CRD

MPI, SHMEM, and UPC Performance on the Cray X1 – A Case Study Using APEX-Map

Hongzhang Shan

Erich Strohmaier

Computational Research Division

Lawrence Berkeley National Laboratory

Hshan, estrohmaier@lbl.gov

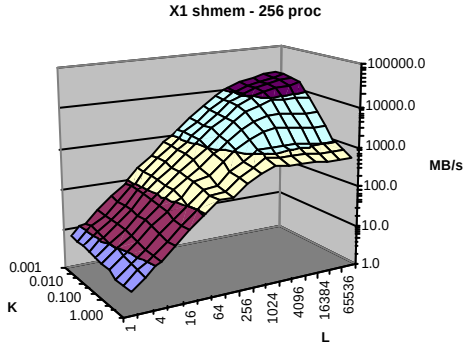
Office of Science
U.S. DEPARTMENT OF ENERGY

APEX-Map

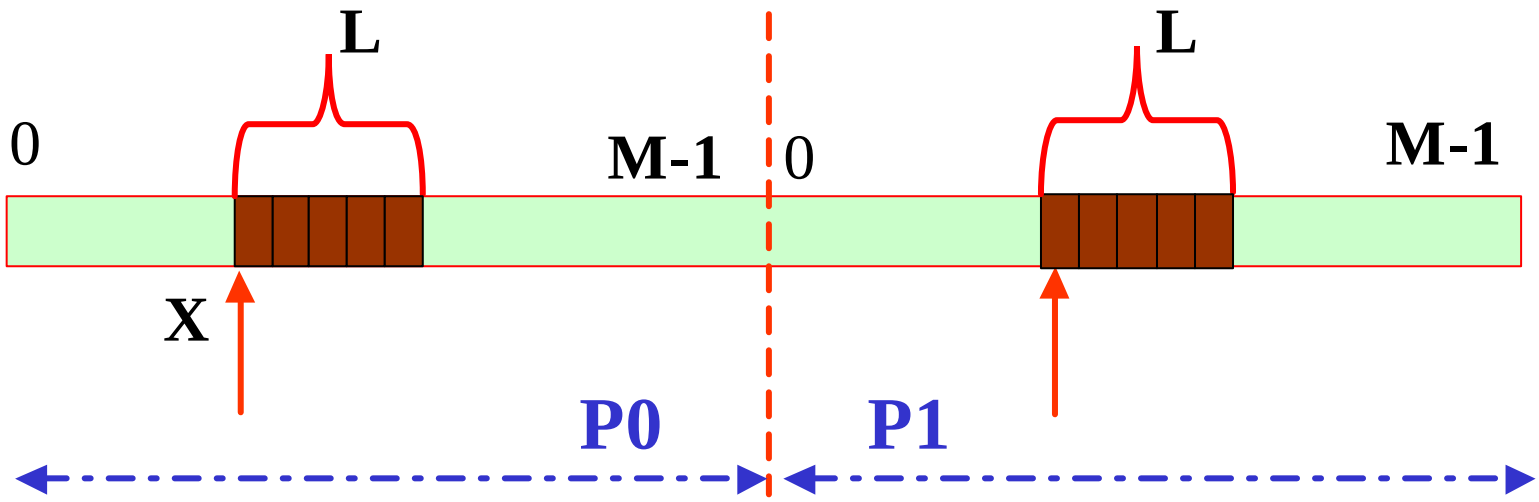
- A synthetic performance probe to study global data access
- It has three parameters:
  - M : memory size accessed
  - K : temporal locality
  - L : spatial locality
- It tries to mimic the data access behavior of real scientific applications

APEX-Map Features

- Measuring how fast the data could be loaded into computing units inside CPUs
- Generating a continuous performance surface
  - Visualize the effect of temporal locality and spatial locality
  - Allow flexible comparison



