High Throughput Sequencing and the IT architecture

Part 1 : Volume dimensioning and filesystems

George Magklaras

The Biotechnology Centre of Oslo, The University of Oslo (Biotek - UiO), Oslo, Norway

Improvements in DNA sequencing technology have reduced the cost and time of sequencing a new genome. The new generation of High Throughput Sequencing (HTS) devices has provided large impetus to the life science field, and genome sequencing is now a necessary first step in many complex research projects, with direct implications to the field of medical sequencing, cancer and pathogen vector genomics, epi- and meta-genomics.

However, despite the falling sequencing cost and time-lines, there are other associated costs and difficulties in the process of maintaining a functional data repository on large-scale research projects. The new generation of HTS technologies [1] has introduced the need for increased data-storage technologies whose capacity is well beyond the average local data-storage facilities [2]. In fact, the computing world has produced a new term for this paradigm, that of data-intensive computing [2a]. Data-storage costs are falling; however, a study of the functional specifications of popular HTS equipment, such as Roche's 454 pyrosequencers [3], Illumina's hardware [4] and ABI SOLiD technology [5], suggests that a single high-throughput experiment run creates several Tbytes of information. If one takes into account that genome sequencing is often performed repeatedly in order to study genetic variation [6], the capacity of a suitable data-archiving facility needs to scale to several Petabytes of information, which is well beyond the scale of most group, departmental or university computing facilities.

Storage of the data is only one of the technical problems. The distribution and post-processing of large data-sets is also an important issue. Initial raw data and resulting post-processing HTS files need to be accessed (and perhaps replicated), analyzed and annotated by various scientific communities at regional, national and international levels. This is purely a technological problem for which clear answers do not exist, despite the fact that large-scale cyber infrastructures exist in other scientific fields, such as particle physics [7]. However, genome sequence data have slightly different requirements from particle physics data and thus the process of distributing and making sense of large data-sets for Genome Assembly and annotation requires different technological approaches at the data-network and middleware/software layers. For instance, security concerns for clinical HTS settings are an issue, as genomic information concerning patients is really a patient record and thus needs to be addressed in concert with hospital IT and security procedures, subsequent to institutional security compliance procedures. Moreover, other field-relevant procedures, such as the de novo genome assembly of vertebrate-size genomes, require an unusually large amount of RAM per processor/core (more than 500 Gigs of RAM), which may be a challenge, depending on the budget size of the HTS facility and the expertise required for running large shared-memory computers.

This series of articles will discuss and attempt to address all of these challenges by means of flagging various existing and emerging IT technologies. In this first part, we will examine a strategy to plan for the amount of disk space you need to store and archive HTS data, and look at various choices for one of the most critical modules of an IT storage infrastructure: the file-system.

Handling HTS data volumes is a classic example of data-intensive computing [8]. One of the most important aspects of handling a data-intensive computing problem is to understand the amount of data you will be dealing with.

Table 1 provides an indicated maximum (sizes vary depending on the exact sequencing parameters and the experiment) data volume per device type on an annual basis. All HTS devices have a simple conceptual pipeline. Each stage

of the pipeline indicates a storage tier. Each storage tier represents different storage functional requirements, in terms of the amount and access pattern of storage needed:

- Tier 1: Includes the raw unprocessed data as they come out from the instrument (mostly images). For most HTS devices, Tier 1 data will generate several Tbytes per run (several thousands of Gigabytes), especially as the instrument's ability to become more precise gets better with time (firmware or device upgrades). This type of storage acts as a front stage area and needs maximum I/O performance, as concurrent disk write and read operations occur most of the time: write ops occur from the HTS devices; read ops are essentially copies of the raw data by the analysis nodes. Normally, the HTS workstations offer local high-performance disk drives to accommodate these requirements per instrument (DAS, Fiber Channel). When the initial sample run is complete, these data need to be moved to the Tier 1 area to clear the local hard drives for more space.

- Tier 2: Initial processing data stage: including base (or colour) calls, intensities and first pass quality scores. These data are currently in the order of several tenths of Gigabytes to 300 Gigabytes per run maximum for certain types of sequencers.

- Tier 3: Includes aligned and analyzed data (alignments of all the reads to a reference or de novo assembly, if required). This can be at least as big as the initial processing stage (Tier 2), as the initial reads themselves have to be preserved as part of the alignment output. At the end of each successful processing step, the raw data of Tier 1 are removed.

- Tier 4: The final, fourth tier includes data that should be backed up off site, in order to provide for disaster recovery, as well as a long-term archive. This includes a mirror of Tiers 2 and 3, plus the archive requirements. It is not financially feasible or technically practical to off-site backup Tier 1 data, at least not for every run, as the volume of data is huge. There is some data redundancy between Tiers 2 and 3, as in theory one could resort to Tier 3 reads according to the alignment output and then discard Tier 2 data. However, this might not be feasible/desirable in all analysis scenarios, and thus we assume it is good practice to backup and archive both Tier 2 and Tier 3 data.

