Cloud RAN Challenges and Solutions

Rajeev Agrawal, Anand Bedekar, Troels Kolding, Vishnu Ram

NOKIA

Tackling future network and service complexity

Operator key drivers for cloud-based radio

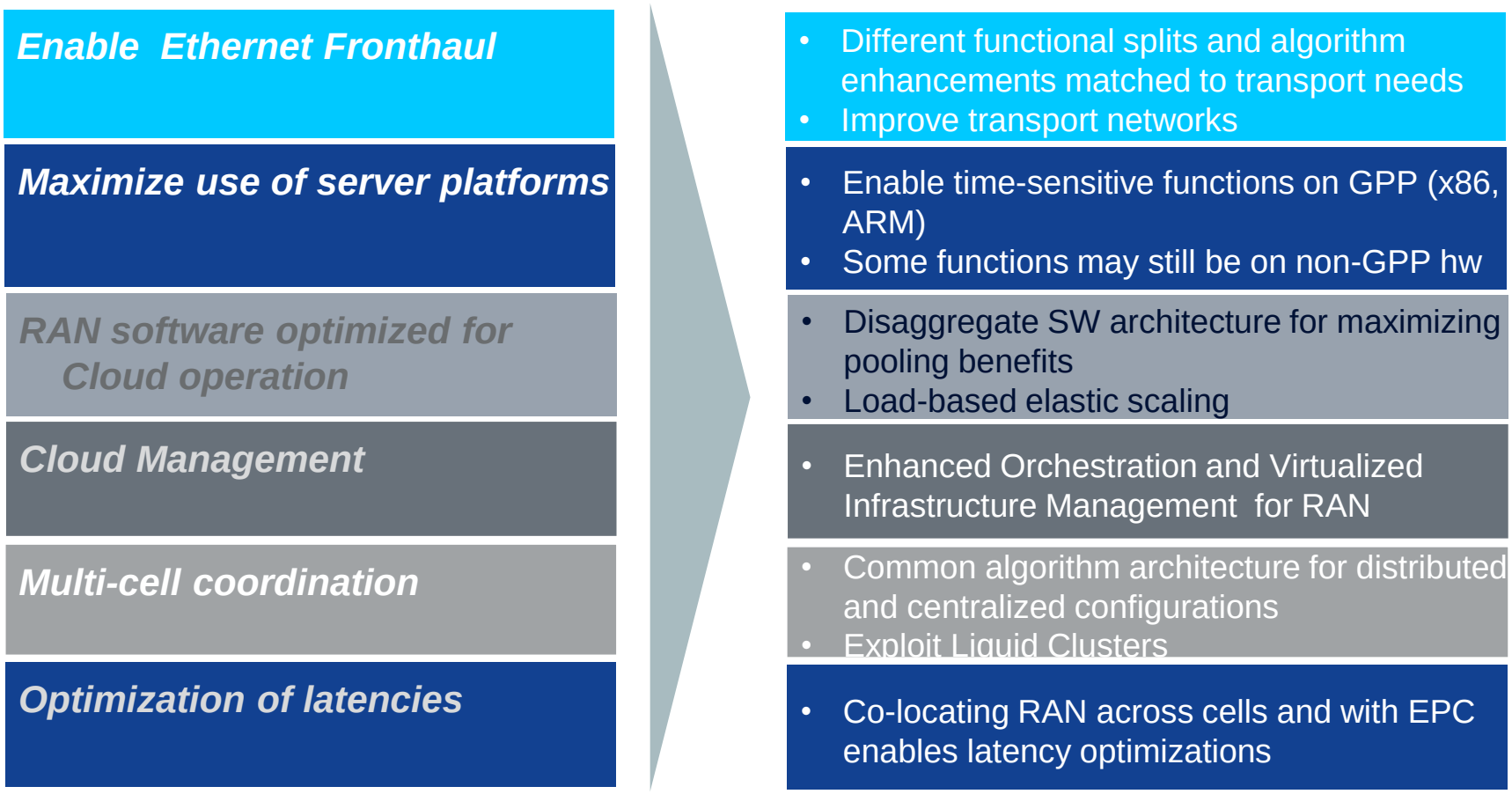
(1) Reduce site rental, energy, and operational costs
(2) Streamline maintenance operations
(3) Faster rollout of new features, services, and capabilities
(4) Lowering costs leveraging the economies of scale of the IT industry

