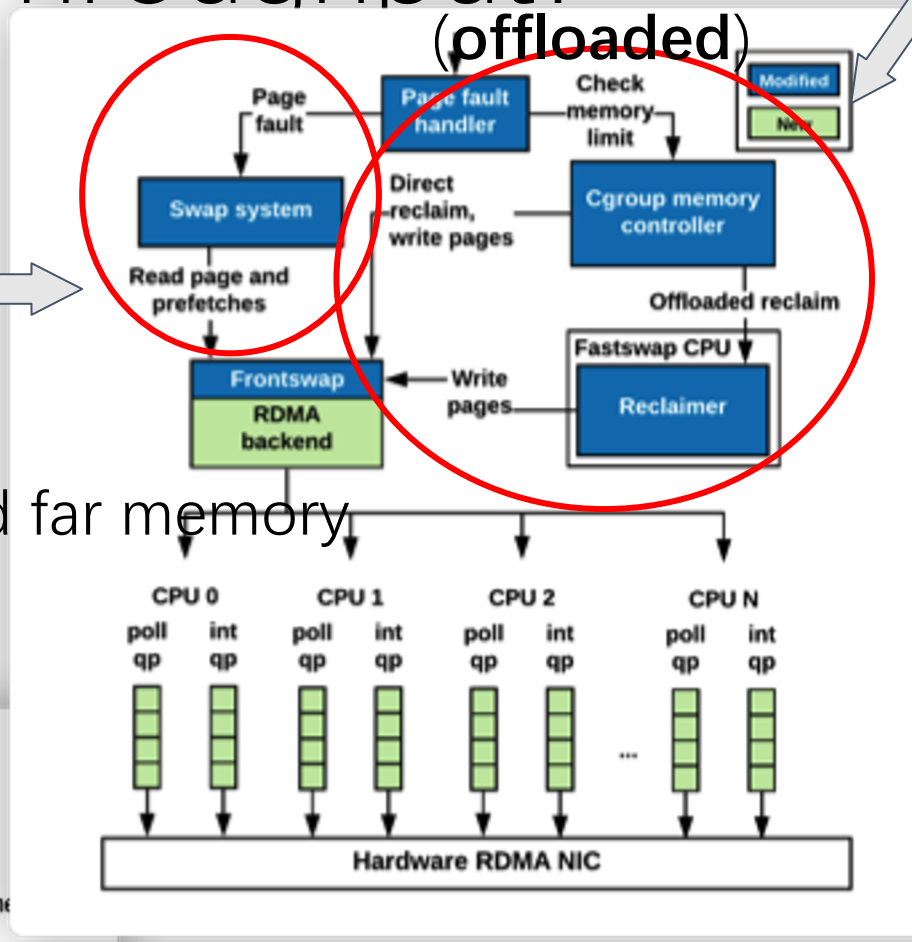
Can Far Memory Improve Job Throughput?

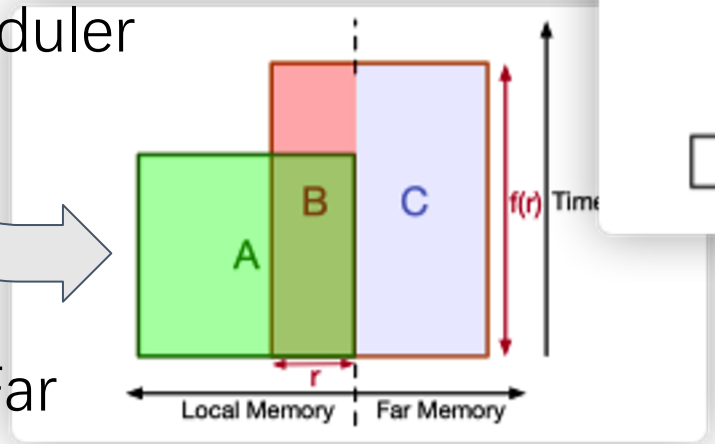
Can Far Memory Improve Job Throughput?

