iShare - Open Internet Sharing Built on Peer-to-Peer and Web

Xiaojuan Ren and Rudolf Eigenmann

Purdue University, School of ECE,
West Lafayette, IN, 47907, USA
{xren, eigenman}@purdue.edu

Abstract. This paper presents design concepts and implementation overview of an Internet-sharing system, iShare. iShare supports end users as well as providers of Internet resources in disseminating, accessing and using these resources, in a way that allows open participation. A fully decentralized organization allows providers to simply post their resources on any web page, imposing no restrictions on resources attributes, administrative rules, and access protocols. Underneath its user surface it employs peer-to-peer information dissemination, advanced resource matching, open migration, and automatic service portal mechanisms. In addition to the qualitative comparison with related work, we evaluate the system in terms its efficiency of resource discovery and job execution.

1 Introduction

Internet-sharing systems harness the rapidly growing, worldwide resources of computer, network, and information systems. Advanced sharing technology bears tremendous potential in creating synergy among the users – both providers and end users – of machine platforms, networks, computer applications, software services, and information. We envision that future end users will be able to quickly learn about the availability of world-wide resources and to employ them as if they were located nearby, without download, installation, or maintenance effort. Providers of resources will be able to easily offer new software, hardware, or data to the community. In doing so, they will be able to define their own rules and create business or open source models, akin to today's economic principles.

In this paper we present design concepts and implementation overview of an Internet-sharing system, iShare[1], which serves as a research platform in pursuit of this vision. iShare facilitates sharing of three types of resources: programs (software tools, computational applications), machines (computers, devices), and data (documents, data bases). While much technology exists today for the sharing of data, the focus of iShare's initial thrust is on technology for sharing executable programs (a.k.a. *services* – we will use the two terms interchangeably) and their underlying compute platforms.

[1] It is based upon work supported in part by the U. S. National Science Foundation under Grants No. 9974976-EIA, 0103582-EIA, and 0429535-CCF.

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 1117–1127, 2005.

iShare extends the concepts of the PUNCH [7, 6] network computing system, which became operational in 1995 at Purdue University and has since served a large user community (over 3000 users in 35 countries) in computational nanotechnology, computer architecture and parallel programming. Both iShare and PUNCH have a strong end-user orientation. They 1) provide the means to disseminate, access and use networked resources and 2) they populate the infrastructure with services for the community. This orientation distinguishes the systems from related work in areas such as Grid computing and Web services. While these areas are pursuing goals similar to the original PUNCH project, their current focus is on developing the underlying technology, programming support, and standards for exploiting computational resources. iShare was motivated by an observed limitation of PUNCH, which it shares with many related efforts: populating the infrastructure with additional resources is limited by eligibility requirements and administrative procedures. For example, machines may only be added if they adopt certain administrative procedures; and programs may only be added after they become "grid-enabled". Furthermore, the new resources may need to be registered in or approved by a central organization before they become visible to the community. Removing these barriers would enable Internet resources to be shared in a truly open and scalable manner.

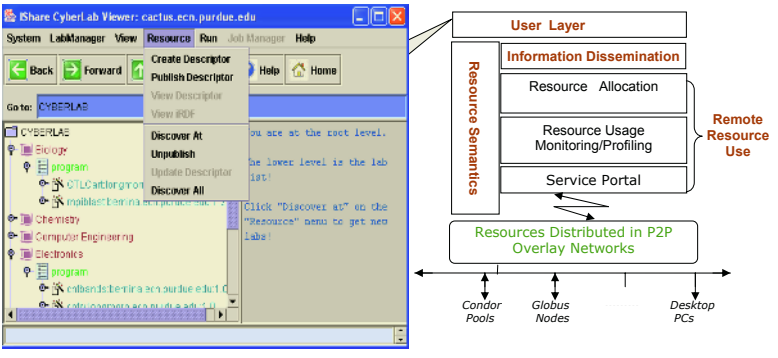
iShare addresses these issues by taking an *open publication* approach, which imposes no restrictions on participating resources. A web-posting mechanism publishes new resources and a fully distributed peer-to-peer mechanism supports resource discovery. Powerful resource matching mechanisms serve to map program resources onto fitting platforms. While iShare builds on existing standards and middleware technology, *protocol plug-ins* allow new protocols and standards to be added, satisfying the needs of resources posted in the future.

