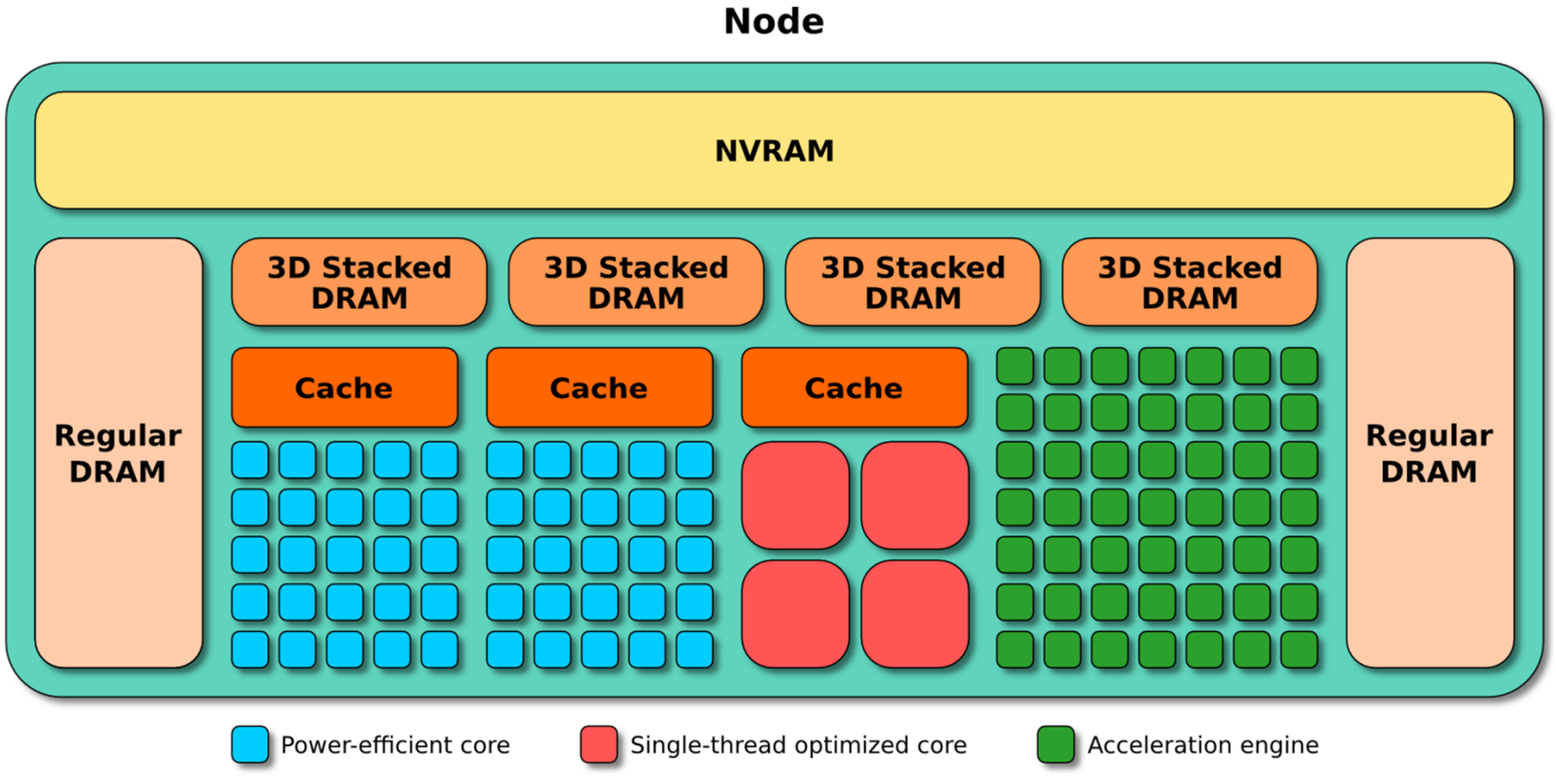

Node OS/R

Node OS/R Team

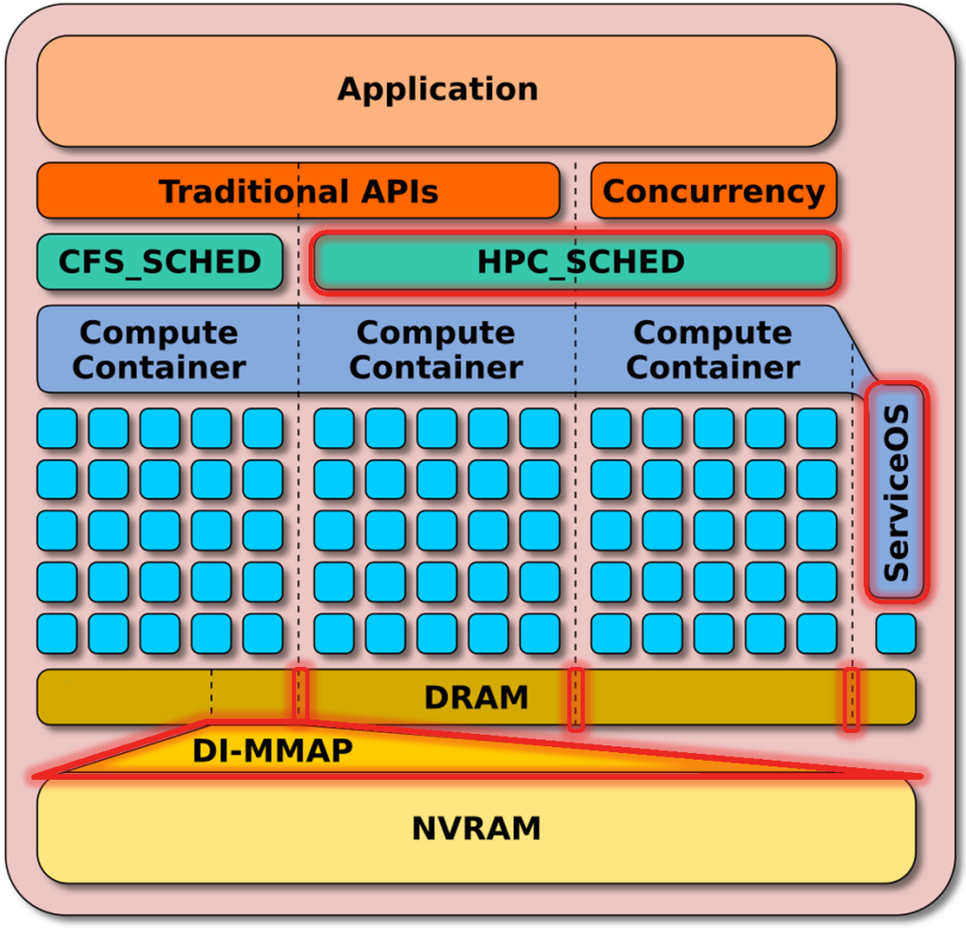
- Argonne: Kamil Iskra, Swann Perarnau, Kazutomo Yoshii, Judicael Zounmevo
- Livermore: Maya Gokhale, Brian Van Essen, Edgar Leon
- Pacific Northwest: Roberto Gioiosa

Exascale Node Architecture



- Heterogeneous compute resources
- Deep memory hierarchy
- Power management
- Fault management

Node OS/R Architecture



- Single kernel image, Linux-based
- Partition hardware resources using containers
  - ease of management, potential scalability improvements
- HPC-specific improvements
  - NVRAM management
  - DRAM management
  - task scheduler

Our key contributions

Highlights

Containers

- Performance isolation of system services, application components
- Co-scheduling of system resources
- Dynamic

FGMN

- Manage physical memory at sub-NUMA granularity
- Greater flexibility
- Reduces memory fragmentation

