CDF/DOC/COMP_UPG/CDFR/5805
October 26th, 2001
Version 1.0

Benchmarking a MOSIX cluster with CDF software

T. Kim, A. Korn, Ch. Paus, M. Neubauer, F. Würthwein
Massachusetts Institute of Technology

D. Waters
University College London

Abstract

We are evaluating an alternative computing model to the existing central analysis facility (CAF) at CDF. This model is based on data being disk resident on cheap IDE disks on commodity PC's. The driving force behind this approach is a factor 50 advantage in CPU power per \$ over the existing SMP solution, as well as a factor $\sim 4-5$ in storage space per \$ as compared to the existing Fibre Channel disk array's.

We present benchmarking results on the scaling behaviour of the proposed architecture, and describe a model for the expected I/O scaling behaviour that is based on our tests. We outline a set of actual and potential bottlenecks and their implications. Finally, we describe three slightly different architecture options and point out their relative advantages.

1 Intro and Roadmap

A Dual PIII 1GHz system can be purchased for less than \$1200 [1]. In comparison, the recently aquired fcdfsun2, a 24 processor SMP, carries a \$400,000 pricetag plus a \$100,000 3 year maintenance contract. CDF software executes twice as fast on a PIII 1GHz processor than on the UltraSPARC processors utelized in fcdfsun2 [2]. One might thus conclude that a farm of Dual PIII 1GHz PC's exceeds the sun SMP by more than a factor 50 in "bang for the buck". The only problem with this number is that it's a comparison between apples and oranges. After all there is added value somewhere in the SMP architecture. An obvious example is the centralized maintenance.

A quick look at pricewatch.com leads us to believe that a 100GB EIDE drive can be currently bought for roughly \$250 . This may be compared with a recent Fibre Channel disk purchase for CAF of 1600GB for \$ 20,000. Taken seriously, this would be equivalent to a factor 5 advantage in disk space per \$ for EIDE over the Fibre Channel disks. And again, the problem in this comparison is that it doesn't take the fundamental differences into account.

In this note we present measurements of the scaling properties of a system architecture previously presented [3] that might allow us to build a computing system out of cheap components that has similar operating aspects than the present CAF, and substantially better performance at a fraction of the cost.

First experience with a prototype system actively used by about 10 users was gained. The purpose of this note is to start an evaluation of the expandability of such a computing system.

In the following section we remind people of the proposed architecture, and contrast it briefly with CAF as well as a central server architecture. Section 2 provides detailed benchmarking results for the CPU as well as I/O scaling behaviour of the proposed architecture.

Section 4 describes known problems and performance bottlenecks. We comment on possible solutions and implications of these problems. Section 5 discusses slightly modified cluster architectures that are in principle much less susceptible to this type of bottleneck.

Section 6 states our conclusions.

1.1 Discussion of Architecture

1.1.1 CDF Run2a data set

To have a significant fraction, say $O(30\%)$, of the expected $\sim 160TB$ of PAD data for Run IIa one would require $O(100)$ nodes of 0.5TB. For comparison, the Level3 farm in it's final configuration for Run IIa has ~ 270 nodes.

If one wanted to be very aggressive, and keep all the PAD data on disk plus Monte Carlo ($\sim$ 40TB, based on the cannonical 500,000 events per day production on the CDF production farm) on a farm no larger than L3 one would need to attach $O(3/4$ TB) per PC. This is quite possible even with present technology.

The key difference, besides disk space, between L3 and a user analysis facility is that the later has to support $O(100)$ times the number of users than the former. Along with more users come a larger variety of applications, and along with disk space come data management issues, both of which add up to an increased maintenance and operations challenge.

1.1.2 EIDE disks in a standard PC

A standard PC enclosure provides space for 5 EIDE disks or more. At roughly 100GB per disk a single PC might thus store up to 0.5TB of disk space easily. The I/O bandwidth of EIDE disks these days is $\sim$ 30MB/sec. Promise Technology [4] offers a variety of 32 and 64bit 33/66MHz PCI cards that function as storage subsystems connecting up to 4 drives per card to the PCI bus. We tested a system with two "Ultra100" 32bit PCI cards [5] and determined the I/O write bandwidth to be 30.2MB/sec for a single disk, 28.5MB/sec for two disks on two cards, and 24.4MB/sec for two disks attached to the same card. These cards cost \$50 and are by no means the only option on the market. Hardware RAID controllers for example allow to connect up to 8 disks reaching throughput up to 80MB/sec [8].

At present, PCI bus bandwidth ranges from 32bit 33MHz = 132 MB/sec to 64bit 66MHz = 528 MB/sec. The next generation PCI bus is reputed to support 64bit at 133 MHz [6]. There is thus clearly plenty of room to grow for this kind of hardware.

