Scalable Wall-Socket Multimedia Grid Computing

F.J. Seinstra, N. Drost, R. Kemp, J. Maassen, R.V. van Nieuwpoort, K. Verstoep, and H.E. Bal
Vrije Universiteit Amsterdam, Faculty of Sciences,
De Boelelaan 1081a, 1081 HV, Amsterdam, The Netherlands
{fjseins, ndrost, jason, rob, versto, bal}@cs.vu.nl

Abstract

Multimedia data is rapidly gaining importance along with recent deployment of publicly accessible digital television archives, and surveillance cameras in public locations. In a few years, analyzing the content of multimedia data will be a problem of phenomenal proportions, as digital video may produce high data rates, and multimedia archives steadily run into Petabytes of storage space. Consequently, for urgent problems in multimedia content analysis, Grid computing is rapidly becoming indispensable.

For the last decade, the easy *and* efficient *use of distributed resources (ultimately at a world-wide scale) has been the foremost visionary aim (or 'promise') of the field of Grid computing. Today, with the advent of easy-to-use programming models and run-time systems, as developed by this proposal's team members, Grids indeed are maturing towards a viable commodity for non-experts in high-performance computing. Moreover, our current results indicate that, particularly with the increasing deployment of high-speed optical networks, efficient world-wide execution of massively communicating distributed applications is achievable. In other words, realization of the 'promise of the Grid' is now finally within reach.*

This proposal aims to realize easy, efficient, and scalable distributed supercomputing for the multimedia domain. It does so by integrating several solutions that are designed to remove the intrinsic difficulties of large-scale employment of Grids. Specifically, we aim to develop an application in which a digital camera (potentially as part of a robot system) is capable of real-time 'recognition' of objects from a set of learned objects, while being connected to a large-scale Grid system comprising of cluster computers located in Europe, and potentially even world-wide. Apart from the appealing nature of a demonstration including visual information (i.e., the camera data) and speech (i.e., the real-time reaction of the system), we aim to show true Wall-Socket Grid Computing *— meaning that the application is being compiled and started on a local desktop machine, with Grid resources being employed entirely transparently.*

1 Application domain and technologies

This is a time of transition in which information is more and more composed of *multimedia items*, i.e. a combination of pictorial, textual, and auditory data. The computerized access to the content of such information is generally recognized as a tremendous challenge [6]. This is because the automatic deduction of semantics from multimedia data requires sophisticated techniques for data structuring, transformation, analysis, classification, and learning. In particular, due to the nature of the data, the challenge is to discover and interpret tiny fractions of useful information in a whirlwind of meaningless noise. Given the increasing storage and connectivity of multimedia data, automatic multimedia content analysis (MMCA) is becoming an ever more important research area.

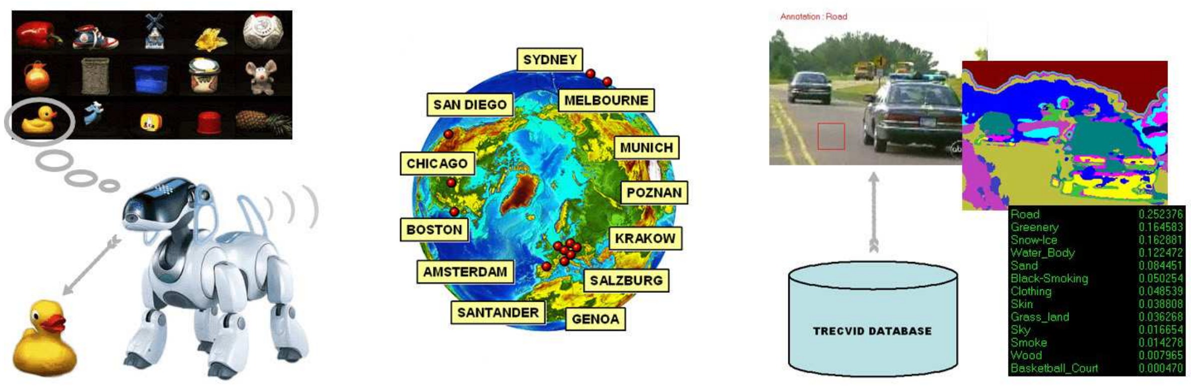
Multimedia content analysis considers all aspects of the automated extraction of new knowledge from multimedia data streams and archives. This proposal concentrates on image and video, as these contain the bulk of all data. Research in this area has made a giant leap forward with the introduction of machine learned multimedia analysis, yielding automatic categorization by visual object types, such as human faces, cars, interviews, etcetera [7]. Fundamental MMCA research questions include:

- Can we automatically find genres in images from a statistical evaluation of large image sets?
- Can we learn to find objects in images and video streams from partially annotated image sets?

