Benchmarking Next Generation Hardware Platforms: An Experimental Approach

Vishakha Gupta
Georgia Institute of Technology

Adit Ranadive
Georgia Institute of Technology

Ada Gavrilovska
Georgia Institute of Technology

Karsten Schwan
Georgia Institute of Technology

Abstract— **Heterogeneous multi-cores–platforms comprised of both general purpose and accelerator cores—are becoming increasingly common. Further, with processor designs in which there are many cores on a chip, a recent trend is to include functional and performance asymmetries to balance their power usage vs. performance requirements. Coupled with this trend in CPUs is the development of high end interconnects providing low latency and high throughput communication. Understanding the utility of such next generation platforms for future datacenter workloads requires investigations that evaluate the combined effects on workload of (1) processing units, (2) interconnect, and (3) usage models. For benchmarks, then, this requires functionality that makes it possible to easily yet separately vary different benchmark attributes that affect the performance observed for application-relevant metrics like throughput, end-to-end latency, and the effects on both due to the presence of other concurrently running applications. To obtain these properties, benchmarks must be designed to test different and varying, rather than fixed, combinations of factors pertaining to their processing and communication behavior and their respective usage patterns (e.g., degree of burstiness).**

The 'Nectere' benchmarking framework is intended for understanding and evaluating next generation multicore platforms under varying workload conditions. This paper demonstrates two specific benchmarks constructed with Nectere: (1) a financial benchmark posing low-latency challenges, and (2) an image processing benchmark with high throughput expectations. Benchmark characteristics can be varied along dimensions that include their relative usage of heterogeneous processors, like CPUs vs. graphics processors (GPUs), and their use of the interconnect through variations in data sizes and communication rates. With Nectere, one can create a mix of workloads to study the effects of consolidation, and one can create both single- and multi-node versions of these benchmarks. Results presented in the paper evaluate workload ability or inability to share resources like GPUs or network interconnects, and the effects of such sharing on applications running in consolidated systems.

I. Introduction

Leading architects have discussed [1] the need for large-scale parallelism, heterogeneous cores, and accelerators to achieve performance and energy efficiency in future systems. The resulting *rapid evolution in functionally and performance asymmetric platforms* is evident from recent industry efforts like Intel's Sandybridge and AMD's Fusion architecture, current evaluation systems like Intel's QuickIA platform combining Atom with Xeon processors [2] and IBM's PowerEN [3] processors. Systems with specialized processors like those used for accelerating computations [4], [5], network processing, or cryptographic tasks have also proven their utility in terms of higher performance and lower power consumption. Concurrent with this evolution and increased diversity in server platforms, cloud computing providers are more aggressive about incorporating heterogeneity into their datacenter infrastructures, so as to better support emerging classes of cloud applications, such as online gaming – OnLive [6], financial applications [7], high quality media delivery and processing – Netflix [8], and others. Furthermore, some of these classes of applications pose stricter requirements on the I/O capabilities of the cloud infrastructure, in terms of increased bandwidth or lower and more predictable latency, thereby raising the need for use of high-end I/O fabrics, such as 10Gig Ethernet [9] and InfiniBand [10]. Recent examples include Amazon's high performance (HPC) and GPU cloud offerings [11], federally funded GPU-based machines [12], as well as datacenter clusters and software offered by vendors that specifically address the needs of high performance parallel codes [13], [14].

With next generation platforms, it is a challenge to understand and evaluate their potential utility for and the limitations they may impose on future applications. This includes understanding the choice between CPU vs. accelerator targets, the relative benefits offered by different interconnects, and the effects of scheduling schemes on application performance metrics like throughput or end-to-end latency. Further, the difficulties in doing so do not lie in understanding whether a certain single-node code can run faster on say, a GPU vs. a CPU, but in what 'balance' of processing vs. say, interconnect performance is needed to obtain high application level performance. In other words, what is needed are benchmarks in which it is possible to separately and easily vary multiple benchmark characteristics to stress different and multiple system balance points.

To address this challenge and understand the relative benefits and trade-offs for different features offered by next generation high end cloud platforms, we have developed 'Nectere' – a benchmark suite that offers the flexibility to easily construct application benchmarks to exercise one or more of the aforementioned platform characteristics. Nectere fills a gap between benchmarks focused on general purpose data center systems like Cloudstone or RuBiS [15], [16] and benchmarks focused on single features, like the use of accelerators on a GPU-based machine [17], [18] or the use of high-end fabrics like InfiniBand(IB) [19]. Further, Nectere makes it possible to go beyond single applications (e.g., financial codes [7]) or specific system components (e.g., to evaluate networks or IO subsystems [20]), by making it easy to combine the capability to use heterogeneous processing resources with that of using high end network interconnects. With Nectere, one can build distributed applications requiring low end-to-end latencies and exercising low-latency interconnects like InfiniBand, as well as construct web-centric high throughput codes that require acceleration to meet the aggregate demands of large numbers of web clients. Addressing the fact that virtualization is the common systems layer in most modern datacenters, Nectere works seamlessly with the Xen hypervisor (as demonstrated by our evaluation).

