ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks

Venkatram Vishwanath [a,*], Robert Burns [b], Jason Leigh [a], Michael Seablom [b]

[a] Electronic Visualization Laboratory (EVL), University of Illinois at Chicago (UIC), United States

[b] Software Integration and Visualization Office (SIVO), National Aeronautics and Space Administration (NASA), Goddard Space Flight Center (GSFC), MD, United States

ARTICLE INFO

Article history:
Received 13 March 2008
Received in revised form
13 July 2008
Accepted 17 July 2008
Available online xxxx

Keywords:
Data-intensive computing
Multi-dimensional remote data striding
Climate modeling and analysis
Distributed data caches
LambdaGrids
Tropical cyclone analysis
Hurricane analysis

ABSTRACT

Data-intensive scientific applications require rapid access to local and geographically distributed data, however, there are significant I/O latency bottlenecks associated with storage systems and wide-area networking. LambdaRAM is a high-performance, multi-dimensional, distributed cache, that takes advantage of memory from multiple clusters interconnected by ultra-high-speed networking, to provide applications with rapid access to both local and remote data. It mitigates latency bottlenecks by employing proactive latency-mitigation heuristics based on an application's access patterns. We present results using LambdaRAM to rapidly stride through remote multi-dimensional NASA Modeling, Analysis and Prediction (MAP) 2006 project datasets, based on time and geographical coordinates, to compute wind shear for cyclone and hurricane and tropical cyclone analysis. Our current experiments have demonstrated up to a 20-fold speedup in the computation of wind shear with LambdaRAM.

1. Introduction

Interactive real-time exploration and correlation of multi-terabyte and petabyte datasets from multiple sources has been identified as a critical enabler for scientists to glean new insights in a variety of disciplines, critical for national security, including climate modeling and prediction, biomedical imaging, geosciences, and high-energy physics [1]. The critical performance bottlenecks in such data-intensive applications are the access latencies associated with storage systems and remote data access. These bottlenecks cause the Goddard Earth Observing System (GEOS5) global climate system, used in NASA's climate Modeling, Analysis, and Prediction (MAP) project [2], to idle for 25%–50% of the execution time on NASA's compute clusters during the analysis segment. Reducing I/O latency would allow employing more complex models, which will then result in faster and more accurate weather prediction and forecasts. Additionally, researchers copy terabyte-sized datasets located at other NASA centers or other organizations to local systems in order to run the models, which also incurs some overhead. If the remote datasets could be accessed quickly and as transparently as possible from the remote location, the overhead costs associated with copying and storing the datasets can be significantly reduced. This would enable real-time remote data analysis, reduce errors associated with replicated data and mitigate the cost of the storage systems for these replicas.

These data-intensive real-time experiments are now being enabled by the OptIPuter [3], a new paradigm in data-intensive distributed computing, funded by the National Science Foundation (NSF), to build a real-time planetary-scale supercomputer by interconnecting distributed storage, computing, and visualization resources over ultra-high speed photonic networks at tens of gigabits per second. This is known as a LambdaGrid. We present LambdaRAM, a high-performance, multi-dimensional, distributed ram-disk abstraction that harnesses the memory of multiple clusters interconnected by ultra high-speed networking, to provide data-intensive scientific applications with rapid access to both local and remote data. LambdaRAM mitigates the latency bottlenecks associated with storage systems and wide-area networking by employing proactive latency mitigation heuristics.

In this paper, we report on our successful experience using LambdaRAM to rapidly stride, based on time and based on geographical co-ordinates, to compute the wind shear for remote multi-dimensional NASA MAP 2006 datasets over high-speed networks. Rapid striding and computation of wind shear is of

* Corresponding address: Electronic Visualization Laboratory (EVL), University of Illinois at Chicago (UIC), Room 1120 SEO, 60607-7053 Chicago, IL, United States. Tel.: +1 312 996 3002; fax: +1 312 413 7585.

E-mail addresses: venkat@evl.uic.edu (V. Vishwanath), robert.w.burns@nasa.gov (R. Burns), spiff@uic.edu (J. Leigh), michael.s.seablom@nasa.gov (M. Seablom).

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.07.005

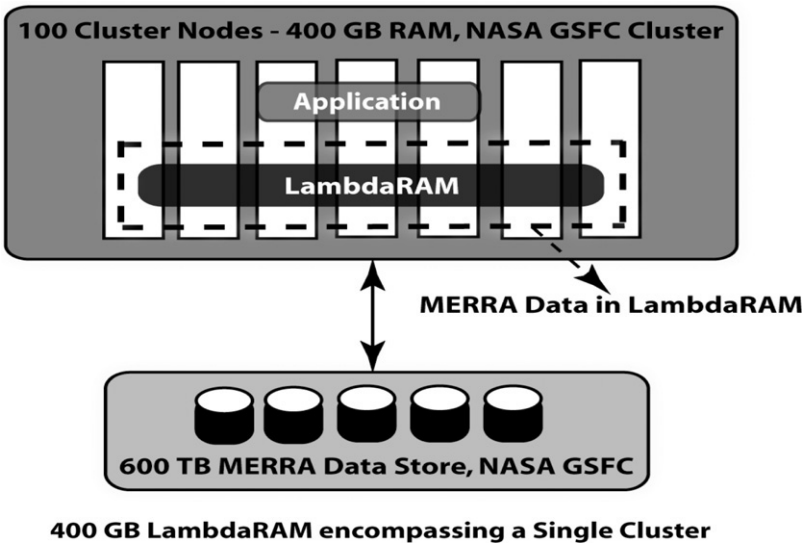

Fig. 1. LambdaRAM encompassing the memory of nodes of a single cluster.

paramount importance in real-time analysis and prediction of tropical cyclones and hurricanes. LambdaRAM yields significant speed-ups in this computation. The paper is organized as follows: In Section 2, we present a brief overview of LambdaRAM. Detailed experimental results using LambdaRAM to stride and compute wind shear is presented in Section 3. We discuss related work in Section 4, describe the ongoing work in LambdaRAM in Section 5, and, finally conclude in Section 6.