The remainder of this paper is organized as follows. Section 2 presents the design concepts and implementation of iShare. Section 3 provides evaluation results. We discuss related work in Section 4, followed by conclusions in Section 5.

2 The iShare Architecture

An Internet-sharing system can be characterized in terms of 1) its user model, 2) the semantics and type of the resources it can support, 3) the information disseminated about resources and activities at remote sites, and 4) the mechanisms that support the use of these resources at remote sites. This section elaborates on iShare's design concepts and implementations in these four areas.

The system architecture is shown in Figure 1(b). For space reasons, several important iShare components were left out of this paper – most notably iShare's authentication and trust mechanism (building on state-of-the-art techniques) as well as facilities for exchanging experience and bridging across diverse user communities. An extended version of this paper describes these components [5].



(a) The End User View in iShare (b) The iShare Architecture

Fig. 1. The End User View and the Structure of iShare

2.1 User Model

Supporting a *Resource Provider* User Class. Internet-sharing systems provide their user communities with facilities to discover and make use of resources created and maintained at remote sites. In addition to its *end users*, iShare considers *providers* of resources as an important user class. Resource developers take advantage of the ease with which resources can be made available to end users.

The basic iShare functionality for resource providers supports 1) creating resource descriptors, 2) publishing and removing resources, and 3) optionally starting and configuring advanced iShare services. Publication of a new resource is fully autonomous and does not require user interaction with any iShare "administrator". To end users, the published resources appear organized into discipline-specific *Cyberlabs*. Providers define (as a resource attribute) the Cyberlabs in which they wish to "place" the new resource. The core functionality for end users is to run the discovered, remote programs, using local files and displays as input/output.

Providing iShare Functionality Within Common Work Environments. Web portals provide convenient user interfaces to many Internet-sharing systems. However, advanced users often prefer to work within their common environment, such as a Unix Shell or Windows desktop. The iShare user functionality is accessible via an API, which allows different user interfaces to be built. Our initial design includes a Windows-based interface, as shown in Figure 1(a) and we are exploring iShare Unix Shell functionality.

2.2 Resource Semantics and Types

Resource Description with RDF. An important issue in Internet-sharing systems is the clear description of resources. Resource providers must define the semantics about what is shared, who is allowed to share, and how sharing occurs.

iShare adopts RDF as the resource description language for consistent encoding of resource attributes. RDF aims to specify semantics for data in a standardized interoperable manner [1] and provides a rich data model for describing objects.

iShare supports three types of resources at an equal level: software services (programs), service platforms (machines), and data. An important program attribute is that of a *pinned* or *roaming* service. Pinned services are applications running only on a specific platform, whereas roaming applications can execute on any matching platform. Machines are published as available hosts for roaming services. They might be individual machines or locally managed sub-systems (e.g., a network of workstations managed by Condor [9]).

Autonomy in Defining Resource Attributes. An important design concept is that resources can define their own rules of usage. Thus, a machine may offer services under its own administrative procedures and access protocols. iShare will find matching programs and platforms. Protocols that are not available initially may be plugged in, making the system incrementally more powerful.

To support the autonomy in defining local access rules, the standardized metadata definition must be extensible enough to support plug-in protocols. In our initial design, we formalized the metadata to include the common attributes of access protocols in widely used local management systems such as Condor [9] and PBS [10]. The metadata includes recompiling/relinking operations, submission syntax and user-commands. For example, accessing and utilizing a Condor pool involves relinking programs to utilize advanced features like checkpointing, creating a "submission files" with predefined syntax, and starting job running via specific commands: condor_submit. All this information must be described as resource access protocols. The information is then used for creating service portals, which hide the details from end users.

