# Jerarquización de la Calidad de Servicio en Grids mediante SLAs

Javier Conejero, Luis Tomás, Carmen Carrión, Blanca Caminero [1]

*Resumen*— **La necesidad de mecanismos para garantizar la calidad de servicio (*Quality of Service* (QoS)) en entornos de computación Grid ha potenciado el interés en los acuerdos de nivel de servicio (*Service Level Agreements* (SLAs)). Los SLAs son contratos de servicio entre usuarios y proveedores de servicios establecidos mediante una negociación.**

**El objetivo de este artículo es afrontar la diferenciación de la QoS en niveles, de forma jerarquizada, en entornos de computación Grid mediante el uso de métricas específicas en los SLAs. Además, la caracterización y clasificación de estos términos, haciendo una diferenciación en niveles, es importante para mejorar la cantidad de acuerdos completados satisfactoriamente. Nuestra propuesta se basa en una arquitectura compatible con la especificación WS-Agreement, desarrollada para ofrecer soporte de SLAs en entornos Grid, así como proveer un marco adecuado para la definición de nuevos términos y políticas.**

**Además de presentarse una propuesta genérica, se evalúa una implementación capaz de ofrecer tres niveles de QoS. Estos niveles son utilizados para clasificar el compromiso adquirido en cada SLA (dependiendo de las expectativas del usuario que lo ha formalizado) y optimizar el funcionamiento del sistema en función de la confianza depositada en los recursos asociados al Grid. La evaluación de esta propuesta en un entorno Grid real muestra su eficicacia y eficiencia.**

*Palabras clave*— **Acuerdos de nivel de servicio; SLA; WS-Agreement; Planificación por adelantado; Reserva; Grid;**

## I. Introducción

La explotación de tecnologías orientadas a servicios, como entornos de computación distribuidos o de alto rendimiento, está en auge actualmente. La importancia de garantizar la QoS y los modelos *pay–per–use* requieren nuevos mecanismos para reforzar esta tendencia [1]. La computación Grid es una de las tecnologias en las que esta tendencia está teniendo un gran impacto, debido fundamentalmente a intereses de explotación económica de sus recursos. Esto fomenta la investigación en el uso de mecanismos como los acuerdos de nivel se servicio (SLAs) para proveer QoS [2].

Un SLA es un contrato entre un usuario y el proveedor de servicio, en el cual se especifican de forma explícita todas las expectativas y obligaciones de ambos participantes de su negociación [3]. Representa una declaración legal de la QoS que el usuario espera recibir por parte del proveedor de servicio, así como las responsabilidades que ambos participantes asumen si los términos no fuesen satisfechos.

El ámbito de este artículo es definir múltiples niveles de QoS en Grids utilizando SLAs. Más detalladamente, desarrollamos una propuesta basada en tres niveles, donde los niveles están definidos explícitamente en los SLAs. La razón principal para definir diferentes niveles se basa en el hecho de que, en entornos Grid, es posible encontrar recursos con diferentes niveles de confianza y/o disponibilidad. Por consiguiente, la utilización de esta información es necesaria para priorizar los trabajos asociados a cada SLA dependiendo del nivel de QoS solicitado. Es importante observar que este nuevo entorno ha sido desarrollado siguiendo la especificación WS-Agreement [4], en la cual se define la estructura de los SLAs y plantillas de negociación, así como el protocolo de negociación.

La estructura del presente artículo se describe a continuación. Para comenzar, en la Sección II se muestra brevemente el trabajo relacionado. En la Sección III se describe la arquitectura, base para la propuesta. A continuación, en la Sección IV se presenta la propuesta principal de este artículo. Y después, en la Seccion V se desarrollan los detalles de implementación, tales como el diseño de las plantillas de negociación (Subsección V-A) y las políticas de evaluación (Subsecciónn V-B). Posteriormente, en la Sección VI se presenta la evaluación del rendimiento. Finalmente, se describen las conclusiones y trabajo futuro en la Sección VII.

## II. Trabajo relacionado

El campo de los SLAs está siendo actívamente investigado, no solo en cuanto a su especificación, sino también en otros campos como su gestión [5], implicaciones con la QoS [6] o explotación de la semántica y virtualización [7].

Pero los esfuerzos más importantes se concentran en su estandarización, encabezados por la especificación WS-Agreement [4] y su reciente extensión del protocolo de negociación [8]. WS-Agreement está propuesto y soportado por el *Grid Resource Allocation Agreement Protocol Working Group* (GRAAP-WG) del *Open Grid Forum* (OGF) y se considera como el estandar de facto en la actualidad. Hay otras especificaciones disponibles, como WSLA [9] y SLAng [10]. Pero carecen de mantenimiento hoy en día.

