Crosswork Network Services Orchestrator (NSO)

Outstanding Performance. Faster Automation.

Cisco Knowledge Network, October 20, 2022

Speakers: Nils-Petter Tisell, Product Manager
Viktor Leijon, Distinguished Engineer

Facilitator: Mina Paik, Product Marketing Leader

Agenda

- Crosswork NSO Facts
- The Value: What’s in it for me?
- Release 6.0
  - Performance Increase
  - Developer Experience
- Technical Deep Dive
- Takeaways

NSO Facts

1000

Devices, platforms and OSes supported

+

30+ Cisco products/offers using NSO or ConfD

All top 10 service providers use NSO in production

250

Customers

400

NED releases per month

Active developer community
6000 unique visitors per month
2000+ posts

What’s in it for me?

Infra Operations

Automate everything in the network. Multiple architectures, one tool.

Service Automation

One single source of truth to manage customer experience including all domains

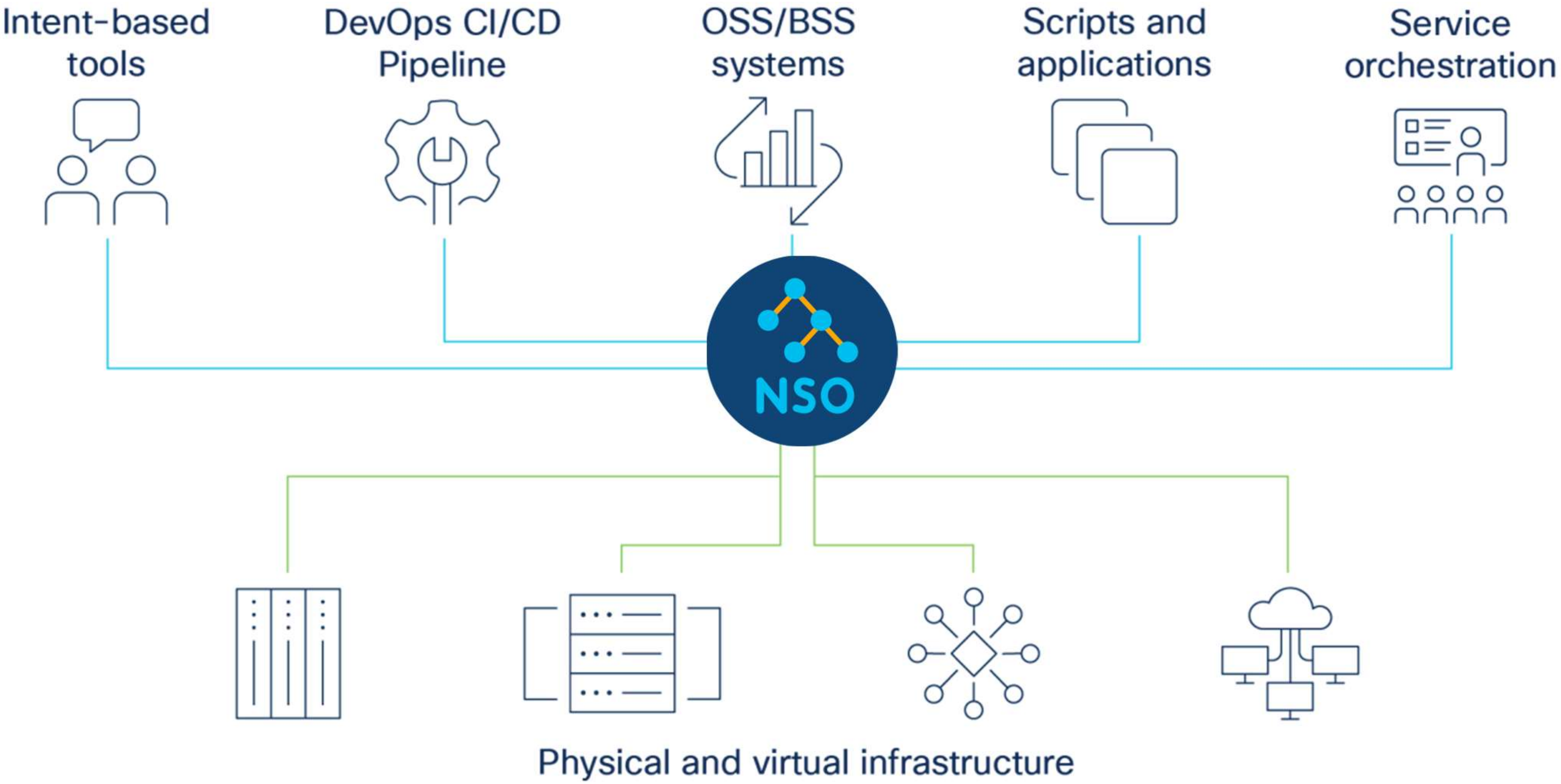
Service Developer

The tool to develop and deploy new services quickly

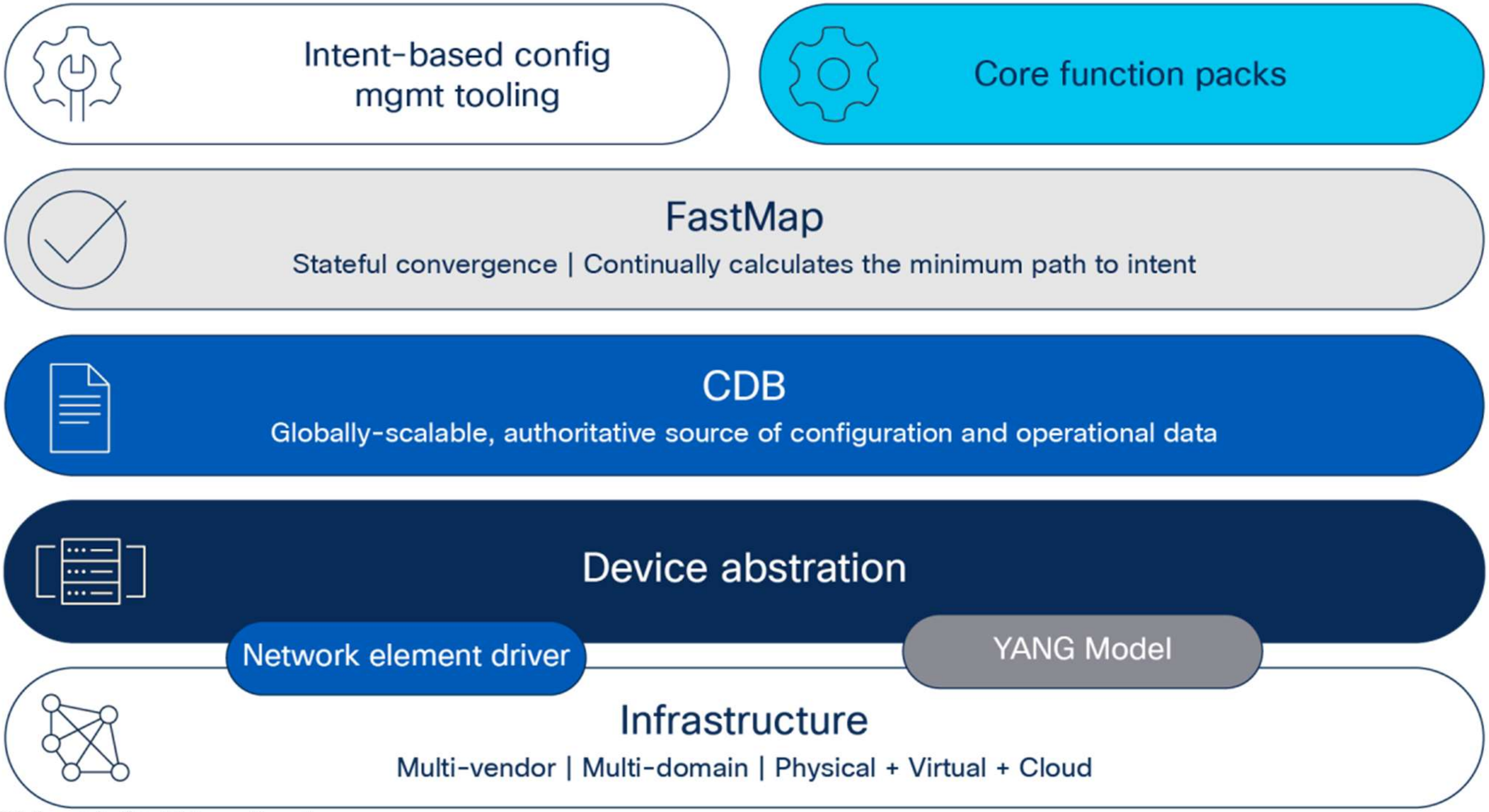
Business Owner

Enable the organization for rapid growth

NSO as the bridge between intent and action



NSO Architecture



Intent-based config mgmt tooling

Core function packs

FastMap

Stateful convergence | Continually calculates the minimum path to intent

CDB

Globally-scalable, authoritative source of configuration and operational data

Device abstration

