Internet, GRID, Self-adaptability and Beyond: Are We Ready?

Petre Dini[1], Wolfgang Gentzsch[2] , Mark Potts[3],
Alexander Clemm[1], Mazin Yousif[4] and Andreas Polze[5]

[1]Cisco Systems, Inc. USA, [2] MCNC – RTP, USA,
[3]Hewlett-Packard Corp., USA, [4]Intel Corporation, USA, [5]University Potsdam, Germany

pdini@cisco.com, wgentzsch@mcnc.org, mark.potts@hp.com,
alex@cisco.com, mazin.s.yousif@intel.com, andreas.polze@hpi.uni-potsdam.de

Abstract

This paper reflects different understanding and positions on future trends of GRID-oriented technologies, applications, and networks, as perceived by representatives from industry and academia. There is no definitive answer on the topic that is raised in the title. Instead, the question serves as a starting point for a discussion of methodologies to apply, of engineering requirements to consider, of technical challenges to resolve harnessing this next stage of evolution of the Internet. The panel guests present their perception on these issues. Is GRID the Internet-ng? What do GRID and self-adaptable applications have in common? Does GRID facilitate the adaptability of applications? Where would the Internet, as it exists today, fall short? Will Web Services come to the rescue? Are there new paradigms associated with heavily shared resources (infrastructure, protocols, services, user communities)? Which major bumps are ahead the roads for self-adaptability over GRID, how can they be crossed, at what cost, and what new paradigms are associated along this road? These are all open questions with no definitive conclusion, discussed from the point of view of what constitutes the state of the art today. Responses are intended to stimulate thought and foster discussion, rather than being a comprehensive treatise of the subject.

1. Introduction

The paper outlines salient topics and challenging questions that this position panel paper deals with. The authors express their perception from the point of view of organizations and communities they are belonging to. For the benefit of the audience, the positions were not synchronized before this paper was released.

The panel report paper is organized as follows. Section 2 presents self-adaptive applications on autonomic GRID, as developed at MCNC (Wolfgang), followed by a Section 3 on GRID evolution and metadata (Andreas). Section 4 brings into the discussion the aspect of partial and intermittent (PI) resources in GRID computing (Petre). Section 5 focuses on utility computing considering GRID concepts (Mark). Self-adaptability in the GRID is presented in Section 6 (Mazin), while challenges of self-management and self-adaptability are discussed in Section 7 (Alex). As conclusion, a summary of open issues is given.

2. Self-adaptive applications on Autonomic GRID

The Internet primarily enables the provisioning, accessing, sharing, exchanging, and managing of INFORMATION. This information often is available statically via hyperlinks, websites, plain text, and more. The Internet itself does not automatically handle this information. Instead, typically a person provisions, accesses, and manages this information that may be accessed by the end user. Thus, the Internet is principally a network with static resources with information flowing directly from the producer to the consumer. Because of the static nature of the Internet, applications (mostly information) cannot easily adapt to changing network or server conditions.

In contrast, far beyond the capabilities of today's Internet, the GRID enables provisioning, accessing, sharing, adding, removing, and managing of RESOURCES, including computers, storage, network bandwidth, databases, and software applications. All these resources are dynamic components of the GRID and can be fully managed automatically through policies and end user interaction.

Thus, a GRID can be programmed to perform certain tasks, for example computer simulations and database searches, to automatically include more resources when needed, or to broker resources according to users' requirements, which may include time, money, priority, security, and capability.

Any GRID can be compared with a community of people addressing a specific need or task.

Resources are distributed, heterogeneous, coordinated, and managed. Each resource is intelligent, specialized, can communicate, and above all, can perform a task as part of a larger solution. This human "GRID" usually is extremely self-adaptable. In situations where someone suddenly doesn't perform as expected, such as being absent due to illness, the whole project often slows down at first. Then, someone else fills the gap to bring the project back on track. The community adapts and provides self-healing and the necessary support to achieve the goal. The entire process realigns itself along the new infrastructure.

A GRID of computing resources ideally behaves exactly the same way. To perform a specific task, such as a complex simulation job, many different computing resources in a GRID are working together in a coordinated and managed way. As in a human "GRID", the computing GRID can dynamically change for a multitude of reasons. To guarantee continuous execution and accomplishment of the task, the GRID has to adapt (heal) itself automatically to any change, without affecting the (execution of the) application.

