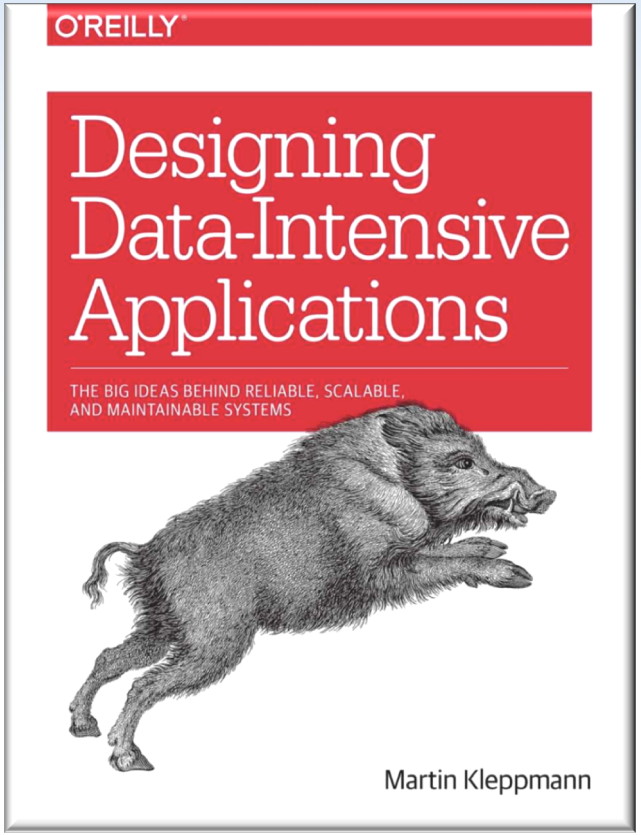
Improving ConfD Application Performance

Inspired by:

- Great book to help navigate the diverse and fast-changing landscape of technologies for processing and storing data.
- Examples of technologies that form the foundation of many popular applications and that have to meet …
  - Scalability
  - Performance
  - Reliability
- ... requirements in production every day.

- Kleppmann, Martin. Designing Data-Intensive Applications O'Reilly Media.



The Quick Elon Brain Dump

- How complex is your design?
- Are you sticking to an overcomplex design? Given unlimited time and resources + flexible customer requirements etc. what would you do?
- What can be improved with the constraints of the interfaces you are integrating with?
- Answer is NETCONF/YANG but what's the question? ;-)
- Needless to say but ConfD's API is of course perfect ;-)

https://youtu.be/clQ36Kt7UVg

Credits: Tim Dodd – Everyday Astronaut

What is the Main Purpose of ConfD?

A Configuration Daemon to Enable Participation in a Transactional Distributed System

- Why do I need configuration?
- Why use transactions?
- Can’t I just continue to optimize my human to machine interfaces, CLI, WebUI?
- Why do we need to automate network services?
- What is a programmable network?

- What are the best practices for using ConfD?
- Am I using ConfD’s APIs they way they are intended to be used?
- If I use ConfD’s APIs like I do, am I part of the 99% or the 1% of use cases?
- My design for integrating with ConfD is complex / “clever”, is there another way to do the integration ?
- Should I use ConfD to implement this functionality?
- What is the root cause(s) of the issue I observe?
- How do I find the root cause(s)?
- Instead of optimizing read speed, can I avoid having to read/write? (oob changes, backup, etc)
- How do I best describe the load my system need to be able to handle?
- How do I measure the performance of my system?
- How many 9:s (availability) am I optimizing for?
- Can I say ”no” to some of my customers requirements?

Why Automate?

Challenges

- Wide variety of legacy equipment
- Human-to-machine tinkering
  - CLI scripting
  - Error-prone
  - Non-standard
- Device centric

Opportunities

- Programmability
- Device abstraction through a common network API
- Standards based
- Machine-to-machine
- Scale

Why Use Transactions?

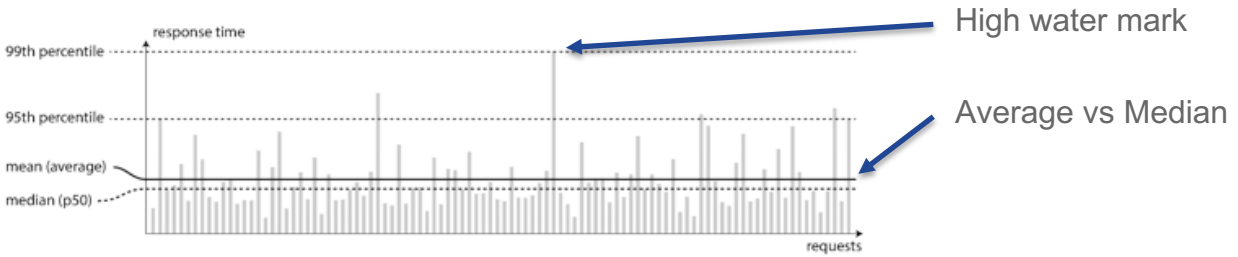
Some authors have claimed that general two-phase commit is too expensive to support, because of the performance or availability problems that it brings. We believe it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise rather than always coding around the lack of transactions.
James Corbett et al., *Spanner: Google's Globally-Distributed Database* (2012)

Systems Operating at Large Scale

- The architecture of systems that operate at large scale is usually highly specific to the application
- There is no such thing as a generic, one-size-fits-all scalable architecture
- The problem may be:
  - the volume of reads, writes
  - the volume of data to store
  - the complexity of the data
  - the response time requirements
  - the access patterns
  - ….or (usually) some mixture of all of these plus many more issues.
- Scalable architectures are nevertheless usually built from general-purpose building blocks, arranged in familiar patterns.

Describing Load and Performance

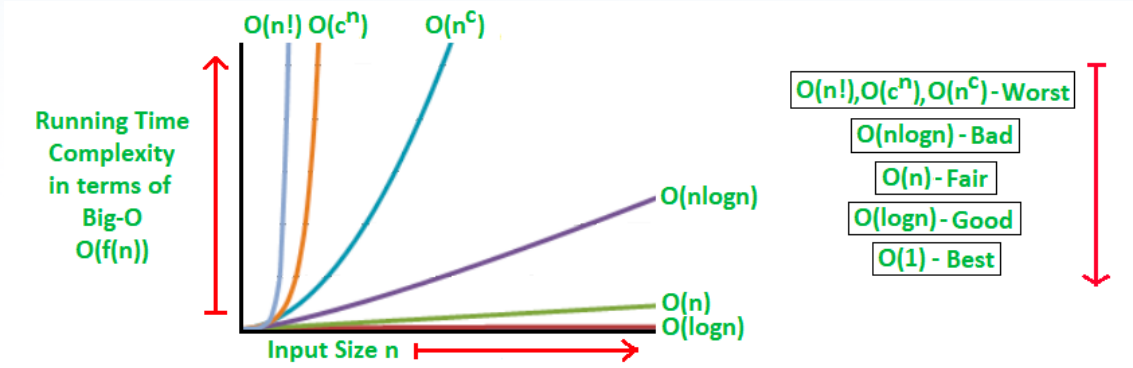
- An architecture that scales well for a particular application is built around assumptions of which operations will be common and which will be rare.
  - The load parameters.
- If those assumptions turn out to be wrong, the engineering effort for scaling is at best wasted, and at worst counterproductive.
- Iterate quickly on product features vs scale to some hypothetical future load?
- What is your response time for the load you are designing / developing / deploying for?



