Performance and Power Consumption Modeling for Green COTS Software Router

Raffaele Bolla, Roberto Bruschi, Andrea Ranieri
Department of Communication, Computer and Systems Science (DIST)
University of Genoa
Via all'opera Pia 13, 16145, Genoa, Italy
{raffaele.bolla, roberto.bruschi, andrea.ranieri}@unige.it

***Abstract*—We consider a new generation of COTS Software Routers (SRs), able to effectively exploit multi-Core/CPU HW platforms. Our main objective is to analyze, to evaluate and to model the impact of power saving mechanisms, generally included in today's COTS processors, on the SR behavior and networking performance. To this purpose, we tried to understand and to separately characterize the roles of both HW and SW layers through a large set of internal and external experimental measures, obtained with a heterogeneous set of HW platforms and SR setups. Moreover, starting from this detailed measure analysis, we propose a simple model, able to represent the SR performance with a high accuracy level in terms of packet throughput and related power consumption. The proposed model can be effectively applied inside "green optimization" mechanisms to minimize power consumption, while maintaining a certain SR performance target.**

Keywords-sowtware routers, green networking, performance measurement.

I. INTRODUCTION

In the last years, the research field of "green" and energy-efficiency networking infrastructures has gained a great interest from both service/network providers and equipment manufacturers.

Despite a more widespread sensibility in ecological issues, such interest springs from heavy and critical economical needs, since both energy cost and network electrical requirements show a continuous growing with an alarming trend over the past years. For example, as shown in [8] and in [9], energy consumption of Telecom Italia network in 2006 was more than 2TWh (about the 1% of the total Italian energy demand), increasing by a 7.95% with respect to the 2005, and by a 12.08% to the 2004. Similar trends can be certainly generalized to a large part of the other telecoms and service providers, since they essentially depend on data traffic volume increase and new services being offered.

In greater detail, to support new generation network infrastructures and related services for a rapidly increasing customer population, telecoms and service providers need for a larger and larger number of network equipments, with sophisticated architectures able to scalably perform more and more complex operations. In such an environment, the challenge for network equipment manufacturers is, nowadays, to design and to develop new architectures able to scale their performance and functionalities preserving a high level of energy efficiency. Despite of some interesting scientific contributions (e.g., [1], [2], [3], [4], [5], [6] and [7] among the others), green networking performance and optimization remain an open and very interesting issue.

Starting from these considerations, we focus on promising new generation architectures for Linux Software Routers (SR) [10][11], based on open source software and on multi-core COTS hardware.

In detail, our work tries to investigate the impact on the Software Routers' performance of power management mechanisms, which are generally available in a large set of general purpose CPUs through the Advanced Configuration and Power Interface (ACPI) interface. Our main objective is to measure, to evaluate and to model the tradeoff between power consumption and performance of SR basic functionalities (e.g., packet forwarding), in order to assess the feasibility of "green optimizations" for COTS SR architecture. With this main idea, we performed several benchmarking sessions with a heterogeneous set of HW platforms to carefully characterize and measure the impact of power management mechanisms on SR performance. These measures gave us the possibility both to understand the correlation between SR internal dynamics and its external/power-related performance, and to derive a simple model to estimate SR behavior according to different setups and traffic offered loads. The proposed model could have a key role, thanks to its accuracy and low computational complexity, in developing innovative "green" optimization mechanisms for multi-Core SR.

The paper is organized as in the following. Sections II and III introduce the new generation SR architectures, specifically optimized for multi-core HW systems, and the SR standard support and features for power management, respectively. In Section IV, we propose a large set of benchmarking sessions with the aim of investigating how power management mechanism impacts on SR performance. A simple empirical model, able to effectively estimate SR behavior in terms of throughput and power consumption, is proposed in Section V. Finally, conclusions and future work objectives are in SectionVI.

II. NEW GENERATION ARCHITECTURES FOR LINUX SR

With the aim of evaluating capabilities and performance levels provided by a SR architecture, we decided to use a set of HW and SW enhancements that allow us to effectively and scalably exploit multi-CPU/core PC systems.

In detail, these HW and SW enhancements, already discussed and evaluated in [10] and in [11], let us go beyond typical performance issues, namely data accessing serialization (i.e., LLTX lock contention) and CPU/core cache coherence, which generally affect SR data-plane operations.

The key element of such SR architectural enhancements is the support for "multi-queuing" network adapters, like the Intel PRO 1000 adapters (with MAC chip 82571 and higher), which support multiple Tx- and Rx Rings and multiple HW interrupts per network interface [12].

Using such innovative network board feature, we deployed a new SW architecture/configuration that allows to optimize SR data-plane performance by reducing memory sharing among CPU/cores and avoiding the LLTX lock contention. In particular, the optimized SR architecture provides three main SW enhancement guidelines:

- to entirely bind all operations carried out in forwarding a packet to a single CPU (minimizing cache invalidations),
- to separate working memory regions of different CPU/cores,
- to equally distribute the computational load among all the processors/cores activated in the system.

