Fast and Scalable Startup of MPI Programs in InfiniBand Clusters

W. Yu, J. Wu, D.K. Panda

Dept of Computer Sci. and Engineering
The Ohio State University
{yuw,wuj,panda}@cse.ohio-state.edu

Presentation Outline

- Background
- Startup of MPI programs over IBA
- Designing Scalable Startup Schemes
- Performance Evaluation
- Conclusions and Future Work

Background

- Cluster-based parallel computing
  - Using MPI as the *de facto* Standard in HPC
  - Evolved into thousands, tens of thousands processors
  - Focus on high performance message passing
  - Fast and scalable startup is also needed

Startup of MPI Programs

- Process Initiation
  - Processes initiated across the cluster
  - Using iterative rsh/ssh
  - Or PBS, MPD, among many others
- Connection Setup
  - Initially not connected, not even knowing how to connect between each other
  - Need to set up Peer-to-Peer connection
  - Need involvement of a third-party

InfiniBand

- A high performance interconnect
  - A switch fabric to aggregate bandwidth and connect nodes with HCA and HTA
  - Blend well with latest trends in HPC
  - Deliver low latency and over 10Gbps Bandwidth
- An emerging industry standard
  - Data-Center
  - Higher performance computing
  - As more IBA clusters being delivered, more parallel programs use MPI over InfiniBand

MVAPICH Software Distribution

- Based on MPICH and MVICH
- Open Source (current version is 0.9.4)
- Have been directly downloaded by more than 150 organizations and industry
- Available in the software stack distributions of IBA vendors

National Labs/Research Centers

Argonne National Laboratory
Cornell Theory Center
Center for Mathematics and Computer Science (The Netherlands)
Inst. for Experimental Physics (Germany)
Inst. for Program Structures and Data Organization (Germany)
Lawrence Berkeley National Laboratory
Los Alamos National Laboratory
Max Planck Institute for Astronomy (Germany)
NASA Ames Research Center
NCSA
National Center for Atmospheric Research
Ohio Supercomputer Center
Pacific Northwest National Laboratory
Pittsburgh Supercomputing Center
Research & Development Institute Kvant (Russia)
Science Applications International Corporation
Sandia National Laboratory

Universities

Georgia Tech
Indiana University
Korea Univ. (Korea)
Korea Inst. Of Science and Tech. (Korea)
Kyushu Univ. (Japan)
Mississippi State University
Moscow State University (Russia)
Northeastern University
Penn State University
Russian Academy of Sciences (Russia)
Stanford University
Technion (Israel)
Technical Univ. of Munchen (Germany)
Technical Univ. of Chemnitz (Germany)
Univ. of Geneva (Switzerland)
Univ. of Houston
Univ. of Karlsruhe (Germany)
Univ. of Massachusetts Lowell
Univ. of Paderborn (Germany)
Univ. of Potsdam (Germany)
Univ. of Rio Grande (Brazil)
Univ. of Sherbrooke (Canada)
Univ. of Stuttgart (Germany)
Univ. of Toronto (Canada)

MVAPICH Users (Cont'd)

Industry

Abba Technology
Advanced Clustering Tech.
AMD
Ammasso
Appro
Array Systems Comp. (Canada)
Atipa Technologies
Agilent Technologies
Clustars Supercomputing-Technology Inc. (China)
Clustervision (Netherlands)
Compusys (UK)
CSS Laboratories, Inc.
Dell
Delta Computer (Germany)
Emplics (Germany)
Fluent Inc.
ExaNet (Israel)
GraphStream, Inc.
HP
HP (France)

IBM
IBM (France)
IBM (Germany)
INTERSED (France)
InfiniCon
Intel
Intel (China)
Intel (Germany)
Intel Solution Services (Hong Kong)
Intel Solution Services (Japan)
JNI
Kraftway (Russia)
Langchao (China)
Linux Networx
Linvision (Netherlands)
Megaware (Germany)
Mercury Computer Systems
Mellanox Technologies
Meiosys (France)
Microway, Inc.
NEC (Japan)
NEC Solutions, Inc.
NEC (Singapore)
NICEVT (Russia)
OCF plc (United Kingdom)

