FAWN: A Fast Array of Wimpy Nodes

Kevin Hernández Rostrán

20 de Junio, 2020

¿Qué es FAWN?

Arquitectura de clúster para computación de datos intensiva y de **bajo consumo de energía** que combina nodos integrados de baja potencia con **almacenamiento flash** para proporcionar un procesamiento **rápido** y **energéticamente eficiente** de cargas de trabajo aleatorias e intensivas en lectura.

Tendencias fundamentales

- Utilizar procesadores de E/S débiles —*"wimpy"*— para minimizar la brecha que existe en los CPU convencionales.
- Utilizan 1/3 de la frecuencia para ejecutar más de mil millones de instrucciones por Joule.

Arquitectura FAWN-KV

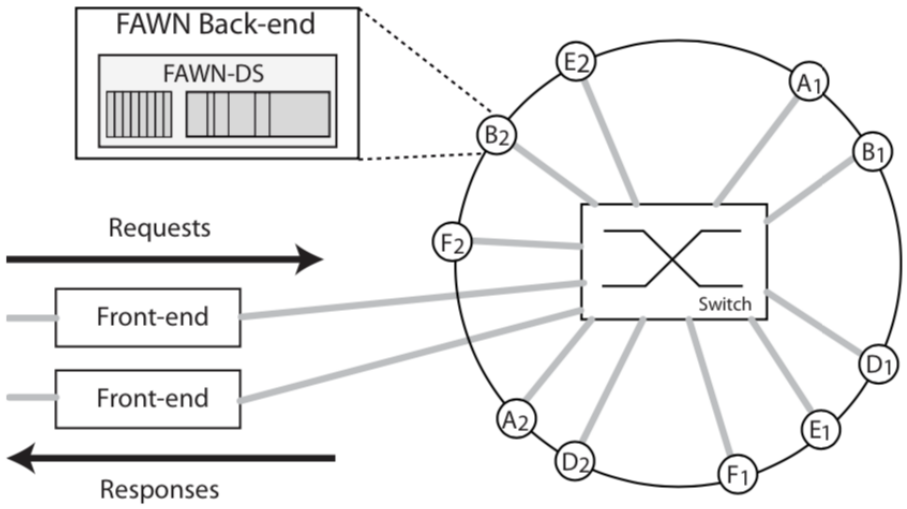

Figura1: Arquitectura FAWN-KV

Arquitectura FAWN-KV —cont.

FAWN-DS. Está diseñado específicamente para funcionar bien con el almacenamiento flash y para operar dentro de la DRAM restringida disponible en nodos débiles *"wimpy"*. Utiliza un índice de hash en memoria (DRAM) para asignar llaves de 160 bits a un valor almacenado en el registro de datos. Soporta operaciones *"Store"*, *"Lookup"* y *"Delete"*.

Arquitectura FAWN-KV —cont.

FAWN-KV. Las aplicaciones cliente envían solicitudes a los *front-end* utilizando una interfaz estándar de *"put/get"*. Los *front-end* envían la solicitud al nodo de *back-end* que posee la llave del espacio para la solicitud. El nodo de *back-end* satisface la solicitud utilizando su **FAWN-DS** y responde a los servicios de *back-end*.

Arquitectura FAWN-KV —cont.

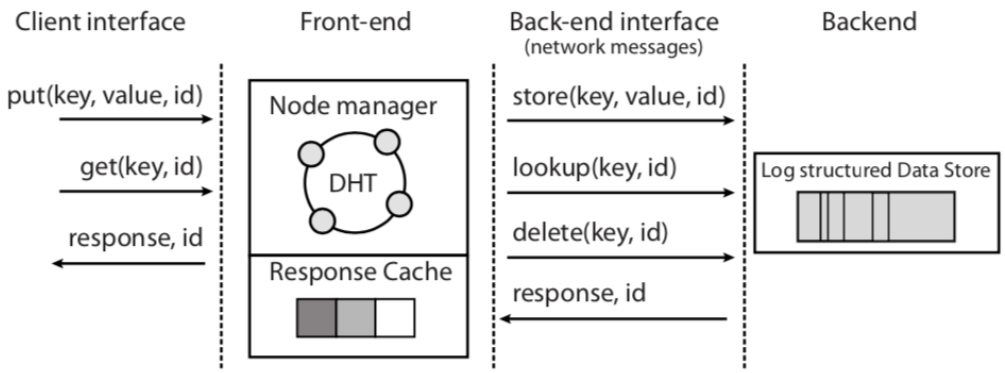

Figura2: Interfaces FAWN-KV

Arquitectura FAWN-KV —cont.

