Volpex Communication for Volunteer Computing

Jaspal Subhlok

University of Houston

Big Picture -- VOLPEX: Parallel Execution on Volatile Nodes

Communicating Parallel Programs ON

Ordinary Volatile Desktop Nodes

Key motivation: Idle desktops represent a massive unused computation resource

Key problem: High failure rates AND coordinated execution

Collaborators

-- *UH Faculty:* Edgar Gabriel (CS), Rong Zheng (CS), Margaret Cheung (Physics)

-- *UH Students*: Nagarajan Kanna, Troy Leblanc, Girish Nanadagudi, Eshwar Rohit, Rakhi Anand, Nat Hammen

-- David Anderson

Major Challenges in VOLPEX

Failure Management

- Replicated execution
- Independent/uncoordinated checkpoint and recovery
- Hybrid

Programming/Communication Model

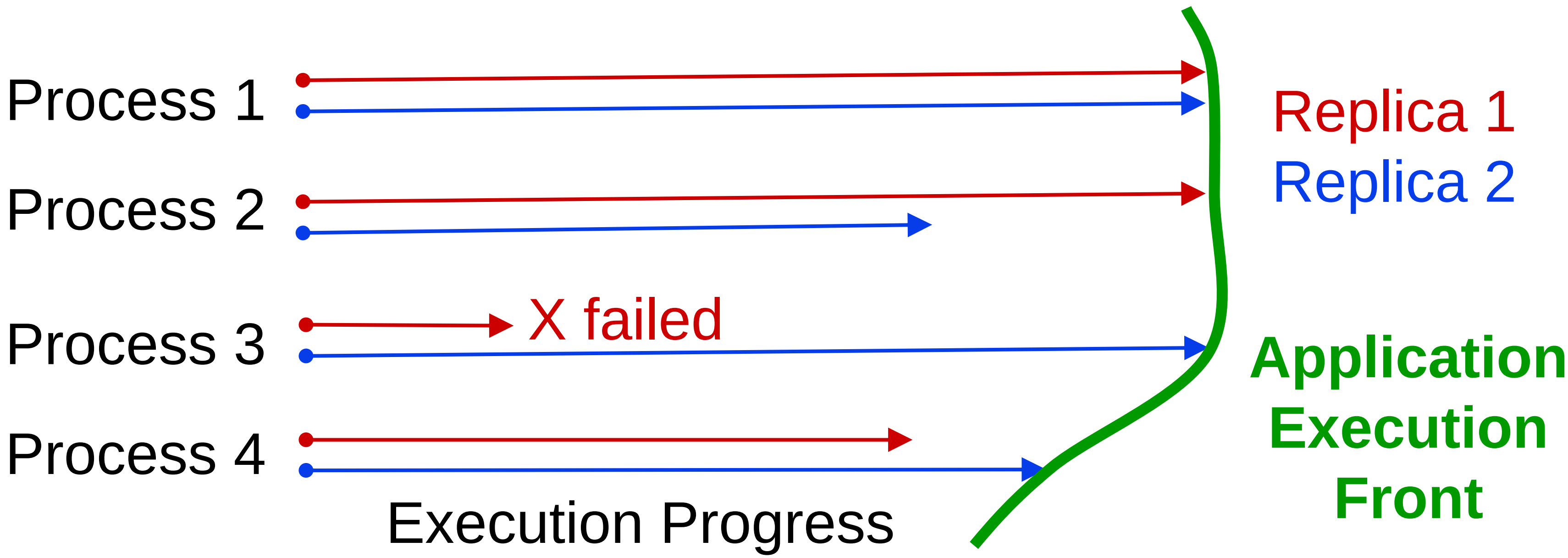
- Volpex MPI
- Volpex Dataspace API

Usage and Applications

- Integration with BOINC/Condor
- Simulation to identify suitable codes (Dimemas)
- Real world value?

Volpex Approach to Fault Tolerant Execution

Redundancy and/or independent checkpoint/restarts
→ *multiple physical processes per logical process*



- **Application progress tied to the fastest process replica**
- **Seamless progress despite failures**
- Minimum overhead of redundancy

Volpex MPI

MPI library designed for 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, including all commonly used calls.
- Runs on clusters, desktops, Condor

