Performance Implications of Communication Mechanisms
in All-Software Global Address Space Systems

Chi-Chao Chang

Dept. of Computer Science

Cornell University

Joint work with Beng-Hong Lim (IBM), Grzegorz Czajkowski and Thorsten von Eicken

Framework

- Parallel computing on clusters of workstations
- Hardware communication primitives are message-based
- Global addressing of data structures

Problem

- Tolerating high network latencies and overheads when accessing remote data

Mechanisms for tolerating latencies and overheads

- Caching: coherent data replication
- Bulk transfers: amortizes fixed cost of a single message
- Split-phase: overlaps computation with communication
- Push-based: sender-controlled communication

Objective

Global Addressing "Languages"

- DSM: cache-coherent access to shared data
  - C Region Library (CRL) [Johnson *et. al.* 95]
    - ✔ Caching
- Global pointers and arrays: explicit access to remote data
  - Split-C [Culler *et. al.* 93]
    - ✔ Bulk transfers
    - ✔ Split-phase communication
    - ✔ Push-based communication

Which of the two languages is easier to program?

Which of the two yields better performance?

- Which mechanisms are more "effective?"

Approach

Develop comparable implementations of CRL and Split-C

- Same compiler: GCC
- Common communication layer: Active Messages

Analyze the performance implications of caching, bulk, split-phase and push-based communication mechanisms

- with five applications
- on the IBM SP, Meiko CS-2, and two simulated architectures

CRL versus Split-C

CRL: Caching (regions), implicit bulk xfers, size fixed at creation

Split-C: No caching, global pointers, explicit bulk xfers, variable size

```
// CRL
rid_t r; double *x, w = 0;
if (MYPROC == 0) {
 r = rgn_create(100*8);
 x = rgn_map(r);
 for(i=0;i<100;i++) x[i] = i;
 rgn_bcast_send(&r);
} else {
 rgn_bcast_recv(&r);
 y = rgn_map(r);
 rgn_start_read(y);
 for(i=0;i<100;i++) w += y[i];
 rgn_end_read(y);
}
```

```
// Split-C
double x[100];
if (MYPROC == 0) {
 for(i=0;i<100;i++) x[i] = i;
 barrier();
} else {
 double *global y;
 double w = 0, z[100];
 barrier();
 y = toglobal(0,x);
 for(i=0;i<100;i++) w += y[i];
 bulk_read(z, y, 100*8);
}
```

CRL versus Split-C

CRL: No explicit communication

Split-C: Split-phase/push-based communication with special assignments and explicit synchronization

```
// Split-C
int i;
int *global gp;

i := *gp; // split-phase get
*gp := 5  // split-phase store
sync();   // wait until til completion
```

Hardware Platforms

Machine	*CPU*	*AM Round-trip*	*AM Bandwidth*
Meiko CS-2	40 MHz Sparc-20	25 μs	39 MB/s
IBM SP2	66 MHz RS6000/590	51 μs	34 MB/s
RMC1	66 MHz RS6000/590	17 μs	500 MB/s
RMC2	66 MHz RS6000/590	217 μs	500 MB/s

Applications

Apps	*Origin*	*Description*	*Inputs*	*Versions*	
				CRL	*SC*
MM	Split-C	C = A * B A and B block-cyclic	512x512 16x16, 128x128 blocks	1	2
FFT	Split-C	FFT butterfly algorithm	1-2 M points	1	1
LU	SPLASH/ CRL	Blocked LU Factorization	512x512 4x4, 16x16 blocks	2	3
Water	SPLASH/ CRL	N-Body System of Water Molecules	64, 512 mols	1	2
Barnes	SPLASH/ CRL	Barnes-Hut N-Body algorithm	512 bodies	1	2

Overall Observations

Some applications benefit from caching:

- MM, Barnes

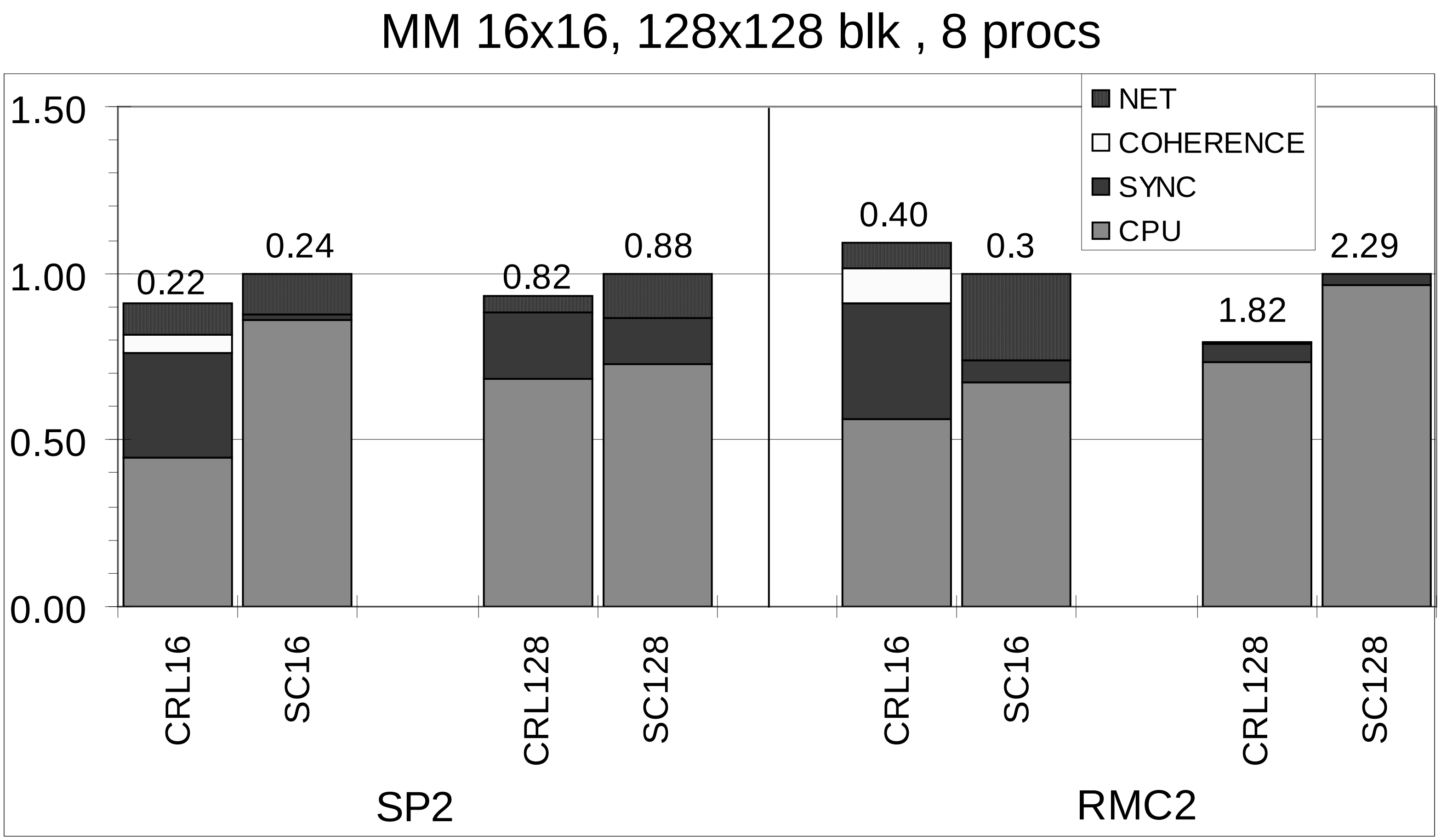
Others benefit from explicit communication:

- FFT, LU, Water

CRL and Split-C applications have similar performance

- if right mechanisms are used,
- if programmer spends comparable effort, and
- if underlying CRL and SC implementations are comparable

Sample: Matrix Multiply



Caching in CRL

Benefits applications with sufficient temporal and spatial locality

Key parameter: Region Size

- Small regions increase coherence protocol overhead
- Large regions increase communication overhead

Tuning region sizes can be difficult in many cases

- Trade-off depends on communication latency
- Regions tend to correspond to static data structures (e.g. matrix blocks, molecule structures)
- Re-designing data structures can be time consuming

