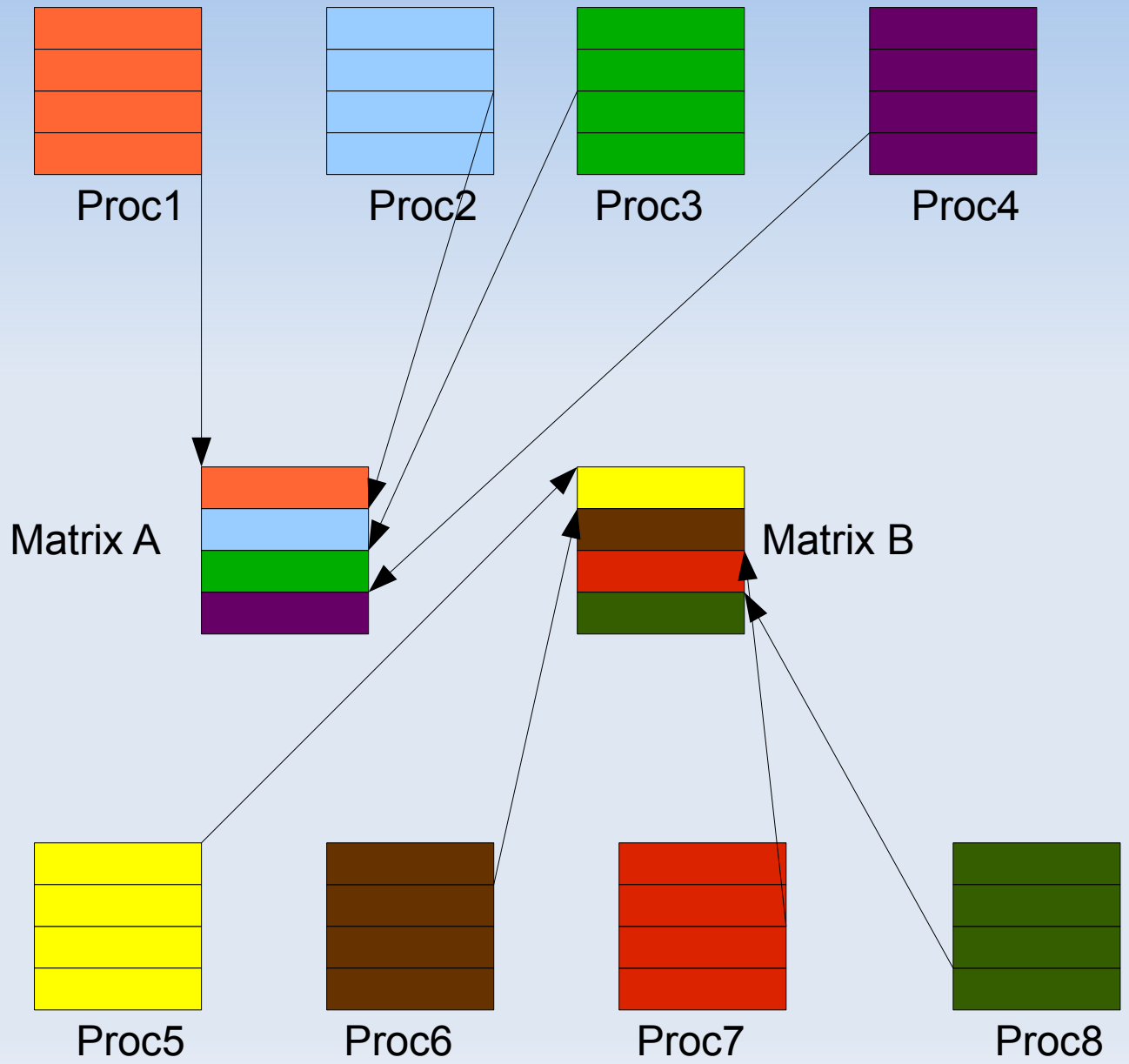
CkDirect: Charm++ RDMA Put

Presented by Eric Bohm
CkDirect Team: Eric Bohm, Sayantan Chakravorty, Pritish Jetley, Abhinav Bhatele
ppl@cs.uiuc.edu
5/4/2008

