# Open Internet-based Sharing for Desktop Grids in iShare

Xiaojuan Ren[1], Ayon Basumallik[1], Zhelong Pan[2], Rudolf Eigenmann[1]
[1]School of ECE, Purdue University [2]VMWare Inc.
West Lafayette, IN 47907 USA Palo Alto, CA 94304 USA
{xren,basumall,eigenman}@purdue.edu zpan@vmware.com

## Abstract

*This paper presents iShare, a distributed peer-to-peer Internet-sharing system, that facilitates the sharing of diverse resources located in different administrative domains over the Internet. iShare addresses the challenges of resource management in desktop grids, and integrates these resources with production grids. In this paper, we present a brief overview of the iShare system and describe how iShare leverages existing standards to provide novel solutions to the problems of resource dissemination, resource allocation and trust in desktop grids. We also discuss how iShare integrates production grid systems, such as the Teragrid, with desktop resources and compare the iShare approach with web-based user portals for production grids. To quantitatively evaluate our techniques, we measured the efficiency of resource allocation in iShare and the overheads associated with establishing trust and providing the iShare user interface for production grids. The evaluation results demonstrate that iShare enables open Internet sharing with efficiency, reliability, and security.*

## 1 Introduction

Revolutionary advances in networking technologies have made it possible to integrate computational and information resources scattered over the Internet into cohesive computing systems. Among the infrastructures supporting various kinds of Internet-wide collaborations, distributed cycle-sharing systems – systems that gather compute power from a large range of machines – have shown success through popular projects such as SETI@home. These projects have attracted a large number of participants who contribute idle CPU cycles on home PCs to a scientific effort.

Distributed cycle-sharing systems have been referred to as *desktop grids*. In this paper, we will consider desktop grids more specifically as systems that address the challenges of (i) managing the resources of cycle-sharing systems and (ii) integrating *ad-hoc* and dedicated grid resources. Examples of dedicated grid resources are those provided by the Teragrid [6].

**Resource management in desktop grids:** The large-scale deployment of desktop grid resources require management mechanisms that are correspondingly scalable. Current desktop grid systems are limited in that they have centralized structures with participants providing idle compute cycles to tasks farmed out by a server. Another challenge arises in that desktop resources are usually non-dedicated and thus present *fluctuating availability* to remote applications that attempt to use these resources. Furthermore, Internet users are usually unknown to each other, implying a lack of trust among resource providers and application users. Common authentication techniques, which use a central authority to verify user identities, are insufficient to provide protection among these untrusted entities.

**Integrating desktop and dedicated grid resources:** It is desirable to integrate production grid resources with external resources. Doing so, exposes scientists to a plethora of possibilities in selecting and managing access to computational platforms and also hides the complexity of different access mechanisms for grid and desktop resources.

This paper presents an open Internet-sharing system, called *iShare*, that addresses these challenges. iShare's resource management is based on a decentralized P2P-based framework, where participants can play the roles of both providers and users [25]. Resource providers can easily describe and publish their resources and usage policies; end users can easily browse and access published resources. These functionalities are provided as a set of tools in the user interface to the iShare system, which is implemented as a lightweight Java application. The assignment of user applications to resources is guided by a proactive resource allocation method, which predicts resource availability using a semi-Markov model [24]. iShare allows untrusted users to access resources, but confines the execution of their applications in virtual machines (VMs) [22].

1-4244-0910-1/07/$20.00 

iShare leverages existing resource description standards in open Internet-based sharing to handle a broad spectrum of hardware, data and applications resources. iShare formalizes basic resource features to represent common operations of widely used local management systems. This formalization can be extended by adding new resource semantics and modules to process these semantics. We will present such extensions in integrating dedicated grid resources with desktop systems. We shall examine iShare in the context of *grid portals* – the other common interface for dedicated grid resources.

The contributions of this paper are –

- We present the iShare system and describe its techniques that allow desktop grid resources to be described, shared and accessed easily.
- We describe techniques used in iShare for interfacing desktop grid with production grid systems

We evaluate these contributions in terms of improved application response time, achieved by our proactive resource allocation techniques, and overheads incurred by our VM and the file transfer techniques. The rest of this paper is organized as follows – Section 2 provides an overview of the iShare system and describes how iShare enables open Internet-based sharing for desktop grids. Section 3 presents the iShare extensions for interfacing with production grid systems. Section 4 describes the related work of iShare.

