A Terabyte Analysis Machine For SDSS Data

James Annis†, Gabriele Garzoglio‡, Kurt Ruthsmandorfer†, and Chris Stoughton†

†Experimental Astrophysics Group and ‡Computing Division, Fermilab, Batavia, IL 60510

Abstract. The Sloan Digital Sky Survey data has driven us to design the Terabyte Analysis Machine, a high-I/O compute cluster built on a cluster of commodity computers, a fibre channel network, and a large disk tower; on Linux, on a new cluster file system called GFS, and on the sophisticated SX database. Our initial implementation is a 7 dual processor nodes with 500 Gb of local disk and 730 Gb of global disk. We expect to use the TAM to search the first 1000 sq-degrees of the SDSS for clusters of galaxies.

INTRODUCTION

The 2.5 Terabytes of data from the finished SDSS will consist of catalogs, atlas images, and binned sky. From the 200 Gb of pre-2000 high quality data we have learned the value of the ability to look at the atlas images, about exactly how slow a supposedly I/O limited task like A-star hunting can be, and about just how much compute power it takes to find clusters of galaxies or to measure on all the atlas images weak lensing optimized ellipticity parameters. Our experiences have driven us to design the Terabyte Analysis Machine[1] (TAM); a machine with an architecture designed to allow minimum I/O time for I/O-bound problems and to bring significant compute power to bear for compute bound problems when presented with terabyte scale data sets.

We need access to terabytes of data, but we are not the first. In the 1990's the database world settled [2] on clusters sharing neither memory nor disk, only the network; both the high energy physicists and movie makers adopted that model. More interestingly, Landsat presented the remote sensing community with terabyte scale spatially coherent data; the group at the University of Maryland responded with an exploration of an I/O limited analysis machines ([3], [4], [5]) leading them to the Realm,[2] their next generation compute cluster. We can use this experience to guide our design.

[1] A machine is one or more computers tightly organized to perform a particular task: the computers of Google are definitely a machine (see [1]), whereas most Condor clusters (http://www.cs.wisc.edu/condor) are not.
[2] http://realm.umd.edu

DESIGN NOTES

The basic insight on optical astronomical data is that all of the measurements we do are local: even cluster finding operates on scales small compared to the whole sky. We are firmly in the "embarrassingly parallel" regime, a pleasant surprise: divide and conquer strategies can turn large jobs into multiple small ones and provide naturally linear speedup[3] and scaleup[4]. It is also the ideal situation for exploiting large numbers of cheap, fast commodity processors, disk, and memory. Astronomy is perpetually cash starved; it is rare that the better solution is also the cheaper one, but this is the case.

> Design Note 1. Money -is- an object: use commodity computers and learn to cluster.

We are not inverting maps or matrices, our problems require message passing between CPUs at a period closer to a kilosecond than to a microsecond.

> Design Note 2. Take full advantage of the embarrassingly parallel regime: we can do without fast, low latency inter-node communications, and probably without message passing parallel programming.

We do, however, need access to terabytes of data. This is far larger than can fit into memory and we will need to store out-of-core and intermediate data onto disk, preferably local disk, where bandwidth is higher than global

[3] performing the same calculation n times faster as the machine gets n times bigger
[4] performing in the same time an n times larger calculation on an n times larger machine

disk and guaranteed not to interfere with I/O requests from other processors.

> Design Note 3. We want significant disk local to each machine. Local disk can transform I/O bound problems into compute bound ones.

For these I/O dominated problems, the initial partitioning of the data to local disk is important. The partitioning cannot be write once, partly because we are still taking data, partly because the simple partitioning that is optimal for small numbers of disk becomes sub-optimal for more disks, but fundamentally because the optimal partition depends on the question being asked. We will need to the ability to redistribute data amongst local disk.

> Design Note 4. We want a global store of data.

Finally, we note that observational optical astronomers analyze data in two ways: using quick interactive analyses (the main stay of the field) and long pipeline jobs applied to large data sets. Far and away the best way for astronomers to develop is to have the data available interactively in the same way it is available to the pipeline jobs. Furthermore, when trying to use the compute power gets in the way of doing the science, astronomers chafe: a machine that is fast but inconvenient to use is worse than a slow machine that is easy to use.

> Design Note 5. Thought cycles of astronomers are the scarcest resource of all: design the machine for efficient large scale analyses, but preserve reasonable interactive ease of use.

CURRENT TECHNOLOGY

These design notes, given the current technology, imply a high I/O compute cluster, able by cleverness to do fast I/O bound searches, and able by brute force to perform quick compute bound problems.

> Tech Note 1: Beowulf clusters are commodity items.

The rise of the Internet economy has created a demand for commodity clusters of commodity computers and many vendors now sell turnkey systems at a price varying by $\lesssim 20\%$. Dual CPU boxes are the current maximal performance/dollar point, and the dimensions of the SDSS camera suggest 6 is a magic number.

> Tech Note 2: The current free interconnect is fast Ethernet.

The only significant cost is a switch which gathers the 100-BaseT connections into a Gigabit Ethernet uplink, and this is at the half-node level.

> Tech Note 3: IDE disk is cheap and nearly as fast as SCSI.

One or two IDE disk per node is close to free, (1/10 of a node cost); local disk is not a problem.

> Tech Note 4: There are many ways to make a global store; we are attracted to a high speed serverless design built on a fibre channel network.

