Managing Network Resources in Condor

Jim Basney and Miron Livny
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706-1685, USA
{jbasney,miron}@cs.wisc.edu

Abstract

Data-intensive applications in the Condor High Throughput Computing environment can place heavy demands on network resources for checkpointing and remote data access. We have developed mechanisms to monitor, control, and schedule network usage in Condor. By managing network resources, these mechanisms provide administrative control over Condor's network usage and improve the execution efficiency of Condor applications.

1 Introduction

Until recently, the Condor research project has focused on the challenges of managing usage of CPU resources for High Throughput Computing (HTC) [4]. However, as the amount of physical memory available to HTC applications has dramatically increased, HTC environments have become an attractive platform for applications which are more data-intensive. As these applications place greater demands on the network, it has become important for Condor to manage usage of network resources in addition to CPU resources to enforce administrative network policies and to ensure that applications receive sufficient network capacity to compute efficiently [1]. As middle-ware, positioned between the operating system and application, Condor can perform *application-aware*, *inter-application* network management. In this paper, we describe mechanisms we have implemented in Condor for network monitoring, network and CPU co-allocation, and checkpoint scheduling.

2 Monitoring Network Usage

We have modified the Condor scheduler to monitor network usage for checkpoint transfers, remote data access, and file staging. Condor applications send a checkpoint over the network to a dedicated checkpoint server when preempted and continue their execution by transferring their checkpoint from the checkpoint server to a new execution site. Applications also perform checkpoints periodically to limit the amount of work lost in case of system failure. Since a checkpoint contains the entire memory state of the application, checkpoints can grow to be large for data-intensive applications. The modified Condor scheduler computes the network capacity required for job placement from the size and location of the application's checkpoint and data files, as specified in the job placement request sent by the *customer agent*. As the application executes, an *application resource manager* (ARM) periodically reports the application's network usage for remote data access to the scheduler. The ARM also notifies the scheduler whenever the application performs a checkpoint. When a job completes its execution, the customer agent may continue using the allocated CPU by running another job in its place. In this case, the customer agent notifies the scheduler of the network capacity required for the placement of the new job.

By monitoring Condor's network usage, we provide statistics to the Condor administrator about how Condor applications are using the network. This information is also used by the scheduler to perform network admission control and scheduling, as we will show in the following sections. More general-purpose network monitoring tools, such as the Network Weather Service [6] and Gloperf [3], could be used to complement this information if we want to make scheduling decisions based on external network conditions.

3 Network and CPU Co-Allocation

In addition to modifying the scheduler to collect network usage statistics, we have modified the scheduling algorithm to implement network and CPU co-allocation, to allocate bundles of network and CPU capacity to Condor applications. There has been much recent work on general frameworks for co-allocation of heterogeneous resources, including the Globus Architecture for Reservation and Allocation [2] and the Condor Gang-Matching framework [5]. We focus here on the specific case of network and CPU co-allocation in the Condor environment.

The modified Condor scheduler performs network admission control to ensure that network resources are not oversubscribed. The scheduler allocates network capacity up to a configured limit for each subnet. The administrator typically configures this limit to less than the subnet's full capacity to reserve capacity for other network users. The scheduler places jobs on remote CPUs only when the network capacity limits would not be exceeded by the jobs' checkpoint and data transfers. If capacity would be exceeded for a given subnet, jobs which may have run on CPUs on that subnet will instead run in other subnets (where network capacity is available) or remain idle.

Co-allocating network and CPU resources in the Condor scheduler also allows customers to control their requests for bundles of network and CPU resources. Customers may specify constraints and preferences regarding the network capacity available from the allocated CPU. For example, the customer may request a CPU with at least 8 Mbps peak network capacity with a preference for a smaller number of network hops to the home filesystem. This additional expressiveness in the request enables the customer to make more efficient use of network resources.

4 Checkpoint Scheduling

The co-allocation mechanisms described in the previous section do not differentiate between network usage for checkpointing, remote data access, and file staging. However, each different use of the network provides unique scheduling opportunities. We focus on checkpoint scheduling, since checkpoint transfers are responsible for the majority of the demand for network capacity in our local Condor pool. In many cases, preemption checkpoints can be pre-scheduled to complete before an eviction deadline. Additionally, periodic checkpoints may be scheduled to reduce contention with other network transfers and to improve checkpointing performance. To take advantage of these opportunities, we have developed two checkpoint schedulers: a *preemption checkpoint scheduler* and a *periodic checkpoint scheduler*.

When a large number of applications are preempted simultaneously, the checkpoints compete with each other for network bandwidth, causing all of the checkpoints to take a long time to complete. Simultaneous application preemptions are in many cases a result of events which may be anticipated. For example, applications running on computers in a classroom will be preempted at the start of class. Condor's network monitoring facilities can help the administrator identify recurring preemption events. The Condor preemption checkpoint scheduler monitors running applications and reserves bandwidth for checkpoints in advance of these scheduled preemption events. When the start time for the bandwidth reservation arrives, the scheduler directs the application to begin its checkpoint.

The periodic checkpoint scheduler monitors network usage in Condor and initiates requested periodic checkpoints when Condor network usage is low. Periodic checkpoints are not initiated when there are active placement or preemption transfers on the network, to avoid slowing those time-critical transfers. Additionally, the periodic checkpoint transfers are serialized, so they don't compete with each other for network resources and therefore complete more quickly, allowing the application to quickly resume its execution.

5 Conclusion

As HTC applications place greater demands on the network, it is important for the HTC environment to manage usage of network resources. We have modified the Condor scheduler to collect statistics on application network usage and perform network and CPU co-allocation. We have also developed two checkpoint schedulers, which reduce network contention among checkpoint transfers to improve checkpointing performance. Together, these mechanisms manage network resources to enable HTC for network-intensive applications.

References

[1] J. Basney and M. Livny. Improving goodput by co-scheduling cpu and network capacity. *International Journal of High Performance Computing Applications*, 13(3), Fall 1999.

[2] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke. A distributed resource management architecture that supports advance reservations and co-allocation. In *International Workshop on Quality of Service*, 1999.

[3] C. Lee, R. Wolski, I. Foster, C. Kesselman, and J. Stepanek. A network performance tool for grid computations. In *Proceedings of the Conference on Supercomputing*, 1999.

[4] M. Livny and R. Raman. High-throughput resource management. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, chapter 13. Morgan Kaufmann Publishers, Inc., 1998.

[5] R. Raman, M. Livny, and M. Solomon. Gang-matching: Advanced resource management through multilateral matchmaking. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, August 2000.

[6] R. Wolski. Dynamically forecasting network performance to support dynamic scheduling using the network weather service. In *Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing*, August 1997.