Interfaz de usuario basada en agentes para la administración de trabajos en un entorno de alto rendimiento

User Interface Based on Agents by Workload Management in a High Throughput Environment

Recibido: 21 de octubre de 2013 • Aprobado: 18 de marzo de 2014

Para citar este artículo: H. Bolívar, L.E. Barreto, «Interfaz de usuario basada en agentes para la administración de trabajos en un entorno de alto rendimiento», *Ingenium*, vol. 15, n.º 29, pp. 37-48, mayo, 2014.

Holman Bolívar Barón*, Luis Eduardo Barreto** y Andrés Armando Sánchez***

Resumen

Los sistemas distribuidos han permitido el desarrollo de entornos computacionales capaces de realizar grandes cantidades de procesamiento durante largos periodos de tiempo. El presente trabajo aborda el problema de la manipulación asociada a la complejidad inherente en la maximización de los recursos a través del desarrollo de un sistema multiagente, aprovechando su naturaleza distribuida, autónoma y solidaria. Aunque, en obras relacionadas, se plantean problemas similares, en estas se ha ignorado la falta de control que el administrador de trabajos tiene sobre los recursos. Por lo tanto, los investigadores proponen una interfaz de usuario basada en agentes para la administración de trabajos en un entorno de alto rendimiento, utilizando el enfoque del Telecom Italia Lab

* Ph. D. (c) Informática, Universidad Pontificia de Salamanca. Ingeniero de Sistemas, Universidad Católica de Colombia, DEA, director del Grupo de Investigación en Software Inteligente y Convergencia Tecnológica GISIC y director del proyecto: «Plataforma de optimización de renderización de ambientes virtuales de aprendizaje inmersivo para interfaces de usuario 3d en dispositivos móviles», Universidad Católica de Colombia. E-mail: hdbolivar@ucatolica.edu.co

** M. Sc. (c), Ingeniería, Pontificia Universidad Javeriana. Ingeniero de Sistemas, Universidad Católica de Colombia., Ingeniero del Proyecto Gris Colombia de la Red Nacional Académica de Tecnología Avanzada le.Universidad Catolica de Colombia.. E-mail: barreto62@ucatolica.edu.co

*** M. Sc. (c), Ingeniería, Pontificia Universidad Javeriana, Ingeniero de Sistemas, Universidad Católica de Colombia, profesor, Universidad Católica de Colombia. E-mail: aasanchez60@ucatolica.edu.co

(TILAB), gracias a que el sistema fue desarrollado utilizando el Java Agent Development Framework (JADE), debido a la facilidad para el cumplimiento de los estándares de la Intelligent Physical Agents Fundation (FIPA).

Palabras clave

Interfaz de usuario, alto rendimiento computacional, sistemas multiagente.

Abstract

Distributed systems have enabled the development of high throughput computing environments capable of processing large amounts over long periods of time. This paper addresses the problem of handling the inherent complexity associated with the maximization of resources through the development of a multi-agent system, taking advantage of its distributed nature, independent and united. Although, in related works, similar problems arise in these has ignored the lack of control the workload management has on resources. Therefore, the researchers propose a user interface for agent-based workload management in a high performance environment, using the approach of Telecom Italia Lab (TILAB), because the system was developed using the Java Agent Development Framework (JADE), due to the ease of compliance with standards of the Intelligent Physical Agents Foundation (FIPA).

Key words

User interface, high throughput computing, multi-agent systems.

I. Introducción

La mayoría de las investigaciones orientadas al uso de la computación en ambientes científicos, se ha centrado en el número de operaciones de punto flotante por segundo que puede proporcionar el entorno computacional. Este es el criterio que utiliza la computación de alto desempeño o High Performance Computing (HPC) [1]. Por otra parte, la comunidad informática, poca atención ha dedicado a entornos que pueden proporcionar grandes cantidades de capacidad de procesamiento durante largos períodos de tiempo, los cuales son denominados: entornos de alto rendimiento computacional o High Throughput Computing (HTC) [1].

El rendimiento computacional, se puede convertir en un factor limitante en variados esfuerzos científicos y de ingeniería. Por ejemplo, si hay que fabricar un chip, que tiene una ventana de tres meses para ejecutar muchas simulaciones antes de llevarlo al mercado, es un problema HTC o si un físico de alta energía requiere hacer una simulación de reconstrucción, enriqueciéndola con Monte Carlo, es un proyecto de uno o dos años donde el requerimiento significativo de cómputo más que en tiempos de respuesta se da en la generación de estadísticas durante un periodo de tiempo prolongado [2].

Partiendo de la importancia de los entornos HTC para la investigación científica y la ingeniería, el presente trabajo aborda el problema de la manipulación asociada a la com-

plejidad inherente en la maximización de los recursos en los sistemas distribuidos [1] [3], para lo cual se emuló un entorno HTC a partir de una arquitectura de clúster genérico que extiende los beneficios clásicos de las máquinas virtuales, proporcionando soporte a entornos heterogéneos según un presupuesto fijo, y estimando el costo de completar una carga de trabajo determinada en un tiempo específico.

