

Mehmet E Belviranli, Seyong Lee and Jeffrey S Vetter
{belviranlime, lees2, vetter}@ornl.gov

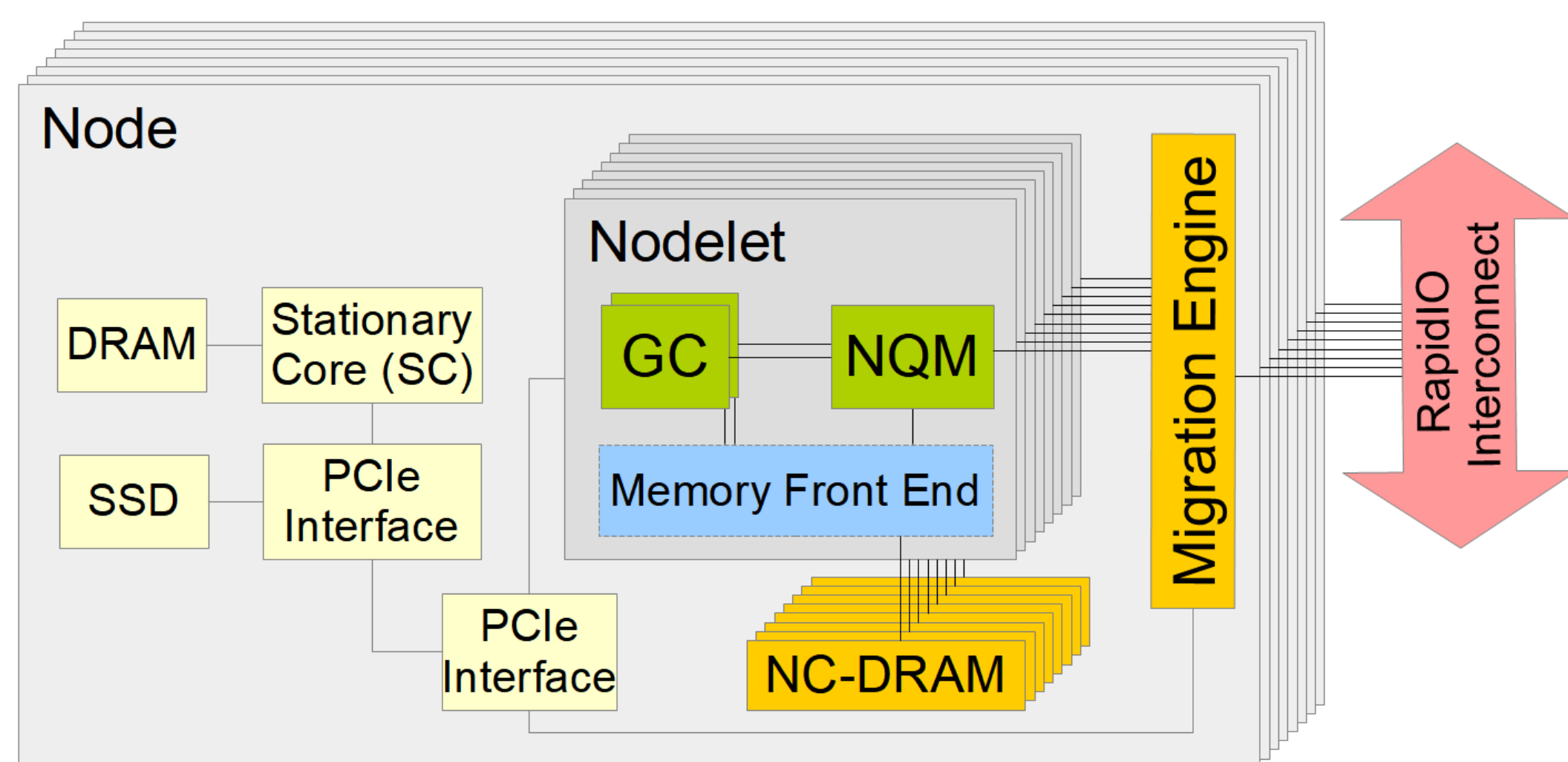
Motivation

- EMU [1] is a novel architecture that attempts to address the weak-locality problem in data-intensive applications by **migrating threads to the location where the data resides**.
- In this study [2], we discuss the **programming considerations that need to be taken into account while designing applications for the EMU architecture**.

