RedisJSON

A Document DB in Rust

Dr. Christoph Zimmermann

Redis Labs

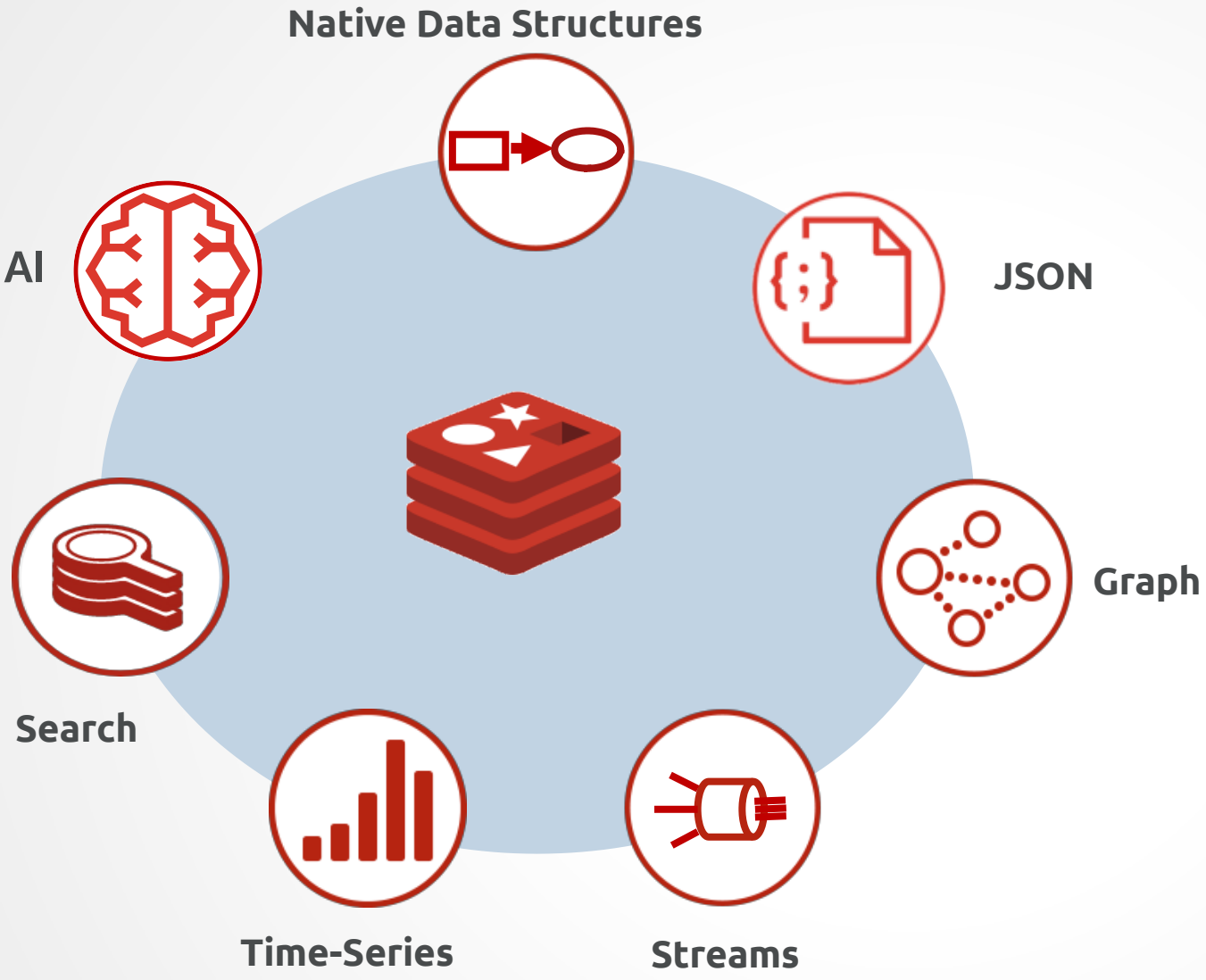
26. 5. 2020 @ FraLUG

cat /etc/motd

- What is Redis / RedisJSON
- Architecture
- The application perspective
- Summary / outlook

