Paper Review Form (1000 words maximum) *cs940*: Fuxi: a fault-tolerant resource management and job scheduling system at internet scale [1]

February 13, 2018

Paper Summary

Fuxi is a cluster management system specialised in resource management and job scheduling for Alibaba's data processing systems. It was designed on top of the open-source Yarn [2], but with improved scalability and fault tolerance. The need to efficiently balance the weakest-link problem in centralised resource management and reliability issues in decentralised systems was addressed by Fuxi through incremental resource management and locality tree-based hierarchical scheduling. Resilience against faults at different levels of the system were achieved without the need to maintain a consensus system between controllers. Fuxi performed well under synthetic workloads and fault scenarios during the authors' evaluations.

Pros and Cons

I believe that the paper has the following positive features:

- The addition of application master in common workflows to manage operations of individual jobs frees up FuxiMaster for other duties, improving performance through increased multiprocessing.
- Under the hierarchical locality tree-based scheduling, pending tasks can be distributed relatively evenly to local queues at different levels, improving overall dequeuing performance.
- For fault recovery at FuxiMaster level, separation of *hard* and *soft* states achieve a good compromise between full state-logging and full re-execution to keep both normal and recovery operations perform at a reasonable level.

I believe that the paper has the following negative features:

- Under the Apsara stack, Fuxi seems somewhat fragmented over two layers, without clear explanation on how it interacts with other parts of Apsara not classified as part of Fuxi.

- The use of synthetic workloads in evaluation may not produce a good performance model under normal operations, and the comparison in GraySort performance ignored differences in machine specifications.
- There is no significant relation between description of jobs and the fault tolerance mechanisms in §4. The descriptions of jobs could have been better placed earlier in the paper.

The Problem/Motivation

Fuxi drew inspirations from cluster management solutions similar to Borg [3], and identified the lack of support for high dimensions of resources and quota-based demand management in existing solutions [2] [4]. As a high-parallel system, messaging overhead and the lack of local resource reuse in Yarn significantly impacted its scalability. In the fault tolerance aspect, Yarn also exhibited an unacceptable amount of recovery overhead for Alibaba's needs. Further more, the ability to dynamically adjust resource allocations was also missing in existing solutions such as Condor [5]. Fuxi was designed as a combined solution to address these scalability and fault tolerance issues. Managing overestimation of resource demand was later partially achieved in Borg [3] through resource reclamation.

The Solution/Approach

Similar to Borg, Fuxi comprises of three levels of control mechanisms: FuxiMaster manages resource requests and demands at the cluster level, with redundancies rather than a consensus system to mitigate failures; FuxiAgent performs monitoring and control coordination on each machine of the cluster; and application masters control the execution of each task over machines and coordinate with the aforementioned components. Scheduling and communication are conducted incrementally to minimise overhead as well as to take advantage of resource and queue locality offered by the locality tree. A dynamic quota system with priority and batch handling is used to provide high performance multi-tenancy support, with resources provisioned in buckets. Fault tolerance of the system is achieved through redundancies and caching of necessary state information at all levels of control mechanisms, each of which can individually recover from failures without significant overall performance impact. Additionally, multi-level health monitoring is used Fuxi to identify and avoid failing machines in the cluster.

Evaluation

After presenting statistics from production operations, the overall scheduling overhead and levels of resource utilisation were evaluated under a synthetic workload in a live 5000-server Fuxi system. Despite some fluctuations in scheduling time under heavy system load, the peak scheduling consumption was deemed as acceptable by authors. Reasonably high levels of memory and CPU utilisation were achieved, with discrepancies explained by the authors.

Under fixed synthetic workload, overheads at application master and worker level were low although not insignificant.

A cross comparison of Fuxi's GraySort performance with prior work was provided, however under similar hardware Yahoo's Hadoop implementation would have achieved greater TB/min/node. Whether Hadoop's perceived higher coordination overhead would have prevented it outperforming Fuxi at 2100 nodes rather than 5000 was not discussed. Finally, a reasonable amount of full-execution overhead was observed when various types of faults were injected into Fuxi's execution.

Your Opinion

In this paper the authors presented a cluster management solution that is even further specialised than Borg in handling internal workloads, but did preserve some user-friendly features such as multi-tenancy support and locality-specifications. The ability for users to debug their own jobs was not mentioned, in comparison with Borg's ability for users to troubleshoot themselves through jailed SSH connections. Unlike for Borg, the authors of Fuxi were able to perform live evaluations rather than through simulations. Discrepancies in benchmark cross-comparison with prior solutions were not well explained, potentially due to limited ability of the authors to adjust the number of nodes in the cluster and execute benchmarking many times on 5000 production servers. The language quality of this paper could have been improved through further proofreading to eliminate some issues with word choice and grammar.

Questions for the Authors

- Fuxi is somewhat similar in its design goals to Google's Borg. Borg provides the convenient *borgssh* facility for users to securely log into machines running their jobs and debug any problems. Does Fuxi provide a similar functionality?
- The machine specifications of your GraySort benchmark and Yahoo's the year prior are almost identical. With no additional coordination overhead, Yahoo's Hadoop could have outperformed Fuxi at 2100 nodes. What considerations have you given into the number of nodes used in benchmarking?

References

[1] Z. Zhang, C. Li, Y. Tao, R. Yang, H. Tang, and J. Xu, "Fuxi: a fault-tolerant resource management and job scheduling system at internet scale," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1393–1404, 2014.

[2] 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.

[3] 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.

[4] 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.

[5] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: the condor experience,” *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323–356, 2005.