DDC: disaggregated datacenter: Each resource type is built as a standalone resource “blade” and a network fabric interconnects these resource blades.

Key enabling/blocking: network, low latency communications

Paper evaluating the minimum (bandwidth and latency) requirements the network in DDC.

7 applications: Hadoop, Spark, GraphLab, Timely datflow, Spark Streaming, memcached, HERD, sparkSQL

**requirements:**

Bandwidth: 40 – 100Gbps: maintain application-level performance: easy for switch and NIC HW

Network latency 3-5 us. Challenging

Rack scale: disaggregation at the datacenter scale is feasible

TCP or DCTCP fail to meet requirement for low latency communication

CPU-memory disaggregation: expanding memory hierarchy to include a remote level

Partial CPU-memory disaggregation: cache coherence traffic cannot be cached and impacts the network. Controller ASIC on blade: implements address translation between CPU’s view and address used internally within the blade.

Network requirements evaluation methodology

The dominant impact to application performance comes from CPU-memory disaggregation.

Remote memory managed at the page granularity:

2 sources of performance penalty: 1. The software overhead for trap and page eviction 2. Time to transfer pages over the network only consider latter

compare server-centric(legacy) architecture (present worst case) VS disaggregated context

application: Hadoop wordcount Hadoop sour, Graphlab CF, memcached YCSB, Spark wordcount, Spark Sort, Spark SQL BDB, Timely Dataflow Pagerank, HERD YCSB

swap device backed by remaining physical memory rather than disk: partition main memory into “local” and remote (page replacement algorithms control when and how pages are transferred between the two). Configure the size of the swap device: remaining is local

fault occurs: swapped into remaining part in the machine

measure: job completion time as compared to zero-delay case

results: relative performance degradations over different network configurations: delay is an artificial parameter may result from network congestion by disaggregation

~125GB data equally distributed across an Amazon EC2 cluster 5 m3.2xlarge servers.

Access links not a bottle neck

Batch application in a cluster with 5 workers and 1 master. Application multi-threaded, same as threads as cores.

2 categories: network latency and bandwidth needed to achieve a low performance latency

network latency: 3 us to 5 us

estimate:

ssh jh2635@ath-1.ece.cornell.edu

1. One way path between servers in different racks across three switches(2 ToR and 1 fabric switch)
2. Inter-rack distances of 40m and intra-rack distances of 4m with a propagation speed of 5ns/m
3. Cut-through switches

Round trip latency: software overheads, moving the page to/from the NIC at both the sending and receiving endpoints, 6 switches traversals, 4 link traversals , page of 4KB page

3 unavoidable delay: 1. Data transmission time, 2. Propagation delay 3. Switching delay

* 1. us latency

RDMA integrated NICs optimizations effectively reduce latencies.

Ignore queuing delay on end-to-end latency and hence application performance. Remedy:

Application-driven input traffic workload.

Transport protocols for DDC

Network access trace using tcpdump, a disk access trace using blktrace

Splitting each node into one compute, one memory, and one disk blade and assigning memory blades to virtual nodes

All memory and disk accesses are associated with address in corresponding CPU’s global virtual address space.

Match network and disk traces across the cluster: disk access request matches a local flow in tcpdump traces -> assume from remote read, others local CPU

FCT: network layer performance,

Injecting latencies for page misses.

However: inject the flow completion times obtained from best-performing network design instead of constant latencies.

Kernel space RDMA block device driver which serves as a swap device:

Sev

Swap device,

Set up interactive latency critical application

Mango DB, cache in memory, doesn’t fit in memory -> disk (docker container)

1 machine client: 1 server (has Mango DB) workload generator

load as high as needed

tcp dump server side (timestamp, source destination, header payload) latency: memory or disk

not disk but remote memory

? rerun application(swap device) or replay trace?

Access to cluster using

2 machines, 1 server 1 client

change in home directory run home directory

sudo access

tcpdmp probably run with sudo

ath 1(submit request, cli) ath 2(ser): 2 servers in 1 rack

tell memory through docker container, docker container-

Zibian: web search: index in memory, data file in disk