Parallel Software Technologies Laboratory PSTL

Optimizing the Execution of Parallel Applications in Volunteer Environments

Presented by: Rakhi Anand

Advisor: Edgar Gabriel

Parallel **S**oftware **T**echnologies **L**aboratory
Department of Computer Science
University of Houston
rakhi@cs.uh.edu

CS@UH

Outline

- Introduction
  - Background, Challenges, Related Work
- Research objectives
- Introduction to VolpexMPI
  - VolpexMPI Design and Experimental results
- Target Selection module
  - Implementation of various algorithms
  - Experiments and results
- Runtime Environment support
  - Event handler and Tools implemented
- Summary and conclusion
- Future Work

Introduction (I)

- Why parallel computing
  - To solve larger problems
  - To solve given problems fast
- Classes of parallel applications:
  - Bag of task applications (no communication): e.g. SETI@home...
  - Low degree, static communication pattern: e.g. CFD...
  - Low degree, dynamic communication pattern: e.g. Adaptive mesh refinement…
  - High degree communication pattern: e.g. FFTs...

Introduction (II)

- Cluster computing
  - ➢ Tightly coupled computers that act like a single system
  - ➢ Expensive computing resources
- Grid computing
  - ➢ The combination of computer resources from multiple administrative domain for a common goal
  - ➢ Support only non-interactive jobs

Introduction (III)

- Cloud Computing
  - Internet based computing providing compute resources and/or software on demand
  - High speed network interconnects are not currently supported

Introduction (IV)

- Volunteer Computing
  - Volunteers around the world donate a portion of computer resources
  - E.g. SETI@home runs on about 1 million computers

- Advantages
  - High resources in terms memory and computing power
  - Easy to use
  - Compute cost

Challenges of volunteer computing (I)

- High failure rate:
  - Hard failures: system crash, shutdown or hardware failure
  - Soft failures: owner starting to utilize his machine
  - Failure rates are much higher in volunteer environment than e.g. on compute clusters

- Communication requirements:
  - No information about where the nodes are located.
  - No guarantee whether public IP addresses are used

Challenges of volunteer computing (II)

- Heterogeneity:
  - Volunteer environment provide heterogeneous collection of nodes
  - Different processor types, frequency, memory size...
  - Node properties change dynamically

Related Work (I)

- Various volunteer computing exploit unused cycles on ordinary desktops.
  - ➢ Boinc : provide support for bag of task applications
    - No mechanism for node-to-node communication
  - ➢ Condor: a batch scheduler that allows to control distributed resources
    - can run MPI jobs on clusters
    - No support for executing MPI applications on volunteer nodes

Related Work (II)

- MPI is the dominant programming paradigm for parallel scientific applications
  - Message passing paradigm
  - Support for heterogeneous environments through
    - Notion of data types
    - Process grouping
    - Collective (group) communication operations
- MPI specification does not provide any mechanism to deal with process failures

Related Work (III)

- Research projects exploring fault tolerance for MPI can be divided into 3 categories
  - Extension of MPI semantics: FT-MPI
    - Requires major changes to user program
  - Roll-back recovery: MPICH-V, LAM/MPI
    - Better for less frequent failures
  - Replication: MPI/FT, P2P-MPI, rMPI
    - All replicas executed in a lock-step fashion

Thesis Goals (I)

- Extend an efficient and scalable communication infrastructure for volatile compute environments
  - Deal with heterogeneity of volunteer computing environments
  - Deal with frequent process failures efficiently
  - Deal with challenges of new computer architecture
- Increase the class of applications that can be executed in volunteer computing

Thesis Goals (II)

- Target Selection :
  - ➢ To optimize the communication operations by contacting the most appropriate copy of target process.
- Runtime environment support :
  - ➢ Support for flexible number of processes, replication levels, process failures and various process management systems.
  - ➢ Support for tools for managing runtime environment

Introduction VolpexMPI

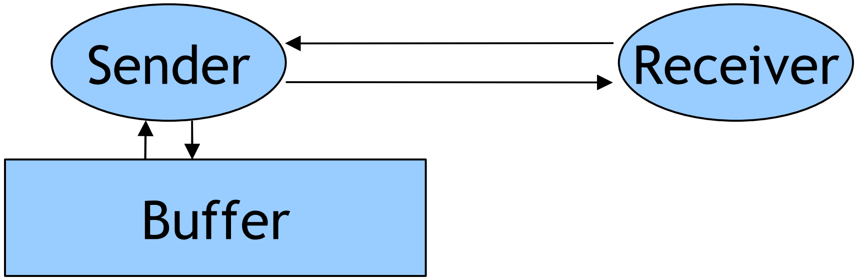
- VolpexMPI is an MPI library designed for executing parallel applications in volunteer environments

- Key Features-
  - Controlled redundancy
  - Receiver based direct communication
  - Distributed sender based logging

Parallel Software Technologies Laboratory PSTL

VolpexMPI Design (I)

- Point to point communication
  - Based on pull model
  - Goal: make the application progress according to the fast replica
- Message matching scheme
  - Virtual timestamp: a sequence counter used to count number of messages having the same message envelope [communicator id, message tag, sender rank, receiver rank].
  - Messages matching based on message envelope + timestamp