APEX-Map Design

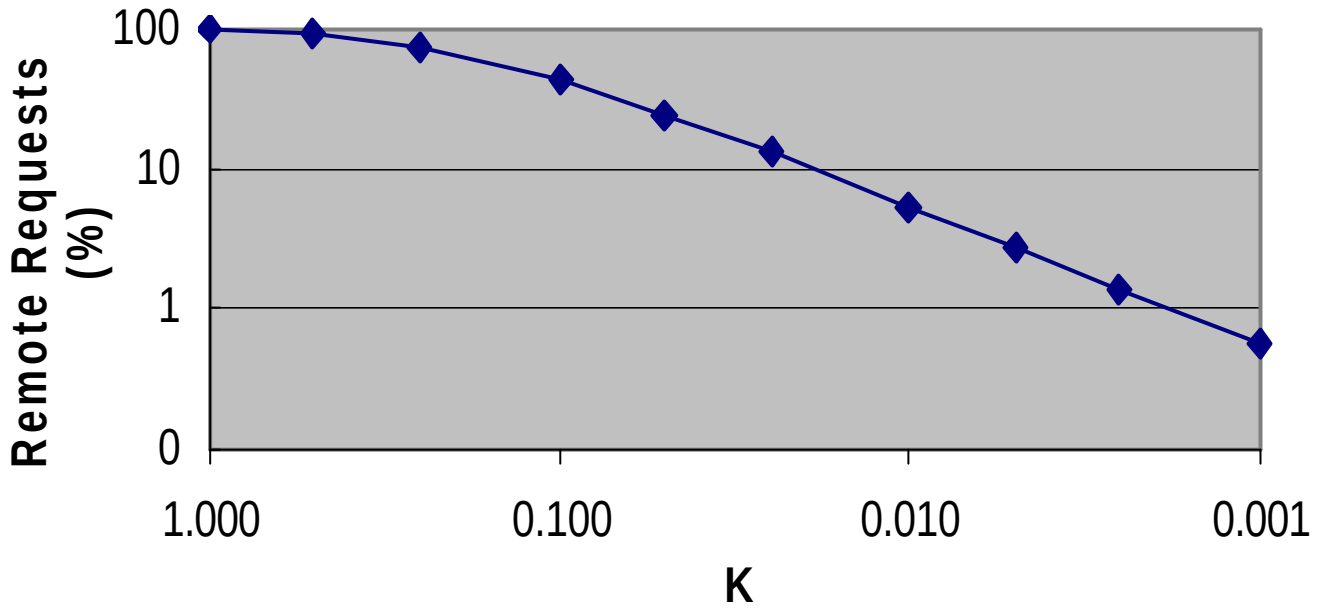


- Data evenly distributed among processes
- L contiguous addresses will be accessed together
- Each remote access is a communication message with length L (in message passing)

Generate Indices X

- Following non-uniform random access
- Generated using a power distribution function controlled by parameter K
- Adjust for process with rank **myid** because local data for each process is different :

Effect of K On Number of Remote Requests



- For 256 processes
- Higher temporal locality, less remote requests

APEX-Map Outline

Repeat N Times

Generate Index Array()

CLOCK(start)

For each Index *i* in the Array

If (not local data)

Get Remote Data ()

End If

Compute ()

CLOCK(start)

RunningTime += end - start

End Repeat

SHMEM Implementation

Repeat N Times

Generate Index Array()

CLOCK(start)

For each Index *i* in the Array

If (not local data)

SHMEM_DOUBLE_GET()

End If

Compute()

CLOCK(start)

RunningTime += end - start

End Repeat

UPC Implementation

Repeat N Times

Generate Index Array()
CLOCK(start)
For each Index *i* in the Array
If (not local data)
//method 1
UPC_MEMGET()
Compute()
//method 2
p = global_data[rid]
for (i = 0; i < L; i++)
Compute(*(p+i))
Else
Compute()
End If
CLOCK(start)
RunningTime += end - start

End Repeat

MPI Implementation

Repeat N Times

Generate Index Array()

CLOCK(start)

For each Index *i* in the Array

If (not local data)

Generate Remote Request()

Else

Compute()

End If

Serve Incoming Requests()

Process Replies ()

CLOCK(start)