This paper describes the Nectere framework and then demonstrates its flexibility by using it to construct and experiment with two different classes of datacenter applications. First, we develop and evaluate a *latency-sensitive* financial benchmark–an options processing application [7] which can be run on a single node and/or in a distributed setting. It can also use different degrees of acceleration to meet certain required levels of end-to-end latency. Second, we use Nectere to construct a *throughput-sensitive* web-based image processing benchmark that emulates and enhances the functionality of commercial web applications like HP's Snapfish, but with the novel feature of combining with such web actions, additional processing for image conversion and manipulation. Either of these applications can run on general purpose (with possible performance or functional asymmetries among the x86 cores) and/or GPU-accelerated (programmed with CUDA) platforms.

The following unique functionality is offered by Nectere:
(1) *Modular configuration for evolving hardware and software.* Benchmarks can be configured to utilize different degrees of parallelism, to address next generation processor hardware like on-chip and off-chip accelerators (e.g., GPUs) and high end network interconnects like InfiniBand.
(2) *Varied execution patterns.* Programmers can define different execution patterns for computational kernels that run on CPUs or accelerators, e.g., pipelined kernel execution of individual stages vs. farm-out of data to computation/IO kernels running in parallel vs. each application stage at an individual node in the cluster. Such diverse configurations can be used to evaluate system aspects like per-node scheduling or to evaluate choices between different kinds of processing units based on input data and platform constraints.
(3) *Distributed execution.* Nectere supports distributed applications, so that evaluation can go beyond running single-node individual computational kernels [21] or specific parallel codes [18], thereby addressing typical datacenter use cases like those targeted by multi-tier benchmarks like RuBiS.
(4) *Diverse communication media.* Communication can be configured to use commodity networks like Ethernet and/or to exploit the high end IB interconnect via IP or native IB libraries.
(5) *Unique dimensions for performance assessment.* The metrics used in this paper extend traditional throughput or end-to-end delay measurements to also consider the variations in performance experienced by applications due to consolidation. This is important when running applications in consolidated systems and/or to understand the effects seen by workloads when they are run on different types of cloud or datacenter resources.

In summary, this paper describes the following technical contributions. It presents the Nectere framework for constructing and running representative benchmarks for next generation systems and hardware. Nectere is used to construct different classes of applications and to characterize their needs and online behavior in realistic usage environments. Evaluation uses metrics that also consider the contexts in which these applications are run, e.g., to address consolidation. Experimental measurements are made with modern accelerators like GPUs and use high end interconnects like InfiniBand.

In the remainder of this paper, we first briefly discuss the Nectere framework in Section II along with the description of our financial and image processing examples. Section III describes the testbed, workloads, and analyses of these applications. Related work appears in Section IV, followed by conclusions and future work in Section V.

II. Nectere Framework and Examples

Researchers currently use a rich set of benchmarks to evaluate modern platforms and high end systems. DayTrader models financial trading applications, RuBiS targets multi-tier datacenter codes, Olio [22] addresses data-intensive operations, and Memcached [23] emulates the multi-node caching required by Facebook and other web companies. Each such benchmark addresses a specific class of applications, with typical workloads targeting homogeneous (i.e., general-purpose CPUs only) commodity machines. Complementing such benchmarks are those for highly parallel and/or GPU-based machines, such as Parsec or Parboil. They are written to provide the different levels of parallelism needed to evaluate the possible speedup attainable on such platforms, but tend to focus on specific platform elements, such as their GPUs or their multiple CPU cores.

More complex and flexible benchmarks are needed to evaluate future platforms combining acceleration, performance- and functionally asymmetric cores, with different types of interconnects. Specifically, a richer evaluation approach is needed in order to understand how effectively such platforms balance their processing, memory, and interconnect performance, for different types of application workloads and characteristics. Nectere addresses the need to separately and easily vary the processing and communication needs of workloads targeting heterogeneous and asymmetric multicore platforms. Nectere can be used to create different types of benchmark codes, which, furthermore, can be deployed on different types of hybrid machines (i.e., with general purpose CPUs and GPUs), and on clusters of such machines interconnected with different types of fabrics (e.g., Ethernet vs. InfiniBand). In addition, with Nectere, performance evaluation can also consider how benchmark behaviors are affected by other applications using shared machine resources, as is the case for consolidated codes running in cloud or datacenter infrastructures.

