Software Resource Disaggregation for HPC with Serverless Computing

Marcin Copik
marcin.copik@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Alexandru Calotoiu (advisor)
alexandru.calotoiu@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Torsten Hoefler (advisor)
htor@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

1 POSTER DESCRIPTION

HPC systems are facing the problem of resource underutilization, regardless of their size, power, and node heterogeneity [8]. However, rigid batch systems on homogeneous nodes with coupled hardware prevent fine-grained resource allocations. Potential solutions to this problem include *resource disaggregation* and *job co-location*. Hardware disaggregation requires redesigning data centers and forces applications to always pay the latency price when accessing remote resources, which is not the case for traditional HPC systems. While co-locating jobs improves the system's efficiency [9], node sharing raises security issues and can lead to performance degradation due to resource contention. Users and system operators need to understand the *symbiosis* of applications [2].

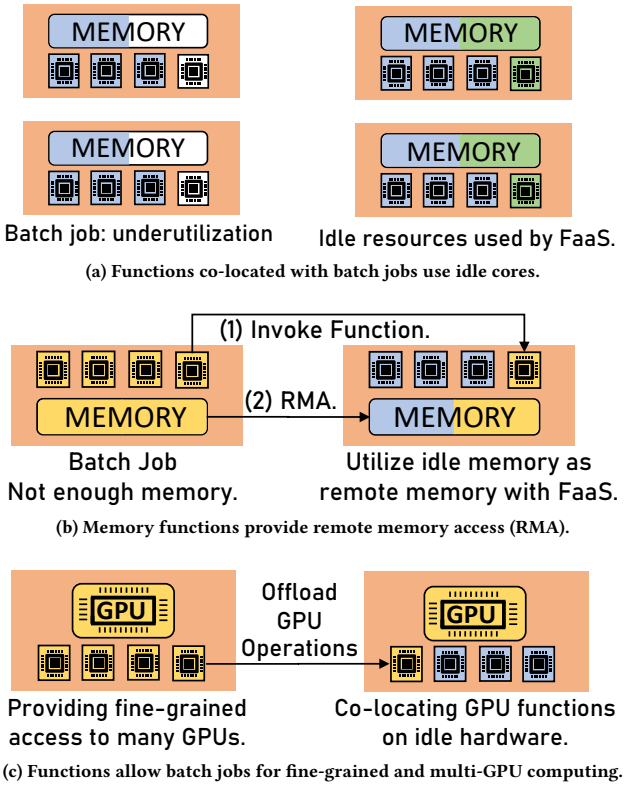
We propose a new approach that meets all the above criteria: fine-grained allocations, improved utilization, and deployment to existing HPC systems. First, we use *Function–as–a–Service (FaaS)*, a new cloud programming paradigm, to allocate short-running functions on dynamically allocated HPC resources. Functions provide **software disaggregation** by allowing users to access remote resources with the help of an elastic programming model (Fig. 1). Then, we integrate **a high-performance FaaS platform** [6] in the HPC software stack. We use HPC benchmarks to show the improved system utilization and demonstrate how HPC applications benefit from elastic computing with functions.

1.1 Software Disaggregation

We allocate functions on idle resources in partially allocated and unoccupied nodes to target three resources: CPU cores, memory, and GPUs (Fig. 1). Instead of using nodes exclusively for one batch job, jobs are co-located with short-running serverless functions. Users are encouraged to share nodes and spread jobs since leaving one core free per node allows hosting remote memory and GPU functions. Users can be compensated for sharing-related performance losses, e.g., by using existing systems for fair pricing on shared nodes [3].

CPU Node Sharing We implement *job stripping* [2, 9] by co-locating functions with applications (Fig. 1a). Short-running MPI processes can be allocated as functions, aiding adaptive MPI implementations [4, 10]. Functions are easy to profile and characterize, helping to match batch jobs with complementary resource consumption.

Memory Sharing In HPC, memory is over-provisioned due to variable memory usage across applications, processes in the same program, and within the job lifetime [11]. To benefit from modern networks that offer latencies low enough for remote memory access [7], we run functions exposing idle node memory (Fig. 1b).



(a) Functions co-located with batch jobs use idle cores.

(b) Memory functions provide remote memory access (RMA).

(c) Functions allow batch jobs for fine-grained and multi-GPU computing.

Figure 1: The three models of software resource disaggregation with FaaS: increasing resource utilization on existing HPC hardware.

Functions offer fine-grained scalability, multitenant isolation, and fast resource reclamation to support high-memory jobs.

GPU Sharing While many HPC applications use GPU acceleration, not all require such a device. Thus, we offer the opportunity to run GPU functions alongside CPU batch jobs because the former requires CPU only to schedule GPU kernels (Fig. 1c). Functions can reside in device memory until they are evicted, providing warmed-up data for faster processing.

1.2 High-Performance FaaS

Cloud functions are designed for cloud applications and are not performance-oriented [5]. On the other hand, *HPC functions* must be specialized for the hardware and software of supercomputers. We use `rFaaS` [6], a high-performance serverless platform that uses

high-speed interconnects common in HPC systems. We extend rFaaS to support HPC-oriented containers like Sarus and Singularity, high-performance filesystems, direct networking, and GPU resources. We enable cheap computations on underutilized resources by offering an API for batch systems to release idle nodes and resources into `rFaaS`. Furthermore, we use the pools of idle memory to host warmed function containers and reduce the frequency of cold startups, a major problem for serverless performance.