1.1.3 MOSIX Architecture

A cluster of Linux PC's with MOSIX [7] installed as described in Ref. [3] could have the following characteristics:

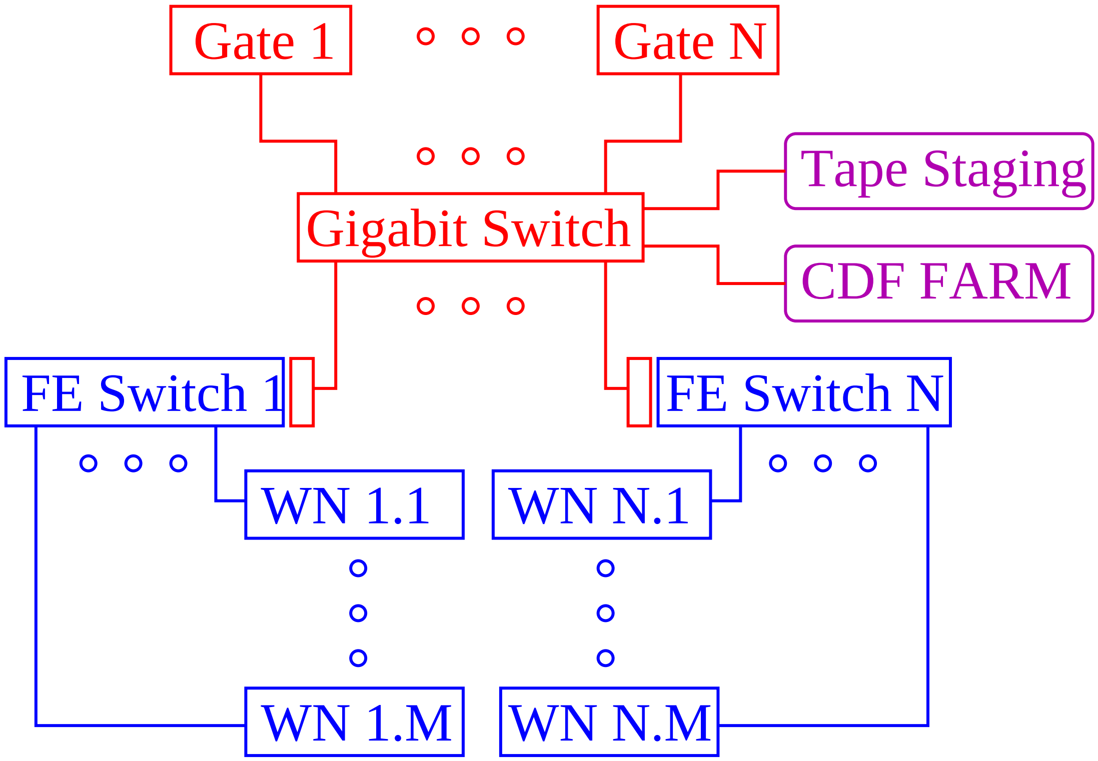

Figure 1: Straw proposal for a central facility based on MOSIX. See text for a discussion of the main features.

- Only a small number of PC's are maintained with full user access. All other PC's are inaccessible to the user except for their CPU power and data access. The bulk of the PC's can thus be maintained with the similar effort as the Level3 or production farms. L3 is presently maintained with 2 FTE.

- Which PC actually processes a given user application is of no concern to the user. The system moves jobs around in order to minimize a cost function based on I/O, CPU, and memory usage of the application. The full farm thus behaves similar to an SMP.

- In particular, under ideal conditions all I/O is performed on local disks because an I/O limited user job will move to the node that has the disk attached. The "re-arrangement" of jobs onto nodes is dynamic. E.g. a CPU limited job will move away from a given node if a new job starts that needs both CPU and I/O from disks attached to this node. This is discussed in more detail in Section 3.

We believe that these characteristics are close to ideal for the kind of requirements CDF has for its user analysis computing. The resulting architecture for a central facility is shown in Figure 1

In practice, the "migration" of user applications across the cluster is possible by splitting up user and system parts of each process. The system portion remains on the PC where the job was started while the user portion "migrates" freely on the cluster to minimize the cost function for the whole cluster.

MOSIX comes with it's own filesystem which allows I/O to be part of the user portion of a process, and thus enabling "migration of applications to the data" rather than "serving the data to an application".

The communication between user and system part of a process is via TCP/IP. "Gateway nodes", i.e. nodes a user can log into and start a job interactively, thus generally may benefit from larger bandwidth connections than "worker nodes", i.e. nodes that own data and CPU power but have no user access. This is especially true if the /home and user data areas are local to the gateway nodes.

If the user application makes system calls then these are routed back to the node where the process was started, and executed there. It is thus crucial to eliminate system calls as best as possible from a user application.

The distinctive features of this farm are thus:

- Small number of gateway notes, e.g. $N = 5$.
- Large number of worker nodes, e.g. $M = 20$.
- FastEthernet between worker nodes as very little network traffic between worker nodes is expected.
- Gigabit Ethernet connections to the gateway nodes to provide extra bandwidth in case there is non-negligible communications overhead or data skimming to user disks on the gateway nodes.