Innovations drive data growth

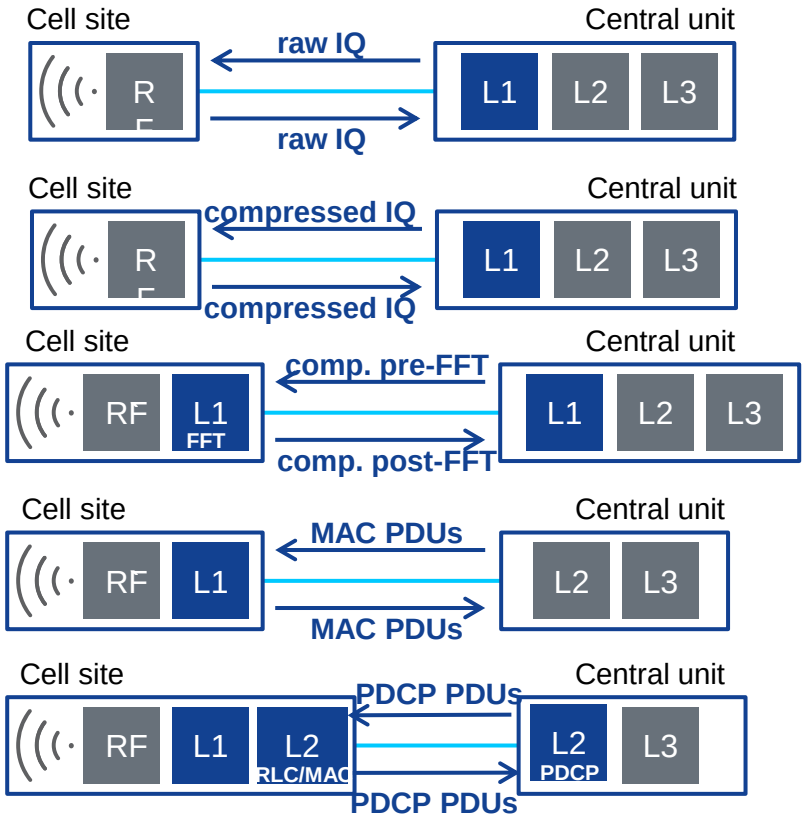
10^1 10^2 10^3 10^4

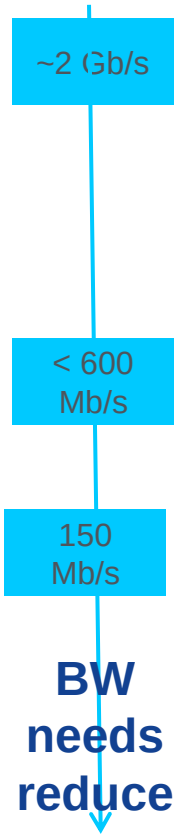
Data growth drives cloud-based radio

Overview of Cloud RAN challenges and solutions



Fronthaul needs affected by Functional Splits





Latency needs

HARQ loop goes across Fronthaul

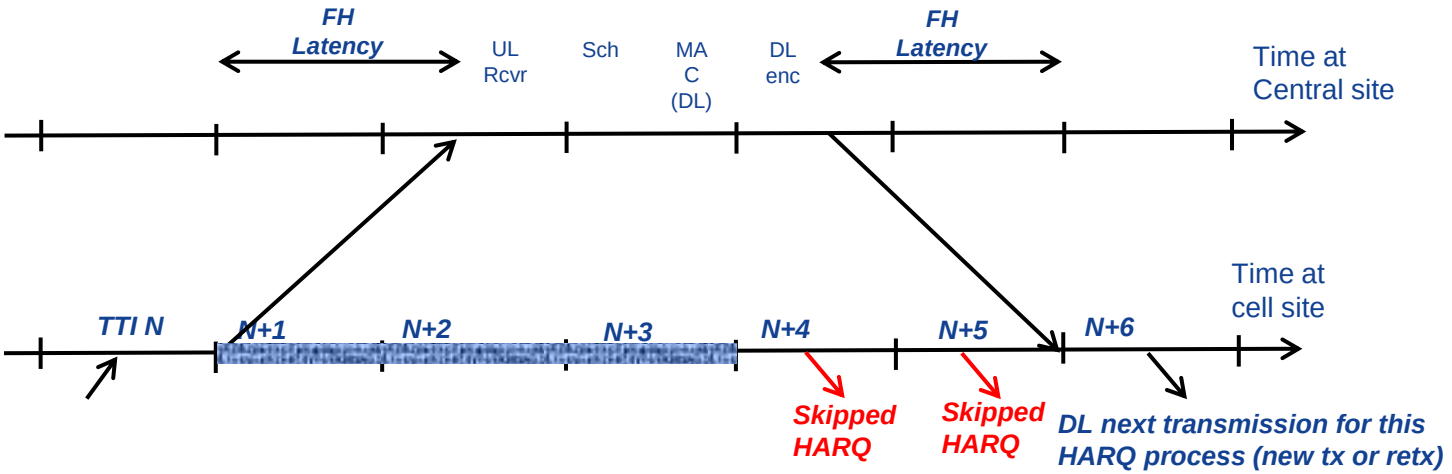
< 0.5 ms one-way for ideal performance

Scale of centralization may be constrained by fronthaul latency

Multi-cell coordination gains also affected by functional split

HARQ loop terminates at at cell site

HARQ centralization constrains FH latency, Centralization scale