DI-MMAP

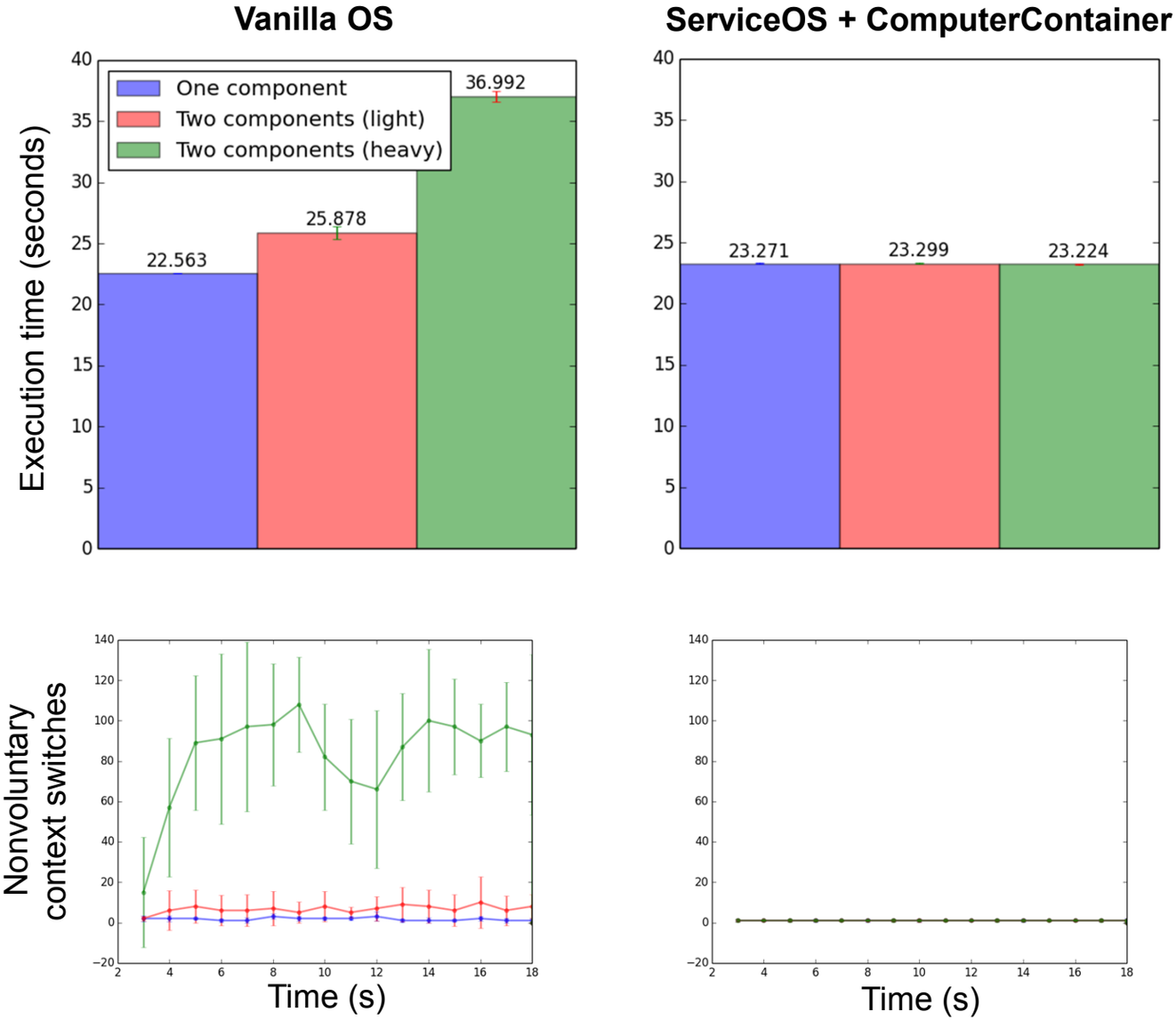
- Integrate NVRAM into HPC node memory hierarchy
- DRAM page cache optimized for HPC applications
- Enable scalable out-of-core data-intensive workloads

HPC_SCHED

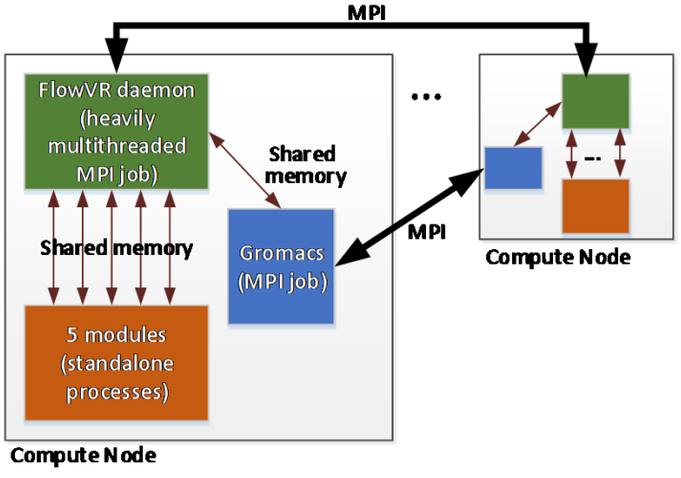
- Reduce task preemption and migration
- Reduce OS jitter, increase process responsiveness

Isolation with Containers

- First component:
  - HSBench (46 threads)
- Second component:
  - CPU load
  - light: 4 threads
  - heavy: 40 threads
- Hardware
  - dual 12-core Haswell (48 hw threads)



