Spare Node Substitution for Failure Nodes

Kazumi Yoshinaga
RIKEN AICS
Background

• In the Exa-flops era, faults could happen more frequently than ever → System MTBF becomes shorter
• Important Issue : Recovery from faults

• Conventional method : System-level Checkpoint-Restart
  – Requires massive I/O
• Many mechanisms to survive failures have been proposed and investigated
  – Less I/O Size
  – One of the mechanisms is ULFM(User-Level Fault Mitigation).
    • User program handles failures
    • The program can survive from the failures and continue its execution
• But there is no discussion how a job should survive from node failures
Purpose of this Research

• **What is the best way to survive from node failures?**
  – Assuming a job can survive from a node failure by using an existing fault mitigation software
  – Not to propose a new fault mitigation mechanism
  – Propose recovery strategy
Survival from Node Failure

• Applications with dynamic load balancing
  – e.g. Distributed Master-Worker model
  – Avoiding failure nodes method
  – Applications continue its execution only with healthy nodes after failure

• How about applications **without** dynamic load balancing?
  – e.g. Stencil Computation
Avoiding Failure Node(s) for Stencil Computation

- Stencil computation characteristics
  - Communication pattern is fixed
  - Load can be balanced
- When a recovery happens, above stencil computation characteristics must be preserved
- However,
  - Hard to balance loads
  - Impossible to preserve communication pattern
  - Every time a new failure happens, communication pattern can differ
- Hard to program !!!

Using spare nodes to solve these problems
Using Spare Nodes

• An application runs with spare nodes
• If node failure happens, migrate the task running on failed node to the spare node
  – Loads are balanced (continues with the same # procs.)
  – Preserve logical communication pattern
  – No change in the kernel part of application
  – Some penalties
Spare Node Penalty-1
-System utilization Degradation-

- Spare node allocation
- System utilization is decreased

\[ nD(\alpha, \beta) \]
- \( n \): Dimensions of networks
- \( \alpha \): # dimensions of spare nodes
- \( \beta \): spare nodes width
Spare Node Penalty-2
-Communication Performance Degradation-

• **Logical** communication pattern can be preserved
  • by creating a new MPI communicator to exclude the failed node and include a spare node.

• However, **physical** communication pattern is not the same, and communication performance (CP) can be degraded.
  • Larger hop counts (latency), and
  • Possible message collisions
Ex. CP Degradation of Spare Node Substitution

• Nodes on the topmost row work as spare nodes

• Up to 5 possible collisions after 1 node failure
  – Independent from the # nodes

How faulty nodes should be replaced by spare nodes?
We proposed “Sliding Substitution” methods

- 0D Sliding (simple replace)
  - Failed rank is continued on an alternative node

- 1D Sliding
  - Processes between the failure node and the spare node are shifted

- 2D Sliding
  - Whole processes between the failure node's row(column) and the spare node's row(column) are shifted

- 3D Sliding, 4D, 5D...
Preliminary Evaluation

-5D stencil on 2D network-

• Spare Allocation
  2D(2,1) > 2D(1,1)

• Max. Failure
  – 0D: up to # Spare
  – 1D: 3 (or more)
  – 2D: up to 2 (2D Cart. Topo.)

• Comm. Perf.
  2D > 1D > 0D
Sliding Substitution (2)

- The higher the dimension
  - The better the performance
  - The smaller the number of the failure nodes it can handle
- 2D or higher dimension Sliding
  - Migrate tasks running on healthy nodes
  - Free nodes works as new spare nodes
- Hybrid Sliding
  - 3D → 2D → 1D → 0D (on 3D network)
Evaluation: 7P-Stencil on the K and BG/Q (Hybrid, 3D(2,1), 4MiB)

- K computer: up to 8 times slower
- BG/Q: up to 12 times slower

The K Computer
12x12x12 Nodes (calc. 11x11x12)

BG/Q
16x8x8 Nodes (calc. 15x7x8)

Smaller is better
Evaluation: Collectives on the K and BG/Q
(Hybrid, 3D(2,1))

- On the K and BG/Q, collective operations are optimized for their network
- Having spare nodes makes the optimization very difficult
- BG/Q’s optimization works only with MPI_COMM_WORLD
Summary

• We proposed and compared “Sliding Substitution” methods.

• Communication performance degradation is observed
  – 7P-Stencil:
    • Simulation results: up to 40 collisions
    • Experimental results: up to 12 times larger latency
  – Collective communications:
    • up to 12 times larger latency (BG/Q, Barrier)
Future Work

• Evaluations with real applications
• Node-Rank re-mapping algorithms, or better substitution methods
• Discussion on the other network topology
  – Experiments using Tsubame 2.5 (Fat-tree) is scheduled