Assessing the Impact of Future Reconfigurable Optical Networks on Application Performance

Jason Maassen, Kees Verstoep, Henri E. Bal
VU University Amsterdam, The Netherlands, {jason,versto,bal}@cs.vu.nl
Paola Grosso, Cees de Laat
University of Amsterdam, The Netherlands, {p.grosso,delaat}@uva.nl

Abstract

The introduction of optical private networks (lightpaths) has significantly improved the capacity of long distance network links, making it feasible to run large parallel applications in a distributed fashion on multiple sites of a computational grid. Besides offering bandwidths of 10 Gbit/s or more, lightpaths also allow network connections to be dynamically reconfigured. This paper describes our experiences with running data-intensive applications on a grid that offers a (manually) reconfigurable optical wide-area network. We show that the flexibility offered by such a network is useful for applications and that it is often possible to estimate the necessary network configuration in advance.

1 Introduction

Computational grids are most commonly used to run single-cluster workloads. Running large parallel applications that use multiple sites simultaneously is more challenging, mainly due to limitations in the middleware [27], heterogeneity issues, and the limited capacity of the wide-area links between the grid sites. Up until recently, wide area networks as provided by the general purpose Internet, had insufficient capacity to fulfill the high demands posed by communication-intensive parallel applications and the data streams emanating from high-end scientific equipment, such as the LHC at CERN. Optical private networks [6], or *lightpaths*, provide the necessary boost that makes computational grids suitable for data-intensive e-Science applications. Using dense wavelength-division multiplexing (DWDM), a single fiber can support multiple lightpaths with bandwidths of 10 Gbit/s each. Lightpaths are often *dedicated* to a point-to-point connection between sites, or even to a single distributed application. As shown in [17, 30], by using lightpaths, grid sites can be combined to support challenging data-intensive applications.

Traditionally, lightpath configurations have been mostly static. A reconfiguration of the topology, involving optical switches, would typically require hours or even days of preparations and testing, and this setup would then be stable for many months. *Dynamically* reconfigurable lightpaths are currently being investigated as a flexible alternative [12]. Using such reconfigurable lightpaths, applications have direct control over network topology and capacity. For example, by allocating an extra lightpath on a certain link, the capacity of the link is immediately increased by 10 Gbit/s, without any changes to the hardware infrastructure. When this network capacity is no longer required, it can be redirected towards another link or be released so that it is available for another application. Research projects such as StarPlane [23], G-Lambda [26] and DRAGON [16] examine how grid middleware can be extended to support such dynamic creation of lightpaths.

It is still unclear, however, how applications can benefit most from reconfigurable lightpaths. For example, if we run data-intensive applications on a grid where we can change the wide-area links from 1G Internet to one or two 10G lightpaths, should we always configure dual 10G, or is a single 10G lightpath sufficient in most cases? Is it even possible to determine an application's wide-area network requirements beforehand, or can we only adapt the network dynamically to the needs of the application? To answer such questions, we selected three parallel applications with varying communication requirements: a fine-grained parallel model-checker, a high-resolution video-processing pipeline, and a data-intensive task farming application for supernova detection. We experimented with these applications on a lightpath-based computational grid, DAS-3 [5], in which we can easily change the lightpath configuration and assess the impact on application performance.

The remainder of this paper is structured as follows. We first discuss DAS-3, the hardware infrastructure we used for these experiments. Next, we describe the three applications, evaluate their performance, discuss how they can benefit from a dynamic network and whether is is possible to estimate their network requirements in advance. We then discuss related work and conclude.

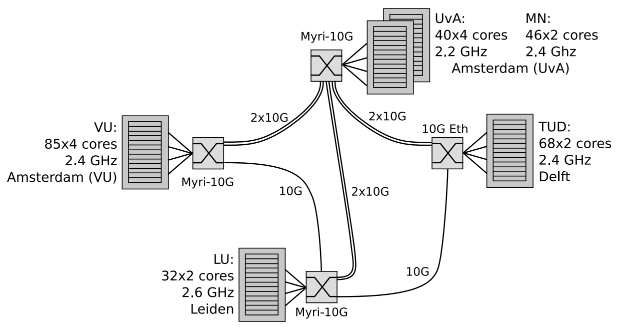

Figure 1. DAS-3 interconnected by StarPlane.

2 Testbed

Our testbed is the Distributed ASCI Supercomputer [5] (DAS-3), a wide-area distributed system for Computer Science research in the Netherlands. DAS-3 consists of five clusters distributed over four sites and uses a dedicated wide-area interconnect based on lightpaths, which are provided by the fully optical DWDM backbone of SURFnet-6 [25] (the Dutch NREN). Besides using 1 and 10 Gbit/s Ethernet, DAS-3 uses Myri-10G networking technology from Myricom [18] both as an internal high-speed interconnect and as an interface to remote DAS-3 clusters. The TUD cluster is only equipped with 1 Gbit/s Ethernet locally. Every cluster uses dual-CPU AMD Opteron nodes, but with different clock speeds and/or number of cores, making DAS-3 somewhat heterogeneous.