OctigaBay (Canada)
PANTA Systems
ParTec (Germany)
PathScale, Inc.
Pultec (Japan)
Pyramid Computer (Germany)
Qlusters (Israel)
Raytheon Inc.
RLX Technologies
Rosta Ltd. (Russia)
SBC Technologies, Inc.
Scyld Software
SGI (Silicon Graphics, Inc.)
SKY Computers
Streamline Computing (UK)
Systran
Tomen
Telcordia Applied Research
Thales Underwater Systems (UK)
Transtec (Germany)
T-Platforms (Russia)
Topspin
Unisys
Voltaire
WorkstationsUK, Ltd. (UK)
Woven Systems, Inc.

Presentation Outline

- Background
- Startup of MPI programs over IBA
- Designing Scalable Startup Schemes
- Performance Evaluation
- Conclusions and Future Work

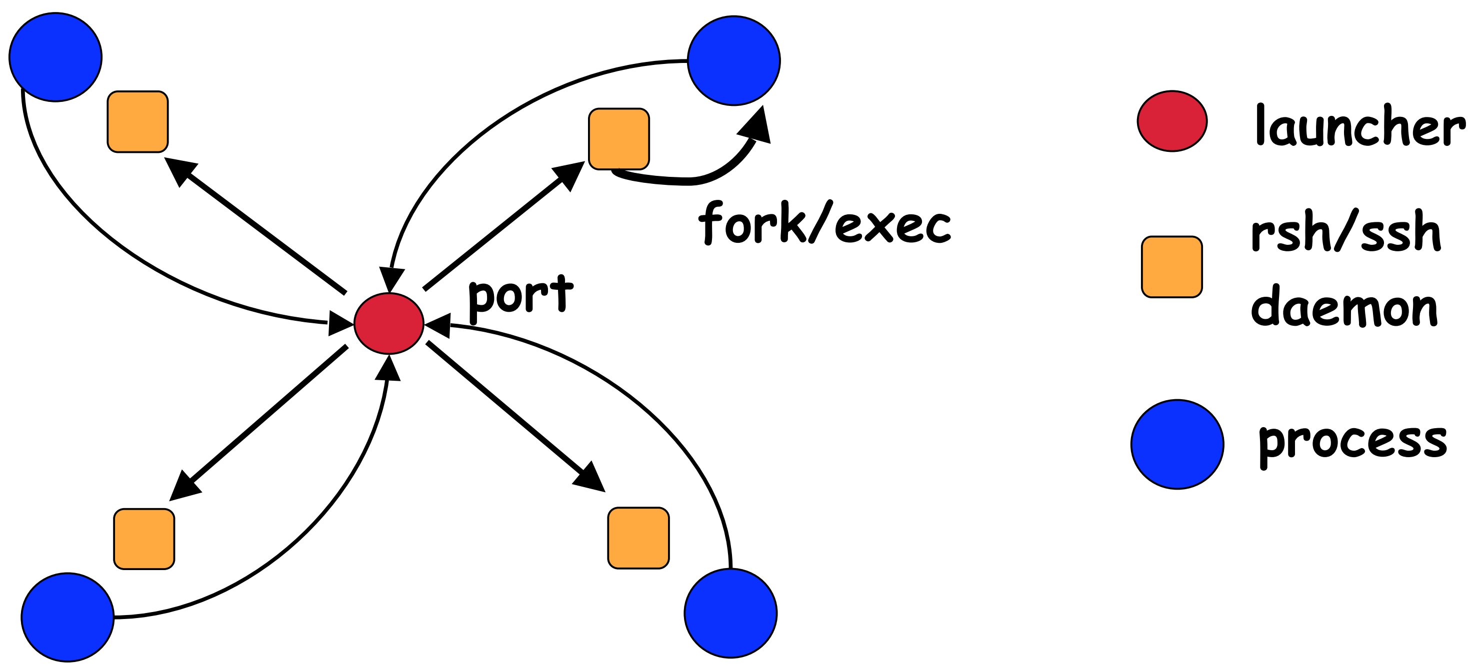
Connection Model over InfiniBand

- Four types of connections
  - Reliable Connection
