Towards Exascale Resilience

Franck Cappello\textsuperscript{\textcopyright}, Al Geist\textsuperscript{\circledast}, Bill Gropp*, Sanjay Kale*, Bill Kramer*, Marc Snir*,
\textsuperscript{\textcopyright}INRIA
\textsuperscript{*}UIUC
\textsuperscript{\circledast}ORNL

06/18/2009

V 0.1
Other contributors are welcome
Background & Overview

- Faults every day in PetaScale systems:
  - Reboot once a week because of too many faults
- Existing OS checkpoint Restart:
  - BLCR
  - Libckpt
  - Condor stand alone ckpt library
- Existing fault tolerant MPI:
  - FT-MPI (UTK)
  - Open MPI (Consortium)
  - MPICH-V (INRIA)
  - LAM-MPI (Indiana)
  - MPVAPICH (Ohio)
  - LA-MPI (Los Alamos)
  - AMPI (UIUC)
- Not well understood:
  - Proactive actions
  - Silent errors
  - Hardware support
  - Software Layer coordination
  - Fault oblivious Algorithms

Exascale Systems: Millions of cores, Billions of threads

- Faults will be “normal” events
- Some faults will not be detected (silent errors)
- Uncorrectable transient memory errors may represent a large fraction of errors
- Long running jobs may be hit by hardware & software faults several times before completion
- Currently in most cases simulations are killed when any fault occurs
- Checkpoint-Restart on remote file system seems to reach a dead-end

⇒ Reliability and Robustness research is critical at every layer
⇒ SW resilience may be more important than HW
⇒ Need to cope with continuous stream of faults
⇒ New paradigms are required for handling faults and transient errors
⇒ New paradigms for new and existing applications

2009/05/21
The community

- Computer Scientists with main field in fault tolerance and resilience

- Computer Scientists with main field in other domains but with some experience or specific knowledge on fault tolerance and resilience (people working for programming models, file systems, libraries, etc.)

- Application developers

- Experts in HPC centers, system managers

- Users of large scale long running applications

- Vendors (software and hardware)

- The community is spread over the world but not organized at the international level. **Team’s skills are complementary:** not any single team contains all the required skills
Environment to develop Exascale Resilience

• Agreeing on definitions and types of faults, errors (silent or not) and failures in HPC (need for characterization):
  ▪ --> Ex1: definition acceptance of “soft errors” (elusive development faults and of transient physical faults)
  ▪ --> Ex2: transient errors in HPC systems:
    ▪ who distinguish between transient and permanent errors? --> system managers? If so how does one recognize that an error is transient? → automatic, software based detection?

• Agreeing on metrics (rate of progress?, resistance factors?, reaction time?, etc.)

• Establishing a central information repository for beginners and experts (WIKI)

• Defining a “standard” format for failure log (+standard error encoding) in collaboration with data mining experts AND ensuring consistence between human entered event logging and automated information

• Collecting, gathering and sharing anonymized failure logs

• Experimental environment for testing Res. approaches
1) Errors detection and understanding
   • Fault detection and layer coordination
     • More software and hardware detectors to reduce the occurrence of undetected errors (silent errors).
     • Higher error detection resolution to determine initial/likely SW component failure
     • Layers coordination in fault detection and management (ex: consistent fault tolerance between the applications and numerical libraries), including propagating errors conditions through SW layers (and HW components)

   • Understanding faults and silent errors
     • Understanding failures root causes
     • Determine commonalities across SW errors (large portion of SW errors seems to come from SW that is in error recovery or exception handling)
     • Understanding Impact of faults (HPC usage scenarios, applications needs, fundamental origin of the algorithm sensibility to faults) --> some systems spend so much time in error recovery that performance becomes unpredictable
     • Silent errors (understanding sources, consequences, detection of their effects, quantification, etc.)
1) Errors detection and understanding (cont.)
   • Situational awareness
     • Make sure that even if errors are automatically corrected (ex: RAID) they are reported to the system managers in order to allow them taking proactive actions (power plants, aircraft, etc spend a lot of time studying situational awareness so that when automated systems fail, humans can step in and make effective judgments.)

2) Errors Recovery
   • Non masking approaches
     • Application level fault tolerance (checkpoint-restart libraries, FT-MPI)
     • Algorithm Based Fault Tolerance (on-line ABFT)
     • Checkpoint size reduction (programmer assisted variable checkpointing)
     • Novel approaches?
2) Errors Recovery (cont.)

