



***DICE Technical Report:***  
**DICE Evaluation of  
eXludus MCOpt Multicore Manager  
Server Throughput Technology**

**eXludus Technologies, Inc.  
407 rue McGill  
Montreal, Quebec  
Canada, H2Y 2G3**

**TR-EXL-001-2011**



4170 Allium Ct., Springfield, OH 45505, [www.diceprogram.org](http://www.diceprogram.org)



## Table of Contents

1. Introduction .....	3
2. Background .....	4
2.1. Offered Capabilities.....	4
3. Key Evaluation Criteria .....	8
4. Evaluation Setup .....	8
5. Evaluation Results.....	9
5.1. Evaluation Findings.....	9
5.1.1. Software Installation and Setup .....	10
5.1.2. Configuration Considerations for the MCOpt and JobProfiler Setup .....	11
5.1.3. GLOBAL Database Configuration .....	17
5.1.4. Management and Operation .....	19
5.1.5. Job Control under MCOpt.....	20
5.1.6. Testing Results .....	22
5.1.7 Functional Testing.....	27
6. Summary.....	31



## 1. Introduction

This project included an in-depth investigation of the eXludus MCOPt Multicore Manager technology. High Performance Computing (HPC) capacity is growing, both in research labs and increasingly in commercial enterprises, and new HPC clusters tend to contain a large number of multicore-based compute nodes, with each node having a large number of processing cores. Multicore processors offer the potential for increased computing capacity but can result in degraded system performance as processing resources may be underutilized by existing applications. This occurs because most applications are not widely parallel and cannot therefore fully use the aggregate processing power available. ***Re-writing applications for multicore is costly and time consuming and when completed may still use only a fraction of the available cores as most problems are not massively parallel by nature.*** In addition, application re-writes only improve the performance of the re-written code, not its performance in relation to other codes that could be running concurrently on the same system.

Another important consideration is that since most individual applications cannot fully exploit multicore capacity, ***users will need to run more concurrent work*** if they are to take advantage of multicore's processing potential. As the number of concurrent tasks increases, the likelihood for application conflicts when accessing shared system resources also increases. These resource conflicts can degrade overall performance and at times lead to application and system instability.

The DICE team investigated the MCOPt software to see how it can help resolve these issues and increase individual system and overall cluster throughput. To mirror the activities of a typical technical computing/HPC environment, a number of technical applications were included in this testing with application processing tasks dispatched across a cluster environment via a traditional workload manager (Torque).

Overall, the testing demonstrated MCOPt is administratively simple to install and use, and requires no application level changes. MCOPt can be used to safely increase system load and prevent resource oversubscriptions from developing. The MCOPt benefits translate into:

- More processing capacity being put to effective use
- Increased individual system – typically 20% or more in multi-user, mixed workload scenarios – and aggregate cluster throughput
- Improved system and application stability in high memory usage scenarios
- Better conversion of consumer energy into productive work

We successfully increased performance by increasing slot-per-core workload ratios for the compute nodes by both 1.5 and 2 times the number of physical CPUs. We suggest careful consideration of increasing this ratio more than 2 times as in-depth knowledge of the applications would be needed.



## 2. Background

In HPC the primary tool for computation is a set of loosely coupled computers that are all connected via a high speed network. These computers, called compute nodes, are usually controlled by a single computer called a head node. The head node is used as the interface to the user to help the user dispatch processing jobs across the HPC cluster. Most of the time, HPC compute nodes run the Linux operating system, which was the case for this technical evaluation. The DICE team executed a workload of many I/O- and compute-intensive jobs which simulated users running these jobs for a lengthy period of time. The same workload was used with the MCOPT software enabled and disabled for comparison to show the improvements available through use of the MCOPT technology. These improvements are measured as a) increasing the number of processing jobs completed within a given time window, and/or b) reductions in average job runtimes.

### 2.1. Offered Capabilities

Table 1 below describes the capabilities of the eXludus MCOPT 3.4 product. The information is available from documentation on the eXludus website ([www.exludus.com](http://www.exludus.com)) and through information provided by eXludus personnel.

Table 1. Capabilities Summary

General	
Name and version of eXludus MCOPT Multicore Manager	MCOPT version 3.4.6. Operating Environment Distribution: Centos 5.5.
General architecture	Scalable, flexible.
Can function in a heterogeneous environment?	No. Supports Linux 2.6.9 and above.
MCOPT features	
Memory swap management	MCOPT monitors system memory pressure (a function of physical and virtual memory used and available and the level of in/out swap activity) and takes proactive steps to limit performance degradation and potential system instability associated with excessive swap activity. MCOPT monitors each application, and if an application requests memory capacity that could lead to intensive swap activity developing, MCOPT will take proactive steps to limit this condition. These steps include slowing down the rate at which a job can build



	memory footprint and temporarily suspending jobs if necessary, subsequently releasing the job when the memory oversubscription problem has been resolved.
Prevent job starvation	Resource contention is limited thus keeping jobs from being blocked by other jobs hogging resources.
Core usage optimization	MCOPt has backfilling and CPU cycle recapture capabilities.
Core 'scavenging' (cores idling due to a wait state are identified and released to reduce unused cycles)	CPU cores are often idle due to access control. MCOPt looks for these idle cores and schedules tasks to be completed thus keeping CPU core utilization high.
Unique job priority assignments and job exclusivity	Kernel level job priority can be assigned allowing higher priority jobs preferential access to resources versus lower priority jobs but still allowing all jobs to complete. An exclusive flag is also available to allow important jobs to immediately take control of all available CPU slots.
Virtual Machine isolation (prevent VMs from impacting other host activities)	This is accomplished by restricting which CPUs MCOPt can control.
Real-time job profiling (separate license required)	MCOPt creates a notion of a "Job" comprised of related processes and threads, and runtime statistics are recorded for CPU time, physical and virtual memory, I/O, and disk file usage. These statistics include, as example, the maximum usage of physical memory and swap. This data can then be analyzed from a job, process and thread perspective, including the ability to plot the use of memory over the life of a job or process. These data points can be used to make intelligent decisions about future job execution.
Core virtualization	By default, the number of virtual CPUs (vcpus) is set to the number of physical CPUs. This value can be set anywhere from 0 to 4X the number of physical CPUs. This variable controls how many tasks can concurrently run on a single core. As an example, using the default setting MCOPt will prevent job time-slicing on an individual



