Painless Transition to Portable Dumper

Emacs 27 is coming with many exciting new features including the portable dumper. There has been attempts to use portable dumper to speed up Emacs startup time. I know Spacemacs does this from a long time ago [1]. But I couldn't find any literature on using portable dumper for one's own init.el. Here I record my take on adopting portable dumper, including gotcha's I found, the general design I use, and some fixes, hacks and tricks I used.

With portable dumper, my startup time reduced from 2.47s to 0.76s (3x). (Everybody measures their startup differently. As a common measure, esup gives 0.38s) This is on Mac, it should be even faster on Linux. Even better, all it takes are ~30 lines of code, and the startup without a dump file works like before.

Table of Contents

- 1. General Design
- 2. Gotcha's
- 3. Tricks
- 4. Final notes
- 5. Some fixes and hacks
- 6. Other speedup tricks

<2020-01-27 Mon>
Note: Eli says bug-free dumping of custom emacs is not a goal for Emacs 27[2]. However, dumping only packages, selectively, works fine for me.

1 General Design

Start a vanilla Emacs, load packages, dump the image out. Then you start Emacs with this dump file. The point is to speed up packages that you can't autoload — those you want immediately after startup. For example, company, ivy/helm, which-key, use-package, themes, highlight-parentheses. Other parts of init doesn't change.

I create a init file for the dump process, `~/.emacs.d/dump.el`, and dump with

```
emacs --batch -q -l ~/.emacs.d/dump.el
```

Once dumped, I can start Emacs with the dump file [3] (use root path, not `~`!):

```
emacs --dump-file="/Users/yuan/.emacs.d/emacs.pdmp"
```

A minimal `dump.el`:

```
(require 'package)
;; load autoload files and populate load-path's
(package-initialize)
;; (package-initialize) doens't require each package, we need to loa
;; those we want manually
(dolist (package '(use-package company ivy counsel org helpful
                   general helpful use-package general which-key
                   recentf-ext swiper ivy-prescient find-char
                   aggressive-indent windman doom-themes winner
```

```
                    elec-pair doom-one-light-theme
                    doom-cyberpunk-theme rainbow-delimiters
                    highlight-parentheses hl-todo buffer-move
                    savehist eyebrowse minions ws-butler
                    expand-region isolate outshine flyspell magit
                    eglot))
  (require package))
;; dump image
(dump-emacs-portable "~/.emacs.d/emacs.pdmp")
```

Now let's extend this minimal configuration with fixes and enhancements.

2 Gotcha's

So it seems trivial: I `(package-initialize)` and `(require)` every packages in `dump.el`, and everything works, except that it doesn't. For one, `load-path` is not stored in the dump image [4]. You need to store `load-path` in another variable.

```
;; in dump.el
(package-initialize)
(setq luna-dumped-load-path load-path)
...
(dump-emacs-portable "~/.emacs.d/emacs.pdmp")

;; in init.el
(setq load-path luna-dumped-load-path)
```

Second, when you start Emacs with a dump file, some default modes are not enabled:

- `transient-mark-mode`
- `global-font-lock-mode`

And `global-undo-tree-mode` makes Emacs segfault during dumping (didn't verify, Spacemacs says so, but why would you enable it when dumping anyway?) Spacemacs also says `winner-mode` and `global-undo-tree` mode doesn't live through dumping, but I enable them in my `init.el`, not in dump, so that doesn't affect me.

The fix is simple, have something like this in `init.el`:

```
(global-font-lock-mode)
(transient-mark-mode)
```

Third, you can't use `~` in the `--dump-file` command line flag. Otherwise Emacs complains about cannot open dump file. So don't use `~/.emacs.d/emacs.pdmp`. The dump file loads in very early stage, many variables are not known yet, so `~` won't expand.

Fourth, scratch buffer behaves differently when Emacs starts with a dump file. Re-run mode hook seems to fix it:

```
(add-hook 'after-init-hook
                (lambda ()
                  (save-excursion
                    (switch-to-buffer "*scratch*")
                    (lisp-interaction-mode))))
```

<2020-01-27 Mon>
As a side note (kindly provided by Damien Cassou), (a relatively new version of) magit uses dynamic modules, which is not dumpable. So don't require magit in your dump. The portable dumper doesn't dump window configurations either, but since I'm dumping only the packages, it doesn't annoy me.

