A Guide to Building and Operating a Large Cluster

Alan Silverman and Dane Skow

Version 0.1 September 2002

Table of Contents

1 Introduction

Recent revolutions in computer hardware and software technologies have paved the way for the large-scale deployment of clusters of off-the-shelf commodity computers to address problems that were previously the domain of tightly-coupled SMP[1] computers. As the data and processing needs of doing physics research increases while budgets remain stable or decrease and staffing levels only incrementally increase, there is a fiscal and computational need that must be and that can probably only be met by large scale clusters of commodity hardware with Open Source or lab-developed software. Near-term projects within high-energy physics and other computing communities will deploy clusters[2] of some thousands of processors serving hundreds or even thousands of independent users. This will expand the reach in both dimensions by an order of magnitude from the current, successful production facilities.

A first Large-Scale Cluster Computing Workshop was held at the Fermi National Accelerator Laboratory (Fermilab, or FNAL), Batavia, Illinois in May 2001 to examine these issues. The goals of this workshop were:

1. To determine from practical experience what tools exist that can scale up to the cluster sizes foreseen for the next generation of HENP[3] experiments (several thousand nodes) and by implication to identify areas where some investment of money or effort is likely to be needed;

2. To compare and record experiences gained with such tools;

3. To produce a practical guide to all stages of designing, planning, installing, building and operating a large computing cluster in HENP;

4. To identify and connect groups with similar interest within HENP and the larger clustering community.

Computing experts with responsibility and/or experience of such large clusters were invited, a criterion for invitation being experience with clusters of at least 100-200 nodes. The clusters of interest were those equipping centres of the sizes of Tier 0 (thousands of nodes) for CERN's LHC project[4] or Tier 1 (at least 200-1000 nodes), as described in the MONARC (Models of Networked Analysis at Regional Centres for LHC) project at http://monarc.web.cern.ch/MONARC/.

---

[1] Symmetric Multi-Processor

[2] We should note here that we use the word clusters only loosely. Clusters come in many shapes and sizes and have widely differing architectures. Some are very closely linked, sharing for example a dedicated interlink for high-speed inter-node communication to main cluster cooperation. Others may use only relatively low-speed interconnects to maintain some minimal form of inter-node communication (for example to ensure failover if one node fails). And there are many variations between these two extremes. The key factor for consideration in the present discussion is that the installation is planned, built and operated at some level as a single entity.

[3] High-energy and Nuclear Physics

[4] LHC, the Large Hadron Collider, is a project now in construction at CERN in Geneva. It is expected to come online during 2007.

The attendees came not only from various particle physics sites worldwide but also from other branches of science, including biophysics and various Grid[5] projects, as well as from industry. They shared freely their experiences and ideas and the material collected by the convenors and offered by the attendees, including the full proceedings is available at the web site

http://conferences.fnal.gov/lccws/.

This "Guide to Building and Operating a Large Cluster" is an attempt to encapsulate the discussions. It is intended to describe all phases in the life of a cluster and the tools used or planned to be used. It may not answer all the questions about clustering but at least the questions should be included here with some pointers on where to seek the answers. This guide should then be publicised (made available on the web, presented at appropriate meetings and conferences) and regularly kept up to date as more experience is gained.

Being that the material in the Guide is extracted and assimilated from notes taken during the Workshops and from other sources, it is not claimed to be a complete review of the field nor of all possible solutions to individual issues. Further, errors may have occurred in the transcription of our notes to this Guide. To this end, we commit to correcting any inadvertent errors appearing in or major omissions from the Guide and we encourage our readers to contact us with any such remarks.

[5] See for example http://www.gridforum.org/

2 Cluster Design Considerations

The very first question to ask is "is a cluster actually the best, most cost-effective, most efficient tool for the computing task in hand?" For this, you need to answer some or all, certainly most, of the following questions. The answer may not be clear: perhaps a mainframe is the most efficient but not the most cost-effective solution. Perhaps you already have all or part of a cluster; can you, should you, use or expand this cluster to emulate a mainframe for this particular application?

2.1 What are the characteristics of the computational problems?

2.1.1 Is there a "natural" unit of work?

Can the target application be broken down such as to tailor the cluster components to the application. For example, if the task is to analyse output data from an HEP physics experiment, one could break this down into the needs to analyse individual events. HEP thus can exploit inexpensive mass market components and has much in common with data mining applications, Internet Service Providers, e-commerce, data marketers.

If the task is to produce a weather model, one may require a supercomputer. To take the two extremes, HEP needs High Throughput Computing, weather forecasting needs High Performance Computing.

2.1.1.1 Executable size

2.1.1.2 Input data size

An analysis job for an HEP experiment typically accesses large data sets but individual records are not large. Thus I/O rates are not large, especially compared to the running time of the jobs. And these data sets are read only, the output data goes to different files that are usually much smaller than the input files. On the other hand, I/O rates for jobs collecting data are much higher; in the case of the newer generation of HEP experiment, they strain I/O capacity to its limit as they read data from the detector, perhaps via a fast network, and collect the result on long-term storage, typically tape.

2.1.1.3 Output data size

Is the output file size known or can it be estimated in advance? If not, and this is especially common if there is a mix of jobs, then a large amount of scratch or buffer space will be required along with well-publicised - and hopefully automated – methods for harvesting space after some reasonable time. Some job schedulers request output file size as one of the job parameters at job submission, but the cluster designer should foresee the overall expected or allowed job mix to create a large enough scratch pool such that the lack of this resource is not a bottleneck. After all, the cost of temporary storage, usually disc, is not the most critical factor.

If the scratch pool is common, shared by multiple users, there must be a unique file naming convention in place. (Example? CHECK)

The danger of job runaway should be considered. Is there an output size beyond which a job should be aborted to avoid it allocating too much resources for itself? Is this a function, perhaps equal to or 10% above the given value at job submission or is it an absolute value in size? Some job schedulers include this feature (e.g. LSF? CHECK)

2.1.1.4 Single job and inter-process communication needs

Once again, taking the example of cluster to analyse HEP data, the analysis of each event is independent from each other. A typical scenario is to determine the CPU (and memory and I/O) needs for a single event, decide how many jobs (which process one event at a time) will be permitted per node (or per CPU) and the result indicates the desired node configuration. This can be considered as a form of trivial parallelisation.