	processor as only one job per core will be launched; if the vcpu is set to 2X the number of physical cores, 2 concurrently executing jobs will be allowed to time-slice on a single core.
Requires kernel modification	No. However, best practice suggests that upgrading the kernel is followed with updated MCOpt binaries.
Requires application modification	No. Execute applications just as before but set MCOpt environment variables or launch the application using <i>mcorerun</i> in order for MCOpt to manage that application.
Prevents shared resource contention	Yes. For example, jobs needing more memory than what is available will be held by MCOpt until the memory becomes available. This is a dynamic function based on real-time system memory usage versus real-time application needs, versus a static memory reservation system.
Supports interactive user mode	This is accomplished using the <i>mcorerun</i> and <i>mcorenice</i> commands.
Requires workload manager	No. Although, MCOpt can work in conjunction with most workload managers. Torque was used in this evaluation. Note: while workload managers control macro-level cluster scheduling (which node to dispatch the next job to), MCOpt controls micro-level (kernel) system resource scheduling.
Does MCOpt work with GPGPUs?	MCOpt will only attempt to improve performance of the code running on the compute node CPUs. It will not attempt to intervene with code running on GPGPUs.
<b>Scalability</b>	
Multi-thread support	Yes
Eliminates cluster dispatch latency	Yes. This is configurable by allowing more jobs to be safely pushed to a compute node where that job can then grab a processing slot immediately once a slot becomes free.
<b>Reliability</b>	
Remote capabilities	Can be managed remotely with secure



	shell.
Is there any mechanism for detecting errors or data corruption?	Only in /var/log/exludus if logged. Logs are available in /var/log/exludus for all compute nodes.
Does the system provide a failover capability for each component?	No
Workload Management Support	Platform LSF Grid Engine PBSpro Torque Loadleveler Runtime Design Automation NC
Systems Management	
Is there a command line interface?	Yes. All functions are controlled through your favorite shell.
Is there a graphic user interface (GUI) provided for administration?	No
What types of reports are available for administrators?	Jobstatus, details currently executing jobs. Jobreport, details historical runtime information. Other extensive reporting is available through the companion JobProfiler product (requires a separate license).
What documentation is provided for the installation and setup?	<ul style="list-style-type: none"><li>• MCOPT and JobProfiler Installation Guide</li><li>• Admin User Guide MCOPT</li><li>• JobProfiler Admin User Guide</li></ul>
Diagnostic and Debugging Tools	
Are there any troubleshooting or debugging tools available?	No. All debugging is accomplished with Linux or workload manager functionality.
How do I know my applications are under MCOPT control?	As each job runs the executable name will be preceded with a marking character. The default marking character is '*'; this can be modified to a site specific character. You can use the top or ps commands to see these markings.
Service and Support	
Support services available	Phone: (514) 227-8411 Web, <a href="http://www.eXludus.com">www.eXludus.com</a> Email, <a href="mailto:support@eXludus.com">support@eXludus.com</a>
How often is software updated and who performs the upgrades?	Targeted semi-annual updates and bug fixes as needed. Upgrades can be accomplished by the site personnel, remotely by eXludus or eXludus support



	personnel can be scheduled onsite.
Training & Professional Services	
Training services available	Training is available onsite or at eXludus headquarters.

### 3. Key Evaluation Criteria

The following evaluation criteria were assessed:

1. The ability to administer MCOPt software, execute jobs and report jobs.
2. The ability for the software to interface with HPC hardware.
3. The security and privacy of other users – permission management.

### 4. Evaluation Setup

For purposes of the evaluation, the HPC compute cluster was configured with the following hardware:

- The head node is a Dell PowerEdge R610 with Dual 2.53 GHz Intel Xeon 5630 Quad-core processors, 16 GB of Memory, Mellanox QDR Infiniband dual port card, Qlogic 8 Gbps fiber channel HBA and Dual 250 GB SATA drives in a RAID 1 configuration with CentOS 5.5 as the operating environment.
- The two I/O servers were Dell PowerEdge R610 with Dual 2.53 GHz Intel Xeon 5630 Quad-core processors, 16 GB of Memory, Mellanox QDR Infiniband dual port card, Qlogic 8 Gbps fiber channel HBA and a single 250 GB SATA drive with CentOS 5.5 as the operating environment.
- The metadata server was a Dell PowerEdge R610 with Dual 2.53 GHz Intel Xeon 5630 Quad-core processors, 16 GB of Memory, Mellanox QDR Infiniband dual port card, Qlogic 8 Gbps fiber channel HBA and a single 250 GB SATA drive with CentOS 5.5 as the operating environment.
- The 16 compute nodes were Dell PowerEdge R610 with Dual 2.53 GHz Intel Xeon 5630 Quad-core processors, 16 GB of Memory, Mellanox QDR Infiniband dual port card and a single 250 GB SATA drive with CentOS 5.5 as the operating environment. (Evaluation was limited to 1 compute node.)

Figure 1 below depicts the solution setup used in the DICE Data Room.

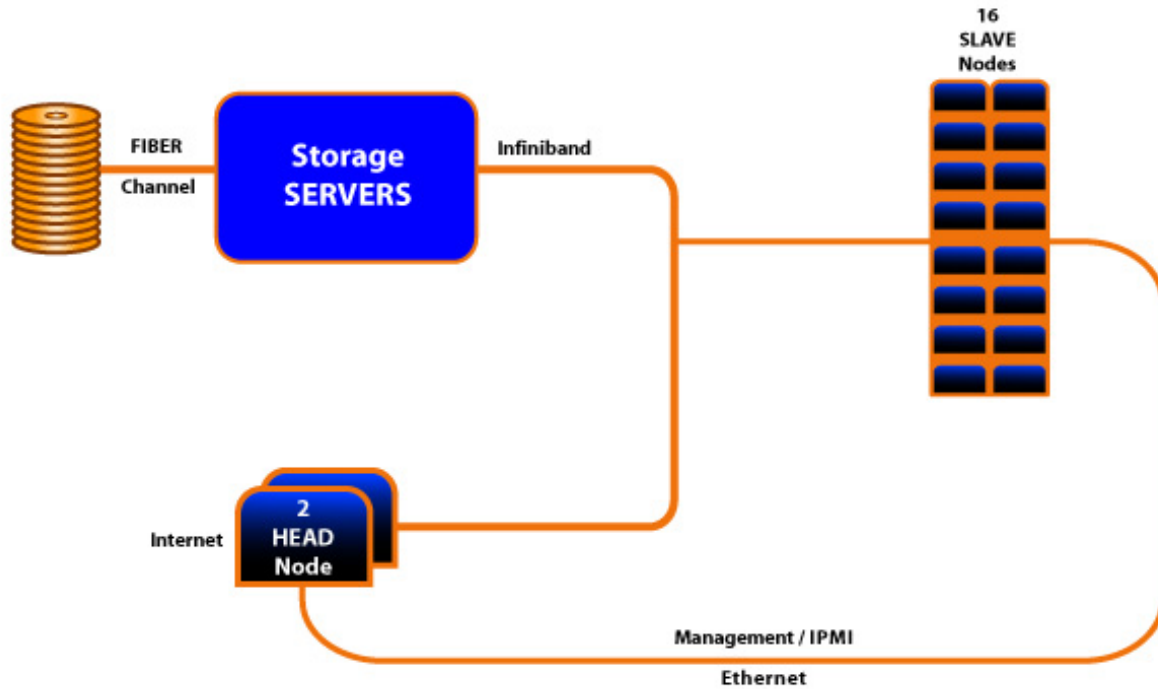


Figure 1

Because the MCOpt software performance enhancement is individual node based, this evaluation only tested workloads using a single compute node at a time in the above mentioned cluster.

