Optimising the Mapnik Toolchain @ SOTM 2012

Optimising the Mapnik Rendering Toolchain 2.0

SOTM TOKYO 2012

Frederik Ramm
frederik@remote.org

stopwatch CC-BY maedli @ flickr

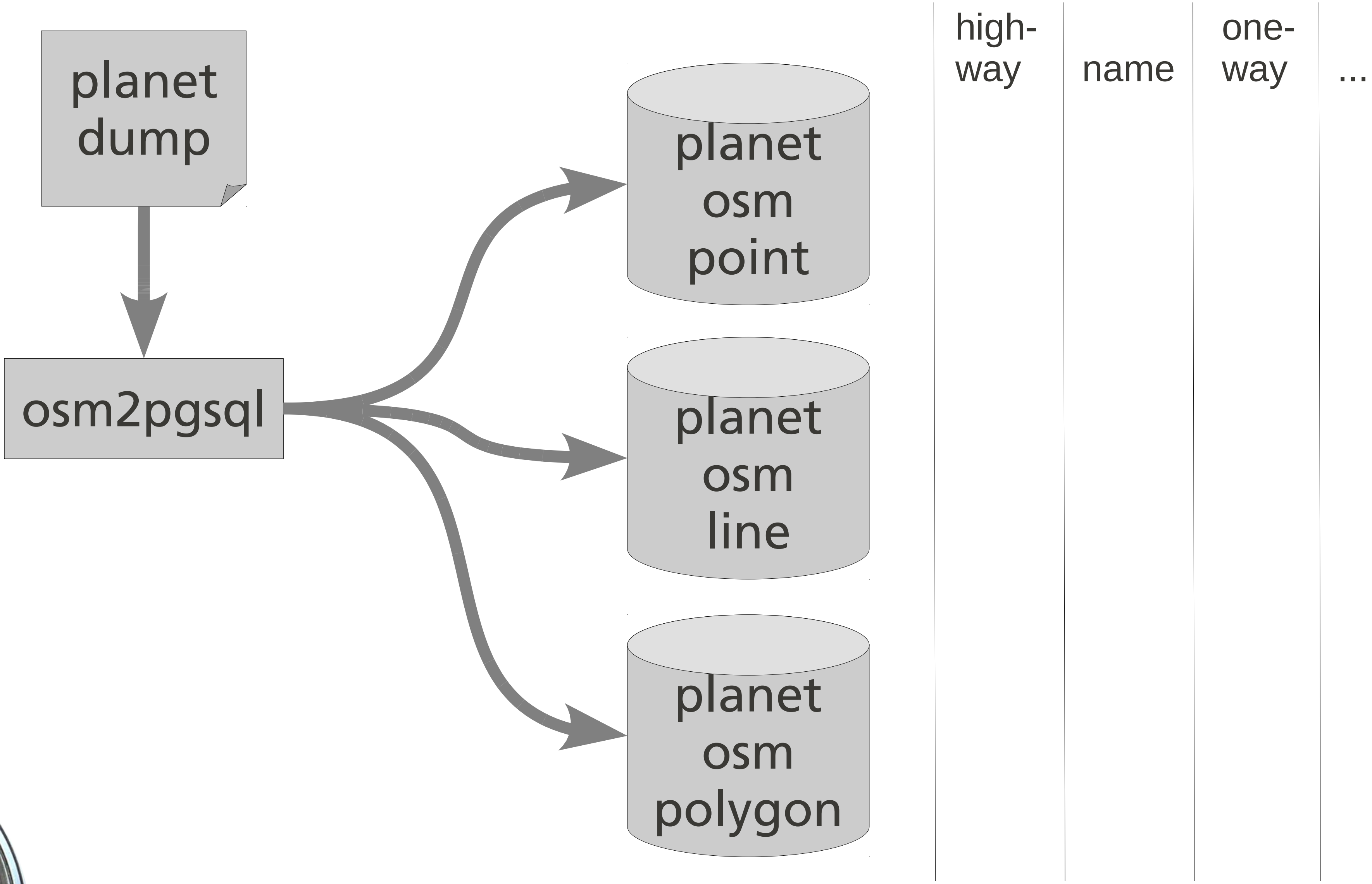
Basic Setup

- “Hetzner” dedicated server (US$ 150/mo)
- Ubuntu Linux
- Mapnik 2.1
- pbf planet file of March 2012
- PostgreSQL 9.1/PostGIS 2.0
- 3x 120 GB SSD
- 32 GB RAM

1.

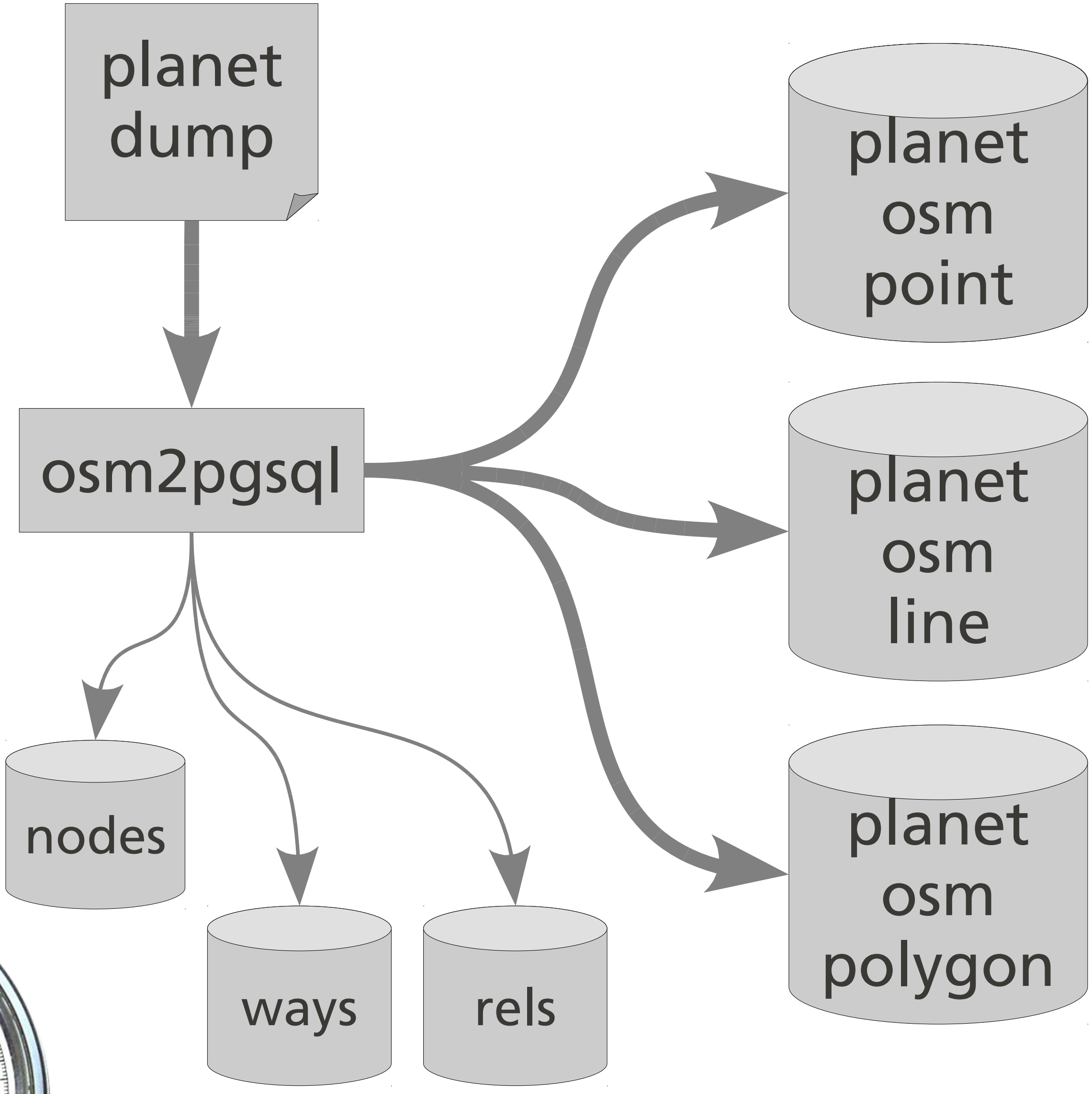
Optimising the Mapnik Toolchain @ SOTM 2012

What does osm2pgsql do?



highway	name	oneway	...

