Midnight Code Documentation

Saturn Installation and Operations Manual

Classified: PUBLIC

Project Code: “Saturn”

Status: RELEASE
Version: 0.1
Issue Date: May 28, 2013
Document Type: CONTROLLED

Table of Contents

1 How to use this manual

This manual is a guide to assist people with the assembly, configuration and operation of a Midnight Code Saturn storage appliance.

The manual has been designed so that it can be read from start to finish to build a capability from infrastructure to service, or it can be used as a ready reference to seek out targeted information, by concept.

1.1 Where to find ..

This manual consists of five sections;

Chapter 1 – How to use this manual
This chapter (the one you're reading now) provides detail on the structure of the manual, how each section should be used and what standard mnemonics have been applied.

Chapter 2 – About Saturn
Understand Saturn: Learn about what Saturn is, what it is intended to be, how it came about and where it is going. Chapter 2 is a good chapter to read if you just want to know if Saturn is going to fit your storage needs.

Chapter 3 – Installation
From hardware to software installation, chapter 3 takes you through a build process from the ground up (including pictures), and calls out things that you should consider before building your own. If you already have hardware then treat chapter 3 like a ready reference for finding the software solution best suited to your build.

Chapter 4 – Operation
Once you have your Saturn device ready, you need to know how to configure it. Learn how to configure the Saturn device, establish a storage resource and serve it to the network. Then read on to understand what maintenance and diagnostic options are available to you and what common problems can be quickly remediated.

Chapter 5 – Licensing
All of the licensing information about Saturn has been published in chapter 5. This includes the licensing of the distribution itself, the copyright of the Midnight Code software, and the licensing of the constituent software packages that make up the distribution.

You will find concepts in each section are also grouped into subsections that further reflect the relationships of that material.

1.2 What does this mean?

This manual uses a set of common notations to improve its readability and reference-ability.

1.2.1 Box

Information found in a grey box like this one means that the included text is a literal command or configuration item;

```
type in this command or configuration item
```

1.2.2 List

Numbered lists should be evaluated in the documented order. Bullet-point lists are lists of items that belong to a concept without any particular order. Both types of list can contain nested lists which reflect an ordered or unordered list of derivative concepts (respectively).

1.2.3 Link

Cross referencing (linking from one part of this document to another) will be used to indicate related concept or dependent process without repeating the same text twice in the manual. External links (linking from this document to other documents or Internet resources) are provided for further reading, or to indicate the source of a concept, data or file.

1.3 Is it complete?

This manual is structured to allow you to test its completeness and relevance;

1. Physically, this manual is 44 pages long. If you have less than 34 pages (including the cover page) then you are missing a page and should obtain a full copy of the document. All pages, other than the cover, are numbered.

2. Logically this manual is at version number 0.1, which means that it was written against version 0.1 of the Saturn software. If you have a version of the software that is greater (more recent) than the version of this manual, then check to see if there is a newer manual.

This manual is stored electronically at the Midnight Code web site. It should be accessible via both the Project site (see http://midnightcode.org/projects/saturn/) and the Papers site (see http://midnightcode.org/papers/).

1.4 Why is it spelt like that?

This manual has been written in English with Australian/British spelling.

2 About Saturn

2.1 Overview

Saturn is an embedded platform that creates a Network Attached Storage (NAS) device from any modern x86 computer and without specialist hardware.

Saturn is built on GlusterFS which caters for most standard RAID-like configurations (distributed, redundant and striped storage models), as well as supporting network-based asynchronous file replication to enable more common enterprise storage network (multi-box, multi-rack, multi-site) configurations. Beneath GlusterFS, Saturn is a portable Linux distribution produced by Midnight Code.

2.2 Objectives of Project Saturn

Saturn aims to be a high quality Network Attached Storage (NAS) appliance that can be assembled by home and office users without the expense and complexity of enterprise solutions.

2.2.1 Drivers

Firstly there were a number of market drivers that lead to the creation of Project Saturn;

- Commercial NAS (Network Attached Storage) and SAN (Storage Area Network) solutions tended to be unnecessarily expensive, despite their impressive features and maturity.
- But the consumer grade storage devices were under powered (10Mbps performance on 1Gbps NICs) with proprietary on-disk data formats that couldn't be recovered in hardware failures and were not extensible (adding a drive added another logical volume, not more capacity to the existing logical volume).

And then there were a number of operational and investment drivers;

- Mirroring as a means to achieve high availability was unfavourable due its cost (mirroring 5 drives costs 10 drives, while RAID5 of 5 drives costs 6 drives for example).
- RAID solutions were limited to a single host – so there was no way to scale beyond one chassis' worth of capacity, nor was there any way to do off-site replication with RAID alone.
- Replacing hardware makes no sense – i.e. replacing a 1TB drive with a 2TB drive gives you one new terabyte of capacity at the cost of two terabytes.
- Adding a drive should add capacity to an existing logical capacity (i.e. concatenation of drives should be achievable without the rebuilding the logical volume and redeploying the data).
- Replication and Concatenation functions should not be limited to a single chassis/enclosure or even a single site – there should be no practical upper limit to the amount of storage that can be created from the acquisition of COTS (Common Off The Shelf) hardware, and the Saturn software.
- It should also be easy to access from many clients, easy to operate and hard to lose data.

2.2.2 Target Features

All Project Saturn design features are considered against the following feature objectives;

- **Distributed**
  - Must be capable of concatenating multiple drives on either a single node or multiple nodes into a single contiguous storage capacity
  - Must be expansible such that adding another drive to an existing storage capacity must not require recreation of that storage capacity (destroying, creating, formatting)

- **Highly Available**
  - Must be capable of replicating data between drives on either a single node or multiple nodes into a single highly available storage capacity
  - Must be capable of multiple replications to support the “two copies in the same rack, one copy in another” principle, and to address the Bathtub curve penalty seen with bulk drive purchases (http://en.wikipedia.org/wiki/Bathtub_curve) i.e. multi-drive/node failures
  - Should support high-latency asynchronous replication to enable multi-site deployments

- **No dependence on strict RAID implementations**
  - Must not employ proprietary on-disk storage - Hardware RAID tends to employ proprietary on-disk storage and thus the volume or, indeed, any individual disk becomes unrecoverable when the RAID set is broken
  - Must enable good throughput on commodity equipment - Software RAID tends to bring about poor performance without accelerated hardware features

- **Recovery of replicated data should not be location specific**
  - Should be able to recover a failed node at one site either at its home site or at any other site with that data (i.e. being able to transport an empty or out-of-sync node to another location for high speed data recovery is highly desirable)

- **No need for manual maintenance**
  - All maintenance tasks should be exceptional and ideally limited to Moves, Adds and Changes
  - All sub-components that require routine maintenance should be automated where-ever practicable

- **No lost data**
  - When configuring the NAS it should not be possible to lose or destroy data without asserting a conscious administrative decision to do so.

- **Ideally accessible from Linux and Windows**
  - The presented storage capacity should be accessible from Linux and Windows
  - Client access should be delivered via native client interfaces (like SMB or NFS), though client-side agents are acceptable.

2.2.3 Use and Test Cases

The following use cases have been defined to help ensure that Project Saturn's development achieves its functional targets;

- **Non-Shared Content Model**
  - Data that is globally distributed but optionally not mounted (not presented locally)
  - The presented file-system is stackable with other modules, such as the Crypto loop

- **Shared Content Model** (p*er above, plus)*
  - Multiple mounts (many simultaneous users)
  - Global name space
  - One user's content should not be delete-able by another's

- **Common Content Model**
  - Local access of remote content should create a local copy (a “transfer once” policy)
  - A user should be able to know if he is the person who is about to delete the last copy of something

In addition to the above, the following test cases have been defined to help validate the technical architecture in the context of the functional targets;

- Saturn will pass the logical architecture tests if it can facilitate;
  - One device with one disk
  - One device with many disks
  - One device locally with one device elsewhere
  - One device locally and two devices elsewhere
  - Two devices locally and one device elsewhere

2.2.4 Abandoned Objectives

Saturn was originally intended to support common LAN file sharing protocols (including SMB/CIFS, NFS and AoE/ATA over Ethernet) with a scalable storage capacity (minimum, one terabyte of storage), and at a LAN access rate of no less than one gigabit per second (1Gbps). There is currently no intention to provide AoE (ATA over Ethernet) support in any future version of Saturn.

We have remained open to the prospect that if no one DFS (Distributed File System) solution would meet our requirements that we would be prepared to consider integration our own replication solution (for example an rsync or torrent solution) into the back-end to create an automated replication overlay. To-date the distribution aspect of the architecture has not been completely resolved.

2.2.5 Key Limitations

Saturn is undergoing constant development. The following key limitations should be noted;

- The BOOTP/DHCP client capability is currently broken in libMidnightCode
- Of the three Gluster storage modes (distribute, replicate, stripe) only distribute has been implemented in libMidnightCode
- The web-based administration interface is read-only.

2.3 History of the Project

Project Saturn was always intended to be a multi-terabyte NAS device that was compatible with both Linux and Windows desktop and server platforms, but it has been a long time coming.

2.3.1 2006

Saturn had been on the cards for quite a while, though a lack of available time had held-back initial in-roads into the project. To motivate myself I ordered hardware for the project on January 11, 2006, which meant that coding had to start! At that time, utilising the best price-to-megabyte ratio for IDE/SATA hard disk drives (320GB SATA I disks), Saturn would provide roughly two terabytes of storage for about US$1,000 (valued as at Q1, 2006).

It was obvious that a number of projects being developed at Midnight Code (i.e. the “Planet Series”) would benefit from a single integrated architecture that was able to provide a common, extensible and lightweight embedded Linux distribution to all projects thereby removing re-work that would have been performed for each. And with the success of the CHAOS Linux distribution (http://midnightcode.org/projects/chaos/), a number of lessons had been learnt regarding the internal ad-hoc management architecture that was used to deploy and regulate the Operating System internals.

Thus a new architecture was conceived to retain the benefits and build on the learnings of the CHAOS experience. This architecture was encapsulated in a framework called libMidnightCode (http://midnightcode.org/projects/libmidnightcode/) which slowly evolved from fundamentals such as text manipulation and socket management (in 2005) to holistic functions such as message brokering, interface (NIC) management and configuration management. Both Project Saturn (http://midnightcode.org/projects/saturn/) and Project ATS (http://midnightcode.org/projects/ats/) were the first Midnight Code projects to employ the incomplete libMidnightCode software, which in turn drove requirements back down into libMidnightCode.

