

---

**SINF803 • Bases de données réparties (Groupe 1)**

Enseignant: Edmond La Chance  
Bureau: P2-4170  
Courriel: edmond.lachance@gmail.com  
Page web du cours: <https://github.com/edmondlachance/SINF803>

---

**Contenu général**

Certains sites et applications ont des millions d'utilisateurs. Ces derniers génèrent de l'information tous les jours. Toute cette information, ce data, doit être enregistré et peut être traité et/ou indexé afin de fournir un service aux utilisateurs. Ce domaine est également appelé le domaine du Big Data. Exemples de sites et applications Big Data : Google (engin de recherche), Netflix (suggestions), Instagram (images, commentaires, votes), Facebook (photos, posts, likes, suggestions d'amis) et tant d'autres.

Le premier problème d'une base de données répartie est le stockage des données. Il faut d'abord pouvoir répartir les données sur plusieurs machines. Nous allons voir deux approches : l'approche de la base de données relationnelle et l'approche d'un système de fichiers distribué comme le Hadoop File System. HDFS enregistre le nouveau data comme un log, ce qui facilite énormément le scaling horizontal et permet également d'utiliser les disques durs conventionnels plus efficacement.

Le deuxième défi est de pouvoir faire des requêtes sur les données. Dans le cas d'un système relationnel, on utilise le langage SQL. Dans le cas d'un système comme celui proposé dans la Lambda Architecture, on utilise plutôt MapReduce, un pipeline d'agrégation (MongoDB) ou une autre technologie pouvant être exécutée sur le cluster comme Apache Spark. Apache Spark permet d'exécuter MapReduce en mémoire vive, ce qui accélère énormément le temps de calcul comparé au MapReduce de Hadoop.

Il faut également apprendre à traiter des données realtime car la faiblesse des technologies de batch computing (Hadoop MapReduce, Spark) est le temps de réponse. Parfois il faut plusieurs heures pour construire des vues à partir du log des données. Certains site webs ou applications ne peuvent pas se permettre d'afficher leurs nouvelles données quelques heures en retard. La solution de la Lambda Architecture est donc d'avoir une couche d'application qui ne s'occupe que des données récentes. Cette couche utilise des technologies et base de données différentes. On parle ici de bases de données incrémentales comme MySQL, Cassandra, de système de Stream Processing comme Apache Storm et Apache Spark et d'une file distribuée comme Apache Kafka.

Le cours a comme objectif de peindre un portrait du monde du Big Data. En premier, on va regarder comment on fonctionne avec un système relationnel lorsqu'on est confronté au Big Data. On va apprendre à connaître la Lambda Architecture, une architecture à trois couches qui permet d'aller répondre à tous les critères d'un système Big Data.

À l'intérieur de la Lambda Architecture, nous allons voir les technologies de bases de données ainsi que les technologies de programmation sur le Big Data (MapReduce avec MongoDB, RDD et GraphX avec Spark)

Deux langages de programmation seront utilisés : Javascript pour le MapReduce avec MongoDB et Scala pour toute la programmation sur Spark.

Certains IDE seront utilisés : IntelliJ IDE pour utiliser Spark avec le langage Scala et WebStorm pour utiliser Javascript avec Node.js et MongoDB.

## **Objectifs du cours**

Au terme de ce cours, l'étudiant aura acquis les compétences suivantes:

- Retour sur la BD relationnelle centralisée.
- Normalisation (1FN, 2FN, 3FN)
- Définir les concepts importants relatifs aux bases de données réparties
- Fragmentation horizontale et verticale
- La Lambda Architecture : Batch Layer, Serving Layer et Speed Layer
- Programmation avec MapReduce
- Utilisation de MongoDB
- Programmation avec Apache Spark
- Programmation avec le module GraphX de Spark
- Connaissance approfondie de Spark, et de son fonctionnement interne.
- Programmation d'un Crawler
- Connaissance du langage Scala
- Connaissance du langage Javascript
- Présentation d'algorithmes distribués plus avancés.
- Des algorithmes et structures de données importantes utilisés en bases de données réparties : Hyperloglog, Log Structured Merge Tree, Bloom Filter etc.

## **Sujets abordés**

Le cours se divise en deux grandes parties.