3GPP-standardized HARQ cycle: UE tx in TTI n => eNB acks, retx in N+4

LTE HARQ RTT limits time budget for Front-haul, Processing

Fiber prop delay: 100us per 20km
FH Jitter consumes time budget (dejittering)

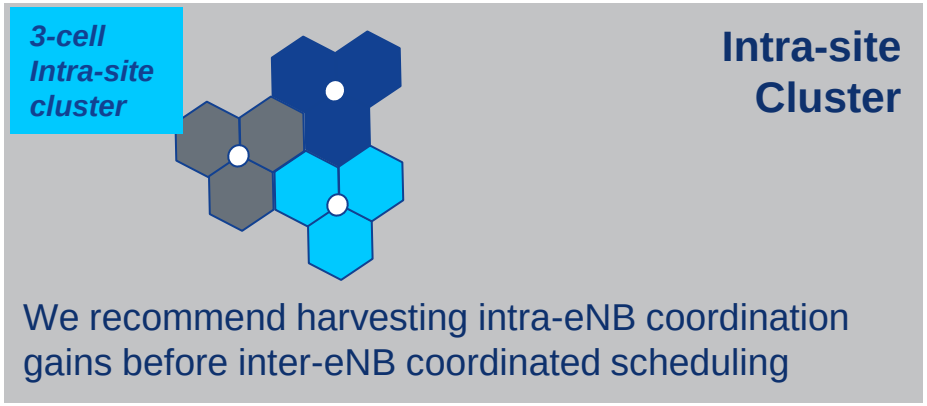
FH Latency (and distance) limits scale of centralization of **real-time RAN:** Likely to have many **"smallish clouds"** (Non-real-time RAN allows larger scale)

Variability in compute/execution environment eats budget – leaves less for FH

Tight control of jitter in processing time

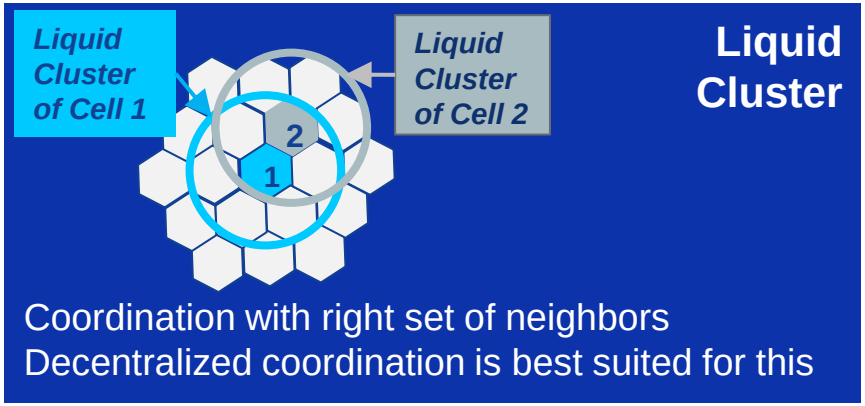
Multi-cell Coordination with Liquid Clusters

