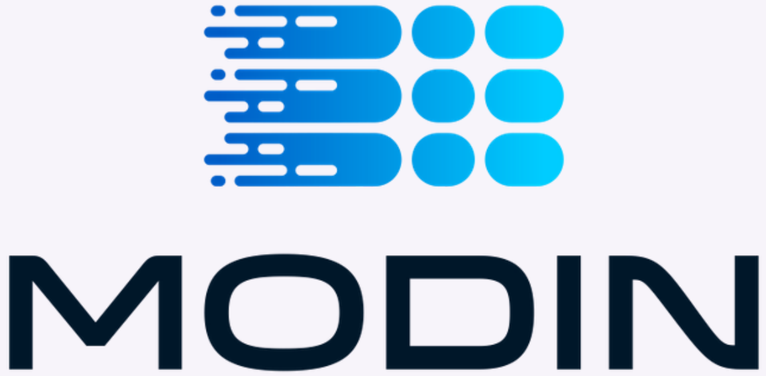
Scalabilità di Tecniche di Analisi di Dati con la libreria Modin

Francesca Prontera

Prof.ssa Sonia Bergamaschi
PhD. Luca Gagliardelli

Obiettivo della ricerca:

Effettuare un confronto tra le librerie *Pandas* e *Modin*, che si occupano della gestione di *DataFrame* ed evidenziare quale fra le due è la vincente per l'analisi di grandi quantità di dati.



Pandas

E' la libreria che rappresenta lo standard *de facto* in ambito di gestione dei *DataFrame*, scritta per il linguaggio di programmazione *Python*, utilizzata per la manipolazione e l’analisi dei dati.
In particolare, offre strutture dati quali i *DataFrame* e fornisce delle operazioni per una loro gestione.

Elevata flessibilità nella gestione della struttura del *DataFrame*

Sistema di gestione dei *DataFrame* intuitivo, che permette all'utente di non specificare i tipi di dato al suo interno