La mayoría de proyectos interesados en la adopción de SLAs ha adoptado la especificación WS-Agreement. Pero debido a su falta de un lenguaje de definición de términos y estructura de los mismos, la mayoría de los proyectos ha desarrollado su propia alternativa, causando incompatibilidades entre proyectos, a pesar de utilizar la misma especificación.

Uno de los proyectos que destaca en este ámbito

[1]Instituto de Investigación en Informática de Albacete (I$^3$A), Universidad de Castilla-La Mancha, e-mail: {Becario.FJConejero, Luis.Tomas, MariaBlanca.Caminero, Carmen.Carrion}@uclm.es

es AssessGrid (*Advanced Risk Assessment and Management for Trustable Grids*) [11], cuyo objetivo es la evaluación de riesgos en entornos Grid y la utilización de SLAs.

Otro proyecto interesante es SLA@SOI (*Service Level Agreement at Service Oriented Architectures*) [12]. Está enfocado en la implantación de SLAs en infraestructuras orientadas a servicios. Centrandose fundamentalmente en el problema de la traducción de métricas observables en parámetros configurables [13]. SLA@SOI está siendo activamente desarrollado y mejorado.

## III. Arquitectura

Para el desarrollo de la presente propuesta se emplea la arquitectura software propuesta en [14] (ver Figura 1). Esta arquitectura ofrece soporte para múltiples niveles de QoS mediante SLAs en Grids ajustada a la especificación WS-Agreement.

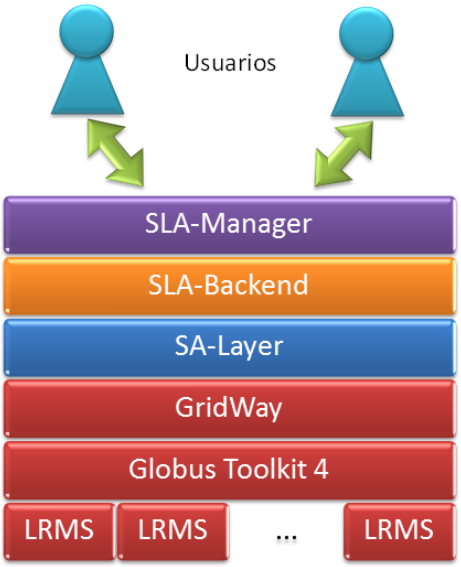


Fig. 1. Arquitectura

La capas superiores, denominadas SLA-Manager y SLA-Backend, son las encargadas de proveer el soporte de SLAs, cumpliendo con la especificación WS-Agreement.

Una característica interesante de esta arquitectura es la capacidad de planificación de trabajos por adelantado gracias a la capa SA-Layer [15]. Esta capacidad es necesaria debido a las restricciones temporales que aparecen en los SLAs, ya que los usuarios pueden solicitar fechas y horas de comienzo y finalización de sus trabajos. Se ha preferido esta capacidad en lugar de realizar reservas por adelantado debido a que las reservas no son siempre posibles en un entorno Grid real.

Finalmente, toda la pila descrita anteriormente se apoya sobre el middleware Globus Toolkit 4 (GT4), el cual está considerado como el estandar "de facto" en Grids. Además, las capas SLA-Manager y SLA-Backend están desarrollados como servicios Web para GT4, y por lo tanto, pueden explotar todas las ventajas provistas por GT4, como: seguridad (GSI), interfaces, etc.

La arquitectura permite la definición de nuevos términos en los SLAs, así como políticas de evaluación (en el SLA-Backend). Esto hace perfecta esta arquitectura para el diseño, implementación y experimentación de nuevos términos.

Como este artículo está enfocado al uso de SLAs, las capas relacionadas con ello (SLA-Manager y SLA-Backend) se describen con mayor detalle en las próximas subsecciones.

### A. SLA-Manager

La capa situada en la cima de la arquitectura es la denominada SLA-Manager. Su objetivo principal es proveer mecanismos para la negociación de SLAs entre usuarios y el proveedor de servicios siguiendo la especificación WS-Agreement. La construcción actual de esta capa implementa la especificación WS-Agreement versión 1.0 [4], pero sin la extensión para la renegociación [8].