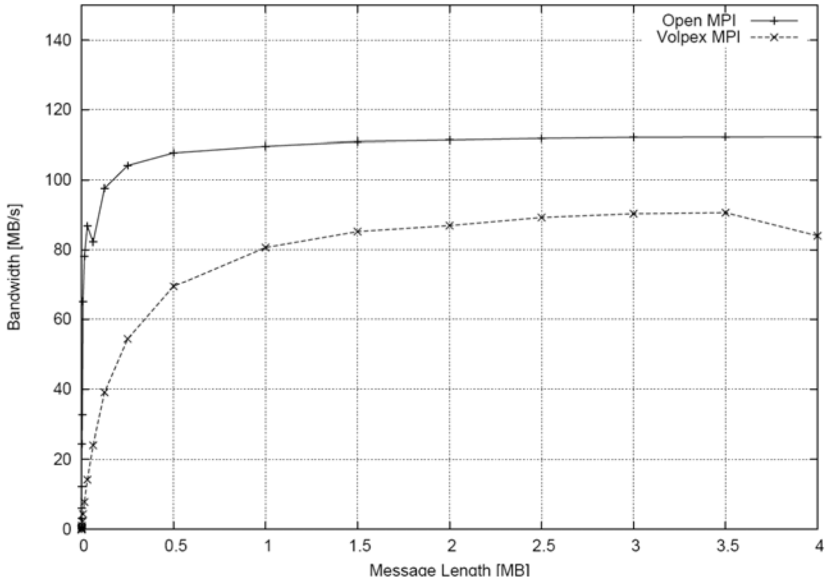
Volpex MPI Communication

- Goal: efficient handling of multiple replicas of MPI processes
  - avoid sending each message to all replicas

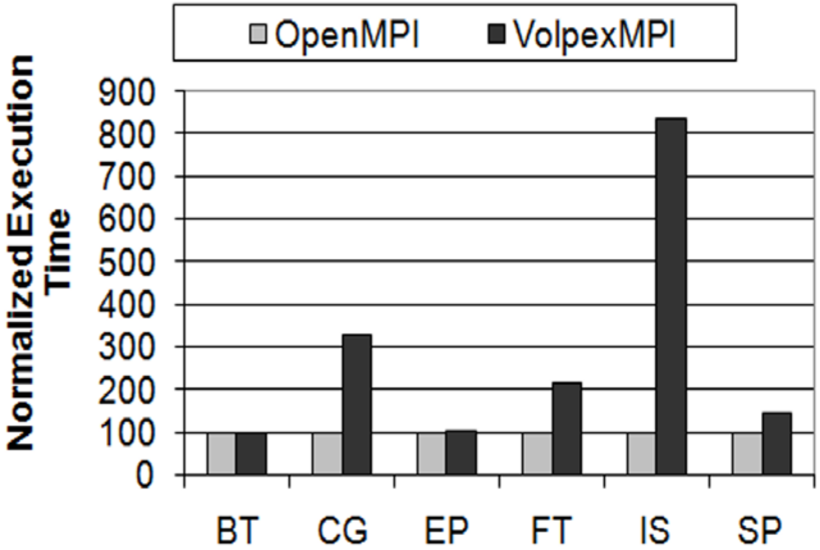
- Receiver initiated communication model
  - sender buffers message locally
  - receiver contacts sender process requesting message
  - logical time stamps (“incarnation id”) used for message matching in addition to the usual message envelope (tag, communicator, sender rank, recv rank)

Performance versus OpenMPI

Latency (4 byte message)
- **Open MPI = 0.5 ms**
- **Volpex MPI = 1.8 ms**



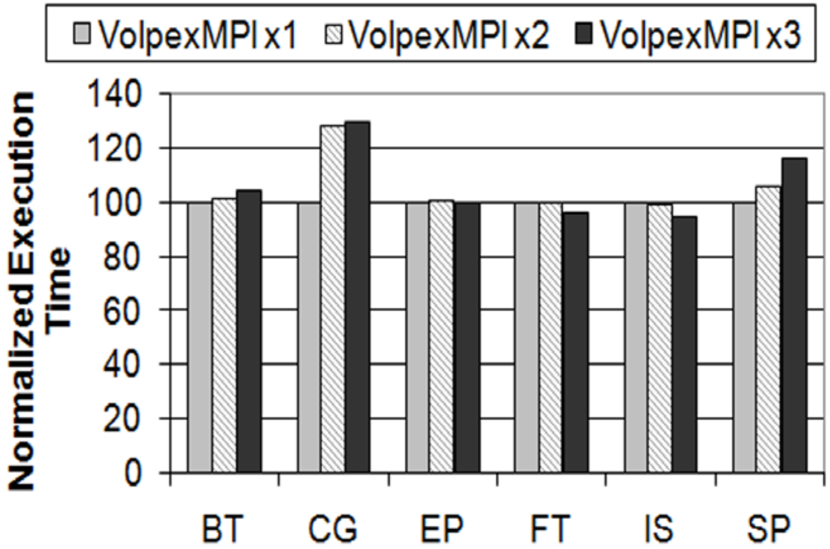
16 process NAS benchmarks
generally similar