- Las solicitudes de los clientes ingresan al sistema en uno de los varios *front-end*.
- Los nodos *front-end* envían la solicitud al nodo **FAWN-KV** —sistema de llave-valor *"key-value"*— *back-end* responsable de servir esa llave en particular.

Arquitectura FAWN-KV —cont.

- El nodo *back-end* atiende la solicitud de su almacén de datos **FAWN-DS** y devuelve el resultado al *front-end* (que a su vez responde al cliente).
- Las escrituras proceden de manera similar.

Replicación y consistencia

- FAWN-KV utiliza la replicación en cadena para proporcionar una fuerte consistencia por llave.
- Las actualizaciones se envían a la cabeza de la cadena, se pasan a cada miembro de la cadena a través de una conexión TCP entre los nodos, y las consultas se envían a la cola de la cadena.

Replicación y consistencia —cont.

Ciclo de vida de una operación *"put"* con replicación de cadena

- El front-end dirige la operación *"put"* al sucesor de la llave, A1, que es el *"head"* de la cadena de réplicas para este rango.
- Después de almacenar el valor en su almacén de datos, A1 reenvía esta solicitud a B1, que almacena el valor de manera similar y reenvía la solicitud a la cola, C1.
- Después de almacenar el valor, C1 envía la respuesta *"put"* al front-end y envía un acuse de recibo de la cadena indicando que la respuesta se manejó correctamente.

Replicación y consistencia —cont.

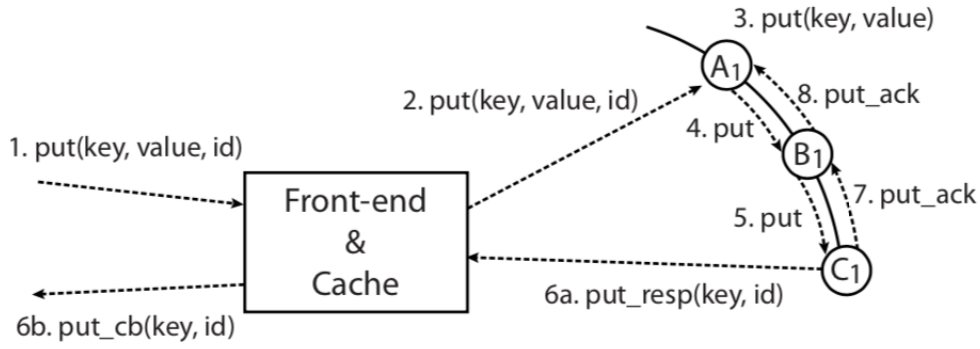

Figura3: Ciclo de vida de una operación *"put"* con replicación de cadena

Blue Gene/L torus interconnection network

Kevin Hernández Rostrán

20 de Junio, 2020

¿Qué es una red de interconexión “Torus”?

La interconexión *“Torus”* es una topología sin-switch que puede verse como una interconexión de malla con:

- Nodos dispuestos en una matriz rectilínea de 2, 3 o más dimensiones.
- Procesadores conectados a sus vecinos más cercanos y procesadores correspondientes en los bordes opuestos de la matriz conectada.

¿Qué es una red de interconexión “Torus”? —cont.

- Las redes “Torus” se utilizan con frecuencia en supercomputadoras de alto rendimiento.
- En la mayoría de arquitecturas los nodos de cómputo intercambian datos con sus vecinos más cercanos.

La red Blue Gene/L

- En **Blue Gene/L** (*BG/L*), la interconexión principal para la mensajería punto a punto es una *red* ***"Torus"*** *tridimensional (3D) con enrutamiento virtual dinámico de corte*.
- El enrutador de la red torus dirige paquetes de tamaño variable, cada nx32 bytes, donde n = 1 a 8 "fragmentos".

La red Blue Gene/L —cont.