osm2pgsql slim mode



planet dump

osm2pgsql

planet osm point

planet osm line

planet osm polygon

nodes

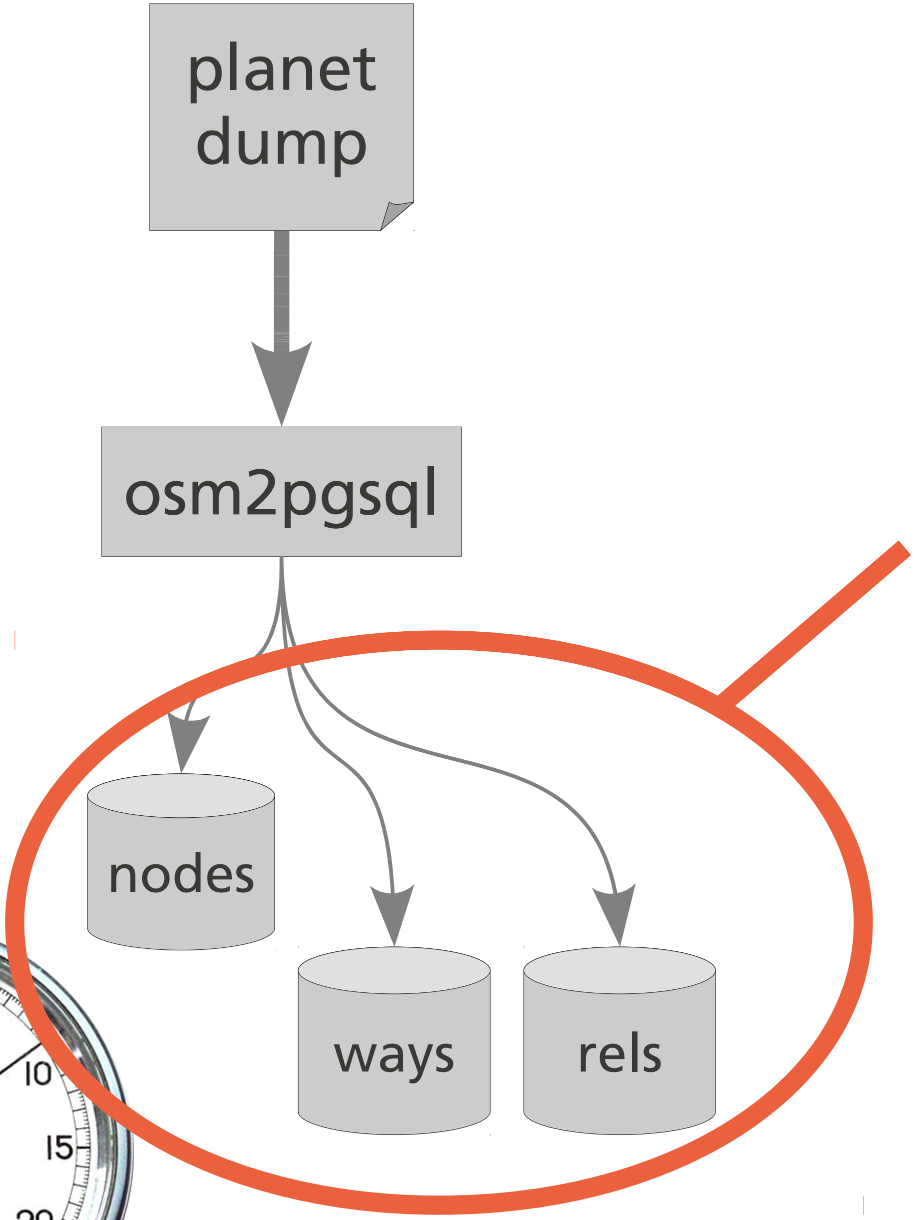
ways

rels

high-way	name	one-way	...

Optimising the Mapnik Toolchain @ SOTM 2012

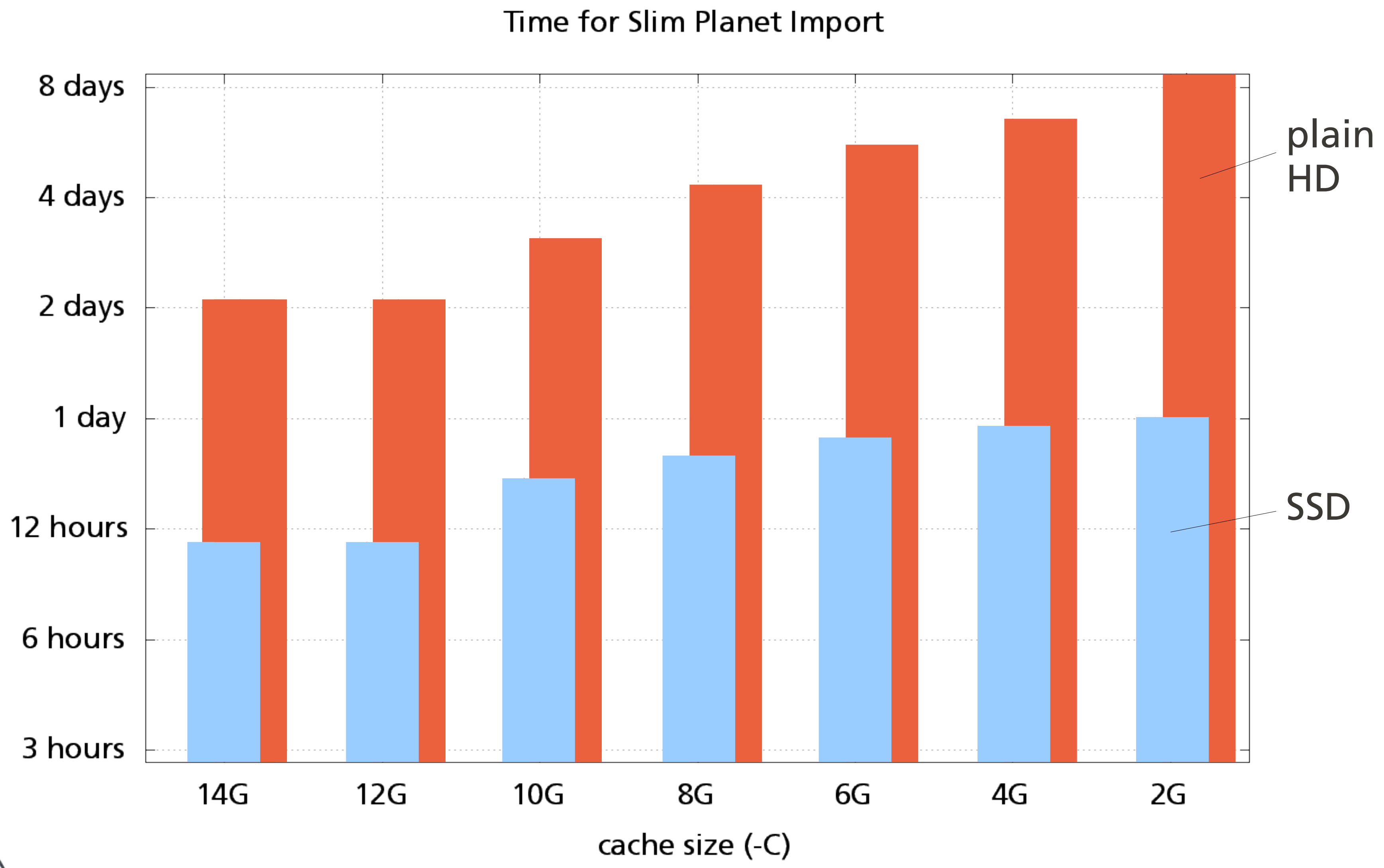
osm2pgsql slim mode



These tables started out as pure "helper" tables for memory-poor systems.

Today they are widely used because they are also a pre-requisite for updates!

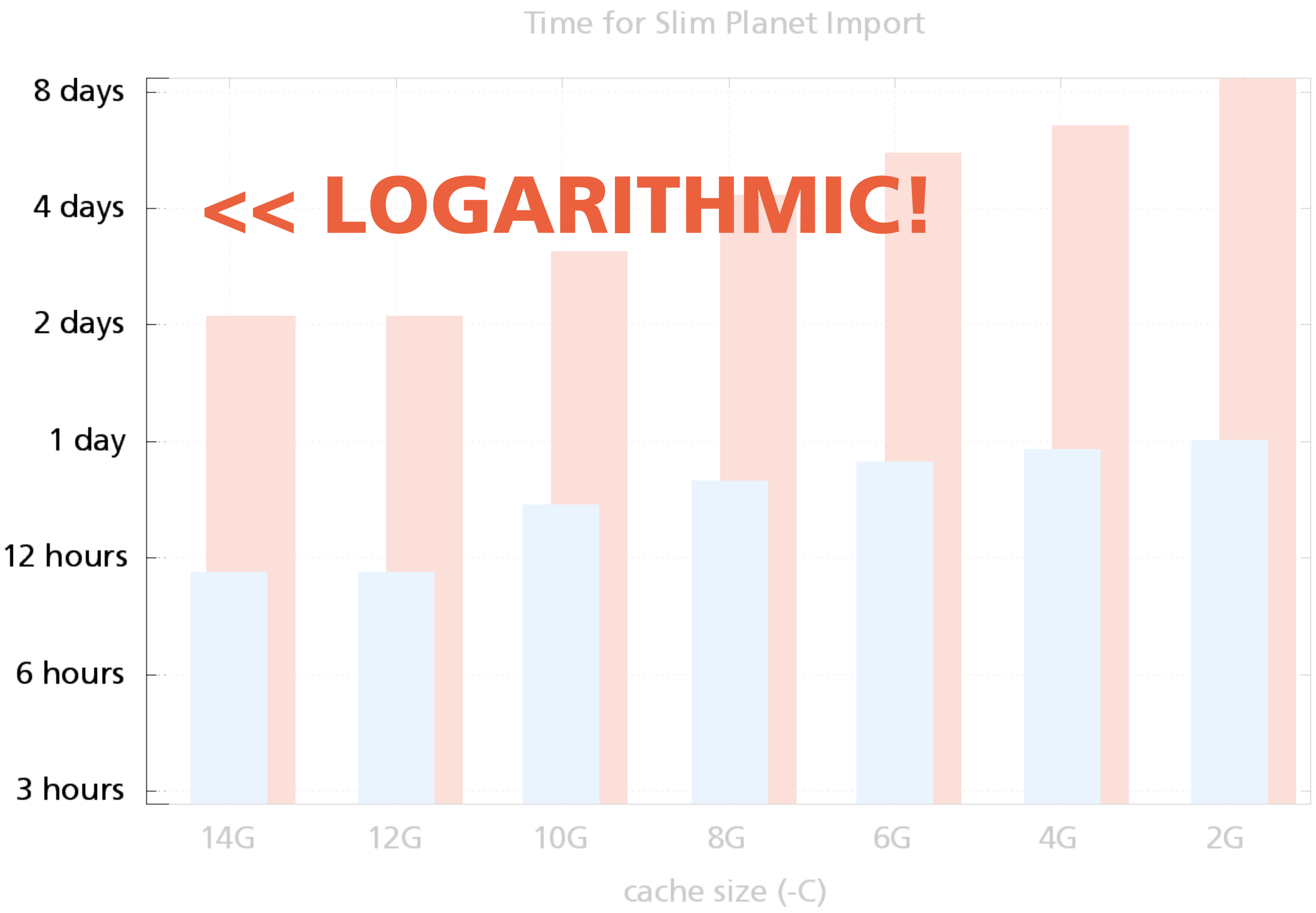
Import Time



(plain HD: Seagate Barracuda 7200.14 3TB; SSD: 3xOCZ Vertex3)