## 2 Overview of the iShare Internet Sharing System

iShare extends the benefits of Internet-based sharing to grid computing systems. Providers (users who wish to make their resources available) can easily post resource features, and others can discover and use these resources as if they resided on their desktop. Among the motivations for the iShare design were to solve some of the weaknesses observed in prior work with the Purdue University Network Computing Hub (PUNCH) [20]: adding new resources needed to go through a central point of management, and the requirements for adding new machine resources at times conflicted with the administrative rules of their owners. iShare removes these barriers by allowing resource providers to describe and publish their resources in an open manner: composing resource descriptors and posting them on the Web. To make the process easy, iShare provides rich tools for resource description, resource publication, and resource access management. End users can search for specific resources and invoke a published software program in a batch or interactive mode.

iShare supports three types of resources at an equal level: software services (application programs), service platforms (machines), and data. Available resources are organized into a hierarchical structure, grouping together semantically-related information. A thorough description of iShare's resource management methods is beyond the scope of this paper. In the rest of this section, we present the techniques that address the challenges of managing desktop resources: (1) the decentralized architecture, (2) the proactive resource allocation, and (3) the protection of resources from untrusted Internet users.

### 2.1 Decentralized Architecture for Resource Dissemination

Resource dissemination entails the publication and discovery of resources. The distributed and dynamic nature of desktop resources suggests a decentralized organization. iShare realizes decentralization via the integration of the Web infrastructure and a peer-to-peer (P2P) system: a provider of resources can describe their features and post their availability on any web page; meta-data are extracted from the resource descriptors and are inserted to a P2P overlay network that enables efficient resource discovery.

iShare adopts the Resource Description Framework (RDF) [18] as the description language for resource semantics. RDF is an semantic-oriented language for describing information contained in a Web resource. We apply RDF to describe basic resource features as a set of attributes. The descriptions are processed by a RDF parser and are used to invoke the corresponding remote access methods. In adding "non-standard" resources, new resource features are included by inheriting from an existing RDF schema; the associated access methods are integrated as *plug-ins* to iShare's software modules.

To enable the efficient search for resources with specific capabilities, iShare organizes resources into a hierarchical name space and distributes the name space to the underlying P2P network. An item in the hierarchical space is mapped to the peer node with `nodeId` closest to the hashing value of the item's prefix path. A child's path name is kept in its parent's repository. Resources in an application discipline can be discovered incrementally by traversing the hierarchical tree from its root. The current implementation of iShare's P2P network is built on a structured overlay, Pastry [26]. Each Pastry node has a unique, uniform, and randomly assigned `nodeId` in a circular 128-bit identifier space. Given a message and an associated 128-bit key, Pastry reliably routes the message to the live node whose `nodeId` is numerically closest to the key. This design enables resource discovery without requiring any knowledge of where the corresponding data items are stored. More details about resource discovery in iShare can be found in [25].

## 2.2 Resource Allocation with Availability Prediction

In cycle-sharing systems, resource owners voluntarily share CPU cycles only if they incur no significant inconvenience from letting a foreign job (*guest process*) run on their own machines (*hosts*). For guest users, the free compute resources come at the cost of highly fluctuating availability. The primary victims of this volatility are large compute-bound guest applications, most of which are either sequential or composed of several tasks as a group and must all complete before the results can be used. Therefore, response time rather than throughput is the primary performance metric for such compute-bound jobs. To improve response time in the presence of fluctuating availability, we have developed proactive resource allocation. This approach predicts resource availability from history and assigns user applications to the resources with maximal available computing capabilities.

We have designed prediction techniques that achieve high accuracy as well as efficiency appropriate for online use [24]. In our techniques, we applied a semi-Markov Process (SMP) to predict the *temporal reliability*, which is the probability that a machine will be available during a given, future time window. This algorithm does not require any model fitting, as is commonly needed in linear regression techniques. To compute the temporal reliability on a given time window, the parameters of the SMP are calculated from the host resource usages during the same time window on previous days. A key observation leading to our approach is that the daily patterns of host workloads are comparable to those in the most recent days [23, 21]. In previous work [24], we evaluated our prediction techniques in terms of accuracy and efficiency. The results show that the prediction achieves the accuracy above $86.5\%$ on average and above $73.3\%$ in the worst case. The SMP-based prediction is also efficient, increasing the completion time of a guest job by less than $0.006\%$.