  - Reliable Datagram (RD)
  - Unreliable Connection (UC)
  - Unreliable Datagram (UD)
- Connection Model used in MPI parallel programs
  - On-demand Dynamic Connection with IB Connection Management support
  - Static connection model:
    - have all processes fully connected before message passing MPI communication
    - Reliable Connection typically used for its performance, e.g., MVAPICH

Reliable Connection

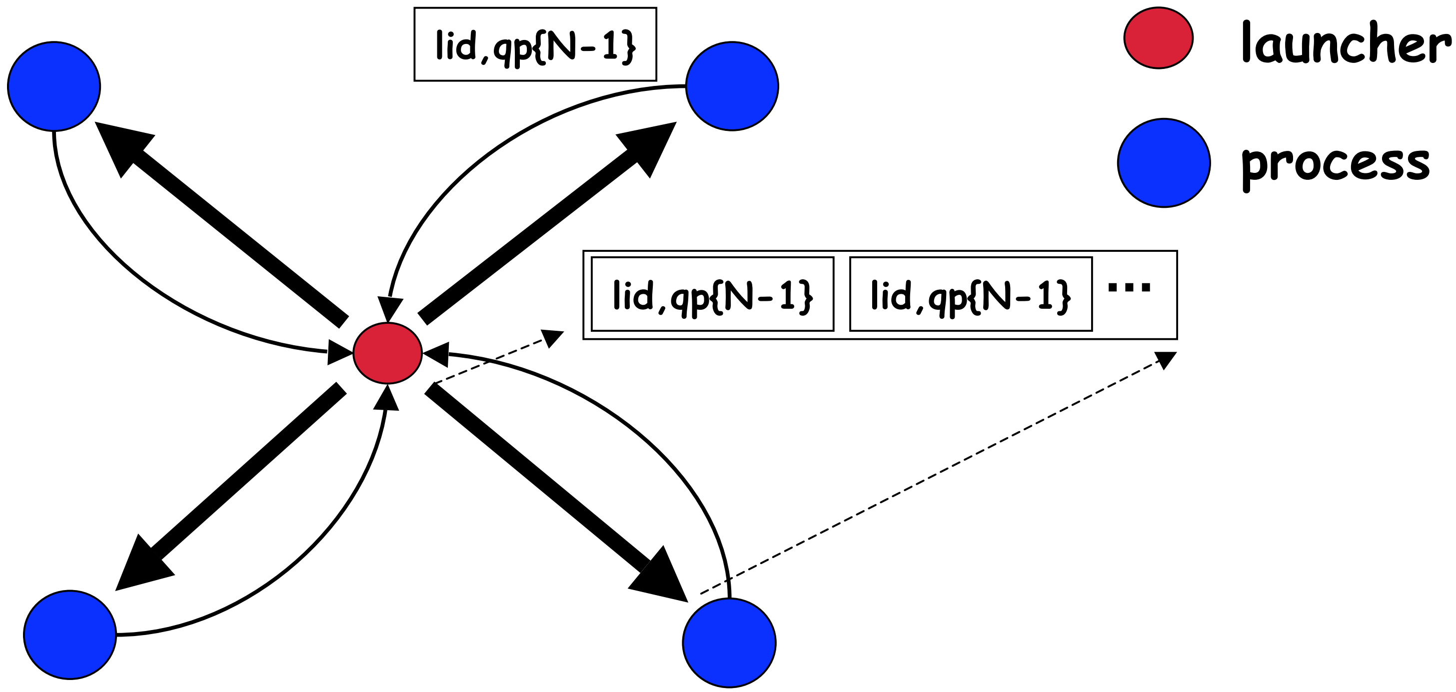
- Characteristics of Reliable Connection
  - Need to have a pair of queue-pair established
  - Need to exchange QP-ID (queue-pair identification) and LID (HCA identification)
  - Connection-oriented
- Representative Implementation, MVAPICH
  - Unique QP-ID per-process
  - N*(N-1) connections among N processes

