added functionalities to simulate human life and the relationship between clans. Finally, it has verified the performance enhancements provided by the parallel model.

Index Terms— ABM, cluster, paralllel computing ,agent,FLAME, CRAG

1 Introducción

Agent-based model (ABM)[1] es un modelo computacional que permite estudiar el comportamiento e interacción de agentes autónomos dentro de un entorno previamente definido y poder evaluar su incidencia en el sistema. El ABM se ha aplicado a proyectos de investigación con la finalidad de estudiar, a partir de simular el entorno social, distintas sociedades históricas. En el marco del proyecto Simulpast [2], encargado de estudiar diferentes etnicidades históricas, quiere estudiar la sociedad precolombina en la Patagonia (Argentina). El estudio consiste en crear un modelo que permita estudiar el comportamiento de agrupaciones culturalmente homogéneas y ver su evolución a lo largo de un periodo de tiempo, concretamente desde la prehistoria hasta la llegada de los europeos en el siglo XVI.

El equipo que se encargó del proyecto Patagonia [3], compuesto de personas de varias disciplinas, consiguió simular el modelo, de manera secuencial, a partir de clanes como unidad mínima. Esto les supuso un problema ya que para poder hacer un estudio con la precisión y calidad deseadas, necesitaban una granularidad más pequeña que el clan. Al tomar como unidad mínima el individuo, el tiempo de ejecución se convirtió en el problema.

El proyecto que nos ocupa, pretende solucionar el problema explicado anteriormente mediante la creación de un modelo paralelo, con las funcionalidades del modelo original y así poder ejecutar la simulación del modelo en un cluster, en concreto en el CRAG [4] y poder verificar la ganancia de la ejecución en paralelo contra la versión serie. En este proyecto el modelo no comienza de cero, sino cuenta con una base ya realizada por Hemalatha Vulchi [5] cuya base será explicada en el punto 3.2.

El presente documento está organizado en las siguientes secciones: En la sección 2, una visión de los objetivos del trabajo, en la sección 3 una explicación de la herramienta utilizada y del estado inicial del modelo, en la sección 4 una explicación de la metodología aplicada, en la sección 5, una explicación de las funcionalidades añadidas al modelo, en la sección 6 una explicación sobre la plataforma que se ejecutó la simulación, en la sección 7 los resultados obtenidos y último lugar se presenta las conclusiones obtenidas y las líneas futuras.

2 Objetivos

El objetivo principal que tiene el trabajo presente es el de diseñar y desarrollar aspectos específicos del modelo final con el fin de aumentar las funcionalidades del simulador, siguiendo las reglas marcadas por el equipo encargados del proyecto original. En este proyecto se decidió

- *E-mail de contacte: alfonso.imbernon@gmail.com*
- *Menció realitzada: Enginyeria de Computadors*
- *Treball Eduardo Cesar Galobardes dept. CAOS*
- *Curs 2014/15*

extender las funcionalidades de los agentes individuo y clan por la importancia que representan en el modelo final.

Además del objetivo principal descrito anteriormente, al ser un proyecto que cooperó Rodríguez [10] en su desarrollo cuya parte esta explicada en el punto 3.4, aparece otro objetivo, el cual hace referencia a coordinar las partes en las que pueda haber intersección. Rodríguez [10] se encargó de resolver la problemática del movimiento de los individuos, según la cantidad de comida que se obtienen del patch. Una de las partes que interaccionan es sobre el movimiento de las personas y el conocimiento de familia.

Una vez acabados los puntos anteriormente descritos, se hizo un análisis de los resultados conseguidos al ejecutar el modelo en el CRAG y comparando la ejecución paralela con la secuencial.

3 Estado del arte

3.1 FLAME

Para realizar el proyecto se ha usado un framework llamado FLAME [6]. "FLAME" es un sistema de modelado basado en agentes genéricos que se puede utilizar para aplicaciones de desarrollo en muchas áreas. En el caso del presente trabajo, se utiliza por sus características de poder simular el modelo de forma paralela dentro de un cluster. Existen diferentes herramientas con el mismo propósito de simular un ABM, por ejemplo Pandora (del BSC), RepastHPC y D-Mason. Comparando resultados, tanto de las características que ofrecen, como las prestaciones obtenidas a la hora de ejecutar un modelo base, se adapta mejor FLAME que cualquier de las otras alternativas. Existe un estudio comparativo de estas cuatro opciones [7].

3.1.2 Funcionamiento FLAME

FLAME especifica un modelo basado en agentes. El comportamiento de este modelo es el de una máquina de estados, que se compone de un único estado inicial, un estado final y un número de estados con funciones de transición entre esos estados. Entre los estados de transición puede haber comunicación a través de mensajes. Esto sucede por cada agente que conforma el modelo.

