An OBS-based Grid Architecture

M. De Leenheer*, E. Van Breusegem*, P. Thysebaert*, B. Volckaert*,
F. De Turck*, B. Dhoedt*, P. Demeester*, D. Simeonidou†,
M. J. O' Mahoney†, R. Nejabati†, A. Tzanakaki‡, I. Tomkos‡
*Ghent University - IMEC, Gent, Belgium
†Essex University, Essex, UK
‡AIT, Athens, Greece

***Abstract*— Grids offer a uniform interface to a distributed collection of heterogeneous computational, storage and network resources. Most current operational Grids are dedicated to a limited set of computationally and/or data intensive scientific problems. The de facto modus operandi is one where users submit job requests to a *Grid Portal*, acting as an interface to the Grid's management system, which in turn negotiates with 'dumb' resources (computing and storage elements, network links) and arranges for the jobs to be executed. In this paper, we present a new Grid architecture featuring generic application support, direct user access (Grid-to-the-home) and decentralized scheduling intelligence in the network. We show how Optical Burst Switching (OBS) enables these features while offering the necessary network flexibility demanded by future Grid applications.**

I. Introduction

Over the past few years it has become evident that local computational resources cannot keep up with the ever-increasing demand for processing power. The solution to this problem came in the form of distributed computing, aggregating the power of a multitude of computational resources in one big Grid. This Grid is named after the analogy with the electricity grid, and provides users with on-demand resource usage. Recent advances in optical networking technology increase the feasibility of global widespread Grid adoption.

Grids, as opposed to cluster computing efforts, consist of geographically distributed heterogeneous resources (computing, storage, data, sensors, etc.), interconnected through a variety of local area/wide area networks and offer a uniform interface to these resources (i.e. a Grid user can submit jobs to the Grid just as if he/she was handling a large virtual supercomputer). Grids also differ from clusters in that they do not have a central administration point. Instead, they consist of resources across multiple administrative domains.

Grid activities fall into several categories [1], each having unique architectural characteristics; here, we will briefly highlight two of them and illustrate their deficiencies when compared to our proposed architecture.

The first is high performance computing, used for processing both computationally intensive jobs and parameter sweep applications. Examples of such jobs include applications used for Very Long Baseline Interferometry (VLBI), High Energy Physics (HEP), and High Performance Computing and Visualization [2], typically requiring massive bandwidth (constant streams of Gbit/s and up), storage ($>>$ TB storage) and computational ($>>$ TFlops) resources. The infrastructure used in these Grids commonly includes performant computing elements (clusters and supercomputers) and networking technologies (Gigabit Ethernet and optical technologies). These Grids are usually dedicated to a limited number of applications, and fairly static in configuration (using dedicated inter-site connections e.g. long-lived wavelength paths in an optical circuit-switched network). Access to the Grid is usually provided by a user interface dubbed the Grid *portal*, where jobs (or rather: job descriptions) are submitted. The different resources are queried by a scheduling entity, which is responsible for allocating an appropriate resource set to each job. In order to scale with increasing number of resources, the scheduler has only a limited view over these resources, either by operating on aggregated data or by considering a subset of available resources.

A second category comprises the so-called cycle-stealing Grids. These Grids are similar to peer-to-peer applications and attempt to harvest unused CPU cycles from a variety of geographically distributed providers (e.g. home user desktop computers), and are usually deployed for processing large parameter sweep applications (e.g. SETI@HOME, United Devices Grid). Clearly, these Grids' infrastructure mainly consists of commodity equipment, as computing power is offered by desktop PCs and network connectivity is provided by the Internet, making this type of Grid unsuited for applications with a high communication to computation ratio. However, as various generic peer-to-peer application frameworks [3] become available, a uniform interface to the processing power supplied by heterogeneous distributed resources can be offered, ultimately resulting in a wider variety of applications that can readily be *Gridified* as cycle-stealing applications. Work units for these applications are actively retrieved by each computing resource from a centralized *task farm*.