Further details of such optimized SR architecture can be found in [11] and in [10].

III. THE POWER MANAGEMENT IN LINUX SR

Power management is a key feature in today's processors across all market segments. While the ACPI provides a well-known standardized interface between the hardware and the software layers, processors use different internal techniques to reduce their energy usage by exploiting the fact that systems do not need to run at peak performance all the time. This is usually accomplished by tuning the frequency and/or the voltage of processors, or by throttling the CPU clock (i.e., the clock signal is gated or disabled for some number of cycles at regular intervals).

Decreasing the operating frequency and the voltage of a processor, or throttling its clock, obviously allows the reduction of the power consumption and of heat dissipation at the price of slower performance.

The ACPI standard introduces two main different power saving mechanisms, namely performance and power states (*P*-states and *C* States) respectively, which can be individually employed and tuned for each core in the largest part of today's processors. Regarding the *C* states, the C_0 power state is an active power state where the CPU executes instructions, while the C_1 through C_n power states are processor sleeping or idle states, where the processor consumes less power and dissipates less heat.

While in the C_0 state, ACPI allows the performance of the Core to be tuned through *P* state transitions. *P* states allows to modify the operating energy point of a processor/core by altering the working frequency and/or voltage, or throttling its clock. Thus, using *P* states, a core can consume different amounts of power while providing different performance at the C_0 (running) state. At a given *P* state, Core can transit to higher *C* states in idle conditions. In general, higher the index of *P* and *C* states is, the lesser will be power consumed, and heat dissipated.

Each processor model generally provides a fixed number of *C* and *P* states for all included Cores[15].

IV. PERFORMANCE SCALING IMPACT ON DATA PLANE

Our main objective in this Section is to evaluate how CPU Performance Scaling Mechanisms (PSMs) impact on SR functionalities and power consumption. With this respect, we have to focus both on HW support and capabilities of PSMs included in different CPU versions, and on how these HW features can affect performance and behavior of main SW functionalities of a SR.

Power consumption is clearly dependent both on the efficiency of *P* and *C* states available in the CPU/Core, and on its computational load. In fact, the latter directly influences the average time spent in the "active state" C_0, or in idle states (C_i, with $i>1$). In detail, we can express the average power consumption $\tilde{\Phi}$ of a CPU/core as in the following:

$$\tilde{\Phi} = p_{idle}\Phi_{idle} + (1 - p_{idle})\Phi_{active} \quad (1)$$

where p_{idle} is the probability that the CPU/core is in an idle state, Φ_{idle} is the average power consumption during idle states, and Φ_{active} the one during active states.

Both Φ_{idle} and Φ_{active} are respectively functions of the *C* and *P* states, available at the HW layer and activated by the ACPI SW drivers.

In particular, today's ACPI SW drivers set CPU power profile through simplified high-level APIs, which allow users to use a range of "equivalent CPU frequencies". Inside the ACPI driver, each equivalent frequency is mapped into a pre-selected set of feasible *C* and *P* states: as the selected value of equivalent frequency is higher, the index of feasible *P* states is larger and the *C* states' one lower. This obviously results in a set of CPU working states with different levels of performance and power saving. In particular, we can re-define Eq. 1 with respect to the selected equivalent frequency:

$$\tilde{\Phi}(f) = p_{idle}\Phi_{idle}(f) + (1 - p_{idle})\Phi_{active}(f), \quad f \in F \quad (2)$$

where *f* is the selected equivalent frequency, and *F* the set of all its possible values.

Note that, while Φ_{idle} and Φ_{active} are mainly fixed by the ACPI HW layer, p_{idle} depends both on the computational load of SW tasks and on the CPU/core computational capacity, which is also a function of the equivalent frequency. Starting from these consideration, we can express p_{idle} as in the following:

$$p_{idle}(f, l) = 1 - \frac{l}{C(f)} \quad (3)$$

where l is the SW computational load, and $C(f)$ is the CPU/core computational capacity at the equivalent frequency *f*.

Thus, to analyze PSM performance and its effect on SR architecture, we need to act and to focus at two different layers, by evaluating, in one hand, the ACPI HW capabilities included

of different CPU/Cores, and, in the other one, the computational complexity and SW dynamics of a SR.

A. The HW layer

Regarding the HW PSM capabilities, we decided to evaluate the performance provided by a heterogeneous set of PC platforms, based on different CPU models and ranging from high-end servers to common workstations and mobile architectures. In detail, for each selected HW platform, we performed two kinds of power consumption measures:

- Φ_{idle} - *Idle measures*: to evaluate the average power consumption when all the CPU/Cores in the system are idle, and do not perform any tasks.
- Φ_{active} - P_n *state measures*: to quantify the average power consumption when one or more CPUs/Cores are running (i.e., while they are performing SW operations for the 100% of their time) at the n-th *P* state.

