**开平主应用CRP优化 - prefetch xhr api**

**业务背景**

|  |  |
| --- | --- |
| |  | | --- | | 在资源加载同时发起请求server接口，提早获取server接口的数据 |   开平挂载菜单之前需要依赖几个server remote的数据，这些依赖的数据会阻塞菜单内容的渲染，所以应该提早发起请求，并行请求js资源请求和用户数据。 |

路由挂载链路有权限管理，需要游戏认证后才可以请求请求menu接口，这个串行逻辑不能去掉，有很多业务需要保证user/permission必须要有值，使用prefetch的方法提早请求CRP被阻塞的接口。

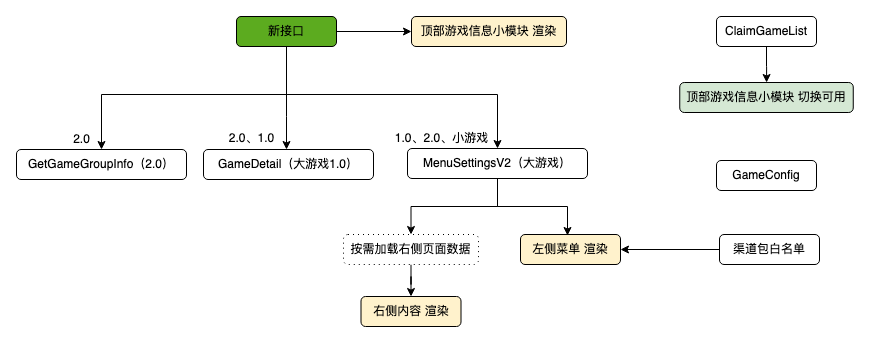
整理开平主应用挂载路由所依赖的接口，prefetch 需要用到的接口，提前prefetch 菜单配置，提前渲染router。

权限获取全部prefetch

1. /open\_platform/perm/role/info
2. /open\_platform/perm/resource\_key\_map

大小游戏

1. /open\_platform/union/v3/game\_open\_platform/info
2. /open\_platform/menu/settings/v2



**实现**

**直接inject html**

将几个纯数据请求的接口在html上写死，等资源加载结束后读window.x变量，获取结果。如果有结果就是成功优化，没有结果就是不赚不亏。

|  |
| --- |
| HTML <script defer>  function prefetchApi({url, hoistField, cb}) {  var resp = null;  var xhr = new XMLHttpRequest();  xhr.open('GET', '//'+window.location.host + url, true);  xhr.onreadystatechange = function (e) {  if (xhr.readyState === XMLHttpRequest.DONE || xhr.readyState === 4) {  const status = xhr.status  if (status === 0 || (status >= 200 && status < 400)) {  try {  const result = JSON.parse(xhr.responseText)  resp = result  } catch (e) {  console.error(e)  }  } else {  console.error(xhr.responseText)  }  window[hoistField] = resp  if (cb) {  cb(xhr);  }  }  }  xhr.send();  }   const hoistUserInfoKey = 'PRE\_USER\_INFO'  const hoistResourceKeyMap = 'PRE\_RESOURCE\_KEY\_MAP'   // init  prefetchApi({  url: '/open\_platform/user/info?subject\_aid=3542',  hoistField: hoistUserInfoKey,  cb(xhr) {  const headers = xhr.getAllResponseHeaders();  window.HAS\_PPE\_HEADER = headers.includes('x-user-ppe');   const resp = window[hoistUserInfoKey];  const role\_id = resp && resp.data && resp.data.role\_id;  const hoistRoleInfoKey = 'PRE\_ROLE\_INFO-'+role\_id   if (typeof role\_id !== 'undefined') {  prefetchApi({  url: '/open\_platform/perm/role/info?subject\_aid=3542&role\_id='+role\_id,  hoistField: hoistRoleInfoKey,  })  }  }  })  prefetchApi({  url: '/open\_platform/perm/resource\_key\_map?subject\_aid=3542',  hoistField: hoistResourceKeyMap,  })   // product\_management  if (location.pathname.includes('/product\_management/')) {  const [\_, \_\_, gameId] = location.pathname.match(/(\/product\_management\/)([^\\/]+)/)  const hoistGameInfoKey = 'PRE\_GAME\_INFO-'+gameId   // game info  prefetchApi({  url: '/open\_platform/union/v3/game\_open\_platform/info?subject\_aid=3542&unified\_game\_id='+gameId,  hoistField: hoistGameInfoKey,  })  } </script> |

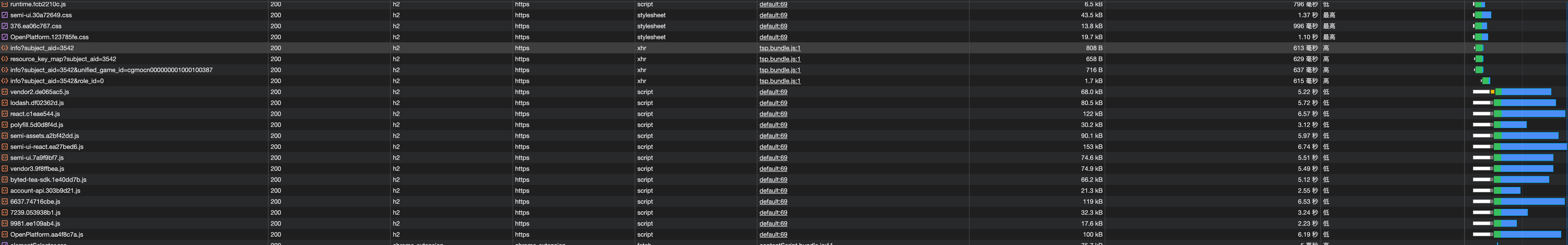
**去掉单点依赖**

但是问题就是如果代码中的请求接口变了后，prefetch忘记改了，就会导致prefetch失效，这里需要用一些编译时的手段解决。在匹配.prefetch.js文件将其内容插入html script标签，提早执行api加载逻辑。

