An Adaptive Power Management Method for Radio Access Network Data Plane Systems

Srihari Das Sunkada Gopinath, Sandeep Burugupally and Ajeet Singh Nathawat
Network Modem Team, Samsung R&D Institute India - Bangalore
{srihari.sgs, deshpandae.b, a.nathawat}@samsung.com

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771989

***Abstract*—The revolutionary 5G network technology is deployed to serve multiple end-users with extremely diverse requirements demanding high throughput and ultra-low latency. Radio Access Network (RAN) data plane systems need to support multiple user traffic flows requiring synchronization to maintain consistency of state variables. Multi-core network processors are used for supporting high throughput requirements due to their high processing capabilities. But network traffic varies over time and peak traffic is observed very rarely. Traffic variations over time provide opportunities for power saving to reduce operational costs. Existing power management solutions are conservative, reactive, and unaware of load distribution across multiple cores. Efficient packet distribution logic is imperative for achieving the desired performance at optimum power. We propose a novel adaptive power management solution. The proposed solution includes the design of a load-balanced multi-core packet distribution method for efficient use of processor cores considering per-flow synchronization, ordered packet delivery. We use a core load predictor to pro-actively decide the optimum frequency of processor cores according to traffic variations. Frequency changes are applied using global Dynamic Voltage and Frequency Scaling (DVFS). Experiments are performed on real network traces with multiple data flows across different Radio Bearers in NR (New Radio) RLC (Radio Link Control) at RAN Distributed Unit (DU). Results show that the proposed solution increases peak throughput of a flow by 65% and reduces processor power consumption by 10% and has at least 5% more power saving compared to traditional power management schemes.**

I. Introduction

5G and Beyond 5G networks target to empower a variety of next-generation network applications such as immersive extended Reality (XR), remote surgery, autonomous vehicles, and smart industry. Future networks are expected to provide high end-user throughput and ultra-low latency with high reliability to support these applications. RAN data plane systems are expected to serve multiple end-users and higher cumulative system throughput demands. Hence, multi-core processors are used to catering to such massive processing challenges. Multi-core processors are designed, equipped with enough processor cores and resources to handle peak traffic loads. They are usually combined with hardware accelerators, schedulers, and multiple co-processors to leverage parallel processing.

Due to the enormous growth of customers and the number of services with different QoS being offered by telecom operators, the energy efficiency issue has become a major concern for next-generation network devices. Telecom operators have reported alarming statistics of energy requirements [1]. Budget limitations and vast deployment requirements call for the design of a power-efficient packet processing method. Fig.1 is the data traffic pattern observed for 15 days at a 5G Base Station. There is continuous variation in traffic and peak throughput is rarely seen. These lower throughput periods can be utilized to operate the processor at lower power consuming state by reducing the frequency.

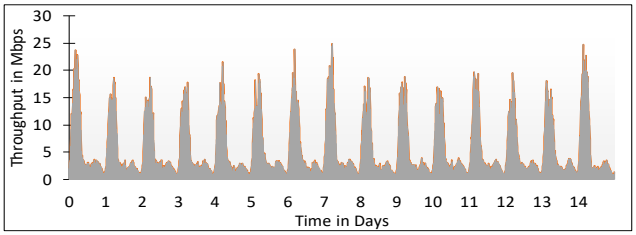


Fig. 1. Traffic pattern in 5G RAN Network

Dynamic Voltage Frequency Scaling (DVFS) is a mechanism to change frequency. Local and Global DVFS are two forms of DVFS. Local DVFS changes frequency per core, global DVFS makes frequency changes for the entire chip as voltage and frequency are shared among all processor cores. Throughput can be maximized when all cores are equally loaded. RAN data-plane protocols are expected to meet lower latency, higher throughput, ordered packet delivery, and data locality. It is expected to have per-flow synchronization to maintain consistency of state variables. The data flow distribution and processing method for load-balancing are to be designed considering peak user throughput and total system throughput requirements.

Predictive mechanisms can give better power reduction compared to reactive schemes as changes in power adaptations are applied before load changes. Power management schemes used in general-purpose processors do not consider throughput fluctuations and load balancing requirements for maximizing throughput in multi-flow packet processing systems. In this paper, we propose a comprehensive adaptive power management solution for multi-core data plane processors with the following contributions.