Este trabajo se enfoca en la adaptación de las plantillas de negociación (conocidas también como *templates*) y los términos de los SLAs.

Dado que la especificación WS-Agreement considera las plantillas como contratos que pueden ser personalizados para satisfacer las necesidades de los proveedores de servicios, cada modificación aplicada sobre ellos implica la adaptación de esta capa. Pero algunas restricciones han de ser respetadas, como su estructura. La estructura definida por la especificación WS-Agreement para la negociación de las plantillas y SLAs es fija y está compuesta por cuatro bloques.

1. *Nombre*: Donde se especifica el nombre del acuerdo (para identificación humana).
2. *Contexto*: En él se especifican todos los detalles del acuerdo (p.ej., identificadores de cliente y proveedor).
3. *Términos de Servicio*: Contiene todos los términos relacionados con las garantías y requisitos esperados.
4. *Términos de Garantía*: Contiene los valores de QoS para cada uno de los términos de servicio especificados.

Además, aparece otro bloque más en las plantillas, en el cual se especifican todas las restricciones de creación necesarias, denominado *Restricciones*.

La especificación WS-Agreement sin la extensión de negociación define un protocolo de negociación simple, el cual comienza con una petición del usuario al proveedor de servicios para iniciar una negociación. Una vez el usuario recibe una plantilla, la rellena con sus expectativas para los términos de servicio definidos y la devuelve al proveedor de servicios, ahora como SLA. A continuación, el proveedor de servicio comprueba si está bien formada y carente de errores y toma la decisión de aceptación o rechazo del SLA. Para ambas decisiones, el usuario recibe un mensaje con su resultado: un mensaje de rechazo o un mensaje de aceptación (ACK). Este último conteniendo el *EndPoint Reference (EPR)* correspondiente.

### B. SLA-Backend

El propósito de la capa SLA-Backend es tomar la decisión de aceptación o rechazo de las peticiones rea-

lizadas por los usuarios a través del SLA-Manager, así como las medidas apropiadas para asegurar los acuerdos establecidos. Es el responsable de la meta-planificación de los trabajos aceptados teniendo en consideración los términos especificados en cada SLA de cada trabajo no relacionados con requisitos temporales (e.g. economía, consumo energético, etc.). Para tomar las decisiones, se deben implementar pol'iticas de evaluación. Estas, además, pueden ser fácilmente modificadas o actualizadas.

Tal y como se muestra en la Figura 1, el SLA-Backend se situa entre las capas SLA-Manager y SA-Layer. Una vez se acepta un SLA, el SLA-Backend envía su trabajo asociado al SA-Layer, el cual se encarga de su ejecución. El SA-Layer acepta el trabajo, el tiempo de comienzo y su deadline como parámetros, y se encargará de la ejecución únicamente en estos términos. Las acciones que involucran otros términos son tomadas por el SLA-Backend. En cada emisión, el SA-Layer comprueba los recursos asociados al Grid y envía el trabajo a uno de ellos, tomando en cuenta el cumplimiento de las restricciones temporales [16]. Por lo tanto, el SLA-Backend toma las decisiones de alto nivel mientras que el SA-Layer se encarga de los trabajos en términos de planificación temporal.

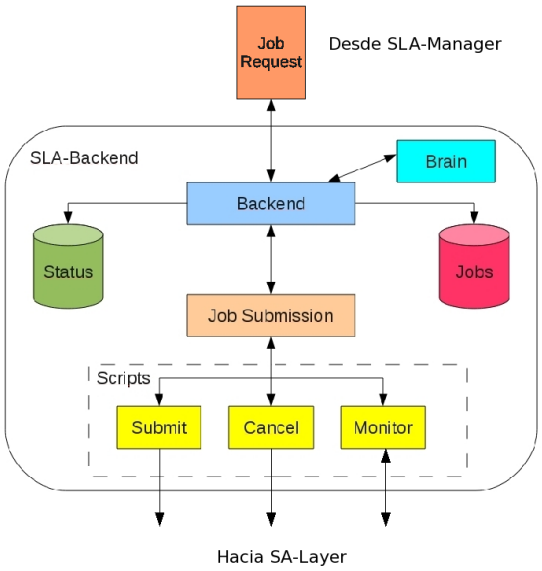


Fig. 2. Estructura interna del SLA-Backend.