Como una solución al problema de la complejidad se ha propuesto la implementación de un sistema multiagente (MAS) debido a que la coordinación y cooperación entre agentes que poseen diversos conocimientos y capacidades, facilita el logro de objetivos globales. Los MAS han demostrado ser muy apropiados para la ingeniería de sistemas abiertos, distribuidos y heterogéneos [4]. En este trabajo se presenta un modelo que se integra a un prototipo de interfaz para la gestión de trabajos en un entorno de alto rendimiento.

Para el desarrollo de la investigación, en primera instancia se configuró un clúster HTC, enseguida se configuró el protocolo de intérprete de órdenes seguras o Secure SHell (SSH) [5] utilizando JSch [6] y el middleware HTCondor [1]. Para relacionar la aplicación de gestión de trabajos con el clúster se desarrolló una base de datos que se conecta con una interfaz de programación de aplicaciones y finalmente se realizaron pruebas de carga de trabajo en tiempos objetivos asociados al porcentaje de uso de recursos.

La estructura del artículo es la siguiente: en la sección II se presentan trabajos relacionados con la implementación de sistemas multiagentes en entornos distribuidos, seguido por la sección III que describe los factores a tener en cuenta en el diseño de entornos de alto rendimiento, en la sección IV se describe la metodología utilizada para la implementación de HTCondor y su integración con la interfaz de usuario desarrollada basada en agentes, luego en la sección V se muestran los resultados de las pruebas realizadas y finalmente se concluye y se presentan los trabajos futuros.

II. Sistemas multiagentes en entornos distribuidos

Existen diversos estudios sobre modelos multiagentes como solución a problemas de administración de tareas entre los que se destaca el Agent-based Resource Allocation Model (ARAM) que utiliza agentes estáticos y móviles buscando servicios flexibles, inteligentes y adaptables. Algunos de los beneficios de ARAM son: negociación flexible, adaptabilidad, personalización, escalabilidad, facilidad de mantenimiento, y soporte para el desarrollo de software basado en componentes [7]. Sin embargo, no se especifica el cumplimiento del estándar propuesto por la Foundation for Intelligent Physical Agents (FIPA) [4], lo cual dificulta la integración con otros sistemas multiagentes, mientras que otros como el propuesto por Aversa [8], plantea un modelo sobre la implementación de servicios web, con ayuda del Web Service Resource Framework (WSRF) que permite mayor flexibilidad y acoplamiento, suponiendo una administración y programación de recursos distribuidos que se integra a todo el modelo, pero sin resolver el problema de la complejidad asociada a la optimización de los recursos.

Wu, et ál., recomiendan el uso de MAS para resolver el problema de la programación de recursos [9], debido a la flexibilidad, la capacidad de reaccionar de forma incremental a la aparición de nuevos eventos, como por ejemplo, la llegada de una nueva orden y la proactividad de los agentes, que les permite mejorar las asignaciones previamente establecidas de acuerdo al rendimiento.

Mulder y Jacobs proponen el sistema Collaborative Tracing Information System (CTIS) el cual consiste en agentes que reúnen los datos de tráfico de los diferentes nodos, para lograr una visión general. CTIS utiliza un mecanismo de aprendizaje colaborativo obteniendo como resultado, observaciones de los modelos locales que se comparten para obtener un modelo global. Se utilizan agentes debido a que se tienen aspectos dinámicos, multiorganizacionales y descentralizados [10]. Zhu, et ál., presentan un modelo de tres capas donde: la inferior comprende recursos físicos como CPU y memoria; la intermedia es la de agente de vigilancia que comprende una gran cantidad de agentes dinámicos para vigilar el estado de los recursos, y La superior es la cooperación de agente de administración que comprende las asociaciones relacionadas de diferentes organizaciones [11].

Otro modelo es el Performance Analysis and Characterization Environment (PACE), el cual proporciona datos cuantitativos sobre el rendimiento de trabajos, donde la velocidad de respuesta a una solicitud depende del número de rutas factibles que se puedan utilizar para llegar al agente proveedor del servicio y del número de requerimientos que se realicen en una unidad de tiempo. En este modelo Junwei, et ál., proponen como estrategias para optimizar la administración del servicio [12]. De igual manera que Chi, et ál., proponen un modelo llamado Decision Support Systems (DSS) abierto y basado en agentes. El modelo se caracteriza por la dinámica y complejidad de los DSS. Se oculta de manera eficaz la heterogeneidad de los recursos con una buena inteligencia, escalabilidad y capacidad de adaptación, teniendo en cuenta que los agentes autónomos son capaces de proporcionar servicios como la autoconfiguración, autocuración, autoprotección y optimización de sí mismos [13].

