

Topologic-aware Allocation Policies for Jellyfish

ABSTRACT

1. INTRODUCTION

Jellyfish topology [1] has been recently proposed as a high bandwidth and low latency interconnect for large scale data centers and HPC systems. In opposite to recently proposed server-centric datacenter networks (DCN), jellyfish is an indirect (switch-centric) network in which the servers are connected to the switches. This network is a *degree-bounded* random regular graph (RRG) among the top-of-rack (ToR) switches, in which all the nodes have the same degree, are bidirectional and are connected randomly. RRGs also provide other desirable properties for an interconnect such as low diameter, high connectivity among nodes and easy incremental expansion.

As stated in the original paper [1], routing in jellyfish is a challenge. Although jellyfish provides high connectivity among switches, classical routing policies are not able to exploit the path diversity offered. In that work the authors evaluated two well-known routing policies, shortest path (SP) and Equal-cost Multi-path (ECMP), assessing that the use of the shortest paths does not provide enough path diversity to utilize the full capacity of the network. This issue was solved using the K-Shortest Path (KSP) [2] routing policy that uses more paths at the cost of being longer. Although KSP performs well compared to SP and ECMP, the author in [3] showed that jellyfish has several features that make it ineffective. In particular they stated the possibly large number of source-destinations (SD) pairs that will share the same K shortest-paths and the random number of short paths between each pair of switches.

These works have studied jellyfish both theoretically, putting bounds to topological properties, and empirically evaluating the performance of several communication patterns. However none of them have considered the natural scenario in which this topology could be used: data centers or HPC centers where many applications run concurrently. To the best of our knowledge, there is no work devoted to evaluate the performance of such applications in this topology. The assignment of resources to application has been widely studied in the context of HPC. Those works clearly differentiate three stages: selection of the application to be executed, allocation of the resources to that application and mapping of the tasks that compose the application to the physical servers.

The objective of this work is the evaluation of existing allocation policies applied to jellyfish in multi-application sce-

narios. We will see how these policies do not perform well in jellyfish due to the random nature of this topology. Because of that, we will apply ideas taken from the HPC world and adapt them to the particularisms of jellyfish. In particular we want to define concepts of locality to jellyfish. In addition we will extend the definition of contiguity and convexity to jellyfish. These policies should improve the performance of applications executed on jellyfish. However, contiguity can severely affect system utilization. This drawback will also be studied.

The evaluation of the allocation policies has been carried out using extensive simulation with a representative mix of parallel application. Results show that traditional and simple allocation strategies do not take advantage of the characteristics of jellyfish. Moreover we will see how locality-aware policies can take advantage of those properties improving the performance of applications running concurrently. In particular locality and contiguous policies ..., We will also evaluate the drawbacks....

The rest of the paper is organized as follows. Section 2 introduces jellyfish and communication patterns used throughout the paper. We continue in Section 3 analyzing simple allocation policies for jellyfish introducing in Sections [?] and [?] the proposed policies. The results are presented and analyzed in Section [?] concluding in Section ?? with some conclusions and future lines of work.

2. BACKGROUND

In this section we describe thoroughly the jellyfish topology and give some definitions and properties that will be used in the rest of the paper. Furthermore, we describe the four routing policies developed for jellyfish assessing the pros and cons of each one of them.

2.1 Jellyfish topology

Jellyfish is a network topology in which the switches are connected randomly. We denote jellyfish as $RRG(N,k,r)$ where N is the number of switches, k is the number of ports of the switches and r is the number of ports used to connect to other switches.

2.2 Routing policies

The jellyfish topology provides high path diversity between any pair of servers. This is one of the characteristic that makes this topology so appealing to execute parallel workloads. As we have seen, there exists a lot of possible

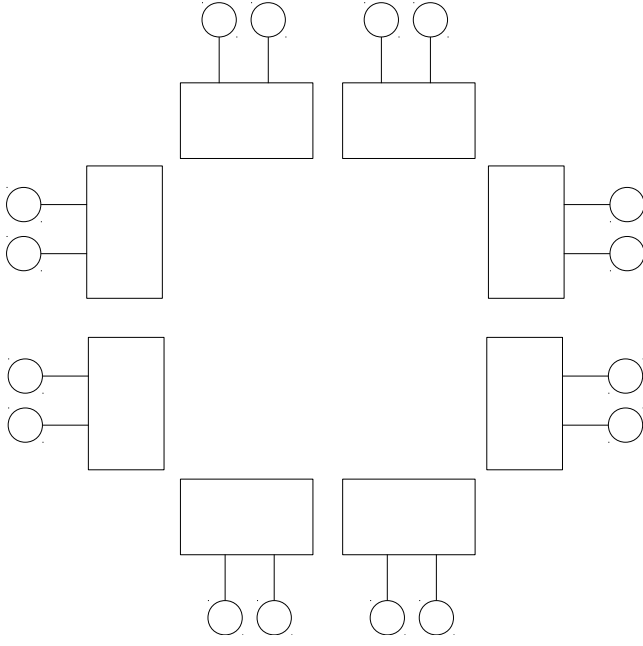


Figure 1:

paths among physical servers to carry out the communications. This is the reason why the choosing of a routing policy is so important in jellyfish in order to avoid contention in network links. There exist only four routing policies proposed and evaluated for jellyfish. Next, we will analyze these policies stating the improvements of one against the others.

