### Dependency-Aware Parallel Tasks

Afshin Zafari, Martin Tillenius, Elisabeth Larsson

---

**Programming Model**

**Motivation**
- Fine-grained parallelism is required for performance
- Need simpler abstractions
- Abstractions must support expressiveness

**Programming Model**
- The program is divided up into tasks
- Tasks are annotated with which data they need
- Dependencies are deduced from these annotations

**Data Versions**
- Versions of task parameters specify dependencies
- More expressive than a DAG
- A task requires a certain version to run

**Distributed Memory Implementation**
- All shared data is distributed over the nodes
- Remote data is requested by listeners
- When the requested versions become ready, data is sent to all requesting nodes
- Communication is non-blocking
- Viewed as state machine to handle async events

---

**Experiment: Dependencies**

**Experiment**: Cholesky factorization, as an example of non-trivial dependencies. 

**Speedup**: 28.2x on 32 cores, compared to a single MKL call. MKL also has parallel Cholesky, but only achieved 21.1x. MKL might not be optimized for the AMD Bulldozer system this experiment was run on.

**Experiment: Distributed Memory**

**Experiment**: Assembly of a matrix, to be used in a partial differential equations solver.

**Speedup**: 97.7x on 128 cores (over best serial speed)

---

**Experiment: Scalability**

**Experiment**: Find what task granularity is required for scaling. We avoid effects from shared resources by executing tasks that only delay for a specific time.

**Speedup**: Tasks give easy access to performance

---

**Conclusions**

**Acknowledgements**

Thanks to Kostis Sagonas for providing access the 64 core machine used in parts of the experiments. The other computations were performed on the Tintin cluster provided by SNIC through UPPMAX under project p2009014.