**Performance**

|  |  |
| --- | --- |
| **Scénario** | Temps de réponse d’une transaction |
| **objectifs d'affaires** | Le client qui envoie une transaction s’attend à avoir une réponse du guichet dans un délai maximal de 10 secondes sinon il s’inquiètera. |
| **Source** | Le client |
| **Stimulus** | Confirme la transaction (événement sporadique, démarrent après la saisie des paramètres) |
| **Artéfact** | Module de communication avec la centrale |
| **Environnement** | Mode normal |
| **Réponse** | Traitement de la transaction par la centrale |
| **Mesure de la réponse** | * Temps de réponse |
| **Questions** | 1. Combien cela coûte-t-il pour avoir un meilleur temps de réponse ? 2. Avons-nous le contrôle sur toute la chaîne de la transaction (ex. y a-t-il un fournisseur externe) ? 3. Qu’arrive-t-il si le temps de réponse n’est pas respecté ? (Ex. annuler la transaction ?) 4. Combien de temps le client est-il prêt à attendre ? 5. Combien de transactions concurrentes la centrale est-elle capable de supporter ? 6. Est-ce que le serveur supporte un mode transactionnel ? possibilité de *commit / rollback* d’une transaction |

**Tactique:** Limiter le temps d’exécution avant d’annuler la transaction

**Description** : Imposer une limite du temps de réponse à 10 secondes entre le guichet et la centrale avant d’annuler la transaction et afficher un message d’erreur au client.

**Justification** : Le lien de communication TCP/IP pourrait être coupé à tout moment ou la centrale pourrait être surchargée par un nombre trop élevé de transactions en même temps. Dans ce cas, il faut interrompre la transaction et éviter que le client ne s’inquiète inutilement.

|  |  |
| --- | --- |
| **Scénario** | Réactivité de l’interface graphique du guichet |
| **objectifs d'affaires** | Le client s’attend à ce que les interactions avec le GUI soient réactives dans un temps moyen de 1 seconde |
| **Source** | Le client |
| **Stimulus** | Appuie sur un bouton du guichet |
| **Artéfact** | Module UI du guichet |
| **Environnement** | Mode normal |
| **Réponse** | Affichage d’une réponse à l’écran |
| **Mesure de la réponse** | * Temps de réponse moyen |
| **Questions** | 1. Combien de processus le guichet est-il capable de gérer en même temps ? 2. Que doit afficher le module UI lorsque les autres processus n’ont pas terminé ? 3. Que peut faire le client pendant qu’une transaction est en cours ? |

**Tactique 2**

**Tactique:** Ajouter des processus à la file d’exécution

**Description** : L’ajout de processus en parallèle permet de traiter plusieurs opérations en même temps (ex. l’affichage à l’écran et communication avec la centrale). Le 2e processus pourrait afficher un message d’attente et permettrait au client d’annuler la transaction à tout moment.

**Justification** : L’ajout d’un 2e processus spécifique pour la gestion de l’affichage (UI) permettrait d’améliorer le temps de réponse perçu par le client. Cela améliore également la facilité d’utilisation.

|  |  |
| --- | --- |
| **Scénario** | Latence d’une transaction de retrait entre le début et la fin de la transaction. Le début commençant lors d’une requête depuis le GAB vers la centrale et la fin étant le retour de l’information depuis la centrale jusqu’au client. |
| **objectifs d'affaires** | S’assurer que la latence de la transaction de retrait respecte les délais prescrits par le SLA prévoyant que 99.9% des requêtes s’effectuent en moins de 500 millisecondes. |
| **Source** | Externe, utilisateur. |
| **Stimulus** | -Demande de retrait après avoir choisi son compte et avoir entré le montant de retrait.  - Événement sporadique. |
| **Artéfact** | - Module logiciel de vérification du montant d’argent disponible dans le GAB.  - Réseau entre le guichet et la banque.  - Centrale de la banque : module logiciel de vérification du montant permis. |
| **Environnement** | * Normal * Peek load du réseau et de la centrale de la banque * Overload du réseau et de la centrale de la banque |
| **Réponse** | Transactions effectuées |
| **Mesure de la réponse** | - Latence de la transaction de retrait(via la latence des opérations des différents artefacts impliqués)  -Variation de la latence  - Caractéristiques des évènements en échec. |
| **Questions** | 1. 1: Quelles sont les latences moyennes et maximales de chacun des artéfacts ?  1: 2: Quelles sont les délais de latences prescrits dans le SLA?  2. 3: Quelles sont les mesures à prendre en cas de dépassement des délais de latence?  4. 4: Peut-on prévoir les problèmes de latences, périodes de peak load ?  5. 5: Est-que d’autres mesures que celles du réseaux permettent de diagnostiquer les problèmes de latence ? |

**Tactique: Réduire la surcharge de calcul, Augmenter les ressources.**

**Description**:

* **Surveiller les ressources des systèmes pouvant impacter la latence.** Permet d’évaluer si la performance du réseau correspond aux attentes au travers d’une multitude de systèmes interdépendant. Les ressources peuvent être le CPU, la mémoire vive, le nombre de requêtes effectués par l’application par seconde, etc.
* **Réduire la surcharge de calcul.**Réduire la consommation en ressources si elle n’est pas nécessaire pour gérer les flux d'événements afin de minimiser la latence du système. Dans le processus de gestion de plusieurs requêtes simultanées, l’utilisation d'intermédiaires est réduite à son minimum ou tout simplement supprimée.
* **Augmenter les ressources**: Augmenter les caractéristiques et la puissance des processeurs du GAB et de la Centrale de la banque et maximaliser la connexion réseau entre ces deux parties.

**Justification**:

* **Surveiller la performance du réseau.** Surveiller les ressources des systèmes permet de prédire la latence avec des régressions linéaires sur les résultats obtenues. Il faut aussi prendre pour acquis que certaines mesure du systèmes tel que les tâches en attente de traitement par le CPU des appareils de bureautiques peuvent être des informations pertinentes aider à comprendre les origines des problèmes de latence. Ainsi, il est possible de générer des alertes lorsqu'une des mesures dont nous savons qui peut impacter la latence dépasse un certain seuil. De plus, il peut-être pertinent d’évaluer la latence afin de savoir si l’organisation s’améliore ou pas dans la tentative de contrôle de la latence par le biais des autres stratégies.
* **Réduire la surcharge de calcul.**  Cette tactique est utilisée pour gérer un flux de demandes de ressources inhabituellement élevé. Satisfaire les clients bien en deçà des SLA augmente les attentes des clients sans toutefois mieux répondre aux règles d’affaire. Ainsi, normaliser les réponses aux requêtes permet de mieux gérer les exceptions lors des jours d’affluences comme les jours de festivités majeures en fonction des attentes des clients. L’utilisation des ressources, CPU, bande passante du réseau, le stockage de données et la mémoire est alors optimisée pour répondre aux SLA mais sans plus. Note : la disponibilité des ressources d’un ordinateur impacte la vitesse de traitement de l’information; par le fait même, ça l'impacte la latence.
* **Augmenter les ressources**: L’augmentation des processeurs du GAB et de la centrale de la banque permet de diminuer le temps d’exécution du module de vérification du montant d’argent disponible dans le GAB et du montant permis dans la centrale. Alors que la maximisation de la connexion réseau permet un échange plus rapide des données entre le GAB et la centrale de la banque. D’ailleurs, cette augmentation de ressource peut-être dynamique afin de favoriser

