The future of Rspack

Although Rspack already meets the needs of many projects, there is still a big gap to reach the full capabilities of webpack. Prioritization will be based on community feedback, so please tell us about your requirements!

Collaboration with community partners

We are very willing to provide support to framework teams and toolchains within the community to unleash the true performance advantages of Rspack. If your framework or toolchain has a demand for high-performance build engines, let us know!

Enhancing plugin capabilities

Rspack already implements the basic `Loader` interface and a small number of webpack plugin APIs. Although our goal is not to achieve 100% compatibility for plugin APIs, we will try our best to implement the mainstream requirements based on community feedback. At the same time, we are also exploring higher-performance plugin communication solutions to reduce the cost of plugin communication, thereby ensuring more plugin APIs can be implemented.

Continuously improving performance

Performance is the core selling point and focus of Rspack development. In the future we'll explore higher-performance concurrent/multi-core-friendly algorithms, higher-performance caching solutions, higher-performance plugin communication solutions, etc.

Expanding the test suite

Today Rspack is primarily tested using a subset of webpack's test cases. In the future, we'll cover more of these tests, while also expanding the test suite and including community projects to ensure compatibility across Rspack releases.