Rapport de stage

Par : Charles Côté

Stage effectué au
Centre de Données Astronomiques de Strasbourg

Superviseur du stage en entreprise
André Schaaff
Anaïs Oberto

Professeur superviseur
Benjamin Lemelin

Programme des Techniques de l'informatique
Cégep de Sainte-Foy
16 mai 2014

Tables des matières

1 Contexte, problématique, demande

1.1 Présentation générale

Centre des Données astronomiques de Strasbourg

Établi en 1972, CDS voulait initialement dire Centre de Données Stellaires, mais le nom a changé. Le Centre des Données astronomiques de Strasbourg est un pionnier mondial dans l'accès aux données précises concernant le monde astronomique. Effectivement, ils offrent différents services sur internet, disponible pour le grand public, mais surtout utilisé par les autres laboratoires et centres de recherche tel que la NASA. En tout, le CDS compte environ 90 employés. Parmi ceux-ci 30 sont des informaticiens à temps plein ici, travaillant sur les différents services qu'offre le CDS. Les deux autres métiers que l'on retrouve ici sont des astronomes et des documentalistes. Les astronomes sont ici pour accompagner les informaticiens dans le développement des applications, mais aussi dans un rôle de recherche astronomique. Pour ce qui est des documentalistes, ils recherchent des nouveaux articles dans les revues et journaux astronomiques du monde et ajoutent à la base de données des services les nouveaux objets qu'ils trouvent afin d'agrandir constamment la base de données.

Parmi les services du CDS, on en retrouve trois majeurs qui sont plus développés et utilisés que les autres. Les voici :

Aladin : Aladin est un service écrit en Java qui est en fait un service de distribution d'images astronomiques. On peut accéder à ce service en téléchargeant l'application sur le site web du CDS. On peut voir Aladin comme si c'était une immense image du ciel que l'on peut agrandir et rechercher dans certaines parties précises de l'espace. De plus, il est possible d'afficher ce qu'on voit sous différentes longueurs d'onde. On peut rechercher par nom d'objet, mais aussi par coordonnées et par multiples critères astronomiques. Depuis quelque temps, un nouvel outil développé par les informaticiens du CDS est disponible; Aladin Lite. C'est une version miniature d'Aladin codé en JavaScript qui permet d'utiliser le service sur le site du CDS sans installer aucun programme.

Simbad : Simbad est un service disponible sur le web qui permet à l'utilisateur d'avoir accès à des données astronomiques telles que la magnitude, les parallaxes, les coordonnées, etc. en tapant le nom de l'objet en question. Simbad tire ses informations des différents catalogues astronomiques recensés (à partir du travail des documentalistes) et contient un resolveur de nom, car tous les astronomes n'utilisent pas le même nom pour tous les objets. À noter que depuis tout récemment, on intègre maintenant Aladin Lite dans les pages de Simbad, ce qui donne en plus une image de l'objet qu'on recherche.

Vizier : Vizer est, comme Simbad, un service disponible sur le web pour le grand public. C'est en fait un outil qui parcourt ce qu'on appelle des catalogues, qui sont des regroupements de données sur des objets dans le ciel. Contrairement aux pages Simbad, les catalogues concernent souvent un groupe d'objet dans une portion du ciel. Vizier contient présentement plus de 12 000 catalogues.

1.2 Problématique

D'année en année, les services du CDS deviennent de plus en plus populaires dans le monde astronomique. À chaque fois qu'un des services est utilisé, un log est créé avec les détails de l'utilisation qui en a été faite. Les logs qui proviennent des différents services sont tous de différents formats et sont éparpillés au niveau des différents services. Il n'y a pas de serveur ou d'endroit commun qui contient tous les logs.

Présentement, on peut faire des statistiques sur l'utilisation des services grâce à ces logs, mais c'est long et c'est un travail laborieux. Dans le plus concret, on parle d'environ 3 jours pour établir des statistiques sur l'utilisation d'une méthode dans un des services. C'est définitivement trop long.

C'est aussi très difficile de faire une statistique qui serait composée de différents logs de différents services, car il faut chercher ces logs qui sont un peu partout dans le Centre de Données de Strasbourg.

1.3 Demande

Plusieurs projets sont présentement en cours au Développement, mais les programmeurs aimeraient développer certains projets qui aideraient grandement aux utilisateurs ou bien doivent se consacrer à la maintenance des services du CDS.