If an Internet information server crashes, we will not be able to access information that resides on the server. The only option is to wait until the server is up again. In a GRID, if a server goes down, this will immediately be recognized by the GRID master node or by an intelligent peer node, and the peer or another healthy node will be brought in to the GRID to take over. From the perspective of the end user, the entire process is seamless and transparent.

The GRID resource infrastructure provides the self-adaptable capabilities. However, as with "human" GRIDs, in addition to specializing and training the intelligent human resources needed for a certain project, we also have to build intelligence into the projects themselves to optimally adapt to our human requirements. The wide field of mechatronics is full of great examples. Therefore, applications themselves should also be intelligent, aware, and self-adaptable to be optimally efficient in a GRID environment.

Originally, none of today's applications were designed from the ground up to run on a GRID. However, for most of them, only slight modifications are required to make them "GRID-ready". Basically, what's needed is adequate information about the resource needs of the application to efficiently run on the GRID. Such information can be:

- Number and type of servers (single desktop, many workstations, parallel supercomputer, visualization server, etc.)
- How much memory will the application need
- Type of operating platform (Linux, Solaris, Windows, etc.)
- The location of the application software and input data
- The requirements of when results are needed – such as whether this is an urgent task or whether the results can be produced in a week
- The security requirements – such as whether the application is highly confidential, top secret, or can be processed using any computer on the Internet
- The determination of whether the application depends on another application which has to be processed first.

Today, this information is provided by the user who wants to run this application job, in a quite cumbersome way, usually by providing a lot of details via a job description file (JDF) or resource description language (RDL). But what if the application changes during runtime? For example, if one process suddenly spawns hundreds of sub-processes which need to run on hundreds of new resources? Or the problem size changes and no longer can be adequately addressed on just one computer? To efficiently handle these kinds of applications in a GRID, the applications themselves have to take care of identifying and selecting the resources they need.

3. GRID Evolution and MetaData

About 25 years ago, a major transition from closed, proprietary mainframe computer systems to distributed computing environments took place. This transition was mainly driven by the introduction of personal computers (PCs) and affordable workstations. Standards in the old days were set by hardware vendors, systems tended to be closed, well maintained, and highly available - at a price.

The PC age was different: standards were set mainly by application software. Systems tended to be loosely connected, they were generally less-than-perfectly maintained and users had to learn to live with glitches, occasional re-boots and low system availability. This situation seemed acceptable because the price for computing power was so much lower than in the old days.

Today, we are seeing the next big transition. This time, it's all about middleware - the software layer between hardware/operating system and the application software. Future systems will be based on open interconnection standards, such as WebServices and there will be an infrastructure provided that allows for distributed job placement, management, and scheduling. Pricing for computing power is no longer based on the prices of individual soft- and hardware, but rather on a service metaphor. This new, promising computing world is commonly referred to as "The GRID".

However, if you look back, there are still some problems left unaddressed: while the service metaphor fits well to the computing needs of many users, non-functional system properties, such as availability, reliability, fault-tolerance, timing behaviour, or security - which were taken for granted in the old days and were lost during the transition to distributed, PC-based systems - have to be re-established for the GRID. Besides standard formats for describing its services (or functional interfaces), GRID computing needs a standard metadata format for describing the user's expectations regarding a GRID service's non-functional properties (or attributes).

Predictable computing on the GRID is the next big challenge. We envision a layered approach towards this goal. Technologies such as the semantic web and ontologies will be used to allow the user to specify her needs using the service metaphor. The user will attach metadata, such as a deadline, security requirements, or a maximum price to his service request.

Technologies for service discovery and composition will be used to figure out whether a user's request can be satisfied directly from the set of existing GRID service. This process will again take into account the user's non-functional requirements (i.e., the service's metadata). Service requests from the user will be broken down into series of smaller requests; it might even be envisioned that certain stages in such a process require the generation, compilation, and deployment of new services on the GRID (i.e., service creation).

A workflow description, the result of the above-mentioned process will then be fed into adaptive workflow engines. Again, the important difference to today's approaches is the ubiquitous nature of service metadata. This metadata will allow for adaptive process management, using re-execution in time or space as a technique for ensuring service availability.

