Tutorial
Introducción a la Programación Paralela en C con el Patrón de Diseño *Fork-Join* y la biblioteca MPI

Miguel Angel Astor R. y Ana Morales Bezeira
Universidad Central de Venezuela, Facultad de Ciencias
Escuela de Computación, Centro CICORE, Laboratorio ICARO
Valle Abajo, Caracas, 1043. Venezuela
Email: miguel.astor@ciens.ucv.ve, ana.morales@ciens.ucv.ve

***Resumen*—En este Taller se presentarán las nociones fundamentales para el desarrollo de aplicaciones paralelas basadas en el paradigma de paso de mensajes, utilizando la biblioteca MPI. Como complemento, se presentará el patrón de diseño para aplicaciones paralelas llamado *fork-join*, el cual se utiliza para diseñar aplicaciones paralelas bajo un enfoque divide-y-vencerás. El enfoque del tutorial es eminentemente práctico, y durante el desarrollo del mismo los participantes desarrollarán una aplicación en el lenguaje de programación C, utilizando las principales herramientas de OpenMPI, la cual es una implementación libre de MPI.**

Palabras clave*—*Paso de mensajes, MPI, OpenMPI, fork-join, divide-y-vencerás, C.

I. Introducción

El patrón de diseño *fork-join* es un patrón de diseño clásico utilizado para diseñar aplicaciones paralelas cuando el problema a resolver por la aplicación puede dividirse en múltiples partes las cuales pueden ser tratadas de forma independiente. Es una aplicación paralela del modelo de desarollo algorítmico divide-y-vencerás [1], [2]. Este patrón de diseño tiene aplicaciones limitadas en lo referente a la solución de problemas con paralelismo; sin embargo, su sencillez conceptual y de implementación permite que sirva de herramienta didactica para introducir herramientas y paradigmas de programación paralela, como lo es por ejemplo el paradigma de comunicación de paso de mensajes.

MPI (*Message Passing Interface* - Interfaz de Paso de Mensajes) es la especificación de una biblioteca estándar para desarrollar sistemas basados en el paradigma de comunicación de paso de mensajes. La especificación de MPI define un modelo de gestión y comunicación de procesos, los cuales se apoyan en una serie de funciones definidas para los lenguajes C/C++ y Fortran [3]. Por si sola, MPI es únicamente una especificación. Actualmente existen en el mercado una serie de implementaciones de esta interfaz, siendo OpenMPI una de las más utilizadas.

OpenMPI[1] es una implementación libre de la interfaz especificada por MPI, desarrollada en el año 2004 como resultado de la integración de varias implementaciones anteriores [4]. Esta implementación incluye soporte oficial para los lenguajes C/C++ y Fortran, aunque existen múltiples interfaces que permiten utilizar la biblioteca con otros lenguajes como MATLAB, GNU Octave, Python y Java. OpenMPI es ampliamente utilizada por la comunidad científica y es la biblioteca de cómputo paralelo preferida por la amplia mayoría de las supercomputadoras del TOP-500 [5].

El presente taller pretende introducir a los participantes en el uso de la biblioteca MPI para el desarrollo de aplicaciones paralelas basadas en el patrón de diseño *fork-join*, utilizando como lenguaje de programación al lenguaje C. Sin embargo, siendo MPI una especificación estandar y *fork-join* un patrón de diseño general independiente del lenguaje, los conocimientos adquiridos durante el taller son facilmente aplicables a otros lenguajes o herramientas de programación paralela.

II. Objetivos

A continuación se presentan los objetivos general y específicos del Tutorial planteado.

II-A. Objetivo general

Introducir y prácticar los conceptos fundamentales del diseño de aplicaciones paralelas siguiendo el patrón de diseño *fork-join*, utilizando el lenguaje de programación C con la biblioteca OpenMPI como herramientas de programación.

II-B. Objetivos específicos

- Introducir el paradigma de comunicación de paso de mensajes.
- Introducir el patrón de diseño para aplicaciones paralelas *fork-join*.
- Practicar el uso de la biblioteca MPI con el lenguaje de programación C.
- Desarrollar un programa sencillo bajo el patrón *fork-join* con el lenguaje de programación C utilizando OpenMPI.

[1] https://www.open-mpi.org

III. Contenidos

Durante el Taller se desarrollará el siguiente contenido temático:

III-A. Introducción al paso de mensajes y fork-join

Mecanismos de comunicación entre procesos. Comunicación por paso de mensajes. Paso de mensajes bloqueante y no bloqueante. El patrón de diseño *fork-join*. Introducción a MPI.

III-B. Fundamentos de MPI

Funciones fundamentales: `MPI_Init`, `MPI_Comm_size`, `MPI_Comm_rank`, `MPI_Send`, `MPI_Recv` y `MPI_Finalize`. Comandos de MPI: `mpicc` y `mpirun`.

