Is the Heap Manager Important to Many Cores?

Ye Liu, Shinpei Kato, Masato Edahiro

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Overview of the heap manager

- Main items related to our work
  - Explicit function invocations (i.e., malloc() and free()) are used by applications to request memory allocation/deallocation operations from/to the heap manager
  - System calls (i.e., mmap() and munmap()) are invoked by the heap manager to request memory blocks from/to the OS (operating system) whenever necessary
  - Applications expect to allocate/deallocate the memory blocks from/to the heap manager as quickly as possible
  - The allocation/deallocation operations from/to the heap manager have the chance to be on the critical path of the application execution
  - The influence from the heap manager on the program performance is expected to be serious when threads on many cores concurrently allocate/deallocate memory blocks (with various sizes), especially using a lock-based heap manager
  - More importantly, the influence from the heap manager might be associated with the memory management of the OS and the cache system of the tiled many-core processors (i.e., KNL and the TILE-Gx72 processor)

Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Why do we focus on the heap manager on many cores?

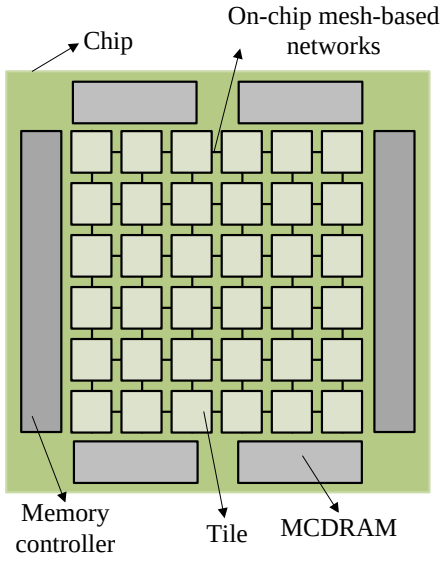
- Solving the scalability problem on many cores is the main topic of our research work
  - It has been proposed that removing the bottleneck from the OS is able to improve the program performance
    - i.e., An Analysis of Linux Scalability to Many Cores
  - It has been proposed that revising the application itself is able to improve the application-level parallelism
    - i.e., Deconstructing the Overhead in Parallel Applications
  - In addition to the OS and application itself, we observed that the program performance could be improved when a scalable heap manager was linked on many cores (i.e., KNL and the TILE-Gx72 processor)
  - Focusing on the heap manager is able to help us further solve the scalability problem on many cores

Outline

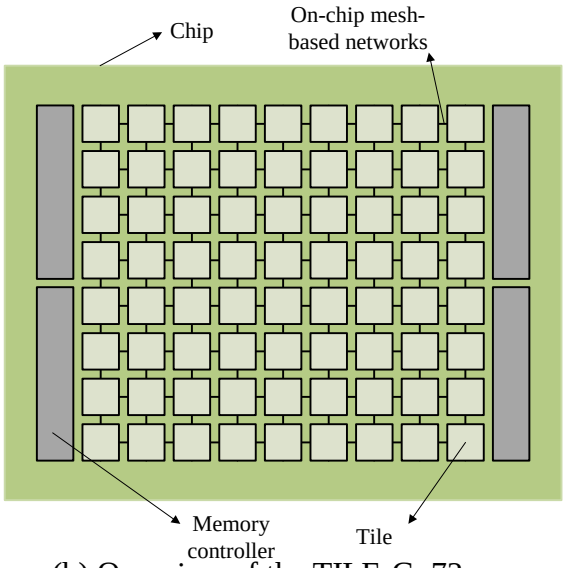
- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Experimental platforms

- Tiled many-core processors (KNL and the TILE-Gx72 processor)
  - Shared-memory system
  - Multiple on-chip memory controllers
  - Processing cores are integrated onto a single chip
  - Processing cores are interconnected via on-chip mesh-based networks
  - The (virtual) last level cache is shared by processing cores



(a) Overview of KNL



(b) Overview of the TILE-Gx72 processor

Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Evaluated heap managers

- Ptmalloc
  - The default heap manager from the GNU C Library on the Linux system
  - The lock is used to protect the data structure named arena
  - Threads must acquire the lock before allocating/deallocating the memory block (with various sizes) from/to the arena
  - The lock on the arena is the main bottleneck of Ptmalloc
- Hoard
  - A scalable heap manager
  - It consists of a global heap and per-processor heaps
  - Superblocks are removed from/to the global heap when the per-processor heap is full/empty based on the design criteria
  - The lock on the global heap is the potential bottleneck of Hoard