Now the `dump.el` is:

```
(require 'package)
;; load autoload files and populate load-path's
(package-initialize)
;; store load-path
(setq luna-dumped-load-path load-path)
;; (package-initialize) doens't require each package, we need to loa
;; those we want manually
(dolist (package '(use-package company ivy counsel org helpful
                    general helpful use-package general which-key
                    recentf-ext swiper ivy-prescient find-char
                    aggressive-indent windman doom-themes winner
                    elec-pair doom-one-light-theme
                    doom-cyberpunk-theme rainbow-delimiters
                    highlight-parentheses hl-todo buffer-move
                    savehist eyebrowse minions ws-butler
                    expand-region isolate outshine flyspell magit
                    eglot))
  (require package))
;; dump image
(dump-emacs-portable "xxx")
```

`init.el`:

```
(global-font-lock-mode)
(transient-mark-mode)
(add-hook 'after-init-hook
                (lambda ()
                  (save-excursion
                    (switch-to-buffer "*scratch*")
                    (lisp-interaction-mode))))
```

3 Tricks

3.1 Keep non-dump-file startup working as before

I want my configuration to still work without a dump file. This is what I do:

```
;; in init.el
(defvar luna-dumped nil
  "non-nil when a dump file is loaded (because dump.el sets this var

(defmacro luna-if-dump (then &rest else)
  "Evaluate IF if running with a dump file, else evaluate ELSE."
  (declare (indent 1))
  `(if luna-dumped
       ,then
     ,@else))

;; in dump.el
(setq luna-dumped t)
```

And I use the `luna-if-dump` in `init.el` at where two startup process differs:

```
(luna-if-dump
    (progn
      (setq load-path luna-dumped-load-path)
      (global-font-lock-mode)
      (transient-mark-mode)
      (add-hook 'after-init-hook
                (lambda ()
                  (save-excursion
                    (switch-to-buffer "*scratch*")
                    (lisp-interaction-mode)))))
  ;; add load-path's and load autoload files
  (package-initialize))
```

In a dump-file-startup, we don't need to `(package-initialize)` because it's done during dumping, but we need to load `load-path` and fix other gotcha's.

3.2 Dump packages selectively

Blindly dumping every package is a recipe for weird errors. I only dump those I want immediately on startup (company, ivy/helm) and those are big (org). Not that dumping everything doesn't work, but it takes more energy to get everything right.

3.3 Dumping themes greatly speeds things up

When profiling my startup with esup, I found Emacs spends 70% of the time loading the theme.

```
Total User Startup Time: 1.063sec     Total Number of GC Pauses: 21

doom-one-light-theme.el:5  0.755sec   71%
(def-doom-theme doom-one-light
"A light theme inspired by Atom One"
...
```

Dumping themes is not as simple as adding `(load-theme theme)` to `dump.el`, if you do that, Emacs complains and doesn't load the theme. I guess that's because it's in batch mode. Instead, require your themes like other libraries and loads them *without enabling them*.

```
;; in dump.el
(require 'doom-themes)
(require 'doom-one-light-theme)
;; the two flags are no-confirm and no-enable
(load-theme 'doom-one-light-theme t t)
```

In `init.el`, we enable the theme, instead of loading it. Unlike require, `load-theme` doesn't check if the theme is already loaded.

```
;; in init.el
(when window-system
  (luna-if-dump
      (enable-theme 'doom-one-light)
    (load-theme 'doom-one-light)))
```

And the speed up is significant.

```
...
init.el:87  0.034sec   7%
(when window-system
(luna-if-dump
(enable-theme 'doom-one-light)
```

```
(luna-load-theme nil t)))
...
```

3.4 Complete example `dump.el` & `init.el`

With everything I just talked about:

`dump.el`:

```
(require 'package)
;; load autoload files and populate load-path's
(package-initialize)
;; store load-path
(setq luna-dumped-load-path load-path
      luna-dumped t)
;; (package-initialize) doens't require each package, we need to loa
;; those we want manually
(dolist (package '(use-package company ivy counsel org helpful
                    general helpful use-package general which-key
                    recentf-ext swiper ivy-prescient find-char
                    aggressive-indent windman doom-themes winner
                    elec-pair doom-one-light-theme
                    doom-cyberpunk-theme rainbow-delimiters
                    highlight-parentheses hl-todo buffer-move
                    savehist eyebrowse minions ws-butler
                    expand-region isolate outshine flyspell magit
                    eglot))
  (require package))
;; pre-load themes
(load-theme 'doom-one-light-theme t t)
(load-theme 'doom-cyberpunk-theme t t)
;; dump image
(dump-emacs-portable "~/.emacs.d/emacs.pdmp")
```

`init.el`:

```
(luna-if-dump
    (progn
      (setq load-path luna-dumped-load-path)
      (global-font-lock-mode)
      (transient-mark-mode)
      (add-hook 'after-init-hook
                (lambda ()
                  (save-excursion
                    (switch-to-buffer "*scratch*")
                    (lisp-interaction-mode)))))
  ;; add load-path's and load autoload files
  (package-initialize))
;; load theme
(when window-system
  (luna-if-dump
      (enable-theme 'doom-one-light)
    (luna-load-theme)))
```

After everything works, I wrapped dump file's path with variables and added `defvar` for variables I introduced, and did other irrelevant stuff.