Intra-site Cluster



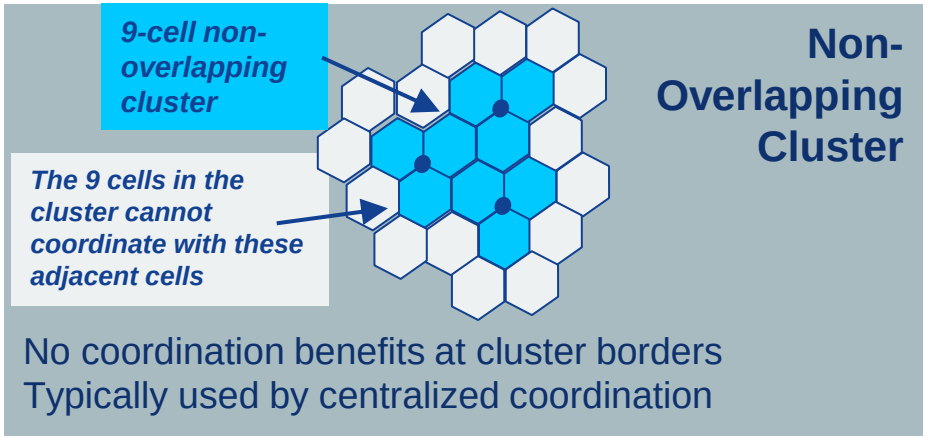
We recommend harvesting intra-eNB coordination gains before inter-eNB coordinated scheduling

Liquid Cluster



Coordination with right set of neighbors
Decentralized coordination is best suited for this

Non-Overlapping Cluster

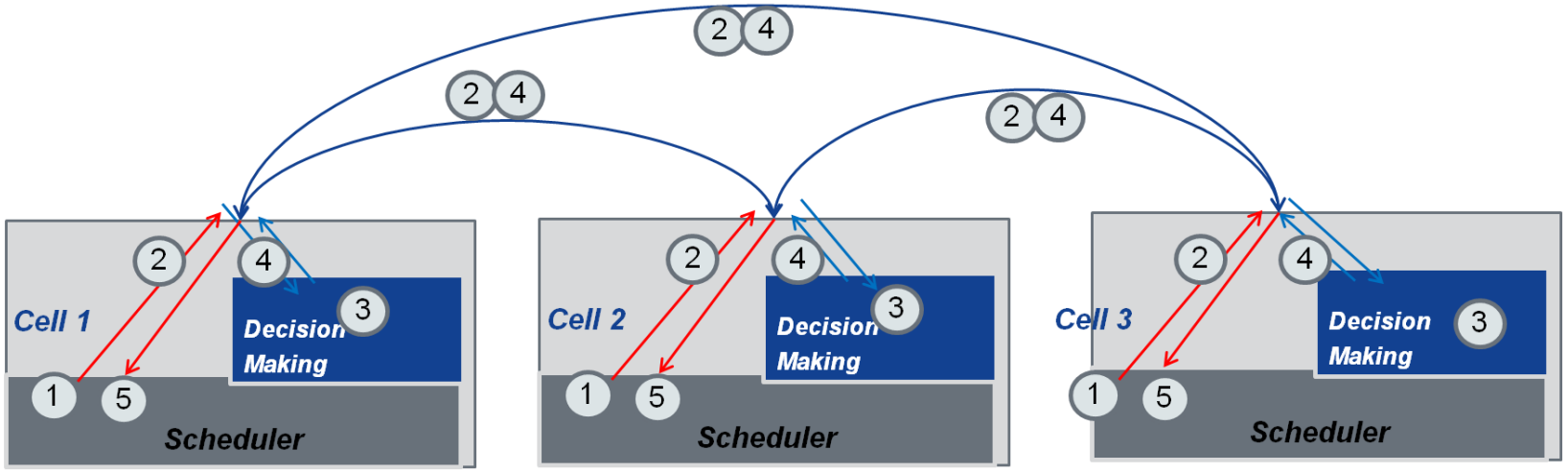


No coordination benefits at cluster borders
Typically used by centralized coordination