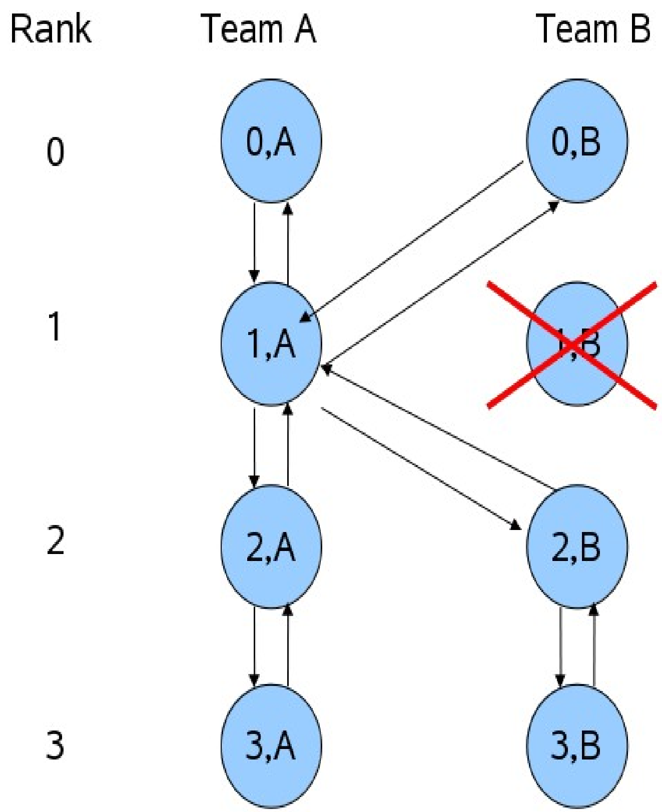


VolpexMPI Design (II)

- Buffer Management
  - Messages are stored with the message envelope
  - Circular buffer is used to store messages
  - Oldest entry is overwritten
- Data transfer
  - Non blocking sockets
  - Handling connection setup on demand
  - Timeout for connection establishment and communication operations

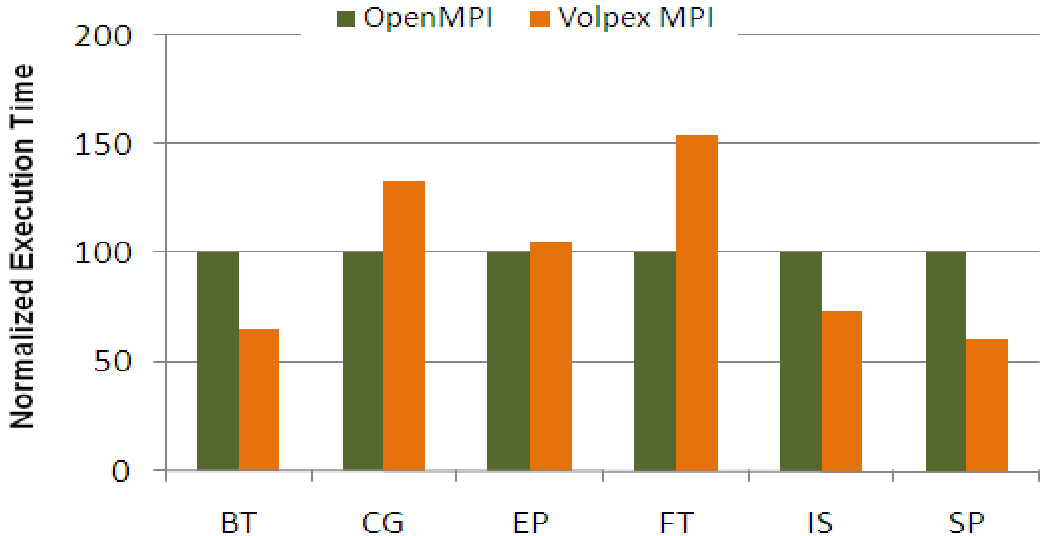
Managing Replicated MPI processes

- Processes are spawned in teams
- Only in case of failure, processes from different team is contacted

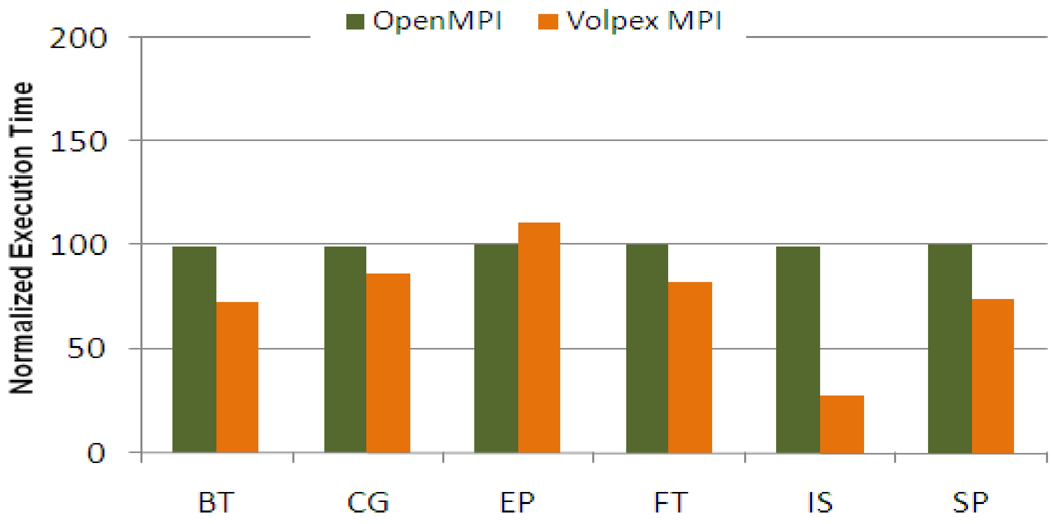


Experimental results VolpexMPI (I)

