



Partnership for Advanced Computing in Europe

The Vagn-Ekman Case Study at SNIC-NSC

Johan Raber, Per Lundqvist and Bengt Persson
(SNIC)

Abstract

Vagn-Ekman is a dual cluster setup with specialized functionality of the two parts. Ekman is a large compute cluster located in Stockholm, and Vagn is a storage and post-processing cluster located at NSC in Linköping. To an extent, the Vagn and Ekman clusters resemble future PRACE operations in the sense that from a large data production facility there will be a need to conveniently transfer the produced data to the researchers, potentially scattered across Europe, in a safe manner with respect to data integrity. The experiences and tools developed during the Vagn-Ekman project can serve as an example for how this data flow can be carried out.

Introduction

Vagn-Ekman is a dual cluster setup with specialized functionality of the two parts. Ekman is a large compute cluster located in Stockholm, Sweden consisting of 1268 compute nodes sporting 8 CPU cores per node and a high bandwidth, low latency Infiniband interconnect. Vagn is a storage and post-processing cluster located at NSC, Linköping with nodes tailored for analysis of the raw data produced at Ekman. In terms of storage space, Vagn hosts about 500 TB of disk storage connected to each post-process node via 8 Gb/s fibre channel in a GPFS file system. In data throughput terms this amounts to 600 – 700 MB/s r/w bandwidth. The Vagn-Ekman cluster is used primarily by two scientific communities: Climatology and Computational Fluid Dynamics.

The Vagn-Ekman cluster duo resembles to an extent future PRACE operations in the sense that from a large data production facility there will be a need to conveniently transfer the produced data to the researchers, potentially scattered across Europe, in a safe manner with respect to data integrity. The experiences and tools developed during the Vagn-Ekman project can serve as example for how this data flow can be carried out.

The Vagn and Ekman Clusters

The Ekman compute cluster is a large cluster located in Stockholm comprising 1268 compute nodes à 8 CPU cores each. It sports a full bisection bandwidth Infiniband® fabric using a multiple root tree structure topology. Ekman has been operational since 2009 and reached the position 89 on the top 500 list of 06/2010 as its best score. It has a rated Rmax of 61760 Gflops out of a theoretical Rpeak of 86024 Gflops.

The split setup of Vagn and Ekman reflects the funding situation and this being a joint project between the two centres National Supercomputer Centre (NSC) and Paralleldatorcentrum (PDC) within SNIC¹, but it also constitutes a logical separation with considerable specialization of the parts.

The Vagn-Ekman cluster was funded by a grant from the “Knut and Alice Wallenberg Foundation” and the Swedish Science Council specifically directed to three Swedish scientific centre constellations catering to two scientific communities, Climatology and Computational Fluid Dynamics (CFD).

The centres representing Climatology are the “Bert Bolin Centre for Climate Research” and the “Rossby center” at the Swedish Meteorological and Hydrological Institute. The CFD research is conducted by members of the

¹

Swedish National Infrastructure for Computing

“Linné Flow Centre” at the Royal Institute of Technology. The two research communities use highly parallelized software codes capable of generating massive amounts of output, to a high degree proportional to the amount of CPU cores utilized. Prime examples of codes used are the EC-EARTH, NEMO and the Simson codes, well regarded codes represented in several work packages of the first implementation phase of PRACE.

1.1. Data Storage on Vagn

Technically, Vagn is a number of SAN boxes exporting disk array volumes on a fibre channel fabric to FC HBA equipped hosts in a GPFS cluster. This allows simple manageability and high availability, for instance online file system expansion and maintenance. These GPFS hosts also serve the purpose of post-processing nodes accessible via a batch queuing system. The data volume produced on Ekman is substantial but its local storage resources are not dimensioned to harbour the data produced long term. As it is purpose built for it, Vagn fills the need for high-throughput, large volume data storage necessary in raw data post-processing. The post-processing work on Vagn is typically visualisation, data-mining and other types of data refinements.

