THE UNIVERSITY OF MELBOURNE

Research Cloud Node Implementation Plan

Document Identification	
File name:	NeCTAR Research Cloud Node Implementation Plan 2.5.pdf
Version:	2.5
Status:	Draft
Date:	18/10/2011
Prepared by:	Tom Fifield
Contributors:	Jesse Andrews, Lennie Au, Will Belcher, Sean Crosby, Tim Dyce, Kingsley Elliot, Nick Golovachenko, Steven Manos, Glenn Moloney, Mark Munro, Rajesh Padmawar, Sina Sadeghi, Barton Satchwill, John Shillington, John Spencer, Everett Toews

Table of Contents

Document purpose

Aim

This document outlines the milestones and activities in order to build the NeCTAR research cloud node at The University of Melbourne. The NeCTAR Research Cloud Node project will inaugurate a production cloud computing and object storage service. NeCTAR will contribute $1.5M to capital purchase and development of the cloud, with the University of Melbourne exceeding this in operational support, in addition to a $350K capital co-contribution. This document aims to detail the requirements, design goals and milestones of this cloud node. It takes into account the selection of the OpenStack cloud framework by the NeCTAR Research Cloud Technical Working group.

Scope

The document assumes a basic knowledge of Virtualisation and Infrastructure-as-a-Service style cloud computing, any description of this is out of scope of this document.

Disclaimer

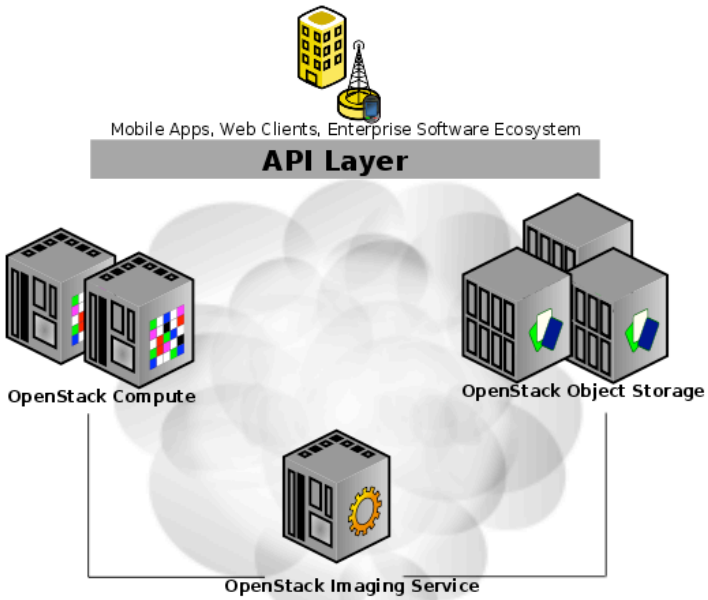
In this development activity, the University of Melbourne budgeted for and conducted several steps that would normally be covered by the lead node. An effort has been made to note where this is the case, however the primary reference to determine whether steps are required is the NeCTAR Request for Proposal document.

Designing the Architecture

Introduction to OpenStack

OpenStack is an open source framework designed to facilitate the construction of private or public clouds. It provides an Amazon Web Services compatible API to (currently) three main services:

- OpenStack Compute ("Nova") – provides all features necessary to operate an Infrastructure-as-a-Service style cloud, such as managing running virtual machine instances, networks
- OpenStack Object Storage ("Swift") – a reliable service to store large, persistent binary files
- OpenStack Imaging Service ("Glance") – a full-featured virtual machine image catalogue and distribution system



All three components will be used in this cloud node build.

OpenStack is under constant development, and many significant changes are contributed daily to their source code repository. OpenStack works on a 6-monthly release cycle, and this implementation is based on the Cactus release, which was released in April 2011. The next scheduled release ("Diablo") is slated for October 2011. Future developments have been taken into account in this architecture and are included in the milestones outlined at the end of this plan.

General Components

The first decisions in the architecture were ones that affected every component. These are listed below.

Operating System

OpenStack currently runs on Ubuntu and the large scale deployments running OpenStack run on Ubuntu 10.04 LTS, so deployment-level considerations tend to be Ubuntu-centric. RHEL (Red Hat Enterprise Linux) is supported, but to a much lesser degree. There is user-community documentation detailing how to install the middleware on other Linux distributions, however this is sporadic and generally not production-worthy.

- We have followed the advice and are using Ubuntu 10.04 LTS Server for all components, especially as:
  - RHEL will undergo significant change in the virtualisation stack in the near future
  - RHEL is not well tested with OpenStack (only tested with KVM)
  - RHEL has some feature limitations (e.g. limited iSCSI support)
  - RHEL RPMS come from a third-party repository
  - RHEL is not supported by OpenStack deployment tools

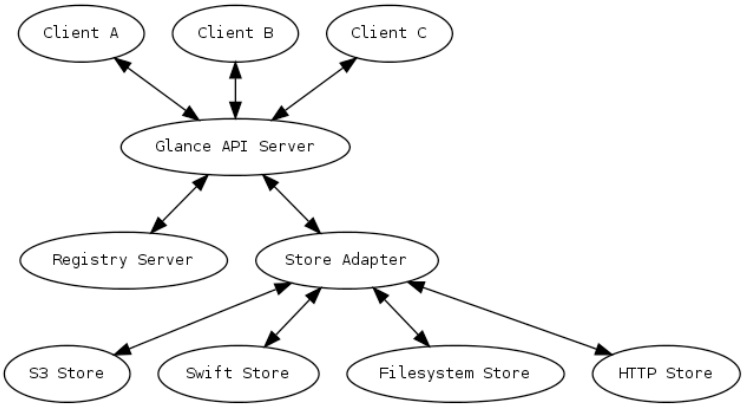
Database

For OpenStack Nova and Glance, you need access to either a PostgreSQL or MySQL database. We have selected MySQL, as:

- It is the reference implementation for OpenStack
- PostgreSQL is unsupported by OpenStack deployment tools
- PostgreSQL not as well tested with OpenStack

We also investigated the use of PostgreSQL and it was possible to use with some configuration changes, however it was not straightforward.