2.3 Information Dissemination

Decentralized Organization Built on P2P and Web. The information to support network-accessible computing is incremental and distributed in the sense that it is created from items at different locations and items available at different points in time. The incremental and distributed nature of the information makes it infeasible to collect and maintain it at a centralized location in a scalable manner. Thus, a decentralized scheme is essential for reliable and efficient information management. A key idea of iShare's decentralized structure is that a provider of resources can post their availability on any web page. Resource metadata is derived from these postings and inserted into a P2P network. While the World Wide Web affords unprecedented access to resource descriptors, metadata distributed in the P2P network improves discovery of and access to resources.

To publish a new resource, the publisher specifies the publication URL via the user layer configuration. Resource descriptors are posted on this URL manually or automatically through iShare's resource publishing tool. The involved web servers serve as a repository to resource description documents and are managed

by the individual resource publishers. The P2P network in iShare consists of all participant nodes. An iShare node could be a host serving a published software service, a host published as an accessible machine resource, or the workstation from which an end user accesses iShare.

Information Naming, Publication and Discovery. Resources in iShare are hierarchically categorized into *Cyberlabs*, as shown in Figure 2. The resources in one lab are semantically related in their functionalities. Resources are described by metadata, which form a tree representing the hierarchical name space. Instead of maintaining central directory service for the tree, iShare distributes the hierarchical name space to the underlying P2P network, which supports the publication and discovery process.

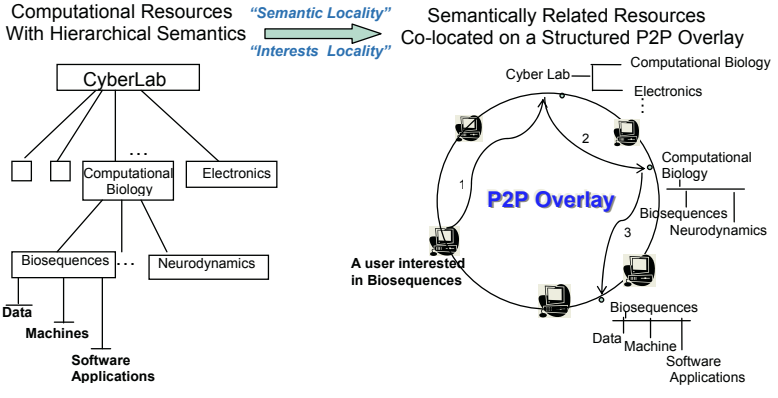


Fig. 2. Resources are organized in a hierarchical tree, which is mapped to a P2P overlay. The big circle represents a P2P overlay, with arrows indicating P2P routing messages to discover resources

Information disseminated in iShare includes resource descriptions, job execution profiles, resource-usage information, and user profiles. Each piece of information is linked to a specific resource. Thus it could be also mapped to the hierarchical structure described above. An item in the hierarchical space is mapped to a peer node by the hashing value of the item's prefix path. The current implementation of the P2P network is built on a structured overlay network, Pastry [3]. A shared data item is distilled into standardized metadata and inserted with Pastry's (*route (msg, key)*) API. Requests for data are routed without requiring any knowledge of where the corresponding data items are stored.

To achieve fault tolerance, each publication operation creates a few replicas. The discussion of the replication and fault tolerance is beyond the scope of this paper, and details could be found in [4]. A local resource cache is designed to keep recently-used metadata. Successful end-user discovery operations will update the cache. Mechanisms to maintain cache consistency are described in [4]. The

impact of caching on resource discovery latency is evaluated in Section 3. Load balancing related to the size of data stored on each peer is currently exploited and will be presented in a future paper.

2.4 Remote Resource Use

Mechanisms for the remote use of resources are at the core of most Internet-sharing systems. They include the functionality for matching discovered software services with execution platforms, employing suitable protocols for remote job execution, and connecting user input/output. iShare builds on a large number of contributions in this area. Two features distinguish iShare's remote resource use: support for roaming services and automatic creation of service portals.

