Comparing Dual-Core SMP/AMP Performance on a Telecom Architecture

Nico De Witte, Robbie Vincke, Sille Van Landschoot, Eric Steegmans and Jeroen Boydens

Abstract – **In the embedded world, symmetric multiprocessing architectures are currently most popular, however more embedded hardware platforms are being developed with asymmetric multiprocessor architectures. These may enable higher performance and provide cleaner separation of subsystems. Telecom applications are typically designed applying a planar architecture pattern. The goal of our experiments is to compare the performance and cross-plane influence in dual-core symmetric and asymmetric multiprocessing environments. Next to a pronounced performance difference, a cross-influence between the different planes has been verified.**

Keywords – **Multi-Core, Telecommunications, Planar Architecture Pattern, Symmetric Multiprocessing, Asymmetric Multiprocessing**

I. INTRODUCTION

While high performance computers and personal computers are already benefiting from multi-core processors, the transition from singlecore to multi-core processors is still in progress for embedded systems. Moore's Law [1] which states that every 18 months the number of transistors on a single chip doubles, is reaching its physical limits. On the other hand, Amdahl's Law [2], which shows that the performance of parallel software can be increased by raising the number of processor cores, is becoming more and more important. Multi-core software is harder to manage and requires a different development approach. The increasing demand for functionality and higher performance of embedded applications pushes these singlecore systems towards multi-core systems [3].

When migrating software from a singlecore to a multi-core environment, a hierarchical design pattern approach [4] depicted in FIGURE 1 can be used.

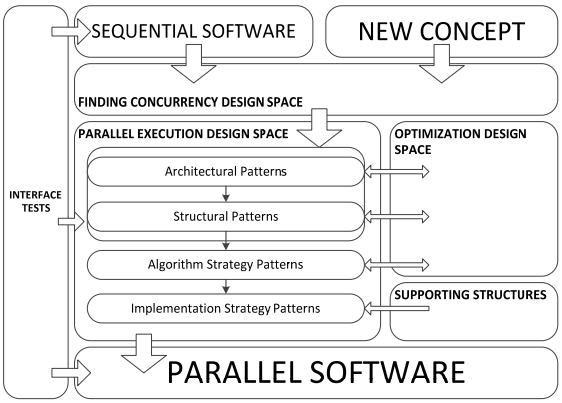

FIGURE 1. PARALLEL DESIGN PATTERN HIERARCHY

Section II describes typical telecom and multi core architectures. In Section III gives an overview of the most important characteristics of the test platform. Section IV describes the test setup. Next, Section V gives an overview of the different tests, while Section VI lists the test results. Section VII describes possible future experiments. Finally, a conclusion is formulated in Section VIII based on our test results.

II. BACKGROUND

A. Telecom Applications

Typically telecom systems are designed applying a planar architecture pattern [5], as shown in FIGURE 2. On a lower level this is based on the Task Decomposition pattern [6]. Software is divided into different planes based on their processing requirements.

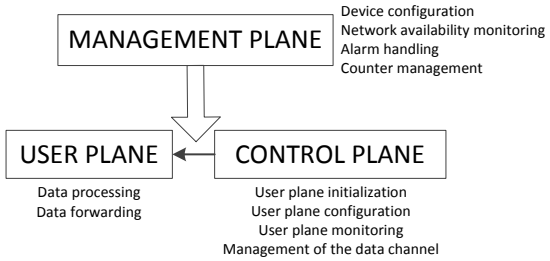

FIGURE 2. TELECOM NETWORK PLANES

The management plane (MP) is responsible for the general management of the system (handling alarms, counters, etc.). The control plane (CP) contains several functions concerning communication initialization, such as codec negotiation. The user plane (UP) provides the actual data processing and packet forwarding.

Non-functional requirements, such as high throughput and low latency, are extremely critical in case of the user plane. The control plane is less critical in terms of latency. Performance requirements for the management plane are even less stringent. Small execution delays are acceptable.

B. Multi-Core Architectures

A typical multi-core system is build using a symmetric multiprocessing (SMP) architecture [7], consisting of a single operating system (OS) managing all processor cores. This setup gives good hardware abstraction and allows for easy system management while providing integrated core load balancing.

