

**Trabajo practico de Sistemas Operativos**

**Universidad Nacional de Lanús**

**UNLa Año 2021**

**Integrantes:**

**Ariel Nicolás Heredia DNI: 33691227**

**Quimey Emmanuel Pérez DNI: 41237009**

**Oscar Ruina DNI: 35639606**

**Nicolás Romero Ortiz DNI: 40136661**

**Luciano Moliterno DNI: 40238958**

Utilizamos este Trabajo Final de Grado para realizar el trabajo práctico:

<https://riunet.upv.es/bitstream/handle/10251/79427/TORRES%20-%20Configuraci%C3%B3n%20y%20an%C3%A1lisis%20de%20las%20prestaciones%20de%20un%20sistema%20multicomputador%20para%20la%20ejecu....pdf?sequence=1&isAllowed=y>

Desde la unidad 4 Página 80 empieza el tema de pruebas.

Utilizamos el MPI(Message Passing Interface) para escribir programas basados en el pase de mensajes en entornos distribuidos.

Leímos y pensamos utilizar el jmeter para obtener las métricas resultantes de las pruebas:

<https://jmeter.apache.org/download_jmeter.cgi> , el paso a paso para configurarlo e instalarlo:

<https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.html>.

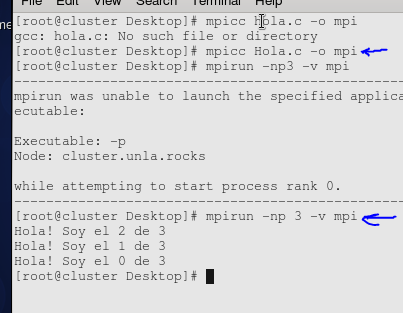
**Problemas en el entorno:**

Al principio tuvimos errores que no nos detectaba el MPI, en las cuales la información escasas por internet estuvimos dando vueltas buscando la solucion, despues tuvimos problemas con el codificado de codigo ya que el sistema operativo tiene un software obsoleto haciéndose incompatible con la mayoría de software actuales, causando inconvenientes y problemas a la hora de querer avanzar y hacer simulaciones, después de unos problemas con los drivers en el equipo, al ejecutar el escritorio virtual empezó a tirar ese mensaje de error, y no encontramos solución para recuperarlos.

Al querer ejecutarlo, tiraba el siguiente error.



Pudimos después de mucho investigar configurar para que funcione el Hello MPI! del Trabajo Final antes mencionado(página 84 apartado 4.7)



Sin embargo debido a los problemas mencionados anteriormente nos fue imposible poder realizar las pruebas que teníamos pensadas, por eso los resultados esperados no van a poder ser verificados sino que serán en base al conocimiento que recibimos de las clases teóricas de la materia.

**Prueba número 1: Prueba de velocidad y resultados**

**Objetivo:** Ejecutar un programa y ver el resultado y el tiempo de ejecución.

**Descripción:** Probar que la ejecución de un mismo programa con todos los nodos(3, un nodo maestro y dos esclavos), con dos nodos, con un nodo, devuelva el mismo resultado y que la velocidad de ejecución sea mas rapida dependiendo de cuantos nodos estemos usando.

**Resultados:** Debido a los problemas mencionados anteriormente no pudimos hacer las pruebas, sin embargo, debido a lo visto en clase asumimos que los resultados esperados son que independientemente de cuántos nodos estemos usando el resultado de la ejecución sea el mismo no así la velocidad de ejecución, a más nodos más velocidad.

**Prueba número 2: Números primos**

**Objetivo:** Probar el rendimiento del sistema.

**Descripción:** PRIME\_NUMBER que incluye llamadas MPI para procesamiento paralelo, esto usará el el 100% del procesador para resolver la cantidad de números.PRIME\_NUMBER devuelve el número de primos entre 1 y N. Para dividir el trabajo uniformemente entre los procesadores P, el ID del procesador comienza en 2 + ID y salta por P. Se usa un algoritmo donde Mathematica puede devolver el número de primos menores o iguales a N mediante el comando PrimePi [N].

**Resultados:** Pensamos que después de realizar la ejecución del programa en diferentes tamaños y variando el número de nodos que intervienen la performance mejora si tenemos 3 nodos trabajando en lugar de dos o solo uno.

**Prueba número 3: Spike Testing o Prueba de Picos**

**Objetivo:** Observar el comportamiento del sistema al variar la cantidad de nodos.

**Descripción:** Por medio de esta prueba se intentó simular el funcionamiento de un servidor web,por la cual una cantidad N de usuarios intentara realizar peticiones continuas al mismo.

**Resultados**: Se espera del mismo un aumento del caudal de peticiones, a la cual el sistema operativo asigna un nuevo thread a cada una de ellas.

Se espera percibir una leve disminución en el tiempo de respuesta , que irá disminuyendo a medida que los usuarios se asignen a los distintos nodos que componen el sistema distribuido.

**Prueba número 4: Distribución de Carga**

**Objetivo:** Probar una cantidad fija de procesos de cálculo de matrices con migración y sin migración.

**Descripción:** Elegimos usar 6 procesos. Cada proceso calcula la multiplicación de dos matrices de 100X100 cada una, primero usando un solo nodo y sin la posibilidad de migrar, después usando los 3 nodos disponibles. Las políticas que pensamos para hacer la migración son: política de transferencia: balanceo de carga, la política de selección: migrar procesos recién creados, política de ubicación: migrar proceso al nodo que tenga menos procesos en su cola.

**Resultados:** Pensamos que al no usar la distribución de carga la velocidad para terminar la ejecución de todos los procesos es más alta que usando la distribución de carga, ya que permite terminar más rápido la ejecución de todos los procesos.

**Informe ROCKS Cluster:**

Es una distribución de Linux para clusters de alto rendimiento. Fue iniciada por la NPCI(National Partnership for Advanced Computational Infrastructure) y la SDCC(San Diego Supercomputer Center) en el año 2000. Se basó inicialmente en la distribución Red Hat Linux, sin embargo las versiones más modernas están basadas en CentOS, con un instalador Anaconda modificado, que simplifica la instalación “en masa” en varios computadores. Incluye varias herramientas como MPI(message passing interface) que no forman parte de CentOS pero son los componentes integrales que hacen un grupo integrado de cluster.

Las instalaciones pueden ser personalizadas con paquetes de software adicionales, utilizando CD especiales(llamados Roll CD).Los Roll extienden el sistema integrando automáticamente los mecanismos de gestión y empaquetamiento usados por el software base, simplificando la instalación y configuración de un gran número de computadores.

Es una de las distribuciones más empleadas en el ámbito de clusters, por su facilidad de instalación e incorporación de nuevos nodos. Otras de sus facilidades es que incorpora gran cantidad de software para el mantenimiento y monitorización del cluster.

Rocks está diseñado para ayudar a los científicos con poca o ninguna experiencia en clusters a construir supercomputadoras basadas en Linux que sean compatibles con los sistemas utilizados por los centros informáticos nacionales y las redes internacionales.

Con el mínimo de dos computadoras, Rocks permite una fácil implementación y gestión de los clusters.