|  |  |
| --- | --- |
| **Scénario** | SC\_01\_Performance – Latence d’une opération ou transaction financière |
| **Objectifs d'affaires** | S’assurer que les transactions financières, initiées par l’utilisateur, se font dans un délai raisonnablement rapide |
| **Source** | Externe (actions déclenchées par l’utilisateur) |
| **Stimulus** | Requête initiée par l’utilisateur |
| **Artéfact** | Système interne GAB |
| **Environnement** | Mode normal |
| **Réponse** | Latence minimale dans la transmission des données et la réception des réponses (du serveur central) |
| **Mesure de la réponse** | Temps de latence maximale pour l’aller-retour (transmission-réception des données) de 1.5 secondes |
| **Questions** | 1. Est-ce-que les algorithmes de traitement de données (sur le système GABA et sur le serveur central) sont optimisés 2. Existe-t-il des mécanismes de sécurité (‘failsafe’) en place dans le cas où la connexion TCP / IP est perdue avec le serveur central de la banque 3. Quelle est l’habileté du serveur central bancaire à combler des requêtes ou transactions parallèles de manière efficace |

**Tactique: Introduce Concurrency (Manage Resources)**

**Description** : Le but principal de cette tactique est de réduire le temps bloqué d'une demande spécifique [BAS2011] [BAS2013]. Ceci est habituellement introduit en utilisant threads qui traitent des ensembles spécifiques d'instructions indépendamment. L'utilisation de cette tactique se fait souvent conjointement avec des tactiques telles que des événements priorisation [BAS2011] [BAS2013]. L'utilisation de plusieurs threads pour le système permet de traiter simultanément différents processus et les résultats intermédiaires sont ensuite reliés entre eux [BAS2011] [BAS2013].

**Justification** : L'utilisation de cette tactique est justifiée quand un certain nombre de processus peuvent être exécutés en parallèle [BAS2011] [BAS2013]. En ce qui concerne le système GAB, de tels procédés semi-indépendants comme la lecture de la carte magnétique, validation de l'information à la clientèle et de vérifier la présence du réseau Interac. En outre, l'utilisation de la concurrence et les threads permet également au système GAB de coordonner divers procédés.

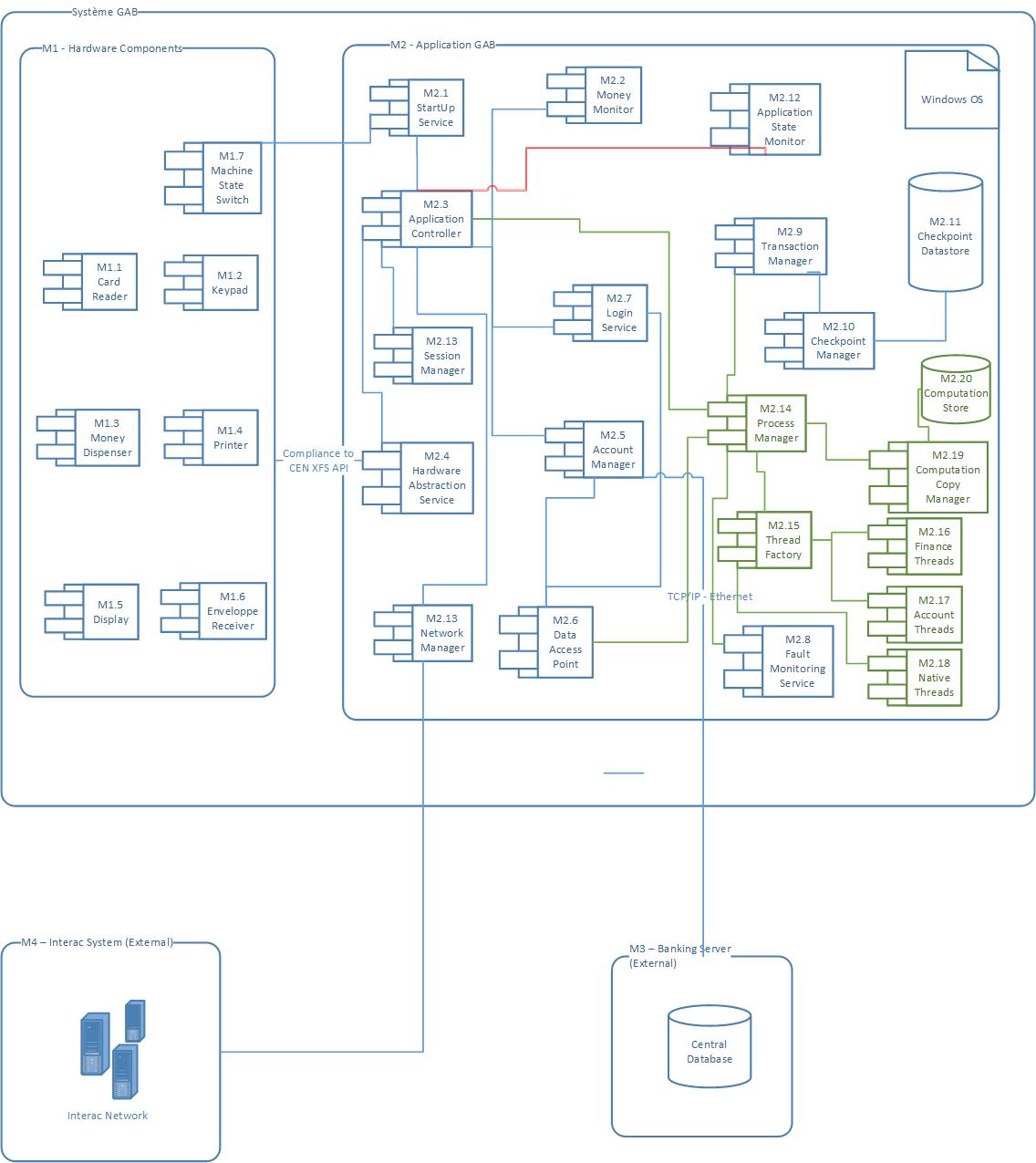
|  |  |
| --- | --- |
| **Scénario** | SC\_02\_Performance – Maintenir plusieurs copies des données calculées (‘computational data’) : Exécution de threads indépendants |
| **Objectifs d'affaires** | Le système GAB doit posséder plusieurs copies (données répliquées) pour aider les processus en parallèles et pour surmonter les petites (‘flicker’) interruptions potentielles des voies de communication TCP/IP. Ceci devient de plus important dans les situations où un accès simultané aux données est exigé (ex. : deux utilisateurs veulent accéder à un même compte) |
| **Source** | Actions d’utilisateurs |
| **Stimulus** | Exigence logicielle (hypothèse) |
| **Artéfact** | Système GAB |
| **Environnement** | Mode normal |
| **Réponse** | Concevoir les modules du système GAB comme les modules de validation des données, de communication et d’enregistrement des données (‘data access layer’) dans le but de diminuer les temps necessaire |
| **Mesure de la réponse** | Soit x le nombre de responsables d’un compte bancaire. Alors, le système GAB doit être en mesure d’exécuter x processus principaux (qui peuvent déclencher à leur tour d’autres threads) se concentrant sur la validation des données, calculs etc. |
| **Questions** | 1. Comment est-ce-que les diverses copies en question doivent être gérées par le système GAB (nécessite un module de gestion ou non etc.) 2. Comment est-ce-que la gestion des diverses copies de données se fait en conjonction avec l’introduction des processus concurrentiels 3. Comment gérer les copies multiples en conjonction avec les voies de communication (TCP/IP) potentiellement interrompues ou intermittentes 4. Est-ce-que l’utilisation de copies multiples est faisable ou atteignable avec les exigences fonctionnelles et matérielles du système GAB (en termes de sécurité etc.) 5. Comment concevoir les modules du système GAB qui vont déclencher l’élaboration des copies multiples (les déclenchements surviennent-ils des actions des utilisateurs ou modules internes au système GAB) |

