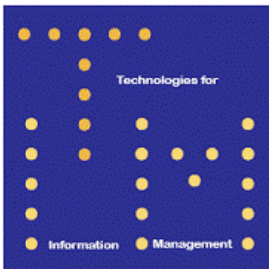
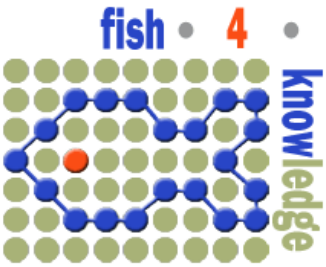




Fish4Knowledge Deliverable D4.3

Process Execution

Principal Author: NARL, UEDIN
Contributors: Gaya Nadarajan, Yu-Jung Cheng, Hsiu-Mei Chou, JihSheng Chang, Sun-In Lin, Bas Boom, Shi-Wei Lo, Fang-Pang Lin
Dissemination: PU

Abstract: The goal of WP4 is to maintain a sustainable infrastructure for marine ecology understanding. The infrastructure is composed of a number of video cameras continuously sending data stream, a massive storage system to store video and processed data, and a high performance computing facility to do data analysis. NARL has made major enhancements to the methods for data storage and capture to enable better quality of the (video) data acquisition. These changes include relocating the video server, increased resolution and frame rate of the video and the use of "stream dumps" for capturing videos. The overall storage has been increased by 220%. NARL created two sets of distributed computing platforms, a group of virtual machines and two dedicated nodes in a supercomputer to explore a variety of process execution flows. Both environments have been tested with fish detection and tracking and fish species recognition modules. NARL has a job dispatcher facility that interfaces the workflow with the Gridengine resource scheduler on the VM platform. This is planned to be extended to interface the workflow to the resource schedulers in both computing platforms.

Deliverable due: Month 24

Contents

1 Introduction

The goal of WP4 is to maintain a sustainable infrastructure for marine ecology understanding. The infrastructure is composed of networking components: a number of (e.g. 10) video cameras continuously sending the data stream, a massive storage system to store video and processed data, and a high performance computing facility to perform data analysis. In Section 2, the overview of F4K's hardware architecture is presented. NARL has provided hardware and computing facilities that consist of high performance computing machines that can be VM servers, storage devices for videos, computing nodes and database servers. These are connected via a high bandwidth network infrastructure.

NARL has made major enhancements to the methods for data storage and capture to enable better quality of the video data acquired (Section 2.1). These include changing the video server location, increasing the resolution and frame rate of the video and using "stream dumps" for capturing videos. 3TB sized RAID hard disks were deployed on state-of-the-art data sharing facilities for increased storage. The overall storage has increased by 220% from 62.2TB (April 2012) to 198.8TB (November 2012). We also deployed a Local Processing Server (LPS) to overcome network bandwidth hurdles. A suitable video format (MPEG4), bitrate and indexing of videos in the database have been selected to improve the quality of the data as much as possible without compromising too much time, compute power or storage. Security measures are in place to protect the data and resources from public attack. Plans are in place to increase the network infrastructure to 10GB.

To fulfill the requirement of compute power, NARL built a high-end tera-scale data storage and created two sets of computing platforms to explore a variety of process execution flows (Section 2.3). One is a group of virtual machines and the other is a multi-core supercomputer. The group of virtual machines serves as computational nodes and additionally as a gateway to access back-end supercomputers when heavy-duty production cycles are required. Both distributed environments have different specifications and resource schedulers (queuing systems) to distribute jobs to their nodes. Considerable effort has been invested in integrating the software modules, workflow management, resource schedulers and database accesses. The interim task distributions on Windrider are discussed in Section 3.

The planned workflow system is presented by giving sample process execution commands in the computing environments (Section 4). To enable the seamless integration of the above computational resources, a job dispatcher is developed to allow workflow to submit jobs in different queuing systems. The achievements are summarised in Section 5 followed by insights into improving the continuous overall hardware capabilities within F4K.

2 Overview of F4K's Hardware Architecture

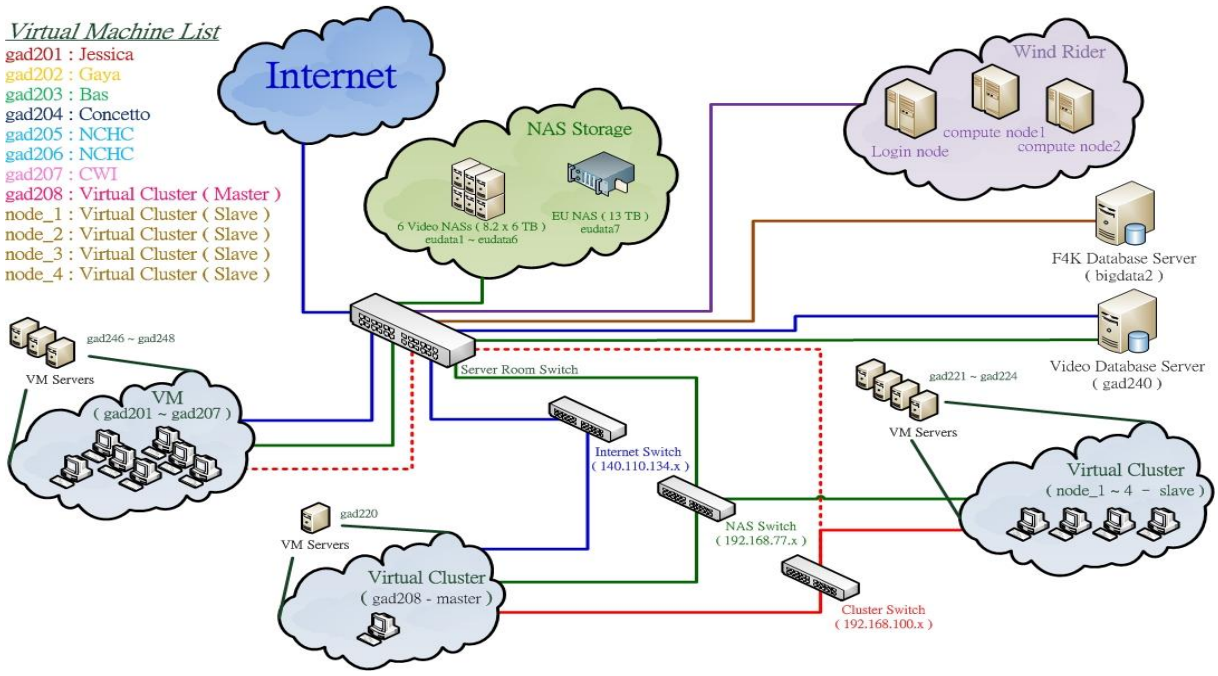


Figure 1: An overview of the hardware architecture available for F4K. The main components include high-performance computing machines, database servers, video storage facilities and web servers.

F4K's parallel capabilities (Figure 1) consist of high performance computing machines that can be VM servers, storage devices for videos, computing nodes and database servers. In the architecture, the virtual machine (VM) group is provided for accessing NARL's computational resources and can also act as computing nodes for process execution. There are several servers to run VMs to support various operation demands and tasks. They are then mapped to VM machines that act as computaional nodes for process execution. There are three networks that inter-connect the computational resources, storage resources and internet for remote ssh/vnc accessing. The video storage and F4K database servers are connected in the network too. Storage is connected in a private network which is not directly accessible from public. Because of the security policy of NARL, the Windrider supercomputer can not be accessed from foreign IP address. Internal virtual machines can be used to access Windrider. Currently, all resources are connected with 1GB network. In the future, the network will be upgraded to a 10GB infrastructure to provide high-speed data transmission between resources.

