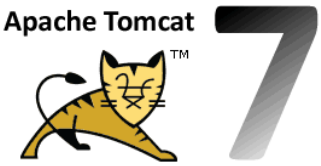
Travail d'étude et de recherche
Monitoring et Data visualisation

Hedda Hedi
Engilberge Swan
Ouleha Nadir

Encadrant : Fabien Hermenier

Juin 2013



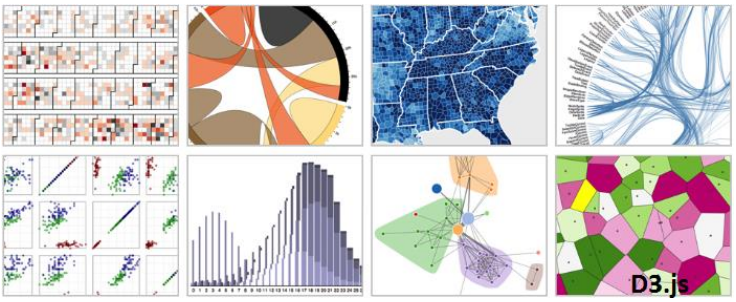


Table des matières

Chapitre 1 Introduction

Les recherche sur la data visualisation sont un domaine phare aujourd'hui. Tout le monde veut représenter ses données sous formes plus exploitable et efficace.

Prenons notre sujet d'étude et de recherche, il consiste à modéliser de manière simple la disponibilité de machines virtuelles sur les clusters de Grid5000, c'est du monitoring sur un ensemble de clusters, noter que nous pouvons nous adapter à n'importe quelle plateforme du moment qu'elle respecte le format du JSON.

Grid5000 est une plateforme distribuée dédiée à l'expérimentation des systèmes concurrents, distribués, parallèles hébergée à Sophia-Antipolis.

Le but étant de représenter les données reçue par BtrPlace un logiciel mis en place par notre encadrant lors de sa thèse, nous devons rendre une visualisation treemap implémentée via la librairie D3 (Data-driven documents) compatible avec les différentes données de BtrPlace telle que les ressources ou les contraintes d'une machine virtuelle ou d'un nœud.
C'est donc une Web application compatible avec BtrPlace.

On doit obtenir une représentation visuelle modélisé via une treemap capable d'être scalable sur le nombre de données à traiter.

C'est à dire pouvoir zoomer mais aussi faire la différence sur l'état d'un nœud ou d'une machine virtuelle si elle ne respecte pas tel type de contraintes de BtrPlace et ceux sur un grand nombre de nœuds.

Se faisant Monsieur Hermenier nous à orienter vers un développement répartie en milestone, centre d'intérêt propre à chaque fonctionnalités que nous développerons un peu plus loin dans ce rapport.

Chapitre 2 Etat de l'art

En ce qui concerne les exploitations de données générées par les plateformes en cluster, les programme avec une visualisation en treemap sont aujourd'hui ce qui vont de pairs avec les clusters pour les analyser (panne, disponibilité,…).

Pour amener au sujet plus pertinent qui nous concerne de prêt, le monitoring en lui-même, il existe déjà quelques outils capables de donner des informations sous divers formes.
Pour en citer deux des plus en vogues, Ganglia et Nagios.

En effet Ganglia est un outil qui monitore une flopée de serveurs calculent la même tâches et Nagios est capable de monitorer un tas de choses tout en alertant sur des critères bien définit au paravent.

Mais ces logiciels ont des points négatifs, Nagios fournit des graphique quelque peu imprécis par moment, quand à Ganglia disons qu'il est spécialisé dans un domaine et ne dépasse pas son cadre d'utilisation mais justement aujourd'hui au niveau reporting il nous faut un outil assez fluide pour avoir à la fois une vue global des zones observées et aussi être capable en quelques secondes d'aller voir où le problème se situe.

