Surviving in the Petascale World [and Beyond]

George Bosilca
Innovative Computing Laboratory
University of Tennessee
Number of Processors Share Over Time
1993-2008

TOP500 Releases

Systems

128k-
64k-128k
32k-64k
16k-32k
8k-16k
4k-8k
2049-4096
1025-2048
513-1024
257-512
129-256
65-128
33-64
17-32
9-16
5-8
Classification of FT approaches

<table>
<thead>
<tr>
<th>Framework</th>
<th>API</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkpoint based</td>
<td>Cocheck [Ste96]</td>
<td>Semi-transparent checkpoint [CLP97]</td>
</tr>
<tr>
<td></td>
<td>Starfish [AF99]</td>
<td>Clip [CLP97]</td>
</tr>
<tr>
<td></td>
<td>Open MPI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LAM/MPI</td>
<td></td>
</tr>
<tr>
<td>Log based</td>
<td>Optimistic log</td>
<td>Optimistic recovery in distributed systems with coherent checkpoint [SY85]</td>
</tr>
<tr>
<td></td>
<td>Causal log</td>
<td>Manetho faults [EZ92]</td>
</tr>
<tr>
<td></td>
<td>Pessimistic log</td>
<td>Coordinated checkpoint</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semi Auto</td>
<td>OpenMPI-V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MPI/FT</td>
<td>Redundance of tasks [BNC01]</td>
</tr>
<tr>
<td></td>
<td>MPI-FT</td>
<td>N fault centralized server [LNLE00]</td>
</tr>
<tr>
<td></td>
<td>LA-MPI</td>
<td>Communications rerouting</td>
</tr>
<tr>
<td></td>
<td>FT-MPI</td>
<td>Modification of MPI routines User Fault Treatment [FD00]</td>
</tr>
</tbody>
</table>
FT a complex solution

**Transparency**
- application ckpt: application stores intermediate results and restart form them
- MP API+FM: message passing API returns errors to be handled by the programmer
- **automatic**: runtime detects faults and handle recovery

**Checkpoint coordination**
- coordinated: all processes are synchronized, network is flushed before ckpt; all processes rollback from the same snapshot
- **uncoordinated**: each process checkpoint independently of the others; each process is restarted independently of the others
FT a complex solution

Message logging

- **pessimistic**: all messages are logged on reliable media and used for replay
- **optimistic**: all messages are logged on non-reliable media. If 1 node fails, replay is done according to other nodes’ logs. If >1 node fail, rollback to last coherent checkpoint.
- **causal**: optimistic + Antecedence Graph, reduces the recovery time.
The problem of inconsistent states

- Order of message receptions are non-deterministic events
- Messages received but not sent are inconsistent
- Domino effect can lead to rollback to the beginning of the execution in case of even a single fault
- Possible loose of the whole execution and unpredictable fault cost
Deterministic Recovery

- Deterministic replay is based on *Event Logging*
- *Piecewise Deterministic* assumption (even suitable for monte carlo applications)
- Each recv is an event (src,send-clk,recv-clk)
- Send the ordering of events to stable storage (event logger)
Benchmark Performance

- Number of logged events to total number of messages

<table>
<thead>
<tr>
<th></th>
<th>BT</th>
<th>SP</th>
<th>FT</th>
<th>CG</th>
<th>MG</th>
<th>LU</th>
</tr>
</thead>
<tbody>
<tr>
<td>#processors</td>
<td>all</td>
<td>4</td>
<td>32</td>
<td>64</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>32</td>
<td>64</td>
<td>256</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>%non-deterministic</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40.33</td>
<td>29.35</td>
</tr>
<tr>
<td></td>
<td>27.10</td>
<td>22.23</td>
<td>20.67</td>
<td>19.99</td>
<td>1.13</td>
<td>0.66</td>
</tr>
</tbody>
</table>

- Impact on latency of forced message logging (Infiniband)
FT-MPI: Why and How?