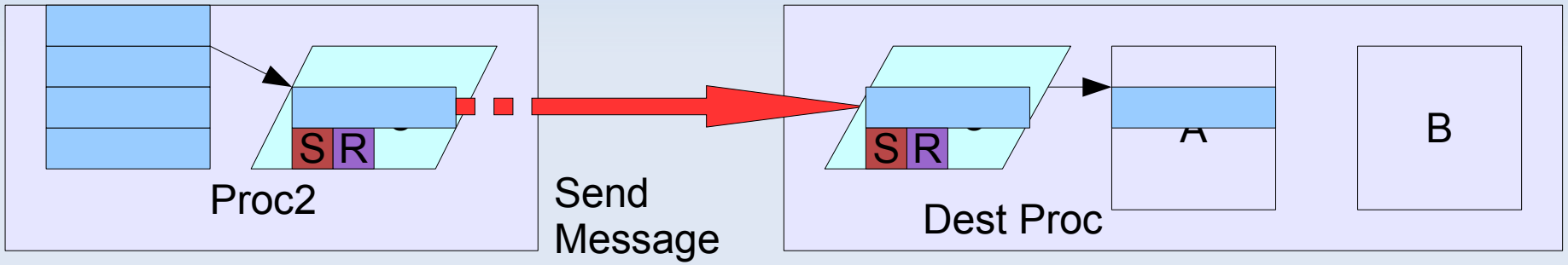
What is CkDirect?

- One-sided communication
  - One-way (put only, so far)
  - Memory to memory interface
  - Uses RDMA for zero copy
  - No protocol synchronization
  - User notification via callback
  - Pair-wise persistent channels

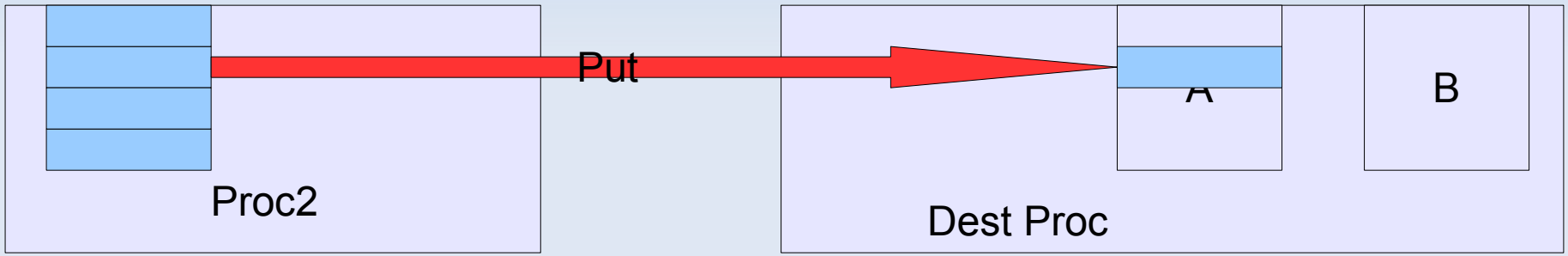
Motivating Example



Messaging Approach



CkDirect Approach



RDMA Challenges

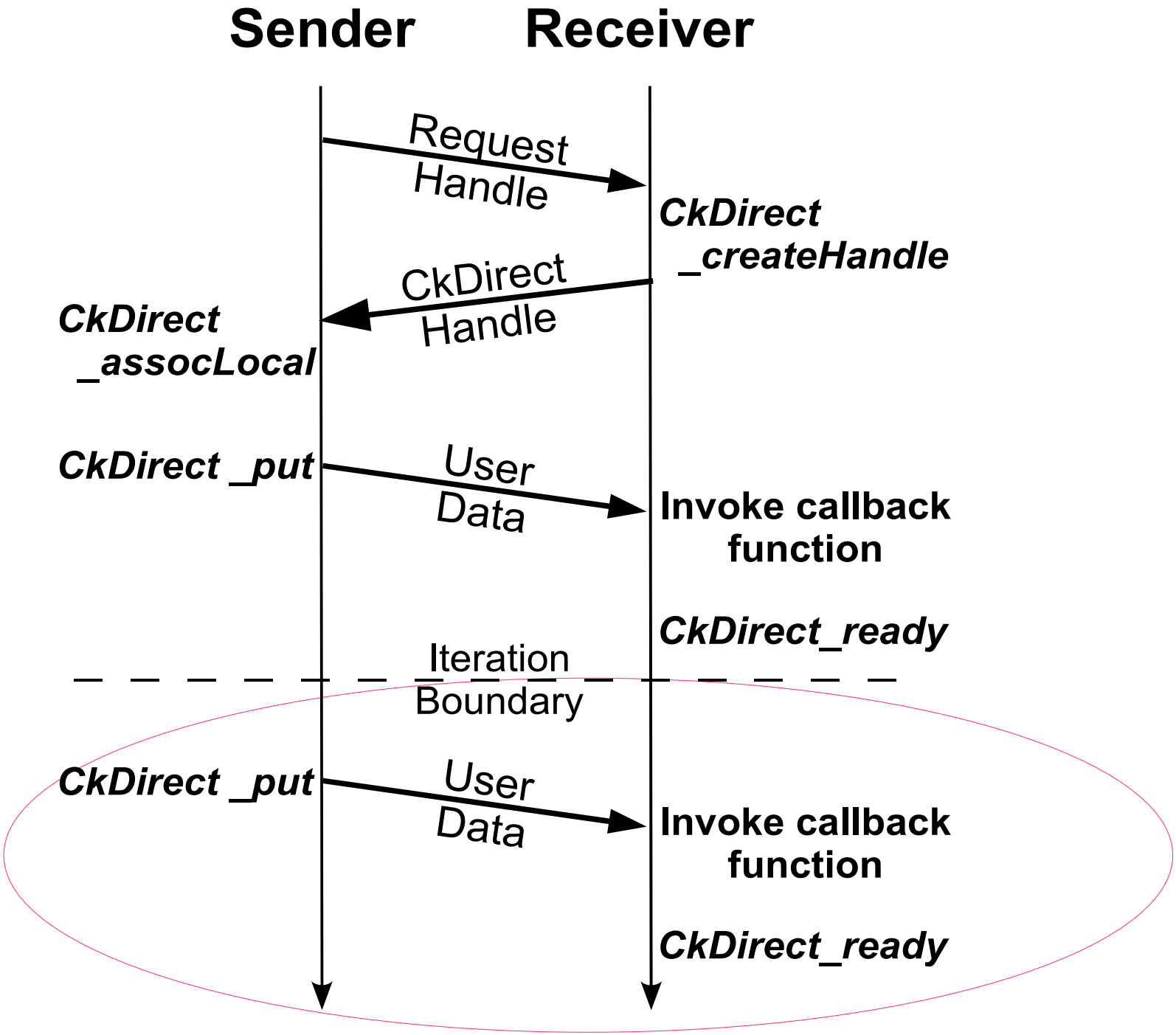
- Remote Direct Memory Access
  - Minimal overhead => fast
- Put is more intuitive for message driven model
  - Get: know remote location and remote data is ready
  - Put: know remote location
- Interfaces for RDMA vary by interconnect
- Put completion notification is lacking
    - either there is no notification
    - or the put performance is hardly better than two-sided
    - through trickery, we can do better than that.

