MODUL 9
Komputasi Paralel dalam Machine Learning

A. Tujuan Praktikum

1. Mengukur perbedaan eksekusi serial dan parallel dalam proses Machine Learning.
2. Mengimplementasikan pelatihan model ML secara serial.
3. Mengimplementasikan pelatihan model ML secara parallel.

B. Materi

1. Machine Learning

   Machine Learning adalah cabang AI yang memungkinkan komputer belajar dari data tanpa aturan eksplisit. Digunakan untuk klasifikasi, prediksi dan clustering. Tantangan yang muncul dalam menggunakan model Machine Learning adalah dataset yang besar sehingga proses training lama, model kompleks dimana terlalu banyak operasi matematis jika proses tersebut dijalankan secara serial, CPU hanya menggunakan 1 core, sehingga proses berjalan lambat, apalagi untuk dataset besar. Diperlukan model Komputasi paralel untuk mengatasi tantangan tersebut dengan membagi pekerjaan ke banyak core CPU atau GPU.

2. Paralel dalam Machine Learning
   a. Random Forest

      Random Forest membuat banyak pohon keputusan (decision trees) secara independen. Konsep paralel bisa diterapkan dengan membangun setiap pohon di CPU berbeda secara bersamaan. Proses prediksi bisa juga dilakukan secara paralel dengan menghitung output setiap pohon sekaligus. Penerapan Random Forest menggunakan n_jobs= -1

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, n_jobs=-1)
model.fit(X, y)
```

   b. K-Means

      KMeans menghitung jarak setiap titik data ke centroid. Konsep K-Means dapat diterapkan model paralel dengan cara menghitung data ke centroid yang dilakukan bersamaan untuk banyak titik. Melakukan update centroid juga bisa dihitung secara paralel.

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, n_init=10, n_jobs=-1)
kmeans.fit(X)
```

   c. Neural Network

      Pada algoritma Neural Network proses Forward propagation, semua neuron layer bisa menghitung output bersamaan. Proses Backward propagation, gradient tiap parameter bisa dihitung secara paralel. Saat proses batch training, tiap batch dapat diproses secara paralel di GPU atau multi-core CPU.

3. Konsep n_jobs
   Di banyak library seperti scikit-learn, pytorch, tensor sudah memanfaatkan konsep komputasi paralel.
   a. scikit-learn, cara memanfaatkan paralel dengan menggunakan parameter n_jobs. Jika n_jobs = 1 maka serial (1 core), jikan n_jobs = -1 maka semua core CPU digunakan, biasanya n_jobs digunakan pada algoritma ensemble yaitu algoritma yang menggunakan teknik menggabungkan beberapa model (base learner) untuk menghasilkan prediksi yang lebih akurat dan stabil dibandingkan menggunakan satu model saja.
   b. PyTorch secara default menggunakan semua thread CPU, dengan mengatur torch.set_num_threads(k). Pada PyTorch, Forward dan Backprop dilakukan bersamaan di ribuan core GPU jika memiliki cuda.
   c. TensorFlow otomatis menjalankan operasi di semua core GPU dan dapat melakukan kontrol jumlah thread.
4. Mengimplementasikan pelatihan model ML secara parallel.

C. Langkah Praktikum

1. Buat sebuah dataset sederhana

```
X, y = make_classification(
    n_samples=50000,
    n_features=20,
    n_informative=15,
    n_classes=2,
    random_state=42
)
```

2. Buat model RandomForest dengan n_jobs=1 (melakukan training dengan serial) dan catat waktu eksekusinya.

```
model_serial= RandomForestClassifier(n_estimators=50, n_jobs=1)
```

3. Buat model RandomForest dengan n_jobs=-1 (melakukan training dengan paralel) dan catat waktu eksekusinya.

```
model_parallel = RandomForestClassifier(n_estimators=50, n_jobs=-
1)
```

4. Bandingkan hasil eksekusi antara training secara serial dan paralel.

Berikut contoh source code implementasi dari randomforest dan n_jobs = -1.

```
import time
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
```

```
from sklearn.ensemble import RandomForestClassifier

X, y = make_classification(
n_samples=50000, # dataset lebih ringan
n_features=20,
n_informative=15,
n_classes=2,
random_state=42
)
print(f"Jumlah sampel: {X.shape[0]}, Fitur: {X.shape[1]}")
print(f"Jumlah CPU Colab: {os.cpu_count()} core\n")

# Serial Training
model_serial = RandomForestClassifier(n_estimators=50, n_jobs=1) # serial

start_serial = time.time()
model_serial.fit(X, y)
end_serial = time.time()

time_serial = end_serial - start_serial
print(f"Waktu eksekusi SERIAL: {time_serial:.4f} detik")

# Parallel Training
model_parallel = RandomForestClassifier(n_estimators=50, n_jobs=-1) # paralel

start_parallel = time.time()
model_parallel.fit(X, y)
end_parallel = time.time()

time_parallel = end_parallel - start_parallel
print(f"Waktu eksekusi PARALLEL: {time_parallel:.4f} detik")
```