A farm like this might easily serve 50TB of data spread over 100 worker nodes. To store all the Run2a PAD data and Monte Carlo on disk on a farm roughly the size of L3 would require $\sim 3/4$ TB per PC.

For a Mosix farm, the optimal amount of disk space per node is given by the amount of CPU power required to process a certain amount of data. I.e. there's little sense in having TB disk space per node unless the node has the compute power to make good use of that disk space.

The costs for a hypothetical 100 node system might be estimated as follows:

- \$ 1200 per Dual CPU system $\times 105 =$\$ 126k.
- \$ 1250 per 500GB of disk space $\times 105 =$\$ 131k.
- \$ 300 per 2 EIDE controller cards $\times 105 =$ \$ 32k.
- \$ 30k for LAN networking.

The total comes to less than the \$ 400k CDF paid for fcdfsun2, and that system has no disk space included in this price !!! The CPU power provided by such a system would exceed fcdfsun2 by a factor 16.

The 2TB of disk space on the gateway nodes might be used for /cdf/scratch , /cdf/spool, /cdf/home, and the CDF software installations, maybe also mySQL versions of database copies.

1.1.4 Theoretical aggregate I/O bandwidth

The I/O bandwidth of a late model EIDE disk is 30MB/s, and largely independent of blocksizes relevant for CDF data. Overheads due to multiple disks attached to a single controller card or alike might reduce this to $\sim$ 25MB/sec. For event sizes of 100KB this would correspond to 250Hz event input rate into an executable if event input into an AC++ executable was I/O limited. With present run2 software, as well as the future move towards multi-branch I/O it is hard to imagine that any realistic CDF application would ever be I/O limited. The following argument about aggregate bandwidth on a MOSIX cluster is thus probably merely academic.

The theoretical aggregate bandwidth limit of this kind of architecture is given by individual disk bandwidth times the number of disks in the system. For 500 disks of 25MB/sec each this would be a staggering 100 Gbit per second.

In Section 3 we describe a simulation that seems to indicate that $\sim$50Gbit aggregate bandwidth may be reached in practice if no more than one job per CPU is allowed on the cluster. It should be noted that these numbers are based on the assumption, that the needed output bandwidth is small compared to the input requirements. This is certainly true for almost all analysis jobs like ntuple production and data skimming.

In contrast, a central server architecture would divide the pile of PC's into two, one of which serves data, using Gigabit ethernet connectivity whereas the other part of the cluster would process the data, thus not needing the same fast LAN connectivity. The performance of such a system is limited by the server connection.

A Mosix cluster is thus clearly at an advantage if average applications benefit from having local storage. In the limit of clearly CPU limited applications the case for Mosix is much weaker.

2 Benchmarking Results

The crucial question is of course "does it work?". We have started to evaluate this with a series of detailed benchmarking. These are described in this Section. For the most crucial tests it is sufficient to evaluate the behaviour of a single 20 Nodes leg of the Architecture shown in Fig. 1.

There are three types of questions to evaluate:

1. Does the MOSIX migration principle work for CPU limited processes? In particular, is the clock time per job independent of the number of jobs started on a gateway machine as long as the total number of jobs $\leq$ number of CPU's in the cluster? Does the elapsed time increase as expected as the number of jobs exceeds the number of CPU's in the cluster?

2. Does the aggregate bandwidth increase linearly with the number of jobs as long as each job reads/writes from a different disk?

3. How does the aggregate I/O bandwidth scale versus number of jobs if all jobs read/write from/to the same disk?

The system used in these benchmarking tests is a set of 20 Dual PII 400MHz, 384MB Ram, with unfortunately rather small ($2\times$6GB) and poor quality disks. Furthermore, the dd read/write tests on the disks showed that the I/O was quite non-uniform, ranging from roughly 5 to 9.5MB/sec. Most disks had performance near the lower end of the range. It was not possible to reformat all disks to obtain uniform behaviour as they contained user data.

2.1 CPU scaling behaviour

To test the CPU scaling behaviour we start 1-80 jobs simultaneously on the 20 nodes test cluster. All jobs are started on the same PC "interactively". There's no remote execution or batch queueing involved. The benchmarking "jobs" consist of a cdfSim executable. No output file was written. All timing calls were switched of as explained in section 4.

We then record the user, system, and total elapsed time to complete each job. Figure 2 plots the elapsed time for each job on the vertical, and the number of jobs running simultaneously on the horizontal axis.

Elapsed time is simply measured by starting a stopwatch when a job is launched, and stoping it when the job has completed. We see a horizontal set of black open circles from zero to 40 at an elapsed time of $\sim$ 500 seconds. Each dot stands for a single job. At $x = 10, 20, 30, 40$ we see 10,20,30,40 such dots etc. The average time for a given number of jobs run is indicated by the red stars.