First the enhancement to the existing capturing and storage methods are outlined, followed by the computing platform and the database servers.

2.1 Enhancement to Video Capturing and Storage Methods

To increase the quality of the videos captured from the undersea observation system of NPP-3 station, the architecture (shown in Figure 2) has been significantly improved to optimise capturing method. NARL added new hardware components for local processing and storing of

these higher quality videos, which will ultimately improve the quality of the video processing results.

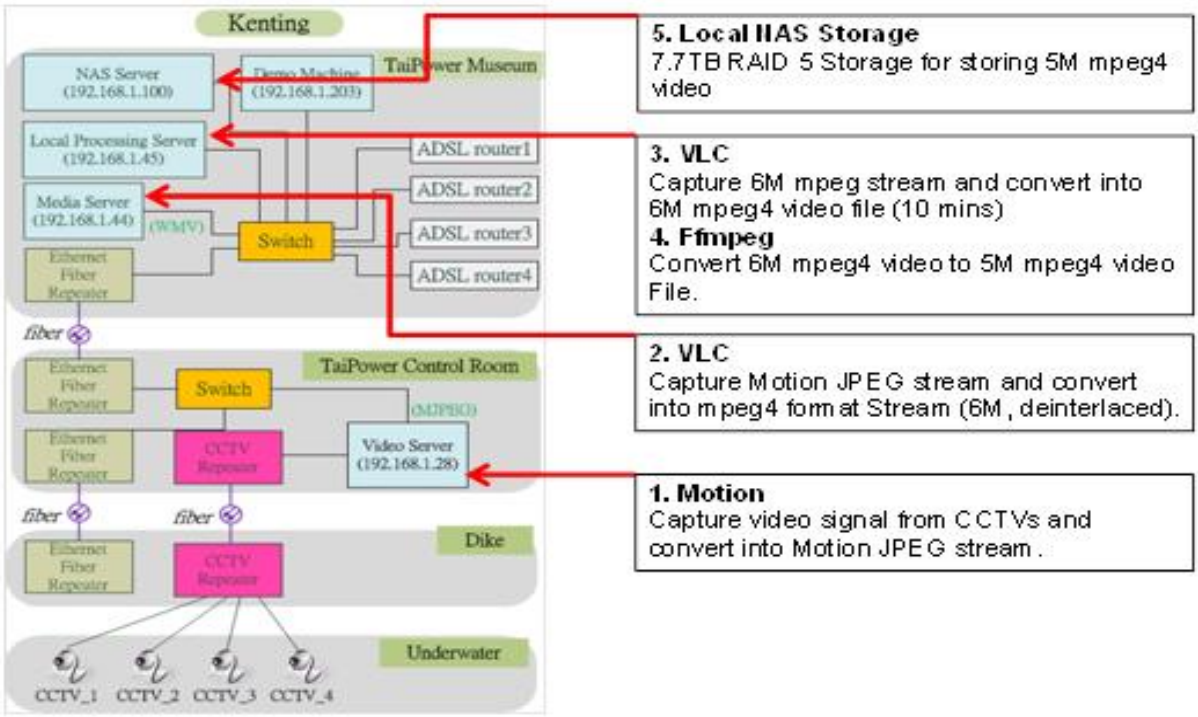


Figure 2: Enhanced architecture for video capturing and storage in NPP-3 site.

The architecture is redesigned to fit the requirement for storing high quality video and also to improve the overall system stability and to reduce the overhead cost of system maintenance. The video server that receives video signals from undersea cameras is relocated to an indoor position, the Tai-power control room. The Tai-power control room provides constant temperature and humidity for long term and steady operation of the video server. This reduces errors in the first stage of video source data production prior to video streaming and storing. Moreover, to enhance quality of the source videos for F4K, the video resolution is increased to 640x480 and the frame rate is increased to 24 (Table 1). Videos are also de-interlaced (linear) to reduce interline twitter in the video. The video capturing method was changed from real-time video streaming to using stream dump which directly saves video data from its streaming source to available storage. The makes video capturing more accurate and controlled.

Due to network bandwidth constraints between NPP-3 site and NARL, it is not feasible to transmit and store real-time high bitrates video in NARL's central storage. Therefore, we implemented a solution by installing a new server (Local Processing Server) and 7.7 TB local storage (NAS Server) in NPP-3 site. The local processing server is used to convert native/raw video stream data to H.264[1] and MPEG4 format in high bitrates (5M). However, converting videos to H.264 format needs six times more processing time than converting to MPEG4. It would not be possible to convert and store each source video within 10 minutes. The MPEG4 format is chosen because the converting time is less than 10 minutes and requires less compute resources. The local processing server is able to capture raw video data from the source and

[1]New video codec standard which can achieve high quality video in relatively low bitrates.

Table 1: New format and method for capturing and storing videos.

	CCTV	CCTV	HD
Format	FLV	MPEG4	FLV
Resolution	640x480	640x480	1280x760
De-Interlace mode	linear	linear	none
FPS	24	24	30
Bit-rate	1M	5M	20M
Capturing method	Stream dump	Stream dump	Stream dump
Site	NPP-3, Lan Yu, Hobihu	NPP-3	NMMBA

store several days of the raw format video (MJPEG) for video analysis. It also compares better with other video formats. The high bitrate videos provide a clear and reliable data source for further video analysis.

The storage is also enhanced, Redundant Array of Independent Disks (RAID) is used in RAID 6 mode in the storage of NPP-3 site to provide higher data protection level. Additionally, Remote Power Management (RPM) is installed to provide feasible system and service recovery when the server crashes. For video storage in NARL, the size of the storage is increased by upgrading it with 3TB sized hard disks. Two video NAS (NAS 1 and 2) are upgraded from size of 8.2 TB to 14 TB (Figure 3). 10 GB network infrastructure is ready for implementation. A 10G Ethernet switch is installed to connect computational and storage resources. 10GB network environment will be ready after all servers and storage resources are installed with 10GB NIC[2] and connected to the 10G Ethernet switch with 10GB Ethernet cable.

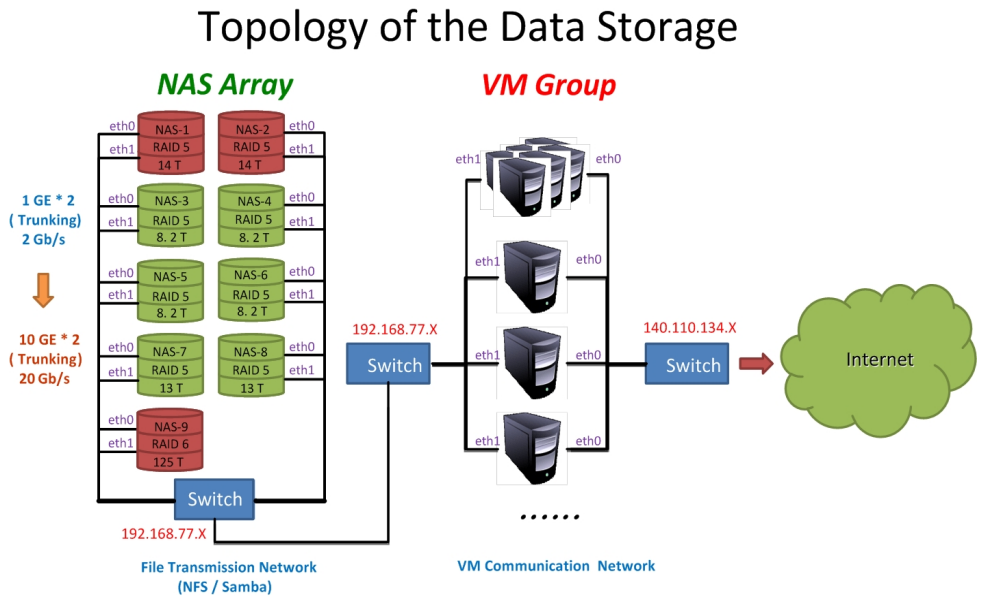


Figure 3: Upgraded and enhanced data storage.