While the first kind of measures (i.e., idle) are useful to evaluate the minimum power consumption for each selected PC platform, the Φ_{active} measures point out how much energy a CPU/core can waste when fully active at a certain ACPI P_n state. In this respect, these two measures can be thought as the upper and the lower bounds of SR performance, respectively, when deployed in a real network.

Moreover, to effectively supplement this last kind of tests, we also decided to consider an additional performance index in order to evaluate CPU/core's computational capacity $C(f)$ at the given frequency.

In detail, since data-plane functionalities certainly are the most CPU intensive and time consuming operations to be carried out in a SR, we decided to consider the maximum throughput, in terms of forwarded packets per second, as the performance index for evaluating CPU computational capacity.

Figures 1 and 2 show the Φ_{active} and the Φ_{idle} results obtained with a heterogeneous set of HW architectures, which includes high-end servers, workstations and mobile architectures. In detail, we used and extensively tested seven HW platforms with the following features:

- *HW platform 1 (X)*: dual CPU Intel XEON Dual Core 5050 "Dempsey", 3.0 GHz, 64 bit, Cache 2x2 MB, FSB 667 MHz, RAM DDR2-533 MHz ECC Fully Buffered, I/O bus PCIe 16x;
- *HW platform 2 (PD)*: Intel Pentium-D 925 Dual Core, 3.0 GHz, 64bit, Cache 2x2 MB, FSB 800 MHz, RAM DDR2-667 MHz Unbuffered, I/O bus PCIe 8x;
- *HW platform 3 (A)*: AMD Athlon 64 Dual Core X2 3800+, 2.0 GHz, 64bit, Cache 2x512 kB, FSB 2000 MHz, RAM DDR2-667MHz Unbuffered, I/O bus PCIe 16x;
- *HW platform 4 (PM)*: Intel Pentium-M 745 "Dothan", Single Core, 1.8 GHz, Cache 2 MB, RAM DDR-266 MHz Unbuffered, I/O bus PCIe 8x;
- *HW platform 5 (O)*: dual CPU AMD Opteron Dual Core 265, 1.8 GHz, 64 bit, Cache 2x1 MB, FSB 1000 MHz, RAM DDR-400 MHz ECC Registered, I/O bus PCIe 16x;
- *HW platform 6 (C2)*: Intel Core 2 Duo E2160, 1.8 GHz, Cache 2x1 MB, FSB 800 MHz, RAM DDR2-667 MHz Unbuffered, I/O bus PCIe 8x;
- *HW platform 7 (A2)*: AMD Athlon 64 3500+, 2.2 GHz, 64bit, Cache 512 kB, FSB 2000 MHz, RAM DDR-200 MHz Unbuffered, I/O bus PCIe 16x.

Figures 1 and 2 point out the gap in terms of power consumption and forwarding performance between the different tipologies of HW platforms. These differences obviously depend not only on the specific CPU model included in the HW platform, but also on other HW components, like for example the ones integrated in the PC mainboard (and server mainboards generally include a larger set of HW components and controller than the workstation ones). For a sake of simplicity, we drop this additional power consumption contribution in the following of this paper, since it does not affect the obtained results and conclusions.

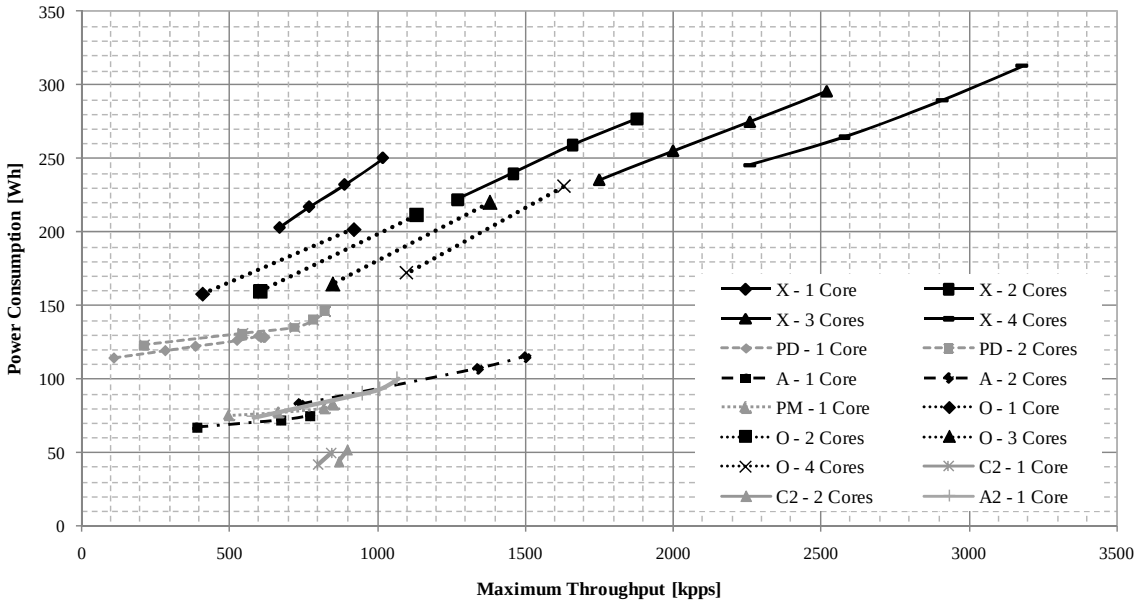

Figure 1. Power consumption and maximum throughput values for all the selected HW platforms. For each HW platform, the tests were performed by changing the CPU/Core *P* state, and, where possible, according to a variable number of active CPUs/Cores.

