Sistemi e Architetture per Big Data - A.A. 2024/25
Progetto 2: Analisi in tempo reale di difetti nella produzione L-PBF con Apache Flink

Docenti: Valeria Cardellini, Matteo Nardelli
Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma Tor Vergata

Background

La *Laser Powder Bed Fusion* (L-PBF) è una tecnologia di stampa 3D impiegata per realizzare parti in metallo a partire da un letto di polveri metalliche fuse per mezzo di un laser ad alta potenza e con un fascio concentrato. L-PBF costruisce i componenti strato dopo strato, fondendo selettivamente polvere metallica tramite un laser, sulla base di un modello digitale (ad esempio, fornito tramite file CAD). Questo approccio consente una maggiore flessibilità progettuale e riduce gli sprechi di materiale rispetto ai metodi sottrattivi tradizionali, che ricavano l'oggetto rimuovendo materiale da un blocco solido. Nonostante i suoi vantaggi, la L-PBF è soggetta a diversi tipi di difetti, dovuti a impurità del materiale, instabilità termiche o errori di calibrazione. Tra i difetti più critici vi è la porosità, ovvero la presenza di vuoti all'interno del materiale, che può compromettere significativamente la resistenza meccanica e la durata del prodotto finale.

Tradizionalmente, il controllo di qualità nella L-PBF viene eseguito dopo la produzione, ma rilevare i difetti solo a posteriori comporta uno spreco di tempo, energia e risorse, poiché i pezzi difettosi devono essere scartati. Per ridurre queste inefficienze, possono essere utilizzate tecniche di monitoraggio e di rilevazione dei difetti in tempo reale. La *tomografia ottica* (OT) consente di catturare immagini termiche del letto di polvere a ogni strato del processo. Queste immagini mostrano la distribuzione delle temperature e, se analizzate correttamente, possono rivelare irregolarità legate a potenziali difetti. L'analisi in tempo reale dei dati OT consente interventi immediati (ad esempio, la sospensione del processo o la modifica dei parametri) per prevenire il degrado della qualità.

Requisiti del progetto

Lo scopo di questo progetto è rispondere ad alcune query su dati di monitoraggio del processo produttivo L-PBF [2], utilizzando l'approccio di processamento a *stream* con Apache Flink[1]. In particolare, si richiede di effettuare un'analisi in tempo reale delle immagini di OT, cioè di istantanee termiche del letto di polvere durante la fabbricazione strato per strato, al fine di stimare la probabilità di difetti man mano che i dati vengono trasmessi.

[1] `https://flink.apache.org/`

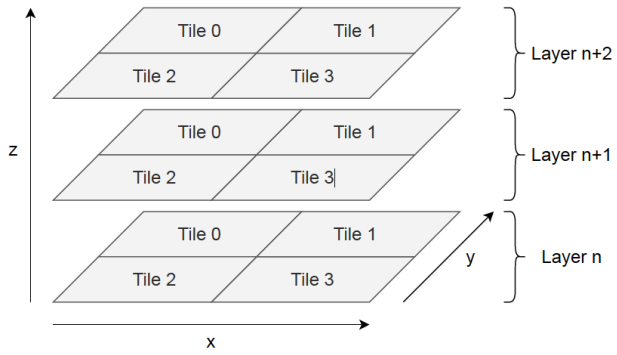

Figura 1: Layout of input data.

Dataset

Per gli scopi di questo progetto, si fa riferimento al dataset indicato nel Grand Challenge della conferenza ACM DEBS 2025 [1, 2]. Il dataset contiene immagini di OT che devono essere analizzate in tempo reale per identificare anomalie nella temperature e clusterizzare potenziali difetti. Il dataset è fornito tramite un LOCAL-CHALLENGER, ovvero un server REST con il dataset embedded. Il server implementa le specifiche OpenAPI [3]. Inoltre, viene fornita un'implementazione sequenziale di riferimento in Python, con l'obiettivo di comprendere le query da risolvere, verificare la corretta interazione con il server REST e validare la correttezza dell'output. Il LOCAL-CHALLENGER è disponibile (per i soli scopi del progetto) ai seguenti URL[2]:

- README:
  http://www.ce.uniroma2.it/courses/sabd2425/project/README.md
- Archivio con tutti i file, tranne il database:
  http://www.ce.uniroma2.it/courses/sabd2425/project/gc25-chall.tgz
- Database con le immagini di OT, ad uso del LOCAL-CHALLENGER:
  http://www.ce.uniroma2.it/courses/sabd2425/project/gc25-chall-data.tgz

