ConfD memory usage and performance improvements

Johan Bevemyr

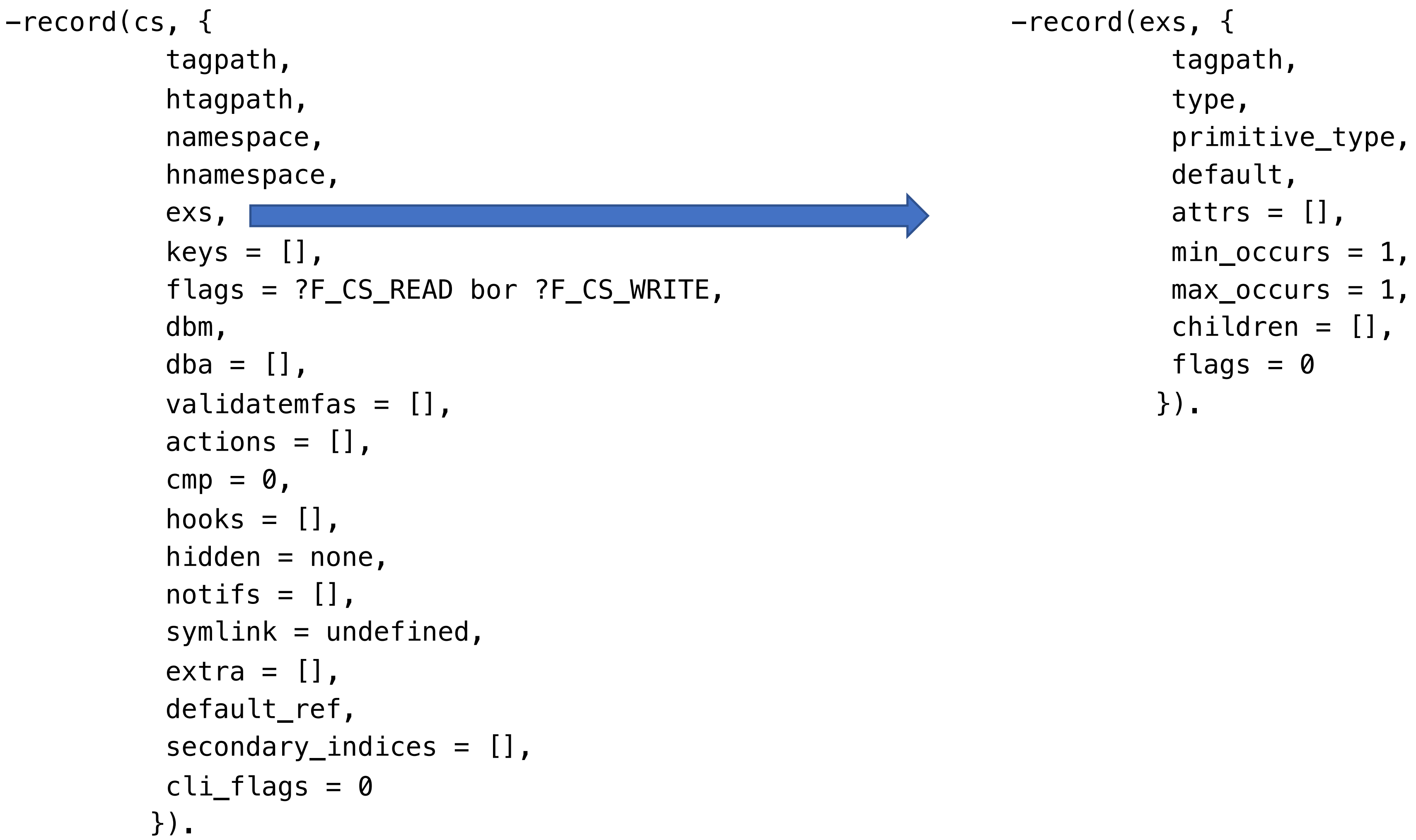
Schema Memory Usage

- Data models are growing
- A single device may have ~700 different models
- Models are rarely used, but must be present on device
- Different model “sets”