N. De Witte, R. Vincke and S. Van Landschoot are scientific staff members at KHBO funded by IWT-110174.

E. Steegmans is a professor and faculty member at KU Leuven, dept. CS, research group iMinds-Distrinet.

J. Boydens is a professor at KHBO and an affiliated researcher at KU Leuven, dept. CS, research group iMinds-Distrinet.

On the other hand an asymmetric multiprocessing (AMP) architecture allows each processor core to run its own instance of an operating system. Different cores can run different operating systems or even run only a baremetal application (freestanding). This way, legacy singlecore applications can be reused and run in full isolation without need for refactoring or redesign to perform migration from a singlecore system to a multi-core system. Whenever applications running on different cores need communication facilities inter-core communication (ICC) techniques such as message passing are required. A specific API such as MCAPI (Multi-Core Communications Application Programming Interface) [8] may be required.

III. PLATFORM

For our experiments we used a Freescale P2020RDB based platform with a dual-core P2020 system-on-chip (SoC) communications processors of the QorIQ family.

The P2020 CPU contains two identical Power Architecture e500v2 cores each equipped with 32KB instruction and data cache. Next to their own L1 cache, the SoC also contains 512KB shared L2 cache.

IV. TEST SETUP

A. Tools

The primary tool used in the different test setups is *Iperf* [9]. A popular tool for testing network performance. *Iperf* is capable of generating as well UDP as TCP packets, while allowing to measure bitrate and packet rate with different payload sizes.

B. Test Automation

The most important requirements of testing are repeatability and consistency. Because of these requirements we decided to work with automated tests. Next to consistency, test automation also minimizes test setup errors and lessens the need for human interaction while allowing faster result analysis.

Several performance characteristics such as packet rate, bitrate, CPU usage, interrupt rate, ... have to be measured on different devices. Therefore the test scripts were written in Python, a platform independent scripting language. The test scripts were modeled following a client-server architecture [10].

The statistics measured are retrieved directly from kernel mapped files instead of using output from system tools such as *vmstat* or *mpstat*. This minimizes overhead during process startup and teardown.

C. Data Terminal Equipment

For all tests the device under test (DUT) was placed between a packet generator device (data terminal equipment or DTE) and a packet sink device (DTE).

In order to get maximum performance out of the DTEs power management and dynamic CPU frequency scaling were disabled. Send and receive socket buffer sizes were increased to support maximum bitrates. Ethernet interfaces were configured to support MTUs (Maximum Transmission Unit) up to 9000 bytes (also known as Jumbo Frames [11]).

To maximize the throughput of the DTEs the test tools (such as *Iperf*) were scheduled [10] with raised extended scheduler priorities, available in Linux [12].

V. MEASUREMENTS

A. Packet Processing Levels

Our first setup, further called the kernel space routing setup, shown in
FIGURE 3, allows the device under test to perform the routing function by enabling IPv4 packet forwarding. This functionality is fully embedded in the Linux kernel and does not require the packet payload to be brought to user space. Therefore the results of these tests served as a reference for the performance capabilities of the system.

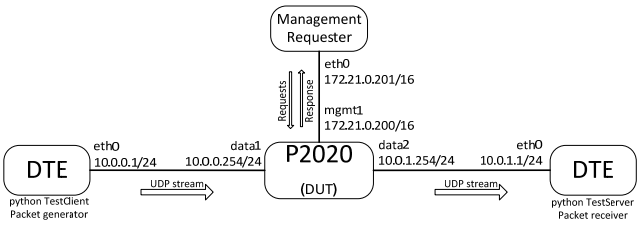

FIGURE 3. KERNEL SPACE ROUTING TEST SETUP

The second test setup, called the user space passthrough, was based on practical telecom applications. For these tests a user plane application was written in C to bring the payload data to user space, process it and forward it to the end DTE, as depicted in
FIGURE 4. The user space passthrough application only forwards traffic from the ingress interface to the egress interface. Therefore, a return link for *Iperf* was created from the sink DTE to the generator DTE.