Maggiore accessibilità grazie al linguaggio ospitante *Python*

Elaborazione query su un singolo core

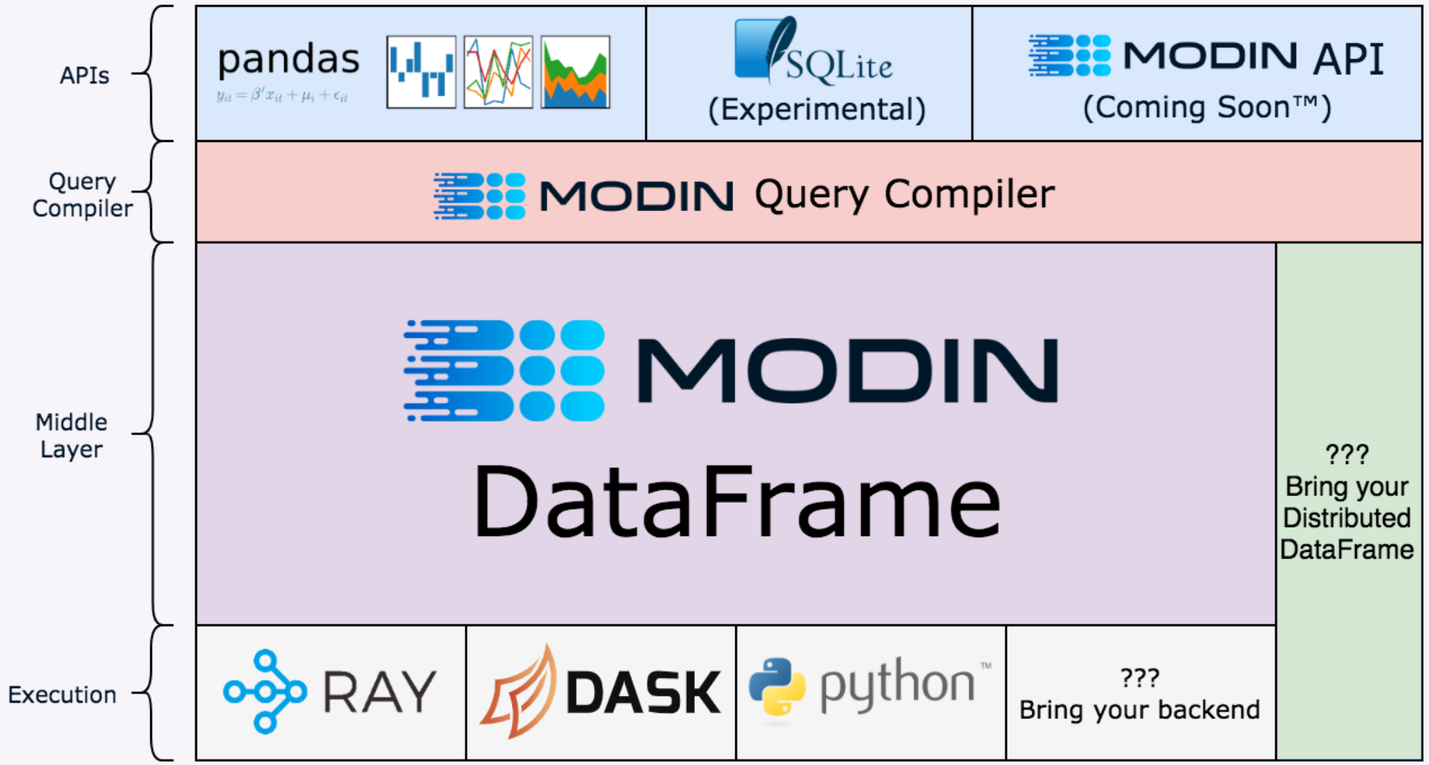
Modin

La nuova libreria ha come obiettivo quello di sostituirsi in modo trasparente a *Pandas*, per sopperire al problema della limitata scalabilità su grandi quantità di dati a cui è soggetta l'*API* precedente.

- Sintassi invariata rispetto a *Pandas*
- Sfrutta il parallelismo grazie alla gestione su più core delle query
- Partizionamento *DataFrame* sia per colonne che per righe, offrendo così flessibilità e scalabilità per entrambe le componenti.
- Progettato per funzionare su una varietà di sistemi

Architettura Modin

E' modulare, per cui è possibile scegliere tra i diversi motori di esecuzione *Ray* e *Dask*.
Entrambi i motori di calcolo sfruttano le risorse hardware in modo differente l'uno dall'altro, ma con l'unico obiettivo di parallelizzare il più possibile l'*API*.



Dask

Punti di forza:

Viene utilizzato quando si vuole manipolare un set di dati di grandi dimensioni in modo distribuito.

Sfruttando il multi-core necessario per l'elaborazione di query *Modin*, lo *scheduler* proprio di *Dask*, distribuisce il carico computazionale su più processi. Se si lavora in ambito distribuito, *Dask* risulta essere così la scelta migliore.