Describing Performance - Big-O Notation

Order of magnitude functions:

- Constant - *O(1)*
- Logarithmic - *O(log n)*
- Linear - *O(n)*
- Log Linear - *O(n log n)*
- Quadratic - *O(n2)*
- Exponential - *O(2n)*
- Factorial - *O(n!)*



ConfD Performance Claims

NETCONF
RESTCONF
CLI
JSON-RPC
SNMP
MAAPI

Management Operation	Performance claim	Comment
Start ConfD / Load databasefrom disk	O(db_size)	ConfD start time varies linearly with the size of the database
User Log in / Start Session	O(all_access_rules)	Whenever a user logs in, all access rules are checked for relevance. Typically dominated by the constant time crypto set up
Read one static object	O(1) +O(access)	Reading an object outside a table is a constant time operation. Unless there are many access rules relevant for this session
Read one dynamic object using filter	O(log(table)) +O(access)	Reading an object in a table is a logarithmic operation in the size of that table
Transaction to modify one dynamic object	O(log(table)) +O(access)	Updating or adding an object in a table is also a logarithmic operation in the size of the table
Read many dynamic objects	O(transaction)* (O(log(table)) +O(access))	Reading a set of objects scales linearly relative to one object
Transaction to create many dynamic objects	O(transaction) *(O(log(table)) +O(access))	Writing a set of objects scales linearly relative to one object
XPath queries	Depends	Depends on query, ranges from O(1) and upwards

ConfD 7.2 CDB and Data Provider APIs
Typical execution times

CDB API

cdb_init(0.007ms) cdb_connect(0.4ms) cdb_set_namespace(0.1ms) cdb_close(0.01ms)	O(1)	The time to set up a CDB connection does not depend on number of already connected clients
cdb_start_session(0.1ms) cdb_end_session(0.07ms)	O(1)	Session start and end times do not depend on the number of already ongoing sessions
cdb_num_instances(0.2ms)	O(table)	Counting the number of instances in a table is very fast, and varies linearly with the table size
cdb_set_elem(0.8ms)	O(1)or O(log(table))	Writing operational data that isn't part of a table is constant time, in a table it depends logarithmically on the table size
cdb_get_int32(0.09ms) cdb_get_ipv4(0.1ms) cdb_get_bool(0.1ms)	O(1)or O(log(table))	Reading configuration or operational data that isn't part of a table is constant time, in a table it depends logarithmically on the table size
cdb_subscribe(0.2ms) cdb_read_subscription_socket (0.04ms) cdb_sync_subscription_socket (0.1ms)	O(1)	Setting up and managing subscriptions are constant time operations, times do not depend on number of subscriptions

DP API

confd_init(0.08ms) confd_init_daemon(0.03ms) confd_connect(1ms) confd_register_trans_cb (0.009ms)	O(1)	To set up a ConfD connection is a constant time operation. It does not depend on the number of already connected clients
confd_trans_set_fd(0.001ms)	O(1)	The Data Provider calls this function whenever a new session is established to dispatch the session to the thread of its choice
confd_data_reply_value(0.01ms)	O(1)	The Data Provider uses this function to respond with the current value for every value requested by ConfD

Performance Tools – Trivial Version

Measure Performance:

- Wall clock time
- Memory resident set size (RSS)
- Memory high water mark (HWM)

```
#/bin/bash
echo "N,time(s),RSS(kB),HWM(kB)"
for i in 1 10 100 1000 10000 100000
do
    for j in 1 2 4 6 8
    do
        NUM=$(($j*$i))
        # my_init_confd_etc.sh
        START=$(date +%s)
        sleep $NUM # my_do_something_script.sh $NUM
        END=$(date +%s)
        TIME=$(($END-$START))
        pid=$(pidof confd)
        MEM=$(cat /proc/$pid/status | grep -A 1 VmHWM)
        arr=($MEM)
        HWM=$(echo ${arr[1]})
        RSS=$(echo ${arr[4]})
        echo "$NUM,$TIME,$RSS,$HWM" # > save.csv
    done
done
```

Monitor Performance:

- Percent of CPU time
- Memory RSS and HWM

```
#/bin/bash
echo "CPU(%),RSS(kB),HWM(kB)"
while(true)
do
        pid=$(pidof confd)
        CPU=$(ps --no-headers -o '%cpu' -p "$pid")
        MEM=$(cat /proc/$pid/status | grep -A 1 VmHWM)
        arr=($MEM)
        HWM=$(echo ${arr[1]})
        RSS=$(echo ${arr[4]})
        echo "$CPU,$RSS,$HWM" # > save.csv
        sleep 5
    done
done
```