Figure 1 shows the current configuration of DAS-3. From the co-located UvA and MN clusters, two 10G lightpaths are available to each of the other sites. A single 10G lightpath is available between the VU and LU clusters and the LU and TUD clusters. Multiple 10G lightpaths between two sites can be combined into a single virtual link by means of LACP (Link Aggregation Control Protocol, IEEE 802.3ad). This will automatically distribute traffic over the available links. Detailed information on the DAS-3 network configuration can be found in [12].

Since the StarPlane middleware is not operational yet, we cannot easily change the physical configuration of the lightpaths. Instead, we manually reconfigure the Myri-10G switches of DAS-3 to select whether one, two or no lightpaths should be used, resulting in 10G or 20G lightpaths, or 1G Internet wide area connectivity. For this paper, only the UvA, VU and LU sites have been used.

3 Application Experiments

In this section we will take a closer look at three applications which exhibit very different communication patterns: a parallel model-checker (using fine-grained irregular all-to-all communication), a high-resolution video-processing pipeline (issuing sequences of large unicasts), and a task farming application for supernovae detection (using large file transfers, all originating from a single cluster). As we will show, these applications perform well when using lightpaths, and their wide-area network requirements vary depending on their input data or configuration.

3.1 Distributed Model Checking

DiVinE is a parallel, distributed-memory explicit-state model-checking toolkit for the verification of concurrent systems, which is developed at Masaryk University, Brno, Czech Republic. The toolkit contains a collection of state-of-the-art distributed verification algorithms suitable for use on clusters. DiVinE uses a Linear Temporal Logic (LTL) approach to model checking, where a verification problem is reduced to cycle detection in a graph representing the state space. The toolkit offers distributed state-space generation for error detection and deadlock discovery, and in addition it contains several parallel cycle detection algorithms that run efficiently on a cluster [1].

Most graph searching algorithms in the DiVinE toolkit are based on breadth-first search which can be parallelized in a straightforward fashion. Every compute node is given a portion of the state space, based on a hash function that randomizes the state distribution. The resulting communication pattern can be characterized as irregular all-to-all: every compute node repeatedly sends asynchronous messages with batched state updates to other compute nodes, in an apparently unpredictable order. The communication rate for a single compute node is not very demanding on the network (in the order of 10 to 20 MByte/s, depending on the problem), but the *accumulated* network load can be very substantial, as it increases linearly in the number of nodes participating in the computation.

DiVinE was recently augmented with a number of optimizations that improve its large-scale performance substantially [29]. To run the DiVinE toolkit on DAS-3 using 10G lightpaths, we used Open MPI [11] in TCP/IP mode. No grid-specific middleware was needed to run DiVinE: the compute nodes were reserved in advance, and MPI process startup was done via Open MPI's *mpirun*, employing ssh.

3.1.1 Performance Analysis

For our performance analysis we used the *Elevator* problem from the BEEM model checking database [19]. This problem deals with the correctness of an elevator controller, given a certain number of floors and persons waiting for the elevator. In this paper we keep the number of persons fixed at 4, and vary the number of floors between 10 and 13. For each problem instance we let DiVinE check the LTL specification of a correctness property using the Maximal Accepting Predecessors tool [1]. In all cases the entire state

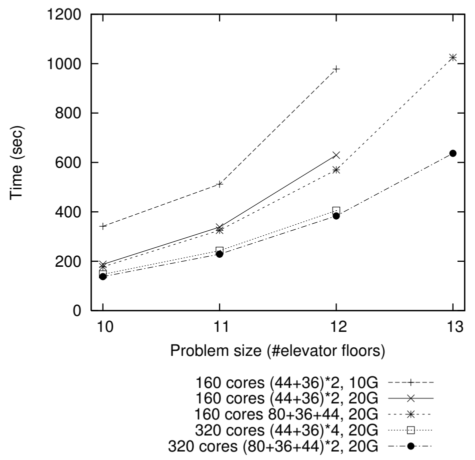

Figure 2. Elevator on 160 and 320 cores

space must be searched, since there is no violation of the property being checked.

Figure 2 shows the performance for the Elevator problem set on 160 and 320 cores with varying grid and network configurations. The compute nodes were allocated from the VU, UvA and MN clusters. As the Figure shows, the 160-core 10G grid configuration performs significantly worse than all others. This will be examined in detail below. On 320 cores, 10G simply is not sufficient for the application. Due to excessive buffering of asynchronous messages that cannot be delivered, the application starts paging. Therefore, this configuration is not included.

