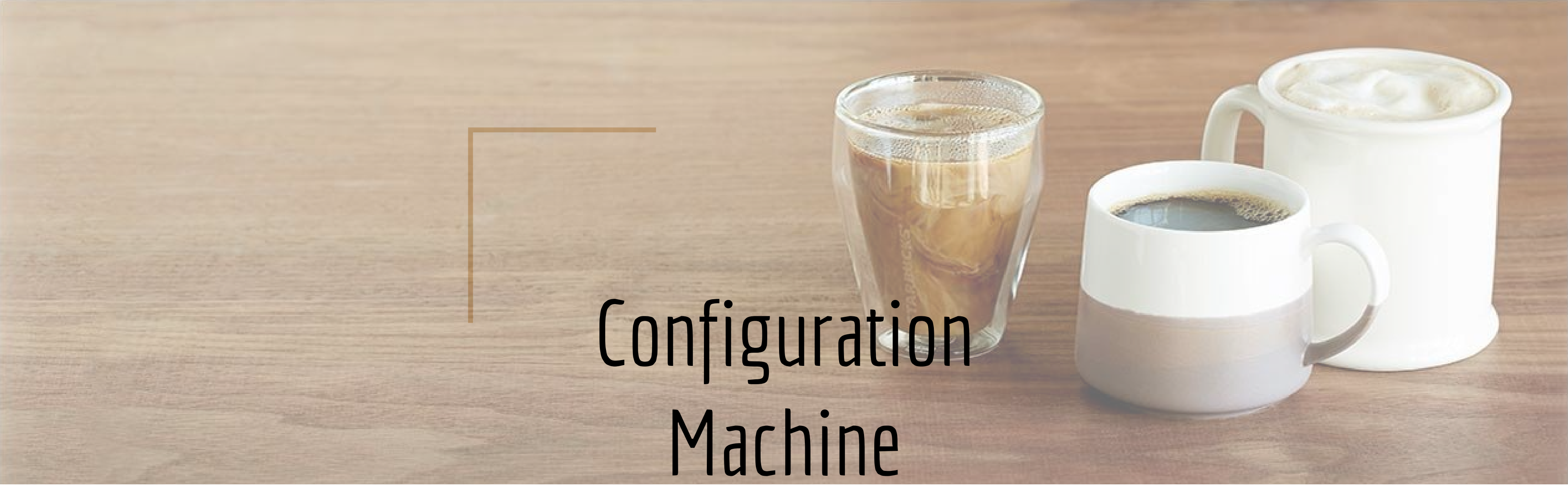
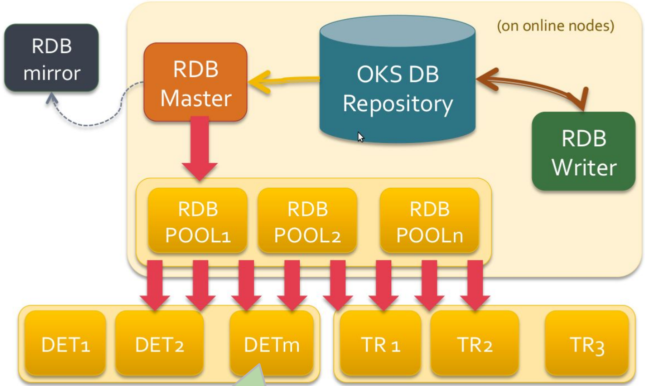


Configuration Machine

Dr. Leonidas Georgopoulos
(Fellow at CERN ATLAS DAQ
CC-WGRP)

Context

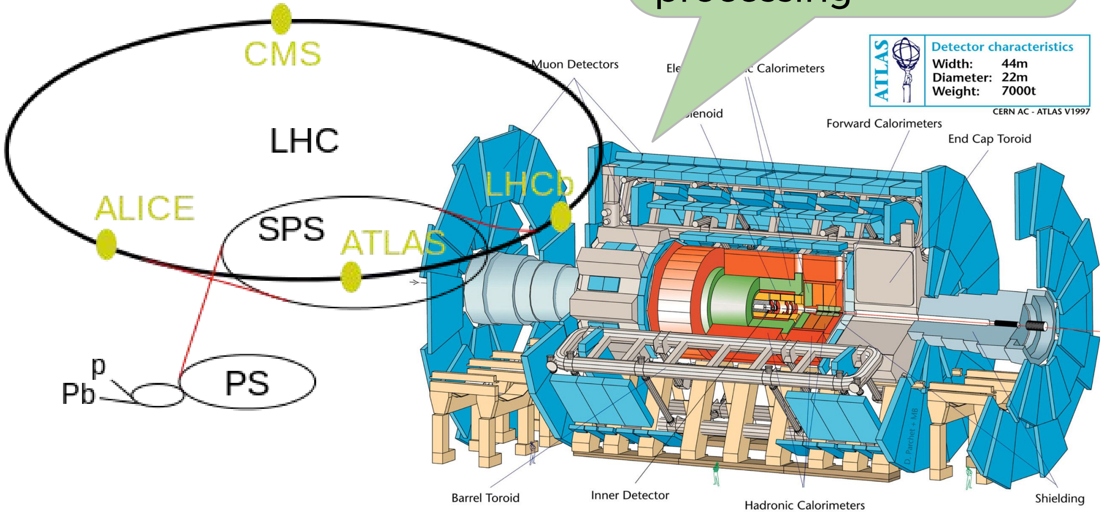
(**A**) (**T**)oroidal [*(**L**)arge Hadron Collider*] (**A**)pparatu(**S**)



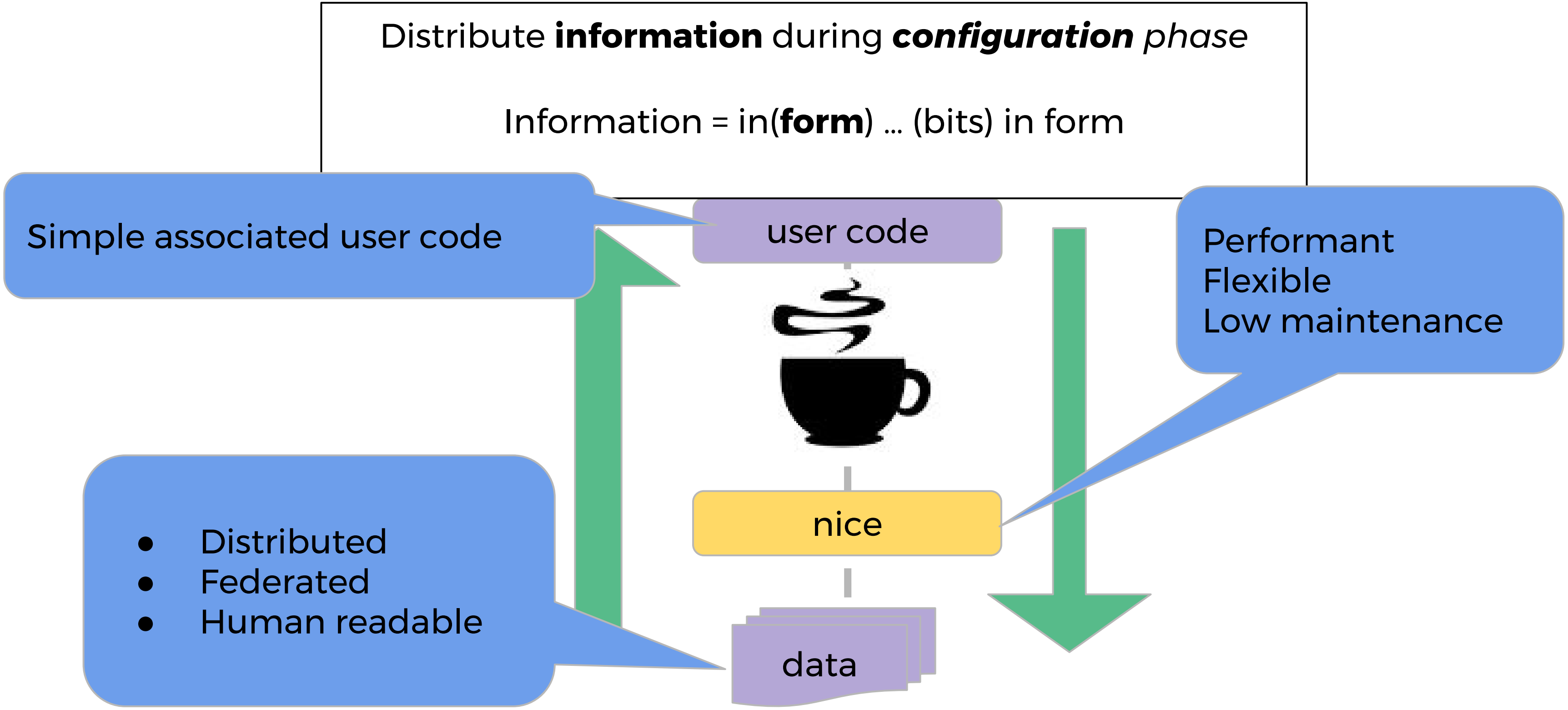
Distribute configuration in a few seconds utilizing a system of proxies and servers over ipc

