Scaling the Earth System Grid to 100Gbps Networks[*]

Mehmet Balman[†]and Alex Sim

Computational Research Division
Lawrence Berkeley National Laboratory
1 Cyclotron Road, Berkeley, CA 94720
mbalman@lbl.gov asim@lbl.gov

Abstract

The SC11 demonstration, titled Scaling the Earth System Grid to 100Gbps Networks, showed the ability to use underlying infrastructure for the movement of climate data over 100Gbps network. Climate change research is one of the critical data intensive sciences, and the amount of data is continuously growing. Climate simulation data is geographically distributed over the world, and it needs to be accessed from many sources for fast and efficient analysis and inter-comparison of simulations. We used a 100Gbps link connecting National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory (LBNL), Argonne National Laboratory (ANL) and Oak Ridge National Laboratory (ORNL). In the demo, the Intergovernmental Panel on Climate Change (IPCC) Fourth Assessment Report (AR4) phase 3 of the Coupled Model Intercomparison Project (CMIP-3) dataset was staged into the memory of computing nodes at ANL and ORNL from NERSC over the 100Gbps network for analysis and visualization. In general, climate simulation data consists of relatively small and large files with irregular file size distribution in each dataset. In this demo, we addressed challenges on data management in terms of high bandwidth networks, usability of existing protocols and middleware tools, and how applications can adapt and benefit from next generation networks.

[*] Disclaimer:This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

[†] contact: mbalman@lbl.gov

The Need for 100Gbps Networks

High-bandwidth connections help increase throughput of scientific applications, opening up new opportunities for sharing data that were simply not possible with 10Gbps networks. However, increasing the network bandwidth is not sufficient by itself. Next-generation high-bandwidth networks need to be evaluated carefully from the applications' perspectives. In this section, we explore how climate applications can adapt and benefit from next generation high-bandwidth networks.

Data volume in climate applications is increasing exponentially. For example, the recent ”Replica Core Archive” data from the IPCC Fifth Assessment Report (AR5) is expected to be around 2PB [8], whereas, the IPCC Forth Assessment Report (AR4) data archive sums up to 35TB. This trend can be seen across many areas in science [2, 7]. An important challenge in managing ever increasing data sizes in climate science is the large variance in file sizes [3, 15, 9]. Climate simulation data consists of a mix of relatively small and large files with irregular file size distribution in each dataset. This requires advanced middleware tools to move data efficiently in long-distance high-bandwidth networks. We claim that with such tools, data can be treated as first-class citizen for the entire spectrum of file sizes, without compromising on optimum usage of network-bandwidth.

To justify this claim, we present our experience from the SC11 ANI demonstration, titled 'Scaling the Earth System Grid to 100Gbps Networks'. We used a 100Gbps link connecting National Energy Research Scientific Computing Center (NERSC), Argonne National Laboratory (ANL) and Oak Ridge National Laboratory (ORNL). For this demonstration, we developed a new data streaming tool that provides dynamic data channel management and on-the-fly data pipelines for fast and efficient data access.

Climate Data over 100Gbps

Climate data is one of the fastest growing scientific data sets. Simulation results are accessed by thousands of users around the world. Many institutions collaborate on the generation and analysis of simulation data. The Earth System Grid Federation[1] (ESGF) [8, 7] provides necessary middleware and software to support end-user data access and data replication between partner institutions. High performance data movement between ESG data nodes is an important challenge, especially between geographically separated data centers.

In this study, we evaluate the movement of bulk data from ESG data nodes, and state the necessary steps to scale-up climate data movement to 100Gbps high-bandwidth networks. As a real-world example, we specifically focus on data access and data distribution for the Coupled Model Intercomparison Project (CMIP) from Intergovernmental Panel on Climate Change (IPCC).