FXS files

- An FXS file is primarily a sequence of #cs{} records (ConfSpec), stored in chunks.
- Loading FXSs consists of
  - Reading #cs{} records from files and storing them in ETS tables, which are generic in-memory hash tables.
  - Resolving augmentations between name spaces.
  - An ETS table consists of records for a namespace, and all augmentation into that namespace.
  - In addition to #cs{} records there are #docs{}, #exs_type{}, and header information.

#cs{} record layout



```
-record(cs, {
          tagpath,
          htagpath,
          namespace,
          hnamespace,
          exs,
          keys = [],
          flags = ?F_CS_READ bor ?F_CS_WRITE,
          dbm,
          dba = [],
          validatemfas = [],
          actions = [],
          cmp = 0,
          hooks = [],
          hidden = none,
          notifs = [],
          symlink = undefined,
          extra = [],
          default_ref,
          secondary_indices = [],
          cli_flags = 0
         }).
```

```
-record(exs, {
          tagpath,
          type,
          primitive_type,
          default,
          attrs = [],
          min_occurs = 1,
          max_occurs = 1,
          children = [],
          flags = 0
         }).
```

A typical distribution

- revision 2018-08-10 of http://cisco.com/ns/yang/Cisco-IOS-XE-native

```
cs: 88.49 MiB
doc: 601.12 KiB
exs_type: 155.57 KiB
fxs_header: 1.34 KiB
info: 896 bytes
augments_header: 40 bytes
```

```
#cs{
 tagpath =
  [address,'ipv6-addrees','eid-cont','database-mapping',ipv6,
   service,'instance-list','instance-container',
   ['http://cisco.com/ns/yang/Cisco-IOS-XE-lisp'|lisp],
   router,native],
 htagpath =
  [1266954393,1103441164,1502191743,1365224135,1228132394,
   855380710,2088542428,608613614,
   [247644804|1303297170],
   532320551,472211213],
 namespace = 'http://cisco.com/ns/yang/Cisco-IOS-XE-native',
 hnamespace = 1270643900,
 exs =
  {exs,
   [address,'ipv6-addrees','eid-cont','database-mapping',ipv6,
    service,'instance-list','instance-container',
    ['http://cisco.com/ns/yang/Cisco-IOS-XE-lisp'|lisp],
    router,native],
   {'urn:ietf:params:xml:ns:yang:ietf-inet-types',
    'ipv6-address'},
   inetAddressIPv6,undefined,[],1,1,[],0},
 keys = [],flags = 618970019642690137449580070,dbm = cdb,
 dba = [],validatemfas = [],actions = [],cmp = 1,hooks = [],
 hidden = none,notifs = [],symlink = undefined,extra = [],
 default_ref = undefined,secondary_indices = [],
 cli_flags = 524288}]
```

Compaction

- Remove redundant information
  - Multiple copies of tagpath
  - htagpath and hnamespace are computed
  - Default values in record

```
-record(cs_trim, {
          key,
          dmap,
          record
         }).
```

- Performance penalty on sync & show ~1-2% slowdown.

Compaction (cont.)

