Review of Paper 1
by- z5284381

Review of: Micro-CLK: Returning to the asynchronicity with Communication-Less Microkernel by Yauhen Klimiankou Institute for System Programming, Russian Academy of Sciences klimenkov@ispras.ru

Summary

Problem being solved:

This research paper critically analyses that Inter-process communication (IPC) has always been the biggest problem with the microkernels and the entire history of microkernel development is tightly coupled with the debates about IPC, its efficiency, and the bottleneck it creates. The message passing has always been considered as the foundation to the microkernel but the research paper intends to revise the place of IPC in the microkernel-based multi-server operating system design and argues that communication can and potentially should be moved out to the userspace, making microkernel the first-tier resource manager solely. The research paper therforefore, describes a novel approach for the same and showcases preliminary evaluation results for the same.

Solution addressed:

The authors through the scope of this paper are trying to create/ put forth a design for creating a microkernel design of one of its own kind, by experimenting with the very nature of the microkernel design itself and by trying to test, implement and experiment with trying to create a microkernel design with the entire IPC operation moved out of the kernel and microkernel only being a bare bones resource manager managing the memory, I/O and scheduling with the communication between the processes being moved out implemented as asynchronous messaging in the multi-server environment and showcases the preliminary results regarding that.

Background and Motivation:

The primordial importance of the communication and IPC inside the microkernels due to historic reasons makes the task being solved a worthwhile one as throughout the history of microkernel research novel ways of making the IPC faster have been looked into and almost all the Microkernel OSes tend to perform more crossdomain communication and thus have more context and process switches. Due to which as a result, the communication facility is the most critical microkernel mechanism concerning efficiency.

Almost all the second generation microkernels till now have been using synchronous message passing as a way for IPC between the processes(with the exception of the notification like objects being used in the recent times within the seL4 and the MCS kernel). The experience has showcased that synchronous message passing has not been able to fit well in all the circumstances of the communication patterns(for example, notification delivery, message broadcasting, polling, and others) and a whole lot of optimizations have been made from the time Liedtke had mentioned the research paper states. The research paper but no tries to do something revolutionary by reconsider and revise the design of the multi-server microkernel-based OS returning to the direct asynchronous message-passing model and creating sort of "communication-less microkernel design". The real aim by doing this is to minimize the number of control transfers while doing the data transfers and the data transfer here is performed directly between communicating processes without kernel involvement. Meanwhile, process activations triggered by message arrival are batched and driven by process switches, not vice versa and the preliminary results showcases that communication less principles actually do work well with the growth of communication intensity (although the results can be slightly off as well).

Design and Implementation:

The basic design that the system intends to use is of the asynchronous message passing and its core feature is the decoupling of control transfer from data transfer mediated by kernel-managed buffering of on-the-fly messages. It was criticized by the early implementation of it as being : too many context switches per one message transfer; extensive memory copying; complex dispatching and buffer management infrastructure, which lead to inadequate cache utilization and also the needs for the specific buffers associated with it in the kernel complicates the task even more and before this was only considered as a auxiliary means aiming to compensate for the drawbacks of synchronous messaging. The research paper deals now and looks into the lower level computer systems world as the world with the event and the data- driven nature which leads to looking at the all the processes being in the idle state/ in its stable equilibrium when it is processing no data and but then it generates a burst-collapse cycles of computational activities wherein each of the processes then produces the data all willing to back in their halting state.

