Large-scale cluster management at Google with Borg

Alexandra Miller, Calvin Hoang, Chadwick Smith, and Jordan Perez Herrera

What problem is the paper solving?

The paper addresses the challenge of efficiently managing and scheduling large-scale computing workloads across massive clusters of machines. At Google's scale, thousands of applications that range from long-running, user-facing services to short-lived batch jobs, must share resources across tens of thousands of machines. The core problem is how to allocate computing resources i.e CPU, memory, and storage, schedule tasks, and handle failures in a way that maintains high performance and reliability while maximizing overall system utilization. This includes dealing with heterogeneous workloads, preventing resource contention, and ensuring that critical services continue running even when parts of the system fail.

Why is this problem important?

This problem is important because modern internet services depend on large-scale data centers, and inefficient management of these resources can lead to significant performance issues and financial costs. Even small improvements in resource utilization can save millions of dollars due to the scale of infrastructure involved. Additionally, systems like Google's must provide high availability and low latency for user-facing applications such as search and Gmail, where failures or slowdowns directly impact users. Efficient cluster management also allows different types of workloads to share resources effectively, reducing waste while maintaining reliability. Without solving this problem, large-scale computing systems would be less efficient, more expensive, and less capable of supporting the demands of modern applications.

What are the authors proposing?

To address the problem of managing large-scale infrastructure, the authors propose Borg, a cluster management system that abstracts thousands of machines into a single resource pool called a cell to maximize hardware efficiency and utilization. This allows engineers to interact with a datacenter as if it were one large computer. The Borg architecture relies on a logically centralized controller, called the Borgmaster, which handles resource scheduling and management. For high availability, the Borgmaster is replicated five times with one elected master. Each replica holds an in-memory copy of the cell's state, which is recorded in a Paxos-based store to maintain a consistent state across replicas. This allows the system to eliminate single points of failure. This master communicates with Borglets, which are local autonomous agents, with one per machine in a cell. Borglets handles tasks and manages resources even if the Borgmaster becomes unreachable. Finally, to optimize utilization, Borg implements a preemptive scheduler that identifies and scores machines based on resource and constraint requirements, while also accounting for low-priority tasks that can be evicted to reclaim resources for higher-priority tasks. Borg ensures that hardware is used with maximum efficiency, and high-priority services will always have the required resources for performance.

How do the authors test their idea, and what do they find?

The authors evaluated Borg using Fauxmaster, a simulation system with real production data from 15 Borg cells. They measured efficiency by evaluating cell compaction, which entailed repeatedly

attempting to fit the workload into fewer machines. The fewer machines used, the more efficient the system. Experiments were repeated 11 times per cell, and the 90th percentile result reflects conservative provisioning. They then modified policies and measured the additional machines needed, quantifying the impact of design choices on real-world cost and utilization.

They tested the effects of cell sharing by running long-running and short-lived batch jobs simultaneously in the same cell. The results indicated that cell sharing could reduce the number of machines needed by 20-30%. Additionally, they tested the effects of splitting a user from a shared cell to a new cell if they used 10 TiB of memory or 100 TiB and found that higher thresholds would require 20-150% more machines. Cell sharing did have a drawback: a negligible 3% reduction in CPU performance within the cell. They also tested the impact of different cell sizes and discovered that smaller cells required significantly more machines compared to larger cells. Furthermore, they tested fine-grained versus fixed-size resource requests and found that fixed-size requests were inefficient, requiring an average of 30-50% more resources.

They also tested the resource reclamation policy, which estimates task resources and reclaims remaining resources for batch jobs. Without this policy, the median cell required 20% more machines. While resource reclamation improved resource utilization, it could lead to out-of-memory events if the estimates were inaccurate. To address this, they tuned the settings on its estimation algorithm and determined that a medium setting was the optimal choice.

Conclusion

There are multiple concepts that Borg applies at the scale of a datacenter that are connected to the process management topics covered in the Operating Systems class. While Borg manages tens of thousands of machines rather than a single OS, the fundamental problems it is designed to solve, such as deciding which processes should run by enforcing priorities, isolating workloads, and presenting an abstraction to users, are similar to some of the common tasks an OS kernel is managing on a single machine.

The class topic of CPU scheduling is closely related to the process management described in the research paper. Scheduling allows the OS to choose which process to run while balancing the limitations of CPU, which affects response time and turnaround time. Borg is managing the choice process on a higher level as well by selecting a machine for a task. This creates a unique version of a queue, where machines with available resources that meet the requirements for a task are getting scored and chosen. Once the machine is selected, the OS then handles the process management at a local level.

The class topics on address spaces and processes are also relevant to the subject of the paper. One of the goals of the OS is to provide a level of abstraction to the user by obscuring the “backend” of physical memory into dedicated per-process address spaces. Borg performs a similar abstraction on a datacenter level. While the “cell” pools thousands of machines, the resource is presented as one logical computer and when users submit jobs they don’t have to worry about which particular machine runs it.

In conclusion, Borg is utilizing both low level and high level process management concepts, adding an extra layer of abstraction on top of a complex system, in order to solve the problems of scaling large volume data processing and resource distribution. While presenting as an intricate system on its face, its scheduling, abstraction and isolation are all shared foundational concepts of operating systems.