Improving SMP Deficiencies Regarding Real-Time Application Characteristics

David Akre, Jon Anderson, Miguel Martinez-Violante,

Nwachukwu Okwudirichi, Greg Romero

Electrical and Computer Engineering, University of Arizona  
Tucson, Arizona 85721

dakre@email.arizona.edu, jonanderson@email.arizona.edu, mmartin01@email.arizona.edu, oknwachukwu@email.arizona.edu, gregrom@email.arizona.edu

Abstract— Multicore Symmetric Multiprocessing systems (i.e. SMP) running real time applications is an area that requires improvements. Most open source SMP operating systems are time sharing based that allows multiple processes to share memory and processing resources. This is great for increasing bandwidth for multi-process multicore systems, but adds additional latency overhead that is not desirable for real-time applications. The goal for this research paper is to identify these deficiencies in multicore systems running SMP Linux, and to propose improvements to these systems to become more real-time oriented.

***Key Terms: SMP, AMP, OS, RTOS, QoS, IPC, CPI, TDD***

1. **Description**

SMP is a hardware and software operating system concept of sharing resources amongst multiple processes across multiple processors. The SMP OS assigns tasks among the processors to provide a natural load balancing amongst all processors [1]. Prior to SMP, there was Asymmetric Multiprocessing (i.e. AMP) which was used for uniprocessor systems. In AMP only one process was allowed to use the processor at a time. Evidently, allowing this type of separation between processes leads to less interfering tasks, which in turn leads to less overhead of context switching [3]. This is ideal for real-time applications. Overall, running applications on an SMP system is advantageous to increase the system’s total bandwidth [2]. In order to achieve more real-time capabilities the SMP architecture needs to decrease process latency and increase multi-process determinism. These improvements could be locking certain processes to certain processors, restricting memory regions to pertain only to specific processes, allowing the ability to have user space applications run at the highest operating system priority, and other such improvements made at in the Linux kernel.

1. **Motivation For Topic**

In industries such as Aerospace and Telecommunications there is a lot of emphasis on real time applications. These applications require deterministic responses from processes due to the fact that mistimed responses from processes can lead to system failure or a loss in quality of service. Achieving real-time typically means there’s a trade-off in increase bandwidth unless there is unlimited CPU resources. Ideally these systems have increased bandwidth while maintaining its real-time determinism. Thus, there is motivation to find improvements to the SMP architecture to maintain its increased bandwidth over AMP, and to achieve real-time capabilities.

1. **Related Work**

Research in improving the SMP architecture regarding Real Time applications is unique. There is related work in this area primarily in the Telecomm and Aerospace industries, as well as academic research at Universities. In the Telecomm industry there is much desire to increase quality of service (i.e. QoS) for internet or digital signal processing applications. Ideally, these applications can produce responses at a deterministic deadline as well as maintaining a high throughput to process data (i.e. higher QoS) [3]. Similarly, in the aerospace industry missed deadlines potentially mean loss of life. NASA, has carried out research in this area to improve processing performance while still maintaining real-time capabilities [1]. Lastly, Dayanada Sagar Institutions has carried out similar research in comparing and contrasting AMP and SMP in terms of real time capabilities [3]. Their approach is more academic, in comparison to corporate or governmental interests.

1. **Proposed Methodology**

The proposed methodology is the following:

1. Research the SMP architecture and find where improvements can be made. SMP inherently is used in most modern multicore systems, so the target hardware for this research will include embedded systems with multicore processors (e.g. Raspberry Pi’s and NVIDIA Jetson TX1), as well as desktop machines that contain modern Intel processors.
2. Find out how these improvements effect an OS such as Linux running on these multicore processors. This part’s focus is on creating an adaptable simulator using open source tools such as simple-sim, gem5, and LTTng to capture various metrics (e.g. CPU execution time for an individual process, CPU execution time for all processes, CPI, IPC, etc.).
3. Implement the improvements. These improvements can be in the user application space level or kernel level (if there are hardware improvements, these will come in the form of designs instead of code implementations).
4. Create benchmarks to test the improvements made to the SMP architecture by running them against the adaptable simulator created in step 2.
5. Reconvene and assemble results and findings into a comprehensive final research paper.
6. **Proposed Timelines**

The following section describes the project timeline and how it maps to deadlines. Additionally this section covers the allocations of tasks assigned to each team member.

1. Project Timeline

The section below describes the projects timeline in terms of development stages and the allocation of work to each team member.

1. *Stages of Development*

The figure below is a visual representation of the different development stages for this project.



Fig. 1. Project development stages

1. *Breakdown of Work*

At this stage of the project there is still the initial research being performed on the topic. Additionally, at this time possible simulations and test implementation details are still being investigated. If an item is TBD it means that tasks have not yet been allocated.

The contents and this chart will be updated as the project progresses.

TABLE I  
Early Stage 2 Breakdown of Tasks

|  |  |  |
| --- | --- | --- |
| Name | | Tasks |
| Akre, David | Initial research and path finding | |
| Anderson, Jon Mychal | Initial research and path finding | |
| Martinez-Violante, Miguel Angel | Initial research and path finding | |
| Okwudirichi Nwachukwu | Initial research and path finding | |
| Romero, Gregory | Initial research and path finding | |

1. *Configuration Management*

The scope of this project allows for software development. Since there are multiple members in our team some type of configuration management must be maintained.

All software will be kept in a group repository using GitHub [4]. The process for modifying files is outlined below.

TABLE II  
Early Stage 2 Breakdown of Tasks

|  |  |
| --- | --- |
| Step 1 | Check file out from repository |
| Step 2 | Perform updates on file |
| Step 3 | Perform regression test (if needed) |
| Step 4 | Commit file back into repository |

1. **Anticipated Results**

There are different degrees of “real-time” such as hard real-time, and soft real time. Hard real time means that there are no system failures and follows the “5 9’s” rule (e.g. the system can only be down 5.26 minutes per year or is 99.999% available). Soft real time does not follow that policy, and allows processes to miss deadlines on an infrequent manner (i.e. missing deadlines does not mean system failure). Since Linux is naturally a time-sharing operating system employing SMP functionality, improvements can be made to make the operating system to become more real-time capable. These improvements most likely would not be considered hard real time due to the short time frame of ensure enough rigorous testing to enforce hard real time requirements. Thus, overall the anticipated results will reflect a soft real time system, where predictable responses can be achieved but not deterministic.

**References**

1. Kenneth Vines, and Len Day, “Multicore Considerations for Legacy Flight Software Migration” JPL NASA [Online] Available:<https://trs.jpl.nasa.gov/bitstream/handle/2014/43215/12-5292_A1b.pdf?sequence=1> Accessed: Feb. 7, 2017
2. Vaidehi M, T.R. Gopalakrishman Nair, “Multicore Applications in Real Time Systems” [Online] Available: <https://arxiv.org/pdf/1001.3539.pdf>, Accessed: Feb 8, 2017
3. Nico De Witte, Robbie Vincke, Sille Van Landschoot, Eric Steegmans, and Jeroen Boydens, “Evaluation of a Dual-Core SMP and AMP Architecture based on an Embedded Case Study” July 2013 KU Leuven <https://www.researchgate.net/publication/309829008_Evaluation_of_a_Dual-Core_SMP_and_AMP_Architecture_based_on_an_Embedded_Case_Study>, Accessed: Feb 8, 2017
4. Group 3’s Source Control Repository: https://github.com/dakre21/ECE-562