Needs be configured, coordinated, and monitored

Event data from the detector is extracted to a several thousand cpus server farm for processing



[Configuration database Evolution]



Distribute **information** during ***configuration*** *phase*

Information = in(**form**) … (bits) in form

Simple associated user code

user code

nice

data

Performant
Flexible
Low maintenance

- Distributed
- Federated
- Human readable

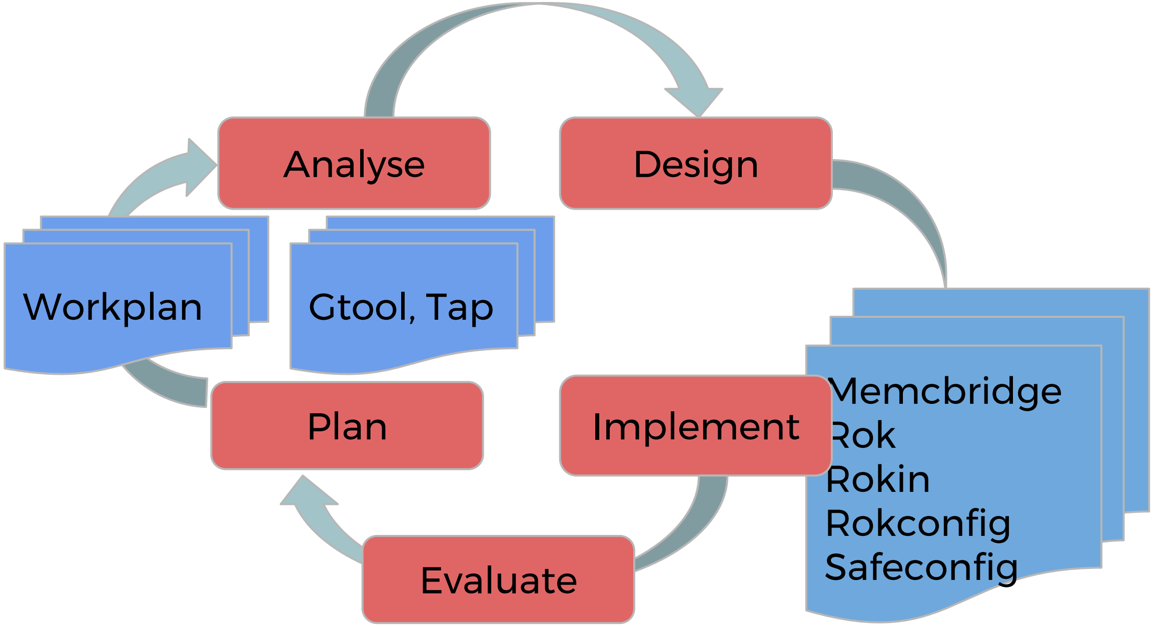
Configuration Database Evolution

A two year project involving four phases:

A. Plan (interface with ...)
B. Analyse (data,system,user)
C. Design (solution,implement)
D. Implement (code,test)

*Evaluate *Buyin

Different artifacts in each phase:
documents, tools, libraries, editors



Online configuration database ecosystem

Ingredients:

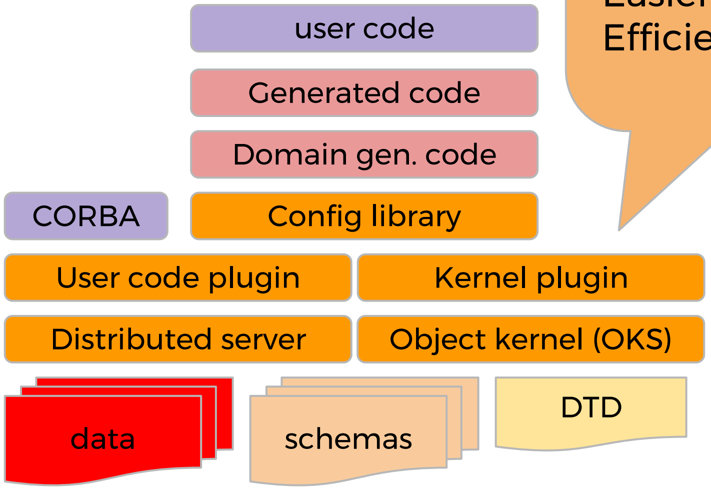
- Data files
- Schema files
- Libraries
- Servers
- Code generators
- Generated code (data acquisition system and sub-detectors)
- Client side algorithms
- Python, C++, Java bindings

Purpose:

Provide data objects to processes in data acquisition system and sub-detectors

Solution traits:

Simpler
Easier
Efficient



user code

Generated code

Domain gen. code

CORBA

Config library

User code plugin

Kernel plugin

Distributed server

Object kernel (OKS)

data

schemas

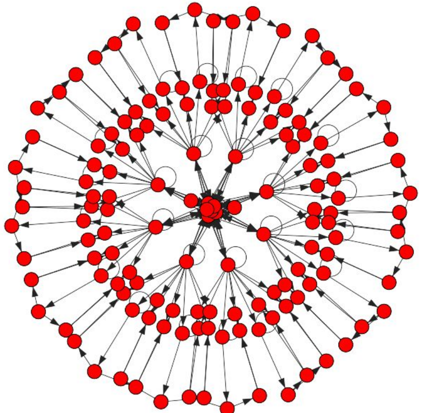
DTD

About the configuration ecosystem:

Aims to serve information to client side processes to instantiate objects as fast as possible