2.1 Ganglia

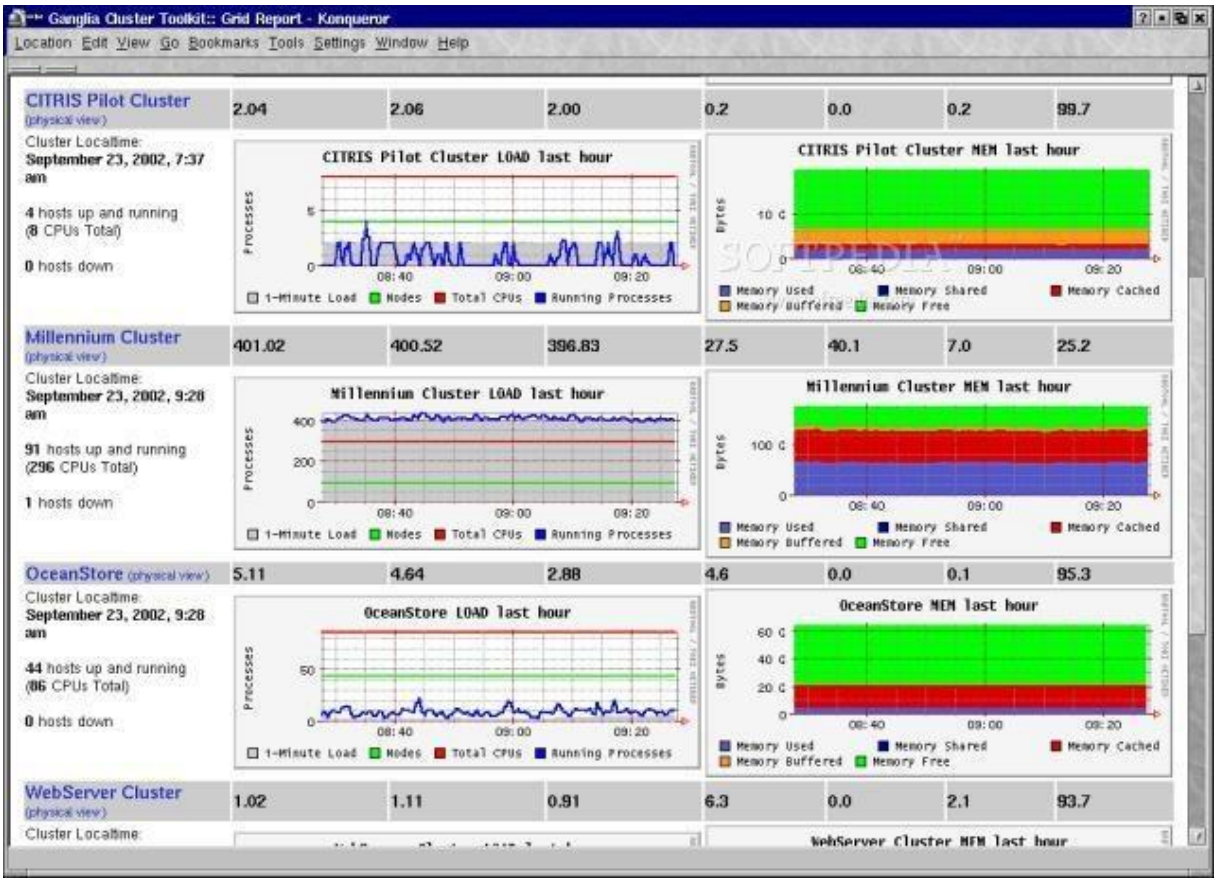


Figure 2.1 : *exemple de monitoring avec Ganglia*

A voir cela on peut se rendre compte que ce logiciel est complet mais au niveau de la visualisation nous n'avons pas une vue global sur l'ensemble observé, seulement des informations sur un cluster mais pas d'informations précise au premier coup d'œil sur l'état des machine virtuelles.

