Comment je me suis retrouvé à devoir garantir qu'un état métier puisse traverser l'Atlantique en 1 ms.

TLDR ; *Face à des problèmes techniques compliqués ou difficilement solvables, nos réflexes de DEV peuvent parfois nous jouer des tours et nous faire passer à côté de solutions simples. Le DDD nous permet d'envisager ce type de solutions en allant chercher l'inspiration plus souvent qu'à l'accoutumée auprès du métier.*

Le change pour les gros

Je me souviens d'un cas il y a une douzaine d'années où une simple discussion avec le business avait permis de résoudre un problème technique a priori insolvable (lié à la vitesse de la lumière ;-) Je travaillais à l'époque pour une grande banque d'investissement.

Lorsque vous êtes une grosse société ou un investisseur et que vous souhaitez convertir un gros montant d'une devise (ex : 3 Millions EUR) dans une autre (ex : USD), vous passez en général par une bourse spécialisée dans ce type d'opération sur le marché des changes. Lorsque vous êtes un gros investisseur ou que vous avez très souvent ce genre de besoin, vous pouvez même mettre en place une connexion dédiée avec votre fournisseur préféré (comme la banque pour laquelle je travaillais) pour obtenir de meilleurs prix, plus vite.

Mes utilisateurs à l'époque étaient les market makers qui s'occupaient de l'animation sur le marché des changes. En d'autres mots, c'était des traders spécialisés qui proposaient à la vente et à l'achat des actifs (des devises majoritairement) toujours au bon prix. Contre cette activité pas forcément rentable mais nécessaire pour l'économie, la banque obtenait d'autres droits d'accès par ailleurs. C'était donc une activité importante pour la banque. Un des enjeux de cette activité est notamment de répondre à toutes les demandes de conversion de devises, d'être fluide et rapide dans l'animation, sans pour autant perdre trop d'argent face aux prédateurs (certains investisseurs, parfois d'autres banques).

Pour faire leur travail, ces traders avaient besoins de différents outils :

- **Des pricers** qui réajustent et calculent des prix en permanence en fonction d'indicateurs de marché en entrée qui servent de références
- **Des systèmes de streaming/contribution** qui vont pousser sur les marchés ces prix calculés par nos pricers plusieurs fois par seconde (en rajoutant une marge en fonction de la catégorie des clients qui se trouvent en face, on appelle ça le Tiering)
- **Des systèmes de streaming de prix à la demande** et de négociation automatique pour certains clients qui en font la requête (on appelle ça des Request for Stream - RFS)

- **Des systèmes pour gérer les exécutions**, c.à.d. de l'acceptation ou non d'une demande de deal d'un client sur la base d'un prix envoyé au préalable

- **Et tout un arsenal de systèmes de contrôles** nécessaires lors de l'exécution pour éviter les fraudes, les abus et les excès de la part de certains clients ou concurrents

Bien entendu, au vu de la vitesse à laquelle ces calculs et ces traitements devaient être réalisés (de quelques centaines de microsecondes à quelques millisecondes seulement), ceux-ci sont totalement automatisés à travers des systèmes que nous leur développions et que les market makers configuraient et pilotaient. Le débrayage du mode automatique vers une intervention manuelle d'un trader qui était prompté sur le desk était très rare et intervenait uniquement pour des cas extrêmes ou critiques (des quantités énormes, une devise exotique sollicitée ou n'importe quelle autre situation identifiée comme à risque).

Du SOA basse latence et Réactif

A l'époque, j'étais l'architecte en chef pour concevoir la nouvelle plate-forme de market making sur le marché des changes et nous développions tous ces différents « services » à l'aide de plusieurs équipes de développement (pricing, exécution, connectivité, credit-check, deal capture, etc.). J'avais poussé une approche orientée service, mais avec des protocoles, de l'infra et des implémentations low-latency (sub-milliseconde). Au passage, nos systèmes épousaient déjà pleinement le modèle réactif et complètement événementiel qui devient de plus en plus à la mode de nos jours, notamment dans la sphère DDD.