1) We propose a load-balanced multi-core packet distribution solution considering per-flow synchronization with no packet reordering. It improves per-flow peak throughput and reduces latency as multiple cores work on packets of same flow in parallel.
2) We propose a power management solution to choose optimum operating frequency based on core load pre-

diction. It also takes care of instantaneous traffic spikes and adjusts the frequency to avoid packet drops. It uses different frequency levels of the processor for frequency scaling.

3) We perform experiments on real network trace with varying throughput across different times of the day and compare with other power management and multi-core packet flow distribution solutions concerning power consumption, load balancing, and per-flow peak throughput.

It is also to be noted that the solution is not limited only to RAN data plane systems and can be considered across any communication network processor. It can be applied to the system where packets of different types are processed together. The rest of the paper is organized as follows. Section II describes the background and motivation. Section III presents some of the relevant works. Section IV describes the software architecture of the proposed Adaptive Power Management (APM) solution. Section V provides simulation setup and performance evaluation with experimental results. Section VI concludes the paper.

II. BACKGROUND AND MOTIVATION

In this section, we provide a brief overview of commonly used multi-core processor architecture in RAN systems, DVFS based power management, and major factors to be considered while designing a packet flow distribution model.

A. The architecture of the common RAN multi-core processor

Usually, in RAN processors, the processing is classified as control plane and data plane as shown in Fig.2. The Control plane or slow path consists of various call and session management messages. Due to limited processing requirements, a lesser number of cores are allocated to the control plane. Data plane or fast path data traffic consists of different packet types like downlink, uplink, and interlayer information data. PKTIO (packet input/output) gets data from interface ports and places packets in corresponding queues based on packet type. Cores can register to receive packets from specific queues. Packet scheduler distributes the packet across multiple cores based on core registrations and priority of packets. Similarly, packets are given back to PKTIO after processing to transmit out. Due to heavy processing requirements, more cores are allocated for the data plane.

B. Power Management using DVFS

The power consumption is mainly governed by the following equation $P \propto CV^2F$ [2] where P is the power, C is the capacitance, V is voltage and F is the working frequency. Authors in [3] observed that a core consumes less power when it operates at high utilization and low frequency, compared to low utilization and high frequency. Power consumption can be reduced by operating at lower frequencies. The transition from higher operating frequency to lower operating frequency during low traffic scenarios helps to achieve better power gains. DVFS based power management solutions also need to consider sharp traffic spikes to avoid packet drops.

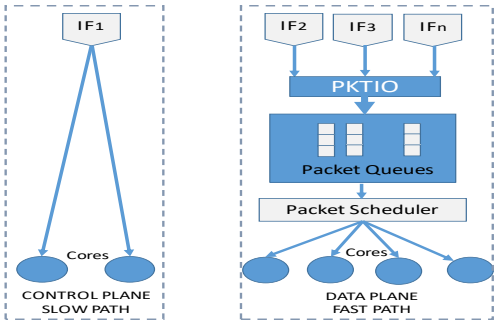


Fig. 2. Common RAN multi-core processor architecture

C. Factors for designing a packet flow distribution

Network traffic constitutes few heavy throughput flows and many low throughput flows. Traffic consists of elephants and mice [4]. Elephants represent the small number of high-volume transmissions that constitute the majority of the traffic mix; mice, on the contrary, are flows that are large in number but consume much less bandwidth.

Most traditional data decomposition methods of packet distribution stick to the core-affine model, i.e. each data flow is constrained to a single core to avoid overheads of critical sections and cache synchronization using hash-based core assignment. Hashing is typically done at the flow level. Values like five tuples identifying TCP/UDP sessions are used as parameters or keys to the hash function to determine the target core to process the packet. A similar decision can be done based on parameters like bearer session-id. At RAN data plane systems, these parameters remain constant for all packets of a flow. The same processor core is selected for processing packets of a flow, therefore packet order is maintained. It also effectively utilizes cache [5]. But core affinity has an upper bound in per-flow throughput, as it is restricted to single-core capacity and it can also cause an unbalanced load distribution resulting in lower system throughputs.

Throughput can be maximized when all cores are equally loaded. It also helps in effective prediction of core load and application of frequency changes using global DVFS. Naively distributing packets of same flow across multiple cores for load-balancing has multiple issues to be considered like data and instruction cache locality, packet reordering. Here, multiple critical sections are needed to maintain synchronization. Consequently, it makes implementation complex and deteriorates performance.

The design of a load-balanced packet distribution solution needs to overcome these limitations along with meeting RAN data-plane protocol functional requirements.