Internamente, el SLA-Backend está compuesto por tres bloques principales, dos bases de datos y tres scripts (Ver Figura 2). El módulo *Backend* hace de interfaz entre el SLA-Manager y los módulos *Brain* y *Job Submission*. El módulo *Brain* implementa las políticas, y por lo tanto, el *Backend* comprueba este módulo para tomar las decisiones. Y el módulo *Job Submission* es el encargado de manejar los trabajos una vez son aceptados mediante los *scripts* (*Submit*, *Cancel* y *Monitor*), los cuales interaccionan con el SA-Layer, y tomar las medidas adecuadas en cuanto a asegurar la QoS de dichos trabajos.

## IV. Niveles de QoS

La propuesta presentada en este artículo para la evaluación de la jerarquización de QoS consiste en un esquema de tres niveles basado en términos de requisitos temporales para la arquitectura detallados en la sección anterior. Estos requisitos son: *start-time* ($t_S$), *deadline* ($t_D$) y nivel de calidad de servicio ($QoS$). Los términos $t_S$ y $t_D$ se utilizan para especificar las restricciones temporales de un trabajo. Es decir, representan la ventana de tiempo deseada para la ejecución de cada trabajo. El término $QoS$ se utiliza para especificar el nivel de QoS esperada, que puede ser: débil, moderada o fuerte.

Los valores de estos parámetros para la propuesta de tres niveles de QoS son:

$Q_1 \rightarrow \; t_S = 0 \quad t_D = \infty \quad QoS = Débil$

$Q_2 \rightarrow \; t_S = t_{1Inicio} \quad t_D = t_{1Fin} \quad QoS = Moderada$

$Q_3 \rightarrow \; t_S = t_{2Inicio} \quad t_D = t_{2Fin} \quad QoS = Fuerte$

El nivel de QoS $Q_1$ corresponde al más bajo, el cual se mapea con un servicio de tipo *best effort*. Los trabajos asociados a este nivel de QoS no tienen restricciones temporales: el *start-time* es cero, lo que significa que será ejecutado tan pronto como sea posible sin importar cuando, y el *deadline* para este servicio es infinito. Además, los trabajos asociados a este nivel de QoS serán planificados únicamente en recursos de baja confiabilidad, como son aquellos en los cuales no se pueden hacer reservas físicas y el sistema estima que pueden no cumplir con las expectativas. Para los niveles de QoS $Q_2$ y $Q_3$, el usuario especifica el *start-time* y *deadline*, pero la diferenciación entre ambos se realiza mediante el término $QoS$. Este término tiene formato de etiqueta y puede tratarse de moderado o fuerte, dependiendo de las necesidades de los usuarios. La decision de este parámetro depende en la exigencia esperada de la ejecución del trabajo: el nivel moderado tiene en cuenta el servicio, pero no asegura la disponibilidad de los recursos en los que la ejecución se planifica; por el contrario, el nivel fuerte asegura la disponibilidad del recurso utilizando únicamente recursos con una alta disponibilidad asociados al Grid con mecanismos para reservar recursos físicamente (como Torque [17]). Por lo tanto, para el nivel de QoS fuerte, solo se utilizan recursos computacionales con capacidad para reserva por adelantado. Mientras que para el nivel de QoS moderado, se utilizan todos los recursos del Grid (pero sin aplicar reserva física en los recursos computacionales que disponen de ella).

Es importante mencionar que si la ejecución no puede ser planificada cumpliendo los requisitos temporales especificados una vez el SLA ha sido aceptado o se excede el *deadline*, se considera que el SLA ha sido violado.

## V. Detalles de implementación

En las siguientes dos subsecciones se describen los cambios necesarios para ajustar el comportamiento de todo el sistema para tres niveles de QoS, y cumplir así con lo descrito en las secciones anteriores.

### A. Plantillas de negociación

La estructura y parámetros de un acuerdo se especifican en las plantillas de negociación [4]. Dado que la especificación WS-Agreement es flexible en cuanto a los parámetros de servicio, su diseño es tarea del proveedor de servicios, y por lo tanto, es el encargado de definir explícitamente los términos y limitaciones de los acuerdos. Una vez están cumplimentados por el usuario, se consideran SLAs.

Para nuestra propuesta, las plantillas cumplen con la especificacioón WS-Agreement estructuralmente pero requieren de parámetros de servicio adicionales con el objetivo de caracterizarlos en tres niveles de QoS, definidos previamente. El usuario debe establecer el valor de estos parámetros para el *start-time*, *deadline* y QoS deseados. Además, para consolidar el sistema, estos parámetros son obligatorios y deben aparecer en todos los SLAs, y por lo tanto, un SLA se considerará mal formado y será rechazado si carece de ellos.

