CloudIQ

Anand Muralidhar (anand.muralidhar@alcatel-lucent.com)

Sourjya Bhaumik, Shoban Chandrabose, Kashyap Jataprolu, Gautam Kumar, Paul Polakos, Vikram Srinivasan, Thomas Woo
Baseband processing at a Base station

- Base station (BTS) consists of Antenna and baseband unit (BBU)
- BBU processes cellular signals
  - Provisioned to handle peak load
  - Needs active cooling, maintenance
- But typical cells have 30% load...
- Can we exploit variations in load to save energy?
  - Energy constitutes 25% of total cost

[C-Ran White paper]
Cloud-RAN: Cellular processing in the cloud

- Cloud-Radio access network (C-RAN)
  - Centralized baseband processing
  - Backhaul IQ samples from BTS

- Many advantages
  - OPEX savings – site visits, upgrades, cooling
    [up to 50%, C-RAN white paper]
  - Higher spectral efficiency – Network MIMO
    [up to 50%, C-RAN white paper]

- Real-time constraints limit fiber length
Transition towards programmable hardware

- From ASIC/SoC to general purpose processors (GPP)
- GPPs have increased ability to process baseband signals
  - Multi-core, SIMD, cache, DVFS
  - Performance-per-watt is increasing
- GPP allows open platforms
  - Cellular operator not bound to a vendor
- Can we replace BBUs with GPPs?
Cellular processing over GPPs

- Baseband processing on GPPs in data center
  - Exploit variations in BTS load
  - Pooling: process multiple BTS on single GPP
  - Energy savings: switch it off if not used
- What are the potential pooling gains?
  - Analyze real-world data
- CloudIQ: A resource management framework
- How do we build the system?
Outline of the talk

- LTE profiling results
- Analysis of real-world traffic
- CloudIQ: A resource management framework
- System Design
A brief primer on LTE

- OFDMA: spectrum is divided to physical resource blocks (PRB)
- Subframe processed every 1 ms
  - Send ACK/NACK after 3 ms
- PHY layer is compute intensive – FFT/IFFT, Turbo codes
Profile processing load in LTE

- OpenAir: LTE, 5MHz
  - Open source, from Eurecom – France
  - Executes on GPPs
- Profile code by varying modulation and coding schemes (MCS)
- Load = time to process a subframe
Observations after profiling code

- Load is (almost) linear → base load + dynamic load
- Large variation – offers potential for pooling many BTS
- Extend LTE observations to WCDMA
Analysis of real-world traffic

- WCDMA in dense urban setting
  - 175 base stations, 2 weeks
- Downlink logs, aggregated at 15 mins
  - QAMs used, codes available, total traffic
- Derive “distribution” on load in subframe
Guarantees for processing multiple BTS on GPP

- Hard guarantee – every subframe is processed correctly
  - Cannot guarantee if multiple BTSs processed in one GPP
  - BTS load can suddenly peak and miss deadlines

- “Statistical guarantees” – pick failure prob $P_F$
  - Calculate load $L$ for BTS
  - Load exceeds $L$ with probability $P_F$
  - Acceptable model for cellular systems
Potential gains from resource pooling

- Choose failure prob $P_F$ – select load $L$ for each BTS
- Compute total load across all BTSs
- All signals processed by one computing resource
  - Assume it can handle peak load
- How much is the resource utilized?
  - 22% pooling gains at $P_F = 10^{-8}$
  - Conservative estimate on total load
CloudIQ: A resource management framework

- Set of BTSs to be scheduled
- Set of compute resources – multi-core GPPs
- Solve two coupled problems:
  - **PART**: partition BTSs to sets to be scheduled on GPP
  - **SCHED**: real-time schedule to process each set on a GPP
- Design real-time system with statistical guarantees
- Separation principle: decouple **PART** and **SCHED**
  - Design around a simple cyclic schedule
Cyclic schedule

- $M$ BTSs need to be scheduled on $N$-core GPP
- BTS processes subframe every 1 ms
  - Processing load = $p$ ms, \( p > 1 \)
  - Deadline = $d$ ms, \( d > p > 1 \)
  - Most real-time systems consider \( p < d = 1 \)
- Cyclic schedule
  - Offline schedule, subframe processed in a core
  - At time $t$, $j$-th BTS

Core ((tM + j) mod (N) + 1)
Example of cyclic schedule

- Cores = 4, BTS = 3
- $B_i(t) = i$-th BTS’s job at time $t$
- Each BTS has $p = 4/3$, $d = 2$
- Same order repeats across cores

![Diagram showing cyclic schedule]

- Time 0
  - Core 1: $B_1(0)$, $B_2(1)$, $B_3(2)$, $B_1(4)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
- Time 1
  - Core 1: $B_1(4)$, $B_2(5)$, $B_3(6)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
- Time 2
  - Core 1: $B_1(0)$, $B_2(1)$, $B_3(2)$, $B_1(4)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
- Time 3
  - Core 1: $B_1(4)$, $B_2(5)$, $B_3(6)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
- Time 4
  - Core 1: $B_1(0)$, $B_2(1)$, $B_3(2)$, $B_1(4)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
- Time 5
  - Core 1: $B_1(4)$, $B_2(5)$, $B_3(6)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
- Time 6
  - Core 1: $B_1(0)$, $B_2(1)$, $B_3(2)$, $B_1(4)$
  - Core 2: $B_2(0)$, $B_3(1)$, $B_1(3)$, $B_2(4)$
  - Core 3: $B_3(0)$, $B_1(2)$, $B_2(3)$, $B_3(4)$
  - Core 4: $B_1(1)$, $B_2(2)$, $B_3(3)$, $B_1(5)$
More examples of cyclic schedules

- Many configurations possible for different loads
  - One configuration = one GPP
- How many GPPs (or configurations) do we need?
Scheduling BTS on GPPs

- Choose $P_F$, compute load for each BTS

- How many GPPs do we need?

- Also, cyclic schedule allows many configs

$p = 2$

$p = 4/3$

$p = 1$
Solving **PART** and **SCHED**

- **PART**: Construct "super-BTS"
  - Solve variable size bin-packing
  - Sets of super BTSs are allocated to GPP

- **SCHED**: schedule the super-BTSs on GPPs using cyclic schedule
  - Delay guarantees come for free!

- Super-BTS creation is conservative
  - Deadline is missed only if *total* load is exceeded
CloudIQ solution

- Pick a $P_F$ and compute load for each BTS from its ccdf
- Solve PART and SCHED
  - Variable-size bin packing and cyclic schedule
  - 59 processors for peak load
  - 16% savings at $P_F = 10^{-8}$
System design

- Hardware: Intel Xeon W3690, 3.47 GHz, 6 cores
- OS: Linux 2.6.31 with PREEMPT_RT
- Made OpenAir multi-threaded
  - Need to make it cache conscious for better isolation
- Super BTS can take less time to execute than total time
  - Cache misses are fewer
Summary

- Analyzed WCDMA traces, pooling gains can exceed 20%
  - Statistical guarantees in processing signals
- Developed CloudIQ framework – achieves gains of up to 16%
  - Simple cyclic schedule for real-time guarantee
- Prototyped system on GPP
  - Multithreaded implementation of LTE
Ongoing efforts

- Heterogeneous systems
  - GPP + GPU + FPGA
  - Algorithms, architecture?

- Pooling decisions at smaller time scales
  - 10 ms, 100ms?
  - What are the savings in energy?

Thanks!