The Ruth & Ted Braun Awards
for Writing Excellence
at Saginaw Valley State University

Data-intensive Applications in Distributed and Grid Systems: Molecular Engineering

Olaf Rudnik

COLLEGE OF SCIENCE, ENGINEERING & TECHNOLOGY

Nominated by Dr. Farid Hallouche
Associate Professor of Computer Science

Olaf Rudnik, from Bay City, Mich., is a second degree senior majoring in Computer Science and Chemistry. Originally from Poland, he holds an M.A. in English from Adam Mickiewicz University, Poznan, Poland. After graduation he hopes to find a career in networking and software development.

Introduction

The increasing necessity for more powerful systems capable of storing and analyzing hundreds of terabytes or even petabytes of data and successfully sharing that data across multiple often incompatible platforms excludes the use of a single personal computer, a powerful mainframe machine, or even a supercomputer. The sheer computational power and storage capabilities are far beyond any single, highly localized system. Until about a decade ago, scientists and organizations depended heavily on the use of the above-mentioned resources for processing of information obtained from a multitude of scientific experiments. Unfortunately, these resources were not sufficient to allow the researchers to share the results of their efforts on a wider scale. Clearly a more global approach needed to be employed.

With the advent of the Internet, online computing, and resource sharing, came the idea of employing whole networks of computers, data storage facilities and even spare cycles of end users' CPUs. As much as individuals would like to think that their home machines are utilized to their full potential, people slowly came to realize that it is an unlikely scenario in most cases. As people browse the Web or check email, their PCs, and possibly millions of other interconnected computers, are sitting somewhat idly by and are wasting some of the most valuable resources that people have access to. Although email, Web browsing, and office applications comprise most of the computing needs for a majority of people, the use of a global supercomputer could most definitely aid scientists by contributing to the development of new drugs, creating more efficient work spaces and aeronautics advancements, and exploring many more applications. This paper will look at the history, development and applications of distributed and grid systems in computation and data-intensive sciences.

Distributed and Grid Systems

The ever-increasing number of scientific research endeavors carries with it a tremendous increase in data size and storage needs. Computer scientists and scientists in general were faced with a problem of providing the research community with a viable solution to obstacles in sharing and processing all of that information. In the early and mid 1990s, two new approaches to resource sharing and processing emerged: distributed computing and grid systems. Neither of these approaches would have been possible without the connectivity that the Internet and dedicated networks have to offer. While superficially similar, these two computing resources have significant underlying differences.

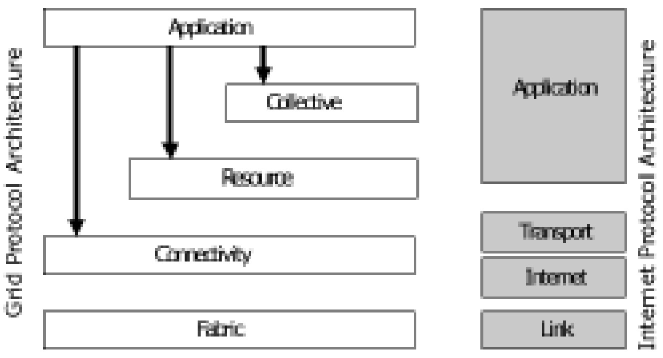
Grid Systems

A grid is a system that encompasses the "collaborative use of computers, networks, databases, and scientific instruments owned and managed by multiple organizations" (Asadzadeh et al., n.d.). Milestone work on the subject by Foster, Kesselman & Tuecke (2004) identifies a grid as "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" in which the focus of attention is not on "file exchange but rather direct access to computers, software, data, and other resources . . . emerging in industry, science, and engineering." Such usage somewhat evenly distributes the costs involved in building, maintaining and operating expensive hardware such as particle accelerators or telescopes, for example. Furthermore, data from a single experiment carried out on such an instrument may need to be looked at by different research and interest groups and analyzed according to their specific needs. Therefore, secure access not only to that data, but also to other resources including, but not limited to, computational power, is of paramount importance.

Historically, grid systems targeted mostly high-powered computers such as servers, mainframes and supercomputers, but they are not exclusively limited to those and can incorporate in their infrastructure end user PCs as well. The software, services and protocols, or middleware as it is known, used for supercomputing purposes are often written in specialized languages such as HPC++ or MPI, which are not easily implemented on PCs or by average PC users. What further exacerbates the difficulties encountered in the implementation of grid systems are the unique challenges these systems face due to dispersion of scientists, their resources and equipment, as well as the multitude of databases of experiment results that need to be stored, retrieved, and analyzed (Erlanger, 2002).