## 5. Evaluation Results

The DICE team evaluated the eXludus MCOpt tool against eXludus published product specifications. This evaluation included testing of functionality as documented, performance as documented and installation and configuration. Performance and functionality testing was conducted using a DICE-generated methodology for evaluating cluster management utilities, which can be found in the technical proposal that was submitted to eXludus.

### 5.1. Evaluation Findings

The DICE team's findings are detailed in the following sections.



## 5.1.1. Software Installation and Setup

To obtain correct MCOPt binaries for the target systems, the user will need to provide eXludus with output from the `uname -a` command (unless running a vanilla distribution for which MCOPt binaries are readily available):

```
brutus:/ghome/diceabe> uname -a  
Linux brutus 2.6.18-194.32.1.el5 #1 SMP Wed Jan 5 17:52:25 EST 2011 x86_64x86_6  
4 x86_64 GNU/Linux
```

eXludus will then provide the installation binaries for the kernel version. In this evaluation the head node and compute nodes all run the same kernel version. If the user has several different kernel versions running on their nodes, they will need separate binaries for each variant of the kernel they are running.

eXludus will provide the following RPM files to install:

- exludus-mcopt-bin-MCOPtVersion.KernelVersion-LinuxOS.ARCH – eXludus MCOPt scripts and binaries
- exludus-mcopt-drv-MCOPtVersion.KernelVersion-LinuxOS.ARCH – eXludus MCOPt kernel drivers
- exludus-gmcoptdb-MCOPtVersion.KernelVersion-LinuxOS.ARCH – eXludus JobProfiler (Licensed Separately)

The eXludus MCOPt software was installed in a shared file system partition on the DICE cluster. The DICE cluster runs Bright Cluster Manager, and its shared file system partition is located in `/cm/shared/apps`.

The path for `mcopt` was `/cm/shared/apps/eXludus/mcopt` and the path for `gmcoptdb` is `/cm/shared/apps/eXludus/gmcoptdb`.

```
headnode% rpm -ivh --prefix /cm/shared/apps eXludus-mcopt-bin-3.4.6-2.6.18-194-  
32.1-CentOS5.4.x86_64.rpm  
headnode% rpm -ivh --prefix /cm/shared/apps eXludus-mcopt-drv-3.4.6-2.6.18-194-  
32.1-CentOS5.4.x86_64.rpm  
headnode% rpm -ivh --prefix /cm/shared/apps eXludus-gmcoptdb-3.4.6-2.6.18-194-  
32.1-CentOS5.4.x86_64.rpm
```

With the software now installed on the head node, the next step is to set up the appropriate links on the cluster nodes so they can see the software on the shared file system.

A script provided by eXludus is used to set up the appropriate links for cluster nodes using a shared file system install. That script is `<InstallDir>/exludus/mcopt/sbin/exludus-svctl`.



```
clusternode% exludus-svctl -p /cm/shared/apps
```

This script is used to get the MCOpt services to run on boot up for the cluster nodes:

```
clusternode% chkconfig -add exludus-mcopt
clusternode% chkconfig -add exludus-gmcoptdb
clusternode% chkconfig exludus-mcopt on
clusternode% chkconfig exludus-gmcoptdb on
```

Now install the license file from eXludus. Once the license file is obtained, it can be placed in the /etc/sysconfig/exludus directory or in the shared file system directory <InstallDir>/exludus/mcopt/etc.

Please refer to the eXludus MCOpt and JobProfiler Installation Guide 3.4 for more discussion on installation.

No problems occurred during the installation portion of this evaluation.

### ***5.1.2. Configuration Considerations for the MCOpt and JobProfiler Setup***

For both MCOpt and JobProfiler, an appropriate configuration file is used to control the setup characteristics for how the software will function. For MCOpt, that file is mcopt.conf and by default is located in /etc. For a shared file system, links are created to the shared file system location.

These are the parameters DICE used for mcopt.conf for the evaluation as well as the default settings:

```
# MCOpt by eXludus Technologies Inc. 2011
# MCOpt configuration file.
#
# Modify options according to your system setup.
# Refer to the MCOpt Admin and User Guide for additional information
#
#
# WLM_PATH is the path to the "qsub/bsub" command of the workload manager
# (bin directory).
#
# Value: path, string, default: not set
#
#WLM_PATH=<path>
#
```



If the user is using a workload manager, the following variable, WLM\_PATH, needs to be set for the path to the qsub/bsub commands. The following value is set to the test platform's directory for torque binaries.

```
WLM_PATH=/cm/shared/apps/torque/2.4.8/bin
#
```

```
# MCOpt installation path
#
# Full path to the MCOpt installation directory.
#
# Value: path string, default: /usr/local/exludus/mcopt/
#
```

This should be set up automatically from installing the rpm and using the `-prefix` option.

```
MCOPT_PATH=/cm/shared/apps/exludus/mcopt
```

```
#
# MCOpt log directory
#
# Full path to MCOpt log directory
#
# Value: path string, default: /var/log/exludus
#
MCOPT_LOG_DIR=/var/log/exludus
```

```
#
# File profiling On|Off
#
# Switch file profiling feature "On|Off", default: Off
#
# Value: option string - "On|Off"
#
FILE_PROFILING=On
```

```
#
# File profiling database path
#
# Full path to the MCOpt file profiling database
#
# Value: path string, default: /var/log/exludus/mcoptdb_vf
#
FILE_PROFILING_DATABASE_PATH=/var/log/exludus/mcoptdb_vf
```

```
#
```



```
# Profiling On|Off
#
# Switch profiling feature "On|Off", default: On
#
# Value: option string - "On|Off"
#
PROFILING=On

# Profiling database path
#
# Full path to the MCOPT profiling database
#
# Value: path string, default: /var/log/exludus/mcoptdb
#
PROFILING_DATABASE_PATH=/var/log/exludus/mcoptdb

#
# Database daemons log file
#
# Full path to MCOPT database daemons log file
#
# Value: path string , default: /var/log/exludus/mcoptdb.log
#
DATABASE_DAEMONS_LOG_PATH=${MCOPT_LOG_DIR}/mcoptdb.log

#
# Time to wait for the boot process to complete.
#
# Value: option string - non-negative integer in seconds, default: not set
#
#BOOT_TIME_TO_WAIT=6

#
# Service daemon options
#
# Value: option string, default SCHEDOPT_OPTS="-C <path_to_config_file> -L
# <path_to_log_file>"
#
SERVICE_DAEMON_OPTIONS="-C ${MCOPT_PATH}/etc/mcopt.conf -L
${MCOPT_LOG_DIR}/mcopt_$(hostname --fqdn).log"

#
# Global profiling uri
#
# URI adress of centralized profiling service
#
```



