# Flat Parallelization

Vitaly Aksenov, Petr Kuznetsov

There are two intertwined factors that affect performance of concurrent data structures: the ability of processes to access the data in parallel and the cost of synchronization. It has been observed that for a large class of "concurrency-unfriendly" data structures, fine-grained parallelization does not pay off: an implementation based on a single global lock outperforms fine-grained solutions. The *flat combining* paradigm exploits this by ensuring that a thread holding the global lock sequentially *combines* requests and then executes the combined requests on behalf of concurrent threads.

In this paper, we propose a synchronization technique that unites flat combining and parallel bulk updates borrowed from parallel algorithms designed for the PRAM model. The idea is that the combiner thread assigns waiting threads to perform concurrent requests in parallel.

We foresee the technique to help in implementing efficient "concurrency-ambivalent" data structures, which can benefit from both parallelism and serialization, depending on the operational context. To validate the idea, we considered implementations of a *priority queue*. These data structures exhibit two important features: concurrent remove operations are likely to conflict and thus may benefit from combining, while concurrent insert operations can often be at least partly applied in parallel thus may benefit from parallel batching. We show that the resulting *flat parallelization* algorithm performs well compared to state-of-the-art priority queue implementations.