2. LambdaRAM — A multi-dimensional distributed data cache

LambdaRAM is a portable user-space library implemented in C++. It provides an intuitive API for applications to access and stride through multi-dimensional scientific datasets. LambdaRAM employs aggressive latency mitigation heuristics, including prefetching, presending and hybrid heuristics, based on the access patterns of an application and utilizes the large network bandwidth in LambdaGrids to fetch data before an application needs it. This mitigates the latency bottlenecks often associated with storage systems and wide-area data access. LambdaRAM currently supports read-only consistency mode wherein the original data is never modified. This is sufficient for many data-intensive high performance computing (HPC) applications [4], including data mining, remote data analysis and remote data visualization.

In Section 2.1, we discuss the physical configurations supported by LambdaRAM and elucidate the functional architecture of LambdaRAM in Section 2.2. In Section 2.3, we briefly describe various features in LambdaRAM.

2.1. Physical configurations supported in LambdaRAM

The physical architectural configurations supported by LambdaRAM include: a local cluster configuration where LambdaRAM spans the memory of nodes of a single cluster; a client–server cluster configuration where LambdaRAM spans memory of two clusters; and, a hierarchical cluster configuration, wherein, LambdaRAM spans the memory of multiple clusters. We now discuss each configuration in detail.

2.1.1. Local cluster configuration

In the local cluster configuration, LambdaRAM harnesses the memory of the nodes of the entire cluster. As depicted in Fig. 1, LambdaRAM spans the entire cluster memory of the NASA GSFC cluster. It manages access to the 600 TB multi dimensional MERRA dataset [5] and enables an application, such a forecast model, to seamlessly stride through the MERRA dataset.

2.1.2. Client-server cluster configuration

The client–server cluster configuration enables an application to seamlessly access remote data, and is the most commonly used configuration. In this case, LambdaRAM encompasses the memory of the cluster where the application runs (client cluster) and the cluster storing the data repository (server cluster). A typical scenario is depicted in Fig. 2, wherein a 256-node NASA Ames cluster in California, USA and a 100-node NASA GSFC cluster in Maryland, USA are interconnected by dynamically provisionable ultra-fast high-speed optical networks over the National Lambda Rail (NLR). In the figure, a parallel weather prediction application running on the NASA Ames cluster routinely needs to access the MERRA dataset located at NASA GSFC in Maryland. In this case, LambdaRAM encompasses the combined memory of the two clusters, caches upto 1.2 TB of the MERRA data in memory at any given time, and, manages the data access to the remote multi-terabyte MERRA data repository for the application.

2.1.3. Hierarchical cluster configuration

LambdaRAM supports a hierarchical cluster configuration, wherein it can encompass the memory of multiple clusters. This is typically used when there are available clusters along the network path between the application cluster and the data repository cluster. In Fig. 3, LambdaRAM, for the MERRA application, harnesses the memory of clusters at NASA GSFC, TeraGrid Chicago and NASA Ames connected via high-speed optical networks, and, manages the application's data requests to the MERRA data repository.

2.2. Functional architecture of LambdaRAM

Fig. 4 depicts the functional architecture of LambdaRAM. An application's data requests are satisfied by the data access layer. The data access layer interacts with the distributed data cache to satisfy these data requests. The distributed data cache spans the memory of multiple clusters and cluster nodes. If the requested data is not present in the distributed data cache, it fetches the data from the storage system using the I/O abstraction layer. We now describe each subsystem in detail.

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005

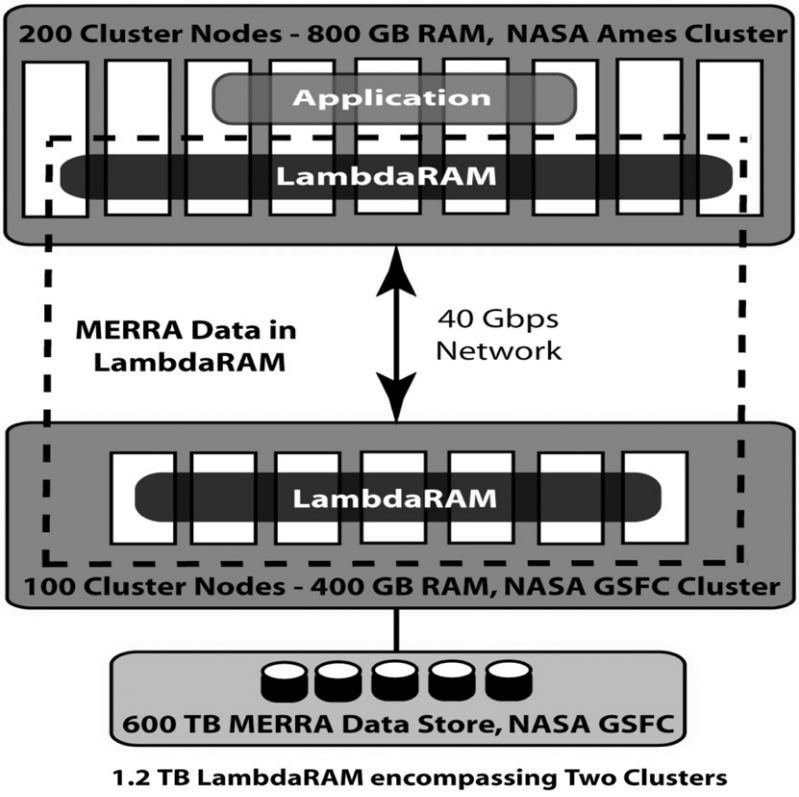

Fig. 2. Client-server LambdaRAM configuration.