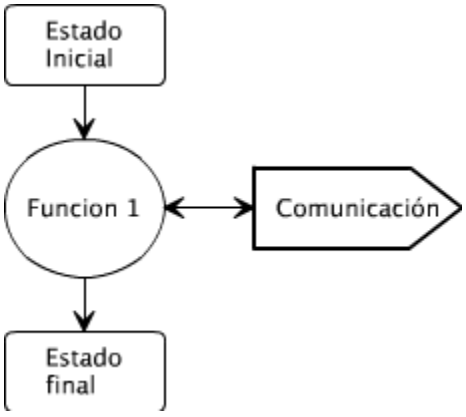

Figura 1: Ejemplo de máquina de estado de un agente

3.1.3Comunicación en FLAME

Por lo que respeta a la memoria, cada agente tiene su propia memoria en la que guardada sus variables, en el caso de un individuo una variable es la edad. Este espacio de memoria es accesible desde las funciones de transición de ese mismo agente, pero no por funciones de transición de otros agentes, por lo tanto el intercambio de información entre agentes es mediante comunicación. Para poder comunicar dos agentes se utiliza el envío de mensajes de forma síncrona. Como FLAME está pensado para ejecutar la simulación dentro de un cluster y los agentes a comunicar puede estar en diferentes nodos, el agente no envía directamente el mensaje al agente destino, sino difunde el mensaje (comunicación por difusión). Por lo tanto el agente receptor deberá hacer un filtraje de mensajes para seleccionar que mensajes son destinados a él.

3.1.4 Ejecución en FLAME

Para ejecutar un modelo en FLAME es necesario indicarle tres componentes: la especificación de un tipo de agente (en xml), sus funciones de transición (en c) y la inicialización de la simulación (en xml). En la especificación incluye la máquina de estado, la declaración de las variables y la especificación de los mensajes del agente, un ejemplo de tipo de agente es el agente clan o el agente individuo. La figura 2 muestra un ejemplo de definición de un estado de un agente:

```
<function>
    <name>example</name>
    <description>eg transition function</description>
    <currentState>start</currentState>
    <nextState>end</nextState>
    <inputs>
        <input>
            <messageName>eg input message</messageName>
            <filter>
                <lhs><value>a.cID</value></lhs>
                <op>EQ</op>
                <rhs><value>m.clanID</value></rhs>
            </filter>
        </input>
    </inputs>
    <outputs>
        <output>
            <messageName>eg output message</messageName>
        </output>
    </outputs>
</function>
```

Figura 2: Ejemplo de función de transito

En el anterior ejemplo se puede ver la declaración de una función de transición. En este caso es la única función del agente, ya que ella misma es la primera y la última función de la máquina de estados. Tiene de comunicación dos mensajes, uno de entrada filtrado a través de la comparación de igualdad entre una variable miembro del agente y una variable del mensaje y un mensaje de salida.

Las funciones de transito indica el comportamiento del agente al llegar a un estado determinado, mientras que la inicialización de la simulación inicializa las variables de los agentes a un valor conocido.

La figura 3 muestra un ejemplo de inicialización.

```
<agents>
        <xagent>
                <name>indv</name>
                <indvID>0</indvID>
                <cID>0</cID>
        </xagent>
</agents>
```

Figura 3: Ejemplo de declaración de agentes

En este ejemplo se inicializa dos variables, indvID y cID a0, del agente indv.

Una vez proporcionado a FLAME las entradas explicadas, genera código fuente cuyo comportamiento sea el indicado en la máquina de estado y las funciones de transición.

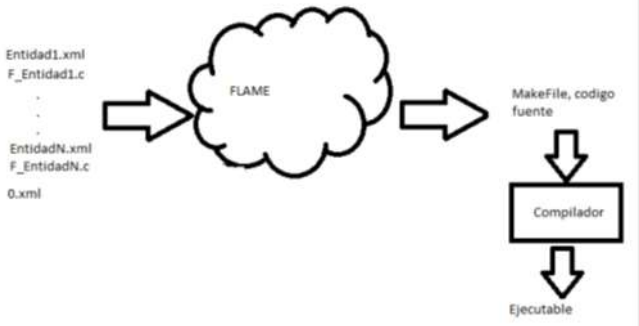

Figura 4: Pasos de compilación en FLAME

3.2 Estado inicial del modelo

El modelo como se ha comentado anteriormente no partía de cero sino contó con una base realizada por el trabajo de final de máster, High Performance Computing de Hemalatha Vulchi [5]. Esta base incluye la creación básica de los tipos de agentes individuo, patch y clan. La descripción de estas entidades es la siguiente:

Individuos. Son las personas que habitan en el territorio, tomando decisiones sobre sus necesidades y las percepciones que obtienen. Entre los individuos puede haber cooperación según su ubicación temporal, ya sea en el intercambio de conocimientos, en la ayuda para cazar y en el emparejamiento.