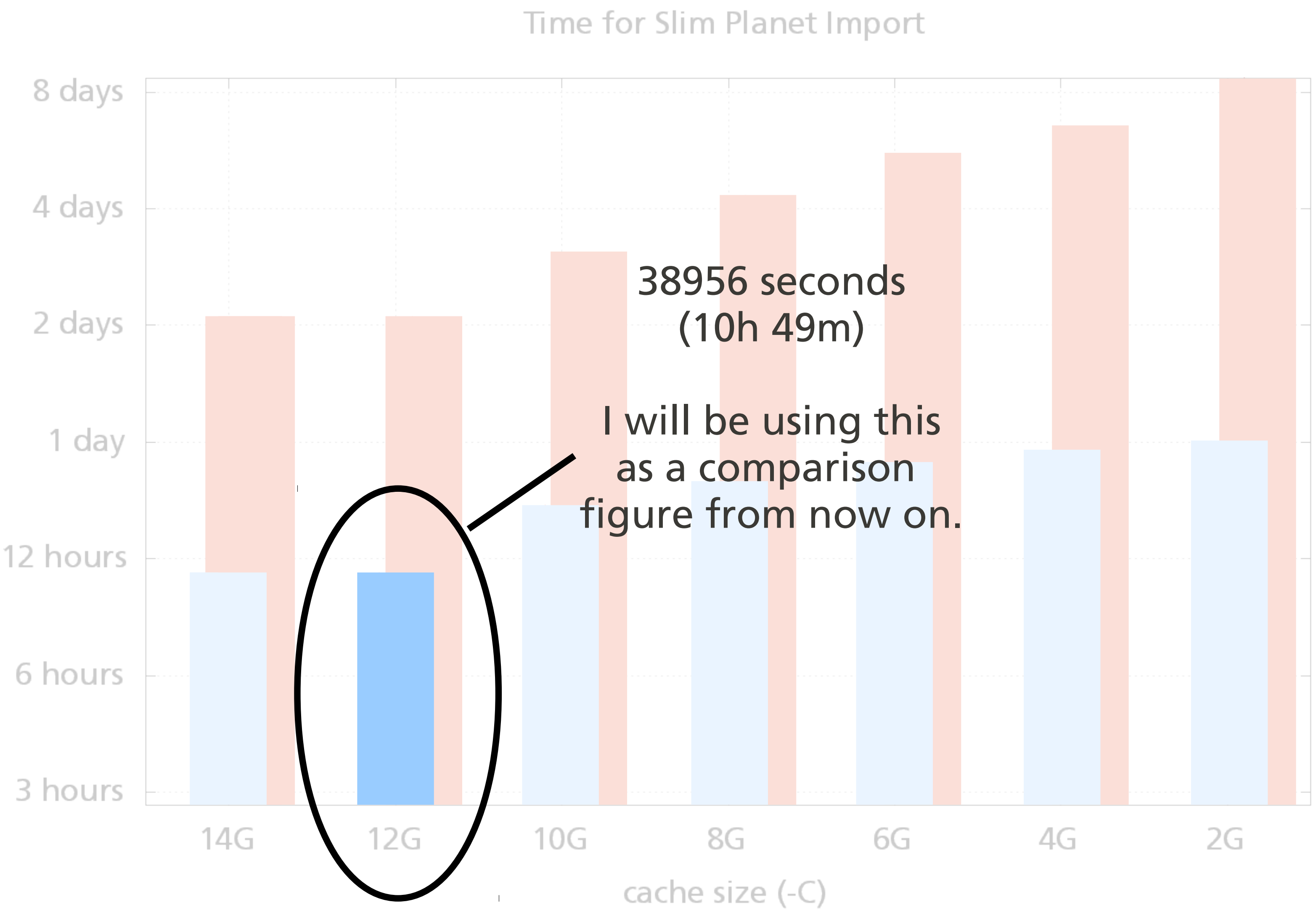
Optimising the Mapnik Toolchain @ SOTM 2012

Import Time



<< LOGARITHMIC!

Import Time

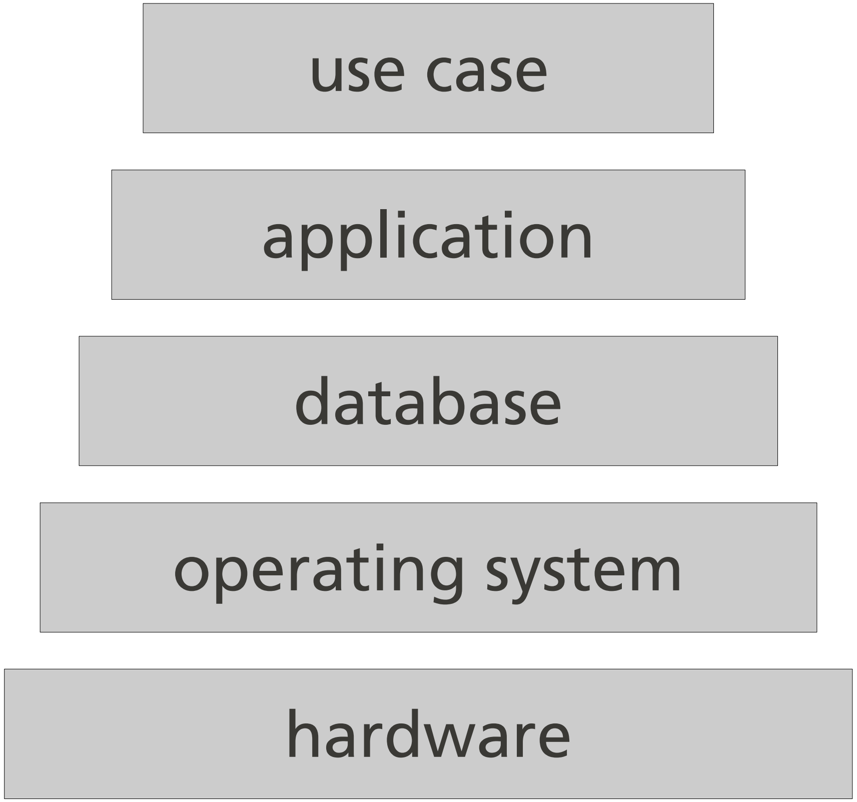


38956 seconds
(10h 49m)

I will be using this
as a comparison
figure from now on.

Optimising the Mapnik Toolchain @ SOTM 2012

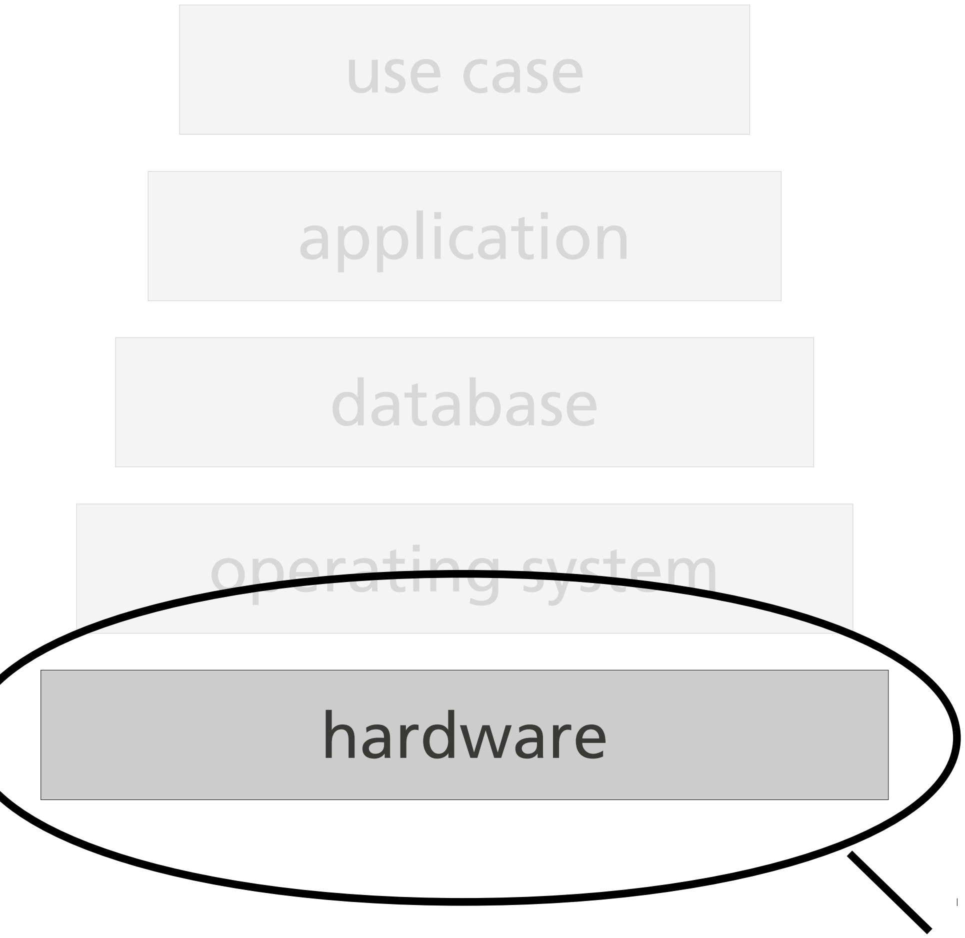
So you want to tune your system...



use case	here: import planet file
application	here: osm2pgsql
database	here: PostgreSQL/PostGIS
operating system	e.g. file system, drivers
hardware	e.g. memory, type of disk

Optimising the Mapnik Toolchain @ SOTM 2012

So you want to tune your system...