III. Factores en el diseño de clúster de alto rendimiento

Un entorno de alto rendimiento a nivel clúster HTC se puede construir de diversas formas según su arquitectura en hardware y sistema operativo, pueden ser homogéneos, los cuales mantienen una arquitectura en hardware y sistema operativo similar; semihomogéneo con diferente arquitectura en hardware y sistema operativo similar. O heterogéneo con diferente arquitectura de hardware y sistema operativo. Esta versatilidad al momento de configurar los clústeres los hace flexibles, fáciles y económicos.

Los componentes de hardware y software básicos necesarios para configurar un clúster son:

- Nodos: es una entidad que debe contar con una unidad de procesamiento, memoria y tarjeta de red como mínimo.

- Sistemas operativos: capa software sobre cada nodo y se encarga de las funciones básicas para el control del hardware de cada computador.
- Conexión de red entre los nodos: Para que el clúster trabaje mejor la red debe ser de alta velocidad.
- Middleware: es una capa de software que se encarga de gestionar el envío, encolamiento, priorización de tareas y la comunicación entre los nodos cumpliendo con el paralelismo de los procesos.
- Aplicaciones que se instalan sobre el middleware con funciones específicas.
- Ambientes de programación paralela: es una ventaja que da el middleware para que los desarrolladores de software aprovechen el súper cómputo que brinda el clúster y aplicaciones en paralelo.

Al momento de la configuración se debe tener en cuenta: acoplamiento, control, homogeneidad y escalabilidad.

La figura 1, presenta los factores que interactúan en el diseño de entornos de alto rendimiento.

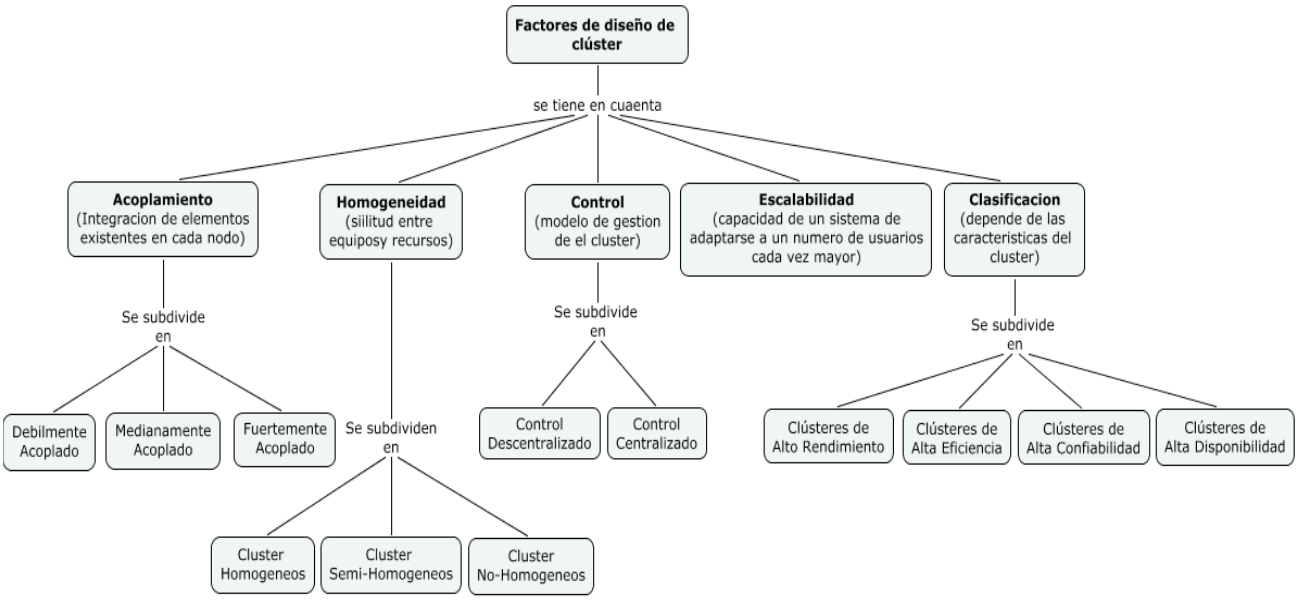

Figura 1. Factores en el diseño de entornos de alto rendimiento.

IV. Metodología de investigación y desarrollo

Para el desarrollo del presente trabajo fue necesario llevar a cabo las siguientes etapas:

A. Configuración del entorno de virtualización HTC

Para la configuración del entorno de alto rendimiento se utilizó Cóndor [1] el cual es un middleware creado por el Cóndor Team de la Universidad de Wisconsin-Madison.

La figura 2 presenta la arquitectura de la configuración del entorno sobre la cual se probó el prototipo interfaz, la distribución de sus demonios y la relación de comunicación que tienen entre ellos.

