



REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE



UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHÉMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Informatique

Par :

MENNAD Ahlam

Sur le thème

Gestion multitâche temps réel optimisée d'allocation et de libération de ressources de calcul

Soutenu publiquement 2021 /2022 à Tiaret devant le jury composé de :

MrKHAROUBI Sahraoui

Grade Université MCA

Président

Mr BELARBI Mostefa

Grade Université Pr

Encadreur

Mr ZIOUAL Tahar

Grade Université MCB

Examineur

2021-2022

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce modeste travail à son terme.

Je remercie chaleureusement mon directeur de thèse, le Dr BELARBIMostefa, qui a eu à cœur de développer mes recherches et a apporté toute sa contribution à la mise en œuvre de ce travail. Ses qualités humanistes et scientifiques, ses conseils et son enthousiasme ont été des leçons de père en fille, ils m'ont été très utiles durant cette année de master.

Je tiens également à remercier Dr. KHAROUBI Sahraoui et Dr. ZIOUAL Tahar qui ont accepté d'examiner mon travail, et exprime mon honneur ainsi que ma gratitude de les avoir comme jury dans ma soutenance.

Je tiens à remercier tous les membres du Laboratoire de Mathématiques et Informatique (LIM) de l'Université Ibn Khaldoun de Tiaret.

Résumé

Sur le marché actuel, qui est piloté par les données, les algorithmes et les applications collectent des données sur les personnes, les processus, les systèmes et les entreprises, ce qui entraîne la génération de volumes de données considérables. Le défi consiste à définir la solution qui permettra de traiter ces volumes de données rapidement, efficacement et sans pertes de connaissance significatives.

C'est là qu'intervient le modèle de programmation MapReduce. Utilisé initialement par Google pour analyser ses résultats de recherche, MapReduce a gagné en popularité grâce à sa capacité à diviser et traiter plusieurs téraoctets de données en parallèle et à obtenir ainsi des résultats plus rapides.

Notre projet de fin d'étude vise à étudier les technologies Hadoop, et nous nous intéresserons particulièrement à ses composants HDFS et MapReduce. Dans ce sens, nous mettons en place un système multitâches temps réel pour intégrer les fonctions principales de MapReduce.

Liste des figures

Figure 1 : Composants fondamentaux de Hadoop.	6
Figure 2 : Architecture de système de fichiers distribués d'Hadoop.	7
Figure 3 : Conception de HDFS.....	9
Figure 4 : Architecture logicielle MapReduce de Hadoop.....	11
Figure 5 : Processus MapReduce d'Hadoop.	14
Figure 6 : Architecture YARN.....	15
Figure 7 : Architecture Hadoop	16
Figure 8 : Architecture de base de Docker.....	23
Figure 9 :Traditionnel Linux Conteneurs.....	27
Figure 10 :Traditionnel Linux Conteneurs vs Docker.....	28
Figure 11 :Hyperviseur de type 1.....	31
Figure 12 :Hyperviseur de type 2.....	31
Figure 13 : Virtualisation vs Conteneurs.	35
Figure 14 : Le système de fichiers du noyau « FreeRTOS ».	39
Figure 15 : Les états du tache sure FreeRTOS.....	39
Figure 16 : Observation de marché des OS et RTOS pour l'embarqué.	41
Figure 17 : exemple de « Prioritized Preemptive Scheduling with Time Slicing »	42
Figure 18 : «Carte Arduino UNO ».....	43
Figure 19 : « Arduino 1.8.1 IDE ».	43
Figure 20 : Observation de marché des cartes pour les systèmes embarqués	45
Figure 21 : « les différents modèles des carte ARDUINO ».	46
Figure 22 : Processus MapReduce d'Hadoop.	49
Figure 23 : diagramme de contexte.....	51
Figure 24 : Diagramme de flot de données.	51
Figure 25 : Diagramme de flot de contrôle.	52
Figure 26 :Schéma multitâche.....	52
Figure 27 :Schéma de câblage.....	52
Figure 28 : Le chronogramme de la séquence d'exécution.	53
Figure 29 : la tâche afficher.	53
Figure 30 :la boite aux lettres.....	54
Figure 31 : Diagramme de contexte.	54

Figure 32 :Diagramme de flot de données.	54
Figure 33 : Diagramme de flot de contrôle.	55
Figure 34 : Schéma multitâche.....	55
Figure 35 : la tâche split.	56
Figure 36 : Digramme de contexte.....	56
Figure 37 : Digramme De Flot De Données.	56
Figure 38 : Diagramme de flot de contrôle.	57
Figure 39 : schéma multitâche.	57
Figure 40 : la tâcheMap.....	58
Figure 41 : Diagramme de contexte.	58
Figure 42 : Diagramme de flot de données.	60
Figure 43 : diagramme de flot de contrôle.	59
Figure 44 : schéma multitâche.	60
Figure 45 : la tâche shuffle.....	60
Figure 46 :La boîte aux lettres.	61
Figure 47 : diagramme de contexte.	61
Figure 48 : Digramme de flot de données.	62
Figure 49 : Diagramme de flot de contrôle.	62
Figure 50 : Schéma Multitâche.	62
Figure 51 : la tâche reduce.	62
Figure 52 :Diagramme de contexte.....	64
Figure 53 : Digramme de flot de données.	64
Figure 54 : Diagramme de flot de contrôle	65
Figure 55 : Schéma multitâche.....	63
Figure 56 : output.	66
Figure 57 :la tâche boîte aux lettres....	66
Figure 58 : diagramme de contexte.	67
Figure 59 : Digramme de flot de données.	67
Figure 60 : diagramme de flot de contrôle.	68
Figure 61 : schéma multitâche.	68

Liste des tableaux

Tableau 01 : Raccourcies de barre d'actions.46

Table de matières

Remerciements	
Dédicace	
Résumé	
Liste d'abréviations	
Liste des figures	
Liste des tableaux	
Introduction générale	

Chapitre 1

Hadoop : Un Framework pour le traitement des données volumineuses et distribuées

1.1 Introduction	05
1.2 Big data définition	05
1.3 Hadoop définition	05
1.4 Composants fondamentaux de Hadoop	06
1.4.1 Hadoop Distributed file system (HDFS)	06
1.4.2 MapReduce	10
1.4.2.1 Architecture MapReduce de Hadoop	11
1.4.2.2 Fonctionnement	13
1.4.3 Hadoop YARN	14
1.5 Architecture Hadoop	16
1.6 Caractéristiques	17
1.7 Problèmes et défis	17
1.8 Conclusion	20

Chapitre 02

Docker : Une Plateforme pour créer et déployer facilement les applications distribuées

2.1 Introduction	22
2.2 Définition docker	22

2.2.1 Les conteneurs Docker	253
2.2.2 docker Fonctionnement	23
2.4.3 Les avantages des conteneurs Docker	24
2.2.4 Avantages de Docker	25
2.2.5 les limites à l'utilisation de Docker	26
2.3 Un conteneur Linux.....	27
2.3.1 projet Linux Containers (LXC)	27
2.3.2 LXC fonctionnement.....	27
2.4 Traditionnel linux conteneurs vs docker	28
2.5 La virtualisation.....	29
2.5.1Définition de la virtualisation.....	29
2.5.2Types de virtualisation	30
2.5.3 La virtualisation système.....	30
2.5.4 La virtualisation applicative	31
2.5.4.1 La virtualisation par conteneurs	32
2.5.5 Vue d'ensemble des conteneurs.....	32
2.5.6. Conteneurs vs machines virtuelles	33
2.7Conclusion.....	35

Chapitre 03

Les plateformes matérielles et logicielles utilisées

3.1 Introduction	38
3.2 L'exécutif temps réel FreeRTOS	38
3.2.1 Pourquoi FreeRTOS ?	40
3.2.2 Fonctionnalités de FreeRTOS :	41
3.2.3 Algorithmes d'ordonnancement de FreeRTOS :.....	42
3.3Les cartes ARDUINO	42
3.3.1 Pourquoi choisir Arduino ?	44
3.3.2 Types de cartes Il y a trois types de cartes :	45
3.3.3 Différentes cartes Des cartes Arduino.....	46
3.4 Conclusion.....	46

Chapitre 04

Conception et implémentation

4.1. Introductions.....	48
4.2 Spécification informelle	48
4.2.1 Aspect matériel :.....	48
4.2.2 Aspect fonctionnel :	49
4.2.3 Le modèle de programmation MapReduce	49
4.2.4 Longueur moyenne des mots.....	50
4.3 Spécification formelle	50
4.3.1 Noyau :	51
4.3.1.1 Spécification :.....	55
4.3.1.2 Conception	52
4.3.1.3 Test.....	52
4.3.2 Transaction du noyau à l'incrément 01 :.....	53
4.3.2.1 Les nouvelles fonctionnalités :	53
4.3.3 Incrément 01 :.....	54
4.3.3.1 Spécification :.....	54
4.3.3.2 Conception	55
4.3.4 Transaction du l'incrément 01à l'incrément 02.....	55
4.3.4.1 Les nouvelles fonctionnalités	55
4.3.5 Incrément 02.....	56
4.3.5.1 Spécification :.....	56
4.3.5.2 Conception	57

4.3.6 Transaction du l'incrément 02 à l'incrément 03.....	58
4.3.6.1 Les nouvelles fonctionnalités	58
4.3.7 Incrément 03.....	58
4.3.7.1 Spécification :.....	58
4.3.7.2 Conception	59
4.3.8 Transaction du l'incrément 03 à l'incrément 04.....	60
4.3.8.1 Les nouvelles fonctionnalités	60
4.3.9 Incrément 04.....	61
4.3.9.1 Spécification :.....	61
4.3.9.2 Conception	63
4.3.10 Transaction du l'incrément 04 à l'incrément 05.....	63
4.3.10.1 Les nouvelles fonctionnalités	63
4.3.11 Incrément 05.....	64
4.3.11.1 Spécification :.....	64
4.3.11.2 Conception	65
4.3.12 Transaction du l'incrément 05 à l'incrément 06.....	64
4.3.12.1 Les nouvelles fonctionnalités	66
4.3.13 Incrément 06.....	67
4.3.13.1 Spécification :.....	67
4.3.13.2 Conception	68
Conclusion générale	70
Bibliographie.....	72

Introduction générale

Introduction générale

De nos jours, des très grandes quantités de données circulent de manière massive, à partir de diverses sources de réseaux sociaux, internet, Google, appareils mobiles, système GPS... etc. Ces données circulent d'une manière rapide et en temps réel et à grande échelle. Elles sont de différents types et structures, On trouve des textes, des audio, des images et des vidéo, le big data est un terme apparu avec l'augmentation de ces données.

Le défi confronté dans ce contexte les systèmes classiques et les entrepôts de données rencontrent le problème de la dégradation des performances face à une quantité de données aussi importante en termes d'analyse et de traitement.

Le Big data résout ce problème et plusieurs domaines utilisent le Big data comme une solution optimale pour la gestion des données parmi ces domaines le domaine de sélection des candidats qu'ils profitent de Big Data pour améliorer la sélection dans les entreprises.

Plusieurs modèles de programmation ont apparu, et l'une des techniques les plus puissantes de traitement et l'analyse des données est le Framework Hadoop, l'une des techniques les plus répandues et les plus utilisées Hadoop est un environnement d'exécution distribuée, performant et scalable, il propose un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File System) et un système d'analyse et de traitement de données basé sur le modèle de programmation MapReduce pour réaliser des traitements parallèles et distribués sur des gros volumes de données.

L'objectif de notre travail est d'étudier les questions liées à la mise en œuvre efficace des problèmes très distribuables à travers les ensembles de données volumineux à l'aide des modèles de programmation basés sur MapReduce. En particulier, le travail se concentrera sur la conception incrémentale d'une application multitâche temps réelle dotée des fonctionnalités MAPREDUCE.

Notre mémoire est organisé comme suit :

Le premier chapitre : nous allons présenter une introduction du Framework Hadoop, les caractéristiques de ses principaux composants HDFS et de MapReduce.

Le deuxième chapitre: est consacré à la présentation de la Plateforme Docker les concepts liés à la virtualisation traditionnelle et à la conteneurisation. L'objectif est de les décrire et expliquer les liens qui existent entre eux.

Le troisième chapitre: Nous présenterons le cycle de vie et la méthode de développement d'un système temps réel.

Le quatrième chapitre : présente toute la procédure de réalisation de notre travail, nous donne une vue détaillée de notre système, et présente les différents étapes de développement du noyau conçu.

Enfin, Nous concluons par une conclusion générale.

Chapitre I

Hadoop : Un Framework pour le traitement des données
volumineuses et distribuées

1.1 Introduction

Ces dernières années, nous avons perçu l'essor d'un nouveau phénomène dans le monde de la technologie, nous nous référons au Big Data.

La croissance exponentielle des données, provenant de différentes sources et formats, nous offrons une excellente occasion d'avoir plus d'informations, la qui peut être un outil précieux pour atteindre nos objectifs, mais dans le même temps, nous avons un grand défi, la possibilité de recueillir, comprendre, analyser et utiliser efficacement ces données.

Dans ce contexte, Apache Hadoop apparaît sur le marché comme une solution pour ce type problèmes, en utilisant un groupe d'ordinateurs travaillant ensemble sur une même tâche, partageant la charge de travail et le stockage des données. Cela peut sembler simple, mais la distribution entraîne des défis majeurs tels que la complexité de la programmation, la synchronisation des données et le traitement ou la gestion des pannes.

Dans ce chapitre, nous décrivons l'architecture de base de Hadoop, son fonctionnement, ses caractéristiques, ainsi que ses problèmes et défis.

1.2 Big data définition

Compte tenu de sa popularité actuelle, la définition du Big Data est plutôt diversifiée et il est difficile de parvenir à un consensus. Fondamentalement, le big data signifie non seulement un grand volume de données mais aussi d'autres caractéristiques qui le différencient des concepts de « données massives » et de « très grandes données ». International Data Corporation (IDC) est un pionnier dans l'étude du big data et de son impact. Il définit le Big data dans un rapport de 2011 comme : « Les technologies Big Data décrivent une nouvelle génération de technologies et d'architectures, conçues pour extraire économiquement la valeur à partir de très gros volumes d'une large variété de données, en permettant une haute vélocité de capture, découverte et / ou d'analyse ». Cette définition délimite les quatre principales caractéristiques des big data «4Vs», à savoir le volume, la variété, la vélocité et la valeur. Le cabinet d'études Gartner en 2011 a noté que les défis et les opportunités de croissance des données sont tridimensionnels, c'est-à-dire qu'ils augmentent le volume, la vélocité et la variété. Une grande partie de l'industrie, y compris IBM et Microsoft utilisent ce modèle «3Vs» pour décrire big data(1).

1.3 Hadoop définition

Hadoop est un Framework logiciel open source permettant de stocker des données, et de lancer des applications sur des grappes de machines standards. Cette solution offre un espace de stockage massif pour tous les types de données, une immense puissance de traitement et la possibilité de prendre en charge une quantité de tâches virtuellement

illimitée. Basé sur Java, ce framework fait partie du projet Apache, sponsorisé par Apache Software Foundation.

Grâce au framework MapReduce, il permet de traiter les immenses quantités de données. Plutôt que de devoir déplacer les données vers un réseau pour procéder au traitement, MapReduce permet de déplacer directement le logiciel de traitement vers les données(2).

1. 4 Composants fondamentaux de Hadoop

Apache Hadoop inclut les modules suivants :

- **Hadoop Common (Hadoop Core):**Ensemble de bibliothèques et d'utilitaires communs dont dépendent les trois autres modules.
- **Hadoop Distributed File System (HDFS):**Système principal de stockage de données qui gère de grands ensembles de données exécutés sur du matériel de base.
- **Hadoop YARN :** Gestionnaire de ressources de cluster qui planifie les tâches et alloue des ressources (par exemple, CPU et mémoire) aux applications. .
- **Hadoop MapReduce :** Divise les tâches de traitement de Big Data en plus petites, répartit les petites tâches sur différents nœuds, puis exécute chaque tâche (3).

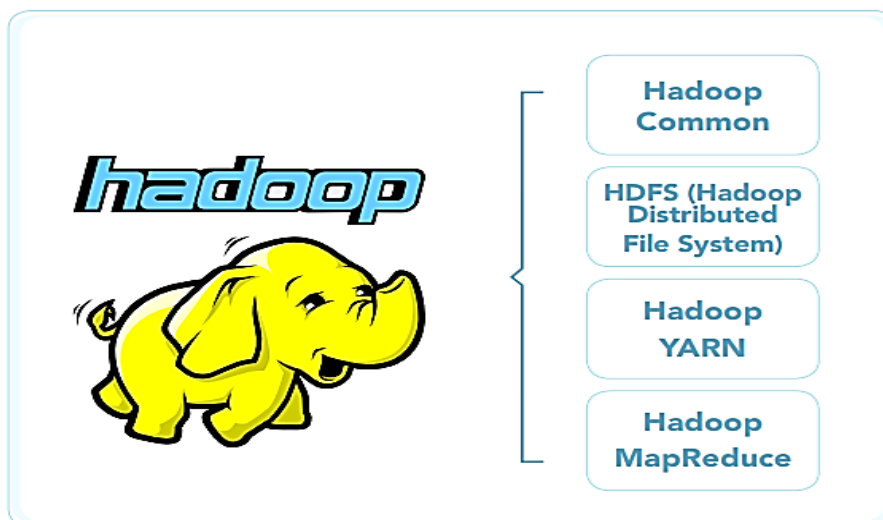


Figure1 : Composants fondamentaux de Hadoop (3).

1.4.1 Hadoop Distributed file system (HDFS)

HDFS est un système de stockage distribué conçu pour le stockage de fichiers distribués sur un nœud HDFS. HDFS a une architecture maître /esclave. Un cluster HDFS possède un NameNode unique, un serveur maître qui gère l'espace de nom du système de fichiers et contrôle l'accès aux fichiers par les clients.

Les données sont divisées en blocs de 64 Mo ou 128 Mo par HDFS et il maintient trois

Copies de chaque bloc sur des emplacements distincts appelés le DataNode. Le NameNode exécute des opérations d'espace de nom de système de fichiers telles que l'ouverture, la fermeture et le renommage de fichiers et de répertoires tandis que le nœud de données est responsable de la réservation des demandes de lecture et d'écriture des clients du système de fichiers. Il détermine également le mappage des blocs aux DataNode.

Les DataNode effectuent la création, la suppression et la réplication de blocs sur instruction du NameNode. HDFS donne un accès à haut débit et convient à l'application qui a de grands ensembles de données. Le HDFS est hautement tolérant aux pannes et peut être déployé sur du matériel bon marché.

Comme le montre la figure 2, l'architecture HDFS se compose d'un seul NameNode et de plusieurs DataNodes dans un cluster. NameNode décide de la réplication des blocs de données. Dans un HDFS typique, la taille de bloc est de 64 Mo et le facteur de réplication est de 3 (la deuxième copie sur le rack local et la troisième sur le rack distant). Système de fichiers distribués d'Hadoop (HDFS) : C'est un système de fichiers distribués(1).

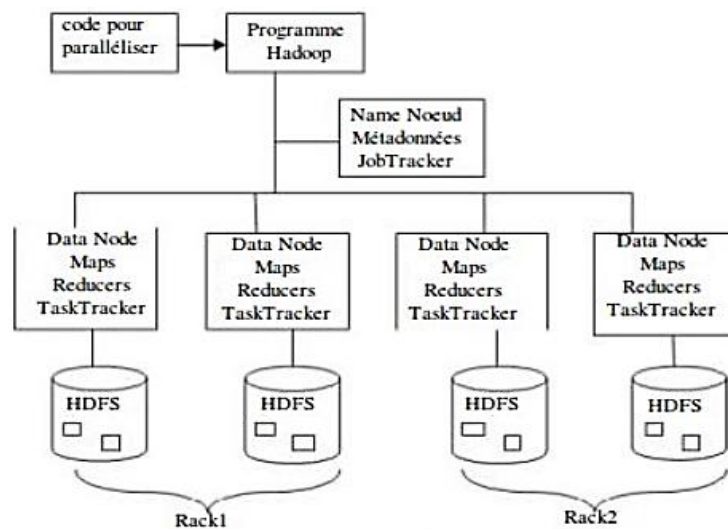


Figure 2: Architecture de système de fichiers distribués d'Hadoop (1).

Name Node

Name Node est un service maître. Il fonctionne dans un nœud unique et en tant que gestionnaire de cluster. La tâche principale de namenode est de gérer l'espace de noms du système de fichiers. L'arborescence du système de fichiers et les métadonnées de tous les fichiers et répertoires sont conservées dans le namenode. C'est l'arbitre et le référentiel de toutes les métadonnées HDFS. Il maintient et stocke l'arborescence des espaces de noms et le

mappage des blocs de fichiers aux nœuds de données présents sur le disque local sous la forme de deux fichiers :

- l'image de l'espace de noms
- le journal des modifications

Toutes les métadonnées du système de fichiers sont stockées sur un serveur de métadonnées. Toutes les opérations de métadonnées peuvent être gérées par un seul serveur de métadonnées, mais un cluster configurera plusieurs serveurs de métadonnées en tant que paires de basculement de sauvegarde principales. Cela inclut l'espace de noms, l'emplacement des données et les autorisations d'accès (4).

Opérations

Les clients contactent le nœud de nom afin d'effectuer des opérations courantes sur le système de fichiers, telles que l'ouverture, la fermeture, le changement de nom et la suppression. Le nœud de nom ne stocke pas les données HDFS lui-même, mais maintient plutôt un mappage entre le nom de fichier HDFS, une liste de blocs dans le fichier et le ou les nœuds de données sur lesquels ces blocs sont stockés. Le système est conçu de telle manière que les données utilisateur ne transitent jamais par le nœud de nom.

Il reçoit périodiquement un rapport Heartbeat et Block de chacun des nœuds de données présents dans le cluster. Lorsque namenode reçoit périodiquement un Heartbeat du nœud de données, cela signifie que datanode fonctionne correctement. Un rapport de bloc contient une liste de tous les blocs sur un nœud de données(4).

Format Name Node

Lorsque le NameNode est formaté, un ID d'espace de noms est généré, qui identifie essentiellement cette instance spécifique du système de fichiers distribué. Lorsque les DataNodes se connectent pour la première fois au NameNode, ils stockent cet ID d'espace de noms avec les blocs de données, car les blocs doivent appartenir à un système de fichiers spécifique.