Isolation with Containers



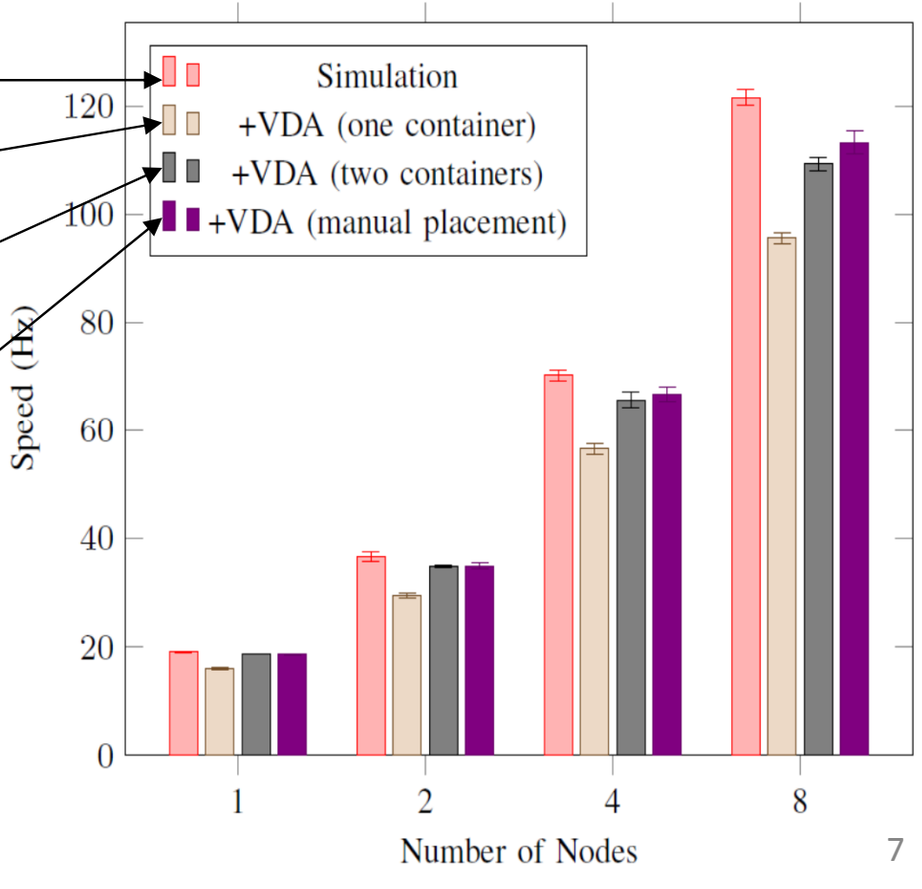
- Simulation + 5-stage VDA pipeline
- Containers improve performance isolation without hindering communication between components

Gromacs only

Gromacs + VDA in 1 container
Separate ServiceOS

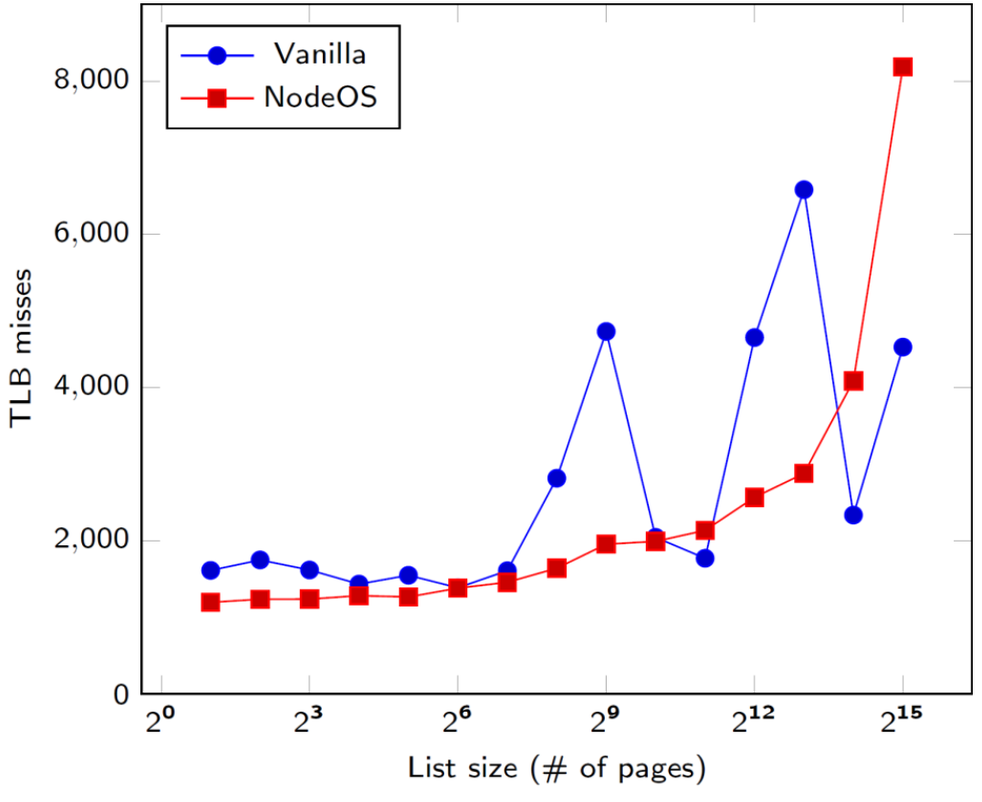
Gromacs in 1 container
Modules in 1 container
FlowVR daemon roams free
Separate ServiceOS