El CE «Computer Element», se encarga de gestionar la cola y recursos del clúster; las negociaciones y comunicación entre equipos. El sistema de archivos en red «NFS» encargado de la sincronización y gestión de datos. Las máquinas «Client Node» desde donde se envían los trabajos a los equipos de ejecución denominados «WN Worker Node», en los cuales radica la capacidad de procesamiento del clúster.

Cada equipo del clúster tenía instalado el sistema operativo Scientific Linux 5.5 de 32 bits y como middleware cóndor-7.4.4-1.rhel5.i386.

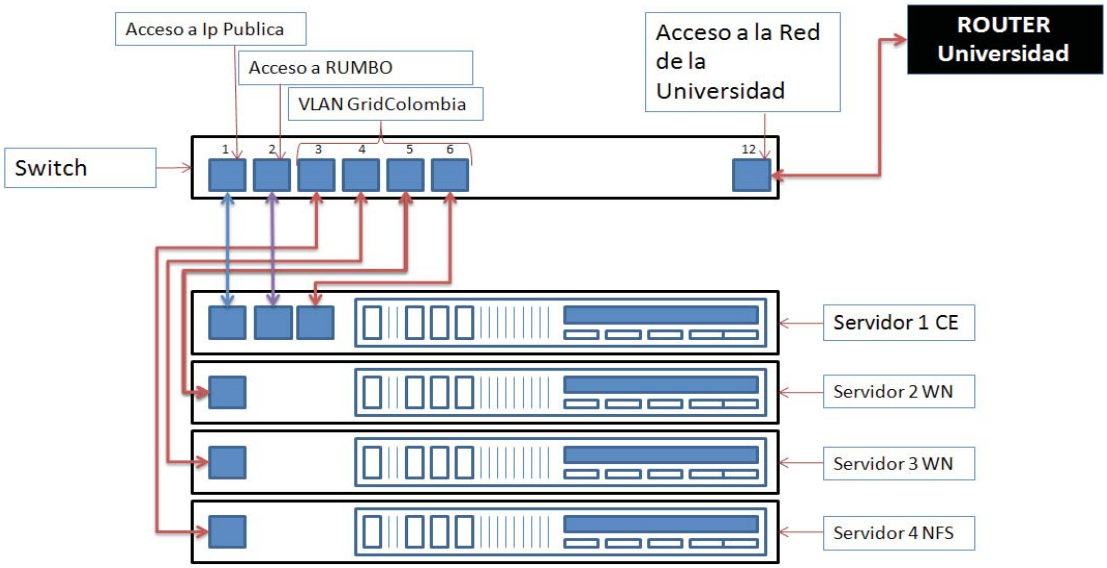

Figura 2. Entorno de virtualización HTC.

B. Desarrollo de la interfaz de usuario

La metodología utilizada para el desarrollo de la interfaz de usuario se basó en la metodología TSP, «Team Software Process», y la PSP, «Personal Software Process», las cuales se enfocan en distintos KPA «Key Process Areas» del desarrollo de un proyecto o de la operación de la organización [14].

Para el diseño de la aplicación interactiva se utiliza el patrón Modelo Vista Controlador (MVC), donde se buscó desacoplar la vista del modelo con la finalidad de mejorar la reusabilidad, de esta manera las modificaciones en las vistas impactan en menor medida.

La gráfica 3, presenta la ejecución de una tarea en el entorno HTC a partir de una prueba sencilla probando el cálculo de *n* números primos en C.