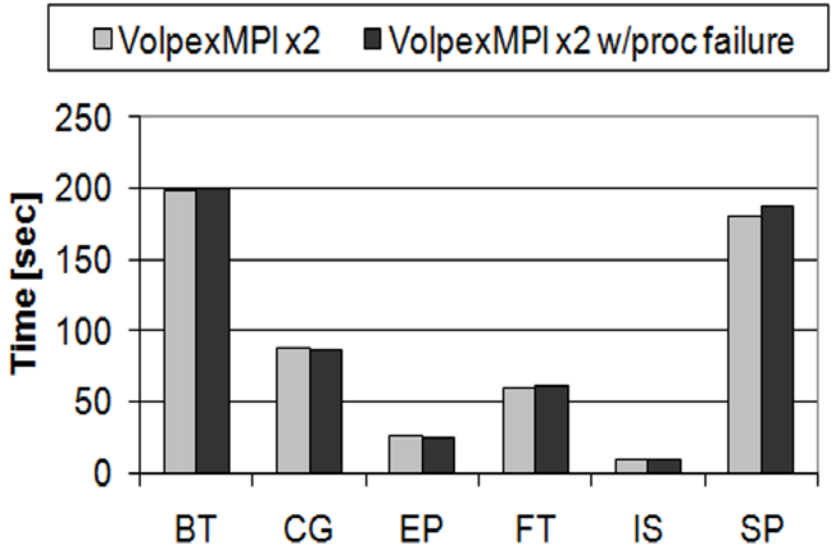


Performance with Fault Tolerance

Execution time with one (x1), two (x2) and three (x3) replicas of each process

Performance impact when one replica process fails





Details: T. LeBlanc, R. Anand, E. Gabriel, and J. Subhlok. *VolpexMPI: an MPI Library for Execution of Parallel Applications on Volatile Nodes.* In Proc. The 16th EuroPVM/MPI 2009 Conference, Espoo, Finland, 2009.

Dataspace Programming Model

Independent processes communicate with one way, PUT/GETs with an abstract dataspace

PUT (tag, data) place **data** in dataspace indexed with **tag**

READ (tag, data) return **data** matching the **tag**.

GET (tag, data) return and remove **data** matching **tag**.

- A Powerful API: can simulate message passing, global variables, producer-consumer, etc.
- Similar to Linda, Javaspaces, Tspaces**..**

Dataspace API and Redundancy

New challenge is consistency with replicated processes

- Independent checkpoint/restart also leads to redundancy

A logical PUT/GET may be executed many times

→ late replica may PUT a value that is out of date

→ late replica may READ a value that is not current

Replication Consistent Implementation

Basic principles: Let PUT1, PUT2, PUT3 and GET1, GET2, GET3 be replica calls in temporal order

- **PUT1 executes normally. PUT2, PUT3 ignored**
- **GET1 executes normally. A copy of the date object is logged. GET2, GET3 get same data as GET1.**

<process id, request #> appended to API calls for identification

Implementation, Experiments, Results

Single threaded implementation.

Applications/Examples

- Replica Exchange Molecular Dynamics (REMD)
- Map-Reduce: *Dataspace intermediary between Map and Reduce*.
- Parallel Sorting by Regular Sampling (PSRS): *Dataspace in place of message exchange*
- Sieve of Eratosthenes. *Dataspace employed for broadcast*

- Testbed consists of clusters and desktop PCs as clients and dataspace server on the LAN