2.2 Nagios

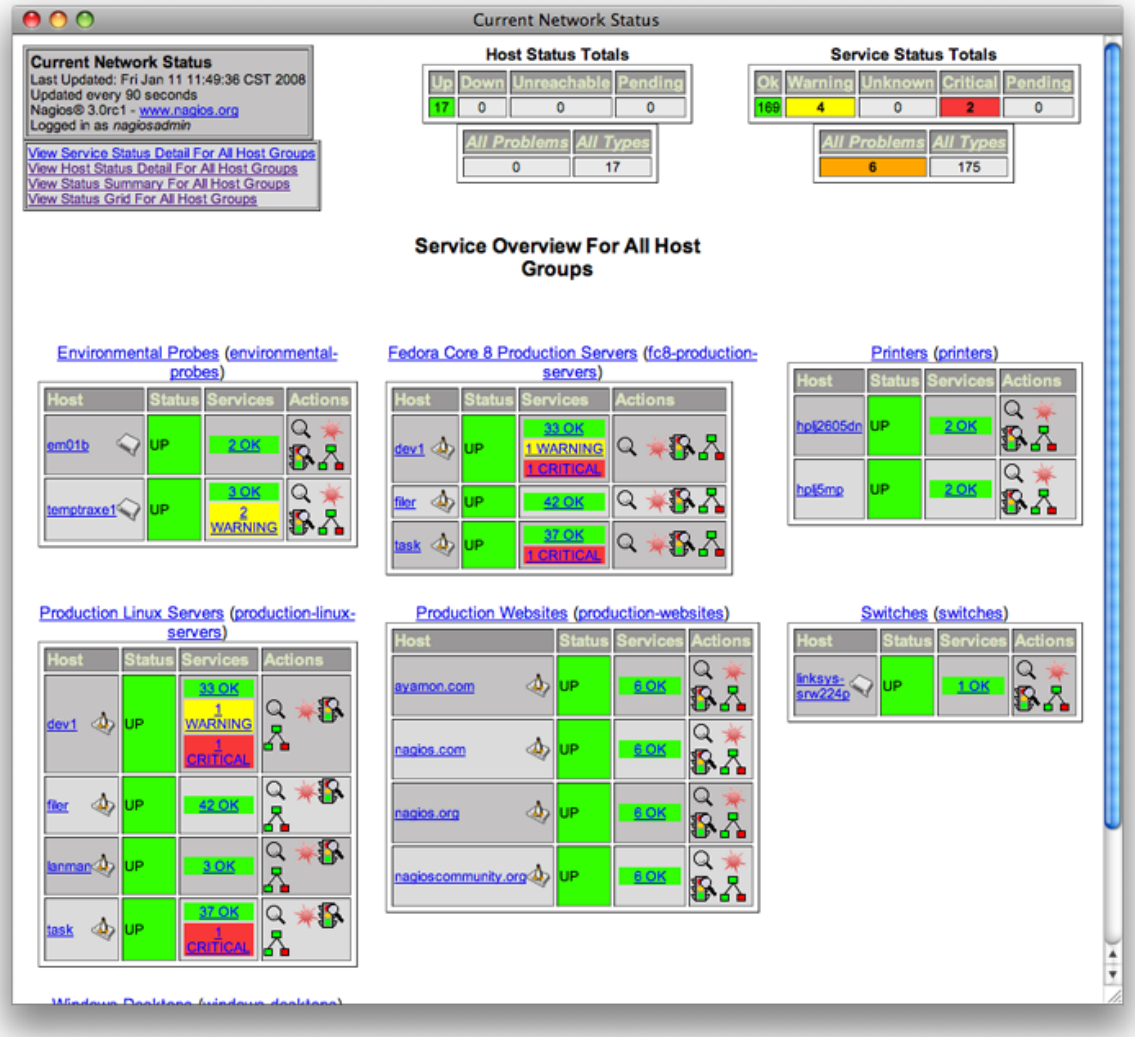


Figure 2.2 *: exemple d'interface de monitoring avec Nagios :*

Comme nous pouvons le constater une interface encore assez lourd pour l'utilisateur qui doit comme même être expérimenter dans le domaine pour intervenir de manière efficace sur les zones qu'il observe.

2.3 Sonar

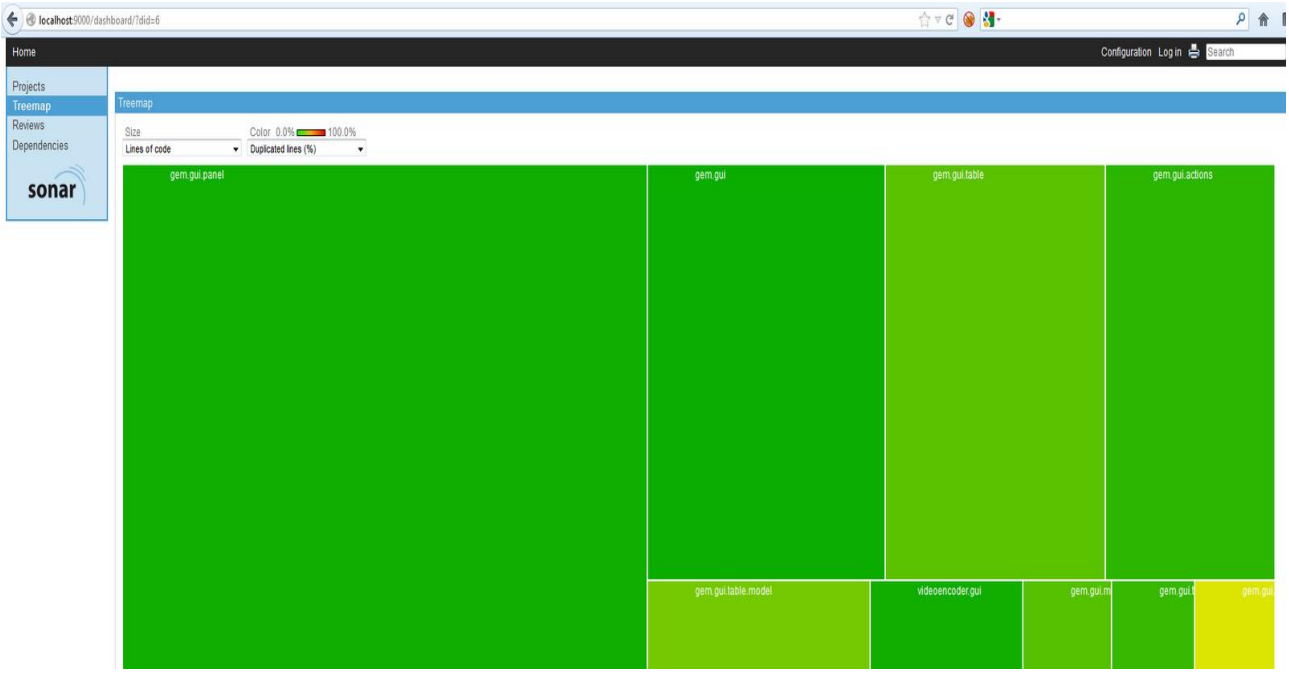


Figure 2.3 : *exemple de monitoring de code avec Sonar (sous forme de treemap)*

Au regard de logiciel tel que sonar, un outil capable de mesurer la quantité de code source sur des projets java, notre encadrant voulais quelque chose allant dans le même sens pour ce qui est du monitoring et du pilotage.

En effet le reporting se fait de manière très visuelle, ce qui ne demande pas de grand effort de recherche de la part de l'utilisateur.

