EVALUACIÓN DEL DESEMPEÑO DE UN CLUSTER DE ALTO RENDIMIENTO SOBRE UNA INFRAESTRUCTURA DE NUBE.

Arellano García Alexis Israel, Fernández Molina José Eduardo.

Ingeniería Electrónica en Redes y Comunicación de Datos

Resumen — **El presente artículo se enfoca en el análisis de los resultados obtenidos al ejecutar programas que utilizan librerías MPI en un Clúster HPC, mediante el cual se utilizó diferentes cantidades de procesadores para distribuir el trabajo y con la variación de los tiempos de respuesta determinar la aceleración, dependiendo de la eficacia del programa, es decir la porción paralelizable del programa. Adicionalmente gracias a la herramienta Torque/PBS se pudo administrar y monitorear los recursos de los sistemas distribuidos del Clúster, agrupándolos en diferentes colas, la una para ejecutar programas que no demanden de mucho procesamiento y la otra, con la mayoría de los procesadores que es utilizada para las tareas de que utilicen gran cantidad de recursos.**

Índice de Términos— CLUSTER, HPC, TORQUE, LINUX, BATCH.

I. INTRODUCCIÓN

El presente artículo presenta un análisis y comparación de los resultados obtenidos al ejecutar una aplicación escrita en lenguaje C++ utilizando librerías MPI y compilada con el paquete openMPI en un entorno de virtualización como escenario de pruebas para posteriormente implementar el Clúster HPC en un entorno de laboratorio.

Un Clúster HPC es un conjunto de computadores que funcionan como si fueran uno solo, es decir, comparten recursos (procesador, memoria RAM, almacenamieto, etc) y ejecutan programas paralelizables que demanden de un alto grado de procesamiento o uso de memoria. Son utilizados en el área científica para resolver problemas o simulaciones que requieran un gran cálculo matemático, en la renderización de gráficos y en mejorar el tiempo de respuesta para la solución de un problema.

Para que un Clúster pueda distribuir un trabajo a sus nodos, debe tener la capacidad de utilizar aplicaciones paralelas, existen diferentes programas que realizan esta tarea, entre estas aplicaciones podemos encontrar openMPI y también MPICH. Estas suites de aplicaciones cuentan con compiladores que permiten adaptar un programa escrito en lenguajes como C, C++ o Fortran para que sea posibles ejecutarlas en entornos paralelos, brindando la opción de elegir el número de nodos y la cantidad de procesadores que se utilizarán para la ejecución del programa [1].

II. DISEÑO DEL CLÚSTER HPC

Para el diseño se tomó en cuenta la utilización de herramientas de software libre, tanto para el sistema base como para las aplicaciones y suites de paquetes que permiten la simulación de un Clúster HPC.

Para motivos de análisis y comparación de resultados se utilizó dos computadores con arquitecturas de procesadores diferentes para la simulación del Clúster. El sistema operativo base seleccionado es Debian Squeeze y los computadores utilizados se detallan en la **Tabla 1**.

Tabla 1Descripción de computadores

	Características			
	Procesador	Velocidad CPU	Núcleos	RAM
PC1	Intel Core i7	2.4Ghz	8	8 Gb
PC2	Intel Core i5	2.4Ghz	4	6 Gb

Una vez que se seleccionaron los computadores y el software apropiados, se procede a la instalación y configuración de las aplicaciones y servicios. Los servicios y aplicaciones que deben estar instalados y correctamente configurados son, como mínimo: DHCP, SSH, NIS, NFS, NTP, BLCR, openMPI, Ganglia.

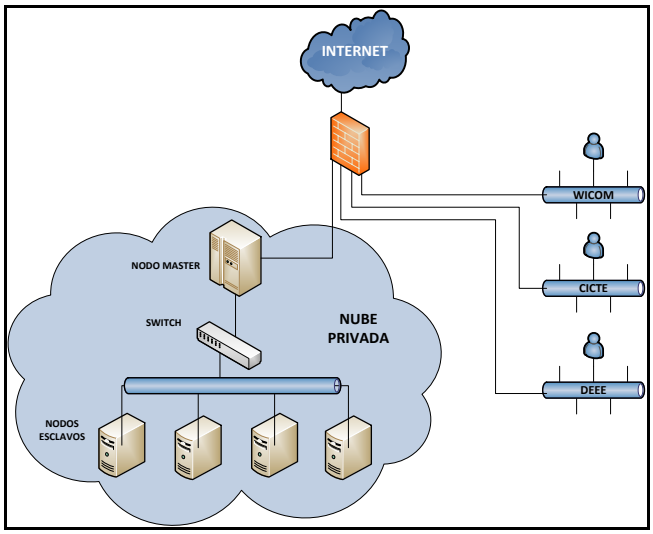
Se debe tomar en cuenta que el computador que ejerce el roll de FrontEnd debe tener al menos 2 interfaces de red, mientras que los computadores nodos solamente necesitan una interfaz de red. La topología y servicios en ambos sistemas operativos será la misma, la Figura 1 muestra en resumen la topología:

La implementación del clúster se lo realizó en el laboratorio de Networking del Departamento de Electrónica de la Universidad de las Fuerzas Armadas – ESPE, permitiendo el acceso de los usuarios a través de una conexión remota hacia el nodo Master que es el que se encarga de administrar la

alexisra@_17hotmail.com; jefernandez89@outlook.com

nube privada en la que se encuentra el Clúster, con lo cual los usuarios podrán hacer uso de los recursos del mismo. La conexión remota se la realiza a través de un cliente SSH que apunte a la dirección de red con la cual el nodo Master se conecta a la red interna de la Universidad, teniendo en cuenta que debe tener un usuario previamente configurado por el Administrador, caso contrario no se podrá acceder a los recursos del Clúster.

Figura 1 Topología del Clúster [2]



III. UTILIZACIÓN DE PROGRAMAS

Se utilizó para la toma de datos un programa escrito en lenguaje C++, que, determina la cantidad de números primos que hay entre cero y un número dado y entrega en su salida el tiempo que tardó en ejecutarse el programa. Los datos se obtuvieron cambiando la cantidad de nodos y el grado de paralelización del programa para resolver el problema planteado, y observar el tiempo de respuesta ante la variación de estos parámetros. Cabe mencionar que en el escenario de virtualización cada nodo contiene un solo procesador.

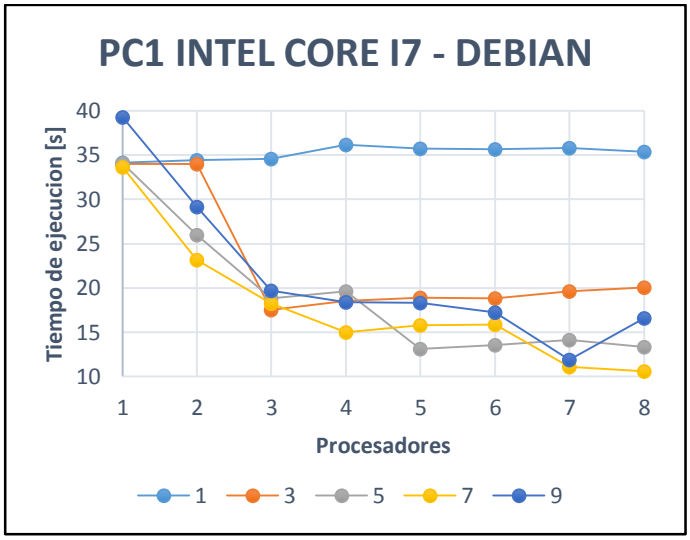
A continuación se presenta los resultados obtenidos, en los Clúster simulados en los computadores:

- PC1

Tabla 2 Resultados de Debian en PC1

Intel® Core ™ i7-4700MQ CPU @ 2.40GHz – Debian					
	Número de Procesos				
	1	3	5	7	9
1	34,1159	34,0204	34,0489	33,6528	39,2246
2	34,3892	33,9639	25,9724	23,1848	29,1753
3	34,5269	17,5077	18,8327	18,2608	19,6818
4	36,1812	18,5621	19,6231	15,0403	18,3786
5	35,7275	18,8838	13,1562	15,8056	18,3017
6	35,6669	18,816	13,5685	15,8915	17,2147
7	35,7589	19,648	14,178	11,1466	11,9012
8	35,3907	20,0952	13,3203	10,617	16,5928

