Green Data Center Computing: A Demonstration Project
NYSERDA Agreement # 22899
Period of This Report: July 16 – Oct 15 2012

Introduction

The objective of this project is to demonstrate the feasibility of deploying a network of Performance Optimized Datacenters (PODs), geographically distributed to exploit the availability of renewable energy for their operation. Such A distributed system with PODs co-located with the power source has the potential to significantly enhance the energy efficiency, reliability, security, and overall performance of data centers. The system is capable of optimizing renewable power use for computing, by intelligently redistributing computational load depending on the availability of renewable energy, thereby minimizing the losses associated with power transmission. As the availability of wind power through New York State and other states grow, this concept would address transmission & distribution (T&D) constraints without expensive utility upgrades. By keeping the infrastructure and T&D costs low and by utilizing the wind power that is currently stranded (i.e. not-delivered to the grid due to the T&D constraints), implementation of this technology is expected yield significant energy and cost savings.

Short Summary

This report presents details of the project work performed to date:

- **Task 1: System Design and Integration.** This task includes POD setup, , Operation Management Unit design and assembly, and Energy Management Unit and Power Supply development. This task is currently in progress. Significant progress was made during this reporting period and includes the completion of the design of the full system and acquisition of several components for the in-house laboratory. The renewable energy emulator architecture was modified to be consistent with the changes and updates made in the complete POD architecture.

- **Task 2: Characterization of the subsystems and the assembly to determine the correlation between Input Power (to POD) to Wind Variability.** This task is currently under investigation. An algorithm, implemented in a numerical environment, was developed to model the power commitment scheduling. The particular scenario described in this report is analogous to the transfer of workload between two PODs to minimize the overall system operation cost. Accordingly, the algorithm was used to demonstrate power usage and net balance between demand (workload) and power availability, either from renewables or from grid.

- **Task 3: Development of a system level model using experimental data for design optimization**, Work on this task has just started. Initial experimental data from system components instrumental to this task are being acquired and investigated.

- **Task 4: Migration of Datacenter Workloads into and out of a POD in a controlled environment.** This task aims to characterize the type of workloads suitable for the POD architecture. A detailed review of existing virtual machine migration technologies and other tools has been completed. The strategies for managing downtime at one POD datacenter and/or shifting work between PODs to take advantage of available renewable power have also been reviewed. A number of other tools and strategies for making POD hosting a viable alternative for a larger range of workloads are under development. Some workloads are a natural fit (e.g. hosting of static content) for POD datacenter operation, while others are not (e.g. write-intensive workloads with high availability and coherency requirements).

The list of hardware acquired for the laboratory demonstration is provided in the next section. The expenditures incurred during this period and a schedule showing the status of progress is also included.

a. Progress of Project to-Date

TASK 1 - Progress to Date

I. Emulator Block

Development of the emulator: This is a tool that generates a power profile typical of a specified wind turbine generator from a given wind profile. By acquiring real wind speed measurements, an identical power profile from a real generator (inclusive of aerodynamic, mechanical, and electrical losses) can be emulated.

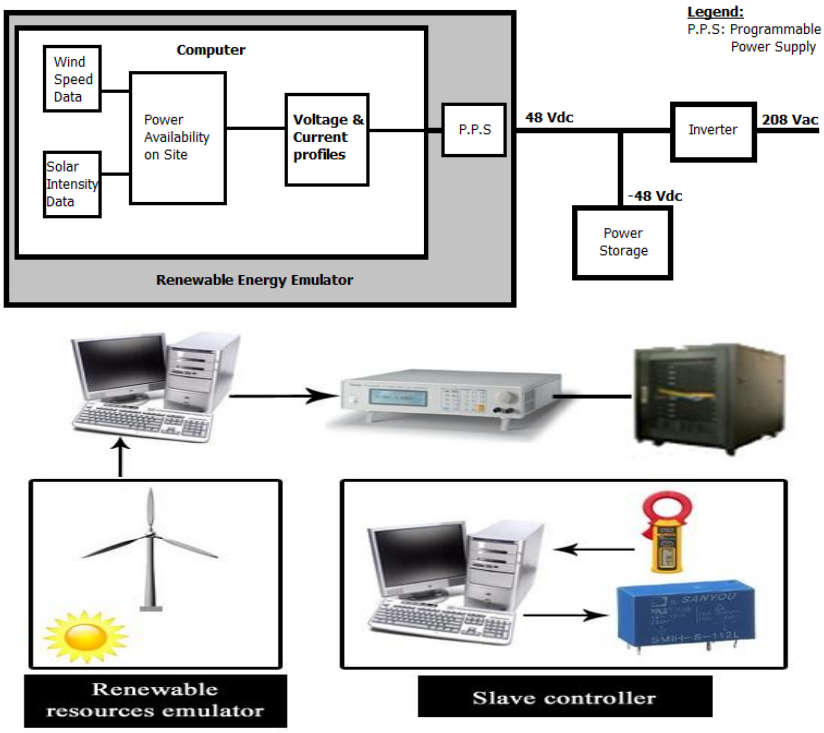


Figure 1 – Renewable Resources Emulator

The renewable resource emulator imports and pre-processes the data, compute the values of voltage and current that a wind turbine can produce from the wind speed measurements, and send voltage/current profile to the programmable DC power supply. The programmable DC power supply will in turn feed power to the POD. A slave controller takes charge of the plant, turbine, and POD. It measures the electric transients and the steady state conditions, all recorded internally in the hardware, which that can be used for control algorithm development. The power can be determined from the DC voltage and current which are measured in the laboratory setup. Fluctuations of current and voltage are also taken in consideration based on the specs of specific wind turbine. A power drop will have more influence on current than voltage. For instance, if the wind speed decreases, the wind turbine will not be able to run several servers, due to a current drop. In those instances, a storage bank is required to support the POD operations.

II. Distributed Control System (DCS)

The first control strategy investigated was to be implemented by controlling different sites from a centralized control system acquiring electric and weather measurements, and then proceeds to forecast for weather availability. Depending on the weather forecast, the controller decides to shift workload to the desired location. However, this procedure keeps the controller continuously to be on with the disadvantage that it might not be able to respond to an external alert or interruption. To address this issue, an alternative solution was proposed. The proposed architecture is a D.C.S that divides the tasks in order to supplement the central controller and rapidly provide decisions. Each individual location has its own controller, denoted "Slave Controller", that controls and monitors the (solar or wind) site where it is located. Then, these slave controllers communicate with a central controller, denoted "Master Controller," that helps taking decisions. Once the master controller determines the available power distribution, it supplies these data to the workload manager that decides the best approach to shift data from one location to another.

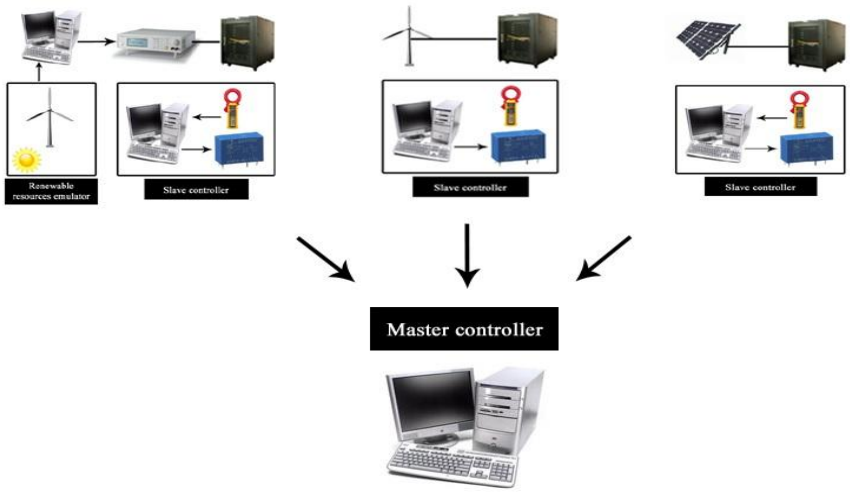


Figure 2 – DCS Architecture

III. Electric panels:

Schematics for the DC bus and AC bus have been made to ensure a secure power transmission from power generation to the POD. They take in consideration the necessary protections to guard the servers from voltage shortcuts and current overshoots. In addition, the possibility of setting the site to idle has been implemented in this panel. The status of each bus can either be displayed on a computer monitoring software or via visualization signals as shown in the figure below.