High-end server architectures (i.e., the X and the O ones) obviously show both heavy power consumption (more than 150 Wh), and high forwarding performance, especially when a large number of Cores is applied.

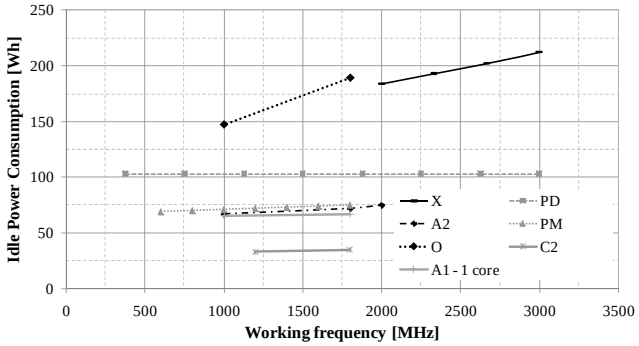

Figure 2. Idle power consumption for all the selected HW platforms.

On the contrary, workstation architectures (i.e. the PD, the A and the A2 ones) exhibit an intermediate performance level, but both power consumption and forwarding performance can heavily change according to the selected HW platform. For example, the PD platform is characterized by a much higher power consumption than the A and the A2 ones, while the A platform, thanks to its high-speed internal busses, shows remarkable values of packet throughput (about 1.5 Mpkt/s). In particular, the forwarding performance of the PD and the A2 HW platforms result to be limited by the bandwidth and the efficiency of internal busses, which do not allow to exploit the full computational capacity of CPUs/Cores.

Similar remarks can be made also for the mobile oriented architectures (i.e., the PM and the C2 ones). In detail, we can note how the C2 HW platform provides a very low power consumption, which ranges between about 30 and 50 Wh.

Besides interesting information on different HW platforms' behavior, Figures 1 and 2 show that all the analyzed PSMs (both in AMD and Intel processors) allow to linearly scale the maximum forwarding rates with respect to power consumption. This linear behavior is maintained not only for different equivalent frequency values, when a single core is used for the SR data plane, but also for a rising number of CPUs/Cores.

B. The SW layer contribution

Besides the effectiveness of the PSM included in today's CPUs, we have also to take into account how the impact of SW functionalities and dynamics on the HW behavior, and especially on the p_{idle} parameter as suggested by Eq. 3.

With this aim, we performed several benchmarking sessions in order to analyze and to empirically evaluate the relationship between the PSMs and the SR SW layer.

As previously outlined, packet forwarding operations are almost the totality of computational load in a SR, since control plane functionalities (i.e., routing protocols, etc.) work at much slower time scales, and their computational weight is quite negligible. Therefore, to characterize SW layer behavior, we decided to use a set of performance indexes typically related to the SR data-plane.

Starting from these considerations, we analyzed and evaluated the correlation between power consumption and Linux data-plane performance, in terms of throughput and packet forwarding latency.

Among all the performed test sessions, Figures 3, 4 and 5 report the results related to the Xeon-based platform. Similar results were obtained for all the other analyzed HW architecture. In detail, these tests were carried out by using only one single Xeon Core to forward a traffic flow, with a constant bit rate and composed by 64B sized IP packets.

Figures 3 and 4 show that data-plane performance increase according to the Core working frequency: higher frequency values assure larger maximum throughput values and lower latencies. This obviously depends on the increase of the Core computational capacity $C(f)$, which allows to elaborate a larger number packet headers with smaller processing times.

In detail, the maximum latency times, reported in Figure 4 and obtained in identical router configurations (e.g., same buffer sizing) except for the core frequency, highlight how the processing times the packet headers rise according to the Core frequency (and then to the $C(f)$ parameter) decrease.

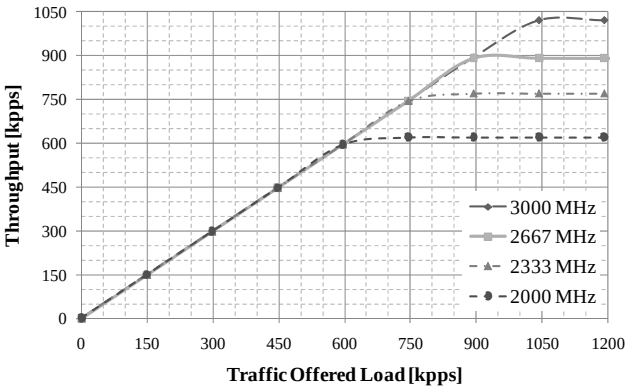

Figure 3. Packet throughput for the Xeon based architecture, in the presence of a single forwarding Core, and according to different Core frequencies and traffic offered loads.