Deployment of Glance



The OpenStack project provides a component called Glance, which is a registry and delivery service for images stored on a number of back-ends (known as 'Stores').

It currently supports references to a simple local file system, the OpenStack Swift Object Store, Amazon S3 and read-only access to external images via HTTP.

Glance itself does not support authentication (though this is planned), however, this can be easily managed using the properties of the Store – Glance is merely a catalogue of images, and the ability to use them is determined by the Store itself.

Initially, a single Glance server will be sufficient for the imaging needs of the Node. As the Node will contain an Object Store powered by Swift, this is used as the default option for image storage and retrieval, with Glance cataloguing all associated metadata.

The file system store, due to its considerably poorer performance is best avoided for production installation, but has been useful in simple testing during development prior to the Swift service being operational.

For user communities that wish to migrate from Amazon EC2, the S3 backend could be supported. However, due to the high latencies and bandwidth costs associated with retrieval from Amazon, creating local copies of images (onto the Swift service) is recommended.

Similarly, there is the potential for user communities that wish to use images produced by a third party, made available by HTTP. Depending on the release cycle and network location of the image, the support for this option should be considered as viable for seeding new images (as the HTTP-based store is read-only).

As Glance is used for delivery of images from Swift Storage, it should be hosted on a machine with significant connectivity to both the Nova and Swift clusters. The Swift Proxy server (noted in later sections) is well suited to also host glance. This results in effective use of hardware.

Additionally, the load on glance is mitigated by the image caching capability of Nova. Nova caches VMs that have been recently launched. As we will be using a shared filesystem (described in the Nova installation section), this means that starting hundreds of virtual machines using the same image has minimal impact on glance.

Use of Virtual Machine Images

Allowing flexible use of virtual machine images is one of the benefits of infrastructure-as-a-service cloud computing, however it is important to note the roles and responsibilities to do with maintenance, especially in a security context.

The following scenarios will be supported:

1. User creates/maintains own image
    - In this scenario, the user of the cloud creates their own image, without the view of sharing it with the community widely. They will be responsible for maintaining their own image and ensuring the security.
2. Research Group Manager creates/maintains image for Users
    - A Research Group has a staff member proficient in Information Technology, who does not actually use the image for research. Responsibility for the general security of the image is delegated to the manager who created it, who will need to maintain the image. However, individual users of the image also have the ability to impact security, and are also responsible for the impact of any changes they make.

3. Community partner creates/maintains image

    - There are certain software packages widely used by the research community, where it is sensible to have a centrally maintained image for easy deployment. In this case, community partners can volunteer to maintain particular images. For example, VeRSI could maintain a Sakai image, or some other institution could maintain a Matlab image that can be used nationally (Software as a Service).
    - In this case, responsibility for the general security of the image is delegated to the community partner who created it, who will need to maintain the image. However, individual users of the image also have the ability to impact security, and are also responsible for the impact of any changes they make.

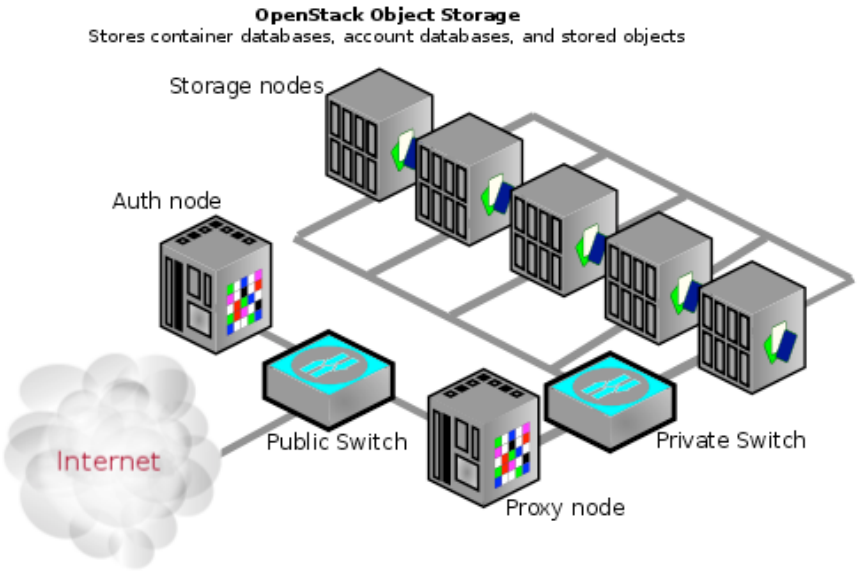
4. User wishes to use image created by party outside the scope of NeCTAR

    - There is a vast array of machine images hosted in the greater Internet, many which could be useful to Australian researchers. A well-defined security update policy may not be present in the upstream provider, so special care needs to be taken in this case.
    - One solution is: the user who uploads the image to the NeCTAR Virtual Machine Image Repository is responsible for the image's general security – they should conduct a brief assessment (for which assistance should be provided) before uploading the image. Then, users of the image have greater responsibility than normal to ensure that security best practice is followed, in addition to their regular usage responsibilities.

Deployment of Swift

The OpenStack Object Storage system - known as Swift - contains three core node types:

- Auth node – which is responsible for access, authentication and authorisation. Only one auth node is required
- Storage Node – which contains the actual storage, and services to provide it. A minimum of five are required, but scalable to N.



- Proxy Node – which is responsible for the management of Storage Nodes, and through which all data flows. A minimum of one is required, but scalable to N.

Network

Swift requires two networks – a public facing network, through which the Proxy and Auth nodes are accessible to the Internet, and a higher bandwidth private network, through which the Storage Nodes communicate. By default, the bandwidth required between the Storage nodes and their associated proxy node should be a factor of three larger than between the Internet and the Proxy.