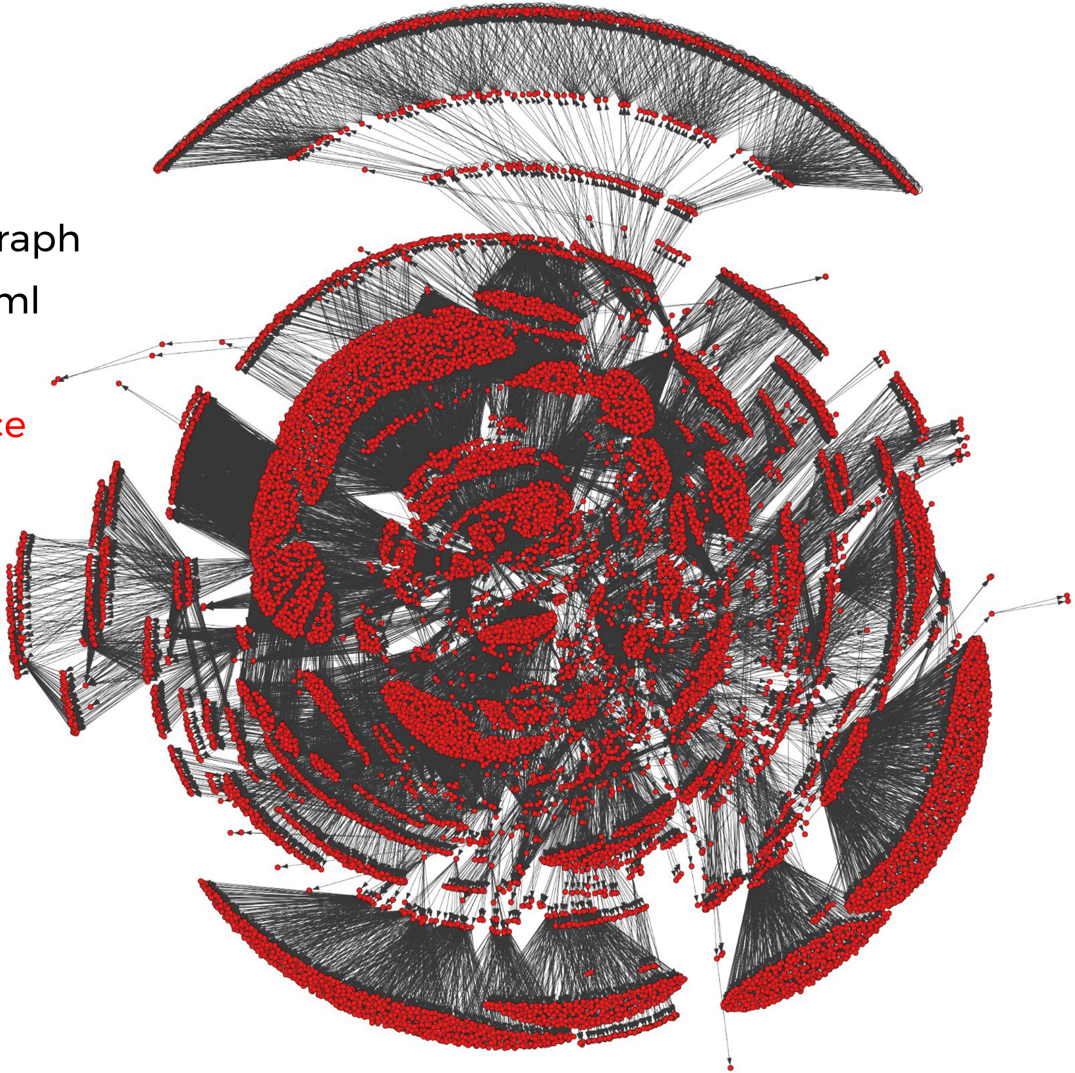
- Distributed
- Federated
- Object oriented
- Client side algorithms

Database and Access Analysis



gtool that extracts graph structure from oks-xml

- Dot per object
- Links if reference



- 1 Grows equivalent
- 60K vertices (i.e. objects)
- 140K edges (i.e. relations)
- 8K independent components !
- 300 components > 32 vertices
- 15 components > 100 vertices
- **1 component > 30000 vertices**

The ATLAS configuration database...