Il fallait trouver un stagiaire qui pourrait consacrer tout son temps à un projet qui homogénéiserait les différents logs des différents services du CDS. L'accomplissement de cette tâche permettrait de faire des statistiques beaucoup plus rapidement qu'auparavant et permettrait aussi, grâce à l'adresse IP, d'établir des schémas d'utilisation pour les utilisateurs (pourquoi utilisent-ils le service, et s'ils en utilisent plusieurs, dans quel ordre sont-ils utilisés).

Pour le projet, il fallait tout d'abord créer un format de pivot qui serait utilisé par tous les services. Après avoir été traduits dans le même format, les logs seraient acheminés vers une base de données. Étant donné la grosseur que celle-ci prendrait, il fallait trouver une façon d'obtenir un temps de réponse aux requêtes plus adéquat. La solution pour ce problème est d'indexer les données à l'aide d'un outil spécialisé dans ce genre de manipulation. Finalement, il fallait aussi construire un outil qui générerait des statistiques en interrogeant la base de données.

1.4 Objectifs de l'organisation

Objectif : Réunir les logs astronomiques au même endroit et les homogénéiser
Bénéfices :
1- Permet d'établir des statistiques par la suite
2- Protège les données

Objectif : Développer un outil de statistiques plus rapide que la méthode utilisée présentement
Bénéfices :
1- Sauve beaucoup de temps
2- Permet d'avoir des statistiques très précises

2 Présentation des travaux à réaliser

2.1 Travaux à réaliser

Je devais tout d'abord commencer par écrire des programmes de traduction pour les trois services principaux à l'observatoire, en Java. Par la suite, il fallait faire de la recherche par rapport à la base de donnée que nous allions utiliser, et choisir la bonne. Après l'avoir trouvée, je devais écrire le programme d'insertion qui allait insérer les documents produits au format pivot par mes autres programmes dans la base de données.

Étant donné le temps de réponse long dû au très grand peuplement de la base de données (centaines de millions d'entrées), il fallait trouver un moyen d'interroger la base de données plus rapidement, et donc trouver un outil qui pourrait indexer les données. Finalement, s'il restait du temps, je devais commencer à réaliser le programme qui générerait des statistiques en interrogeant l'index.

2.2 Planification

La planification des mes activités comprenaient tous les travaux que j'avais à réaliser en plus du temps consacré à la recherche d'outil. Le tableau qui suit montre la répartition de ces activités prévue pour les onze semaines du stage

Activité	**Semaine**
Familiarisation et explication du projet	1
Écriture des programmes de traduction pour les trois services principaux et du programme d'insertion.	2 à 4
Recherche et choix de la base de données, application sur le programme d'insertion.	4 et 5
Peuplement de la base de données	6
Recherche et application de l'outil d'indexation pour les données	7 à 9
Tests sur l'indexation des données + Test sur les performances des outils	
Regard sur la construction d'un outil de statistiques	11

3 Description des activités réalisées

3.1 Programmes de traductions vers le format pivot

La toute première étape de mon stage a été de définir quel serait le format pivot que l'on choisirait pour l'ensemble des logs du système. Nous avons donc eu une réunion à ce sujet et avons discuté des éléments importants que l'on doit garder et surtout qui se retrouvent dans chaque log de chaque service. Finalement, nous en sommes venus avec ce format:

identificateur	adresse_ip	date	service	query-string	method
0000001	130.79.30.10	2010/09/10 23:12:03	Simbad	id=M31 coord=5.93.34 magn=45	Sim-id
0000002	10.69.39.129	2010/09/10 23:12:04	Aladin	-	Allskyimage

Le format pivot contient donc un identificateur unique, qui se crée automatiquement à l'étape d'insertion dans la base de données, l'adresse IP de l'utilisateur, la date à la seconde près de l'utilisation du service, le nom du service utilisé, la méthode du service utilisé et les query-strings. Les query-strings sont en fait des paramètres précis par rapport à la recherche que fait l'utilisateur dans le système. Ce sont des associations de terme avec leur valeur dans la recherche. À noter qu'il peut y en avoir beaucoup pour un seul log.