Le immagini di OT rappresentano la distribuzione della temperatura sul letto di polvere per ciascuno strato dell'oggetto in fase di fabbricazione. Ogni immagine codifica i valori di temperatura T_p per ogni punto $p = (x, y)$ del letto. Il dataset è organizzato per strati (o layer, lungo la coordinata z) ed è ulteriormente suddiviso in **tile** (sotto-immagini) per facilitare l'elaborazione, come illustrato nella Figura 1. Questa strategia di suddivisione in tile migliora il parallelismo e riflette scenari pratici di produzione, nei quali i dati vengono acquisiti da più sensori o da diverse prospettive di ripresa.

I dati sono forniti come uno stream continuo. Ogni layer è ricevuto in forma di molteplici tile. Ogni elemento dello stream contiene i seguenti campi:

- `seq_id`: numero di sequenza unico per l'elemento di input;
- `print_id`: identificatore dell'oggetto in corso di stampa;

[2] sha1sum del file `gc25-chall.tgz` (formato TAR GZ): 3131c29acd3bf9a8f8e095e4a124c794c9bf0c83
sha1sum del file `gc25-chall-data.tgz`: d3ee79331da56c5ed9b66361c9a8c4e90c1a4aa3

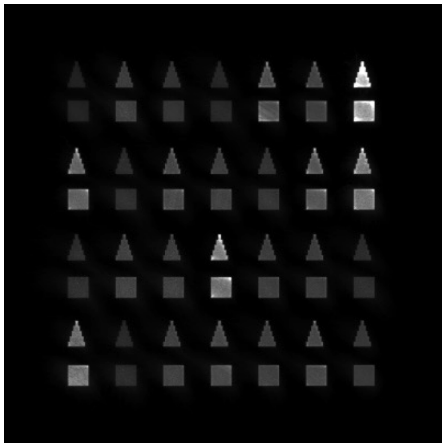

Figura 2: Esempio di immagine TIFF che rappresenta un layer. La luminosità di ogni punto rappresenta il valore di temperatura su una scala da 0 (nero) a 65 536 (bianco).

- `tile_id`: identificatore del tile all'interno del layer;
- `layer`: coordinata z del layer corrente;
- `tiff`: dati binari che rappresentano il tile in formato TIFF.

L'immagine TIFF fornisce i dati di temperatura ad alta risoluzione, codificati in 16 bit. La Figura 2 riporta un esempio di layer rappresentato in formato TIFF.

Implementazione di riferimento

L'applicazione da sviluppare in Flink dovrà interagire con il LOCAL-CHALLENGER tramite le API REST, che prevedono la creazione di una sessione, l'avvio di un benchmark, una sequenza di richieste per ottenere l'immagine OT da processare ed inviare i risultati del processamento e la terminazione del benchmark. Secondo gli script forniti, il LOCAL-CHALLENGER viene avviato in un container ed è in ascolto sulla porta 8866. Inoltre, il LOCAL-CHALLENGER mette a disposizione una dashboard[3] che riporta le prestazioni dell'applicazione che interagisce con il server, in termini di latenza e throughput.

Con il materiale scaricato dagli URL sopra indicati, vi è un'implementazione di riferimento, scritta in Python, che ha lo scopo di aiutare la comprensione del problema da risolvere ed esemplificare l'interazione con il LOCAL-CHALLENGER.

Query

Le immagini vengono fornite un layer per volta, con ogni layer diviso in molteplici tile. Ogni elemento dello stream corrisponde ad una singola tile. Si richiede di sviluppare un'applicazione di processamento in 4-stadi in pipeline: (1) analisi basata su soglia; (2) windowing (3) analisi degli outlier e (4) clustering degli outlier. Le query a cui rispondere coprono gli stati di questa pipeline di processamento.

[3] Disponibile all'indirizzo: `http://localhost:8866/dash`

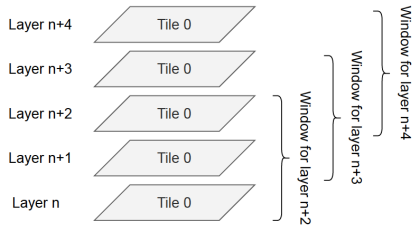

Figura 3: Esempio di finestra scorrevole sui layer, per un singolo tile.

