Memory Disaggregation in Serverless Computing

Joshua E. Lowe
Rose-Hulman Institute of Technology
Terre Haute, IN, USA
lowej2@rose-hulman.edu

Nanda Velugoti (Advisor)
Illinois Institute of Technology
Chicago, IL, USA
nvelugoti@hawk.iit.edu

Kyle C. Hale (Advisor)
Illinois Institute of Technology
Chicago, IL, USA
khale1@iit.edu

Abstract

The slowing of advancements in memory technology and applications' increasing demand for memory have resulted in computation becoming bottlenecked by availability of memory. "Far memory" mitigates this by swapping pages to a remote machine rather than a local disk. This paper first explores the viability of integrating one Function-as-a-Service (FaaS) tool, Globus Compute[4], with a remote swap system for far memory, FastSwap[1]. Then, it examines the workloads for which using remote memory is possible without excessive overhead cost. We find that for certain workloads, including breadth-first search and minimum spanning tree, it is possible to use up to 30% remote memory without significant slowdowns.

Keywords: far memory, function-as-a-service, remote-direct-memory-access

ACM Reference Format:
Joshua E. Lowe, Nanda Velugoti (Advisor), and Kyle C. Hale (Advisor). 2024. Memory Disaggregation in Serverless Computing. In *Proceedings of The International Conference for High-Performance Computing, Networking, Storage, and Analysis (SC24).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Memory technology is no longer advancing as rapidly as other computational resources, while applications grow increasingly memory-dependent, resulting in memory bottlenecks. One solution explored previously is "memory disaggregation," the ability of compute nodes to access memory on remote machines. Fastswap implements this by modifying the Linux kernel to redirect swaps to a remote server. This has been shown to improve overall performance and throughput for compute clusters, by increasing memory availability and decreasing idle time.

SC24, November 17–22, 2024, Atlanta, GA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Serverless computing services have become increasingly prevalent. Globus Compute is one such FaaS platform developed by Globus Labs. It allows users to both submit tasks and host compute endpoints. The integration of remote memory with an existing FaaS platform has not been extensively tested; hence, we will explore whether these two softwares are compatible, how much overhead is associated with running a workload on remote memory, and which workloads can safely incorporate remote memory.

2 Related Work

The most closely related work is rFaaS [2], a remote-memory-based FaaS platform for high-performance computing. The authors use MILC and LULESH to benchmark their system, measuring the overhead of jobs when memory on the executing machine is accessed repeatedly. They find that there is up to 12% overhead depending on the memory activity.

Another related work is Āpta [5], a memory disaggregation system that uses Compute Express Link (CXL). Āpta improves upon both CXL and RDMA solutions, offering 21-90% improvement over RDMA and 15-42% improvement over CXL on common FaaS workloads.

Unlike Fastswap, rFaaS uses Remote Memory Access (RMA) instead of Remote Direct Memory Access (RDMA). It also measures the overhead of running simultaneous workloads rather than the overhead of remote memory itself. Apta uses CXL rather than RDMA, and it focuses more on cache coherence than speed.

3 Methodology

For this study, we use Illinois Institute of Technology's tinker-0 and tinker-1 machines, which are physically connected with a network switch. Each machine has the following specifications:

- 2x Intel® Xeon® E5-2630 v4 Processor – @3.10GHz
- 2x 16GB + 4x 4GB (48 GB) of DDR4-2,133 ECC Registered RAM
- Broadcom NetXtreme II BCM57800 1/10 Gigabit Ethernet
- Mellanox MT27500 Family ConnectX-3 card
- Linux Ubuntu 16.04 LTS

We first install the required RDMA drivers and set up IPs for the network cards. We then set up the Fastswap client and server along with the Globus Compute server, in accordance with Figure 1.

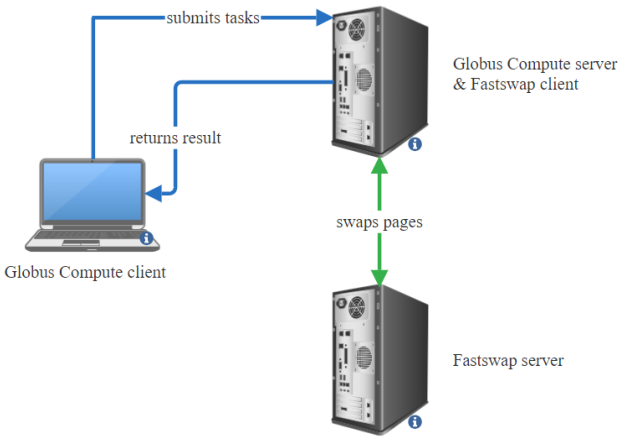

Figure 1. Configuration of Fastswap and Globus Compute

We create scripts using the breadth-first search and minimum-spanning tree functions in the SeBS benchmarking suite [3] that submit a task to a Globus Compute endpoint running on the tinker-0 machine. We use Cgroups2 to restrict the amount of available local memory, then modify Globus to set the limit based on a function parameter. We also execute the TaPS benchmarking suite [6]'s Molecular Design benchmark.