Figura 2 Resultados de Debian en PC1

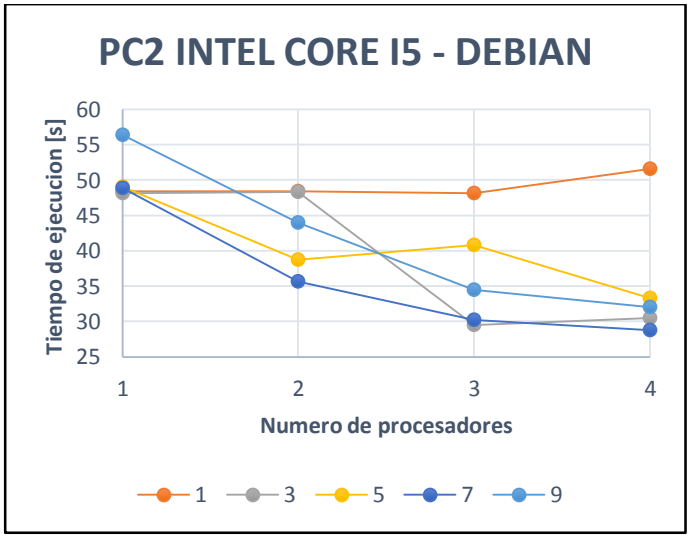


- PC2

Tabla 3 Resultados de Debian en PC2

Intel® Core™ i5 M450 @ 2.4Ghz- Debian					
	Número de Procesos				
	1	3	5	7	9
1	48,4141	48,1195	49,0035	48,8818	56,3563
2	48,4401	48,3275	38,7436	35,6389	43,9255
3	48,1471	29,5052	40,7559	30,2305	34,4962
4	51,5584	30,4680	33,2511	28,7569	32,0611

Figura 3 Resultados de Debian en PC2

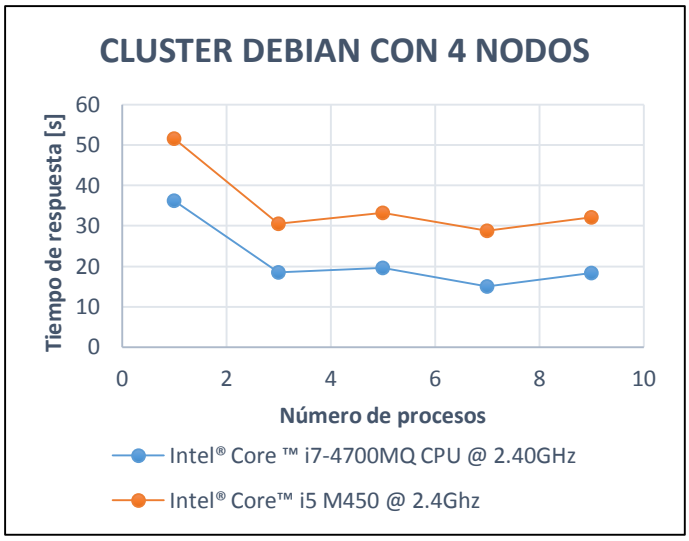


- COMPARACIÓN

Tabla 4 Comparación de resultados entre PC1 y PC2

Número de Procesos	Intel® Core™ i7-4700MQ CPU @ 2.40GHz	Intel® Core™ i5 M450 @ 2.4Ghz
1	36,1812	51,5584
3	18,5621	30,468
5	19,6231	33,2511
7	15,0403	28,7569
9	18,3786	32,0611

Figura 4 Comparación de resultados entre PC1 y PC2



IV. IMPLEMENTACIÓN EN ENTORNO DE LABORATORIO:

Tabla 5 Características de las PC disponibles para la implementación

PC	Procesador	Velocidad	Núcleos	RAM	Disco
Master	Intel Core 2 duo	2.08 Ghz	2	2 GB	150 GB
Nodo 1	Intel Pentium 4	3.12 Ghz	2	1 GB	110 GB
Nodo 2	Intel Pentium 4	3.12 Ghz	2	1 GB	110 GB
Nodo 3	Intel Pentium 4	3.12 Ghz	2	1 GB	110 GB
Nodo 4	Intel Core 2 duo	2.08 Ghz	2	2 GB	150 GB
Nodo 5	Intel Core 2 duo	2.08 Ghz	2	1 GB	150 GB
Nodo 6	Intel Core 2 duo	2.08 Ghz	2	1 GB	150 GB
Nodo 7	Intel Core 2 duo	2.08 Ghz	2	2 GB	150 GB
Nodo 8	Intel Core 2 quad	2.33 Ghz	4	2 GB	300 GB
Nodo 9	Intel Core 2 quad	2.33 Ghz	4	3 GB	360 GB

Se generaron 2 colas para realizar el análisis del rendimiento total del Clúster implementado en el entorno de laboratorio, la primera (Allow) que utilizará los nodos 1, 2 y 3 que se describen en la Tabla 5, debido a que estos nodos son Intel Pentium 4 y aumentan el tiempo de respuesta al interactuar con los nodos restantes. La segunda cola (Batch) contendrá a los nodos 4, 5, 6 y 7 que poseen un procesador Intel Core 2 Duo de 2 núcleos cada uno y los nodos 8 y 9 que poseen un procesador Intel Core Quad de cuatro núcleos.
El programa Primos.c++ será utilizado para medir el tiempo de respuesta arrojado por el clúster al variar la cantidad de nodos disponibles y utilizando cada una de las colas creadas.

