Workflow

1. Python script reads the log files (including compressed files) line by line. The meaningful data in the log files are intercepted with regular expression and appended into appropriate tables in the DB (Postgress).
2. Due to the size of logged data involved (~8m records every month), sub tables containing an abstract of the larger table are created. This can reduce the querying time when we try to fetch data from the visualisation window.
3. When a visualization window is opened from a web browser, it makes a call to the php script at the server. The PHP script runs the SQL query to fetch data from the sub tables in no time.
4. Graphs are loaded through ajax scripts in order to avoid page refresh.
5. iVec website css is integrated into the visualisation pages.

Visualization Graphs

15 Tables in DB

1. Archives:
    1.1. Every record from the archive log file is imported into the table ‘archives’.
    1.2. A sub table, which groups the data based on day, project and user, is created from the ‘archives’ table. So, this sub table will contain substantially less number of records which are sufficient enough to process the SQL queries quickly by a standard server.
    1.3. Another sub table ‘Archive_many’, will be pre populated with the list of files that are archived more than once in a week. This table is also populated by running query on the 'archives’ table.
2. Stage
    2.1. Every record from the Stage log file is imported into the table ‘stages’.
    2.2. A sub table, which groups the data based on day, project and user, is created from the ‘stages’ table. So, this sub table will contain substantially less number of records which are sufficient enough to process the SQL queries quickly by a standard serve.
    2.3. Another sub table ‘Stage_many’, will be pre populated with the list of files that are staged more than once in a week. This table is also populated by running query on the 'stages’ table.
3. Releaser
    3.1. Releaser records the beginning & ending timestamps and number of passes involved in each of its run into a table releaserhistory.
    3.2. It also records every file path and block count released corresponding to the specific releaser timestamps.
4. Recycler
    4.1. Recycler records the beginning & ending time stamp along with total capacity and available capacity into the table recyclerhistory.
    4.2. Another table (recyler_tape_list) records the tape specific information (like percentage of obsolete/free/used space) for each tape.
5. Snapshot
    5.1. It records the everyday snapshot generated by the system. Duration of the snapshot is the field of interest here.
6. File Count
    6.1. Stores data on the count of file based on various sizes, project and user group for the last date the script ran.
    6.2. Another table keeps track of the file count of various sizes at specific import dates.
7. Tape Count
    7.1. This records the number of free slots, partially full, full and blank tapes in the library for different media types.