**Tactique: Maintain Multiple Copies of Computations (Manage Resources)**

**Description** : Le but de cette tactique est de répliquer les données qui autrement seraient situées sur un serveur central [BAS2011] [BAS2013]. Une des raisons pour laquelle cette tactique peut être appliquée est basée sur l'hypothèse que le système de GAB peut être considéré comme un système client - serveur, où le client est l'unité de GAB (matériel physique et système informatique) et le serveur, qui est l’unité centrale bancaire. Cette tactique est très semblable au mécanisme de mise en cache des données qui est couramment utilisé dans des applications web par exemple.

**Justification** : L’objectif principal des calculs répliqués est de réduire les conflits qui pourrait potentiellement se produire si tous les calculs (‘computations’) avaient lieu sur un serveur central [BAS2011] [BAS2013]. Pour illustrer ceci, un exemple est fourni. L’utilisateur se présente au guichet bancaire et s’identifie avec succès. Par la suite, il précise un compte bancaire et il initie une série de dépôts. Un processus pourrait faire les calculs de dépôts (au niveau de la machine bancaire) tandis qu’un autre processus pourrait s’exécuter dans l’arrière-plan qui valide les nouvelles données avec celles qui ont été récupérées auparavant du serveur central.

### Diagramme d’allocation (style architectural de déploiement)





### 

### Description du diagramme

Il est primordial de noter que ce document poursuit l’analyse et l’étude fait du système GAB lors de la première itération qui s’est concentrée sur les attributs de qualité de disponibilité et modificabilité. Alors, il ne doit pas très surprenant de voir des modules ou composantes (physiques ou logiciels) qui reviennent dans cette analyse. Par conséquent, il y aura un chevauchement avec les diagrammes architecturaux concernant la disponibilité et la modificabilité. De plus, ce chevauchement sera aussi présent dans la section suivante qui décrit les éléments du diagramme.

Comme c’était le cas de l’attribut de qualité de disponibilité, l’attribut de performance est bien illustré et bien encadré par la vue architecturale d’allocation et plus précisément par le style architectural de déploiement [BAS2011] [BAS2013]. En effet, cette vue et ce type architectural permettent de mieux illustrer et documenter comment l’allocation des éléments logiciels (nœud de traitements de données, processus de traitement des voies de communication etc.) doivent être répartis sur les composantes physiques [BAS2011] [BAS2013]. De plus, cette vue et ce style permettent également de mieux comprendre les limitations des tactiques envisagées tout en tenant compte des propriétés (métriques / mesure de réponse) qui permettent de décrire et quantifier la performance du système.

Parmi les caractéristiques du système qui sont bien illustrées par cette vue architecturale et qui ont un impact indéniable sur le système en termes de performance, on distingue les suivantes [BAS2011] [BAS2013]:

* Propriétés du CPU
* Propriétés de mémoire (capacité maximale, capacité d’expansion,
* Bande passante (nœuds de communication etc.)

Dans le diagramme présenté ci-dessus, un effort additionnel a été mis afin d’illustrer l’addition des nouveaux modules tenant compte du nouvel attribut de qualité en étude, soit la performance. En effet, les modules verts sont les modules qui ont nouvellement ajoutés au diagramme d’allocation (style déploiement). De plus, les relations en vert sont des nouvelles relations ou bien des relations qui ont été modifiées depuis la version précédente (partie disponibilité). En fait, les nouvelles additions font ressortir l’application des deux tactiques choisies, soit l’introduction de concurrence et la réplication des données calculées.

En bref, ce deuxième diagramme de déploiement est un amalgame des modules issues de l’analyse des attributs de qualité disponibilité et performance. Encore une fois, cette vue permet d’illustrer la nature symbiotique d’un système informatique et le matériel sous-jacent. En effet, cette vue ou perspective permet à l’architecte de voir comment organiser les diverses composantes informatiques et comment les déployer sur les diverses composantes physiques. De plus, ce diagramme permet également d’observer la manière dont la complexité d’un système augmente en appliquant soigneusement la méthode ADD.

### Description des éléments du diagramme

La table suivante décrit sommairement les éléments du diagramme présenté ci-dessus et leur relation avec les cas d’utilisation et les contraintes fournies dans le document de spécifications. Les sections précédées d’un astérisque (\*) sont les nouveaux modules issus de l’analyse de la performance. Le restant des modules a été élaborées lors de la première itération d’analyse se concentrant sur la disponibilité.