A. Framework

Figure 1 depicts the modular Nectere framework, which includes separate components to represent application logic, performance monitoring utilities, configuration scripts, and networking libraries. The goal is to provide (1) sufficient building blocks for applications that go beyond individual kernels to instead, consist of multiple, different, and potentially distributed components/tiers, external 'client' interactions, etc., coupled with (2) the tools needed for configuring or interpreting these applications' desired behaviors. Building blocks include (1) an extendible library of computation kernels supporting different programming models via a common interface, so that they can be combined into a single, richer application with multiple potential processing targets, (2) networking libraries capable of using TCP/IP or InfiniBand RDMA-based connection for low latency communication, (3) utilities like timers and statistics routines useful for performance measurement, and (4) configuration options that govern applications' execution patterns.

In the figure, the *CPU-only* and *CUDA-based* branches within the *Options Processing* and *Image Processing* benchmarks contain the CPU portion and the GPU portion of functionalities, respectively, for our sample applications. The computational or IO kernels for these benchmarks are encapsulated in libraries. The application driver (henceforth termed 'driver') present in the main benchmark folders can then combine these CPU, GPU, or other such application portions in an order relevant for evaluation, as described by the configuration. The user is also responsible for specifying the combined execution order of this driver. Further, the driver can read requests from files on the same machine or listen on an Ethernet/InfiniBand connection for remotely issued requests (e.g., by workload generators like Faban [24]). There is a main application launcher within the *Applications* branch that can launch network server and client portions or create handles for the input and output data files. It also launches the driver for selected benchmarks, as requested by the user. The following example applications highlight some of the functionality and usage models.

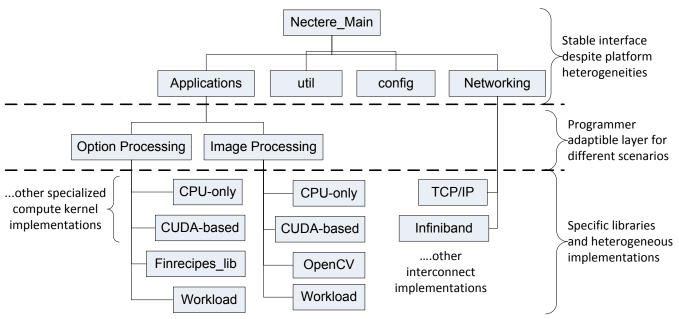
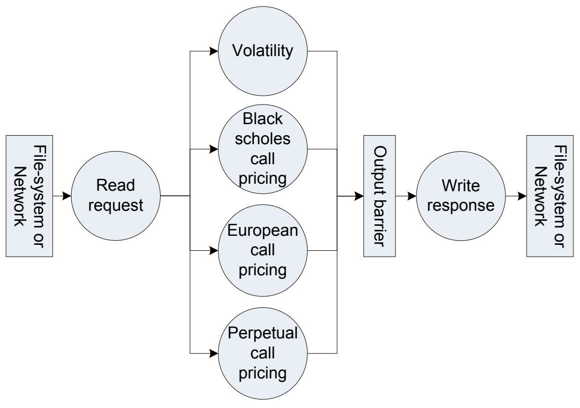

Nectere_Main
Applications
util
config
Networking
Stable interface despite platform heterogeneities
Option Processing
Image Processing
Programmer adaptible layer for different scenarios
...other specialized compute kernel implementations
CPU-only
CUDA-based
Finrecipes_lib
Workload
CPU-only
CUDA-based
OpenCV
Workload
TCP/IP
Infiniband
....other interconnect implementations
Specific libraries and heterogeneous implementations

Fig. 1. Nectere framework with example applications

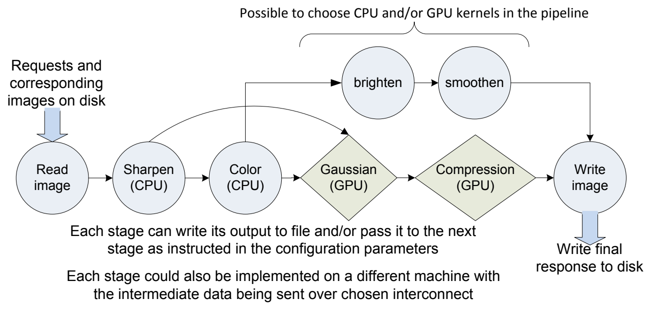
B. Sample Applications

We have chosen to create two application benchmarks to represent future multicore and multi-node codes able to exploit heterogeneous platforms combined into high end systems.