REDMD

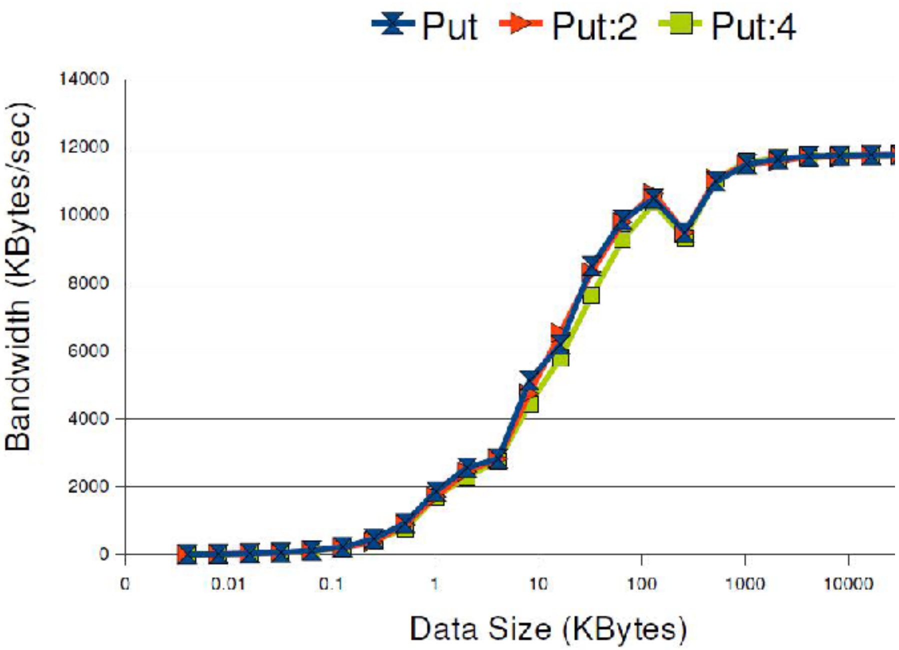
(Collaboration with Prof. Margaret Cheung)

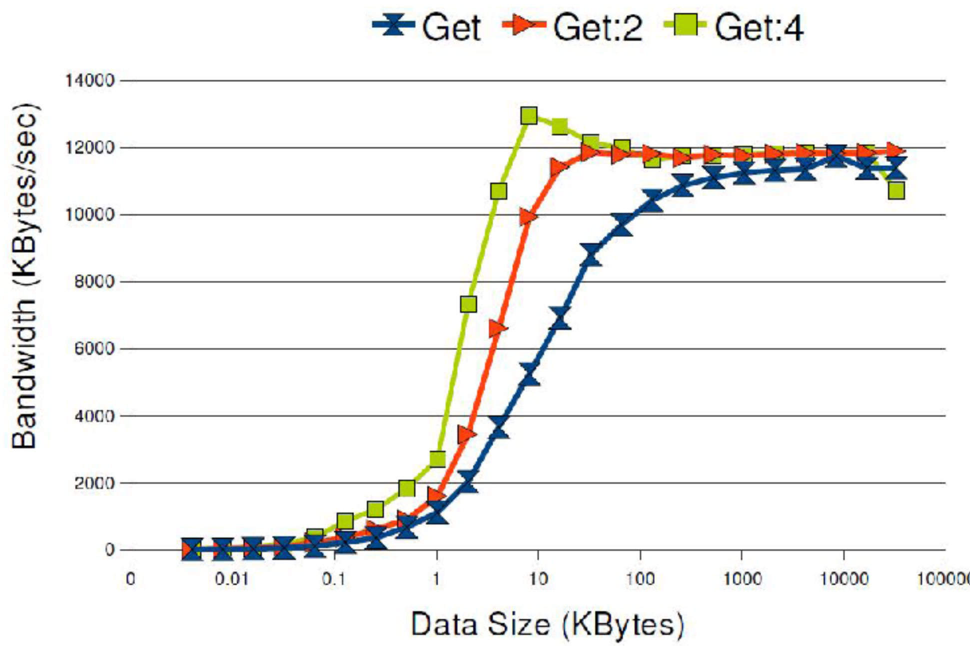
- Studying the folding thermodynamics of small to modest size proteins in explicit solvent.
- Use of Dataspace for modest communication
  - Exchange temperature/energy values between neighbors between simulation runs
  - Example run with 8 replicas (temperatures). *Processes that swap temperatures at a step have same background color*

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

BANDWIDTH: 'PUT/GET WITH REPLICAS
(Measured at Dataspace Server)

Microbenchmark code repeatedly does PUT/GET





No difference with replica PUTs that are ignored

Additional GET traffic with Replication saturates link earlier

N. Kanna, J. Subhlok, E. Gabriel, E. Rohit, and D. Anderson, *A Communication Framework for Fault-tolerant Parallel Execution.* The 22nd International Workshop on Languages and Compilers for Parallel Computing

CS@UH

Status and Discussion

General: Both Dataspace and MPI code bases are available to interested groups.

Yet several important developments are ongoing.

BOINC Integration: Dataspace has been tested with BOINC. Will be integrated better in coming months.

Should we work on MPI integration ?

Applications: Collaboration is critical. Keen to working with BOINC projects at all levels – from conceptual to code.

- **Thanks to NSF**

- jaspal@uh.edu   www.cs.uh.edu/~jsteach/volpex/