Swift requires that two storage nodes successfully write an object before returning success (in the default configuration where there are three replicas). This second copy is essential, because a single hardware failure immediately after a successful write to a single object server may cause the data to be permanently lost.

Storage nodes interact with typically insecure protocols (such as rsync without authentication), and the private network should not be Internet accessible. However, it is still recommended that the storage nodes have a secondary network for management purposes. We implemented our storage with a separate public, private and management network.

Hardware

OpenStack Object Storage specifically is designed to run on commodity hardware. At Rackspace, Storage nodes are currently using generic 4U servers with 24 2T SATA drives and 8 core CPUs. The smaller the storage node, the better the redundant distribution and performance can be. We aimed for a similar configuration to Rackspace, scale in groups of 5 Storage Nodes and 1 Proxy Node:

We opted for a smaller size node to lower the initial entry cost (recall – 5 nodes are required), and a finer granularity of storage.

1 Processor(s) from the X86_64 family
2 At least 24GB of ECC RAM, with capacity for expansion
3 10 GB network interface, which:
    1. Is compatible with Cisco 10GBASE-SR modules
    2. Supports PXE boot
4 A system storage array which:
    1. Has a capacity of 120GB or more, where capacity is measured post RAID, as actual available capacity

2. Is configured RAID1
3. Is hot-swappable

5 Data storage arrays of disks which:
1. Provide total storage of at least 12TB per unit, and a maximum of 24T of storage per unit (excluding the system array).
2. Are presented as separate physical disks – please note that RAID is not required and not a supported configuration.
3. Are hot-swappable

6 Remote management and monitoring capabilities; specifically, remote console
7 Hardware which is compatible with Ubuntu Linux
8 Redundant hot swap power supplies
9 Rails to mount the unit into a 19" EIA-310-D compliant rack

The Proxy node requires significant amounts of bandwidth available recommend: 2x10Gbit – at least 3 times the bandwidth of a single storage node. The Proxy is very I/O focused, and this should be maximised. As such, the requirements for a proxy node are:

1. 2 or more processors from the X86_64 family; Intel or AMD
    a. At least 12 cores
2. At least 24GB of RAM; with facility to expand further
3. Two or more ten-gigabit Ethernet connections, which
    a. Is compatible with Cisco 10GBASE-SR modules
    b. Support PXE boot
4. A system storage array which
    a. Has a capacity of 300GB or more, where capacity is measured post RAID, as actual available capacity
    b. Is configured RAID1
    c. Is hot-swappable
5. Remote management and monitoring capabilities
6. Redundant, hot-swappable power supplies
7. Rails to mount the unit in a 19" EIA-310-D compliant rack
8. Hardware which is compatible with Ubuntu Linux

Both node types can further be extended with:
1. Additional RAM
2. Higher performance HDDs such as 15K SAS, SSD etc.

The Proxy node requires significant amounts of bandwidth available recommend: 2x10Gbit – at least 3 times the bandwidth of a single storage node. The Proxy is very I/O focused, and this should be maximised.

File System

OpenStack Storage works best when it has low-level access to disks, as it manages the distribution of data across multiple storage servers and disks internally. RAID on the storage drives is not required and should be avoided, as this will result in the loss of the relationship between stored data's location and individual partitions. Swift's disk usage pattern is the worst case possible for RAID, and performance degrades very quickly using RAID 5 or 6.

XFS is recommended as the file system, as it demonstrated the best overall performance for the Swift use case after considerable testing and benchmarking at Rackspace.

We use XFS, and have seen reasonable performance.

Segregation of OpenStack services

In addition to the three node types above, OpenStack has the facility to separate individual services (such as the splitting the account, container and object services associated with Storage nodes). However, Rackspace's deployment which is significantly larger than that of NeCTAR finds this unnecessary, and instead installs all services on all Storage nodes – scaling out horizontally in this way. It is similarly trivial

to combine the Proxy and Auth nodes, though in the NeCTAR case, separation of these is advised due to the potential to investigate additional authentication schemes such as Shibboleth.

In summary:

- All Storage node services on all Storage nodes
- Initially, one Proxy server to five Storage nodes (we have 8 storage nodes, so two proxies)
- Separate Auth server (hosted on NSP)

For improved performance of the Container service, SSD technology can be used. Whether this is required depends on the use cases (e.g. frequent listings of extremely large directories, or lots of metadata access)

Load Balancing of Proxy Services

As all traffic from the Object Store flows through the Proxy node, this can become a choke point for data access. Thankfully, Swift is well designed to allow load balancing of the proxy services. This can be achieved using two main methods: DNS round robin, or the use of a load balancer such as Pound[1], or the University's existing F5 system.

One proxy server for every five storage nodes will be purchased to allow for sufficient bandwidth, then the best of the above methods will be employed to load balance the service.

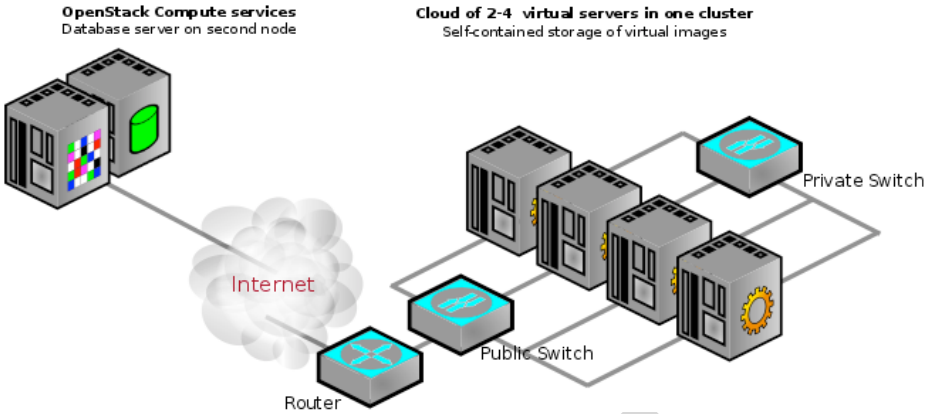
Redundancy of Storage

Swift requires that 2 storage nodes successfully write the object before returning success (in the default configuration where there are 3 replicas). The third copy will be attempted, but if it fails, replication will handle creating it on the third object server. The second copy is essential, because a single hardware failure immediately after a successful write to a single object server may cause the data to be permanently lost.