Finally, a series of jobs will be handed over to the GRID. These job requests will be attributed to allow for placement and scheduling by the underlying GRID infrastructure. An application's non-functional properties, such as resource requirements, security settings, timing requirements, and fault-tolerance assumptions have to be mapped onto the underlying operating systems mechanisms. The challenge here is to interconnect today's crude GRID scheduling mechanisms with the more elaborate concepts found in server operating systems.
Actually, the problem is two-fold, as we encounter two different breeds of GRID computing infrastructures: High-Performance or Cluster computing on the GRID is focusing mainly on execution of parallel, communicating tasks on multiple nodes in the GRID, assuming that those compute nodes are dedicated to GRID computing. This is indeed valid for many scientific environments. Job placement and scheduling for this class of environments is relatively good addressed today. Idle time computing - the other big motivation for the GRID – faces more severe obstacles. Although many organizations have huge amounts of PCs and workstations sitting idle for significant portions of the day, users are often unwilling to donate processing capacity of their machines to the GRID. The reason here is that it is very difficult to establish contracts between the user and a GRID job, saying what amount of resources the job may acquire and how long it might take to adapt the job's behaviour to changing requirements of the interactive user. To make idle computing resources available to the GRID, further research on how to integrate metadata-based GRID scheduling with the corresponding operating systems mechanisms is needed.

A standard metadata format will be the red line interconnecting all the stages of service provisioning on the GRID. On the infrastructure level, certain framework-specific metadata representations have already been established (such as those in a job description file), however, higher layers, such as the adaptive process management still have to be extended towards that direction. Using metadata to provide predictable computing services on the GRID - resembling the "good old days" - will be the ultimate goal.

Closely related to the mechanisms supporting heavily shared resources is the notion of partial and intermittent resources (PI-resources).

4. PI-Resources

Essential in this realm is the paradigm shift occurring in the last decade in telecommunications and networking considering partial and intermittent resources (pi-resources) [1]. Internet, converged networks, delay tolerant networks, ad-hoc networking, GRID-supporting networks, and satellite communications requires a management paradigm shift that takes into account the partial and intermittent availability of resources, including infrastructure (networks, computing, and storage) and service components, in distributed and shared environments.

A resource is called partial (p-resource) when only a subset of conditions for it to function to complete specification is met, yet it is still able to provide (potentially degraded) service, while an intermittent or sporadic resource (i-resource) will be able to provide service for limited and potentially unpredictable time intervals only. Partial and intermittent services are relevant in environments characterized by high volatility and fluctuation of available resources, such as experienced in conjunction with component mobility or ad-hoc networking, where the notion of traditional service guarantees is no longer applicable. Other characteristics, such as large transmission delays and storage mechanisms during the routing, require a rethinking of today's paradigms with regards to service assurance and how service guarantees are defined.

Several aspects and challenges in defining, deploying, and maintaining partial and intermittent resources that may collocate with traditional resources have been identified. The pi-resources can support new types of applications, and may require semantics, models, and associated management mechanisms. Most of the currently known paradigms may be revisited in the light of pi-resources.

Pi-resources are present in ad-hoc, sensor, and overlay networks, as well as in co-operative and adaptive applications. It is estimated that implications in several areas are unavoidable: (i) on current communication protocols, security, and middleware, (ii) on QoS, and traffic modelling, (iii) on the architecture of network devices and networks, and (iv) in intermittent GRID services and architectures.

Other well-known mechanisms may require certain adaptation: (i) traffic analysis under sporadic data transfer and (ii) service level agreement (SLA) for partial and intermittently available behaviours. Additionally, procedures to identify and discover pi-resources may differ from the classical ones: (i) adaptive time-out discovery mechanisms, (ii) hybrid sequential and parallel, and (iii) new semantic of high availability.

Management of pi-resources faces additional challenges when considering (i) context-aware resources, (ii) user behaviour, (iii) autonomic pi-components, and (iv) management of mobile pi-resources, including accounting fault/alarm processing, performance evaluation, metering, etc.

Managing PI-resources refers to all kind of issues pertaining to bandwidth allocation, policy-based operations, service monitoring, intelligent architectural systems, mobility and wireless, protocol aspects and performances across heterogeneous domains. This brings in to the picture the next topic on utility computing.