Tier 1 could be implemented as a disk redundant (RAID 1, 6, other) data-storage area with capacity given by the following equation:

$$Tier1_{store} = \sum (N_{hts} \times G_{bpr} + (N_{hts} \times G_{bpr})/4)$$
$$N_{hts} = number\ of\ per\ type\ HTS\ devices,$$
$$G_{bpr} = Gigabytes\ per\ run$$

The (Nhts x Gbpr)/4 factor represents a small recommended buffer to accommodate unexpected stalls of the HTS pipeline (loss of computing nodes, problems in copying/referencing Tier 1 data, etc).

Tiers 2 and 3 can occupy a common set of disks to form the analysis staging area, according to the following equation:

$$Tier2,3_{store} = \sum (N_{runs} \times G_{analysis} + (N_{runs} \times G_{analysis})/3)$$

$$N_{runs} = expected\ number\ of\ runs\ per\ year,$$
$$G_{analysis} = Gigabytes\ per\ run\ for\ Tiers\ 2\ and\ 3\ (Table\ 1)$$

Finally, Tier 4 backup and storage requirements depend on the data retention policies. We as-

Table. 1: Associating HTS devices and data volumes

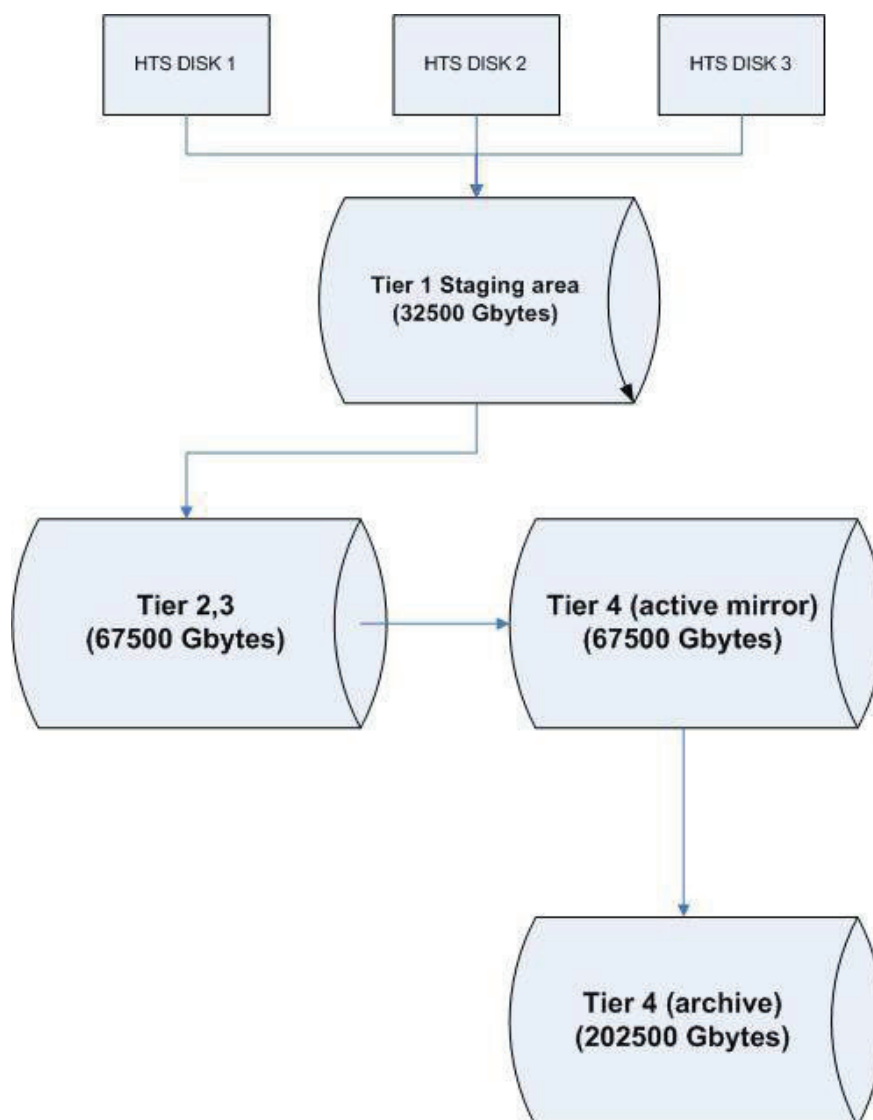| HTS Device	No. of runs per year	Tier 1 Data/run (Gbytes)	Tier 2 Data/run (Gbytes)	Tier 3 Data/run (Gbytes) (Analysis)	Tier 4 Data/run (Gbytes) (Backup and archive)	Produced data per year (Tbytes)
Illumina	100	9728	100	300	400	990
454	100	200	50	25	75	27
SOLiD	100	6144	100	100	200	80

