构建性能： 并行处理

并行处理

- HappyPack： 多进程运行 Loader （停止维护）
- Thread-loader： 多进程运行资源加载 (官方驱动)
- Parallel-Webpack： 多进程运行多个 Webpack 构建实例 （多入口场景）
- TerserWebpackPlugin： 支持多进程方式执行压缩丑化

HappyPack

- 作者已经明确表示不会继续维护，扩展性与稳定性缺乏保障，随着 Webpack 本身的发展迭代，可以预见总有一天 HappyPack 无法完全兼容 Webpack；

Bash

```bash
pnpm i happypack -D
```

service/happypack.js

Bash

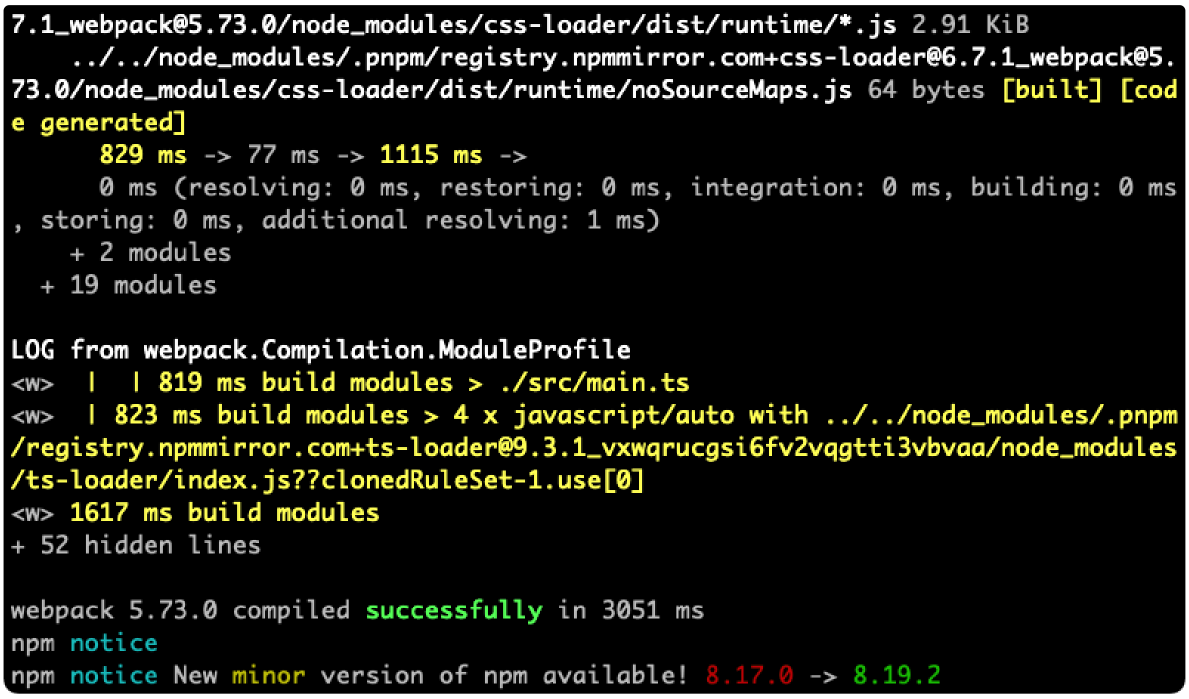
```bash
const HappyPack = require("happypack");

module.exports = (base) => {
  let loader = base.module.rules.find((v) => v.test.toString() === "/\\.css

  Object.assign(loader, {
    test: /\.css?$/,
    exclude: /node_modules/,
    // 使用 `id` 参数标识该 Loader 对应的 HappyPack 插件示例
    use: "happypack/loader?id=css",
  });
  base.plugins.push(
    new HappyPack({
      // 注意这里要明确提供 id 属性
      id: "css",
      loaders: ["style-loader", "css-loader", "postcss-loader"],
    })
  );

  return base;
};
```