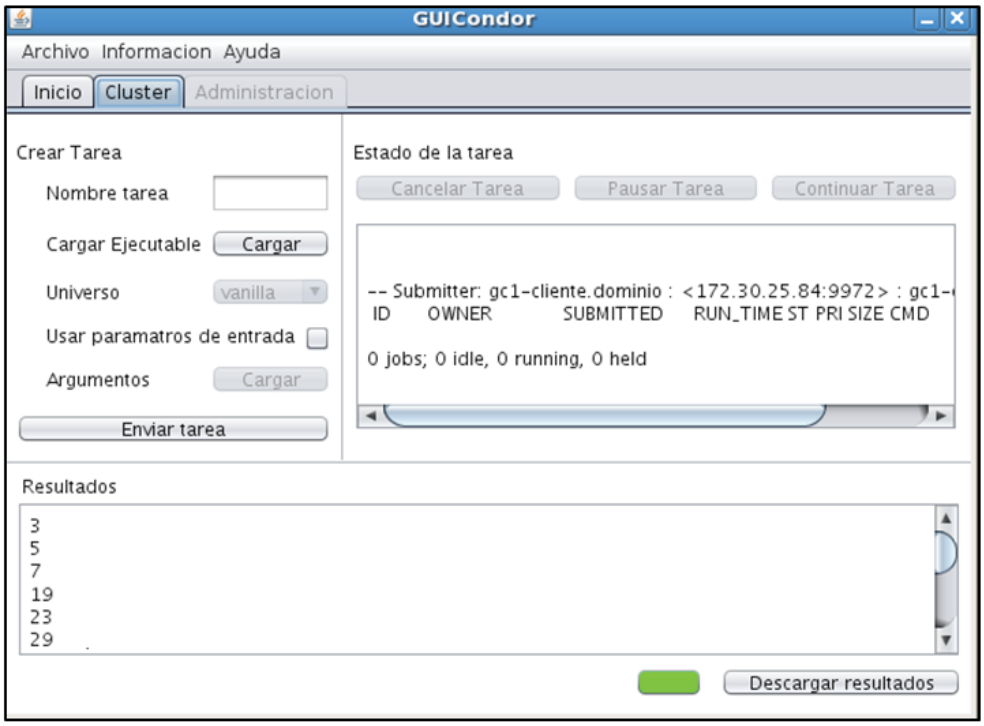

Figura 3. Interfaz gráfica de usuario.

C. Pruebas de manipulación de trabajos y uso de recursos preagentes

Para validar el rendimiento del sistema se realizarán varias pruebas de carga buscando maximizar los recursos del clúster, con diferentes criterios de rendimiento y de optimización, debido a que es un problema multiobjetivo, en su formulación general se tuvieron en cuenta los criterios [15] :

- Porcentaje de utilización de los recursos en general.
- Porcentaje de utilización de cada recurso.
- Rendimiento acumulado.
- Plazos incumplidos.
- Tiempo total ponderado de realización.

Debido a la alta dependencia del tiempo de ejecución de una tarea respecto a la aplicación a la que pertenece y la máquina donde se está ejecutando, para identificar el tiempo de ejecución Al-Azzoni [16] propone clasificar las tareas en grupos (o clases) con distribuciones idénticas para los tiempos de ejecución.

Para lo cual se analizaron 5 escenarios diferentes, modificando el número de dominios, el número de worker nodes por dominio y el número de estados por dominio, tal como se muestra en la tabla 1.

Escenarios	N.º dominios	Worker Nodes (WN) por dominio	N.º estados por dominio
A	30	4	6
B	10	6	
	20	3	8
C	5	10	6
	10	5	
		2	
D	5	10	8
	10	5	
		2	
E	2	18	
	3	9	6
	5	5	8
	6	2	6

Tabla 1. Escenarios de prueba.

Por cada escenario se realizaron 15 pruebas de carga, en la figura 5 se presenta el porcentaje ponderado de uso de recursos de cada escenario.

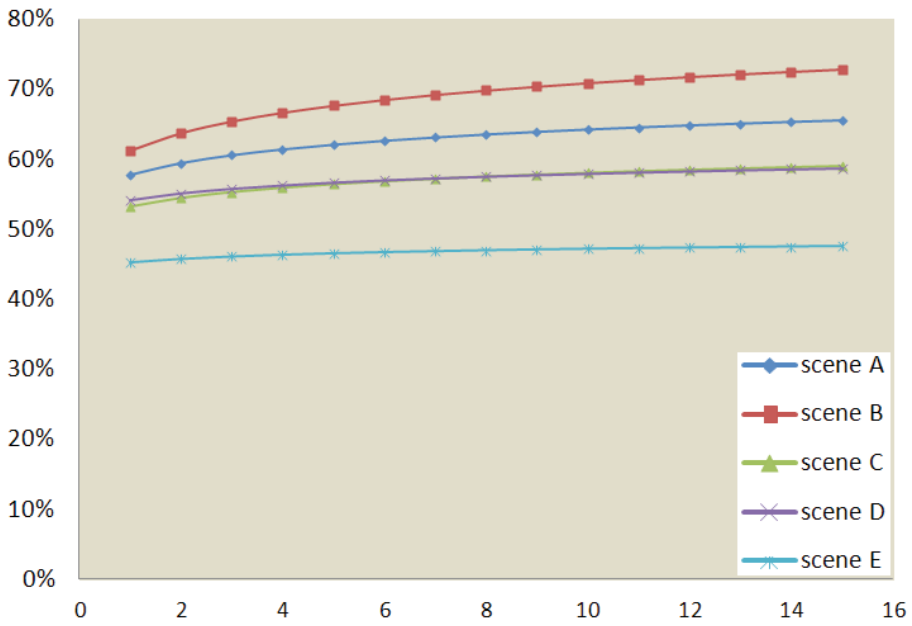

Figura 5. Porcentaje de uso de recursos.

D. Desarrollo del sistema multiagente

Para el desarrollo de la interfaz para la administración de trabajos a partir de agentes, se tomó como base la metodología propuesta por Nikraz [4] debido a que integra las diferentes etapas del desarrollo de software y sus artefactos en UML con distintas metodologías para la construcción de sistemas multiagentes como MESSAGE y GAIA.