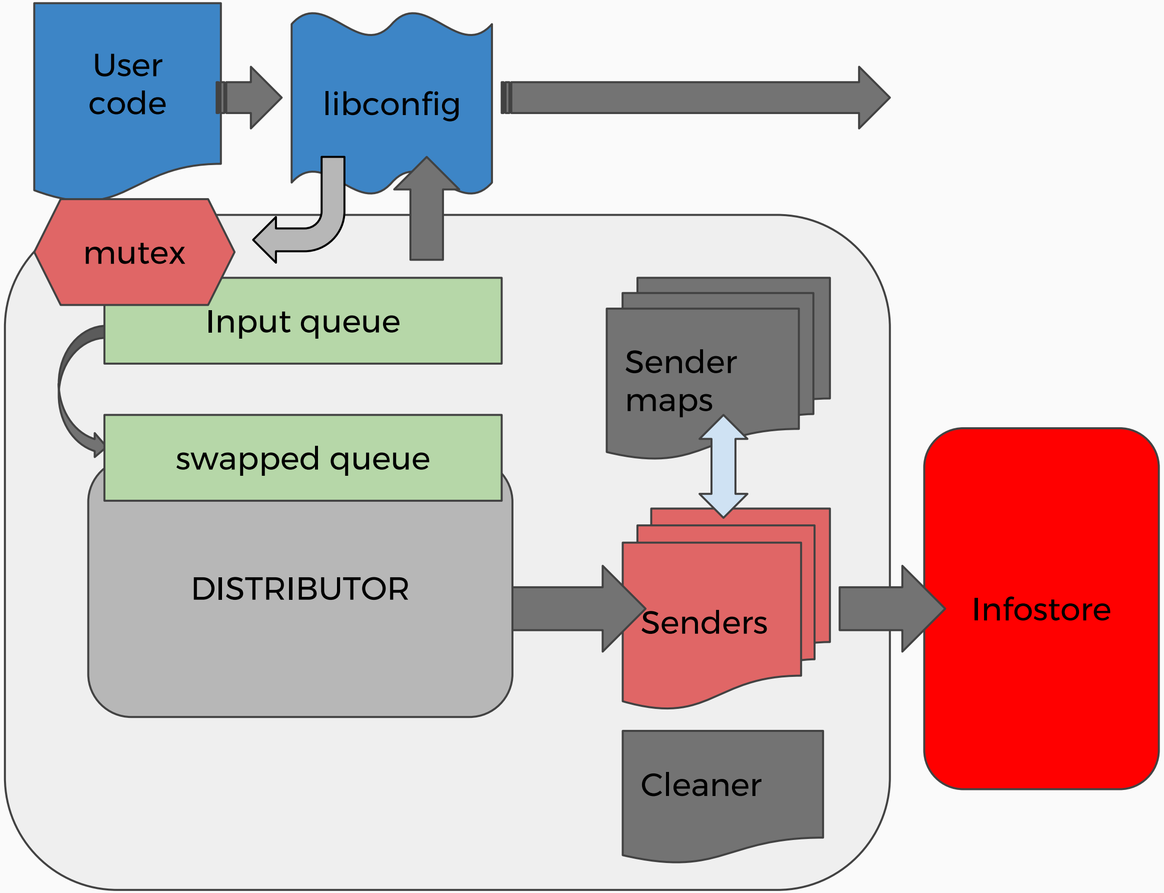
Tapping into it

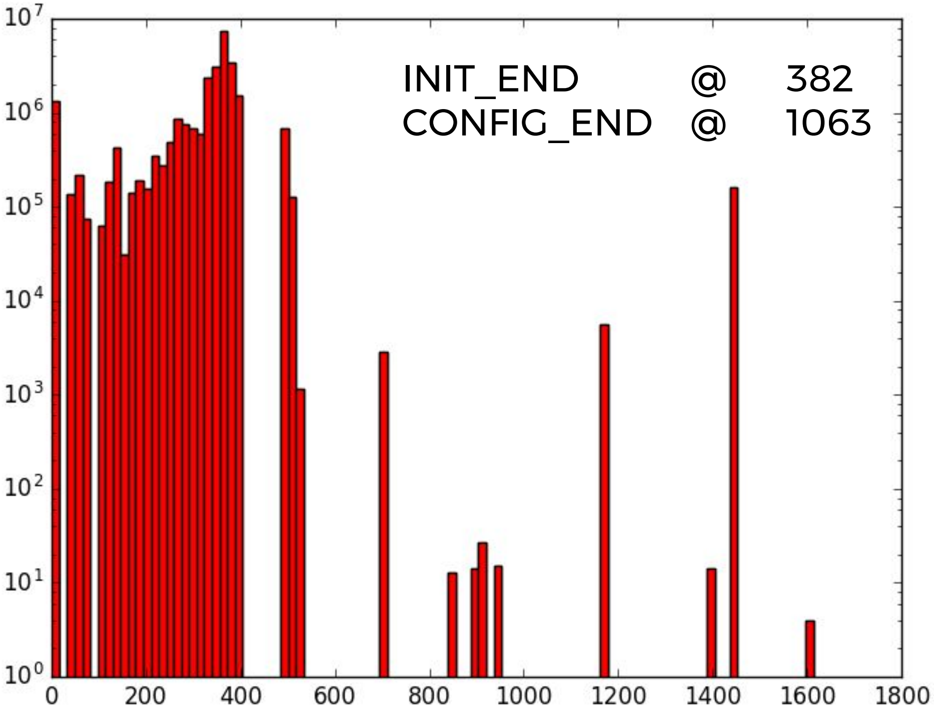
Daq::config::profile::archiver

- Archive application accesses
- Send to Information server

+ Microsecond accuracy
+ Lock-less distribution
+ Strong Queueing guarantee
+ No wait states
+ Handle backend unavailability

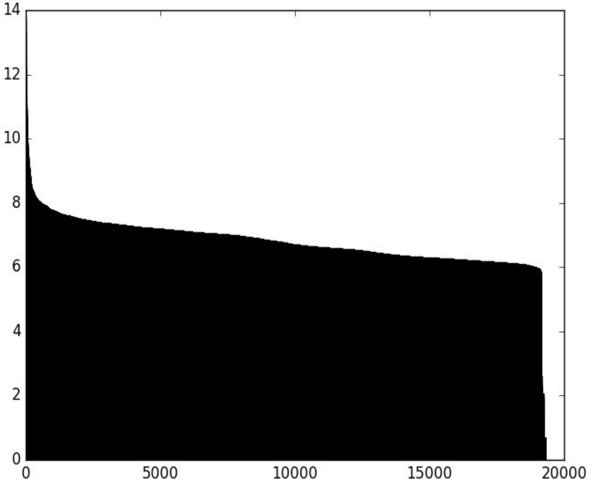
- Weak send guarantee





Configuration accesses over time

Accesses over objects



Observations:

- Access until CONFIG completion
- Uniform access over objects

Conclusion: Performance overall matters

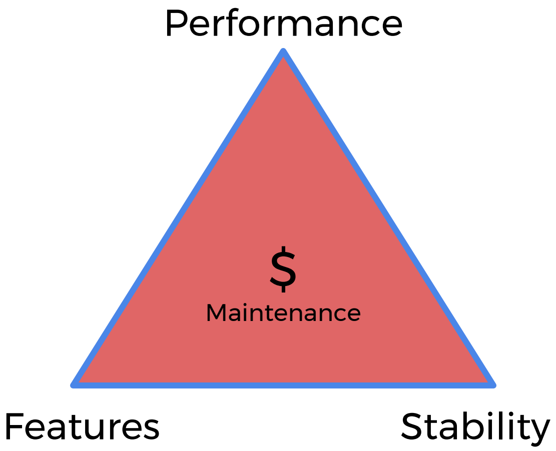
About ATLAS configuration data:

Hierarchical medium sized structure

- Tree like
- Forest like
- Few cliques
- Average diameter
- Few connected components
- Simple retrievals
- Mostly uniform accesses

Evolution: Simplification

Information source system design

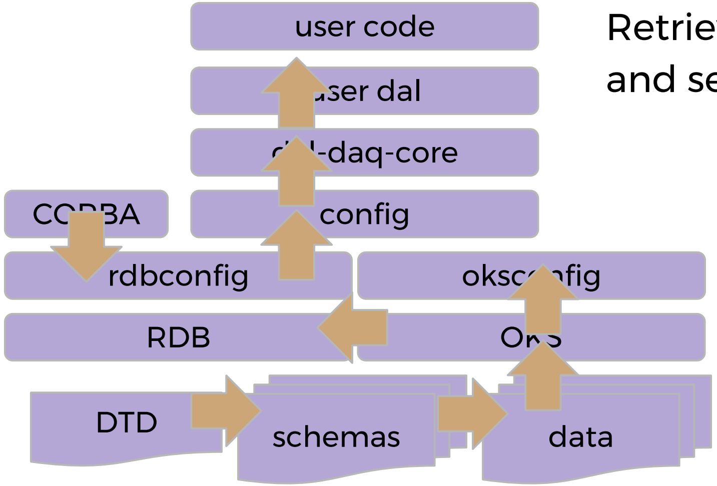


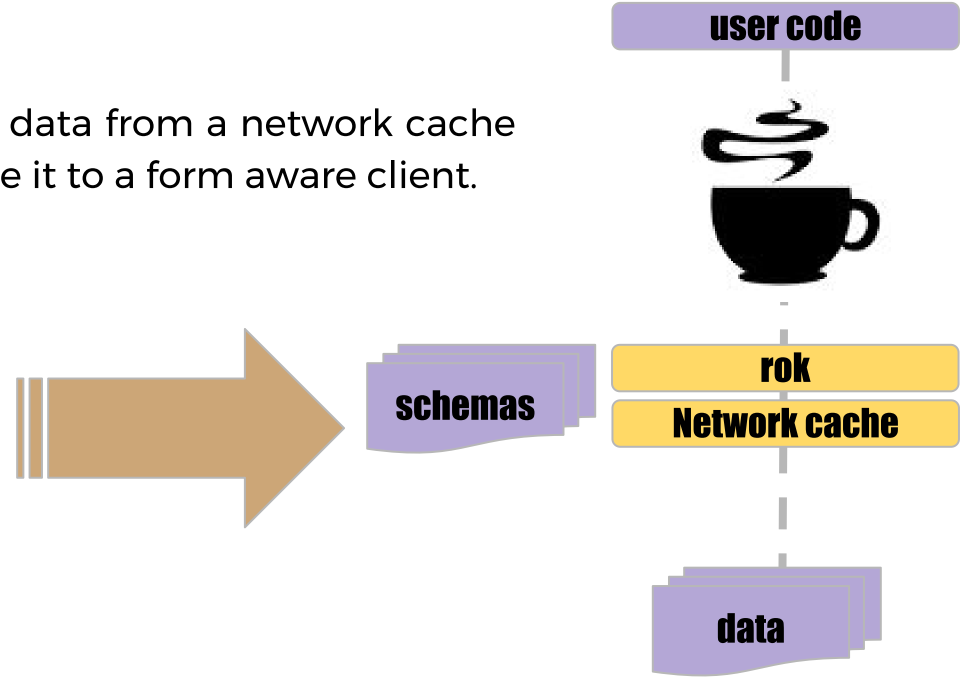
- stable
- relatively performant given custom optimizations
- features are scarcely implemented
- high complexity

Remove two out of three dimensions by choosing an already **performant** and **stable** solution.