# Value: string, default: localhost:9011
#
GLOBAL\_PROFILING\_URI=headnode.cm.cluster:9011

When using the global profiling option, all data from each compute node is temporarily stored locally on that node. The excludus-gmcoptdb service moves that data to the centralized profiling server once the job has been completed.

The above settings are only read when the MCOPT daemons are started. So any change to them will require the excludus-mcopt service to be re-started.

The following kernel driver options are also read when the MCOPT daemons are started but can be changed on the fly using the /sbin/sysctl command. However, changing these parameters on the fly will require the device or resource to be free. If a job is running under MCOPT control then it is likely that the user will be unable to change a setting during the job execution.

For example:

To read the current setting for scanning rate:

```
headnode% ssh clusternode /sbin/sysctl -a | grep dev.exmcore.pfrate
```

To change the current setting for scanning rate:

```
headnode% ssh clusternode /sbin/sysctl -w dev.exmcore.pfrate=0
```

To save all currently running kernel driver settings to be permanent on startup:

```
headnode% ssh clusternode service excludus-mcopt update
```

(Section 5.1.3 will further discuss the use of the excludus-mcopt service.)

```
#
#####
#
# Kernel driver options
# Change to kernel driver options will not take effect after restarting
# the service.
##
#
# Enable CPU affinity
# Value= on|off, default= off
# dev.cpu.affinity=off
dev.exmcore.cpu.affinity=off
```



If the user wants their jobs to run on cores located on the same physical CPU then turn CPU affinity on.

```
#
# Enable pre-emptive backfilling
#   Value= on|off, default= on
#dev.exmcore.backfilling = on
dev.exmcore.backfilling=on
```

If the user wants to allow jobs to backfill when a currently running job has stalled on the availability of a resource, turn backfilling on.

```
#
# Time to wait in queue before job gets expired
#   Value= non-negative integer in minutes, default= 0 (no expiration)
#dev.exmcore.qttl = 0
dev.exmcore.qttl=0
#
# Restrict physical CPUs to run jobs with
#   Value= all|[CPU list string], default= all.
#   CPU list can be written as comma (',') separated, or with '-' to
#   specify a range.
#dev.exmcore.cpu.selection = all
#dev.exmcore.cpu.selection = 0,1,3
#dev.exmcore.cpu.selection = 1-5
dev.exmcore.cpu.selection=all
```

If the user has virtualized a server and does not want to interrupt that virtual server with MCOPT, set the CPU selection variable.

```
#
# Number of virtual CPUs
#   Value= [1:4 * number of physical CPUs detected],
#   default= number of physical CPUs detected
#dev.exmcore.cpu.virtual = 4
dev.exmcore.cpu.virtual=32
```

```
#
# Size of queue for each CPU
#   Value= 1:50 default= 50
#dev.exmcore.cpu.qdepth = 50
dev.exmcore.cpu.qdepth=50
```

```
#
# Memory swapping prevention
#   Value= on|off, default= on
```



```
#dev.exmcore.memory.sandbox = on  
dev.exmcore.memory.sandbox = on
```

This sandbox variable is no longer in use.

```
#  
# Process name marking character, prefix to process name.  
#   Value: off|[marking character], default: '*'  
#dev.exmcore.mchar = off  
#dev.exmcore.mchar = *  
#dev.exmcore.mchar = +  
dev.exmcore.mchar=*  
  
#  
# Abort on license Invalid is used to give the opportunity to the admin  
# to decide, when the license becomes invalid, if the job should be rejected  
# with error message or  
# if the job should run but without MCOPt control still with the error message.  
#   Value= yes|no, default= yes  
#dev.exmcore.abort_license_invalid = yes  
dev.exmcore.abort_license_invalid=yes  
  
#  
# VM Prediction enables MCOpt to monitor the growth of a job's virtual memory  
# consumption and predict how much virtual memory a job is going to consume  
# following the current trend. Having VM prediction turned on may help the memory  
# management handling jobs that grow memory in lock-step.  
#   Value= off|on, default= off  
#dev.exmcore.memory.vm_prediction = off  
dev.exmcore.memory.vm_prediction=on  
  
#  
# Hard preemption: e.g. a new job will always preempt a running job of lower  
#priority, be it a primary or backfiller.  
#   Value= on|off, default= on  
# dev.exmcore.hard_preemption= on  
dev.exmcore.hard_preemption=on  
  
#  
# Swap slot: e.g. number of jobs that are allowed to run simultaneity when the  
# swap-in starts.  
#   Value= 1:31, default= 1  
# dev.exmcore.memory.swap_slot= 1  
dev.exmcore.memory.swap_slot=1  
  
#
```



```
# Maximal rank e.g. priority value MCOPT will support
#   Value= 3:16, default= 16
# dev.exmcore.cpu.sched_max_priority=16
dev.exmcore.cpu.sched_max_priority=16

#
# Schedule algorithm switch
#   Value= on|off, default= on
# dev.exmcore.schedule=on
dev.exmcore.schedule=on

#
# Default priority level for new jobs
#   Value= 2:(dev.exmcore.cpu.sched_max_priority-2), default = 10
# dev.exmcore.cpu.sched_dft_priority=10
dev.exmcore.cpu.sched_dft_priority=10

#
# Scanning frequency in seconds used to record job activities.
#   Value=0:~, default =0
# dev.exmcore.pfrate=0
dev.exmcore.pfrate=300
```

This dev.exmcore.pfrate parameter has a dual purpose. This setting turns on the profiling using pfrate > 0, but the value chosen by the user also determines how often data records in seconds. Careful consideration should be given on the value chosen here. The smaller the value, the larger the data will grow. A large database size will slow down the reporting tool execution.

Please refer to the eXludus Admin User Guide MCOPT Ver 3.4 for more discussion on the use of mcopt.conf.