Signals	Meaning	Status
Red	idle/OFF	Code executed properly
Green	ON	
Yellow	Electric failure	Necessary intervention

Table 1 - Monitoring signals

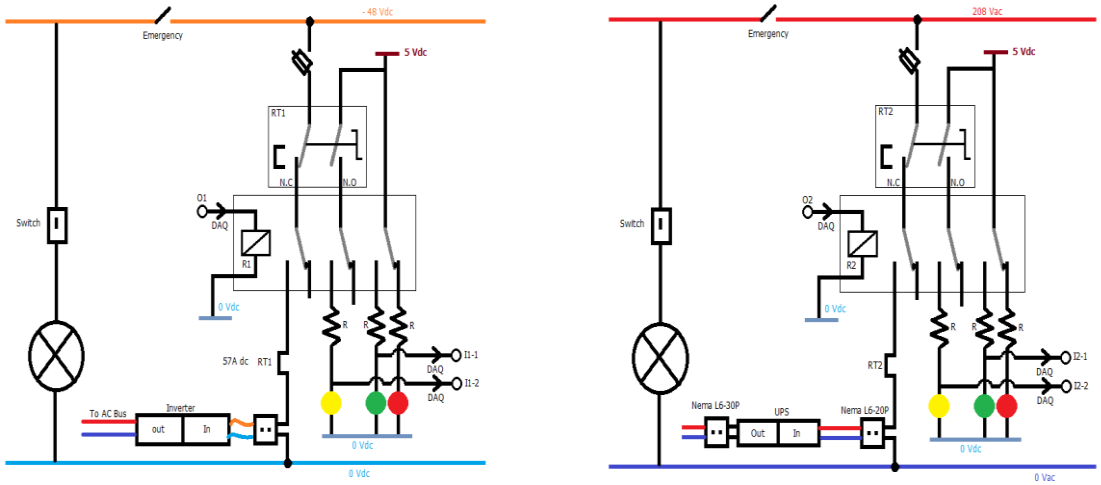


Figure 3 – Electric panels' schematics

TASK 2 - Progress to Date

As a part of Task 2 – Characterization of the Subsystems and the Assembly to Determine the Correlation between Input Power (to POD) and Wind Variability – in the second quarter report an algorithm was presented to meet the demand and schedule the power commitment. The program gives a schedule of running several electricity units in order to meet a pre-deterministic workload. In electricity market operation, for the same power load and same plants or network, electricity price computed based on UC [1] algorithm increases when constraints related to network transmission lines are taken into consideration (NCUC[2]). The constraints related to outage of transmission lines or generators are also included; when these constraints are added to the NCUC the price increases. These constraints are also included in the SCUC[3] algorithm. Significant improvements were made during this quarter.

I. Unit Commitment (U.C) problem:

Workload and power allocation strategy was proposed based on minimizing the price of available energy. Such criterion is subjected to reserve limitations, generator capabilities, and power bus limitations, and other constraints.

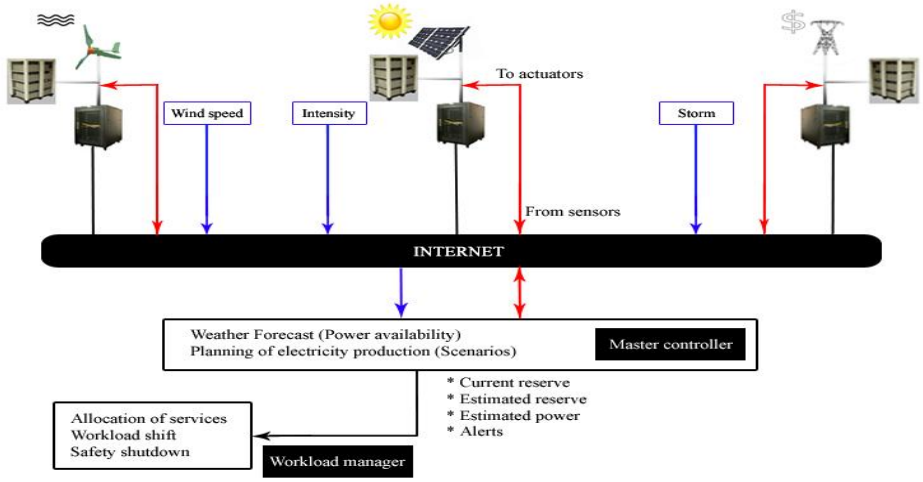


Figure 4 – GDC Architecture

[1] UC: Unit Commitment
[2] NCUC: Network Constraints UC
[3] SCUC: Security Constraints UC

Based on available power in each location and associated price, the code generates a schedule and a power distribution that can meet the desired load demand. As described in Task1-I, the strategy adopted relies on individual sites sending data to the master controller where a Unit Commitment (UC) algorithm is used to make data energy scheduling decisions. If the demand profile can be met, then the Workload Manger will make final decision on how the data should be scheduled. The master controller also sends out alerts about grid outage, drop of wind power, etc. as needed. The algorithm has been further developed to extend its capability to manage 24 hours instead of 8 hours. Available periods of the day are taken into consideration to evaluate the availability of power. Some basic conditions were also implemented (for example the controller avoids relying on solar sites during night cycle).The economic model that represents our scenario includes:

- The price objective function
- The bus capability that is the cumulative power that a bus can get from a generator plus the storage that is coupled to compensate the power drop.
- The generators limitations, that is the minimum power and maximum power that a generator can produce.
- The reserve capability that is the limitations of the storage bank.

Extra constraints, as listed below, were added to better model the site where the grid unit is supplying power to a POD:

- Start Up cost and its constraints
- Shut Down cost and its constraints
- Maximum sustained rate
- Quick Start Capability
- Minimum Up Time
- Minimum Down Time
- Ramp Up rate
- Ramp Down rate

The Appendix A regroups all the equations and inequalities that model our system.

II. UC algorithm simulation

Some initial simulations are provided, using the UC algorithm to show its capability. A 24 hours UC scheduling is presented. The power requirement of the system is provided in Fig. 5 with power demand and desired reserve. Figure 6 shows the results of a feasible scenario. Eventually, if the renewable resources cannot meet the power demand, the grid will be used to compensate. If all locations cannot meet the demand, an infeasibility flag is triggered and workload manager will look for other sites. The results of this simulation are presented in the following example. The figures 5 and 6 represent the scenario reported in table 2.

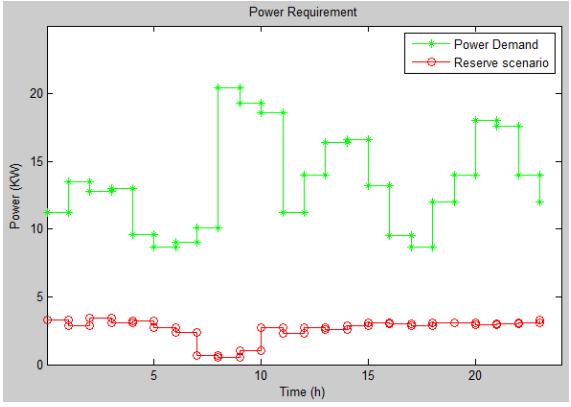


Figure 5 – Power requirement

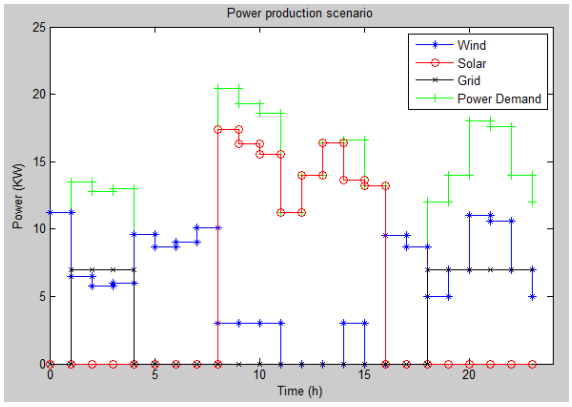


Figure 6 – Power production scenario

Example:

The example shows a possible scenario with power distribution that could meet the following power requirements

Hour	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Load	11.2	13.5	12.8	13	9.6	8.7	9	10.1	20.4	19.3	18.6	11.2	14	16.4	16.6	13.2	9.5	8.7	12	14	18	17.6	14	12
SR	3.26	2.9	3.4	3.1	3.2	2.76	2.4	0.7	0.5	1	2.7	2.3	2.7	2.6	2.9	3.1	3	2.9	3.1	3.05	2.94	3	3.1	3.3

Table 2 - Power requirements

Hour	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Wind	11.2	6.5	5.8	6	9.6	8.7	9	10.1	3	3	3	0	0	0	3	0	9.5	8.7	5	7	11	10.6	7	5
Solar	0	0	0	0	0	0	0	0	17.4	16.3	15.6	11.2	14	16.4	13.6	13.2	0	0	0	0	0	0	0	0
Grid	0	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7	7
Load	11.2	13.5	12.8	13	9.6	8.7	9	10.1	20.4	19.3	18.6	11.2	14	16.4	16.6	13.2	9.5	8.7	12	14	18	17.6	14	12

Table 3 - Optimal power distribution

Hour	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Wind	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1
Solar	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Grid	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Table 4 - ON/OFF status

The status indicators give an hourly clear idea about available sites, which is important information for the workload manager. Once the sites are known, the workload manager can read the power distribution to assess the number of virtual machines that could be shifted from one site to another and that could be hosted in a particular site. The figures of the unit commitment algorithm are represented as stair functions for the evolution of power generation for one day ahead. We can easily observe that solar power (red line) is reliable between 10:00 am and 4:00 pm while wind power is mostly used during nights (blue line). Another observation from the figure is that the grid is used between 1:00 am - 4:00 am and 6:00 pm - 12:00 am. This is due to the constraint that sets the minimum up time to 3 hours. Finally, the UC algorithm will help us to develop models for the next day's power generation plan. However, this is not sufficient to run a real time system especially if the source of energy is intermittent. Hence, the current model will be extended to account for the intermittency in wind speed and wind power properties.

III. Wind measurements:

Wind measurements were taken by two undergraduate students using the weather station on the met tower located at Clarkson University. Measurements at two locations (top and bottom of the antenna), help estimating the potential power available at our site, that can be used in the emulator specs during our laboratory testing. A sample data is reported in Table 5.

