Rack-aware operating systems

Moritz Hoffmann
Systems Group, Department of Computer Science, ETH Zürich
Advisor Timothy Roscoe

Abstract

Applications running on a rack can only scale out to a certain limit. The main limit is congestion of the interconnect and unbalanced resource allocation. To solve this I want to turn the inter-machine interconnect into a managed resource. If the interconnect was a managed resource, one could assign appropriate resources based on demand to applications allowing them to run more efficient and more balanced inside a rack.

1 Introduction

Rack-scale computing has recently gained a lot of attention because racks are considered the new basic unit of computing in warehouse-scale data centers [7] running appliances such as SAP HANA [6], Oracle Exadata [10] or IBM PureData for Analytics. Racks consist of 100s to 1,000s of servers connected by a low-latency switched interconnect. Applications can benefit from dedicated resources and fast networks. However, applications currently using the network can only scale up to the limits of network links and existing network protocols. Major sources of this problem are the lack of management of the interconnect as well as missing support of the software stack for transfer and routing protocols [5].

In every computer, scarce resources need to be managed to allow efficient operation. These resources include the CPU bandwidth, physical and virtual memory, devices and storage. Applications running on a single machine can assume these resources to be managed by one of today's operating systems.

But there's another scarce resource which is vitally important in rack-scale computing: the networked interconnect between the computers in the rack. Traditionally, the interconnect has not been managed at all, but has been treated as sort of a black box.

Distributed applications heavily rely on machine-local resources but also on the functionality of the interconnect and suffer a lot if it does not perform well. For example, a load balancer needs both information about a single machine's utilization as well as available network bandwidth in order to guarantee low response times.

At this time, there exists no resource allocation system that has a rack-wide view including the interconnect. Having applications managing the resources by themselves leads to problems. I believe that system-wide resource management could lead to better results. Current operating systems only manage local resources and are not aware of the interconnect as a resource, so they are unable to maintain a rack-local view of resources.

2 Towards a solution

Previous work [2, 3, 4, 8, 9, 10] has shown alternative approaches to structure an operating system as a distributed system both within a single machine and spanning multiple machines.

My approach is to extend a single-machine distributed operating system to a rack. By doing so I want to extend the existing resource management capabilities to the interconnect.

A second step would be to revise how the operating system manages traditional resources in the context of a rack. When crossing the machine boundary, resource management concepts need to be considered from a different angle. Task scheduling changes significantly and opens many new possibilities. Devices might have a spatial locality the operating system needs to take into account when managing applications. We are currently working on handling memory across multiple address spaces.

3 Approach

I want to use the *Barrelfish* [1] operating system as a base for researching rack-aware operating systems. It is

Technology	Latency	Bandwidth
InfiniBand FDR	700 ns	56.0 Gbit/s
DDR4-2133	c. 24 ns	204.8 Gbit/s
10G Ethernet	5,000...50,000 ns	10.0 Gbit/s

Figure 1: Bandwith and latency for selected technologies

a single-machine distributed operating system and provides a multi-kernel architecture with a micro-kernel approach. Most of Barrelfish's functionality is encapsulated in independent domains while the kernel provides communication, basic memory management and capability control. Currently, the network is not a managed resource. In order to make Barrelfish rack-aware I will start with building a managed interconnect.

I am working on supporting InfiniBand as a low-latency interconnect between different machines. Table 1 shows a comparison of latency and bandwidth for different interconnects. InfiniBand seems like a reasonable compromise offering bandwidth and latency between Ethernet and local memory access. It supports RDMA, which can be used to create high-performance message passing channels similar to those used by Barrelfish within a single machine. My work on integrating InfiniBand aims at providing a managed network interconnect for Barrelfish. This is a first step towards a rack-aware operating system and will enable future research such as cross-machine resource management, scheduling and simplified distributed computing.

References

[1] BAUMANN, A., BARHAM, P., DAGAND, P.-E., HARRIS, T., ISAACS, R., PETER, S., ROSCOE, T., SCHÜPBACH, A., AND SINGHANIA, A. The multikernel: A new os architecture for scalable multicore systems. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles* (New York, NY, USA, 2009), SOSP '09, ACM, pp. 29–44.

[2] BAUMANN, A., PETER, S., SCHÜPBACH, A., SINGHANIA, A., ROSCOE, T., BARHAM, P., AND ISAACS, R. Your computer is already a distributed system. Why isn't your os? *12th Workshop on Hot Topics in Operating Systems* (2009).

[3] BOYD-WICKIZER, S., CHEN, H., CHEN, R., MAO, Y., KAASHOEK, F., MORRIS, R., PESTEREV, A., STEIN, L., WU, M., DAI, Y., ZHANG, Y., AND ZHANG, Z. Corey: An operating system for many cores. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2008), OSDI'08, USENIX Association, pp. 43–57.

[4] CONNORS, S. H., SEARS, B., SULLIVAN, T., AND WILKES, J. Brevix design 1.01.

[5] COSTA, P., BALLANI, H., AND NARAYANAN, D. Rethinking the network stack for rack-scale computers. In *Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing* (2014), USENIX Association, pp. 12–12.

[6] FÄRBER, F., CHA, S. K., PRIMSCH, J., BORNHÖVD, C., SIGG, S., AND LEHNER, W. SAP HANA database: Data management for modern business applications. *SIGMOD Rec. 40*, 4 (Jan. 2012), 45–51.

[7] NITZBERG, B., AND LO, V. Distributed shared memory: A survey of issues and algorithms. *Distributed Shared Memory-Concepts and Systems* (1991), 42–50.

[8] SCHWARZKOPF, M., GROSVENOR, M. P., AND HAND, S. New wine in old skins: the case for distributed operating systems in the data center. In *Proceedings of the 4th Asia-Pacific Workshop on Systems* (2013), ACM, p. 9.

[9] VINTER, S. T., AND SCHANTZ, R. E. The cronus distributed operating system. In *Proceedings of the 2Nd Workshop on Making Distributed Systems Work* (New York, NY, USA, 1986), EW 2, ACM, pp. 1–3.

[10] WENTZLAFF, D., AND AGARWAL, A. Factored operating systems (fos): The case for a scalable operating system for multicores. *SIGOPS Oper. Syst. Rev. 43*, 2 (Apr. 2009), 76–85.