Caching: Region Size

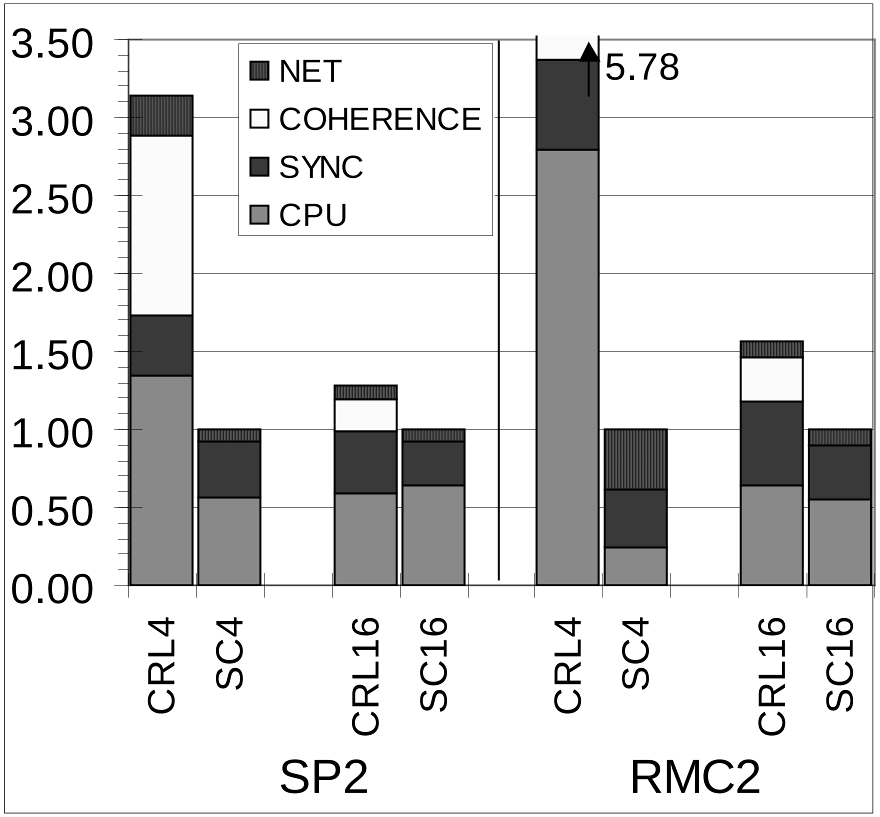
① Small regions can hurt caching, especially if latency is high

