



BERKELEY LAB

LAWRENCE BERKELEY NATIONAL LABORATORY



U.S. DEPARTMENT OF
ENERGY

System-level Checkpoint/Restart with BLCR

Paul H. Hargrove

Work with Eric Roman and Jason Duell

checkpoint@lbl.gov

<http://ftg.lbl.gov/checkpoint>



Introduction

BLCR is Berkeley Lab Checkpoint/Restart for Linux.

Project goals. What is BLCR's approach to CR?
Why use checkpoint/restart?

System design. How does BLCR work?

Current status. What does BLCR do now?

Plans. Where is BLCR going?

Disclaimers (what this talk is *not* about)

- **I do not work in NERSC operations**
 - Can't accurately describe faults seen at NERSC
 - Can't comment on NERSC timeline to deploy BLCR software
- **I do not work for Cray, nor do I ask Cray questions when I might not be able to repeat the answers**
 - Can't provide any details of BLCR integration w/ Cray's MPI
 - Can't provide any details of BLCR integration w/ Cray's aprun
 - Honestly don't know details of Cray release schedules
- **I work on roll-back recovery, but am not a zealot**
 - I do not think system-level checkpoint is the *only* or *best* solution

Project Goals

- **Provide checkpoint/restart for Linux clusters running scientific workloads.**
 - Checkpoint and restart jobs (shell scripts) running MPI applications.
 - Support a wide variety of networks.
- **Fit easily into production systems.**
 - Run unmodified application source.
 - Run unmodified binaries where possible.
 - No special compile/link in most cases.
 - Run on unpatched kernels (as a Linux kernel module).
 - Run with unmodified system libraries (e.g. libc).
- **Unrelated features have low implementation priority.**
 - ptrace, Unix domain sockets, etc.
- **Why would our users checkpoint?**
 - We see three main scenarios: scheduling, fault tolerance and debugging.

Usage Scenarios

- **Batch Scheduling**

- C/R can be used to pre-empt and/or migrate running jobs.

- Drain queues quickly for maintenance.

- Increase system throughput by switching job mix between long jobs and wide jobs.

- Increase system utilization by allowing the scheduler to correct for bad decisions.

- Gang scheduling. Divide system time up into slots.

- Priority scheduling. Run jobs with the highest priority.

- **Fault Tolerance**

- Not every application can checkpoint itself.

- Periodic checkpoints can reduce lost work in case of failure (but adds cost to normal fault-free execution).

- Proactive checkpoints can respond to non-yet-fatal problems (like loss of a fan).

- **Debugging**

- Rollback execution to a checkpoint taken before a fault, restart with a debugger.

- “Forks” or “Branchpoints” to (re)run with multiple inputs.

Implementation Approach

- **BLCR provides single node checkpoint/restart through kernel modules and a runtime library.**
 - Full or Stub* run library:
 - Full: can register callbacks, request checkpoints, etc.
 - Stub: tiny library with only a default checkpoint handler
 - Kernel module: coordinates the process checkpoints, saves/restores kernel data structures, interfaces with library and command line tools.
- **BLCR does *not* provide built-in support for distributed runtime features**
 - TCP sockets, bproc namespaces, etc.
- **Instead, BLCR provides ability to register “callbacks”**
 - Apps and/or libs must coordinate checkpoint/restart of distributed processes
 - So, the MPI library must know how to checkpoint;
but the user application does not.

Basic Operation of a Checkpoint Request

- **Rough idea**

- Send a signal that gets handled in BLCR library.

- May come from the same process, or from another.

- The signal handler “coordinates” and then calls the kernel code.

- **Refinements**

- By default, user code doesn't need to do anything to handle it.

- If desired, user code may register a callback to handle it.

- If desired, user code may block requests (critical sections).

Status Highlights

- **Processes, process groups and sessions**

- Shell scripts (bash, tcsh, python, perl, ruby, ...).

- Multithreaded processes (pthreads with standard NPTL).

- Resources shared between processes are restored.

- Restore PID and parent PID.

- **Files**

- Reopen files during restart: open, truncate, and seek.

- Pipes and named FIFOs.

- Files may exist in same location in the filesystem,
or may use option for file path relocation.

- Memory mapped files are remapped
or may use option to save in the context file
(e.g. to migrate shared libraries and executable).

Supported Platforms and Software

- **Linux kernel 2.6 kernels**

Support of custom patched kernels through autoconf (detect features not versions)

Test with kernels from kernel.org, Fedora, SuSE, Debian, etc.

Cray's CNL tested by Cray

ZeptoOS for BG/P ("link test" only)

- **MPI Implementations**

MVAPICH2

LAM/MPI 7.x (sockets and GM)

MPICH-V 1.0.x with sockets

OpenMPI 1.3 (most networks)

Cray Portals

- **Architectures**

x86/x86-64, ppc/ppc64 & ARM

Xen dom0 and domU

- **Batch Queue Systems**

TORQUE support

available in recent releases.

qhold, qrls, and periodic checkpoints tested.

BLCR with Condor and Parrot

HOWTO available.

SGE

Old (serial) HOWTO available

User Communities

- **HPC (our target audience)**
Serial and MPI jobs
- **Task-farm and cycle-stealing**
Using Condor + Parrot
- **Military applications (for FT and Branchpoints)**
US Army TRADOC Analysis Center
Australian Defence Organisation
Ported BLCR to the ARM embedded processor

Parting remarks

- **Near-term work items (desired for SC09 release)**
 - In-kernel compression of checkpoints
 - Incremental/differential checkpoints
 - App/lib/compiler directed memory exclusion support
- **Future Work**
 - MPI support for partial/live migration anticipated
 - More MPI implementations
 - MPICH-2 support anticipated
 - Hope to encourage vendor support (e.g. Quadrics, Intel, LSF)?
 - Interested in other queue systems (LSF, SGE, SLURM, etc.)
 - Ship support with distributions (ROCKS, OSCAR)?
 - In-memory checkpointing for MPI (like Charm++)?
- **We expect BLCR to be deployed in a production batch environment before the end of the calendar year.**
- **You should be able to install BLCR on your system and checkpoint serial and MPI applications with it.**

For More Information

- <http://ftg.lbl.gov/checkpoint>
- **Papers (available from website):**
 - “Design and Implementation of BLCR”: high-level system design, including description of user API
 - “Requirements for Linux Checkpoint/Restart”: exhaustive list of Unix features we expect to support (or not).
 - “A Survey of Checkpoint/Restart Implementations”: focusing on open source versions that run on Linux
 - “The LAM/MPI Checkpoint/Restart Framework: System-Initiated Checkpointing”: implementation with LAM/MPI
- **CIFTS**
Coordinated Infrastructure for Fault Tolerant Systems
Parent project. Building a notification infrastructure for fault info.
<http://www.mcs.anl.gov/research/cifts/>