An Analysis of SMP Memory Allocators: MapReduce on Large Shared-Memory Systems

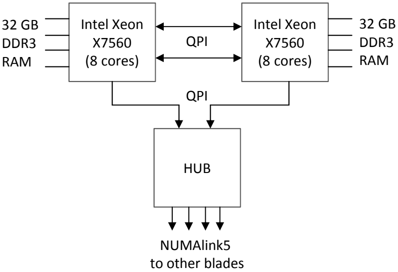
Robert Döbbelin, Thorsten Schütt, Alexander Reinefeld

Zuse Institute Berlin

September 10, 2012

SGI Altix UltraViolett (UV) 1000

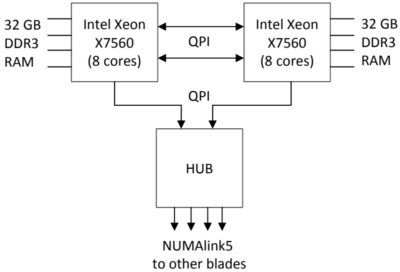
- 32 blades in one rack
- 2×8 cores per blade
- 64 GB memory per blade

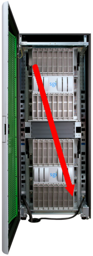


- QPI for memory on same blade
- inter-blade communication via NUMAlink5

SGI Altix UltraViolett (UV) 1000

- 32 blades in one rack
- 2×8 cores per blade
- 64 GB memory per blade





- QPI for memory on same blade
- inter-blade communication via NUMAlink5

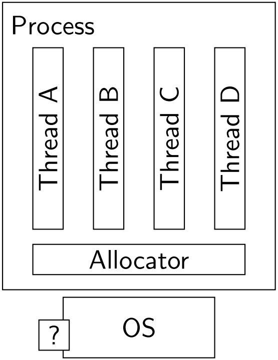
First-touch policy

When a process requests memory from the OS

- threads gets (unmapped) virtual address
- page fault on first touch
- OS allocates physical pages to NUMA node on which accessing thread is running

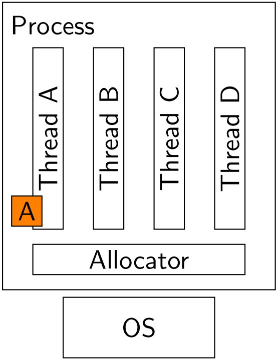
Once a virtual address is mapped, this mapping persists until the page is released to the OS.

Successive `malloc`/`free` operations



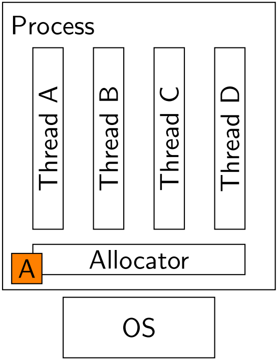
Successive `malloc`/`free` operations

- `malloc`: Thread A gets virtual page and touches it



Successive `malloc`/`free` operations

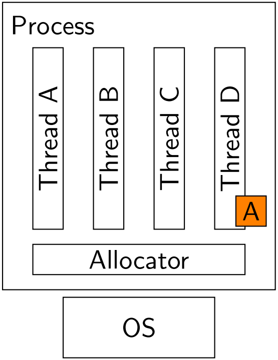
- `malloc`: Thread A gets virtual page and touches it
- `free`: page may be released to the allocators cache



Successive `malloc`/`free` operations

- `malloc`: Thread A gets virtual page and touches it
- `free`: page may be released to the allocators cache
- `malloc`: Thread D gets this page

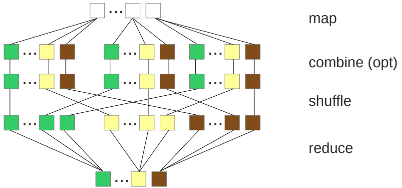
Thread D got remote memory from the allocator!



MapReduce workflow

MapReduce stages

- `map`: apply map-function to input
- `shuffle`: merge partitions
- `reduce`: apply reduce-function to all kv-pairs with the same key



- size of buffers unknown a priori
- iterative MapReduce: output of one MR step is input for the next

How to speed things up?

Memory allocators for SMPs (`tbbmalloc`)

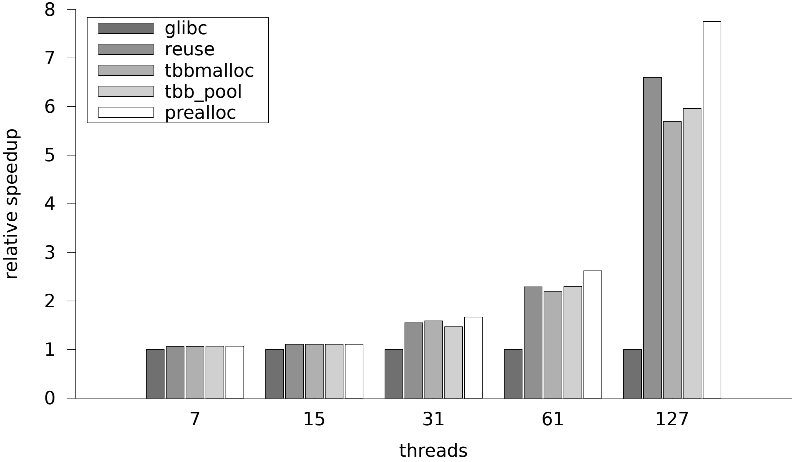
- provide fast concurrent allocations

Memory reuse (`reuse`)