The average time per job is roughly 2% shorter for running 10 simultaneous jobs than for running 40 jobs simultaneously. We thus conclude that there is a negligible overhead per job. In addition, running 80 jobs on 40 CPU's takes 1020 seconds on average per job, as compared to roughly 510 per job when each job get's it's own CPU (e.q. 10 jobs on 40 CPUs). Again, we see no significant overheads.

We should note that we needed to modify the standard AC++ framework to achive this behaviour. This is discussed in more detail in Section 4. After these modifications we find near perfect scaling behaviour with system overheads below the 1% level.

2.2 I/O scaling behaviour

Figure 3 shows results of an I/O scaling test involving 1 to 30 different disks. The top two plots show aggregate and average I/O bandwidth. Each job uses Edm_ReadWriteSeqRoot_byTBuffer [9] to read a 1 GB $t\bar{t}$ Monte Carlo file.

The full circles (black) depict aggregate and average I/O bandwidth for these jobs. The open circles (red) show the bare disk I/O bandwidth as determined using the results from the direct measurement in the lower left plot. We start 1 to 30 jobs simultaneously on one node to each read a file from a different disk.

As reference we determine the physical bandwidth of the disks by running dd locally on the same files for each node. The I/O bandwidth of the various disks differ considerably as is seen in the lower left plot. We have reason to believe that this is due to different levels of fragmentation of the small disks, many of which where close to 100% full. The average performance of these disks is with 7 MB/sec considerably worse than what is obtainable with more recent models. Disks with particular bad performance were not used in this test.

The intrinsic variation of disk performance influences the behaviour of the curves for aggregated and averaged bandwidth, as seen in the deviation from linear in the former and the unexpected rise in the later. The ordering of the disks is what creates the peculiar curve in the top right plot. Fortunenatly only the difference between theoretical and achieved throughput is of relevance.

The lower right plot shows the ratio of benchmarked bandwidth over dd bandwidth. The open circles show this ratio for each and every job. The solid line (red) shows the average over all jobs that ran simultaneously at any given point.