Après avoir défini ce format, il était temps de commencer à écrire les programmes de traduction pour chaque sorte de log. Le premier programme que j'écris est celui pour le service Simbad qui reçoit des logs Apache HTTP qui ont cette forme :

```
130.79.128.30 - - [01/Jun/2013:00:00:18 +0200] "GET /simbad/sim-normid?ident=hd25204
HTTP/1.0" 200 10 "-" "wwwget/3.09"
```

Grâce aux fonctions de manipulation de chaînes de Java, j'ai pu décortiquer les différentes parties de chaque log pour emmagasiner toute l'information dont j'ai besoin. Certaines informations sautent aux yeux comme l'adresse IP et la date, mais il faut aller plus loin pour la méthode et les query-strings. La méthode utilisée ici ce trouve à être le mot avant le point interrogation, ici sim-normid. Pour ce qui est des query-strings, ce sont toutes les associations de clé et de valeur qui se trouvent après le point d'interrogation. Ici donc, on en a seulement une qui est ident=hd25204.

Écrit complètement en java, mon programme garde en mémoire toutes les données importantes au format pivot et les place ensuite dans une instance de la classe JSON. Effectivement, le JSON a été choisi ici

devant un modèle de données relationnelles entre autres à cause des query-strings. Un document JSON permet d'avoir plusieurs entrées dans un même champ, ce qui est très utile dans le cas présent. Avec une base de données relationnelle, il aurait fallu créer un lien vers une table d'intersection et on préfère tout avoir dans la même table pour ce projet. À noter qu'en NoSQL, le terme « table » est plutôt remplacé par le terme « collection » qui sera utilisé pour le reste de ce rapport.

Le deuxième programme de traduction que j'ai écrit fut celui pour le service VizieR. Voici à quoi ressemble un log VizieR :

```
----votable/ 2014.02.23,22:58:35Z 446511515s: GET from 84.130.178.108 (84.130.178.108)
$AGENT: Aladin/v7.533 Java/1.7.0_45
-source=USNO-B1
-c=15 17 09.00 +60 16 43.0
-out.add=_RAJ,_DEJ
-oc.form=dm
-out.meta=DhuL
-out.max=999999
-c.rm=9.122
-out=_VizieR,*Mime(image/fits),*
-mime=TSV
```

Encore une fois ici, on voit bien la date et l'heure au tout départ. J'ai écrit une méthode dans chaque programme de traduction qui reformate la date complète dans un même format pour tous les logs afin que l'on puisse utiliser celle-ci pour faire des recherches par rapport au temps. On voit bien aussi l'adresse IP. Pour ce qui est de la méthode, elle est nulle pour VizieR, c'est un des seuls services où l'on ne retrouve pas de méthode utilisée. Au niveau des query-strings, elles sont assez claires ici. Après les 2 premières lignes du log, ce ne sont que des query-strings qui défilent jusqu'à la fin du log. Je devais demander aux astronomes lesquelles ils voulaient que le programme retienne, car certaines sont complètement inutiles.

Finalement, le dernier programme de traduction que j'ai écrit fut celui pour les logs du service Aladin, qui ont cette forme :

```
2012/01/01 00:04:52 (java on 201.230.199.239) Start standalone v7.073 perf=0
java=1.6.0_29/Sun Microsystems Inc. syst=Windows XP/x86/5.1 from=CDS-WebStart lang=en
```

Le log d'Aladin est assez simple à décortiquer. On voit très clairement la date ainsi que l'adresse IP un peu plus loin. La méthode ici est ce que l'on retrouve après la parenthèse, ici Start standalone. Les query-

strings sont les clés et valeurs qui se trouvent après jusqu'à la fin du log. Le volume de ces logs était définitivement moins gros que les deux autres services, car ce service n'est pas en fonction depuis très longtemps au CDS.

Voici un exemple de ce qui est généré par mon programme à la sortie. Comme on peut voir, ce ne sont que des documents JSON qui sont produits, et le format est le même, peu importe le service d'entrée. Ici on voit le contenu d'un fichier de logs du service VizieR.


```
{
	"ip_address":"84.130.178.108",
	"query_strings":{
		"source":"USNO-B1",
		"c":"15;17;09.00;+60;16;43.0",
		"return-type":"VOTable"
		},
	"service":"VizieR",
	"method":"votable",
	"date":"2014-02-23 22:58:35"
}

{
	"ip_address":"180.76.5.190",
	"query_strings":{
		"source":"I\/317\/sample",
		"return-type":"HTML"
	},
	"service":"VizieR",
	"method":"Vizier",
	"date":"2014-02-23 22:58:01"
}
```


Peu importe le log qu'on traduit, en sortie on aura un format comme celui-ci.