Maximize cooperation gains – no hard cluster boundaries
Applies in a variety of coordination algorithms – Coordinated Scheduling, Dynamic Point Selection, UL Joint Reception CoMP, …

Liquid Clusters enable flexible configuration of cooperating cells

Decentralized Algorithm Architecture for Multi-Cell Coordination

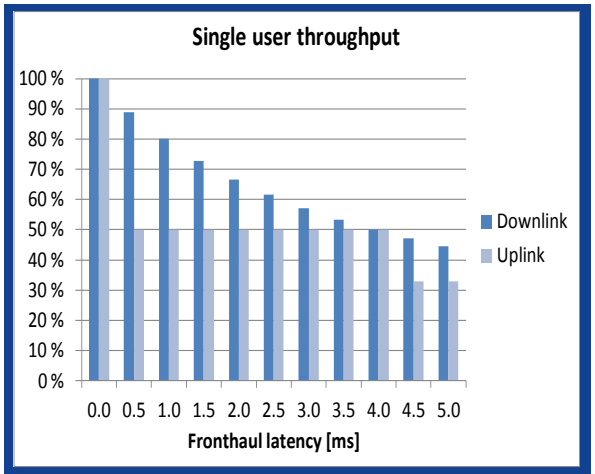


Decentralized algorithm architecture for Multi-cell coordination: can achieve equal or better performance than centralized algorithms

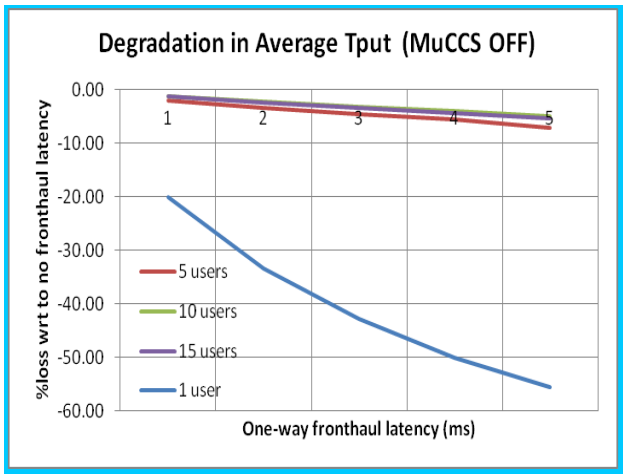
When scheduler (or L1) is centralized: Within central site, coordination among distributed processors

Liquid Clusters and Decentralized algorithm architecture enable flexible mapping of functionality within physical cloud architecture

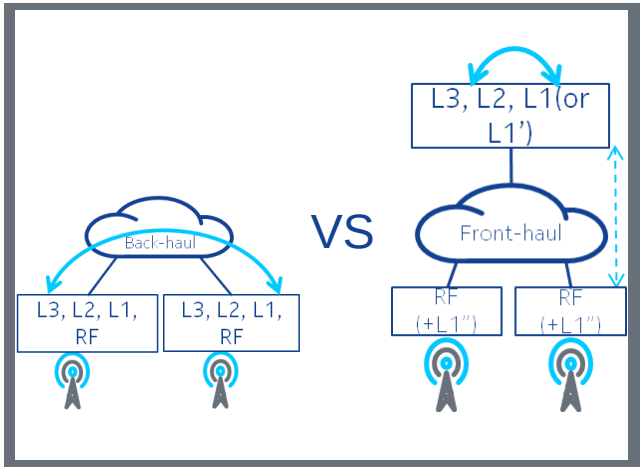
Impact of non-ideal fronthaul latency*



Single-user throughput: Significant impact – UL worse than DL



DL Multi-user scenario:
At >5 UEs, recovers system throughput
UL needs Sch enhancements



For a given latency (FH or BH), Distributed coordination will outperform coordination in central site

ompanion paper - To appear in IEEE VTC Spring 2016

Virtualization of Real-time RAN Functions

L1 functions currently considered more efficient on DSP/SoC/FPGA hardware
For some L1 functions, special hw acceleration may be required (e.g. turbo decoder)