In this paper, we propose a new Grid architecture seeking to combine the strenghts of both Grid types mentioned above; providing the general public with Grid technology (meaning direct user access and generic application support), while at the same time offering enough network and computational resources to support future data and processing intensive end user applications. At the same time, we seek to realize a fully decentralized job scheduling mechanism by assigning a certain level of intelligence to the different resources.

This paper continues as follows: in section II we describe

why current network solutions are unsuitable for widespread Grid deployment. We continue in section III by elaborating on the components and functionality of a Grid architecture meeting the requirements set in the previous section, and then explain in section IV why the use of OBS as a basis for this architecture would be a natural choice. Finally, in section V we present our conclusions.

II. Future Grid Applications: requirements and impact

For the average home user today the network cannot sustain Grid computing. With a home access bandwidth of only a few Mbps, to at most 100 Mbps download speeds, and an order of magnitude smaller upload speeds, transmission of jobs would simply take too long. However, if the current trend holds and bandwidth availability (doubling each year) keeps growing faster than the computing power (at most doubling every 18 months) of an average end user, tapping into the Grid at home becomes viable.

Let us assume that in such a future Grid, home users are connected through a symmetrical access link offering a bandwidth of about 2.5 Gbps (in the optical range). While this kind of bandwidth is certainly not readily available to end users at the time of writing, extrapolation of past trends shows that in about 20 years such an evolution can be expected. In analogy, if computational capacity doubles every 18 months, an increase in high-end desktop PC performance with a factor in the order of magnitude 2^{10} should be envisaged.

The resulting processing power will offer the possibility to process extremely demanding applications (at least by today's standards) on an ordinary desktop PC. However, as we show below, it is reasonable to assume that application demands will experience a similar increase in their requirements, making it unfeasible to execute them locally. The needed aggregate power for these applications is drawn from the Grid, offering idle end user resources (most desktop computers have a low average processing load) and dedicated computing farms offered by commercial providers (with a processing power comparable to that of hundreds or thousands of desktop PCs). This means that in this future Grid, a large user base will have direct access to a vast pool of shared resources as access bandwidth catches up with processing power.

In what follows we present some typical application requirements and their impact on the underlying Grid system, indicating that existing Grid infrastructures will fail to cater for their needs. A first application example comes from the area of multimedia editing; video editing applications are widely adopted, and allow users to manipulate video clips, add effects, restore films etc. Advances in recording, visualization and video effects technology will demand more computational and storage capacity, especially if the editing is to be performed within a reasonable time frame (e.g. allowing user feedback).

More specifically, 1080p High Definition Television (HDTV) [4] offers a resolution of 1920x1080, which amounts to about 2 MPixel per frame. If we suppose that applying an effect requires 10 floating point operations per pixel per frame, and if the user would like to evaluate the effect of 10 different options, then processing a 10 second clip (25 fps) already requires over 50 GFlop. This will take about 0.5 s to complete locally (we assume local processing power is 100 GFlops), while execution on a provider's resource should only take 5 ms (assuming the capacity of providers is a factor 100 higher). Transmission time of 10 s of compressed HDTV video (bitrate 20 Mbit/s or a 25 MB filesize) on a 2.5 Gbit/s access link is 80ms. While the 2.5 Gbps is likely to be realized through optical technologies, it is unfeasible to assume that each end user is allowed to set up end-to-end wavelength paths for each multimedia editing operation. Indeed, even if wavelength path set-up times were to decrease sharply (currently in the range of 100 ms), the use of OCS would waste a significant amount of (network) resources and one would have to devise a mechanism able to handle path set-up and tear-down requests from vast amounts of users.