IPCC climate data is stored in common NetCDF data files. Metadata from each file, including the model, type of experiment, and the institution that generated the data file are retrieved and stored when data is published. Data publication is accomplished through an Earth System Grid (ESG) gateway server. Gateways work in a federated

[1] Earth System Grid Federation esgf.org

manner such that the metadata database is synchronized between each gateway. The ESG system provides an easy-to-use interface to search and locate data files according to given search patterns. Data files are transferred from a remote repository using advanced data transfer tools (i.e. GridFTP [1, 6, 15]) that are optimized for fast data movement. A common use-case is replication of data to achieve redundancy. In addition to replication, data files are copied into temporary storage in HPC centers for post-processing and further climate analysis.

Depending on the characteristics of the experiments and simulations, files may have small sizes such as several hundreds of megabytes, or they can be as large as several gigabytes [8]. IPCC data files are organized in a hierarchical directory structure. Directories are arranged according to experiments, metadata characteristics, organization lists, and simulation models. In addition to having many small files, bulk climate data consists of many directories. This puts extra burden on filesystem access and network transfer protocols. An important challenge in dealing with climate data movement is the lots-of-small-files problem [11, 16, 6]. Most of the end-to-end data transfer tools are designed for moving large data files. State-of-the-art data movement tools require managing each file movement separately. Therefore, dealing with small files imposes extra bookkeeping overhead, especially over high latency networks.

The Globus Project also recognized the performance issues with small files, and added a number of features to their GridFTP tool to address these [6]. This includes an option to do multiple files concurrently (-concurrency), and an option to do pipelining (-pipeline). They also have the -fast option, which reuses the data channel operations. Other similar parallel data mover tools include FDT [13] from Caltech and bbcp from SLAC [10].

Climate Data Distribution

Scientific applications for climate analysis are highly data-intensive [5, 2, 8, 7]. A common approach is to stage data sets into local storage, and then run climate applications on the local data files. However, replication comes with its storage cost and requires a management system for coordination and synchronization. 100Gbps networks provide the bandwidth needed to bring large amounts of data quickly on-demand. Creating a local replica beforehand may no longer be necessary. By providing data streaming from remote storage to the compute center where the application runs, we can better utilize available network capacity and bring data into the application in real-time. If we can keep the network pipe full by feeding enough data into the network, we can hide the effect of network latency and improve the overall application performance. Since we will have high-bandwidth access to the data, management and bookkeeping of data blocks would play an important role in order to use remote storage resources efficiently over the network.

The standard file transfer protocol FTP establishes two network channels [14, 16]. The control channel is used for authentication, authorization, and sending control messages such as what file is to be transferred. The data channel is used for streaming the data to the remote site. In the standard FTP implementation, a separate data channel is established for every file. First, the file request is sent over the control channel, and a data channel is established for streaming the file data. Once the transfer is completed, a control message is sent to notify that end of file is reached. Once acknowledgement for transfer completion is received, another file transfer can be requested. This adds at least three additional round-trip-times over the control channel [6, 16]. The data channel stays idle while waiting for the next transfer command to be issued. In addition, establishing a new data channel for each file increases the latency between each file transfer. The latency between transfers adds up, as a results, overall transfer time increases and total throughput decreases. This problem becomes more drastic for long distance connections where round-trip-time is high.

Keeping the data channel idle also adversely affects the overall performance for window-based protocols such as TCP. The TCP protocol automatically adjusts the window size; the slow-start algorithm increases the window size gradually. When the amount of data sent is small, transfers may not be long enough to allow TCP to fully open its window, so we can not move data at full speed.