- Los mensajes, como los que se ajustan a la interfaz “Message Passing Interface Standard” (MPI), pueden consistir en muchos paquetes que se construyen, envían y reciben por software que se ejecuta en uno o ambos procesadores BG/L asociados.

La red Blue Gene/L —cont.

- Los primeros ocho bytes de cada paquete contienen:
  - Información de protocolo de "link-level" (por ejemplo, número de secuencia).
  - Información de enrutamiento, incluyendo el destino.
  - Información del canal virtual y del tamaño.
  - Y una comprobación de redundancia cíclica —"cyclic redundancy check"— (CRC) de bytes[1].

[1] Esta detecta la corrupción de datos del encabezado durante la transmisión.

La red Blue Gene/L —cont.

El protocolo de detección y recuperación de errores es similar al utilizado en las redes de interconexión de el IBM High Performance Switch (HPS) y en el estándar HIPPI-6400[2].

[2] ver https://tools.ietf.org/html/rfc2835

Estrutura general de un router “Torus”

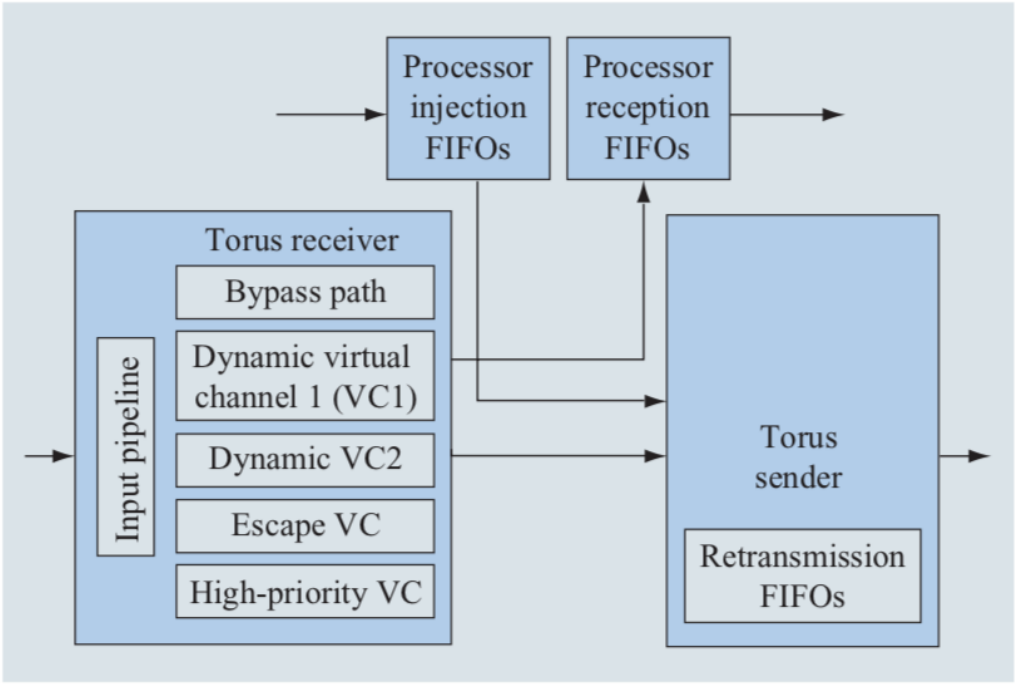

Figura1: Estrutura general de un router “Torus”

Estrutura general de un router “Torus” —cont.

La lógica de torus consta de tres unidades principales:

- **Interfaz del procesador**. Consta de colas FIFO de inyección y recepción en red. Las colas se dividen en grupos:
  - Dos colas de alta prioridad (para mensajes del sistema operativo intre-nodo).
  - Seis colas FIFO de prioridad normal.

  que son suficientes para la conectividad del vecino más cercano.

Estrutura general de un router “Torus” —cont.

- **Unidad de recepción**. Acá también hay grupos de FIFO, cada grupo contiene siete FIFO, uno de alta prioridad y uno dedicado a cada una de las direcciones entrantes.

Estrutura general de un router “Torus” —cont.

- **Unidad de envío**. Se encarga de la comunicación de su nodo adjunto.

Estrutura general de un router “Torus” —cont.

Para el enrutamiento de los datos cada uno de los receptores se compone de una tubería de entrada de ocho etapas, cuatro canales virtuales (VC) y un canal de derivación.