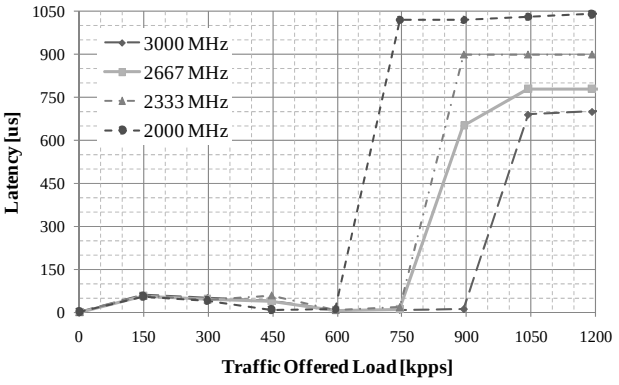

Figure 4. Packet latencies for the Xeon based platform, in the presence of a single forwarding Core, and according to different Core frequencies and traffic offered loads.

However, the forwarding performance gain in terms of maximum throughput and latency, due to the CPU frequency increase, is not exactly linear. In fact, by examining in the detail Figures 3 and 4, we can note how the performance gain, in terms of throughput and latency, slightly decreases as the core frequency raises. This effect can be observed in large number of PC HW architectures, and it is mainly due to a tiny performance decay of other HW elements (i.e., RAM, busses, etc.) at high-speed.

Figure 5 reports the power consumption of the Xeon-based architecture according to different values of Core frequency, and of throughput. Note that the throughput values, for each test session, were obtained with respect to a rising traffic offered load.

Thus, as clearly shown in Figure 5, the correlation between power consumption and packet throughput is not linear, all the same it highlights a similar profile according different frequency values.

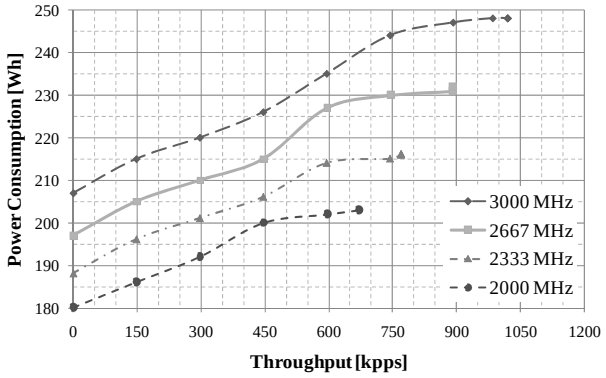

Figure 5. Power consumption and related thoghput values, for the Xeon based platform, according to different Core frequencies and traffic offered load.

The cause of such profile similarity, kept both for different frequency values and for HW platforms, can certainly be found in the architecture and dynamics of the Linux networking stack. In fact, as described in [13], the entire framework and dynamics of Linux packet processing architecture are guided by the NAPI mechanism, which allows to scalably handle packet processing operations.

In short, the NAPI mechanism handles network interface requests with a interrupt (IRQ) moderation mechanism, which allows to adaptively switch from a classical interrupt management of the network interfaces to a polling one, when the traffic offered load rises. In detail, the NAPI provides that packets' processing operations are performed during priority kernel tasks, namely SoftIRQs. A maximum number of packets can be elaborated during each SoftIRQ. When a SoftIRQ ends, if there are any more packets to be processed, a new SoftIRQ will be scheduled. On the contrary, when no SoftIRQs are scheduled or active, and new packets are received, SoftIRQs are triggered by network board's IRQ.

Figure 6 shows a session of internal measures performed with the Xeon based platform at the 3 GHz frequency, obtained using kernel profiling tools [14] and SW counters. These internal measures point out how the NAPI mechanism lowers the number of IRQs and SoftIRQs per forwarded packet, as the traffic offered load rises. In fact, a higher traffic offered load can obviously produce the increase of both the average number of packets processed in a same SoftIRQ, and the SoftIRQ re-scheduling probability. Such decrease of IRQs and SoftIRQs per forwarded packet, obviously produces an increase of SW efficiency, since it allows to avoid a certain number of "overhead" SW operations. In confirmation of this, both the CPU idle time and the IRQ number get near to zero at about 700 kpps, before the maximum throughput is reached. Then, the throughput rise between 700 and 1040 kpps can be attributed to the SW efficiency increase.

Moreover, we can note how the zeroing of IRQs and CPU idle time at 700 kpps agrees with the gradient change of the power consumption profile in Figure 5. Therefore, while we have an almost linear increase of power consumption up to the IRQ zeroing, for higher throughput values the power consumption achieves more or less its maximum value. Despite the complexity of the Linux architecture and dynamics, this is a clear indication of the relationship between the power consumption profiles in Figure 5 and the SW internal dynamics.

