THIS DOCUMENT IS PUBLIC

Hyphenation IPC on Android

kojii@chromium.org, Aug 2016

Overview

To implement CSS Hyphens in Blink (launch bug, design doc), the renderer needs to read the hyphenation dictionaries. The renderer can mmap dictionary files to read but in the sandbox architecture, it needs to ask browser process to open them first. On Android, a discussion at platform-architecture-dev@ reached a consensus to measure the sync IPC at dev channel.

UMA Results and Other Data

One of M54 dev was changed to open hyphenation dictionaries in *all* pages, once per process, and measure how long it takes. Measuring only hyphenated pages was not possible, because it requires to expose the "`hyphens`" property to public.
Mean: 5ms
95 percentile: 10ms
98 percentile: 20ms
99 percentile: 40ms
The histogram is "Hyphenation.Open", the actual data is available only to Googlers (sorry.)
This code path will kick in only for pages with "hyphens" property set. We do not have UMA for this property, only some data publicly available:

- Crawler data for the "`-ms-hyphens`" property by Microsoft indicates 0.4% of pages. Note that UMA could be very different from crawler data, larger or smaller, by the factor of more than 10.
- This property is available on Firefox/Safari/IE/Edge (with prefix other than Firefox, data in caniuse.com.) The usage might shift once Blink supports.

Next Action Options

1. Ship the current implementation

Mean 5ms is expected. 99 percentile does not look great though. Shipping the property allows us to get the real data of the property usage and improve incrementally.

2. Improve the IPC in the worst cases

- Probably the file thread is too busy to pick the IPC request. Would having more threads, or a dedicated thread help? If all cores are really busy, adding more threads may not help?
- Other ideas?

3. Issue requests earlier in style recalc

Elliott suggested we could issue the request at style recalc to maximize the parallel time.
We need a method to wait for async IPC to finish, which AFAIU we don't have today? Can mojo add this capability? Or should we just spin loop to avoid yielding CPU to other threads?
This might not be a great help for the worst case, as style recalc to first line break shouldn't be that long? I'm not as familiar with style performance as to be confident on this comment.

4. Open all dictionaries at startup

The renderer then has all file handles when it needs, no IPC is needed. This option helps overall performance if opening files is fast but IPC is not on high load.
Currently N preview has 33 hyb files. We'll add cost to open them in every startup, including when hyphenation is not used at all.
Android opens all dictionary files in java Hyphenator.init(). If future Android can give the list of file handles for all languages, it'll be cheaper. It does not help existing Android though.
We could rely on system languages to limit the number of dictionaries to open at startup, and use IPC for other languages.
We don't have data how much hyphenated pages are in different languages than the system language. We can identify hyphenated pages when we ship the property.

5. Layout without hyphenation, then re-layout

This does not cost to the first layout, but this doubles layout cost, wastes batteries, and flicker.
Also changing layout after layout complete is not web compatible. Some developers already expressed concerns about changing layout after layout complete.
We could add, for instance, 'hyphens: async' to make it asynchronous, but it does not solve existing pages.

6. Interrupt layouts and yield when dictionaries are opened

We don't have capability to interrupt layout today. Maybe in future.
This does not help first layout at all, even if we had the capability. UI being responsive on white screen does not look very helpful to me.

7. Punch a hole to sandbox

Mac sandbox has a feature to punch holes for specific files/directories. Blink for Mac uses this feature as Mac Core Foundation API opens and reads dictionary files. WebKit does the same. This option isn't feasible since, as far as I understand, Linux/Win sandbox does not have this feature.

Other options?

Summary of Options

Options	Pros	Cons
1. Ship	• Makes users happy, [launch bug] starred by 176 users. • We get more real data on the property usage. • Catch up with other browsers.	• Pages with the "hyphens" property can be unexpectedly slower.
2. Improve IPC	Currently not feasible.	
3. Request in style recalc	• Maximize the parallel time. • Renderer is faster.	• Adds small cost to style recalc? • Doesn't make renderer faster if the task is fast but IPC is not on high load. • Not sure if this is feasible, see the discussion above.
4. Open at startup	• Renderer is faster. • File open cost still exists, but IPC cost is gone.	• Process starts slower even when hyphenation is not used at all. • It's unlikely to use all dictionaries.
5. Async	Currently not feasible.	
6. Interrupt/yield	Currently not feasible.	
7. Punch a hole to sandbox	Currently not feasible.	

Recommendations

- Add some more metrics:
  - Time to open files in addition to overall IPC time. This helps to investigate option 2 and 3.
  - Count when the hyphenation language is not the system language. This helps to investigate option 4.
- Ship. This will give us the real data only on pages with ‘hyphens’ property set.
- Continue investigations on:
  - “3. Request in style recalc” can make the performance better if it’s done properly.
  - “4. Open at startup” can make the performance better if it’s done properly.
  - “7. Punch a hole to sandbox”.