The 20G grid configurations, on the other hand, allow DiVinE to obtain good scaling. As shown in another recent paper [29], with 20G, grid performance is in fact remarkably close to performance on a single cluster. During most of the application's runtime, every core sends around 2500 messages per second, of 8 KByte each, to arbitrary destinations. Considering the large number of cores and the huge number of *fine-grained* data transfers involved, the performance can thus be considered excellent, showing the effectiveness of asynchronous communication in DiVinE.

The state space for problem size 13 is about twice as big as that for problem size 12. As a result, it does not fit on 80 compute nodes anymore (the state space alone requires 382 GByte of main memory). In the three-cluster grid configurations (using 80+36+44 machines) sufficient memory is available. As Figure 2 shows, the application runs efficiently on these configurations, despite the significant amount of wide-area communication required.

To examine the performance difference of 10G and 20G lightpath configurations, we instrumented the MPI networking layer of the DiVinE toolkit and plotted the effective WAN throughput as a function of the application runtime. The results are shown in the first two graphs of Figure 3. The wave pattern in the graphs is a direct reflection of the

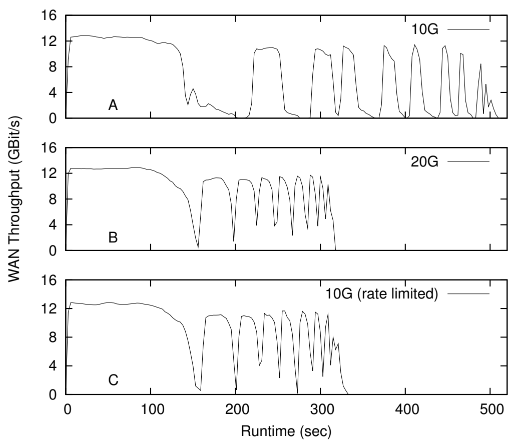

Figure 3. WAN throughput, 160 cores

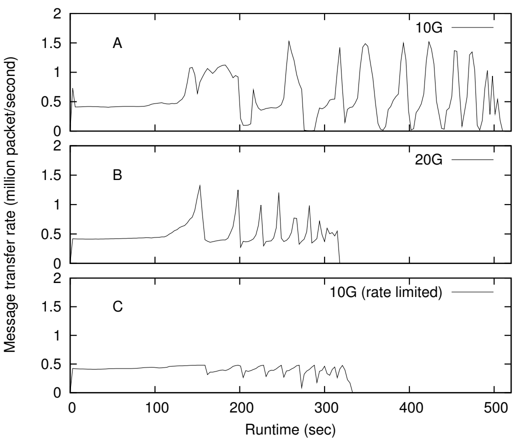

Figure 4. Message rate, 160 cores

various phases of the LTL analysis algorithm. The first phase takes longer because this is where the entire state space is generated on-the-fly. In both the 10G and the 20G case the cumulative peak WAN throughput (i.e., summed over both directions) is over 12.8 Gbit/s: each of the 160 cores transfers about 10.1 MByte/s over the WAN, and a similar amount to cores within the same cluster. In the case of a single 10G link, the application thus effectively fills a large fraction (65%) of the link's full-duplex 20 Gbit/s bandwidth with user data alone.

However, in the case of a single 10G link, the wave pattern is interrupted several times, showing a large reduction in cumulative throughput before picking up again after a number of seconds. To understand what is happening, we also need to look at the message rate; see Figure 4. The 20G configuration, which spreads the traffic over two 10G links, enables a higher overall message rate than a single 10G link. The spikes in the 20G message rates illustrate that during a significant part of the application runtime it is important to transfer small packets efficiently. This happens when cores

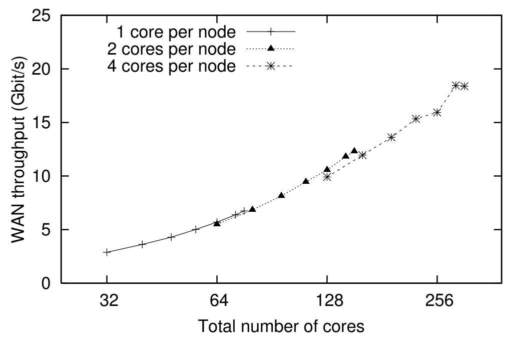

Figure 5. Peak cumulative WAN throughput

run out of local work, and start flushing buffered remote work to other cores, to obtain fast distributed termination of each compute phase. However, when already running at 65% capacity, the single 10G setup is not always able to handle this increased message rate. As a result, the switches start dropping packets, and the overall performance is significantly reduced due to TCP congestion control.

The bottom graphs in Figures 3 and 4 show the results of applying rate limitation to the DiVinE runtime system that prevents packets being flushed prematurely, which increases the effectiveness of message combining. As a result, a 160-core configuration with a single 10G lightpath is able to offer the same performance as a dual 10G lightpath setup. However, when a higher number of cores are employed, or when a model checking instance induces a higher per-core data transfer rate, using multiple 10G lightpaths will still be necessary to obtain good performance.