使用前

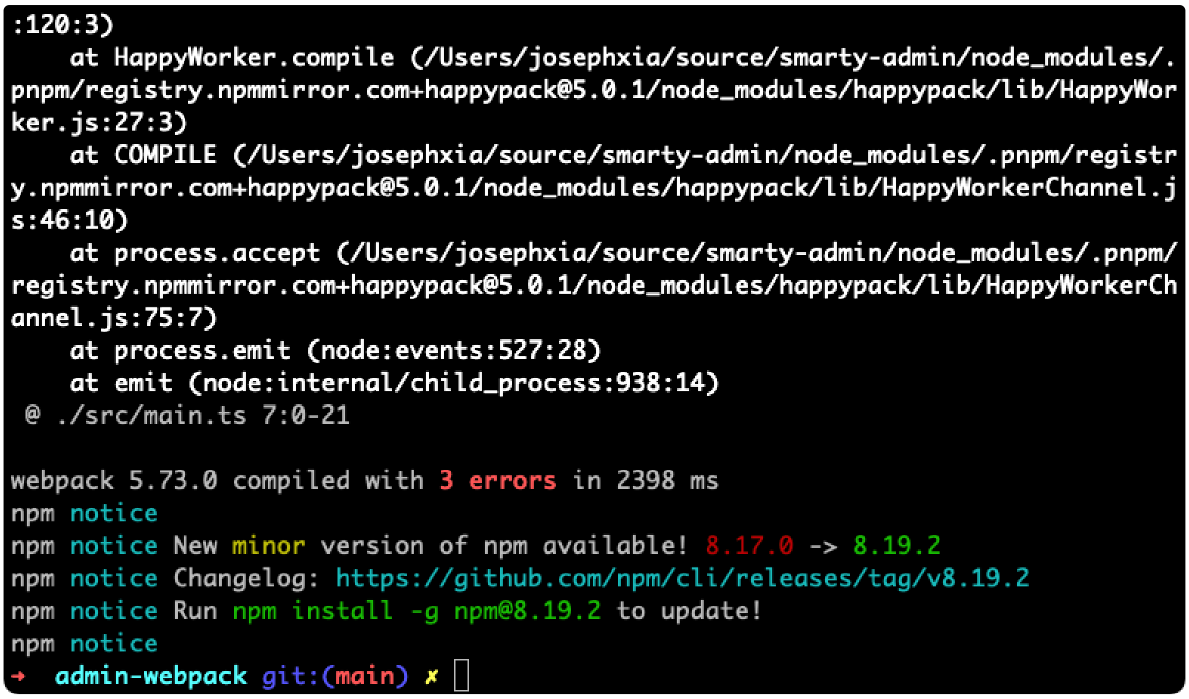


```
7.1_webpack@5.73.0/node_modules/css-loader/dist/runtime/*.js 2.91 KiB
    ../../node_modules/.pnpm/registry.npmmirror.com+css-loader@6.7.1_webpack@5.
73.0/node_modules/css-loader/dist/runtime/noSourceMaps.js 64 bytes [built] [cod
e generated]
      829 ms -> 77 ms -> 1115 ms ->
      0 ms (resolving: 0 ms, restoring: 0 ms, integration: 0 ms, building: 0 ms
, storing: 0 ms, additional resolving: 1 ms)
    + 2 modules
  + 19 modules

LOG from webpack.Compilation.ModuleProfile
<w>  |  | 819 ms build modules > ./src/main.ts
<w>  | 823 ms build modules > 4 x javascript/auto with ../../node_modules/.pnpm
/registry.npmmirror.com+ts-loader@9.3.1_vxwqrucgsi6fv2vqgtti3vbvaa/node_modules
/ts-loader/index.js??clonedRuleSet-1.use[0]
<w> 1617 ms build modules
+ 52 hidden lines

webpack 5.73.0 compiled successfully in 3051 ms
npm notice
npm notice New minor version of npm available! 8.17.0 -> 8.19.2
```