use case — here: import planet file

application — here: osm2pgsql

database — here: PostgreSQL/PostGIS

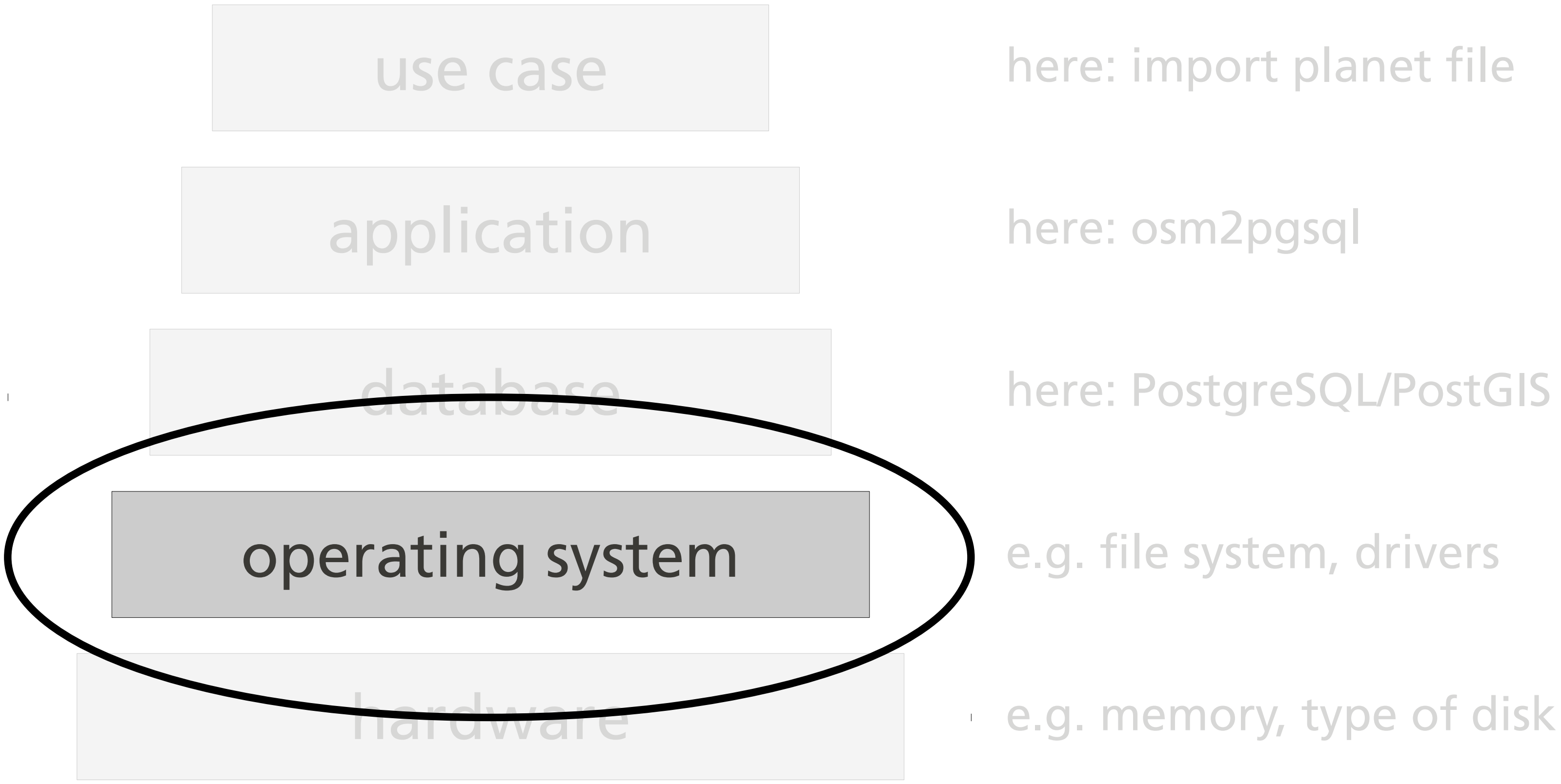
operating system — e.g. file system, drivers

hardware — e.g. memory, type of disk

we've looked at this only, yet.

Optimising the Mapnik Toolchain @ SOTM 2012

So you want to tune your system...



use case

here: import planet file

application

here: osm2pgsql

database

here: PostgreSQL/PostGIS

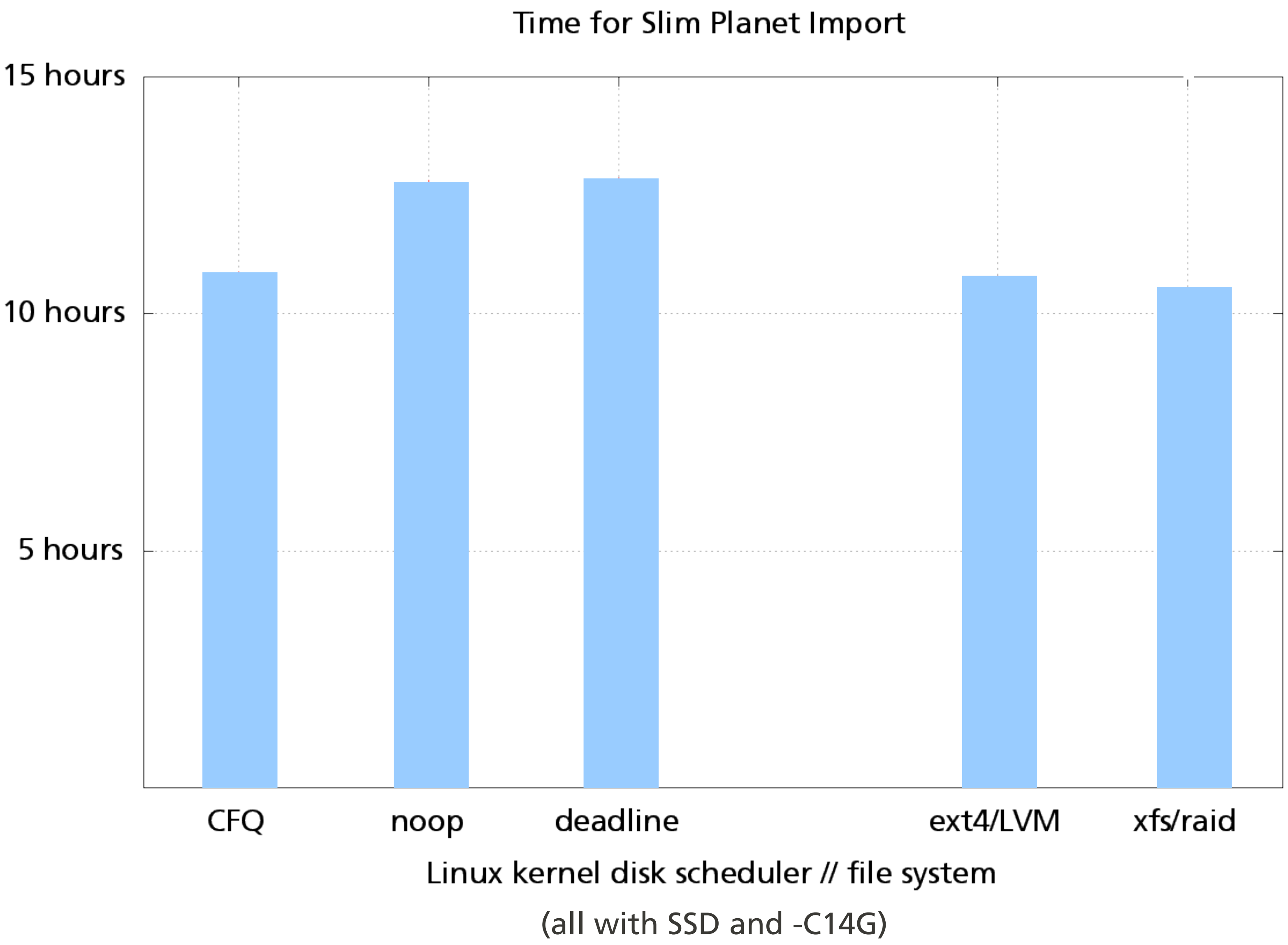
operating system

e.g. file system, drivers

hardware

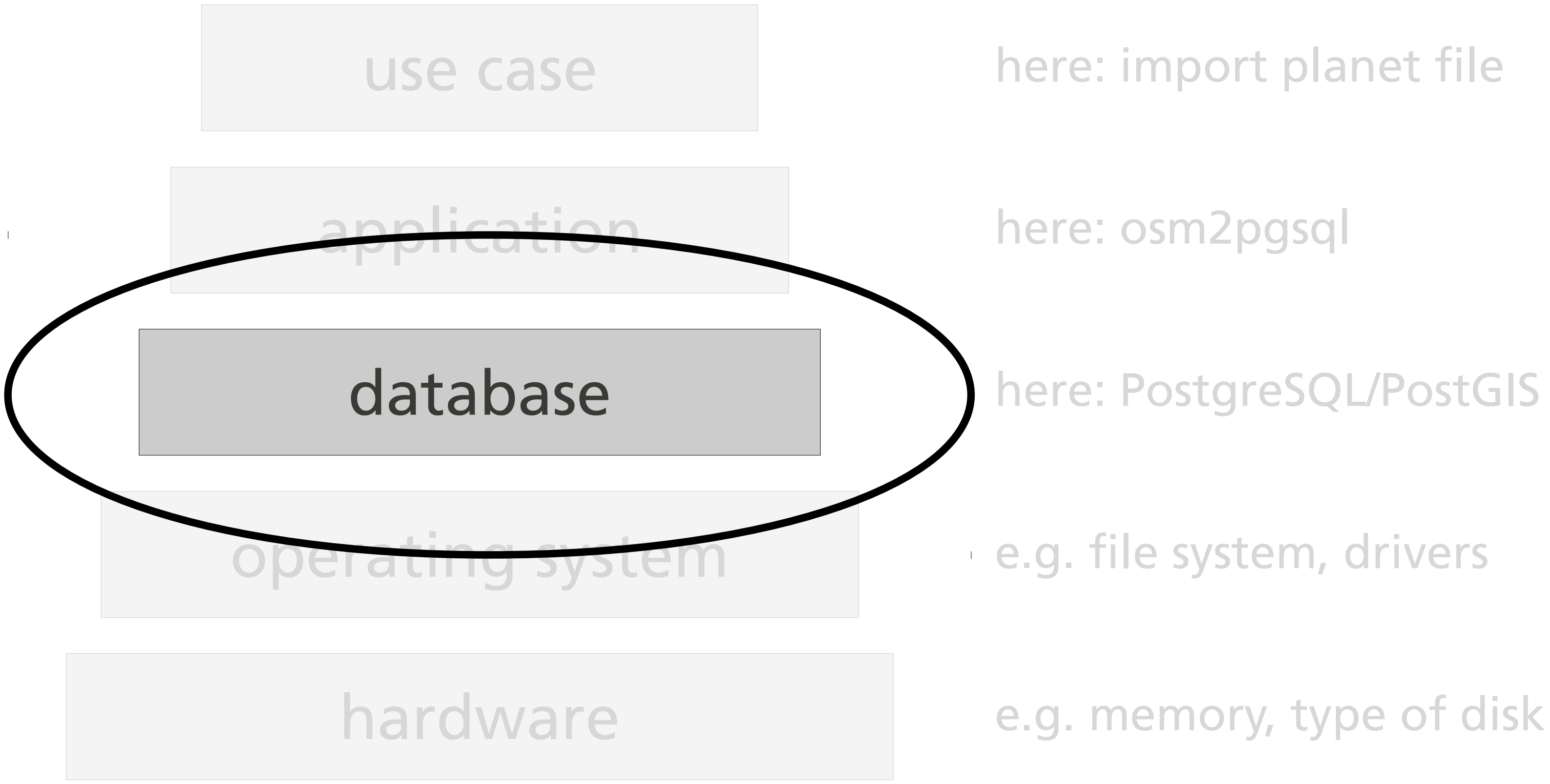
e.g. memory, type of disk

Import Time


Time for Slim Planet Import
15 hours
10 hours
5 hours
CFQ
noop
deadline
ext4/LVM
xfs/raid
Linux kernel disk scheduler // file system
(all with SSD and -C14G)