Runs for 32 procs

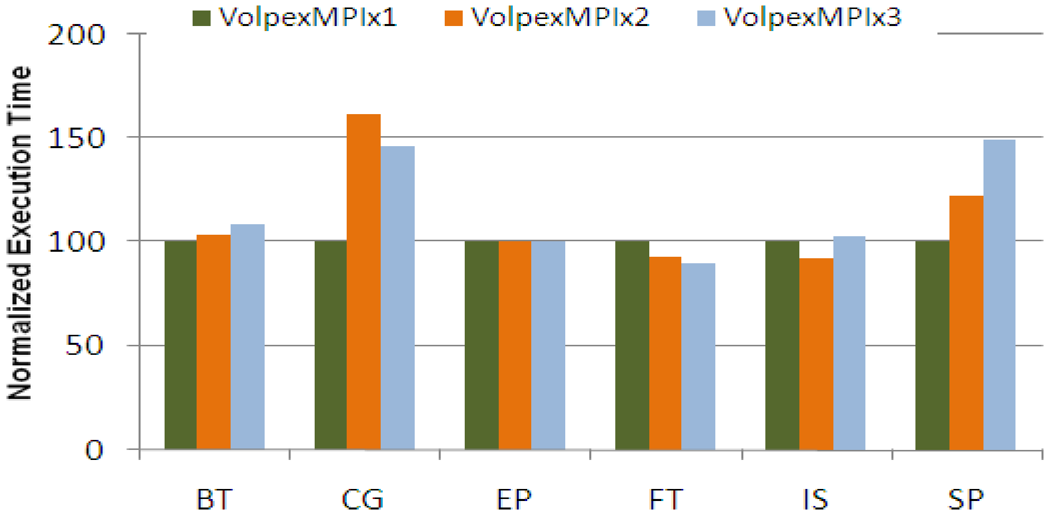


Runs for 64 procs

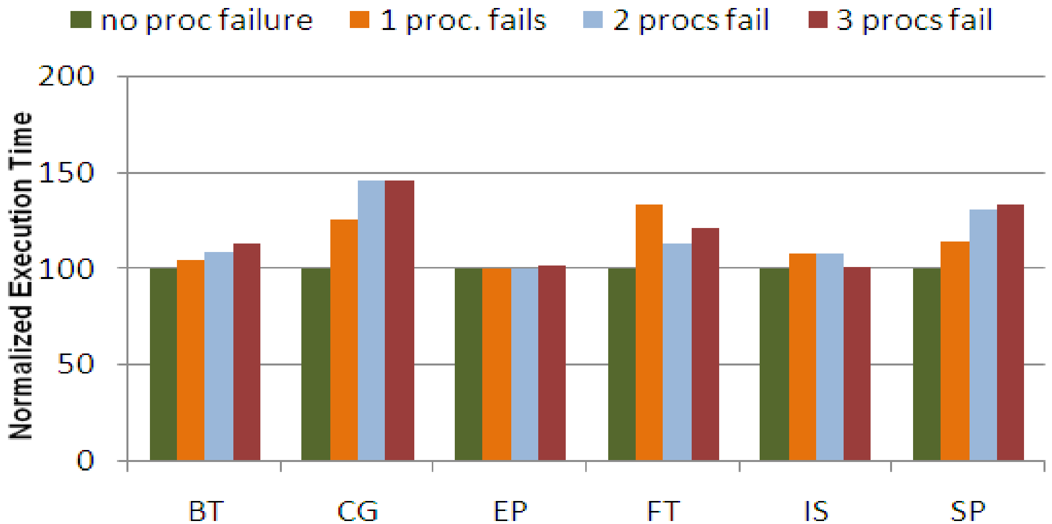


Experimental results VolpexMPI (II)

Redundancy runs(16 procs)

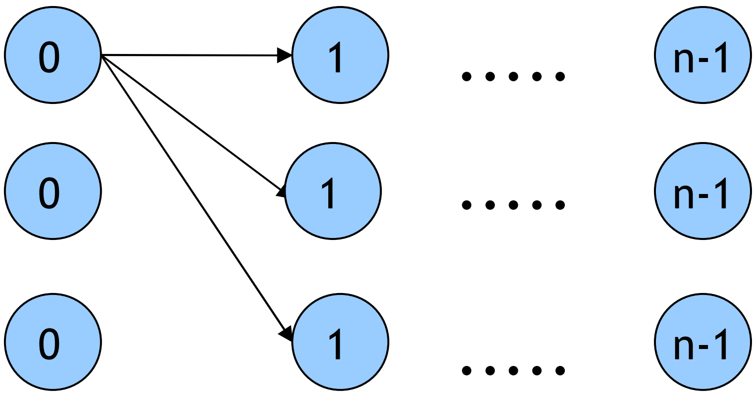


Failure runs (16 procs)



The Target Selection Problem (I)

- Identifying best set of replicas



- Beneficial to connect to fastest replica
- Will make fast replica slow by making it handle more number of requests

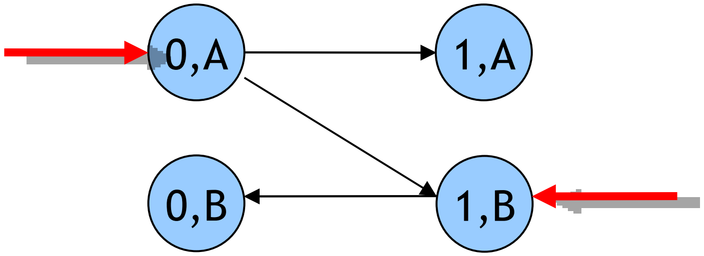
The Target Selection Problem (II)

- Definition: create an order of replicas for each application process in order to optimize the performance of application

- Four algorithms explored:
  - ➢ Network performance based
  - ➢ Virtual timestamp based
  - ➢ Timeout based
  - ➢ Hybrid approach

Network performance based algorithm

- Each process prioritize replicas based on latency/bandwidth values
  - Measurements performed during the regularly occurring communication operations of the application
- Advantage: can dynamically detect changes in network characteristics
- Disadvantage: misleading performance numbers due to overlapping, asynchronous communication operations