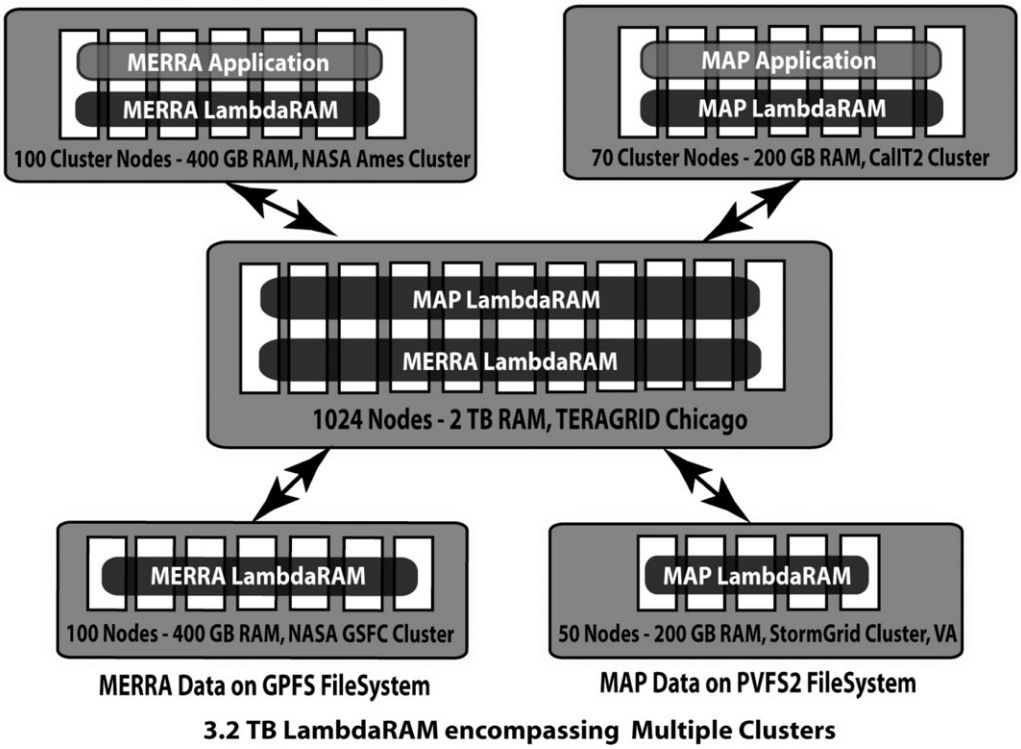

Fig. 3. Multi-cluster hierarchical LambdaRAM configuration.

2.2.1. Data access layer

An application can access data using the LambdaRAM API or the interposer layer. The LambdaRAM API presents an application with an intuitive multi-dimensional array-like interface, to access scientific datasets. It enables the manipulation of a multi-dimensional dataset, distributed over several files, as a single multi-dimensional array and relieves a scientist from spending time on data management. It currently supports 1D, 2D, 3D and 4D arrays. The interposer layer enables an application to use LambdaRAM without modifying a single line of the application's code, and thus enables

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005

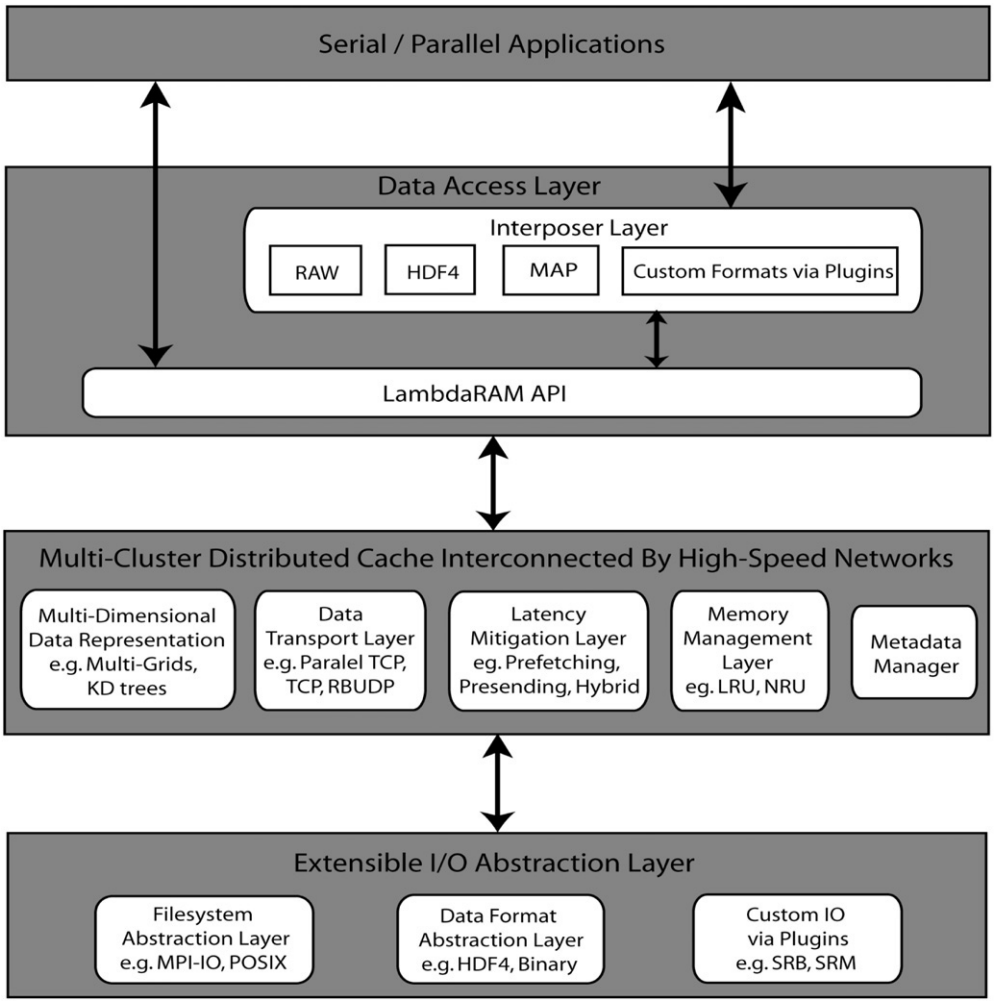

Fig. 4. Functional architecture of LambdaRAM.

easy integration of LambdaRAM with existing scientific applications. An interposer layer works by automatically mapping the data access calls of an application, including standard Unix file I/O, into equivalent LambdaRAM API calls. LambdaRAM currently supports seamless integration with applications using the TIFF, HDF4, Binary and Raw data format. Additional data formats can be supported using plug-ins.

2.2.2. Multi-dimensional distributed cache

The multi-dimensional distributed cache enables efficient access to local and geographically distributed data. It distributes the data among the various clusters via the multi-dimensional data representation layer, and manages the caches using the memory management layer. The latency mitigation layer helps mitigate the data access latencies by fetching data just before an application needs it. The data is transferred efficiently between the various levels of caches, using the data transport layer.