Options processing application [Nectere-options]. This financial application is modeled using inputs from a financial company with whom we have been interacting [7]. It uses a server and client communicating over InfiniBand, and it makes use of the high end CPU-only kernels for options processing [25] provided through the finrecipes library (see Figure 1). While these computational kernels, CPU-only or CUDA-based hybrid ones, can be combined in any fashion expected of the server, we run them in parallel on all incoming requests to calculate the response that is then returned to the client, as shown in Figure 2(a). Further, instead of receiving(sending) requests(responses) over the network, these could be read(written) from(to) a file. The use of files enables experimentss to proceed on a single machine, which can be quite useful while testing say, the scheduling logic offered by the platform under evaluation. The rate at which client requests are sent(read) can be tuned to the request rate desired in the scenario being evaluated.



(a) One possible configuration of Nectere-options


Possible to choose CPU and/or GPU kernels in the pipeline
Requests and corresponding images on disk
brighten
smoothen
Read image
Sharpen (CPU)
Color (CPU)
Gaussian (GPU)
Compression (GPU)
Write image
Each stage can write its output to file and/or pass it to the next stage as instructed in the configuration parameters
Write final response to disk
Each stage could also be implemented on a different machine with the intermediate data being sent over chosen interconnect

(b) Possible hybrid or CPU-based Nectere-image

Fig. 2. Nectere framework allows programmers to add different computation kernels and use the existing data structures as well as utility functions to construct applications exhibiting various execution characteristics

Since this options processing benchmark is quite sensitive to the latency experienced while processing a client request, it places stringent requirements on each stage through which a request passes. It can also have a high volume of client requests, thereby representing the category of applications that demand low latency with low performance variability, even when processing a large number of requests.

Image processing application [Nectere-image]. We use Intel's OpenCV [26] library functions and image processing functions implemented within the CUDA SDK and uploaded on the NVIDIA CUDA developer zone to compose a high throughput image processing server. Figure 2(b) shows multiple execution patterns implemented for the CPU-only as well as the hybrid version of the application. The requests in question are images of various sizes that can require different or all of the processing shown in the figure. The images can be read(written) from(to) disk(network - Ethernet or InfiniBand). We have also evaluated a CPU-based version with each processing stage running on different cluster nodes, thereby creating a pipeline extending across multiple machines.

The frequency of client requests, the sizes of images, and the number of processing stages govern the throughput and round trip response time of this application. Its multi-stage nature makes it representative of the many service-based applications that transfer large amounts of data at different rates, at different expected levels of service (i.e., with differing service level agreements – SLAs).

C. Implementation Detail

Nectere provides high resolution timer functions which are part of *util* in Figure 1. They can be used to profile the applications and each application stage. Further, it is easy to create new kernel interaction patterns by using alternate kernels and application examples. We use the fastflow [27] library for lock-less IO request buffers, if required for data/request exchanges between different stages. The top level makefile and configuration files determine the compilation and execution of an application run. Sample input generators can be used for patterns displayed by real data.

III. Workload Analysis

Nectere-constructed benchmarks can have different combinations of execution properties, like CPU intensity, network utilization, affinity to disk usage, the potential to benefit from acceleration, etc. Such benchmarks can be used to learn how these classes of applications with their different behaviors behave on some given platform and/or benefit from specific platform features. We now use our sample Nectere benchmarks to discuss some evaluation metrics and the different dimensions of analysis useful in cloud-like environments.

A. Testbed

Two hardware configurations are used for this evaluation.

Config-IB. This configuration consists of two Dell PowerEdge 1950 servers. Server 1 has dual-socket quad-core Xeon 1.86Ghz processors, while Server 2 has dual-socket dual-core Xen 2.66Ghz processors. Both servers have 4GB of RAM and Mellanox MT25208 HCAs installed. The machines are connected via a Xsigo 10Gbps IB Switch. Xen 3.3 is used on all servers, along with using para-virtualized InfiniBand driver modules that work under the Linux 2.6.18.8 dom0 and domU kernels. The guest operating systems are configured with 512 MB of RAM and have the OFED-1.2 distribution installed.

Config-GPU. The GPU-based evaluation is carried out on a system comprised of (1) a 2.5GHz Xeon quad-core processor with 3GB memory and (2) an NVIDIA 8800 GTX GPU managed through the v169.09 GPU driver. Since our primary target is cloud-based installations, we run these benchmarks in virtual machines. The virtualization for this configuration is provided by the Xen 3.2.1 [28] and the 2.6.18 Linux kernel running in Dom0 and guest domains. The GPU virtualization extension is implemented as described in [29].