Startup in MVAPICH: Process Initiation



- Application processes launched through rsh/ssh daemons
- All application processes connect back to launcher with an open port, but not yet connected among each other

Connection Setup



- The launcher gathers a LID and N-1 QP-ID from each process
- The combined N copies of LID and QP-ID are sent to each process
- Application processes then use LID and QP-ID to set up RC connections

Scalability Bottlenecks

- Connection setup phase
  - Receive: N copies of LID + (N-1) QP ID
  - Send: N copies of (N* LID + N * (N-1) QP-ID)
  - Amount of data:
    - $O(N^3)$ for N processes
    - 4GB for 1024-processes
- Also at the process initiation phase
  - Iterative and serialized rsh/ssh

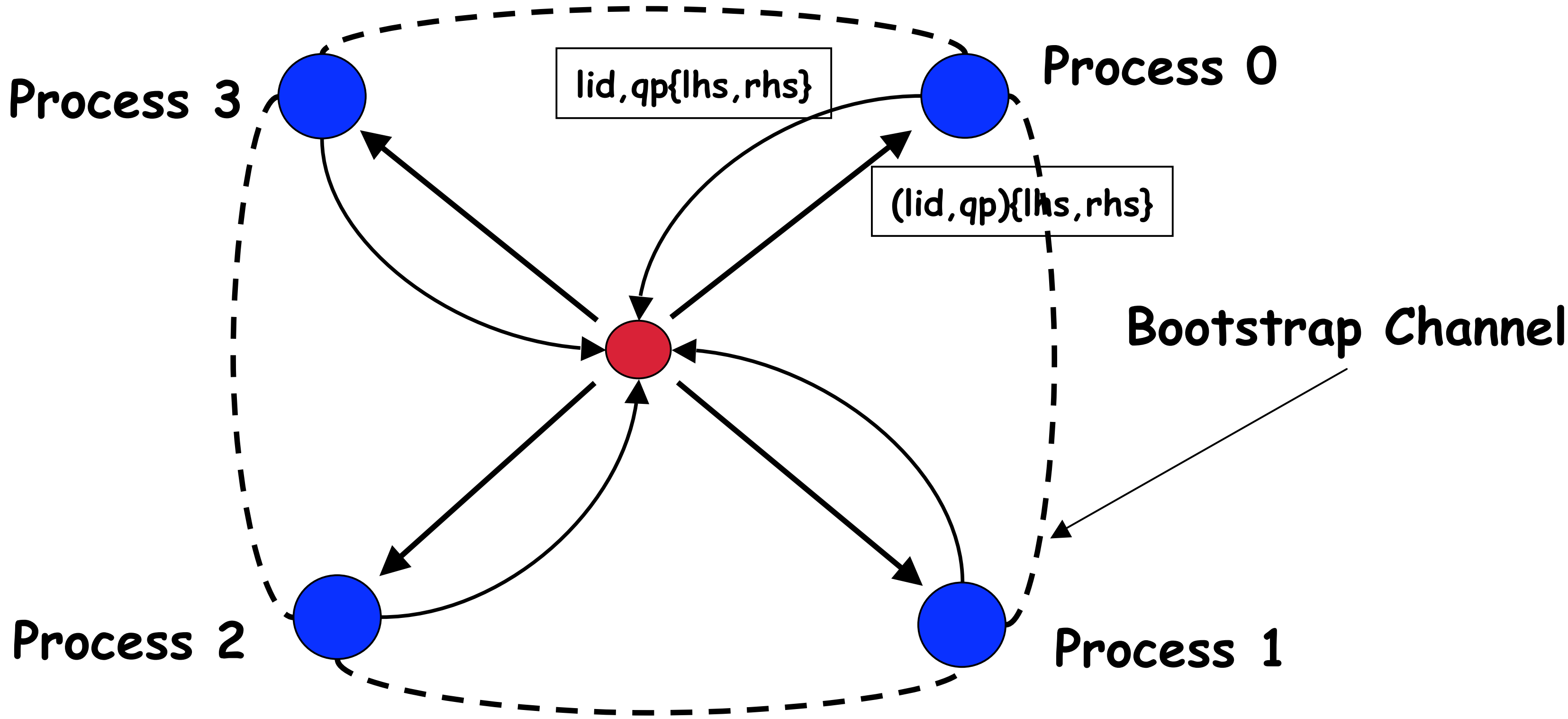
Presentation Outline

- Background
- Startup of MPI programs over IBA
- Designing Scalable Startup Schemes
- Performance Evaluation
- Conclusions and Future Work

Efficient Connection Setup

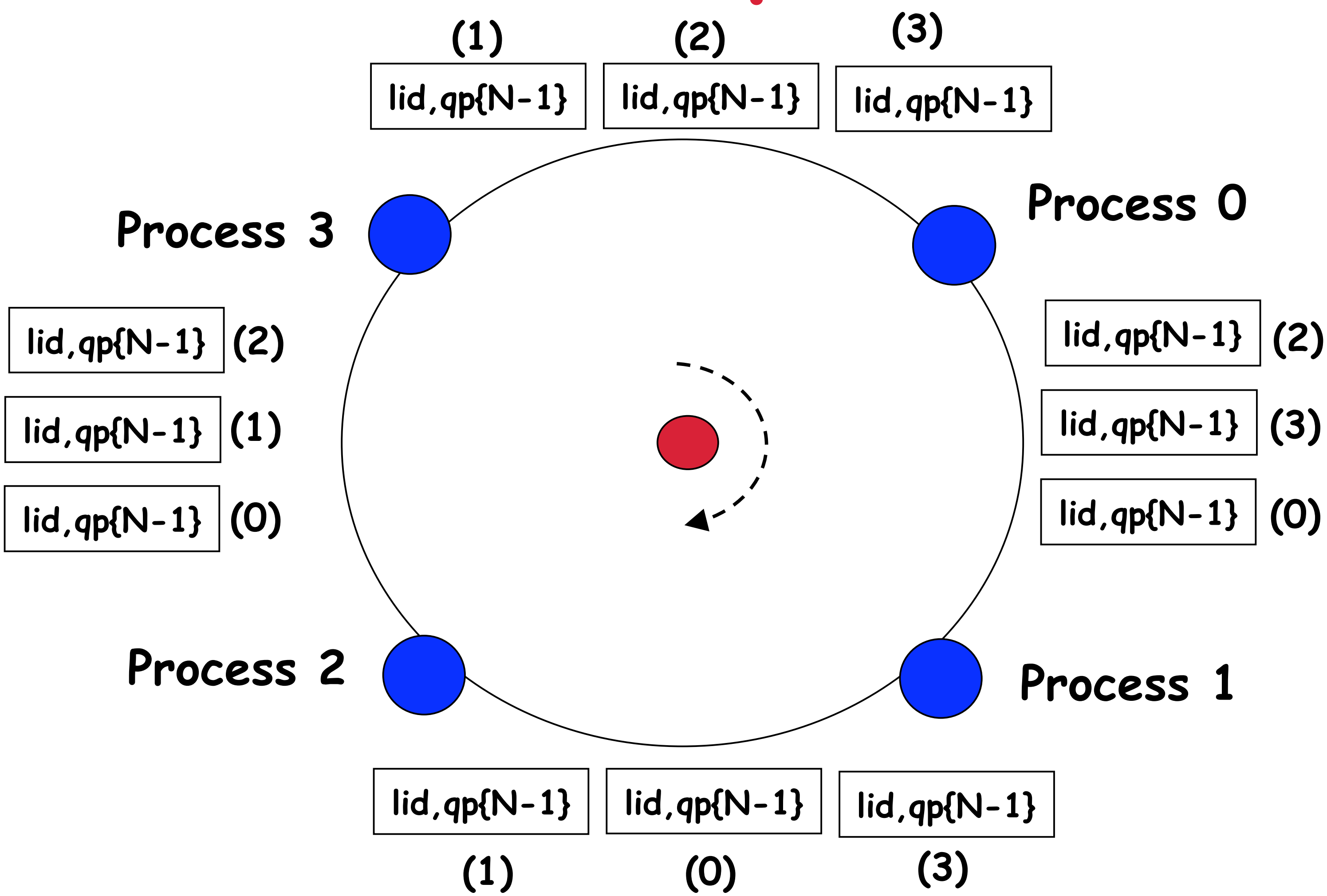
- Reducing the data volume
  - Out of (N-1) QP-ID from each process, only one QP-ID is needed for a particular peer process to set up IB connection
  - Data reassembly at the job launcher
    - Instead of sending the combined N*(N-1) QP-ID, select (N-1) QP-ID for a particular processes
  - Reducing the total data volume from N^3 to N^2

