Laurence Loewe
started evolution@home, the first global computing system for evolutionary biology, and continues to develop it.

Global computing for bioinformatics

Laurence Loewe

Date received (in revised form): 24th September 2002

Abstract

Global computing, the collaboration of idle PCs via the Internet in a SETI@home style, emerges as a new way of massive parallel multiprocessing with potentially enormous CPU power. Its relations to the broader, fast-moving field of Grid computing are discussed without attempting a review of the latter. This review (i) includes a short table of milestones in global computing history, (ii) lists opportunities global computing offers for bioinformatics, (iii) describes the structure of problems well suited for such an approach, (iv) analyses the anatomy of successful projects and (v) points to existing software frameworks. Finally, an evaluation of the various costs shows that global computing indeed has merit, if the problem to be solved is already coded appropriately and a suitable global computing framework can be found. Then, either significant amounts of computing power can be recruited from the general public, or – if employed in an enterprise-wide Intranet for security reasons – idle desktop PCs can substitute for an expensive dedicated cluster.

Keywords: *global computing, Internet distributed computing, Grid,* in silico *biology, simulations, structure prediction*

INTRODUCTION

The idea to distribute computing tasks is old. People have been thinking about this since 1973, when the first Ethernet network was installed in the Xerox Palo Alto Research Center. Recently, interest exploded because (i) many cheap CPUs always outperform the fastest single CPU available for work that can be distributed, (ii) most CPUs in desktop PCs are only used marginally, (iii) the success of the Internet has connected many of these little used CPUs and (iv) the demand for CPU power in science is exploding. As supercomputers nowadays consist of a large number of CPUs, many scientists already structure their programs in such a way that they use multiple CPUs. These facts and the shrinking gap between the computing power of CPUs in ordinary desktop PCs and supercomputers have led to the idea of using idle cycles of CPUs connected by the Internet for scientific computing. The success stories of SETI@home and similar systems have long become more than a proof of concept for global computing.[1] The realisation that 10,000 desktop PCs with an average performance of 500 MFLOPS and appropriate software can be turned into a 5 TeraFLOPS world-class supercomputer, has caught the interest of big pharmaceutical enterprises and global computing software companies alike.

Large-scale scientific computing enterprises took another route to a similar end. When physicists noted that their demands were too large for any single computing facility, they started to develop systems that allowed pooling of resources in an organisation, thus the *Grid* was born.[2] Its name stems from the notion that eventually it will become possible to plug computing tasks into the Grid as electrical devices are plugged into the electrical power grid. Nobody cares where the actual power comes from. Today the Grid is no longer constrained to physics, but is well on its way into life sciences.[3] The Grid gave rise to e-Science, a new term describing research done through distributed global collaborations enabled by the Internet, using very large data collections, tera-scale computing resources and high-performance visualisation.[4–7]

Both approaches have in common that corresponding applications present

Laurence Loewe,
Microbial Ecology Group,
Department of Biosciences,
Technische Universität München,
Weihenstephaner Berg 3,
85354 Freising,
Germany

Tel.: +49 (8161) 71 3851
Fax.: +49 (8161) 71 4492
E-mail: Laurence.Loewe@evolutionary-research.net

themselves to the user as one very powerful system – no matter how complex their underlying structure.[8] This paper reviews opportunities for bioinformatics offered by these new approaches with the main focus on global computing.

Levels of global computing

DEFINITION

Global computing[9,10] *can be defined as massive parallel distributed computing using the Internet for data transfer to build a centrally controlled meta-computer from the idle CPU cycles of PCs of voluntary participants from the general public.* Other names for the same approach include Internet-based distributed computing, free grid computing and peer-to-peer (P2P) computing. However, in a narrow sense P2P refers to a structure where any peer can commission computing tasks, while in global computing all tasks are distributed by a central system. As this special form of grid computing is very young, related nomenclature is still evolving.[11,12] The definition above requires (1) voluntary participation as opposed to parasitic computing that can force computers on the Internet to produce results by misusing the TCP/IP protocol[13] and (2) openness to the general public as opposed to typical Grid projects that allow participation only for members of carefully selected institutions. Table 1 shows a short history of global computing. A global computing project can be operated at two levels:

Other names for global computing

Grid computing

- *Semi-automated* projects involve occasional manual interaction on the participant's side to get new computing tasks and submit results.
- *Fully automated* projects install a program that automatically gets computing tasks and submits results. It sometimes even automatically updates itself to newer versions.