The multi-dimensional data representation layer enables support for multi-dimensional scientific datasets. It currently supports regular structured scientific data representations, including multi-grids and octree. The data representation is typically a distributed data structure spread among the various nodes participating in LambdaRAM at a given cluster. In the current implementation, the data distribution is static. An application can tailor this layer, based on the structural characteristics of its data. This layer can be extended to support additional representations such as unstructured grids and KD-trees using plug-ins.

The memory management layer deals with memory management of LambdaRAM on the cluster nodes. The memory management can be tailored to an application's needs. This can either be a complex global heuristic or a simpler local scheme. Global heuristics are pertinent at the clients, while local heuristics are necessary at the servers. Currently, the memory management layer supports heuristics including least recently used (LRU) and most recently used (MRU).

The data transport layer enables the use of a number of transport protocols to optimize data transfer over local and wide-area networks. For intra-cluster communication, one could use TCP or protocols optimized for the interconnect networks such as Myrinet and Infiniband. For inter-cluster communication over high-bandwidth ultra-fast optical networks, LambdaRAM supports TCP and advanced data transport protocols including RBUDP [6,7] and parallel TCP. Additional protocols can be supported with the help of plug-ins.

Latency mitigation heuristics are essential to mitigate the effects of data access latency of storage systems, and wide-area networking. LambdaRAM employs heuristics such as prefetching, presending and a hybrid scheme, combining prefetching and presending. Prefetching, a pull-based scheme, is typically initiated by the clients while, presending, a push-based scheme, is initiated by the server on behalf of the clients. In presending, the servers speculate the future access patterns of the clients and push the data onto them. Presending also effectively reduces the latency associated with prefetching by half. It improves the bandwidth

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005

usage and is useful in high-speed networks. The hybrid scheme employs presending, which is tuned using feedback received from the clients. The current implementation supports prefetching and the other schemes are under development. The latency mitigation heuristics can be tuned to an application's access pattern with the help of a customized plug-in that improves the accuracy of the prediction.

The metadata manager maintains the metadata of the datasets managed by LambdaRAM. It also keeps track of the open datasets, associated dataset files and the state of the various caches.

2.2.3. Extensible I/O abstraction layer

The extensible I/O layer enables efficient access to datasets in scientific data formats residing on storage systems. This layer is composed of the filesystem abstraction layer and the data format abstraction layer. The filesystem abstraction layer enables the use of various file IO interfaces, including POSIX based IO and high-performance parallel interfaces such as MPI-IO, to efficiently access files in high-performance filesystems including PVFS2 [8] and GPFS [9]. The data format abstraction layer enables support for accessing datasets present in scientific data formats. Currently, the data format abstraction layer supports HDF4 [10], MAP-Binary and TIFF data formats. Additional data formats and file IO interfaces can be supported with the help of plug-ins.

2.3. Other features of LambdaRAM

LambdaRAM provides an application with Memory Quality of Service (MemQoS). An application can specify the amount of memory LambdaRAM can use on each node and for each dataset. MemQoS is useful for assigning priorities among datasets and caching frequently accessed datasets in memory. As seen in Fig. 3, LambdaRAM supports multiple concurrent applications. It can manage multiple datasets in the cache simultaneously. This is important for data analysis applications that need to access several datasets during the analysis phase. This is also necessary for applications, such as remote visualization applications using multi-resolution data wherein each resolution is a unique dataset.

We have used symbolic model checking to formally verify the design of the read-only LambdaRAM mode [11]. Formal verification enables reliable deployment of LambdaRAM in safety-critical environments such as NASA's real-time climate analysis and forecasting applications. Formal verification helped to identify bugs in the memory management heuristic of LambdaRAM.

3. Computing the wind shear of NASA's modeling analysis and prediction (MAP) 2006 project data using LambdaRAM

In the summer of 2006, The Earth–Sun Exploration Division of Goddard Space Flight Center (GSFC) and the Science and Mission Systems Office at Marshall Space Flight Center brought together resources from NASA and from corporate partners to study tropical cyclones. The primary objective of MAP '06 was the application of NASA's advanced satellite remote sensing technologies and earth system modeling capabilities, to improve the understanding and prediction of tropical cyclones that develop in and move across the Atlantic basin. This project began in the early portion of the 2006 hurricane season and continued through late autumn. MAP '06 implemented the Goddard Earth Observing System (GEOS5), the fifth-generation global atmospheric model and the Grid point Statistical Interpolation (GSI) data analysis system. In addition, the ability of GEOS5 to initialize the Weather Research and Forecast (WRF) regional model was evaluated. The data from the model runs were used to analyze cyclone and hurricane formation with the goal of improving future hurricane forecast systems. A critical factor affecting the formation and destruction of hurricanes and cyclones is wind shear. Wind shear could be defined as the vector difference between the wind velocities at 850 mb and 200 mb pressure heights in the atmosphere. In the case of hurricanes, wind shear is important primarily in the vertical direction, and gives us a deep insight into the strength of a hurricane. A high wind shear value results in increased latent heat dissipation that reduces the strength of a hurricane over time. A lower wind shear results in a hurricane of higher intensity. Rapidly striding over remote MAP '06 datasets to compute wind shear would enable earth scientists to efficiently analyze and predict tropical cyclones and hurricanes.

The wind shear computation application is written in C++. It runs on a single node and strides over the MAP'06 data to compute the wind shear. The MAP'06 datasets are multi-dimensional and stored in a MAP-Binary format. The MAP-Binary is a customized format for MAP data and stores the data along with the associated metadata in a big-endian binary format. MAP'06 datasets from August 15 2006 to August 31 2006 were used for the experiments. This dataset was approximately 250 GB and consisted of 21 files for each day and a total of 357 files for the 17 days. The datasets are 4 dimensional with the dimensions being time, pressure levels, latitude and longitude. The wind shear computation application uses POSIX IO to read the MAP-Binary dataset. We also modified this application to use the LambdaRAM API to access the data. We would like to note that LambdaRAM supports MAP-Binary format via a data format abstraction layer plug-in.

We evaluate the performance of striding and computing the wind shear of the 4D MAP'06 dataset using LambdaRAM over high-speed networks. The experiments included striding over the entire world data, a critical component for global models, and regional striding such as striding over the Atlantic basin, critical for analysis of tropical cyclones and hurricanes developing in the Atlantic basin. We present experimental evaluation of LambdaRAM to stride based on time and based on geographical co-ordinates to compute wind shear for the MAP'06 data.