Sin embargo, con el objetivo de disponer de un sistema más descriptivo, se propone utilizar etiquetas para el nivel de QoS en lugar de débil, moderado o fuerte. Y para ello, y tratandose de tres niveles, hemos optado por la analogía con las medallas deportivas. Estas etiquetas son: **Bronce**, **Plata** y **Oro** respectivamente.

### B. Modificaciones de la arquitectura

El objetivo principal de esta propuesta es conseguir diferenciar la QoS en diferentes niveles de forma jerarquizada mediante requisitos temporales y un parámetro de confianza. Por lo tanto, es necesario tener esto en cuenta en la arquitectura (Ver Figura 1). Sin embargo, la plantilla y la estructura de los SLAs no son los únicos donde es necesario aplicar modificaciones. Cuando se definen nuevos parámetros en los SLAs, la capa SLA-Backend necesita ser adaptada mediante la implementación de una política capaz de permitir la diferenciación de QoS y actuar en consecuencia con ella.

Los recursos asociados al Grid pueden ser heterogéneos, pero su tipo, atendiendo al sistema de gestión de recursos local (*Local Resource Management System* (LRMS)) puede ser determinado por GridWay [18]. El SLA-Backend y SA-Layer utilizan esta información para establecer la confianza de cada recurso. Aquellos que no disponen de mecanismos para realizar reservas físicas son considerados de *baja confiabilidad* dado que no pueden asegurar potencia computacional ni memoria principal libre. Por otro lado, aquellos con soporte para reservas físicas son considerados de *alta confiabilidad* porque pueden garantizar su disponibilidad cuando sea necesario, así como espacio de almacenamiento y memoria principal.

Determinada la confiabilidad de los recursos, el SLA-Backend puede elegir un grupo de recursos sobre los que la ejecución es preferente. Cuando el SLA-Manager recibe un SLA, delega la decisión de aceptación o rechazo al SLA-Backend. Si se considera que los términos de un SLA pueden ser satisfechos (comprobando si es posible planificar el trabajo en, al menos, uno del grupo de recursos a través del SA-Layer), es aceptado, y entonces delegado al SA-Layer para ser gestionado.

El SLA-Backend selecciona el conjunto de recursos asociado, es decir, los recursos con capacidad para reservas para los SLAs con QoS de Oro, y los demás recursos para de Plata y/o Bronce.

Por otro lado, GT4 no provee soporte para reservas por adelantado, pero permite la emisión de trabajos a otro LRMS que si lo provee, como Torque [17]. Con una pequeña modificación de GT4 es posible añadir otro parámetro al comando de emisión del trabajo, y asignarlo a una reserva física creada desde el SLA-Backend.

## VI. Resultados experimentales

La evaluación y resultados obtenidos de la arquitectura descrita en las secciones anteriores se muestra en la presente sección. Para ello, se somete a pruebas reales enfrentando el sistema a las peticiones de usuarios.

Los trabajos solicitados por los usuarios del sistema, y asociados a cada SLA (independientemente del nivel de QoS esperado), es el *benchmark 3node* [19]. Esta elección se debe a su capacidad para generar carga de red, de cómputo y de almacenamiento de forma controlada mediante parámetros definidos previa ejecución.

Para esta evaluación, los parámetros establecidos en el *benchmark 3node* son: ficheros de origen de 10 Mb (requisito de transmisión) y un parámetro de procesamiento aleatorio entre 10 y 100. La razón de estos parámetros es la de modelar la evaluación frente a un conjunto de usuarios capaces de enviar un amplio rango de trabajos con características muy diferentes entre sí.

La evaluación se ha realizado sobre un Grid real, compuesto por cuatro ordenadores personales y un *cluster*. Los ordenadores personales disponen de una CPU AMD Opteron a 1.8 Ghz, 1 Gb de RAM, 80 Gb de almacenamiento y el sistema operativo Debian 4.0. El *cluster* utilizado dispone de 20 nodos de cómputo, pero 15 de ellos están dedicados a otros propósitos. Los disponibles poseen CPUs AMD Opteron Dual-Core a 3 Ghz, con 8 Gb de RAM, 150 Gb de almacenamiento y el sistema operativo CentOs 5.

