Composer 2.0

Nils Adermann
@naderman
Private Packagist
https://packagist.com
Goals for 2.0

- Performance Improvements
- Better reproducibility
  - Most serious 1.x bugs are edge cases which are difficult to debug and hard to reproduce
- Better error reporting
- New features which become easier to add by BC breaks/refactoring
- Keep upgrading as painless as possible
Why 2.0 at all and not 1.x?
Improving Performance

- What makes Composer slow?
  - I/O
    - Network
      - Metadata JSON downloads
      - Package file downloads
    - Memory access
      - Writing, accessing and modifying GBs of memory
  - CPU
    - Sequential unpacking of code archives
Improving Performance

What does Composer use memory for?

- JSON representation of every version of every package that may fit your requirements
- representation of dependencies/conflicts for SAT solver between all of these packages

Solutions

- Reduce number of package versions which "may fit my requirements"
- Represent dependencies/conflicts more efficiently
Reduce number of package versions which “may fit my requirements”

- **Composer 1** lazy loads packages while creating memory representation of dependencies
  - Idea: Solver only loads what it needs when it gets to that point
  - Problems
    - Solver just waits for same info at a later point
    - Impossible to reduce set of packages before generating dependencies
    - Parallelized network access becomes hard to manage

**Composer 2.0** refactors process into multiple clearly separated steps:

- Recursively download metadata only for package versions which may really get installed
- Reduce number of package versions in memory as far as possible
- Generate solver memory representation of dependencies

=> BC Break (for plugins) => 2.0
Represent dependencies/conflicts more efficiently

SAT Solver takes boolean expressions, e.g.

foo/bar 1.0 requires baz/qux ^2.0
foo/bar 1.0 conflicts with baz/qux ^2.0

(- foo/bar 1.0 | baz/qux 2.0.0 | baz/qux 2.0.1 | baz/qux 2.1.0)
(- foo/bar 1.0 | baz/qux 2.0.0) & (- foo/bar 1.0 | baz/qux 2.0.1) &
(- foo/bar 1.0 | baz/qux 2.1.0)

You can only install one version of a package
=> automatically generate a conflict for each pair of versions

foo/bar 1.0, 1.1, 1.2
(- foo/bar 1.0 | foo/bar 1.1) & (- foo/bar 1.0 | foo/bar 1.2) &
(- foo/bar 1.1 | foo/bar 1.2)

Extreme Growth \( \binom{n}{2} = \frac{n!}{2(n-2)!} \)

<table>
<thead>
<tr>
<th></th>
<th>3 versions</th>
<th>6 versions</th>
<th>100 versions</th>
<th>500 versions</th>
<th>1000 versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composer 1</td>
<td>3 rules</td>
<td>15 rules</td>
<td>4,950 rules</td>
<td>124,750 rules</td>
<td>499,500 rules</td>
</tr>
<tr>
<td>Composer 2</td>
<td>1 rule</td>
<td>1 rule</td>
<td>1 rule</td>
<td>1 rule</td>
<td>1 rule</td>
</tr>
</tbody>
</table>

Composer 2.0 uses a special single multi conflict rule representation for all of these rules

foo/bar 1.0, 1.1, 1.2
oneof(foo/bar 1.0, foo/bar 1.1, foo/bar 1.2)
Improving Performance: Network

- JSON Metadata & Package archive downloads
  - **Parallelization** of HTTP requests with curl multi
  - Use of **HTTP/2** features to reduce server round-trips
  - More reliable and feature complete than Composer 1 plugin implementations (hirak/prestissimo, symfony/flex) which were limited by plugin interface

- Packagist.org protocol improvements
  - Reduced amount of data transferred
  - Stability improvements to packagist.org infrastructure

*Note: Improvements require ext-curl*
Improving Performance: Archive Extraction

- Composer 2.0 unzips all downloaded archives in parallel
  - Requires Linux/OS X/WSL
  - Requires CLI command unzip in $PATH
Improving Performance

- What makes Composer slow?
  - I/O ✔
    - Network ✔
    - Metadata JSON downloads ✔
    - Package file downloads ✔
  - Memory access ✔
    - Writing, accessing and modifying GBs of memory ✔
    - Reduce number of package versions which “may fit my requirements” ✔
    - Represent dependencies/conflicts more efficiently ✔
  - CPU ✔
    - Sequential unpacking of code archives ✔
Improving Performance

- Benchmarks?
  - Only anecdotal information so far

“Whoa, I tried Composer V2 alpha 1, nearly 80% faster on a composer install”

“Fast. Faster. Composer 2.0” “If you are still on PHP 7.3 you gain the most, Composer 2.x is about 2.5 times faster than Composer 1.x. If you are already on PHP 7.4, Composer 2.x will be about 1.8 times faster. This is really impressive!”