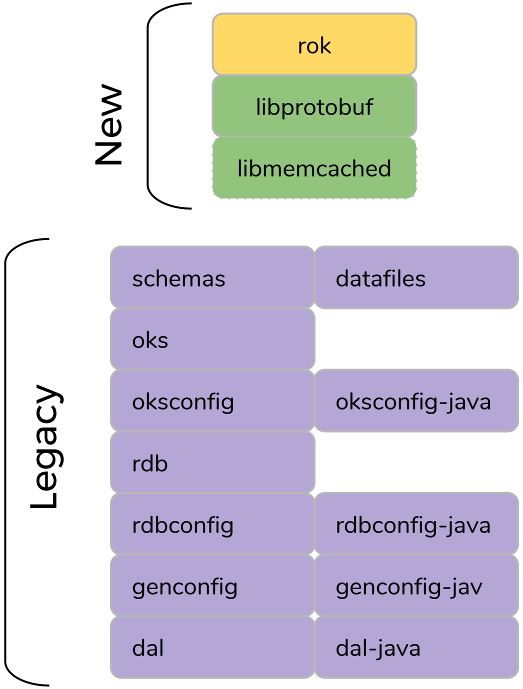
Simplified (Coffee machine) architecture

Retrieve data from a network cache and serve it to a form aware client.





Information ... components



New

rok

libprotobuf

libmemcached

Legacy

schemas

datafiles

oks

oksconfig

oksconfig-java

rdb

rdbconfig

rdbconfig-java

genconfig

genconfig-jav

dal

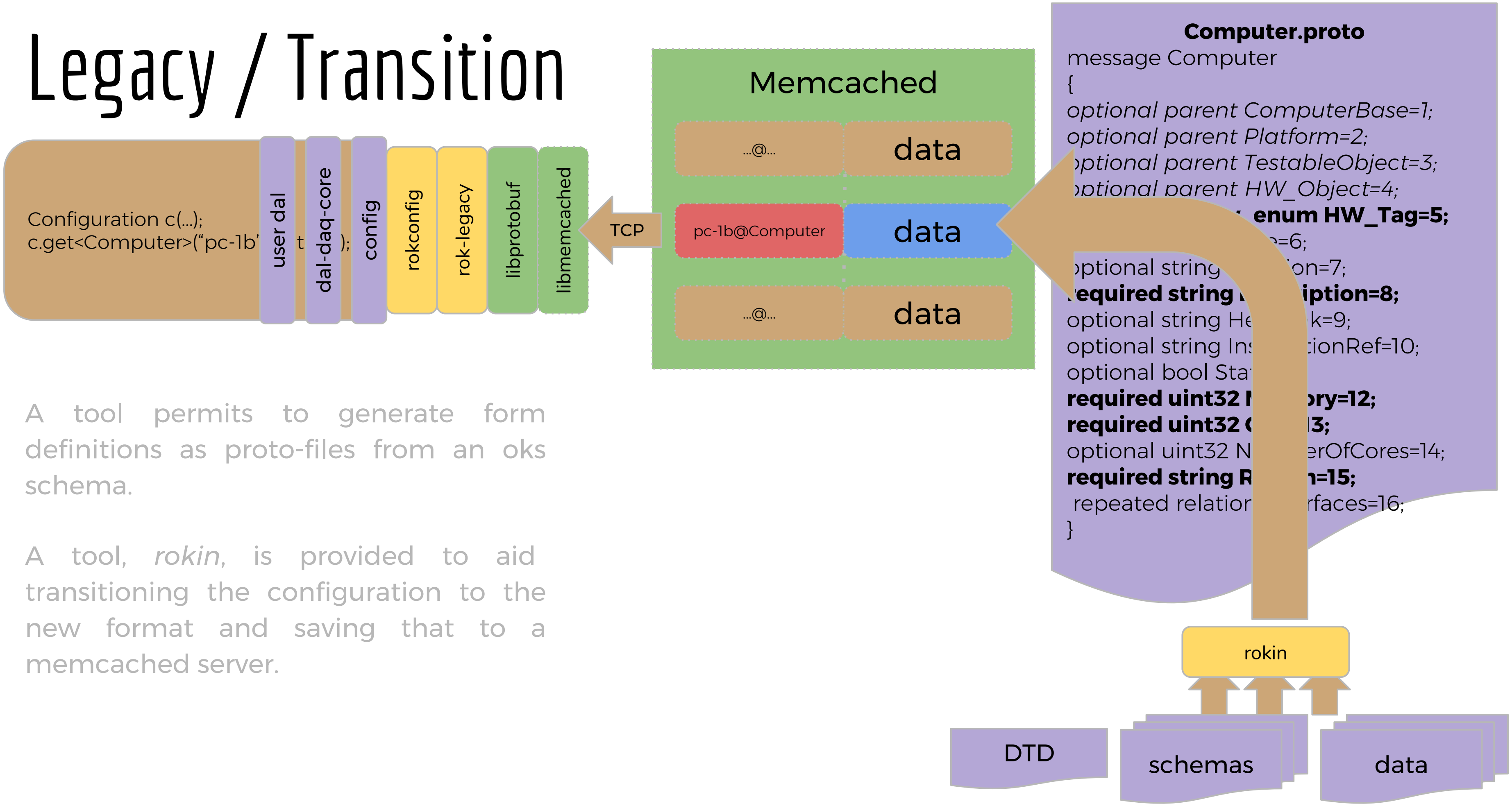
dal-java

Instead of maintaining the entire database infrastructure a thin wrapper and caching logic suffices.

Serialization , deserialization, and backend logic become independent.

Legacy plugin has been provided to aid data transition and migration.

Legacy / Transition



```
Configuration c(…);
c.get<Computer>("pc-1b"    t    );
```