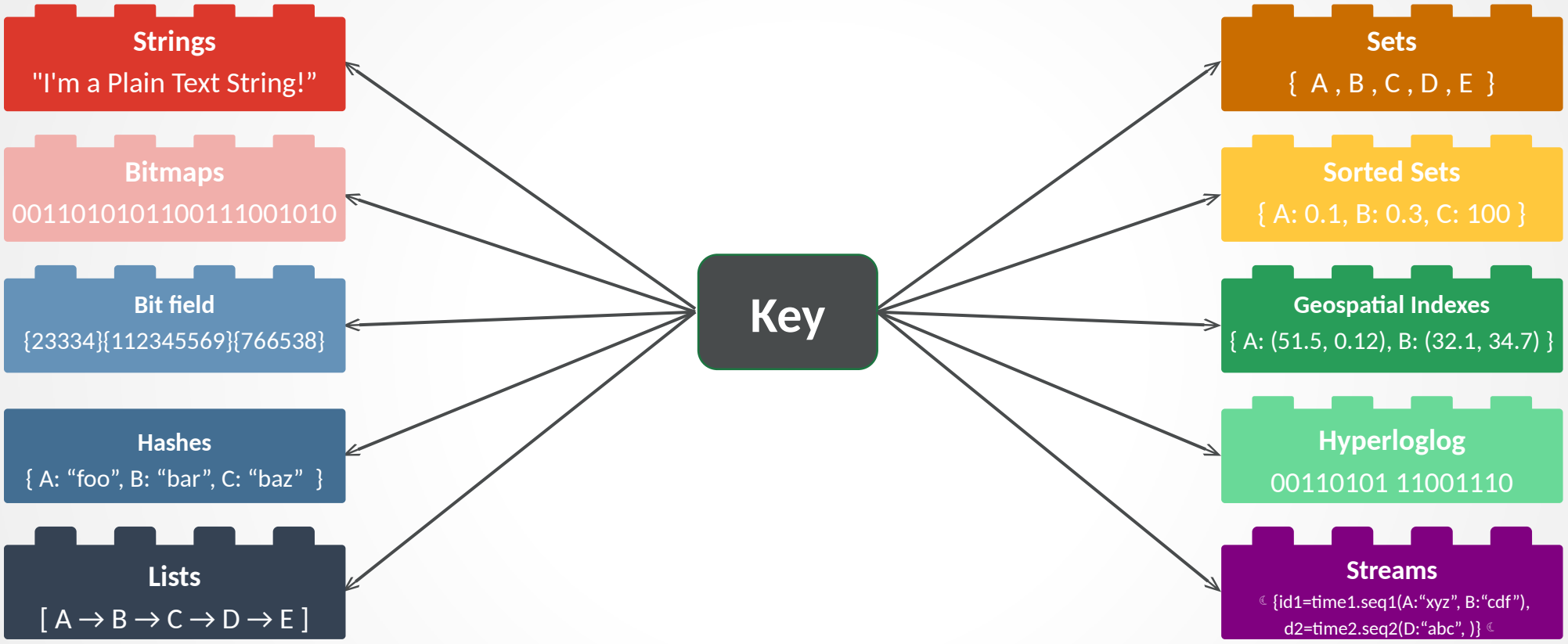
cat /etc/group

ls /opt/redis



- Dedicated engine for each data model (vs. API only)
- Models engines can be selectively loaded, according to use case
- All model engines access the same data, eliminating the need for transferring data between them