1. La première moitié du cours est théorique. On fait un retour sur les approches de bases de données traditionnelles (Modèle relationnel et normalisation) et on parle du problème de scaling. On introduit ensuite la Lambda Architecture, une architecture pour faire des sites internet et applications Big Data. Cette architecture demande l'utilisation de certaines technologies et algorithmes. Ces derniers sont par la suite introduits. On parle ici de MapReduce, HyperLogLog, Bloom Filter, Log Structured Merge Tree etc. Ensuite, on regarde

comment certaines bases de données populaires (Cassandra, Redis) font pour gérer une demande de stockage venant de millions d'utilisateurs. On introduit également le pattern de programmation MapReduce.

2. La deuxième moitié du cours porte sur la programmation avec Apache Spark, un framework populaire de cluster-computing. Tout comme une bonne connaissance du langage SQL permet de traiter les données d'une base de données relationnelle, une bonne connaissance de la technologie Spark permet de traiter les données d'un log réparti ou d'un stream très efficacement. Spark peut également être utilisé pour programmer des algorithmes distribués. On montre également à l'étudiant comment optimiser ses algorithmes Spark en expliquant le fonctionnement interne de cette technologie. On y voit donc comment utiliser le cache, et le checkpointing.

Puisque ce cours est donné en ligne, la méthode pédagogique change. Le professeur et les élèves ont un espace de discussion commun avec le logiciel gratuit Discord. Le professeur se rend disponible toute la semaine pour les étudiants. Pendant la semaine, le professeur donne des lectures aux étudiants, et apporte également une assistance pour l'installation des technologies essentielles à leur travail. Finalement, les étudiants doivent se mettre en équipe et travailler sur deux travaux maisons. Ce sont ces travaux qui sont évalués à la fin du cours. Le professeur entreprend une entrevue audio ou vidéo avec chaque étudiant à un certain moment vers la fin de la session.

Le lien vers le serveur Discord est communiqué par email aux étudiants.

## Évaluation

L'évaluation consistera en les éléments suivants:

- o Deux travaux à faire (1 à 4 personnes) à remettre les **19 Octobre 2020** et **17 Décembre 2020**. Les travaux seront composés de questions à développement ainsi que d'exercices de programmation faisant appel aux technologies et aux concepts présentés en classe. Tout travail remis en retard sans motif valable sera pénalisé de 10% par jour de retard.

Le premier travail compte pour **30%** de la note, et le deuxième travail compte pour **50%** de la note. Le professeur effectue une entrevue avec chaque étudiant à la fin du cours. Cette entrevue a pour but de converser avec l'étudiant pour voir son aisance avec le contenu du cours. L'entrevue dure au maximum 20 minutes. Elle compte pour **20%** de la note finale.

La note finale, sur 100, est la somme de toutes ces évaluations. La note de passage est fixée à **60%** ou D. La date de la fin de la session est le 21 Décembre 2020.

## Date limite d'abandon sans mention d'échec

20% de l'évaluation aura été transmise à l'étudiant avant la date limite d'abandon sans mention d'échec, soit le 9 novembre 2020.

## **Qualité du français écrit**

Tout travail remis doit être conforme aux exigences de la politique institutionnelle en matière de maîtrise du français écrit. Comme il en est fait mention dans le Manuel de gestion (3.1.1-012).<sup>1</sup> Tout travail dont la qualité du français serait jugée non conforme par l'enseignant pourra être pénalisé jusqu'à concurrence de 10% du résultat maximal prévu.

## **Évaluation de la qualité de l'enseignement**

Ce cours sera évalué en fonction de la Procédure relative à l'évaluation des activités aux programmes d'études de cycles supérieurs (Manuel de gestion, 3.1.2-008).

## **Formule pédagogique**

Le cours sera dispensé en classe par un professeur. Il n'y aura pas de séance de travaux pratiques associées à ce cours.

## **Situation du cours dans le programme**

Le cours est optionnel pour les étudiants inscrits aux programmes de cycles supérieurs. Aucun cours ne lui est préalable.

## **Références**

Aucun livre ou recueil de notes n'est obligatoire pour ce cours. Les diapositives utilisées durant les séances et les lectures préalables seront rendues disponibles en format électronique sur le site du cours.

Ces livres ont été utilisés pour construire le cours :

- Nathan Marz (2015). Big Data: Principles and best practices of scalable realtime data systems
- Michael S. Malak and Robin East (2016). Spark GraphX in Action
- Petar Zečević and Marko Bonaći (2016). Spark in Action
- Holden Karau, Rachel Warren (2017). High Performance Spark

---

<sup>1</sup>[http://www.uqac.ca/direction\\_services/secretariat\\_general/manuel/index.pdf](http://www.uqac.ca/direction_services/secretariat_general/manuel/index.pdf)