Unidad 2: Modelo de mapeo y reducción

Modelo costo-comunicación y teoría de la complejidad

Blanca Vázquez

Febrero 2022

Costo de un algoritmo

- De manera general, el rendimiento de un algoritmo puede ser descrito en los siguientes costos
  - Tiempo: ¿cuánto tiempo lleva en ejecutar una tarea? *entre menos tiempo es mejor*
  - Espacio: ¿cuánta memoria usa? *entre menos espacio es mejor*
  - Energía: ¿cuánta energía usa? *entre menos energía es mejor*

Funciones para calcular costos

- Para un algoritmo dado, el objetivo es encontrar las siguientes funciones:
  - $T(n)$: el costo del tiempo para resolver el problema
  - $S(n)$: el costo del espacio para resolver el problema
  - $E(n)$: el costo de la energía para resolver el problema

Costos en el modelo de mapeo y reducción (1)

1. Costo de cómputo
   - Trabajos de mapeo
   - Trabajos de reducción
2. Costo de comunicación
   - Transmisión de los pares llave-valor de los nodos de mapeo a los de reducción.
   - En las tareas de mapeo usualmente no se incurre en costos de comunicación (se busca llevarlas a cabo en el mismo nodo donde se encuentran los datos, aunque se puede incurrir en costos de lectura de los datos de disco).

Costos en el modelo de mapeo y reducción (2)

- En un escenario común
  - El costo de comunicación domina al de cómputo
  - El costo de los trabajos de mapeo es una pequeña fracción del costo de comunicación
  - El costo que incurre el sistema por el ordenamiento es proporcional al costo de comunicación

- En los servicios de la nube (por ej. EC2) se paga tanto por el cómputo como por la comunicación, por lo que es importante tener un balance de ambos en el diseño de los algoritmos

Modelo de costo - comunicación

En MapReduce el costo de un algoritmo se calcula de la siguiente manera:

- Costo de comunicación
- Costo de comunicación transcurrido
- Costo de computación transcurrido

Costo de comunicación

- Costo de comunicación = la suma del tamaño de todos los archivos pasados del proceso de mapeo hacia el proceso de reducción + la suma del tamaño de todos los archivos de salida en los procesos de reducción.