Where is it useful?

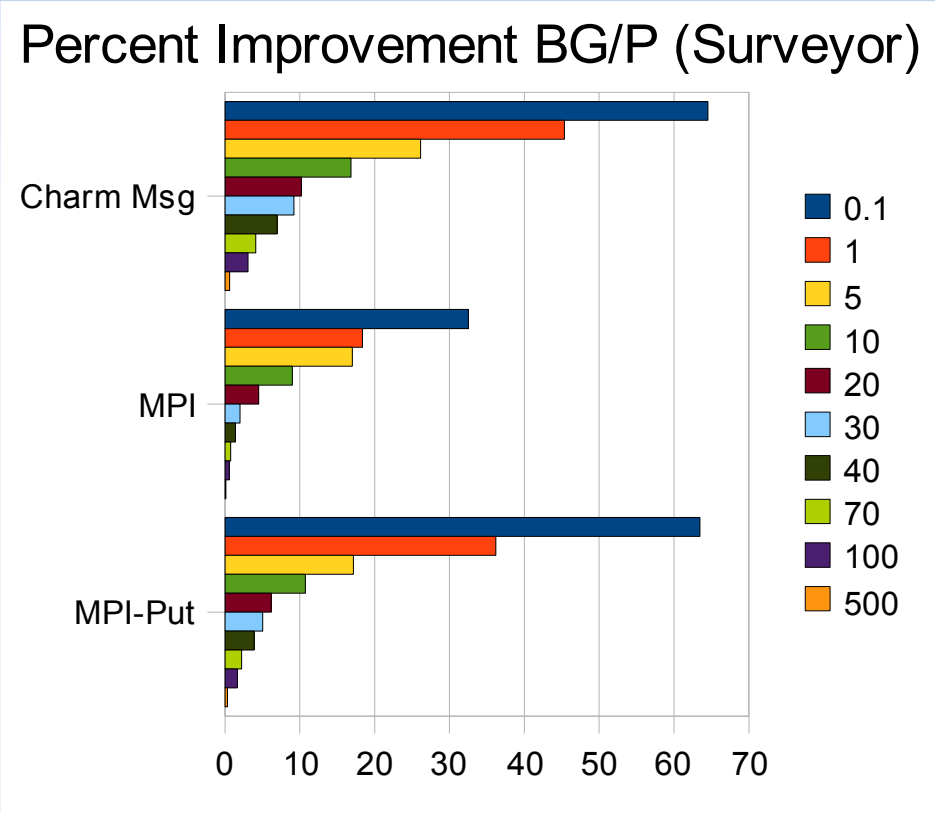
- When the same size data is transferred between the same partners each iteration to buffers which are reused
- When the application already enforces iteration boundaries
- Especially when you need to aggregate data from disparate sources into a contiguous buffer before processing