EMU Architecture

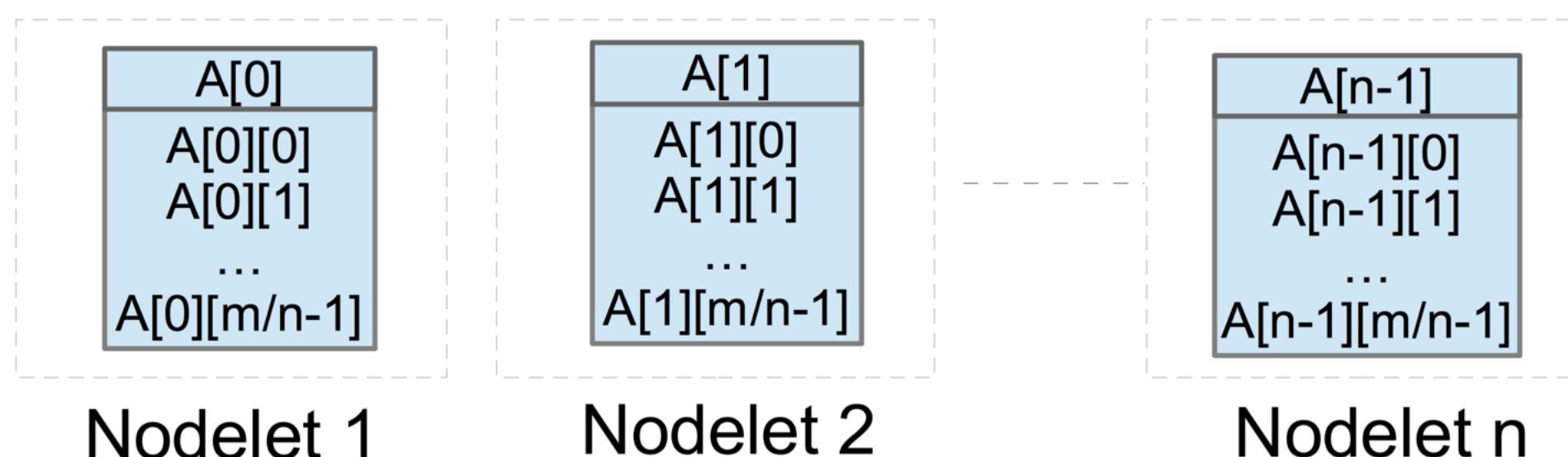
Hardware

- EMU-Chick System:** 8 nodes, 64GB memory per node.
- Node:** Each node has 8x nodelets, an array of DRAMs, a migration engine, PCI-Express interfaces, and a stationary core (SC), accompanied with an SSD.
- Nodelet:** A nodelet contains 2x Gosamer cores (GC), each of which supports 64 concurrent in-order, single-issue hardware threads
- Memory:** Each node has a 64-byte channel DRAM, divided into eight 8-byte narrow-channel-DRAMs (NC-DRAM).



Programming Model

- EMU provides scalable access to a common partitioned global address space (**PGAS**) through a simple programming interface.
- The programming model is based on Cilk and thread migration is automatically performed by the device runtime as the memory accesses occur.
- `cilk_spawn` creates a new thread via a non-blocking call. `cilk_sync` causes the calling (i.e., parent) thread to wait for all spawned children.
- There are three types of memory allocation **intrinsic**s:
 - `mw_localmalloc` allocates a single chunk of memory on the nodelet local to the pointer given.
 - `mw_malloc1dlong` allocates an array, where each element (i.e., long) is striped across all nodelets.
 - `mw_malloc2d` lets the programmer specify the total number of blocks and the size of each block.