De manière très concrète, chaque mouvement d'indicateurs clés sur les marchés que nous observions déclenchait des re-calculs au niveau des pricers concernés. Chaque nouveau prix recalculé était ensuite poussé vers les clients intéressés et déjà abonnés dans un mode de consommation « annule et remplace ». Un peu comme si nos traders envoyaient en permanence de nouveaux mini devis (les prix) aux clients abonnés. À tout moment un client pouvait décider de nous solliciter pour une exécution à partir d'un des prix qu'il avait reçu.

Contrairement à certains secteurs ou marchés où l'on peut envoyer des ordres ou des prix fermes (c.ad. Qui engagent obligatoirement celles et ceux qui les envoient), les market makers sur le marché des changes se donnent un droit de regard, de contrôle et une possibilité de refus à chaque demande d'exécution émise par un client. En effet, sur les centaines ou les milliers d'updates de prix qu'on envoie à un client, pas sûr qu'on ait toujours envie de faire un deal sur un de nos vieux prix si le marché s'est « retourné » entre-temps.

L'exécution, une phase critique

L'exécution est une phase extrêmement critique où il faut aller vite (idéalement sous la milliseconde, mais pas plus de 3 ms en tout, latence réseau comprise). Si on est trop lent pour procéder à cette exécution, on prend alors le risque de passer notre temps à refuser des exécutions à nos clients parce que les prix du marché ont trop bougé entre le moment où il a reçu votre prix et le moment ou vous lui répondez.

Pour voir l'effet que cela peut procurer au client, imaginez-vous seul dans une zone désertique en plein été. Il fait extrêmement chaud. Vous avez terriblement soif et un distributeur de boissons fraiches qui se trouve là par hasard vous propose une grande bouteille d'eau pour 5 centimes. 5 centimes ! De l'eau fraiche. Vous êtes heureux comme tout, vous mettez votre pièce de monnaie... et la machine refuse de vous vendre la bouteille tout en vous rendant votre pièce. Vous réessayez : pareil... vous réessayez : encore un refus de la machine, et encore et encore... Et bien c'est comme ça qu'on perd des clients pour toujours ;-)

Vous l'aurez compris, nous n'avons donc que quelques millisecondes pour orchestrer en parallèle un ensemble de contrôles, de mécanismes de protection et de vérifications avant de pouvoir dire oui au client. Parmi ceux ci :

- **Le Last Look :** où l'on vérifie que notre prix est toujours en phase avec le reste du marché en les comparant (pour éviter de se faire allumer par les prédateurs)
- **Le KYC & black listing :** pour vérifier si le client n'est pas un mauvais payeur ou un infréquentable avec qui on ne veut pas faire affaire (embargo, terrorisme)
- **Le Rate limiting :** pour vérifier que ce client n'est pas en train de nous assommer avec trop de demandes d'exécution consécutives. Dans le cas contraire, on lui refuse certaines exécutions si elles sont trop proches les unes des autres.

Le Credit Check, une étape clé

Enfin, on doit également vérifier que le client, par cette exécution, ne va pas consommer plus que les lignes de crédits maximales autorisées pour lui auprès de notre banque (pour éviter les abus en provenance d'un seul client). En caricaturant a l'extrême, on peut voir une ligne de credit d'un client auprès de notre banque un peu comme une sorte de montant maximum pour un découvert autorisé.

Et bien le service que notre orchestrateur d'exécution doit solliciter pour faire ce type de vérification s'appelle **le Credit Check**.

Celui-ci a la responsabilité de répondre à la question : étant donné tout ce qui a déjà été exécuté par ce client auprès de nos services, mais aussi de sa ligne de crédit maximale autorisée restante, a-t-on encore le droit de faire un deal de x USD avec lui ou a-t-il déjà épuisé toutes ses lignes de crédit chez nous ?

Si oui, l'orchestration continue et le deal a lieu (en plus de diminuer au passage la limite de crédit restante pour ce client auprès du Credit Check service). Si non : on retourne un refus d'exécution auprès du client en lui indiquant qu'il a dépassé ses limites de crédit auprès de notre établissement.