By the end of January that year;

- libMidnightCode had been extended to support automatic detection of the first 16 IDE devices and the first 16 SCSI devices available under a Linux 2.6 kernel (16 being an arbitrary limit). Discovery information included the Vendor, Product, Revision, Serial Number and Capacity for each drive.
- libMidnightCode was also extended to support configuration file writing (as well as

configuration file reading, which it already supported). This should allow the device to manipulate its own configuration file, leaving the user free to manage it via HTTP

- A proof-of-concept build (the “reference platform”) was made from the Linux 2.4 kernel combined with a libMidnightCode capable of low level disk and interface management along with the corresponding configuration file controls, and bundled into a USB key booting image.
- Development effort was then focused on getting libMidnightCode to interface with a HTTP user environment so that management could be enabled from user to system.

By using libMidnightCode, the reference platform immediately took on the lightweight nature of CHAOS. Even in its experimental form was still a tiny distribution (about 12Mbytes in size) and it was able to utilise USB mass-storage hardware for its embedded OS (i.e. it could boot from a USB key) guaranteeing that no disk storage capacity would be wasted to the host OS – every disk in the resultant NAS would be shared storage – for a Saturn appliance built from the reference platform.

Unfortunately, there was limited hardware support for SATA controllers in the 2.4 kernel series, so the reference platform failed to boot the then Project Saturn hardware.

By the end of October, version 1.2 of libMidnightCode had been integrated with a proof of concept Linux 2.6 kernel based reference platform and been used to USB boot the Project Saturn hardware to multi-user state. Testing had shown that the proof of concept platform with its integrated libMidnightCode software (particularly with improvements to the init process) was able to detect all of the drives in the system, be user configured via static config file, and also fall back to a default known-good state in case of a missing config file; something CHAOS has previously been able to do.

Project Saturn was well on its way.

2.3.2 2007 – 2008

By March 2007 the high level web interface (webint) library framework was in place, but was read-only (not usable as an interactive administration interface).

Unfortunately for Saturn, 2007 saw all of my project time consumed by Project ATS. The advances in libMidnightCode therefore favoured physical control and interface systems;

- In March, libMidnightCode saw improvements in the socket library, and the addition of Video4Linux UVC camera support.
- By June, the Casting protocol had been implemented, allowing for the future creation of a very simple, yet significantly more robust, Tyd replacement for CHAOS - dreamt of since the socket library code of August 2006. libMidnightCode version 1.4 also includes a new Servo motor control library and GPS library (for gpsd interaction), amongst a number of minor tweaks such as a child socket finder, safe list iteration, etc ad nausea.
- By July 2007, the Bootp/DHCP protocol had been implemented, facilitating integrated interface address discovery. A Syslog client library had also been added to allow for integrated system logging. A lot of debugging went into finding an issue with some Project

ATS equipment (to do with global variables in the library that were assigned static values which were not being shared by the application that was using the library - thus there was a pseudo-random dependency on system memory being zeroed).

The creation of a Bootp/DHCP client removed the last dependency for the boot prompt options in the original CHAOS image and meant that the reference platform no longer needed boot parameters to configure the booted operating environment. The reference platform was running kernel 2.6.22.

In 2008 I started a new role and my night time project work was replaced with late nights in the office. People were becoming interested in Saturn, they were all starting to see the same personal storage problems (scale and availability issues) and realised that the off-the-shelf solutions were very poor fits for their problem. But there was nothing of Saturn that could even be played with – apart from standalone concept demonstrators and a boot-able reference platform that essentially provided you with a web server and a shell prompt from bare metal.

Although it had a firm base and broad interest, Project Saturn was well and truly stalled.

2.3.3 2009 – 2010

The year 2009 was another great year for Saturn;

- In February I was mobbed – people who had heard about Saturn wanted it – they also wanted it to solve problems that hadn't been anticipated because they started comparing Saturn's original objectives to enterprise solutions and thought that the gaps could be closed. With the additional view-points a hefty debate ran to evolve Saturn's requirements from the various commodity and enterprise experiences the group had.
- One of the most persistent views was that of the recently publicised Google File System (GFS) used internally to Google (http://web.archive.org/web/20090220151747/http://labs.google.com/papers/gfs.html) which seemed to feature most of the benefits that were being sought, and certainly drove the availability view of “2 copies in the same rack, 1 copy in another”.
- At the same time, a comprehensive review was done of all of the available distributed file systems. We went to prototype on a couple of these;
  - We prototyped the MySQLFS but found that it was disappointingly slow on our 50GB test volume
  - We found that Hadoop had been modelled off the Google File System principles by Yahoo (http://en.wikipedia.org/wiki/Hadoop#Architecture), however when we went to prototype Hadoop we learned that it was written in JAVA and that there was no open way to distribute the JAVA runtime commercially. It was possible to compile Hadoop using “gcj” but only up to version 0.4.0 (we had tested Hadoop versions 0.4.0, 0.5.0, 0.6.2 and 0.19.1 – the latest at that time).
- By June we had decided on GlusterFS as the Open Source technology that would provide the storage feature-set that most closely matched our requirements, developing configurations for Distributed and Replicated storage in GlusterFS version 2.02.
- And, in July, GlusterFS was introduced to the reference platform to create a distinct “Project

Saturn" image.
- In August I ran a thought experiment with Chris to see if we could get a "de-dupe" layer into GlusterFS for Saturn and spent two weeks trying to implement the straw-man before I fell ill.
- At some stage in the 2009 development season the Bootp/DHCP client code had been broken.

Unfortunately in September 2009 I was assigned to an urgent piece of work that ran for a month and and then a large Program of work that ate into my night-times again, for most of the next two years.

By March 2010, GlusterFS 3.0.3 had been released changing the architecture of the GlusterFS software and disabling one of the core features we had successfully deployed – the replication of an irregularly sized distributed volume. Instead GlusterFS required a distributed/replicated volume to be consistent of replicated bricks (matched disks on alternate hosts) that were then added to distribute volumes.

Project Saturn had seen a great run but was stalled again.

2.3.4 2011 – 2012

After I moved house, in October 2011, the BIOS battery had to be replaced in the Project Saturn hardware – a clear indication of both the age of the project and the lack of recent time investment.

Project Saturn had its third wave in 2012 when I took some time off which ended with both Michael and I sharing a week of holiday hours on Saturn;
- We initially persisted with GlusterFS 3.2.6 because we were so entrenched in the storage architecture that it enabled.
- A number of patches were created and submitted to the Gluster project identifying and resolving 32bit issues in GlusterFS 3.2.6, amongst other features sought for Saturn
- The aged Project Saturn hardware was refreshed - Four 2TB drives were added (replacing 4 x 320GB drives) along with 32GB if RAM to create a 7.2TB volume under a GlusterFS 3.2.5 and with Linux Kernel 3.2.14.
  - The move to GlusterFS 3.2.5 also created a number of embedding issues due to Gluster managing its state via extended attributes in the root directory of the managed file system
  - libMidnightCode 1.6 was extended to manage GlusterFS and remove the need for custom shell scripts
- A month later the Project Saturn hardware was increased to five drives on GlusterFS 3.3.0 (replacing another 320GB drive)
- A portmapper was added to the Saturn image to enable Gluster's native NFS daemon
- Two months later another a sixth drive was added to the first volume and a second volume of two 3TB nodes was added – driving out a number of 32bit issues in libMidnightCode.
- The user interface (HTTP Administrative interface that was still read-only) was expanded to show all storage configuration information

By Q3 2012 there were two production Saturn nodes (including the refreshed Project Saturn hardware) with combined total of around 30TB of distributed NAS-presented storage and a number

of development nodes in existence.

Although it wasn't complete, Project Saturn was finally good enough to run in production.

2.3.5 2013

In early 2013 the two production Saturn nodes had hit their first ceiling – there was insufficient chassis space to add further physical drives, and the capacity projections showed that there would be insufficient logical space to store additional content in about 3 months. This lead to a piece of work on the embedding of GlusterFS's clustering capability, including circumvention of the additional state controls in Gluster software.

Due to the increasing interest in a (finally) usable Saturn, Michael and I were beginning to see the problems that was going to come with trying to explain Project Saturn, the Saturn software and the tips and tricks that we'd developed from our experience in building production Saturn environments. So I developed the first Saturn Manual in May.

Thus, at the time of writing, Saturn is capable of presenting a volume of capacity that is created from both the local and remote storage capacity of multiple Saturn nodes.

Later this year both production sites will add a second node each and cluster to additional capacity and we are anticipating another production site and a couple of new experimental sites.

2.4 With Many Thanks

Midnight Code thanks Michael Welton for his massive time investment in the theory, specification and testing of Saturn.

Michael first embraced the intent of the project in 2009 when he saw its potential, and its shortcomings. He then helped drive much of the thinking that shaped the Saturn we have today including some of the key enterprise characteristics that we have come to depend on. I could only guess at the hundreds of hours that Michael has contributed in nights, weekends, and even through holiday stretches – across cafes, dining rooms, lounge room floors and the occasional cramped office – working through requirements, proposals, architectures and then tirelessly building and testing Saturn implementations (How many USB keys did we buy in the end?). Through the heated debates about theoretical use cases to the vast data losses of failed early builds, Michael has persisted with the project and has driven it with a genuine desire to achieve its ambitious goals. Champion.

Midnight Code also sends its thanks to Chris Kelada for his review of the revitalised Saturn proposal in 2009, as well as his contribution to one of my most favourite coffee-table thought experiments – Why can't we have an "enterprise" style open source de-dupe? How hard could it possibly be!? The resultant straw-man became the Midnight Code GlusterFS De-duplication Xlator – still incomplete (bogged down in the Gluster/FUSE internals due to my coding skills and availability) at the time of writing – but very achievable thanks to those two hours of coffee table time one winter's eve in 2009.

2.5 About Midnight Code

Midnight Code is a singular resource created by Ian Latter to house and share the most useful open source software that he has developed.