### **5.1.3. GLOBAL Database Configuration**

When using the Global database for the JobProfiler, the gmcoptdb.conf file is used for the required variable settings. Below is the configuration file used for the evaluation as well as the default settings:

```
#
# exludus-gmcoptdb configuration file
#
#
# MCOPT_PATH=/usr/local/exludus/gmcoptdb
#
MCOPT_PATH=/cm/shared/apps/exludus/gmcoptdb
```



```
#
# Path to the MCOpt directory.
# Value: string, default: /usr/local/exludus/mcopt/
#
# MCOPTDB_PATH=/var/log/exludus/gmcoptdb
#
MCOPTDB_PATH=/var/log/exludus/gmcoptdb

#
# Path to the MCOpt profiling database
# Value: string, default: /var/log/exludus/gmcoptdb
#
# MCOPTDB_VF_PATH=/var/log/exludus/gmcoptdb_vf
#
MCOPTDB_VF_PATH=/var/log/exludus/gmcoptdb_vf

#
# Path to the MCOpt profiling database vf
# Value: string, default: /var/log/exludus/gmcoptdb_vf
#
#
# MCOPTDB_URI=hostname:port
#
MCOPTDB_URI=headnode.cm.cluster:9011

# Full URI adress of centralized database
# Value: string, default: localhost:9011
#
```

Please refer to the eXludus JobProf Admin User Guide ver 3.4 for more discussion on gmcoptdb.conf.

No problems occurred during the configuration portion of MCOpt or JobProfiler.

### 5.1.3.1. Modules Environment

The DICE cluster uses modules for dynamic modification of the user's environment via modulefiles. The following modulefile was created to include MCOpt into our environment via the command

```
headnode# module load exludus/MCOpt/3.4

#%Module *- tcl *-
##
## dot modulefile
##
```



```
proc ModulesHelp { } {  
    puts stderr "\tAdds eXludus MCOPT and App Profiler to your environment variables."  
}
```

module-whatism "adds eXludus MCOPT and App Profiler to your environment variables"

```
set      root      /cm/shared/apps/exludus/mcopt  
set      groot     /cm/shared/apps/exludus/gmcoptdb
```

```
append-path  PATH      $groot/bin:$root/bin  
append-path  LD_LIBRARY_PATH  $groot/lib/  
append-path  MANPATH  
/cm/shared/apps/exludus/gmcoptdb/man:/cm/shared/apps/exludus/mcopt/man
```

For more on the modules environment go to <http://modules.sourceforge.net>.

### 5.1.4. Management and Operation

To control the operation of MCOPT and the JobProfiler, two scripts are setup as services. The MCOPT script or service is exludus-mcopt and the JobProfiling script or service is exludus-gmcoptdb. Both of these scripts are run at startup and reside in the /etc/init.d directory.

For exludus-mcopt, there are the following arguments:

```
help:          shows help on correct usage  
start :        starts the daemon  
start_mcoptdb: starts the mcoptdbm daemon only  
stop:          stops the daemon  
status:        status on the running daemons  
stop_mcoptdb: stops the mcoptdbm daemon only  
backup_db:     creates a new backup of database in /var/log/exludus.  
Backup files will have a .bk extension (mcoptdb.bk and  
mcoptdb_vf.bk). Backup file is also in ASCII text format.  
stopforce:    forcibly stops the daemon  
restart:      stops then re-starts the daemon  
update:       updates the /etc/mcopt.conf configuration with the current  
active  
parameters, the updated parameters will be used for future  
restarts  
lateload:     updates the mcopt.conf configuration file with SND_CHK=6  
(service boot timeout)  
sharpload:    removes the SND_CHK configuration variable from  
/etc/mcopt.conf
```



For example:

To capture the current MCOpt parameters being used and save them to `mcopt.conf` (be certain to create a backup file beforehand):

```
headnode% service mcopt update
```

To stop the `mcopt` service on a node:

```
headnode% ssh clusternode service mcopt stop
```

For `exludus-gmcoptdb`, there are the following arguments:

```
help: shows help on correct usage
start: starts the daemon
stop: stops the daemon
status: status on the running daemons
restart: stops then re-starts the daemon
```

For example:

To re-start the JobProfiler on the head node:

```
headnode% service gmcoptdb restart
```

To stop the `gmcoptdb` service on a node:

```
headnode% ssh clusternode service gmcoptdb stop
```

### 5.1.4.1. Database backup and rotation

As mentioned briefly above, the databases can be backed up using the `service exludus-mcopt backup_db` command. MCOpt automatically installs a `logrotate` configuration in `/etc/logrotate.d`,

which both rotates the logs and backs up the database. The backup will need to be initialized for automatic backup sequence by running the command `service exludus-mcopt backup_db`.

### 5.1.5. Job Control under MCOpt

When using a workload manager, two environment variables must be set for jobs to run under MCOpt control: `EXL_MCOPT_SCHEDULE` and `EXL_MCOPT_JOBNAME`. The



second of these variables does not have to be set explicitly. It is set by MCOPT looking at the job name the workload manager has assigned. Here is an example using PBS:

```
export EXL_MCOPT_SCHEDULE=yes
qsub -V file.pbs
```

Here is a list of additional environment variables that could be used to manage the job execution:

**EXL\_MCOPT\_AFFINITY=yes:** keep job on same cpu.  
**EXL\_MCOPT\_BKONLY=yes:** run job as a backfill job only.  
**EXL\_MCOPT\_PRIONLY=yes:** run job as a primary job only.  
**EXL\_MCOPT\_RANK=number:** give job a desired job priority.

Another method to launch jobs is using the mcorerun command provided with the MCOPT software. The mcorerun program executes interactively under MCOPT control by setting the two environment variables above and exec'ing its argument:

```
mcorerun name_of_executable
```

The mcorerun command must be used when there is no workload manager available. The mcorerun command also has many options similar to the environment variables used for a workload manager. For further documentation on these options for launching jobs under MCOPT, see section 3 of the Admin User Guide MCOPT.

### 5.1.5.1. Job Reporting with JobProfiler

A JobProfiler is provided which retrieves statistical data about jobs that were run under MCOPT control. This data is stored in a central database which was installed in /var/log/exludus. This location is configurable via the *PROFILING\_DATABASE\_PATH* variable in /etc/mcopt.conf. The gmcoptdb service is started via the startup script exludus-gmcoptdb. To start the service the script was executed as

```
/etc/init.d/exludus-gmcoptdb start
```

With the service started on a given node, statistics are compiled during the execution of any MCOPT-enabled application on that node and stored in a central database. These statistics can be retrieved and analyzed using the *jobreport* utility. There are many options available for the jobreport command. For example, analysis of a job with identification number 12345 is obtained via

```
jobreport -J 12345.brutus.cm.cluster
```

The output obtained using jobreport this way represents per-*job* totals rather than per-*process* totals. This explains what might otherwise be somewhat surprising data. For example, the output from a 16-core job might contain (in addition to other output):



Total #proc:	361
Total #thread:	37
Maximum #proc:	19
Maximum #thread:	32

We have found that even though jobreport collects a substantial amount of information, the information can be cumbersome to sort and interpret. The newer version on MCOPT due out in the near future will have improvements to alleviate those reporting issues.