En résumé donc, en plus de répondre à toutes ces questions en moins de 3 ms, on a un état des lignes de crédit restantes par client à maintenir de manière fiable, rapide et transactionnelle dans nos systèmes.

La Terre, un vaste terrain de jeu…

Ça commence à devenir plus compliqué quand on se rends compte que le même client (un grand groupe international par ex) peut nous exécuter simultanément sur différents continents...

En effet sur le marché des changes, nos plateformes sont ouvertes 24/24h - 6.5/7d et ont des points de connectivités partout dans le monde (New York, Londres, Hong-Kong). Concrètement, cela permet à nos clients qui auraient eux aussi des systèmes sur toutes ces places, de nous faire simultanément des demandes d'exécution sur la même paire de devise (EUR/USD par ex) à NY, à Londres et à HK. Pour simplifier les explications suivantes, on fera comme si on n'avait que 2 zones de connectivités à gérer ici : NY et Londres.

Pour rappel, **la latence pour un aller-retour entre Londres et NY quand tout va vraiment bien pour vous est un peu en dessous 60ms**. Ça fait donc presque 60 fois plus que la latence maximale autorisée pour chaque demande d'exécution unitaire... à caler comme une des étapes de celle-ci...

Le problème devient donc : comment maintenir un état global mutable et partagé mondialement des limites de crédit de ce client, tout en gardant des temps de réponses acceptables pour ses exécutions (c.ad. Ici de l'ordre de la milliseconde) ? Dans quelle région doit-on positionner le composant du Credit Check qui contient cet état global-monde des limites de credit de nos clients ?

Compliqué hein ? (Je vous avais prévenu ;-)

Bon en fait, le problème n'est pas de maintenir un état mutable partagé mondialement pour ce client en défiant les lois de la physique et la vitesse de la lumière, vous l'aurez compris. Ça c'est ce qu'on peut être amené à se poser comme question quand on est fatigué ou quand on se piège soi-même dans un modèle mental un peu trop simple (et qu'on essaie d'optimiser à fond chaque partie au lieu de réévaluer le problème à la base). "Et si on essayait de mettre à jour l'état distant du Credit Check de nos clients en essayant l'intrication quantique ?" ;-)

Non, le problème ici est plutôt de trouver un moyen d'éviter le dépassement des lignes de crédit pour un client, malgré les exécutions qu'il peut nous lancer en parallèle sur différents sites (Londres et NY par exemple), tout en continuant de faire en sorte que nos systèmes sur chaque site ne soit pas ralentis et répondent bien à chaque demande d'exécution en moins de 3ms.

Pour ce faire, ne cherchez pas plus longtemps un moyen ou une optimisation technique. C'est du côté du métier qu'il faut aller chercher une astuce ou un compromis qui soit suffisant et acceptable pour résoudre notre problème.

Euh… il faut qu'on se voie

Car le véritable problème que l'on souhaite éviter, c'est en discutant un peu plus avec le métier qu'on le découvre en fait. Comme toujours.

En l'occurrence ici, ce que l'on souhaite éviter absolument avec notre service mondial de Credit Check, c'est qu'un client au bord de la faillite s'arrange pour envoyer 2 grosses demandes d'exécution au même instant sur les deux continents, pour nous laisser toute sa dette sur les bras

avant même qu'on ait eu le temps de se rendre compte qu'il nous a dupé (en nous faisant exécuter 2 fois son montant maximal autorisé). Une race condition qui peut nous couter cher si elle survient... Et formulé comme ça, ça devient tout de suite plus accessible. Alors, comment éviter ça ? Je vous laisse chercher quelques minutes avant de poursuivre votre lecture (...)

Bon. Vous avez eu le temps d'y réfléchir un peu ? Il est temps pour moi de vous livrer ce qu'on avait trouvé à l'époque, pour tenir nos objectifs.