Q1 La query Q1 si concentra sullo stadio (1) della precedente pipeline. Per ogni tile, identificare i punti dove la temperatura è sotto la soglia di $5\,000$ oppure sopra la soglia di $65\,000$. I punti sotto la soglia inferiore rappresentano aree di vuoto che devono essere esclusi dalle query successive. I punti sopra la soglia superiore sono considerati saturati e possono indicare dei difetti. Il conteggio di questi ultimi deve essere riportato; tuttavia, i punti sopra la soglia non devono essere inclusi nel rispondere alle query successive.

Per ogni tile nello stream in ingresso, la query deve produrre un output con il seguente schema:

```
seq_id, print_id, tile_id, saturated
```

dove:

- `seq_id`: numero di sequenza dell'elemento processato;
- `print_id`: identificativo dell'oggetto in stampa;
- `tile_id`: identificativo del tile;
- `saturated`: conteggio del numero di punti saturati.

Nota: l'output della Q1 non deve essere fornito al LOCAL-CHALLENGER.

Q2 La query Q2 si concentra sugli stadi (2) e (3) della pipeline. Per ogni tile, mantenere una finestra scorrevole degli ultimi 3 layer (un esempio è mostrato in Figura 3). Questa finestra temporale consente l'analisi dell'evoluzione della temperatura tra layer consecutivi. Per ogni punto p nel layer più recente di una finestra sulla tile, calcolare la *deviazione di temperatura locale*, che viene definita come la *differenza assoluta* tra:

- La temperatura media dei *vicini prossimi* di un punto, cioè di tutti i punti con *distanza di Manhattan* $0 \leq d \leq 2$ da p, considerando i tre layer.
- La temperatura media dei *vicini esterni*, cioè di tutti i punti con distanza di Manhattan $2 < d \leq 4$ da p, considerando i tre layer.

Classificare il punto come *outlier* se la sua deviazione di temperatura locale supera la soglia di $6\,000$.

L'output della query deve riportare, per ogni tile e finestra, la classifica dei 5 punti a deviazione di temperatura locale maggiore. L'output della query ha il seguente schema:

`seq_id, print_id, tile_id, p_1,` d_{p_1}`, p_2,` d_{p_2}`, p_3,` d_{p_3}`, p_4,` d_{p_4}`, p_5,` d_{p_5}

dove:

- `p_x`: Punto con deviazione di temperatura locale in posizione x nella top-5;
- d_{p_x}: Valore della deviazione di temperatura locale del punto `p_x`.

Nota: L'output della Q2 non deve essere fornito al LOCAL-CHALLENGER.

Q3 La query Q3 si concentra sullo stadio (4) della precedente pipeline. I punti outlier identificati nella Q2 devono essere clusterizzati usando l'algoritmo DBSCAN[4], usando la *distanza euclidea* come metrica.

L'output della query ha il seguente schema:

```
seq_id, print_id, tile_id, saturated, centroids
```

dove:

- `saturated`: Conteggio del numero di punti saturati;
- `centroids`: Lista dei centroidi dei cluster, includendo per ognuno le coordinate x, y e la dimensione del cluster.

Nota: L'output della Q3 deve essere fornito al LOCAL-CHALLENGER, per ottenere - a termine del benchmark - le metriche di prestazioni direttamente dal LOCAL-CHALLENGER.

Il risultato di ciascuna query deve essere consegnato in formato CSV (Flink supporta la scrittura di file in questo formato). Si chiede inoltre di valutare sperimentalmente i tempi di latenza ed il throughput delle query durante il processamento sulla piattaforma usata per la realizzazione del progetto. Riportare l'analisi del confronto nella relazione e nella presentazione del progetto.

Composizione dei gruppi

Il progetto è dimensionato per un gruppo composto da **2 studenti**.