Date-Time	Ave Speed	Gust Speed	Wind Direction
6/20/2012 0:00	10.3	13.8	192
6/20/2012 0:01	11	16.6	189
6/20/2012 0:02	10.5	14.2	191
6/20/2012 0:03	8.7	14.7	179
6/20/2012 0:04	10.3	15.2	187
6/20/2012 0:05	13.3	19.8	172
6/20/2012 0:06	9.9	16.6	179
6/20/2012 0:07	11	18	191
6/20/2012 0:08	12.8	18	192
6/20/2012 0:09	8.5	14.2	186
6/20/2012 0:10	8.3	11	188
6/20/2012 0:11	9.7	13.8	189
6/20/2012 0:12	6.9	9.6	191
6/20/2012 0:13	9.2	14.7	191
6/20/2012 0:14	10.7	17	185
6/20/2012 0:15	9	12.8	185
6/20/2012 0:16	10.2	14.7	186

Table 5 – Example of wind data from met tower

A significant size database is available, up to 1440 data points per day. The measurements were split to 24 intervals of 60 minutes each. The 24 plots in Fig. 7 represent the power available data for a full day, for each individual hour. The red dotted line in each plot is the mean power that can be generated during each hour as presented in the table 6.

Hour (h)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
P (KW)	3.10	2.50	2.00	1.50	0.8	0.15	0.68	1.62	1.18	1.03	1.46	1.32	1.53	2.09	2.21	1.79	3.31	7.22	5.64	3.25	2.21	1.12	2.11	2.82

Table 6 - Mean power available

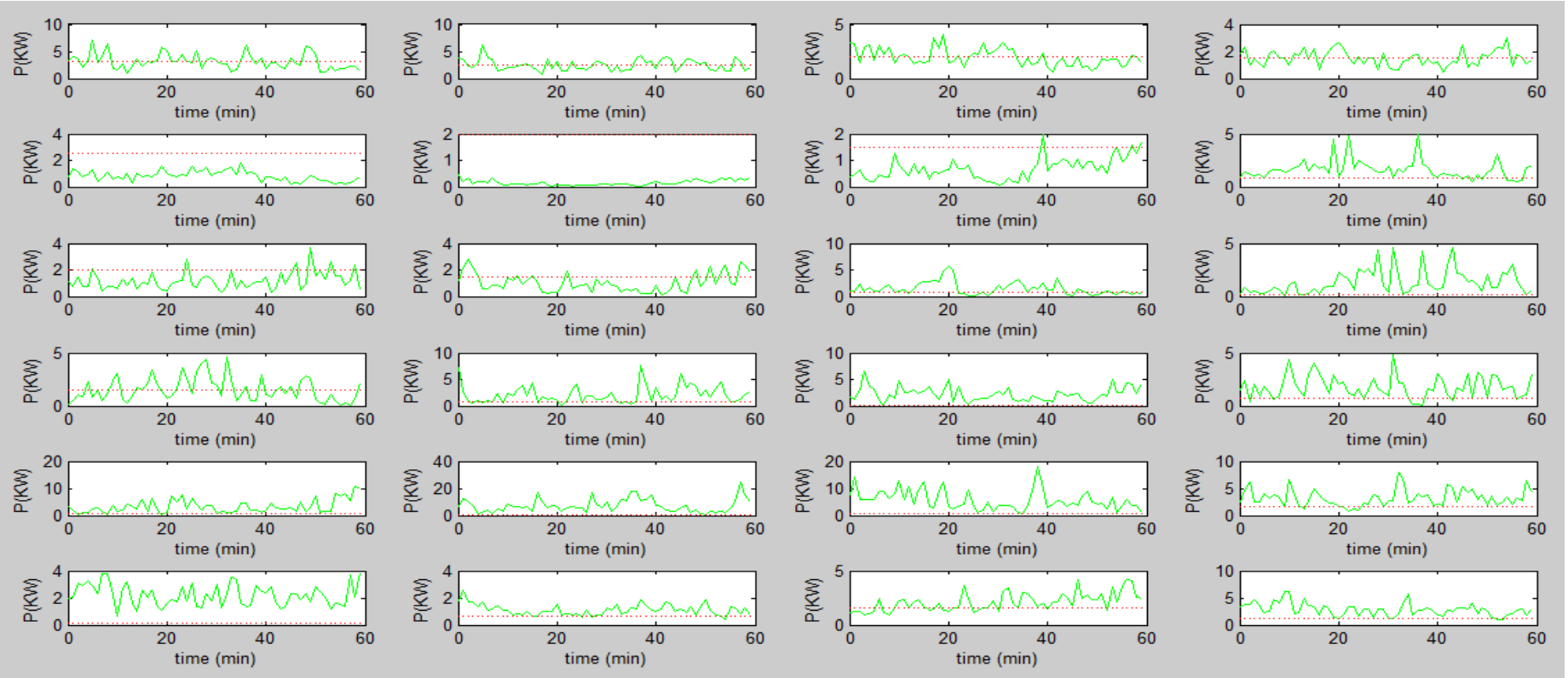


Figure 7 – Example of Wind Power data

Summary:

In the current report period important modifications have been added to the system architecture. A Distributed Control System with a set of local slave controllers communicating with a central master controller, was developed. Such a strategy reduces the computational load of one central controller. The workload manager is now responsible of taking decisions about workload allocation. This approach is significantly different from what has been currently exploited as unit commitment, where it is usually the power manager unit that commits the amount of electricity needed to meet a specific workload demand. Within the proposed model the power manager commits to provide a certain power profile, and then the workload manager will determine the best options on shifting data from one site to another according to the power profile. An overview about how unit commitment can be linked to real data is also provided. The power produced during each hour will be used to meet workload demand.

IV. Hardware Acquisition:

Company	Product	Quantity	Total
Chroma	DC Power Source	1	4,440.00
Iomega	Storage Unit 12TB	1	3,283.20
HP	DL165 G7 Opteron6220	2	7,547.60
	DL385 G7 Opteron6220	1	5,264.98
NetGear	Switch	2	164.74
Unipower	Sabre 3KVA (-48Vdc to 208Vac)	1	3,577.11
APC	SmartUPS SUA3000RMT2U	1	1,502.39
Ioxus	2 KW Ultra-capacitor bank	1	3,290.00
Grand Total			**29,070.02**

Table 7 – Inventory of purchased equipment

V. AC and DC Architectures Updates:

1. Comparison between HP based AC and DC architectures:

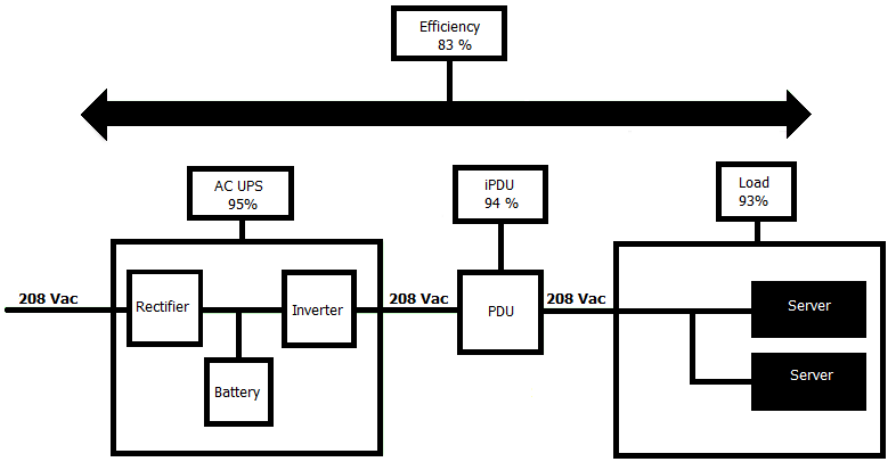


Figure 8a – Updated AC Architecture

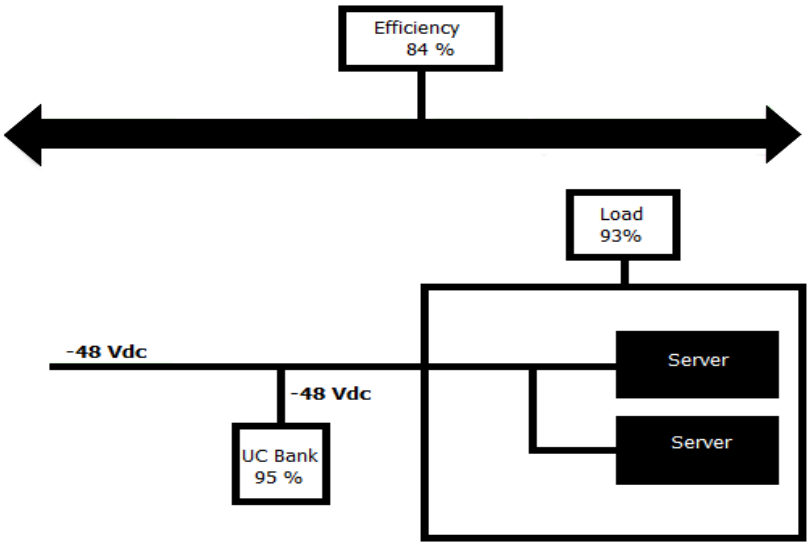


Figure 8b – Updated DC Architecture

VI. Test Bench Organization:

Two experiments will be conducted as shown in Figure 9a and Figure 9b. The experimental set up includes two electric panels, one with 48VDC and the other with 208 VAC.