Si un DataNode se connecte ultérieurement à un NameNode et que l'ID d'espace de noms déclaré par le NameNode ne correspond pas à l'ID d'espace de noms stocké sur le DataNode, il refusera de fonctionner avec l'erreur "ID d'espace de noms incompatible". Cela signifie que le DataNode s'est connecté à un NameNode différent et que les blocs qu'il stocke n'appartiennent pas à ce système de fichiers distribué (4).

NameNode Secondaire : Son rôle principal est de mixer l'image du NameNode avec le journal des transactions exécutées, pour éviter que le journal ne devienne trop volumineux.

Normalement, ce démon s'exécute sur une machine physique distincte, car ce processus nécessite beaucoup de CPU et beaucoup de mémoire. Conserve une copie de l'image de l'espace de noms afin qu'elle puisse être utilisée en cas d'échec du NameNode (4). **DataNode** : nœuds esclaves responsables de la lecture et de l'écriture des requêtes des clients. Lorsqu'un client demande une lecture ou une écriture de données, le fichier est divisé en blocs et le NameNode se charge de dire où se trouve ou sera stocké chacun de ces blocs.

De plus, les DataNodes communiquent avec d'autres nœuds pour répliquer les données augmentant la redondance et favorisant le contrôle contre les erreurs (4).

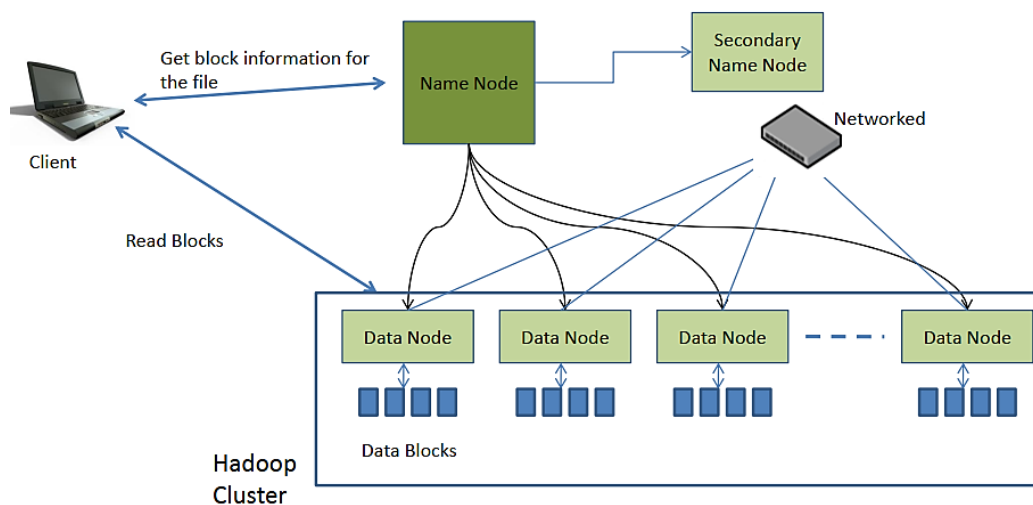


Figure 3 : Conception de HDFS (5)

Le système de fichiers HDFS est géré par deux démons

- **NameNode** et **DataNode**
- Fonction NameNode&DataNode en mode maître/esclave
- NameNode gère l'espace de noms du système de fichiers
- Maintient l'arborescence du système de fichiers et les métadonnées de tous les fichiers et répertoires
- Image du système de fichiers
- Modifier le journal
- Les Datanodes stockent et récupèrent les blocs pour les fichiers lorsqu'ils sont indiqués par NameNode
- NameNode conserve les informations sur lesquelles DataNodes tous les blocs d'un fichier donné sont situés
- Les DataNodes rapportent périodiquement à NameNode la liste des blocs qu'ils

Stockent

- Avec NameNode désactivé, le HDFS est inaccessible
- Noeud de nom secondaire
- Pas une sauvegarde pour le NameNode
- Aide simplement à fusionner l'image du système de fichiers avec le journal des modifications pour éviter que le journal des modifications ne devienne trop volumineux

(5).1.4.2 MapReduce

MapReduce est un modèle de programmation pour une technique de traitement basé sur l'informatique distribuée. Le modèle de programmation MapReduce a deux fonctions importantes, Map et Reduce.

Le fonctionnement de Map consiste à prendre les données d'entrée sous la forme d'un ensemble de paires de valeurs clés et à produire une sortie sous forme de paires de valeurs de clés intermédiaires supplémentaires. La fonction Reduce reçoit cette paire de valeurs de clé intermédiaire générée à partir de la fonction Map en tant qu'entrée et réduit ces paires de valeurs de clé intermédiaires en un ensemble de paires plus petit.

La fonction Map est exécutée après la fonction Reduce selon le nom du modèle de programme MapReduce.

Le principal avantage du modèle MapReduce est évolutif pour le traitement des données sur plusieurs nœuds de calcul. Il y a trois étapes Map, Shuffle et Reduce dans le traitement de MapReduce. Le traitement de l'étape Map consiste à traiter les données d'entrée et à créer divers petits paquets de données. HDFS est utilisé pour stocker les données d'entrée sous la forme d'un fichier ou d'un répertoire et ces fichiers d'entrée passent à la fonction Map pas à pas. Dans l'étape Reduce, les données de sortie reçues de la fonction Reduce proviennent de l'étape Map et sont utilisées comme données d'entrée, les traitent et produisent un nouvel ensemble de sorties stockées dans HDFS.

Hadoop envoie les tâches Map et Reduce au serveur approprié dans les clusters pendant le travail MapReduce. Il incombe à Hadoop de gérer les détails des tâches pertinentes, comme la vérification de l'exécution des tâches, l'émission des tâches et la copie des données autour du cluster entre les nœuds. Diverses tâches informatiques qui réduisent le trafic réseau sont placées sur des nœuds avec des données sur des disques locaux. Lorsque les tâches sont terminées, le cluster collecte les données de sortie et les réduit à un format approprié pour les renvoyer au serveur Hadoop (1).

1.4.2.1 Architecture MapReduce de Hadoop

Comme pour HDFS, la gestion des tâches de Hadoop se base sur deux serveurs (des daemons):

- **Le JobTracker**, qui va directement recevoir la tâche à exécuter (un .jar Java), ainsi que les données d'entrées (nom des fichiers stockés sur HDFS) et le répertoire où stocker les données de sortie (toujours sur HDFS). Il y a un seul JobTracker sur une seule machine du cluster Hadoop. Le JobTracker est en communication avec le NameNode de HDFS et sait donc où sont les données.

- **Le TaskTracker**, qui est en communication constante avec le JobTracker et va recevoir les opérations simples à effectuer (MAP/REDUCE) ainsi que les blocs de données correspondants (stockés sur HDFS). Il y a un TaskTracker sur chaque machine du cluster.

Comme le JobTracker est conscient de la position des données (grâce au NameNode), il peut facilement déterminer les meilleures machines auxquelles attribuer les sous-tâches (celles où les blocs de données correspondants sont stockés). Pour effectuer un traitement Hadoop, on va donc stocker nos données d'entrée sur HDFS, créer un répertoire où Hadoop stockera les résultats sur HDFS, et compiler nos programmes MAP et REDUCE au sein d'un .jar Java(6).

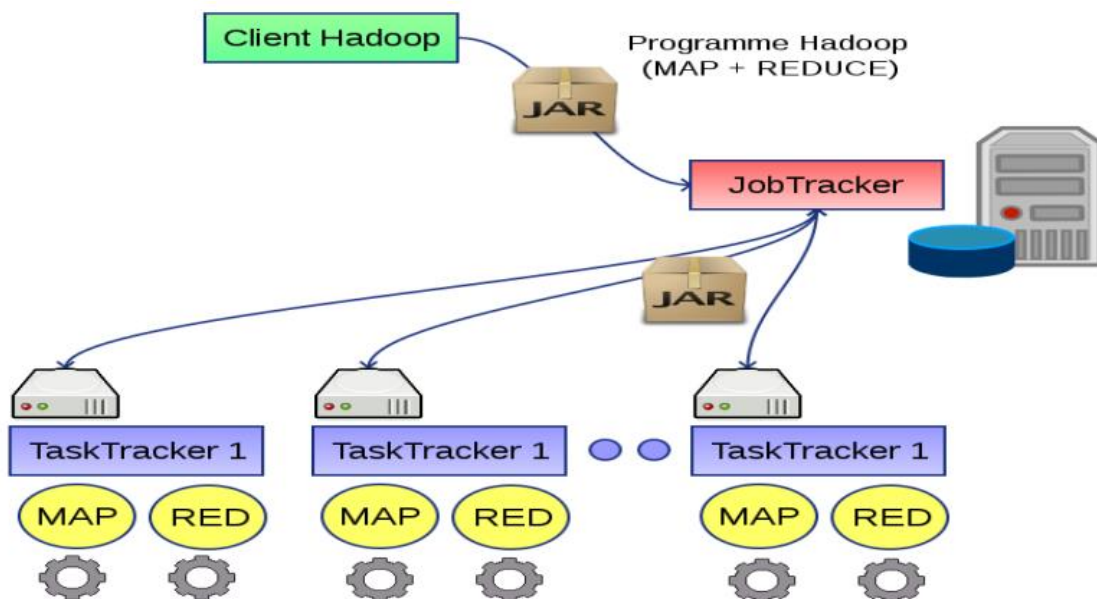


Figure 4: Architecture logicielle MapReduce de Hadoop (6)

On soumettra alors le nom des fichiers d'entrée, le nom du répertoire des résultats, et le .jar lui-même au JobTracker: il s'occupera du reste (et notamment de transmettre les programmes MAP et REDUCE aux serveurs TaskTracker des machines du cluster). L'entrée de MapReduce doit être fait à partir de fichiers dans HDFS où Chaque bloc contient une liste de paires clé-valeur(6).

Le JobTracker

Le déroulement de l'exécution d'une tâche Hadoop suit les étapes suivantes du point de vue du JobTracker :

1. Le client (un outil Hadoop console) va soumettre le travail à effectuer au JobTracker : une archive java .jar implémentant les opérations Map et Reduce. Il va également soumettre le nom des fichiers d'entrée et l'endroit où stocker les résultats.
2. Le JobTracker communique avec le NameNode HDFS pour savoir où se trouvent les blocs correspondant aux noms de fichiers donnés par le client.
3. Le JobTracker, à partir de ces informations, détermine quels sont les nœuds TaskTracker les plus appropriés, c'est à dire ceux qui contiennent les données sur lesquelles travailler sur la même machine, ou le plus proche possible (même rack/rack proche).
4. Pour chaque « morceau » des données d'entrée, le JobTracker envoie au TaskTracker sélectionné le travail à effectuer (MAP/REDUCE, code Java) et les blocs de données correspondants.
5. Le JobTracker communique avec les nœuds TaskTracker en train d'exécuter les tâches. Ils envoient régulièrement un « heartbeat », un message signalant qu'ils travaillent toujours sur la sous-tâche reçue. Si aucun heartbeat n'est reçu dans une période donnée, le JobTracker considère la tâche comme ayant échoué et donne le même travail à effectuer à un autre TaskTracker.
6. Si par hasard une tâche échoue (erreur java, données incorrectes, etc.), le TaskTracker va signaler au JobTracker que la tâche n'a pas pu être exécutée.
7. Le JobTracker va alors décider de la conduite à adopter :
 - Demander au même TaskTracker de réessayer.
 - Redonner la sous-tâche à un autre TaskTracker.
 - Marquer les données concernées comme invalides, etc.
 - Il pourra même blacklister le TaskTracker concerné comme non-fiable dans certains cas.
8. Une fois que toutes les opérations envoyées aux TaskTracker (MAP + REDUCE) ont été effectuées et confirmées comme effectuées par tous les nœuds, le JobTracker marque la tâche comme « effectuée ». Des informations détaillées sont disponibles (statistiques, TaskTracker ayant posé problème, etc.) (6).

➤ **Remarques** : Par ailleurs, on peut également obtenir à tout moment de la part du JobTracker des informations sur les tâches en train d'être effectuées : étape actuelle (MAP, SHUFFLE, REDUCE), pourcentage de complétion, etc (6).

Le TaskTracker

Le TaskTracker dispose d'un nombre de « slots » d'exécution. A chaque « slot » correspond une tâche exécutable (configurable). Ainsi, une machine ayant par exemple un processeur à 8 cœurs pourrait avoir 16 slots d'opérations configurées.

Lorsqu'il reçoit une nouvelle tâche à effectuer (MAP, REDUCE, SHUFFLE) depuis le JobTracker, le TaskTracker va démarrer une nouvelle instance de Java avec le fichier .jar fourni par le JobTracker, en appelant l'opération correspondante.

Une fois la tâche démarrée, il enverra régulièrement au JobTracker ses messages heartbeats. En dehors d'informer le JobTracker qu'il est toujours fonctionnels, ces messages indiquent également le nombre de slots disponibles sur le TaskTracker concerné.

Lorsqu'une sous-tâche est terminée, le TaskTracker envoie un message au JobTracker pour l'en informer, que la tâche se soit bien déroulée ou non (il indique évidemment le résultat au JobTracker(6)).

1.4.2.2 Fonctionnement

Lorsqu'un utilisateur soumet un travail à Hadoop. Il doit spécifier l'emplacement des fichiers d'entrée et de sortie dans HDFS. La fonction Map and Reduce est implémentée dans les classes Java sous la forme de fichiers jar et les différents paramètres doivent être corrigés afin de permettre l'exécution des tâches pour la configuration des tâches. Ensuite, l'utilisateur de Hadoop soumet un travail la configuration du travail au Job Tracker. Maintenant Job Tracker distribue le travail et les configurations aux tâches de planification. .

Il contrôle également les travaux et fournit des informations sur l'état et le diagnostic à l'utilisateur Hadoop. La responsabilité de TaskTracker est d'exécuter une tâche sur un nœud différent selon l'implémentation de MapReduce et de sauvegarder la sortie dans les fichiers de sortie sur HDFS(1).

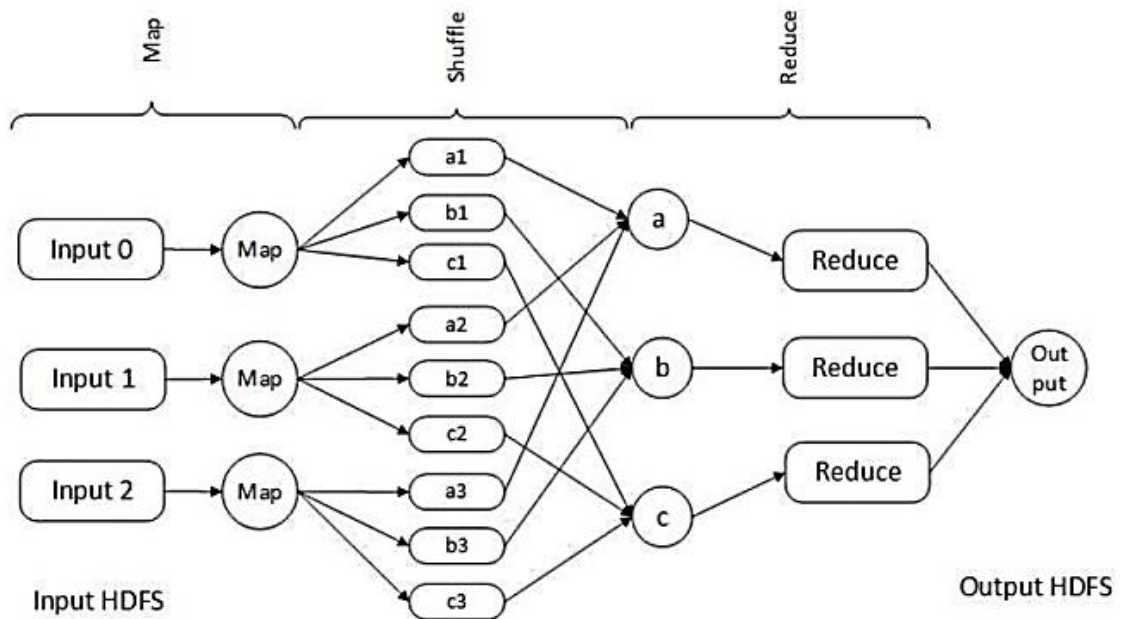


Figure 5: Processus MapReduce d’Hadoop(1).

La figure 5 : Définie la méthode traditionnelle de MapReduce, mais maintenant Hadoop travaille avec un nouveau paramètre nommé YARN qui donne une approche vraiment nouvelle au traitement des données, ce qui donne un moyen facile de traiter les données avec une seule plate-forme et fournit une nouvelle perspective analytique (1).

1.4.3 Hadoop YARN

Dans la nouvelle génération de Framework Hadoop, YARN (YetAnother ResourceNégociateur) est une partie architecturale de base (Figure 6) qui permet le traitement dedonnées multiples comme le traitement par lots, la science des données, le streaming temps réel et le SQL interactif. Elle permet aussi de gérer le traitement des données avec une plateforme unique et fournir une nouvelle perspective à l’analyse.

Le but fondamental de YARN est de diviser les fonctionnalités de JobTracker / TaskTracker en démons séparés. Dans cette nouvelle version de Hadoop, un gestionnaire de ressources global, un gestionnaire de nœud esclave pour chaque nœud, maître d’application et conteneur pour chaque application (un seul travail ou peut-être un DAG de travaux) s’exécute sur un gestionnaire de nœud (1).

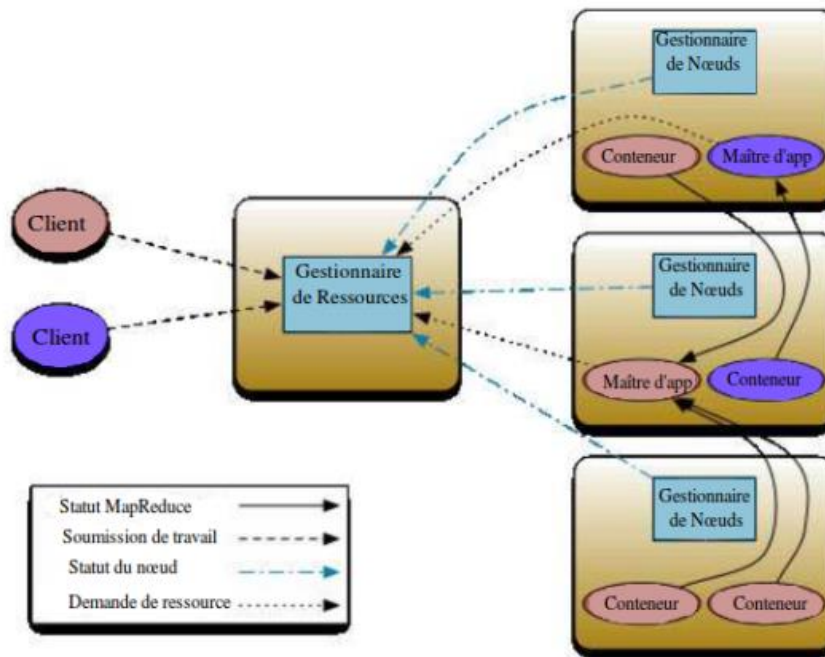


Figure 6: Architecture YARN (1).

Le Gestionnaire de Ressources et le Gestionnaire de Nœuds fonctionnent ensemble comme un framework traitement-données. Un gestionnaire de ressources global s'exécute en tant que démon maître qui arbitre les ressources de cluster parmi les applications concurrentes disponibles dans le système.

Le travail de gestionnaire de ressources consiste à suivre les nœuds s'exécutant sur le cluster et les ressources disponibles. Il prend également en considération les demandes soumises par l'utilisateur devrait obtenir des ressources chaque fois que nécessaire. Il est responsable d'allouer ces ressources à l'application utilisateur en fonction d'une priorité d'application, d'une capacité de file d'attente, de la localité de données, etc. Dans Hadoop2.0 TaskTracker est remplacé par Node Manager qui est une version plus générique et efficace. Le gestionnaire de nœud n'a pas de conteneur de ressources créé dynamiquement ou slots Map et Reduce fixe comme dans TaskTracker. La taille du conteneur dépend du nombre de ressources qu'il contient comme la mémoire, le disque, CPU, réseau, etc. Le gestionnaire de nœuds est responsable de l'application du conteneur et surveille l'utilisation de ses ressources comme le CPU, le disque, la mémoire et le réseau.

Dans Hadoop2.0 Job Tracker est légèrement modifié avec un peu de perspicacité de conception et nous utilisons la terminologie pour cela en tant que Application Master. Le maître d'application a la capacité de n'importe quel type de tâche à l'intérieur du

conteneur. Ainsi, lorsqu'un utilisateur soumet une application, Le maître d'application est chargé de coordonner l'exécution de toutes les tâches dans l'application. Le maître d'application a une plus grande responsabilité qui était précédemment assignée à Job Tracker pour tous les travaux comme la tâche de surveillance, le démarrage des tâches échouées, le calcul des valeurs totales du compteur d'applications et l'exécution spéculative de tâches lentes. Le gestionnaire de nœuds contrôlait le maître d'application et les tâches qui appartenaient à l'application s'exécutant dans le conteneur de ressources. Dans YARN le rôle de l'application distribuée est dégradé par MapReduce mais il est toujours très populaire et utile et maintenant il est connu comme MapReduce version 2 (MR2). MR2 est une implémentation classique du moteur MapReduce qui s'exécute en haut de YARN. MapReduce dans Hadoop2 maintient la compatibilité de l'API avec la version stable précédente alors que tous les travaux de MapReduce fonctionnent toujours avec YARN via une recompilation (1).

1.5 Architecture Hadoop

De manière similaire au NameNode de HDFS, il n'y a qu'un seul JobTracker et s'il tombe en panne, le cluster tout entier ne peut plus effectuer de tâches. Des résolutions aux problèmes sont ajoutées dans la version 2 de Hadoop.

Généralement, on place le JobTracker et le NameNode HDFS sur la même machine (une machine plus puissante que les autres), sans y placer de TaskTracker/DataNode HDFS pour limiter la charge. Cette machine particulière au sein du cluster (qui contient les deux « gestionnaires », de tâches et de fichiers) est communément appelée le nœud maître (« Master Node »). Les autres nœuds (contenant TaskTracker + DataNode) sont communément appelés nœuds esclaves (« slave node ») (6).