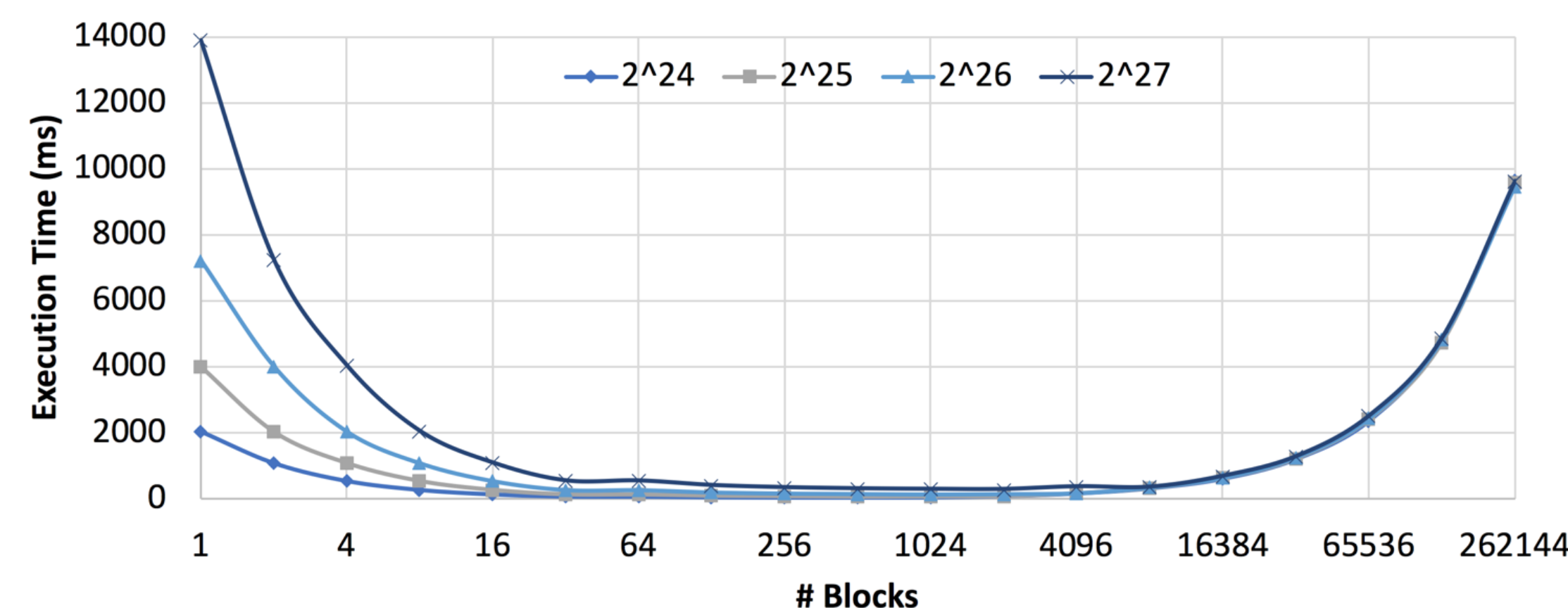


Programming Considerations

Three factors play a major role in performance on EMU systems:

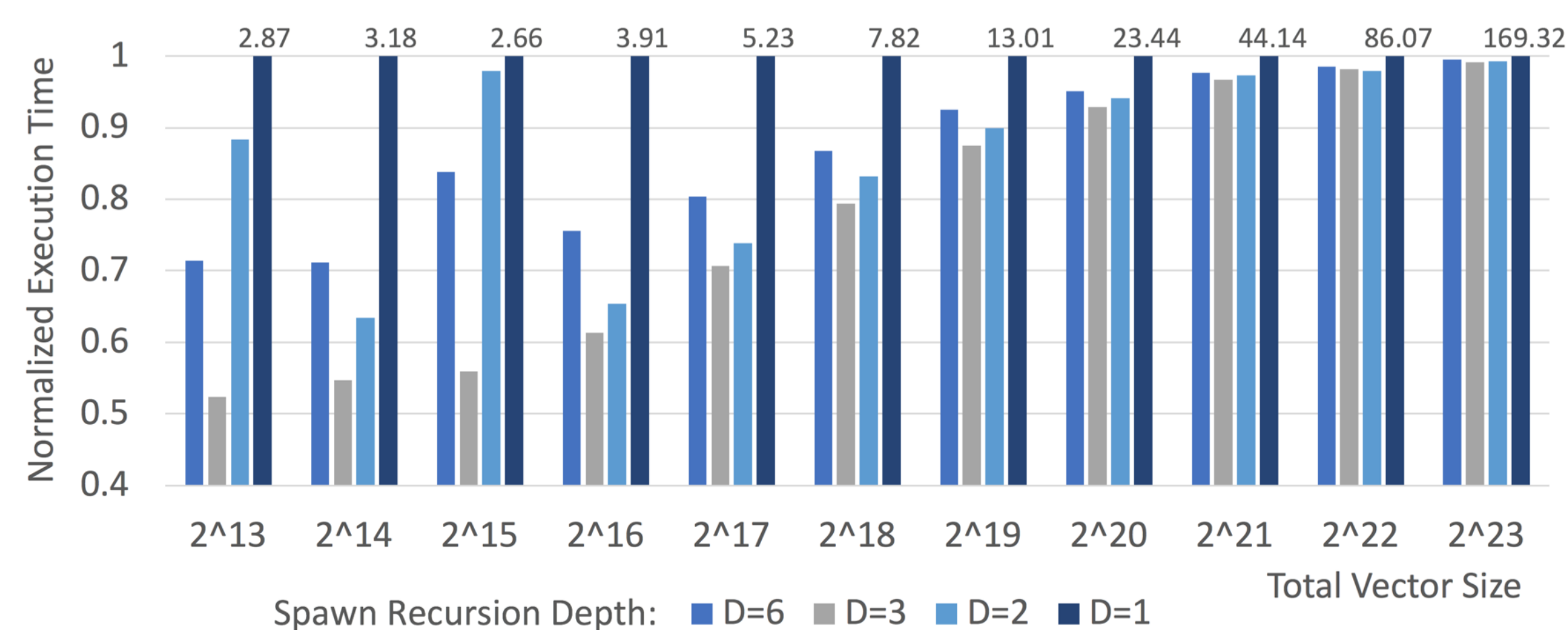
(1) Number of memory blocks

- Using higher number of memory blocks in `mw_malloc2d` will force threads to migrate more frequently across nodelets.
- Allocating fewer blocks will distribute memory into fewer nodelets and will restrict parallelism.