- UC Berkeley + VMware Research
- Motivation
  - Far Memory + Cluster => ?
- Challenge
  - enabling far memory to be swapped in quickly
  - deciding how to schedule jobs across local and far memory
- Contribution
  - Fast Swapping
  - Far Memory-Aware Scheduler

Reclaim Far Memory (**offloaded**)

Alloc and **Prefetch** Far Memory





Before: **A** use Local Mem Totally

Later: **B+C** use Local Mem and Far Mem

How to use Far Memory?

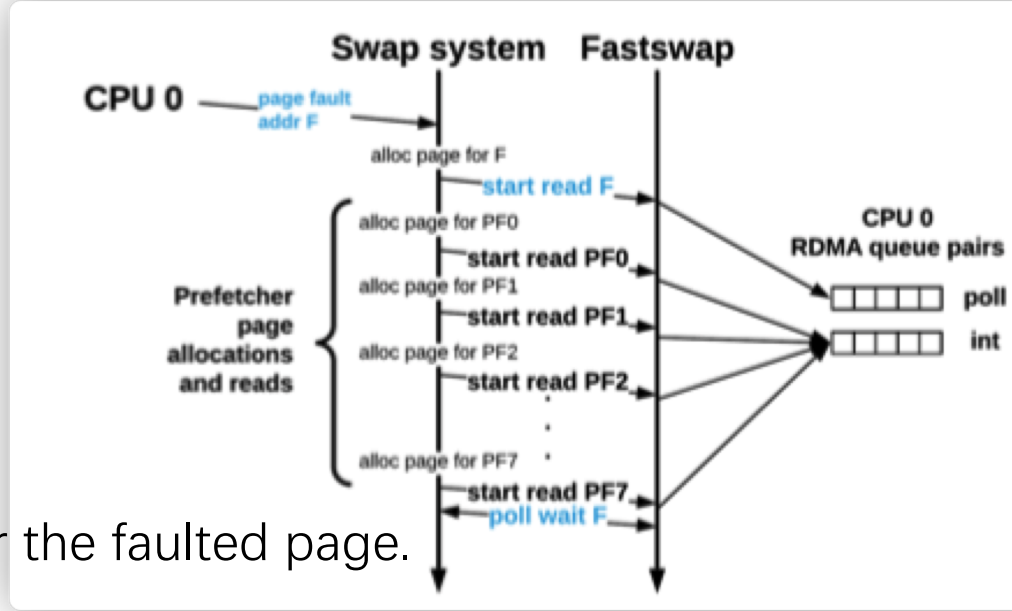
- Swapping (happens after page fault) [1]
  - Traditionally data swaps between **memory** and **disk** => latency is **millisecond-scale**.
  - data swaps between **local memory** and **far memory** => latency is **microsecond-scale**.
- Cgroups (Linux control groups)
  - Cgroups control the amount of physical memory allocated to a group of processes
  - CFM uses the swap system to keep the excess in far memory
- RDMA
  - CFM leverages RDMA API for swapping pages over the network.

[1] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. 2017. Efficient Memory Disaggregation with INFINISWAP. In Symposium on Networked Systems Design and Implementation (NSDI' 17). 649–667.

Fastswap

- Queue pairs
  - Why not use shared queue pair?
    - **critical operations** may queue behind **less urgent prefetch reads**.
  - critical operations => **poll** qp
  - prefetch => **interrupt** qp
- Frontswap
  - designed for swapping at **page granularity** rather than supporting general **block I/O** operations
  - **native Frontswap cannot distinguish** between critical and non-critical operations
  - Modified Frontswap:
    - **critical** path operations => **poll** for completion
    - **non-critical** path operations => **trigger interrupts** on completion

