Future Generation Computer Systems 25 (2009) 147–152

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Integrated resource management for lambda-grids: The Distributed Virtual Computer (DVC)

Andrew A. Chien[a,*], Nut Taesombut[b]

[a] Intel Research, Santa Clara, CA, United States

[b] Department of Computer Science and Engineering, University of California, San Diego, United States

ARTICLE INFO

Article history:
Received 31 January 2008
Received in revised form
28 June 2008
Accepted 12 July 2008
Available online 29 July 2008

Keywords:
Optical networks
Management
Grids
Middleware
Network abstraction
Network resource management
Internet protocol
Adminstrative domain
High performance computing
High speed networking

ABSTRACT

The Distributed Virtual Computer (DVC) is the key unifying element of the OptIPuter software architecture. It provides a simple, clean abstraction for applications or higher-level middleware, allowing them to use lambda-grids with the same convenience as a VPN. The DVC is successful because it employs integrated network and end-resource selection, achieving high quality results so that there is little incentive for end-users to expose and manage lower-level interfaces. We describe the development of the DVC abstraction, key results, and experience with multiple applications and testbeds.

1. Introduction

When the OptIPuter project [1] began in September 2002, it was a heady time. We set out to build "lambda grids" which would tap the communication capabilities of the extraordinary quantities of fiber that had been laid in speculation during the late 1990's and thru the "dot com" boom. While a few "stunts" had been done in the university community, the challenge of opening these elements of the physical layer to software applications – which had never before had visibility below the IP layer in the wide area – was a radical and daunting challenge. As prior application stunts with optical networks required heroic manual intervention [2], the prospect of user-controlled, highly-optimized access to "lambdas", much less optimized "networks of lambdas and end resources" – a lambda-grid – seemed quite remote.

At the same time, a significant community had sprung up around "grid computing" [3], the exciting notion – now widely adopted – that large-scale resource sharing could be done across the wide area, given appropriate abstractions, middleware, and of course compute, data, and network resource infrastructures. These grid computing systems, focused on efficient sharing and exploitation of massive supercomputing systems and data servers, paying little attention to the networks between them [4,5].

Other high visibility projects in the community [6] had concluded that optical networks were too difficult for applications or end-users to access directly, but rather that sophisticated "network engineers" were needed to set up sharable network structures — that would constrain the use of "lambda's" in simple, structured patterns, that the applications would conform themselves to. Essentially, this approach posited that dynamic resource management, and efficient automatic management and selection could not be achieved.

To bring together the innovative protocols and applications demanding terabit performance with IP service, and low-level optical network management protocols involving muxes and demuxes, oxc switch control protocols, and lambdas, we developed the OptIPuter software architecture (see Fig. 1). The Distributed Virtual Computer (DVC) abstraction [7] framed the challenge to make computing across lambda-grids as easy for applications as computing on a VPN — a staple of local- or wide-area networking. However, the challenge was much greater — the DVC abstraction is more like a VPN with control over sharing and guaranteed

* Corresponding author.
E-mail addresses: andrew.chien@intel.com (A.A. Chien), nut@cs.ucsd.edu (N. Taesombut).

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.07.007

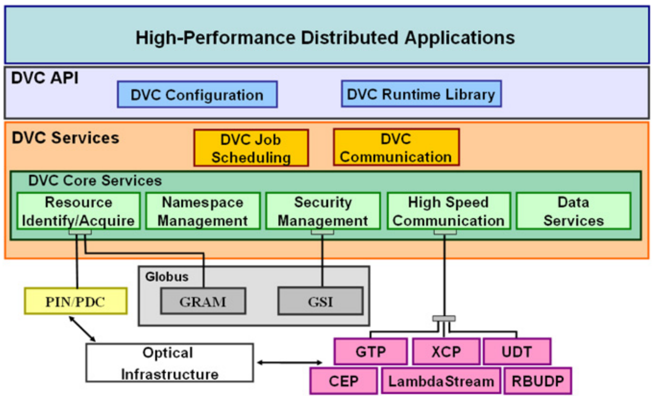

Fig. 1. OptIPuter system software architecture.

extremely high performance. In this paper, we summarize the major contributions of the DVC project, and the history of its development. Specific contributions include:

1. Describing the DVC architecture and showing how it solves the central architectural problem, providing a high-level application view to optical networks and resource management systems,
2. Showing DVC's layer atop other grid services for efficient implementation, and provide a high-level abstraction with minimal cost across a wide variety of network information models, and
3. Describing how the DVC abstraction was applied to a wide variety of science applications and other middleware.