- Cola Allow

La cola Allow es la que contiene los nodos con procesador Intel Pentium 4 y será utilizada para operaciones que no sean de alta prioridad y no demanden gran capacidad de procesamiento.

- Cola Batch

Como se explicó previamente, la Cola Batch cuenta con 6 nodos, de los cuales los nodos del 4 al 7 poseen un procesador Intel Core 2 Duo de dos núcleos cada uno y los nodos 8 y 9 que poseen un procesador Intel Core Quad de cuatro núcleos. Se recalca esta información ya que se observa en la Tabla 7 que el tiempo de respuesta al momento de ejecutar el programa en un solo procesos se reduce aproximadamente a la mitad, eso se debe a las características del procesador Core Quad son mejores frente a las de los Core 2 Duo.

Tabla 6 Resultados obtenidos utilizando la Cola Allow

COLA ALLOW			
Número de procesos	Procesadores		
	2	4	6
1	158,502	158,692	158,678
3	148,635	79,3324	79,5515
5	149,939	115,024	74,3422
7	152,355	82,2274	75,9155

Tabla 7 Resultados obtenidos utilizando la Cola Batch

COLA BATCH						
Número de procesos	Procesadores					
	2	4	6	8	12	16
1	70,422	70,416	70,409	70,404	35,670	35,666
3	35,283	35,222	35,297	35,266	35,268	18,816
5	35,269	27,024	17,688	17,684	17,684	16,568
7	35,322	17,929	17,960	11,801	11,801	11,811
9	41,926	23,547	17,444	11,795	11,800	11,780
11	35,332	18,107	14,116	11,231	7,100	7,101
13	35,727	20,875	14,921	11,714	8,885	5,906

Figura 5 Gráfica de resultados obtenidos con la Cola Allow

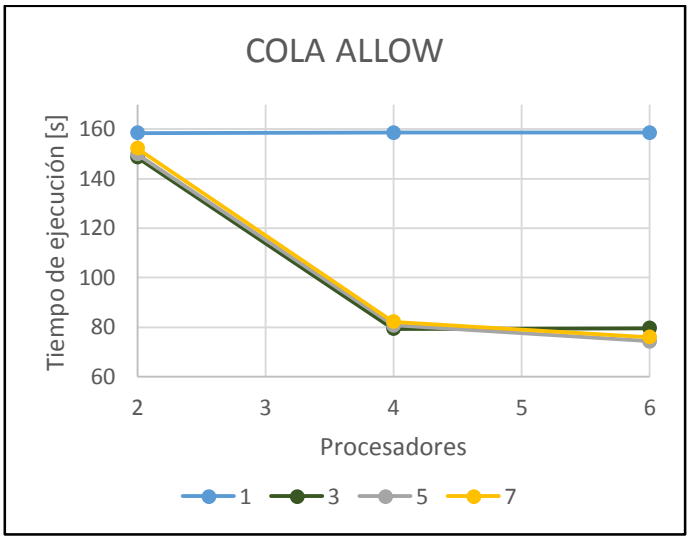
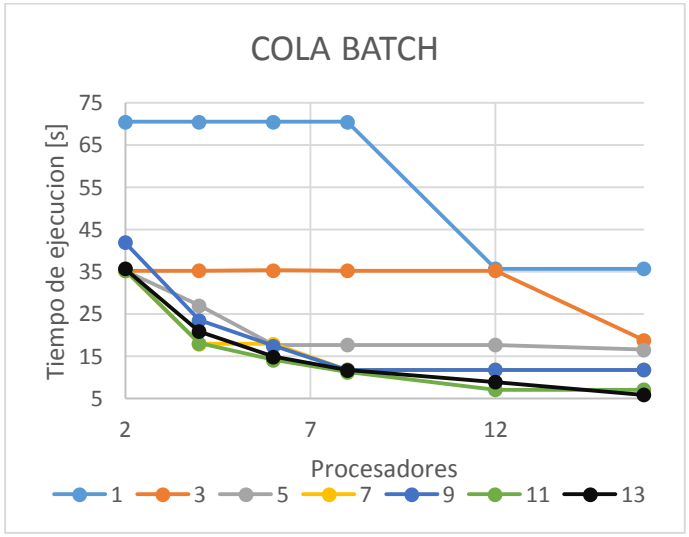


