GridUI, a RESTful Web Service and Ubiquitous UI for Accessing HPC Resources

J.E. Serrano
Facultad de Ingeniería
Universidad Tecnológica de Bolívar
130010, Cartagena, Colombia
jairo@utbvirtual.edu.co

J. Leguizamón
Facultad de Ingeniería
Universidad Tecnológica de Bolívar
130010, Cartagena, Colombia
juan@utbvirtual.edu.co

J.C. Martínez-Santos
Facultad de Ingeniería
Universidad Tecnológica de Bolívar
130010, Cartagena, Colombia
jcmartinezs@unitecnologica.edu.co

ABSTRACT

This paper describes the design of a user interface between a high-throughput computing pool system like HTCondor and a device-independent user interface. This design aims to bring parallel tools and resources to a novel academic community. No matters how and where are resources allocated, users can be access from own their hands with GridUI.

Categories and Subject Descriptors

H.1.2 [**User/Machine Systems**]: Human factors; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*User interfaces*; H.3.5 [**Information storage and retrieval**]: Online Information Services—*Web-based services*

Keywords

Grid Computing, RESTful, Web Service, Server-Client, HTCondor

1. INTRODUCCIÓN

En toda comunidad académica que haga uso de recursos de computación avanzada, existen al menos tres tipos de usuarios, los usuarios avanzados, los intermedios y el usuario novel. Los usuarios avanzados son quienes programan aplicaciones usando librerías y componentes propios o de terceros. Estos usuarios también en la mayoría de los casos se hacen responsables de seleccionar un gestor de colas de trabajo y velan por la correcta ejecución de sus tareas. Existe también el usuario intermedio, quien esta consciente de todo el esfuerzo necesario y se asocia con un equipo de trabajo, repartiendo tareas y obligaciones, usa librerías y aplicaciones que su equipo de trabajo construye o sugiere. Por último esta el usuario novel, un usuario enfocado en aprender. Es un tipo de usuario que está en etapa de acercamiento a la tecnología, posiblemente tomando clases [8] o iniciando proyectos de investigación que tienen un fuerte componente de procesamiento de datos y requieren el uso de computación en paralelo o distribuida. Una característica de la mayoría de herramientas que procesan gran cantidad de información en paralelo es que las interfaces de usuario son en modo texto o consola, y no disponen de interfaz gráfica agradable o enfocada al usuario. El diseño de nuestra aplicación esta enfocado a cubrir las necesidades de estos usuarios más noveles, brindado el acceso a las herramientas desde interfaz de usuario fresca, ágil y accesible desde computadores o móviles (ver Figura 1) y que además pueda ser desplegada en los servidores locales o en la nube sin problemas de licenciamiento.

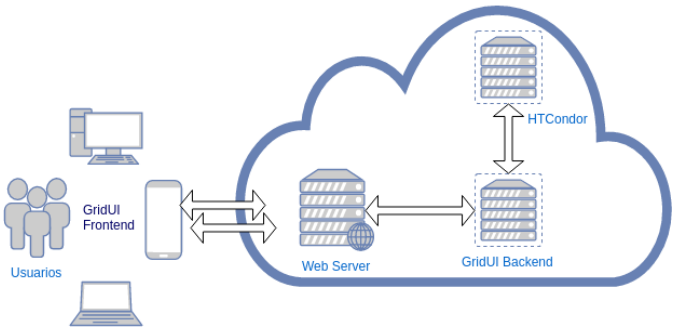

Figure 1: Esquema de acceso a GridUI, una forma fácil de acceder a servicios de computación en paralelo independiente del dispositivo de acceso y experiencia del usuario.

Este documento está organizado de la siguiente manera. Sección 2 contiene el estado del arte. Sección 3 presenta el diseño de nuestra aplicación, Sección 4 describe los detalles de la implementación de nuestra aplicación. Sección 5 muestra el estado actual de desarrollo y finalmente, Sección 6 presenta las conclusiones y expresa el trabajo futuro.

2. ESTADO DEL ARTE

Esta no es la primera vez que se abordan este tipo de aplicaciones. Nosotros nos referiremos a las aplicaciones de código abierto, las cuales nos permiten hacer una comparación mas justa entre lo que se propone y los que ya esta desarrollado.