2.4 Disk Inventory X :

Pour citer un autre style d'utilisation des treemap pour visualiser des informations, il existe un utilitaire sous MAC qui s'appelle Disk Inventory X

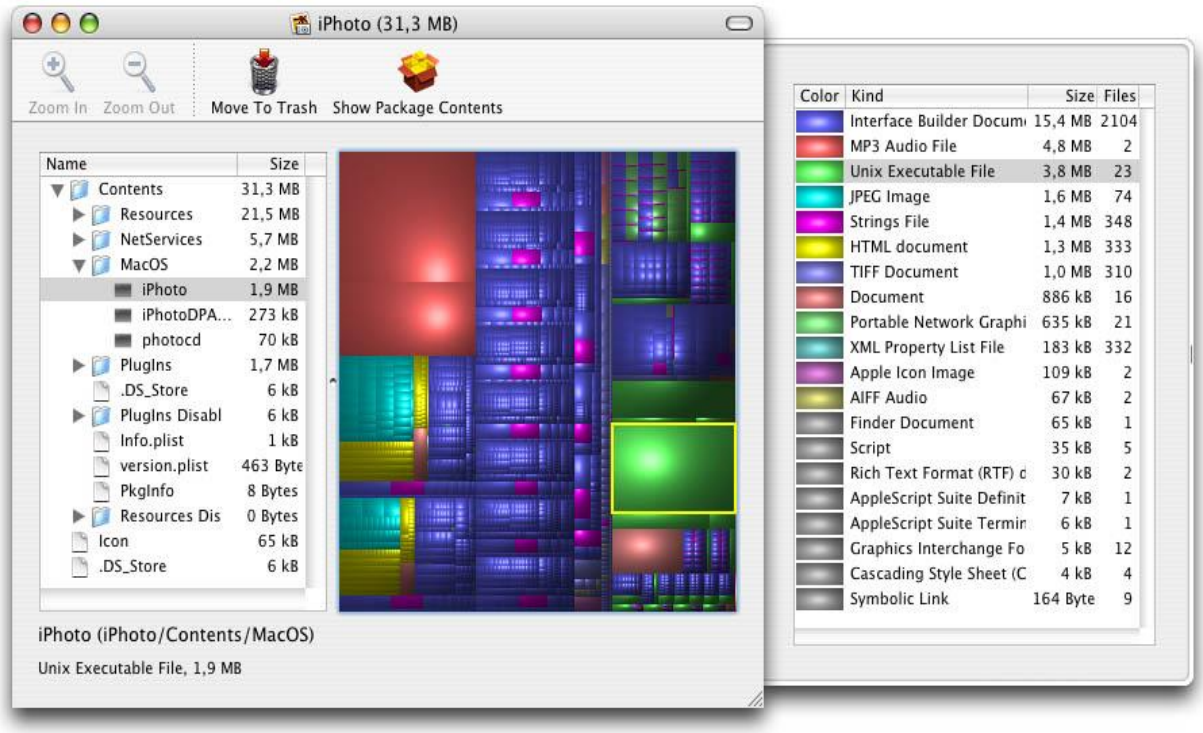


Figure 2.4 : *Interface de Disk inventory X*

Encore une fois on voit tout de suite l'avantage d'une visualisation en arbre hiérarchisé (treemap), on peut voir qui consomme le plus d'espace disk mais il n'y a pas d'impact direct avec le jeu de couleur, nous verrons que dans Hotplaces nous prenons en compte cela car c'est très important pour gérer des ressources d'avoir une vision pertinente.

C'est donc avec cette idée de base que nous nous somme lancé dans le développement de hotplaces, capable de monitorer avec une granularité permettant d'aller identifier l'erreur à sa source (pas de manière très fine pour le moment)

Avec les phénomènes de bigdata et autre du même acabit nous comprenons vite que de nos jours il est vital de pouvoir exploiter correctement les données qui existe autour de nous, dans un cluster il en est de plus important car il faut être capable à tout moment de savoir si oui ou non un machine virtuelle ou même un site sont en panne ou ont un problème quelconque lié à une contraintes quelle qu'elle soit.

Tout repose sur la visualisation des informations qui vont permettre à l'interlocuteur de comprendre grâce à ses capacités de perception.
Les techniques qui sont le plus fréquemment utilisée sont celle qui représente les données via une projection de plan, l'organisation en arbre, l'organisation en réseau ou en graphe, ces dernières étant spécialisées dans la représentation conjointe des données et de leurs relations potentielles.

Hotplaces tente en sachant cela de répondre à cette demande tout en étant un outil performant (lié à BtrPlace), sur cette base solide nous tentons de proposer à l'utilisateur une expérience à la fois agréable mais aussi efficace pour observer l'état des machines virtuelles.