使用后


```
:120:3)
    at HappyWorker.compile (/Users/josephxia/source/smarty-admin/node_modules/.
pnpm/registry.npmmirror.com+happypack@5.0.1/node_modules/happypack/lib/HappyWor
ker.js:27:3)
    at COMPILE (/Users/josephxia/source/smarty-admin/node_modules/.pnpm/registr
y.npmmirror.com+happypack@5.0.1/node_modules/happypack/lib/HappyWorkerChannel.j
s:46:10)
    at process.accept (/Users/josephxia/source/smarty-admin/node_modules/.pnpm/
registry.npmmirror.com+happypack@5.0.1/node_modules/happypack/lib/HappyWorkerCh
annel.js:75:7)
    at process.emit (node:events:527:28)
    at emit (node:internal/child_process:938:14)
 @ ./src/main.ts 7:0-21

webpack 5.73.0 compiled with 3 errors in 2398 ms
npm notice
npm notice New minor version of npm available! 8.17.0 -> 8.19.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.19.2
npm notice Run npm install -g npm@8.19.2 to update!
npm notice
➜  admin-webpack git:(main) ✗ 
```

开启 ts-loader 的 Happypack 模式

```JavaScript
const HappyPack = require("happypack");

module.exports = (base) => {
  loader = base.module.rules.find((v) => v.test.toString() === "/\\.tsx?$/"
  Object.assign(loader, {
    test: /\.tsx?$/,
    exclude: /node_modules/,
    use: [
      "babel-loader",
      {
        loader: "ts-loader",
        options: {
          transpileOnly: true, // 关闭项目运行时的类型检查
          appendTsSuffixTo: ["\\.vue$"], // 给 .vue文件添加个 .ts后缀用于编译。
          happyPackMode: true,
        },
      },
    ],
  });

  return base;
};
```

Thread-loader

- Thread-loader 由 Webpack 官方提供，目前还处于持续迭代维护状态，理论上更可靠；
- 创建与销毁进程带来性能问题可能会造成反向优化
- 对很多 Loader 有不兼容情况需要自己甄别

```TypeScript
pnpm i thread-loader -D
```

threadLoader.js

```Bash
const ThreadLoader = require("thread-loader");
ThreadLoader.warmup(
  {
    // 可传入上述 thread-loader 参数
    workers: 4,
    workerParallelJobs: 50,
  },
  [
    // 子进程中需要预加载的 node 模块
    "vue-loader",
  ]
);
module.exports = (base) => {
  let loader = base.module.rules.find((v) => v.test.toString() === "/\\.vue

  Object.assign(loader, {
    test: /\.vue$/,
    use: [
      //   {
      //     loader: "thread-loader",
      //   },
      "vue-loader",
    ],
  });

  return base;
};
```

反向优化

Parallel–Webpack

- 多进程方式运行 Webpack
- 针对多 entry 场景
- 缺点进程间无法通讯，所有资源编译是重复的

JavaScript
```javascript
cluster.fork()
childprocess()
```

并行压缩

- 可以认为是组件内部提供的并行处理能力

terser.js

JavaScript
```javascript
const TerserPlugin = require("terser-webpack-plugin");

// 获取cpu
const os = require("os");
const cpuNum = os.cpus().length;
module.exports = (base) => {
  base.optimization = {
    minimize: true,
    minimizer: [
      new TerserPlugin({
        parallel: cpuNum, // number | boolean
      }),
    ],
  };
  return base;
};
```

适用场景

- 对于 Webpack4 之前的项目，可以使用 HappyPack 实现并行文件加载；
- Webpack4 之后则建议使用 Thread–loader；
- 多实例并行构建场景建议使用 Parallel–Webpack 实现并行；
- 生产环境下还可配合 `terser-webpack-plugin` 的并行压缩功能，提升整体效率。