Optimising the Mapnik Toolchain @ SOTM 2012

So you want to tune your system...



use case — here: import planet file

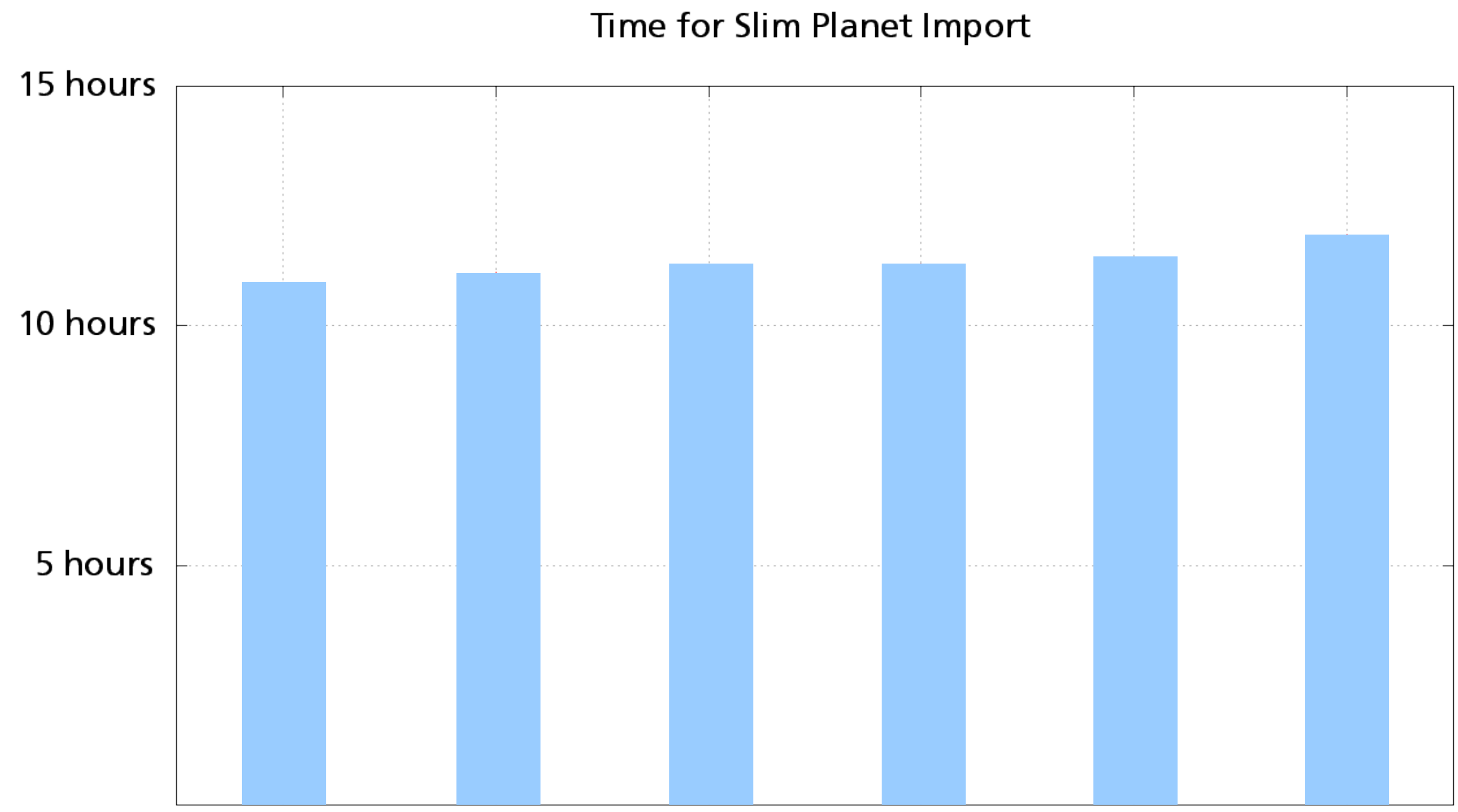
application — here: osm2pgsql

database — here: PostgreSQL/PostGIS

operating system — e.g. file system, drivers

hardware — e.g. memory, type of disk

Import Time



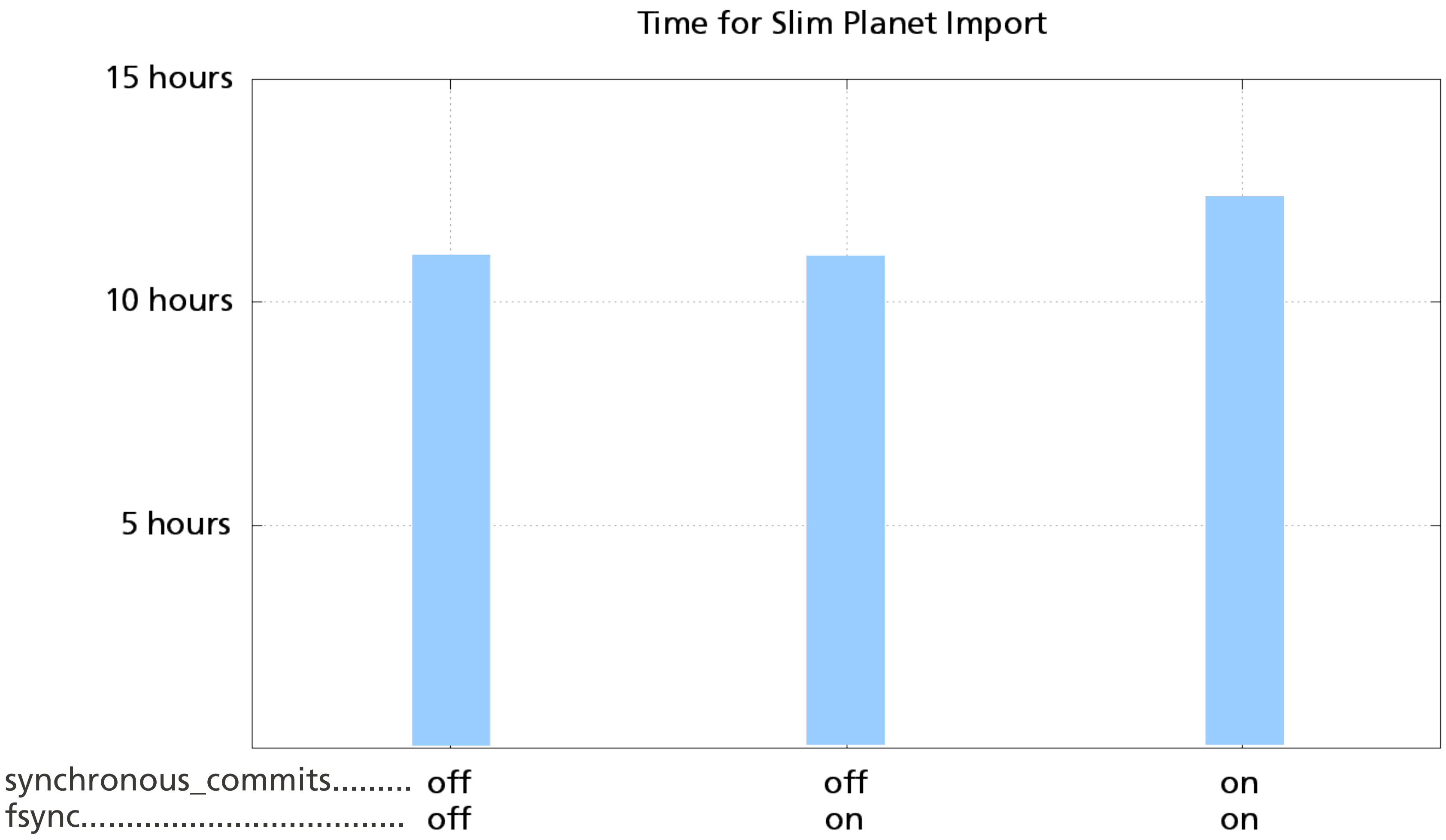
shared_buffers.........	8	1024	4096	8	1024	4096
work_mem................	1	128	64	1	1	1
maintenance_ ...... work_mem	4096	256	1024	16	4096	512

PostgreSQL 9.1 memory settings

(all in MB // benchmarks with SSD and -C14G)

Optimising the Mapnik Toolchain @ SOTM 2012

Import Time



PostgreSQL 9.1 fsync settings

(all benchmarks with SSD and -C14G)