It should be noted that the resource-sharing approach is virtually opposite to traditional client–server computing:[11] traditionally, clients hand some tasks over to the server (worker) and wait for the answer. In global computing the server (distributor) hands out computing tasks to clients (workers) upon request. While the initiative for a contact is on the client side in both models, the clients do the actual work in global computing.

This paper does not review *Grid* technologies. These help discover, authenticate and marshal IT resources within virtual organisations to enable distributed collaborations between computers and sometimes even between the people behind the computers. Some of the higher goals of the Grid are to facilitate e-Science and e-Business[7] and to build an intelligent semantic web[5] that contains many types of interconnected data, metadata and services in order to easily evaluate even complex questions. To glue these highly diverse functionalities together, web services[14,15] and the widely embraced

Table 1: Short history of global computing

Year	Milestone
1995	George Woltman: use free computing resources on the Internet for GIMPS, the 'Great Internet Mersenne Prime Search', the first semi-automated global computing project
1997	Independent start of the first fully automated global computing projects: PrimeNet (automated GIMPS) and distributed.net (brute force cracking of encryption keys)
1999	Start of SETI@home, currently the largest project with >3.8 million participants, >1 million CPU years and $>1.7 \times 10^{21}$ floating point operations in 38 months since start
2000	Global computing goes commercial. United Devices, Entropia, Parabon and others start business
2001	First companies go out of business
2002	About 60 active and 20 completed global computing projects listed by Kirk Pearson (http://www.aspenleaf.com/distributed/)

Global computing in its context of Grid computing

Open Grid Service Architecture (OGSA[16]) play an important role. To facilitate standardisation of fast-evolving Grid technologies, the Global Grid Forum was founded.[17] For further details, see some of the innumerable reviews about the Grid.[2,3,16–26] Nevertheless, a few remarks will help the reader to see the larger context of global computing in the field of Grid computing. Global computing builds a computational grid. It is very much centred on distribution of computations to substitute a dedicated supercomputer. Other types of grids, as reviewed by Skillicorn[12] include *access grids* (different users interact with a virtual environment, as if it was a single dedicated hardware platform where performance is not the first priority), *data grids* (allow easy handling of extremely large data sets, eg for next generation particle accelerators) and *data centric grids* (access and compute with extremely large data sets that cannot be moved to one place, eg distributed data mining).

Free grids

Another way of characterising grids is by ownership:[12] *free grids* exist because some individuals either allow unused cycles on their computer to be used for interesting projects or allow unused disk space and bandwidth to be used for file sharing. True global computing systems build free computational grids to allow everybody from the general public with a suitable Internet PC to contribute CPU cycles. This is especially interesting for researchers who cannot afford a dedicated compute cluster and, hence, are willing to spend the time to develop such a highly specialised system. While some frameworks are available to help build global computing systems (see below), the effort needed to do this and to advertise such 'anything@home–systems' should not be underestimated. Moreover, developers must be willing to deal with the security problems associated with this approach:

Virtual private grids

- Developers must have nothing to hide, be it valuable results or special algorithms in the code that may be extracted by reverse engineering.
- Developers must have a way to make sure that results are computed correctly and work from malicious participants is excluded by using checksums, repetition of work units, etc.
- Participants have to trust developers that participation cannot damage their system, because, for example, only trusted developers are allowed to distribute code, a sandbox-mechanism as in Java is employed.
- Participants have to trust developers that their software does not spy out a participant's system, because, for example, they know the developers, the source code is publicly available, a sandbox prevents critical operations.

An easy way to deal with the security issues is to employ only computers from within a virtual private network that sits behind the same firewall and belongs to a single corporate owner. Such *virtual private grids* are quite successful in large enterprises that need considerable computing power, eg to screen for potential drugs by protein–ligand docking. Instead of buying a dedicated cluster, such organisations can buy software that allows them to run a Windows version of the same code on the desktop PCs of their employees or a Unix version on servers with free capacities (see Table 5 below). Existing software can usually solve issues related to security, property, priority management and payment within the same organisation, as a certain level of natural trust exists within such a group. However, no company would risk the chance that competitors may be part of (and spy into or corrupt) its latest grid efforts. Thus, it is easy to see that trading of computing power in the industry will have a difficult time becoming a public market, even if calculating fair prices and accounting

were no problem. Owing to these reasons, companies that had the vision to make money with global computing have shifted their focus to deliver tools for building virtual private computational grids that save their owners from costs due to dedicated clusters.