Fastswap

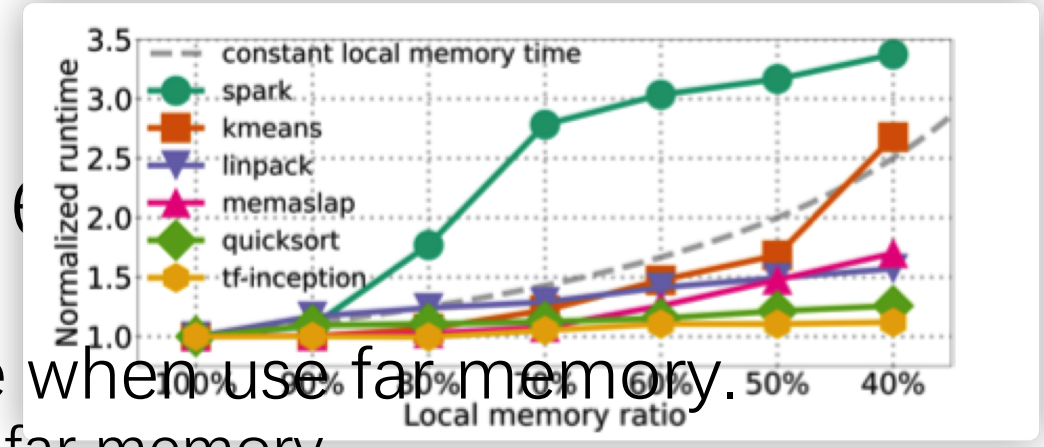


- Page Fault Handler
  - **OVERLAP** to hide latency
    - allocating physical pages for the faulted reads
    - prefetch RDMA reads, with the RDMA read for the faulted page.
- Memory Reclaim
  - When we use memory reclaim?
    - Traditionally, after reading a faulted page, the memory controller charges the page to its cgroup.
    - If there are excess pages, they are **directly reclaimed** and possibly evicted to far memory. (In Linux, memory reclaim is expensive.)
  - How to reduce memory reclaim overhead?
    - Use **offloaded reclaim**
    - but not panacea, e.g., **large memory allocations** or **large limit shrink**.
    - may become a bottleneck because the reclaimer is **shared across CPUs**

Workloads	% of ker
linpack	
quicksort	
kmeans	
tensorflow-inception	
spark	

Table 3. Fraction of kern
ory reclamation without th
percentage reduction when t

Far Memory-Aware Schedule

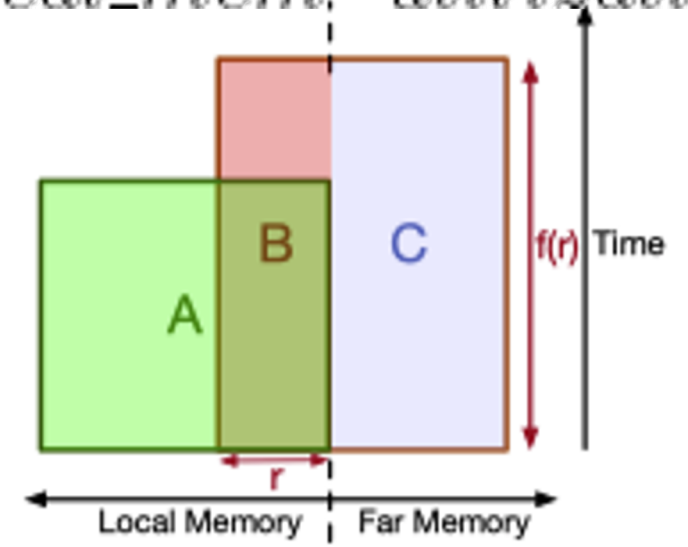


- Some apps have different performance when use far memory.
  - spark’ s runtime triples when using 40% far memory
  - tensorflow-inception experience little slow down when using far memory
- Far memory scheduling policies
  - Uniform Policy
    - shrink all jobs **uniformly** up to a minimum ratio a.
    - a=0.75, every job use 75% local mem + 25% far mem
  - Variable policy (**linear slowdown**)
    - **minimum ratios** = the memory ratio of jobs geting a **20%** slowdown (empirical value)
    - reduce local memory propotionally for each job according to its minimum ratio.
  - Memory-time policy (**non-linear slowdown**)
    - memorytime = the sum of all the jobs’ memory requirements multiplied by their runtime
    - local_mem = the total available local memory in the cluster
    - utilization = the average utilization of local memory in the cluster