Per gruppi composti da 3 studenti: in aggiunta ai requisiti sopra elencati, si richiede di ottimizzare il processamento. In particolare, l'applicazione di analisi delle immagini di OT, si presta a diverse forme di ottimizzazione [1]. Una è relativa al processamento dei dati in pipeline, attraverso i 4 stadi (coperti dalle query Q1–3). Un'altra è legata alla possibilità di processare, per gli oggetti in stampa (`print_id`), le tile (`tile_id`) in parallelo, per analizzare più efficientemente un layer. Una terza forma, invece, è relativa alla logica applicativa: il calcolo della deviazione di temperatura locale di un punto mostra, nel calcolo, simmetria spaziale e si basa su sole informazioni locali ad ogni punto. La deviazione di un punto $p = (i, j, 0)$ nel layer più recente all'interno della finestra W (che contiene 3 layer), può essere espresso come:

$$
\begin{aligned}
C_p &= \{q \mid \delta(p, q) \leq 2\} \\
O_p &= \{q \mid \delta(p, q) \leq 4\} \\
d_p &= \left| \frac{1}{||C_p||} \sum_{q \in C_p} T_q - \frac{1}{||O_p||} \sum_{q \in O_p} T_q \right|
\end{aligned}
$$

[4] `https://it.wikipedia.org/wiki/DBSCAN`

dove $\delta(p, q)$ indica la distanza tra i punti p e q; d_p segue direttamente dalla definizione data in Q2. Questa struttura si presta ad un'ottimizzazione tramite *convoluzione* (discreta)[5], usando un kernel (o filtro) apposito che calcola l'intera mappa delle deviazioni in un'unica operazione, come $|W * K|$, dove W è un tensore a 3 dimensioni rappresentante la finestra (x, y, e profondità 3) e K è il kernel convoluzionale che calcola la differenza come:

$$K(i,j,k) = \begin{cases} \frac{1}{||C_p||}, & \text{if } \delta((0,0,0),(i,j,k)) \leq 2 \\ \frac{-1}{||O_p||}, & \text{if } 2 < \delta((0,0,0),(i,j,k)) \leq 4 \\ 0, & \text{altrimenti} \end{cases}$$

Opzionale: in aggiunta ai requisiti sopra elencati, si richiede di utilizzare *Kafka Streams* oppure *Spark Streaming* per rispondere (almeno) alle query 1 e 2 e di confrontare, sulla stessa piattaforma di riferimento, le prestazioni in termini di tempo di latenza e throughput con quelle ottenute usando Apache Flink. Riportare l'analisi del confronto nella relazione e nella presentazione del progetto.

Svolgimento e consegna del progetto

Comunicare la composizione del gruppo ai docenti entro **venerdì 27 giugno 2025** (sole se diversa rispetto al progetto 1).

Per ogni comunicazione via email è necessario specificare *[SABD]* nell'oggetto dell'email. Il progetto è valido **solo** per l'A.A. 2024/25 ed il codice deve essere consegnato **entro martedì 8 luglio 2025**.

La consegna del progetto consiste in:

1. link a spazio di Cloud storage o repository contenente il codice del progetto da comunicare via email ai docenti **entro martedì 8 luglio 2025**; inserire i risultati delle query in formato CSV in una cartella denominata `Results`.
2. relazione di lunghezza compresa tra le 4 e le 8 pagine, da inserire all'interno della cartella denominata `Report`; per la relazione si consiglia di usare il formato ACM proceedings (https://www.acm.org) oppure il formato IEEE proceedings (https://www.ieee.org);
3. slide della presentazione orale, da inviare via email ai docenti o caricare nel repository **dopo** lo svolgimento della presentazione.

Le presentazioni si terranno **giovedì 10 luglio e venerdì 11 luglio 2025**; ciascun gruppo avrà a disposizione **massimo 15 minuti** per presentare la propria soluzione.

Valutazione del progetto

I principali criteri di valutazione del progetto saranno:

1. rispondenza ai requisiti;
2. originalità;

[5] `https://en.wikipedia.org/wiki/Convolution#Discrete_convolution`

3. architettura del sistema e deployment;
4. organizzazione del codice;
5. efficienza;
6. organizzazione, chiarezza e rispetto dei tempi della presentazione orale.

Riferimenti bibliografici

[1] L. De Martini, J. Tahir, A. Margara, C. Doblander, S. Frischbier, R. Mayer, and H.-A. Jacobsen. The debs 2025 grand challenge: Real-time monitoring of defects in laser powder bed fusion (l-pbf) manufacturing. In *Proc. of 19th ACM Int'l Conf. on Distributed and Event-Based Systems*, DEBS '25, page 223–228, 2025.

[2] DEBS 2025. Call for Grand Challenge Solutions. `https://2025.debs.org/call-for-grand-challenge-solutions/`, 2025.

[3] openAPI. The OpenAPI Initiative. `https://www.openapis.org/`, 2025.