DISTRIBUTED CONFIGURATION WITH CLOJURE

AGENDA

- Definition
- Key Aspects
- Available Solutions
- Can we do better?
- Demo

WHO?

- @reborg
- https://github.com/reborg
- Hacking @ Mailonline - London
- “Clojure Weekly” @ http://reborg.net

DEFINITION

Configuration is any “external handle” that greatly affects the application behaviour without the need for re-writing it.

EXAMPLES

- environments (int, prod, test etc.)
- user based (e.g. account settings)
- multi-tenant applications (blurring into proper feature)
- feature toggles (per-request on/off checks)
- A/B tests (toggling between two features)

VALUE LIFETIME

- Bootstrap (no need for “if” statements)
- Scheduled time (e.g. daily)
- Real-time (requires condition checks)

STORAGE

- Sources (same as hosting language)
- External files (potentially dedicated format)
- Relational DB (easily accessible, real-time)
- Key-Value stores (separation from other application data)
- Dedicated service (just for config)

ORGANISATION

- Single file (or unit)
- Multiple files/units (might need merge for reading)
- Relational (tables)
- Hierarchical

CACHING

- Boost performance for reading configuration
- Needs to be invalidated on config changes
  - Active check on timestamps
  - TTL entries in the cache
  - Pull: on-demand read, scheduled read
  - Push: nohup, queues, custom sockets

DISTRIBUTION

- For clustered applications
- Changes need to be distributed across the cluster
- Changes need to happen roughly at the same time
- Local caches might need explicit invalidation

VERSIONING

- A must have for real-time configurations
- Need for tracking/auditing changes
- Revert when necessary
- “Diffing”: what is about to change, what was changed

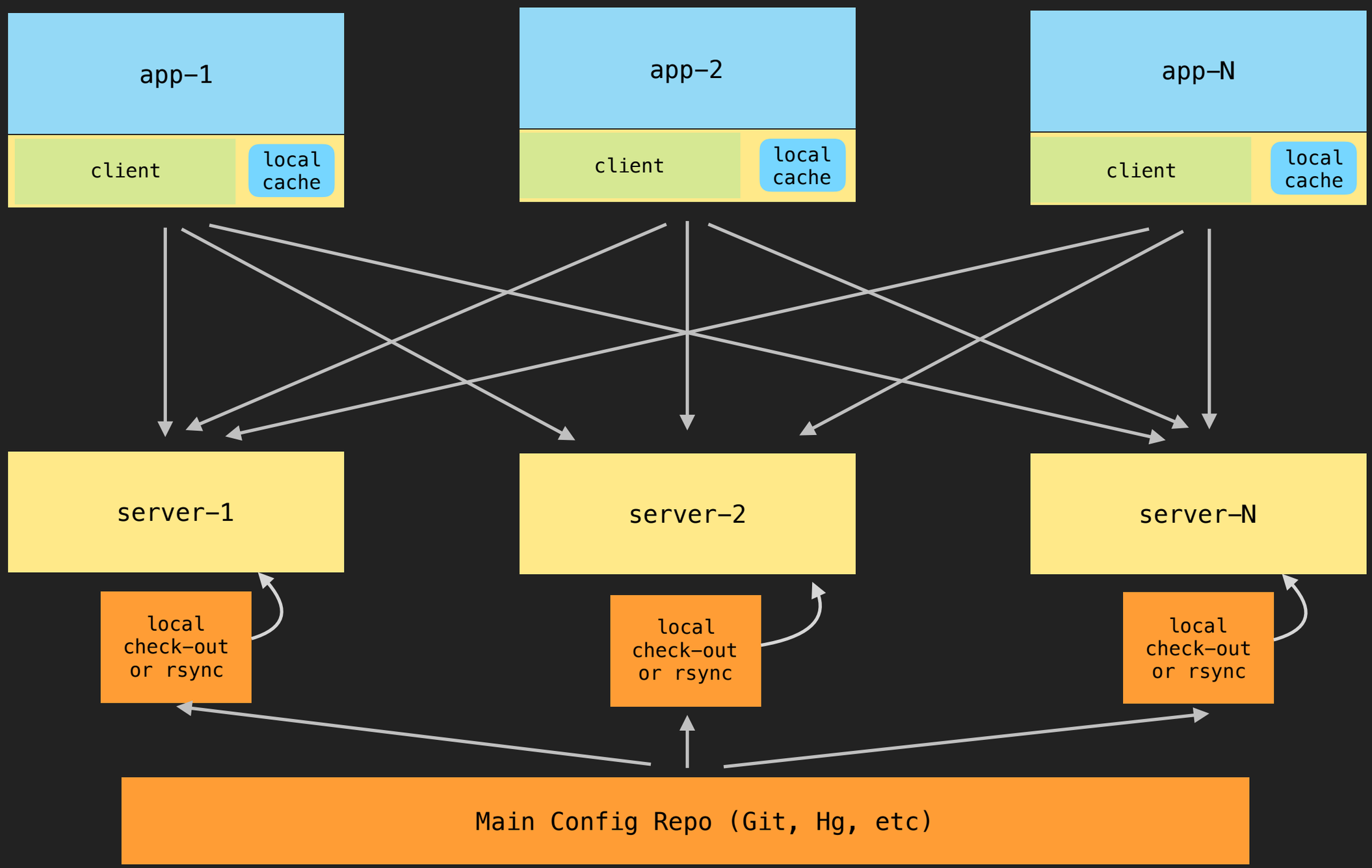
COMPARING OPTIONS

	Plain text	MySQL	Redis	Etcd
Updating	Custom	Triggers	Keyspace notifications	Watchers
Versioning	Any VCS	Custom schema	custom time-series	custom time-series
Diffing	Easy	Custom	Custom	Custom
Organisation	Basic	Tabular	Hierarchical/Tabular	Hierarchical/Tabular
Distribution	Custom	Built-in	Built-in	Built-in
Format	Any	Binary/Proprietary	Binary/Proprietary	Binary/Proprietary
Dedicated?	Yes	No	No	No

CAN WE DO BETTER?

- Plain Text (versioning, diffs, any format, dedicated)
- Hierarchical model (flexible configuration)
- Push model (immediate reaction to changes)
- Dedicated to configuration only
- Distributed, Resilient, Scalable

PROPOSED MODEL



WHY FUNCTIONAL?

- Immutable data structures perfect fit for versioning
- Built-in CAS semantic for local cache changes
- “Liveness” of the system allows instant re-evaluation
- Easy concurrency for distributed environments

CLOJURE

- Aleph/Manifold to deal with WebSockets (TCP or UDP)
- Java.NIO watch service for file changes notification
- Atoms for local caches and concurrent updates
- Metadata (stores information on raw-sources)
- Runs on any Java install

WHAT'S IN THE DEMO?

- 4 servers, 2-8 clients
- configuration folder: `fluorinedemo/apps`
- clients subscribe to `apps/test-json` subfolder
- configuration sent on client startup
- file-system as a tree, merged into hash-map
- will change one configuration parameter in one file

CURRENT SOLUTION TECHNICALITIES

- keep track of reconnecting clients
- keep-alive pings
- client local caches
- supporting edn and json
- JavaScript (and others) Client: WIP
- code and demo: https://github.com/reborg/fluorine

~ FIN ~

PDF SLIDES AVAILABLE: HTTP://TINYURL.COM/ZYAE7U3