**Taller 11**

Solución aproximada

La programación aproximada es un enfoque donde se acepta una solución menos precisa con el objetivo de reducir el tiempo de ejecución o los recursos utilizados. A continuación, se muestra un ejemplo de un algoritmo de conteo de palabras aproximado utilizando un enfoque similar a MapReduce en C++ y CentOS 7.

|  |
| --- |
| #include <iostream>  #include <vector>  #include <map>  #include <algorithm>  #include <string>  #include <sstream>  #include <random>  // Función map aproximada: convierte un texto en pares de palabras y sus frecuencias aproximadas  void map\_aproximado(const std::string &texto, std::map<std::string, int> &frecuencias, double probabilidad) {  std::istringstream iss(texto);  std::string palabra;  std::default\_random\_engine generador;  std::uniform\_real\_distribution<double> distribucion(0.0, 1.0);  while (iss >> palabra) {  if (distribucion(generador) < probabilidad) {  ++frecuencias[palabra];  }  }  }  // Función reduce: combina las frecuencias de palabras de varios mapas  void reduce(const std::vector<std::map<std::string, int>> &mapas, std::map<std::string, int> &resultado) {  for (const auto &mapa : mapas) {  for (const auto &par : mapa) {  resultado[par.first] += par.second;  }  }  }  int main() {  // Datos de entrada: un vector de textos  std::vector<std::string> entradas = {  "MapReduce es un modelo de programación",  "MapReduce permite procesar grandes volúmenes de datos",  "MapReduce se popularizó con Hadoop",  };  // Probabilidad de muestreo (0.0 a 1.0, donde 1.0 es 100% preciso)  double probabilidad = 0.5;  // Aplica la función map aproximada a cada entrada  std::vector<std::map<std::string, int>> mapas(entradas.size());  for (size\_t i = 0; i < entradas.size(); ++i) {  map\_aproximado(entradas[i], mapas[i], probabilidad);  }  // Aplica la función reduce a los mapas para obtener el resultado final  std::map<std::string, int> resultado;  reduce(mapas, resultado);  // Escala el resultado por la probabilidad de muestreo  for (auto &par : resultado) {  par.second = static\_cast<int>(par.second / probabilidad);  }  // Imprime el resultado  for (const auto &par : resultado) {  std::cout << par.first << ": " << par.second << std::endl;  }  return 0;  } |

Compila el programa utilizando g++:

|  |
| --- |
| g++ -o map\_reduce\_aproximado map\_reduce\_aproximado.cpp |

Ejecuta el programa:

|  |
| --- |
| ./map\_reduce\_aproximado |

Este programa de ejemplo implementa un algoritmo de conteo de palabras aproximado utilizando un enfoque similar a MapReduce. La función map\_aproximado muestrea las palabras con una probabilidad dada, lo que significa que no todas las palabras se contarán, lo que resulta en una solución aproximada. La función reduce sigue siendo la misma que en el ejemplo anterior.

La precisión de la solución aproximada depende del valor de probabilidad. Cuanto más cerca esté el valor de 1.0, más preciso será el resultado, pero también requerirá más recursos. Cuanto más cerca esté el valor de 0.0, menos preciso será el resultado, pero se ahorrarán más recursos.

Este enfoque de programación aproximada puede ser útil en situaciones donde los recursos son limitados o cuando el tiempo de ejecución es crítico, y se acepta una solución menos precisa. Puedes modificar las funciones map\_aproximado y reduce de acuerdo con tus necesidades y aplicar la programación aproximada a otros problemas de procesamiento de datos.

Solicitud

Demostrar la mejora en rendimiento de utilizar mapReduce para contar la frecuencia de palabras en un conjunto de textos, utilizando un enfoque de programación aproximada de acuerdo a una fiabilidad del 70% de los datos de la muestra, respecto un conteo sin utilizar mapReduce, argumentando como esto podría servir para grandes volúmenes de datos, realizando un muestreo con diferentes escalas de volúmenes procesados, estimando como podría ser la mejora futura con X volumen de crecimiento.