Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems

Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath

Department of Computer Science
Rutgers University
Piscataway, NJ 08854-8019

{edpin,ricardob,vinicio,taliver}@cs.rutgers.edu

Technical Report DCS–TR–440, May 2001

Abstract

In this paper we address power conservation for clusters of workstations or PCs, such as those that support a large number of research/teaching organizations and most Internet companies. Our approach is to develop systems that dynamically turn cluster nodes on – to be able to handle the load imposed on the system efficiently – and off – to save power under lighter load. The key component of our systems is an algorithm that makes load balancing and unbalancing decisions by considering both the total load imposed on the cluster and the power and performance implications of turning nodes off. The algorithm is implemented in three different ways: (1) at the application level for a cluster-based, locality-conscious network server; (2) at the operating system level for an operating system for clustered cycle servers; and (3) by application/operating system negotiation, again using the cluster-oriented operating system and a negotiation API. Our experimental results are very favorable, showing that our systems conserve both power and energy in comparison to traditional systems, which do not adapt their configurations.

1 Introduction

Power and energy consumption have always been critical concerns for laptop and hand-held devices, as these devices generally run on batteries and are not connected to the electrical power grid. The amount of power consumed by such a device determines the required battery capacity and the temperature of the device components, whereas the energy consumed ultimately determines the battery life. As a result, a tremendous amount of research has been directed towards low-power and low-energy design and conservation (e.g. [16, 33, 22, 11, 13]).

In contrast with this line of research, in this paper we propose a *new research direction*, namely power and energy conservation for clusters of workstations or PCs, such as those that support a large number of research and teaching organizations and most Internet companies. Power consumption is an important concern in the context of clusters as it directly influences the cooling requirements of each cluster node and of the overall cluster ensemble. Our own cluster of high-performance workstations is an example of these per-node requirements. Each of our single-processor, dual-disk nodes uses 7 fans. Each node consumes approximately 78 Watts during normal operation under our group's cluster workloads. Interestingly, a non-trivial fraction of this power consumption, approximately 9 Watts (or 12%), is consumed by the fans themselves. Higher performance nodes consume an even larger fraction of their total power consumption just in per-node cooling hardware.

The cooling requirements of cluster ensembles can also be substantial. In fact, a medium to large number of high-performance nodes racked closely together in the same room, as is usually the case with clusters, requires a significant investment in cooling, both in terms of sophisticated racks and heavy-duty air conditioning systems. At each Google site, for instance, there are 40 racks, each of which with 80 PCs, for a total power consumption of nothing less than 180 kWatts! Cooling such a large installation is an expensive challenge. Even worse, *the investment in cooling systems is becoming an ever larger fraction of the cost of clusters, as off-the-shelf hardware prices constantly decrease.* Given a fixed budget, one dollar spent on air conditioning is one dollar not spent in hardware that can actually produce better service or higher throughput, such as cluster nodes, disks, etc.

Taking a broader perspective, the power requirements of clusters have become a major issue for several states, such as California, New York, and Massachusetts. The New York Times reported that the planned 46 data centers for the New York city metropolitan area alone have asked for a minimum draw of 500 megawatts of power, which is enough to power 500,000 households [29]! The inability of these states to meet such enormous power requirements has already resulted in data center projects being cancelled or delayed [29].

Besides power consumption, energy consumption is also important for large, high-performance clusters. Both the computational and the air conditioning infrastructures consume energy. This energy consumption is reflected in the electricity bill, which can be significant for a large cluster in a heavily air-conditioned room. In fact, the electricity bills for large cluster systems are likely to soar in certain states (especially California) in the near future, if charges per Wh indeed become a function of overall consumption. Research and teaching organizations supported by large clusters, in particular, may have

a hard time finding the funds for covering energy costs. Again, the news services provide anecdotal confirmation of how critical energy consumption is for large computer systems. An article from ComputerWorld [9] discusses last year's surge in energy consumption in the heart of Silicon Valley and includes a quote from a spokesman for Silicon Valley Power to the effect that a single large data center can consume more energy than the largest manufacturing plant his company serves. A Google site, for instance, consumes no less than 130 MWh per month just with the cluster infrastructure.

Our approach to conserving power/energy is to develop systems that can leverage the widespread replication of resources in clusters. This paper presents our first steps in this new research direction. In particular, we develop systems that can dynamically turn cluster nodes on – to be able to handle the load imposed on the system efficiently – and off – to save power under lighter load. This research is inspired by previous work in cluster-wide load balancing (e.g. [2, 14, 24, 26, 8, 27, 4]). When performing load balancing, the goal is to evenly spread the work over the available cluster resources in such a way that idle nodes can be used and performance can be promoted. The inverse of the load balancing operation concentrates work in fewer nodes, idling other nodes that can be turned off. This *load concentration* or *unbalancing operation* saves the power consumed by the powered-down nodes, but can degrade the performance of the remaining nodes and potentially increase their power consumption. Thus, load concentration involves an interesting *performance* vs. *power tradeoff*. Our systems use load concentration as a first-class technique, rather than as a remedial technique like in systems that harvest idle workstations (e.g. [2, 24]) or as a management technique for manually excluding a cluster node.