These programs are the publicly publishable, cumulative and structured outputs of the seemingly ceaseless need to create that stems from the author himself. Some of the projects have a long meandering history that is due to their unique evolution, while others have been created simply to fit a niche need. The works that have been developed by the author under contract for commercial organisations are not public, and hence have not been published here. Though public works by the author, as published here, have been used to develop private (commercial) software and appliances.

The main project grouping is "The Planet Series" project set. This series of projects is designed to bring Linux to life, in the Home or Office, to fulfil the complete spectrum of communications and life-style technologies for all non-enterprise consumers. These projects start at Mercury with the development environment required to get you started, and end at Pluto with connectivity from your LAN to the rest of the universe.

Each project is clearly defined, and contains screen shots, documentation, source code, links and activity information, as identified.

3 Installation

In this chapter we look at the hardware and software installation of a Saturn node. If you already have spare/available hardware then you can skip straight down to the software installation.

3.1 Considerations, Before You Begin

The following considerations are here to help you design your Saturn deployment.

3.1.1 Service Qualities

Ideally you should consider the service qualities for the storage facility that you are designing;

- **Availability** (the degree to which something is available for use), including:
  - **Manageability**, the ability to gather information about the state of something and to control it
  - **Serviceability**, the ability to identify problems and take corrective action, such as to repair or upgrade a component in a running system
  - **Performance**, the ability of a component to perform its tasks in an appropriate time
  - **Reliability**, or resistance to failure
  - **Recoverability**, or the ability to restore a system to a working state after an interruption
  - **Locatability**, the ability of a system to be found when needed
- **Assurance**, including:
  - **Security**, or the protection of information from unauthorized access
  - **Integrity**, or the assurance that data has not been corrupted
  - **Credibility**, or the level of trust in the integrity of the system and its data
- **Usability**, or ease-of-operation by users, including:
  - **International Operation**, including multi-lingual and multi-cultural abilities
- **Adaptability**, including:
  - **Interoperability**, whether within or outside the organization (for instance, interoperability of calendaring or scheduling functions may be key to the usefulness of a system)
  - **Scalability**, the ability of a component to grow or shrink its performance or capacity appropriately to the demands of the environment in which it operates
  - **Portability**, of data, people, applications, and components
  - **Extensibility**, or the ability to accept new functionality
  - The ability to offer access to services in new paradigms such as object-orientation

From the The Open Group Architecture Framework (TOGAF) Taxonomy of Service Qualities: http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap19.html

3.1.2 Infrastructure

With service qualities in mind you should consider the infrastructure options that are available to you to facilitate that service outcome.

- **If you are doing this an experiment**, for example, then consider utilising any available commodity components. An old laptop is a good storage device if you consider its portability (size) and availability (battery backed power), but is not as advantageous when you consider its performance (typically low end CPU and Disk specifications) and its reliability (laptop drive life cycle).

- **If you are looking to produce a high performance solution**, then consider higher performance x86 CPUs and large RAM footprints. Consider also high performance hard disks (what the industry calls “black” drives) on high through-put SATA controllers.

- **If you are looking to produce a high recoverability solution**, then consider deploying on highly available (clustered virtualised) infrastructure, or consider deploying to multiple physical sites with high speed and low latency interconnect, and a logical architecture that ensures that data is always present at two or more locations.

Whatever your priorities are, be true to your requirements and build accordingly. This isn't wasted work – it will be the case whether you choose to use Saturn or another storage technology.

3.2 Hardware

What follows is an example hardware solution – it is, in fact, the original 2006 Project Saturn infrastructure build from commodity parts. In that regard, the hard drives, memory and CPU are not the same ones in Saturn today. Please see Considerations for reasons you may wish to deviate from the supplied pattern or advice.

When the Project Saturn prototype hardware was bought the primary goal was capacity and the secondary was availability. This meant drives – lots of drives – as many as would fit in the smallest possible space – but then hard active hard drives get hot and need good cooling, and good cooling can mean loud fans.

3.2.1 What you'll need

Regardless of the infrastructure decisions that you have made you will be looking at the following components;

- **Computer Case (or Chassis)**
  Choose the chassis that meets your needs – rack-mount or not, number of drives, type of motherboard, etc. At the time the best computer case for drives, density, cooling and noise, that I could buy off the street, was the Thermaltake Armor VA8000BWS with capacity for nine 3.5 inch drives in a very manageable case (see http://www.thermaltake.com/product/Chassis/fulltower/armor/va8000bws.asp).

- **Drive Bays/Enclosures** (may not need, depending on your case)
  The VA8000BWS comes with one iCage, so you will need to buy two more to get to the full nine bays (see http://www.thermaltake.com/product/chassis/icage/a2309.htm). The iCage, having the fan at the front, keeps air pulled across the drives keeping them running very cool and very quiet.

- **Motherboard**
  Choose a motherboard that has the performance and manageability and scalability that you need. i.e. make sure you have the bus interfaces needed for your hard disk controllers, the socket you need for your CPU and the slots available for your RAM. We used an MSI K8N Neo in the original Project Saturn prototype.

- **CPU**
  Choose a CPU that has the performance you need – we worked to a ratio of two disks per core and haven't been held back by CPU (though we haven't tried compression or encryption at this stage).

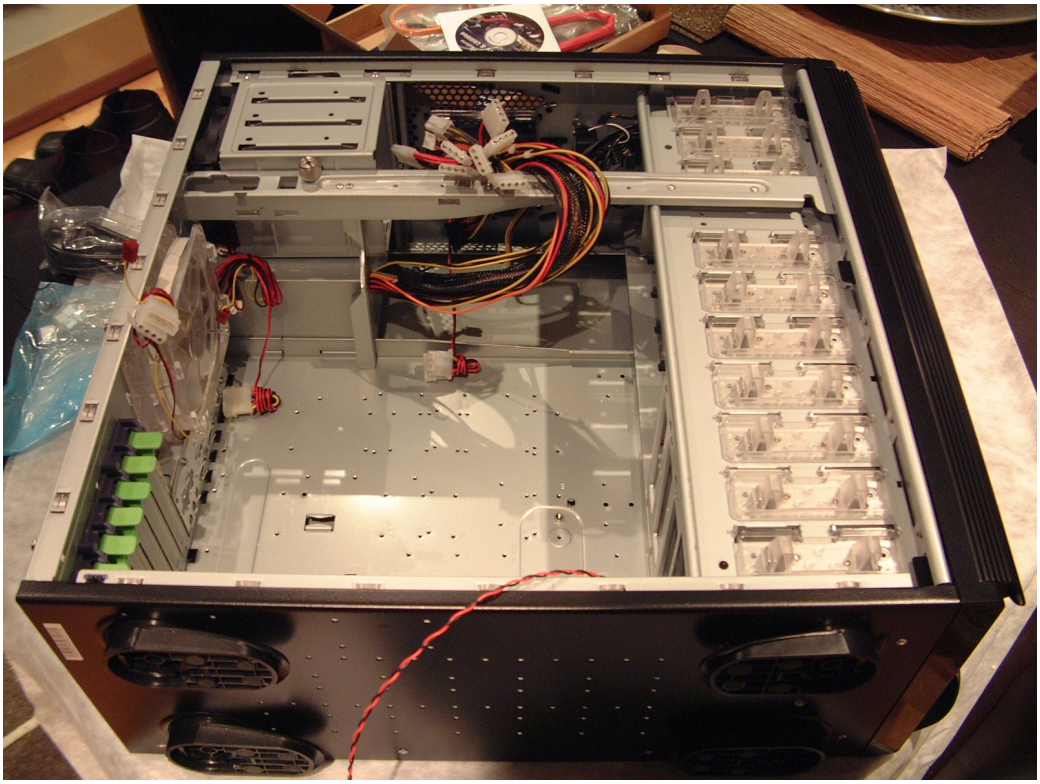
- **RAM**
  Get as much RAM as you can – although Saturn itself only consumes around 100Mbytes of RAM, GlusterFS loves it. The original Project Saturn prototype had 2GB of RAM which was more than enough for early GlusterFS versions, but didn't work at all with layer versions. The current Project Saturn production host as 32GB of RAM for 16TB of storage. Remember that in a NAS architecture, your NAS' RAM is your new “disk cache”, so don't be shy with it.

- **Hard Disk Controllers**
  Depending on your design decisions you may be able to get away with the hard disk controller built in to the motherboard. In the Project Saturn prototype we consumed all six of the motherboard SATA controller ports and added two more PCI SATA controllers to service the additional three drives. Be aware that many controllers over-subscribe their data paths so that you may never achieve “n channels” x 6Gbps through-put, as you might think by looking at the marketing material.

- **Hard Disk Drives (and Cables)**
  Choose drives based on your power, performance, capacity and availability needs. Over time the Project Saturn prototype has had a wide variety of SATA I, SATA II, black and green drives of various sizes. Depending on your work-load you may see big differences between these drive types, or you may see none at all. We have worked with low IOPS (Input/output Operations Per Second) workloads so we have large capacity disks (which yields a low spindle/head to Gigabyte ratio). If your workload is IOPS heavy (like a file-system servicing a high performance database) then you may need lots of small capacity drives in order to increase your IOPS to the application. It is important to research the performance of the drives and consider them carefully against your needs and service expectations.