Resource Allocation for Roaming Services. Roaming services are programs that pick the best combination of service replica and matching platform for every invocation. Our concepts include advanced resource matching and *open migration* of the service to the matching platform. Advanced matching needs to consider the option that services whose program source code was included in the publication may be recompiled and assigned to a different platform. Decision making for such matching in heterogeneous environments involves replication and caching strategies for service binaries and performance prediction techniques that consider possible recompilation. Open migration installs a service on a potentially unreliable platform. This involves monitoring, safeguarding mechanisms, and possibly relocating the service.

Automatic Creation of Service Portals. A service portal establishes the user interface to a remote program. It is mapped to the native job running environment at run-time. The challenges are to create this interface automatically from the service description and generate it in the user's preferred software environment. Service portals may be batch-oriented or fully interactive. They support plug-in access protocols by automatically generating job submission files and keeping track of site- and application-specific information.

3 Evaluation

Of the four areas of contributions, as described in Section 2, this section provides quantitative evaluations of the information dissemination (in terms of the latency of resource discovery) and the remote resource use techniques (in terms of the efficiency of job execution). For qualitative evaluations of the user model, and resource semantics components we refer to [5].

3.1 Efficiency of Resource Discovery

We simulated the P2P based resource discovery, *iDiscover*, on a GT-ITM router network using the transit-stub model [2]. The size of the IP network is 1050

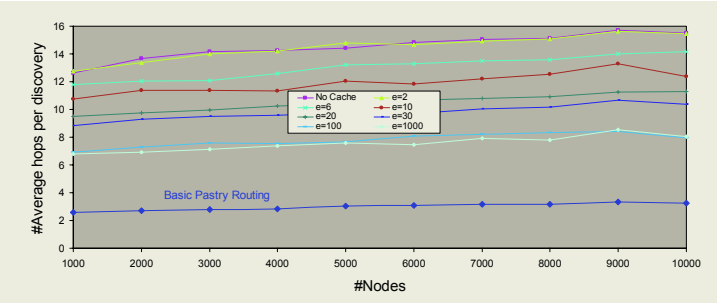


Fig. 3. Average number of hops per discovery. e is the normalized cache expiration time (cache expiration time/average request period)

routers, 50 of which are used in transit domains and the remaining 1000 in stub domains. To test the scalability of *iDiscover*, we simulated several iShare testbeds with the number of nodes ranging from 500 to 10,000. We assume the iShare nodes are uniformly attached to the routers.

In the experiment, we measured the effect of caching on discovery latency with different normalized cache expiration time e (e = cache expiration time/average request period). Each discovery operation starts from searching the root "Cyberlab". One out of five nodes in the P2P network was randomly chosen to initiate the search request. Figure 3 plots the discovery response time with e ranging from 2 to 1,000. The figure shows that the discovery latency increases very slowly with the total number of nodes. We also see that with fair expiration time ($e = 6$), the response time is reduced by 10.37% on average compared to a discovery without local cache. The measurement results for cache hit rates are analyzed in [4]. Compared with the basic Pastry message routing, the resource discovery takes only a factor of 2-4 times longer, while supporting searches of resources with specific functionalities.

3.2 Efficiency of Remote Job Execution

In this section, we compare the efficiency of remote job execution in iShare with that in SSH-based remote access. The goal is to show that iShare provides advanced network computing environment with acceptable costs in performance.

Remote access to computing services is commonly provided via explicit login to remote platforms (via REXEC, RSH and SSH). These mechanisms have several limitations [7], such as users having to manually identify and select remote machines as well as being exposed to site- and platform-specific idiosyncrasies. iShare addresses these limitations by decoupling the computing environment perceived by users from the underlying physical environment. After getting the required access (e.g., account/password, certificate or public/private keys), end users can start an arbitrary job by interacting with the service portal. iShare translates and maps the user inputs automatically to the native service interface.

