

## Summary

Scientific applications from various domains (e.g. astronomy, bioinformatics) are typically modeled as many-task computing (MTC) workloads. In this computing paradigm scientists and researchers express their complex data analyses as sets of (loosely-coupled) scripts, bags-of-tasks, scientific workflows, MapReduce, or a combination of these. MTC lies at the intersection of high performance computing and high-throughput computing, as typical applications are composed of loosely-coupled tasks that communicate via file system operations, as opposed to the message-passing mechanisms of high-performance computing.

Such applications are composed of large numbers of tasks that exhibit data dependencies, employing file-based communication schemes that generate large amounts of data during runtime. Structurally, MTC applications are composed of large parallel stages, linked by data aggregation and partitioning tasks. The former read data generated by many other tasks, while the latter produce data to be read by many other tasks. Such communication schemes often lead to *storage and network traffic imbalance* and *high variability of the achieved parallelism*, and thus to *limited scalability*. Moreover, typical MTC applications also exhibit a highly-variable data footprint.

Previous studies have shown that traditional, disk-based storage systems limit the performance and scalability of running MTC applications as they are unable to cope with the rate at which applications produce data during runtime. In this thesis, we investigate the performability implications of storing data in memory, as opposed to traditional, disk-based, storage when running MTC applications. We propose to store the application data in the memory of the compute nodes, unified into a fast, in-memory distributed file system. The downside of storing data in the main-memory of the compute nodes is that disks still offer orders of magnitude higher capacities. Therefore, at the moment only applications whose data fit into memory can benefit

from in-memory distributed file systems.

Our approach considers a dual perspective: application-specific versus platform-specific design decisions. First, we focus on tailoring the in-memory distributed file system to alleviate application-specific bottlenecks: storage and network traffic imbalances, and limited scalability. The storage and network imbalances *decrease application performance*, while the limited scalability leads to *poor resource utilization*.

Second, we focus on tailoring the in-memory storage system to the target computing platforms for MTC applications: private clusters, or public clouds. For the former, network performance is stable and predictable. However, the latter suffer from large degrees of *network bandwidth variability*, due to colocation and virtualization overheads, which severely limits performance. Furthermore, current compute clusters exhibit large degrees of *memory and network under-utilization*. Reports show that up to 50% of the memory is unused, while there is also a large amount of network bandwidth available.

To improve application performance, in Chapter 2, we present the design of **MemFS**, a locality-agnostic, in-memory storage system. MemFS distributes data evenly across compute nodes achieving a very balanced storage and network traffic. To achieve good performance, MemFS takes advantage of high-speed modern interconnects such as InfiniBand or 10G Ethernet. Our system outperforms a state-of-the-art locality-based approach and is applicable in both clusters and clouds with premium networks. This work has been published in [108, 112].

To improve resource utilization, in Chapter 3, we present the design of **MemEFS**, an elastic in-memory distributed file system. MemEFS is able to improve resource utilization efficiency by dynamically adding or removing compute resources based on the application storage demand. This work has been published in [110]. Furthermore, MemEFS has been selected as an IEEE TCSC Scale Challenge finalist [111], showing that it can achieve **three dimensions of scalability** - horizontal, vertical, and elastic.

To address the network variability issue, in Chapter 4, we present the design of a **network-adaptation mechanism** for MemEFS. This mechanism monitors the bandwidth capacities of the compute resources, and balances the data distribution accordingly. Compared to its network-agnostic counterpart, the network-adapted MemEFS largely improves the runtime of MTC applications on clouds. This work is currently under submission [107].

To take advantage of the unused cluster resources and increase total resource utilization efficiency, in Chapter 5, we propose a **memory scavenging** distributed file system - **MemFSS**. Through resource disaggregation, MemFSS is able to seamlessly scavenge for available memory in other users' node reservations. Therefore, the total cluster application throughput and resource utilization are higher. Furthermore, the performance impact on the tenant application due to memory scavenging is negligible. This work has been published in [113].

To conclude, this thesis presents our approach of optimizing the execution of

MTC applications from a storage perspective. The techniques we propose largely improve application performance and judiciously use the available resources.