We have applied the SMP-based prediction in iShare's proactive resource allocator. In the proactive algorithm, the predicted availability is factored in the estimation of job completion times, guiding the allocator to select resources with both high computation capabilities and high reliability during the execution of a given task. We compared the proactive approach with an algorithm that ranks resources by CPU clock rate, which is used in the Condor system [28]. Figure 1 shows the results of the comparison.

The metric used in Figure 1 is the relative slowdown of the allocation strategies by comparing them to an *omniscient* algorithm, which has full knowledge of resource utilization in the future. To derive this metric, we first collected the average makespan of all the jobs scheduled and finished (including those failed and then restarted) using a specific allocation method, and then compared the value to that obtained by the omniscient algorithm. There are two sources for the slowdown: the ineffectiveness in selecting the best resource and the computational overhead of the allocation algorithm. The omniscient algorithm can make the perfect selection knowing future traces and its overhead is set to zero to serve as the baseline. The Condor-like algorithm is computationally fast, but its resource selection ignores the resource availability. This results in undesirable failures and restarts of guest jobs, and, as a consequence, the slowdown as high as $22\%$ (for jobs of 4 hours). Our proactive algorithm outperforms the Condor-like algorithm in all the cases, improving the average makespan up to $14\%$.

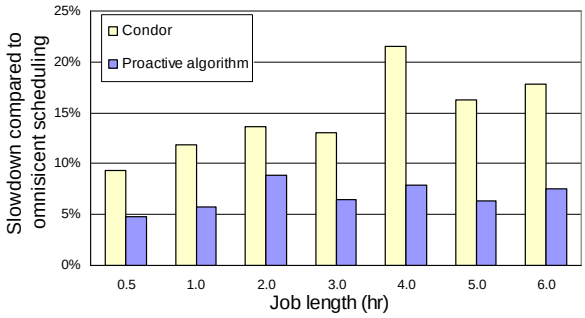


**Figure 1. Slowdown of jobs under different scheduling algorithms. The baseline is the omniscient algorithm, which has full knowledge of host resource utilization in the future. Lower bars are better.**

## 2.3 Protection in the Presence of Untrusted Internet Users

iShare applies virtual machine [19] (VM) technologies to confine the execution of applications that are potentially malicious. Virtual machines enhance system security by providing strict isolation between the host and guest jobs. However, they incur significant overhead for communication-bound applications. Also, setting up a virtual network typically involves root privileges, which is not feasible in an open Internet sharing system. To solve these problems, we have developed fast virtual network techniques that can be deployed at the user-level [22]. This section presents the user-level virtual machine approach to MPI program execution on distributed resource nodes in iShare.

We deployed VMs at the user level, building on User Mode Linux (UML) [8] VM techniques. We developed an efficient *socket virtualization* technique to improve the virtual machine communication performance of the original UML network solution. The key idea in our approach is to realize guest socket functions directly via host socket functions. In our VM system, socket system calls from

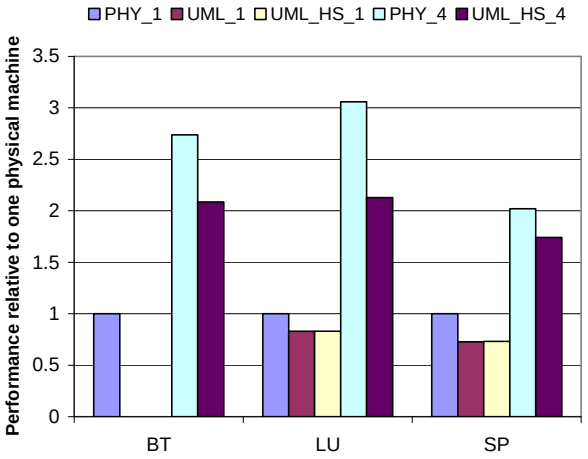


**Figure 2. MPI program performance relative to one physical machine. The metric is wall-clock_time_on_one_physical_machine / wall-clock_time, which reflects execution speed. Higher bars are better. PHY_1: one physical machine; UML_1: one virtual machine of original UML; UML_HS_1: one virtual machine of UML with host socket support; PHY_4: four physical machines; UML_HS_4: four distributed virtual machines of UML with host socket support. BT can not run on one VM because its memory requirement exceeds the capacity of one original UML virtual machine.**

a guest process are intercepted by UML's tracing mechanism. These calls are diverted to our socket implementation, which includes the socket system calls and all the file access functions. This approach leads to lower overhead of virtualization compared to the original UML approach. Furthermore, because VMs are connected in the same way as physical machines, it eliminates the need of system administration steps, as required by the original UML to dynamically allocate new IPs for VMs, for setting up VM networks.

