Can Far Memory Improve Job Throughput?

Eurosys 2020 Talk

Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo,
Amy Ousterhout, Marcos K. Aguilera (VMware Research),
Aurojit Panda (NYU), Sylvia Ratnasamy, Scott Shenker

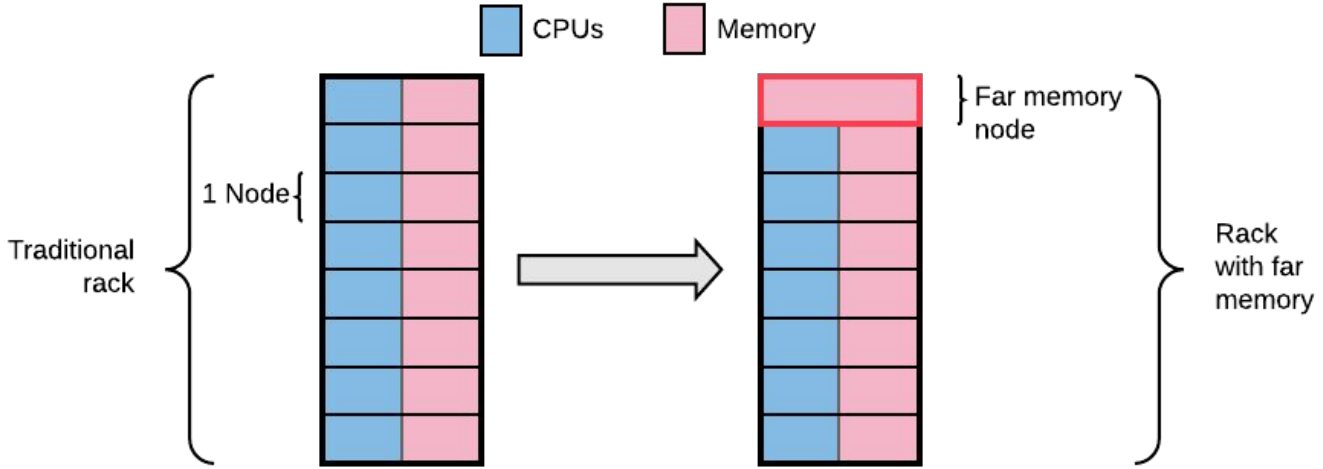
Berkeley
UNIVERSITY OF CALIFORNIA

NETSYS

The rise of far memory

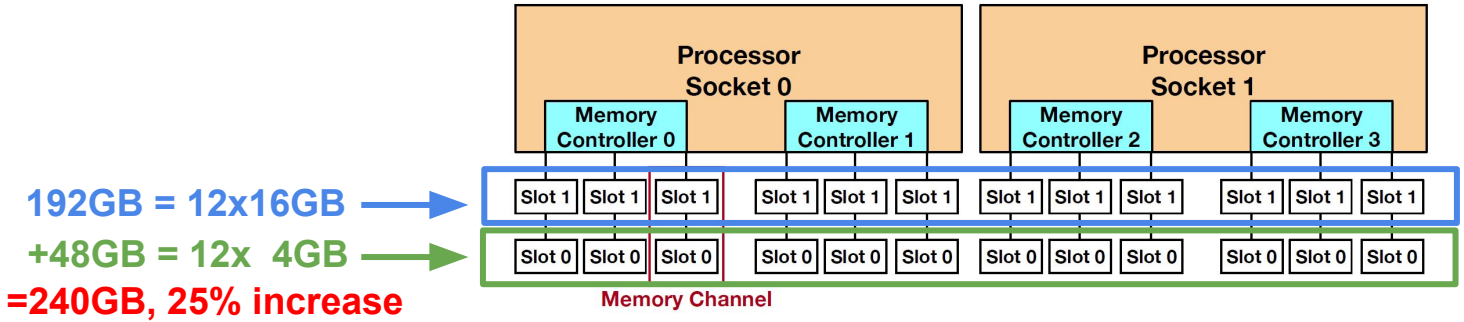
- Demand for memory has grown faster than availability
  - Prevalence of in memory workloads
  - End of Moore's Law hinders DRAM progress