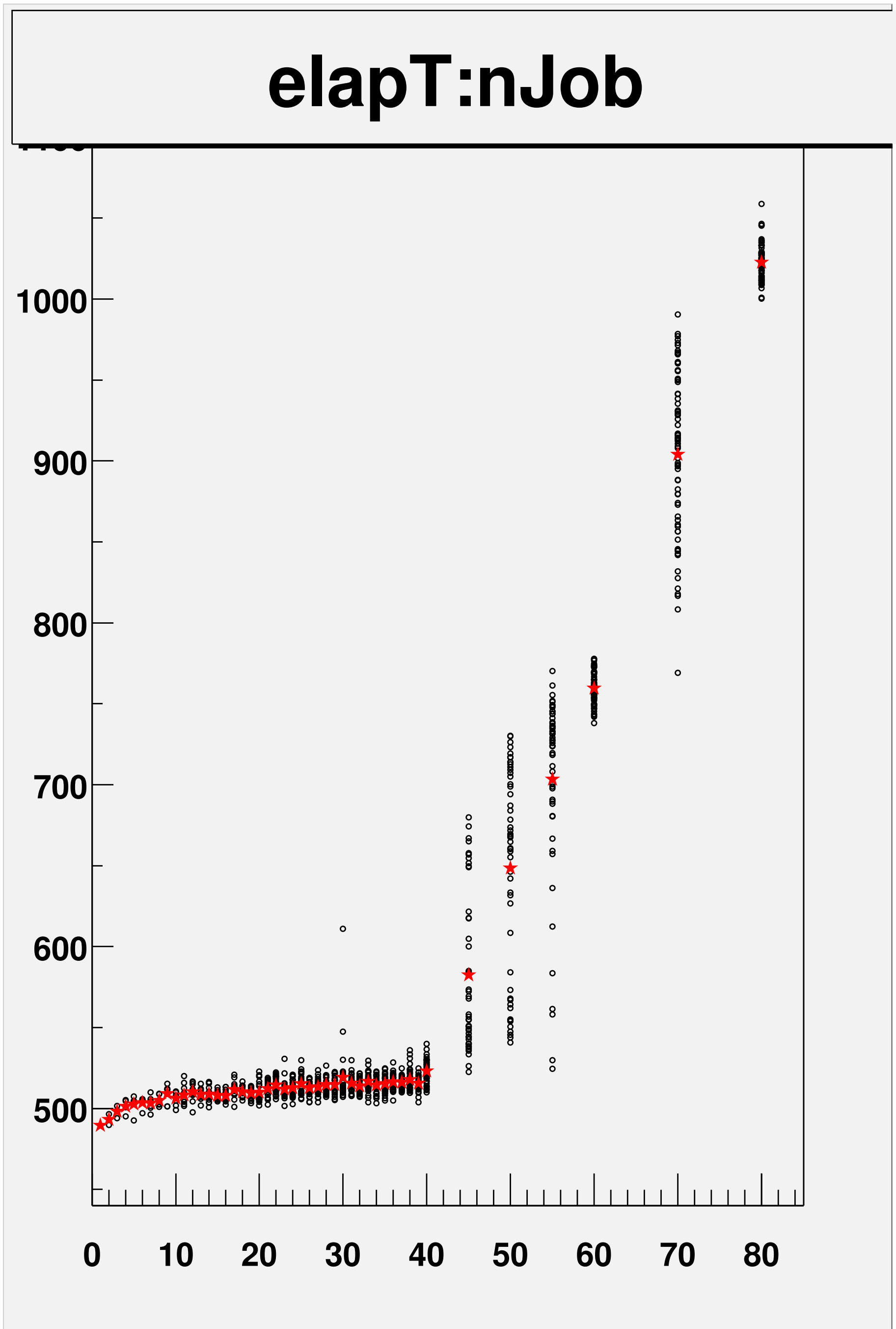

Figure 2: CPU scaling using cdfSim. The Y-axis starts at around 440 seconds in order to show the small slope from 10-40 simulataneous jobs. The red stars show the average while there is a black open circle for every job that is run.

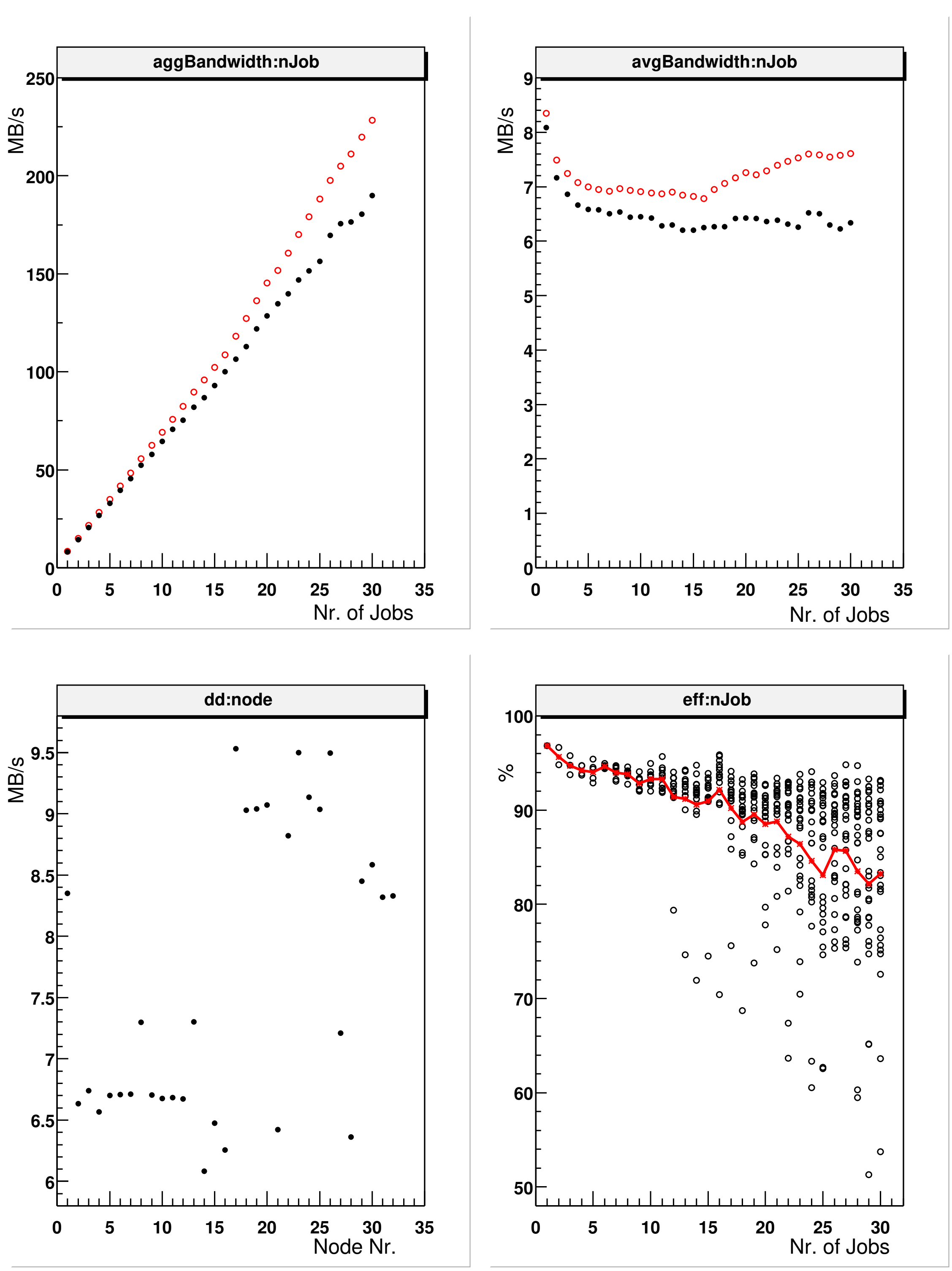

Figure 3: I/O scaling using 30 different disks. For more details see text.