Scientifically, these questions deal with the philosophical foundations of cognition and the naming of things. Practically, solutions are urgently needed given the increasing volume of multimedia data.

1.1 Color-based object recognition

The path to finding computerized MMCA solutions that can compete with the visual capabilities of the human brain is littered with many fundamental problems. One such problem — which is the focus of this proposal — is that of



```
Road             0.252376
Greenery         0.164583
Snow-Ice         0.162881
Water_Body       0.122472
Sand             0.084451
Black-Smoking    0.050254
Clothing         0.048539
Skin             0.038808
Grass_land       0.036268
Sky              0.016654
Smoke            0.014278
Wood             0.007965
Basketball_Court 0.000470
```

Figure 1. *Real-time (left) and off-line (right) distributed multimedia computing on a world-wide scale. The real-time application constitutes a visual object recognition task performed by a robot dog, for which we obtained a 'most visionary research award' at AAAI 2007. It is one of the few known instances of a real-time task using massively communicating resources world-wide. The off-line application constitutes our winning TRECVID system, for which we obtained a 'best technical demo award' at ACM Multimedia 2005.*

automatic *object recognition*. The problem of object recognition is to determine which, if any, of a given repository of objects appears in an image or video stream. It is a computationally demanding problem that involves a non-trivial tradeoff between specificity of recognition (e.g., discriminating between different faces) and invariance (e.g., to different lighting conditions).

Color is a powerful cue in the recognition of objects. Recognition based on color, rather than just intensity, provides a broader class of discrimination between objects. The use of RGB values, however, does not directly increase recognition performance, certainly not when variations in imaging conditions are encountered. Differences in intensity, direction, and color of the illumination, as well as shading and cast-shadow significantly effect the appearance of an object. Therefore, it is meaningful to transform the RGB values to invariant properties, which relate to surface properties rather than to object appearance. Many invariants have been derived [2] which are known to be robust under noisy conditions. These invariants can be scaled to the size of the object structure. Recognition with these invariants boils down to learning an invariant representation of the object, rather than learning every possible appearance of a single view of the object.

A solution to our problem is obtained by decomposing the recognition of object *appearance* into two schemes. First, we have different views or aspects of an object, each of which has to be learned. Secondly, there is the illumination, drastically influencing object appearance. As stated above, for this class of appearance effects we can apply invariants effectively. Object recognition may be based on a weak description of the important features in the scene, as long as mutual correspondence between observation and objects in the world is maintained. Therefore, a solution is obtained by learning local histograms of invariant features for each aspect of an object. Due to the rapid increase in the size of multimedia repositories consisting of 'known' objects, and the inherently high computational costs for 'deep' inspection of the images or video frames at hand, state-of-the-art sequential computers no longer suffice. As a result, to obtain a working solution distributed superomputing on large collections of clusters is indispensible.

1.2 Distributed supercomputing and MMCA

In our earlier work [5] we have convincingly shown that Grid-based distributed supercomputing, involving hundreds of massively communicating compute resources covering our entire globe, can bring efficient solutions for off-line applications and (soft) real-time problems in the multimedia domain (see Figure 1). This work has brought decisive advantages in the international TRECVID evaluation for content-based video retrieval [5], and earned us awards at high-profile international conferences in multimedia computing and artificial intelligence. Despite these successes, one fundamental problem remains: our earlier approach to Grid execution requires thorough understanding of Grid technologies, far beyond the level of expertise of multimedia researchers. Clearly, there is an urgent need for solutions that can shield non-expert users from the intrinsic complexities of Grid programming and execution. The foremost research question that underlies this proposal, therefore, is stated as follows: *How, or to what extent, can state of the art Grid computing technologies be integrated to adhere to the specific needs of researchers and application developers in the MMCA domain?*