As of November 25th, 2012, there are a total of 573,737 videos recorded in video database and 41,977 high bitrates video collected in storage (Table 2). As more and more video data is

[2]Network Interface Card

captured and updated to the video database, fast querying and retrieval of video information and data is designed for video download stage in workflow. Universally unique identifier (UUID) is implemented in the video database for indexing videos to improve data query efficiency. This is more powerful than using integers as indexes. A five tier hierarchical database acceleration architecture is used for caching video information and reducing database search loading. Because distributed storage is used in video NAS, the directory for storage mounting is redesigned, which allows data accessing across many storages devices in the storage pool.

Table 2: New format and method for capturing and storing videos.

Video Format	Video Bitrate	Site Name	No. of video in storage	No. of video record in database
FLV	200K / 480K / 1M / 2M	All Sites	595,465	573,737
MPEG4	5M	NPP-3	41,977	none

2.2 Provision of Data Storage Facility

Data storage facility is built for storing huge amounts of video data and processed data after video processing. NARL built high-performance Network Attached Storage (NAS) to store video data in the first stage. It allows data sharing in many different types of machines and operation systems and will provide 10G data transmission capability in 10G network after upgraded with 10G NIC. It also takes advantages of less administration overhead but still provides a console interface for fine-tuning and provides many additional ready-to-use services for data sharing such as Network File System (NFS), Samba and HTTP.

In order to enable the use of storage facilities by computational resource servers, NFS is used to provide feasible and reliable storage mounting. NFS is easy to setup on any Linux system and provides efficient large data transmission. Samba will be considered as an alternative and its suitability for storage sharing and accessing will be further evaluated.

Table 3: Capability of storage in different File System and RAID mode.

Disk Size	Num. of Disk	File System	RAID mode	Total Array Size
1.5TB	7	XFS	0	9.6 TB
1.5TB	7	XFS	5	8.2 TB
1.5TB	7	EXT4	5	7.7 TB
1.5TB	7	EXT4	6	6.9 TB
3.0TB	7	EXT4	5	Error
3.0TB	7	XFS	5	Error
3.0TB	7	EXT4	6	13.9 TB
3.0TB	7	XFS	6	13.9 TB
3.0TB	6	EXT4	5	13.9 TB
3.0TB	6	XFS	5	13.9 TB
3.0TB	7	ZFS	5	16.7 TB

RAID storage technology is used to take advantage of data security in the NAS with RAID 5 and RAID 6. Different RAID modes provide different levels of security, performance overhead

and storage capability, as shown in Table 3. RAID 5 is adopted for video NAS to maximise storage capability for storing source video. It also prevents the storage array from being destroyed by a single disk failure. NARL built a high-end NAS storage which installed 58 hard disks to establish a single huge volume of storage. The newly built storage (NAS 9) comes with 125 TB storage capability in RAID 6 mode which provides hardware acceleration to read and write data in RAID. Total up to 198.8TB of data storage size in NARL is built for data storage (Table in Figure 4).

NAS Storage	Size	% Used	% Available	Comment
NAS 1	14 TB	60 %	40 %	Historical video storage
NAS 2	14 TB	44 %	56 %	
NAS 3	8.2 TB	1 %	99 %	
NAS 4	8.2 TB	25 %	75 %	
NAS 5	8.2 TB	96 %	4 %	
NAS 6	8.2 TB	42 %	58 %	
NAS 7	13 TB	8 %	92 %	VM NFS Shared storage
NAS 9	125 TB	0 %	100 %	F4K data storage
Total Storage	**198.8 TB**	**14.6 %**	**85.4 %**	

Figure 4: Data storage and usage in NCHC.

To reduce the potential risk of network attacks such as Denial of Service (DoS) and to maintain network performance, security of data storage the storage is protected by locating it in a private network to avoid direct network attacks from public network. Database accessing is allowed from internal VM group and Windrider. In the future, we will use *curlfs* or *WebDAV* to mount storage in Storage Area Network (SAN) as random-access file system.

2.3 Computing Platform Specifications

In this project, we created two sets of computing platforms to explore a variety of process execution flows. One is a VM group composed of 9 nodes of virtual machines and the other is a multi-core supercomputer, nicknamed Windrider [11], the most powerful computing facility in Taiwan.

Windrider consists of 8 compute clusters and one large memory cluster, in total over 25,600 CPU cores, 73,728 GB memory, and 1,074 TB disk. The system offers an aggregate performance over 177 TFLOPS, is ranked 138 among the 500 most powerful commercially available computer systems known to the world in Nov. 2012. Two compute nodes of Windrider are dedicated to F4K project, with the potential to recruit more nodes if necessary. Table 4 contains the specifications for Windrider.

The VM platform is made up of 9 computing nodes – a 4+1 VM cluster (gad208, Node_1, Node_2, Node_3, Node_4), a master workflow node (gad202), and 3 other computing nodes

Table 4: Windrider Specification Detail (96 cores on 2 nodes).

Machine Name	ALPS Acer AR585 F1 Cluster
Machine Structure	SMP Cluster
Processors	AMD Opteron 6174, 12 cores, 2.2GHz (compute nodes)
	AMD Opteron 6136, 8 cores, 2.4GHz (fat nodes)
Main Memory Size (per node)	128 GB (compute nodes) 256 GB (fat nodes)
Operating System	Novell SuSE Linux Enterprise 11 SP1
Job Scheduler & Queuing	Platform LSF (Load Sharing Facility) 7.06
Directories	/home (209 TB), for user's home /pkg (16 TB), for package /work (162 TB), for working scratch
Parallel Filesystem	Lustre
Installed Software	ABINIT, Amber, CASINO, CHARMM, Chemsoft, CPMD, DL_POLY, GAMESS, Gaussian, GROMACS, Molpro, NAMD, NWChem, octopus, OpenMX, Quantum ESPRESSO, siesta, VASP, WIEN2k

(gad201, gad205 and gad206). Their specifications are contained in Table 5. See Figure 8 for a pictorial overview of the VM group. The VMs are mapped to physical VM servers, shown in Table 6 and whose detailed specifications are contained in Table 7. The workflow VM is set up with Gridengine [4] resource scheduler with all the VM nodes as computing nodes, not all of which are active yet.

Table 5: VM nodes assigned to different F4K members. Gad203 and Gad204 are not used as compute nodes in the VM group, while Gad201 and Gad205 are still inactive. All the other 7 nodes are operational with 50 cores available for workflow processing.