- **Masking approaches**
  - Rollback Recovery
    - Diskless checkpointing
    - Message logging protocols (MPI, PGAS, distributed objects, etc.)
    - Ckpt size reduction (incremental ckpt, compiler dead variable detection)
  - Proactive actions
    - Error prediction (notification from sensors, error rates, heuristics, etc.)
    - Understanding the benefit of pro-actions (50% of faults are from software)
  - Replication
  - Novel approaches?

- **Specific approach**
  - Based on error characterization and impact analysis (ex: BG/L L1 cache error)
    - Fault containment and determination of minimal response

- **Hardware support** (help accelerate expensive fault tolerance/resilience mechanisms, help software by providing more information or more stability)
  - SSD devices, Specific networks
Research Topics 4/4:

3) Error-tolerant algorithms
   • Techniques for fault oblivious simulations (simulation algorithms being indifferent to fault or capable to adapt themselves to failures)

4) Error prevention and handling facilitation
   • End to end correctness for data (like ANSI T10-DIF)
   • Better development methods to make code more robust (robust journaling capability?)
   • Making more SW errors transient

5) Stressing proposed solutions
   • Improve fault testing (Current ideas of black box testing - perturb a dimension of a system and see how it fails for example - is harder to do at large scale)
   • Investigate gains and losses of using High Availability versions of the OS and SW components
   • Experimental environment (test and compare fault tolerant / resilience approaches)
Collaboration Scenarios

1. Almost no collaboration
   - Joint workshops are held
   - WIKI for beginners and experts
   - Book on FT/Resilience
   - ....

2. Loosely coupled collaboration
   - Stress test are defined
   - ...

3. Collaboration and Standardization (in parallel)
   - Format for error/event logging
   - Stress tests and evaluation metrics
   - Interface for layers interoperation in fault detection and management
   - Multi-languages resilient runtime environments
   - ...

4. Tightly Coupled collaboration
   - Developing joint researches on proprietary topics with funding support
   - ...

Establish priorities
## Environment Milestones (example)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitions (fault, errors, failures, etc.)</td>
<td>V 1.0 For SC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIKI</td>
<td>V0 for SC</td>
<td>WIKI 1.0 for SC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition of a standard format for event logging</td>
<td>V0 for SC</td>
<td>V1 for SC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collect event traces for large scale machines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Establishing the community</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition of environment features + definition of stress and metrics (SC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design of the environment + preparatory tests (presentation at SC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sart of the development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First prototype available for SC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First results of large scale stress tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2009/05/21
## Milestones on research topics (example)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware support</strong></td>
<td>Collect SSD device roadmaps</td>
<td>First design of a SSD based resilience tech.</td>
<td>Test of SSD for resilience on Tsubame</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Diskless Checkpoint</strong></td>
<td>State of the art (coding., hardware, memory footprint, etc.)</td>
<td>A diskless checkpointing library</td>
<td>Large scale test on real applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Diskless checkpoint (2)</strong></td>
<td>Feasibility &amp; usefulness of a specific hardware operator</td>
<td>If interesting: Design of a hardware system for diskless ckpt.</td>
<td>Large scale test on real applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proactive actions</strong></td>
<td>Establish collaboration with HPC centers</td>
<td>Test efficiency of prediction algorithms</td>
<td>Design new approaches with data mining teams</td>
<td>First results of failure prediction on large scale computers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Silent errors</strong></td>
<td>Establish the community</td>
<td>Categorization of silent errors</td>
<td>Quantification of Soft errors</td>
<td>Feasibility of Silent error detectors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fault oblivious algorithms</strong></td>
<td>Establish the community</td>
<td>Fundamental sensibility of current algo.</td>
<td>Establish a list of possible approaches</td>
<td>Prototyping for selected key applications</td>
<td>First results demonstrated at SC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2009/05/21
### Milestones on research topics (example)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination across layers</td>
<td>Definition of an event bus</td>
<td>Definition of an event bus</td>
<td>First pieces of software using the EB API</td>
<td>Result of tests on the first pieces of soft.</td>
<td>Improvement of the EB API</td>
<td>Full stack of software using EB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABFT</td>
<td>Establish the community</td>
<td>Feasibility of on-line ABFT for Linear Algebra</td>
<td>Feasibility of on-line ABFT for other num. kernels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduction of the checkpoint size (memory exclusion)</td>
<td>Establish the community</td>
<td>Incremental checkpointing</td>
<td>Programmer / compiler guided checkpointer</td>
<td>Hybrid &amp; alternative approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replication</td>
<td>Establish the community</td>
<td>Investigate cost of active and passive replication</td>
<td>Investigate the feasibility of hybrids (replicat.+ckpt)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2009/05/21