Network element driver

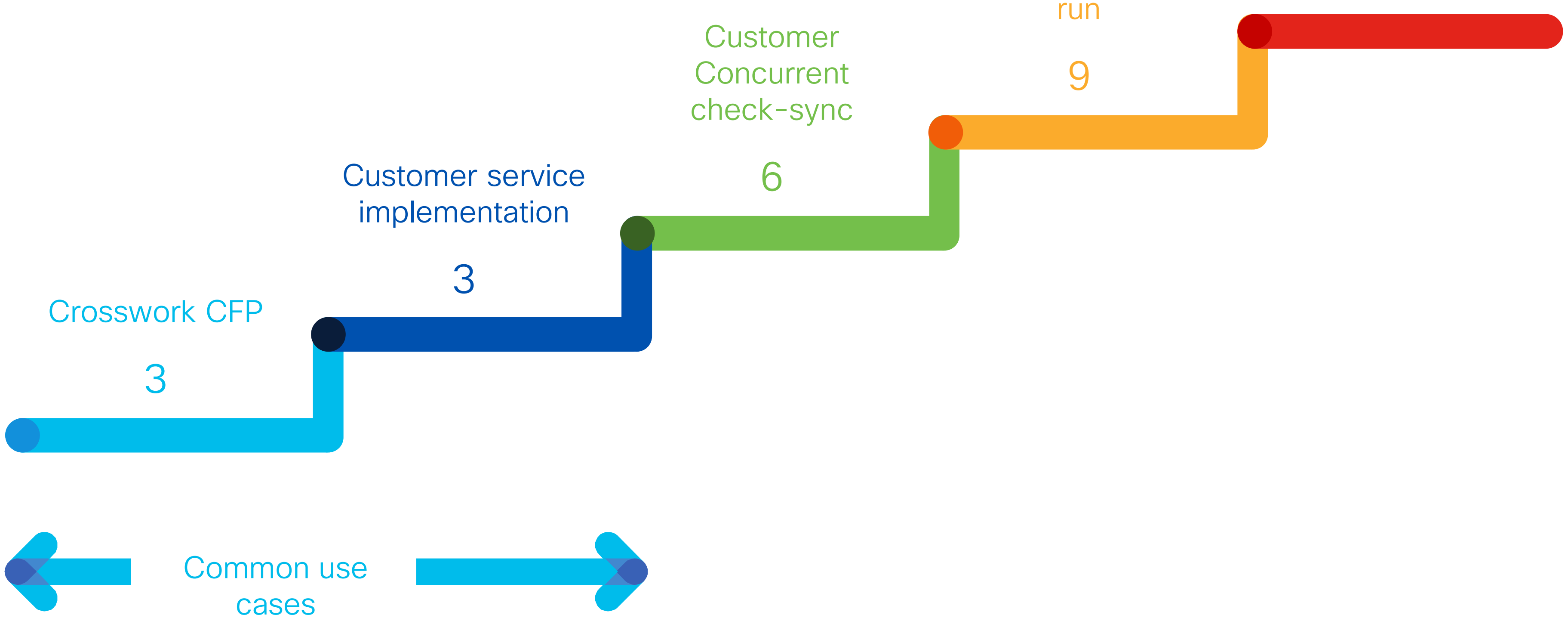
YANG Model

Infrastructure

Multi-vendor | Multi-domain | Physical + Virtual + Cloud

NSO 6.0

NSO 6.0 performance boosts



Crosswork CFP

3

Customer service implementation

3

Customer Concurrent check-sync

6

Concurrent dry-run

9

Optimized concurrent checksync

9

Common use cases

1000+

Device integrations

NETCONF YANG

Standard based DIY device integration out of the box

170+ NEDs

Growth of one NED per month

YANG

One model, multiple protocols

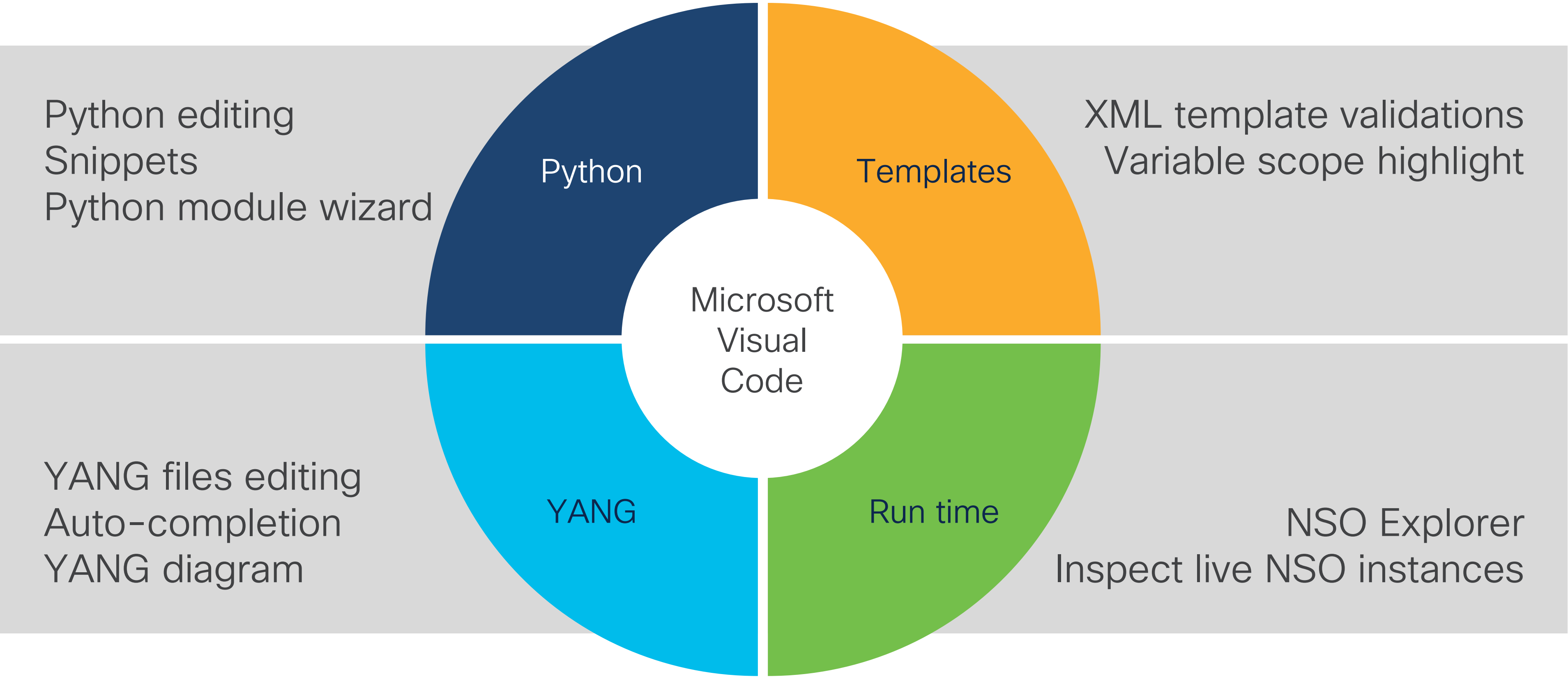
NETCONF

RESTCONF

gNMI

Developer Experience

NSO Developer Studio



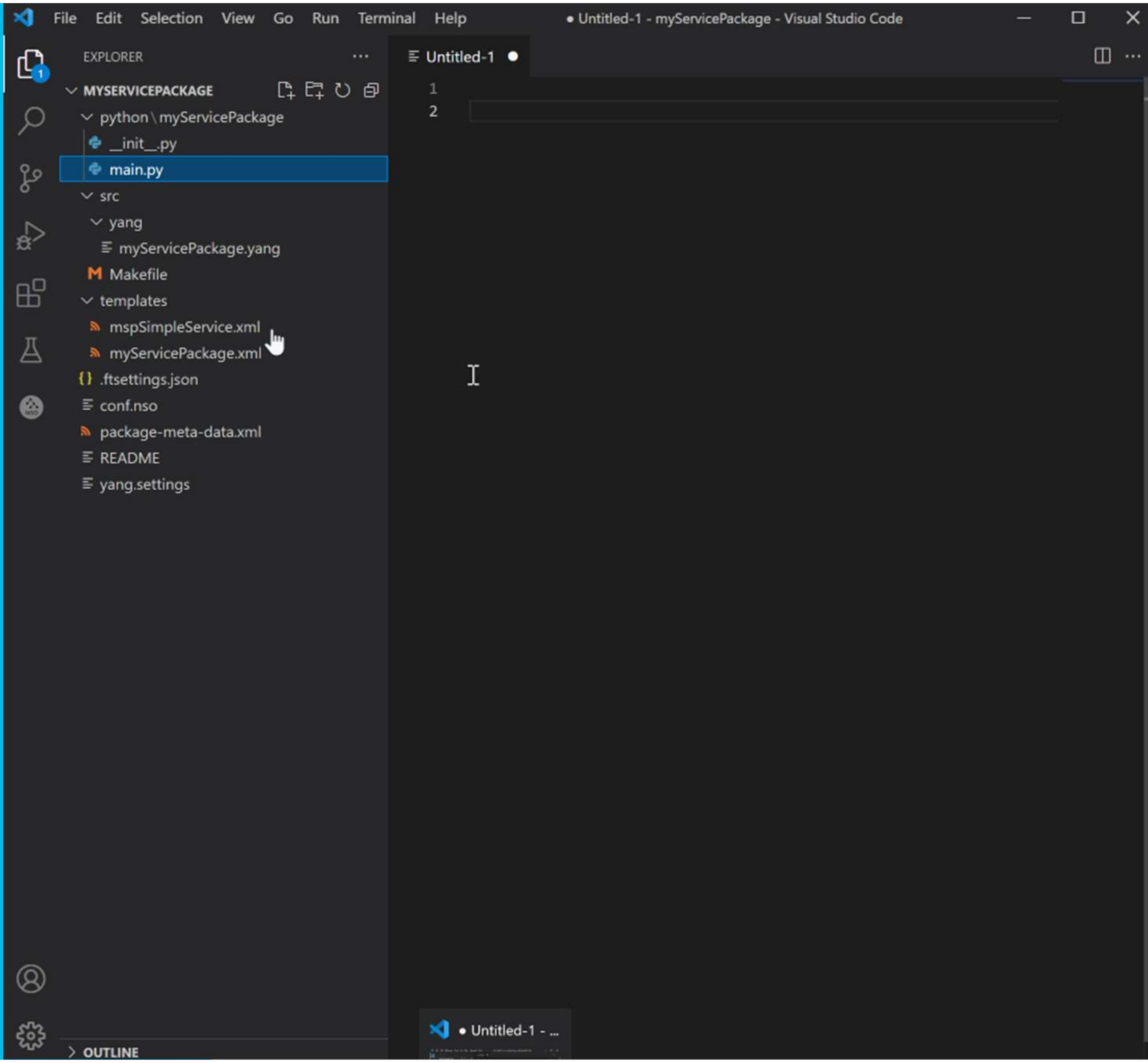
Python editing
Snippets
Python module wizard