III. RELATED WORK

A. Related works on packet flow distribution method

Multiple works are done for supporting load balancing in hash-based flow distribution methods. In [5], Shi et al. consider migrating only the flows which have high data rates. In [6], Dittman et al. propose load balancing by migrating flows

based on flow queue size. In [7], Iqbal et al. optimize per-flow statistics to identify aggressive flows and need additional hardware to identify aggressive flows. These solutions have a disadvantage when single flow throughput itself is above the maximum processing capacity of a core. They also have issues related to packet reordering when flows are migrated from one core to another. Since multiple transport flows can be mapped to a single flow at RAN, there is the possibility that traffic can be varied resulting in frequent flow migrations across cores. In [8], Guo et al. present batch scheduling for packets to minimize out-of-order delivery and load-balancing. In [9], Shi et al. propose hardware based ordering to avoid packet reordering. These solutions do not support per-flow synchronization requirements.

Our proposed solution guarantees multi-core packet distribution by maintaining per-flow synchronization, ordered packet delivery, and load balancing to increase per-flow peak throughput by leveraging all cores.

B. Related works on power management

Multiple works are done for reducing the power consumption of multi-core processors using DVFS methods. Kuang et al. [10] use DVFS for scheduling pipelined networking applications. This scheme allocates frequencies to the pipeline stages statically. Basireddy et al. [11] use DVFS, which allocates core frequency based on various parameters used in the calculation of core workload. These schemes do not consider balanced core workload requirement for effective core utilization in reduction of power consumption. Kokku et al. [12] use variation in traffic to schedule packet processing tasks and use set of processors to perform the same type of packet processing. It has the disadvantage of not maintaining per-flow synchronization and has packet reordering issues. Luo et al. [13] consider DVFS techniques based on the idle time of the processors in multi-core systems. These schemes are not aimed at dynamic traffic variations and do not explore load balancing requirements during different data flow packet processing. Iqbal et al. [14] propose DVFS techniques along with dynamically altering the number of active cores. It predicts traffic considering similar packet processing on the group of cores and accordingly changing the frequency of each core. It does not consider multiple packet type processing requirements. Linux ondemand and conservative governors [15] are reactive mechanisms where the decision is done based on instantaneous measured values. Frequent frequency oscillations are observed in ondemand governor. These schemes are reactive to instantaneous traffic fluctuations.

The Proposed Adaptive Power Management (APM) is different from earlier schemes as it pro-actively adapts to varying traffic conditions using global DVFS. It also identifies the importance of load-balanced system for better performance. It is a platform-independent solution. It does not operate specific to applications or packet types and preserves ordered packet processing and reduces the power consumption of the system without compromising data plane protocol functional requirements.

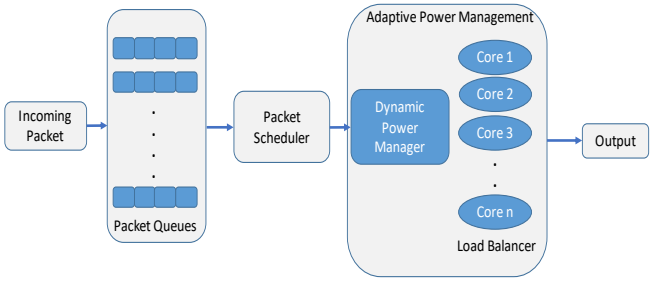


Fig. 3. Software architecture of Adaptive Power Management

IV. ADAPTIVE POWER MANAGEMENT (APM)

A. Software Architecture Overview

As shown in Fig.3 Adaptive Power Management (APM) solution consists of two modules, namely, load balancer and dynamic power manager. The load balancer module as the name indicates does load balancing across processor cores with efficient packet distribution and processing considering per-flow synchronization, packet ordering, and throughput requirements. Dynamic power manager monitors each processor's core load and adjusts frequency based on future core load prediction.

B. Load balancer - Multi-core packet distribution method

The processing of each flow is divided into three phases. I. Parallel Part – 1 or initial validation, II. sequential flow, III. Parallel Part – 2 or final transmission or post-sequential section. Phase 1 and Phase 2 can be done in parallel without any critical sections. Parallel processing increases the maximum per-flow throughput due to split and distribution in packet processing across cores and also reduces latency. Fig. 4 shows packet-processing steps concerning $flow_i$. The data packets are received from PKTIO queues, which are distributed to different cores based on corresponding priority. All packets belonging to a flow enter the following 3 phases.