3.1. Striding and computing wind shear using LambdaRAM and with parallel filesystem

In this experiment, the 250 GB MAP'06 dataset was stored on a 17 TB PVFS2 parallel filesystem across 28 nodes of a 30-node cluster at EVL, UIC. The cluster nodes consisted of 64-bit dual processor 2.4 Ghz AMD Opterons with 4GB RAM and a 1 GigE network interface card (NIC). The nodes were interconnected via a high-performance Cisco 3750 switch with 96 Gbps bisection bandwidth. The wind shear computation application was run on one node of the cluster. We compare the performance of computing wind shear by striding over the data residing in PVFS2 to striding over the same data using LambdaRAM. We would like to note that in our experiments we read large unrelated datasets between each experimental run to mitigate the effects of the filesystem cache on the results. We discuss the performance of striding, based on time, to compute the wind shear for the entire earth in Section 3.1.1. In Section 3.1.2, we present results on striding based on geographical co-ordinates and time to compute the wind shear for the Atlantic basin.

3.1.1. Striding and computing the wind shear for the entire earth

We evaluate the performance of striding, based on time, to compute the wind shear for the entire earth. From Fig. 5, we see that a single LambdaRAM server yields a 100% speedup over the performance of striding using PVFS2. This is mainly due to the multi-dimensional data management and prefetching based on the application's access patterns in LambdaRAM. In case of PVFS2, the filesystem deals with individual files while LambdaRAM treats the entire set of files a single dataset and manages it collectively. We observe a linear speedup as we increase the

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005

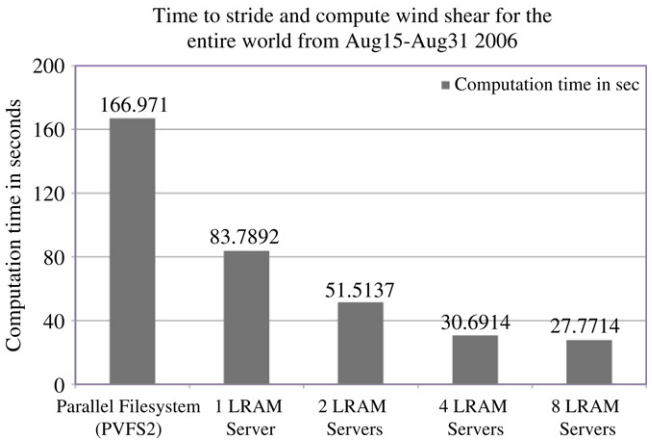

Fig. 5. Wind shear computation for the entire world over Local Area Networks. Lower computation time indicates improved performance. Results show a linear speedup as we scale the number of LambdaRAM servers to four. The bottleneck beyond four servers is due to the saturation of the 1 Gbps link of the client node.

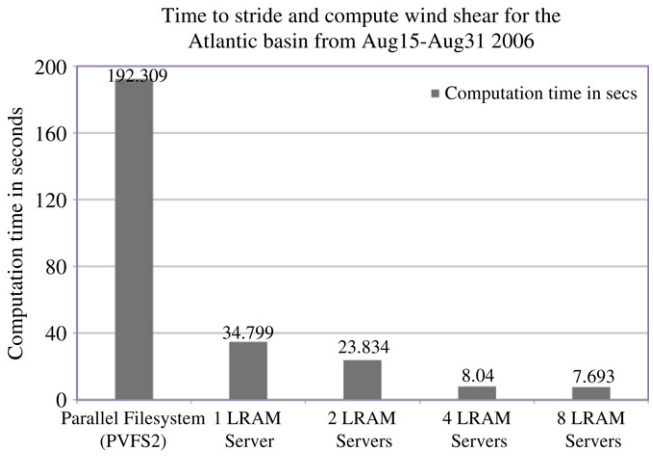

Fig. 6. Wind shear computation for the Atlantic basin over Local Area Networks. Lower computation time indicates improved performance. Results show a speedup of twenty-fold as we scale the number of LambdaRAM servers to four. The bottleneck beyond four servers is due to the saturation of the 1 Gbps link of the client node.

number of LambdaRAM servers from a single server to four servers. This is due to the increase in the available memory for caching data as we increase the number of servers. We observe a five-fold performance improvement using four servers. Increasing the number of LambdaRAM servers beyond four servers does not result in an increased speed-up due to the saturation of the 1 Gbps network bandwidth of the client node. Thus, LambdaRAM, with its multi-dimensional data management and latency mitigation heuristics, can help improve the performance of scientific applications accessing data residing in a parallel filesystem such as PVFS2.

3.1.2. Striding and computing the wind shear for the Atlantic basin

We evaluate the performance of striding, based on geographical co-ordinates and time, to compute the wind shear for the Atlantic basin. We note that striding using PVFS2 through the 4D MAP'06 dataset for the Atlantic basin involves accessing multiple noncontiguous regions, which incurs a lot of overhead. Hence, even though the Atlantic basin is a subset of the entire earth, we observe from Figs. 5 and 6 that with PVFS2 it takes more time to compute the wind shear for the Atlantic basin than the entire earth. In the case of LambdaRAM, as the data is cached in the memory of the servers, it takes less time to compute the wind shear for the Atlantic basin than the entire earth as we access less data from remote memory. Additionally, in PVFS2, the filesystem deals with individual files while LambdaRAM treats the entire set of files a single dataset and manages it collectively. From Fig. 6, we see that a single LambdaRAM server yields a five-fold speedup over the performance of striding using PVFS2. With 4 servers, we are able to achieve a twenty-fold performance improvement. Increasing the number of LambdaRAM servers beyond four servers does not result in an increased speed-up due to the saturation of the 1 Gbps network bandwidth of the client node. Thus, LambdaRAM, with its multi-dimensional data management and latency mitigation heuristics, can help improve the performance of scientific applications striding through data over many dimensions in a parallel filesystem such as PVFS2.

