CollabTracker

Tool for collaboratively authoring artifacts

& for tracking work progress near real-time

Introduction

- The Information System architecture is undergoing radical changes

  - thick desktop applications >> thin clients of distributed systems >> cloud

- But the farther the info layers move from end users, the more painful it becomes to perform business intense functions.

  - e.g. Aggregations from multiple artifacts. The end users have to live with the network latencies and session maintenance while working on remote contents.
  - The difficulties amplify if multiple team members have to co-author an artifact without stepping over each other's work.

Workarounds

- Of course, there had been workarounds such as AJAX and Progressive Web Apps to address latencies. But one has to still wade through myriads of unproductive formalities before doing any intelligent decision making while dealing with the data on cloud.

- An ideal IT system should extend itself to the users for receiving Information instead of forcing users to stretch up to its portals.

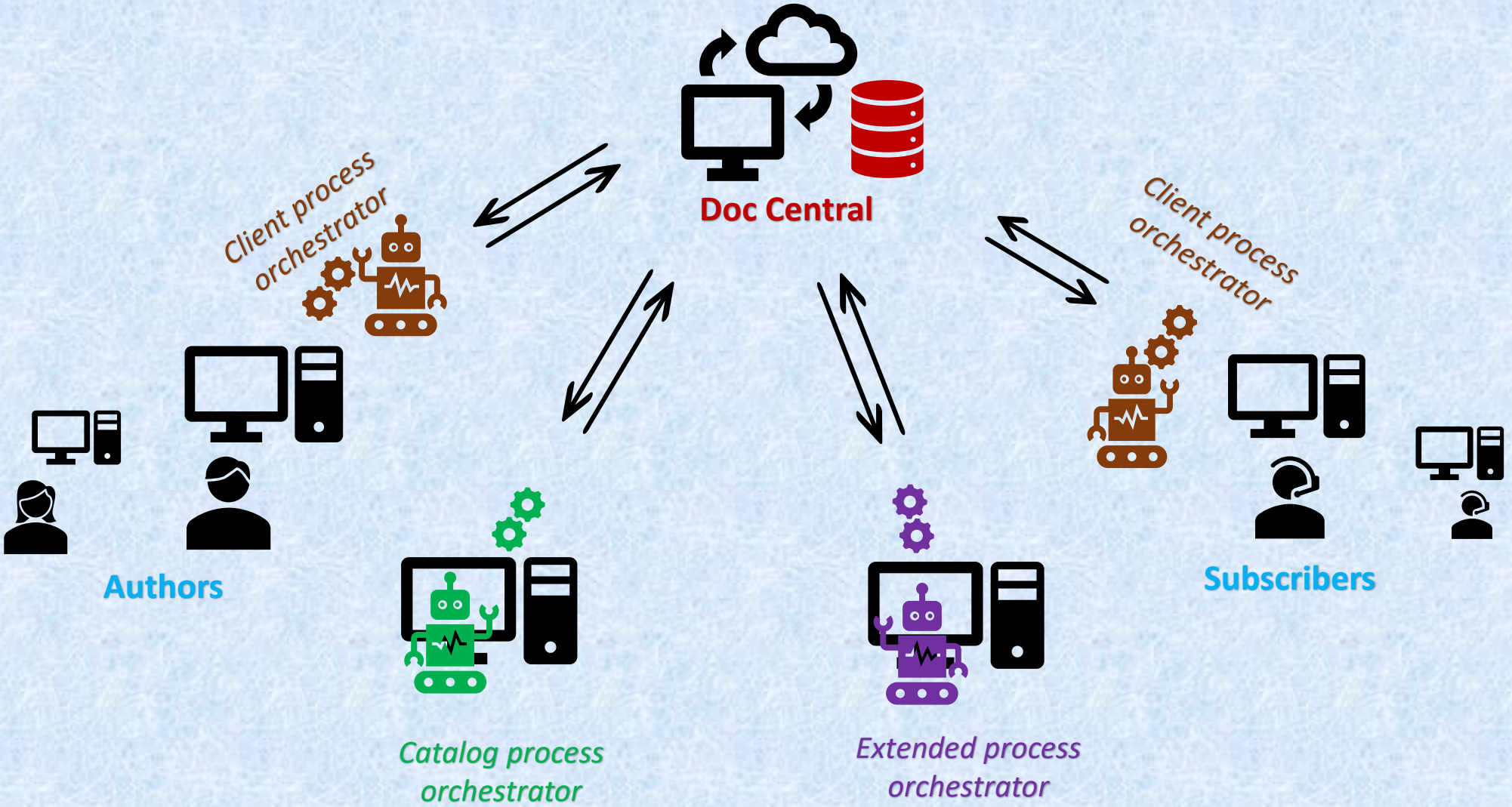
What is CollabTracker?

