Kommunikation in HPC-Clustern
Communication/Computation Overlap in MPI

W. Rehm and T. Höfler

Department of Computer Science
TU Chemnitz

http://www.tu-chemnitz.de/informatik/RA

11.11.2005
Outline

1 Motivation

2 Current Research
   - Optimized Collective Operations
   - Non-blocking Collective Operations
   - Example
Motivation

- Optimize run-time of (highly) parallel applications
- Use powerful parallel libraries that
- Support aggressive overlap of communication and computation
- A typical library (e.g. ScaLapack) is based on MPI/BLACS
Overlap is enabled by **non-blocking MPI** operations.

Until now only non-blocking Point-to-Point operations could take advantage of the special HW support of the underlying network.

**Non-blocking** MPI **collectives** could take advantage of the special HW support of the underlying network.
Outline

1. Motivation

2. Current Research
   - Optimized Collective Operations
   - Non-blocking Collective Operations
   - Example
Our Current Research

- Tuning collective communication routines e.g. optimized MPI_Barrier()
- Targeting at special support of the underlying network (InfiniBand’s multicast, atomic ops, RDMA)
- Implementing Non-blocking Collectives within the framework of Open MPI (”coll” component)
The LogP model abstracts for giving a rough communication performance estimation.

W. Rehm and T. Höfler  Communication/Computation Overlap in MPI
An Optimized InfiniBand Barrier

But 1:P-P:1 benchmark reveals an implicit parallelism for InfiniBand

- ping
- pong

W. Rehm and T. Höfler
Communication/Computation Overlap in MPI
An Optimized InfiniBand Barrier

- Thus, $f$ is introduced (the first $f$ small messages are essentially for free)
  - T. Hoefler et. al.: LogfP - A Model for small Messages in InfiniBand
- For the first $f$ messages it seems that there are $f$ ports - instead of one
- All current algorithms have been modeled and optimized for single-port nodes
- The Dissemination Barrier Algorithm works best for single ported nodes (proven with LogP)
- The idea came up to create an n-way Dissemination Barrier
An Optimized InfiniBand Barrier

The Dissemination Barrier Algorithm

- run-time $O(\log_2 P)$
An Optimized InfiniBand Barrier

- The n-way Dissemination Barrier Algorithm

- run-time $O(\log_{n+1} P)$
  - T. Hoefler et. al.: An optimal Synchronization Algorithm for Multi-Port Networks
  - T. Hoefler et. al.: Fast Barrier Synchronization for InfiniBand
Outline

1 Motivation

2 Current Research
   - Optimized Collective Operations
   - Non-blocking Collective Operations
   - Example
Helping to Answer

- Overlapping Comm/Comp programming style (threads) too complex
- What is the tradeoff of performance and programming effort?

  J.B. White: "...programming for overlap may be of little benefit for synchronous applications in general. Many current MPI implementations - all those tested here - do not support the required level of overlap. The additional development effort and code complexity required to program for overlap seems unjustified..."

- We contribute to answer by implementing non-blocking collectives in the context of a real application (Abinit)!
Its not just Performance - Why Collectives?

- Gorlach, ’04: ”Send-Receive Considered Harmful”
- ⇔ Dijkstra, ’68: ”Go To Statement Considered Harmful”

point to point:

```java
if ( rank == 0) then
    call MPI_SEND(...)
else
    call MPI_RECV(...)
end if
```

vs. collective:

```java
call MPI_GATHER(...)```

cmp. math libraries vs. loops
Putting Everything Together

- non blocking collectives?
- JoD mentions “split collectives”
- example:
  - MPI_Bcast_begin(…)
  - MPI_Bcast_end(…)
- no nesting with other collectives
- very limited
- not in the MPI-2 standard
- votes: 11 yes, 12 no, 2 abstain
Why non blocking Collectives

- many collectives synchronize unnecessarily
- scale with $O(\log_2 P)$ sends
- wasted CPU time: $\log_2 P \cdot 2L$
  - Fast Ethernet: $L = 50-60$
  - Gigabit Ethernet: $L = 15-20$
  - InfiniBand: $L = 2-7$
  - $1\mu s \approx 4000$ FLOPs on a 2GHz Machine
Outline

1 Motivation

2 Current Research
   - Optimized Collective Operations
   - Non-blocking Collective Operations
   - Example
Pseudocode Overlap

do ....
    ! send communication buffer
    MPI_Isend(buffer_comm, ..., reg, ...)
    ! do useful work
    do_work(buffer_work)
    ! finish communication
    MPI_Wait(req, ...)
    ! swap the buffers
    buffer_tmp = buffer_comm
    buffer_comm = buffer_work
    buffer_work = buffer_tmp
endo
Linpack Problem

- solves $Ax = b$ with $A = PLU$ decomposition
- row partial pivoting
- $A$ and $b$ are given
- Gauss Algorithm for $A$
- $O(n^3)$ FP operations
- applied to $b$ (no need to store $P$ or $L$)
- solution by backwards substitution of $U$
- our example assumes column-wise distribution
- block distribution more complicated
Example - Linpack Problem - traditional

```
do k=1,n,1 ! loop over columns
  if (IHaveColumn(k)) max = findMaxColumn(k)
  MPI_Bcast(max);
  exchangeRow(max,k)
do i=k+1,n,1 ! loop over rows
  if (IHaveColumn(k)) div = A(i+1,k) / A(k,k)
  MPI_Bcast(div)
do j=k+1,n,1 ! essentially DAXPY
    A(i,j) = A(i,j) - div * A(k,j)
  enddo
  b(i) = b(i) - div * b(k)
endo
```

W. Rehm and T. Höfler

Communication/Computation Overlap in MPI
Example - Linpack Problem - non-blocking Collectives

do k=1,n,1 ! loop over columns
    if (IHaveColumn(k+1)) nextmax = findMaxColumn(k+1)
    MPI_Ibcast(nextmax, h1);
    exchangeRow(max, k)
    do i=k+1,n/p,1 ! loop over rows
        if (IHaveColumn(k)) nextdiv = A(i+1,k) / A(k,k)
        MPI_Ibcast(nextdiv, h2)
        do j=k+1,n,1 ! essentially DAXPY
            A(i,j) = A(i,j) - div * A(k,j)
        enddo
        b(i) = b(i) - div * b(k)
        MPI_Wait(h2); div = nextdiv
    enddo
    MPI_Wait(h1); max = nextmax
enddo