**Jean-Pierre Lozi and Florian David and Gael Thomas and Julia Lawall and Gilles Muller,**

**Remote Core Locking: Migrating Critical-Section Execution to Improve the Performance of Multithreaded Applications**

**USENIX Annual Technical Conference (ATC'12), 2012.**

1. What is the motivation for this work? What are some sources of scalability problems? What is their specific focus? What are the precise goals of RLC? Why do previous approaches not meet these goal?
2. Which applications/workloads with high lock contention did they find? Does high lock contention seem to be the common case? What is the scalability of the applications with high lock contention? What is the scalability of those without lock contention?
3. How does a basic CAS spinlock work and what are its bottlenecks? Ticket lock? MCS lock? When do current spinlocks do okay? What is an existing alternative to spinlocks? What is MCS-TP?
4. What is the general idea of Remote Core Locking? When is it useful to have more than one server core?
5. With RCL, how many cache misses will be incurred across the client and server to execute a critical section? How many atomic instructions?
6. How are existing applications reengineered to use RCL? Which locks should use RCL and which should use traditional spinlocks? How many locks did they transform into RCL?
7. What would happen if a server blocked during critical section? How do they handle this? How do they ensure correct mutual exclusion on server? (Algorithm 1) What happens if a critical section contains a wait on a condition variable?
8. Is contention a good predictor of performance improvement? When is it not?
9. How much do their techniques improve application scalability? (Figure 9: what do the numbers above the bars mean?)
10. Why are two servers needed for Berkeley DB? How well do two servers work for Berkeley DB?
11. What do they still need to implement to make RCL more useful?