Processes under MCOPT control are prefixed with a character in the output of commands such as “top” and “ps.” The character used is user-definable and defaults to an asterisk:

`dev.exmcore.mchar=*` (as describe above in section 5.1.1)

This way, MCOPT-controlled processes are clearly distinguishable from non-MCOPT processes.

Several small inconsistencies were discovered regarding this process-adornment feature. For example, the asterisk is seen when using “ps -u <username>” but not when using “ps -ef.” Additionally, behavior for parallel programs varied depending on the parallel program launcher used. On the test system it was observed that process adornment occurs for programs launched by the mpirun of the OpenMPI package version 1.4.3. However for programs launched by the mpirun of the mvapich package version 1.1, the process adornment does not occur.

### 5.1.6. Testing Results

The idea behind MCOPT is to safely enable higher system loading levels by dynamically preventing resource oversubscription (to cores and memory) problems that might otherwise occur. This reduces idle compute capacity and leads to higher overall system utilization levels. *By safely increasing system utilization levels, job throughput is increased.*

Jobs are dispatched from a workload manager to a compute node, and MCOPT then continuously analyzes application resource requests and system state. It also applies underlying queuing theory algorithms to determine “best fit” of applications to resources. MCOPT does not replace the Linux scheduler but controls which requests are presented to the scheduler. If no core is available for the dispatched job to run on, MCOPT will hold the job in its queue until a core becomes available. An MCOPT goal is to prevent processor time-slice behavior which increases average job runtimes. After a job begins execution, if the job then begins an I/O wait state, MCOPT will place that job back in its queue and schedule another job (a “backfiller”) to that CPU. A potential problem is that most HPC systems use their own queuing system based on the number of slots or virtual CPUs that each cluster node physically can operate. The MCOPT backfilling



behavior could cause a roadblock to HPC cluster throughput if jobs which could possibly run are sitting suspended in the MCOPt queue. However, MCOPt parameters can be adjusted to prevent or modify this backfilling behavior.

For MCOPt to maximally increase throughput on the compute nodes, the workload manager needs to dispatch more work to the nodes. By pushing more work to the node, MCOPt can then put as much resource to use as possible while preventing resource overload conditions. While this workload increase is somewhat dependent on application characteristics (i.e. long-running, large memory workloads need little increase; very short running, CPU intensive workloads may benefit from larger increases), the eXludus “rule-of-thumb” is to increase workloads by a factor of 1.25 to 1.5x the ratio used when running a standard Linux distribution. In this evaluation we tested varying levels of increased workload ratios ranging from 1.5x, 2x, to a very high 4x level.

The DICE team installed the following two sets of applications for evaluation of the eXludus MCOPt software. The first set used 2x and 4x workload ratios for applications that do a high amount of disk I/O along with the high CPU bound codes that could take advantage of extra CPU cycles.

The first set of applications consisted of the following:

- MPIBlast 1.60 compile with GCC 4.1.2 and MVAPICH 1.1
- Dakota 5.1 compiled with PGI compilers and OpenMPI 1.4.3
- Bonnie++ 1.96
- Flash 3.3 compiled with GCC 4.3.4 and OpenMPI 1.4.3
- OpenFOAM 1.7.1 with GCC 4.3.4 and OpenMPI 1.4.3
- IOR 2.10.2 compiled with Open64 and MVAPICH 1.1

The second set of applications used to obtain a 1.5x workload ratio were bioinformatics codes. These codes can create a high rate of memory consumption resulting in memory pressure situation for which MCOPt is designed to handle. All were compiled with GCC 4.3.4. They consisted of the following:

- BEAST (Version 1.5.4) is a cross-platform program for Bayesian MCMC analysis of molecular sequences.
- Bowtie (Version 0.12.5) maps short sequence reads to a reference genome sequence. Bowtie can use pthreads but for this benchmark it was run as a serial job.
- Muscle (Version 3.8.31) is a multiple sequence alignment program. It is a serial program that depending on the input sequences can require less than 1Gb RAM or over 16Gb.
- Velvet (Version 1.0.12) is a serial short read assembler for Next Gen Sequencer data. It can require a large amount of memory ( $\geq 12GB$ ).



For the 2x and 4x workload ratio tests this evaluation compared the wall clock time consumed for a workload of jobs with MCOpt enabled and disabled (i.e., native Linux). For each test case we ran a workload of 11 jobs consisting of the applications mentioned above. Each application was run twice except for Bonnie++ which was executed only once. We also attempted to submit the jobs to the workload manager in such a way that the I/O bound and compute jobs ran at the same time. The tests were also run on 2 separate nodes. One node with MCOpt scheduling enabled and a second node with MCOpt scheduling disabled. This allowed us to run a test case scenario comparing Linux vs. MCOpt at the same time. However, we also ran the test at separate times because the nodes are sharing the same Lustre file system, and we wanted to be sure that the shared file system was not causing any I/O bottlenecks.

Table 2. Results for 2x Workload Ratio

	MCOpt disabled/Native Linux control	MCOpt enabled scheduling	Concurrent running on 2 nodes	Improvement in wall time
Test case 1	5 Hrs 48 Min	4 Hrs 57 Min	Yes	51 Minutes
Test case 2	6 Hrs 29 Min	4 Hrs 48 Min	Yes	1 Hr 41 Min
Test case 3	6 Hrs 22 Min	5 Hrs 10 Min	Yes	1 Hr 12 Min
Test case 4	6 Hrs 55 Min	5 Hrs 10 Min	No	1 Hr 45 Min
Test case 5	6 Hrs 15 Min	5 Hrs 11 Min	No	1 Hr 4 Min

**The MCOpt enabled runs produced a 26% time savings in completing the workload.**

We did some testing with a 4x workload ratio, thus increasing the slots from 16 to 32. However, we would not recommend doing this unless you are very knowledgeable of the applications, as this caused some of the compute intensive applications to take as much as 8-10 times longer to run. While not generally recommended, it is worth noting that the resulting extreme level of resource oversubscription caused the native Linux runs to fail, whereas the MCOpt enabled runs did complete. This test did demonstrate MCOpt's ability to gracefully manage even extreme levels of system resource overload.