$$makespan \approx \frac{memorytime}{local\_mem \cdot utilization}$$

Memory-time policy

$$makespan \approx \frac{memorytime}{local\_mem \cdot utilization}$$



- Target: reduce **makespan**
  - decrease memorytime
  - increase local_mem
  - increase utilization
- Without far memory
  - memorytime and local_mem are always fixed.
- With far memory
  - decrease local memorytime and increase utilization to further improve makespan
  - Since we use far memory only when local memory is fully utilized, utilization is commonly very high.

memorytime $\sum_{i=1}^{N} mem_i \cdot f_i(1) \longrightarrow \sum_{i=1}^{N} mem_i \cdot r_i \cdot f_i(r_i)$

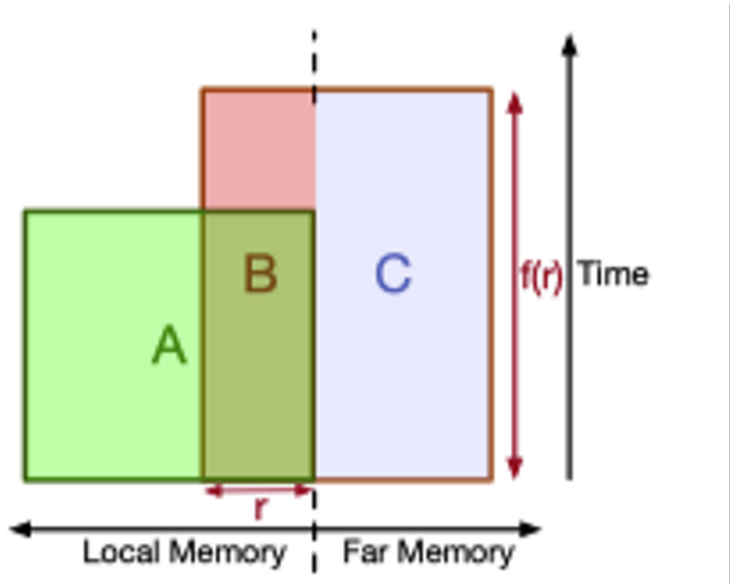
How to schedule in memory-time policy?

$$\underset{r_i : i=1,\ldots,N}{\text{maximize}} \quad \frac{A-B}{C}$$

$$\text{subject to} \quad \sum_{i=1}^{N} mem_i \cdot r_i = local\_mem$$

$$A-B = \sum_{i=1}^{N} mem_i \cdot (1-p_i) \cdot f_i(1) - mem_i \cdot (1-p_i) \cdot r_i \cdot f_i(r_i)$$

$$C = \sum_{i=1}^{N} mem_i \cdot (1-p_i) \cdot (1-r_i) \cdot f_i(r_i)$$



p_i = a ratio between 0 and 1 that represents the progress of this job according to its profile.

$(1-p_i) \cdot f_i(r_i)$ r_i

= the remaining run time for job i using local memory ratio

Evaluation-1

m2c	NoFar	Far (+0%)	Far (+11%)	Far (+33%)
1.0	1.00	1.05	1.04	1.07
1.2	1.00	1.12	1.08	1.10
1.4	1.00	1.07	1.12	1.11
1.6	1.00	1.15	1.21	1.28

