Department of Computer Science, ETH Zürich

Supporting Parallelism in Operating Systems & Programming Languages

Fall 2013

Exercise 4: Memory Management

1 Per-thread object free-lists

The goal of this exercise is to implement application-specific memory management for the multi-threaded task scheduler developed in the previous exercises. Specifically, we target objects that are allocated/freed frequently. To allow for fast and scalable allocation/reclamation, your system should use per-thread free-lists for these objects.

For your intermediate report:

- Identify the objects in your system that are allocated/freed frequently and give an overview of a design with per-thread free-lists for these objects.
- Describe the allocation/reclamation steps and a synchronization scheme to make these operations thread-safe.
- Your system should have a bound on the total amount of unused memory it maintains. Describe how you plan to enforce this bound.

For your final report:

- Implement your design
- Provide an updated discussion of your implementation as in your intermediate report. Mention changes, if any.
- Evaluate the performance of your design. You can use the recursive Fibonacci number computation or other benchmarks. Compare against the default `malloc()` of your system, and against `tcmalloc`[1]. Discuss the results.

2 NUMA allocation

For your final report:

- Briefly describe how would you optimize your free-lists design from the previous section for NUMA architectures. Start with what your design goal would be and give a sketch on how would you plan to achieve it.

[1] http://goog-perftools.sourceforge.net/doc/tcmalloc.html

3 Stack size

For your final report:

- Why static task stack size is a hindrance for parallel task execution?
- Briefly describe how would you you deal with this issue, if you were to design a task parallel language from scratch.

4 Dates & Deliverables

29.11.2013: Intermediate report
06.12.2013: Final version of the code & final report