DPDK y Openflow-OKO

Trabajo Final de Estudios

Enrique Fernández Sánchez

Revisión 19 Febrero 2021

Índice

1 Introduction

2 DPDK

2.1 ¿Qué es?

DPDK (*Data Plane Development Kit*) es un proyecto Open Source controlado por la Fundación Linux. Dicho proyecto tiene por objetivo proveer unas librerías de "plano de datosz controladores para comunicarse con los propios controladores de las interfaces de red, con la finalidad de descargar el procesamiento de los paquetes TCP/UDP desde el núcleo del sistema operativo a los diferentes procesos que se ejecutan en el espacio de usuario (aplicaciones). Con esta descarga, lo que conseguimos es liberar al procesador de ciertas tareas especificas, dando por resultado una mayor eficiencia informática y un mayor rendimiento de paquetes, ya que introducimos mejoras a las propias instrucciones que nos proporciona el kernel.

Data Plane Development Kit surge a una alternativa competitiva para el procesamiento de paquetes de red. Si bien es cierto que actualmente el procesado que hace Linux de los paquetes es bastante eficiente, con DPDK podemos llevar a incrementar muchísimo la velocidad de procesado. Esto es muy importante ya que de cara a tecnologías futuras, comunicaciones de fibra óptica o el 5G, es muy necesario que los servidores del backbone, o aplicaciones específicas de routers y switches, puedan responder a las grandes necesidades que se nos proponen.

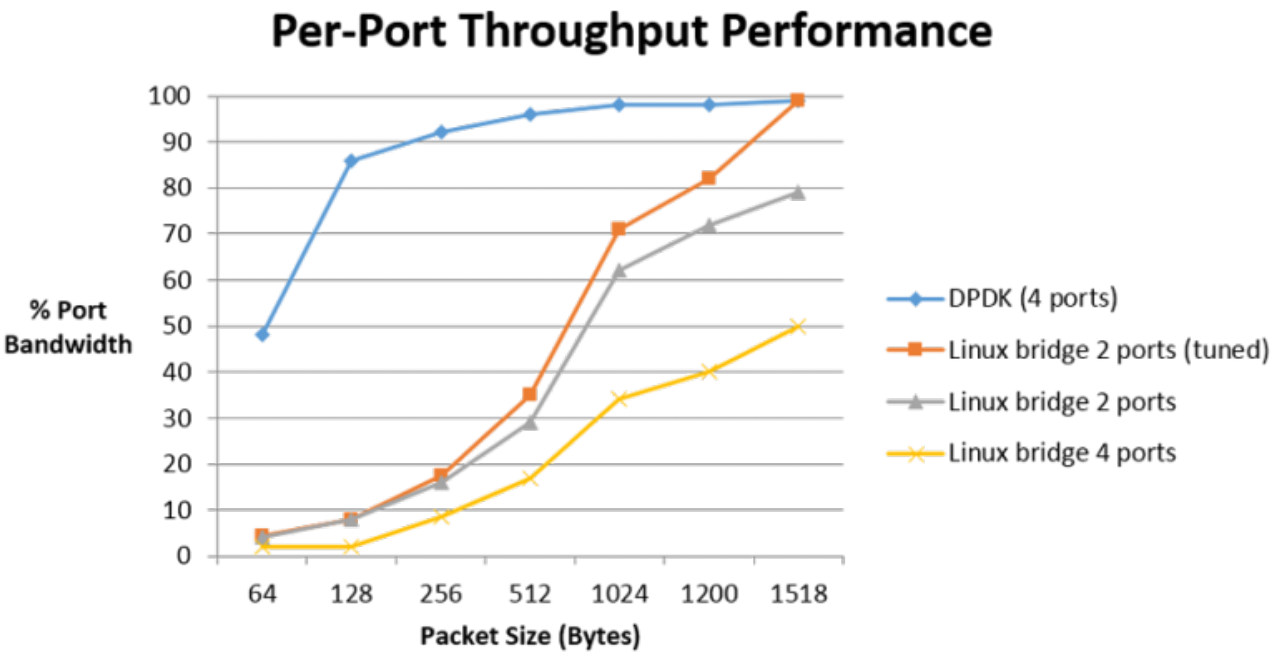

Figura 1: Comparación entre Linux y DPDK. (**??**)

2.2 Requisitos para utilizar DPDK.

2.3 ¿Qué son las *huge pages*?

Para entender correctamente lo que son las *huge pages*, primero tenemos que profundizar en el concepto de *pages*. Cuando un proceso utiliza algo de memoria, la CPU marca la RAM como que está siendo utilizada por dicho proceso. Para aumentar la eficiencia, la CPU asigna la RAM en "porciones"(*chunks*) de 4K bytes (valor *default* en la mayoría de plataformas). Estas porciones se les llama *pages*.

Como el espacio de direccionamiento del proceso es virtual, la CPU y el SO tienen que recordar que páginas pertenecen a que procesos, y además, donde se almacenan. Por consecuencia, cuantas más páginas tengas, más tiempo tardarás en encontrar donde está la memoria alojada. En comparativa, si un proceso utiliza 1GB de memoria, haciendo la cuenta, serían 262144 entradas a comprobar (1GB/4K). Sin embargo, si una *Page Table Entry* utiliza 8 *bytes*, serían 262144 * 8 entradas a comprobar.

La mayoría de CPU actuales soportan páginas de mayor tamaño (por lo que la CPU tiene muchas menos entradas para comprobar), estas pueden recibir el nombre de *Huge pages* (en Linux), *Super pages* (en BSD) o *Larger pages* (en Windows), aunque todas son lo mismo.

2.3.1 Cómo configuramos las *huge pages* para DPDK.

Según la documentación oficial de *DPDK: Quick Start*, tenemos que realizar una intervención manual en la instalación de DPDK, en relación a las *huge pages*. Necesitamos reservar"las *huge pages*, lo haríamos tal que:

```
mkdir -p /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
echo 64 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
```

3 Bibliografía

Reference of bibliography

1. Introduction to DPDK: Architecture and Principles
2. Wikipedia: Data Plane Development Kit
3. Imagen comparativa de BW en diferentes tests case.
4. Hugepages
5. DPDK: Quick Start