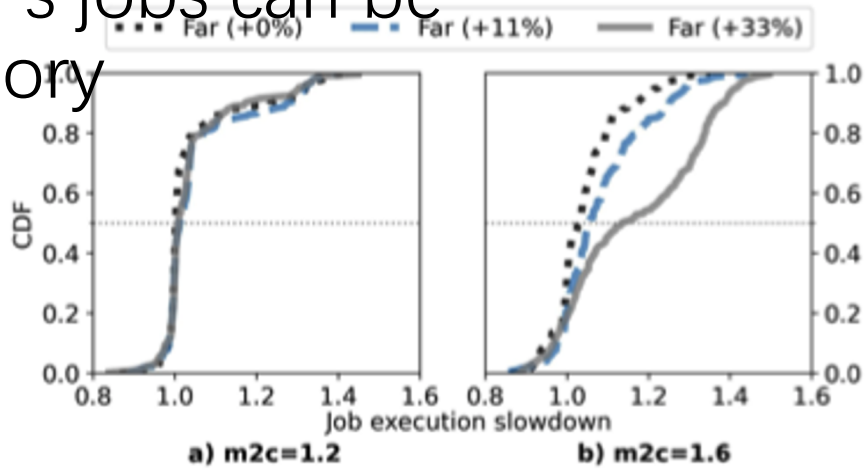
- use **m2c** and **packability** to characterize workloads.

$$m2c(W,C) = \frac{\sum_{j=1}^{N} mem_j \cdot duration_j}{\sum_{j=1}^{N} cpu_j \cdot duration_j} \cdot \frac{C_{cpu}}{C_{mem}}$$

memory-constrained

- **m2c** captures how a workload’ s **demand for memory** relative to compute compares to that of the underlying cluster.
- **Packability** captures how easily a workload’ s jobs can be scheduled in a cluster without using far memory
- Use Far(+X%) **improves** cluster throughput
- It also **slow down** the execute time.

Far(+X%) means using far memory and total memory is X% more than no far



Evaluation-2

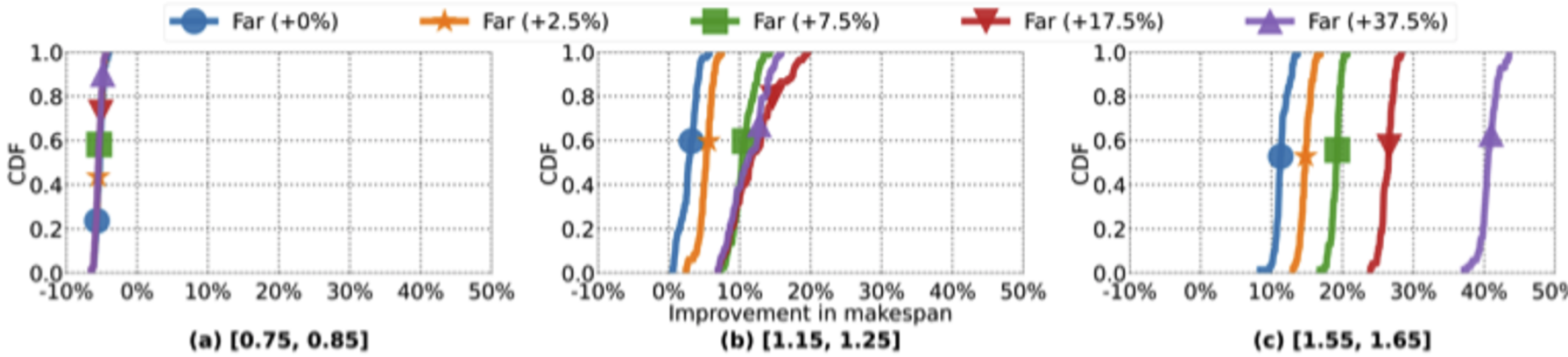


Figure 7. The percent improvement in workload makespan, relative to the NOFAR configuration, for workloads with thre different ratios of memory to compute (m2c).