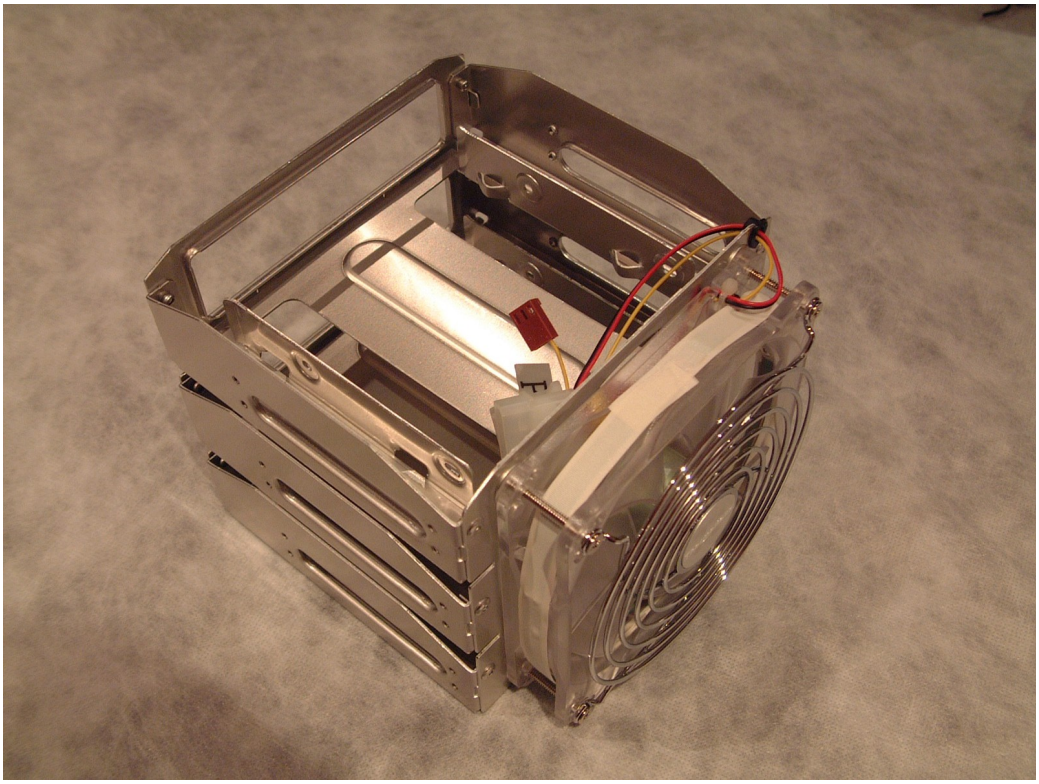
3.2.2 Computer Case/Chassis

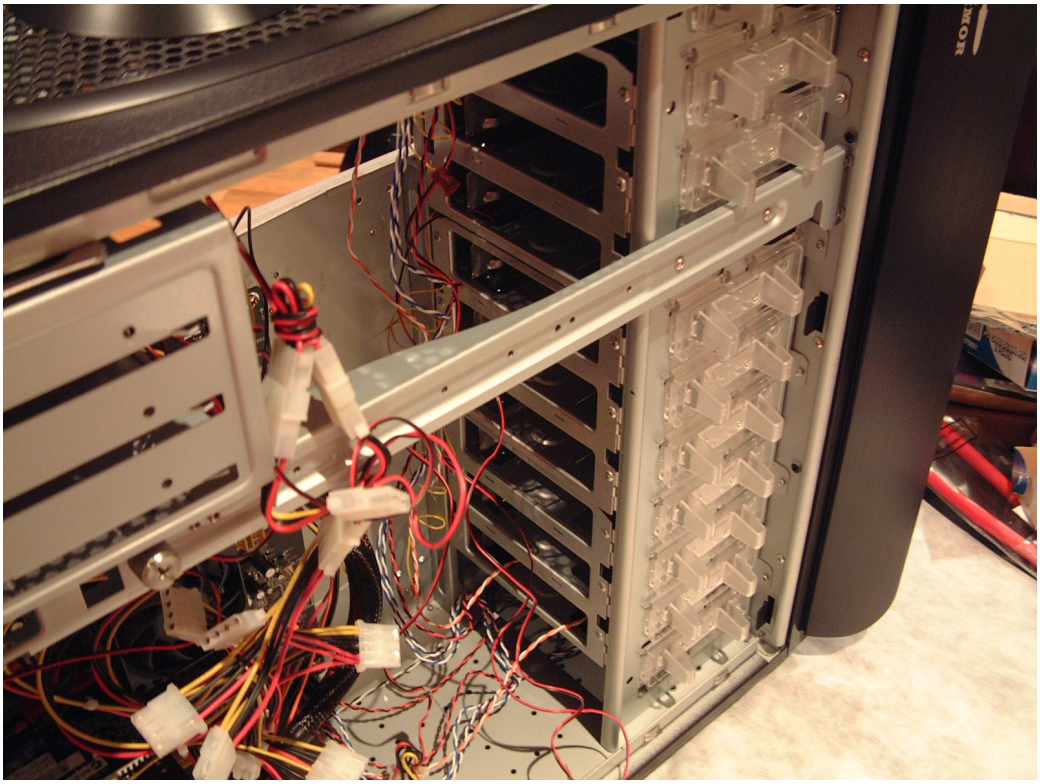
The Thermaltake Armor VA8000BWS (comes with one iCage) – perfect home Saturn box;



3.2.3 Drive Bays

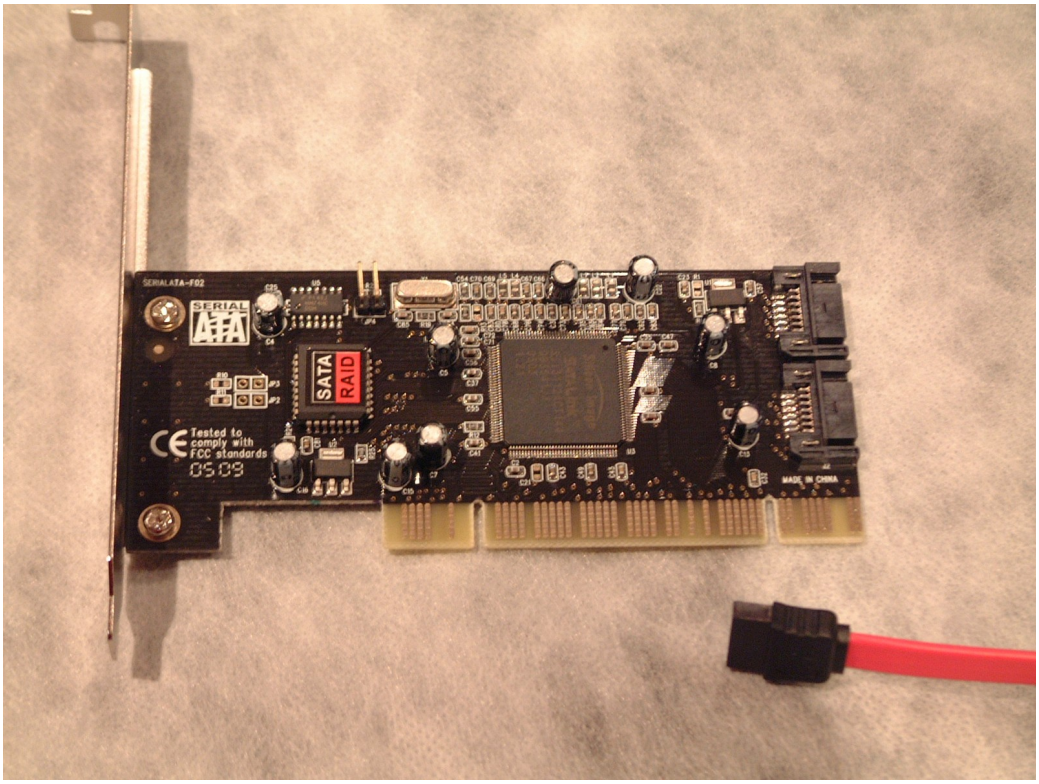
Add two more iCages to get to nine 3.5” drive bays;