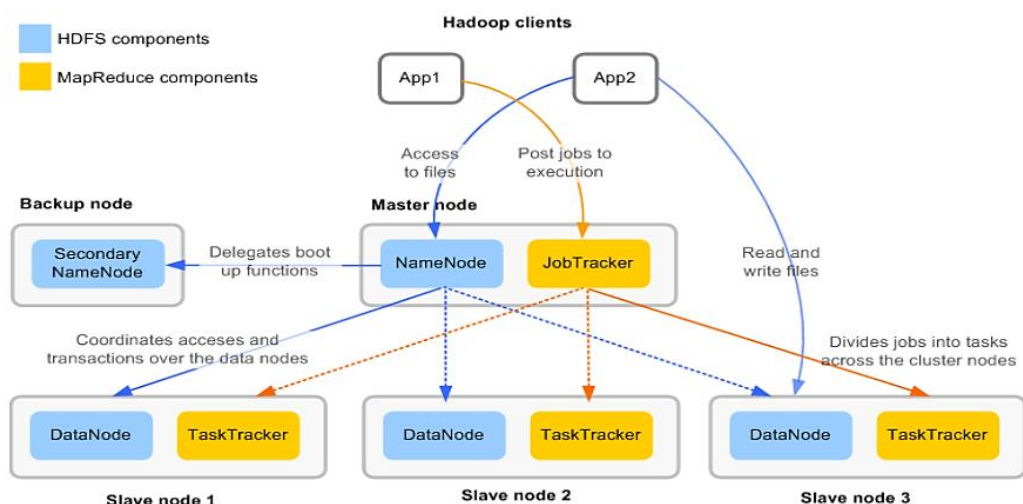


Figure 7: Architecture Hadoop (6)

1.6 Caractéristiques

Hadoop présente de nombreux avantages, et les fonctionnalités suivantes rendent Hadoop particulièrement adapté à la gestion et à l'analyse de Big Data :

- **Évolutivité** : Hadoop permet d'augmenter et de réduire l'infrastructure matérielle sans avoir à modifier les formats de données. Le système redistribuera automatiquement les données et les tâches de calcul pour s'adapter aux changements de matériel.
- **Rentabilité** : Hadoop apporte un calcul massivement parallèle aux serveurs de base, conduisant à une réduction importante du coût par téraoctet de stockage, ce qui rend le calcul massivement parallèle abordable pour le volume croissant big data.
- **Flexibilité** : Hadoop est libre de tout schéma et capable d'absorber n'importe quel type de données provenant d'un nombre quelconque de sources. En outre, différents types de données provenant de sources multiples peuvent être agrégés dans Hadoop pour une analyse plus approfondie. Ainsi, de nombreux défis de Big Data peuvent être résolus.
- **Tolérance aux pannes** : les données manquantes et les échecs de calcul sont courants dans les analyses de Big Data. Hadoop peut récupérer les données et les échecs de calcul causés par une panne de nœud ou une congestion du réseau (1).

1.7 Problèmes et défis

Les ordonnanceurs Hadoop sont conçus pour une meilleure utilisation des ressources et une amélioration des performances. Les performances de l'ordonnanceur ont été affectées par certains problèmes et des efforts de recherche sont encore nécessaires pour améliorer l'efficacité des ordonnanceurs Hadoop.

Certains des principaux défis de recherche sont discutés ici comme Efficacité énergétique, Schéma de Mapping, Équilibrage de charge, Automatisation et configuration, optimisation de la réorganisation des données, Optimisation des performances, Équité, Localité des données et Synchronisation.

Efficacité énergétique

La croissance continue de la taille des données dans un data center contenant un cluster Hadoop pour prendre en charge de nombreux utilisateurs et opérations d'un grand nombre de données doivent être effectuées dans ce data center. Pour surmonter ces opérations, une grande quantité d'énergie requise dans le centre de données et augmenter l'énergie dans le data center augmentera le coût global. La minimisation de l'énergie dans le centre de données est un grand défi pour Hadoop.

Equilibrage de charge

Il y a deux étapes Map et Reduce liés avec l'étape de partition. La partition est considérée comme le facteur principal pour mesurer les performances.

Par défaut, les données sont également réparties par les algorithmes de partition et gèrent le déséquilibre du système en cas de données asymétriques. Parce que seule la clé est considérée dans le traitement pas la taille des données. Donc, le problème d'équilibrage de charge se produit dans cette situation.

Schéma de mapping

Il est nécessaire d'affecter des ensembles d'entrée au réducteur afin que le réducteur reçoive toutes les entrées possibles avant l'exécution pour chaque sortie. Aucune entrée fixe ne peut être assignée parce que le réducteur a une capacité limitée d'entrées dues tandis que la taille d'entrée individuelle peut très bien conduire à augmenter le coût de communication.

De nombreuses solutions ont été développées pour faire face à la restriction de la taille des entrées, mais l'optimisation des coûts de communication entre les phases map et reduce a rarement été observée. Ainsi, un schéma map qui pourrait être utile pour minimiser les coûts de communication est requis.

Automatisation et configuration

L'automatisation et la configuration aident au déploiement du cluster Hadoop en définissant tous les paramètres. Pour une configuration correcte, le matériel et la quantité de travail sont connus au moment du déploiement.

Lors de la configuration, une petite erreur peut être une raison pour l'exécution inefficace du travail entraîne une dégradation des performances. Pour surmonter ces types de problèmes, nous devons développer une nouvelle technique et des algorithmes qui effectuent le calcul de telle manière que l'établissement puisse être effectué de manière efficace.

Combinaison de données optimisées

Dans MapReduce, les performances globales du système diminuent car l'entrée /sortie intensive du disque augmente le temps d'exécution pendant la phase de combinaison. Donc, réduire le temps d'exécution pendant la phase de combinaison est plus difficile.

Optimisation des performances

Il y a plusieurs facteurs tels que l'ordonnancement, le temps d'initialisation des tâches et la dégradation de la performance de la surveillance dans divers travaux, mais ils ne prennent pas en charge les fonctions telles que le chevauchement des phases MapReduce.

Pour améliorer Map and Reduce, certains aspects nécessitent une optimisation supplémentaire, comme la création d'index, l'exécution rapide de requêtes et la réutilisation des résultats calculés précédemment.

Équité

Équité se réfère à l'équité des algorithmes d'ordonnancement sur la répartition qui table des algorithmes divisant les ressources entre les utilisateurs. Une mesure équitable des ressources est nécessaire parmi les utilisateurs sans famine. Le cluster entier est utilisé par MapReduce avec une lourde charge de travail car ce travail léger n'obtient pas le temps de réponse souhaité.

L'équité traite de la dépendance et de la localité entre les phases Map et Reduce. S'il y a une part égale de travail de mappage et de réduction et que la distribution des fichiers d'entrée a été effectuée en clusters, la dégradation des performances du temps de réponse et du débit doit être effectuée.

Localité de données

La distance entre le nœud de tâche et le nœud d'entrée est appelée localité. Le taux de transfert de données dépend de la localité. Le temps de transfert de données sera court si le nœud de calcul est proche du nœud d'entrée.

La plupart du temps, il n'est pas possible d'acquérir la localité de nœud, dans une telle situation, nous acquérons la localité de rack en effectuant la tâche au même rack. Pour prendre en charge les périphériques matériels, un cadre métallique est utilisé, appelé Rack.

Synchronisation

La synchronisation est un facteur de signification dans MapReduce.

Le processus de transfert des données intermédiaires de sortie du processus de mappage en tant que données d'entrée pour le processus reduce est connu sous le nom de synchronisation. Tandis que le processus reduce est démarré après l'achèvement du processus Map, si un processus Map est retardé, les performances de l'ensemble du processus vont ralentir.

C'est un problème commun qui émerge généralement dans un environnement hétérogène car chaque nœud a une bande passante, un matériel et une infrastructure de calcul uniques, ce qui conduit à une réduction des performances du cluster Hadoop (1).

1.8 Conclusion

Nous avons présenté dans ce chapitre les différents composants d'Hadoop, principalement, HDFS et le modèle de programmation MapReduce. Nous nous avons étalé sur les aspects théoriques pour aborder les approches et techniques pour l'ordonnancement des tâches dans le chapitre suivant.

Chapitre 02

Docker : Une Plateforme pour créer et déployer facilement
les applications Distribuées

2.1 Introduction

Docker a explosé sur la scène ces derniers temps, devenant rapidement la norme de facto pour les développeurs et les administrateurs système pour l'emballage, le déploiement et l'exécution d'applications.

Docker simplifie considérablement le flux de travail des opérations de développement (DevOps) en permettant une plus grande automatisation pour promouvoir la cohérence et la fiabilité.

Dans ce chapitre, nous décrivons l'architecture de base de docker, son fonctionnement et la différence avec la virtualisation.

2.2 Définition docker

Docker est une plateforme qui va vous permettre d'exécuter votre code à l'intérieur d'un conteneur indépendamment de la machine sur laquelle vous êtes ! Un conteneur ressemble à une machine virtuelle sauf qu'il n'embarque pas tout un système d'exploitation avec lui ce qui lui permet de s'exécuter en quelques secondes et d'être beaucoup plus léger.

Docker peut donc résoudre notre problème d'environnement, car quelle que soit la machine que nous utiliserons, le code s'exécutera de la même manière.

La plateforme Docker est composée de deux éléments :

Le démon Docker : qui s'exécute en arrière-plan et qui s'occupe de gérer vos conteneurs.

Le client Docker : qui vous permet d'interagir avec le démon par l'intermédiaire d'un outil en ligne de commande.

Avec la technologie Docker, vous pouvez traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs vous offrent une grande flexibilité : vous pouvez les créer, déployer, copier et déplacer d'un environnement à un autre, ce qui vous permet d'optimiser vos applications pour le cloud (7) .

2.2.1 Les conteneurs Docker

Docker est l'incarnation la plus moderne de la conteneurisation. Il s'agit d'un projet ouvert développé par la société *Dot cloud* (renommée par la suite Docker) sur la base de *LXC*. Fondamentalement, Docker étend *LXC* avec le *Docker daemon*. Ce dernier est une *API (Application Programming Interface)* installée entre le noyau du système d'exploitation et les *VEs*, qui, ensemble, exécutent des processus isolés : processeur, mémoire, entrées/ sorties (E/S), réseau, etc (8).

Depuis la version 0.9 en 2014, Docker n'utilise plus *LXC* comme environnement d'exécution par défaut. Il l'a remplacé par sa propre bibliothèque *Lib container*, implémentée en Go, un langage de programmation orientée service. Docker est un écosystème complexe mais intuitif pour le développement des conteneurs (8).

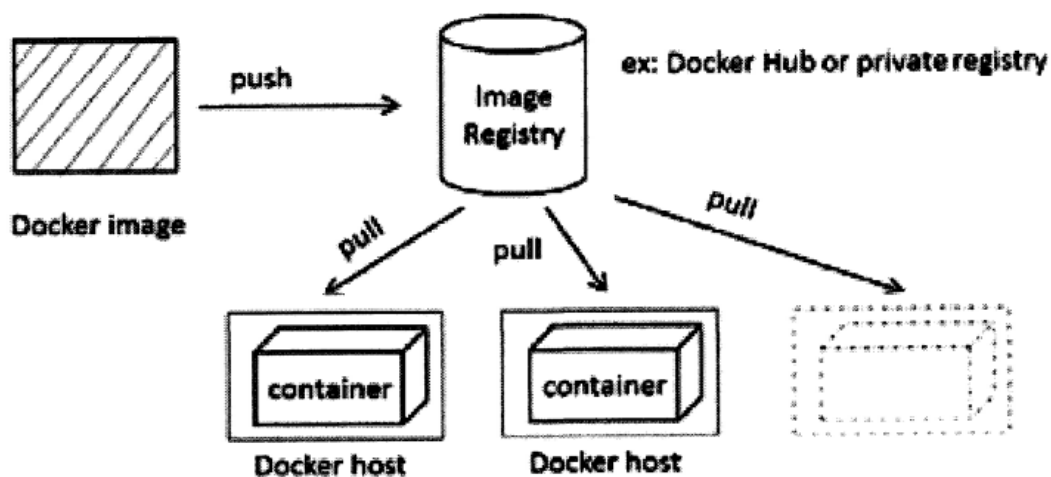


Figure 8: Architecture de base de Docker(8).

Il est riche en fonctionnalités, notamment son système d'image, ses registres et son interface de lignes de commande. La figure 8 présente l'architecture de base de Docker (8).

2.2.2 docker Fonctionnement

La technologie Docker utilise le noyau Linux et des fonctions de ce noyau, telles que les groupes de contrôle *cgroups* et les espaces de noms, pour séparer les processus afin qu'ils puissent s'exécuter de façon indépendante. Cette indépendance reflète l'objectif des conteneurs : exécuter plusieurs processus et applications séparément les uns des autres afin d'optimiser l'utilisation de votre infrastructure tout en bénéficiant du même niveau de sécurité que celui des systèmes distincts.

Les outils de conteneurs, y compris Docker, sont associés à un modèle de déploiement basé sur une image. Il est ainsi plus simple de partager une application ou

un ensemble de services, avec toutes leurs dépendances, entre plusieurs environnements. Docker permet aussi d'automatiser le déploiement des applications (ou d'ensembles de processus combinés qui forment une application) au sein d'un environnement de conteneurs. Ces outils conçus sur des conteneurs Linux (d'où leur convivialité et leur singularité) offrent aux utilisateurs un accès sans précédent aux applications, la capacité d'accélérer le déploiement, ainsi qu'un contrôle des versions et de l'attribution des versions (9).

2.4.3 Les avantages des conteneurs Docker

Modularité

L'approche de Docker en matière de conteneurisation repose sur la décomposition des applications : c'est-à-dire la capacité de réparer ou de mettre à jour une partie d'une application sans devoir désactiver l'ensemble de cette dernière. En plus de cette approche basée sur les micro services, Docker vous permet de partager des processus entre différentes applications quasiment comme vous le feriez avec une architecture orientée services (SOA).

Couches et contrôle des versions d'image

Chaque fichier image Docker est composé d'une série de couches. Ces couches sont assemblées dans une image unique. Chaque modification de l'image engendre la création d'une couche. Chaque fois qu'un utilisateur exécute une commande, comme *run* ou *copy*, une nouvelle couche se crée.

Docker réutilise ces couches pour la construction de nouveaux conteneurs, accélérant ainsi le processus de construction. Les modifications intermédiaires sont partagées entre les images, ce qui optimise la vitesse, la taille et l'efficacité. Qui dit superposition de couches, dit contrôle des versions. À chaque changement, un journal des modifications est mis à jour afin de vous offrir un contrôle total des images de votre conteneur.

Restauration

La fonction la plus intéressante de la superposition de couches est sans doute la restauration. Chaque image est composée de couches. Aussi, si l'itération actuelle d'une image ne vous convient pas, vous pouvez restaurer la version précédente. Cette fonction favorise le développement agile et vous aide à mettre en œuvre les pratiques d'intégration et de déploiement continu (CI/CD) au niveau des outils.

Déploiement rapide

Avant, il fallait plusieurs jours pour mettre en place du nouveau matériel, le faire fonctionner, l'approvisionner et le rendre disponible. C'était un processus complexe et fastidieux. Aujourd'hui, avec les conteneurs Docker, vous pouvez effectuer tout cela en quelques secondes seulement. En créant un conteneur pour chaque processus, vous pouvez

rapidement partager les processus similaires avec les nouvelles applications. De plus, comme vous n'avez pas besoin de redémarrer le système d'exploitation pour ajouter ou déplacer un conteneur, le délai de déploiement s'en trouve encore réduit. Et ce n'est pas tout. La vitesse du déploiement est telle que vous pouvez vous permettre de créer et de détruire facilement et à moindre coût les données de vos conteneurs, sans aucun problème.

Pour résumer, la technologie Docker propose une approche plus granulaire, contrôlable et basée sur des micro services, qui place l'efficacité au cœur de ses objectifs (9).

2.2.4 Avantages de Docker

Légèreté :

Contrairement à la virtualisation classique, Docker exploite et partage le kernel du système d'exploitation de l'hôte, ce qui le rend très efficace en termes d'utilisation des ressources du système

Portabilité et multicloud :

Il est possible de créer, déployer et démarrer des conteneurs sur n'importe quel ordinateur ou serveur distant.

En effet, contrairement aux machines virtuelles, les conteneurs Docker n'ont pas d'OS séparé, ils se reposent sur l'OS de la machine physique. Ainsi, un conteneur peut être porté sur un autre OS et peut démarrer immédiatement.

Par ailleurs, les conteneurs peuvent s'exécuter dans pratiquement n'importe quel environnement (cloud public, privé ou hybride, on premise (serveur local), sous Linux, Windows, Mac, machine virtuelles ou dédiés...), ce qui en facilite le développement et le déploiement.

Performance :

Docker permet une montée en charge facile des applications car il va pouvoir les multiplier sur tout un ensemble d'OS, et fournir de la haute disponibilité.

Les conteneurs permettent de faire beaucoup plus de travail avec moins de matériel informatique, et n'ont pas besoin de machine physique aussi performante que les machines virtuelles.

Vous l'avez maintenant compris, la virtualisation présente de nombreux avantages, que ce soit via la technologie des machines virtuelles ou l'utilisation des conteneurs Docker, la virtualisation vous permet de réaliser un gain de temps et d'argent dans la gestion et l'exécution de vos serveurs et de vos applications.

Si la technologie Docker présente de nombreux avantages, et a été une véritable révolution dans le milieu informatique, dès lors qu'un grand nombre de conteneurs tournent sur une même machine, il devient plus difficile d'en gérer le trafic, l'évolution...

C'est alors qu'entre en jeu les orchestrateurs de conteneurs, qui vont, eux, permettre de faciliter la gestion d'un parc de conteneurs en automatisant la communication entre ceux-ci et répartissant des charges sur les machines. On entend d'ailleurs beaucoup parler de Kubernetes qui, à l'instar de Docker, domine lui aussi son marché (10).

2.2.5 les limites à l'utilisation de Docker

Docker est une technologie très efficace pour la gestion de conteneurs uniques. Cependant, à mesure qu'augmente le nombre de conteneurs et d'applications conteneurisées (tous décomposés en centaines de composants), la gestion et l'orchestration se complexifient. Au final, vous devez prendre du recul et regrouper plusieurs conteneurs pour assurer la distribution des services (réseau, sécurité, télémétrie, etc.) vers tous vos conteneurs. C'est précisément à ce niveau qu'intervient la technologie Kubernetes.

Avec Docker, vous ne profitez pas des mêmes fonctionnalités de type UNIX qu'offrent les conteneurs Linux traditionnels, c'est-à-dire que vous ne pouvez notamment pas utiliser de processus tels que cron ou syslog au sein du conteneur, en parallèle de votre application. Il existe aussi des limites au niveau du nettoyage des processus petits-enfants après l'arrêt des processus enfants, alors que ce nettoyage était effectué par les conteneurs Linux traditionnels. Vous pouvez contourner ces problèmes en modifiant le fichier de configuration et en configurant ces fonctions dès le début, même si ce n'est pas immédiatement évident.

De plus, il existe d'autres sous-systèmes et périphériques Linux qui n'appartiennent pas à un espace de noms, notamment les périphériques `_SELinux`, `cgroups` et `/dev/sd*`. Cela signifie que si un pirate prenait le contrôle de ces sous-systèmes, l'hôte serait compromis. Afin de préserver sa légèreté, le noyau de l'hôte est partagé avec les conteneurs, ce qui ouvre une brèche de sécurité. Ce n'est pas le cas avec les machines virtuelles, car elles sont bien mieux isolées du système hôte.

Le démon Docker peut également poser des problèmes de sécurité. Pour utiliser et exécuter des conteneurs Docker, il est conseillé d'utiliser le démon Docker, qui s'exécute en permanence pour la gestion des conteneurs. Le démon Docker nécessite des privilèges root, il est donc important de surveiller l'accès à ces processus, ainsi qu'à leur emplacement. Par exemple, un démon en local est plus difficile à attaquer qu'un démon qui réside dans un emplacement public, comme un serveur Web (9).

2.3 Un conteneur Linux

Un conteneur Linux est un processus ou un ensemble de processus isolés du reste du système. Tous les fichiers nécessaires à leur exécution sont fournis par une image distincte, ce qui signifie que les conteneurs Linux sont portables et fonctionnent de la même manière dans les environnements de développement, de test et de production. Ainsi, ils sont bien plus rapides à utiliser que les pipelines de développement qui s'appuient sur la réplication d'environnements de test traditionnels. En raison de leur popularité et de leur facilité d'utilisation, les conteneurs constituent également un élément essentiel de la sécurité informatique(9).

2.3.1 projet Linux Containers (LXC)

Projet Linux Containers (LXC) est une plateforme de conteneurs Open Source qui fournit un ensemble d'outils, de modèles, de bibliothèques et de liaisons de langage. Le projet LXC dispose d'une interface en ligne de commande simple qui facilite la prise en main pour les nouveaux utilisateurs.

Il contient un environnement de virtualisation au niveau du système d'exploitation qu'il est possible d'installer sur de nombreux systèmes basés sur Linux. Vous pouvez peut-être y accéder dans le référentiel du package de votre distribution Linux (9).

2.3.2 LXC fonctionnement

La manière la plus simple et la plus judicieuse d'utiliser Linux Container est de connecter chaque conteneur avec un processus. Cela permet d'obtenir un contrôle complet. Pour chaque processus, les espaces de noms qui mettent des ressources à la disposition d'un ou plusieurs processus qui utilisent le même espace de noms sont particulièrement importants.

En outre, les processus servent de contrôle d'accès pour sécuriser les conteneurs. Pour utiliser un environnement LXC, les caractéristiques et leurs fonctions doivent être claires. Les cgroups (groupes de contrôle du noyau) limitent et isolent les ressources des processus telles que l'unité centrale, les E/S, la mémoire et les ressources réseau. En outre, le contenu d'un groupe de contrôle peut être administré, surveillé, priorisé et modifié.



Figure 9 : Traditionnel Linux conteneurs.

Les fonctions sont claires et la plateforme LXC est donc très conviviale, ce qui est un avantage non négligeable notamment pour les débutants (11).

2.4 Traditionnel linux conteneurs vs docker

À l'origine, la technologie Docker a été créée sur la base de la technologie LXC, que la plupart des utilisateurs associent aux conteneurs Linux « traditionnels », mais elle s'est depuis émancipée. LXC était un outil de virtualisation léger très utile, mais il n'offrait pas une expérience à la hauteur pour les utilisateurs ou les développeurs. La technologie Docker permet non seulement d'exécuter des conteneurs, mais aussi de simplifier leur conception et leur fabrication, l'envoi d'images, le contrôle des versions d'image, etc.