Public grids

This may be different with *public grids*, as their computations and their results do not have to remain confidential. Here, a main issue is how to protect providers of CPU power from people who have no right to access it, as public grids are not open to the general public like free grids. Public grids are mostly being developed by public research agencies that have their own computing facilities and wish to level out peaks in requests for computing power. Thus, participating organisations should possess enough computing power to meet their average needs and rely on the Grid for peaks only. Before a public grid can become an effective market of computing power, a common set of protocols[16] and policies are needed as much as an effective way to compute prices and manage accounts.[12,23] Before this is achieved, public grids are likely to play a role only for those organisations that already have about as much dedicated computing-power as they need. They can use the Grid to level out compute peaks, mediate access to common resources or handle large amounts of data in a scalable way – very much the needs of high-energy physicists that invented the Grid.[27,28] While these communities set up their own public grids to meet their needs, the killer application that may drive global standardisation of a big public grid may be healthcare. For example, if all mammography data in the USA are stored for future analysis (which is sensible), many petabytes of data need to be handled.[29] However, before a big public grid takes off, enormous reservations regarding security and payment issues will have to be overcome.

Problems of a compute-power market

Examples of Grid applications

Nevertheless, these issues are no problem for an increasing number of biology-related projects that use the Grid very effectively to do e-Science.[3,6,7,25,26,29–31] Applications include:

- 3D Monte Carlo simulations of molecular interactions in living cells using MCell;[32]
- collection, analysis and visualisation of 3D optical microscopy data on *in situ* gene expression using the 3D OPT Microscopy Grid;[7]
- getting an overview over global and local biodiversity details using GRAB (GRid And Biodiversity);[7]
- comparison of the brain image of a patient against the population average using The Dynamic Brain Atlas;[7]
- sophisticated homology and fold recognition methods to visualise and assign 3D protein structures in the proteomes using the Structure Based Proteome Annotation Grid;[33]

and many other interesting applications.

In summary, global computing systems are *free computing grids* that are often employed by scientists that (i) cannot pay for the computing power they need, (ii) do not mind whether results or code remain confidential and (iii) can raise enough public interest in the topic of their computational research to get enough participants. Closely related are virtual private grids that basically use the same approach, but restrict it to 'ordained voluntary' participants that belong to the same organisation and sit behind the same firewall.

BIOINFORMATICS IN GLOBAL COMPUTING

Some active and completed global computing projects that touch bioinformatics are listed in Table 2. While some do sequence analysis, the most prominent projects engage in folding proteins or finding drugs (screen for interactions between potential drugs and proteins using docking algorithms).

Table 2: Some active and completed global computing projects that touch bioinformatics. For links and other projects see http://www.aspenleaf.com/distributed/ or http://www.evolutionary-research.net/Archive/Links/GlobComp/

Category	Existing systems
Linear regression	Exhaustive linear regression of clinical trial data (Parabon)
Sequence analysis	HMMER, BLAST, SIM4 (United Devices), TurboBLAST (Entropia), multiple sequence alignments (Übero)
Protein structure prediction	Folding@home, Genome@home, Parabon, Folderol, Distributed Folding
Docking drugs to protein structures	THINK search for drugs against cancer and anthrax (United Devices) AutoDock in FightAIDS@home (Entropia), Screen for oral drugs against anthrax + smallpox + Ebola (D^2OL) Screen for drugs against Tuberous Sclerosis Complex (Rothberg Institute for childhood diseases) Find-a-Drug (Treveren Consultants, the makers of THINK)
Individual-based models	evolution@home
Monte Carlo simulations	Compute future world climate (Climateprediction.com)

The speed of global computing . . .

However, any bioinformatics problem can be addressed by global computing, if it has a suitable structure (see below). Together with the rise of global computing,[1,34] grid computing is used more and more to solve computational problems in biology.[3,6,7,25,26,29–32]

. . . **does not solve algorithmic problems**

Global computing power is no substitute for developing intelligent software. If a simple BLAST query is run on a global (or distributed) network, it is faster, not better. It still includes as many false positives and negatives as if it were run on a slower computer. Most research and development in bioinformatics today is personnel-limited, not computing power-limited and it is often cheaper to buy an additional dedicated PC than to pay the price in staff for going global. However, when many BLAST searches have to be run in a short time, many proteins have to be folded, many ligands have to be docked or many complex individual-based models have to be simulated, then global computing can indeed help.