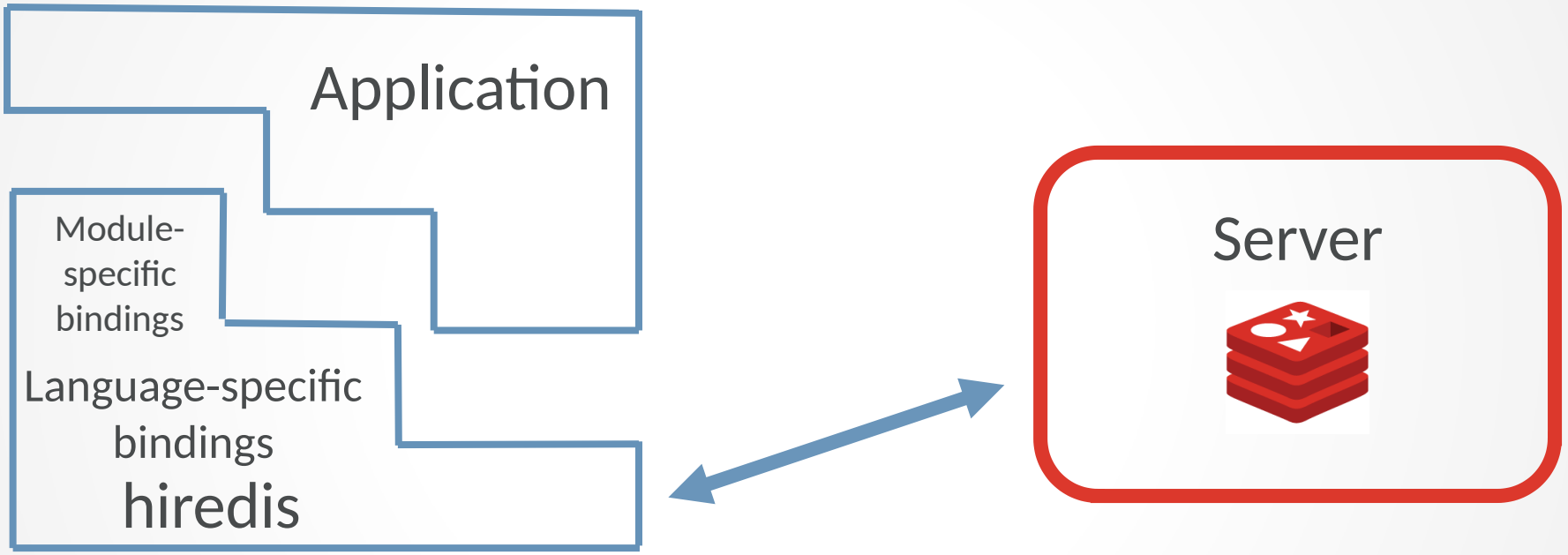
type which


Strings
"I'm a Plain Text String!"
Bitmaps
0011010101100111001010
Bit field
{23334}{112345569}{766538}
Hashes
{ A: "foo", B: "bar", C: "baz" }
Lists
[A → B → C → D → E]
Key
Sets
{ A , B , C , D , E }
Sorted Sets
{ A: 0.1, B: 0.3, C: 100 }
Geospatial Indexes
{ A: (51.5, 0.12), B: (32.1, 34.7) }
Hyperloglog
00110101 11001110
Streams
{id1=time1.seq1(A:"xyz", B:"cdf"),
d2=time2.seq2(D:"abc",)}