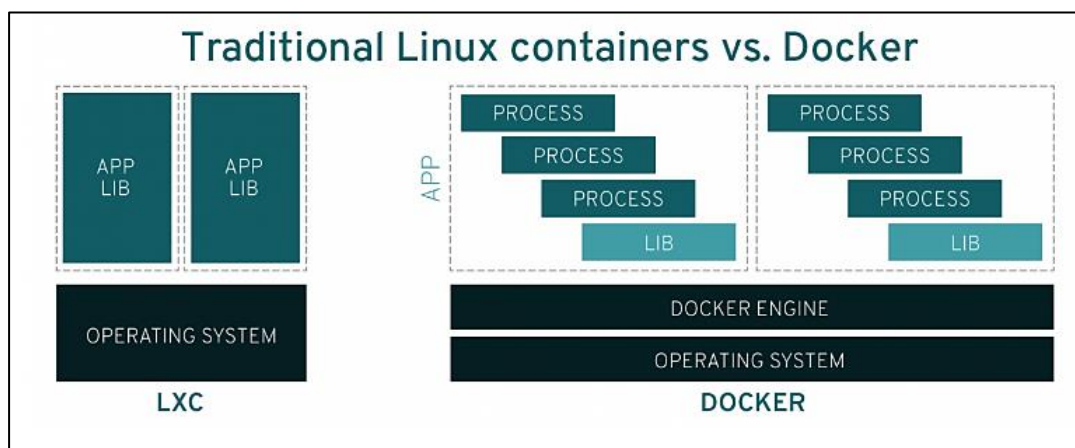


Figure 10 : Traditionnel Linux conteneurs vs Docker(9).

Les conteneurs Linux traditionnels utilisent un système init capable de gérer plusieurs processus. Ainsi, des applications entières peuvent s'exécuter comme un bloc. La technologie Docker encourage la décomposition des applications en processus distincts et fournit les outils nécessaires pour y parvenir. Cette approche granulaire présente bien des avantages (9).

2.5 La virtualisation

La mise en place de l'isolation a rendue possible effectuée grâce à l'introduction des techniques de virtualisation dans la création du *cloud*. Toutefois, il existe plusieurs manières de mettre en œuvre l'isolation :

- La première façon consiste à allouer la ressource matérielle entière à une entreprise même si celle-ci ne souscrit qu'à une fraction de cette ressource.
- Cette méthode ne permet qu'une résolution partielle des problèmes d'isolation. En effet, certaines ressources comme le réseau et sa bande passantedemeurent partagées entre les entreprises dans le *cloud*. À moins que le fournisseur alloue exclusivement à chaque entreprise des équipements et une bande passante dédiée (ce qui n'est pas raisonnable), il

serait impossible avec cette méthode d'éviter des situations de monopolisation des ressources par une entreprise. Cette solution matérielle doit être complétée par une solution logicielle.

- La seconde méthode consiste à laisser la responsabilité aux entreprises d'implanter les mécanismes d'isolation. Cette méthode n'est pas envisageable dans la mesure où le *cloud* ne dispose d'aucun moyen d'introspection des applications des entreprises afin de s'assurer de l'implantation de ces mécanismes....,
- La dernière méthode est hybride (matérielle et logicielle). Tout en implémentant les mécanismes d'isolation, elle donne l'illusion aux clients d'avoir un accès direct et exclusif à la ressource matérielle. Parallèlement, elle garantit au *cloud* un accès contrôlé des clients aux ressources matérielles. Il s'agit de l'isolation par virtualisation (8).

2.5.1 Définition de la virtualisation

La virtualisation consiste à créer une version virtuelle d'un dispositif ou d'une ressource, comme un système d'exploitation, un serveur, un dispositif de stockage ou une ressource réseau. Elle peut donc être considérée comme l'abstraction physique des ressources informatiques. En d'autres termes, les ressources physiques allouées à une machine virtuelle sont abstraites à partir de leurs équivalentes physiques. Chaque dispositif virtuel, qu'il s'agisse d'un disque, d'une interface réseau, d'un réseau local, d'un commutateur, d'un processeur ou d'une mémoire, correspond à une ressource physique sur un système informatique.

Les machines virtuelles hébergées par la machine hôte sont donc perçues par ce dernier comme des applications auxquelles il est nécessaire de dédier ou distribuer ses ressources. Cependant, il existe plusieurs types de virtualisation .

Chaque type correspond à un cas d'utilisation spécifique, comme cela sera présenté dans la section suivante (8).

2.5.2 Types de virtualisation

Il existe de nombreux domaines d'application à la virtualisation, s'agissant généralement de la virtualisation du stockage, du réseau, ou du serveur.

- **La virtualisation du stockage.** Elle est obtenue en isolant les données de leur localisation physique. Elle permet de masquer les spécificités physiques des unités de stockage. Vues de l'extérieur, les unités de stockage sont perçues comme étant un volume unique. La virtualisation du stockage est couramment utilisée dans un réseau de stockage (*Storage Area Network-SAN, Network Attached Storage-NAS*).

- **La virtualisation du réseau.** Elle consiste à combiner des ressources réseaux physiques (commutateurs, routeurs, etc.) et logiques (bande passante, adressage, etc.) pour créer des réseaux virtuels, découplés du matériel réseau sous-jacent. L'objectif est de garantir

aux systèmes et aux utilisateurs un partage efficace et sécurisé des ressources physiques du réseau.

- **La virtualisation du serveur** : Ce type de virtualisation se définit comme l'ensemble des techniques matérielles ou logiciels qui permettent de faire fonctionner plusieurs systèmes, serveurs ou applications sur une seule machine en donnant l'illusion qu'ils fonctionnent sur différentes machines.

Dans la suite de ce mémoire, nous nous intéressons à la virtualisation du serveur. Plus précisément, nous étudions ci-après les deux techniques de virtualisation du serveur, à savoir la virtualisation système et la virtualisation applicative (8).

2.5.3 La virtualisation système

Cette technique de virtualisation du serveur fait référence à l'utilisation d'un hyperviseur pour permettre à la machine hôte d'exécuter plusieurs instances virtuelles en même temps. Ces dernières sont communément appelées des *VMs* (Virtual Machines), tandis que l'hyperviseur est connu sous le nom de *VMM* (Virtual Machines Monitor), c'est-à-dire gestionnaire de *VM*. Ces nouvelles notions sont définies comme suit :

1. Machine virtuelle (VM) : il s'agit d'une version virtuelle d'un matériel dédié avec son propre système d'exploitation (*Operating System- OS*). Comme dans une machine réelle, chaque *VM* conserve un fonctionnement normal, défini comme suit : une *VM* gère les accès mémoire (*Random Access Memory RAM*), disque, réseau, processeurs (*Central Processing Unit- CPU*) et autres périphériques de ses processus.

2. Hyperviseur (VMM) : il implante les mécanismes d'isolation et de partage des ressources matérielles. Il est capable de démarrer simultanément plusieurs machines virtuelles de différents types (Linux, Mac ou Windows) sur le même matériel. Quoique les *VM* gèrent les accès aux ressources, aucun accès aux ressources n'est possible sans l'aval du *VMM*. Ce dernier décide notamment des attributions de temps processeurs aux machines virtuelles.

Quant à la communication avec l'extérieur, l'hyperviseur peut fournir plusieurs techniques pour rendre accessible ou non les *VM*. Il peut réaliser cette tâche par l'assignation d'adresses *IP* (*Internet Protocol*) et par implantation de mécanismes d'accès réseau aux *VM* (par routage, filtrage de communication, etc.) (8).

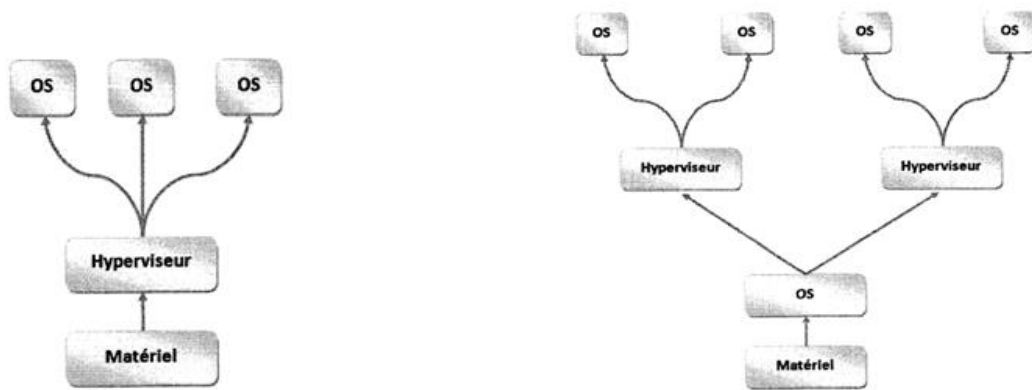


Figure 11 : Hyperviseur de type 1(8)..**Figure 12 :**Hyperviseur de type 2(8)..

On peut distinguer deux types d'hyperviseur :

- **Hyperviseur de type 1 (natif)** : Ils'installe directement sur la couche physique du matériel du serveur. Ainsi, au démarrage de ce dernier, ce VMM prend le contrôle exclusif du matériel. Hyper-V, Xen, VMware, ESXi, KVM sont des hyperviseurs de type 1.
 - **Hyperviseur de type 2 (hébergé)** : Comparable à un émulateur, il s'installe sur l'OS installé sur le matériel du serveur. Il démarre comme un logiciel installé sur le système d'exploitation. Comme exemples, on peut citer *Oracle VM (VirtualBox)*, *Workstation*, *Parallelset Fusion*.
- Les figures 9 et 10 présentent les deux types d'hyperviseur de la virtualisation système (8).

2.5.4 La virtualisation applicative

Contrairement à la virtualisation système, le principe de cette technique est de mettre en œuvre une couche de virtualisation sous forme d'une application qui elle-même crée des instances virtuelles et les isole des spécificités de la machine hôte, c'est-à-dire de celles de son architecture ou de son système d'exploitation. Ceci permet au concepteur de l'application de ne pas avoir à rédiger plusieurs versions de son logiciel pour qu'il soit exécutable dans tous les environnements. Les machines virtuelles applicatives les plus connues sont la JVM (Java Virtual Machine) et les conteneurs.

- **Java Virtual Machine (JVM)** : Elle est gratuite, propriétaire jusqu'à la version 6 (stable) et libre sous la GPL (General Public License) à partir de la version 7. La JVM permet aux applications Java compilées de produire les mêmes résultats quelle que soit la plateforme, tant que celle-ci est pourvue de la machine virtuelle Java adéquate.
- **Conteneurs** : Mieux que la JVM, les conteneurs fonctionnent avec toutes les technologies compatibles avec Linux (par exemple Java, Python, Ruby, Nodejs et Go). Les conteneurs sont isolés les uns des autres et peuvent optimiser l'utilisation des ressources (CPU, RAM, stockage) de la machine hôte (8).

2.5.4.1 La virtualisation par conteneurs

Définition

La virtualisation par conteneurs (également connue sous le nom de conteneurisation) constitue une alternative à la virtualisation système. Il s'agit d'une approche qui s'appuie directement sur les fonctionnalités du noyau (*Kernel*) pour créer des environnements virtuels (*VirtualEnvironment- VE*) isolés les uns des autres. Ces *VE* sont appelés conteneurs, tandis que les fonctionnalités fournies par le noyau du système d'exploitation sont les groupes de contrôles (*cgroups*) et les espaces de noms (*namespaces*). Les *namespaces* permettent de contrôler et de limiter la quantité de ressources utilisée pour un processus, tandis que les *cgroups* gèrent les ressources d'un groupe de processus.

Un conteneur fournit donc les ressources nécessaires pour exécuter des applications comme si ces dernières étaient les seuls processus en cours d'exécution dans le système d'exploitation de la machine hôte (8).

2.5.5 Vue d'ensemble des conteneurs

Les conteneurs constituent une solution pour la gestion des ressources entre les applications. Le terme est dérivé des conteneurs d'expédition, une méthode standard pour stocker et expédier tout type de cargaison.

Le concept de base des conteneurs est né en 1979 avec la fonction Unix *chroot*. Cette dernière permet de créer plusieurs sessions isolées sur une même machine, de façon à partager parallèlement les ressources de processus et de stockage de la machine. Vu cette caractéristique particulière de la fonction *chroot*, celle-ci peut être considérée comme étant le précurseur de la conteneurisation.

Introduite en l'année 2000, la solution *FreeBSD Jail* a permis de partitionner le système d'exploitation de la machine hôte en des sous-systèmes isolés. Par rapport à *chroot*, la fonctionnalité supplémentaire de l'environnement *FreeBSD* était la possibilité d'isoler également le système de fichiers, les utilisateurs et le réseau, en les rendant plus sécurisés.

Mais cette solution présente des difficultés en termes de mise en œuvre. L'année suivante a vu le développement d'une solution semblable à *FreeBSD Jail* appelée *Linux VServer*. Cette dernière utilise un système *VPS* (*Virtual Private Server*) pour partitionner les ressources d'une machine en des contextes de sécurité. Mettant les bases d'une utilisation optimale des fonctionnalités du noyau du système d'exploitation, le projet *VServer* conduira, en 2008, au projet *Linux Container (LXC)*. Dans ce nouveau projet, chaque distribution Linux importe ses propres bibliothèques qui supportent les conteneurs *LXC*, mais qui ne sont pas compatibles entre elles. Cependant, c'est à partir de 2013 que Docker a rendu simple le

développement et le déploiement des conteneurs tout en les standardisant. Docker résout un problème de longue date dans le *cloud computing*: la portabilité des applications.

Vu l'importance de cette propriété de portabilité dans le contexte du *cloud computing*, nous nous concentrons dans la suite sur la technologie Docker (8).

2.5.6. Conteneurs vs machines virtuelles

Les conteneurs ne remplacent pas nécessairement les machines virtuelles, les deux technologies ont leurs propres forces et faiblesses et doivent être utilisées le cas échéant.

Voici quelques différenciateurs clés entre les deux :

Scalabilité

Les conteneurs vous permettent d'optimiser l'efficacité de vos serveurs existants. Ils sont très légers et donc très évolutifs. Les machines virtuelles, d'autre part, entraînent des frais généraux supplémentaires pour le système d'exploitation.

Les conteneurs offrent une meilleure portabilité que les machines virtuelles. Il existe une variété de formats d'images de machines virtuelles disponibles. Malheureusement, la compatibilité des formats varie entre les produits concurrents et, par conséquent, la compatibilité croisée des images est limitée.

Productivité

Les conteneurs sont généralement beaucoup plus simples à configurer et à entretenir. Chaque machine virtuelle nécessite l'installation et la maintenance de sa propre instance de système d'exploitation (y compris les pilotes de prise en charge), ce qui entraîne une complexité supplémentaire.

Security

Les machines virtuelles offrent un degré d'isolation plus élevé entre la machine virtuelle et le système d'exploitation hôte. Les machines virtuelles tirent parti de l'isolation matérielle à l'aide du processeur VT d'Intel et des extensions matérielles de virtualisation –V d'AMD. Malheureusement, le matériel n'a pas encore rattrapé les solutions de virtualisation basées sur des conteneurs - du moins au moment de la rédaction. Pour cette raison, les machines virtuelles sont considérées comme plus sécurisées, et donc privilégiées dans les environnements aux exigences de sécurité élevées.

Performance

Selon Docker, une application s'exécutant dans un conteneur fonctionnera généralement deux fois plus vite qu'une application s'exécutant dans une machine virtuelle. Les conteneurs surpassent les machines virtuelles car ils sont considérablement plus légers

que les machines virtuelles. Ceci est plus évident dans les temps de démarrage; les machines virtuelles doivent démarrer un système d'exploitation complet.

Les conteneurs utilisent mieux les ressources système que les machines virtuelles.

Les machines virtuelles consomment généralement plus de mémoire en raison des frais généraux supplémentaires, en particulier le système d'exploitation.

Flexibilité

Les machines virtuelles offrent un degré élevé de flexibilité en permettant de contrôler la plate-forme du système d'exploitation sous-jacente. Si vous utilisez des conteneurs, votre choix de système d'exploitation sera lié à la plate-forme sous-jacente de votre machine hôte. Par conséquent, vous ne pouvez pas exécuter un conteneur Windows sur un système d'exploitation hôte basé sur Linux.

Les conteneurs Docker sont hautement portables et peuvent être transférés entre les machines sans impact sur les applications qui s'exécutent à l'intérieur (12) .

Ce n'est pas tout à fait la même chose. Les conteneurs et la virtualisation sont plutôt complémentaires. Voici deux définitions simples pour y voir plus clair :

- La virtualisation permet à vos systèmes d'exploitation (Windows ou Linux) de s'exécuter simultanément sur un seul système matériel.
- Les conteneurs partagent le même noyau de système d'exploitation et isolent les processus de l'application du reste du système. Par exemple : les systèmes Linux ARM exécutent des conteneurs Linux ARM, les systèmes Linux x86 exécutent des conteneurs Linux x86 et les systèmes Windows x86 exécutent des conteneurs Windows x86. Les conteneurs Linux sont extrêmement portables, mais ils doivent être compatibles avec le système sous-jacent.

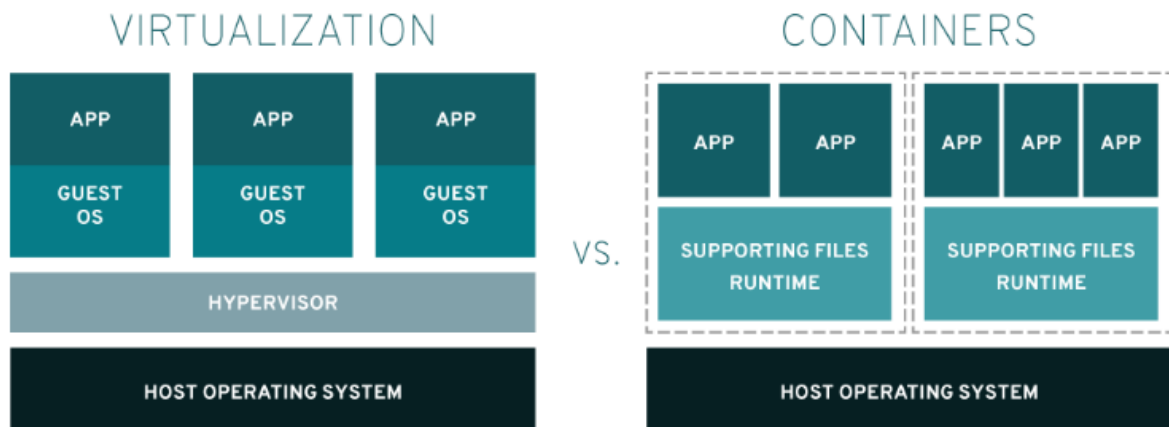


Figure 13: Virtualisation vs Conteneurs(9).

En premier lieu, la virtualisation utilise un hyperviseur pour émuler le matériel, ce qui permet d'exécuter plusieurs systèmes d'exploitation en parallèle. C'est une solution moins légère que les conteneurs. Lorsque vous disposez de ressources limitées, aux fonctionnalités limitées, il est nécessaire que vos applications soient légères et puissent être déployées de manière dense. Les conteneurs Linux s'exécutent en natif sur leur système d'exploitation, qu'ils partagent entre eux. Vos applications et services restent ainsi légers et s'exécutent rapidement en parallèle.

Les conteneurs Linux représentent une nouvelle évolution de la manière selon laquelle nous développons, déployons et gérons des applications. Les images de conteneurs Linux permettent d'assurer la portabilité et le contrôle des versions des applications. Les développeurs ont ainsi la garantie que ce qui fonctionne sur leur ordinateur portable fonctionnera aussi dans l'environnement de production. Un conteneur Linux mobilise moins de ressources qu'une machine virtuelle. Il propose une interface standard (démarrage, arrêt, variables d'environnement, etc.), assure l'isolation des applications et peut être géré plus facilement en tant que module d'une application plus importante (plusieurs conteneurs). En outre, ces applications à plusieurs conteneurs peuvent être orchestrées dans différents clouds. Il existe même des outils qui associent l'orchestration de conteneurs et la gestion de machines virtuelles (9).

2.7 Conclusion

Dans ce chapitre, nous avons présenté les notions de base associées à la conteneurisation. Plus précisément, nous avons défini le domaine d'application de la conteneurisation, qui n'est autre que le cloud computing.

L'étude comparative entre l'utilisation des machines virtuelles et les conteneurs Docker montre que ces derniers ont un avantage inhérent sur les VMs en raison de l'amélioration de plusieurs métriques de performances (ex : CPU, RAM, stockage, temps de démarrage, etc.).

En général, Docker permet de créer et de déployer des logiciels (services) sous forme de conteneurs dans une machine physique ou virtuelle grâce à une collection d'outils gratuits.

Toutefois, il n'est pas adapté au déploiement de plusieurs conteneurs sur plusieurs machines distribuées (système distribué). Des plateformes d'orchestration de conteneurs ont été créées afin de remédier à cette limitation.

Chapitre 03

Les plateformes matérielles et logicielles utilisées

3.1 Introduction

Pour la réalisation de notre projet on utilise l'exécutif temps réel FreeRTOS avec les carte ARDUINO, pour exploiter la simplicité d'utilisation et la licence libre, ces plateformes ont été développées par des professionnels, strictement contrôlé de qualité, robuste, pris en charge, ne contient aucune ambiguïté de propriété intellectuelle et est vraiment gratuit à utiliser dans des applications commerciales sans aucune obligation d'exposer votre code source propriétaire.

3.2 L'exécutif temps réel FreeRTOS

FreeRTOS est un système d'exploitation en temps réel pour microcontrôleurs développé en partenariat avec les principaux fabricants de puces électronique au monde sur une période de 15 ans, et maintenant téléchargé toutes les 170 secondes, leader du marché pour les microcontrôleurs et les petits microprocesseurs, distribué gratuitement sous la licence open source, FreeRTOS comprend un noyau et un ensemble croissant de bibliothèques adaptées à une utilisation dans tous les secteurs industriels. FreeRTOS est construit en mettant l'accent sur la fiabilité et la facilité d'utilisation (13).

FreeRTOS est un RTOS assez limité par rapport aux grands RTOS commerciaux (tels que Vx Works, Windows CE, QNX, etc.) Nous avons choisi de travailler avec FreeRTOS car il fournit du code source, est relativement facile à comprendre et à utiliser, est gratuit pour un usage commercial avec des problèmes minimes, et largement utilisé, l'inconvénient majeur de son utilisation est le manque de prise en charge de fonctions plus complexes.

FreeRTOS est parfaitement adapté aux applications en temps réel profondément embarquées qui utilisent des microcontrôleurs ou de petits microprocesseurs.