Necessarily, creating the grid architecture, which needs to span not only various heterogeneous resources, but also entire research sites and organizations spread across the world, is a daunting task. A generic, high-level representation of grid layers can be seen in Fig.1 (Foster et. al., n.d.).

Figure 1. Layered grid architecture and its relationship to the Internet protocol architecture



Source: Foster, et. al., n.d.

As observed by Avery (2002), the lowest level, the *Fabric,* includes "shared resources such as computer clusters, data storage systems, catalogs, networks, etc." Above it, the *Resource and Connectivity* layers furnish access to resources and various communication tools needed to use the *Fabric* layer. The *Collective* layer enables coordination in using numerous resources spread over multiple sites. Finally, all of these allow scientists to tailor the system to the needs of their specific tasks within the *Application* layer.

The software and applications that enable the use of grids are known as the middleware; however, any lengthy description of middleware used for a particular project is beyond the scope of this paper. It is worth pointing out, though, that the need for interoperability and uniformity of systems has given rise to many undertakings in creating universal tools that can resolve the issues arising from the distribution and lack of homogeneity among scientific resources. One such example is the Globus toolkit which is implemented using C and Java, languages well known to nearly all programmers. Furthermore, it provides secure authentication and log-on services to end users through "PKI (Public Key Infrastructure) technology coupled with Web Services" (Globus Toolkit, 2004). It allows an interested party to perform a single log-on via a smartcard or a certificate particular to the user without the necessity to keep track of sites, credentials, etc.

As far as the hardware platforms for grid systems are concerned, it should be noted that grids were geared more towards clusters of computers, mainframes, and machines located mostly at research sites. Their connectivity was also initially relegated to high speed networks that were oftentimes designed and built specifically for grid use. In recent years, however, the trend of merging high speed networks (GB/s) into the structure of the World Wide Web has created a new potential for unification of computational and storage resources.

Distributed Systems vs. Grid Systems

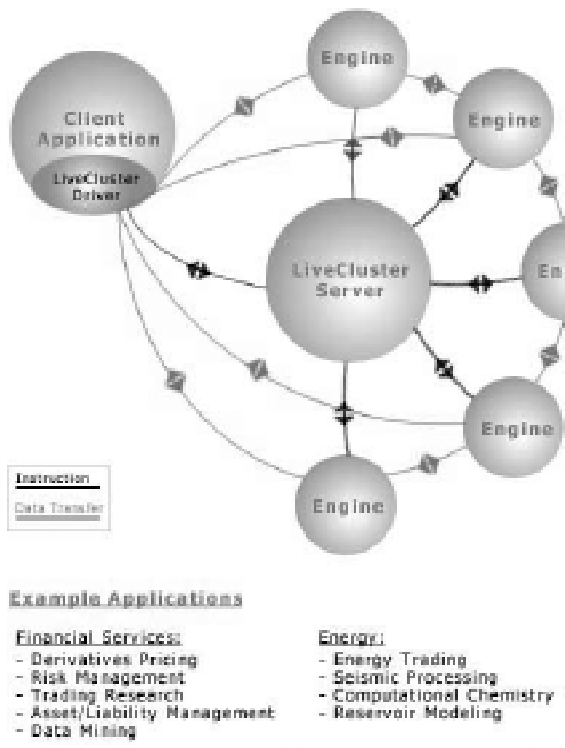
The two terms–grid and distributed computing–may seem confusing at first and quite rightly so. After all, the base upon which they rely is more often than not the World Wide Web and both use highly dispersed resources. The best way to draw a clear dividing line between them is to look at who provides the resources and who uses them.

On the resource side, such as power, memory, and storage, the grid computing uses dedicated and often high-end computers such as mainframes or supercomputers which often run incompatible operating systems. On the other hand, distributed systems mainly target the end-user PCs with some piece of code installed locally on the end-user's computer, which then connects to the managing server for data to process. That code is provided in versions that can be run on a specific software platform such as Windows, Linux, or Mac OS in their various iterations.