5. Utility computing and GRID

Utility computing (UC) is a new paradigm dealing with heavily distributed on-demand resources. The UC model for computing is defined as a service oriented architecture (SOA). This allows supporting (i) federation, composition & incremental evolution, (ii) the separation of concerns by clearly defined roles relative to horizontal & vertical domains, (iii) policy driven management decisions, where constraints are explicit and modifiable vs. implicit and immutable, (iv) combination of model centric and model agnostic design with a common meta-model enabling leverage & interoperability, and (v) solutions oriented approach, supporting "composable" solutions (building blocks).

1. The architecture is defined as a service oriented architecture (SOA) This principle is driven from the realization that the architecture cannot mandate a "green field" approach to utility computing. There is a broad range of existing management technologies at multiple levels that can and should be leveraged into utility computing solutions. The architecture must also support the federation, composition and incremental evolution of utility computing solutions. The best way to support this goal is through a SOA, which will allow greater levels of integration and alignment, between the compute utility and its consumers, as those consumers themselves adopt a SOA as the way in which they develop, deploy and expose their business processes.

2. The architecture supports the separation of management concerns This principle is driven by the recognition that management capabilities may be specific to either horizontal or vertical domains. Thus we must embrace, define, and standardize manageability as it pertains to the different technologies, platforms and the roles they play in support of UC.

3. The architecture is defined to be policy driven The goal of this principle is to enable the compute utility to provide services that meet the requirements and objectives of the consumers, through policy driven management. The integration and coordination of manageable IT resources is accomplished through the specification of policy. This enables the constraints on the utility computing services to be explicit and modifiable rather than implicit and immutable.

4. The architecture is model centric but also model agnostic To enable the effective delivery of UC solutions, manageability itself needs to be "virtualized" in terms of the management models and semantics associated with manageable resources and management applications. Without this virtualization, the ability to aggregate or compose manageability across many different resource models to the business process layer, at which the business wants to manage, will be restricted in terms of time and cost. This principle does not mean that there is no management model but dictates a management meta-model that allows may other models to plug into (even if this requires some mediation and manipulation).

5. The architecture is solutions oriented (supports composable solutions) There are many point products on the market today for management in specific domains or targeted at specific resource types. This architecture must facilitate the composition of the appropriate management services through different types of relationships (composition / encapsulation and collaboration / interoperation).

The management processes through policies allow for the separation of the process of management from the individual capabilities of the management services in the management system. What is the role of policies as external decisions versus self-adaptability in GRID is the next question.

6. Self-adaptability in the GRID

Autonomic GRID applications that are self-adapting, self-optimizing and self-configuring should be a goal to achieve full-fledged GRID adoption. Such goals are needed because of two main reasons: (i) applications could be multi-phased, dynamic and heterogeneous requiring large number of software components with complex interaction between them; (ii) the GRID infrastructure is heterogeneous, dynamic and changing due to the aggregation of large number of independent compute, storage and networking resources. These two factors do put a major burden on applications development, *composition dynamism* and *management* to achieve the full promise of the GRID and to maintain applications QoS.

Let us for the time being focus on self-adaptability to hanging environments only and how such adaptability could be achieved. Adaptability needs to be enforced at multiple levels, from the applications to the compute, storage, and networking resources to the management infrastructure to the middleware.

Applications need to be designed to dynamically adapt themselves to handle changes such as resource unavailability, networking latency and bandwidth variations, and link failures. One basis for adaptability and self reconfiguration is software architecture models that are maintained at runtime and used as basis for adaptability. A three-layer software architecture has been proposed to accomplish adaptability and self-repair, namely, the *runtime layer*, *model layer* and *task layer* [3]. The runtime layer includes applications, OS, middleware and monitoring software; the model layer is responsible for interpreting system observations; and the task layer is responsible for setting overall system objectives such as when an application should execute based on performance goals or resource availability. Current software architectures lack, or possibly include limited such similar capabilities.

One major requirement is that applications need to be developed with open standards and constructed around middleware software that enables different technologies to smoothly and transparently interact with each other. Also, web services providing universal access to information and computing in a collaborative environment are key to fulfil the vision of GRID. This is a hurdle companies will be unwilling to pass unless they envision a considerable return on investment.