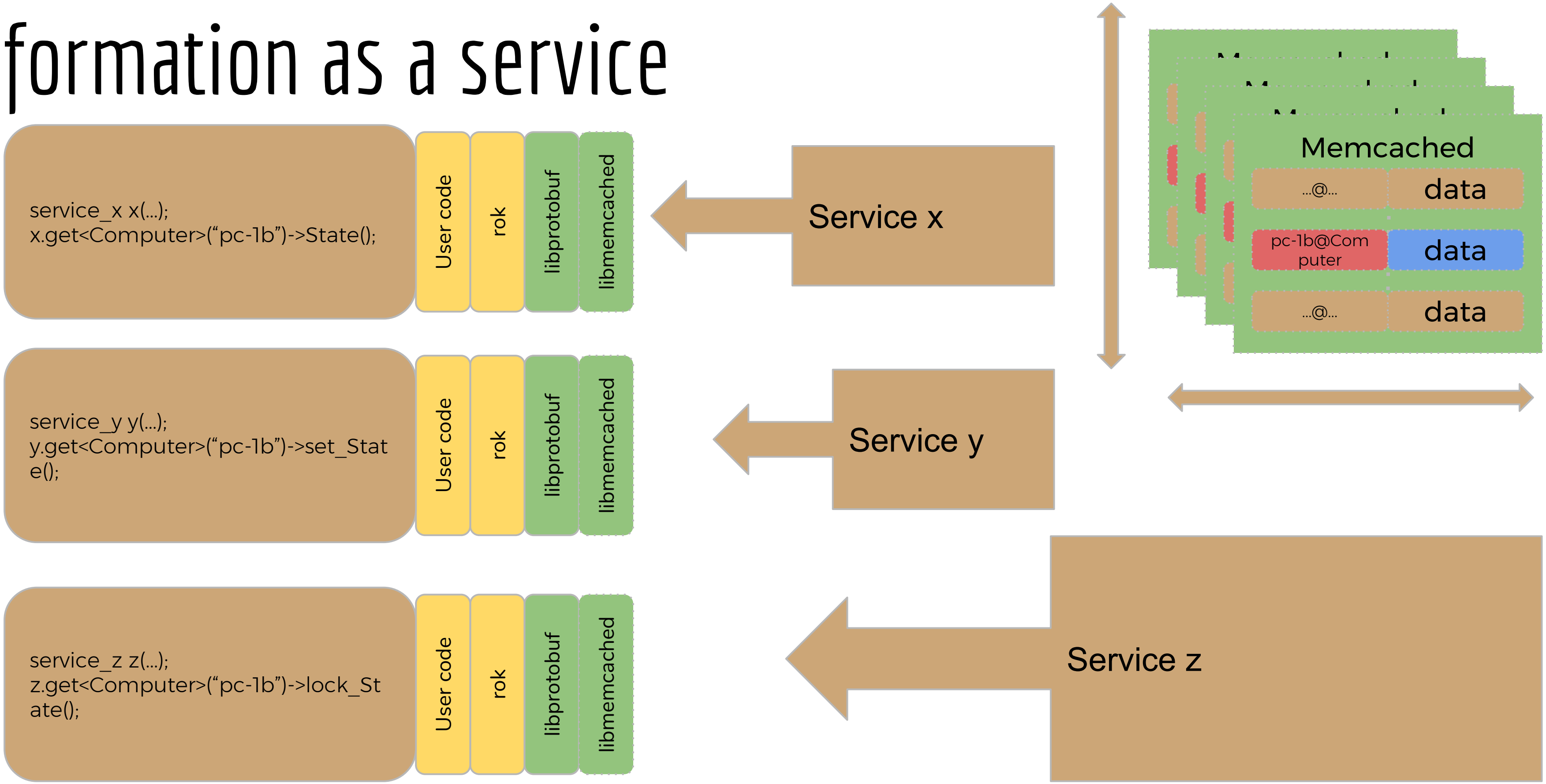
Computer.proto

```
message Computer
{
optional parent ComputerBase=1;
optional parent Platform=2;
ptional parent TestableObject=3;
ptional parent HW_Object=4;
              enum HW_Tag=5;
                         =6;
ptional string           on=7;
equired string          iption=8;
optional string He        k=9;
optional string Ins       tionRef=10;
optional bool Sta
required uint32 M         ory=12;
required uint32 C         13;
optional uint32 N         erOfCores=14;
required string R         n=15;
 repeated relation        rfaces=16;
}
```

A tool permits to generate form definitions as proto-files from an oks schema.

A tool, *rokin*, is provided to aid transitioning the configuration to the new format and saving that to a memcached server.

Information as a service

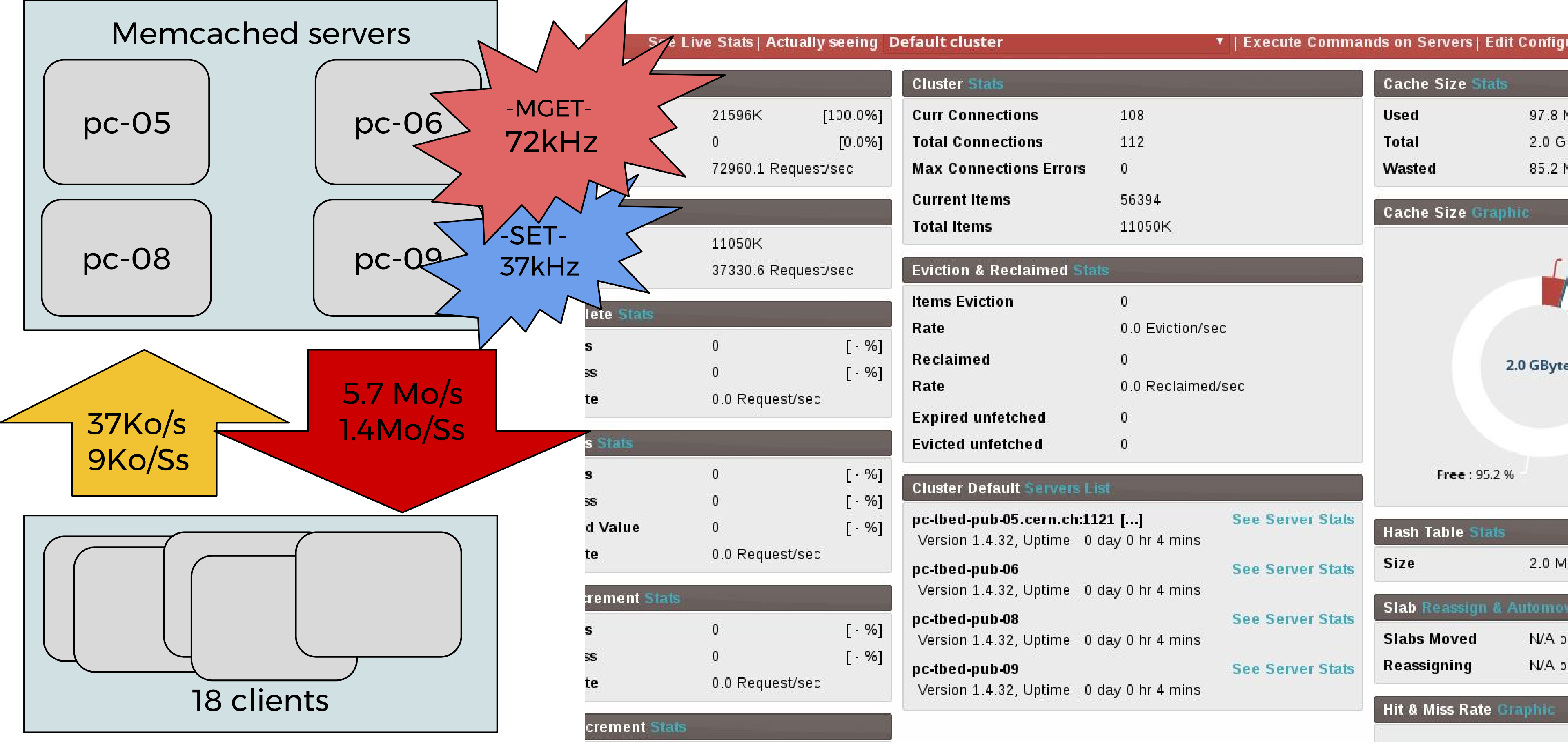


```
service_x x(...);
x.get<Computer>("pc-1b")->State();
```

```
service_y y(...);
y.get<Computer>("pc-1b")->set_State();
```

```
service_z z(...);
z.get<Computer>("pc-1b")->lock_State();
```

