Performance

Tuning

Use `M-x lsp-doctor` to validate if your `lsp-mode` is properly configured. In the section below, you could find description for each of the checks:

When configured properly `lsp-mode`'s performance is on par with mainstream LSP clients (e. g. `VScode`, `Theia`, etc). Here are steps to achieve optimal results.

JSON native serialization/deserialization

- Use Emacs 27+ with native json support. (Note: this requires that you have libjansson installed, and that emacs was compiled with `–with-json` passed to `./configure`.) You can check your installation for native json support by running `M-:` `(functionp 'json-serialize)` `RET`. Benchmarks show that Emacs 27 is `~15 times` faster than Emacs when using Elisp json parser implementation.

Adjust `gc-cons-threshold`

The default setting is too low for `lsp-mode`'s needs due to the fact that client/server communication generates a lot of memory/garbage. You have two options:

~~~
- Set it to big number(100mb) like most of the popular starter kits like
Spacemacs/Doom/Prelude, etc do:

```elisp
(setq gc-cons-threshold 100000000)
```

- Follow the method recommended by Gnu Emacs Maintainer Eli Zaretskii: "My
suggestion is to repeatedly multiply gc-cons-threshold by 2 until you stop
seeing significant improvements in responsiveness, and in any case not to
increase by a factor larger than 100 or somesuch. If even a 100-fold increase
doesn't help, there's some deeper problem with the Lisp code which produces so
much garbage, or maybe GC is not the reason for slowdown." Source:
<https://www.reddit.com/r/emacs/comments/brc05y/is_lspmode_too_slow_to_use_for_an
~~~

Increase the amount of data which Emacs reads from the process

Again the emacs default is too low 4k considering that the some of the language server responses are in 800k - 3M range.

```
(setq read-process-output-max (* 1024 1024)) ;; 1mb
```

Use `plists` for deserialization.

`lsp-mode` can be compiled in 2 modes `plist` and `hash-table` based `lsp-use-plists` flag. `plist`s provide better performance in deserialization and also put less presure than `hash-table`s. To switch to `plist` you have to perform 2 steps:

1. Configure the following env variable. Make sure that `Emacs` can see that variable. For example, this can be done by starting `Emacs` from the shell.

   ```
   export LSP_USE_PLISTS=true
   ```

   or by setting it in `early-init.el`:

   ```
   (setenv "LSP_USE_PLISTS" "true")
   ```

2. Delete `lsp-mode` related packages. This can be done with `package-delete`.
3. Make sure that `lsp-use-plists` is non-nil.
4. Restart `Emacs` and install again `lsp-mode` related packages.

NB: make sure that `lsp-use-plists` does not change after you compile the file. Furthermore, if you are using something like `exec-path-from-shell` you'll need to make sure to add `LSP_USE_PLISTS` to `exec-path-from-shell-variables`.

Optional steps

- Optional: Disable `lsp-ui`. Normally, `lsp-ui` is very fast but in some systems (especially when using `Windows`) `lsp-ui` overlays and popups might slow down emacs.
- Optional: fine-tune `lsp-idle-delay`. This variable determines how often lsp-mode will refresh the highlights, lenses, links, etc while you type.

```
(setq lsp-idle-delay 0.500)
```

Using Emacs 28.1 or later

Emacs 28.1 includes "native compilation" of elisp code (changelog). For optimal performance, using Emacs 28.1 or later (with native compilation enabled) is recommended.

Ignore watch folders/files

If the server supports watch files, by default `lsp-mode` tries to watch all files and folders of the project ignoring the regexp from `lsp-file-watch-ignored`. If you don't want some file or folder to be watched for performance reasons, you can add a regexp to that variable excluding the file or folder.

Check the file watchers section for details.

Check if logging is switched off.

Make sure `lsp-log-io` is `nil`. You might have forgotten it after a debugging session, for example. It can cause a great performance hit.

```
(setq lsp-log-io nil) ; if set to true can cause a performance hit
```

Sometimes you might need to check logging for specific LSP server configuration as well, i.e. for `lsp-eslint` it is: `lsp-eslint-trace-server`.

Reporting performance problems

If you have tried all of the non-optional steps from the list and `emacs` is still not very responsive please open a PR with the following information:

- Include emacs performance report. Use the following step to collect it:
- `M-x profiler-start` and select `CPU`
- Reproduce the slow behavior.
- `M-x profiler-stop`
- `M-x profiler-report` to create a report
- In the profiler report expand all nodes by doing `C-u TAB`.

Note: - `lsp-mode` is just a frontend and the performance depends on server as well. Some servers (e. g. Palantir's Python Language Server) might be slow when performing auto-completion.

Last update: December 2, 2024