Un buen ejemplo son las primeras aproximaciones para crear un servicio web directamente sobre HTCondor [5] y un cliente desarrollado en lenguaje perl "*Condor Grid Computing from Mobile Handheld Devices*" [2]. El problemas de esta solución es su interfaz HTML para envío de trabajos al

2016 8th Euro American Conference on Telematics and Information Systems (EATIS), Cartagena de Indias - Colombia. April 27 - 29
978-1-5090-2435-3/16/$31.00 ©2016 IEEE

gestor de colas. Aunque es posible acceder a ella desde un dispositivo que entienda HTML, no hace uso de las nuevas herramientas y lenguajes para la web.

Otros proyectos, que nacen después, son YABI[4] "*Bringing Scientific Workflow to the Masses via Pegasus and HUBzero*" [7]. Ambos proyecto brindan a usuario expertos y novatos un gran conjunto de herramientas pre-configuradas con parámetros de entrada pre-diseñados y salidas esperadas. YABI dispone de su *frontend* (interfaz en html) que está fuertemente atada al *backend* (plataforma en Python). Mientras que Pegasus hace uso de HTML y HUBZero, que es una mezcla de HTML y una visualización directa de las herramientas mediante el protocolo VNC. Esto una desventaja si se tiene la necesidad de disponer de múltiples clientes y múltiples dispositivos de acceso.

Otro trabajo previo muy interesante es "*Personal Cloudlets: Implementing a User-centric Datastore with Privacy Aware Access Control for Cloud-based Data Platforms*" [6]. Aquí, los autores hacen referencia a un diseño basado en un servicio RESTful para administrar información de manera segura en diversos tipos de arquitecturas en la nube. Los servicios de autenticación, acceso a la información, y comunicación interactúan de forma transparente creando un único punto final de acceso para que los clientes usando JSON se pueden desarrollar aplicaciones consumidoras de estos datos. Sin importar que tipo de aplicación haga uso de ellos, se puede asegurar autenticación, acceso a los recursos, acceso transparente, y casi ubicuo.

3. DISEÑO

Nuestra propuesta es tomar lo mejor de cada una de las propuesta anteriores1. Igual que los trabajos mostrados, nosotros hacemos uso de herramientas abiertas. Similar a [2], nosotros usamos HTCondor como nuestro sistema de gestión de carga de trabajo. Así mismo, tomamos con referencia el modelo de gestión en YABI y el modelo de infraestructura de Pegasus. Por último, para la administración y el despliegue de los servicios web, nosotros usamos RESTful.

Por lo general los usuarios crean proyectos donde se agrupa la ejecución de tareas una a una, una tarea constituye un paquete de datos que es procesado por una herramienta generando una salida y almacenándola para realizar el análisis de parte del investigador, en determinados casos será necesario el uso de un workflow (flujo de trabajo) (Yu y Buyya [10]) para conectar múltiples tareas que usen los resultados por otra tarea previa así generando nuevos resultados, según las dependencias que se establezcan entre los trabajos pueden ser representados en dos formas: DAG (Directed Acyclic Graph) y non-DAG. En un workflow de estructura DAG, existen tres tipos de subcategorias que son de secuencia, paralelismo y de elección. Como diferencia entre DAG y non-DAG, solo aparece una subcategoria adicional en los de estructura non-DAG llamada de iteración, GridUI estará, al menos en esta etapa de desarrollo a crear y ejecutar workflow DAG secuenciales.

Una de las principales consideraciones para este proyecto, es que el diseño debería ser modular. Los componentes del sistema son: la interfaz de usuario (frontend - UI), el *middleware*, el *backend*, el sistema de autenticación (Auth), el gestor de archivos (Storage) y el gestor de ejecución de tareas (HTCondor). Todos son componentes están separados, son interoperables y reemplazables, por ejemplo, si la interfaz de usuario construida en HTML y AngularJS quisiera ser reemplazada por una en Android nativo, esto es viable y posible, gracias a que está construido en base a una arquitectura RESTful.