Communication Parallelization



- Each process sends its LID and QP-ID's for its left-hand side (lhs) and right-hand side (rhs) processes
- In return, each processes receives LID and QP-ID, from both lhs and rhs processes

Queue Pair Exchange over the Bootstrap Channel



Bootstrap Channel

- Pros:
  - Queue pair exchange with inband communication over InfiniBand
  - Fast IB communication compared to Ethernet
  - Ring-based All-to-all Broadcast
    - Each process is sending N copies of lid, qp{N-1}
    - Parallelized queue pair exchange over the bootstrap channel
- Cons:
  - An overhead of setting up the bootstrap channel

Fast Process Initiation

- We also utilize a fast job launcher to replace iterative rsh/ssh-based process initiation
  - MPD is chosen as it is widely distributed along with MPICH
  - Can be applied to others such as PBS.
- Incorporated with the inband bootstrap channel to improve the scalability of QP exchange

Presentation Outline

- Background
- Startup of MPI programs over IBA
- Designing Scalable Startup Schemes
- Performance Evaluation
- Conclusions and Future Work

Experimental Testbed

- A 128-node InfiniBand cluster
  - Dual-SMP Intel xeon processors
  - 2.4GHz, 4GB RAM
  - PCI-X 133MHz/64-bit
- File system Effects
  - NFS access could impact the startup
  - All binary files are first broadcasted to local disks to avoid file system bottleneck

Experiments

- Four Startup schemes were evaluated with varying number of processes
  - Original: the original startup scheme in MVAPICH 0.9.1
  - SSH-DR: data reassembly to reduce the data volume
  - SSH-BC:
    - parallelized queue pair exchange over inband bootstrap channel
  - MPD-BC:
    - Fast process initiation with MPD
    - Inband bootstrap channel for scalable connection setup
- Analytical Modeling of the scalabilities of these four schemes

Startup Time

Table 1. Performance Comparisons of Different Startup Schemes (sec)

Number of Processes	4	8	16	32	64	128
Original	0.59	0.92	1.74	3.41	7.3	13.7
SSH-DR	0.58	0.94	1.69	3.37	6.77	13.45
SSH-BC	0.61	0.95	1.70	3.38	6.76	13.3
MPD-BC	0.61	0.63	0.64	0.84	1.58	3.10