Nectere benchmarks can run in distributed GPU cluster configurations with machines connected using InfiniBand fabrics [30]. We are not aware of other open-source virtualized software solutions that support both GPUs and IB cards. We next evaluate different aspects of these unique platforms, running Nectere-options and Nectere-image on the two hardware configurations described above.

B. Targeting Low Latency Interconnects

First, using a Nectere benchmark representing a latency-sensitive distributed financial application, we highlight various aspects the application behavior when running over a low latency interconnect, InfiniBand. The application consists of a server and a client, which communicate using RDMA. Nectere measures the I/O and compute latencies experienced by the server and client and reports these for every request sent by the client. We can also configure Nectere to send messages of different sizes to the server, to change the application's I/O behavior.

The results in Figures 3 and 4 show the utility of the Nectere benchmark in analyzing various aspects of the application behavior on the IB-based platform. We show the effect of varying data sizes on the latencies in Figure 3. Since request latencies and therefore, application performance depend on the I/O load present on the link, it is important to understand the effects of workload consolidation on this type of latency-sensitive applications. Figures 3 and 4 show the effects of consolidation with workloads with differing I/O behaviors, in this case I/O message sizes.

In the first figure, both VMs use the same IO message sizes, while in the latter figure, the 'Other VM' message size is changing. From Figure 3, we see that as the I/O workload increases, there is a consequent increase in the latency variability experienced by both VMs. This means that both VMs suffer equally, and it also shows that at smaller data sizes, the interconnect is able to support I/O workloads almost as well as seen for the NoConsolidation Latencies. Conversely, Figure 4 shows the breakdown of total server latency for a 64KB configured VM with increasing message sizes for other VM. It shows that compute latency is constant for every request, but I/O latencies increase as the 'Other VM' message sizes increase. As variability in I/O latencies increase, they will ultimately affect the SLA required by the application.

We have composed an image pipelining benchmark, as mentioned in the previous Section II, using our communication library. This allows us to construct distributed applications that leverage the framework. We evaluate this workload for

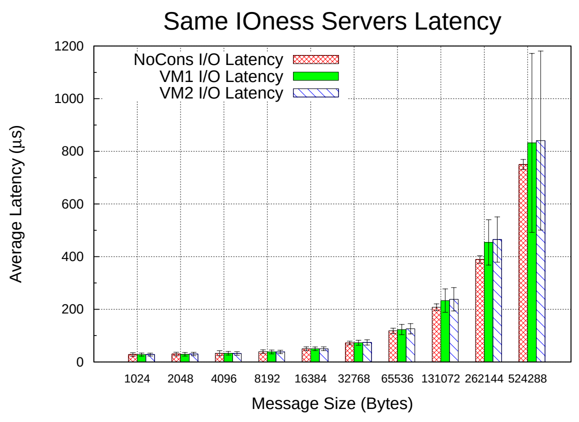


Fig. 3. Effect on Nectere IB Latency when consolidating workloads with same IO properties

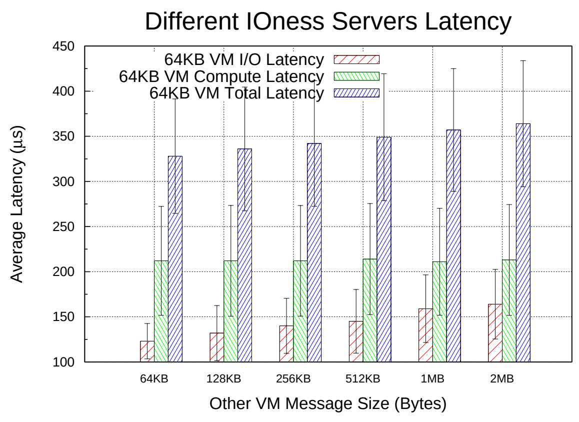


Fig. 4. Effect on Nectere IB Latency when consolidating workloads with different IO properties

TABLE I
IMAGE PIPELINING THROUGHPUT FOR DIFFERENT CONFIGURATIONS

Configuration	Throughput(KPixels/sec)
Native-Ethernet	1041.5
Native-InfiniBand	1489.4
Virt-Ethernet	655.0
Virt-InfiniBand	741.0

its throughput when running on top of InfiniBand as well as Ethernet. Table I shows the throughput achieved by our image pipelining code in various configurations for both Native and Virtualized cases with Ethernet and InfiniBand interconnects. We see that InfiniBand always does better than Ethernet but only about 13% more in the Virtualized case and 43% more in the Native case. However, when comparing InfiniBand performance it is *doubled* in the Native case when compared to Virtualized case. We attribute this to the sharing of the device between VMs (we use a single machine with 4 VMs) versus using 4 physical machines in the Native case. Such insights are useful in deciding the type of interconnect best suited for an application as well as consolidation constraints on that application.