|  |  |  |
| --- | --- | --- |
| ÉLÉMENTS | DESCRIPTION | COUVERTURE DES CAS D’UTILISATION / CONTRAINTES |
| M1 – Hardware Components  Représente l’ensemble des composantes physiques utilisées par les usagers | | CT2 – CT4 |
| M1.1 – Card Reader | Représente le lecteur magnétique physique de carte-client |  |
| M1.2 – Keypad | Représente le clavier physique avec lequel l’utilisateur intéragit avec le système GAB |  |
| M1.3 – Money Dispenser | Représente le distributeur pour fournir de l'argent (par multiples de 20 $) |  |
| M1.4 – Printer | Représente l’imprimante physique du système GAB qui fournit les recues aux utilisateurs |  |
| M1.5 – Display | Représente l’écran qui affiche les menus etc. aux utilisateurs |  |
| M1.6 – Envelope Feeder | Représente une fente pour recevoir les enveloppes de dépôt |  |
| M1.7 – Machine State Switch | Représente un interrupteur à clé pour permettre à un opérateur de démarrer ou d'arrêter la machine. |  |
| M2 – Application GAB  Ensemble de composantes logicielles de haut niveau représentant les diverses exigences fonctionnelles et non fonctionnelles du syst;eme GAB (basé sur le document de specification et du qualité d’attribut de disponobilité) | | CT1 – CT2 – CT3 – CT4 |
| M2.1 – Startup Service | Composante qui s’assure que l’application GAB se démarre dans un bon état et s’assure de la fermeture du système (shut down). Cette dernière action est effectuée en conjunction avec M2.12. | CU1 – CU2 |
| M2.2 – Money Monitor | Composante qui surveille la quantité d’argent physiquement disponible dans l’appareil GAB (en billets de 20$). | CU4 – CU5 – CU6 – CU7 |
| M2.3 – Application Controller | Composante qui le point focale de l’application GAB. Elle permet d’intercepter les actions de l’utilisateur et les acheminer vers les autres composantes. Elle représente un sorte de façade de haut niveau [GAM1995]. | CU3 – CU4 – CU5 – CU6 – CU7 – CU8 – EXT1 |
| M2.4 – Hardware Abstraction Service | Composante qui représente les composantes physiques du système GAB du coté applicative. Cette composante traduit l’interaction de l’utilisateur du coté physique et les transforme en événements, qui sont interprétés par la suite par M2.3. | CU3 – CU4 – CU5 – CU6 – CU7 – CU8 |
| M2.5 – Account Manager | Composante qui permet le gestion des comptes bancaires d’un utilisateur. Elle permet de récupérer les informations nécessaires pour qu’un utilisateur puisse effectuer une transaction dans un ou plusieurs comptes bancaires. | CU5 – CU6 – CU7 – CU8 |
| M2.6 – Data Access Point | Composante qui représente d’un niveau d’abstraction plus élevé la base de données centrale. En effet, cette composante fournit les fonctionalités nécessaires pour accéder aux informations emmagasinées dans la base de données. | CU3 – CU4 – CU5 – CU6 – CU7 – CU8 |
| M2.7 – Login Service | Composante qui gère le processus d’authetification de l’utilisateur dans le système. En effet, la combinaison de la carte (puce etc.) et le NIP fourni permet l’authetification. De plus, cette composante gère aussi le cas où le NIP fournit est invalide ainsi que le nombre de tentatives possibles. | CU3 - EXT1 |
| M2.8 – Fault Monitoring Service | Composante qui utilise le mécanisme PING/ECHO afin de s’assurer que la voie de communication avec la base de données centrale est toujours en bon état et disponible. |  |
| M2.9 – Transaction Manager | Composante qui permet la gestion des transactions des utilisateurs. Parmi les différents types de transactions, on retrouve les retraits, les depots, les transferts. | CU3 – CU4 – CU5 – CU6 – CU7 |
| M2.10 – Checkpoint Manager | Composante qui permet de gérer les diverses checkpoints ou points critiques des diverses transactions. Cette composante permet entre autres d’exécuter des undo au niveau des transactions en cas de problème. | CU3 – CU4 – CU5 – CU6 – CU7 |
| M2.11 – Checkpoint Data Store | Composante qui représente une base de données interne qui permet d’emmagasiner les divers checkpoints pout une utilisation ultérieure et pour des fins de logs. | CU3 – CU4 – CU5 – CU6 – CU7 |
| M2.12 – Application State Manager | Composante qui surveille l’état de l’application GAB (in use – idle). Cette composante avec M2.1 permet également de verifier que le shut down puisse s’effectuer gracieusement. | CU2 |
| M2.13 – Network Manager | Composante qui permet la gestion des différents réseaux qui peuvent se joinder au système GAB. En effet, cette composante représente le point d’interaction entre les réseaux externs et l’application GAB. | CT4 |
| \*M2.14 – Process Manager | Composante qui permet la gestion des divers procès semi-indépendants. Cette composante possède la responsabilité de gérer le cycle de vie des threads nécessaires. Cela inclu le démarrage de ces derniers, les divers temps de délais au niveau d’exécution, et la fin de ces threads. Parmi les éléments de gestion, cette composante s’occupe également de s’assurer que les threads ne consomment pas des resources aléatoirement et qu’il n’y a pas de fuites de mémoire. |  |
| \*M2.15 – Thread Factory | Cette composante est sémantiquement similaire au patron de conception Fabrique présenté par les auteurs GOF [GAM1994]. Elle s’occupe de créer les divers types de threads en tenant compte de la stabilité du système (materiel déféctueux, communication interrompu etc.) |  |
| \*M2.16 – Finance Threads | Cette composante représente le type de thread ou procès qui s’occupe des calculs financiers. Par exemple, ces threads pourront faire les calculs de conversion de devise en accédant à des systems externes ou bien faire les calculs d’intérets etc. |  |
| \*M2.17 – Account Threads | Cette composante représente le type de thread qui vont faire les calculs sur les comptes bancaires en tant que tells. |  |
| \*M2.18 – Native Threads | Cette composante représente les types de threads qui vont faire des calculs (validation etc.) sur les systèmes sous-jacents physiques. Des calculs comme ‘checksum’ ou bien la validation d’une carte magnétique (si le système peut la lire ou non etc.). |  |
| \*M2.19 – Computation Copy Manager | Cette composante s’occupe de la gestion des calculs répliqués. En effet, elle s’occupe de la persistence temporaire des calculs et intéragit avec les modules de transaction afin de s’assurer que le débobinnement d’un ensemble d’actions puisse se faire. |  |
| \*M2.20 – Computation Store | Cette composante permet de persister localement les données répliquées de calculs. Elle assume le role du serveur central afin de minimiser ou plutot diminuer le temps d’exécution necessaire. |  |
| M3 – Banking Server (External)  Composante qui représente la base de données centrale de la banque. | | CT3 |
| M4 – Interac Network (External)  Composante qui représente le réseau externe Interac. | | CT4 |

### Relation entre les éléments et les tactiques

La première tactique qui vise l'introduction de la concurrence a pour objectif principal de réduire le temps bloqué d'une demande spécifique. L'utilisation de plusieurs threads pour le système permet de traiter simultanément différents processus et les résultats intermédiaires sont ensuite reliés entre eux.

Dans le cas présent, l'introduction de la concurrence se fait par l'intermédiaire de plusieurs composantes. Les composantes impliquées sont :

* Process Manager
* Thread Factory
* Native Threads
* Account Threads

Alors ces modules permettent de créer des procès de manière indépendante l'un de l'autre (lorsqu'il est nécessaire) qui suivent le paradigme 'fitness for purpose' et donc s'occupent à compléter une tache en particulier sans à avoir à attendre la conclusion d'un procès [BAS2011] [BAS2013]. En effet, en absence de cette habilité concurrentielle, les instructions doivent s'exécuter séquentiellement [BAS2011] [BAS2013].

En ce qui concerne la deuxième tactique, soit garder multiples copies répliquant des calculs (‘computations’), son objectif principal est de répliquer les données (‘computations’) qui autrement seraient situées sur un serveur central [BAS2011] [BAS2013]. En admettant l’hypothèse que le système GAB suit le modèle client-serveur, il est dorénavant nécessaire d’avoir localement une copie des données calculées [BAS2011] [BAS2013]. Dans le diagramme ci-dessus, cette tactique est illustrée par l’addition des modules suivants :

* Computation Copy Manager
* Computation Store

De plus, la composante Computation Copy Manager interagit avec les module Process Manager directement. De cette manière, le Process Manager signale la nécessité des données emmagasinées dans le Computation Store ou bien d’apporter des modifications au Computation Store. Enfin, les calculs persistés dans le Computation Store seront ultérieurement envoyés au serveur central afin de finalisée la transaction via la composante Data Access.

