Varnish / APC / Memcache

There are only two hard things in Computer Science: cache invalidation and naming things.
-- Phil Karlton

Varnish

Reverse-proxy -> element entre internaute et serveur
peut faire

- - cache
- - redirections http
- - routage
-

Comment purger le cache ?

On peut mettre une durée de cache spécifique pour des pages (exemple, 1j pour un sitemap, 5minutes pour la front... etc). Mais ce n'est pas forcément adapté pour tous les sites/pages ; on peut vouloir avoir des bouts de page avec des durées ou politiques de cache différentes (quand authentifié par exemple les pages sont différentes entre les différents users : newsfeed, theme specifique...)

Les durées de cache peuvent se déterminer dans les entêtes HTTP (cache-control), ou en dur via varnish, mais c'est moins cool, car par exemple, non administrable.

On peut aussi vouloir purger le cache sur des actions. Par exemple, à la publication d'un node. Pour cela, envoyer une requête à Varnish, en implémentant un hook.

Les headers sont le language principal qui permet à Varnish de dialoguer avec le navigateur et Drupal
La majorité des tags utilisée par Varnish commencent par "X-". Cela permet, notamment, de dire qu'un fichier de cache contient les nodes avec tel id (par exemple), pour ensuite lui dire de vider toutes les fichiers de cache contenant ces nodes, s'ils sont modifiés.

Le cache authentifié

Regrouper le contenu par population, par theme
Pour gérer le tagging de logique, on va utiliser les ESI (Edge Side Include, Edge pour proxy-server)
Politique de cache différente par "zone" dans une même page HTML
Spécification W3C poussée par Akamai
fonctionne avec des namespace xml <esi> dans le code html
exemple pour une structure trois colonnes : <esi:left> sur un un noeud xml de la page

Necessité de montrer les maquettes au dev avant d'attaquer le dev, pour qu'il commence à réflechir aux problèmatiques de cache (techno utilisée... voir revoir la conception) peut-même se faire sur le wireframe

Twig implémente nativement les ESI au sein de son système

Gestion des cookies
On peut dire à varnish de retirer les cookies pour tricher et demander à Drupal une page ou une partie anonyme puis jouer avec les ESI pour récupérer les blocs authentifiés

Attention varnish + drupal:

- Variable de session (pour les anonymes du coup pas de cache)
- Plutot travailler avec les cookies JS
- attention mauvaise utilisation des requetes HTTP par les modules contrib (exemple flag qui utilise un GET pour mettre à jour un flag) => du coup il faut mettre une exception
- on ne peut pas renvoyer/passer un cookie depuis un ESI

Qui est en charge de faire cette config ? Dev ? Sysadmin ? Devops ?
La config est à versionner absolument avec log des modifications.

Varnish vs Nginx

Varnish ne supporte pas le https
On met les deux à dialoguer entre eux en HTTP et Nginx en frontal pour gérer le HTTPS : internaute -> nginx -> varnish -> nginx/apache+drupal (attention, dans ce cas, à bien configurer Drupal pour qu'il génère bien les urls en https)
nginx peut cacher les assets static, pour soulager varnish

Gestion de l'espace

Varnish élit seul le cache à expirer selon sa dernière utilisation et d'autres critères (garbage collector)
-> Les caches nuked sont à monitorer car se sont les caches supprimés complètement aléatoirement lorsque le garbage collector n'arrive plus à élire un cache à expirer. Si trop nombreux, il faut envisager d'augmenter l'espace de varnish ou revoir les règles de cache.

APC

Bytecode cache -> cache le PHP compilé
Cache metier

Opcode

Opcode fait comme APC, mais en plus granulaire.
Par contre il faut un php5.4 mini sinon APC est plus performant.

Memcache

Peut servir à mutualiser des données pour un multi-backend PHP

Cache de données

Cache de sessions

/!\ bien vérifier avec des outils de metrics pour vérifier que varnish est bien configuré.