Checkpointing and Recovery of Shared Memory Parallel Applications in a Cluster

R. Badrinath – INRIA / IIT Kharagpur (India)
C. Morin – INRIA (France)
G. Vallee – EDF R&D (France)
Introduction

• Today, clusters are widely used as alternative of small parallel computer
• Targeted applications
  • Sequential applications
  • Message Passing applications
  • Shared Memory applications using a DSM
• An approach : Single System Image (SSI) (federation of all resources)
  • Failures of cluster nodes (due to the hardware): Fault Tolerance mechanisms
Kerrighed

• Kerrighed OS (INRIA - PARIS research team) is an OpenSource project (http://www.kerrighed.org)
• The current Kerrighed OS
  • Linux kernel +
  • Process Management Module (Aragorn)
  • Memory Management Module (Gandalf)
  • IPC object Management Module (Elrond)
  • Communication Management Module (Gimli/Gloin)
Background

• To support fault tolerance in a SSI
  • distributed system / cluster support DSM and MP
  • we use checkpointing and recovery to do this

• We have many options
  • coordinated checkpointing
  • uncoordinated checkpointing
  • partly coordinated checkpointing
    • … and corresponding recovery strategies too!

• We wish to implement common mechanisms
• We wish to try out several strategies
Goals

- Support checkpointing and recovery of parallel applications using both shared memory and message passing in a cluster.
- Support through some set of basic common mechanisms.
- Support to experiment with a variety of checkpoint / recovery protocols.
Outline

• Basic common mechanisms
  • Dependency tracking => memory
• Implementation of checkpoint policy: the case of coordinated checkpoint
Basic Common Mechanisms: Recovery lines

- Checkpointing must be such that we can compute recovery lines and rollback to them.
- What is a recovery line?

A recovery line is a consistent set of checkpoints

Consistent wrt send-receive relation.
Basic Common Mechanisms: Dependencies

- To compute a recovery line requires to track dependencies caused by interactions.
- For a DSM, Interactions include:
  - DSM Read
  - DSM Write
- We will treat shared memory as entities!
Basic Commom Mechanisms: The Theory Says...

- Many things…
- We need to
  - Track *direct* dependencies between entities
  - Create the dependence graph
  - Compute the latest recovery line by reachability analysis
- In Practice
  - … we need to support optimisations
Basic Common Mechanisms: The Theory to Track Dependencies

Sequence number

Checkpoint Forced by entity 2

entity 1 (task)

entity 2 (task)

entity 3 (page)

entity 4 (task)

Sequence number

Direct Dependency Vector (DDV)

Timestamp
Basic Common Mechanisms: Dependency tracking

- Add data items
  - A sequence number \((sn)\)
  - A DDV

- Update actions on interactions
  - \(sn\)
  - \(DDV\)

- On checkpoint
  - Save timestamp, update \(sn\)

- On recovery
  - Re-initialise timestamps, \(sn\)
Basic Common Mechanisms: Entity states

- Task private states
  - process management module
- Shared memory states
  - in cooperation with memory manager module
- IPC states:
  - Locks, barriers states (native support of checkpointing in Kerrighed)
Policies - Checkpoint/Restart Protocols

• Have threads to run the protocol for each entity.
  • Threads participate in:
    • Checkpointing
    • Restart / Rollback

• This would be an implementation of the policy.
  • We will initially focus on coordinated checkpointing and recovery.
Policy: How checkpointing works
Policy: Coordinated Checkpointing

- Interaction phase - *Nothing special*

- Checkpoint phase -
  - Checkpoint coordinator
    - Maybe the checkpoint initiator application process
    - Maybe an application call from one of the tasks themselves.
  - Protocol thread on behalf of each entity
    - Kernel mode of a task
    - Page manager thread
  - Optimisations may require a more elaborate protocol.
    - Ex. Not all entities need to be coordinated always.
Policy: Coordinated Recovery

- Recovery phase -
  - Recovery initiator/coordinator
  - Recovery protocol handler for each entity
    - We may use kernel mode threads for tasks
    - Page managers for pages
  - Optimisations make use of the dependency tracking mechanism.
Prototype: Status

• We have a simple coordinator
  • System initiated coordinated checkpoint.
  • Protocol aborts if tasks not in « good » state.
  • Incremental Checkpointing of DSM states.
  • Recovery not yet complete.

• Checkpointing performance results say that the checkpointing protocol is not a bottleneck.

• Checkpoint saving time is the bottleneck
  • Motivation for a high speed reliable storage..
    (containers for checkpoints?)
Effect of Checkpointing on Execution Time

![Bar chart showing the effect of checkpointing on execution time for different scenarios.](chart.png)
Effect of Destination on Checkpointing Time

- **Rem**: remote node RAM
- **NFS**: an NFS directory
- **Dsk**: local disk
- **Mem**: local node RAM

**Checkpoints Size (4KB pages)**

- 1024
- 3072
- 4096
Conclusion / Future works

- **Common mechanisms** can support dependency tracking involving different IPC forms.
- **Important to design** tasks/memory/barriers in a way as to support « checkpointing » and « recovery » of their states.
- Coordinated checkpointing has been demonstrated on Kerrighed…. and motivates work on storage systems.
- Implement new checkpoint protocols
- Integrate to the Kerrighed's version available on the web
Thanks!