Phase1: Parallel Part-1 - Initial validation of flow identifier and other validations can be done in any of the available cores.

Phase2: It requires the usage of one TryLock $TryLock_i$, custom-defined queue $queue_i$ which is locked using TicketLock $TicketLock_i$ for each flow $flow_i$. All the critical sections like enqueue and dequeue operations on the queue are performed after acquiring TicketLock. TicketLock is used to maintaining egress packet ordering same as ingress packet ordering.

Phase3: Parallel Part-2 - At the end of phase2, the packet is again given back to any available core, which can complete the phase3 in a parallel manner, which is generally transmitting the data by a device driver in the case of network systems.

C. Dynamic Power Manager

The main goal of the dynamic power manager is to predict the future core load and adjust processor core frequency accordingly. In a Load-balanced system, similar core utilization is maintained across all the cores. Global frequency change does not impart imbalance to individual flows as all core

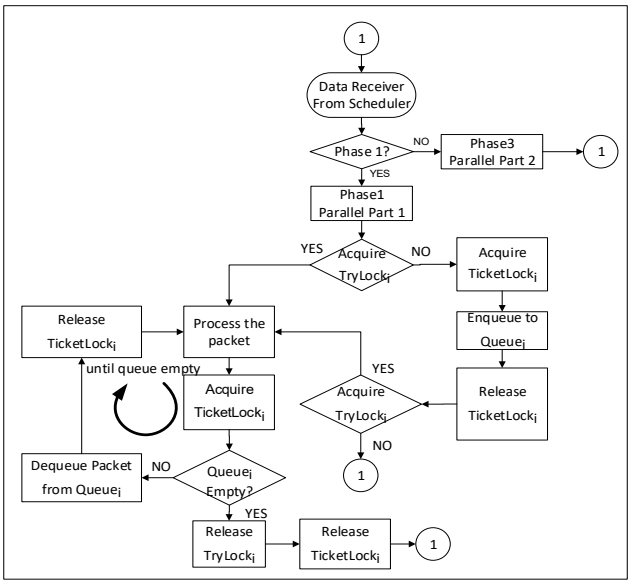


Fig. 4. Operational Flow diagram Load balancer: Multi-core packet distribution method

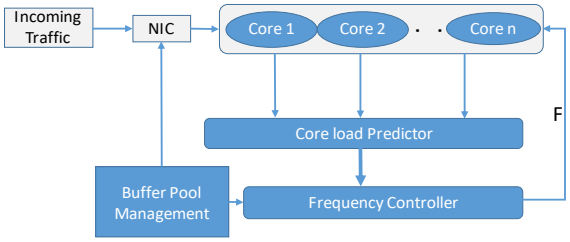


Fig. 5. Dynamic Power Manager

capacities are equally varied. RAN data plane systems, need to process a wide variety of packet types such as downlink, uplink, IPC (inter-process communication), timer events, interlayer information packets that needs different processing mechanism with varying priorities. Each packet type may take different processing cycles. The proposed solution can be applied to the system where packets of different types are processed together. As shown in Fig.5 Dynamic Power Manager monitors individual core load, computes average core load in every sampling interval and predicts future core load. Appropriate frequency is decided based on decision logic and applied using global DVFS.

1) Core load prediction: In data plane systems, processor cores are always active in busy while loop polling for the packet. Core load is calculated from busy and idle cycles. Busy cycles are processor core cycles when it does packet processing and idle cycles are processor core cycles when the processor keeps polling for data from the scheduler. Core load is a percentage value.

$$coreload = \frac{busycycles}{busycycles + idlecycles} \times 100 \tag{1}$$

Let CL_i be the core load of ith core and N be the total number of cores. The average core load is calculated as below.

$$CL_{avg} = \sum_{1}^{N} \frac{CL_i}{N} \tag{2}$$

Let F = F_{min}(F1), F2, F3... F_{max} are CPU operating frequency scaling levels and F_c be current frequency. Let CL_{highth} and CL_{lowth} be high and low core load thresholds.