4 Results

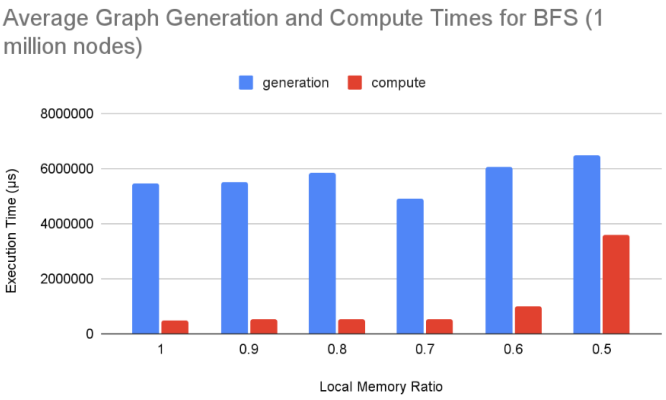

Figure 2. Runtime of the breadth-first search benchmark on a problem space of 1 million nodes. Times are averaged over 10 runs with different local-remote memory ratios.

In Figure 2, we observe that graph generation time is not significantly affected by variation in memory ratio. Computation time does not seem to be affected for 70% local memory or higher. Moving to 60% and 50% local memory results in a slight increase and a major increase respectively.

In Figure 3, similar results to Figure 2 are obtained. The runtime of MST significantly increases when local and remote memory are split equally.

In contrast, Figure 4 demonstrates drastically different behavior. Increases in runtime follow a semi-linear trend, and the overhead of using remote memory is much more costly.

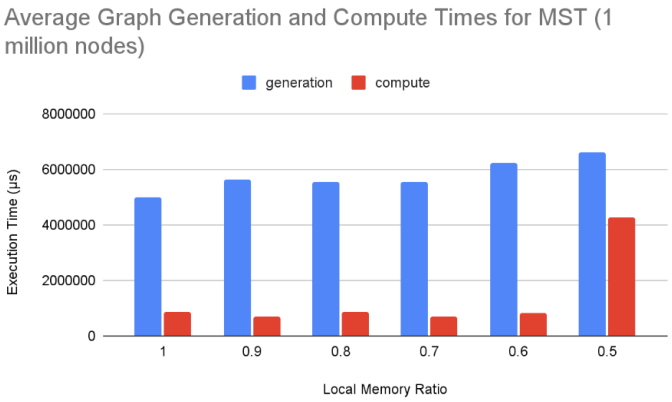

Figure 3. Runtime of the minimum spanning tree benchmark on a problem space of 1 million nodes. Times are averaged over 10 runs with different local-remote memory ratios.

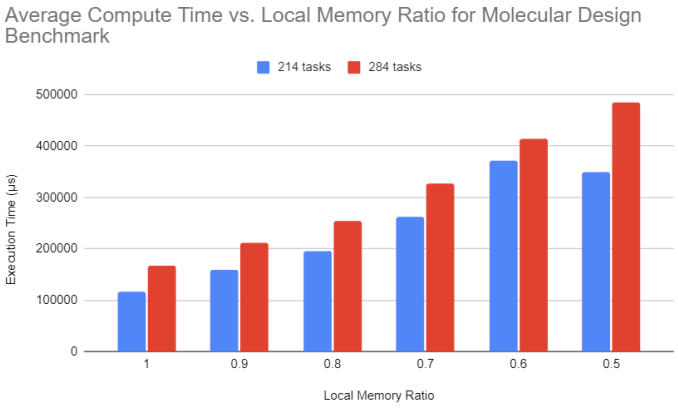

Figure 4. Runtime of the molecular design benchmark with 16 molecules simulated. Times are averaged over 10 runs with different local-remote memory ratios.

5 Discussion

It is plausible that the sharp increase in runtime originates from the job no longer fitting in local memory. As both benchmarks store and traverse a graph, the memory cutoff may be lower than the graph size, resulting in "hot" pages being swapped to remote memory.

One explanation for the difference between Molecular Design and the graph-based benchmarks is the task execution pipeline. Many tasks are submitted to Globus Compute for Molecular Design as opposed to just one, and the tasks are dependent on previous tasks' results, leading to compounding costs.

6 Conclusions and Future Work

In this paper, we have demonstrated the viability of remote memory for certain workloads. We aim to extend this work to other real-world FaaS workloads, particularly TaPS's Protein Docking and Mosaic benchmarks. By running these and profiling their memory usage, we hope to draw conclusions about the usability of this system for FaaS workloads. We also hope to develop a policy for remote memory usage, allowing the optimization of FaaS platforms' throughput.

References

[1] Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo, Amy Ousterhout, Marcos K. Aguilera, Aurojit Panda, Sylvia Ratnasamy, and Scott Shenker. 2020. Can far memory improve job throughput?. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 14, 16 pages. https://doi.org/10.1145/3342195.3387522

[2] Marcin Copik, Marcin Chrapek, Larissa Schmid, Alexandru Calotoiu, and Torsten Hoefler. 2024. Software Resource Disaggregation for HPC with Serverless Computing. *arXiv preprint arXiv:2401.10852* (2024).

[3] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. 2021. Sebs: A serverless benchmark suite for function-as-a-service computing. In *Proceedings of the 22nd International Middleware Conference*. 64–78.

[4] Globus Labs. [n. d.]. Globus Compute. https://github.com/globus/globus-compute. Accessed: Aug. 7, 2024.

[5] Adarsh Patil, Vijay Nagarajan, Nikos Nikoleris, and Nicolai Oswald. 2023. Āpta: Fault-tolerant object-granular CXL disaggregated memory for accelerating FaaS. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 201–215. https://doi.org/10.1109/DSN58367.2023.00030

[6] Proxystore. [n. d.]. TAPS. https://github.com/proxystore/taps/tree/main/taps. Accessed: 2024-08-07.