RunningTime += end - start

End Repeat

Computing

Waiting

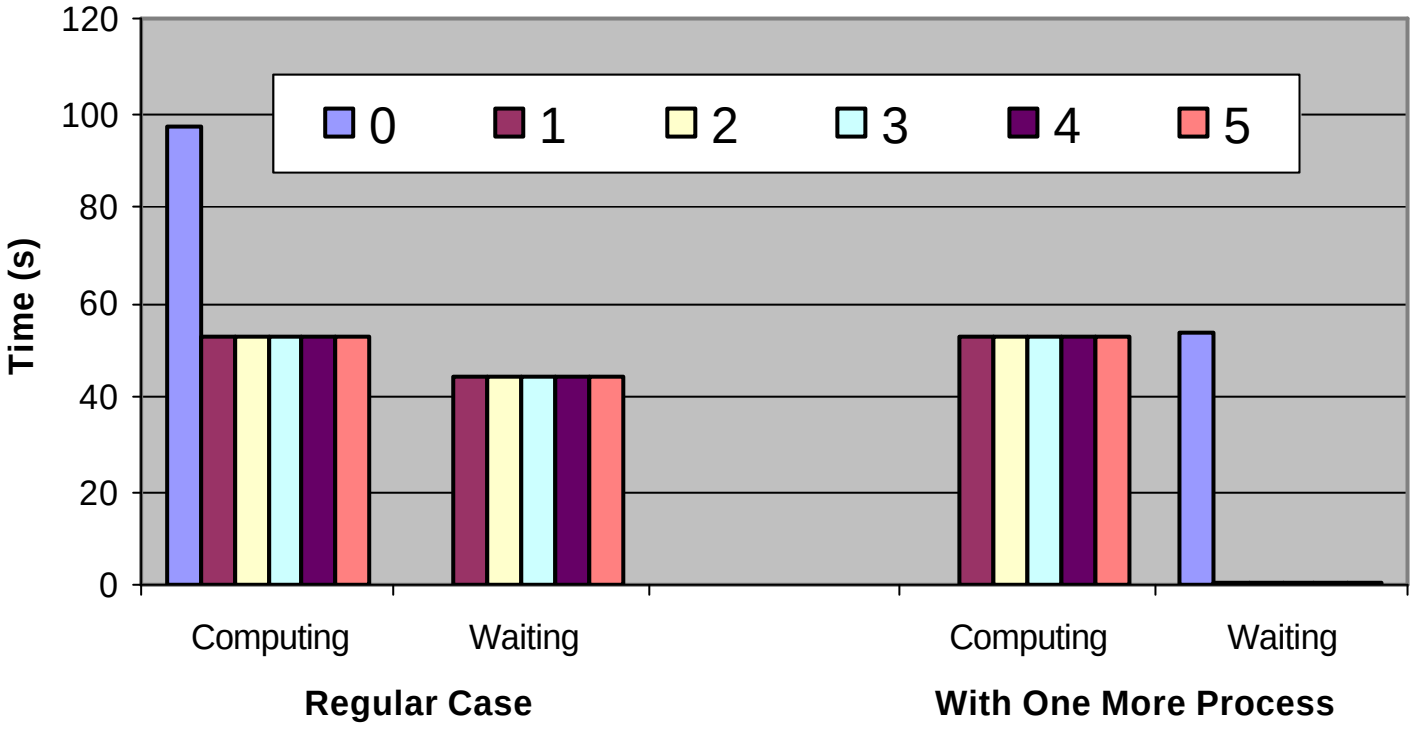
CLOCK(start)

Wait For Finish ()