De acuerdo al estándar FIPA un MAS debe contener tres componentes: el Agent Management System (AMS) encargado del ciclo de vida de los agentes, de los recursos locales, los canales de comunicación y proporciona un servicio de páginas blancas para buscar los agentes por el nombre; el Directory Facilitator (DF) que presta un servicio de páginas amarillas ubicando a los agentes por su servicio; el Agent Communication Channel (ACC) encargado de gestionar el paso de mensajes entre los diferentes agentes [17].

El MAS se implementó utilizando el Java Agents Development Enterprise Framework (JADE) [17], donde todo agente hereda de la clase Agent, que se encuentra ubicado en el paquete jade.core. Se debe implementar como mínimo el método void setup(), el cual realiza todas las tareas necesarias para que el agente pueda comenzar su ejecución, para el paso de mensajes se utiliza la clase ACLMessage que se encuentra en el paquete Jade.lang.acl, que luego se transformará en un flujo de bits para representar la estructura de mensajes FIPA [18]. Las funcionalidades de los agentes se definen mediante comportamientos, los cuales son los encargados de la preparación del trabajo para su ejecución por medio de la generación de los script necesarios para tal fin. El JobAdapterBehaviour establece las variables de en torno necesarias, invoca los scripts generados para la ejecución e informa sobre cualquier posible error o cambio.

En un agente pueden estar activos a la vez tantos comportamientos como sea necesario. Decidir cuál se ejecuta en cada momento es tarea del AMS, para que cada agente tenga un único hilo de ejecución, con el consiguiente ahorro de ciclos de CPU y memoria que esto implica [17].

La cola de mensajes es única para cada agente y, por lo tanto, es compartida por todos los comportamientos, un comportamiento puede ser bloqueado en espera de la recepción de un mensaje. Quien define el tipo de diálogo que se establece entre los agentes es la performativa que es el único campo obligatorio en el mensaje entre dos agentes, tomada directamente del estándar.

La figura 5 presenta el modelo estático de la interfaz, donde se evidencia el paquete exception con las clases legalOrphanException, PreexistingEntityException y NonexistentEntityException, el paquete JSch para controlar la creación de sesión y el canal de comunicación, una clase dedicada a la comunicación vía el protocolo ssh y las clases asociadas a los paneles de usuarios como la clase UsurioappJpaController, Información del usuario y la administración del clúster.

Se utilizaron maquinas dedicadas buscando identificar la capacidad de una máquina para ejecutar una tarea y si es el caso poder migrarlo hacia otra máquina que se encuentre libre, proceso conocido como «checkpointing», la interfaz también permite pedir el estado de una tarea y en qué máquinas se está ejecutando, enviar una tarea a múltiples máquinas, conocer los recursos de cada máquina del sistema y administrar la cola de los trabajos.

Se implementó el sistema ClassAd, que proporciona un diseño claro que simplifica el envío de trabajos y encontrar la máquina adecuada para ejecutar el proceso.