We have measured the network performance of our VM implementation. In this paper, we listed the results of executing MPI programs on our virtual network. More experiments and results can be found in [22]. We measured the MPI program performance by the speedup relative to program execution on one physical machine. we ran three applications in the NAS parallel benchmark suite [3]: BT, LU and SP, on four 1.5 GHz Pentium IV machines in a 100 Mbps LAN. Figure 2 shows the results.

The figure measures scenarios possible without root privilege. The UML_HS_4 bars are measured by running the programs on four VMs, each of which resides on a different physical machine. The VM is the UML with our host socket support. The applications run faster on four VMs with the host socket support than on one physical machine. Our solution also achieves a reasonable speedup relative to the execution with one VM. Comparison of the results for PHY_4 (four physical machines) and UML_HS_4 shows that running MPI programs on four VMs is slower than on four physical machines. Note that running on 4 processors with original UML would require root privilege and hence is not feasible. The performance overhead has two sources: (1) UML itself introduces overhead, indicated by the bar of UML_1. (2) Our socket communication has larger latency than the physical case, due to the cost of virtualization.

We conclude that the performance degradation experienced by an MPI application, relative to the speed on the physical host, is acceptable. In return, our VM solution gains a higher degree of guest isolation and customization.

## 3 Integrating Desktop Grids with Production Grid Systems

Production grid systems, such as the Teragrid [6], provide significant data, application and hardware resources for the execution of science and engineering applications. While grid resources include software stacks for activities such as authentication, job submission and resource monitoring, directly learning and using these may be a daunting task for a large class of users. Additionally, production grid systems are being increasingly targeted towards an audience beyond the traditional scientific and HPC communities, with users from arts and humanities making use of grid resources. Thus, creating user interface for production grids that is intuitive for desktop users is becoming increasingly important. In this section, we describe an effort to make production grid resources accessible through iShare.

The previous section described how iShare enables a broad class of resources to be shared and accessed through a desktop interface. In interfacing with production grid systems, iShare needs to address some additional issues –

- *Single Authentication Mechanism :* While desktop grid resources may each specify their own authentication mechanisms, production grid systems usually have a single authentication mechanism for accessing resources (often through X.509 certificates). The user interface should shield the user from the complexities of different authentication mechanisms for desktop grids and production grids.

- *Batch Processing :* Many production grids are batch processing systems – a paradigm that most current desktop users are not accustomed to. Additionally, each site within a single production grid usually has its own job queues and schedulers.Therefore, the iShare system needs to shield the user from the complexities of batch processing associated with production grids.

In this section, we describe how these issues were addressed in implementing an interface to the Teragrid using iShare.

The base iShare system has modules for resource publishing, parsing resource semantics and remote execution using SSH. We implemented plug-ins to these modules to allow Teragrid resources to be accessed from iShare. Additionally, to allow easy access to the data resources on the Teragrid, we created a general FTP client type user interface and implemented plug-ins that allow data to be accessed from GridFTP [13] servers and Storage Resource Brokers [17]. A schematic representation of these plug-ins and their functions is depicted in Figure 3.2.

### 3.1 RDF Extensions to Support Production Grid Resources

Apart from the usual parameters, specification of grid resources typically require additional details such as queue names and accounting information. We extended the iShare user interfaces to allow these parameters to be specified, wherever appropriate, during resource publication and remote execution. We also implemented two sets of plug-ins – the first accommodates these additional details specified during resource publication in RDF and the second extends the RDF parser module to make appropriate use of these parameters for job submission on Teragrid resources.

### 3.2 Remote Execution

iShare has a remote execution module that uses SSH to run remote jobs. We implemented a set of plug-ins that extend this module for job submission to grid systems. For this implementation, we used the Java CoG kit [29]. This provides a pure Java implementation of a subset of Globus tools and libraries. Since iShare is also implemented in Java, the Java CoG kit provides a suitable set of libraries for developing the Teragrid plug-ins.

