Programación Matemática y Software (2014) 6 (2): 10-16. ISSN: 2007-3283

Caracterización de un sistema *multicluster* HPC basado en un *middleware* de *clusters* de estaciones de trabajo

Characterization of a HPC multicluster system based on a middleware of a workstations cluster

Abelardo Rodríguez-León [1]*, Marco A. Romo-Medina [1], Guillermo Ovando-Chaco [1], Pedro Moreno-Bernal [2]

[1] Instituto Tecnológico de Veracruz.
Calzada Miguel Ángel de Quevedo 2779. Veracruz, México
[2] Posgrado en Ingeniería y Ciencias Aplicadas, Universidad Autónoma del Estado de Morelos.
Av. Universidad 1001, colonia Chamilpa. Cuernavaca, Morelos
*Correo-e: arleon@itver.edu.mx

PALABRAS CLAVE:

grid, programas paralelos *multi-site*, HPC

RESUMEN

En la actualidad, las plataformas que dan soporte al cómputo intensivo se han visto superadas por la complejidad de los problemas que en ellas intervienen; esto quiere decir que en muchas ocasiones no alcanzan a generar todo el poder de cómputo que los problemas requieren. Una solución que se había utilizado es la tecnología *grid*; sin embargo, su uso necesita una infraestructura (hardware y software) más compleja que la de computación de alto desempeño (HPC). Otra limitante en esta tecnología es que no se pueden ejecutar programas paralelos *multi-site*. En el presente artículo se muestra una alternativa que puede resolver estos problemas denominada *multicluster* HPC. Ésta permite la ejecución de programas paralelos *multi-site* sin la limitación de la *grid*. Lo anterior se logra con la unión de varios *clusters*, cada uno con un *middleware* y unidos por una VPN.

KEYWORDS:

grid, multi-site parallel programming, HPC

ABSTRACT

At present, the platforms that support the intensive computation have been overtaken by the complexity of the problems that they run. This means that in many cases, these platforms do not reach to cover all the computing power that the problems require. A solution that had been using is the grid technology. However, the use of this technology needs a more complex infrastructure (hardware and software) than the high-performance computing (HPC). Another limiting factor of this technology is that makes it impossible to run parallel programs multi*site*s. This article shows an alternative that can solve these problems called multicluster HPC. This alternative allows the execution of parallel multisite programs without the limitation of the grid. This is accomplished with the union of several clusters linked by a VPN, each of which has a middleware.

1 INTRODUCCIÓN

Hablar de cómputo intensivo es hablar de supercomputadoras, *cluster*s y *grids*. En los inicios del cómputo intensivo, la única manera de obtener gran capacidad de poder de cómputo era mediante las supercomputadoras, las cuales son el tipo de computadoras más rápido y potente que existe en un lapso determinado. Su principal característica es que pueden procesar enormes cantidades de información en poco tiempo [1].

Una evolución natural de las supercomputadoras se dio con el surgimiento de los *cluster*s, un conjunto de computadoras (personales o servidores) interconectadas por una red de alta velocidad y software especializado (*middleware*) que funcionan como si se tratara de una sola supercomputadora [2]. Los *cluster*s se han convertido en el sustituto de las supercomputadoras porque alcanzan el mismo poder de cómputo pero a costos mucho más bajos y con la ventaja adicional de una escalabilidad progresiva. Existen tres diferentes tipos de *cluster* según el servicio que prestan:

- *Cluster* de alto rendimiento (HPC)
- *Cluster* de alta disponibilidad (HA)
- *Cluster* de alta productividad (HTC)

La computación *grid* es el siguiente nivel de tecnología de uso de recursos distribuidos. Una *grid* es la tecnología que permite utilizar diferentes recursos como cómputo (supercomputadoras y *cluster*s), almacenamiento (BD y RAIDS) y aplicaciones específicas, que no están centralizadas geográficamente [2]. Sin embargo, el uso de una *grid* requiere una serie de conocimientos adicionales tanto para su administración como para su utilización, lo que ha creado ciertas dificultades para algunos usuarios, ya que tienen que dedicar más tiempo para poder realizar pruebas en la infraestructura *grid*, en vez de destinarlo a la solución de los problemas en sus proyectos.