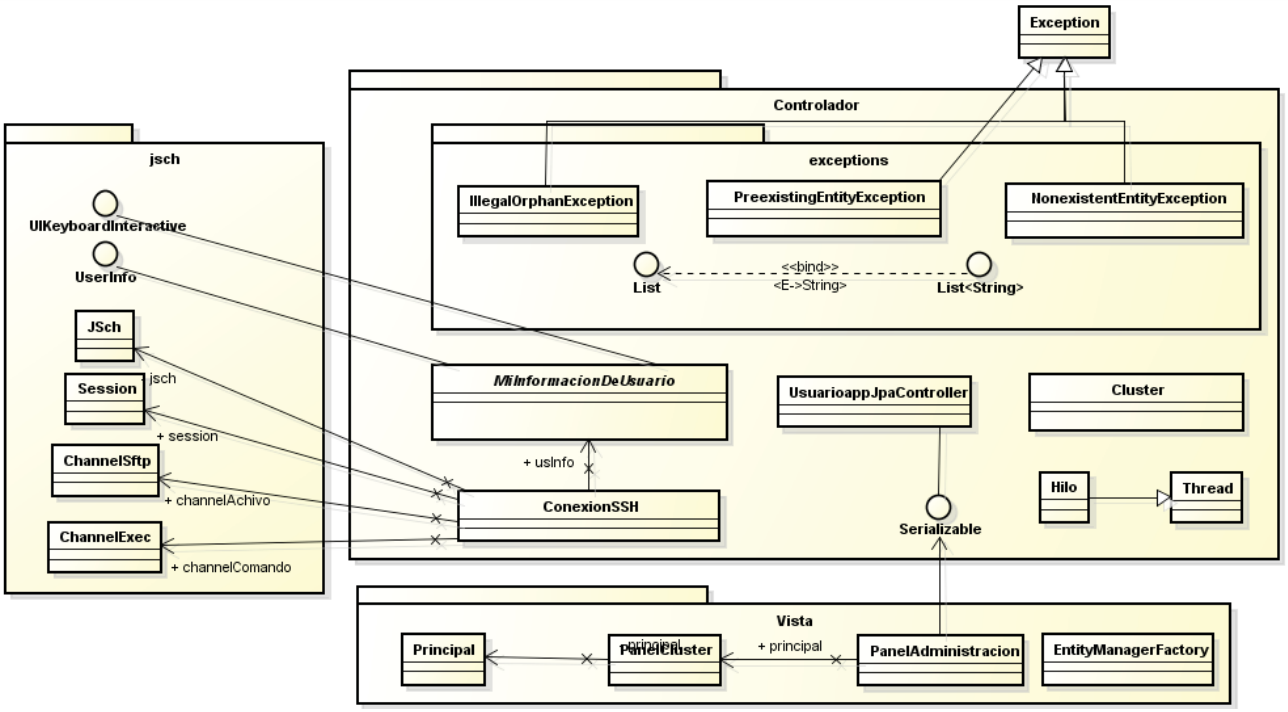

Figura 5. Modelo estático de la interfaz de usuario basada en agentes para la administración de trabajos en un entorno de alto rendimiento.

Cóndor crea entornos de ejecución con características especiales llamados universos. Un universo define las características que se necesitan según los requerimientos de las tareas a ejecutar. Para el presente trabajo se utilizó el Universo Vanilla el cual es el universo por defecto, es el menos restrictivo con los trabajos que se pueden enviar. Acepta cualquier programa escrito en cualquier lenguaje. Para acceder a los datos de otras máquinas se utilizó el sistema de archivos NFS.

La comunicación entre la interfaz alojada en una máquina cliente y el CE del clúster se hace a través del protocolo SSH para evitar intercepción de la comunicación por medio del uso de firmas digitales para verificar la identidad de los usuarios. Adicionalmente, toda la comunicación entre los sistemas cliente y servidor está encriptado.

V. Pruebas y validación

Tomando como base los escenarios presentados en la tabla 1, luego de integrar el SMA a la interfaz de usuario y al clúster HTCondor, se realizaron las mismas pruebas y los resultados obtenidos se presentan en la figura 6 (ver figura 6 en la siguiente página).

Se evidencia un incremento en el uso de los recursos cercano al 80 % en los escenarios con menor número de nodos y un descenso cercano al 15 % en el uso de los recursos en los escenarios con mayor número de nodos.

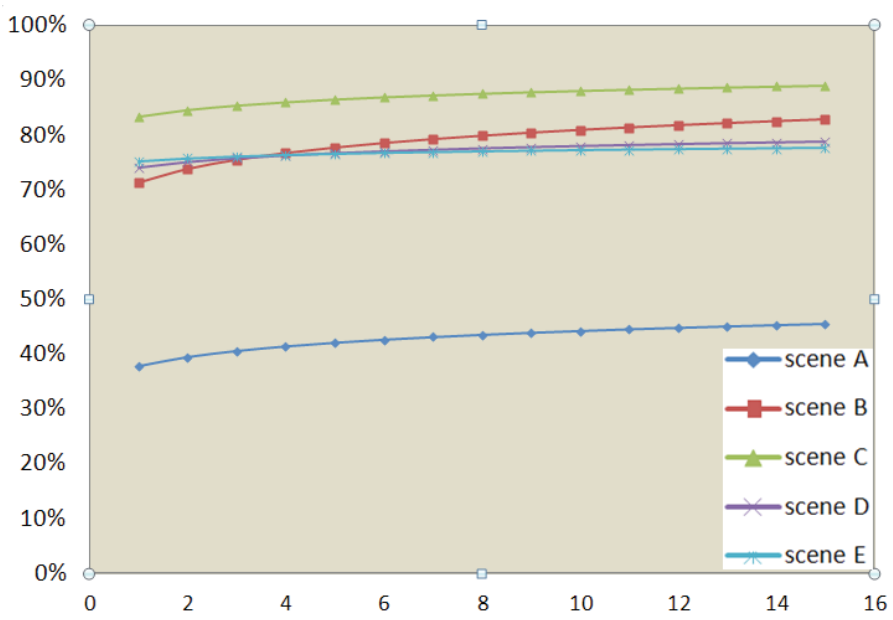

Figura 6. Porcentaje de uso de recursos con SMA.

La asignación del worker node libre, para la ejecución de una tarea corresponde a una evaluación de la probabilidad de la secuencia a partir del registro de los históricos de disponibilidad del servidor analizado $O = (o_1\ o_2\ \ldots\ o_T)$, dado el modelo λ, es decir, $P(O|\lambda)$. La forma de resolver el problema consiste en aplicar un algoritmo de avance (*forward algorithm*), en el cual se parte de $\alpha_t(i) = P(o_1\ o_2\ \ldots\ o_t, q_t = i \mid \lambda)$, entonces la probabilidad de la secuencia de observación parcial $o_1\ o_2\ \ldots\ o_t$ en el estado *i* hasta el tiempo *t*, dado el modelo λ. Puede calcularse, así: $\alpha_t(i)$:

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \le i \le N$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), 1 \le t \le T-1, 1 \le j \le N$$

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

VI. Conclusiones y trabajos futuros

El modulo GRAM de cada dominio, facilita la integración de JADE gracias a la administración de los archivos JDL los cuales pueden ser gestionados desde java que facilita el paso de mensajes ACL entre los agentes.

