

Fault Tolerance in Charm++ and AMPI

Fault Tolerance: Motivation

- As machines grow in size
 - MTBF decreases
 - Applications have to tolerate faults
- Checkpoint/Rollback does not **scale**
 - All nodes are rolled back just because one crashed
 - Even nodes independent of the crashed node are restarted
 - Typically requires same configuration for restart

Background

- **Checkpoint-based methods**
 - Coordinated – Blocking [Tamir84], Non-blocking [Chandy85]
 - Co-check, Starfish, Clip – fault tolerant MPI
 - Uncoordinated – suffers from rollback propagation
 - Communication – [Briatico84], doesn't scale well
- **Msg-Log schemes**
 - Pessimistic – MPICH-V1 and V2, SBML [Johnson87]
 - Optimistic – [Strom85] unbounded rollback, complicated recovery
 - Causal Logging – [Elnozahy93] Manetho, complicated causality tracking and recovery

Early Implementations in Charm++

- **Disk-based Checkpointing**
 - Checkpointing implemented by “migrating” to disk file
 - Support for restart with different number of processors
 - Available for Charm++ and AMPI programs
- **In-Memory Checkpointing: FTC-Charm++**
 - Checkpoint data kept in memory
 - Very fast recovery at restart
 - Most useful for apps with low memory footprint
 - Implemented in Charm++ and AMPI
 - Described in [Zheng-04]

Current Research Directions

- **Message-Logging Fault Tolerance**
 - Keep log of messages, save them at checkpoint
 - Replay lost messages in case of faults
 - Low overhead on forward path (i.e. no faults)
- **Proactive Fault Tolerance**
 - Detect warnings of imminent faults
 - Migrate away from processors where faults are imminent
 - After migration, apply load-balance to achieve best performance with remaining resources

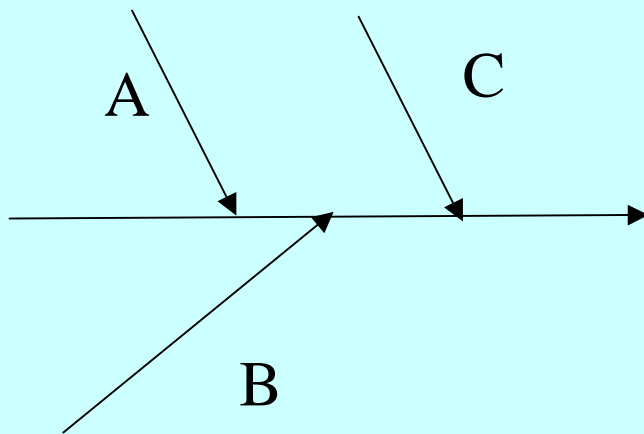
Message-Logging Design

- Message-Logging Scheme
 - Sender-side message logging
- Asynchronous checkpoints
 - Each processor has a buddy processor
 - Stores its checkpoint in the buddy's memory
- Processor Virtualization
 - Speed up restart, via migration

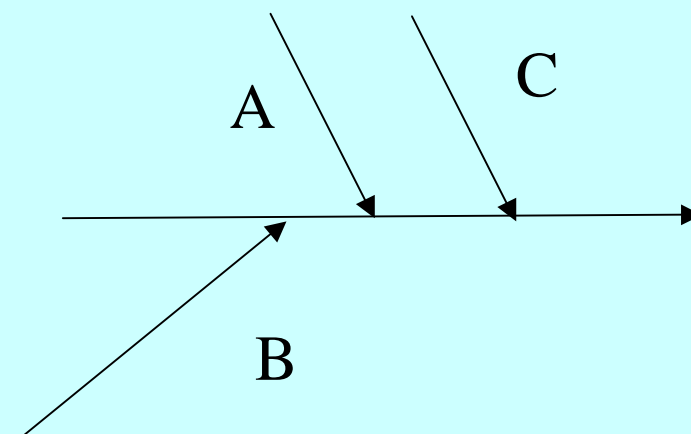
Message-Logging Protocol

Correctness: Messages should be processed in the **same order** before and after the crash

Problem:



Before Crash

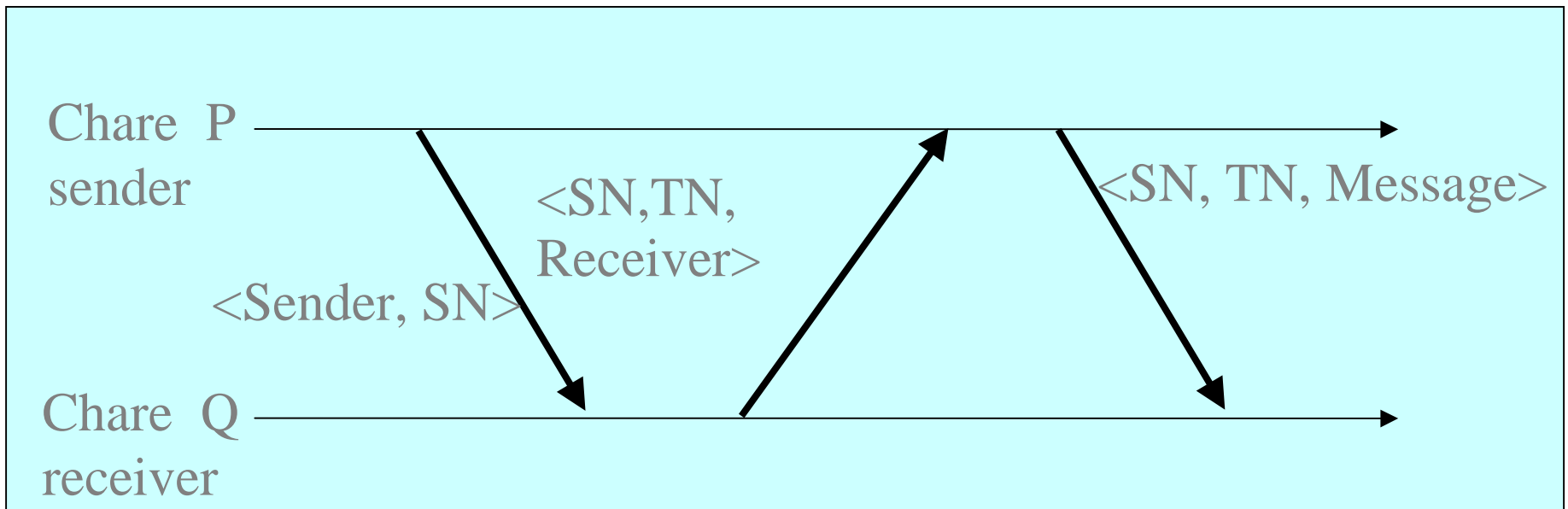


After Crash

Message-Logging Implementation

- Solution:
 - Fix an order the first time and always follow it
 - Assign to each received message a ticket number
 - Process messages in order of ticket number
- Each message contains
 - Sender ID – who sent it
 - Receiver ID – to whom was it sent
 - Sequence Number (SN) – together with sender and receiver IDs, identifies a message
 - Ticket Number (TN) – decide order of processing