FreeRTOS, comme n'importe quel autre noyau, est un outil purement logiciel, ce n'est qu'un système de fichiers. FreeRTOS nous propose une API de programmation pour gérer un environnement multitâche, vous trouverez ci-dessous le système de fichiers du noyau :

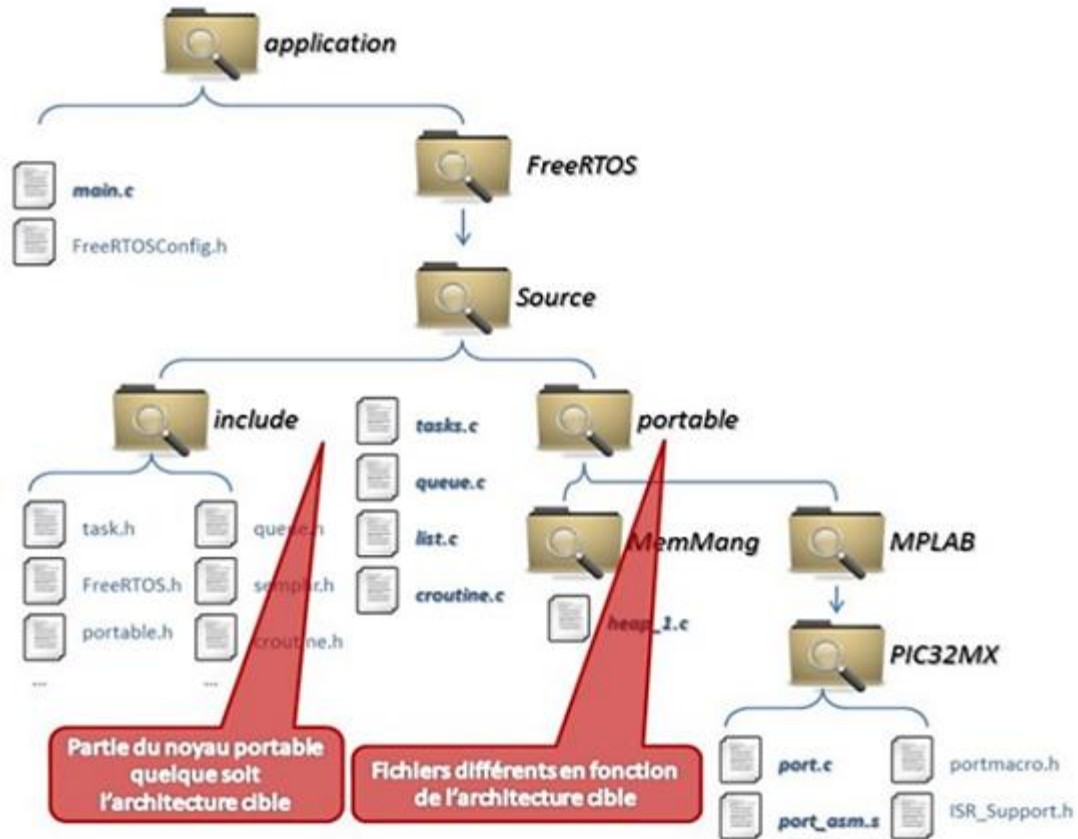


Figure 14 : Le système de fichiers du noyau « FreeRTOS »(13).

En son cœur, FreeRTOS est un ensemble de bibliothèques C et en particulier un planificateur de tâches. À chaque « tick » (réglé à 15 ms sur l'Arduino), le planificateur lance une interruption et considère toutes les tâches « prêtes » à s'exécuter. Il exécute la tâche prête avec priorité la plus élevée. S'il existe une égalité pour la tâche de priorité la plus élevée prête à être exécutée, il utilise la planification à tour de rôle pour basculer entre les tâches de cette priorité. Le planificateur peut également « bloquer » les tâches et les faire supprimer de la liste prête jusqu'à ce qu'un événement se produise (des exemples d'événements incluent un sémaphore en cours de définition et un certain temps qui s'écoule). Il peut également « suspendre » des tâches et les rendre imprévisibles jusqu'à leur reprise explicite. Voir le diagramme ci-dessous pour une représentation visuelle(13).

- **Running**
 - Task is actually executing
- **Ready**
 - Task is ready to execute but a task of equal or higher priority is Running.
- **Blocked**
 - Task is waiting for some event.
 - **Time:** if a task calls `vTaskDelay()` it will block until the delay period has expired.
 - **Resource:** Tasks can also block waiting for queue and semaphore events.
- **Suspended**
 - Much like blocked, but not waiting for anything.
 - Tasks will only enter or exit the suspended state when explicitly commanded to do so through the `vTaskSuspend()` and `xTaskResume()` API calls respectively.

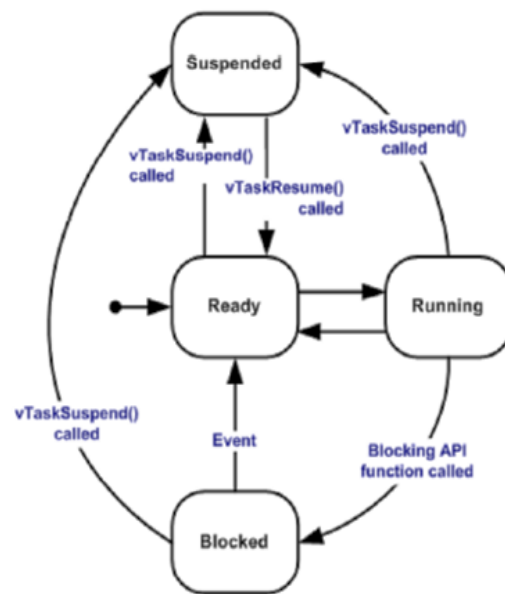


Figure 15: Les états du tâche sur FreeRTOS (13)

3.2.1 Pourquoi FreeRTOS ?

Noyau de confiance : Avec une robustesse éprouvée, un faible encombrement et une large prise en charge des périphériques, le noyau FreeRTOS est considéré par les entreprises de renommée mondiale comme la norme de facto pour les microcontrôleurs et les petits microprocesseurs.

Accélérez la mise sur le marché : Grâce aux démos préconfigurées détaillées et aux intégrations de référence de l'Internet des objets, il n'est pas nécessaire de déterminer comment configurer un projet. Téléchargez, compilez et accédez instantanément au marché plus rapidement.

Prise en charge d'un large écosystème : Notre écosystème de partenaires offre un large éventail d'options, notamment des contributions de la communauté, un support professionnel, ainsi que des outils intégrés d'IDE et de productivité(14).

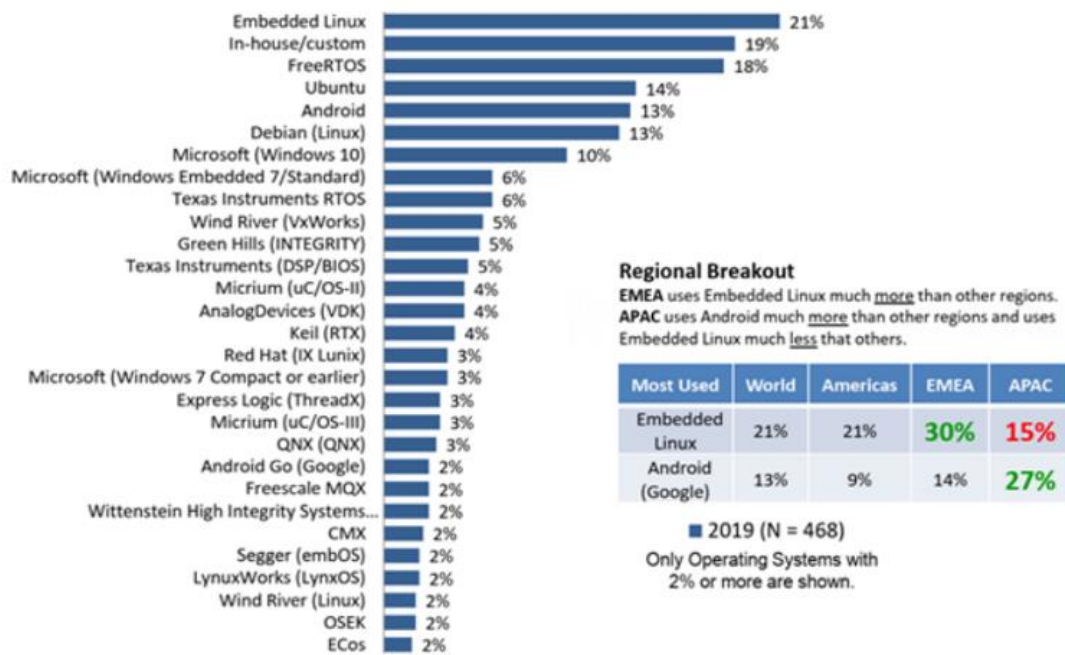


Figure 16: Observation de marché des OS et RTOS pour l'embarqué (14).

Remarque :

EMEA, est une appellation que certaines entreprises utilisent pour désigner la région économique qui regroupe les pays d'Europe, du Moyen-Orient et de l'Afrique.

APAC est un ensemble géographique constitué de l'Extrême-Orient, du sous-continent indien et de l'Océanie.

3.2.2 Fonctionnalités de FreeRTOS :

- Opération préventive ou coopérative
- Affectation des priorités de tâches très flexible
- Mécanisme de notification des tâches flexible, rapide et léger
- Files d'attente
- Sémaphores
- Mutex
- Minuteurs
- Groupes d'événements
- Vérification du dépassement de la pile
- Enregistrement de trace
- Collecte des statistiques d'exécution des tâches
- Licence commerciale et assistance en option
- Modèle d'imbrication d'interruption complète (pour certaines architectures)

- Pile d'interruption gérée par logiciel le cas échéant (cela peut aider à économiser la RAM) (15).

3.2.3 Algorithmes d'ordonnement de FreeRTOS :

Il existe un certain nombre d'algorithmes de planification assez standard qui sont utilisés dans les RTOS, l'algorithme idéal serait capable de : planifier tout ensemble de tâches pour lequel il existe un calendrier, échouer normalement lorsque les tâches ne sont pas planifiables et seraient simples à utiliser (ayant des priorités fixes ou "statiques" pour chaque tâche serait un bon début pour être simple...). Bien sûr, un tel algorithme n'existe pas, nous avons donc un nombre d'algorithmes différents que nous pouvons envisager, chacun avec son propre ensemble d'avantages et d'inconvénients.

FreeRTOS n'utilise que deux algorithmes d'ordonnement :

- « **Round-robin Scheduling** », Dans cet algorithme, toutes les tâches de priorité égale obtiennent le processeur en parts égales de temps processeur.
- « **FixedPriorityPreemptiveScheduling** », cet algorithme sélectionne les tâches en fonction de leur priorité. En d'autres termes, une tâche à haute priorité obtient toujours le processeur avant une tâche à faible priorité. Une tâche de faible priorité s'exécute uniquement lorsqu'il n'y a pas de tâche de haute priorité à l'état prêt.

✎ **Remarque** : Dans notre projet, nous utilisons l'ordonnement en mélangeant les deux algorithmes et il est connu sous le nom « **PrioritizedPreemptiveSchedulingwith Time Slicing** »(16).

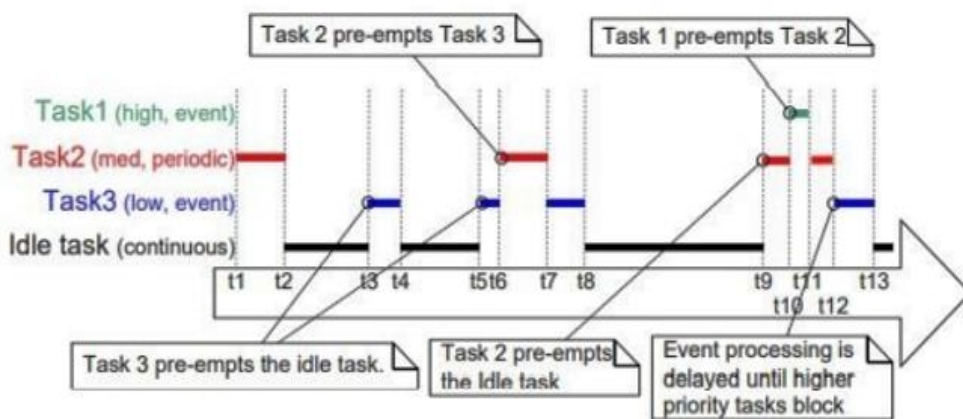


Figure 17: exemple de « **Prioritized Preemptive Scheduling with Time Slicing** »(16)

3.3 Les cartes ARDUINO

Arduino est une plate-forme de prototypage d'objets interactifs à usage créatif constituée d'une carte électronique et d'un environnement de programmation. Sans tout connaître ni tout comprendre de l'électronique, cet environnement matériel et logiciel permet à l'utilisateur de formuler ses projets par l'expérimentation directe avec l'aide de nombreuses ressources disponibles en ligne.

Le matériel : Il s'agit d'une carte électronique basée autour d'un microcontrôleur Atmega du fabricant Atmel, dont le prix est relativement bas pour l'étendue possible des applications(16).



Figure18:« Carte Arduino UNO » (16).

Le logiciel : Le logiciel va nous permettre de programmer la carte Arduino en langage C. Il nous offre une multitude de fonctionnalités.

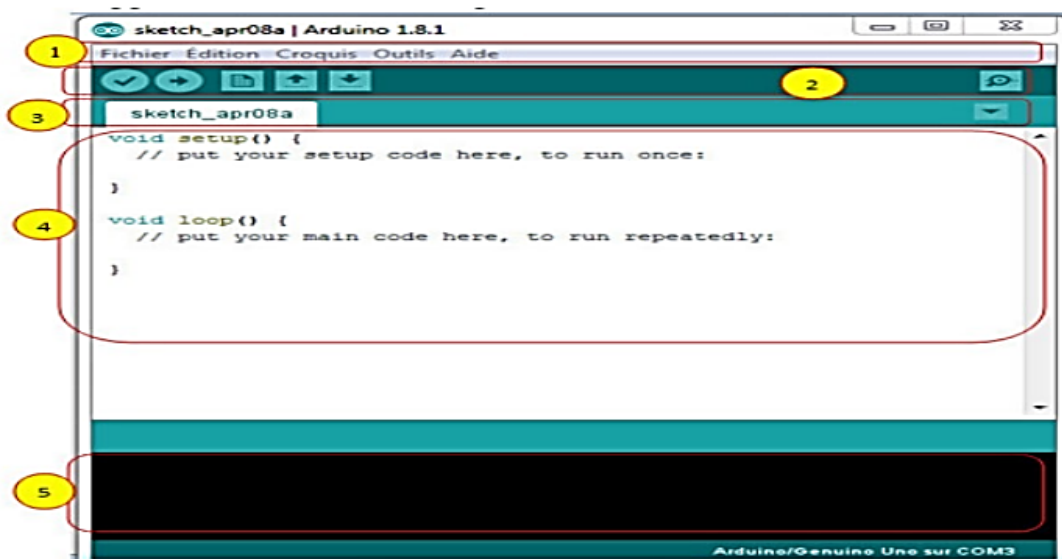


Figure19: « Arduino 1.8.1 IDE» (16).

1. Un menu.
2. Une barre d'actions.
3. Un ou plusieurs onglets correspondant aux sketches.
4. Une fenêtre de programmation.
5. Une console qui affiche les informations et erreurs de compilation et de Téléversement du programme.






	Bouton « Verify » (Vérifier) ; il permet de compiler votre programme et de vérifier si des erreurs s'y trouvent. Cette procédure prend un certain temps d'exécution et lorsque est terminée, elle affiche un message de type « Binary sketch size : ... » indiquant la taille du sketch téléversé.
	Pour transmettre le sketch compilé avec succès sur la carte Arduino dans le microcontrôleur.
	Bouton « New » (Nouveau) ; ce bouton permet de créer un nouveau sketch.
	Bouton « Open » (Ouvrir) ; il fait apparaître un menu qui permet d'ouvrir un sketch qui figure dans votre dossier de travail ou des exemples de sketches intégrés au logiciel.
	Bouton « Save » (Sauvegarder) ; il permet de sauvegarder votre sketch.

Tableau 01 :Raccourcies de barre d'actions (16).

Le système Arduino nous permet de réaliser un grand nombre des systèmes électronique complexes, qui ont des applications dans tous les domaines, nous pouvons donner quelques exemples :

- Contrôler et/ou télécommander les appareils domestiques
- Contrôler et/ou télécommander les machines industrielles
- Communiquer avec des capteurs et / ou un ordinateur
- Contrôler et/ou télécommander des appareils mobiles (modélisme) etc.
- Fabriquer votre propre robot.

Avec Arduino, nous allons faire des systèmes électroniques tels qu'un système de contrôle électronique, un système de lubrification, etc. Tous ces systèmes seront conçus basé sur une carte Arduino, des capteurs et d'autres composants électroniques (16).

3.3.1 Pourquoi choisir Arduino ?

Il existe pourtant dans le commerce, une multitude de plateformes qui permettent de faire la même chose, notamment les microcontrôleurs « PIC » du fabricant Micro chip, nous allons voir pour quoi choisir l'Arduino.

a. Le prix En vue des performances qu'elles offrent, les cartes Arduino sont relativement peu coûteuses, ce qui est un critère majeur pour le débutant.

b. La liberté C'est un bien grand mot, mais elle définit de façon assez concise l'esprit de L'Arduino, elle constitue en elle-même deux choses Le logiciel : gratuit et open source, développé en c, dont la simplicité d'utilisation relève du savoir cliquer sur la souris. Le matériel : cartes électroniques dont les schémas sont en libre circulation sur internet. Cette liberté a une condition : le nom « Arduino » ne doit être employé que pour les cartes «

officielles », en somme, vous ne pouvez pas fabriquer votre propre carte sur le modèle Arduino et lui assigner le nom « Arduino ». Les cartes non officielles, on peut les trouver et les acheter sur Internet et sont pour la quasitotalité compatibles avec les cartes officielles Arduino.

c. La compatibilité Le logiciel, tout comme la carte, est compatible sous les plateformes les plus courantes (Windows, Linux et Mac), contrairement aux autres outils de programmation du commerce qui ne sont, en général, compatibles qu'avec Windows.

d. La communauté La communauté Arduino est impressionnante et le nombre de ressources à son sujet est en constante évolution sur internet. De plus, on trouve les références du langage Arduino ainsi qu'une page complète de tutoriels sur le site arduino.cc.

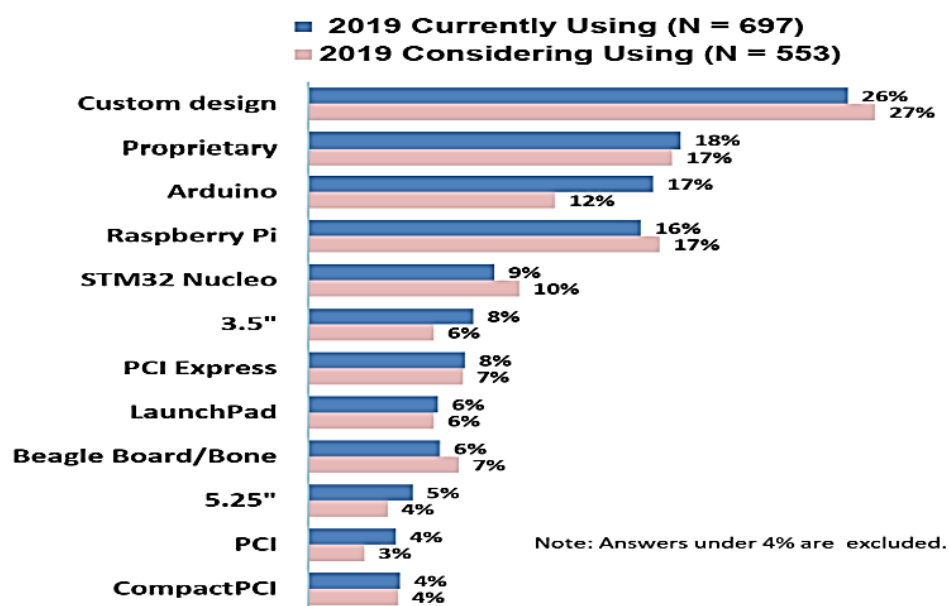


Figure 20: Observation de marché des cartes pour les systèmes embarqués (14)

3.3.2 Types de cartes Il y a trois types de cartes :

- Les cartes dites « officielles » qui sont fabriquées en Italie par le fabricant officiel : Smart Projets.
- Les cartes dites « compatibles » qui n'ait pas fabriqués par Smart Projets, mais qui sont totalement compatibles avec les Arduino officielles.
- Les « autres » fabriquées par diverse entreprise et commercialisées sous un nom différent (Freeduino, Seeduino, Femtoduino, ...).

3.3.3 Différentes cartes Des cartes Arduino

il en existe beaucoup, on représenter les plus connues dans l'image suivante:



Figure 21 : « les différents modèles des carte ARDUINO »(17).

3.4 Conclusion

Comme pont tendu entre le monde réel et le monde numérique, Arduino permet d'étendre les capacités de relations humain/machine ou environnement/machine. Arduino est un projet en source ouverte : la communauté importante d'utilisateurs et de concepteurs permet à chacun de trouver les réponses à ses questions. FreeRTOS est un environnement exécutif à très faible empreinte mémoire quelque KB fournissant un ordonnanceur et les mécanismes associés pour le partage de ressources entre tâches, en utilisant une bibliothèque libre et compatible avec plusieurs architectures.

Chapitre 04

Conception et implémentation

4.1. Introductions

Dans le cadre de Conception et Optimisation d'allocation et libération des tâches temps réel on prendra comme cas d'étude Hadoop.

Hadoop possède les structures de base nécessaires à effectuer des calculs : un système de fichiers, un langage de programmation, une méthode pour distribuer les programmes ainsi générés à un cluster, un mode pour obtenir les résultats et arriver à une consolidation unique de ces derniers, ce qui est le but.

Nous allons démontrer le deuxième composant du Framework Hadoop appelé MapReduce.

4.2 Spécification informelle

La spécification informelle est un type de cahier de charge pour exprimer les besoins de fonctionnement de ce système en présentant une description de l'architecture matérielle du système et une description des fonctions du système et les paramètres utilisés.