Gromacs + VDA
No NodeOS config.
Manual process placement



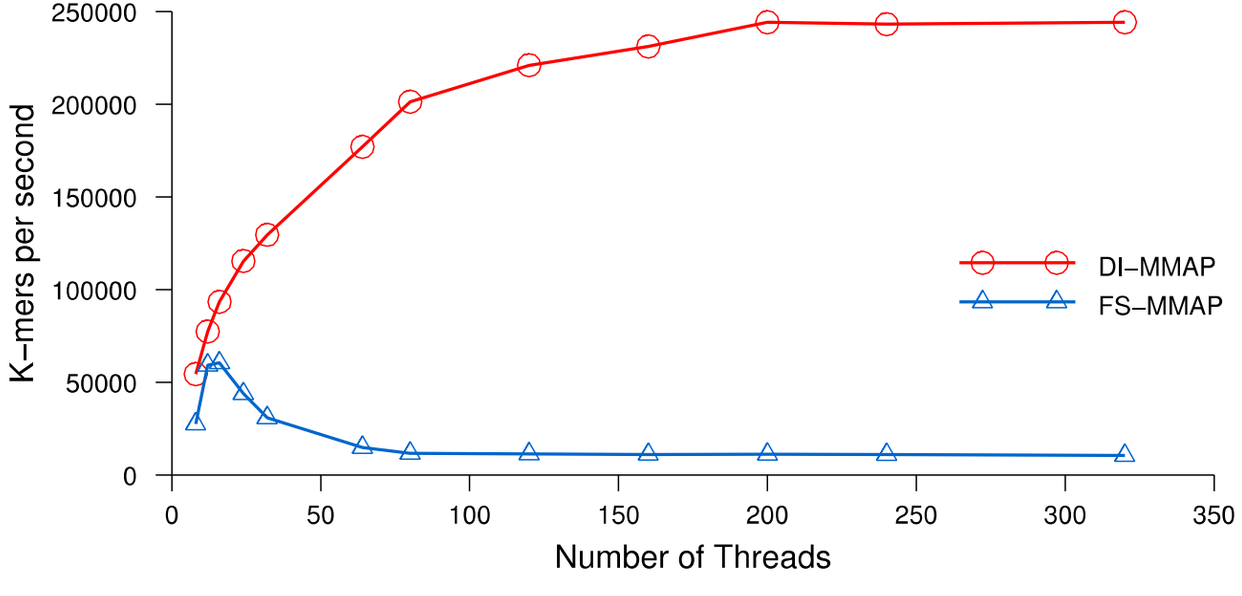
Performance Benefits of FGMN

- Partitioning memory improves isolation, reduces physical memory fragmentation
  - Graph shows how a memory stress workload can affect the performance of latency-sensitive workload (random walk) by preventing the vanilla Linux kernel from using Transparent Huge Pages due to memory fragmentation



DI-MMAP – Impact

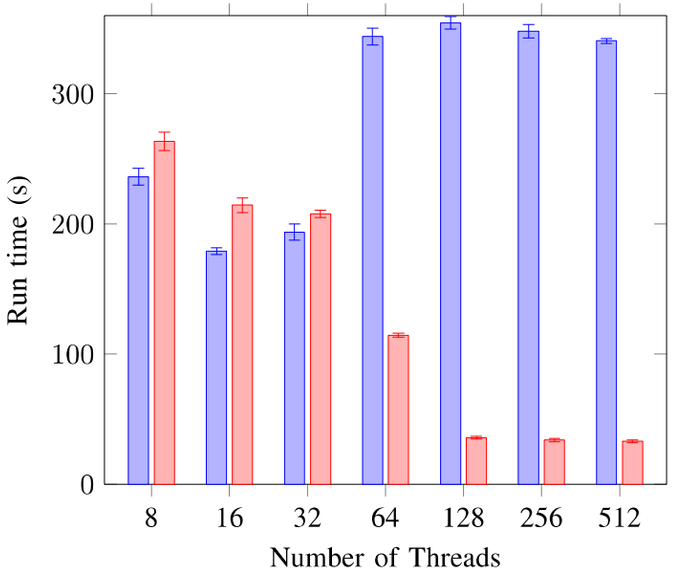
- Significant performance improvements over Linux mmap with out-of-core data intensive workloads
  - 3–4x on Livermore Metagenomics Analysis Toolkit
  - 2.4x on Graph500 Scale 40
- Transparent support eases portability for many existing HPC applications
- **Future:**
  - Multiple caching policies customized to HPC applications
  - Application-tailored prefetch