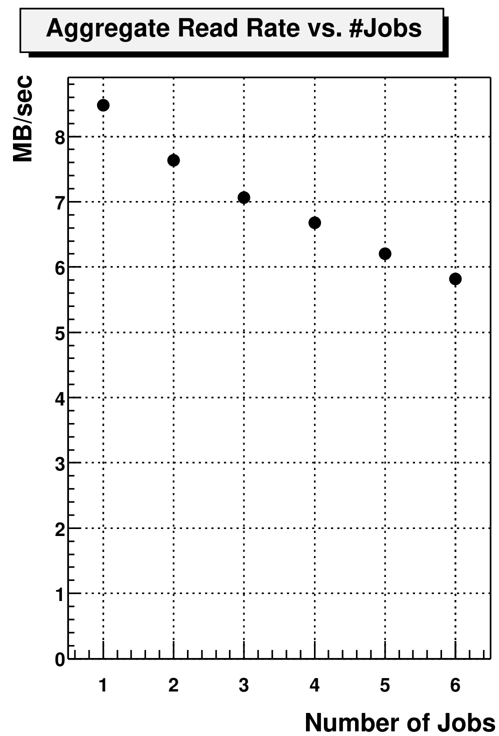

Figure 4: I/O scaling using a single disk

We note that the chosen 1/20 ratio of login/worker nodes is approximately what one might want to consider for a CAF farm of 100 worker nodes. I.e. the cluster benchmarked here may be viewed as one of 5 legs of a 100 node farm. Simply scaling up the 200MB/sec observed in Figure 3 by 5 legs would thus provide at least 8Gbit/sec aggregate bandwidth. Assuming modern EIDE disks with I/O bandwidth of 30MB/sec per disk and fast PCI bus one might reasonably expect something more like 40Gbit/sec aggregate bandwidth. In other words, this is roughly the equivalent of 40 servers attached to a LAN via one Gigabit network link each.

To understand the deviation from perfect scaling we ran strace and tcpdump while a job was reading from a non-local disk via either dd or the CDF Edm tools. There appears to be network traffic in time with mmap, lseek, read, write, munmap system calls indicating that we are indeed seeing a limitation of Mosix migration rather than an effect that is CDF software specific.

2.2.1 Many jobs accessing one disk

To verify that MOSIX doesn't fall apart when many jobs try to access one and the same disk we rerun the previous test with Edm_ReadWriteSeqRoot_byTBuffer but this time with up to 6 jobs accessing the same disk.

As seen in Fig. 4 we find $6-7\%$ overhead per additional job reading from the same disk. The one disk used for many jobs reading from same disk was reformatted such that the maximum I/O it provided was roughly 30% larger than the 30 disks used for Figure 3.

It should be pointed out that this isn't really a very useful benchmark as the actual performance here will depend strongly on the details of the hardware used. E.g. we'd expect a much larger drop if our 20 nodes had better quality disks attached. The reason is that Edm_ReadWrite_byTBuffer consumes more than 50% of a PII 400MHz CPU. What is measured here is thus some complex mix of network, disk, and CPU performance. We did not make any effort in disentangling the pieces.

3 Estimating Aggregate Bandwidth

The architecture we describe here is in principle designed to maximize I/O bandwidth by migrating jobs such that I/O is predominantly local. In this section we discuss a simple model for assessing to what extend all I/O can indeed be local I/O.

We point out again that this is largely an academic exercise as we find it hard to imagine that any reasonable CDF application will be disk I/O limited.

For optimal cluster I/O performance one would probably want to spread a dataset over enough disks such that the problem of assigning disks to jobs is basically a stochastic process. At the same time, one would want a job to always find enough data on a disk to stay there for at least $O(minutes)$ to avoid frequent process migration.

Let's take a look at the CDF datasets to see if these two conditions are met. The triggers and data streams described in CDF notes 4718 and 5565 range from about 20-100nb cross section. This translates into 4e7 to 2e8 events per $2fb^{-1}$, or 4 to 20TB of disk space for an event size of 100kB. Assuming a file size of 1GB this corresponds to 4000 to 20,000 files, or about 40 disks of 100GB each for a 4TB dataset.

Given that our intention is to build a facility that allows efficient skim of huge data samples, and given the nature of physics analysis, it is probably not unreasonable to assume that any user we care to consider will want to analyze enough data to consider job $\rightarrow$ disk matching a random process.