Chapitre 3 Présentation générale

3.1 Principe :

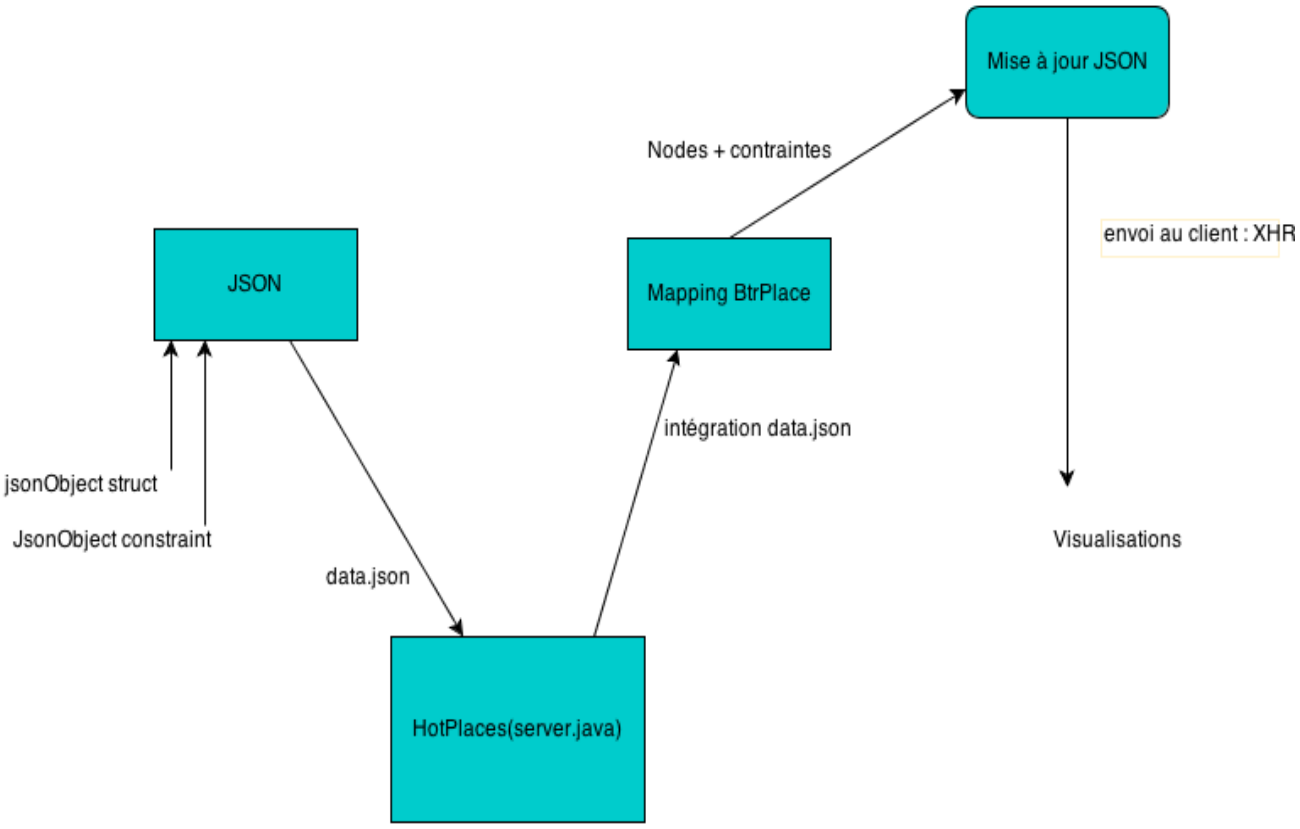


Figure 3.1 : *schéma du principe de l'application Hotplaces*

Notre fichier Json se compose de deux JsonObject, un pour la structure et un pour les contraintes, ce fichier est transmis à server.java de Hotplaces qui va se charger de l'intégrer à BtrPlace pour que celui-ci puisse résoudre les contraintes correctement, il s'agit du mapping, ensuite nous récupérons le fichier Json modifier et affichons différentes visualisations.