En primer lugar, se diseña una prueba para simular el comportamiento del sistema frente a usuarios (procesos remotos). La frecuencia de solicitudes realizadas por los usuarios se establece en 3 cada 2 minutos, un poco menos que la productividad máxima alcanzada por el Grid. Y la distribución de SLAs, clasificadas por requisitos de QoS, se describe en la Tabla I. Finalmente, las restricciones temporales se establecen de forma aleatoria, entre los 2 y 4 minutos para el *start-time*, y entre los 5 y 10 minutos para el *deadline*.

Una vez realizadas las pruebas, se observa que el

TABLA I
DISTRIBUCIÓN APROXIMADA DE SLAS POR REQ. DE QOS.

| Escenario | Oro | Plata | Bronce |
|---|---|---|---|
| A | ∼20% | ∼40% | ∼40% |
| B | ∼20% | ∼50% | ∼30% |
| C | ∼20% | ∼60% | ∼20% |
| D | ∼20% | ∼70% | ∼10% |

ratio de aceptación (ver Figura 3) se incrementa paralelamente con el porcentaje de emisiones por tipo de QoS en cada escenario. Pero el ratio de rechazo (ver Figura 4) es más elevado en cada escenario, comenzando con un nivel de rechazo muy bajo en el escenario A, que aumenta en los escenarios B y C, y se observa un nivel de rechazo moderado en el escenario D. Debido a este comportamiento, la cantidad global de trabajos procesados se ve afectado y disminuye entre escenarios. Estas tendencias son las esperadas, debido al hecho de que en cada escenario, el porcentaje de SLAs con requisitos temporales es mayor (SLAs con QoS de Oro y Plata). Por lo tanto, el sistema no puede aceptar más SLAs sin comprometer los que ya han sido aceptados.

Es importante notar que el rechazo solo se presenta en SLAs de Oro o Plata. Esto se debe a que las peticiones de servicio de Bronce pueden ser procesadas cuando sea posible. Por lo tanto, el sistema acepta todos los SLAs de Bronce y los encola hasta que el sistema detecta un recurso libre y tiempo para procesarlos.

En cuanto al porcentaje de violaciones de los términos y procesamiento satisfactorio de los SLAs establecidos, los experimentos muestran que, en todos los escenarios, todos los SLAs con QoS de Oro aceptados fueron procesados satisfactoriamente (ninguno de ellos fue violado) (ver Figura 5). Además, en el escenario A, todos los SLAs con QoS de Plata fueron procesados satisfactoriamente también. Pero en el escenario B, comienzan a aparecer violaciones en un 6.08% de los SLAs con QoS de Plata aceptados, y se incrementa en el escenario C al 8.15%. Finalmente, en el escenario D, se muestra la peor condición para los SLAs con QoS de Plata dado que presenta el mayor nivel de violaciones con un 13.18% de los aceptados.

Con estos resultados, se puede concluir que los SLAs de Plata son penalizados frente a los SLAs de Oro, lo cuál es el comportamiento esperado del sistema de prioridades.

El comportamiento del sistema (ver Figura 6) como resultado de las pruebas definidas anteriormente muestra un decremento global en el nivel de aceptación conforme aumenta el porcentaje de SLAs con restricciones temporales (SLAs de Oro y Plata). Pero también es interesante observar que el porcentaje de violaciones aumenta en paralelo desde el escenario A al D. Por lo tanto, el sistema procesa menos trabajos de forma satisfactoria en el escenario D que en el A. Además, se producen más violaciones

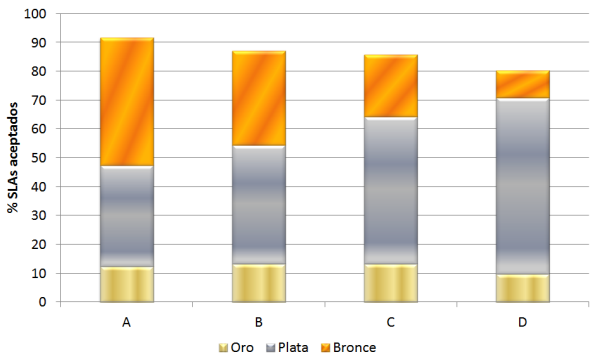


Fig. 3. Aceptación de SLAs.

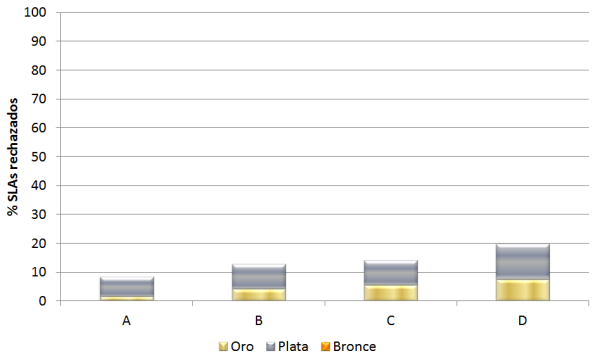


Fig. 4. Rechazo de SLAs.