Patch (zonas del territorio). Hace referencia a las zonas que están divididos el territorio (Patagonia), son él terreno por donde se mueven los individuos y guanacos obteniendo suministros para sobrevivir. Estas zonas pueden ser de cuatro tipos, tal como se muestra en la figura 5, montaña (marrón), pre-montaña (verde oscuro), costa (azul), llano (verde). Los patch tiene dos estaciones, seca y húmeda, que afectan a la cantidad de suministro que disponen y a su regeneración.

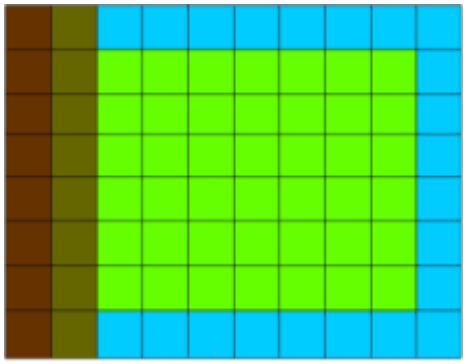

Figura 5: División del terreno en patches

Clanes. Agrupaciones de individuos con relaciones familiares, donde uno de ellos hace la función de líder. El clan se encarga de proveer de comida a los individuos que lo conforman. La ubicación del clan es variable dentro de los diferentes patch.

3.3 Funcionalidad implementada:

El estado inicial del modelo por cada agente es el siguiente:

Individuos: Comunican sus necesidades alimentarias al clan al que pertenecen, envejecen con el paso del tiempo y con una cierta probabilidad mueren cuya probabilidad varía según la edad y la necesidades alimentarias.

Clanes: Recibe las demandas de alimentos de sus miembros, obtiene alimentos del patch en el que se encuentra y los redistribuye entre los miembros según sus peticiones.

Patch: Suministra a los clanes de alimentos y se regenera cada cierto tiempo cuya regeneración varía según la época, húmeda o seca.

3.4 Cooperación

Rodríguez [10] se encargó de crear el nuevo agente guanaco para el modelo. Este agente puede representar a una manada de guanacos o a una familia de guanacos, de forma simplificada, la manada es un grupo de machos mientras que la familia de guanacos representa a un macho con varias hembras. La utilidad de este agente es la de proporcionar calorías a los individuos, por lo tanto, en un trabajo futuro, los individuos de varios clanes cooperarán en busca de cazar manadas o familias de guanacos con el fin de obtener calorías para poder sobrevivir. El movimiento de los guanacos depende de dos factores, la estacionalidad y las cercanías de individuos. En el primer caso, en época seca, el movimiento es hacia zonas de costa mientras que en época húmeda puede recorrer cualquier parte del llano. En lo que concierne al individuo, los guanacos intentan moverse a patch donde la cantidad de humanos sea menor.

Otro punto realizado fue el movimiento de clanes. En este caso los clanes intentan moverse a patch donde la presencia de guanacos sea mayor, además teniendo en consideración los recursos naturales del patch. El movimiento de clanes también es afectado por la estacionalidad, cambiando la tendencia de movimiento hacia zonas costeras en épocas secas y recorriendo por el llano en

época húmedas, de la misma forma que los guanacos.

4 METODOLOGÍA

Para la realización el trabajo, después de analizar los diferentes modelos de ciclos de vida del software[8], se aplicó una metodología incremental donde a partir de un modelo base, se fue incrementando el modelo con más funcionalidades una vez fueron diseñadas, implementadas y testadas.

Esta metodología se eligió por las ventajas que aporta, ya que no hacía falta tener todo el modelo acabado para poder ver los resultados, esto permite que el usuario (en mi caso los sociólogos) pudieron dar su opinión y haber tenido una retroalimentación desde un principio, a partir de la reunión. Además, otra ventaja es la disminución de la posibilidad de falla (metodología con bajo riesgo), al poder separar el trabajo, por orden de prioridad, en diferentes tareas que se implementaron y verificaron antes de proseguir con un incremento de funcionalidad.

