Farview

We have seen the attempts to disaggregate compute and storage in systems like LegoOS and different services provided by famous cloud providers like AWS and GCP. This helps to provide more scalability and flexibility. Taking it one step further, Farview disaggregated the RAM from the CPU and connected it over a RoCE v2 at 100 Gbps network(effectively separating query processing memory buffer management). They use RDMA to transfer data effectively and directly read data from the remote buffer pool. It provides support for query offloading. Since there's a growing popularity of in-memory databases, but the size of DRAM is restricted. This leads to a lot of data movement and the biggest pitfall. Using the disaggregated model has the following advantages: Reducing data movement and centralizing the buffer cache to remove unnecessary copying of data to compute node. FV supports projection, selection, grouping and encryption/decryption. FV follows the FGPA based architecture since it supports concurrent access to memory, stream processing capacity for operator push-down and substantial amount of local memory.
FV is divided into three stack: network, memory and operator.
The network stack manages the external connections and its scheduling for effective and fair share of the resources and handles the RDMA requests.
The memory stack handles the address translation and isolation for the concurrent accesses.
The operator stack acts as a stream processor on the data that moves from memory to network.
The data is interfaced using AXI streaming handshake protocol and helps with portability. It also has a FV verb for controlling the operators and contains parameters to the memory on how to read and process the data for no additional memory overhead. It has multiple Dynamic regions for handling multiple concurrent queries. Hence, FV is able to effectively implement a network attached memory and addresses its challenges in an efficient manner.

Comments:
I am really curious if they further released a paper to cover the cache replacement policies query processing elasticity. I feel they should have had a future work section in the paper. They also mentioned about porting it to Enzian and just feeds the reader one line input description about Enzian. I would have appreciated if they had mentioned it in more elaborated manner on why it could have been more useful and how it would have further affected the evaluations.

I understand that 100 Gbps is a lot of bandwidth but how will the evaluation change if there's a heavy network congestions and how are tail latencies handled.

RDMA was I feel a great choice since there's no CPU intervention and helps with scaling.
I think there's a typo in 2.1,, it should be LegoOS and not LegOS. Also in fig.2 for the operator stack, shouldnt it be operator N(or some other variable) instead of operator 3 because in the memory stack, there are M channels.

Since the data is transferred over the network, are they serialized/deserialized or do they use any open source file format like parquet.

In section 5.1, it mentions that the operator pipelines are "precompiled". It also mentions that FV deployments are portable on different platforms. If it is precompiled, shouldnt it be architecture specific making it not portable? If it is not precompiled, you just send the code to where the data is like something we had seen in redshift and other papers. I am kind of confused with this part.

Since Fpga is a trusted module, shouldn't the hardware owner be a potential malicious actor who can gain access to the data? Something like the enclave mechanism in Azure AE would be helpful here.