Figure 1. Breaking down the storage requirements per tier.

sume that processed experimental data should be kept for a number of years, thus:

$$Tier4_{store} = Tier2,3_{store} + R_{period} \times Tier2,3_{store}$$

$$R_{period} = number\ of\ years\ to\ keep\ the\ data$$

Based on these equations, and Table 1 requirements, for an example that includes 2 Illumina sequencers, a couple of 454, a single SOLiD machine and a retention period of 3 years, we need a tiered infrastructure as shown in Figure 1. Note how Tier 4 is broken down to the active mirror (facilitates disaster recovery) and the archive, which could include slower Disk-to-Disk (D2D) or tape robot archive solutions.

This methodology shows how easily data volumes can scale to large data-sets. The tiered architecture allows one to scale different storage requirements independently.

Whilst the previous paragraphs quantified the amount of data produced by an HTS facility and the volumes, it is really important to consider how to build those volumes. It becomes clear that handling the size of the HTC data-sets can be an IT engineering challenge. Traditional High Performance Computing (HPC) has addressed the issues of tackling large data-sets by introducing a number of technologies (queue batch/grid

systems, specialized processor interconnects, parallel programming). Not all of them are adequate to address the challenges of large HTS data-sets. In particular, two IT areas need a significant engineering overhaul to handle the volume of HTS data efficiently.

A file-system is a key component of the Operating System that dictates how the files are stored and accessed. Weaknesses in the file-system ability to handle large data-sets can severely affect HTS operations.

Today's common file-system choices include options such as ext3 [9] on Linux platforms, NTFS [10] for Windows platforms, and HFS+ [11] for Apple-based computers. They perform well with common desktop storage loads and with certain multi TiB scenarios on a file server. However, they do not scale well up to hundreds of Terayte or Petabyte-scale scenarios, either because of specific file and volume limits (ext3) and limits in handling efficiently a large number of small files in deeply nested directory structures. Finally, there are issues with concurrent access to a set of files and directories, a common scenario in large HPC clusters.

For these reasons, HPC setups have employed different file-system varieties to satisfy large and concurrent access file scenarios. The first category is commonly referred to as 'clustered' or 'shared' disk file-systems. Some commonly employed examples of clustered file-systems include IBM's General Parallel File System (GPFS) [12], Apple's XSAN[13] and other commercial offerings, such as Isilon's OneFS solution [14]. There is a great variety of academic HPC setups that already run large GPFS installations, handling concurrency and high availability with varying degrees of success. Isilon's system solutions also had varying degrees of success acting as a Tier 1 storage solution for some large academic setups.

Most (if not all) clustered file-system implementations assume the existence of a block-device layer, in order to connect client devices and backend storage disks. So, they offer the illusion of a backend storage device appearing as local disk to a system, provided that a Fiber Channel (FC) or iSCSI based solution is employed. This is a highly desirable functionality on the Tier 1 storage area. For example, instead of employing common utilities such as FTP/Rsync to move data off the HTS device to the Tier 1 area, the same thing could be performed by a simple file level copy operation. Depending on the purity of the block-layer protocol (pure FC will not include TCP/IP overhead), copying multiple TiBs of data from the instrument disks to the Tier 1 staging area could be performed more efficiently. Latter paragraphs will touch on a promising technology to further simplify this process (FCoE).

The second (and less commonly employed) category of file-systems is referred to as distributed file-systems. A distributed file-system allows access to files located on another remote host as though working on the actual host computer. In contrast to clustered file systems, the file locality illusion now is facilitated by means of network-layer code and not by block-layer device emulation. NFS and the emerging pNFS standard [15] are commonly employed in many bioinformatics setups; however, the protocol suffers from scalability and data-redundancy problems. To address these problems, a new generation of distributed fault-tolerant and parallel file-systems exists. Two notable examples of such file-systems are the Sun's Lustre file-system [16] and the Hadoop Distributed File System (HDFS) [17].

It is sometimes difficult to distinguish between certain features of clustered and distributed file-system solutions, as the feature-set of new generation distributed file-systems is expanding. However, one notable difference between traditional cluster file-systems and new distributed file-systems is that the latter are explicitly designed to handle data-sets in the order of several Petabytes, something that might not be entirely true for most of the previously mentioned cluster file-systems. In that sense, both Lustre and HDFS are highly suited to facilitate the file-system storage of Tiers 2 and 3, and at least the active mirror of Tier 4. Which of the two is more suitable is a matter of choice and experimentation. Tailored solutions that lead to definite conclusions do not yet exist, and understanding the pros and cons of these two platforms is work in progress.