XML template validations
Variable scope highlight

YANG files editing
Auto-completion
YANG diagram

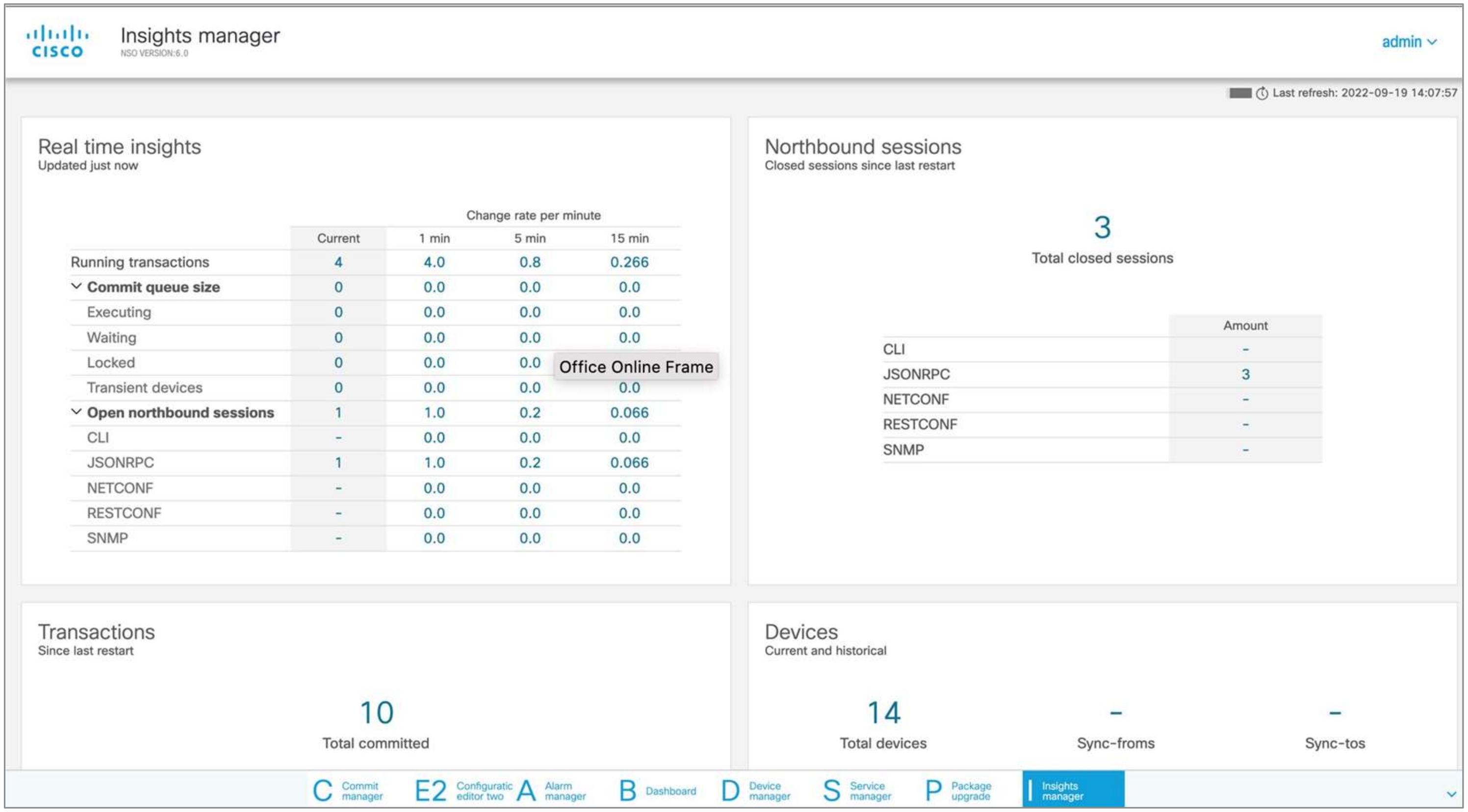
NSO Explorer
Inspect live NSO instances

NSO Developer Studio in Action



Developer Insights Manager

Provides data on operational state



Last refresh: 2022-09-19 14:07:57

Real time insights

Updated just now

		Change rate per minute		
	Current	1 min	5 min	15 min
Running transactions	4	4.0	0.8	0.266
Commit queue size	0	0.0	0.0	0.0
Executing	0	0.0	0.0	0.0
Waiting	0	0.0	0.0	0.0
Locked	0	0.0	0.0	
Transient devices	0	0.0	0.0	0.0
Open northbound sessions	1	1.0	0.2	0.066
CLI	-	0.0	0.0	0.0
JSONRPC	1	1.0	0.2	0.066
NETCONF	-	0.0	0.0	0.0
RESTCONF	-	0.0	0.0	0.0
SNMP	-	0.0	0.0	0.0