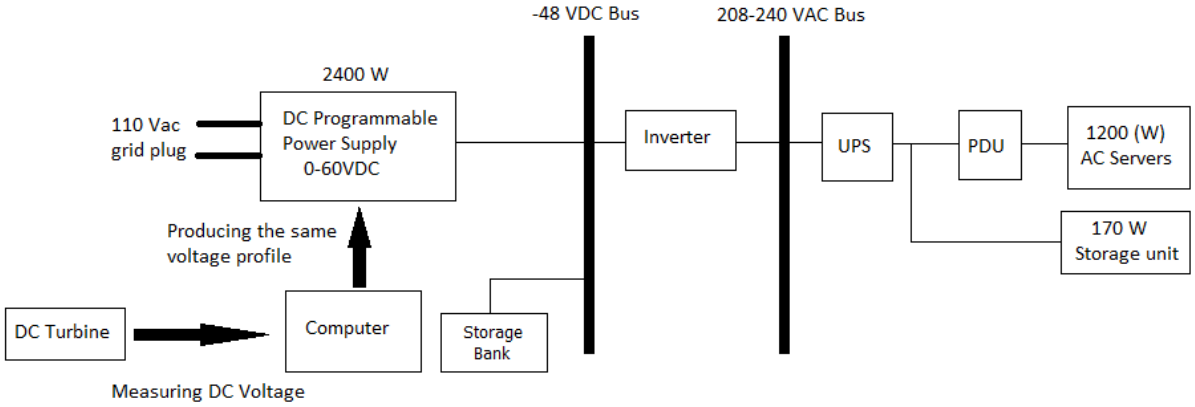


Figure 9a – AC Server Architecture

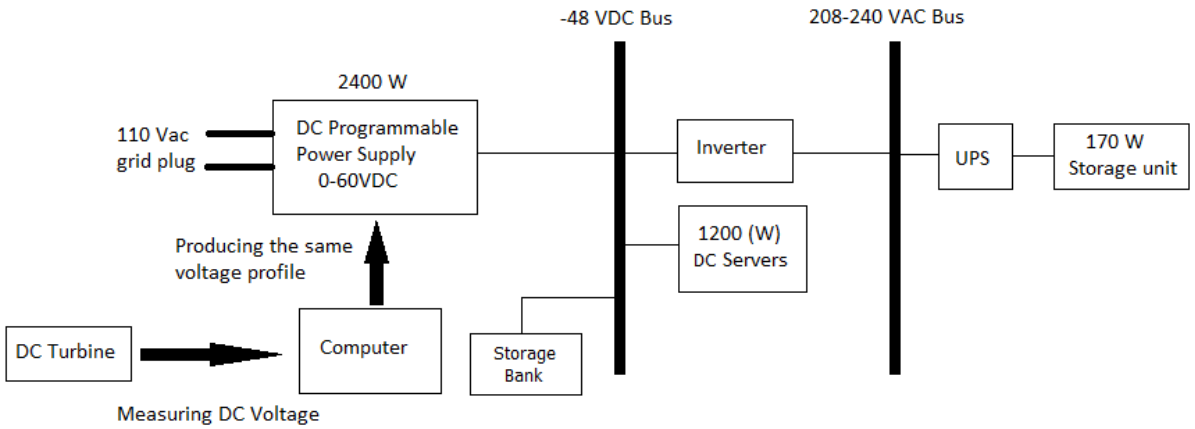


Figure 9b – DC Server Architecture

TASK 3 - Progress to Date

The work on Task 3 just started this month. This task deals with the development of a system level model using experimental data to be used in design optimization. We are acquiring initial experimental data from system components which are instrumental in this task.

TASK 4 - Progress to Date

The goal of Task 4 is to characterize the workloads for which hosting in a POD datacenter would be a good match. Part of this task includes a detailed study of existing virtual machine migration technologies and an evaluation of the role virtual machine migration could play in shifting workloads from one POD datacenter to another to take advantage of available power. In addition, we would like to explore a number of other tools and strategies for making POD datacenter hosting a viable alternative for a larger range of workloads. We expect that there will be some workloads for which POD datacenter hosting is a natural fit (e.g. hosting of static content) and other workloads for which POD datacenter hosting may never be a good fit (e.g. write intensive workloads with high availability and coherency requirements). In addition to virtual machine migration, we plan to explore other tools and strategies for managing downtime at one POD datacenter and/or shifting work from one POD datacenter to another according to available power.

I. Virtual Machine Migration

The first set of subtasks we have identified involves conducting a series of quantitative experiments of virtual machine migration. Most of the popular server class hypervisor/ virtualization technologies including KVM, VMware, Xen, and Hyper-V offer a form of virtual machine migration. However, in many instances, it is assumed that the migration will take place between machines in the same datacenter and possibly even on the same network switch. In many cases, the machine initiating the migration must have access to the same network attached storage device (NAS) as the machine accepting the migrated VM and only the memory state is migrated from one machine to another. Some hypervisors also support live migration with storage migration where the disk state of the VM is transferred as well as the memory state. Our first task is to perform a thorough evaluation of these migration technologies in a wide range of environments – between two machines in the same rack sharing a NAS, between two machines in the same rack not sharing a NAS, between two machines located in two different labs on campus with and without NAS and finally between an on-campus machine and an off-campus machine with and without NAS.We are in the process of collecting a variety of interesting measurements of virtual machine migration including the total time to accomplish the migration (from start to finish), the downtime or time the virtual machine is unresponsive during migration (typically during the last stage of migration) and the total amount of data transferred to accomplish the migration. Eventually, we would also like to collect data on the total power required on the transmitting side to complete the migration. We will also collect these same measurements of two extreme configurations – “cold” migration and a high-availability pair. In cold migration, the VM is suspended, the files transferred to other side and the VM resumed. This represents a worst-case scenario for the amount of time a VM will be unresponsive but makes the fewest assumptions about the environment (no NAS required for example). In a high-availability pair, VMs run constantly on both machines so full VM migration is never required. In some

cases, data will be shipped from one VM to another to keep the two VMs in sync. The only thing that changes is where requests are sent (to both VMs if available, to just one, etc.).

1. Progress to Date

- Surveyed advertised features/requirements of several major hypervisors and cloud orchestration tools, and added this information into a master spreadsheet
- Set up mini-GDC architecture in the lab with new HP servers and Iomega NAS. We are using this new hardware to repeat measurements taken on some older hardware we had available in the lab over the summer.
- Set up a similar rack of servers in one of the labs in CAMP which can, in part, be used for testing cross-campus migration
- Resolved the problem of being able to tune network bandwidth by using Netgear managed switch
- Improved GDC testbed to be compatible with all kinds of virtualization by removing dependencies on using the hypervisor or host OS for monitoring
- Captured a substantial amount of additional data on live migration with KVM (both memory migration and storage migration) using existing (older) hardware in our lab with VMware (both memory migration and storage migration)
- Assembled models for using GDC beyond virtual machine migration including rolling queries converging to a final answer

2. Next Steps

- Extend our test to Citrix XenServer and Microsoft Hyper-V on HP servers
- Improve our test suite (more automation, more automatic correlation of measurements, modifications to investigate anomalies)
- Explore several puzzles discovered by the current results of live migration
- Obtain XenServer's license and set up Xen testbed and complete Xen-related quantitative testing.
- Obtain Microsoft Server license and set up Hypver-V testbed and complete HyperV-related quantitative testing.
- Complete the live migration quantitative tests, publish the results for external review

II. Current Status (Detailed Status)

1. Logical Overview of GDC Testbed

Upon experience of previous testing, we have further refined our testbed and made it more general for quantifying the performance of live migration from a variety of perspectives.

- Our testbed now includes two network segment – internal and external – allowing us to isolate the migration traffic being measured from any network traffic caused by the measurement itself.
- We moved all of our measurement infrastructure to an separate monitor machine (the GDC workload controller). We no longer rely on taking measurements from the hypervisor or a management VM on the same machine with the migrating VM. This reduces a source of measurement noise. It also increases the compatibility with different hypervisor platforms because we no longer need to rely on support for our measurements tools in each hypervisor.

- We added a fine granularity test for VM downtime. This test involves frequent pings to the migrating CM to demonstrate clearly how network service is impacted by migration.

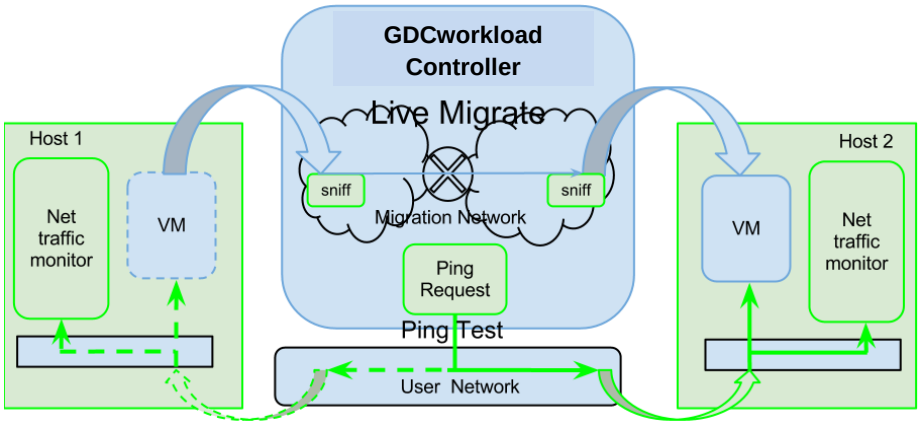


Figure 10 – GDC Testbed Architecture

2. Detailed Setup of Hardware

We purchased two HP servers with two AMD Opteron 6220 8-core 3.0G CPUs and 64G memory. These specifications are a better match for modern datacenter hardware than the older hardware we used over the summer. We install the virtualization hypervisor under test on each server installed Virtualization hypervisor to host VMs. Each site is also equipped with an 8-Terabyte Network Attached Storage (NAS) which will host VM images on RAID 10. Together a HP server and a NAS represent a small GDC in our prototype. Customer services are hosted in VM and can be relocated or migrated across the servers. The picture below is taken in Clarkson Computer Lab and thanks to them for allowing us to host our GDC project in their server room.

