RAMP: RDMA Migration Platform

Babar Naveed Memon, Xiayue Charles Lin, Arshia Mufti, Arthur Scott Wesley, Tim Brecht, Kenneth Salem, Bernard Wong, and Benjamin Cassell
Contact @ firstname.lastname@uwaterloo.ca
RDMA and RDMA-based Systems

- What and why?
RDMA and RDMA-based Systems

- What and why?
- What is the right programming model?
RDMA and RDMA-based Systems

- What and why?
- What is the right programming model?

Shared Memory
- FaRM
- NAM-DB
Motivation
Motivation

Support Configuration Operations in Loosely Coupled Distributed Systems
Motivation

Support **Configuration Operations** in **Loosely Coupled Distributed Systems**

- Loosely Coupled Applications
Motivation

Support **Configuration Operations** in **Loosely Coupled Distributed Systems**

- Loosely Coupled Applications
Motivation

Support Configuration Operations in Loosely Coupled Distributed Systems

- Loosely Coupled Applications
- Configuration Operations
  - Scale out, scale in or load balance
Motivation

Support **Configuration Operations** in **Loosely Coupled Distributed Systems**

- Loosely Coupled Applications
- Configuration Operations
  - Scale out, scale in or load balance
- Is shared memory overkill?
Desired Properties for RAMP
Desired Properties for RAMP

● On-The-Fly Bulk Data Movement
  ○ Minimize interference with on-going application workload
  ○ Particularly at the source
Desired Properties for RAMP

● On-The-Fly Bulk Data Movement
  ○ Minimize interference with on-going application workload
  ○ Particularly at the source

● Non-Intrusive
  ○ Stay out of the way, except during configuration options
  ○ Avoid “shared storage” approach
  ○ Local memory access faster than RDMA
Desired Properties for RAMP

- **On-The-Fly Bulk Data Movement**
  - Minimize interference with on-going application workload
  - Particularly at the source

- **Non-Intrusive**
  - Stay out of the way, except during configuration options
  - Avoid “shared storage” approach
  - Local memory access faster than RDMA

- **Application-Managed**
  - Application controls when data moves, and where it moves
RAMP: The Big Picture
RAMP: The Big Picture

- Coordinated memory segments
  - Single reader/writer
  - Contains application data
RAMP: The Big Picture

- Coordinated memory segments
  - Single reader/writer
  - Contains application data

- Segments are migratable
RAMP: The Big Picture

- Coordinated memory segments
  - Single reader/writer
  - Contains application data

- Segments are migratable

- No serialization/deserialization of application data during migration
RAMP Functionality
RAMP Functionality

Cluster of Servers

Server 1
- Partition1
- Partition2
- Partition3

Server 2
- Partition4
- Partition5
- Partition6

Server 3
- Partition7
- Partition8
- Partition9

Client 1
Client 2
Client 3
... Client n
RAMP Memory Segment Allocation

Server_1  Server_2  Server_3
RAMP Memory Segment Allocation

Server_1  Server_2  Server_3

RAMP Arena
RAMP Memory Segment Allocation

Server_1

Server_2

Server_3

RAMP Segment

RAMP Arena
RAMP Memory Segment Allocation

Server\textsubscript{1} \hspace{1cm} Server\textsubscript{2} \hspace{1cm} Server\textsubscript{3}

RAMP Segment

RAMP Arena

RAMP Segment
Using RAMP Segments
Using RAMP Segments

Application
Data
Structures

Server$_1$
Using RAMP Segments

- Optional RAMP-provided in-segment C++ containers (vectors, maps) using custom memory allocators
Normal Operation

Cluster of Servers

Server 1
- Partition1
- Partition2
- Partition3

Server 2
- Partition4
- Partition5
- Partition6

Server 3
- Partition7
- Partition8
- Partition9

Client 1
Client 2
Client 3
\cdots
Client n
RAMP Segment Migration (Phase 1)
RAMP Segment Migration (Phase 1)
RAMP Segment Migration (Phase 1)
RAMP Segment Migration (Phase 1)

- Registration has a high latency cost (100’s ms)
- … but segment remains available
RAMP Segment Migration (Phase 2)
RAMP Segment Migration (Phase 2)
RAMP Segment Migration (Phase 2)

- Transfers segment ownership (not data)
- Low latency operation (20 microseconds)
RAMP Segment Migration (Phase 3)
RAMP Segment Migration (Phase 3)
RAMP Segment Migration (Phase 3)
RAMP Segment Migration (Phase 3)

- Implemented using one-sided RDMA reads
- Application managed vs RAMP managed
RAMP Segment Migration (Phase 4)
RAMP Segment Migration (Phase 4)

Server\textsubscript{1} \quad \text{CLOSE} \quad Server\textsubscript{2}

- Clean up
Segment Migration Performance
Segment Migration Performance

- STL map with 8B keys and 128B values in 256MB segment
Segment Migration Performance

- STL map with 8B keys and 128B values in 256MB segment
- Single thread at receiver starts using the map immediately after TRANSFER
Segment Migration Performance

- STL map with 8B keys and 128B values in 256MB segment
- Single thread at receiver starts using the map immediately after TRANSFER
- Latency of map “get” operations, as a function of time since TRANSFER
Segment Migration Performance

- STL map with 8B keys and 128B values in 256MB segment
- Single thread at receiver starts using the map immediately after TRANSFER
- Latency of map “get” operations, as a function of time since TRANSFER
Segment Migration Performance

- Paging first access
  40 µs
- Stop-and-copy first access
  310 ms
Conclusion

- Overview of RAMP, lightweight support for configuration operations in loosely coupled systems
Conclusion

- Overview of RAMP, lightweight support for configuration operations in loosely coupled systems
- Coordinated allocation of segments, fast ownership transfer, application-managed data movement
Conclusion

- Overview of RAMP, lightweight support for configuration operations in loosely coupled systems
- Coordinated allocation of segments, fast ownership transfer, application-managed data movement
- In the paper:
Conclusion

● Overview of RAMP, lightweight support for configuration operations in loosely coupled systems
● Coordinated allocation of segments, fast ownership transfer, application-managed data movement
● In the paper:
  ○ Many more details
Conclusion

● Overview of RAMP, lightweight support for configuration operations in loosely coupled systems
● Coordinated allocation of segments, fast ownership transfer, application-managed data movement
● In the paper:
  ○ Many more details
  ○ Rcached: memcached-like in-memory k/v store, using RAMP for load balancing
Feedback

- The right abstraction for the application
- Is shared memory abstraction overkill for loosely coupled data intensive applications?
Thank You
Rcached

- Memcached with
  - RAMP based Hash-Maps
  - Ability to migrate partitions
- 128 partitions hashed across 4 servers
- 40 million keys (key = 8 Bytes, Value = 128 Byte)
- 100 closed loop clients
- Per server latency noted over 40000 request windows
Rcached (2)