A second application example is the online visualization of (and interaction with) a virtual environment. Virtual environments are typically made up of various objects, described by their shape, size, location, etc. Also, different textures are applied on these objects. A user should not only be able to visualize selected scenes in the environment by adjusting his viewing angle, but should also be able to interact with the rendered objects. Usually the description of a scene can be realized in limited storage space, the size of a texture being limited to a few kilobytes. It follows that a scene can be stored in a rather small storage space, typically around a few Megabytes. However, rendering the scene is a different problem altogether: if we demand a performance of 300 million polygons per second, computational capacities as large as 10000 GFlops are required [5]. Clearly, the rendering of these scenes, preferably in real-time, is unfeasible using only local resources. Suppose a user has at its disposal an archive of different scene description, with a requested frame rate of 25 frames per second. This amounts to a latency smaller than 40 ms between the submission of the scene description, and the actual displaying of the scene. Assuming a scene is 2.5 MB in size, we obtain a transmission time of only 8 ms; this leaves us with about 30 ms for processing and retransmission of the final rendering, which should be possible with the given capacities of the (local) resource providers. Considering the delay associated with setting up an optical circuit, OCS can only be used when a user employs the same resource, hereby severely limiting the flexibility of the Grid concept. On the other hand, the lack of adequate QoS in the standard IP protocol makes it near-impossible to meet the real-time constraints.

The above analysis leads to some important observations:

- Building a dedicated network for each high bandwidth and computationally intensive application is economically unsound.
- A large user base and a variety of applications give rise to highly unpredictable Grid service requests. This means that, basically, a dedicated static infrastructure is not the most optimal solution.

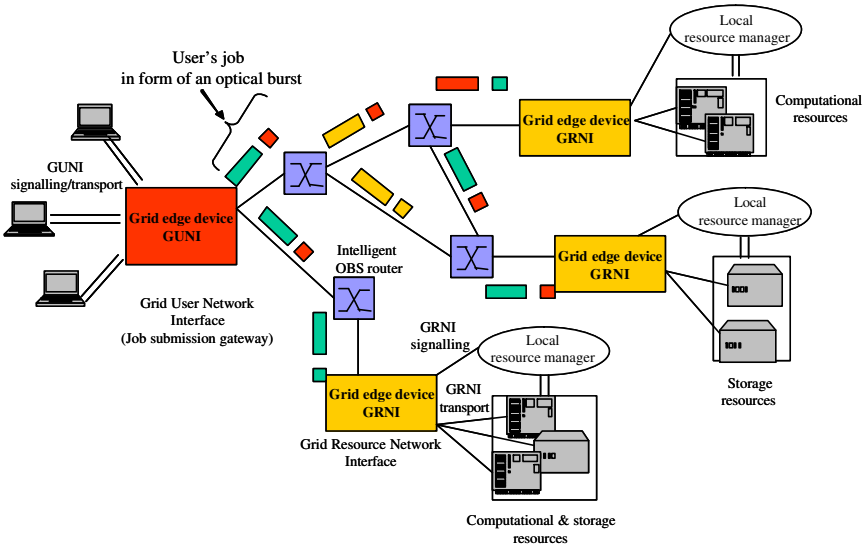


Fig. 1. Generic view of the proposed OBS architecture

- The emergence of data intensive end-user applications implies that these users will need high access bandwidths, making the case for optical technologies, as they are used in dedicated network infrastructure for computational Grids today.
- In many cases transmission times (i.e. job sizes) will be rather short (few 100 us to tens of ms). This means that using OCS end to end will prove to be too wasteful, as holding times of wavelength paths will be too small compared to their setup times.
- At the same time, for real time applications, waiting for a scheduler to queue jobs and allocate resources to them simply takes too long. Low response times can only be achieved using highly decentralized scheduling intelligence in a high-bandwidth network.

These points make the current network solutions (OCS for computational Grids, the Internet for peer-to-peer applications) unsuited for providing Grid access to everyone. The dedicated infrastructure will be too wasteful and inflexible, and a classical queueing/scheduling based architecture with electronic management of Grid resources may prove unsuited for real-time applications. To overcome these problems, a new infrastructure will be required for future Grids. To that end, we present an OBS based architecture which copes with the observations and challenges stated above. We refer the reader to [6] for further details about the Optical Burst Switching technique.

