Struttura database htm

|  |  |  |
| --- | --- | --- |
| **Tabella hj** | | |
| **Column** | **Type** | **Description** |
| *ts* | integer | Tempo di inizio del job in epoche (secondi trascorsi dalle ore 0 del 1/1/1970) |
| *jobid + idx* | bigint + integer | Identificativo del job |
| *queue* | character varying(32) | Coda / gruppo |
| *hn (hostname)* | character varying(32) | Dove gira il job |
| *js* | integer | * 1 = pending (in coda); * 2 = running; * 3 = rimosso; * 4 = finito; * 5 = “sospeso” (qui vale come rimosso); |
| *nc* | integer | Numero di core usati dal job |
| *hsj* | double precision | Potenza in HS06 di un core nella macchina dove gira il job |
| *hsm* | double precision | Potenza totale in HS06 della macchina dove gira il job |
| *cpt (cputime)* | bigint | Tiene conto solo del tempo in cui il programma usa la CPU, in secondi   * **User time** is the amount of time the CPU was busy executing in user space. (*Aka your code*) * **System time** is the amount of time the CPU was busy in kernel space. (*Aka system calls*) |
| *rt (runtime)* | bigint | Tiene conto anche di I/O e delle attese dovute alla condivisione delle risorse con altri programmi, in secondi |
| *owner* | character varying(60) | Utente unix che esegue il job |
| *rss* | double precision | Memoria RAM usata dal job, in KB |
| *swp* | double precision | Memoria SWAP usata dal job, in KB |
| *sn (submitnode)* | character varying(40) | Da quale macchina è stato sottomesso il job |
| *disk* | double precision | Disco in uso dal job, in GB |

|  |  |  |
| --- | --- | --- |
| **Tabella htjob** | | |
| **Column** | **Type** | **Description** |
| *jobid + idx* | integer | Identificativo del job |
| *username* | character varying(80) | Username utente della coda / gruppo |
| *queue* | character varying(80) | Coda / gruppo |
| *fromhost* | character varying(80) | Da quale macchina è stato sottomesso il job |
| *jobname* | character varying(80) | Nome del job |
| *chargedasap* | character varying(180) | ? |
| *exechosts* | character varying(255) | Dove gira il job |
| *resreq* | character varying(255) | ? |
| *submittimeepoch* | integer | Tempo di sottomissione del job in epoche |
| *starttimeepoch* | integer | Tempo di inizio del job in epoche |
| *eventtimeepoch* | integer | Tempo di fine del job in epoche |
| *stime* | integer | System time, in secondi |
| *utime* | integer | User time, in secondi |
| *runtime* | integer | Wall time, in secondi |
| *maxrmem* | integer | Massima memoria RAM usata dal job, in KB |
| *maxrswap* | bigint | Massima memoria SWAP usata dal job, in KB |
| *hostfactor* | real | ? |
| *exitstatus* | integer | ? |
| *numprocessors* | integer | Numero di core usati dal job |
| *bljobid* | character varying(255) | ? |
| *clientid* | character varying(80) | ? |
| *ceid* | character varying(80) | ? |
| *userrdn* | character varying(255) | Indicates the distinguished name for the user in a registry |
| *userfqan* | text | ? |
| *gpu* | integer | * 1 = gpu utilizzata; * 0 = gpu non utilizzata; |
| *voname* | character varying(80) | ? |
| *completionepoch* | integer | Tempo di fine del job in epoche |
| *jobstatus* | integer | * 1 = pending (in coda); * 2 = running; * 3 = rimosso; * 4 = finito; * 5 = “sospeso” (qui vale come rimosso); |

|  |  |  |
| --- | --- | --- |
| **Tabella hm** | | |
| **Column** | **Type** | **Description** |
| *ts* | timestamp with time zone | Tempo di inizio del job in epoche |
| *hn* | character varying(30) | Dove girano i job |
| *cpus* | integer | Tempo di utilizzo della CPU dal job |
| *disk* | numeric | ? |
| *gpfs\_check* | text | ? |
| *cpu1\_t* | numeric | ? |
| *cpu2\_t* | numeric | ? |
| *syst\_t* | numeric | ? |
| *peripheral\_t* | numeric | ? |
| *memavail* | integer | Memoria disponibileRAM nella macchina, in KB |
| *memtot* | integer | Memoria totale RAM nella macchina, in KB |
| *swapfree* | integer | Memoria disponibileSWAP nella macchina, in KB |
| *swaptot* | integer | Memoria totale SWAP nella macchina, in KB |
| *condorload\_avg* | numeric | ? |
| *totload\_avg* | numeric | ? |
| *selfage* | integer | ? |

**Job efficiency**:

**Cpu Time, Wall Time normalizzati**:

,

**SQL e pandas**:

Si può eseguire facilmente una parte di elaborazione dei dati in SQL prima di passarli a pandas. Quali operazioni conviene eseguire in SQL e quali in pandas?

Pandas lavora in RAM, per cui conviene effettuare in SQL operazioni che riducano a monte il volume dei dati importati.

* Effettuare join tra tabelle, selezionare colonne specifiche, creare nuove feature per fare previsioni, classificazioni con attributi calcolati
* Un RDMS esegue queste operazioni con grande efficienza

Una volta importati i dati d’interesse, si usano pandas e altre librerie Python per eseguire elaborazioni più complesse

* applicare trasformazioni arbitrarie ai valori, estrarre parole dai testi, …
* Alcuni operazioni sono possibile anche in SQL, tramite estensioni specifiche dei vari DBMS

(*Pysocopg* is the PostgreSQL database adapter for the Python programming language)