Virtual timestamp based algorithm

- Two processes which are close in execution perspective should contact each other
- Advantage: processes close in execution state will group together without interfering the execution of other processes
- Disadvantage: difficult to determine for synchronised MPI applications

Timeout based algorithm

- Switching the replica if the request is not handled within the given time frame

- Advantage: if a replica is too slow the application will advance at the speed of fast set of replicas

- Disadvantage: difficult to determine good timeout value

Hybrid algorithm

- Combination of network based and virtual timestamp based algorithms
- First step:
  - Pairwise communication is initialised during initialization
  - Best target is determined based on network parameters
- Second step:
  - Based on virtual timestamp
  - If process is lagging behind it changes its target to slow process
- Disadvantage: Increased initialization time

Experiments (I)

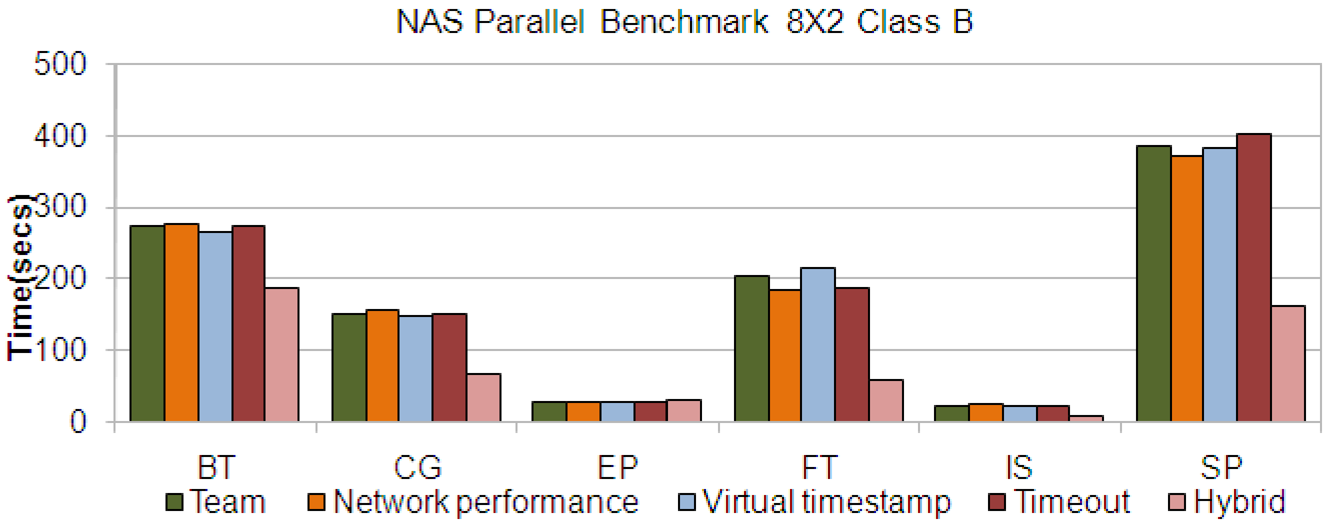
- Three different sets of experiments are performed
  - **Heterogeneous network**: reduce network performance to fast Ethernet for selected nodes
  - **Heterogeneous processor**: reduce processor frequency to 1.1 GHz for selected nodes
  - **Heterogeneous network and processor:** reduce network performance to fast Ethernet and processor frequency to 1.1 GHz for selected nodes
- Executed NAS Parallel Benchmarks (Class B)
  - BT, CG, EP, FT, IS, SP (for double redundancy x2)
  - CG, EP, IS (for triple redundancy x3)

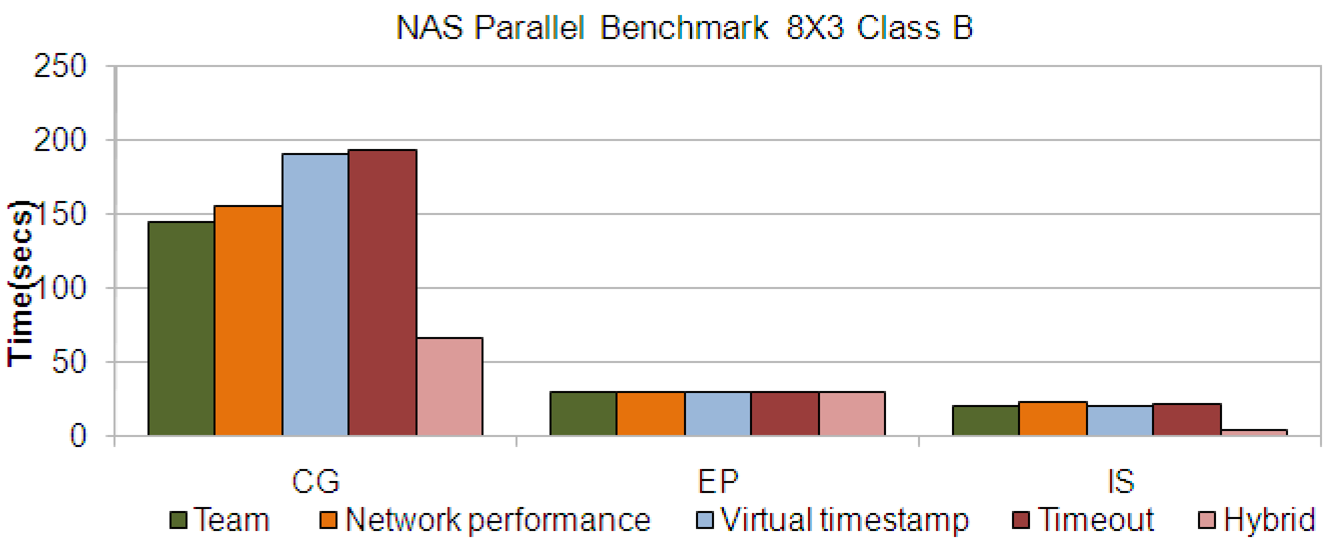
Experiments (II)