This is illustrated by Figure 5. The figure shows how the peak cumulative WAN throughput scales with number of cores. For this experiment, the VU and UvA clusters were used, connected via two 10G lightpaths. As the figure shows, the required wide-area bandwidth scales roughly linearly with the number of cores used, indicating a fairly stable per-core throughput. Due to this behavior, the required lightpath configuration for a certain verification problem on a given number of cores can be estimated in advance, provided that some performance results (e.g., on a smaller number of cores) are available. We can thus see that the 20G WAN configuration will again become a bottleneck for this application if the number of cores is increased beyond the 320 used in this experiment.

As the peak wide-area bandwidth required by DiVinE is fairly stable, an alternative approach can be used if its requirements for a particular instance are still unknown. When the application is started, the network can initially be configured to use an arbitrary number of lightpaths. By monitoring the throughput once the application is running, an estimate can be made of the actual network requirements, and the number of lightpaths can adapted accordingly. This does require a short lightpath configuration time (e.g., in the order of seconds), but this is considered to be feasible in next-generation lightpath infrastructure. [12]

3.2 4K video processing

Currently, technology is being developed to create, serve and display video streams at 4K (4000*2000) resolution, more than four times the resolution used in standard high-definition TV. This format is expected to be widely used for digital cinema and scientific visualization [21].

4K video is already used in *digital intermediate workflows*, a part of the movie production process where the 'look' of a movie is adjusted by changing the color, grain or sharpness of the images. The input of such workflows consists of a series images, obtained by converting the RAW image data produced by digital cinematography cameras or by scanning the individual frames of analog 35mm film. To maintain quality, the images are stored uncompressed or using lossless compression.

Current solutions for 4K digital intermediate workflows are often based on specialized hardware, designed to be used in a single location. CineGrid [3] aims to use grid computing technologies and high performance, long distance networks for collaboration on processing 4K video. In this section we will look at a proof-of-concept application that shows that a grid using lightpaths is indeed suitable for real-time processing of digital intermediates.

The input data for this application consists of a series of 3840*2160 video frames with a color depth of 48 bits per pixel, each stored in a separate file. The frames are stored in an uncompressed RGB format, requiring 47 MB of storage each. As a result, 12 Gbit/s is needed to simply play the full video stream in real time at 30 fps (frames per second). For this experiment, downscaled 2K (1920*1080) and 1K (960*540) versions of the same video frames have also been created, requiring 12 MB/frame (3.0 Gbit/s) and 3 MB/frame (0.8 Gbit/s) respectively.

The application uses a pipelined architecture and consists of a number of stages. The first stage reads the frames from disk. Next, an arbitrary number of stages process the frames to do filtering, scaling, compression, etc. The last stage displays the processed frames, or stores them on disk. To increase the processing throughput, a number of pipelines is used in parallel. Depending on their CPU requirements, the stages of each pipeline are distributed over multiple machines and/or multiple grid sites. Since the data rate required for real-time processing can be high, it is essential that sufficient wide-area bandwidth is available when the pipeline stages are distributed over multiple grid sites.

The video processing application is implemented using the Ibis communication library [15, 28]. No grid-specific middleware was needed to run the application: the compute nodes were reserved in advance, and process startup was then done via Ibis' *ibis-run*, employing ssh.

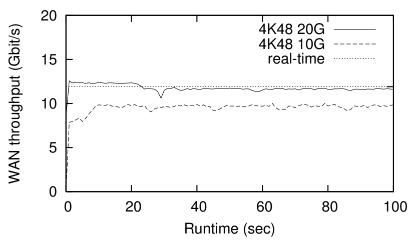

Figure 6. 4K, two sites

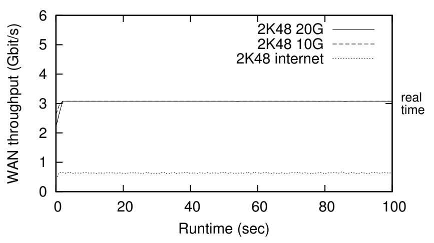

Figure 7. 2K, two sites

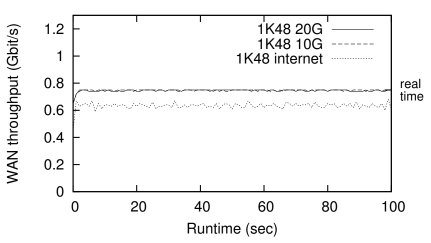

Figure 8. 1K, two sites

3.2.1 Performance Analysis

For our evaluation we will use a four-stage color correction pipeline: *read, filter, compress, count*. The read stage reads the video frames from disk and forwards them to the filter stage which slightly adjusts the color of each of the pixels. The compression stage then converts each frame to a JPEG image. Finally, the count stage, gathers all compressed frames and ensures that they have all been received in the correct order. Each pipeline is distributed over four machines, one for each stage. For this experiment, only a single core is used on each of the machines, except for the compression stage which uses two cores.

