

Brief Announcement:

Optimally Work-Competitive Scheduling for Cooperative Computing with Merging Groups^{*}

Chryssis Georgiou[†]

Alexander Russell[†]

Alex A. Shvartsman[†]

The problem of cooperatively performing a set of N tasks in a decentralized setting where the computing medium is subject to failures is one of the fundamental problems in distributed computing. This problem is normally abstracted in terms of a set of N tasks that need to be performed in the distributed environment consisting of P processors, where the environment is prone to processor and/or communication failures. Algorithmic solutions for this problem are evaluated according to their computational effectiveness that measures the number of computation steps taken in performing the tasks. This measure is called *work*.

In the settings where network partitions may interfere with the progress of computation, the challenge is to maintain efficiency in performing the tasks and learning the results of the tasks, despite the dynamically changing group connectivity. However, no amount of algorithmic sophistication can compensate for the possibility of groups of processors or even individual processors becoming disconnected during the computation. In general, an adversary that is able to partition the network into g components will cause any task-performing algorithm to have work $\Omega(N \cdot g)$ even if each group of processors performs no more than the optimal number of $\Theta(N)$ tasks. In the extreme case where all processors are isolated from the beginning, the work of any algorithm is $\Omega(N \cdot P)$.

Even given the pessimistic lower bounds on work for partitionable networks, we aim to design and analyze efficient algorithmic approaches that can be shown to be better than the oblivious approach where each processor or each group performs all tasks.

Prior work established reasonably tight (in the length of the processor schedule) results for a *single* merge [3], and showed that a constant overhead relative to the lower bound can be achieved by algorithms using group communication services for a limited pattern of reconfigurations starting with a *single* group [2].

In order to understand better the *practical* implications of

performing work in partitionable settings, and especially in the settings where initially there may be several disconnected groups, we open a new research direction in the study of distributed load-balancing and pursue *competitive* analysis [5] of algorithms. Specifically, the competitive analysis that compares the work of a chosen algorithm with the work of an optimal algorithm that has complete information about the future changes in the network topology.

Our ultimate goal is to produce algorithms with guaranteed work efficiency for any pattern of fragmentations and merges of the underlying network. In the full paper [1] we report our current results for the abstract setting where asynchronous processors start performing tasks in isolation and where an adversary can force an arbitrary pattern of *merges*. Following a merge, processors are able to share their knowledge about the computational progress prior to the merge.

The summary of our results is as follows: (a) We formalize an adversary model that forces processors to start in isolation and be subjected to a pattern of merges: specifically, we model the adversary as a directed acyclic graph where each vertex corresponds to a group of processors (or a single processor) and a directed edge is placed from g_1 to g_2 , if g_2 was created by a merge operation involving g_1 . We furthermore label such edge with the total number of tasks which are completed by g_1 , before its merge into g_2 . We refer to such a DAG as a *Merge-DAG*. (b) We establish lower bounds on competitiveness: any α -competitive deterministic or randomized algorithm has $\alpha \geq 1 + 1/e$. In particular, we show that for every scheduling algorithm there is a Merge-DAG that causes the algorithm to be at least $(1 + 1/e)$ -work competitive. (c) We formalize a natural randomized algorithm for this problem, and we demonstrate, by induction on the number of merges, that it is $(1 + 1/e)$ -work competitive: in particular, the expected work performed by the algorithm is never more than $(1 + 1/e)$ times that of any algorithm, *even one with full knowledge of the future sequence of merges*. We also consider implementations of the algorithm using group communication services [4].

REFERENCES

- [1] C. Georgiou, A. Russell, and A. A. Shvartsman. Optimally work-competitive scheduling for cooperative computing with merging groups. At <http://www.engr.uconn.edu/~cg2/GRSmerges.ps>.
- [2] C. Georgiou and A. Shvartsman. Cooperative computing with fragmentable and mergeable groups. *Journal of Discrete Algorithms*. Accepted for publication. (Also in SIROCCO 2000, pages 141–156.)
- [3] G. Malewicz, A. Russell, and A. A. Shvartsman. Distributed cooperation during the absence of communication. In DISC 2000, pages 119–133.
- [4] *Special Issue on Group Communication Services*, volume 39(4) of *CACM*, 1996.
- [5] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *CACM*, 28(2):202–208, 1985.

^{*}This research is supported by the NSF Grants 9988304, 0121277, 0093065 and 9984774.

[†]Dept. of Computer Science and Engineering, 191 Auditorium Rd., Unit 3155, University of Connecticut, Storrs, CT 06269, USA. Emails: {cg2, acr, aas}@cse.uconn.edu.