Based on the study of traffic traces, Iqbal et al. [16] have proved that Double Exponential Smoothing (DES) predictor is a low complexity predictor with good accuracy when compared to complex predictors like Artificial Neural Network (ANN) or Wavelet transform-based predictors. The proposed solution uses DES to predict the future core load to make frequency decisions. The equation for DES-based prediction for time series $X(t)$ is as below.

$$X(t+1) = S_t + b_t \tag{3}$$

$$S_t = \alpha X(t) + (1-\alpha)(S_{t-1} + b_{t-1}) \tag{4}$$

$$b_t = \gamma(S_t - S_{t-1}) + (1-\gamma)b_{t-1} \tag{5}$$

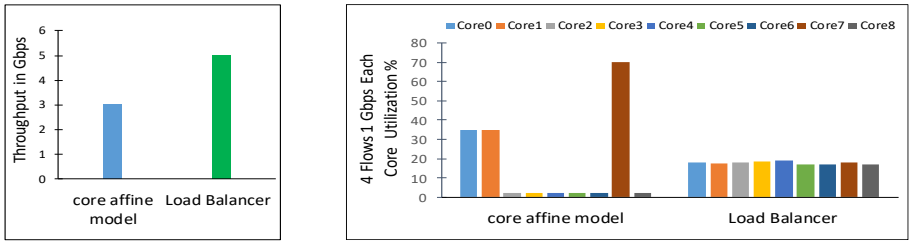
Where S_t and b_t are smoothened values of time series value X(t) and trend respectively. α and γ are smoothening factors. Their values are learned during the predictor training phase. S_t and b_t are added to get the predicted value. CL_{avg} is provided as input to the DES predictor to get predicted core load CL_{pred}.

2) Deciding frequency level: Pseudo-code for frequency selection is shown in Algorithm 1. The predicted core load for each frequency level is extrapolated from CL_{pred} for comparison with thresholds. A mapping table of core loads to different frequency values is maintained. For example, core load of 60% at 1.2Ghz frequency is 55% at 1.3Ghz frequency.

Algorithm 1 Deciding frequency level

```
for (every sampling interval) do
    if (availableBufferCount < minBufferThreshold) then
        NewProcFreq ← F_max
    else if (CL_pred > CL_highth) then
        NewProcFreq ← FindFreqUp(CL_pred)
    else if (CL_pred < CL_highth) then
        NewProcFreq ← FindFreqDown(CL_pred)
    end if
end for
```

When the predicted core load is greater than the CL_{highth}, frequency is increased to a suitable higher frequency. $FindFreqUp()$ gives minimum frequency above current frequency F_c for which predicted core load is within CL_{lowth} and CL_{highth}. When the predicted core load is less than the CL_{lowth}, frequency is reduced. $FindFreqDown()$ gives minimum frequency below current frequency F_c for which predicted core load is within CL_{lowth} and CL_{highth}. Frequency is not changed when the predicted core load is within low and high thresholds. When frequency change is done, the



(a) (b)

Fig. 6. Static Core-affine Model vs Load Balancer: Multi-core Distribution Model (a) Single flow maximum throughput comparison (b) Load balancing comparison

history of the predicted core load is adjusted to the present frequency level for future DES measurements.

3) Handling sharp traffic spikes and abnormal scenarios: When data rate increases suddenly due to heavy incoming traffic or core load predictor underpredicts, and the system is operating at a lower frequency, available buffer count is used for adjusting the frequency. During such scenarios of lower operating frequency, the available buffer count drops drastically due to packet buffers get accumulated in queues as the system is not able to process incoming packets at such a high rate. If the available buffer count becomes less than minimum buffer threshold, frequency is quickly scaled up to maximum frequency F_{max}. This helps in adjusting to dynamic variation of traffic and prevents packet drops.

4) Threshold selection: APM compares the predicated core load and available buffer count in the buffer pool with pre-defined thresholds. The core load threshold is set to avoid frequent frequency changes with fluctuating traffic. In this study, low and high thresholds are configured as 40 and 80 to make cores operate at a frequency level where core load is nearer and above 50%. The high threshold is configured as 80 to avoid reaching 100% core load due to fluctuation in traffic. Minimum buffer threshold is configured considering the time required for changing the frequency and sampling interval. Buffer space should be enough such that no packet drop happens before the frequency is set to maximum on detection of buffer depletion. Considering sampling interval of 500μs, frequency change time of 1ms, maximum system processing requirement of 2000K Packets Per Second for our experiment, the number of buffers required per ms is 2000. In this study, the buffer threshold is configured as 3000 considering sampling interval and frequency scaling time together.

