Parallelizing Quickref
ELS 2019

Didier Verna
EPITA / LRDE

didier@lrde.epita.fr
Quickref parallelized

Performance improvement: 4x

Thank you! Any questions?
Introduction

- Reference manuals for CL (Quicklisp) libraries
- quickref.common-lisp.net or local builds
- Originally: sequential loop over 1700+ libraries
- 1h30 – 7h depending on the conditions
- Parallelism worth investigating
Plan

Toolchain

Experiments

Parallel Solution

Discussion and Perspectives
Plan

Toolchain

Experiments

Parallel Solution

Discussion and Perspectives
**Important remarks:**

- Declt works by introspection
  - Compilation / loading (of dependencies) may be required
  - Avoid loading 1700+ libraries in the same Lisp image!
  - Run Declt in external processes
- Makeinfo is a Perl/C script
  - *Ditto*
Plan

Toolchain

Experiments

Parallel Solution

Discussion and Perspectives
Quickref Options

- Libraries and Update
  - full / installed-only
  - download / compilation may occur

- Compilation cache Policy
  - global / local
  - global cache may cause problems

Scenarios

1. All libraries already compiled (1h 27m)
2. Nothing compiled, global compilation cache (1h 51m)
3. Nothing compiled, local compilation cache (7h 01m)
Plan

Toolchain

Experiments

Parallel Solution

Discussion and Perspectives
Usable in all scenarios

Dependency graph management cheap

Scenario 2

- Best results: 4 Declt threads / 4 Makeinfo threads
- Total time: 28m 17s (25% of sequential time)
Library Pool Management

Dependency Graph

Library Pool

Done
Library Pool Management

Dependency Graph

Library Pool

Done

parallelizing Quickref / ELS 2019 – Didier Verna
Library Pool Management

Dependency Graph

Library Pool

Done

Parallelizing Quickref / ELS 2019 – Didier Verna
Library Pool Management

Dependency Graph

5 ➔ 6

Library Pool

5

Done

2 1 3 4
Library Pool Management

Dependency Graph

Library Pool

1. 2. 3. 4. 5. 6.

Done

Parallelizing Quickref / ELS 2019 – Didier Verna
Library Pool Management

Dependency Graph

Library Pool

Done

1 2 3 4 5 6
Plan

Toolchain

Experiments

Parallel Solution

Discussion and Perspectives
Discussion and Perspectives

- Alternative Algorithms
- Dependency Management Issues
  - Based on static information provided by Quicklisp
  - Fragile, not always correct
- CPU vs. I/O Consumption
  - 4x is a bit disappointing
  - Open the Declt and Makeinfo black boxes
- SJF-like Scheduling
  - Very difficult to figure out where complexity comes from
  - Collect timings and use them in next run
- SSD!
Acknowledgments

- Initial code base: Antoine Martin
- Author index and parallel algorithm #4: Antoine Hacquard
- Hosting (code & website): CLF / common-lisp.net