. . . **it creates new ones**

While sometimes hopes run high to easily solve complex problems by global computing, in reality it is not always feasible to start a global computing project. Besides the general structure of the computational problem, practical details of software and project development can hamper such efforts considerably. And if the whole system is not run on privately owned PCs, public participation psychology and promotion efforts need to be considered.

STRUCTURE OF PROBLEMS ADDRESSABLE BY GLOBAL COMPUTING

There are well-defined characteristics of problems that can be efficiently addressed by global computing:

- They express trivial parallelism (so-called embarrassingly parallel or multi-parameter applications): work can be divided into completely independent units of reasonable size.
- Work units have a high computation per bandwidth demand and corresponding files are of reasonable size.
- Memory requirements are small enough to fit desktop PCs.
- The worker is 'folder-centric' (ie all input files are automatically read from its current working directory and all results files are written to it; there are *no* external interactions, except getting new tasks and submitting results).

How to break a big problem into small tasks

Corresponding applications analyse a large amount of data by targeting one subset after another, as for example many independent protein sequences that need to be folded into 3D structures or a large sequence database that can be searched one chunk at a time. Other examples include individual-based simulations of evolution that just start with different parameters or regression analysis of many small subsets of a big clinical trial database. This general structure usually requires some pre-processing (to divide the big task into smaller work units) and some post-processing (to synthesise the big picture from all smaller results). It should be noted that network bandwidth and Moore's law impose lower and upper computing time limits for tasks that are worth distributing.[35] The resulting range from seconds to months is reflected in current global computing projects, with several hours as a typical value.

When not to use global computing

However, if (i) network demand is too high, (ii) different work units need to communicate with each other occasionally, (iii) deadlines for completing tasks need to be enforced, (iv) the data cannot be kept in one place or (v) other special Grid features are needed, then some Grid solution should be employed. If work units need to communicate extensively among themselves, a supercomputer should be used. The price of the latter solutions increases, as network speed increases cost. Figure 1 depicts this relationship graphically.

ANATOMY OF SUCCESSFUL PROJECTS

Observing existing projects allows one to derive requirements for success in global computing. To run a highly successful global computing project, you need to:

- have a good way to pack your problem into independent work units of appropriate size (CPU time hours to weeks, transmitted data as little and as rare as possible, depends on the network);
- develop worker software that has almost no interference with the users'

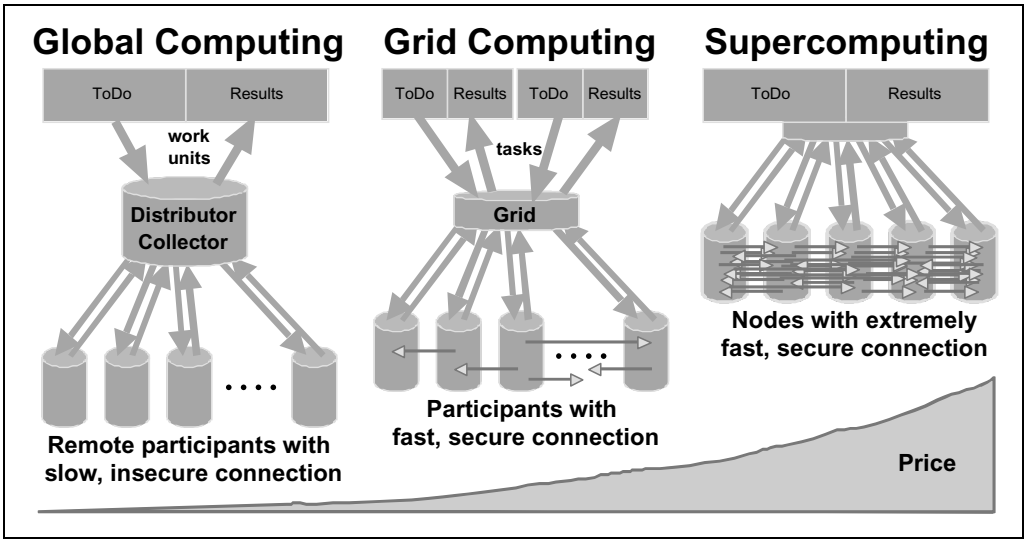

Figure 1: Problems that can be addressed by global computing have structures other than those only addressable by traditional grid computing or supercomputing. The small arrows symbolise potential communication between different tasks on different nodes, which is not possible in global computing. While the number of nodes is fixed in a supercomputer, a grid can add new nodes dynamically, if they belong to the same virtual organisation and comply with its policy. Participation in global computing is free, but a central authority controls computation of more or less uniform work units. In true peer-to-peer computing (not shown), there is no central authority; thus each node not only works, but also can schedule new tasks