One of the most significant differences between the two approaches is in the software and connectivity with which they are implemented. Distributed computing software can be written in C, C++, Java or other high-level languages which are more easily understood by an average user, but may not be sufficient for math-intensive applications. Moreover, the distribution and availability of computers renders this approach much less useful for programs running in real time due to dropped connections, delays in send/receive operations, end users' need to utilize the full power of their PCs, PCs going offline, etc. (Erlanger, 2002). Also, distributed computing shifted its workload mostly to the Internet, as we know it, the speed of which may sometimes leave a lot to be desired. Therefore, the applications, with the need for nearly instant response upon query/request from user(s), puts constraints on the use of distributed computing in data-intensive sciences. For example, a request from an astronomer for transfer of Hubble Space Telescope (HST) images for analysis of light spectra from distant galaxies might take days if not longer, especially considering the fact that the size of the database from HST counts in terabytes (NASA, 2004). The calculations need not be very intricate themselves, but the dispersal and size of data to be analyzed may complicate the process sufficiently to render such approach impractical or even impossible.

Another crucial difference in the use of distributed computing versus grid systems is the manner in which software exploits the computational power of the machine(s) it runs on. Distributed computations, being mostly run on office or home computers, need to place emphasis on their non-intrusive character, that is, the ability of the users to regain prompt control over their machines should the need arise. Distributed computing can, unfortunately, be limited by security and management issues as well as lack of standardization in software and protocols, oftentimes followed by bandwidth restrictions imposed on the users by their ISPs (Erlanger, 2002). Another variation on the subject of global computing is a construct known as cluster. That particular arrangement allows for direct communication not only between the end users' PCs and managing server(s), but also between the PCs themselves. Figure 2 provides a simple pictorial representation of a typical cluster system.

Figure 2. A cluster system



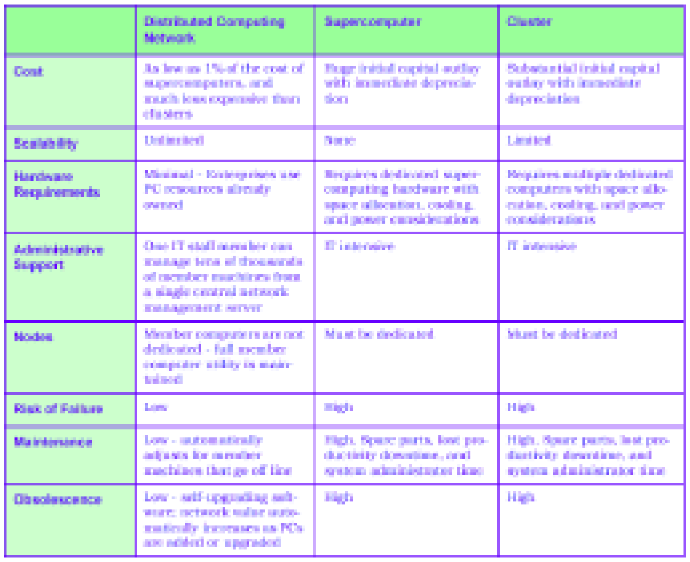
Source: Erlanger, L., 2002

We can thus see that distributed computing is much more centralized with one or more servers allocating particular jobs. In the end it is the users who tell the system (grid) what they need, not the system telling the users what needs to be done (distributed). To provide some interesting statistics, Erlanger takes a look at a practical application of distributed computing:

The performance improvement over typical enterprise servers for appropriate applications can be phenomenal. In a case study that Intel did of a commercial and retail banking organization running Data Synapse's LiveCluster platform, computation time for a series of complex interest rate swap modeling tasks was reduced from 15 hours on a dedicated cluster of four workstations to 30 minutes on a grid of around 100 desktop computers. Processing 200 trades on a dedicated system took 44 minutes, but only 33 seconds on a grid of 100 PCs.

Figure 3 provides a straightforward comparison of distributed computing versus dedicated systems.

