# On-Demand Buffers For Large Data Streams In High Performance Computing Clusters

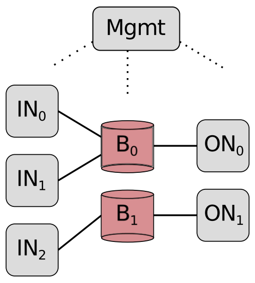
Philip Pum, Robert Kolmhofer
S1010304010@students.fh-hagenberg.at, Robert.Kolmhofer@fh-hagenberg.at

## Problem

Computationally expensive algorithms can quickly become **bottlenecks** in data-intensive high performance computing applications. When using realtime analysis of **continuous data streams** these bottlenecks can lead to significant delays and eventual data loss. This paper presents a framework for **temporary**, **high performance storage** on **commodity hardware**.

## Storage Framework

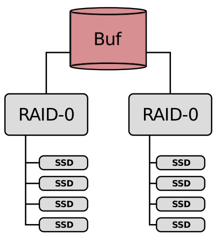
The **CuteforceAnalyzer** is a parallel computing system for the implementation of **cryptoanalytic algorithms**, developed at the University of Applied Sciences Upper Austria. It is designed for **realtime analysis** of different types of data streams like observation of encrypted network traffic or radio communications. The decryption and interpretation of network streams is not a trivial task and involves a significant amount of computing power. At peak times computing nodes can become unable to cope with heavy workloads caused by increased network traffic.



### Hardware

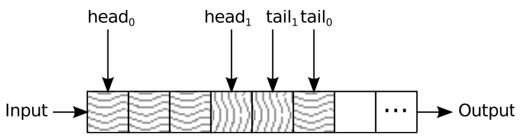
- » **InfiniBand HCA** (40 GBit/s)
- » **Two RAID Controllers** (Mega-RAID SAS 9260-4i)
- » **Eight Solid State Drives** (OCZ VERTEX 3 120 GB)

Initial tests have shown that a single RAID controller can not handle more than four SSDs before hitting its performance limits.



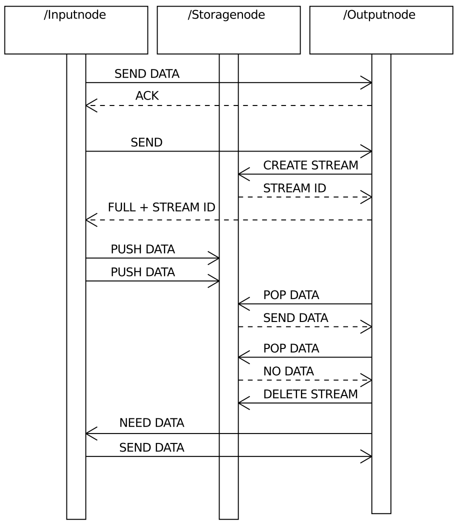
### Filesystem

Storing a continuous stream of data is equivalent to a **sequential write** access on disk. Allocated blocks are marked in **bitmaps** and stored in **FIFO queues**.



### Framework

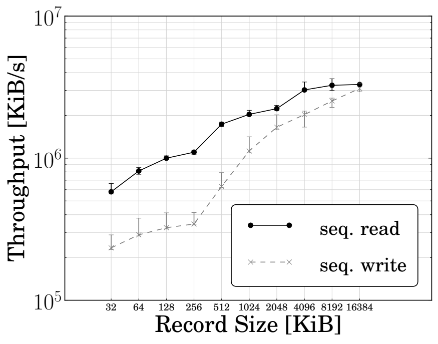
The adjoining figure illustrates a sample data flow between two computing nodes and a single storage node (*SN*). The input node (*IN*) sends data to the output node (*ON*) which in turn acknowledges the reception. The second data transmission from *IN* to *ON* can not be completed since the output node can not process any additional data. When the process queue of the output node is full a command is sent to *SN* requesting a new stream buffer. The identifier for the newly created stream buffer is sent back to *ON* and passed on to the input node along with a status message informing *IN* that *ON* can not process any further information. Any subsequent data packages are sent from *IN* to the storage node for buffering. In addition *ON* starts polling for packets from *SN* until the data queue on *SN* is empty and the buffer is deleted. Finally, *ON* informs *IN* that further data can be sent directly to *ON* again.



## Results

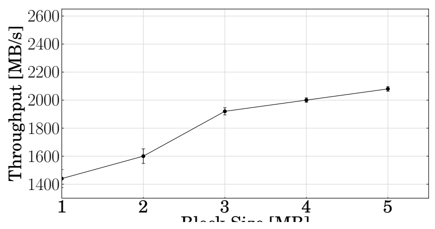
### Storage

The setup allows a single node to **write** up to **3 GiB** per second and **read** up to **3.3 GiB** per second during sequential disk access (without file system).



### Buffer

The following figure shows performance results for buffering a **single stream** between two nodes. A random sample stream was generated on an input node and sent to the storage node. Simultaneously, the output node periodically polled for available data packets. The performance test has been conducted several times for **different block sizes** on the storage node. It is shown that, for large block sizes, a continuous stream can be buffered with up to **2.1 GiB/s**.



## Contact Information

**Philip Pum**
Department of Secure Information Systems,
University of Applied Sciences Upper Austria
Softwarepark 11, Hagenberg, Austria
`S1010304010@students.fh-hagenberg.at`
`http://tinyurl.com/cuteforce`

## Funding

Bundesministerium für Verkehr, Innovation und Technologie
KIRAS Sicherheitsforschung
FFG
OBERÖSTERREICH University of Applied Sciences
Sichere Informationssysteme