The most notable differences between Lustre and HDFS is that the second is less mature in generic production environments and requires a substantial effort to express the data-set using the map/reduce [18] concept. HDFS is part of a data-processing framework and as such most data-sets must be converted before they can take advantage of its power. There have been successful attempts to employ the Hadoop

framework on bioinformatics problems [19]. On the other hand, Lustre is a much more file-oriented storage framework. It also requires substantial effort to setup, especially when it comes to converting data from older Lustre versions, but it does not require any explicit data-set conversion.

I hope that I have given your storage architects plenty of food for thought. In the next article, I shall look at the data-network layer.
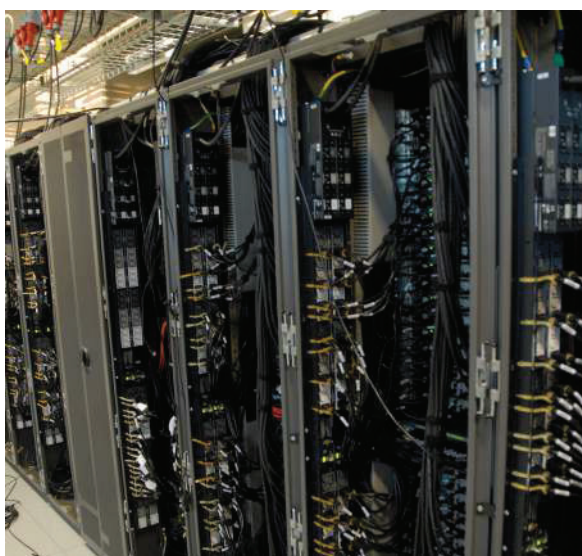

Figure 2. A computer cluster.

References

1. "Genome sequencing: the third generation", Published online 6 February 2009 | Nature | doi:10.1038/news.2009.86

2. "Big data: Welcome to the petacentre", Published online 3 September 2008 | Nature 455, 16-21 (2008) | doi:10.1038/455016a

2a. Gorton I., Greenfield P., Szaley A., Williams R. (2008), "Data Intensive Computing for the 21st Century, IEEE Computer, April 2008 issue.

3. The 454 sequencer product line specifications: http://www.454.com/products-solutions/product-list.asp

4. Illumina Genome sequencers product line specifications: http://www.illumina.com/pagesnrn.ilmn?ID=26

5. Applied Biosystems SOLiD system specifications: http://www3.appliedbiosystems.com

6. 1000 genomes project portal: http://1000genomes.org/page.php

7. The worldwide Large Hydron Collider (LHC) Computing Grid portal at CERN: http://lcg.web.cern.ch/LCG/

8. Kouzes et al (2009), "The changing paradigm of data-intensive computing", Computer, January 2009 issue, IEEE Computer Society, pages 26-34.

9. Stephen C. Tweedie (May 1998). "Journaling the Linux ext2fs Filesystem" (PDF). Proceedings of the 4th Annual LinuxExpo, Durham, NC. http://jamesthornton.com/hotlist/linux-filesystems/ext3-journal-design.pdf.

10. The NTFS wikipedia page: http://en.wikipedia.org/wiki/NTFS

11. "Technical Note TN1150: HFS Plus Volume Format". Apple Developer Connection. March 5, 2004. http://developer.apple.com/technotes/tn/tn1150.html.

12. Schmuck, Frank; Roger Haskin (January 2002). "GPFS: A Shared-Disk File System for Large Computing Clusters" (pdf). Proceedings of the FAST'02 Conference on File and Storage Technologies: 231-244, Monterey, California, USA: USENIX. t

13. "Apple Ships Xsan Storage Area Network File System". Apple Inc.. http://www.apple.com/pr/library/2005/jan/04xsan.html.

14. Isilon Systems OneFS product literature: http://www.isilon.com/products/index.php?page=software_OneFS

15. The wikipedia entry on NFS: http://en.wikipedia.org/wiki/Network_File_System_(protocol)

16. The Lustre filesystem community Wiki: http://wiki.lustre.org/index.php?title=Main_Page

17. The Apache Hadoop project: http://hadoop.apache.org/core/

18. J. Dean and S. Ghemawat, 'MapReduce: Simplified DataProcessing on Large Clusters,' in OSDI 2004: Sixth Symposium on Operating System Design and Implementation, 2004. [Online]. Available: http://labs.google.com/papers/mapreduce-osdi04.pdf

19. Simone Leo et al (2008), 'Parallelizing bioinformatics applications with MapReduce', poster presented at the Cloud Computing Applications 2008 Conference, available at: http://www.cca08.org/papers/Poster10-Simone-Leo.pdf