關於斷點續傳上傳與秒傳技術分析

瀏覽：337 2020-02-26 編輯：香港互聯 來源：互聯網

超大文件上傳就無法避免斷點上傳，想要實現秒傳妳就饒不開文件特征。本文不涉及具體代碼。只講我自己對斷點上傳於秒傳的學習經驗。

首先我們分析壹下要實現斷點續傳於秒傳的具體要點。

1、前端拆分碎片

2、前端生成文件特征

3、傳輸過程

4、後臺接收文件合並

5、合並完成後，後臺的特征對比

看似原理簡單,粗狂分析壹下就可以動手了,其實每壹步都有很多技術難點於陷阱。

1、前端拆分文件碎片

在以前只要斷點續傳就必須采用前端控件。控件主要工作就是在拆分文件與碎片與分析md5□現在的html5本身已經支持對文件拆分讀取上傳。至於如何拆分自己搜索壹下很容易找到demo□這裏陷阱不多只要考慮壹下兼容性就可以了,IE系列可以考慮用flash□

2、前端生成文件特征

這裏說明壹下,我沒有直接說是md5 是因為md5現在在大型網盤項目裏碰撞的幾率已經很高了,不能完全依賴md5,盡量多取幾種哈西值壹起傳到後端做特征。(取文件的幾部分做特征是很不靠譜的不用考慮了)

在前端生成特征才能實現秒傳功能(也許有不需要的方法反正我不知道)。文件生成哈西值，是采用流運算方式把整個文件讀取分析壹遍。由於需要讀取，那麼超大文件的耗時是不可避免的。目前js 有個很好的庫類可以支持采用文件流方式計算md5 □spark-md5.js 這是個好東西。只需要把文件碎片挨個分析壹遍,文件全

部跑完就能生成壹個md5了理論上可以生成任意大小文件的md5測試過8G的文件生成壹個md5碼需要十幾分鐘。這取決於客戶機器性能和磁盤速度。網上還有其他哈西的文件流獲取算法,請自行百度。

3、傳輸

傳輸現在html5現在可以采用ajax和websocket兩種方式可以任選。這裏我選用的是自己熟悉的ajax方式,websocket只是初略的看了壹下。就不發表具體看法了。

ajax生成傳輸隊列的時候有個坑需要註意壹下。當拆分碎片上傳，也就是每壹個碎片是壹個ajax請求。那麼壹個8G的文件將會生成多少請求呢。壹個循環下去瀏覽器死了。所以要采用遞歸方式進行控制ajax堵塞數量。壹次循環傳輸10個碎片,傳完在傳下壹組。

4、後臺的文件合並。

接收到文件就需要考慮合並問題了。我之前百度了壹下。基本都告訴我等待文件碎片全部上傳完成然後合並。當我上傳壹個8G的文件到後臺後然後循環合並,太恐怖了上萬的碎片合並。想想就難受。

第壹個想法是放到隊裏裏在後面慢慢跑。後來放棄了感覺體驗不是很好。

後來想到了邊上傳邊合並,把合並時間分散到每壹個請求裏。每個請求只處理1-2mb的碎片還是很快的。

剛開始參考迅雷想生成壹個和目標文件體積相等的空白文件。然後等待碎片來拼入文件。後來發現php不能快速幹這個事情。有其他方法也懶著研究了。就寫了壹個擴容算法

算法說明：

收到請求後如果第壹個碎片是2號，那麼在生成的文件裏填充字符串把1號碎片的位置留下，然後寫入2號碎片,等1號碎片 達到，在把1號碎片寫入他自己的位置。

算法基本上很快沒有什麼延時。好在我們不是做下載,上傳時不會直接收到最後的碎片。

寫好後進行測試發現文件大於2g就不在生成了。塊是很快了不過不能處理大文件。後來發現是php文件指針在32位系統下最大值是2G如果移動大於2G 就會返回-1失敗。那麼理論上這個算法最大生成文件是2G(或者是4G 指針從文件尾巴向前計算還有2g不過我沒有繼續研究下去,理論上php讀取文件內容最大值是4G)下面換壹種算法。

最簡單的辦法，就是往文件尾巴裏追加內容,這樣就不受文件指針大小影響了ajax 在並發的時候到達順

序是不壹樣的。但是這個不影響，不會對我們後臺編程產生太大的障礙。後臺進行搬磚壹樣的排序寫入就好了。

舉例：

如果是第壹個碎片收到的是2號，那麼把碎片文件拷貝出來備用。

下壹個請求碎片如果是1那麼寫入文件，寫入後詢問備用裏有沒有2號如果有寫入。以此類推。

這個算法和上面的擴容方法比起來效率要低了壹些。因為要做計數器和詢問利用。但是影響不是很大。

我兩種算法都采用了如果文件小於1.99g用上面得擴容算法,大於就用排序。

合並也完成了剩下就是對結果進行效驗了。

5、後臺文件效驗。

不要相信 php 或者其他語言的 md5_file這玩意處理不來大文件。幾個G基本上就崩潰了。記住壹點只要妳想獲得文件的md5 就必須對這個文件進行讀取分析壹遍才行怎麼做都是很耗時的。

像百度那種大型網盤已經完成對TB級別md5隨時生成,真的很牛叉。後來我分析了壹下他們應該是這麼做的。

應該和我們處理文件合並的方式差不多。把計算md5的時間分攤到了每壹個碎片上。我們可以這麼理解,md5 需要順序分段讀取整個文件進行分析才能生成壹個md5值,那麼我們在上傳過程中每壹個碎片都是文件的壹部分。我們只要順序分析每壹個文件碎片,那麼上傳完成了md5也就生成了(理論上也能到TB級別)。

我嘗試用這種方式找了壹個php 版本的md5算法,改進了壹下,實現了超級大文件上傳完成後臺即使獲得到md5的類。（測試過與 md5_file函數獲得的值壹樣.）

經過測試完全可行。 "但是,可但是,但可是" php真的不適合幹這種大型純運算很慢，基本上js 算md5的效率 是php幾十倍。壹個2mb的碎片計算需要9秒左右。

這麼做每壹個碎片後臺響應時間都超長，也就直接影響上傳速度了。

幾經研究無果。決定放棄php 找師哥用c寫個擴展來幹這個事情了。

項目基本完成了,至於其他的續傳 和 秒傳 沒什麼好講的,妳都已經有碎片了續傳不是問題。妳都能獲得前後端特征碼了,那麼秒傳也不是問題了。

這個過程最耗時的是前端計md5,其他運算都分攤到每壹碎片上了。