1.1 Problemática y estado del arte

El uso de *cluster*s para cómputo intensivo y paralelo ha crecido en los últimos años. En la actualidad hay una gran cantidad de *cluster*s de producción en los que corren softwares paralelos. Al hablar de *cluster*s paralelos no se puede omitir el modelo de programación por pasos de mensajes, ya que estas dos tecnologías van de la mano. Dentro de las bibliotecas de paso de mensajes más populares se encuentran PVM (Parallel Virtual Machine) y MPI (Message Passing Interface); la última se convirtió rápidamente en un estándar para el desarrollo y ejecución de programas paralelos. El éxito inicial de esta biblioteca se debe, en gran medida, a su versión libre en dos implementaciones: MPICH [3] y OPENMPI, antes LAM MPI [4].

Otro factor importante es la creación y fortalecimiento de una infraestructura de red de datos de alta velocidad para cómputo intensivo, que involucra la formación de recursos humanos altamente capacitados.

En una revisión de los trabajos relacionados con la problemática planteada, se encontró que básicamente se cuenta con dos soluciones. Una de ellas se orienta hacia el uso de la tecnología *grid* y la otra deriva en el uso de las capacidades ampliadas de los *cluster*s a través de *multiclusters*.

Trobec *et al*. [1] definen *grid* como "un tipo de sistema paralelo y distribuido geográficamente que permite la compartición de recursos de forma dinámica en tiempo de ejecución en función de su disponibilidad, capacidad, rendimiento, costo y usuarios con requerimientos de calidad para sus servicios".

La computación *grid* permite compartir recursos sobre una red de sistemas heterogéneos a través de estándares abiertos, con el fin de optimizar los resultados. Esto significa que se cuenta con mayor poder de CPU, almacenamiento, memoria y recursos de red, con la finalidad de que el usuario pueda acceder cuando sea necesario sin importar la ubicación geográfica en la que se encuentre.

Las *grid* se usan principalmente para servicios y minería de datos. Para [5], en general, las bibliotecas de paso de mensajes resuelven básicamente dos problemas técnicos:

- Identificación única de procesos. Los programas paralelos son un conjunto de procesos que se pueden identificar de manera unívoca, independientemente de que se ejecuten en un ambiente de memoria compartida o distribuida.
- Transferencia de datos entre los procesos. Se reconoce que las primitivas básicas de comunicación son del tipo punto a punto *send*() y *receive*() entre dos procesos; también se suelen incluir otras variantes como las comunicaciones colectivas (del tipo de *broadcast*, por ejemplo).

Tabla 1. Comparativo *grid* vs *multicluster* HPC

CARACTERÍSTICAS	Escalabilidad	Distribución geográfica	Heterogeneidad	Recursos compartidos	Múltiples administradores	Seguridad del *middleware*	Administración del *middleware*	Programas paralelos con *middleware*	Acceso
Grid	Buena	Buena	Buena	Buena	Buena	Buena	Buena	Regular	Buena
MULTICLUTER	Regular	Buena	Buena	Buena	Mala	Regular	Buena	Buena	Buena

En este sentido, en cualquier plataforma (hardware) que pueda resolver estos problemas se podría utilizar una implementación de paso de mensajes como lo es MPI. Los *cluster*s han adoptado de manera satisfactoria estas librerías, ya que resuelven estos dos problemas de una manera sencilla. Adicionalmente, todo lo relacionado con la transferencia de datos y comunicaciones lo resuelve el *middleware* de los *cluster*s con protocolos estándares como lo es TCP/IP.

Lo antes mencionado se complica si se tiene la idea de ejecutar programas paralelos en más de un *cluster*, dicho de otra forma: en ejecución *multicluster*. Algunos problemas que se tienen al ejecutar programas paralelos *multicluster* son los siguientes:

- No se puede asumir que todas las comunicaciones punto a punto tengan el mismo rendimiento, puesto que un proceso puede enviar datos a otros dos procesos: uno ubicado en un nodo dentro del mismo *cluster* y el otro en un nodo de un *cluster* remoto.
- La heterogeneidad de las máquinas que componen a los diferentes *cluster*s y la asimetría de las unidades de procesamiento en un mismo *cluster* son muy comunes, por lo tanto, también es probable que se den en los nodos del *cluster* remoto.