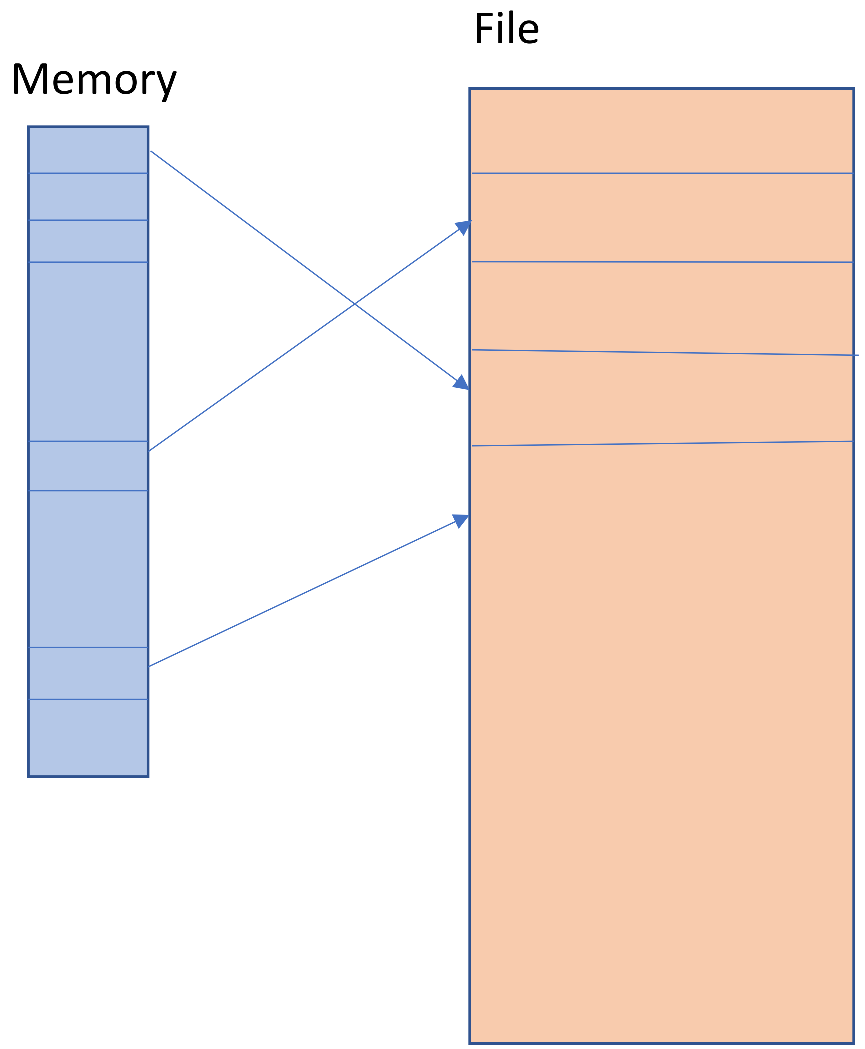
```
#cs_trim{
    key =
        ['Cellular','update-source',neighbor,
         ['http://cisco.com/ns/yang/Cisco-IOS-XE-bgp'|bgp],
         router,native],
    dmap = 229264,
    record =
        {'http://cisco.com/ns/yang/Cisco-IOS-XE-native',
            {exs_trim,1970,
                {'http://www.w3.org/2001/XMLSchema',string},
                string,0},
            618970019642690345755493894,cdb,
            [{cli_allow_join_with_value_r,154742504910672534362390528,
                 []}],
            20769187434139310514121985316880384}}
```

Version	ETS table size	ETS table size (compressed)	Ratio
ConfD 6.7	93.82 MiB	56.03 MiB	1
Diff-dep bugfix	82.90 MiB	50.31 MiB	1.1
#cs_trim{}	36.70 MiB	23.54 MiB	2.4

Lazy loading

- Flush all #cs_trim{} records to disk, and only store
  - Path
  - Position in file
- Read records on demand and store in ETS table.

```
-record(cs_file, {
          tagpath,
          pos
         }).
```



Lazy loading (cont.)