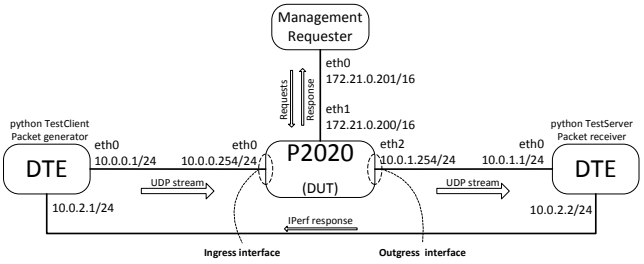

FIGURE 4. USER SPACE PASSTHROUGH TEST SETUP

To test cross-plane influence between user plane applications and management plane applications the response time of a computational intensive CGI script was measured in both test configurations. This allowed us to determine the effect of a heavy loaded user plane on the performance of the management plane and vice versa.

B. Dual-Core SMP

In the dual-core SMP case all hardware components were assigned to a single operating system, as shown in Figure 5.

This allowed the operating system to manage all hardware and schedule processes as needed.

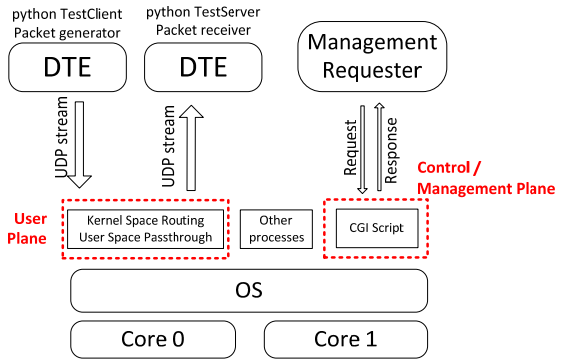

FIGURE 5. DUAL-CORE SMP SYSTEM

The user plane packet processing was handled by the kernel space routing process or by the user space passthrough program. The management plane was in turn emulated by the CGI script. All tests were run with Ethernet interrupt priorities divided between the available cores [10].

C. Dual-Core AMP

In the dual-core AMP case the goal was to isolate the user plane from the control and management plane applications, minimizing the chance of cross-plane influence effects. All hardware components were divided between the available operating systems. Each operating system represented a different plane and was hosted on a separate core.

In case of the P2020RDB system used for this study however, there were only two cores available. One was used to host the user plane while the other core served the control and management plane as indicated in FIGURE 6.

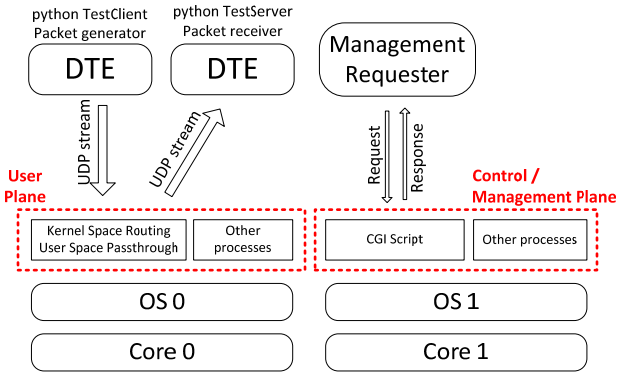

FIGURE 6. DUAL-CORE AMP SYSTEM

VI. RESULTS

The first test results, shown in Figure 7, are these of the kernel space routing tests. The graph indicates the resulting throughput of the SMP and AMP systems in function of the payload size. In both cases the tests were conducted with and without management requests. The graph also shows the difference percentage between the SMP and AMP throughput without management requests.

The results indicate that the AMP system was not able to process as much packets per second as the SMP in case of smaller payloads. Once the payload became higher than 564 bytes the AMP configuration was able to catch up to the SMP system.