Figure 11 – Current GDC computing setup

At the same time, we also have two auxiliary machines to accomplish the test. The GDB workload contoller or monitor machine mentioned above is an IBM eServer xSeries345 installed with Ubuntu. It is responsible for starting the migration, measuring the migration time and downtime. When testing VMware, we also use an additional machine to run

VMware vCenter which is required to manage the VMware ESX servers. In this case, the workload controller will contact vCenter and ask vCenter to start the live migration. For these two additional machines, we were able to repurpose some exsting machine in the lab.

3. Hypervisors Under Test

We have retested the live migration capabilities of KVM3.2.0 and VMware vSphere5.0 on the new HP servers.

KVM supports two types of live migration:

1. Migrate CPU and memory but storage stays on shared datastore (virsh migrate --live)
2. Migrate all CPU, memory, and storage. Storage will also be migrated together with CPU and memory (virsh migrate --live --copy-storage-all)

VMware supports Host Live Migration and Storage Live Migration.

1. Host Live Migration only migrate CPU and memory, storage has to be on shared datastore (MigrateVM_task)
2. Storage Live Migration only migrate storage while CPU and Memory stays on the host (RelocateVM_task)

Thus, VMware Live migration requires shared datastore as the media. If VMware wants to migrate all CPU, memory, and storage of a VM, it has to be split into two separate phases: first the host is migrated with Host Live Migration and then the storage is migrated with Storage Live Migration or vice versa.

4. Live Migration Stress Test Results

Our current tests are focusing on how activity running inside the migrating virtual machine impact migration time. Based on the previous coarse-granularity testing from the summer, we found that memory activity is a bigger factor during live migration compared to CPU activity or read write traffic to disk.

We use our memory stress test, which is programmed to allocate a specific memory size and then dirty a specified portion of that total allocation. We can control both the amount of memory that is written or dirtied as well as the rate at which data is dirtied.

i. Results on new hardware compared to previous results

We have repeated our testing from over the summer in our new testbed. For these new tests, we have used a more realistic VM specification with 2G memory and 40G virtual disk instead of 1.791G memory and 5G virtual disk. Due to hardware limitations on the older machines, we were forced to use less realistic VM specifications over the summer. Below is the comparison of previous result on the older Dell Optiplex750 and new results on HP DL165G7:

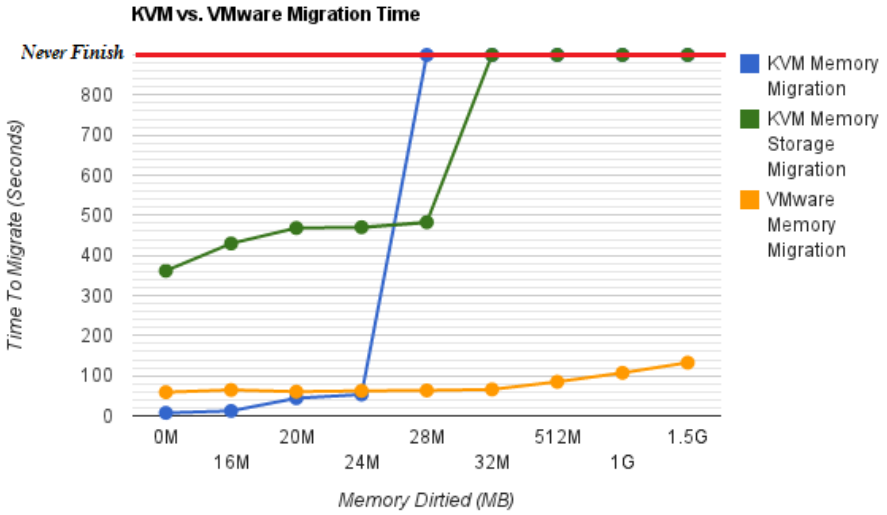


Figure 12 – Previous Coarse-granularity Results on Dell Optiplex750

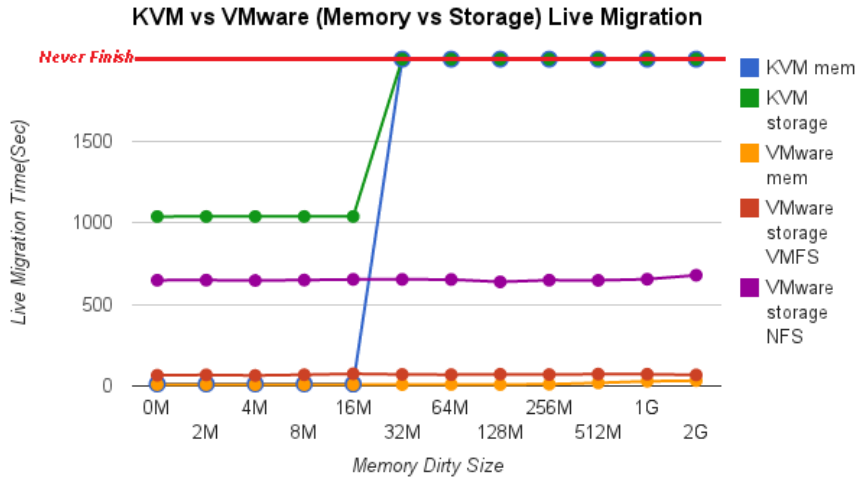


Figure 13 – Latest Result of KVM vs. VMware with memory and storage migration on HP DL165 G7

From the re-test of KVM vs. VMware, we can confirm for live migration with the following conclusions.

- With intensive memory dirty rate, KVM still cannot finish memory migration upon 32 MB dirty memory size while VMware can always finish memory migration and as memory dirty size grows, the migration time also prolongs. With small amount of memory dirty size variants, memory migration time difference is not that obvious. Actually VMware takes less migration time than KVM although the difference is not that dramatic. Storage migrations take a dramatic and dominant time compared to memory migration. It is easier to migrate VM memory than Storage.
- KVM storage migration is also stuck at 32MB impacted by dirty memory size. It is due to KVM storage migration including memory migration.
- Notice that in the new results we have also included results on VMware storage migration. KVM and VMware has the same specification of VM storage, but KVM takes more time than VMware's. For VMware, storage migrations from VMFS and from NFS also make a big difference. More details will be discussed in the section of Storage migration.

ii. Status of Xen Server testing

We've tried to deploy Citrix XenServer on HP server, but failed to deploy Ubuntu test VM on Xenserver, reporting the error message

"`Warning: Failure trying to run: chroot /target dpkg --force-depends --install /var/cache/apt/archives/debconf_1.5.42ubuntu1_all.deb`"

during the installation. The root cause is Ubuntu failed to recognize the avx CPU flags specific to AMD CPU by eglibc and caused the confusion.

For more details, you can refer to the bug:

https://bugs.launchpad.net/ubuntu/+source/eglibc/+bug/956051 .

This bug only applies to AMD CPU Operon 62xx Family. We are still waiting for the patch so that we can test on Xenserver.

iii. Overview of Memory Migration

In this section, we define a variety of terms to better understand how memory affects live migration.

Configured Memory (VM Configured Memory)

Configured Memory is the amount of memory given to the virtual machine by the hypervisor. To the VM guest, this looks like the amount of "physical memory" available to it for use.

Allocated Memory (Hypervisor Allocated Memory)

Allocated Memory is the amount of physical memory on the underlying hardware that the Hypervisor has actually allocated to a guest VM. This may be less than the VM configured memory and is reported from the perspective of the hypervisor. Allocated Memory indicates the portion of VM Configured Memory that VM has actively used. If a VM uses memory and then frees it, the allocated memory may or may not be reduced.

Used Memory (VM OS Used Memory)

Used Memory is the memory currently actively in use by VM. Those memory pages are requested by VM system and resident lively inside VM memory. It is reported from the perspective of the guest VM (e.g. with a utility like top). VM total Memory is the same as VM Configured Memory.

Requested Memory (Application Requested Memory)

Requested Memory is the memory an application running inside VM has requested from the VM's guest. The requested memory is not guaranteed to be all resident in memory and those not currently in use may be swapped out to VM disk when VM Configured Memory are all used. Requested or allocated memory for the application may not all be in use.

Dirtied Memory (Application Actively Dirtied Memory)

Dirtied Memory is the memory that an application is actively modifying via write to that memory. Dirtied memory is part of the requested memory of an application that are actively used. They are typically resident rather than swapped out to disk..