- Both SSH-DR and SSH-BC reduce the startup time
- MPD-BC perform the best because it takes advantage of MPD fast process initiation and fast connection up with the bootstrap channel
- With up to 128-processes, the improvement can be more than 4 times

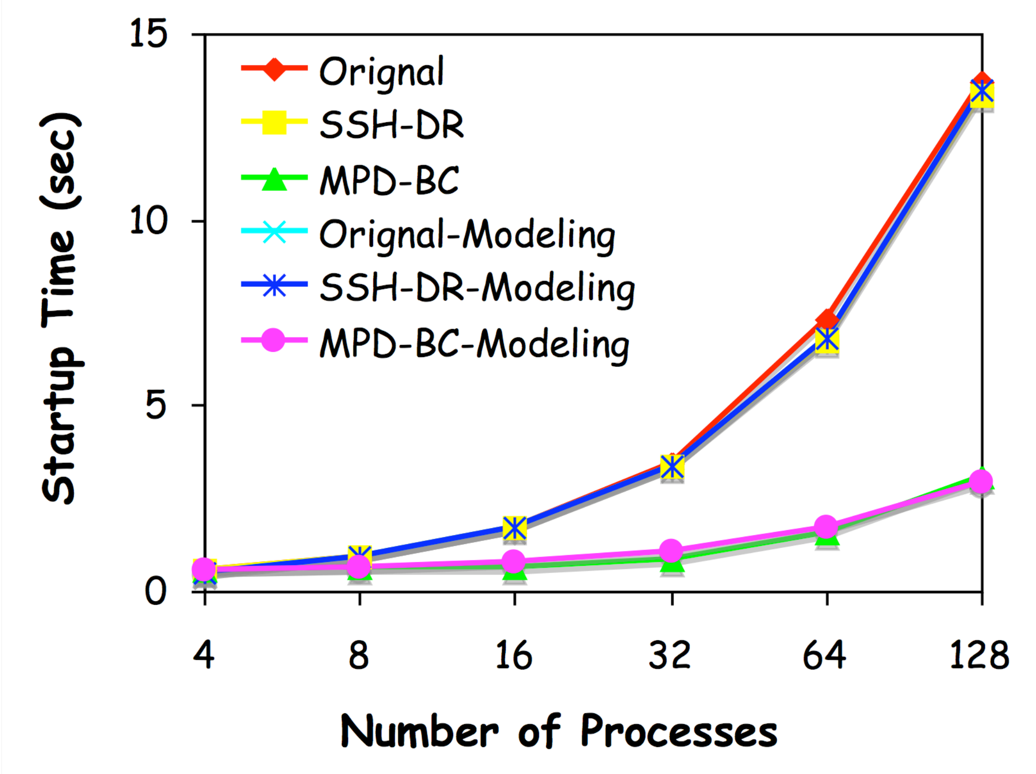
Modeling the Startup Time

- General Formula: $T_{startup} = T_{init} + T_{conn} + Constant$
  - $T_{startup}$: Total startup time
  - T_{init} : Process Initiation Time
  - T_{conn}: Connection Setup Time
- Original Scheme:
  - $T_{startup} = O_0 * N + O_1 * (W_N + W_N^2) * N + O_2$
  - O_0, O_1, O_2 are constants;
  - W_N, W_N^2 : Transfer time for N, N2 bytes, respectively