VM Name	Number of Processors	Number of cores for Processing	Memory Size	HD size	Owner
Gad201	8	None yet	16GB	64GB	Jessica
Gad202	16	12	32GB	64GB	Gaya
Gad203	8	0	16GB	64GB	Bas
Gad204	8	0	16GB	64GB	Concetto
Gad205	8	None yet	16GB	64GB	NARL
Gad206	8	8	16GB	64GB	NARL
Gad207	8	0	16GB	64GB	CWI
Gad208	6	6	6GB	64GB	NARL
Node_1	6	6	6GB	64GB	NARL
Node_2	6	6	6GB	64GB	NARL
Node_3	6	6	6GB	64GB	NARL
Node_4	6	6	6GB	64GB	NARL

Table 6: VM Servers Brief Specification and Mapping to VM Nodes.

VM Server Name	Number of Processors	Memory Size	VM Node
Gad246	48	128 GB	Gad201 Gad203 Gad204 Gad207
Gad247	16	64 GB	Gad202
Gad248	16	64 GB	Gad205 Gad206
Gad220	8	8 GB	Gad208
Gad221	8	8 GB	Node_1
Gad222	8	8 GB	Node_2
Gad223	8	8 GB	Node_3
Gad224	8	8 GB	Node_4

Table 7: VM Server Detail Specification.

Server Name	Gad246	Gad247 - Gad248	Gad220 - Gad224
Manufacturer	Super micro	ASUS	ASUS
Server Model	B8812F48W8HR	ESC2000	BM6660
Num. of CPU Socket	4	2	1
CPU Model	AMD Opteron 6176 SE 2.3 GHz	Intel(R) Xeon E5620 @ 2.4GHz	Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
Memory Size	128 GB installed	64 GB installed	8 GB installed
Chipset Model	AMD SR5690 + SP5100	Intel 5520 IOH	NVIDIA NF200
HD Size	2TB + 5.4TB installed	823 GB	500 GB
Network	1GB Ethernet x2	1GB Ethernet x3	1GB Ethernet x3
OS	Ubuntu 10.04	Ubuntu 10.04	Ubuntu 10.04
Kernel	2.6.32-21-generic	2.6.32-21-generic	2.6.32-21-generic

Platform specifications of the VM cluster and Windrider are summarised as follows:

- VM cluster: 4+1 nodes, each node with Intel i7-2600@3.4GHz CPU, 6 CPU cores, 8GB RAM, gigabit Ethernet link to massive storage. Resources are managed by Gridengine. Both interactive and batch jobs are acceptable.
- Windrider: 2 nodes dedicated, each with AMD Opteron 6174 2.2 GHz CPU, 48 CPU cores, 128GB RAM. Resources are managed by Platform LSF. Batch jobs only.

Given that these two platforms have different resource schedulers, an interface to bridge the two schedulers is required so users can compose and submit jobs based on computation requirements without knowing the details of the schedulers. A middleware, job dispatcher, was developed to direct the workflow engine to submit jobs to the proper platform and tracks status. Details of job dispatcher is contained in Section 4.2. Figure 5 shows a conceptual system accessing diagram of the main components – raw data, storage, database, applications, job dispatcher and the computing platforms (Windrider and VM group).

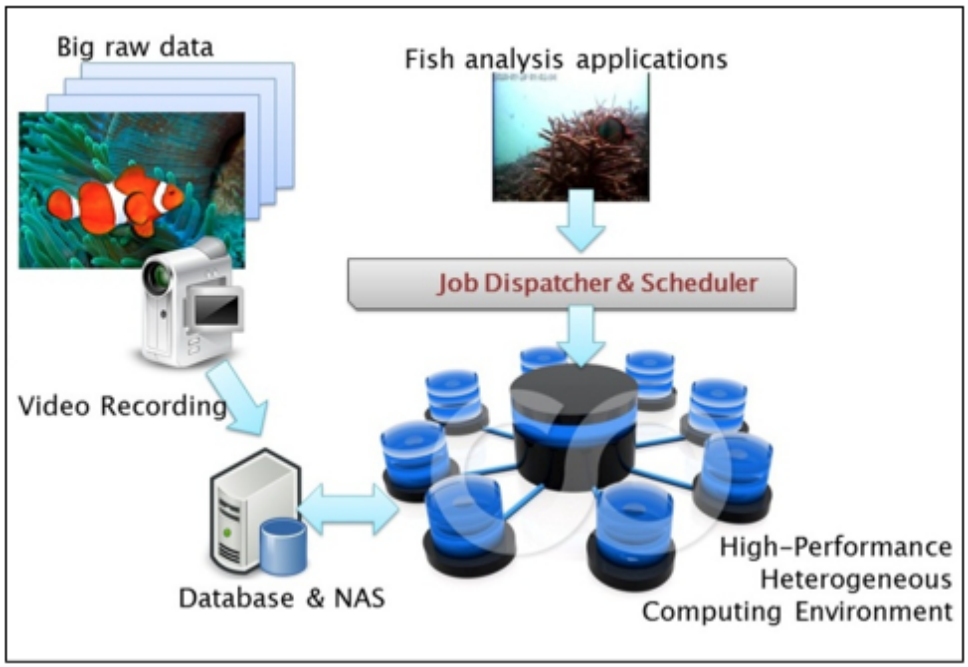


Figure 5: A conceptual system accessing diagram.

2.4 Database Servers

We have provided a live database and several test databases for F4K use. The main one is a MySQL Database, referred to as "bigdata2". In this database, all the main processing results are stored for analysis. Two test databases, called "bigdata1" and "gleon" were provided mainly for workflow testing and for fallback should anything happen to the main database. These two databases are clones of "bigdata2". All databases are hosted at NARL.

In the video processing tasks, the fish detection component extracts the contour from the video data and this data is stored efficiently in the relational database. In the distributed computing environment, massive amounts of detection tasks are running at same time and these processes need to communicate with the database instantly to retrieve parameters and store results. Rapid accesses to the database at same time can create heavy loading on the database server which eventually becomes a bottleneck in the overall workflow. To overcome this issue we adopted a load balancing mechanism which uses a two-node master-slave replication cluster and redirected read-write queries to the master and slave separately. Mysql-proxy [12] is used as the redirecting gatekeeper. With this replication cluster + proxy load balancing mechanism we can easily scale out process capability of database server by adding more slaves when necessary. Schematic description of database server is given by Figure 6.

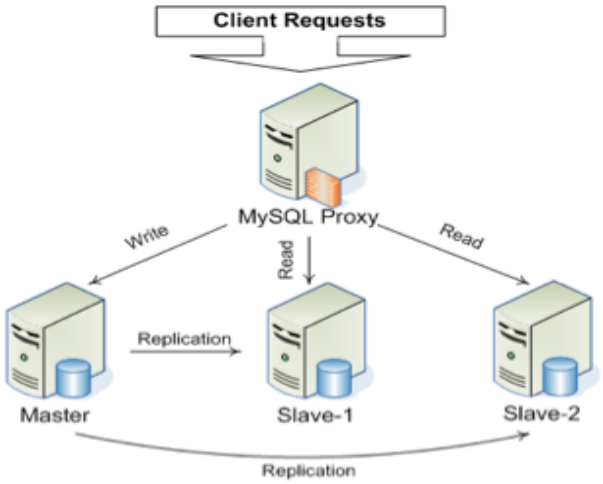


Figure 6: Requests from clients (image processing tasks) are redirected to master and slaves by proxy server to achieve the goal of load balancing.