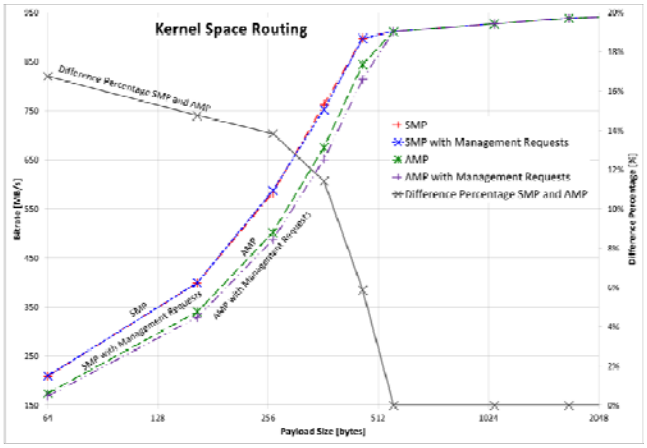

FIGURE 7. KERNEL SPACE ROUTING TEST RESULTS

The test results show no explicit cross-plane influence from the management plane to the user plane in case of SMP. There is however a noticeable performance difference for the AMP case. The introduction of the management requests seem to have a negative effect on the throughput of the user plane application.

The next test results, given in FIGURE 8, are those from the user space passthrough configuration. Again both systems (SMP and AMP) were tested with and without management requests.

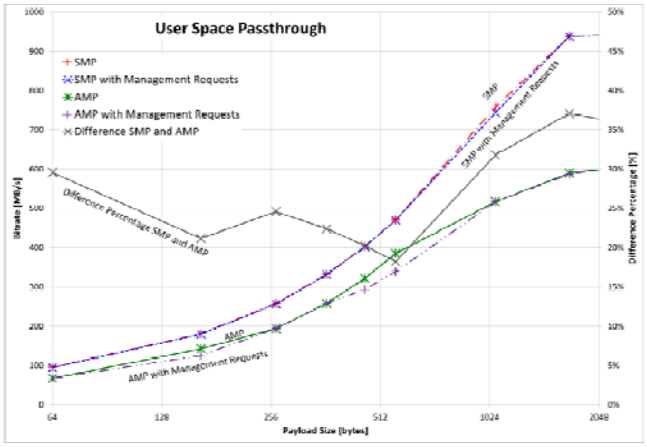

FIGURE 8. USER SPACE PASSTHROUGH TEST RESULTS

The throughput is a lot lower for both systems in case of smaller payloads (< 1064 bytes). The test results show that the throughput of the SMP system is noticeably higher than that of the AMP system. SMP is almost able to utilize the full bandwidth, while the AMP setup seems to saturate at a bitrate of about 600Mb/s.

The AMP configuration displays a clear indication of cross-plane influence from the management plane to the user plane. Even while the applications were hosted on separate cores.

In case of the SMP system the load of the management plane has almost no impact on the throughput of the user

plane. The response time of the management requests does however suffer under the processing load of the user plane traffic, especially when the payload is processed in user space, as can be seen in FIGURE 9. In some cases the requests resulted in timeouts (60s in graph).

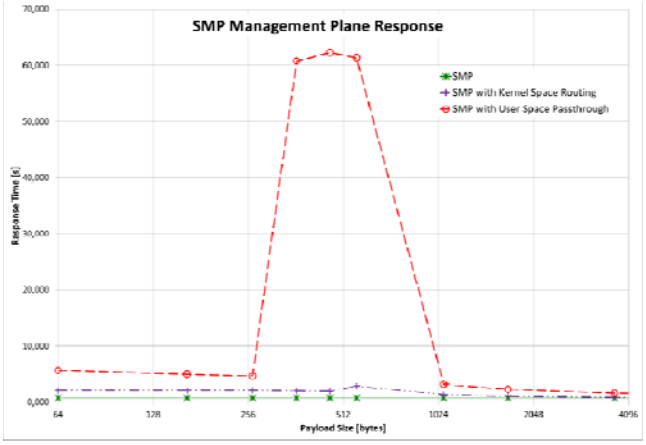

FIGURE 9. SMP MANAGEMENT PLANE REQUEST TEST RESULTS

The cross-influence effect in the AMP configuration is even better pronounced when comparing the base response times of the management requests to the response times with user plane traffic processing, as was done in FIGURE 10.