### Comment la tactique répond à chacun des scénarios de qualité

Les deux scénarios envisagés initialement sont présentés sommairement ci-dessous :

* Scénario 1 : Latence d’une opération ou transaction financière
* Scénario 2 : Maintenir plusieurs copies des données calculées (‘computational data’) : Exécution de threads indépendants

Or, il est d’une importance primordiale de mentionner que les deux tactiques proposées dans ce document concernant a performance, soit l’introduction de concurrence et le maintien de multiples copies des données calculées, possèdent un rôle symbiotique l’une vers l’autre [BAS2011] [BAS2013]. Cela signifie que les tactiques proposées ont un effet positif l’une sur l’autre. En effet, les deux tactiques permettent de diminuer le temps d’exécution soit d’un cycle ou trajet aller-retour des données (action initiée par l’utilisateur – serveur central – affichage à l’utilisateur), soit d’un conflit potentiel si tous les calculs (‘computations’) avaient lieu sur un serveur central [BAS2011] [BAS2013].

La première tactique, l’introduction de concurrence, en ce qui concerne le système GAB, de tels procédés semi-indépendants comme la lecture de la carte magnétique, validation de l'information à la clientèle et de vérifier la présence du réseau Interac. En outre, l'utilisation de la concurrence et les threads permet également au système GAB de coordonner divers procédés. Plutôt qu'avoir le système exécuter toutes les instructions de façon séquentielle, cette tactique permet de raccourcir le temps total d'exécution (scénario 1). De plus, cette première tactique permet d’organiser l’exécution, l’accès aux données répliquées de manière optimale et non séquentielle et donc le temps total d’exécution est de nouveau diminué.

La deuxième tactique vise à maintenir des copies de données autant sur le serveur que sur le client. Dans un premier temps, cette tactique permet d’éviter les aller-retours non nécessaires au serveur central. Donc les latences du réseau sont minimisées et donc le temps d’exécution d’une transaction est forcément diminué. Ceci répond positivement le premier scénario. Dans un deuxième temps, cette tactique a pour objectif principal de réduire les conflits qui pourrait potentiellement se produire si tous les calculs avaient lieu uniquement sur le serveur central. Par exemple, un processus pourrait faire les calculs de dépôts (au niveau de la machine bancaire) tandis qu’un autre processus pourrait s’exécuter dans l’arrière-plan qui valide les nouvelles données avec celles qui ont été récupérées auparavant du serveur central.

**Conclusion**

En conclusion, cette deuxième itération dans l’analyse du système GAB a mis l’emphase sur les attributs de qualité : performance et sécurité. Ce document se concentre sur la performance. L’application de la méthodologie ADD a permis d’observer la difficulté d’introduire des attributs de qualité dans l’analyse d’un système. En effet, chaque nouvelle itération de l’analyse a le potentiel d’introduire des tactiques proposées avec un impact négatif sur d’autres tactiques. Dans le cas présent, cela ne fut pas observé. Au contraire, les tactiques proposées ont eu un effet mutuellement positif en ce qui concerne le temps d’exécution. Enfin, ce deuxième volet d’analyse a permis d’apprécier les difficultés que doit affronter un architecte logiciel au niveau du choix de tactiques et le compromis qui doivent être faits lorsque les tactiques proposées entrent en conflits l’une par rapport à l’autre.

**Références**

**BAS2011**

Bass, Len, Clements, Paul, Bachman, Felix et Al. 2011. Documenting Software Architecture, 2nd, Upper Saddle River, New Jersey: Pearson-Addison-Wesley, 537.

**BAS2013**

L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, 2013, Upper Saddle River, New Jersey: Pearson-Addison-Wesley, 537.

**CHA2013**

R. Champagne, Conception de l’architecture logicielle Attribute-Driven Design (ADD), v1.0, été 2013

**ETS720**

École de technologie supérieure, Département de génie logiciel.2014.<<Notes de cours>>. En ligne. < https://cours.etsmtl.ca/log720/>.

**GAL2004**

Galin, Daniel. 2004. Software Quality Assurance, 1st, Upper Saddle River, New Jersey: Pearson-Addison-Wesley, 590.

**GAM1995**

Gamma, Erich, Helm, Richard, Johnson, Ralph, Vlissides, John. 1995. Design Patterns (Elements of Reusable Object Oriented Software), Addison-Wesley, 394.

**SEI2013**

Site web du SEI, <http://www.sei.cmu.edu/architecture/tools/define/add.cfm> , Consulté le 27-07-2013

|  |  |
| --- | --- |
| **Scénario** | Délai de transaction « requête de solde » long |
| **objectifs d'affaires** | Réduire le délai de la transaction « requête de solde » |
| **Source** | Client |
| **Stimulus** | Requête de solde |
| **Artéfact** | Le logiciel du serveur de la centrale de la banque |
| **Environnement** | Mode normal |
| **Réponse** | Requête traitée, solde affiché et reçu émis |
| **Mesure de la réponse** | * Temps de réponse : Le délai au bout duquel le système affiche à l’utilisateur le solde de son compte et lui fournir le reçu. * Échéance : Le délai que le système ne doit pas dépasser pour répondre au client tel que Temps de réponse échéance |
| **Questions** | 1. Est-ce que le délai est dû au manque de ressources au niveau du serveur de la banque? 2. Quel est le temps de réponse toléré? 3. Comment améliorer l’efficacité du serveur de la banque pour gérer plusieurs requêtes simultanément? |

**Tactique :** Increase resources, introduce concurrency, maintain multiple copies of computations

**Description**

L’utilisateur initie une requête de solde, le système prend du temps pour répondre car le système s’est bloqué pour traiter l’arrivée de plusieurs requêtes simultanément.

Nous utilisons la tactique « **Increase resources** » pour augmenter les ressources du serveur en ajoutant des processeurs plus rapides, une mémoire additionnelle, etc.

Nous utilisons aussi la tactique « **Introduce concurrency** » afin de traiter les requêtes en parallèle en introduisant la notion du multithreading.

Afin de maximiser le nombre de requêtes gérées par le serveur, on utilise la tactique « **Maintain multiple copies of computations** » en introduisant plusieurs serveurs et un load balancer.