3 Overview of Interim Task Farming

In Windrider, the bulk jobs to process the video data are submitted, controlled, and monitored by the Load Sharing Facility (LSF) queue system [3]. It is simple and easy to use within HPC environment, using similar scheduling and control concepts as those for Gridengine. Basic automatic process control can be done by warping the LSF control command. Also, it is anticipated to be relatively easy to relay from or re-direct to workflow and Gridengine instances. Here we present some task distribution that has been done on Windrider for fish detection and tracking and fish species recognition modules.

3.1 Task Farming for Fish Detection and Tracking

The fish detection and tracking component is a standalone executable that should be applied on all the videos as it would allow for population-related analysis and also for fish recognition component to execute. It primarily updates the 'fish' and 'fish_detection' tables.

Currently, the pre-production run of VIP component were accomplished via 1) queue "fpl" with 96 cores on Windrider; and 2) queue "monos01" with 1000 cores. This subsection will discuss the task farming on the "fpl" queue while subsection 3.3 will discuss the results of execution on the "monos01" queue. The distribution of jobs in Windrider was achieved by using simulation scripts. The simulations spawned jobs based on several criteria:

- Run the jobs based on a contiguous set of dates for all cameras in all locations and all resolutions.
- Run the jobs based on a contiguous set of dates for a specific location and all resolutions.
- Run the jobs based on a contiguous set of dates for a specific location and a specific resolution (fixed, maximum or minimum resolution).
- Run the combinations above using a specific software component (detection algorithm).

- Run the jobs based on a random selection of videos.

To execute the fish detection and tracking component, the following arguments are needed:

- Video: The fish detection and tracking run on a single video, so the video_id is necessary to determine which video will be used for the fish detection and tracking module.
- Detection algorithm: Different detection algorithms have been developed and improved to be used for different video types. The four detection algorithms are 'Vibe2', Gaussian mixture model, 'CB' and Intrinsic model. These values, with other settings such as the tracking algorithm,
- Database parameter: Every component has to communicate to the database, multiple parameters such as the hostname, username, password and database name are encoded in a configuration file, that is passed as a parameter.
- Video Storage: The workflow can indicate which directory to use in case of video file on the local computer so that a video file does not have to be downloaded twice to the computer, but we will use the copy first used by the fish detection component instead.
- Log and Results: The executable produces log files that will help with error diagnosis. Results in the form of output XML file and detection summary images will also stored in the directory (specified as an argument or by default otherwise).

3.2 Task Farming for Fish Species Recognition

The fish recognition component is a standalone executable that is dependent on the fish detection and tracking component. To perform fish recognition, the location and segmentation of the fish in the video is required, which is saved in the tables 'fish' and 'fish_detection'. The executable also uses information from the video and cameras table to obtain the frame rate and resolution of the video recordings. The fish recognition results are saved in the table 'fish_species'.

To execute the fish recognition component, the following arguments are needed:

- MCR enviroment: because the executable is developed in matlab, an matlab environment path needs to be defined in order to link to the underlining function that matlab offers a developer.
- Model: Our recognition component works using a model which is save in a .mat file. This model contains data which is necessary for the underlining method and contains the features that are used to recognise the fish.
- Database parameter: Every component has to communicate to the database, multiple parameters like the sort of database (MySQL or MonetDB), hostname, username, password, dbname are necessary to establish a database connection
- Video: The fish recognition run on a single video, so the video_id is necessary to determine which video will be used for the fish recognition
- Detection component: The fish recognition is dependent on the data from the fish detection component, in order to find this data we need to know which detection component run on the data.

- Segmentation: Different Segmentation methods are developed to improve the fish recognition methods. The default method is the segmentation of the detection component (option 1). However, this can be improved by grabcut (option 2) or chan-vese (option 3) segmentation method, but these methods are computational more expensive.
- Video Storage: The workflow can indicate which directory to use in case of video file on the local computer so that a video file does not have to be downloaded twice to the computer, but we will use the copy first used by the fish detection component instead.
- Windrider: Because certain libraries are not installed on the Windrider machine, reading the videos is more difficult, which this arguments fixes.
- Debug options: For debugging purposes, we have an option where the results are not saved to the database.

At the moment, the fish recognition component uses a single CPU because most of the code is sequencial anyway. The nice thing is that we can easily execute the fish recognition distribution on the different videos. Because the fish recognition is dependent on the fish detection, we developed a script that runs on all the videos that are already processed by the detection. Psuedo code of the script is given in Algorithm 1 below:

```
Main thread:
  Define vm_queue => 5 cores, windrider_queue => 50 cores
  Get all process_videos by detection_component
  For each process_video
    If not processed by recognition_component
      If queues(vm_queue,windrider_queue) are not full
        Add process to not full queue

Queue thread:
  Add to process_videos table
  Start running
  If(VM)
    fork(recognition_component.exe)
  If(WINDRIDER)
    fork(ssh windrider bsub recognition_component.exe)
  While no log file
    Pause(1 minute)
    Break after 24 hours
    If logfile
      Add 'finished' to process_videos table
```

Algorithm 1. Task distribution for fish species recognition component on Windrider.

This script executes on a main thread that has two queues, called 'vm_queue' and 'windrider_queue' in Algorithm 1, where jobs can be added to. We obtain a list of processed videos 'process_videos' by the detection component. For all these videos, we check if they are not already processed by us and we put them on an empty queue. (if the queues are full, we wait until one becomes empty). In the queue, we added the processing information to the

database, start the processing by executing the software on the VM machine or using bsub on the Windrider server. If the executable is finished, we can detected this because it will write a log file. We report to the database that the executable is finished and remove the process from the queue.

3.3 Testing on a 1000-core Facility

The first attempt at large-scale execution was done using the fish detection and tracking component on the 'monos01' queue, a 1000-core facility. The aim was to find bottlenecks in the system prior to conducting longer production runs. It also allowed for more videos to be processed and hence more data to be produced that can be aggregated and presented to the user.

This test was conducted in a 32-hour period on hourly videos in 2011. The videos were a mixture of low resolution (320x240) and high resolution (640x480) ones. Each job (component on a video) was assigned to 5 cores and 200 jobs were spawned simultaneously. Out of 9972 jobs (videos) processed, 99.2% (9973) were completed successfully, with an average of 5.2 completions per minute. The average processing time for the low resolution videos was 10 minutes and 34 seconds and for the high resolution videos was 51 minutes and 45 seconds. The test was overall successful but the database could not handle the load from all the processes. Hence we concluded that the database is the bottleneck.

Concurrent accesses to the database seems to be the main issue. It is anticipated that the database will not scale for 100-GB sized tables. A temporary solution that we devised was to write the results to XML files first, and import the results into the database later on.