C. Targeting GPU-based Systems

Combining processing kernels through Nectere can help analyze their behavior when consolidated or scheduled on asymmetric platforms. Figure 5 shows the execution behavior of the hybrid version Nectere-image (composition as shown in Figure 2(b)). We use images of size 512 by 512. While it fully occupies the GPU through the image processing kernels, CPU utilization remains only around 50-55%. This benchmark will therefore, be a good candidate for running on a slower core (e.g., an Atom) instead of a faster one (e.g., a Xeon) or consolidation when another CPU-intensive benchmark without a GPU component is run on the same platform.

Heterogeneous systems offer choices between alternatives, like scheduling a task on a CPU vs. GPU, which can be

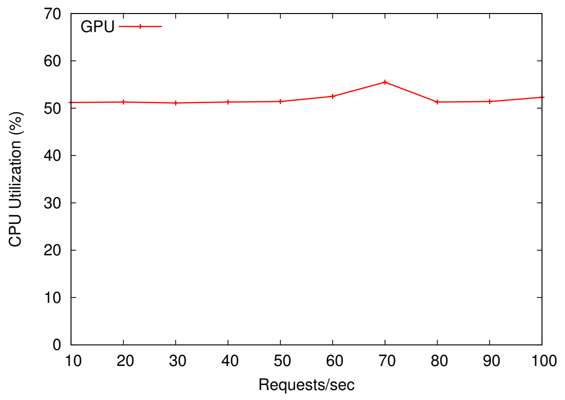


Fig. 5. Nectere framework makes different dimensions of analysis possible on accelerator-based systems. For example, in this scenario, CPU utilization remains approximately constant leaving room for other benchmarks along side Nectere-image. Thus, we could potentially improve the utilization of this platform by adding another workload that can use the rest of the CPU or we could move the CPU to a lower power state and reduce power consumption.

analyzed by composing such tasks in Nectere. Figure 6 shows the number of options processing requests processed by a GPU-only vs. a CPU-only version of the options processing benchmark in Nectere. As discussed earlier for the GPU, Figure 6 shows how CPU performance can closely follow GPU performance. After a size of 5000 options processing requests, GPU performance starts increasing beyond that of the CPU. Therefore, depending on the limits set within the system for tolerable latency and the 'busyness' of resources, CPU and GPU can be used interchangeably to service requests up to a certain request size.

We can also use Nectere benchmarks to understand the effects of different configurations of the platform software stack, such as the scheduler. Figure 7 demonstrates this from our previous use of Nectere-image to evaluate our existing GPU-virtualization infrastructure, termed Pegasus [29]. Figure 7 shows the effect of consolidation of the benchmark

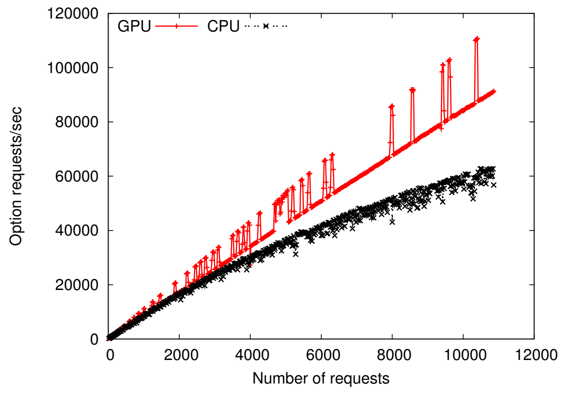


Fig. 6. CPU performance closely follows GPU performance for smaller data sizes in Nectere-options

combination shown in Figure 2(b), with GPU components, using different scheduling policies. We run three dual-core 512MB guests on our testbed. One VM (Dom2) is used for priority service and hence, given 1024 credits and 1 GPU, while the remaining two are assigned 256 credits, and they share the second GPU. VM2 is latency-sensitive, and all of the VMs require high throughput. Figure 7 shows the average throughput (pixels/sec to incorporate different image sizes) seen by each VM with four different policies. The important insight is that Nectere can be used to evaluate different software policies along with the heterogeneous hardware platforms on which they are used. In this case, AugC, CoSched, and SLAF are useful policies implemented in Pegasus, and None is the baseline in which workload scheduling is not controlled.