2.1.1.5 CPU requirements

These may be difficult to estimate and liable to change over time. Can the CPU be upgraded in place or does this require changing more of the configuration (backplane, memory). And it is normal behaviour for users to expand their needs to fill any available CPU capacity.

2.1.1.6 Memory utilization

The goal should always be to avoid memory swapping or paging, except for rare cases, certainly not during normal stable production running.

2.2 How broad a range of these parameters will be allowed?

Are you able to influence your users in order that they modify some of these parameters in order to make better use of the cluster resources? In other words, can you change the compute model to make better use of the cluster environment?

A cluster dedicated to running batch jobs for a particular task (physics analysis) for example is far easier to configure than one for general interactive use where the overall workload is referred to as "chaotic" in the sense that it is hard to predict it in advance and in that at any one moment it is going to be made up of a jobs with a wide variety of CPU, memory and I/O requirements.

If the cluster is dedicated to batch jobs, do you need to allow any interactive logins? At first thought, the answer is probably no. But you may prefer to permit user group administrators to login to check job and system status rather than have them call you or your operations team for the latest situation report. Further, a user may request access in order to debug a failing job, in particular when he or she finds it impossible to reproduce the failure on another cluster and yet it crashes consistently on your cluster.

2.3 Are the variations schedulable?

In some environments it may be possible to schedule varying workloads: a cluster may be reserved for interactive use during prime shift (perhaps with a background job queue to "soak" up any unused CPU) but given over to batch processing overnight and at weekends. Obviously one cannot change the configuration as the load switches back and forward but the cluster may be tailored to fit one load better than the other, or perhaps a compromise needs to be used to cover both situations.

2.4 How rapidly does the user community turn over ?

2.5 What are characteristics of the performance requirements?

2.5.1 Job throughput/turnaround

Do the users require predictable turnaround or interactive response? If the nodes are shared, it is going to require considerable investment or clever software to ensure this.

2.5.2 Availability

Is high availability essential or merely desirable? If the former, what kind of failover or failsafe scheme is affordable? Is a backup centre essential, is there a disaster scenario? This is covered in more detail below.

2.5.3 Ratio of I/O to computation

In the analysis of HEP physics processing, the I/O rates are low compared to the CPU needs.

2.5.4 Absolute CPU requirements

Does the calculation require powerful floating point processing? In HEP, perhaps surprisingly, the answer is not really, SPECint performance is much more important.

Is there a target processing time for the job? In HEP, experiments may set targets for the processing of each event. As described earlier, HEP data processing tends to make use of trivial parallelisation. On the other hand, when scaling up the jobs to process a given data sample, the aggregration brings its own problems (data sharing and access, scheduling, monitoring). The next generation of experiment is going to need resources which will be beyond any single centre; hence the need for computing grids.

2.5.5 Absolute I/O requirements

Taking an extreme example, experiments planned for the LHC at CERN, will seek to record data at rates exceeding 1 GB per second for sustained periods. The date will be sent from the experiments over fast networks to the Computer Centre where it should be recorded on to long term storage media (tapes). The cluster I/O rates thus need to be twice this, input from the network and output to the tape drives.

2.6 What fault tolerance requirements are needed?

High availability clusters are inevitably more difficult to design and more expensive to implement and operate. You must be really sure this is justified. There are various kinds of high availability:

- Job failover – the job(s) on a failing node fail-over to another node and continue from where they stopped. This is probably the most difficult scheme to design.
- Job restart - the job(s) on a failing node fail-over to another node and continue from the last checkpoint (or from the beginning)
- Redundancy – the job(s) run in parallel on several nodes and there is a “vote” on the most consistent result – certainly the most expensive in CPU cycles. On the other hand, some applications such as licence serving are well suited to this scheme.

2.6.1 Component failure in cluster?

2.6.2 Job restart/checkpointing/fail-over?

Condor offers checkpointing as a feature, but only if jobs are linked against special Condor libraries for checkpointing. If the job includes code or a private library which cannot be linked against these Condor libraries, for example Objectivity libraries, then the job may not be a suitable for checkpointing and hence not a suitable candidate for scheduling by Condor. (CHECK)

2.6.3 Service interruptions due to membership/role changes?

2.6.4 What configuration restore features are needed?

2.6.4.1 Backup of user areas ?

2.6.4.2 Backup of system areas ?

2.6.4.3 Backup of services and/or states ?

2.7 Cluster Interconnect

2.7.1 High speed interconnects

These include SCI and Myrinet (CHECK)

2.7.2 Low latency interconnects

These include devices based on VIA (virtual interface architecture such as Servenet II or clan (CHECK)

2.7.3 Wide area clusters

We use this term to cover two cluster schemes in common use – Condor and seti@home. The first is a scheme to use spare cycles on existing workstations. Such nodes declare themselves and their characteristics to a master node that also maintains or has access to a queue of jobs. There is a form of "match-making" and jobs are dispatched to nodes as appropriate. At any moment, the owner of a node can "take back" the node for his or own work and the running job is suspended and perhaps returned to the central queue for re-scheduling (from the point of suspension) elsewhere.

The second method, seti@home is somewhat similar except that there is typically only a single or a small number of jobs which is/are made up of discreet calculations or actions. While running, the job returns its results to a central collection point where they are aggregated. Taking back a node in this case merely marks where it has reached and suspends or cancels the job. Re-activation or re-starting simply means restarting from where it left off.

2.8 What are the characteristics of the budget available?

You need to estimate the total cost of ownership from initial investment, through maintenance and operation to final decommissioning. In charting a long-term plan, you need to make plans for eventual upgrades during the course of the life of the system. You also need to be aware of when it no longer makes sense to upgrade because of technology creep. Such future planning is more and more difficult because of the fast pace of technology change but past experience (yours and that of colleagues) can help.

Is the budget from a central fund (as in most sites) or does it come directly from the users. The NERSC centre in Berkeley charges users for their computer use and this makes up a large part of the investment funding. There are two variations of this: where the users' income is used directly to purchase systems that are then dedicated to that user; or the more flexible alternative, that used at NERSC, where the users' funds go into a central pot and the user is simply guaranteed the CPU power contracted for. NERSC further augments their scheme by offering to schedule any idle time to paying customers over and above their contracted allocation.

