Fault Tolerance and the MPI standard meet at the Ultra-Scale

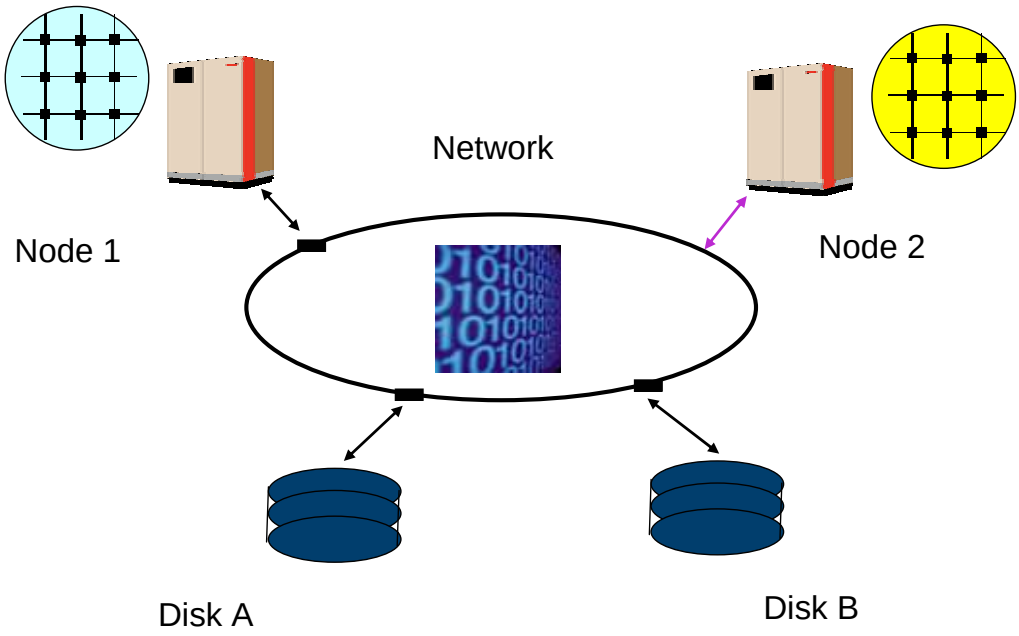
Richard L. Graham
Computer Science and Mathematics Division
National Center for Computational Sciences

Outline

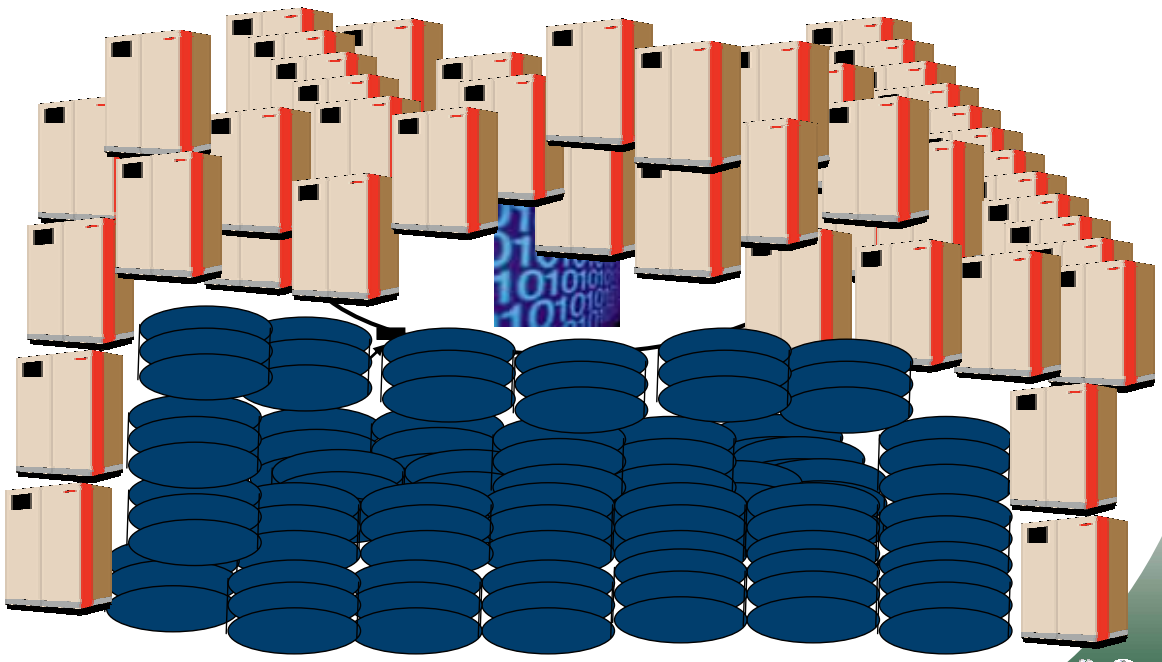
- **Problem definition**
  - **General**
  - **MPI Specific**
- **General approach for making MPI fault tolerant**
- **Current status**
- **Is this all ?**