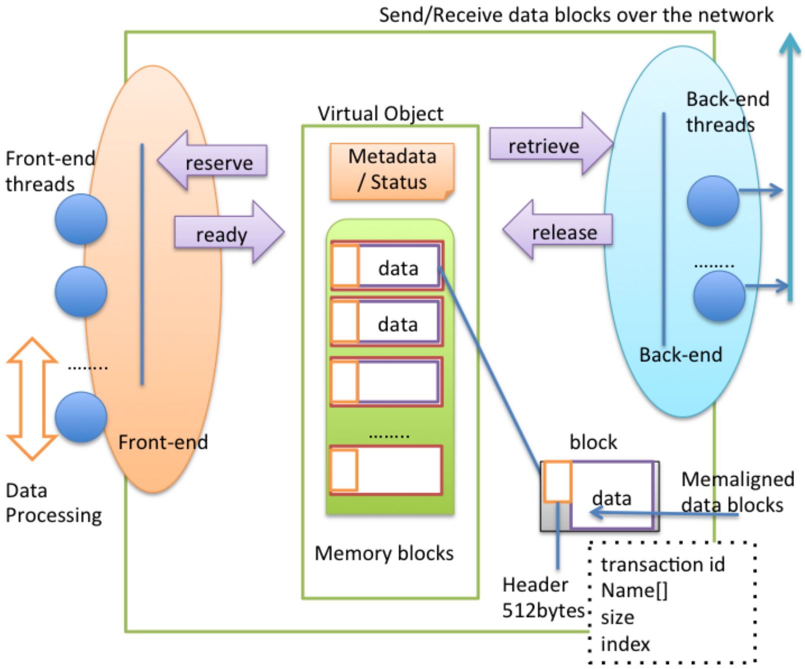


Figure 1: Climate Data Mover Framework

On the other hand, data movement requests, both for bulk data replication and data streaming for large-scale data analysis, deal with a set of many files. Instead of moving data from a single file at a time, the data movement middleware could handle the entire data collection. Therefore, we have developed a simple data movement utility, called the Climate Data Mover, that provides dynamic data channel management and block-based data movement. Figure 1 shows the underlying system architecture. Data files are aggregated and divided into simple data blocks. Blocks are tagged and streamed over the network. Each data block's tag includes information about the content inside. For example, regular file transfers can be accomplished by adding the file name and index in the tag header. Since there is no need to keep a separate control channel, it does not get affected by file sizes and small data requests. The Climate Data Mover can be used both for disk-to-disk data replication and also for direct data streaming into climate applications.

Climate Data Mover

Data movement occurs in two steps. First, data blocks are read into memory buffers (disk I/O). Then memory buffers are transmitted over the network (network I/O). Each step requires CPU and memory resources. A common approach to increase overall throughput is to use parallel streams, so that multiple threads (and CPU cores) work simultaneously to overcome the latency cost generated by disk and memory copy operation in the end system. Another approach is to use concurrent transfers, where multiple transfer tasks cooperate together to generate high throughput data in order to fill the network pipe [18, 4]. In standard file transfer mechanisms, we need more parallelism to overcome the cost of bookkeeping and control messages. An important drawback in using application level tuning (parallel streams and concurrent transfers) is that they cause extra load on the system and resources are not used efficiently. Moreover, the use of many TCP streams may over subscribe the network and cause performance degradations.

In order to be able to optimally tune the data movement through the system, we decoupled network and disk I/O operations. Transmitting data over the network is logically separated from the reading/writing of data blocks. Hence, we are able to have different parallelism levels in each layer. Our data streaming utility, the Climate Data Mover, uses a simple network library. It consists of two layers, a front-end and a back-end. Each layer works independently; so, we can measure performance and tune each layer separately. Those layers are tied to each other with a block-based virtual object, implemented as a set of shared memory blocks. In the server, the front-end is responsible for the preparation of data, and the back-end is responsible for the sending of data over the network. On the client side, the back-end components receive data blocks and feed the virtual object, so the corresponding front-end can get and process data blocks.

The front-end component requests a contiguous set of memory blocks from the virtual object. Once they are filled with data, those blocks are released, so that the back-end components can retrieve and transmit the blocks over the network. Data blocks in the virtual object include content information, i.e. file id, offset and size. Therefore, there is no need for further communication between client and server in order to initiate file transfers. This is similar to having an on-the-fly 'tar' approach bundling and sending many files together. Moreover, by using our tool, data blocks can be received and sent out-of-order and asynchronously. Figure 2 shows client/server