4. IMPLEMENTACIÓN

Para la implementación se hizo uso intensivo de:

- HTML 5 y CSS 3 para implementar la interfaz gráfica y servicio web.
- AngularJS para la funcionalidad de lado del cliente (frontend).
- JWT [9] para la autenticación basada en tokens en el cliente.
- Django + REST framework para el backend, manejo de datos.
- HTCondor python API, para la generación de tareas.

4.1 Mecanismo de Autenticación

Al construir un servicio RESTful se debe tener presente que la velocidad obtenida es brindada porque funciona sobre un el protocolo TCP *stateless* (sin estado) [3]. En nuestra aplicación no se tiene un mecanismo de autorización tradicional como *cookies* o sesiones en HTTP. En cambio fue necesario implementar un sistema de autenticación mediante claves de acceso *JSON Web Tokens* (JWT). Con esto, y haciendo uso de un canal seguro HTTPS se logra un buen margen de confiabilidad (confidencialidad e integridad) en la comunicación entre los diversos tipos de clientes y el servicio de autenticación, al cual hemos llamado *GridUI*. La Figura 2 muestra el esquema de usado.

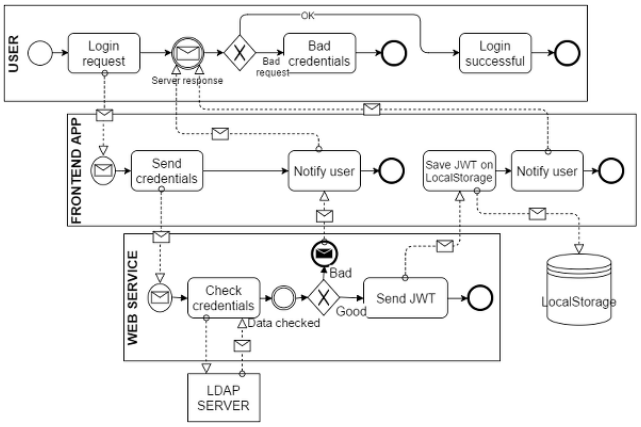

Figure 2: Esquema de autenticación, usando JWT en el cliente y un servidor LDAP en el backend.

4.2 Administración de Herramientas

El sistema está diseñado para que el flujo de trabajo de un usuario esté compuesto por la interacción entre las diferentes herramientas. Las herramientas son el software puesto a disposición de los usuarios por parte de los administradores del sistema para ser usado según sea requerido.

Los administradores del sistema deben poner a prueba la estabilidad y el rendimiento del software antes de agregarlo al set de herramientas del portal. Una vez la herramienta se

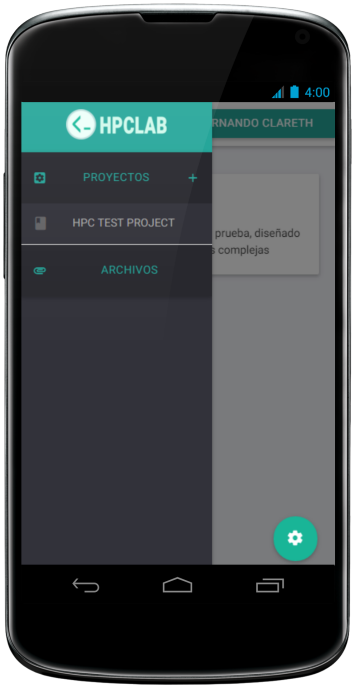

Figure 3: Listado de proyectos por usuario, se comparte la interfaz con al administrador de archivos unificado para los proyectos.

encuentra lista para su uso, el administrador crea un nuevo registro que contiene el nombre y una documentación para su correcto uso. A partir de ahí se deben cargar los archivos necesarios para el funcionamiento de la herramienta y agregar los parámetros que permitan al usuario flexibilidad en el uso de esta en caso de ser necesario.

Actualmente sobre la infraestructura se encuentra funcionando el *framework* de HTCondor. Este permite al usuario la creación de *workflows* con la herramienta HTCondor DAGMan. El servicio crea las tareas mediante la API en Python que el framework tiene disponible.