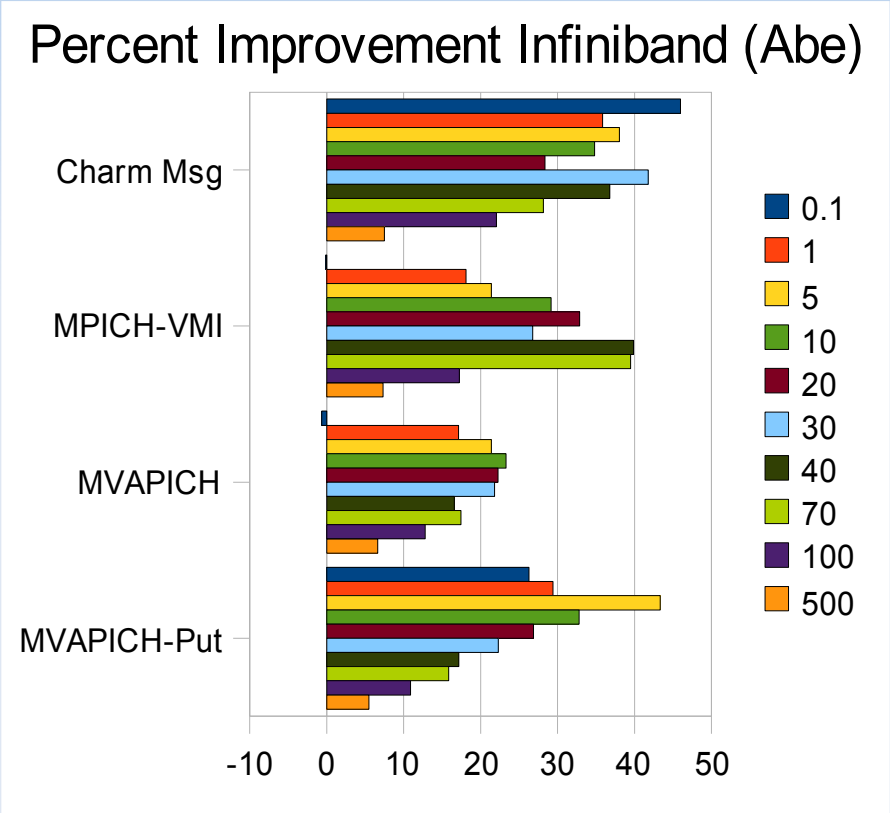
How does it work?

- User callback triggered on put completion
- Application must:
  - register send and receive processor and memory pairs in a handle
  - register put completion callback for handle
  - register out of band pattern for handle
  - call ready when done using the received put data
  - only 1 transaction per handle at a time
  - trigger message from callback for real computation


Sender
Receiver
Request Handle
CkDirect _createHandle
CkDirect Handle
CkDirect _assocLocal
CkDirect _put
User Data
Invoke callback function
CkDirect_ready
Iteration Boundary
CkDirect _put
User Data
Invoke callback function
CkDirect_ready