Goal: Let MPI survive partial system failure

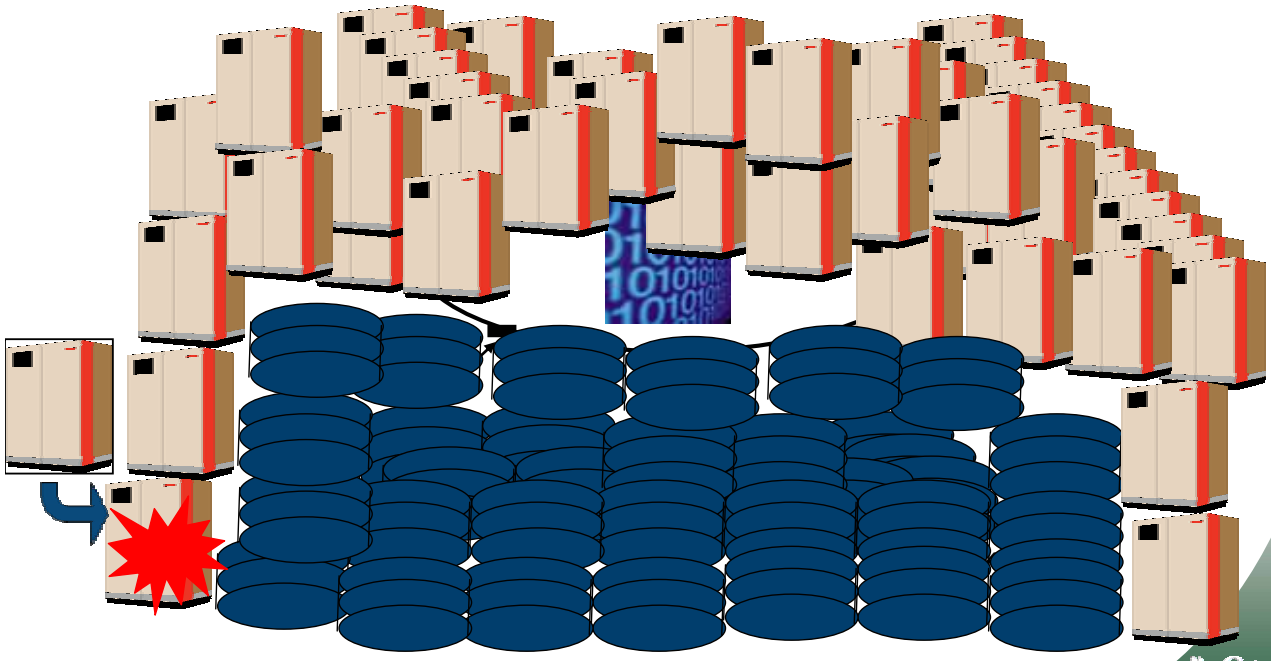
Problem definition



Problem definition – A bit more realistic

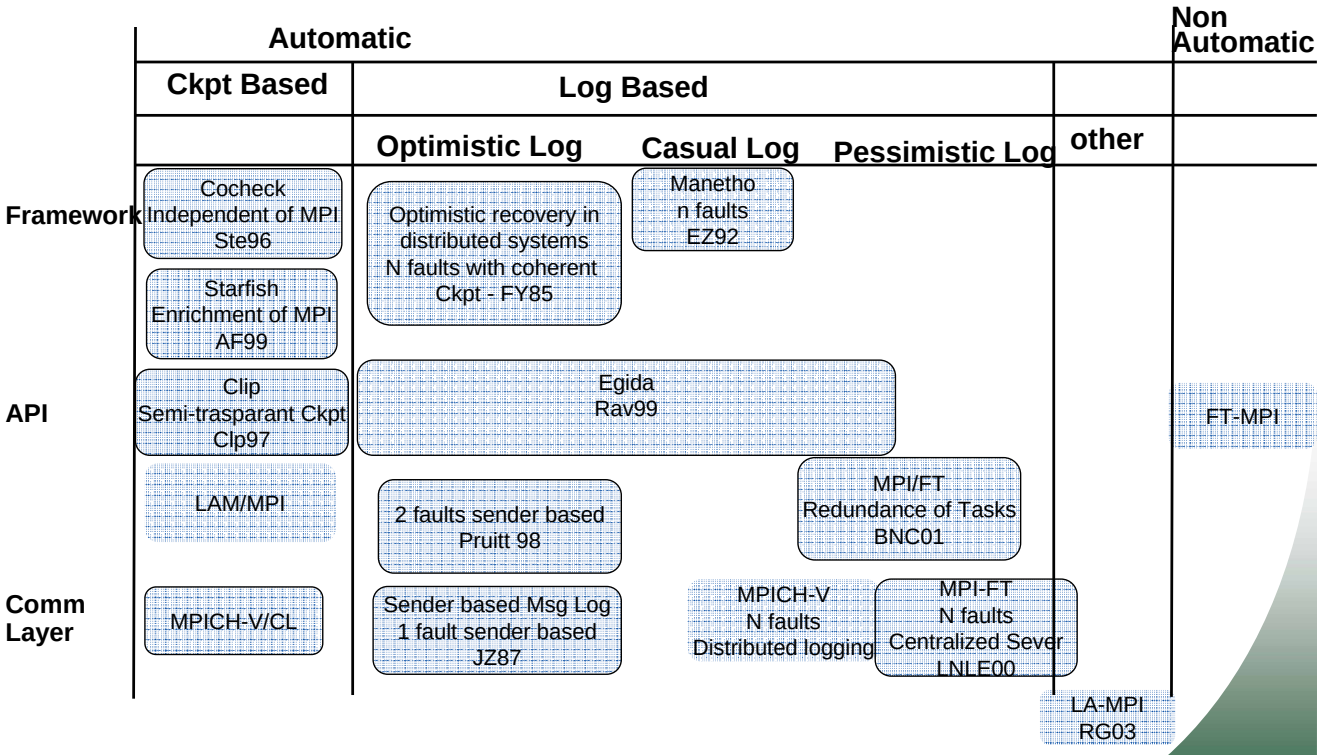


Failure example – node failure



- **Problem:** A component affecting a running MPI job is compromised (H/W or S/W)
- **Question:** Can the MPI application continue to run correctly ?
  - Does the job have to abort ?
  - If not, can the job continue to communicate ?
  - Can there be a change in resources available to the job ?

Related Work*



*Borrowed from Jack Dongarra

Why address this problem now ?

- **There have been quite a few predictions over the last decade that we would reach a scale at which hardware and software failure rates would be so high, we would not be able to make effective use of these systems.**