Given this setup, the users need to regularly transfer data between the systems. The limited size of the Ekman cluster local storage emphasizes the need to actually *move* the data, i.e. remove the original copy on Ekman, as opposed to making a copy to Vagn and leave the original untouched. Not having secure means of doing this makes users understandably reluctant to remove the original, thereby filling the Ekman file system. An automated tool like the one detailed below implementing best practises in data transfer can offer relief in this respect.

Large Volume Data Transfer

Ekman and Vagn are administrated by different organizations – PDC and NSC respectively. They are interconnected via a 10G WAN over a physical distance of roughly 200 km. Both systems run CentOS 5. NSC systems are accessed with SSH using password or public key authentication and PDC systems are accessed with Kerberos telnet or SSH with GSSAPI.

In addition to having POSIX compliant file systems on both Vagn and Ekman, PDC also provides AFS volumes. An automated file transfer tool must be able to transparently deal with file system and authentication differences on both sending and receiving side.

1.2. Problem Description

The trivial synchronous procedure familiar to everyone using HPC for moving data between hosts goes something like this: i) log in and ii) copy local source files to the remote destination or vice-versa. iii) On successful copy operation delete the source and iv) log out. In order to making an application automatically handling the above, such an application will need to handle at least the following problems:

- *Interrupts.* Over a long time, there is a high risk of interrupts due to system failures and network failures. A normal synchronously initiated move requires the user to be logged in while moving and is as such also vulnerable to anything that interrupts the initiating process.
- *Errors.* When moving data, users must themselves make sure that she has transferred the data correctly to the other host before deleting the original. It is a seemingly trivial operation, but in a scripted environment, with many users and over a long timespan, there is a significant risk for human mistakes.
- *Restarts.* Although restarting of file transfers is in a sense trivial, the information on how to restart, e.g. search paths and authentication etc., has to be provided by the users themselves. It is also the responsibility of the user not forgetting to restart and restart the correct way with respect to search paths. Potentially, this could be a task of some complexity, depending on the error causing the restart and the amount of restarts.
- *Resuming.* The larger the file sets, the higher the risk of being interrupted when moving it. Moving a large enough file set risk being starved out, taking very long time to finish or never finish at all if the transfer tool does not support resuming. Resuming from the last file not already transferred guarantees that even very large file sets eventually will complete even when being frequently interrupted.
- *Isolation.* For performance reasons no more than one process should move the same data to the other host. Ideally, the origin data should be immutable or the user should be able to detect if it has been modified.

Given the problem domain described above and the heterogeneous setup of Vagn and Ekman, the following minimal requirements also had to be fulfilled by the transfer tool: It had to support CentOS 5 as installation host and also remote host and additionally support POSIX file systems and AFS as an installation file system and also as a file system source or destination in transfers. It also had to support SSH using either public key authentication or GSSAPI.

FFV² (“Filflyttningsverktyget”) was developed for the purpose of moving data between Ekman and Vagn and is designed to handle the problems earlier mentioned. The basic design idea is simple: 1) Move the data to be transferred into a hidden subdirectory to its parent directory. 2) Make a hidden transfer directory on the target side and 3) repeatedly sync this directory to its remote (hidden) location leveraging the *rsync* application from a self contained job script run via *cron* until considered finished. Then 4) move the hidden target directory to its final, visible location and 5) remove the local transfer directory.

Using FFV, the typical workflow can be summarized as:

- A user submits an FFV job. The minimal information supplied to FFV is the source directory, target directory, with their location and references to necessary credentials.
- FFV does minimal verification that the user request may succeed, e.g. establish a connection to any remote system, validate paths, etc. When using public key authentication through an SSH agent, it establishes an SSH session with an SSH control socket to be used such that the agent does not need to be contacted again unless something causes the job to pause. Lastly, it moves the origin data into a transfer directory within the same parent directory as the origin data.
- A self contained transfer job script is automatically created on permanent storage containing all information needed to move the requested data to the target location.
- The user is supplied with a handle, that is, an FFV job identification token which can be used to query FFV about the transfer status, yielding a brief report on progress or possibly problems and solutions.
- The self-contained job script is then executed asynchronously via *cron*, the prompt is returned to the user and FFV requires no further input unless an exceptional event occurs such as expired credentials, or other non-transient errors requiring user intervention. When necessary, the user is notified either by mail or to a file in the user’s home directory on the system where the job was created.