3.2 Interface de monitoring Hotplaces

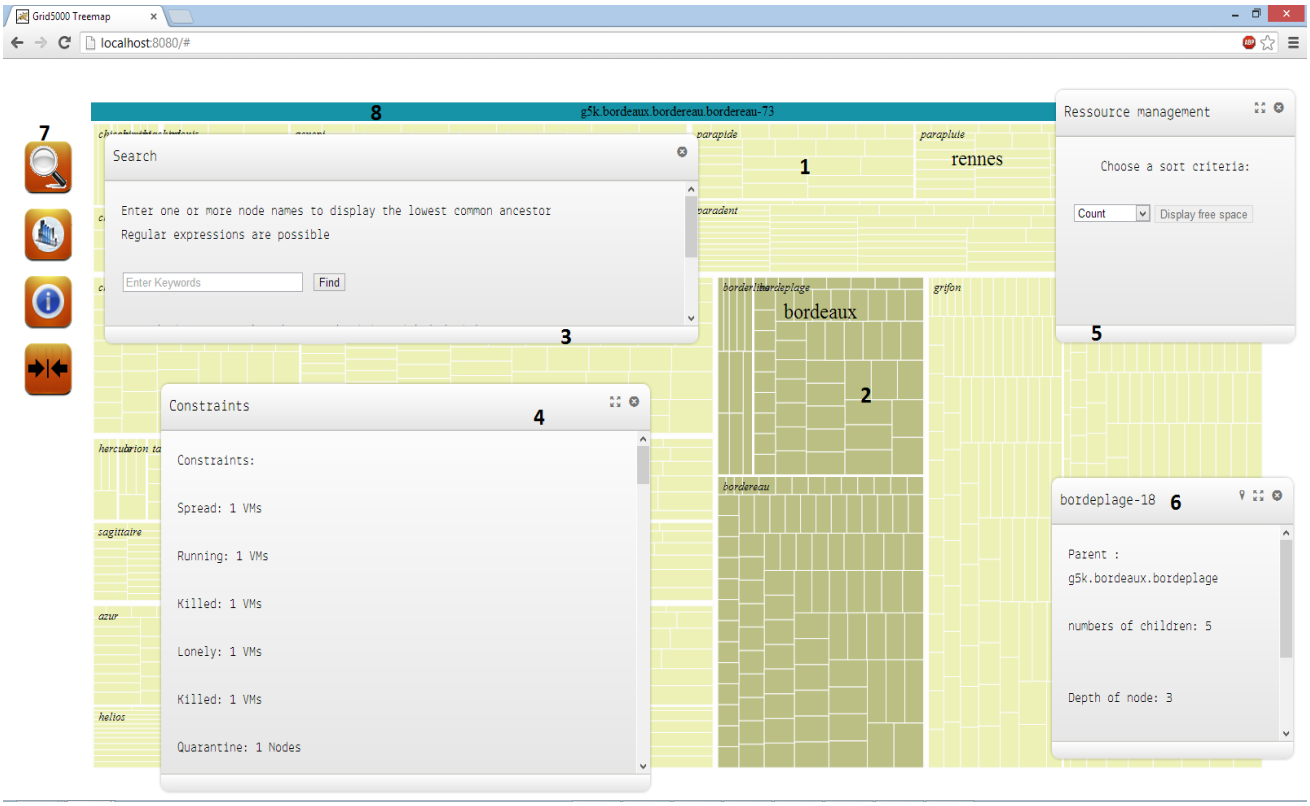


Figure 3.2 : *Interface Hoplaces monitoring amélioré*

Légende

1 : Treemap en elle même
2 : Effet de survole de la zone
3 : pop-up pour la recherche d'un élément
4 : pop-up pour les informations sur les contraintes non respectées
5 : pop-up pour réorganiser la treemap selon différents critères de ressources
6 : pop-up d'infomation via la touche espace ou onglet information contenent des informations sur l'élément survolé avec la souris
7 : Dock qui permet l'accès au différents pop-up
8 : correspond au chemin dans la treemap qui est survolé ou la racine par défaut

Chapitre 4 Environnement

4.1 Environnement logiciel

Diverse technologie entre en corrélation pour l'élaboration de cette web application, en effet il nous a fallu un serveur en local mais aussi une librairie puissante pour représenter toute les données à exploiter, de plus il fallait utiliser le solveur BrtPlace pour récupérer les différentes visualisations qui résultent des contraintes non respecté entre machine virtuelle/cluster

4.1.1 D3 : Data-Driven Documents

a) Description :

D3.js est une librairie javascript créé par Mick Bostock pour manipuler des documents basée sur les données. D3 donne vie aux données en utilisant le HTML, SVG et CSS.
Permet d'être utilisé sur des navigateurs récents, combine des composants de visualisation puissants et une approche axée sur les données à la manipulation du DOM.
Concrètement cette librairie surpuissante nous permet de modéliser n'importe quel flux de données avec des effets de transition très bien réalisés.

b) Présentation:

D3 est une amélioration de Protovis (qui composent avant D3 des vues personnalisée avec des repères simples : bar, points un ensemble de données).
D3 s'est imposé comme librairie de visualisation de par sa puissance et sa capacité à exploiter les données, par exemple il est tout à fait possible de prendre un tableau de nombre et de générer un tableau en HTML, de plus avec le même jeu de données il est possible encore de ré exploiter ces données pour dessiner un histogramme.
Du à sa flexibilité hors du commun D3 permet la réutilisation de code, en minimisant la surcharge D3 est très fluide au niveau de la représentation.

c) Treemap avec D3 :

Nous nous sommes intéressés plus particulièrement à l'implémentation via une treemap avec D3. [1]

En effet suivant l'exemple sur le site de D3, nous avons entrepris sous conseils avisé de notre encadrant la mise en place d'une interface complète exploitant au maximum les capacités d'une treemap, avec des fonctionnalités qui compte parmi elle le zoom, ou encore des fonctions de recherches avancées au sein de notre JSON contenant toute l'arborescence compatible avec Grid5000.

Sur d3.js, c'est une partie à part entière qui permet de gérer tout ce qui concerne la génération et la manipulation d'une treemap, la plus importante étant celle qui permet de dessiner la treeamap : d3.layout.treemap

4.1.2 BtrPlace :

BtrPlace est ce avec quoi nous communiquons pour récupérer ce qu'il calcul à notre place, notamment sur le respect de contraintes diverse, par exemple « Ready » qui va forcer une ou plusieurs machine virtuelle à prendre le statut ready.

Donc pour être plus exact c'est un algorithme de placement des machines virtuelles sur une plateforme d'hébergement, qui permet une configuration personnalisée via des contraintes sur paramètres des nœuds et machines virtuelles d'un cluster. [4]

4.2 Environnement de travail

4.2.1 Tomcat/Maven

Pour mettre en place la structure de l'environnement de travail nous avons utilisé les outils qui s'imposés et était imposés, Tomcat pour le serveur en local ainsi que Maven comme gestionnaire de projet (path, jar, war,…) qui permet de déployer la web Application sur le serveur local.

Pour nous c'était donc une première notamment pour la mise en place de ceux-ci correctement, mais c'est outil se sont révélé très utile car via Maven nous pouvons spécifier facilement des dépendances ainsi que l'ajout de plugin supplémentaire nécessaire au bon fonctionnement de notre application.

Tomcat étant développé dans l'optique des applications web, cet outil nous à montrer l'étendue de son potentiel grâce à sa simplicité d'utilisation.

Maven permet de déployé notre serveur Tomcat facilement via n'importe quelle terminal ce qui nous a permis à chacun de conserver son IDE ou éditeur de code favoris.

Le projet est actuellement hébergé sur Github, très performant pour l'organisation de code, nous en reparlerons dans la partie gestion de projet.