- No two replicas of same rank are on same network or same processor type
- All teams have processes on each set of nodes
- Runtime of original team based approach is compared with all other algorithms

PARALLEL SOFTWARE TECHNOLOGIES LABORATORY PSTL

Results for heterogeneous network configuration

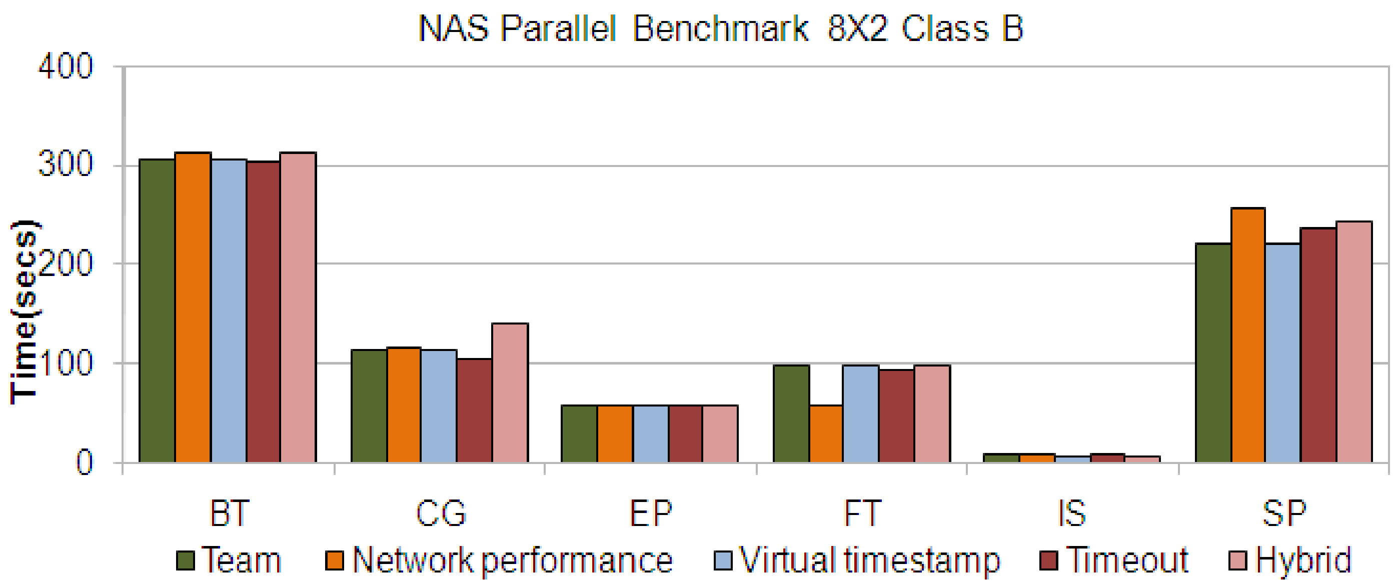




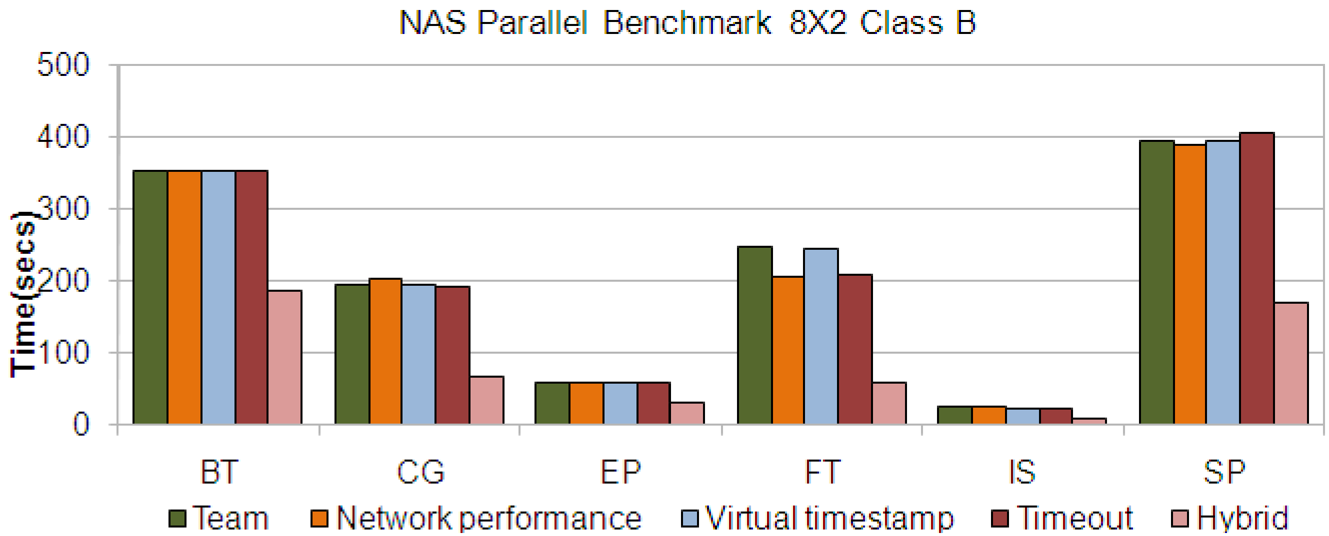
Rakhi

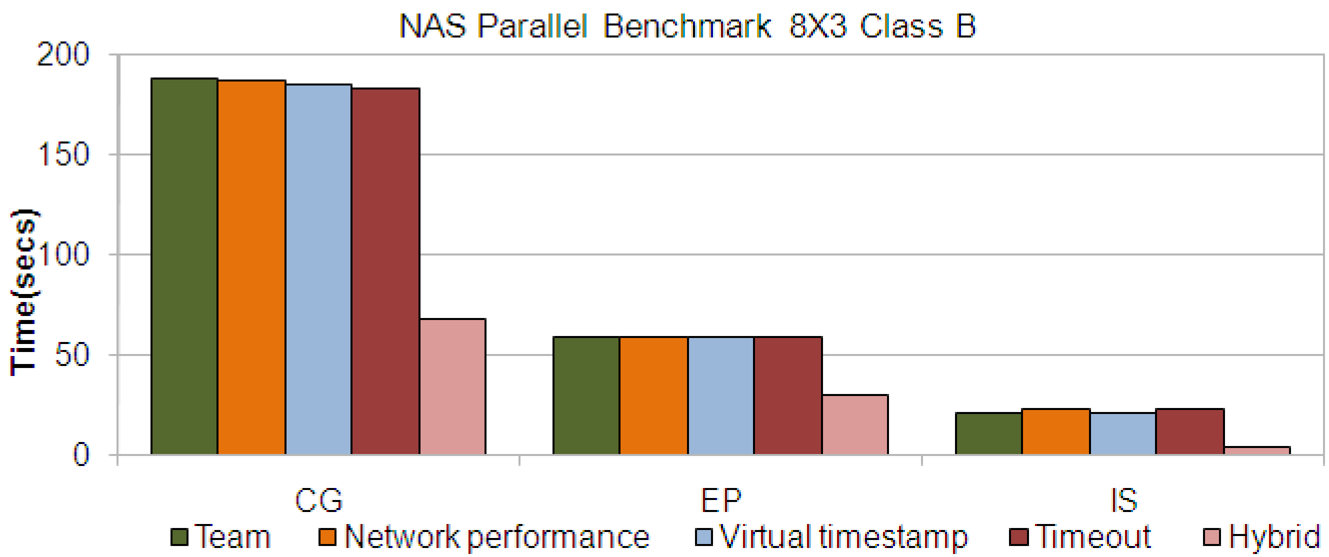
CS@UH

Results for heterogeneous processor configurations



Results for heterogeneous network and processor(I)





Results for heterogeneous network and processor(II)

	Team A	Team B	Team C
CG	87.93	67.69	184.29
IS	3.56	8.15	23.82
EP	29.69	29.72	117.83

- Team A running on shared memory
