## 6. Conclusiones

En este trabajo de tesis se presentó un framework capaz de abstraer los diferentes servicios que se brindan sobre Cloud y soportar migraciones entre diversos proveedores de servicios y herramientas. Ambos ejes fueron tratados en pos de hacer frente a los problemas con que se encuentran las empresas al momento de decidir un cambio de proveedor de servicios para sus aplicaciones.

En este capítulo se describirán las conclusiones extraídas luego de experimentar los resultados del uso de la herramienta.

Al comienzo del documento se analizaron los puntos fuertes y débiles de cada una de las herramientas actuales. El resultado de este análisis puso en evidencia una serie de problemas comunes a todas las herramientas que se relacionan fundamentalmente con las capacidades de migración. Por un lado se distinguió la migración de aplicaciones entre servicios, por ejemplo de Amazon S3 a Google Storage. Este tipo de migración suele ser tenida en cuenta por los diseñadores de las herramientas para cloud computing, pero no se encuentra disponible un mecanismo que realmente facilite la tarea al usuario ya que en muchos casos debe volver a codificar para adaptarse a los objetos del nuevo servicio. Por otro lado se analizó la migración de aplicaciones entre herramientas o frameworks para cloud computing. Un ejemplo de esto sería tener una aplicación codificada con el SDK de Amazon S3 y querer comenzar a utilizar Google Storage provisto por jClouds. Como pudimos ver, ninguna de las herramientas disponibles en la actualidad considera este tipo de migración y esto deriva en que el desarrollador deba recodificar la aplicación para utilizar la nueva plataforma.

De acuerdo a la problemática planteada, el framework desarrollado propone reducir el esfuerzo de codificación al migrar una aplicación entre proveedores de servicios de Cloud Computing.

Como se pudo apreciar en los resultados experimentales, el trabajo presentado cumple satisfactoriamente con los objetivos planteados.

### 6.1 Limitaciones de la herramienta

La principal limitación de la herramienta radica en que está desarrollada para utilizarse solamente en aplicaciones escritas en lenguaje Java. Esto reduce el rango de aplicabilidad a desarrollos ya existentes escritos en un lenguaje diferente.

Otra limitación que encontramos se corresponde con la administración de las dependencias. Debido a que se utiliza MAVEN para administrar las dependencias utilizadas por el framework, es posible caer en la situación en que un proveedor de alguna librería modifique el código fuente de la versión que se utiliza y cree algún tipo de conflicto con las clases que la utilizan. Esta limitación es sencilla de solventar. En caso de que ocurra algún problema de este tipo se podría recurrir a descargar al repositorio de MAVEN la versión de la librería que se estaba utilizando de manera correcta o bien agregarla al proyecto como una librería estática e indicar esto en el archivo de configuración de MAVEN (pom.xml).

Con respecto a la usabilidad de la herramienta, como se mencionó en capítulos anteriores, nos encontramos con dificultades al momento de utilizar este framework en aplicaciones ya desarrolladas utilizando otro framework y que poseen un cargador de clases (classloader) personalizado. En estos casos el cargador de clases de Aether no funcionará de manera correcta debido a que las clases son cargadas por el classloader de la aplicación. Para solventar este problema el programador deberá modificar el código de la aplicación desarrollada para delegar la carga de clases al cargador de Aether permitiendo de esta forma que este cargue y modifique las clases que sean necesarias.

### 6.2. Trabajos futuros

Existen distintos aspectos de la herramienta que pueden ser mejorados como trabajos futuros e incluso extendidos para lograr tener un mejor provecho de su utilización:

* Usabilidad: La usabilidad es un atributo muy importante en este tipo de sistemas. Los experimentos que se llevaron a cabo a lo largo del documento fueron realizados por usuarios expertos en el dominio. Debido a esto sería deseable poder probar la utilización de la herramienta por parte de otros usuarios con diferentes conocimientos.
* Versatilidad de servicios: sería bastante útil extender el framework para abarcar otros tipos de servicios además del servicio de almacenamiento ya implementado. Este framework fue diseñado para poder extender los servicios e implementar el que fuera necesario de manera sencilla. Debido a que nos basamos en servicios de cloud para desarrollar el trabajo sería importante agregarle los servicios por ejemplo de colas (queue) y cómputo (compute).
* Soporte de métricas: Otro punto interesante sería agregar al framework soporte para obtener métricas sobre las ejecuciones de los servicios. Debido a que es muy sencillo intercambiar los proveedores de servicios y no es necesario modificar el código fuente de la aplicación se podrían tomar métricas fácilmente para los diferentes proveedores y poder comparar de esta manera el rendimiento de cada uno. Por ejemplo comparando el tiempo que tarda en ejecutar un determinado servicio, o realizar algún cálculo específico.
* Soporte de backup: Aunque la gran mayoría de los proveedores de servicios tienen métodos para realizar backups para solventar ciertos errores y caídas de sistemas, puede extenderse el framework para proveer al usuario una alternativa a los mecanismos utilizados por cada proveedor. Esto podría ser utilizado para mantener de manera transparente al usuario copias de los datos vitales de una aplicación en otro servidor o proveedor alternativo y de esta manera tener acceso a esos datos en casos de fallos en el servicio principal.