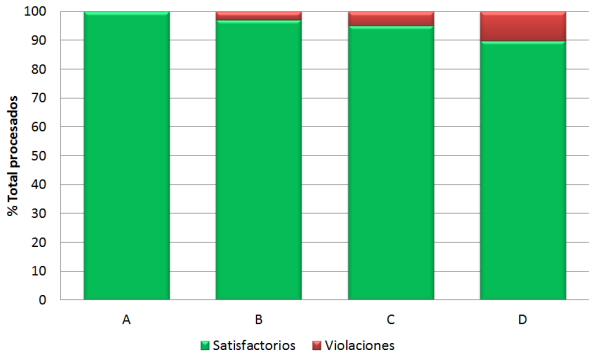


Fig. 5. Resultado de los SLAs.

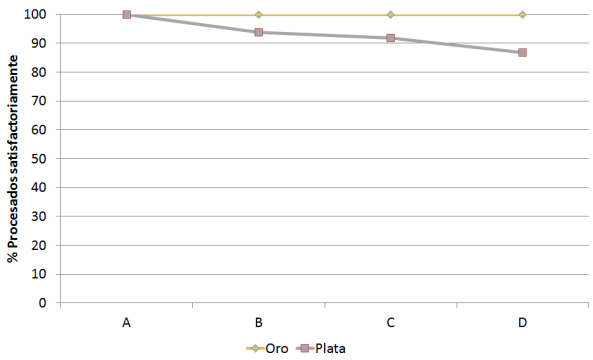


Fig. 6. QoS conseguida por nivel.

de QoS conforme se aumenta la cantidad de SLAs con restricciones temporales.

El resultado obtenido de los resultados experimentales muestran que la arquitectura propuesta se

adapta a las necesidades de los usuarios dependiendo de la confianza que ellos esperan de la ejecución de sus trabajos. Esto refuerza la necesidad de disponer múltiples niveles de QoS soportado por SLAs en los Grids.

## VII. Conclusiones y Trabajo Futuro

En el presente artículo se desarrolla y evalúa una extensión de la arquitectura propuesta antoriormente por los autores en [14] en un Grid real. Se propone una diferenciación de QoS organizada en tres niveles y se muestran los mecanismos que permiten la evaluación y gestión de los SLAs en función de cada nivel de QoS.

Esta propuesta se basa en la confiabilidad de los recursos, mediante una arquitectura software para ofrecer las garantías establecidas a través de los SLAs.

Los resultados experimentales obtenidos muestran que el concepto de tres niveles de QoS basados en la confiabilidad de los recursos es factible, consiguiéndose la mayor QoS solicitada a través de un SLA mediante la utilización de la técnicas de planificación por adelantado y reserva física por adelantado simultáneamente. La QoS moderada se consigue mediante la utilización de planificación por adelantado (sin reserva física), y dejando el tiempo ocioso de los recursos para peticiones de servicio de más baja QoS, en las que no se utiliza ninguna de las técnicas anteriores. Además, se estudian los niveles de aceptación y rechazo, así como los niveles de violaciones de los términos establecidos en los SLAs, destacando la importancia de la diferenciación de la QoS en niveles.

Algunos de los trabajos futuros son el diseño e implementación de nuevos algoritmos para la evaluación de la confianza en cada recurso o incluso predicción de la confianza, incluyendo técnicas de replanificación capaces de mover (temporalmente o incluso entre recursos) los trabajos asociados a SLAs en los que se ha acordado una baja QoS para aceptar otros trabajos con mayores requisitos. Finalmente, otro problema que puede afrontarse es la mejora del SLA-Manager para optimizar las decisiones de forma más inteligente y/o eficiente, tomando en cuenta los costes asociados o la energía consumida.

## Agradecimientos

Este trabajo ha sido soportado conjuntamente por el MEC y MICINN Español y la Comisión Europea (fondos FEDER) a través de los proyectos "Consolider Ingenio-2010 CSD2006-00046", "TIN2009-14475-C04" y a través de una beca FPI asociada al proyecto "TIN2009-14475-C04-03". Conjuntamente con la JCCM a través del proyecto "PII1C09-0101-9476".