“composer update Spryker is seeing 64% memory reduction & 51% less time - from 3.4GB to 1.2GB and down from 2 minutes to 1 minute! Thanks to @sprysys for financially supporting this work through a Private Packagist subscription!”
Better Reproducibility:

composer update
vs
composer install
Separating update & install

Metadata

Packagist

composer update

composer.json

composer.lock

composer require

composer remove

Code

Github

vendor
Separating update & install

```
vendor
  - symfony/http-foundation 5.1.2  previous local upgrade attempt
composer.lock
  - symfony/http-foundation: 4.4.10  old production state
composer.json
  - symfony/http-foundation: 5.0.*  limited upgrade for now, because of 5.1 issues
```

```
naderman@saumur:~/projects/composer/test/symfony-http-foundation$ composer update
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
- Upgrading symfony/http-foundation (v4.4.10 => v5.0.10)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 1 update, 1 removal
- Removing symfony/deprecation-contracts (v2.1.3)
- Installing symfony/polyfill-php72 (v1.17.0): Extracting archive
- Installing symfony/polyfill-intl-idn (v1.17.1): Extracting archive
- Installing symfony/mime (v5.1.2): Extracting archive
- Downgrading symfony/http-foundation (v5.1.2 => v5.0.10): Extracting archive
Generating autoload files
6 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```
New Features
Ignoring specific platform requirements

Trying to test your project on PHP8?

```bash
composer update --ignore-platform-reqs
```
Installs on PHP8
May install packages requiring PHP extensions you do not have

```bash
composer update --ignore-platform-req=php
```
Installs on PHP8
Checks all extension requirements as usual
Partial Updates to specific versions

// composer.json

"require": {
  "symfony/http-foundation": "^4.0 || ^5.0",
}

// composer.lock

"packages": [
  {
    "name": "symfony/http-foundation",
    "version": "v4.4.10",
  }
]

$ composer update symfony/http-foundation:5.0
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
  - Upgrading symfony/http-foundation (v4.4.10 => v5.0.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
  - Downloading symfony/http-foundation (v5.0.0)
  - Upgrading symfony/http-foundation (v4.4.10 => v5.0.0): Extracting archive
Repository Priorities

- Repositories are canonical by default:
  - First repository which has a package for a given name wins
  - use "canonical": false to restore old behavior of merging package versions
- Limit packages a repository can provide

```json
{
  "type": "composer",
  "url": "https://some-third-party.com/composer-repo/",
  "only": ["foo/*", "bar/baz"],
  "exclude": ["foo/qux"]
}
```
Upgrading your projects
Best Case

composer self-update --2
composer update / composer install

No errors, everything works as before.
foo/bar requires composer-plugin-api ^1.0.0 -> no matching package found.

- Update foo/bar if new version with Composer 2.0 support available
- Contact author of foo/bar plugin
- Temporarily remove the requirement for the plugin to test

**symfony/flex** is compatible as of 1.8.0!

[https://github.com/symfony/flex/pull/617](https://github.com/symfony/flex/pull/617)
- Compatible with Composer 2.0 as of 1.8.0
  - requires PHP ^7.4
- composer require composer/package-versions-deprecated
  - We forked the package, now compatible with PHP ^7.0
  - replaces ocramius/package-versions
    => satisfies all requirements of ocramius/package-versions

- Building new code requiring runtime access to package info?

  Runtime Composer Utilities
  

  Automatically autoloaded in every Composer project

  ```php
  \Composer\InstalledVersions::isInstalled('vendor/package'); // returns bool
  
  use Composer\Semver\VersionParser;
  \Composer\InstalledVersions::satisfies(new VersionParser, 'vendor/package', '2.0.*');
  ```
Autoloading Issues

Check deprecation warnings in Composer1

Class Foo\Bar located in ./src/SomeName/Bar.php does not comply with psr-4 autoloading standard. It will not autoload anymore in Composer v2.0. in phar:///usr/local/bin/composer/src/Composer/Autoload/ClassMapGenerator.php:18

Make sure directories match class names as defined in PSR-0/4.
How far along is Composer 2.0?
Composer 2.0-alpha2 released June 24, 2020

- Works reliably in nearly all use cases
- A few refactorings and optimizations left before a beta/RC release
- 1.2% of packagist.org installs from Composer 2.0 within last 30 days
Help us test Composer 2.0

`composer self-update --preview`

Updating to version 2.0.0-alpha2 ( preview channel).
  Downloading (100%)
Use `composer self-update --rollback` to return to version 1.10.8

Just run it locally for now, your lock file is committed, no risk involved!

Back to v1? `composer self-update --1`
Resources

- Changelog  
  https://github.com/composer/composer/blob/master/CHANGELOG.md

- Upgrade Guide  
  https://github.com/composer/composer/blob/master/UPGRADE-2.0.md

- Packagist Blog: Development Update (April 24, 2020)  
  https://blog.packagist.com/composer-2-development-update/

- Composer Plugin Readiness for 2.0  
  https://github.com/composer/composer/issues/8726
Questions / Feedback?

Private Packagist
https://packagist.com

E-Mail: n.adermann@packagist.com
Twitter: @naderman