Table 3: Features of the ideal worker software for free global computing projects

Category	Details
Readily available	Easy distribution without registration over the web to get started. Cross-platform code included in major share-ware collection CDs. Small installer size (download-times and space on the local hard disk).
Fully automated	Get work units and submit results without participants' interaction. Automated computing code download. Automated upgrading of the worker itself.
Non-interfering	Lowest background priority possible. No use of virtual memory (may block the rest of the system). Immediate stop without losing results at participant's wish. Stop automated online connections at participant's wish. Screen-saver only version for less interference.
High security standards	Secure execution environment shields participant's computer from potentially malicious code of a computing application (Sandbox). Secure results transmission keeps participants from deceptive practices. Saves intermediate results to minimise losses in case of system crash. System service version for computing after logout.
Nice graphical user interface	Show beautiful picture of what is being computed. Show progress and local computing statistics and allow easy interruption. Easily edit preferences that describe the commitment of the participant. (Allow commercials? What project? High-score identity? Work unit size in terms of CPU-time, RAM, disk space and network bandwidth?)

Requirements for success in computing

daily work; see Table 3 for features of the ideal worker software;

- set up a central server that automatically distributes work units;
- set up a central server that automatically collects results, checks their integrity and facilitates analysis;
- build a web site that explains the problem you are trying to solve, distributes the worker software for free and publishes participants' computing statistics;
- offer a compelling motivation to participate (see Table 4 for examples);
- organise a good, international advertisement campaign.

Low effort global computing projects

As can be easily seen from this list, the reward of global computing power comes only at the cost of dealing with numerous problems that have no primary connection to the original content of the research. Owing to dependency on public participation, even non-IT issues such as advertisement and motivation have to be considered.

After describing the highest possible standards above, it should be noted that one does not necessarily need a big team to run a global computing project. Several projects actually run as a one-person effort. If everything has to be as simple as possible, then choose a semi-automated mode of operation (e-mail or web distribution of groups of tasks and e-mail submission of results), support only the Windows platform, make only a

Table 4: Possible motivations for participants in global computing projects

- Interest in the specific problem
- Contribute to global progress in general
- Affinity to high-score races with meaningful side effects
- Get famous by finding something significant
- Get the chance to win some money
- Commercial selling of computing time
- Execution of orders of their company on its computer
- Philanthropic sponsoring by a company
- Like beautiful screen savers
- Want to have some fun
- Compute because their friends are doing it

Automating basic tasks in global computing

simple, static project web site and advertise by one e-mail to a global computing review site.[1] This may well be an option for some situations, as 10 years of global computing time are easily worth more than 50,000 CPU hours on a supercomputer.

Between these two extremes are numerous paths to success in global computing, but each one requires building or using an appropriate code that enables global computing.

EXISTING GLOBAL COMPUTING FRAMEWORKS

To solve the general problems that every global computing project encounters, a number of software frameworks have been developed (Table 5). The easiest frameworks provide code that is compiled with and statically linked to the application that shall be distributed. Such code can manage automated distribution of tasks and collection of results. Other frameworks allow execution of native code and facilitate automated updates of the distributed worker software. Some even provide a sandbox for execution to keep such code from damaging the participant's system. Some frameworks give a lot of attention to security and authentication details, while others do not. Many frameworks employ Java. While this has the big advantage of being (potentially) platform independent, it sometimes requires that you have to distribute the full Java platform with the worker software to allow simple installation. The resulting increase in download size and potential conflicts with other JavaVMs belong to the downside of that approach. Some frameworks, especially those of some global computing companies, include support for some pre- and post-processing tasks (divide labour and integrate results).