|  |
| --- |
| JavaScript const { createUnplugin } = require('unplugin') const fs = require('fs'); const path = require('path'); const HtmlWebpackPlugin = require('html-webpack-plugin'); const terser = require("terser"); const babel = require('@babel/core'); const presetEnv = require('@babel/preset-env');  const prefetchRequestRE = /(?:\.)prefetch\.js$/ const postfixRE = /[?#].\*$/s const commentRE = /\/\\*[\s\S]\*?\\*\/|\/\/.\*$/gm  function cleanUrl(url) {  return url.replace(postfixRE, '') }  const preHandleCode = content => {  // minifiy and iife wrap script  const iifeContent = `(() => {${content}})()`  const polyfill = babel.transformSync(iifeContent, {  ast: false,  babelrc: false,  configFile: false,  compact: false,  presets: [  presetEnv.default,  ]  }).code  return terser.minify(polyfill, {  mangle: {  toplevel: true,  },  nameCache: {}  }).code; }  const innerHTMLScript = content => {  return {  tagName: 'script',  innerHTML: preHandleCode(content),  attributes: {  }  } };  const prefetchHelperCode = preHandleCode(fs.readFileSync(path.join(\_\_dirname, './runtime.js')));  /\*\*  \* webpack组件 会把这些代码抄一份到Html上提前执行 \*/ module.exports = createUnplugin(() => {  const pluginName = 'prefetch-virtual-module';  const collectCode = {}  return {  name: pluginName,  webpack(compiler) {  compiler.hooks.compilation.tap(  pluginName,  compilation => {  HtmlWebpackPlugin.getHooks(compilation).alterAssetTags.tap(  pluginName,  (htmlPluginData) => {  const codes = Object.values(collectCode);  console.log('[collectCode]', codes.length);  htmlPluginData.assetTags.scripts = [innerHTMLScript(prefetchHelperCode)]  .concat(codes.map(el => innerHTMLScript(el)))  .concat(htmlPluginData.assetTags.scripts);   return htmlPluginData;  }  )  }  )  },  loadInclude(id) {  return prefetchRequestRE.test(id);  },  load(id) {  console.log('[PrefetchVirtualModule load]', id);  const code = fs.readFileSync(id, { encoding: 'utf-8' });  const cleanCode = code  .replace(/export\s+default/gm, '')  .replace(/export\s+/gm, '')  .replace(commentRE, '')  .trim();   if (cleanCode.includes('import')) {  console.log('[.prefetch.js]', cleanCode);  this.error('[.prefetch.js]文件需要html script标签直接执行，来进行预处理，所以不可以有import词组出现')  return code;  }   collectCode[id] = `  ${cleanCode}   if (bootstrap) {  bootstrap();  }  `   return code;  }  } }) |

**加载优先级问题**

使用了prefetch发现请求的优先级有问题。  
加载的瀑布流被prefetch api的xhr打歪了，因为<script defer的优先级没有xhr的优先级高。



这时候就得把script的加载优先级提前，至少要和xhr平级。把script的加载和运行分开，使用<link rel="preload" 去加载script标签，获得同等的加载优先级，排平瀑布流。

这里改写了一下@vue/preload-webpack-plugin让preload的节点插入符合预期。



|  |
| --- |
| JavaScript const PreloadWebpackPlugin = require('@vue/preload-webpack-plugin'); const HtmlWebpackPlugin = require('html-webpack-plugin');  // https://github.com/vuejs/preload-webpack-plugin module.exports = class PreloadScriptWebpackPlugin extends PreloadWebpackPlugin {  constructor (options) {  super(options);  }   apply (compiler) {  const skip = data => {  const htmlFilename = data.plugin.options.filename  const exclude = this.options.excludeHtmlNames  const include = this.options.includeHtmlNames  return (  (include && !(include.includes(htmlFilename))) ||  (exclude && exclude.includes(htmlFilename))  )  }   compiler.hooks.compilation.tap(  this.constructor.name,  compilation => {  HtmlWebpackPlugin.getHooks(compilation).beforeAssetTagGeneration.tapAsync(  this.constructor.name,  (htmlPluginData, callback) => {  if (skip(htmlPluginData)) {  callback()  return  }  this.generateLinks(compilation, htmlPluginData)  callback()  }  )   HtmlWebpackPlugin.getHooks(compilation).alterAssetTags.tap(  this.constructor.name,  (htmlPluginData) => {  if (skip(htmlPluginData)) {  return  }  if (this.resourceHints) {  htmlPluginData.assetTags.scripts = [  ...this.resourceHints,  ...htmlPluginData.assetTags.scripts  ]  }  return htmlPluginData  }  )  }  )  } } |

**结论**

周纬度性能曲线