3.2. Performance comparison between LambdaRAM over metropolitan area high-speed network (MAN) and an ultra-fast storage system

We evaluate the performance of striding and computing wind shear for the 250 GB MAP'06 data stored locally on an ultra-fast "state-of-the-art" storage system with accessing this data remotely, using LambdaRAM, over a high-speed optical network. The MAP'06 datasets were stored on an ultra-fast multi-terabyte storage system located at the Starlight facility [12] in downtown Chicago. The storage system consists of quad processor Intel 3.2 Ghz Xeon with 24 SATA-1 disks configured in RAID5 using three hardware PCI-X RAID cards and XFS filesystem. In the LambdaRAM case, a client–server cluster configuration was used. The LambdaRAM server configuration was run on the ultra-fast storage system at Starlight while the client configuration and the wind-shear computation application were run on a cluster node at EVL. The cluster node consisted of 64-bit dual processor 2.4 GHz AMD Opterons with 4 GB RAM and a 1 GigE NIC. EVL is connected to Starlight via 2×10 Gbps Optical Network; however, the effective bandwidth between the client node and the Storage server was limited to 1 Gbps due to the 1 GigE NIC on each system. In the "ultra-fast storage system" case, the wind shear application is run locally on the storage system and accesses data using POSIX IO. We would like to note that in our experiments that we read large unrelated datasets between each experimental run to mitigate the effects of the filesystem cache on the results. We discuss the performance of striding, based on time, to compute the wind shear for the entire earth in Section 3.2.1. In Section 3.2.2, we present results on striding, based on geographical co-ordinates and time, to compute the wind shear for the Atlantic basin.

3.2.1. Striding and computing the wind shear for the entire earth

In Fig. 7, we compare the performance of striding, based on time, to compute the wind shear on the ultra-fast storage system with the same using LambdaRAM over ultra-fast networks. We observe that it takes less time to remotely stride and compute the wind shear using LambdaRAM, than computing it locally on the ultra-fast storage system. The increase in performance using LambdaRAM is due its efficient multi-dimensional data caching, management and latency mitigation heuristics to mitigate access latencies. The current bottleneck in LambdaRAM's performance is due to the saturation of the 1 Gbps network bandwidth of the client node. With 10 GE NIC becoming increasingly ubiquitous, the performance of computing wind shear with LambdaRAM for remote data will most likely increase.

3.2.2. Striding and computing the wind shear for the Atlantic basin

We compare the performance of striding, based on geographical co-ordinates and time, to compute the wind shear locally on the ultra-fast storage system with the same using LambdaRAM over

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005

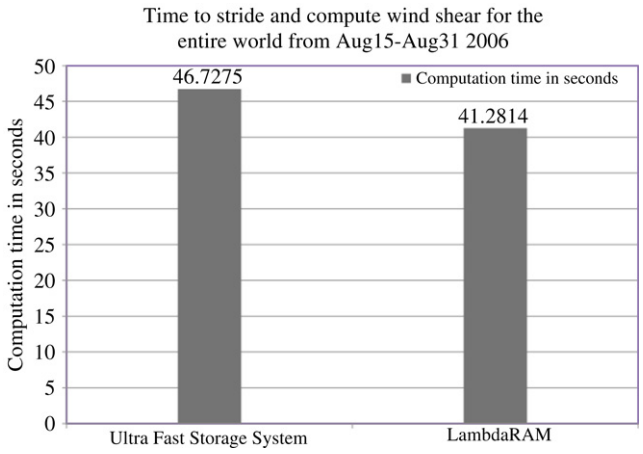

Fig. 7. Wind shear computation for the entire world on an ultra-fast storage system and using LambdaRAM between EVL and Starlight over 1 Gbps Networks. Lower computation time indicates improved performance.

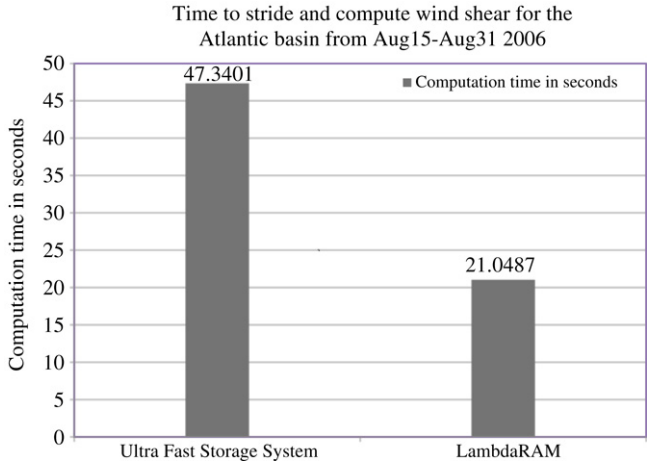

Fig. 8. Wind shear computation for the Atlantic basin on an ultra-fast storage system and using LambdaRAM between EVL and Starlight over 1 Gbps Networks. Lower computation time indicates improved performance. Results show a two-fold increase in performance to stride and compute the wind shear for remote data over computing the wind shear locally on the storage system.

ultra-fast networks. We observe from Figs. 7 and 8 that in case of the ultra-fast storage system it takes longer to compute the wind shear for the Atlantic basin than the entire earth. This is due to the fact that striding through the 4D MAP'06 dataset over the Atlantic basin involves accessing multiple noncontiguous regions on disk that incurs a lot of overhead. From Fig. 8, we observe a two-fold speed up with LambdaRAM in comparison to the ultra-fast storage system. The increase in performance using LambdaRAM is due its efficient multi-dimensional data caching, management and prefetching data to overcome access latencies. Thus, we observe that it takes less time to remotely stride over multiple dimensions and compute the wind shear using LambdaRAM than computing it locally on the ultra-fast storage node.

4. Related work

Global Arrays [13] and Charm++ [14] are two commonly used distributed shared memory implementations in HPC. Similar to LambdaRAM, they work on multi-dimensional datasets. While these systems support only local cluster operation, LambdaRAM leverages ultra-high-speed wide-area networking and low-latency reliable high-speed transport protocols [6,15,16] to extend it over multiple geographically distributed clusters in both local and wide-area networks. Parallel Filesystems, including PVFS2 [8], GPFS [9,17], and Network Filesystems, including NFS [18], support client-side file caching and work on a local cluster scale at a file-level granularity. This is complementary to LambdaRAM that works at a dataset-granularity spanning multiple files. In fact, a future goal in LambdaRAM is to use it as a caching layer for Parallel Filesystems and provide access to multi-dimensional data stored in these filesystems over wide-area networks. Distributed file systems, including, Storage Resource Broker [19], Storage Resource Manager [20] and IBP [21], and P2P-based storage systems, including Sector [22], provide efficient access to files over wide-area networks. While these filesystems work on file-level granularity, LambdaRAM works at the granularity of multi-dimensional scientific datasets spread over multiple files, and can harness the memory of multiple intermediate clusters. Memory servers based on Kernel-level implementations, including, Global Memory Service (GMS) [23], Anemone [24] and dRamDisk [25], and, user-level implementations, including, Dodo [26], NetRAM [27], cooperative caching and JumboMem [28], have been proposed. However, these implementations support only local area networks. The key innovation in LambdaRAM is that it extends memory caches in clusters, regardless of whether the clusters are located over a local-area, metropolitan-area or wide-area network.