2.8.1 What is the initial investment?

2.8.2 What are the ongoing maintenance and operational charges?

Almost all new systems come with a warranty period during which there is a basic maintenance cover. This may be at a sufficient level from the point of views of ongoing service offered to the users. Typically, the larger the number of systems, the lower level of cover needed – for clusters of thousands of processors, in-site replacement within a short time is typically sufficient. Other clusters may need higher service levels, within a few hours or even more stringent.

Once the warranty period has expired comes the question of whether a maintenance contract should be established. On the other hand, when the technology is fast-moving and the warranty period is longer than 3 to 5 years, some sites simply let warranties expire and run the systems without cover until they die or become so obsolete that they are replaced naturally.

2.8.3 What is the annual budget available?

2.8.4 What is the maximum lump sum available?

2.8.5 Are there advantages to large/long-term bid methods?

2.8.6 Is there a difference between hardware, operations, and personnel cost accounting?

2.8.7 What is the budget cycle for planning futures

2.8.8 What is the budget cycle for responding to "found money"

2.8.9 What sensitivity/advantage is there to vendor budget cycles?

There may be no change in pricing from one moment to another but it is a fact that discounts are often found towards the end of a financial quarter or financial year as the salesmen work to meet some budget/sales threshold. It certainly does no harm to discover the financial year calendar of your supplier and to be aware at least of the possibilities. But it will not work every time so use such timing with care, keep it for when you really could make the best use of it.

Another, less contentious, method of getting the best value, is not to buy the latest and greatest system. Studies have shown that you often get better value for money (more CPU power for the price) by buying systems of the previous cycle. For example, if the fastest CPU speed is 2GHz, and you can afford N of them, perhaps you can purchase 2N systems of speed 1.5GHz and if your application can take advantage of the extra number of processors (for a high-throughput workload), then you have a net gain.

2.9 What Design Tools are needed?

2.9.1 Are modeling tools required/sufficient/available ?

No testimonial evidence yet that MONARCH has proven useful in design upfront. [*** **IS THIS REALLY TRUE. Ask Harvey or ? ***]**

Still validation tests and hopes.

2.9.2 How is experience fed back into the system ?

2.10 How will Expansion be accommodated?

3 Hardware Issues

3.1 How is hardware selected?

The options here range from buying pre-configured clusters through buying commodity processors (also known as COTS – Commodity Off The Shelf) which you then configure yourself into the required cluster down to building the entire configuration yourself from the components.

In pure hardware acquisition terms, it is usually cheaper to purchase individual components (even at the box level but even more so at the board level if you are buying PCs). However, you then need the human resources and skills, and preferably the experience, to assemble these into a working configuration. If you already have these, or part of your overall budget can be allocated for engaging the required staff, then "do-it=yourself" may be the preferred solution for you.

If you decide to buy at the processor level, it is general consensus that usually the best strategy is to buy at the knee of the price/performance curve and get the benefit of external stability testing and price discounting. Even if this means an upgrade within 6 months of CPU/memory, this was usually a good strategy.

Are all the nodes of one architecture? Even if you decide to base the main processing power of the cluster on one architecture, there may be a case for a different architecture for nodes with a dedicated task, for example I/O nodes have different needs to job execution nodes. Different architectures in one cluster increase, often significantly, the management overhead. Is this justified by the better throughput you expect to achieve?

3.1.1 What specifications are used?

3.1.1.1 Physical Space Issues

Is floor space at a premium? If yes, then look into acquiring 1U systems or even the more recent systems which accommodate multiple processors in a block, the so-called Blade systems. Be aware however that denser racking and packing more CPUs into a box leads to greater power and heat problems to be solved.

In general dual or quad CPU boxes offer space saving as well as diminishing management overhead. (Management overhead is typically more dependent on the number of systems, not the number of CPUs). However, multiple-CPU systems do not normally run at the fastest speeds in the market. This could be a problem if your main task absolutely requires the fastest speed in each CPU as opposed to aggregate speed across the cluster.

3.1.1.2 Binary code compatibility?

Is there pre-existing code with which the configuration must be compatible? Hopefully you have access to source code that could be rebuilt for the target architecture but this may not always be the case, especially if some part of the application relies on existing commercial software or legacy code.

3.1.1.3 Number/speed of processors?

Note that it is usually not essential that all CPUs in a cluster operate at the same speed, for example you might upgrade the cluster during its lifetime by adding new nodes and these will usually be

faster than those purchased earlier. However, especially for parallel applications, the overall throughput may be determined by the speed of the slowest node.

When building the configuration, consider the current price curve for chip speeds. Buying more systems with speeds at the knee of the curve may be more cost-effective than buying systems equipped with the fastest chips on offer.

3.1.1.4 Size of memory

There should be sufficient memory to avoid process swapping, after all memory is much cheaper than disc space and much faster to access. However, it may not be easy to estimate the optimal value so the initial configuration should feature sufficient expansion possibilities.

Most systems have a finite and fixed number of memory slots and it is recommended on the original purchase to acquire the target memory size using the largest memory modules available, leaving a maximum number of free slots for future expansion.

3.1.1.5 Number/size of disks

Do you require local storage? Is the operating system installed on a local disc or is the system booted from a central server? Even in the latter case, you might wish to ave a small local disc for certain operating system files, libraries, temporary buffers, caches, swap space, etc.

Do the applications require to access local data? Many schemes exist to buffer I/O data, especially data on magnetic tape, to intermediary temporary disc caches.

If there are local discs, there may be a choice of connection (on PCs) between SCSI and IDE buses. The former may be faster but it is more expensive.

3.1.1.6 Motherboard features

Can the system box be upgraded with faster chips over time or would this imply the need to also upgrade memory or motherboard or something else.

Comment here about modifying BIOS for serial console connection.

3.1.1.7 Power Supply

Often the most critical part of the configuration with respect to continuous operation. It is a known fact that after a power interruption, a certain number of components will not restart and the sources of such power interruptions should be minimized. We cover this later in discussing power supplies for the computer centre. With respect to individual systems, beware of selecting cheap power supplies in search for short-term savings. Power supply failures are probably the source of more system failures than any other source.

3.1.1.8 Network connection

Do the nodes need to be interconnected or attached to a single segment? This depends partly on the degree of clustering. If they need direct interconnections, this generally means some kind of central switching device.