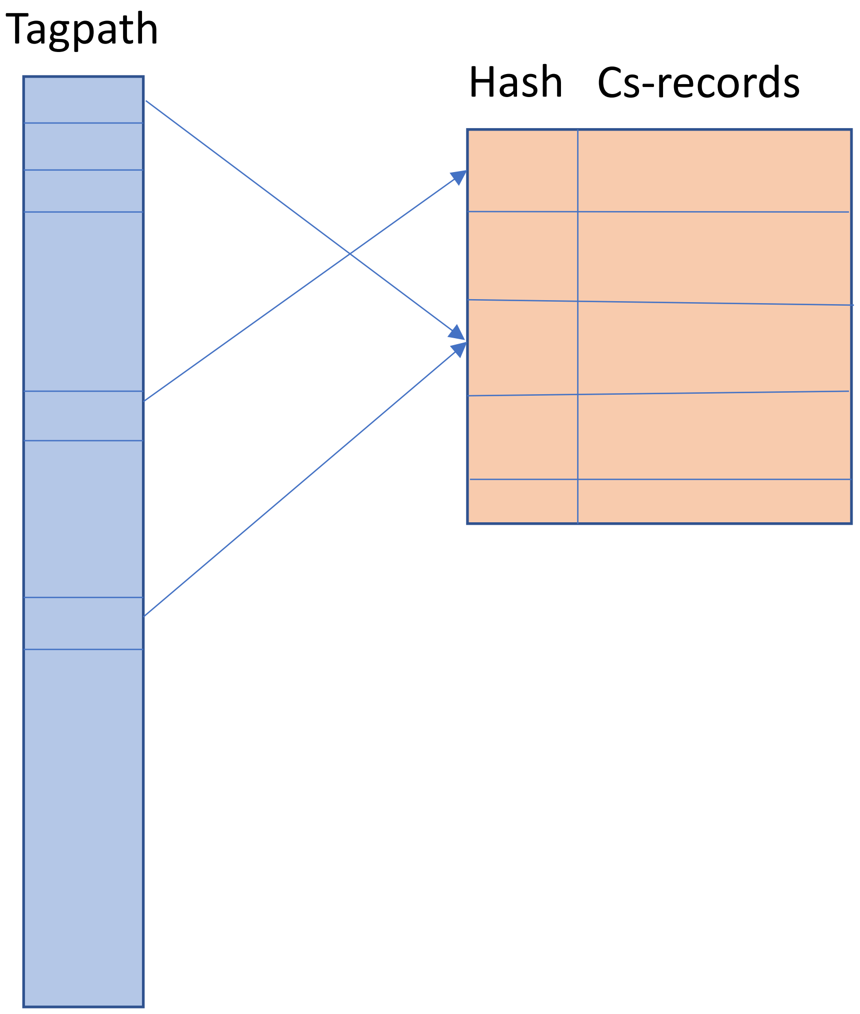
Version	ETS table size	ETS table size (compressed)	Ratio
ConfD 6.7	93.82 MiB	56.03 MiB	1
Diff-dep bugfix	82.90 MiB	50.31 MiB	1.1
#cs_trim{}	36.70 MiB	23.54 MiB	2.4
#cs_file{}	15.10 MiB	15.79 MiB	3.5

- Pros
  - Use less memory
- Cons
  - Require a baking phase
  - Slower access
  - Somewhat unpredictable memory usage
  - Requite disk space

De-duplicate

- Only 10% of #cs{} records are unique when removing tagpath
  - Store #cs_trim{} records under a hash of record
  - Store #cs_ref{} records mapping tagpath to #cs_trim{}

Version	ETS table size	ETS table size (compressed)	Ratio
ConfD 6.7	93.82 MiB	56.03 MiB	1
Diff-dep bugfix	82.90 MiB	50.31 MiB	1.1
#cs_trim{}	36.70 MiB	23.54 MiB	2.4
#cs_file{}	15.10 MiB	15.79 MiB	3.5
#cs_ref{}	19.68 MiB	17.73 MiB	3.2