1.3 Results

We evaluate our approach by co-locating function-like workloads with `LULESH` and the `su3_rmd` from MILC. First, we simulate function executions by co-locating short-running NAS benchmarks with batch jobs, showing an improvement in node utilization of up to 52%. We co-locate batch jobs with functions serving remote memory write and read operations, observing a slowdown $< 5\%$ and 15% for LULESH and MILC, respectively, even when handling 10 GB/s of traffic. Then, we co-locate batch jobs with Rodinia benchmarks simulating GPU functions running for just a few hundred milliseconds, demonstrating low overhead.

Finally, we show how functions can help parallel applications to offload computing tasks to remote resources. First, we evaluate an MPI benchmark where each rank offloads half of Jacobi solver iteration to a function, obtaining a speedup between between 1.7x and 2.2x. Then, we use the Black-Scholes application from PARSEC suite [1] to compare the OpenMP version against remote execution with rFaaS. We demonstrate high efficiency on millisecond-scale computations and efficient scalability until network saturation is reached.

1.4 Summary

We propose a novel *software disaggregation* approach to co-locate long-running batch jobs with serverless functions to open new ways of using remote and underutilized resources in HPC applications. We port a high-performance FaaS platform to a supercomputing system and demonstrate the efficiency of software disaggregation on three major domains: processors, memory, and GPUs. Finally, we show that FaaS can accelerate MPI and OpenMP programs by using idle resources.

REFERENCES

[1] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques* (Toronto, Ontario, Canada) *(PACT '08)*. Association for Computing Machinery, New York, NY, USA, 72–81. https://doi.org/10.1145/1454115.1454128

[2] Alex D. Breslow, Leo Porter, Ananta Tiwari, Michael Laurenzano, Laura Carrington, Dean M. Tullsen, and Allan E. Snavely. 2016. The case for colocation of high performance computing workloads. *Concurrency and Computation: Practice and Experience* 28, 2 (2016), 232–251. https://doi.org/10.1002/cpe.3187 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3187

[3] Alex D. Breslow, Ananta Tiwari, Martin Schulz, Laura Carrington, Lingjia Tang, and Jason Mars. 2013. Enabling fair pricing on HPC systems with node sharing. In *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–12. https://doi.org/10.1145/2503210.2503256

[4] Isaías Comprés, Ao Mo-Hellenbrand, Michael Gerndt, and Hans-Joachim Bungartz. 2016. Infrastructure and API Extensions for Elastic Execution of MPI Applications. In *Proceedings of the 23rd European MPI Users' Group Meeting* (Edinburgh, United Kingdom) *(EuroMPI 2016)*. Association for Computing Machinery, New York, NY, USA, 82–97. https://doi.org/10.1145/2966884.2966917

[5] 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 (Middleware '21)*. Association for Computing Machinery. https://doi.org/10.1145/3464298.3476133

[6] Marcin Copik, Konstantin Taranov, Alexandru Calotoiu, and Torsten Hoefler. 2021. rFaaS: RDMA-Enabled FaaS Platform for Serverless High-Performance Computing. arXiv:2106.13859 [cs.DC]

[7] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. 2017. Efficient Memory Disaggregation with Infiniswap. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 649–667. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/gu

[8] Awais Khan, Hyogi Sim, Sudharshan S. Vazhkudai, Ali R. Butt, and Youngjae Kim. 2021. An Analysis of System Balance and Architectural Trends Based on Top500 Supercomputers. In *The International Conference on High Performance Computing in Asia-Pacific Region* (Virtual Event, Republic of Korea) *(HPC Asia 2021)*. Association for Computing Machinery, New York, NY, USA, 11–22. https://doi.org/10.1145/3432261.3432263

[9] Matthew J. Koop, Miao Luo, and Dhabaleswar K. Panda. 2009. Reducing network contention with mixed workloads on modern multicore, clusters. In *2009 IEEE International Conference on Cluster Computing and Workshops*. 1–10. https://doi.org/10.1109/CLUSTR.2009.5289162

[10] Ao Mo-Hellenbrand, Isaías Comprés, Oliver Meister, Hans-Joachim Bungartz, Michael Gerndt, and Michael Bader. 2017. A Large-Scale Malleable Tsunami Simulation Realized on an Elastic MPI Infrastructure. In *Proceedings of the Computing Frontiers Conference* (Siena, Italy) *(CF'17)*. Association for Computing Machinery, New York, NY, USA, 271–274. https://doi.org/10.1145/3075564.3075585

[11] Darko Zivanovic, Milan Pavlovic, Milan Radulovic, Hyunsung Shin, Jongpil Son, Sally A. Mckee, Paul M. Carpenter, Petar Radojković, and Eduard Ayguadé. 2017. Main Memory in HPC: Do We Need More or Could We Live with Less? *ACM Trans. Archit. Code Optim.* 14, 1, Article 3 (mar 2017), 26 pages. https://doi.org/10.1145/3023362