Pre- and post-processing are important

These pre- and post-processing facilities turn out to be a key factor in global computing. Basically there are two types of problems here: the 'ruby-

Table 5: Some major frameworks for global computing and grid computing. Commercial products are marked with *. This table includes most of the important frameworks for global computing, but only a few from the multitude of grid computing frameworks. For more, see collections in http://www.aspenleaf.com/distributed/distrib-devel.html, http://dmoz.org/Computers/Computer_Science/Distributed_Computing/Platforms/and http://www.GridComputing.com/

Platform name	Developer	Web site
Fida	Ding *et al.*	http://odinn.chem.washington.edu
COSM	Beberg at Mithral *et al.*	http://cosm.mithral.com
BOINC	SETI@home II, Anderson *et al.*	http://boinc.ssl.berkeley.edu
Models@Home	Kriege & Vriend	http://www.cmbi.nl/models
XtremWeb	Cappello *et al.*	http://www.xtremweb.net
MetaProcessor*	United Devices	http://www.ud.com
DCGrid*	Entropia	http://www.entropia.com
Frontier*	Parabon Computation	http://www.parabon.com
P2P Accelerator Kit	Intel	http://www.intel.com/ids/p2p
JXTA	Sun	http://www.jxta.org
Sun Grid Engine*	Sun	http://www.sun.com/gridware
Platform LSF*	Platform	http://www.platform.com
Globus Toolkit	US universities and labs	http://www.globus.org
Condor	University of Wisconsin	http://www.cs.wisc.edu/condor
DataGrid	CERN	http://www.eu-datagrid.org

Ruby-in-the-rubbish problems

in-the-rubbish' type problems screen a large number of input possibilities for a small number of valuable results (eg GIMPS, SETI@home or potential drug candidates). This makes data handling easy: all tasks have the same priority initially and either give an interesting result that is further investigated manually or are uninteresting and can be written to an archive as they are.

Ant-hill problems

In contrast, the 'ant-hill' type problems are much more complicated. Here, things are usually connected, as early simulation results have implications for parameter combinations scheduled in the future, because scientists are watching results and adjust their questions correspondingly. At the end, more or less all results are needed simultaneously for interactive evaluation, as it is not the single result, but its overall context that is interesting (eg evolution@home).

Commercial global computing

Such hypothesis testing calls for flexible scheduling, prioritising of particular tasks and handling of work units that span several orders of magnitude in computational complexity. Often, it also requires handling of incomplete results when predictions of computational complexity proved inadequate,[35] as biological questions tend to be painfully unaware of associated computational complexities. Since such features are very specific to particular problem domains, most global computing frameworks feel definitively more comfortable with searching for a ruby in the rubbish.

Hidden costs of participation

Hardly any global computing framework available allows users to choose the computational complexity of their work units. Experiences with evolution@home, however, have shown that it makes a great deal of sense to support such a choice and participants like it.[35] This should be kept in mind, as individual-based models are probably not the only type of problem where natural work unit sizes have a high variability. Similarly, other problems may require features that are yet unsupported by current global computing frameworks.

Participant choices of computational complexity

COSTS OF GLOBAL COMPUTING

When deciding whether global computing or other means should solve a particular problem, the price tag plays an important role. In a commercial setting, any global computing framework will only run on PCs behind the company's firewall to avoid security problems associated with public participants. If a company considers buying a dedicated cluster for running some existing 'folder-centric' code, it should consider employing a solution from one of the global computing or Grid companies. For a few per cent of the costs of a corresponding dedicated Linux cluster, one gets secure worker software for Windows that executes native applications in a secure sandbox and automatically gets tasks from and submits results to the server software included.

On the participant's side

Participation in global computing is free of fees. However, every participant has to pay for their connectivity and electrical energy. The latter is the equivalent of about one light bulb per PC, while the former is not very expensive in many projects. Finally, there are no administration costs, if your system is fine. Uninterrupted computation, however, may expose a weakness in your system that was not apparent before (cooling problems in summer, hard disk failures, RAM problems, etc.; especially watch out for problems in notebooks that are often not designed for long-term uninterrupted computation). Therefore participation can only be recommended if participants have a working air conditioning in summer and a reasonable backup strategy (for most ordinary PCs the cheapest solution is to regularly copy everything to a dedicated backup hard disk). Since system administrators of larger facilities have to provide that anyhow, additional costs to make a large number of PCs participate this way is low from the hardware perspective.

Software administration costs on the

More automatisation

participant's side depend on the project. With fully automated worker software that even updates itself, administration is reduced to a minimum. On the other extreme, semi-automated projects require regular interaction, which is no longer trivial if the number of PCs increases. Therefore, worker software that is as simple as possible is important for global computing projects.