5 APORTACIÓN

Al modelo inicial se le añadió un conjunto de características con la finalidad de acercar su funcionalidad final a lo marcado por el estudio del equipo de sociólogos. Como se comentó anteriormente, estas características están relacionadas con simular el comportamiento de los humanos en la época precolombina, de manera que al agente individuo fue completado con memoria para poder recordar su árbol genealógico, la capacidad de emparejarse con otro individuo, de poder tener descendencia y un lenguaje que evoluciona al cabo de los años.

5.1 Ancestros

El ser humano tiene la capacidad de poder recordar y conocer quiénes forman parte de su familia, por ello en una simulación de una sociedad, fue un punto a incluir y en este caso es la base de los otros puntos realizados, como se podrá ver en los siguientes apartados. En el caso de la simulación, el individuo recuerda su árbol genealógico de profundidad cuatro, esto implica que el individuo recuerda hasta la cuarta generación incluyéndose así mismo. De esta manera un individuo puede determinar los siguientes miembros de su familia: padres, hermanos, abuelos, tíos y primos. La figura 6 muestra un ejemplo gráfico de árbol genealógico.

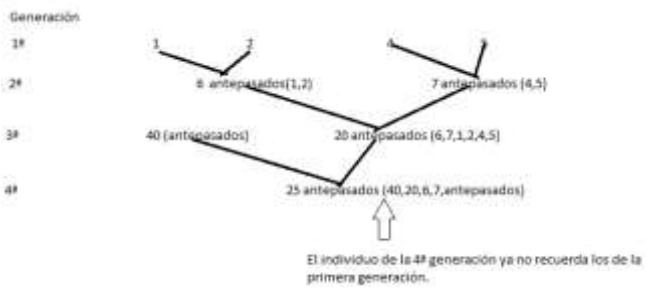

Figura 6: Arbor genealógico

5.2 Emparejamiento

Una vez determinado que individuos forman parte de mi conjunto familiar y quienes no, el ser humano tiende a emparejase con otro individuo, que no tenga relación familiar, con la finalidad de formar una familia. En este caso se añadió la capacidad de vincular dos individuos que no tengan lazo familiar. Para ello se siguió las reglas del emparejamiento, proporcionadas por el equipo de sociólogos, al modelo. Estas reglas hacen referencia al lazo familiar, distancia física entre individuos, la proximidad cultural, la edad y el sexo. La primera regla hace referencia a lo comentado anteriormente, un individuo solo se puede emparejar con otro individuo que no se haga referencia en su árbol genealógico. Otra regla es la distancia física entre los individuos, ya que dos individuos lejanos no pueden emparejarse. Un factor importante es el entendimiento entre clanes, esto se debe a que si dos clanes no se pueden comunicar entre ellos por diferencias lingüísticas, no puede existir emparejamiento. En el caso de la edad, una mujer no puede emparejarse al sobrepasar una edad (fijada en cuarenta años). Por último, el emparejamiento existe entre un hombre y una mujer.

En el contexto de la época simulada, en el clan se intenta emparejar a sus miembros femeninos que no tuvieran pareja con un miembro masculino de otro clan y viceversa. En el modelo el encargado de gestionar lo explicado es el agente clan. La figura 7 muestra la metodología de intercambio de individuos femeninos entre clanes.

```
// El individuo sin pareja envía a su clan la información
// necesaria para el emparejamiento mediante el mensaje ancestor
add_ancestor_message (Info_ancestros,ID_del_clan,
    ID_del_individuo,sexo_del_indv);

// El clan guarda en dos grupos distintos, según el sexo,
// el ID del individuo y la información de sus ancestros.
if (ancestor_message->sexo_del_indv == MASCULINO){
    Grupo_de_indviduos_masculinos_ID[male] =
        ancestor_message->ID_del_individuo;
    Grupo_de_indviduos_masculinos_ancestros[male] =
        ancestor_message->Info_ancestros;
    male++;
}
if (ancestor_message->sexo_del_indv == FEMENINO){
    Grupo_de_indviduos_femeninos_ID[female] =
        ancestor_message->ID_del_individuo;
    Grupo_de_indviduos_femeninos_ancestros[female] =
        ancestor_message->Info_ancestross;
    female++;
}

//El clan envía a otros clanes cercanos el grupo de individuos femeninos
add_Grupo_chicas_message (Grupo_de_indviduos_femeninos_
    ID,Grupo_de_indviduos_femeninos_ancestros);
```

Figura 7: Intercambio de hembras

Para realizar la comprobación de la distancia entre clanes, se usó el filtro box2D de FLAME. El funcionamiento del box2D es el siguiente: Se crea una caja cartesiana alrededor del agente de mida N donde N/2 es la distancia del agente hacia un lado de la caja. Por lo tanto el clan solamente recibe los mensajes de los clanes que estén dentro de la caja.