We have thus simulated the following I/O model:

1. Assume the cluster has 100 nodes, and each node has a number of disks.

2. Assume that 1,2,3, and more jobs per node result in an aggregate I/O bandwidth for that node of 30,54,86,100MB/sec. These numbers are motivated by our measurements on a PC with two Promise cards as discussed in Section 1.1.2. It implies that the PCI bus has a limit of 100MB/sec. We consider this to be conservative given presently available technology.

3. Assume that *numJobs* jobs run on the cluster at a given point in time.

4. Match jobs to nodes at random irrespective of how many jobs are already paired with a given node.

5. Calculate the total useable bandwidth by just adding up the aggregate I/O bandwidth delivered of all 100 nodes.

Figure 5 shows a simulation of aggregate bandwidth versus number of jobs running for a 100 node MOSIX cluster. The horizontal scale shows the number of jobs on the cluster while the vertical scale depicts aggregate I/O bandwidth of the whole cluster.

We conclude that saturation effects do not appear to be a major issue until there are roughly 50% more jobs than CPU's on the cluster.

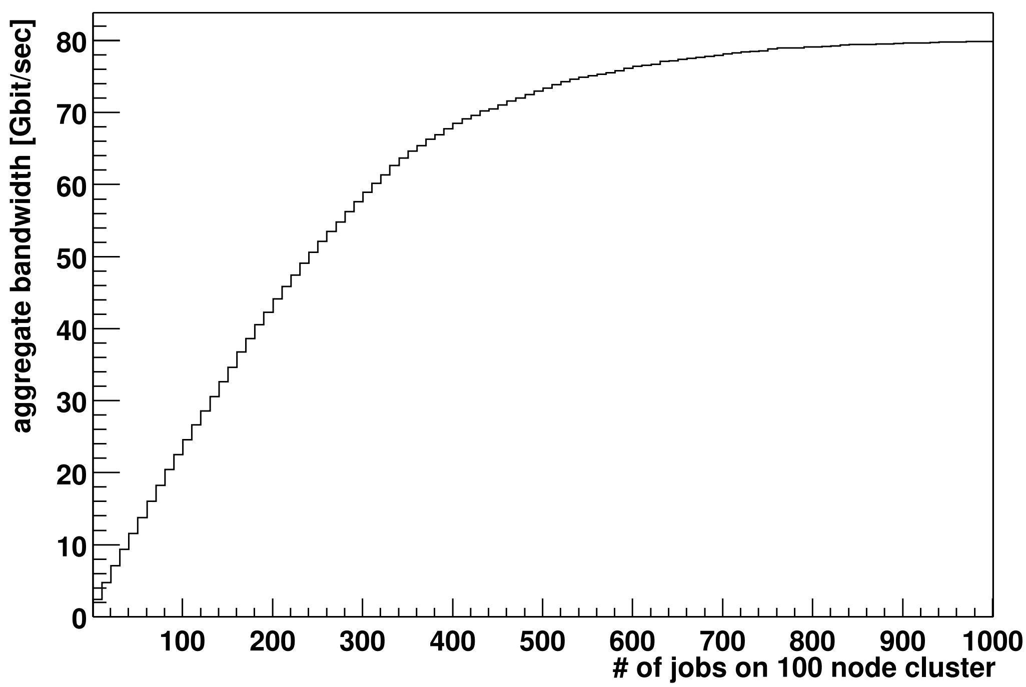

Figure 5: Simulation of useable I/O bandwidth per number of jobs on the cluster. A cluster with 100 nodes was simulated.

4 Known Problems and Potential Bottlenecks

A typical analysis or reconstruction job in CDF is comprised of many modules that are invoked by a framework refered to as AC++. The framework times each and every module using the system calls *gettimeofday* and *times*. A typical AC++ job thus leads to many such system calls per event.

Mosix executes all of these system calls on the node the job was started. Starting many jobs from the same node does cause the node to lock up fairly quickly unless one disables the timing calls in the framework. This was done for all the benchmarks presented in Section 2.

This example shows that Mosix is quite vulnerable to potential idiosyncracies of the user applications. Performance evaluation of common applications is thus crucial for satisfying cluster performance.

- We have not tested the scaling behaviour for CDF software compilation. Compilation consists of many short jobs with substantial I/O. Good scaling for this type of application might not be obtainable out of the box. We have some indications from the prototype, but haven't spend time to optimize the MOSIX configuration. A complication we did encounter is related to the Prelink Server of the Kai Compiler. It uses special files (Unix-domain socket file) that are not supported by MFS. We were able to solve this problem with a small modification to `KCC.real`.

- We have not tested the scaling when analyzing actual CDF data. It is conceivable that accesses to calibration as well as other databases might pose a bottleneck because at present MOSIX can not migrate network connections. All database accesses by all jobs started on a gateway node would have to go through that node. The successful usage of the prototype system at FNAL for data analysis shows that this issue is manageable. But conclusive tests don't exist yet and it is not clear if this might pose a problem with a bigger system.