We distributed the four-stage pipeline over two clusters. At the UvA, 32 machines (and thus 32 disks) are dedicated to storage: they contain the video frames and read stage. The disk speed of each node is limited, and can only produce about one 4K frame per second. By distributing the files in a round-robin fashion over 32 machines, however, the aggregate disk throughput is high enough to reach the 12 Gbit/s required for the 4K video stream. To simulate real-time visualization, the read nodes try to maintain a combined frame rate of 30 fps by throttling the speed at which they send messages to the next stage in the pipeline. The other stages are placed at the VU, and use 32 cores (16 machines) for filtering, 64 cores (32 machines) for compression, and 1 core for gathering the compressed frames.

Figures 6, 7 and 8 show the wide-area throughput obtained between the two clusters when processing uncompressed 4K, 2K, and 1K video frames. As Figure 6 shows, when two 10G lightpaths are used, the throughput obtained by the application is sufficient for 30 fps processing of 4K video frames. When only a single 10G lightpath is available, however, the wide-area throughput becomes a bottleneck and the processing speed drops to 25 fps.

When using 2K or 1K video frames, a single 10G lightpath is sufficient. As Figures 7 and 8 show, the application obtains the required throughput, regardless of whether one or two lightpaths are used. These figures also show the throughput obtained when using the 1G Internet connection for communication between the clusters. For 2K video frames this is clearly too slow, reducing the processing speed to 6 fps. For 1K video however, the difference is much smaller. There, the application is already capable of reaching 26 fps using the Internet instead of a lightpath.

These experiments clearly show that the bandwidth requirements of this application completely depend on the resolution and color depth of the input and output video frames, and the distribution of the pipeline stages over the grid sites. As a result, determining the required lightpaths is quite straightforward and this result could be directly included in a resource reservation when the application is started [12]. Allocating more lightpaths than needed does not improve application performance in any way.

3.3 Supernovae detection

The next application is our winning contribution to the International Data Analysis Challenge for Finding Supernovae, which was held in conjunction with the IEEE Cluster/Grid 2008 conference [4]. For this challenge, a large number of celestial images were provided, based (in part) on data collected at Subaru Telescope, which is operated by the National Astronomical Observatory of Japan [24]. A sequential supernova detection application was also provided. This application takes two celestial images of the same piece of sky, taken about a month apart. It aligns the images, subtracts them from each other, and uses the result to detect objects with varying light intensity (i.e., supernova candidates). Since this object detection is based on heuristics, the amount of processing time required varies significantly between image pairs. The sizes of the images also vary, further increasing the variation in processing time.

The application uses a *task-farming* (or *master-worker*) infrastructure, based on tools provided by the Ibis project [15]. A single *master* process controls a set of worker processes on DAS-3. In addition, the master uses simple file servers, implemented using the Ibis communication library [28] to determine the location of each of the image pairs. Each worker process consist of a wrapper which (for every core in the machine) repeatedly requests a job from the master, transfers the input image pair to local disk (using the file servers), and invokes the sequential supernova detection application. The output of this application is then returned and a new job is requested. Both workers and file servers are deployed automatically by the master application onto the machines using the JavaGAT toolkit [27].

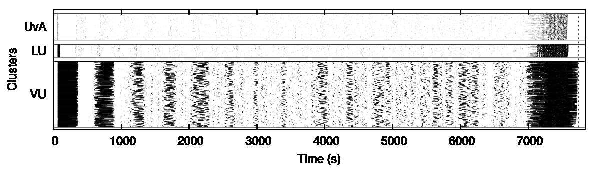

Figure 9. Data transfer activity on three sites (1G Internet)

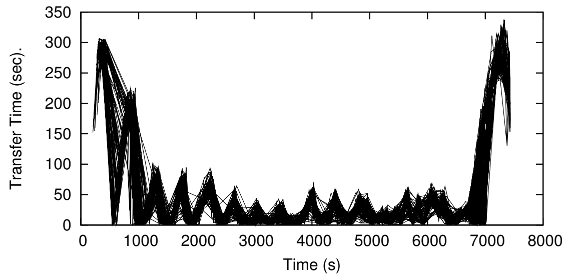

Figure 10. Job transfer times on VU (Internet)

3.3.1 Performance Analysis

To test if this application could benefit from the use of lightpaths, we transferred part of the original challenge dataset (approx. 8 GBytes) to DAS-3, and duplicated it 128 times. The resulting 1 TByte dataset (15360 image pairs) was distributed over 32 nodes (and thus 32 disks) of the UvA cluster to prevent disk access from becoming a major bottleneck. We then ran the supernova detection application on a combination of the clusters at VU (80 machines, 320 cores), UvA (32 machines, 128 cores) and LU (31 machines, 62 cores). The image pairs are sorted by size and the largest pairs are processed first. On average each job required 68 MByte of data, and took 220 seconds to process.