5. Future work

We are currently evaluating the performance of LambdaRAM over wide-area networks. We are working towards a performance model of the latency mitigation heuristics for the various physical architectural configurations. We believe that this will yield a deep insight towards architecting an efficient application-oriented LambdaGrid. We are currently working on using LambdaRAM to stride through the 600 TB MERRA data expected to be available for researchers, in full, in 2009. Write-once consistency mode is useful for real-time data coupling between geographically distributed applications and we plan on extending LambdaRAM to support write-once consistency mode. LambdaRAM has been used for remote visualization with Geoscience and Bioscience applications [29,30]. We are investigating the use of LambdaRAM for remote visualization on NASA's Hyperwall — a large cluster-driven tiled visualization display [31]. This would facilitate in-situ visualizations of real-time simulations, including visualizing the formation and paths of hurricane in the Atlantic basin, in real-time, on the Hyperwall. Additionally, we are looking into using LambdaRAM as an improved memory-based checkpointing solution. This would mitigate the disk latencies associated with disk-based checkpoint solutions that adversely affect the performance of the real-time simulations that regularly use checkpointing.

6. Conclusions

LambdaRAM mitigates latency associated with storage systems and remote data access. It enables rapid striding of multi-dimensional remote datasets and efficiently manages the data for an application. Striding and computing wind shear for remote MAP'06 data using LambdaRAM gave up to a 20-fold improvement in performance over an ultra-fast storage system, and, a 5-fold improvement in performance over a parallel filesystem. LambdaRAM enables time-critical, high-performance data collaboration over both local and wide-area for data-intensive applications. In the NASA's climate modeling and analysis, LambdaRAM would result in faster weather prediction and improve the accuracy of the forecast by enabling models of higher complexity. Computational Chemistry, Genomics, Biomedical imaging suffer from similar bottlenecks and would benefit significantly from LambdaRAM. We strongly believe that LambdaRAM will aid in the design of efficient I/O systems for petascale applications and in the design of efficient LambdaGrids for data-intensive applications.

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005

Acknowledgements

We would like to thank Bill Putnam (NASA, GSFC), Carlos Cruz (NASA, GSFC), Pat Gary (NASA GSFC), Bill Fink (NASA, GSFC), Paul Lang (NASA, GSFC), Alan Verlo (EVL, UIC), Lance Long (EVL, UIC), Maxine Brown (EVL, UIC), Luc Renambot (EVL, UIC), Larry Smarr (CalIT2, UCSD), Tom DeFanti (CalIT2, UCSD), Joseph Greenseid (NGC), Anand Patwardhan (UMBC) and Milt Halem (UMBC). This material is based upon work supported by the National Science Foundation (NSF), awards CNS-0420477, OCI-0441094, and OCI-0225642, as well as funding from the State of Illinois, Sharp Laboratories of America, Pacific Interface on behalf of NTT Network Innovation Laboratories in Japan, and Northrop Grumman Corporation on behalf of NASA. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies and companies.