The default configuration of three replicas is used (recall: RAID is not used, so RAW capacity is the same as available capacity).

1 http://www.apsis.ch/pound/

Deployment of Nova



The OpenStack Compute system, known as Nova, consists of separate control nodes and Compute nodes (which run virtual machines). Each Compute node can run multiple guest Virtual Machines, while Control node services can be run together, or segregated (displayed above, two control nodes – one with Nova services, the other with the database, and 5 Compute nodes).

Hardware

The pilot currently uses:

7 Dell R815s (336 processors total), each with

- Processors from the X86_64 family; Intel or AMD
  - Support hardware virtualisation extensions
- 5GB or more of ECC RAM per core; with facility to expand this further (eg 48 cores, 256GB RAM per node)
- One 10-gigabit Ethernet connections, which
  - Is compatible with Cisco 10GBASE-SR modules
  - Support PXE boot
- A system storage array which;
  - Has a capacity of 120GB or more, where capacity is measured post RAID, as actual available capacity
- 48 cores (AMD Opteron 6176)
  - ➔ Is configured RAID1
  - ➔ Consists of at least 2 spindles
  - ➔ Uses software or hardware RAID (hardware RAID preferred)
  - ➔ Is hot-swappable
- At least basic remote management and monitoring capabilities
- Rails to mount the unit in a 19” EIA-310-D compliant rack
- Hardware which is compatible with Ubuntu Linux

Several expansion options are available for this specification:

- An unused PCI-e 8x (or faster) slot for potential future network connectivity expansion
- High or low profile acceptable
  - Hardware RAID, and/or higher performance HDDs such as 10K SATA, 10K/15K SAS etc.
  - ➔ 256GBAdditional RAM
  - 146GB SAS RAID0 HDD
  - 2X10Gbit NIC
    - Enhanced remote management and monitoring capabilities

One of the key design features at the University of Melbourne is the reliance on an external SAN/NAS for providing virtual volumes and running-machine-image storage – hence the relatively low internal storage requirements. If designing a node where external storage is not used, the guiding principle is to provide as many spindles as possible to the compute node.

Similarly, contention between number of VMs and network bandwidth should be considered. We estimate that at most 48 cores should be attributed to a single 10gbit connection. Actual will guide bandwidth requirements here – however allowing for provision of additional 10gbit connections is wise. We have

allowed for an extra 10Gbit connection per node, and have a spare slot to insert an extra card should the need arise.

Note that Control Components – such as the nova-* services with the exception of compute are hosted on the National Servers Program.

Network

OpenStack provides three networking options:

- flat – all IPs are fixed & injected via the file system into Virtual Machines, all VMs are on the one VLAN
- flatDHCP – similar to 'flat', but a DHCP server is used to assign addresses
- VLAN – uses host-based 802.1q tagging on compute nodes to provide separate VLANs for each project. DHCP is used to assign addresses

For simplicity resulting in rapid deployment flatDHCP is recommended. Due to the presence of a DHCP server, the compute nodes will need to be on their own network subnet, with broadcast traffic segregated from the rest of the network.

After further communication with research communities surrounding their application use cases, the VLAN network deployment model could be seriously investigated. This would facilitate the creation of so-called virtual data centres, where all VMs in a project can communicate privately with each other.

However, the VLAN deployment model has an important caveat – all traffic to the guest VMs is routed through the host running nova-network. High Availability and Load Balancing options for the nova-network service **(see segregation of OpenStack services)** are not yet at a mature state. As such, any investigation into this option should wait for appropriate middleware development.

Storage

There are several items that require storage space with good I/O performance available on the compute nodes (in our case provided via SAN):