3.1.1.9 Monitoring/Management connectivity

If you a configuring a large cluster, you should consider at an early stage how you intend to manage and monitor these. You certainly do not want a display screen attached to each console port. At the very least, you could consider some multiplex scheme connecting the console ports to a console concentrator (e.g. Polycenter – GET a REF). Another popular option, common on modern PCs, is to acquire or patch a BIOS to connect its serial (COM) port to Ethernet and enable remote monitoring and/or remote management.

A third scheme, used in Chiba City (CHECK and ref) and in VACM written by VA Linux (CHECK) and available from Sourceforge (ref) connects a rack of systems to a single console controller for management purposes. VACM in such cases then accesses each motherboard directly, or a special board connected to the motherboards, to get its monitoring data and pass its commands. Chiba City's scheme is hierarchical (CHECK and amplify).

3.1.2 How are vendors selected?

Working relationships with vendors are very important to "peace in the house" but there is no clear method on to achieve this goal. It clearly takes multiple years to develop (3+) and a steady expenditure of money in an amount significant to the vendor helps.

However, should you concentrate all purchases on a single vendor? While seemingly good for the relationship with that vendor, and helping to reduce the variety of systems, and vendor relations, to manage, there are some potential drawbacks. These include

- being tied to one architecture; at least in the HEP world physicists like to be able to prepare and run their codes on multiple architectures to check for strange effects at very high accuracy

- being tied to a single vendor with the risks if that vendor begins to take the customer a little for granted in the form of less advantageous prices or less than top quality support; also a risk if that vendor runs into trouble commercially.

For such reasons, many sites prefer to have systems of at least a second architecture, even if just a test and development system for users to check their codes. If you decide to restrict your systems to one architecture, then good practice is often to pre-qualify more than one supplier and have these tender for each subsequent purchase. If you choose to include equipment from multiple vendors, you may need to commit extra resources to managing these – including for example having dedicated experts for different architectures.

3.1.2.1 Do you pre-qualify a list of vendors

Some sites, Fermilab is one example, pre-qualify a small number of vendors for a given period of time, typically 1 to 2 years. At this frequency, the qualification procedures can be quite lengthy and detailed. Fermilab's procedures include a set of formal benchmark suites. Other sites, CERN is an example here, group orders together and issue a new invitation to tender each time. To perform detailed qualifications on each order in this scheme is only justified if individual orders are of significant size; it is still probably more work than remaining with one qualified vendor. And subsequent support issues are likely to be complicated by having to deal with multiple vendors. This is the trade-off for having the most competitive offer for each order.

3.1.2.2 Do you bid a total package and buy "turnkey"?

Is there a market for turnkey installations in scientific research environments? Some firms make a lot of money from such business in the commercial marketplace, especially for PC/Linux systems where there is no "home" vendor such as SUN or HP or IBM. SLAC and BNL have experience using a Linux support firm (VA Linux in both cases) but even they used only part of the services offered and the firm itself has since ceased this particular activity. Most other sites we have contact with have decided to acquire the hardware and install the systems themselves, either using methods developed in-house or picked off public domain sources on the web.

3.1.2.3 Hardware Refresh Issues

The component technology of today's computer systems is in constant progression. This must be built-in to the planning. If the cluster is always growing, perhaps simply adding the latest and greatest will suffice. [This applies even if the local purchasing policy is to acquire systems at the "knee" of the technology curve.] But, depending on the degree of clustering, this could involve its own problems of compatibility (do the different generations work together) and conformance (will the same software environment be applicable).

Adding extra memory or disc space may be simple; adding faster CPU nodes is also usually straightforward; substituting the latest chips in installed CPUs may be less so, and will involve a lot of system intervention and effort. An alternative is sometimes referred to as "trickle down" where older systems are recycled to the less demanding tasks (licence serving, statistics gathering, alarms). Depending on the possibilities to add a graphics display, older servers may supply the computing needs for quite satisfactory simple desktop systems when equipped with a display.

3.1.3 What method of procurement is used?

3.1.3.1 Competitive bids on what frequency?

3.1.3.2 Do you require split winners?

3.1.3.3 Do you develop preferred supplier?

3.1.3.4 Buy or Lease?

Although most clusters are today based on purchases, it is not unknown for them to be built from leased components. Advantages of leasing include lower capital costs and built-in technology refresh (but be aware that the faster the refresh rate, the higher the cost). Disadvantages include a higher overall cost measured over the life of the lease period. There are many other pros and cons but if you decide to investigate leasing further, be sure to include an early termination cost. At least one HEP lab was forced to stay on RISC technology for about 2 years later than desired because of long lease deals, and thus 2 years later in benefiting from the lower price/performance of PC-based farms and clusters.

3.1.4 Should you build them yourself?

Some sites reported that it was still monetarily advantageous to construct PC yourself (even in high density configuration) and that the labor was tractable. Will there be any difference in maintainability of such systems? And is the intellectual challenge still there when one needs hundreds or thousands of systems, all identical?

If you intend to build your own cluster, you must at an early stage produce both a physical layout drawing and a wiring map and carefully check these against the assigned floor space. You also need to check that the required facilities (power distribution outlets with sufficient power, network connection points, cooling power) are available and include some reserve for future expansion.

3.2 How is repair provided?

A frequent problem in dealing with vendors is to "prove" a problem and establish bona fides/trusts between both partners. This has non-negligible value at all sites and is often a reason for preferring "local" vendors.

Some repair contracts, especially for commodity equipment, stipulate one-for-one replacement. In a fast-moving technology, this may not always be so simple. Unless the supplier maintains stock of exactly the same model of equipment installed (and this could be expensive over a number of years) it is entirely possible that the replacement is in fact a more modern version. It may be simply a later model of a chip or board or interface. But it may be enough for some process like image re-installation to fail because some vital piece of the kernel software needs to be re-built or updated.

3.3 Computer Centre Issues

3.3.1 Power

This is cited as the number one computer centre concern by most computer managers. Not only must there be enough for planned and future capacity, it must be stable, cheap and, if affordable, backed-up. There must be plenty of distribution points across the floor space, with local breakers to isolate areas (often racks). Determining how much power in watts per square meter may not be the best measure as it does not account for the fact that some equipment will require more or less power than the average. An alternative proposed by SUN is "rack location units" – how much power each rack would need.