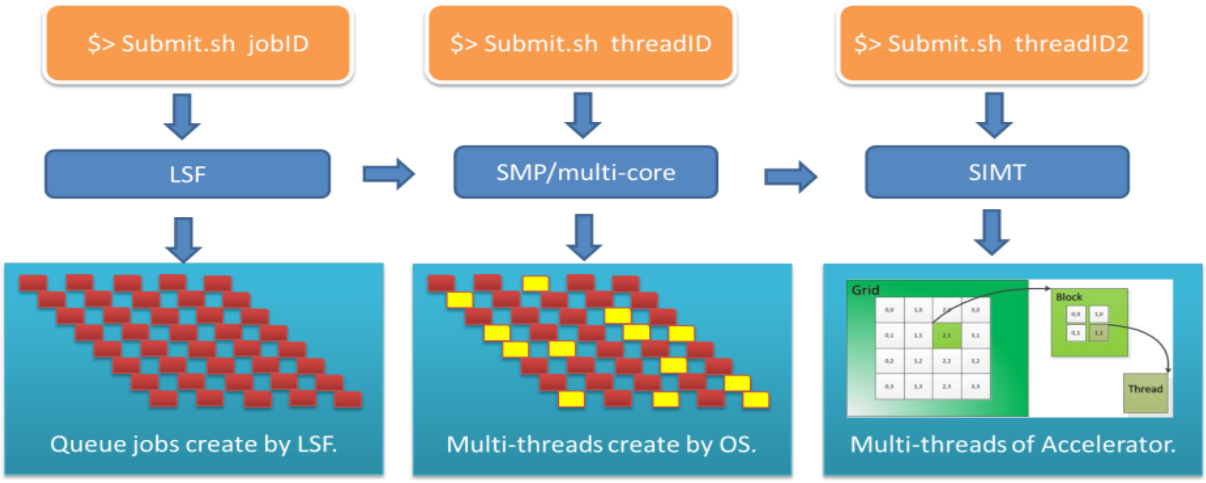


Figure 7: Present computational framework and the future developing new architecture for many core/thread accelerators. The SIMT accelerators such as GPU and Intel Many Integrated Core Architecture are more suitable for F4K computing tasks.

From a hardware architectural perspective, the facility with 1000-core cluster is still not producing optimum benefit for big data processing tasks. This is due its SIMD (Single Instruction Multiple Data) architecture, high computing intensity is the bottleneck of the bulk video/image processing problem.

The aim of high performance computing is not only to develop an effective computing method or algorithm but also to seek an efficient and low power-demanding processor for gen-

eral computation. In the case of the F4K project, most of tasks are due to SIMD problems, where repetition of high intensity tasks using the same program but various and input data causes errors. Therefore, in the future, the SIMT (single-instruction multiple-thread) architecture will be considered.

Next the planned workflow and job dispatcher facilities will be described to describe the utilisation of the hardware and software capabilities for process execution in F4K.

4 Overview of the Planned Workflow System

The workflow system acts as the bridge between the front-end of the F4K system which deals with user requests and the back-end which consists of the video analysis modules on the HPC facilities. The workflow manager composes VIP workflow instances (or jobs) based on incoming high level queries, distributes the jobs onto the HPC facilities and monitors the execution of these jobs. The workflow composition mechanism is described in Deliverable 3.2 [7].

Once composed, the VIP workflow instances are scheduled for execution using a resource scheduler. Each instance sent for execution is known as a job. Jobs are monitored so to keep track of their progress, and appropriate control mechanisms can be deployed to detect any errors that may occur at various levels.

4.1 Computing Environment

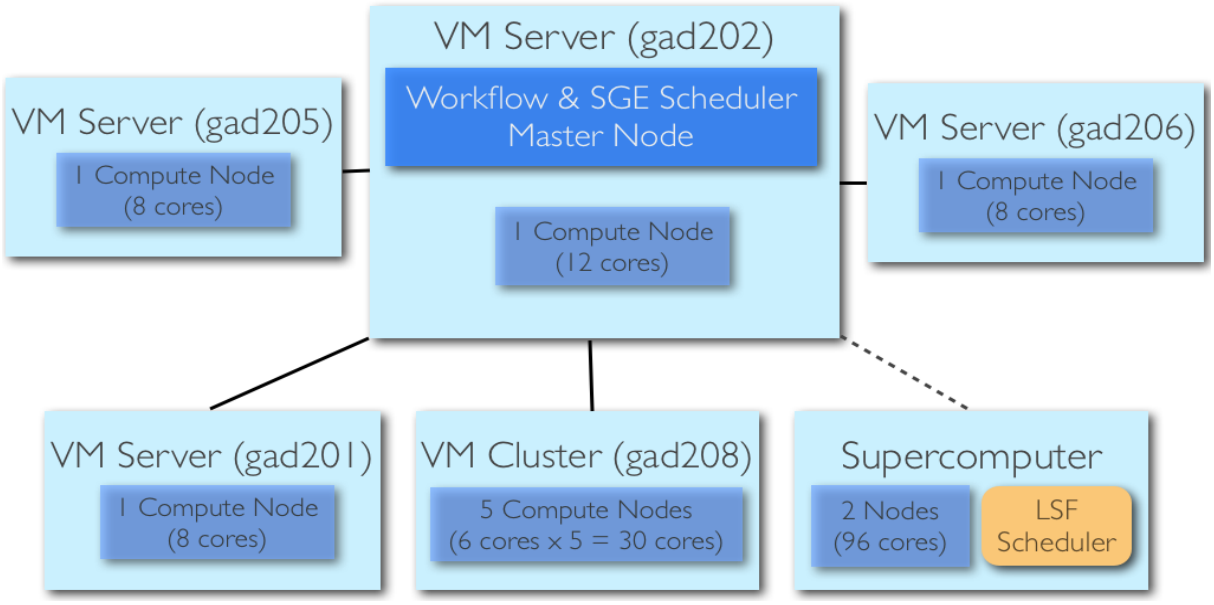


Figure 8: The HPC computing environment for F4K's process execution.

As mentioned previously we have two resource schedulers installed on our HPC platforms. The open source Gridengine (SGE)[4] is installed on the VM platform while the commercial load sharing facility (LSF)[3] is installed on Windrider. The workflow manager is installed on the VM master node and will be the starting point for process scheduling and execution.

Figure 8 shows the computing environment available for execution. The SGE scheduler is shaded in blue and LSF in orange. While all the VM machines are connected to the master node (that contains the workflow and scheduler), connection to Windrider is established via the SSH protocol. This work is still in progress.

4.2 Job Dispatcher: Interface between Workflow and Resource Schedulers

The critical issue we are encountering is how to deal with job queries with big data processing in an efficient way on a heterogeneous computing environment. Therefore, we proceed to cope with the following key issues:

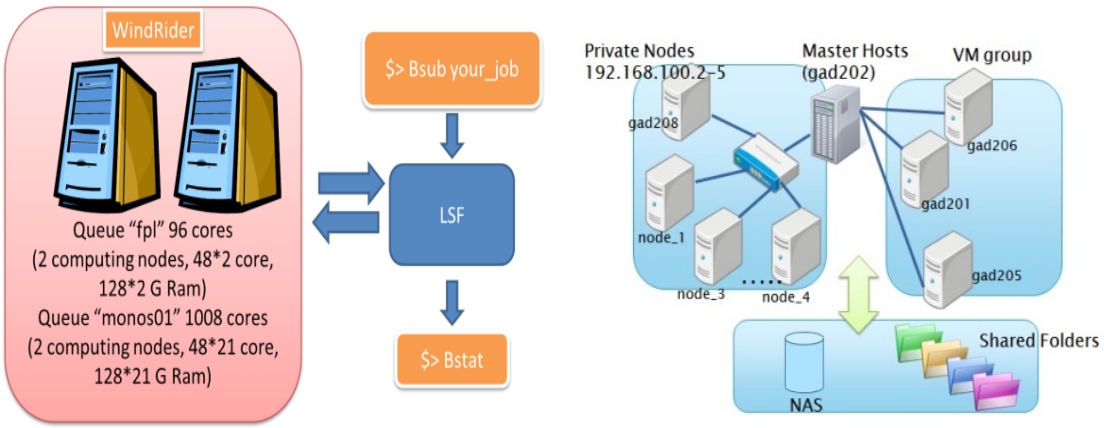
1. Resource Selection and Allocation
   Resource matching according to job types, dependencies, resource requirements, system status, etc.

