



构建性能：持久化缓存

构建效率方法：

- 不重复干：缓存
- 一块干：并行处理
- 快干：提高效率、高效编译器、编译路径
- 少干：减少步骤、缩减范围

在开发过程中，如果轻微的修改就需要运行全部的打包过程显然是一种巨大的浪费。

使用缓存技术，存储中间产物就成了最有效的优化手段。

- 将首次构建过程持久化到文件系统中
- 在下次执行是跳过解析、链接、编译过程

webpack 中启动缓存

webpack.config.js

```
cache: {  
  type: "filesystem",  
  profile: true,  
},
```

JavaScript

对比

第 1 列

```
../node_modules/.pnpm/registry.npmirror.com+css-loader@6.7.1_webpack@5.73.0/node_modules/css-loader/dist/runtime/noSourceMaps.js 64 bytes [built] [code generated]
771 ms -> 84 ms -> 64 ms -> 8 ms -> 1139 ms ->
0 ms (resolving: 0 ms, restoring: 0 ms, integration: 0 ms, building: 0 ms, storing: 0 ms)
+ 2 modules
+ 19 modules

LOG from webpack.Compilation.ModuleProfile
<w> | | 759 ms build modules > ./src/main.ts
<w> | 763 ms build modules > 4 x javascript/auto with ../node_modules/.pnpm/registry.npmirror.com+ts-loader@9.3.1_vxwqrucgsi6fv2vqgtti3vbvaa/node_modules/ts-loader/index.js??clonedRuleSet-1.use[0]
<w> 1493 ms build modules
+ 56 hidden lines

webpack 5.73.0 compiled successfully in 2855 ms
+ admin-webpack git:(main) * pnpm build

> admin-webpack@1.0.0 build /Users/josephxia/source/smarty-admin/packages/admin
```

第 2 列

```
assets by path js/*.js 1.63 MiB
asset js/main.js 1.5 MiB [emitted] (name: main)
asset js/src_pages_dashboard_index_vue.js 64.7 KiB [emitted]
asset js/src_pages_login_vue.js 47.3 KiB [emitted]
asset js/src_pages_react_vue.js 14.1 KiB [emitted]
asset js/src_pages_vue_vue.js 6.82 KiB [emitted]
assets by path *.png 1.59 MiB
asset b69ba4db76ec7d68f9f6.png 1.21 MiB [emitted] [immutable] [from: src/assets/login/bg.png]
asset 0da9ba094b4747daa1cab.png 284 KiB [emitted] [immutable] [from: src/assets/login/gao.png]
asset d21a4b9edc288bd7c22f.png 80.9 KiB [emitted] [immutable] [from: src/assets/login/cunzhang.png]
asset 88ccd17efc924bf83304.png 20.4 KiB [emitted] [immutable] [from: src/assets/login/ranshu.png] (auxiliary name: main)
asset index.html 193 bytes [emitted]
cached modules 1.17 MiB (javascript) 1.59 MiB (asset) 7.06 KiB (runtime) [cached] 134 modules
webpack 5.73.0 compiled successfully in 212 ms
+ admin-webpack git:(main) * []
```

常用参数

- `cache.type`：缓存类型，支持 `'memory' | 'filesystem'`，需要设置为 `filesystem` 才能开启持久缓存；
- `cache.cacheDirectory`：缓存文件路径，默认为 `node_modules/.cache/webpack`；
- `cache.buildDependencies`：额外的依赖文件，当这些文件内容发生变化时，缓存会完全失效而执行完整的编译构建，通常可设置为各种配置文件
- `cache.managedPaths`：受控目录，Webpack 构建时会跳过新旧代码哈希值与时间戳的对比，直接使用缓存副本，默认值为 `['./node_modules']`；
- `cache.profile`：是否输出缓存处理过程的详细日志，默认为 `false`；
- `cache.maxAge`：缓存失效时间，默认值为 `5184000000`。

service/build.js

```
"use strict";

let base = require("../config/base");
const cache = require("../config/cache");
let config = base;
config.mode = "production";

module.exports = config;
```

TypeScript

service/dev.js

JavaScript

```
"use strict";

let base = require("./config/base");
const cache = require("./config/cache");
let config = base;
config.mode = "development";

module.exports = config;
```

config/cache.js

JavaScript

```
module.exports = (base) => {
  Object.assign(base, {
    cache: {
      type: "filesystem",
      profile: true,
    },
  });

  return base;
};
```

config/speedMeasure.js

JavaScript

```
const SpeedMeasurePlugin = require("speed-measure-webpack-plugin");
const smp = new SpeedMeasurePlugin();
module.exports = (base) => {
  // 屏蔽VueLoader与speedMeasure不兼容的问题
  loader = base.plugins[0];
  base.plugins.shift();
```

```
base = smp.wrap(base);
base.plugins.unshift(loader);
return base;
};
```

package.json

```
"scripts": {
  "build": "webpack --config ./service/build.js",
  "dev": "webpack serve --progress --config ./service/dev.js"
},
```

JSON