VolpexMPI: an MPI Library for Execution of Parallel Applications on Volatile Nodes

Troy LeBlanc, Rakhi Anand,
Edgar Gabriel, and Jaspal Subhlok

Department of Computer Science,
University of Houston

Volpex: Parallel Execution on Volatile Nodes

- Key motivation: Idle desktops represent a massive unused computation resource pool.
- BOINC & CONDOR
  - BOINC: 500,000+ volunteer nodes worldwide, many application projects
  - CONDOR: job scheduler, widely used for desktops and clusters, 100s of installations
  - Sequential and “bag of tasks” parallelism
- Volpex Goals: Execution of communicating parallel programs on volatile ordinary desktops
- Key problem: High failure rates AND coordinated execution

Example Application: REMD

- Collaboration with Prof. Cheung, Department of Physics, UH
- A approach for studying the folding thermodynamics of small to modest size proteins in explicit solvent
- High computational requirements coupled with relatively small (!= 0) communication requirements
- Communication required for
  - Synchronization of processes
  - Store/Read energy values between neighbors
  - Exchange temperate values for next simulation step

REMD – Temperature swapping between replicas

STEP	P1	P2	P3	P4	P5	P6	P7	P8
1	270	280	290	300	310	320	330	340
2	280	270	300	290	320	310	330	340
3	290	270	300	280	320	310	330	340
4	290	270	300	280	310	320	340	330
5	280	270	310	290	300	330	340	320

- Application run with 8 replicas (8 temperatures)
- Parameters: nstlimit=4000; nsteps=5
- *Processes that swap temperatures at a step have same background color*

Volpex MPI

- MPI library for execution of parallel application on volatile nodes
- Key features:
  - controlled redundancy: each MPI process can have multiple replicas
  - Receiver based direct communication between processes
  - Distributed sender logging
- Prototype implementation supports ~40 MPI functions
  - blocking and non-blocking point-to-point operations
  - collective operations
  - communicator management

Point-to-point communication

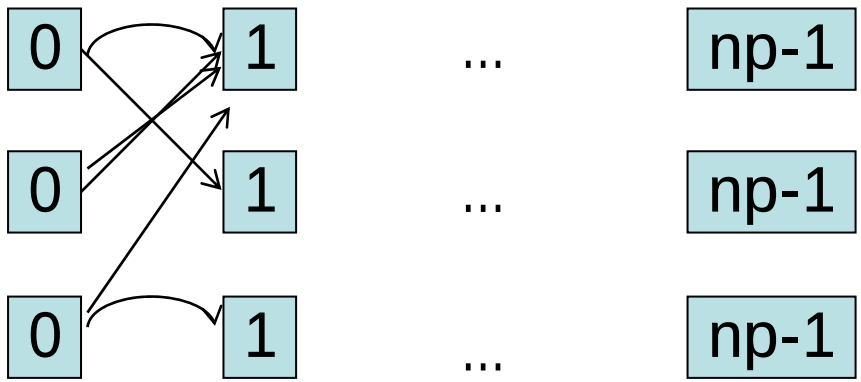
- Goal: efficient handling of multiple replicas for each MPI process
  - avoid sending each message to all replicas
- Concept:
  - receiver based communication model
    - sender buffers message locally
    - receiver contacts sender process requesting message
    - logical time stamps ("incarnation id") used for message matching in addition to the usual message envelope (tag, communicator, sender rank, recv rank)
  - no support for `MPI_ANY_SOURCE` as of today

Volpex MPI design

- Data transfer based on non-blocking sockets
  - supports timeout of messages and connection establishment
  - handling of failed processes
  - adding of new processes at runtime
- Sender buffer management:
  - circular buffer containing message envelopes and data
  - oldest log-entries are being overwritten
  - size of the circular buffer limits as of today ability to retrieve previous messages
    - can be balanced with timeouts of the communication layer and checkpointing intervals of application

Target selection

- Problem: which replica to contact for a message?
- Example: rank 1 sends a message to rank 0
  - which means rank 0 contacts rank 1 for the data



Target selection

- Challenge:
  - choose fastest replica for performance reasons
  - do not want to slow the fastest replica down by overloading it with requests from all processes