- This tool attempts to address two key challenges in information collaboration.
  - Assigning single ownership for data elements at enterprise level and rolling up to wider forms with least human intervention.
  - Spare the authors from unproductive information transfer and collaboration steps which machines are well apt to handle.

How CollabTracker addresses the challenges?

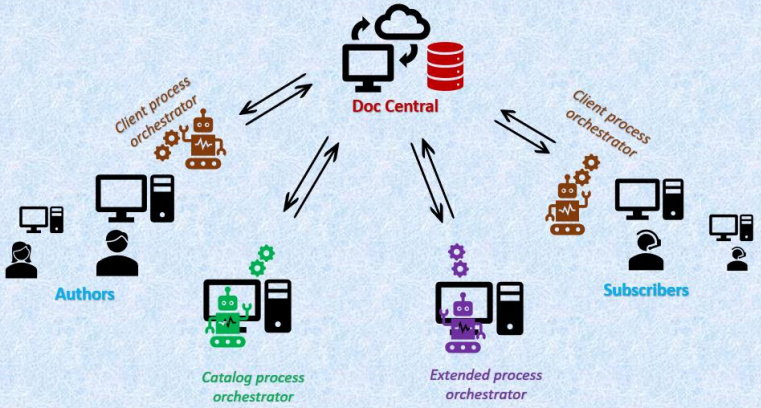
- It distributes the compute processes aptly amongst user's desktop, Doc Centrals, catalog processors and extended processing servers.
- Performs De-Clouding
  - Similar to De-Normalization which reverses an overly *normalized* database to a state where data processing is quicker, the CollabTracker brings the business data from cloud closer to users.
- Technology-agnostic foundation to integrate with any DocCentral in market
- An *open-source* framework which any software engineer can easily add newer content-types.

Compute Components



Compute Components

- Doc Central
  - Any 3rd party content-repository to host the published artifacts at Content Root and content-type specifications at Platform Root. Current version is designed to support Google Drive, Windows file system and WebDAV enabled storage services such as Sharepoint. (*Sharepoint validations pending).
- Desktop UI
  - Allows requesters to create base templates and assign ownership to specific authors to prepare content.
  - Allows authors to create new content, update and publish.
  - Displays catalog of available contents to subscribe, review and collaborate.
  - Allows authors and requestors to update status and re-assign ownership.
- Desktop Synch-up Orchestrator
  - Submits contents and requests Doc Central to process it.
  - Downloads refreshed catalogs and the subscribed content from Doc Central.
- Server orchestrator
  - Publishes *as is* or rolls up the contents received from users and extended processors.
  - Publishes catalog via SQLite file which carries the information of content in Doc Central.
- Extended orchestrators
  - Transforms contents as per special business requirements.



Three Orchestraters

- Client orchestrator runs at user desktop
  - It reads the latest catalog from the Doc Central and syncs up the local repositories at user's desktop. Also when the user flags a new draft *ready-for-upload*, it pushes the *content* into the Doc Central's *content drop box* and writes a *request file* into *request drop box* that tells server orchestrator to process the uploaded content.

- Catalog Server orchestrator runs at a defined server machine
  - It reads the request files and moves the corresponding contents into their destination folders *as is* or moves into rollup contents. After processing all such requests, it publishes the renewed catalog in the publication folder.

- Extended orchestrator runs at a defined server machine
  - It mimics a human's effort for any enrichment viz. sending the task allocation records to team members based on project trackers created by PMs, creating a *Projects* dashboard, combining multiple spreadsheets to provide a summary view and persisting timecards.