Un compromis suffisant

Puisqu'on ne peut -a priori- pas défier la vitesse de la lumière (ni encore exploiter l'intrication quantique;-) on a diminué les risques autrement en divisant le montant total qu'un client a le droit d'exécuter chez nous par le nombre d'endroits où il peut nous exécuter et où l'on va déployer notre service de Credit Check pour conserver une latence acceptable localement. Ce faisant, si le client nous exécute en parallèle sur chaque continent, il ne pourra que dépenser au mieux l'intégralité de sa limite dédiée sur ce continent. Mettons que la limite totale de crédit de ce client chez nous soit de 100 Millions d'Euros. Si on a 2 sites pour simplifier : NY et Londres, on divisera ce montant restant pour lui en 2 : 50 Millions max exécutables à NY et 50 Millions max exécutables à Londres.

Cette répartition d'un état global pour un client sur plusieurs subdivision locales du credit check et de ses limites de credit, combiné à la protection d'un rate limiting local, tout ceci nous laisse finalement suffisamment de temps pour synchroniser et consolider notre nouvel état global mondial du Credit Check pour chaque client (compter moins de 100 ms entre deux exécutions sur la même place géographique donc). Tout ceci est plus qu'acceptable pour l'activité des market makers.

Ça a l'air tout simple dit comme ça, mais cela avait été assez difficile pour nous à l'époque de sortir du cadre de la réflexion d'une optimisation sur le plan purement technologique, et de réfléchir à autre chose qu'un naïf : « comment réduire la latence intercontinentale pour faire tenir la synchro du crédit check » sous la milliseconde ;-)

Bien entendu cette décision n'a pu être prise que parce que les impacts métier de celle-ci ont été évalués et acceptés par nos interlocuteurs métiers. Il s'agit d'un compromis explicite pour tout le monde du côté de la banque. Pas d'une décision prise unilatéralement par l'IT.

Comme au début de notre histoire de Credit Check, un problème mal posé peut avoir de lourdes conséquences. Comment espérer dans ce cas proposer une solution qui soit adaptée et non une usine à gaz technique juste au cas où le besoin serait peut-être un peu de ci, un peu de ça...

Conclusion

Ici, c'est en discutant plus sérieusement avec le métier que nous avons pu prendre conscience des véritables enjeux (les 5 Whys encore une fois ;-) et trouver ce compromis astucieux qui nous a évité de sérieuses pertes opérationnelles. Bon dans ce cas, la recherche d'une solution purement technique n'a pas duré très longtemps car nous savions toutes et tous que nous étions contraints par une constante forte de l'univers ;-) Mais cette anecdote est néanmoins un exemple parmi d'autres d'interaction plus étroite entre le métier et le Dev, qui débouche sur un

compromis qui convienne à tout le monde. Le genre d'interactions qu'il faut tacher d'avoir le plus souvent possible.

Depuis toujours attaché aux usages et à l'espace du problème, le DDD a été une confirmation pour moi. Un allié inestimable pour continuer d'essayer d'aligner mes solutions avec les véritables problèmes des gens pour qui je travaille. Et c'est une lutte continuelle à mener, tellement nos modèles mentaux sont parfois distants, nos biais de confirmation et les implicites omniprésents.

Thomas PIERRAIN

eXtreme Programmer et architecte obsédé depuis longtemps par les usages et l'espace du problèmes, Thomas anime depuis plus de 3 ans le meetup #DDDFR et fait partager à ses clients son expérience sur le sujet ainsi que ses autres marottes (event sourcing, bi-temporalité, programmation réactive, multithreading, architecture, middlewares low latency, Outside-in TDD, XP, Event Storming...). Fervent promoteurs de l'architecture hexagonale depuis des années, Thomas a aussi beaucoup écrit et live codé sur le sujet, notamment une fois sur scène avec Alistair Cockburn.

Après avoir travaillé 12 ans dans la finance de marché (sur 20 ans d'expérience), Thomas a créé la société 42 skillz avec son associé Bruno Boucard. Après avoir gouté au domaine de la santé hospitalière, Thomas travaille aujourd'hui pour un gros client dans le domaine de l'hôtellerie.

Références :

- Reactive Manifesto: https://www.reactivemanifesto.org
- Pragmatic architecture: http://tpierrain.blogspot.com/2013/04/the-pragmatic-architect.html