Figure 3. Distributed computing vs. dedicated system



	Distributed Computing Network	Supercomputer	Cluster
Cost	As low as 1% of the cost of supercomputers, and much less expensive than clusters	Huge initial capital outlay with immediate depreciation	Substantial initial capital outlay with immediate depreciation
Scalability	Unlimited	None	Limited
Hardware Requirements	Minimal - Enterprises use PC resources already owned	Requires dedicated supercomputing hardware with space allocation, cooling, and power considerations	Requires multiple dedicated computers with space allocation, cooling, and power considerations
Administrative Support	One IT staff member can manage tens of thousands of member machines from a single central network management server	IT intensive	IT intensive
Nodes	Member computers are not dedicated - full member computer utility is maintained	Must be dedicated	Must be dedicated
Risk of Failure	Low	High	High
Maintenance	Low - automatically adjusts for member machines that go off line	High. Spare parts, lost productivity downtime, and system administrator time	High. Spare parts, lost productivity downtime, and system administrator time
Obsolescence	Low - self-upgrading software; network value automatically increases as PCs are added or upgraded	High	High

Source: Erlanger, L., 2002

As an example of distributed computing, one particular application stands out: SETI@home. The Search for Extra Terrestrial Intelligence has captivated the minds of many people and is known worldwide. At present its software works as a screen saver and, as a ~700 KB download, poses no problem for most users, even those who still connect to the Internet via dial-up modems (SETI@home, n.d.).

Data-Intensive Sciences and Molecular Engineering

Various technological advances in scientific tools have allowed researchers to explore the surrounding world both on a macro- and microscale.

> Computers have improved these tools and made . . . new methods of studying molecules, *molecular modeling* and *combinatorial chemistry,* possible. In molecular modeling, chemists use computers to simulate the structure and motion of macromolecules. In combinatorial chemistry, a chemist can use robotic tools . . . to make a huge number of slightly different molecules. This set of molecules helps the chemist search for a useful molecule (Molecule, 2001).

No discussion of existing and practical applications would be complete without mentioning a few examples. The research in grid computing has contributed to the rise of many national and international infrastructures. The following, although by no means exhaustive, is a list of some of the major branches of science which can benefit from the increase in computational power as well as data storage provided by spreading the workload over multiple systems and domains. In addition to that, most of the entries on the list are in fact tied with molecular engineering, which is a basis for nanotechnology and biotechnology of the 21st century.

High Energy Particle and Nuclear Physics: Of special interest in this field are the particle accelerators such as the Large Hadron Collider at CERN, Switzerland. Although not yet fully operational, it is scheduled to begin experiments in 2007. Scientists estimate that its "detector will record events," that is, collisions of particle beams, "at a rate of approximately 100 Hz, accumulating 100 MB/s of raw data or several petabytes per year of raw and processed data in the early years of operation" (Avery, 2002).

Astronomy: Surveys of the sky using gigapixel CCD arrays will soon increase the data size from tera- to petabytes, just as will the many satellites orbiting our planet and other planets of the solar system.

Biology and Medicine: Genome sequencing of many earthly organisms as well as study of the structure and function of proteins will greatly contribute to advances in the field (Leach, 2001).

Bioinformatics: Design of bio systems (such as DNA computers) will help solve complex problems.

Environment Management: Prediction of spills based on geological and geographical data will enable scientists and emergency response teams to minimize the consequences of industrial accidents.

Molecular Engineering and Pharmaceutical Industry: Many sciences such as chemistry, molecular physics, and nanosciences use a lot of software that is related. Molecular engineering in particular is a large part of the research and development of pharmaceutical companies. These companies use computers for molecular modeling to screen for drugs that might not be successful and to begin research on new ones. This is extremely important to the pharmaceutical companies, because the computational methods of modeling are projected to save $50 million per drug developed. Major drug companies might create and test new molecules, yet these may prove not to be what the company needs at the moment. Thus, another scientist seeking a protein for drug testing/interaction might use the company's existing database and run modeling against his own database (Karelson, n.d.).

Molecular simulations and modeling are not exclusively limited to drug design. Their potential application in other projects lets the scientists and engineers create new exciting designs. Industrial research and development teams are also interested in molecular engineering, but their interests lie in "novel catalysts for environmentally sustainable production of chemicals, optically nonlinear organic materials, and nanobiostructures" (Karelson, n.d.). The reasons these advances have been made are the amount of knowledge gained through years of research and the powerful software now available.

However, molecular modeling is not without its stumbling blocks. First, molecular engineers at present cannot fully utilize all of the resources out there because they are spread out in many different research laboratories across the world. Secondly, any time scientists would like to combine those various resources, they have to worry about incompatibility of hardware and software. The answer to these problems is the use of computational grid systems. Grid middleware can deal with the volume of data produced, as well as provide an integration of resources from public and commercial databases.