Universal Power Supplies (UPS) with enough capacity for large centres are very expensive. But every time there is even a short power interruption, some equipment will not restart. Experience will demonstrate the frequency of power interruptions and how these affect your installed systems. Also, how long does it actually take after a power interruption to return to a full production state? What is the cost of lost production time? Perhaps smaller UPS for specific services or individual racks may be a reasonable compromise.

Related to power is the need for a clear description of what to do in the case of a power cycle. Power outages are inevitable; even with a UPS or battery backup, you will suffer or need to schedule a power down cycle. If you can schedule it, you need to decide the correct order to stop systems. More usually, you do not get the chance and the power restore is where to concentrate the effort. If power is restored, do you want systems to restart automatically? Probably not – apart from the randomness of this and the risk of missed dependencies, there is likely to be a huge power spike if power is

simply restored to a large number of systems at once. Foresee breakers which trip on power down and which are reset by hand when ready; or set systems not to restart automatically.

Build a power restore sequence with basic services starting first and dependent systems in the correct order. And don't forget to test it from time to time, although it will be extremely unpopular to schedule a time for this! Use the (hopefully) occasional opportunities which inevitably come your way.

3.3.2 Cooling

A good temperature control scheme is also essential. Vendors will supply required specifications and you need to aggregate these over the planned and future capacity.

3.3.3 Floor space

Is there enough floor space for the planned and possible future capacity? It should be possible to cite particular systems together for ease of working, allowing for expansion of these over time. For example, all network services in one part, file systems servers in another part, etc. This modularity will help you as the centre builds up.

Don't forget to check floor weight limits and false floor requirements. And, if you're in an earthquake zone, check building and floor stability such that racks do not fall over in the event of seismic activity!

3.3.4 Network connections

You need to provide the required connectivity between all the systems in the centre and to the outside world.

3.3.5 Security Access Control

A computer centre needs a good access control scheme in order to protect physically the systems from interference, deliberate or accidental. The more sensitive the application, the more stringent the scheme.

3.4 What commitment timescales to do your procurement policies tie you to?

Can you declare how long you expect to have to maintain your cluster? This could define how often you need a technology refresh, either replacement or in-situ upgrade or simply more of the same. If you have leased your equipment, make sure the lease agreement has enough flexibility not to commit you to out-of-date technology for too long a period. One major HEP site felt itself excluded from the price/performance advantage of PCs for some years because of long-term lease contracts on a certain RISC architecture.

If you rely on vendor support, how long will the vendor commit to that support for your installed systems, both hardware and software if that is appropriate?

3.5 Acceptance Tests

Few sites surveyed perform real acceptance tests as they have come to be understood from the days of mainframes. This may be due to the low price of individual elements which can be isolated and

replaced during operation and also partly to the pressure to get systems installed. The NERSC site in Berkeley ask their suppliers to perform some form of acceptance tests before shipping systems. Fermilab has some burn-in tests which it performs. Other sites do not appear to systematically perform such tests but rely on the warranty period to cover serious problems.

3.6 Inventory Control

The scale of modern clusters has reached the level where an inventory is essential. Also, large clusters are "living" objects: individual nodes change as they are upgraded, repaired, replaced. Systems must be labeled and there should be a "map" of the whole centre. Ideally, one would need some kind of automated location services (GPS on a card, network maps, blinking lights). Some IBM PCs have built-in radio code that can be read with a scanner. Labelling each system box with a bar code would be another option.

At the very least you need a form of database which is easy to modify and strict instructions to staff who work with the hardware boxes to enter every change. The rule to remember is that physical location is nearly as volatile as configuration, perhaps more so.

4 Cluster Design

4.1 What usage case is anticipated and how realistic is the prediction?

Why cluster, why not a large mainframe, why not a collection of independent processors?

You must first and foremost understand the projected workload, the usage case(s). For example, in HEP data processing, you are mainly concerned with extracting data from independent events and gathering statistics. This is ideally suited for a farm or cluster of independent processors able to read input data from a data source (magnetic tape or intermediary disc storage) and aggregating the results. The form of clustering may be very loose in such cases. The following stage, data analysis, calls for more CPU than I/O capacity but still benefits from the price/performance of commodity (PC) systems.

In the bioinformatics genome environment, a typical analysis job consists of a M*N object comparison used to build the physical map of a genome. Here there is more parallelisation than in the HEP usage case. A cluster is also well-suited to this application, this time with a higher degree of interconnection.

4.2 Who will develop for this cluster and how are their needs incorporate?

4.3 Who will operate this cluster and how are their needs incorporated?

4.4 What are the design principles and assumptions?

Answers to the following questions affect the target configuration:

- do you need volume management of disc space?
- what is the disc layout – standard, striped, mirrored, RAID, etc?
- is I/O storage local to individual nodes, or hosted by specific nodes and shared to the rest of the cluster or served by remote hosts?
- are the nodes directly interconnected (does not scale) or linked by a central hub (switching) or by a cross-bar
- do your applications need a fail-over strategy? Even more stringent, do you need a fail-safe environment for your applications?
- is there some node naming strategy to be followed (recommended)?
- is there a local security policy and is it sufficient for this cluster?
- is there a local file backup and restore policy and is it sufficient for this cluster?

4.5 How is the cluster design/operation reviewed in light of changes in the base technology?

4.6 What are the assumed switching costs?

4.7 Is the design scalable?

5 Data Movement

5.1 What are the sources of data?

5.1.1 NFS

NFS is universally disparaged as a central data service for larger clusters. Many sites use it quite successfully but the largest sites complain that it really does not scale to large numbers of servers and clients. And this is still true in the latest releases of NFS 3.

5.1.2 AFS

AFS is used at many large sites without complaint for home directories although it is disparaged for bulk data access. The main reason appears to concern its caching algorithm. One of AFS's advantages is that the file is cached in client memory; when a file is read, treated and written out and there is no further access to it, there is no advantage to using AFS. Further, there is a disadvantage in that it passes through the local memory cache, possibly flushing out files which will be re-read and thus will need to be re-read from the central server.