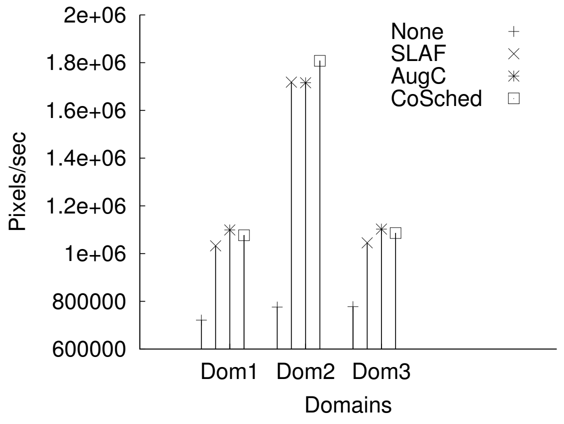


Fig. 7. Use of Nectere-gpu to evaluate different scheduling schemes

D. Summary

In summary, the results shown in this section demonstrate (1) the feasibility of using Nectere to represent different types of applications that each exercise different platform features – accelerators, high-end fabrics, etc. Results also show (2) the utility of such benchmarks in evaluating the effects of specific application properties (e.g., amount of I/O, percentage of accelerated computation, etc.) on application performance, as well as (3) to gain an understanding of the effects of consolidation and different workload mixes on application performance, or (iv) to provide insights into the effectiveness of different platform configuration options in better supporting a target workload mix (e.g., the effect of different VMM-level scheduler alternatives).

IV. Related Work

Binnig et. al. [31] argue that traditional TPC-C kind of benchmarks will no longer be sufficient to evaluate cloud systems that offer different system architecture and constraints based on pay-as-you-go services. While those metrics still have relevance for the cloud applications there is a need of different ways for measuring them for scalable (i.e., dynamic) systems where resources come and go. Nectere framework for benchmarks are been designed keeping in mind these requirements and can be extended to add various metrics while keeping the core benchmarks at the representative ones.

There are some cloud benchmarking efforts already well past the research stage. Cloudstone [15], is designed to measure the performance of clouds designed to run Web 2.0 applications. MalStone [32] is a stylized analytic computation of a type that is common in data intensive computing. Traditional 3-tier applications like RuBiS [16] and DayTrader [33], that simulate ebay-like online shopping and web-based day trading activities respectively, also form a representative set of benchmarks for evaluation of cloud and data-centers.

Research groups have developed several benchmark suites like Rodinia [17], parboil [18] and CUDA libraries for Intel's OpenCV [34] to evaluate systems with increasing number of accelerators like NVIDIA GPUs. NVIDIA provides benchmarks as part of its CUDA SDK. These benchmarks are typically used to compare the performance of applications run on CPUs vs. GPUs and for comparing among GPUs. They are also useful in learning architectural features and shortcomings of GPUs and hence lend themselves to in-depth analysis.

Lacking from all these benchmarking efforts are requirements that address challenges from these different systems through one benchmark suite. With increasing number of accelerators in cloud, data-center and HPC based systems and a push towards higher end interconnects like IB, there is a clear need for a benchmark suite that can take advantage of this modern day hardware configuration. Nectere fills that gap with its libraries capable of communicating over high performance IB connections and its CUDA integration for GPUs.

V. Conclusions and Future Work

The Nectere benchmark framework targets heterogeneous, high end server platforms that can leverage accelerators like GPUs and high end interconnects like InfiniBand. For virtualized server and cloud infrastructures, Nectere-based benchmarks can be varied easily with respect to their use of CPU, memory, and network performance requirements. This makes it possible to evaluate future platforms for multiple 'balance points' concerning the respective capacities of different platform resources. Two types of benchmarks are constructed with Nectere and used in this paper, one focused on high throughput using image processing codes and the other requiring low end-to-end latency emulating financial codes like those in futures trading. In earlier work, these were used to evaluate scheduling policies in virtualized accelerator-based systems [29], and additional work [30] will use them to better understand the performance of multi-node, multi-GPU systems as well as performance trade-offs seen for low-latency applications running across InfiniBand-connected machines [35].

The analyses conducted with Nectere, presented in this paper, demonstrate that there remain significant challenges in extracting high levels of performance from such systems. We also show that basic virtualization technologies remain insufficiently rich to permit the effective consolidation of such workloads. Our future work will use metrics other than those measuring performance, like power consumption due to data movement, when running combined accelerator- and IB-based codes.

References

[1] S. Borkar and A. A. Chien, "The future of microprocessors," *Communications of the ACM*, vol. 54, May 2011.

[2] N. Chitlur, G. Srinivasa, *et al.*, "QuickIA: Exploring Heterogeneous Architectures on Real Prototypes," in *HPCA-18*, February 2012.

[3] C. Johnson, D. H. Allen, J. Brown, *et al.*, "A Wire-Speed PowerTM Processor: 2.3GHz 45nm SOI with 16 Cores and 64 Threads," in *ISSCC*, San Francisco, USA, 2010.

[4] NVIDIA Corp., "NVIDIA's Next Generation CUDA Compute Architecture: Fermi," http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf.