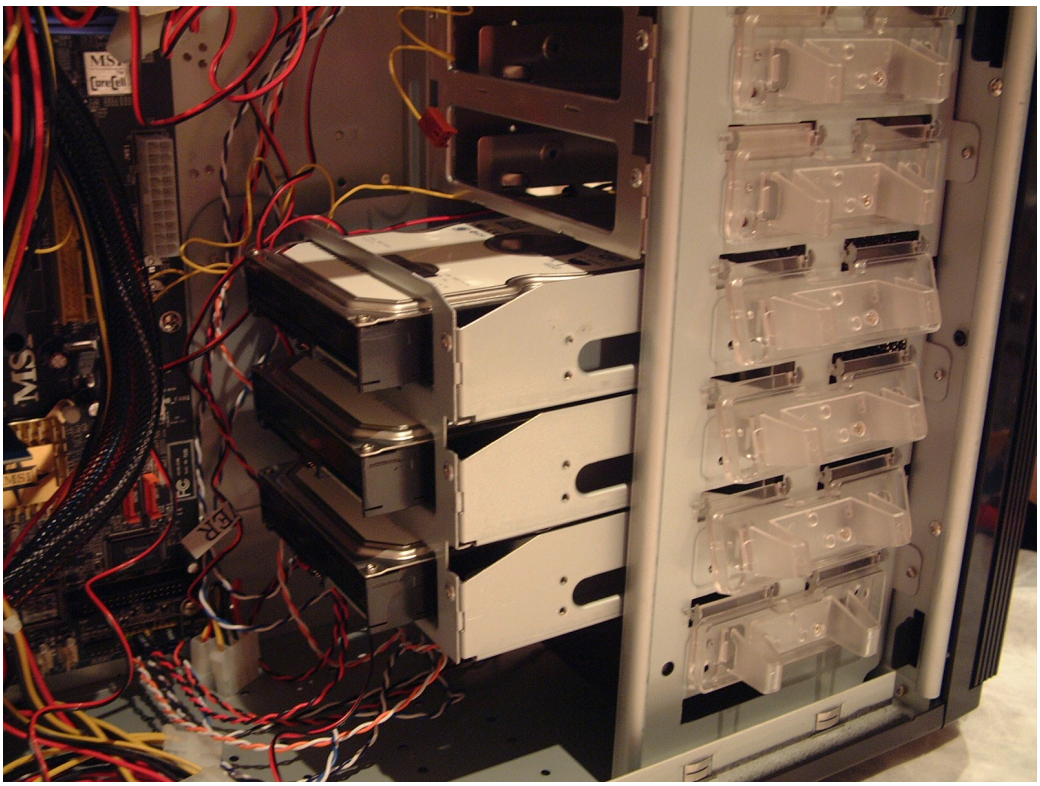
3.2.4 Hard Disk Controllers

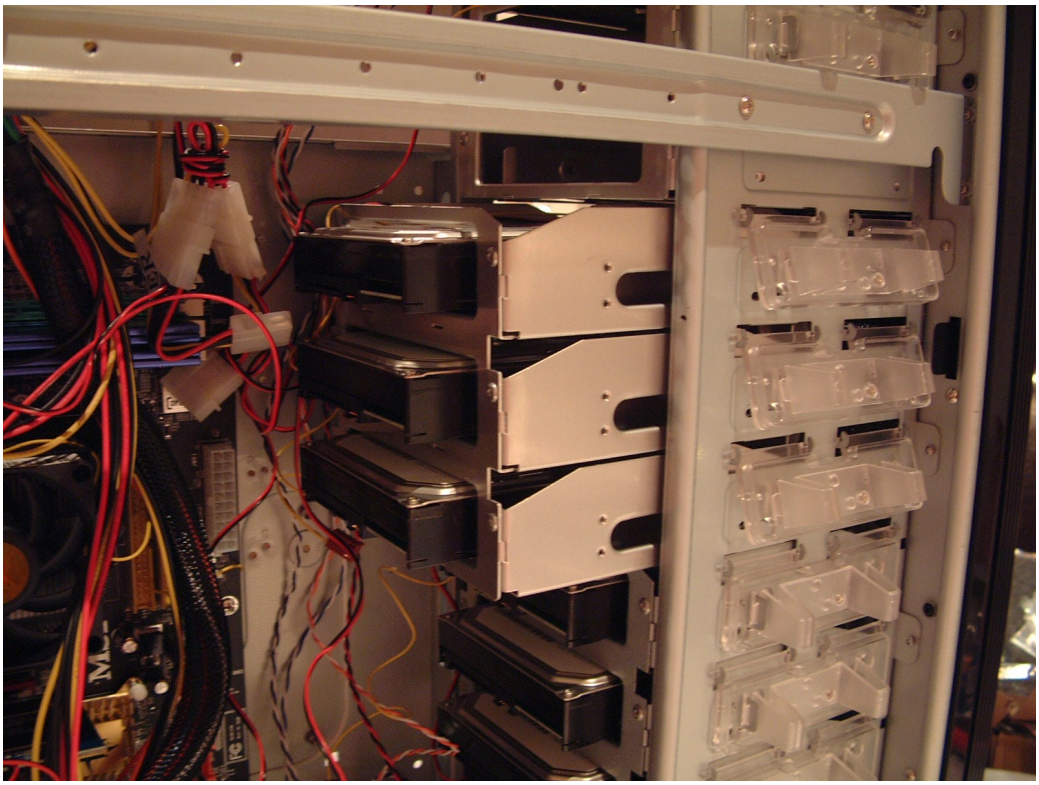
A basic two-channel (two port) SATA I controller, RAID feature included but not used;

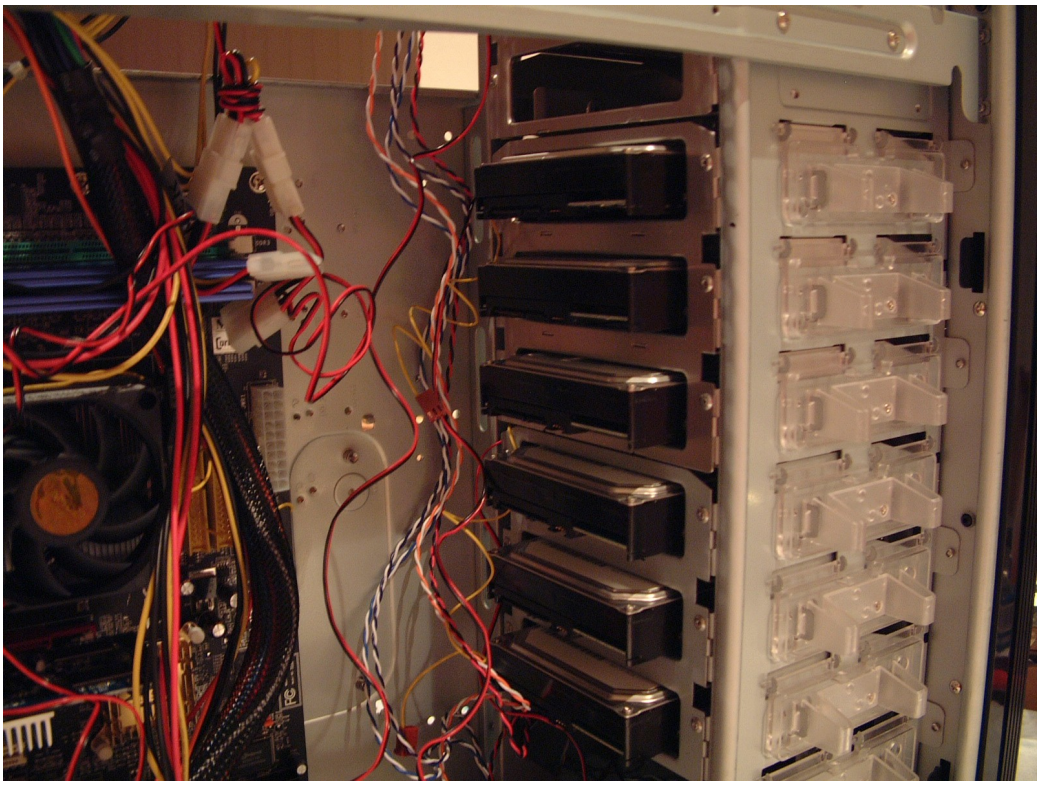


3.2.5 Hard Disk Drives

In the Project Saturn prototype environment disks directly correlate to capacity, and in 2006 this is what 1TB of capacity looked like;

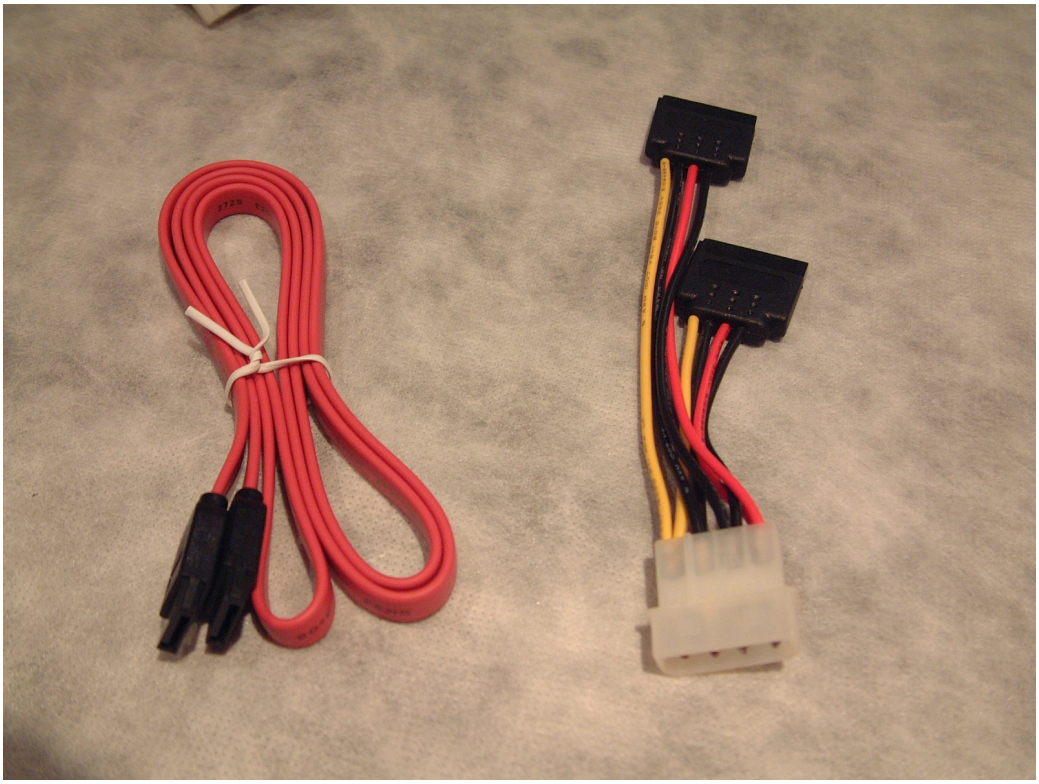


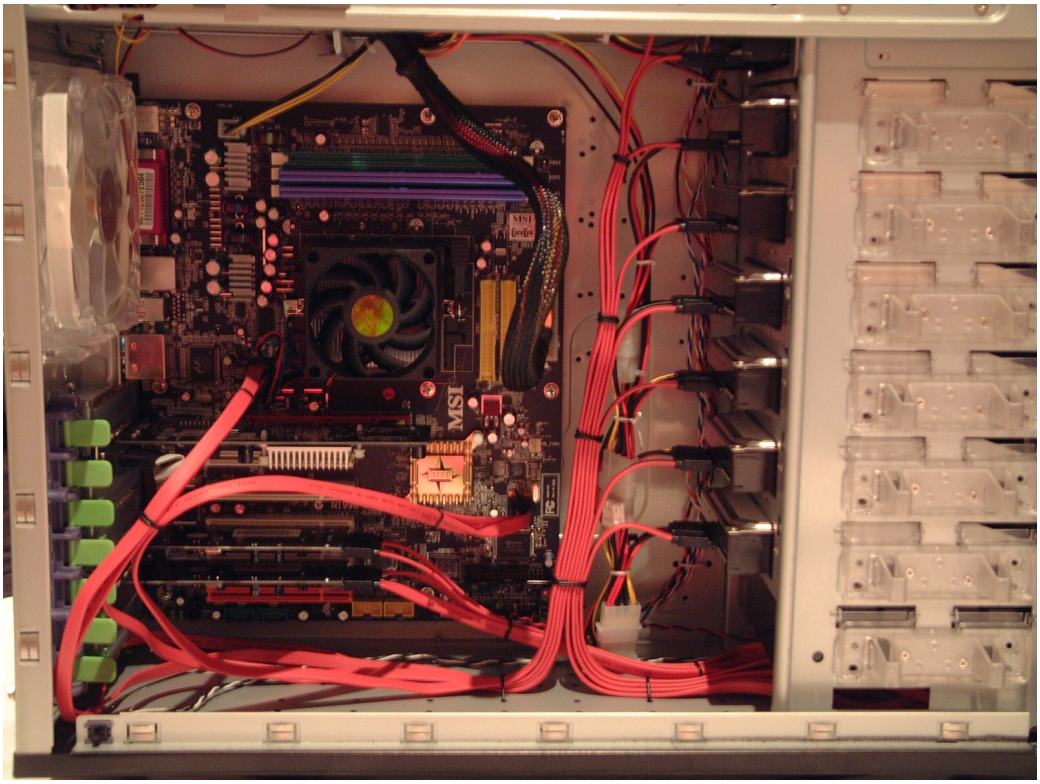




3.2.6 Hard Disk Cables

In a large case standard length cables won't reach your drives, get long data cables and extension/doubling power leads;





3.2.7 Make it cool

With great power comes great responsibility. Its your responsibility to make the hardware as cool as the software. Saturn, needs you!