Path Sharing

- Large parts of data is now tagpaths
- Tag paths are paths from root

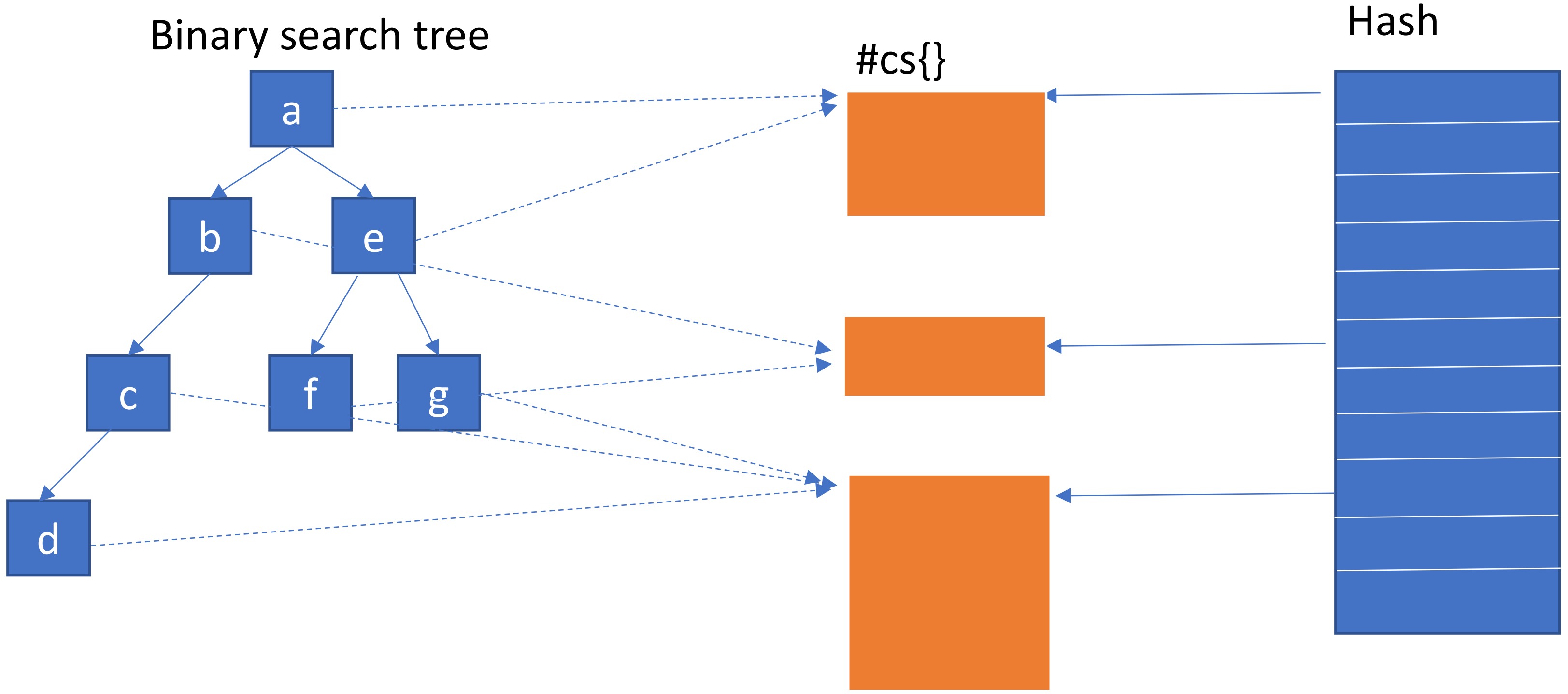
```
[]
[native]
[native, router]
[native, router, ['http://cisco.com/ns/yang/Cisco-IOS-XE-bgp'|bgp]]
[native, router, ['http://cisco.com/ns/yang/Cisco-IOS-XE-bgp'|bgp], neighbor]
[native, router, ['http://cisco.com/ns/yang/Cisco-IOS-XE-bgp'|bgp], neighbor, 'update-source’]
[native, router, ['http://cisco.com/ns/yang/Cisco-IOS-XE-bgp'|bgp], neighbor, 'update-source’, router,nati
```