Four flat-file SQLite databases

- Sysdbfile published on platform root repository
  - Specifies which handler shall be picked to process a content, whether they are of individual type or roll-ups, any extended handling involved etc. The updates on sysdbfile such as introduction of new content types are automatically propagated.

- Catalogdbfile on content root repository
  - Holds details of the published contents at the Doc Central, their location, author etc. in the form of an Enterprise Resource Locator (ERL) master record. It is refreshed and re-published at a predefined location of the Doc Central by server orchestrator every time a content is published so the clients can pick it up during their periodic checks.

- Clientdbfile on users' desktop and extended servers
  - Holds details of local drafts, subscribed ERL contents and their local availability.

- Extendedcatalogdbfiles
  - Data related to any extended processing for special content types such as project trackers. This stays only at extended servers.

Content Handlers

- While the processors handle generic house-keeping functions, they won't have any clue on the specific needs of the contents. For that they trigger the corresponding *content handlers* at run time.

- The framework deploys a lightweight dependency injection function to load content handlers during runtime using the *dynamic class loading* feature of Java.

- A content can be either an individual type which is maintained as is, or a rollup type which will sit inside another rolled up content.

- The content handlers handle the unique requirements of contents for displaying at client side, for processing by catalog server and for any special processing by extended processor.

Pre-loaded Content Types

- **General Request Generator**
  - Acts as a general-purpose requests collector from users. Helps to categorize and take actions.
- **Idea Generator**
  - Captures ideas from team members and rolls them up at relevance (i.e. team) level. Reviewers can comment and help improvize.
- **Timesheet Capture**
  - Captures timecards in real time. Allows one to create a new time-capture function to track the time spent at specific interval and start tracking the time expended in real time. This module can be further evolved to interact via mobile devices.
- **ToDo Generator**
  - Team leaders can create a template and assign to team members to author artifacts. Todo items of a team member are rolled up into a single list for easier view.
- **Simple Tracker**
  - Enables members of a team to maintain consistent data sheets which can be merged onto a DeckerLite rollup sheet.
- **Decker Lite**
  - A simple decking process that combines the records of individual contents into a group level view.
- **Project Tracker**
  - Provides an All-in-One Dashboard to project managers that tracks effort, defects, impediments and independent task progress in near real time. (Limitation: doesn't auto-capture impacts from inter-dependency. Large projects may require an additional Project Planning tool since few aspects like interdependencies and holiday calendars are not covered by this handler.)
- **Decker Grouper**
  - With this one can setup a combiner for special contents such as project trackers into a summary view to upper management.

How to install?

- Choose the computers to execute catalog server and extended server orchestrators.
- Set the computers' (user desktop, catalog server and extended orchestrator) environmental variable PATH to include java installation location.
- Set up the servers and then the desktop users using the installation package. The installation package is downloadable from GitHub https://github.com/vibeeshK/CollabTracker file: CollabTrackerInstaller_1.x.jar. It performs the following:
  - Captures the Installation folder path, Desktop User name who will be using the application and also the proxy IP and port details.
  - Sets the property files - Common, Client, System, Server and Extendedserver with user's choices.
  - Stores the executables in the installation folder and maps the working folder to user specific folder (i.e. c:\users\Vibeesh).
- Unless you intend to implement a new custom content type for your own users, you don't have to set up a platform server. In that case you can suppress the periodic refresh by setting the flag suppressSysCompRefresh in commons.properties.
- The default installation points the demo root DemoGShContentRoot at Google drive with public read access. *Hence, with the installation steps so far, one can start training with the tool.*

Administrator initial activities:

- At the root server side replicate the model folders of DemoRoot and provide access to users as below:
  - 1_allmembersreadable: Contains artifacts, catalog publications and requests' responses folders. All users shall be provided read access to this folder.
  - 2_contributorswritable: Contains request drop box and content drop box. Provide write access on the request drop box to all contributors to place their requests. Create subfolders within content drop box with the names of the contributors and provide write access to the corresponding personnel to place their contents.
    - <INSTALL_PATH>\WindowsRoots\DemoWinContentRoot\2_contributorswritable \contentdropbox\<ApplicationUserName>
    - e.g. C:\Kannan\Java\ColbTrk\WindowsRoots\DemoWinContentRoot\2_contributorswritable\contentdropbox\DEMOUSER
  - 3_behindscene: Contains housekeeping folders.
- Configure the relevance structure appropriate to the team in the catalog master database. Add contributing users with ADMIN login using the user-maintenance content type. Few common purpose user IDs are factory set e.g. ADMIN, DEMOUSER, XTDSTDPROC, XTDTMSHPROC, XTDDECKERLITE, XTDDECKERGRPR.
- Assign a server processing machine to execute the Server Orchestrator to perform housekeeping operations on the Doc Central contents.
  - Install the CollabTracker on this machine and configure the trigger mechanism to initiate the Server Orchestrator.
  - Copy the model catalogMasterDb file from catalogMasterDbFileOf<DemoWinContentRoot> into the new root specific file.
  - Update the tables Relevance and Users in the catalogDb as per need.

Administrator initial activities (continued):

- Admin shall run the server orchestrator first so the catalog publication gets initiated and all other users can initiate sync ups before starting to contribute.
- Set up authorized contributors using User Maintenance content type.
- Assign as many extended machines to execute the Extended Server Orchestrators to handle extended functions of the doc central contents.
  - Install the CollabTracker on this machine and configure the trigger mechanism to initiate the Extended Orchestrators.
  - Copy the folder extdSrvrDeckrLite with its content i.e. its own clientdb file.
  - Ensure the placement of the extended catalog db file such as <catalogXtdTmCapture>DbFileOf<DemoGShContentRoot> in extendedcatalogdbfiles folder.
- Optionally, further customization can be done through updating *properties* files placed in the config folder viz. Common, Client, Server, ExtendedServer etc.
- User desktop side triggering:
  - Once the admin completes the server side set up, users can invoke the client UI and view published contents via catalog display.
  - The client processor needs to be triggered so the user side contents are in sync with server side.

Sourcecode Organization

- **Master Project - ColbTrkerProj:**
  - The source code packages are modularly organized within this Multi-module Maven project for easier maintenance and enhancements.
- **Mdule - ColbTrk (Base Components):**
  - Contains the base components to handle the Desktop and Doc Central processes.
  - Common routines that simplifies the work of business layer
  - Abstracts of content handlers with generic functions
  - Interface definitions to standardize Remote processing and OS handling
  - Dynamic loaders of content handlers, OS handlers and Remote processors.
  - Refreshers to synch up newer versions of content handlers without manual intervention.
  - ReviewHandlers through which the users can log remarks directly on the artifacts.
- **Module - Content Handlers:**
  - Provides the handling mechanism for each content type at client desktops and at catalog servers.
- **Module - OS Handlers:**
  - File viewing at user's desktop. Readily available for Windows and can be easily extended on other systems.
- **Module - Remote Accessors:**
  - Enables interactions with different document collaboration tools viz. Windows file system, Google Drives, WebDAV enabled portals in a consistent manner.
  - Few remote accessors viz. GoogleDrive accessor incorporate content location caching to speed up the process.
- **Module - Extended Server Components:**
  - Provides the Orchestrators for Extended processing of special contents.
- **Module – Extended Handlers:**
  - Implementations of the extended processing of special contents that require unique enrichments at server side.
- **Modules - CommonOpenCldFns and XtdCommonOpenCldFns**
  - Packages commonly used methods.
- **Project – ZColbTrkInstaller**
  - Packages installation executables.

Technology stack

- **jdk 1.8** Java Development Kit
- **SWT** Standard Widget Toolkit, a lightweight framework for desktop UI
- **SQLite** Serverless self-contained database engine for storing and exchanging data.
- **Google APIs** For accessing Google Drive based doc centrals
- **Sardine** For accessing WebDAV enabled doc centrals
- **Apache POI** For rolling up excel artifacts
- **Log4J** Error Logging
- **Maven** For organizing project modules and dependency setting with external repositories.
- **Eclipse Neon** IDE
- **Izpack** For packaging and installation

Happy Collaborating!

Thank you for innovating with us