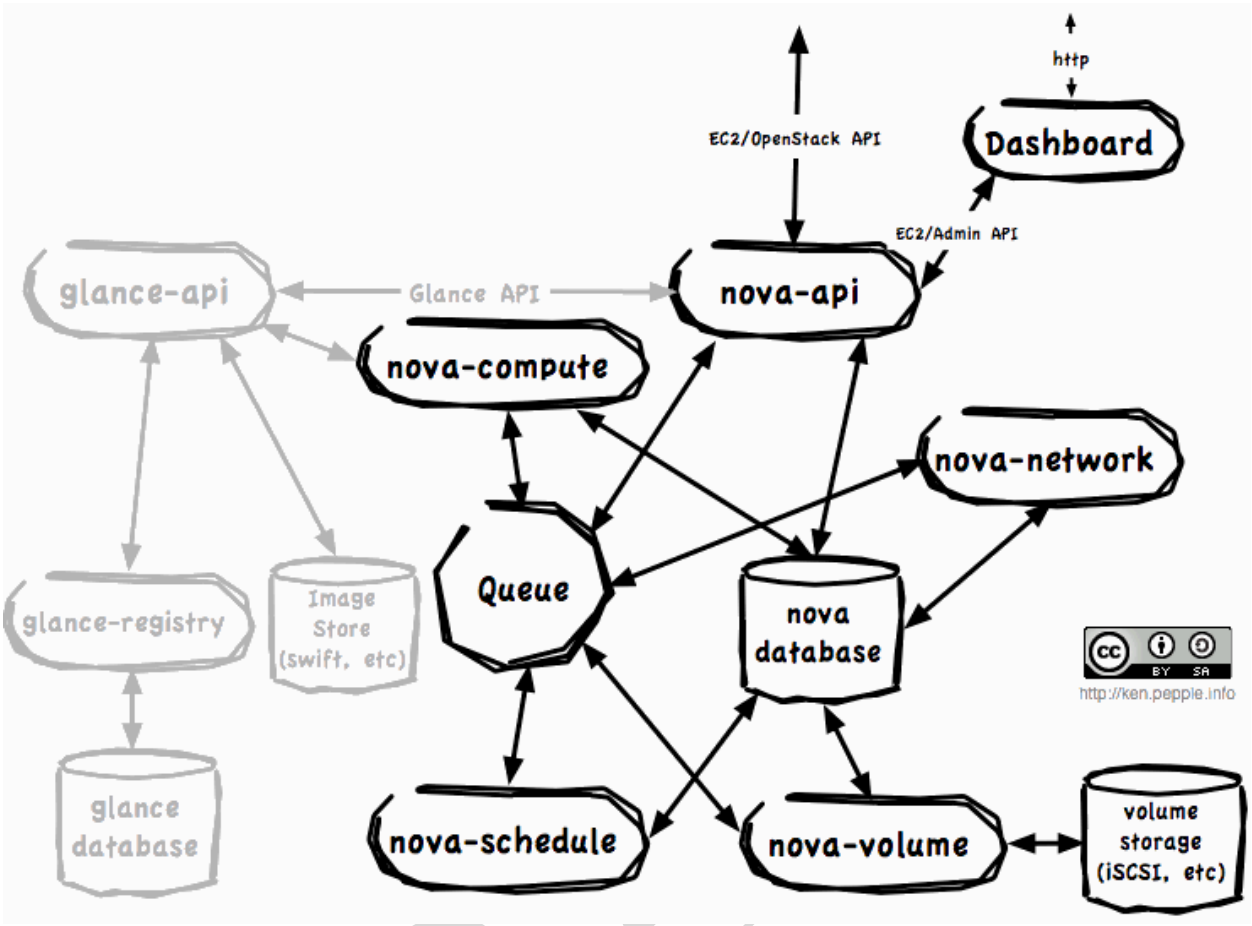
- Image space for Instantiated virtual machines (maximum 10GB per VM in our case)
- Space for instantiated VM's associated storage allocation (e.g. 60GB/VM)
- Space for any attached volumes i.e. Elastic Block Store (large capacity requirements)
- Cached VM Images - nova caches VMs that have been recently launched (10GB/VM depending on churn

The limiting factor in Virtual Machine I/O performance is the disk on which the instantiated Virtual Machine runs, in addition to the usage of any elastically-attached storage – commonly known as Elastic Block Storage. A reasonable approach to conquering any bottlenecks is to have a reasonable spindle to core ratio. In our case, we found it more cost effective to leverage an existing peta-scale SAN, rather than specifying large numbers of hard disks in every compute node. This gives us more control and storage capacity, at the expense of some network overhead performance impact.

Segregation of OpenStack services

Aside from the division of Computing nodes and Control infrastructure, OpenStack has the facility to separate individual control services – as in the below diagram. A typical deployment separates the RabbitMQ Queue and MySQL Nova Database onto separate machines, while running other services on the one. This approach is recommended with one addition – also running the dashboard on a separate machine. Due the extra development envisaged here, greater flexibility in update would be a bonus.

Each of the control services is hosted on the National Server Program.



Hypervisor

The default hypervisor of Nova, the one which is well documented, receives most testing and supports the most features is KVM. As a result, the NeCTAR Research Cloud will use KVM.

User Role Management

A common thread through all OpenStack components is OpenStack Compute's rights management system. The system supports four roles detailed below, which seem well suited for our purposes. First, installation-wide roles:

- Cloud Administrator (admin): Users of this class enjoy complete system access.
    - ➔ This can be used for System Administrators working on the NeCTAR cloud

Next, a core principle is projects, which are isolated resource containers forming the principal organizational structure within Nova. Each project has a separate volumes, instances, images, keys, users (and potentially VLANS).

For projects, quota controls are available to limit the:

- Number of volumes which may be created
- Total size of all volumes within a project as measured in GB
- Number of instances which may be launched
- Number of processor cores which may be allocated
- Publicly accessible IP addresses

Roles per-project include:

- Project Manager (projectmanager): The default for project owners, this role affords users the ability to add other users to a project, interact with project images, and launch and terminate instances.

    ➔ This role should be assigned to those responsible for creating and maintaining images, whether IT staff or a researcher themselves

- Network Administrator (netadmin): Users with this role are permitted to allocate and assign publicly accessible IP addresses as well as create and modify firewall rules.

    ➔ Researchers who require broad network access should be freely allocated this role

- Developer (developer): This is a general purpose role that is assigned to users by default.

    ➔ Researchers that do not require broad network access and use images created by others can be allocated this role

A user can specify which project he or she wishes to use by appending :project_id to their access key. If no project is specified in the API request, Nova attempts to use a project with the same id as the user. This could be an important training point for users working on multiple projects.

Presently this system does not support shibboleth-based authentication (required for AAF integration). However, it follows a modular design – shibboleth support could be developed, or a bridge using the existing LDAP functionality investigated.

Access Methods

The means through which users will manage images and control virtual machines is of high importance to this project. Though often deemed less critical than the robustness of the service itself, the User Interface is high visibility – meaning a poor experience here alters perceptions of the service, regardless of the actual robustness.

HybridFox

HybridFox is a cross-platform plugin for the Firefox web browser that provides a graphical interface to an EC2-compatible cloud. The plugin is based on ElasticFox, which is in wide-ranging usage by users of the Amazon EC2 cloud. The plugin has a wide feature set, allowing for full control of instances, images, keypairs, security groups, volumes and others that OpenStack does not yet support. It is actively developed by Amazon staff, and then modified by the Eucalyptus team.

It is a trivial exercise to configure HybridFox to support the NeCTAR cloud, and the plugin could even be modified and packaged with the NeCTAR availability regions, support HybridFox as an access mechanism, and have packaged a NeCTAR-configured version.