LU 4x4: CRL much slower than SC

② Large regions usually improve caching

LU 16x16: CRL closes performance gap

LU 4x4, 16x16 blk, 8 procs

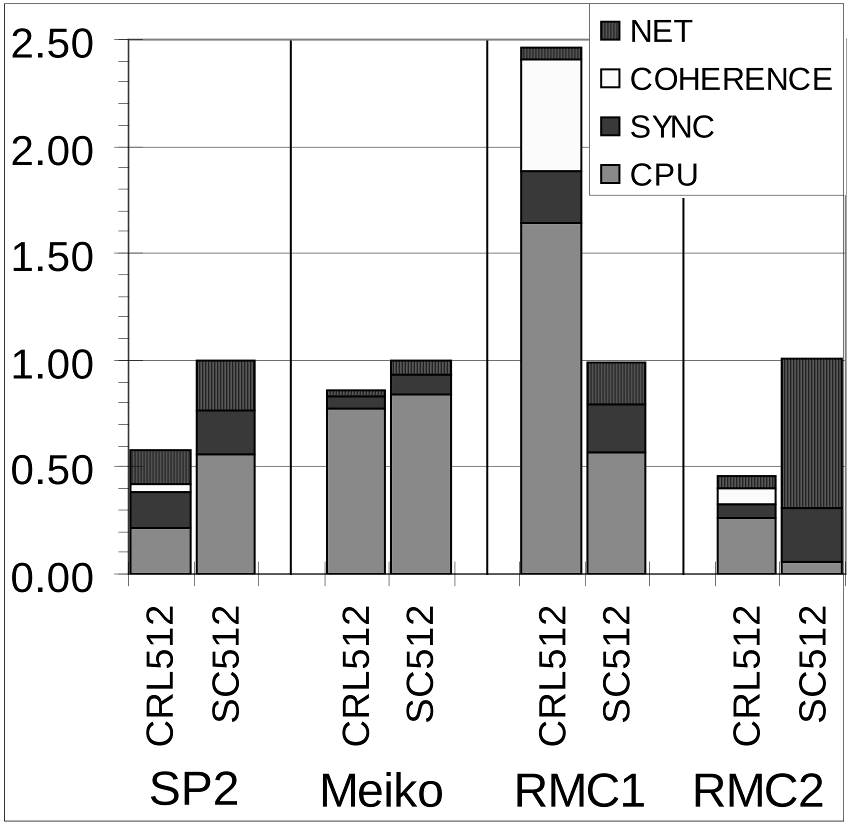


Caching: Latency

③ Advantages of caching diminish as communication latency decreases

Barnes: Split-C closes performance gap on Meiko and is faster on RMC1

Barnes 512 bds, 8 procs



Caching vs. Bulk Transfer

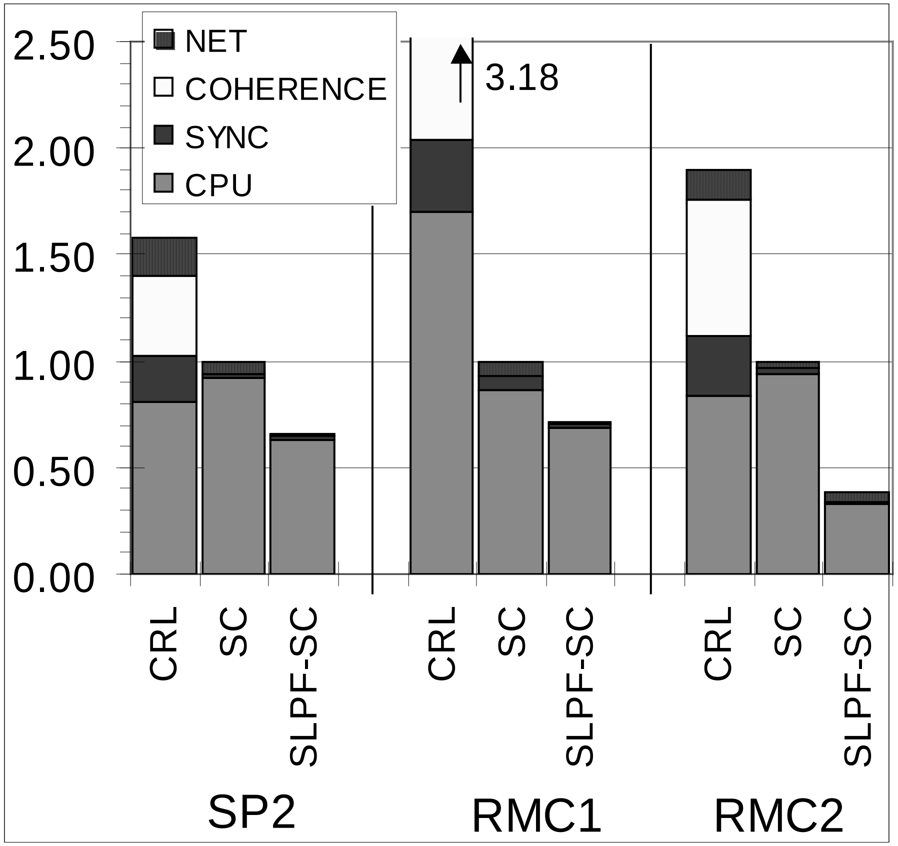
④ Large regions are harmful to caching when region size doesn't match the actual amount of data used (a.k.a. false sharing)