4.2.1 Aspect matériel :

MapReduce

MapReduce est un modèle de programme pour une technique de traitement basé sur l'informatique distribuée. Le modèle de programmation MapReduce a deux fonctions importantes, Map et Reduce. Le fonctionnement de Map consiste à prendre les données d'entrée sous la forme d'un ensemble de paires de valeurs clés et à produire une sortie sous forme de paires de valeurs de clés intermédiaires supplémentaires. La fonction Reduce reçoit cette paire de valeurs de clé intermédiaire générée à partir de la fonction Map en tant qu'entrée et réduit ces paires de valeurs de clé intermédiaires en un ensemble de paires plus petit. La fonction Map est exécutée après la fonction Reduce selon le nom du modèle de programme MapReduce.

Le principal avantage du modèle MapReduce est évolutif pour le traitement des données sur plusieurs nœuds de calcul. Il y a trois étapes Map, Shuffle et Reduce dans le traitement de MapReduce. Le traitement de l'étape Map consiste à traiter les données d'entrée et à créer divers petits paquets de données (1).

HDFS est utilisé pour stocker les données d'entrée sous la forme d'un fichier ou d'un répertoire et ces fichiers d'entrée passent à la fonction Map pas à pas. Dans l'étape Reduce, les données de sortie reçues de la fonction Reduce proviennent de l'étape Map et sont utilisées comme données d'entrée, les traitent et produisent un nouvel ensemble de sorties stockées dans HDFS (1).

4.2.2 Aspect fonctionnel :

Lorsqu'un utilisateur soumet un travail à Hadoop. Il doit spécifier l'emplacement des fichiers d'entrée et de sortie dans HDFS. La fonction Map and Reduce est implémentée dans les classes Java sous la forme de fichiers jar et les différents paramètres doivent être corrigés afin de permettre l'exécution des tâches pour la configuration des tâches. Ensuite, l'utilisateur de Hadoop soumet un travail la configuration du travail au Job Tracker. Maintenant Job Tracker distribue le travail et les configurations aux tâches de planification.

Il contrôle également les travaux et fournit des informations sur l'état et le diagnostic à l'utilisateur Hadoop. La responsabilité de Task Tracker est d'exécuter une tâche sur un nœud différent selon l'implémentation de MapReduce et de sauvegarder la sortie dans les fichiers de sortie sur HDFS (1).

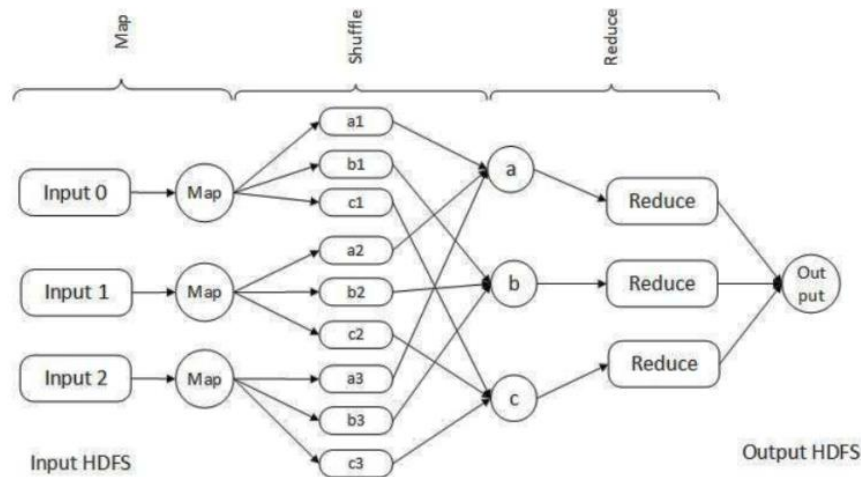


Figure 22: Processus MapReduce d'Hadoop (1).

La figure 22 définit la méthode traditionnelle de MapReduce, mais maintenant Hadoop travaille avec un nouveau paramètre nommé YARN qui donne une approche vraiment nouvelle au traitement des données, ce qui donne un moyen facile de traiter les données avec une seule plate-forme et fournit une nouvelle perspective analytique (1).

4.2.3 Le modèle de programmation MapReduce

Dans le modèle de programmation MapReduce, le développeur implémente 2 fonctions : la fonction Map et la fonction Reduce

Fonction Map : prend en entrée un ensemble de « Clé, Valeurs » et retourne une liste intermédiaire de « Clé1, Valeur1 » :

Map(key, value) -> List(key1, value1).

Fonction Reduce : prend en entrée une liste intermédiaire de « Clé1, Valeur1 » et fournit en

sortie une ensemble de « Clé1, Valeur2 » :

Reduce(key1,list(value1)) -> value2.

L'algorithme MapReduce s'exécute en 5 phases :

1. La phase Initialisation
2. La phase Map
3. La phase regroupement (Shuffle)
4. La phase de Tri
5. La phase Reduce

4.2.4 Longueur moyenne des mots

On s'intéresse maintenant à calculer la longueur moyenne des mots dans un texte. On procède alors au découpage comme précédemment au texte d'entrée en lignes. Pour chaque ligne, on calcule le nombre de mots et la longueur de la ligne (ie le nombre de caractères). La longueur moyenne se fait selon la formule :

$$\mu = \frac{1}{n} \sum_{1}^{n} x_i$$

L'entrée de Map est un ensemble de mots {w} d'un fragment du texte, la fonction « Map » calcule le nombre de mots dans le fragment et la longueur totale des mots. La sortie de la fonction « Map » contient deux couples : <“count”, #mots> et <“length”, longueur totale>. L'entrée de Reduce est un ensemble {<clé, {valeur}>}, où clé = “count” ou “length” et valeur est un entier. La fonction « Reduce » calcule le nombre total de mots : N = somme de toutes les valeurs “count” et la longueur totale des mots : L = somme de toutes les valeurs “length”. La sortie « Reduce » est <“count”, N> et <“length”, L>, Le résultat est obtenu en faisant la division $\mu=L/N$.

4.3 Spécification formelle

La spécification formelle a été faite selon la méthode d'analyse fonctionnelle et opérationnelle des applications temps réel SA-RT, cette méthode permet de réaliser une description graphique et textuelle de l'application en termes de besoins selon un modèle incrémental. La SA-RT considère la spécification des systèmes temps réel selon trois points de vue orthogonaux :

- Fonctionnel
- Informationnel
- Dynamique

4.3.1 Noyau :

On va commencer avec un petit noyau simple, en suit-on vas ajouter une seule fonctionnalité à la fois.

4.3.1.1 Spécification :

Aspect fonctionnel : La décomposition de système de contrôle des données en temps-réel selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

User

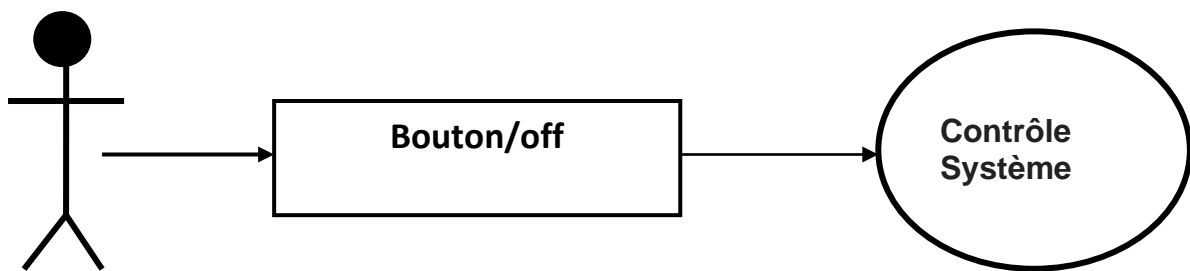


Figure 23: diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d'une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

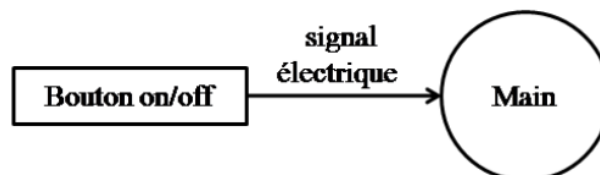


Figure 24:Diagramme de flot de données.

Aspect dynamique : Le contrôle du flux des donnée et l'activation des fonctions est exprimer par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SA-RT est représenter par le diagramme de flot de contrôle. Les événements sont généralement liés à l'activation ou à la désactivation :

- **E** pour activation « Enable ».
- **D** pour désactivation « Disable ».
- **T** pour déclenchement « Trigger ».

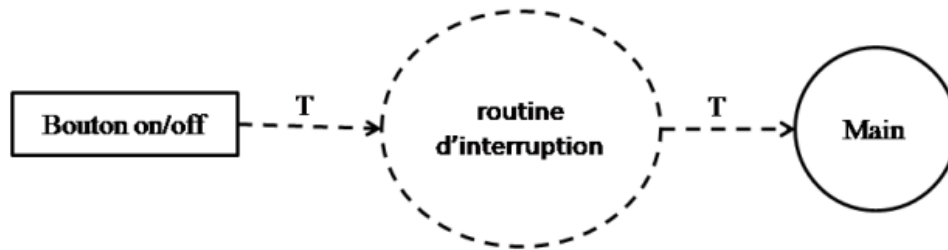


Figure 25: Diagramme de flot de contrôle.

4.3.1.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d'un modèle de spécification base sure une analyse fonctionnelle et structurée qui été réalisée avec la méthode SA-RT aux des entités de programmes dans un langage exécutable.

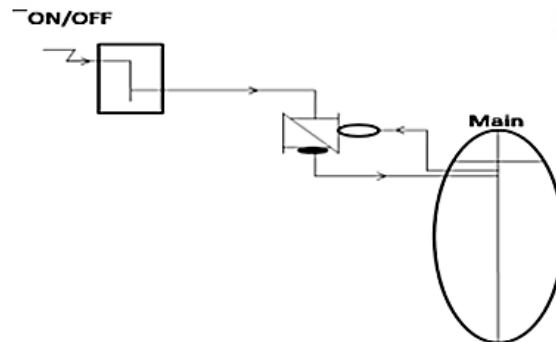


Figure 26 :Schéma multitâche.

4.3.1.3 Test

Le test se fait en assemblant les composants et en programmant la carte Arduino, les images suivantes montrent comment cela a été fait.

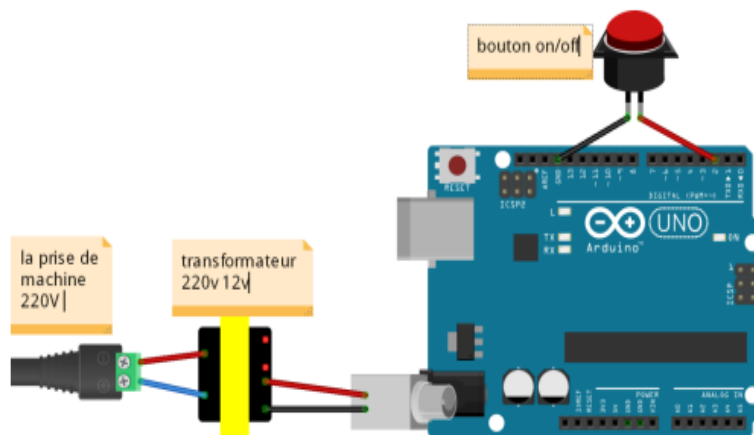


Figure 27: Schéma de câblage.

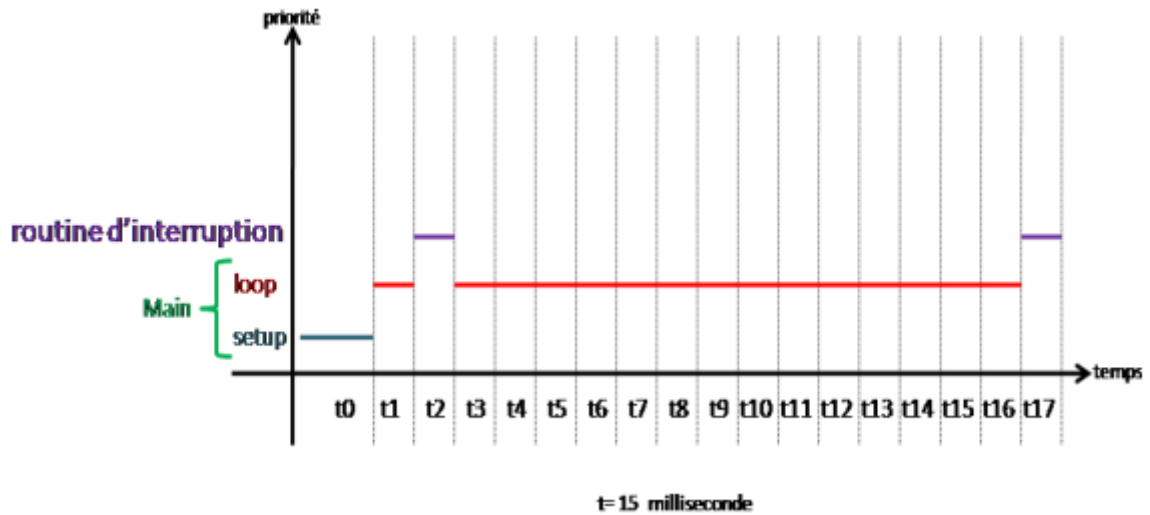


Figure 28 :Le chronogramme de la séquence d'exécution.

4.3.2 Transaction du noyau à l'incrément 01 :

On va ajouter un afficheur pour afficher l'état de la machine, et un boîte aux lettres pour la communication entre les tâches.

4.3.2.1 Les nouvelles fonctionnalités :

01- Afficher :

Pour afficher les données, avec les paramètres suivants :

Input : une chaîne de caractères.

Output : un signal numérique.

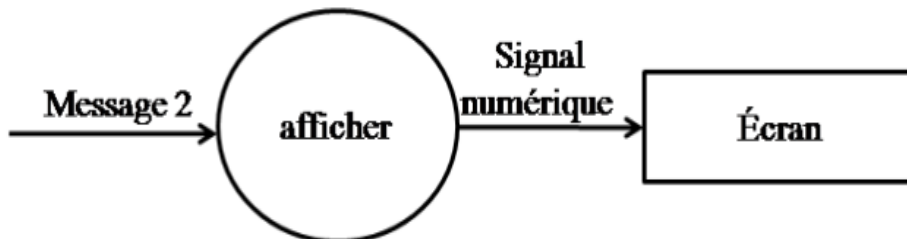


Figure 29: la tâche afficher.

02- Boîte aux lettres :

Pour stocker les données a utilisé ultérieurement par les tâches avec les paramètres suivant.

Input : une chaîne de caractères.

Output : une chaîne de caractères.

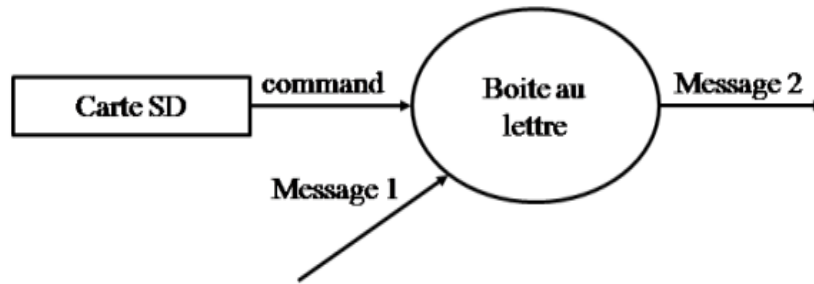


Figure 30 :la boite aux lettres.

4.3.3 Incrément 01 :

Dans cet incrément on va ajouter un lecteur de carte sd pour faciliter l’entrée des commandes exécuter sur la machine.

4.3. 3. 1 Spécification :

Aspect fonctionnel : La décomposition de système de contrôle des données en temps-réel selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

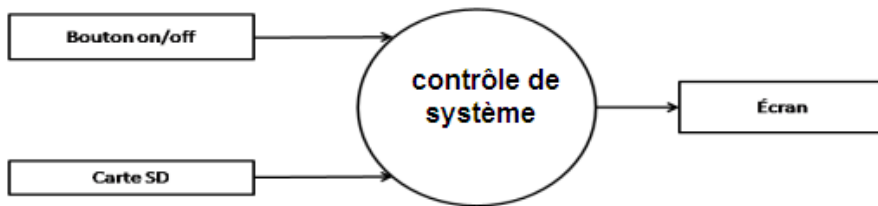


Figure 31:Diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les va leurs d’une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot dedonnées suivant.

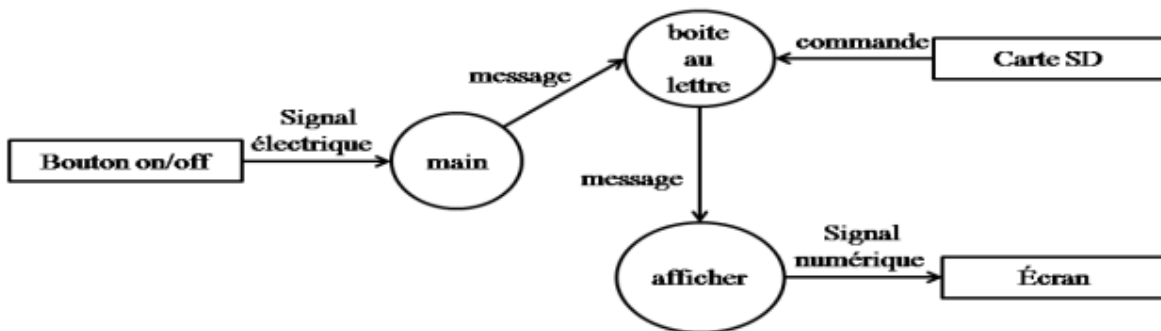


Figure 32 :Diagramme de flot de données.

Aspect dynamique : Le contrôle du flux des données et l'activation des fonctions est exprimé par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SA-Rt est représenté par le diagramme de flot de contrôle.

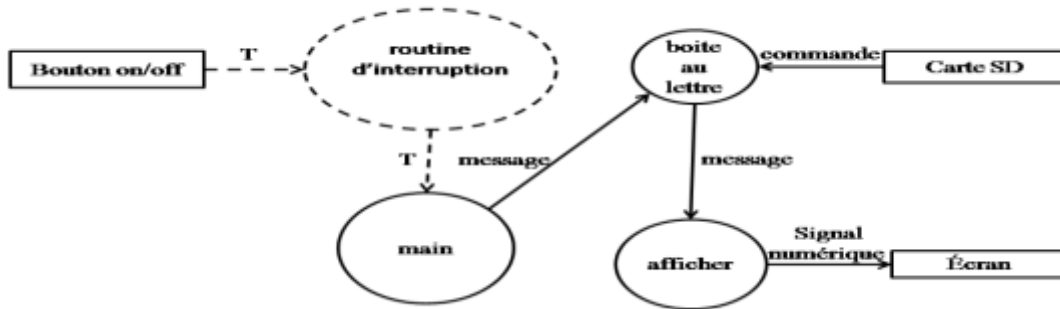


Figure 33:Diagramme de flot de contrôle.

4.3. 3.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d'un modèle de spécification basé sur une analyse fonctionnelle et structurée qui a été réalisée avec la méthode SA-RT aux entités de programmes dans un langage exécutable.

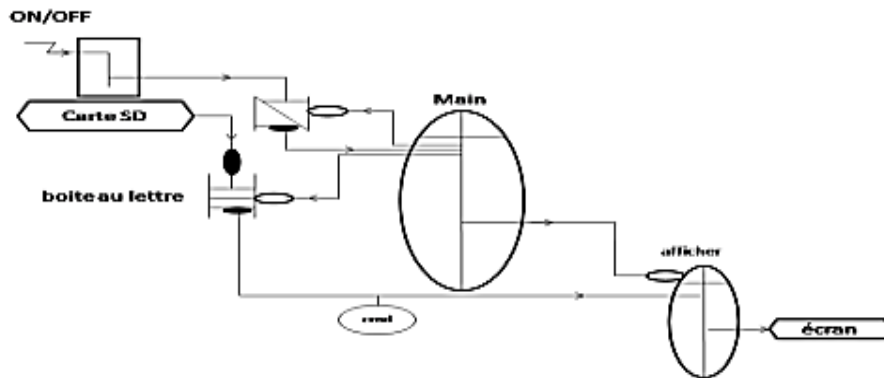


Figure 34: Schéma multitâche.

4.3.4 Transaction du l'incrément 01à l'incrément 02

4.3.4.1 Les nouvelles fonctionnalités

01- split les données :

Le système découpe automatiquement les données en entrée en bloc de données de même taille. Puis, il planifie l'exécution des tâches sur les nœuds disponibles.

Input : file de données.

Output : bloc de données de même taille.

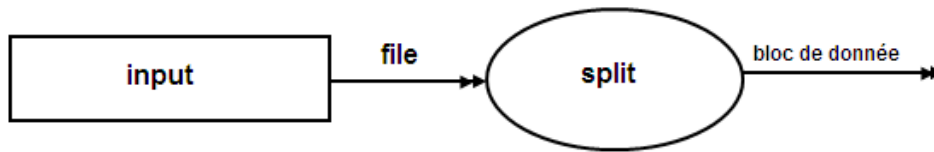


Figure 35: la tâche split.

4.3.5 Incrément 02

4.3.5.1 Spécification :

Aspect fonctionnel : gérer des flux de données en temps réel en une hiérarchie de fonction selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

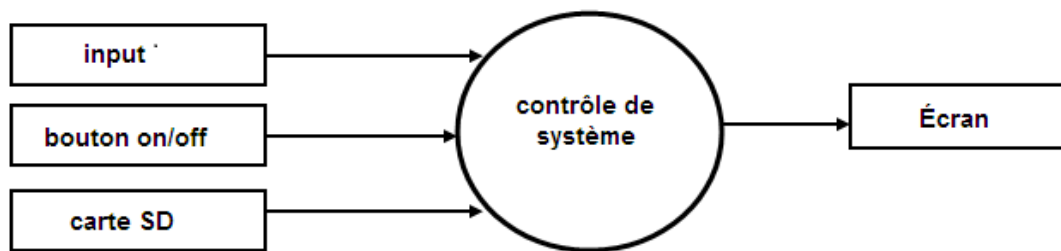


Figure 36: Diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d'une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

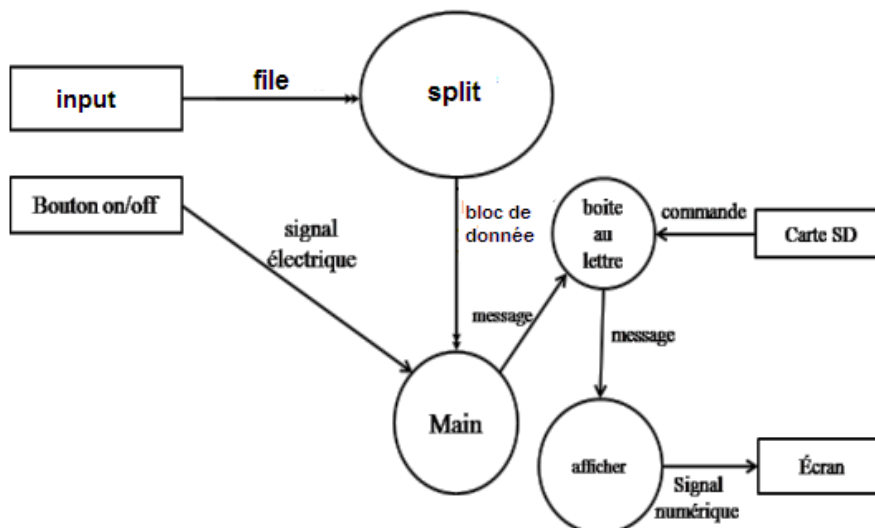


Figure 37: Diagramme De Flot De Données.