OpenStack Dashboard

Currently under development, and available only from a source code repository, the OpenStack Dashboard is the framework-specific user interface for OpenStack. A user is currently able to:

- manage existing images
- create and delete keypairs
- attach and create volumes
- launch instances

within the confines of their project's quota. Significant development effort is going into this software, and it will form an integral part of the Diablo release slated for October 2011. In addition to the user-facing functionality, significant amounts of administrator-level configuration options will be presented via the OpenStack Dashboard.

The OpenStack Dashboard is written in Python, in the Django framework and is relatively simple to extend.

The OpenStack Dashboard is not yet mature enough for direct user access. In addition, it is a framework-specific technology – which is best avoided lest we need to change framework. However, it will form an important part for administrators, so needs to be monitored for developments.

Deployment

The architecture of the Node is broken up into 11 basic hosts (of which some have multiple instances, for scalability).

Computing

1. Cloud Controller (Nova)
2. Database Server
3. Messaging Server (RabbitMQ)
4. Image Server (Glance)
5. Compute Nodes x7 (nova-compute)

Storage

6. Storage Proxy x2
7. Storage Auth (combined with proxy)
8. Storage Nodes x8

- System Management

9. Build Server (TFTP, DHCP, puppet)
10. Syslog Server
11. Backup Server

Datacentre deployment

The University of Melbourne is fortunate to have three high-quality datacentres on its asset register. These are comprised of a facility at Noble Park – 30km to Melbourne's South East, and two diverse datacentres on the Parkville campus to Melbourne's North, known as Data Hall 1 and Data Hall 2.

In order to best exploit the redundancy provided by this infrastructure, all three will be used[2]. In general, Compute infrastructure will be in Data Hall 2 and Noble Park equally, with Object Storage 20% in Data Hall 1 and 40% in each of Data Hall 2 and Noble Park. Within each datacentre, diverse racks are used to minimise the reliance on a single point of failure.

Management Infrastructure

The use of an automated deployment and configuration infrastructure is essential to minimise the operational cost of running the cloud offering. A system to automatically install operating system and provide an initial configuration is detailed in the Automated Deployment section. A system to coordinate the configuration of all services automatically and centrally, in addition to the configuration items is defined in the Automatic Configuration Management section.

It is envisaged that this system can be used at other nodes, to streamline management and deployment of configuration changes and updates.

Automated Deployment

The aim of the automated deployment system is, after the most minimal manual work possible (physical racking, MAC ↔ IP assignment, power on), new servers automatically have their operating system installed and are configured and ready for use without intervention. Much work in the field of OpenStack has gone into this area. Typically solutions rely on wrappers around PXEBoot and TFTP servers, for the basic Operating System install, then hand off to an automated configuration management system.

[2] in the full node, initially, only one datacentre is used

An Automated Deployment System for our purposes has been investigated and setup.

Automatic Configuration Management

The purpose of automatic configuration management is to establish and maintain the consistency of a system with no human intervention. Proper use of automatic configuration management tools provides surety that components of the cloud systems are in particular states, in addition to simplifying deployment, and configuration change propagation.

Conveniently, a large body of work has been done by the OpenStack community in this space. Puppet – a configuration management tool – is even provided as an official deployment mechanism for OpenStack[3]. Additionally, given experience in the Australian eResearch community with puppet, it is recommended that this deployment option be extended to facilitate an automated configuration management system.

An integral part of a configuration management system is the items that it controls. The following is a list of configuration items for hosts:

- packages – software applications installed on nodes
- apt.conf.d – ubuntu package manager configuration
- nova.conf – for configuration of Nova & related services (e.g. glance)
- deploy.conf – for Puppet automated deployment of Nova
- glance.conf – for configuration of the OpenStack Imaging Service
- swift.conf – configuration of the the OpenStack Storage Service
- /etc/swift/object-server.conf – configuration for the OpenStack Swift Object Server
- /etc/swift/container-server.conf – configuration for the OpenStack Swift Container Server
- /etc/swift/account-server.conf – configuration for the OpenStack Swift Account Server
- /etc/swift/proxy-server.conf – configuration for the OpenStack Swift Proxy Server
- Swift Ring configuration files
    - For the ring, Rackspace specifically recommends administrating the ring on one server and then distributing the ring files out to all of the nodes (proxy and storage)
- network and firewall configuration
    - SSH host keys and daemon configuration
- host certificates
- NTP configuration
- logging configuration
- memcached configuration
    - /etc/sysctl.conf – configuration of kernel parameters
- nagios checks and user

Central Logging

The provision of centralised service logging ensures additional security against measures such as log tampering, and facilitates ease-of-audit. All OpenStack components and system logs are:

- Using rsyslog with stunnel to pipe log files back to a central secure logging service, for audit trail.

Central Monitoring

We run a Groundwork-powered nagios server that monitors all services on all hosts for incident.

[3] http://wiki.openstack.org/NovaInstall/NovaDeploymentTool

Operational Activities

Note first that Operational Activities are not fundable under EIF rules for the University of Melbourne NeCTAR Research Cloud node. All operational activities listed are funded using in-kind contribution from the University of Melbourne.