3.2 Programme d'insertion dans la base de données

Après avoir généré les fichiers de logs en format pivot en JSON, il fallait trouver dans quelle base de données on allait stocker toute cette information. On avait besoin d'un SGBD qui pouvait :

- Supporter les documents JSON et donc être de type No-SQL
- Permettre de faire des répliques et des sauvegardes
- Être performant avec de gros volumes ou permettre l'indexation
- Disposer d'une bonne documentation sur internet
- Être un outil multiplateforme et gratuit

Après des recherches et des comparaisons de performance avec MySQL (question de voir si la base de données relationnelle aurait pu fonctionner dans notre contexte), notre choix a été MongoDB. C'est un SGBD gratuit qui est un des plus gros joueurs dans les nouveaux SGBD du No-SQL. Il dispose de collections qui contiennent des documents JSON. Au niveau des performances, j'ai pu remarquer qu'il répond plus rapidement que MySQL pour des requêtes avec une collection contenant environ 200 000 entrées.

Je devais donc écrire un programme en Java qui sera implanté sur le serveur de log qui reçoit les différents fichiers JSON et les insère dans MongoDB. Un autre point important est que je risque des fois de recevoir un fichier JSON un matin par exemple, et recevoir un autre en après-mi du même service, qui contiendrait aussi des entrées du matin. Cette situation crée des doublons dans la base de données, ce qu'on veut éviter. J'ai donc écrit une méthode qui, après une importation réussie, enregistre dans un fichier texte la dernière ligne importée dans MongoDB. J'ai aussi une méthode qui, avant l'importation, vérifie ce fameux fichier texte qui contient tous mes deltas pour chaque service. Si on retrouve la ligne du delta dans le fichier qu'on tente d'importer, on continue l'importation à partir de cette ligne dans le fichier JSON. Sinon, c'est que c'est un nouveau fichier et donc on importe le tout.

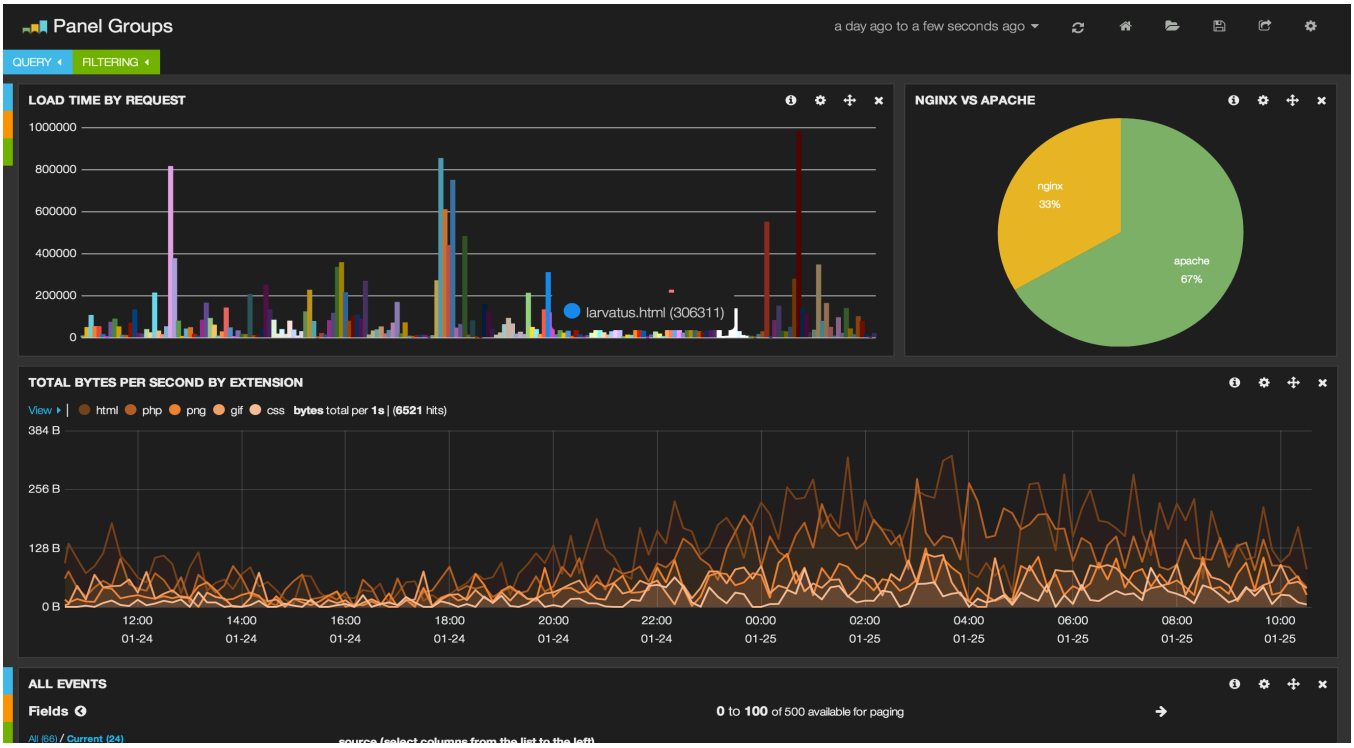
3.3 Recherche et choix d'un outil d'indexation et application de l'outil