Water 512: CRL is much slower than SC

⑤ The ability to specify the transfer size is a plus for bulk transfers

Water 512: Selective prefetching reduces SC time substantially

Water 512 mols, 8 procs

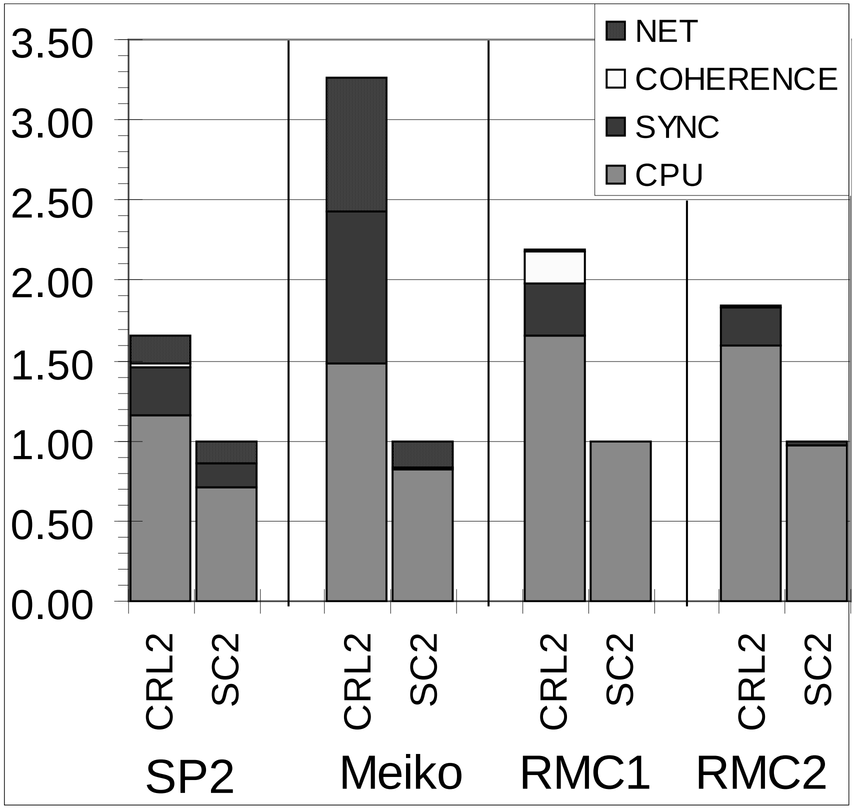


Caching vs. Bulk Transfer

- Caching harmful if lack of temporal locality

FFT: SC faster than CRL on all platforms

FFT 2M pts, 8 procs



Split-Phase and Push-Based

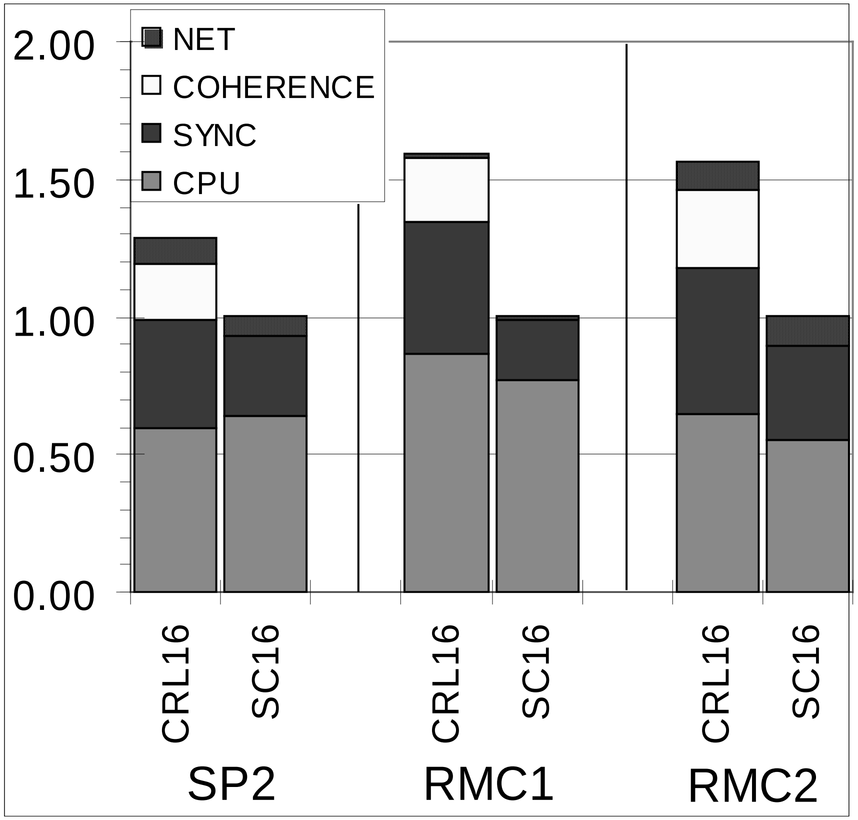
Two observations:

- Bandwidth is not a limitation
- Split-phase/Push-based allow pipelined communication phases

⑧ Split-phase/Push-based outperforms caching

LU 16x16: Base-SC is substantially faster than CRL

LU 16x16 blk, 8 procs



Related Work

- Previous research (WindTunnel, Alewife, FLASH, TreadMark) shows:
  - the benefits of explicit bulk communication with shared-memory
  - that overhead in shared-memory systems is proportional to the amount of cache/page/region misses
- Split-C shows the benefits of explicit communication without caching
- Scales and Lam demonstrate the benefits of caching and push-based communication with caching in SAM
- First study that compares and evaluates the performance of the four communication mechanisms in global address space systems

Conclusions

Split-C and CRL applications have comparable performances

- if a carefully controlled study is conducted

Programming experience: “what” versus “when”

- CRL Regions: Programmer optimizes what to transfer
- Split-C: Programmer optimizes when to transfer...
  - Pipelining communication phases with explicit synchronization
  - Managing local copies of remote data

Paper contains detailed results for:

- multiple versions of 5 applications
- running on 4 machines