En mode sécurisé, le répartiteur créer un thread pour chaque serveur qu’il connait. Ainsi, il est la responsabilité de chaque thread de tenter de donnée des opérations à exécuter à chaque serveur. Les opérations à exécuter se trouvent dans une liste d’opération non complété qui est verrouiller par un thread chaque fois que celui-ci tente d’accéder à de nouvelles opérations afin d’éviter la duplication d’opérations. Pour connaitre le nombre d’opération que le serveur acceptera de faire lors de la prochaine requête, le répartiteur commence toujours par demander au serveur si celui-ci accepte une valeur cinq fois supérieure à son qi. Chaque fois que le serveur se fait refuser une demande, il diminue le facteur multiplicatif du qi de 1 jusqu’à l’acceptation du serveur. À la première demande, le taux de refus est donc de 80% ensuite de 60%, 40%, 20% et à la cinquième demande, le serveur accepte obligatoirement. Dans les faits, on observe rarement plus de 2 ou 3 refus par serveur et souvent une acceptation après seulement 2 demande. Pour ce qui est des pannes, le répartiteur reçoit ‘’catch’’ une exception lorsqu’il fait un appel de fonction à un serveur et que celui-ci plante lors de l’exécution. Lors d’un tel cas, le thread associer à ce serveur se termine après avoir remis les opérations qu’il avait réserver dans la liste opération à faire.

En mode non-sécurisé, le fonctionnement est très similaire. Le répartiteur possède deux listes. La première contient les opérations à exécuter et la deuxième contient tous les serveurs disponibles. Le répartiteur créer autant de thread qu’il y a de serveur disponible. Ensuite chaque thread tente d’acquérir deux servers contenus dans la liste de serveur disponible en verrouillant bien sûr cette liste à chaque accès. Ensuite, le répartiteur prend la plus petite valeur de qi des deux serveurs et fait le même processus de demande d’acceptation de calcul d’opération que lors du mode sécurisé pour les deux serveurs jusqu’attend que les deux serveurs s’entendent sur une quantité d’opération à effectuer. À ce moment, les deux serveurs exécutent les opérations sélectionner de la même façon que lors de la méthode sécurisée. Dans le cas ou les serveurs s’entendent sur la valeur du calcul, le répartiteur prend en compte le résultat, remet les deux serveurs dans la liste de serveur disponible et retente ça chance pour exécuter de nouvelles opérations. Cependant, dans le cas ou les serveurs ne sont pas d’accord, le serveur remet encore une fois les deux serveurs dans la liste de serveur disponible, mais il remet aussi les opérations dans la liste d’opération à faire et ne tiens pas compte des résultats donnés par les deux servers. Ces opérations seront exécutées tant et aussi longtemps que deux serveurs ne s’entendent pas sur le résultat. En cas de crash de l’un des deux serveurs, le répartiteur reconnait le serveur fautif, remet le bon serveur dans la liste de serveur disponible et remet aussi les opérations dans la liste d’opération à faire avec de simplement se terminer.

Nous avons implémenté le système en mode sécurisé avec un thread par serveur puisque c’était la méthode la plus simple à laquelle nous avons pus penser qui permettait de tirer avantage du multithreading et de l’synchronisation des serveurs. Dans le cas du mode non-sécurisé, nous avons autant de thread que de serveur pour la simple et unique raison que le système est capable de continuer de travailler peu importe le nombre de serveur qui tombe en panne. La plupart des autres décisions que nous avons prises ont aussi été faite de façon à se que l’implémentation soit rapide et tire avantage de l’synchronisation des serveurs au maximum.