Visualize

Using for Example Prometheus and Grafana

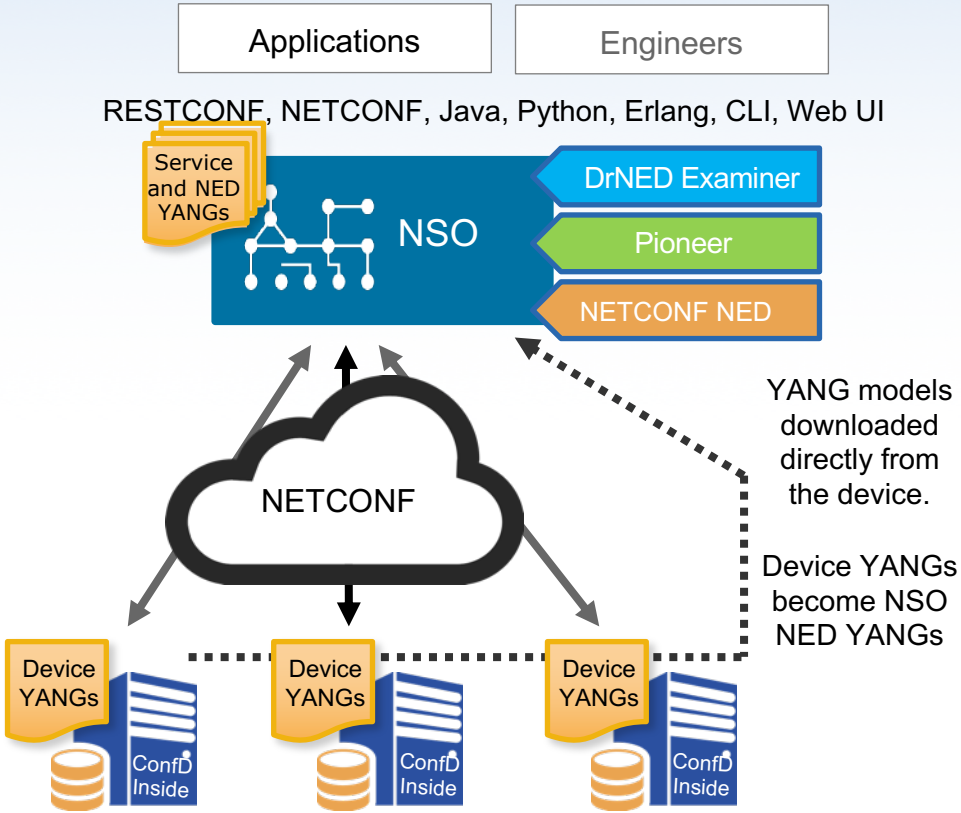
ConfD Progress Trace

- Very useful information for debugging, diagnostics and profiling.
- When a transaction or action is applied, ConfD emits timestamped progress events.
- Print in the CLI, log to file, log as operational data in CDB, subscribe to notifications, maapi.reportProgress()
- Control verbosity – normal, verbose, very verbose, debug
- Filter on context and/or user
- See ConfD's User Guide chapter "Progress Trace" for more details.

```
 2018-01-02T16:00:38.228 applying transaction...
entering validate phase for running (tid=105, session-id=43)...
 2018-01-02T16:00:38.229 run pre-trans-lock service callbacks...
 2018-01-02T16:00:38.229 run transforms and transaction hooks...
 2018-01-02T16:00:38.231 run transforms and transaction hooks done [0.000 sec]
 2018-01-02T16:00:38.231 pre-trans-lock service callbacks done [0.000 sec]
 2018-01-02T16:00:38.231 grabbing transaction lock... ok [0.000 sec]
 2018-01-02T16:00:38.232 creating rollback file... ok [0.000 sec]
 2018-01-02T16:00:38.233 run transforms and transaction hooks...
 2018-01-02T16:00:38.246 run transforms and transaction hooks done [0.000 sec]
 2018-01-02T16:00:38.246 mark inactive... ok [0.000 sec]
 2018-01-02T16:00:38.246 pre validate... ok [0.000 sec]
 2018-01-02T16:00:38.247 run validation over the change set...
 2018-01-02T16:00:38.248 validation over the change set done [0.001 sec]
 2018-01-02T16:00:38.248 run dependency-triggered validation...
 2018-01-02T16:00:38.248 dependency-triggered validation done [0.000 sec]
 2018-01-02T16:00:38.249 check configuration policies...
 2018-01-02T16:00:38.249 configuration policies done [0.000 sec]
leaving validate phase for running (tid=105, session-id=43)
entering write-start phase for running (tid=105, session-id=43)...
 2018-01-02T16:00:38.249 cdb: write-start
leaving write-start phase for running (tid=105, session-id=43)
entering prepare phase for running (tid=105, session-id=43)...
 2018-01-02T16:00:38.250 cdb: prepare
leaving prepare phase for running (tid=105, session-id=43)
entering commit phase for running (tid=105, session-id=43)...
 2018-01-02T16:00:38.262 cdb: commit
 2018-01-02T16:00:38.262 releasing transaction lock
leaving commit phase for running (tid=105, session-id=43)
 2018-01-02T16:00:38.263 applying transaction done [0.035 sec]
```

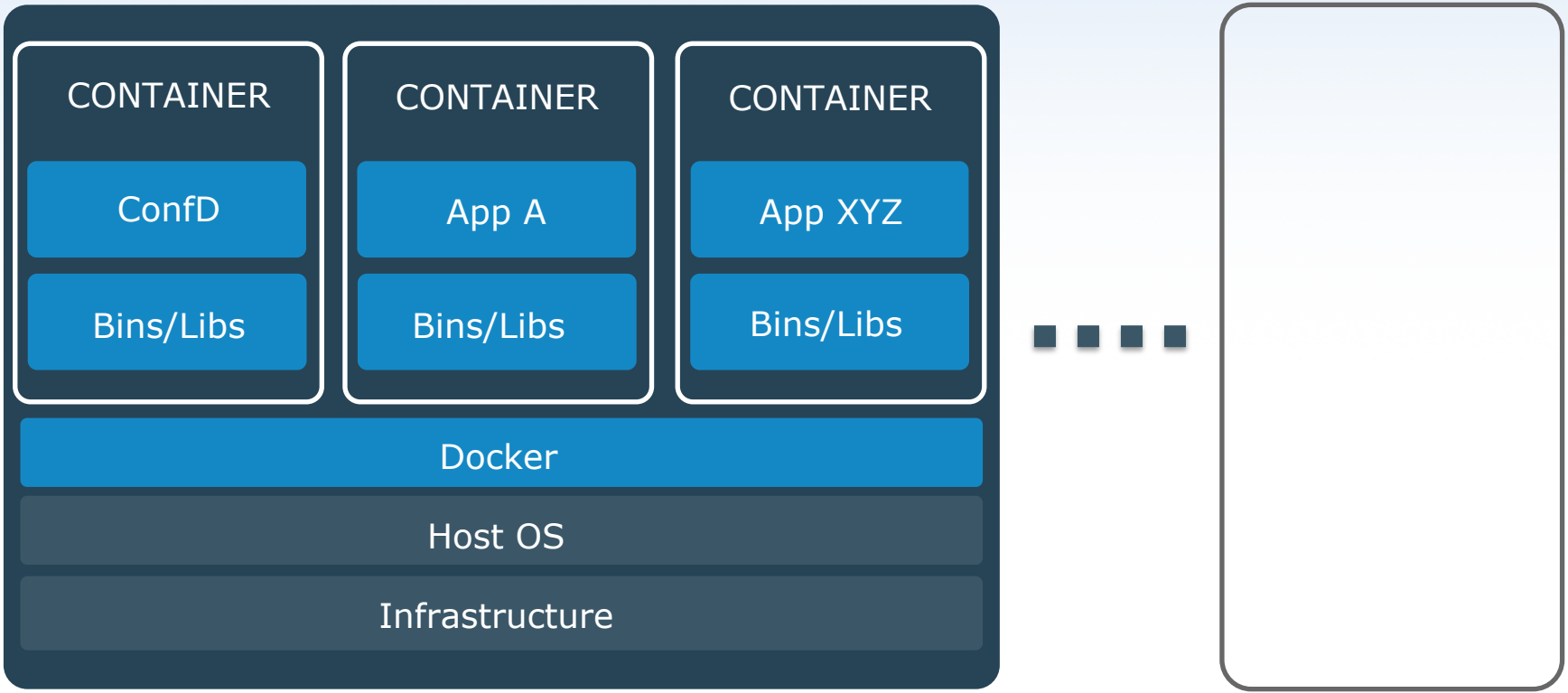
Test your ConfD Network Element with NSO

1. Setup ConfD instance(s)
2. Update the NSO configuration to plug in the new NETCONF device
3. Build a NETCONF NED for the device using
   a) NSO >= 5.2 - NETCONF NED Builder NSO < 5.2 - Pioneer NSO package
4. Run tests
   a) Verify that our expectations are met using the DrNED Examiner NSO package
   b) Add you own tests