- Far memory allows you to use memory that is remote to the server



Context: Memory provisioning

- Local memory can only be provisioned at coarse granularity



- Unbalanced memory configurations significantly limit memory bandwidth
  - 1 DIMM per controller → 35% of max system bandwidth
  - Balanced configuration: all slots 0 equal capacity and all slots 1 equal capacity

- If we measure the granularity of upgrades in **memory per core** in the cluster:
  - Far memory can be upgraded at much finer granularity than local memory

Focus of our work

1. How to make transparent access to far memory **fast**?

2. How to decide how much far memory **each job** uses?

3. Once we solve 1 and 2, can far memory improve **job throughput**?

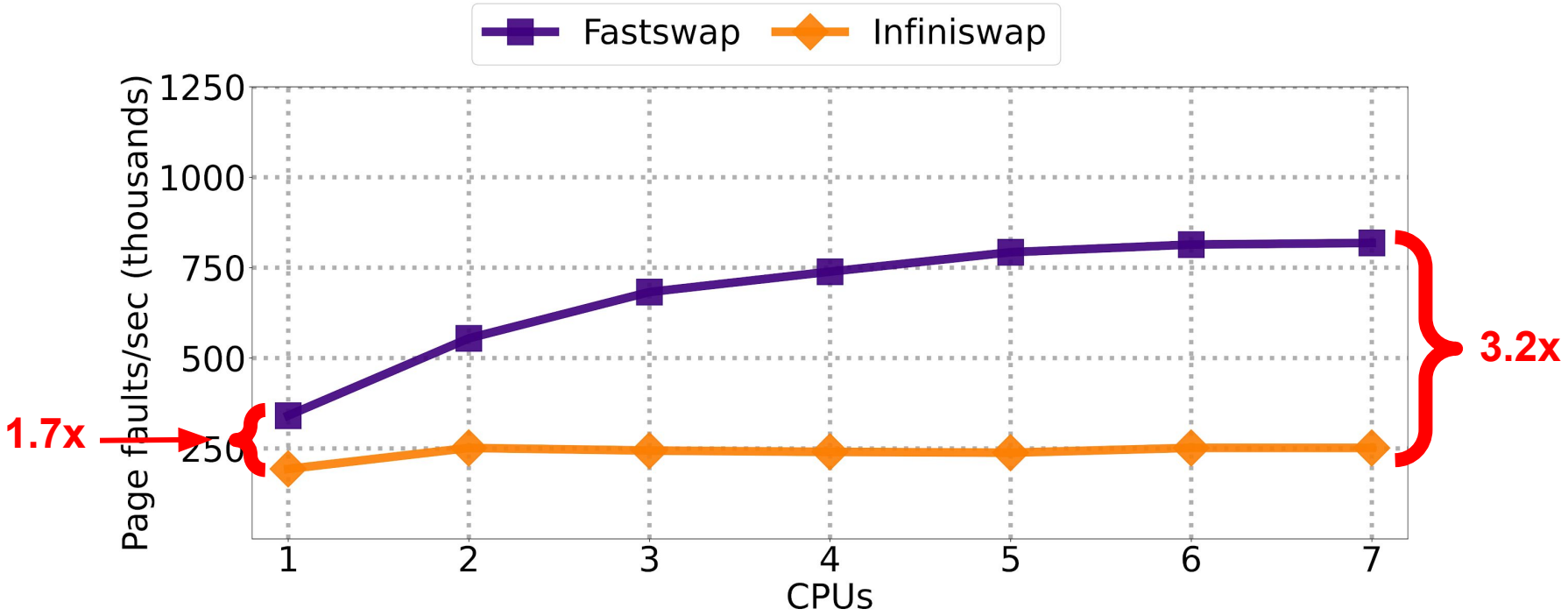
Transparent and fast far memory access

- Operating system support → swapping with RDMA
  - Page fault handler brings pages from far memory into local

- Poor latency and bandwidth in previous systems due to overheads in page fault handler:
  - Head-of-line blocking (high priority reads queued behind low priority reads)
  - Asynchronous critical page reads (require context switch)
  - Page reclamation during page faults

- Fastswap solves key overheads:
  - Average page reads <5us
  - Applications can access far memory at **10Gbps (one thread)**, and **25Gbps (7 threads)**.