V. SIMULATION AND EXPERIMENTAL RESULTS

Solutions are evaluated on a multi-core processor CN96XX Octeon processor testbed with 9 cores dedicated for RLC application with available scaling frequencies of 1.2Ghz, 1.3Ghz, 1.4Ghz, 1.5Ghz, 1.6Ghz, 1.7Ghz, and 1.8Ghz. The solution is implemented in Radio Link Control (RLC) [17] layer of Distributed Unit (DU). RLC processes packets from PDCP and delivers them to MAC Scheduler. Radio Link Control is responsible for data segmentation and reliable data delivery on the air interface using Automatic Repeat Request (ARQ).

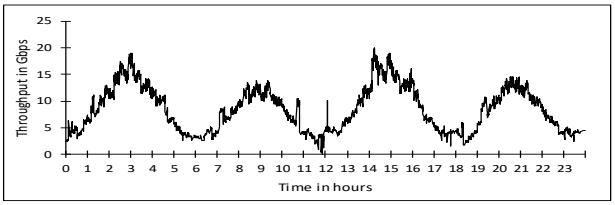


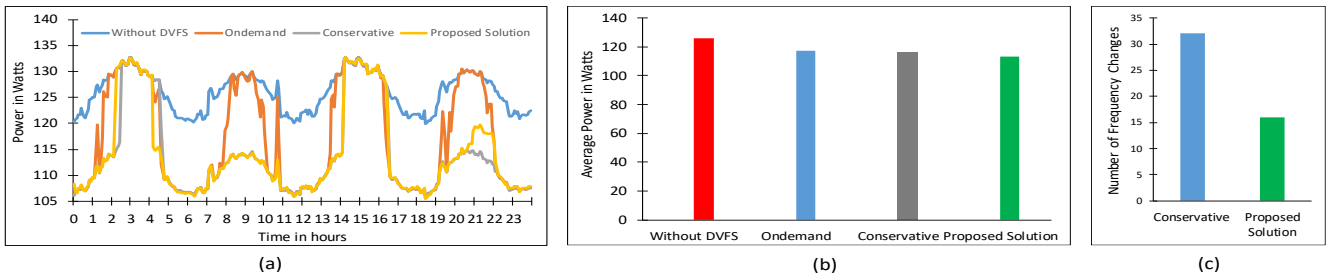
Fig. 7. 24-hour long-run real network traffic pattern for comparison

Data of each user is handled as a single flow with the requirement to support multiple users. Packet receiver and processing applications are developed with Open Data Plane (ODP) [18] API. ODP is an API specification that allows many implementations to provide platform independence, automatic hardware acceleration, and CPU scaling to high-performance networking applications.

1) Performance of proposed load balancer multi-core packet distribution model: The static core-affine model assigns packets from a flow to the same core during the lifetime of the flow. The core is assigned based on the hash-based function. From Fig. 6(a), we can see that maximum throughput of a flow in proposed multi-core packet distribution model is approximately 65% more compared to the static core-affine model. Fig. 6(b) shows the proposed multi-core packet distribution method has better load balancing compared to the static hash-based core-affine model when tested with a similar traffic pattern of 4 Flows with 1 Gbps each.

2) Traffic pattern for performance comparison: Fig. 7 shows real traffic traces collected over 24 hours from commercially deployed DU in RAN, where multiple users are connected to gNB. The same traffic pattern is used as traffic input to the testbed for comparing the APM solution with ondemand and conservative governor power management schemes.

3) Comparison between various power management schemes: Ondemand governor is an aggressive reactive governor which gradually settles to the lowest frequency at lower loads and highest frequency when core load percentage crosses the threshold. For verification of ondemand governor, 80 is configured as the upper core load threshold. Conservative governor tries to maintain core load between lower and upper threshold and gradually scales up and down. For verification of conservative governor and Adaptive Power Management (APM) scheme, 40 and 80 are configured as lower and upper core load thresholds respectively. The Adaptive Power



(a) (b) (c)

Fig. 8. Comparison between Without DVFS, Ondemand, Conservative and Proposed Solution (APM) (a) Power variation (b) Average power consumption (c) Number of frequency changes.

Management (APM) scheme has a better performance compared to ondemand and conservative governors on validating with the same traffic pattern for 24 hours.