De acuerdo con [5], en los primeros esfuerzos por realizar herramientas o bibliotecas que pudieran llevar a cabo la ejecución de programas paralelos *multicluster* se puede mencionar la biblioteca MagPIe del proyecto Albatros [6]. Ésta fue orientada a la optimización de las comunicaciones colectivas de MPI en redes WAN. Otras bibliotecas fueron MPI-Connect, que después fue remplazada por PVMPI y, por último, IMPI; todas estas bibliotecas fueron propuestas al final de la década de 1990.

Con la intención de comparar gráficamente las tecnologías estudiadas, en la tabla 1 se muestran las diferencias entre cada una de las características de la computación *grid* y la *multicluster* HPC.

Con base en el comparativo entre las dos tecnologías, es evidente que la tecnología *grid* es mucho más robusta que el *multicluster*. Sin embargo, esta cualidad conlleva la implementación de más recursos y, aun con todas las características en las que supera la computación *grid* al *multicluster*, la primera no puede realizar la ejecución de programas paralelos *multicluster*.

La primera característica comparada es la escalabilidad. En la computación *grid* es muy buena, puesto que las diferentes plataformas existentes en la actualidad tienen, desde unos cuantos cientos, hasta millares de usuarios; en *multicluster* esta característica es un poco limitada, pues depende de la tecnología que se utilice para la interconexión de los *cluster*s que conforman la plataforma en su totalidad; la más utilizada es OpenVPN, que en su versión libre se ve limitada a unos cuantos usuarios.

La segunda característica que se compara fue la distribución geográfica, para la que ambas tecnologías tienen una buena distribución, ya que ninguna está limitada. En el caso de la *grid*, un *site* puede estar en el continente europeo y otro en Latinoamérica sin que exista algún problema entre ellos. En el caso de *multicluster*, de igual manera pueden estar separados los *cluster* sin presentar inconvenientes.

La heterogeneidad es la tercera característica que se compara en la tabla 1, donde no se encuentra ningún problema al respecto, ya que ambas aprovechan los recursos de diferentes arquitecturas, lo cual no representa un factor que impida su buen funcionamiento.

En relación con la compartición de recursos, en ambas es favorable puesto que optimizan al máximo la disponibilidad de sus recursos entre sus usuarios.

La siguiente característica que se compara es la de múltiple administración. En este aspecto, la *grid* cuenta con un muy buen esquema de administración de tres diferentes niveles (local, regional y global). En el caso de *multicluster*, esta tarea se lleva a cabo por una sola entidad centralizada que administra todo.

Al hablar de *middleware*, que es la siguiente característica comparada, es importante señalar que cada una de las tecnologías usa diferentes tipos. En el caso de la computación *grid*, el *middleware* es el encargado de la mayor parte de la administración. En cambio, en *multicluster* el *middleware* es independiente en cada *cluster* y la parte de la comunicación entre éstos es manejada por la VPN. La comparación entre los *middleware* se realizó en dos ámbitos: la seguridad y la administración. En el primero, el *middleware grid* supera al de *multicluster* al contener fuertes mecanismos de seguridad propios de la *grid*. En el segundo ámbito, ambos *middleware* son muy buenos, puesto que cuentan con los mecanismos necesarios como la administración de usuarios, permisos y demás, pertinentes de cada tecnología.

La penúltima característica comparada entre las tecnologías es la ejecución de programas paralelos con paso de mensajes. En ésta la computación *grid* se ve superada porque, aun cuando en una *grid* se puede ejecutar un programa paralelo, la ejecución se limita a un solo *site*. Esto quiere decir que, si un programa quiere correr sus procesos simultáneamente en más de un *site*, no es posible hacerlo; en cambio, en *multicluster* no existe tal limitación.

La última comparación corresponde a la accesibilidad. En ésta, las dos tecnologías garantizan de manera adecuada el acceso a su infraestructura a todos los usuarios adecuadamente identificados con certificados propios provenientes de una entidad.

2 PROPUESTA DE SOLUCIÓN

La idea central de este trabajo es mostrar la posibilidad de ejecutar programas paralelos en más de un *cluster* sin la necesidad de instalar toda una infraestructura (*middleware*) de computación *grid*.

Es conveniente mantener la propuesta de cómputo paralelo *multicluster* separada de la de *grid* computing por al menos dos razones:

- Compartición de recursos: la computación *grid* está propuesta como una solución integral para compartir recursos de cómputo y almacenamiento a gran escala de distribución y capacidad.
- Administración y seguridad de la infraestructura *grid*: esta tecnología tiene más de lo necesario en comparación con el cómputo paralelo *multicluster*, ya que la computación *grid* implementa un sistema muy elaborado para proveer la adaptación de los diferentes sistemas de seguridad que se utilizan localmente por cada institución conectada.