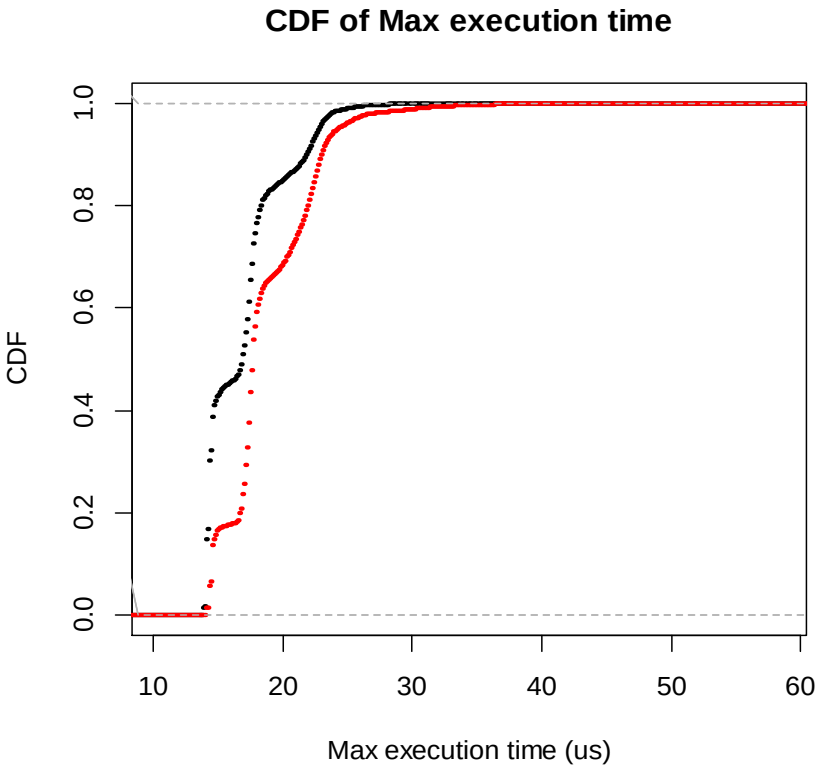
L1 may require non-GPP hardware

L2, Scheduler can be virtualized on GPP/Linux

"Real-time" functions L2, Scheduler can be virtualized

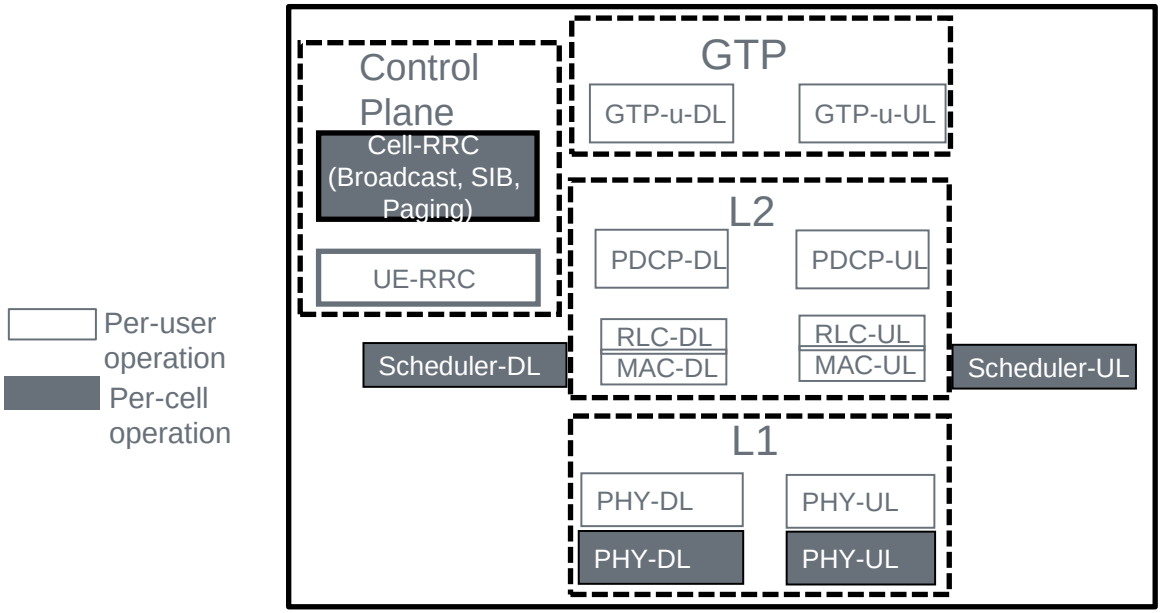
Methods of controlling execution jitter:
Poll-driven task dispatch, rather than interrupts
Careful mgmt. of host OS and BIOS interrupt handling
CPU core isolation and pinning, etc.
Both guest as well as host OS need careful tuning