Aspect dynamique : Le contrôle du flux des donnée et l'activation des fonctions est exprimer par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SA-RT est représenter par le diagramme de flot de contrôle.

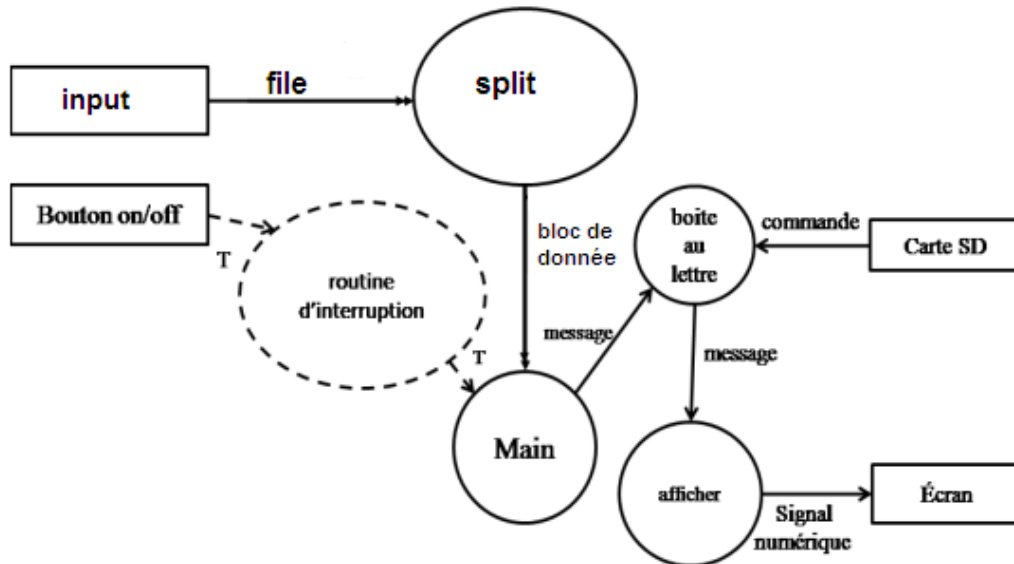


Figure 38: Diagramme de flot de contrôle.

4.3.5.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d'un modèle de spécification base sure un analyse fonctionnel et structuré qui été réalisée avec la méthode SA-RT aux des entités de programmes dans un langage exécutable.

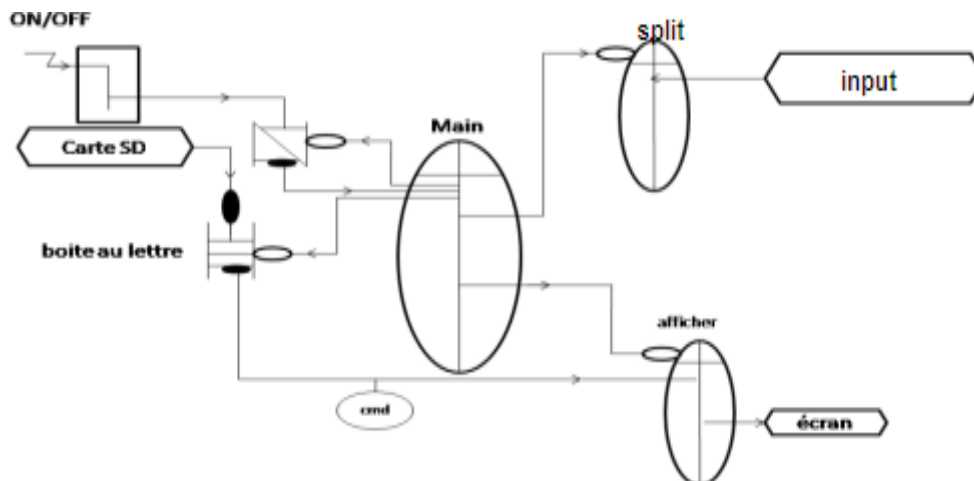


Figure 39: schéma multitâche.

4.3.6 Transaction du l'incrément 02 à l'incrément 03

4.3.6.1 Les nouvelles fonctionnalités

01-fonction Map: une tâche de map prend une paire clé/valeur du lecteur d'entrée, effectue des calculs dessus, puis produit également le résultat sous la forme d'une paire clé/valeur. Les résultats des tâches de carte sont initialement sortis dans la mémoire tampon principale et, lorsqu'ils sont presque pleins, se répandent sur le disque.

Input : paire clé/valeur du lecteur d'entrée.

Output : paire clé2/valeur2.

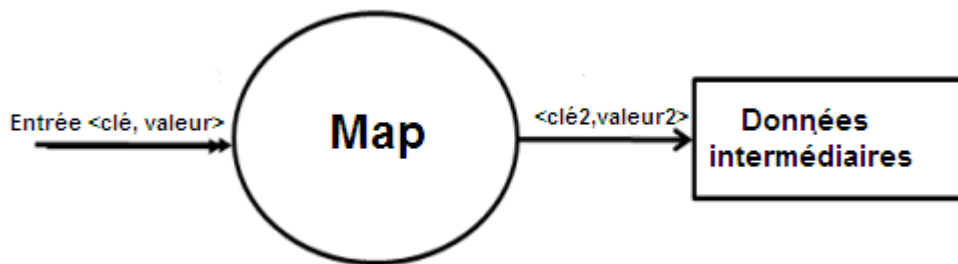


Figure 40 : la tâche Map.

4.3.7 Incrément 03

4.3.7.1 Spécification :

Aspect fonctionnel : gérer des flux de données en temps réel en une hiérarchie de fonction selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

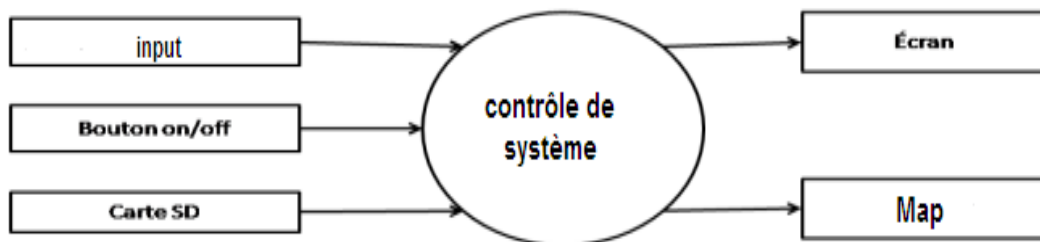


Figure 41: Diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d'une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

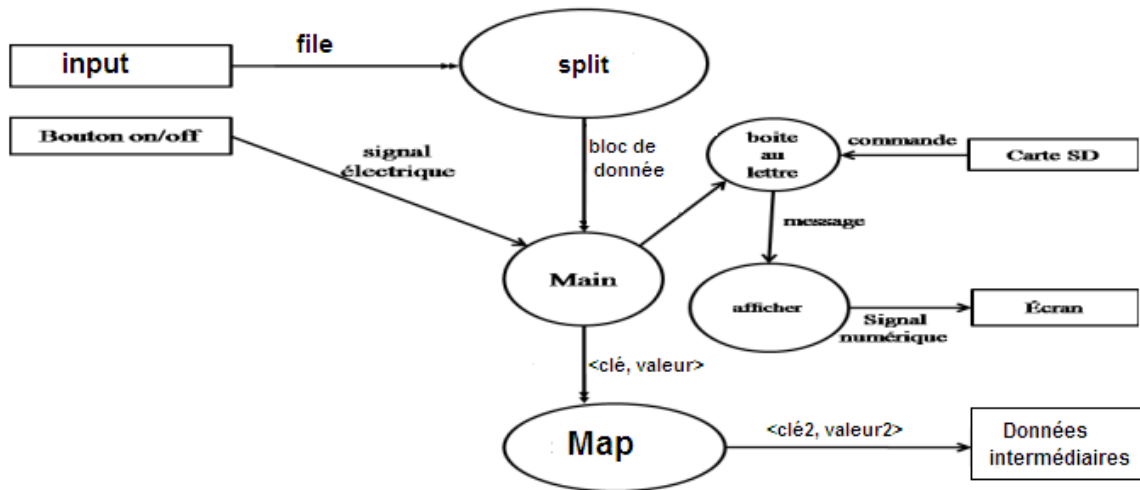


Figure 42: Diagramme de flot de données.

Aspect dynamique : Le contrôle du flux des données et l'activation des fonctions est exprimé par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SA-RT est représenté par le diagramme de flot de contrôle.

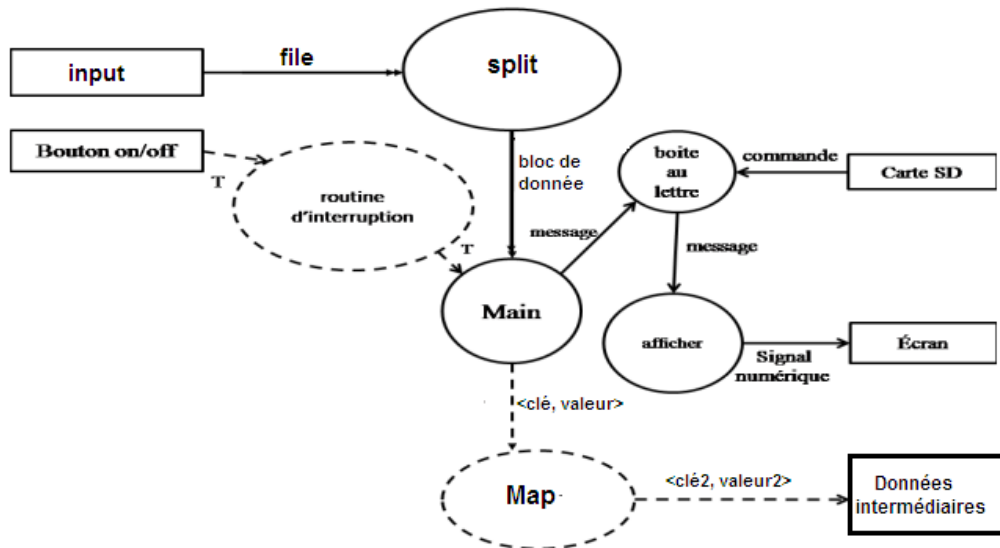


Figure 43: diagramme de flot de contrôle.

4.3.7.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d'un modèle de spécification basé sur une analyse fonctionnelle et structurée qui a été réalisée avec la méthode SA-RT aux entités de programmes dans un langage exécutable.

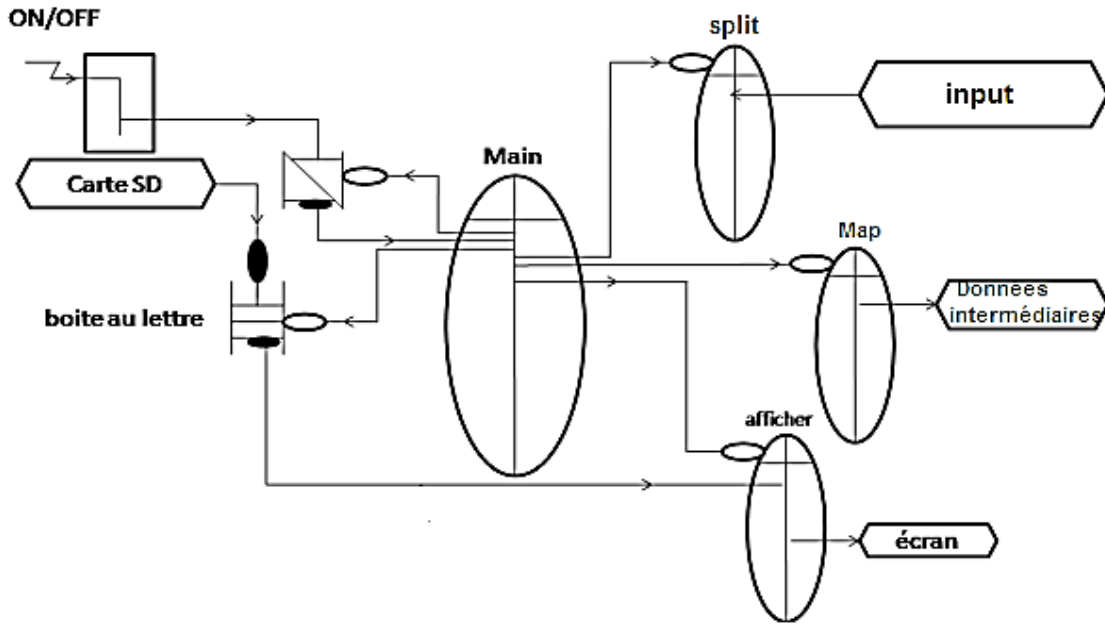


Figure 44: schéma multitâche.

4.3.8 Transaction du l'incrément 03 à l'incrément 04

4.3.8.1 Les nouvelles fonctionnalités

01-Shuffle

Le processus Shuffle agrège toutes les sorties du Mapper en regroupant les valeurs clés de la sortie du Mapper et la valeur sera ajoutée dans une liste de valeurs. Ainsi, le format de sortie du Shuffle sera le suivant :

- une map<clé, Liste<liste de valeurs>>.
- La clé de la sortie Mapper sera consolidée et triée. La sortie du Mapper sera envoyée au Reducer en utilisant la séquence de clés triées.

Input : <clé2, valeur2>.

Output : <clé2, liste(valeur2)>.

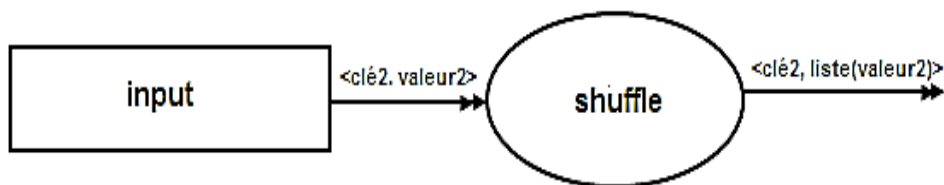


Figure 45: la tâche shuffle.

02- Boite aux lettres :

Pour stocker les données a utilisé ultérieurement par les tâches avec les paramètres suivant

Input : une chaîne de caractères.

Output : une chaîne de caractères

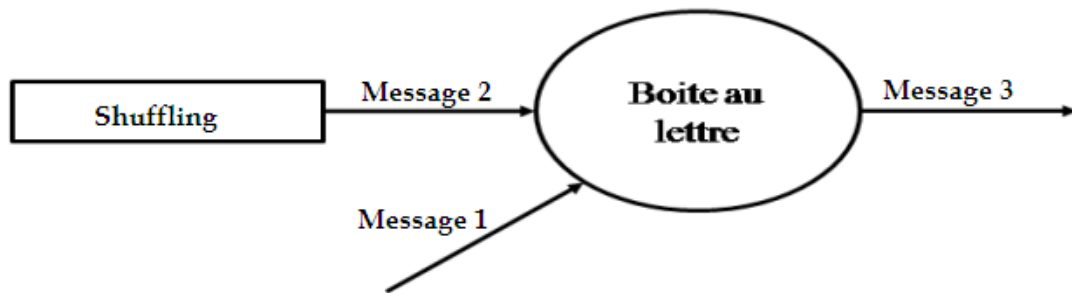


Figure 46 :La boîte aux lettres.

4.3.9 Incrément 04

4.3.9.1 Spécification :

Aspect fonctionnel : gérer des flux de données en temps réel en une hiérarchie de fonction selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

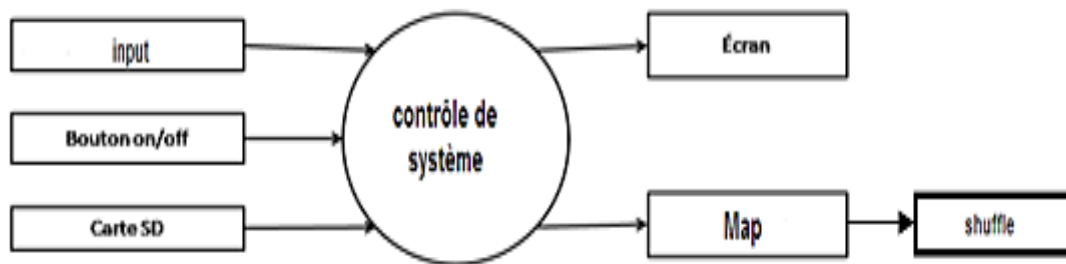


Figure 47: diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d'une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

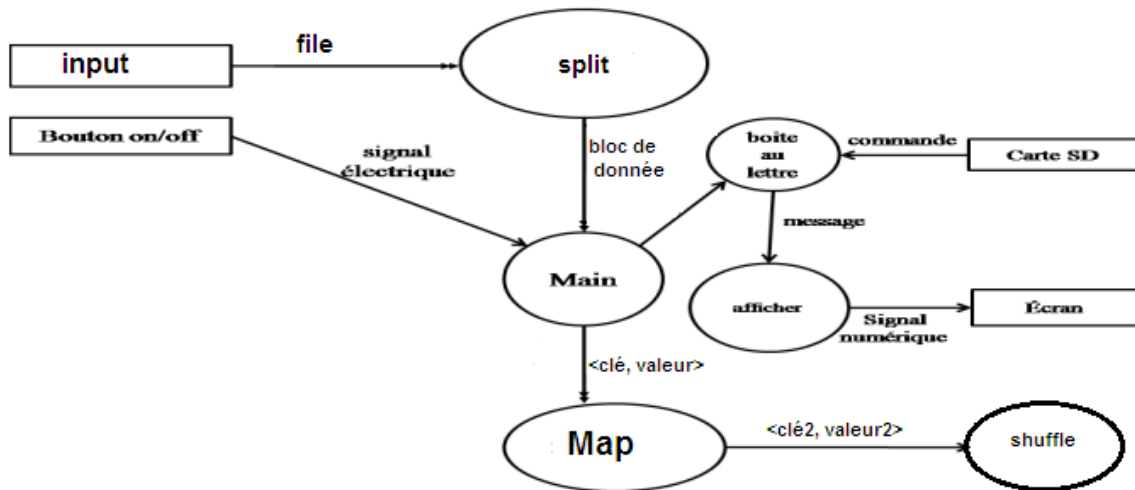


Figure 48: Digramme de flot de données.

Aspect dynamique : Le contrôle du flux des donnée et l’activation des fonctions est exprimer par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l’exécution des processus de transformation de données, selon la méthode SA-RTTest représenter par le diagramme de flot de contrôle.

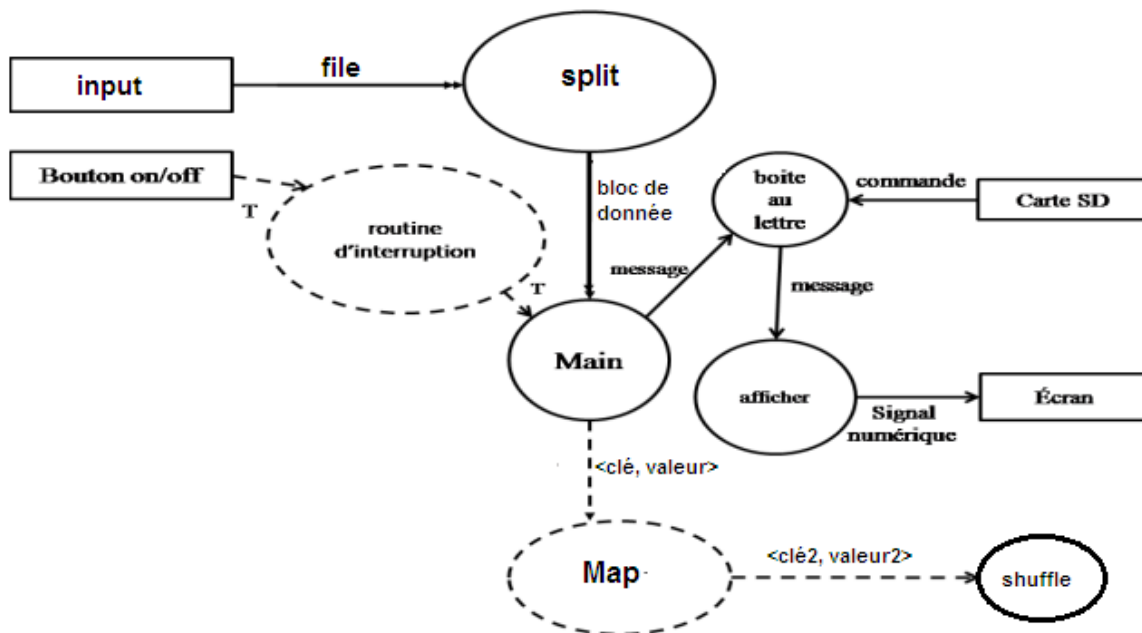


Figure 49: Diagramme de flot de contrôle.

4.3.9.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d'un modèle de spécification basé sur une analyse fonctionnelle et structurée qui a été réalisée avec la méthode SA-RT aux des entités de programmes dans un langage exécutable.