- Team B running on Gigabit Ethernet
- Team C running on Fast Ethernet

Findings

- The hybrid approach shows the significant performance benefit over other algorithms for most common scenarios.
- Pairwise communication is required to determine network parameters

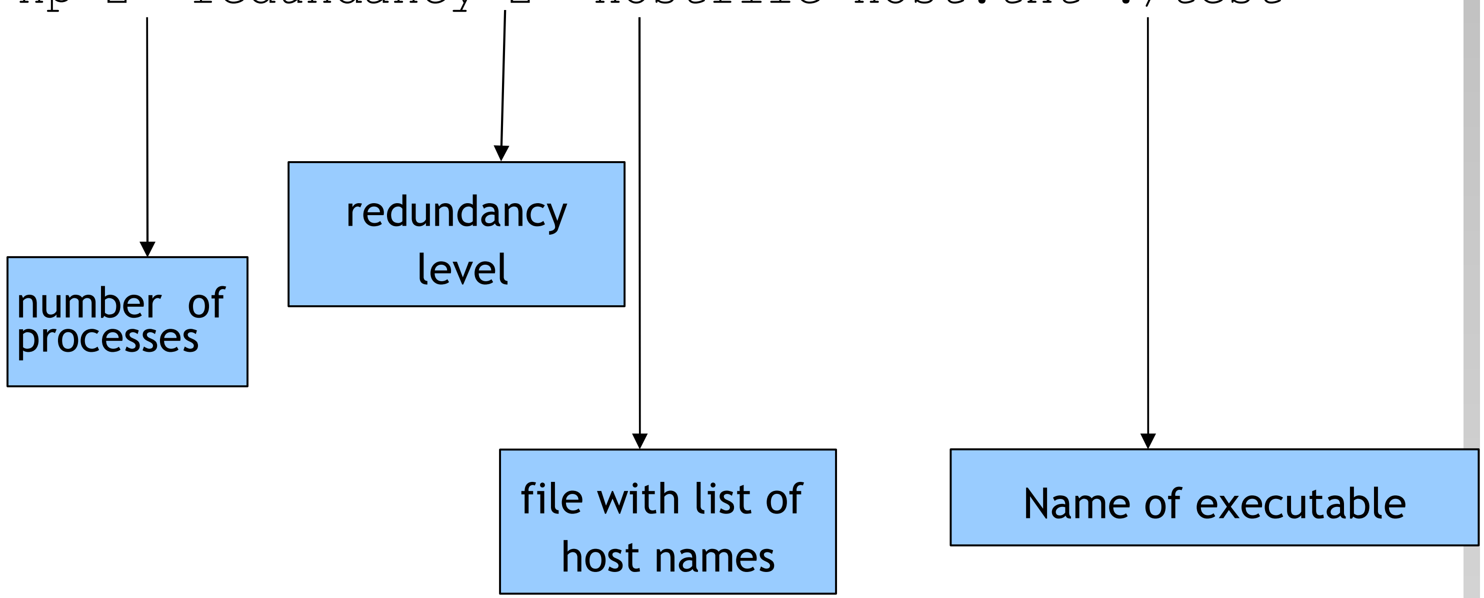
Run Time Environment Support

- A light weight environment support for spawning processes in volunteer environments
- Challenges involved
  - flexible number of processes
  - replication levels
  - process failures
  - supporting various software infrastructures(ssh, Condor)
  - to manage MPI processes
  - dynamically add new processes, new hosts or delete already existing processes

