Brandenburg
University of Technology
Cottbus - Senftenberg

26th August 2014

SHARED MEMORY IN THE MANY-CORE AGE

Stefan Nürnberger, Gabor Drescher, Randolf Rotta, Wolfgang Schröder-Preikschat, and Jörg Nolte

BTU Cottbus-Senftenberg
FAU Erlangen-Nuremberg

What does it provide?

- enable shared memory programming where hardware does not (directly)
- either in HW (e.g. Cache Coherence), HW/SW (e.g. most page-based DSMs), SW (Global Arrays, X10, PGAS systems, ...)

Can simplify programming:

- handling of complex and dynamic data dependencies
- decouples task scheduling from data placement
- enables load balancing when data dependencies are unknown upfront

. . . FOR MANY-CORE SYSTEMS???

Hardware Characteristics changed!

Many-Core hardware looks vastly different from what DSM systems were designed for in the past.

New Memory Models!

Shared memory programming came a long way in defining sane semantics for concurrent memory access.

Programming Environments evolved!

DSM will coexist and cooperate with other programming models.

A NEW GOLDEN AGE FOR DSM RESEARCH?

1. **Many-Core Challenges**
2. **Memory Model Opportunities**
3. **Building Blocks for a DSM System**

A CHANGE IN ARCHITECTURE

past:	network of single-cores
today:	network of multi- and many-cores (e.g. Xeon Phi)

General many-core implications:

- hardware provides basic consistency within node
- intra-node parallelism is mandatory
- low-latency networks vs. slow cores
- inherent NUMA hierarchy

MUST EXPLOIT CONSISTENCY ISLANDS

Brandenburg
University of Technology
Cottbus - Senftenberg

Powerful Memory Model within Consistency Islands

- illusion of shared memory
- observability of stores within the island
- synchronizing memory operations

Between Consistency Islands

- networks provide remote memory access
- consistency does not scale-out across nodes
- no observability of stores in other islands
- limited memory order enforcement

Why are they important?

- separate replica for each thread waste memory and cache space
- can use efficient hardware caches
- threads sharing replica split cost of management

MUST EXPLOIT PARALLELISM

Remote memory is quite fast

- typical remote memory access (IB put/get) is $\sim 2\mu s$
- latency of approx. 8 cache misses (~ 260 cycles/miss)

$\Rightarrow$ low overhead management is a must

$\Rightarrow$ parallelize DSM protocol/management

The effect of Huge Pages

- using 2MB pages instead of 4k increases TLB reach (important!)
- page-based DSM needs to deal with factor $\times 512$ in size

$\Rightarrow$ parallelize and vectorize diff/merge

A CHANGE IN MEMORY MODELS

past:	evade communication, hide latency
today:	be explicit about concurrency

DSM inspired memory models

“What can we optimize? How can we eliminate synchronization overhead?”

- DSM systems introduced memory models that fit their protocol
- non-uniform models: (Lazy) Release Consistency, Scope Consistency, ...

Uniform vs. non-uniform models

- uniform models consider only data read and write
- non-uniform models add explicit synchronization primitives

CONSISTENCY IS RELAXED TODAY

Brandenburg
University of Technology
Cottbus - Senftenberg

The Memory Consistency Model

describes a memory abstraction’s behavior and what measures are provided to make it behave!

Relaxed Memory Models

Use relaxation of memory consistency for optimization

HW: start from weak model and add measures (e.g. fences) to strengthen (e.g. Sparc-TSO, ARM, . . .)

SW: start from strong model and add hints (annotations) to weaken (e.g. C/C++11 with low level atomics)

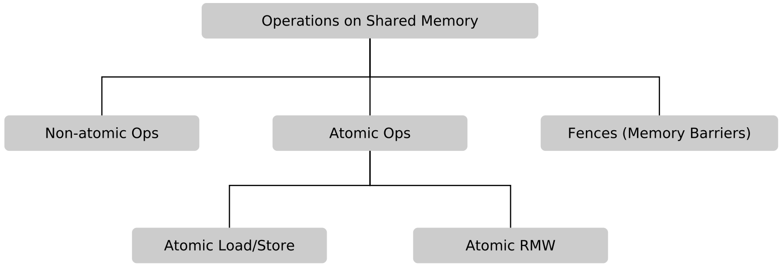
DATA RACES VS. CONSISTENCY

Brandenburg University of Technology Cottbus - Senftenberg

- **There is no benign data race!**
- all data races threaten consistency, operation outcome is undefined

$\Rightarrow$ guarantee SC for DRF only (or probably SC for HRF)

$\Rightarrow$ further relaxations based on C++11 model



AND THEN THERE'S TM

Transactional Memory affects the whole Memory Model

- not a trivial thing (see Intel TSX bug)
- reactive data race freedom
- expected to allow lock elision

Software Transactional Memory

- Language extensions for transactions available
- provides data access and modification tracking in transactions

⇒ viable alternative to access traps of page-based DSM

A CHANGE IN USAGE STRATEGY

Brandenburg
University of Technology
Cottbus - Senftenberg

past: “one size fits all” approach

today: in combination with PGAS, function shipping, message passing

The former DSM approaches

- Wrappers running legacy code (wrap C-library calls, OS-based)
- Library/Framework based
  - Threads cannot be implemented as library (H.-J. Boehm)
- Language embedded

A future for DSMs…

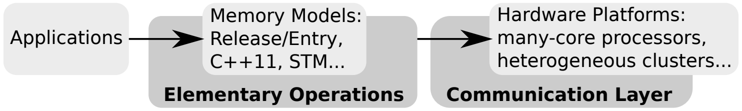
- use DSM below the programming model, not as programming model
- Enable use of modern memory models across clusters
- Most programs benefit from caching (in general)

$\Rightarrow$ Think: ‘PGAS + Caching’, ‘Software Managed Caches’

FROM APPLICATION TO PLATFORM

- optimal DSM strategy depends on target programming/memory model, access patterns, hardware characteristics, . . .
- evaluate many protocols with common mechanisms

$\Rightarrow$ need infrastructure (framework) for the commonalities and variabilities


Applications
Memory Models: Release/Entry, C++11, STM...
Elementary Operations
Hardware Platforms: many-core processors, heterogeneous clusters...
Communication Layer

Elementary PGAS operations

Allocators	for globally coordinated memory management
Remote	memory access
Atomic	memory operations
Thread	groups to represent consistency islands

Elementary operations for replication

Replica	management - creation / update / invalidation - asynchronous replica notifications - including efficient replica group update/invalidation
Diff/Merge	mechanism,
Access tracking	for read and write access to each replica,
Versioned	modification management

WHERE DO WE GO FROM HERE?

Brandenburg
University of Technology
Cottbus - Senftenberg

- implement minimal event-driven kernel
- implement basic mechanism on top of kernel
- envisioned as lightweight "firmware" for many-core consistency

Research Projects

COKE	consistency kernel exploring elementary operations
MyThOS	many threads operating system
OctoPOS	includes evaluation of new memory models

CONCLUSIONS

Brandenburg
University of Technology
Cottbus - Senftenberg

It is an interesting time to do DSM research again

- Consistency Islands need to be exploited
- Overhead of replica management is challenging

Memory Consistency Models changed dramatically

- C/C++11, SC for DRF, . . . for software portability
- Old DSM models are not the best fit, inhibit portability

Infrastructure for DSM is needed

- DSMs share common tasks (allocate, update/invalidate, . . .)
- Elementary operations allow for experimental implementations