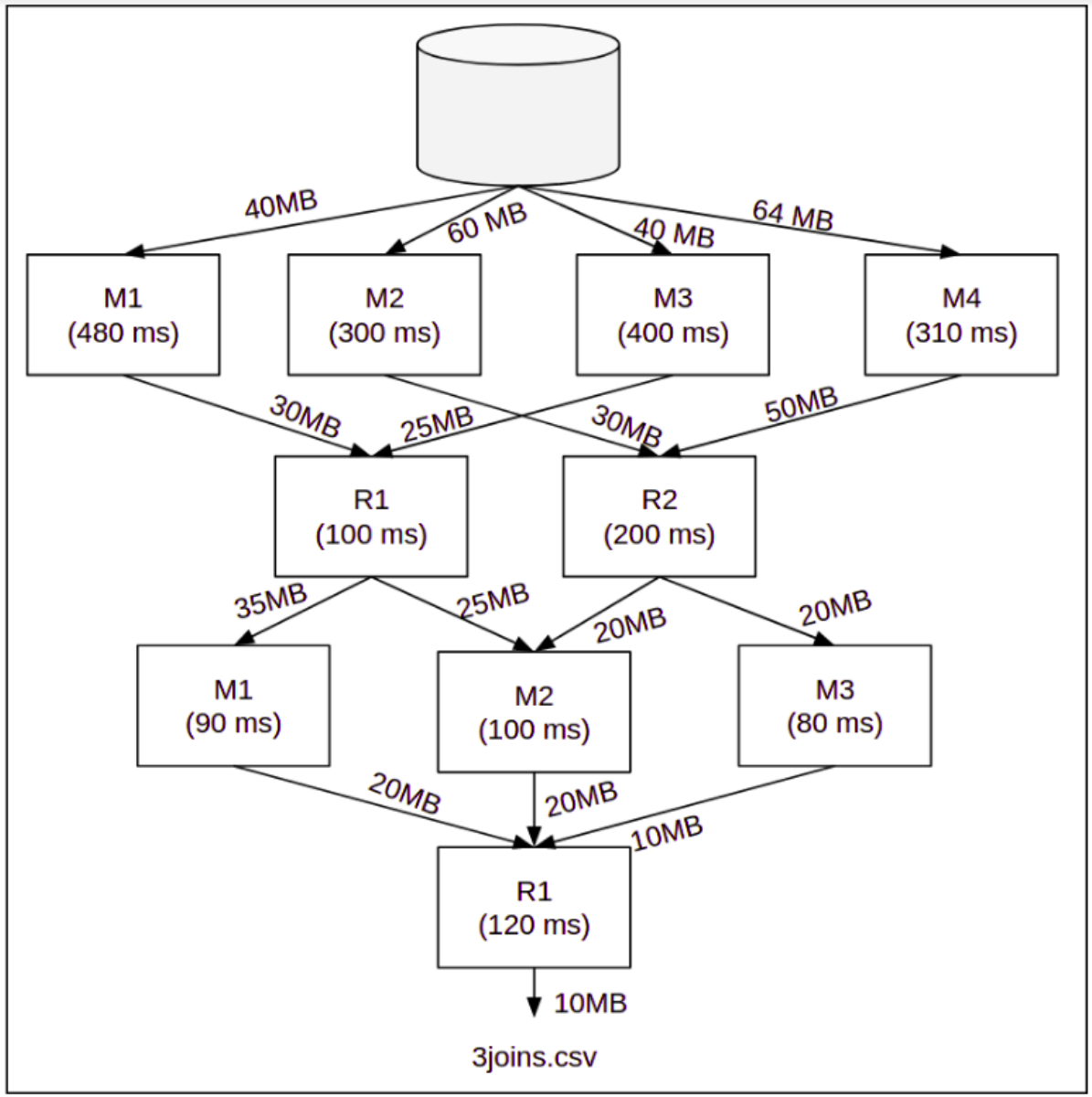
Costo de comunicación transcurrido

- Costo de comunicación transcurrido = es la suma de la salida y de la entrada más grande para el proceso de mapeo + la suma de la salida y de la entrada más grande para el proceso de reducción.

Costo de computación transcurrido

- Costo de computación transcurrido = suma únicamente el tiempo de ejecución de los procesos.

Ejercicio: modelo costo comunicación



Modelo de costo de comunicación

- El costo de comunicación de un algoritmo es la suma de los costos de todas las tareas que requiere
- La meta es medir la eficiencia de un algoritmo basado en su costo de comunicación
  - El cómputo que realiza cada tarea es usualmente muy simple y con complejidad lineal en el tamaño de la entrada.
  - Transmitir una entrada sobre la red puede ser muy lento y requerir tiempo par poder recibirla de otros nodos debido al tráfico en la red.

Costos de algoritmos

- El costo de un algoritmo depende de varias características
  - La semántica del modelo de programación
  - La topología de la red
  - El tipo de datos
  - Protocolos de comunicación

Problema de ejemplo: unión de dos relaciones

- Problema: $R(A, B) \bowtie S(B, C)$
  - Las tareas de mapeo tienen fragmentos de cualquier de las dos
- Función de mapeo
  - Recibe tupla t de R o S
  - Produce par llave-valor
- Función de reducción
  - Recibe llave con lista de valores, la cual puede contener la tupla de R, la de S o ambas
  - Identifica tuplas que son de R de aquellas que son de S y las empareja
  - Produce par (t, t)

Costo de comunicación de la unión de dos relaciones: mapeo

- Supongamos que los tamaños de R y S son r y s
- La entrada a las tareas de mapeo es un fragmento de los archivos que contienen a R o a S
- La suma del costo de comunicación de cada tarea de mapeo es $r + s$
- El costo de comunicación del mapeo es $O(r + s)$

Costo de comunicación de la unión de dos relaciones: reducción

- La reducción se aplica a uno o más atributos
- Cada reductor toma su entrada y la divide entre las tuplas que vienen de R y las que vienen de S.
- Cada tupla de R se empareja con cada tupla de S para producir una salida.
- El tamaño de la salida puede ser mayor o menor a $r + s$, dependiendo de qué tanto se unan las tuplas.

Tiempo de reloj

- Al diseñar un algoritmo para un clúster es importante también tomar en cuenta el tiempo que toma un algoritmo en ejecutarse en paralelo.

Unidad 2: Modelo de mapeo y reducción

Teoría de la complejidad para el modelo de mapeo y reducción

Teoría de la complejidad para el modelo MapReduce

- En MapReduce, el paralelismo se generada debido a que cada tarea de mapeo y reducción se ejecutan al mismo tiempo en diferentes nodos.
- El programador define el número de nodos para cada tarea.
- El modelo se encarga de planificar los pasos de mapeo, agrupación, reducción, rutas de los datos y tolerancia a fallos.

Rounds en MapReduce

- Un *round* consiste de las tareas de mapear - agrupar - reducir.
- La salida de una tarea de reducción puede ser la entrada a otro round.
- En aplicaciones reales, MapReduce es una secuencia de tareas que se ejecutan a través de *rounds*.
- Idealmente se espera que el número de *rouds* se una constante.

