Parallel Computing and I/O

Adaptable I/O Caching Layer

Machine learning and metadata workloads often perform many small and random I/O accesses.Depending on the underlying storage devices, latency can be problematic for such workloads and significantly impact performance; this is especially true for traditional HDDs, which still make up the bulk of storage capacity. Moreover, data storage is often managed in a centralized fashion and accessed via the network, introducing another source of latency. SSDs can be used to cache data and improve performance.

Existing caching solutions often only provide block-level caching and therefore cannot be used on top of existing network file systems. They usually also operate using simple heuristics that keep block copies on SSD if the underlying block is accessed frequently, mainly providing benefits for use cases such as booting or starting applications.

As part of this thesis, you will implement a caching layer that acts as an intermediary between slow network storage and fast SSDs. Two possibilities for implementation are a user-space file system using FUSE or a pre-loadable library that overwrites the necessary I/O functions. Due to this, the implementation will likely have to be done in a system language such as C, C++ or Rus Additionally, the caching layer should support policies to influence the caching behavior. For example, it might make sense to cache a complete directory on SSD whenever a file from said directory is accessed.

- https://github.com/libfuse/libfuse
- http://www.goldsborough.me/c/low-level/kernel/2016/08/29/16-48-53-the_-ld_preload-_trick/

Contact: › Michael Kuhn (https://parcio.ovgu.de/People/Michael+Kuhn.html) and › Christian Lessig (http://isgwww.cs.uni-magdeburg.de/graphics/