## Referencias

[1] Django Armstrong and Karim Djemame, "Towards Quality of Service in the Cloud," in *Proc. of the 25th UK Performance Engineering Workshop*, Leeds, UK., 2009.

[2] Vladimir Stantchev and Christian Schröpfer, "Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing," in *Proc. of the 4th Intl. Conference on Advances in Grid and Pervasive Computing (GPC)*, Geneva, Switzerland, 2009.

[3] James Padgett, Karim Djemame, and Peter Dew, "Grid-Based SLA Management," in *Proc. of the European Grid Conference (EGC)*, Amsterdam, The Netherlands, 2005.

[4] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu, "Web Services Agreement Specification (WS-Agreement) GFD-R-P.192," Tech. Rep., October 2011, https://forge.gridforum.org/projects/graap-wg/.

[5] James Padgett, Karim Djemame, and Peter Dew, "Predictive Adaptation for Service Level Agreements on the Grid," *Intl. Journal of Simulation Systems, Science and Technology*, vol. 7, no. 2, pp. 29–42, 2006.

[6] Ivona Brandic, Dejan Music, Schahram Dustdar, Srikumar Venugopal, and Rajkumar Buyya, "Advanced QoS Methods for Grid Workflows Based on Meta-Negotiations and SLA-Mappings," in *Proc. of the 3rd Workshop on Workflows in Support of Large-Scale Science. In conjunction with Supercomputing*, Austin, USA, 2008.

[7] Jorge Ejarque, Marc de Palol, Ínigo Goiri, Ferran Julià, Jordi Guitart, Rosa M. Badia, and Jordi Torres, "Exploiting semantics and virtualization for SLA-driven resource allocation in service providers," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 5, pp. 541–572, 2010.

[8] Oliver Waeldrich, Dominic Battrè, Francis Brazier, Kassidy Clark, Michel Oey, Alexander Papaspyrou, Philipp Wieder, and Wolfgag Ziegler, "WS-Agreement Negotiation Version 1.0 GFD-R-P.193," Tech. Rep., October 2011, https://forge.gridforum.org/projects/graap-wg/.

[9] "WSLA: Web Service Level Agreements," Web page at http://www.research.ibm.com/wsla/, Último acceso: 26 Marzo, 2012.

[10] D. Davide Lamanna, James Skene, and Wolfgang Emmerich, "SLAng: A Language for Defining Service Level Agreements," in *Proc. of the Intl. Workshop of Future Trends of Distributed Computing Systems*, San Juan, Puerto Rico, 2003.

[11] "AssessGrid," Web page at http://www.assessgrid.eu, Último acceso: 24 Febrero, 2012.

[12] "SLA at SOI," Web page at http://sla-at-soi.eu/, Último acceso: 26 Marzo, 2012.

[13] Hui Li, Wolfgang Theilmann, and Jens Happe, "Challenges in SLA Translation," *SLA@SOI Technical Paper*, Diciembre 2009, http://sla-at-soi.eu/results/publications/.

[14] Javier Conejero, Carmen Carrión, and Blanca Caminero, "A SLA-based Framework with Support for Meta-scheduling in Advance for Grids," in *Intl. Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, Las Vegas, USA, Julio 2011.

[15] Luis Tomás, Agustín C. Caminero, Carmen Carrión, and Blanca Caminero, "Network-aware meta-scheduling in advance with autonomous self-tuning system," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 486 – 497, 2011.

[16] Luis Tomás, Agustín Caminero, Carmen Carrión, and Blanca Caminero, "Meta-Scheduling in Advance using Red-Black Trees in Heterogeneous Grids," in *Proc. of the High Performance Grid Computing Workshop (HPGC), hold jointly with the Intl. Parallel & Distributed Processing Symposium (IPDPS)*, Atlanta, USA, 2010.

[17] "Torque Resource Manager," Web page at http://www.adaptivecomputing.com/products/torque.php, Último acceso: 27 Febrero, 2012.

[18] Constantino Vázquez, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente, "Federation of TeraGrid, EGEE and OSG infrastructures through a metascheduler," *Future Generation Computing Systems*, vol. 26, pp. 979–985, Julio 2010.

[19] Greg Chun, Holly Dail, Henri Casanova, and Allan Snavely, "Benchmark probes for grid assessment," in *Proc. of the Intl. Parallel & Distributed Processing Symposium (IPDPS)*, Santa Fe, USA, 2004.