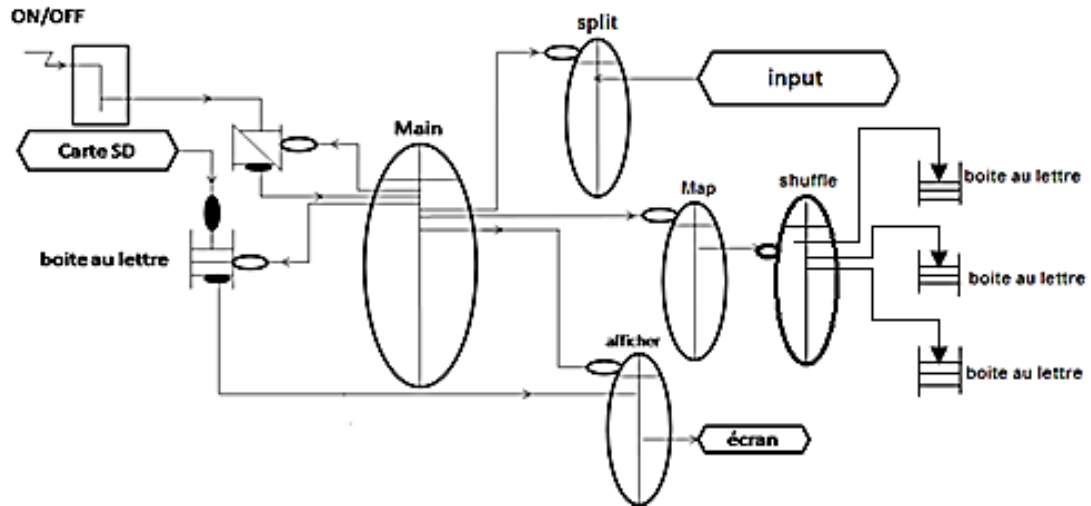


Figure 50: Schéma Multitâche.

4.3.10 Transaction du l'incrément 04 à l'incrément 05

4.3.10.1 Les nouvelles fonctionnalités

01- fonction Reduce

- C'est le processus d'agrégation où les données après le shuffle et le tri:
- sont envoyées au Reducer où nous avons $\langle \text{key}, \text{List}\langle \text{list of values} \rangle \rangle$, et le Reducer va traiter la liste des valeurs.
- Chaque clé peut être envoyée à un Reducer différent.

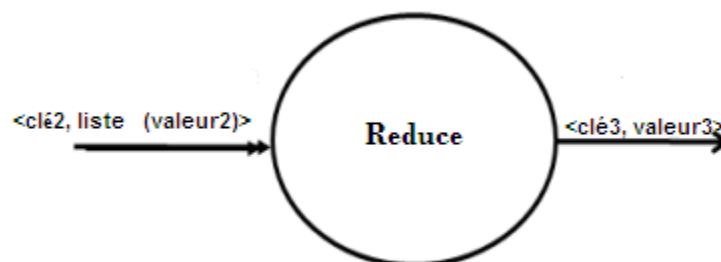


Figure 51: la tâche reduce.

4.3.11 Incrément 05

4.3.11.1 Spécification :

Aspect fonctionnel : gérer des flux de données en temp réel en une hiérarchie de fonction selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

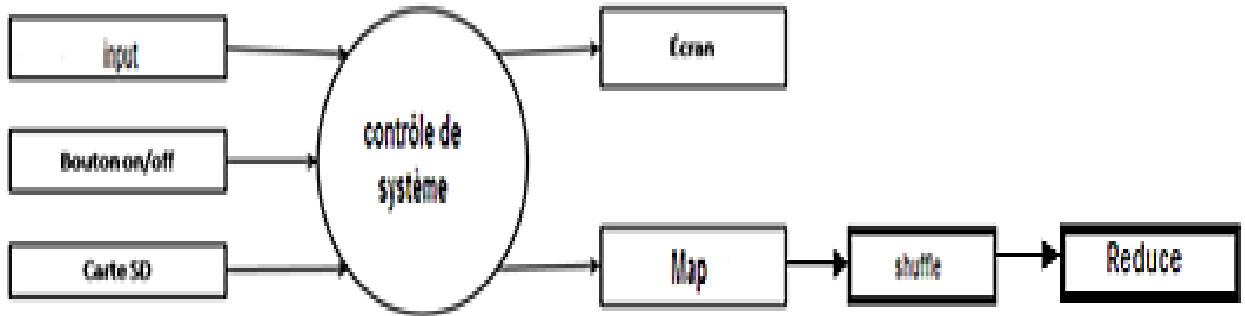


Figure 52: Diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d’une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

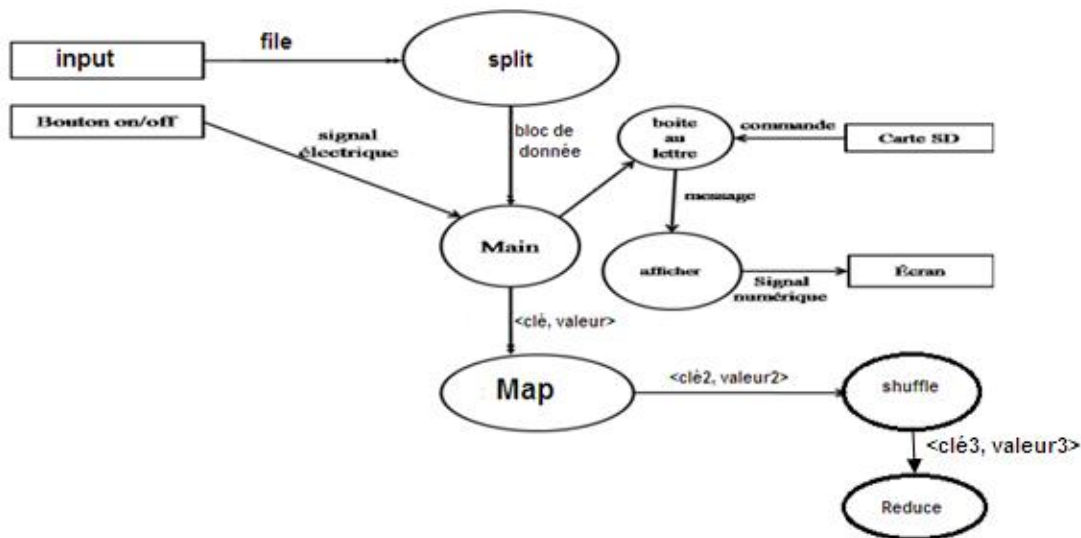


Figure 53: Diagramme de flot de données.

Aspect dynamique : Le contrôle du flux des données et l'activation des fonctions est exprimé par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SARTest représenté par le diagramme de flot de contrôle.

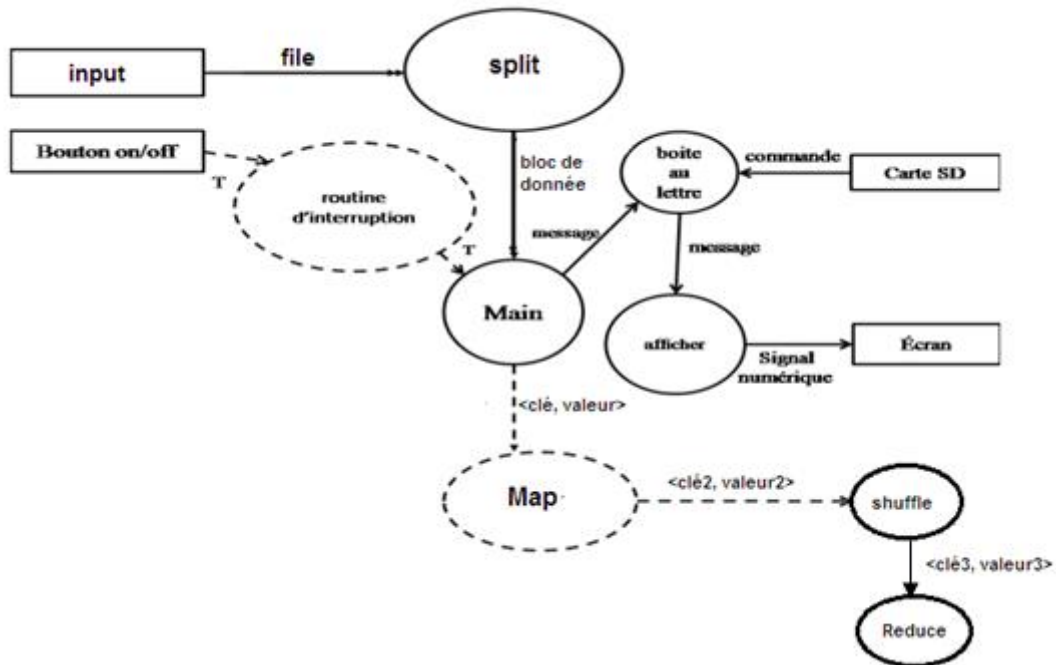


Figure 54: Diagramme de flot de contrôle.

4.3.11.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d'un modèle de spécification basé sur une analyse fonctionnelle et structurée qui a été réalisée avec la méthode SA-RT aux entités de programmes dans un langage exécutable.

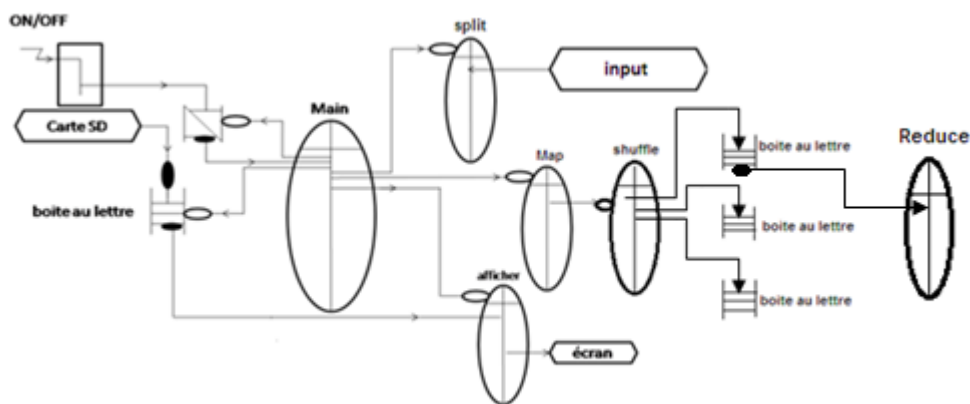


Figure 55: Schéma multitâche.

4.3.12 Transaction du l'incrément 05 à l'incrément 06

4.3.12.1 Les nouvelles fonctionnalités

01-Result : Les résultats de chacun des nœuds sont regroupés et triés pour stockage et/ou restitution.

Chaque Reducer calcule le nombre total des exceptions, par exemple :

Reducer 1 :

Reducer 2 :

Reducer 3 :

Les données générées montrent que l'exception A est déclenchée plus souvent que les autres et donc qu'elle justifie un examen plus approfondi. Si l'analyse porte sur plusieurs semaines ou plusieurs mois de données, MapReduce montre alors toutes ses qualités.

$$\text{Map: } (key_1, value_1) \rightarrow list(key_2, value_2)$$

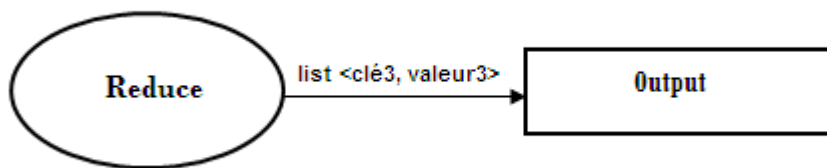
$$\text{Reduce: } (key_2, list(value_2)) \rightarrow list(key_3, value_3)$$


Figure 56: output.

02- Boîte aux lettres :

Pour stocker les données a utilisé ultérieurement par les tâches avec les paramètres suivant :

Input : une chaîne de caractères.

Output : une chaîne de caractères



Figure 57 :la tâche boîte aux lettres.

4.3.13 Incrément 06

4.3.13.1 Spécification :

Aspect fonctionnel : gérer des flux de données en temps réel en une hiérarchie de fonction selon la méthode SA-RT est représentée par le diagramme de contexte suivant.

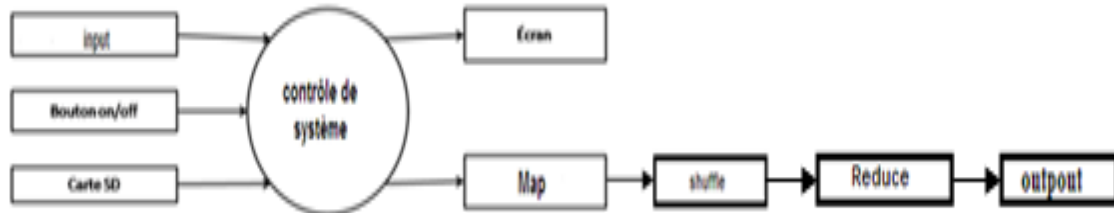


Figure 58: diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d'une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

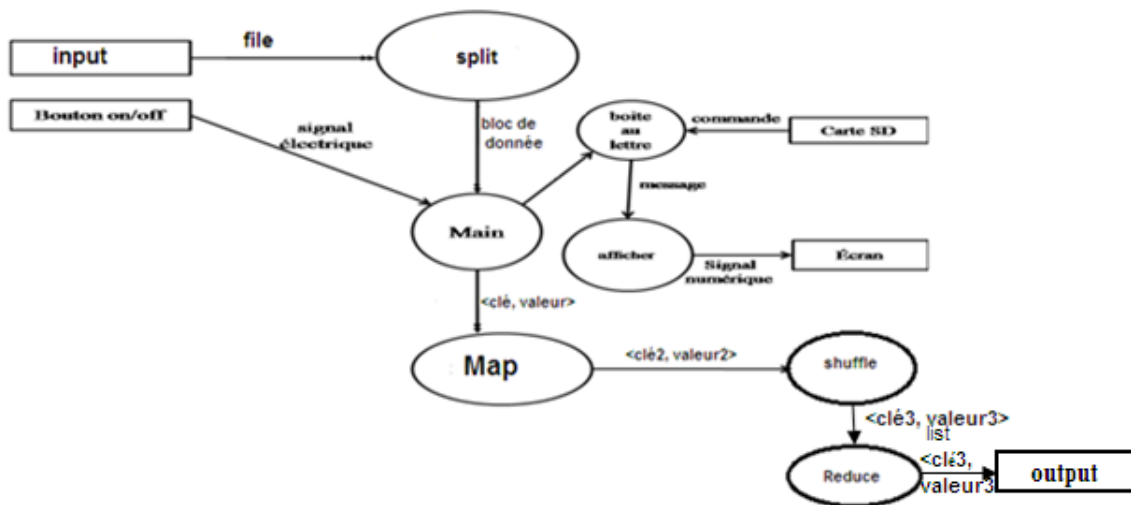


Figure 59: Diagramme de flot de données.

Aspect dynamique : Le contrôle du flux des données et l'activation des fonctions est exprimé par le flot de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SA-RT est représenté par le diagramme de flot de contrôle.

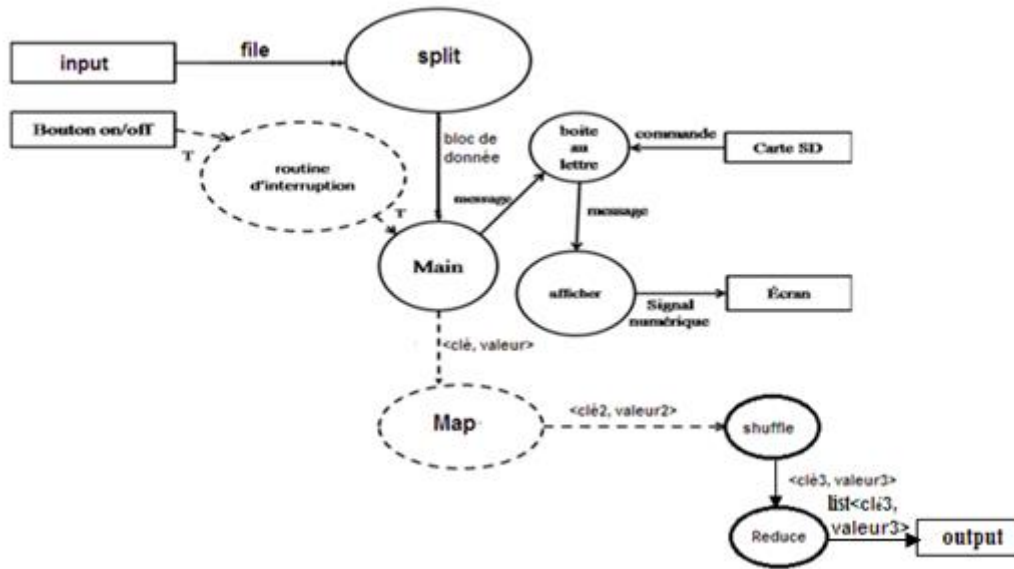


Figure 60: diagramme de flot de contrôle.

4.3.13.2 Conception

On utilise la méthode de conception multitâche « LACATRE » pour permettre le passage d’un modèle de spécification base sure un analyse fonctionnel et structuré qui été réalisée avec laméthode SA-RT aux des entités de programmes dans un langage exécutable.

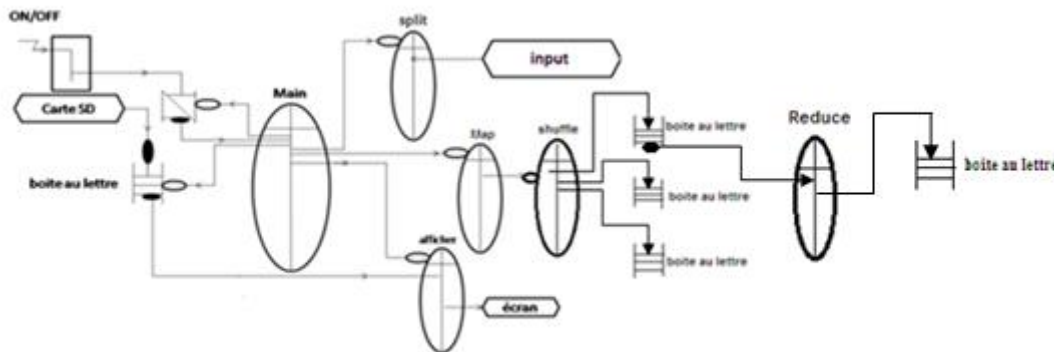


Figure 61 : schéma multitâche.

Conclusion

Dans ce chapitre, nous avons démontré une conception détaillée avec l’implémentation de FreeRTOS En particulier, le travail se concentrera sur la conception incrémentale d’une application multitâche temps réelle dotée des fonctionnalités MAPREDUCE.

Conclusion générale

Conclusion générale

Hadoop est aujourd'hui le standard de fait en tant que plateforme d'exécution de MapReduce. Conçu pour fonctionner sur des clusters homogènes de machines peu coûteuses, il prend en charge l'essentiel des problématiques de bas niveau lié à la planification des tâches, à la répartition de charge et à la réplication des données.

Le noyau d'Hadoop est constitué d'une partie de stockage: HDFS (Hadoop Distributed File System), et d'une partie de traitement appelée MapReduce. Hadoop fractionne les fichiers en gros blocs et les distribue à travers les nœuds du cluster. Pour traiter les données, Hadoop transfère le code à chaque nœud et chaque nœud traite les données dont il dispose. Cela permet de traiter l'ensemble des données plus rapidement et plus efficacement que dans une architecture supercalculateur plus classique qui repose sur un système de fichiers parallèle où les calculs et les données sont distribués via les réseaux à grande vitesse.

Ce projet avait pour objectif d'appliquer les notions de la méthodologie de conception SA-RT selon une approche incrémentale pour concevoir un système de contrôle command pour présenter un nouveau Framework MapReduce pour les calculs incrémentaux à grande échelle.

Basé sur plusieurs nouvelles techniques pour maximiser la réutilisation des résultats d'un calcul précédent. En particulier, intègre une segmentation basée sur le contenu au système de fichiers pour détecter les modifications incrémentielles dans le fichier d'entrée et pour partitionner les données afin de maximiser la réutilisation ; il ajoute une phase de contraction pour contrôler la granularité des tâches dans la phase de réduction, et un nouveau planificateur qui prend en compte l'emplacement des résultats précédemment calculés.

Nous l'avons implémenté en tant qu'extension de Hadoop. Notre évaluation des performances montre qu'il est possible d'améliorer l'efficacité des exécutions incrémentielles (cas courant), à un coût modeste lors de la première exécution initiale (cas peu courant) où aucun calcul ne peut être réutilisé.

Bibliographies

Bibliographies

- [1] M.Mohamed, « Un ordonnancement efficace des tâches pour les applications Big data », Département d'informatique Laboratoire EEDIS, Sidi Bel Abbès, 2019.
- [2] N. Hasna, « HADOOP pour Big Data », chez *Big Data*, Gabès, TUNISIE, 2022.
- [3] « BIG DATA & HAODOP », Dexlab, 2018.
- [4] R. Solium, « Big Data & Hadoop », chez *Architecture and Development*.
- [5] H. BAH, « Big Data, Hadoop et MapReduce », Faculté des Sciences de l'Ingénieur Département Informatique, ANNABA, 2019.
- [6] « Introduction à Docker », 11 06 2015.
- [7] G. M. DIOUF, « UNE PLATEFORME D'ORCHESTRATION DE CONTENEURS DOCKER TOLÉRANTE AUX FAUTES BYZANTINES », UNIVERSITÉ DU QUÉBEC À MONTRÉAL, Canada, 2019.
- [8] « RedHat », 9 janvier 2018. [En ligne]. Available: <https://www.redhat.com/fr/topics/containers/what-is-docker>.
- [9] H. Bour, « Docker et machines virtuelles », 06 janvier 2006. [En ligne]. Available: <https://www.padok.fr/blog/machines-virtuelles-docker>.
- [10] « Digital Guide », 16 09 2020. [En ligne]. Available: <https://www.ionos.fr/digitalguide/serveur/know-how/les-lxc-linux-container-quest-ce-que-cest/#c266658>.
- [11] S. John « docker », sabel publishing, 2016.
- [12] « ASPENCORE 2019 Embedded Markets Study Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments », 2019. [En ligne]. Available: www.eetimes.com. [Accès le 2020].
- [13] B. Richard, « Mastring the FreeRTOS real time kernel », 2020. [En ligne]. Available: <http://www.FreeRTOS.org>. [Accès le mai 2020].
- [14] L. Bastien « Hadoop – tout savoir sur la principale plateforme Big », 2018.
- [15] <http://webcache.googleusercontent.com/search?q=cache:f86gHrK39wEJ:e-biblio.univ-mosta.dz/bitstream/handle/123456789/15682/M%25C3%25A9moire.pdf%3Fsequence%3D1&cd=5&hl=fr&ct=clnk&gl=>
- [16] H. ADIM. tiaret: CONCEPTION D'UN NOYAU DE CALCUL POUR CONTROLE COMMANDE TEMPS REEL DE LA MACHINE-TOUR, 2020.
- [17] *DOMOTICS.fr*. mai 20, 2019. <https://domotics.fr/index.php/2019/02/25/les-differentes-cartes-arduino/>.