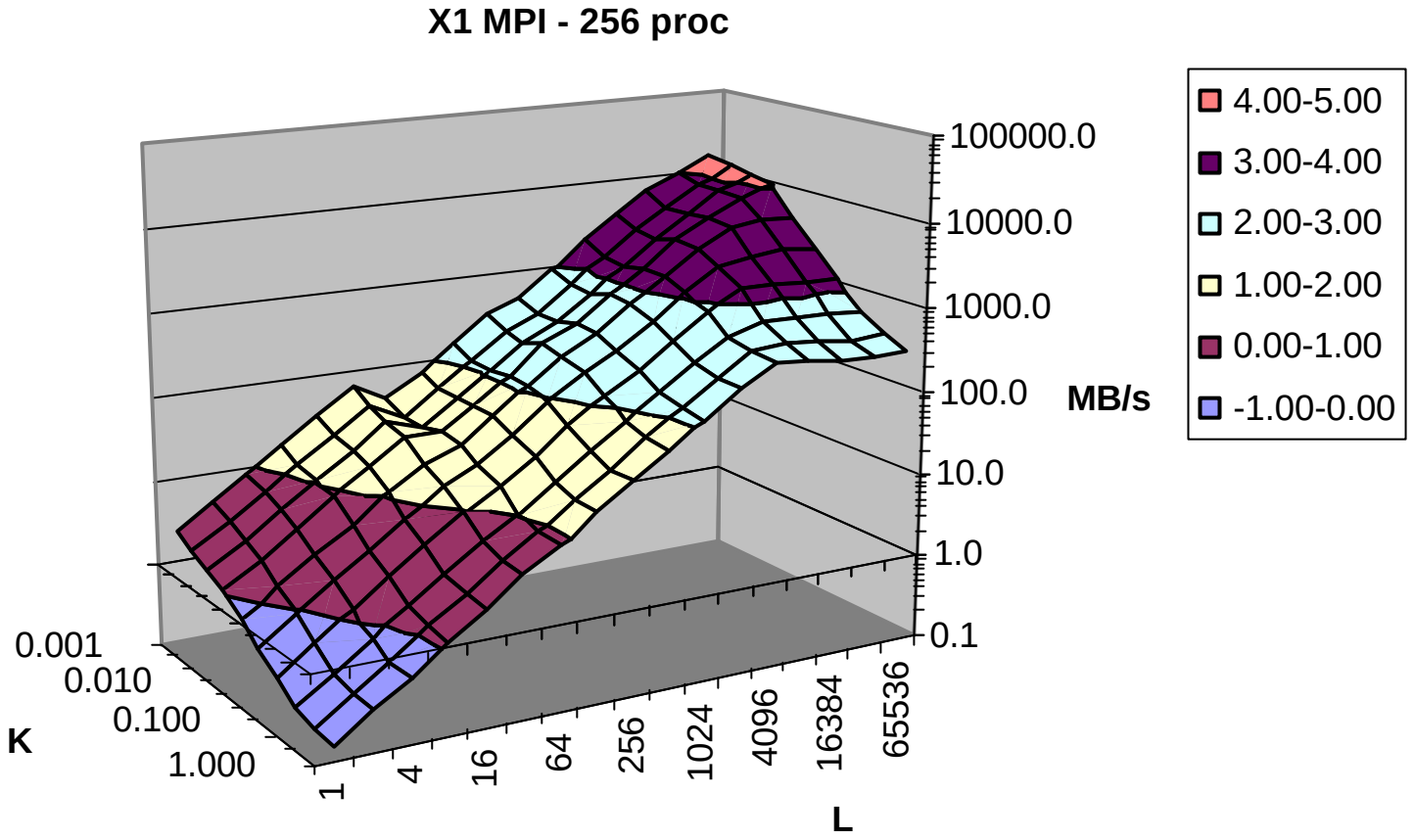
CLOCK(start)

RunningTime += end - start

MPI Problem (K=1, L=1)