1.3 The ’Promise of the Grid’

In the year 2001, Grid experts Ian Foster, Carl Kesselman, and Steven Tuecke published one of the most influential and defining papers in the field of Grid Computing [1]. The authors indicate that the fundamental problem that underlies the Grid concept is ”flexible, secure, coordinated resource sharing and problem solving in dynamic collections of individuals, institutions, and resources”. Also, the field is ”distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation”. With ”Grid-enabled programming systems” available that ”enable familiar programming models to be used”, this definition holds a promise, i.e. *the promise of efficient and easy-to-use wall-socket computing over a distributed set of resources*.

From the remarks made by Foster et al., it follows that Grid-based programming models must adhere to the following general requirements that underlie ’the promise of the Grid’:

- Transparency — the shielding of inherent Grid-related complexities, to enable ease of programming and execution by non-expert users.
- Efficiency — obtaining runtime performance approaching the maximum capabilities of available Grid resources (which is what we refer to as *distributed supercomputing*).

For full transparency, the user must be shielded from all issues that complicate the programming and use of Grid systems in comparison with the programming and use of a standard desktop computer. To this end, methodologies must be available that provide transparent support for:

- Platform-independence — In large-scale Grid systems heterogeneity is omnipresent, to the effect that applications designed for one system are generally guaranteed to fail on others. This problem must be removed by hiding the physical characteristics of resources from users.
- Middleware-independence — Today, Grid programmers must write their applications using a middleware system that abstracts away part of the Grid’s complexities. This is a daunting task, as the many available Grid middlewares change frequently, are low-level, unstable, and incomplete [4]. For example, Globus significantly changed its interface over the last three major releases, each time exposing its enabling technology in detail. Whilst the research underlying these technologies is essential, such details severely complicate Grid usage. A single high-level interface on top of multiple Grid middleware systems is therefore essential.
- Connectivity — Getting distributed applications to execute at all in a Grid system is difficult. This is because firewalls, transparent renaming of IP addresses, and multi-homing (machines with multiple addresses), can severely complicate or limit the ability of resources to communicate. Moreover, in many cases no direct connection with certain machines is allowed at all. Despite solutions that have been provided for firewall issues (e.g., NetIbis, Remus), integrated solutions must be made available that remove connectivity problems altogether.
- Fault-tolerance and malleability — Grids are *open world* and *faulty*, meaning that resources can be added and removed at will, and can crash at any moment. These issues are particularly prominent in Grids consisting of hundreds or thousands of machines (or more). Hence, for robust Grid execution, users must be given solutions that make programs fault-tolerant and malleable, such that processors can be added and removed at application run-time.
- Parallelization — For execution on clusters or Grids it is generally up to the programmer to identify the available parallelism in a problem at hand. For the programmer — generally a domain-expert (i.e. multimedia researcher) with limited or no expertise in distributed supercomputing — this is often an insurmountable problem. Clearly, automatic parallelization tools must be made available that hide the inherent complexities of parallelization.

A set of methodologies that adheres to all of these transparency requirements — in turn — must allow distributed applications to run efficiently in a Grid environment. More specifically, execution performance must compare well with, or even be identical to, the performance of non-user transparent tools that provide the same or similar functionality. Also, as transparency generally follows from high level abstractions on top of lower level functionality, the abstraction overhead costs should be kept to a minimum.

1.4 Integration of methodologies

In this Scalable Computing Challenge proposal we claim that for the abovementioned requirements we have working methodologies available. As a consequence, a working solution to our object recognition problem is obtained by coordinated integration of these methodologies.

The methodologies, for the large part developed by ourselves, are the following:

- Platform-independence: **Java** — Today’s approach to abstracting computer resources is referred to as virtualization. The technique has been applied in several application environments (e.g., VMware, .NET) and languages (e.g., Java, Perl, Python). From these, Java seems to have become the most widely adopted solution in Grid computing. For example, the Globus Toolkit 4 release is entirely implemented in Java. Given the fact that Java often provides competitive performance in comparison with C/C++, our efforts are entirely Java-based.