Performance



See Live Stats | Actually seeing Default cluster | Execute Commands on Servers | Edit Config

	21596K	[100.0%]
	0	[0.0%]
	72960.1 Request/sec	

	11050K
	37330.6 Request/sec

lete Stats

s	0	[- %]
ss	0	[- %]
te	0.0 Request/sec	

s Stats

s	0	[- %]
ss	0	[- %]
d Value	0	[- %]
te	0.0 Request/sec	

rement Stats

s	0	[- %]
ss	0	[- %]
te	0.0 Request/sec	

crement Stats

Cluster Stats

Curr Connections	108
Total Connections	112
Max Connections Errors	0
Current Items	56394
Total Items	11050K

Eviction & Reclaimed Stats

Items Eviction	0
Rate	0.0 Eviction/sec
Reclaimed	0
Rate	0.0 Reclaimed/sec
Expired unfetched	0
Evicted unfetched	0

Cluster Default Servers List

pc-tbed-pub-05.cern.ch:1121 [...] See Server Stats
Version 1.4.32, Uptime : 0 day 0 hr 4 mins

pc-tbed-pub-06 See Server Stats
Version 1.4.32, Uptime : 0 day 0 hr 4 mins

pc-tbed-pub-08 See Server Stats
Version 1.4.32, Uptime : 0 day 0 hr 4 mins

pc-tbed-pub-09 See Server Stats
Version 1.4.32, Uptime : 0 day 0 hr 4 mins

Cache Size Stats

Used	97.8 M
Total	2.0 G
Wasted	85.2 M

Cache Size Graphic

2.0 GByte

Free : 95.2 %

Hash Table Stats

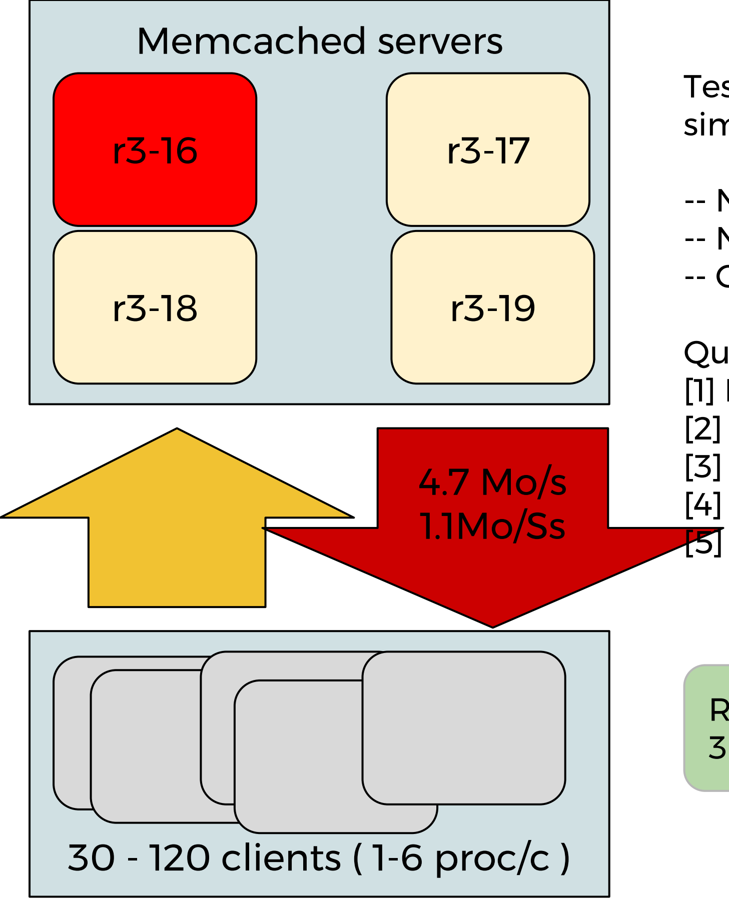
Size	2.0 M

Slab Reassign & Automov

Slabs Moved	N/A o
Reassigning	N/A o

Hit & Miss Rate Graphic

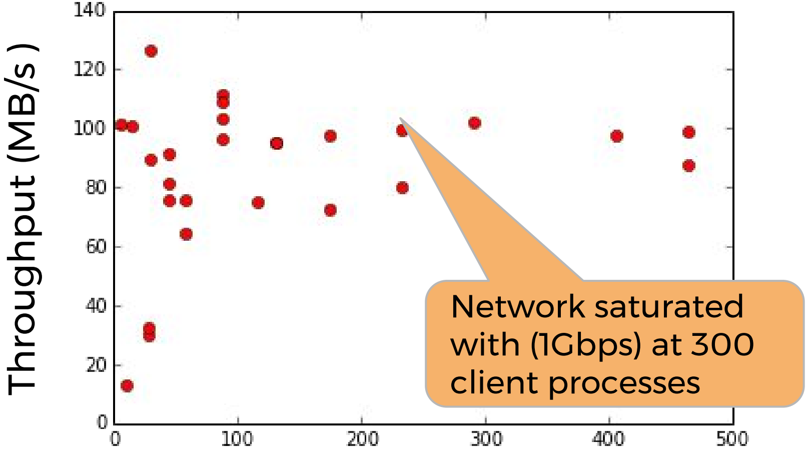
Peak into performance (1Gbps)



Test on a downscaled simulated environment

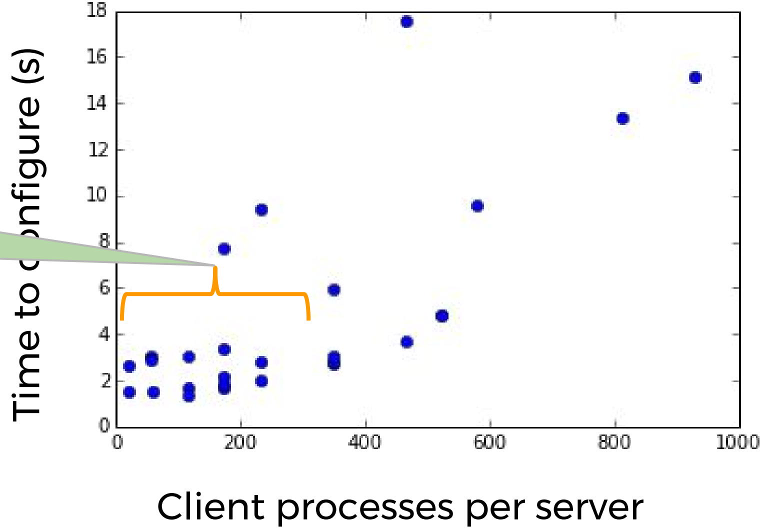
-- No optimizations --
-- No proxies --
-- Out of the box --

Query Mix :
[1] Full search
[2] Get range of keys
[3] Search missing key
[4] Get range of values
[5] Search across