III-C. Aplicaciones paralelas con MPI

Compilación y ejecución de programas de MPI. Separación del programa en procesos maestro y esclavos. Paso de mensajes. Cálculo y envío de resultados. Finalización de programas. Manejo de errores de MPI.

III-D. Tópicos avanzados de MPI

Broadcast. *Reduce*. Tipos de datos derivados. Sincronización.

III-E. Solución de problemas clásicos con fork-join

Aplicaciones en análisis numérico. Algoritmos genéticos. *Ray-tracing* distribuido. Aplicaciones en procesamiento digital de imágenes.

IV. Metodología

La metodología del Taller está basada en una presentación de conocimiento teórico apoyada por una serie de ejercicios prácticos de programación.

El Taller se compone de una única sesión de 4 horas. Esta sesión se divide a su vez en tres partes. La primera parte consistirá en una presentación de contenido teórico en la cual se presentará el punto A del contenido temático. Esta primera parte tendrá una duración de 30 a 45 minutos. La segunda parte compone el grueso del taller y consistirá en varios ejercicios prácticos sobre la biblioteca MPI, el primero de los cuales será el clásico ejercicio del *hello world*, culminando con un ejercicio real con aplicaciones en análisis numérico, como lo es el cálculo numérico de integrales utilizando la regla del trapecio.

Los úlitmos 10 a 15 minutos del taller consistiran en una breve presentación sobre el uso del patrón *fork-join* para solucionar problemas clásicos de Ciencias de la Computación.

V. Duración

Se estima una duración de 4 horas para el Taller (medio día).

VI. Público objetivo

Personas interesadas en conocer y utilizar herramientas de programación paralela.

Los participantes deben tener un dominio intermedio del lenguaje C en ambientes Linux. No se necesita tener conocimientos previos con OpenMPI para desarrollar el taller.

Los participantes deben tener buen dominio de la terminología básica de Sistemas Operativos (no limitativo).

VII. Requerimientos

Para poder llevar a cabo el tutorial se requiere lo siguiente:

- Un proyector.
- Acceso a Internet en la sala.
- Computadoras donde puedan trabajar hasta dos personas por computadora, las cuales deben poseer lo siguiente:
  - Sistema operativo basado en Linux.
  - El compilador GCC.
  - Editor de texto con resaltado de sintaxis.
  - La biblioteca OpenMPI (paquetes `openmpi-bin`, `libopenmpi1.X` y `libopenmpi-dev` en sistemas basados en Debian).

VIII. Cupo

Se espera poder dar cabida a por lo menos 26 participantes, por supuesto aunado a las limitaciones físicas del espacio donde se dicte el taller.

IX. Idioma

La exposición, dictado, ejercicios y ejemplos serán dados en español.

Referencias

[1] M. E. Conway, "A multiprocessor system design," in *Proceedings of the November 12-14, 1963, fall joint computer conference*. ACM, 1963, pp. 139–146.

[2] M. D. McCool, A. D. Robison, y J. Reinders, *Structured parallel programming: patterns for efficient computation*. Elsevier, 2012.

[3] M. Schul, *MPI: A Message-Passing Interface Standard Version 3.1*, Message Passing Interface Forum, junio 2015.

[4] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, y T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[5] G. Bosilca, J. Squyres, y N. Hjelm, "Supercomputing 2015 slides," in *The International Conference for High Performance Computing, Networking, Storage, and Analysis, Super Computing SC15: HPC Transforms, BoF Sessions*, noviembre 2015, pp. 139–146.

PRESENTADORES

Miguel Ángel Astor Romero

Profesor investigador, categoría Instructor, de la Escuela de Computación, de la Universidad Central de Venezuela (UCV). Es Licenciado en Computación de la Universidad Central de Venezuela (2014). Actualmente cursa estudios de Maestría en Ciencias de la Computación en la UCV bajo la tutoría del prof. Wilmer Pereira Gonzales.

Dirección: Paseo los Ilustres, Urb. Valle Abajo. Facultad de Ciencias, Cdad. Universitaria de Caracas, Venezuela.

Tlf: +58-212-6051170. Correo: miguel.astor@ciens.ucv.ve

Ana Verónica Morales Bezeira

Profesora investigadora, categoría Agregado, de la Escuela de Computación, de la Universidad Central de Venezuela (UCV). Es Licenciada en Computación de la Universidad del Zulia (1999), M.Sc. en Telemática (2002) y candidata a Doctora en Ciencias de la Computación por la UCV.

Dirección: Paseo los Ilustres, Urb. Valle Abajo. Facultad de Ciencias, Cdad. Universitaria de Caracas, Venezuela.

Tlf: +58-212-6051329. Correo: ana.morales@ciens.ucv.ve