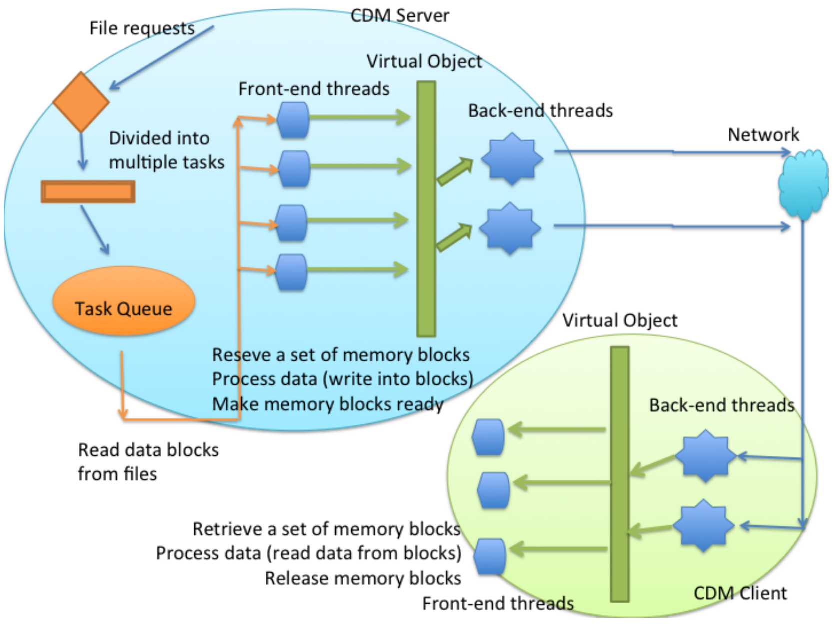


Figure 2: Climate Data Mover Server/Client Architecture

architecture for data movement over the network. Since we do not use a control channel for bookkeeping, all communication is mainly over a single data channel, over a fixed port. Bookkeeping information is embedded inside each block. This has some benefits for ease of firewall traversal over wide-area [12].

In our test case, we transfer data files from the NERSC GPFS filesystem into the memory of ANL/ORNL nodes. The Climate Data Mover server initiates multiple front-end and back-end threads. The front-end component reads data, attaches a file name and index information, and releases blocks to be sent to the client. The client at the remote site receives data blocks and makes them ready to be processed by the corresponding front-end threads. For a disk-to-disk transfer, the client's front-end can simply call file write operations. The virtual object also acts as a large cache. For disk to memory, the front-end keeps the data blocks and releases them once they are processed by the application. The main advantage with this approach is that we are not limited by the characteristics of the file sizes in the dataset. Another advantage over FTP-based tools is that we can dynamically increase/descrease the parallelism level both in the network communication and I/O read/write operations, without closing and reopening the data channel connection (as is done in regular FTP variants).

Test Results

Figure 3 represents the overall system details for the SC11 demo. We used 10 host pairs to achieve 83 Gbps of bandwidth. Each host was connected to the network with a 10 Gbps link. We used TCP connection between host pairs; default settings have been used so that TCP window size is not set specifically. We have tested the network performance between NERSC and ANL/ORNL with various parameters, such as, total size of virtual object, thread count for reading from GPFS filesystem, and multiple TCP streams to increase the utilization of the available bandwidth. According to our test results, we have manually determined best set of parameters for the setup. Specific host tuning issues about IRQ binding and interrupt coalescing described in [17] have not been applied, and are open to future explorations.

Our experiment moved data stored at NERSC to application nodes at both ANL and ORNL. We staged the Coupled Model Intercomparison Project (CMIP) data set from Intergovernmental Panel on Climate Change (IPCC) from the GPFS filesystem at NERSC. The default filesystem block size was set to 4MB. We also used 4MB blocks in Climate Data Mover for better read performance. Each block's data section was aligned according to the system pagesize. Total size of the virtual object was 1GB both at the client and the server applications. The servers at NERSC used eight front-end threads on each host for reading data files in parallel. The clients used four front-end threads for processing received data blocks. In the demo, four parallel TCP streams (four back-end threads) were used for each host-to-host connection. We have observed 83 Gbps total throughput both from NERSC to ANL, and NERSC to ORNL, as shown in Figure 4.