Chapitre 5 JSON

5.1 JSON mock

Nous avons pour développer Hotplaces généré à l'aide d'un fichier python une structure Json mocké, c'est-à-dire une fausse capable d'être interprétée par la treemap de D3 et contenant tout le nécessaire niveau attribut pour chaque membre de la hiérarchie de la grille.

Extrait du fichier mock …

Json utilisé pour milstone 1(définition du design minimum)

5.2 JSON et BtrPlace

Pour ce qui est de cette interaction comme définit dans la figure «3.1 » BtrPlace à son propre format de Json acceptable pour le traitement, il comporte :

5.3 Intégration BrtPlace vers Hotplaces

La grosse partie du projet est celle-là, en effet tout notre interface n'a pour but que d'exploiter correctement ou plutôt permet d'interpréter les reporting que BtrPlace nous fournis sur notre architecture de grille.

Du côté de l'interface il faut être capable en environs une minute même moins de localiser où se trouve le problème que BtrPlace soulève pour nous.

Pour réaliser cela il a fallu modifier pas mal la structure globale de nos fichiers, notamment server.java et le fichier python qui génère la structure du Json, pour le rendre exploitable par BtrPlace il fallait adapter la hiérarchie que suggérer notre Json avec le mapping de BtrPlace, création de nœuds et machines virtuelles et la liaison entre eux.

Ainsi on obtient une première granularité via une première intégration de BtrPlace qui permet d'observer les alertes mais il est possible d'obtenir une meilleure granularité, ce que nous n'avons pas eu le temps d'implémenter.

Extrait jsongen et data.json commenté

Chapitre 6 Gestion de projet

Le TER a débutés le 14 mars avec un jeudi par semaine puis à partir du 31 mai nous nous somme entièrement consacrer au TER

Comme dit plus haut, monsieur Hermenier nous a dit de répartir le travail en échelonnent les fonctionnalités importantes comme suis :

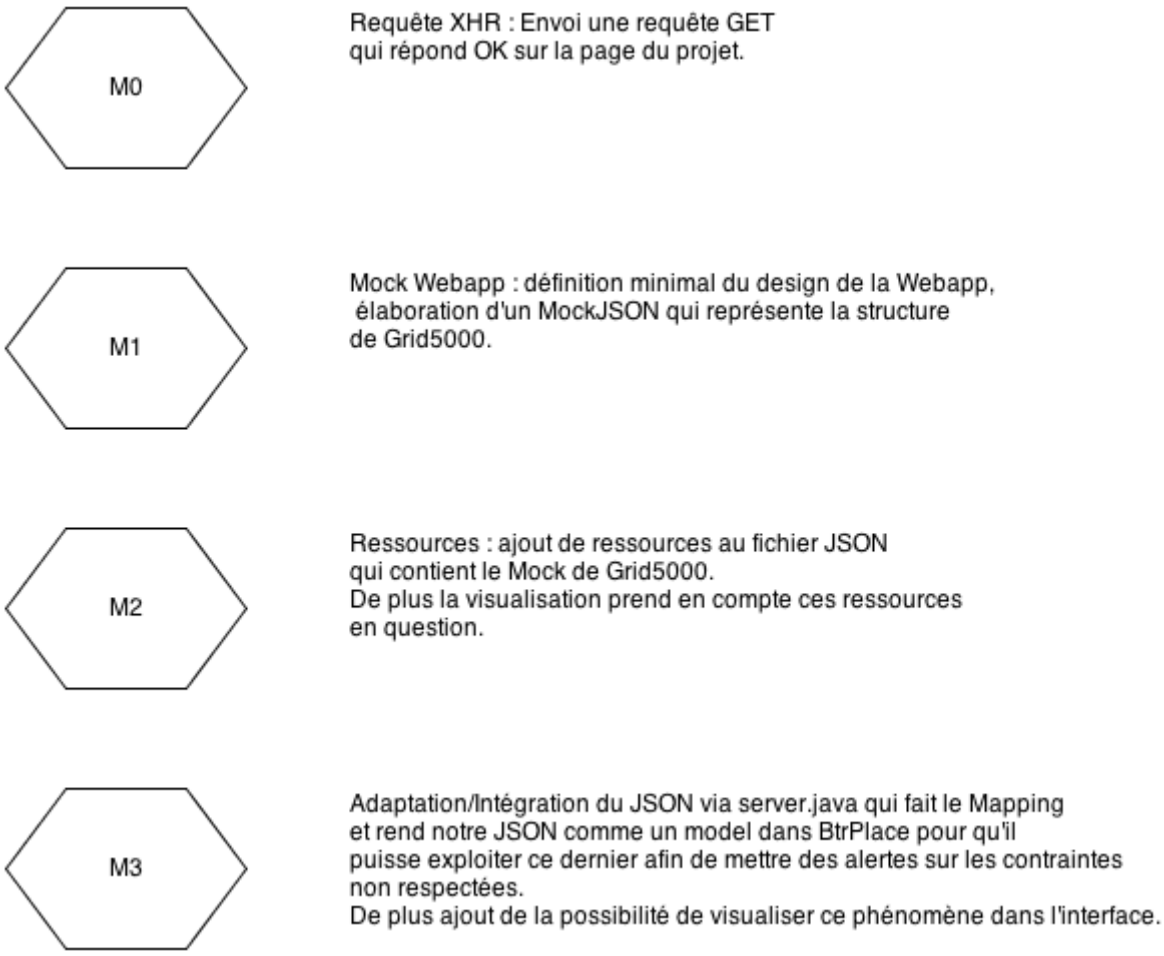


Figure 6 : *milestone ou découpage du projet*

Calendrier prévisionnel de Hotplaces :

M0 : début avril (comprend aussi la mise en place du projet)
M1 : 2/3 jours 1ere semaine avril
M2 : 1/ 2 jours mi-fin avril
M3 : 4 / 5 jours fin mai

Calendrier réel voir Github et les tags dans master

Le calendrier prévisionnel a été difficile à suivre à cause des contraintes de l'emploi du temps.