Optimising the Mapnik Toolchain @ SOTM 2012

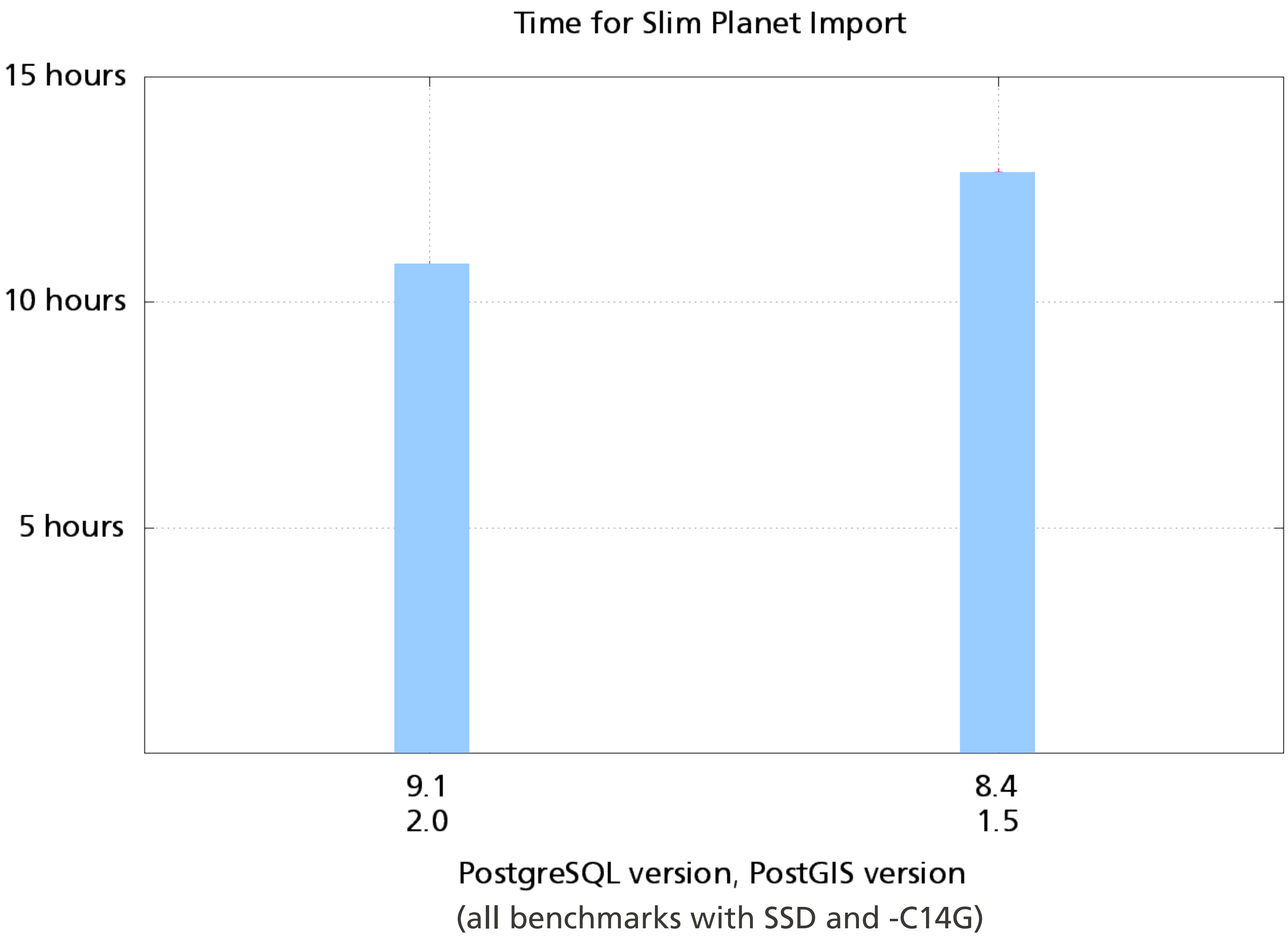
PostgreSQL Tuning for N00bs

/etc/postgres/9.1/main/postgresql.conf:

option	default	recommended
shared_buffers	24 MB	8 MB
work_mem	1 MB	1 MB
maintenance_work_mem	16 MB	4096 MB
fsync	on	off
autovacuum	on	off (*)
checkpoint_segments		60
random_page_cost	4.0	1.1
effective_cache_size		
effective_io_concurrency	1	

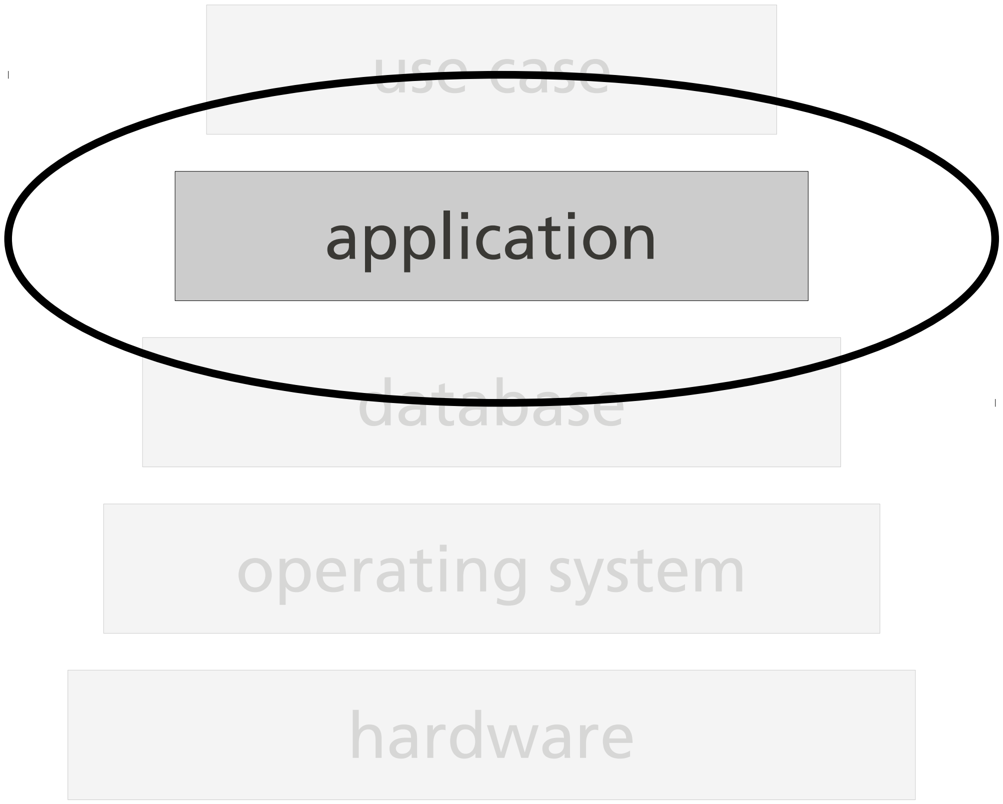
Optimising the Mapnik Toolchain @ SOTM 2012

Import Time



Optimising the Mapnik Toolchain @ SOTM 2012

So you want to tune your system...



use case — here: import planet file

application — here: osm2pgsql

database — here: PostgreSQL/PostGIS

operating system — e.g. file system, drivers

hardware — e.g. memory, type of disk

Various osm2pgsql options

reference figure: 38956s (short of 11h)

- add reprojection (no -l): + 0.3% (slower)
- use 64bit osm2pgsql: + 11.7% (slower)
- unlogged tables: +17% (slower)

- flat node storage: +25% (slower)
- hstore: +10% (slower)

batch 20100629-1005, 20100630-1721

What is hstore?

- uses a hash column to keep tag key/value combinations instead of tons of columns
- different flavours
- best results: --hstore and --hstore-match-only, with a list of “must-have” tags and a list of “drop” tags

- approx. 10% slower
- disk usage same as before (~ 280G)
- access to all extra tags (e.g. name:xx)

batch 20100629-1005, 20100630-1721

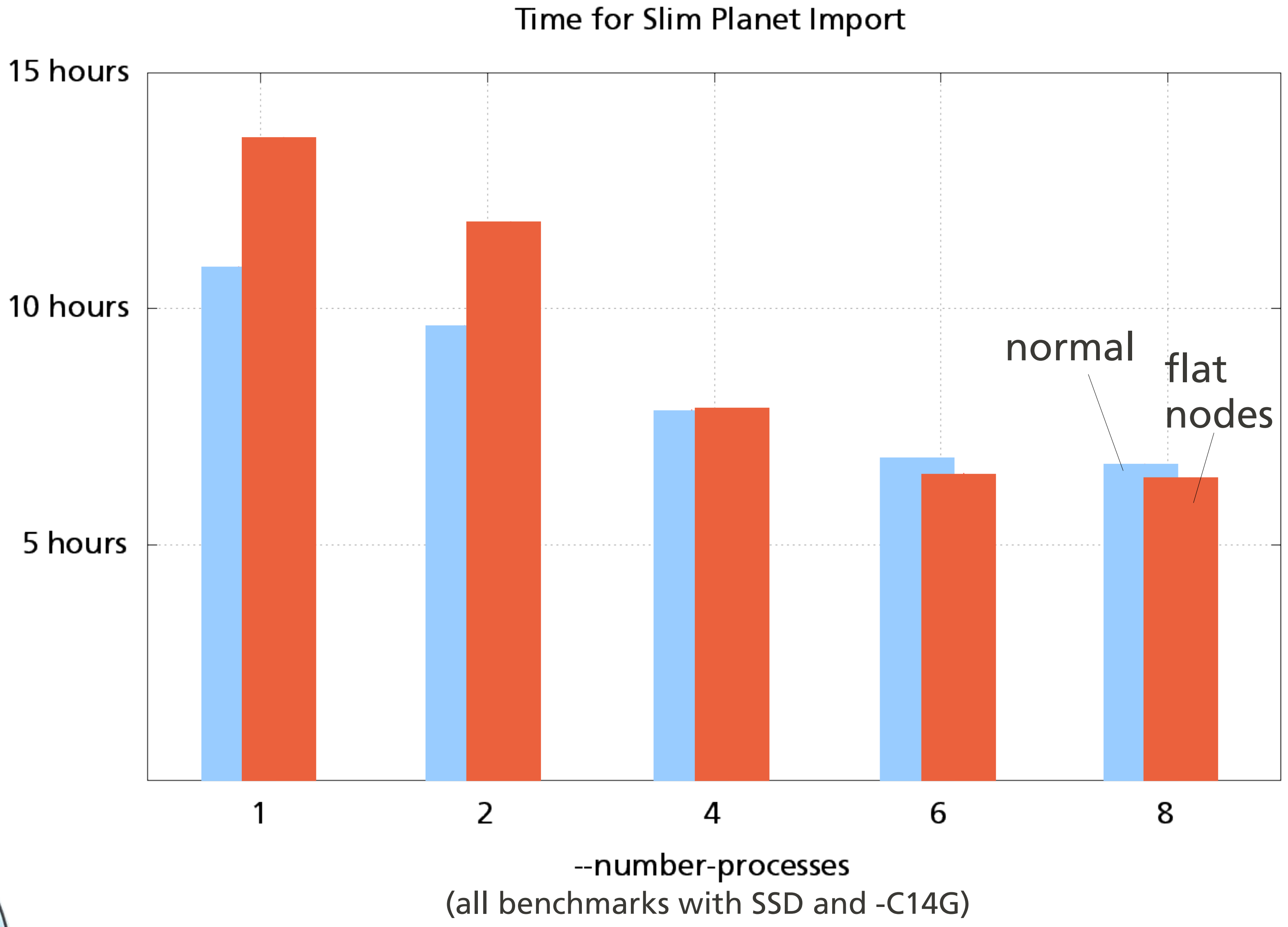
What is flatnodes?