3.3 Software

In this section you will find the software solution best suited to your build.

3.3.1 Understanding the boot options

There will be a number of ways to boot the Saturn image, however only two have been prepared to date. Even once all boot options have been made available, the recommended boot option will be the USB key.

3.3.2 Preparing Boot Media

Before you can boot Saturn you must prepare your boot media.

Please note that these instructions assume that you're using Linux to prepare your boot media. The extra umount/mounts are there because of the auto mounting and random locations based on different distributions.

3.3.2.1 USB key (thumb-drive)

The USB key is the preferred boot method for Saturn as it allows the user to readily edit the configuration file.

Insert the USB key into your workstation and note which device it is;

```
dmesg | grep sd | tail
```

Lets assume that your USB key is now in “/dev/sdx”.

Download the Saturn USB image;

```
wget http://midnightcode.org/projects/saturn/code/midnightcode-saturn-0.1-20130526230106-usb.burn.gz
```

Write the image to the USB key (replace “/dev/sdx” with your USB location);

```
umount /dev/sdx*
zcat midnightcode-saturn-0.1-*-usb.burn.gz | dd of=/dev/sdx bs=512 conv=sync
```

Wait for this process to fully complete, and then unplug the USB key. Your USB key is now ready.

3.3.2.2 ISO / CDROM / Optical Disc

There is currently no media preparation instructions for the ISO version of Saturn.

3.3.3 Advanced Boot Options

The advanced boot options are not yet completely defined.

3.3.3.1 PXE and TFTP booting

There is currently no process available for the PXE and TFTP booting option.

3.3.3.2 USB Booting Saturn in CD Booting Environments

VMWare Player (up to version 5.0.2) doesn't appear to have the means to boot from either a USB key image or a physical USB key. The common work-around for this (and other environments that can't boot USB devices) is to use a small ISO boot-loader that bootstraps the USB key on behalf of the native BIOS. Plop is a tiny boot-loader that does exactly that – it boots USB Linux distributions from an CD (disc) or ISO (file).

At the time of writing the available Plop version was 5.0.14, check here for a current version if it is no longer available (http://www.plop.at/en/bootmanager/download.html).

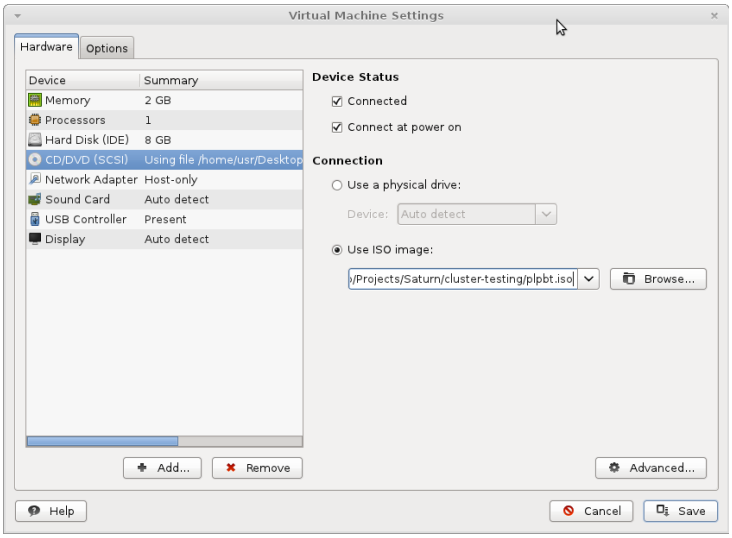
Download Plop;

```
wget http://download.plop.at/files/bootmngr/plpbt-5.0.14.zip
```

Extract the “plpbt.iso” file from that archive;

```
unzip plpbt-5.0.14.zip */plpbt.iso
```

Then either burn plpbt-5.0.14/plpbt.iso to a physical disk (if its a physical computer your booting from) or configure your virtualisation software to boot from the ISO file;



When you boot the system, select “USB” from the Plop boot menu.

4 Operation

Once you have successfully built the hardware and prepared the boot media for your Saturn node, you are ready to boot, configure and operate the system.

4.1 Concepts

The following section describes concepts that may be referred to elsewhere in the documentation without further definition.

4.1.1 GlusterFS

GlusterFS is a powerful network/cluster filesystem written in user space which uses FUSE to hook itself with VFS layer. GlusterFS takes a layered approach to the file system, where features are added/removed as per the requirement. Though GlusterFS is a File System, it uses already tried and tested disk file systems like ext3, ext4, xfs, etc. to store the data. It can easily scale up to petabytes of storage which is available to user under a single mount point.

- **Brick** The brick is the storage filesystem that has been assigned to a volume.
- **Client** The machine which mounts the volume (this may also be a server).
- **Server** The machine (virtual or bare metal) which hosts the actual filesystem in which data will be stored.
- **Subvolume** A brick after being processed by at least one translator.
- **Volume** The final share after it passes through all the translators.

From the GusterFS Concepts:
http://gluster.org/community/documentation/index.php/GlusterFS_Concepts

4.1.2 Saturn

There are a set of principles and limitations that you should observe within Saturn;

- For configuration purposes Saturn is able to differentiate a disk from a brick
- There is no partitioning – A GlusterFS brick consumes an entire Saturn disk
- The default system configuration file at /etc/default.cfg will be used to fall-back from any configuration failure
- Saturn does its drive configuration matching by Linux device (/dev entry), not UUID.
- There are two partitions on the Saturn USB key – the first contains the compressed kernel and compressed memory root file-system, the second contains the configuration file “device.cfg” for your node.
- Saturn is ACPI aware and will attempt to shut the system down properly in less than the 8 seconds required to do a forced power off (if you're BIOS is not set-up for instant-off).
- Saturn runs entirely from memory – once it has booted you can safely remove the USB key.
- GlusterFS logging has been disabled in Saturn to prevent it from filling the memory file-system.

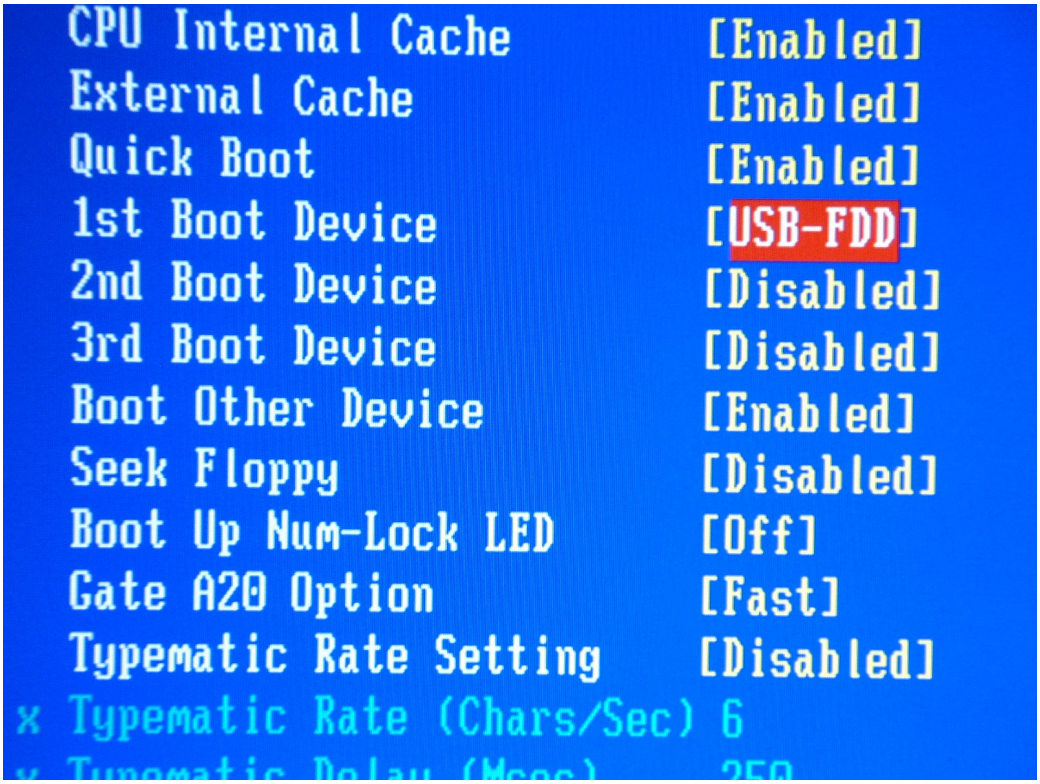
4.2 Booting for the first time

With your boot media prepared you are now ready to boot your system for the first time.

4.2.1 Configure the BIOS

Be sure to configure your BIOS for your boot preferences. If you have chosen USB media, for example, then you need to configure a USB Mass Storage device as your boot device.