Fastswap read throughput



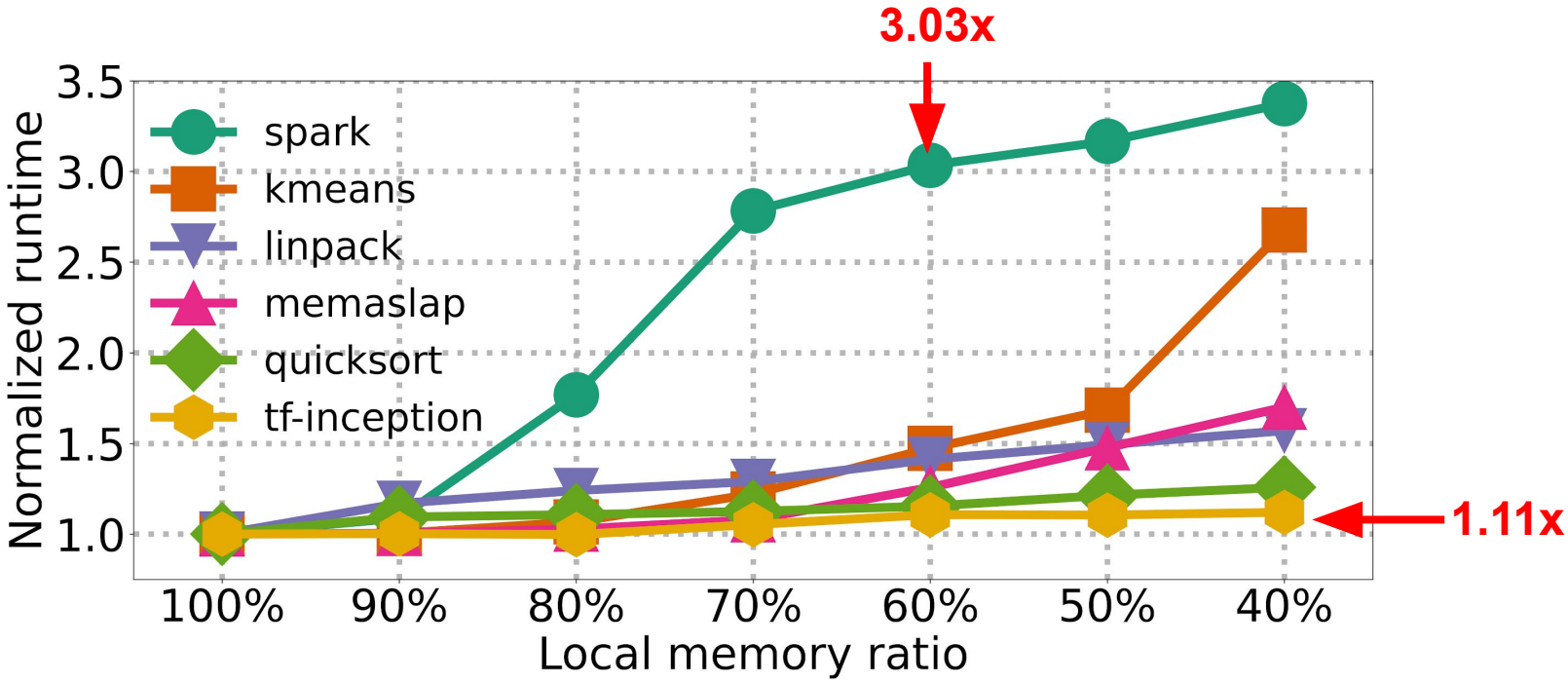
How much far memory each job should use?

- Far memory aware cluster scheduler
  - Improve job throughput
  - Pack the cluster densely by using far memory

- Strategy:
  - If memory **is not** the constraining resource to **admit more jobs** → normal scheduler
  - If memory **becomes** the constraining resource
    - Scheduler **shrinks** local memory on existing jobs ⟶ residuals placed on far memory