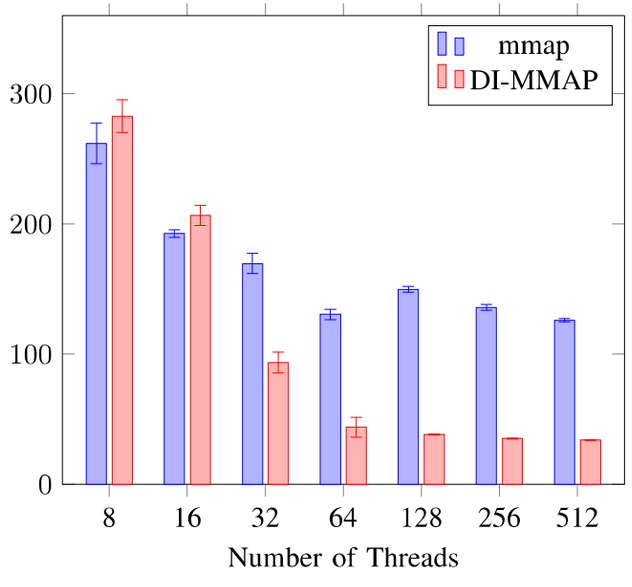
Bioinformatics database query: DI-MMAP vs linux mmap

Integration tests: Affinity vs Containers

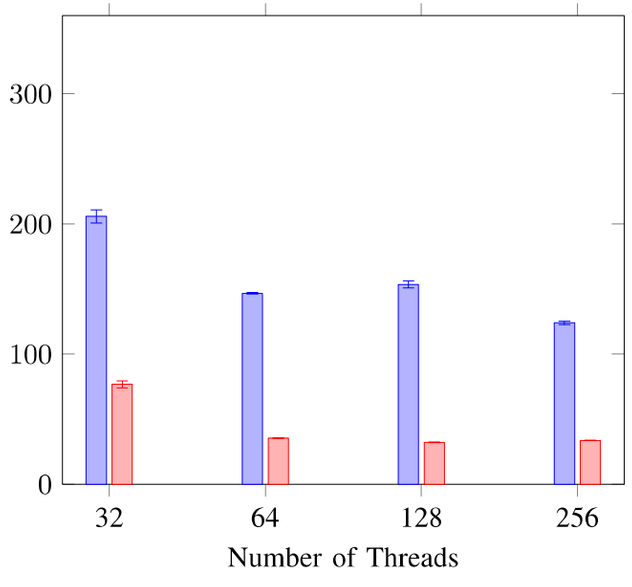
LRIOT run times



(a) Application and buffers unrestricted.



(b) Applications limited to a single socket with `taskset`.



(c) Application and buffers limited to a single socket with containers.

- Containers provide convenient isolation mechanism
  - Improves DI-MMAP performance by up to 20% over uncontained DI-MMAP
- For highly concurrent, threaded applications with read-heavy I/O DI-MMAP is substantially faster than regular Linux mmap

