Procesamiento de datos masivos

Jesús Morán

viu | Universidad Internacional de Valencia

24.11.21

- Patrones de ordenación
- Patrones para añadir datos
- Patrones de resúmenes estadísticos
- Patrones de filtrado
- Patrones de optimización
- Patrones de unión
- Patrones de estructuración del programa

- Ordenación:
  - Valores ordenados en Reduce -> Secondary Sort:
    - Map debe emitir <{clave, valor}, valor>
    - Partitioner debe particionar por clave (no por {clave, valor})
    - Sort ordena primero por clave y luego por valor
    - Group agrupa por clave

- Ordenación:
  - Claves ordenadas en todos los Reducer -> Total Sort
    - Notar que Hadoop ordena por defecto en cada Reducer las claves, pero en varios Reducer no están ordenadas
    - InputSampler: hace un muestreo y obtiene cómo deberían ser las particiones
    - Cache distribuida: se distribuye el archivo de particiones para que sea accesible
    - Partitioner: utiliza TotalOrderPatitioner para que las claves bajas vayan a reducer bajos, y al revés

- Añadir datos:
  - Configuration: permite añadir pares propiedad-valor y estará accesible en MapReduce
  - Caché distribuida: Permite enviar pocos datos en modo lectura
  - Otros tipos de almacenamiento: bases de datos, sistemas de archivos distribuidos, copiar archivos a todo el cluster, etc.

- Resúmenes estadísticos:
  - Map-Reduce: permite hacer agregaciones
    - Map agrupa por clave
    - Reduce hace la operación de agregación
  - Contar Outliers, problemas etc -> Contadores

- Filtrado:
  - Map: recibe un dato y puede filtrarlo
  - Mapper: puede filtrar antes de emitir
    - Map: no emite nada, sino que guarda los pares <clave, valor>
    - Clean-up: determina qué pares <clave, valor> se emiten y cuáles no
  - Top-K:
    - Driver: 1 única tarea Reducer
    - Mapper: Emite sólo k datos (top-k)
    - Reducer: Recibe k datos de cada Mapper y se queda con el top-k
  - Eliminar información similar, idéntica o redundante
    - Map: emite <clave, NullWritable> donde la clave es el dato (se pueden asignar varias claves con funciones de distancia)
    - Combiner + Reducer: Sólo emite uno de ellos
    - Puede ser ineficiente si no tenemos muchos datos que se parezcan o sean iguales

- Optimización:
  - Combiner:
    - Eliminar los datos irrelevantes para Reducer
    - Pre-procesar los datos para que a Reducer tenga menos trabajo que hacer (puede implicar cambiar las claves o los valores)
  - In-Mapper Combiner:
    - Driver: no se ejecuta Combiner
    - Map: no emite nada, sino que guarda los pares <clave, valor>
    - Clean-Up: sólo emite los pares <clave, valor> esenciales

- Unión:
  - Existen muchas técnicas dependiendo de los datasets
  - Algunos ejemplos:
    - Luo, G., & Dong, L. (2010). Adaptive join plan generation in hadoop. Duke University, USA
    - Miner, D., & Shook, A. (2012). MapReduce design patterns: building effective algorithms and analytics for Hadoop and other systems. O'Reilly Media, Inc.:, California, Estados Unidos.
  - Replicated Join:
    - Sólo cuando uno de los datasets es grande
  - Reduce side join:
    - Si los dos datasets son grandes

- Estructura del programa:
  - Map-Only: ejecuta Map, pero no Reduce
  - ChainMapper: permite concatenar la salida de un Mapper con otro Mapper
  - ChainReducer: permite concatener la salida de un Reducer con un Mapper
  - Job Merging: se ejecutan varios “job” en un único MapReduce
    - Map: emite <clave, {job1: valor}>, <clave, {job2: valor}>
    - Reducer: implementa dos funcionalidades
      - Reduce: guarda para “job” los pares <clave, valor> de salida
      - MultipleOutputs: permite guardar en diferentes archivos las salidas de cada “job”
  - MultipleInputs: varios datasets analizados con diferentes Mapper y un Reducer común

- Patrones de ordenación
- Patrones para añadir datos
- Patrones de resúmenes estadísticos
- Patrones de filtrado
- Patrones de optimización
- Patrones de unión
- Patrones de estructuración del programa

Gracias

viu | Universidad Internacional de Valencia