Why address this problem now ? – cont'd

- **This has not happened (?) ➔**

  Why should we believe it will happen this time ?

  Actually, we have adjusted

  Wasting resources

  Automating simple forms of recovery (restart – John Daly)

Why address this problem now ? – cont'd

- Systems are getting much larger
  - NCCS ~15000 ('07) -> ~30,000 cores('08) -> ~150,000 cores (end '08)
- Impact on the applications is increasing
- As we go to 1,000,000+ processes being used in a **single** job, application MTBF will suffer greatly

Why is Coordinated Checkpoint Restart not Sufficient ?*

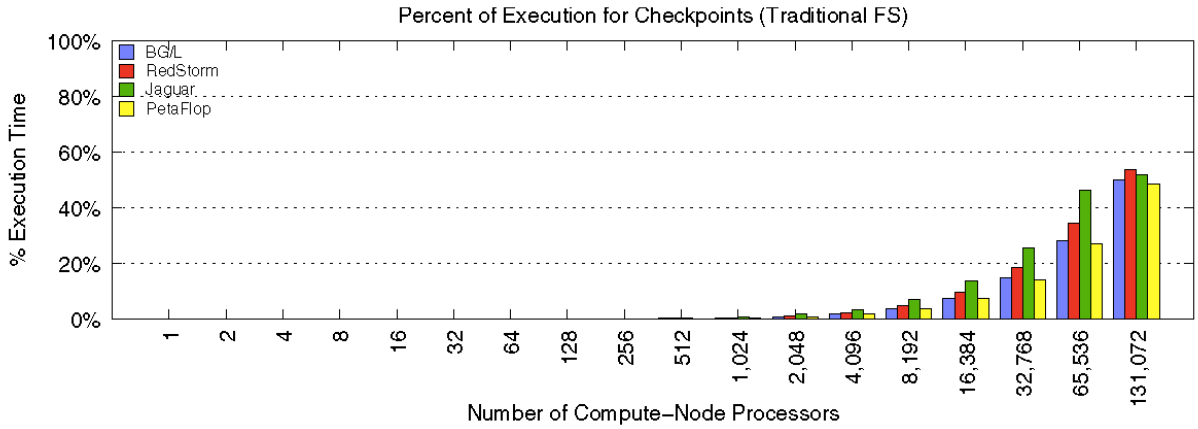


Figure 4. Aggregate checkpoint overhead as a percentage of application execution time.