The fibre channel network allows multiple nodes to access the same disks over 100 Mbyte/s links. The open source file system GFS[5] controls multiple node access via disk firmware file locks to prevent file system corruption and uses journaling to prevent corruption in the case of a node going unresponsive during I/O.

> Tech Note 5: For most of our uses, a load sharing system is better than a batch system.

Just beyond the state of the art is using Mosix with GFS. Mosix[6] is a kernel modification that allows processes to transparently slide from one machine to another based on machine load.

THE FERMILAB IMPLEMENTATION

Our baseline machine consists of 6 compute nodes and 1 interactive node. All are dual 600 Mhz Pentium IIIs with 1 Gb of RAM and 72 Gb of local IDE disk.

Initially we planned 16×50 Gb disk packaged in two JBODs, which with simple partitioning could allow up to 200 Mbytes/s I/O from the global store. As we had the funds and disliked the idea of retrieving from tape the data on a lost disk, we instead turned to a hardware RAID controller with 10×73 Gb disk.

Our initial testing of GFS has been encouraging. We ran tests with multiple processes reading large files off of a single logical volume on the RAID disk set, both with the nodes accessing the logical volume via GFS, and with one node acting as a NFS server of the logical volume to the other nodes. Using NFS (without any fine tuning), we achieved only 10 Mbytes/sec aggregate throughput. Using GFS we achieved 90 Mbytes/sec, limited by the rate allowed by the single RAID hardware controller.

---

5 Global File System; http://www.globalfilesystem.org
6 http://www.mosix.org

THE TERABYTE ANALYSIS MACHINE

The TAM is meant to enable science; one analysis it is optimized for happens to be the architect's own: cluster finding need access to overlapping sky area and brute compute power to examine large areas of sky. In general, one of TAM's strengths is in bringing large amounts of CPU to bear on easily accessible datasets.

Beyond this, we expect to integrate TAM with SX, the SDSS science archive [6] in two different ways. First, we will access data over sockets connected to the SX database cluster, a cluster optimized for raw I/O speed . In this use, TAM is the analysis cluster attached to the SX database cluster. In a second mode, we will run a SX server on the TAM itself. While accessing the data from this server is a possibility, what really drives us is the exploration of a facility for optimal re-partitioning and re-indexing of the entire SX dataset. For cluster finding, there are efficient algorithms for finding the k^{th} nearest neighbor. In general these involve re-partitioning and re-indexing of the data, and it usually is faster to just do a straight scan of the existing database. However, if one is in the code/run/debug/explore cycle of scientific programming, speed counts. Our plan is to set up a facility to perform the re-partitioning and re-indexing of the data onto the local IDE disk, and leave that version of the data there for a few weeks while the scientist works; a variation of disk caching. For the k^{th} nearest neighbor this will be called the distance machine, but we expect that the infrastructure we build will easily allow the swapping in of modules implementing other database efficient algorithms for other specific problems.

Lastly, having access to all of the data opens up all of the possibilities of the visualization database of Kent: color GIF images of the SDSS data, with position links to the NED database, to the photo output files and spectra, and full resolution reconstructed frames. This is a highly useful tool for understanding the data.

THE TERABYTE ANALYSIS MACHINE AND SDSS DATA

In the year 2000, 2.5 Terabytes is 50 disks and \$50k, so the budgets of interest top out at \$100k. An astronomy department committed to serious SDSS analysis (perhaps together with an aggressive junior faculty startup funds) is capable of building a machine optimized for SDSS work. While one option is to wait and let the compute power and storage space come to you[7], but in reality the scale of interesting problems also grows with time. Good compute nodes cost $\sim$ \$5k, and large fast disks $\sim$ \$1k, roughly independent of time, and roughly scaling with the problems of current interest. The machine we are implementing, 7 nodes with 500 Gb of local disk and 730 Gb of global disk, runs $\sim$ \$80k; about $\sim$ \$110k with hardware RAID.

[7] highly recommended in this regard is [7]

REFERENCES

1. Brin, S. and Page, L. 1998 *WWW7/Computer Networks*, 30(1-7): 107-117
(http://dbpubs.stanford.edu:8090/pub/1998-8)
2. deWitt, J. and Gray, J. *Communication of the ACM* 35(6), 1992
(http://research.microsoft.com/~gray/CacmParallelDB.doc)
3. Acharya, A., Uysal, M., Bennett, R., Mendelson, A., Beynon, M., Hollingsworth, J., Saltz, J., and Sussman, A. 1996., In *Proceedings of the Fourth ACM Workshop on I/O in Parallel and Distributed Systems*, May 1996.
(http://www.cs.umd.edu/projects/hpsl/html-papers/iopads96.html)
4. Chang, C., Moon, B., Acharya, A., Shock, C., Sussman, A., and Saltz, J. 1997. In *Proceedings of the 13th International Conference on Data Engineering*, Apr. 1997.
(http://citeseer.nj.nec.com/chang97titan.html)
5. Shock, C. Chang, C., Moon, B., Acharya, A., Davis, L., Saltz, J. and Sussman, A., 1998. *Parallel Computing*, 24(1):65-90, Jan. 1998.
(http://citeseer.nj.nec.com/shock98design.html)
6. Szalay, A., Kunszt, P., Thakar, A., Gray, J., Slutz, D., and Brunner, R.J. 1999. astro-ph/9912382.
(http://arXiv.org/abs/astro-ph/9912382)
7. Gottbrath, C., et al. Bailin, J., Meakin, C., Thompson, T., and Charfman , J.J. 1999. astro-ph/9912202.