5.1.3 PVFS

The Parallel Virtual File System was developed at ?????????? (CHECK)

5.1.4 Stager Access

Most successful methods seem to be remote file stages through some stager access method (RFIO, rcp, encp, etc.) [*** put links or refs for these ***] Staging can involve writing to a memory cache but more usually to a fast-access disc pool. Staging here can offer not only local caching but also file sharing, although there needs to be some locking scheme if the files can be modified or deleted.

5.1.5 Where does the data flow?

5.2 What intermediate storage is required ?

5.3 How do choices change with parameter or technology or usage changes?

For example if, as predicted by some people, disk becomes cheaper than tape?

6 Operations

6.1 Usage

6.1.1 What are characteristics of the user community?

6.1.1.1 How many different computation types are represented?

Do you need to cater for a range of job types (heavily interactive, long number crunching, large memory, very many short tasks, etc)? If the answer is yes, you may need to configure and tune the cluster for the most "significant" where there could be many definitions of "significant" – most often, most resource-hungry, the source of most of the funds, etc.

If the load is batch-oriented, you can perhaps make use of batch queues to ease the tailoring needed. You could define different queues with the characteristics of the different types of job, tailor certain nodes for each type and only target corresponding queues at their respective nodes. Be aware however that this might lead to unused CPU cycles if one or more queue is empty while jobs in another queue are waiting.

6.1.1.2 How many independent groups are represented?

In a similar manner to that for defining different queues for different job types, you can define queues per user group and target each individual queue for a specific set of nodes according to need, funding or whatever criteria is appropriate for you. Once again be aware of the potential for wasted capacity unless the groups agree to share resources to avoid this.

6.1.1.3 How many total users of the system?

6.1.1.4 What level expertise is available?

6.1.2 Development Feedback

Is it understood how for example stability problems get more operations resources ? Design changes ? (Remember BNL experience here) [*** what is this ref? See BNL notes ***]

6.2 Computer Operations

Will there be a team of computer operators available or is there some form of "lights-out" surveillance? Operators may be essential for certain activities (tape mounting for example, unless all tape drives are housed in tape robots) or for security reasons. Operators can also be useful to assist users with simple questions, forming some first-line support for example.

Lights-out operations can be achieved by acquiring a commercial tool but this is usually very expensive and usually only found in major commercial enterprises. In the research and academic environments, it is much more common to find simple tools combined in such a way as to detect, and perhaps recover from, simple incidents and to call for help for more serious situations.

In large centres, there should be well-documented procedures for operations, how to recover simple errors, who to call for more serious ones; how to configure individual components; who are the users; etc.

There may require to exist a disaster plan, in particular how to close down the centre in a controlled manner – assuming the operations team is given enough time for this. In some centres, usually those protected by a UPS, operations teams are given instructions on which services should be closed in priority order to ensure as smooth a restart as possible. Typically, file servers and database servers should be among the first systems to be closed down. For example, it is usually better to have batch jobs failing because of failed data access than have files or databases corrupted because they were open when the node crashed for lack of power.

6.3 Management

There is a vast range of management tools available, how to choose? One solution, as proposed by the Chiba City project at ANL (CHECK) is adapt the traditional UNIX philosophy of building tools from small blocks, embodied in Chiba City's Scalable UNIX Tool kit, a set of parallelised versions of the usual UNIX tools plus a few new commands to fill gaps. Another tool suite is OSCAR, from the Oak Ridge National Lab, a collection of tools for basic cluster operations.

Especially when the clusters are configured from mass-market components (in other words from PCs), they tend more and more to be made up of large numbers of such devices. There is a very great need to marry these aggregates with inexpensive, highly scalable management tools in order not to produce a cluster which would effectively replace hardware investment costs with management overhead costs.

6.4 Installation

Many sites reported buying HW installation services and/or integration services. Almost no one reported using software installation services for final software installation (perhaps for burn-in testing).

If there are few "tricks" to shorten the time needed for the physical installation of the hardware, there are many schemes to shorten the software installation process. If you have large numbers of nodes to install, you should select one of the many public domain schemes which have been developed for this purpose. Examples include SystemImager (written by VA Linux and available from Sourceforge (CHECK and ref) and the etherboot and netboot projects (see http://etherboot.sourceforge.net/).

SystemImager works by creating a "master" configuration and then cloning this from a central node. After installation, it can be used to keep nodes in step with each other as regards software environment. Care must be taken about scaling: typically a server node takes care of 20-30 clients, beyond that one can adopt a hierarchical scheme with "masters of masters". SystemImager clearly works best on homogeneous clusters where all the nodes are, for all practical purposes, identical. SLAC's Computer Centre quotes times of less than 1 hour to install 256 nodes (once the hardware had been set up and the MAC address mapping).

There are several others, each with its own features and drawbacks and each with its own proponents and supporters.

6.5 Testing

Once a cluster has entered production state, there must be some means to test and evaluate enhancements, new system software versions, new hardware, new production suites. If the production cluster cannot be taken offline for a dedicated test period, a test farm of ~10% identical

equipment is the ideal alternative. Most testing and development can be done on systems of this size although performance analysis and optimization may need the full-scale facility.

Is the acceptable failure rate well understood? People reported experience of tolerance of niggling issues at levels of 10s of machines that became mighty headaches at levels of 100s of machines. What level "DC frustration" is acceptable?

6.6 Monitoring

6.6.1 Commercial Tools or Build Your Own

It goes without saying that production computer systems must be monitored and there is a very large and developed market for commercial monitoring tools. There is also a vast range of individual tools in the open source world to "roll your own". And many computer science projects have and continue to address this area. So you have a first choice – buy-in or build or develop your own?

Be aware that many commercial tools in this segment of the market are large and expensive. They are consequently to be considered carefully, both in terms of initial investment and initial configuration but also in terms of ongoing support. And they may not always be as flexible to configure to your environment as you might expect. FNAL is one example of a site which surveyed the market and decided to build their own tool in order to better match their needs.

On the other hand, building your own tool is a major investment also, in direct manpower costs as opposed to software investment in this case. Sites such as CERN and IN2P3 discovered this and indeed the latter subsequently abandoned its plans to build its own and adopted that from Fermilab.

A point to remember is to make the scheme scalable, whether bought-in or home-built. And for the largest of clusters, when these get to thousands of individual nodes, does it need a hierarchical scheme? Does it make sense to monitor a cluster of computing and/or storage elements (as for example is planned for various Grid projects)? This leads into the theme of Fabric Monitoring and the area covered by the Work Packages 3 and 4 of the European Data Grid for example.