III. Proposed OBS architecture

Fig. 1 gives an overview of the important concepts of our architecture, with the optical network interconnecting intelligent OBS routers at its heart. In the rest of this section we detail

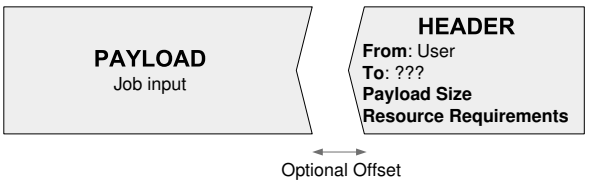
- how bursts travel to their destination
- what a burst contains
- how to address the asymmetry in user-Grid and Grid-user traffic.

A. *Travelling bursts: the principle of* anycast

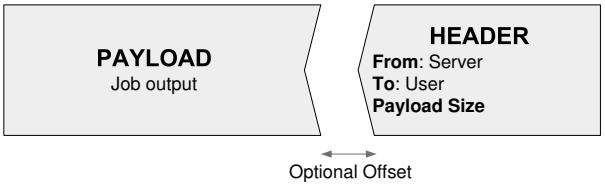
A major difference between a Grid request and Internet (e.g. HTTP over TCP/IPv4) traffic is the absence of a predetermined destination in the former case. Indeed, a Grid user is merely interested in the successful completion of his submitted requests within the specified requirements. The exact resources handling his requests are of no importance; in fact, *any* resource set able to offer the requested service is acceptable (as far as the user is concerned). A user which has direct access to a Grid may therefore benefit from a network in which his requests travel according to an *anycast* paradigm ([7], [8], [9]). Freenet [10] is an example implementation of this paradigm in the application layer: it is a peer-to-peer information sharing client which does not reveal exact content location to interested users. Our proposed architecture adheres to the anycast principle by coding Grid requests into optical bursts (see below), and by introducing intelligence in the OBS routers in order to lead each burst to a suitable destination.

B. *Job coding in optical bursts*

A user realizing that some of his computing jobs cannot be handled by his local desktop computer, may want to make use of the Grid to accelerate job processing. Ideally, he wants to anycast those jobs into the Grid and let them be processed on remote resources satisfying the jobs' demands. In order to achieve this, a job (together with its input data from the submission site) is encapsulated in an optical burst and accompanied by a header containing the job's soft and hard requirements (e.g. deadline, necessary storage, policy, etc.), as shown in Fig. 2. Fitting each job into a single burst is an important design decision (simplifying transport and network operation) which will be discussed in detail later (in case of



(a) Job encoded in optical burst



(b) Returning traffic encoded in optical burst

Fig. 2. Burst contents

handling very large non-parallizable jobs exceptions to this rule will have to be made). At the user's end, a software component may aid that user in splitting big requests into independent constituents (each filling a single burst).

Since we are dealing with an anycast situation, no destination address is needed. The burst is simply sent out into the OBS network. This burst then travels along a link, with the intermediate routers not being notified in advance of its arrival, much like JIT [11] or JET based schemes [6]. When this burst arrives at an intermediate router, it then decides on the fly where to forward the burst to, based on information contained in the preceding header (see Fig. 2) augmented with the limited amount of aggregate (i.e. non-detailed) resource load information available to the intermediate router. Examples of such information are link load, delay requirements, estimated free computing power or storage space in a certain outgoing direction, and estimated required computing time or storage space. This information is used to push the burst closer to (what seems) a suitable destination. Because each router has limited load information available, this forwarding can be handled very quickly. The end user doesn't specify the network location where the job will be processed, and hence it is implicitly scheduled through its itinerary in the network. This makes the Grid scheduling process completely distributed and thus more scalable and robust, while transforming the resource allocation into the earlier mentioned anycast problem (no preset destination).

Ultimately, every burst is bound to arrive at an edge router (connecting to a Grid processing site) which is supposedly able to fulfill the job's requirements. If this edge resource is indeed capable of processing the job, the job has arrived at its destination; if not, then a *deflection* mechanism reposts the job into the OBS network.

