Carnegie Mellon University
Research Showcase @ CMU

Computer Science Department School of Computer Science

1989

The design of Nectar : a network backplane for heterogeneous multicomputers

Emmanuel A. Arnould
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/compsci

This Technical Report is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Computer Science Department by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

Find authenticated court documents without watermarks at docketalarm.com.

The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers

Emmanuel A. Arnould François J. Bitz Eric C. Cooper
H. T. Kung Robert D. Sansom Peter A. Steenkiste

January 1989

CMU-CS-89-101

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Also published in *The Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, April 1989

ABSTRACT

Nectar is a "network backplane" for use in heterogeneous multicomputers. The initial system consists of a star-shaped fiber-optic network with an aggregate bandwidth of 1.6 gigabits/second and a switching latency of 700 nanoseconds. The system can be scaled up by connecting hundreds of these networks together.

The Nectar architecture provides a flexible way to handle heterogeneity and task-level parallelism. A wide variety of machines can be connected as Nectar nodes and the Nectar system software allows applications to communicate at a high level. Protocol processing is off-loaded to powerful communication processors so that nodes do not have to support a suite of network protocols.

We have designed and built a prototype Nectar system that has been operational since November 1988. This paper presents the motivation and goals for Nectar and describes its hardware and software. The presentation emphasizes how the goals influenced the design decisions and led to the novel aspects of Nectar.

This research was supported in part by Defense Advanced Research Projects Agency (DOD) monitored by the Space and Naval Warfare Systems Command under Contract N00039-87-C-0251, and in part by the Office of Naval Research under Contracts N00014-87-K-0385 and N00014-87-K-0533.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

1 Introduction

Parallel processing is widely accepted as the most promising way to reach the next level of computer system performance. Currently, most parallel machines provide efficient support only for homogeneous, fine-grained parallel applications. There are three problems with such machines.

First, there is a limit to how far the performance of these machines can be scaled up. When that limit is reached, it becomes desirable to exploit coarse-grained, or task-level, parallelism by connecting together several such machines as nodes in a multicomputer. Second, there is a limit to the usefulness of homogeneous systems; as we will argue below, heterogeneity (at hardware and software levels) is inherent in a whole class of important applications. Third, parallel machines are typically built from custom-made processor boards, although they sometimes use standard microprocessor components. These machines cannot readily take advantage of rapid advances in commercially-available sequential processors.

Current local area networks (LANs) can be used to connect together existing machines, but this approach is unsatisfactory for a heterogeneous multicomputer with both general-purpose and specialized, high-performance machines. It is often not possible to implement efficiently the required communication protocols on special-purpose machines, and typical applications for such systems require higher bandwidth and lower latency than current LANs can provide.

The Nectar (*network computer architecture*) project attacks the problem of heterogeneous, coarse-grained parallelism on several fronts, from the underlying hardware, through the communication protocols and node operating system support, to the application interface to communication. The solution embodied in the Nectar architecture is a two-level structure, with fine-grained parallelism within tasks at individual nodes, and coarse-grained parallelism among tasks on different nodes. This system-level approach is influenced by our experience with previous projects such as the Warp[1] systolic array machine [1] and the Mach multiprocessor operating system [12].

The Nectar architecture provides a general and systematic way to handle heterogeneity and task-level parallelism. A variety of existing systems can be plugged into a flexible, extensible network backplane. The Nectar system software allows applications to communicate at a high level, without requiring each node to support a suite of network protocols; instead, protocol processing is off-loaded to powerful network interface processors.

We have designed and built a prototype Nectar system that has been operational since November 1988. This paper discusses the motivation and goals of the Nectar project, the hardware and software architectures, and our design decisions for the prototype system. We will evaluate the system with real applications in the coming year.

Section 2 of this paper summarizes the goals for Nectar. An overview of the Nectar system and the prototype implementation is given in Section 3. The two major functional units of the system, the HUB and CAB, are described in Sections 4 and 5. Section 6 describes the Nectar software. Some of the applications that Nectar will support are discussed in Section 7. The paper concludes with Section 8.

2 Nectar Goals

There are three major technical goals for the Nectar system: *heterogeneity*, *scalability*, and *low-latency, high-bandwidth communication*. These goals follow directly from the desire to support an emerging class of large-grained parallel programs whose characteristics are described below.

[1]Warp is a service mark of Carnegie Mellon University.

University Libraries

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

2.1 Heterogeneity

One of the characteristics of these applications is the need to process information at multiple, qualitatively different levels. For example, a computer vision system may require image processing on its raw input at the lowest level, and scene recognition using a knowledge base at the highest level. A speech understanding system has a similar structure, with low-level signal processing and high-level natural language parsing. The processing required by an autonomous robot might range from handling sensor inputs to high-level planning.

At the lowest levels, these applications deal with simple data structures and highly regular number-crunching algorithms. The large amount of data at high rates often requires specialized hardware. At the highest level, these applications may use complicated symbolic data structures and data-dependent flow of control. Specialized inference engines or database machines might be appropriate for these tasks. The very nature of these applications dictates a heterogeneous hardware environment, with varied instruction sets, data representations, and performance.

Software heterogeneity is equally significant. The most natural programming language for each task ranges from Fortran and C to query languages and production systems. As a result, the system must handle differences in programming languages, operating systems, and data representations.

2.2 Scalability

Often the most cost-effective way of extending a system to support new applications is to add hardware rather than replacing the entire system. Also, by including a variety of processors, the system can take advantage of performance improvements in commercially available computers. In the Nectar system, it must therefore be possible to add or replace nodes without disruption: the bandwidth and latency between existing tasks should not be affected significantly, and it should not be necessary to change existing system software. Using the same hardware design, Nectar should scale up to a network of hundreds of supercomputer-class machines.

2.3 Low-Latency, High-Bandwidth Communication

The structure of these parallel applications requires communication among different tasks, both "horizontally" (among tasks operating at the same level of representation) and "vertically" (between levels). The lower levels in particular require a high data rate (megabyte images at video rates, for example). Moreover, applications often have response-time requirements that can only be satisfied by low-latency communication; two examples are continuous speech recognition and the control of autonomous vehicles.

In general, by providing low-latency, high-bandwidth communication the system can rapidly distribute computations to multiple processors. This allows the efficient parallel implementation of many applications.

Lowering latency is much more challenging than increasing bandwidth, since the latter can always be achieved by using pipelined architectures with wide data paths and high-bandwidth communication media such as fiber-optic lines. Latency can be particularly difficult to minimize in a large system where multi-hop communication is necessary. Nectar has the following performance goals for communication latency: excluding the transmission delays of the optical fibers, the latency for a message sent between processes on two CABs should be under 30 microseconds; the corresponding latency for processes residing in nodes should be under 100 microseconds; and the latency to establish a connection through a single HUB should be under 1 microsecond.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.