Consider using Containers

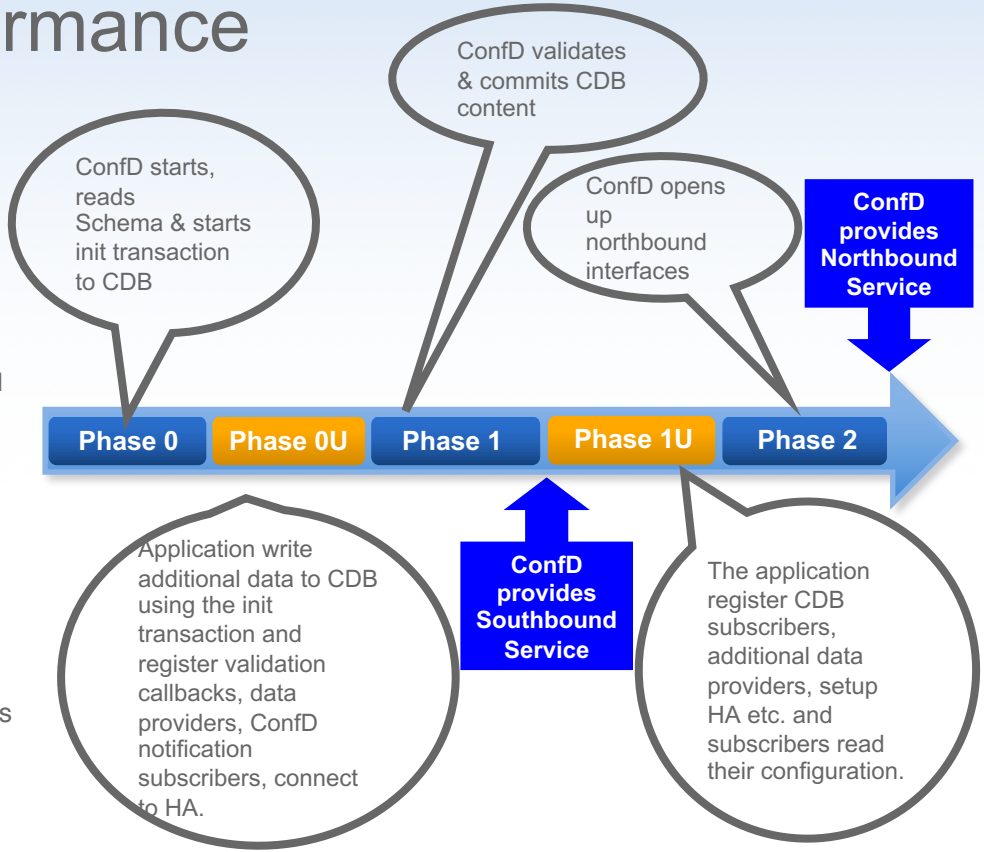
Even for Performance Tests



Startup / Upgrade Performance

Consider

- Pre-populating CDB - An XML init file with data is around a magnitude slower to initialize from
- cdb_trigger_subscriptions() – you can be selective with which subscribers and their scope
- confd --ignore-initial-validation to skip validation points in Phase 1
- A specialized upgrade client app can be a good investment instead of saving then loading XML
- Consider ConfD's HA or your own specialized replication application to switch to a hot standby
- Avoid using the startup datastore
- Simple demo that measure time spent in each start phase when loading data from CDB files, XML file, and startup datastore exists
- ConfD 7.1 optimized FXS files

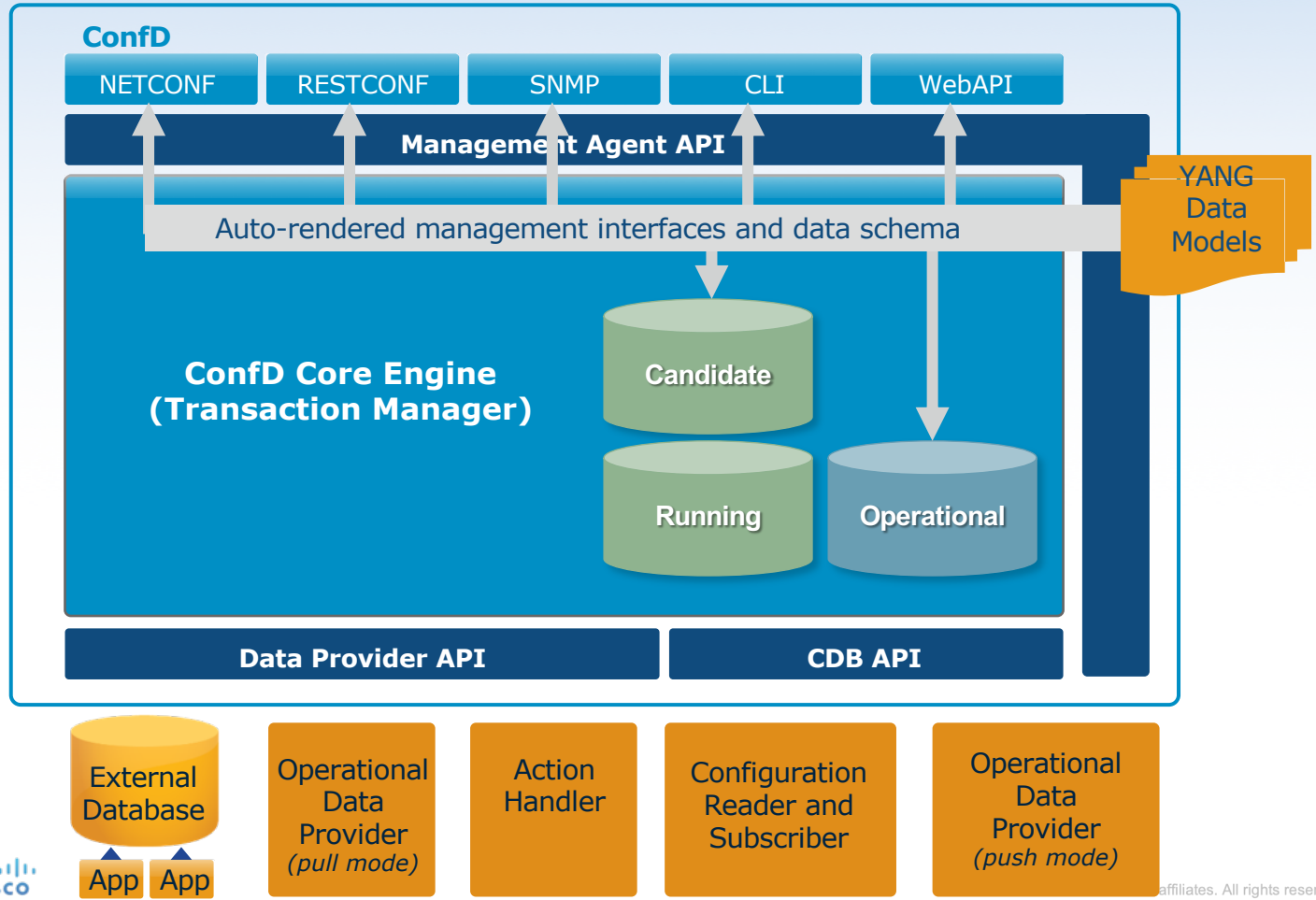


CDB Subscriber API - cdb.get_modifications()