Figures 9 and 11 show the data transfer activity during the application run. In these figures, a line is plotted for each core. The line is black when a job is being transferred on behalf of this core, and white when the core is computing or idle. The lines are clustered into three groups, one for each DAS-3 cluster.

In Figure 9, only the Internet links are used to transfer data between the three clusters. For the largest cluster, at the VU (320 cores), the figure clearly shows extended dark areas where most cores are waiting for their data to arrive. This indicates that the Internet link between VU and UvA (where the data is stored) is insufficient for the needs of the application. Due to the variation in job processing time, this network contention is slowly reduced. At the end of the run, however, the contention increases again due to the decrease in average job processing time.

Figure 10 shows the data transfer times for each core of the VU cluster in more detail. The figure clearly shows that at the beginning and end of the run, a single job requires 250 to 300 seconds to transfer. Since the average job size is

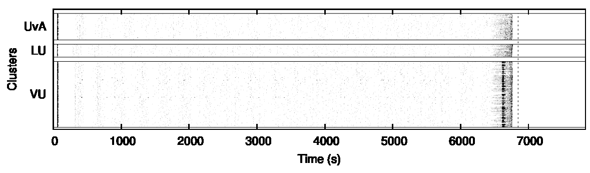

Figure 11. Data transfer activity on three sites (10G lightpaths)

68 MByte, over 21 GByte needs to be transfered at the start of the run to satisfy each of the 320 cores. Using the 1 Gbit/s (shared) Internet link, this will take at least 183 seconds, without taking other network traffic (such as the transfer to LU) into account. Although the performance is better during the rest of the run, job transfers can still take 50 to 100 seconds.

LU is the smallest cluster (62 cores). As a result, the Internet link between UvA and LU is sufficient for most of the run. Network contention mainly occurs at the start and the end of the application, as can be seen in Figure 9. Although the worst-case job transfer takes about 40 seconds, for most of the run, the transfer time is below 2 seconds, resulting in a throughput of over 34 MByte/s per transfer.

For the UvA cluster, all data is local and there is little data transfer overhead. At the end of the run, however, even the UvA nodes experience some delays. These are mainly caused by the overhead of serving files to the other clusters.

Figure 11 shows the same experiment when using 10G lightpaths between VU, UvA and LU. There, the amount of network contention is reduced significantly. For most of the run, the transfer times are well below 2 seconds, resulting in a throughput of over 34 MByte/s for each transfer. Although using dual 10G lightpaths improves data transfer performance even further, the effect is not as large at the transition from 1G Internet to a single 10G lightpath.

Interestingly, this significant improvement in data transfer times to the VU only results in a 12% speedup of the application, as the end times of Figures 9 and 11 show. Even when the data transfer overhead is high, e.g., the 50 to 100 seconds mentioned above, the average 220 second job processing time is still only increased by 22% to 45%. In addition, due to the load-balancing properties of task farming applications, a slowdown of the VU cluster (due to congestion) automatically results in an increase in the number of jobs processed on the other two clusters.

As our experiments show, the average wide-area bandwidth required varies during the lifetime of the application. The peak wide-area bandwidth will only occur during the start and end phase, when many cores are fetching data simultaneously. In addition, the smallest cluster (LU) encounters less congestion problems than the largest cluster (VU). Therefore, the network configuration could be adapted both statically to the size of the destination cluster, and dynamically to the changes in congestion during the application run. However, to truly adapt to the behavior of Figure 10, a finer control over the wide-area network bandwidth would be required than the 10 Gbit/s (single lightpath) steps that are currently available.

4 Related Work

Several projects have recently been focusing on providing grid users with an alternative, high-performance, lightpath-based networking infrastructure, both in a national and international setting [2, 6, 7, 10, 20, 22]. Currently, getting a well-functioning optical testbed typically still requires a large effort due to multi-domain administrative aspects, multi-vendor issues, and the evolving nature and implementation of standards, such as GMPLS [8]. As a result, the lightpath configurations provided by these testbeds are mostly static. Although distributed applications can benefit significantly from such setups, we have shown in this paper that *dynamic* lightpaths (as targeted by StarPlane) can be very advantageous in several distinct, but realistic application scenarios.

A prototype architecture for lightpath management in grid environments is discussed in [2]; as an example application for this system, the paper discusses a small-scale GridFTP (grid-based file transfer) experiment and the automatic creation of a virtual network connecting two servers. An OGSA-based approach for lightpath management and reservation is discussed in [9]. The paper shows the impact of lightpaths on large-scale data transfers using various network protocols, taking lightpath setup costs into account. In [14] a possible architecture for an advance reservation system for lightpath setup is discussed. The paper also contains a detailed discussion of the associated algorithms and shows simulation results for a prototype of the system.