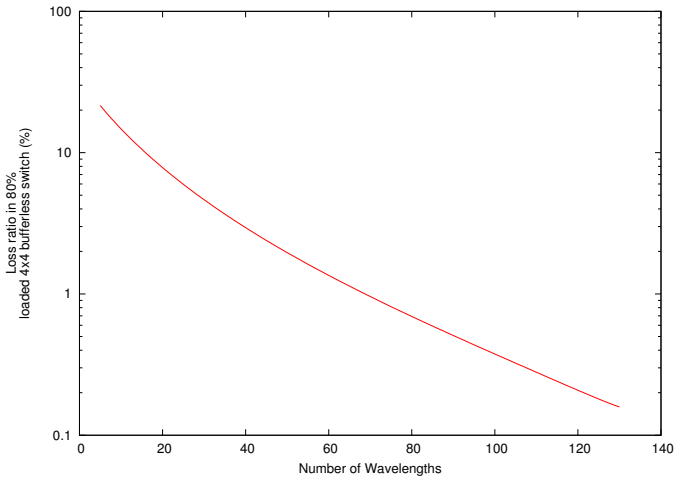


Fig. 3. Evolution of blocking behaviour as a function of wavelengths

C. Traffic asymmetry

The previous section has primarily dealt with traffic originating at the user's end (i.e. job submission). This traffic, as we have explained, is routed in the network following an anycast paradigm. However, while the exact processing site may be of no importance to the user, the destination where computed results (e.g. the bitmap of a rendering job) should be forwarded to most certainly is! More often than not, these results should be sent back to a specific address (e.g. the user's site itself) and cannot be 'anycast' into the OBS network. This implies that the 'traditional' forwarding mechanisms need to be employed. Thus, optical bursts on the return path need to be tagged with an explicit destination address and be delivered to that address over the OBS network.

IV. OBS-technology advantages

In the above sections, we have shown how the use of OBS in the Grid network arises naturally from the requirements and preferences in Grid problem domains. Below, we give a concise summary of the incentives to use OBS as our Grid-enabling infrastructure.

A. Native job-burst mapping

As we mentioned earlier, a single job will be mapped into one burst. This is a very straightforward and elegant solution, as a typical job will be in the order of magnitude of a few (tens) of megabyte, which at 2.5 Gb/s can be translated in a burst ranging from a few hunderd us to several ms. The entire job either gets through, or not at all, which means the transport network can be simplified. We do not consider segmentation [12] of individual bursts (note that applications may consist of multiple independent tasks, each of which can be mapped to such a burst). This eliminates the out of order delivery (and reassembly) problem, and allows for simple flow control mechanisms. This means that assured delivery of a job (which is now a single burst) becomes easier. Note, however, that - due to the anycast principle - TCP is no longer possible

as reliable transport protocol, as there is no known destination to setup a connection with.

B. Bufferless operation

A classical problem encountered in OBS and OPS is the highly complex buffering, usually in the form of bulky fiber delay lines. These buffers are necessary as otherwise througput of the switch is rather low. As Fig. 3 however clearly shows, the problem becomes smaller with an increasing wavelength count. It shows the blocking of a single node simulation; the loss as a function of the load is depicted for a four fibre by four fibre switch without contention buffers. The simulation assumes uniformly distributed traffic employing Poisson arrival processes of $100us$ long bursts (more details can be found in [13]). It doesn't consider segmenting (which is fine considering we want a job retained in its entirety) nor does it consider deflection routing. As the figure shows, as soon as more than 100 wavelengths in one direction are reached (which may or may not be carried by multiple fibers), an effective throughput of 80% can be reached (loss rate $< 1\%$). Note that this doesn't consider deflection routing, which may lower the loss another order of magnitude. Thus, considering that the network may require hundreds of wavelengths in the future, bufferless operation may become an attractive option.

C. Separate control

A classical argument for OBS is that, by decoupling header and payload, the processing delay line can be removed. If, in addition, buffers aren't needed (as suggested in the previous section), delay lines are no longer needed at all. At the same time, dedicating few wavelengths for headers only (i.e. out of band signaling) may prove to be very effective to reduce costly O/E/O conversions. Combined with deflection routing and the principle of anycast, a wider range of timing inaccuracies can probably be tolerated.