- Middleware-independence: **JavaGAT** — The Java Grid Application Toolkit (JavaGAT [9]) offers high-level primitives to Grid access independent of the middleware that implements this functionality (e.g., Globus, gLite, Unicore). Using JavaGAT, Grid applications can, among other functionality, transparently access remote data and spawn off jobs, allowing users to focus on the actual computational problem to be solved. The JavaGAT middleware abstraction overhead has been shown to be close to zero [9].

- Connectivity: **SmartSockets** — The SmartSockets library [3] automatically solves Grid connectivity problems, a.o. by way of port forwarding, and TCP splicing. The power of the approach was demonstrated in an experiment consisting of 30 realistic connectivity scenarios in which SmartSockets was always capable of establishing a connection, while traditional sockets only worked in 6 of these [3].

- Fault-tolerance and malleability: **IPL** — The Ibis Portability Layer (IPL) [8] is a Grid programming system that combines flexible treatment of dynamically available resources with competitive communication performance (outperforming Sun RMI and even C/MPI transfer times for complex data structures). Most importantly, IPL supports *abstract addressing*, a globally unique and protocol-independent identification scheme to distinguish between participating resources, and *membership notification*, or global, adaptable, information regarding the dynamic pool of participating resources. IPL provides these properties through a high-level API that can be applied to develop higher level extensions and programming models that provide full support for malleability (e.g., Satin [8]).

- Parallelization: **Parallel-Horus** — Parallel-Horus [5] is a cluster programming library that allows its users to implement parallel multimedia applications as fully sequential programs, using a carefully designed set of building block operations. The library hides all complexities of parallelization behind a fully sequential interface. For reasons of efficiency, Parallel-Horus adapts to the performance characteristics of a cluster at hand. Parallel-Horus further applies runtime communication minimization with close to zero overhead.

2 Application scenario

We aim to perform object recognition by using a video camera (potentially as part of a robot system), and a large collection of small objects (e.g., cups, bottles, pencils, etcetera). Each video frame, obtained from the camera, is to be processed on one of the available clusters (see again Figure 1). Due to the Parallel-Horus system, data parallel execution is obtained transparently on each cluster system. Additional task parallel execution is obtained by calculating over multiple frames (using multiple clusters) concurrently. Additional issues, including transparent job submission, tolerance against crashes, and transparent firewall circumvention, will be taken care of by JavaGAT, IPL, and SmartSockets. Most importantly, we aim to show true *wall-socket Grid Computing* — meaning that the application is being compiled and started on a local desktop machine, with Grid resources being employed entirely transparently (see also Figure 2).

In our demonstration we aim to use the recently installed DAS-3 (Distributed ASCI Supercomputer 3) Grid test bed. DAS-3 is a five-cluster wide-area distributed system, with individual clusters located at four different universities in The Netherlands: Vrije Universiteit Amsterdam (VU), Leiden University (LU), University of Amsterdam (UvA), and Delft University of Technology (TUD). The MultimediaN Consortium (UvA-MN) also participates with one cluster, located at the University of Amsterdam. As one of its distinguishing features, DAS-3 employs a novel internal wide-area interconnect based on optical 10G links, causing DAS-3 sometimes to be referred to as ”the world’s fastest Grid”.

Alternatively, we will use the GRID’5000 system, a 9-cluster Grid system in France, which is constantly expanding its available resources. DAS-3 and GRID’5000 have recently been linked by an optical 10G connection, such that combined use of both systems in our demonstration is clearly our intention. If time permits, we will also seek further resources elsewhere, potentially at a world-wide scale (i.e., including cluster systems in the United States and Australia).

Using the available computing capacity, we first demonstrate a ’learning phase’, in which our system will learn new objects, potentially obtained from people in the audience. Next, we demonstrate the recognition phase, by showing some of the previously learned objects to our system again. In case of recognition, our system will speak out loud the object’s name.

In case our proposal is accepted, the only essential requirement for showing the demonstration would be a dedicated Internet connection, with a minimum of 5 Mbit/sec uplink capacity. An additional table, located next to a wall, and a large screen or beamer for public display of user interfaces and additional video footage would be among our further desires.