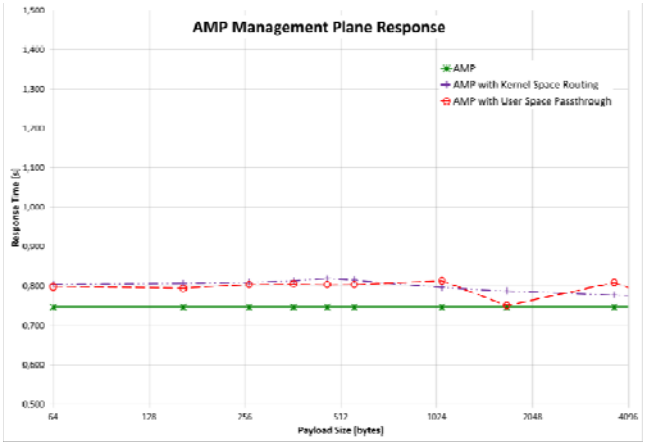

FIGURE 10. AMP MANAGEMENT REQUEST TEST RESULTS

VII. FUTURE WORK

Both operating systems were isolated on separate cores in an AMP configuration. However, shared hardware resources such as L2 cache and memory could introduce cross-plane influence effects. The exact cause of this effect could not be determined and requires further investigation.

Throughput and response time are important parameters, but so are delay and jitter, especially in a telecommunication system. These characteristics could not be accurately measured using *Iperf* and should be measured using an external hardware analyzer capable of high precision time-stamping.

The throughput of the AMP user plane application may be increased by load balancing the application across multiple operating systems. This strategy was not implemented as we only had a dual-core platform at our disposal.

Typically the control plane monitors and configures the user plane. This was not implemented in this case study and would effectively require an ICC (Inter-Core Communication) API for the AMP configuration to enable communication between both planes.

VIII. CONCLUSION

From the tests performed with the dual-core P2020RDB based platform it we conclude that in this case the dual-core SMP outperforms the dual-core AMP configuration. Especially when the telecommunication system requires user space traffic manipulation. This is due to the fact that in case of AMP only one core was available to process Ethernet interrupts and serve the user plane application. In case of SMP it was however crucial that Ethernet interrupt affinities and user plane process scheduling were modified based on the performance requirements of the system.

Cross-plane influence was present in both setups. While the SMP bitrate does not suffer under the extra management load, the AMP throughput did. This was not expected since both planes were handled by separated systems hosted on separated processor cores. The exact cause of this cross-influence has not yet been determined up until this point, but is likely to be found with a shared resource such as L2 cache or memory.

REFERENCES

[1] G. E. Moore, “Cramming More Components onto Integrated Circuits,” vol. 86, no. 1, pp. 82–85, 1998.
[2] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *Proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS ’67 (Spring)*, 1967, p. 483.
[3] J. Nowotsch and M. Paulitsch, “Leveraging Multi-core Computing Architectures in Avionics,” *2012 Ninth European Dependable Computing Conference*, pp. 132–143, May 2012.
[4] R. Vincke, S. Van Landschoot, E. Steegmans, and J. Boydens, “Refactoring Sequential Embedded Software for Concurrent Execution Using Design Patterns,” pp. 157–160, 2012.
[5] A. MacKay, “Three design models for multicore systems,” *EETimes-India*, no. February, p. 2, 2008.
[6] T. G. Mattson, B. A. Sanders, and B. L. Massingill, *Patterns for Parallel Programming*. Addison-Wesley Professional, 2004, p. 384.
[7] Freescale, “Running AMP , SMP or BMP Mode for Multicore Embedded Systems,” pp. 92–97.
[8] “Multicore Communications API (MCAPI) Specification,” 2011.
[9] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, “Iperf: The TCP/UDP bandwidth measurement tool,” 2006.
[10] N. De Witte, R. Vincke, S. Van Landschoot, and J. Boydens, “Evaluation of a Dual-Core SMP and AMP Architecture based on an Embedded Case Study,” 2013.
[11] A. C. Study, W. Feng, J. G. Hurwitz, H. Newman, S. Ravot, R. Les Cottrell, O. Martin, F. Coccetti, C. Jin, X. D. Wei, and S. Low, “Optimizing 10-Gigabit Ethernet for Networks of Workstations , Clusters , and Grids□:”
[12] R. Love, *Linux Kernel Development (3rd Edition)*. Addison-Wesley Professional, 2010, p. 440.