In general, the memory size relationship can be viewed as below: VM Configured Memory > Hypervisor Allocated Memory > VM OS Used Memory > Application Requested Memory > Application Actively Dirtied Memory. Their relationship can be illustrated in the figure below:

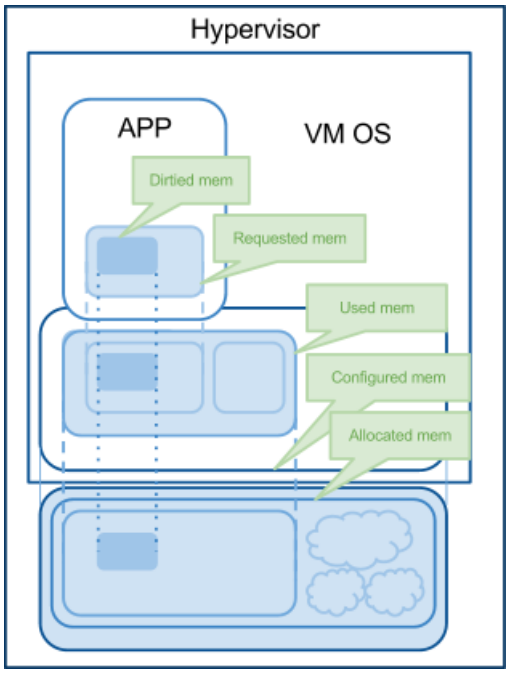


Figure 14 – Concepts of VM Memory in Virtualization

For Memory Live migration, the allocated memory is a more accurate parameter to describe the migration data and time rather than used memory. Configured memory is only a cap for VM but doesn't directly impact live migration traffic or time.

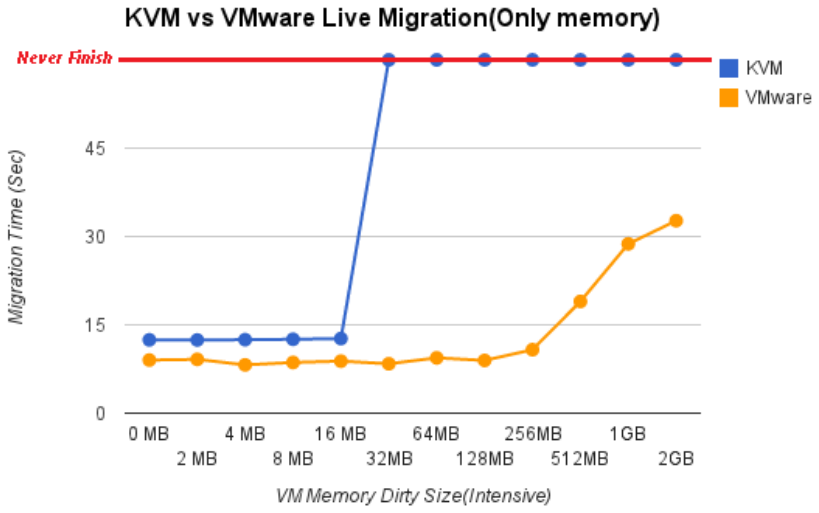


Figure 15 – More close look at KVM vs VMware Memory Migration Results with dirty memory Update on Citrix XenServer

As dirty memory size goes up, the migration time will also increase. Each time a memory page is changed it must be copied over again. For each memory migration periods, when Hypervisor copies all the memory over, it will examine and figure out the new dirtied pages and continue to copy it over. And then at some point, Hypervisor will freeze the whole VM CPU and memory state and copy them all over and renew it on the destination hypervisor. Dirty memory size plays a secondary important role in migration time in the VMware case. In KVM case, it is hard to tell because the dirty size has no chance to scale up while dirty rate already becomes an important factor to prevent it from finishing memory migration.

iv. Overview of Storage Migration

In storage migration, the dominant factor is the size of the disk space being transferred. By its sheer size, this swamps the impact of memory size and amount of memory dirtied. As the storage data goes up, the migration time can become prohibitively high.

To better understand the storage migration; first have a look at the different storage concepts:

Virtual Disk Size (VM Virtual Disk size)

Virtual Disk size is the size of the "physical" disk offered to the VM for its use. It is typically defined when the VM is created and hard-coded inside VM image which Hypervisor can read and will be the maximum data size that VM image can hold.

Allocated Blocks (Hypervisor Allocated Disk size)

Allocated Disk size is the disk block size that Hypervisor-Hosted File system has allocated to the VM image. Most likely, it is the maximum-size-once-used by VM. Once disk block is allocated to VM image, Hypervisor FS will rarely garbage collects them back unless FS has some scan and chkdisk mechanisms like Linux gparted program. It depends on Hypervisors' capability and most of which is not aware.

VM FS size

VM File system size is the real data size stored in VM image and used by VM OS. It is VM-aware but hard for hypervisor to be aware.

In general, the disk size relationship is VM Disk > VM Allocated Disk > VM FS. Their relationship can be illustrated in the figure below:

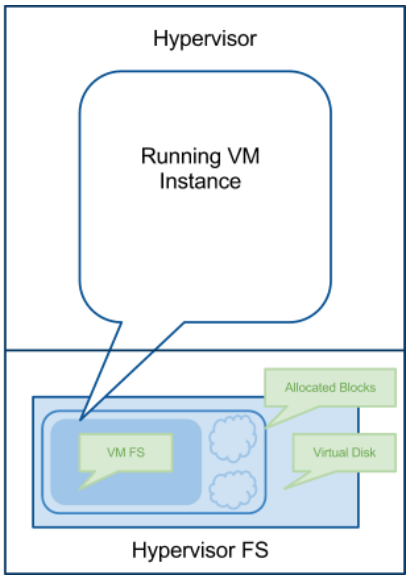


Figure 16 – Concepts of VM Storage in Virtualization

For live storage migration, KVM migrates the whole virtual disk from local Ext4 File system instead of allocated blocks while VMware has two different scenarios: When migrating VM image from VMware FS VMFS, it only migrates the allocated blocks rather than the whole virtual disk. However, when migrating VM image from NFS, it is the same as KVM storage migration. You can see in Figure "Total Migration Data", KVM storage line (Green) is collapse with VMware storage NFS (Purple).

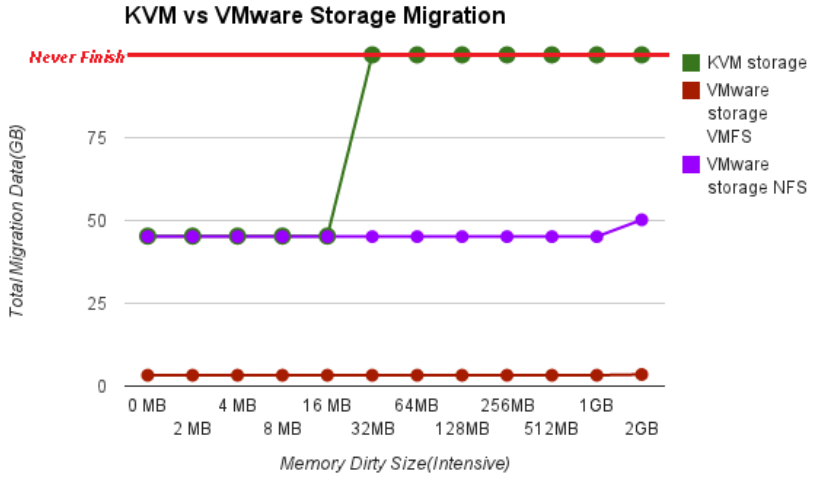


Figure 17 – KVM vs VMware Storage Migration Transferred Data over Network

Migrating allocated blocks or virtual disk can make a big difference when VM OS doesn't fully use the virtual disk space. Although KVM storage migration also includes memory migration, but VMware still has less migration time than KVM.

v. Drilldown on Live Migration Downtime

On one hand, we care about the total live migration time. On the other hand, we also need to consider the downtime or the amount of time the migrating VM is unresponsive. In our tests, we measure this downtime by regularly pinging the VM during its migration and noting the pattern of its replied. This reveals how long the network service is interrupted due to the VM transition. We used advanced feature of ping by intensively ping the VM with the interval of 0.01 second.

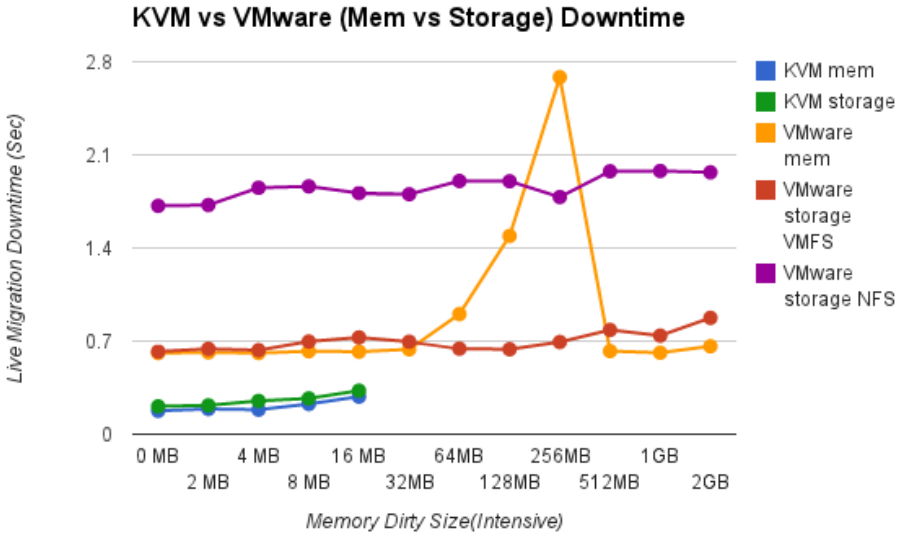


Figure 18 – Live Migration introduced Downtime by KVM and VMware

From the re-test of KVM vs. VMware, we can conclude the following:

- KVM memory migration has much less downtime than VMware memory migration, although VMware has better migration time than KVM.