- Only tail of path is unique, rest is shared with parent

Implementation for Path Sharing

- Custom data structure, mix of trees and hash tables

```
[d,c,b,a]
[c,b,a]
[b,a]
[a]
[f,e,a]
[e,a]
[b,e,a]
```



New FXS Implementation

Version	ETS table size	ETS table size (compressed)	Ratio
ConfD 6.7	93.82 MiB	56.03 MiB	1
Diff-dep bugfix	82.90 MiB	50.31 MiB	1.1
#cs_trim{}	36.70 MiB	23.54 MiB	2.4
#cs_file{}	15.10 MiB	15.79 MiB	3.5
De-duplicate	19.68 MiB	17.73 MiB	3.2
NIF	9.21 MiB		
NIF trim	8.03 MiB	5.81 MiB	9.6

Approximately twice as fast as the original implementation, and 1/10:th the size. Present in ConfD 7.1.

Cisco-style CLI Parsing

- Large configurations takes a long time to process
- Reading from CLI files (C-style)
- Loading configurations in ConfD
- Native implementation
  - Reduced memory usage compared to Java implementation

CLI: Top-down, breadth-first (on drop-node-name)

Goal: User friendly CLI, provide help and alternatives, precise error reporting

```
interface Loopback0
  ip address 127.0.0.1 255.255.255.255
exit
```

Algorithm:

Step1: /: **collect** interface, policy, mpls, ... **select** interface

Step2: /interface: **collect** Ethernet, ..., Loopback, … **select** Loopback

Step3: /interface/Loopback/: **collect** ip, ntp, peer, vrf, … **select** ip

Step4: /interface/Loopback/ip: **collect** access-group, address, arp, rip,…
select address

YANG tree:

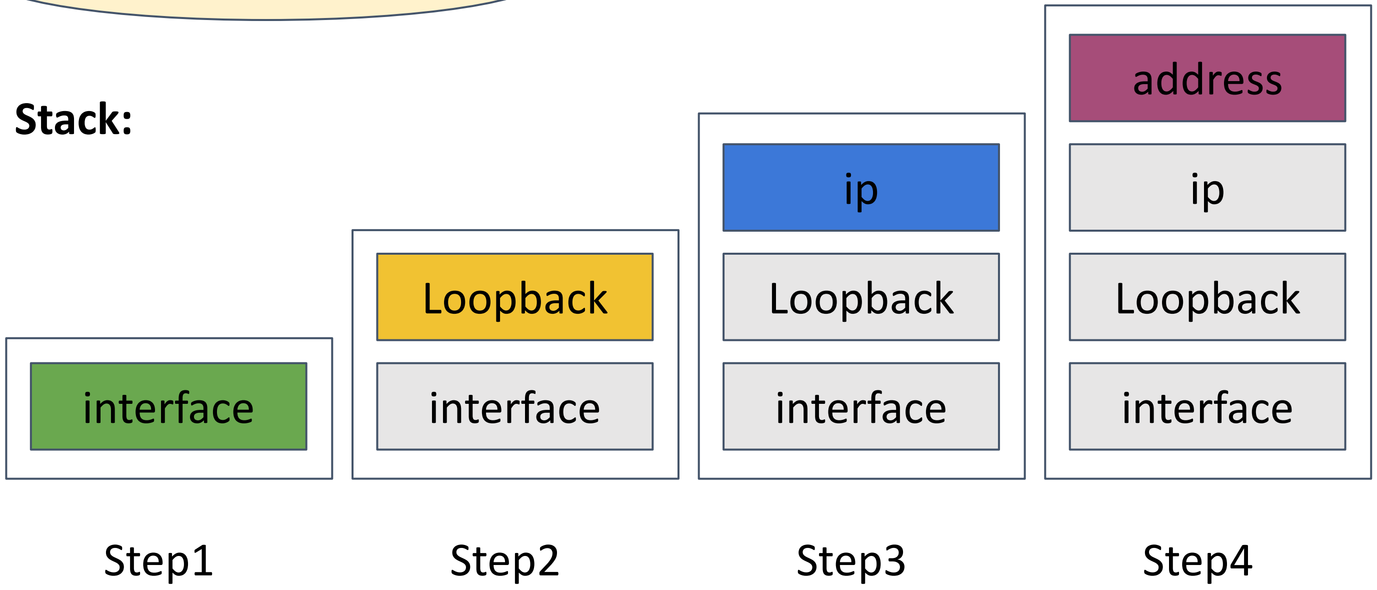
```
interface
   Ethernet
   FastEthernet
   GigabitEthernet
   ...
   Loopback
      ...
      ip
         address
         ....
      ....
   …
policy
mpls
…
```