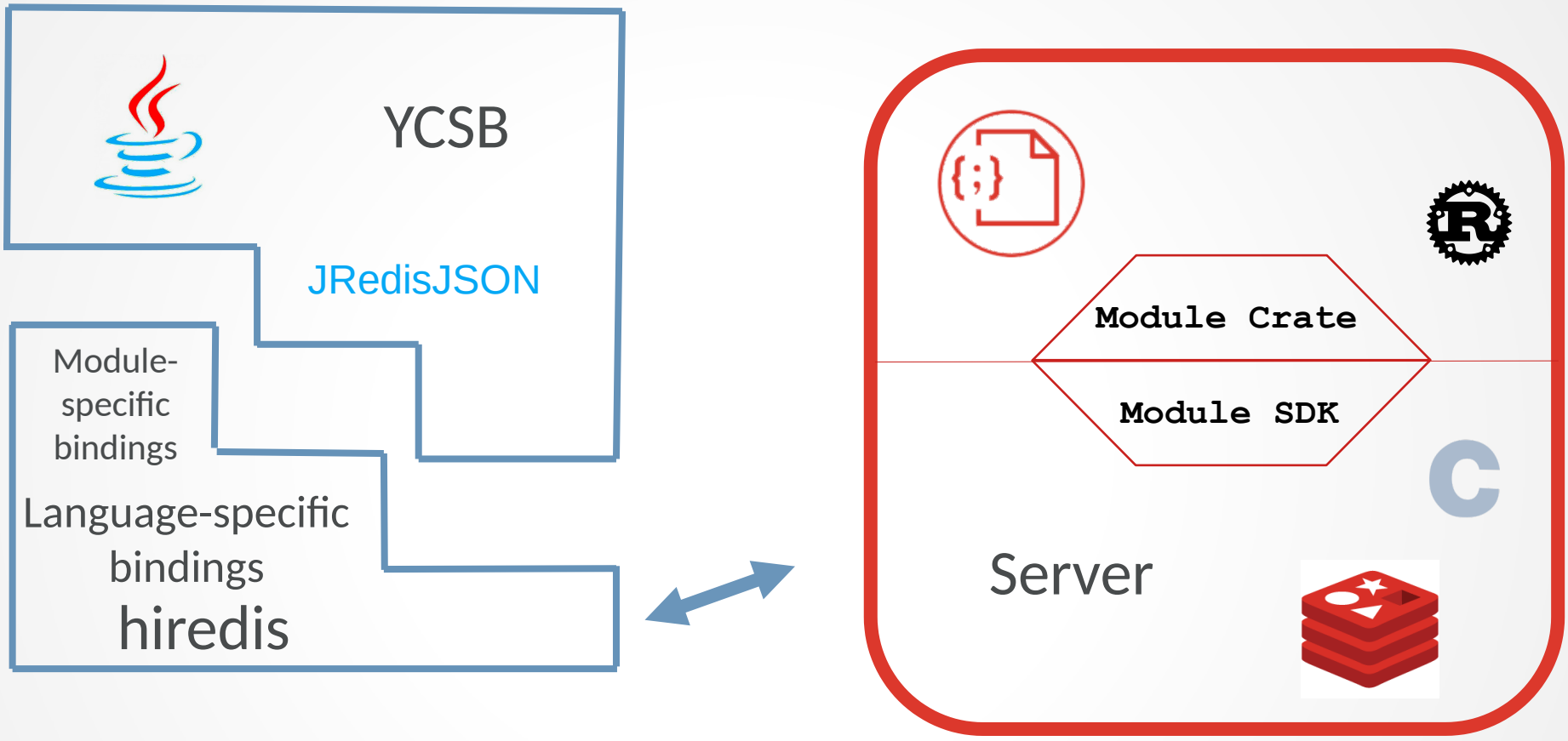
loadmodule redisjson

- Document DB extension: ECMA-404 compliant module
- Typical commands: json.set, json.get, json.arrappend, json.arrinsert
- Navigation via JSONPath:
  - .foo.bar <=> foo ['bar'] <=> ['foo']['bar']
  - Wildcard support

redis-server -h



bin/ycsb


YCSB
JRedisJSON
Module-specific bindings
Language-specific bindings
hiredis
Module Crate
Module SDK
C
Server

cat /etc/motd

- Original C-based implementation: ~ 5.2 kLoC
- New implementation in Rust: ~ 3.2 kLoC
- Main drivers:
  - Future technology stack
  - Less technical debt => lower QA effort => lower TCO
  - Shorter TTM

man rustc

- Background:
  - Team with diverse dev background
  - Main reason for C as first choice: in line with server implementation
- Lessons learned:
  - ✘ Steep learning curve
  - ✔ Comprehensive ecosystem
  - ✔ Responsive community
  - ✔ Toolchain support

info ycsb

- Yahoo Cloud Serving Benchmark: standard Java-based framework
- Pluggable backend for DB integration:
  - Already contains huge variety of NoSQL i/fs
  - Integration of new DB: thin interface for client library
- Relevant workloads: A (R/W: 1/1), B (R/W: .95/.05), F (read-modify-write)

time ycsb

Workload	A			B			F		
# of threads / database	1	4	8	1	4	8	1	4	8
Redis	62.19	31.41	30.14	53.99	32.04	30.13	53.28	30.28	30.49
RedisJSON	75.27	44.26	42.30	73.73	46.30	43.86	75.71	46.63	41.90
RedisJSON2	81.19	49.88	49.32	83.22	52.25	48.43	73.72	44.66	42.54

Benchmark spec:

- Stock Eoan on Dell XPS 13 (i7-7560U @ 2.40GHz, 16 GB RAM, 512 GB SSD) with 1M records
- Redis 5.0.5
- RedisJSON (git clone from 1/2/2020), RedisJSON2 (1/3/2020)
- No persistence

apropos doc_db | wc

- RedisJSON vs. friends:
  - Redis extension => in-memory
  - Focus: performance
  - CAP theorem: flexibility powering variety of use-cases
- Outlook:
  - Integration with RediSearch
  - Functionality improvements

apropos redis

- redis.io: Redis documentation
- redisjson.io: JSON module
- github.com/RedisJSON: RedisJSON + RedisJSON2
- github.com/antirez/redis: Redis
- github.com/brianfrankcooper/YCSB: Yahoo Cloud Server Benchmark (PR for RedisJSON pending soon :-)
- university.redislabs.com: Redis Labs university

Questions?

Thank you!

Dr. Christoph Zimmermann

monochromec at <ignore>space</ignore>gmail<dot></dot>com