Northbound sessions

Closed sessions since last restart

3

Total closed sessions

	Amount
CLI	-
JSONRPC	3
NETCONF	-
RESTCONF	-
SNMP	-

Transactions

Since last restart

10

Total committed

Devices

Current and historical

14	-	-
Total devices	Sync-froms	Sync-tos

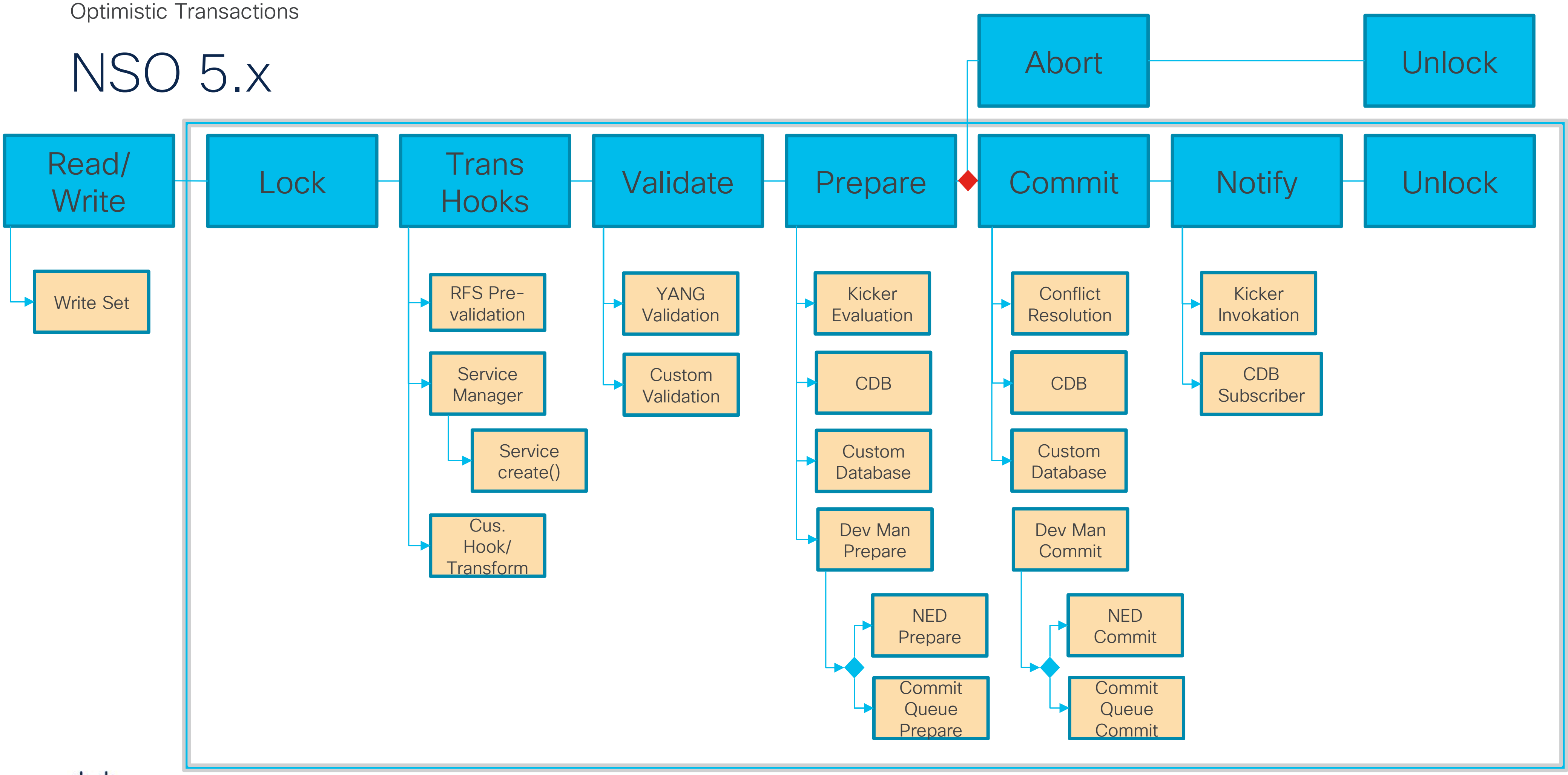
Optimistic Transactions

What have we achieved?

- Kept the NSO programming model
  - ACID transactions
  - FASTMAP Services
- Reduced the locking time needed for atomicity
- Moved to optimistic transactions
  - *Trust, but verify*
  - Instead of locking we check if we should have locked
- Greatly increased throughput for concurrent transactions

Optimistic Transactions

NSO 5.x



Abort

Unlock

Read/ Write

Write Set

Lock

Trans Hooks

RFS Pre-validation

Service Manager

Service create()

