High-Performance Distributed RMA Locks

TORSTEN HOFELER

with support of Patrick Schmid, Maciej Besta @ SPCL
presented at Wuxi, China, Sept. 2016
ETH, CS, Systems Group, SPCL

- ETH Zurich – top university in central Europe
  - Shanghai ranking ‘15 (Computer Science): #17, best outside North America
  - 16 departments, 1.62 Bn $ federal budget
- Computer Science department
  - 28 tenure-track faculty, 1k students
- Systems group (7 professors)
  - Focused on systems research of all kinds (data management, OS, …)
- SPCL focuses on performance/data/HPC
  - 1 faculty
  - 3 postdocs
  - 8 PhD students (+2 external)
  - 15+ BSc and MSc students
  - [http://spcl.inf.ethz.ch](http://spcl.inf.ethz.ch)
  - Twitter: @spcl_eth
crClim – Cloud-resolving Climate Simulations

**COSMO NWP-Applications**

- **DWD (Offenbach, Germany):**
  - NEC SX-8R, SX-9

- **MeteoSwiss:**
  - Cray XT4: COSMO-7 and COSMO-2 use 980+4 MPI-Tasks on 246 out of 260 quad core AMD nodes

- **ARPA-SIM (Bologna, Italy):**
  - Linux-Intel x86-64 Cluster for testing (uses 56 of 120 cores)

- **USAM (Rome, Italy):**
  - HP Linux Cluster XEON biproc quadcore System in preparation

- **Roshydromet (Moscow, Russia), SGI**

- **NMA (Bucharest, Romania):**
  - Still in planning / procurement phase

- **IMGW (Warsawa, Poland):**
  - SGI Origin 3800: uses 88 of 100 nodes

- **ARPA-SIM (Bologna, Italy):**
  - IBM pwr5: up to 160 of 512 nodes at CINECA

- **COSMO-LEPS (at ECMWF):**
  - Running on ECMWF pwr6 as member-state time-critical application

- **HNMS (Athens, Greece):**
  - IBM pwr4: 120 of 256 nodes
NEED FOR EFFICIENT LARGE-SCALE SYNCHRONIZATION
Locks

Inuitive semantics

Various performance penalties

An example structure
LOCKS: CHALLENGES
LOCks: CHALLENGES

We need intra- and inter-node topology-awareness

We need to cover arbitrary topologies
LOCKS: CHALLENGES

We need to distinguish between readers and writers

We need flexible performance for both types of processes

What will we use in the design?
WHAT WE WILL USE

MCS Locks
WHAT WE WILL USE
Reader-Writer Locks
How to manage the design complexity?

How to ensure tunable performance?

What mechanism to use for efficient implementation?
REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p
Memory
A
B

A put

get B

flush

Process q
Memory
A
B

Cray
BlueWaters
REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA
How to manage the design complexity?

How to ensure tunable performance?

What mechanism to use for efficient implementation?
How to manage the design complexity?

Each element has its own distributed MCS queue (DQ) of writers.

Readers and writers synchronize with a distributed counter (DC).

MCS queues form a distributed tree (DT).

Modular design.

2 2 2 2
R1 R2 R3 R4
W1 W2 W3 W4

2 2 3
R5 R6 R7
W5 W6 W7 W8

2
R8 R9
W8
How to ensure tunable performance?

Each DQ: fairness vs throughput of writers

DC: a parameter for the latency of readers vs writers

DT: a parameter for the throughput of readers vs writers

A tradeoff parameter for every structure
DISTRIBUTED MCS QUEUES (DQs)
Throughput vs Fairness

Larger $T_{L,i}$: more throughput at level $i$. Smaller $T_{L,i}$: more fairness at level $i$.

Each DQ: The maximum number of lock passings within a DQ at level $i$, before it is passed to another DQ at $i$.

$T_{L,i}$
DISTRIBUTED TREE OF QUEUES (DT)
Throughput of readers vs writers

DT: The maximum number of consecutive lock passings within readers ($T_R$).
**DISTRIBUTED COUNTER (DC)**

Latency of readers vs writers

DC: every $k$th compute node hosts a partial counter, all of which constitute the DC.

$$k = T_{DC}$$

A writer holds the lock

Readers that arrived at the CS

Readers that left the CS

$$T_{DC} = 1$$

$$T_{DC} = 2$$
THE SPACE OF DESIGNS

- Higher throughput of writers vs readers
- Locality vs fairness (for writers)

Design A
Design B

Lower latency of writers vs readers

$T_{DC}$

$T_{L,i}$

$T_{R}$
**Lock Acquire by Readers**

A lightweight acquire protocol for readers: only one atomic fetch-and-add (FAA) operation.

- A writer holds the lock.
- Readers that arrived at the CS.
- Readers that left the CS.
LOCK ACQUIRE BY WRITERS

Acquire the main lock

Acquire MCS

Acquire the main MCS lock
EVALUATION

- CSCS Piz Daint (Cray XC30)
- 5272 compute nodes
- 8 cores per node
- 169TB memory
EVALUATION CONSIDERED BENCHMARKS

- The latency benchmark

Throughput benchmarks:
- Empty-critical-section
- Single-operation
- Wait-after-release
- Workload-critical-section

DHT
Distributed hashtable evaluation
EVALUATION
DISTRIBUTED COUNTER ANALYSIS

Throughput, 2% writers
Single-operation benchmark

Throughput [mln locks/s] vs. MPI processes (P)

- $T_{DC}$
- 64
- 32
- 16
- 8
- 4
- 2

Throughput maxima observed at different process counts for various $T_{DC}$ values.
EVALUATION
READER THRESHOLD ANALYSIS

Throughput, 0.2% writers,
Empty-critical-section benchmark

Throughput [mln locks/s]

MPI processes (P)

T_R
- 6000
- 5000
- 4000
- 3000
- 2000
- 1000
EVALUATION
COMPARISON TO THE STATE-OF-THE-ART

EVALUATION
COMPARISON TO THE STATE-OF-THE-ART

Throughput, single-operation benchmark

Percentages are values of $F_W$

EVALUATION
DISTRIBUTED HASH TABLE

20% writers

10% writers

Scheme
- foMPI–A
- foMPI–RW
- RMA–RW

Total time [s]

MPI processes (P)

EVALUATION
DISTRIBUTED HASHTABLE

2% of writers

0% of writers

OTHER ANALYSES