Evaluated heap managers

- Jemalloc
  - A scalable heap manager
  - Small memory blocks are allocated/deallocated from/to the data structure named thread cache without locking
  - The lock is used to protect the data structure named arena when huge memory blocks are needed
  - Thread cache of Jemalloc is beneficial to applications with numerous non-huge memory allocation/deallocation operations
- Overview of the evaluated heap managers
  - Drawback
    - They are lock-based heap managers
  - Advantage
    - The evaluated heap managers can be used on both KNL and the TILE-Gx72 processor without considering the limitation of the atomic operations

Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Performance Evaluation

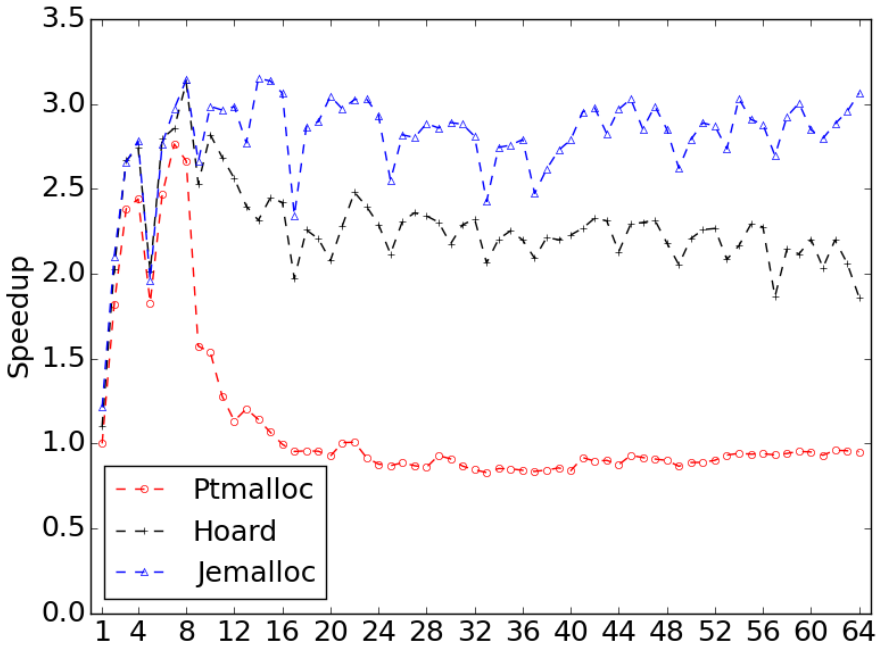
- Overview
  - Applications are from the PARSEC benchmark suite
  - The evaluated heap managers (Ptmalloc, Hoard and Jemalloc) were linked to run the application respectively

Program	KNL	The TILE-Gx72 processor
blackscholes	✘	✘
bodytrack	✔	✘
dedup	✔	✔
facesim	✔	✘
fluidanimate	✔	✘
swaptions	✔	✔

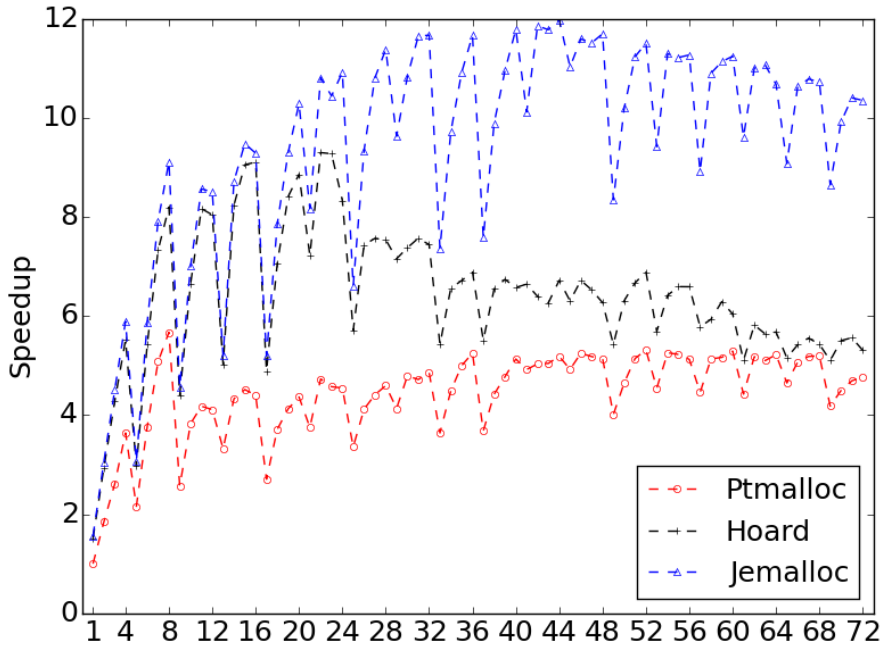
Table: Whether or not the performance variation appears when the heap manager is altered