2. Big Data Computing
   Efficient data access with high-performance on large network storage systems.

3. Heterogeneous Computing Environment
   To increase the efficiency and performance of heterogeneous computing resources for different kind of jobs and to integrate with the Workflow team.

4. Performance Evaluation
   Delay estimation and system performance estimation.

Currently, we are developing the components of a job dispatcher on top of two queuing systems, Gridengine on the VM group and LSF on Windrider. A uniform process execution interface is required regarding both different queuing systems and resource types. Figure 9 shows the different access models of Windrider and VM group.



(a) Windrider is managed by LSF scheduler. (b) VM cluster is managed by Gridengine.

Figure 9: Heterogeneous compute platforms and their respective resource schedulers.

In order to set up Gridengine with the nodes in the VM group, all the following procedures and checks were performed:

1. Does the VM group (including VM server and VM) use the public IP or NAT ? If using NAT, there is only one direction of binding and listening socket is allowed.
2. Check the firewall rules of the iptables in all of the VM group machines.
3. Check the MAC / IP pair mapping in accordance with the new filter policy in the switch.
4. Check the necessary TCP port table which the Gridengine environment is using and the mapping rule of the Linux OS.

Shared directories were also created to connect data between the nodes, and all the relevant software and libraries were installed in all the nodes. Table 8 shows the various directories created and mounted onto the VM nodes for sharing data.

Table 8: Shared Directories mounted in Gridengine nodes.

Directory	Mount Source	Size	Available	Usage
/home/gaya/F4K_Share /home/gaya/work /home/nchc/work	NAS	13TB	12TB	Data Sharing Log files and downloaded videos
/home/nchc/eudata1	NAS	14TB	5.5TB	Video storage
/home/nchc/eudata5	NAS	8.2TB	0	Video storage
/home/gaya	Gad202	48GB	29GB	Data exchange

To enable the seamless integration of heterogeneous compute platforms, a middleware component, named job dispatcher, was developed. Job dispatcher allows workflow components to access all the computing resources using a uniform interface for job submission, job control, and job monitoring. Computing resources can be dynamically allocated according to the demands of the workflow. Integrated access model of job dispatcher is shown in Figure 10.

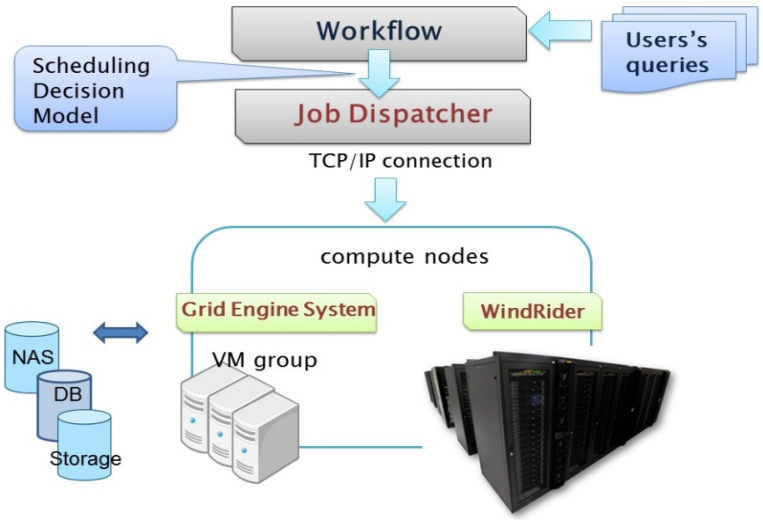


Figure 10: Interactions between workflow and two schedulers are uniformed by job dispatcher.

The job dispatcher makes use of the Distributed Resource Management Application API (DRMAA) [6] to interface workflow jobs to Gridengine. The facilities available to the workflow include a job submission utility that takes in VIP workflow instances along with Gridengine options, job control (suspend, resume, terminate) and job status checking. Furthermore, the dependence of parallel jobs can be manipulated as well.

At the moment we have completed a dispatcher component which can accept job queries from the workflow engine and route the queries to the appropriate computing resources according to the requirements of user or workflow daemon. A uniform interface between two different kinds of computing resources is important because of system integrity and performance. The job dispatcher can maximise the system's performance as well as minimize the developing effort of cooperating research teams. Furthermore, we are going to collaborate with the Workflow team to develop a performance evaluation model in the near future for the workflow engine to do resource selection effectively.

4.3 Scheduling for Execution using SGE on VM Platform

The SGE queuing system is shown in Figure 11. It consists of a master daemon and an execution daemon.

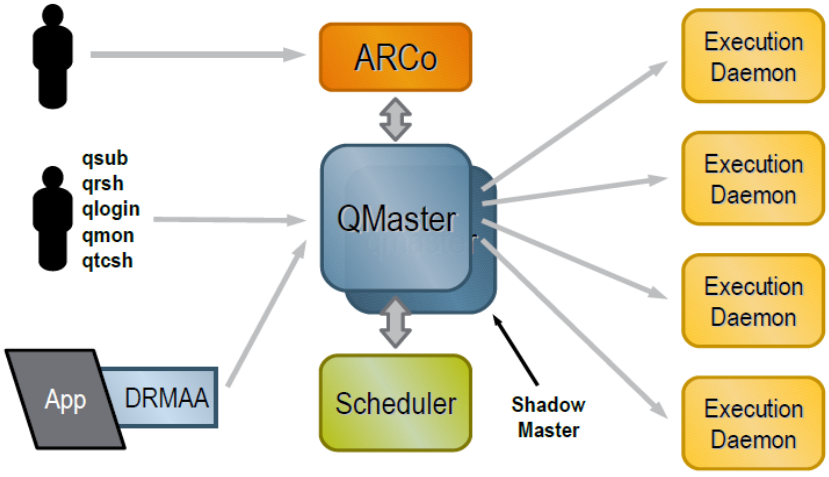


Figure 11: Queuing system of Gridengine.

The workflow invokes a job using the dispatcher via the `drmaa_job_submit` utility. The command invocation is as follows:

```
~/dispatcher/drmaa_job_submit ''<VIP_command>'' ''<SGE_options>''
```

The `VIP_command` clause refers to the low level command line invocation for a VIP job, which typically requires a video and a set of parameters that need to be selected by the workflow. `SGE_options` are additional parameters that can be used to control and manipulate various scheduling and execution options. Among some options that are planned to be manipulated are:

- Specifying the job priority using the "-p" prefix.
- Specifying a queue name for a job using the "-q" prefix.

- Specifying dependencies between jobs using the "-hold_jid" prefix.
- Specifying the time that the job is eligible for execution using the "-a" prefix.

4.4 Scheduling for Execution using LSF on Windrider

For submitting jobs in Windrider, there is the LSF queue system to do the resources and queue management. Figure 12 shows the LSF job control and states. User only can access to the Windrider by SSH tunnel and with SCP for data exchange between client and host. Additionally, the window-based can relay the X client window on the screen with Xming(windows)/Xhost(Linux) or others. Figure shows the relay window form host. Figure 13 shows the fish video processed window form Windrider host. It is easy to show the prompt window of inactive jobs from login host for real time visual checking. All libraries are set to `/work/f4k/LIB` location and can be easy to access and re-modify for new or third party lib and source code. These libraries include third party and comment OpenCV , libavcodec, x264, FFmpeg, XVID, Matlab, etc were already setup in `/work/u00F4K00/` the exclusive directory.