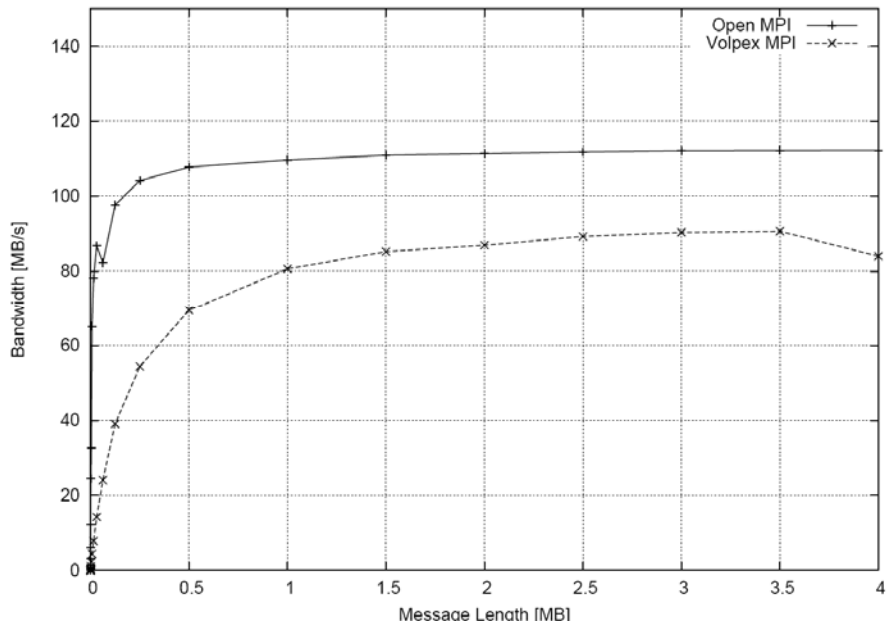
  → find the replica that is closest to your execution stage
- Current approach:
  - processes are grouped in teams
  - processes of other teams only contacted in case of a process failure
- New algorithms currently being implemented/evaluated

Test environment

- Initial tests on a cluster
  - 24 SUN X2100 nodes
    - single dual-core AMD Opteron processor
  - 5 SUN X2200 nodes
    - two quad-core AMD Opteron processors
  - (4xInfiniBand network interconnect)
  - Gigabit Ethernet network interconnect
- Only one process per node for consistency reasons
- Reference performance: Open MPI v 1.2.6 using GE network