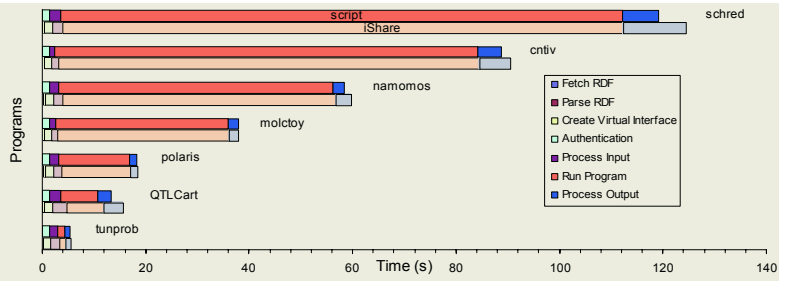


Fig. 4. Remote job execution through iShare and standard SSH

In iShare, the job execution includes the steps to fetch resource descriptor (from Web or local cache), parse the descriptor, create the service portal, authenticate, transfer input files, execute the program (including input processing and environment configuration) and process output. We chose a set of pinned services on a machine accessed via SSH. After getting accesses to these services, we ran them on iShare's built-in SSH client and monitored the time spent at each of the above steps. To test the efficiency of the standard SSH, we manually coded Shell scripts to run each of these programs on the same remote machine with standard ssh and scp commands. A ssh agent is started manually to provide the authentication similar to iShare's single sign-on mechanism (end users only need to input a password once). The efficiency difference between the public key authentication and password authentication is ignored in this experiment.

Figure 4 shows the results for running a set of programs in iShare and with the manually-coded scripts. The time for parsing RDF descriptors and creating service portals is less than one second. The SSH-based job submission in iShare performs similar to the standard SSH protocol. The standard SCP protocol outperforms the iShare's built-in file transfer protocols. The reason is that file I/O operations involved in SCP cause more overhead in iShare, because it is implemented with high level programming language (Java). Optimization of data transfer will be considered in future work.

4 Related Work

The research community is exploring a variety of approaches for constructing software infrastructures for Internet-sharing. On-going research can be divided into five categories with different foci and goals. The first category includes global standardization efforts for grid computing, represented by GGF [19] and NMI [18]. The focus of the second category is to provide application programmers a set of tools to harness "Grid" resources, e.g., to distribute massively parallel applications with message-passing. Examples of such work include Globus [8] and GridLab [11]. Work in the third category aims to develop "Grid-enabled" domain-specific applications. Active projects include EuroGrid [12],

CrossGrid [13]. Work in the fourth category is geared towards providing end users (and resource providers) with the means to disseminate, access and use networked resources. Related work in this category includes active software web portals such as PUNCH [7] and NCSA-portals [14]; and web service techniques such as IBM WebSphere Application Servers [15]. The work in the fifth category is motivated by using P2P techniques to manage and share globally distributed resources. Active research includes large-scale file sharing systems [16] and compute cycle sharing [17].

iShare belongs to the forth category, which differs from the other four categories by its user-orientation, its network accessibility to executable programs, its focus on "single-platform" rather than parallel applications and its support for generic instead of domain-specific programs. Within this category, we introduce the related work in end-user-oriented Internet-sharing systems and compare iShare in terms of the design concepts in Section 2.

User Model: Software web portals provide end users direct access to unmodified software tools via standard Web browser. However, they don't provide any open functionalities to resource providers. Most often, adding a new tool on a web portal has to involve the portal developers' administration. iShare solves this problem by decentralized web-posting and P2P message routing. Web service techniques enable users to build Web-based applications using preferred object model, programming language and platform. Service developers publish service descriptions to a central information location. This differs from iShare's support for unmodified applications with no programming requirement and decentralized resource publication.

Resource Semantics: Both web portals and web services target software application resources that are bound to fixed machine resources. iShare's roaming services combine program and machine resources provided by different communities, thus offering a more flexible and open environment for resource sharing. Resource descriptions in web portals exist as static web pages containing documentations for specific programs. These web pages are manually maintained by portal administrators. Web services use WSDL for service interface specification and the specification is registered to a central UDDI registry. The RDF language adopted in iShare focused on semantics specification rather than syntax specification in WSDL. iShare's resource description supports autonomy on the definition of access protocols, while in web portals and web services, standard access protocols are required.