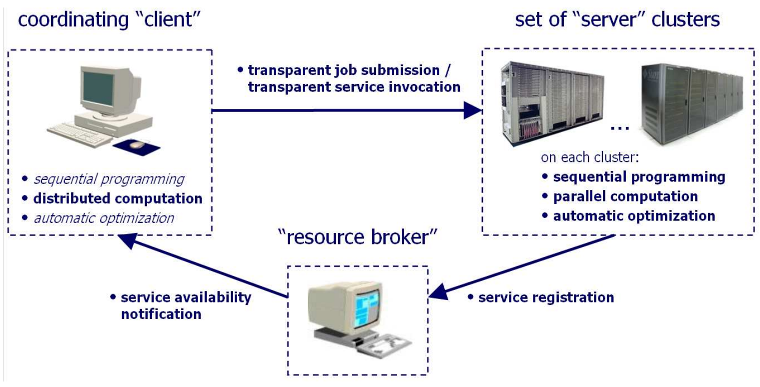


Figure 2. *Collaboration between a client, a set of multimedia compute servers, and a resource broker. Each cluster executes a server program, which is implemented in a fully sequential manner. Parallel-Horus parallelizes and optimizes the server programs automatically at run-time. Services may be made available by third parties, or may be started transparently from the client. Services register themselves at the resource broker. In turn, the client obtains registration information and server capability information from the resource broker.*

3 Innovation, outcome, and impact

A 'realization of the promise of the Grid', by integration of the described software solutions, would be a unique result. We have implemented a similar system before (see Figure 1), but this merely represents a proof of concept. This earlier system is not platform- and middleware-independent, is not fault-tolerant, and can not deal with connectivity problems. As such, our earlier system can not be described as a true Grid computing solution. This proposal aims to integrate all of the abovementioned requirements, to obtain one of the first — if not *the* first — wall-socket solution for the multimedia domain.

Given the fact that access to Grid resources is made fully transparent, our demonstration has further benefits for the Grid computing field. The most significant of these is the enhanced capability for investigating scalable supercomputing techniques, aimed at employing collections of tens or hundreds of cluster systems concurrently, each consisting of hundreds or thousands of compute elements (or more). Doing this on a regular basis, using today's mainstream Grid technologies, is simply impossible — not even mentioning the problem of debugging in systems of this size. With the help of an integrated software platform, all of these urgent problems become manageable.

Our approach also has major benefits for the field of multimedia content analysis. With a transparent Grid programming model available, researchers and developers in the MMCA domain, will be able to:

- enhance the development of new computing technologies for large-scale multimedia data analysis,
- improve the evaluation of new methodologies on very large multimedia data sets,
- obtain a deeper understanding of emerging multimedia computing methodologies and their efficiency.

In other words, a robust and easy-to-use software platform for Grid execution is an essential enabling technology for multimedia researchers and developers to maintain a leading position in the MMCA domain.

References

[1] I. Foster et al. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High-Performance Computing Applications*, 15(3):200–222, 2001.

[2] J. Geusebroek, R. van den Boomgaard, A. Smeulders, and T. Gevers. Color constancy from physical principles. *Pat. Rec. Let.*, 24(11):1653–1662, 2003.

[3] J. Maassen and H. Bal. SmartSockets: Solving the Connectivity Problems in Grid Computing. In *Proc. HPDC-16*, Monterey, USA, June 2007.

[4] R. Medeiros et al. Faults in Grids: Why are they so bad and what can be done about it? In *Fourth International Workshop on Grid Computing*, Phoenix, USA, Nov. 2003.

[5] F. Seinstra et al. High-Performance Distributed Video Content Analysis with Parallel-Horus. *IEEE Multimedia*, 15(4):64–75, Oct. 2007.

[6] A. Smeulders et al. Content Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[7] C. Snoek et al. The Semantic Pathfinder: Using an Authoring Metaphor for Generic Multimedia Indexing. *IEEE Trans. Pat. Anal. Mach. Intel.*, 28(10):1678–1689, 2006.

[8] R. van Nieuwpoort et al. Ibis: A Flexible and Efficient Java-based Grid Programming Environment. *Concur. Comput.: Pract. Exp.*, 17:1079–1107, 2005.

[9] R. van Nieuwpoort, T. Kielmann, and H. Bal. User-friendly and Reliable Grid Computing Based on Imperfect Middleware. In *Proceedings of SC'07*, Nov. 2007.