Messaging Protocol

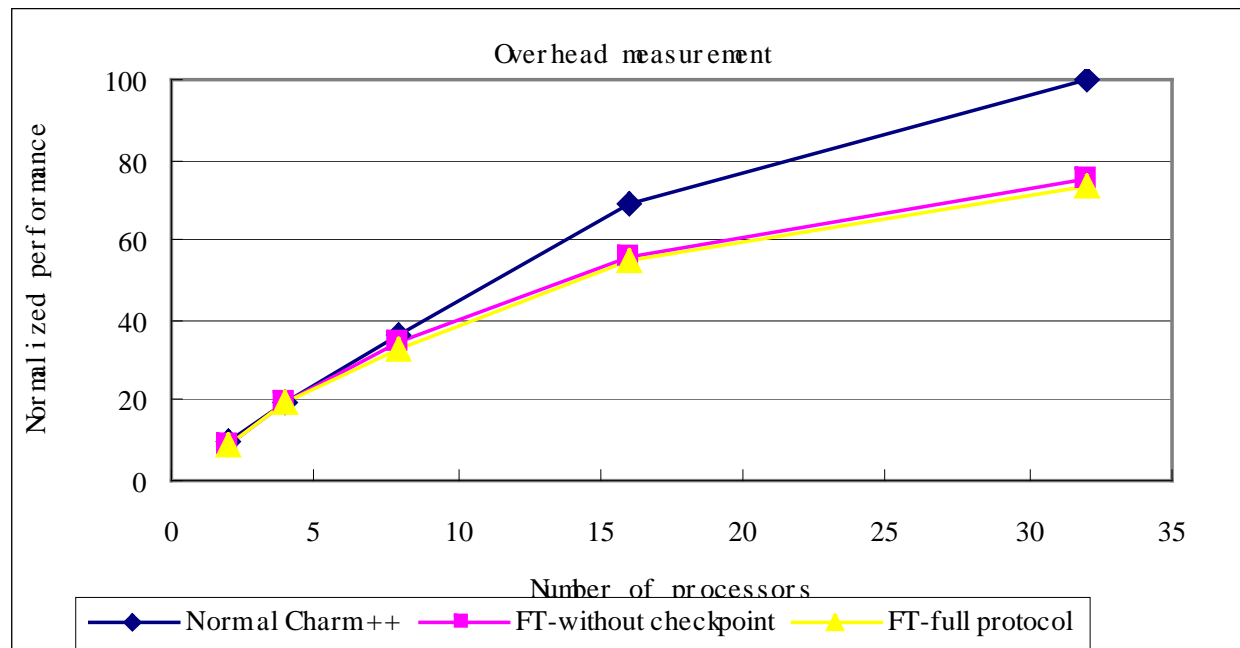


Checkpointing of Message Logs

- A processor asynchronously decides to checkpoint
- Packs up the state of all its chares and sends it to the buddy
 - Message logs are part of a chare's state
- Message log on senders can be garbage collected
- Deciding when to checkpoint is an interesting problem

Msg-Logging: Status and Performance

- Most of Charm++ and AMPI has been ported
 - Support for migration has not yet been implemented in the fault tolerant protocol
- Parallel restart not yet implemented
- Described in [Chakra-04]
- Performance (app with high comm/comp ratio):

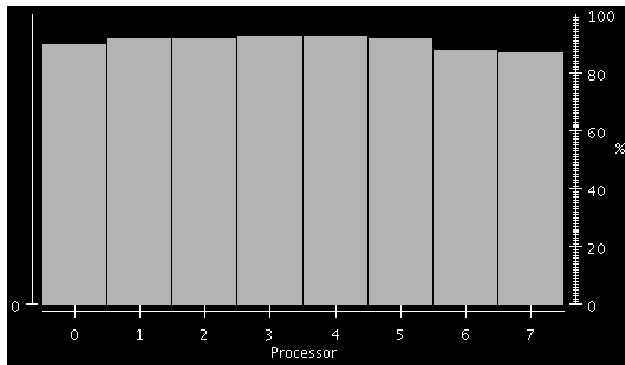


Proactive Fault Tolerance

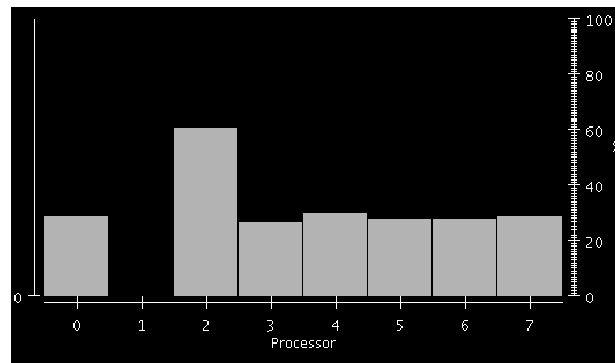
- **Rationale:** modern hardware supports early fault indication
 - SMART protocol, motherboard temperature sensors, Myrinet interface cards, etc.
 - Possible to create mechanism for fault prediction
- **Fault Tolerance Scheme:**
 - Detect that faults are imminent on a processor
 - Migrate all tasks from this processor
 - Apply load-balance to surviving processors
 - Backup to other schemes if faults not predictable

Proactive FT: Current Status

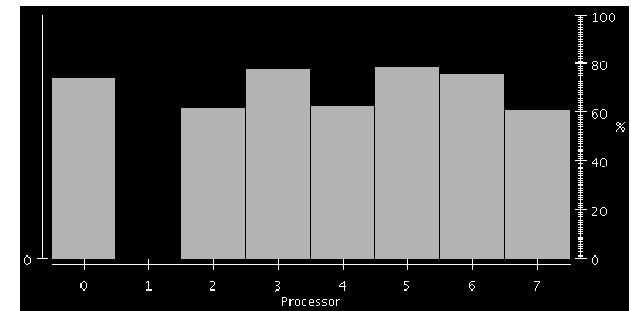
- **Status**
 - Support for multiple faults ready; currently testing support for simultaneous faults
 - Faults simulated via signal sent to process
 - Current version fully integrated to Charm++ and AMPI
- **Example:** sweep3d (MPI code) on NCSA's tungsten



Original utilization



Utilization after fault



Utilization after LB

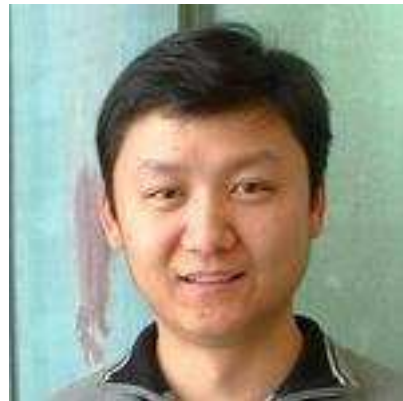
Fault Tolerance Team @ PPL



Sayantan Chakravorty



Celso Mendes



Gengbin Zheng