Network saturated with (1Gbps) at 300 client processes

Region of interest:
300 clients / Gbps



Peak into performance

Scalability

[1] Performance scales linearly

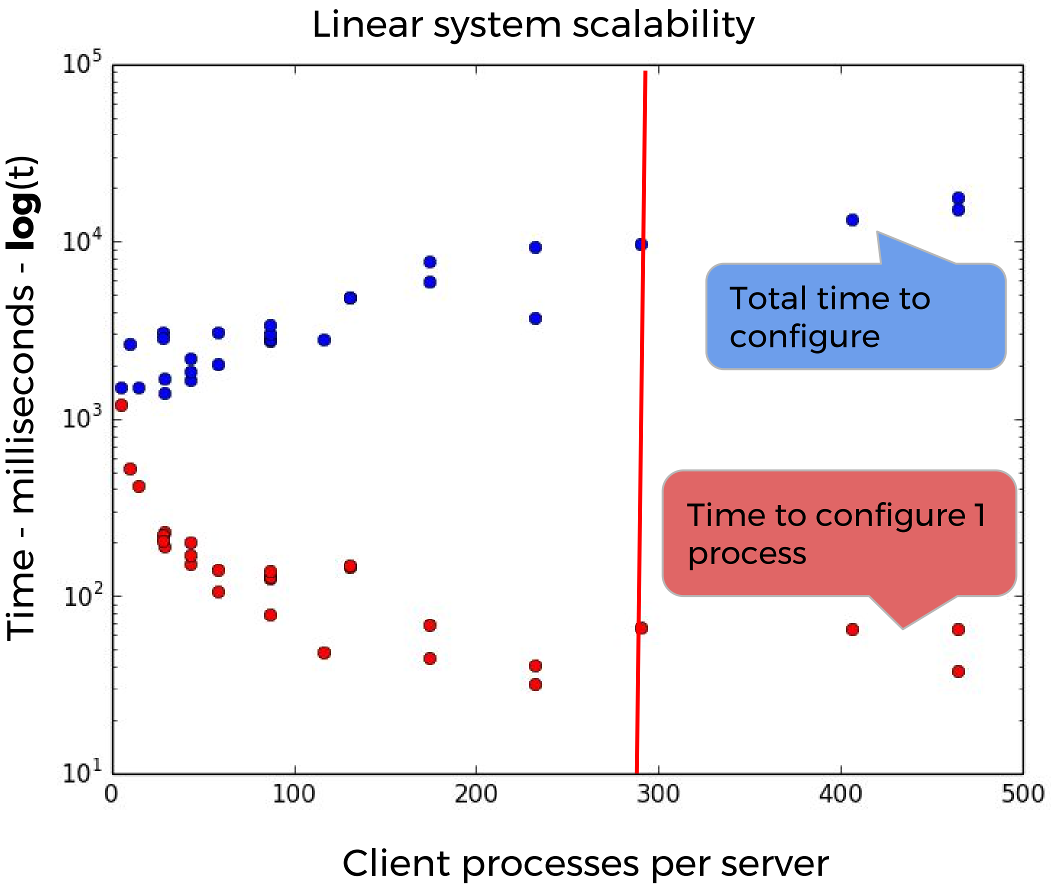
[2] Add more servers to increase throughput

[3] Low server utilization

[4] Network limited solution

[5] Max throughput 1/4Gbps per server client .

[6] The more clients the better (heavily multithreaded)



Overview:

A performant multipurpose solution

- Technology:

[External:] memcached + protobuf + TCP/IP

[Internal:] ROK

- Performance:

Network limited

- Scalability:

Linear horizontal + vertical

A first working implementation was evaluated on

[1] Stability
[2] Features
[3] Performance

and results demonstrate feasibility of the suggested multipurpose solution for the configuration service, based on well accepted technologies (memcached + protobuf).

Conclusion

APPENDIX

Artifacts

ROK [representational object kernel] is a replacement for the OKS [object kernel support]. It provides a simple implementation - schema - data independent kernel for retrieval and updating of information across a distributed processes:

ROKIN tool loads OKS data to a set of memcached servers (requires libmemcached)

GENPROTO a modified GENCONFIG to generate Google protobuf definitions from OKS-XML schema files:

ROKCONFIG is a plugin for libconfig to provide seemless access to legacy applications

MEMCBRIDGE offers simplified access to memcached servers through libmemcached .

SAFECONFIG provides safe and easy access to config , it is derived from work to make dbe robust

DBE, a qt based editor for the ATLAS configuration database

TAP , a library to archive object accesses on a distributed system

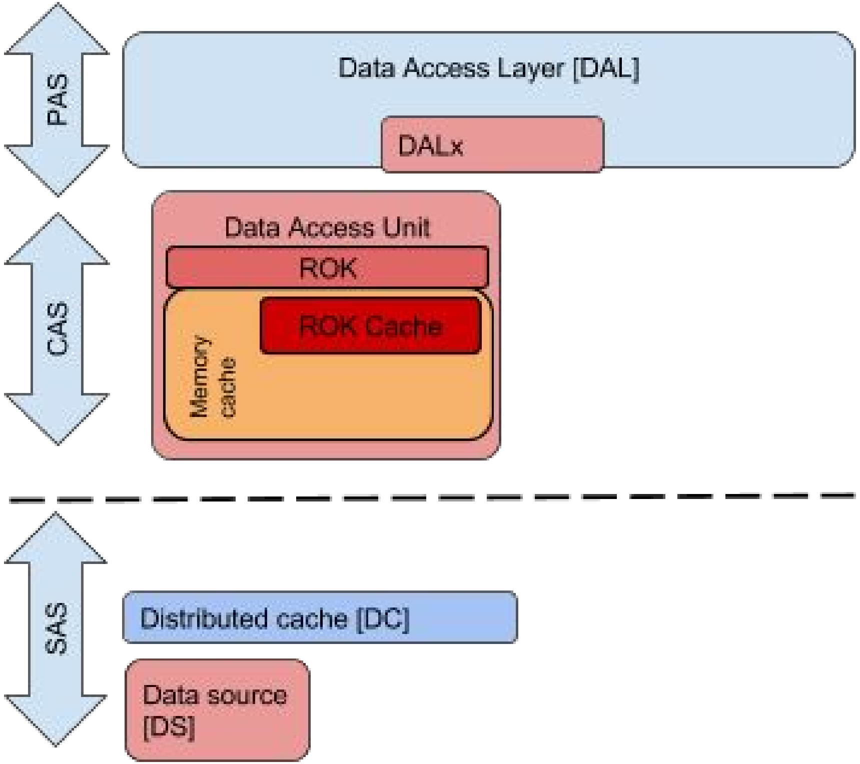
First working implementation

Three layered architecture:

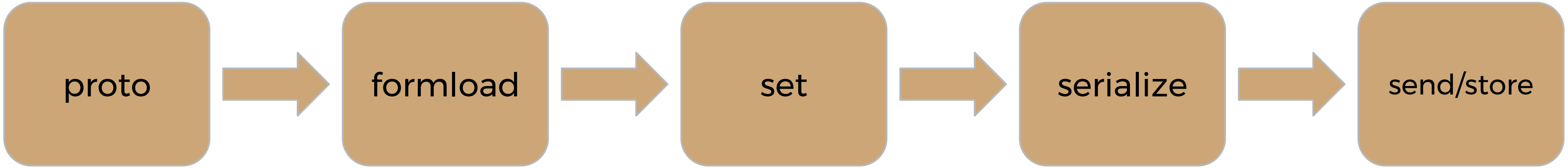
(1) PAS : Program access support
(2) CAS : Client access support
(3) SAS : System access support

Major components:

- ROK : Representational Object Kernel
- ROKIN : [ROK][IN]put
- DALx : Data access layer - x direction



Dynamic object creation



proto → formload → set → serialize → send/store

```
formloader<type_proto>::type_input finput{ // arguments type specified };

formloader<type_proto> f { finput };

f.set(formloader<type_proto>::type_attr_y , // ... some value );

// ... do more sets

std::string fserial = f.to<std::string>();
```