Le fait d'avoir hébergé le projet sur Github, nous a permis d'avoir une organisation à la hauteur du projet, la création d'issue qui permet de soulever un problème à régler en l'attribuant à un membre du groupe mais encore la possibilité de pouvoir créer des branches qui sont en fait des clones de la principale ou d'une de ses sœurs.

Les branches permettent de se focaliser sur une fonctionnalité en particulier, par exemple la branche treemap qui avait pour but de dessiner la treemap à l'aide de la librairie D3, nous avions pris du retard sur cette fonctionnalité dû à l'apprentissage des quelques nouvelles technologie.

Effectivement nous avions commencé par développer la treemap à l'aide de div en DOM mais on s'est aperçu après avoir obtenu le premier niveau de treemap que pour zoomer dans des états plus profond, c'est-à-dire une vue différente de celle initial où nous voyons apparaître la racine et ses fils contenant eux même des états plus profond nous ne trouvions pas comment conserver la forme car l'approche avec les div n'est pas graphiquement aisé.[2]

Donc nous avons recherché une solution plus appropriée pour le rendu qui nous était demandé et cette solution exister, nous avions trouvé un exemple qui faisait un zoom basique utilisant les svg de DOM à l'aide de transition il nous était possible de définir les contraintes à respecter par le zoom et ainsi obtenir l'effet visuel escompté. [1]

Répartition des tâches :

Pour chaque milestones le travail a été découpé de manière à minimiser les conflits et à ne jamais empêcher le développement des autres parties.
Le choix des parties a été fait en fonction des préférences et des compétences de chacun.

Engilberge Swan :
Hedda Hedi :
OULEHA Nadir :

Résultats :

L'état du projet Hotplaces à la deadline est telle qu'on peut observer ce qui était demander dans le sujet de TER, on a une visualisation en treemap(hiérarchisé), de plus on la possibilité de zoomer sur un groupe de machine virtuelle mais aussi on peut rechercher un élément en particulier, Hotplace est mappé sur BtrPlace pour que celui-ci puisse alerter de toute contraintes non respéctées.
Il est possible de trier par ressources toute la treemap mais encore obtenir des informations sur la machine survolée en appuyant sur la touche espace.
Des liens sont présents pour faciliter la navigation entre chaque élément (dans les bulles d'informations)

Discussions avec l'encadrant :

Discussion avec l'encadrant (comment c'est passé l'encadrement) recadre dans le sujet, conseils, l'état d'esprit d'un chercheur

Celles-ci nous ont permis d'évoluer dans l'avancement du projet, quand on bloqués un peu trop sur une fonctionnalité monsieur Hermenier nous donné quelques conseils notamment sur des techniques pour simplifier la manière de résoudre notre problème.

Si nous nous égarions un peu trop du sujet, nous étions recadré comme il faut par notre encadrant, ce qui permettais de toujours remettre en questions la répartition des tâches et surtout l'ordre de priorité de celles-ci.

En somme nous avons pu observer de prêt comment un chercheur opère quand il doit prendre des décisions ou encore lors qu'il organise un projet en général.

Chapitre 7 Conclusion

En choisissant ce projet nous nous demandions s'il serait intéressant, ce fut le cas car nous avons pu manipuler plusieurs technologies et langages de programmation pour les faire cohabiter dans un même projet.
En effet nous avons eu l'occasion de côtoyer des chercheurs sur leur lieu de travail et voir au sein de l'équipe OASIS que le travail en équipe était le quotidien de chacun à l'INRIA.
Ce projet a permis à chacun d'évoluer dans sa manière d'appréhender un projet en équipe, il est arrivé parfois qu'on fasse fausse route lors d'un choix de mise en place d'une fonctionnalité, cela est à mettre dans les impondérable à notre stade d'étudiant mais nous tendons à devenir bien plus efficace quand on prend en compte l'expérience accumulée sur un projet d'une telle ampleur.
A l'heure actuelle l'état du projet permet à ce qui en hériterons de pouvoir améliorer la granularité des alertes levées car pour l'instant nous avons juste un binaire qui nous informe si oui ou non une contraintes n'est pas respectée, c'est donc un approfondissement de l'aspect concernant les contraintes que BtrPlace envoi à Hotplaces.
A notre avis pour mettre en place cette fonctionnalité

8 Webographie :

[1] *http://bost.ocks.org/mike/treemap/*
[2] *http://bl.ocks.org/mbostock/4063582*
[3] http://btrp.inria.fr/sandbox/
[4] https://github.com/mbostock/d3/wiki