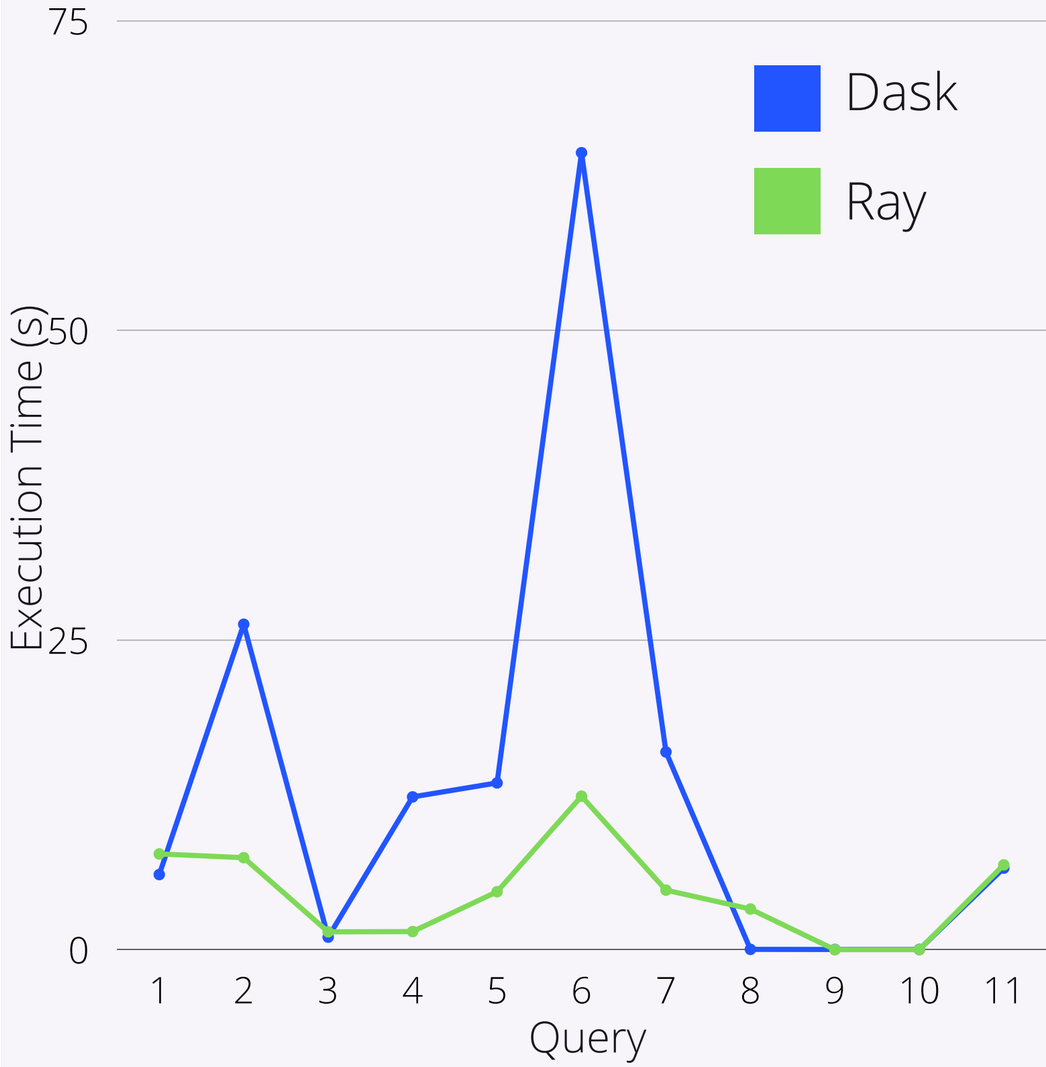
Ray

Punti di Forza:

Per avere un elevato throughput delle attività, *Ray* utilizza una pianificazione di tipo bottom-up distribuita.