[5] J. A. Turner, "The Los Alamos Roadrunner Petascale Hybrid Supercomputer: Overview of Applications, Results, and Programming," Roadrunner Technical Seminar Series, March 2008.

[6] OnLive Inc., "OnLive," http://www.onlive.com.

[7] E. Marcial, "The ICE Financial Application," http://www.theice.com, 2010, private Communication.

[8] Netflix Inc., "Netflix," http://en.wikipedia.org/wiki/Netflix.

[9] "10 gigabit ethernet," http://en.wikipedia.org/wiki/10_Gigabit_Ethernet.

[10] "InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.2," www.infinibandta.org.

[11] Amazon Inc., "High Performance Computing Using Amazon EC2," http://aws.amazon.com/ec2/hpc-applications/.

[12] J. Vetter, D. Glassbrook, J. Dongarra, *et al.*, "Keeneland - Enabling Heterogeneous Computing For The Open Science Community," http://www.nvidia.com/content/PDF/sc_2010/theater/Vetter_SC10.pdf, 2010.

[13] R. Inc., "Rackspace hosting," http://www.rackspace.com/.

[14] OpenCirrus, "Open cirrus," http://www.opencirrus.org/.

[15] W. Sobel, S. Subramanyam, A. Sucharitakul, *et al.*, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," in *Cloud Computing and Its Applications*, Chicago, USA, 2008.

[16] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel, "Performance comparison of middleware architectures for generating dynamic web content," ser. Middleware, Rio de Janeiro, Brazil, 2003.

[17] S. Che, M. Boyer, J. Meng, *et al.*, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *IISWC*, Austin, USA, 2009.

[18] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, *et al.*, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *PPoPP*, Salt Lake City, USA, 2008.

[19] OpenFabrics Group, "OpenFabrics," http://www.openfabrics.org.

[20] A. Theurer, "NetBench Performance Evaluation for Linux," http://lse.sourceforge.net/benchmarks/netbench/.

[21] NVIDIA Corp., "NVIDIA CUDA Compute Unified Device Architecture," http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_Programming_Guide_1.0.pdf, 2007.

[22] S. Subramanyam and A. Sucharitakul, "The apache olio project," http://incubator.apache.org/olio/index.data/Olio_Overview_long.pdf.

[23] Danga Interactive, "Memcached," http://code.google.com/p/memcached/.

[24] Faban, "Faban: Open source performance and load testing tool," http://www.faban.org/.

[25] B. A. degaard, "Financial Numerical Recipes in C++," http://finance.bi.no/~bernt/gcc_prog/recipes/index.html, 2007.

[26] Intel Corp., "Open Source Computer Vision Library: Reference Manual," http://www.cs.unc.edu/Research/stc/FAQs/OpenCV/OpenCVReferenceManual.pdf.

[27] M. Aldinucci, M. Danelutto, M. Meneghin, *et al.*, "Efficient streaming applications on multi-core with fastflow: the biosequence alignment test-bed," in *Parallel Computing*, 2009.

[28] P. Barham, B. Dragovic, K. Fraser, *et al.*, "Xen and the art of virtualization," in *SOSP*, Bolton Landing, USA, 2003.

[29] V. Gupta, K. Schwan, N. Tolia, *et al.*, "Pegasus: Coordinated scheduling for virtualized accelerator-based systems," in *USENIX ATC*, Portland, USA, 2011.

[30] A. M. Merritt, V. Gupta, A. Verma, *et al.*, "Shadowfax: scaling in heterogeneous cluster systems via gpgpu assemblies," ser. VTDC, San Jose, California, USA, 2011.

[31] C. Binnig, D. Kossmann, T. Kraska, *et al.*, "How is the weather tomorrow?: towards a benchmark for the cloud," ser. DBTest, Providence, Rhode Island, 2009.

[32] C. Bennett, R. L. Grossman, D. Locke, *et al.*, "Malstone: towards a benchmark for analytics on large data clouds," in *Knowledge Discovery and Data Mining*, San Diego, USA, 2010.

[33] Apache Geronimo v2.0, "Apache DayTrader Benchmark Sample," https://cwiki.apache.org/GMOxDOC20/daytrader.html.

[34] Y. Allusse, P. Horain, A. Agarwal, *et al.*, "GpuCV: an opensource GPU-accelerated framework forimage processing and computer vision," ser. MM, Vancouver, British Columbia, Canada, 2008.

[35] A. Ranadive, A. Gavrilovska, and K. Schwan, "ResourceExchange: Latency-Aware Scheduling in Virtualized Environments with High Performance Fabrics," in *IEEE Cluster*, Austin, Texas, 2011.