Ejemplo: una consulta triangular

- Input: three tables
  R(X, Y), S(Y, Z), T(Z, X)

  size(R)+size(S)+size(T)=M

- Output: compute

  $Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x)$

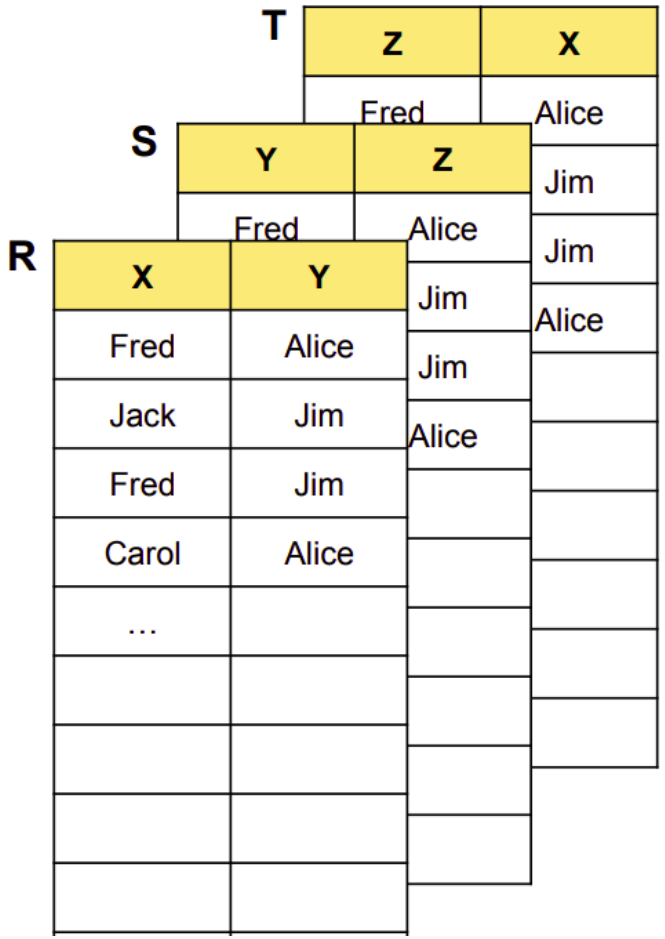


Imagen tomada de Dan Suciu, University of Washington

M = tamaño de datos de entrada en bits

Haciendo la unión en dos pasos

- Paso 1: calcular primera unión (almacenarlo en *temp*)
  $temp(X, Y, Z) = R(X, Y) \bowtie S(Y, Z)$

- Paso 2: calcular un segunda unión
  $Q(X, Y, Z) = temp(X, Y, Z) \bowtie T(Z, X)$

Nota:

- *temp* puede generar largas relaciones intermedias
- El tiempo de comunicación y computación transcurrido puede verse afectadas (dependerán del tamaño de *temp*).

Eficiencia en MapReduce

Para medir la eficiencia del algoritmo MapReduce sobre el transcurso de su ejecución debemos obtener:

- R: número de rounds que el algoritmo usará
- Cr: la complejidad de comunicación del round r.
- t_r: tiempo de ejecución interna

Complejidad de comunicación Cr del round r

Denotamos n_r, i como el tamaño de las I/O de las tareas de mapeo y reducción i en el round r.

$$C_r = \sum_i n_{r,i}$$ complejidad de comunicación del round r

$$C = \sum_{r=0}^{R-1} C_r$$ complejidad de comunicación para todo el algoritmo

Cálculo del tiempo de ejecución interno t_r para cada round r

Denotamos como t_r como el tiempo máximo de ejecución interno para la tarea de mapeo o reducción en el round r.

$$t = \sum_{r=0}^{R-1} t_r \text{ tiempo total de ejecución interno}$$

Parámetros adicionales

Para calcular la eficiencia total de MapReduce debemos considerar:

- **Latencia de la red (L):** se refiere al número de pasos que la tarea de mapeo o reducción tienen que esperar para recibir su 1era entrada en una ronda.
- **Ancho de banda (B):** es el número de elementos que puede ser entregado por la red en una unidad de tiempo.

Tiempo total de ejecución

El tiempo total de ejecución para la implementación de un algoritmo se define como:

$$T = \Omega(\sum_{r=0}^{R-1}(t_r + L + C_r/B)) = \Omega(t + RL + C/B)$$

Ω representa el conjunto de todas las funciones de complejidad para un algoritmo (cota inferior).

El número de rounds es clave

El número de *rounds* está proporcionalmente relacionado con el tiempo total de comunicación y la complejidad del algoritmo.

Retos en MapReduce

- Si nos enfocamos a tener un solo round, tendríamos un algoritmo ineficiente.
- Comunicación limitada: existe una estricta modularidad que prohíbe que los nodos de reducción se comuniquen entre sí.
- Porción limitada de datos: para cada nodo de mapeo y de reducción.
- Dado lo anterior, frecuentemente hay más de round.

Paralelimos en MapReduce

- Limitar la tasa de replicación
- Cálculo de la complejidad previo al algoritmo
- Uso de grafos