- Storage Migration even without memory migration also faces downtime. Memory migration suffers downtime because there is a CPU state transition happens. But for VMware, VM CPU don't need to transit but only the image location changes, VM still suffers the downtime.
- For the same virtualization, memory migration and storage migration experience the similar downtime.

Furthermore, we drill down and try to picture exactly what happens to the packets during the downtime for each virtualization. Our testbed can capture exactly what happens during the downtime for each virtualization. We found out that their downtime and packet loss patterns are also different:

- In VMware only loses packets at source, once VM shows up on destination, VM can immediately receive ICMP requests and reply normally. It indicates VM is back instantly after migration while KVM only loses packets at source. It indicates VM takes some time to come back to normal after migration.
- VMware caches incoming packets at source when migrating, and then replies immediately all at once at the destination. VMware doesn't need to cache any packets at destination. However, KVM doesn't cache but lose some packets at source. At destination, KVM caches packets at destination and cannot reply immediately when it finishes migration, then reply all at once at some point.

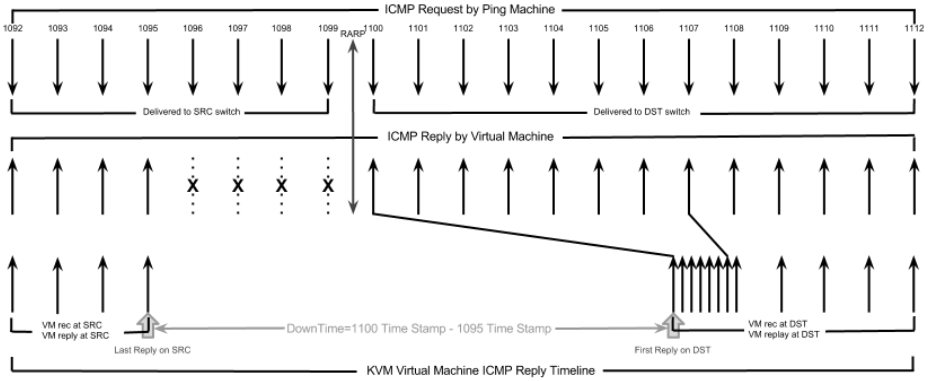


Figure 19 – KVM ping timeline at interval 0.01 second

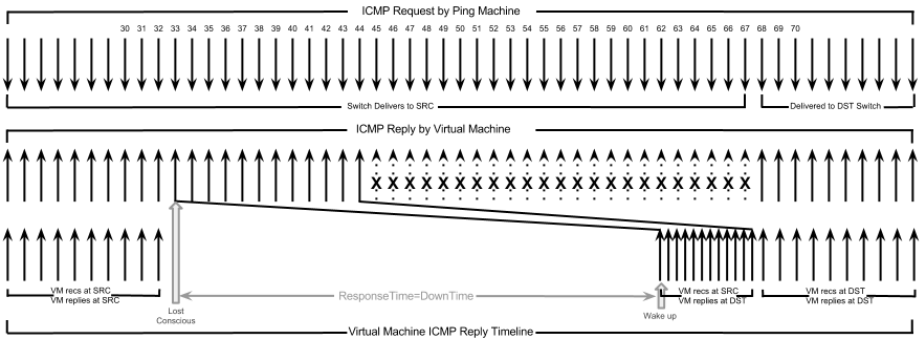


Figure 20 – VMware ping timeline at interval 0.05 second

5. Consistency and Availability Concerns

Consistency and availability concerns for the POD-GDC conceptual approach has been the focus of a graduate student course led by J. Matthews at Clarkson (CS644). Consistency

and availability concerns for the POD-GDC conceptual approach has been the focus of a graduate student course led by J. Matthews at Clarkson (CS644). Systems that trade-off the consistency, availability, and performance of workloads hosted in geographically distributed datacenters has been the focus on work published in major system conferences by researchers at Google, Amazon, Facebook, Microsoft, Yahoo and by a variety of academic institutions. In general, this existing literature assumes that whole datacenter failure will be a rare event and that the primary failure model will be individual machines or racks of machines within a datacenter. This limits the applicability of the existing literature to the GDC model where datacenters fail regularly due to unavailability of power. Still, we are carefully reviewing this existing literature and cataloging lessons we can take from it and apply to this project.

In general, for any distributed system containing multiple copies/replicates, it is impossible to maintain strong consistency, provide high availability, and tolerate network partitions at the same time. This is known as the CAP theorem [4]. Availability means that a user can always access the data. Consistency means multiple users see the same result when accessing the same data, and partitions refer to ad-hoc network disconnections, hardware failure, or server power outage/crash. Performance means that both read and update operations will regularly experience low latency.

Following the guidance of the CAP theorem, the GDC project faces intermittent power outages and thus cannot achieve high availability and strong consistency at the same time. Web services will be interrupted under GDC operations unless the availability or consistency requirements are relaxed.

Current data center services such as Google or Amazon are designed to handle hardware failure, not frequent site outages. The GDC model has a different vulnerability. When hardware fails in these conventional services they compromise by providing weak consistency but high availability to achieve a user-acceptable service. However, their model is based on an assumption of a small amount of hardware failure at any given time. No approach can handle datacenter failure seamlessly.

Yet under GDC conditions of intermittency and high distribution, a variety of web services and applications have serious potential. These are generally simpler, less demanding, and non-mission-critical server functions that can be hosted in GDC. The GDC model can provide either high availability or strong consistency (but not both).

Potential GDC workload applications:

- Develop web service APIs (application programming interfaces) for Google, Amazon, Facebook, Tweeter, etc. Provide archived data services for these firms via the GDC model.
- A model for higher latency access to older data, similar to Amazon Glacier.
- Provide a "green server" service that allows for occasional data/service non-availability. Many services and features exist which may have less reliability or more expensive outcomes but which are still very attractive to some customers because they are energy efficiency and "green."

Finally, a highly developed GDC-POD model that is implemented at scale may actually have minimal limitations in availability and consistency. The most important factors will be the success of the master controller algorithms and also the degree of diversity of energy source locations and types. Many locations, with many different types of renewable sources, could considerably reduce the amount of service failure, essentially making it a relatively minor

[4] S.Gilbert and N.Lynch, Brewer's conjecture and the feasibility of Consistency, Availability, Partition-tolerant web services. ACM SIGACT News, 33(2), 2002

problem. Future research should focus on simulations of the GDC availability mode to better understand the conditions that GDC web service will face.

b. Identification of Problems & Planned Solutions.

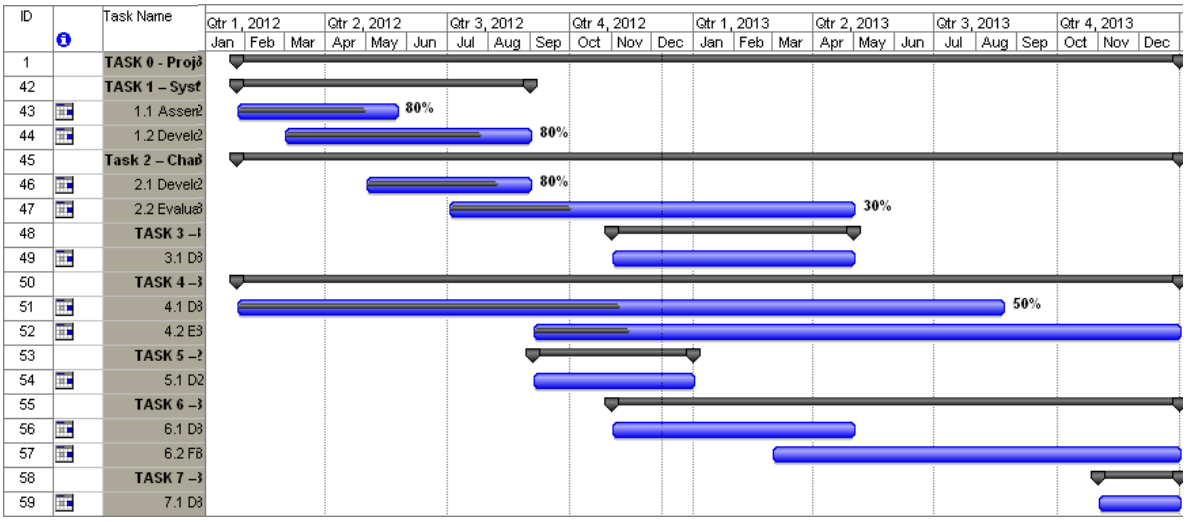
Tasks 2 and Task 4 are on schedule. Task 1 suffered a short delay associated with the lead time for some of the hardware components, customized for our specific application. We are still waiting for the ultra-capacitor bank but we are progressing on different fronts as not to encounter further delay.

c. Ability to meet schedule, reasons for slippage in schedule, unforeseen obstacles, promising research directions not originally identified in current SOW