- Direct user support and training for this project will be coordinated by the NeCTAR Research Cloud Lead Node – it is envisaged that a support plan will be drafted by the Lead Node in conjunction with consultation with the sector
- In general terms, this system will be operated by a team of three system administrators who work in Australian Eastern Standard Time business hours. In addition, an on-call roster will allow for out-of-hours intervention should a situation arise that threatens user data integrity.
- The node will continue to operate parallel test/development and production infrastructure from the go-live date.
- Task attribution, including shared tasks with the National Servers program staff (marked *) is below:

Staffing Break-down

Task/Responsibility	FTE	Description
Infrastructure Management System	0.1	Maintain and extend the node infrastructure management system. E.g. build system, centralized logging
Network*	0.1	Make minor network changes and liaise with site network team to coordinate, plan and follow-up on major changes
Database	0.1	Maintain all databases, including migration and backup
Hardware*	0.1	Respond to hardware incidents, including component replacement and communication with vendors
User Interfaces	0.1	Take ownership for all User-facing services, such as dashboards and APIs
On Call Night Technician*	0.1	Respond to data-loss critical incidents with hardware or services during non-business hours.
Security*	0.1	Monitor and enforce security policy, including responding to security incidents
Virtual Machine Image Catalogue	0.2	Take ownership for the Virtual Machine Image Catalogue service, including its interface with back-end storage
Storage – Cloud Virtual Volumes and related SAN/NAS	0.2	Manage the interface between the Cloud Compute offering and back end storage used to support it – virtual volumes and scratch space
Upgrades	0.3	Investigate, test and plan the deployment of upgrades
Storage – Object Store	1	Take ownership of Object Storage service, including all storage nodes and proxy nodes.
Cloud – Compute Offering	1	Take ownership of Cloud Computing service, including all compute nodes and control infrastructure.

The node is responsible for

- Management of host hardware
- Power, cooling and environmental management for the host hardware
- Network infrastructure within the data centre, cabling infrastructure and storage
- Networking configuration
- Network security and physical security
- Managed virtual server infrastructure
- Researcher access to relevant virtual server guest management components
- Researcher access to virtual server guests
- Monitoring virtual server infrastructure

Service Interdependencies

NeCTAR National Servers Program

The National Servers Program (NSP) – the other NeCTAR infrastructure sub-project – provides a highly robust, redundant hosting platform for services of national significance. The Research Cloud is one of the first customers of this – with the control infrastructure being hosted on the NSP.

In addition, there is a degree of migration expected between the platforms. For example, a highly-used service developed on the Research Cloud could eventually meet the requirements of NSP hosting. Alternately, the Research Cloud can be used to support NSP services requiring additional resources for periods of time.

Finally, there are strong parallels between the operations of the NSP and the Research Cloud, and it would seem efficient that some degree of staff cooperation occur.

University of Melbourne Research Storage

The University of Melbourne is fortunate to have a large, robust, reasonably priced storage offering to provide to researchers. As mentioned in 'Deployment of Nova", this external SAN will be used to provide virtual volumes, and storage for running virtual machine instances. This has enabled the hardware specification for compute to require significantly less storage, resulting in greater value for money.

Research Data Storage Infrastructure project

The Research Cloud will not provide large-scale permanent or archival data storage or archival facilities. However, it is foreseen that this will be covered by future RDSI initiatives.

High Performance Computing facilities

High Performance Computing facilities for research are provided by many institutions: the University of Melbourne, state Partnerships for Advanced Computing and the National Computing Initiative. The Research Cloud is a good compliment to these traditional batch processing systems. It facilitates the provisioning of computing 'applications', and as noted in the NEON report can free up cycles on HPC resources currently taken up by tasks which don't require the same level of parallelism.

Governance

Governance arrangements are outlined in detail in the NeCTAR Project Plan for the Research Cloud. In summary however, there will be two groups with national representation:

- NeCTAR Platforms Steering Committee, which will provide oversight and strategic guidance to the participants in the NeCTAR Research Cloud and National Server Programs to achieve the NeCTAR Project objectives.
- NeCTAR Platforms Technical Advisory Group, which will provide expert technical advice to the NeCTAR Platforms Steering Committee, the Program Lead Nodes and the Program Nodes on matters of infrastructure architecture, implementation and operation.

Deliverables & Milestones

Milestone	Planned Date	Required Accuracy	Governance	NeCTAR CAPEX	UoM CAPEX and co-investment
Gate 1 – Feasibility	27/05/2011	+/- 10%	ITS Portfolio / Initiation Governance	Nil	35,000
Pilot Go-Live (Production node with limited service and simple monitoring in place)	15/08/2011	+/- 30%	Project Steering Group	Nil	295,000
Gate 2 – Project Brief	13/09/2011	+/- 30%	ITS Portfolio / Initiation Governance	Nil	20,000
Parkville Node Go-Live (Production node with redundant data centres)	13/01/2012	+/- 30%	Project Steering Group	1,113,000	135,000
Stage 1 Closure Completed Documentation, preparation for external development (e.g. 2nd node)	13/01/2012	+/- 30%	Project Steering Group	-	-
Enhanced Parkville Node (Additional compute/storage)	30/03/2012	+/- 30%	Project Steering Group	387,000	200,000
Upgrade to OpenStack ESSEX	29/06/2012	+/- 30%	Project Steering Group	Nil	200,000
Post-Implementation Review	28/09/2012	+/-30%	Project Steering Group	Nil	200,000
Upgrade OpenStack F Release	15/12/2012	+/-30%	Project Steering Group	Nil	155,000
Uptake of 100 users Security Audit Review	29/03/2013	+/- 30%	Project Steering Group	Nil	215,000
Upgrade OpenStack G Release	28/06/2013	+/- 30%	Project Steering Group	Nil	215,000
Operational Phase Review	27/09/2013	+/- 30%	Project Steering Group	Nil	215,000
Upgrade OpenStack H Release	16/12/2013	+/- 30%	Project Steering Group	Nil	215,000
Node Sustainability Plan in operation	30/03/2014	+/- 30%	Project Steering Group	Nil	292,500
Project Closure	30/06/2014	+/- 30%	Project Steering Group	Nil	292,500

Gate milestones correspond to approval gating within the project framework in ITS at the University of Melbourne.

The project steering group will be formed upon commencement of the project once approved. The above timelines are aggressive and can be shifted depending on the timing of the NeCTAR/DIISR approval cycle.

Budget

Project Costs

The source of the project funding below is NeCTAR EIF capital fund and The University of Melbourne CAPEX co-investment.