- Shortest Path (SP):
- Equal-Cost Multi-Path (ECMP):
- K-Shortest Path:
- Limited Length Spread K-shortest Path (LLKSR):

2.3 Workloads and performance metric

- Nearest neighbor:
- Collective:
- : Permutation and shift pattern:

3. ALLOCATION IN JELLYFISH

When an application is submitted to be executed, the allocator is in charge of finding the appropriate set of resources to place it. There are many works that propose allocation policies for other topologies such as meshes, tori [4] [5], or fattrees [5] [6], but, to the best of our knowledge, not for jellyfish. Only in [3] the authors use some simple mono-application allocation policies without taking into account how these policies behave when multiple application run concurrently. In this section we describe and analyze those policies adding a new version of one of them due to random structure of jellyfish.

In order to define the allocation policies we must first assign a node ordering to the servers and switches that

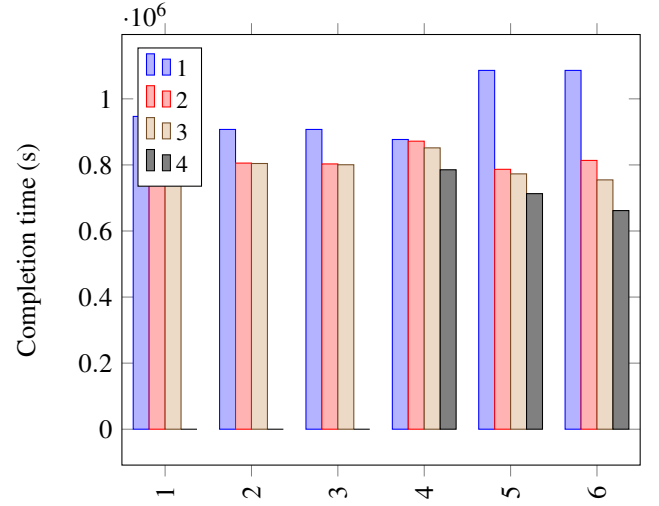


Figure 2: Completion time in second employed to process three traffic patterns comparing.

compose the network. Due to the random structure of jellyfish, we first define order the switches and then, inside each switch, we order the server consecutively.

- Sequential: The tasks are assigned to the servers consecutively using the identity function.

$$\pi : T \rightarrow P\pi(t_i) = p_i \quad (1)$$

- Random: The tasks of the application are assigned to the servers uniformly at random.

$$\pi : T \rightarrow P\pi(t_i) = p_j \text{ where } p_j \text{ is selected at random} \quad (2)$$

- Random*: This is a special version of the previously presented policies. In this case we select a switch uniformly at random and then we map (k-r) tasks to the servers directly connected to it.

$$\pi : T \rightarrow P\pi(t_i) = p_j \text{ where } p_j \text{ is selected at random} \quad (3)$$

LEMMA 1.

4. LOCALITY AND CONTIGUITY IN JELLYFISH

5. EXPERIMENTAL SET-UP

5.1 Simulation environment

5.2 Communication patterns

5.3 Performance metrics

6. ANALYSIS OF THE RESULTS

7. CONCLUSIONS AND FUTURE WORK

8. REFERENCES

- [1] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, “Jellyfish: Networking data centers randomly,” in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI’12, (Berkeley, CA, USA), pp. 225–238, USENIX Association, 2012.
- [2] J. Y. Yen, “Finding the k shortest loopless paths in a network,” *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.
- [3] X. Yuan, S. Mahapatra, W. Nienaber, S. Pakin, and M. Lang, “A new routing scheme for jellyfish and its performance with hpc workloads,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC ’13, (New York, NY, USA), pp. 36:1–36:11, ACM, 2013.
- [4] J. A. Pascual, J. Miguel-Alonso, and J. A. Lozano, “Optimization-based mapping framework for parallel applications,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 10, pp. 1377 – 1387, 2011.
- [5] J. Navaridas, J. A. Pascual, and J. Miguel-Alonso, “Effects of job and task placement on parallel scientific applications performance,” in *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pp. 55–61, Feb 2009.
- [6] J. A. Pascual, J. Navaridas, and J. Miguel-Alonso, *Effects of Topology-Aware Allocation Policies on Scheduling Performance*, pp. 138–156. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.