**5.1.6.1. Results for the 1.5x Workload Ratio**

For the 1.5x ratio testing, the idea was to test a situation where the free memory of the compute node would become very small causing high memory pressure for the jobs running on the node. This scenario has been observed using the bioinformatics codes at other installation sites. We had great difficulty in getting these codes to create memory pressure with 16GB of memory installed, so we cut the memory down to 8 GB. Still getting the needed memory pressure was difficult. The timing of the code execution was very close, with Linux edging out better results than MCOpt. That result would be



expected because MCOPt did not identify memory pressure problems to intervene with and MCOPt will be neutral to performance when no scheduling opportunities or resource conflicts exist (ie, MCOPt has no work to perform so the results will be predicted by base Linux behaviors).

Given that information, we added the Dakota code from the previous tests to see if that would lead to out-of-memory (OOM) errors and/or applications problems. The Dakota application could easily require 5 to 8GB of memory to execute and thus should create the much needed memory pressure. Indeed we saw a big difference in the time needed to run Dakota with the bioinformatics codes. We ran the Dakota code using 4 slots and the bioinformatics codes in the remaining 8 slots. We continually ran a total of 71 bioinformatics jobs (each bioinformatics execution needed just one slot) with Dakota running twice as the first and last jobs in the queue. As a result the first Dakota job had much more memory contention compared to the last and thus took much longer to complete.

For the Native Linux system, the Dakota jobs took much longer wall time to execute. The first Dakota job used a larger percentage of wall time consumed because all 12 slots available were being used. The last Dakota job consumed many times less wall time because the bioinformatics codes had completed execution.

Table 3. Wall Time totals of Both Dakota Runs

	MCOPt Enable	MCOPt disabled/Native Linux control	Performance increase
Run 1	14831 Seconds	53557 Seconds	3.61x
Run 2	13478 Seconds	37646 Seconds	2.79x
Run 3	16390 Seconds	53848 Seconds	3.28x

In addition we did a second memory pressure test using the bioinformatics codes with the eXludus memory validation test. In this case the Native Linux compute node could not complete the job and the compute node locked up. While the MCOPt-enabled compute node completed execution of all the jobs.

These results show that using MCOPt will enable a compute node to perform better under extreme memory conditions. This type of memory condition is very rarely monitored and hard to track for any size cluster. MCOPt will allow bad situations to become manageable rather than find out many hours later that a job is still running or that compute nodes crashed without your knowing why. Using MCOPt in this way would allow unforeseen changes in applications as the data or application changes over time. It could prevent failures for important computational cycles such as seasonal or end of month cycles.

Additional testing of MCOPt was performed with several validation tools provided by eXludus.



Table 4. Backfilling of Jobs with 2x workload ration increase

Number of Jobs Run	Linux time to completion	MCOPt time to completion	%Benefit
32	7993 Seconds	6940 Seconds	15%
48	11972 Seconds	9904 Seconds	21%
64	13979 Seconds	12390 Seconds	13%
80	17952 Seconds	14905 Seconds	20%

Table 5. Memory Pressure Tests with 2x workload ration increase 8 CPU with 8 GB Memory

	Linux Average Execution Time	MCOPt Average Execution Time	% Benefit
12 Jobs 8 Concurrent 15 Processes	296.5	256.625	15%
12 Jobs 8 Concurrent 20 Processes	275.25	209.5	31%
12 Jobs 8 Concurrent 25 Processes	228.375	220.125	3.7%
8 Jobs 8 Concurrent 15 Processes	170.71	147.14	16%
8 Jobs 8 Concurrent 20 Processes	266.25	241.875	10%
8 Jobs 8 Concurrent 25 Processes	164.09	156.36	5%

Table 6. Vector Jobs using 2x workload ratio increase

CPU Utilization	Number of Jobs Submitted	Linux Time (Minutes)	MCOPt Time (Minutes)	% Gain
60%	60	23.58 Min	17.21 Min	37.02%



70%	60	27.31 Min	19.42 Min	40.66%
75%	60	29.11 Min	21.79 Min	33.59%
80%	60	26.49 Min	22.87 Min	15.86%
60%	50	24.00 Min	16.14 Min	48.73%
70%	50	27.17 Min	16.96Min	60.27%
75%	50	28.67 Min	19.11 Min	50.04%
80%	50	25.30 Min	19.54 Min	29.47%
60%	40	17.63 Min	13.70 Min	28.68%
70%	40	19.86 Min	15.39 Min	29.07%
75%	40	20.82 Min	16.18 Min	28.66%
80%	40	18.34 Min	16.53 Min	10.90%

**5.1.7. Functional Testing**

Table 7 below describes the test requirement, evaluation results and evaluation ranking for the critical and high requirements defined by the eXludus MCOpt evaluation team.

The following criteria are used for the evaluation ranking:

- Met                The solution offered the minimum functionality.
- Surpass         The solution offered more than expected functionality.
- Partially Met    The functionality is partially available but is incomplete.
- Missed           The solution offered less than minimum functionality.
- N/A                The requirement for this solution is not applicable.

Table 7. Test Requirements, Results and Ranking

Requirement	Evaluation Results	Evaluation Ranking
<b>Functionality</b>		
<b>Installation and Configuration Requirements</b>		
Software should be easy to install on an HPC compute cluster.	Software was installed on a shared file system for a HPC compute cluster running Bright Cluster Management.	Met
<b>Administrator Interface Tool</b>		
The tool should have the ability to administer an HPC cluster from a head node or single point of access.	MCOpt administration is a collection of shell scripts that can be run from the head node via ssh or pdsh for administration across a cluster.	Met
The tool should allow the administrator to start, stop, obtain status, and restart/reconfigure (re-read	The eXludus-mcopt and eXludus-gmcoptdb services are used for this: Ex: node% service eXludus-	Met