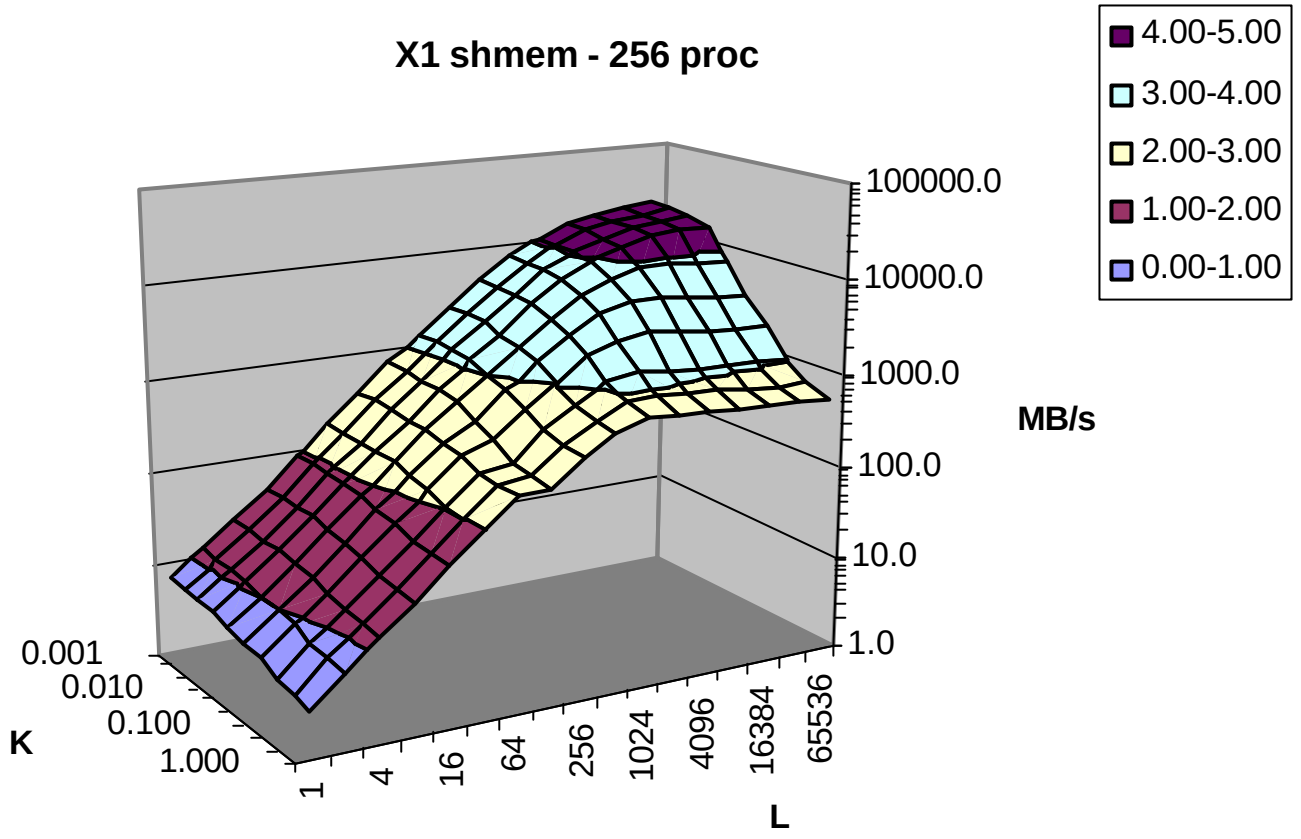


MPI Performance



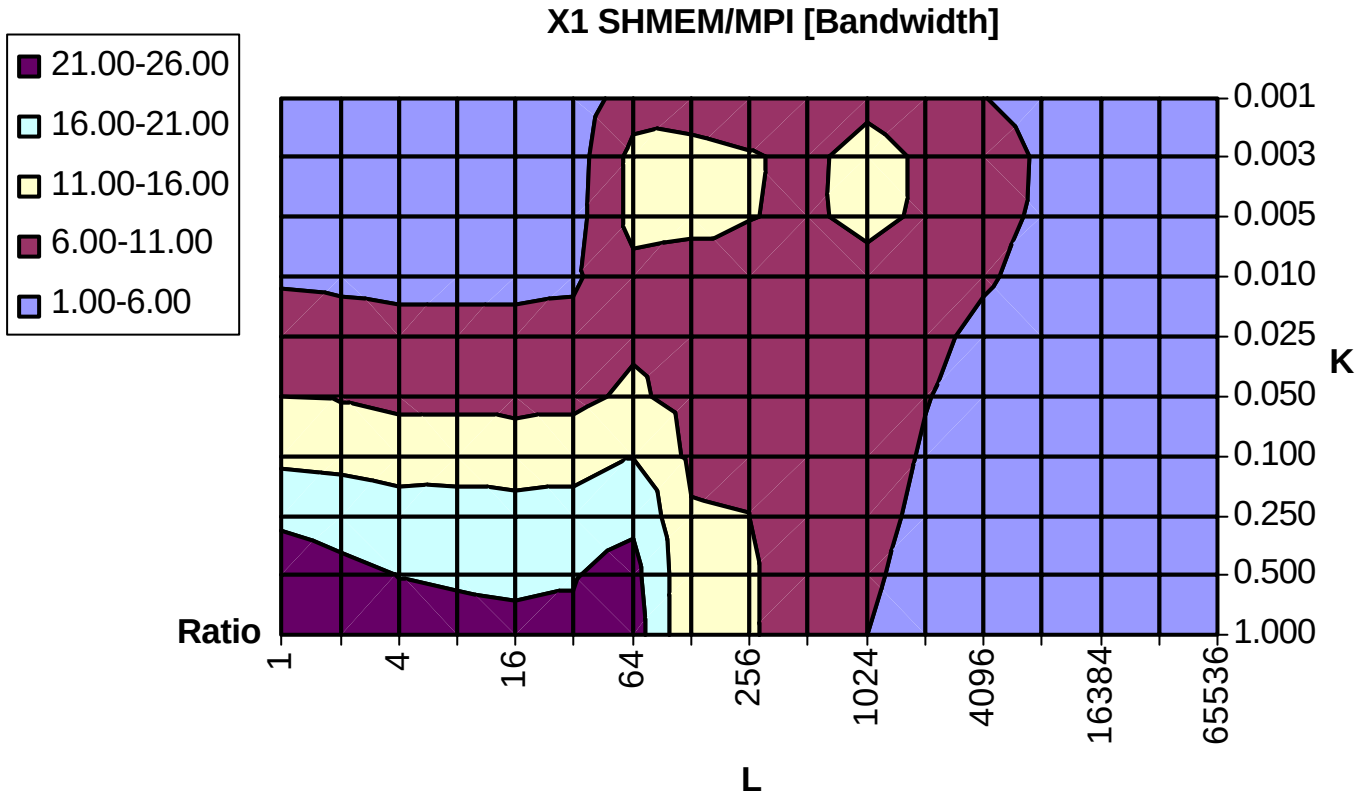
- **Both K and L affect the performance**
- **Performance drop due to MPI_Iprobe**