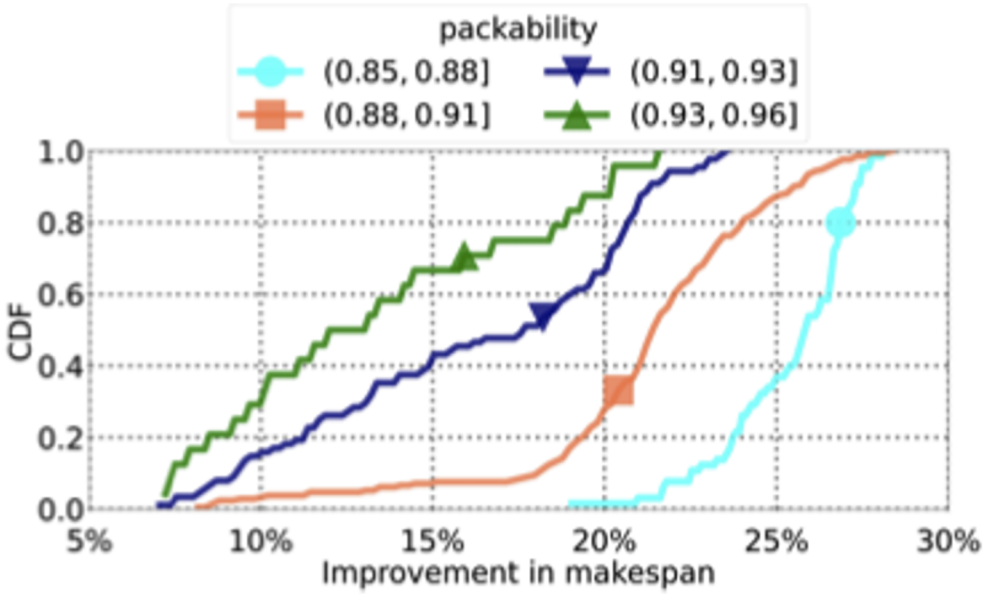


Figure 8. The impact of packability on far memory's ability to improve makespan with FAR (+17.5%) for workloads with $1.15 < m2c \leq 1.65$.

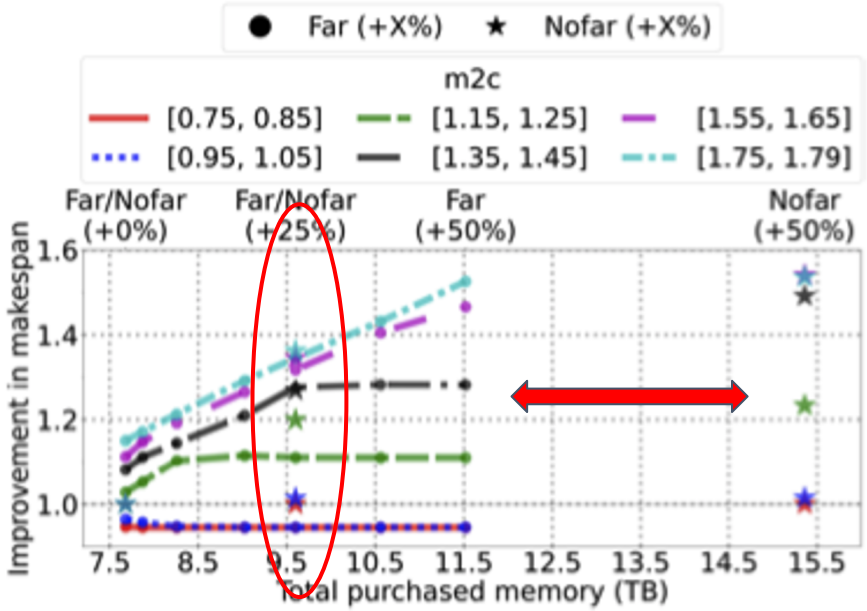


Figure 9. Makespan improvement as we add memory to the cluster in different ways, for workloads with different m2c values. Dots show addition of far memory, while stars are addition of local memory, for a given amount of purchased memory.