**Proposal – 2023**

**Title**

- sOS: OS's memory manager is actually a scheduler, so let's treat it this way for greater efficiency and velocity

**Supervisor and location**


- Alain Tchana (Grenoble INP): alain.tchana@grenoble-inp.fr
- ERODS team, LIG lab, Grenoble


**Context**

How many times have we found ourselves not doing the dishes after dinner because the dishwashing liquid was finished? Do we ever think about using shower gel, sink gel, or laundry detergent in those moments? Never, because none of them are called "*dishwashing liquid*" even though they are all just soaps. All of this is to say that the names we give to things have consequences on how we think about those things.
The Operating System (OS) community has been calling the memory manager the *memory allocator* since time immemorial. The community calls the processor manager the *scheduler*. The specialization of the processor allocator as a scheduler shows the importance that the community has given to processors compared to memory, even though both resource types are of equal importance. As we demonstrate below, the goal, the actions, and the events of the scheduler and the memory allocator are comparable at every point, which leads us to consider the memory allocator as a scheduler.

We believe that it is useful way to draw parallels between these two fundamental aspects of system resource allocation and control for two reasons:

- OS development velocity: OS's maintainers will only need to follow one code base instead of two as currently;
- Memory management will take a rapid improvement from the tremendous amount of prior studies realized on CPU management.

The similarity between processor management and memory management:

- RAM is a collection of physical page (PP) frames <-> The CPU package is a collection of cores.
- Virtual pages (VP) virtualize physical pages <-> Threads virtualize physical cores.
- Memory allocator maps VP to PP <-> The scheduler maps threads to cores.
- Swapping in/out <-> Context switching.
- A thread can release memory (e.g., madvise MADV_DONTNEED) <-> A process can yield.
- Page fault <-> Tick timer.
- VP pinning to a NUMA node <-> CPU affinity.
- And so on.

This comparison highlights the analogous relationships between memory management and processor management in an OS.

**Goal**

Our goal is to introduce sOS, a framework that unifies processor and memory management under the same API. Policies written for one of the resources can be applied to the other without rewriting. Furthermore, by unifying the two resources, communication between the two managers (which is important in the OS) will be simplified. Expertise in one domain can be used in the other.

The internship will be conducted in four steps:

1. **API:** Formalize a common interface for implementing memory and CPU allocators.

2. **Prototyping:** Develop a prototype of sOS using USM, a user-space memory allocator developed within our team. The idea is to reshape USM to conform to the unified interface determined in step 1.
3. **Specializing:** Implement several memory and CPU allocation policies.
4. **Evaluating:** Evaluate the prototype.