For remote execution on the Teragrid, plug-ins need to handle (1) GSI based authentication using Teragrid X.509 certificates and (2) Job submission using the Globus Resource Allocation Manager (GRAM) protocol. We implemented certificate management and authentication using the certificate management API provided by the Java CoG kit. For job submission, instead of creating a low level GRAM client with the *org.globus.gram* package, we used the *Task* abstraction [2] provided by the CoG kit. The use of abstraction enhances the stability of this iShare plug-in by decoupling it from the actual protocol implementation, thus insulating it from any protocol changes with future versions of GRAM and Globus. The use of the task abstraction also provides for a more simple, maintainable and reusable implementation of remote job submission on Grids.

### 3.3 Extensions for Data Transfer

Data resources constitute an important class of Teragrid resources. iShare enhances the end-user's computing experience by allowing seamless access to these data resources as well as non-grid data resources. Using this interface, users may browse and transfer data between locations that have different access protocols (Secure FTP, GridFTP, SRB, local and network file systems). To implement GridFTP based transfers, we used the *FileResource* and *Task* abstractions of the Java CoG kit. As in the case of remote execution, the use of these abstractions enabled a cleaner, stable and reusable implementation. For implementing transfers from and browsing of data collections on Storage Resource Brokers (SRB), we used the Jargon [16] API.

We extended the iShare RDF to accommodate descriptions of these diverse data resources. A salient feature of this RDF extension is that it allows any number of format or item-specific attributes to be stored in the data descriptor. This provides the flexibility of using domain specific attributes in the future for enhanced functionality, such as transferring only those parts of a file matching the user's requirements.

To evaluate the overhead of providing the iShare user with the desktop interface to grid resources, we measured the performance of GridFTP transfers initiated using three different methods – (i) from the Unix command line on a host where the full *Globus* toolkit is installed using the *globus-url-copy* command, (ii) from the iShare FTP user interface and (iii) from a Gridsphere based portal. Figure 4 shows the relative performance of file transfers from each of these user interfaces. As a baseline, the graph also shows the secure transfer time – the fastest available method for data transfers. For submission through a portal, in case of medium to large file sizes, the portal queues the transfer as a separate batch task. Therefore, for submission through a portal, we have measured both the "submission time" (the time taken from when the user clicks a *File Copy* button to when the portal responds stating that the job has been submitted) and the "completion time" (the time taken by the submitted transfer to complete). From the graph, we find that iShare overheads and the command-line submission overheads are similar, with iShare being about 30% faster for small transfers and 25% slower for large transfers. iShare is almost two orders of magnitude faster than the Gridsphere based portal for file transfers.

### 3.4 Comparison with Grid Portals

With the emergence of production grid systems, user interfaces for accessing grid resources have become important. Portals or gateways are one of the user interfaces that have gained popularity in terms of user interfaces to produc-

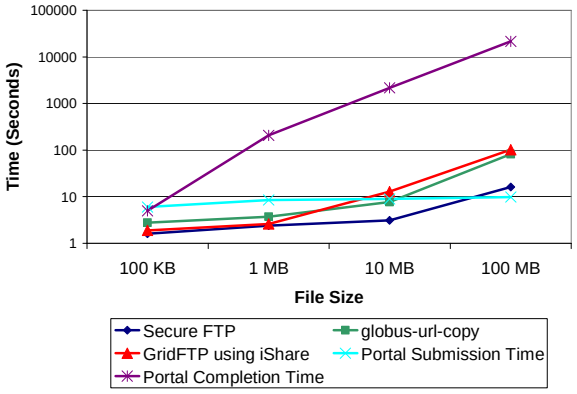


**Figure 4. Performance Comparison of GridFTP initiated from (i) the command line using *globus-url-copy*, (ii) from iShare and (iii) from a Gridsphere portal.**

tion grids. A portal is a web-based application that usually aggregates content to provide a single entry-point to a set of tools or application. When these tools and applications are grid-based, the portal is called a grid-portal. A distinction is often made between the terms *portal* and *gateway*. A *gateway* is a portal that provides services to a specific user community. The NanoHUB (http://www.nanohub.org), for example, is a science gateway. For the rest of this section, however, we shall use the terms *portal* and *gateway* interchangeably. Drawing from our previous experience with the PUNCH [20] portal and our current experiences in creating a portal for climate simulations on the Teragrid, we compare and contrast certain aspects of iShare and grid portals.