SHMEM Performance



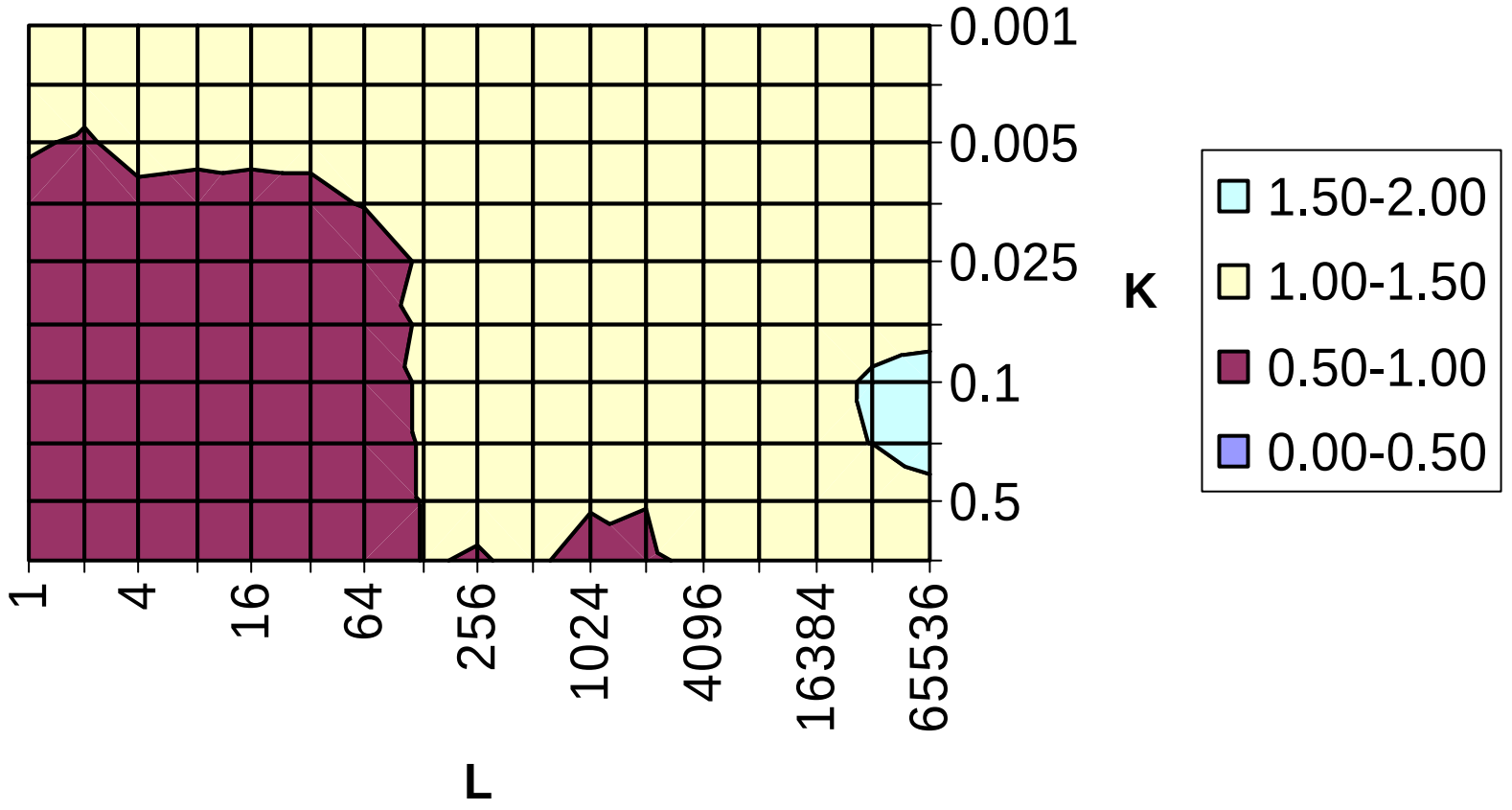
- **Better performance than MPI**
- **Effect of temporal locality becomes smaller**