A high level description of the move procedure initiated by the self contained job script is:

- Transfer the transfer directory to the requested location until both successful and no more files are transferred.
- Transfer the transfer directory as above but do so while verifying the data checksums on the receiver side and transfer side (as opposed to only comparing meta data).
- Move transfer directory at the receiver side to the target location. Remove transfer directory at sender side. FFV addresses all of the above itemized problems:
- Interrupts and error handling. The job will silently retry when detecting certain transient errors (any file system temporarily unavailable, source or target host temporarily unreachable due to one or both of the hosts being down, or otherwise being unreachable). If everything is OK the job ends in state finished. The job may also end in state failed if the origin transfer directory vanishes prematurely or if job is cancelled by the user. At any time the job can also pause upon unknown errors or if credentials needs to be renewed. The user is notified and must then manually resume the job or cancel it. If a job is failing it tries to restore the original contents, move back contents of sender transfer directory and then remove both transfer directories. FFV contains tools to see current jobs, get job status, and pause, resume, authenticate, cancel any current jobs. By making the move procedure completely asynchronous, we remove one big source of interrupt. FFV may be used to completely detach the process of moving data from computations. FFV also gets the ability to automatically resume operations, even if the host it is executed from temporarily goes down, given that the necessary credentials still are present when the system is up again.
- Restarting and resuming transfers. FFV uses *rsync* as the file transfer tool and gets the ability to restart and resume for free and it may be used with SSH as remote shell.
- Isolation. By moving the origin data into the transfer directory, we get some isolation from other processes and a cue to the user that the data is being moved asynchronously. Given the unique name of this directory FFV may safely delete it when finished without worrying about whether the user by

mistake has put any more data within it. It is also hard for the user to accidentally modify the origin data, except for the case of deleting its parent directory by mistake.

There are many additional features of FFV. In terms of security, FFV has been designed with significant focus on security. It runs as the actual user creating the job, it does not copy any credentials - it only refers to existing credentials.

FFV is designed to be portable. On the remote side, FFV has only a few requirements: rsync, SSH, Bourne shell and some widely available tools like; df, tail and awk to detect if file system is mounted. Further; rm, mv, mkdir, rmdir and additionally md5sum, xargs and find for data tamper detection. A possible further improvement is to replace all of this with corresponding sftp commands at the cost of losing the tamper detection ability.

Since FFV relies on SSH as a transport layer, it can potentially support all methods of authentication that SSH implements. Not all SSH authentication methods are implemented though. Currently implemented FFV credential supports are: Path to SSH agent socket, Kerberos Ticket and X.509 proxy certificate as well as a path to an already established SSH control socket. By using this, FFV supports any authentication method already supported by SSH.

Conclusion

At its core, FFV is an application automating best practices in data transfer also providing convenience and error checking to the user. Anyone can replicate this functionality provided they have the scripting or programming skills necessary.

However, learning best data transfer practices and automating them should not be required of the \emph{user} since it is usually not their core competence and this provided the impetus to make the file transfer tool. Many things can be improved in the tool and it could benefit from being re-written in a different scripting language as well as a general refactoring. This aside, the basic design choices in FFV are sound however, we believe, and the focus on low maintenance overhead and ease of use for the end user are its core virtues as compared to other means of file transfer.

On a different note: A slightly surprising effect which surfaced in the Vagn-Ekman project was the very high load that encrypted transfers using SSH can put on the login node with transfer bandwidths made possible over 10 Gb Ethernet. Since we use HPN patched SSH servers (parallelized AES crypto) in both ends, the CPU use on the login nodes many times reached full load on all available CPU cores. This prompted setting up a separate transfer node on the Ekman side to keep a workable environment on the login node. Before using 10 Gb Ethernet this was masked by the bandwidth bottleneck. Whether you use HPN patched or not, many simultaneous transfers can accomplish the same effect on any system.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557.