Cus. Hook/ Transform

Validate

YANG Validation

Custom Validation

Prepare

Kicker Evaluation

CDB

Custom Database

Dev Man Prepare

NED Prepare

Commit Queue Prepare

Commit

Conflict Resolution

CDB

Custom Database

Dev Man Commit

NED Commit

Commit Queue Commit

Notify

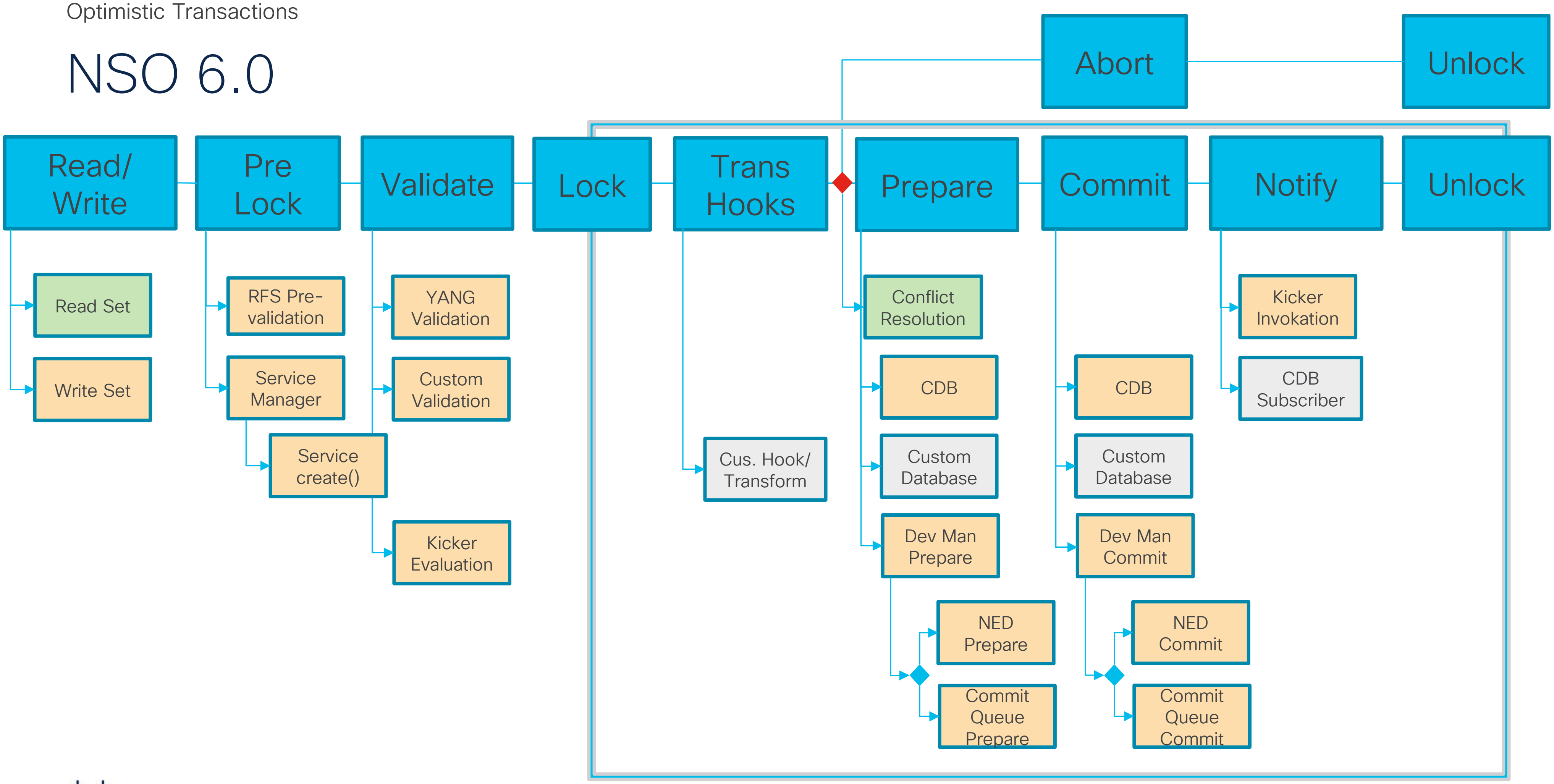
Kicker Invokation

CDB Subscriber

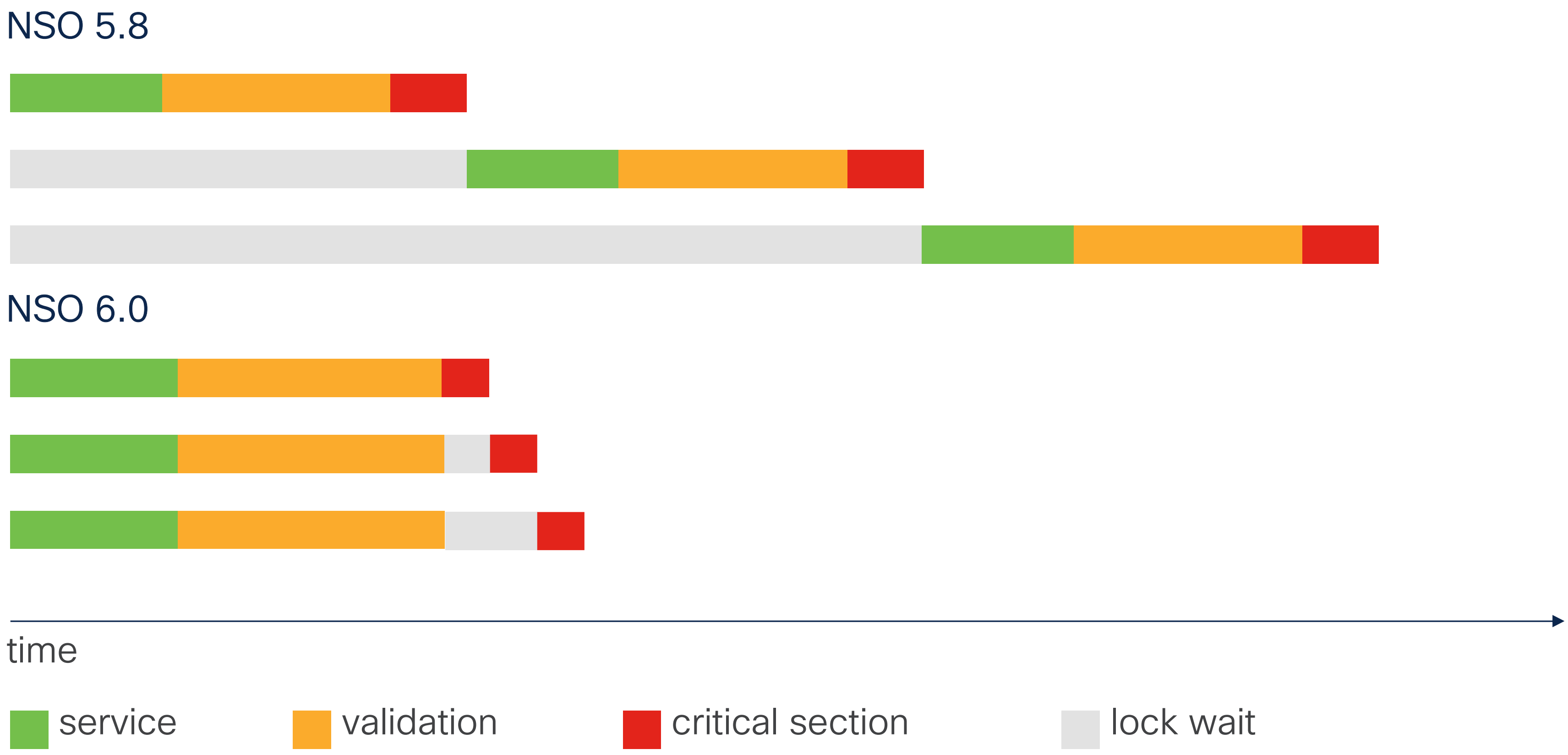
Unlock

Optimistic Transactions

NSO 6.0



Running 3 services in parallel



Performance Increase

Three major areas with performance boost

1 **Service Creation**

The `create` code is outside the lock. Drastic concurrency improvements.

2 **Commit Dry-Run / Service Check-Sync**

Runs completely outside the lock, even for `outformat native`.

3 **Validations**

Run outside of the lock.