Figura 6 Gráfica de resultados obtenidos con la Cola Batch



V. ANÁLISIS DE RESULTADOS:

Al ejecutar el programa sin dividirlo en varios procesos se verifica que el tiempo de respuesta no varía significativamente aunque se aumente la cantidad de nodos para su ejecución, ya que solamente tomará un procesador para realizar la tarea.

Los tiempos de respuesta obtenidos al variar el número de procesos en los que se divide el programa se reducen aproximadamente a la mitad, cuando se aumenta el tamaño del clúster a 2 nodos como se observa en la Figura 5. Pero al momento de aumentar un nodo más, es decir al contar con 3 nodos en el clúster, el tiempo de respuesta no se reduce a la mitad como ser esperaría según el comportamiento anterior, esto se debe a que los procesadores alcanzan su máxima capacidad, y, a pesar que se aumente la cantidad de procesos en los que se divide el programa, el tiempo de respuesta no se podrá reducir más.

También se observa en la Tabla 7 que al momento de dividir el programa en 3 procesos y utilizando los nodos del 4 al 8, el tiempo se mantiene casi constante, pero al momento de aumentar el nodo 9, el tiempo de respuesta se reduce casi a la mitad de los anteriores, esta variación se debe a que al momento de integrar el Nodo 9 al clúster, se disponen de 16 procesadores, 8 de ellos son Core 2 Duo y los 8 restantes son Quad Core, motivo por el cual los 3 procesos en los que se dividió el programa tienen mayor posibilidad de ejecutarse en un procesador Quad Core, lo cual reduce el tiempo de respuesta. La tendencia de reducción del tiempo de respuesta es que, al aumentar el número de procesadores se llegará a un punto en el cual se estabilizará y no podrá reducirse más dicho tiempo, esto dependerá del número de procesos en que se divida el programa.

Al aumentar el número de procesos quiere decir que tan paralelizable es nuestro programa, que para nuestro caso se llegó hasta un 83,449% es decir una aceleración de aproximadamente 6,03838 veces de nuestra aceleración original. La cual se obtuvo de la siguiente forma.

Tabla 8 Resultados del porcentaje de paralelización del programa

Número de procesos	Tiempo	Porcentaje de Aceleración
1	35,6669	100%
3	18,816	47,25%
5	16,5685	6,30%
7	11,8115	13,34%
9	11,7801	0,09%
11	7,1012	13,12%
13	5,9067	3,35%
Total		83,44%

VI. CONCLUSIONES

Se estudiaron los componentes de Software necesarios para realizar la implementación de un Clúster de alto rendimiento, tomando en cuenta que se utilizaron solamente herramientas de Software libre. De acuerdo a este estudio se pudo determinar las herramientas que aprovechan de mejor manera los recursos de los equipos disponibles.

Una vez determinados los recursos de hardware y software, se procede a realizar el diseño del Clúster, para lo cual se determinó que el modelo Beowulf es el más óptimo para la implementación del mismo, debido a que brinda la posibilidad de utilizar computadores personales con características estándar, sin la necesidad de adquirir equipos de elevados costos que son dedicados para este tipo de implementaciones.

Se realizó la simulación del Clúster HPC utilizando el sistema operativo de software libre Debian, de acuerdo a los resultados

obtenidos en la simulación, y luego de seleccionar el sistema operativo Debian Squeezy como base, se realizó la instalación del Clúster HPC en los equipos del laboratorio de Networking, y se realizó las pruebas necesarias validando el correcto funcionamiento del Clúster y sus componentes.

Como punto final se realizó una estimación del rendimiento del Clúster HPC, mediante la utilización de programas desarrollados con librerías MPI, con lo cual se obtuvo una aceleración de aproximadamente 6 veces con respecto al tiempo original de respuesta.