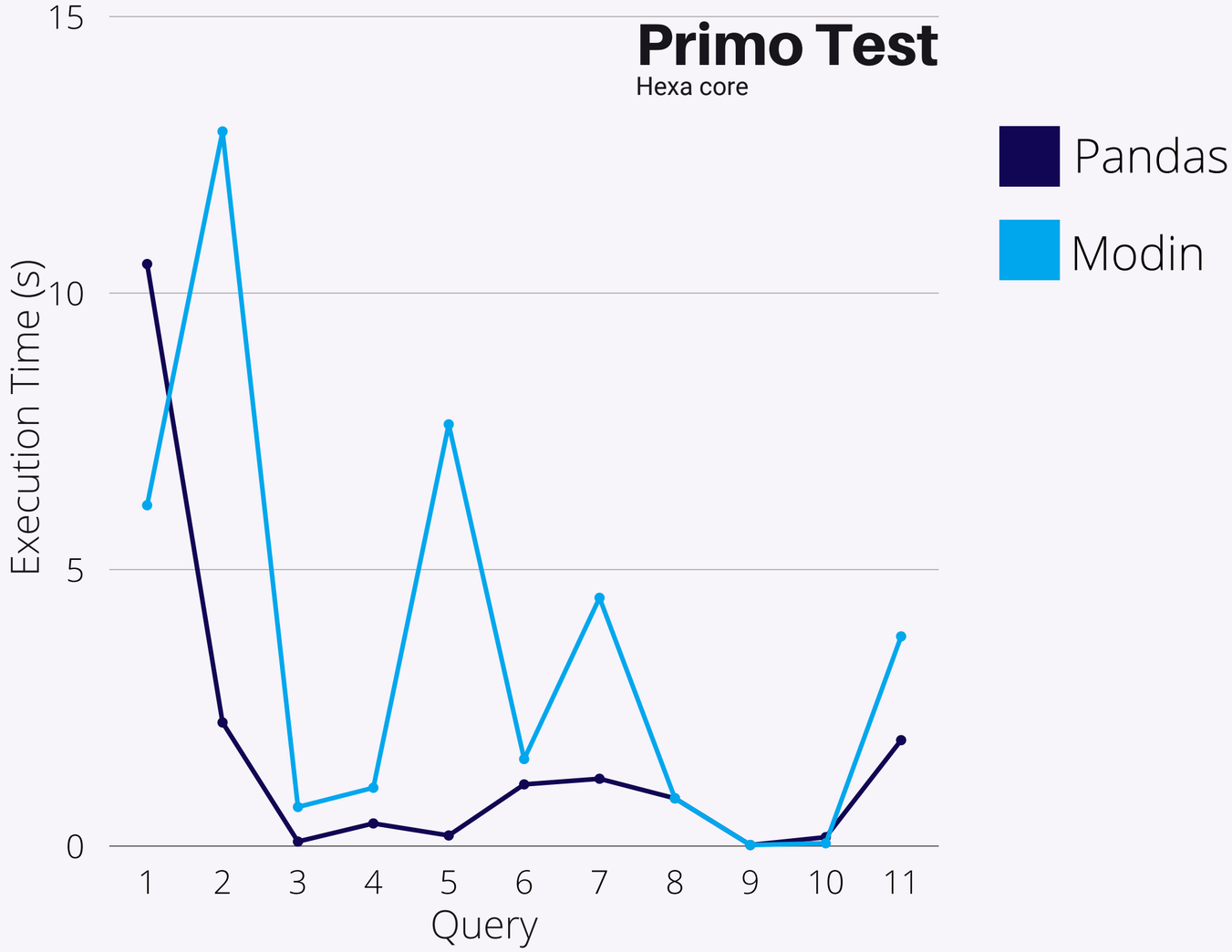
Si concentra maggiormente sulla latenza, ottenendone così una di circa 30 volte inferiore rispetto a *Dask*, punto di forza che lo rende la scelta migliore in ambito centralizzato.