A potential use of lightpaths for the visualization of large-scale simulations is discussed in [13]. The paper discusses how network conditions can impact the interactive component steering a molecular-dynamics simulation running on a remote cluster. An experimental architecture for resource-intensive applications is discussed in [17]. There, a simulation running on multiple clusters connected via one or more optical links is described, but a performance analysis of the various configurations is not included.

Parallel, explicit-state model checking as discussed in this paper is an interesting representative of the class of parallel applications that are suitable for running on a grid, despite their fine grain size. While analyzing the DiVinE toolkit's implementation and performance, we found it showed a remarkable resemblance to Awari [30], a distributed game-tree analysis application we recently optimized to run efficiently on DAS-3. Awari, like DiVinE, searches huge state spaces, and it shows similar (irregular, asynchronous, all-to-all) traffic patterns and data rates. The optimizations with the highest impact for Awari were related to improving the overall message rate in the application phases that send smaller packets. This is very similar to the large-scale model checking application discussed.

The real-time streaming of 4K video was first performed during a demonstration as part of iGrid 2005 [21]. There, the video stream was transferred between the USA and Japan (a distance of 15.000 km). However, only compressed JPEG 2000 data was transferred, reducing the required bit rate to 200-450 Mbit/s. Specialized hardware was used to perform the compression.

5 Conclusions

In this paper we have analyzed how applications can benefit from the use of dynamically allocatable lightpaths. The three applications we used exhibit very different communication patterns. Nevertheless, all applications performed well when distributed over two or more lightpath-connected grid sites.

For all three applications, the wide-area network requirements changed depending on the cluster configuration and/or the data set that was used. For the Supernova detection application, a single 10G lightpath was sufficient due to the relatively high computation to communication ratio and the limited size of most DAS-3 clusters. The video processing application did require 20G wide-area connectivity, but only for the largest (4K) data set. For the DiVinE tool, a single 10G lightpath was sufficient when running on 160 cores, provided that message rate limitation was used. For 320 cores two 10G lightpaths were required.

For each of the applications it is possible to get a good estimate of the required number of lightpaths in advance. For the video processing application, such an estimation is straightforward, since the required wide-area bandwidth directly depends on the resolution, color depth and required frame rate of the input video, and the distribution of the processing pipelines over the grid sites. These parameters are generally fixed and known in advance.

The wide-area bandwidth required by DiVinE to process a particular problem scales roughly linearly with the number of cores used in the computation. Therefore, the required lightpath configuration can be estimated in advance, provided that benchmark results on a smaller number of cores are available. Since the peak wide area bandwidth required by DiVinE is fairly stable, a more dynamic approach, based on monitoring the network requirements and adapting the number of lightpaths on-the-fly, may also be an option when benchmark results are not available. We plan to experiment with this approach as soon as the network reconfiguration of DAS-3 is fully operational. The issues involved there are outlined in another recent paper [12].

For the Supernova detection application, the wide-area bandwidth required is directly related to the number of cores used in each cluster and the average size of the jobs. However, due to the variation in job execution times, the average wide-area bandwidth required will be significantly lower than the maximum, during a large part of the run. As our experiments show, the peak wide-area bandwidth will only occur during the start and end phase, when many cores are fetching data simultaneously. As with DiVinE, it would be interesting to test if we can dynamically adapt the wide-area bandwidth to this changing network requirement.

Our experiments clearly show that the flexibility offered by dynamically allocatable lightpaths can be used effectively by applications. For all three applications described in this paper, it is possible to estimate the wide-area bandwidth requirements in advance. This allows a suitable lightpath configuration to be set up for each application, thereby improving its performance, and preventing the waste of unneeded wide-area bandwidth.

Acknowledgments

We want to thank the developers of the DiVinE model checker (in particular Jiří Barnat) of the Masaryk University in Brno for making their excellent tool publicly available. StarPlane is a collaboration of VU University, Amsterdam and University of Amsterdam, and supported by NWO grant 643.000.504, SURFnet, the BSIK program GigaPort-NG and VL-e. We wish to thank SURFnet for their continuous support, in particular Roeland Nuijts, Bram Peeters and Antoon Prins.

References

[1] J. Barnat, L. Brim, and I. Černá. Cluster-Based LTL Model Checking of Large Systems. In *Proc. Formal Methods for Components and Objects*, volume 4111 of *LNCS*, pages 259–279, 2006.

[2] R. Boutaba, W. Golab, Y. Iraqi, T. Li, and B. S. Arnaud. Grid-Controlled Lightpaths for High Performance Grid Applications. *Journal of Grid Computing*, 4:387–394, 2003.