6.6.2 Monitoring Architecture

The monitoring architecture needs to be considered. There are many models to choose from[6] but most common is that of a collection of sensors reporting to a central monitoring broker which compares its inputs against a measurement repository and perhaps also a correlation engine. The sensors collect their data from local or remote monitoring agents on individual nodes or devices (e.g network switches). Sensors may include some intelligence to perform some event correlation and even fault recovery. Sensors and agents communicate via some defined protocol and many schemes are based on SNMP – the Simple Network Message Protocol[7]. SNMP also allows the setting of a

---

[6] Many references could be given here. A good starting point is the USENIX organisation, see http://www.usenix.org and in particular its conference site for presented papers or its publication site for articles published in its :login publication (CHECK)

[7] See for example "Essential SNMP" by D.Mauro and K.Schmidt, July 2001, ISBN 0596000200.

Also see the Net-SNMP project at http://net-snmp.sourceforge.net

local variable to alter the state of the device being monitored. Sensors and, especially, agents must consume minimal resources in terms of CPU and memory but may use local disc caches to collect the data before sending it to the central repository.

Results (alarms and performance statistics for example) are posted via a variety of display mechanisms such as dedicated alarm displays or formatted web pages, different users of monitoring needing different information and/or a different level of detail. Typically, users of a cluster want to know about the current state and performance of their jobs while operators and system administrators are more immediately concerned with the correct and efficient operation of individual nodes and the cluster as a whole.

When reporting and displaying statistics, not only averages or means are important. Maxima and minima can also indicate important events. For example when available memory goes below some value it may be time to invest in new memory. When usage of some licence goes above some level, it may be time to acquire some more or limit the number of processes.

6.6.3 Event or Performance Monitoring

We suggest the terminology of monitoring events where an event is a change of state. Faults are only one type of event. Normally you want to signal faults but you also need to monitor events some of which may include advance warning of a fault.

Performance monitoring involves tracking metrics of how the system behaves, with respect to both individual applications and the entire cluster. Typical metrics are system availability, system throughput and idle time. However, it is not always simple to define these terms. In a dual CPU node, is it still available if one CPU is down? How to measure system throughput in terms of useful cycles?

Event and performance measurements often overlap as at least some of the metrics being gathered may be the same although their handling is different. Performance metrics will be stored and graphed; events will be signaled, perhaps with some processing first. Also you have to recognise when metrics are no longer being collected. Is the node down?

Another question is whether and for how long to keep monitoring information. Do you want to perform trend analysis over some period of time, for example resource usage? Do you need to keep accounting information for users jobs for some period? Do you want record information about individual nodes or sub-systems in a “logbook”?

6.6.4 Alarms

How to handle alarms? You need to apply filters when multiple alarms occur, especially if they visibly are provoked by a common source – you don’t want operations flooded by 1000 occurrences of the same network failure warning for example. Nor do you want to receive the same warning from the same source every N seconds. These questions apply to both visible and logged alarms although you may wish different filters in each case.

Repeatedly people mentioned the need and value to benchmark against actual production code and to have monitoring tools that actually tested the operation of the facility (test jobs submitted through batch, test email, etc.) against these same benchmarks. Presence of services was viewed as sufficient at a pinch but much less desirable than test through operation.

6.7 Upgrading

People reported that upgrades usually took place within the first 6-12 months of operation of a machine. After that, a user community was well developed and hardware had changed such that it was usually best just to run out frozen.

6.8 Replacement/decommissioning

The typical planning schedule was 3 years, but there were several sites that reported also doing the optimization of infrastructure/support contract costs by compressing the number of boxes. No information on timescale when that compression makes sense in that metric.

6.9 Staffing and Training

Do not underestimate the staffing needs of the cluster. However, the more automation tools in use, the lower the number of staff required. Similarly, homogeneous clusters require fewer staff than those built from a variety of architectures.

The SLAC Computer Centre has invested a lot of resources into building its clusters to be very low in operations overhead by making use of many in-house developments and some public-domain tools to automate the routine monitoring and maintenance tasks. It has also reduced architectural diversity to only SUN/Solaris and PC/Linux clusters and it has measured a fall in human resource needs from 1 FTE person per 15 nodes in 1998 to around 100 nodes per FTE in 2001.

7 Resource Allocation

7.1 What resources need to be administered?

7.1.1 Instantaneous load?

7.1.2 What response interval?

7.1.3 Job predictability?

Do the users expect to have a guaranteed turn-around for their jobs? If yes, then you either need to dedicate nodes (usually inefficient) or implement some kind of queueing system, commercial or public domain.

7.2 Features of a job queuing scheme

The order of importance of the following list is dependent on particular local requirements. And the list is not guaranteed to be complete.

- If possible, it should operate in a heterogeneous environment, dispatching jobs to different architectures as directed in some way by supplied parameters if necessary.

- Especially for long jobs, it should inter-operate with system checkpointing/restart features and it should also offer user interfaces to local process checkpointing/restart features.

- It must be able to enforce job limits in respect to CPU time, clock time, I/O resources, memory, etc. If any of these overrun, the job must finish gracefully, reporting the reason in a log.

- It must offer a complete administration interface to the system managers enabling them to control every aspect of its operation and that of the scheduled processes for example forcing process checkpointing, changing process priority, modifying process limits.

- It must be configurable and scalable. Interfaces should be simple and intuitive and if possible graphical. A web interface is desirable. And any remotely-accessible interface must have an appropriate level of security.

- If the local applications include parallel applications, it must support either PVM or MPI or both. Local applications may demand one of these in particular.

- It must support or cooperate with secure authentication and authorization schemes. For example, if file access at a site is based on the AFS or OpenAFS file scheme, any queuing scheme must be capable of somehow issuing or renewing an AFS token for file access at the moment when the job is queued for execution, even if the valid token when the job was submitted has since expired.

- It should offer a number of different scheduling policies. For example the simplest would be first-in, first-out. Others may involve use of a job or user priority. There may be some kind of fair shares option, for example to avoid one user monopolizing a queue. Job size may be a paramater, or other required resources.

- In some cases there may need to be restrictions on which users can access which queues, or a need for some interactive use of some batch queues, for example to debug a process.