**Justification**

* La tactique « **Increase resources** » est considérée car elle permet de limiter le problème de manque de ressources qui a causé le blocage du serveur suite à une surcharge.
* La tactique « **Introduce concurrency** » est utilisée car elle permet de réduire le temps de blocage en traitant les requêtes en parallèle.
* La tactique « **Maintain multiple copies of computations** » est considérée car elle permet d’éviter la concentration de toutes les requêtes sur un seul serveur en les répartissant sur plusieurs. Cela évite ainsi le blocage du serveur.

|  |  |
| --- | --- |
| **Scénario** | Délai de démarrage du GAB trop long |
| **objectifs d'affaires** | Raccourcir le délai de démarrage du GAB |
| **Source** | Opérateur |
| **Stimulus** | Démarrage du GAB par positionnement de l’interrupteur sur « marche » |
| **Artéfact** | GAB |
| **Environnement** | Mode normal |
| **Réponse** | GAB démarré et disponible pour servir des clients |
| **Mesure de la réponse** | Latence : temps de démarrage du GAB + temps de connexion à la centrale de la banque + temps mis par l’opérateur pour renseigner le montant d’argent actuellement disponible |
| **Questions** | 1. Les 3 étapes de démarrage du GAB peuvent-elles être effectuées en concurrence ? 2. La latence est-elle due à un « processing time » et / ou à un « blocked time » ? 3. Une optimisation est-elle possible au niveau du délai de démarrage du GAB ? 4. Est-il possible de réduire le délai de connexion à la centrale (demand side) ? 5. Une optimisation est-elle possible au niveau de la réponse de la centrale (response side) ? |

**Tactique:** Increase resources, Increase resource efficiency

**Description**:

1. Nous utilisons la tactique « **Increase resources** » en intervenant sur les différents composants du GAB :

* processeur plus performant et multi-cœur
* ajout de mémoire
* interface réseau plus performante

1. Nous utilisons la tactique « **Increase resource efficiency** » en optimisant les algorithmes liés au démarrage du GAB.

**Justification**:

Les 3 étapes de démarrage du GAB ne peuvent pas être effectuées en concurrence : il faut que le GAB démarre, puis qu’il se connecte à la centrale de la banque, puis que l’opérateur renseigne le montant d’argent disponible dans le GAB. Par conséquent, aucune des tactiques liées à la concurrence ne peut être appliquée.

Parmi les tactiques de « **Manage Resources** », nous utilisons la tactique « **Increase resources** » afin de réduire la latence du GAB dans les 2 étapes du démarrage qui sont liées au hardware : le démarrage du GAB et le temps de connexion à la centrale. Le remplacement du processeur et l’ajout de mémoire permettent d’accélérer le démarrage du GAB et le remplacement de l’interface réseau permet d’accélérer la connexion à la centrale.

Parmi les tactiques de « **Control Resource Demand** », nous utilisons la tactique « **Increase resource efficiency** » afin d’optimiser le démarrage du système.

Les autres tactiques ne sont pas adaptées à ce scénario.

|  |  |
| --- | --- |
| **Scénario** | L’utilisateur a pour but que les caractères qu’il entre au clavier au moment de l’authentification soient affichés à l’écran en moins de 0.1 secondes. |
| **objectifs d'affaires** | L’affichage est suffisamment rapide pour que l’utilisateur ne se trompe pas en tapant son code au clavier. |
| **Source** | L’utilisateur |
| **Stimulus** | L’utilisateur appuie sur une touche du clavier. |
| **Artéfact** | Module d’authentification |
| **Environnement** | Opération normale du GAB |
| **Réponse** | Affichage immédiate des caractères lorsque le client touche le clavier. |
| **Mesure de la réponse** | Latence maximale de l’affichage de caractères à l’écran. |
| **Questions** | 1. Quelles sont les autres fonctionnalités concurrentes du module ? 2. Quel pourcentage de clients est ennuyé par ce problème ? 3. À quel niveau de “coupling” ou “cohésion” a-t-on affaire ? 4. Est-ce simplement un problème de hardware ? |

**Tactique: Prioritize events**

**Description** : Le système priorise le traitement de l’entrée au clavier et celui de l’affichage des caractères par rapport aux autres événements (contact au serveur, etc).

**Justification** : S’il n’y a pas de concurrence l’événement de l’entrée au clavier et l’affichage doivent être traités en priorité.

**Tactique: Increase resources**

**Description** : On améliore le hardware pour que l’entrée et la sortie soient traitées plus rapidement.

**Justification** : Si l’efficacité du traitement de l’entrée au clavier et de l’affichage à l’écran est due plus au hardware qu’à l’architecture et si le prix de l’amélioration du hardware est plus bas que celui de l’amélioration du logiciel, il convient d’améliorer le hardware.

**Tactique: Introduce concurrency**

**Description** : Le système utilise la concurrence pour pouvoir traiter plusieurs événements en parallèle.

**Justification** : En utilisant la concurrence on n’a pas à prioriser les événements puisque qu’on peut traiter l’entrée au clavier et l’affichage en parallèle des autres traitements d’événements.

|  |  |
| --- | --- |
| **Scénario** | Lorsqu’un client veut consulter le solde de son compte bancaire, il constate un délai pendant la transaction. |
| **objectifs d'affaires** | Réduire la latence lorsque le client demande une requête de solde |
| **Source** | Le client |
| **Stimulus** | Le client déclenche le processus, lorsqu'il choisit la compte à consulter. |
| **Artéfact** | Module de transactions du logiciel |
| **Environnement** | Situation normale du système |
| **Réponse** | Afficher le plus tôt possible le solde d’une compte bancaire. |
| **Mesure de la réponse** | Au maximum un délais de 5 secondes pour afficher le solde. |
| **Questions** | 1. Combien de transactions passent en moyenne vers un même serveur ? 2. Est-il important que toutes les transactions soient traitées jusque complétion, et peut-il y avoir des transactions qui ne peuvent être traitées par le serveur ? |

**Tactique: Bound execution time + Increase resource efficiency**

**Description** : Le serveur reçoit plusieurs demandes au même temps, alors une queue des transactions reste en attente jusqu’à ce que le serveur puisse terminer toutes ces demandes. Cela force le serveur à travailler de manière plus lente. Par ailleurs, le cryptage et décryptage de l’information peut rendre aussi le processus long.

**Justification** : Fixer le temps maximum de réponse lors d’une demande de solde permet au serveur de continuer de façon immédiate avec la prochaine demande. De plus, une amélioration dans l’algorithme de cryptage permettra aussi de réduire le temps de transformation des données.

|  |  |
| --- | --- |
| **Scénario** | Minimiser le temps requis pour effectuer un retrait |
| **objectifs d'affaires** | Le client souhaite qu’un utilisateur puisse effectuer un retrait en moins d’une minute dans le but de se démarquer de la compétition et augmenter la clientèle. |
| **Source** | Externe (action de l’utilisateur) |
| **Stimulus** | Événement stochastique (l’utilisateur veut effectuer un retrait) |
| **Artéfact** | L’application du guichet |
| **Environnement** | En période de pointe |
| **Réponse** | L’utilisateur peut effectuer un retrait avec succès dans les temps requis |
| **Mesure de la réponse** | * Latence (temps pour une communication avec le serveur). Moins de 3 secondes. * Temps total pour toutes les étapes. Moins d’une minute. * Nombre de communications avec le serveur requises pour compléter la transaction : Maximum de 6. |
| **Questions** | 1. Quelles sont les règles d’affaires supplémentaires qui pourraient ralentir le processus (ex : date de naissance si montant plus grand que x) ? 2. Quelle est la valeur de « timeout » pour chaque étape ? 3. Que fait-on sur un timeout ? (Annuler ou redemander une confirmation) ? 4. Est-ce que la vérification du solde disponible est effectuée en temps réel à chaque transaction ? |