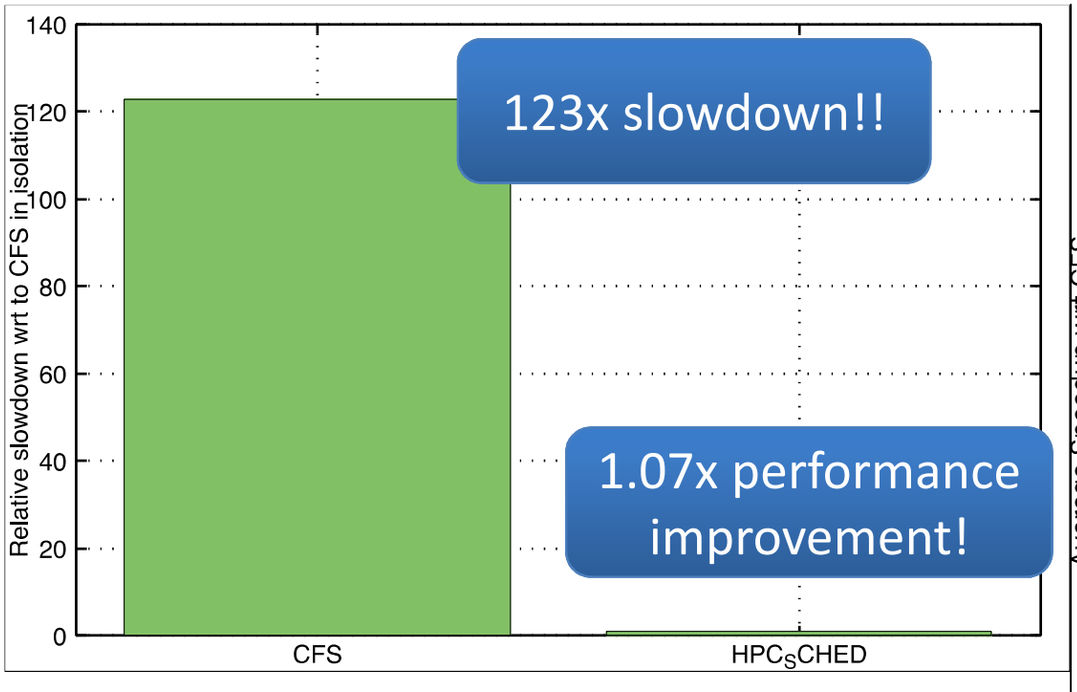
Argo NodeOS Scheduler: Impact

- HPC_SCHED and the noise-canceling w
  within a conta
  - Reduce system noise within a container
  - Allow Argobots runtime to hand-off cores to thr
  - Reduce task preemption and migration
  - Increase responsiveness to events (e.g., NVRAM read requests)