Real-time virtualization requires control of execution jitter



Execution jitter of <30us shown to be achievable on x86/Linux
→ Good for LTE where TTI=1ms: 5G may need even tighter control due to shorter TTIs

SW Architecture – How to achieve Pooling for RAN functions



Certain functions have ***per-user operation,*** while others have ***per-cell operation***

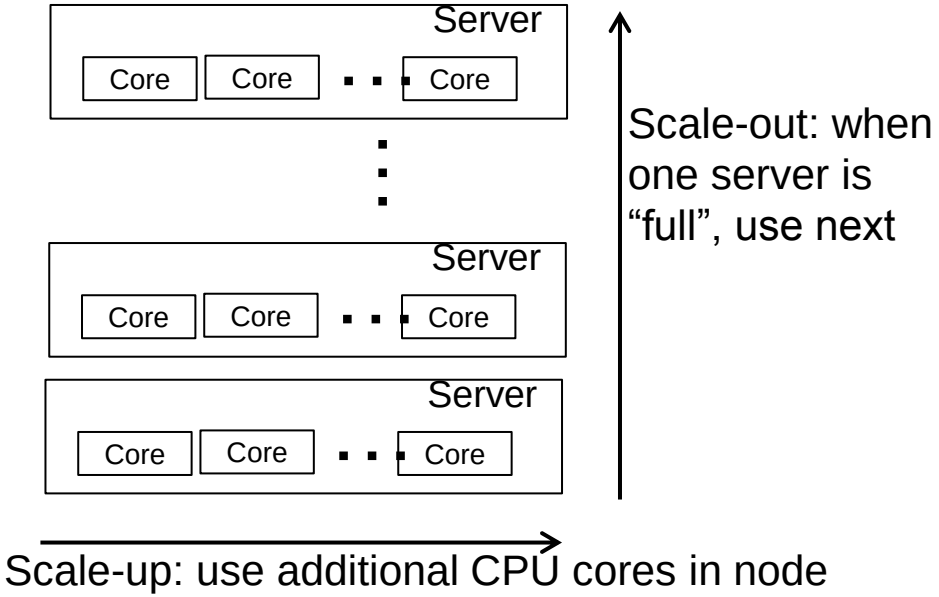
Per-user operation: Compute requirements scale with the number of users

Per-cell operation: Compute requirements scale with number of cells

Disaggregate and Reassemble RAN functions: provide the right form of instantiation for each function

For Per-user operations, can do pooling with ***horizontal aggregation*** – any processing unit (e.g. VM) can handle any user from any cell