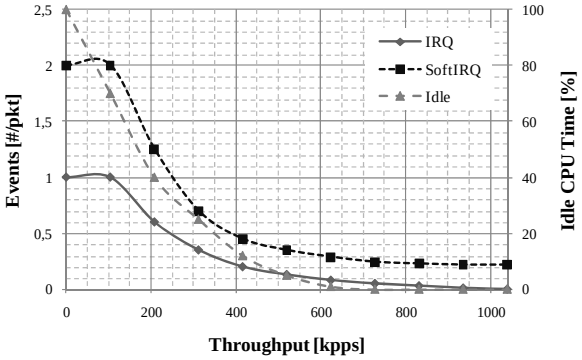

Figure 6. Linux kernel internal measures obtained with SW counters, and the Oprofile SW suite. The IRQ profile is refered to the second y-axis.

V. AN EMPIRICAL MODEL FOR SR POWER CONSUMPTION

Starting from the analysis and the benchmarking results in the previous Section, we propose a simple empirical model for describing the SR behavior in terms of throughput and power consumption. The main aim of the proposed model is to represent the SR performance with respect to traffic offered load and SR setups, in terms of active Core number and working frequencies. Such a model obviously represents an indispensable tool to develop optimization mechanisms for SR power saving.

To derive such empirical model, we starts by considering the simplest scenario, with only a single "forwarding" Core, in the sub-Section V.A, and then we refine the model in the sub-Section V.B for the multi-Core scenario.

A. The single Core case

As already sketched, Figure 5 shows a clear dependency between the power consumption and the packet throughput, which is very similar for all the used Core frequencies. More in particular, these tests (and many other ones not reported in this paper) absolutely suggest that the frequency impacts on such

correlation in terms both of a scaling factor, and of a shifting one. Thus, with the aim of investigating such idea in detail, we decided to process all the measured samples, collected in the power consumption versus packet throughput tests (including the ones shown in Figure 5).

In particular, by shifting such curves of their minimum power consumption (the value when throughput is equal to 0), and by normalizing both the power consumption and the throughput values, we obtained the interesting results shown in Figure 6, which suggests a same form envelope for power consumption according to Core frequencies, thanks to the narrow superposition of normalized measured samples.

In order to represent this profile form, we decided to introduce the normalized form function $\chi(\cdot)$, and to derive it as a polynomial fitting function. In detail, we define $\chi(\cdot)$ as a 7th-order polynomial as in the following:

$$\chi(t) = \sum_{i=0}^{i=7} a_i t^i \quad (4)$$

The choice of the 7th-order simply depends on the fact that it is the minimum order polynomial, which guarantees an acceptable accuracy level.

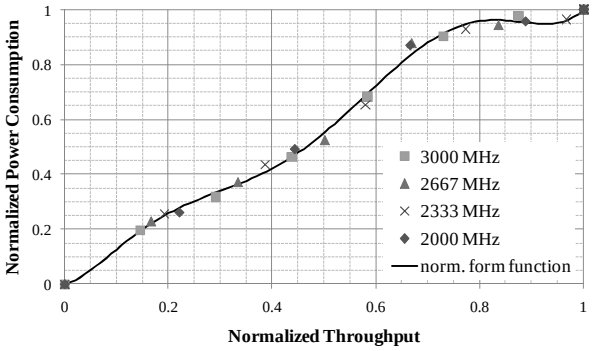

Figure 7. Normalized and shifted measure samples, obtained according to traffic offered loads and to different Core frequencies, and the normalized form function $\chi(\cdot)$.

Starting from the normalized and shifted values of collected measures for all the selected HW architecture, we calculated the a_i parameters through the method of least squares, and we obtained the following numerical values: a_0=0, a_1=0.470487, a_2 =15.0124, a_3 = -89.8629, a_4 =223.074, a_5 =-256.469, a_6=130.477, a_7=-21.6999.

Starting from the $\chi(\cdot)$ definition, for a single Core, working at the frequency $f \in F$ and forwarding r packets per second, we can express the power consumption in the following way:

$$\Phi(f, r) = \Phi_{idle}(f) + [\Phi_{active}(f) - \Phi_{idle}(f)] \cdot \chi\left(\frac{r}{r_{max}(f)}\right) \quad (5)$$

where $r_{max}(f)$ is the maximum throughput at the frequency *f*.