Ron Oldfield, et al. – Modeling the Impact of Checkpoints on Next-Generation Systems

The Challenge

- **We need approaches to dealing with fault-tolerance at scale that will:**
  - **Allow applications to harness full system capabilities**
  - **Work well at scale**

Technical Guiding Principles

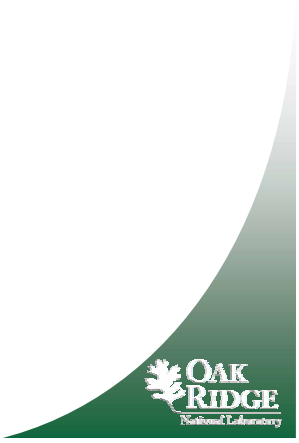
- **End goal: Increase application MTBF**
  - **Applications must be willing to use the solution**
- **No One-Solution-Fits-All**
  - **Hardware characteristics**
  - **Software characteristics**
  - **System complexity**
  - **System resources available for fault recovery**
  - **Performance impact on application**
  - **Fault characteristics of application**
- **Standard should not be constrained by current practice**

Why MPI ?

- **The Ubiquitous standard parallel programming model used in scientific computing today**
- **Minimize disruption of the running application**

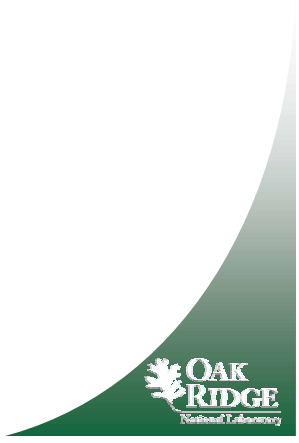
What role should MPI play in recovery ?

- **MPI does NOT provide fault-tolerance**
- **MPI should enable the survivability of MPI upon failure.**



What role should MPI play in recovery ? – Cont'd

- **MPI provides:**
  - **Communication primitives**
  - **Management of groups of processes**
  - **Access to the file system**



What role should MPI play in recovery ? – Cont'd

- **Therefore upon failure MPI should: (limited by system state)**
  - **Restore MPI communication infrastructure to correct and consistent state**
  - **Restore process groups to a well defined state**
  - **Able to reconnect to file system**
  - **Provide hooks related to MPI communications needed by other protocols building on top of MPI, such as**
    - **Flush the message system**
    - **Quiesce the network**
    - **Send "piggyback" data**
    - **?**

What role should MPI play in recovery ? – Cont'd

- **MPI is responsible for making the internal state of MPI consistent and usable by the application**
- **The "application" is responsible for restoring application state**

Layered Approach

CURRENT WORK

Active Members in the Current MPI Forum

Argonne NL, Bull, Cisco,

Cray, Fujitsu,

HDF Group, HLRS,

HP, IBM, INRIA,

Indiana U., Intel,

Lawrence Berkeley NL,

Livermore NL, Los Alamos NL,

Mathworks, Microsoft,

NCSA/UIUC, NEC,

Oak Ridge NL , Ohio State U.,

Pacific NW NL,

Qlogic, Sandia NL,

SiCortex, Sun Microsystems,

Tokyo Institute of Technology,

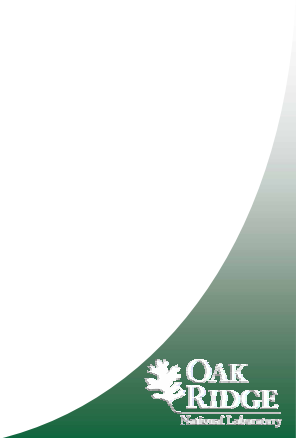
U. Alabama Birmingham,

U. Houston,

U. Tennessee Knoxville,

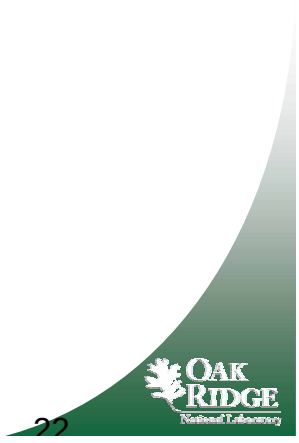
U. Tokyo

Motivating Examples



Collecting specific use case scenarios

- **Process failure – Client/Server, with client member of inter-communicator**
  - **Client process fails**
  - **Server is notified of failure**
  - **Server disconnects from Client inter-communicator, and continues to run**
  - **Client processes are terminated**



