Porting Distributed Data Structures to Future Many-core Architectures

Panagiota Fatourou*†, Nikolaos D. Kallimanis*, Eleni Kanellou*, Odysseas Makridakis†, and Christi Symeonidou*

* Institute of Computer Science - Foundation of Research and Technology Hellas
† Department of Computer Science, University of Crete
A foreseeable future in architecture

- $m$ islands (or clusters)
  - each comprised of $c$ cores
  - containing one or more processors
- **No (hardware) cache coherence**
  - main memory is split into modules
  - each module associated to a distinct island (or core)
- **Message passing.**
- DMA is possible.

Data structures crucial for portability and productivity of software design!
Exploiting architecture characteristics

**DS** stored in distributed directory.
- One server acts as synchronization manager.
- Communication is distributed over the system.

**DS** distributed over local memories in token ring.
- Communication with current token server.
- When local memory fills up, token is forwarded.

Exploiting communication characteristics:
- **Hierarchical** design exploits fast intra-island communication.
- Requests are **batched** in each island and forwarded to appropriate server.
Factors that lead to good scaling

![Graph showing throughput vs. number of cores for different queue types]

- Hierarchical and other optimizations play a big role in scalability.
- Directory-based + Hierarchical: Scales well!
- Token-based: suited for moderately sized data structures.

Total # of messages in the system doesn't seem to play a role in scalability.
Our Contribution

→ A first step towards designing customized scalable data structures for future non cache-coherent many-core architectures.

- We studied techniques for implementing distributed DS for many-core architectures.
  - Challenges:
    - Non or partially cache-coherent memory
    - Message-passing paradigm
  - We combined variants of these techniques to design a rich collection of DS
    - stacks, queues, dequeues, lists, sets
  - This collection could be utilized by high-productivity languages
    - Easy porting to new and future architectures!
    - Collection can be used as library even by non-experts
  - Our experiments illustrate the scalability power of the hierarchical approach.