Regarding the *r* parameter, we decided by observing Figure 3to approximate the throughput r as the minimum value between the traffic offered load *l* and its maximum forwarding rate $r_{max}(f)$ (both in terms of packets per second):

$$r = min\{l, r_{max}(f)\} \quad (6)$$

Note that, in the case of variable throughput values during a $[0, T]$ time period, the mean power consumption can be simply expressed as the $\Phi(f, r)$ contribution, averaged on the $r(t)$ rate values in the same time window $t \in [0, T]$:

$$\tilde{\Phi} = \frac{1}{T}\int_{t=0}^{T} \Phi\big(f, r(t)\big)\mathrm{d}t \quad (7)$$

By replacing the $\tilde{\Phi}$ and the $\Phi\big(f, r(t)\big)$ parameters from Eqs. 2 and 5, we obtain the following relationship between the normalized form function and the p_{idle}:

$$p_{idle} = 1 - \frac{1}{T}\int_{t=0}^{T} \chi\left(\frac{r(t)}{r_{max}(f)}\right)\mathrm{d}t \quad (8)$$

From Eq. 8, the $\chi\left(\frac{r(t)}{r_{max}(f)}\right)$ function clearly represents the probability that the Core is active at a certain working frequency *f* and with a instantaneous throughput equal to *r(t)*.

In confirmation of what previously indicated in Section IV, the p_{idle} parameter mainly depends both on SW efficiency and feature, taken into account thanks to the $\chi(\cdot)$ normalized form function, and on the Core computational capacity, expressed as $r_{max}(f)$, at the working frequency *f*.

Regarding the Φ_{idle}, the Φ_{active} and the r_{max} parameters, they obviously depend on the adopted HW architecture. In detail, Φ_{idle} and Φ_{active} and the set of admissible working frequencies *F* are specifications/features of CPU/Cores, usually published by chip manufactures.

On the contrary, the $r_{max}(f)$ parameter depends not only on the features of CPU/Core, but also on the ones of other HW elements, such as I/O busses and RAM.

In detail, since the complexity of SR architecture, the $r_{max}(f)$ values cannot be a priori fixed or estimated, but they must be obtained by experimentally measuring the performance of each selected HW architecture. An example of such experimental measures can be found in Figures 1 and 2, which clearly show a linear inter-dependency between the $r_{max}(f)$ and the $\Phi_{active}(f)$ parameters according to different frequency values and HW architectures.

B. The multi-Core case

As explained in Section II, new generation SR architectures exploit multi-Core HW platform in a very effective way, since they allow to parallelize the forwarding process among the active Cores as much as possible. However, the results in Figure 1 clearly highlight that, increasing the number of active forwarding Cores, the performance gain is not full for all the selected HW platforms. In other words, for certain HW platforms (e.g., the Opteron based one), when we increase the number of forwarding Cores from 1 to 2, the maximum throughput does not redouble, but it achieves a value equal to about the 150% of the single Core throughput. On the contrary, other HW platforms (e.g., the Xeon based one) show a performance gain near to the 200%.

Then, this variable gain depends on the specific HW platform, and it is probably caused by performance bottlenecks in other HW components of the SR platform (i.e., I/O busses, RAM, Front Side bus, etc.).

To represent this HW-dependent feature, and to take into account how the overall performance of the SR increase according to a certain number active Cores, we introduce the constant γ. The γ parameter must be measured and evaluated for each used HW platform.

Extending Eq. 5 to the multi-CPUs/Cores' case, we can express the power consumption as in the following:

$$\Phi(f_c, r_c | \forall c \in C) = \sum_{\forall c \in C} \Phi_{idle}(f_c) + \sum_{\forall c \in C} [\Phi_{active}(f_c) + -\Phi_{idle}(f_c)] \cdot \chi\left(\frac{r_c}{\frac{\gamma}{|C|} r_{max}(f_c)}\right) \quad (9)$$

where C is the set of Cores in the SR, and $|C|$ its cardinality; the f_c and the r_c parameters are the working frequency, and the packet throughput of the Core $c \in C$, respectively.

Regarding the SR overall throughput value r, we can simply extend Eq. 6 as in the following:

$$r = \sum_{c \in C} min\left\{l_c, \frac{\gamma}{|C|} r_{max}(f)\right\} \quad (10)$$

where l_c is the traffic offered load for the Core c.

C. A Simple Validation

With the aim of validating the proposed model, we perform further benchmarking sessions by using the HW platform introduced in the sub-Section IV.A. In detail, for each benchmarking session, we collected a set of power consumption measures according to different throughput values and Core number, and, then, we compared these measured values with the ones estimated by the proposed model.

Figure 8 shows the maximum model error, with respect to the measured values, in terms of estimated power consumption for each selected HW architecture. The obtained maximum errors underline that the proposed model allows to describe SR behavior with a high accuracy level.

VI. Conclusions

In this contribution, we focused on the "green" capabilities of new generation COTS SRs, able to effectively exploit multi-Core/CPU HW platforms. Our main objective was to analyze, to evaluate and to model the impact of power saving mechanisms, generally included in today's COTS processors, on the SR behavior and networking performance.

To this purpose, we understood and separately characterized the roles of both HW and SW layers through a large set of internal (i.e., Linux SW counters and kernel profiling) and external (i.e., throughput, latencies and power consumption) experimental measures, obtained with a heterogeneous set of HW platforms and SR setups. This detailed measure analysis gave us the chance to derive a simple model, able to represent the SR performance with a high accuracy level in terms of packet throughput and related power consumption. Regarding the future work, our objective is to extend this contribution by developing a "green optimization" mechanism, based on the model here proposed, in order to minimize power consumption while maintaining a certain SR performance target.