This configuration may not be straight-forward, depending on the age of your computer hardware. For most BIOS systems the USB setup is for a USB HDD, for others it may be a HDD that is listed as the manufacturer of your USB key, why in the older example below, it happened to be a USB FDD that booted the key;



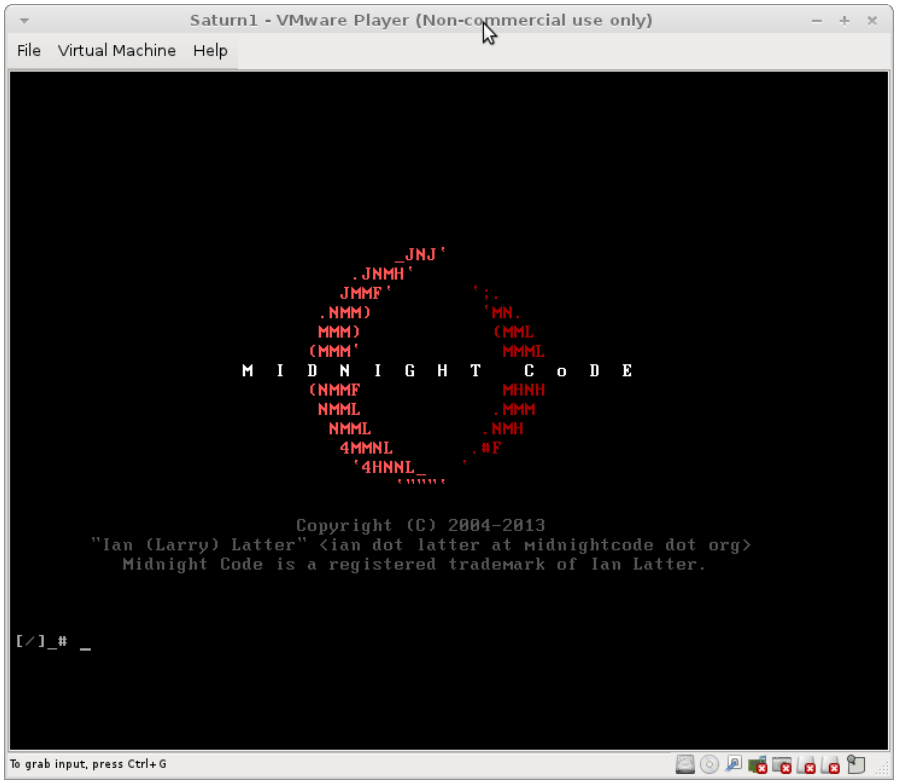
Please refer to documentation that came with your motherboard for the configuration options needed to enable USB, PXE or CDROM booting, if you aren't sure how to configure your system.

If you intend for your Saturn system to have no physical console (keyboard or screen) then this is where you should configure hardware to ignore those missing peripherals.

The BIOS is also responsible for the response to power events. It is recommended that you set-up an 8 second shut-down on power button push. It is up to you if you want your Saturn NAS to return to service automatically on power return in case of a power outage/incident.

4.2.2 The Saturn Console

If all goes well the boot process should stop at a Midnight Code logo with a shell prompt (the *console*);



4.3 Establish an initial Configuration File

Unfortunately, due to the read-only nature of the current administrative interface, you will need to work from the command line to configure Saturn, via the console.

When Saturn boots, all that it discovers is written to a configuration file in the “/tmp” directory. The easiest way to establish your initial configuration is to copy this file to the configuration directory on your boot media (we'll assume you're USB key /dev/sdx);

```
mount /dev/sdx2 /mnt/usb
cp /tmp/out.cfg /mnt/usb/device.cfg
sync
umount /mnt/usb/
```

Your Saturn shouldn't complain about this new config, and you are welcome to test it by rebooting;

```
reboot
```

4.4 Configuring Saturn

Eventually, the following configuration process will be graphical. However, the graphical administrative interface for Saturn, does not yet take user input. This means that you can see the results of your configuration there (HTTP to port 90) can't make those changes there.

4.4.1 Examples

For the configuration examples used in this section consider Saturn nodes;

- **Saturn1**
  - 192.168.179.101 / 255.255.255.0
- **Saturn2:**
  - 192.168.179.102 / 255.255.255.0

4.4.2 How to Edit the device.cfg File

You don't have to use “vi”

The following documentation assumes that you know how to use “vi”, the common Unix text editor, so that you can work on the Saturn text console. However you can plug the USB key into a Linux or Windows desktop system and edit the device.cfg file using a graphical text editor if you wish.

Taking a copy of your device.cfg file is also the recommend way to backup your Saturn node configuration.

If you do use “vi”

If you are editing the configuration file on the console of your Saturn node, be sure to mount the configuration device;

```
mount /dev/sdx2 /mnt/usb
```

Then open the configuration file for editing;

```
vi /mnt/usb/device.cfg
```

When you're done, write your changes and quit the editor;

```
:wq
```

Then unmount the configuration device;

```
umount /mnt/usb/
```

Note that Saturn's *vi* is a light-weight implementation – some functionality may be lacking.

4.4.3 Assign an IP address

To assign a static IP address, make the following changes to your device.cfg;

```
# Configuration Group: "interface"

set interface eth0      is_enabled 1
set interface eth0      is_alias 0
set interface eth0      flags 4163
set interface eth0      ifindex 4
set interface eth0      vendor_name "VMware, Inc."
set interface eth0      mac_address 00:0C:29:6F:27:3D
set interface eth0      is_static 1
set interface eth0      ipv4_address 192.168.179.101
set interface eth0      ipv4_mask 255.255.255.0
set interface eth0      ipv4_broadcast 255.255.255.255
set interface eth0      mtu 1500
```

4.4.4 Set-up a Default Route

To assign a default route, make the following changes to your device.cfg;

```
# Configuration Group: "route"

set route static0       ipv4_target 0.0.0.0
set route static0       ipv4_mask 0.0.0.0
set route static0       ipv4_gateway 192.168.179.1
set route static0       metric 1
set route static0       interface eth0
```

4.4.5 Set-up DNS

To assign DNS resolvers, make the following changes to your device.cfg;

```
# Configuration Group: "resolver"

set resolver domain     name domain.local
set resolver search     name domain.local
set resolver nameserver0  ipv4_address 192.168.179.1
```

4.4.6 Set-up NTP

To assign network time providers, make the following changes to your device.cfg;

```
# Configuration Group: "time"

set time ntpserver0      ipv4_address 0.pool.ntp.org
set time ntpserver1      ipv4_address 1.pool.ntp.org
set time ntpserver2      ipv4_address 2.pool.ntp.org
set time ntpserver3      ipv4_address 3.pool.ntp.org
```

Note that there is currently no method to configure your timezone via the device.cfg file.

4.5 Operations/Maintenance

What follows in this section is set of common operations or maintenance tasks for your Saturn.

4.5.1 Accessing Saturn

Other than the console there are two ways to administratively access a Saturn node.

4.5.1.1 From a Remote Shell

By popular demand you can obtain a remote shell into Saturn via SSH;

```
ssh user@192.168.179.101
```

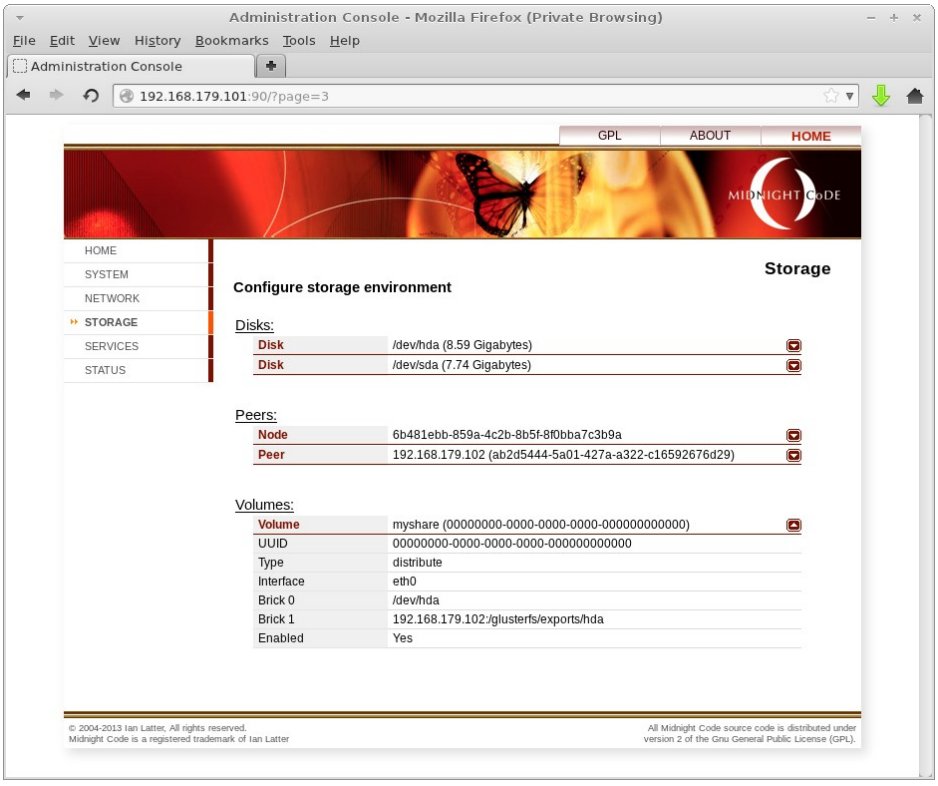
The default *user* and *root* passwords are “default1”. Do not expose this service to the Internet.

4.5.1.2 From the Web

To access the read-only administrative interface open a browser for HTTP to your Saturn IP address at port 90;

```
http://192.168.179.101:90
```

You should see an administration screen like this one;



4.5.2 Adding a Disk

One of the most common activities you are likely to perform on your Saturn node is the addition of a new disk. The following steps provide detail of the formatting and configuration of the disk as a GlusterFS *brick* in a distributed *volume*, once you have physically added the disk to the system.

4.5.2.1 Format the Disk

Assuming your new disk is at "/dev/sdz" the following shell command will format (ERASE) the entire disk as a single EXT4 file system with a 256MB Journal and no partition table;

```
mke2fs -t ext4 -F -J size=256 /dev/sdz
```

Any data that was on this disk before the format will be unrecoverable with standard operating system tools after the format.

4.5.2.2 Add the Disk to the Saturn Configuration

To assign the disk as a brick in a GlusterFS volume, make the following changes to the node's device.cfg;

```
# Configuration Group: "gluster"

set gluster server      uuid 6b481ebb-859a-4c2b-8b5f-8f0bba7c3b9a
set gluster volume0     name myshare
set gluster volume0     is_enabled 1
set gluster volume0     uuid 00000000-0000-0000-0000-000000000000
set gluster volume0     interface eth0
set gluster volume0     type distribute
set gluster volume0     brick0 /dev/hda
set gluster volume0     brick1 /dev/sdz
```

4.5.2.3 About GlusterFS UUIDs

First of all, note that neither GlusterFS nor Saturn intend to expose the end user to these numbers, but due to limitations in the Saturn administrative interface, you will need to understand them for now.