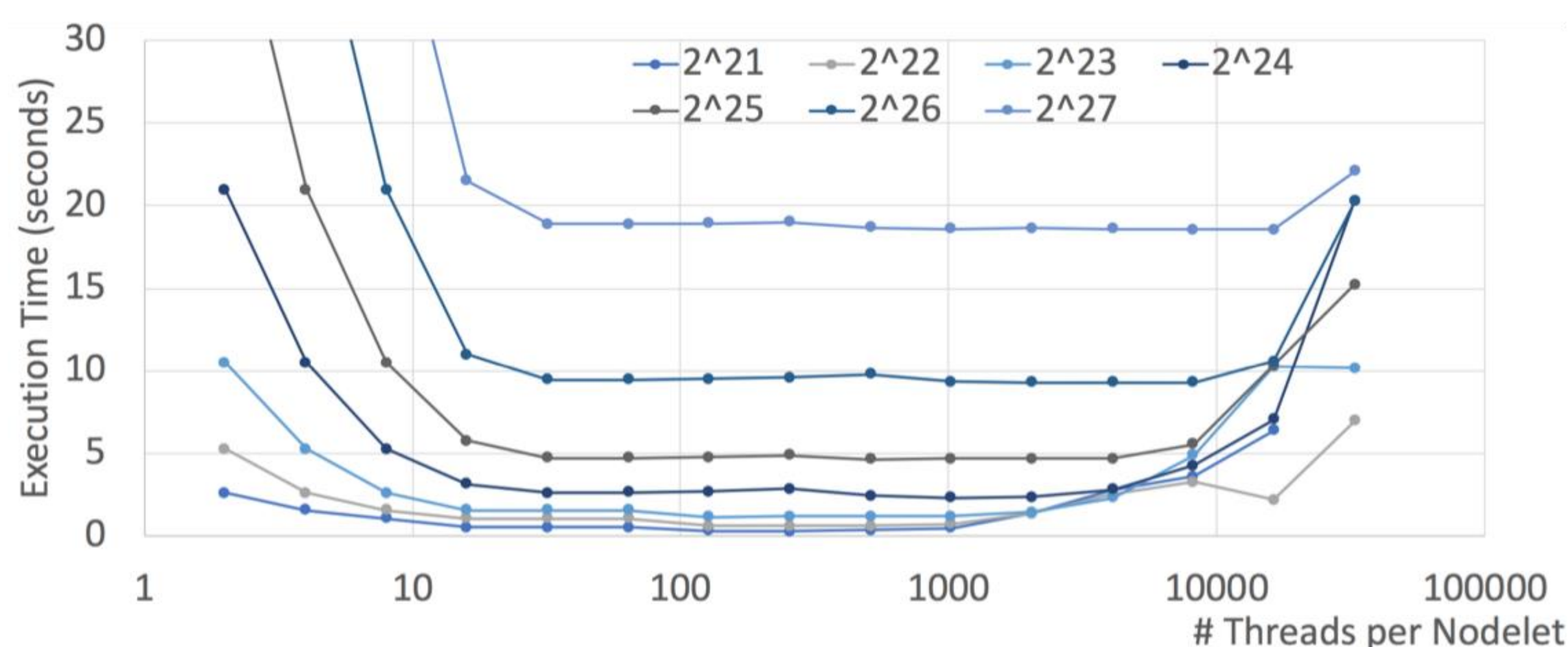
(2) Recursive spawn

- Flat spawn:** The parent thread sequentially iterates over the for loop and spawns children. Thread creation complexity is $O(n)$, where n is the number of threads.
- Tree spawn:** The process of creating children in EMU can be parallelized by using a recursive, hierarchal spawn. Thread creation complexity is $O(\log n)$.