Según [5], desde hace algún tiempo se han estudiado características específicas de cómputo paralelo *multicluster* tales como el problema generado por las interconexiones no dedicadas y los problemas de seguridad involucrados. A modo de resumen, en dicho trabajo se reportan algunos detalles técnicos importantes que hay que tener en cuenta para implementar el cómputo paralelo *multicluster*:

- En los ambientes no dedicados, muchos de los problemas de disponibilidad de los nodos de los *cluster*s son propios de la falta de control sobre los mismos, no de las comunicaciones o estabilidad de los nodos.
- Aunque la interconexión de los *cluster*s no es exclusiva, siempre hay conectividad entre los *cluster*s utilizados salvo algunas excepciones poco frecuentes. Esto se debe a que la interconexión se encuentra mezclada con el tráfico de Internet, por lo que se debe procurar mantener una buena administración de la red de comunicaciones, independientemente de la utilización de cómputo paralelo *multicluster*.
- El rendimiento de las comunicaciones entre los *cluster*s no es constante dado que no es dedicado, pero en general es muy cercano al máximo absoluto, al menos para transferencias de pocos datos. Como es de esperar, el rendimiento para las comunicaciones *multiclusters* fluctúa dependiendo de días y horarios.
- Los mecanismos básicos de seguridad que se imponen en las instituciones (y que son de uso común en casi todas las instalaciones de computadoras) normalmente impiden la utilización directa de implementaciones de MPI como MPICH y OPENMPI. Estas implementaciones imponen un patrón de tráfico TCP/IP que normalmente es cancelado por los *firewalls* o mecanismos de seguridad de las instituciones.

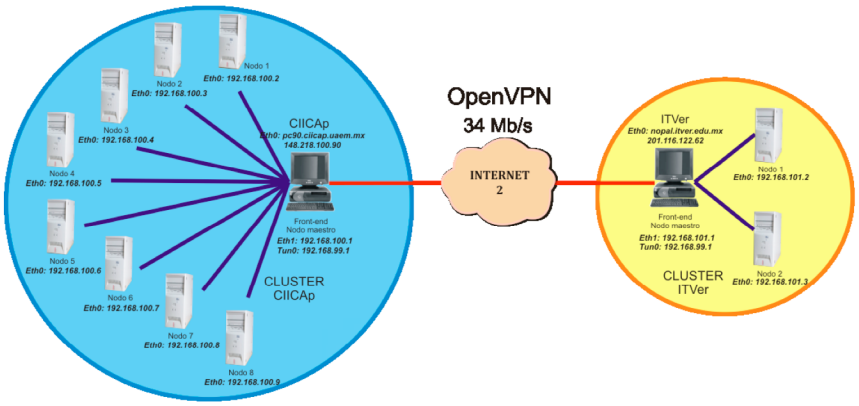

Figura 1. Diagrama estructural inicial del *multicluster*

Después de analizar todas las limitaciones que presenta la ejecución de programas paralelos *multicluster*, se estudió una posible solución, la cual provee las herramientas necesarias para atacar cada uno de los puntos antes expuestos.

Para [5], una solución es la utilización de una VPN. Desde el punto de vista técnico, todo lo que necesita MPI para ser utilizado en un *cluster* es tener conectividad TCP/IP en una red local, y esto es precisamente lo que una VPN provee en cualquiera de sus versiones e implementaciones. Sin lugar a duda, se tienen algunos costos por la implementación de una VPN para cómputo paralelo *multicluster*, los cuales son:

- La instalación y configuración del software de la implementación VPN que se haya elegido, en nuestro caso OpenVPN.
- Se debe de tomar en cuenta que se afectarán los controles de seguridad (*firewalls*) implementados en las instituciones donde se encuentren ubicados los *cluster*s.
- En el tiempo de ejecución existe una sobrecarga de procesamiento y en el tráfico de datos por el entubamiento (encriptación, enrutamiento y desencriptación) de las comunicaciones sobre el esquema de cliente/servidor a través de un puerto bien conocido.