- Given our experience with system calls, in particular timing calls, in AC++ it is conceivable that a single user could inadvertendly cause the whole system to perform poorly by overloading a gateway in some fashion. It is thus clear that some effort needs to be put into monitoring the gateway performance, and if necessary automatically kill certain user processes. In the long term it might be possible to find migratable solutions to most system calls that could be incorporated into the MOSIX kernel. Alternatively, one might consider alternative arrangements of nodes as discussed below.

5 Architecture Variations

The MOSIX architecture described so far is based on the idea of small number of "gateway" or "login" nodes and large number of "worker" or "data and compute" nodes.

Any architecture that has a small number of entry points is inherently succeptible to developing bottlenecks, especially if rogue or inexperienced users are not prevented from hogging precious "login" node resources.

We can in principle turn the architecture "upside down" and allow login to all worker nodes, and have only a few file server nodes that have access restrictions. Those file servers would be the ones with the gigabit connection and would serve /cdf/home /cdfsoft disks. In addition, each worker node could have some amount of disk space reserved for user data. Each user could possibly have significant amount of disk space spread over a couple of worker nodes. A load averaging program as it is in use at CERN as well as SLAC [10] would asign a worker node statically to a user at login.

A second, possibly more attractive option would be to replace the idea of the login node by the user desktop. MOSIX allows configuration of a node such that jobs migrate away from but not to a given node. In fact, that's the way the gateway nodes in the MIT MOSIX cluster are configured at present. One could thus imagine all Linux desktops in B0 annex that are anyway maintained by CDF task force to be running MOSIX kernels. None of the "data and compute" nodes would need to allow user login. Each user could thus only hogg their own resources. Any concern about compile performance scaling would be moot as users would compile on their desktops. A user unsatisfied with the compilation speed would presumably just upgrade to a more powerful desktop. At the same time every user would have full data access to all PAD data from their desktop, and have the full cluster CPU resources available for interactive as well as batch processing.

While this last option is probably the most attractive, it has significant computer security issues. The Mosix kernel is likely to be quite vulnerable to hacker attacks if process migration were to proceede on the public LAN. The MIT group has been operating a MOSIX cluster with 10 worker nodes and 3 gateways for 6-9 months. And we have verified the scaling behaviour of MOSIX in principle up to 20 worker nodes per gateway. We thus consider the architecture variant with few entry points and many worker nodes sufficiently well tested to deploy a 30-40 node cluster for CDF. The more desirable options, especially "MOSIX on the desktop for everybody" is probably a bit more of a stretch at this point.

6 Conclusions

We have demonstrated that the CPU performance of a MOSIX cluster scales lineraly for up to 40 CPU limited jobs started from a single node in a cluster of 20 Dual CPU nodes.

In addition, we see roughly linear scaling behaviour for Edm_ReadWriteSeqRoot_byTBuffer up to 30 jobs reading from 30 different disks. The observed aggregate I/O for 30 jobs is 20% lower than expectation based on I/O performance measured locally using dd. We suspect this to be a feature of MOSIX rather than the CDF software. It is thus clear that a large scale MOSIX cluster will require a batch queueing system both for implementing fair share policies, as well as load balancing or limiting of the total number of jobs started on a given node.

We tested important characteristics of a subsystem of what one might want to consider for a Central Analysis Facility of 100 worker nodes.

Assuming modern EIDE disks with I/O bandwidth of 30MB/sec one might reasonably expect such a farm to have an aggregate bandwidth of roughly 40Gbit/sec.

Thus pure CPU and I/O considerations make a CDF MOSIX farm system feasable. In order to clarify operational and maintenance issues further investigations and a larger prototype are needed. Whether or not one would actually want to implement a large scale MOSIX CAF depends to a significant extend on the CPU/IO ratio for realistic user applications.

We would like to thank the computing group of the MIT Laboratory of Nuclear Studies for providing us with the 20 node cluster we used for our benchmarking. In particular, we'd like to thank Kuba Tatarkiewicz and Maynard Gelinas for their support.

References

[1] The 120 dual PIII 1GHz nodes recently purchased to complete the L3 farm were priced at less than $1200 per node.

[2] Benchmarking Report of CDF CAF computing review, CDF note 5743.

[3] A. Korn, Ch. Paus CDF Note 5707

[4] http://www.promise.com/

[5] http://www.promise.com/Products/Default.htm#ADAPTERS

[6] http://www.pcisig.org/

[7] http://www.mosix.org/

[8] See: for example the 3Ware Escalade 7000 Series http://www.3ware.com . The benchmarking is courtesy Patrick Schemitz, using an Escalade 7410 controller. He also benchmarked an 8 disk RAID5 array reaching read speeds of 50MB/sec.

[9] http://www-cdf.fnal.gov/upgrades/computing/projects/edm/faq/UtilityPrograms.html

[10] http://wwwinfo.cern.ch/pdp/is/servicedef/PLUS.htm

[11] http://quirk.fnal.gov/code/stager