In addition to the technical achievements, one of the most gratifying aspects of the project was the acceptance and use of the DVC middleware (the invention of our research) within and beyond the OptIPuter project. Its utility was clearly appreciated, and we believe it should be adopted in a wide variety of areas. We learned a great deal about how to get a major change in software architecture established, and of course the major benefits and software modularity it can bring. Applications ported to DVC were fully portable — all aspects of the underlying special networking structures were cleanly captured. As well, full performance could be delivered thru the DVC selection and configuration mechanisms.

The remainder of this paper is organized as follows. Section 2 describes the Distributed Virtual Computer (DVC) architecture and key components. Section 3 presents integrated network- and end-resource selection algorithms, evaluation methodology and key results. Section 4 describes our experience with real scientific applications and network testbeds. Section 5 summarizes the paper.

2. Distributed Virtual Computer architecture

The DVC provides an integrated resource abstraction enabling applications to exploit lambda-grids (lambdas, computers, data servers, etc.) with simple use and controllable performance models. Unlike traditional grid and optical network middleware [4,6,8], it allows an application to describe a combined set of desired communication and end resources, then automatically optimizes their configuration and manages them for reliable and high performance. The DVC abstraction not only presents a simple, unified interface to underlying networks and resource management systems, but also provides integrated network/end-resource management enabling both high application capabilities and resource efficiencies.

The DVC architecture is shown in Fig. 2. Key elements include:

- **DVC-ISL** — an "integrated specification language" [9] that describes application resource requirements, including traditional end resource specification and explicit high-level description of communication resources.
- **DVC-RCP** — a "resource configuration planner" which takes a DVC-ISL specification as input, and returns a *resource configuration* (DVC-RC) matching the specification.
- **DVC-RB** — a "resource binder" which takes a DVC-RC as input and negotiates with grid resource managers and optical network services for co-allocation of the end resources and private networks [9,16].

Here, an application (or user) can conveniently describe, acquire and use a private set of distributed end resources and optical networks. Specifically, the application creates a DVC-ISL specification describing its resource needs, and passes it to the DVC-RCP. In response, the DVC-RCP retrieves information about available resources from grid and network information services, and matches the given specification with a DVC-RC, a combined set of selected end resources and optical network. The DVC-RC is then presented to the application. If not satisfied with the result, the application modifies its specification and repeats the process. Then, the application passes the DVC-RC to the resource binder (DVC-RB) that instantiates it by allocating the corresponding end resources and optical networks. All these resources are then bound into a *DVC environment* [7]. The DVC middleware uses them to implement a simple environment for applications — comparable to a private, local distributed system. Cushioned by the DVC middleware, applications can be written simply — avoiding the full complexity of a grid environment. Further, the application can use DVC API's to configure the DVC environment and modify its configurations according to evolving application requirements and resource conditions.

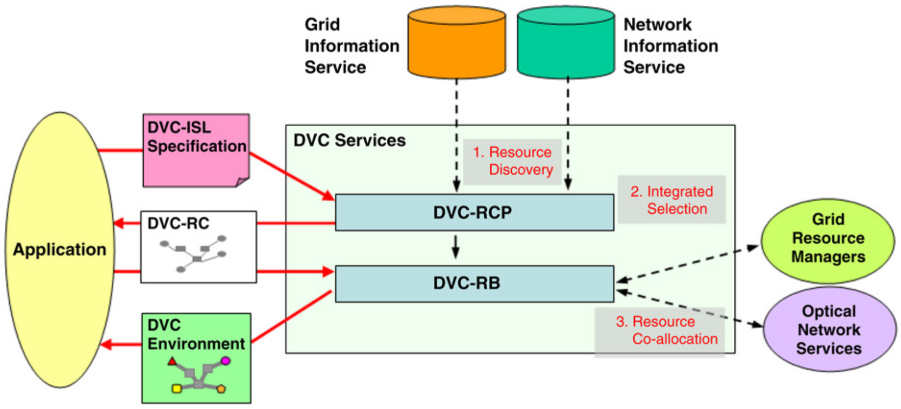

Fig. 2. DVC Integrated resource management architecture.