Lightweight GPGPU Checkpoint Modelings

Presenter: Box. Leangsuksun
SWEPCO Endowed Professor*, Computer Science
Louisiana Tech University
box@latech.edu

S. Laosooksathit, N. Naksinehaboon, Box. Leangsuksun, A. Dhungana, C. Chandler
Louisiana Tech U

K. Chanchio
Thammasat Univ

Amir Fabin
U of Texas, Arlington

4th HPCVirt workshop, Paris, France, April 13, 2010
Outlines

- Motivations
- Background – VCCP
- GPU checkpoint protocols: Memcopy vs simpleStream
- CheCUDA (related work)
- GPU checkpoint protocols: CUDA Streams
- Restart protocols
- Scheduling model and Analysis
- Conclusion
Motivations

- More attention on GPUs
- ORNL–NVDIA 10 petaflop machine
- Large scale GPU cluster → fault tolerance for GPU applications
  - Normal checkpoint doesn’t help GPU applications when a failure occurs.
  - GPU execution isn’t saved when do checkpoint on CPU
VCCP: A Transparent, Coordinated Checkpointing System for Virtualization-based Cluster Computing – GOALS

- **High transparency**
  - Checkpoint/restart mechanisms should be *transparent* to applications, OS, and runtime environments; no modification required

- **Efficiency**
  - Checkpoint/restart mechanisms should *not* generate unacceptable overheads
    - Normal Execution
    - Communication
    - Checkpointing Delay
Virtual Cluster Architecture

Run apps/OS unmodified

Checkpoint/restart protocols

FIFO, Reliable
Virtual Cluster CheckPointing (VCCP) Protocol

1. Pause VM computation
2. Flush messages out of the network
3. Locally Save State of every VM
4. Continue computation
VCCP checkpoint protocol
VCCP checkpoint protocol

- Head
- compute0
- compute0

- channel empty
- save VM & buffer
- channel empty
- save VM & buffer
- channel empty
- save VM & buffer

Flush communication channel
VCCP checkpoint protocol
More details in VCCP

- Publication in IEEE cluster 2009
- Average overhead 12%
- Provide transparent checkpoint/restart
Heterogeneous Computing – GPGPU

1. Device Initialization
2. Device memory allocation
3. Copies data to device memory
4. Executes kernel (Calling __global__ function)
5. Copies data from device memory (retrieve results)

- Issues – latency round trip data movement
Long running GPU application

High (relatively) failure rate in a large scale GPU cluster in MPI & GPU environment

Save GPU software state

Move data back from GPU in low latency

- Memcopy (pause GPU) vs simpleStream (concurrency)
Related Work (CheCUDA)

- “CheCUDA: A Checkpoint/Restart Tool for CUDA Applications” by H. Takizawa, K. Sato, K. Komatsu, and H. Kobayashi
- A prototype of an add-on package of BLCR for GPU checkpointing
- Memcopy approach
GPGPU Checkpoint protocols

1. GPU checkpointing
2. Migration/ CPU checkpoint

2. CPU checkpointing

Reliable Storage
GPU checkpoint protocol: memcpy

1. Process starts
2. H–D memory copy
3. CPU checkpoint/migration
4. GPU checkpoint duration
5. Kernel completes
6. D–H memory copy
7. Process ends
CheCUDA: Checkpoint Protocol

1. Copying all the user data in the device memory to the host memory
2. Writing the current status of the application and the user data to a checkpoint file
CheCUDA: Restart protocol

1. Read the checkpoint file
2. Initialize the GPU and recreating CUDA resources
3. Sending the user data back to the device memory
GPU checkpoint protocol: memcpy vs simpleStream

- Transfer data from device to host = overhead
  - Must pause GPU computation until the copy is completed

- SimpleStream
  - Using latency hiding (Streams) to reduce the overhead
  - CUDA streams = overlap memory copy and kernel execution
GPU checkpoint protocol: Streams

1. **Process starts**
2. **H–D memory copy**
3. **Kernel starts**
4. **CPU checkpoint/migration**
5. **GPU checkpoint duration**
6. **Syncthread()**
7. **Kernel completes**
8. **D–H memory copy**
9. **Process ends**

**Code Analysis**

After the sync point, OVERWRITE?

- YES
- NO

4th HPCVirt workshop, Paris, France, April 13, 2010
GPU checkpoint protocol: ensure consistency

Process starts
H-D memory copy

Kernel starts

Code Analysis
After the sync point, OVERWRITE?

YES
NO
After the sync point, OVERWRITE?

- NO

Process starts
H–D memory copy

Kernel starts

Syncthread()

GPU checkpoint duration

Kernel completes

D–H memory copy

Process ends
GPU Checkpoint Protocol: Streams (cont.)

- Process starts
- H–D memory copy
- After the sync point, OVERWRITE?
- Duplicate image
- CPU checkpoint/migration
- D–H memory copy
- Copy the sync image in GPU
- GPU checkpoint duration
- Kernel completes
- Syncthread()
- Process ends

4th HPCVirt workshop, Paris, France, April 13, 2010
GPU Restart Protocol

- Restart CPU
- Transfer the last GPU checkpoint back to CPU
- Recreate CUDA context from the CKpt file
- Restart the kernel execution from the marked synchronization point
Scheduling Model

- GPU checkpoint after a thread synchronization
- NOT every thread synchronization
- QUESTION???
  - Which thread synchronization should a checkpoint be invoked?

- FACTORs
  - GPU checkpoint overhead
  - Chance of a failure occurrence
Scheduling model (cont.)

Perform the checkpoint:

\[ P_f \left( \sum_{j=m}^{n} C_j + \hat{C} \right) + \left( 1 - P_f \right) O \]
Scheduling model (cont.)

Skip the checkpoint:

\[ P_f \left( \sum_{j=m}^{n} C_j + \hat{C} \right) \]

Perform the checkpoint:

\[ P_f \left( O + \hat{C} \right) + \left( 1 - P_f \right) O \]
Model Analysis

- Simulate failures & the wasted time
  - total checkpoint overhead + re-computing due to a failure
- Overhead
  - Non-stream: 10 milliseconds – 3 seconds
  - Streams: negligible
- MTTF: 12 hours – 7 days
- Thread sync interval: 10 and 30 minutes
Results (various overhead = size of transfer)

Thread sync interval = 10 mins

Thread sync interval = 30 mins
Results (various MTTF)

Thread sync interval = 10 mins

Thread sync interval = 30 mins
Results (skipped VS non-skipped)

- Against MTTFs
- Against overheads
Conclusions

- GPU checkpointing with Stream to reduce overhead
- Non-stream and stream checkpoints are insignificantly different if data transfer is insignificant
- BUT stream checkpoint potentially performs better when the checkpoint overhead of memcpy is larger.
Future work

- Implement GPU checkpoint/restart mechanism
- Work on other checkpoint protocol
- Include GPU process migration