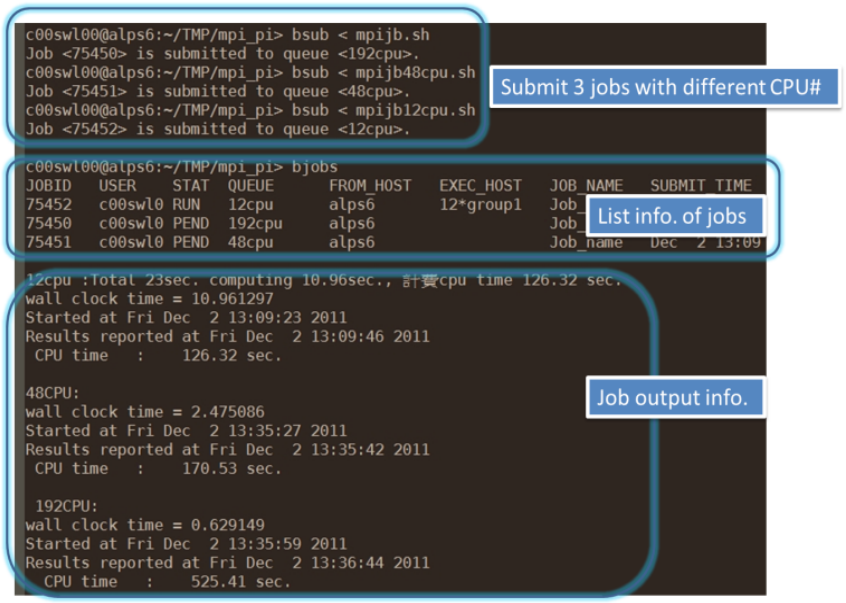


Figure 12: LSF resource and job control display.

For example, submitting jobs that consist of executing fish recognition modules, the following sample queuing commands could be used for 'Interactive' and 'Silent' modes:
Interactive:

```
bsub -n 1 -I -q fpl ./script_fish_recog_nosave dfa7eb84dd44429a23eb05e
a1c14b924#201104020750 26 1
```

Silent:

```
bsub -n 1 -q fpl ./script_fish_recog_nosave dfa7eb84dd44429a23eb05ea1c
14b924#201104020750 26 1
```


Figure 13: Fish processing video is relayed to client from Windrider host.

5 Discussions and Conclusions

In this work, the workflow computational platform is designed and built for F4K, which allows video and image processing (VIP) components that constitute the tasks of the workflow to be composed and executed efficiently over a Tera-scale video dataset. The main accomplishments to date are the following:

- More sophisticated video capture and storage methods for better data provision and processing by other F4K components. This was done via an improved observation system for capturing undersea video, and enhanced video quality for VIP by adopting new video format and using stream dump as capturing method.
- Built 200 TB data storage facility.
- A total of 162 cores available for workflow computational platform service, including 66 cores server which provides virtual machine running platform (50 cores on 7 active nodes and another 16 cores on 2 disabled nodes anticipated soon), and 96 cores from Windrider supercomputer used as dedicated compute service.
- Two main video processing modules, fish detection and tracking and fish species recognition have been installed and running on both VM and Windrider platforms.
- Customised VM Portal, web-based interface to access all computational resources, is developed. A process execution platform based on virtual machines is developed to assist F4K developers to migrate and integrate their codes from local development sites to the centralized production site.
- Workflow can access computing resources by the uniform interfaces for job submission, job control, and job monitoring provided by job dispatcher.
- Computing resources can be dynamically allocated according to the demand of workflow

- Fish detection and tracking component using 200KB videos distributed over 1000 CPU cores was partly successfully run on ALPS. (2012/11/13)

Effort is underway to continue the provision of capture, storage and high performance computing facilities for F4K. The planned work in progress include:

- Improve network infrastructure to 10GB.
- Investigate SIMT architecture for multi-core machines.
- Develop uniform components for process execution on Gridengine nodes (VM) and Windrider.
- Performance based workflow execution and monitoring via the provision of metrics such as node status at a point in time and network traffic between two nodes.

References

[1] H.M. Chou, Y.H. Shiau, S.W. Lo, S.I. Lin, F.P. Lin, C.C. Kuo, and C.L. Lai. A Real-time Ecological Observation Video Streaming System Based on Grid Architecture. In *HPC Asia*, 2009.

[2] T.L. Chung, W.Y. Chang, W.F. Tsai, F.P. Lin, E. Strandell, L.C. Ku, J.G. Lee, J.Y. Chang, T.H. Lee, J.H. Wu, et al. Cyberinfrastructure for Flood Mitigation in Taiwan. *Water management*, 163(1):3–11, 2010.

[3] IBM Platform Computing. *Load Sharing Facility (LSF)*. 2012. `http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf/index.html`. Last accessed: Dec 4th, 2012.

[4] Oracle Corporation. *Open Grid Scheduler*. 2012. `http://gridscheduler.sourceforge.net/`. Last accessed: Dec 4th, 2012.

[5] W. W. Eckerson. Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications. *Open Information Systems*, 10(1), 1995.

[6] Open Grid Forum. *Distributed Resource Management Application API (DRMAA)*. 2012. `http://www.drmaa.org/`. Last accessed: Dec 4th, 2012.

[7] G.Nadarajan and Y.-H.-Chen-Burger. Process Planning and Composition. Deliverable 3.2, Fish4Knowledge Consortium, June 2012.

[8] N.C. Lin, T.Y. Fan, F.P. Lin, K.T. Shao, and T.H. Sheen. Monitoring of of Coral Reefs at the Intake Inlet and Outlet Bay of the 3rd Nuclear Power Plant in Southern Taiwan. In *Annual Meeting of The Fisheries Society of Taiwan*, pages 19–20, 2009.

[9] S.W. Lo, C.W. Hsieh, and F.P. Lin. Comparison of Monte Carlo Methods for the performance of CBE, CPU and GPU. In *HPC Asia*, 2009.

[10] G. Nadarajan. *Semantics and Planning Based Workflow Composition for Video Processing*. PhD thesis, University of Edinburgh, 2010.

[11] Taiwan National Center for High Performance Computing (NCHC), NARL. *Windrider Supercomputer*. 2011. http://www.nchc.org.tw/tw/services/supercomputing/supercomputing_1/ar585f1.php. Last accessed: Dec 4th, 2012.

[12] MySQL Open Source Software. *MySQL Proxy*. 2012. http://dev.mysql.com/downloads/mysql-proxy/. Last accessed: Dec 4th, 2012.

[13] W.F. Tsai, W. Huang, F.P. Lin, B. Hung, Y.T. Wang, S. Shiau, S.C. Lin, C.H. Hsieh, H.E. Yu, Y.L. Pan, et al. The Human-centered Cyberinfrastructure for Scientific and Engineering Grid Applications. *Journal of the Chinese institute of engineers*, 31(7):1127–1139, 2008.

[14] J.H. Wu, T.L. Chung, F.P. Lin, and W.F. Tsai. A Scalable Middleware for Multimedia Streaming. In *HPC Asia*, 2009.