En el reporte que presentan [5], mencionan que no todas las implementaciones de MPI se pueden ejecutar en una VPN, por lo que optan por utilizar MPICH, que no presentó ninguna complicación. Adicionalmente, se ha comprobado que MPICH2 soporta, sin problemas, arquitecturas heterogéneas *multicluster*; es decir, arquitecturas de 32 y 64 bits. Este es un punto muy importante a considerar para los proyectos que desarrollan los autores del presente trabajo.

Las conclusiones descritas por [5] indican que se experimentó en un escenario en el cual se conectaban dos *cluster*s mediante una VPN para ejecutar un programa paralelo con procesos que se realizaban en ambos de manera simultánea. Lo anterior comprueba que sí es posible la ejecución de programas *multicluster* sin la necesidad de la instalación de una infraestructura (*middleware*) como la que proporciona la computación *grid*.

En este proyecto en específico, el *middleware* utilizado fue Rocks, el cual se define como una colección de software libre para la administración de *cluster*s HPC. El objetivo principal al utilizar el diseño de Rocks es hacer la instalación del *cluster* tan fácil como sea posible. Sin lugar a dudas, se ha recorrido un largo camino hacia el logro de este objetivo. Para cumplirlo, la instalación por defecto hace una serie de suposiciones razonables acerca de lo que el software debe incluir y cómo debe estar configurado el *cluster*. Sin embargo, con un poco más de trabajo, es posible personalizar muchos aspectos de Rocks [2].

Otra tecnología empleada en este proyecto es una VPN. La VPN crea una red privada sobre una red pública (usualmente Internet) para conectar sitios remotos o usuarios entre sí. Una VPN usa conexiones "virtuales" enrutadas a través de Internet desde una red privada hacia un sitio público externo. Mediante el uso de una VPN se puede garantizar la seguridad entre los nodos interconectados [7]. La implementación de esta tecnología fue a través de OpenVPN la cual brinda una serie de características:

- Compresión en tiempo real
- Redes de túneles a través de NAT
- Utilizar toda la encriptación, autenticación y certificación de las características de la biblioteca OpenSSL para proteger el tráfico de red privada, ya que transita por Internet
- Posibilidad de crear un túnel de cualquier subred IP virtuales o adaptador ethernet través de un único puerto UDP o TCP

Además de Rocks y OpenVPN se utiliza Anaconda, el programa de instalación de Red Hat, que está escrito en Python con algunos módulos personalizados en C. Anaconda está organizado en etapas: la primera es un instalador que carga los módulos del kernel necesarios; posteriormente utiliza un mecanismo de autoinstalación que permite instalaciones masivas a través del archivo de configuración *kickstart* [8]; la instalación con *kickstart* se puede hacer usando un CD-ROM, disco duro o en una red mediante FTP, NFS o HTTP.[8] Estas dos herramientas son utilizadas para la automatización de la instalación de los elementos del *multicluster*.

3 ANÁLISIS DE RESULTADOS

Integrar una distribución de *middleware* para *cluster* base (en este caso Rocks) con la tecnología VPN (OpenVPN) permite obtener una herramienta para la ejecución de programas paralelos *multi-site* y aprovechar al máximo cada una de las características de las dos tecnologías fusionadas dejando de lado las complicaciones que conlleva la utilización de la computación *grid*, tanto las de infraestructura como las administrativas.

La implementación se puso a prueba en dos *cluster*s distribuidos geográficamente: uno se encuentra en el Instituto Tecnológico de Veracruz y el otro en el Centro de Investigación en Ingeniería y Ciencias Aplicadas de la Universidad Autónoma del Estado de Morelos. En las primeras pruebas se realizaron las siguientes tareas: instalación de la VPN e instalación y configuración de los paquetes adicionales de forma manual (sin la utilización de Anaconda y *kickstart*).

Actualmente se están realizando las modificaciones en el instalador de Rocks sobre Anaconda para poder llevar a cabo todo el proceso de forma automatizada. Esto tiene como finalidad que los administradores de *multicluster* puedan realizar instalaciones y configuraciones de manera sencilla, sin tener que adquirir conocimientos adicionales sobre sistemas operativos y redes.

3.1 Proyección a futuro

La meta de este proyecto es tener la implementación estructurada en su totalidad. Esto conlleva, desde la realización de la VPN y su configuración, hasta realizar el proceso de manera automatizada.