D. Absence of electronic bottleneck

As electronics evolve slower than optics in terms of speed (the gap between the two widens continuously), it is beneficial to perform as much of the routing and forwarding as possible in the optical layer. This is exactly what happens in our intelligent OBS routers: electronics are only used for what they excel at, i.e. intelligent decisions. The high speed (end to end) data transport is completely left to the optics.

E. Low set-up time

The use of OBS eliminates the overhead caused by end to end connection setups (which occurs in circuit switched networks). Since no set-ups are needed, response time (as experienced by the users) can be decreased drastically which may be a hard requirement for future applications.

V. Conclusion

As access bandwidth increases, it will become easier for end users to plug in to the Grid (either donating their resources for others to use, or using the aggregated Grid resource power for their own purposes). We showed through some application examples that currently deployed network infrastructures are not future proof for interconnecting resources and users in a large scale all-purpose Grid. We proposed a new Grid architecture built on OBS, which proves to be a natural solution to the encountered limitations of current Grid deployments. Native job-burst mapping, the ability for bufferless operation, the absence of an electronic bottleneck and low set-up times are but some of the added benefits of using OBS.

Acknowledgment

This work was partly funded by the european commission through IST projects NOBEL, as well as e-Photon/ONe. The flemish government also providing partly funding through the IWT-GBOU project 010058 "Optical Networking and Node Architectures". E. Van Breusegem and B. Volckaert would like to thank the IWT for financial support through their Ph. D. grant. P. Thysebaert and F. De Turck are Research Assistant and Postdoctoral Fellow of the Fund for Scientific Research - Flanders (F.W.O.-V), respectively.

References

[1] K. Krauter, R. Buyya, M. Maheswaran,*A Taxonomy and Survey of Grid Resource Management Systems*, International Journal of Software: Practice and Experience (SPE), 32(2):135–164, 2002

[2] D. Simeonidou (ed.) et al., *Optical Network Infrastructure for Grid*, Grid Forum Draft, Sept. 2003, http: /projects/ghpn-rg/document/draft-ggf-ghpn-opticalnets-1/en/1

[3] Project JXTA, website, `http://www.jxta.org/`

[4] Li Zong and Nikolaos G. Bourbakis, *Digital Video and Digital TV: A Comparison and the Future Directions*, Proc. of the International Conference on Information Intelligence and Systems, Mar 1999.

[5] Tsuneo Ikedo, *Creating Realistic Scenes in Future Multimedia Systems*, IEEE MultiMedia, 9(4):56–72, Oct 2002.

[6] C. Qiao, M. Yoo, *Optical Burst Switching - A new Paradigm for an Optical Internet*, Journal of High Speed Networks, Spec. Iss. On Optical Networking, vol. 8, no. 1, Jan. 2000, pp. 36-44

[7] C. Partridge, T. Mendez, W. Milliken, *Host Anycasting Service*, IETF RFC 1546, Nov 1993

[8] S. Bhattacharjee, M.H. Ammar, E. Zegura, V. Shah, Z. Fei, *Application-Layer Anycasting*, Proc. of IEEE Infocom'97, Apr 1997

[9] E. Basturk, R. Engel, R. Haas, V. Peris, D. Saha, *Using Network Layer Anycast for Load Distribution in the Internet*, IBM Research Report (RC 20938), Jul 1997

[10] I. Clarke et al., *Protecting Free Expression Online with Freenet*, IEEE Internet Computing, Jan - Feb 2002, pp. 40-49

[11] J.Y. Wei, J.L. Pastor, R.S. Ramamurthy, Y. Tsai, *Just-in-time optical burst switching for multi-wavelength networks*, 5th Int. Conf. on Broadband Comm. (BC'99), 1999. 339–352

[12] V. M. Vokkrane, J. P. Jue, S. Sitaraman, *Burst Segmentation: An Approach for reducing burst loss in optical burst switched networks*, IEEE ICC '02, New York, NY, April 2002

[13] E. Van Breusegem, J. Cheyns, B. Lannoo, A. Ackaert, M. Pickavet, P. Demeester, *Implications of using offsets in all-optical packet switched networks*, ONDM 2003, Budapest, Hungary, february 2003