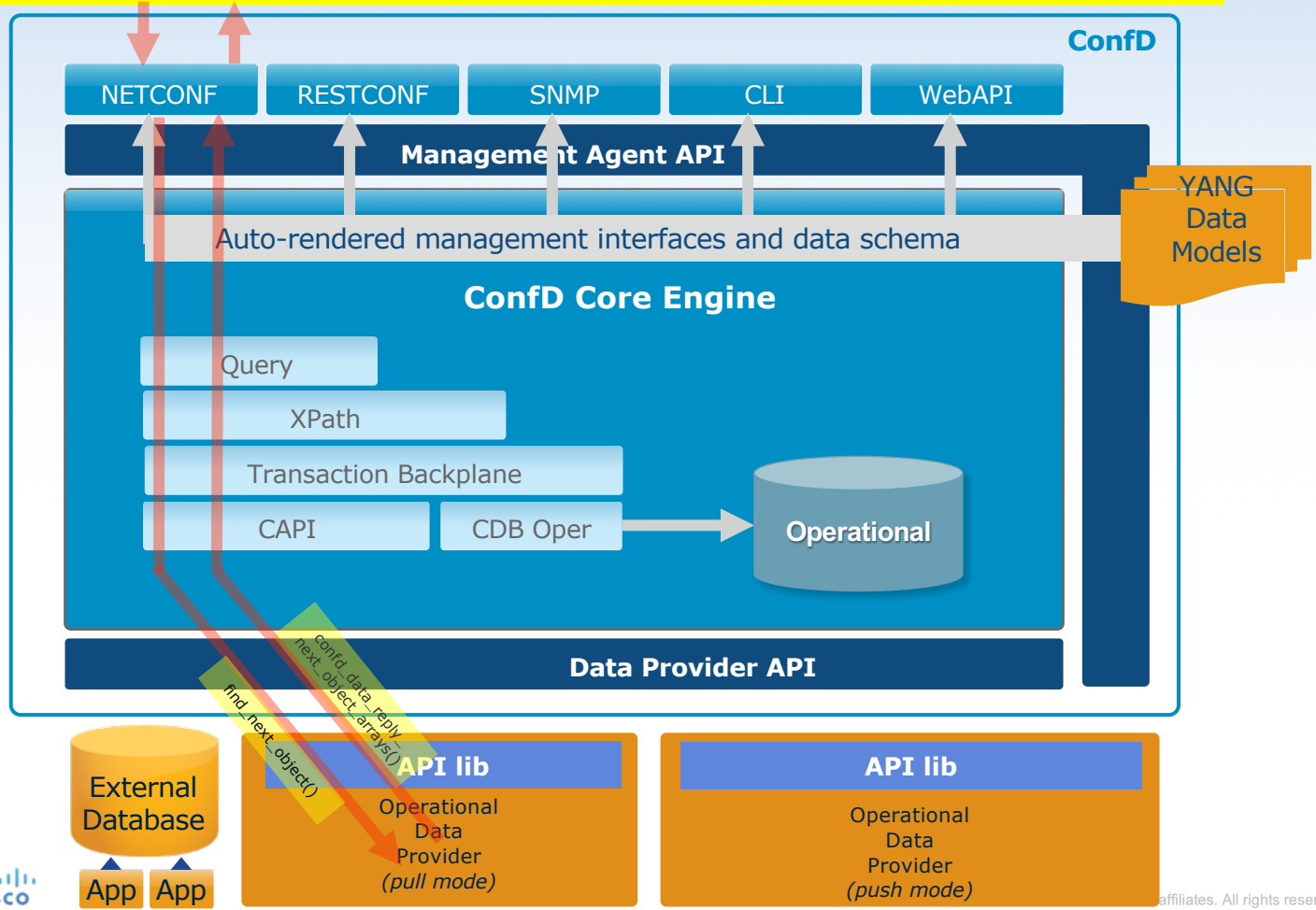
Handling config changes become simple and efficient

```
try:
    flags = cdb.GET_MODS_INCLUDE_LISTS
    (type, sub_flags, _) = cdb.read_subscription_socket2(subsock)
    if type == cdb.SUB_COMMIT:
        mods = cdb.get_modifications(subsock, sid, flags, subpath)
        if mods == []:
            print('No modifications')
        else:
            print('\n***Config updated --> Commit')
            print_modifications(mods)
            handle_modifications(mods, sock)
```

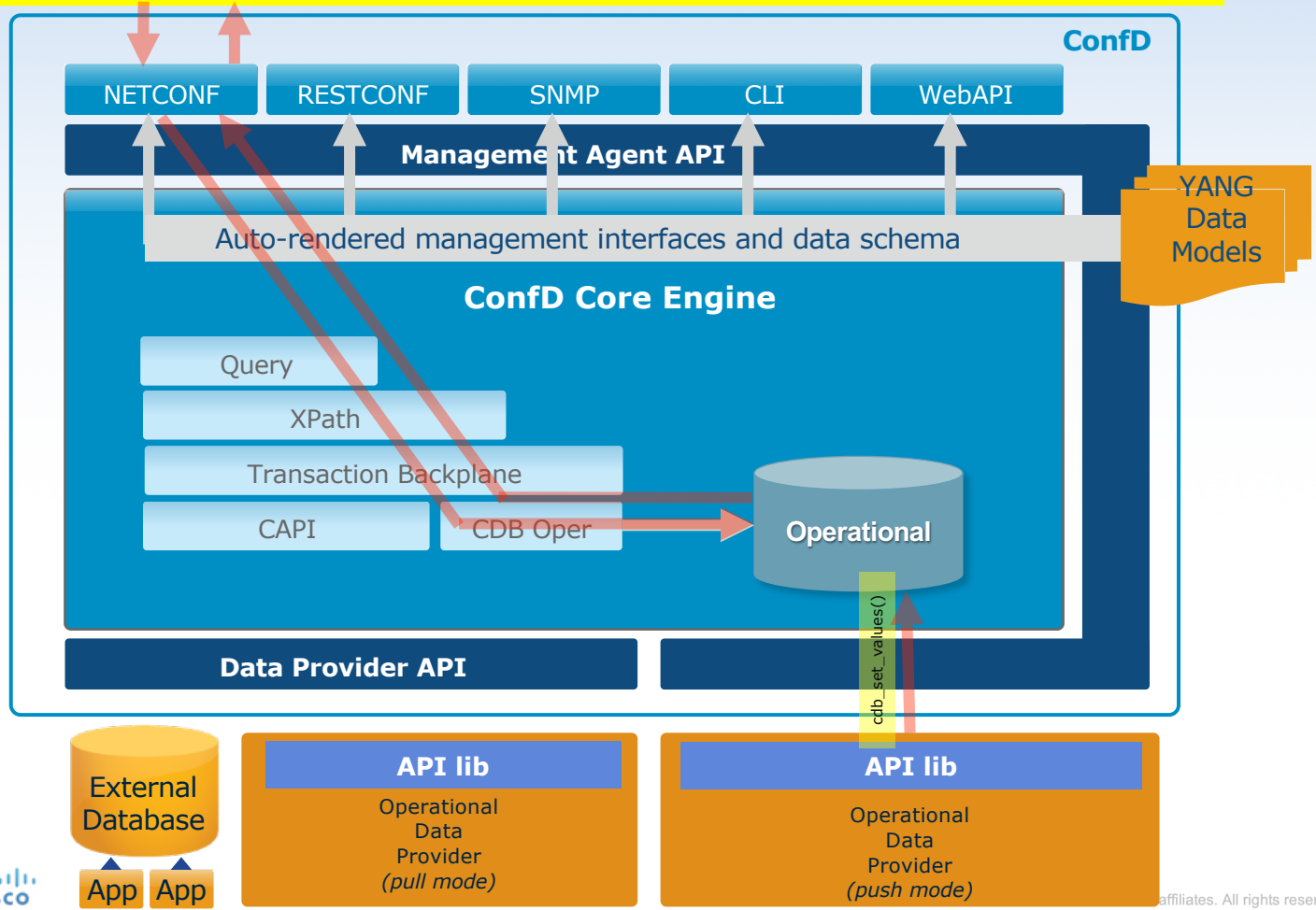
```
def handle_modifications(mods,sock):
    cdb.start_session(sock, cdb.OPERATIONAL)
    cdb.set_namespace(sock, fbox_ns.ns.hash)
    ni = cdb.num_instances(sock, "/facebox/container-ip")
    iplist = []
    ipstr = str(cdb.get(sock, "/facebox/container-ip"))
    iplist = ipstr.split()
    for m in mods:
        ct = m.v.confd_type()
        if ct  == _confd.C_XMLBEGIN:
            mod = Modifications.CREATE
        elif ct == _confd.C_XMLBEGINDEL:
            mod = Modifications.DELETE
        elif ct == _confd.C_XMLEND:
            if mod ==  Modifications.CREATE:
                data = '{"url":"' + url + '","name":"' + name + '","id":"' + face_id + '"}'
                for ip in iplist:
                    print("Add to Facebox container IP list: " + ip)
                    response = requests.post('http://' + ip + ':8080/facebox/teach',
                                             headers=headers, data=data)
            else:
                for ip in iplist:
                    print("Delete from Facebox container IP list: " + ip)
                    response = requests.delete('http://' + ip + ':8080/facebox/teach/%7B' +
                                               face_id + '%7D')
        elif ct not in [ _confd.C_XMLTAG, _confd.C_NOEXISTS ]:
            tag = _confd.hash2str(m.tag)
            if tag == "name":
                name = str(m.v)
            else:
                url = str(m.v)
                face_id = url.rsplit('/', 1)[-1]
    cdb.end_session(sock)
```