Dicho de esta manera el proyecto se enfoca en construcción y ejecución de *workflows* de estructura DAGMan. Actualmente se soportan *workflows* de tipo secuencia.

Al construir un nuevo workflow, el usuario obtiene un espacio de trabajo (ver Figura) en el cual puede agregar las herramientas que cumplen con sus necesidades y establecer un orden que permita la interacción entre ellas mediante dependencias.

En un principio el usuario debe agregar los archivos que van a ser procesados durante la ejecución del *workflow*. Luego, el usuario elige las herramientas que va a utilizar, establece los parámetros de ejecución, los archivos de entrada, y las dependencias en cada una de ellas de manera correcta para seguir con el envío a ejecución de este trabajo. Una vez terminada la tarea, se envía una notificación al correo electrónico al usuario.

4.3 Administración Archivos

Cada usuario tiene un espacio de almacenamiento (ver Figura 4), dicho espacio es compartido entre los proyectos, estos datos son usados por las herramientas, procesándose y generando archivos resultantes. Una consideración de diseño para optimizar el espacio ocupado en disco duro de los servidores es mantener internamente indexados todos los datos mediante un hash único por archivo, esto impide que se almacenen archivos duplicados, optimizando los recursos.

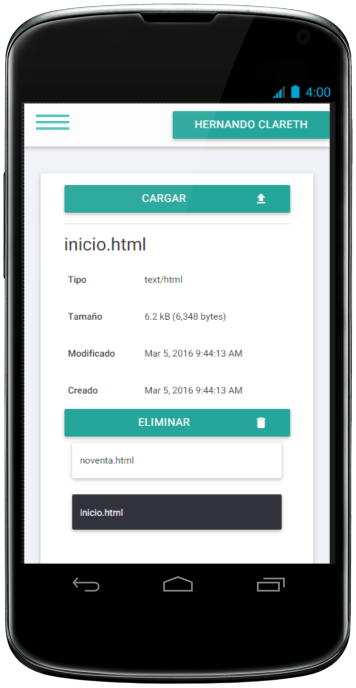

Figure 4: Visualización de archivos y resultados.

5. ESTADO ACTUAL

Despues de varios meses dedicados al diseño y construcción de la herramienta se tienen avances significativos, se cuenta en la actualidad con un prototipo funcional capaz de ejecutar tareas (ver figura 5) y recibir resultados generados por las herramientas configuradas según el proyecto y el workflow de procesamiento.

La integración con HTCondor se da generando los archivos de tareas genéricos organizados dentro de los proyectos o según el flujo de la información a procesar, básicamente se generan los archivos de trabajo o tarea", estos archivos tienen la siguiente estructura:

```
executable      = herramienta
universe        = vanilla
input           = test.data
output          = test.out
error           = test.error
log             = test.log
queue
```

Y para los flujos de trabajo o la integración de varias tareas o las dependencias entre tareas:

```
job  A  A.condor
job  B  B.condor
job  C  C.condor
job  D  D.condor
parent A CHILD B C
parent B C CHILD D
```

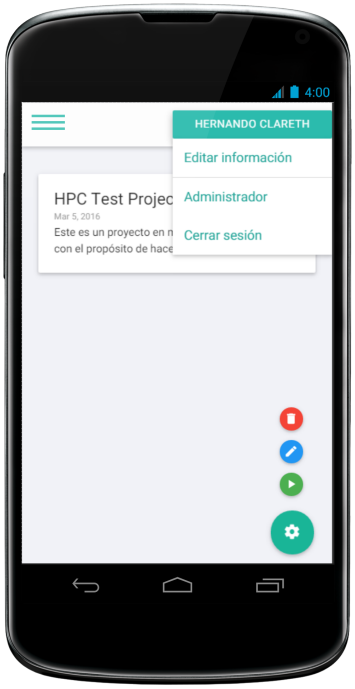

Figure 5: Vista proyecto, es posible desde los proyectos agendar la ejecución de tareas o un flujo de trabajo entre estas.

Sin GridUI el usuario novato debería entrar a la interfaz de terminal en el Cluster, editar estos archivos manualmente y estar atento a cuando las tareas terminan, ahora todo este proceso se automatiza e inclusive se sabe cuando terminan las tareas con la ayuda de las notificaciones que llegan al correo electrónico registrado por los usuarios.