SHMEM / MPI

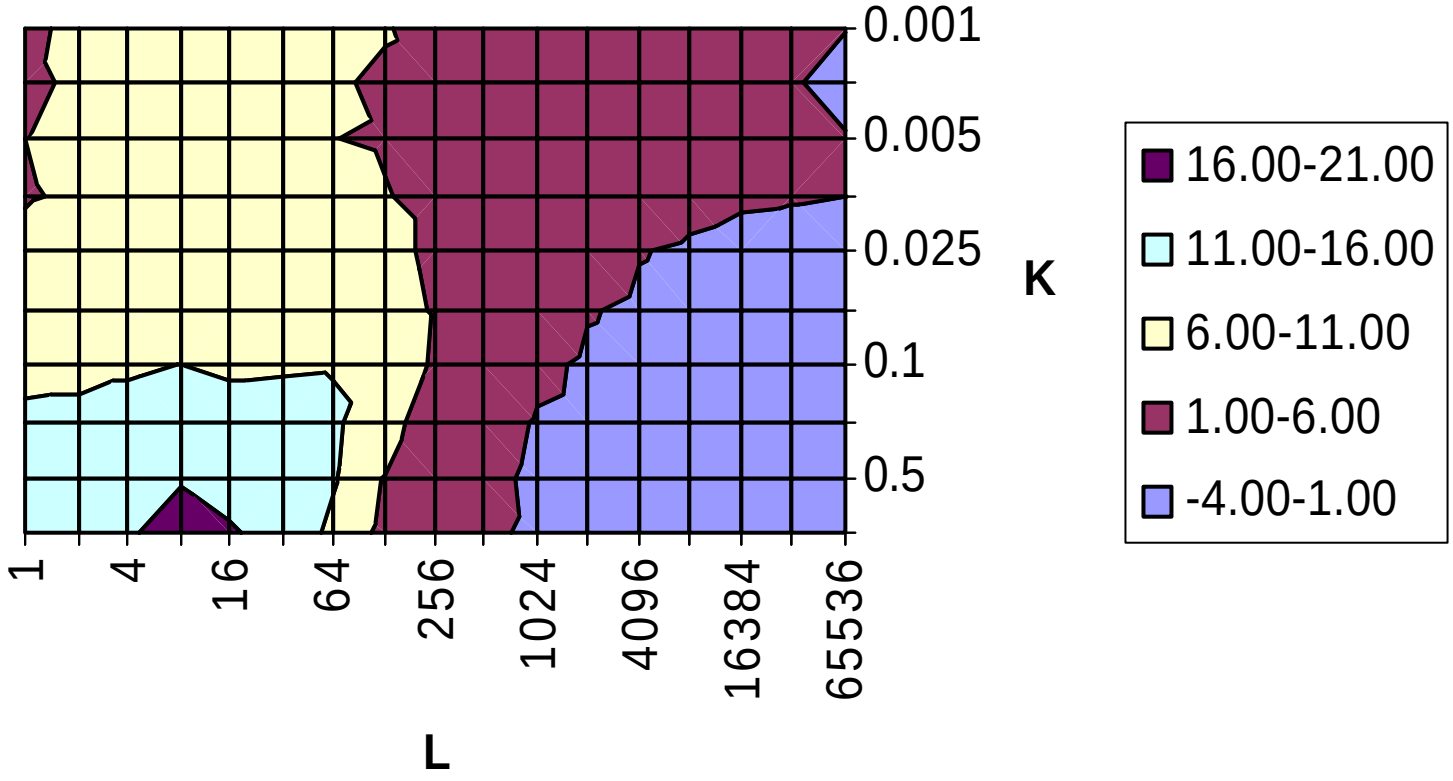


- **The advantage of SHMEM is much stronger in low temporal locality and low spatial locality area**