Se pretende agregar progresivamente servicios que simplifiquen cada vez más la administración de sistemas *multicluster* HPC dentro de los cuales se incluyen: el seccionamiento de las redes privadas, agregar nuevos miembros a la VPN, así como automatizar todo el proceso.

Otro punto que se pretende lograr en trabajos posteriores es la evaluación en conjunto de la implementación, es decir, poner en marcha la implementación y observar los procesos que realiza, desde la instalación de los paquetes adicionales, hasta comprobar que haya realizado de manera correcta cada una de las tareas asignadas, así como el tiempo que lleve realizar la instalación íntegra.

4 CONCLUSIONES

Contar con un *multicluster* HPC permite realizar la ejecución de programas paralelos *multi-site*, sin necesidad de instalar toda la infraestructura que implicaría la computación *grid*. El uso de un *middleware* popular como lo es Rocks-*Cluster* modificado y con servicios adicionales facilita la instalación debido a que no se requieren conocimientos adicionales para su administración ni para la ejecución de programas paralelos en varias plataformas sin intervención del usuario.

REFERENCIAS

1. Trobec, R., Vajtersic, M., Zinterhof, P. *Parallel Computing Numerics, Applications and Trends*. London: Springer, 2009.
2. Joseph, D. *High Performance Linux Clusterswhitc OSCAR, Rocks, openMosix and MPI*. United States of America: O'Reilly, 2004.
3. MPICH. *Frequently asked questions*. Homepage. (Consultado mayo 2014). Disponible en: http://wiki.mpich.org/mpich/index.php/Frequently_Asked_Questions
4. OpenMPI. *General information about the Open MPI Project*. (Colnsultado mayo 2014). Disponible en: https://www.open-mpi.org/faq/?category=general
5. Tinetti, F. G., Aroztegui, W. J. Factibilidad y rendimiento de las comunicaciones para computo paralelo intercluster. Argentina: CACIC, 2006.
6. Albatross. Wide Area*Cluster* Computing Homepage. (Consultado mayo 2011). Disponible en: http://www.cs.vu.nl/albatross/
7. Cisco. How VPNs work. (Consultado enero del 2013). Disponible en: http://www.cisco.com/en/US/tech/tk583/tk372/technologies_tech_note09186a0080094865.shtlml
8. Landmann Rudiger, et all. Red Hat Enterprise Linux 6 Guia de Instalación. Red Hat Inc. 2011. (Consultado enero del 2013). Disponible en https://access.redhat.com/documentation/es-ES/Red_Hat_Enterprise_Linux/6/pdf/Installation_Guide/Red_Hat_Enterprise_Linux-6-Installation_Guide-es-ES.pdf

Acerca de los autores

Guillermo Ovando Chaco es Ingeniero Mecánico egresado del Instituto Tecnológico de Tuxtla Gutierrez. Realizó la Maestría en Ciencias en Ingeniería Mecánica (térmica) en el Instituto Tecnológico de Veracruz y cuenta con Doctorado en Ingeniería Mecánica (termofluidos) por el Centro de Investigación en Energía de la Universidad Nacional Autónoma de México. Sus áreas de interés son: dinámica de fluidos computacionales, transferencia de calor y masa y elemento finito.

Marco Alberto Romo Medina es Ingeniero Industrial en Eléctrica egresado del Instituto Tecnológico de Aguascalientes. Realizó la Maestría en Ingeniería Nuclear en la Escuela Politécnica de Montreal, Canadá. Su área de interés es la aplicación de métodos numéricos a diversas ramas de las ingenierías.

Abelardo Rodríguez León es Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Veracruz. Realizó la Maestría en Ciencias Computacionales por la Universidad Veracruzana y cuenta con Doctorado en Ciencias Computacionales por la Universidad Politécnica de Valencia, España. Actualmente es profesor investigador en el Departamento de Computación y Sistemas del Instituto Tecnológico de Veracruz. Sus áreas de interés incluyen la computación de alto rendimiento paralela y en *grid*, además de estudios de modelos gráficos en 3D.

Pedro Moreno es Maestro en Ingeniería y Ciencias Aplicadas con mención honorifica por el Centro de Investigación en Ingeniería y Ciencias Aplicadas de la Universidad Autónoma del Estado de Morelos. Es responsable de la instalación y administración de la infraestructura de cómputo de alto rendimiento que conforma la *Grid* Morelos. Actualmente estudia el doctorado en Ciencias Computacionales en el Ciicap de la UAEM.