Collecting specific use case scenarios

- **Process failure – Client/Server, with client member of intra-communicator**
  - **Client process fails**
  - **Processes communicating with failed process are notified of failure**
  - **Application specifies response to failure**
    - **Abort**
    - **Continue with reduced process count, with the missing process being labled MPI_Proc_null in the communicator**
    - **Replace the failed process (why not allow to increase the size of the communicator ?)**

Collecting specific use case scenarios

- **Process failure – Tightly coupled simulation, with independent layered libraries**
  - **Process fails**
    - **Example application: POP (ocean simulation code) using conjugate gradient solver**
  - **Application specifies MPI's response to failure**

Design Details

- **Allow for local recovery, when global recovery is not needed (scalability)**
  - **Collective communications are global in nature, therefore global recovery is required for continued use of collective communications**
- **Delay recovery response as much as possible**
- **Allow for rapid and fully coordinated recovery**

Design Details – Cont'd

- **Key component: Communicator**
  - **A functioning communicator is required to continue MPI communications after failure**
- **Disposition of active communicators:**
  - **Application specified**
  - **MPI_COMM_WORLD must be functional to continue**
- **Handling of surviving processes**
  - **MPI_comm_rank does not change**
  - **MPI_Comm_size does not change**

Design Details – Cont'd

- **Handling of failed processes**
  - **Replace**
  - **Discard (MPI_PROC_NULL)**
- **Disposition of active communications:**
  - **With failed process: discard**
  - **With other surviving processes – application defined on a per communicator basis**

Error Reporting Mechanisms – current status

- **Errors are associated with communicators**
- **By default errors are returned from the affected MPI calls, are are returned synchronously**

Example:

Ret1=MPI_Isend(comm=MPI_Comm_world, dest=3, …request=request3)

Link to 3 fails

Ret2=MPI_Isend(comm=MPI_Comm_world, dest=4, …request=request4)

Ret3=Wait(request=request4) // success

Ret4=Wait(request=request3) // error returned in Ret

Can ask for more information about the failure

- **May request global event notification**
  - **Several open questions remain here**

- **A collective call has been proposed to check on the status of the communicator**
  - **No extra cost is incurred for consistency checks, unless requested (may be relevant for collective operations)**
  - **Provides global communicator state just before the call is made**

Examples of proposed API modifications Surviving Processes

- **MPI_COMM_IRECOVER(ranks_to_restore, request, return_code)**
  - **IN ranks_to_restore array of ranks to restore (struct)**
  - **OUT request request object (handle)**
  - **OUT return_code return error code(integer)**
- **MPI_COMM_IRECOVER_COLLECTIVE(ranks _to_restore, request, return_code)**
  - **IN ranks_to_restore array of ranks to restore (struct)**
  - **OUT request request ob ject (handle)**
  - **OUT return_code return error code(integer)**

Examples of proposed API modifications Restored Processes

- **MPI_RESTORED_PROCESS(generation, return_code)**
  - **OUT generation Process generation (integer)**
  - **OUT return_code return error code (integer)**
- **MPI_GET_LOST_COMMUNICATORS(comm_names, count, return_code)**
  - **OUT comm_names Array of communicators that may be restored (strings)**
  - **OUT count Number of Communicators that may be restored (in- teger)**
  - **OUT return_code return error code(integer)**

Examples of proposed API modifications Restored Processes – Cont'd

- **MPI_COMM_REJOIN(comm_names, comm, return_code)**
  - **IN comm_names Communicator name (string)**
  - **OUT comm communicator (handle)**
  - **OUT return_code return error code(integer)**

Open Questions

- **How heavy handed do we need to be at the standard specification level to recover from failure in the middle of a collective operation ? Is this more than an implementation issue ? (Performance is the fly in the ointment)**
- **What is the impact of "repairing" a communicator on implementation of collective algorithms (do we have to pay the cost all the time?)**

Is this All ?

- **Other aspects of fault tolerance**
  - **Network**
  - **Checkpoint/Restart**
  - **File I/O**
- **End-to-end solution**

For involvement in the process see:

meetings.mpi-forum.org

Backup slides