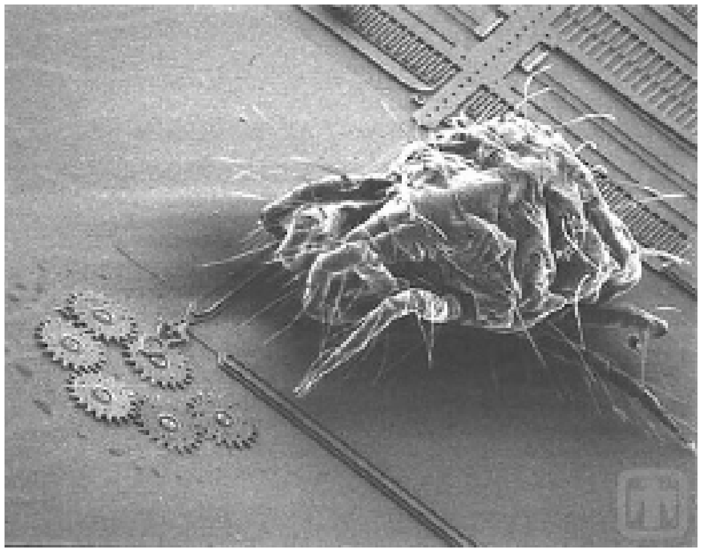
Modeling software can produce thousands of compounds a day, so this amount of data requires storage space in the order of terabytes. Placing this information on a network is the best way to store it. Given the fact that a job designing a single drug may need to screen millions of compounds, a cluster-based supercomputer is not fast enough to deal with this load. It might take such a computer many months to do all of those calculations when a grid system does this more quickly by taking advantage of many other computers whose full power is not being used.

The idea of using the power of geographically diverse home and office computers has the disadvantage of limited bandwidth. In order to remedy the issue, it has been suggested that the molecular models be partitioned and computations be duplicated such that the data from the processors overlap (Ware, 1997). Each processor would be given its own set of instructions with little to no knowledge of what other processors are doing. This approach, however, has one major obstacle. If atoms interact with each other in the areas where multiple processors are involved, those processors will have to be in contact to determine what the model really looks like. Still, creating partitions in molecular models with infrequent communication between processors is a significant challenge at present (Ware, 1997).

Two projects in Europe are using grid systems that are testing grid middleware and its use for molecular engineering. The BIOGRID (ICM, n.d.) is using computational grids to integrate software commonly used in chemistry and molecular dynamics, and those using this system have access to databases via the Internet. The OpenMolGRID project (OpenMolGRID, n.d.), started in September 2002, concentrates more on molecular design. This system has created thousands of new compounds that may lead to new drugs to fight cancer. The future of the integration of resources with grid systems could help make great strides in genomics, as well.

Besides creating new drugs, nanotechnology, or molecular nanotechnology (MNT), is a great motivator to create grid systems capable of handling lots of data. A design at that level could possibly include billions of atoms, which means a lot of memory and storage will be needed. Due to the complexity of problems it faces, nanotechnology has been described as trying to build a Lego machine while wearing big boxing gloves (Wikipedia, n.d.). Besides the positive items nanotechnology might bring, such as nanobots (see Fig. 4) or nanomedicine, scientists are prepared for the fact that not everyone may agree that nanotechnology is important or should be explored. Worse yet, there is a growing concern that such micro machines could be misused by parties engaging in unlawful activities.

Figure 4. Mite and nanogear

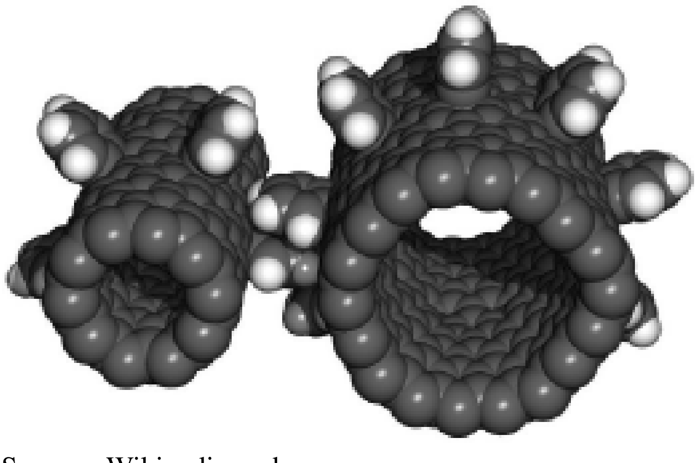


Source: Wikipedia, n.d.