The design proposed here tries to copies the message into the dedicated kernel managed buffer assigned to the receiver and multiplexes all incoming messages and the kernel ensures that the receiver is not on the blocked state while this is happening and if not then it tries to move it in the ready-to-run queue. The design system that they have proceed tries to significantly reduce the number of traps inside the kernel which definitely is quite a powerful idea and for doing that they

consider for not having any IPC calls itself when the data is being sent between the processes(i.e no send() or recv() will be called) and also more than one instances of the data can be shared at the same time which makes it really worthwhile of a task to do. The dramatically increased amount of inter-server messaging in multi-server OS has been considered to be a significant contributor to the performance loss. The asynchronicity in the system is managed by just a basic system call called equilibrium() where the thread tries to voluntarily block itself and the kernel removes it from the ready-to-run queue. As a part of the design an abstract message channel has been hypothesised for being able to asynchronously be able to transfer the message across the processes. It decouples the control transfer from the data transferring and message buffering therefore is a subset of it. So, message channel as being described is just an shared memory region which can be used to communicate between the two uses without invoking the kernel itself. But will rely on the kernel for the control transfers, and has a private bidirectional message exchange protocol well in palace before this happens. The second abstract notion being created is of the IPC metachannels it is a memory page mapped back in both the user space and the kernel space and contains the send and recv parts of 512 slots?? For pulses. Pulse is a short 2-byte word by which user space application and kernel exchange lazily IPC-related information. Placing a pulse with a local id of channel into its metachannel, the thread notifies the kernel that it has sent a message and asks it to assure that the receiver is not in a blocked state an the pulse from its IPC metachannel on the receiver side provides the receiver a hint about the channel containing one or more new messages. Thus, the sole purpose of the pulse is to bring the attention of the kernel to the receiver process and then bring the attention of the receiver process to the channel containing new messages. In a real setting a combination of both the pulses and the message channel is used to asynchronously communicate. The real genius thing about this design is that no IPC call is never made to call the kernel while sending the messages which on the outset looks like could improve the performance a lot during high communicative settings..!!

Performance/Results/Evaluation:

On the basis of results of preliminary evaluation and comparison with seL4, μCLK. The author states that while executing a sequential application the results are more or less similar to seL4 but i think that the author was not very clear about this result in itself and it was just shown that it can probably change on the basis of the architecture wherein it looked like the results here were slightly deceptive as it and probably message length was not a good parameter to judge and compare up fro the results. The next result shows that with a 4 server setup a clint is able to send a message with about 27 &% less cost in the μCLK and in general tries tp convince that with the increase in the number of the

communication intensity the more stable cost it has and comparatively with the other system it showcases that it is more efficient.

Verdict and utility:

The work done by μCLK is definitely quite unique, very well researched and is one its kind and based on the references provided the work has not been done before and definitely has been quite impressive in terms of theorising the work. But they definitely could have done better when it comes to producing the results and testing the performance. The results do not necessarily show that separating the IPC out of the microkernel in actuality increases the performance per se. On the level of ideation it definitely tells us that the system is powerful and the creation of communication less microkernel is considered as a big step from the second generation microkernels, but still of lot of development, testing and formal verification will be required to actually bring this out close to its competitors like seL4(wherein the new MCS kernel has been able to solve much of the issues reflected with being formally verified, and properly benchmarked).

Strengths

- Solving the long lived problem of segregating the IPC from the microkernel.
- The issue being solved is highly relevant and the general results and the design structure looks really promising and look like a new and stronger design pattern for the execution of microkernel with the
- Having no IPC class ang no coupling between the data and the control switches is definitely a huge plus.
- Great referencing and the previous work in the domain has been looked into very carefully.
- Very well researched work and the work can turn to be ground breaking for the results that it has downs and the ideas being reflected
- Reproducibility of the results in general is fine if the system being created is available as open source.
- Great explanation throughout and has been able to convince the reader with the problem that they are trying to solve by properly explaining the context in the beginning.
- The paper makes the good use of diagrams to explain the design decision.
-

Weakness

- Safety of the system has been compromised, or nothing in general has been talked about the security of the system and using the system in this way will make things difficult to be formalised.
- Some grammatical mistakes like in the related works have pointed to the link [27] with writing even but it doest connect correctly throughout.
- The paper's approach to security has been a bit optimistic and a bit more was expected to talk about the security of the system.
- Testing and benchmarking was not done properly.
- Only microbenchmarks, no macro benchmarks were done.
- The architecture was quite specific, meeting their system needs for the testing and the performance evaluation which lead to their results superseding other systems.
- Nothing has been talked about the erroneous situations and the error handling in the system.
- Nothing has been explicitly mentioned about how do the system deal with the notions if the shared buffer gets full or gets written with a garbage then how to invalidate or find that in the first instant.
- In general security of the system and formalisation has not been spoken of, which still makes it just a theoretical piece with a lot more to do to actually be on par wit the counter parts like seL4.