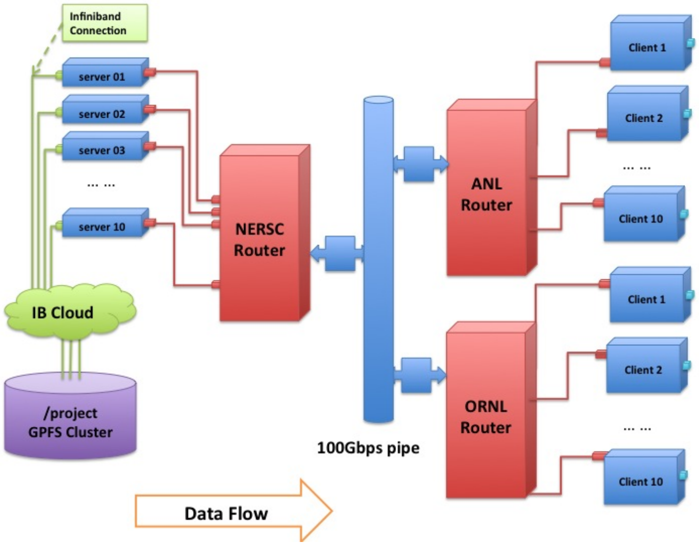
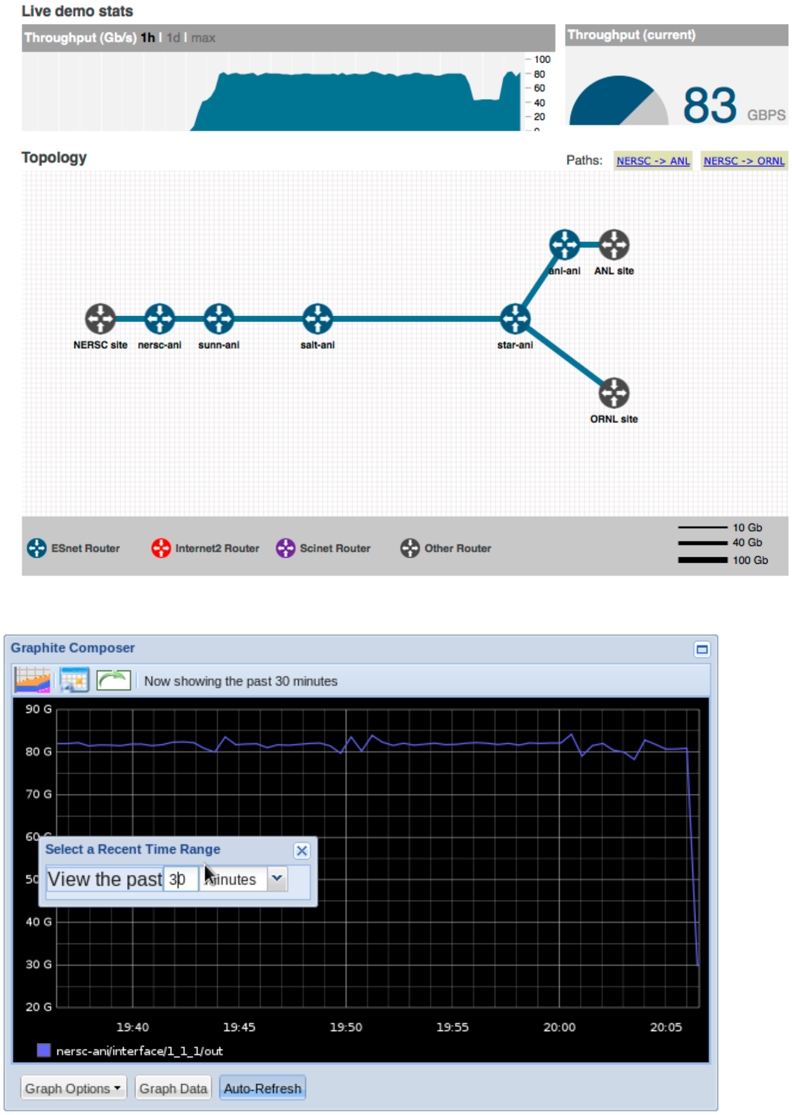


Figure 3: System diagram for the demo at SC11.