(Update <2020-03-08 Sun>) I forgot to mention how I dump Emacs from within Emacs:

```
(defun luna-dump ()
  "Dump Emacs."
  (interactive)
  (let ((buf "*dump process*"))
    (make-process
     :name "dump"
     :buffer buf
     :command (list "emacs" "--batch" "-q"
                    "-l" (expand-file-name "dump.el"
```

```
                                        user-emacs-directory)))
    (display-buffer buf)))
```

4 Final notes

You can be more aggressive and dump *all* packages and init files. But 1) since current approach is fast enough, the marginal benefit you get hardly justifies the effort; 2) if you dump your init files, you need to re-dump every time you change your configuration. Oh, and there are a bunch of Lisp objects that cannot be dumped, e.g., window configuration. Just think about the work needed to handle those in your init files. If you really care that much about speed, Dark Side is always awaiting.

5 Some fixes and hacks

Here I record some problems I encountered that's not related to dumping.

5.1 recentf-ext

When dumping recentf-ext, I found some problems and changed two places in `recentf-ext.el`. It has a `(recentf-mode 1)` as a top level form. That means `recentf-mode` enables whenever `recentf-ext.el` loads. Not good. I removed it. It also has a line requiring for `cl`, although it didn't use it, I removed that as well. My fork is at here.

5.2 Use esup with dump file

esup is a great way to see what package is taking most time in startup. It helps me to find what package to dump. However, esup doesn't support loading dump files, and we need to modify it a bit. We also want to know if we are in esup child process, so we don't start an emacs server (and do other things differently, depends on your configuration). Go to `esup` in `esup.el` (by `find-library`), and change the `process-args`:

```
("*esup-child*"
 "*esup-child*"
 ,esup-emacs-path
 ,@args
 "-q"
 "-L" ,esup-load-path
 "-l" "esup-child"
 ;; ++++++++++++++++++++++++++++++++++++++++
 "--dump-file=/Users/yuan/.emacs.d/emacs.pdmp"
 "--eval (setq luna-in-esup t)"
 ;; ++++++++++++++++++++++++++++++++++++++++
 ,(format "--eval=(esup-child-run \"%s\" \"%s\" %d)"
          init-file
          esup-server-port
          esup-depth))
```

6 Other speedup tricks

6.1 early-init.el

This post about early-init.el speedup is useful. Here is my early-init.el.

6.2 Start with correct frame size

Normally Emacs starts with a small frame, and if you have `(toggle-frame-maximized)`, it later expands to the full size. You can eliminate this annoying flicker and make Emacs show up with full frame size. I learned it from this emacs-china post. Basically you use `-g` (for geometry) and `--font` flags together to size the startup frame. I use

```
~/bin/emacs -g 151x50 -font "SF Mono-13"
```

At the point (<2020-01-18 Sat>) you can't use `--dump-file` with `-g` and `-font` because of a bug, but it should be fixed soon. See here.

6.3 Eliminate theme flicker

Manateelazycat sets default background to theme background in custom.el. This way Emacs starts with your theme's background color, instead of white.

Footnotes:

[1] And people have been using the old dumping facility for a even longer time, you can found more on EmacsWiki.

[2] Quote from reddit:

> Caveat emptor: Re-dumping is still not 100% bug-free in the current Emacs codebase (both the emacs-27 release branch and master). There are known issues, and quite probably some unknown ones. Making re-dumping bug-free is not a goal for Emacs 27.1, so this feature should be at this point considered as experimental "use at your own risk" one.

[3] Apart from `--dump-file`, `--dump` also works, even though `emacs --help` didn't mention it. Spacemacs uses `--dump`.

[4] You can found more about it in Emacs 27's Manual. I was foolish enough to read the online manual (Emacs 26 at the time) and not aware of the `load-path` thing until I read Spacemacs's implementation.

Written by Yuan Fu <casouri@gmail.com>

First Published on 2020-01-17 Fri 23:34

Last modified on 2020-03-12 Thu 16:30