La parametrización de los trabajos en la implementación del MAS encargado de controlar el envío, recepción y monitoreo de trabajos asegura en un 72,25 % la fiabilidad de los archivos y requerimientos a los dominios de ejecución, mientras que sin la parametrización la fiabilidad es solo del 35 %.

El porcentaje ponderado de uso de los recursos del sistema, para el dominio uniforme fue del 66 % con un porcentaje de entregas del 73 %, lo cual refleja un incremento ponderado del 8 % con respecto a la primera prueba, donde no se tiene entrenamiento en los worker nodes de acuerdo a la fórmula de probabilidad de la secuencia a partir del registro de los históricos.

Aunque hay incremento y es constante en el tiempo, cada vez es menor y para asegurar un porcentaje de efectividad del 90 % se requerirían más de 15 000 pruebas, sobre una misma especificación del sistema, por lo tanto como investigación futura se pretende identificar un modelo para asignar la distribución de probabilidad asociada a cada worker node, que permita un porcentaje de efectividad mayor en menor tiempo.

Referencias

[1] HTConder. [En linea] http://research.cs.wisc.edu/htcondor/htc.html

[2] A. Beck, High Throughput Computing: An Interview with miron livny. [En linea] http://research.cs.wisc.edu/htcondor/HPCwire.1

[3] R. Montero, R. Moreno and I. Llorente. «An elasticity model for High Throughput Computing clusters». *Journal of Parallel and Distributed Computing, v*ol. 71, Issue 6, June 2011, pp. 750-757. www.sciencedirect.com/science/article/pii/S0743731510000985

[4] M. Nikraz, B. Caire, G. Bahri. «A methodology for the analysis and design of multi-agent systems using JADE». *Proceedings of International Journal of Computer Systems Science and Engineering*. 2006, p. 40.

[5] www.openssh.org/

[6] www.jcraft.com/jsch/

[7] S. Manvi, M. Birje, «An agent-based resource allocation model for grid computing». En: *Services Computing, 2005 IEEE International Conference on*. Noviembre 2005, vol. 01, pp. 311-314.

[8] R. Aversa, B. Di Martino, S. Venticinque. «Integration of Mobile Agents Technology and Globus for Assisted Design and Automated Development of Grid Services». En: *Proceedings of the 2009 International Conference on Computational Science and Engineering*. 2009. Vol. 01, pp. 118-125.

[9] J. Wu, X. Xu, P. Zhang, C. Liu. «A novel multi-agent reinforcement learning approach for job scheduling in Grid computing». *Future Generation Computer Systems*, n.° 27, 2011, pp. 430-439.

[10] W. Mulder, C. Jacobs. «Grid management support by means of collaborative learning agents». *Proceedings of the 6th international conference industry session on Grids meets autonomic computing*, 2009, pp. 43-50.

[11] Y. Zhu, J. Gao, B. Jiang, F. Zhang, H. Guo. «An Agent Federations Based Infrastructure for Grid Monitoring and Discovery». *Machine Learning and Cybernetics, 2006 International Conference on*, 2009, pp. 13-16.

[12] C. Junwei, J. Kerbyson, J. Nudd. «Performance evaluation of an agent-based resource management infrastructure for grid computing». *Cluster Computing and the Grid*, 2001, Proceedings. First IEEE/ACM International Symposium on, 2002, pp. 311-318.

[13] D. Cox, Y. Al-nashif, S. Hariri. «Application of autonomic agents for global information grid management and security». Proceedings of the 2007 summer computer simulation conference, 2007, pp. 1147.

[14] J. L. Morena. (2004) Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres en Colección de Tesis Digitales Universidad de las Américas Puebla. [En línea] Disponible: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo2.pdf

[15] F. Xhafa, A. Abraham, «Computational models and heuristic methods for Grid scheduling problem». *Future Generation Computer System*. 2010, vol. 26, pp. 608-621.

[16] I. Al-Azzoni and D. G. Dwon, «Decentralized Load Balancing for Heterogeneous Grids», in *Future Computing, Service Computation, Cognitive, Adaptive, Content, Pattern. Hamilton, ON, Canada*, Nov. 2009, pp. 545-550.

[17] F. Bellifemine, G. Caire, D. Greenwood. «Developing Multi-Agent Systems with JADE». England. Editorial John Wilye & Sons Ltd. 2007, pp. 115.

[18] FIPA. «FIPA Agent Management Specification». Foundation for Intelligent Physical Agents. 2004. www.fipa.org/