Now showing the past 30 minutes

View the past 30 minutes

Figure 4: SC11 Climate100 demonstration results, showing the data transfer throughput

Acknowledgments

This work was supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster. The globus striped gridftp framework and server. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, SC '05, pages 54–, Washington, DC, USA, 2005. IEEE Computer Society.

[2] M. Balman and S. Byna. Open problems in network-aware data management in exa-scale computing and terabit networking era. In *Proceedings of the first international workshop on Network-aware data management*, NDM '11, pages 73–78, 2011.

[3] M. Balman and T. Kosar. Data scheduling for large scale distributed applications. In *Proceedings of the 5th ICEIS Doctoral Consortium, in conjunction with the International Conference on Enterprise Information Systems (ICEIS'07)*, 2007.

[4] M. Balman and T. Kosar. Dynamic adaptation of parallelism level in data transfer scheduling. *Complex, Intelligent and Software Intensive Systems, International Conference*, 0:872–877, 2009.

[5] BES Science Network Requirements, Report of the Basic Energy Sciences Network Requirements Workshop. Basic Energy Sciences Program Office, DOE Office of Science and the Energy Sciences Network, 2007.

[6] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, and I. Foster. Gridftp pipelining. In *Proceedings of the 2007 TeraGrid Conference*, June 2007.

[7] D. N. Williams et al. Data Management and Analysis for the Earth System Grid. Journal of Physics: Conference Series, SciDAC 08 conference proceedings, volume 125 012072, 2008.

[8] D. N. Williams et al. Earth System Grid Federation: Infrastructure to Support Climate Science Analysis as an International Collaboration. Data Intensive Science Series: Chapman & Hall/CRC Computational Science, ISBN 9781439881392, 2012.

[9] S. Doraimani and A. Iamnitchi. File grouping for scientific data management: lessons from experimenting with real traces. In *Proceedings of the 17th international symposium on High performance distributed computing*, HPDC '08, pages 153–164, 2008.

[10] A. Hanushevsky, A. Trunov, and L. Cottrell. Peer-to-peer computing for secure high performance data copying. In *Proceedings of computing in high energy and nuclear physics*, September 2001.

[11] R. Kettimuthu, S. Link, J. Bresnahan, M. Link, and I. Foster. Globus xio pipe open driver: enabling gridftp to leverage standard unix tools. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, TG '11, pages 20:1–20:7. ACM, 2011.

[12] R. Kettimuthu, R. Schuler, D. Keator, M. Feller, D. Wei, M. Link, J. Bresnahan, L. Liming, J. Ames, A. Chervenak, I. Foster, and C. Kesselman. A Data Management Framework for Distributed Biomedical Research Environments. e-Science Workshops, 2010 Sixth IEEE International Conference on, 2011.

[13] Z. Maxa, B. Ahmed, D. Kcira, I. Legrand, A. Mughal, M. Thomas, and R. Voicu. Powering physics data transfers with fdt. *Journal of Physics: Conference Series*, 331(5):052014, 2011.

[14] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959 (Standard), Oct. 1985. Updated by RFCs 2228, 2640, 2773, 3659, 5797.

[15] A. Sim, M. Balman, D. Williams, A. Shoshani, and V. Natarajan. Adaptive transfer adjustment in efficient bulk data transfer management for climate dataset. In Parallel and Distributed Computing and Systems, 2010.

[16] D. Thain and C. Moretti. Efficient access to many samall files in a filesystem for grid computing. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, GRID '07, pages 243–250, Washington, DC, USA, 2007. IEEE Computer Society.

[17] W. Wu, P. Demar, and M. Crawford. Sorting reordered packets with interrupt coalescing. *Comput. Netw.*, 53:2646–2662, October 2009.

[18] E. Yildirim, M. Balman, and T. Kosar. Dynamically tuning level of parallelism in wide area data transfers. In *Proceedings of the 2008 international workshop on Data-aware distributed computing*, DADC '08, pages 39–48. ACM, 2008.