On the operator's side

... leads to lower administration costs ...

Costs on the operator's side depend heavily on the framework and on what exactly these costs include. If these costs are defined in terms of distributing worker software and work units on one side and collecting results on the other, then this is extremely cheap: United Devices manages about 500,000 participant PCs with just two system administrators. However, if data analysis is included, things may easily get expensive, as unclear objectives and little automation can result in exploding costs. This should not surprise operators, as a global computing project should not start at all if methods of analysis and goals are not clear.

... easier website updating ...

Besides the results-oriented part of administration, no global computing project can be active without a web site. Keeping it up to date and organising the publicity needed demand human resources too. In the case of a public project one should not forget the costs to maintain a high-scores list. As a general rule of thumb, global computing projects with an automatically updated high-scores list have several times more participants than projects without, as many people want to monitor their contribution and see whether the project is still active. When all this work is highly automated, operator costs are low. However, to automate this may require considerable development efforts.

On the developer's side

... and higher development costs ...

This is the biggest hurdle. In order to efficiently concentrate on the scientific issues, a global computing framework has to take care of all low-level tasks such as secure task and result transmission. Unfortunately, selection of the best framework for a particular situation (features required, resources available) is by no means a small task as such a foundation determines what can be built upon it. If a good framework is found and the application has an appropriate structure, costs of development can be relatively low. Such applications already have a 'folder-centric' worker and corresponding pre- and post-processing code. Then, from the perspective of the application, the global computing framework needs only to transfer input files, output files and worker updates.

For those, who do not find an appropriate ready-to-use framework for their problem, only three possibilities remain: abort going global, develop a new framework or pick the best framework available and supplement it with the required functionality. The latter two propel those engaged to the exciting front of research in global computing.

Acknowledgments

I want to thank S. Scherer and the FML Institute of Microbiology at TUM for financial support, L. Westfall for proofreading and the anonymous reviewers for their helpful suggestions.

References

1. Pearson, K. (2002), 'Internet-based Distributed Computing Projects' (URL: http://www.aspenleaf.com/distributed/).
2. Foster, I. and Kesselman, C. (1998), 'The Grid: blueprint for a new computing infrastructure', Morgan Kaufmann Publishers, San Francisco.
3. Goble, C. (2002), 'The grid: from concept to reality in distributed computing', *Scientific Computing World*, May/June, *Bioinformatics World*, Vol. 2, pp. 8–11 (URL: http://www.bioinformaticsworld.info/feature3b.html).
4. Oxford e-Science Centre (2002), 'Homepage', (URL: http://e-science.ox.ac.uk/).
5. deRoure, D., Jennings, N. and Shadboldt, N. (2001), 'Research agenda for the semantic grid: a future e-science infrastructure' (URL: http://umbriel.dcs.gla.ac.uk/Nesc/general/technical_papers/DavidDeRoure.etal.SemanticGrid.pdf).