[3] CineGrid project. http://www.cinegrid.org.

[4] The First International Data Analysis Challenge for Finding Supernovae, in conjunction with IEEE Cluster/Grid 2008. http://www.cluster2008.org/challenge.

[5] The Distributed ASCI Supercomputer 3 (DAS-3). http://www.cs.vu.nl/das3.

[6] T. DeFanti, C. de Laat, J. Mambretti, K. Neggers, and B. S. Arnaud. TransLight: A Global-Scale LambdaGrid for E-Science. *Commun. ACM*, 46(11):34–41, Nov. 2003.

[7] EnLightened project. http://www.enlightenedcomputing.org.

[8] E. Escalona et al. Deployment and Interoperability of the Phosphorus Grid Enabled GMPLS (G2MPLS) Control Plane. In *Proc. CCGRID'08*, Lyon, France, May 2008.

[9] S. Figueira et al. DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks. In *Proc. CCGrid'04*, pages 707–714, Washington, DC, USA, 2004.

[10] G-lambda project. http://www.g-lambda.net.

[11] R. L. Graham, G. M. Shipman, B. W. Barrett, R. H. Castain, G. Bosilca, and A. Lumsdaine. Open MPI: A Flexible High Performance MPI. In *Proc. 6th Int. Conf. Par. Proc. and Appl. Math.*, pages 228–239, Poznan, Poland, Sept. 2005.

[12] P. Grosso, D. Marchal, J. Maassen, E. Bernier, L. Xu, and C. de Laat. Dynamic Photonic Lightpaths in the StarPlane Network. *Future Gener. Comput. Syst.*, 25(2):132–136, Feb. 2009.

[13] M. Harvey, S. Jha, M. Thyveetil, and P. Coveney. Using Lambda Networks to Enhance Performance of Interactive Large Simulations. In *Proc. E-Science'06*, Washington, DC, USA, 2006. IEEE Comp. Soc.

[14] E. He, X. Wang, and J. Leigh. A Flexible Advance Reservation Model for Multi-Domain WDM Optical Networks. In *BROADNETS*. IEEE, 2006.

[15] The Ibis Project. http://www.cs.vu.nl/ibis.

[16] T. Lehman, J. Sobieski, and B. Jabbari. DRAGON: a Framework for Service Provisioning in Heterogeneous Grid Networks. *IEEE Comm. Mag.*, 44(3):84–90, Mar. 2006.

[17] J. Mambretti, R. Gold, F. Yeh, and J. Chen. AMROEBA: Computational Astrophysics Modeling Enabled by Dynamic Lambda Switching. *Future Gener. Comput. Syst.*, 22(8):949–954, 2006.

[18] Myricom. http://www.myricom.com.

[19] R. Pelánek. BEEM: Benchmarks for Explicit Model Checkers. In *Proc. SPIN Workshop*, volume 4595 of *LNCS*, pages 263–267. Springer, 2007.

[20] Phosphorus project. http://www.ist-phosphorus.eu.

[21] T. Shimizu et al. International Real-time Streaming of 4K Digital Cinema. *Future Gener. Comput. Syst.*, 22(8):929–939, 2006.

[22] L. L. Smarr, A. A. Chien, T. DeFanti, J. Leigh, and P. M. Papadopoulos. The OptIPuter. *Commun. ACM*, 46(11):58–67, 2003.

[23] StarPlane project. http://www.starplane.org.

[24] The Subaru Telescope. http://subarutelescope.org.

[25] SURFnet. http://www.surfnet.nl.

[26] A. Takefusa et al. G-lambda: Coordination of a Grid scheduler and lambda path service over GMPLS. *Future Gener. Comput. Syst.*, 22:868–875, 2006.

[27] R. V. van Nieuwpoort, T. Kielmann, and H. E. Bal. User-Friendly and Reliable Grid Computing Based on Imperfect Middleware. In *Proc. of the ACM/IEEE Conf. on Supercomputing (SC'07)*, Nov. 2007.

[28] R. V. van Nieuwpoort, J. Maassen, G. Wrzesinska, R. Hofman, C. Jacobs, T. Kielmann, and H. E. Bal. Ibis: a Flexible and Efficient Java based Grid Programming Environment. *Concurrency and Computation: Practice and Experience*, 17(7-8):1079–1107, June 2005.

[29] K. Verstoep, H. Bal, J. Barnat, and L. Brim. Efficient Large-Scale Model Checking. Proc. IEEE Int. Par. and Distr. Proc. Symp., IPDPS'09, 2009.

[30] K. Verstoep, J. Maassen, H. E. Bal, and J. W. Romein. Experiences with Fine-grained Distributed Supercomputing on a 10G Testbed. In *Proc. CCGRID'08*, Lyon, France, May 2008.