Avoiding linked lists in the HPC job scheduler Slurm

Slurm from SchedMD has gained traction as one of the most common job scheduling systems on HPC clusters, including the clusters at UPPMAX. Unfortunately, there are some scalability limitations in Slurm, especially when exposed to a high churn of short jobs, due to some internal data structures growing unwieldy. Many of these data structures are implemented in raw C as linked lists.

Linked lists are in theory suitable if there are remove operations within a list. However, plain traversal of the list is slow on a modern architecture due to the repeated memory accesses that put a strain on the memory subsystem, since addresses cannot be predicted. If every entry is allocated as a separate heap object, a significant overhead is also incurred, increasing the memory footprint and making caches less efficient.

Linked list handling is done as a separate local API within Slurm. This means that one can implement a new list solution, but restricted to the limitations of this API (objects being stored being represented as opaque pointers). We want to explore one or several of the following options:

1. Use a C++ vector<void*> internally. Simple, but will imply C++ linkage. This could work as a prototype, but is unlikely to be accepted into the real Slurm codebase.
2. Implement an autogrowing array list ourselves in C.
3. Keep the current list implementation, but allocate all entry helper objects from a pool allocator (in turn implemented with or without C++ data structures).

These options can be modified and extended in different ways, e.g. postponing full removal of entries from an array list by temporarily replacing them with null entries and occasionally performing “defragmentation”.

The options have different performance consequences and a benchmark will need to be developed and tested in a semi-realistic scenarios on the UPPMAX test cluster Terry, which will be made available to the student. Especially, scenarios that trigger a lot of delete events in random order have to be considered, since this is the scenario where a linked list should, in theory, be preferable over an array list.

Contact: carl.nettelblad@it.uu.se