Modeling the Startup Time

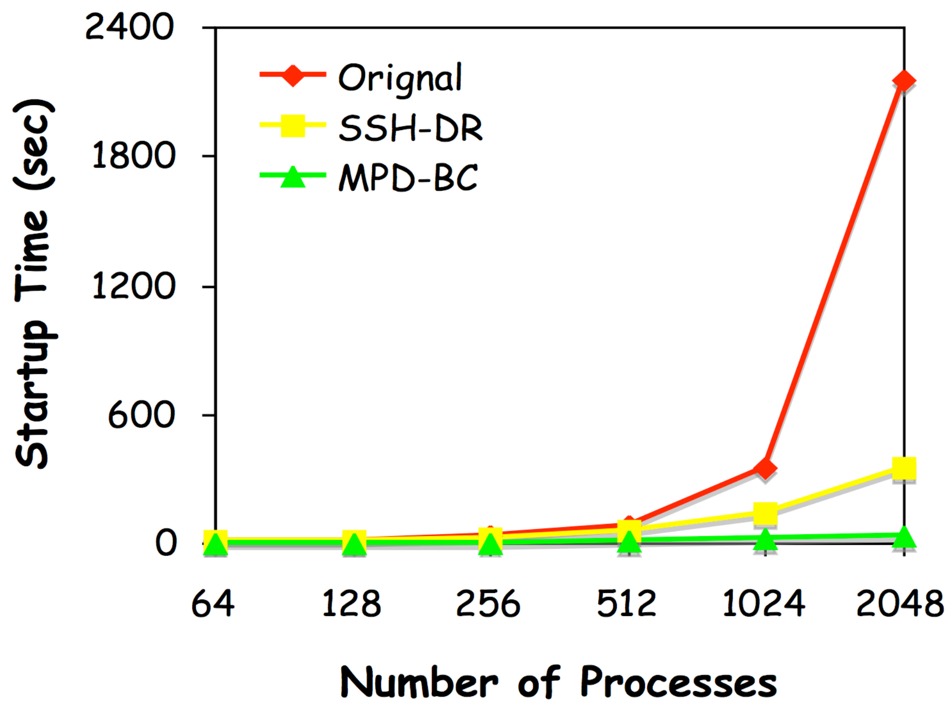
- SSH-DR:
  - $T_{startup} = D_0 * N + D_{comp} * N + D_1 * (W_N + W_N) * N + D_2$
  - D_o, D_1, D_2 and D_{comp} are constants
  - $D_{comp} * N$: Computation time for Data Reassembly
  - $W_N + W_N$: data transfer time with reduced data volume
- MPD-BC:
  - $T_{startup} = (M_o + M_{req} * N) + (M_{ch\_setup} * N + M_1 * W_N * N) + M_2$
  - M_0, M_1, M_2, M_{req} and M_{ch_setup} are constants;
  - $(M_o + M_{req} * N)$: Parallelized process initiation in MPD, with a launch request going through the ring of MPD daemons
  - $M_{ch\_setup} * N$: The time to setup a bootstrap ring

Effectiveness of the Modeling



- Parameters for the analytical models are computed based on experiment results up to 128 processes
- The modeling results is rather effective to reflect the trend of the experiment results

Scalability



- SSH-DR and MPD-BC have lower order of scalability trends
- Over 2048 processes, MPD-BC can improve the startup time by more than two orders of magnitudes

Presentation Outline

- Background
- Startup of MPI programs over IBA
- Designing Scalable Startup Schemes
- Performance Evaluation
- Conclusions and Future Work

Conclusions

- Studied Scalable Startup of MPI programs over InfiniBand Clusters
- Scalable connection setup with two schemes
  - Data reassembly to reduce the data volume
  - Parallelized queue pair exchange over a bootstrap channel
- Fast process initiation with MPD
  - With the bootstrap channel to improve connection setup
- Improve startup time by 4 times over 128 processes
- Analytical model indicates two magnitudes of improvement over 2048 processes

Future Work

- Incorporate a file broadcast mechanism for even faster process initiation
- Explore a hypercube-based data exchange to enhance scalability of queue pair exchange for large size systems
- Explore the possibility of on-demand dynamic connection with InfiniBand connection management support

More Information

NBCL home page

http://www.cse.ohio-state.edu/~panda/

http://nowlab.cis.ohio-state.edu/

E-mail: {yuw,wuj,panda}@cse.ohio-state.edu