Une bonne partie de mon stage était consacré à la recherche d'outils, que ce soit pour la base de données ou bien pour l'outil d'indexation que l'on allait utiliser. Après quelques heures de recherche et grâce aux propositions de mes superviseurs, j'ai choisi l'outil ElasticSearch. Cet outil était déjà familier pour mes superviseurs, car un ancien stagiaire l'avait déjà utilisé pour son sujet de stage.

ElasticSearch est un outil spécialisé dans l'indexation de données, particulièrement pour le Big-Data. Codé en Java et multiplateforme, cet outil est basé sur un autre outil, Lucene d'Apache et est sur les mêmes licences que celui-ci.

Ce n'était pas tout de découvrir l'outil, il fallait aussi que je voie s'il y avait possibilité de le connecter avec MongoDB. J'ai trouvé un « plug-in » d'ElasticSearch qui se connecte à MongoDB et qui indexe tout le contenu d'une collection (précisé lors de la configuration du module).

L'utilisation d'ElasticSearch allait simplifier une grande partie de la dernière étape du stage qui est la génération de statistiques. Effectivement, un autre outil appelé Kibana qui fonctionne en grande collaboration avec ElasticSearch permet d'établir de faire des graphiques simples à partir de l'information qu'on a dans un index défini.



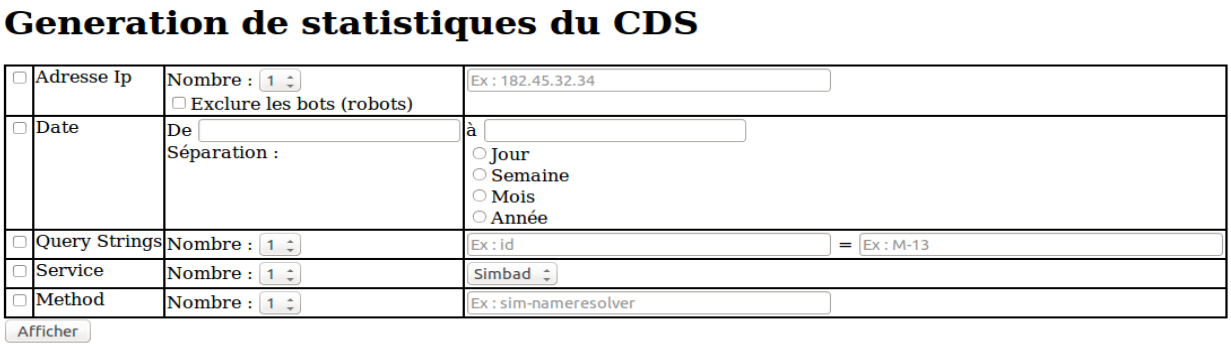
Comme on peut voir, Kibana permet vraiment de faire des graphiques intéressants et sans trop d'effort de paramétrage. Toutefois, je n'ai pas été capable d'attacher le champ de date de mes logs au champ de temps que prend Kibana pour faire ses graphiques. C'est dommage, car le but premier des statistiques est justement de faire des graphiques par rapport au temps (par exemple : combien y a-t-il eu de requêtes sur la méthode sim-nameresolver dans Simbad durant les 6 derniers mois). Ce problème, que j'ai essayé de résoudre pendant presque 2 jours, n'a jamais été réglé et il a donc fallu trouver une alternative.