Information Dissemination: Resource discovery is not a critical issue in web portals, because all application resources are listed explicitly on web pages linked through the portal's main web page. In web services, users locate the services from the UDDI registry via SOAP. In contrast to the centralized structures in the two systems, iShare disseminates information via posting on individual web pages and registering to a P2P network. There is no central registry server or storage space in the whole iShare system.

Remote Resource Use: The remote job execution in web portals and web services is supported by standardized protocols. They can be secured with HTTP basic authentication, HTTPS and SSL encryption, and digital signature. iShare supports plug-in protocols defined by individual providers via learning from the resource descriptions. The initial design of iShare supports commonly used authentication protocols such as SSH. Future work will extend iShare's security implementation to also support the Grid Security Infrastructure.

5 Conclusions

We have presented the design concepts and a implementation prototype of iShare - an Internet-sharing system built on P2P technology and the Web. iShare differs from related work in its user functionality for both providers and end users, its autonomy in defining and controlling local resources, its P2P-based information dissemination mechanisms and its support for roaming services and dynamic service portal creation. The evaluation results on resource discovery and remote execution confirm that iShare is able to deliver scalable and efficient Internet-based computing.

References

1. K. Selcuk Candan, H. Liu, R. Suvarna: Resource Description Framework: Metadata and Its Applications. ACM SIGKDD Exploration Newsletter, **3** (2001) 6–19
2. Ellen Zegura, Kenneth Calvert, Samrat Bhattacharjee: How to Model an Internetwork. Proc. IEEE INFOCOM, (1996)
3. A. Rowstron, P. Druschel: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), (2001) 329–350
4. Xiaojuan Ren, Zhelong Pan, Rudolf Eigenmann, Y. Charlie Hu: Decentralized and Hierarchical Discovery of Software Applications in the iShare Internet Sharing System. Proc. PDCS, (2004)
5. Xiaojuan Ren, Rudolf Eigenmann: iShare – Internet Sharing of Programs, Machine and Data. Technical Report ECE-HPCLab-04203, High-Performance Computing Laboratory, Department of ECE, Purdue University, (2004)
6. Insung Park, Nirav H. Kapadia, Renato J. Figueiredo, Rudolf Eigenmann, et. al.: Towards an Integrated, Web-executable Parallel Programming Tool Environment. Proc. Supercomputing Conference, (2000)
7. Nirav H. Kapadia, Jose A. B. Fortes: PUNCH: An Architecture for Web-enabled Wide-area Network-computing. Cluster Computing, **2** (1999) 153–164
8. I. Foster, C. Kessleman: Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, **11(2)** (1997)
9. Litzkow M. J, Livny M, Mutka M. W: Condor - A Hunter of Idle Workstations. Proc. ICDCS, (1988) 104–111
10. R. Henderson, D. Tweten: Portable Batch System: External Reference Specification. Technical Report, NASA Ames Research Center, (1996)

11. Gabrielle Allen, Kelly Davis, K. N. Dolkas, N. D. Doulamis, et. al.: Enabling Applications on the Grid - A GridLab Overview. International Journal of High Performance Computing Applications, (2003)
12. Christian Hoppe, Pallas GmbH D. Mallmann, F. Julich: EUROGRID - European Testbed for GRID Applications. GRIDSTART Technical Bulletin, (2002)
13. Marian Bubak, Maciej Malawski, Katarzyna Zajac: The CrossGrid Architecture: Applications, Tools, and Grid Services. AxGrids, (2003)
14. http://www.ncsa.uiuc.edu/AboutUs/FocusAreas/ScientificPortalsExpedition.html
15. Supporting Open Standards for Web Services and the Java Platform. ftp://ftp.software.ibm.com/software/webserver/appserv/v5/G325-1971-00.pdf
16. KazaA. http://www.kazaa.com/us/index.htm
17. D. Anderson, et. al.: Internet Computing for SETI. ASP Conference Series, 2000.
18. NFS MiddleWare Initiative. http://www.nsf-middleware.org/
19. Global Grid Forum. http://www.ggf.org/