Process Manager (I)

- Command used to start different processes on different hosts

```
./mpirun -np 2 -redundancy 2 -hostfile host.txt ./test
```



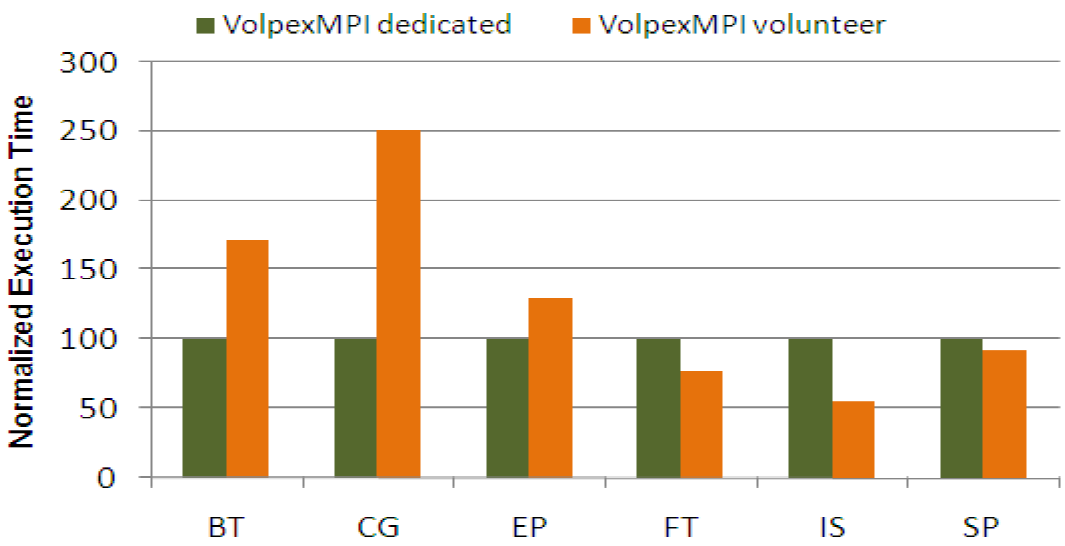
Process Manager (II)

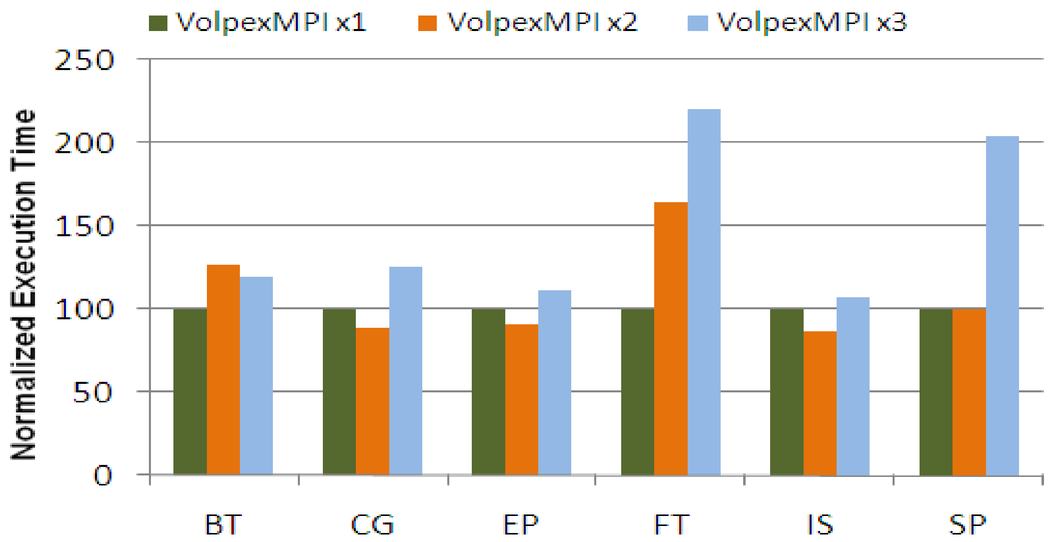
- Goal: to spawn given number of processes according to the desired replication level
- Maintains the information about hosts and processes
- Processes are spawned according to the requested execution environment(ssh, Condor)
- If hostlist is provided processes are distributed to each host in round robin manner

Process Manager (III)

- Spawning on Condor(Volunteer environment)
  - Process manager creates a submit file for Condor and spawns the executable on each available node
  - During initialization each process sends their hostname and ask for their rank
  - On receiving the message from each process, process manager sends the necessary information to each process