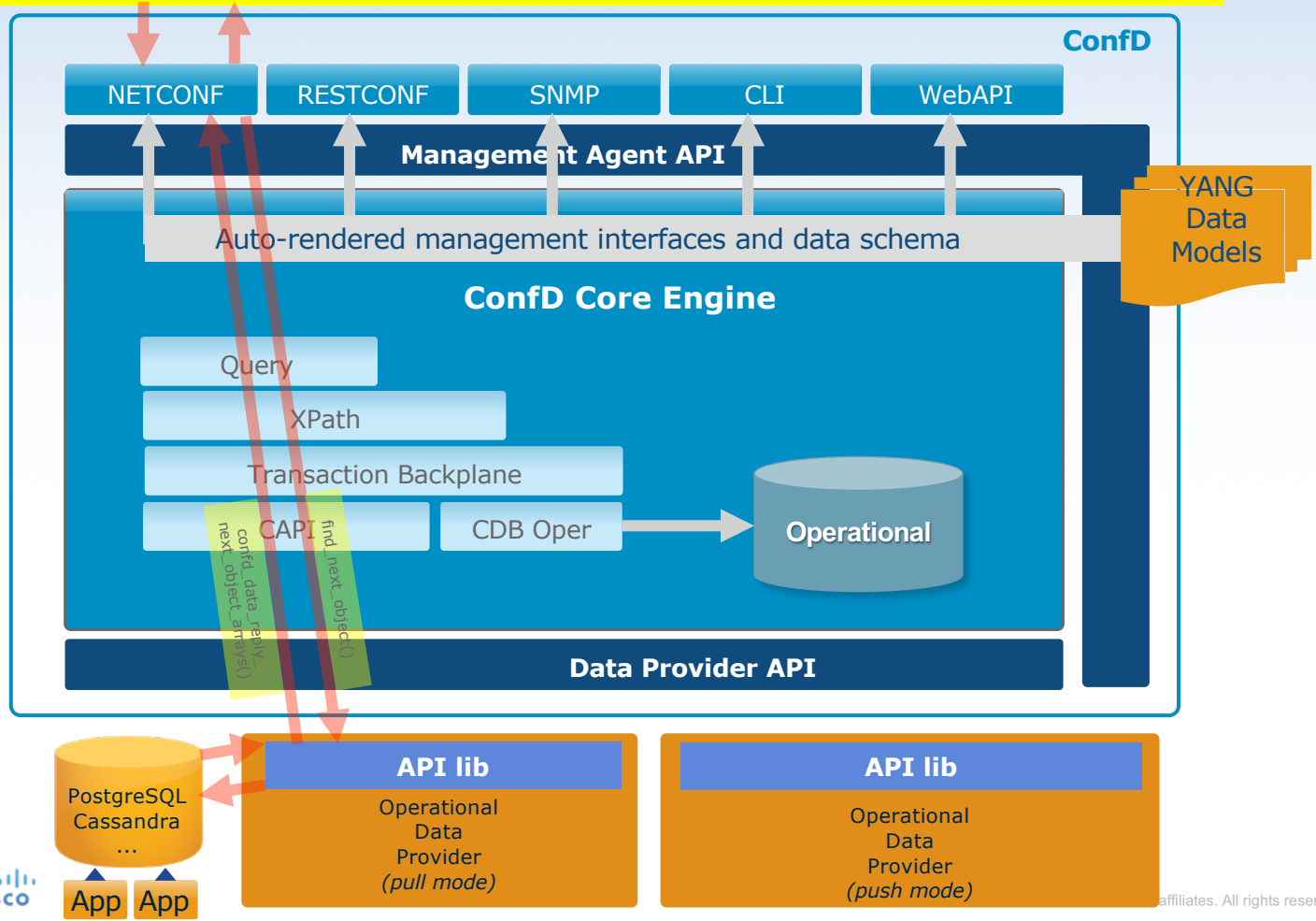
<get> operation with <filter type= "subtree"> OR <filter type="xpath"> OR a tailf <immediate-query> operation



<get> operation with <filter type= "subtree"> OR <filter type="xpath"> OR a tailf <immediate-query> operation



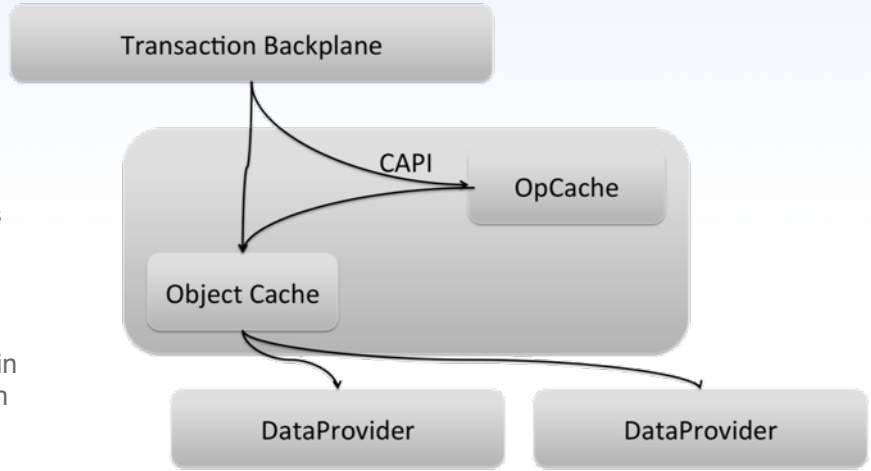
<get> operation with <filter type= "subtree"> OR <filter type="xpath"> OR a tailf <immediate-query> operation



Data Provider

Consider

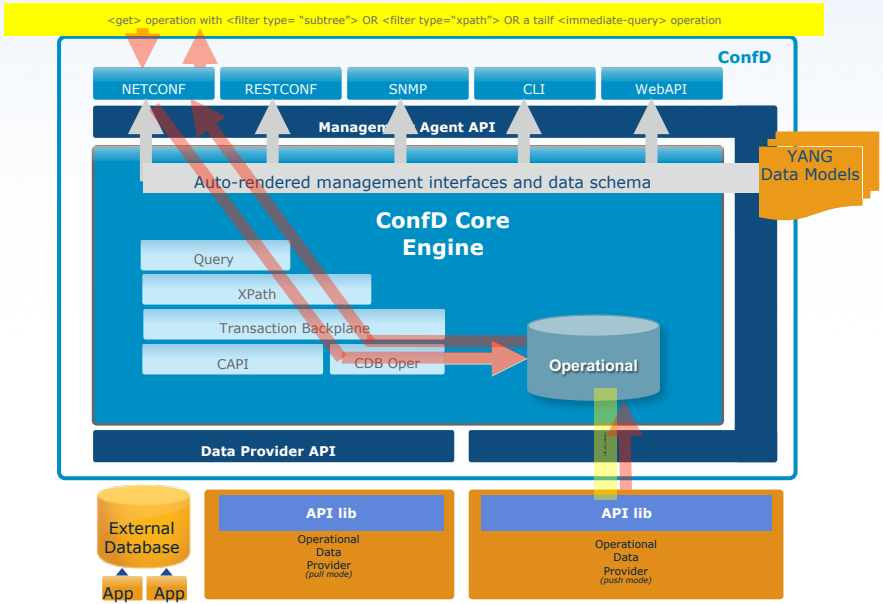
- Start simple with the operational data store (CDB API)
- Utilize the CDB operational data store and OpCache where possible
- Make sure the Object Cache timeout is large enough
- Consider find_next_object(), get_object(), num_instances() -- and get_next_object() too if you are using non-unique secondary-indexes
- num_instances() is used for example when a list has a max/min-elements statement
- Passing state data for lists in the t_opaque field has advantages but in some cases disadvantages. Can often be implemented stateless with little overhead to not have to use the traversal_id to check for concurrent list traversals.
- Consider if doing the filtering of data in your application is more efficient than having ConfD do it.
- Multiple best practices demo variants exists that measure performance