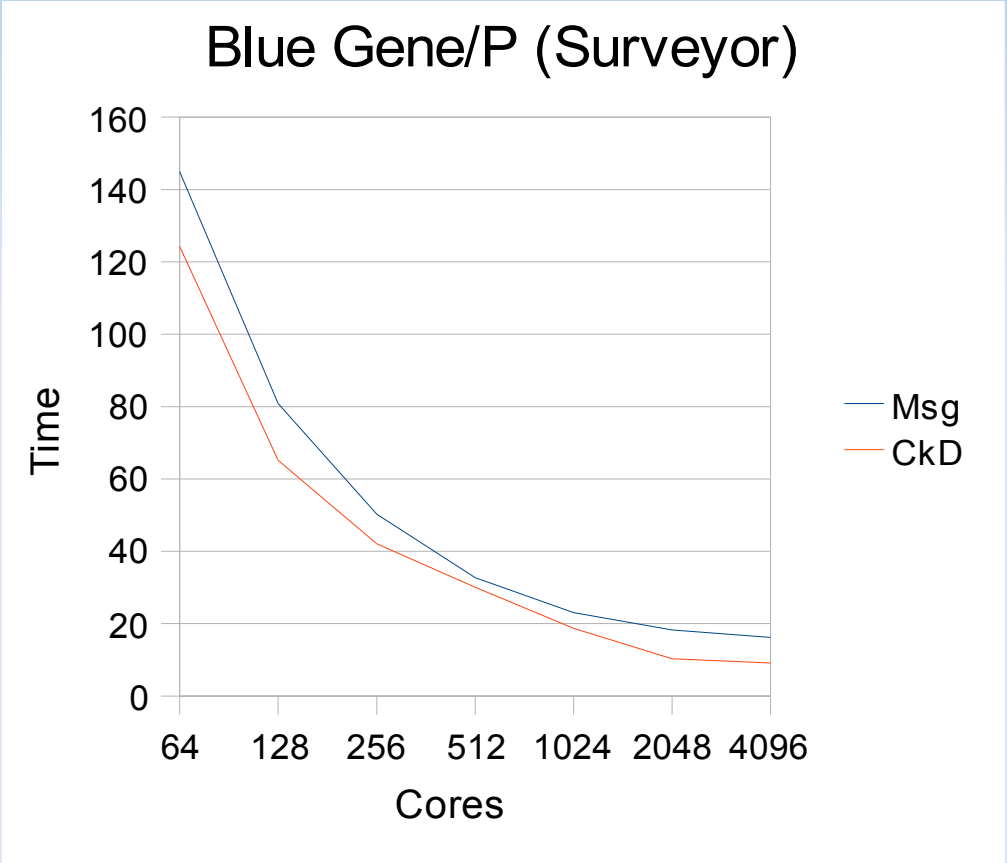
Ping Pong Results

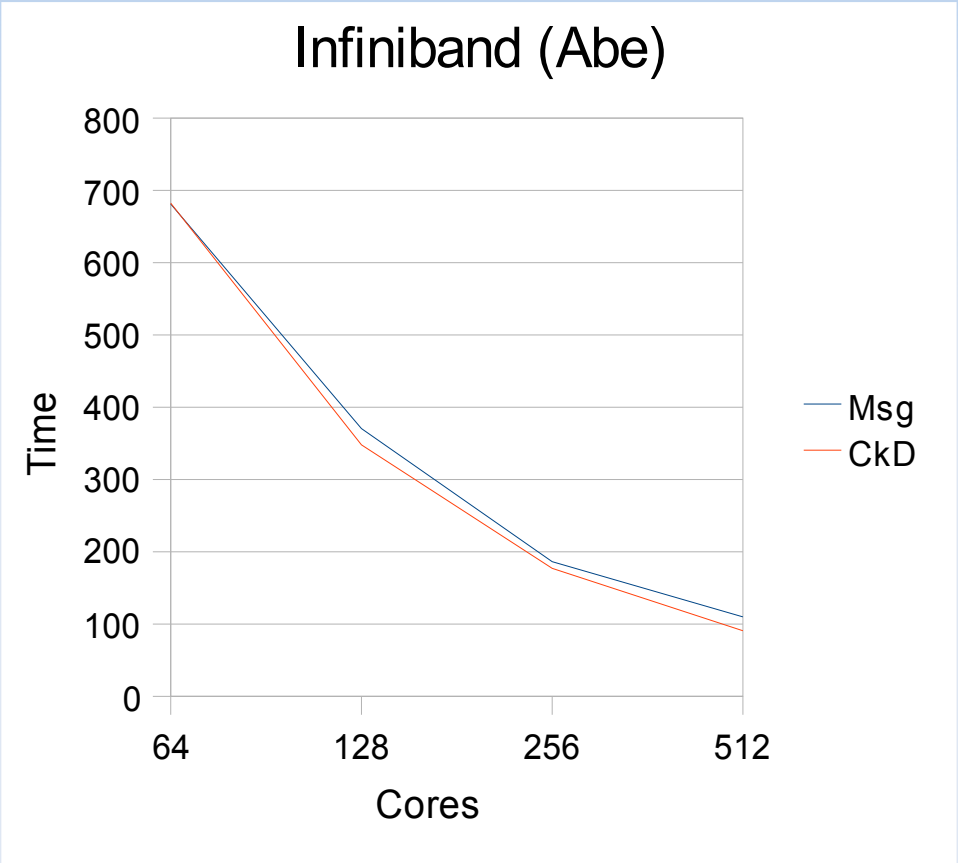




Ping Pong, CkDirect relative improvement, by message size in 1000s of bytes

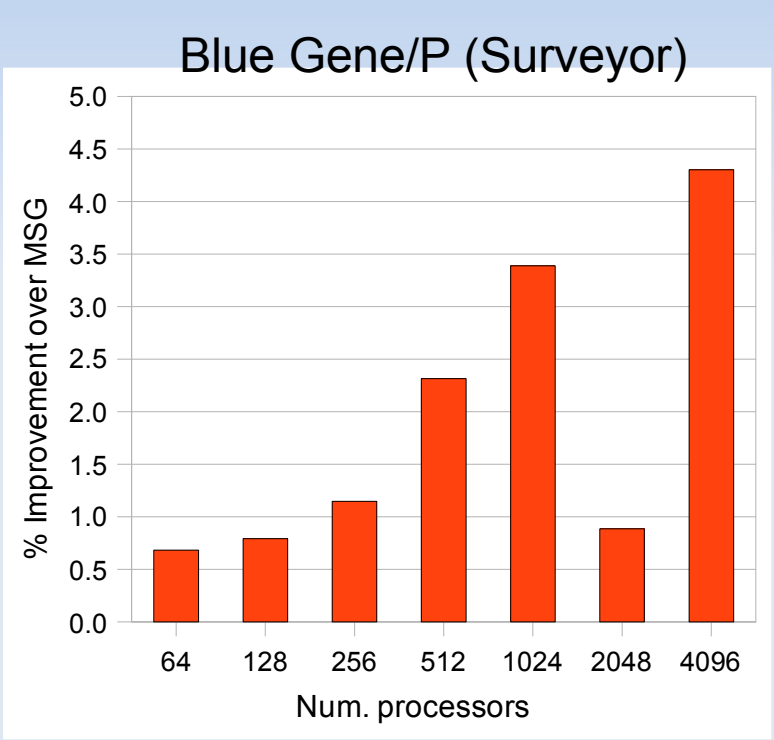
Matrix Multiply Results

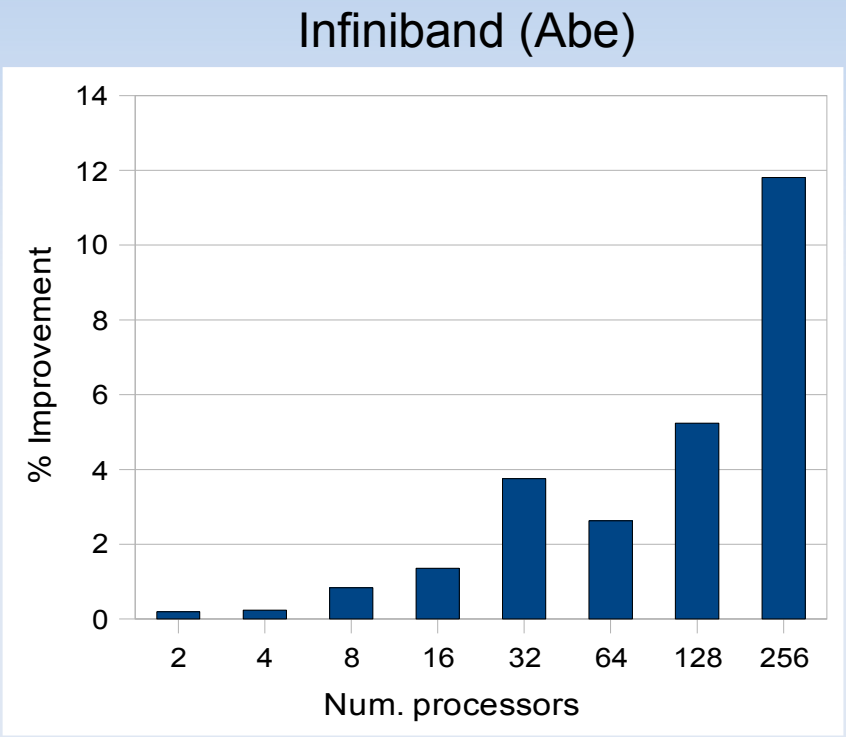




Matrix Multiply 2048*2048 average time in milliseconds

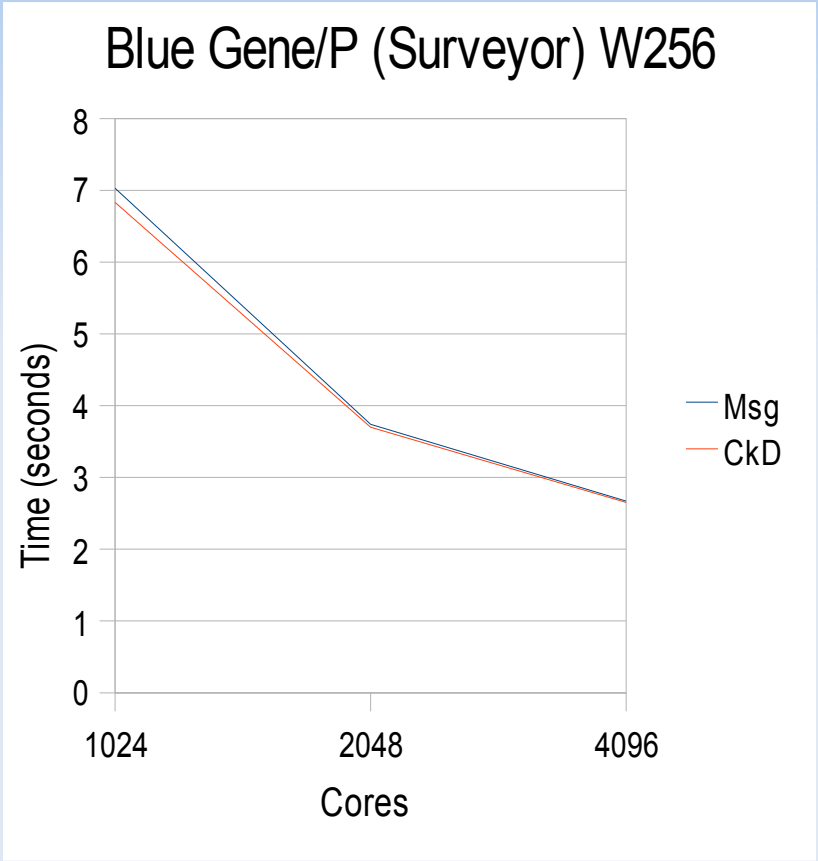
Jacobi 3D Results

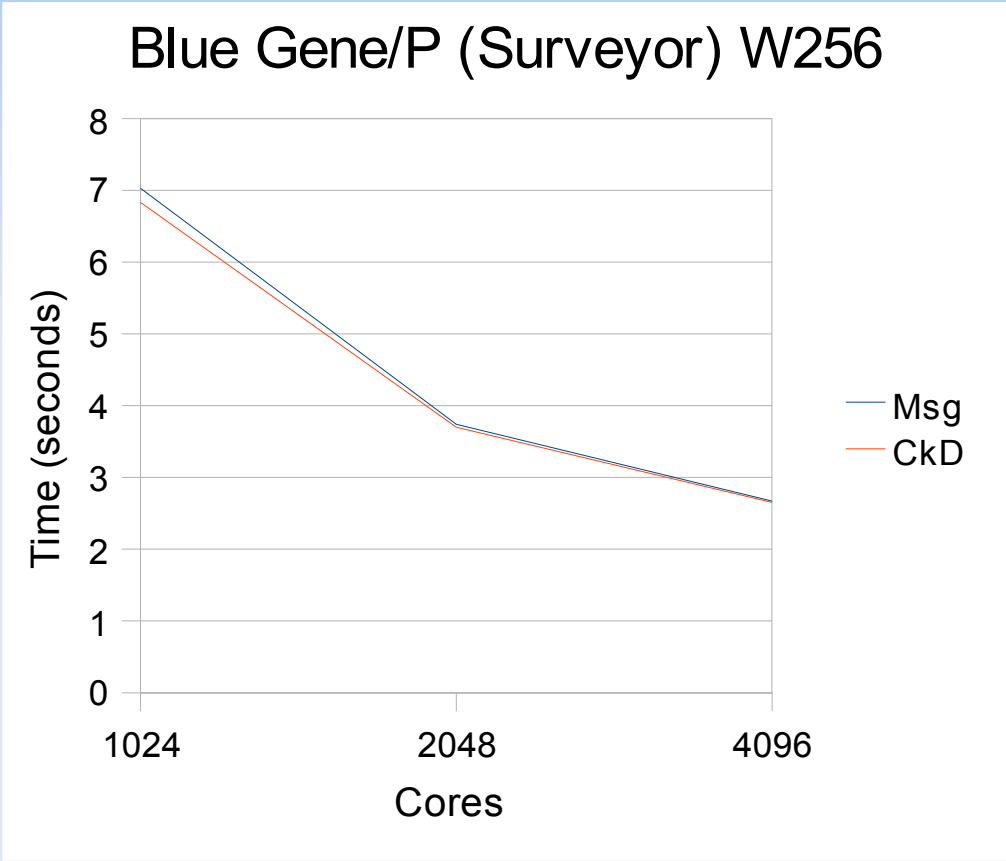




Jacobi 3D 1024*1024*512, iteration time improvement from CkDirect

OpenAtom Results

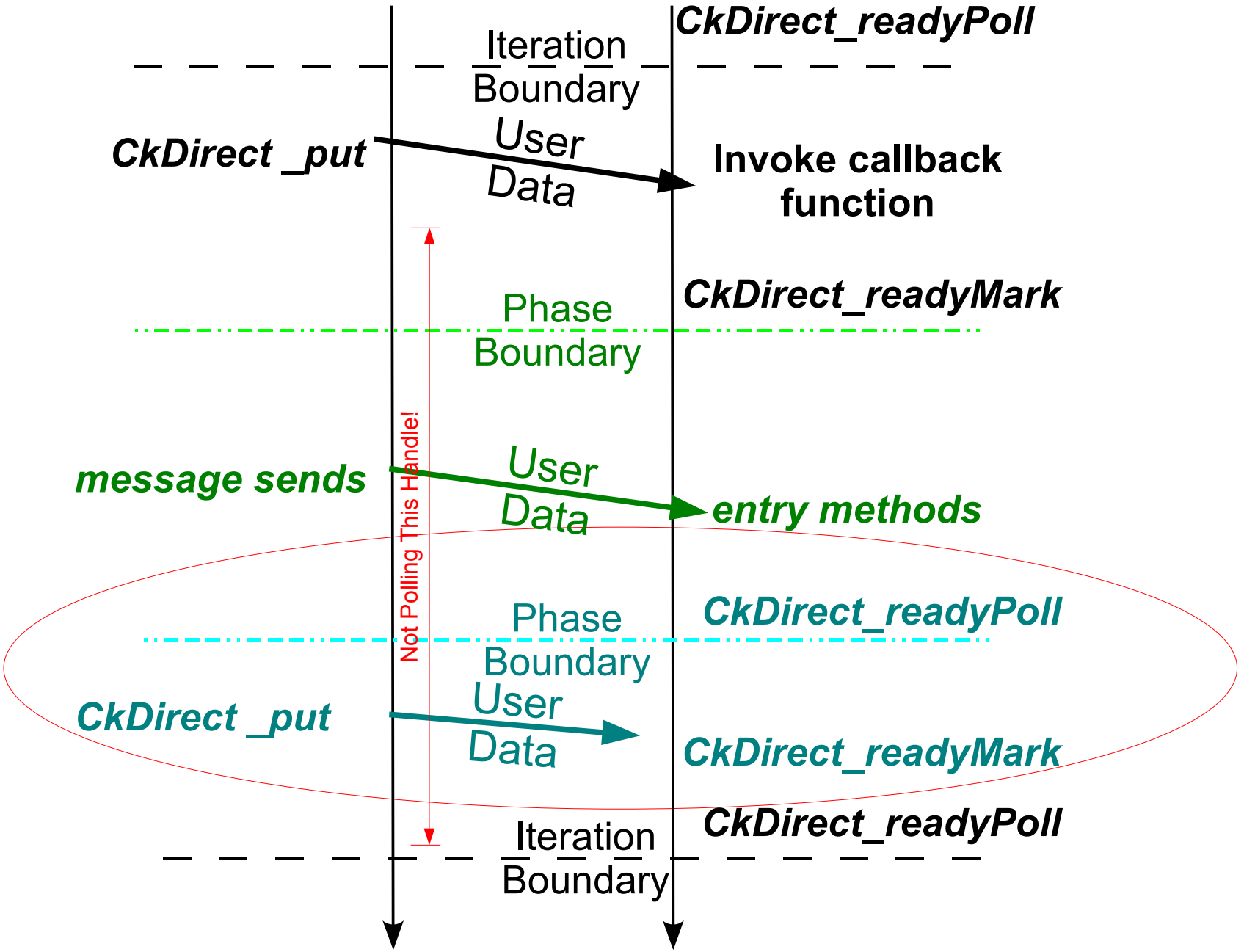