UPC (Method1) / SHMEM



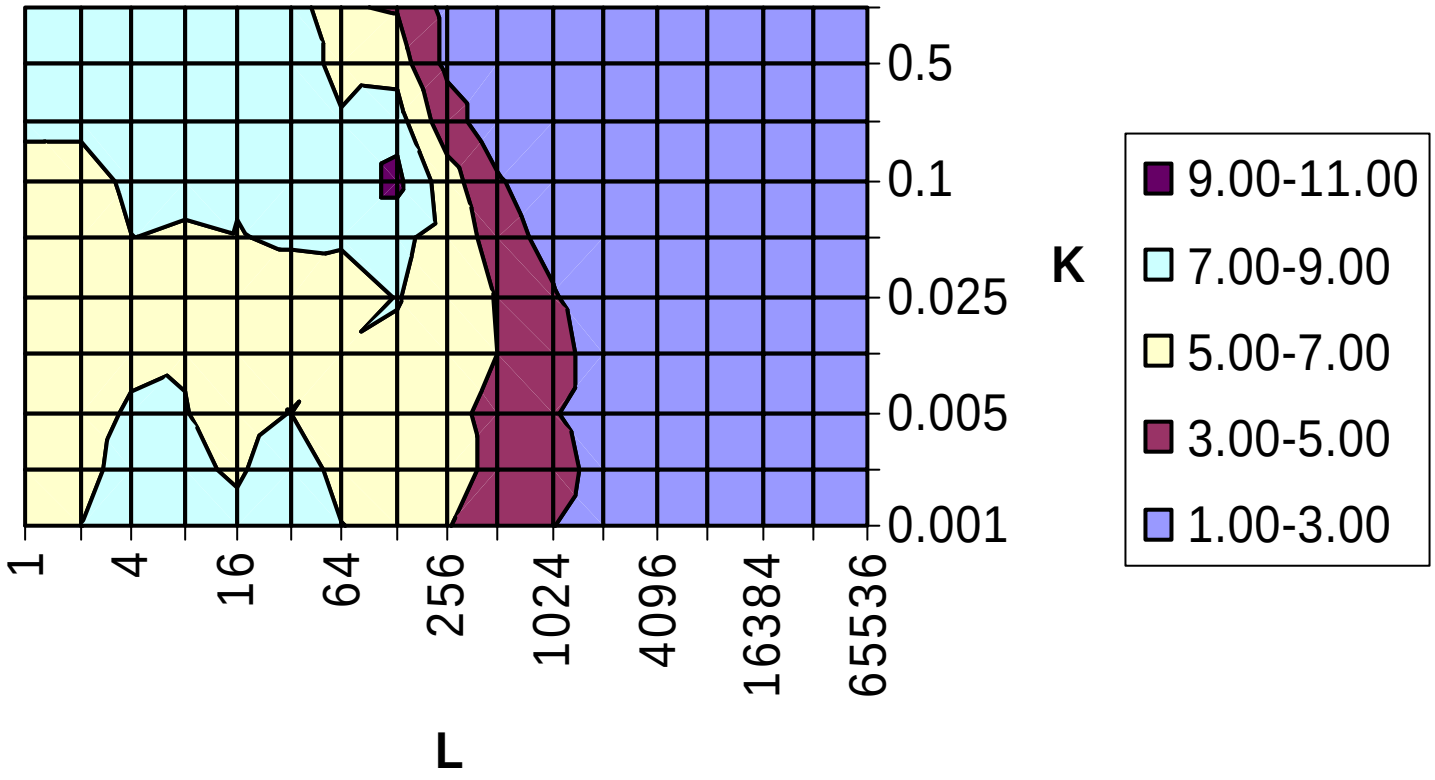
CRD UPC (Method2) / UPC (Method1)



- Block transfer is efficient for large messages
- Regular load is efficient for short messages

Office of Science
U.S. DEPARTMENT OF ENERGY

UPC / SHMEM



- UPC is better than SHMEM if we choose best implementation for UPC for all cases

Summary

- UPC delivers the best performance, followed by SHMEM, MPI is worst
- Optimal UPC implementation depends on temporal locality and spatial locality
- The current MPI Implementation is not fully optimized