- MPI is the de-facto programming model for parallel applications
- MPI Standard: "Advice to implementors: A good quality implementation will, to the greatest possible extent, circumvent the impact of an error, so that normal processing can continue after an error handler was invoked."
- Define the behavior of MPI [state] in case an error occurs and give the application the possibility to recover from a node-failure
- A regular, non fault-tolerant MPI program will run using FT-MPI
- Follows the MPI-1 and MPI-2 specification as closely as possible (e.g. no additional function calls)
- On error user program must do something (!)
Recovery modes

- ABORT, BLANK, SHRINK and REBUILD
- REBUILD: a new process is created, and it will return MPI_INIT_RESTARTED_PROC from MPI_Init
- BLANK: dead processes replaced by MPI_PROC_NULL, all communications with such a process succeed, they do not participate in the collectives
  - two sub-modes: local and global
Shallow Water (PSTSWM) & HPL

HPL

PSTSWM

ICL
Diskless checkpointing

- **P1** + **P2** + **P3** + **P4** = **Pc**
  - Add a fifth and perform a checkpoint (Allreduce)

- **P1** + **P2** + **P3** + **P4** + **Pc**
  - Ready to continue

- **Failure**

- **P1** + **P2** + **P3** + **P4** + **Pc**
  - Ready for recovery

- **Pc** - **P1** - **P3** - **P4** = **P2**
  - Recover the processor/data
Diskless Checkpointing

• How to checkpoint?
  – either floating-point arithmetic or binary arithmetic will work
  – If checkpoints are performed in floating-point arithmetic then we can exploit the linearity of the mathematical relations on the object to maintain the checksums

• How to support multiple failures?
  – Reed-Salomon algorithm
  – support p failures require p additional processors (resources)
PCG

Fault Tolerant CG

- Performance of PCG with different MPI libraries

<table>
<thead>
<tr>
<th>Size of the Problem</th>
<th>Num. of Comp. Procs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prob #1</td>
<td>164,610</td>
</tr>
<tr>
<td>Prob #2</td>
<td>329,220</td>
</tr>
<tr>
<td>Prob #3</td>
<td>658,440</td>
</tr>
<tr>
<td>Prob #4</td>
<td>1,316,880</td>
</tr>
</tbody>
</table>
PCG

<table>
<thead>
<tr>
<th>Time</th>
<th>Prob #1</th>
<th>Prob #2</th>
<th>Prob #3</th>
<th>Prob #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ckpt</td>
<td>2.6</td>
<td>3.8</td>
<td>5.5</td>
<td>7.8</td>
</tr>
<tr>
<td>2 ckpt</td>
<td>4.4</td>
<td>5.8</td>
<td>8.5</td>
<td>10.6</td>
</tr>
<tr>
<td>3 ckpt</td>
<td>6.0</td>
<td>7.9</td>
<td>10.2</td>
<td>12.8</td>
</tr>
<tr>
<td>4 ckpt</td>
<td>7.9</td>
<td>9.9</td>
<td>12.6</td>
<td>15.0</td>
</tr>
<tr>
<td>5 ckpt</td>
<td>9.8</td>
<td>11.9</td>
<td>14.1</td>
<td>16.8</td>
</tr>
</tbody>
</table>

Checkpoint overhead in seconds

PCG Checkpoint Overhead

PCG Recovery Overhead

Number of Checkpoint Processors

Number of Failed Processors
• The algorithm maintains the consistency of the checkpoints of the matrix C naturally
Failure Overhead

- FT-MPI will take care of the fault management
- Once the new process joins the MPI_COMM_WORLD we have to rebuild the communicators
- Then we have to retrieve the data from the checkpoint processor
PBLAS vs. ABFT BLAS (no failure)
Strong Scalability
Conclusion

• Fault Tolerance is a requirement
• Which model is the best depend on many factors
  – FT-MPI is a viable approach with algorithms already available.