**Tactique: Bound times execution**

**Description**: Déterminer une durée normale maximale pour une étape du processus.

**Justification**: On connait les délais normaux d’opérations donc, en période de pointe, si on dépasse de 100% le délai normal d’exécution, on va lever une erreur et le système devra prendre une décision (annuler ou recommencer). Le but étant d’accélérer le traitement et de ne pas bloquer inutilement le système si une seule transaction bloque. On veut pouvoir continuer de servir d’autres clients rapidement. Dépendamment de la décision des parties prenantes, on pourrait accepter une perte monétaire si le système est lent et que le montant de la transaction est petit (ex : moins de 60$).

|  |  |
| --- | --- |
| **Scénario** | Minimiser l’impact de l’encryptions sur la performance |
| **objectifs d'affaires** | Pour satisfaire les officiers de la sécurité, toutes les communications et les données sur disques doivent être encryptées (sécurité) en ayant un impact minimale (moins de 1 seconde par transaction) sur la performance du système. |
| **Source** | L’utilisateur au guichet |
| **Stimulus** | Événement stochastique (l’utilisateur déclenche une communication avec le serveur) |
| **Artéfact** | Algorithme d’encryptions |
| **Environnement** | En opération |
| **Réponse** | Fonctionnement normal de l’application (ne devrait pas être visible par l’utilisateur) |
| **Mesure de la réponse** | Temps d’encryptions par transaction  Nombre d’encryptions possible par secondes  Ratio des encryptions qui dépassent le temps prévu (1 seconde) |
| **Questions** | 1. Force minimale de l’encryptions ? 2. Est-ce qu’un algorithme précis est requis (légal) ? 3. Si c’est le cas, quelle est la marge de manœuvre ? 4. Que faire si on dépasse le temps requis (annuler la transaction, poursuivre la transaction avec un message à l’utilisateur, autre) ? |

**Tactique: Reduce overhead ou increase resource**

**Description**: Utiliser des algorithmes ou une architecture qui minimise le nombre d’instructions à exécuter (au risque de compliquer la maintenance) ou ajouter des ressources matérielles.

**Justification**: Si un algorithme précis est requis, 2 choix s’offre à nous : réduire la latence dans le code en l’optimisant au maximum et, si ce n’est pas suffisant pour respecter les requis, augmenter les capacités matériels des guichets. Si on a le choix de l’algorithme, prendre le plus performant qui respecte les requis de sécurité.

|  |  |
| --- | --- |
| **Scénario** | Délai de transaction (communication entre serveur de la banque et le guichet automatique) pour une transaction effectuée au guichet |
| **objectifs d'affaires** | S’assurer qu’une transaction entre le guichet automatique et la banque se fasse avec un délai maximum de 15 secondes. Ceci nous permet d’éviter le délai d’attente pour le client ainsi que d’éviter le bottleneck à la banque (Il peut y avoir plusieurs personnes qui font des retraits en même temps). |
| **Source** | Utilisateurs, détenteurs de comptes |
| **Stimulus** | L’arrivée de façon stochastique de la transaction - requête (dépôt, retrait, transfert...) |
| **Artéfact** | Le serveur (système central de la banque) et la composante de communication (composante et réseau) |
| **Environnement** | Mode opérationnel normal ou en surcharge (plusieurs transactions effectuées en même temps) |
| **Réponse** | Les transactions sont processés et une réponse est donné au client  Enregistrer dans un journal les transactions avec le temps début et fin de la transaction.  Afficher un message de mise en attente sur l'écran du guichet  Si délai avertir le client |
| **Mesure de la réponse** | Le temps de réponse pour approuver la transaction (le temps de latence de la réponse du serveur) doit être inférieur à 15000 millisecondes)  Vérifier le temps écoulé entre le début et la fin de la transaction. |
| **Questions** | 1. Est-ce que le changement va affecter les autres attributs qualité? 2. Le coût et le temps pour ce changement. 3. Combien de transactions simultanées peuvent avoir lieu?? 4. Quel serait un délai acceptable pour effectuer une transaction selon le client? 5. Est-ce qu’on permet au client de changer de type de transaction si celle qu’il essayait d’effectuer n’a pas marcher? |

**Tactique: Optimisation des calculs, concurrence**

**Description**: Limiter le nombre de composantes et réduire la fréquence de communication entre les composantes. Augmenter l’efficacité des algorithmes critiques, pour générer une réponse le plus rapidement possible. Traiter des transactions en concurrence (en même temps) tant que les comptes touchés par les transactions soient différentes. Dans les cas où plusieurs transactions ciblent le même compte, une priorité FIFO (first in / first out) serait implémenté.

**Justification**:

Lors d’une transaction, pour que la réponse retourne rapidement à l’utilisateur, il est important de minimiser le couplage et d’augmenter la cohésion entre les composantes. , Ceci peut être considéré comme un compromis par rapport à la modifiabilité du système (à cause d’un module avec beaucoup de responsabilités) mais nous permet de gagner de la rapidité pour traiter la requête. L'implémentation d’un système qui peut lancer des fils d'exécution différents pour chaque transaction permet que plusieurs transactions soient lancées en même temps.

|  |  |
| --- | --- |
| Scénario | Le temps d'attente entre le dépôt d’espèces et la disponibilité des fonds |
| objectifs d'affaires | Réduire le délai d’attente pour que le guichet reçoit et reconnaît exactement la somme déposée en espèces par le client à travers du guichet électronique. Faire en sorte que l’argent soit effectivement ajouté à son compte pour qu’il puisse utiliser la somme déposé .ex: faire des achats avec sa carte debit, faire un virement, payer sa facture, etc. Le temps pour traiter un dépôt et fournir au client un reçu ne doit pas surpasser 10 secondes (suggéré) |
| Source | Utilisateurs, détenteurs de comptes |
| Stimulus | L’utilisateur fait un dépôt en espèces dans le guichet électronique |
| Artéfact | Module de dépôt en espèces au guichet |
| Environnement | Mode opérationnel normal |
| Réponse | L’argent est bel et bien ajouté au compte de l’utilisateur  Afficher un message montrant le nouveau solde du client  Avertir le client du délai pour que les sommes soit disponible pour faire d’autres transactions. |
| Mesure de la réponse | Le temps pour faire le dépôt, le temps pour la mise à jour des données  Vérification du montant déposé dans le guichet (l’enveloppe a-t-elle réellement le montant déposé)  Débloque les fonds et envoyer un message au client pour avertir de la disponibilité des fonds |
| Questions | 1. Est-ce que le changement va affecter les autre attributs qualité? 2. Le coût et le temps pour ce changement. 3. Faut-il augmenter de personnelle pour valider le montant déposé et rendre disponible les fonds? 4. Quel est le temps d’attente maximum raisonnable. |

**Tactique:** Optimisation des calculs (augmentation de l’efficacité des ressources)

**Description:** Un algorithme spécialisé qui a pour but de détecter la somme d’argent déposé par le client sera mis en place dans le logiciel installé au guichet. Une fois la somme exacte soit détecté, le guichet devra initier une transaction avec le serveur pour ajouter telle somme au compte de l’utilisateur. Une fois le serveur confirme que les fonds ont bel et bien été ajoutés, le guichet donnera l’option à l’usager d’imprimer un reçu comme confirmation de la transaction..