OpenAtom Water256M Benchmark, minimization, time per step in seconds

Reducing Polling Overhead

- Polling overhead is proportional to the number of ready handles.
- To minimize the number of ready handles we have a split scheme.
  - CkDirect_readyMark
    - Done with data, but don't start polling yet
  - CkDirect_readyPoll
    - Data was already marked, start checking
- Can detect puts completed since readyMark


CkDirect_readyPoll
Iteration
Boundary
CkDirect _put
User
Data
Invoke callback
function
CkDirect_readyMark
Phase
Boundary
Not Polling This Handle!
message sends
User
Data
entry methods
CkDirect_readyPoll
Phase
Boundary
CkDirect _put
User
Data
CkDirect_readyMark
CkDirect_readyPoll
Iteration
Boundary

The CkDirect API

```
/* Receiver side create handle */
struct infiDirectUserHandle CkDirect_createHandle(int senderNode,void *recvBuf, int
recvBufSize, void (*callbackFnPtr)(void *), void *callbackData,double initialValue);

/* Sender side register memory to handle */
void CkDirect_assocLocalBuffer(struct infiDirectUserHandle *userHandle,void *sendBuf,int
sendBufSize);

/* Sender side actual data transfer */
void CkDirect_put(struct infiDirectUserHandle *userHandle);

/* Receiver side done with buffer */
void CkDirect_readyMark(struct infiDirectUserHandle *userHandle);

/* Receiver side start checking for put */
void CkDirect_readyPollQ(struct infiDirectUserHandle *userHandle);

/* Receiver side done with buffer start checking for put */
void CkDirect_ready(struct infiDirectUserHandle *userHandle);
```

Conclusions

- Availability: cvs version of charm
  - net-linux-amd64-ibverbs
  - bluegenep
- Future Work
  - CkDirect multicasts
  - Ports to other architectures
- Questions?
- Feedback?