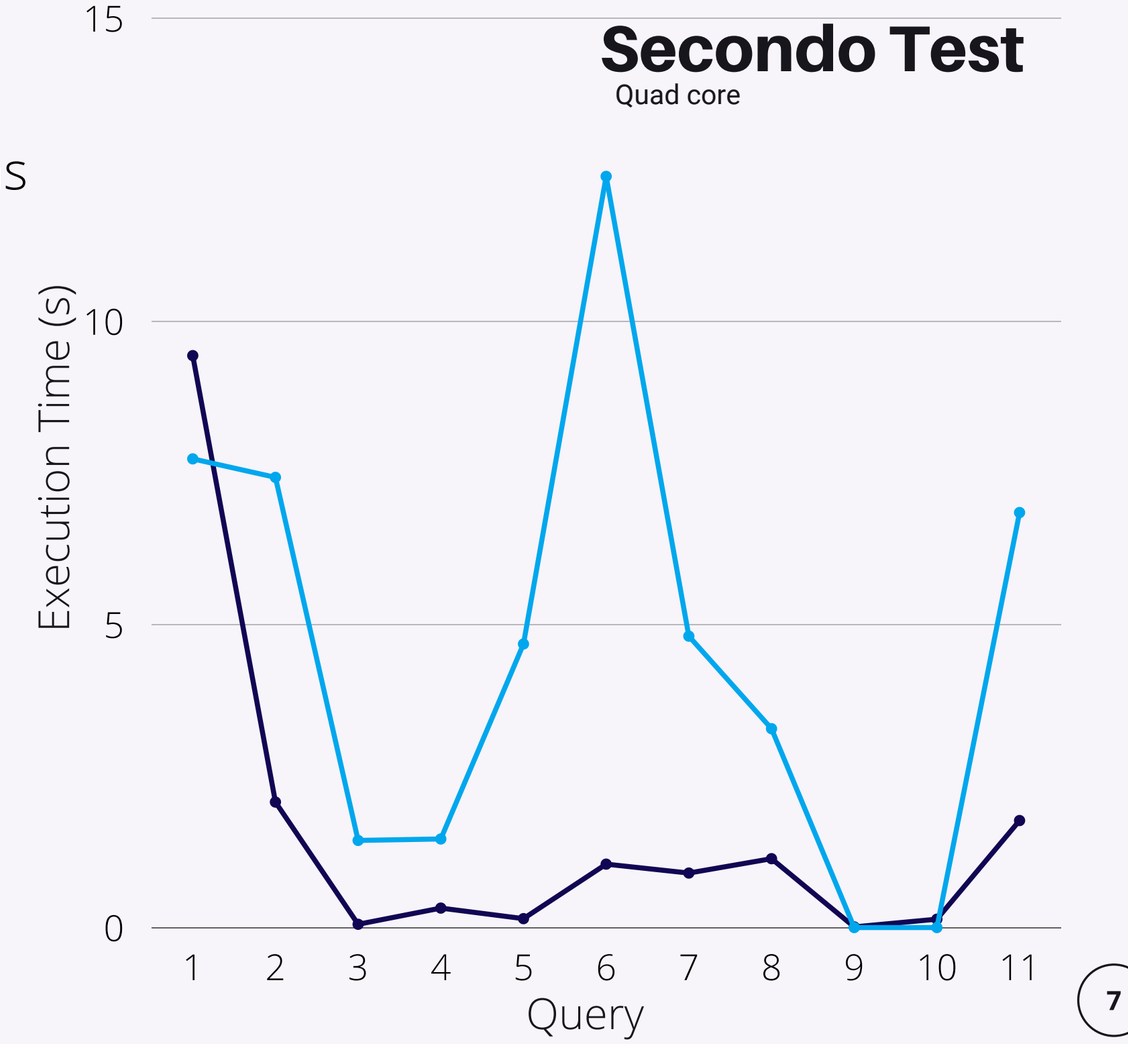
Dati raccolti dai Banchmark effettuati:



Pandas vs Modin in ambiente centralizzato

Sono state svolte su più computer, con capacità computazionali differenti, delle operazioni di analisi e filtraggio dei dati, all'interno di un dataset contenente più di 6 milioni di records.





Analisi del funzionamento di Modin in uno scenario centralizzato:

Con *Modin*, l'esecuzione di una query in modalità *multicore*, prevede la creazione di un processo per ogni core logico presente sul computer.
Nella gestione di più processi che devono comunicare tra loro per portare a termine il task, si crea un *overhead* di controllo e un utilizzo eccessivo della CPU non proporzionale al lavoro richiesto. Di conseguenza si ha un tempo di esecuzione delle query nettamente maggiore rispetto a *Pandas*.