No significant slippage or unforeseen obstacles encountered in the project as of now. Work is going according to the plans, although there is a short delay in completing Task 1 as indicated above. A set of HP servers with AMD processors and other equipment has been acquired and are currently used as testbed for our initial laboratory demonstration.

d. Schedule - percentage completed and projected percentage of completion

ID	Task Name	Duration	Start	Finish
1	**TASK 0 - Project Management**	**504 days?**	**Thu 1/26/12**	**Tue 12/31/13**
42	**TASK 1 – System Design and Integration**	**157 days?**	**Thu 1/26/12**	**Fri 8/31/12**
43	1.1 Assemble the PODs and the Operation Management Unit	86 days?	Thu 1/26/12	Thu 5/24/12
44	1.2 Development of the Energy Management Unit and Power Supply	132 days?	Thu 3/1/12	Fri 8/31/12
45	**Task 2 – Characterization Subsystems and Assembly, Determine Correlation Input Power (to POD) to Wind Variability**	**504 days?**	**Thu 1/26/12**	**Tue 12/31/13**
46	2.1 Development of Input Power Profile	89 days?	Tue 5/1/12	Fri 8/31/12
47	2.2 Evaluate POD System Power Performance from the Measured Data and Derive Correlations	218 days?	Mon 7/2/12	Wed 5/1/13
48	**TASK 3 – Development of a System Level Model Using Experimental Data for Design Optimization**	**129 days?**	**Thu 11/1/12**	**Tue 4/30/13**
49	3.1 Development of a System Level Model Using Experimental Data for Design Optimization	129 days?	Thu 11/1/12	Tue 4/30/13
50	**TASK 4 – Migration of Datacenter Workloads into and out of a POD in a controlled environment**	**504 days?**	**Thu 1/26/12**	**Tue 12/31/13**
51	4.1 Develop a Model for Predicting the Time, Bandwidth and Power Required to Migrate Datacenter Workload	409 days?	Thu 1/26/12	Tue 8/20/13
52	4.2 Experiment with Virtual Machine (VM) Migration to/from a POD	347 days?	Mon 9/3/12	Tue 12/31/13
53	**TASK 5 – Development of Prototype System and Field Demonstration Specification**	**86 days?**	**Mon 9/3/12**	**Mon 12/31/12**
54	5.1 Development of Prototype System and Field Demonstration Specificatio	86 days?	Mon 9/3/12	Mon 12/31/12
55	**TASK 6 – Field Demonstration**	**304 days?**	**Thu 11/1/12**	**Tue 12/31/13**
56	6.1 Design of Field Demonstratio	129 days?	Thu 11/1/12	Tue 4/30/13
57	6.2 Field Demonstration of Prototype System	218 days?	Fri 3/1/13	Tue 12/31/13
58	**TASK 7 – Development of Design Guidelines for Subsequent Demonstrations**	**43 days?**	**Fri 11/1/13**	**Tue 12/31/13**
59	7.1 Development of Design Guidelines for Subsequent Demonstrations	43 days?	Fri 11/1/13	Tue 12/31/13



e. Dissemination and Meetings

Several presentations were made earlier this summer to different forums as presented in the second quarterly report. The team is working on a number of presentation/publications that will be used to disseminate the finding of this research and promote the GDC concept.

S. Bird completed a draft concept paper to be submitted for publication before the end of the year. A first draft has been circulating among faculty and colleagues at AMD. This paper outlines the general conceptual approach of the POD project and lays out the research agenda for the project. It also defines some of the challenges for this model in at-scale implementation. UT Austin collaborators are also being included in the edit process. NYSERDA will be consulted prior to any publication. The team is evaluating potential opportunities for the GDC project with NSF including the System Science program and the Grant Opportunities for Academic Liaison with Industry (GOALI) project. The team has been invited by Brookhaven National Lab (DOE) to participate in the Advanced Energy 2012 Conference, October 30-31 in NY City. The GDC concept will be presented in a session on Grid Integration. Graduate students are also attending this meeting and presenting posters. O. Aitmaatallah and P. Werden are presenting posters on Algorithm Development and on System Integration respectively.

Additional Activities: S. Bird has brought another graduate Master's student, Paul Werden, from Queen's University (Canada) to assist in analysis of the project. Paul is currently being co-advised by Prof. Bird and Professor Andrew Pollard (Queen's Research Chair in Fluid Dynamics and Multi-scale Phenomena). Werden's research is focused on optimization models to determine ideal sizing of transmission capacity, storage capacity and POD size for a given renewable energy source. His work will include empirical analysis of cases studies in both New York and Ontario, and include aspects of the grid system operations in his analysis (regulatory constraints, reliability standards, renewable incentives, and green energy policy). Paul is currently self-funded through his Canadian University. His research started in early September.

f. Analysis of actual cost incurred in relation to budget

Total cost incurred up to July 15th: 95,070.87 (NYSERDA portion)
Student Salary – $30,537.97 (budget allocation - $89,320.00)
Faculty Salary Statutory – 11,155.08 (budget allocation - $37,692.00)
Travel expense – $1,307.80 (budget allocation - $10,000.00)
Equipment - 29,000.00 (budget allocation - $40,000.00)
Indirect cost – $23,000.00 (budget allocation - $88,946.00)

Cost-share – **$73,981.00** (Clarkson)

Student Salary	$11,000.00
Student Tuition	$54,481.00
Summer Students	$5,000.00 (McNair Scholar)
Travel	$3,500.00

Cost-share – **$40,000.00** (Partners)

AMD Corporation	$30,000
HP Corporation	$10,000
WindE Systems	$5,000
Ioxus	$5,000

Appendix A

Objective function: $\min \sum_{t=1}^{NT}(\sum_{m=1}^{NG}(Co_i I_{it} + \sum_{n=1}^{Nseg} IC_{ni} P_{nit} + CSU_{it} * y_{it} + CSD_{it} * z_{it}))$

Constraints:

Power Capability:

$$P_{it} = P_{min,i} I_{it} + \sum_{n=1}^{Nseg} P_{nit}$$

$$P_{min,i} I_{it} \leq P_{it} \leq P_{max,i} I_{it}$$

Power-Load balance:

$$\sum_{m=1}^{NG} P_{it} = D_t$$

Startup (y), Shutdown (z), and ON/OFF (I) indicators:

$$y_{it} - z_{it} = I_{it} - I_{i(t-1)}, \quad y_{it} + z_{it} \leq 1$$

Reserves:

$$sr_{it} + P_{it} \leq P_{max,i} I_{it}$$

$$0 \leq sr_{it} \leq (10MSR_i) * I_{it}$$

$$SR_t \leq \sum_{i=1}^{NG} sr_{it}, \quad OR_t \leq \sum_{i=1}^{NG} or_{it}$$

Quick start capability:

$$or_{it} = sr_{it} + (1 - I_{it}) * QSC_i$$

Ramping up constraints:

$$P_{it} - P_{i(t-1)} \leq y_{it} P_{min,i} + (1 - y_{it}) * RU_i$$

Ramping down constraints:

$$P_{i(t-1)} - P_{it} \leq z_{it} P_{min,i} + (1 - z_{it}) * RD_i$$

Minimum up time constraints:

$$UT_i = \max\{0, \min[NT, (MU_i - TU_{i0}) * U_{i0}\}$$

$$\sum_{t=1}^{UT_i} (1 - I_{it}) = 0$$

$$MU_i * y_{it} \leq \sum_{m=t}^{t+MU_i-1} I_{it}$$

$$0 \leq \sum_{m=t}^{NT} (I_{it} - y_{it})$$

Minimum down time constraints:

$$DT_i = \max\{0, \min[NT, (MU_i - TU_{i0}) * (1 - U_{i0})\}$$

$$\sum_{t=1}^{DT_i} I_{it} = 0$$

$$MD_i * z_{it} \leq \sum_{m=t}^{t+MD_i-1} (1 - I_{it})$$

$$0 \leq \sum_{m=t}^{NT} (1 - I_{it} - z_{it})$$

Congestion constraints:

$$P_{min,i} * I_{it} \leq P_{it}^c \leq P_{max,i} * I_{it}$$

$$|P_{it}^c - P_{it}| = 0$$

Boundaries:

$$0 \leq P_{it}, P_{it}^c \leq \infty$$

$$0 \leq sr_{it}, or_{it} \leq \infty$$

$$0 \leq I_{it}, y_{it}, z_{it} \leq 1$$

Index	Designation
i	unit index
t	hour index
m	segment index for piecewise curve
Variables	**Designation**
I	Status indicators (I=1 ON, I=0 OFF)
y	Startup indicator
z	Shutdown indicator
P	Power produced
sr	Spinning Reserve
or	Operating Reserve
Constants	**Designation**
NT	Number of hours
NG	Number of generators
Nseg	Number of segments of the piecewise curve
D	Demand (in term of Power)
SR	Spinning Reserve requirement
OR	Operating Reserve requirement
ST	StartUp cost
SD	ShutDown cost
Co	No-Load cost
IC	Incremental cost
Pn	Power at each piecewise curve segment
Pmin	Minimum Capacity
Pmax	Maximum Capacity
MSR	Minimum sustained Rate
QSC	Quick Start Capability
MU	Minimum UP time
MD	Minimum Down time
TD	Number of hours a unit has been offline initially
TU	Number of hours a unit has been online initially
RU	Ramp Up rate
RD	Ramp Down rate

Table 8 – Variables and their Designation