- instead of storing node lat/lon in a database table, uses a file on disk
- saves about 80G (30%) of disk space
- takes 25% longer but parallelises better

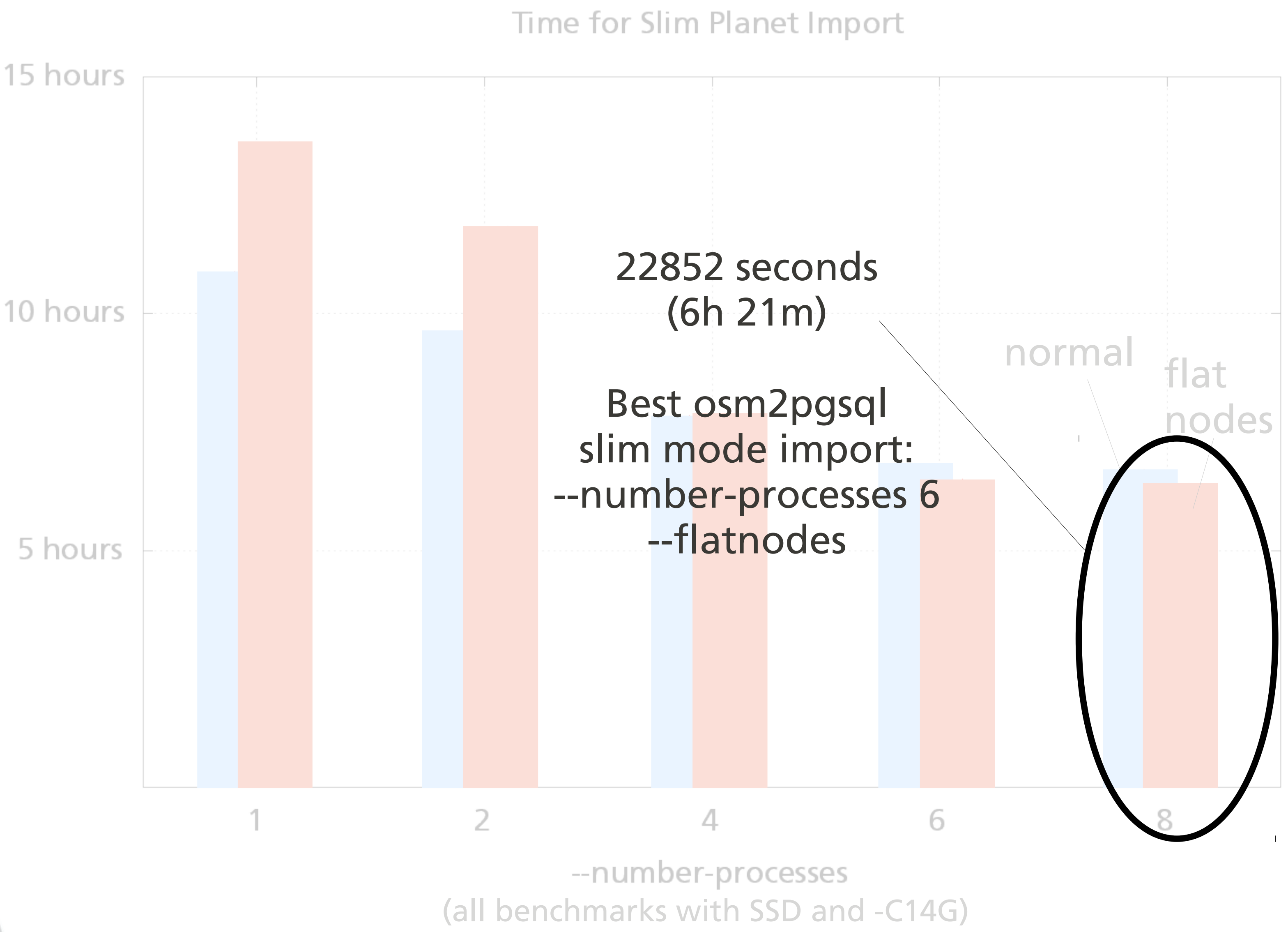
- if you must use a magnetic disk, flatnodes will usually give you a huge performance boost due to much reduced seeking

batch 20100629-1005, 20100630-1721

Parallelisation (8-core machine)



Parallelisation (8-core machine)

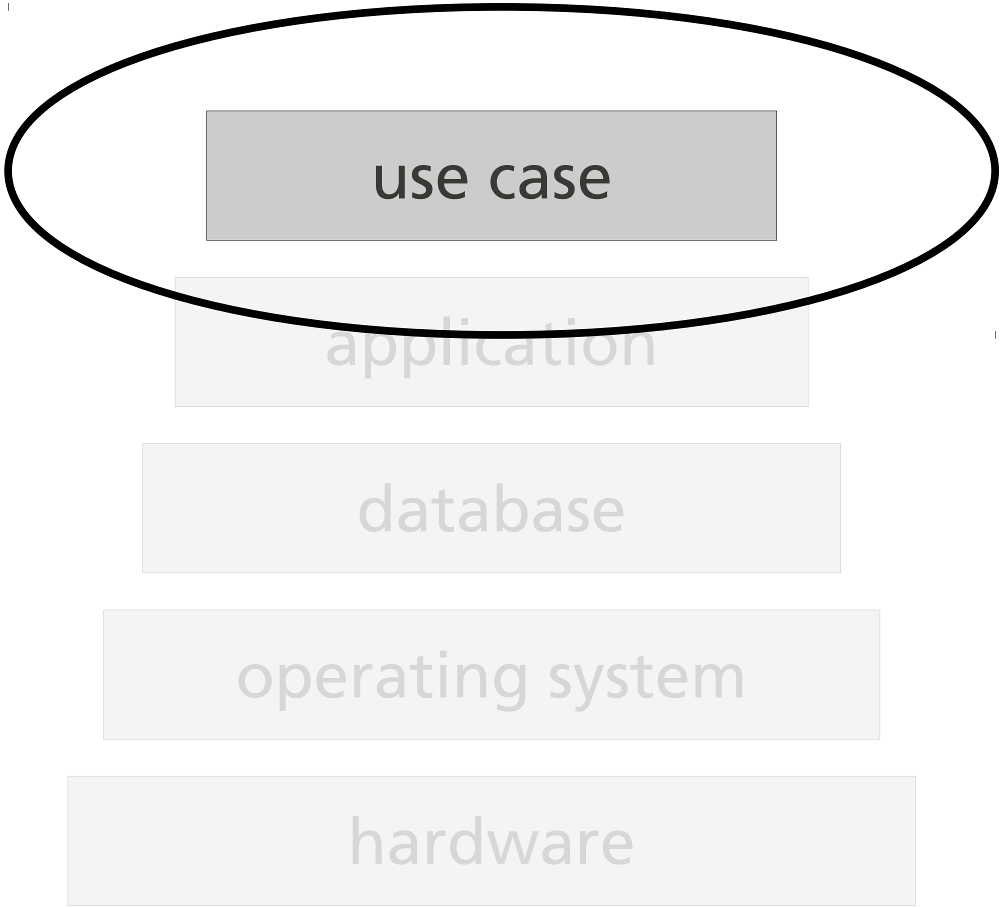


22852 seconds
(6h 21m)

Best osm2pgsql
slim mode import:
--number-processes 6
--flatnodes

Optimising the Mapnik Toolchain @ SOTM 2012

So you want to tune your system...



use case

application

database

operating system

hardware

here: import planet file

here: osm2pgsql

here: PostgreSQL/PostGIS

e.g. file system, drivers

e.g. memory, type of disk

Alternatives to osm2pgsql

- imposm:
  total import time 22893s “out of the box”,
  no updates
  55 GB on disk (+17 temporary)
  (osm2pgsql with –drop: 66 GB)
  different table structure;
  50 minutes extra gives you simplified tables

- osmosis:
  unsuitable for rendering

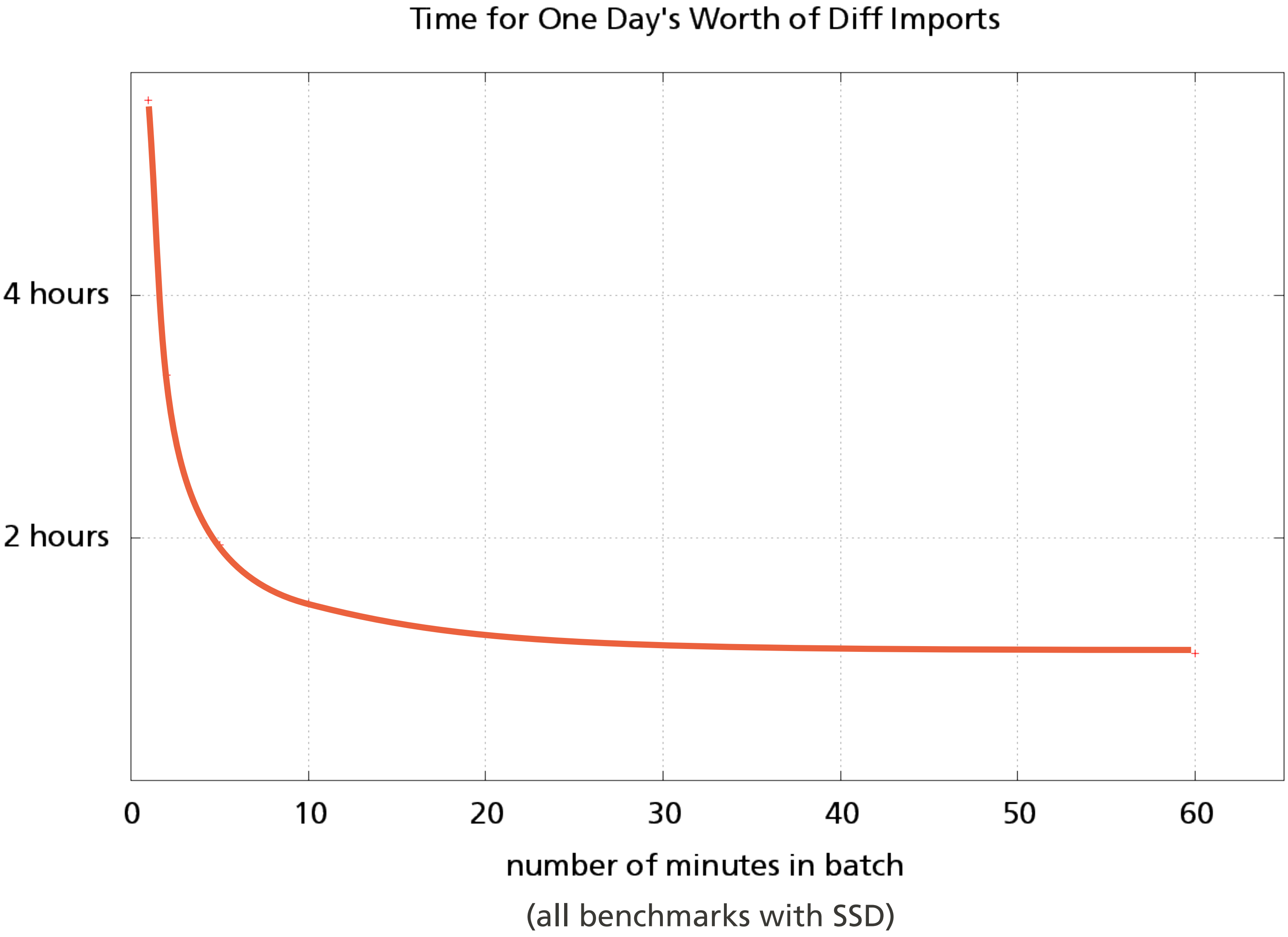
- new OGR driver:
  no benchmarks yet

Slim Mode Updates

- scenario:

  one day's worth of updates applied to slim import
- basic settings as before

Update Time


Time for One Day's Worth of Diff Imports
4 hours
2 hours
0
10
20
30
40
50
60
number of minutes in batch
(all benchmarks with SSD)

