KIP-20 - Enable log preallocate to improve consume performance under windows and some old Linux file system

Status

Current state: *Accepted*

Discussion thread: Here

JIRA: *KAFKA-1646*

Motivation

Currently, when create on LogSegment, always create on empty file and keep APPEND data to it, for Linux file system (ext2/ext3/ext4 etc), it works fine. But for windows and some old unix/linux file system, after a while, there will be more and more fragments on hard disk, and affect consume performance a lot. So if we pre allocate file with one bigger value (for example, 512MB) when create file, it will help us reduce fragments on hard disk and improve consume performance.

Public Interfaces

This proposal add one config.

log.preallocate - Should pre allocate file when create new segment? Default value is false for backward compatible. **If you are using Kafka on Windows, you probably need set it to true.**

Proposed Changes

1. Configuration - add one configuration item "log.preallocate", parse it in KafkaConfig.scala, and transfer to KafkaServer.scala, LogConfig.scala.
2. In Log.scala
    a. add one function initFileSize(). if the log preallocation is enabled, the value is config.segmentSize, else the value is 0.
    b. pass the initFileSize and config.preallocate to LogSegment when need create one new LogSegment.
    c. When need roll to a new file, trim log file of active segment
3. In LogSegment.scala
    a. add 3 parameters to constructor and pass the parameters (**fileAlreadyExists, initFileSize and preallocation)** to FileMessageSet.
4. In FileMessageSet.scala
    a. add one function trim(). Call it when close or roll a new LogSegment.
    b. Move openChannel from CoreUtil.scala and add three more parameter "fileAlreadyExists", "initFileSize" and "preallocate".
    c. Add one constructor function with three more parameters "fileAlreadyExists", "initFileSize" and "preallocate".
        i. if file is one new file (fileAlreadyExists = false) and preallocate is true, end will be set to 0, and position will be set to 0
        ii. Else the end will be set to Int,MaxValue and position will be set to file size
    d. Change calculation of the position after create or open file
        i. The code changed to "" channel.position(math.min(channel.size().toInt, end)) ""
        ii. if file is one new file (fileAlreadyExists = false) and preallocate is true, end will be set to 0, position will be set to 0
        iii. Else the end will be set to Int,MaxValue and position will be set to file size
5. In CoreUtils.scala
    a. Move openChallel to FileMessageSet.scala

Compatibility, Deprecation, and Migration Plan

This change is fully backward compatible.

Below are detail description for the scenarios:

1. If do clear shutdown, the last log file will be truncated to its real size since the **close()** function of **FileMessageSet** will call **trim(),**
2. If crash, then when restart, will go through the process of **recover()** and the last log file will be truncate to its real size, (and the position will be moved to end of the file)
3. When service start and open **existing file**
    a. Will run the LogSegment constructor with parameter "fileAlreadyExists" as true
    b. Then in FileMessageSet, the "**end**" in FileMessageSet will be **Int.MaxValue**, and then "**channel.position(math.min(channel.size().toInt, end)**)" will make the position be end of the file,
    c. If recover needed, the recover function will truncate file to end of valid data, and also move the position to file end,
4. When service running and need create **NEW** log segment and new FileMessageSet
    a. If preallocate = true
        i. the "**end**" in FileMessageSet will be **0,** the file size will be "**initFileSize**", and then "**channel.position(math.min(channel.size().toInt, end)**)" will make the position be **0**,
    b. Else if preallocate = false
        i. backward compatible, the "**end**" in FileMessageSet will be **Int.MaxValue**, the file size will be "**0**", and then "**channel.position(math.min(channel.size().toInt, end**))" will make the position be **0**,

Rejected Alternatives