Criticism

- No macro benchmark results have been given and neither the details of the setup on which the experimentation was done has been provided(which is a serious crime in gernots
- Although the research paper does a great job in setting up the appropriate context but it seemed that it was a bit too much of the introduction and talking about a lot of fundamental issues which for the most part were pretty standard.
- The system does not do a great job in explaining how the process identities are found and how does the process when maps its data in the shared buffer the and the id in the multichannel the receiver is correctly able to find the process data and can it affect in having the garbage at the other end and somehow finding the wrong process ids to give back the data to, more depthness into the design decision could have been slightly better.
- Currently the receiver either periodically checks either if its IPC metachanne hs something or whether the shared buffer has got something targeted to it(of which no clarity has been given as to how the different processes know the communication endpoints)l page by the receiver can cause wasting of CPU cycles, having a sort of the notification object which notifies the waiting and then dealing with it can save some cpu Cycles and is a slight shortcoming of the design being used at the receivers end (could potentially be a different issue when in multi- core but still could be slightly better)..!!

- Since there are no capabilities and the endpoint objects, how is the correct process know to whom the reply has to be sent back if no metachneel is used and if the process were to ust check if it as received something in its shared buffer(probably being a part of its TCB).
- Nothing has been talked about the multiple calls to the receiver at the same time and how the queuing is done??.
- How does the system deal with if the shared memory gets full, no explanation??.
- Does the system assume that it can run at most 512 processes(no explicit mention of it) as the 512 are the number of slots that it assumes for the send and recv channel in the meta channels for each of the processes.
- Nothing has been talked about the inconvenience in the state of the princess if equilibrium is called an inconvenient periods i.e if in the middle of execution of another operation and can potentially cause the system to be in invconvenient states..!!
- Possibility of the shared memory being mapping into the incorrect processes shared spaces and leading to the inconsistent errors but nothing has been spoken of that..!!
- Not much explanation regarding the **'Current time quantum**' has been given.
- Not much details about the multichannel being a reserved memory has been given.
- Communication between the sender and the receiver is not necessarily real time as all of them will first queue up on the kernel and lets say if the receiver is expecting to get a result and it is not constantly checking up on the sheared buffer and only looking on the meta channel which will only be reflected when the kernel update it then it can be an issue but only when the other receiver is not in equilibrium case, which can be a slight issue if it were an real time OS (although could be made real time if it is guaranteed that the kernel thread updates the metachannel after a certain period of time).
- No measure of variability (e.g. standard deviation, error bars) is given for any measurement. These benchmarks are likely to be fairly consistent between runs, but some confirmation is necessary.
- Percentage of improvement and the measurement using the message length is not properly the best way of checking the system is better, the benchmark results could have been better off tested with respect to number of messages being sent.
- Performance increase of 3.61x was mentioned in the beginning but nothing has been shown regarding that in the research paper.

Generalization and Conclusion

Overall the research paper does make some really solid points and gives a good overview of the possibility of having a full fledged Operating system based on asynchronous communication channels and not having to not wrap into the kernel itself. The idea seems great but it is still in the developmental phase and would still take some time to come in close conjunction with the other game players like seL4. The problem being solved has always been relevant and the solution seems elegant but the results and testing were not very through and no formal proofs of the systems working has still been shown and above that there are still small minor issues with the system that can be a potential issue and would have to be tackled with but definitely would

be one of the things to look for in the future. Although the research paper does mention some really relevant points but still it was more on an experimental phase and further development with proper benchmark results and security proofs with slightly better design could possibly make it one of the most ground breaking things in the Microkernel research.