Turbo CLI: Top-down, depth-first, stack based parser

Goal: Fast parsing for programmatic users, pick first best match

```
interface Loopback0
  ip address 127.0.0.1 255.255.255.255
exit
```

Stack:



YANG tree:

```
interface
   Ethernet
   FastEthernet
   GigabitEthernet
   ...
   Loopback
      ...
      ip
         address
         ....
      ....
   ...
policy
mpls
...
```

Turbo Parser vs Regular Parser

File	Regular CLI	Turbo Parser	Ratio
1000-lb-config	30 s	5.8 s	5.2
10,000-ace-config-cli-sequence	320 s	26 s	12.3
30,000-ace-config-cli-sequence	940 s	76 s	12.3
huge/nonwireless_CLI.txt	50 s	4.2 s	11.9
huge/ncs4216-all_config.txt	2.7 s	0.67 s	4.0

Filtering on List Instances in Data Provider

Would be nice to pass “filters” to the DP instead of filtering in ConfD – subtree filter, XPath, CLI.

- Avoid round trips between ConfD and data provider
- Simple XPath evaluation in DP

Important when

- Reading large sets of operational data and configuration
- Configuration validation
- Evaluating XPath expressions (e.g. must & when expressions)

Filter Example

```
list foo {
  key name;
  leaf name { type string; }
  leaf value { type int32; }
}

list bar {
  key name;
  leaf name { type string; }
  leaf fooref {
    type leafref {
      path "/foo/value";
    }
  }
```