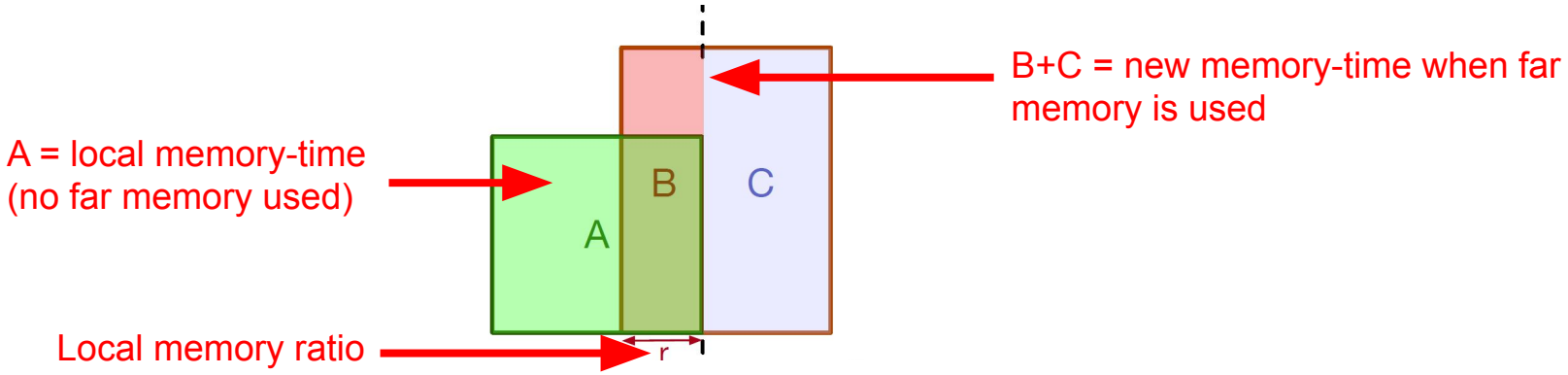
- Key challenge: Performance degradation is application-dependant
  - Scheduler needs to take this into account when shrinking

Job degradation profiles with Fastswap



Memory-time policy

- Uses memory-time products to find optimal shrink ratios for a set of jobs



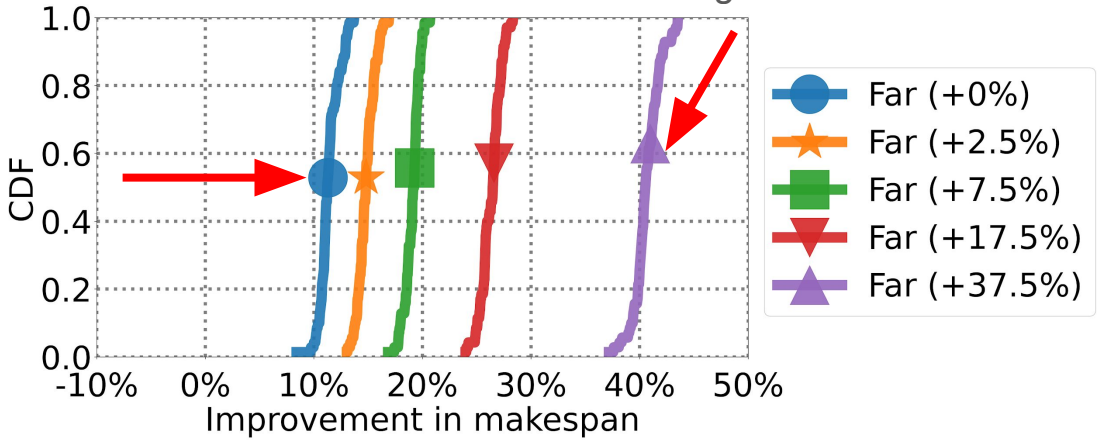
- Optimization problem, intuition:
  - Want to find the ratios r for each job in the set
  - Such that we minimize B (local memory) usage over time

- Optimization runs when a job is admitted, or when a job finishes

Can Far Memory Improve Job Throughput?

- Baseline rack = No far mem, 40 servers (each 192GB and 48 cores)
  - Far (+0%) = convert compute node into far memory node; i.e. 192GB of far mem
  - Far (+X%) = X% additional rack memory available in far memory node

- Workload: a list of 6000 mixed jobs with uniformly random arrivals
  - Each workload is executed in different rack configurations



Conclusion

- How to make transparent access to far memory fast?
  - Fastswap provides transparent, and higher throughput far memory access than previous approaches, by **1.7x on single thread** and **3.2x on multithreaded**
- How to decide how much far memory each job uses?
  - Our far memory aware scheduler decides by using its **memory-time policy**
- Can far memory improve job throughput?
  - Yes, makespan improvements range from **10 to 40%**

Thank you.