No benefits from CPU affinity

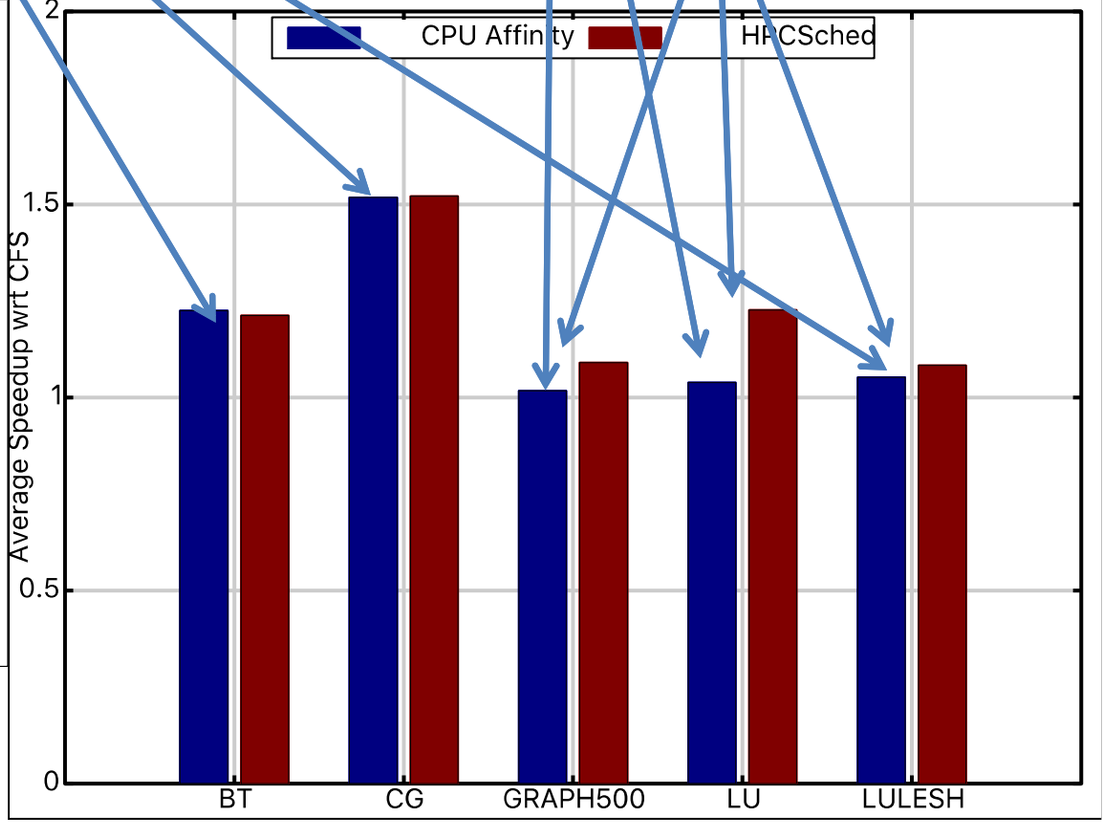
Benefits from CPU affinity

Benefits from CPU affinity **and** responsiveness

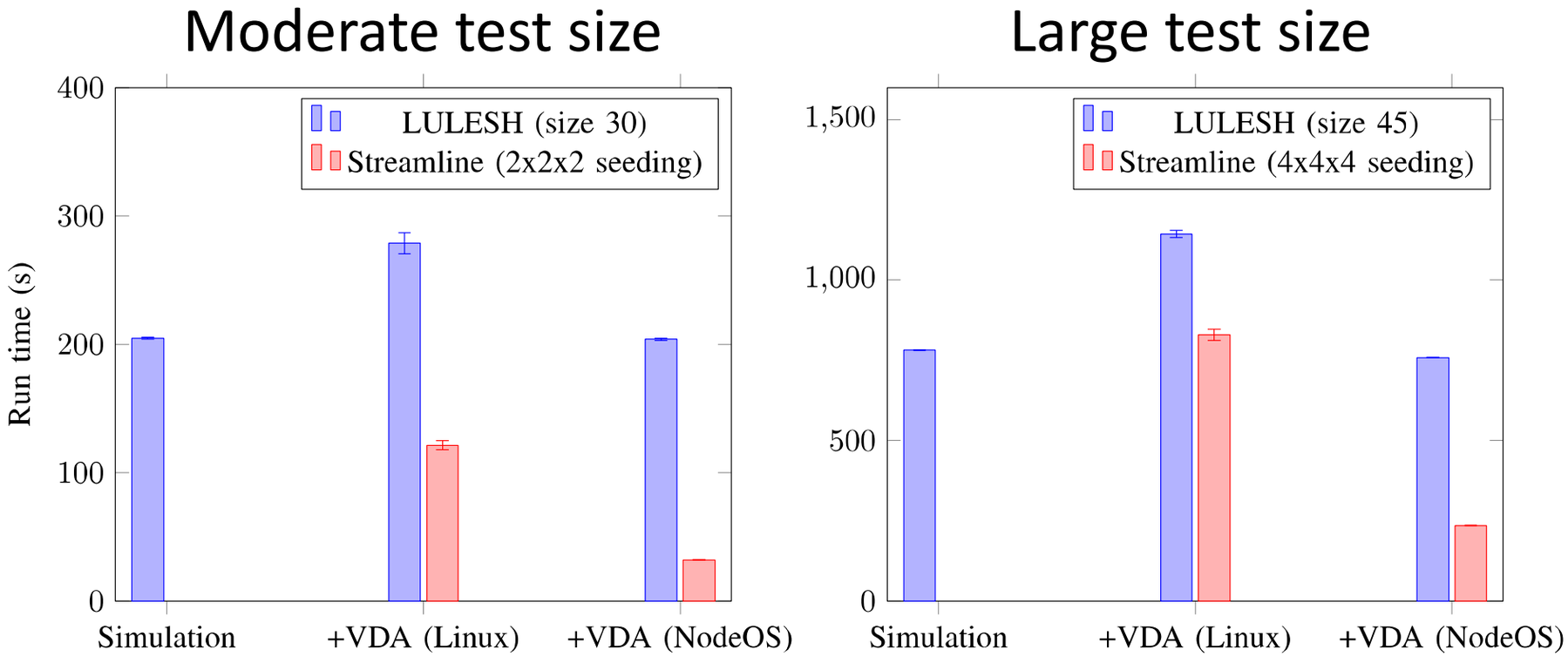


123x slowdown!!

1.07x performance improvement!



Simulation + VDA in-situ experiment



- Containers + DI-MMAP improves overall performance for in-situ experiment
  - 36% to 46% degradation in performance when VDA interferes with LULESH
  - Containers effectively isolate VDA from simulation and improve VDA performance by ~3.5x

Key Results

Containers

- Effective management of node resources in complex scenarios
- Transparent to both applications and system services
- Performance isolation comparable to manual placement

DI-MMAP

- Memory-mapped file I/O to Flash work well with over-decomposed, asynchronous concurrency
- Introspection enables application adaptation and tuning
- Non-native page size support

FGMN

- Effective management of physical memory
- Performance improvements through reduction of memory fragmentation

HPC_SCHED

- Improvements extend beyond process affinity
- Massive improvements over regular scheduler in some cases of oversubscription

Next Steps

- Integrated Node Resource Manager
- Integration with Power management
- Partitioning of NIC
- Support for holding data structures of tasking libraries in NVRAM
- Migration of custom DI-MMAP functionality to user level
- Deep memory management