Surviving in the Petascale World [and Beyond]

George Bosilca
Innovative Computing Laboratory
University of Tennessee
Classification of FT approaches

<table>
<thead>
<tr>
<th>Framework</th>
<th>Automatic</th>
<th>Semi Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Checkpoint based</td>
<td>Log based</td>
</tr>
<tr>
<td></td>
<td>Optimistic log</td>
<td>Causal log</td>
</tr>
<tr>
<td>Cocheck</td>
<td>Independent of MPI</td>
<td>[CLP97]</td>
</tr>
<tr>
<td>Starfish</td>
<td>N/A</td>
<td>[EZ92]</td>
</tr>
<tr>
<td>Clip</td>
<td>N/A</td>
<td>[RAV99]</td>
</tr>
<tr>
<td>Open MPI</td>
<td>N/A</td>
<td>[BNC01]</td>
</tr>
<tr>
<td>LAM/MPI</td>
<td>N/A</td>
<td>[LNLE00]</td>
</tr>
</tbody>
</table>

FT a complex solution

**Transparency**
- application ckpt: application stores intermediate results and restart form
- 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)

**Event Logger**

---

Benchmark Performance

- Number of logged events to total number of messages

<table>
<thead>
<tr>
<th>#processors</th>
<th>BT</th>
<th>SP</th>
<th>PT</th>
<th>CG</th>
<th>MG</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<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>
<td>27.10</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
### 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

### PCG Performance on AMD Opteron Cluster

<table>
<thead>
<tr>
<th>Size of the Problem</th>
<th>Num. of Comp. Proc</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>

### Checkpoint overhead in seconds

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

\[ \begin{align*}
C &= C + A \times B
\end{align*} \]

- The algorithm maintain the consistency of the checkpoints of the matrix C naturally

<table>
<thead>
<tr>
<th>PDGEMM-SUMMA</th>
<th>ABFT-PDGEMM-SUMMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{2n^2}{F} )</td>
<td>( \frac{2n^2 + 2(n+3) \beta}{F} )</td>
</tr>
<tr>
<td>( 2n + 2(n+3) \beta )</td>
<td>( 2n + 2(n+3) \beta )</td>
</tr>
</tbody>
</table>

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.