Fig. 8(a) shows the power and frequency variation comparison between without DVFS scheme, ondemand, conservative, and APM. The APM operates at lower frequencies and lower power consumption compared to other schemes. Fig. 8(b) shows the average power comparison between various power management schemes. The proposed Adaptive Power Management (APM) solution has 10% reduced power consumption compared to without DVFS systems. We can expect more power savings during longer low throughput scenarios. The proposed solution also has almost 5% reduced power consumption compared to ondemand and conservative schemes. From Fig. 8(c), we can see that APM also has a lower number of frequency transitions compared to the conservative method. The aggressive nature ondemand governor results in a very high number of frequency changes.

VI. CONCLUSION

We have proposed a global DVFS based Adaptive Power Management (APM) solution to exploit traffic fluctuations in RAN data plane systems. This solution can recognize optimum frequency of processor cores for sustaining input traffic by predicting processor core load. Our solution adapts to sharp traffic spikes by monitoring available buffer status. Traditional power management schemes result in higher power consumption as they are reactive and unaware of load distribution across cores. Our solution brings effective processor core utilization by a load-balanced multi-core packet distribution method. APM solution is not specific to any application or packet types and can be applied where multiple packet type processing is required. The solution can achieve 60% gains in peak throughput of a flow. Experiments on real network traces show that the proposed solution reduces the processor power consumption by 10% and has at least 5% more power saving compared to other reactive schemes. Operational expenses can be significantly reduced considering the volume of network device deployments.

REFERENCES

[1] C. Bianco, F. Cucchietti, and G. Griffa, "Energy consumption trends in the next generation access network—a telco perspective," in *INTELEC 07-29th International Telecommunications Energy Conference*. IEEE, 2007, pp. 737–742.

[2] J. M. Rabaey and M. Pedram, *Low power design methodologies*. Springer Science & Business Media, 2012, vol. 336.

[3] A. Pathania, S. Pagani, M. Shafique, and J. Henkel, "Power management for mobile games on asymmetric multi-cores," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2015, pp. 243–248.

[4] L. Guo and I. Matta, "The war between mice and elephants," in *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*. IEEE, 2001, pp. 180–188.

[5] W. Shi, M. H. MacGregor, and P. Gburzynski, "Effects of a hash-based scheduler on cache performance in a parallel forwarding system," in *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2003), Orlando, FL, USA*, 2003, pp. 130–138.

[6] G. Dittmann and A. Herkersdorf, "Network processor load balancing for high-speed links," in *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, vol. 735. Citeseer, 2002.

[7] M. F. Iqbal, J. Holt, J. H. Ryoo, L. K. John, and G. De Veciance, "Flow migration on multicore network processors: Load balancing while minimizing packet reordering," in *2013 42nd International Conference on Parallel Processing*. IEEE, 2013, pp. 150–159.

[8] J. Guo, J. Yao, and L. Bhuyan, "An efficient packet scheduling algorithm in network processors," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2. IEEE, 2005, pp. 807–818.

[9] L. Shi, Y. Zhang, J. Yu, B. Xu, B. Liu, and J. Li, "On the extreme parallelism inside next-generation network processors," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 1379–1387.

[10] J. Kuang and L. Bhuyan, "Optimizing throughput and latency under given power budget for network packet processing," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[11] K. R. Basireddy, E. W. Wachter, B. M. Al-Hashimi, and G. Merrett, "Workload-aware runtime energy management for hpc systems," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2018, pp. 292–299.

[12] R. Kokku, T. L. Riché, A. Kunze, J. Mudigonda, J. Jason, and H. M. Vin, "A case for run-time adaptation in packet processing systems," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 107–112, 2004.

[13] Y. Luo, J. Yang, L. N. Bhuyan, and L. Zhao, "Nepsim: A network processor simulator with a power evaluation framework," *IEEE Micro*, vol. 24, no. 5, pp. 34–44, 2004.

[14] M. F. Iqbal and L. K. John, "Efficient traffic aware power management in multicore communications processors," in *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, 2012, pp. 123–134.

[15] "The cpufreq subsystem," https://developer.ibm.com/tutorials/l-cpufreq-1.

[16] M. F. Iqbal and L. K. John, "Power and performance analysis of network traffic prediction techniques," in *2012 IEEE International Symposium on Performance Analysis of Systems & Software*. IEEE, 2012, pp. 112–113.

[17] "3GPP TS 38.322 5G NR Radio Link Control (RLC) protocol specification"," 2018.

[18] "What is odp?, OpenDataPlane," https://opendataplane.org.