Paper title: Everything You Always Wanted To Know About Synchronization But Were Afraid To Ask
Paper authors: T. David, R. Guerraoui, V. Trigonakis
Your name: Parth Kansara (115135130)

What problem does the paper address? How does it relate to and improve upon previous work in its domain?
The paper offers a thorough insight into synchronization schemes and how it is dependent closely on the hardware, e.g. the cache-coherence protocols. The authors analyze the varying latencies of the cache-coherence protocols, the performance of various atomic operations, the locking and message passing techniques along with a concurrent hashtable, in-memory key-value store and transactional memory in software. Further, observations were made by varying the architecture, from single-socket uniform and non-uniform to multi-socket multi-core directory and broadcast.

Hackenberg conducted studies on cache-coherence protocols in multi-sockets, measuring the latency of only data loading. **Molka** studied the effect of memory hierarchy on various workloads which was not very relevant to high contention systems. **Moses** demonstrated a decrease in performance of spin locks under non-uniformity, but this was confined to one type of lock and hardware model. This paper builds upon the previous works and generalizes the effect on synchronization for a variety of scenarios.

Mellor-Crumney, **Anderson** and **Luchangco** observed the limited scalability of specific locks, which did not cover different hardware platforms. The paper also presents some optimizations targeted at improving the scalability of the plain spin locks.

Several works study the OS scalability on multi-core systems and restructure the OS. **Tornado**, **Corey**, **Barrelfish** and **fos** are few of the works in this domain that look at tweaking the kernel design. This paper however shows how these issues stem from the hardware and offer software techniques and optimizations can eliminate the need for a kernel overhaul.

What are the key contributions of the paper?
The paper offers an exhaustive analysis of synchronization by building a cross-platform synchronization suite called SSYNC. This implementation includes various state-of-the-art lock algorithms, implemented with certain optimizations. It also abstracts message passing for various platforms. Further, it provides microbenchmarks that can compute the latencies of the cache-coherence protocols, the locks and the message passing. SSYNC also provides libraries that can be used to build software implementation of a portable transactional memory, a concurrent hash table, and a key-value store.

This paper leverages SSYNC to perform measurements on various platforms. Two multi-socket multi-core systems were selected - 4-socket directory-based AMD Opteron and 8-socket broadcast-based Intel Xeon. Two chip multi-processors (CMPs) were selected - 8-socket uniform Sun Niagara 2 and 36-core non-uniform Tilera TILE-Gx36. This wide range of platforms allows for a comprehensive study.

Briefly describe how the paper's experimental methodology supports the paper's conclusions.
Using SSYNC, the authors provide an insight into the scalability of various synchronization schemes across the above mentioned hardware architectures. It was observed that inter-socket latencies for any operation on cache lines does not scale, even under no contention. Thus, sharing across sockets should be avoided. Even in cases where threads are explicitly placed on the same socket, an incomplete cache directory and a non-inclusive LLC may give rise to inter-socket communication, which is expensive, as observed on Opteron. This can however be overcome by explicitly maintaining a modified state on the cache line. Further, it was observed that load and store operations are not particularly cheaper than the atomic operations,

Under high contention, the uniformity in the distance of the cores from the LLC inside a socket improves scalability of synchronization. Also, in such situations, message passing performs better. But locks outperform message passing in cases of low contention, and so message passing can be confined to highly contended parts of the system. In particular, none of the locking schemes can be labeled as the most optimal across all architectures and workloads, and the choice of lock is based on the particular scenario. However, the authors state that an efficient implementation of ticket lock would be optimal in most situations.

Based on these observations, the authors reinforce their conclusion that scalability of synchronization is in fact based on the underlying hardware and that it can prove to be a bottleneck when scaling systems.

Write down one question you may want to ask the authors.
How would the introduction of a multi-kernel OS impact the scalability of synchronization?