**Justification:** Il faut que l’algorithme de détection de l’argent d’un dépôt soit assez rapide pour que la transaction puisse se compléter au plus vite et donc pour que les usagers soient intéressés par telle fonctionnalité, sinon, ils seront moins motivés de faire des dépôts dans leur compte.

|  |  |
| --- | --- |
| **Scénario** | Diminuer le temps de réponse du GAB à une demande de retrait |
| **objectifs d'affaires** | Une transaction de retrait doit être traitée (acceptée ou refusée) dans un plus bref délai de 3 secondes |
| **Source** | Le GAB |
| **Stimulus** | Système du GAB reçoit une requête de retrait de la part du client  Évènement sporadique : Temps écoulé entre l’évènement déclencheur du retrait et celui confirmant ce dernier. |
| **Artéfact** | Système du GAB, (module qui s’occupe des retraits) |
| **Environnement** | RunTime |
| **Réponse** | Message affiché à l’utilisateur : « Traitement de votre demande en cours…». |
| **Mesure de la réponse** | Temps de réponse (latence) du serveur bancaire à la demande de retrait. |
| **Questions** | 1. Comment est constituée l’architecture de la banque ? 2. Quel serait le coût de l’augmentation des ressources ? 3. Quel serait le temps de la mise en place de la concurrence et des ressources supplémentaires ? |

**Tactique: Prioriser les évènements + Introduire la concurrence + Augmenter les ressources**

**Description**: Nous décidons d’utiliser la priorisation d’évènement pour donner la priorité à des demandes de retrait vis-à-vis des autres transactions.

Nous décidons également de traiter les transactions de façon parallèle, en introduisant plus de concurrence (plus de threads). Si l’augmentation de la concurrence ne suffit pas, nous ajouterions alors des ressources supplémentaires au système.

**Justification**: Nous considérons que les clients effectuent en moyenne plus d’opérations de retraits que d’autres types de transactions. Dans ce sens, nous accordons la priorité aux retraits dans l’objectif de d’obtenir une réponse plus rapidement du serveur bancaire.

L’introduction de la concurrence et l’augmentation des ressources permettra également de diminuer ce temps de réponse.

|  |  |
| --- | --- |
| **Scénario** | Optimiser le nombre de transaction maximum traité par le serveur bancaire |
| **objectifs d'affaires** | Le système bancaire doit pouvoir traiter N demandes de transaction de GAB par minute lors des peak load pour éviter la surcharge des serveurs. |
| **Source** | Le serveur bancaire |
| **Stimulus** | Le serveur bancaire reçoit un nombre très important de requête de la part des GAB pendant les heures de pointes. Ceci peut donc, être catégorisé d’événement périodique, car se produit régulièrement tous les jours ouvrables aux heures de pointes. |
| **Artéfact** | Système bancaire, gestion des transactions |
| **Environnement** | Surcharge (Overload) |
| **Réponse** | Le serveur ne peut pas gérer toutes les demandes. Le temps de latence augmente. |
| **Mesure de la réponse** | Nombre de transactions traitées par minute + Temps moyen de réponse du serveur bancaire à une requête. |
| **Questions** | 1. Qu’elle est la charge maximale actuellement supportée par le système 2. Y- a-t-il d'autres opérations effectuées à part les requêtes des GAB considérées dégradeurs de performance aux (ex. les opérations de Backup,..)? 3. La banque utilise-t-elle un système de connexion internet rédondé ? |

**Tactique: Prioriser des évènements + Augmenter la concurrence + Augmenter les ressources + Programmer les ressources**

**Description**: Nous décidons de prioriser les évènements liés à des transactions vis-à-vis des autres tâches en cours, en suivant des critères d’importances et suivant les périodes de la journée (surcharge ou non). Afin de garantir un maximum de ressources disponibles pour traiter les transactions lors des piques de charge, nous décidons de reporter des actions moins prioritaires ou d’une durée importante lors des périodes creuses de la journée (Schedule ressources).

Il faut également augmenter la concurrence (traitement des ordres de transactions de façon parallèle). Si cela s’avère nécessaire nous augmenterons les ressources matérielles

**Justification**: Lors des surcharges, prioriser les tâches de transaction, augmenter la concurrence et repousser les tâches longues et moins prioritaires nous permet de mettre à disposition un maximum de ressources pour traites les demandes des GAB.

|  |  |
| --- | --- |
| **Scénario** | Transfert d’argent entre deux comptes |
| **objectifs d'affaires** | Assurer un temps de réponse entre les deux comptes inférieur à deux secondes. |
| **Source** | Client |
| **Stimulus** | Demande le transfert d’argent de façon aléatoire. |
| **Artéfact** | Le processus de gestion de transactions bancaires. |
| **Environnement** | Opération normale. |
| **Réponse** | Le GAB va traiter la demande. |
| **Mesure de la réponse** | Latence de transfert d’argent. |
| **Questions** | 1. Combien ça coute pour diminuer le temps de réponse ? 2. Quelle est la procédure qui doit être faite si la latence n’est pas respectée? 3. Quel est le temps de réponse toléré? |

**Tactique:** Manage ressources - Maintain multiple copies of data / – data replication

**Description** : la réplication de données permet de garder des copies séparées de données dans le but de réduire la congestion du réseau au cas où de multiples clients accèdent simultanément à ces données.

**Justification** : Le recours à cette tactique est intéressant dans la mesure ou avoir plusieurs copies de l’information sur les comptes des utilisateurs permet d’améliorer la performance du système en diminuant le temps de réponse lors d’une transaction inter-comptes. Les bases de données peuvent être à différents emplacements géographiques pour diminuer la latence. Ainsi, le serveur recevant les requêtes va utiliser ces réplicas locaux afin de répondre le plus rapidement (temps inférieur à deux secondes) possible aux requêtes de transfert d’argent entre les comptes.

|  |  |
| --- | --- |
| **Scénario** | Retrait d’argent |
| **objectifs d'affaires** | Assurer un temps de réponse entre le GAB et le client inférieur à une seconde. |
| **Source** | Client |
| **Stimulus** | Demande le retrait d’argent de façon aléatoire |
| **Artéfact** | Le processus de gestion de transactions bancaires |
| **Environnement** | Opération normale |
| **Réponse** | Le GAB va traiter la demande. |
| **Mesure de la réponse** | Latence de retrait d’argent. |
| **Questions** | 1. Quels sont les impacts de l’amélioration de la performance sur les autres fonctionnalités du système ? 2. Quel sera le coût avoir une meilleure latence ? 3. Quel est le temps de réponse toléré? |

**Tactique:** Manage ressources - Maintain multiple copies of computation / – load balancer

**Description** : L’idée est de multiplier les réplicas de calcul dans le but de réduire la charge assignée à un seul serveur. Ainsi, l’équilibrage de charge permettra d’affecter une tache à l’un des serveurs dupliqués disponibles, de façon à ce que le serveur le moins saturé prend en charge la requête suivante.

**Justification :** Ce mécanisme de répartition du traitement sur différentes machines permet d’augmenter la capacité totale et par la suite de traiter le maximum de requêtes et répondre dans le délai spécifié (temps de réponse entre le GAB et le client inférieur à une seconde).