Un autre problème que j'ai rencontré fut les échecs d'indexation de la collection de MongoDB. Avec l'aide de mes superviseurs, j'ai découvert qu'en faite, ElasticSearch finissait par planter, car il utilisait tout son espace mémoire alloué. En démarrant ElasticSearch avec un paramètre pour modifier son espace de mémoire maximum, on réglait partiellement le problème, car pour l'instant on a 400 millions de données dans la base MongoDB (j'ai importé des logs qu'on m'a fournis des différents services) et en montant l'espace mémoire à 8 Gigas, ElasticSearch à importé 235 millions d'entrées sur 400 millions. En faisait des recherches par la suite, j'ai bien vu qu'il fallait absolument changer ce paramètre pour des utilisations de serveur, comme dans notre cas. Par contre, on parle plus de 32 Gigas à 48 Gigas de mémoire maximum, ce que je ne disposais pas sur mon poste.

On m'a demandé, pour vérifier si c'était vraiment les bons outils à utiliser, de faire des tests de performance lors de l'importation de données vers la base MongoDB et lors de l'indexation de ceux-ci, ainsi que calculer les différents temps de réponse sur les différentes requêtes que je pouvais faire sur MongoDB. Aussi, question d'avoir une base de données « grandeur-nature », j'ai passé un temps significatif à importer des données dans MongoDB pour voir si l'outil continuait d'être efficace même avec beaucoup plus de données que seulement 1 an de log de chaque service. J'ai donc ajouté 5 ans de logs de Simbad (les plus volumineux) dans la collection, et j'ai regroupé plus de 400 millions d'entrées dans la collection, ce qui était déjà plus réaliste comme nombre.

3.4 Générateur de statistiques

Malheureusement, quand est venu le temps de commencer à regarder cette partie, nous étions déjà rendus à la dernière semaine du stage et je n'est pas pu faire beaucoup de travail pour cette partie. J'ai tout de même écrit en HTML ce qui pourrait être le formulaire de requête une fois que les données seront indexées comme il faut.



De plus, j'ai réussi dans les derniers jours à établir une connexion entre MongoDB et PHP, ce qui permettrait de générer les statistiques via cette interface (mais beaucoup plus lentement étant donné que sur MongoDB, les données ne sont pas indexées).

4 Évaluation des activités réalisées et des biens livrables produits

4.1 Programmes de traduction et choix du format pivot

Tous les programmes ont été faits, testés et complétés comme il se doit. Les trois services principaux traduisent leurs logs et créent les documents JSON appropriés. De plus, le choix du format pivot a été le bon; il permet de produire les statistiques que l'on a besoin au CDS et tous ces champs sont contenus dans presque tous les logs des différents services.

4.2 Programme d'insertion et choix de la base de données

Je suis très content du choix de la base de données, je pense vraiment que MongoDB est presque parfait pour notre situation. Je suis aussi très fier du programme d'insertion dans le SGBD. Le module qui regarde les delta fonctionne bien et l'insertion se fait à merveille, et beaucoup plus rapidement que par la commande de MongoDB prévue à cet effet.

4.3 Recherche et choix d'un outil d’indexation et application de l'outil

Je suis content d'avoir travaillé avec ElasticSearch même si j'ai rencontré beaucoup de problèmes et que cette partie n'est pas entièrement finie. Il reste à trouver une façon de tout indexer le contenu de MongoDB (probablement en ajoutant plus de mémoire au serveur de log) et surtout de trouver comment attacher les champs date de Kibana et des logs ensemble.

4.4 Générateur de statistiques

Honnêtement, j'aurais aimé avoir eu plus de temps pour développer cet outil, car ceci constituait un bon défi en programmation web pour moi, et j'aurais bien aimé toucher un peu au PHP pour construire cette application. J'ai tout de même laissé une interface ainsi que le code pour établir une connexion (et un guide) entre MongoDB et PHP.

5 Sommaire des méthodes et des normes

5.1 Programmation des applications

5.1.1 Équipement :

Pour tout mon stage, sauf pour les 4 dernières semaines :

- Mon poste a été un x86 avec 4 Go de mémoire vive et équipé d'un écran 17 pouces. Le système d'exploitation installé est Ubuntu 13.05.
- Je suis situé dans la bibliothèque de l'observatoire et j'ai mon espace bien à moi, aucun problème de ce côté.

Pour les 4 dernières semaines de mon stage

- Afin de faire les importations de logs et les indexations plus rapidement, on me donne une nouvelle machine assez similaire à l'ancienne, mais contenant maintenant un processeur i5 et 16 Go de mémoire vive.