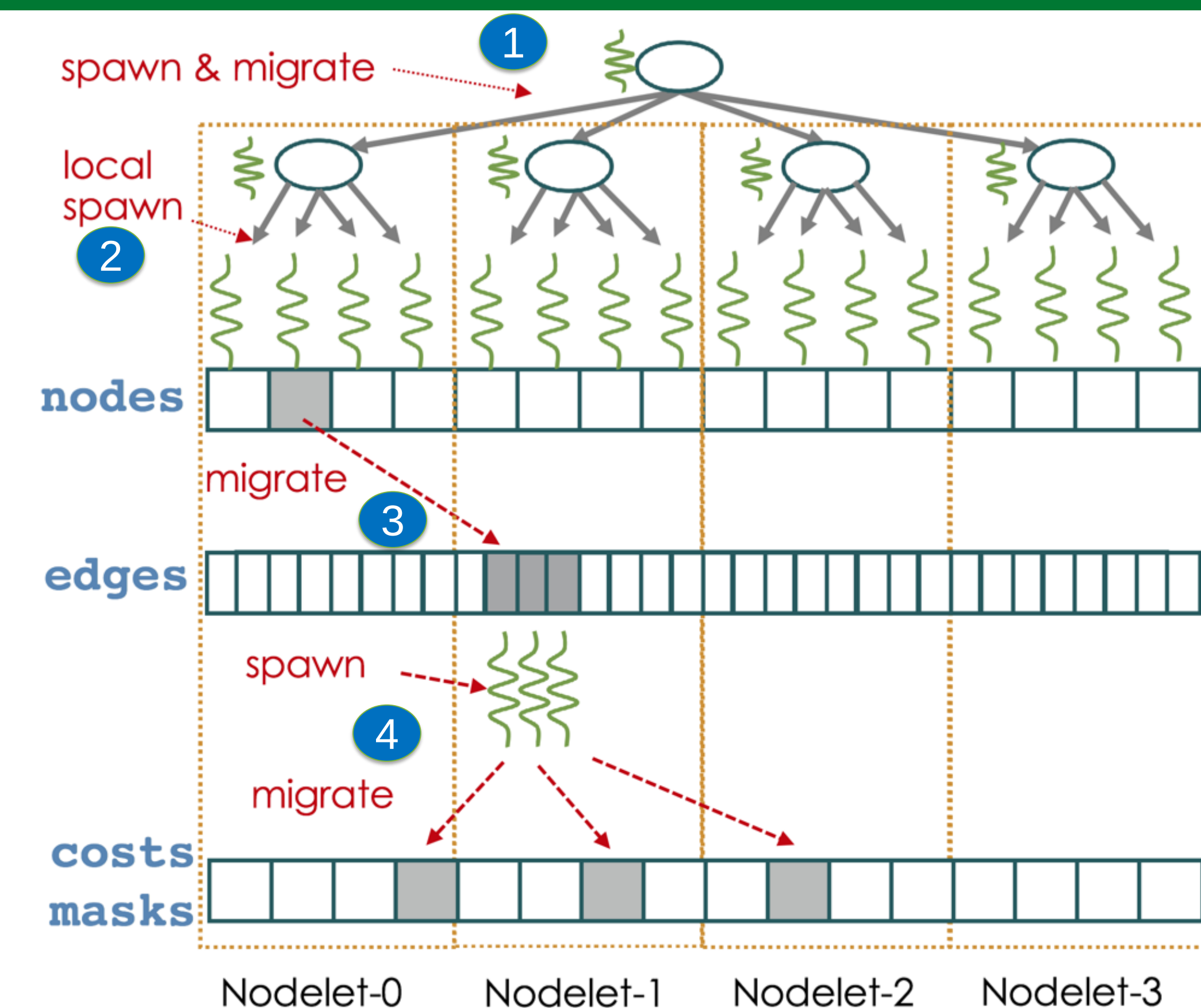


(3) Threads per nodelet

- Each nodelet in EMU-chick can support up to 256 live threads and also can hold context information of up to 500 threads.
- Threads that exceed this level will fail to spawn and execute as a regular function. The optimal number of threads per nodelet varies by the load.



Level-Synchronous BFS for EMU



- We use compressed sparse row representation (CSR) to represent graphs in memory.
- BFS-EMU follows a hierarchical launch strategy
 - First, `visit_node_block` is spawned for every node block, so that further node related operations become local to the spawned children.
 - Then, `visit_node` is spawned for each node in the nodelet. Each thread of this function further migrates to the nodelet where the portion of the edges array holding current node's neighbor indicies resides.
 - Finally, for each neighbor, `visit_neighbor` is spawned. Each thread of this function migrates to the nodelet where the corresponding indicies for the costs and masks arrays are located. They are updated with proper values.

Conclusion

- EMU presents an unorthodox approach to solve the increasingly worsening memory bottleneck problem in HPC.
- There are unique architectural considerations that need to be addressed while developing for the EMU platform.

References

- Kogge Dysart et.al "Highly scalable near memory processing with migrating threads on the emu system architecture." In Proceedings of the Sixth Workshop on Irregular Applications: Architectures and Algorithms (IA3), ACM, 2016
- Mehmet E. Belviranli, Seyong Lee and Jeffrey S. Vetter. "Designing Algorithms for the EMU Migrating-threads-based Architecture." In High Performance Extreme Computing Conference (HPEC), IEEE, 2018

Want to collaborate with us on our EMU system?
Contact us: <https://excl.ornl.gov/contact/>