Performance Evaluation

- dedup
  - A pipeline application
  - It consists of five stages, of which the intermediate three stages are parallel separately
  - X-axis represents the thread count for the parallel phase



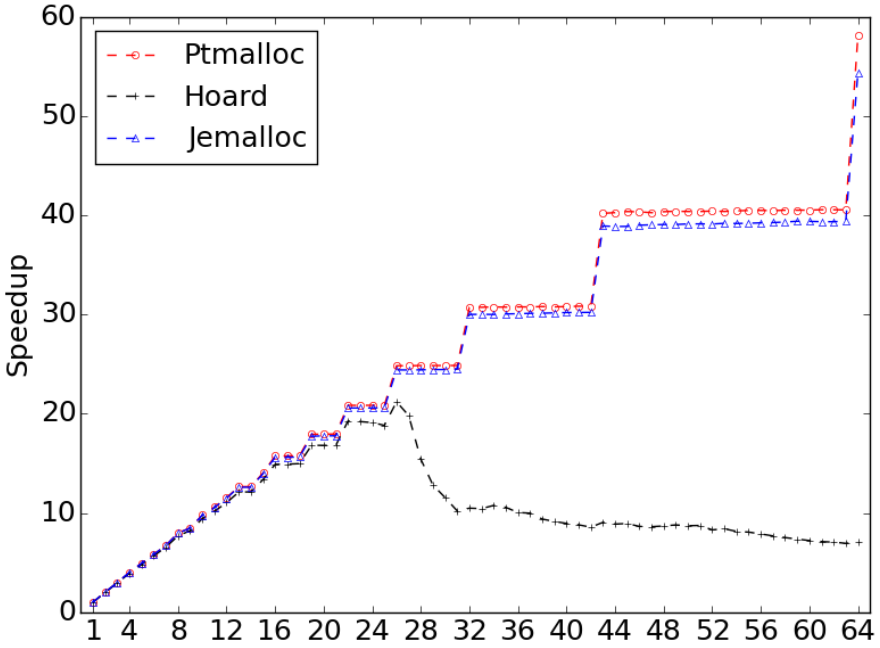
(a) Performance evaluation on KNL



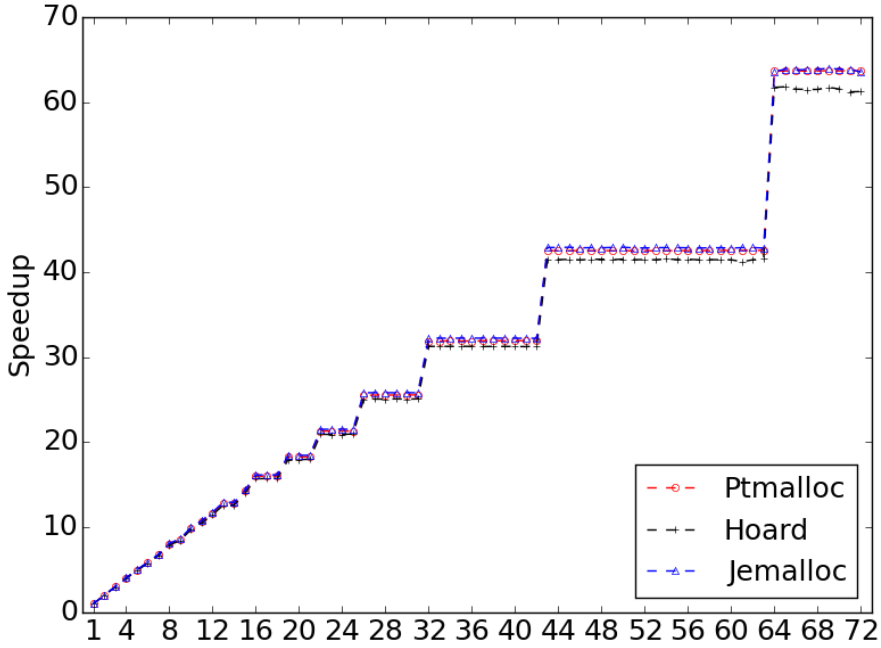
(b) Performance evaluation on the TILE-Gx72 processor