Resource adaptability and flexibility is another hurdle. Resources here refer to the pools of compute, storage and networking resources. For compute, self-adaptability exhibits itself in ability to increase processing power when workload demands increase. This could be through additional hardware threads, processor cores and complete processors. It should also include the ability to perform processors hot add/remove with the least disruption possible. For networking, self-adaptability could include bandwidth adaptability such as resorting to wider or faster links when bandwidth requirements increase. Self-adaptability could also include latency adaptability, especially for applications that cannot tolerate long or variable response latencies. Both bandwidth and latency will require intelligent resource scheduling algorithms, in addition to comprehensive management infrastructure to support adaptability. The InfiniBand Architecture [4] is one such network that promises a great deal of self-adaptability. For storage, self-adaptability could include on-line capacity expansion and storage virtualization. Further, resource adaptability in accordance to data locality also needs to be addressed.

Comprehensive interoperable management infrastructure is a major hurdle that requires a great deal of attention from researchers, as well as companies. We consider management the area that requires the most to achieve any GRID vision; let alone GRID with self-adaptability. Management infrastructure needs to be built with open standards to enhance interoperability, and includes network, storage, compute as well as software management. Distributed management models along with a combination of intelligent rule- and policy-based monitoring algorithms will be needed. Efforts from industry-wide associations such as the Distributed Management Task Force (DMTF) and the newly formed Enterprise GRID Alliance (EGA) are definitely going in the right direction, but there is a great deal to be done.

Self-adaptability may very likely put burden on security, which must not be compromised when adapting any number of resources due to workload changes or resources faults.

Virtualizing resources and the wide deployment of virtual machines, along with the ability to add/remove resources of virtual machines in accordance to workload or failure demands, also need to be in place for the full vision of GRIDs.

7. Self-management and self-adaptability: The end of management?

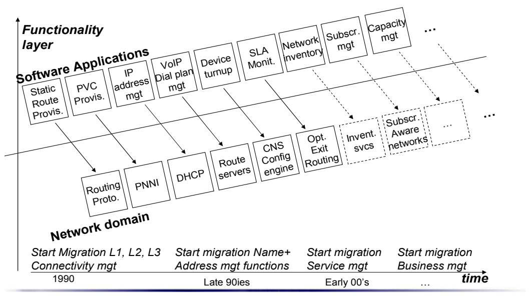
Self-management and self-adaptability have recently become very popular buzzwords throughout the networking and computing industry. It follows the realization that a significant portion of the total cost of ownership of complex systems, specifically distributed systems, is typically related to their operation, administration and maintenance, and that in some cases automation of these tasks is a prerequisite to make certain applications and services feasible in the first place, such as in the case of GRID. As systems grow more powerful, so does their complexity, which compounds the situation. The goal, therefore, becomes to allow users to reap the benefits of the power of such systems, while at the same time not requiring them having to deal with their internal complexity. The general idea is to make the systems "self-sufficient" by using some of their power to implement intelligent capabilities that allows the systems to self-tune, self-heal, and self-manage. The theme is the same, whether the systems constitute networks or networked application infrastructure, the Internet or the GRID: more intelligence is pushed into the systems, intelligence that formerly resided in outside entities such as management applications or human operators. This means that fundamentally control loops that used to be open (i.e. involve multiple parties) are getting increasingly closed.

While this is clearly a concept that makes a lot of sense, contrary to the impression that is given by the current hype, it is hardly all that novel – precisely because it does make so much sense. Migration of intelligence from outside operations and applications into the systems themselves is a theme that has been occurring constantly for decades. As this migration occurs, functions that were previously clearly related to operations and management become a part of the system's function itself. In fact, they become immersed so much that consequentially it becomes hard to imagine them as anything else. Consider for example routing: instead of provisioning static routes across a network, routing protocols today perform this function in an automatic, resilient, "self-adapting" and "self-managing" way. Another example is the public switched telephone network, where connections are automatically switched by the network instead of needing to be set up and torn down by an operator. The examples are taken from networking; the situation for GRID is essentially the same, the difference being the types of entities that are being subjected to management and that the GRID starts out at a higher level of advanced control features than was originally the case for networking.