The key component of our systems is an algorithm that makes load balancing and concentration decisions by considering both the total load imposed on the cluster and the power and performance of different cluster configurations. In more detail, the algorithm periodically considers whether nodes should be added to or removed from the cluster, based on the expected performance and power consumption that would result, and decides how the existing load should be re-distributed in case of a configuration change. To be able to understand the implications of our algorithm, we implemented it for two popular types of cluster-based systems: a locality-conscious network server and a load balancing distributed operating system (OS) for clustered cycle servers. The implementations were performed in three ways: (1) at the application level for the network server; (2) at the OS level for the cycle server; and (3) by application/OS negotiation for the cycle server, extending the load balancing OS with a simple negotiation API.

Even though we target power conservation primarily, our experimental results show that our secondary goal of saving energy is achieved as well. Our initial results show that our modified network server can reduce the total power consumption of the cluster by as much as 86% and the energy consumption by 43% in comparison to the original server running on a static cluster configuration with 8 nodes. The modified OS can reduce power consumption by as much as 86% for a synthetic workload, while attempting to keep performance degradation below 20%, again in comparison to the original system on a static 8-node cluster. The energy savings it accrues in this case is 32%. Finally, our application/OS interaction experiments on 4 cluster nodes show that we can save as much as 25% power and 29% energy, while respecting performance degradation directions provided by the applications.

As mentioned above, the previous work on power and energy is concentrated on laptop computers and embedded or hand-held devices. As far as we are aware, our work is the first to study power and energy conservation for clusters in detail. A very recent paper [7] also suggests this research direction, but does not include an evaluation of their proposed energy-conscious switch. In addition, most of the previous work (e.g. [11, 22, 7]) is focused on reducing consumption by moving resources to standby or idle states, which are states that are not as power-hungry as the active state, but still consume power. In this paper we take the extreme approach of turning whole nodes completely off. Our algorithm and systems apply novel strategies for distributing load in clusters, in particular they apply load concentration as a first-class technique. In terms of application/OS interaction, two previous papers [25, 13] have proposed such interaction for energy conservation. We propose yet another type of interaction.

Based on our experimental results, we believe that our algorithm and systems should be useful for organizations and companies that rely on large clusters of workstations or PCs, in that they can revert the dollars to be spent in air conditioning and electricity to dollars to be spent on additional computational hardware.

The remainder of this paper is organized as follows. Section 2 describes our cluster configuration and load distribution algorithm and its different implementations. Section 3 describes our experimental set-up and the methodology used. Section 4 discusses our experimental results. Section 5 discusses the related work. Finally, section 6 concludes the paper and mentions our future work.

2 The Cluster Configuration and Load Distribution Algorithm

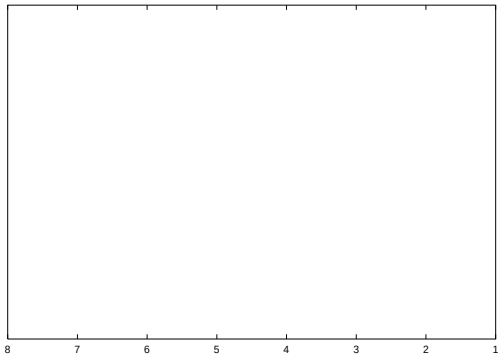
2.1 Overview

Power vs. performance. We consider the tradeoff between power and two types of performance, namely throughput and execution time performance. Throughput is the key issue for systems such as modern network servers, in which the goal is to service as many requests as possible; the latency of each request at the server is usually a small fraction of the overall latency of wide-area client-server communication. Execution time is key for systems such as cycle servers, as users may object to significant delays in the execution of their jobs.

The cluster configuration and load distribution algorithm we propose is based on the overall trends shown in figure 1. The figure plots the behavior of power consumption and performance (in this particular case, throughput performance) for an imaginary fixed workload, as a function of the number of cluster nodes. The power consumed by a cluster decreases as we decrease the number of nodes, whereas throughput suffers beyond the point where resources achieve maximum utilization.

Our algorithm conceptually moves around the graph in figure 1, evaluating the "best" move at each time. More specifically, for each cluster configuration and currently offered load, the algorithm decides whether to add (turn on) or remove (turn off) nodes, according to the expected performance and power implications of the decision.

For simplicity, the algorithm assumes that the cluster is comprised of homogeneous machines. Furthermore, the algorithm assumes that the removal of a node does not cripple the file system. In practice this assumption is often verified, since the responsibility for serving file


8
7
6
5
4
3
2
1