Una vez que el clan conoce los individuos femeninos no emparejados de los clanes vecinos, comprueba la compatibilidad lingüística entre el clan origen y el mismo, descartando aquellos clanes cuya compatibilidad no su-

pere el umbral fijado en 2%. En este punto el emparejamiento se realiza en tres pasos. Un primer paso, el clan realiza propuestas de emparejamiento con su grupo de pretendientes masculino, cumpliendo las reglas de emparejamiento explicadas en el punto anterior y envía las propuestas al clan origen de las hembras seleccionadas. Un segundo pasos, el clan al que pertenece las hembras, recibe las propuestas que hay sobre sus miembros femeninos y las acepta en orden FIFO, por lo tanto, en el caso de haber dos propuestas sobre la misma hembra, el clan acepta la primera propuesta que ha recibido. Por último, el clan que ha propuesto los emparejamientos, recibe las propuestas aceptadas y formaliza el emparejamiento notificándoselo a los individuos implicados.

Llegados a este punto, el individuo al recibir la notificación cambia su estado formal y en el caso del individuo femenino, cambia el clan al que pertenece por el de su pareja.

Existe una única excepción del método de emparejar explicado, en el caso de que un individuo femenino se le muera la pareja no sigue lo explicado anteriormente ya que esta hembra no se añade al grupo de hembras libres del clan. En este caso la hembra informa al clan que quiere una nueva pareja y el clan le busca un miembro masculino libre del clan que cumpla con las reglas de emparejamiento. En el caso de que el miembro que pierde la pareja sea masculino, sí que sigue la metodología explicada.

Una vez el emparejamiento haya sido un éxito, los dos clanes implicados guardan en un buffer circular el identificador del otro clan, esto implica que los clanes recuerdan con quien sus miembros se han emparejado y provocará en un futura continuación del modelo, los clanes que hayan emparejado miembros tendrán más afinidad que con los clanes que no se haya hecho. La afinidad entre clanes es un concepto que no se da en este proyecto, pero será muy importante en la cacería.

5.3 Reproducción

Una característica de un ser vivo para mantener la existencia es la reproducción, de la misma manera sucede en la simulación de la sociedad precolombina.

En la inclusión de la reproducción en el modelo, se creó una serie de estados por los que una mujer atraviesa desde el momento en el que contrae pareja hasta que deja de ser fértil. Se creó el concepto "embarazable", cuyo significado es el de una hembra en edad fértil, casada y no embarazada. Una vez explicado el concepto clave, la metodología es la siguiente:

```
//Embarazo de la hembra con un 10% de probabilidad
probability= rand () % 100;
if (probability < 10){
        pregnant ("si");
        embarazable("no");
}
//contabilizar los meses de embarazo
if (pregnant )
                month ++;
if (month == 9){
        id = petición_clan_nuevo_id_hijo();
//construcción del array ancestros tal como muestra la figura número 9.
```

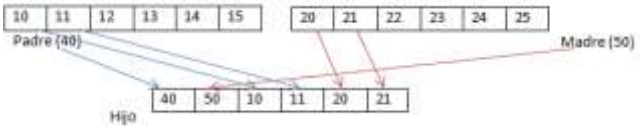

Figura 8: Creación del nuevo array ancestros

```
// Para finalizar se crea al nuevo individuo
        new_agent (id,ancestros);
}
```

En el momento de la creación del hijo aun la hembra no vuelve a ser "embarazable" ya que la simulación intenta ser lo más parecido a la realidad, existe un periodo de tiempo que tiene que pasar, una vez acabado el embarazo para que la mujer pueda volver a embarazarse, por lo tanto en la simulación, una vez tenido el hijo debe de esperar cinco meses más hasta poder quedarse embarazada.

5.4 Creación de nuevos clanes

Dada la inclusión de la funcionalidad anterior surge la problemática del crecimiento del clan. Con el fin de que el clan no aumente su población sin control, se ha incluido la división de clan. El agente clan está limitado a treinta individuos, una vez superada la cantidad máxima el clan entra en la fase de división. Esta fase consiste en cuatro pasos en los que intervienen el agente clan y sus agentes individuos. En un primer paso, el clan comunica a todos sus miembros que comienza la fase de división. Una vez el individuo recibe la noticia de división del clan, envía al clan los datos necesarios que el agente clan utilizará para decidir cómo clasifica a sus miembros. Llegados a este punto, el clan comienza la búsqueda con restricción (que el nuevo clan no sobrepase un cierto número de personas), el clan selecciona al primer individuo de la lista y busca su pareja (en caso de que la tenga), a continuación dependiendo del sexo, si es hombre, busca los hijos dependientes (hijos que no tienen pareja) de su pareja. Por el contrario, si es una mujer, directamente busca a sus hijos dependientes. Una vez decidido que individuos formarán parte del nuevo clan, se crea el nuevo agente clan. En la realización de este paso, el clan utilizando la hora del sistema, determina el nuevo identificador único que tendrá el nuevo clan. Por último paso, se comunica a los miembros afectados, cuál es su nuevo clan.