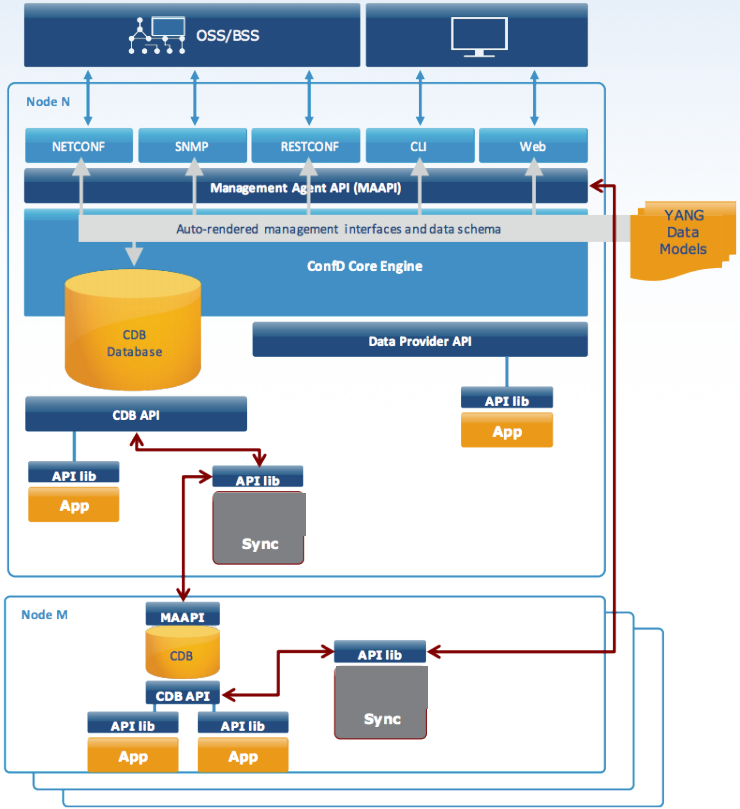
CDB's Operational Datastore

Consider

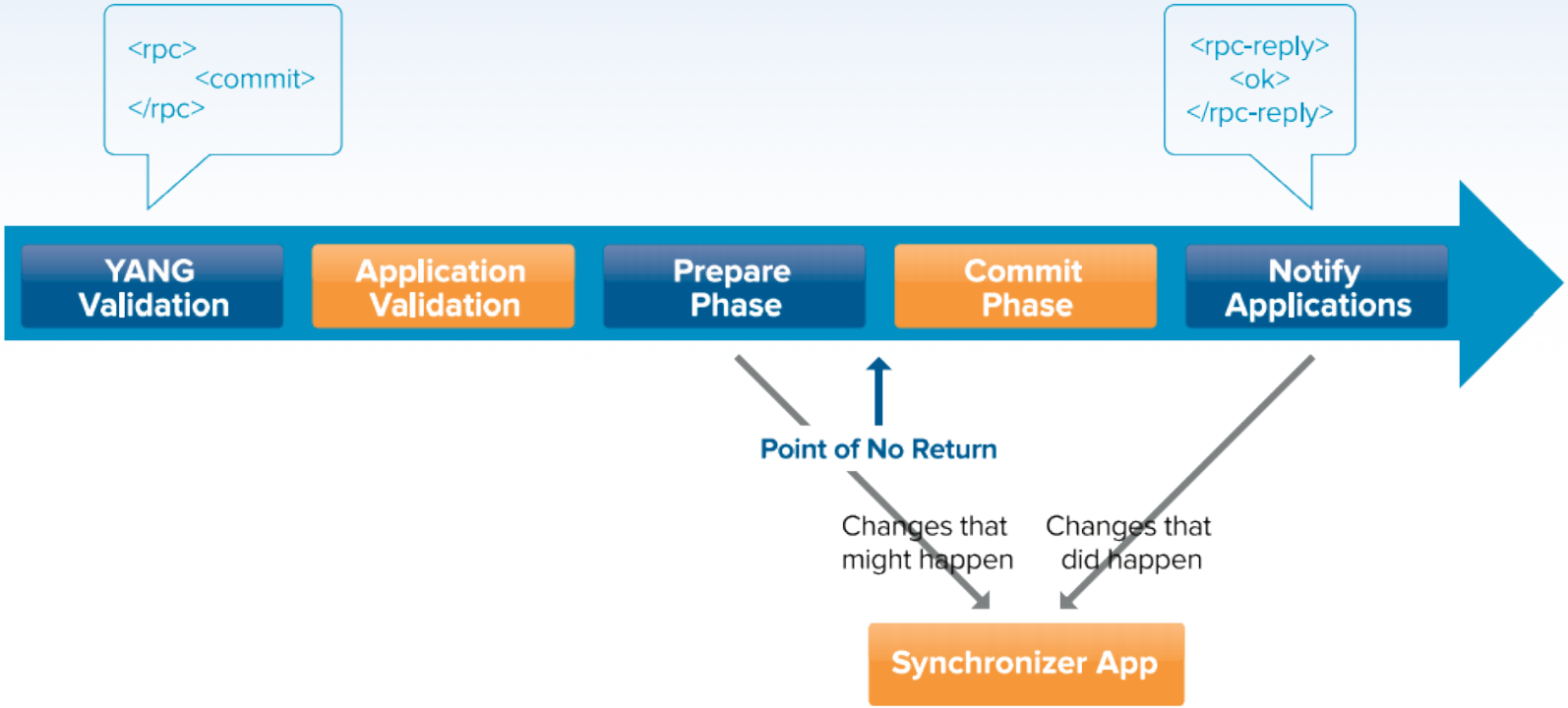
- maapi_get/set_values(), cdb_get/set_values()
- MAAPI goes through the transaction engine to CDB while the CDB API go directly to CDB.
  - CDB API has a built-in list "index" / "instance integer" feature
  - Changes to data appear only when the "transaction" is applied.
- Use an external DB if you have a massive amount of state/telemetry data and just need expose a small subset over NETCONF, RESTCONF, CLI, etc. for a manager such as NSO while:
  - Very frequently passing large amounts of application state data around between applications.
  - HA replication need to be throttled, use ring mechanisms, eventual consistency, etc.
  - You have a need for multiple indexes and joins on the fly, etc, to present your data in real time to for example directly to a web interface. Not over NETCONF.
  - Legacy. Your project manager is somehow able to convince you that changes and new things are bad and very costly.
- Again, multiple best practice demos measuring performance exists.