5.1.2 Logiciels :

- Eclipse Juno
- Open Office (Apache)
- Mozilla Firefox 27.0.1
- Notepad++
- Terminal (Linux)
- MongoDb 2.6
- ElasticSearch 1.0.1

5.1.3 Méthodes :

- Tout d'abord, le fichier de log de la journée est récupéré par l'administrateur qui lui le passe dans son programme de traduction, dépendamment du service qui lui est approprié.
- Le fichier JSON produit est ensuite acheminé au serveur. Dès que le serveur reçoit un fichier JSON, il roule le programme d'insertion (et vérifie en même temps les doublons).
- Une fois insérés dans la base, les nouvelles entrées doivent maintenant être indexées, mais ce processus se fait automatiquement quand ElasticSearch et MongoDb sont ouverts simultanément.
- Les données apparaissent maintenant sur les statistiques qui sont faites à partir d'ElasticSearch.

5.1.4 Normes :

- J'ai suivi les normes du langage Java pour écrire mes applications
- J'ai fait des tests pour mes applications en Java
- Le travail que je faisais restait au CDS car je travaille avec des addresses IP confidentielles

6 Appréciation personnelle du stage

Mon intégration au CDS a été vraiment très bien. On m'a tout de suite indiqué ma place de travail dans la bibliothèque qui fut un endroit superbe et calme pour travailler. Il était très intéressant de pouvoir avoir des discutions avec les autres stagiaires qui, même si ils n'étaient pas en informatique, ont quand même pu me donner de l'information sur le domaine d'affaire (ici l'astronomie).

Le travail d'équipe se faisait bien, nous avion souvent des réunionsles trois ensemble (mes deux superviseurs et moi) où nous discutions de mon avancement et des points qu'il faudrait que je corrige sur une partie du travail effectué. J'adorais ces réunions, car elles me permettaient de donner mon point de vue sur le projet et de participer à sa construction.

En arrivant au CDS, je n'avais pas une grande connaissance des logiciels libres mais j'ai vite appris qu'en Europe, « l'open-source » est beaucoup plus répandu qu'en Amérique du Nord. Il fallait donc que je commence à utiliser Linux pour la première fois de ma vie. En plus d'être plongé dans un univers complètement différent de Windows, je travaillais avec des outils open-source dont je n'avais jamais entendu parler avant.

J'ai vraiment prit confiance en moi, j'ai été capable de m'adapter et de produire assez vite dans un environnement inconnu.

J'ai été vraiment surpris du rôle qu'on m'a donné en stage comme technicien. Je m'attendais à me faire dire quoi faire et comment le faire et de finir cette tâche avant un certain délai, mais non, pas du tout. J'étais au cœur des discutions et certaines décisions importantes m'étaient laissées (comme le choix de la base de données et du système d'indexation).

7 Conclusion

Je peux dire que mes objectifs personnels et ceux de l'entreprise ont été atteints. En effet, personnellement, j'ai acquis de l'expérience dans la manipulation de chaînes de caractères en Java, dans l'utilisation de MongoDB, dans l'utilisation d'un outil d'indexation et dans l'utilisation de Linux. De plus j'ai pu voir si l'informatique était vraiment le domaine pour moi concrètement. Je suis très content d'avoir vécu cette expérience et le fait de la vivre en France l'a rendue encore meilleure.

Au niveau de mes superviseurs, ils sont contents du travail accompli et m'ont dit qu'ils auraient aimé que je reste un peu plus longtemps en France étant donné que le développement allait si bien. Certaines parties n'ont pas pu être finies à temps, mais c'était prévisible et on a préféré mettre plus de temps sur le bon choix des outils avant de passer à la phase finale, ce que je comprends et accepte totalement.

Remerciements

J’aimerais remercier les personnes qui m’ont supporté et aidé durant mon stage. Tout d'abord, j'aimerais remercier Mr André Schaff et Mme. Anaïs Oberto qui m'ont donné la chance de faire ce magnifique stage et qui m'ont aidé tout le long de celui-ci. Ensuite, j'aimerais remercier Mme. Éliane Thal qui en plus de m'avoir contacté avec Mr Schaff, m'a évalué pour ma soutenance faite en France. Finalement, j'aimerais remercier Mr. Benjamin Lemelin pour tout les conseils qu'il me donnait dans les entretiens hebdomadaires que nous avions.