Memory Disaggregation in Serverless Computing

Joshua Lowe[1,2] Nanda Velugotti (Advisor)[1,3] Kyle Hale (Advisor)[1,3]

[1] BigDataX, [2] Rose-Hulman Institute of Technology, [3] Illinois Institute of Technology

Introduction

Problem Statement: Integrate a memory disaggregation system into a FaaS platform and benchmark the resulting system, characterizing different workloads' susceptibility to remote memory

Goals:

- Integration of Fastswap and Globus Compute platforms
- Analyze FaaS benchmarks' performance on different memory ratios

Motivation

- HPC workloads are increasingly bottlenecked by availability of memory
- Disaggregation mitigates memory shortages and allows more tasks to run on Globus

Software and Setup

Fastswap [1]: A memory disaggregation system that uses RDMA to support far memory

- Connects two computers with Mellanox ConnectX-3 network cards for fast data movement
- Applies a Linux kernel patch to modify swap system
- Takes advantage of pre-activated swap code paths to redirect swaps to external memory
- Once memory usage hits a threshold, swaps pages to external remote machine
- Up to 10% performance improvement for memory-intensive workloads

Globus Compute [2]: A Function-as-a-Service platform for scientific high-performance remote function execution

- Built on top of the Parsl HighThroughputExecutor to distribute and execute tasks
- Tasks are submitted through the cloud, executed, and results are sent back

Integration:

- Modified Globus to manage memory via cgroups2
- Prior to execution, local memory limit is restricted to given value

Architecture

Tasks are submitted from a local machine to the Globus Compute server, which is also running the Fastswap client. This allows the machine to swap pages back and forth with a connected Fastswap server machine. 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
- Mellanox MT27500 Family ConnectX-3 card
- Linux Ubuntu 16.04 LTS

Architecture

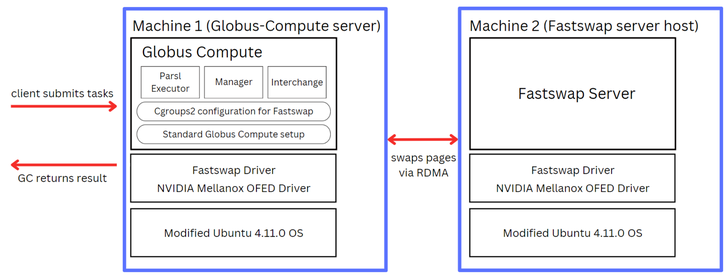

Fig. 1: Architecture

Results

Benchmarks run:

- Breadth-first search (SeBS [4])
- Minimum spanning tree (SeBS [4])
- Molecular Design (TaPS [3])

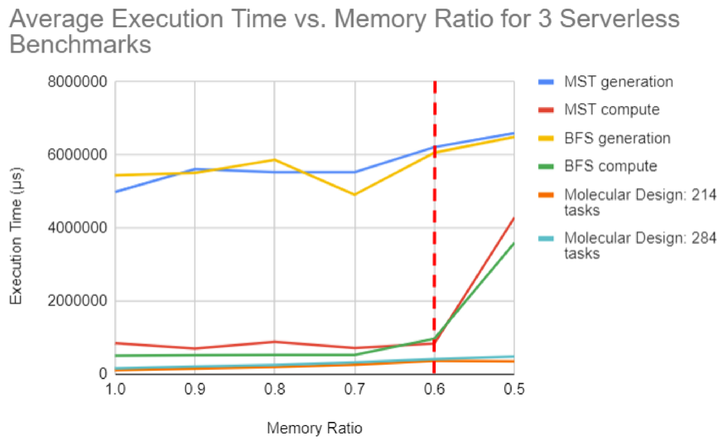

Fig. 2: Average Execution Time vs. Local Memory Ratio for Multiple Benchmarks

Both experiments were run with 10 repetitions. An initial run was completed without memory restrictions to determine peak memory usage, from which the appropriate restricted memory values were derived.

Discussion

- The red dashed line indicates the threshold at which significant performance loss occurs for the computation portion of the graph-based benchmarks
  - BFS and MST show very little effect until a 0.6 local/remote memory ratio
  - Likely due to the large workload and built-in cloud latency
  - Also possible that the graph is no longer fully stored in local memory (and thus is swapping "hot" nodes
  - Graph generation likely does not exhibit the same behavior because it doesn't perform computation on the nodes (and thus does not require many swaps)
- Until that point, Molecular Design's relative cost is more significant than the graph problems
  - Reasonable to assume that this is due to the different memory access pattern
- Molecular Design has two types of executions: 214 tasks (with no failed tasks) or 284 tasks (with some failed tasks)

Future Work

- Further categorize the class of workloads with acceptable overheads in runtime
- Characterize memory usage and determine correlation to performance loss
- Identify thresholds where significant loss occurs
- Predict thresholds for new workloads
- Determine a policy that can be implemented in Globus to restrict memory prior to execution for such workloads
- Engage in additional experiments with weak and strong scaling behaviors
- Explore configurations of Slurm for testing workloads on supercomputers

References

[1] Emmanuel Amaro et al. "Can far memory improve job throughput?" In: *Proceedings of the Fifteenth European Conference on Computer Systems*. EuroSys '20. Heraklion, Greece: Association for Computing Machinery, 2020. ISBN: 9781450368827. DOI: 10.1145/3342195.3387522. URL: https://doi.org/10.1145/3342195.3387522.

[2] Globus. *Globus Compute*. https://github.com/globus/globus-compute. Accessed: Aug. 7, 2024.

[3] Proxystore. *TAPS*. https://github.com/proxystore/taps/tree/main/taps. Accessed: 2024-08-07.

[4] SPCL. *SEBS (Serverless Benchmark Suite)*. https://github.com/spcl/serverless-benchmarks/tree/master/sebs. Accessed: Aug. 7, 2024.