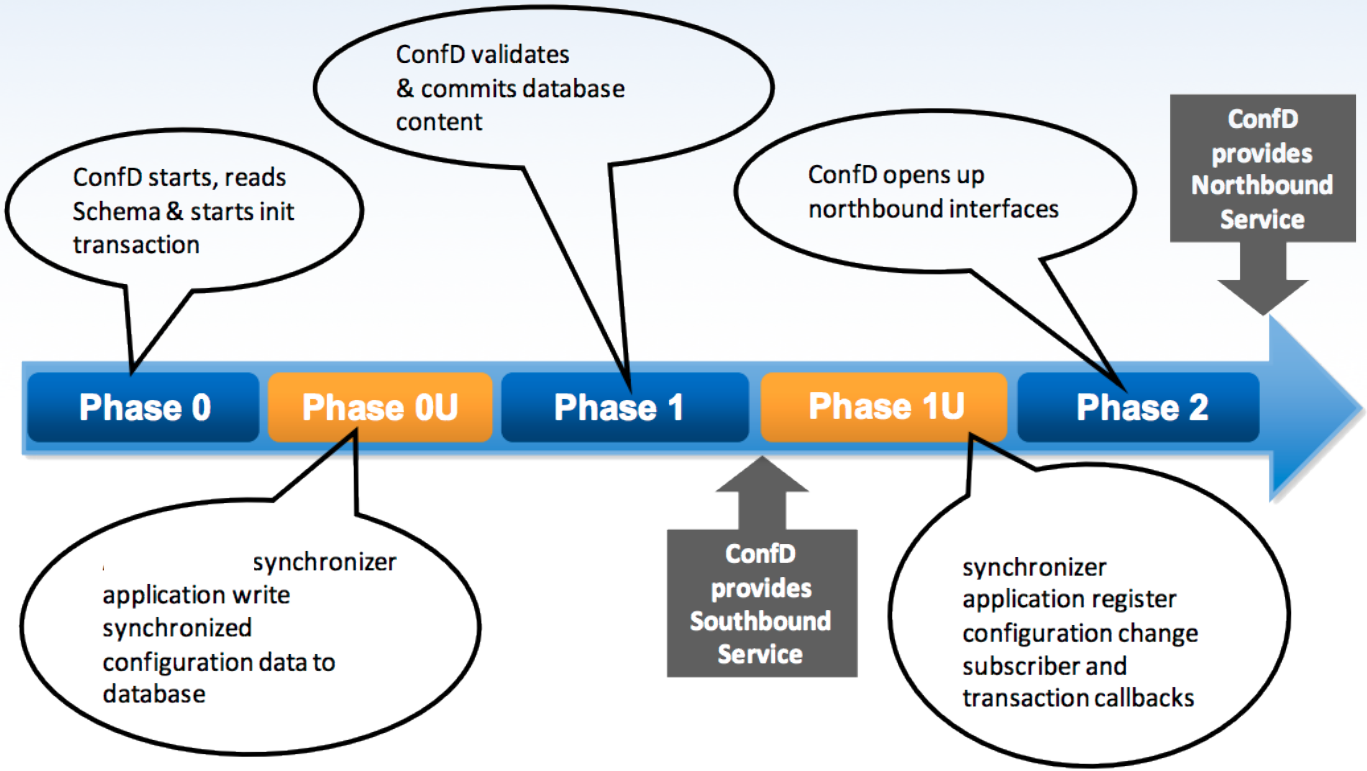
Synchronizing Using an DIY Application Overview



Synchronizer High Level Overview

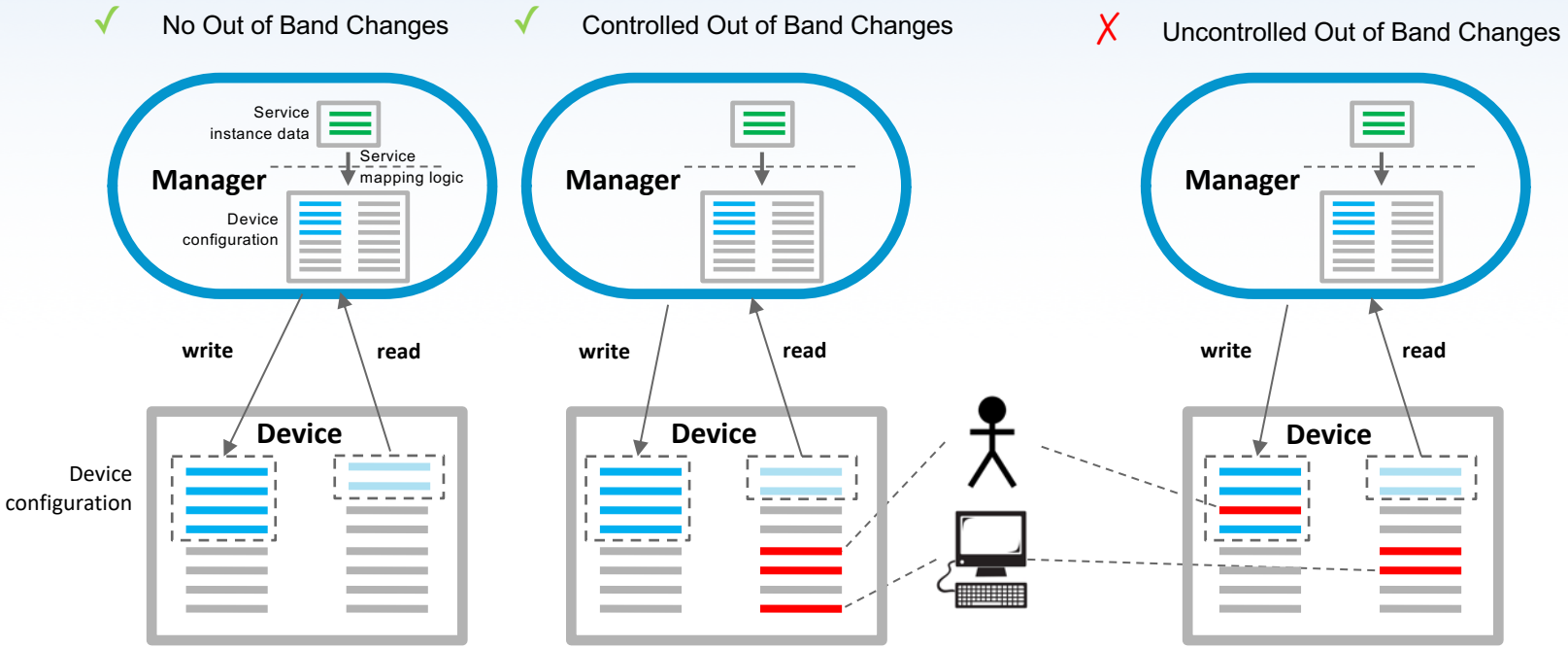


Synchronizer High Level Overview



M2M Kryptonite – Uncontrolled Out of Band Changes

Give priority to machine-to-machine communication



For More Information

- ConfD - https://www.tail-f.com
  - The ConfD Developer space on GitHub: https://github.com/ConfD-Developer
- NSO - https://www.cisco.com/c/en/us/solutions/service-provider/solutions-cloud-providers/network-services-orchestrator-solutions.html
  - NSO Developer Space on GitHub (e.g. DrNED Examiner): https://github.com/NSO-developer

ConfD - NSO Interoperability Testing program https://info.tail-f.com/nso_interop_lab

TOMORROW starts here.