**Architecture :** Portals, by design, have a centralized architecture. They are not necessarily monolithic – different parts of a portal such as a database, a credential repository and the actual portal container may run on different physical machines. However, the machines usually need to be within a single administrative domain. The portal administrator decides the full set of applications and services that the portal will provide. Users can access a subset of these services. If a new application is to be added to the portal, it requires the direct participation of the portal administrator and portal developers.

On the other hand, iShare has a completely distributed architecture. There is no central entity on which the iShare system depends – a collection of clients form peer-to-peer rings to store meta-data for the resources. Each client in itself is capable of accessing any iShare resource. A salient feature of the distributed architecture of iShare, discussed in Section 2, is the ease with which resources can be added to iShare. Another direct consequence of this distributed design is improved fault tolerance. Portal services become unavailable whenever the physical resources hosting the portal container or any of its components fail, whereas the iShare system has no such central point of failure.

Some recent work has been building portals with the goal of providing a collaborative environment for grid application development and cross-grid resource usage. One example is the P-GRADE Portal [27]. It allows the collaboration between *workflow* developers, as well as the access to grid resources across multiple virtual organizations by using the standard Globus middleware components. iShare accomplishes the goal of collaborative development as well by providing an intuitive and standards based approach (through RDF) for describing resources. The RDF-based approach and the integration of Grid protocols using the Java CoG kit also gives iShare the flexibility and interoperability to interact with grid and non-grid resources.

**Authentication :** Grid portals usually provide accesses to resources within a single grid system. Consequently, the authentication scheme used by the portal itself is usually tied to the authentication mechanism of the allied grid system. Portals may provide a login/password interface to users, though the allied grid system may require, for example, X.509 certificates from a trusted certifying authority. In such a case, portals usually do one of three things - (a) The portal temporarily maps a user to one out of a pool of portal certificates and uses this portal certificate to submit the user's job to the grid system. The Purdue NanoHUB, for example, uses this scheme to submit jobs to the Teragrid. (b) The portal generates a certificate for each user after a certifying authority has endorsed the user using some other mechanism. (c) The user has valid credentials for accessing the grid resources and the portal retrieves these resources from some credential repository, such as MyProxy. In cases (b) and (c), systems such as PURSE and GAMA [14, 4] may be used to hide the complexities of credential generation and retrieval from the user.

iShare allows users to access only those production grid resources for which they have valid credentials. The credential management module in iShare uses Java CoG kit functionality to generate proxy certificates if the user's credentials are stored on the user's desktop and retrieves credentials from a MyProxy repository otherwise. In both cases, user's are still shielded from the complexities of proxy initialization and retrieval – they see a username/password type of interface, both for local proxy initialization as well and for proxy retrieval from a credential repository.

**Job Submission :** As described earlier in this section, iShare uses Java CoG kit abstractions to implement job submission and file transfers to Globus based grid systems. In this respect, its implementation technology is similar to current portal systems, such as the Teragrid User Portal (https://portal.Teragrid.org/gridsphere/). Many current portal systems also use the Java CoG API within their servlets and portlets for job submission and allied functionality (ex-

amples include GridPortlets [15] and OGCE portlets [12]).

**User Interface :** To compare the user interface provided by iShare and grid portals, we shall discuss both the visual user interface and the interface provided for accessing services provided by each. In terms of the visual interface, grid portals usually allow the user a level of customization both in terms of look and feel and in terms of the services that are visible. Portals, especially those composed of portlets often strive for a more "desktop" feel, allowing users to individually access specific sub-windows within a single browser window. iShare, being a desktop client, implicitly offers a visual interface intuitive for desktop users. In terms of content organization, portal administrators usually categorize applications and tools into groups and usually offer users the flexibility of subscribing to these groups selectively. In iShare, resources are hierarchically organized, by semantics, into a hierarchical name space. In portals, user privileges may determine the visibility of services, based on policies set by the portal administrator. In iShare, all published resources are visible to all users - access privileges of a specific user are checked only when the user tries to access a specific resource. An important difference between the visual interfaces in iShare and grid portals is in how and when the user interface for a resource is implemented. iShare provides tools for the resource publisher to create an user interface for the resource being published, which becomes part of the resource description and is rendered by iShare whenever the resource is accessed. For portals, the interface is created by the portal developer using the technology being used to serve content (usually portlets and servlets). In terms of extensibility, iShare has certain advantages. First, iShare clients are desktop java applications and thus do not have the constraints of web based interfaces (applets and servlets). Second, the resource description and access interface in iShare can be updated easily. In case of portals, this usually entails reprogramming of some portal component.