GlusterFS uses UUIDs (Universally Unique IDentifiers) to provide clear identity handles for internal resources. In that regard there are two that we manipulate – the UUID for the *server* and the UUID for the *volume*.

At present there is no requirement for the *volume* UUID, so this is always "zero" (00000000-0000-0000-0000-000000000000) – even for multiple volumes on the same server, or multiple volumes in the same cluster. You can generate UUIDs for these values if you wish, but you don't need to (they will be silently ignored).

However, each server (Saturn instance) must have a consistent UUID (the same one must be presented on each boot), and for clustered servers each server (node or peer) UUID must also be unique in that cluster. In this documentation, random examples have been used – if you have either a single server or two server cluster, then you can copy the supplied UUIDs. If you would like to use your own UUIDs, use any appropriate UUID generator – i.e.; http://www.uuidgenerator.net/ (version 1 or version 4 is fine).

4.5.3 Creating a Cluster

To assign nodes membership in a GlusterFS cluster, make the following changes to the first node's device.cfg;

```
set gluster server      uuid 6b481ebb-859a-4c2b-8b5f-8f0bba7c3b9a
set gluster peer0       uuid ab2d5444-5a01-427a-a322-c16592676d29
set gluster peer0       ipv4_address 192.168.179.102
set gluster volume0     name myshare
set gluster volume0     is_enabled 1
set gluster volume0     uuid 00000000-0000-0000-0000-000000000000
set gluster volume0     interface eth0
set gluster volume0     type distribute
set gluster volume0     brick0 /dev/hda
set gluster volume0     brick1 192.168.179.102:/glusterfs/exports/hda
```

And make the following changes to the second node's device.cfg;

```
set gluster server      uuid ab2d5444-5a01-427a-a322-c16592676d29
set gluster peer0       uuid 6b481ebb-859a-4c2b-8b5f-8f0bba7c3b9a
set gluster peer0       ipv4_address 192.168.179.101
set gluster volume0     name myshare
set gluster volume0     is_enabled 1
set gluster volume0     uuid 00000000-0000-0000-0000-000000000000
set gluster volume0     interface eth0
set gluster volume0     type distribute
set gluster volume0     brick0 /dev/hda
set gluster volume0     brick1 192.168.179.101:/glusterfs/exports/hda
```

4.5.4 Clients

In order to use a share, clients may access Saturn in the following ways.

4.5.4.1 Manual NFS Mount

The client should now be able to mount the “myshare” volume – note that it is NFS v3 only;

```
mkdir /mnt/saturn
mount -t nfs -o vers=3 192.168.179.101:/myshare /mnt/saturn/
```

4.5.5 Diagnostic Tools

At the text console, Saturn is a regular Linux distribution with a limited tool-set. You will find the following commands useful when performing diagnostic processes.

4.5.5.1 What version is this?

To confirm the version of Saturn that is running;

```
cat /etc/version
```

4.5.5.2 What is it doing?

The “top” command shows a list of processes ordered by CPU consumption and memory usage, as well as information about the current system load by function (such as IO Wait percentage);

```
top
```

4.5.5.3 Checking the Kernel Log

The Linux kernel sends certain event based messages directly to the console. These can be retrieved and reviewed via the “dmesg” command;

```
dmesg
```

4.5.5.4 Checking the System Log

In Saturn the system log (or “syslog”) is actually a ring buffer of log data that is allowed to overwrite itself over time, in order not to fill the embedded file-system with log data. To retrieve the system log use the “logread” command;

```
logread
```

4.5.5.5 Checking Running Processes

The “ps” utility will show a complete list of processes that a currently loaded;

```
ps
```

4.5.5.6 Verifying Mounted Disks

If you are looking to verify which disks have been mounted by Saturn you can use the “mount” command. Note that all GlusterFS exported bricks can be found as mounts via the following command;

```
mount | grep exports
```

4.5.5.7 Checking Free Disk Space

The amount of free disk space can be confirmed, by mount point, with the “df” command – the “-h” parameter formats the numeric output so that its “human readable”;

```
df -h
```

4.5.5.8 Reboot the System

You can reboot Saturn with the following command;

```
reboot
```

4.5.5.9 Shut-down the System

You can shut-down Saturn and power-off the device with the following command;

```
poweroff
```

4.5.6 Console Short-Cuts

If you are working on the console of Saturn you won't have the benefit of familiar features like scroll bars, which you may have in your graphical terminal emulator. However, these features do exist in the Linux console, and so these short-cuts may be useful;

4.5.6.1 Switching Consoles

There are four virtual consoles on Saturn. To switch to console “n” use the function key “n”. i.e. to switch to console two, use the following short-cut;

```
<Ctrl><Alt><F2>
```

4.5.6.2 Reviewing Console Output

You can scroll up and down through the history of the console output by using the following commands. Scrolling up is;

```
<Shift><PgUp>
```

Scrolling down is;

```
<Shift><PgDown>
```

4.5.6.3 Managing Command Output

Some commands provide vast amounts of output (substantially more than one screen's worth) and faster that can be read. To read the output of a command one page at a time, use “| more”;

```
dmesg | more
```

To see just the last few lines of output from a command, use the “| tail” command;

```
dmesg | tail
```

5 Licensing

The following sections detail the licensing of the Saturn and the components that comprise Saturn.

5.1 Midnight Code Trademark

The term *Midnight Code* and the two half-moon mnemonic are registered trademarks of Ian Latter.

5.2 The Midnight Code Applications and libMidnightCode Library

All Midnight Code source code, including the libMidnightCode library and the Midnight Code applications are released with the following copyright, trademark and licensing terms.

```
/*
                                  _JNJ`
                               .JNMH`
                              JMMF`       `;.
                             .NMM)          `MN.
                             MMM)            (MML
                            (MMM`             MMML
                    M  I  D  N  I  G  H  T    C  o  D  E
                            (NMMF             MHNH
                             NMML            .MMM
                              NMML          .NMH
                               4MMNL       .#F
                                `4HNNL_    `
                                    `""`

                          Copyright (C) 2004-2013
                "Ian (Larry) Latter" <ian dot latter at midnightcode dot org>

    Midnight Code is a registered trademark of Ian Latter.

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, and mirrored at the Midnight Code web
    site; as at version 2 of the License only.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
    (version 2) along with this program; if not, write to the Free
    Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    02111-1307 USA, or see http://midnightcode.org/gplv2.txt

*/
```

5.3 The Saturn Linux Distribution

Saturn is released under the GNU GENERAL PUBLIC LICENSE Version 2, June 1991.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                       51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".
```

```
Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
```

```
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.
```

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

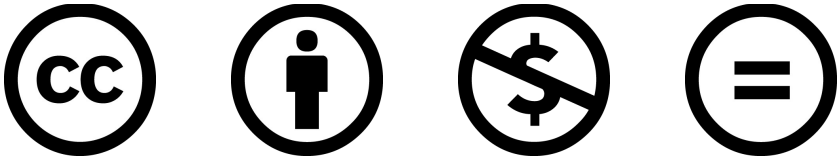
END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

```
  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA


Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

5.4 This Document



5.5 Constituent Software

In addition to the Midnight Code applications and the libMidnightCode library, the following open source software has been used in the Saturn distribution;

acpi-1.0.10	http://optusnet.dl.sourceforge.net/sourceforge/acpid/acpid-1.0.10.tar.gz
attr-2.4.46	http://download.savannah.gnu.org/releases/attr/attr-2.4.46.src.tar.gz
busybox-1.19.4	http://www.busybox.net/downloads/busybox-1.19.4.tar.bz2
devmap-1.02.20	ftp://sources.redhat.com/pub/dm/device-mapper.1.02.20.tgz
e2fsprogs-1.42.7	http://waix.dl.sourceforge.net/project/e2fsprogs/e2fsprogs/1.42.7/e2fsprogs-1.42.7.tar.gz
elf-0.8.13	http://www.mr511.de/software/libelf-0.8.13.tar.gz
glibc-2.17	http://ftp.gnu.org/gnu/glibc/glibc-2.17.tar.bz2
gluster-3.3.1	http://download.gluster.org/pub/gluster/glusterfs/3.3/3.3.1/glusterfs-3.3.1.tar.gz
gmp-5.0.4	http://ftp.gnu.org/gnu/gmp/gmp-5.0.4.tar.bz2
gpsd-2.96bis	http://download.berlios.de/gpsd/gpsd-2.96bis.tar.gz
http-2.25b	http://www.acme.com/software/thttpd/thttpd-2.25b.tar.gz
iptables-1.4.12.2	http://www.netfilter.org/projects/iptables/files/iptables-1.4.12.2.tar.bz2
kernel-3.2.14	http://kernel.org/pub/linux/kernel/v3.0/linux-3.2.14.tar.bz2
kmod-5	http://packages.profusion.mobi/kmod/kmod-5.tar.xz
ncurses-5.9	http://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz
ntp-4.2.6p5	http://www.eecis.udel.edu/~ntp/ntp_spool/ntp4/ntp-4.2/ntp-4.2.6p5.tar.gz
openssl-1.0.1e	http://www.openssl.org/source/openssl-1.0.1e.tar.gz
pcap-1.2.1	http://www.tcpdump.org/release/libpcap-1.2.1.tar.gz
pcmcia-018	http://ftp.cc.uoc.gr/mirrors/ftp.kernel.org/pub/linux/utils/kernel/pcmcia/pcmciautils-018.tar.bz2
portmap-6456249	http://neil.brown.name/git?p=portmap;a=snapshot;h=64562491929464207b8048501a317c6c5df83bb6;sf=tgz
readline-6.2	http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz
ssh-6.2p2	http://mirror.aarnet.edu.au/pub/OpenBSD/OpenSSH/portable/openssh-6.2p2.tar.gz
sysfs-1.1.0	http://www.kernel.org/pub/linux/utils/kernel/hotplug/sysfsutils-1.1.0.tar.bz2
udev-181	http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-181.tar.bz2
xml2-2.9.1	ftp://xmlsoft.org/libxml2/libxml2-2.9.1.tar.gz
xz-5.0.4	http://tukaani.org/xz/xz-5.0.4.tar.bz2
zlib-1.2.8	http://zlib.net/zlib-1.2.8.tar.gz