- reuse buffers for subsequent MapReduce iterations

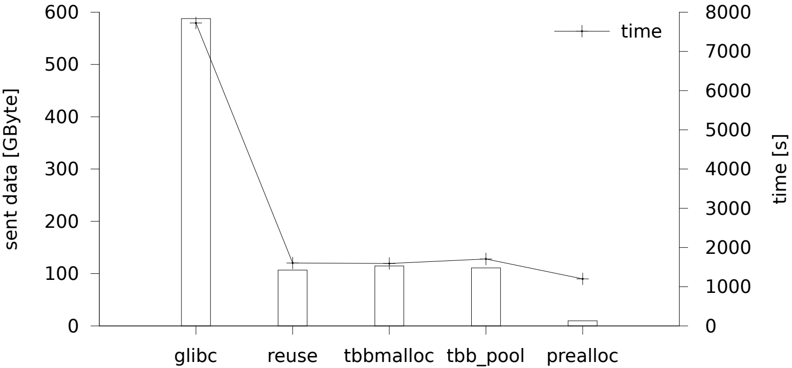
Memory preallocation (`prealloc`)

- allocate needed amount of memory for each buffer



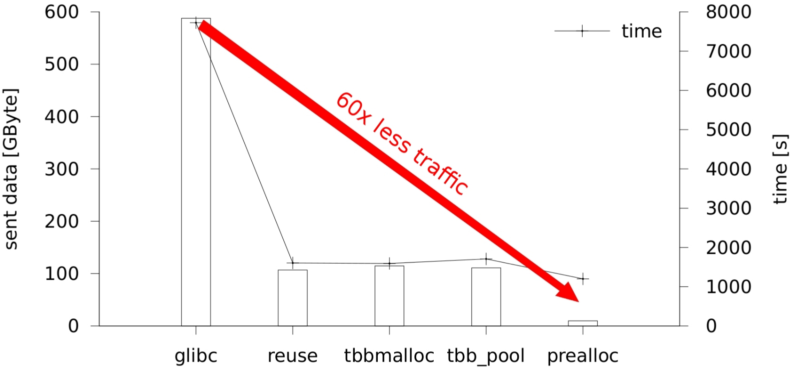
MR-Search with various allocators. Speedup is relative to *glibc*.

- Significant speedup if more than one blade is used.



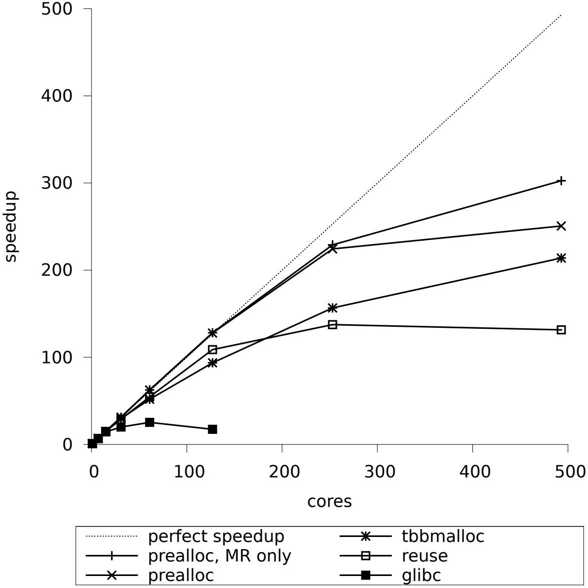
NUMA traffic and runtime with various allocators (127 Threads).

- Traffic on NUMAlink traced with Performance Co-Pilot
- TBB does not prevent remote memory

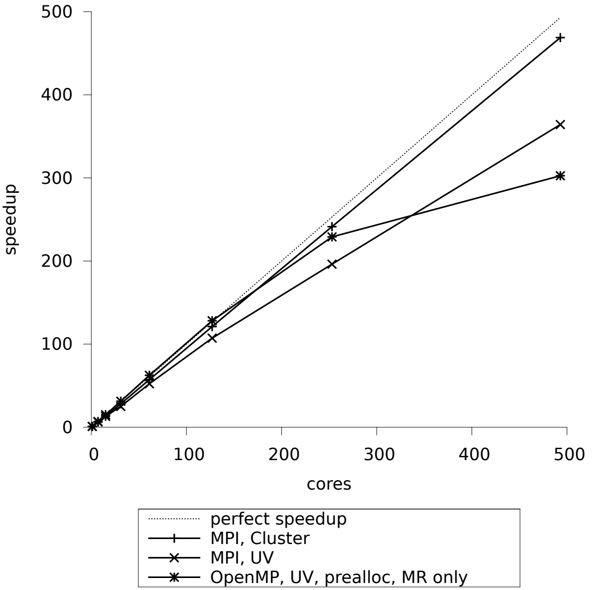


NUMA traffic and runtime with various allocators (127 Threads).

- Traffic on NUMAlink traced with Performance Co-Pilot
- TBB does not prevent remote memory



Scalability with various allocators.



Comparing scalability: OpenMP vs. explicit message passing

Summary

It is not that easy to write scalable code for large SMPs.

- large variability of memory access costs on large SMPs
- allocators for SMPs help to increase scalability
  - they do not prevent remote memory
- programmer needs to keep track of memory location (if possible)