5.5 Lenguaje

La comunicación es otra característica más de ser humano que se ha integrado en el modelo. Esta nueva funcionalidad tendrá más implicaciones de las que se han realizado en el proyecto.

El lenguaje se representa mediante un gen que es capaz de discriminar diferentes variaciones de un mismo lenguaje original. Por lo tanto al clan se le ha dotado de un genoma lingüístico. Este genoma es representado como un array binario de longitud máxima de 100 posiciones cuyo contenido variará según los siguientes casos: Al unirse una hembra al clan, transmite según una probabilidad, variaciones del genoma de su clan de origen. Para

ello, por cada posición del array del clan que sean 0, si se supera un cierto umbral con la variable aleatoria, se aplica una OR en la posición del genoma del clan origen de la hembra.

Anualmente, con una cierta probabilidad el clan varía su genoma lingüístico. En este caso se utilizan dos constantes: learn (30%) y forget (10%), siguiendo la regla de que forget es mucho menor a learn. Por lo tanto a partir de estas probabilidades, se determina si se aprende palabras nuevas (una posición del genoma que era 0 pasa a 1) o por el contrario se olvida una palabra conocida (Una posición del genoma que era 1 pasa a 0).

Esta nueva habilidad tendrá de utilidad, en una posible continuación del modelo, a la hora de determinar que clanes pueden cazar guanacos juntos.

6 EXPERIMENTACIÓN

Tal como se explicó en la introducción del artículo, el modelo será ejecutado en un cluster con tal de comparar los resultados de las versiones serie y paralela. Antes de empezar explicando los resultados es conveniente explicar la plataforma donde se ejecutara el modelo explicado.

El cluster disponible en el CRAG, está formado por tres módulos conectados por una red Ethernet 10 Gigabit/s.

6.1 Modulos

- **Batman y catwoman**: Incorpora cuatro procesadores AMD Opteron de 16 núcleos, por lo tanto el total de núcleos son 64. Por lo que hace referencia a memoria son 128GB de RAM compuesta por 16 memorias de 4 GB DDR3.
- **Robin**: Utiliza un procesador Intel xeon E5-2630 de 6 núcleos, con una memoria interna de 64GB DDR3 y un raid5 (con paridad distribuida entre los discos) con discos de capacidad de 1TB. También incorpora dos aceleradores, un xeon phi y una nvidia GTX titan .
- **Huberman** : Este módulo tiene dos Intel xeon E5-2650 de 8 núcleos como procesado con una memoria interna de 256GB de RAM dividida en 8 memorias de 32 GB DDR3. Como aceleradores tiene una Nvidia tesla M2090 y otra Nvidia k20c Kepler.
- **Penguin**: Este módulo tiene 4 Intel xeon E5-4620 cada uno con 8 núcleos. En temas de memoria tiene 128 GB de RAM divididas en 16 unidades de 8GB DDR3.

6.2 Sistema de colas

El cluster CRAG utiliza el sistema de colas SLRUM [9] de código abierto diseñado para cluster Linux. Su funcionamiento es el siguiente:

El primer paso que realiza es la comprobación si el usuario ha pedido los recursos de forma exclusiva o no para un periodo de tiempo.

En un segundo paso, proporciona un framework para el inicio, ejecución monitorización del trabajo que realizan los nodos asignados.

En tercer lugar se distribuye los trabajos encolados en la cola de pendientes de acuerdo a las políticas.