Performance Improvement

CNC Core Function Pack Overview

- Transport SDN Automation Solution
- Function pack that automates
  - L2/L3 VPN Configurations
  - Segment Routing (SR) Policy Configurations
- Part of Crosswork Network Controller (CNC)

Test Results Summary: 10K Service Create

Seq	Number of Services	Services per Txn	No. of Threads	Number of Devices	Devices per Service	LSA	Commit Queue	HA
1	10K	1	1/5	300	4	No	Yes	No

Elapsed Time

	NSO 5.4 (Single Thread)	NSO 6.0 (Single Thread)	NSO 6.0 (5 Threads)
Total Time to converge	18 hr 39 min	11 hr 2 min	6 hr 46 min
Average Time/Service	9 services per min	15 services per min	25 services per min
Average Commit Time	6.7 seconds	3.95 seconds	12.16 seconds

Resource Utilization

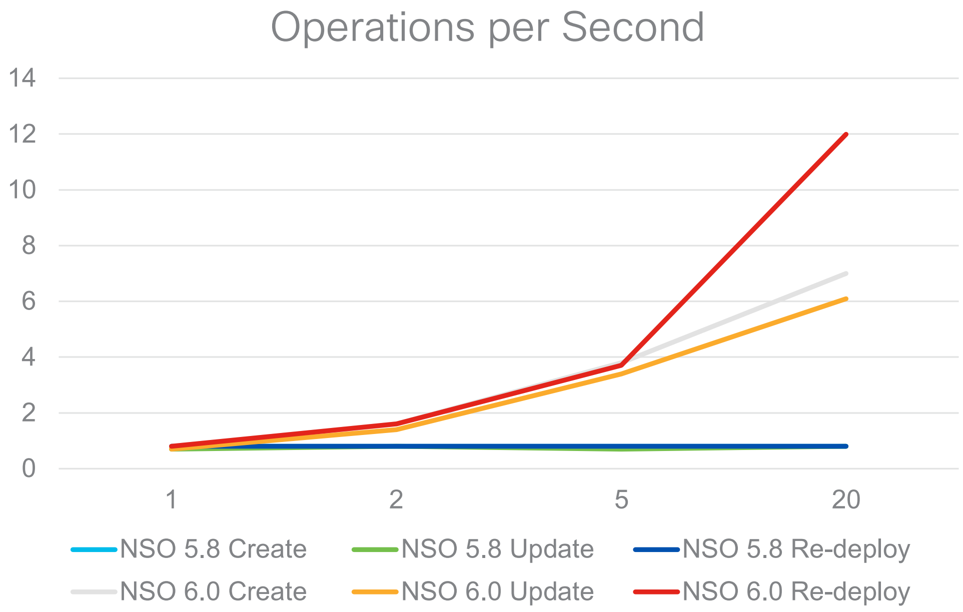
	NSO 5.4 (Single Thread)	NSO 6.0 (Single Thread)	NSO 6.0 (5 Threads)
Memory	7.6 GB	9.8 GB	10.8 GB
CDB	9.24 GB	7.8 GB	7.6 GB

Performance Improvement

Test Results

Key observations:

- Two threads give double throughput
- Maxes out at 9x create / updates
- Dry-run/Re-deploys about 15x



Simple python service

- 1 device per service
- 1 s to run create code
- 200 lines of device config
- Commit Queues

Implications for Developers

Conflicts and Restarts

- Optimism is not always justified
- Conflicting transactions must be aborted
- Automatic Service restarts
  - Advanced restart strategies implemented
  - The fallback is to serialize
  - Restarts mean the transaction is delayed
- Non-service conflicts have to be handled by the caller

Implications for Developer

Service Code

- Thread Safe
- Side-effect free
- Can be automatically restarted

Optimization

- Use Commit-Queues
- Limit reads
- Custom validation may slow things down

Less Locking

- Reduced lock for services
- No lock for dry-run
- Tools for restarting non-service transactions

Takeaways

- Up to 9X performance boost for adding new services, devices, workflows for optimized applications
- Up to 3X performance enhancement with existing code
- Advanced developer tools

For more information

Crosswork NSO:

www.cisco.com/go/nso

Crosswork Network Automation:

www.cisco.com/go/crosswork

The bridge to possible

Additional Slides

Service Commit

NSO 5.7

216 ms

```
2022-01-26T22:12:07.055 applying transaction...
entering validate phase for running usid=43 tid=8969
 2022-01-26T22:12:07.056 grabbing transaction lock... ok (0.000 s)
 2022-01-26T22:12:07.086 creating rollback file... ok (0.003 s)
 2022-01-26T22:12:07.090 run transforms and transaction hooks...
 2022-01-26T22:12:07.090 run pre-transform validation... ok (0.001 s)
 2022-01-26T22:12:07.093 service /simple[name='myservice']: run service... ok
(0.132 s)
 2022-01-26T22:12:07.227 run transforms and transaction hooks: ok (0.137 s)
 2022-01-26T22:12:07.227 mark inactive... ok (0.001 s)
 2022-01-26T22:12:07.229 pre validate... ok (0.001 s)
 2022-01-26T22:12:07.230 run validation over the changeset... ok (0.001 s)
 2022-01-26T22:12:07.232 run dependency-triggered validation... ok (0.000 s)
 2022-01-26T22:12:07.232 check configuration policies... ok (0.000 s)
leaving validate phase for running usid=43 tid=8969 (0.176 s)
entering write-start phase for running usid=43 tid=8969
 2022-01-26T22:12:07.232 cdb: write-start
 2022-01-26T22:12:07.232 ncs-internal-service-mux: write-start
 2022-01-26T22:12:07.237 check data kickers... ok (0.000 s)
leaving write-start phase for running usid=43 tid=8969 (0.005 s)
entering prepare phase for running usid=43 tid=8969
 2022-01-26T22:12:07.238 cdb: prepare
 2022-01-26T22:12:07.238 ncs-internal-device-mgr: prepare
leaving prepare phase for running usid=43 tid=8969 (0.011 s)
entering commit phase for running usid=43 tid=8969
 2022-01-26T22:12:07.249 cdb: commit
 2022-01-26T22:12:07.262 ncs-internal-device-mgr: commit
 2022-01-26T22:12:07.272 releasing transaction lock
leaving commit phase for running usid=43 tid=8969 (0.023 s)
 2022-01-26T22:12:07.272 applying transaction: ok (0.217 s)
commit-queue {
    id 1643263927247
    status async
}
Commit complete.
```