4. Conclusions

Undoubtedly, the proliferation of scientific experiments and creation of new research sites will greatly increase the amount of data available for analysis and sharing. As of today the grid and distributed systems are still in their budding phase and some of them are not yet operational. Those already on-line still operate on software that is not well developed and is in testing stages for the most part. It appears that the systems of today are best suited for modeling calculations. However, when molecular modeling moves out of the realm of simple molecules and into the realm of multiple thousand atom systems (such as the one in Fig. 5), the number of calculations needed to be performed increases exponentially and in some cases factorially!

Figure 5. Nanogear at the molecular view



Source: Wikipedia, n.d.

The calculations for multiple interacting atoms need to be performed oftentimes on a femtosecond scale. The resulting range of probabilities is staggering and certainly no single PC or a mainframe can achieve this task in a reasonable amount of time (Meyyappan & Srivastava, 2003). The distribution of computing resources seems to be the most logical step towards the next supercomputer. That supercomputer might no longer be a bulky machine, running esoteric applications, but the friendly PC sitting on our desks, its strength lying in the sheer numbers.

References

Asadzadeh P., Buyya, R., Kei, C. L., Nayar, D., & Venugopal, S. (n.d.). Global grids and software toolkits: A study of four grid middleware technologies. Retrieved November 29, 2004, from www.gridbus.org/papers/gmchapter.pdf

Avery, P. (2002). Data grids: A new computational infrastructure for data-intensive science. Retrieved November 30, 2004, from www.griphyn.org/documents/document_server/uploaded_documents/doc--479--royalsoc_v7_with_figs.pdf

Erlanger, L. (2002). Distributed computing: An introduction. Retrieved November 28, 2004, from http://www.extremetech.com/article2/0,1558,11769,00.asp

Foster, I., Kesselman, C., & Tuecke, S. (n.d.) The anatomy of the grid. Retrieved December 1, 2004, from www.globus.org/research/papers/anatomy.pdf

ICM. Interdyscyplinarne Centrum Modelowania Matematycznego i Komputerowego. BIOGRID. (2004). Retrieved November 28, 2004, from http://bioGrid.icm.edu.pl/

Globus Toolkit. (2004). Globus Toolkit 3.0 FAQ. Retrieved November 29, 2004, from http://www-unix.globus.org/toolkit/faq.html

Green, M. L., & Miller, R. (2004). Molecular structure determination on a computational & data grid. Retrieved December 1, 2004, from http://drifters.ccr.buffalo.edu/grid/download/Molecular-Structure-Determination-Grid.pdf

Karelson, M. (n.d.) Molecular engineering and drug design. Retrieved December 1, 2004, from http://www.reuna.cl/redID/docs/docs_internacionales/karelson.pdf

Leach, A. R. (2001). Molecular modelling principles and applications. New York: Prentice Hall.

Meyyappan, M., & Srivastava, D. (2003). Carbon nanotubes. In Goddard, W. A. III, Brenner, D. W., Lyshevski, S. E., & Iafrate, G. J. (Eds.), *Handbook of nanoscience, engineering, and technology* (pp. 1-18, 26). Boca Raton, FL: CRC Press.

“Molecule.” *Microsoft Encarta Encyclopedia,* 2001.

NASA. (n.d.). The Hubble Space Telescope. Retrieved December 2, 2004, from http://hubble.nasa.gov/overview/intro.php

OpenMolGRID (n.d.). Retrieved November 27, 2004, from http://www.openmolGrid.org

SETI@home. (n.d.). Retrieved December 1, 2004, from http://setiathome.ssl.berkeley.edu/windows.html

Ware, Will. (1997.) Distributed molecular modeling over very-low-bandwidth computer networks. Retrieved November 28, 2004, from http://www.foresight.org/Conferences/MNT05/Papers/Ware/index.html

Wikipedia. (n.d.) Molecular Nanotechnology. (2004). Retrieved December 1, 2004, from http://en.wikipedia.org/wiki/Molecular_nanotechnology