Como usuario del sistema de colas, la manera de lanzar un trabajo se consigue de la siguiente manera:

Primero, como es habitual en la ejecución de un trabajo dentro de un cluster, se tiene que indicar una serie de recursos que necesitaras tales como numero de nodos, numero de CPU de un nodo, numero de tareas que se ejecutaran en un nodo, el tiempo máximo que necesitará el trabajo y un conjunto de parámetros más para configurar. Para facilitar la modificación y la lectura, la declaración de los parámetros comentados anteriormente, se realiza a través de un script. Un ejemplo de script se muestra en la figura 9.

```
#!/bin/bash

$SBATCH –p p_hpca4se
$SBATCH –N 5
$SBATCH – n 40
$SBATCH –time =00:02:00
$SBATCH – men-per-cpu =100
srun ./main 36000 states/0_patch.xml
exit 0;
```

Figura 9: Ejemplo de script

En este ejemplo se puede ver que se está pidiendo las maquinas del hpca4se, concretamente cinco nodos donde por cada nodo se requiere cuarenta tareas, también se está limitando la ejecución del trabajo en dos minutos y requiriendo un mínimo de memoria de 100 MB por cpu.

Una vez declarado los parámetros, en el mismo script, se indica el lanzamiento del trabajo mediante el comando srun. Al finalizar el script se procede a su ejecución mediante el comando sbatch <nombre del script>.

La verificación del estado del trabajo lanzado se realiza mediante scontrol show job <id trabajo>. El comando nos muestra, a parte del estado en el que se encuentra el trabajo, parámetros tales como los recursos demandados por el usuario y lo tiempos de encolamiento, empezado y finalizados entre otros.

7 RESULTADOS

Una vez finalizada la aportación al modelo paralelo, se realizó un conjunto de pruebas de rendimiento para conocer el speedup respecto a la ejecución serie del modelo.

El primer test se realizó variando la cantidad de procesadores utilizados, entre 2, 4 ,8 y16, sobre una simulación de 100 años (36000 iteraciones). Los resultados obtenidos se muestran en la figura 10.

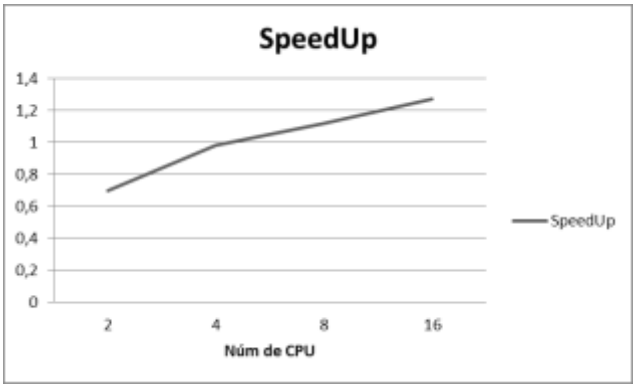

Figura 10: Grafica de speedUp 100 años

Una vez analizados los resultados se puede ver que utilizando 4 procesadores se obtiene el mismo rendimiento que en la versión serie, mientras que si aumentamos a 8 o 16 se consigue mejorar el tiempo serie. También el análisis de los resultados nos indica que el aumento de 2 a 4 procesadores, el speedUp aumenta de mayor forma que de la manera que aumenta de 4 a 8 o de 8 a 16 procesadores.

En un nuevo test se comprobó cómo afectaba al rendimiento el disminuir el tamaño del problema. En este caso se simularon 50 años (18000 iteraciones). Los resultados obtenidos se muestran en la figura 11.

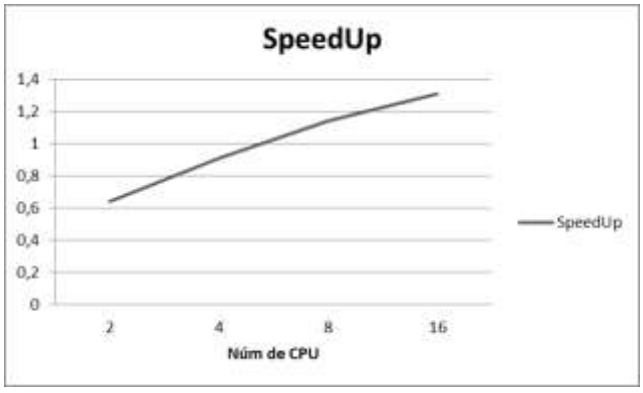

Figura 11: Grafica de speedUp 50 años

Respecto a la simulación anterior de 100 años, no se vio ningún cambio significativo en el rendimiento, por ello se hizo un nuevo test donde la cantidad de trabajo se redujo a 10 años con los resultados que se muestran en la figura 12.