- There must be facilities for job accounting, error reporting and statistics gathering.

7.3 Load balancing

As measured across the cluster, there are many ways to allocate resources between processes, whether these are batch jobs or interactive processes or a mixture. In the case of batch jobs, some batch queuing schemes include their own allocation methods. Several HEP sites use the LSF product from Platform Computing and the latest release includes LSF Fairshare as an option; this is a rather sophisticated method of load sharing and takes account of many load parameters.

Simpler methods include the use of the network DNS (Distributed Naming Service). For example, the cluster is given a generic network (DNS) name and jobs, batch or interactive, are directed to this generic name instead of at individual nodes. The DNS server can be programmed to allocate jobs to individual nodes using any defined method from the simplest round-robin to more complicated algorithms taking account of free memory, free process slots, CPU load factor, etc.

All the above assumes all nodes are configured identically with respect to required resources. If a process requires something in particular such as extra memory which only some nodes possess, or access to local I/O resources on certain nodes, then the above load sharing scheme no longer applies or is greatly complicated.

Another caveat is that sharing a node between batch jobs and interactive sessions complicates any load sharing. Batch and interactive processes often have widely different characteristics and it may not be obvious how to tune a node to handle both efficiently. One aspect in particular is the predictability of the batch load given that the duration of interactive sessions by their nature are not predictable. This however does not rule out using otherwise idle cycles for running batch jobs in a node intended for interactive use. In this mode, both system administrator, and more importantly the user, realize that there can be no expectation on when a job will start or finish nor on how long it will take to run.

7.4 What restrictions are imposed on the user applications by the implementation?

7.5 What methods of influencing the allocation system are needed?

7.5.1 User interactive control

7.5.1.1 Kill jobs

7.5.1.2 Reorder jobs

7.5.1.3 Suspend jobs

7.5.2 User programmatic control

7.5.3 Admin interactive control

7.5.4 Admin programmatic control

8 Authentication and Authorization

8.1 The difference between Authentication and Authorization

8.2 Is the user community well described?

8.3 For what durations are authorizations appropriate?

8.4 How will users be able to authenticate themselves?

8.5 Who will be able to grant authorization?

How will authentication be checked?

Certificates – issued by national agency? Globus?

Kerberos (V4 or V5?)

8.6 Are there reports required?

8.6.1 For users?

8.6.2 For Administrators?

8.6.3 For management?

8.6.4 For funding agencies?

9 Software Issues

9.1 Programming Model

9.1.1 What is the system model that is being used (batch, SSI, …)?

Do you take the Unix approach of many small tools lashed together and torn down frequently or a more monolithic approach?

9.2 System Software

Do you use commercial operating system or public domain systems? On some architectures, there is normally a default (e.g. Solaris on SUNs) but nowadays, Linux is available almost everywhere. And on PCs the choice is normally between Linux and Windows although there are there other options including BSD UNIX, Solaris and others.

Commercial software will have licence charges but "freeware" is not free, it is typically more labour-intensive and requires certain skills. And some architectures, especially Windows, will require client licences on nodes which access central servers.

9.2.1 How and when to upgrade system software?

Performing system upgrades is highly intrusive and sometimes inevitably interfering. By the first we mean that it usually requires dedicated system time and all user jobs must be stopped and users kept off the system. With a large cluster and a large userbase, this makes scheduling such updates difficult.

The risk of interference is two-fold.

1. Some codes may depend on a particular system environment, for example on particular versions of libraries or compilers; if these change during a system upgrade, users may need to re-build their programs and perhaps even require a new cycle of debugging or tuning

2. Even worse is when some users require or demand a particular system version and other users require or demand a different one. How to arbitrate between conflicting requirements is clearly a local issue but a mechanism to resolve this should exist from the start such that the computer management alone does not decide.

9.3 Application Development

9.3.1 What tools are required?

Several sites reported the selection of tools/infrastructure products being heavily influenced by the desires for wide distribution and ability of receiving sites to create the infrastructure/licenses necessary to use the products. Not clear how this need competes with the developers desires to use standard/new tools, and management's desire to leverage development investments.

9.3.1.1 Is there a set of defined tools required?

9.3.1.2 Is there a set of optional tools allowed?

9.3.1.3 Are there forbidden tools?

9.3.1.4 How do these lists change?

9.3.2 How do developers exchange code?

9.3.3 How do developers validate code?

9.3.4 What design rules are enforced?

Question raised on whether implementing static builds as standard would improve protection from detailed OS configuration variation? Are dynamic libraries preferred?

9.3.5 How do developers get access to the development environment?

9.3.6 How are licenses administered?

9.3.7 Should we be using/referencing the SourceForge (etal) efforts directly?

9.4 Infrastructure Elements

There is a common scale of logical elements (mayors as they are known in the Chiba City scheme [*** check citation and add link ***]) of 30-60 units. Is this ergonomically significant? Or is this just a measure of the current scale?

Sandia has logical unit of 256 (in the plane) is there substructure maintained at that scale operations?

9.5 Operational Tools

Comment was made that system administrators want monitoring and alarms, developers want performance information.

9.6 Migration Tools

9.7 Training

10 User Interfaces

10.1 Helpdesk

10.2 User Feedback

Surveys are significant work and rather frequently done. The results and criteria of surveys are not widely shared. Often because of bias (explicit and

Implicit) by local requirements that are not well explained (understood ?) and potentially confusing/embarrassing to present in public.

Appendix A. Cluster References

Beowulf Clusters

These use parallel processing to improve on the performance of individual applications. See reference http://www.beowulf.org. CHECK Beowulf clusters feature commodity CPUs interconnected by a LAN, running Linux. Processes can spawn child processes on different nodes on the cluster and there is the notion of a global cluster-wide process ID.

MPI-based clusters

These use a message passing interface for parallel applications

MOSIX clusters

MOSIX, developed by the Israeli army in the early 80s, extends the functionality of the kernel in order to be able to move running processes between nodes in the cluster according to load. It requires a large bandwidth for the cluster interconnect and all process I/O is performed via the node on which the process was initiated. (CHECK)

Appendix B. Design and Build Summary

Is this a better ordering?

Establish cluster requirements (Design Goals)

Implement a configuration to meet these goals

Plan the installation including all components

Labeling

- Minimize single points of failure
- Reduce complexity where possible