All this implies is that when speaking about self-adaptability and self-management, what is really meant is that functions that are today associated with open control loops are migrated into the systems. The reference point of what constitutes "self-management" and "self-adaptability" changes over time; a system with functions that would be considered self-managing today may not be considered self-managing tomorrow. That is the other observation: while more and more functions and intelligence have migrated into the underlying systems, the need to manage those systems from the outside has not gone away. However, was has changed is the particular function of management, and the level at which it occurs. For example, instead of managing devices, it is now services that are being managed. Instead of monitoring system components, it is service levels that are being monitored. Instead of being concerned with tuning low-level system parameters, what is tuned are higher-level system parameters that might be associated with business policies. As some of those tasks move into self-adapting, self-managing systems, other tasks that take management to a higher level will occur. In fact, new, higher-level functions to manage and keep the systems under control may even be necessary, as the new level of self-management and self adaptation enables the next level of complexity of the underlying system. Again, historically this is nothing new: for example, as the public switched telephone network moved from manual to automatic switching, unprecedented and hitherto unthinkable scales of the network became possible along with entirely new types of services, such as 1-800 and E.911

services. While the task of automatic switching went away, it was now dial plans and their distribution that needed to be managed.

One aspect that perhaps is new about the current wave of self-adaptability and self-management is the attempt to address it using general-purpose architectures that are potentially applicable to many different functions. Past approaches have been more geared towards individual point problems, addressing functions in a custom-tailored way one at a time. It will be interesting to see how this will in practice play out – whether indeed one-size-fits-all approaches will prove to be generic enough to truly address broad categories of self-management, or whether we will see a return to narrower solutions. A possibility that is quite likely is that general-purpose approaches will become part of a middleware that will be absorbed by operating systems, which then better enable special-purpose functions that will still be built on top. One reason for optimism that further progress will be made is that control theory provides the field with a solid theoretical underpinning and hence footing to stand on.



Migrating Intelligence into the Networks

So, are we ready for a new age of self-management and self-adaptability? We are undoubtedly ready to take systems to the next level from where we are today, as has happened for wave after wave after wave before. We are probably also ready for some GRID applications. But we are by no means done, nor will the need to manage or interfere with complex systems go away. We should have no illusions that self-management and self-adaptability are a journey, not a destination. As we climb the current peaks, we will find that others open up behind it.

8. Conclusion

A GRID-enabled application has to be intelligent, dynamic, adaptable, scalable, aware, and robust, like any other resource in the GRID. To accomplish this, there have been exciting developments recently of so-called GRID Application Frameworks, which help users to GRID-enable their applications. One example of such a framework is CACTUS, which is an open source, problem solving environment designed for scientists and engineers [2].

It is acknowledged that job placement and scheduling for this class of environments is relatively good addressed today. Integrating metadata-based GRID scheduling with the corresponding operating systems mechanisms is needed, including adaptive process management. Using metadata to provide predictable computing services on the GRID - resembling the "good old days" - will be the ultimate goal.

In this context, considering partial and intermittent services may lead to a more realistic virtualization of GRID resources. In this case, revisiting concepts such as SLA/QoS, user-profile, SLA violation, traffic optimization, and user-driven self-management networks and services it appears to be a must.

The technical demands of GRID architectures require increasing amounts of adaptability, intelligence, collaborative ability and mobility to provide continuous operation and availability of resources. Does this mean that we don't longer need management as we all know it today?

Therefore, we are undoubtedly ready to take systems to the next level from where we are today, as has happened for wave after wave after wave before. But we are by no means done, nor will the need to manage or interfere with complex systems go away. We should have no illusions that self-management and self-adaptability are a journey, not a destination. As we climb the current peaks, we will find that others open up behind it. The solution that once and for all puts to rest the need for systems to be managed, tuned, and adapted by an outside entity is not yet in sight, although of course we are gradually pushing the level at which this occurs higher.

9. References

[1] P. Dini et al., Service Assurance with Partial and Intermittent Resources [SAPIR], Manifesto, in LNCS-3126, Springer Verlag, 2004

[2] Cactus, [please see www.cactuscode.org]

[3] S. Cheng, et al., "Software Architecture-based Adaptation for GRID Computing," Proceedings of the 11th International Symposium on High-Performance Distributed Computing (HPDC), 2002

[4] www.infinibandta.org, The InfiniBand Architecture (IBA) Specification, 2002

[5] A. Clemm and G. Lin, Rethinking Manageability - Advances in the Management of IP Networks. Tutorial, IFIP/IEEE 2004 Network Operations and Management Symposium (NOMS), Seoul, Korea, April 21, 2004.