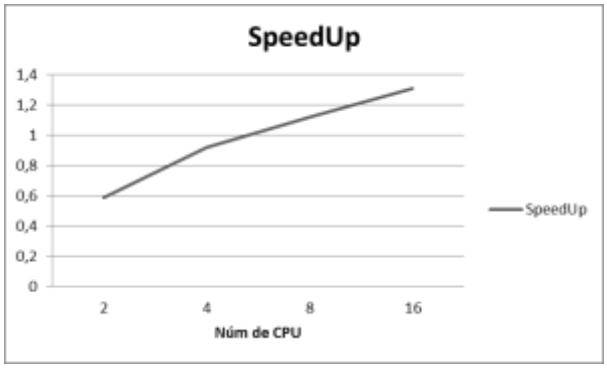

Figura 12: Grafica de speedUp 10 años

Al reducir la cantidad de repeticiones, tanto de 100 años a 50 años y 10 años, no se observan cambios significativos en el rendimiento del modelo respecto a la versión serie.

Hay que tener en cuenta que las ejecuciones se han realizado en el nodo Pinguin cuyo nodo incorpora 32 núcleos, por lo tanto aunque en las pruebas se pidieran 16 núcleos, no se ha utilizado hyperthreading. El motivo por el que no se ha incluido en las gráficas los resultados utilizando más núcleos, ha sido por la carga de trabajo que tendrían. Esto es porque el número de agentes creados inicialmente, provoca que hayan cores sin una gran cantidad de trabajo.

Una vez obtenidos los resultados expuestos anteriormente hay que indicar que este proyecto no se ha enfocado en optimización por lo tanto queda en un trabajo futuro el intentar sacar el máximo rendimiento posible al cluster

8 Conclusión

En este proyecto se ha desarrollado funcionalidades de los agentes individuo y clanes sobre el modelo paralelo. Las nuevas características permiten a la simulación que los individuos se puedan emparejar y también ahora pueden tener descendencia siguiendo unas determinadas reglas en ambos casos. Por lo que respecta al clan, se le ha añadido la posibilidad de dividirse al llegar a un número determinados de miembros y se le ha dotado de un gen lingüístico que permite diferenciar idiomas diferentes.

El modelo en paralelo se está realizado para ver si mejora a la versión serie, por lo tanto, al finalizar las funcionalidades incluidas se ha hecho un análisis de los resultados obtenidos de la ejecución del modelo en el CRAG. Después del análisis de los resultados se ha podido verificar que con el modelo en paralelo se consigue un mejor rendimiento que el modelo serie.

9 Líneas futuras

Actualmente las funcionalidades que se habían planificado añadir al modelo inicial ya se cumplieron, además de añadir una parte del factor culturar que no estaba contemplado en la planificación inicial. Recordando un punto indicado al principio del documento, el modelo no estará completo con la finalización del presente trabajo y el trabajo de Rodríguez. En este caso el trabajo próximo a realizar al modelo:

Caza. Añadir la posibilidad de cooperación entre clanes con el fin de obtener calorías a través de la caza de guanacos.

Cultura. Completar el factor cultural del clan con la utilidad en las relaciones entre clanes.

Trabajo de optimización del modelo con tal de aumentar el rendimiento en paralelo.

AGRADECIMIENTOS

El autor quiere aprovechar este espacio para agradecer al Dr. Eduardo Cesar Galobardes por haber dirigido y supervisado el proyecto.

BIBLIOGRAFÍA

[1] Marco A. Janssen, Agent-Based Modelling, March 2005

[2] SIMULTPAST. [en línea] http://www.simulpast.es [consulta : 16 enero 2015]

[3] F. del Castillo, J.A. Barceló, J.A.Cuesta, J.M.Galán, L. Mameli, F.Miguel, J. Santos, X.Vila.,Simulating ethnogenesis and ccultural diversity in small-scale social formations, February

[4] CRAGENOMICA. [en línea]<http://www.cragenomica.es> [consulta : 16 septiembre 2014]

[5] Hemalatha Vulchi, Agent Based Social Modeling with FLAME Simulation Framework, June 2014

[6] FLAME. User manual. [en línea] <http://www.flame.ac.uk/docs/user_manual.html> [consulta : 16 septiembre 2014]

[7] Alban ROUSSET, Bénédicte Herrmann, Christophe LANG, Laurent PHILIPPE. A survey on parallel and distributed Multi-Agent Systems , 2014

[8] INTECO. INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA. [en línea] <https://www.inteco.es/file/N85W1ZWFHifRgUc_oY8_Xg>[consulta : 13 octubre 2014]

[9] SLURM, Sistema de colas https://computing.llnl.gov/linux/slurm/ [en línea] [consulta: 28/01/2015]

[10] Andrés Rodríguez, TFG: Simulación paralela basada en agentes de socie-dades precolombinas: movimiento de guanacos y clanes, febrero 2015.

APÉNDICE

A1. MÁQUINAS DE ESTADOS DE LOS AGENTES CLAN E INDIVIDUO.