Project Estimated Costs (+/- 30% tolerance)					
	2011 $	**2012 $**	**2013 $**	**2014 $**	**Total $**
NeCTAR funding					
Salaries	220,000	70,000	Nil	Nil	290,000
Supplies	8,000	2,000	Nil	Nil	10,000
Services	85,000	15,000	Nil	Nil	100,000
Expensed Assets	800,000	300,000	Nil	Nil	1,100,000
Sub Total	*1,113,000*	*387,000*	*Nil*	*Nil*	***1,500,000***
UoM CAPEX co-investment					
Salaries	65,000	Nil	Nil	Nil	65,000
Expensed Assets	275,000	Nil	Nil	Nil	275,000
Sub Total	*350,000*	*Nil*	*Nil*	*Nil*	***350,000***
Total					**1,820,000**

Operational Costs resulting from Project outputs

The source of the operational funding below is The University of Melbourne.

UoM Operational funding co-investment Estimated Cost (+/- 30% tolerance)					
	2011 $	**2012 $**	**2013 $**	**2014 $**	**Total $**
Salaries	15,000	350,000	455,000	235,000	1,055,000
Supplies	Nil	Nil	Nil	Nil	Nil
Services (UoM in-kind)	120,000	405,000	405,000	350,000	1,280,000
Expensed Assets					
Income:					
Sub Total	135,000	755,000	860,000	585,000	***2,335,000***
Total					**2,335,000**

Service Definition

The Research Cloud will provide Australian Researchers with access to virtualised resources. At a glance, the proposed cloud computing service offering will be:

- The availability of three different types of virtual machines
  - **Small**: 1 core, 4GB RAM, 30GB of instance storage
  - **Medium**: 2 cores, 8GB RAM, 60GB of instance storage
  - **Extra-Large**: 8 cores, 32GB RAM, 240GB of instance storage
- Each virtual machine will have a public IP address
  - Includes unlimited on-net traffic
  - 1GB of off-net traffic per core per month
- Additional storage ("virtual volumes") can be added to virtual machines
  - Additional storage is purchasable in blocks of 250GB, for a fee.

In addition, an Object Storage solution will provide facilities for hosting Virtual Machine Images, and other items of large small-write, high-read data, with an emphasis on read performance and resilience against failure.

This will be further refined in consultation with the sector as well as potential users of the Research Cloud.

Limitations

The maximum size of a Virtual Machine image is 10GB

- Encourages researchers to treat data separately to Virtual Machines
- Facilitates sharing of generic Virtual Machines
- Results in faster loading of Virtual Machines
- Less impact on the Virtual Machine distribution network

The maximum Internet bandwidth currently available from any node is 10Gbps, shared between all virtual machines at that node. The maximum expected deliverable bandwidth to any individual node is 250Mbps on average, bursting to 10Gbps.

The maximum number of public IPv4 addresses that can be allocated to an instance is one. There is a limited pool of public IPv4 addresses available worldwide, and use of IPv6 is encouraged.

Access Methods

The research cloud will need to be easily accessible to first-time cloud users, with other methods of access available. These methods will be periodically reviewed based on demand and uptake.

Four different processes that can be used to access research cloud resources are detailed below:

Research Cloud instant access

Instantaneous access without intervention to a low-specification virtual machine and associated storage is available. A researcher merely needs to login with their AAF credentials (and others in due course as required), and provide a few sentences about their aims. The purpose of the free tier is to allow researchers to explore the cloud, and perform some limited testing and development with a very low barrier to entry.

- The Free Tier consists of
  - 2250 hours (or three months) of runtime of one small type virtual machine

    - 20GB of Object Storage – aimed at storing virtual machines
- Available per individual researcher to all research institutions in Australia

It is envisaged that users of the Free Tier will migrate to another access method prior to the exhaustion of their free allocation.

NeCTAR Resource Allocation Process

A proposal from an individual researcher, a research group or a Virtual Laboratory can be submitted to the NeCTAR Resource Allocation Committee. The applicant will be required to provide details of their intended usage and expected resources, including:

- Primary Instance Type (Small, Large, Extra Large)
- Maximum number of Virtual Machines required
- Frequency of usage, or usage patterns
- A description of the use case
- Geographic requirements (e.g. located next to computational resources or near data-intensive research instruments), or Nationally.

Institutional Allocation

As planned at The University of Melbourne, and widely expected at other nodes, individual institutes have the ability to make capital purchases to add to the NeCTAR cloud. This will provide access to cloud infrastructure (corresponding to this co-investment) for those local researchers.

Researcher-funded Hardware

There may be cases where research groups have a requirement for significantly more resources than can be provided through the other access methods. In this case, research groups can provide funds to a NeCTAR node to have hardware – from a restricted list of choices – added to the NeCTAR cloud. The access model then follows a similar process to the Individual Institutes owning hardware – the owning research group gains priority access to an amount of resources proportional to their purchase. It will be up to each individual Research Cloud node to manage this process.

Target Service Levels

Though the cloud service is only supported (both user and systems support) during Australian business hours, it is expected that significant usage will occur outside of this time. As such, an on-call roster will allow for out-of-hours intervention should a situation arise that threatens user data integrity.

During the supported period, reliability is expected to be greater than 98% - or offline for less than 9.6 minutes of every working day.

Communication targets

As in the maintenance section - a set of maintenance windows, both for small and large maintenance tasks should be defined. For each of these windows, we can define a minimum amount of notice:

- For scheduled downtimes less than four hours in duration – one week
- For scheduled downtimes greater than four hours in duration – one month

In the event of an unscheduled downtime lasting greater than one hour, a Service Incident Report should be created and made available to the community.

References

- NeCTAR Research Cloud Technical Working Group Report
- NeCTAR Research Cloud Initial Node - Project Brief
- Dell Inc – Bootstrapping OpenStack Clouds
- OpenStack Compute, Object Storage, Image Catalogue Administrator and Developer Manuals
- An Internationally Distributed Cloud for Science - The Cloud-Enabled Space Weather Platform, Everett Toews, Barton Satchwill, Robert Rankin SECLOUD'11
- NEON – Northern Europe Cloud Computing - Final Report, Åke Edlund and Maarten Koopmans et al