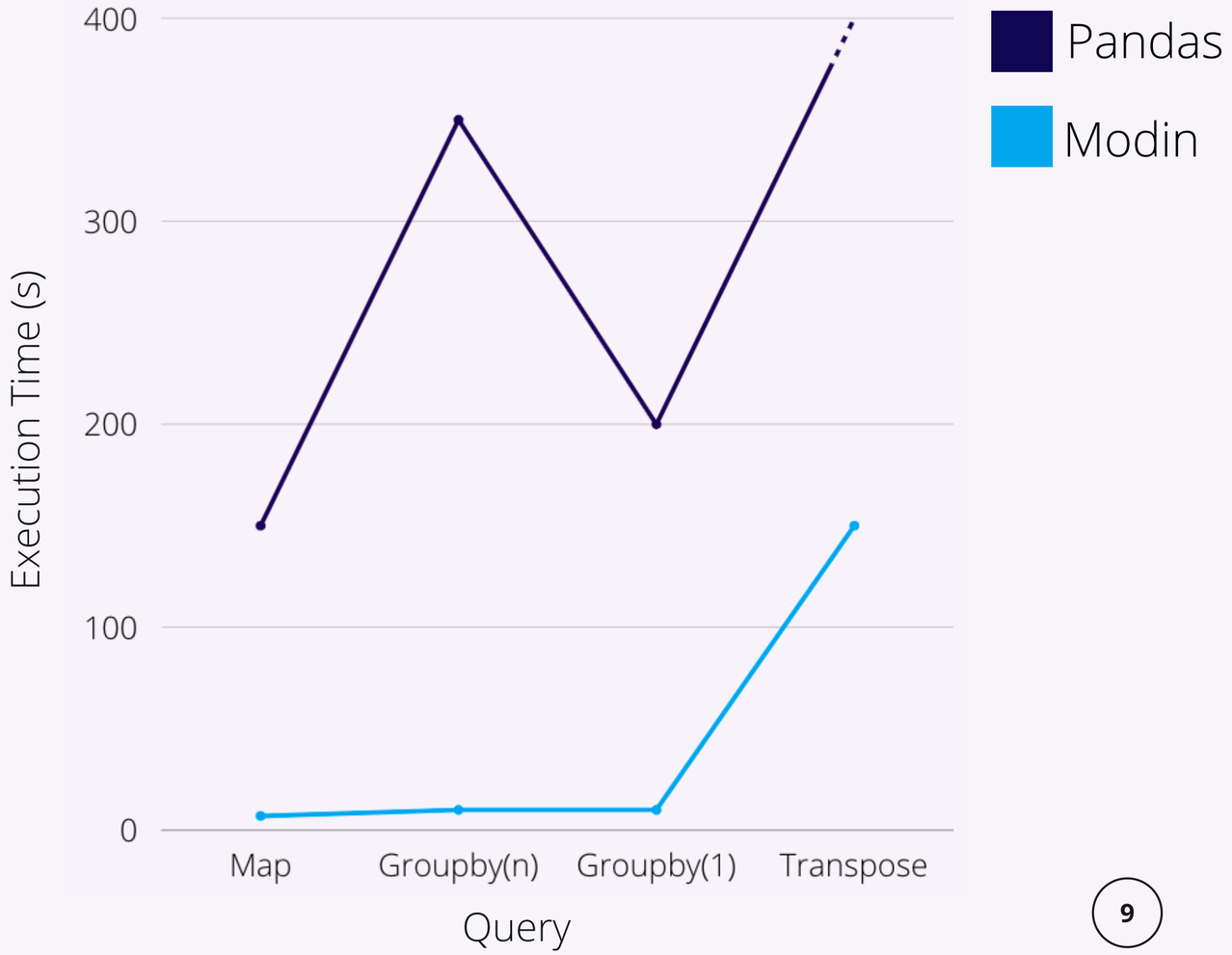
Inoltre, si è constatato da un terzo test in cui si disponeva di un computer con meno di 4 core fisici, che l'esecuzione di alcune query in *Modin* non giunge al termine o necessita di tempi eccessivi.

Modin non riesce ancora a sostituirsi a *Pandas* in un sistema centralizzato

Pandas vs Modin in ambiente distribuito

Risultati ottenuti dallo studio *Towards Scalable DataFrame Systems*, in cui si è operato su un dataset di dimensioni pari a 250 GB con più di 1,6 miliardi di righe utilizzando un cluster di 128 cores.

Durante l'operazione di transpose, Pandas non è stato in grado di trasporre nemmeno un DataFrame più piccolo di 20 GB, dopo 2 ore. È stato poi dimostrato attraverso dei test separati al caso studio, che Pandas può trasporre solo dataset fino a 6 GB, prendendo in considerazione lo stesso hardware utilizzato per i test.



Analisi del funzionamento di Modin in uno scenario distribuito:

All’interno di un sistema distribuito nel quale più macchine collaborano insieme per manipolare grandi quantità di dati, scegliere l’implementazione fornita da *Modin*, col supporto dello *scheduler* di *Dask*, produce uno speed-up significativo e in alcuni casi diviene assolutamente necessario il suo utilizzo per poter concludere l’esecuzione dei task.

In ambiente distribuito *Modin* è ancora in fase sperimentale, ma dai dati di cui si dispone, si ha ragion di credere che questa nuova *API* possa effettivamente apportare delle migliorie allo stato dell’arte, divenendo così necessaria se si hanno delle esigenze computazionali superiori a quelle supportate da *Pandas*.

Modin ottiene prestazioni migliori rispetto a *Pandas* in ambito distribuito

Grazie per l'attenzione