References

[1] Noguera, J.; Kennedy, I.O., "Power Reduction in Network Equipment Through Adaptive Partial Reconfiguration," *Proc. of the 2007 International Conference on Field Programmable Logic and Applications (FPL 2007)*, Aug. 2007, pp. 240-245.

[2] Paleologo, G.A.; Benini, L.; Bogliolo, A.; De Micheli, G., "Policy optimization for dynamic power management," *Proc. of Design Automation Conference*, Jun 1998, pp. 182-187.

[3] "Intel is Leading the Way in Designing Energy-Efficient Platforms", Intel White paper.

[4] Chabukswar, R.; "Maximizing Performance and Energy-Efficiency on Intel@ Core™ Microarchitecture using Multi-Threading", Intel White paper.

[5] Miyoshi, A., Lefurgy, C., Van Hensbergen, E., Rajamony, R., and Rajkumar, R., "Critical power slope: understanding the runtime effects of frequency scaling," *Proc. of the 16th international Conference on Supercomputing (ICS'02)*, New York, NY, USA, June 2002.

[6] Herbert, S. and Marculescu, D., "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," *Proc. of the 2007 international Symposium on Low Power Electronics and Design (ISLPED '07)*, Portland, OR, USA, Aug. 2007, pp. 38-43.

[7] Gross, P.; Godrich, K. L., "Using Energy-Efficient Technologies to Lower Operational Costs in High-Density Environment," *Proc. of the 28th International Telecommunications Energy Conference (INTELEC '06)*, Providence, RI, USA, Sept. 2006, pp.1-4.

[8] Bianco, C.; Cucchietti, F.; Griffa, G., "Energy consumption trends in the next generation access network — a telco perspective," *Proc of the 29th International Telecommunications Energy Conference, 2007. (INTELEC 2007)*, Rome, Italy, Sept. 2007, pp.737-742

[9] Telecom Italia Website, "The Environment", URL: http://www.telecomitalia.it/sostenibilita2006/English/B05.html

[10] Bolla, R.; Bruschi, R., "An effective forwarding architecture for SMP Linux routers," *Telecommunication Networking Workshop on QoS in Multiservice IP Networks, 2008. IT-NEWS 2008. 4th International* , Venice, Italy, Feb. 2008, pp.210-216.

[11] Bolla, R.; Bruschi, R., "PC-based Software Routers: High Performance and Application Service Support," *Proc. of the ACM Sigcomm Worshop on Programmable Routers for Extensible Services of Tomorrow* (PRESTO'08), Seattle, WA, USA, Aug. 2008, pp. .

[12] Yi, Z.; Waskiewicz, P.J., "Enabling Linux Network Support of Hardware Multiqueue Devices," *Proc. of the 2007 Linux Symposium*, Ottawa, Canada, June 2007, pp. 305-310.

[13] Bolla, R.; Bruschi, R., "Linux Software Router: Data Plane Optimization and Performance Evaluation," *Journal of Networks (JNW)*, vol. 2, no. 3, Academy Publisher, pp. 6-11, 2007.

[14] Oprofile, http://oprofile.sourceforge.net/.

[15] ACPI Specification, www.acpi.info/DOWNLOADS/ACPIspec30b.pdf

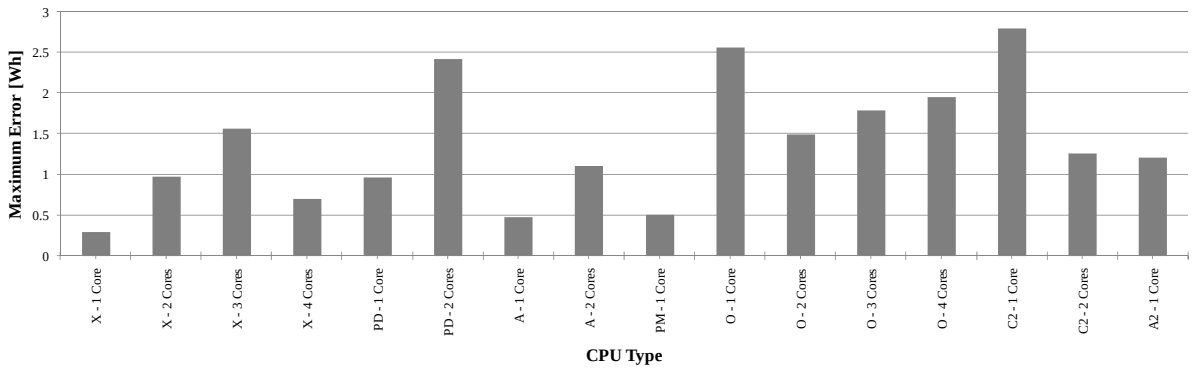

Figure 8. Maximum model error with respect to benchmarking measures and according to different HW platforms and Core number.