In terms of access methods, portals often allow their services to be accessed as *Web Services*. Users can now compose basic web services into arbitrarily sophisticated workflows. iShare, on the other hand, provides a more traditional notion of composition. Output from an iShare application can be piped into another iShare application, allowing for a type of composition that desktop users, accustomed to using pipes and redirection, may find more intuitive.

## 4 Related Work

On-going research on middlewares for Internet-sharing systems can be divided into four categories. The focus of the first category is to provide application programmers a set of tools to harness grid resources., Examples of such work include Globus [10] and GridLab [1]. Work in the second category aims to develop "grid-enabled" domain-specific applications. Active projects include EuroGrid [11] and CrossGrid [5]. Work in the third category is geared towards providing end users and resource providers with the means to disseminate, access, and utilize networked resources. Related work in this category includes systems to support high throughput computing on large collections of distributed computing resources (such as Condor [28]), web service techniques using Application Servers (http://www.w3.org/TR/ws-arch), and active web portals such as the NanoHUB (http://www.nanohub.org). The forth category targets desktop resources for supporting large-scale computation and storage. Active research includes P2P-based systems in file sharing (http://www.kazaa.com) and CPU cycle sharing (http://setiathome.ssl.berkeley.edu).

The work described in this paper best fits the third category (end-user-oriented systems) and the forth category (desktop grids). In the context of end-user-oriented systems, we have already compared iShare with grid portals. Condor, the other type of system in this category, and iShare are similar in the fact that both systems make use of desktop resources as well as dedicated grid resources. However, openness is not the design thesis of Condor; it deploys system-specific standards in authentication, job submission, and job control. This is in contrast to iShare's open approach in embracing heterogeneous resources. Another difference is iShare's fully decentralized structure leveraging the scalability and self-management of a structured P2P overlay, versus the constitution of central units (such as matchmakers and managers) in Condor. Furthermore, iShare's approach of proactive resource allocation achieves better application performance on desktop resources.

Existing desktop grid systems federate voluntary facilities of computation. Some of them, e.g., Entropia [7] and BOINC (http://boinc.berkeley.edu), are limited in the structures with participants providing idle CPU cycles to tasks farmed out by a centralize server. These systems assume that all applications are trusted and apply naive scheduling methods, such as FCFS, for allocating resources. Other systems such as XtremWeb [9] and WaveGrid [30] manage providers and consumers as peers in a more decentralized way. They proposed mechanisms to verify the validity of user applications (XtremWeb planned to use *software fault isolation* and WaveGrid proposed a *quiz* mechanism), but did not measure the introduced overhead of their methods. Similar to iShare, WaveGrid exploited the knowledge of resource availability to achieve fast turnaround. However, the knowledge is based on the simple intuition that more idle cycles are available at night; it does not factor the fluctuating availability caused by arbitrary resource behavior.

## 5. Conclusion

In this paper, we have presented iShare, an open system for federating networked resources: both voluntary desktop machines and dedicated resources from production grids. The iShare system incorporates innovative techniques for managing desktop resources. By posting resource descriptors on the Web and inserting resource meta-data into a peer-to-peer network overlay, iShare achieves decentralization in resource dissemination and flexibility in adding new resources. The resource availability is predicted and factored into resource allocation in iShare. This proactive approach obtains desirable application performance under fluctuating resource availability. Last, iShare confines the execution of untrusted applications within virtual machines. We also discussed the methods that enable iShare to extend its intuitive Internet-sharing paradigm to production grid systems, and compared with another paradigm using web-based user portals.

We measured iShare's end user performance in terms of the average job completion time of our proactive resource allocation, the overhead introduced by confining untrusted applications, and the efficiency of data transfer enabled by integrating production grids. The evaluation results demonstrate that iShare provides efficient, reliable and secure management of Internet-based resources.