6. CONCLUSIONES Y TRABAJO FUTURO

El proyecto a su vez, está disponible para su descarga e instalación desde el repositorio en GitHub[1], se espera que para finales de marzo del 2016 se tenga una versión de prueba disponible al publico interesado.

El proyecto a su vez, está disponible para su descarga e instalación desde el repositorio en GitHub[1], se espera que para Marzo del 2016 se tenga una versión de prueba disponible al publico interesado.

Como trabajo futuro se espera continuar con el desarrollo y ajuste a la interfaz móvil de GridUI, contar con mayor grado de interacción y configuración de las herramientas.

En la actualidad solo está soportado los flujos de trabajo secuenciales, se espera realizar la implementación de jobs en paralelo usando las características implementadas HTCondor.

Agradecimientos

Se debe hacer un especial reconocimiento al programa de jóvenes investigadores del Departamento Administrativo de Ciencia, Tecnología e Innovación - COLCIENCIAS, gracias al cual se logró financiar la vinculación de un estudiante de maestría al equipo de investigación y desarrollo (Proyecto - Interfaz web para acceso a recursos de supercomputación disponibles en la red nacional académica de tecnología avanzada - Renata y computación avanzada 3CoA, Convocatoria Joven Investigador 0645-2014 COLCIENCIAS), Hernando Ariza, Angel Jimenez, Carlos Buelvas, Jorge Franco estudiantes de Ingeniería de Sistemas en prácticas investigativas en el Laboratorio de Computación de alto desempeño en la Universidad Tecnológica de Bolívar.

7. REFERENCES

[1] GitHub. Ingenieriadesistemasutb, 2015.

[2] F.˜J. González-Castaño, J.˜Vales-Alonso, M.˜Livny, E.˜Costa-Montenegro, and L.˜Anido-Rifón. Condor grid computing from mobile handheld devices. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(2):18–27, Apr. 2002.

[3] D.˜A. Hayes, M.˜Welzl, G.˜Armitage, and M.˜Rossi. Improving http performance using "stateless" tcp. In *Proceedings of the 21st International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '11, pages 57–62, New York, NY, USA, 2011. ACM.

[4] A.˜A. Hunter, A.˜B. Macgregor, T.˜O. Szabo, C.˜A. Wellington, and M.˜I. Bellgard. Yabi: An online research environment for grid, high performance and cloud computing. *Source Code for Biology and Medicine*, 7(1):1–10, 2012.

[5] M.˜Litzkow, M.˜Livny, and M.˜Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[6] D.˜McCarthy, P.˜Malone, J.˜Hange, K.˜Doyle, E.˜Robson, D.˜Conway, S.˜Ivanov, L.˜Radziwonowicz, R.˜Kleinfeld, T.˜Michalareas, T.˜Kastrinogiannis, N.˜Stasinos, and F.˜Lampathaki. Personal cloudlets: Implementing a user-centric datastore with privacy aware access control for cloud-based data platforms. In *Proceedings of the First International Workshop on TEchnical and LEgal Aspects of Data pRIvacy*, TELERISE '15, pages 38–43, Piscataway, NJ, USA, 2015. IEEE Press.

[7] M.˜McLennan, S.˜M. Clark, F.˜McKenna, E.˜Deelman, M.˜Rynge, K.˜Vahi, D.˜Kearney, and C.˜X. Song. Bringing scientific workflow to the masses via pegasus and hubzero. In T.˜Kiss, editor, *Proceedings of the 5th International Workshop on Science Gateways, Zurich, Switzerland, 3-5 June, 2013*, volume 993 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[8] G.˜Tel-Zur. Pdc education in the bgu ece department. In *Proceedings of the Workshop on Education for High-Performance Computing*, EduHPC '14, pages 15–20, Piscataway, NJ, USA, 2014. IEEE Press.

[9] Tools.ietf.org. Rfc 7519 - json web token (jwt), 2015.

[10] J.˜Yu and R.˜Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3):44–49, Sept. 2005.