References

[1] Harvey B. Newman, Mark H. Ellisman, John A. Orcutt, Data-intensive e-science frontier research, Communications of the ACM 46 (11) (2003) 68–77.
[2] Modeling, Analysis and Prediction Project 2006, NASA Goddard Space Flight Center. http://map06.gsfc.nasa.gov.
[3] Larry L. Smarr, Andrew A. Chien, Tom DeFanti, Jason Leigh, Philip M. Papadopoulos, The OptIPuter, Communications of the ACM 46 (11) (2003) 58–67.
[4] C. Zhang, J. Leigh, T.A. DeFanti, M. Mazzucco, R. Grossman, TeraScope: Distributed visual data mining of terascale data sets over photonic networks, Journal of Future Generation Computer Systems (FGCS), 08/01/2003–08/01/2003.
[5] Modern Era Retrospective Reanalysis for Research and Applications (MERRA), http://gmao.gsfc.nasa.gov/research/merra/intro.php.
[6] Eric He, Jason Leigh, Oliver Yu, Thomas A. DeFanti, Reliable blast UDP: Predictable high performance bulk data transfer, in: Proceedings of IEEE Cluster Computing, Chicago, IL, September, 2002.
[7] T. DeFanti, J. Leigh, O. Yu, N. Krishnaprasad, J. Eliason, J. Alimohideen, E. He, Quanta: A toolkit for high performance data delivery, Future Generation Computer Systems 1005 (2003) 01/01/2003–01/01/2003.
[8] Parallel Virtual File System 2. http://www.pvfs.org/.
[9] General Parallel File System, IBM Corporation. http://www-03.ibm.com/systems/clusters/software/gpfs/index.html.
[10] Hierarchical Data Format, HDF Group. http://hdf.ncsa.uiuc.edu/products/hdf4/index.html.
[11] V. Vishwanath, L. Zuck, J. Leigh, Specification and verification of LambdaRAM – a wide-area distributed cache for high performance computing, in: Proceedings of the 6th IEEE/ACM Conference on Formal Methods and Models for Codesign (MEMOCODE) 2008, Anaheim, CA, USA, June 5–7 2008.
[12] Starlight – The optical startap. http://www.startap.net/starlight/.
[13] Jarek Nieplocha, Bruce Palmer, Vinod Tipparaju, Manojkumar Krishnan, Harold Trease, Edo Apra, Advances, applications and performance of the global arrays shared memory programming toolkit, International Journal of High Performance Computing Applications 20 (2) (2006) 203–231.
[14] Jayant DeSouza, Laxmikant V. Kale, MSA: Multiphase specifically shared arrays, in: Proceedings of the 17th International Workshop on Languages and Compilers for Parallel Computing, West Lafayette, IN, USA, September 22–25 2004.
[15] C. Xiong, J. Leigh, E. He, V. Vishwanath, T. Murata, L. Renambot, T. DeFanti, LambdaStream – a data transport protocol for streaming network-intensive applications over photonic networks, in: Proceedings of The Third International Workshop on Protocols for Fast Long-Distance Networks, Lyon, France, 02/02/2005–02/03/2005.
[16] V. Vishwanath, J. Leigh, E. He, M.D. Brown, L. Long, L. Renambot, A. Verlo, X. Wang, T.A. DeFanti, Wide-area experiments with LambdaStream over dedicated high-bandwidth networks, in: Proceedings of IEEE INFOCOM 2006, 04/24/2006–04/26/2006.
[17] Wei-keng Liao, Kenin Coloma, Alok Choudhary, Lee Ward, Eric Russel, Sonja Tideman, Collective caching: Application-aware client-side file caching, in: Proceedings of the 14th International Symposium on High Performance Distributed Computing, HPDC, July 2005.
[18] Network File System, RFC 1094.
[19] Reagan W. Moore, Data Management Systems for Scientific Applications, in: The Architecture of Scientific Software, Kluwer Academic Publishers, 2001, pp. 273–284.
[20] Arie Shoshani, Alex Sim, Junmin Gu, Storage resource managers: Middleware components for grid storage, in: Nineteenth IEEE Symposium on Mass Storage Systems, MSS'02, 2002.
[21] Alessandro Bassi, Micah Beck, Terry Moore, James S. Plank, Martin Swany, Rich Wolski, Graham Fagg, The internet backplane protocol: A study in resource sharing, Future Generation Computing Systems 19 (4) (2003) 551–561.
[22] Yunhong Gu, Robert L. Grossman, Alex Szalay, Ani Thakar, Distributing the sloan digital sky survey using UDT and sector, in: Proceedings of e-Science, 2006.
[23] M.J. Feeley, W.E. Morgan, F.H. Pighin, A.R. Karlin, H.M. Levy, C.A. Thekkath, Implementing global memory management in a workstation cluster, in: 15th ACM Symposium on Operating Systems Principles, SOSP 1995, Copper Mountain, CO, December 3–6, in: Ser. Operating System Review, vol. 29(5), 1995, pp. 201–212.
[24] M.R. Hines, J. Wang, K. Gopalan, Distributed Anemone: Transparent low-latency access to remote memory, in: 13th International Conference on High Performance Computing, HiPC 2006, Dec. 18–21, in: Y. Robert, M. Parashar, R. Badrinath, V.K. Prasanna (Eds.), Ser. Lecture Notes in Computer Science, vol. 4297, Springer, Bangalore, India, 2006, pp. 509–521.
[25] V. Roussev, G.G. Richard III, D. Tingstrom, dRamDisk: Efficient RAM sharing on a commodity cluster, in: 25th IEEE International Performance, Computing, and Communications Conference, IPCCC 2006, Phoenix, Arizona, April 10–12, 2006, pp. 193–198.
[26] S. Koussih, A. Acharya, S. Setia, Dodo: A user-level system for exploiting idle memory in workstation clusters, in: The Eighth IEEE International Symposium on High Performance Distributed Computing, HPDC'99, Redondo Beach, California, August 3–6, 1999, pp. 301–308.
[27] E.A. Anderson, J.M. Neefe, An exploration of network RAM, EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-98-1000, December 9, 1994.
[28] Scott Pakin, Greg Johnson, Performance analysis of a user-level memory server. in: Proceedings of the 2007 IEEE International Conference on Cluster Computing, Cluster 2007, Austin, Texas, September 2007, pp. 249–258.
[29] N. Krishnaprasad, V. Vishwanath, S. Venkataraman, A. Rao, L. Renambot, J. Leigh, A. Johnson, JuxtaView – a tool for interactive visualization of large imagery on scalable tiled displays. in: Proceedings of IEEE Cluster Computing 2004, San Diego, CA, 09/20/2004–09/23/2004.
[30] R. Singh, N. Schwarz, N. Taesombut, D. Lee, B. Jeong, L. Renambot, A. Lin, R. West, H. Otsuka, S. Peltier, M. Martone, K. Nozaki, J. Leigh, M. Ellisman, Real-time multi-scale brain data acquisition, assembly, and analysis using an end to end OptIPuter, Future Generation Computer Systems, 10/01/2006–10/31/2006.
[31] T.A. Sandstrom, C. Henze, C. Levit, The hyperwall, in: Proc. Conference on Coordinated and Multiple Views in Exploratory Visualization, July 2003, pp. 124–133.

Venkatram Vishwanath is a Ph.D. candidate in the Electronic Visualization Laboratory (EVL) and the Department of Computer Science at the University of Illinois at Chicago (UIC). His research interests include high performance networking, high-speed transport protocols, real-time data intensive applications and petascale systems.

Rob Burns is a software engineer from Northrop Grumman IT that is currently working in the Software Integration and Visualization Office (SIVO) at NASA Goddard Space Flight Center. He has been professionally involved in software design and development for nearly twelve years in environments involving embedded, desktop and high performance computing software to support telecommunication, simulation and government research.

Jason Leigh is an Associate Professor of Computer Science and director of the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago. Leigh is a co-founder of VRCO, the GeoWall Consortium and the Global Lambda Visualization Facility. Leigh currently leads the visualization and collaboration research on the National Science Foundation's OptIPuter project, and has led EVL's Tele-Immersion research since 1995. His main area of interest is in developing collaboration technologies and techniques for supporting a wide range of applications ranging from the remote exploration of large-scale data, education and interactive entertainment.

Michael Seablom is the head of the NASA Goddard Software Integration & Visualization Office (SIVO). He has the responsibility for providing a broad array of information technology services to the Earth Sciences Division in the areas of software engineering for high performance numerical modeling applications, scientific visualizations, observing systems simulation support, and education and public outreach. His office works closely with NASA's Modeling, Analysis, and Prediction (MAP) program in providing software integration services for the development of Earth system models. His technical expertise is in the areas of scientific software development and atmospheric data assimilation.

Please cite this article in press as: V. Vishwanath, et al., Accelerating tropical cyclone analysis using LambdaRAM, a distributed data cache over wide-area ultra-fast networks, Future Generation Computer Systems (2008), doi:10.1016/j.future.2008.07.005