S/P C6560

Slim Mode Updates

- 64 bit: + 10% (slower)
- --number-processes=6: -5% (faster)
- flatnodes: -5% (faster)
- hstore: +/- 0

- With HDD, +150% on minutely diffs (spending 14h per day on updates)

Optimising the Mapnik Toolchain @ SOTM 2012

3.

Rendering Performance

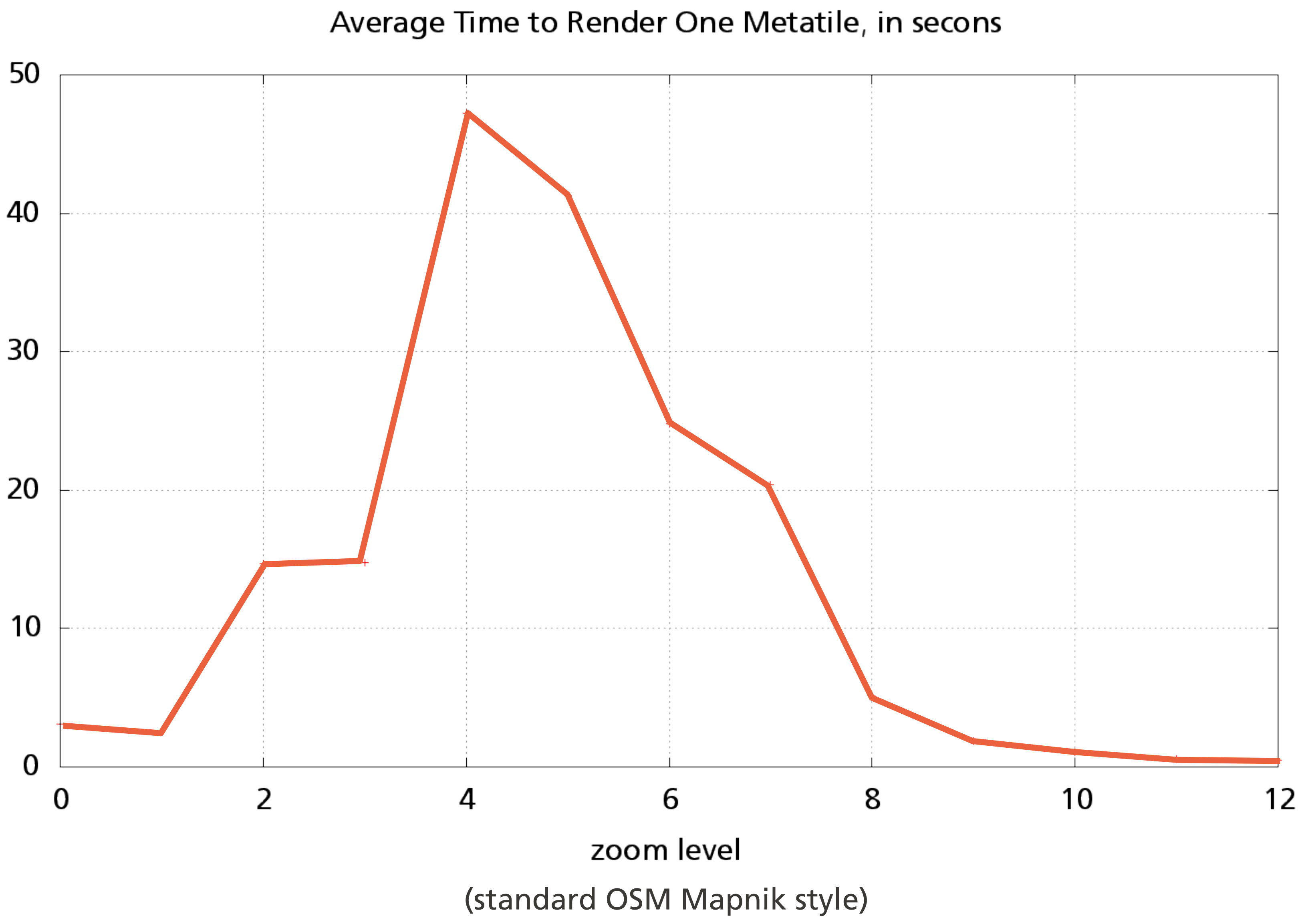
- scenario:
  a batch of PostgreSQL queries were logged while rendering ~ 17k meta tiles, and replayed in various settings.

Rendering Performance

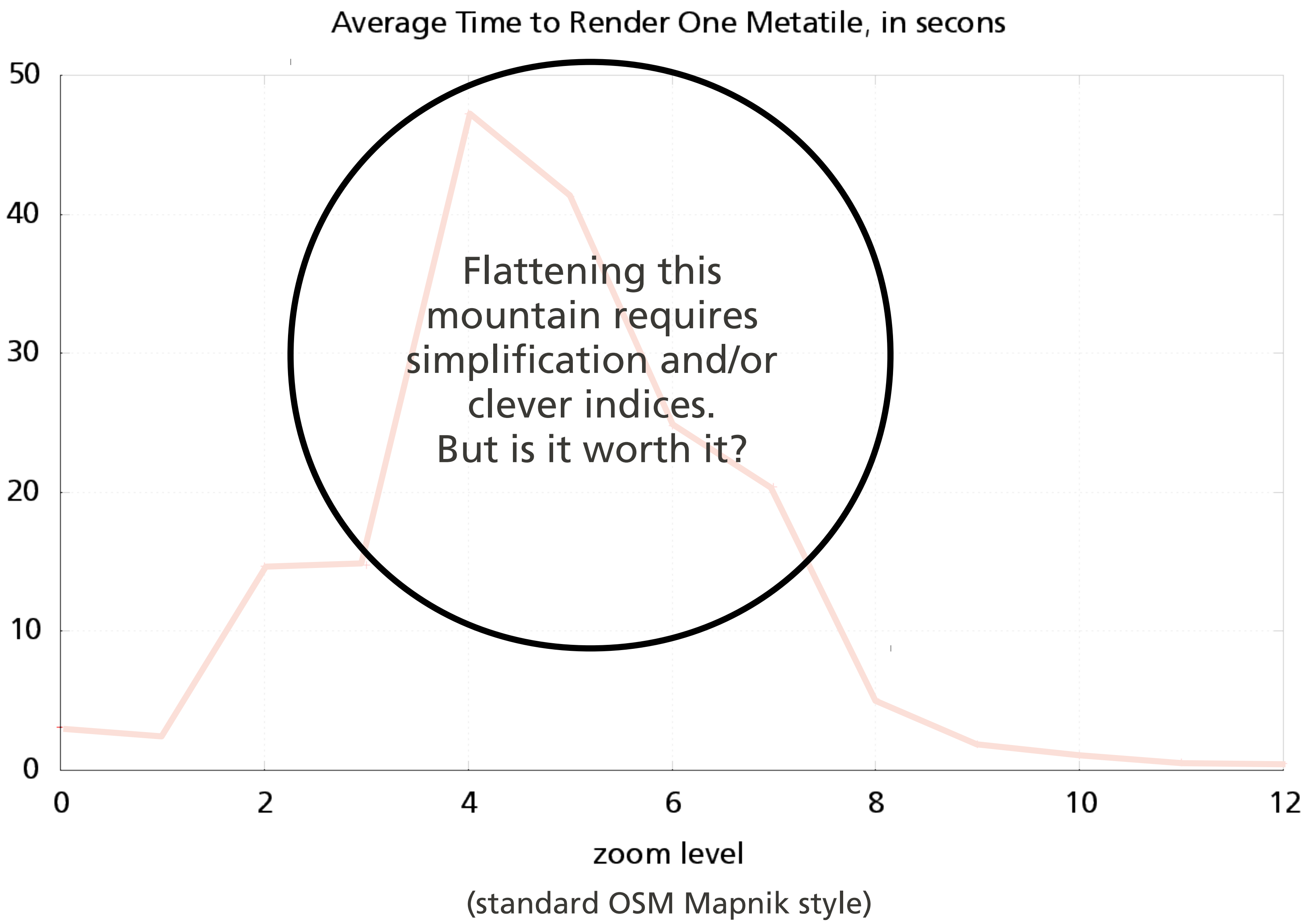
- basic result: 71 minutes (~ 4 MT/s)
- 64bit: + 1%
- HDD: +55%
- flatnode: +/- 0
- hstore and views: + 1%

- even deliberately “breaking” PostgreSQL config (different shared_memory etc) did not change much.

Rendering Performance



Rendering Performance



Flattening this mountain requires simplification and/or clever indices.
But is it worth it?

Rendering Performance

- geometry simplification:

  beforehand (imposm) or on-the-fly (Mapnik option)
- indices:

  use analyze_postgis_log.pl from

  svn.openstreetmap.org/applications/utils/tirex/utils
- clipping: make sure you don't have giant geometries

Optimising the Mapnik Toolchain @ SOTM 2012

In a Nutshell

- use SSD
- configure PostgreSQL right
- update every 10-15 minutes
- depending on use case, make indexes and simplify geometries

Optimising the Mapnik Toolchain @ SOTM 2012

Thank you

Frederik Ramm
frederik@remote.org