Performance Evaluation

- swaptions
  - A data-parallel application



(a) Performance evaluation on KNL

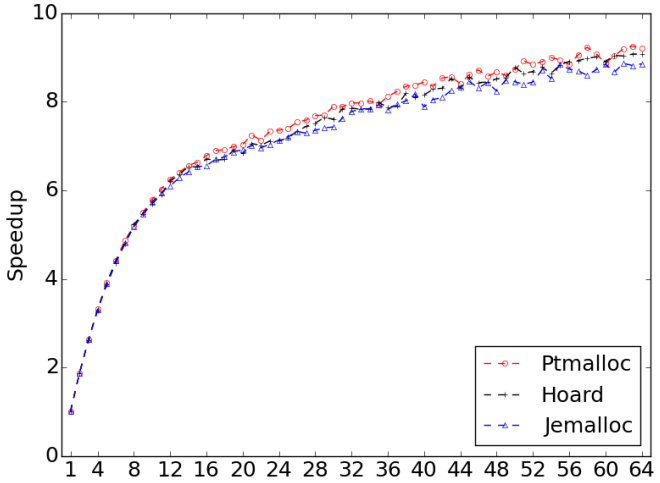


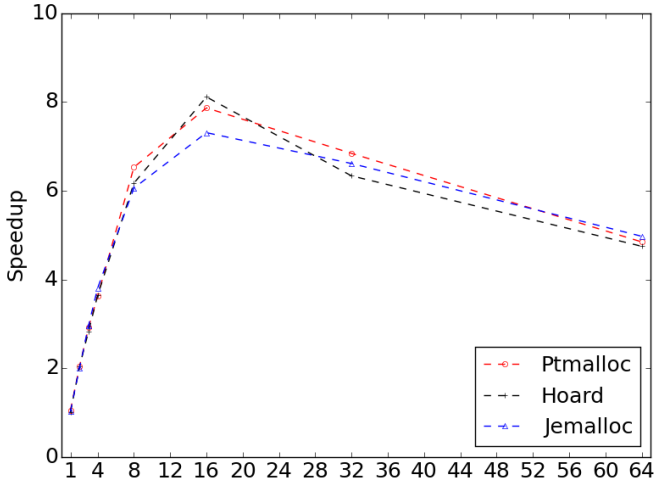
(b) Performance evaluation on the TILE-Gx72 processor

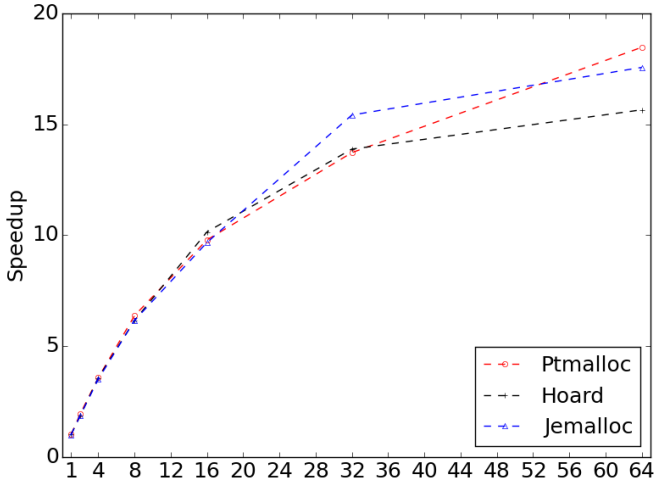
Performance Evaluation

- Other data-parallel applications on KNL
  - bodytrack (upper right)
  - facesim (lower left)
  - fluidanimate (lower right)

  - None of the evaluated heap managers works best for these applications on KNL







Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Concluding Remarks

- Heap manager should be paid attention to as well when analyzing the scalability problem on many cores
  - The performance improvement can be acquired when Jemalloc/Hoard is linked to run the pipeline application (dedup)
  - The performance degradation can be observed when Hoard is linked to run the data-parallel application (swaptions)

- The analysis on the influence from the heap manager should be associated with the memory request patterns of the application
  - It is not easy to analyze how the program performance gets affected by the heap manager when only focusing on the heap manager itself

- The influence from the heap manager is closely related to the experimental platform
  - The performance variation does not appear on the TILE-Gx72 processor but exists on KNL when running bodytrack, facesim and fluidanimate respectively

Outline

- Background
  - Overview of the heap manager
  - Why do we focus on the heap manager on many cores?
  - Experimental platforms
- Heap Manager Design
  - Evaluated heap managers
- Performance Evaluation
- Concluding Remarks
- Future Work

Future Work

- Lock-free (synchronization-free) heap managers will be added to evaluate the program performance
  - i.e., Streamflow, SFMalloc
- Memory request patterns from the application will be analyzed in order to fully understand how the program performance is influenced by the heap manager
- More multithreaded applications designed for the shared-memory system will be added to further evaluate the performance variation caused by the heap manager
- The influence from the memory management of the OS and the cache system of the tiled many-core processors, which can be associated with the heap manager, will be further analyzed

Thanks for your listening!

&

Any questions?