VII. ANEXOS

1) Programa MPI: primos.c++ [3]

```
# include <cstdlib>
# include <iostream>
# include <iomanip>
# include <cmath>
# include <ctime>

# include "mpi.h"

using namespace std;

int main ( int argc, char *argv[] );
int prime_number ( int n, int id, int p );
void timestamp ( );

int main ( int argc, char *argv[] )

{
  int i;
  int id;
  int master = 0;
  int n;
  int n_factor;
  int n_hi;
  int n_lo;
  int p;
  int primes;
  int primes_part;
  double wtime;

  n_lo = 1;
  n_hi = 999999;
  n_factor = 2;
  MPI::Init ( argc, argv );
  p = MPI::COMM_WORLD.Get_size (  );
  id = MPI::COMM_WORLD.Get_rank ( );

  if ( id == master )
  {
    cout << "\n";
    cout << "'Cuenta primos\n";
    cout << "  C++/MPI version\n";
    cout << "\n";
    cout << "  Programa para contar cantidad de primos para un N dado.\n";
    cout << "  Corriendo en " << p << " procesos\n";
    cout << "\n";
    cout << "        N       S         Tiempo\n";
    cout << "\n";
  }

  n = n_lo;
  while ( n <= n_hi )
  {
    if ( id == master )
    {
      wtime = MPI::Wtime ( );
    }
    MPI::COMM_WORLD.Bcast ( &n, 1, MPI::INT, master );

    primes_part = prime_number ( n, id, p );

    MPI::COMM_WORLD.Reduce ( &primes_part, &primes, 1, MPI::INT, MPI::SUM,
      master );

    if ( id == master )
    {
      wtime = MPI::Wtime ( ) - wtime;

      cout << "  " << setw(8) << n
           << "  " << setw(8) << primes
           << "  " << setw(14) << wtime << "\n";
    }
    n = n * n_factor;
  }
  MPI::Finalize ( );

  if ( id == master )
  {
    cout << "\n";
    cout << "PRIME_MPI - Procesos maestro:\n";
    cout << "  Finalizacion del calculo normal.\n";
  }

  return 0;
}

int prime_number ( int n, int id, int p )

{
  int i;
  int j;
  int prime;
  int total;

  total = 0;

  for ( i = 2 + id; i <= n; i = i + p )
  {
    prime = 1;
    for ( j = 2; j < i; j++ )
```

```
    {
     if ( ( i % j ) == 0 )
     {
       prime = 0;
       break;
     }
    }
    total = total + prime;
  }
  return total;
}

void timestamp ( )

{
# define TIME_SIZE 40

  static char time_buffer[TIME_SIZE];
  const struct tm *tm;
  size_t len;
  time_t now;

  now = time ( NULL );
  tm = localtime ( &now );

  len = strftime ( time_buffer, TIME_SIZE, "%d %B %Y
%I:%M:%S %p", tm );

  cout << time_buffer << "\n";

  return;
# undef TIME_SIZE
}
```

VIII. REFERENCIAS

[1] Universidad Rey Juan Carlos, «Centro de Apoyo Tecnológico,» [En línea]. Available: http://www.urjc.es/cat/hidra/. [Último acceso: 16 Octubre 2013].

[2] Rocks Cluster Organization, «Open-Source Toolkit for Real and Virtual Cluster,» [En línea]. Available: http://central6.rocksclusters.org/roll-documentation/base/6.1/roll-base-usersguide.pdf. [Último acceso: 29 Noviembre 2013].

[3] J. Burkardt, «Count Primes Using OpenMP,» [En línea]. Available: http://people.sc.fsu.edu/~%20jburkardt/c_src/prime_openmp/prime_openmp.html. [Último acceso: 29 Noviembre 2013].

Autores:

Alexis Israel Arellano nació el 21 de Julio de 1989 en la ciudad de Quito, Ecuador. Realizo sus estudios secundarios en la Academia Militar Borja 3 obteniendo el título de Bachiller con especialidad Físico Matemático. Actualmente es egresado de la Escuela Politécnica del Ejercito de la Carrera de Ing. Electrónica en Redes y Comunicación de Datos y se encuentra realizando el proyecto previo a la obtención del título de ingeniería.

Jose Fernandez Molina nació el 26 de enero de 1989 en la ciudad de Quito, Ecuador. Obtiene a la edad de 18 años los títulos de Bachiller con especialidad Físico – Matemático y Auxiliar en programación de sistemas en el Colegio Particular "Jesús de Nazareth", luego ingresa a la Escuela Politécnica del Ejército en donde se encuentra cursando el último nivel de la Carrera de Ingeniería en Electrónica en Redes y Comunicación de Datos además de realizar el proyecto previo a la obtención del título de ingeniería.

alexisra@_17hotmail.com; jefernandez89@outlook.com