## References


[1] G. Allen, K. Davis, K. N. Dolkas, N. D. Doulamis, et al. Enabling applications on the Grid - a GridLab overview. *International Journal of High Performance Computing Applications*, 17(4), 2003.

[2] K. Amin, M. Hategan, G. von Laszewski, and N. J. Zaluzec. Abstracting the Grid. In *Proc. of PDP' 04*, pages 250–257, 2004.

[3] D. H. Bailey, E. Barszcz, J. T. Barton, et al. The nas parallel benchmarks-summary and preliminary results. In *Proc. of SC'91*, pages 158–165, 1991.

[4] K. Bhatia, K. Mueller, and S. Chandra. Gama: Grid account management architecture. In *Proc. of IEEE International Conference on EScience and Grid Computing*, page 10 pages, Dec. 2005.

[5] M. Bubak, M. Malawski, and K. Zajac. The CrossGrid architecture: Applications, tools, and grid services. In *Proc. of AxGrids*, 2003.

[6] C. Catlett. The philosophy of TeraGrid: Building an open, extensible, distributed terascale facility. In *Proc. of CCGrid*, 2002.

[7] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 63(5):597–610, 2003.

[8] J. Dike. A user-mode port of the Linux kernel. In *the 4th Annual Linux Showcase and Conference*, pages 63–72, 2000.

[9] G. Fedak, C. Germain, V. N'eri, and F. Cappello. Xtremweb: A generic global computing system. In *Proc. of CCGrid'01*, 2001.

[10] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2), 1997.

[11] C. Hoppe, P. Gmbh, D. Mallmann, and F. Julich. Eurogrid - european testbed for grid applications. *GRIDSTART Technical Bulletin*, 2002.

[12] http://www.collab ogce.org/ogce2/. OGCE : Open Grid Computing Environments, 2004.

[13] http://www.globus.org/toolkit/docs/4.0/data/gridftp/. FTP for Grid Systems, 2001.

[14] http://www.grids center.org/solutions/purse/. Purse: Portal-based user registration service, 2005.

[15] http://www.gridsphere.org/. Grid Portlets User's Guide, 2005.

[16] http://www.sdsc.edu/srb/index.php/Jargon. A java client api for the datagrid, 2005.

[17] http://www.sdsc.edu/srb/index.php/Main_Page. The sdsc storage resource broker, 2004.

[18] http://www.w3.org/RDF. Resource Description Framework, 2004.

[19] X. Jiang, D. Xu, and R. Eigenmann. Protection mechanisms for application service hosting platforms. In *Proc. of CCGrid'04*, pages 656–663, 2004.

[20] N. H. Kapadia and J. A. B. Fortes. PUNCH: An architecture for Web-enabled wide-area network-computing. In *Proc. of Cluster Computing*, 1999.

[21] M. W. Mutka. Estimating capacity for sharing in a privately owned workstation environment. *IEEE Trans. On Software Engineering*, 18(4):319–328, 1992.

[22] Z. Pan, X. Ren, R. Eigenmann, and D. Xu. Executing MPI programs on virtual machines in an internet sharing system. In *Proc. of IPDPS'06*, page 10 pages, 2006.

[23] X. Ren and R. Eigenmann. Empirical studies on the behavior of resource availability in fine-grained cycle sharing systems. In *Proc. of ICPP'06*, pages 3–11, 2006.

[24] X. Ren, S. Lee, R. Eigenmann, and S. Bagchi. Resource availability prediction in fine-grained cycle sharing systems. In *Proc. of HPDC'06*, pages 93–104, 2006.

[25] X. Ren, Z. Pan, R. Eigenmann, and Y. C. Hu. Decentralized and hierarchical discovery of software applications in the ishare internet sharing system. In *Proc. of PDCS'04*, 2004.

[26] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001.

[27] G. Sipos and P. Kacsuk. Multi-grid, multi-user workflows in the p-grade portal. *Journal of Grid Computing*, 3(3-4):221–238, 2006.

[28] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The condor experience. *Concurrency - Practice and Experience*, 17(2-4), 2004.

[29] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001.

[30] D. Zhou and V. Lo. Wave scheduler: Scheduling for faster turnaround time in peer-based desktop grid systems. In *Proc. of JSSPP'05*, 2005.