NSO 6.0

6 ms

```
admin@ncs(config)# commit | details
 2022-10-12T19:32:20.012 applying transaction...
 2022-10-12T19:32:20.012 waiting to apply... ok (0.000 s)
entering validate phase for running usid=49 tid=191 trace-id=829c6877-e512-48f0-82bb-
1cd0defaf37b
 [...]
 2022-10-12T19:32:20.043 creating pre-transform checkpoint... ok (0.010 s)
 2022-10-12T19:32:20.053 run pre-transform validation... ok (0.000 s)
 2022-10-12T19:32:20.053 creating transform checkpoint... ok (0.019 s)
 2022-10-12T19:32:20.073 run transforms and transaction hooks...
 2022-10-12T19:32:20.074 taking service write lock... ok (0.000 s)
 2022-10-12T19:32:20.074 holding service write lock...
 2022-10-12T19:32:20.281 service /simple[name=myservice']: run service: ok (0.205 s)
 2022-10-12T19:32:20.282 run transforms and transaction hooks: ok (0.208 s)
 [...]
 2022-10-12T19:32:20.284 check for read-write conflicts... ok (0.010 s)
 2022-10-12T19:32:20.294 taking transaction lock... ok (0.000 s)
 2022-10-12T19:32:20.294 holding transaction lock...
 2022-10-12T19:32:20.295 check for read-write conflicts... ok (0.000 s)
 2022-10-12T19:32:20.295 applying service meta-data... ok (0.001 s)
leaving validate phase for running usid=49 tid=191 trace-id= [...]
entering write-start phase for running usid=49 tid=191 trace-id= [...]
 2022-10-12T19:32:20.297 cdb: write-start
 2022-10-12T19:32:20.297 ncs-internal-service-mux: write-start
 2022-10-12T19:32:20.297 cdb: match subscribers... ok (0.000 s)
 2022-10-12T19:32:20.297 cdb: create pre commit running... ok (0.000 s)
 2022-10-12T19:32:20.297 cdb: write changeset... ok (0.000 s)
 2022-10-12T19:32:20.298 check data kickers... ok (0.000 s)
leaving write-start phase for running usid=49 tid=191 trace-[...]
entering prepare phase for running usid=49 tid=191 trace-id=[...]
 2022-10-12T19:32:20.298 cdb: prepare
leaving prepare phase for running usid=49 tid=191 trace-id=[...]
entering commit phase for running usid=49 tid=191 trace-id=[...]
 2022-10-12T19:32:20.299 cdb: commit
 2022-10-12T19:32:20.299 cdb: switch to new running... ok (0.000 s)
 2022-10-12T19:32:20.300 holding service write lock: ok (0.226 s)
 2022-10-12T19:32:20.300 holding transaction lock: ok (0.006 s)
leaving commit phase for running usid=49 tid=191 trace-id= [...]
 2022-10-12T19:32:20.301 applying transaction: ok (0.288 s)
Commit complete.
```

Check-Sync

NSO 5.7

511 ms

```
admin@ncs# simple thr1-1 check-sync | details
 2022-01-27T07:49:04.022 applying transaction...
entering validate phase for running usid=3025 tid=9536 trace-id=...
 2022-01-27T07:49:04.022 grabbing transaction lock... ok (0.000 s)
 2022-01-27T07:49:04.024 run transforms and transaction hooks...
 2022-01-27T07:49:04.025 run pre-transform validation... ok (0.000 s)
 2022-01-27T07:49:04.027 service /simple[name='thr1-1']: run service... ok
(0.470 s)
 2022-01-27T07:49:04.518 run transforms and transaction hooks: ok (0.494 s)
 2022-01-27T07:49:04.518 mark inactive... ok (0.000 s)
 2022-01-27T07:49:04.519 pre validate... ok (0.000 s)
 2022-01-27T07:49:04.519 run validation over the changeset... ok (0.000 s)
 2022-01-27T07:49:04.520 run dependency-triggered validation... ok (0.000 s)
 2022-01-27T07:49:04.520 check configuration policies... ok (0.000 s)
leaving validate phase for running usid=3025 tid=9536 trace-id=... (0.498 s)
entering write-start phase for running usid=3025 tid=9536 trace-id=...
 2022-01-27T07:49:04.521 cdb: write-start
 2022-01-27T07:49:04.522 ncs-internal-service-mux: write-start
 2022-01-27T07:49:04.523 check data kickers... ok (0.000 s)
leaving write-start phase for running usid=3025 tid=9536 trace-id=... (0.003 s)
entering prepare phase for running usid=3025 tid=9536 trace-id=...
 2022-01-27T07:49:04.524 cdb: prepare
 2022-01-27T07:49:04.525 ncs-internal-device-mgr: prepare
leaving prepare phase for running usid=3025 tid=9536 trace-id=... (0.007 s)
entering abort phase for running usid=3025 tid=9536 trace-id=...
 2022-01-27T07:49:04.531 cdb: abort
 2022-01-27T07:49:04.533 releasing transaction lock
leaving abort phase for running usid=3025 tid=9536 trace-id=... (0.002 s)
 2022-01-27T07:49:04.533 applying transaction: error (0.511 s)
in-sync true
```

NSO 6.0

```
admin@ncs# simple thr1-1 check-sync | details
 2022-01-27T15:37:56.152 applying transaction...
entering validate phase for running usid=916 tid=11564 trace-id=...
 2022-01-27T15:37:56.152 run transforms and transaction hooks...
 2022-01-27T15:37:56.153 run pre-transform validation... ok (0.000 s)
 2022-01-27T15:37:56.155 service /simple[name='thr1-1']: run service... ok
(0.499 s)
 2022-01-27T15:37:56.716 run transforms and transaction hooks: ok (0.564 s)
 2022-01-27T15:37:56.717 mark inactive... ok (0.000 s)
 2022-01-27T15:37:56.717 pre validate... ok (0.000 s)
 2022-01-27T15:37:56.718 run validation over the changeset... ok (0.000 s)
 2022-01-27T15:37:56.719 run dependency-triggered validation... ok (0.001 s)
 2022-01-27T15:37:56.720 check configuration policies... ok (0.000 s)
leaving validate phase for running usid=916 tid=11564 trace-id=... (0.568 s)
entering write-start phase for running usid=916 tid=11564 trace-id=...
 2022-01-27T15:37:56.721 ncs-internal-device-mgr: write-start
leaving write-start phase for running usid=916 tid=11564 trace-id=...
(0.005 s)
 2022-01-27T15:37:56.726 applying transaction: interrupted (0.574 s)
in-sync true
```

No lock!

Real World Improvements
Customer A

- Unmodified customer code
- Represents the base setup of a router
- The bottle-neck moves to the device

Test Results

	NSO 5.6.4	NSO 6.0	
Treads	t/min	t/min	Speed up
1	43	43	0
2	47	71	1.7x
4	46	106	2.5x
16	45	122	2.8x

Testcase
Customer service in python + complex template
Cisco IOS XR device
Create code alone ~1s / ~600 config nodes/service