Paper Review Form (1000 words maximum)
cs940: Large-scale cluster management at Google with Borg [1]

February 13, 2018

Paper Summary

This paper describes the Borg cluster management system, which had been in use as the primary cluster manager of Google for many years at the time of publication. Borg abstracts cluster management away from the user while providing high-availability and high-parallel tasking, with a quota-based system to manage a multi-user environment. Both availability-critical production tasks and burst-workload non-production tasks can be scheduled with minimal latency through a series of scalable optimisations. Security and performance isolation are preserved on shared-task machines without hindering economically-optimal levels of utilisation. The improvements on Borg leading to the new Kubernetes solution were also discussed by the paper.

Pros and Cons

I believe that the paper has the following positive features:

- The introduction section is succinct and effective in summarising the features of Borg, without overstepping on the role of the related works section later.
- In addition to effective numerical optimisations, user behaviours in the quota system were also taken into account during design and planning.
- Implementation details were first described from the user perspective, then broken down into how different goals were achieved on the architecture level. This description method is particularly effective in describing complex systems with a range of stakeholders.

I believe that the paper has the following negative features:

- The high task startup latency (25s) on machines without prior execution of the task can be obstructive when a sudden spike in demand requires expedited upscaling of a service.
- In evaluating CPU interference, CPI was a poor choice of metric that led to inconclusive results, due to the vast number of factors influencing the metric.

- Despite extensive discussions on how performance isolation has been achieved in practice, this was not backed up with quantitative evaluations on the effects of disruptive or malfunctioning clients.

The Problem/Motivation

The extreme scale and heterogeneity of services [1, §7] operated by the largest internet product providers require very different cluster management strategies in comparison with the computing needs of smaller organisations. In Google's case, the lack of resource assignment and reclamation in open source solutions such as *Mesos* [2] and *YARN* [3] at the time motivated them to develop their own solution from scratch, with controller coordination handled by the lock-based Chubby [4]. Other solutions were also in lack of support for disparate workloads [5] or high resource dimensions [6].

The Solution/Approach

Borg separates user workloads into two classes: availability-critical *prod* jobs, and opportunistic *non-prod* jobs that are usually batched, with control strategies designed accordingly. Available computing machines are arranged in "cells" each controlled by replicated *Borgmaster* processes, which in turn coordinate *Borglet* agents on each machine. This control architecture enforces resource reservation and quota from users, and manages the admission, scheduling, monitoring and debugging access of tasks initiated by each user job. Task placement and scheduling are optimised towards both high resource utilisation and preserving headroom redundancy to ensure availability and minimal latency of *prod* jobs. Container-based security and performance isolation are used to allow task cohabitation on each system, instead of more application-specialised isolation solutions such as Unikernels [7]. Better grouping mechanisms and user accessibility are being implemented for Kubernetes, which is an open-source successor to Borg.

Evaluation

Owing to the scale of deployment, it was not possible for the authors to conduct an evaluation with live workloads. Therefore, snapshots of states from a small number of cells were replayed with the accompanied debugging simulator to evaluate the performance of Borg. Evaluation of availability demonstrated that high availability was achieved for *prod* jobs, with runtime incidents mitigated successfully for both types of jobs. The cell compaction criteria was used to evaluate the utilisation under Borg, through which – despite the inconclusiveness of CPI-based metrics – the authors showed that Borg achieved an economically-optimal utilisation while leaving sufficient expansion headroom for *prod* jobs. The choice of cell size, and Borg's handling of resource requests and reclamation were also shown to be effective. However, performance aspects of Borg's isolation were evaluated through practical observations only.

Your Opinion

Through Borg, an one-size-fits-all solution for cluster management was achieved, albeit specifically designed for Google's needs. There is probably no other workload (nor the amount of computing power) in the world that can be used to evaluate Borg under live conditions, which resulted in limitations on the authors' evaluation methodology. It is also not possible to evaluate the suitability of deploying Borg for other organisations' computation workloads due to its closed-source nature. Therefore, the applicability of research published by this paper may be very limited. However, it was a huge step up for Google to publish how they designed their systems to enable the scale of availability and parallelism they achieved, which supplied many good ideas for implementing more powerful and universal open-source cluster management systems.

Questions for the Authors

- Borg avoided the use of virtualisation due to the lack of hardware virtualisation support at the time of design. With the general availability of virtualisation support in today's hardware, did the new Kubernetes take advantage of the improved security and performance isolation virtualisation offers, and why?
- Borg gives heavy priority to *prod* tasks to ensure their availability. In the event of malfunctioning code in *prod* tasks causing disruptions to other tasks running in the same cell, what facilities are available in Borg to mitigate these disruptions?

References

[1] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015, p. 18.

[2] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center." in *NSDI*, vol. 11, no. 2011, 2011, pp. 22–22.

[3] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 5.

[4] M. Burrows, "The chubby lock service for loosely-coupled distributed systems," in *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006, pp. 335–350.

[5] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: flexible, scalable schedulers for large compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 351–364.

[6] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: Scalable and coordinated scheduling for cloud-scale computing." in *OSDI*, vol. 14, 2014, pp. 285–300.

[7] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, "Unikernels: Library operating systems for the cloud," in *Acm Sigplan Notices*, vol. 48, no. 4. ACM, 2013, pp. 461–472.