XPath example:

```
    /foo[value > 42 or starts-with(name, “eth-”)]
```

NETCONF subtree example:

```
    <foo>
      <value>42</value>
    </foo>
```

CLI

```
      show foo eth-*
      show foo * value 42
```

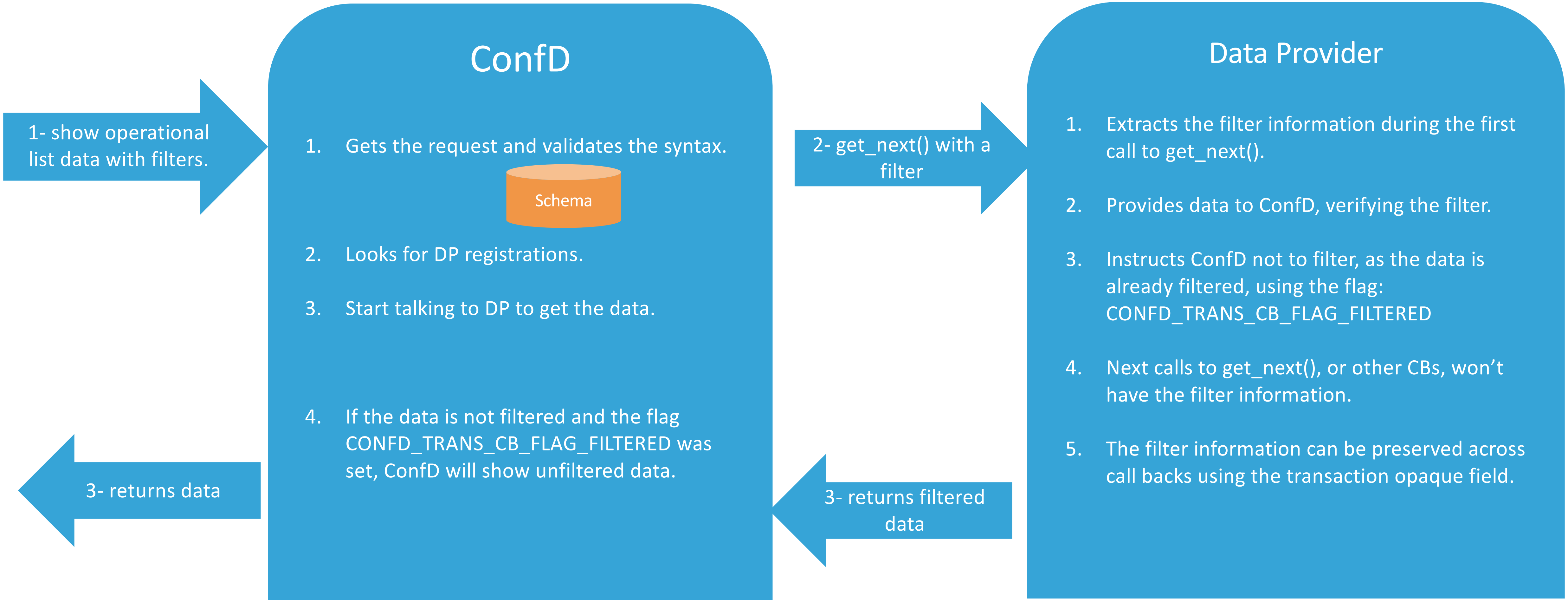
Validation

```
list foo {
  key name;
  leaf name { type string; }
  leaf value { type int32; }
}

list bar {
  key name;
  leaf name { type string; }
  leaf fooref {
    type leafref {
      path "/foo/value";
    }
  }
```

Delete of “/foo” with “value” 42 will get_next through the entire list bar to see if any list entry has a leaf “fooref” with value 42.

Flow



1- show operational list data with filters.

ConfD

1. Gets the request and validates the syntax.

2. Looks for DP registrations.
3. Start talking to DP to get the data.
4. If the data is not filtered and the flag CONFD_TRANS_CB_FLAG_FILTERED was set, ConfD will show unfiltered data.

3- returns data

2- get_next() with a filter

3- returns filtered data

Data Provider

1. Extracts the filter information during the first call to get_next().
2. Provides data to ConfD, verifying the filter.
3. Instructs ConfD not to filter, as the data is already filtered, using the flag: CONFD_TRANS_CB_FLAG_FILTERED
4. Next calls to get_next(), or other CBs, won’t have the filter information.
5. The filter information can be preserved across call backs using the transaction opaque field.

Filter syntax

```
<expr> = <expr> 'or' <expr>
       / <expr> 'and' <expr>
       / 'not' <expr>
       / 'cmp' <op> <node> <value>
       / 'exec' <func> <node> <value>
       / 'exists' <node>

<op> = 'eq' / 'gt' / 'lt' / 'gte' / 'lte' / 'neq'

<func> = 're-match' / 'starts-with'
       / 'derived-from' / 'derived-from-or-self'

<node> = "array of tags (points to a leaf or leaf-list)"
```

Supported Callbacks

Provided in next() callbacks:

```
get_next()
get_next_object()
find_next()
find_next_object()
```

Get filter using the function (at the first invocation of the callback)

```
confd_data_get_list_filter()
```

Free the filter when done using:

```
confd_free_list_filter()
```

Performance

Operation	Regular	With filters	Ratio
Validate leaf-ref list of 10,000	22 s	1.5 s	14.7