Elastic Scaling for RAN

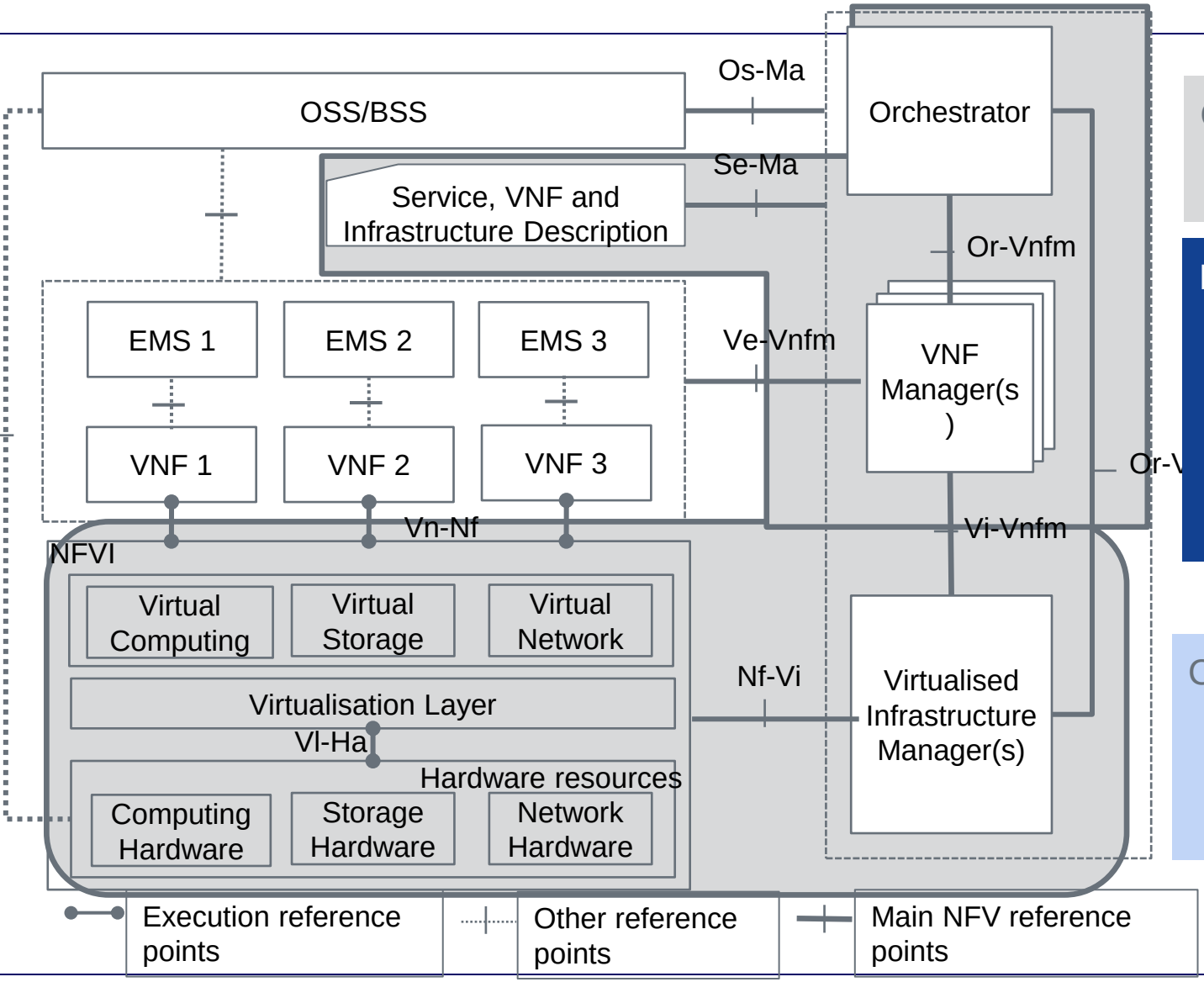


Elasticity: Dynamically change amount of processing resources consumed

Support **Scale-up** (cores within a server/VM) as well as **Scale-out** (number of servers/VMs)

Algorithm to automatically adapt number of CPUs based on load, transport latency/jitter, processor type, real-time deadlines, etc.

Cloud Management for RAN



Consistent with NFV reference architecture

But RAN has special needs:

- Functionality at guest, host OS for real-time operation
- Mix of GPP, non-GPP HW
- Elastic Scaling needs special cooperation b/w guest and host OS

Current tools e.g. Openstack need extensions to meet RAN needs

Summary

Maximize coordination gains, flexible placement of coordinating functions

Multi-cell coordination with Decentralized Algorithm Architecture and Liquid Clusters

Choice of functional split is a complex Tradeoff BW, latency/jitter, multi-cell coordination, cost/complexity

Functional splits enable fronthaul flexibility: Multi-dimensional tradeoff

L1 may still need non-GPP HW, accelerators
L2 and above can be virtualized: but requires control of execution jitter

Maximize use of GPP/Linux Some functions need non-GPP

Disassemble and Reassemble RAN functions for optimal pooling
Elastic Scaling for automatic adaptation to changing loads, real-time operation KPIs

Pooling: Exploit characteristics of Per-user and Per-cell ops

Mix of GPP and non-GPP hardware in compute infrastructure
Support operation and scaling of real-time RAN ops on GPP

Cloud Management and Orchestration may need special support for RAN

Thank you!

#ICIN2016

NOKIA