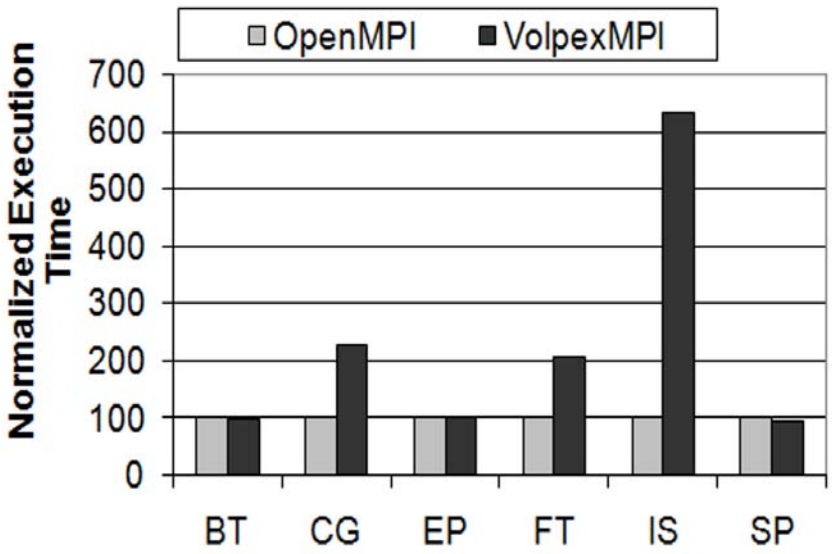
Bandwidth comparison

- 4 byte latency over GE:
  - Open MPI: ~0.5ms
  - VolpexMPI: ~1.8ms

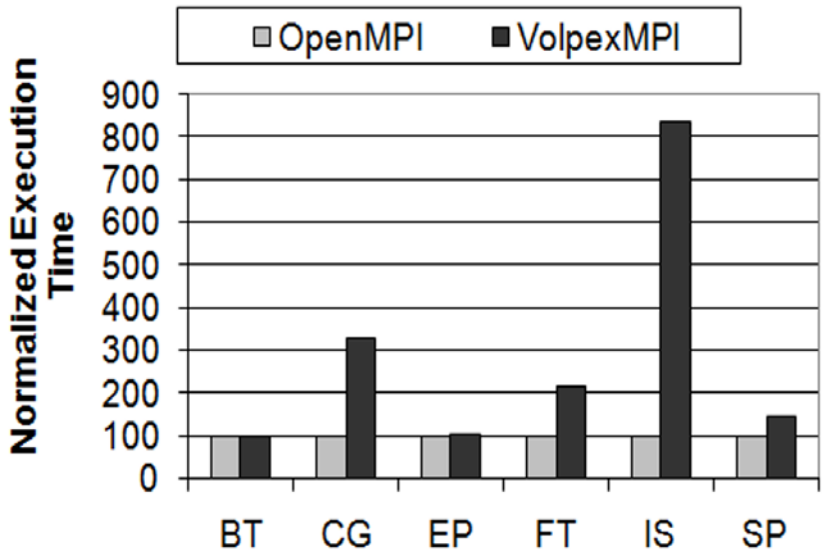


NPB Comparisons with Open MPI

- NAS Parallel Benchmarks class B
  - 8 and 16 processes test cases
  - MG and LU skipped due to `MPI_ANY_SOURCE`



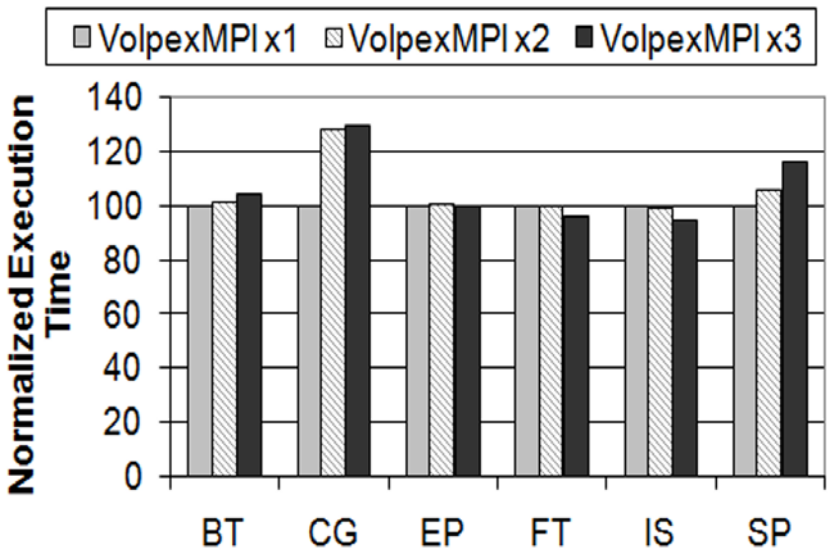
8 processes



16 processes

Influence of redundancy level

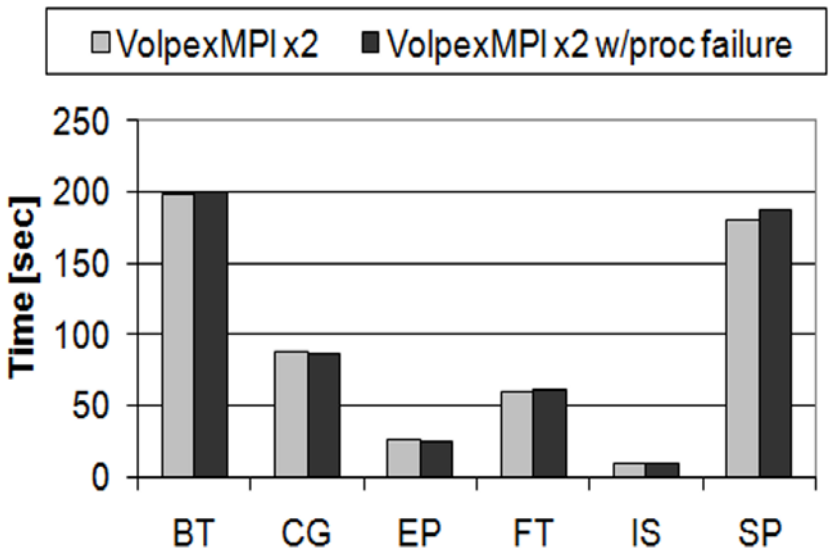
- Performance impact of executing one (x1), two (x2) and (x3) replicas of each process



8 processes

Influence of process failures

- failing second replica of rank 1 in the first iteration of the code
  - one worst case scenario!



8 processes

Summary

- Volpex MPI allows for the seamless handling of multiple process replicas of MPI process
  - minimal or no performance penalty due to replication
  - seamless handling of process failures
- Application have to carefully chosen for volunteer computing
  - communication/computation ratio
  - low degree of communication

Ongoing work

- Integration with CONDOR and testing using volunteer computing resources
- New target selection algorithms:
  - performance based algorithms
  - distance based algorithms
  - log-length based algorithms
  - time-out based algorithms
- Extend buffering concept
- Integration of checkpointing mechanisms
  - also required to add new replicas if necessary