6. UK e-Science (GRID) core programme (2002), 'eScience Homepage' (URL: http://www.escience-grid.org.uk/).
7. Gridoutreach (2002) 'The Grid Outreach Programme 2002–2003 Homepage' (URL: http://www.gridoutreach.org.uk/).
8. Tannenbaum, A. S. (1995), 'Distributed operating systems', Prentice Hall, Englewood Cliffs, NJ.
9. Germain, C., Neri, V., Fedak, G. and Cappello, F. (2000), 'XtremWeb: building an experiemental platform for Global Computing', in 'Proceedings of GRID'2000 (URL: http://www.lri.fr/~fedak/XtremWeb/grid2000.ps.gz)
10. Fedak, G., Germain, C., Néri, V. and Cappello, F. (2001), 'XtremWeb: a generic global computing system' in: Buyya, R. and Mohay, G., Eds, 'First International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems at the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid – CCGrid2001', IEEE Press, Brisbane, pp. 582–588 (URL: http://www.lri.fr/~fedak/XtremWeb/Gcpd.ps).
11. Kant, K., Iyer, R. and Tewari, V. (2002), 'A framework for classifying peer-to-peer technologies', in Bal, H. E., Löhr, K.-P. and Reinefeld, A., Eds, 'Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)', IEEE Computer Society, Berlin, pp. 368–375.
12. Skillicorn, D. B. (2002), 'Motivating computational grids', in Bal, H. E., Löhr, K.-P. and Reinefeld, A., Eds, 'Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)', IEEE Computer Society, Berlin, pp. 401–406.
13. Barabasi, A. L., Freeh, V. W., Jeong, H. and Brockman, J. B. (2001), 'Parasitic computing', *Nature,* Vol. 412, pp. 894–897.
14. W3C (2002), 'Web Services Homepage' (URL: http://www.w3.org/2002/ws/).
15. WebServices.Org (2002), 'The Web Services Community Portal' (URL: http://www.webservices.org/).
16. Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S. (2002), 'The physiology of the Grid: An Open Grid Services Architecture for distributed systems integration' (URL: http://www.globus.org/research/papers/ogsa.pdf).
17. Global Grid Forum (2002), 'GGF Homepage' (URL: http://www.globalgridforum.org/).
18. Foster, I. (2000), 'Internet computing and the emerging Grid' (URL: http://www.nature.com/nature/webmatters/grid/grid.html).
19. Foster, I. (2002), 'The Grid: a new infrastructure for the 21st century science', *Physics Today,* Feb., pp. 42–47 (URL: http://www.aip.org/pt/vol-55/iss-2/p42.html).
20. Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S. (2002), 'Grid services for distributed systems integration', *Computer,* June, pp. 37–46 (http://www.globus.org/research/papers/ieee-cs-2.pdf).
21. Foster, I., Kesselman, C. and Tuecke, S. (2001), 'The anatomy of the Grid: enabling scalable virtual organizations', *Int. J. Supercomputer Appl.,* Vol. 15, pp. 200–222 (URL: http://www.globus.org/research/papers/anatomy.pdf).
22. Foster, I. and Kesselman, C. (1998), 'Computational Grids' in Foster, I. and Kesselmann, C., Eds, 'The Grid: Blueprint for a New Computing Infrastructure', Morgan Kaufmann Publishers, San Francisco (URL: http://www.globus.org/research/papers/chapter2.pdf).
23. Buyya, R. (2002), 'Economic-based Distributed Resource Management and Scheduling for Grid Computing', PhD Thesis, Monash University, Melbourne, Australia (URL: http://www.buyya.com/thesis/).
24. Berman, F. (2002), 'Application level scheduling on the computational Grid' (URL: http://apples.ucsd.edu).
25. Goble, C. (2001), 'Review: the low down on e-science and grids for biology', *Comp. Funct. Genomics,* Vol. 2, p. 365–370 (URL: http://www.hgmp.mrc.ac.uk/CCP11/cfg_grid.pdf).
26. Oinn, T. (2002), 'myGrid – directly supporting the eScientist' (URL: http://www.ebi.ac.uk/mygrid/mygrid.pdf).
27. European Union (2002), 'The DataGrid Project' (URL: http://www.eu-datagrid.org/).
28. GriPhyN – Grid Physics Network (2002) 'Homepage' (URL: http://www.griphyn.org/).
29. IBM Life Science Solutions (2002), 'National Digital Mammography Archive: University of Pennsylvania consortium and IBM develop computing grid for breast cancer screening' (URL: http://www-3.ibm.com/solutions/lifesciences/pdf/NDMA_8pg_FINAL.pdf).
30. European Bioinformatics Institute (2002), 'E-Science @ EBI' (URL: http://www.ebi.ac.uk/escience/).
31. myGrid (2002), 'myGrid Homepage' (URL: http://www.mygrid.org.uk/).
32. Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F. (2000), 'Heuristics for scheduling parameter sweep applications in Grid environments', in 'Proceedings of the 9th Heterogeneous Computing Workshop

(HCW'2000)', Cancun, Mexico, IEEE, New Jersey, pp. 349-363.

33. Sternberg, M., Jones, D., Thornton, J. *et al.* (2002), 'Structure-based proteome annotation grid' (URL: http://www.lesc.ic.ac.uk/projects/das-grid.html).

34. Trunnell, M. (2001), 'Computing power: drawing power from the people', *Scientific Computing World*, Nov./Dec., pp. 14–16.

35. Loewe, L. (2002), 'evolution@home: experiences with work units that span more than 7 orders of magnitude in computational complexity', in Bal, H. E., Löhr, K.-P. and Reinefeld, A., eds, 'Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)', IEEE Computer Society, Berlin, pp. 425–431 (URL: http://www.evolutionary-research.net/Science/Papers/2002/Loewe2002-EaHworkunits.pdf).