configuration changes) for the software.	mcopt status node% ssh compute-node1 service eXludus-gmcoptdb stop	
The tool should allow the administrator to read active configuration and save it as a static configuration.	This is accomplish by doing: node% service eXludus-mcopt update It is a good idea to backup up the old mcopt.conf file beforehand.	Met
The tool should allow the administrator to customize the software configuration.	Customization is accomplished by editing the mcopt.conf file. The MCOPt service will then need to be restarted.	Met
<b>Software Interfaces</b>		
The tool should have the ability to work with several Workload Managers.	By defining WLM_PATH in the mcopt.conf this will allow MCOPt determine the executable path for qsub/bsub commands.	Met
<b>Database</b>		
The database should provide clustering in order to improve performance.	The databases are stored in SQLite3 database files and do not have clustering capability.	Missed
The database must provide the ability to export or access the data through a third party application.	This can be accomplished with the use of sqlite3 command.	Met
<b>Operational</b>		
The software should support many operating environments (Red Hat, SUSE, CentOS, etc.).	The study was performed on CentOS 5.5. Any Linux running kernel version 2.6 is supported.	Met
The user must have the ability to execute jobs with or without MCOPt control.	Execution of jobs with MCOPt control is achieved by setting EXL_MCOPT_SCHEDULE=yes, then submitting your batch script.	Met
The user should have the ability to execute only a portion of a job under MCOPt control.	You can choose to run commands in the batch script as follows to have them run under MCOPt control: mcorerun flash.ex -parameters  The mcorerun command is also used for interactive mode.	Met
The user should have the ability to change the ranking	Setting the variable EXL_MCOPT_RANK=<1-16>	Met



of a job under MCOPT control.	with 1 being the lowest rank and 16 being the highest rank. Default is 10.  Command line: mcorerun -R.	
The user should have the ability to run a job as a primary or backfilled job.	All jobs are submitted as primary by default. Variables are as follows for batch jobs: EXL_MCOPT_PRIONLY=yes for primary job or mcorerun -P EXL_MCOPT_BKFONLY=yes for backfill job or mcorerun -B.	Met
The user should have the ability for a new job to preempt a lower priority running job.	Preemption is on by default and configurable in mcopt.conf using the dev.exmcore.hard_preemption variable.	Met
The user should have the ability to make use of all available CPU slots of the cluster.	This is a function of the workload manager and not MCOPT.	N/A
The tool should have the ability to store and report job properties after job completion.	Running job profiling will record this data in an SQLite3 database. If using the global database be aware that a large amount of data is stored for file profiling if enabled.	Met



The user should have the ability to obtain reports of running jobs in real time.	This is available through the jobreport command using jobreport -R<interval> -J <jobid>. However running jobreport from the head node on a job running on a compute node with any interval, only gives the output at a single time not at a repeated interval.	Partially Met
<b>Security and Privacy</b>		
The MCOPt administrator (if not system administrator) must be able to perform functions without having system-level root privileges.	Can be accomplished through implementing Linux sudo.	Met
The system must be able to detect data corruption and react either automatically or report to the site system administrator.	Data corruption, should it occur, will be logged into the /var/log/messages or /var/log/eXludus/mcopt.log. If you are running profiling then it may be detected in /var/log/eXludus/lighttpd/error.log.	Met
The system must provide the ability that all operations are subject to access permissions, authorizations of target objects and user privileges on accounts for all of the system involved in any operation.	MCOPt use is open to all users who have accounts on a cluster. No access permissions are implemented or needed at this level.	Met
Assurance that the software is properly using and protecting privileged actions and credentials is required.	A user can see another user's jobs running through the workload manager but cannot modify those jobs. A user also cannot query the job profiling info for another user.	Met
<b>Audit Trail</b>		
The system must keep log files for the MCOPt instance running on a node.	Each server has its own /var/log/eXludus/mcopt.log file for logging errors.	Met
The system must keep an audit trail of administrator transactions.	If the configuration files for MCOPt need to be tracked with an audit trail, then the built-in Linux audit tools will need to be used.	Met



Performance		
The system should improve HPC cluster throughput.	MCOPt successfully increase throughput for 1.5x and 2x oversubscription of compute nodes. See section 5.1.5 for complete details.	Met
Error Handling		
Success or error results from all operations, including those on remote systems, must be available to the administrator at the conclusion of the operation either through the functionality of the tool or some other work around (stored in log files, etc.).	Failures from the MCOPt software are logged in /var/log/eXludus/mcoptdb.log file or the /var/log/eXludus/mcopt_hostname.log file	Met

## 6. Summary

eXludus MCOPt software has shown that it can increase workload throughput by safely allowing more concurrent processing tasks per compute node, preventing resource oversubscriptions that degrade performance and stability and scavenging idle CPU cycles spinning on I/O. MCOPt is easy to install and maintain and works transparently with all existing applications without any modifications. This is important given the large number of applications that exist today. The only caution we have is that it helps for the user to understand the applications they are running in terms of I/O and memory use. MCOPt works best with a workload that has applications with high memory usage or I/O in order to take advantage of spare CPU cycles that can be used for the computational code. The amount of workload increase used in order to increase throughput also needs to be considered. Extreme levels of oversubscription can lead to runtime increases, but MCOPt did demonstrate that it will prevent application and system instability even in the extreme cases.

Core counts are increasing and organizations need to get full value from their compute infrastructure. It is important that multicore processing capacity be used effectively in order to improve throughput (i.e., more work processed in less time), but this can be difficult because most applications have not been designed for multicore. MCOPt demonstrated that it can provide an automated means to improve system utilization and throughput for unchanged applications. MCOPt is worth considering both for new system deployments and as a means to forego forklift upgrades by extending the useful life of existing assets.

As with all technologies today, innovation and improvement are constant, and it is no exception with MCOPt. We have learned that the next version of MCOPt will



incorporate what is called “core & memory containers.” This is a lightweight virtualization with significant contrast from traditional virtual machine approaches or Linux group functionality. Containers can be dynamically configured on a per-project (user/application/job) basis in support of Service Level Agreements (SLAs) with that resource level guaranteed to be available. Individual projects that exceed their allocations may be penalized (such as being forced to use the swap device instead of physical memory), but this resource overuse will not negatively impact others projects that are consuming at or below their committed allocation. This prevents situations such as when one application spawns large numbers of threads, effectively taking over a large percentage of the system, or when one application allocates all shared memory thus creating memory starvation or excessive swapping for other applications. However, if assigned projects use less than their allocation, other projects are automatically allowed to make use of any available resource. For example, a single application running by itself will be allowed to allocate as much memory as it wishes. If core or memory resource contention develops, all projects are then forced back into their SLA assigned containers.

With these new technology innovations an initial reaction may be to incorporate this technology into your infrastructure. There will be times that adding new innovations will greatly improve operations. But many times, a look at the bigger picture may be needed. It is not enough for a product manufacturer to say a product or technology works and meets your organizations specifications. That is where Avetec’s DICE program can help greatly. DICE is the community’s source for independent, third-party product and technology testing, data management research and advisory services. Working with data centers and vendors, DICE serves as a collaborative bridge bringing insight and perspective to the entire community. It is the only test and evaluation environment of its kind. DICE has become a trusted source of independent, credible and representative information about data intensive computing and efficiency challenges, demands and issues. Uncover the unexpected in your new technology upgrade by first contacting the Avetec DICE program and let us paint a complete picture.