Experimental results for Condor





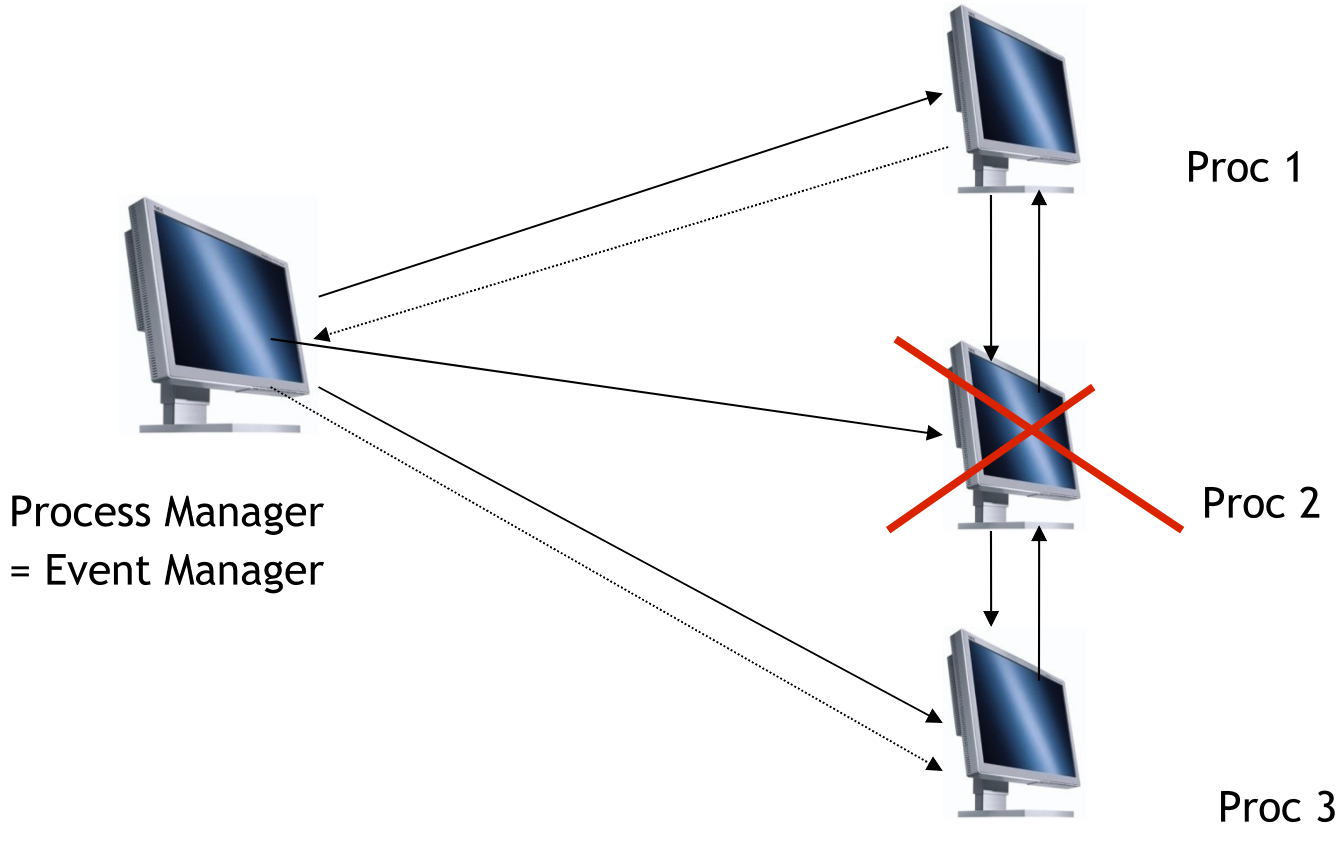
Event Management

- System messages use an integrate event management system
  - ➢ PRODUCER: who generates an event
    - existing processes or external process
  - ➢ DISTRIBUTOR: who distributes the generated event
    - Event Manager
    - distributor event handling functions
  - ➢ CONSUMER: who should be aware of the event
    - existing processes
    - consumer event handling functions

Types of Events

- ADD : adding a new process
- DEATH : deleting an existing process
- EXIT : ask a process to stop execution
- ABORT : aborting the jobid
- INFO : get details of a process, host, or job

Handling process failure



Tools Implemented

- Process Monitor :
  - Analyses the progress of an application
  - Retrieves the information about processes, hosts, jobs
- Process Controller:
  - Adds or deletes processes
  - Adds a new host
  - Provides the capability to spawn a new replica

Summary and conclusion (I)

- New target selection algorithm lead to significant benefits in heterogeneous settings
  - Hybrid algorithm performance numbers similar to the results as if all processes are running on fast nodes
  - Initial node distribution plays a very important role
  - Communication characteristic of an application can help to reduce the initialization time using hybrid approach

Summary and conclusion (II)

- Architecture for a dynamic, fault-tolerant run-time environment developed
  - Support for various replication levels
  - Support for dynamic process management
  - Support for various software infrastructures

Future Work (I)

- Initial node selection
  - Using available network proximity algorithms
  - Distributing processes according to the communication patterns
- Extending the Run-time environment
  - Integration with BOINC
  - Adding support for a new event: CHECKPOINT
  - Adding a policy component which allows to specify rules for automatic actions

Future Work (II)

- Multi-core optimizations
  - Matching the number of processes with the number of cores offered by volunteer clients
  - Share buffer management between multiple processes on a multi-core processor
  - Adapt communication scheme to process layout
- Explore real-world applications in volunteer compute environments
- Optimizing collective operations
  - Using underlying network topology
  - Extend topology aware algorithms

Timetable

	2010							2011											
	June	July	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Initial node selection																			
Runtime Environment Support																			
Multicore optimizations																			
Applications in volunteer environments																			
Optimizing collective operations																			
Final writeup																			

Publications

•Troy LeBlanc, Rakhi Anand, Edgar Gabriel, and Jaspal Subhlok. **VolpexMPI: an MPI Library for Execution of Parallel Applications on Volatile Nodes.** in M. Ropo, J. Westerholm, J. Dongarra (Eds.) 'Recent Advances in Parallel Virtual Machine and Message Passing Interface', LNCS 5759, pp. 124-134

•Rakhi Anand, Edgar Gabriel, and Jaspal Subhlok. **Communication Target Selection for Replicated MPI Processes.** Submitted to: EuroMPI 2010 Conference, Stuttgart, Germany, September 12-15,2010

•Troy LeBlanc, Rakhi Anand, Edgar Gabriel, and Jaspal Subhlok. **A Robust and Efficient Message Passing Library for Volunteer Computing Environments.** Submitted to: Journal of Grid Computing, April 2010.